Microsoft Azure

Azure Kubernetes Service Summit

Network and security best practices

Mohammad Nofal

Cloud Native Apps Architect – Global Black Belt Team – EMEA

@mohmd_nofal

Microsoft Azure

Agenda

- Network Plugins
- Network Policies
- Ingress and Egress traffic
- API Server Visibility Models
- Identity and access management
- Security workloads

What Network Plugin Should I Use?

Microsoft Azure

AKS Network Plugins

- AKS supports 2 network plugins
  - Kubenet
  - Azure CNI
- You can bring your own network Plugin i.e. Cilium, however, this is outside the AKS support scope, and a contract need to be maintained with the 3rd party provider to get support

Kubenet

- Implements a network bridge in each node (cbr0)
- Pods are assigned a logical subnet
- Pod to Pod cross node communication is handled automatically by Azure by implementing a route table, each node will add one route entry to the route table
- Simple and stable
- Pods and Nodes are on different subnets
- Supports Calico Network Policy and Linux Nodes Only (No Windows Support)
- VNET subnet sizing

Number of IPs = No. of Nodes + Node Surge Count + Scale Out Nodes + No. of ILB Services

- Example 10 node cluster, with surge size 2, scale out to 12, and 200 pods, 10 ILB SVCs
  10 + 2 + 2 + 10= **24 IPs or /27 Subnet**

Azure CNI V1

- Implementation of the up-stream Container Network Interface (CNI) specification
- Native to Azure, pods are attached to the virtual network
- Pods and Nodes are on the same subnet
- Pods can assume VM NIC performance, no IP forwarding involved
- Supports Calico and Azure CNI Network Policies
- VNET subnet sizing

Number of IPs = No. of Nodes + Node Surge Count + Scale Out Nodes + No. of ILB Services + (Max No. of Pods Per Node * No. of Nodes)

- Example 10 node cluster, with surge size 2, scale out to 12, and 200 pods, 10 ILB SVCs
  10 + 2 + 2 + 10 + (14 * 30) = **442 IPs or /23 Subnet**

Microsoft Azure

Azure CNI vNext – Dynamic Allocation of IPs (Preview)

- Designed to overcome the number of IPs required for CNI V1 https://docs.microsoft.com/en-us/azure/aks/configure-azure-cni#dynamic-allocation-of-ips-and-enhanced-subnet-support-preview
- Dynamic IP allocation for pods, only allocates IPs when the pods are running
- Nodes and Pods can live in different VNET subnets and can be scaled individually
  - Allows to have a different Network security policies for nodes and pods, i.e. Nodes are allowed to access update.ubuntu.com but pods can't
- Multiple subnets per cluster support
- Multiple clusters can share the same subnet within a VNET
- Only Linux clusters are supported for now!

Azure CNI vNext (Preview) VNET Forecasting

- **Number of IPs**
  - **Nodes Subnet**: No. of Nodes + Node Surge Count + Scale Out Nodes
  - **Pods Subnet**: No. of Pods + ILB Services
  - Example 10 node cluster, with surge size 2, scale out to 12, and 200 pods, 10 ILB SVCs
    - Nodes Subnet: 10+2+2=14 or /27 Subnet (taking into account the reserved IPs)
    - Pods Subnet: 200+10=210 or /24 Subnet

- Expected to GA towards the end of H2CY21

Microsoft Azure

3rd Party Network Plugins

- For example Calico and Cilium
- Both deliver good performance and great integration with network policies (i.e. Rules based on FQDNs)
- Both deliver enterprise solutions with great features such as enhanced observability
- Draw back
  - Installation is not trivial, need to load the Plugin after cluster bootstraps which means you need to reboot the nodes or the pods after installing the new CNI
  - Support on your own unless you buy it

Network Plugins Comparison

	Azure CNI and CNI vNext	Kubenet	Others (Cilium/Calico)
When to use	Always + When VM like Performance is required **Unless** you're constrained on IPs	When you are constrained on IPs, network team only provide small subnets	Looking for specific features which aren't provided by the managed plugins
Performance	VM Like	Good Performance (IP Forwarding has an overhead)	Good Performance
NW Policy Support	Calico and Azure Network Policy	Calico	Native
Support	Best	Best	On your own or you buy it
Integrations with Azure Services	Full	Lags behind, Partially Full	Test
Max Cluster Size	As big as the subnet	400 Nodes (Route Table Limits)	Depends

Microsoft Azure

Network Design Recommendations

- Pod CIDR
  - CNI: Part of the VNET
  - Kubenet: logical, ensure it doesn’t overlap with connected networks, can overlap with other clusters so long no pod to pod cross cluster communication is needed
- Docker bridge (no longer required in 1.19+)
  - Shouldn’t overlap with connected networks, can be reused across clusters
- Service CIDR
  - Shouldn’t overlap with connected networks, can be reused across clusters
- Network security groups: shouldn’t modify auto-provisioned ones, need more control assign NSGs to the subnet
- CNI and Kubenet work perfectly in a hub and spoke model, with design differences such as the route table

Network Policies

Microsoft Azure

Network Policies in AKS

- AKS supports Azure Network Policy and Calico
- Both Network Policies allow you to create ingress and egress policies based on labels or IP CIDRs
- Network policies can only be enabled at cluster provisioning time, always enable network policies even if you’re not planning to use them at the beginning

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

Microsoft Azure

Calico Extra Features

- Calico Supports
    - Global Network Policies: apply to all namespaces (great for default rules)
    - Global Network Set: group network CIDRs under a set that can be referenced in a policy
    - Host Endpoint: apply policies to the host Network Interface
    - Ordering of rules
- **Calicoctl** is required to apply such polices
    - Calico API server is in “tech preview” now to allow to apply such policies using kubectl (unlocks gitops scenarios)

Calico Global Network Policy Example

```
$ calicoctl apply -f calico_globalnwpolicy.yaml

calico_globalnwpolicy.yaml
apiVersion: projectcalico.org/v3
kind: GlobalNetworkPolicy
metadata:
name: deny-circle-blue
spec:
selector: color == 'red'
ingress:
- action: Deny
protocol: TCP
source:
selector: color == 'blue'
namespaceSelector: shape == 'circle'
```

3rd Party Network Policies

- Cilium, Calico Enterprise, and Some Service Meshes allow you to create L7 network policies or FQDN based policies

```
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
name: "fqdn"
spec:
endpointSelector:
matchLabels:
org: prod
egress:
- toFQDNs:
  - matchPattern: "*.ubuntu.com"
- toEndpoints:
  - matchLabels:
    "k8s:io.kubernetes.pod.namespace": kube-system
    "k8s:k8s-app": kube-dns
  toPorts:
  - ports:
  - port: "53"
    protocol: ANY
  rules:
   dns:
   - matchPattern: "*"
```

Microsoft Azure

Network Plugins Comparison

	Azure CNI Network Policy	Calico	3rd Party
What is it	Developed by Azure Networking team	Developed by Tigera team and its open source	Any 3rd party i.e. Cilium
Supported Network Policies	Azure CNI	Azure CNI, Kubenet, 3rd party distributions	Any
Support	Supported by Azure NW Engineering	Community support	On your own or you buy it
Advantages	One stop shop for support	Most wildly used network policy with the largest community and ships with extra neat features	Varies i.e. L7 network policies

Controlling Egress and Ingress Traffic

Microsoft Azure

Outbound-type=userDefinedRouting

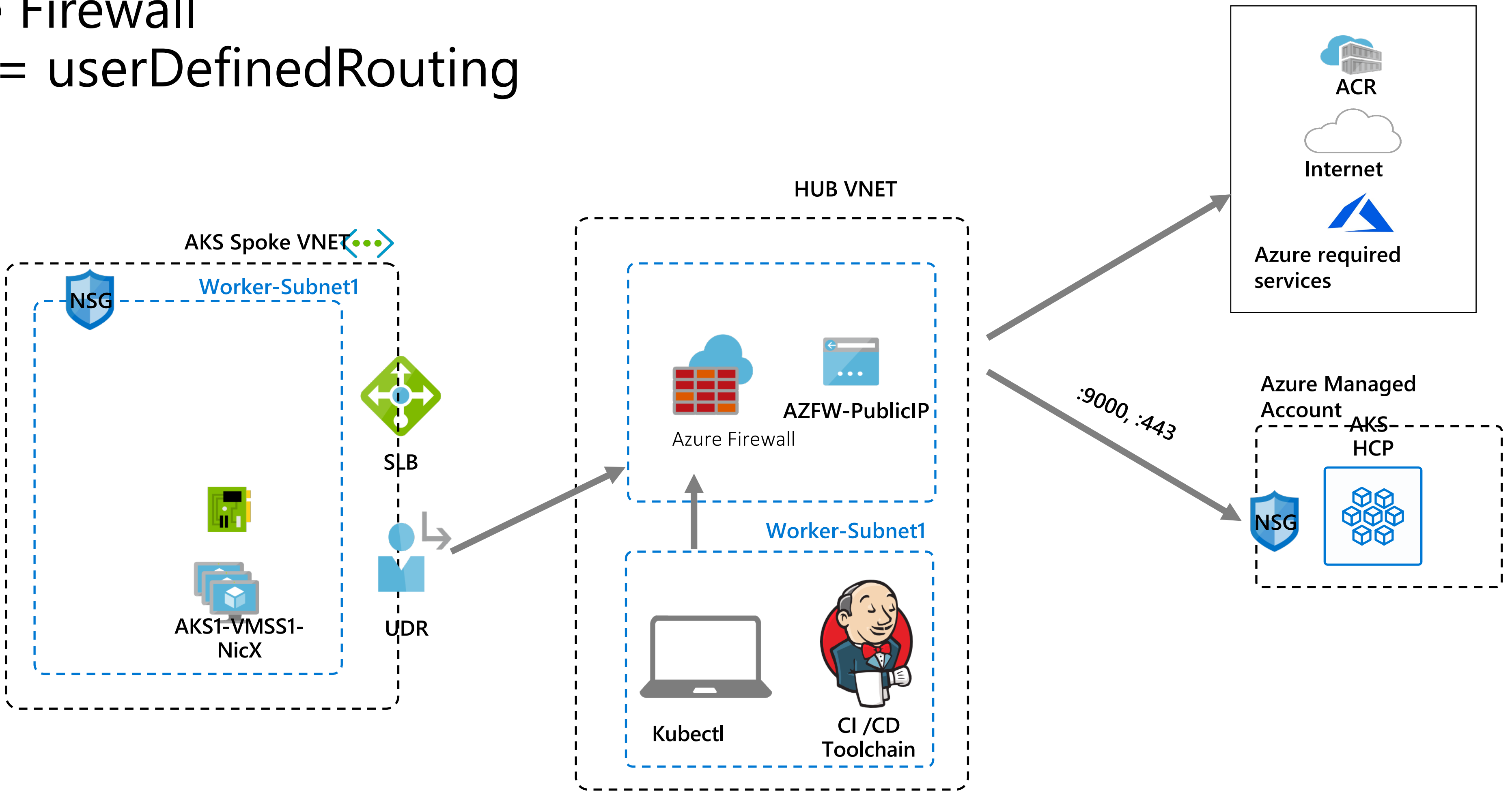
- AKS now allows to configure the outbound-type for the Cluster egress traffic

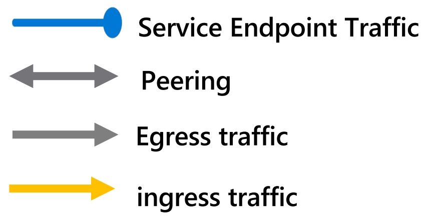
- The parameter will allow you to choose between 'Loadbalancer' current default behavior, **or**

- Outbound-type=userDefinedRouting , in this model the assumption is that you have a NVA/FW in place which you can route all the traffic to, and this NVA will handle the egress to the internet
- Works with Azure CNI and Kubenet

  https://docs.microsoft.com/en-us/azure/aks/egress-outboundtype

Egress to Azure Firewall
OutboundType= userDefinedRouting





Microsoft Azure

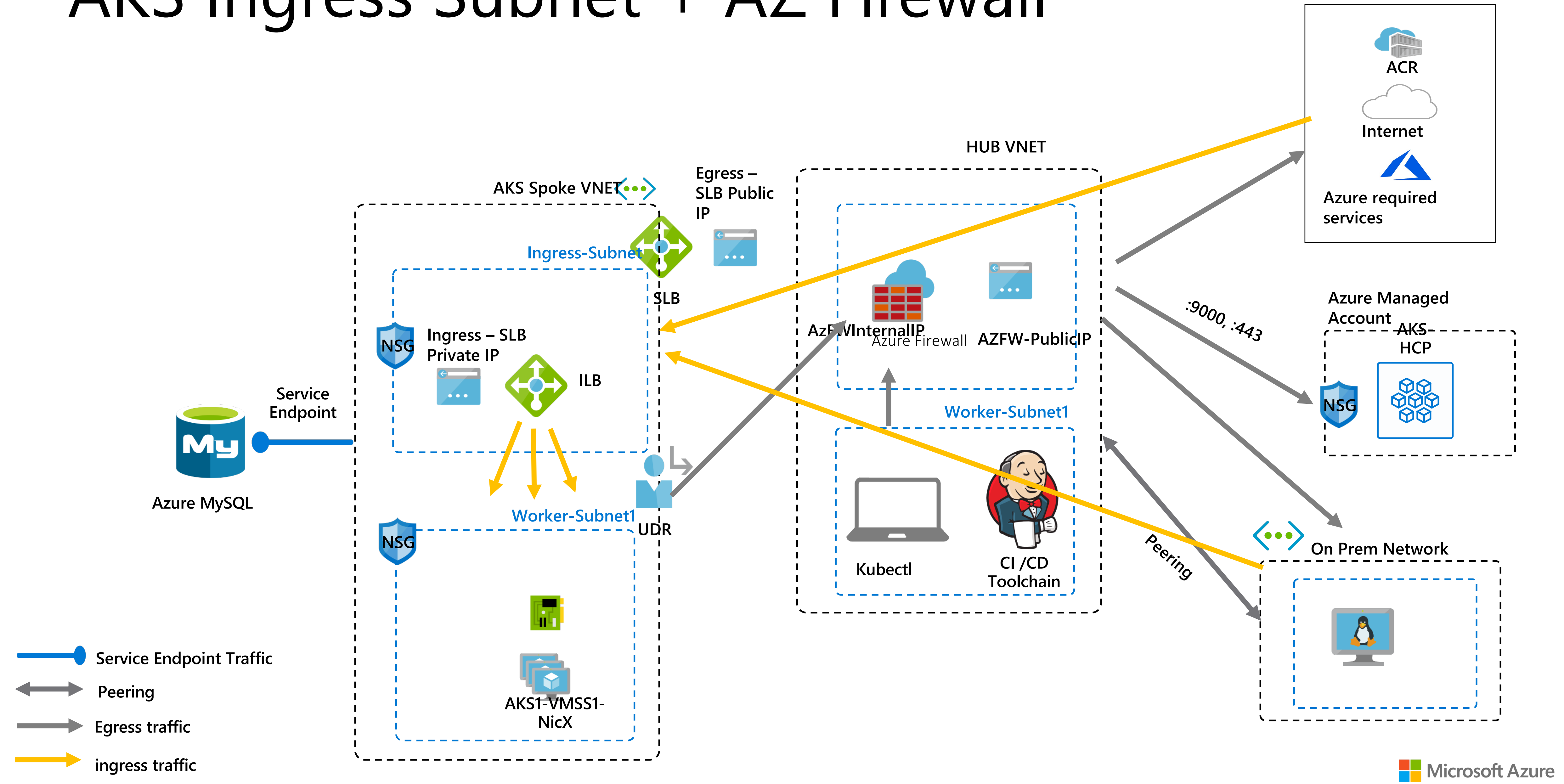
Dedicated Ingress subnet

- Dedicating a subnet for the ingress traffic provides clear seperation of concerns between ingress and egress traffic

```
apiVersion: v1
kind: Service
metadata:
name: internal-app
annotations:
service.beta.kubernetes.io/azure-load-balancer-internal: "true"
service.beta.kubernetes.io/azure-load-balancer-internal-subnet: "apps-subnet"
spec:
type: LoadBalancer
ports:
- port: 80
selector:
app: internal-app
```

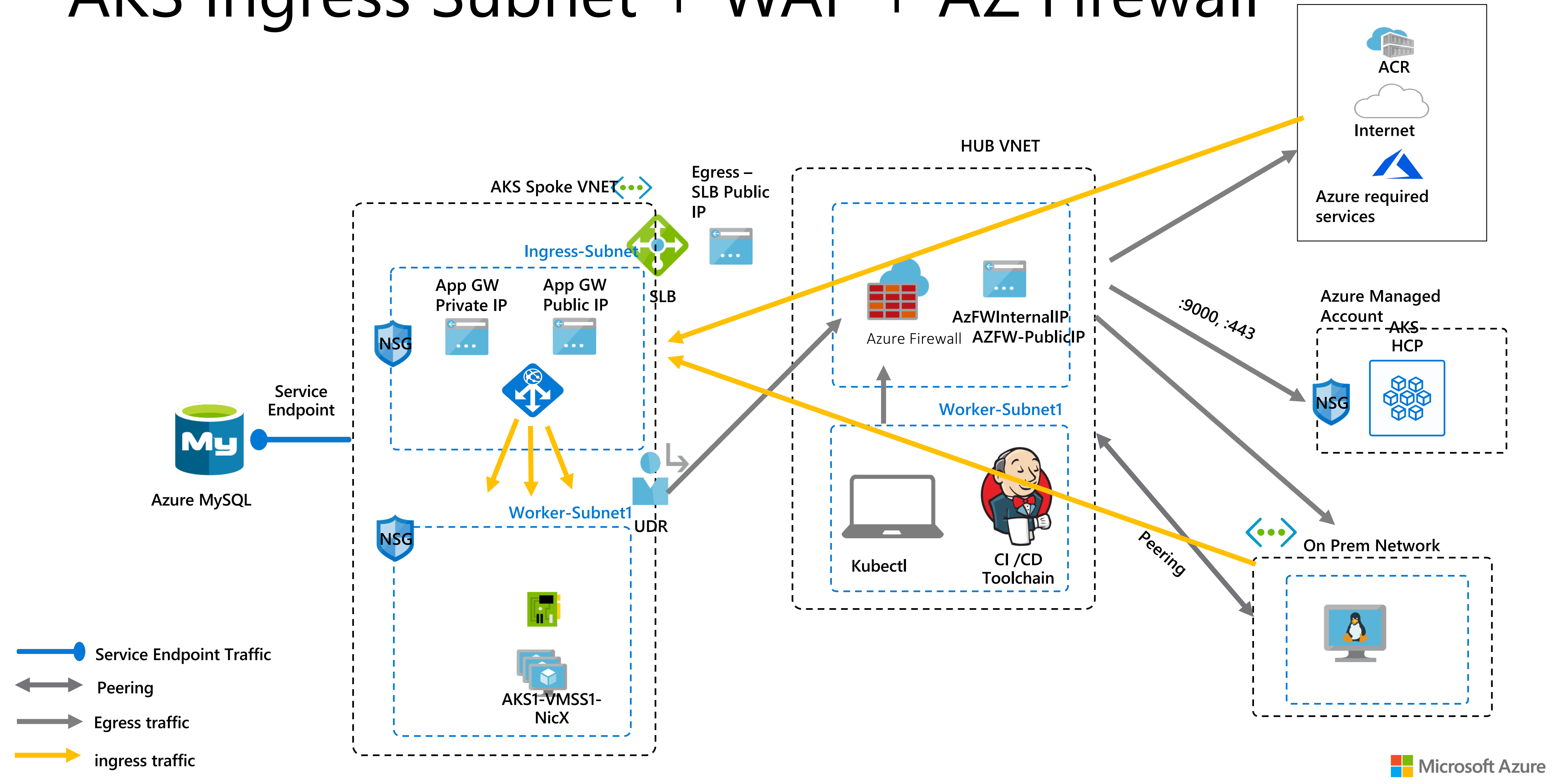
- ty rules for
- rewall) can
- and you gain

AKS Ingress Subnet + AZ Firewall



Microsoft Azure

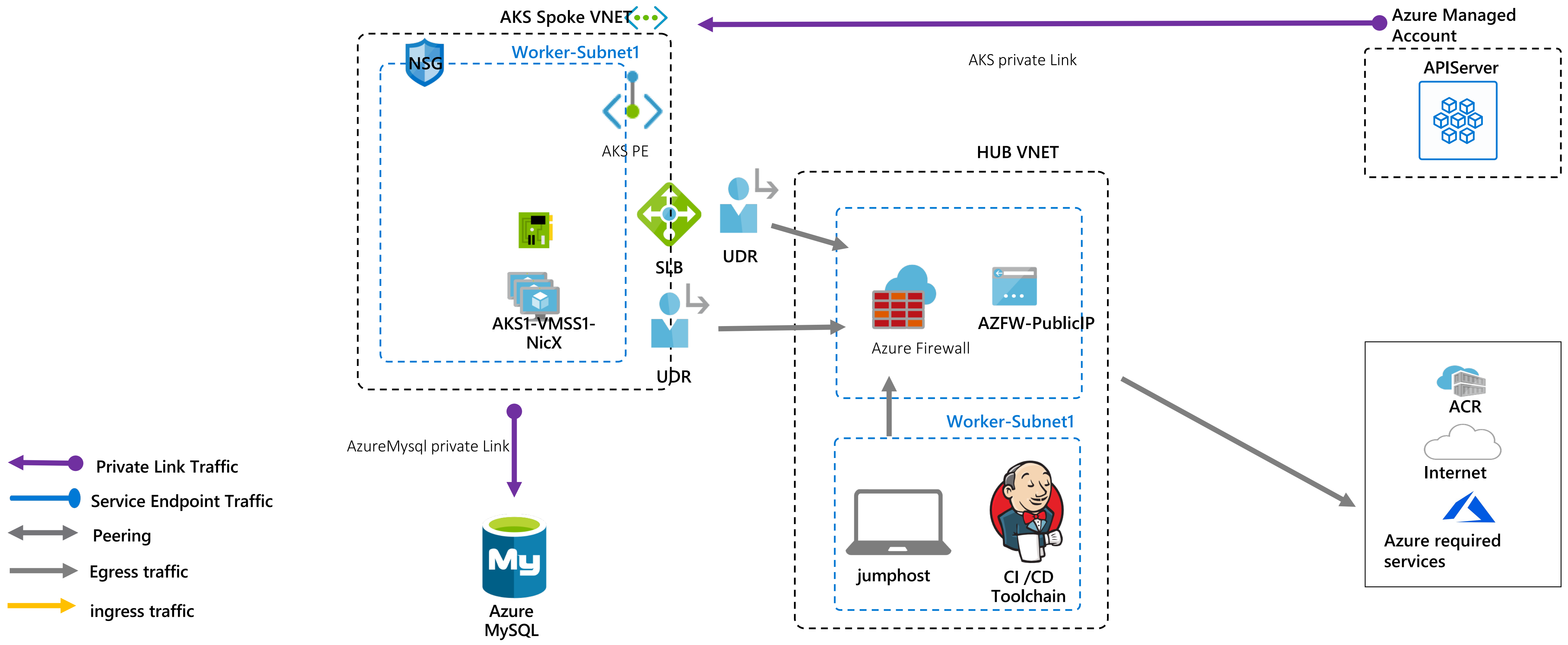
AKS Ingress Subnet + WAF + AZ Firewall



AKS Private Clusters

Microsoft Azure

AKS Private Cluster

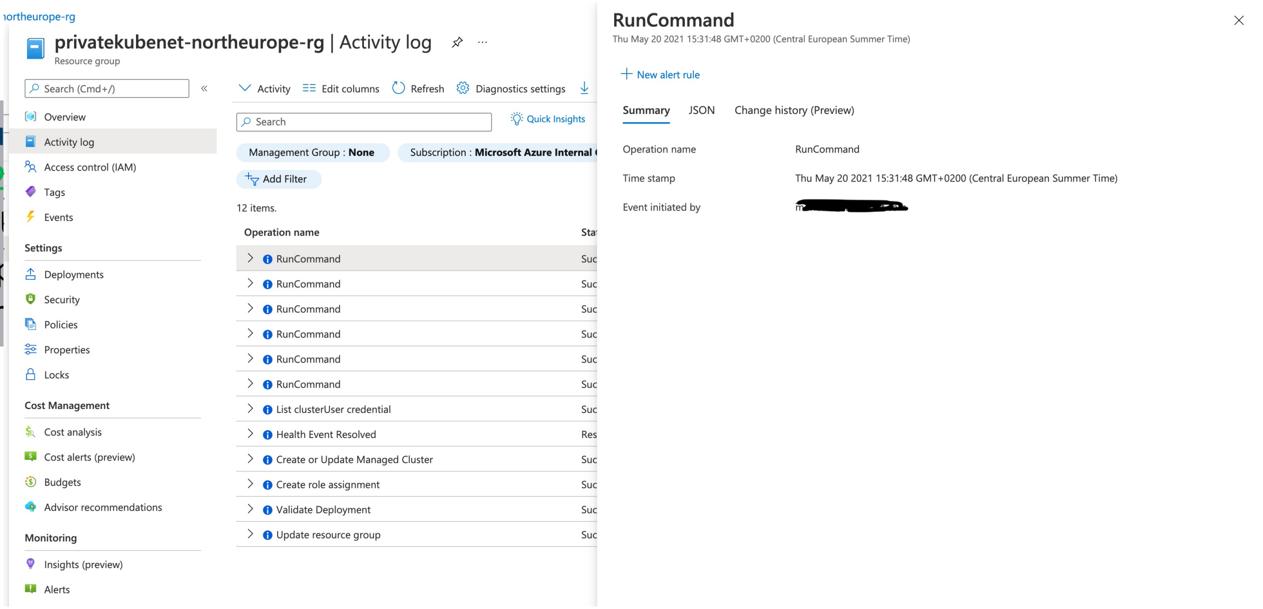


Microsoft Azure

AKS Run Command (Preview) – Manage Private Clusters

- Solves the need to have a jump box to manage or interact with private clusters
- Manage private clusters without the need for a Jumpbox
- Your commands will run inside your cluster as "kubectl" commands, you can also pass helm, and kustomize commands, more to come…
- Calls are audited and logged
- Work on AAD integration is undergoing

```
$ az aks command invoke -g $RG -n $AKS_CLUSTER_NAME -c "ku
command started at 2021-05-20 13:31:33+00:00, finished at 202
NAME                                      READY   STATU
command-ab5d3c6add224c43aa9ed7c41a3ccbb9  0/1     Comp
command-be25b67036874c97adaeaaebca25a64f  1/1     Runn
```



ıortheurope-rg

privatekubenet-northeurope-rg | Activity log

Resource group

Management Group : **None** Subscription : **Microsoft Azure Internal**

12 items.

Operation name	Sta
RunCommand	Suc
RunCommand	Suc
RunCommand	Suc
RunCommand	Suc
RunCommand	Suc
RunCommand	Suc
List clusterUser credential	Suc
Health Event Resolved	Res
Create or Update Managed Cluster	Suc
Create role assignment	Suc
Validate Deployment	Suc
Update resource group	Suc

RunCommand

Thu May 20 2021 15:31:48 GMT+0200 (Central European Summer Time)

Operation name	RunCommand
Time stamp	Thu May 20 2021 15:31:48 GMT+0200 (Central European Summer Time)
Event initiated by	[redacted]

Microsoft Azure

Private AKS DNS Zone Options

- **System** default, AKS creates the private DNS zone on your behalf in the nodes RG
  - Need to be linked with the HUB (if exist) to manage resolution
- **Bring your own**
  - Approved format: privatelink.<region>.azmk8s.io
  - You pass the **resource ID** and the **FQDN subdomain**
  - AKS will update the entry (FQDN subdomain) in the private zone with the API server private IP
  - The zone can be pre-linked with the HUB in order to ensure smooth resolution
- **None**
  - No zone will be created, the AKS API server FQDN will be created inside the nodes host file and coreDNS
  - You will need to bring your own DNS servers and ensure creating the entries there
- **Public Zone for Private clusters [in progress]** https://github.com/Azure/AKS/issues/2176
  - A public domain which resolves to a private IP
  - Simplifies the whole setup, there will be no need to create DNS infrastructure or deal with the complex linking strategies

Closing thoughts on private clusters

- Private clusters are great at hiding your API server from the public internet, however, introduces its own challenges with having to design all the surrounding services to be privately connected to your cluster
- AKS Public API server IP can be secured by using **Authorized IP Ranges** to block all unwanted traffic, which is easier to manage
  - can be added at any stage of the cluster lifecycle

```
az aks create \
 --resource-group myResourceGroup
 --name myAKSCluster \
 --node-count 1 \
--api-server-authorized-ip-ranges 73.140.245.0/24

Or

az aks update \
 --resource-group myResourceGroup \
--name myAKSCluster \
--api-server-authorized-ip-ranges 73.140.245.0/24
```

Microsoft Azure

Identity and Access Management

Microsoft Azure

Service Principles in AKS

- AKS cluster requires an identity to create resources i.e. LBs and Disks
- Service Principle is the original AKS identity
- This approach has limitations
  - Requires rotation
  - A security flow

```
root@aks-nodepool1-24959813-vmss000000:/# cat /etc/kubernetes/azure.json
{
    "cloud": "AzurePublicCloud",
    "tenantId": "72f988bf-86f1-41af-91ab-[redacted]",
    "subscriptionId": "2db66428-abf9-440c-[redacted]",
    "aadClientId": "b1c7bfef-b0c3-4051-[redacted]",
    "aadClientSecret": "Ck5Hv-Y0Cc7jP4[redacted]",
```

Enter MSI (Managed Service Identity) in AKS

```
root@aks-nodepool1-20930433-vmss000000:/# cat /etc/kubernetes/azure.json
{
"cloud": "AzurePublicCloud",
"tenantId": ”XXXXXXXX",
"subscriptionId": " XXXXXXXX ",
"aadClientId": "msi",
"aadClientSecret": "msi",
"resourceGroup": "MC_storage-westeurope_storage-cluster_westeurope",
"location": "westeurope",
"vmType": "vmss",
"subnetName": "storage-subnet",
"securityGroupName": "aks-agentpool-20930433-nsg",
"vnetName": "storage-vnet",
"vnetResourceGroup": "storage-westeurope",
"routeTableName": "aks-agentpool-20930433-routetable",
...
"userAssignedIdentityID": "c3a2ccca-84fa-4102-ba63-ec991869d94e",
..
}
```

Provisioning a managed identity cluster

```
# Create the identity
$ az identity create --name $IDENTITY_NAME --resource-group $RG
$ IDENTITY_ID=$(az identity show --name $IDENTITY_NAME --resource-group $RG --query id -o tsv)
$ IDENTITY_CLIENT_ID=$(az identity show --name $IDENTITY_NAME --resource-group $RG --query
clientId -o tsv)
# Assign MSI Permission to the VNET and RG (can be more restrictive)
$ az role assignment create --assignee $IDENTITY_CLIENT_ID --scope $RG_ID --role Contributor
# Assign MSI Permission to the VNET (can be more restrictive)
$ az role assignment create --assignee $IDENTITY_CLIENT_ID --scope $VNETID --role Contributor
# create the cluster
$ az aks create \
-g $RG \
-n $AKS_CLUSTER_NAME \
-l $LOCATION \
--enable-managed-identity \
--assign-identity $IDENTITY_ID \
--assign-kubelet-identity $IDENTITY_ID ##PREVIEW
```

User Identities

- Use Azure AAD integration for users (admins, devs, ops, etc…)
- For CICD systems
  - Use service accounts **OR**
  - Use “**kubelogin**” which supports non-interactive login for pipelines (CICD) - (recommended) https://github.com/Azure/kubelogin
- Managing RBAC in K8s
  - Today customer is required to pull the (--admin-credentials) to implement RBAC in the cluster, which has a flow that this identity is root on the cluster
  - Or use Azure RBAC for Kubernetes Authorization (recommended)

Azure RBAC for Kubernetes Authorization

- Allows you to manage RBAC for Kubernetes using Azure RBAC or natively using the Kubernetes RBAC API
- Covers Azure AAD users/principles only, k8s service accounts are managed by Kubernets
- Has built in roles i.e. Cluster Admin, Reader, Writer, which can be applied on a namespace or a cluster level. You can create custom roles too.

```
#Provision a cluster and enable the feature or update an existing cluster
$ az aks create -g MyResourceGroup -n MyManagedCluster --enable-aad --enable-azure-rbac
or
$ az aks update -g myResourceGroup -n myAKSCluster --enable-azure-rbac

#Assign reader role on "testnamespace" for a given user
az role assignment create --role "Azure Kubernetes Service RBAC Reader" --assignee <AAD-ENTITY-ID> --scope
$AKS_ID/namespaces/testnamespace
```

Disable local accounts [Preview]

- Currently AKS deploys clusters (AAD or non-AAD) with local accounts enabled
- Customer gain access to this by
  $az aks get-credentials -n $AKS_CLUSTER_NAME -g $RG –admin
- This account is a root level account and overrides RBAC and AAD, in order for customers to be able to configure RBAC policies inside their clusters
- With the introduction of “Azure RBAC for Kubernetes Authroziation” there is no need for root access to configure RBAC

```
$ az feature register --namespace "Microsoft.ContainerService" --name "DisableLocalAccountsPreview"
$ az provider register --namespace Microsoft.ContainerService

$ az aks create -g <resource-group> -n <cluster-name> \
--enable-aad \
--aad-admin-group-object-ids <aad-group-id> \
--disable-local
```

Microsoft Azure

Pod Identity

- An open source project to assign AAD identities to Pods so they can access Azure Resources such as KeyVault, SQL Server, Storage Accounts, etc…
- Available as an AKS plugin in Preview

```
$ az aks update -g $MY_RESOURCE_GROUP -n $MY_CLUSTER --enable-pod-identity --network-plugin azure
```

- You can install it on your own too
  https://azure.github.io/aad-pod-identity/docs/getting-started/
- Recommended setup with Azure CNI Network Plugin
- If you're to deploy the plugin or install the tool your self on Kubenet cluters, then some extra work is required to ensure security
  - Kubenet is suspectable to ARP spoofing, you need to disable NET_RAW capabilities on your pods to prevent this.
    https://docs.microsoft.com/en-us/azure/aks/use-azure-ad-pod-identity#mitigation

Microsoft Azure

Pod Identity Cont.

- Comes with 2 controllers
  - Managed Identity Controller (MIC) – watches for changes to pods and add/delete identities as needed – Not used in the plugin
  - Node Managed Identity (NMI) – requests the identity token on behalf of the pods by intercepting the instance metadata service (IMDS) traffic on the nodes (iptables rule) – Only one installed by the plugin or “Managed” mode
- The plugin assigns pods “User Assigned Managed Identities” to ensure faster provisioning and better responsiveness when we scale out the nodes

Microservice Identities - mTLS

- Customers require service to service communication to be encrypted and authenticated
- Customers can implement this in code today, but its hard to maintain
- Enter Service Mesh
  - Secure service to service communication – mTLS
  - Advanced traffic routing (shifting, A/B testing, etc…)
  - Observability
- Popular Service Mesh solutions
  - Istio
  - Linkerd
  - Consul by hashicorp

Open Service Mesh

- An open source project based on the Service Mesh Interface specification
- Provides the most popular Service Mesh features (routing, mTLS, observability)
- The only Service Mesh solution which will be managed in AKS
- Currently managed plugin is in preview in AKS

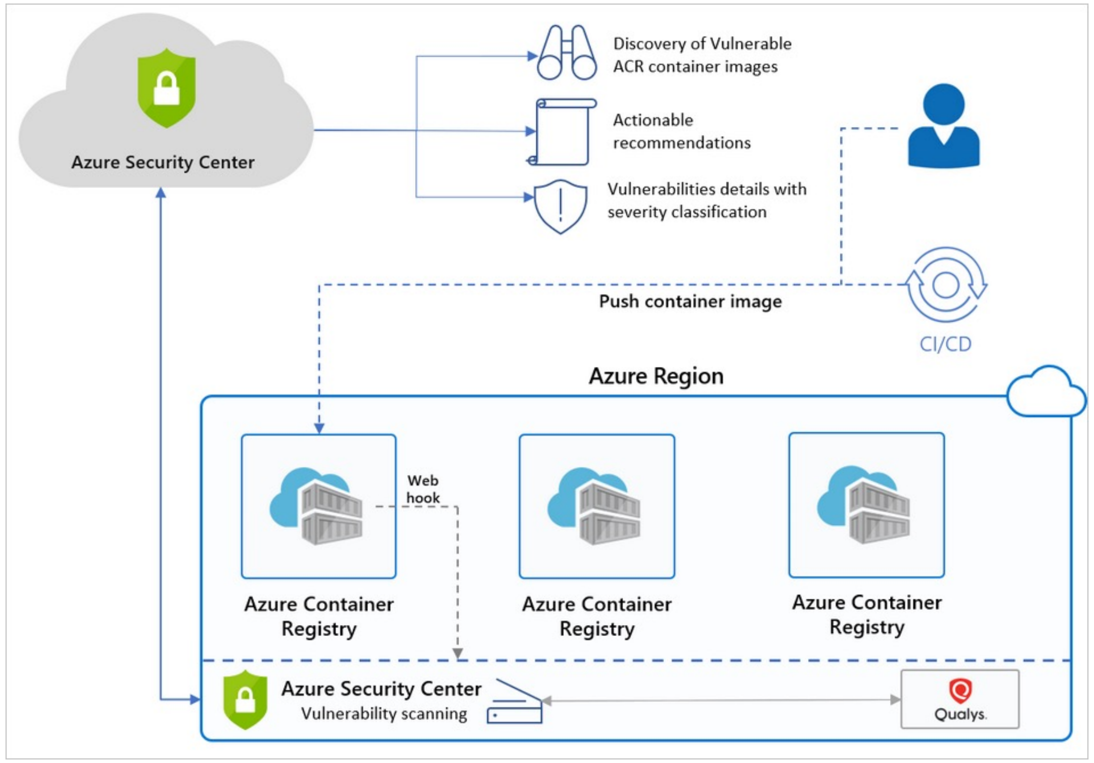
```
az aks create \
-n osm-addon-cluster \
-g <myosmaksgroup>\
--node-osdisk-type Ephemeral \
--network-plugin azure \
--enable-managed-identity -a open-service-mesh
```

Securing Workloads

Microsoft Azure

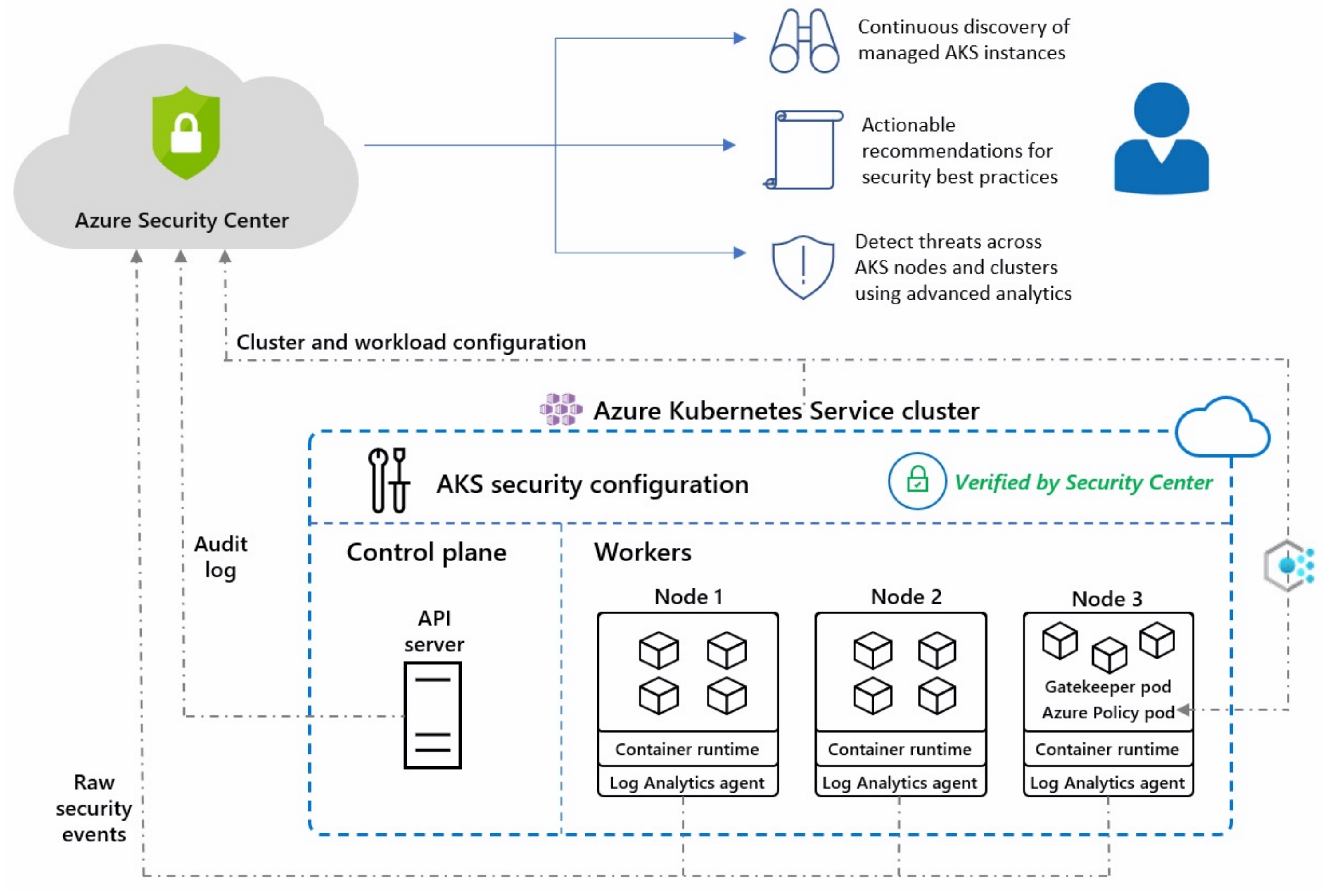
Azure Defender for Container Registries

- A tool powered by Qualys to scan container images for vulnerabilities
- Provides actionable recommendations for each vulnerability
- Scans images on Push, recently pulled, or on Import actions



Microsoft Azure

Scanning Containers – Azure Defender for Kubernetes


Azure Security Center
Continuous discovery of managed AKS instances
Actionable recommendations for security best practices
Detect threats across AKS nodes and clusters using advanced analytics
Cluster and workload configuration
Azure Kubernetes Service cluster
AKS security configuration
Verified by Security Center
Audit log
Control plane
API server
Workers
Node 1
Node 2
Node 3
Gatekeeper pod
Azure Policy pod
Container runtime
Log Analytics agent
Raw security events

Microsoft Azure

Azure Defender for AKS – Current State

- Azure security center relies on the Azure Log Analytics agent (don’t confuse this with the Azure Monitor Agent)
  - Currently it isn’t possible/not recommended to install the agent as a custom resource extension on the virtual machine scaling sets
  - It works on Virtual Machine Availability Sets, however, its no longer recommended to use VMAS with AKS
- Azure Defender team is already working on providing a native Kubernetes deployment for the Defender addon, keep an eye for new announcements

3rd party container scanning tools

- The container security ecosystem is maturing fast in the market
- Great tools like Aqua, Prisma Cloud, sysdig, neuvector, and others …
- All available on Azure marketplace
- All in one tool to provide image scanning, container and host run time security, network policies, pod policies, inventory, etc…
- Providers comprehensive security coverage, and can help with enforcing security defaults

Azure Policy for Kubernetes

- Uses Gatekeeper v3, which is a admission controller webhook for Open Policy Agent(OPA)
- Helps to apply enforcements and safeguards on your AKS, AKS Engine, and Arc connected k8s clusters
- Currently Supports built in policy only, soon custom policies, watchout for related announcement
- Can't co-exist with Pod Security Policy
- The policy language follows the existing Azure Policy definitions

Resources

- Demos, slides, and videos can be found here https://github.com/mohmdnofal/aks-best-practices
- Familiarize your self with the below
  - https://docs.microsoft.com/en-us/azure/aks
  - https://docs.microsoft.com/en-us/azure/aks/best-practices
- Azure CSI secret store driver https://docs.microsoft.com/en-us/azure/aks/csi-secrets-store-driver

Microsoft Azure

Thank you!

Microsoft Azure