TECHNICAL WHITE PAPER

Modernized Data Protection for Microsoft 365

Drew Russell, GTM Lead for Microsoft 365
Johanna Braun, Software Engineer
Kiersten Nordin, Product Marketing Manager
Filip Verloy, Field CTO

May 2021
RWP-0567

TABLE OF CONTENTS

3 INTRODUCTION

3 Considerations for Microsoft 365 Protection

3 Microsoft 365 Shared Responsibility Model

4 ARCHITECTURE AND COMPONENTS

4 Installation

7 Microsoft Azure Components

8 Logical Architecture (Customer-hosted)

13 Microsoft 365 API Management

14 CONCLUSION

14 VERSION HISTORY

INTRODUCTION

The purpose of this white paper is to provide you with a clear technical reference regarding Rubrik's Microsoft 365 data management offering. After reading this document, you should be able to answer the following questions:

- How does Rubrik protect Microsoft 365?
- What are the benefits of Rubrik's Microsoft 365 solution?
- How does Rubrik's Microsoft 365 solution work?

Such information will prove valuable while evaluating, designing, or implementing the technologies described herein.

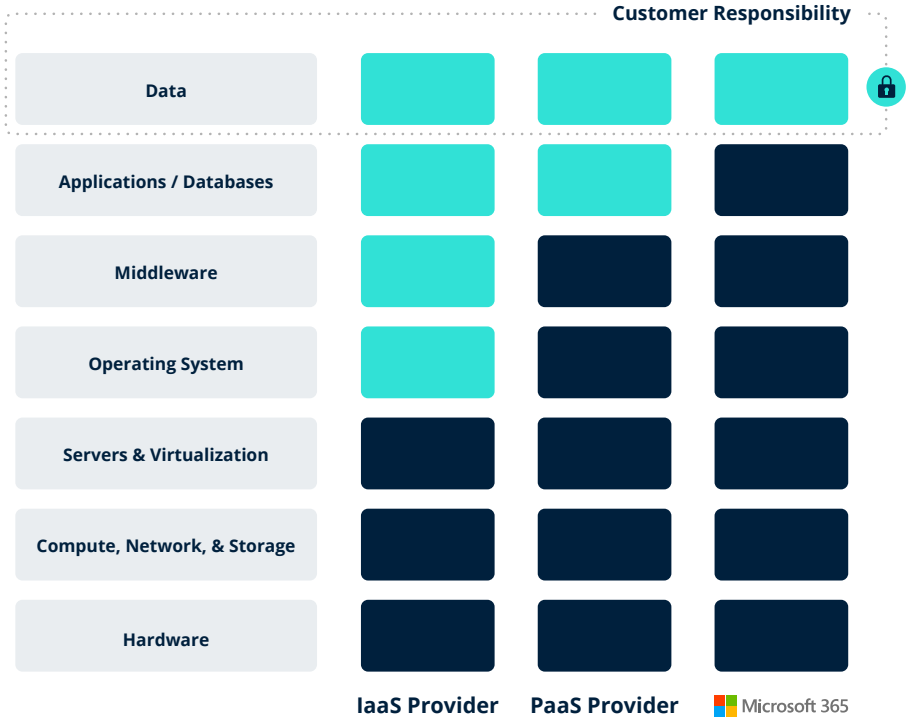
CONSIDERATIONS FOR MICROSOFT 365 PROTECTION

Enterprises are rapidly adopting Microsoft 365 services to leverage the agility of the cloud, reduce operational costs, and enable employee productivity. However, protecting this data presents numerous challenges, including operational limitations, slow recovery from data loss, and difficulty meeting compliance requirements.

MICROSOFT 365 SHARED RESPONSIBILITY MODEL

Organizations understand the value of backing up their data on-premises but often are not investing in protecting SaaS-based data in the same way. Many IT leaders assume that their cloud providers have "assured backup" in place — a dangerous assumption that can result in data loss. The cloud providers terms and conditions state that data protection in the cloud is a shared responsibility between you (the customer) and the cloud provider. The Microsoft Services Agreement takes this a step further and states that you should **"regularly backup Your Content and Data that you store on the Services or store using Third-Party Apps and Services."**

Ultimately it is the customer's responsibility to protect their applications and data running in a public cloud, regardless of provider. The shared responsibility in the cloud published by Microsoft is a great point of reference for these concepts. This leaves the customer at a critical decision point – How do I efficiently and reliably protect my assets that reside in the cloud? While the question seems simple on its face, it is in fact quite complex.



How do I efficiently and reliably protect my assets that reside in the cloud? While the question seems simple on its face, it is in fact quite complex.

ARCHITECTURE AND COMPONENTS

At the core of the Rubrik Polaris for Microsoft 365 solution is the belief that a data protection architecture should share similar principles to the product it's protecting. Which means that for Microsoft 365—the "world's productivity **cloud**"—a traditional on-premises design was a non-starter for Rubrik.

INSTALLATION

In today's Microsoft 365 data protection competitive landscape, it's common for vendors to still adopt the on-prem mindset. One of the biggest advantages for an on-prem architecture is a *perceived* simplicity of installation and management. In our experience, we found it surprisingly easy to install Microsoft 365 data protection solutions utilizing standard on-prem deployment models. Microsoft 365 is simply treated as any other data source to be configured in an existing platform or requires the downloading of an installation wizard which can be installed in virtual environments. Or even easier, these platforms have been lifted and shifted to the Azure Marketplace where one-click installation can be utilized. In most cases the installation takes just a few minutes.

Once you are past that Day 0—or usually in these models Hour 0—deployments, that simplicity begins to disappear. How do you scale these infrastructures? How are they load balanced? What does the upgrade process look like? How do you ensure the underlying infrastructure is on the latest security patches? What type of data protection do you need on the environments themselves (i.e. backup the backup software)? These are all situations Rubrik set out to automatically handle and create a *truly* simple installation and management process using **cloud-centric** architectures.

To accomplish this, Rubrik Microsoft 365 solution offers two deployment models. Both utilize the Azure Kubernetes Service (AKS), Azure blob storage, and various other Azure components that are outlined below. These two architectures are:

- Rubrik-hosted (Recommended) - infrastructure components live in a *Rubrik owned* Azure environment
- Customer-hosted - infrastructure components live in a *Customer owned* Azure environment

Similar to the on-prem models mentioned above, both Rubrik architectures are configurable in a few minutes. The high-level installation process for each is as follows:

Rubrik-hosted (Recommended):

1. Microsoft 365 - Authenticate with a user account that has been assigned the global administrator role for the protected Microsoft 365 subscription.

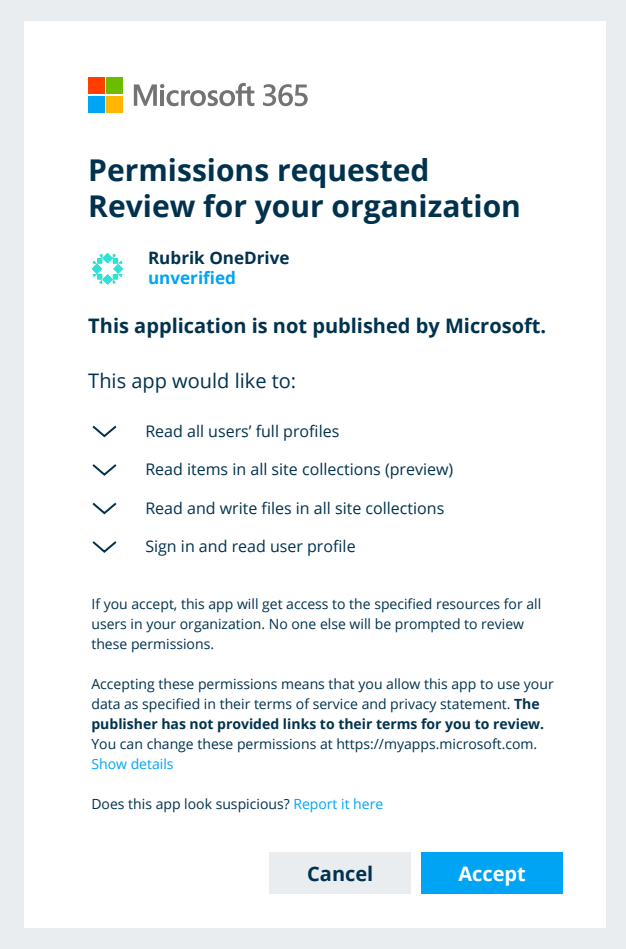
Customer-hosted:

1. Microsoft 365 - Authenticate with a user account that has been assigned the global administrator role for the protected *Microsoft 365 subscription*.
2. Microsoft Azure - Authenticate with a user account that has been assigned the global administrator role for the *Azure subscription*.

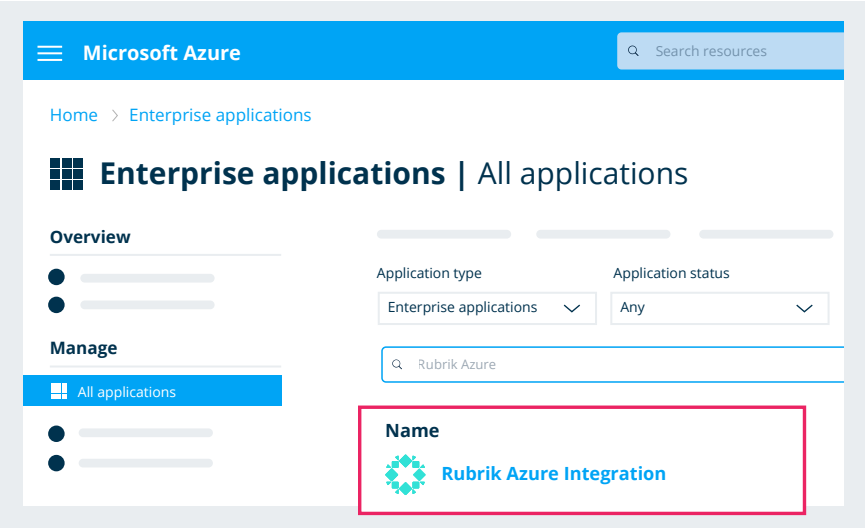
That's it. Rubrik will automatically handle the rest for you.

Each step above is centered around granting Rubrik access to your Microsoft 365 or Microsoft Azure environment through an authentication process. The legacy approach to authentication is Basic Authentication, which as of the writing of this white paper, is still commonly found as an option in Microsoft 365 protection solutions. "Basic Authentication makes it easier for attackers armed with today's tools and methods to capture users' credentials and increases the chance of credential re-use against other endpoints or services". So much so that as part of the Azure AD security defaults, Basic Authentication is disabled. Additionally, Microsoft has stopped supporting Basic Authentication which means any solution that relies on Basic Authentication will stop functioning when Microsoft fully deprecates their support. For these reasons, supporting Basic Authentication was never considered as an option for the Rubrik solution.

Instead, the Rubrik authentication process for both Microsoft 365 and Microsoft Azure utilizes OAuth 2.0, sometimes referred to as "Modern Authentication" by Microsoft, which is an industry standard protocol for authorization. OAuth allows you to grant Rubrik access to your environments through the use of a **one-time** access token rather than your admin credentials directly. Or in other words, Rubrik never has direct access to your username and password.

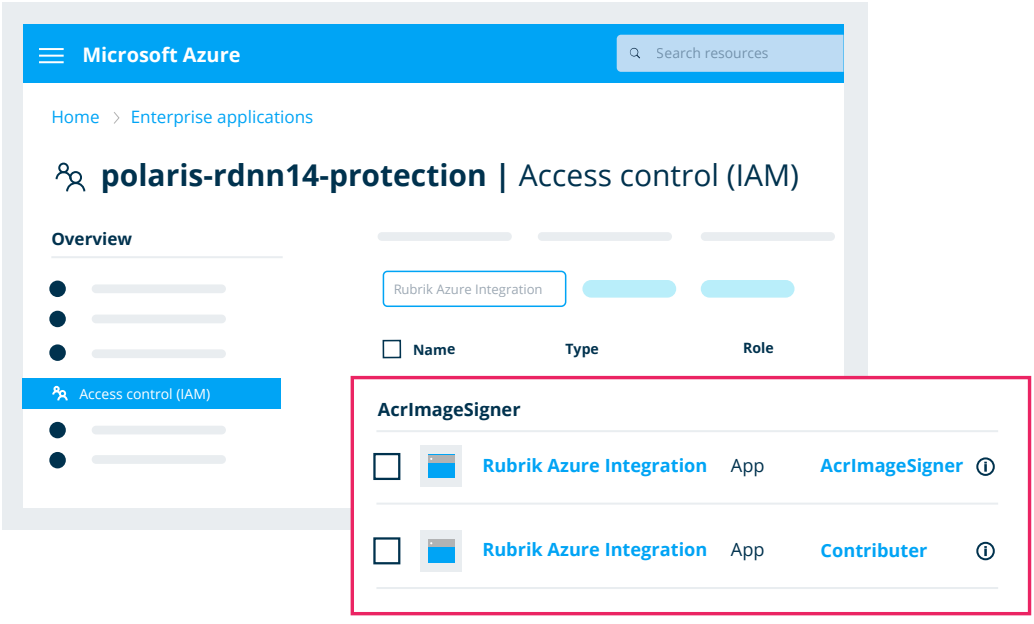


In the Customer-hosted model, after the infrastructure configuration process—which is covered below—has been completed, Rubrik automatically creates a "Rubrik Azure Integration" service principal—which is an identity created to control access to Azure resources—and then revokes the previously established OAuth access token. Moving forward, all access to a customer's Azure environment will be through this service principal account which can be found in the Azure Portal's Enterprise applications blade.



The Rubrik Azure Integration service principal access is scoped specifically to the customer defined Resource Group and is assigned to the following predefined roles:

- AcrImageSigner - grants the service principal the ability to verify the source of the container images used by AKS as well as ensuring it has not been modified after it was published.
- Contributor - "grants full access to manage all resources, but does not allow you to assign roles in Azure RBAC, manage assignments in Azure Blueprints, or share image galleries."



Similar to the "Rubrik Azure Integration" service principal, Rubrik will also create Microsoft 365 specific service principles for each of the Microsoft 365 apps supported by Rubrik. These service principles are used to grant access to the various Microsoft 365 APIs required for data protection and are used in both the Rubrik-hosted and Customer-hosted solutions. More information on these service principles, and their importance beyond simple authorization, can be found in the API Management section of this document.

After finishing the authentication process, there is no further configuration required with the Rubrik-hosted model. If using the Customer-hosted model, each Microsoft Azure component (compute, storage, networking, security, etc) required is automatically created by Rubrik. More importantly, Rubrik will also maintain these components moving forward, so there is no Day 2 management required. All the questions around scale, load balancing, upgrading, security patching, protecting the environment itself, etc. are no longer relevant. That's the difference between perceived simplicity and true simplicity.

MICROSOFT AZURE COMPONENTS

The architecture of both the Rubrik-hosted and Customer-hosted model share similar components. The main difference being the two is that the components in the Rubrik-hosted model are transparent and not accessible by customers.

This section provides additional details for each component and describes its role with the protection of Microsoft 365. These components are:

- **Resource Group -** A resource group is a container that holds related resources for an Azure solution. We recommend that the Resource Group used is dedicated to Rubrik so each of the requirement components are completely segmented from the rest of a customers Azure environment. This also simplifies the ability to view the Azure costs associated with each component.
- **Azure Kubernetes Service (AKS) -** orchestrates the provisioning and deprovisioning of containers used for Rubrik data processes. The number of nodes scales automatically based off of the workload demands.
- **Azure Container Registry -** an Azure container registry that is centrally managed by Rubrik. It contains the docker images utilized by the Azure Kubernetes Service.
- **Kubernetes Node -** the machine, in this case a Standard E2s v3 Azure Virtual Machine, where the Kubernetes Pods are run.
- **Kubernetes Pod -** a logical grouping of one or more containers
- **Rubrik Infra Node Resource Group -** The node Resource Group is a second Resource Group automatically created by AKS. It contains all the infrastructure resource associated with the AKS clutter including the node VMs, virtual networking, and storage. The Resource Group name will always being with "Rubrik-Infra".
- **Azure Virtual Network -** a `/16` network is provisioned by default. If an existing network is used, it must be `/20`, at a minimum. The large size of this network is related to the use of the Azure-CNI networking mode for AKS which is Microsoft's recommendation for production systems. The Azure-CNI networking mode assigns all AKS-managed containers their own IP from the subnet. To help with this, AKS reserves all the IPs it may need (110 per-node) in advance. By having a large network size, Rubrik is able to ensure adequate IP space as more Microsoft 365 apps are available for protection.

  While at first glance the size of the network may cause concern, it's important to remember that as a best practice the Resource Group where the Virtual Network was created should be dedicated to Rubrik and **will not have any connectivity or overlap to the rest of your network**. For the sack of clarity, we will repeat that last line. The Virtual Network does not have any connectivity or overlap to the rest of the network. Its size does not have any bearing to the rest of your environment.
- **Network Security Group -** the default Azure policy is used; the subnet allows outbound access but no inbound access. (i.e. no inbound access by default).
- **Azure Storage Account -** general purpose storage v2 for backup data. The Customer-hosted model utilizes locally-redundant storage (LRS), which copies your data three times within a single physical location in the primary region, and the Rubrik-hosted model utilizes zone-redundant storage (ZRS), which copies your data across three Azure availability zones in the primary region. This can be a new storage account or an existing storage account. It is recommended that

this storage account is dedicated to Microsoft 365 backup data which enables customers to secure and manage the storage account with controls specific to user data. The Azure Storage firewall managed feature will block all network traffic to and from the Storage Account except for traffic from the Polaris API address where the customer's Polaris account is located and the Azure Virtual Network for the Resource Group. If you manually created the storage account, this policy the Firewall will also need to be configured. The Rubrik-hosted model utilizes zone-redundant storage in lieu of

- **Key Vault -** stores keys/secrets used to encrypt data stored in the Azure Storage Account.

LOGICAL ARCHITECTURE (CUSTOMER-HOSTED)

Rubrik Polaris uses a connection to the customer's Azure subscription to provision storage and compute resources that run the containerized data management software and store the indexed data.

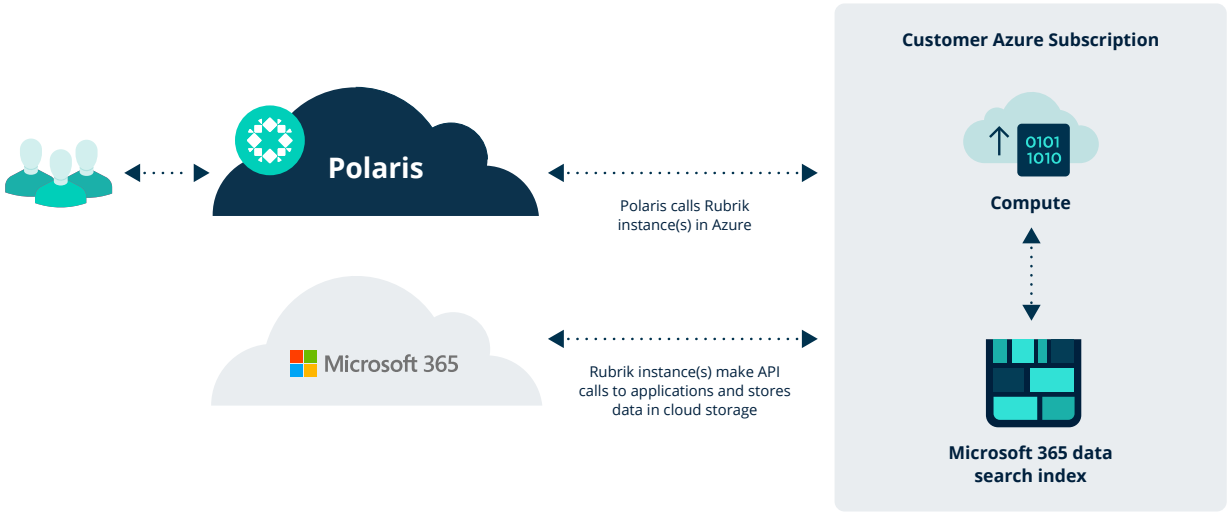


Figure 1 - Logical Solution Overview

In the Customer-hosted model, Polaris only stores metadata, such as the list of all users managed by the Microsoft 365 subscription. All data protection data and indexes are stored in the customer's Azure Storage Account.

In the customer's Azure subscription, Rubrik automatically provisions a Resource Group that contains a Virtual Network (vNet), Subnet, Network Security Group, Azure Kubernetes Service (AKS), Key Vault, and Storage Account. These resources are always available, and containers are dynamically spun up on-demand using AKS to perform backups, indexing, searching and restores. When creating the Azure Storage Account, Rubrik will also configure the Azure Storage Firewall to restrict access to the Polaris IP address where the customers Polaris account resides as well as the VNet that was created as part of the process. If you are using a manually created Storage Account it is highly recommended configuring a similar restriction. More information on this process can be found in the relevant Rubrik Support KB article.

The following diagram, *Figure 2*, depicts the logical architecture of the Microsoft 365 data management solution. Rubrik Polaris acts as the control plane to instruct AKS to provision resources on-demand to complete certain processes (e.g. backup, index, restore).

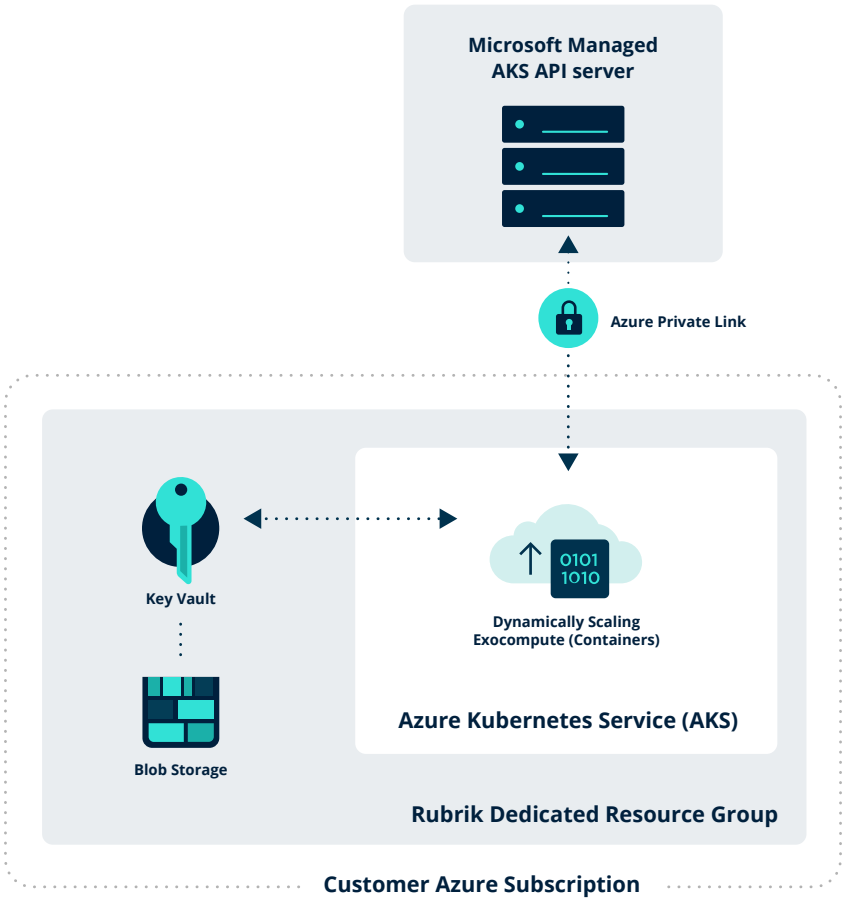
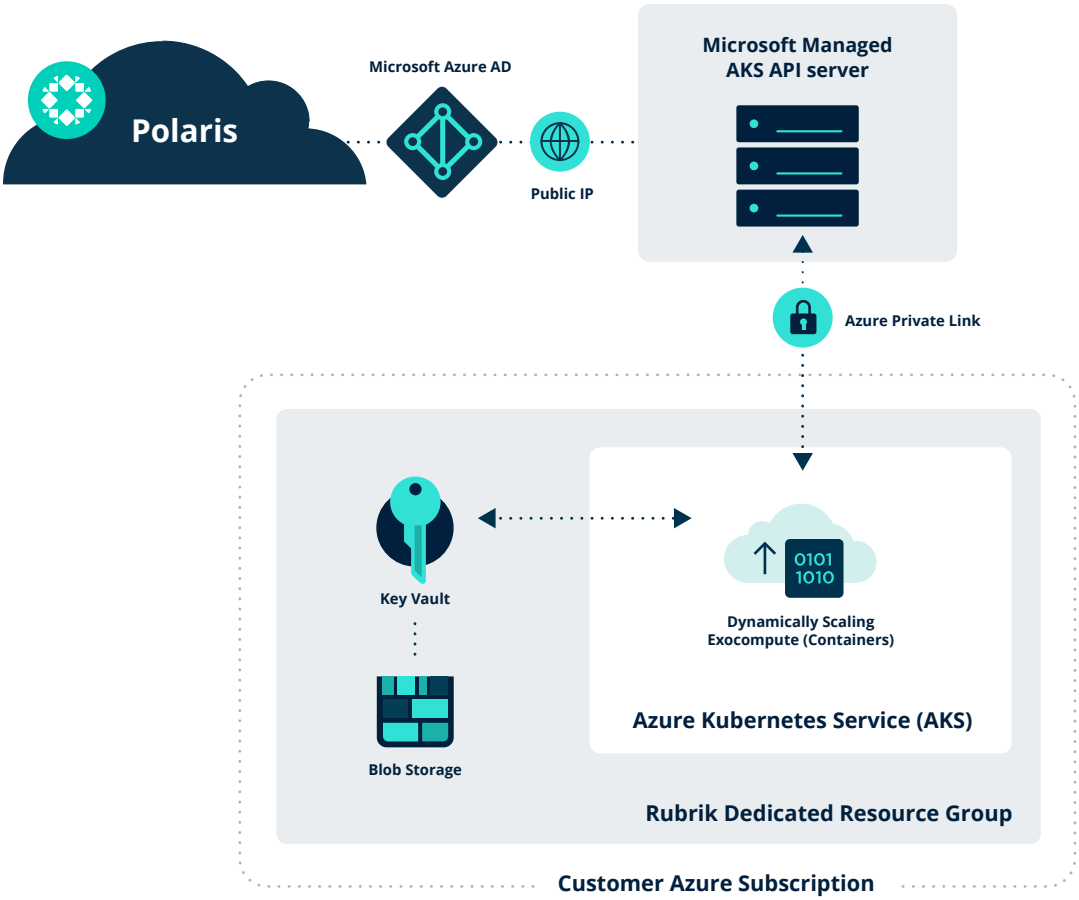


Figure 2 - Logical Architecture for Microsoft 365 Protection

Rubrik communicates to the customer's environment through the Azure-managed control plane of the AKS. More specifically, the *kube-apiserver* which provides a single connection point for Polaris to provision new compute containers in AKS. As an Azure-managed component, the API Server lives "outside" the customers Resource Group and has a public, internet facing address which is secured through Azure Active Directory and the Rubrik App Service Principal that was created during the initial configuration process. The API Server has direct connectivity to the AKS cluster through the use of an Azure Private Link which means no Firewall configurations in Azure are needed.



Also included in the Azure-managed control plane is the *kube-scheduler* which controls the number of AKS Nodes running in AKS as well as which Node each Pod should be running on. Using a custom defined schedule policy, the number of Nodes in AKS are scaled up and down as needed to meet the demands of the processes orchestrated by Polaris.

The below diagram shows an example of how the scaling can work over a week period of time. At peak usage, AKS utilized 26 nodes and then scaled down to a single node as demand decreased. Individual customer usage will vary depending on SLA Domain schedule and number of objects being protected.

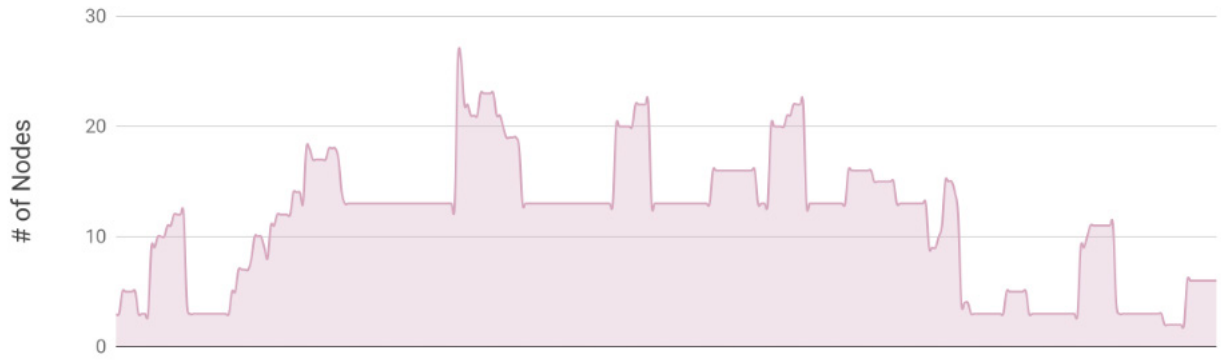
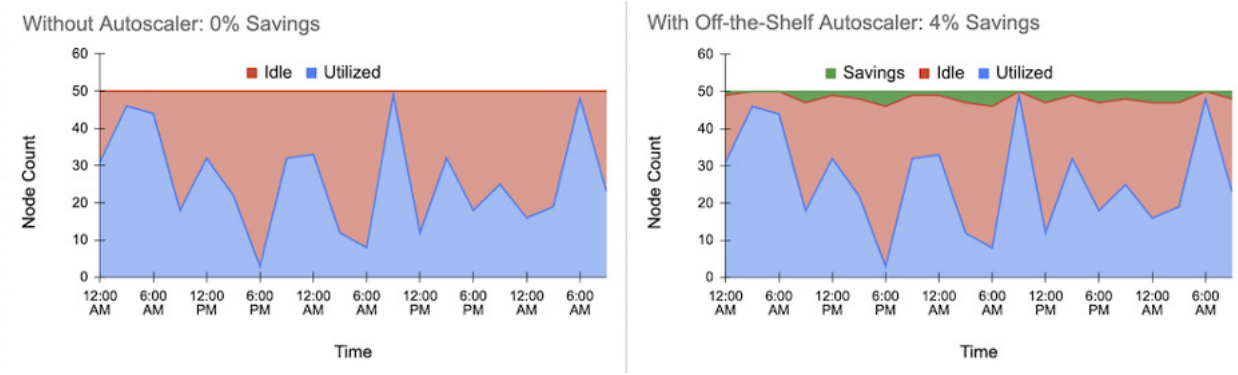


Figure 3 - Dynamically scaling Nodes over a week period of time

Autoscaling of the infrastructure is a core tenant of a Kubernetes cluster, but it turns out to be more complex than enabling the cluster autoscaler. To start we first had to evaluate if creating our own autoscaler was worth the effort. The final design of the homegrown solution was very similar to the off-the-shelf autoscaler logic and maintaining our own system in the fast evolving

cloud environment seemed to add a lot of unnecessary overhead. That made the decision to utilize the standard cluster autoscaler an easy decision.

After successfully deploying the autoscaler, we evaluated the initial cost savings. Below is the exemplary load pattern utilizing up to 50 nodes. The blue area is the amount of required (utilized) resources, and the red area represents the idle resources that should be minimized. The green area displays the savings from enabling the autoscaler.



We obtained about 4% cost savings. A good start but there was still room for improvement! What prevented the autoscaler from actually reducing the idle resources further? Let's take a closer look at the scaling logic of the cluster autoscaler.

The autoscaler will monitor the pod queue of the Kubernetes cluster at a constant frequency and switches between scaling up and scaling down mode. These are the main scaling criteria:

- Scale up if there are pods waiting in the pod queue. Scale up by enough nodes to ensure all pending pods can run in parallel. The node estimator of the autoscaler uses a bin packing approach to determine the required number of nodes.
- Scale down if a node is completely unused or only runs pods that are safe to evict for a specified period of time.

Several parameters allowed us to customize the scaling logic of the Kubernetes autoscaler further. The optimal parameter settings for the autoscaler logic depend, of course, on the load pattern of the cluster.

We set the minimum node count min-nodes to 1 to maximize cost savings during zero load times. The maximum node count max-nodes should be the maximum number of nodes needed in the worst-case scenario. For our load case, we chose 50 nodes. We also found it beneficial to increase the parameter scale-down-unneeded-time to 30 minutes. Our load pattern included large bursts at a 20-minute frequency, and scaling up and down in between these bursts was adding unnecessary noise to the AKS without actually saving us much cost. Setting the parameter scale-down-delay-after-add to 1 minute allowed the autoscaler to almost immediately consider scaling a node down again when added to only handle a short burst load. Finally, changing the scaling frequency scan-interval from 10 to 30 seconds reduced the rate of API calls from the autoscaler pod to the AKS API to avoid throttling issues.

Customizing these parameters of the scaling logic already significantly improved the autoscaler performance. But there was one more challenge for us to overcome before we could achieve our target cost savings. By only customizing the autoscaler, we ended up with a very low node utilization. The application pods were distributed over all available nodes, preventing the nodes from being scaled down by the autoscaler. Using a customized scheduler solved this issue.

To reduce the compute resources and thereby the cloud cost, we mainly relied on the scale down logic to remove nodes from the cluster as fast as possible. The scale down criteria was stated as:

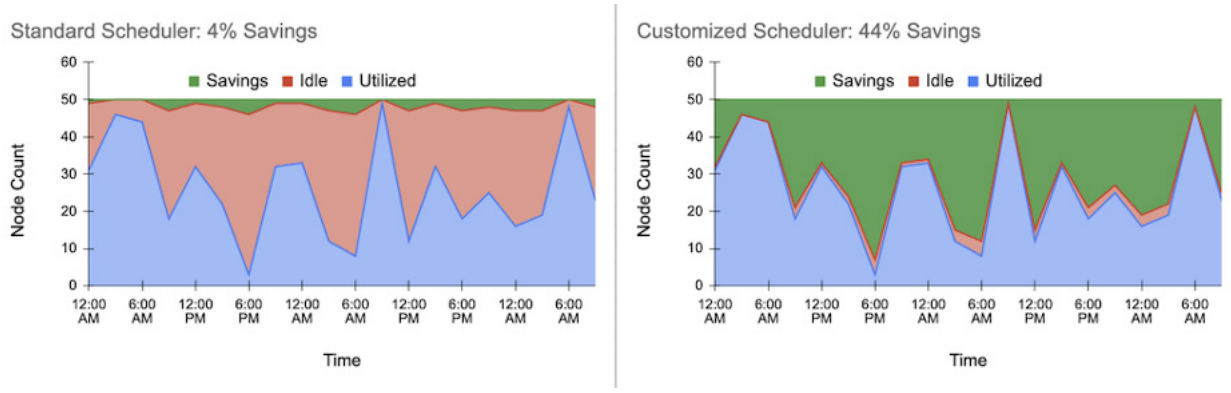
- Scale down if a node is running only pods that are safe to evict for a certain period of time.

Pods can be safe to evict because of their priority (see Pod Priorities and Preemption) or because they are backed up by a controller (e.g. a replica set) or if they have a non-restrictive PodDisruptionBudget and don't fall into any other pod categories that would prevent the autoscaler from scaling down the underlying node. The autoscaler can then consider deleting such "safe-to-evict" pods and redeploying them on a different node to optimize the bin packing problem and to remove nodes with low utilization from the cluster. So, if your application only uses pods that are safe to evict, there is no need for customized scheduling.

But what happens if an application uses stateful pods that are not safe to evict? These pods cannot be moved to a different node and as soon as one of these pods runs on a node, the node is assumed to be "needed." The node is blocked for scaling down, and the countdown for the scale-down-unneeded-time is set back to the initial 30 minutes. Unfortunately, the default Kubernetes scheduler optimizes for load balancing and thereby distributes the pods across nodes as evenly as possible. In the worst case, this could lead to each node running only one pod at a time, blocking the autoscaler completely from scaling down.

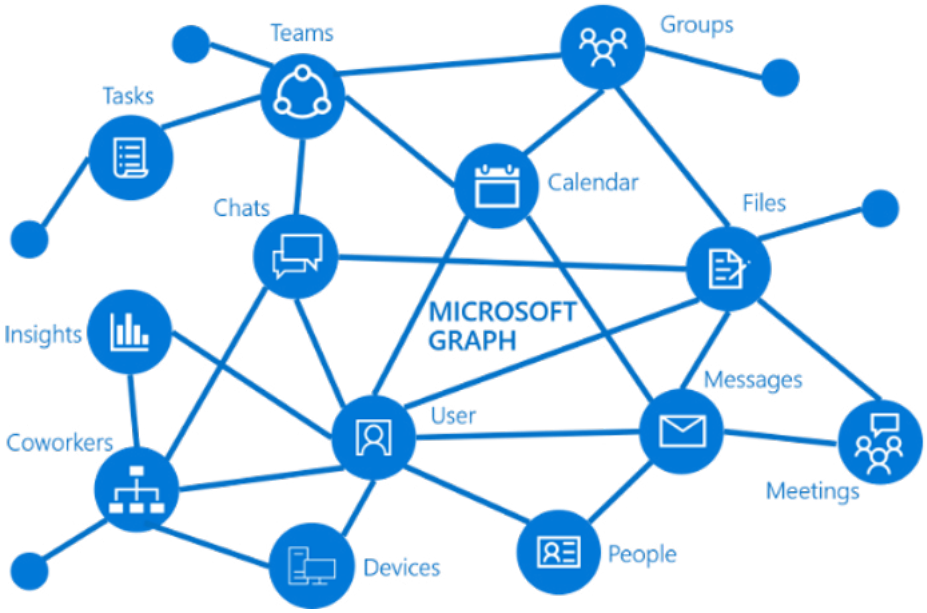
So for our application, instead of evenly distributing the pods across nodes, we needed to pack the nodes as much as possible by scheduling pods on nodes that are already used the most. Thereby we minimized the number of utilized nodes and let the autoscaler remove the unneeded nodes. To pack the nodes as much as possible we decided to use a customized Kubernetes scheduler. A customized scheduler can change the logic of assigning pods to nodes (Scheduling Policies) by setting specific flags during its deployment. If the weight of the scheduling priority flag MostRequestedPriority is set to 100 using a ConfigMap, pods get scheduled on the node that is already used the most. The customized scheduler can be deployed to a Kubernetes cluster as an additional pod alongside the default scheduler. The pods of an application then specify which scheduler to use.

So after deploying the customized scheduler to our Kubernetes cluster, we looked again at the cost savings. For our application, the customization of the autoscaler and the scheduler was the big win - we observed cost savings above 40%!



MICROSOFT 365 API MANAGEMENT

Rubrik communicates with Microsoft 365, via the AKS cluster, through multiple Microsoft APIs. The recommended Microsoft 365 API is known as Microsoft Graph. It provides a single endpoint (https://graph.microsoft.com) that allows Rubrik to connect to each of the underlying Microsoft 365 apps.



While Microsoft Graph exposes a wealth of data, it also has its shortcomings. Which is why Rubrik utilizes a "hybrid" API approach that also encompasses the more traditional APIs specific to each Microsoft 365 product. This approach allows us to automatically take advantage of each APIs strong points with the goal of creating the most performant and reliable experience possible. This also includes automatic fall back between APIs in case of failure scenarios outside the control of Rubrik. For example if Microsoft Graph fails we have the ability to switch to the Exchange Web Services API to complete the task without any failures exposed in the data protection process.

In the context of the Microsoft 365 APIs one of the most important topics to discuss is throttling. As globally available resources, Microsoft has a duty to ensure each API is consistently available for each of their customers. One way they ensure this is through throttling. After a certain number of concurrent calls Microsoft will automatically return an HTTP 429 status code (Too many requests) and fail the request along with a suggested wait time to provide guidance on when the throttling will be removed.

Along with batching multiple queries into a single request, Rubrik will also dynamically load-balance the API calls through multiple Microsoft 365 service principles (also known as apps). Each of these service principles are specific to the individual Microsoft 365 products (Exchange Mailbox, OneDrive, SharePoint, Teams, etc.) and have different recommendations for optimal backup and recovery speeds. The **most important** thing you can do to ensure optimal backup and recovery speeds is to create the recommended number of Enterprise Applications per product. Due to Microsoft security restrictions, each app will need to be created manually. Thankfully this is still an extremely quick process and will only take a few seconds per app. At a minimum we recommend two apps for each product. In additional to those two apps we recommend the following:

- Exchange Mailbox - One additional app for every 5000 mailboxes
- OneDrive - One additional app for every 1500 users
- SharePoint - One additional app for every 1000 Document Libraries
- Teams - No additional apps are required

CONCLUSION

This document provided a detailed understanding as to the benefits of Rubrik's Microsoft 365 solution, how it works underneath the hood, and gave insights into how the data is maintained and secured.

VERSION HISTORY

Version	Date	Summary of Changes
1.0	May 2021	Initial Release
1.1	June 2021	Minor grammar fixes
1.2	June 2021	Minor update for Teams support

rubrik

Global HQ
1001 Page Mill Rd., Building 2
Palo Alto, CA 94304
United States

1-844-4RUBRIK
inquiries@rubrik.com
www.rubrik.com

20210701_v1