E-book series

Microsoft Azure

Cloud-native security for container apps

An end-to-end approach for Azure Kubernetes Service

Introduction

Containerization is a game-changer for application development and deployment, but its steep learning curve makes security a challenge. Containers are dynamic—constantly being created, destroyed, and orchestrated across multiple nodes in a continuous software development lifecycle. Threats come in many forms, from vulnerabilities in your code, to overprivileged access and critical misconfigurations, to ever-evolving exploits. Traditional security solutions weren't designed to handle the ephemeral, dynamic nature of containers, with their rich data, controls, and configuration layers. Even popular security solutions rely on the visibility provided by a cloud service provider's APIs, and each has its own strengths and weaknesses. The blind spots in a traditional security solution make it much harder for security engineers and analysts to triage and respond to threats in cloud-native container applications.

Container app security requires a multilayered defense strategy and a security-first mindset throughout the entire software supply chain—every process, action, and person involved, from development, to build, to deployment , to runtime. DevSecOps provides a model for best practices, and implementation pays off.

According to a study by Ponemoan Institute, organizations that go all in with their DevSecOps adoption showed a sizable return on investment ($1.68 million) compared to those that don't.[1]

You also need cloud-native tools that establish a strong cloud security posture and provide real-time threat protection and response for your Kubernetes workloads. When you host your container apps in a managed service like Azure Kubernetes Services (AKS), you share responsibility for security with your cloud service provider. With AKS, you benefit from the security components in Kubernetes, such as pod security standards and Secrets, in addition to the widely trusted data and application protections and compliance offerings in Azure.

This guide provides an overview of best practices in cloud-native security for each stage in your container app's journey from code to cloud and how AKS works better with Microsoft Defender for Containers.

[1] Cost of a Data Breach Report. Ponemon Institute. 2023.

Start secure, stay secure

Container app security follows a simple formula—start secure and stay secure. The first step is to understand the unique risks in cloud-native environments:

1 **Compromised account.** Cloud account credentials must be secured to prevent attackers from overtaking your cluster.

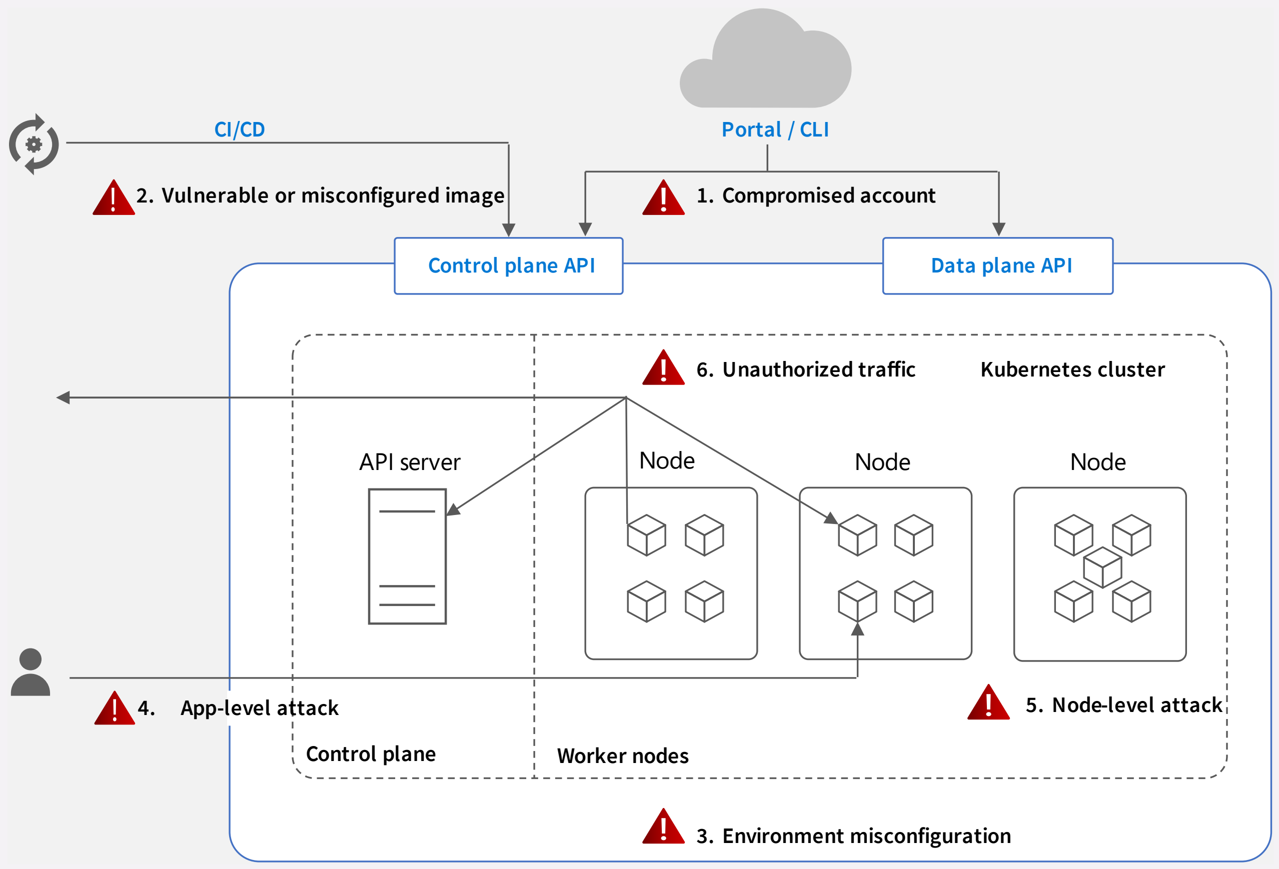
2 **Vulnerable or misconfigured images.** Poor image configuration can lead to security vulnerabilities down the pipeline.

3 **Environment misconfiguration.** Network policies, access controls, secrets, and data must be handled correctly.

4 **App-level attack.** Attackers can exploit vulnerabilities in the application code, configuration, or runtime environment.

5 **Node-level attack.** The underlying nodes that host your clusters, including the host file system, must be secured.

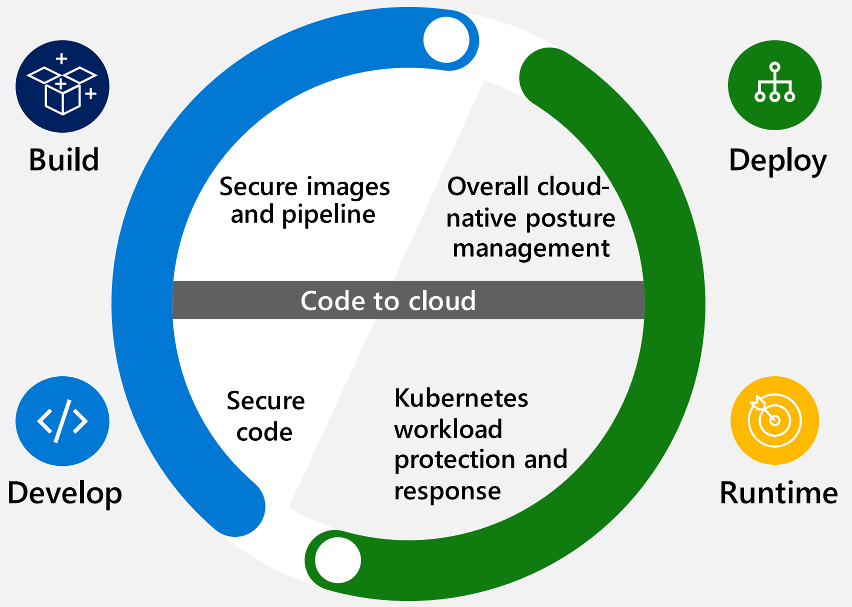
6 **Unauthorized traffic.** Attackers can exploit exposed clusters and ports to compromise resources and sensitive data.



Cloud-native security requires threat detection and mitigation in all six areas, including the environments where the core Kubernetes services and orchestration of application workloads are executed.

To help you start secure and stay secure, a new breed of cloud-native application protection platform (CNAPP) has evolved. Gartner[2] recognizes Microsoft Defender for Cloud as a pioneer, with its comprehensive protection and agentless scanning for end-to-end visibility. Defender for Cloud can help strengthen your security posture, reduce risk throughout the cloud application lifecycle, and help protect against cyberthreats—across multicloud and hybrid environments.

Proactive posture hardening helps you start secure. Threat protection, together with automated security controls and gating, helps keep things that way. For specific container threat detection, you need cloud workload protection (CWP). Microsoft Defender for Containers is a cloud-native solution to help protect complex container estates. It features vulnerability assessments and threat detection and remediation that span multicloud and on-premises workloads.


Build
Deploy
Secure images and pipeline
Overall cloud-native posture management
Code to cloud
Secure code
Kubernetes workload protection and response
Develop
Runtime

A secure supply chain starts at coding, the Develop stage. In the Build stage, containers are built, tested, and packaged, then stored in a container registry. In the Deploy stage, the container image is pulled from the registry and deployed to Azure. During runtime, container apps running in AKS need ongoing threat protection and response.

[2] MacDonald, Neil, et al. Market Guide for Cloud-Native Application Protection Platforms, Gartner, March 14, 2023.

SAPIENS

“Overall, AKS has proven to be a reliable, agile, and scalable solution for our needs.”

Alex Zukerman
Chief Strategy Officer
Sapiens International

Develop

Security applies to all the artifacts you create, including infrastructure as code (IaC) and application and container manifests. However, 54% of organizations don't include security in the development phase,[3] and 87% of containers run with critical or high vulnerabilities.[4]

You've probably heard the phrase *shift left*—that is, start performing security testing earlier in the software development lifecycle. It's much easier for developers to detect and fix problems in their code while it's still fresh in their mind, and it's exponentially more expensive to remediate errors detected later in the cycle.

When you apply shift-left thinking to container apps, you check the integrity of your integrated development environment (IDE) and make sure your code follows recommended design patterns.

In the Develop stage, you should:

- Use code review tools to automate and improve your code quality and efficiency. For example, you can add static application security testing (SAST) to your IDE to check source code for vulnerabilities and software composition analysis (SCA) to track and govern any open-source components you're using.
- Scan your codebase to identify dependencies and any vulnerabilities they may introduce. In GitHub, we recommend adding GitHub Advanced Security to help you identify security vulnerabilities in your codebase and environment.
- Store secrets in a protected vault, including passwords, keys, tokens, storage account keys, certificates, and credentials used in shared nonproduction environments.

[3] Microsoft Cloud Security Priorities and Practices Research.
[4] "Sysdig 2023 Cloud-Native Security and Usage Report." Sysdig.

Build

During the Build stage, developers package their application code and dependencies into a container image and push it to a registry. Some container images may come from an external source or vendor, such as service proxies like nginx and Envoy or logging and metrics images like Fluentd and Kafka. Build security protects the integrity of your workload using safeguards that help ensure image compliance and that help prevent the use of vulnerable images.

With AKS, you can authenticate with the container registry in several ways to make sure that deployments are coming from a trusted location and sign container images to ensure their authenticity and integrity. AKS enables you to attach signatures to your images using the open-source Notation, and it also lets you use Azure Key Vault to store certificates and signing keys with the Notation AKV plugin (azure-kv).

The results of code scanning during builds can help you gate deployments. Defender for Containers helps detect vulnerabilities, and it also provides guided remediation.

In addition to protecting your container images from threats, you must also harden the entire continuous integration (CI) and continuous delivery (CD) pipeline and infrastructure. Acquiring, cataloging, and building images are the first three stages in the Microsoft Containers Secure Supply Chain. This framework provides best practices to help your organization reduce risk and create a container infrastructure with a standard security portfolio.

In the Build stage, you should:

- Follow the Containers Secure Supply Chain recommendations.
- Host container images in a catalog that is monitored continuously.
- Scan container images for vulnerabilities and malware and to detect drift—that is, an executable that doesn’t belong to the container base image or didn't come from your build environment.
- Make sure security reports are signed, to help ensure authenticity and integrity.
- Monitor the container image lifecycle and retire any that are out of support.

Deploy

In this stage, you use preflight checks to help ensure that container images are trusted and comply with your policies *before* they're deployed. Untrustworthy, noncompliant container images can introduce malware or vulnerable code in the hosting environment, so a best practice is to use automated gating to prevent unapproved images from being deployed. In addition, you need continuous, real-time validation of compliance and a way to see, resolve, and mitigate any anomalies that appear.

Observability metrics must be deployed alongside workloads. You can use any number of monitoring and visualization tools, including Azure Monitor, Prometheus and Grafana, to collect and analyze these metrics and logs.

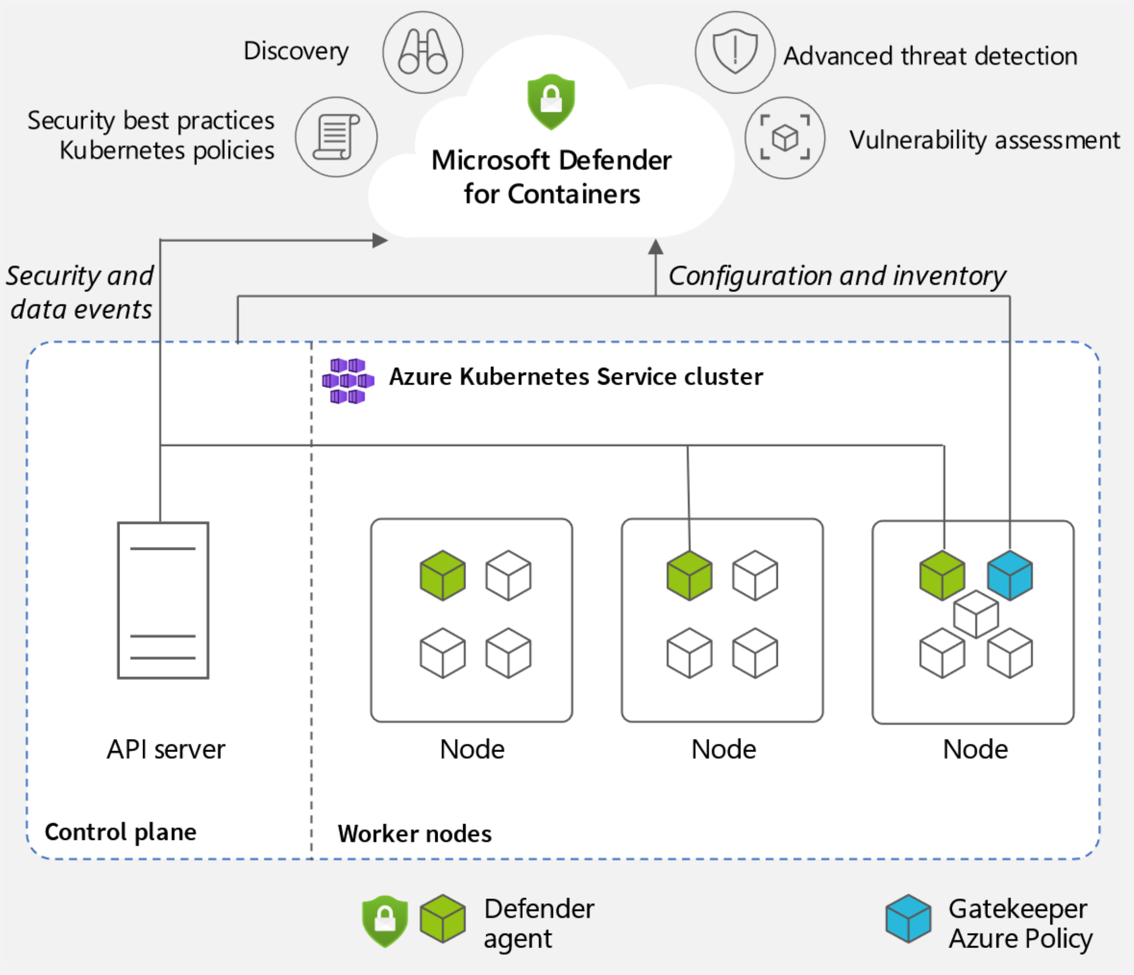
To make it easier to scan for Kubernetes misconfigurations, we recommend starting with deployment safeguards based on Azure Policy rules. These safeguards apply best practices and notify you of issues in your AKS cluster so that you can enforce rules and deny noncompliant deployments.

For greater visibility across Azure, Defender for Cloud performs agentless discovery and inventory of your Kubernetes and container registry estate across the software development lifecycle, with no footprint on your workloads or runtime.

However, to include cloud workload protection, use Defender for Containers. It helps harden your Kubernetes environment using an admission controller webhook (shown on the next page) that detects and prevents cluster-level misconfigurations, denying deployment of images from untrusted registries, making sure that containers don't run with root privileges, and more. Its support extends across multicloud environments and includes drift detection and attack disruption. In addition, to centralize safeguards and enforcement at scale for your clusters, we recommend also deploying the Azure Policy for Kubernetes add-on, an extension of the Gatekeeper v3 admission controller webhook for Open Policy Agent.

As the image shows, Defender for Containers agents help secure your container estate in four core domains:

- Security posture management based on best practices and Kubernetes policies.
- Sensor-based capabilities that support discovery and Kubernetes data plane hardening.
- Advanced runtime threat protection on clusters, nodes, and workloads.
- Agentless vulnerability assessment for your container images, including registry and runtime recommendations, remediation guidance, quick scans of new images, real-world exploit insights, exploitability insights, and more.



Defender for Containers assists you with four core domains of container security: security posture management, discovery, runtime threat protection, and vulnerability assessment.

In the Deploy stage, you should:

- Follow the Deploy stage recommendations for the Containers Secure Supply Chain.
- Verify all the image signature and integrity checks done in previous stages as part of your preflight checks and automate the verification process in your CI/CD pipeline.
- Use autoinstrumentation to enable Azure Monitor Application Insights to make telemetry like metrics, requests, and dependencies available in your Application Insights resource.
- Check that security and regulatory standards are followed, such as the secure configuration benchmarks and best practices recommended by the Center for Internet Security (CIS). Microsoft Defender for Cloud does this automatically.
- Require strict authentication, access control, and file permissions to deny unauthorized access to the deployment platform and to prevent leaked credentials or unauthorized changes.
- Use ongoing image scans to detect newly discovered vulnerabilities and end-of-support images. Assign a vulnerability score such as *critical*, *high*, *medium*, or *low*, and then enforce deployment policies and restrict images above a certain threshold.

Runtime

Up to this stage, container security means doing everything possible to prevent an attack. However, you also need assurance that your container apps are running in a trusted execution environment, protected from compute, storage, and access threats. Only sanctioned processes should be operating within a container namespace. Any unauthorized resource access must be instantly detected and prevented, and network traffic must be monitored for hostile intruders.

A determined adversary can find a way past your existing guardrails and checks. Defender for Containers provides continuous runtime threat protection and helps you identify, investigate, and mitigate a possible breach in real time.

In the Runtime stage, you should:

- Follow the Run stage recommendations for the Containers Secure Supply Chain.
- Clean up stale images from Kubernetes nodes to remove the risk of non-compliant images. For example, you can use Eraser or AKS Image Cleaner to quickly identify and remove stale images.
- Enable AKS auto-upgrade so that your clusters stay up to date and you don't miss the security releases or patches from AKS and upstream Kubernetes.
- Ensure that your cloud-native detection and response solutions are compliant and secure across all environments. A unified security operations platform can do this. For example, Microsoft Defender for Cloud provides recommendations to secure and configure your endpoint detection and response solutions.
- Work toward Zero Trust, a "never trust, always verify" strategy for designing and implementing applications.

CANARY SPEECH

“This [passing security audits] was much easier with Azure than our previous platforms, and it was far quicker than building the environment out ourselves.”

Henry O'Connell

Cofounder and Chief Executive Officer
Canary Speech

Better together

Azure Kubernetes Service and Defender for Containers work better together to provide multilayered security for container apps.

Microsoft Defender for Cloud is a CNAPP solution to strengthen security posture, reducing risk throughout the secure supply chain and helping protect against cyberthreats—across multicloud and hybrid environments. It provides wide coverage of cloud-native workloads, assets, and resources, including Kubernetes, container registries, code repositories, servers, storage, databases, and APIs. It supports security operations (SecOps) on a unified platform with full capabilities of extended detection and response (XDR) and security information and event management (SIEM).

Microsoft Defender for Containers provides advanced threat protection for containers. As a cloud-native CWP solution, it can help you improve, monitor, and maintain the security of your containerized assets and applications across multicloud and on-premises environments. Graph-based queries give you deeper visibility into complex container landscapes, on top of added image assurance and control plane and data plane insights.

New Azure subscribers can try Defender for Containers for three months at no charge (for new subscriptions). Go to Protect your Azure containers with Defender for Containers.

Learn more at azure.com