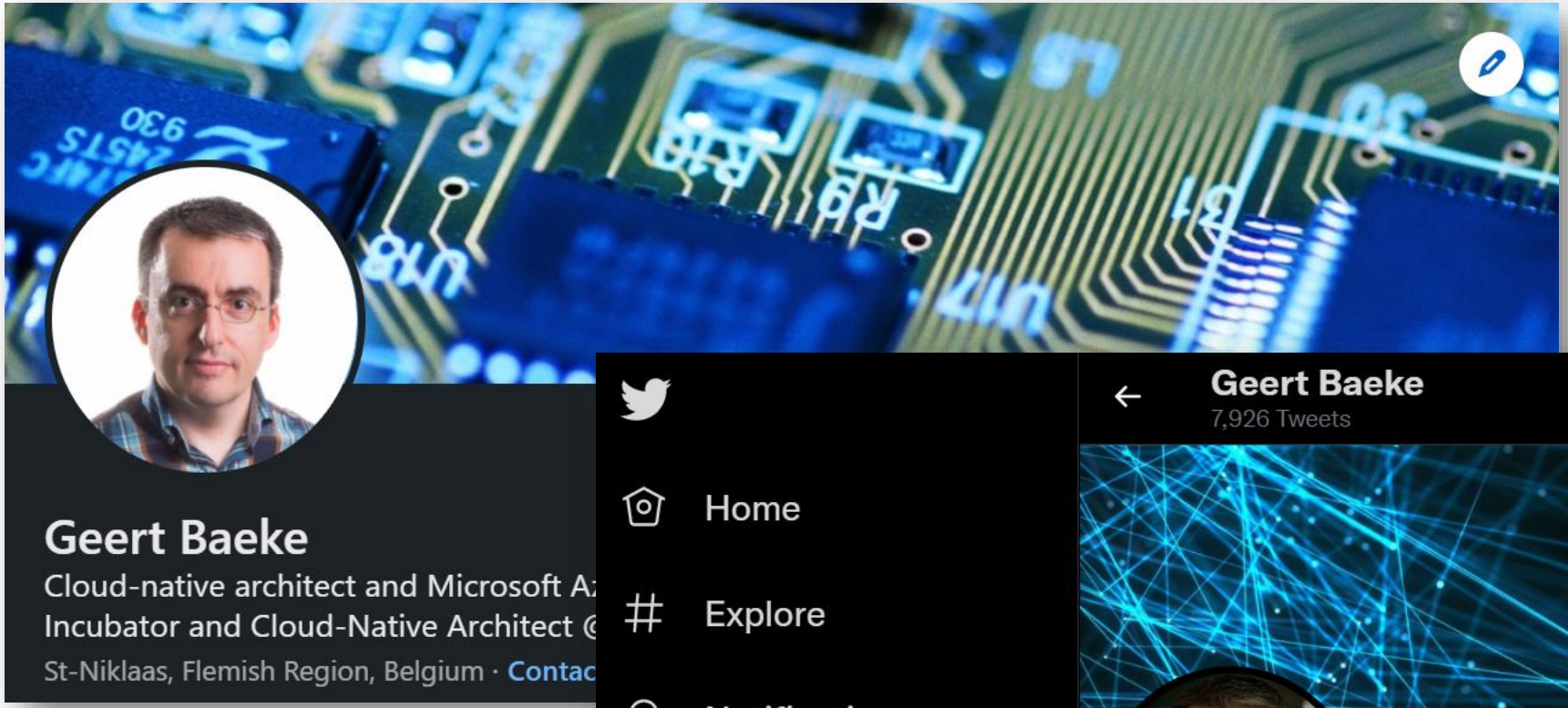
Kubernetes Deep Dive

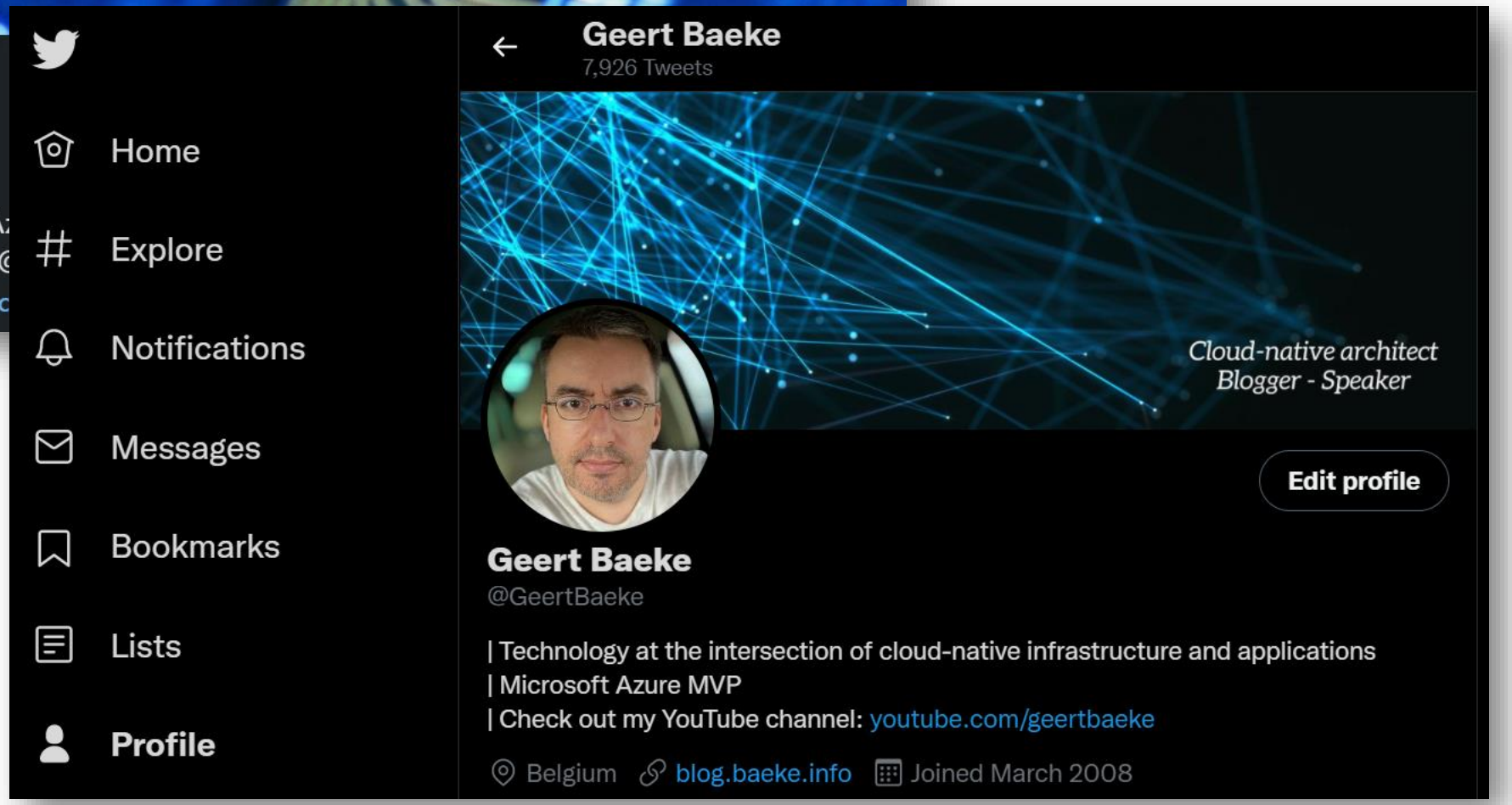
Course Overview

Deep Dive

- Builds upon the Fundamentals course
- Requires knowledge of Kubernetes architecture and the basic building blocks of **Pods**, **Deployments**, **ReplicaSets**, **ConfigMaps**, **Secrets**, ...

Deep dive ≠ under the hood look at Kubernetes


Geert Baeke
Cloud-native architect and Microsoft A
Incubator and Cloud-Native Architect @
St-Niklaas, Flemish Region, Belgium · Contac


Home
Explore
Notifications
Messages
Bookmarks
Lists
Profile
Geert Baeke
7,926 Tweets
Cloud-native architect
Blogger - Speaker
Edit profile
Geert Baeke
@GeertBaeke
| Technology at the intersection of cloud-native infrastructure and applications
| Microsoft Azure MVP
| Check out my YouTube channel: youtube.com/geertbaeke
Belgium
blog.baeke.info
Joined March 2008

What’s in this course?

AKS Networking

- Ingress and egress control
- Service Mesh

Advanced pod concepts

- Init containers
- Multi-container pods
- Sidecars
- Dapr

Advanced release concepts

- Kustomize
- Helm
- Progressive Delivery
- GitOps

Security, authentication and authorization

- RBAC
- Secrets handling
- Pod identity

AKS Networking

Kubernetes Deep Dive

What’s in this module?

- Azure Virtual Networks
- AKS and VNET Integration
- Egress control
- Network Policies
- Service Mesh

Azure Virtual Networks

Azure Virtual Network Concepts

Virtual Network

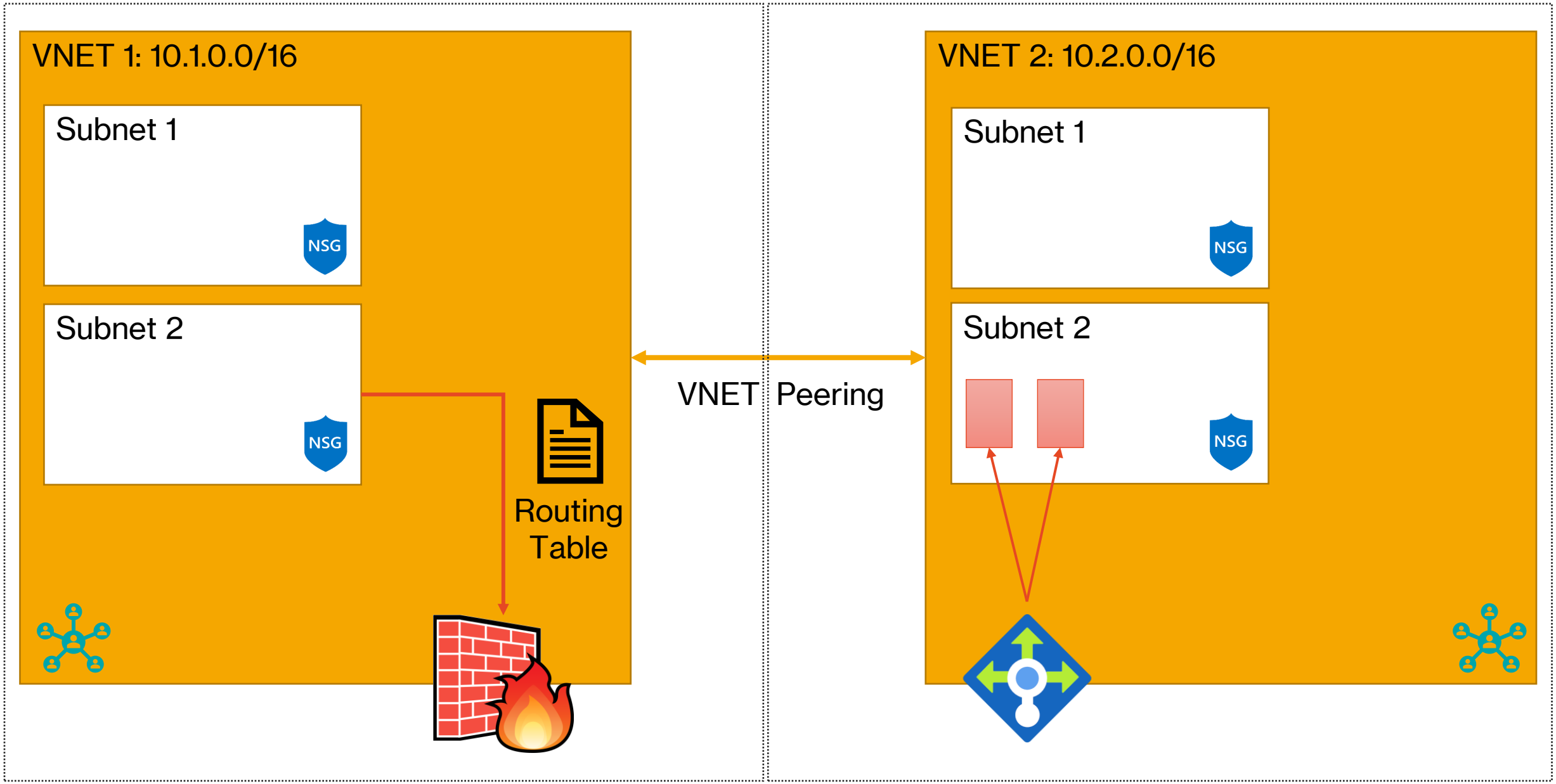
Subnet

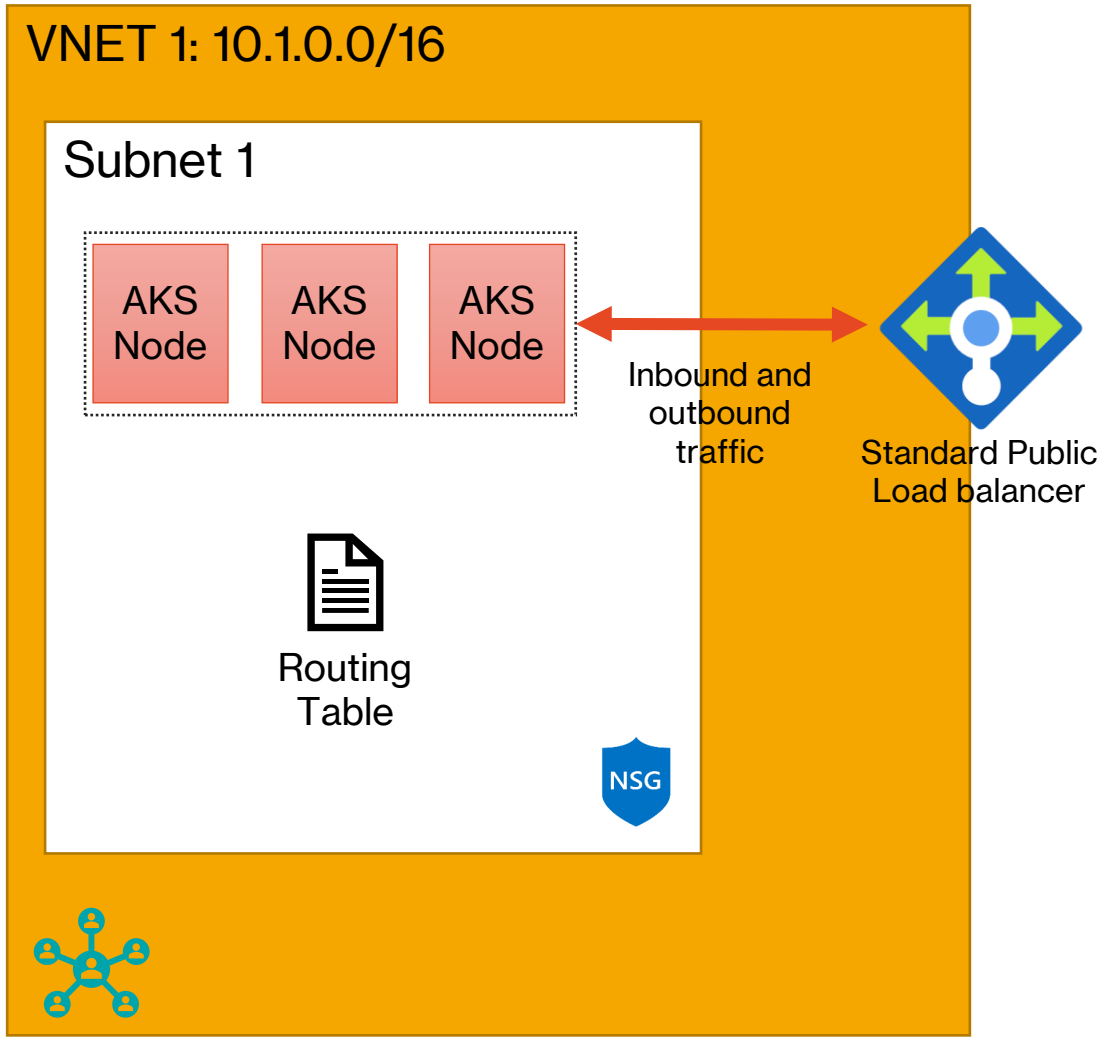
Load balancer

Routing table

Network Security Group

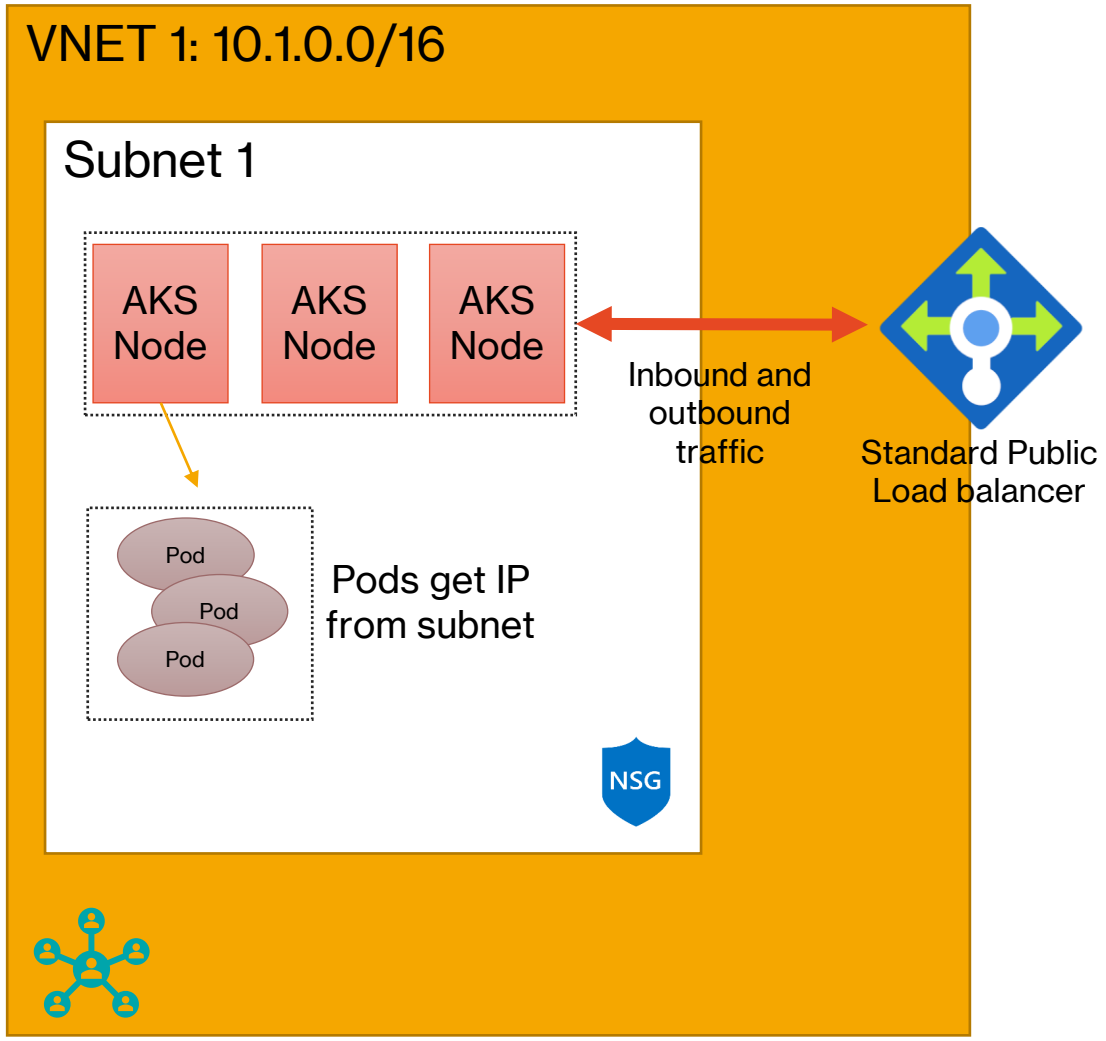
Azure Firewall


Subscription A
Subscription B
VNET 1: 10.1.0.0/16
Subnet 1
NSG
Subnet 2
NSG
Routing Table
VNET Peering
VNET 2: 10.2.0.0/16
Subnet 1
NSG
Subnet 2
NSG



AKS Kubenet networking

- Node-only routable IP address
- Pods use NAT
- 110 pods per node
- 400 nodes per cluster (400 routes in a UDR)
- Supports Calico Network Policies
- Does not support:
  - Virtual nodes with ACI
  - Azure Network Policies

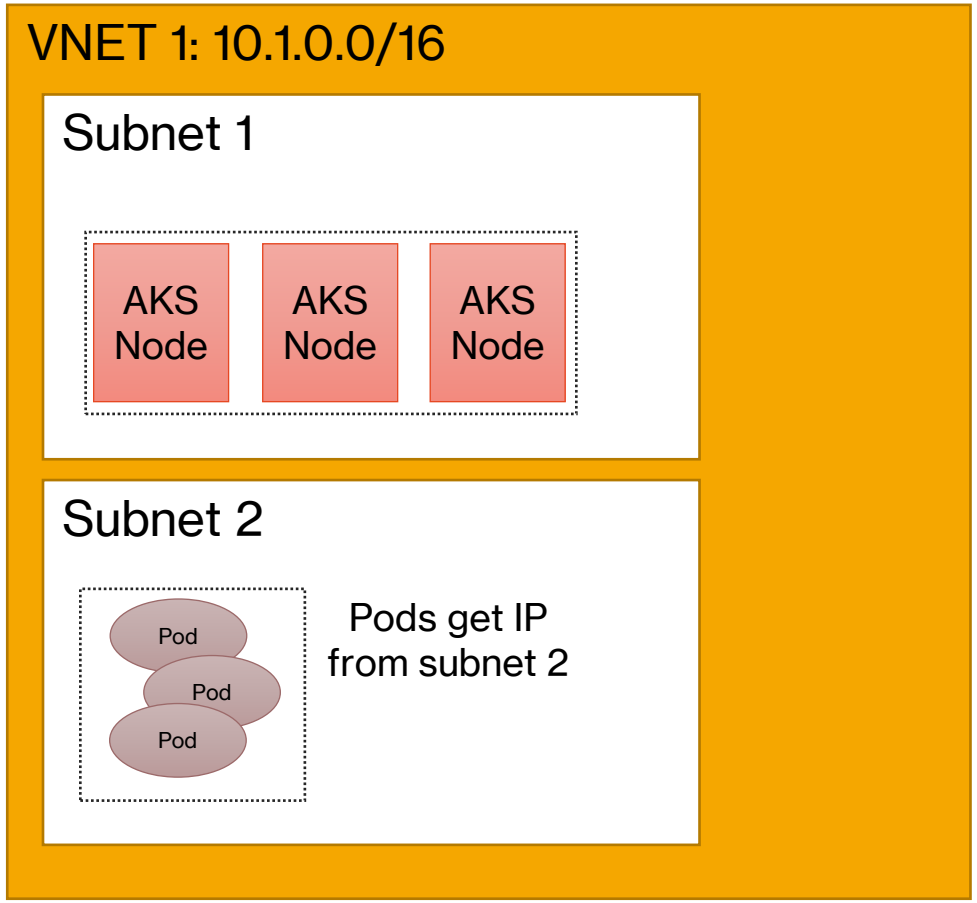


AKS Azure CNI networking

- Every pod gets a routable IP address
- IP blocks reserved per node
- 30 default, up to 250 pods per node
- 1000 nodes per cluster (100 per node pool)
- Supports:
  - Virtual Nodes
  - Calico and Azure Network Policies

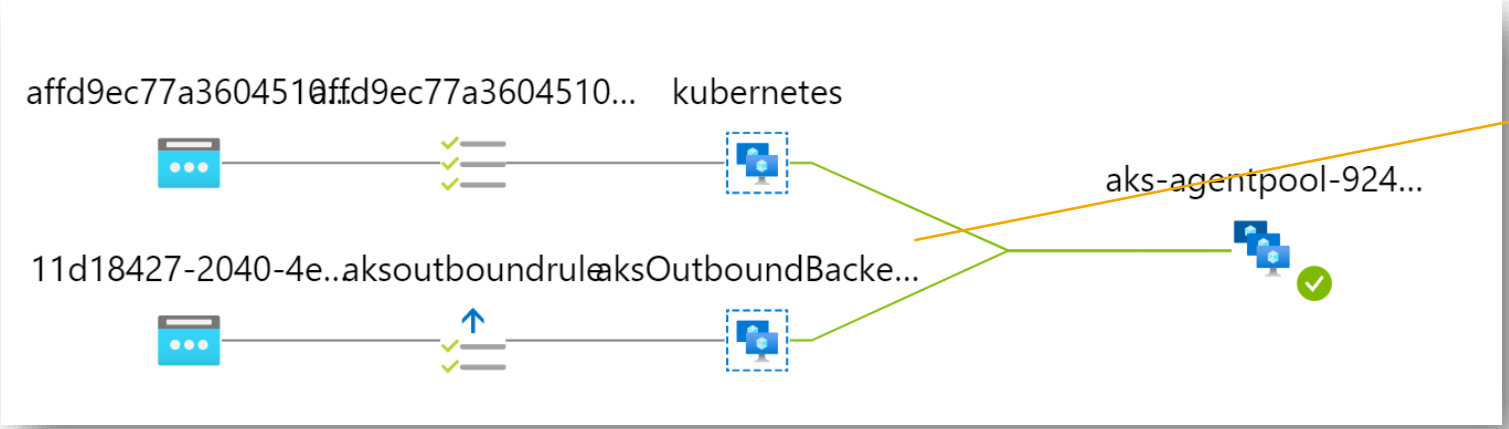
Networking preview features

- Dynamic allocation of IPs and enhanced subnet support
- IPs are dynamically allocated from a pod subnet
- Each node pool can use a different subnet for nodes and pods
- Scale node and pod subnet independently
- Works with network policies (Azure and Calico)



Controlling outbound traffic

- AKS clusters have unrestricted outbound Internet access
- Allows nodes and pods access to external resources as needed
- AKS needs access to several external resources for proper operation (via FQDNs without static IPs)



Outbound rule on standard public load balancer

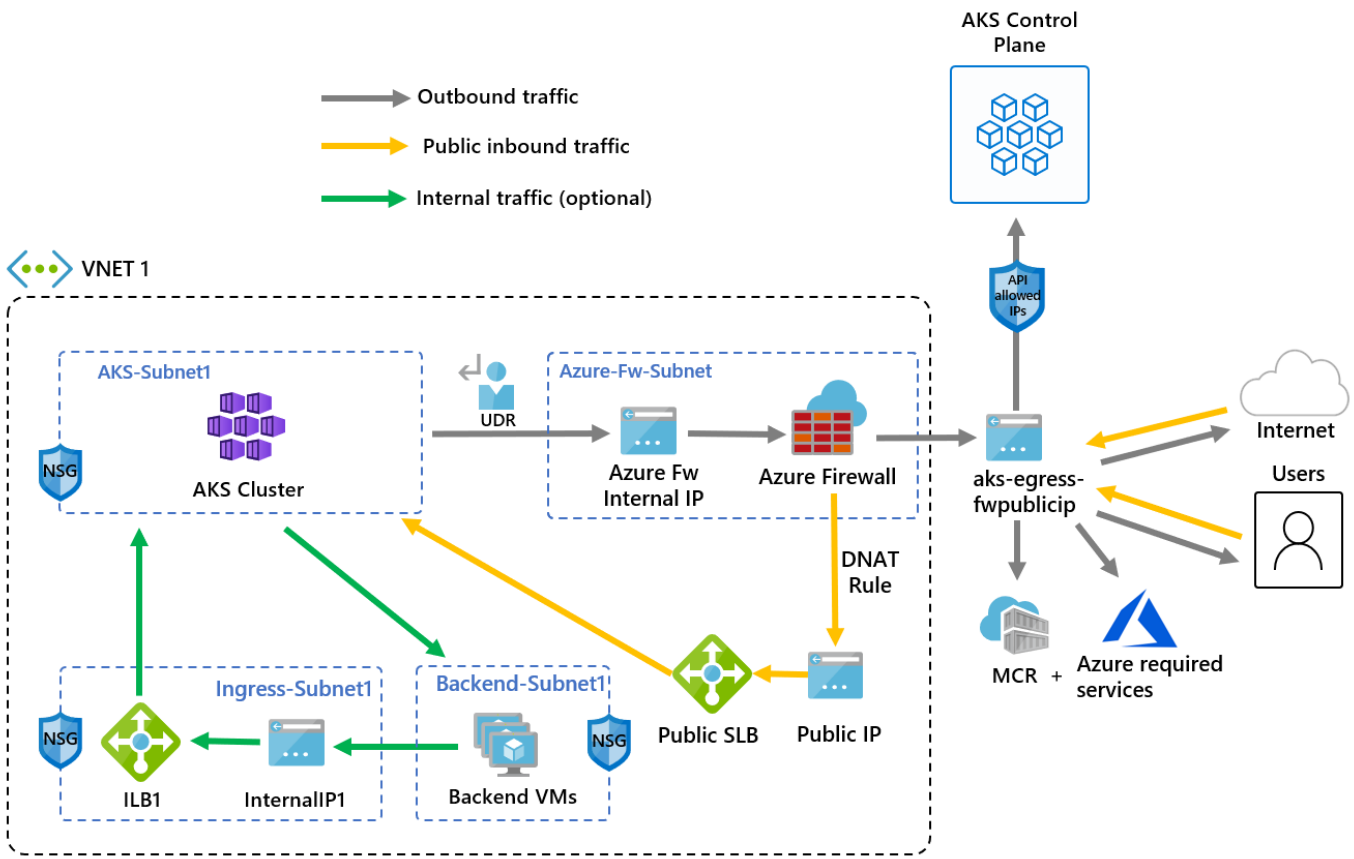
Preview: Managed NAT Gateway

- Support up to 64000 UDP and TCP traffic flows per IP address
  - → maximum 16 IP addresses
- New **--outbound-type**: managedNATGateway
  - → set # of IPs with --nat-gateway-managed-outbound-ip-count

Preview: HTTP Proxy support

- Configure AKS nodes to use a proxy to access the Internet
- Requires a proxy configuration file (JSON) and the use of the **--http-proxy-config** parameter
- Several limitations at the moment:
  - Linux only
  - Does not support monitoring addon
  - User/password authentication not supported

Using Azure Firewall

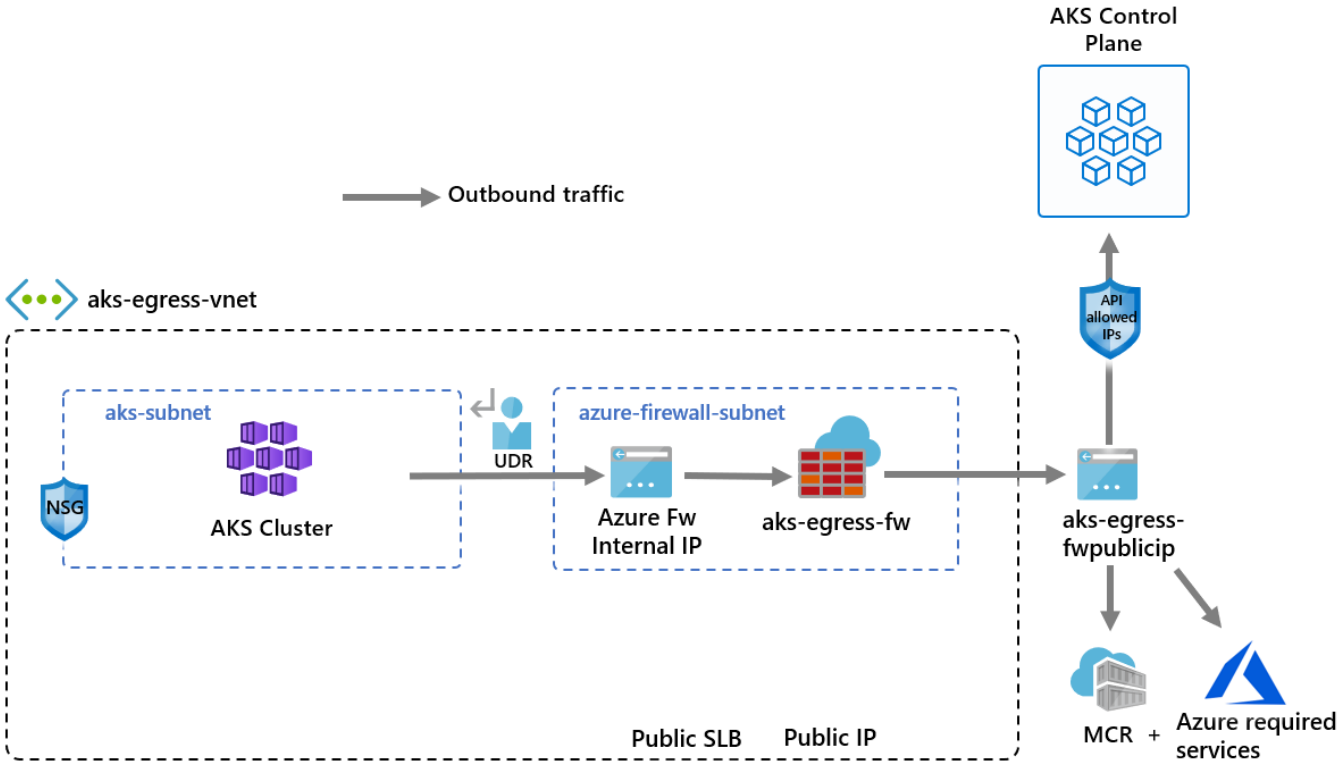


⚠ Azure Firewall has an *AzureKubernetesService* FQDN tag to simplify the configuration of outbound rules

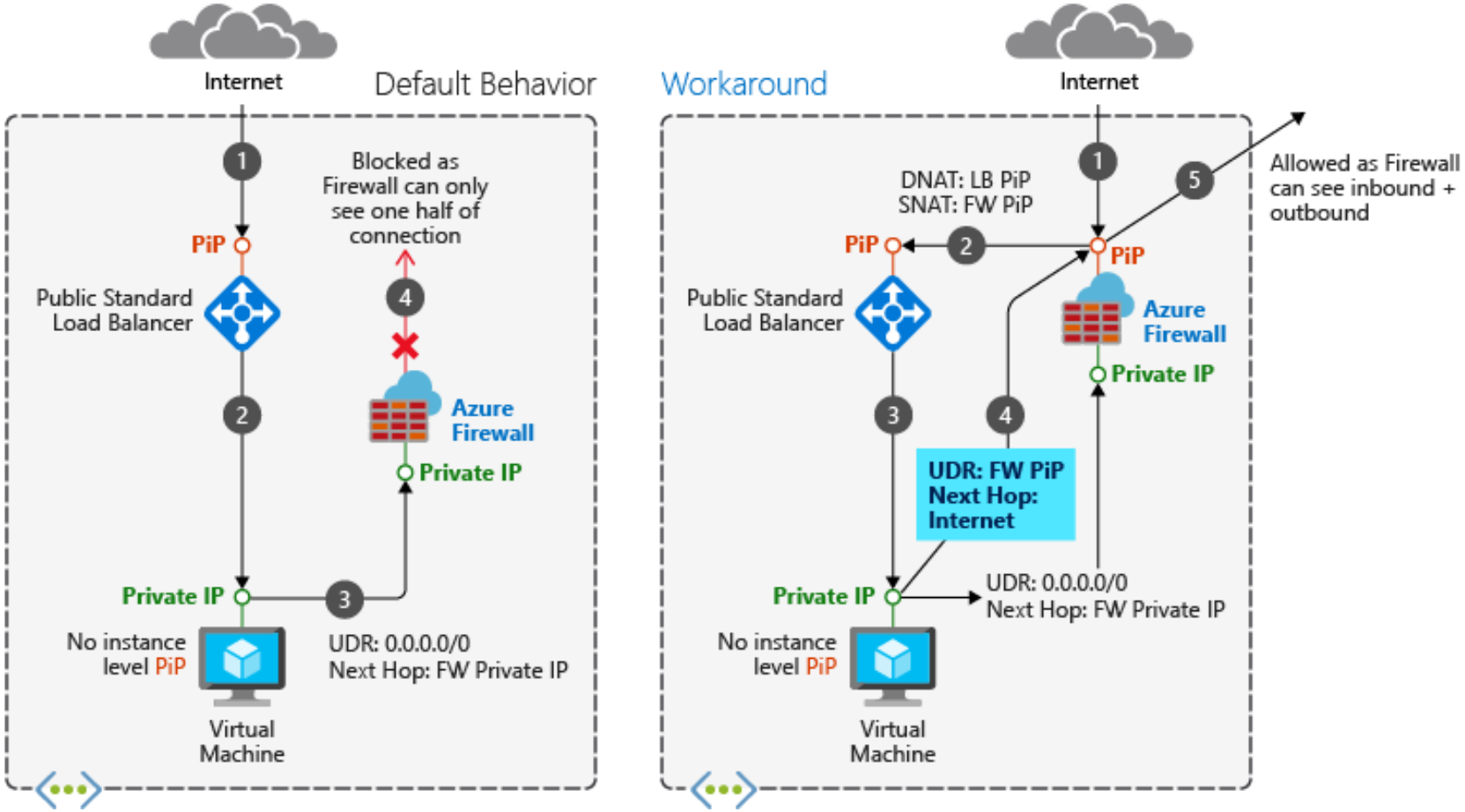
Setting the outboundType

AKS clusters have an **outboundType** setting:

- loadBalancer
- userDefinedRouting
- managedNATGateway (preview)



Beware of asymmetric routing



Cluster Deployment Tips

Home > Kubernetes services > Create Kubernetes cluster >

Create Kubernetes cluster

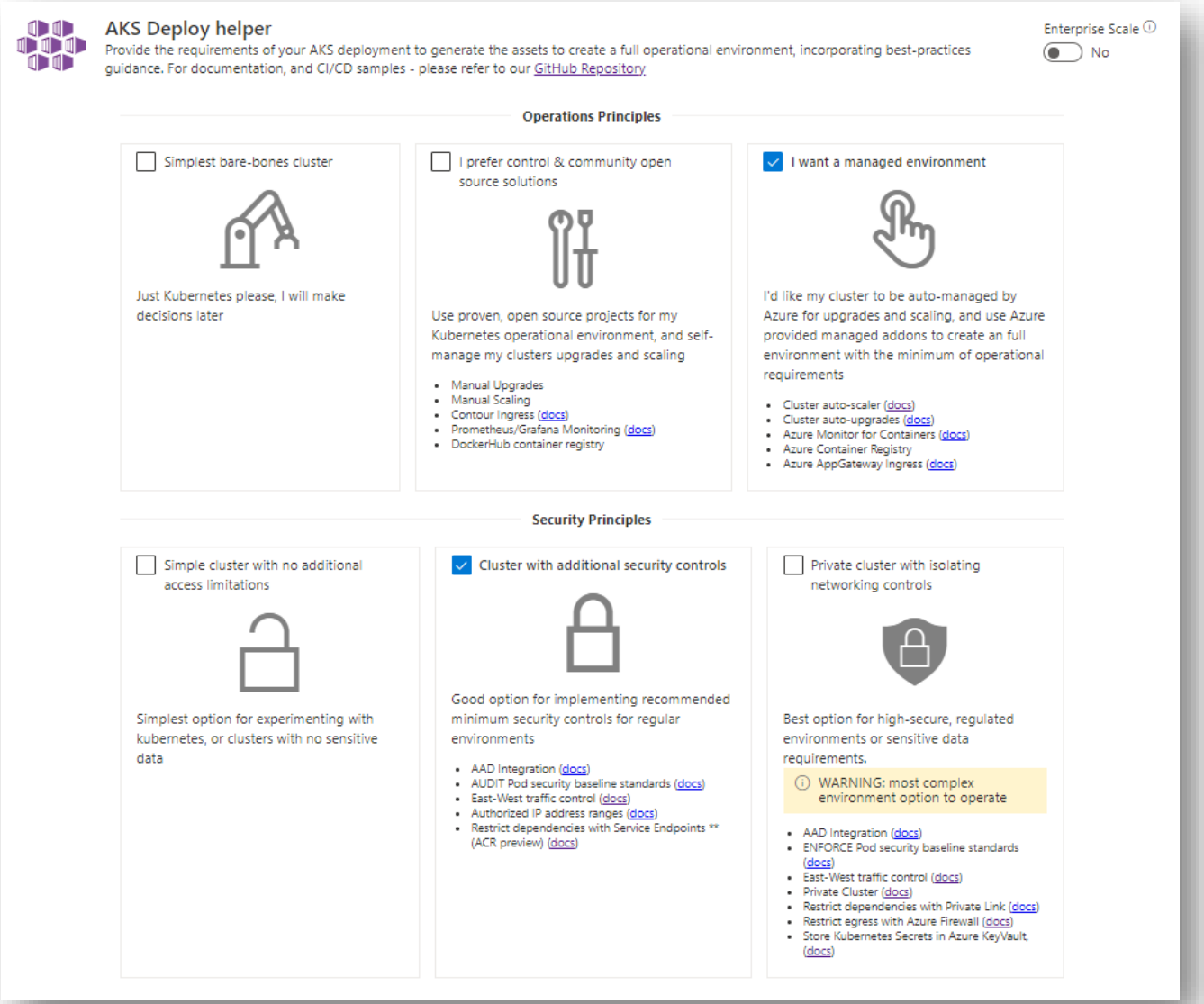
Choose your scenario to view and apply the recommended configurations suited to your needs. The settings in the table below will be updated to the specified values based on your selection. All other cluster settings will remain unchanged. Learn more

	Standard ($$)	Dev/Test ($)	Cost-optimized ($)	Batch processing ($$$)	Hardened access ($$$)
	• Best if you're not sure what to choose. • Works well with most applications.	• Best for experimenting with AKS or deploying a test app.	• Best for reducing costs on production workloads that can tolerate interruptions.	• Best for machine learning, compute-intensive, and graphics-intensive workloads. • Suited for apps requiring fast scale-up and scale-out.	• Best for large enterprises that need full control of security and stability.
System node pool node size	DS2_v2	B4ms	B4ms	D4s_v3	D4s_v3
User node pool node size	-	-	B4ms	NC6s_v2	D4s_v3
Cluster autoscaling	✓	-	✓	✓	✓
Private cluster	-	-	-	-	✓
Availability zones	✓	-	-	-	✓
Azure Policy	-	-	-	-	✓
Azure Monitor	✓	-	-	✓	✓

AKS Deploy Helper

See https://azure.github.io/Aks-Construction

Use for learning; not recommended to use as-is in production



Uptime SLA & HA

- Use the Uptime SLA in production and availability zones
- 99.95% vs. 99.5% SLO

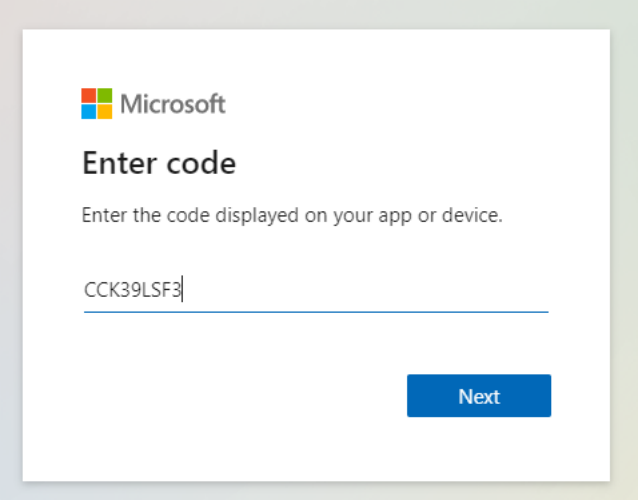
```
gbaeke  ~  $   kubectl describe nodes | grep -e "Name:" -e "failure-domain.beta.kubernetes.io/zone"
Name:                aks-agentpool-13192139-vmss000003
                     failure-domain.beta.kubernetes.io/zone=westeurope-1
Name:                aks-agentpool-13192139-vmss000004
                     failure-domain.beta.kubernetes.io/zone=westeurope-2
```

label on each node
use **kubectl describe node <nodename>** to see all labels

⚠ k8s 1.17.0 and later: use of **topology.kubernetes.io/zone** label (other is deprecated)

Azure Active Directory Authentication

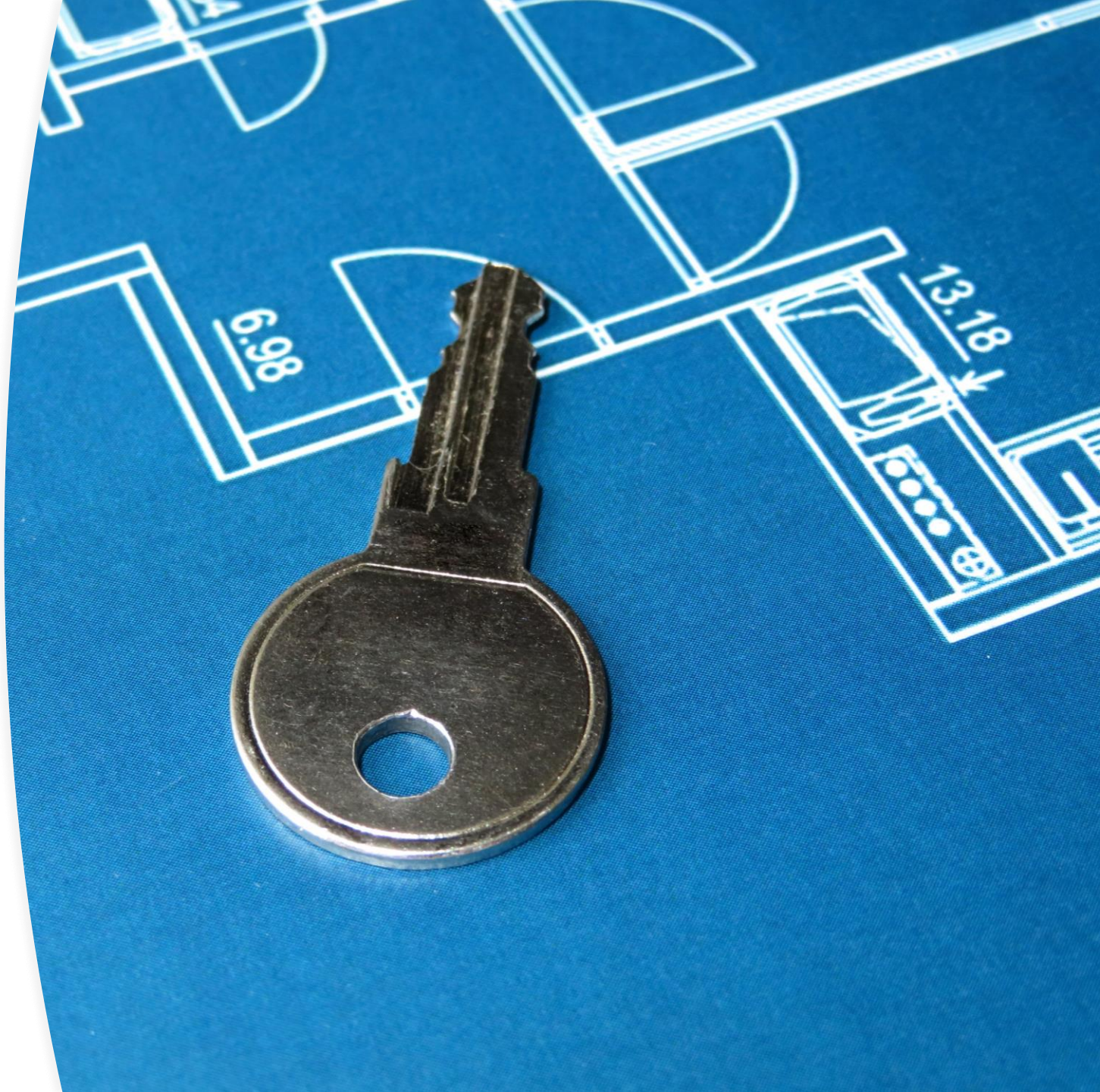
- Use AKS-managed option
- Add one or more Azure AD groups




```
gbaeke  ~  $   az aks get-credentials -n myCluster2 -g rg-aks
The behavior of this command has been altered by the following extension:
Merged "myCluster2" as current context in /home/gbaeke/.kube/config
gbaeke  ~  $   kubectl get nodes
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code CCK39L
SF3 to authenticate.
```

Cluster infrastructure authentication

- AKS comes with a cloud provider that needs to deploy additional resources:
  - Load balancers
  - Managed disks
- Cloud provider needs an identity
  - Service principal – or –
  - Managed Identity
- Use managed identity 👍
  - Wrapper around service principals
  - No need to update secrets



Control plane and kubelet identities

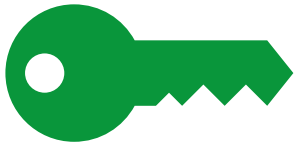
Control plane

Create load balancers

Create AKS managed public Ips

Cluster autoscaler

Azure Disk and Azure File



kubelet

Authentication with Azure Container Registry (ACRPull role)

MC_rg-aks_clu-vn_westeurope | Access control (IAM)

Resource group

Search (Ctrl+/)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Resource visualizer
- Events

Settings

- Resource costs
- Deployments
- Security
- Policies
- Properties
- Locks

Monitoring

- Insights (preview)

Add | Download role assignments | Edit columns | Refresh | Remove | Got feedback?

Check access | **Role assignments** | Roles | Deny assignments | Classic administrators

Number of role assignments for this subscription

56 2000

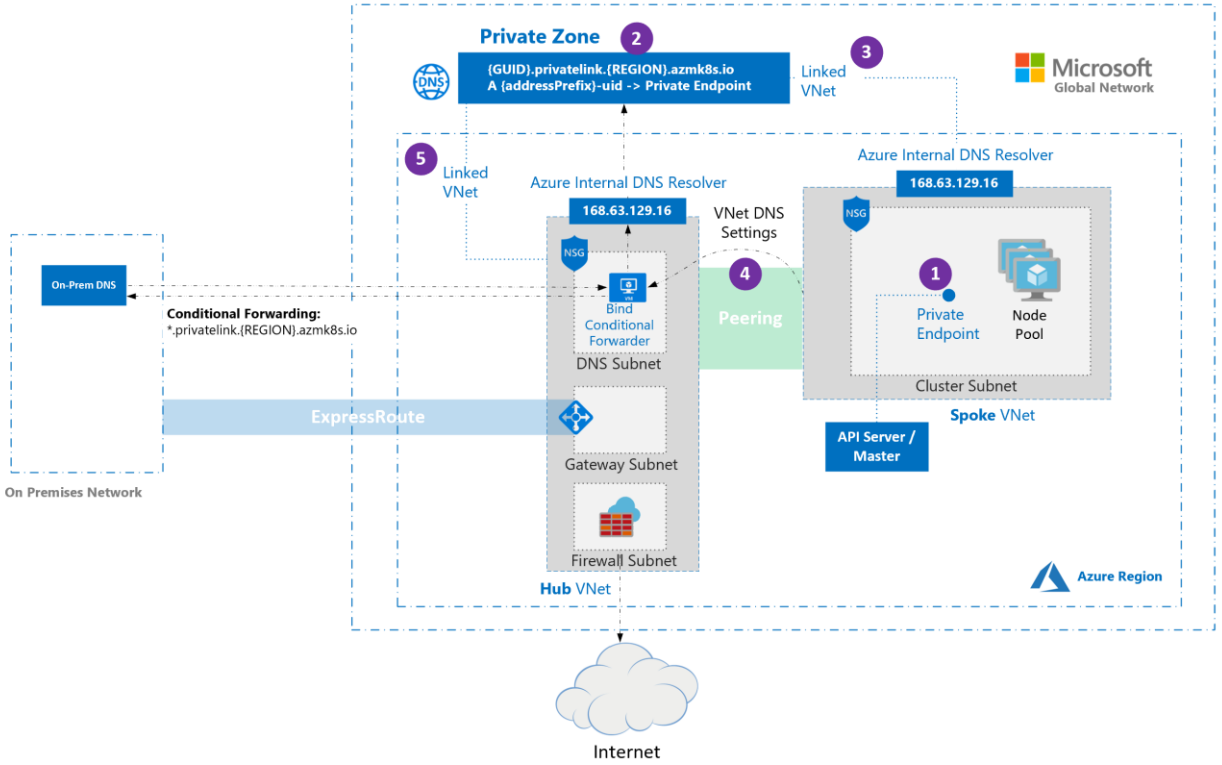
Search by name or email | Type : **All** | Role : **All** | Scope : **All scopes** | Group by : **Role**

12 items (2 Users, 1 Groups, 5 Service Principals, 2 Unknown, 2 Managed Identities)

Name	Type	Role	Scope	Condition
Contributor				
[illegible]	[illegible]	Contributor	This resource	None
[illegible]	[illegible]	Contributor	Subscription (Inherited)	None
clu-vn	App	Contributor	This resource	None
[illegible]	[illegible]	Contributor	Subscription (Inherited)	None
[illegible]	[illegible]	Contributor	Subscription (Inherited)	None

Private Cluster

- Gives the API server an IP address in the AKS subnet
- Microsoft exposes API server via **Private Link**, connected to **Private Endpoint** in your own network



19a824aa-0796-4738-b3a1-06b7e16c5fa1.privatelink.westeurope.azmk8s.io

Private DNS zone

Search (Ctrl+/)

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems

Settings
Virtual ne
Properties
Locks

Monitoring
Alerts
Metrics

Automation

Record set | Move | Delete zone | Refresh

Essentials

Resource group (Move) : mc_rg-aks_clu-pri_westeurope
Subscription (Move) : Microsoft Azure Sponsorship
Subscription ID : d1d3dadc-bc2a-4495-b8dd-70443d1c70d1
Tags (Edit) : Click here to add tags

You can search for record sets that have been loaded on this page. If you don't see what you're looking for, you can try scrolling to allow more recor

Search record sets

19a824aa-0796-4738-b3a1-06b7e16c5fa1.privatelink.westeurope.azmk8s.io | Virtual network link

Private DNS zone

Search (Ctrl+/)

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems

Settings
Virtual network links

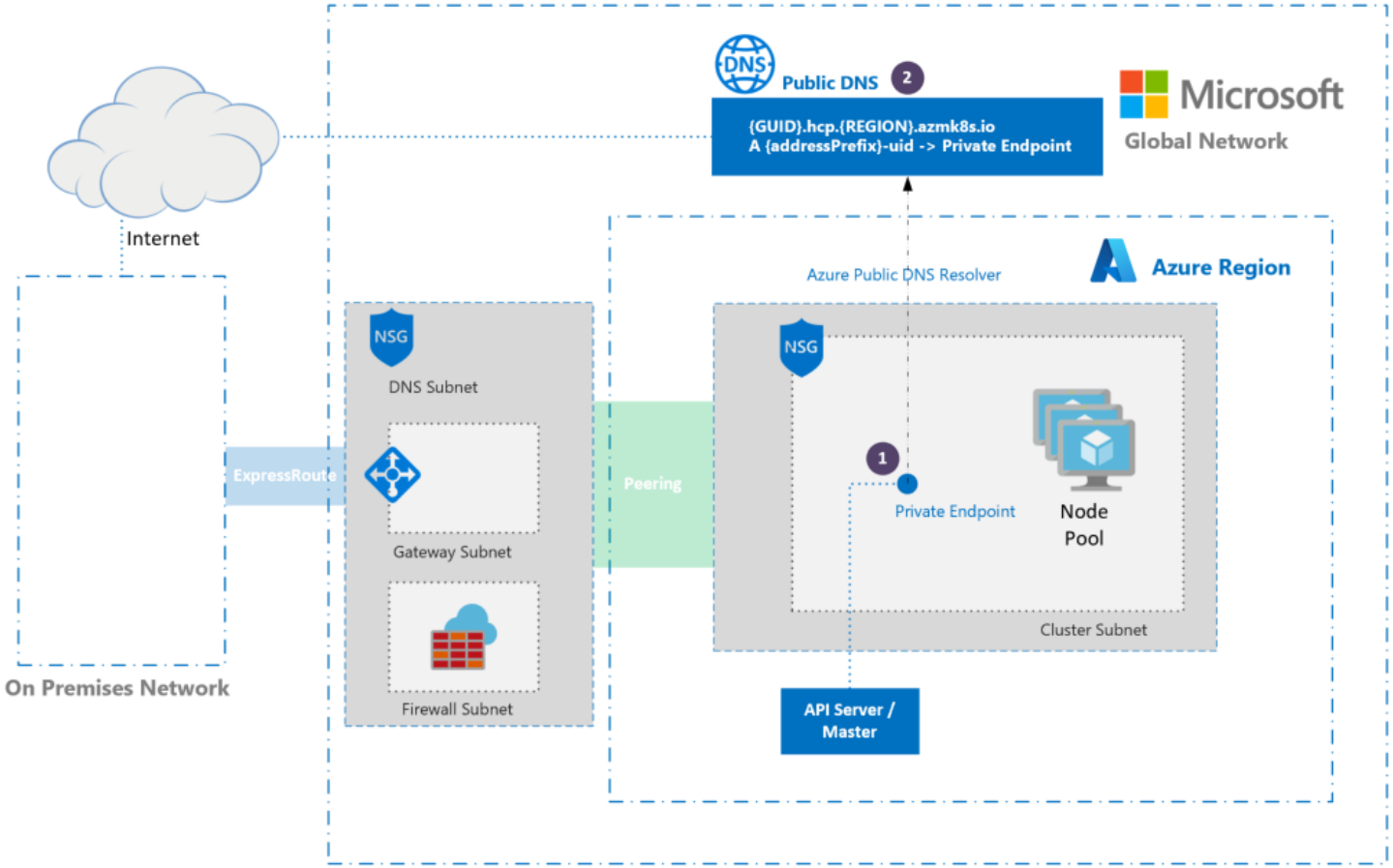
Add | Refresh

Search virtual network links

Link Name	Link status	Virtual network
clu-pri-dns-843e9087	Completed	rg-aks-vnet

Public DNS for Private Cluster

- DNS resolution for private clusters can be complex: see https://docs.microsoft.com/en-us/azure/aks/private-clusters
- Easiest option: use public FQDN



Deploy your cluster

- Deploy with AKS Deploy Helper
  - Managed environment
  - Additional security controls
  - Use one node pool (system and user)
  - Turn off Ingress Controller (defaults to AG with WAF)
  - Specify AAD Group ObjectID
    ⚠ Deploy Helper might fail setting it properly; update Configuration afterwards

Network Policies

Network Policy

Control **ingress** and **egress** for pods in AKS

Two options:

- Azure Network Policies
- Calico Network Policies

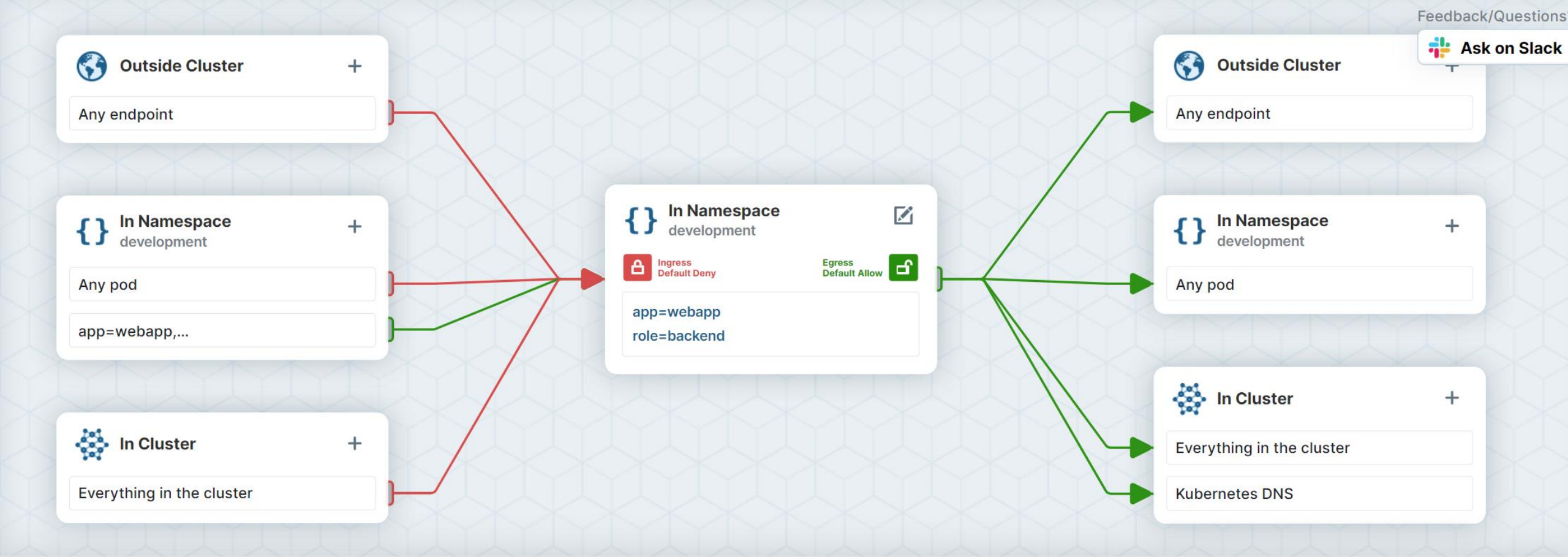
Enable network policy at cluster creation time

Example policy

- In the **development** namespace, allow traffic to pods with **backend** labels from pods with **frontend** labels.

⚠ The **frontend** pods can be in any namespace.

```yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: backend-policy
  namespace: development
spec:
  podSelector:
    matchLabels:
      app: webapp
      role: backend
  ingress:
  - from:
    - namespaceSelector: {}
      podSelector:
        matchLabels:
          app: webapp
          role: frontend
```



Kubernetes Network Policy | Cilium Network Policy

Policy Rating

Download

Share

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: backend-policy
  namespace: development
spec:
  podSelector:
    matchLabels:
      app: webapp
```

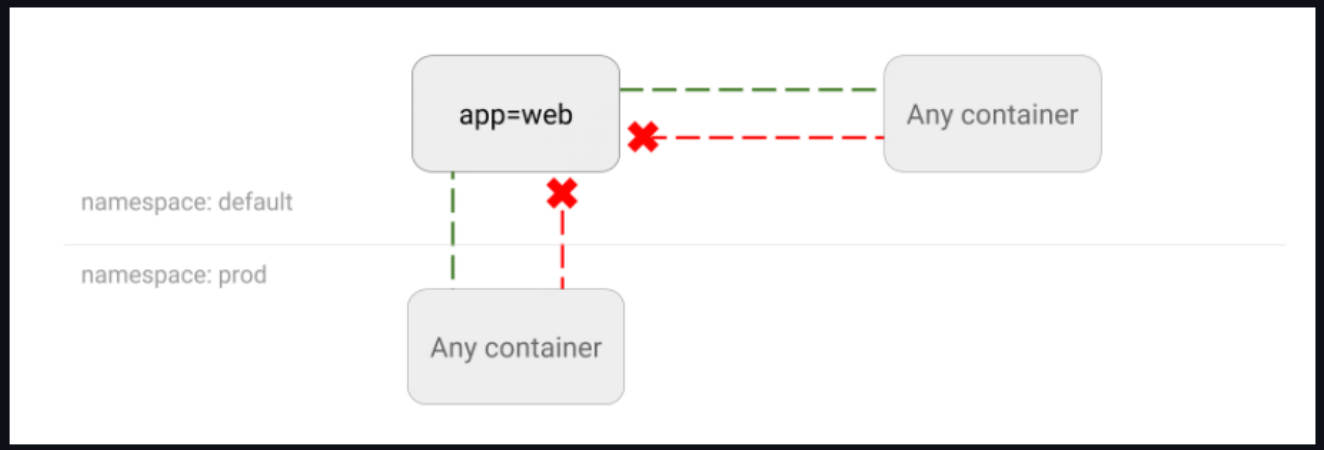
Main tutorial | Flows upload

Welcome to the Network Policy Editor! *βeta*

This tutorial will teach you how to create a network policy using the Editor. It explains basic network policy concepts and guides you through the steps needed to achieve the desired least- privilege security and zero-trust concepts.

Step 1. What pods do you want to secure?

README.md



You can get stuff like this with Network Policies...

Kubernetes Network Policy Recipes

This repository contains various use cases of Kubernetes Network Policies and sample YAML files to leverage in your setup. If you ever wondered how to drop/restrict traffic to applications running on Kubernetes, read on.

https://github.com/ahmetb/kubernetes-network-policy-recipes

Calicoctl

- Separate CLI
- Check version of client and server with **calicoctl version**
- Other uses:
  - Implement Calico network policies (namespace)
  - Implement Calico GlobalNetworkPolicy (independent of namespace)
  - Creating NetworkSets and GlobalNetworkSets

```yaml
apiVersion: projectcalico.org/v3
kind: NetworkPolicy
metadata:
  name: calicopol
  namespace: default
spec:
  selector: app == 'superapi'
  ingress:
  - action: Allow
    protocol: TCP
    source:
      selector: app == 'debug'
    destination:
      ports:
        - 8080
```

Service Mesh

What does a service mesh provide?

Observability

Collecting metrics

Golden signals

Security

Mutual TLS

Reliability

Retries

Timeouts

Load balancing

Traffic shifting

Useful in microservice applications

👍 Services use synchronous communication with HTTP or gRPC

👎 Monoliths or services that communicate via queues/topics

Open Service Mesh

LINKERD

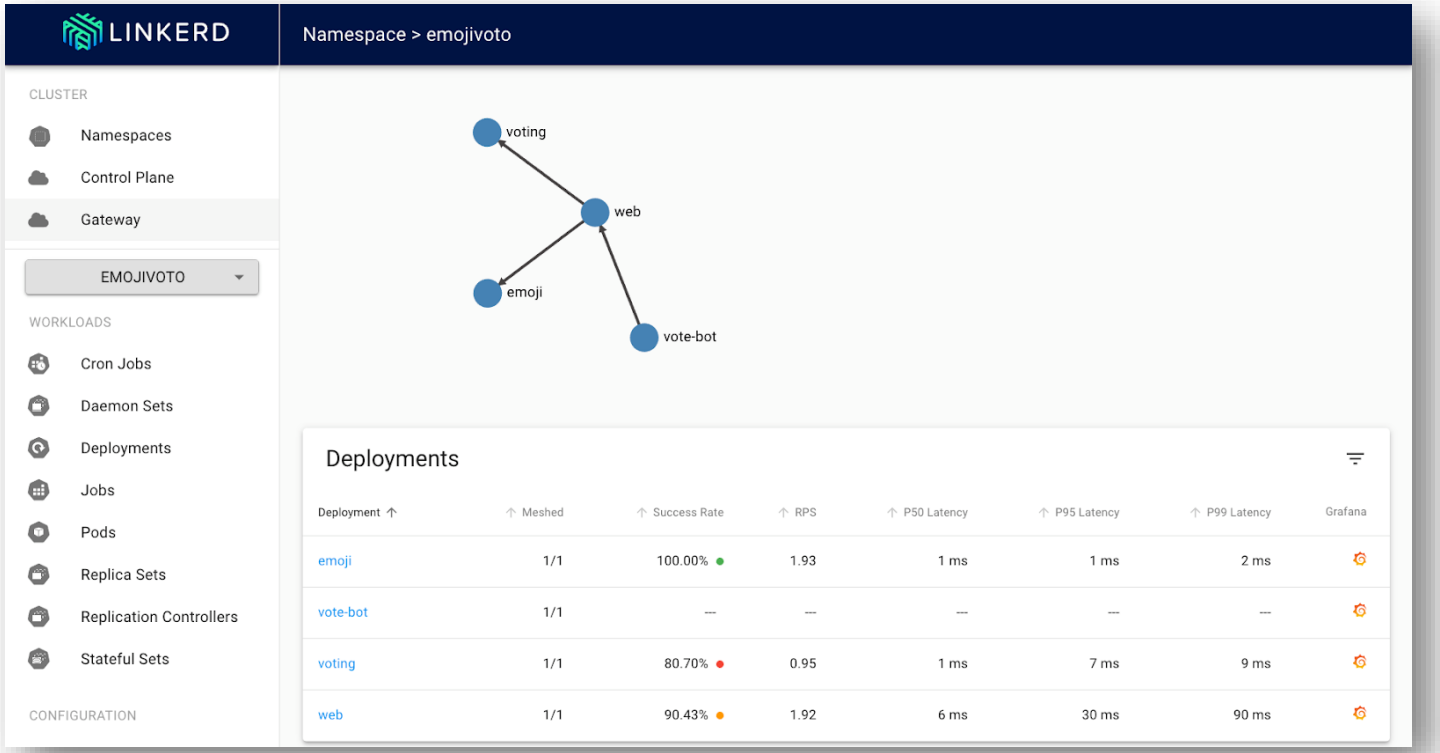
Istio

Consul

Golden Signals

- Latency
- Traffic (RPS)
- Successful responses (SR)

No need to instrument your application to obtain these signals!



Security

- Transparent mutual TLS between services
  - ➔ no configuration required
- Provides both **confidentiality** and **authentication**

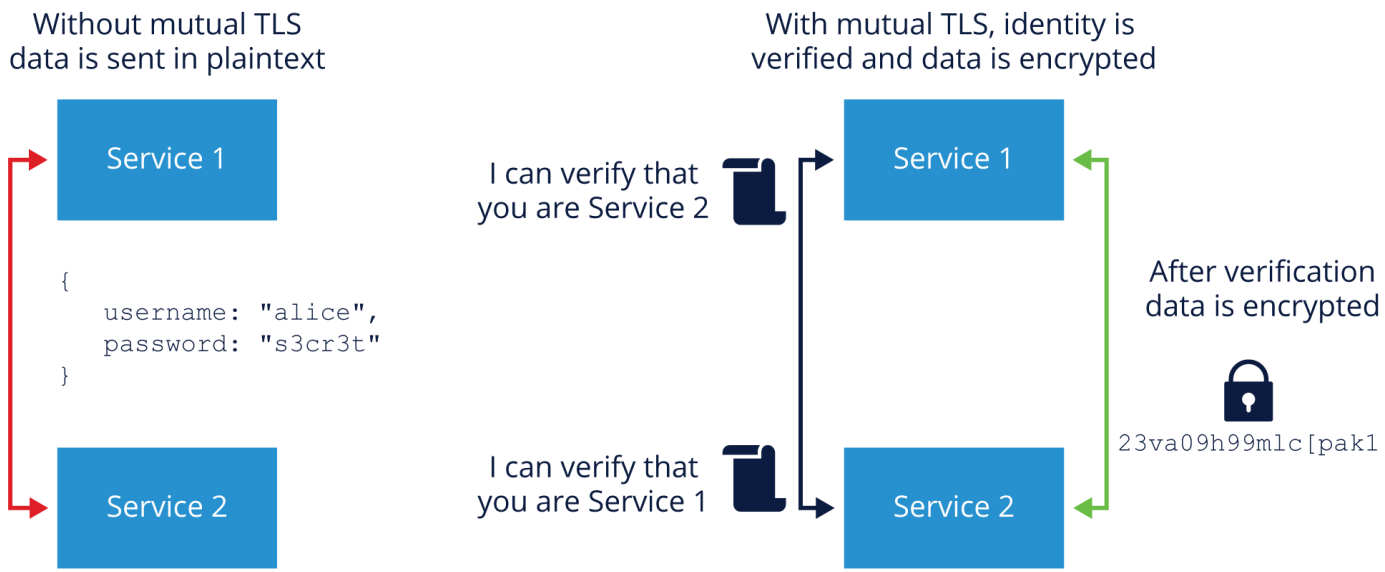


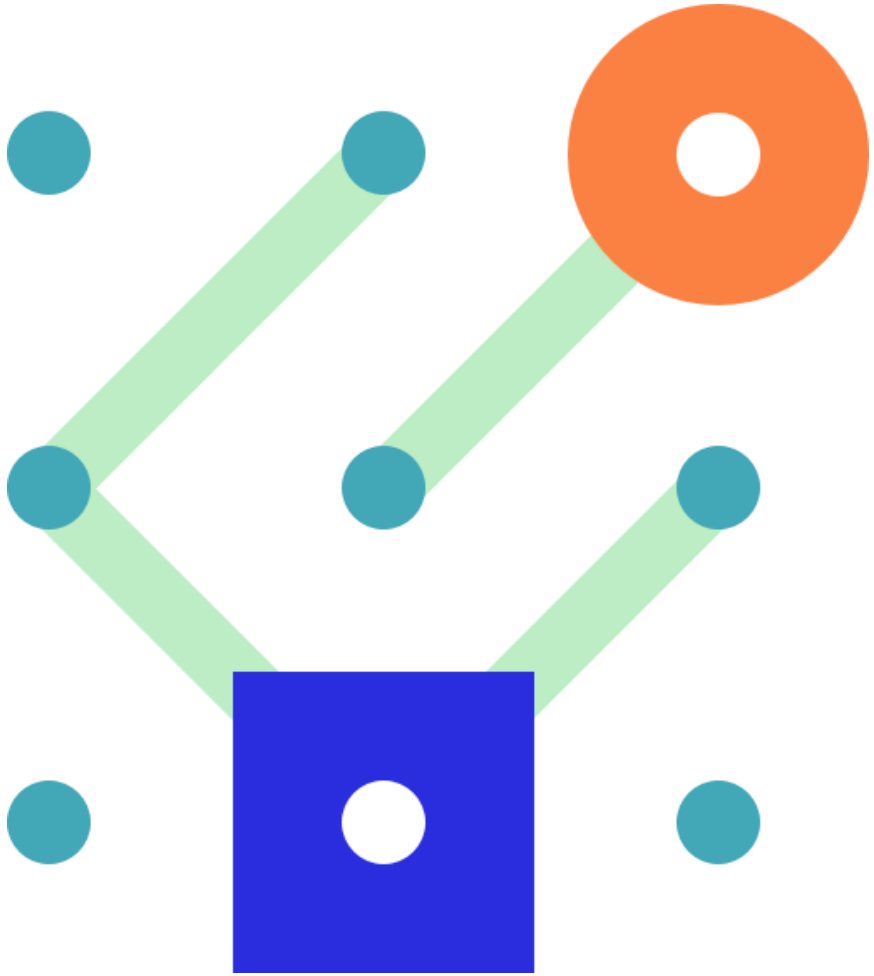
Image source: Linkerd

Reliability

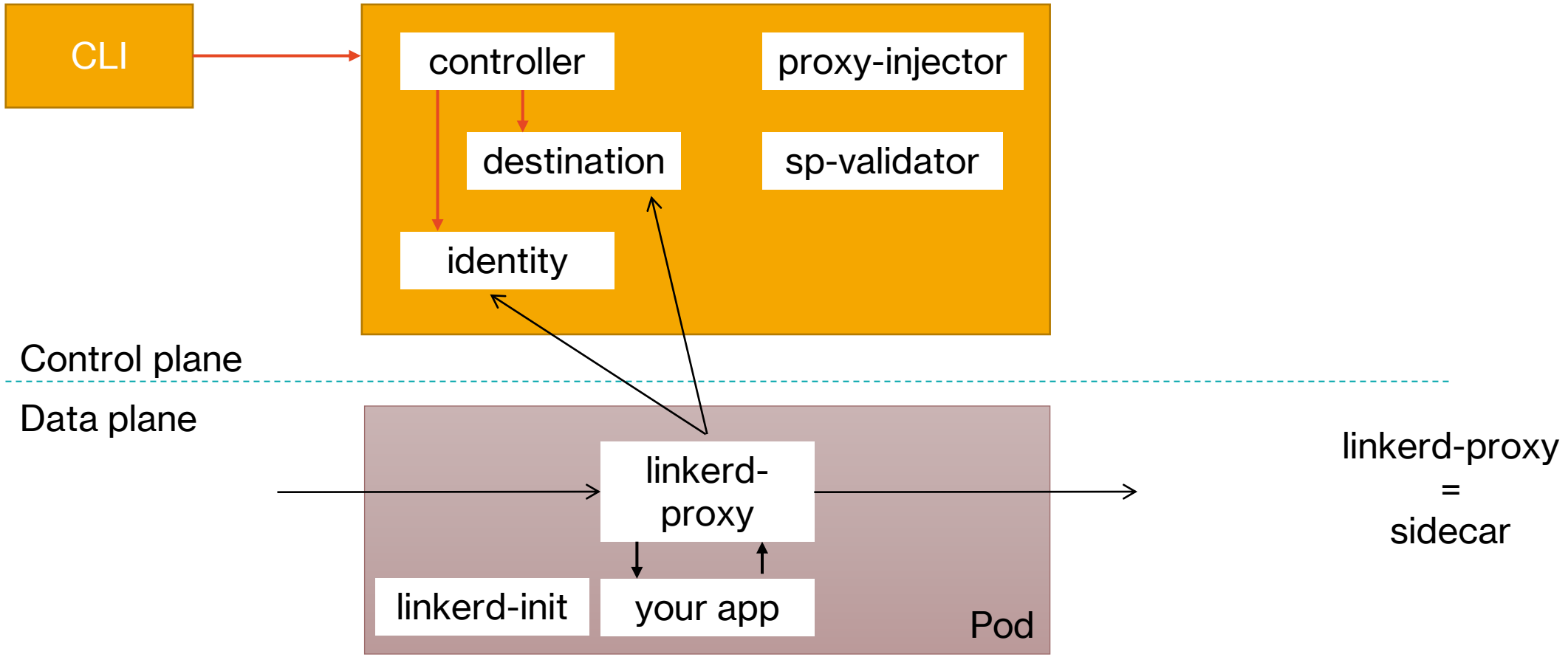
- Retry failed requests
- Configurable timeouts
- Load balancing based on latency
- Traffic shifting

Service Mesh Interface (SMI)

- Standard set of interfaces that all service mesh projects can use
- Specifications:
  - Traffic policy
  - Traffic telemetry
  - Traffic Management
- Linkerd implements some of the above (e.g. TrafficSplit)



Architecture



"Meshing" pods

- To use Linkerd's features, pods need to have the linkerd-proxy sidecar
- Pods with a linkerd-proxy are "meshed" pods
- Add **linkerd.io/inject: enabled** annotation
- Most functionality at the client so ensure clients are "meshed"

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: debug
  labels:
    name: debug
  annotations:
    linkerd.io/inject: enabled
```

Linkerd init container

- Linkerd adds an init container to your pod that modifies **iptables** rules
- This requires **NET_ADMIN** capabilities
  - ➔some clusters prohibit this
- Solution: use the Linkerd CNI plugin
  - ➔ https://linkerd.io/2.11/features/cni

Load Balancing

- Kubernetes Load Balancing
  - Connection level
  - Issue with gRPC which uses HTTP/2; HTTP/2 does request multiplexing over a single connection
- Linkerd Load Balancing
  - Request level based on latency (EWMA)
  - Works well with gRPC out of the box
  - Client-side so client needs to be meshed

Protocol Detection

- Built-in protocol detection
- HTTP/1.1, HTTP/2 or gRPC
  - For these, Linkerd provides full functionality
- What if it is another protocol?
  - Simply proxy the request as raw TCP
  - TLS is also processed as raw TCP
- You can mark ports as **opaque** to tell Linkerd to skip protocol detection

  config.linkerd.io/opaque-ports (see [TCP Proxying and Protocol Detection | Linkerd])

Service Profiles

- Out of the box, **golden metrics** are at the service level
- You can obtain the metrics per **endpoint** of your service
  →via a **Service Profile**
- Service profile is extra config: more YAML!
  →or use CLI: can use OpenAPI/Swagger or protobuf file to learn about endpoints
  →or monitor traffic: **linkerd viz profile**
- They are not only for per route metrics
  - Retries and timeouts
- Optional ⚠

Exercise

- Install Linkerd on AKS
- Install an application with meshed pods

Ingress and Service Mesh

- Linkerd does not come with its own Ingress Controller
- Add your Ingress Controller of choice
  - NGINX, Traefik, ...
- Mesh the Ingress Controller to secure traffic to backend services and to obtain metrics

  → not possible for Ingress solutions with external proxies (e.g., App Gateway)

⚠ Beware of **sticky sessions**

Exercise

- Install meshed Ingress Controller
- Configure the ingress resource

Key Takeaways

Cluster deployment

- Azure CNI
- Azure Firewall for egress control
- Network Policies for east-west traffic control

Service Mesh

- Golden metrics
- Encryption of traffic in the cluster
- Improved load balancing
- Application-level features: retries, timeouts