AI, AUTOMATION AND ANALYTICS

Transforming Device Management with Intune

AUTHOR: JANNIK REINHARD

About the book

Device management has a long history and continues to evolve with new innovations. Today, we have a significant interest in and development of artificial intelligence (AI). However, it’s important to remember that AI relies heavily on data to unlock its full potential. Without data, AI cannot offer much value.

The purpose of this book is to help you understand what AI is, explore its potential, and guide you on how to access and harness the data you need to build effective analytics and automation solutions using Microsoft Intune and the ecosystem around. By the end of this book, you'll have a deeper understanding of AI, automation, and analytics, and how these can transform device management in your organization to a new level.

This book includes numerous hands-on projects that you can replicate. These projects will also serve as inspiration for building your own custom solutions. We’ll cover a wide range of topics, from Intune and Microsoft Graph to Azure, providing you with a broad toolkit to create innovative solutions.

Additionally, various companies have supported the creation of this book. They will share their unique insights and experiences in the fields of analytics, automation, and AI, enriching your understanding of these topics from multiple perspectives.

This book does not require extensive prior knowledge, although a basic familiarity with Intune and Azure will be beneficial. Whether you're a beginner or an expert, you'll find value in the diverse topics and scenarios we cover. I hope you enjoy reading this book and discover new insights that will empower you in your journey of leveraging AI, automation, and analytics in device management.

Thanks to the sponsors

We understand the challenges IT teams face every day: late-night packaging sessions, deployment headaches, and constant pressure to do more with less. That's why we created a better way to manage endpoints.

Through the integration of Robopack and SoftwareCentral, we've turned endpoint management from a daily struggle into a smooth experience. Robopack automates packaging and patching tasks, while SoftwareCentral enhances both Intune and Configuration Manager, giving you the control you've always needed.

Our solutions grow with your organization, adapting to unique needs while maintaining simplicity. We believe IT professionals deserve tools that work as cleverly and reliably as they do. By automating routine tasks and simplifying complex ones, we free your team to focus on driving your organization forward.

Our solutions are all about giving IT teams back their time and making work more fulfilling. When technology works as it should, everyone benefits.

About the author

Jannik Reinhard, a 27-year-old German senior solution architect, began his career in 2014 with educational training as an IT system specialist at the world's largest chemical company. His career path in IT has been marked by continuous learning and progression through various roles within the company.

His first major project, which focused on implementing a cloud management solution for iOS smartphones in a large enterprise, introduced him to Microsoft Intune in its early stages. Specializing in modern device management, Jannik contributes to the transitioning of the company's standard workplace environment from on-premises management using Microsoft Configuration Manager to cloud management with Intune. This transition extended to managing special devices from the cloud, enhancing the

flexibility and scalability of device management across the organization.
Jannik also expanded his expertise into analytics, reporting, and AI, deepening his knowledge of Azure Cloud and other Microsoft technologies. He currently works as the technical lead for an initiative called AIOps (AI for IT Operations). This initiative focuses on improving end-user experiences by applying AI to proactively detect and resolve issues in the IT environment.

In recognition of his contributions, Jannik was awarded as a Microsoft MVP title for Enterprise Mobility in February 2023, and in 2024, his expertise was further recognized with a dual MVP award in Security - Intune and AI Platform - Azure AI Services. He actively participates in the largest LinkedIn and Discord Intune communities and manages one of the largest Twitter communities for Intune.
Beyond his professional responsibilities, IT is Jannik's passion. He dedicates significant time outside of work to learning and innovating, sharing his knowledge through his blog, jannikreinhard.com, and speaking at global events. His blog offers insights on modern device management and related topics, providing valuable resources for anyone looking to enhance their IT expertise.

Jannik is also a mentor to young professionals, guiding them in their careers and personal growth. He contributes as an auditor at a university and has collaborated on other books, demonstrating his commitment to fostering knowledge and learning in the IT community.

About the reviewers

Florian Salzmann

Florian Salzmann is a Microsoft MVP and consultant from Switzerland, specializing in Microsoft Intune, endpoint management and security. With over a decade of experience in IT since 2012, he enjoys tackling everyday challenges in the modern workplace, whether through PowerShell scripting or making the most of cloud tools.

On his blog, scloud.work, Florian shares practical tips, solutions, and ideas to help IT pros navigate and improve their day-to-day work. The blog’s tagline, "Mastering the Modern Workplace with Intune," reflects his focus on keeping things simple and effective for IT teams.

When he’s not blogging or scripting, Florian speaks at conferences, travels the world in his free time, or spends a little too much time tinkering with his smart home setup.

Sune Thomsen

Sune Thomsen is a Windows and Windows 365 MVP based in Denmark with over 19 years of experience in the IT industry. He has spent at least a decade specializing in client management via Microsoft Configuration Manager and Intune, and he's currently helping large enterprise customers build solutions toward the Cloud. Sune works as a consultant for a consulting company called Mindcore. Prior to joining Mindcore, Sune gained 10 years of experience in the engineering industry, managing and deploying various Microsoft solutions and projects. He's passionate about community work. Besides blogging and speaking at tech events, he's also an official contributor within the Windows 365 community and the Modern Endpoint Management LinkedIn group.

First, I'd like to thank Jannik for giving me the opportunity to review the book. It has been a great honor and an educational journey to be part of! Last but not least, I want to take a moment to express my deepest gratitude to my lovely family (Annie, Carl, and Lucas). Your support and understanding have allowed me to dedicate significant time to the community. I am truly blessed to have you by my side. With all my love, Sune.

Nicklas Ahlberg

Nicklas is a dual MVP (Intune and Windows) and work as a Trusted Security Advisor at Onevinn, a leading corporate specializing in providing cutting-edge Microsoft security solutions. His primary objective is assisting organisations in navigating the terrain of Intune, ensuring they obtain an optimal and highly secure user experience. He specializes in products such as Microsoft Intune, Windows 11, and Microsoft Defender XDR.

AI, Automation and Analytics: Transforming Device Management with Intune

Published by: Jannik Reinhard, 76855 Annweiler, Germany

First published date: 28.01.2025

Foreword

Lavanya Lakshman (Principal PM Manager (Copilot in Intune)

My heartfelt congratulations to Jannik on this remarkable milestone! I am truly honored to have been given the unique opportunity to write the foreword for his debut book.

I have had the pleasure of knowing Jannik since 2022. From our very first conversation, it was evident that he possessed a unique blend of passion and expertise in modern device management and AI operations. Over the years, I have witnessed his dedication and profound understanding of these fields, making him uniquely qualified to write this book.

Jannik has spent the last several years researching and working in IT, specifically in the areas of modern device management and AI operations. His extensive experience includes serving as a senior solution architect and technical lead of AIOPS at the largest chemical company in the world. Additionally, he is a dual Microsoft MVP in Security-Intune and AI Platform-Azure AI Services, which significantly contributes to advancements in these areas. This book, "**AI, Automation and Analytics - Transforming Device Management with Intune,**" offers a comprehensive guide to modern device management, filled with practical advice and real-world examples that will benefit anyone interested in this subject.

One of the most exciting aspects of this book is its focus on the breakthroughs in Copilot and AI. The author delves into how these technologies are reshaping the landscape of device management. The integration of AI-driven Copilot into Intune is a game-changer, providing unprecedented levels of automation and intelligence. This advancement not only streamlines routine tasks but also enhances decision-making processes, making device management more efficient and secure.

The book transitions smoothly from discussing the fundamentals of going Cloud Native with Intune to exploring the seamless migration into the AI world with Copilot. The author expertly guides the reader through the process of leveraging Intune's capabilities to manage devices in a cloud-native environment and then seamlessly integrates AI to further enhance these capabilities. This progression is designed to help readers understand the full potential of combining cloud-native solutions with advanced AI technologies.

Additionally, the book covers the topic of device queries at length, providing readers with a comprehensive understanding of how to effectively query and manage devices. It also includes a quick tutorial on Kusto Query Language (KQL), which is invaluable for anyone looking to harness the power of data analytics within Intune. This tutorial offers practical examples and step-by-step instructions, making it accessible even for those new to KQL.

The author's creativity shines through as he details custom solutions for building custom analytics and automation solutions with Intune. He not only covers the product capabilities but also brings in his own innovative approaches to solving complex problems. His insights into creating tailored solutions for specific organizational needs are particularly valuable for IT professionals looking to optimize their device management strategies.

Moreover, the book addresses the reality of rapid developments and the adoption of Copilots. It provides a detailed account of how these advancements are being quickly integrated into various products and services, reflecting the fast-paced nature of technological innovation in the field. This perspective is crucial for understanding the current and future landscape of device management.

I remember a time when the author and I worked together at the MMS 2024 conference during the session "The Dream Team of Intune and AI". It was an enlightening experience where we explored the integration of AI with Intune to enhance device management. His insights and innovative approaches were instrumental in demonstrating the potential of AI in transforming device management practices. His ability to convey complex ideas in an accessible manner is truly remarkable.

The author's journey in the IT industry is nothing short of inspiring. His early career was marked by a relentless pursuit of knowledge and a passion for technology. He quickly rose through the ranks, earning recognition for his innovative solutions and leadership skills. His role as a senior solution architect and technical lead of AIOPS at the largest chemical company in the world is a testament to his expertise and dedication.

Throughout his career, he has been a strong advocate for the adoption of modern technologies in device management. He has consistently pushed the boundaries of what is possible, leveraging AI and automation to create more efficient and secure systems. His work has not only benefited his organization but has also had a significant impact on the broader IT community.

His contributions to the field have been recognized through numerous awards and accolades. As a dual Microsoft MVP in Security-Intune and AI Platform-Azure AI Services, he has been at the forefront of technological advancements, sharing his knowledge and expertise with others. His commitment to continuous learning and professional development is evident in his extensive research and hands-on experience.

In this book, he shares his wealth of knowledge and experience, providing readers with a comprehensive guide to modern device management. He covers a wide range of topics, from the basics of going Cloud Native with Intune to the advanced integration of AI-driven Copilot. Each chapter is filled with practical advice, real-world examples, and step-by-step instructions, making it an invaluable resource for IT professionals at all levels.

One of the standout features of this book is its focus on the practical application of AI and automation in device management. The author provides detailed insights into how these technologies can be used to streamline processes, enhance security, and improve decision-making. His innovative approaches to solving complex problems are particularly valuable for those looking to optimize their device management strategies.

He also addresses the challenges and opportunities presented by rapid technological advancements. The book provides a detailed account of how Copilots are being integrated into various products and services, reflecting the fast-paced nature of innovation in the field. This perspective is crucial for understanding the current and future landscape of device management.

In addition to his technical expertise, his ability to communicate complex ideas in an accessible manner is one of his greatest strengths. His writing is clear, concise, and engaging, making this book a pleasure to read. Whether you are a seasoned IT professional or new to the field, you will find valuable insights and practical advice that you can apply in your own work.

I wholeheartedly recommend this book to anyone looking to deepen their understanding of device management with Intune. It is a must-read that will leave a lasting impression. The author's passion, expertise, and

dedication shine through on every page, making this book an invaluable resource for anyone interested in modern device management and AI operations.

Table of Contents

Chapter 1: The Evolution of Device Management and the Rise of AI

Introduction

Artificial Intelligence (AI) is a buzzword that's hard to miss in today's world. From personal gadgets to enterprise systems, AI is becoming an integral part of our lives. However, AI is more than just a passing trend; it's a transformative force poised to redefine the way we work, live, and interact with technology. In both personal and professional life, AI is helping us distinguish between tasks that require human effort and those that can be handled by machines. Beyond automation, AI offers valuable insights, streamlines processes, and enhances decision-making.

In the realm of IT, particularly device management, AI is set to be a game-changer. For instance, Microsoft's AI-powered Copilots are already in everyone's attention, deeply integrating into various products and significantly enhancing productivity. These Copilots are reshaping how we interact with applications, operating systems, and even the internet. Imagine the evolution of search engines: where once you sifted through countless Google search results to find the perfect recipe for baking a cake, AI-driven Copilots like Bing now provide you with a perfect summary of the top results. This advancement eliminates the need to visit multiple pages, offering a streamlined experience that saves time and effort.

This chapter will explore the evolution of device management, the impact of AI, and how the convergence of automation, analytics, and AI within Microsoft Intune is setting the stage for a future of more efficient and secure IT management. We'll begin with a historical perspective on device management, tracing the journey from manual setups to cloud-based solutions like Intune, and then look forward to the integration of AI and advanced analytics.

The Early Days: Manual Setup and Configuration

The story of device management begins with manual setups—an era when IT departments had to individually configure each device within an organization individually. This process was both labor-intensive and time-consuming. Every device required the installation of operating systems, software applications, and personalized configurations tailored to individual user needs. The lack of automation meant that IT teams were frequently overwhelmed, especially as companies expanded and technology advanced. Scalability was a constant challenge, with manual processes proving to be insufficient as the number of devices grew.

In those days, IT administrators were often bogged down by repetitive tasks, leaving little room for innovation or strategic planning. This environment created a pressing need for more efficient management tools—solutions that could handle the growing complexity of IT infrastructure without requiring a proportional increase in manual effort.

The manual setup era also posed security risks, as individual device configurations often led to inconsistencies and potential vulnerabilities. Without centralized oversight, ensuring compliance across the board was difficult, leading to gaps that malicious actors could exploit. As organizations grew, these inefficiencies and risks underscored the urgency of developing a more streamlined, secure approach to device management.

Active Directory: A Paradigm Shift

A significant leap forward in device management came with the introduction of Microsoft's Active Directory (AD) in 1999, alongside Windows 2000. AD revolutionized IT by centralizing user and device management, providing a way to apply group policies and manage access to network resources from a single interface. This innovation reduced the workload on IT departments by automating the application of security settings and access controls across multiple devices.

Active Directory also enabled devices to join a network more seamlessly, applying group policies that standardized configurations across the organization. This shift from decentralized, manual configurations to a centralized management model brought about greater consistency and security. With AD, IT administrators could more effectively manage permissions, implement password policies, and ensure devices were compliant with organizational standards.

However, despite its improvements, AD still required physical network connections and couldn’t fully automate the management process. The need for constant updates and maintenance continued to pose challenges, particularly for growing organizations with distributed workforces. As companies began to expand globally, AD's dependency on on-premises infrastructure started to limit its ability to adapt to the increasing mobility of users and devices. This created a new demand for solutions that could support a more flexible, modern workplace.

From SMS to MDT: Early Steps Toward Centralized Management

Before the advent of modern cloud-based solutions, Microsoft laid the groundwork for centralized device management through a series of tools and frameworks that gradually evolved over time.

- **Microsoft Systems Management Server (SMS):** Released in 1994, SMS 1.0 was Microsoft’s first major foray into enterprise device management. SMS introduced the concept of centralized control over software distribution, inventory collection, and patch management. Though rudimentary by today’s standards, SMS allowed IT administrators to gain visibility into their hardware and software assets, reducing the complexity of manual updates on a large scale. Over time, SMS would evolve into a more robust solution, eventually becoming the foundation for what we know today as System Center Configuration Manager (SCCM).

- **Microsoft Deployment Toolkit (MDT):** Initially released in August 2003 as Business Desktop

Deployment (BDD), this toolkit provided a structured framework for automating the deployment of Windows operating systems and applications. MDT simplified the imaging process and reduced inconsistencies in configurations. By consolidating best practices for deployment, MDT helped organizations streamline their processes, ensuring that each device was set up according to standardized guidelines. MDT played a critical role in transitioning from laborious manual setups to more predictable and efficient deployments.

These early tools—SMS and MDT—highlight Microsoft's ongoing efforts to reduce the complexity and cost of managing large fleets of devices. Although these solutions were not as comprehensive or flexible as modern cloud-based services, they introduced key concepts, such as centralized management and standardized deployments, that would underpin future innovations.

System Center Configuration Manager (SCCM)

Building on the advancements of SMS and the deployment best practices introduced by MDT, Microsoft introduced System Center Configuration Manager, which represented a significant evolution in enterprise device management. Originally known as Systems Management Server, the product underwent continuous refinement, culminating in the release of SCCM under its new name in 2007.

SCCM offered IT administrators a holistic solution for managing devices, applications, and security policies across an organization's entire IT infrastructure. With SCCM, administrators could remotely package and deploy software, enforce security policies, and perform system updates—all from a central console. This capability was a substantial step forward, enabling organizations to manage their IT environments at scale.

SCCM also introduced features like inventory management, allowing IT teams to track hardware and software assets across the organization. This visibility helped administrators make informed decisions about updates, compliance, and resource allocation. Moreover, the ability to deploy patches and updates remotely reduced the need for IT staff to be physically present at each device, saving both time and operational costs.

However, the adoption of SCCM came with its own set of challenges. The infrastructure required to support SCCM was often complex and resource-intensive, especially for large, globally distributed organizations. On-premises servers needed regular maintenance and upgrades, and the system's reliance on network connectivity limited its flexibility in supporting increasingly mobile workforces. Additionally, configuring and managing SCCM itself required specialized knowledge, often necessitating dedicated staff just to handle its operations.

Despite these limitations, SCCM represented a considerable improvement over manual processes and earlier tools like SMS. Its continuous evolution and integration of hybrid management scenarios—combining on-premises and cloud-based approaches—have kept SCCM relevant in modern IT landscapes. The progression from SMS to MDT and eventually SCCM underscores a broader trend in IT: the constant push for efficiency, scalability, and security. Each step forward addressed specific pain points of the previous era, paving the way for increasingly sophisticated tools that would ultimately transform how IT departments operate.

The Era of Mobility and Remote Work

As the workforce became more mobile, with employees increasingly relying on smartphones, tablets, and laptops to stay connected, the limitations of traditional device management approaches became more apparent. The reliance on constant network connections—typically via VPN—was cumbersome and impractical, particularly for remote workers. This shift in work dynamics called for a more flexible and scalable approach to device management, one that could support a wide variety of devices and work environments.

Remote work trends highlighted the need for device management solutions that could operate seamlessly across different locations without relying on on-premises infrastructure. The growing demand for Bring Your Own Device (BYOD) policies further complicated the management landscape, as IT departments had to find ways to secure and manage personal devices alongside company-owned hardware. These challenges underscored the need for a cloud-based, scalable solution that could meet the demands of a modern, mobile workforce.

COVID-19 Pandemic: Accelerating the Shift

The global COVID-19 pandemic acted as a catalyst, accelerating the shift toward remote work and exposing the inadequacies of traditional device management strategies. Virtually overnight, organizations had to adapt to a new normal where employees were working from home, often on personal devices. For those relying on on-premises solutions like SCCM, this sudden change created significant challenges. IT teams found themselves struggling to manage devices remotely, maintain security protocols, and ensure productivity in a rapidly changing environment.

This period highlighted the need for a more agile, scalable, and secure approach to device management—one that could support remote workforces and reduce reliance on physical infrastructure. Organizations that had already begun adopting cloud-based solutions were better positioned to navigate this transition, while those still reliant on on-premises systems were forced to accelerate their digital transformations.

Cloud Native Device Management with Intune

Enter Microsoft Intune. Introduced as a cloud-based service, Intune represented a significant shift in the way organizations approached device management. Unlike traditional on-premises solutions, Intune offered out-of-the-box global infrastructure scalability, eliminating the need for extensive hardware investments and ongoing maintenance. By leveraging the power of the cloud, Intune made it possible to manage devices remotely, no matter where they were located or how they connected to the network.

Intune facilitates a wide range of device management functions, including remote configuration, application deployment, security policy enforcement, and continuous compliance monitoring. With Intune, IT administrators can ensure that devices meet the organization's security requirements, regardless of whether they are company-owned or part of a BYOD policy. The cloud-based nature of Intune also allows for more rapid updates and innovation, enabling organizations to stay ahead of the curve in a constantly evolving technology landscape.
However, Intune is more than just a cloud-based management tool—it's a platform that enables organizations to automate and optimize their IT operations. As we'll explore in later chapters, Intune's integration with AI and advanced analytics is paving the way for a new era of device management, where routine tasks can be automated, and predictive analytics can preemptively address issues before they impact users.

Looking Ahead: AI and Advanced Analytics

As we look to the future of device management, AI and advanced analytics are emerging as transformative forces within Microsoft's ecosystem. These technologies have the potential to redefine how we approach efficiency, security, and decision-making in device management.

One of the most promising advancements is AI's role in supporting and automating configurations. By analyzing vast amounts of data from thousands of customers, AI can suggest optimal settings for devices based on specific organizational needs and user preferences. This capability not only streamlines the configuration process but also ensures that devices operate with the most effective balance of performance and security. AI can help organizations avoid the common pitfalls of manual configuration, such as inconsistencies and oversights, by applying best practices automatically.

In addition to configuration support, AI offers the ability to proactively address and solve issues before they impact users. Continuous monitoring of device health and performance allows AI to identify potential problems early. Whether through automated fixes or by alerting IT staff to take action, AI helps minimize downtime and improve the user experience. This shift from reactive to proactive management represents a significant step forward in device management, reducing the time and effort required to maintain optimal performance across an organization’s IT infrastructure.

Security, always a critical concern in device management, is also being transformed by AI and analytics. Advanced algorithms are capable of detecting and neutralizing security threats with unprecedented speed and accuracy. AI’s continuous learning capabilities allow it to adapt to evolving threats, safeguarding devices against both known and emerging risks. By integrating AI into security protocols, organizations can significantly enhance their security posture, reducing the likelihood of breaches and other security incidents.

Microsoft's extensive ecosystem, which includes Entra ID (formerly Azure Active Directory) and Intune, offers unique opportunities for leveraging AI and analytics in device management. The rich network of devices and services within this ecosystem generates a wealth of data, and analyzing this data provides valuable insights that drive innovation and inform strategic decisions. These insights are not only useful for optimizing device management practices but also for predicting future trends and challenges, allowing organizations to stay ahead of the curve.

Incorporating AI and advanced analytics into Intune marks a significant shift in the landscape of device management. The automation of lazy tasks, the power of predictive analytics, and the reinforcement of security—combined with the deep insights gained from Microsoft's broader ecosystem—position Intune at the forefront of the next evolution in device management. This transformation promises to make device management more intuitive, responsive, and secure, enabling organizations to focus on strategic initiatives rather than being bogged down by day-to-day IT operations.

As we continue through this book, we'll explore these advancements in greater detail, providing practical guidance on how to leverage AI, automation, and analytics to optimize device management with Microsoft Intune. Together, we'll navigate this exciting evolution, unlocking the full potential of modern device management.

Chapter 2: Understanding Microsoft Intune: The Heart of Modern Device Management

Introduction

Before we delve into the details of analytics, automation, and AI within Microsoft Intune, it's essential to first understand what Microsoft Intune is and its pivotal role in the modern IT management landscape. Intune is more than just another tool in the enterprise mobility sector and security space; it is the leader in Unified Endpoint Management (UEM) and is Microsoft's approach to managing devices, applications, and security in today's dynamic and increasingly mobile work environments.

In this chapter, we will explore what Microsoft Intune is, its core functionalities, and its significance in managing modern devices. We will also examine how Intune integrates with other key Microsoft services to offer a comprehensive, unified approach to device management. This foundational understanding will prepare you for the deeper dive into analytics, automation, and AI that follows in subsequent chapters.

What is Microsoft Intune?

Microsoft Intune is a cloud-based service that falls under the category of Software as a Service (SaaS). It provides organizations with powerful tools for managing the devices their employees use to access corporate data. Intune operates within the device management and security space, allowing organizations to manage both personal devices (in a bring-your-own-device, or BYOD, scenario) and company-owned devices.

Intune supports a wide range of devices, including Windows, macOS, iOS/iPadOS, Android, and even some Linux platforms. It also manages specialized devices, such as those running Windows for HoloLens, Teams Room devices, or Surface Hub. This extensive support allows organizations to have a unified device management strategy across different types of devices, ensuring consistent security and compliance.

Intune allows IT administrators to control and manage how devices are used within the organization. This includes configuring device settings, enforcing security policies, and managing applications. Its cloud-based nature enables IT departments to manage devices regardless of their location, which is critical in today’s world, where employees expect the flexibility to work from anywhere.

Intune also helps ensure compliance with organizational policies and industry regulations. With tools for monitoring device health, security settings, and applications, Intune assists IT administrators in maintaining compliance and generating reports when necessary.

Additionally, Intune's integration with Entra ID allows for the use of conditional access policies, which are crucial for enforcing compliance. Conditional access lets IT administrators create automated policies that control access to corporate resources based on conditions like

user identity, device compliance status, and network location. This ensures that only authorized users can access sensitive data and applications, further improving security and compliance.
If you want to learn more about the basics of Intune, you can check out the reactor series from Nicklas Olsen and me. You can find it here: https://learn.microsoft.com/de-de/shows/reactor/

Core Features of Microsoft Intune

One of Intune's key features is its ability to ensure that all managed devices meet an organization's security requirements. In an era of increasing mobile computing and security risks, this functionality is critical. Intune provides IT administrators with tools to enforce security policies, such as requiring devices to be secured with a PIN or password, enforcing encryption of sensitive data, and remotely wiping data from lost or stolen devices to prevent unauthorized access.

Intune also offers a rich set of reporting and monitoring tools that allow IT administrators to track device compliance, application installations, and security status across the entire device fleet. These tools provide valuable insights that help IT teams identify and address potential issues. Intune makes it easier for organizations to demonstrate the compliance of each managed device.

In addition to this, Intune offers a wide range of application management features. IT administrators can use Intune to install, update, and manage applications on a variety of devices. Beyond deployment, Intune provides

Mobile Application Management (MAM) features for fine-grained control over how corporate data is used within applications. For instance, Intune can restrict actions like copying and pasting between company-managed apps and personal apps, adding a layer of data protection to prevent sensitive information from being leaked or shared without authorization.

Intune also integrates with other Microsoft services to enhance its functionality. For example, it integrates with Entra ID (formerly Azure Active Directory) for identity and access management, which is the foundation of device management. This integration enables user and group management and enhances security, e.g., by enabling multi-factor authentication and conditional access policies.

Intune works with Microsoft Configuration Manager (formerly SCCM), offering additional capabilities for managing PCs, servers, and other devices in hybrid environments where both cloud and on-premises management are needed. This flexibility allows organizations to transition to the cloud at their own pace while maintaining control over legacy systems and benefiting from the scalability and automation that cloud-based management provides.

Additionally, Intune integrates with Microsoft Defender for Endpoint to provide advanced threat protection for managed devices. By leveraging Microsoft's threat intelligence, Intune helps identify, prevent, and respond to security incidents, further enhancing an organization's overall security posture.

These integrations make Intune a powerful suite for managing the full spectrum from identity management to security and compliance, all within a single pane of glass.

While the integrations mentioned are some of the most relevant, Intune supports many more integrations across the Microsoft ecosystem and third-party services.

Another important feature of Intune is its ability to provide remote support for end users. IT administrators can use Intune to troubleshoot and resolve issues on devices without requiring users to bring the device into the office. This remote support capability is crucial in modern work environments where employees may be working from home or in different locations.

Enhancing Application Management with Intune

In the realm of application management, Intune offers a robust and flexible toolset. IT administrators can deploy applications to a variety of devices, ensuring that users have access to the tools they need, no matter where they are working from. Intune allows administrators to automate the installation and updating of applications, ensuring that devices are always equipped with the latest versions of important software. This automation not only saves time but also reduces the risk of vulnerabilities caused by outdated software.

Beyond deployment, Intune provides control over how applications interact with corporate data. IT administrators can create policies that dictate which apps can access corporate data, set rules for data sharing between apps, and prevent unauthorized actions like copying data from a corporate app to a personal app. These policies add an extra layer of security, helping organizations protect sensitive information while allowing users to work flexibly and efficiently.

Intune’s application management capabilities are particularly valuable in BYOD environments, where employees use their personal devices for work. With Intune, IT can ensure that corporate data remains secure on personal devices without infringing on the user’s privacy. Intune’s app protection policies allow IT to manage corporate apps and data on personal devices without requiring full device management. This approach protects corporate data even if the device itself remains under the user’s control.

Beyond Device Management: Intune as a Platform for Automation and Intelligence

As we continue through this book, we’ll explore how analytics and automation enhance the capabilities of Microsoft Intune. While Intune’s core functionality revolves around device and application management, its integration with automation tools and AI-driven analytics transforms it into much more than just a management tool—it becomes a platform for intelligent IT operations.

The rise of AI and advanced analytics within Microsoft’s ecosystem allows organizations to automate routine tasks, predict potential issues before they arise, and make more informed decisions based on data-driven insights. Intune plays a critical role in this evolution, acting as the foundation upon which organizations can build intelligent, automated workflows that reduce the burden on IT staff and enhance overall efficiency.

For example, Intune can be used in combination with Microsoft Power Automate or Logic Apps to create automated workflows that respond to specific triggers. This integration allows IT teams to automate repetitive tasks, such as notifying users of policy violations, initiating software updates, or provisioning new devices. By reducing manual intervention, these automated processes help to minimize errors, speed up response times, and free up IT resources for more strategic initiatives.

Intune also leverages Microsoft Graph, an API that provides access to data across Microsoft 365 services. By using Microsoft Graph, IT administrators can gain deep insights into device health, application usage, and user behavior. These insights can then be used to make data-driven decisions that improve device performance, enhance security, and optimize resource allocation. For example, administrators can identify devices that are frequently out of compliance and take proactive steps to address the underlying issues before they lead to security incidents.

From automating software updates and security patches to deploying AI-driven insights that optimize device performance, Intune provides endless features.

By leveraging Intune’s capabilities, organizations can move beyond traditional device management and embrace a future of proactive, intelligent operations that support business goals while keeping IT infrastructure secure and compliant.

In the chapters ahead, we'll dive deeper into these capabilities, showing you how to harness the full power of Microsoft Intune to transform your organization's device management strategy. Whether you're focused on improving security, increasing operational efficiency, or leveraging AI to gain new insights, Intune provides the tools and platform to help you achieve your goals.

Chapter 3: The Evolution of Microsoft Intune: From Early Beginnings to a Leading Device Management Platform

But let’s take a step back and explore the development of Microsoft Intune. The evolution of Microsoft Intune is a story of innovation, adaptation, and strategic foresight in a commodity business. From its early days as "Windows Intune" to becoming a player in the cloud-based device management space, Intune’s journey reflects Microsoft’s ability to anticipate and respond to the changing needs of modern IT environments. This chapter chronicles the major milestones in Intune's development, highlighting the key decisions and technological advancements that have shaped it into the essential tool for unified endpoint management it is today.

Early Beginnings: The Birth of Windows Intune (2011)

Microsoft Intune was first introduced to the world as "Windows Intune" in 2011. At that time, the concept of cloud-based device management was still relatively new, and organizations were primarily relying on on-premises solutions like Microsoft Configuration Manager (formerly SCCM) to manage their Windows environments. Windows Intune aimed to offer a cloud alternative, providing organizations with the ability to manage devices remotely without the need for a significant on-premises infrastructure.

However, in its early years, Intune’s adoption was limited. Microsoft Configuration Manager remained the dominant platform for Windows management, and many organizations were hesitant to make the leap to cloud-based solutions. Windows Intune’s initial feature set was also somewhat limited compared to Microsoft Configuration Manager, which made it difficult for enterprises to see it as a complete replacement for their existing systems. Despite these challenges, Microsoft remained committed to the vision of cloud-based management, and this persistence would soon pay off.

The Rebranding and Strategic Pivot: Intune Gains Momentum (2014)

In 2014, Microsoft rebranded Windows Intune as "Intune" and shifted its focus toward mobile device management (MDM). This strategic pivot was crucial as organizations began to grapple with the challenges of managing a growing number of mobile devices, including smartphones and tablets, alongside traditional PCs. The rise of Bring Your Own Device (BYOD) policies also highlighted the need for more flexible and scalable management solutions that could handle both personal and company-owned devices.

Intune’s ability to manage devices across multiple platforms, including Windows, iOS/iPadOS, Android, Windows Phone, and macOS, positioned it as a versatile tool for organizations looking to simplify their device management processes. Despite its early limitations, Intune's integration with Azure Active Directory (Azure AD) and its agentless architecture for BYOD management began to attract attention. Although it still lagged behind market leaders in some areas, Intune's appearance on the Gartner Magic Quadrant for enterprise mobility management in 2014 was a clear sign of its emerging potential.

The Rise of Intune: A Focus on Mobile Devices and Strategic Bundling (2017-2018)

By 2017-2018, Microsoft Intune had gained significant traction, particularly in the area of mobile device management. Organizations increasingly recognized the need to manage not only their PCs but also the growing fleet of mobile devices that employees were using for work. Microsoft capitalized on this trend by bundling Intune with its popular Microsoft 365 subscriptions, effectively making Intune available to many organizations as part of their existing software packages.

This bundling strategy proved to be a game-changer. Many organizations that were already using tools like Office 365 and Azure AD found Intune to be a natural extension of their IT management toolkit. The idea that Intune was "free" with Microsoft 365 subscriptions encouraged organizations to trial the service, and while early deployments often had their challenges—particularly for organizations still relying on Microsoft Configuration Manager for Windows device management—the stage was set for Intune's ascendance.

Microsoft's engineering teams in Seattle played a critical role in driving Intune’s development during this period. They introduced key features like Windows Autopilot, which automated the configuration and deployment of new Windows devices, and Windows Hello, which provided secure biometric authentication options. These innovations, combined with Intune's integration with Entra ID, Microsoft Defender, and some more, as mentioned above, elevate Intune from a challenger to a major player in the device management market.

COVID-19: Accelerating Intune’s Adoption and Growth (2020)

The COVID-19 pandemic in 2020 acted as an accelerant for the adoption of cloud-based device management solutions like Intune. As organizations around the world transitioned to remote work almost overnight, IT departments faced unprecedented challenges in managing and securing devices outside of the traditional corporate network. For many organizations still relying on on-premises solutions like Microsoft Configuration Manager, the pandemic exposed significant limitations in their ability to manage remote devices effectively.

Intune, with its cloud-based architecture, emerged as a lifeline for these organizations. Its ability to manage devices remotely, enforce security policies, and ensure compliance without requiring physical access to devices made it an invaluable tool during this period of rapid change. The pandemic accelerated the shift toward cloud-based management, propelling Intune to the forefront of the device management landscape.

Microsoft's decision to bundle Intune with Microsoft 365 also paid off during this time. Many organizations already had access to Intune as part of their existing subscriptions, making it easier to adopt the service without additional costs. This widespread availability, combined with the urgency of the remote work shift, led to a surge in Intune deployments across industries.

Intune's Maturity: Becoming a Leader in Unified Endpoint Management (2021-2022)

By 2021-2022, Microsoft Intune had matured into a comprehensive and powerful platform for unified endpoint management. The days of Intune being seen as a secondary option to Microsoft Configuration Manager were over. Microsoft's relentless focus on enhancing Intune's capabilities had paid off, and the platform now boasted a robust feature set that could rival any competitor in the market.

Key innovations during this period included Zero Touch Provisioning, which automated the entire device setup process from procurement to deployment, and comprehensive automation of Windows updates, application patching, and compliance actions. These features made it possible for organizations to manage their entire device fleet with minimal manual intervention, revolutionizing device management after decades of manual processes.

Intune's success was also recognized by industry analysts. Gartner's designation of Microsoft as a leader in unified endpoint management (UEM) was a testament to the

platform's evolution. The idea of managing a million devices on a single Intune tenant, once thought impossible, had become a reality. Organizations were no longer questioning whether Intune could handle their needs—they were embracing it as the future of device management.
The challenges during this period were less about the technology itself and more about adapting people and processes to this new, cloud-first way of managing devices. As organizations continued to modernize their IT environments, Intune became a critical component of their digital transformation strategies.

The Launch of the Intune Suite and Continued Innovation (2023 and beyond)

Microsoft's commitment to Intune's evolution didn't stop with the platform's initial success. In 2023, Microsoft announced the launch of the Intune Suite, which brought together a range of innovative features under a single license. This move further solidified Intune's position as the go-to solution for device management, eliminating the need for many third-party tools that organizations had previously relied on.

The Intune Suite introduced several key features, including Endpoint Privilege Management, Advanced App Management, Cloud Certificate Management, Advanced Endpoint Analytics, Remote Help, and Tunnel for BYOD devices. These features provided organizations with even greater control, security, and flexibility in managing their device environments. The Intune Suite also demonstrated Microsoft's commitment to continuous innovation, ensuring that the platform remained at the cutting edge of device management technology.

Additionally, AI and automation began to play an increasingly important role in Intune’s development. Microsoft invested heavily in integrating AI-driven insights and automation capabilities into the platform, helping organizations optimize their device management processes and enhance security through predictive analytics and automated remediation.

Looking forward, Microsoft’s ongoing investment in AI and automation promises to take Intune to new heights. As organizations continue to embrace remote work, hybrid environments, and digital transformation, Intune will remain a critical tool for managing and securing their IT infrastructure.

Conclusion

The evolution of Microsoft Intune is a remarkable story of growth, innovation, and adaptation. From its humble beginnings as Windows Intune to its current status as a leader in unified endpoint management, Intune has continuously evolved to meet the changing needs of modern organizations. Through strategic pivots, technological advancements, and a commitment to cloud-based management, Microsoft has positioned Intune as the future of device management.

As we move forward, the Intune Suite and the integration of AI-driven capabilities will ensure that Intune remains at the forefront of the industry, helping organizations manage their devices more efficiently, securely, and intelligently. The journey of Intune is far from over—in fact, it's only just beginning.

Chapter 4: Intune's Capabilities in Analytics and Automation

Introduction

As organizations embrace cloud-based device management solutions, the ability to harness data for analytics and automation has become increasingly crucial. Microsoft Intune isn't just a tool for managing devices; it's a powerful platform that delivers valuable insights and streamlines IT operations through analytics and automation. These capabilities enable IT teams to move beyond manual, repetitive tasks and focus on strategic initiatives that drive business growth, productivity, and innovation.

In this chapter, we will explore the depth of Intune’s capabilities in analytics and automation, examining how these features are already transforming device management. From foundational reporting tools to advanced automation features, Intune provides organizations with the means to manage their IT environments intelligently and efficiently.

The Foundation of Device Management: Reporting and Analytics

At the core of effective device management lies the ability to understand the health, compliance, and security status of the devices within your organization. Intune's robust reporting and analytics tools provide critical insights into these areas, serving as the foundation for more advanced automation and AI-driven operations.

Comprehensive reporting allows IT administrators to make informed decisions, troubleshoot issues, and ensure that devices comply with organizational policies. These reports not only offer snapshots of the current state but also help in identifying trends over time, providing a historical context that can inform long-term strategies. Whether it's monitoring the security posture of devices, tracking compliance with policies, or analyzing device configurations, Intune’s reporting features are vital for maintaining an efficient and secure IT environment.

Before diving into custom analytics solutions or advanced AI integrations, it's essential to explore what Intune offers out of the box. Microsoft continuously enhances Intune’s built-in analytics and automation features, with new capabilities being added regularly. By fully leveraging Intune’s native capabilities, organizations can often meet their needs without the need for extensive custom solutions. In cases where out-of-the-box features fall short, Intune’s flexibility allows for community-driven or custom-built enhancements.

In-Depth Reporting Capabilities

Intune's reporting infrastructure is designed to provide relevant data where it’s needed most. Reporting is decentralized, meaning that reports are accessible within the specific sections of the Microsoft Intune admin center where they are most applicable. For example, app-related reports are available in the App section, while device-

related reports can be found in the Device section. This contextual approach ensures that IT administrators can quickly access the information they need without unnecessary navigation.
Intune's reporting capabilities span a wide range of critical areas:

- **Device Compliance Reports**: These reports provide snapshots of how well devices adhere to the organization's policies, ensuring that security and compliance standards are met.

- **Device Configuration Reports**: Insights into the current configurations of devices across the organization, allowing IT teams to assess and optimize settings.

- **Device Reports**: Detailed information about individual devices, including hardware specifics, installed applications, and update status. This data helps in managing device lifecycles and planning for upgrades or replacements.

- **Device Security Reports**: Overviews of the security posture of devices, highlighting antivirus software status, detected threats, and overall device security health.

These reports cater to different focus groups within the organization and are categorized into several types:

- **Operational Reports**: These reports deliver real-time data that support quick decision-making and immediate actions. They are particularly valuable to administrators, subject matter experts, and helpdesk professionals who need accurate, up-to-date information.

- **Organizational Reports**: Offering a high-level overview, these reports provide managers and administrators with a broad perspective of the overall device management landscape, allowing them to assess organizational health at a glance.

- **Historical Reports**: By uncovering patterns and trends over specified timeframes, these reports offer historical context that can inform strategic decisions and long-term planning.
- **Specialist Reports**: These customizable reports allow administrators to extract raw data and craft reports that meet specific requirements, offering flexibility to tailor reporting to unique organizational needs.

The Reporting Infrastructure: V1 and V2

Intune is transitioning its reporting infrastructure from V1 to V2, with the majority of reports already migrated to the more advanced V2 system. The V2 reporting framework offers a more comprehensive and consistent reporting experience, enhancing the functionality and user experience. Key features of V2 include:

- **Search and Sort Across All Columns**: Regardless of dataset size, the ability to search and sort data across all columns ensures that administrators can quickly find the information they need.

- **Data Paging**: V2 allows for efficient navigation through large datasets by paging through data or jumping directly to specific pages.

- **Scalability**: The V2 framework is designed to handle substantial tenant data, ensuring that reports can be generated and viewed efficiently, even for large organizations.

- **Export Capabilities**: Administrators can swiftly export reporting data, making it easier to share insights and integrate with other tools or workflows.

To access reports in Intune, users must have one of the following roles:

- **Global Administrator**
- **Intune Administrator**
- **Administrators Assigned to an Intune Role with Read Permissions**

Reports are accessible through the Microsoft Intune admin center, and many can also be accessed via Microsoft Graph, which offers a powerful Export API for asynchronous data export.

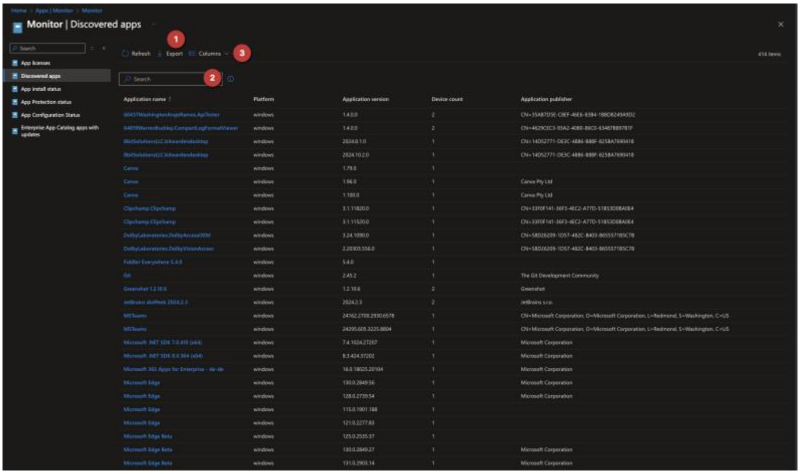
Reporting Examples: Discovered Apps and Assignment Status Reports

Let’s look at two examples of Intune’s reporting capabilities in action:

1. **Discovered Apps Report:** This report provides detailed information about the apps installed on devices across the organization. Key features include:

- **Export Button:** Easily download the report for offline analysis or sharing with stakeholders.
- **Filtering Options:** Quickly locate specific apps by using the built-in text filter, streamlining the management of large app inventories.

- **Column selection**: Column Selection: Users can customize the Discovered Apps Report by selecting from a variety of columns, such as: Name, Publisher, Version, Device Count, Platform and many more

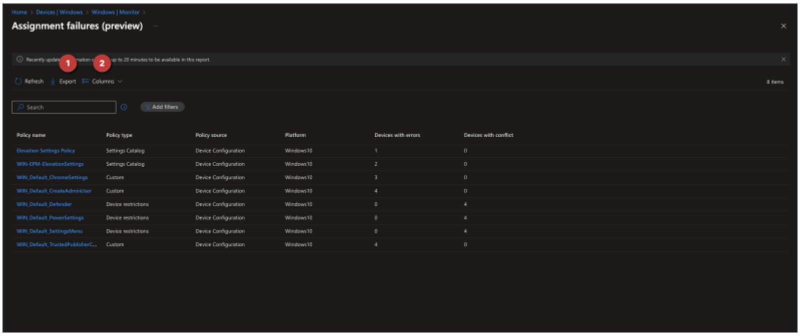


2. **Assignment failures:** This report is crucial for tracking the deployment status of failed policies and configurations. It provides a clear view to see

for which issues may have arisen. Additional features include:

- **Customizable Columns:** Tailor the report to focus on the specific data points that matter most to the current task by adding or removing columns.

- **Full Data Display:** Unlike some reports, the assignment status report displays all relevant data in one view, making it easy to assess the overall deployment landscape without needing to navigate through multiple pages.



Using Microsoft Graph for Data Export

To truly harness the power of Microsoft Intune, it's essential to understand how to work with Microsoft Graph, the backbone of data access and intelligence in the Microsoft ecosystem. Before diving into exporting data from Intune reports, let’s first clarify what Microsoft Graph is and how it functions.

What is Microsoft Graph?

Microsoft Graph is a unified API that allows developers and IT administrators to access data from across Microsoft services, including:

- **Microsoft 365 core services**
- **Enterprise Mobility + Security services**
- **Windows services**
- **And many more**

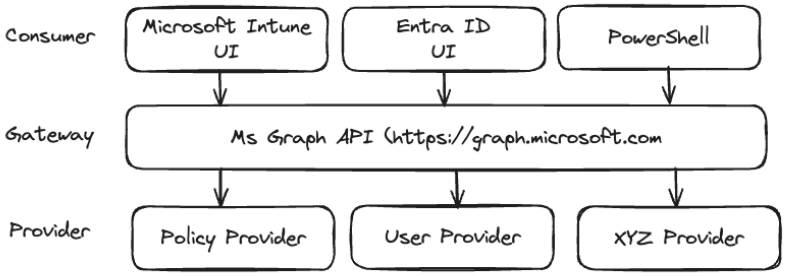
What many may not realize is that most Microsoft portals, such as the Microsoft Intune admin center or Entra ID, are essentially user interfaces built on top of Microsoft Graph. Actions performed in these portals—whether assigning policies or retrieving data—are, in the background, Graph API calls that the UI displays in a user-friendly format. Understanding this underlying structure empowers administrators to directly interact with the Graph API, offering more flexibility and control.

How Does Microsoft Graph Work?

Microsoft Graph acts as a gateway to various Microsoft services. Whether you're working in the Intune portal, Entra ID, or running a PowerShell script, your actions are translated into Graph API calls. For instance, if you want to retrieve a list of users from Entra ID, the Graph API call would be:

https://graph.microsoft.com/v1.0/users

This call routes through the Graph gateway, which then communicates with the appropriate service provider (in this case, the Entra ID service) to retrieve the data.



Graph API Methods

The Microsoft Graph API supports several REST methods, enabling a wide range of operations:

- **POST**: Adds new data to Microsoft Graph (e.g., creating new users or devices).
- **GET**: Retrieves data from Microsoft Graph (e.g., pulling device compliance reports).
- **PUT**: Updates existing data in Microsoft Graph (e.g., modifying user attributes).
- **PATCH**: Partially updates resources in Microsoft Graph.
- **DELETE**: Removes specific resources from Microsoft Graph.

Authentication and Token Management

To access data through Microsoft Graph, authentication is required via an authentication token. Several flows exist for acquiring these tokens, tailored to different scenarios:

- **Authorization Code Flow**: For user sign-in, often used in web applications and mobile apps.

- **Client Credentials Flow**: Used for server-to-server communication, often employed in automation scripts without user interaction.

- **Device Code Flow**: Ideal for devices with limited input capabilities, such as smart TVs or IoT devices.

- **On-Behalf-Of (OBO) Flow**: Allows an upstream API to access downstream APIs on behalf of the user.

- **Implicit Grant Flow**: A less recommended flow now replaced by more secure methods like PKCE (Proof Key for Code Exchange).

- **Integrated Windows Authentication (IWA) Flow**: Used in domain-joined or Entra-joined devices to acquire tokens silently.

- **Username/Password (ROPC) Flow**: Handles user credentials directly, generally not recommended due to security concerns.

Each flow is designed for specific use cases, and choosing the right one is critical for secure and efficient access to Microsoft Graph.

The Role of the Microsoft Authentication Library (MSAL)

The Microsoft Authentication Library (MSAL) simplifies the process of acquiring authentication tokens. It supports various platforms, including .NET, JavaScript, Python, and more, making it a versatile tool for developers and IT professionals. MSAL manages the lifecycle of access tokens, including refreshing them as needed to ensure continuous, secure access to Microsoft services.

Token Types and Refresh Mechanisms

When interacting with Microsoft Graph, three primary types of tokens are involved:

- **Access Tokens**: Used to authenticate API requests, typically valid for a short duration (e.g., a few hours).

- **Refresh Tokens**: Issued alongside access tokens, they allow the application to obtain new access tokens without requiring the user to re-authenticate. Refresh tokens are valid for much longer periods (days or even weeks).

- **ID Tokens**: Used in OpenID Connect scenarios to prove the identity of the user.

When an access token expires, the refresh token is used to request a new one, ensuring that the user experience remains seamless and uninterrupted.

Structure of a Graph API Call
A typical Microsoft Graph API call follows this structure:

https://graph.microsoft.com/[version]/[resource]/?[query-parameters]

- **Base URL**: The base URL for all Graph API requests is https://graph.microsoft.com/.

- **Version**: Specifies the API version, such as v1.0 for production-ready APIs or beta for preview features.

- **Resource**: The specific entity you're interacting with (e.g., users, devices, me).

- **Query Parameters**: Optional parameters to modify the request, such as $filter, $select, $top, or $orderby.

Examples of Graph API Calls:

- **Retrieve the Profile of the Signed-in User**: https://graph.microsoft.com/v1.0/me

- **List the First 10 Users in an Organization**: https://graph.microsoft.com/v1.0/users?$top=10

- **Search for Users by Name**: https://graph.microsoft.com/v1.0/users?$filter=startsWith(displayName, 'John')

Exporting Data from Intune Reports with Microsoft Graph

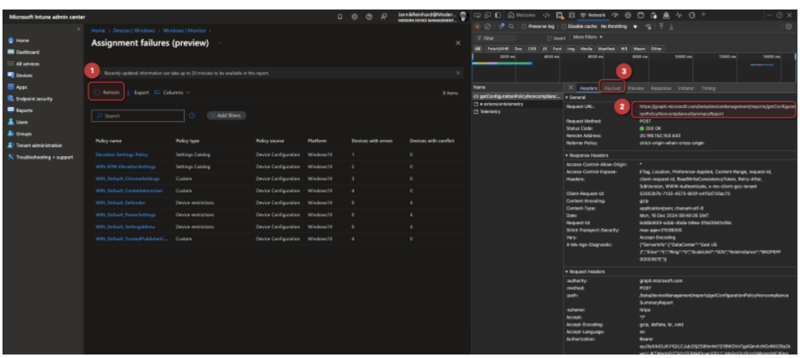
Now that we've covered the fundamentals let's apply this knowledge to exporting data from Intune reports via Microsoft Graph. The goal here is to leverage Graph to extract the same data you would typically retrieve from the Microsoft Intune admin center.

Step 1: Identify the Correct Graph API Call

There are two primary ways to find the appropriate Graph API call for exporting Intune report data:

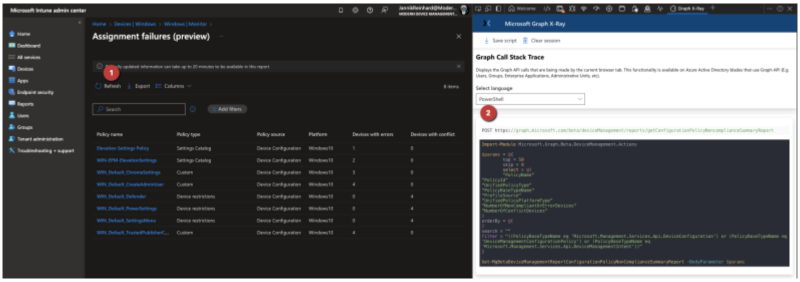
1. **Using Browser Developer Tools**:

- Open the report in the Intune Admin Center.
- Press F12 (or navigate to More Tools > Developer Tools) to open the browser's developer tools.
- Switch to the Network tab and refresh the report.
- Find the corresponding API call and body in the network requests log.



2. Using Graph X-Ray:

- Graph X-Ray is a tool that helps identify Graph API calls and can generate code in your preferred programming language.
- After refreshing the report, Graph X-Ray will capture the API call and provide the necessary code snippets.

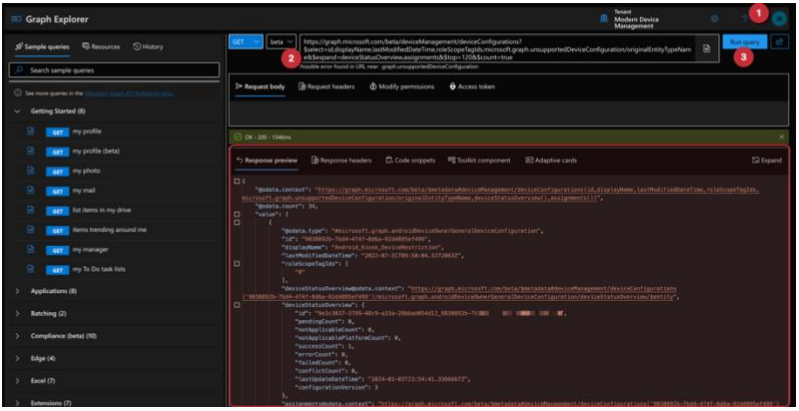


Step 2: Running the Graph Call

There are two primary methods to execute the Graph API call:

3. **Using Microsoft Graph Explorer**:

- Open the Graph Explorer at https://aka.ms/graph
- Authenticate on the page.
- Enter the Graph endpoint identified earlier (e.g., the API call for an assignment report).
- Run the query to retrieve the data. Graph Explorer is particularly useful for testing and tweaking queries.



4. Using PowerShell:

- You can also run the Graph API call using a PowerShell script. This approach is especially useful for automating data exports and integrating them into broader workflows. Below is a basic example of a PowerShell script that performs the same actions as Graph Explorer:

https://github.com/JayRHa/Book/blob/main/graph/example.ps1

```
# Check if Microsoft Graph module is installed
$module = Get-Module -Name Microsoft.Graph -ListAvailable
if ($module -eq $null) {
    Install-Module -Name Microsoft.Graph -Scope CurrentUser -Force
    Import-Module -Name Microsoft.Graph
} else {
    Write-Host "Microsoft Graph module is already installed."
}

# Authentication
Connect-MgGraph

# Execute Graph call
Get-MgBetaDeviceManagementDeviceConfiguration -Property
"id,displayName,lastModifiedDateTime,roleScopeTagIds,microsoft.graph.unsupportedDeviceConfiguratio
-ExpandProperty "deviceStatusOverview,assignments" -Top 120 -CountVariable CountVar
```

```
# Check if Microsoft Graph module is installed
$module = Get-Module -Name Microsoft.Graph -ListAvailable
```

```
if ($module -eq $null) {
    Install-Module -Name Microsoft.Graph -Scope
CurrentUser -Force
    Import-Module -Name Microsoft.Graph
} else {
    Write-Host "Microsoft Graph module is already
installed."
}

# Authentication
Connect-MgGraph

# Execute Graph call
Get-MgBetaDeviceManagementDeviceConfiguration -Property
"id,displayName,lastModifiedDateTime,roleScopeTagIds,micro
soft.graph.unsupportedDeviceConfiguration/originalEntityTy
peName" -ExpandProperty "deviceStatusOverview,assignments"
-Top 120 -CountVariable CountVar
```

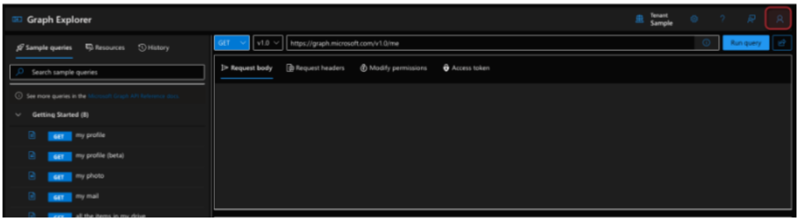
Using the Graph Export API for Large Data Exports

When dealing with large datasets, such as exporting all discovered apps across thousands of devices, pagination becomes necessary. However, the Graph Export API is designed to handle these large data exports efficiently by providing the results as downloadable CSV files.

Step-by-Step Guide to Using the Graph Export API

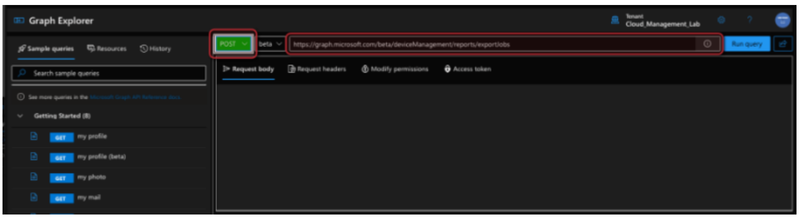
Initiate the Export Job

- Start by logging into the Graph Explorer and switching the method to POST.



- Use the following endpoint for your request:

https://graph.microsoft.com/beta/deviceManagement/reports/exportJobs



- Define the body of the request based on the report you want to export. For example, to export all discovered apps:

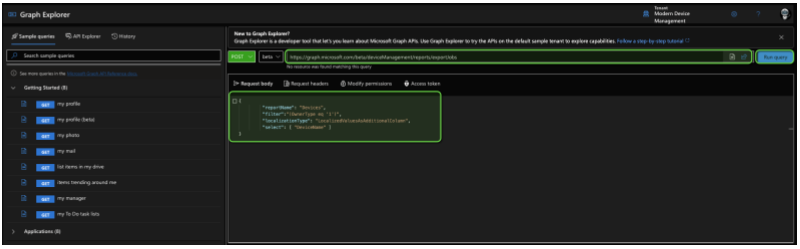
```
{
    "reportName": "DevicesByAppInv",
    "localizationType":
"LocalizedValuesAsAdditionalColumn"
}
```

```
{
         "reportName": "Devices",
         "filter":"(OwnerType eq '1')",
         "localizationType":
"LocalizedValuesAsAdditionalColumn",
         "select": [ "DeviceName" ]
}
```

This is the list of all export APIs

ReportName (Export Parameter)	**Associated Report in Microsoft Intune**
DeviceCompliance	Device Compliance Org
DeviceNonCompliance	Non-compliant devices
Devices	All devices list
FeatureUpdatePolicyFailuresAggregate	Under Devices > Monitor > Failure for feature updates
DeviceFailuresByFeatureUpdatePolicy	Under Devices > Monitor > Failure for feature updates > *click on error*
FeatureUpdateDeviceState	Under Reports > Window Updates > Reports > Windows Feature Update Report
UnhealthyDefenderAgents	Under Endpoint Security > Antivirus > Win10 Unhealthy Endpoints
DefenderAgents	Under Reports > MicrosoftDefender > Reports > Agent Status
ActiveMalware	Under Endpoint Security > Antivirus > Win10 detected malware
Malware	Under Reports > MicrosoftDefender > Reports > Detected malware
AllAppsList	Under Apps > All Apps
AppInstallStatusAggregate	Under Apps > Monitor > App install status
DeviceInstallStatusByApp	Under Apps > All Apps > *Select an individual app*
UserInstallStatusAggregateByApp	Under Apps > All Apps > *Select an individual app*
ComanagedDeviceWorkloads	Under Reports > Cloud attached devices > Reports > Co-Managed Workloads
ComanagementEligibilityTenantAttachedDevices	Under Reports > Cloud attached devices > Reports > Co-Management Eligibility

DeviceRunStatesByProactiveRemediation	Under Reports > Endpoint Analytics > Proactive remediations > *Select a remediation* > Device status
DevicesWithInventory	Under Devices > All Devices > Export
FirewallStatus	Under Reports > Firewall > MDM Firewall status for Windows 10 and later
GPAnalyticsSettingMigrationReadiness	Under Reports > Group policy analytics > Reports > Group policy migration readiness
QualityUpdateDeviceErrorsByPolicy	Under Devices > Monitor > Windows Expedited update failures > *Select a profile*
QualityUpdateDeviceStatusByPolicy	Under Reports > Windows updates > Reports > Windows Expedited Update Report
MAMAppProtectionStatus	Under Apps > Monitor > App protection status > App protection report: iOS, Android
MAMAppConfigurationStatus	Under Apps > Monitor > App protection status > App configuration report
DevicesByAppInv	Under Apps > Monitor > Discovered apps > Discovered app> Export
AppInvByDevice	Under Devices > All Devices > Device > Discovered Apps
AppInvAggregate	Under Apps > Monitor > Discovered apps > Export
AppInvRawData	Under Apps > Monitor > Discovered apps > Export

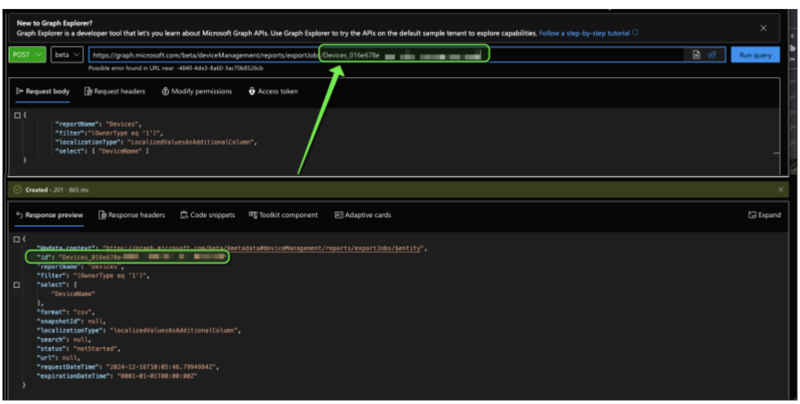


Monitor the Export Job:

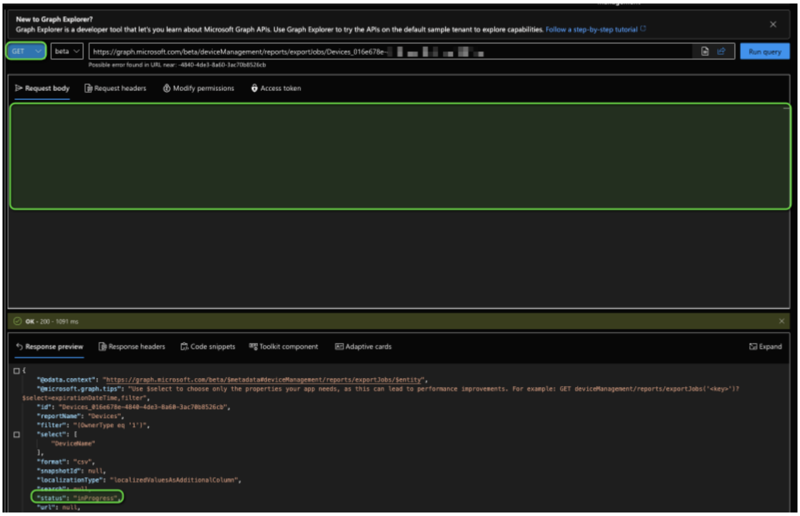
- After running the initial POST request, you will receive a job ID. Append this ID to the URL for the next API call:

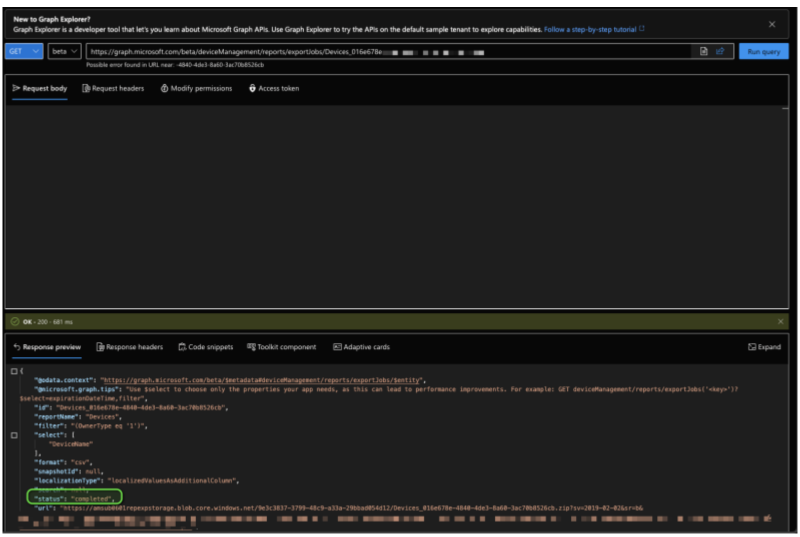
https://graph.microsoft.com/beta/deviceManagement/reports/exportJobs('{jobId}')

Like:
https://graph.microsoft.com/beta/deviceManagement/reports/exportJobs('Devices_xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx')



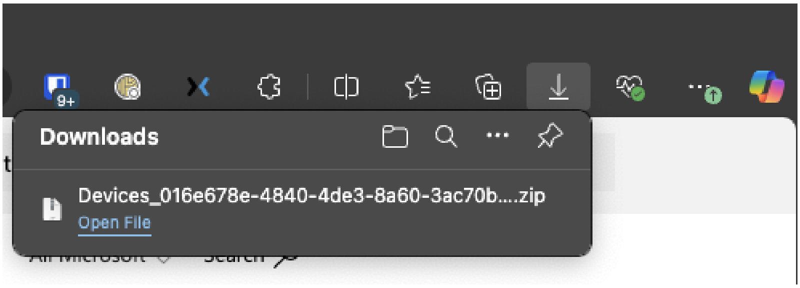
- Change the method to GET and run the query to check the status of the export job. Once the status is marked as "completed," a download URL will be provided.





- **Download the Exported Data:**

- Copy the download URL and paste it into your browser to retrieve the CSV file. The file will be delivered as a zipped archive, which you can then unpack to access the data.



Automating the Export Process:

You can automate this entire process with PowerShell, allowing you to schedule regular exports or trigger them as part of a larger automation pipeline. Below is an example script that initializes and monitors the export, then downloads and unpacks the resulting CSV file:

https://github.com/JayRHa/Book/blob/main/graph/exportApi.ps1

```
# Check if Microsoft Graph module is installed
$module = Get-Module -Name Microsoft.Graph -ListAvailable
if ($module -eq $null) {
    Install-Module -Name Microsoft.Graph -Scope CurrentUser -Force
    Import-Module -Name Microsoft.Graph
} else {
    Write-Host "Microsoft Graph module is already installed."
}

# Authentication
Connect-MgGraph

$reportName = 'DetectedAppsRawData'

$body = @"
{
"reportName": "$reportName",
"localizationType": "LocalizedValuesAsAdditionalColumn"
}
"@

$id = (Invoke-MgGraphRequest -Uri
https://graph.microsoft.com/beta/deviceManagement/reports/exportJobs -Method POST -Body
$body).id
$status = (Invoke-MgGraphRequest -Uri
"https://graph.microsoft.com/beta/deviceManagement/reports/exportJobs('$id')" -Method
GET).status

while (-not ($status -eq 'completed')) {
    $response = Invoke-MgGraphRequest -Uri
"https://graph.microsoft.com/beta/deviceManagement/reports/exportJobs('$id')" -Method Get
    $status = ($response).status
}

Invoke-WebRequest -Uri $response.url -OutFile "./intuneExport.zip"
Expand-Archive "./intuneExport.zip" -DestinationPath "./intuneExport"
```

```
# Check if Microsoft Graph module is installed
$module = Get-Module -Name Microsoft.Graph -ListAvailable
if ($module -eq $null) {
    Install-Module -Name Microsoft.Graph -Scope
CurrentUser -Force
    Import-Module -Name Microsoft.Graph
} else {
    Write-Host "Microsoft Graph module is already
installed."
}

# Authentication
Connect-MgGraph

$reportName = 'DetectedAppsRawData'

$body = @"
{
```

```
"reportName": "$reportName",
"localizationType": "LocalizedValuesAsAdditionalColumn"
}
"@

$id = (Invoke-MgGraphRequest -Uri
https://graph.microsoft.com/beta/deviceManagement/reports/
exportJobs -Method POST -Body $body).id
$status = (Invoke-MgGraphRequest -Uri
"https://graph.microsoft.com/beta/deviceManagement/reports
/exportJobs('$id')" -Method GET).status

while (-not ($status -eq 'completed')) {
    $response = Invoke-MgGraphRequest -Uri
"https://graph.microsoft.com/beta/deviceManagement/reports
/exportJobs('$id')" -Method Get
    $status = ($response).status
}

Invoke-WebRequest -Uri $response.url -OutFile
"./intuneExport.zip"
Expand-Archive "./intuneExport.zip" -DestinationPath
"./intuneExport"
```

This script handles the entire process, from initiating the export to downloading and unpacking the data, allowing you to automate complex data extraction tasks with minimal manual intervention.

Run multiple calls in one

There is one more thing, and this is the Microsoft Graph Batch endpoint.
The usage of this endpoint is very easy. What you must do is build a JSON where the different calls are listed. One example is from the Intune Portal to get a summary of the tenant state. This JSON looks like this:

```
{
    "requests": [
        {
            "id": "getDeviceComplianceStateSummary",
            "method": "GET",
            "url": "/deviceManagement/deviceCompliancePolicyDeviceStateSummary",
            "headers": {
                "x-ms-command-name": "fetchDeviceComplianceStateSummaryBatch"
            },
        },
        {
            "id": "fetchSubscriptionState",
            "method": "GET",
            "url": "/deviceManagement/subscriptionState",
            "headers": {
                "x-ms-command-name": "fetchSubscriptionStateBatch"
            },
        },
        {
            "id": "getFailedAppCount",
            "method": "POST",
            "url": "/deviceManagement/reports/getFailedMobileAppsSummaryReport",
            "body": {
                "filter": ""
            },
            "headers": {
                "Content-Type": "application/json",
                "x-ms-command-name": "fetchFailedAppCountBatch"
            },
        },
        {
            "id": "getDeviceConfigPolicySummary",
            "method": "POST",
            "url": "/deviceManagement/reports/getDeviceConfigurationPolicyStatusSummary",
            "body": {
                "filter": "(PolicyBaseTypeName eq 'DeviceManagementConfigurationPolicy') or
(PolicyBaseTypeName eq 'Microsoft.Management.Services.Api.DeviceConfiguration') or
(PolicyBaseTypeName eq 'Microsoft.Management.Services.Api.DeviceManagementIntent')"
            },
            "headers": {
                "Content-Type": "application/json",
                "x-ms-command-name": "fetchDeviceConfigPolicySummary"
            },
        },
    ]
}
```

```
{
    "requests": [
        {
            "id": "getDeviceComplianceStateSummary",
            "method": "GET",
            "url":
"/deviceManagement/deviceCompliancePolicyDeviceStateSum
mary",
            "headers": {
                "x-ms-command-name":
"fetchDeviceComplianceStateSummaryBatch"
            },
        },
```

```
        {
            "id": "fetchSubscriptionState",
            "method": "GET",
            "url":
"/deviceManagement/subscriptionState",
            "headers": {
                "x-ms-command-name":
"fetchSubscriptionStateBatch"
            },
        },
        {
            "id": "getFailedAppCount",
            "method": "POST",
            "url":
"/deviceManagement/reports/getFailedMobileAppsSummaryRe
port",
            "body": {
                "filter": ""
            },
            "headers": {
                "Content-Type": "application/json",
                "x-ms-command-name":
"fetchFailedAppCountBatch"
            },
        },
        {
            "id": "getDeviceConfigPolicySummary",
            "method": "POST",
            "url":
"/deviceManagement/reports/getDeviceConfigurationPolicy
StatusSummary",
            "body": {
                "filter": "(PolicyBaseTypeName eq
'DeviceManagementConfigurationPolicy') or
(PolicyBaseTypeName eq
'Microsoft.Management.Services.Api.DeviceConfiguration'
) or (PolicyBaseTypeName eq
'Microsoft.Management.Services.Api.DeviceManagementInte
nt')"
            },
            "headers": {
                "Content-Type": "application/json",
                "x-ms-command-name":
"fetchDeviceConfigPolicySummary"
            },
        },
    ]
```

```
}
```

You only have to define the method, header, URL, and optional body in an array.

Attached you can find an example script from me on how you can use it in Python:

https://github.com/JayRHa/Book/blob/main/graph/bstchApi.py

```python
import requests
from azure.identity import InteractiveBrowserCredential

credential = InteractiveBrowserCredential()
token = credential.get_token("https://graph.microsoft.com/.default")

def call_graph(access_token: str, url: str, body, method: str = "GET"):
    headers = {
        "Authorization": f"Bearer {access_token}",
        "Content-Type": "application/json",
    }
    if method == "GET":
        response = requests.get(
            url,
            headers=headers,
        )
    else:
        response = requests.post(
            url,
            headers=headers,
            json=body,
        )
    response.raise_for_status()
    return response.json()



url = "https://graph.microsoft.com/beta/$batch"
body = {
    "requests": [
        {
            "id": "getDeviceComplianceStateSummary",
            "method": "GET",
            "url": "/deviceManagement/deviceCompliancePolicyDeviceStateSummary",
            "headers": {"x-ms-command-name": "fetchDeviceComplianceStateSummaryBatch"},
        },
        {
            "id": "fetchSubscriptionState",
            "method": "GET",
            "url": "/deviceManagement/subscriptionState",
            "headers": {"x-ms-command-name": "fetchSubscriptionStateBatch"},
        },
        {
            "id": "getFailedAppCount",
            "method": "POST",
            "url": "/deviceManagement/reports/getFailedMobileAppsSummaryReport",
            "body": {"filter": ""},
            "headers": {"Content-Type": "application/json", "x-ms-command-name":
"fetchFailedAppCountBatch"},
        },
        {
            "id": "getDeviceConfigPolicySummary",
            "method": "POST",
            "url": "/deviceManagement/reports/getDeviceConfigurationPolicyStatusSummary",
            "body": {
                "filter": "(PolicyBaseTypeName eq 'DeviceManagementConfigurationPolicy') or
(PolicyBaseTypeName eq 'Microsoft.Management.Services.Api.DeviceConfiguration') or
(PolicyBaseTypeName eq 'Microsoft.Management.Services.Api.DeviceManagementIntent')"
            },
            "headers": {"Content-Type": "application/json", "x-ms-command-name":
"fetchDeviceConfigPolicySummary"},
        },
    ]
}
response = call_graph(token.token, url, body, "POST")
print(response)
```

```
import requests
from azure.identity import InteractiveBrowserCredential

credential = InteractiveBrowserCredential()
token =
credential.get_token("https://graph.microsoft.com/.defa
ult")

def call_graph(access_token: str, url: str, body,
method: str = "GET"):
    headers = {
        "Authorization": f"Bearer {access_token}",
        "Content-Type": "application/json",
    }
    if method == "GET":
        response = requests.get(
            url,
            headers=headers,
        )
    else:
        response = requests.post(
            url,
            headers=headers,
            json=body,
        )
    response.raise_for_status()
    return response.json()



url = "https://graph.microsoft.com/beta/$batch"
body = {
    "requests": [
        {
            "id": "getDeviceComplianceStateSummary",
            "method": "GET",
            "url":
"/deviceManagement/deviceCompliancePolicyDeviceStateSum
mary",
            "headers": {"x-ms-command-name":
"fetchDeviceComplianceStateSummaryBatch"},
        },
        {
            "id": "fetchSubscriptionState",
            "method": "GET",
            "url":
"/deviceManagement/subscriptionState",
```

```
            "headers": {"x-ms-command-name":
"fetchSubscriptionStateBatch"},
        },
        {
            "id": "getFailedAppCount",
            "method": "POST",
            "url":
"/deviceManagement/reports/getFailedMobileAppsSummaryRe
port",
            "body": {"filter": ""},
            "headers": {"Content-Type":
"application/json", "x-ms-command-name":
"fetchFailedAppCountBatch"},
        },
        {
            "id": "getDeviceConfigPolicySummary",
            "method": "POST",
            "url":
"/deviceManagement/reports/getDeviceConfigurationPolicy
StatusSummary",
            "body": {
                "filter": "(PolicyBaseTypeName eq
'DeviceManagementConfigurationPolicy') or
(PolicyBaseTypeName eq
'Microsoft.Management.Services.Api.DeviceConfiguration'
) or (PolicyBaseTypeName eq
'Microsoft.Management.Services.Api.DeviceManagementInte
nt')"
            },
            "headers": {"Content-Type":
"application/json", "x-ms-command-name":
"fetchDeviceConfigPolicySummary"},
        },
    ]
}
response = call_graph(token.token, url, body, "POST")
print(response)
```

Azure Monitor Integration

One of the standout features of Microsoft Intune is its seamless integration with **Azure Monitor** and **Log Analytics**. This integration allows for the collection and analysis of detailed log information, which can be used to perform custom queries using **Kusto Query Language (KQL)**. By leveraging Azure Monitor, you can gain a more granular understanding of your device management environment—ranging from tracking software versions to monitoring device check-ins and overall compliance.

To use this feature, you'll first need to deploy a **Log Analytics workspace** and configure Intune to forward logs to this workspace. Here’s a step-by-step guide on how to enable and utilize this integration:

Supported Logs in Azure Monitor

Intune supports several types of logs that can be forwarded to Azure Monitor for analysis:

- **AuditLogs**: These logs record activities that generate a change in Intune, including create, update (edit), delete, assign, and remote actions.

- **OperationalLogs**: These logs provide details on user and device enrollments, including whether the enrollment succeeded or failed. They also provide insights into device compliance.

- **DeviceComplianceOrg**: These logs provide an organizational report for device compliance, including details on noncompliant devices.

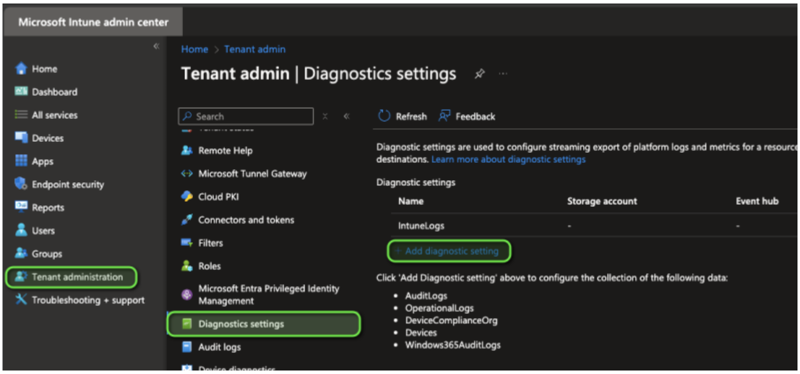
- **Devices**: These logs offer inventory and status information for Intune-enrolled and managed devices.

- **Windows365AuditLogs**: These audit logs capture activities that result in a change to a Cloud PC, such as create, update, delete, assign, and remote actions. Audit events are generated for most Cloud PC actions processed through Microsoft Graph, and this auditing is enabled by default for all customers.

Configuring Azure Monitor Integration

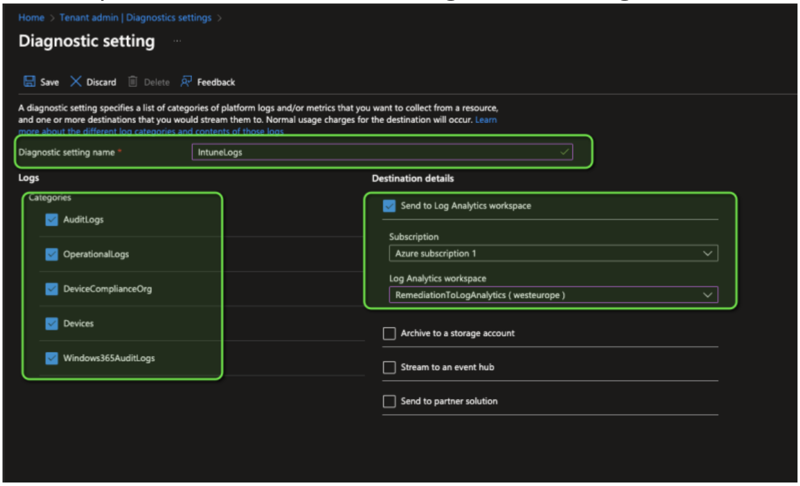
To get started, you need to set up a **Log Analytics workspace**:

1. **Deploy the Log Analytics Workspace**: In the Azure portal, create a new Log Analytics workspace if you don't have one already.
2. **Configure Log Forwarding**: Once the workspace is deployed, navigate to the **Intune Admin Center**. Go to **Tenant Administration > Diagnostics settings** and click on **+ Add diagnostic setting**.



3. **Select Logs and Destination**: In the next menu, select the logs you want to forward to the Log Analytics workspace. You'll need to choose **Send to Log Analytics workspace** as the destination and

select the appropriate workspace. Before saving, provide a name for the diagnostic setting.



Working with Kusto Query Language (KQL)

Once your logs are being forwarded to Azure Monitor, you can perform detailed analysis using KQL. KQL is a powerful query language that allows you to search, filter, and visualize data within Log Analytics. Here's a basic example of a KQL query that retrieves the last 100 Intune device compliance events:

```
DeviceComplianceOrg
| top 100 by TimeGenerated desc
| project DeviceName, ComplianceStatus, TimeGenerated
```

This query fetches the latest compliance events, showing the device name, compliance status, and the time the event was generated.

KQL allows for more complex queries, such as joining multiple data sets, aggregating results, or generating visualizations. For example, you can monitor trends in device compliance over time or identify patterns in enrollment failures.

A Full Tutorial for Kusto Query Language

Kusto Query Language (KQL) is a powerful query language used to interact with data in Azure Monitor, Log Analytics, and other Microsoft services. KQL is designed for exploring large datasets, filtering and transforming data, and producing summaries or visualizations. This section will provide you with a full tutorial on how to work with KQL, covering the basics as well as more advanced techniques to get the most out of your data in Azure Monitor.

Introduction to Kusto Query Language (KQL)

KQL is the primary language for querying data in Azure Monitor and Log Analytics. It is optimized for performing read-only queries on large datasets and is ideal for tasks like monitoring, diagnostics, and reporting.

Basic Structure of a KQL Query:

1. **Data Source**: The starting point is always a table or dataset, such as DeviceComplianceOrg or AuditLogs.

2. **Pipes (|)**: KQL uses the pipe operator to chain together commands, similar to how PowerShell works. Each command processes the output of the previous one.

3. **Commands**: Commands are used to filter, sort, aggregate, and transform data. Common commands include where, summarize, project, and extend.

Basic Query Examples
To help you get started with KQL, let's go through some basic queries.

1. **Retrieve All Data from a Table**:

```
IntuneDeviceComplianceOrg
```

This simple query retrieves all data from the DeviceComplianceOrg table.

2. **Filter Data with the where Command**

```
IntuneDeviceComplianceOrg
| where ComplianceState == "NonCompliant"
```

This query filters the data to show only devices that are noncompliant.

3. **Project Specific Columns with the project Command**:

```
IntuneDeviceComplianceOrg
| project DeviceName, ComplianceState, TimeGenerated
```

The project command allows you to select specific columns to include in the output. Here, we are selecting DeviceName, ComplianceStatus, and TimeGenerated.

4. **Sort Data with the order by Command**:

```
IntuneDeviceComplianceOrg
| order by TimeGenerated desc
```

This query sorts the data by the TimeGenerated column in descending order, showing the most recent entries first.

5. **Limit the Number of Results with the top Command**:

```
IntuneDeviceComplianceOrg
| top 10 by TimeGenerated
```

The top command limits the number of results returned by the query. In this example, it returns the 10 most recent compliance records.

Aggregations and Summarizations

KQL is powerful when it comes to aggregating and summarizing data. You can use the summarize command to perform calculations like counts, sums, averages, and more.

1. **Count the Number of Noncompliant Devices**:

```
IntuneDeviceComplianceOrg
| where ComplianceState == "NonCompliant"
| summarize NonCompliantDeviceCount = count()
```

This query counts the number of devices with a ComplianceStatus of "NonCompliant."

2. **Group Data by a Specific Field**:

```
IntuneDeviceComplianceOrg
| summarize NonCompliantDevices = count() by OS
```

The summarize command here groups the data by the OperatingSystem column and counts the number of noncompliant devices for each operating system.

3. **Calculate the Average Compliance Check-in Time**:

```
IntuneDeviceComplianceOrg
| summarize AvgCheckinTime = avg(TimeGenerated)
```

This query calculates the average time of compliance check-ins across all devices.

Using Joins to Combine Data

In more complex scenarios, you may need to combine data from multiple tables. KQL supports joins, similar to SQL, allowing you to merge datasets based on common fields.

1. **Inner Join Example**:

```
IntuneDeviceComplianceOrg
| join kind=inner (
    IntuneDevices
    | project DeviceId, DeviceName
) on DeviceId
```

This query performs an inner join between the DeviceComplianceOrg and Devices tables based on the DeviceId column. The result will include only records that exist in both tables.

2. **Left Join Example**:

```
IntuneDeviceComplianceOrg
| join kind=leftouter (
    IntuneDevices
    | project DeviceId, DeviceName
) on DeviceId
```

A left outer join retrieves all records from the DeviceComplianceOrg table, along with matching records from the Devices table. If a device is present in the compliance table but not in the devices table, it will still appear in the result.

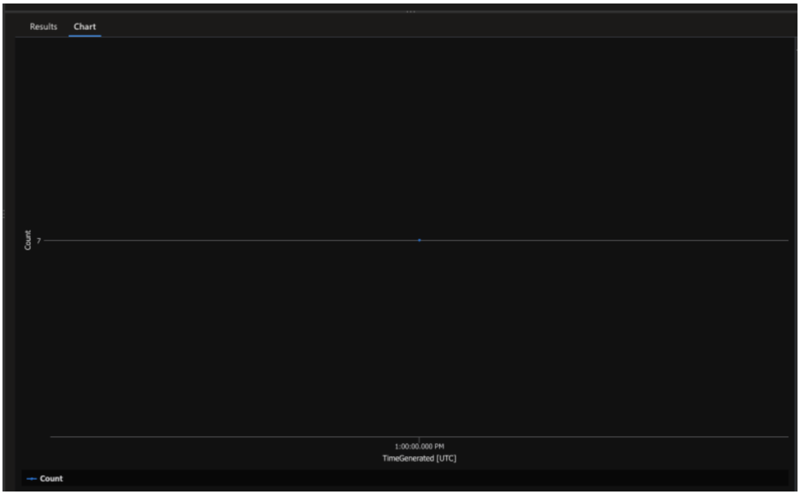
Visualizing Data with KQL

KQL supports built-in visualization commands, allowing you to create charts and graphs directly from your queries.

1. **Render a Time Chart**:

```
IntuneDeviceComplianceOrg
| summarize Count = count() by bin(TimeGenerated, 1h)
| render timechart
```

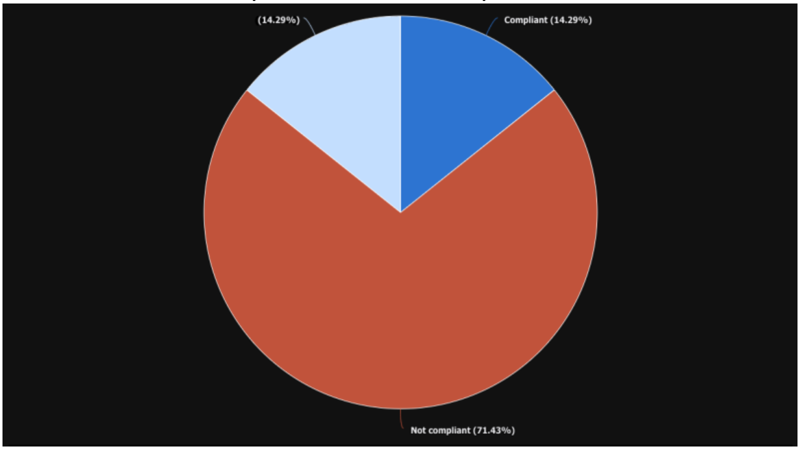
This query aggregates the data into hourly intervals and renders a time chart showing the count of compliance events over time.



2. **Render a Pie Chart**:

```
IntuneDeviceComplianceOrg
| summarize NonCompliantCount = count() by ComplianceState
| render piechart
```

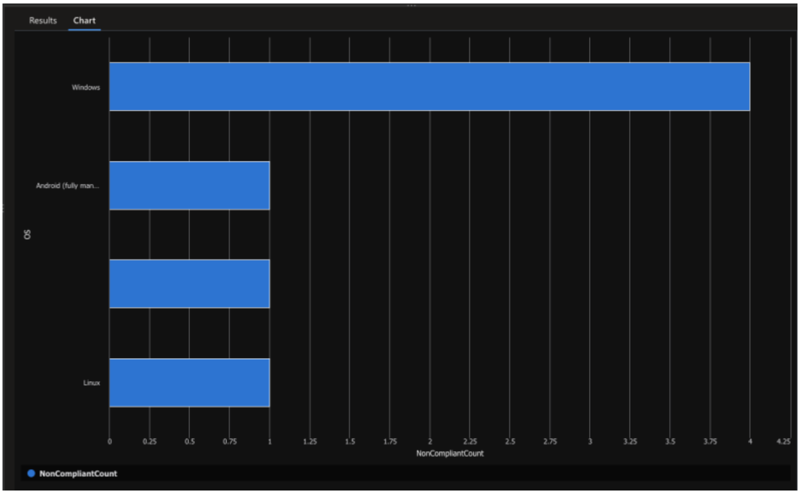
Here, the data is grouped by ComplianceState, and the results are visualized as a pie chart, showing the distribution of compliant vs. noncompliant devices.



3. **Render a Bar Chart**:

```
IntuneDeviceComplianceOrg
| summarize NonCompliantCount = count() by OS
| render barchart
```

This query creates a bar chart that displays the number of noncompliant devices for each operating system.



Advanced KQL Techniques

KQL also provides advanced features for handling complex queries and scenarios. Some of these include using **let statements** for reusable expressions, **union** to combine data from multiple tables, and **functions** to encapsulate complex logic.

1. **Using let Statements**:

```
let NonCompliantDevices = IntuneDeviceComplianceOrg
```

```
| where ComplianceState == "NonCompliant";
NonCompliantDevices
| summarize NonCompliantCount = count()
```

The let statement allows you to define a reusable subquery or expression. In this example, we define a subquery for noncompliant devices and then use it to count the results.

2. **Combining Data with union**:

```
IntuneDeviceComplianceOrg | union IntuneDevices
```

The union command combines data from the DeviceComplianceOrg and Devices tables, returning all records from both tables.

3. **Creating Functions**:

```
let GetNonCompliantDevices = () {
    IntuneDeviceComplianceOrg
    | where ComplianceState == "NonCompliant"
};
GetNonCompliantDevices()
| summarize NonCompliantCount = count()
```

This example shows how to encapsulate a query as a reusable function using let. Functions are especially useful in larger queries or when you need to reuse logic across multiple queries.

Example Scenario: End-to-End KQL Query

Let’s put everything together with a more complex query that demonstrates filtering, aggregating, joining, and visualizing data in one go.

1. **Scenario**: You want to analyze device compliance status, track it over time, and visualize the results with a bar chart.

2. **Solution**:

```
let ComplianceData = IntuneDeviceComplianceOrg
| where TimeGenerated >= ago(7d)
| where ComplianceState == "NonCompliant";

ComplianceData
| join kind=inner (
    IntuneDevices
    | project DeviceId, DeviceName, OS
) on DeviceId
| summarize NonCompliantCount = count() by OS
| render barchart
```

Explanation:

- We start by filtering IntuneDeviceComplianceOrg to retrieve compliance data from the past 7 days.
- The data is grouped into daily intervals and filtered to show only noncompliant devices.
- We join the compliance data with the Devices table to retrieve additional details about each device, such as its operating system.
- Finally, we summarize the data by operating system and render the results as a bar chart.

This query combines several KQL features into a comprehensive analysis, demonstrating the power and flexibility of KQL in real-world scenarios.

KQL Exercises

Exercise 1: Filter Non-Compliant Devices

Goal: Retrieve a list of all non-compliant devices.

Task:

- Filter the DeviceComplianceOrg table to show only devices with a ComplianceStatus of "NonCompliant".
- Display the DeviceName, ComplianceStatus, OperatingSystem, and TimeGenerated columns.

Solution:

```
IntuneDeviceComplianceOrg
| where ComplianceState == "NonCompliant"
| project DeviceName, ComplianceState, OS, TimeGenerated
```

Exercise 2: Count Devices by Operating System

Goal: Count the number of devices by their operating system.

Task:

- Group the devices by the OperatingSystem column.
- Count the number of devices for each operating system.

Solution:

```
IntuneDevices
| summarize DeviceCount = count() by OS
```

Exercise 3: Identify Recently Enrolled Devices

Goal: Find devices that were enrolled in the past 7 days.

Task:

- Filter the IntuneDevices table to show only devices that were enrolled in the last 7 days.
- Display the DeviceName, EnrollmentStatus, and TimeGenerated columns.

Solution:

```
IntuneDevices
| where TimeGenerated >= ago(7d)
| project DeviceName, CreatedDate, TimeGenerated
```

Exercise 4: Track Compliance Trends Over Time

Goal: Analyze how device compliance has changed over the past month.

Task:

- Filter the data to include the past 30 days.
- Group the data by daily intervals using bin(TimeGenerated, 1d).
- Focus on devices with a ComplianceStatus of "NonCompliant".

Solution:

```
IntuneDeviceComplianceOrg
| where TimeGenerated >= ago(30d)
| summarize NonCompliantCount = count() by
bin(TimeGenerated, 1d), ComplianceState
| where ComplianceState == "NonCompliant"
```

Exercise 5: Find Devices with Multiple Compliance Violations

Goal: Identify devices that have had multiple compliance violations.

Task:

- Filter the data to include only non-compliant devices.
- Count the number of non-compliant events for each device (ViolationCount).
- Return only devices with more than one violation.

Solution:

```
IntuneDeviceComplianceOrg
| where ComplianceState == "NonCompliant"
| summarize ViolationCount = count() by DeviceName
| where ViolationCount > 1
```

Exercise 6: Create a Pie Chart of Compliance Status

Goal: Visualize the distribution of device compliance statuses as a pie chart.

Task:

- Group the data by ComplianceStatus.
- Count the number of records for each compliance status.
- Render the results as a pie chart.

Solution:

```
IntuneDeviceComplianceOrg
| summarize ComplianceCount = count() by ComplianceState
```

```
| render piechart
```

Exercise 7: Audit Log: Track Admin Actions

Goal: Retrieve the latest 50 administrative actions from the audit logs.

Task:

- Sort the AuditLogs data by TimeGenerated in descending order.
- Return the most recent 50 actions.

Solution:

```
IntuneAuditLogs
| order by TimeGenerated desc
| top 50 by TimeGenerated
```

Exercise 8: Visualize Compliance Violations Over Time

Goal: Create a time chart showing the number of non-compliant devices over the past 30 days.

Task:

- Filter the data to include only non-compliant devices from the past 30 days.
- Group the data by daily intervals bin(TimeGeneratedm 1d)
- Render the results as a time chart to visualize compliance violations over time.

Solution:

```
IntuneDeviceComplianceOrg
```

```
| where TimeGenerated >= ago(30d) and ComplianceState ==
"NonCompliant"
| summarize NonCompliantCount = count() by
bin(TimeGenerated, 1d)
| render timechart
```

Sending Logs to Other Destinations

In addition to forwarding logs to a Log Analytics workspace, you can also configure Intune to send logs to other destinations, such as:

- **Azure Storage Account**: Store logs for long-term retention and archival.
- **Event Hub**: Stream logs to an event hub for real-time processing with other systems.
- **Partner Solutions**: Integrate with third-party tools that offer advanced monitoring or SIEM capabilities.

This flexibility allows organizations to tailor their log management and analytics to their specific needs.

In the next menu you can select the logs you are interested in and the destination. If you want to write the logs in the Log Analytics workspace you have to select **Send to Log Analytics workspace** and select your workspace. Before you can save you have to enter a name.

Analytics with Power BI

For those looking to create more advanced analytics and visualizations, integrating Intune with **Power BI** through the **Intune Data Warehouse** offers powerful capabilities for custom dashboards and reports. Power BI allows you to visualize data, track compliance trends over time, and gain insights into device usage and management at a more strategic level.

One of the advantages of using Power BI with the Intune Data Warehouse is that it provides access to **historical data**, allowing you to analyze trends over extended periods. This is particularly useful for tracking compliance changes, software deployment success rates, and device performance metrics over time.

Connecting Power BI to the Intune Data Warehouse

To connect Power BI to the Intune Data Warehouse, follow these steps:

1. **Retrieve the Data Warehouse URL**: In the **Intune Admin Center**, go to **Reports > Data Warehouse**. Copy the URL for your Intune Data Warehouse.
2. **Open Power BI Desktop**: Start Power BI Desktop and select **Get Data**.
3. **Connect to OData Feed**: In the **Get Data** window, select **OData Feed** and paste the Data Warehouse URL you copied earlier.
4. **Authenticate**: Sign in with your Intune admin credentials to access the data.
5. **Select Data**: Choose the datasets you want to import into Power BI, such as Devices, CompliancePolicies, or SoftwareInventory.

Creating a Basic Power BI Dashboard

Once your data is imported into Power BI, you can begin creating visualizations and dashboards. Here's a basic example to get you started:

1. **Create a Compliance Overview**: Use the DeviceCompliance dataset to create a bar chart that shows the number of compliant vs. noncompliant devices. This gives you a quick snapshot of your organization's compliance posture.
2. **Track Compliance Over Time**: Use a line chart to visualize how device compliance has changed over the past month. This can help you identify trends or recurring issues with device configurations.
3. **Device Inventory Breakdown**: Create a pie chart to categorize your device inventory by operating system, giving you insights into the diversity of your managed devices.

Power BI's drag-and-drop interface makes it easy to build these visualizations, and its powerful analytics engine allows you to dig deeper into the data to uncover insights that can drive strategic decisions.

By combining the real-time capabilities of Azure Monitor with the advanced analytics and historical insights provided by Power BI, Intune empowers organizations to create a comprehensive and forward-looking device management strategy.

Advanced Endpoint Analytics in Intune

Understanding Endpoint Analytics vs. Intune Advanced Analytics

Endpoint Analytics is a foundational feature within Microsoft Intune that provides valuable insights into device performance, application reliability, and user productivity. With Endpoint Analytics, IT administrators can proactively detect and resolve issues impacting employee experience across both on-premises and remote/hybrid environments. It focuses on helping organizations understand why users might be experiencing slow startup times, which applications are unstable, and how well remote devices are performing. The core objective is to enhance user productivity by identifying and remediating problems before they significantly affect day-to-day operations.

Intune Advanced Analytics, on the other hand, builds upon this foundation by offering enhanced, data-driven capabilities that further streamline troubleshooting, deliver more granular insights, and optimize the IT support process. Advanced Analytics is available as an add-on to Intune and provides extended visibility, faster data collection, and additional reporting areas—such as battery health and detailed device timelines—that aren't available in the base product. In short, while Endpoint Analytics sets the stage by giving you essential insights and remediation tools, Intune Advanced Analytics takes these capabilities to the next level, helping you more proactively manage and improve the end-user experience.

Key Components of Base Endpoint Analytics

1. **Startup Performance:**

- **What it does:** Monitors how quickly devices boot up and become usable.
- **Why it matters:** Slow startup times directly affect user productivity. Endpoint Analytics pinpoints which devices experience delays, enabling IT to implement targeted improvements.

2. **Proactive Remediations:**
    - **What it does:** Runs custom scripts on devices to detect and fix common issues automatically, such as clearing temporary files or adjusting incorrect configurations.
    - **Why it matters:** This helps maintain device health and minimizes disruptions for end-users by resolving problems silently in the background.
    - **Hint:** This feature was moved out to device section

3. **Application Reliability:**
    - **What it does:** Tracks the performance and stability of installed applications.
    - **Why it matters:** Identifies frequently crashing or malfunctioning apps so IT teams can address them—either by updating, replacing, or further investigating—to improve overall user experience.

4. **Work from Anywhere:**
    - **What it does:** Provides insights into the connectivity and performance of devices used remotely.
    - **Why it matters:** Evaluates VPN usage, Wi-Fi quality, and other network factors so administrators understand and optimize the remote user experience.

5. **Resource Performance:**
    - **What it does:** Monitors CPU and RAM usage, disk performance, and other device resource metrics.
    - **Why it matters:** High resource utilization can slow down device performance. By identifying resource-heavy processes, IT can take action to balance workloads or upgrade hardware where needed.

6. **Remoting Connection:**
    - **What it does:** Provides insights into remote desktop connection quality and responsiveness.
    - **Why it matters:** Ensures that users relying on remote desktop sessions receive a smooth and stable experience, especially in distributed work scenarios.

Additional Capabilities with Intune Advanced Analytics

Intune Advanced Analytics extends these foundational features with richer diagnostics, more responsive data, and deeper insights. Key enhancements include:

1. **Battery Health:**
    - **What it does:** Offers insights into battery performance and degradation over time.
    - **Why it matters:** Proactive battery health monitoring allows organizations to plan hardware refreshes or configure power settings to improve device longevity and user satisfaction.

2. **Custom Device Scopes:**
    - **What it does:** Allows filtering and segmenting Endpoint Analytics reports by scope tags.
    - **Why it matters:** Administrators can hone in on specific device groups—such as those in a particular region, department, or managed by a specific IT team—to get more focused recommendations and insights.

3. **Enhanced Device Timeline:**

- **What it does:** Provides a richer, more detailed timeline of device events with lower data latency.
- **Why it matters:** Faster and more granular event data means quicker troubleshooting. IT can pinpoint when and why an issue started, resulting in faster resolution times.

4. **Anomalies:**
    - **What it does:** Monitoring device health for regression after configuration changes.
    - **Why it matters:** IT can immediately understand the impact of policy changes or new configurations on user experience and roll back or adjust as needed.

5. **Device Query:**
    - **What it does:** Provides near real-time access to device state and configuration data.
    - **Why it matters:** Rapid data retrieval means IT administrators can quickly verify if applied changes are effective, troubleshoot issues more efficiently, and minimize user downtime.

How to Use Endpoint Analytics

To get started with Endpoint Analytics, you need to enable it within the Microsoft Intune admin center. Here’s a step-by-step guide:

1. **Enable Endpoint Analytics**:
   - Navigate to Microsoft Intune admin center.
   - Go to Reports > Endpoint Analytics.
   - Select Start to enable Endpoint Analytics for your organization.

2. **Access Insights**:
   - Once enabled, you can access various insights into device performance, startup times, and application reliability. These insights are presented in dashboards that allow you to drill down into specific metrics and devices.

3. **Create Remediations**:
   - Remediations is one of the most powerful features of Endpoint Analytics. This was in the past part of Endpoint Analytics but was moved to the device section. It allows you to create custom scripts that detect and automatically fix issues on devices. Here’s how to create and deploy a Proactive script:

Example Script: Clear Temporary Files

https://github.com/JayRHa/Book/blob/main/remediation/clearTmpFilesDetection.ps1

```
# Detect Script: Check if temp files are larger than 500MB
$TempPath = $env:TEMP
$TempSize = (Get-ChildItem $TempPath -Recurse | Measure-Object -Property Length -Sum).Sum /
1MB

if ($TempSize -gt 500) {
    Write-Host "Temp folder size is greater than 500MB."
    exit 1  # Non-zero exit code indicates remediation is needed
} else {
    Write-Host "Temp folder size is within acceptable limits."
    exit 0
}
```


```
# Detect Script: Check if temp files are larger than 500MB
$TempPath = $env:TEMP
$TempSize = (Get-ChildItem $TempPath -Recurse | Measure-
Object -Property Length -Sum).Sum / 1MB

if ($TempSize -gt 500) {
    Write-Host "Temp folder size is greater than 500MB."
    exit 1  # Non-zero exit code indicates remediation is
needed
} else {
    Write-Host "Temp folder size is within acceptable
limits."
    exit 0
}
```


https://github.com/JayRHa/Book/blob/main/remediation/clearTmpFilesRemeditation.ps1

```
# Remediation Script: Clear temp files
Remove-Item "$env:TEMP\*" -Recurse -Force
Write-Host "Temp files cleared."
```


```
# Remediation Script: Clear temp files
Remove-Item "$env:TEMP\*" -Recurse -Force
Write-Host "Temp files cleared."
```


Steps to Deploy Proactive Remediations:

1. Navigate to **Devices > Scripts and remediations > Remediations**.

2. Click + **Create**.
3. Upload the **Detect Script** and **Remediation Script**.
4. Assign the remediation to a group of devices.
5. Monitor the results in the **Proactive Remediations** dashboard.

Example Script: Fix Wi-Fi Connectivity Issues

https://github.com/JayRHa/Book/blob/main/remediation/fixWifiIssuesDetection.ps1

```
# Detect Script: Check if the device is experiencing Wi-Fi connection issues
$wifi = Get-NetAdapter | Where-Object {$_.Status -eq 'Up' -and $_.MediaType -eq 'Native
802.11'}
if ($wifi) {
    Write-Host "Wi-Fi is connected."
    exit 0
} else {
    Write-Host "Wi-Fi is not connected."
    exit 1  # Non-zero exit code indicates remediation is needed
}
```


```
# Detect Script: Check if the device is experiencing Wi-Fi
connection issues
$wifi = Get-NetAdapter | Where-Object {$_.Status -eq 'Up'
-and $_.MediaType -eq 'Native 802.11'}
if ($wifi) {
    Write-Host "Wi-Fi is connected."
    exit 0
} else {
    Write-Host "Wi-Fi is not connected."
    exit 1  # Non-zero exit code indicates remediation is
needed
}
```


https://github.com/JayRHa/Book/blob/main/remediation/fixWifiIssuesRemediation.ps1

```
# Remediation Script: Restart the Wi-Fi adapter
$wifiAdapter = Get-NetAdapter | Where-Object {$_.MediaType -eq 'Native 802.11'}
if ($wifiAdapter) {
    Restart-NetAdapter -Name $wifiAdapter.Name
    Write-Host "Wi-Fi adapter restarted."
}
```


```
# Remediation Script: Restart the Wi-Fi adapter
$wifiAdapter = Get-NetAdapter | Where-Object {$_.MediaType
-eq 'Native 802.11'}
if ($wifiAdapter) {
    Restart-NetAdapter -Name $wifiAdapter.Name
    Write-Host "Wi-Fi adapter restarted."
}
```


This script first checks if the Wi-Fi adapter is connected, and if not, it restarts the adapter to resolve connectivity issues.

4. **Monitor and Analyze**:
    - After deploying the remediation scripts, you can monitor their success in the **Remediations** section. This will show you how many devices required remediation, and whether the script successfully resolved the issue.

If you are searching for more remediation scripts, you can check out one of the largest remediation repositories created by MVP friends (Andrew Taylor, Florian Salzmann, Joey Verlinden) and me, but there was an immense contribution from the community. New contributions are always welcome.

Link: https://github.com/JayRHa/EndpointAnalyticsRemediationScripts

Device Queries

Now we came to the brand new and very nice stuff. I would name it an old friend. This feature is very well known from Microsoft Configuration Manager and is now also added to the Intune Suite via the Intune Advanced Analytics license. Intune Device Query can help you get more live insights from your endpoint devices.

What are device queries?

The Intune device query feature offers IT administrators a powerful way to quickly gain insights into the status and configurations of Windows devices. As part of the Intune Advanced Analytics suite, it enables the execution of Kusto queries for real-time data on device conditions, settings, and more. This capability is crucial for diagnosing device issues, verifying compliance, and collecting detailed inventory information.
With the use of the Kusto Query Language (KQL) (You can look in the section earlier in the book how it works), administrators can formulate custom queries that target a wide array of device attributes, including hardware, software, registry entries, and network configurations. By leveraging these queries, you can effectively access device information on demand, which helps in addressing security issues and troubleshooting.

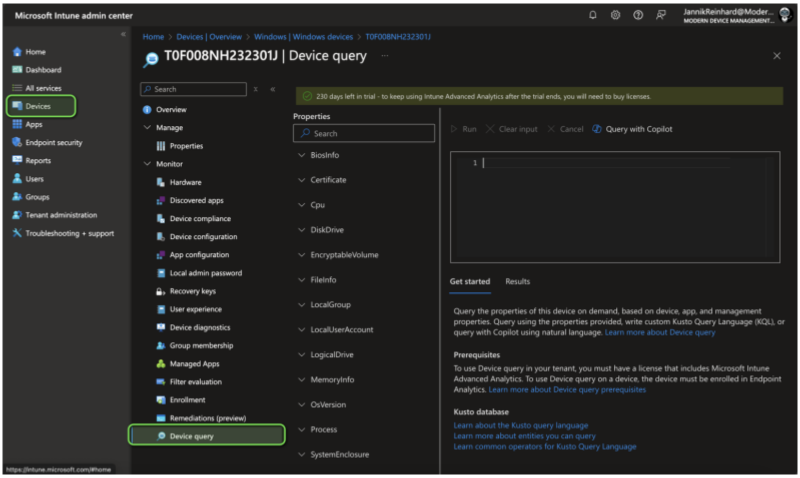
Prerequisites for Using Intune Device Query

To utilize Intune device query, you need an Intune Advanced Analytics Add-on or a Microsoft Intune Suite license. Licenses can be purchased or tried for 90 days (up to 250 users) via the Intune portal under Tenant Administration > Intune Add-ons.

Additionally, devices must be enrolled in both Intune and Endpoint Analytics, and the admin needs the appropriate Managed Devices – Query permission.

How to Run an Intune Device Query

1. Open the Intune Portal and go to Devices.
2. Select the desired device (Windows).
3. Choose the Device Query option.
4. Use KQL syntax to create and execute your queries.



Examples of Intune Device Queries

- **CPU Information**: Retrieve data about the device's CPU.

```
Cpu | project ProcessorId, CurrentClockSpeed,
MaxClockSpeed, CpuStatus
```

- **Windows Registry**: Obtain registry details.

```
WindowsRegistry('PATH_OF_YOUR_REGISTRY_KEY') | project
RegistryKey, ValueName, ValueType, ValueData
```

- **Windows Event Logs**: Fetch specific Windows event logs.

```
WindowsEvent('System', 7d) | where tostring(EventId) ==
'EVENT_ID' | project EventId, LoggedDateTime, LogName,
Message, ProviderName, WindowsUserAccount
```

These examples demonstrate how easy and powerful the Intune device query feature is, allowing administrators to utilize KQL to explore a wealth of device information.

Intune Device Query Operators

Table Operators

Table operators can be used to filter, summarize, and transform data streams. The following operators are currently supported:

Table operators	Description
`count`	Returns a table with a single record containing the number of records
`distinct`	Produces a table with the distinct combination of the provided columns of the input table

join	Merge the rows of two tables to form a new table by matching Rows for the same device
order by	Sort the rows of the input table into order by one or more columns
project	Select the columns to include, rename, or drop, and insert new computed columns
take	Return up to the specified number of rows
top	Returns the first N records sorted by the specified columns
where	Filters a table to the subset of rows that satisfy a predicate

Scalar Operators

Operators	Description	Example
==	Equal	1 == 1, ‘aBc’ == ‘AbC’
!=	Not Equal	1 != 2, ‘abc’ != ‘abcd’
<	Less	1 < 2, ‘abc’ < ‘DEF’
>	Greater	2 > 1, ‘xyz’ > ‘XYZ’

<=	Less or Equal	1 <= 2, ‘abc’ <= ‘abc’
>=	Greater or Equal	2 >= 1, ‘abc’ >= ‘ABC’
+	Add	2 + 1, now() + 1d
–	Subtract	2 – 1, now() – 1h
*	Multiply	2 * 2
/	Divide	2 / 1
%	Modulo	2 % 1
like	Left Hand Side (LHS) contains a match for Right Hand Side (RHS)	‘abc’ like ‘%B%’
!like	LHS doesn’t contain a match for RHS	‘abc’ !like ‘_d_’
contains	RHS occurs as a subsequence of LHS	‘abc’ contains ‘b’
!contains	RHS doesn’t occur in LHS	‘team’ !contains ‘i’
startswith	RHS is an initial subsequence of LHS	‘team’ startswith ‘tea’
!startswith	RHS isn’t an initial subsequence of LHS	‘abc’ !startswith ‘bc’

`endswith`	RHS is a closing subsequence of LHS	`‘abc’ endswith ‘bc’`
`!endswith`	RHS isn’t a closing subsequence of LHS	`‘abc’ !endswith ‘a’`
`and`	True if and only if RHS and LHS are true	`(1 == 1) and (2 == 2)`
`or`	True if and only if RHS or LHS is true	`(1 == 1) or (1 == 2)`

Aggregation Functions

Aggregation functions can be used with the summarize table operator to calculate summarized values. The following aggregation functions are currently supported:

Function	**Description**
`avg()`	Returns the average of the values across the group
`count()`	Returns a count of the records per summarization group
`countif()`	Returns a count of rows for which Predicate evaluates to true
`dcount()`	Returns the number of distinct values in the group
`max()`	Returns the maximum value across the group
`maxif()`	Starting in version 2107, you can use maxif with the summarize table operator.

	Returns the maximum value across the group for which *Predicate* evaluates to true.
`min()`	Returns the minimum value across the group
`minif()`	Starting in version 2107, you can use minif with the summarize table operator. Returns the minimum value across the group for which *Predicate* evaluates to true.
`percentile()`	Returns an estimate for the specified nearest-rank percentile of the population defined by Expr
`sum()`	Returns the sum of the values across the group
`sumif()`	Returns a sum of Expr for which Predicate evaluates to true

Scalar Functions

Scalar functions can be used in expressions. Currently, the following scalar functions are supported:

Function	**Description**
`ago()`	Subtracts the given timespan from the current UTC clock time
`bin()`	Rounds values down to many datetime multiple of a given bin size
`case()`	Evaluates a list of predicates and returns the first result expression whose predicate is satisfied

`datetime_add()`	Calculates a new datetime from a specified date part multiplied by a specified amount, added to a specified datetime
`datetime_diff()`	Calculates the difference between two datetime values
`iif()`	Evaluates the first argument and returns the value of either the second or third arguments depending on whether the predicate evaluated to true (second) or false (third)
`indexof()`	The function reports the zero-based index of the first occurrence of a specified string within the input string
`isnotnull()`	Evaluates its sole argument and returns a Boolean value indicating if the argument evaluates to a non-null value
`isnull()`	Evaluates its sole argument and returns a Boolean value indicating if the argument evaluates to a null value
`now()`	Returns the current UTC clock time
`strcat()`	Concatenates between 1 and 64 arguments
`strlen()`	Returns the length, in characters, of the input string
`substring()`	Extracts a substring from a source string starting from some index to the end of the string
`tostring()`	Converts input to a string representation

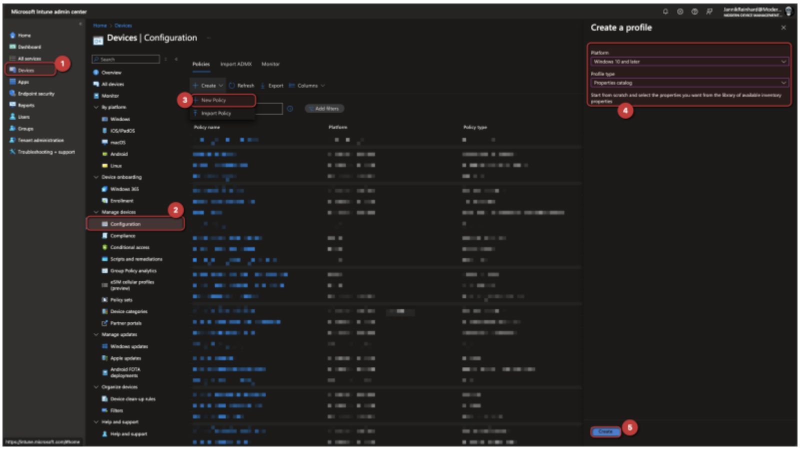
Properties catalog

Yes, you read correctly. Intune now provides a **powerful way to collect custom inventory**. This has been a long-awaited feature, and finally, it is here. The best part? It's an **Intune core feature**, and no additional licensing is required.

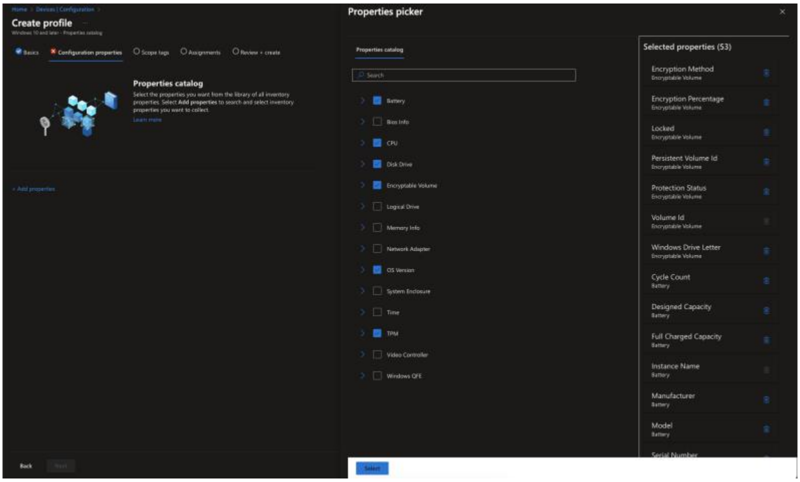
This exciting feature was announced at this year's **Microsoft Ignite** event. You can access it in the **Microsoft Intune Admin Center** under the **Device** section. To start collecting inventory from clients, you must first **create a collection rule**.

Create a Properties Catalog Profile:

- **Navigate** to the Microsoft Intune admin center: **Devices** > **Windows** > **Configuration**
- Click **Create** > **New policy**.
- Select **Windows 10 and later** > **Properties catalog**.
- Click **Create**.



- Enter **Name** for the profile and click **Next**.
- Click **Add settings** to browse and select the desired properties for the inventory and click **Next**.



- Assign the policy to the required **Users or Devices** on the **Assignments** page and click **Next**. Verify your settings and click **Create**.

Note: Applying this profile installs an additional agent on the assigned devices to perform inventory tasks.

How the Properties Catalog Works

The **Properties Catalog** profile enables you to enhance device inventory for **Windows devices** by collecting additional hardware attributes. This capability is powered by the **Intune Data Platform**. While the **Device Query** feature allows real-time querying, the **Device Inventory** feature gathers the same data for reporting and analysis purposes. This distinction enables organizations to gain deeper insights into their Windows fleet and make **data-driven decisions**.

The newly installed **Microsoft Device Inventory Agent** is responsible for gathering the hardware properties. The agent:

- Runs as a service: Microsoft Device Inventory Agent (InventoryService)
- Stores collected data locally in C:\Program Files\Microsoft Device Inventory Agent.

Verifying the Configuration

- Verify the **agent installation** on assigned devices:
    - Navigate to: C:\Program Files\Microsoft Device Inventory Agent.
    - Check for logs and verify the service is running.
- Confirm the collected data via the **Resource Explorer** in the Intune admin center:
    - Navigate to: **Devices** > **Monitor** > **Resource Explorer**.
    - The explorer will display the **hardware properties** configured in the Properties Catalog profile.

Here’s a completed and polished version of your text:

Access data via Microsoft Graph

The inventory data collected by Intune can be accessed programmatically via **Microsoft Graph**. This allows IT administrators to integrate the data into **automation workflows** for further analysis or reporting.

This is the graph call to get inventory information’s of a Device:

```
https://graph.microsoft.com/beta/deviceManagement/managedDevices('DEVICEID')/deviceInventories
```

Or for one collection topic:

```
https://graph.microsoft.com/beta/deviceManagement/managedDevices('DEVICEID')/deviceInventories('Battery')?$expand=instances($expand=Microsoft.Graph.deviceInventorySimpleItem/properties)
```

Automation in Intune: Streamlining IT Operations

Beyond reporting, Microsoft Intune offers robust automation capabilities that help organizations streamline their IT operations. Automation in Intune allows IT teams to reduce manual workload, minimize errors, and ensure that processes are consistent and repeatable.

Some key areas where Intune automation shines include:

- **Policy Enforcement:** Intune can automatically enforce security and compliance policies across all managed devices. For example, if a device falls out of compliance, Intune can automatically apply remediation actions or notify IT administrators to take corrective measures.

- **Application Deployment:** Automating the deployment of applications ensures that users always have access to the tools they need without requiring manual intervention from IT staff. Intune can schedule deployments, update apps, and even remove apps from devices when they are no longer needed.

- **Device Configuration:** Intune's ability to automatically apply device configurations—such as Wi-Fi settings, VPN profiles, and security protocols—reduces the time IT teams spend manually setting up and managing devices. This automation is especially valuable in large organizations with diverse device environments.

- **Zero-Touch Provisioning:** With features like Windows Autopilot, Intune enables zero-touch provisioning, allowing devices to be configured and deployed automatically without the need for IT to physically handle the device. This capability streamlines new device rollouts and ensures that devices are ready for use as soon as they arrive in the hands of employees.

As we move into the era of AI and more advanced automation, Intune's automation features provide the foundation for even more intelligent operations. Automating routine tasks frees up IT staff to focus on higher-level strategic initiatives, such as implementing new solutions to improve productivity and efficiency.

Chapter 5: Building Custom Analytics and Automation Solutions with Intune

Planning Your Custom Solution

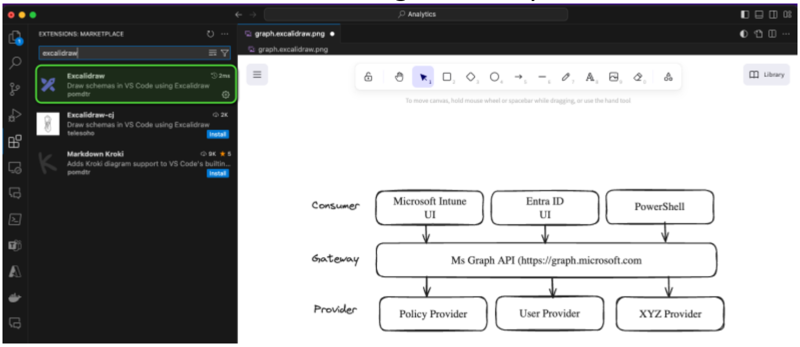
Before jumping into implementing custom analytics and automation solutions with Microsoft Intune, it's crucial to follow a structured approach to solution design. Proper planning ensures that your solution is scalable, maintainable, and aligned with your organization's goals. In this section, I will guide you through the process of designing a custom solution, along with some helpful tools and best practices.

Tools for Solution Design

When it comes to creating solution designs, visual representation is key to communicating ideas clearly and effectively. Here are two powerful tools that I personally use for creating design diagrams and graphics:

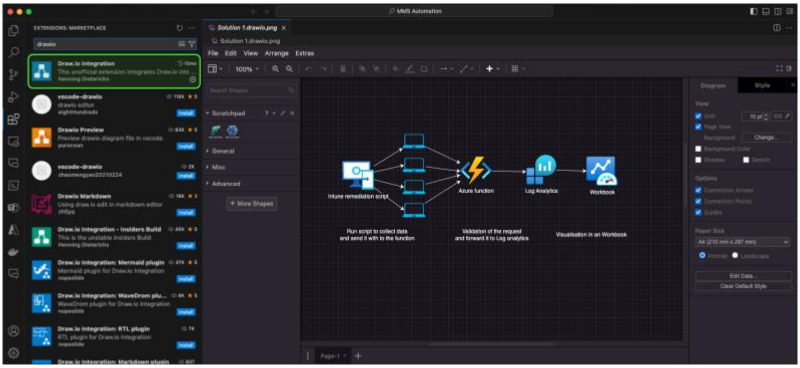
1. **Excalidraw**:
    - Excalidraw is a simple yet versatile tool that allows you to create "comic-like" graphics. I use it to create visuals for this book, and it’s great for sketching out ideas quickly.
    - One of the standout features of Excalidraw is its integration with Visual Studio Code (VS

Code). When you create a file with the extension *.excalidraw.png, you can directly edit and integrate it into your documentation (e.g., README.md files) without needing to leave your code editor.



2. **Draw.io (diagrams.net)**:

    - Draw.io is another fantastic tool that offers a wide range of icons, including Azure and Microsoft icons, making it ideal for designing technical architectures. Like Excalidraw, Draw.io also integrates with VS Code. You can create files with the extension *.drawio.png, making it easy to edit and version-control your diagrams directly within your development environment.



Both tools help you create visuals that clarify your architecture and solution design, making them easier to understand and implement.

Understanding Interfaces for Data Export and gathering

Before you start drawing or designing your solution, it’s important to have a solid understanding of the interfaces that Intune provides for exporting and interacting with data. While we've covered many of these in previous chapters, let’s briefly recap the most important ones:

1. **Microsoft Graph API**:
    - The Graph API is a versatile and powerful way to access and export data from Intune. It allows you to retrieve nearly all data available in the Intune UI, along with additional details that may not be directly accessible through the portal. This includes both standard Graph API calls and the

report-specific API for exporting larger datasets.

2. **Log Analytics Connector**:
   - Intune can forward logs to an Azure Log Analytics workspace. This is especially useful for collecting and analyzing audit logs, device information, and other operational data. It allows for advanced querying using Kusto Query Language (KQL) and enables deeper insights into your Intune-managed environment.

3. **Intune Data Warehouse**:
   - For organizations looking to create custom dashboards and reports, the Intune Data Warehouse provides an easy way to integrate with Power BI. It also offers historical data, which can be invaluable for tracking trends and long-term analysis.

4. **Remediations Scripts**:
   - Proactive remediations provide a way to run custom scripts on devices managed by Intune. These scripts can be used to fix issues, apply settings, and even collect data from devices. This is a highly flexible option for implementing targeted automation and ensuring that your devices remain in compliance with your organization's policies.

5. **Third-Party Agents**:

- While Microsoft offers robust native tools for data collection and automation, you may find that third-party agents can enhance your capabilities. These tools often come with pre-built analytics and dashboards, along with the ability to collect custom data that isn't available through Intune's native interfaces. However, they may introduce additional costs and load on your devices, so it's important to evaluate them carefully based on your needs.

With a clear understanding of the available interfaces, you can begin to map out how data will flow through your system, how it will be processed, and where it will be stored.

Working with, Processing, and Storing Data

Once you've identified how you will collect and export data, the next step is to plan how you will work with, process, and store that data. Azure offers a wide array of services that can help you build scalable and efficient data pipelines.
Here are some of the key Azure services you might consider for your custom solutions:

1. **Azure Storage**:
   - Azure Storage offers a variety of options for storing structured and unstructured data, including Blob Storage, File Storage, and Table Storage. It's a good choice for storing

raw data exported from Intune before processing or analysis.

2. **Azure SQL Database**:
    - If your solution requires structured data with relational capabilities, Azure SQL Database can provide a scalable cloud-based database service. It's ideal for storing processed data that needs to be queried or integrated with other systems.

3. **Azure Data Factory**:
    - For more complex data integration and processing workflows, Azure Data Factory provides a fully managed data integration service that can orchestrate data movement and transformation. It's well-suited for ETL (Extract, Transform, Load) processes where you need to collect data from multiple sources, transform it, and load it into your data warehouse or analytics platform.

4. **Azure Functions**:
    - Azure Functions enable you to run serverless code in response to events. You can use Azure Functions to automate processing tasks, such as triggering data transformations or sending notifications based on specific conditions detected in your data.

5. **Azure Synapse Analytics**:

- Azure Synapse Analytics is a powerful analytics service that brings together big data and data warehousing. If you need to perform complex analytics on large datasets, Synapse can provide the compute power and scalability needed to process and analyze your data efficiently.

6. **Power BI**:
   - For visualization and reporting, Power BI is an excellent choice. It integrates with various data sources, including the Intune Data Warehouse, and allows you to create interactive dashboards and reports. Power BI can help you track key performance indicators (KPIs), compliance metrics, and other important data points.

7. **Microsoft Fabric:**
   - Microsoft Fabric is an end-to-end analytics platform that unifies data engineering, data integration, data warehousing, data science, real-time analytics, and business intelligence into a single SaaS solution. It provides a consistent, integrated experience for working with data across Microsoft's ecosystem, including Azure, Power Platform, and Microsoft 365. Fabric is designed to simplify data workflows, enhance productivity, and support seamless collaboration. It's particularly useful for organizations looking to create a modern data architecture without managing multiple separate tools or infrastructures.

8. **Azure Machine Learning**:
    - If your solution involves predictive analytics or AI-driven insights, Azure Machine Learning provides a comprehensive environment for building, training, and deploying machine learning models. You can use it to analyze patterns in your data and generate forecasts or anomaly detection alerts.

But there are many more. This is only a small subset.

Designing Your Solution: Step-by-Step Process

Now that you have an overview of the tools and interfaces available, let's outline the steps for designing a custom solution with Intune:

1. **Define Your Objectives**:
    - Start by clearly defining the problem you want to solve or the insight you want to gain. Are you looking to automate compliance enforcement? Do you need detailed reporting on device health? Understanding your goals will guide the rest of the design process.

2. **Choose Data Sources**:
    - Based on your objectives, identify the data sources you need to work with. This could include data from Graph API, Log Analytics, or the Intune Data Warehouse. Make sure

you understand the type of data each source provides and how it will contribute to your solution.

3. **Plan Data Collection and Export**:
   - Determine how you will collect and export data from Intune. Will you use Graph API calls to retrieve specific data sets, or will you configure Log Analytics to forward logs to a workspace? Consider whether you need real-time data or if periodic batch exports will suffice.

4. **Design Data Processing Workflows**:
   - Plan how you will process the data once it's collected. Do you need to clean, transform, or aggregate the data before analysis? Azure Data Factory or Azure Functions can help automate these tasks. Also, consider how you will handle data storage—whether in Azure Storage, a SQL database, or another solution.

5. **Build Automation Pipelines**:
   - If your solution involves automating specific tasks, such as applying remediation scripts or enforcing compliance policies, plan your automation pipelines. You may use Azure Logic Apps, Azure Functions, or Intune's built-in automation features to achieve this.

6. **Design Visualizations and Reports**:
   - If reporting and visualization are part of your solution, design dashboards in Power

BI that present the data in an actionable way. Consider the KPIs that matter to your stakeholders and ensure your visualizations are easy to interpret.

7. **Test and Iterate**:
    - Build a prototype of your solution and test it with a subset of your data. Ensure that it meets your objectives and performs efficiently. Make adjustments as necessary before scaling it out to your entire environment.

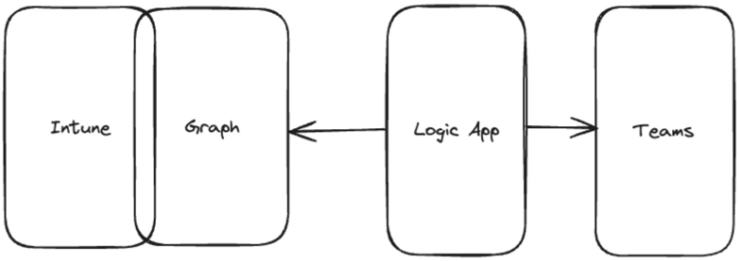
8. **Deploy and Monitor**:
    - Once your solution is finalized, deploy it to your production environment. Set up monitoring to ensure that everything runs smoothly and that you are alerted to any issues or anomalies that arise.

By following this structured approach, you can design and implement custom analytics and automation solutions with Intune that meet your organization’s unique needs. In the next sections, we’ll dive deeper into specific implementation examples and advanced integrations, helping you bring your designs to life.

Solution 1: The Power of Microsoft Graph and Logic Apps

In Chapter 4, we explored how Microsoft Graph works and how to get started, including exporting reports from the UI and via PowerShell. Now, let's dive deeper into building your own reporting solutions using Microsoft Graph. I will provide example implementations that you can use directly in your environment.

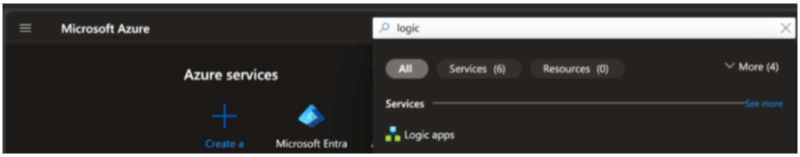
In this example, we will demonstrate how to use data from Microsoft Graph in combination with a Logic App to send daily reports showing the current device count.
This is a simple example, but the principles can be applied to many use cases to gain an overview of the current tenant state.



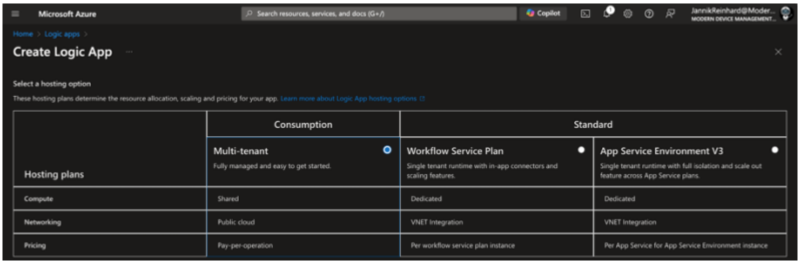
Step 1: Set Up a Logic App

1. **Open the Azure Portal**
   - Navigate to the [Azure Portal] and log in with your credentials.

2. **Create a Logic App**
   - In the portal, search for "Logic App" and select the Logic Apps service.

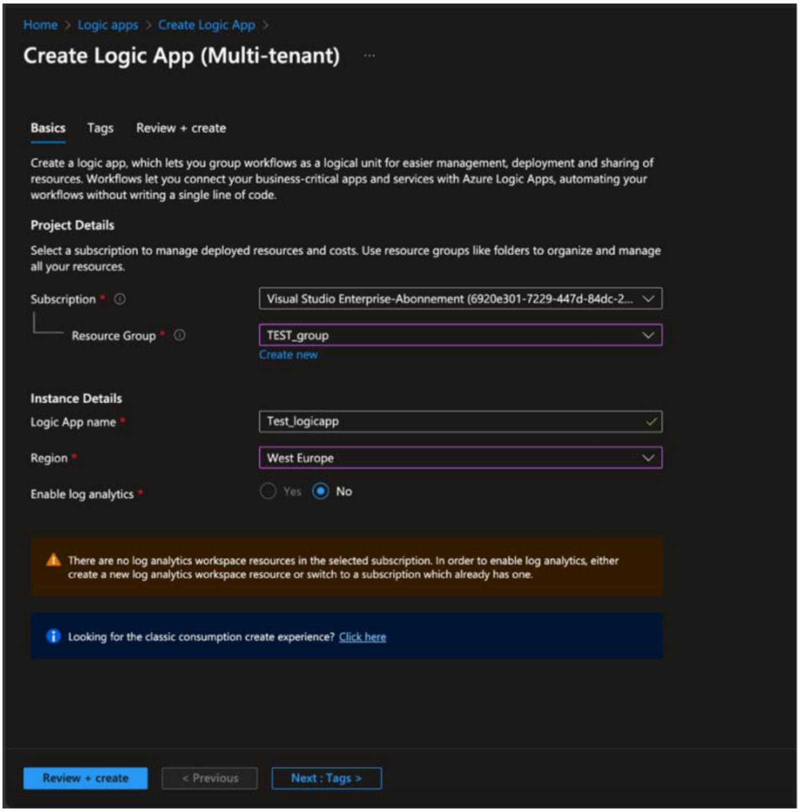


3. **Select the Plan**
   - Choose the plan you want to use. For initial development, I recommend the Consumption Plan.



Step 2: Configure Your Azure Logic App

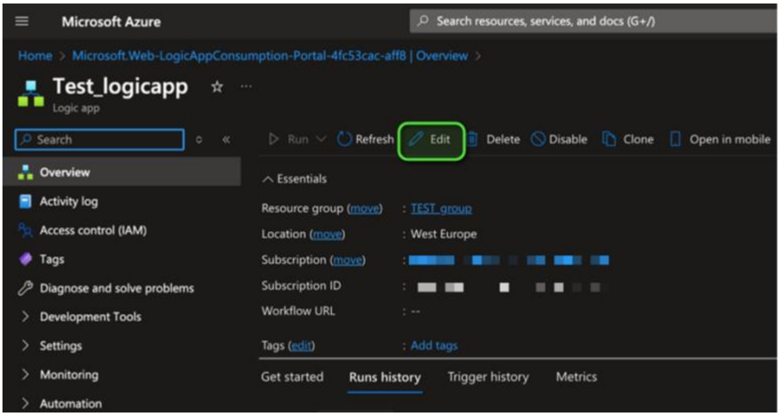
1. **Resource Group**
   - Choose an existing Resource Group or create a new one.
   - Provide a name for your Logic App (e.g., “Test_logicapp”).
   - Select your preferred region.
   - Click Review + Create, then click Create to deploy your Logic App.



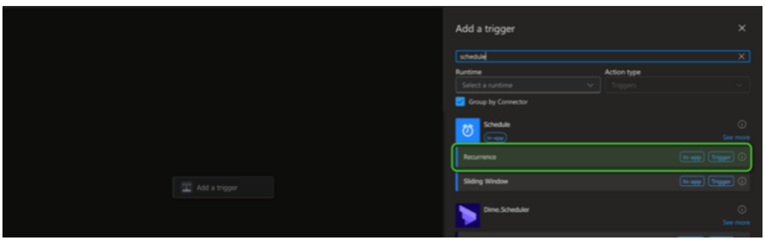
Step 3: Define Your Logic App Workflow

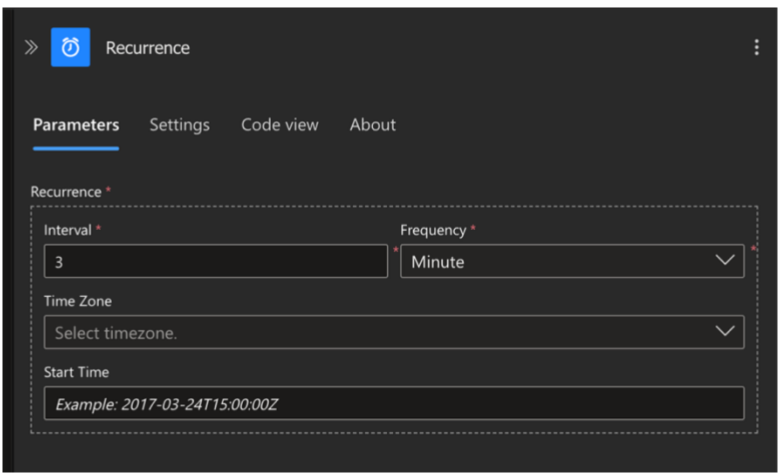
1. Edit the Workflow

- After deployment, navigate to your newly created Logic App resource.
- Click Edit to define the workflow.



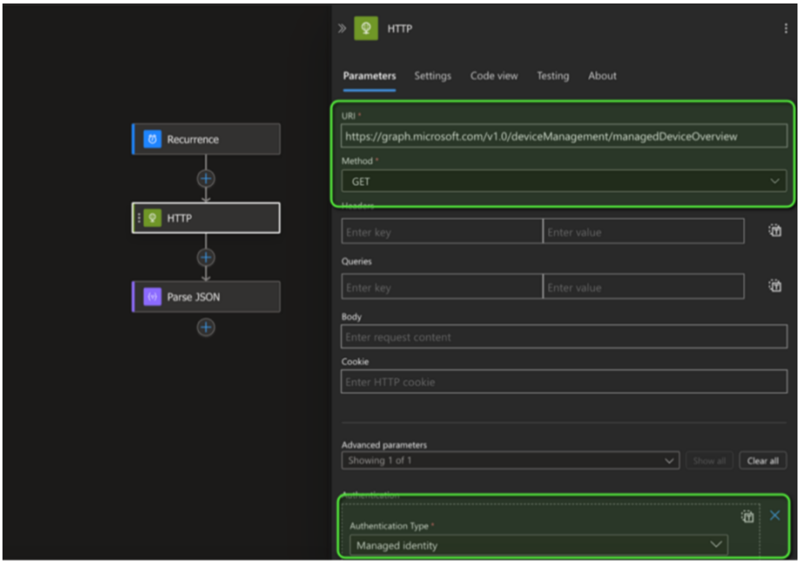
- Select Recurrence as the trigger to define the interval at which the Logic App will run.





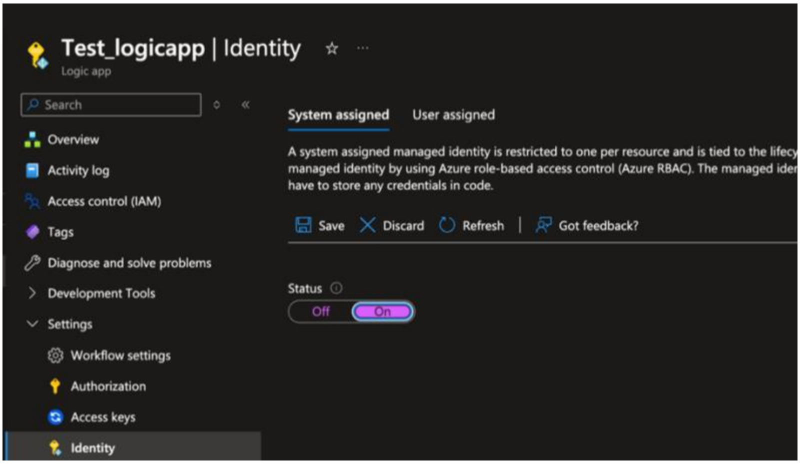
2. **Add an Action**
   - To send a daily email with the count of your devices, add an HTTP Action with the following Graph URL as a GET request:

```
https://graph.microsoft.com/v1.0/deviceManagement/managedDeviceOverview
```



Step 4: Add Permissions

1. **Assign Managed Identity Permissions**
   - To make this Graph call, add permissions to your managed identity in the Logic App settings under Identity.
   - Turn on the system-assigned identity.



2. **Use the Azure PowerShell**
    - Execute the following script in Azure PowerShell to assign permissions:

https://github.com/JayRHa/Book/blob/main/addMIpermissions.ps1

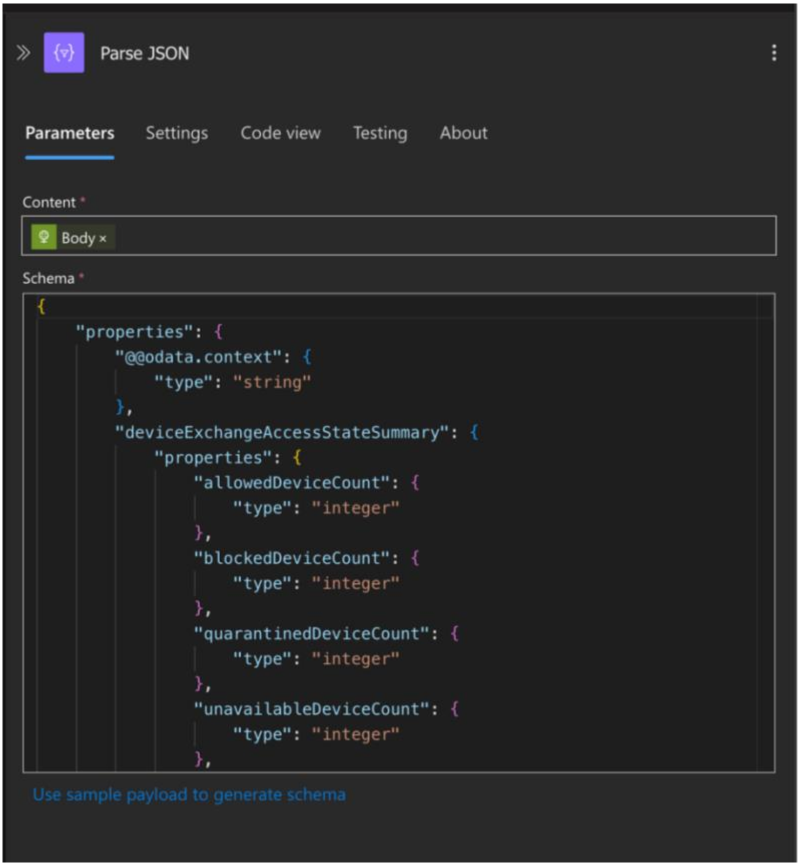
```
Install-Module Microsoft.Graph -Scope CurrentUser

Connect-MgGraph -Scopes Application.Read.All, AppRoleAssignment.ReadWrite.All,
RoleManagement.ReadWrite.Directory

$managedIdentityId = "Managed Identity Object ID"
$roleName = "DeviceManagementManagedDevices.Read.All"

$msgraph = Get-MgServicePrincipal -Filter "AppId eq '00000003-0000-0000-c000-000000000000'"
$role = $Msgraph.AppRoles| Where-Object {$_.Value -eq $roleName}

New-MgServicePrincipalAppRoleAssignment -ServicePrincipalId $managedIdentityId -PrincipalId
$managedIdentityId -ResourceId $msgraph.Id -AppRoleId $role.Id

Disconnect-MgGraph
```

```
Install-Module Microsoft.Graph -Scope CurrentUser

Connect-MgGraph -Scopes Application.Read.All,
AppRoleAssignment.ReadWrite.All,
RoleManagement.ReadWrite.Directory
```

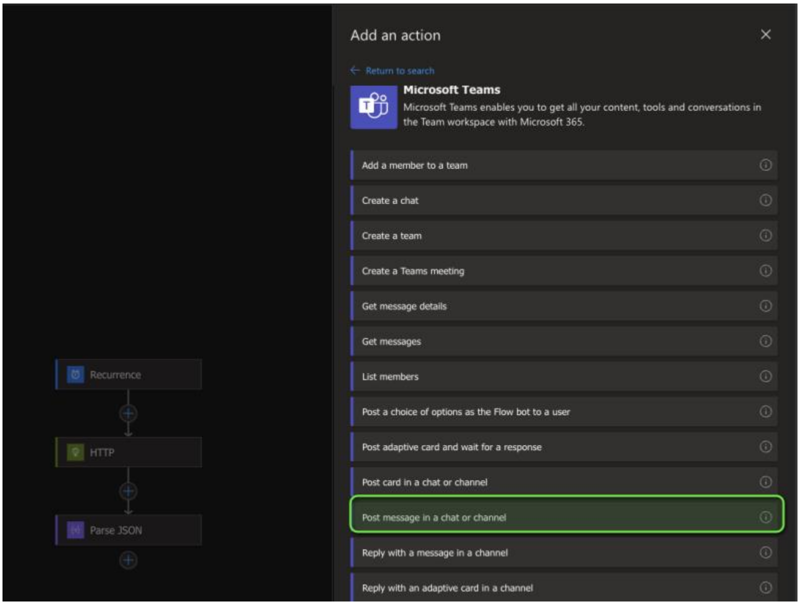
```
$managedIdentityId = "Managed Identity Object ID"
$roleName = "DeviceManagementManagedDevices.Read.All"

$msgraph = Get-MgServicePrincipal -Filter "AppId eq
'00000003-0000-0000-c000-000000000000'"
$role = $Msgraph.AppRoles| Where-Object {$_.Value -eq
$roleName}

New-MgServicePrincipalAppRoleAssignment -
ServicePrincipalId $managedIdentityId -PrincipalId
$managedIdentityId -ResourceId $msgraph.Id -AppRoleId
$role.Id

Disconnect-MgGraph
```

Step 5: Parse the Output and Send the Email

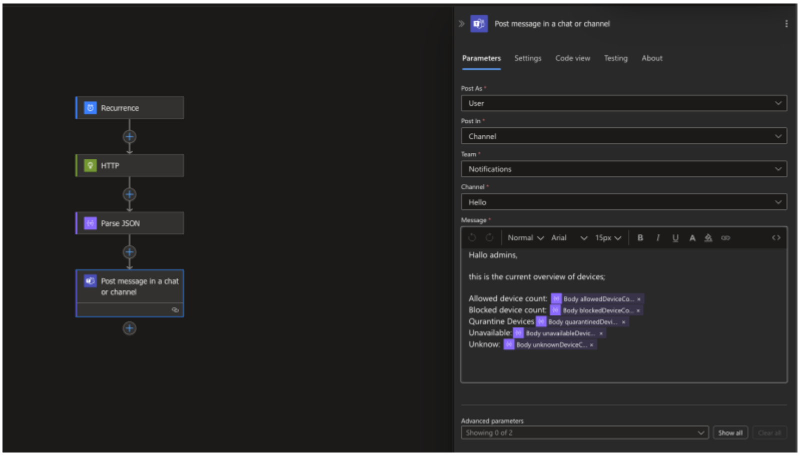
1. **Parse JSON**
    - Parse the JSON response from the API to use in the email.
    - Click on Use sample payload to generate a schema from the results of a call to this API in Graph Explorer.



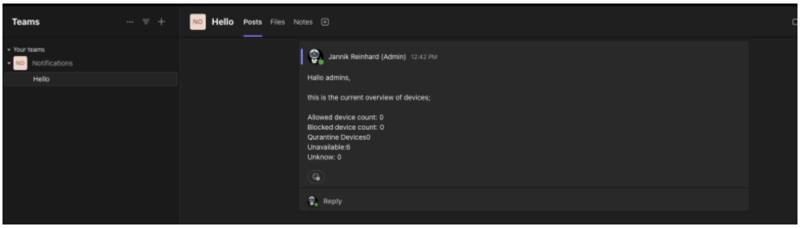
2. **Send a Teams Message**
   - To send a message via Teams, select Post message in a chat or channel and sign in with your account.



- Select whether you want to send the message on behalf of a bot or a user. Generate the message using the parameters from the parse action.



This is how the message can look like, but you can use every other graph call or a mix of multiple ones to generate a message:



Solution 2: The Power of Azure Automation Runbooks

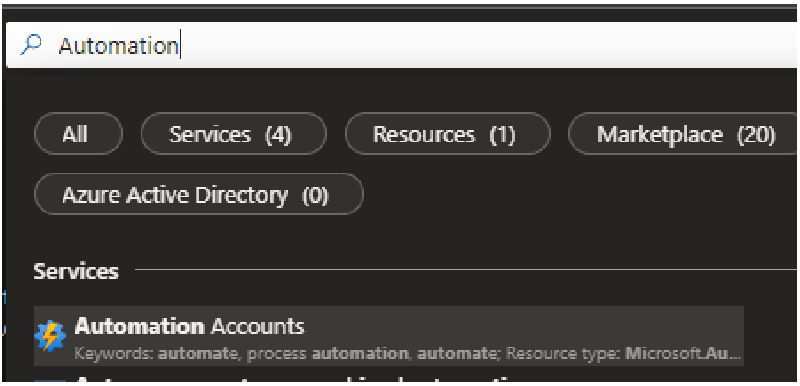
Another powerful service is Azure Automation. In this example, I will show you how to get a daily email with the current update status of your devices.

Overview of Azure Automation

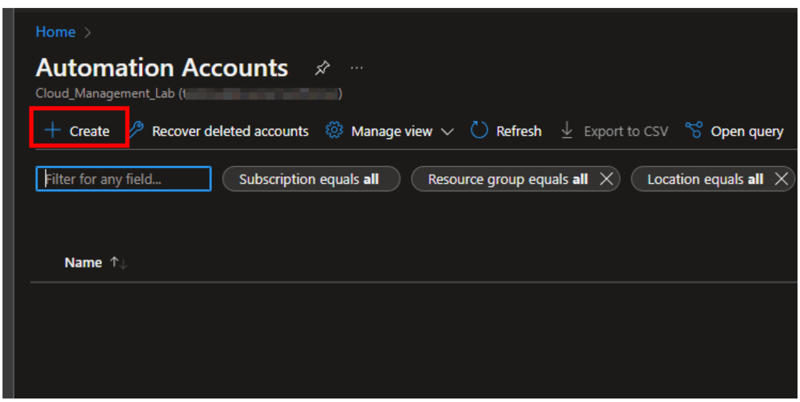
Before setting up an automation account, let me explain what it is. Azure Automation is a service that helps automate repetitive and time-consuming tasks across Azure and on-premises environments. Azure Automation Runbooks are an essential component of this service, allowing you to create, run, and manage scripts that automate processes—such as managing Intune devices, performing cleanups, or creating groups—using the Microsoft Graph API. Runbooks can be written in PowerShell, Python, or Bash, making it easy to create automated solutions tailored to your specific needs.

Step 1: Create an Automation Account

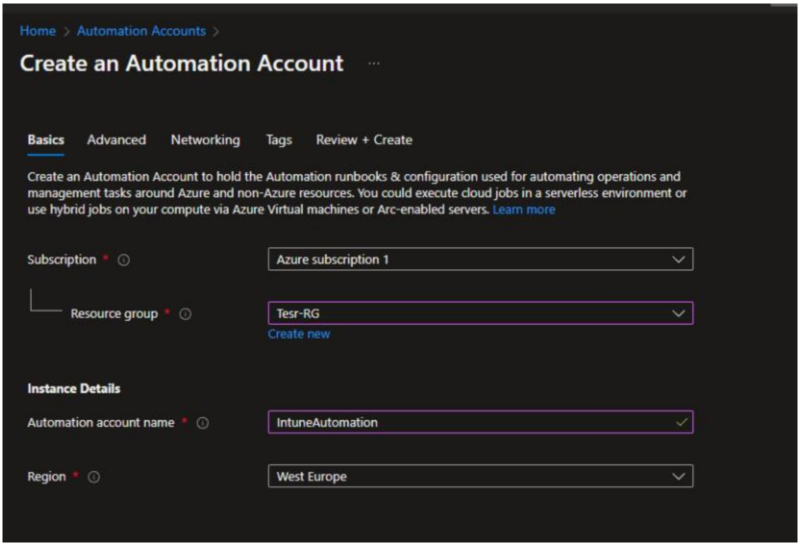
1. **Search for Automation Accounts**
    - Search for "**Automation Accounts**" in the Azure Portal.



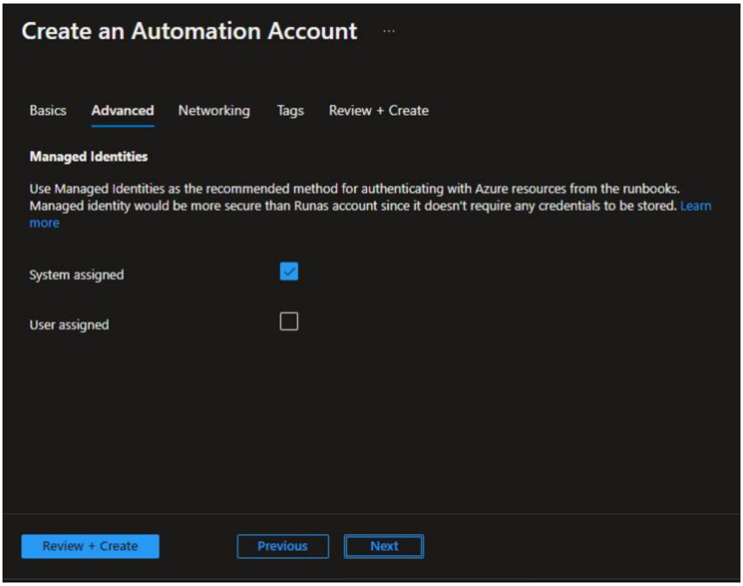
2. **Create the Account**
    - Click **+ Create**.



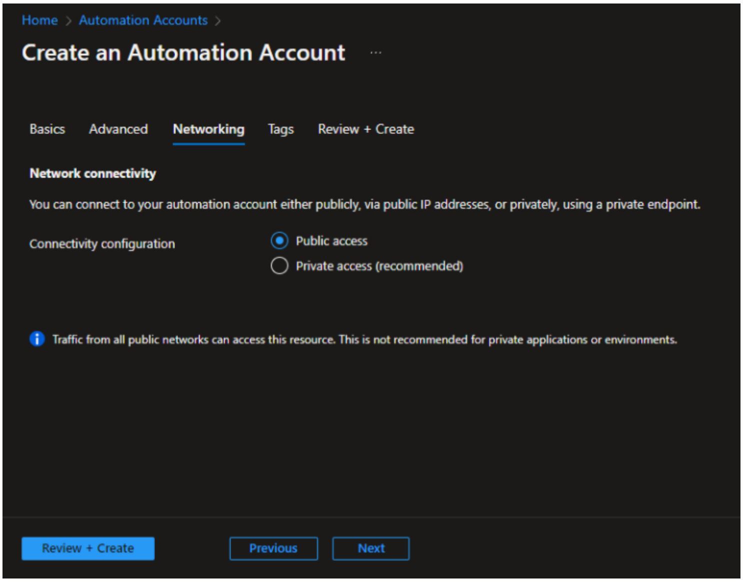
    - Select a **subscription** and a **resource group.**
    - Enter an **account name** and select a **region**.



- Click **Next** and activate the **System assigned Managed Identity**

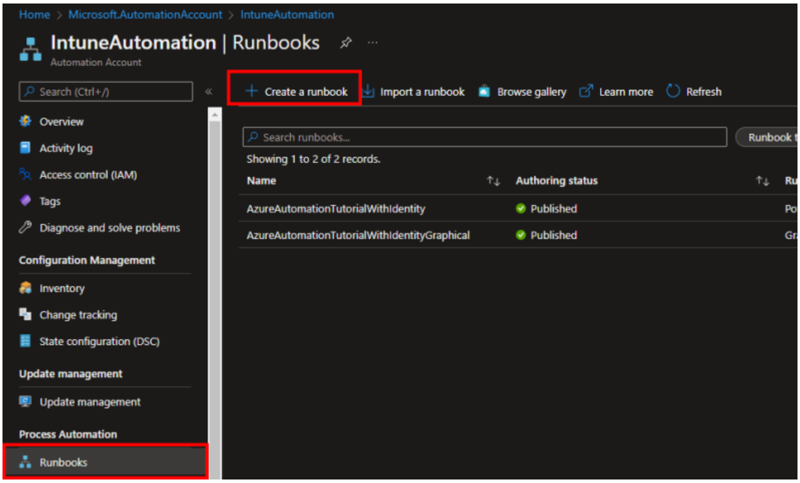


- Proceed through the steps until you can click **Review + Create**.



Step 2: Create the Runbook

1. **Open the Automation Account**
   - Select **Runbooks**
   - Click **+ Create a runbook**.



- Enter a name.
- Select **PowerShell** as the Runbook type.
- Select the runtime version.
- Click **Create**.

The next step is like the Logic app we have to use the same script to add permissions to the managed identity:

3. Use the Azure PowerShell

- Execute the following script in Azure PowerShell to assign permissions:

https://github.com/JayRHa/Book/blob/main/addMIpermissions.ps1

```
Install-Module Microsoft.Graph -Scope CurrentUser

Connect-MgGraph -Scopes Application.Read.All, AppRoleAssignment.ReadWrite.All,
RoleManagement.ReadWrite.Directory

$managedIdentityId = "Managed Identity Object ID"
$roleName = " Mail.Send"

$msgraph = Get-MgServicePrincipal -Filter "AppId eq '00000003-0000-0000-c000-000000000000'"
$role = $Msgraph.AppRoles| Where-Object {$_.Value -eq $roleName}

New-MgServicePrincipalAppRoleAssignment -ServicePrincipalId $managedIdentityId -PrincipalId
$managedIdentityId -ResourceId $msgraph.Id -AppRoleId $role.Id

Disconnect-MgGraph
```


```
Install-Module Microsoft.Graph -Scope CurrentUser

Connect-MgGraph -Scopes Application.Read.All,
AppRoleAssignment.ReadWrite.All,
RoleManagement.ReadWrite.Directory

$managedIdentityId = "Managed Identity Object ID"
$roleName = " Mail.Send"

$msgraph = Get-MgServicePrincipal -Filter "AppId eq
'00000003-0000-0000-c000-000000000000'"
$role = $Msgraph.AppRoles| Where-Object {$_.Value -eq
$roleName}

New-MgServicePrincipalAppRoleAssignment -
ServicePrincipalId $managedIdentityId -PrincipalId
$managedIdentityId -ResourceId $msgraph.Id -AppRoleId
$role.Id

Disconnect-MgGraph
```


- Do the same again with the “DeviceManagementManagedDevices.Read.All” permissions

The next thing we must do is to add this script to the automation account and fill out the MailTo and MailSender variables.

```
# Variables
$MailTo = ""
$MailSender = ""

# Authenticate and connect to Microsoft Graph
Connect-AzAccount -Identity
$token = Get-AzAccessToken -ResourceUrl "https://graph.microsoft.com"

#Connect to Microsoft Graph API
Connect-MgGraph -AccessToken $token.Token

# Get Windows devices from Intune
$devices = Get-MgDeviceManagementManagedDevice -all -Filter
"contains(operatingSystem,'Windows')"

# Filter Windows 11 devices
$windows11Devices = $devices | Where-Object { $_.OsVersion -ge "10.0.22000" }

# Calculate device counts
$totalDevices = $devices.Count
$windows11DevicesCount = $windows11Devices.Count

# Calculate pie chart percentages
$windows11Percentage = ($windows11DevicesCount / $totalDevices) * 100
$otherWindowsPercentage = 100 - $windows11Percentage

# Create HTML report
$html = @"
<style>
    body {
        font-family: Arial, sans-serif;
    }
    h1 {
        font-size: 28px;
        color: #0078d4;
        margin-top: 0;
        text-align: center;
    }
    .chart-container {
        display: flex;
        flex-wrap: wrap;
        justify-content: center;
    }
    .pie-chart {
        width: 300px;
        height: 300px;
        margin: 20px;
    }
    .device-list {
        margin-top: 40px;
    }
    table {
        border-collapse: collapse;
        width: 100%;
    }
    th {
        background-color: #0078d4;
        color: #fff;
        font-weight: bold;
        padding: 8px;
        text-align: left;
    }
    td {
        border: 1px solid #ddd;
        padding: 8px;
    }
    tr:nth-child(even) {
        background-color: #f2f2f2;
    }
</style>
<h1>Windows 11 Adoption Report</h1>
<div class="chart-container">
    <div class="pie-chart">
        <canvas id="chart"></canvas>
    </div>
</div>
<div class="device-list">
    <table>
        <thead>
            <tr>
                <th>Device Name</th>
                <th>User</th>
                <th>Last Sync DateTime</th>
                <th>OSVersion</th>
            </tr>
        </thead>
        <tbody>
"@
```

```
foreach ($device in $devices) {
    $html += "<tr><td>$($device.DeviceName)</td><td>$($device.EmailAddress)</td>
<td>$($device.LastSyncDateTime)</td><td>$($device.OSVersion)</td></tr>"
}
$html += @"
        </tbody>
    </table>
</div>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
    var ctx = document.getElementById('chart').getContext('2d');
    var chart = new Chart(ctx, {
        type: 'pie',
    data: {
            labels: ['Windows 11', 'Other Windows'],
            datasets: [{
            backgroundColor: ['#0078d4', '#f2f2f2'],
            data: [$windows11Percentage, $otherWindowsPercentage]
            }]
            },
            options: {
            legend: {
            display: true,
            position: 'bottom',
            labels: {
            fontColor: '#333',
            fontSize: 14,
            padding: 16
            }
            }
            }
            });
            </script>
"@

$html | Out-File -FilePath "Windows11AdoptionReport.html"
$base64 = [Convert]::ToBase64String([IO.File]::ReadAllBytes(".\Windows11AdoptionReport.html"))

#Send Mail
$URLsend = "https://graph.microsoft.com/v1.0/users/$MailSender/sendMail"
$BodyJsonsend = @"
{
    "message": {
      "subject": "Intune error report",
      "body": {
        "contentType": "Text",
        "content": "Dear Admin, this Mail contains the Windows 11 Adoption report"
      },
      "toRecipients": [
        {
          "emailAddress": {
            "address": "$MailTo"
          }
        }
      ],
      "attachments": [
        {
          "@odata.type": "#microsoft.graph.fileAttachment",
          "name": "Windows11AdoptionReport.html",
          "contentType": "text/plain",
          "contentBytes": "$base64"
        }
      ]
    }
  }
"@

$response = Invoke-MgRestMethod -Method POST -Uri $URLsend -Body $BodyJsonsend
```

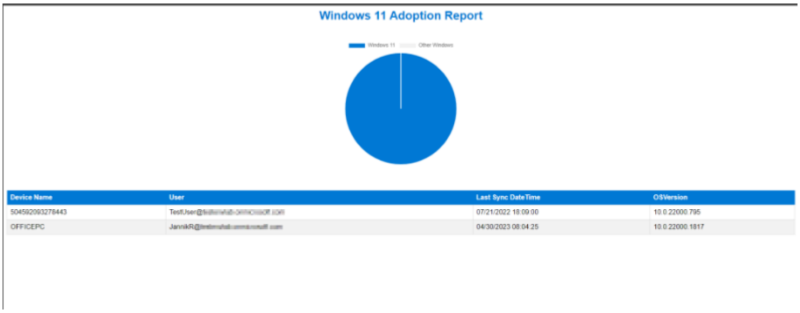
```
# Variables
$MailTo = ""
$MailSender = ""

# Authenticate and connect to Microsoft Graph
Connect-AzAccount -Identity
$token = Get-AzAccessToken -ResourceUrl
"https://graph.microsoft.com"

#Connect to Microsoft Graph API
Connect-MgGraph -AccessToken $token.Token

# Get Windows devices from Intune
$devices = Get-MgDeviceManagementManagedDevice -all -
Filter "contains(operatingSystem,'Windows')"

# Filter Windows 11 devices
$windows11Devices = $devices | Where-Object { $_.OsVersion
-ge "10.0.22000" }

# Calculate device counts
$totalDevices = $devices.Count
$windows11DevicesCount = $windows11Devices.Count

# Calculate pie chart percentages
$windows11Percentage = ($windows11DevicesCount /
$totalDevices) * 100
$otherWindowsPercentage = 100 - $windows11Percentage

# Create HTML report
$html = @"
<style>
    body {
        font-family: Arial, sans-serif;
    }
    h1 {
        font-size: 28px;
        color: #0078d4;
        margin-top: 0;
        text-align: center;
    }
    .chart-container {
        display: flex;
        flex-wrap: wrap;
        justify-content: center;
    }
    .pie-chart {
```

```
        width: 300px;
        height: 300px;
        margin: 20px;
    }
    .device-list {
        margin-top: 40px;
    }
    table {
        border-collapse: collapse;
        width: 100%;
    }
    th {
        background-color: #0078d4;
        color: #fff;
        font-weight: bold;
        padding: 8px;
        text-align: left;
    }
    td {
        border: 1px solid #ddd;
        padding: 8px;
    }
    tr:nth-child(even) {
        background-color: #f2f2f2;
    }
</style>
<h1>Windows 11 Adoption Report</h1>
<div class="chart-container">
    <div class="pie-chart">
        <canvas id="chart"></canvas>
    </div>
</div>
<div class="device-list">
    <table>
        <thead>
            <tr>
                <th>Device Name</th>
                <th>User</th>
                <th>Last Sync DateTime</th>
                <th>OSVersion</th>
            </tr>
        </thead>
        <tbody>
"@
foreach ($device in $devices) {
    $html +=
"<tr><td>$($device.DeviceName)</td><td>$($device.EmailAddr
```

```
ess)</td><td>$($device.LastSyncDateTime)</td><td>$($device
.OSVersion)</td></tr>"
}
$html += @"
        </tbody>
    </table>
</div>
<script
src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>
    var ctx =
document.getElementById('chart').getContext('2d');
    var chart = new Chart(ctx, {
        type: 'pie',
    data: {
            labels: ['Windows 11', 'Other Windows'],
            datasets: [{
            backgroundColor: ['#0078d4', '#f2f2f2'],
            data: [$windows11Percentage,
$otherWindowsPercentage]
            }]
            },
            options: {
            legend: {
            display: true,
            position: 'bottom',
            labels: {
            fontColor: '#333',
            fontSize: 14,
            padding: 16
            }
            }
            }
            });
            </script>
"@

$html | Out-File -FilePath "Windows11AdoptionReport.html"
$base64 =
[Convert]::ToBase64String([IO.File]::ReadAllBytes(".\Windo
ws11AdoptionReport.html"))

#Send Mail
$URLsend =
"https://graph.microsoft.com/v1.0/users/$MailSender/sendMa
il"
$BodyJsonsend = @"
```

```
{
    "message": {
      "subject": "Intune error report",
      "body": {
        "contentType": "Text",
        "content": "Dear Admin, this Mail contains the
Windows 11 Adoption report"
      },
      "toRecipients": [
        {
          "emailAddress": {
            "address": "$MailTo"
          }
        }
      ],
      "attachments": [
        {
          "@odata.type":
"#microsoft.graph.fileAttachment",
          "name": "Windows11AdoptionReport.html",
          "contentType": "text/plain",
          "contentBytes": "$base64"
        }
      ]
    }
  }
"@

$response = Invoke-MgRestMethod -Method POST -Uri $URLsend
-Body $BodyJsonsend
```

Once this is done, you can run the automation, and you will receive a report like this via Email:



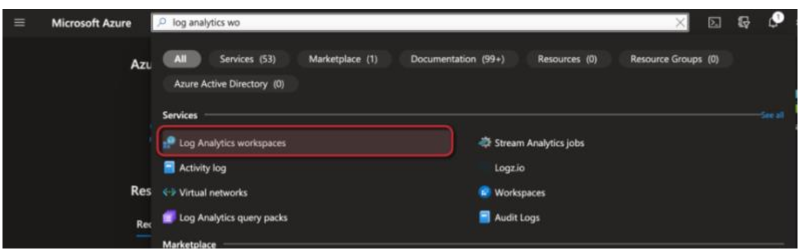
Solution 3: The Power of Remediation Scripts

In this example, I will show you how to write data in a log analytics workspace from a Toast notification on the device. This notification is generated by a remediation script.

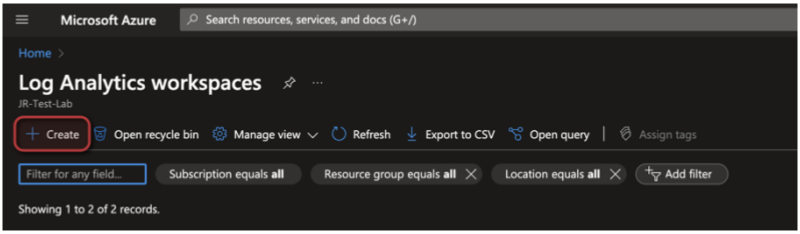
To do this, we have to create first a log analytics workspace:

Step 1: Deploy a log analytics workspace

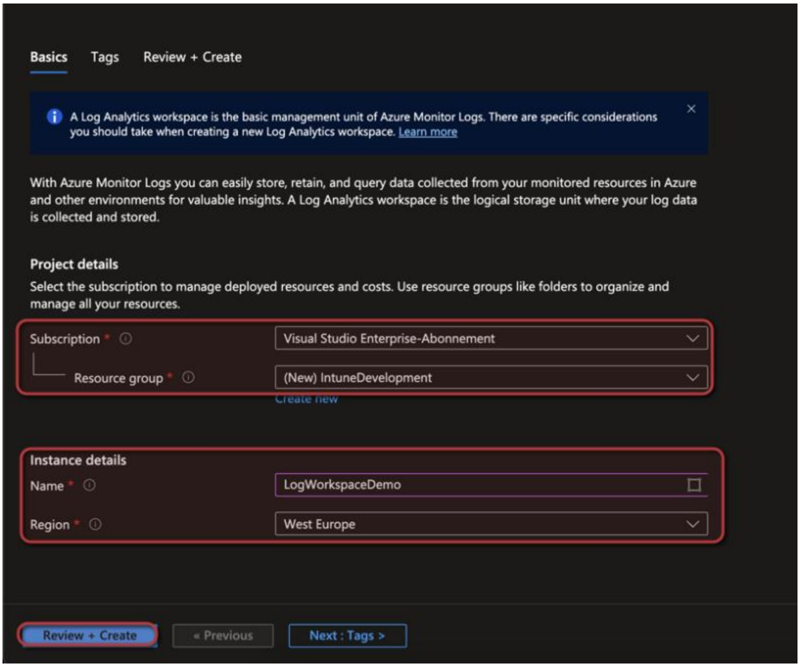
- Log in to the [Azure portal]
- Search for **Log Analytics** and select the **Log Analytics workspace** service



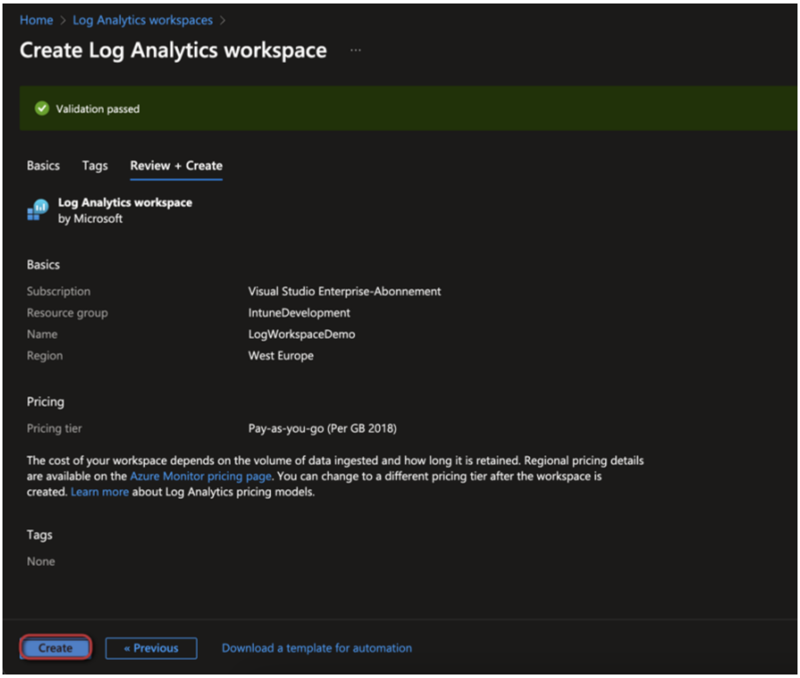
- Click **+ Create**



- Select the **Subscription** and a **Resource group** or create a new one
- Enter a **Name** for the Workspace and select a **Region**
- Click **Review + Create**

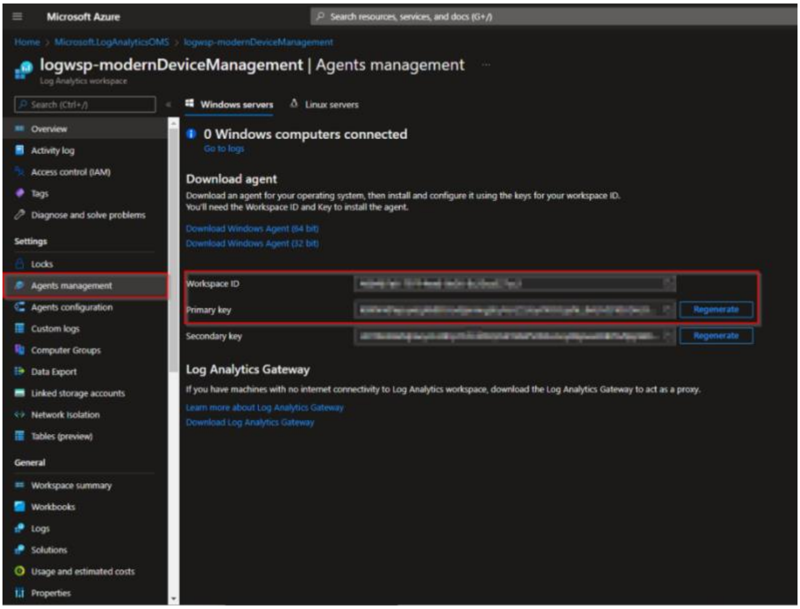


- Click **Create**



Step 2: Get Workspace information

- Open the new Log Analytics Workspace
- Navigate to **Agent management**
- Here, you can find the **Workspace ID** and the **Primary key**. You need this information later to insert this in the script



Step 3: Adapt the script

The script that opens the toast notification and sends the user selection to the log analytics workspace can be found here:

https://github.com/JayRHa/Book/blob/main/remediation/toastNotification.ps1

```
function Register-NotificationApp {
    param (
        [string]$AppID,
        [string]$AppDisplayName
    )

    [int]$ShowInSettings      = 0
    [int]$IconBackgroundColor = 0

    $IconUri           = "C:\Windows\ImmersiveControlPanel\images\logo.png"
    $AppRegPath        = "HKCU:\Software\Classes\AppUserModelId"
    $RegPath           = "$AppRegPath\$AppID"
    $NotificationsReg =
'HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Notifications\Settings'

    if (-not (Test-Path -Path "$NotificationsReg\$AppID")) {
        New-Item -Path "$NotificationsReg\$AppID" -Force
        New-ItemProperty -Path "$NotificationsReg\$AppID" -Name 'ShowInActionCenter' -Value 1
-PropertyType 'DWORD' -Force
    }

    if ((Get-ItemProperty -Path "$NotificationsReg\$AppID" -Name 'ShowInActionCenter' -
ErrorAction SilentlyContinue).ShowInActionCenter -ne '1') {
        New-ItemProperty -Path "$NotificationsReg\$AppID" -Name 'ShowInActionCenter' -Value 1
-PropertyType 'DWORD' -Force
    }

    try {
        if (-not (Test-Path $RegPath)) {
            New-Item -Path $AppRegPath -Name $AppID -Force | Out-Null
        }

        $DisplayName = Get-ItemProperty -Path $RegPath -Name DisplayName -ErrorAction
SilentlyContinue |
            Select-Object -ExpandProperty DisplayName -ErrorAction SilentlyContinue

        if ($DisplayName -ne $AppDisplayName) {
            New-ItemProperty -Path $RegPath -Name DisplayName -Value $AppDisplayName -
PropertyType String -Force | Out-Null
        }

        $ShowInSettingsValue = Get-ItemProperty -Path $RegPath -Name ShowInSettings -
ErrorAction SilentlyContinue |
            Select-Object -ExpandProperty ShowInSettings -ErrorAction SilentlyContinue

        if ($ShowInSettingsValue -ne $ShowInSettings) {
            New-ItemProperty -Path $RegPath -Name ShowInSettings -Value $ShowInSettings -
PropertyType DWORD -Force | Out-Null
        }

        New-ItemProperty -Path $RegPath -Name iconUri -Value $IconUri -PropertyType
ExpandString -Force | Out-Null
        New-ItemProperty -Path $RegPath -Name IconBackgroundColor -Value $IconBackgroundColor
-PropertyType ExpandString -Force | Out-Null
    }
    catch {
        # Handle exceptions if needed
    }
}

function Create-Action {
    param (
        [string]$ActionName
    )

    $MainRegPath  = "HKCU:\SOFTWARE\Classes\$ActionName"
    $CommandPath  = "$MainRegPath\shell\open\command"
    $CmdScript    = "C:\Users\Public\Documents\$ActionName.cmd"

    New-Item -Path $CommandPath -Force
    New-ItemProperty -Path $MainRegPath -Name "URL Protocol" -Value "" -PropertyType String -
Force | Out-Null
    Set-ItemProperty -Path $MainRegPath -Name "(Default)" -Value "URL:$ActionName Protocol" -
Force | Out-Null
    Set-ItemProperty -Path $CommandPath -Name "(Default)" -Value $CmdScript -Force | Out-Null
}
```

```
# Encode the image in base64: https://www.base64-image.de/
$ToastImageBase64 = ""

############################################# Variables
#######################################################
# Toast Text
$ToastTitle          = "Companyname IT Support"
$ToastHeadline       = "A reboot of your system is required!!"
$ToastText           = "We have installed updates on your system and a reboot is required. You sho
reboot your system as soon as possible. If now is the right time, perform the reboot now."
$ToastMessage        = "`nRun reboot now?"
$ToastLogoPath       = "C:\Windows\ImmersiveControlPanel\images\logo.png"
$ToastImagePath      = "$env:TEMP\ToastImage.png"
$ScriptExecutionPath = "C:\Users\Public\Documents"

################################################# Action
########################################################
$ActionScriptCmdReboot = @'
shutdown -r
'@

$ActionScriptCmdReboot | Out-File -FilePath "$ScriptExecutionPath\ActionReboot.cmd" -Force -Encodi
ASCII
Create-Action -ActionName "ActionReboot"

############################################## Notification
#####################################################
# Create PNG file from Base64 string
[byte[]]$Bytes = [Convert]::FromBase64String($ToastImageBase64)
[System.IO.File]::WriteAllBytes($ToastImagePath, $Bytes)

# Create toast notification XML
[xml]$Toast = @"
<toast scenario="reminder">
    <visual>
        <binding template="ToastGeneric">
            <image placement="hero" src="$ToastImagePath" />
            <image placement="appLogoOverride" hint-crop="circle" src="$ToastLogoPath" />
            <text>$ToastHeadline</text>
            <text>$ToastText</text>
            <group>
                <subgroup>
                    <text hint-style="body" hint-wrap="true">$ToastMessage</text>
                </subgroup>
            </group>
        </binding>
    </visual>
    <actions>
        <action activationType="protocol" arguments="ActionReboot:" content="Reboot Now" />
    </actions>
</toast>
"@

Register-NotificationApp -AppID $ToastTitle -AppDisplayName $ToastTitle

# Create toast notification
[Windows.UI.Notifications.ToastNotificationManager, Windows.UI.Notifications, ContentType =
WindowsRuntime] | Out-Null
[Windows.Data.Xml.Dom.XmlDocument, Windows.Data.Xml.Dom.XmlDocument, ContentType = WindowsRuntime]
Out-Null
$ToastXml = New-Object -TypeName Windows.Data.Xml.Dom.XmlDocument
$ToastXml.LoadXml($Toast.OuterXml)

# Show the toast
[Windows.UI.Notifications.ToastNotificationManager]::CreateToastNotifier($ToastTitle).Show($ToastX
```

```
function Register-NotificationApp {
    param (
        [string]$AppID,
        [string]$AppDisplayName
    )

    [int]$ShowInSettings      = 0
    [int]$IconBackgroundColor = 0

    $IconUri         =
"C:\Windows\ImmersiveControlPanel\images\logo.png"
    $AppRegPath      =
"HKCU:\Software\Classes\AppUserModelId"
    $RegPath         = "$AppRegPath\$AppID"
    $NotificationsReg =
'HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Notificat
ions\Settings'

    if (-not (Test-Path -Path "$NotificationsReg\$AppID"))
{
        New-Item -Path "$NotificationsReg\$AppID" -Force
        New-ItemProperty -Path "$NotificationsReg\$AppID"
-Name 'ShowInActionCenter' -Value 1 -PropertyType 'DWORD'
-Force
    }

    if ((Get-ItemProperty -Path "$NotificationsReg\$AppID"
-Name 'ShowInActionCenter' -ErrorAction
SilentlyContinue).ShowInActionCenter -ne '1') {
        New-ItemProperty -Path "$NotificationsReg\$AppID"
-Name 'ShowInActionCenter' -Value 1 -PropertyType 'DWORD'
-Force
    }

    try {
        if (-not (Test-Path $RegPath)) {
            New-Item -Path $AppRegPath -Name $AppID -Force
| Out-Null
        }

        $DisplayName = Get-ItemProperty -Path $RegPath -
Name DisplayName -ErrorAction SilentlyContinue |
            Select-Object -ExpandProperty DisplayName -
ErrorAction SilentlyContinue

        if ($DisplayName -ne $AppDisplayName) {
```

```
            New-ItemProperty -Path $RegPath -Name
DisplayName -Value $AppDisplayName -PropertyType String -
Force | Out-Null
        }

        $ShowInSettingsValue = Get-ItemProperty -Path
$RegPath -Name ShowInSettings -ErrorAction
SilentlyContinue |
            Select-Object -ExpandProperty ShowInSettings -
ErrorAction SilentlyContinue

        if ($ShowInSettingsValue -ne $ShowInSettings) {
            New-ItemProperty -Path $RegPath -Name
ShowInSettings -Value $ShowInSettings -PropertyType DWORD
-Force | Out-Null
        }

        New-ItemProperty -Path $RegPath -Name iconUri -
Value $IconUri -PropertyType ExpandString -Force | Out-
Null
        New-ItemProperty -Path $RegPath -Name
IconBackgroundColor -Value $IconBackgroundColor -
PropertyType ExpandString -Force | Out-Null
    }
    catch {
        # Handle exceptions if needed
    }
}

function Create-Action {
    param (
        [string]$ActionName
    )

    $MainRegPath  = "HKCU:\SOFTWARE\Classes\$ActionName"
    $CommandPath  = "$MainRegPath\shell\open\command"
    $CmdScript    =
"C:\Users\Public\Documents\$ActionName.cmd"

    New-Item -Path $CommandPath -Force
    New-ItemProperty -Path $MainRegPath -Name "URL
Protocol" -Value "" -PropertyType String -Force | Out-Null
    Set-ItemProperty -Path $MainRegPath -Name "(Default)"
-Value "URL:$ActionName Protocol" -Force | Out-Null
    Set-ItemProperty -Path $CommandPath -Name "(Default)"
-Value $CmdScript -Force | Out-Null
}
```

```
# Encode the image in base64: https://www.base64-image.de/
$ToastImageBase64 = ""

############################################# Variables
########################################################
# Toast Text
$ToastTitle          = "Companyname IT Support"
$ToastHeadline       = "A reboot of your system is
required!!"
$ToastText           = "We have installed updates on your
system and a reboot is required. You should reboot your
system as soon as possible. If now is the right time,
perform the reboot now."
$ToastMessage        = "`nRun reboot now?"
$ToastLogoPath       =
"C:\Windows\ImmersiveControlPanel\images\logo.png"
$ToastImagePath      = "$env:TEMP\ToastImage.png"
$ScriptExecutionPath = "C:\Users\Public\Documents"

################################################# Action
#########################################################
$ActionScriptCmdReboot = @'
shutdown -r
'@

$ActionScriptCmdReboot | Out-File -FilePath
"$ScriptExecutionPath\ActionReboot.cmd" -Force -Encoding
ASCII
Create-Action -ActionName "ActionReboot"

##############################################
Notification
#####################################################
# Create PNG file from Base64 string
[byte[]]$Bytes =
[Convert]::FromBase64String($ToastImageBase64)
[System.IO.File]::WriteAllBytes($ToastImagePath, $Bytes)

# Create toast notification XML
[xml]$Toast = @"
<toast scenario="reminder">
    <visual>
        <binding template="ToastGeneric">
            <image placement="hero" src="$ToastImagePath"
/>
```

```
            <image placement="appLogoOverride" hint-
crop="circle" src="$ToastLogoPath" />
            <text>$ToastHeadline</text>
            <text>$ToastText</text>
            <group>
                <subgroup>
                    <text hint-style="body" hint-
wrap="true">$ToastMessage</text>
                </subgroup>
            </group>
        </binding>
    </visual>
    <actions>
        <action activationType="protocol"
arguments="ActionReboot:" content="Reboot Now" />
    </actions>
</toast>
"@

Register-NotificationApp -AppID $ToastTitle -
AppDisplayName $ToastTitle

# Create toast notification
[Windows.UI.Notifications.ToastNotificationManager,
Windows.UI.Notifications, ContentType = WindowsRuntime] |
Out-Null
[Windows.Data.Xml.Dom.XmlDocument,
Windows.Data.Xml.Dom.XmlDocument, ContentType =
WindowsRuntime] | Out-Null
$ToastXml = New-Object -TypeName
Windows.Data.Xml.Dom.XmlDocument
$ToastXml.LoadXml($Toast.OuterXml)

# Show the toast
[Windows.UI.Notifications.ToastNotificationManager]::Creat
eToastNotifier($ToastTitle).Show($ToastXml)
```

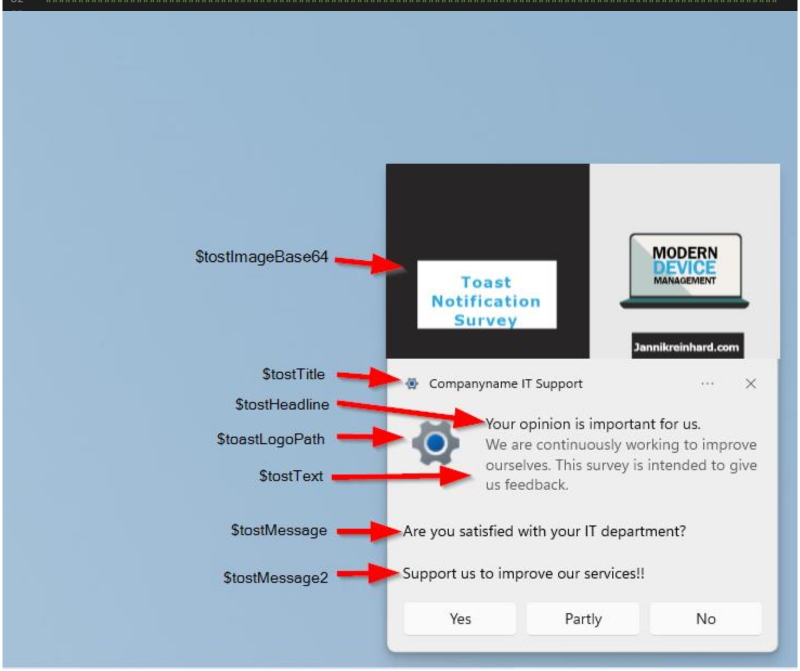
Open it in a code editor like Visual Studio Code.

- First you need an Image for the toast header. To add this image into the PowerShell script you must encode this to base64. I use the following tool for the encoding: https://www.base64-image.de/
When the picture is encoded, you can past the

base64 string into the *$tostImageBase64* Variable in the script

- Next, we have to adapt the following variables with the content you want to show:

```
############################################# Variables ##############################################################
########### Adjust the Variables here ################################################################################
######################################################################################################################
# Toast Text
$tostTitle       = "Companyname IT Support"
$tostHeadline    = "Your opinion is important for us."
$tostText        = "We are continuously working to improve ourselves. This survey is intended to give us feedback."
$tostMessage     = "`nAre you satisfied with your IT department?"
$tostMessage2    = "`Support us to improve our services!!"
$toastLogoPath   = "C:\Windows\ImmersiveControlPanel\images\logo.png"
#######################################################################################################################
```

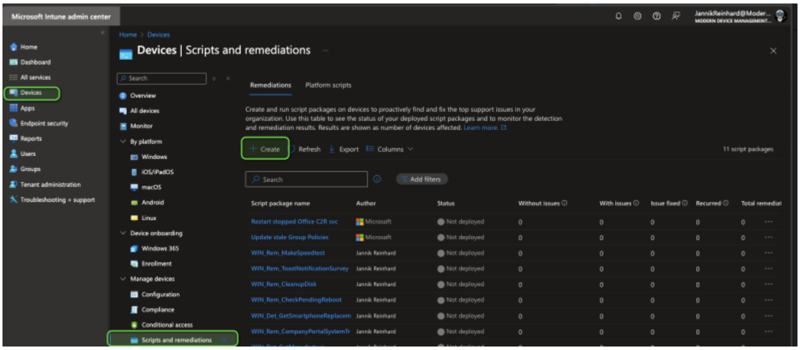


Afterwards we have to insert the *$customerId* and the *$sharedKey*. Both information we have copied in the ***Get Workspace informations*** section.
The *$customerId* = **Workspace ID** and the *$sharedKey* = **Primary key**. Optional you can change the name of the log analytics table with the $logType Variable.

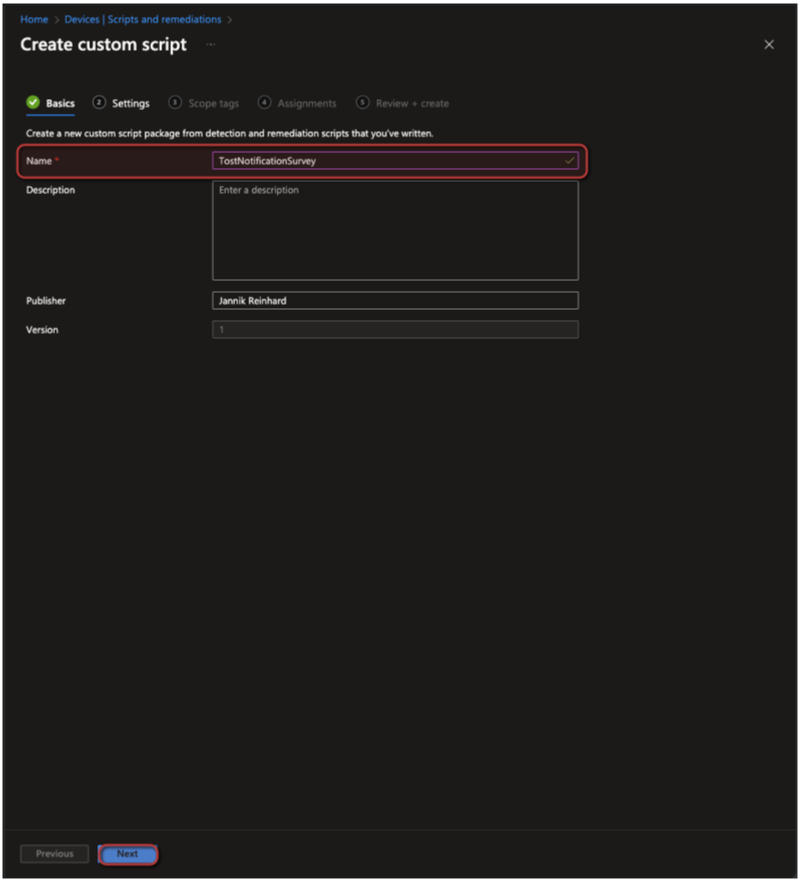
```
#####################################################################################################################
### PS Script for Action #### Add customerId and sharedKey from the log analytics workspace here ###############
#####################################################################################################################
$actionScriptPre = @'
    $customerId = "xxxxx"
    $sharedKey = 'xxxxx'
    $logType = "Survey"

    # Log analytics functions
    Function Build-Signature ($customerId, $sharedKey, $date, $contentLength, $method, $contentType, $resource)
    {
        $xHeaders = "x-ms-date:" + $date
        $stringToHash = $method + "`n" + $contentLength + "`n" + $contentType + "`n" + $xHeaders + "`n" + $resource

        $bytesToHash = [Text.Encoding]::UTF8.GetBytes($stringToHash)
        $keyBytes = [Convert]::FromBase64String($sharedKey)

        $sha256 = New-Object System.Security.Cryptography.HMACSHA256
        $sha256.Key = $keyBytes
        $calculatedHash = $sha256.ComputeHash($bytesToHash)
        $encodedHash = [Convert]::ToBase64String($calculatedHash)
        $authorization = 'SharedKey {0}:{1}' -f $customerId,$encodedHash
        return $authorization
```

Step 4: Deploy Remediation Script

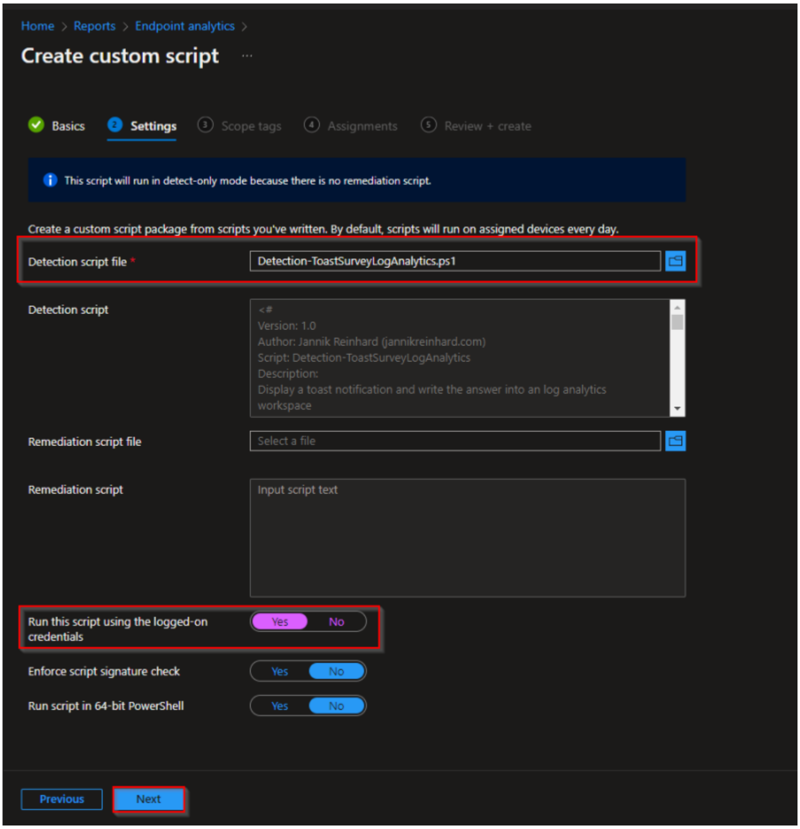
- To deploy this toast notification to the clients, we open the Microsoft Intune admin center and navigate to **Devices** -> **Scripts and Remediation** -> **Remediations**
- Click + **Create**



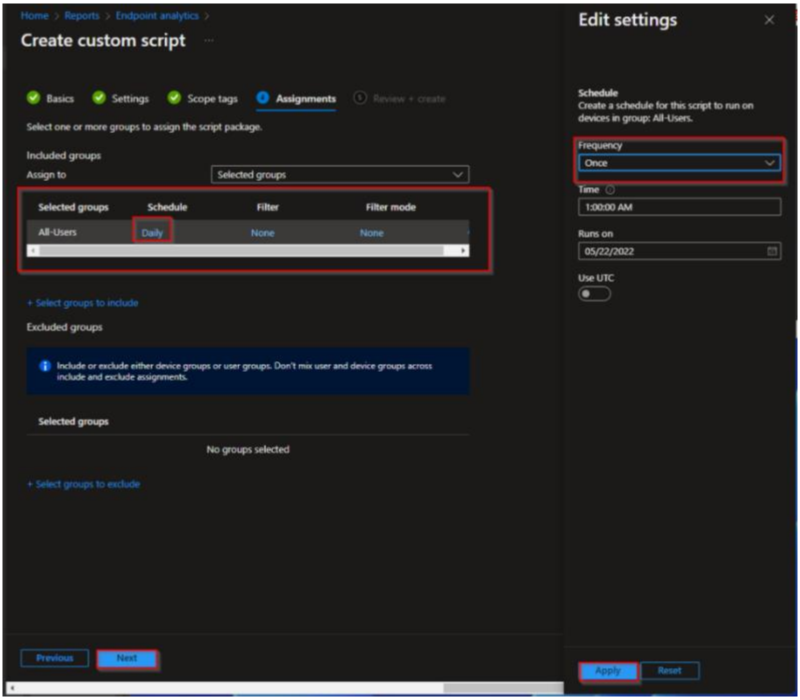
- Enter a **name**
- Click **Next**



- Upload the script as **detection**
- Select **Yes** for ***Run this script using the logged on credentials***
- Click **Next**



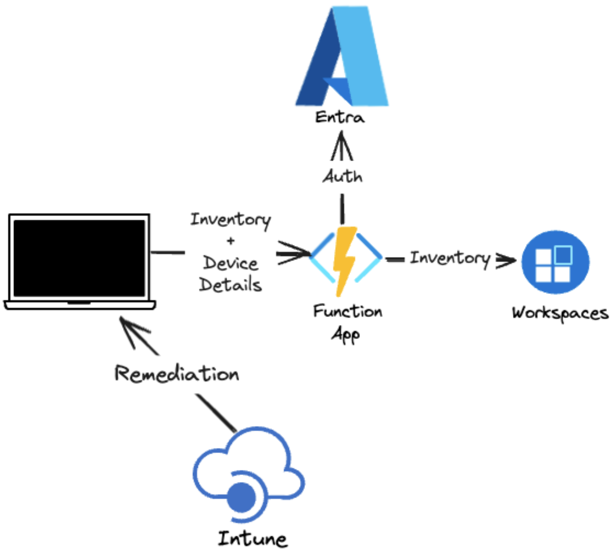
- Click **Next**
- **Assign** the Script to a group and click on **Daily**
- Change the Schedule to **Once** (You can also select another schedule like hourly or daily) and click **apply**
- Click **Next**



- Click **Create**

Solution 4: The Power of Function

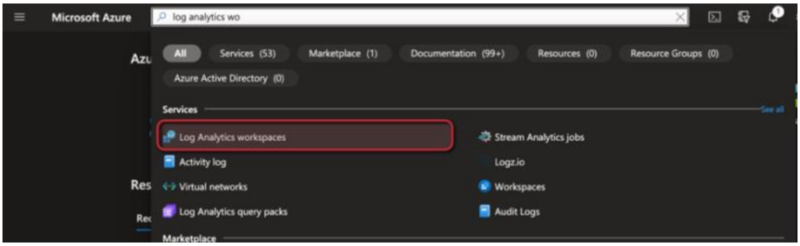
In this solution, I will show you a community tool from Sandy Zeng and Jan Keitil Skanke. This tool is named Intune Enhanced Inventory, which is a combination of remediation scripts, log analytics, and Azure functions to collect inventory from clients in a secure way.



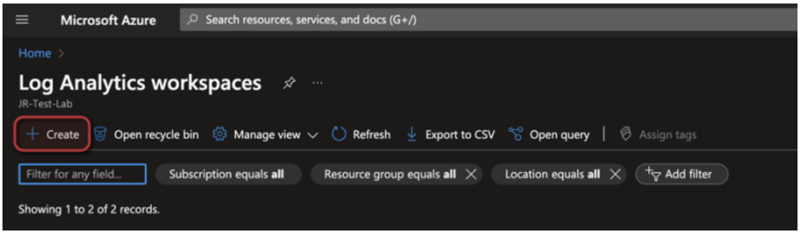
Deploying the custom Inventory Solution in Intune

The first thing you have to do is to setup a log analytics workspace.

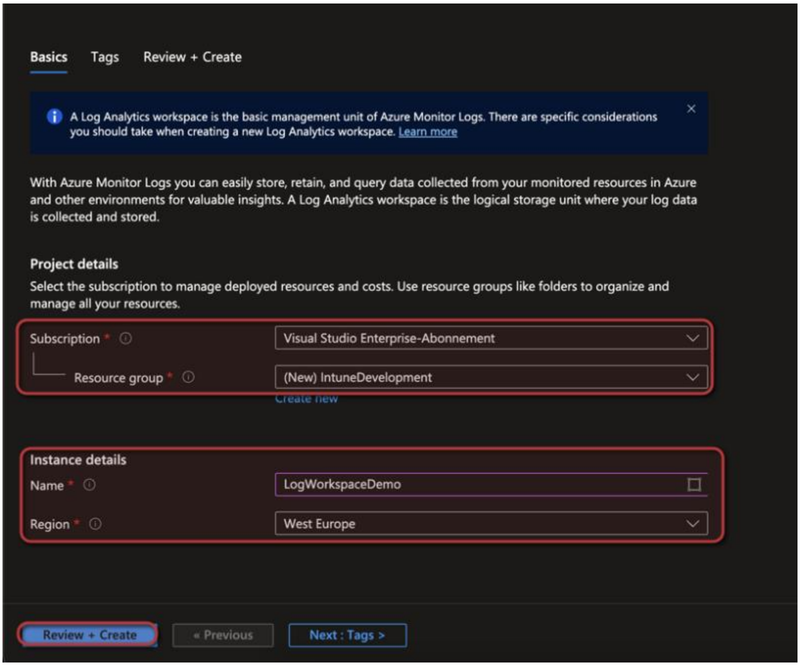
- Log in to the Azure portal
- Search for **Log Analytics** and select the **Log Analytics workspace** service



- Click **+ Create**

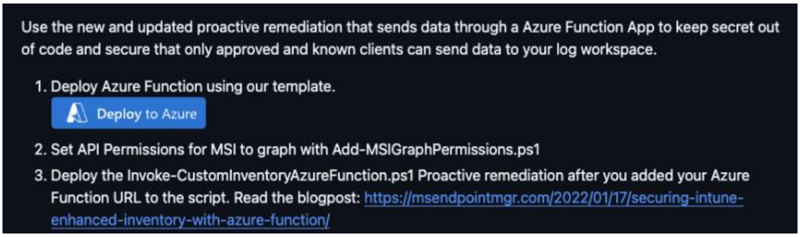


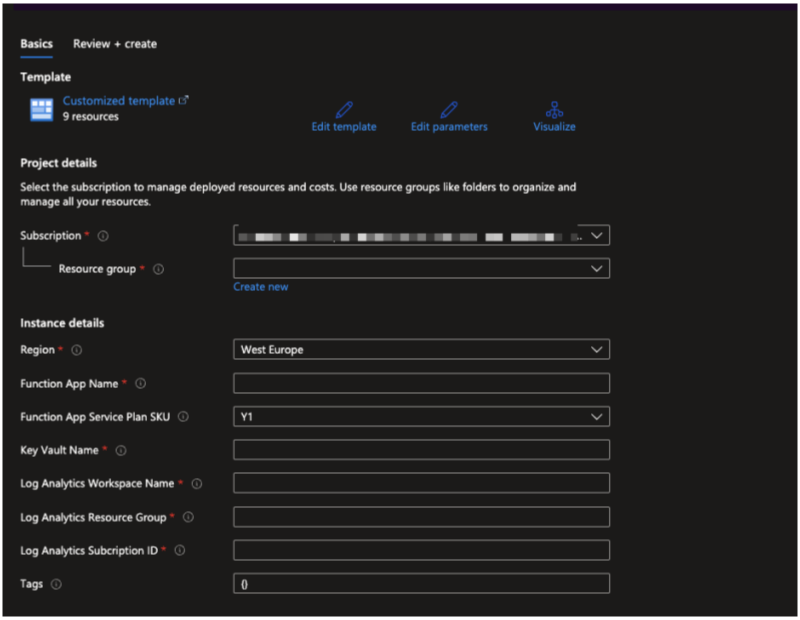
- Select the **Subscription** and a **Resource group** or create a new one
- Enter a **Name** for the Workspace and select a **Region**
- Click **Review + Create**



- Click **Create**

The rest of the deployment is simple. In the GitHub project https://github.com/MSEndpointMgr/IntuneEnhancedInventory/tree/main, click the "Deploy to Azure" button. Fill in details like the function's name.





Once completed, the deployment will automatically run, creating an Azure Function, Log Analytics Workspace, Key Vault, and other helper functions.

Afterward, add the Permissions to the MSI. A script in the repository can assist you with this step.

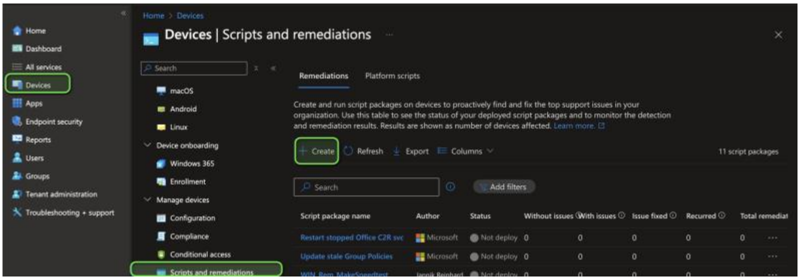
https://github.com/MSEndpointMgr/IntuneEnhancedInventory/blob/main/Add-MSIGraphPermissions.ps1

```
#Requires -Modules Microsoft.Graph
# Install the module. (You need admin on the machine.)
# Install-Module Microsoft.Graph

# Set Static Variables
$TenantID=""
$ServicePrincipalAppDisplayName =""

# Define dynamic variables
$ServicePrincipalFilter = "displayName eq '$($ServicePrincipalAppDisplayName)'"
$GraphAPIAppName = "Microsoft Graph"
$ApiServicePrincipalFilter = "displayName eq '$($GraphAPIAppName)'"

# Scopes needed for the managed identity (Add other scopes if needed)
$Scopes = @(
    "Device.Read.All"
)

# Connect to MG Graph - scopes must be consented the first time you run this.
# Connect with Global Administrator
Select-MgProfile -Name "beta"
Connect-MgGraph -Scopes "Application.Read.All","AppRoleAssignment.ReadWrite.All"  -TenantId
$TenantID -UseDeviceAuthentication

# Get the service principal for your managed identity.
$ServicePrincipal = Get-MgServicePrincipal -Filter $ServicePrincipalFilter

# Get the service principal for Microsoft Graph.
# Result should be AppId 00000003-0000-0000-c000-000000000000
$ApiServicePrincipal = Get-MgServicePrincipal -Filter "$ApiServicePrincipalFilter"

# Apply permissions
Foreach ($Scope in $Scopes) {
    Write-Host "`nGetting App Role '$Scope'"
    $AppRole = $ApiServicePrincipal.AppRoles | Where-Object {$_.Value -eq $Scope -and
$_.AllowedMemberTypes -contains "Application"}
    if ($null -eq $AppRole) { Write-Error "Could not find the specified App Role on the Api
Service Principal"; continue; }
    if ($AppRole -is [array]) { Write-Error "Multiple App Roles found that match the request";
continue; }
    Write-Host "Found App Role, Id '$($AppRole.Id)'"

    $ExistingRoleAssignment = Get-MgServicePrincipalAppRoleAssignment -ServicePrincipalId
$ServicePrincipal.Id | Where-Object { $_.AppRoleId -eq $AppRole.Id }
    if ($null -eq $existingRoleAssignment) {
        New-MgServicePrincipalAppRoleAssignment -ServicePrincipalId $ServicePrincipal.Id -
PrincipalId $ServicePrincipal.Id -ResourceId $ApiServicePrincipal.Id -AppRoleId $AppRole.Id
    } else {
        Write-Host "App Role has already been assigned, skipping"
    }
}
```

```
#Requires -Modules Microsoft.Graph
# Install the module. (You need admin on the machine.)
# Install-Module Microsoft.Graph

# Set Static Variables
$TenantID=""
$ServicePrincipalAppDisplayName =""

# Define dynamic variables
$ServicePrincipalFilter = "displayName eq
'$($ServicePrincipalAppDisplayName)'"
$GraphAPIAppName = "Microsoft Graph"
$ApiServicePrincipalFilter = "displayName eq
'$($GraphAPIAppName)'"

# Scopes needed for the managed identity (Add other scopes
if needed)
$Scopes = @(
    "Device.Read.All"
)

# Connect to MG Graph - scopes must be consented the first
time you run this.
# Connect with Global Administrator
Select-MgProfile -Name "beta"
Connect-MgGraph -Scopes
"Application.Read.All","AppRoleAssignment.ReadWrite.All"
-TenantId $TenantID -UseDeviceAuthentication

# Get the service principal for your managed identity.
$ServicePrincipal = Get-MgServicePrincipal -Filter
$ServicePrincipalFilter

# Get the service principal for Microsoft Graph.
# Result should be AppId 00000003-0000-0000-c000-
000000000000
$ApiServicePrincipal = Get-MgServicePrincipal -Filter
"$ApiServicePrincipalFilter"

# Apply permissions
Foreach ($Scope in $Scopes) {
    Write-Host "`nGetting App Role '$Scope'"
    $AppRole = $ApiServicePrincipal.AppRoles | Where-
Object {$_.Value -eq $Scope -and $_.AllowedMemberTypes -
contains "Application"}
```

```
    if ($null -eq $AppRole) { Write-Error "Could not find
the specified App Role on the Api Service Principal";
continue; }
    if ($AppRole -is [array]) { Write-Error "Multiple App
Roles found that match the request"; continue; }
    Write-Host "Found App Role, Id '$($AppRole.Id)'"

    $ExistingRoleAssignment = Get-
MgServicePrincipalAppRoleAssignment -ServicePrincipalId
$ServicePrincipal.Id | Where-Object { $_.AppRoleId -eq
$AppRole.Id }
    if ($null -eq $existingRoleAssignment) {
        New-MgServicePrincipalAppRoleAssignment -
ServicePrincipalId $ServicePrincipal.Id -PrincipalId
$ServicePrincipal.Id -ResourceId $ApiServicePrincipal.Id -
AppRoleId $AppRole.Id
    } else {
        Write-Host "App Role has already been assigned,
skipping"
    }
}
```

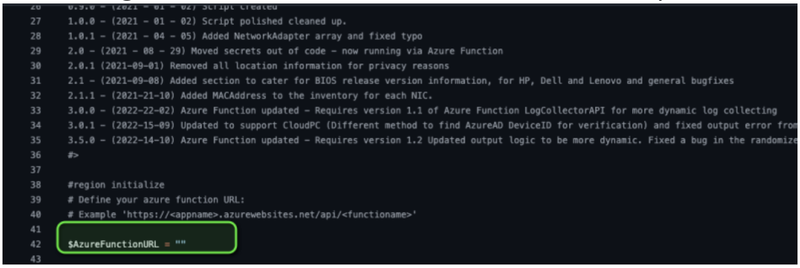
Once the Azure deployment is complete, add the remediation script to Intune. Deploy it as a 64-bit detection script running outside the user context, unless you want to collecting user data.



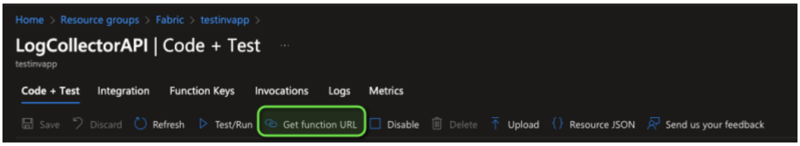
The script you must deploy is the following:

https://github.com/MSEndpointMgr/IntuneEnhancedInventory/blob/main/Proactive%20Remediation/Invoke-CustomInventoryAzureFunction.ps1

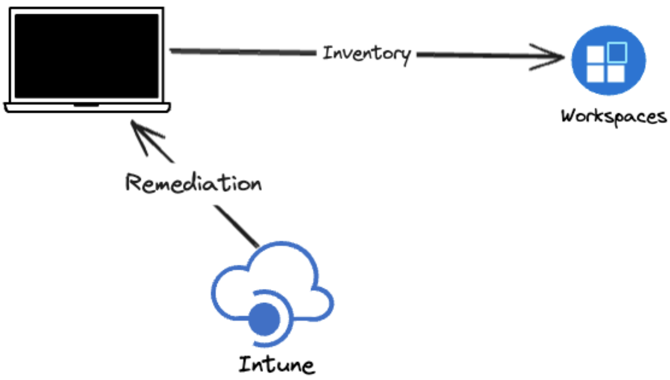
Don’t forget to fill in the function URL into this script:



You can find the URL in the Azure Function header behind the “Get function URL” button. This URL is pre-authenticated with a key.



This is the most advanced and secure method. For a simpler option, you can write directly from the remediation script into the log analytics workspace. The downside is that you must include the workspace key in the script, which leaves a footprint on the device. Someone who finds this key could write random data into the workspace, but since it’s write-only, I don't see this as a major risk.



To do this, you only need the log analytics workspace, and you can use this script to write into the log analytics workspace.

https://github.com/JayRHa/Book/blob/main/remediation/sendInventoryToLogAnalytics.ps1

```powershell
$workspaceId = ""
$workspaceKey = ""
$logType = "RemediationData"

function Send-LogAnalyticsData {
    param (
        [string]$CustomerId,
        [string]$SharedKey,
        [string]$Body,
        [string]$LogType
    )

    $method = "POST"
    $contentType = "application/json"
    $resource = "/api/logs"
    $rfc1123date = [DateTime]::UtcNow.ToString("r")
    $contentLength = $Body.Length

    $signature = Get-SignatureBuilded `
        -customerId $CustomerId `
        -sharedKey $SharedKey `
        -date $rfc1123date `
        -contentLength $contentLength `
        -method $method `
        -contentType $contentType `
        -resource $resource

    $uri = "https://$CustomerId.ods.opinsights.azure.com$resource?api-version=2016-04-01"

    $headers = @{
        "Authorization"        = $signature
        "Log-Type"             = $LogType
        "x-ms-date"            = $rfc1123date
        "time-generated-field" = ""
    }

    $response = Invoke-WebRequest -Uri $uri -Method $method -ContentType $contentType -Headers
$headers -Body $Body -UseBasicParsing
    return $response.StatusCode
}

$inventory = @{
    "attribute1" = "value1"
    "attribute2" = "value2"
}

try {
    $data = $inventory | ConvertTo-Json -Depth 100
    $params = @{
        CustomerId = $workspaceId
        SharedKey  = $workspaceKey
        Body       = [System.Text.Encoding]::UTF8.GetBytes($data)
        LogType    = $logType
    }

    $logResponse = Send-LogAnalyticsData @params
}
catch {
    Write-Error $_.Exception.Message
    exit 1
}

if ($logResponse -eq 200) {
    Write-Output "Data sent successfully"
}
else {
    Write-Error "Failed to send data. Response code: $logResponse"
    exit 1
}
```

```
$workspaceId = ""
$workspaceKey = ""
$logType = "RemediationData"

function Send-LogAnalyticsData {
    param (
        [string]$CustomerId,
        [string]$SharedKey,
        [string]$Body,
        [string]$LogType
    )

    $method = "POST"
    $contentType = "application/json"
    $resource = "/api/logs"
    $rfc1123date = [DateTime]::UtcNow.ToString("r")
    $contentLength = $Body.Length

    $signature = Get-SignatureBuilded `
        -customerId $CustomerId `
        -sharedKey $SharedKey `
        -date $rfc1123date `
        -contentLength $contentLength `
        -method $method `
        -contentType $contentType `
        -resource $resource

    $uri =
"https://$CustomerId.ods.opinsights.azure.com$resource?api
-version=2016-04-01"

    $headers = @{
        "Authorization"        = $signature
        "Log-Type"             = $LogType
        "x-ms-date"            = $rfc1123date
        "time-generated-field" = ""
    }

    $response = Invoke-WebRequest -Uri $uri -Method
$method -ContentType $contentType -Headers $headers -Body
$Body -UseBasicParsing
    return $response.StatusCode
}

$inventory = @{
    "attribute1" = "value1"
    "attribute2" = "value2"
```

```
}

try {
    $data = $inventory | ConvertTo-Json -Depth 100
    $params = @{
        CustomerId = $workspaceId
        SharedKey  = $workspaceKey
        Body       =
[System.Text.Encoding]::UTF8.GetBytes($data)
        LogType    = $logType
    }

    $logResponse = Send-LogAnalyticsData @params
}
catch {
    Write-Error $_.Exception.Message
    exit 1
}

if ($logResponse -eq 200) {
    Write-Output "Data sent successfully"
}
else {
    Write-Error "Failed to send data. Response code:
$logResponse"
    exit 1
}
```

Chapter 6: Simplify your life and use amazing third-party tools

Advertisement: This chapter contains promotional content from my partners.

RECAST SOFTWARE

Introduction

Cybersecurity threats continue to escalate in both volume and complexity. According to the [2024 IBM Data Breach Report](https://www.ibm.com/reports/data-breach) (https://www.ibm.com/reports/data-breach), the global average cost of a data breach has reached $4.88 million, reflecting a 10% increase over the previous year. Attackers increasingly deploy ransomware, zero-day exploits, and advanced persistent threats across diverse endpoints, including desktops, mobile devices, IoT sensors, and cloud workloads.

Meanwhile, teams struggle with staffing constraints. As attackers grow bolder and more inventive, organizations must take on overlapping responsibilities that extend into compliance, data protection, and other domains once considered separate from basic endpoint management.

AI as a Transformative Element in Endpoint Management

Effective endpoint management establishes a reliable foundation for defense. Integrating AI-driven insights further transforms this approach, shifting from basic maintenance tasks toward proactive, data-informed strategies that stay ahead of evolving risks.

The Benefits of AI in Endpoint Management and Cybersecurity

Enhancing Threat Detection and Response

Traditional security approaches rely on predefined rules and signatures. AI employs adaptive models that continuously refine themselves using real-time telemetry. Organizations gain early warning signals as the system identifies anomalies and subtle patterns that suggest a breach in progress. When threats emerge, AI-guided tools can isolate devices immediately and contain attacks before they spread.

Automating Compliance and Security Protocols

Staying compliant often involves repetitive tasks: validating controls, rotating passwords, generating reports, and enforcing configuration baselines. AI-driven automation handles these responsibilities continuously. By reducing manual effort and the possibility of human error, teams maintain higher levels of compliance while freeing staff to tackle more strategic initiatives.

Improving Decision-Making with AI-Driven Insights

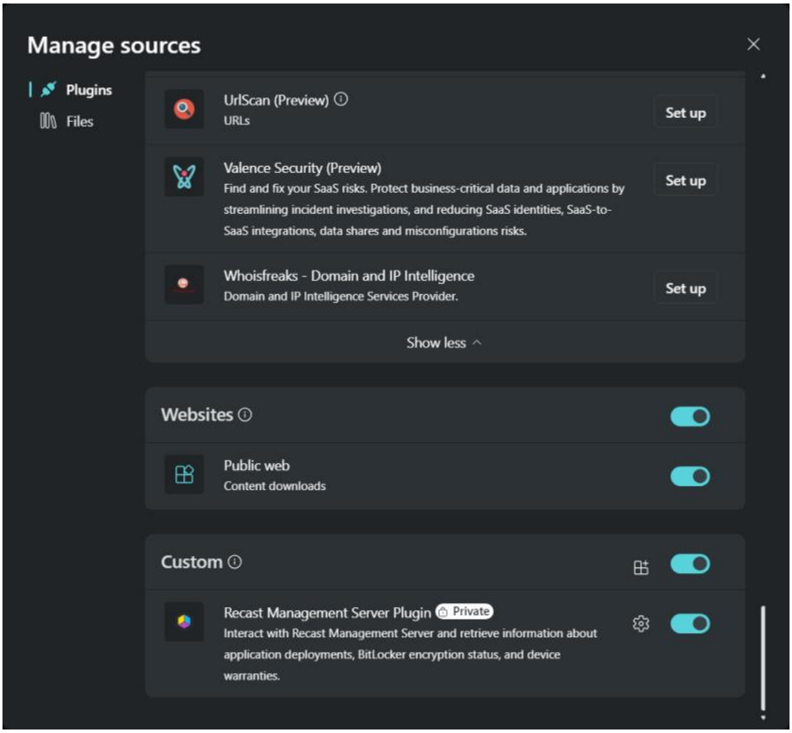
The volume and diversity of security data can overwhelm even the most experienced professionals. AI helps interpret this information more efficiently, turning raw data into actionable insights that guide security policies, device configurations, and authentication approaches. **The technology also lowers the barrier to accessing complex security data.** Team members simply describe what they need—such as warranty details or recent patch status—and AI determines the correct steps. Less-experienced staff can contribute more effectively, and experts gain time for deeper analysis.

Expanding Intune with Microsoft Copilot for Security and the Recast Plugin

Microsoft's Vision and Recast's Alignment

Microsoft's planned Copilot for Security experience introduces new ways to access and interpret security data within familiar management consoles. While the timing of public availability remains in Microsoft's hands, the direction is clear: a more intuitive and accessible security intelligence layer built into Intune.

Recast Software is working closely with Microsoft to support this initiative. By using the Recast Management Server, organizations can unify data from Intune, Configuration Manager, Active Directory, MBAM, and the Recast Agent. As Microsoft continues to refine Copilot for Security, Recast's plugin will evolve alongside it.

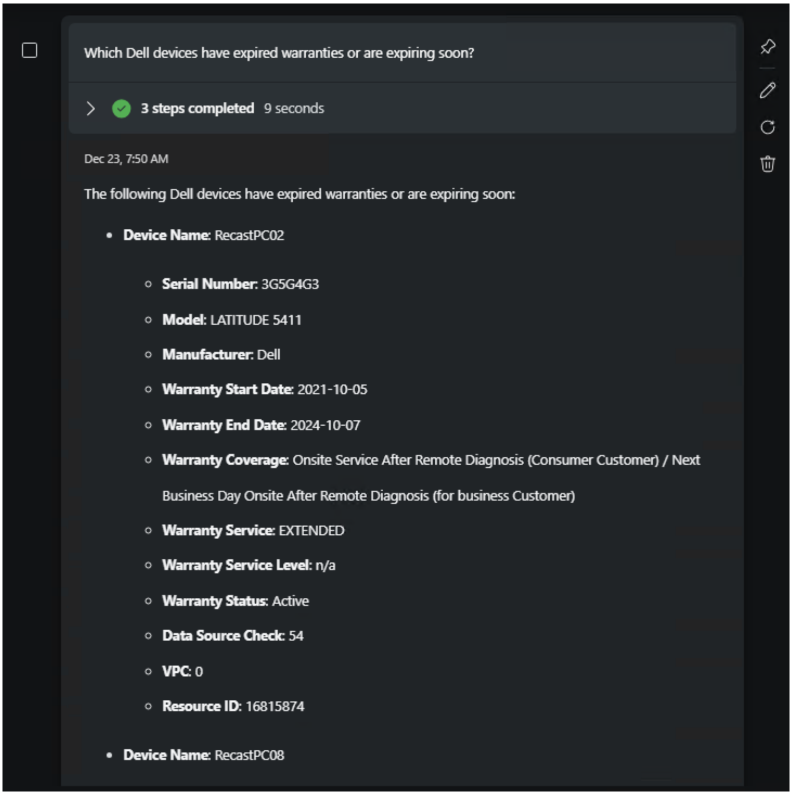


Initial Skill Groups and Data Sources

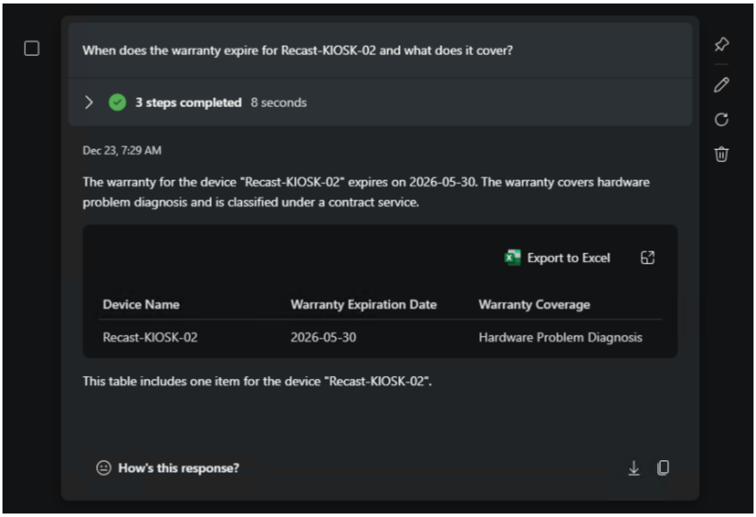
The Recast Copilot for Security Plugin initially targets three critical data streams to enhance Intune's functionality:

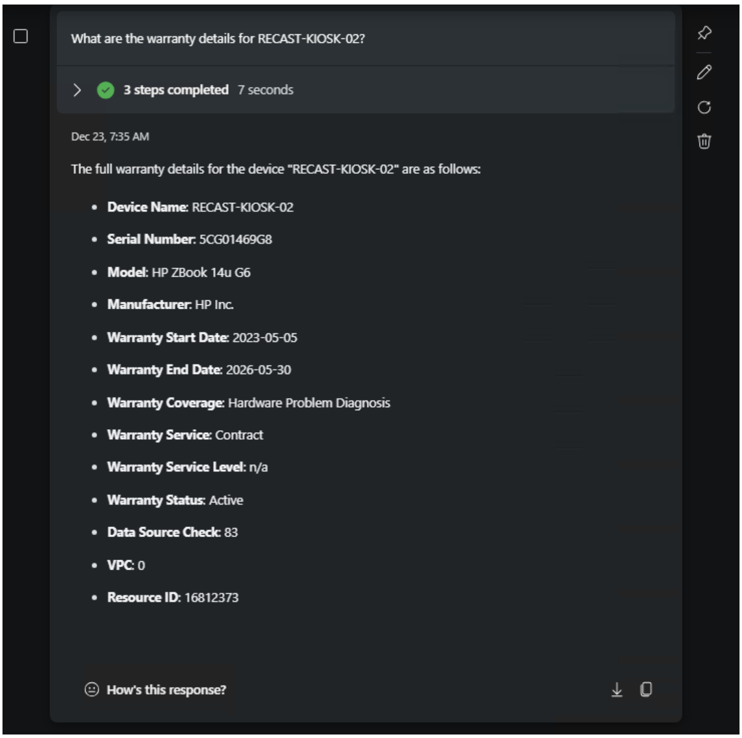
<u>Endpoint Insights</u>

Access warranty information, inventory details, and device age data directly through the Plugin's Copilot interface. Instead of digging through multiple reports, teams can ask, "Which laptops are nearing end-of-warranty?" and receive a straightforward list of devices.



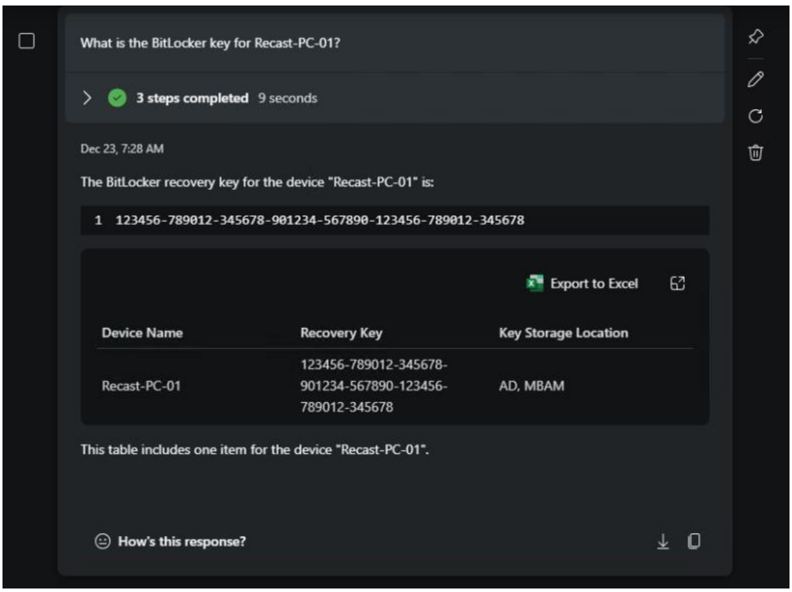
This intelligence helps with budgeting, risk reduction, and proactive hardware refresh cycles. Since Endpoint Insights automatically keeps tabs on computers, monitors, and docking stations—even as they change hands or leave the office—administrators can plan more effectively and avoid unexpected downtime.

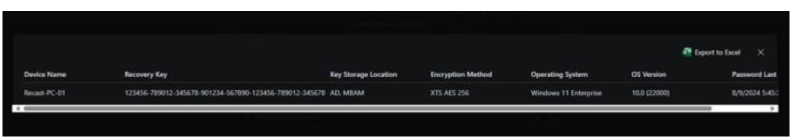

When does the warranty expire for Recast-KIOSK-02 and what does it cover?
3 steps completed 8 seconds
Dec 23, 7:29 AM
The warranty for the device "Recast-KIOSK-02" expires on 2026-05-30. The warranty covers hardware problem diagnosis and is classified under a contract service.
Export to Excel
Device Name
Warranty Expiration Date
Warranty Coverage
Recast-KIOSK-02
2026-05-30
Hardware Problem Diagnosis
This table includes one item for the device "Recast-KIOSK-02".
How's this response?


What are the warranty details for RECAST-KIOSK-02?
3 steps completed 7 seconds
Dec 23, 7:35 AM
The full warranty details for the device "RECAST-KIOSK-02" are as follows:
Device Name: RECAST-KIOSK-02
Serial Number: 5CG01469G8
Model: HP ZBook 14u G6
Manufacturer: HP Inc.
Warranty Start Date: 2023-05-05
Warranty End Date: 2026-05-30
Warranty Coverage: Hardware Problem Diagnosis
Warranty Service: Contract
Warranty Service Level: n/a
Warranty Status: Active
Data Source Check: 83
VPC: 0
Resource ID: 16812373
How's this response?

Right Click Tools

Quickly verify encryption status and ensure compliance with encryption policies through intuitive queries—no more guessing or manual checks. For example, “What is the BitLocker key for Recast-PC-01?” will return the BitLocker key data. With Right Click Tools integrated into the Plugin, organizations maintain a tighter security posture, avoiding scenarios where lost or stolen devices expose sensitive information. This direct line of sight into encryption compliance saves time, reduces uncertainty, and ultimately strengthens data protection.





Application Manager

Supplies continuous intelligence about third-party application releases and updates, making it simpler to identify and remediate vulnerabilities through the Copilot for Security interface. Traditionally, staying current with third-party patches and managing the full lifecycle of applications—from installation to retirement—has been time-consuming and prone to oversight. With Application Manager integrated into the plugin, teams can ask questions like, “What is the latest version of Adobe Acrobat in the Recast application catalog?” and immediately see results. This combination of broad application catalog coverage and proactive update delivery helps reduce security gaps and manual effort, all accessed through the same Copilot-driven workflow that powers the rest of your endpoint security operations.
Future enhancements will draw upon the extensive capabilities of the Recast platform, deepening the AI-driven support for endpoint management tasks.

A New IT Interface: Recast Copilot for Security Plugin

The Recast Copilot for Security Plugin integrates directly with Microsoft Copilot for Security. By combining Intune data with Recast’s resource-rich datasets, security teams gain a new, user-friendly interface for accessing critical information. Instead of switching between consoles, administrators can pose direct questions—“What is the warranty expiration for Device x-01?”—and receive immediate, actionable answers. Enabling or disabling the plugin is done through the Copilot interface, allowing teams to configure their environments as needed. Organizations can apply granular access to the plugin by activating and deactivating it at either the individual or organizational level.

Key Features and Benefits

- **AI-Driven Insights:** The plugin simplifies compliance checks and security setting adjustments, guiding teams toward configurations that lower risk.
- **Expanded Data Context:** Warranty details, BitLocker status, and application information enrich native Intune data, helping professionals manage devices more effectively.
- **Operational Advantages:** A consolidated data view reduces guesswork, speeds response times, streamlines compliance management, and supports better decisions. Natural language prompts also open the door for newer team members, minimizing reliance on specialized experts.

Future Vision: AI in 2025 and Beyond

The influence of AI on endpoint management and cybersecurity will continue to expand. Future advancements may feature simulation environments to evaluate defenses against hypothetical attacks and sophisticated algorithms to detect novel, subtle threats. Deep learning models, attuned to context and user intent, will better understand the patterns behind network traffic and system behavior.

Automated systems may increasingly self-correct, freeing security teams to prioritize strategic initiatives. AI-assisted Security Operations Centers will streamline event handling, while decision-support systems help guide where to invest in tools or how to adjust policies.

AI Challenges and Considerations

Amid this progress, organizations must remain vigilant. Ensuring transparency in AI decision-making, addressing biases, and respecting data privacy will remain top priorities. Maintaining compliance in a landscape shaped by rapidly evolving regulations will require ongoing attention and adaptation.

Recast Software's Vision for the Future

[Recast Software](https://www.recastsoftware.com/technology-intune) (https://www.recastsoftware.com/technology-intune) will continue to partner and advance AI-driven endpoint management and security, broadening data sources, refining the user experience, and working closely with Microsoft and others to give organizations greater control, context, and confidence. AI has reshaped the security landscape, moving from static policies to continuous learning and proactive defense. Those who adopt AI-driven tools, including the Recast Copilot for Security Plugin, can more easily anticipate threats and adjust their strategies. As attackers evolve their methods, defenders who embrace ongoing collaboration, feedback, and development will maintain a more resilient security posture, better equipped to meet emerging challenges.

At the halfway point, you've learned how to extract and automate Intune reports using Graph, enabling you to improve your daily workflows into more proactive and efficient routines. Before exploring how large language models (LLMs) can take your administration to the next level, let's take a moment to examine additional opportunities for automation and management. These are the tasks you likely wish you didn't have to handle manually.

One of the most time-consuming aspects of managing Intune is application management. With the ever-increasing number of CVEs and zero-day exploits, ensuring that applications are up-to-date and secure is critical. However, it's also a process that can quickly consume valuable time. This is where Robopack can help.

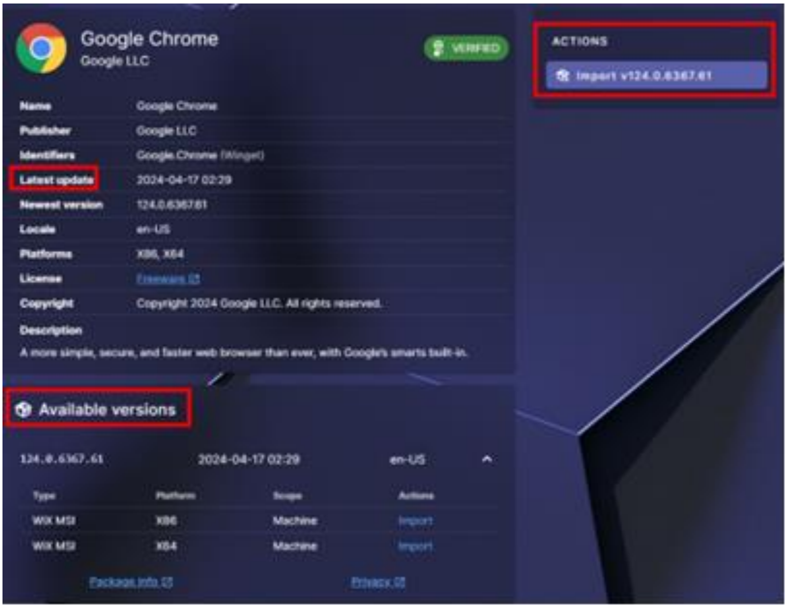
Robopack simplifies application management through its immense collection of pre-tested and configured applications, which includes over 40,000 applications at the time of writing. With just a few clicks, your standard off-the-shelf applications can be seamlessly deployed to your tenant(s), ready for immediate use.

But Robopack goes beyond basic deployment. With its revolutionary Robopack Waves feature, application updates are entirely self-managed. You can configure update groups, and when a new version of an application is released, it is automatically deployed to your initial group. The software continuously monitors installation success rates and only proceeds to the next group of devices once the predefined success threshold is achieved. Robopack's automated update checks further simplify your workflow. There's no need to monitor for new versions or plan updates manually as the software takes care of it for you.

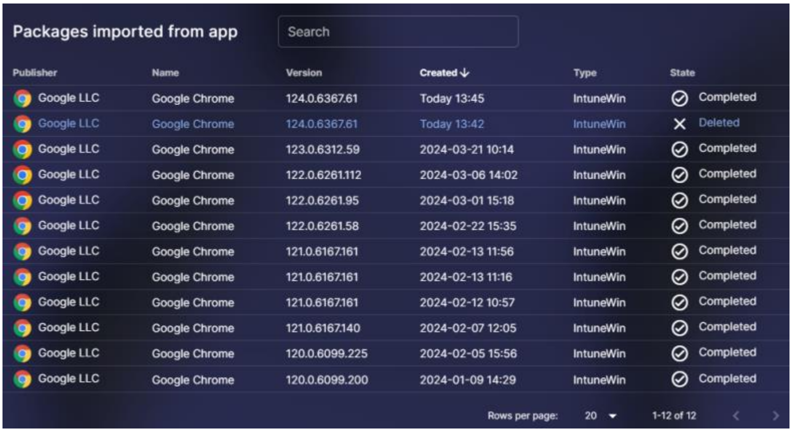
Applications run through a testing process before being packaged and deployed. As a result, you can review all associated files and registry keys, providing transparency and peace of mind, particularly in industries with stringent compliance requirements. For publicly available applications tailored to your environment, Robopack enables you to add custom installation strings and configurations, ensuring compatibility with your specific needs.

Here are some screenshots to demonstrate just how simple it is to deploy an application:

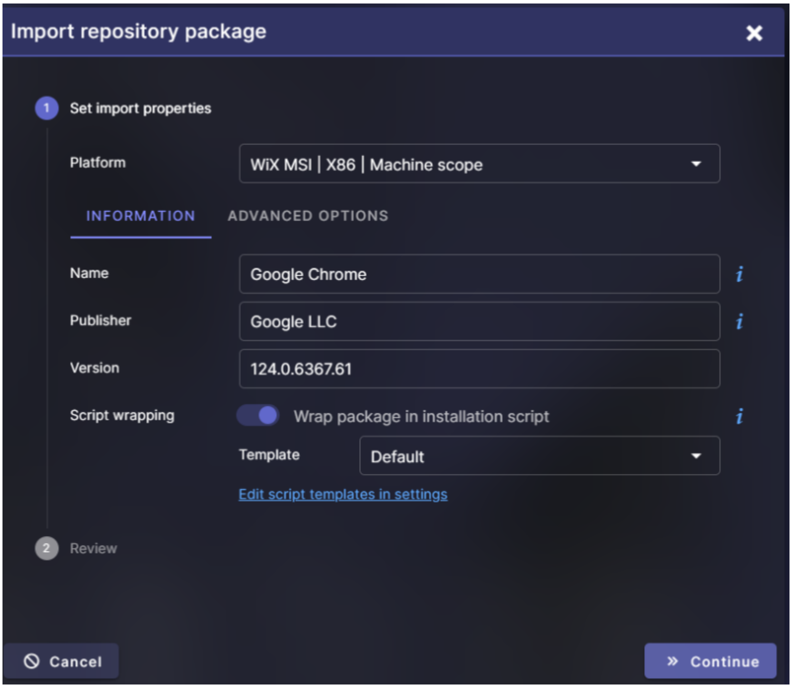
An informative thumbnail shows key details about the latest version and the package source, whether Winget or MS Store. A green badge marks the application as "special," indicating that any available update has passed Robopack's thorough verification process. When you select an application, you can access detailed information about the package, including a list of all available versions for download.



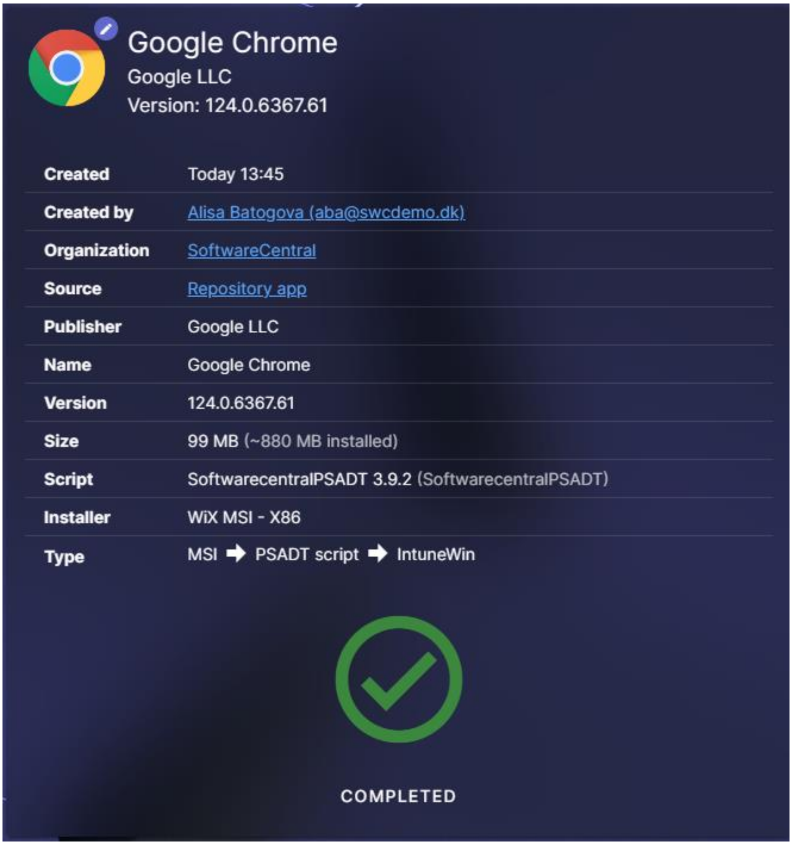
If an application package has been imported before, Robopack provides an overview of its update history and offer the ability to delve into the details of each earlier version.



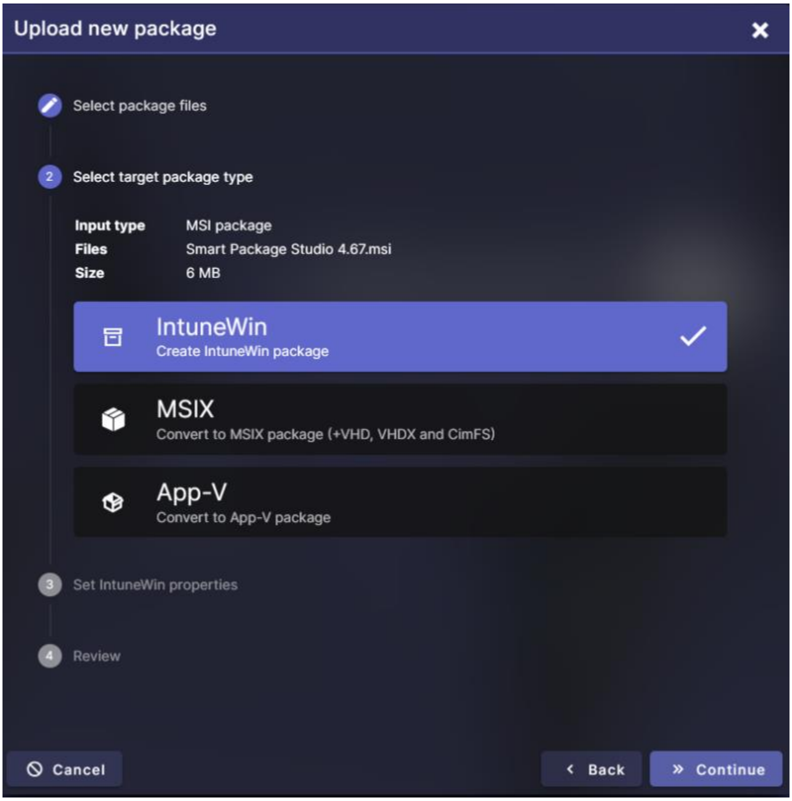
Before importing an application Robopack allows users to verify and, if necessary, customize wrapping settings.



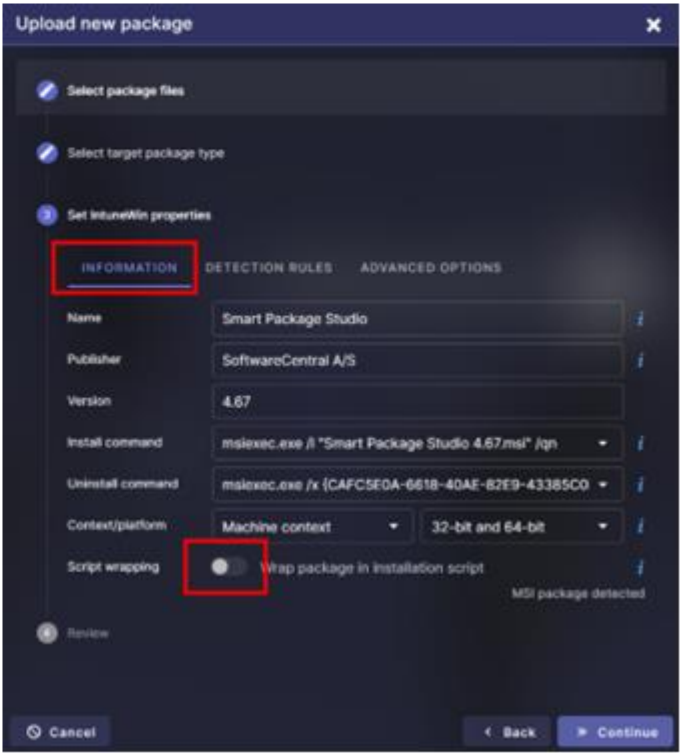
Follow on import status and enjoy auto-conversion to IntuneWin.



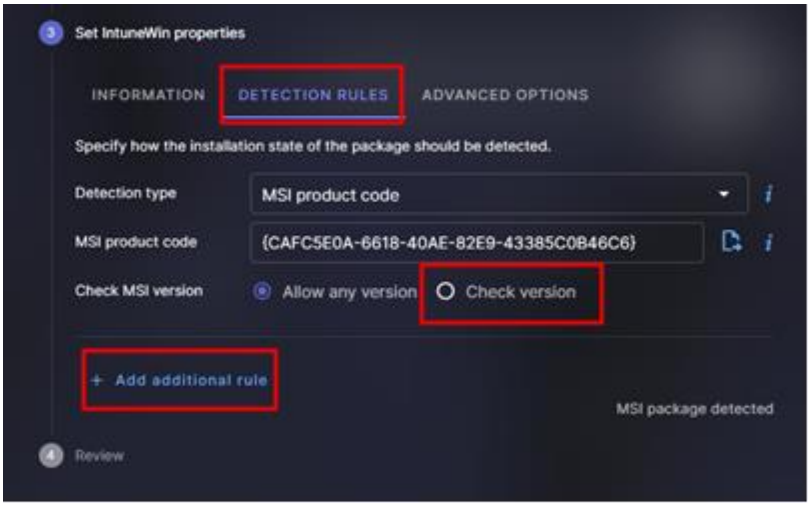
Similar to manually uploaded packages, they can be auto-imported to Intune, downloaded (including both the source and newly created packages), or pushed through a Wave Deployment (Intune-only).
If your applications are not listed in the catalog, you can configure custom applications so all of your applications are centrally managed and monitored within a single portal.



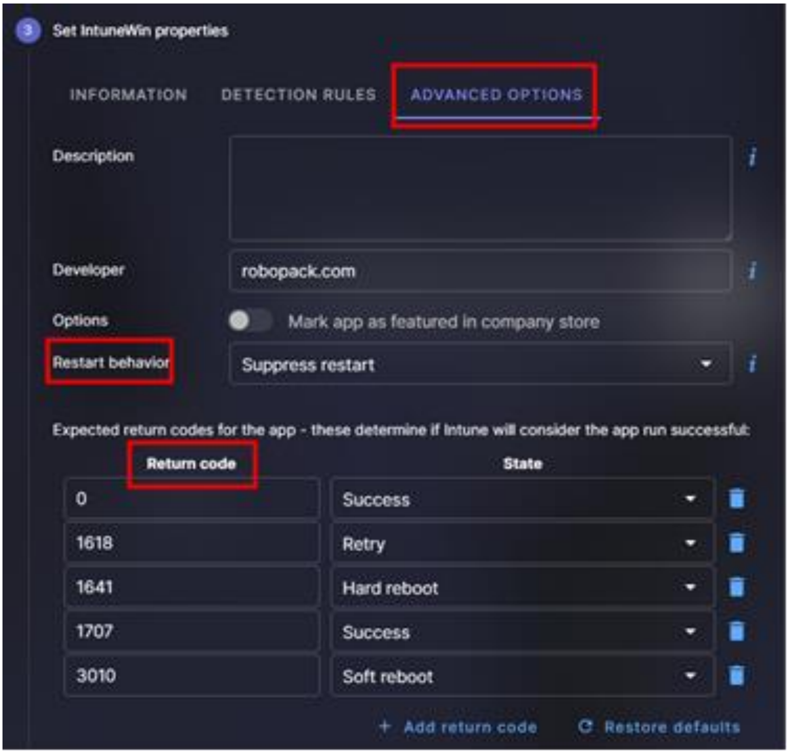
The Information tab allows you to update package details, choose the necessary platform, and, if applicable, select an installation script for wrapping. You can opt for either the default script or a custom one.



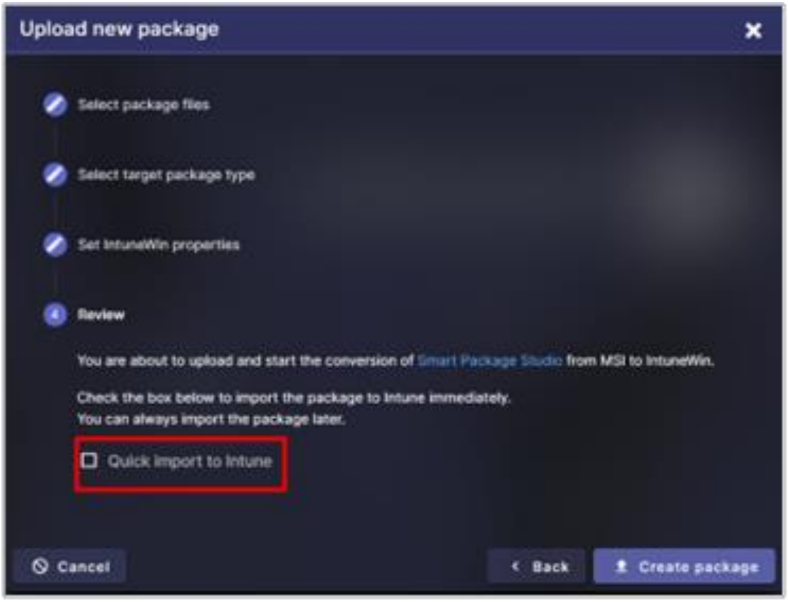
Detection Rules: lets you select the detection type, such as MSI product code, file or directory, or registry value or key. You can also specify if a particular MSI version needs to be checked.



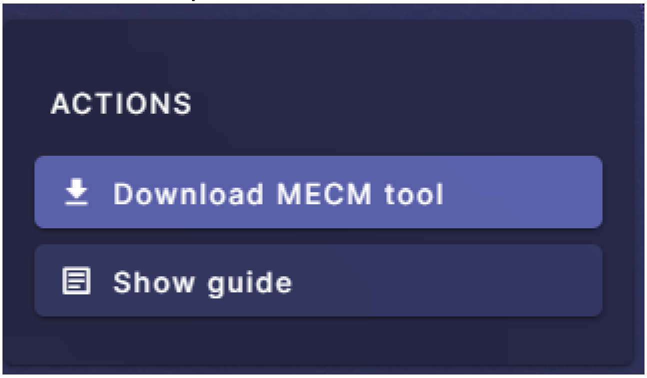
Advanced Options: lets you add a description to display in the Company Portal, set restart behavior (Suppress or Force restart), and specification of expected return codes.



Set IntuneWin properties
INFORMATION
DETECTION RULES
ADVANCED OPTIONS
Description
Developer
robopack.com
Options
Mark app as featured in company store
Restart behavior
Suppress restart
Expected return codes for the app - these determine if Intune will consider the app run successful:
Return code
State
0
Success
1618
Retry
1641
Hard reboot
1707
Success
3010
Soft reboot
Add return code
Restore defaults



For those migrating from Configuration Manager, Robopack's migration tool has you covered. It can extract your applications, package them into Win32 format, and deploy them directly into your Intune environment. It truly serves as a one-stop shop for application management. The MECM tool is standalone, requiring no server installation. Simply download it, run the process, and delete it once you're finished.



For those of you who have applications already in Intune, either manually packaged, or from an alternative package manager, Robopack can detect these applications and bring them under management without any impact on the end-user and even clean up your environment afterwards. Best of all, Robopack is fully cloud-based, requiring no on-premises infrastructure. It integrates seamlessly with Entra SSO and is multi-tenant out of the box, allowing you to manage all your applications across tenants from a single interface.

That’s your applications nicely managed. Your security team will be happy, your users will be happy, and your life will be easier. But what about Intune itself?
This is where the new Intune Manager tool from SoftwareCentral steps in to take your Intune Management to the next level.

If you have experience managing servers and Active Directory, you know the importance of taking backups before making changes. So why should editing Intune policies, which affect all your devices, be any different? What if you had a tool to back up an entire tenant configuration and restore it if issues arise? What if the same tool could deploy a single policy across multiple tenants? The Intune Manage tool offers these features and more.
Beyond backup and restore, you can quickly deploy a baseline template—either one we provide or your own—to set up a new tenant in minutes. Imagine onboarding a new customer, configuring their tenant, and completing a backup all within an hour!

That’s not all. If you manage multiple customer tenants, you need to know if a tenant falls out of sync with your default policies so you can sort it before it causes issues. Drift management will alert you if the policies are changed from your default values and give you the ability to quickly revert.

As with all good enterprise products, every action is audited, and you can enforce change control requirements to ensure that all actions are properly documented with valid references.

The tool is multi-tenant by design, with full RBAC controls so you can define who has access to each customer tenant and what they can do within them.

To make it as cost-neutral as possible, your licenses are regularly reviewed. You'll receive alerts for unused licenses, helping you save on costs.

Link this with Robopack and you have true multi-tenant management from a single pane of glass and when you are managing multiple tenants, deploying policies from one central location without having to navigate to the Intune console for each tenant will free up a significant amount of time which can be used for other billable tasks.
Let Robopack and SoftwareCentral handle the mundane day-to-day activities so you can concentrate on revenue generation.

Chapter 7: Understanding on Copilots and LLM

In the ever-evolving landscape of technology, one of the most groundbreaking advancements has been the integration of Copilots across various products. These advanced automated assistants are not just a fleeting trend; they represent a significant shift in how we interact with applications, the internet, and even our operating systems. In this chapter, we will explore the world of Copilots and clarify everything you need to know about them. It may feel like Copilots are appearing in every Microsoft product, which is an exciting development, but there is still a journey ahead before all of them are fully optimized.

What is a Copilot?

In essence, a Copilot is an AI-powered tool designed to assist users by providing intelligent suggestions, automating tasks, and enhancing productivity across various platforms. Copilots are deeply integrated into applications, operating systems, and the cloud, making them more than just virtual assistants—they are companions that help users navigate complex tasks with ease.

Let's make it simple by asking ChatGPT:

Microsoft Copilot is an AI-powered tool designed to assist developers in writing and maintaining code. It is integrated into various Microsoft products, including Visual Studio and GitHub, where it’s known as GitHub Copilot. This tool leverages machine learning models, trained on a vast corpus of source code, to provide suggestions for code completion, debugging, and even writing whole blocks of code.

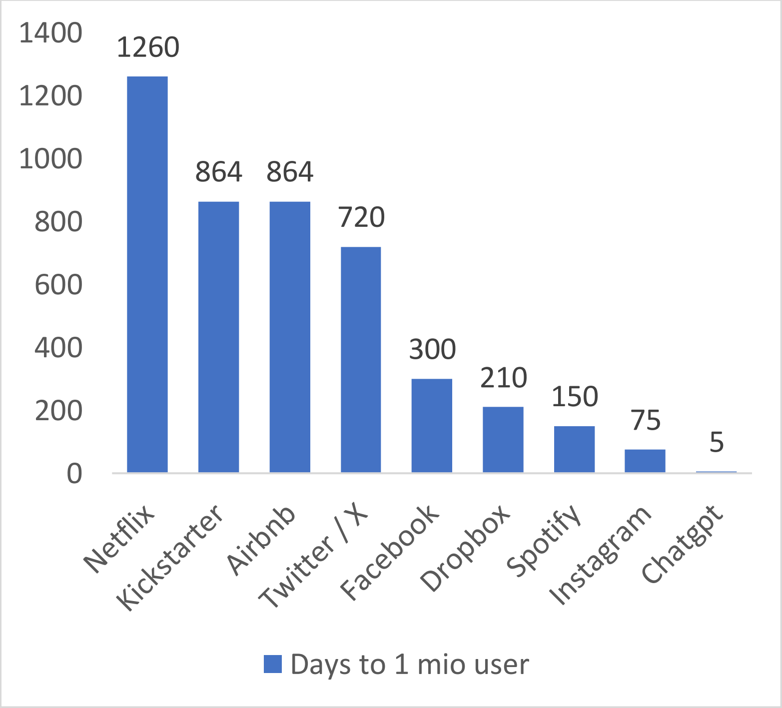
While the quote above describes GitHub Copilot, the concept of a Copilot extends far beyond development tools. In a broader sense, Copilots are redefining how we interact with the internet, our PCs, and digital tools. They assist in a range of activities, from researching online to generating presentations, automating tasks, and even writing code.

A Copilot functions as a virtual assistant, engaging in natural conversations and seamlessly integrating into various tools. For example, **Windows Copilot** can initiate apps or set timers, enabling smoother interaction with the operating system. These Copilots possess deep knowledge and access to vast information, enabling them to enhance user productivity by simplifying complex tasks. We will examine these co-pilots in greater detail later in this chapter.

The Rapid Adoption of Copilots

To understand the impact of technologies like Copilot, consider the speed of their adoption. Mobile phones took 16 years to reach 100 million users. Instagram reached the same milestone in just 30 months, and Google Translate in 78 months. However, ChatGPT, a Large Language Model (LLM) that powers many Copilot experiences, reached 100 million users in just 2.5 months / 1mio in 5 days. This rapid adoption underscores the immense potential and user demand for AI-powered assistants.

Here you can find an overview of big tools how long it took to get the first million users.



One of the pioneers in this space is **GitHub Copilot**, which is specifically designed to aid developers by providing code suggestions, automating repetitive coding tasks, and even helping with debugging. This is just one example of how Copilots are transforming the way professionals work across industries.
But bevor we jump into copilots I will talk very in detail about LLMs.

The Journey of Copilots: Current State and Future Potential

While Copilots are already making a significant impact in various products, there is still a journey ahead to refine and optimize these tools. Microsoft and other tech giants are continuously working to improve the accuracy, reliability, and integration of Copilots across their ecosystems.

The ultimate goal is to create a seamless experience where Copilots not only assist with specific tasks but also proactively anticipate user needs, offer contextual recommendations, and facilitate more intuitive interaction with technology.

As these technologies evolve, we can expect to see even more sophisticated use cases for Copilots, including deeper integration with AI and machine learning models, improved personalization, and expanded capabilities across more domains.

In the next chapters, we will also explore the various types of Copilots, their use cases, and how they integrate with different Microsoft products. We will also delve deeper into the underlying technologies that power these advanced assistants and examine their implications for the future of work and automation.

Which copilots does Microsoft offer?

The answer is a lot. Microsoft has taken significant strides in integrating artificial intelligence (AI) into its ecosystem to improve productivity, streamline workflows, and enhance creativity across its products and services. Under the banner of `Copilot`, Microsoft provides AI-powered assistants tailored to specific applications and tasks. These Copilots are powered by advanced models, such as OpenAI's GPT-4o, integrated seamlessly into the tools many organizations and individuals already rely on.

It is not possible to list and describe all of them, we will discuss the most important ones. Below is an overview of the key Microsoft Copilots available today:

Microsoft 365 Copilot

Microsoft 365 Copilot is embedded across Microsoft 365 apps, including Word, Excel, PowerPoint, Outlook, and Teams. It acts as a productivity assistant, helping users create content, analyze data, and collaborate more efficiently.

- **In Word:** Generate, edit, summarize, and refine text. Copilot can assist with drafting reports, proposals, and creative content, saving users significant time.

- **In Excel:** Analyze data, create insights, and automate complex tasks such as generating pivot tables or creating forecasts. It simplifies data-driven decision-making.
- **In PowerPoint:** Transform ideas into visually appealing presentations by generating slides, images, and content with simple prompts.
- **In Outlook:** Manage emails, summarize threads, and assist in drafting and prioritizing replies to improve email efficiency.
- **In Teams:** Summarize meetings, identify key decisions, and keep team members on track by generating insights during and after discussions.

Microsoft 365 Copilot serves as a comprehensive assistant to enhance personal and team productivity.

GitHub Copilot

GitHub Copilot, referred to as the `AI pair programmer`, is an AI-powered coding assistant integrated into popular code editors such as Visual Studio Code.

- **Key Features:** It suggests code snippets, autocompletes functions, and even generates entire blocks of code based on natural language prompts.
- **Languages Supported:** GitHub Copilot supports a wide range of programming languages, including Python, JavaScript, Java, C#, and more.

- **Benefits:** It reduces repetitive coding tasks, helps developers learn new languages, and accelerates software development workflows.

Developers worldwide use GitHub Copilot to enhance coding efficiency and creativity. For real-life use cases, check out this free e-book that I also contributed to:

https://www.microsoft.com/de-de/techwiese/news/kostenfreies-e-book-tipps-und-tricks-zu-github-copilot-vol-2.aspx

Azure AI Studio and Azure OpenAI Copilot

Azure OpenAI Copilot integrates OpenAI's language models into Microsoft Azure to support business-specific applications.

- **Customization:** Organizations can build their own AI copilots tailored to specific use cases using Azure OpenAI Service.
- **Applications:** Whether it's automating customer support, generating insights from documents, or summarizing information, businesses can design copilots for their unique needs.
- **Azure AI Studio:** A platform that provides tools for developing, testing, and deploying AI models for various enterprise applications.

Azure OpenAI Copilot empowers businesses to integrate AI-powered workflows into their operations.

Dynamics 365 Copilot

Dynamics 365 Copilot brings AI capabilities to Microsoft's suite of business applications, such as Dynamics 365 Sales, Customer Service, and Supply Chain Management.

- **For Sales Teams:** It provides AI-generated insights, drafts personalized email responses, and assists in creating proposals based on CRM data.
- **For Customer Service:** Dynamics 365 Copilot summarizes customer conversations, suggests responses, and automates common tasks to enhance support efficiency.
- **For Supply Chain Management:** It predicts demand, streamlines operations, and assists in identifying potential disruptions with AI-powered insights.

Dynamics 365 Copilot transforms enterprise workflows, improving both efficiency and decision-making.

Power Platform Copilot

Power Platform Copilot helps users build applications, automate workflows, and analyze data without requiring deep technical expertise.

- **In Power Apps:** Copilot assists in building applications by suggesting components and generating logic with natural language inputs.
- **In Power Automate:** It helps automate workflows by generating flows and processes based on descriptions of tasks.

- **In Power BI:** Copilot enhances data visualization by generating insights, summaries, and visuals from datasets.

Power Platform Copilot enables businesses and users to innovate and automate processes quickly and efficiently.

Windows Copilot

Windows Copilot is designed to integrate AI assistance directly into the Windows operating system.

- **Key Features:** It provides system-level assistance, such as adjusting settings, launching applications, and summarizing content from web pages or documents.
- **Productivity Enhancements:** By leveraging integrations with Microsoft Edge and other applications, Windows Copilot makes multitasking smoother and more efficient.

Windows Copilot offers a centralized AI-powered assistant for the operating system, improving the user experience and productivity.

Security Copilot

Microsoft Security Copilot is tailored to assist security professionals in responding to threats and managing security operations.

- **Incident Response:** It summarizes security incidents, correlates threat intelligence, and helps create actionable remediation plans.
- **Threat Analysis:** By leveraging AI, Security Copilot assists in identifying risks and patterns across large datasets, providing faster detection and response.
- **Many more use cases**

Security Copilot empowers organizations to bolster their cybersecurity posture with AI-powered insights and automation. We will have a deeper look later into this.

The Role of LLMs in Copilots

LLMs power the intelligence behind Copilots, enabling them to understand natural language input, contextualize user requests, and generate relevant and helpful responses. For instance, when a developer is writing code, GitHub Copilot can understand the context of the code and suggest the next line or even complete an entire function based on best practices and learned patterns.

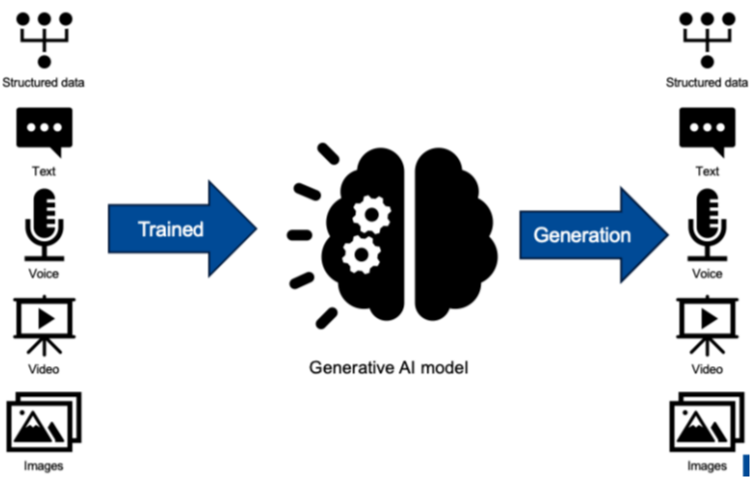
Similarly, Copilots in productivity tools like **Microsoft Word** or **Excel** can assist in drafting documents, generating summaries, or automating data analysis tasks. They do this by leveraging their deep understanding of language and patterns learned from vast amounts of data.

In essence, LLMs make Copilots more than just command-based automation tools—they turn them into conversational assistants capable of handling complex, context-sensitive tasks.

What is a Large Language Model?

To understand how Copilots function, it's important to grasp the concept of Large Language Models (LLMs). These models are the engine behind most Copilot functionalities, enabling them to process natural language and provide intelligent responses.

A **Large Language Model** is a sophisticated type of transformer model based on the innovative transformer architecture. Introduced in the groundbreaking paper *"Attention Is All You Need"* in 2017, the transformer model revolutionized the field of natural language processing (NLP). The core innovation of transformer models is the **attention mechanism**, which allows the model to weigh the importance of different words within a sentence, thereby understanding context more effectively than previous models that processed words sequentially.



Breaking Down the Basics

The significance of transformer models lies in their ability to consider context holistically rather than just focusing on individual words in isolation. This allows LLMs to generate coherent and contextually relevant responses that feel more natural and human-like.

Large Language Models (LLMs) are vast in scale and trained on enormous datasets that often include a significant portion of publicly available data from the internet, books, articles, and other sources. This extensive training enables these models to generate, comprehend, and respond to human language with remarkable fluency and accuracy.

At their core, LLMs are transformer-based neural networks with an input layer, multiple hidden layers, and an output layer. The scale of these models is defined by the number of parameters they have:

- **GPT-2** has 1.5 billion parameters.
- **GPT-3** expands to 175 billion parameters.
- **GPT-4** grows exponentially to 1.4 trillion parameters.

This graphic from MS Build shows the dimensions of the next generation of OpenAI models.

This rapid increase in scale illustrates the swift advancements in the field of NLP and AI. Training such massive models requires substantial computational resources, particularly in terms of GPU power and distributed computing.

As these models grow in scale and complexity, so too does their capacity for specialized expertise. Early language models were broad and somewhat limited in their ability to adapt deeply to niche topics. However, as researchers continue to refine training techniques and expand the variety and depth of their datasets, LLMs are becoming adept at handling domain-specific tasks with striking precision.

For instance, newer generations of models are not just capable of delivering eloquent general knowledge but can also offer nuanced guidance across a wide range of specialized fields. In medicine, finance, law, and countless other professional domains, modern LLMs are demonstrating abilities that closely mirror the thought processes and decision-making skills of seasoned experts.

This shift—from broad generalization to specialized mastery—is fueled by advances in fine-tuning methods, prompt engineering, and the growing synergy between model parameters and high-quality training data.

With today's models, such as GPT-o1, the performance can be so exceptional in certain specialized areas that it effectively hits every doctor in his own area becoming an invaluable tool for professional decision support. Although these models are not replacements for the nuanced judgment and ethical considerations of human experts, they serve as powerful allies—offering rapid insights, sifting through extensive research, and presenting information in a coherent, context-rich manner.

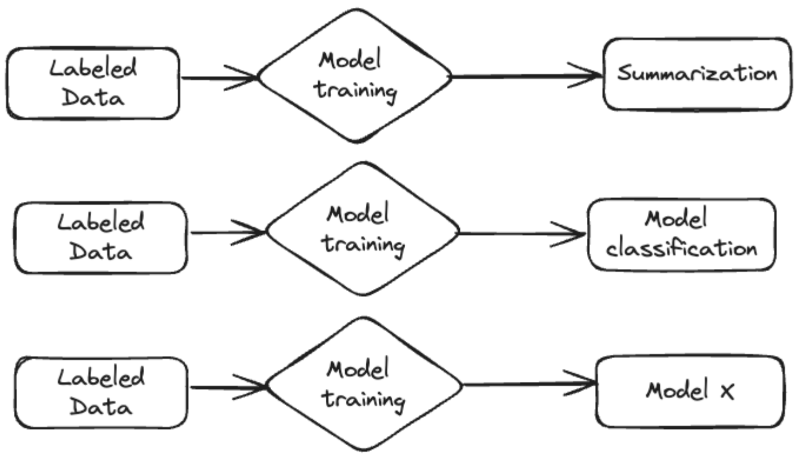
In essence, the evolution of Large Language Models marks a pivotal moment in AI. They not only raise the bar for what is possible in natural language processing but also open doors to new applications and breakthroughs. As we continue to push the boundaries of model size, training methodologies, and interpretability, the future of language-driven AI promises to be as transformative as it is expansive.

Understanding the Differences: LLMs vs. Classical NLP Models

Classical NLP Models and **Large Language Models (LLMs)** represent two distinct approaches to natural language processing, each with its own strengths and use cases.

<u>Classical NLP Models</u>

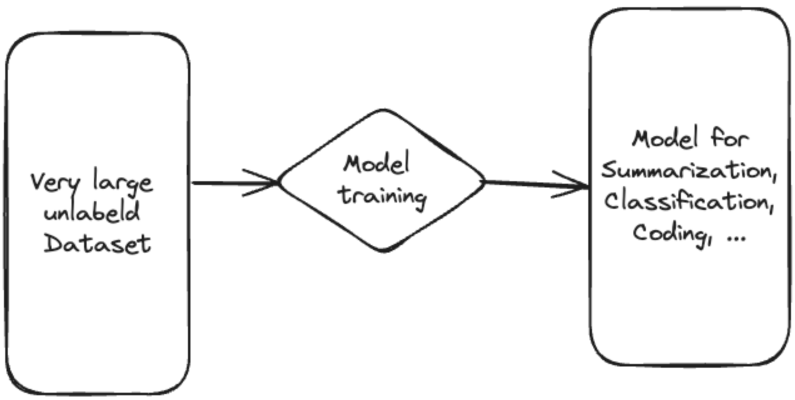
These models are typically designed for specific tasks, such as text classification, entity detection, sentiment analysis, or part-of-speech tagging. Each model is individually developed and trained to excel in its particular function. For example, a text classification model might be optimized to categorize emails into spam or non-spam. While classical NLP models perform exceptionally well in their designated tasks, they often require separate development efforts and may not generalize well across multiple NLP applications. They are also limited in their ability to handle complex, contextual tasks beyond their specific training.



Large Language Models (LLMs)

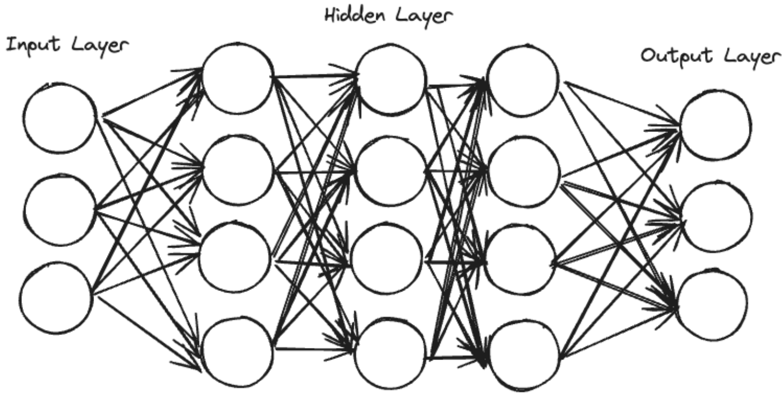
In contrast, LLMs like GPT-4o are trained on massive datasets, encompassing a broad range of information and domains. LLMs utilize transformer architectures, which allow them to understand and generate text with remarkable coherence and context-awareness. Unlike classical NLP models, LLMs can handle various tasks without needing task-specific retraining. Their ability to generalize across different domains makes them highly versatile and capable of performing anything from text generation to complex problem-solving, all within the same model.

The key difference between classical NLP models and LLMs lies in their scope and adaptability. While classical models excel at specific tasks, LLMs offer broader applicability across a range of tasks and domains, making them more versatile tools in the field of AI and machine learning.



How Does a Large Language Model Work

Large Language Models (LLMs), such as GPT, operate through a structured process involving multiple layers of neural networks. These layers are responsible for transforming the input data (usually text) into meaningful responses. Let’s break down how each component—**Input Layer**, **Hidden Layers**, and **Output Layer**—contributes to the overall functioning of an LLM.



Input Layer

The **Input Layer** is the starting point of the model's processing pipeline. This is where raw data, such as text, enters the model. In the case of a language model, the input is usually a sequence of tokens (words or subwords) that represent the text. The input layer converts this text into a numerical form that the model can process, often using techniques like word embeddings, which map each word or token to a vector of real numbers.

For example, when you input the sentence "What is Microsoft Intune?" the input layer converts each word in the sentence into a vector of numbers representing its meaning based on the model's training.

Hidden Layers

The **Hidden Layers** are where the heavy lifting of the model takes place. In LLMs, these hidden layers consist of transformer blocks, which are responsible for processing the input data and capturing complex patterns, relationships, and contextual information.

Each hidden layer applies a series of transformations to the input data using mechanisms such as **self-attention** and **feedforward neural networks**:

- **Self-Attention**: This mechanism allows the model to weigh the importance of each token in relation to others in the sequence. For example, in the sentence "The cat sat on the mat," self-attention helps the model understand the relationships between words like "cat" and "sat" or "on" and "mat." It assigns different levels of importance to each word based on the context.
- **Feedforward Networks**: After the self-attention mechanism processes the data, feedforward networks perform additional transformations on the data, refining the model's understanding of the relationships and patterns within the input.

These hidden layers are stacked in multiple layers (often 12, 24, or even more), allowing the model to progressively refine its understanding and capture increasingly complex patterns in the input data.

Output Layer

The **Output Layer** is the final stage of the model's processing. After the data has been transformed and refined by the hidden layers, the output layer generates the final response based on the model's understanding.

In an LLM, the output is usually a sequence of tokens (words or subwords) representing the model's response.

The output layer uses probabilities to determine which tokens to generate next based on the patterns it has learned from the input and hidden layers. The model selects the token with the highest probability and generates the corresponding word, continuing this process until the response is complete.

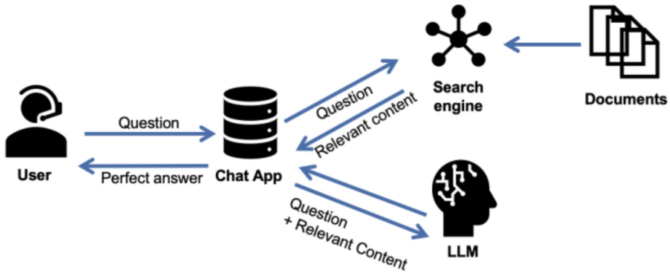
For instance, when you ask, "What is Microsoft Intune?" the output layer generates a coherent and contextually relevant response by selecting words that most accurately answer the question.

What is grounding?

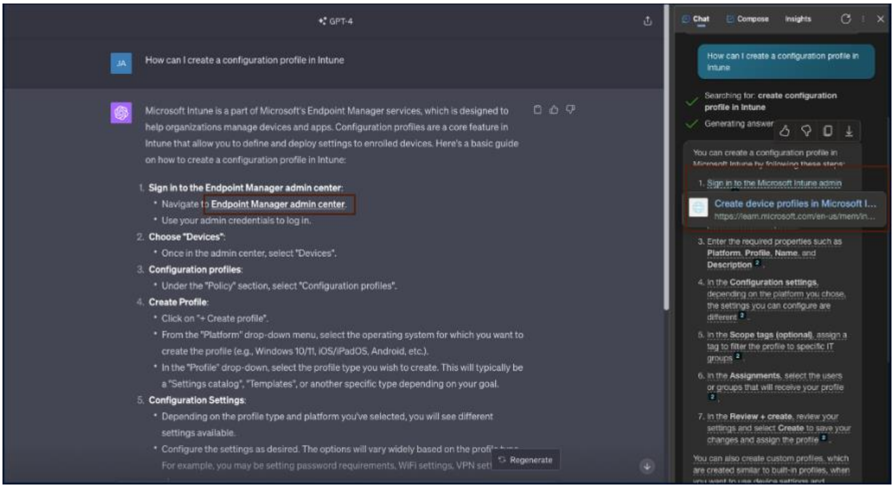
When working with Large Language Models (LLMs), it's crucial to recognize that they aren't all-knowing repositories. Despite their vast training, they can sometimes produce outdated, incorrect, or irrelevant information, especially when dealing with specific internal processes, guidelines, or rapidly changing fields. This phenomenon is known as "hallucination," where the model generates plausible sounding but incorrect or fabricated information.

This is where **grounding** comes into play.
Grounding in the context of LLMs refers to the model’s ability to connect its general knowledge with real-world contexts and up-to-date information. Grounding often involves using external data sources or user-provided context to inform the model's responses, ensuring that the output is relevant, accurate, and appropriately contextualized. A common method for grounding is

Retrieval-Augmented Generation (RAG), which integrates external data sources to enhance the model's responses.



For example, if you ask ChatGPT, "How can I create a configuration profile in Intune?" it might reference outdated terms like "Endpoint Manager admin center" instead of the current "Microsoft Intune Admin Center." However, a grounded model, like Microsoft's Bing chat, would first perform a web search to retrieve up-to-date information before generating a response, ensuring accuracy by grounding the response in real-time data. Grounding improves the reliability of AI systems, especially in professional and technical environments where up-to-date, precise information is crucial



What is prompt engineering?

Prompt engineering is the art of crafting effective prompts to communicate with Large Language Models (LLMs) and other AI systems. A **prompt** is a short instruction or input provided to the AI to initiate a specific action or response. Prompts guide the AI on what to do, whether it's answering a question, generating text, or even describing an image.

Prompt engineering involves understanding how the AI interprets and responds to different inputs and then optimizing the prompt to achieve the desired outcome. Effective prompt engineering can significantly improve the accuracy, relevance, and creativity of the AI's responses.

For example, consider two different prompts:

1. **Basic Prompt**: "Write an article about AI."
2. **Refined Prompt**: "Write a 500-word article discussing the impact of AI on the healthcare

industry, focusing on patient care and medical research."

The refined prompt provides the AI with clearer guidance, resulting in a more targeted and relevant response.

Example of Prompt Engineering in Practice:

- **Task**: Generate a report summary using a large dataset.
- **Prompt**: "Analyze the dataset on employee productivity and provide a summary highlighting key trends and areas for improvement, specifically focusing on remote work performance."

This prompt directs the AI to focus on specific aspects of the dataset, leading to a more precise and useful output.

Prompt engineering is essential when working with LLMs, as it helps ensure that the AI produces responses that meet specific requirements, particularly in professional and technical contexts.

What is fine-tuning?

Fine-tuning involves taking a pre-trained Large Language Model and further training it on a specialized dataset to adapt its responses to a particular domain or task. Fine-tuning allows the model to develop deeper expertise in specific areas beyond its general training.

For example, an LLM that has been pre-trained on general internet text can be fine-tuned on a dataset of medical research papers to improve its ability to generate accurate and relevant medical information.

Fine-tuning requires significant computational resources and time, making it a more resource-intensive process compared to grounding. It involves retraining the model on a specific dataset to enhance its performance in a particular domain. However, once fine-tuned, the model becomes highly specialized and capable of delivering expert-level responses in that area.

Fine-tuning vs. grounding

When considering how to customize an LLM for specific use cases, there's a hierarchy of approaches to consider:

1. **Grounding and Prompt Engineering**:
    - For the majority of use cases, grounding and prompt engineering are sufficient to guide the LLM in producing accurate and contextually relevant responses. Grounding is highly flexible, cost-effective, and quick to implement, as it doesn't require additional model training. Instead, it uses external data sources to inform the model's responses. Prompt engineering, on the other hand, refines how you ask questions to extract the most accurate and useful information from the model.
2. **Fine-Tuning**:
    - Fine-tuning comes into play when grounding and prompt engineering are not enough to meet specific requirements. Fine-tuning allows for deeper customization of the model, enabling it to perform highly specialized tasks. However, it is resource-intensive and requires retraining the model whenever the underlying data changes. As a

result, fine-tuning should be reserved for scenarios where precise, domain-specific responses are essential and the use case cannot be fulfilled through grounding alone.

3. **Training a New Model**:
    - In some extreme cases, neither grounding nor fine-tuning may suffice, leading to the need to train a completely new model. This approach requires the highest level of resources, including large datasets, significant computational power, and ongoing maintenance.

Comparison: Fine-Tuning vs. Prompt Grounding

Criteria	**LLM Fine-Tuning**	**Prompt Grounding**
Definition	Involves training an LLM on a specific dataset to adapt its responses.	Adding context or data into the prompt to guide the LLM.
Data Requirement	High. Requires a domain-specific dataset for training.	Low. Utilizes existing LLM knowledge with added context.
Flexibility	Once fine-tuned, the model is specialized in the trained domain.	Highly flexible; adapt to various tasks by adjusting the prompt.
Cost and Resources	Resource-intensive; requires additional	Cost-effective and quick; no

	training, computational power, and time.	additional training needed.
Customization Level	Deep customization for specific tasks or domains.	Limited customization; relies on prompt design.
Time to Implement	Time-consuming due to training and optimization.	Quick to implement; depends on crafting the right prompt.
Scalability	Scalable but may need retraining for different tasks.	Highly scalable; adjust prompts for various tasks with ease.
Use Cases	Ideal for specialized tasks where in-depth knowledge is essential.	Best for general tasks or when adapting quickly to different requirements.

Prompt Engineering: A Full Tutorial with Techniques and Examples

Prompt engineering is an essential skill when working with Large Language Models (LLMs) like GPT. It involves crafting effective prompts to guide the model in generating accurate, relevant, and useful responses. In this tutorial, we'll cover various prompt engineering techniques, ranging from basic prompting to more advanced methods, along with practical examples to demonstrate each technique.

The Anatomy of a Great Prompt

A good prompt usually has four main components:

- **Instruction**: Define what you want the AI to do.
- **Context**: Provide relevant background information.
- **Input/Question**: Specify the query or data you want addressed.
- **Output Format**: Describe how you want the response structured.

Additionally, a **system prompt** can be used to set the AI's behavior throughout the interaction.

Example Breakdown:

- **System Prompt**: "You are a helpful assistant classifying text as 'friendly,' 'negative,' or 'neutral."
- **User Prompt**: "Text: I absolutely love writing prompts! Please classify the sentiment of this text and provide the result in JSON format."

Desired AI Response:

```
{
  "text": "I absolutely love writing prompts!",
   "classification": "friendly"
}
```

Core Techniques for Prompt Engineering

Mastering the basics of prompt engineering is essential for effective interaction with AI models. These foundational techniques ensure that your requests are clear, precise, and easy for the AI to interpret. Let's explore these techniques with practical examples:

Be Precise but Concise

The AI performs better when your prompts are direct and to the point. Avoid unnecessary words that could introduce ambiguity.

- **Good**: “Summarize the key points of this article on climate change.”
- **Bad**: “Can you, maybe, if it’s not too much trouble, tell me what this article is about?”

Provide Context

Providing relevant background information helps the AI better understand your request and deliver more accurate results.

- **Good**: “As a historian, explain the significance of the Renaissance period.”
- **Bad**: “Tell me about the Renaissance.”
- Including context aligns the response with your specific needs, making it more tailored and insightful.

Use Clear Structures

Clearly defined structures, such as bullet points, numbered lists, or specified output formats, improve the readability and utility of the AI’s response.

- **Example**: “List the top 5 benefits of exercise in bullet points.”

By following these techniques, you can guide AI models like GPT to produce responses that are not only accurate but also easy to understand and act upon.

How to Write Better Prompts

As mentioned, Prompt engineering is a critical skill when working with Large Language Models (LLMs). Writing effective prompts requires thought and precision to ensure that the AI understands your request and provides accurate, relevant information. Here are some tips to improve your prompt-writing skills:

1. **Be Specific**: Clearly define what information you are looking for. Vague prompts can lead to inaccurate or incomplete responses.
    - **Bad Prompt**: "What's H493958343?"
    - **Better Prompt**: "Generate a short summary of the device H493958343 and include general information as well as the assigned configurations."

2. **Provide Context**: Adding context helps the model understand the scope of your question.
    - **Example**: "In the context of recent security incidents, summarize the impact of malware on **[DEVICE_NAME]**."

3. **Use Step-by-Step Instructions**: Break down complex requests into manageable steps so the model can process them sequentially.
    - **Example**: "First, list all devices with failed security updates. Then, provide a brief analysis of potential risks."

4. **Leverage Prompt Templates**: Develop reusable templates for common queries. This not only saves time but ensures consistency in your requests.

- **Example Template**: "Summarize [OBJECT_NAME] with details on [SPECIFIC_ATTRIBUTE]."

5. **Experiment and Iterate**: Don't be afraid to refine your prompts based on the responses you get. Experimenting with different phrasings and structures can help you achieve better results.

Advanced Prompt Techniques

To unlock the full potential of prompt engineering, it's essential to tailor prompts for specific roles and tasks. Here's how to adapt advanced techniques for various security personas:

<u>Basic Prompting</u>

Basic prompting involves providing the AI with simple instructions or questions. This is the most straightforward way to interact with an LLM but may require refinement for more complex tasks.

Example 1: Simple Question

- **Prompt**: "What is Microsoft Intune?"
- **Response**: Microsoft Intune is a cloud-based service that focuses on mobile device management (MDM) and mobile application management (MAM). It helps organizations manage the devices and apps used by employees.

Example 2: Requesting a Summary

- **Prompt**: "Summarize the key features of Microsoft Intune."
- **Response**: Microsoft Intune provides features like mobile device management (MDM), mobile application management (MAM), app protection policies, and integration with Azure Active Directory and Microsoft 365 for comprehensive security and compliance management.

While basic prompting can provide useful information, it often benefits from additional context or refinement to get more specific or nuanced responses.

Contextual Prompting

Enhance specificity by embedding contextual information into your prompt. Context can include domain-specific knowledge, use cases, or scenarios.

- **Prompt**: "In the context of remote work, how does Intune enhance device security, especially for BYOD (Bring Your Own Device) policies?"
- **Response**: Intune enhances device security for remote work and BYOD policies by enabling secure access to corporate resources through conditional access, enforcing device compliance policies, managing app protection, and ensuring data encryption and wipe capabilities on personal devices without intruding on personal data.

Contextual prompts are instrumental in obtaining responses that are aligned with real-world scenarios and challenges.

Role Prompting

Assigning a specific role or persona to the AI ensures its responses are tailored to the intended audience. This technique is particularly valuable for simulating expert advice or focusing the AI's expertise on a given domain.

For Security Admin/ SOC Analyst

- **Prompt**: "As a cybersecurity expert, summarize the top threats detected in the last 24 hours, including their severity and affected systems."
- **Response**: "The most critical threats detected in the last 24 hours include a ransomware attack affecting 10 high-priority devices (severity: critical), phishing attempts on executive email accounts (severity: high), and unauthorized access attempts to the internal database (severity: medium). Recommended actions include isolating compromised devices, reviewing email security protocols, and enhancing database access policies."

For Developer

- **Prompt**: "As a software developer, explain how to integrate Microsoft Intune's API into an enterprise application for automated device enrollment."
- **Response**: "To integrate Microsoft Intune's API, use the Microsoft Graph API. Begin by authenticating using Azure Active Directory OAuth2 tokens. Retrieve the tenant ID and use the `/deviceManagement` endpoint to configure enrollment settings. Test the integration using

RESTful API tools like Postman and implement error handling for robust operations."

For IT Admin (Specialist)

- **Prompt**: "As an IT Admin, provide a step-by-step guide to configure conditional access policies in Intune to restrict access based on device compliance."
- **Response**: "1. Sign in to the Azure AD admin center and navigate to 'Security' > 'Conditional Access.' 2. Click 'New Policy' and name the policy. 3. Select 'Users or Groups' and specify the target group. 4. Under 'Cloud Apps or Actions,' select the required apps. 5. Configure 'Conditions' such as location or device state. 6. Under 'Grant,' select 'Require device to be marked as compliant.' 7. Save and enable the policy."

For Helpdesk

- **Prompt**: "As a helpdesk agent, troubleshoot why a device is non-compliant in Intune and suggest corrective actions."
- **Response**: "1. Verify device enrollment status in the Intune admin portal. 2. Check if compliance policies are applied correctly to the device. 3. Ensure the device meets all compliance requirements (e.g., updated OS, encryption enabled). 4. If unresolved, unenroll and re-enroll the device. 5. Communicate the resolution steps to the user."

For End User

- **Prompt**: “As a guide for end users, explain how to enroll a device in Intune in simple steps.”
- **Response**: “1. Download the Company Portal app from your device’s app store. 2. Open the app and sign in with your work credentials. 3. Follow the prompts to register your device. 4. Complete any required configurations, such as setting up a PIN. 5. Contact your IT department if you encounter issues.””
- Response: Provides a user-friendly walkthrough with visuals and troubleshooting tips.

Chain-of-Thought Prompting

Chain-of-thought prompting encourages the model to break down its reasoning step by step. This is useful for tasks that require logical reasoning or multi-step problem-solving, helping the model clarify its thought process.

Example 1: Breaking Down a Solution

- **Prompt**: “Explain step-by-step how to create and assign a compliance policy in Intune, ensuring devices meet security standards.”
- **Response**:
    - Sign in to the Microsoft Intune admin center.
    - Navigate to “Devices” and select “Compliance policies.”

  - Click on “Create Policy” and choose the platform (e.g., Windows, iOS, Android).
  - Configure the policy settings, such as password requirements, encryption, and device health checks.
  - Name the policy and add a detailed description for clarity.
  - Assign the policy to specific user or device groups based on organizational needs.
  - Review the settings and save the policy.
  - Monitor policy deployment and compliance status through the “Reports” section in the Intune dashboard.

This method ensures clarity, improves execution accuracy, and facilitates easier troubleshooting by breaking down tasks into manageable steps.

Few-Shot Prompting

Few-shot prompting gives the AI a few examples of what you're looking for before asking it to generate a new response. This technique is helpful when you want the model to follow a specific format or pattern.

Example 1: Providing Examples

- **Prompt**: "Here are some examples of compliance policies in Microsoft Intune. Example 1: Require password protection. Example 2: Encrypt data on the device. Now, provide a new example."
- **Response**: Example 3: Enable device lockout after multiple failed login attempts.

Example 2: Structuring Output

- **Prompt**: "List the steps for setting up a new device in Microsoft Intune. Example 1: Enroll the device. Example 2: Assign compliance policies. Now, continue the list."
- **Response**: Example 3: Configure device profiles. Example 4: Deploy necessary applications.

Few-shot prompting helps guide the model toward a specific structure or output format, improving the consistency of the responses.

Zero-Shot Prompting

Zero-shot prompting involves asking the AI to perform a task without providing any examples or specific instructions beforehand. This technique is useful when you want the model to generate something entirely new based on its existing knowledge.

Example 1: Generating a Creative Solution

- **Prompt**: "Create a new feature idea for Microsoft Intune that improves remote device management."
- **Response**: A new feature idea for Microsoft Intune could be a "Remote Device Recovery" option that allows IT administrators to locate, lock, or wipe lost or stolen devices even when they are offline, using a backup battery-powered tracking system.

Zero-shot prompting allows the model to leverage its broad knowledge and creativity to develop novel ideas or solutions.

Instruction-Based Prompting

Instruction-based prompting gives the model a set of explicit instructions to follow. This technique is especially useful for tasks that require the AI to adhere to specific guidelines, such as writing in a particular style or format.

Example 1: Writing in a Formal Style

- **Prompt**: "Write a formal email informing employees about a new security policy enforced by Microsoft Intune."
- **Response**:

Subject: Important Update: New Security Policy Enforced by Microsoft Intune Dear Team, We would like to inform you that, effective immediately, a new security policy will be enforced on all company devices through Microsoft Intune. This policy includes mandatory device encryption, password protection, and compliance checks. Please ensure that your devices meet the necessary requirements to maintain access to corporate resources. Thank you for your cooperation. Best regards, IT Department

By giving explicit instructions, you can control the tone, style, or structure of the AI's output.

Multi-Turn Prompting

Multi-turn prompting involves interacting with the model across several turns to refine the output iteratively. This is useful when the task is complex or requires back-and-forth dialogue to clarify the request or improve the response.

Example 1: Iterative Dialogue

- **Prompt**: "Explain how Microsoft Intune handles app deployment."
- **Response**: Microsoft Intune handles app deployment by allowing administrators to assign and manage applications on enrolled devices. Apps can be assigned as required, available for installation, or uninstalled remotely by the administrator.
- **Follow-up Prompt**: "Can you provide more details on how apps are assigned to different device groups?"
- **Response**: Administrators can assign apps to device groups or user groups in Intune. Apps assigned as "required" will automatically install on all devices in the group, while apps marked as "available" can be installed by users at their convenience through the Company Portal app.

Multi-turn prompting allows for a more conversational approach, where the model's response can be refined over time based on additional input.

Comparative Prompting

Comparative prompting asks the model to compare or contrast two or more concepts, tools, or methods. This is useful for analyzing options or making decisions.

Example 1: Comparing Features

- **Prompt**: "Compare the app deployment features of Microsoft Intune and VMware Workspace ONE."
- **Response**: Microsoft Intune and VMware Workspace ONE both offer app deployment features, but they differ in several ways. Intune integrates closely with Microsoft 365 and Azure Active Directory, offering seamless deployment for Windows apps and deep integration with Microsoft's security tools. Workspace ONE, on the other hand, is more platform-agnostic, supporting a wide range of operating systems and offering advanced features like conditional access and multi-platform deployment. Both solutions provide centralized management, but the choice between them may depend on the specific needs of the organization and the existing infrastructure.

By comparing the two options, the model provides a nuanced perspective that helps with decision-making.

By practicing prompt engineering, you'll be able to extract more useful and accurate information from Security Copilot and other AI-driven tools.

Useful Example Prompts for Security Copilot

Microsoft Security Copilot is a powerful tool designed to simplify and enhance your security workflows (We will investigate this more in detail below). By leveraging intelligent prompts, users can quickly retrieve critical insights, automate repetitive tasks, and respond effectively to security incidents. Below are some practical examples of prompts tailored to specific personas and tasks, providing value across various roles and scenarios.

1. **Device-Specific Queries**:
   - "Can you give me a summary of the device **[HOSTNAME]**?"
   - "Show me the security compliance status of the device **[DEVICE_NAME]**."

2. **User-Specific Queries**:
   - "Tell me everything about **[UPN/Username]** in a markdown table."
   - "What recent security incidents are associated with **[USER_NAME]**?"

3. **Configuration Insights**:
   - "Generate a list of all devices with non-compliant configurations."
   - "Provide an overview of all security policies applied to **[DEVICE_NAME]**."

4. **Threat Detection and Response**:
   - "Summarize the most critical threats detected in the last 24 hours."
   - "How many devices have active malware incidents?"

5. **Proactive Security Tasks**:
   - "Identify any security vulnerabilities that need immediate attention."
   - "Give me a list of all unpatched systems in the organization."

6. **General Security Overviews**:
    - "Provide a security posture summary for the organization."
    - "List all recent security alerts and their resolutions."

Here are also some prompts from the view of the different personas:

Security Admin/SOC Analyst

- "Identify users who triggered the most security incidents in the last 30 days."
- "Generate a detailed report on phishing attempts blocked in the last week."
- "Provide threat intelligence on the latest ransomware attacks."

Developer

- "List all security vulnerabilities from the application [APP_NAME]."
- "Check for compliance issues in the containerized environment running [APP_NAME]."

IT Admin (Generalist/Specialist)

- "Give me a list of all devices missing endpoint protection software."
- "Summarize patch compliance status across all servers."
- "Generate a report of all users with admin privileges and their activity logs."

Helpdesk

- "Provide a step-by-step guide to remediate a malware infection on [DEVICE_NAME]."
- "Check if [USER_NAME] has unresolved security issues."
- "List devices flagged as non-compliant and the steps to resolve compliance issues."

End User

- "Why is my device flagged as non-compliant?"
- "What steps can I take to secure my account better?"
- "Check if any suspicious login attempts have been detected for my account."

These example prompts can help you quickly retrieve security insights and automate some of your daily security tasks.

Chapter 8: Delving into Microsoft Security Copilot

Before exploring how to create your own Copilot, it is essential to examine one of the most transformative tools in Microsoft's AI portfolio: Microsoft Security Copilot. Seamlessly integrated into products like Intune, Entra, Purview, Sentinel, Defender and many more, Security Copilot represents a groundbreaking evolution in how security professionals manage and respond to threats.

What is Microsoft Security Copilot?

Microsoft Security Copilot is a generative AI-powered assistant that revolutionizes cybersecurity workflows by leveraging advanced large language models (LLMs) and Microsoft's unique security expertise. Designed to process and analyze security signals at machine speed, it provides actionable insights, enabling professionals to detect, investigate, and respond to threats more effectively.

Microsoft's Definition:

> *"Copilot for Security is an AI cybersecurity product that enables security professionals to respond to cyberthreats quickly, process signals at machine speed, and assess risk exposure in minutes."*

This integration of Transformer models—the same technology behind GPT—with Microsoft's security infrastructure allows Copilot to:

- Summarize vast data signals into concise, actionable insights.
- Identify emerging threats and vulnerabilities.
- Provide real-time, natural language interaction for comprehensive threat analysis.
- Automate routine and complex security tasks, empowering security teams to focus on strategic priorities.

Products Integrated with Microsoft Security Copilot

Microsoft Security Copilot seamlessly integrates with a suite of tools and platforms within the Microsoft Security ecosystem. These integrations enhance its capabilities, ensuring comprehensive protection across your organization's digital environment. Here are the key integrated products:

- **Unified Security Operations Platform**
  Copilot enhances collaboration and incident response by unifying signals across all integrated tools.

- **Microsoft Sentinel**
  A cloud-native SIEM and SOAR solution, Sentinel benefits from AI-driven insights and incident management with Security Copilot.

- **Microsoft Defender XDR**
  Extend detection and response (XDR) capabilities with AI-powered analysis to protect endpoints, identities, apps, and more.

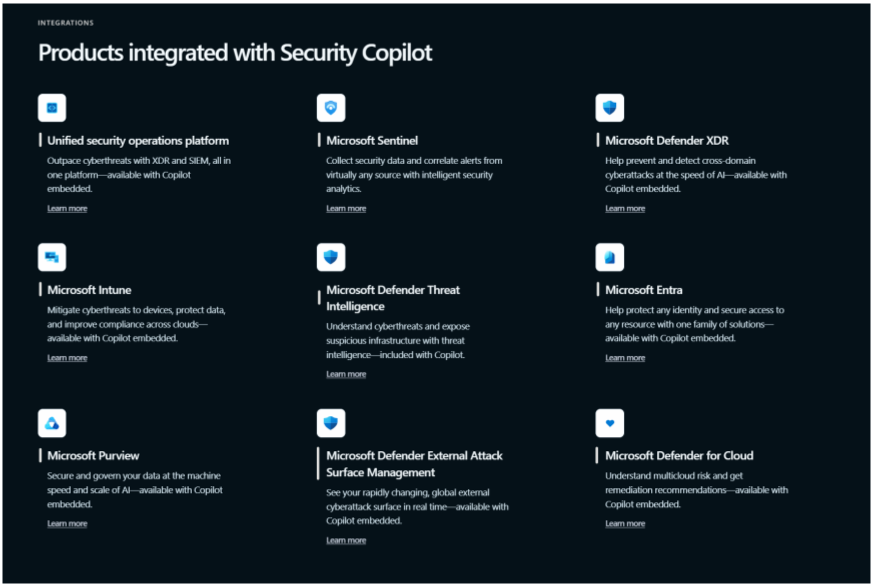
- **Microsoft Intune**
  Insights into devices, applications, and compliance while leveraging Security Copilot’s insights for risk assessment and incident response as well to get live information’s from devices with the help of Advanced Endpoint Analytics.

- **Microsoft Defender Threat Intelligence**
  Strengthen threat detection with AI-powered analysis of adversary behaviors and vulnerabilities.

- **Microsoft Purview**
  Simplify data protection, governance, and compliance tasks with actionable insights provided by Security Copilot.

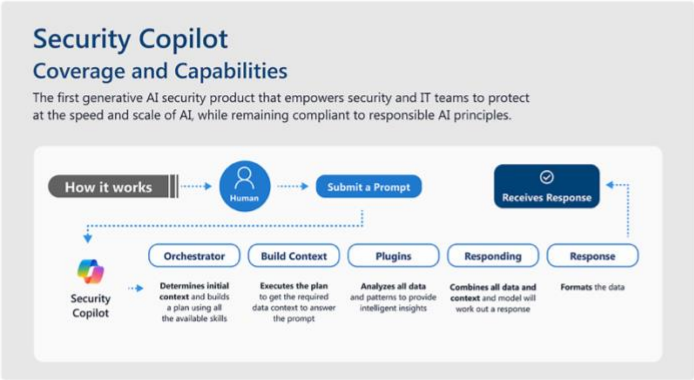
- **Microsoft Defender External Attack Surface Management**
  Identify and manage external vulnerabilities with real-time insights to minimize risk exposure.

- **Microsoft Defender for Cloud**
  Enhance cloud security posture management by uncovering threats and misconfigurations with AI assistance.



How Does Microsoft Security Copilot Work?

At its core, Security Copilot functions as an orchestrator, combining human inputs with AI-driven intelligence to execute workflows efficiently.



Here's a simplified overview how it works:

- **Input and Interaction:**
  Users interact with Security Copilot through natural language prompts, asking questions or issuing commands. This intuitive interface eliminates the need for specialized syntax like KQL, making security operations accessible to a broader audience.

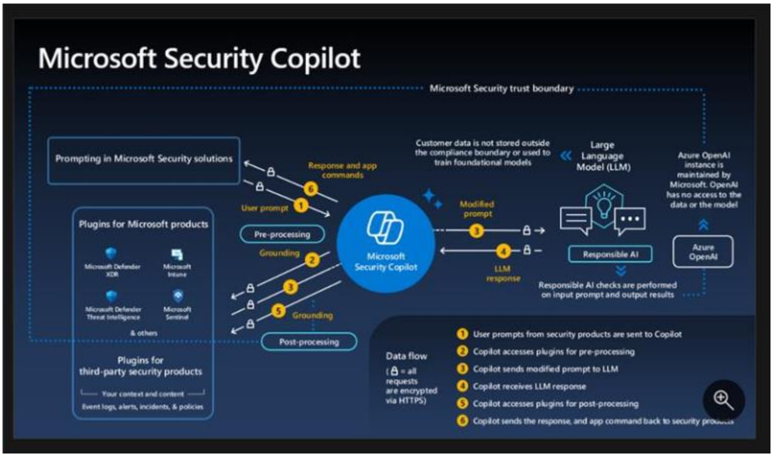
- **Grounding and Contextualization:**
  Security Copilot refines user inputs by leveraging plugins, knowledge bases, and authoritative sources, ensuring the query or command is contextually understood. This grounding process aligns responses with relevant and actionable data, enhancing precision and relevance.

- **Data Integration:**
  Security Copilot processes inputs by integrating data from Microsoft's comprehensive suite of security tools—such as Defender, Sentinel, and Entra—alongside third-party solutions. It continuously pulls insights from Microsoft's threat intelligence, powered by 78 trillion daily signals and enriched with human expertise.

- **AI-Driven Analysis:**
  By utilizing advanced AI models, Security Copilot generates actionable insights. These insights are tailored to the user's specific needs, providing detailed reports, step-by-step guidance, and enriched visualizations that make complex security incidents easier to understand and address.

- **Customized Responses:**
  Responses are tailored to the user's needs, whether in the form of text, code, or visuals. Copilot translates security incidents into actionable next steps, helping teams understand the full picture, evaluate impacts, and implement effective measures.

This multi-step process makes it a highly adaptable and efficient tool for addressing security challenges at scale.



Source: https://learn.microsoft.com/en-us/copilot/security/microsoft-security-copilot

But let's have an more deeper look into the architecture describe this more in an step by step approach:

1. User Prompt
   - **Process**: A user interacts with Microsoft Security Copilot by sending a prompt or query from a Microsoft Security solution (e.g., Defender, Sentinel, Intune).

- **Purpose**: This prompt represents the user's specific security question or action request.

2. Pre-Processing and Accessing Plugins
   - **Process**: The Copilot accesses integrated plugins for Microsoft security products (like Microsoft Defender, Sentinel, Intune) and third-party security tools.
   - **Purpose**: These plugins help preprocess the user prompt to add relevant context, such as event logs, alerts, incidents, or policies. This ensures the input is accurate and actionable.

3. Grounding and Modified Prompt
   - **Process**: Grounding involves enhancing the user's input with contextual data and security insights to create a refined or "modified prompt."
   - **Purpose**: To make the query precise, contextualized, and ready for processing by the large language model (LLM).

4. Large Language Model (LLM) Processing
   - **Process**: The modified prompt is sent securely to the Large Language Model (LLM), part of the Azure OpenAI service.
   - **Responsible AI Checks**: The LLM processes the input while adhering to Responsible AI guidelines to ensure ethical and compliant output.
   - **Security**: Customer data remains within Microsoft's compliance boundary, and OpenAI does not have access to the data or model.

5. LLM Response and Post-Processing

- **Process**: The LLM generates a response based on the refined query and passes it back to Microsoft Security Copilot.
- **Post-Processing**: The Copilot accesses plugins again for post-processing, such as verifying the response or enriching it with actionable insights and commands.
- **Purpose**: To ensure the response is secure, actionable, and aligned with the user's context and intent.

6. Response and App Commands
   - **Process**: The final response or application command is securely sent back to the user in the Microsoft Security solution.
   - **Purpose**: Enables the user to execute actions, gain insights, or resolve security issues efficiently.

Key Features and Use Cases

Security Copilot transforms security workflows across various roles, from analysts to CISOs:

For Security Administrators / SOC Analysts

- **Streamline Threat Response:** Receive actionable summaries of complex alerts and incidents.
- **Simplify Analysis:** Decode malicious scripts in plain language for quick understanding.
- **Guided Playbooks:** Follow step-by-step instructions for incident response and threat mitigation.
- **Proactive Defense:** Identify vulnerabilities and recommend remediations.

For Developers

- **Secure Code Insights:** Detect vulnerabilities in code repositories and recommend fixes.
- **Compliance Automation:** Ensure applications meet security compliance with minimal manual effort.
- **Threat Modeling Assistance:** Provide contextual suggestions to enhance application security design.

For IT Admins (Generalists and Specialists)

- **Policy Conflict Resolution:** Identify and resolve policy misconfigurations effortlessly.
- **Real-Time Insights:** Draft KQL queries and analyze device and user behavior in real time.
- **Cloud Optimization:** Enhance cloud PC performance, update management, and resource utilization.
- **Configuration Management:** Ensure secure and optimized configurations across on-premises and cloud environments.

For Helpdesk Teams

- **Accelerate Ticket Resolution:** Simplify troubleshooting steps with AI-guided resolutions.
- **User Guidance:** Provide plain-language solutions to technical issues for non-expert users.
- **Automated Root Cause Analysis:** Identify recurring issues and recommend fixes to minimize disruptions.

For End Users

- **Security Awareness:** Alert users to potential threats and provide clear steps to mitigate risks.
- **Access Troubleshooting:** Help resolve critical access failures while ensuring compliance with organizational policies.
- **Data Privacy Guidance:** Educate users on protecting sensitive information with contextual tips.

For CISOs

- **Executive Summaries:** Generate high-level overviews of the organization’s security posture.
- **Tailored Security Planning:** Build and customize strategic security roadmaps aligned with organizational goals.
- **Incident Prioritization:** Highlight critical incidents for efficient resource allocation and response.
- **Compliance Insights:** Monitor adherence to industry standards and recommend improvement strategies.

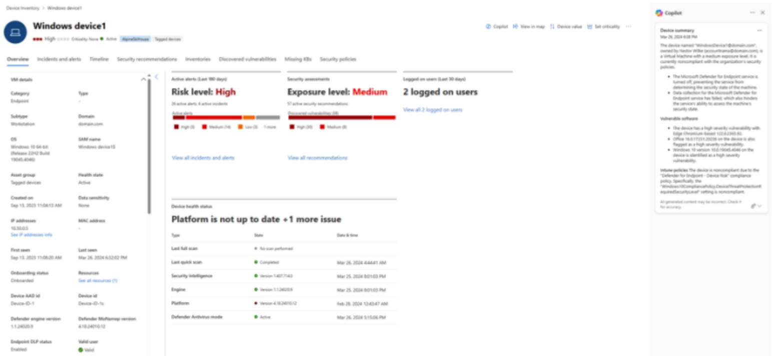
Transforming Security Dashboards

Traditional dashboards often overwhelm IT and security professionals with static data requiring constant monitoring and manual interpretation. **Microsoft Security Copilot** revolutionizes this process with a dynamic, conversational interface that caters to diverse roles, enabling:

- **Simplified Interaction:** Ask natural-language questions instead of navigating complex portals, whether you're a Security Admin, SOC Analyst, Developer, IT Admin, Helpdesk agent, or even an End User.
- **Real-Time Insights:** Access critical data or summaries on demand, tailored to your role. For example, developers can check code integrity, IT Admins can monitor device configurations, and Helpdesk agents can quickly access device health and incident details.
- **Data Where You Need It:** Break down silos by viewing comprehensive data—including security, compliance, and Intune information—directly in relevant tools like the Defender Portal. This ensures everyone, from specialists to generalists, gets the insights they need in one place.
- **Enhanced Efficiency:** Automate routine tasks such as log analysis, anomaly detection, and incident prioritization, freeing up time for proactive threat management and strategic work.

This unified approach transforms operations for all personas, ensuring data is not only dynamic and actionable but also seamlessly integrated into workflows. For example, Security Copilot can summarize device information, including security posture, anomalous behaviors, vulnerable software, and Microsoft Intune insights, enabling quicker and more informed decisions. By shifting from reactive monitoring to proactive management, Security Copilot reduces dependency on static dashboards and enhances collaboration across roles.

This Screenshot showcasing device summaries in the Defender Portal, integrating Microsoft Intune data.



Source: https://learn.microsoft.com/en-us/defender-xdr/security-copilot-in-microsoft-365-defender

Security Copilot Advantages

Microsoft Security Copilot sets itself apart with several unique capabilities. One of its core strengths is its seamless integration across Microsoft's security ecosystem, enabling tools like Defender, Sentinel, and Intune to work together in harmony. This integration creates a unified security platform that streamlines operations and reduces complexity.

Additionally, Security Copilot's extensibility allows it to incorporate third-party solutions such as ServiceNow and Jamf, further enhancing its versatility and functionality. The use of generative AI—powered by OpenAI's advanced models—is another defining feature. This capability allows Security Copilot to analyze complex scenarios at unprecedented speed and scale, delivering insights that are both actionable and timely. Complementing this is its access to evergreen threat intelligence, which ensures that security teams are always equipped with the most current global threat data.

The inclusion of cyber skills and promptbooks redefines the way security tasks are managed. These pre-designed workflows and customizable configurations empower security teams to handle intricate processes with efficiency and precision. By providing tailored solutions, promptbooks enable professionals to streamline repetitive tasks, reduce errors, and optimize their responses to threats.
Furthermore, this adaptability makes Security Copilot a critical tool across diverse industries, offering scalable solutions that align with the unique needs of each organization. The ability to incorporate organizational knowledge and align with specific security frameworks ensures that teams can address challenges proactively, ultimately enhancing their operational capacity and strategic oversight.

Consideration of Role-Based Access Control (RBAC) in Security Copilot

Implementing **Role-Based Access Control (RBAC)** within **Security Copilot** ensures secure, efficient, and tailored use of its capabilities across various roles within an organization. RBAC is a critical element in managing access to sensitive information and tools while aligning with the principle of least privilege.

What is RBAC and Why is it Important?

RBAC is a security model that assigns permissions to users based on their roles within an organization. This approach:

- Minimizes the risk of unauthorized access to sensitive information.
- Simplifies management by standardizing permissions for specific job functions.
- Enhances compliance with regulations and internal policies.
- Supports the principle of least privilege, limiting access to only what is necessary for a role.

Security Copilot integrates RBAC to provide role-specific features, ensuring users access only the capabilities relevant to their responsibilities.

RBAC is essential for Security Copilot, ensuring users access appropriate tools and data without sacrificing security.

Prerequisites and Pricing

Security Copilot operates on a billing model based on Security Compute Units (SCUs). Each SCU costs $4 per hour, amounting to approximately $2,920 per month for continuous usage. While this cost might seem significant for smaller organizations, the efficiency, enhanced security, and operational value it brings can justify the expense for enterprises with complex security needs.

Planning is a critical step when implementing Security Copilot. Once deployed, the size of an SCU cannot be changed, requiring organizations to carefully assess their compute requirements upfront to avoid unnecessary costs.

Beyond cost considerations, Security Copilot delivers a compelling return on investment by reducing the time spent on repetitive tasks, enhancing response accuracy, and empowering teams with actionable insights. By automating complex workflows, organizations can free up resources for strategic initiatives, improve incident resolution times, and strengthen overall security posture. For larger enterprises dealing with multifaceted security landscapes, these advantages often outweigh the initial expenditure, making Security Copilot an invaluable part of modern cybersecurity infrastructure.

Organizations are also encouraged to measure their metrics, such as mean time to respond (MTTR) and average incident resolution time, to evaluate the tangible impact of adopting Security Copilot. These measurements help demonstrate its effectiveness and build the case for scaling its usage further.

How to Deploy Microsoft Security Copilot

There are two primary methods to deploy Microsoft Security Copilot: through the Azure Portal and via code using an ARM (Azure Resource Manager) template and PowerShell script. Below, I will walk you through both deployment methods.

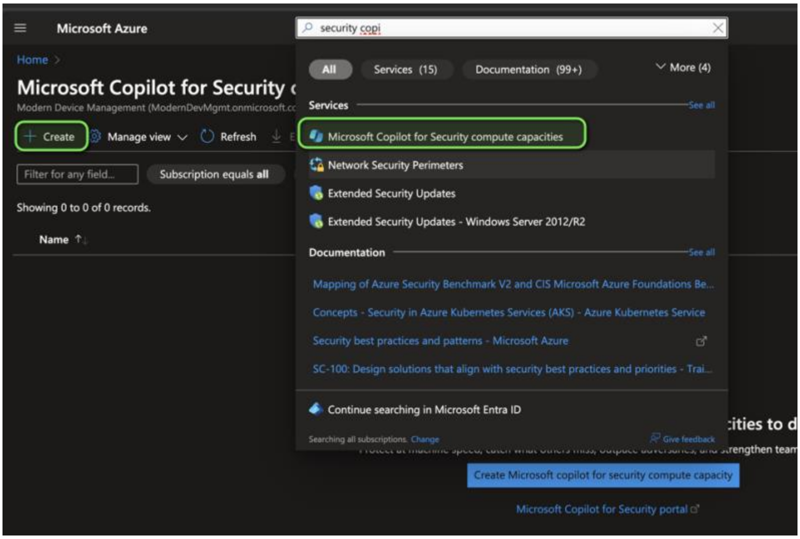
Deploying Security Copilot via Azure Portal

This is the easiest and most user-friendly way to deploy Security Copilot. Here’s a step-by-step guide:

1. **Log in to Azure Portal**:

- Navigate to the [Azure Portal](#).

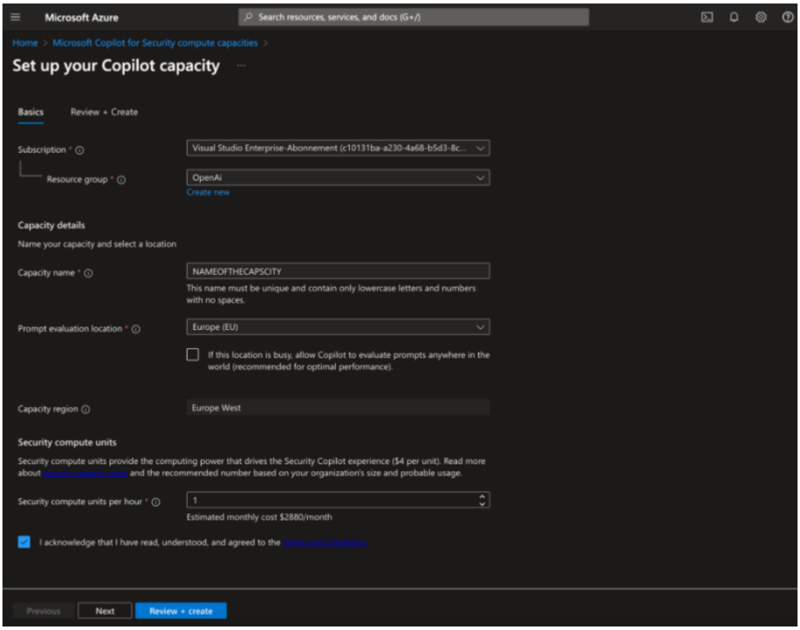
2. **Search for Security Copilot:**
   - In the search bar at the top of the Azure Portal, type "Security Copilot" and select the result. Alternatively, you can browse through the available services.



3. **Create a New Deployment**:
   - Click on the **+ Create** button to start a new deployment of Security Copilot.

4. **Configure Basic Settings**:
   - **Resource Group**: Select an existing resource group or create a new one.
   - **Name**: Enter a unique name for your Security Copilot instance.

- **Prompt Evaluation Location**: Choose the region where your prompts will be evaluated.
- **Compute Units**: Specify the number of Security Compute Units (SCUs) you want to allocate per hour. This determines the computing power dedicated to running your Copilot.



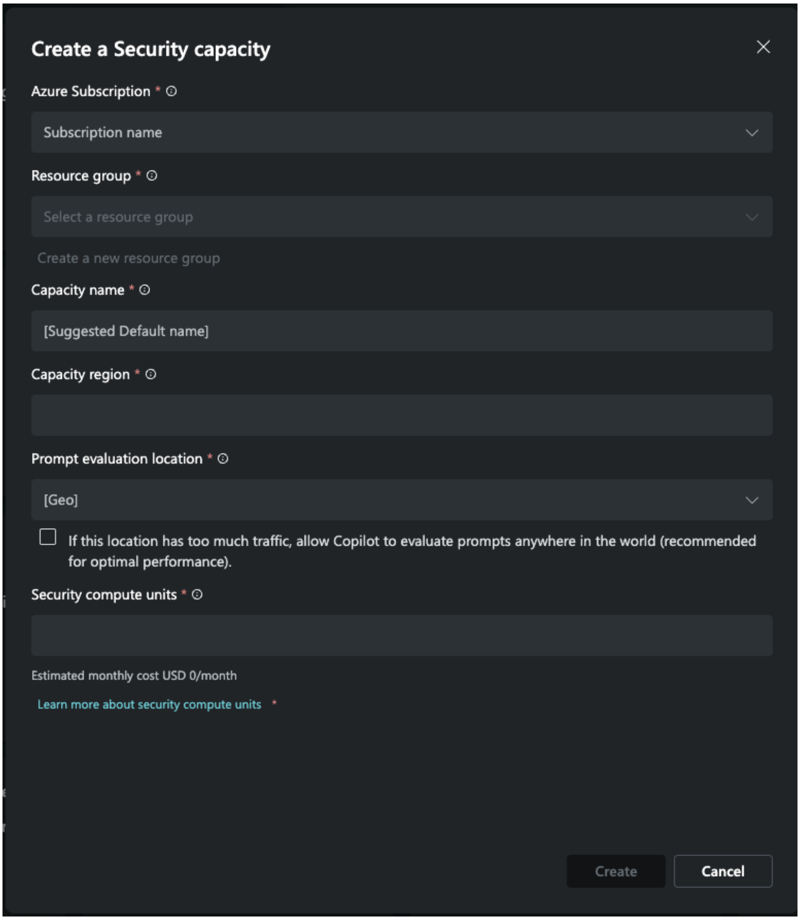
5. **Review and Create**:
   - After filling in the necessary details, click **Review + Create** to validate the configuration.
   - Once the validation passes, click **Create** to deploy Security Copilot. The deployment process will take a few minutes.

6. **Monitor Deployment**:
    - You can monitor the deployment process in the "Notifications" area of the Azure Portal. Once completed, you will see the Security Copilot instance listed in your resources.

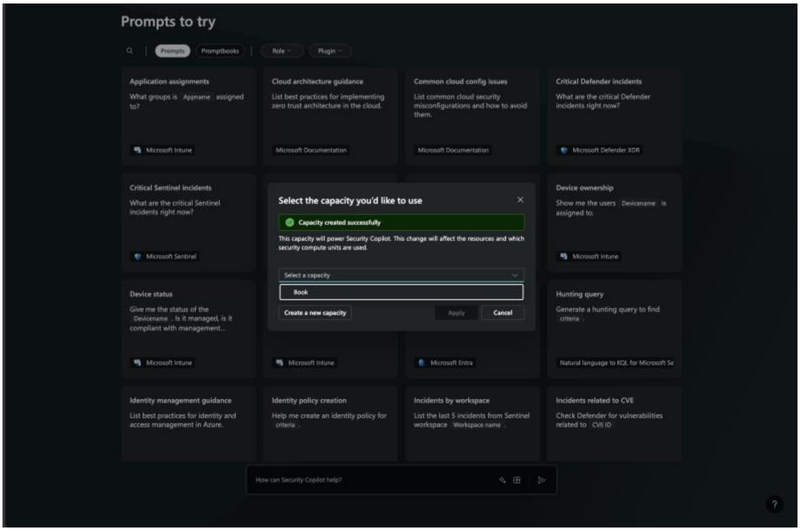
Deploying via Microsoft Security Copilot portal

An additional method to deploy Microsoft Security Copilot is through the dedicated portal. To begin, navigate to https://securitycopilot.microsoft.com/tour/admin and log in with your credentials.

Upon logging in, if no Security Compute Unit (SCU) has been deployed in your environment, a prompt will guide you through the deployment process. This step-by-step interface simplifies the onboarding of Security Copilot, ensuring that even those unfamiliar with the platform can deploy it very easy.



After the deployment is complete, a prompt will appear allowing you to select your desired security capacity. After this you can begin prompting the system.



Deploying Security Copilot via Code

For more advanced users or automated deployments, you can deploy Security Copilot using an ARM template and PowerShell script. This method provides more flexibility and can be integrated into CI/CD pipelines for infrastructure as code (IaC) scenarios.

Step 1: Prepare the ARM Template

An ARM template defines the infrastructure and configuration for your deployment. Below is a simplified version of an ARM template for Security Copilot:

```
{
    "$schema":
"https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "resources": [
```

```
        {
            "type":
"Microsoft.SecurityCopilot/copilotInstances",
            "apiVersion": "2023-01-01-preview",
            "name": "[parameters('copilotName')]",
            "location": "[parameters('location')]",
            "properties": {
                "computeUnitsPerHour":
"[parameters('computeUnits')]"
            }
        }
    ],
    "parameters": {
        "copilotName": {
            "type": "string",
            "metadata": {
                "description": "Name of the Security
Copilot instance"
            }
        },
        "location": {
            "type": "string",
            "metadata": {
                "description": "Azure region for the
deployment"
            }
        },
        "computeUnits": {
            "type": "int",
            "defaultValue": 1,
            "metadata": {
                "description": "Number of compute units
per hour"
            }
        }
    }
}
```

Step 2: Write the PowerShell Script

You'll need a PowerShell script to deploy the ARM template to Azure. Here's an example script:

```
# Login to Azure

Connect-AzAccount

# Define parameters

$resourceGroupName = "YourResourceGroupName" $location =
"YourLocation"
$copilotName = "YourCopilotName"
$templateFile = "path-to-your-arm-template.json"
$computeUnits = 1

# Create resource group if it doesn't exist

if (-not (Get-AzResourceGroup -Name $resourceGroupName -
ErrorAction SilentlyContinue)) {
New-AzResourceGroup -Name $resourceGroupName -Location
$location }

# Deploy ARM template
New-AzResourceGroupDeployment -ResourceGroupName
$resourceGroupName -TemplateFile $templateFile -
copilotName $copilotName -location $location -computeUnits
$computeUnits
```

Step 3: Execute the Deployment

Prepare Your Environment:

- Ensure that you have the **Az PowerShell module** installed and that you are logged into your Azure account using Connect-AzAccount.

Run the Script:

- Execute the PowerShell script. This script will create the resource group (if it doesn't already exist) and deploy the ARM template, provisioning the Security Copilot instance with the specified configuration.

Monitor the Deployment:

- You can monitor the deployment in the Azure Portal or use the Get-AzResourceGroupDeployment command to check the status of the deployment via PowerShell.

Setting Up Microsoft Security Copilot

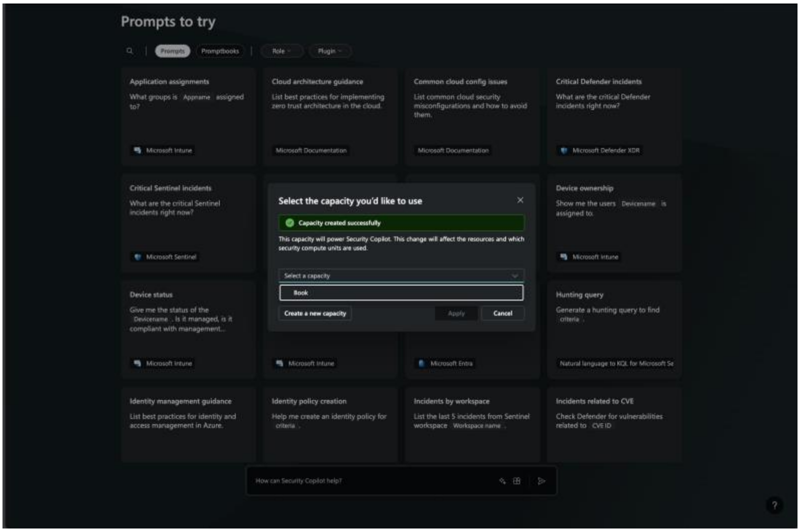
Setting up Microsoft Security Copilot is a straightforward process that involves enrolling your organization, selecting your desired capacity, and finalizing the configuration. Below is a step-by-step guide to help you get started.

Step 1: Open the Security Copilot Enrollment Page

- **Navigate to the Security Copilot Enrollment Page**: Open your web browser and go to https://securitycopilot.microsoft.com/tour/admin.
- **Start Enrollment**: Click the **Start Enrollment** button to begin the setup process. You will be guided through the necessary steps to configure Security Copilot for your organization.

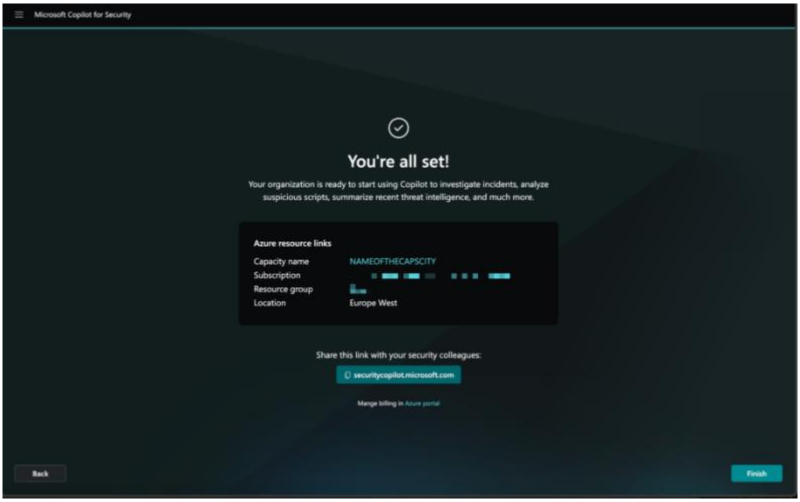
Step 2: Select Capacity

- **Choose Your Capacity**: During the enrollment process, you will be prompted to select your desired capacity. This refers to the number of Security Compute Units (SCUs) that you want to allocate for Security Copilot. Choose the capacity that best fits your organization's needs and click **Continue** to proceed.



Step 5: Complete the Setup

- **Finalizing the Setup**: Once you've configured the capacity and diagnostic settings, the setup process is nearly complete. Review the information on the final page, and when you're ready, click **Finish**.



Confirmation: After clicking **Finish**, you will receive confirmation that your Security Copilot has been successfully set up. You can now begin using it to enhance your cybersecurity operations.

How to access and use Microsoft Security Copilot

There are two primary ways to access and use Microsoft Security Copilot: through the central portal and via the embedded experience in Intune. Both provide powerful AI-driven assistance to streamline security tasks, but they differ in how they are integrated into your workflows.

Opening the Security Copilot Portal

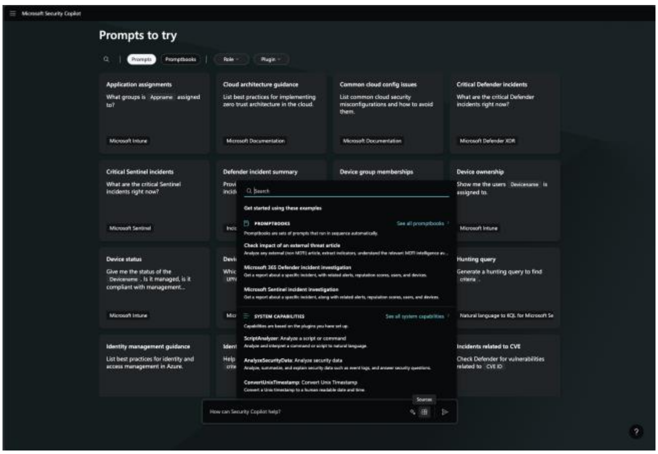
The central experience for Security Copilot provides an open prompt interface where you can interact directly with the AI. Here's how you can access and use this feature:

1. **Access the Central Portal**:
   - Open your web browser and navigate to https://securitycopilot.microsoft.com/sessions/new.

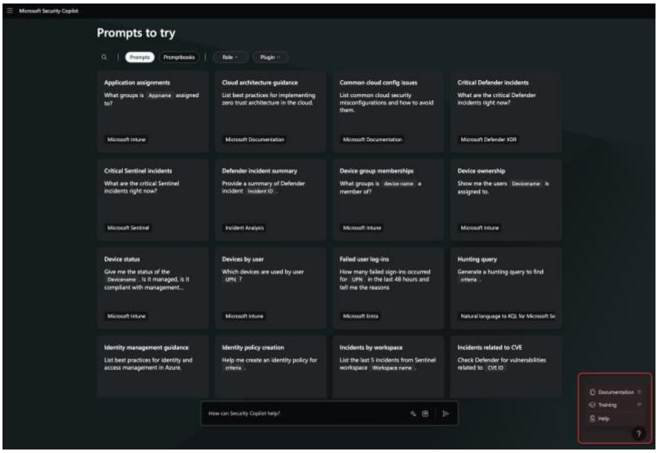
2. **Explore the Open Prompt Experience**:
   - Once on the page, you'll see an open prompt interface where you can ask Security Copilot various security-related questions. This experience allows you to utilize the full range of

Copilot's capabilities by simply typing your queries.

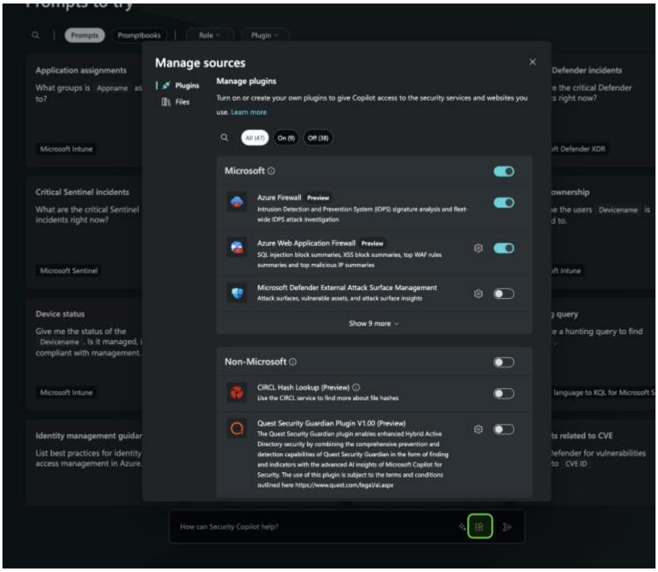
- You can explore **prompt hints** that suggest questions to ask or tasks you can perform with Security Copilot. This feature is useful for new users who want to learn the kinds of questions Copilot can handle.



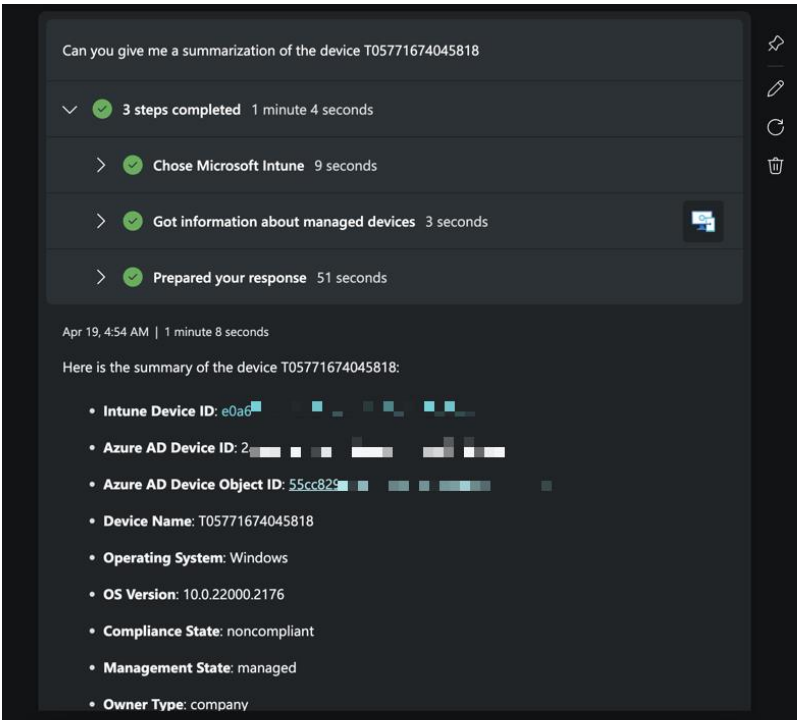
- You can also find at the start page a wealth of information and learning resources about using Security Copilot effectively.



3. **Enable or Disable Plugins**:
   - On the interface, you’ll find a button that allows you to turn plugins on or off. These plugins enhance Copilot's capabilities by connecting it to additional tools and data sources, enabling more comprehensive responses to your queries.



4. **Ask Questions and Get Summaries**:
   - You can ask Security Copilot specific questions, such as "Can you give me a summarization of the device **[DEVICE_NAME]**?" Security Copilot will provide a detailed summary of the device, including any relevant security insights.
   - You can continue the conversation with follow-up questions to refine your inquiry or gather more details.



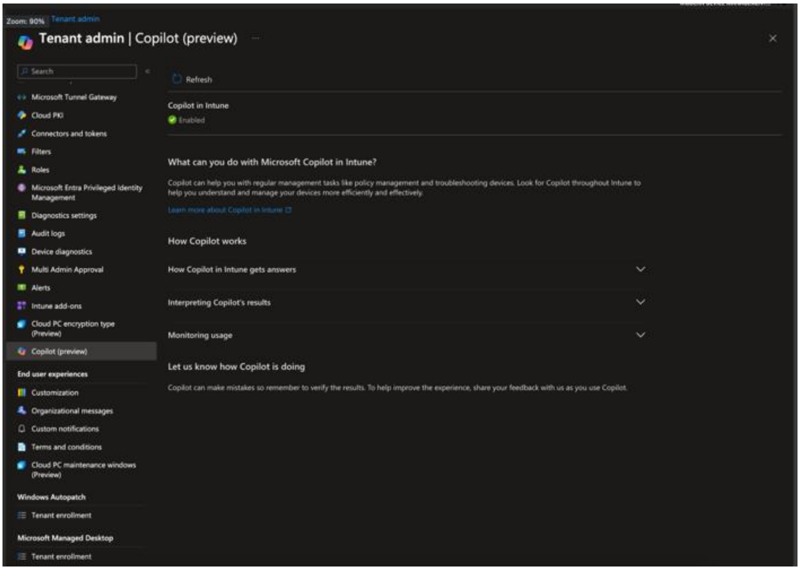
Using the Embedded Experience in Intune

For a more guided experience, Security Copilot is embedded directly into the **Intune** interface. Here's how to use this integration:

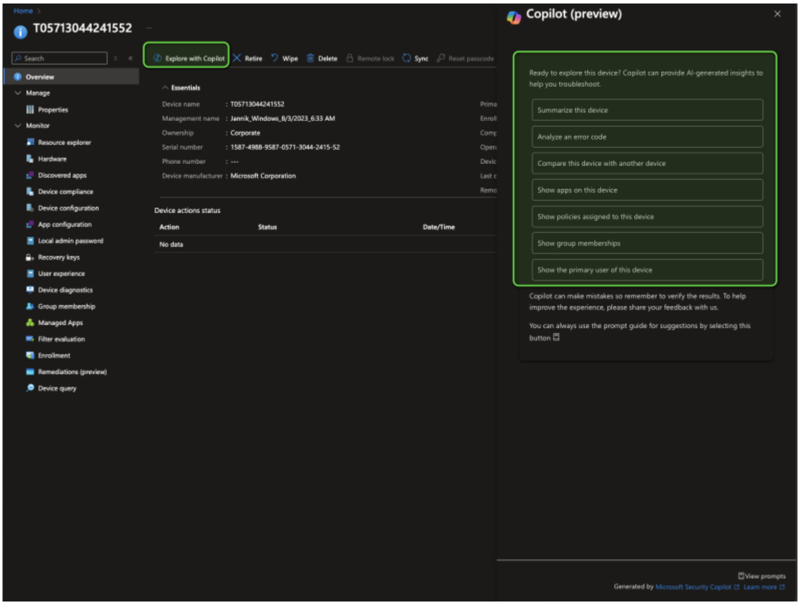
1. **Access Intune**:
   - Open your web browser and go to https://intune.microsoft.com.

2. **Activate Security Copilot in Intune**:
   - Ensure that Security Copilot is activated in your tenant. To do this, navigate to the **Tenant**

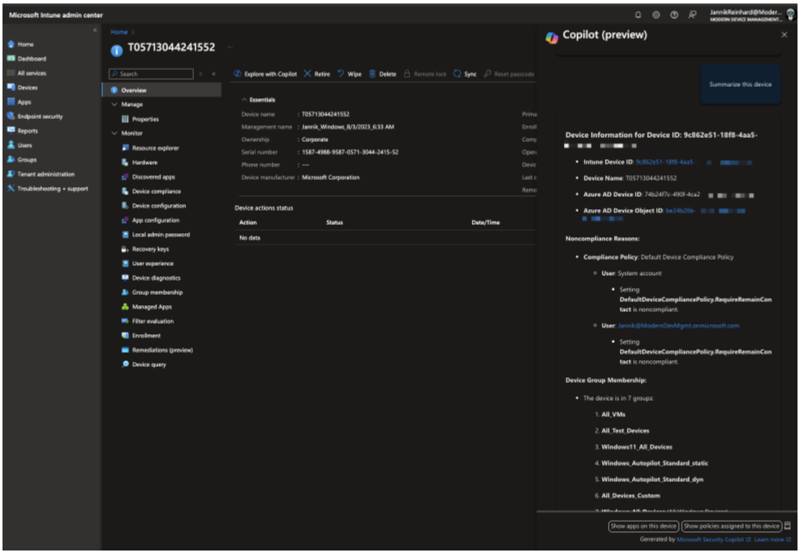
Administration section of Intune and verify that Security Copilot is enabled.



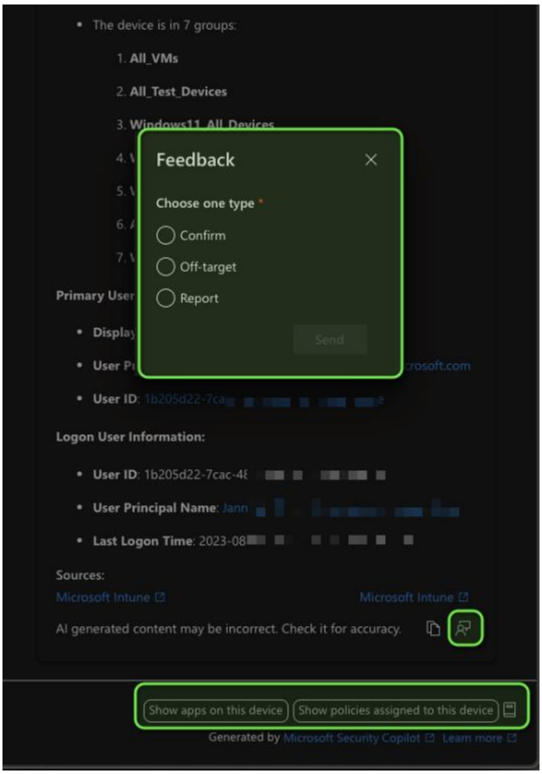
3. **Guided Prompts in Intune**:
   - The Intune integration of Security Copilot differs from the open prompt experience in the central portal. Here, Copilot provides more guided prompts that are context specific.
   - For example, when you open a device, policy, or other objects in Intune, you'll see a Copilot icon. If you open a configuration profile, you'll find a button labeled **"Explore with Copilot."** Clicking this button will get an overview of the available actions of the Microsoft Security Copilot in Intune.



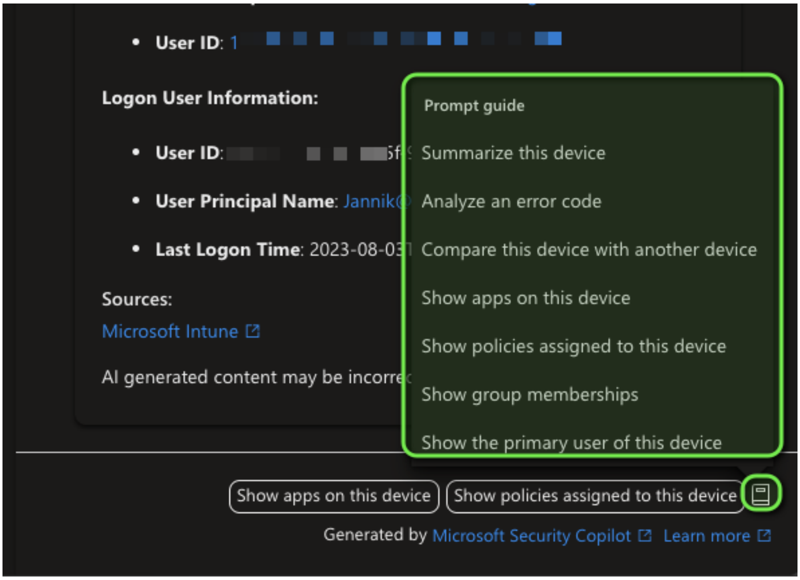
I selected the action Summarize this device and what I get are all the relevant information's around this device in an summarized view:



4. **Interact and Provide Feedback**:
   - After using Copilot to generate a summary or other responses, you'll see options to provide feedback. This helps improve the product by refining how Copilot responds to different prompts.



   - You can also enter follow-up prompts directly in the interface to continue refining the information you receive. Additionally, Copilot offers a **prompt book** with suggestions for further actions, making it easy to continue working without needing to think of the next question yourself.



If you are not already part of this LinkedIn group, I can also recommend joining:
https://www.linkedin.com/groups/14345161/

Chapter 9: Deep dive into other Copilots and services

We have already discussed the Microsoft Security Copilot, but we will also examine the M365 Copilot to understand how it functions behind the scenes.

The Architecture of M365 Copilot: An Overview

When using M365 Copilot, such as in Office applications like Word, Excel, or Outlook, the behind-the-scenes architecture is centered around the powerful integration of **Microsoft Graph**, **AI orchestration**, and **OpenAI Services**. To demonstrate how such a copilot is build we will have a deeper look into the architecture. This architecture is designed to provide seamless, context-aware assistance by tapping into a wealth of user-specific data and leveraging advanced AI models to generate responses.

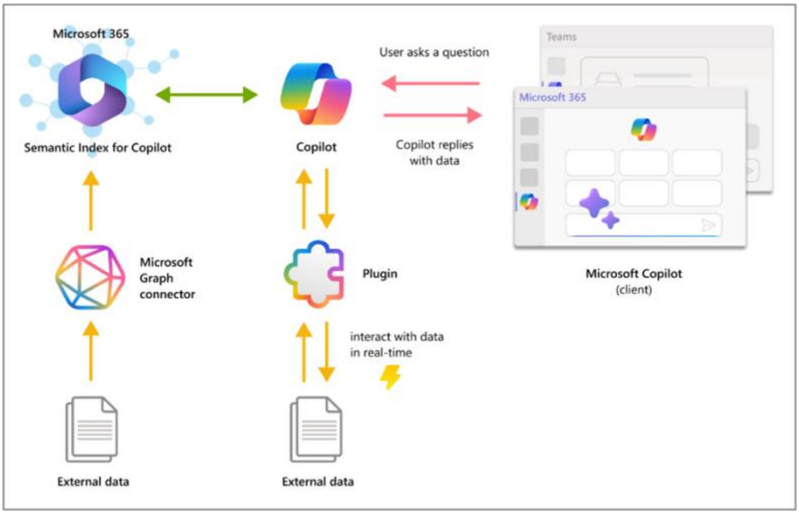
Here's a simplified breakdown of how the architecture works:

1. User Input and Intent Detection:

When you enter a prompt or request in an M365 product (e.g., asking Word to generate a summary or Excel to analyze data), your input is sent to the **M365 Chat Engine**. This engine is responsible for understanding your intent by analyzing the input text.

2. Query to Microsoft Graph:

Based on the detected intent, the M365 Chat Engine performs a **query to Microsoft Graph**. Microsoft Graph is the core data platform for Microsoft 365 services, acting as a gateway to the user's data across various services (e.g., emails, documents, calendar events, etc.). The query retrieves relevant data in the context of the user, grounding the AI's response in real, personalized information.



Source: https://learn.microsoft.com/en-us/compliance/anz/blueprint-copilot-servicecom

3. Prompt Modification:

After retrieving the necessary data from Graph, the system builds a modified prompt. This prompt is enriched with templates and the context fetched from Microsoft Graph, ensuring the response generated by the AI is relevant and grounded in the user's specific data.

4. Sending to OpenAI Service:

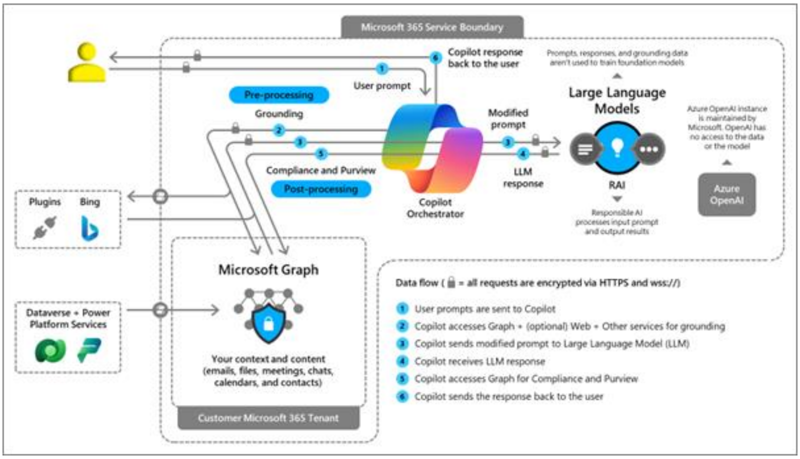
The enriched prompt is then sent to the **OpenAI Service**, where the AI model (such as GPT) generates a response based on the given context and the user’s data.

5. Post-Processing:

Once the OpenAI Service provides the response, there is a **post-processing** step. During this phase, additional content may be attached to the AI-generated response, such as raw data (e.g., a list, a file, or a document). This ensures that the final output is not only accurate but also enriched with any additional data that the user might need.

6. Response Delivery:

Finally, the complete bundle (AI-generated response plus any additional data) is sent back to the user within the M365 application they are using.



Source: https://learn.microsoft.com/en-us/compliance/anz/blueprint-copilot-servicecom

This entire process happens seamlessly in the background, allowing users to experience real-time assistance without needing to understand the complex mechanics at play.

Deep Dive: Copilot Stack

To better understand the M365 Copilot, let's break down its **stack** into different layers, each playing a critical role in delivering AI-powered assistance:

Azure Infrastructure and Foundational Model:

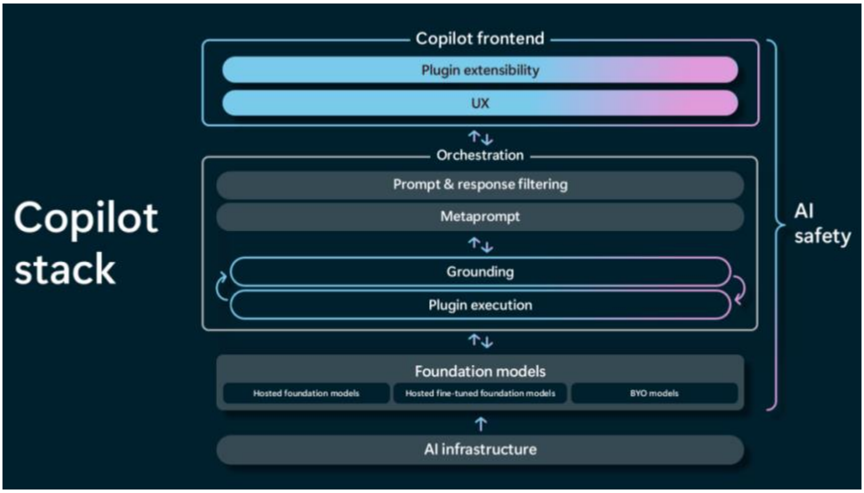
At the core, M365 Copilot relies on **Azure's cloud infrastructure** and the foundational AI models, such as OpenAI's GPT. This provides the raw computational power and intelligence needed to process large volumes of data and generate coherent, context-aware responses.

AI Orchestration Layer:

On top of the foundational model lies the **AI Orchestration Layer**. This layer acts as the brain of the Copilot, managing the flow of information between different services, deciding which data to retrieve, which actions to take, and how to prioritize tasks. It coordinates between Microsoft Graph, AI models, and other external services to deliver a seamless user experience.

Frontend and Extensions:

The **Frontend** is the user interface where users interact with the Copilot (e.g., in Word, Excel, or Outlook). The Copilot's functionality can also be extended with **Add-Ons, Skills, or Plugins**. These extensions allow third-party developers to build additional features on top of the Copilot, making it even more versatile. These add-ons have corresponding components within the AI Orchestration Engine, ensuring that their functionalities are smoothly integrated into the overall experience.



How can build my own grounding with internal knowledge?

After exploring Microsoft's Copilot solutions and understanding how they work, we will examine the possibilities Microsoft provides for building our own Copilots. But to do this we have to understand some essential things like grounding.

Grounding a Large Language Model (LLM) with internal knowledge involves integrating your specific data or content into the model's decision-making process, ensuring the AI generates more accurate, relevant, and context-specific responses. There are multiple ways to achieve this, depending on the complexity of your application and the scale of the data you need to integrate.

Simple Grounding: Using Direct Prompts

For basic queries, the easiest grounding method is to directly include the relevant content or knowledge in your prompt. This approach is useful for one-off queries where you can copy-paste text or details from a document into the prompt. However, this method becomes less practical as your use case scales or when building more complex applications.

Example:

- **Prompt**: "Based on the following document, summarize the key points: [insert text from the document]."
- **Response**: The model generates a summary based on the provided content.

While effective for single queries, this approach does not scale well for large datasets or continuous interactions, making it unsuitable for more sophisticated applications.

Advanced Concept: Using Embeddings and Vector Databases

For more advanced implementations, especially when dealing with large datasets or requiring more precise information retrieval, you can take a more sophisticated approach by creating **embeddings** of your content and storing them in a **vector database**.

1. **Create Embeddings**: An **embedding model** transforms your documents and internal knowledge into dense vectors (numerical representations). These embeddings capture the semantic meaning of the content, enabling similarity-based retrieval. During this step, you can also preprocess the data by translating, standardizing, or extracting key information.

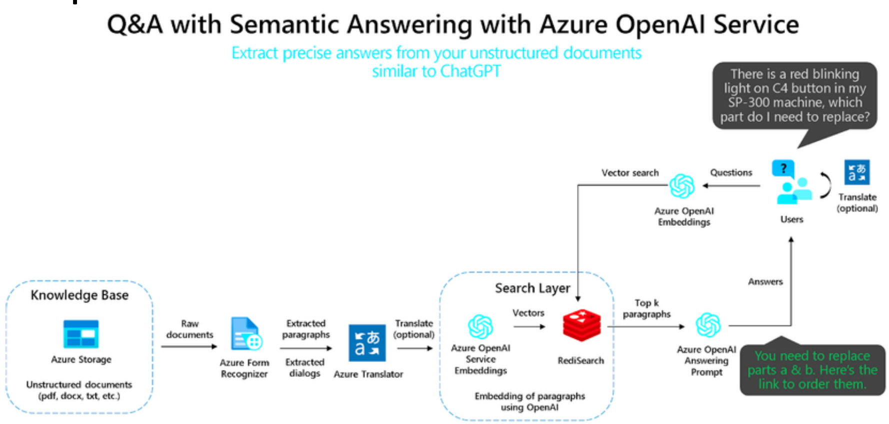
2. **Store in a Vector Database**: The generated embeddings are stored in a **vector database**. This database is optimized for searching and retrieving similar vectors, allowing you to efficiently find relevant information based on a user's query.

3. **Querying with Vectorized Search**: When a user submits a query, the query is also transformed into an embedding. This vectorized query is then matched against the stored embeddings in the vector database. The best-fitting content is

retrieved and used to ground the response generated by the LLM.

This approach is more complex but offers superior performance and accuracy, particularly in large-scale applications where you need to provide highly relevant and precise responses.

Example Architecture:



Source: https://techcommunity.microsoft.com/t5/microsoft-developer-community/vector-similarity-search-with-azure-cache-for-redis-enterprise/ba-p/3822059

This architecture typically involves using **Azure Functions**, **Azure App Service**, and **Azure AI Services (formerly Cognitive Services)**, providing a robust foundation for building enterprise-grade AI solutions.

Advanced Grounding: Azure AI Search and Blob Storage Integration

For more complex applications, a better approach involves storing your documents and content in a structured way, such as in **Azure Blob Storage**, and using **Azure AI Search** (formerly Cognitive Search) to index your documents. Here's how this setup works:

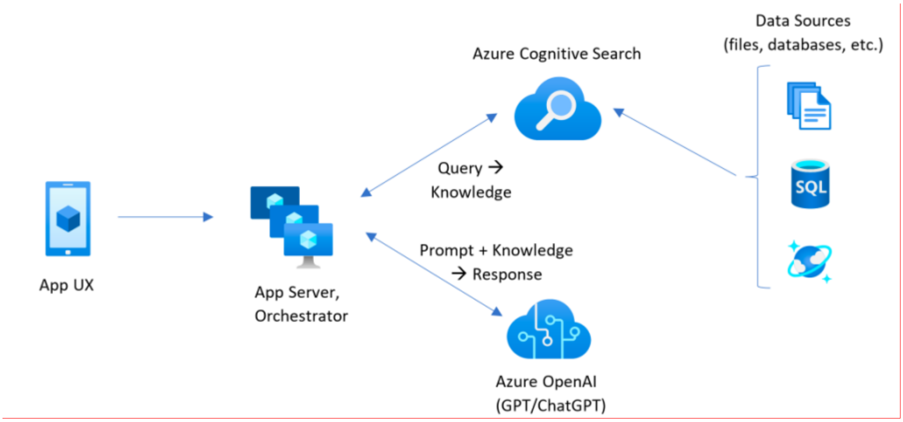
1. **Store Documents in Azure Blob Storage**: Upload your internal documents, knowledge bases, or other relevant content into Azure Blob Storage. This is a scalable storage solution that integrates well with Azure AI services.

2. **Index Content with Azure AI Search**: Use Azure AI Search to index your content stored in Blob Storage. Azure AI Search applies natural language processing (NLP) to create a searchable index of your documents, making it easy to retrieve relevant information based on user queries.

3. **Querying with Azure AI Search**: When a user asks a question in your application, the query is first sent to Azure AI Search, which attempts to find the most relevant content from the indexed documents. This content is then retrieved and passed along with the user’s query to the GPT service to generate a more accurate response.

This architecture enables dynamic grounding of the model with real-time, relevant information from your internal knowledge base. Microsoft provides a reference architecture for implementing this solution using Azure services.

Reference Architecture:



Source: https://learn.microsoft.com/de-de/azure/search/retrieval-augmented-generation-overview

Deploying and Understanding Azure AI Search

The Azure AI Search is one of the most powerful search engines available today, and in my experience, it's particularly relevant in the age of AI and large language models (LLMs). What was once considered a routine technology—search engines—has gained renewed interest thanks to AI advancements. In this chapter, I will guide you through deploying and configuring Azure AI Search and explain how it can transform your data into a highly searchable, AI-enhanced resource.

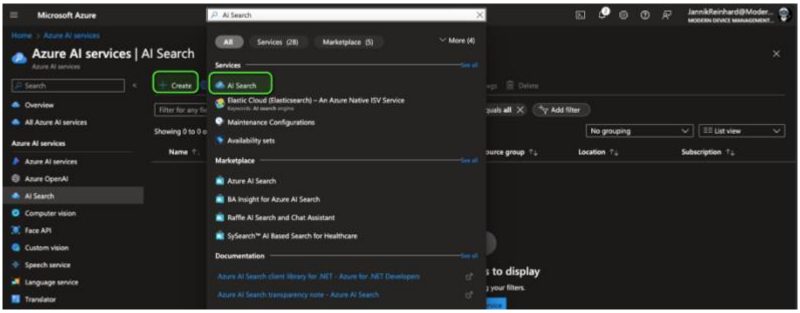
What is Azure AI Search?

Azure AI Search is a Platform-as-a-Service (PaaS) offering from Microsoft Azure. It allows organizations to create search indexes from various data sources and formats, such as SQL databases, PDF documents, PowerPoint files, and more. One of its standout features is vector search, which uses AI to understand and match search queries based on meaning rather than just keywords. Azure AI Search leverages models like Azure OpenAI's text-embedding-ada-002 or newer versions to enhance search capabilities, offering more contextually relevant results.

Deploying the Service

Deploying Azure AI Search is straightforward. Follow these steps:

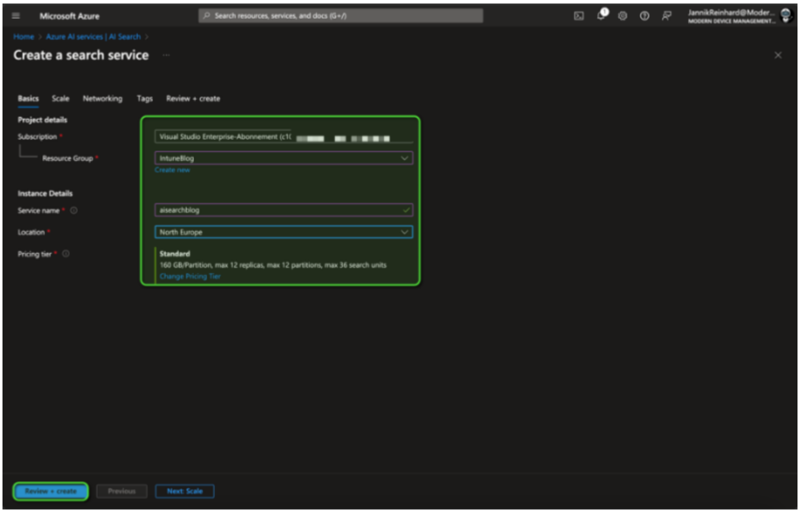
1. **Open the Azure Portal:** In the search bar, type "Azure AI Search" and select it from the results.



2. **Create the Service:** Click on the "+ Create" button. Then, select your subscription, resource group, and region, and give your search resource a name.

3. **Security Considerations:** For production environments, it's best to secure the resource by placing it within the same network as your application. For this tutorial, we'll skip this step, but it's recommended for production deployments.

4. **Review and Deploy:** After reviewing your settings, click "Review + Create" to deploy the resource.

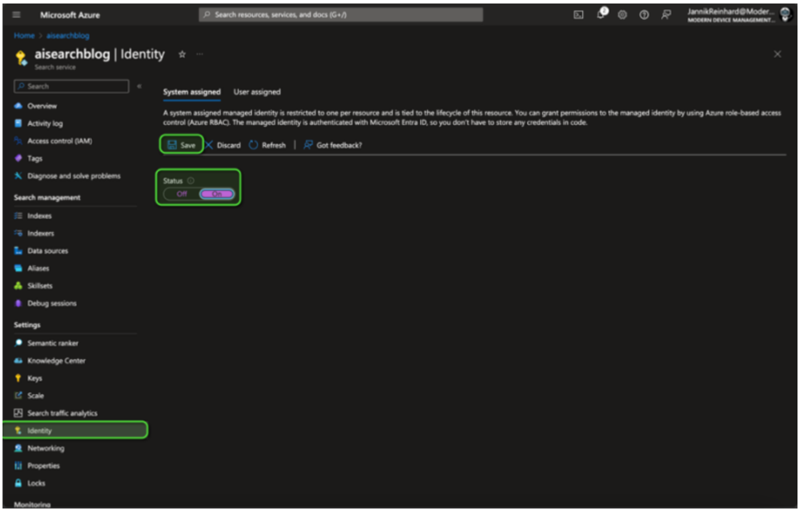


This deployment process is simple, but it’s only the first step in creating a powerful search engine. Next, you need to prepare your environment and create search indexes.

Preparing Your Environment

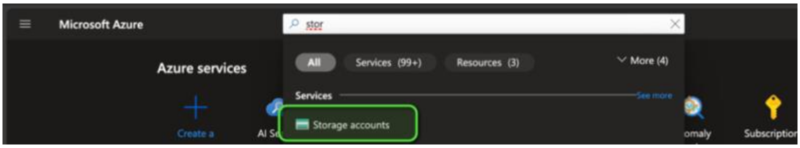
Before you can create a search index, you need to take a few preparatory steps:

1. **Enable System-Assigned Managed Identity:** This allows your search service to securely access other Azure resources. You can activate this in the Identity settings of your Azure AI Search instance.

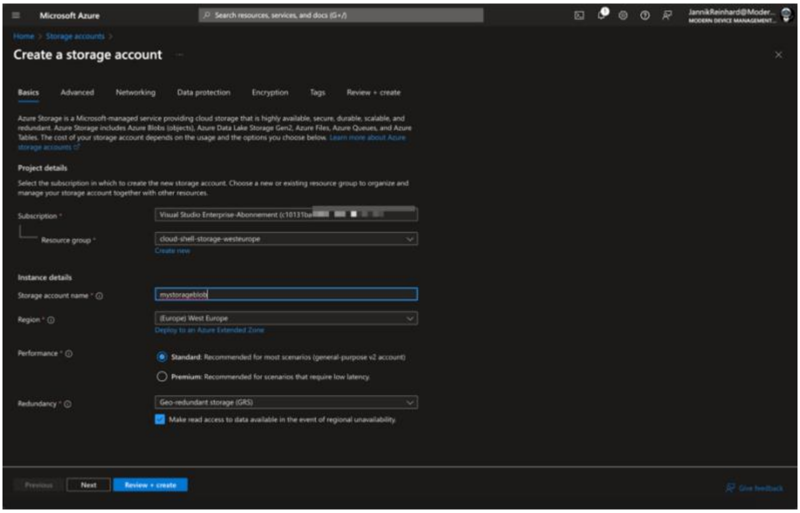


2. **Set Up Blob Storage:** Create a Blob storage account and container to store the data you want to index. Azure Blob Storage serves as a flexible repository where you can upload various types of documents for indexing.

- Log in to the [Azure portal]
- Search for **Storage Account** and select the **Storage accounts** service



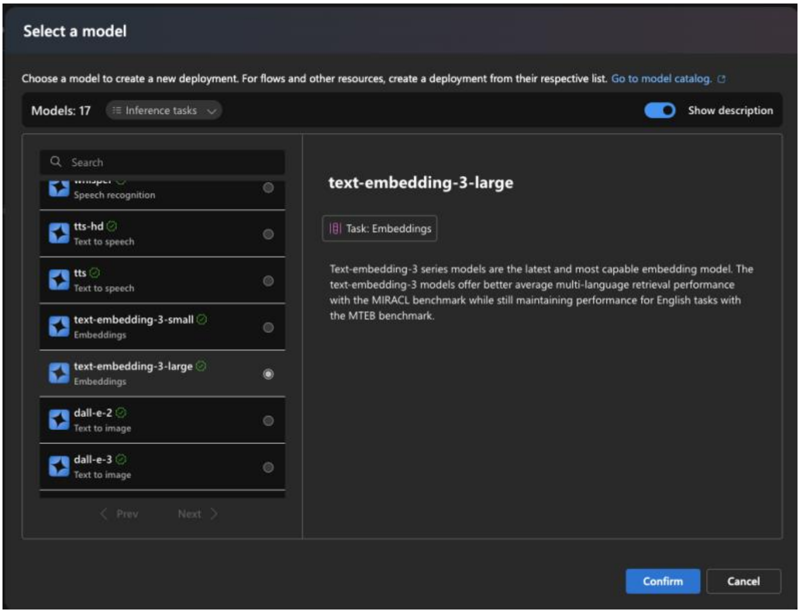
- Click on **+ Create** on the top
- Select the Subscription, Resource group and Region and enter a name for the account.
- Click **Review + create** and **create**



- You should also check out all the other settings. (e.g., for a production environment, it is recommended to put the storage in a network, enable encryption via customer-managed keys, and configure data protection).

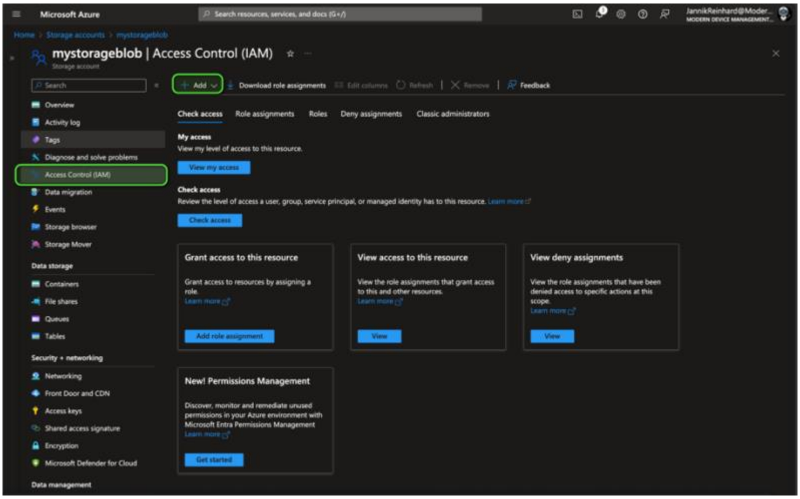
- To create a container, click on **+ Container** and enter a **name**

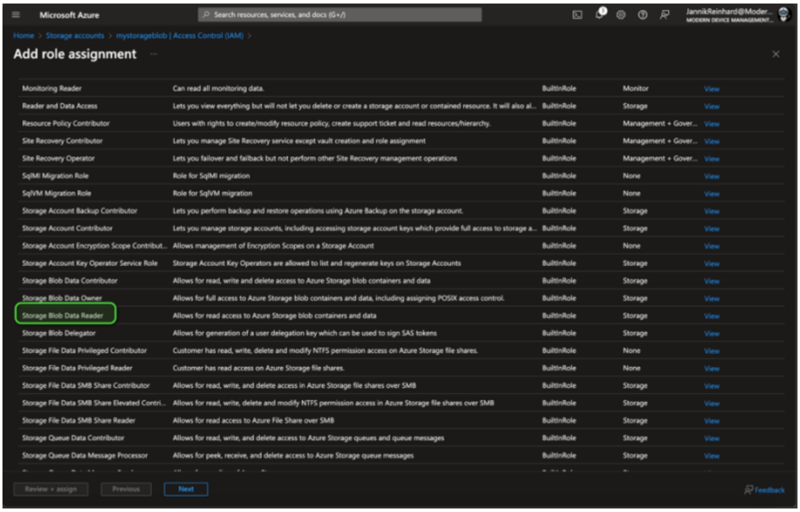
3. **Deploy an Embedding Model:** To use vector search, you must deploy an embedding model. Start by creating an Azure OpenAI instance. Once deployed, go to the "Deployments" section and select the text-embedding-ada-002 model or the latest available version.



Once these steps are complete, you'll be ready to create your search index and begin using the powerful features of Azure AI Search.

4. **Give Storage permissions to the AI Search:** To access the documents on the blob storage, give *Storage Blob Data Reader* or for the File share *Reader and Data Access.*





Creating an Index

Creating an index in Azure AI Search is a key step in making your data searchable. You can do this through the Azure Portal web interface:

1. **Use the Assistant:** On the Azure AI Search start page, you'll find an assistant that guides you through the process of creating an index. Click on "Import and Vectorize Data."

2. **Select Data Source:** Choose your Blob storage account and container where the files you want to index are stored.

3. **Authenticate:** For authentication, select "System-Assigned Identity" to allow your search service to access the Blob storage.

4. **Add Embedding Model:** Select the embedding model that you deployed earlier.

5. **Create Schedule:** You can skip the image enrichment step if you don't need to index images. In the Advanced settings step, select a schedule for refresh and update cycles.

6. **Review and Complete:** Follow the assistant's steps to finalize your index creation.

For more complex environments with multiple stages (e.g., development, testing, and production), it's often better to automate this process via code. This ensures consistency across deployments and makes it easier to manage changes in configuration.

Automating with Code

While the Azure Portal provides a user-friendly interface for creating indexes and managing your search environment, automation is crucial for larger organizations with complex workflows. By using Azure's API, you can automate the creation of data sources, indexes, and other components, ensuring that your deployments are reproducible across different environments.

```
PUT
https://{SEARCHRESSOURCENAME}.windows.net/datasources/{DTA
SOURCENAME}?api-version=2024-05-01-Preview
```

```
{
    "name": "DATSOURCENAME",
    "description": "DESCRIPTIONOFTHEDATASOURCE",
    "type": "azurefile",
    "credentials": {
        "connectionString": "CONNECTIONSTRINGORTHERAUTH"
    },
    "container": {
        "name": "SHARENAME",
        "query": "QUERY"
    },
    "dataDeletionDetectionPolicy" : {
        "@odata.type"
:"#Microsoft.Azure.Search.SoftDeleteColumnDeletionDetectio
nPolicy",
        "softDeleteColumnName" : "IsDeleted",
        "softDeleteMarkerValue" : "true"
    }
}
```

Below is a simple Python script that automates the creation of a data source for Azure File Share:

https://github.com/JayRHa/Book/blob/main/aiSearch/data_source_creation.py

```
import os
import requests

# Set up environment variables
```

```
azure_search_secret = os.getenv("AZURE_SEARCH_KEY")
azure_search_endpoint = "https://your-search-
resource.search.windows.net"
file_share_connection_string =
os.getenv("FILE_SHARE_CONNECTION_STRING")

# Define the data source
data_source_name = "ds-secret-intune-info"
datasource_url =
f"{azure_search_endpoint}/datasources/{data_source_name}?a
pi-version=2024-05-01-Preview"

body = {
    "name": data_source_name,
    "description": "It is very secret",
    "type": "azurefile",
    "credentials": {"connectionString":
file_share_connection_string},
    "container": {"name": "intune", "query": "secret"},
    "dataDeletionDetectionPolicy": {
        "@odata.type":
"#Microsoft.Azure.Search.SoftDeleteColumnDeletionDetection
Policy",
        "softDeleteColumnName": "IsDeleted",
        "softDeleteMarkerValue": "true",
    },
}

# Send the request to create the data source
response = requests.put(
    datasource_url,
    json=body,
    timeout=1000,
    headers={"api-key": azure_search_secret, "Content-
Type": "application/json"},
)

# Output the response
print(response.json())
```

This script demonstrates how to automate the creation of a data source. Similar methods can be applied to create indexes, indexers, aliases, and skillsets. The URLs for each API endpoint are structured similarly, but the body of the request will differ depending on the component you are creating:

- Index:

```
f"{azure_search_endpoint}/indexes/{index_name}?api-version=2024-05-01-Preview"
```

- Indexer:

```
f"{azure_search_endpoint}/indexers/{indexer_name}?api-version=2024-05-01-Preview"
```

- Skillset:

```
f"{azure_search_endpoint}/skillsets/{skillset_name}?api-version=2024-05-01-Preview"
```

- Alias:

```
f"{azure_search_endpoint}/aliases/{alias_name}/reset?api-version=2024-05-01-Preview"
```

This method is ideal for environments where consistency and scalability are critical across multiple stages.

Enabling Vector Search

Vector search is a game changer for applications that need to deliver relevant and contextual search results. Unlike traditional keyword-based searches, vector search understands the semantic meaning of queries by leveraging AI-powered embeddings.

How It Works:

1. **Embedding Models:** Azure AI Search integrates with models like text-embedding-3-small or large to transform your data into vectors. These vectors

represent the meaning behind words and phrases, making search results more accurate and contextually relevant.

2. **Practical Use Case:** For example, if you're indexing documents related to customer support, vector search allows users to find relevant content based on their queries, even if the exact wording doesn't match the indexed documents.

By combining Azure AI Search with vector search, you can provide a much richer and more meaningful search experience for your users.
To enable this, you'll need to create a skillset with an AI embedding model and configure the necessary fields in your search index. Below is an example skillset configuration that includes a split skill and embedding skill:

```
{
  "name": "SKILLSETNAME",
  "description": "...": [
    {
      "@odata.type":
"#Microsoft.Skills.Text.AzureOpenAIEmbeddingSkill",
      "name": "RANDOMNAME",
      "description": null,
      "context": "/document/pages/*",
      "resourceUri": "https://your-oai-
resource.openai.azure.com",
      "deploymentId": "text-embedding-3-large",
      "apiKey": "APIKEY",
      "modelName": "experimental",
      "inputs": [
        {
          "name": "text",
          "source": "/document/pages/*"
        }
      ],
      "outputs": [
```

```
        {
          "name": "embedding",
          "targetName": "vector"
        }
      ],
      "authIdentity": null
    },
    {
      "@odata.type": "#Microsoft.Skills.Text.SplitSkill",
      "name": "RANDOMNAME",
      "description": "Split skill to chunk documents",
      "context": "/document",
      "defaultLanguageCode": "en",
      "textSplitMode": "pages",
      "maximumPageLength": 2000,
      "pageOverlapLength": 500,
      "maximumPagesToTake": 0,
      "inputs": [
        {
          "name": "text",
          "source": "/document/Description"
        }
      ],
      "outputs": [
        {
          "name": "textItems",
          "targetName": "pages"
        }
      ]
    }
  ],
  "cognitiveServices": null,
  "knowledgeStore": null,
  "indexProjections": {
    "selectors": [
      {
        "targetIndexName": "NAMEOFTHEINDEX",
        "parentKeyFieldName": "parent_id",
        "sourceContext": "/document/pages/*",
        "mappings": [
          {
            "name": "chunk",
            "source": "/document/pages/*",
            "inputs": []
          },
          {
            "name": "vector",
```

```
            "source": "/document/pages/*/vector",
            "inputs": []
          },

          ...ALL_OTHER_ATTRIBUTES...
        ]
      }
    ],
    "parameters": {
      "projectionMode": "skipIndexingParentDocuments"
    }
  },
  "encryptionKey": null
}
```

In your index configuration, you'll need to define vector search fields like this:

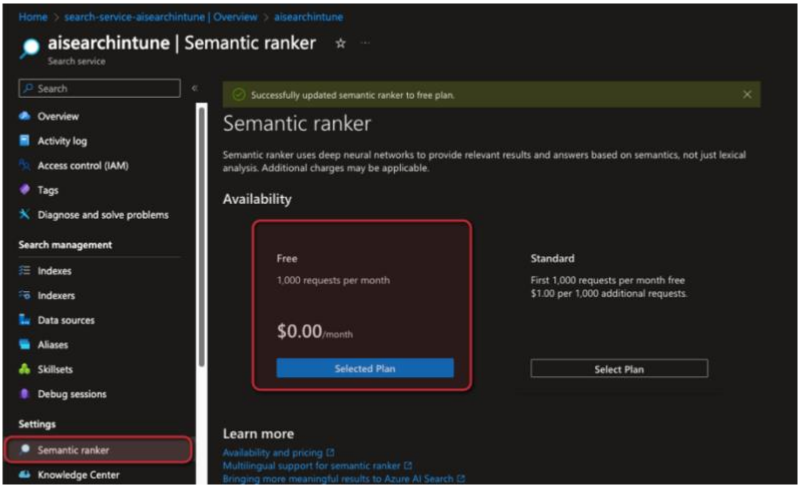
```
{
  "name": "INDEXNAME",
  "defaultScoringProfile": "SCORINGNAME",
  "fields": [
    ...ALL_OTHER_ATTRIBUTES...
    {
      "name": "parent_id",
      "type": "Edm.String",
      "searchable": false,
      "filterable": true,
      "retrievable": true,
      "sortable": false,
      "facetable": true,
      "key": false,
      "indexAnalyzer": null,
      "searchAnalyzer": null,
      "analyzer": null,
      "normalizer": null,
      "dimensions": null,
      "vectorSearchProfile": null,
      "synonymMaps": []
    },
    {
      "name": "chunk",
      "type": "Edm.String",
      "searchable": true,
      "filterable": false,
```

```
      "retrievable": true,
      "sortable": true,
      "facetable": false,
      "key": false,
      "indexAnalyzer": null,
      "searchAnalyzer": null,
      "analyzer": null,
      "normalizer": null,
      "dimensions": null,
      "vectorSearchProfile": null,
      "synonymMaps": []
    },
    {
      "name": "vector",
      "type": "Collection(Edm.Single)",
      "searchable": true,
      "filterable": false,
      "retrievable": true,
      "sortable": false,
      "facetable": false,
      "key": false,
      "indexAnalyzer": null,
      "searchAnalyzer": null,
      "analyzer": null,
      "normalizer": null,
      "dimensions": 3072,
      "vectorSearchProfile": "vector-profile",  #Should be
the same
      "synonymMaps": []
    }
  ],
  "scoringProfiles": [
    YOUR_SCORING_CONFIG
  ],
  "corsOptions": null,
  "suggesters": [],
  "analyzers": [],
  "normalizers": [],
  "tokenizers": [],
  "tokenFilters": [],
  "charFilters": [],
  "encryptionKey": null,
  "similarity": {
    "@odata.type":
"#Microsoft.Azure.Search.BM25Similarity",
    "k1": null,
    "b": null
```

```
  },
  "semantic": {
    "defaultConfiguration": null,
    "configurations": [
      YOUR_SEMANTIC_SEARCH_CONFIG
    ]
  },
  "vectorSearch": {
    "algorithms": [
      {
        "name": "RANDOMNAME",
        "kind": "hnsw",
        "hnswParameters": {
          "metric": "cosine",
          "m": 4,
          "efConstruction": 400,
          "efSearch": 500
        },
        "exhaustiveKnnParameters": null
      }
    ],
    "profiles": [
      {
        "name": "vector-profile", #Should be the same
        "algorithm": "RANDOMNAME",
        "vectorizer": "RANDOMNAME"
      }
    ],
    "vectorizers": [
      {
        "name": "RANDOMNAME",
        "kind": "azureOpenAI",
        "azureOpenAIParameters": {
          "resourceUri": "https://your-oai-
resource.openai.azure.com",
          "deploymentId": "text-embedding-3-large",
          "apiKey": "APIKEY",
          "modelName": "experimental"
        },
        "customWebApiParameters": null
      }
    ]
  }
}
```

Adding the Semantic Ranker and Understanding Search Algorithms in Azure AI Search

In addition to the powerful capabilities of vector search, Azure AI Search also supports other search algorithms, including keyword search and semantic search. These options allow you to customize the search experience based on your specific needs, providing a flexible and robust solution for various search scenarios.



What is the Semantic Ranker?

The **Semantic Ranker** is a feature in Azure AI Search that leverages natural language understanding to rank search results more effectively. Instead of simply matching keywords, the Semantic Ranker analyzes the intent behind a query and ranks documents based on their relevance to that intent. This makes it particularly useful for queries where the user is looking for answers or insights rather than specific keywords.

Semantic search uses advanced machine learning techniques to understand the context of both the query and the documents. This allows Azure AI Search to return more relevant results, even when the query is not an exact match to the document's content.

For instance, in a customer support scenario, if a user searches for “How do I reset my password?” a traditional keyword search might only match documents containing the words "reset" and "password." However, a semantic search would understand the context of the query and rank documents that provide actual instructions for resetting a password higher, even if they don't contain the exact phrasing of the query.

Different Search Algorithms Supported by Azure AI Search

Azure AI Search supports three main search algorithms, each designed for different use cases and levels of complexity:

1. **Keyword Search:**

- **Description:** This is the most traditional form of search, where results are ranked based on the presence and frequency of specific keywords in the documents. The algorithm scans the text for exact matches or variations of the keywords provided in the search query.
- **Use Case:** Keyword search is most effective for scenarios where precision is key, and users are

likely to search for exact terms, such as in legal document searches or database lookups.

- **Strengths:** Fast and efficient for exact term matching.
- **Limitations:** Does not account for context or semantic meaning, which can lead to less relevant results for complex queries.

2. **Semantic Search:**

- **Description:** Semantic search goes beyond keywords and attempts to understand the meaning behind a query. It uses machine learning and natural language processing (NLP) to match the intent of the query with the content of the documents. This type of search is particularly useful for queries that are more conversational or require a deeper understanding of the context.
- **Use Case:** Ideal for scenarios where users expect intuitive, human-like interactions, such as customer support systems, knowledge bases, or any environment where users ask questions in natural language.
- **Strengths:** Provides highly relevant results based on context and intent, even when exact keyword matches are absent.
- **Limitations:** Requires more computational resources and may have slightly higher latency than keyword search.

3. **Vector Search:**

- **Description:** Vector search leverages AI-powered embeddings to transform both the query and the content into vector representations that capture semantic meaning. Instead of matching words, vector search matches the meaning behind the query and the content by comparing the distance between vectors in a high-dimensional space. This approach is particularly powerful for complex and nuanced queries.

- **Use Case:** Best suited for scenarios where context is crucial and where traditional keyword or semantic search may fall short, such as in recommendation engines, content discovery, or advanced knowledge retrieval systems.
- **Strengths:** Offers the most contextually relevant results, especially for complex or ambiguous queries, by understanding the relationships between concepts.
- **Limitations:** Requires advanced setup, including embedding models, and may have higher computational overhead.

Each of these algorithms can be combined and fine-tuned to suit your specific application needs. For instance, you could start with a keyword search for broad matching, followed by a semantic ranker to improve result relevance, and finally apply a vector search for the most contextually rich and accurate results.

Enabling Semantic Ranking

To enable semantic ranking in Azure AI Search, you need to configure your search index to use semantic configurations. This involves defining how the system

should rank documents based on semantic relevance rather than just keyword matches.
Here’s an example of how to set up semantic ranking in your search index configuration:

```
{
    "semantic": {
        "defaultConfiguration": {
            "name": "default",
            "prioritizedFields": [
                {
                    "fieldName": "content",
                    "weight": 1.0
                }
            ]
        }
    }
}
```

In this example, the search results will be ranked based on the semantic relevance of the "content" field in your documents. You can adjust the configuration to prioritize other fields or fine-tune the weights to optimize for different types of queries.

Combining Search Algorithms

One of the powerful aspects of Azure AI Search is that you can combine these different algorithms to create a layered search experience. For example:

1. **Initial Search with Keywords:** Start by filtering out broad results using traditional keyword matching to quickly narrow down the document set.

2. **Refinement with Semantic Search:** Apply semantic ranking to reorder the results based on contextual relevance.

3. **Contextual Enhancement with Vector Search:** Use vector search to further refine results, ensuring that the most contextually relevant documents appear at the top, even when the exact keywords are not present.

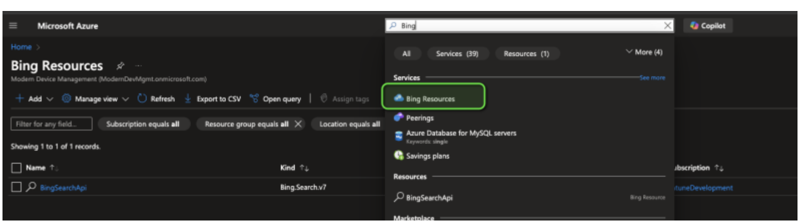
This layered approach can significantly improve the quality of your search results, providing a more intuitive and user-friendly experience.

Integrating a Web Search into Your App

If you need real-time information from the web, you can integrate a web search into your application using **Bing Search API**. Here's how you can do it:

Set Up Bing Search Resources:

In the Azure portal, create a Bing Search resource. This provides you with an API endpoint that allows you to send search queries and retrieve results from near real-time content on the internet.



Integrate Bing Search into Your Application:

When a user submits a query, your application can send the query to the Bing Search API. The results from Bing can be processed and integrated into the model's response, providing up-to-date information directly from the web.

Combining Web Search with Grounded Knowledge:

You can combine the results from Bing Search with your internal knowledge to provide a well-rounded, grounded response. This hybrid approach ensures that the model can pull from both your internal knowledge base and the latest available information online.
Example:

- Bing Search API documentation - Bing Search Services | Microsoft Learn

What is Orchestration?

In the context of M365 Copilot, **orchestration** is about seamlessly integrating various components, services, and data sources to create a cohesive and intelligent experience. It involves managing multiple tasks, making informed decisions, and effectively interacting with users. Here are the key aspects of orchestration:

Integration of Services:

The Copilot needs to integrate with various services, such as Microsoft Graph, Bing, and external APIs. Orchestration ensures that all these services work together smoothly to retrieve the necessary data and provide accurate responses.

Task Management:

Orchestration involves managing and prioritizing tasks. For example, if a user asks for a report based on data from multiple sources, the system must efficiently gather and process that data in the correct order, ensuring that the response is delivered quickly and accurately.

Decision-Making:

The orchestration layer helps the AI Copilot make decisions based on the data it processes. For example, if there are multiple ways to accomplish a task, the orchestration layer decides which method is most efficient or relevant to the user's needs.

User Interaction:

Efficient orchestration ensures that the Copilot interacts with users in a clear, timely, and relevant manner. This includes generating responses that are easy to understand and providing additional context or data when necessary.

In most implementations, the orchestration is handled by a specialized **LLM framework** that ensures all these components work together harmoniously, delivering a cohesive user experience.

This detailed architecture is the backbone of how M365 Copilot works, enabling Microsoft 365 applications to provide powerful, AI-driven assistance that enhances user productivity across various tasks.

Common Frameworks for Building AI-Powered Applications

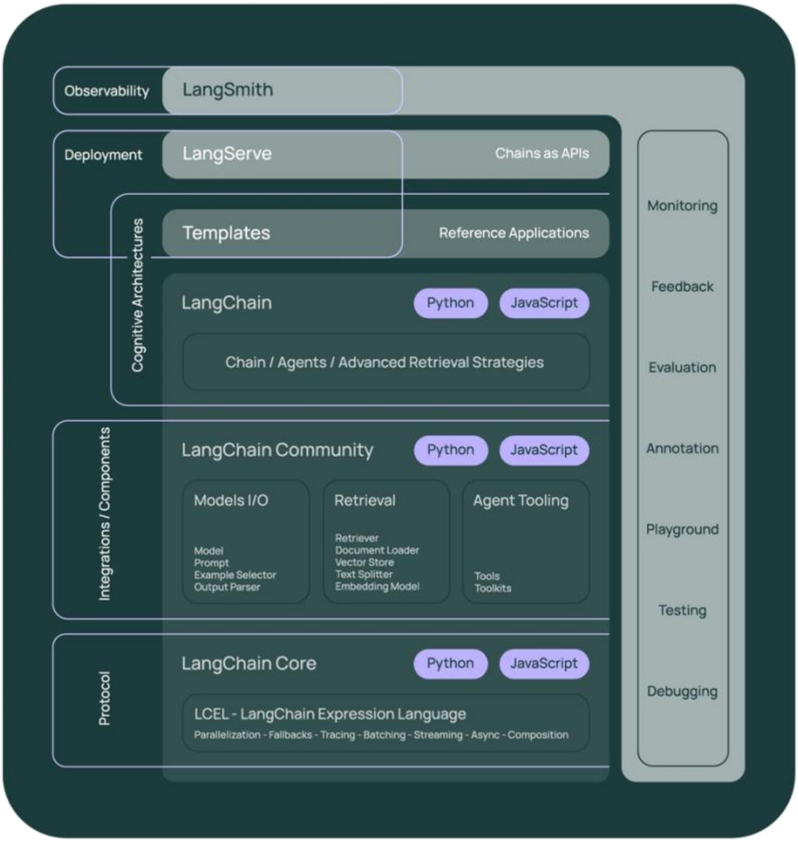
When building AI-powered applications, especially those involving Large Language Models (LLMs), several frameworks can help streamline the development process. Two of the most common and powerful open-source frameworks are **LangChain** and **Semantic Kernel**. Both have gained popularity for their ability to integrate LLMs into various applications, offering different strengths depending on your use case. Let's dive into what these frameworks offer and how they compare.

LangChain

LangChain is a community-driven open-source project that provides a robust framework for creating AI applications such as chatbots, document summarizers, or code analyzers. It is designed to help developers connect LLMs with various tools, services, and data sources, allowing them to build complex language-based applications. Here’s how LangChain operates:

- **Model I/O**: Facilitates communication with language models (e.g., GPT) for generating text-based outputs.
- **Retrieval**: Gathers specific data needed for tasks, such as querying databases or searching documents.

- **Chains**: Links different actions together, creating sequences of tasks that the application performs in response to user input.
- **Agents**: Uses decision-making capabilities to determine which tools or actions to employ based on the user's intent.
- **Memory**: Stores and recalls information from previous interactions, enabling the application to maintain context across conversations.
- **Response**: Provides the final output to the user after processing the input through the appropriate chains, agents, and retrieval methods.



Source: https://js.langchain.com/v0.1/docs/get_started/introduction/

Flexibility: LangChain supports multiple programming languages, including Python and JavaScript, making it a versatile choice for developers. It is particularly powerful for use cases that require integrating various services and maintaining conversation history across sessions. LangChain is popular among developers building chatbots, content generators, and more complex AI-powered workflows.

GitHub Repository: LangChain GitHub

Semantic Kernel

Semantic Kernel is another open-source project that is driven and widely used by Microsoft. It focuses on providing a simpler and more streamlined way to integrate LLMs into applications. While it may not offer as many out-of-the-box tools as LangChain, it is designed to easily integrate with common programming languages like C# and Python and work seamlessly within existing enterprise applications.

Here’s how Semantic Kernel operates:

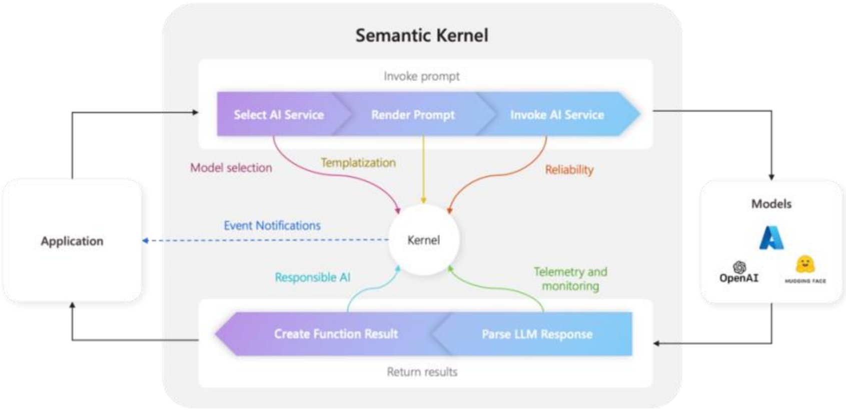
- **Kernel**: Acts as the core manager of tasks, orchestrating interactions between different components.
- **Memories**: Stores and recalls information from previous interactions, similar to LangChain's memory capabilities.
- **Planner**: Decides which tools or plugins to use based on the task at hand, helping automate decision-making.
- **Connectors**: Links to external data sources such as databases, APIs, or other enterprise systems.
- **Plugins**: Allows developers to add custom functions in C# or Python, enabling them to extend the model’s capabilities with bespoke logic.
- **Response**: Delivers the final result after processing, similar to LangChain’s output mechanism.

Simplicity: Semantic Kernel is designed to be lightweight, focusing on efficiently integrating AI capabilities into existing systems. It is particularly suited for scenarios where you want to add intelligent features to an application without needing to build a complex architecture from scratch.

Semantic Kernel Use Cases:

- **AI-Enhanced Enterprise Applications**: Seamlessly embedding AI into existing business tools and workflows.
- **Task Automation**: Automating repetitive tasks within applications, leveraging AI for decision-making.
- **Custom AI Solutions**: Building tailored AI models with integrations to specific business data and services.

GitHub Repository: Semantic Kernel GitHub



Source: https://learn.microsoft.com/en-us/semantic-kernel/concepts/kernel?pivots=programming-language-csharp

Comparison of LangChain and Semantic Kernel

Aspect	LangChain	Semantic Kernel

Flexibility	Highly flexible, with more tools and integrations out of the box. Supports Python and JavaScript.	More streamlined and focuses on integrating AI into existing applications. Strong support for C# and Python.
Use Cases	Ideal for complex workflows that require chaining tasks, maintaining conversation history, and using multiple services.	Best for simpler, more direct integrations of AI into existing enterprise applications.
Community	Large and active community with widespread usage in various AI applications.	Growing community, especially among developers looking to integrate AI into enterprise solutions.
Tooling	Extensive tooling for chatbots, document analysis, and more.	Focuses on extending existing applications with AI features.
Learning Curve	Steeper, due to the range of features and capabilities available.	Easier to pick up, with a focus on simplicity and direct integration.

Both frameworks are excellent choices depending on the requirements of your project. LangChain offers more flexibility and out-of-the-box tools, making it ideal for more complex applications, while Semantic Kernel excels in simplicity and direct integration, particularly in enterprise environments.

Chapter 10: Using Azure OpenAI Services

Now that we've explored Intune's built-in capabilities, custom analytics solutions, and the power of AI-driven copilots, it's time to take the next step: building your own chatbot. This chatbot will leverage the Microsoft Graph API to interact with Intune, enabling automation, querying data, and managing devices using natural language commands. By integrating the Graph API, this custom bot will provide an intuitive and powerful interface to enhance the user experience in endpoint management.

Deploying and Understanding Azure OpenAI Services

Before diving into practical use cases, the first step is deploying Azure OpenAI Services in your Azure environment. This will allow us to build and customize the chatbot to suit our specific needs.

Step 1: Deploying Azure OpenAI Services

1. **Log in to the Azure Portal**: Go to portal.azure.com and log in with your credentials.

2. **Search for Azure OpenAI**: In the search bar, type "Azure OpenAI" and select the service from the results.

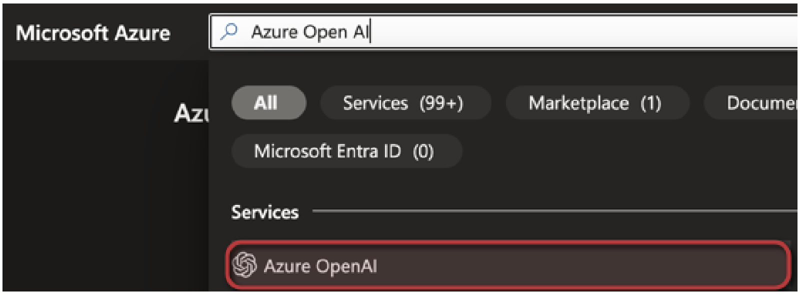


Figure 1: Search for Azure OpenAI Services in the Azure Portal

3. **Create a New Instance**: Click the **+Create** button to begin setting up a new instance of Azure OpenAI Services.

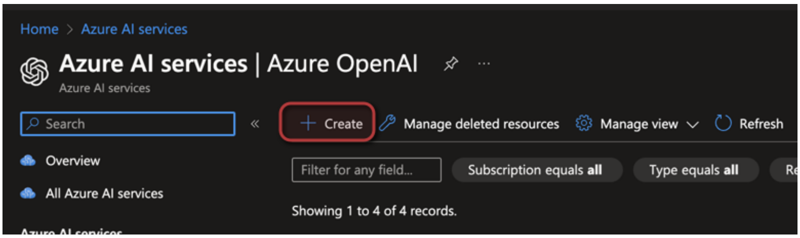


Figure 2: Add Azure OpenAI Service

4. **Configure the Deployment**:

   - **Subscription**: Select the subscription that has been enabled for Azure OpenAI Services.
   - **Resource Group**: Choose an existing resource group or create a new one.
   - **Region**: Select a region for deployment (note that different models may be available in various regions).
   - **Resource Name**: Enter a name for your Azure OpenAI Service instance.

- **Pricing Tier**: Choose the 'Standard S0' pricing tier, which provides sufficient capacity for testing and production scenarios.

After configuring these details, follow the remaining steps in the portal to complete the deployment.

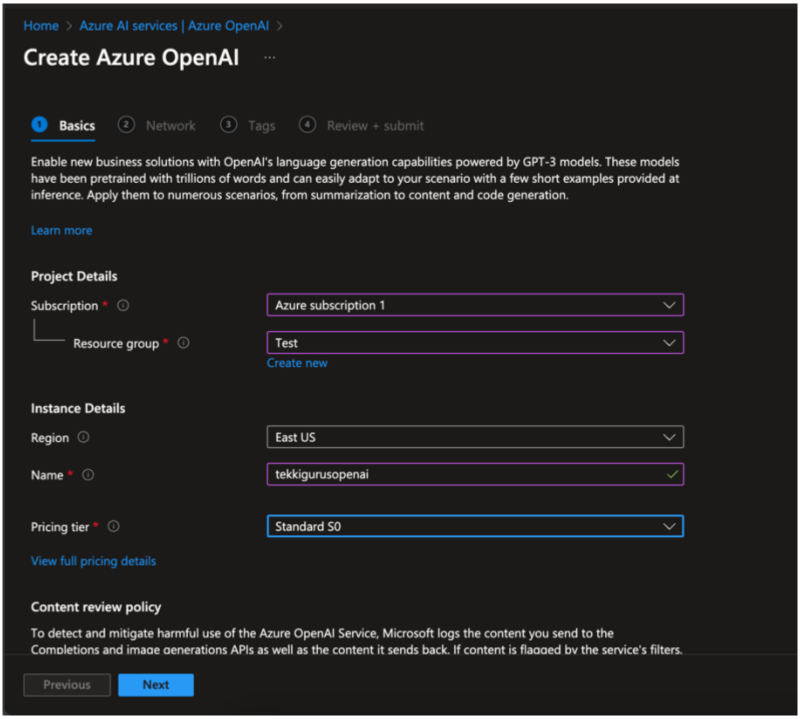


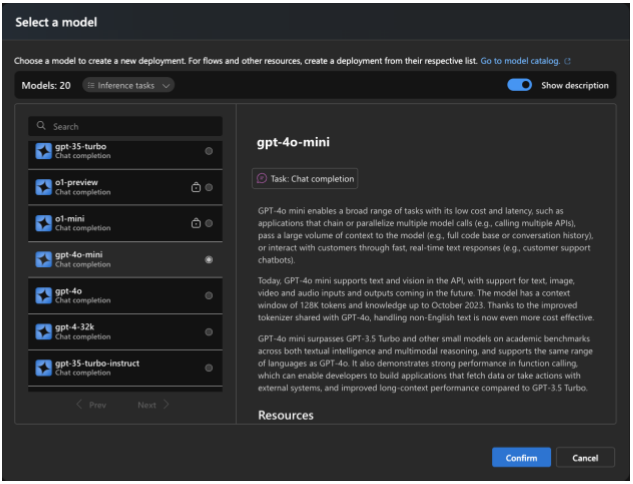
Figure 3: Deploy Azure OpenAI Service

Step 2: Using Azure OpenAI Services

Once your Azure OpenAI Service is deployed, you can begin working with the model by accessing the **Azure OpenAI Studio**.

1. **Access Azure OpenAI Studio**: Navigate to https://oai.azure.com/, log in with your Azure credentials, and explore the tools available for testing and deploying models.

2. **Create a New Deployment**:
   - Go to the **Deployments** section in the Azure OpenAI Studio.
   - Click **+ Create new deployment**.
   - Select the desired AI model (e.g., GPT-4o), name your deployment, and click **Create**.



There are different deployment types:

- **Global Batch**: Ideal for offline, large-scale data processing (e.g., content generation, NLP tasks) where latency isn't critical. Lower cost, file-based async processing.
- **Global Standard**: Best starting point, allowing global routing with moderate real-time performance. Higher quotas than Standard but may have latency variability for high usage.

- **Global Provisioned**: Optimized for real-time, high-throughput, and predictable performance with global routing. Suitable for consistent, large-volume tasks needing stability.
- **Data Zone Standard**: Routes within defined data zones for high availability. Suitable for high volume but with potential latency variability; provisioned offers lower latency for large volumes.
- **Standard**: Pay-per-call model, fast and easy to start, optimized for low to medium volume with high burstiness. Latency may vary with high consistent volume.
- **Provisioned**: Dedicated throughput via PTUs, guaranteeing stable capacity for consistent high-volume real-time applications.

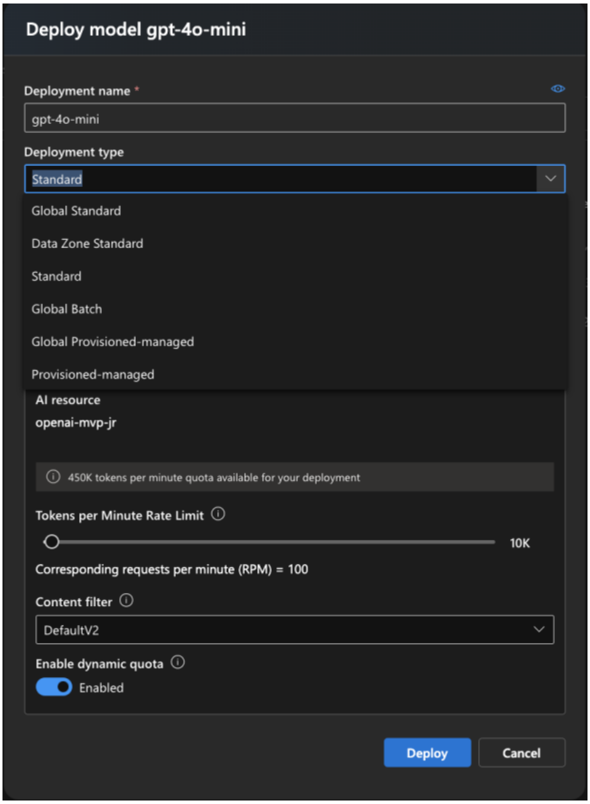


Figure 4: Deploy Model in Azure OpenAI Studio

3. **Explore Azure OpenAI Studio**: Azure OpenAI Studio offers a wide range of features for experimenting, configuring, and deploying your chatbot:

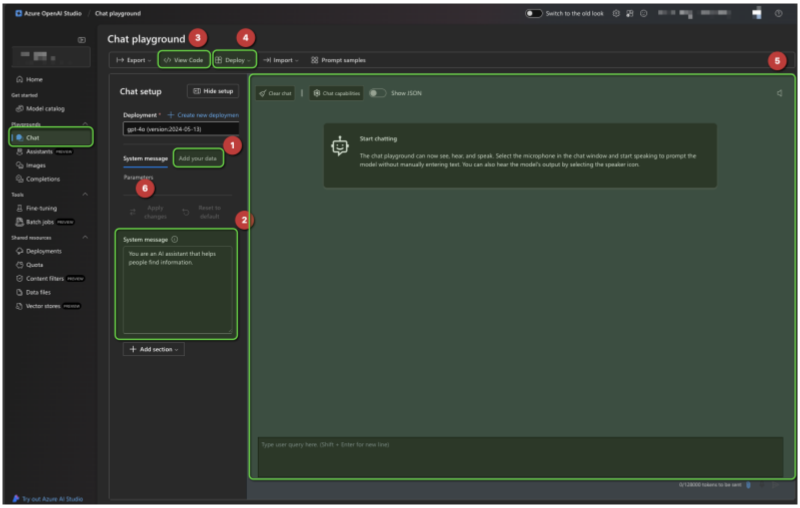
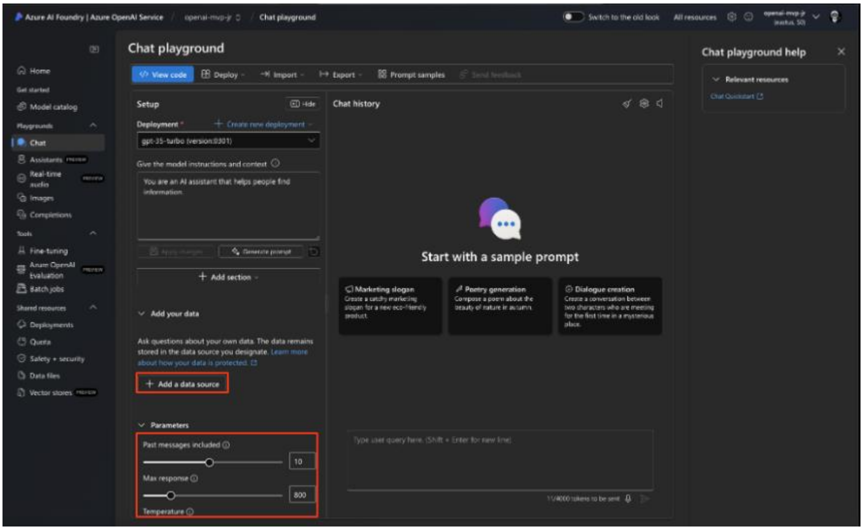
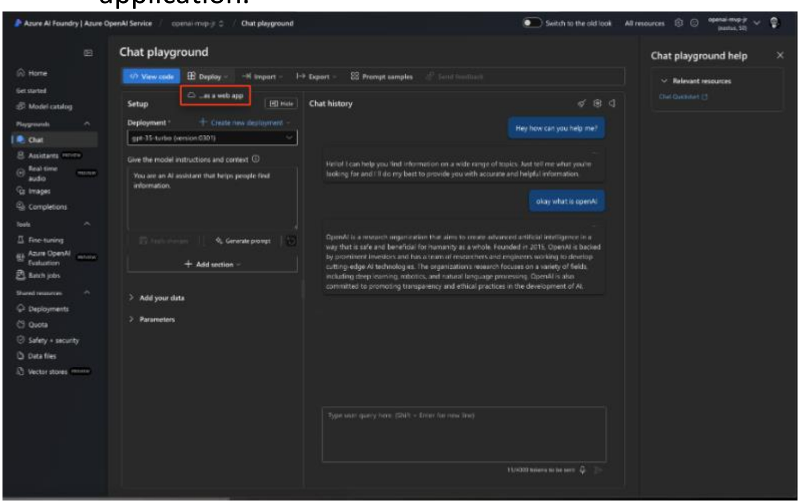


Figure 5: Azure OpenAI Studio Tour

1. **Custom Data Integration**: Link your custom data, such as data stored in Azure Blob Storage or indexed using Azure AI Search, to ground the model's responses.
2. **System Message Settings**: Set the tone or persona of your model and provide specific instructions for how it should respond.
3. **Code Generation**: Generate and integrate code in various programming languages to automate tasks or extend functionality.



4. **Easy Chatbot Deployment**: Use the **Deploy to** button to deploy your chatbot to an Azure App Service, enabling quick deployment of a working application.



5. **Interactive Playground**: Use the ChatGPT-like interface to test prompts and refine the model's output in real time.

6. **Parameter Adjustments**: Modify hyperparameters like temperature to control the creativity and variability of the model's responses.

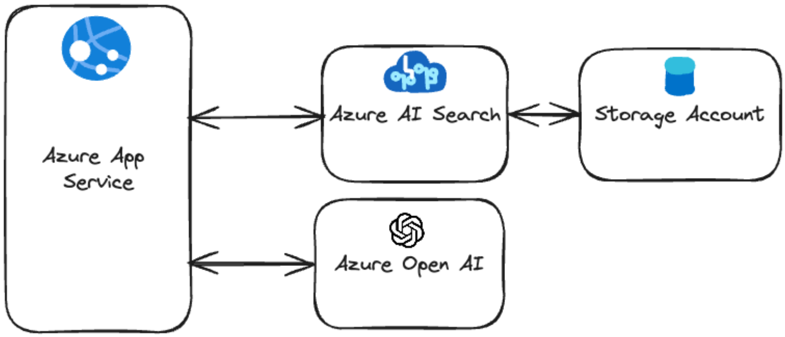
Example Implementations

Now, let's look at some practical implementations of these search algorithms, focusing on integrating them into your chatbot co-pilot setup.

Example 1: Setting Up the Chatbot Co-Pilot

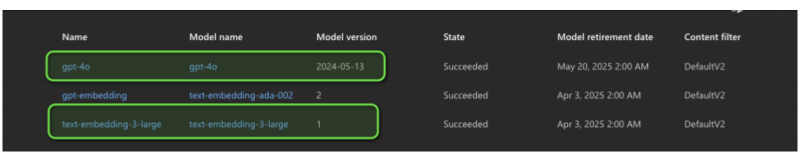
Deploying a chatbot without coding can be a simple and powerful way to utilize Azure OpenAI Services. By integrating your custom documentation, knowledge articles, or any other resources, you can build a chatbot that aligns with your specific needs, all without writing a single line of code.

This guide will walk you through setting up a chatbot that uses your documents stored in Azure Blob Storage, indexed by Azure AI Search, and connected to Azure OpenAI Services. The result will be a dedicated webpage featuring a chat interface that responds to user queries by referencing your custom data.



Prerequisites:

- Azure Blob Storage Account: To store your documents. (Shown above)
- Azure AI Search Instance: To enable content retrieval and indexing. (Shown above)
- Azure OpenAI Account: For leveraging Azure OpenAI Services. (Shown above)
    - Create a deployment for GPT-4o and **text-embedding-3-large**



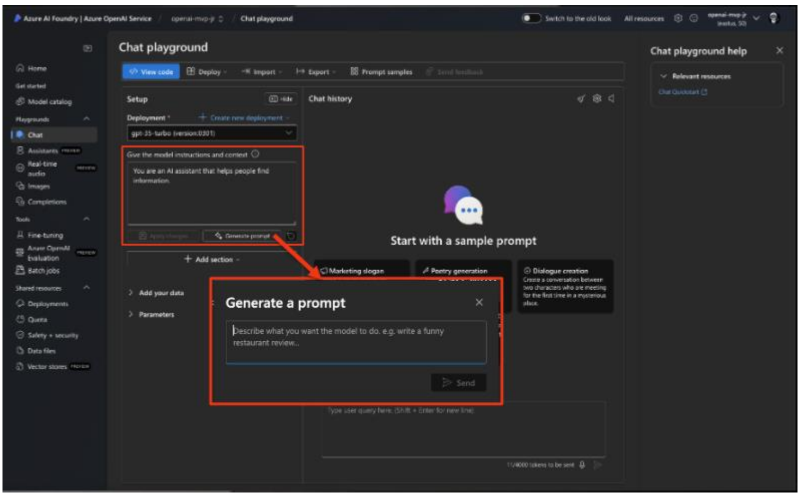
Name	Model name	Model version	State	Model retirement date	Content filter
gpt-4o	gpt-4o	2024-05-13	Succeeded	May 20, 2025 2:00 AM	DefaultV2
gpt-embedding	text-embedding-ada-002	2	Succeeded	Apr 3, 2025 2:00 AM	DefaultV2
text-embedding-3-large	text-embedding-3-large	1	Succeeded	Apr 3, 2025 2:00 AM	DefaultV2

Configuring the Chatbot in Azure OpenAI Studio

Configure various settings and parameters, and connect it to your data, such as with AI Search. The following sections will provide detailed information about AI Search.

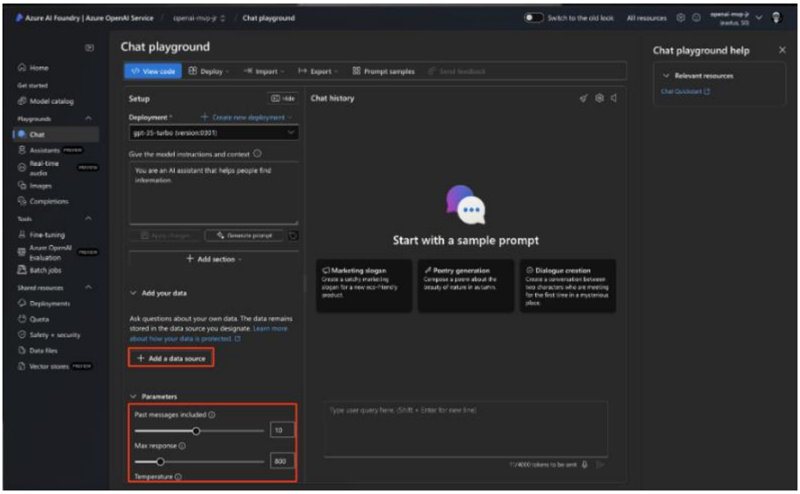
Define the System Message:

- Go to the Chat section in Azure OpenAI Studio.
- Set up a system message to define the chatbot's behavior. For example: "You are a senior expert in Intune. Please only answer questions related to Intune and respond only when you are confident in your answer."
- Click Apply changes to save.



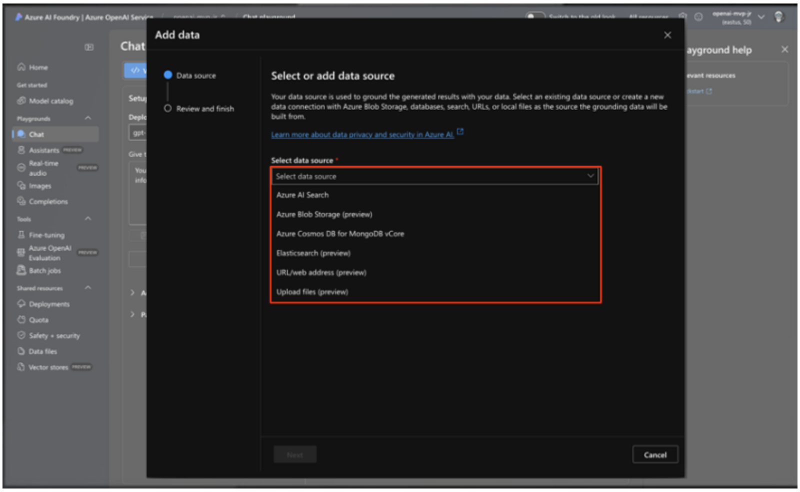
Adjust Parameters:

- Change the temperature parameter to 0 to minimize the likelihood of hallucinations, ensuring the chatbot provides more deterministic and factual responses.

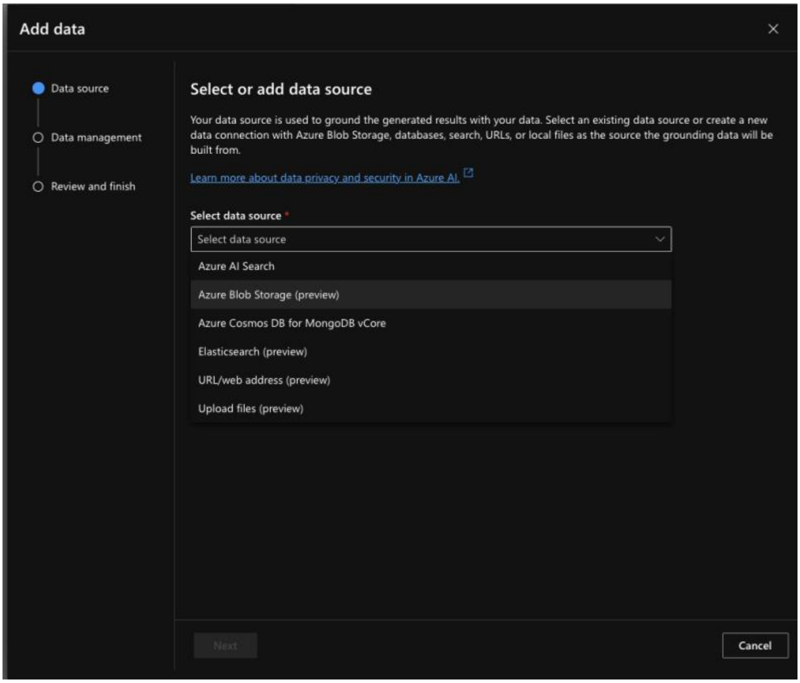


Integrate Your Data:

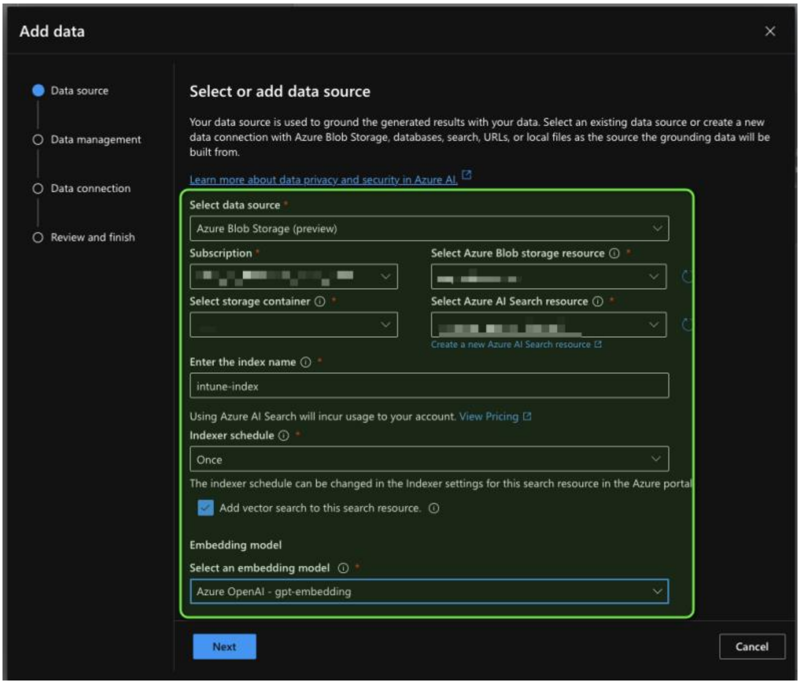
- In the Add your data section, click + Add a data source.



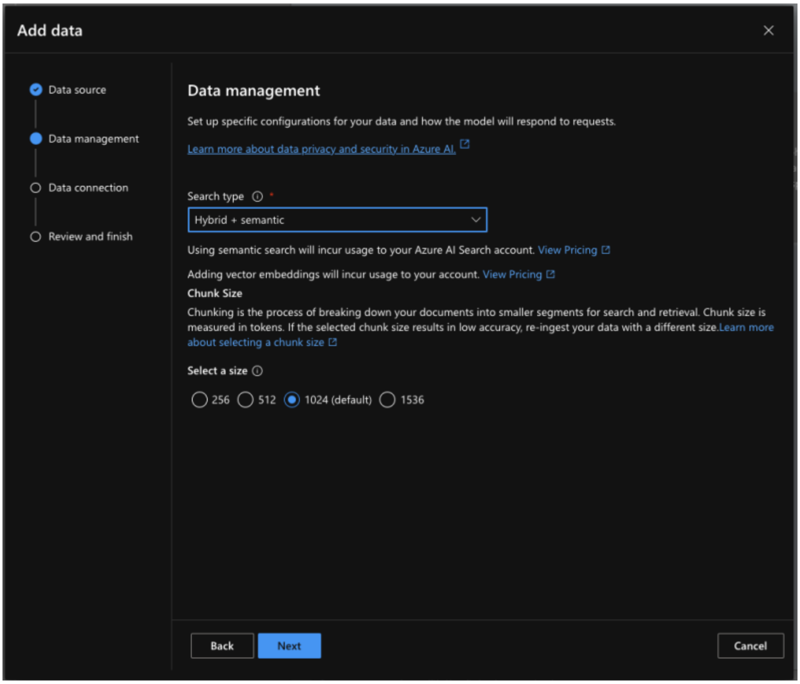
- Choose Azure Blob Storage as the data source.



- Configure the connection to your Blob Storage and Azure AI Search. Enter a name for the index, select the embedding model, and configure indexing schedules.



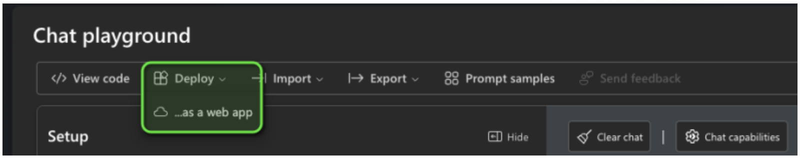
- Activate Vector Search:
    - Enable vector search for better results by selecting your embedding model.



- Finalize Setup:
  - Click Save and Close after configuring the search and data integration. The process may take a few seconds to complete.

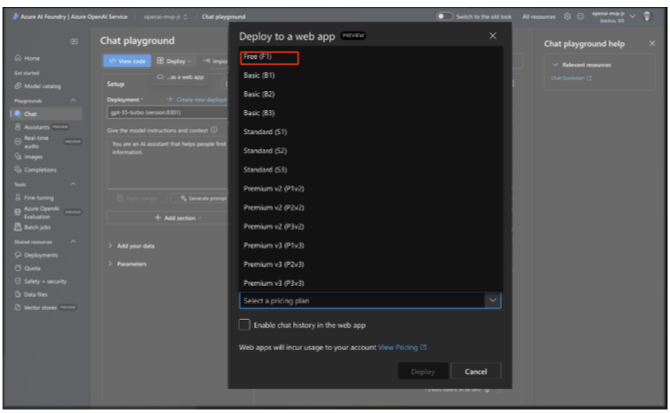
Deploy the Chatbot Webpage

After configuring and testing in Azure OpenAI Studio, deploy your chatbot to a production environment using Azure App Service. To do this click on deploy and ...as a web app



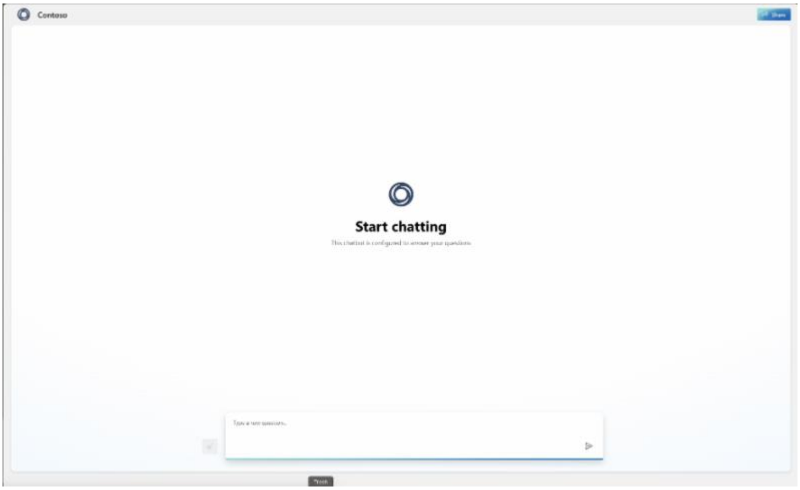
Once the deployment is done you will get a webpage like this which you can provide to your users.

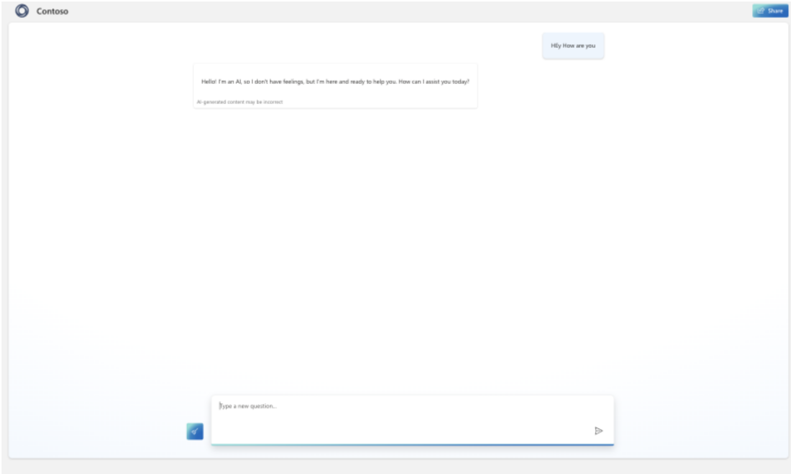
- Choose the subscription and resource group for the web app deployment.
- Enter a name for the web app (this will form part of the URL).
- For cost efficiency, select the Free pricing plan (note that this plan comes with no SLAs).
- Click Deploy.



<u>Monitor Deployment:</u>

The deployment may take a few minutes. Once it's completed, you can open the deployed web app to test your chatbot. The chatbot will be functional and ready to respond to user queries by referencing your custom documents.





Customizing the Chatbot's Look and Feel

If you want to customize the appearance and functionality of the chatbot interface, you can do so by cloning the sample repository provided by Microsoft:

1. **Clone the Repository:**
   - Use this GitHub repository: Microsoft Sample App - AOAI ChatGPT.
   - Modify the HTML, CSS, or JavaScript files as needed to tailor the chatbot to your organization's branding and functionality requirements.

2. **Deploy the Customized Version:**
   - After making changes, redeploy the web app with your custom configurations.

Next Steps: Building and Customizing Your Chatbot

Once you've deployed your bot you can use the page and the code as basis to make your own assistant out of it. E.g. that the bot will leverage Microsoft Graph API to interact with Intune and other Microsoft services, enabling natural language automation and management tasks.

Potential Features to Implement:

- **Device Management**: Query device compliance, install or remove apps, and get detailed reports on device health.

- **Policy Enforcement**: Automate the application of security policies and configurations.

- **Real-Time Alerts**: Notify administrators of security incidents or non-compliance issues.

- **Custom Analytics**: Generate reports based on specific queries, such as application usage, patch status, or device performance metrics.

Example 2: Code Implementation of a Copilot

For a deeper understanding, let’s look at a code-based implementation of the chatbot that integrates Azure AI Search with Azure OpenAI.

Here’s a Python example that demonstrates how to query Azure OpenAI and Azure AI Search:

https://github.com/JayRHa/Book/blob/main/copilot/copilot.py

```
import openai, os, requests

openai.api_type = "azure"
openai.api_version = "2023-08-01-preview"

# Azure OpenAI setup
endpoint_name = "your-openai-endpoint"
openai.api_base = f"https://{endpoint_name}.openai.azure.com/"
openai.api_key = os.getenv("OPENAI_API_KEY")

# Azure AI Search setup
search_endpoint = "https://aisearchintune.search.windows.net"
search_key = os.getenv("SEARCH_KEY")
search_index_name = "intune-index"

def setup_byod(deployment_id: str) -> None:
    class BringYourOwnDataAdapter(requests.adapters.HTTPAdapter):
        def send(self, request, **kwargs):
            request.url = f"
{openai.api_base}/openai/deployments/{deployment_id}/extensions/chat/completions?api-version=
{openai.api_version}"
            return super().send(request, **kwargs)

    session = requests.Session()
    session.mount(prefix=f"{openai.api_base}/openai/deployments/{deployment_id}",
adapter=BringYourOwnDataAdapter())
    openai.requestssession = session

deployment_id = "gpt-4o"
setup_byod(deployment_id)

message_text = [
    {"role": "user", "content": "What are the differences between Azure Machine Learning and
Azure AI services?"}
]

completion = openai.ChatCompletion.create(
    messages=message_text,
    deployment_id=deployment_id,
    dataSources=[
        {
            "type": "AzureCognitiveSearch",
            "parameters": {
                "endpoint": search_endpoint,
                "indexName": search_index_name,
                "semanticConfiguration": "default",
                "queryType": "vectorSemanticHybrid",
                "key": search_key,
                "topNDocuments": 5,
            },
        }
    ],
    temperature=0,
    top_p=1,
    max_tokens=800,
)
print(completion)
```

```
import openai, os, requests

openai.api_type = "azure"
openai.api_version = "2023-08-01-preview"

# Azure OpenAI setup
endpoint_name = "your-openai-endpoint"
openai.api_base =
f"https://{endpoint_name}.openai.azure.com/"
openai.api_key = os.getenv("OPENAI_API_KEY")

# Azure AI Search setup
search_endpoint =
"https://aisearchintune.search.windows.net"
search_key = os.getenv("SEARCH_KEY")
search_index_name = "intune-index"

def setup_byod(deployment_id: str) -> None:
    class
BringYourOwnDataAdapter(requests.adapters.HTTPAdapter):
        def send(self, request, **kwargs):
            request.url =
f"{openai.api_base}/openai/deployments/{deployment_id}/ext
ensions/chat/completions?api-version={openai.api_version}"
            return super().send(request, **kwargs)

    session = requests.Session()

session.mount(prefix=f"{openai.api_base}/openai/deployment
s/{deployment_id}", adapter=BringYourOwnDataAdapter())
    openai.requestssession = session

deployment_id = "gpt-4o"
setup_byod(deployment_id)

message_text = [
    {"role": "user", "content": "What are the differences
between Azure Machine Learning and Azure AI services?"}
]

completion = openai.ChatCompletion.create(
    messages=message_text,
    deployment_id=deployment_id,
    dataSources=[
        {
            "type": "AzureCognitiveSearch",
            "parameters": {
```

```
                "endpoint": search_endpoint,
                "indexName": search_index_name,
                "semanticConfiguration": "default",
                "queryType": "vectorSemanticHybrid",
                "key": search_key,
                "topNDocuments": 5,
            },
        }
    ],
    temperature=0,
    top_p=1,
    max_tokens=800,
)
print(completion)
```

This script integrates Azure OpenAI with Azure AI Search to provide more contextual responses by leveraging vector search.

How to Use This Code in PowerShell

If you prefer PowerShell, you can achieve similar functionality using REST API calls. Here's a PowerShell script example that queries Azure OpenAI:

https://github.com/JayRHa/Book/blob/main/copilot/copilot.ps1

```
# Set API details
$endpoint_name = ""
$apiBase = "https://$($endpoint_name).openai.azure.com"
$apiVersion = "2023-07-01-preview"
$apiKey = [System.Environment]::GetEnvironmentVariable("OPENAI_API_KEY", "User")
$deployment = "gpt-4o"

$headers = @{
    "Content-Type" = "application/json"
    "api-key" = $apiKey
}

$body = @{
    messages = @(
        @{
            role = "system"
            content = "You are an expert in Intune."
        },
        @{
            role = "user"
            content = "What is Intune?"
        }
    )
    max_tokens = 800
    temperature = 0.7
} | ConvertTo-Json

$url = "$apiBase/openai/deployments/$deployment/chat/completions?api-version=$apiVersion"
$response = Invoke-RestMethod -Uri $url -Method Post -Headers $headers -Body $body
$response | Out-String
```

```
# Set API details
$endpoint_name = ""
$apiBase = "https://$($endpoint_name).openai.azure.com"
$apiVersion = "2023-07-01-preview"
$apiKey =
[System.Environment]::GetEnvironmentVariable("OPENAI_API_K
EY", "User")
$deployment = "gpt-4o"

$headers = @{
    "Content-Type" = "application/json"
    "api-key" = $apiKey
}

$body = @{
    messages = @(
        @{
            role = "system"
            content = "You are an expert in Intune."
        },
        @{
            role = "user"
```

```
            content = "What is Intune?"
        }
    )
    max_tokens = 800
    temperature = 0.7
} | ConvertTo-Json

$url =
"$apiBase/openai/deployments/$deployment/chat/completions?
api-version=$apiVersion"
$response = Invoke-RestMethod -Uri $url -Method Post -
Headers $headers -Body $body
$response | Out-String
```

This PowerShell script integrates directly with Azure OpenAI to generate responses, similar to the Python example.

Example 3: Build Your Own Chatbot Interface using Streamlit

There are powerful frameworks, such as **Streamlit** and **Chainlit**, which simplify the process of building web-based chatbot interfaces. Both frameworks are Python-based and have large communities that provide support.

Streamlit Overview

Streamlit allows you to quickly create interactive web applications, including chatbots, with minimal coding. It’s an excellent option for developers who want to prototype and deploy web apps quickly.

To build a chatbot interface using Streamlit, follow these steps:

1. **Install Streamlit:**

```
pip install streamlit
```

2. **Create a Chatbot Interface:**

```
import streamlit as st
import openai

openai.api_key = 'your-openai-api-key'

st.title("Azure AI-Powered Chatbot")

user_input = st.text_input("Ask me anything:")

if st.button("Send"):
    response = openai.Completion.create(
        engine="gpt-4o",
        prompt=user_input,
        max_tokens=150
    )
    st.write(response.choices[0].text)
```

3. **Run the Application:**

```
streamlit run app.py
```

This simple Streamlit app demonstrates the creation of an interactive chatbot interface powered by Azure OpenAI. You can customize the interface and functionality as needed to suit your specific use case.

By using frameworks like Streamlit, you can build and deploy sophisticated chatbot interfaces with ease, making it accessible for developers to iterate and refine their solutions quickly.

Example 4: GPT Device Troubleshooter

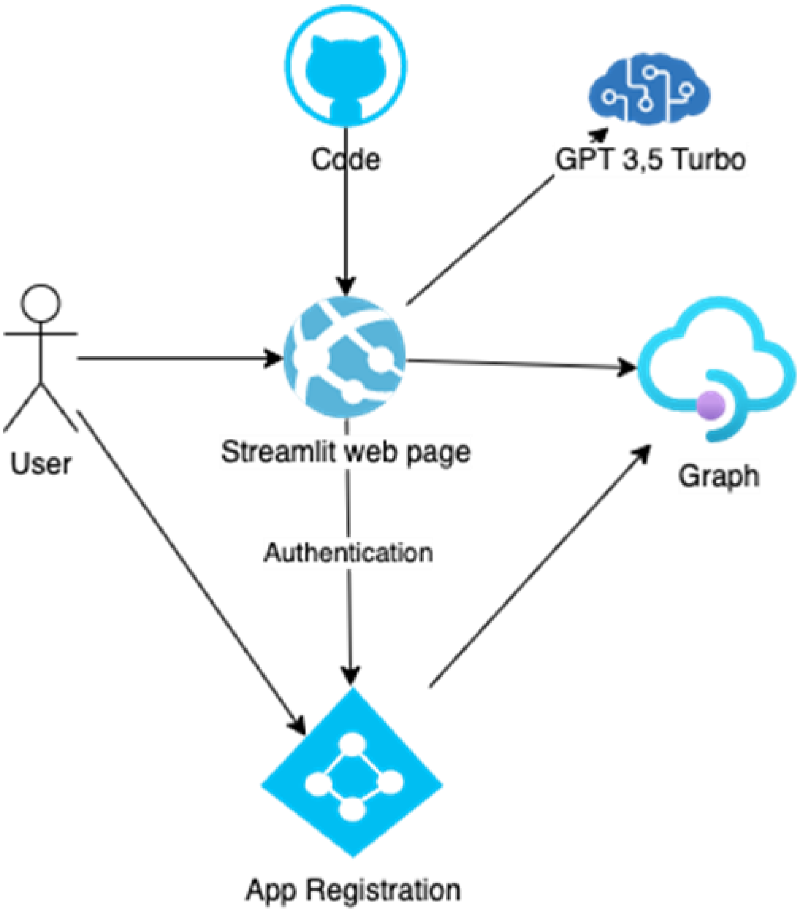
This was a fun project two years ago, where I aimed to demonstrate how you can use GPT to interact with your Intune Tenant. The tool allows users to ask questions and receive precise answers by dynamically generating Microsoft Graph API calls.

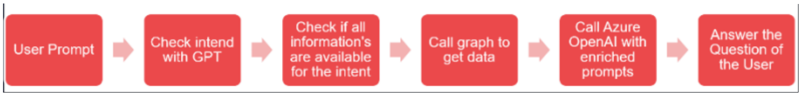
You can find the project here:
https://github.com/JayRHa/GPTDeviceTroubleshooter

How Does It Work?

The GPTDeviceTroubleshooter involves several components working together to deliver an interactive troubleshooting assistant. Below is an overview of the deployment and operation process:

- **Full Infrastructure Deployment via ARM**: The tool's infrastructure is deployed using Azure Resource Manager (ARM) templates, ensuring that all necessary resources are provisioned consistently.
- **Code Pulled from GitHub**: The logic and website code are hosted on GitHub and automatically integrated into the deployment pipeline.
- **User Delegated Permissions**: Users authenticate with their Microsoft account to delegate permissions for accessing Microsoft Graph. This ensures secure and role-based access to tenant resources.
- **Azure App Service**: The assistant is hosted as a web application on Azure App Service, combining a user-friendly interface with server-side logic for API calls and GPT integration.





But let me take you behind the scenes in this book to show you how it works and explain the concept in more detail:

The main component is the website/app.py. Here you can find the webpage and the orchestration logic.

Behind the Scenes

Let's take a closer look at how the tool works under the hood:

Core Components

The main component of the project is the `website/app.py` file. This script acts as the heart of the application, providing the following:

- A user interface powered by **Streamlit**.
- Authentication through the **Microsoft Authentication Library (MSAL)**.
- Backend logic for orchestrating GPT interactions and making Microsoft Graph API calls.

Here's a breakdown of the key features:

User Interface with Streamlit

The application leverages **Streamlit** (Described above in detail), a Python library for creating interactive web applications. Users interact with the application via a sidebar and chat interface.

- **Sidebar Features**:
    - Login/Logout functionality.
    - Buttons to clear the chat conversation or download it as a JSON file.
    - Dynamic user greeting based on authentication status.

- **Main Chat Interface**:

  - Users can input questions related to their Intune devices.
  - The assistant responds by analyzing the prompt, determining its category, and providing appropriate answers or actions.

Authentication and Authorization

The tool uses the **MSAL Streamlit Authentication** module to securely authenticate users. This involves:

- **OAuth Authentication**:
  - Users log in via Azure AD using their credentials.
  - An access token is retrieved, enabling the tool to make secure Graph API calls on behalf of the user.

- **Role-Based Access**:
  - Permissions are granted based on the user's role and their delegated access rights.

Azure OpenAI GPT Integration

The GPT integration adds intelligence to the troubleshooting process:

- **Azure OpenAI Configuration**:
  - The assistant connects to Azure OpenAI Service using an API key, endpoint, and deployment name provided in environment variables.
- **Conversation Handling**:

    - Chat messages (both user input and GPT responses) are stored in the session state to maintain conversation context.
    - The GPT model analyzes user queries and generates dynamic Graph API calls as needed.

Microsoft Graph API Calls

A significant feature of the GPTDeviceTroubleshooter is its ability to interact with Microsoft Intune via the **Microsoft Graph API**. The tool dynamically generates API calls to fetch relevant data or perform actions, such as:

- **Fetching Device Details**: Retrieves a list of devices, their statuses, or detailed information about a single device.
- **Device Group Membership**: Identifies the groups to which a device belongs.
- **Configuration Profiles and Compliance Policies**: Retrieves configuration profiles and compliance policies applied to a device or tenant.
- **Custom Graph Calls**: Users can input custom queries, and the assistant translates them into Graph API calls for execution.

Category-Based Query Handling

The application categorizes user queries into predefined categories to streamline troubleshooting. Some of these categories include:

- *GetDeviceList*: Retrieves a list of devices in the tenant.

- *GetDeviceStatus*: Fetches the operational status of a specified device.
- *GetSingleDevice*: Provides detailed information about a single device.
- *CompliancePolicies*: Lists compliance policies applied within the tenant.
- *DeviceGroupMembership*: Displays group memberships for a specific device.

Each category triggers specific logic in the utility module to generate relevant API calls and process responses.

Key Functions

Here are some of the critical functions within the application:

- **get_graph_access_token**: Generates headers for making authenticated Graph API calls using the retrieved access token.
- **ask_for_device_name**: Ensures the user provides a valid device name before executing actions requiring device-specific information.
- **write_assistant_answer**: Formats and logs the assistant's responses within the chat interface.
- **Category-Specific Logic**: Each category (e.g., *GetDeviceList, GetDeviceStatus*) invokes specific utility functions to handle Graph API calls and return meaningful responses to the user.

Example Workflow

Here's an example of how the GPTDeviceTroubleshooter handles a query:

- **User Interaction**:

    - The user logs in and types a query like, “What is the status of device ABC123?”
- **Query Categorization**:
    - The tool categorizes the query as *GetDeviceStatus*.
    - It extracts the device name (“ABC123”) from the query.
- **Graph API Call**:
    - The utility function generates and executes a Graph API call to fetch the status of device “ABC123”.
- **GPT Response**:
    - GPT interprets the response from the Graph API and translates it into a user-friendly message.
- **Output**:
    - The status of “ABC123” is displayed in the chat interface.

<u>Summary</u>

The GPTDeviceTroubleshooter demonstrates the power of combining GPT intelligence with Microsoft Graph to deliver an intuitive troubleshooting experience. The project showcases how advanced AI models and APIs can work together to create innovative solutions for IT operations.

Chapter 11: How to Build Your Own Chatbot via the Copilot Studio

The Copilot Studio is the next evolution of the Power Platform's Virtual Agent, designed to make building intelligent assistants and chatbots even more accessible and powerful. This chapter will guide you through the process of creating a Copilot using this innovative new tool, which utilizes the Azure OpenAI Service in the background to deliver AI-powered responses.

Prerequisites

Before you begin building your Copilot, ensure you have the following prerequisites in place:

- **A License for Microsoft Copilot Studio:** You'll need a valid license for Microsoft Copilot Studio or an existing Power Virtual Agents license. If you don't have a license yet, you can start with a free trial.

- **Microsoft 365 Tenant Admin Rights:** Your tenant admin must deploy the Dynamics 365 and Copilot Studio apps via the Microsoft 365 admin center.

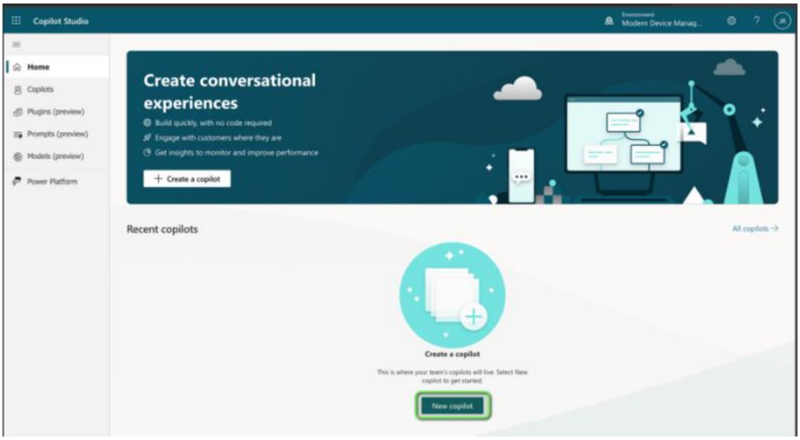
- **Access to the Limited Preview:** Since Copilot Studio is still in limited preview, you must submit a support ticket to gain access.

Getting Started

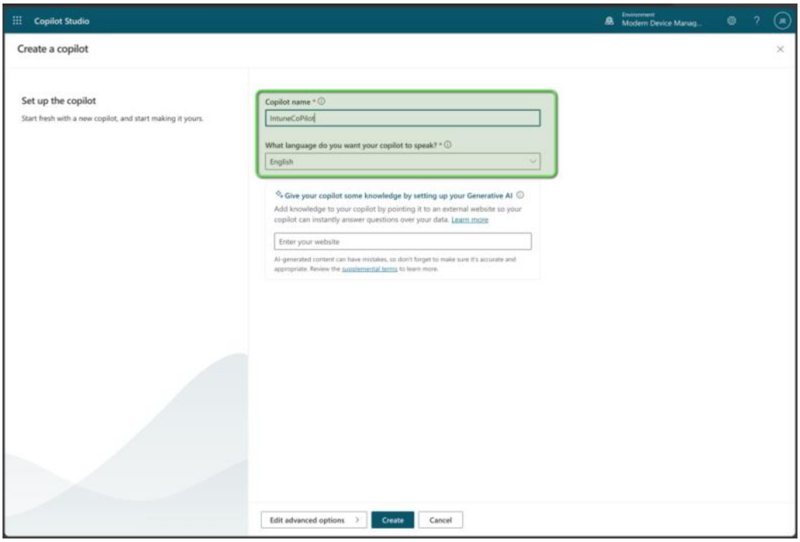
To begin building your Copilot, follow these steps:

1. **Open the Copilot Studio:** Navigate to [Copilot Studio]() and log in with your Microsoft account.

2. **Create a New Copilot:**
    - Click on **New Copilot** to start building your AI-powered assistant.



    - Enter a name for your Copilot and select the language it will use to interact with users.



Once you've named your Copilot and selected a language, your bot will be created, and you can start adding custom data and functionalities using skills and plugins.

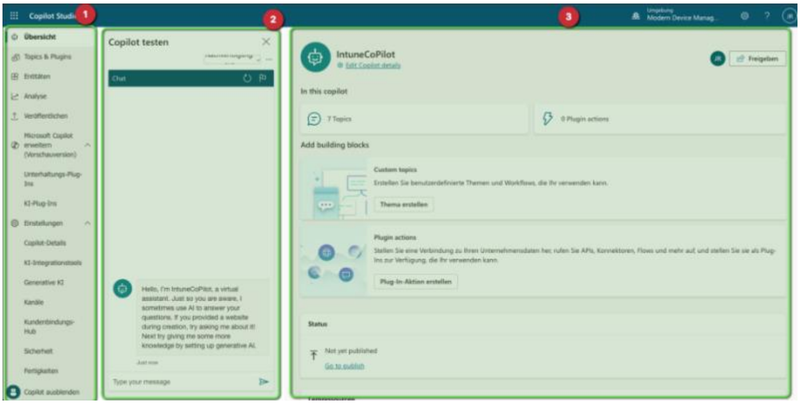
The Studio

The Copilot Studio consists of three essential parts:

1. **Menu Element (#1):** This section allows you to navigate the various options available in the studio, such as creating plugins, adding documents, building flows, and other customization features.

2. **Copilot Playground (#2):** The playground is where you can test your Copilot and validate that everything works as expected after configuring it. It's a live environment for trying out conversations and seeing how the AI responds.

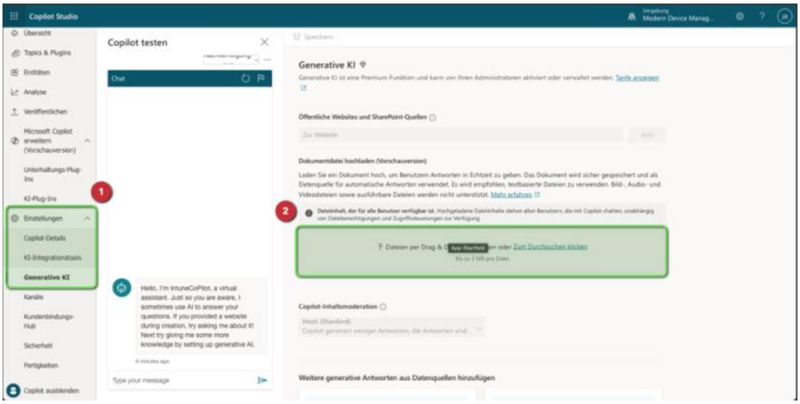
3. **Main Part (#3):** This is the core configuration area where you can set up and customize the options available in the menu element. It's where you'll do most of your work, such as configuring data sources, adding plugins, and defining how your Copilot will operate.



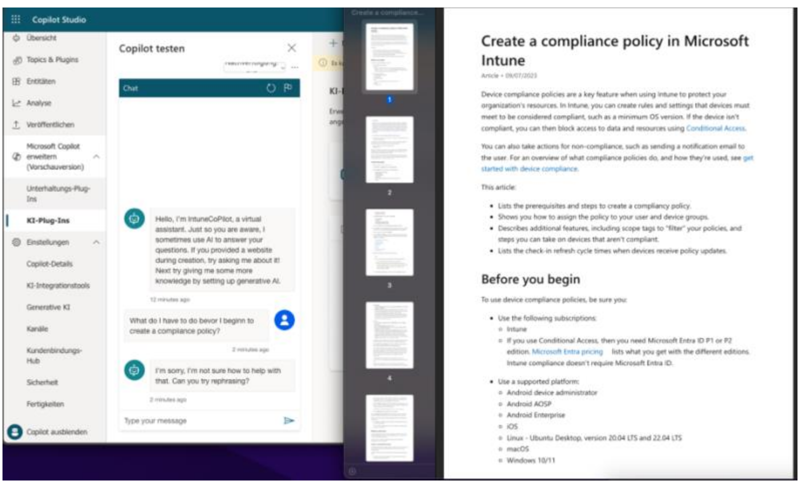
Adding Your Documents

To make your Copilot more knowledgeable and responsive, you can upload documents that the AI will use to generate answers. Here's how:

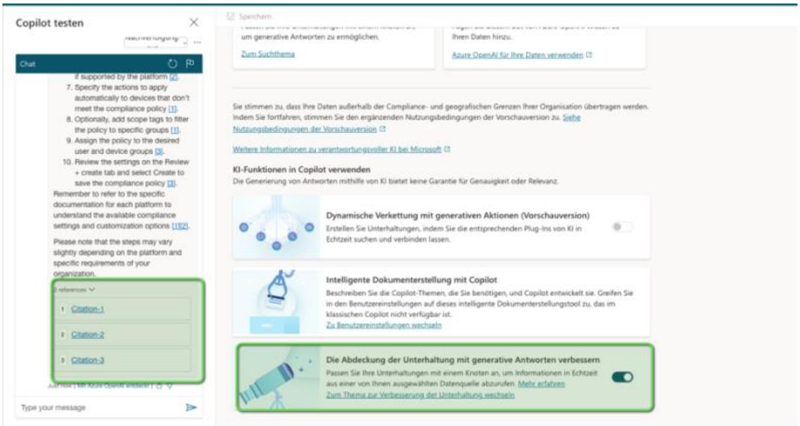
1. **Go to Settings -> Generative AI:** This is where you can upload documents to your Copilot. For example, you might upload an export of a Microsoft Learn document, company-specific training materials, or any other relevant content.



2. **Test Your Copilot:** Once you’ve uploaded a document, test your Copilot by asking it questions related to the content. For example, if you’ve uploaded a Microsoft Learn document, ask a technical question covered in that document.



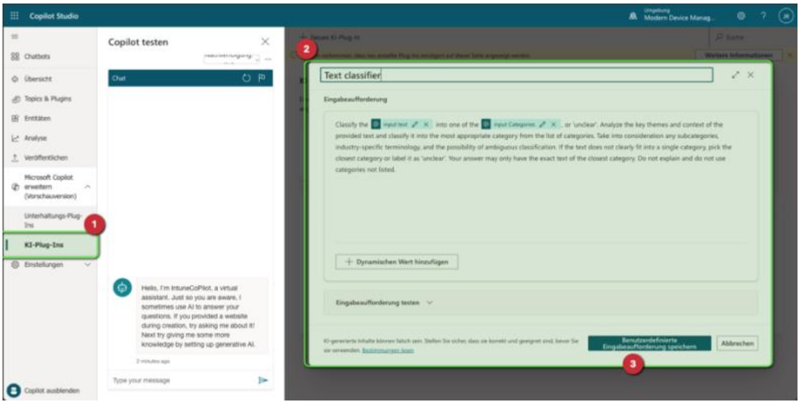
If your Copilot’s response is lacking, check to ensure that you’ve enabled the setting to "Improve the reach of the conversation with generative responses." Once activated, your Copilot will provide answers derived from the documents you've uploaded.



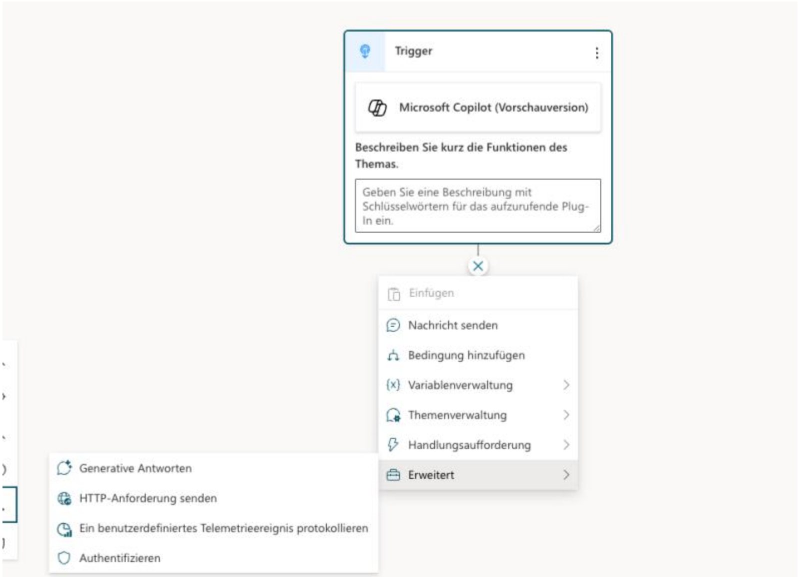
Creating a KI Plug-In

To enhance the capabilities of your Copilot, you can create AI (KI) plugins. There are two types of plugins you can build:

1. **AI Plugin:**
   - This plugin generates responses using GPT based on input variables. For example, you could create a text classifier that takes an input text and assigns it to a category based on predefined options.



2. **Conversational Plugin:**
    - This plugin allows you to build a more complex flow, offering full flexibility to connect to your data. You can create conditions, integrate with the GPT Playground for more complex interactions, and even prompt for authentication if necessary. Conversational plugins are ideal for building custom workflows that guide the user through multi-step processes.



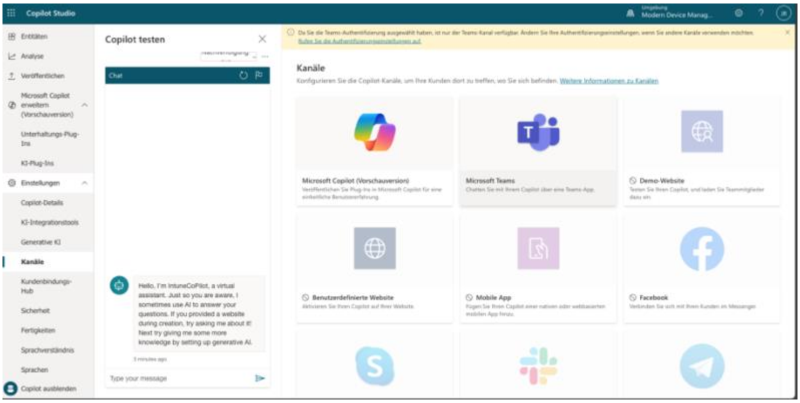
To create a plugin, select **Create a Plugin** in the menu and follow the guided steps. Once your plugin is set up, you can integrate it into your Copilot to extend its functionality.

How to Integrate with Microsoft Teams

One of the powerful features of Copilot Studio is its seamless integration with Microsoft Teams, making it easy to deploy your Copilot to a broader audience within your organization. Here's how to do it:

1. **Navigate to Channels in Settings:** In the settings, select **Channels** and choose **Microsoft Teams** from the available options.

2. **Activate Teams Integration:** Enable the Teams integration by selecting the **Activate Teams** option. Once activated, your Copilot will be available as an add-on within Microsoft Teams.



Now, your team members can interact with your Copilot directly in Teams, making it a convenient tool for internal communication, support, and task automation.

Example: Building a Simple Chatbot Flow

To illustrate the power of Copilot Studio, let's walk through an example of building a simple chatbot flow using a Conversational Plugin:

1. **Create a New Conversational Plugin:**
   - Go to the Plugin section in Copilot Studio and select **New Conversational Plugin**.
   - Name the plugin (e.g., "Employee Onboarding Assistant").
2. **Design the Flow:**

- Start by setting up a welcome message that greets the user and asks how the chatbot can assist with onboarding.
- Add conditions to guide the conversation based on the user's responses. For example, if the user asks about setting up email, the chatbot can provide step-by-step instructions or a link to documentation.

3. **Connect to Data Sources:**
    - You can connect this plugin to your company's internal knowledge base, ensuring that the chatbot pulls accurate and up-to-date information to assist employees.

4. **Deploy to Teams:** Once the plugin is set up and tested in the Copilot Playground, deploy it to Teams so that new employees can easily access onboarding assistance from within their Teams environment.

Conclusion

The Copilot Studio is an incredibly powerful tool that leverages the Azure OpenAI Service to help you build intelligent chatbots without requiring deep coding knowledge. With its ability to integrate with Microsoft Teams, handle complex flows via plugins, and utilize AI-driven responses from uploaded documents, it offers a versatile platform for building AI-powered assistants that can transform your organization's communication and automation efforts.

By following the steps in this chapter, you'll be well on your way to creating your own Copilot, tailored to meet the specific needs of your business. Whether you're looking to enhance customer support, streamline employee onboarding, or create internal tools for your team, Copilot Studio provides the features and flexibility to make it happen.

Some community resources I read and recommend to following

Blogs I read

Intune is more than just a platform we work with every day; it's a passion. The Intune community is comprised of many smart individuals who share content, answer questions, and offer help. Thanks to these people, getting started or finding solutions becomes very easy. I aim to summaries my important links and essential information that you should definitely know. I would like to acknowledge and give special thanks to all of them for this community work

- Scloud (https://scloud.work/de/about) by Florian Salzmann
- Oceanleaf (https://www.oceanleaf.ch/about/) by Nicklas Tinner
- Call4Cloud (https://call4cloud.nl/about/)by Rudy Ooms
- Intune Newsletter (https://andrewstaylor.com/category/newsletter/) by Andrew tailor
- Systanddeploy (https://www.systanddeploy.com/) by Damien Van Robaeys
- Rockenroll.tech (https://www.rockenroll.tech/) by Nicklas Ahlberg
- MS Endpoint Manager (https://msendpointmgr.com/) by Nickolaj Andersen, Maurice Daly, Jan Ketil Skanke (plus multiple outstanding Contributors)

Important to mention is that there are a lot of other outstanding blogs out in the field I really can recommended and I am 1000% sure that I forgot some of them.

Communities

The most outstanding community maybe in the complete IT area is the "Modern Endpoint Management (SCCM | Intune | W365 | AVD | Security | macOS | iOS)" Linked in group founded by Angel Garcia Ayas. You can find here many Microsoft FTEs and MVPs but also a lot of other Experts who share content like blogs and answer questions. In addition there a lot of community events with the highlight of the MEM Summit in Paris.
Other communities I can recommend is it Intune Twitter Community and the Microsoft Enterprise Mobility + Security community on Discord from Jonas Bogvard.

Events

My personal highlight was the MEM Summit in Paris and the MMS. But also there are much more outstanding events like the Workplace Ninja Summit, Experts Live, and many more.

Follow me!

If you are looking for more content from me, you can follow me on:

- **X**: https://x.com/jannik_reinhard
- **LinkedIn**: https://www.linkedin.com/in/jannik-r/

- **Bluesky**: https://bsky.app/profile/jannikr.bsky.social
- **Youtube**: https://www.youtube.com/@ModernDevMgmt
- **Blog**: https://jannikreinhard.com/

Last words

As I write these final words, I find myself overwhelmed with gratitude. This book represents a dream that once felt so far away, an idea that lived in the distant future. And yet, here I am, holding the reality of that dream in my hands.

To my friends and family, thank you for your unwavering support throughout my entire journey. You have been my foundation, my encouragement, and my strength. Without you, none of this would have been possible.
I am endlessly grateful to be a Microsoft MVP, a recognition that has opened so many doors and allowed me to explore my passions. Through blogging, public speaking across the globe, podcasts, videos, and countless other projects, I've been able to contribute to the community I hold so dear.

To my incredible team and colleagues at BASF, thank you for standing by me and for making every day at work a joy. Your support and camaraderie mean the world to me, and I am proud to be part of such an inspiring team.
This book is more than a culmination of my work—it's a testament to the power of community, passion, and perseverance.

To everyone who has supported me along the way, thank you for believing in me and in this vision. This is as much your achievement as it is mine.

With heartfelt gratitude,
Jannik

Download the book as PDF

Would you like to download this book as a PDF for easy sharing with colleagues or store it on your device for quick access? Simply scan the QR code below to get your copy!

https://jannikreinhard.com/book-ai-automation-and-analytics-transforming-device-management-with-intune/

In today's rapidly evolving world, artificial intelligence (AI) is reshaping the landscape of device management. But AI is only as powerful as the data behind it. This book bridges the gap between understanding AI, accessing the right data, and building innovative solutions using Microsoft Intune, Microsoft Graph, Azure, and their surrounding ecosystems.

Designed for both beginners and experienced professionals, this hands-on guide will empower you to:

- Grasp the fundamentals of AI and its transformative potential.
- Harness data to create robust analytics and automation solutions.
- Explore real-world projects you can replicate and customize.
- Gain insights from leading companies on analytics, automation, and AI in device management.

Through practical examples, expert perspectives, and actionable techniques, you'll learn how to take device management to the next level. Whether you're exploring AI for the first time or seeking inspiration for advanced solutions, this book equips you with the tools and knowledge to thrive.

Take the next step in your AI and device management journey.