Securing Containers

Mark Heath
MICROSOFT AZURE MVP

@mark_heath https://markheath.net

Overview

Securing container images

Securing container hosts

Managing container privileges

Securing container networking

Monitoring containers

Recommendations and “best practices”

- Azure Kubernetes Service (AKS)

Securing Container Images

Container Image Security

Third party container images

- External container “registry”

Custom container images

- Private container registry

Images must be trusted

- Free from malware and vulnerabilities

Using “Official” Container Registries

Elasticsearch

- “Elastic Docker Registry”
- docker.elastic.co/elasticsearch/elasticsearch:7.1.1

.NET Core

- “Microsoft Container Registry” (MCR)
- mcr.microsoft.com/dotnet/core/aspnet:2.1

Confidence in image integrity

Docker Hub

- Use with caution: anyone can upload

sixeyed/elasticsearch ★

By sixeyed • Updated a month ago

Windows Nano Server image for Elasticsearch.

Container

Pulls 8.4K

Overview Tags

Elasticsearch

Windows Nano Server image for Elasticsearch. (Dockerfile)

Usage

Run in the background to start a containerized Elasticsearch node:

```
docker run -d -p 9200:9200 -p 9300:9300 --name elasticsearch sixeyed/elasticsearch:nanoserver
```

Docker Pull Command

```
docker pull sixeyed/elasticsea
```

Owner

sixeyed

https://hub.docker.com/r/sixeyed/elasticsearch

Building Container Images

Base images

- Use official images
- Use lean images (e.g. Alpine or CoreOS)
- Minimize attack surface area
- Approval process

Building images

- Trusted CI server
- Scan images after building
- e.g. Twistlock or Aqua

Storing Container Images

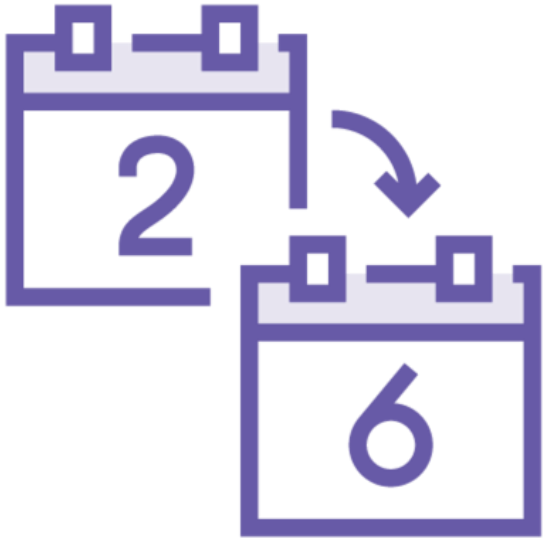
Azure Container Registry (ACR)

- Private image hosting
- Host images where they will run
- Guard credentials carefully

ACR Premium

- Content trust (signed images)
- Restrict network access

Keeping up to Date



New vulnerabilities are discovered regularly

- You need a process for updating images

Don't patch running containers

- Replace them

Automate building new images

- Triggered by updates to base image

Automate scanning for new vulnerabilities

Azure Container Registry (ACR) Tasks

- Triggered by Git commits
- Or base image updates

Securing Access to the Container Host

Keeping Container Hosts Updated

Container host options:

- Azure Container Instances (ACI)
- Azure Web App for Containers
- Azure Service Fabric
- Azure Kubernetes Service (AKS)
- Virtual machines

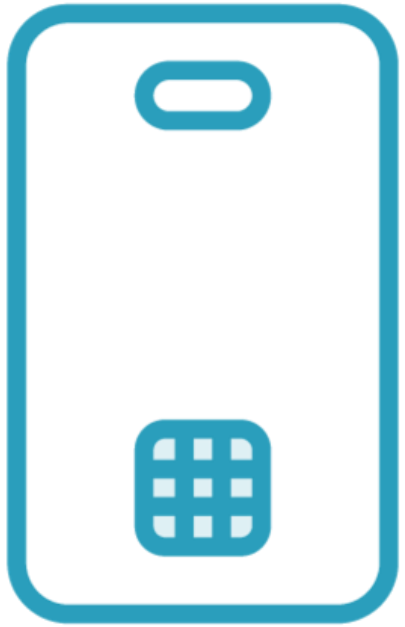
Securing container hosts

- Patch with OS updates
- Use latest container runtime

A Service Fabric cluster can be configured for automatic upgrades

Kubernetes is **not** automatically upgraded

Container Host Administrator Privileges



Restrict administration access

- Use Role Based Access Control (RBAC)
- Grant specific permissions to users

Operations and support staff

- Able to diagnose and resolve problems
- Not able to access customer data

Securing the Kubernetes API

Kubernetes API

- Supports Role-Based Access Control (RBAC)

Integrate AKS with Azure Active Directory

- Grant AD users access to the Kubernetes API
- Define “roles” with permissions within a Kubernetes “namespace”

Best practices for authentication and authorization in Azure Kubernetes Service (AKS)

04/24/2019 • 6 minutes to read • Contributors all

In this article

As you deploy and maintain clusters in Azure Kubernetes Service (AKS), you need to implement ways to manage access to resources and services. Without these controls, accounts may have access to resources and services they don't need. It can also be hard to track which set of credentials were used to make changes.

This best practices article focuses on how a cluster operator can manage access and identity for AKS clusters. In this article, you learn how to:

- ✓ Authenticate AKS cluster users with Azure Active Directory
- ✓ Control access to resources with role-based access controls (RBAC)
- ✓ Use a managed identity to authenticate themselves with other services

Use Azure Active Directory

Best practice guidance - Deploy AKS clusters with Azure AD integration. Using Azure AD centralizes the identity management component. Any change in user account or group status is automatically updated in access to the AKS cluster. Use Roles or ClusterRoles and Bindings, as discussed in the next section, to scope

https://docs.microsoft.com/en-us/azure/aks/operator-best-practices-identity

Securing Container Privileges

Apply the principle of "least privilege"

e.g. Avoid running containers as root

e.g. Avoid distributing master keys to databases

Secrets

Connection strings, API keys, etc

Environment variables

Azure Container Instances

- az container create … -e Secret=Value

Web App for Containers

- “Application Settings”

Azure Key Vault

Stores secrets, keys, certificates

Fetch at deployment time

Or fetched on demand by the container

Azure Container Instances

- Store the secret in a “secret volume”

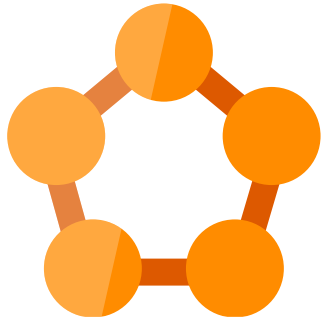
Web App for Containers

- “Key Vault References”

@Microsoft.KeyVault(SecretUri=https://myvault.vault.azure.net/secrets/mysecret/ec96f02080254f109c51a1f14cdb1931)

Secrets in Service Fabric and AKS

Service Fabric



"Secret Management"

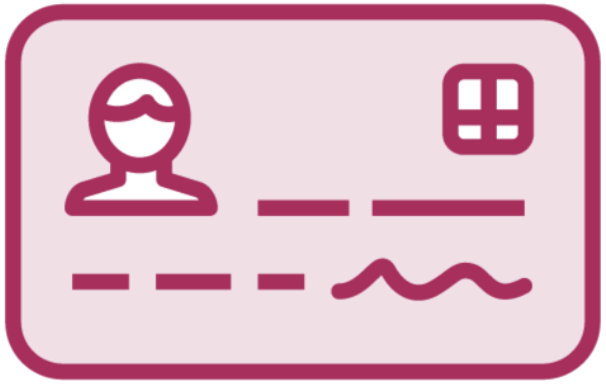
Encrypt secrets with a certificate

Kubernetes Secrets

Managed with the Kubernetes API

Can load secrets from Key Vault

Managed Identities



Creates a “service principal” in Azure AD

- Existing service principal can also be used

Container uses the service principal for authentication

- Access other Azure resources

Using Managed Identities

1 Assign a managed identity

- **ACI Container Groups**
- **Azure Web App for Containers**
- **Service Fabric**
  - Virtual Machine Scale Set
- **Azure Kubernetes Service**
  - AAD Pod Identity
  - Node Management Identity (NMI)
  - Managed Identity Controller (MIC)

2 Grant privileges to the service principal

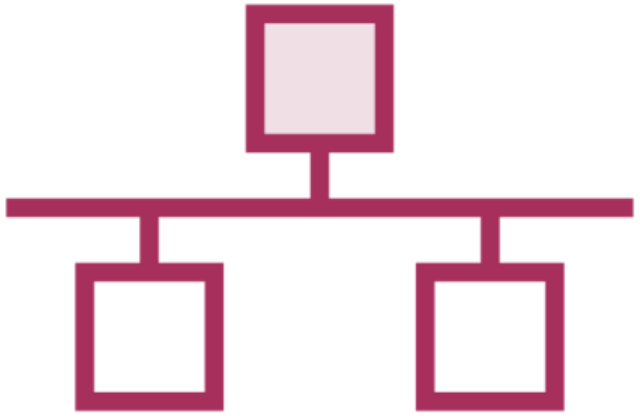
- **Permission to access other resources**
- **Apply principle of least privilege**

3 Acquire an access token

- **Request from a locally available endpoint**
- **Use token to access resources**
- **Azure SDKs simplify the process**

Securing Container Networking

Exposing Ports



Network access is locked down by default

- Ports must be exposed

Kubernetes

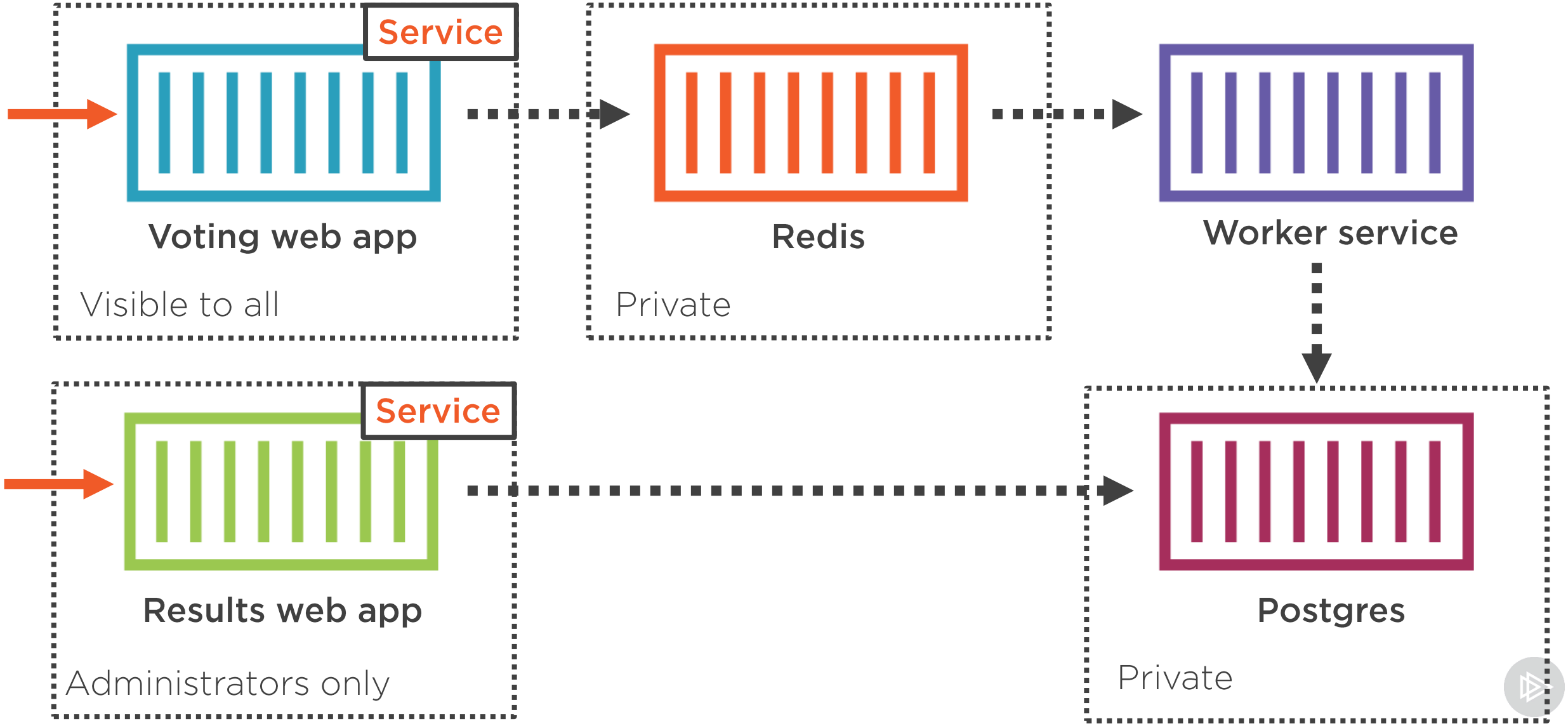
- Ingress controller

az container create ...
--ports 2368 --ip-address public

Protecting web apps

- HTTPS
- Whitelist incoming traffic

Microservice Application



Azure Container Instances

Optional public IP address

Can deploy to a Virtual Network

- (Currently in preview)

Azure Web App for Containers

Designed to expect incoming traffic

- WEBSITES_PORT

Web app listens on ports 80 and 443

- Handles HTTPS termination

Custom domain name & SSL certificate

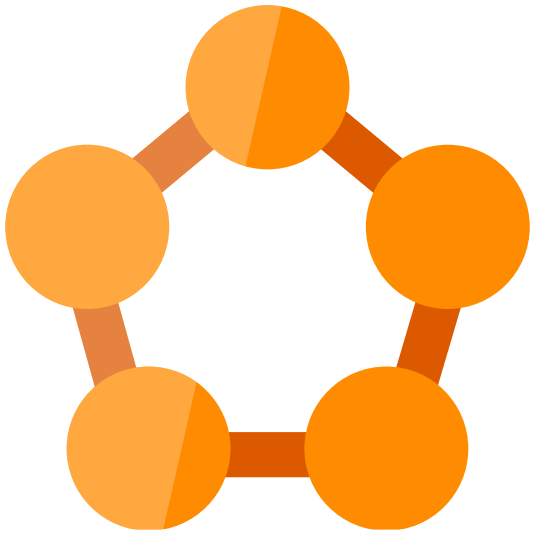
Access restriction rules

- Whitelist incoming IP addresses

App Service Authentication

- Supports several identity providers

Azure Service Fabric



Runs in a Virtual Machine Scale Set

- Can run in a Virtual Network
- Configure NSGs
- Static public IP address
- Load balancers

Securing Container Networking with AKS

Kubernetes Networking

Ingress controllers and resources

- Route incoming traffic

Networking modes

- Kubenet
- Azure CNI

https://docs.microsoft.com/en-us/azure/aks/operator-best-practices-network

Best practices for network connectivity and security in Azure Kubernetes Service (AKS)

12/10/2018 • 9 minutes to read • Contributors

As you create and manage clusters in Azure Kubernetes Service (AKS), you provide network connectivity for your nodes and applications. These network resources include IP address ranges, load balancers, and ingress controllers. To maintain a high quality of service for your applications, you need to plan for and then configure these resources.

This best practices article focuses on network connectivity and security for cluster operators. In this article, you learn how to:

- ✓ Compare the kubenet and Azure CNI network modes in AKS
- ✓ Plan for required IP addressing and connectivity
- ✓ Distribute traffic using load balancers, ingress controllers, or a web application firewalls (WAF)
- ✓ Securely connect to cluster nodes

Choose the appropriate network model

Best practice guidance - For integration with existing virtual networks or on-premises networks, use Azure CNI networking in AKS. This network model also allows greater separation of resources and controls in an enterprise environment.

Virtual networks provide the basic connectivity for AKS nodes and customers to access your applications. There are two different ways to deploy AKS clusters into virtual networks:

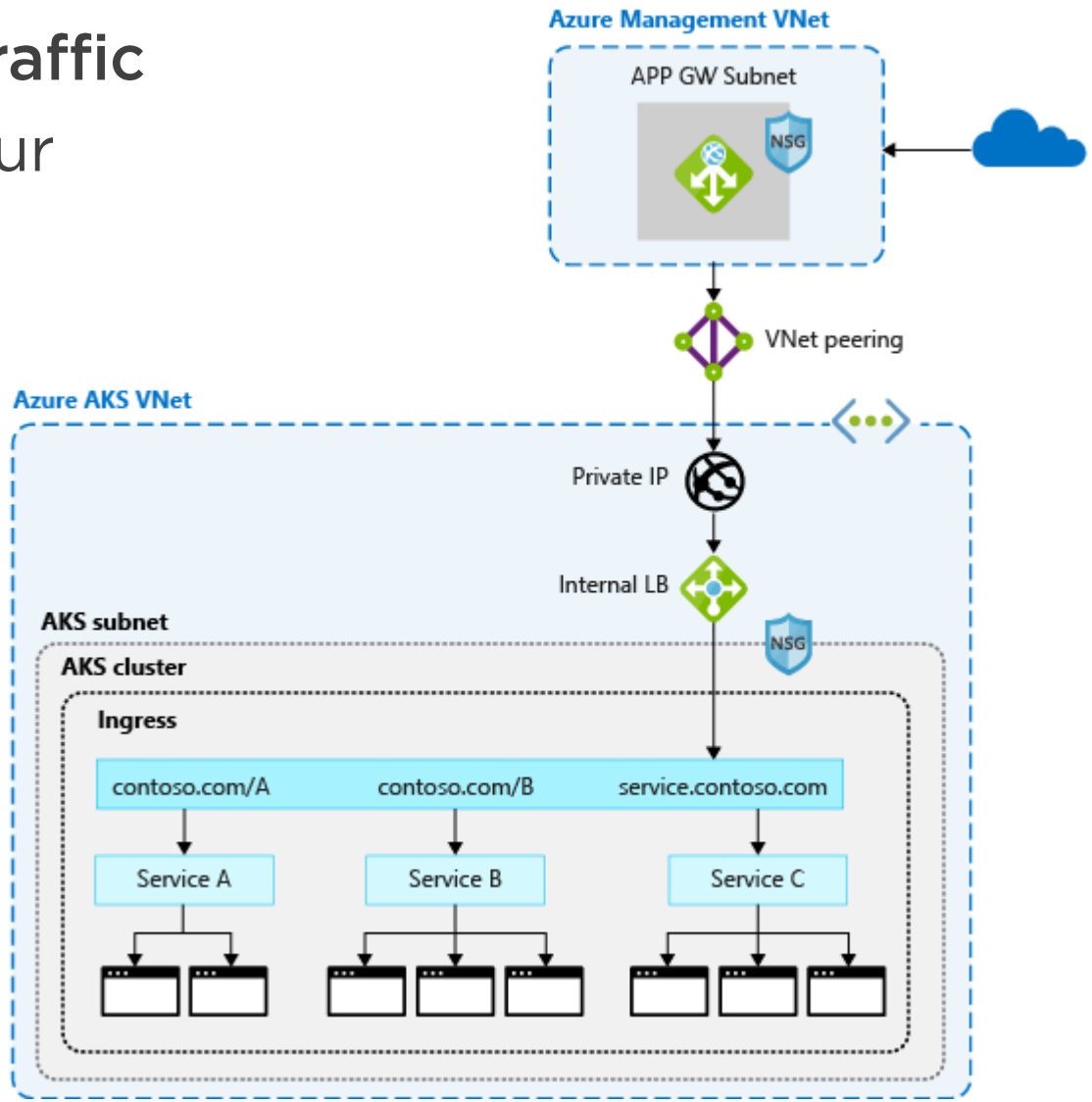
In this article

Should we allow incoming traffic from the public internet to directly reach our container hosts?

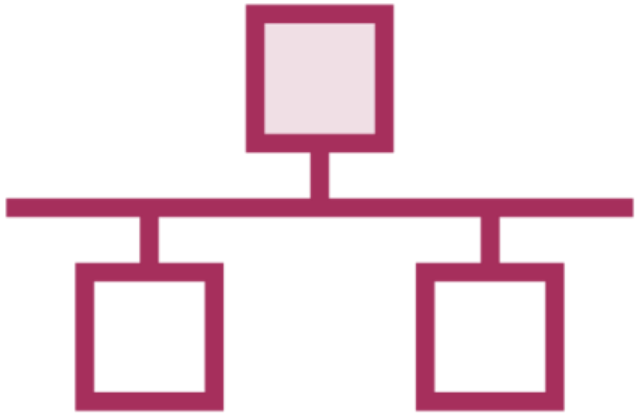
Azure Application Gateway

Receive incoming traffic

- Forward it to your container host



Communication between Containers



Which containers can communicate with each other?

Restrict to only required access

Kubernetes “network policies”

Example Network Policy

```yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: backend-policy
spec:
  podSelector:
    matchLabels:
      app: backend
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: frontend
```

Service Meshes

Istio

Linkerd

- Installs a “sidecar” container in every Kubernetes pod
- Handles all communication between services
- Implements encryption and enforces access rules
- Observability of container communications

Avoid using a shared Kubernetes cluster for production and dev/test.

Monitoring Containers

Monitoring Containers

Know what's happening in your containers

- Configure alerts

Integrate with Azure Monitor

- Capture container logs with Azure Log Analytics

Audit everything

Azure Security Center

Summary

Use official container registries

- Scan images for vulnerabilities

Use RBAC for access to container hosts

- Also for Kubernetes API

Containers should have access to only what they need

Store secrets securely

- e.g. Kubernetes secrets & Key Vault
- Use managed identities

Summary

Lock down incoming network access

- Host containers in a Virtual Network
- NSGs, Application Gateways, WAF

Secure Kubernetes networking

- Ingress controllers
- Network policies
- Service meshes

Monitor containers

- Detect and respond to attacks

https://github.com/markheath/azure-deploy-manage-containers

@mark_heath