CHAPTER 5

# Securing and Monitoring Applications Running on AKS

## Introduction

First, congratulations on completing 60 percent of this book. After reading about architecture designs and patterns, in this chapter, you'll go through another interesting and important aspect—monitoring and securing Azure Kubernetes service-based applications. You must be wondering why I didn't mention microservices. The answer is simple—when I say applications, that covers it all.

Digital platform initiatives are making organizations embrace the cloud culture. Cloud enablement includes building strategies to deliver more value to their clients. One of these strategies focuses on application security. With the growing pace of developments, it's important to plan the security of the applications to avoid any impediments in business operations.

Here, I present you with the security concepts for applications and clusters, concluding with one of the best reckoners for applications with respect to security.




K. Ahmed Shaikh and S. S. Agaskar, *Azure Kubernetes Services with Microservices*,
https://doi.org/10.1007/978-1-4842-7809-3_5

# Security Concepts

Kubernetes and Microsoft Azure both include their respective security components. The Azure Kubernetes Service combines these security components to:

- Make sure your AKS Cluster is running the latest Kubernetes releases
- Ensure its up-to-date with OS security updates
- Secure Pod traffic
- Provide trusted access to sensitive credentials

Let's dive into a few core concepts for securing application and clusters in the Azure Kubernetes Service.

## Master Security

In AKS, each cluster has a dedicated Kubernetes master-enabling API server, scheduler, and so on. This Kubernetes API sever uses an FQDN—a fully qualified domain name—along with a public IP address. As a PaaS service, Kubernetes Master components are included in managed services and are maintained by Microsoft.

You can adopt them in the following ways:

- You can create a private cluster, limiting server access to dedicated virtual networks.
- Using authorized IP ranges, you can restrict access to API server endpoints. Refer to Figure 5-1.

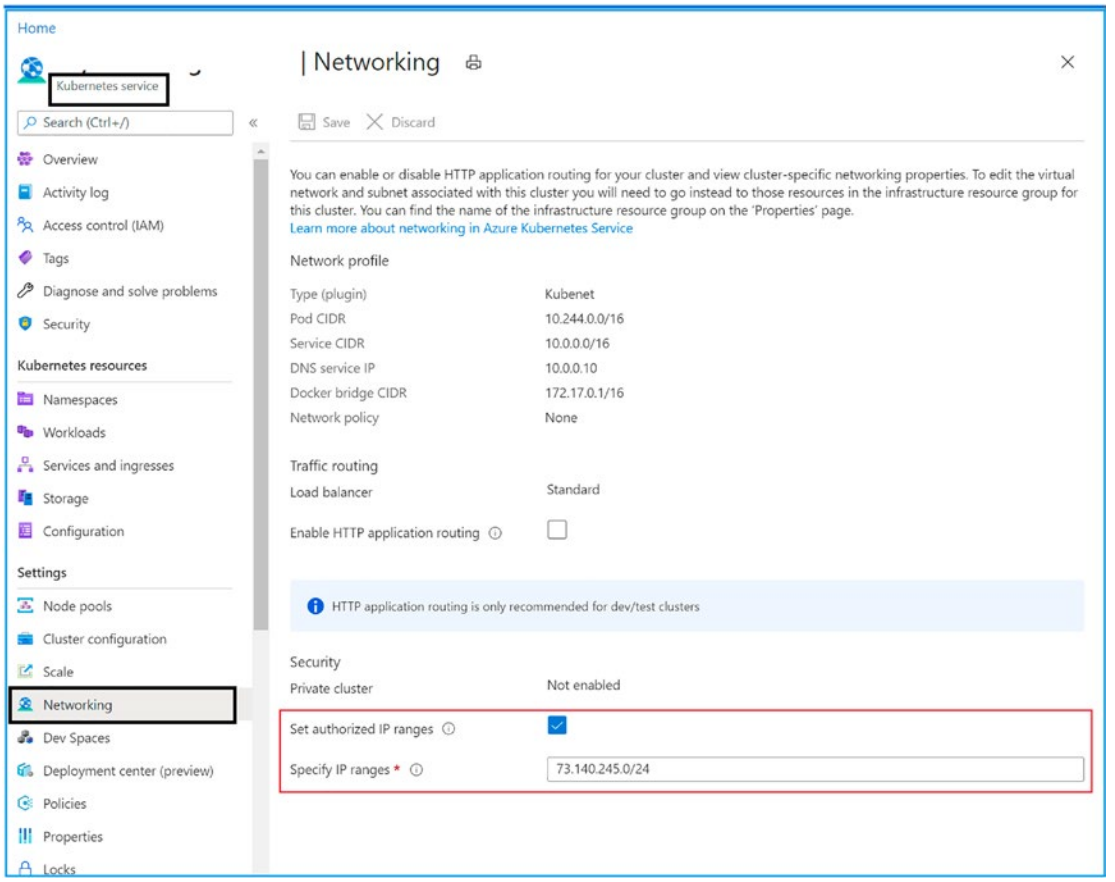


***Figure 5-1.*** *Authorized IP range feature in Azure PortalImage source: Microsoft Documentation*

Also, you can control access by using Kubernetes RBAC and Azure RBAC. See the implementation details at `https://docs.microsoft.com/en-us/azure/aks/managed-aad`.

## Node Security

Azure Kubernetes service nodes are nothing but the virtual machines that you manage. Linux and Windows server nodes both run an optimized Ubuntu distribution and Windows Server 2019 release respectively, using the Docker container runtime.

The latest OS security updates and configuration are deployed automatically on to nodes, whenever an AKS Cluster is created and also when it's scaled up.

Security patches for Linux nodes and Windows server nodes can be achieved by running simple azure CLI commands. For better understanding, the following Azure CLI command will upgrade the node pool called `myaksbooknodepool`:

```
az aks nodepool upgrade
    --resource-group myAKSBookResourceGroup \
    --cluster-name myAKSBookCluster \
    --name myaksbooknodepool \
    --kubernetes-version KUBERNETES_VERSION \
    --no-wait
```

This is just an example of one of the various simple Azure CLI commands available for managing nodes.

Nodes are always deployed to private virtual networks. Even the storage used by nodes are premium, Azure managed disks, backed by SSDs. Within the Azure platform, the data on this disk is always encrypted at rest. Azure also provides the option for isolated VMs, required as a part of compliance and regulatory requirements. This applies to Linux and Windows virtual machines. At the time of writing this book, there are a few options available for isolated VMs:

- `Standard_E80ids_v4`
- `Standard_E80is_v4`
- `Standard_F72s_v2`
- `Standard_M128ms`
- `Standard_DC8_v2`

## Cluster Upgrades

An Upgrade Orchestration tool is provided by Azure, which includes the Kubernetes Master components and the Agent component. This tool enables the following actions:

- Upgrades of AKS Clusters and their components
- Security maintenance
- Compliance maintenance
- Access to the latest features

All it needs is the available Kubernetes version and the rest is taken care of, of course, with fewer commands.

## Network Security

To communicate with your on-premises networks, you can have your AKS deployed to Azure virtual network subnets. Using Site-to-Site VPN or Express Route, this virtual network connects to your on-premises network. You can also define Kubernetes ingress controllers with private, internal IP addresses to limit services access to network connections. You can leverage Azure network security groups and even Kubernetes network policy to control the traffic flow.

Azure Kubernetes Service offers support to Kubernetes network polices, to limit network traffic between Pods in the given cluster based on namespaces, label selectors, and so on.

## Kubernetes Secrets

With a Kubernetes Secret, you can add your sensitive data, such as access credentials and keys—to Pods. Use of Secrets minimizes the use of this sensitive information in (Pod or service YAML) manifests. You can request the secret as part of your manifest, limiting access to the information to specific Pods only.

Kubernetes secrets are stored in `Etcd`, a distributed key-value store. The `Etcd` store is fully managed by AKS and data is encrypted at rest within the Azure platform. See Figure 5-2.

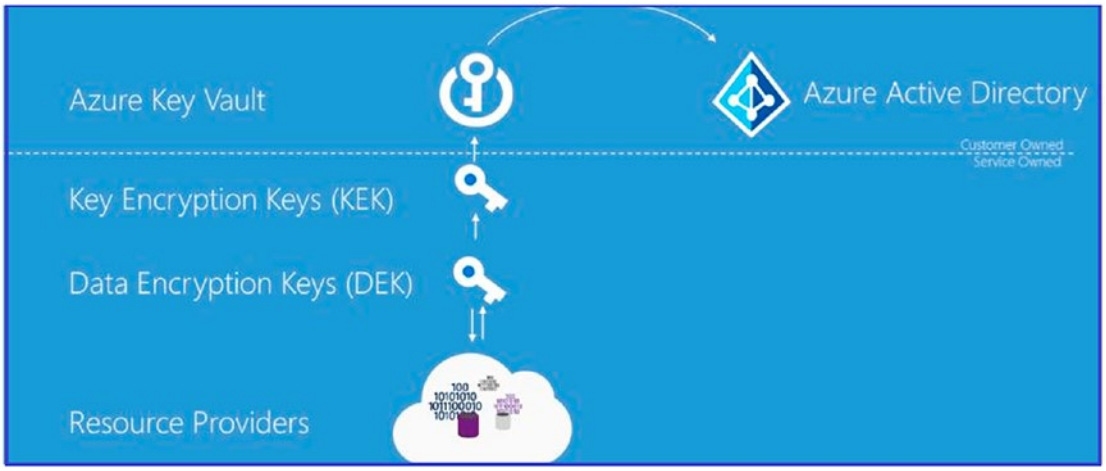


***Figure 5-2.*** *Azure encryption at rest componentsImage source: Microsoft Documentation*

There are a few points you need to know about the Kubernetes Secrets:

- You can create a secret using the Kubernetes API.
- You define and request the secret for your Pod.
- Secrets are not written on disk, but are stored in `tmpfs`.
- When the Pod requiring the Secret is deleted, the secret is also deleted from the `tmpfs` nodes.
- Secrets are stored in a namespace, making them accessible only to the Pods within the same namespace.

Apart from these security concepts with respect to application and clusters, it's better to also go through the following:

- Azure security baseline for Azure Kubernetes Service

  (See `https://docs.microsoft.com/en-us/security/benchmark/azure/baselines/aks-security-baseline?context=/azure/aks/context/aks-context`.)

- Azure Policy Regulatory Compliance controls for Azure Kubernetes Service (AKS)

  (See `https://docs.microsoft.com/en-us/azure/aks/security-controls-policy`.)

# Azure Kubernetes Service Checklist

This checklist contains some of the best practices to follow while working with AKS. It's not a kind of bible to follow, but simply best practices to adhere to when it comes to security

- **Refrain from injecting sensitive information into images and use Secrets instead**. As mentioned, avoid entering sensitive information like passwords directly into images or the manifest. Rather, always use Secrets—either Kubernetes Secrets or Azure Key Vaults—for such information. See Figure 5-3.

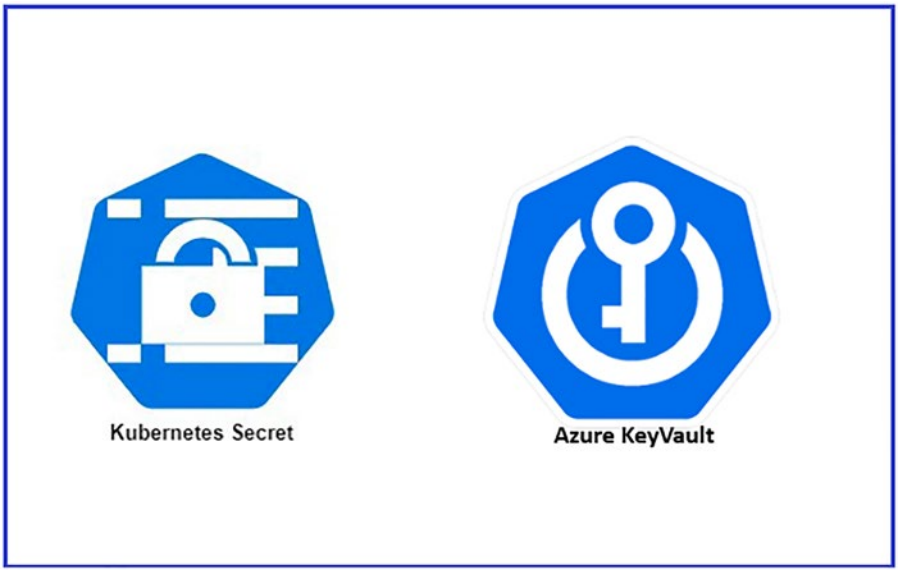


***Figure 5-3.*** *Kubernetes Secrets and Azure KeyVault icons*

- **Implement Pod identity**. Don't have fixed credentials stored in Pod images. Rather you can use Pod identities, which use the Azure Identity solution for all the access to the desired (Azure) resources. These credentials could be any credentials used to talk with other Azure services, like Azure SQL or Azure Storage. You can define them in Kubernetes Secrets, but it needs a manual management. Here you can miss the best practice of rotating the Secrets being used.

Pod-managed identities for Azure resources can be used to have the access request via Azure AD. Figure 5-4 depicts the request flow.

---

**Note** At the time of writing this book, Pod-managed identities for AKS is in preview.

---

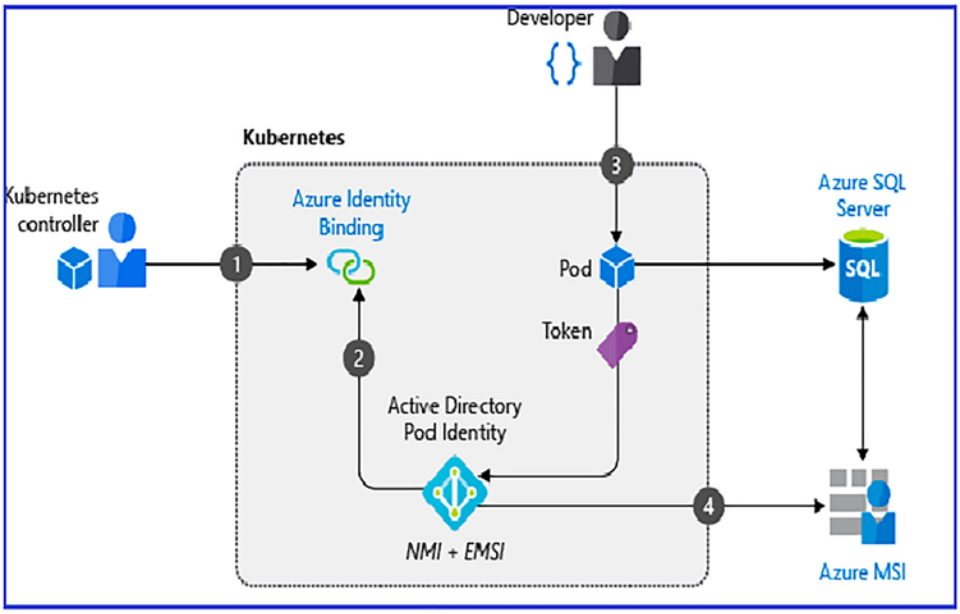


***Figure 5-4.*** *A developer created a Pod that uses a managed identity to request access to an Azure SQL databaseImage source: Microsoft Documentation*

- **Use a Kubernetes Namespace**. Namespaces are the logical partitions of your resources. They not only enforce the separation of resources but also limit the permissible user scope. For simpler understanding, you can have different namespaces for different business units or groups. You should use the Kubernetes Namespace to isolate your Kubernetes resources. See Figure 5-5.

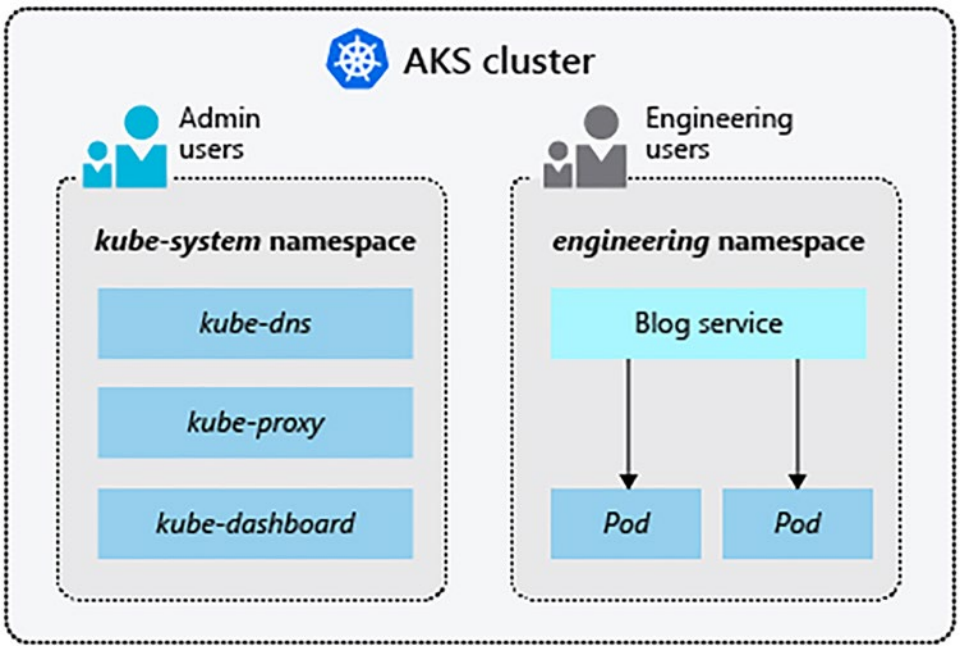


***Figure 5-5.*** *Example of Namespaces in an AKS ClusterImage source: Microsoft Documentation*

The following namespaces are available when you create an AKS Cluster:

- `Default`: When no namespace is given, this is where Pods are created.
- `kube-system`: Where core resources exist.
- `kube-public`: These resources can be viewed by any user.

---

**Important** Avoid using the default namespace.

---

- **Specifying the correct security context for a Pod**. This is an important factor in deciding your Pod access control settings. If the context is not set, the Pod gets the default one, which exposes it with more rights.
- **Manifest with best practices**. Ensure that the configuration of the manifest follows the best practices. A good manifest presents a good cluster. ☺

- **Static analysis of images on the build**. Introduce DevSecOps into the environment to promote a proactive security model that starts to shift the responsibility left. Azure Defender for container registries can be used here.
- **Enforcement of compliance on the build image**. You must go through Azure Policy built-in definitions for Azure Kubernetes Service; see https://docs.microsoft.com/en-us/azure/aks/policy-reference?ref=akschecklist.

---

**Note** Bookmark the following URL: https://www.the-aks-checklist.com/. The seven commandments explained previously are the part of this forum, and they keep changing based on new context.

---

## Security Concepts: Conclusion

AKS security aspects could be covered in an entire book of their own. However, considering the scope of this chapter, I tried to sum up things you should know when working with AKS-based applications. Consider these as answers you would expect from an interviewer panel, when discussing AKS-based application security.

The next section covers another important area, monitoring.

# Monitoring Concepts

Your applications are secured but need to be up and available all the time. Securing applications that have no availability is useless. Hence, monitoring is an important area for you to be cognizant of. This section presents you with the details of monitoring AKS-based applications along with monitoring the AKS using Azure Monitor.

## Container Insights

Resources that generate performance metrics and resource logs can be monitored for health and performance. Like other Azure resources, the Azure Kubernetes Service also has logs.

Azure Monitor has a feature called Container Insights that runs these checks for managed Kubernetes hosted with AKS. What makes it a favorite is its ability to present interactive views on data coming from different monitoring scenarios. It's natively integrated with Azure Kubernetes Service, helping to collect critical logs, send alerts, and visualize. Figure 5-6 shows Container Insights in Azure Portal.

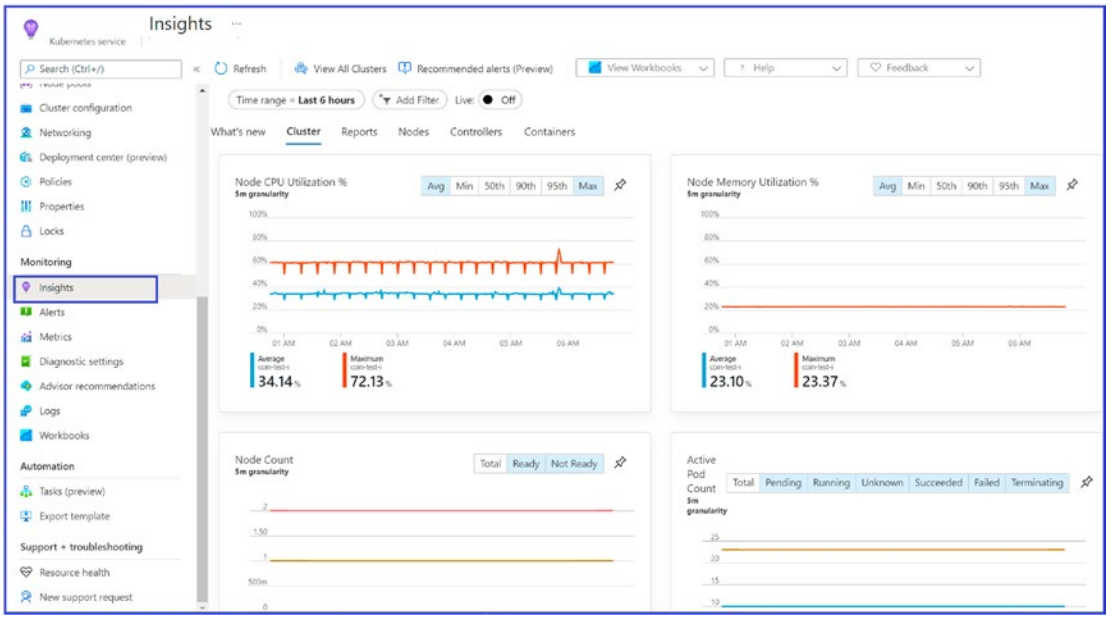


***Figure 5-6.*** *Features of Container insightsImage source: Microsoft Documentation*

Here are the steps required to configure monitoring with Azure Monitor for an AKS Cluster:

1. Create a log analytics workspace. You must have one log analytics workplace to have telemetry data collected from the AKS Cluster. Container Insights requires at least one log analytics workplace.

2. Enable Container Insights. Enabling Container Insights depends on the AKS Cluster you are working on. Does the AKS Cluster already exist or is it a newly created one? Once enabled, a containerized version of the Log Analytics agent is deployed, which sends data to Azure Monitor.

3. Configure a collection from Prometheus. Using Container Insights allows you to collect Prometheus metrics without requiring a Prometheus server. This makes the combination successful for E2E monitoring. Refer to Figure 5-7, which depicts the working of Container Insights and Prometheus.

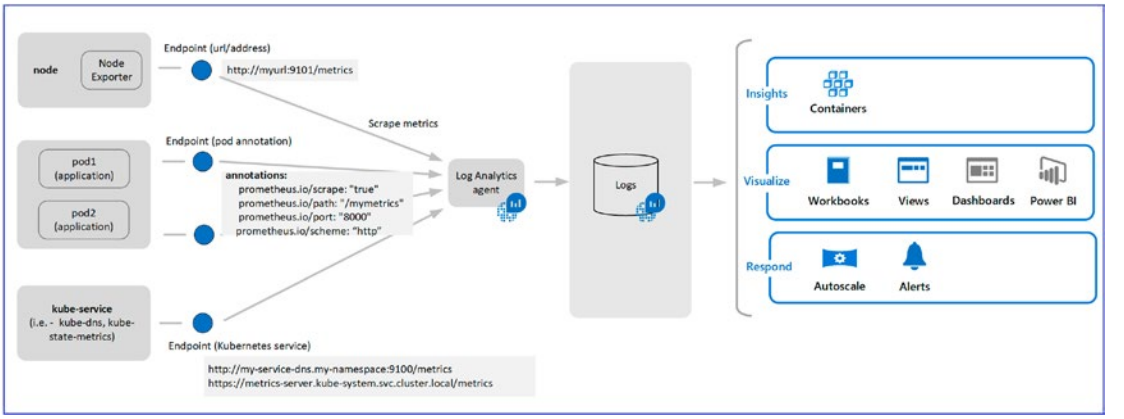


***Figure 5-7.*** *Configure scraping of Prometheus metrics with Container InsightsImage source: Microsoft Documentation*

4. Collect the resource log. These are the logs for the AKS control plane components implemented in Azure. You need to have a diagnostic setting to collect these logs. You can have it in the log analytics workspace or in Azure storage. Figure 5-8 shows the screen for configuring diagnostics settings.

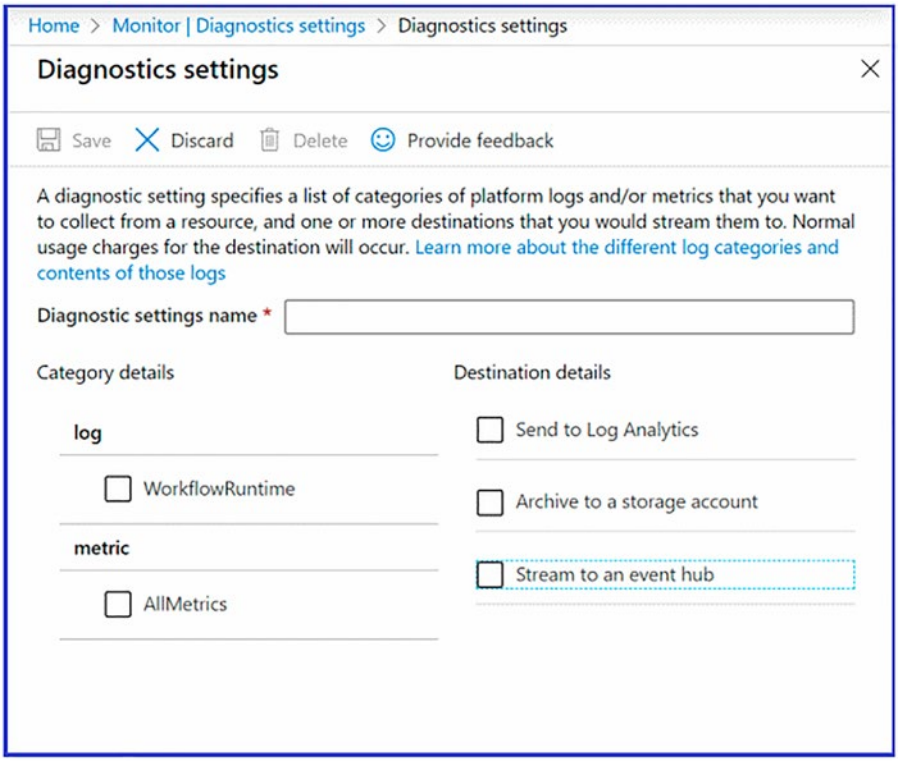


***Figure 5-8.*** *Screen for configuring diagnostics settings in Azure Portal*

## Azure Monitor Features

There are two ways you can view the Azure Monitor features for AKS Clusters:

- Go to Azure Portal ➤ Kubernetes Service and choose the Monitor section from the left pane. This is mostly for single AKS Clusters. Refer to Figure 5-9.

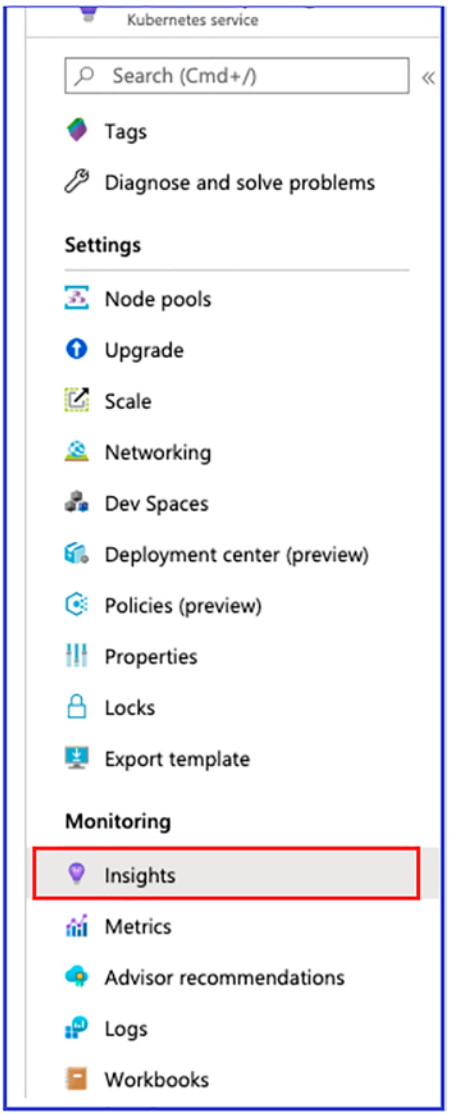


***Figure 5-9.*** *Insights option for the Kubernetes service screen in Azure Portal*

- Go to the Azure Portal. Type **Monitor** into the search box and then choose Insights Section ➤ Containers.

Figure 5-10 shows all the AKS Clusters in a subscription.

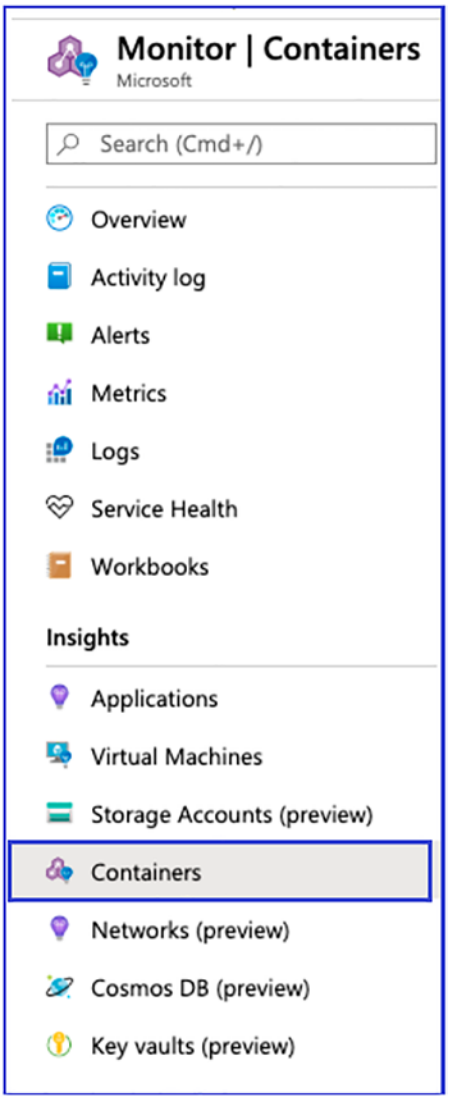


***Figure 5-10.*** *Pane for Container Insights from the Monitor screen in Azure Portal*

Figure 5-11 is a fresh screen view for Monitor in Azure Portal. It is an entry point for monitoring clusters, creating alerts, and many more cool implementations.

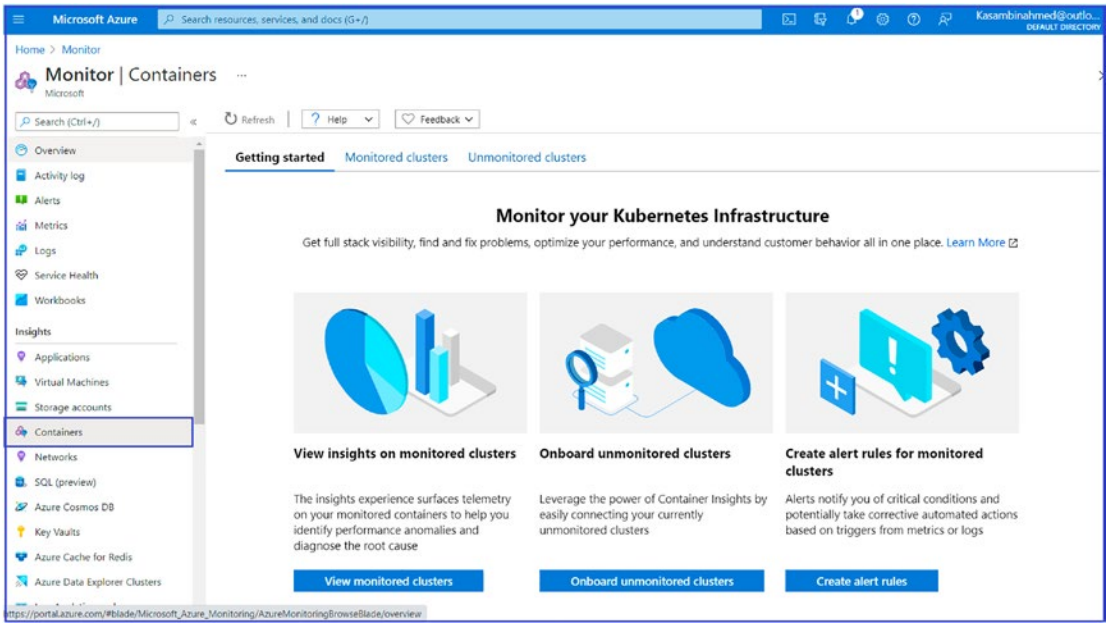


***Figure 5-11.*** *Containers fresh screen under Monitor in Azure Portal*

Azure Kubernetes Service monitoring comes with variance implementation and unique requirements. This approach relies on these requirements. It deals with different layers, from infrastructure to application, and comes with distinct monitoring requirements based on the layer.

Considering a bottom-up approach, these layers can be listed as shown in Figure 5-12.

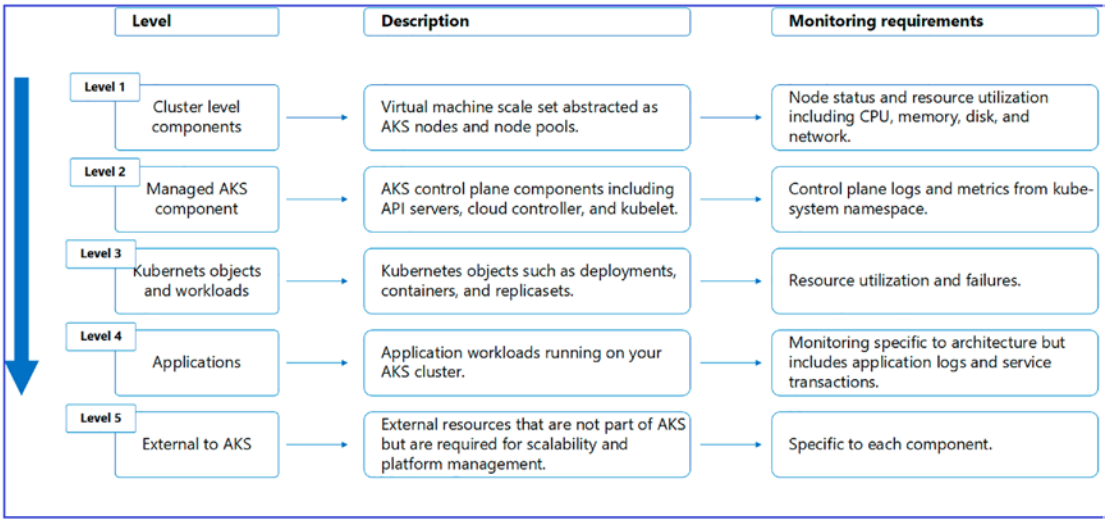


***Figure 5-12.*** *Layers of AKSImage source: Microsoft Documentation*

To maintain the scope and brevity of this chapter, I will discuss Level 4, the monitoring application layer that includes application workloads running in an AKS Cluster.

This layer mainly focuses on monitoring the microservices application and identifying application failures, along with information like request rates, response time, any exceptions encountered, and so on. For complete monitoring of applications running on AKS, you can use Application Insights, as shown in Figure 5-13.

***Figure 5-13.*** *Icon for Application Insights*

Depending on your application stack, you need to configure code-based monitoring to collect the required data. It could be anything—Java, Python, .Net, or any other platform. In this example, I am more interested in ASP.NET Core applications.

You need to have a valid Application Insights instrumentation key and create the Application Insight resource.

To create one, go to Azure Subscription. Type **Application Insights** in the search and click Create New. You will be presented with the screen to add the basic mandatory details required to create the service. Enter with all the details and click Review + Create. Refer to Figure 5-14.

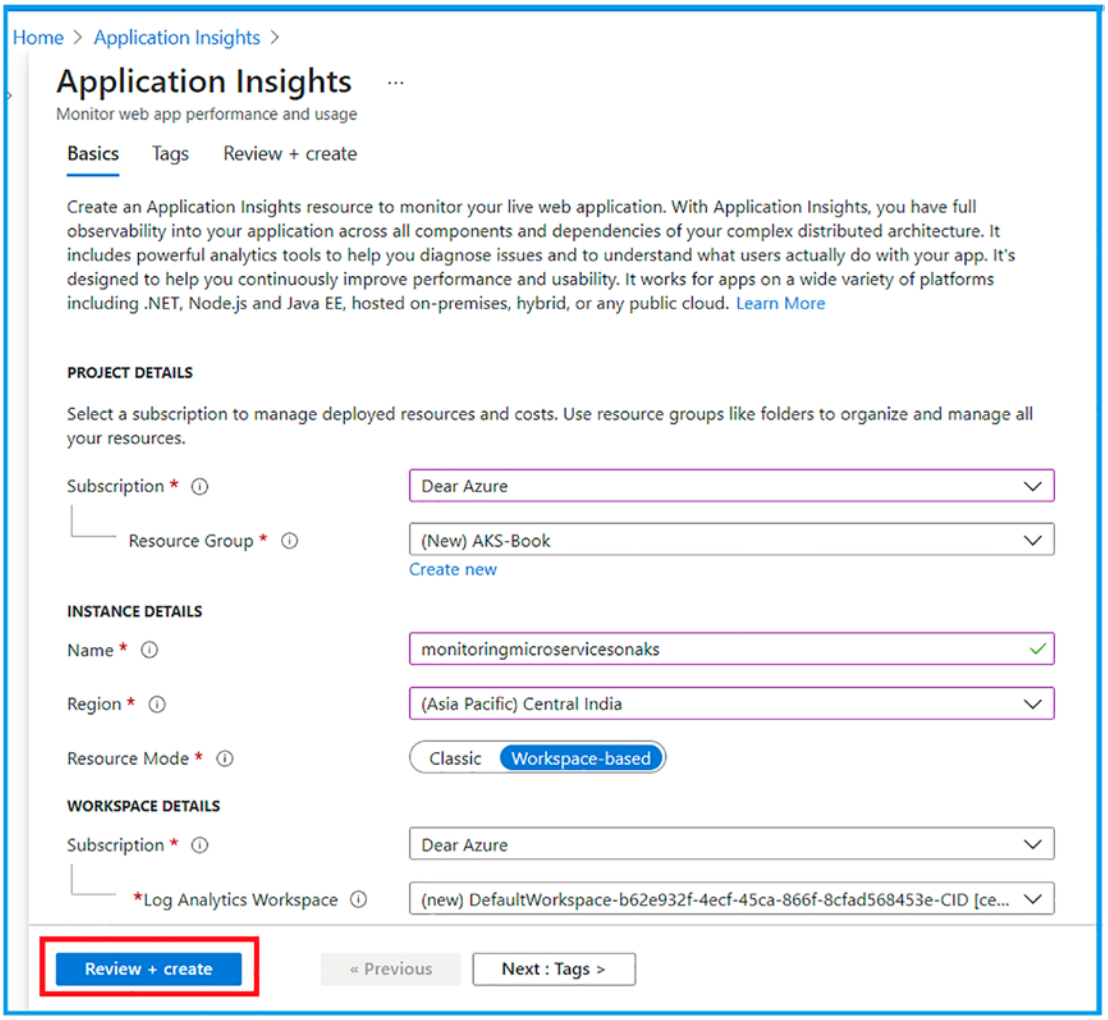


***Figure 5-14.*** *Screen to create Application Insights in Azure Portal*

Once it's validated, click Create to complete the service creation.

Go to the newly created service screen to find the instrumentation key, connection string, and other details, all under the Overview section. Refer to the highlighted sections in Figure 5-15.

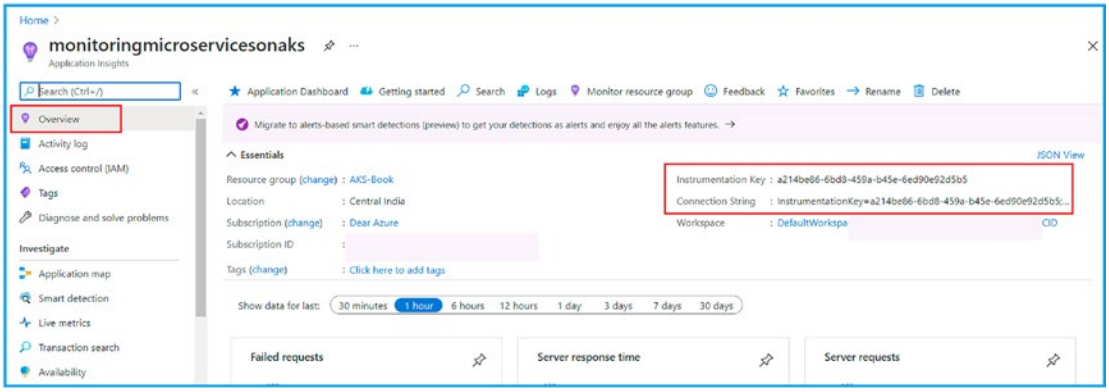


***Figure 5-15.*** *Instrumentation keys for Application Insights in Azure Portal*

---

**Tip** Using a connection string is highly recommended over an instrumentation key, as new Azure regions require connection strings. In either case, you need to create the service.

---

Next, you need enable Application Insights in your IDE. My favorite IDE is Visual Studio, but you can use other IDEs, such as like Visual Code, as well. Once you are done enabling Application Insights server-side telemetry with your IDE, download the latest stable release of the SDK by choosing NuGet Packages ➤ Microsoft.ApplicationInsights.AspNetCore. Refer to Figure 15-16.

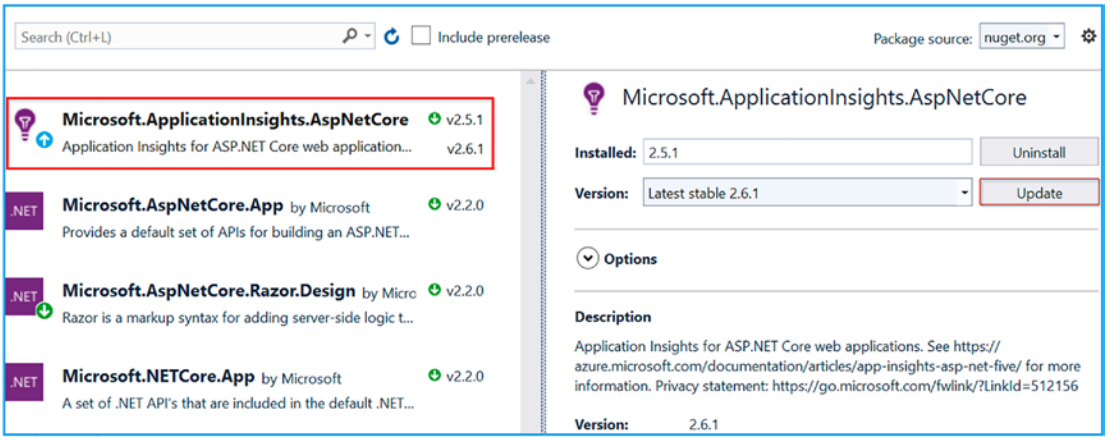


***Figure 15-16.*** *SDK from the NuGet packageImage source: Microsoft Documentation*

By adding a few lines of code to the application, it will start with the telemetry data, and it can be presented in the Applications Insights screen in Azure Portal. This data can be analyzed and can be used to create alerts based on the state of your applications.

For the code and other options, refer to `https://docs.microsoft.com/en-us/azure/azure-monitor/app/asp-net-core`.

# Summary

This chapter presented things you should know about securing and monitoring AKS-based applications. With the provided links, make sure you have proper hands-on experience with these issues. This chapter ends the theoretical part of the book. The next chapter uses a practical step-by-step approach to implementing CICD for Azure Kubernetes Service-based applications. Until then, happy Azure learning.