Dutch Microsoft & Security Meetup

Wie is Bram?

+

+

=

@

FULLSTAQ

Waar houd ik me zoal mee bezig?

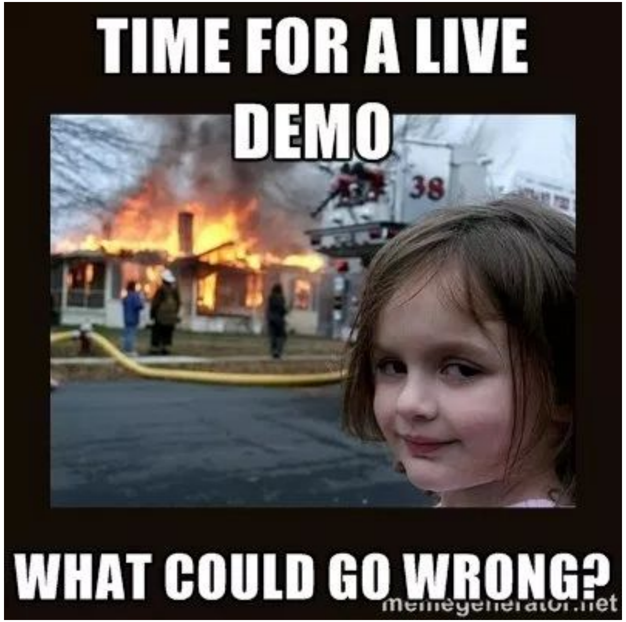
Agenda

Azure Kubernetes Service (AKS)

Application Container
Security Guide

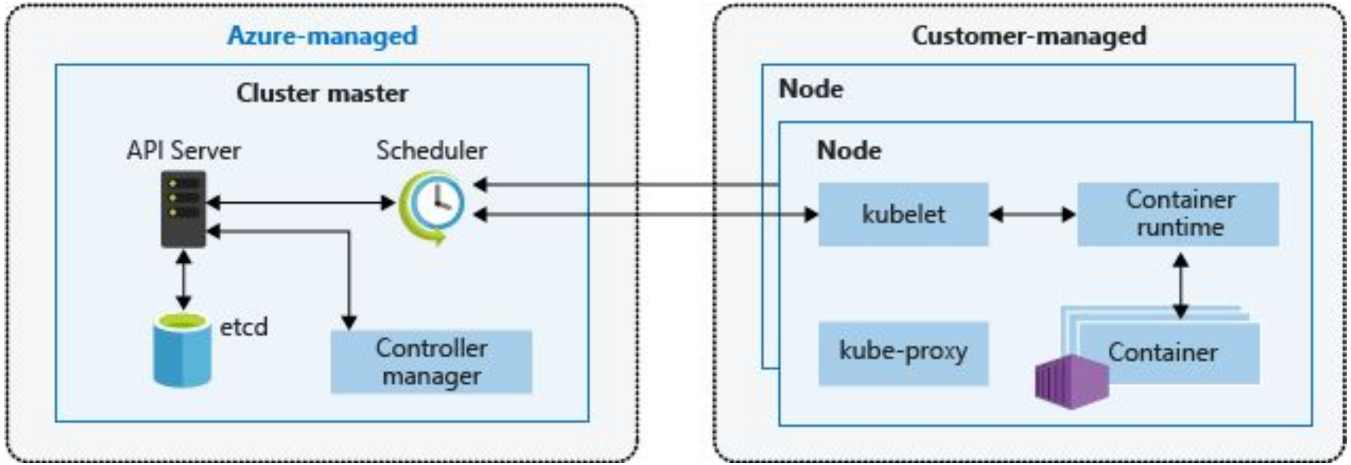
NIST 800-190

aqua



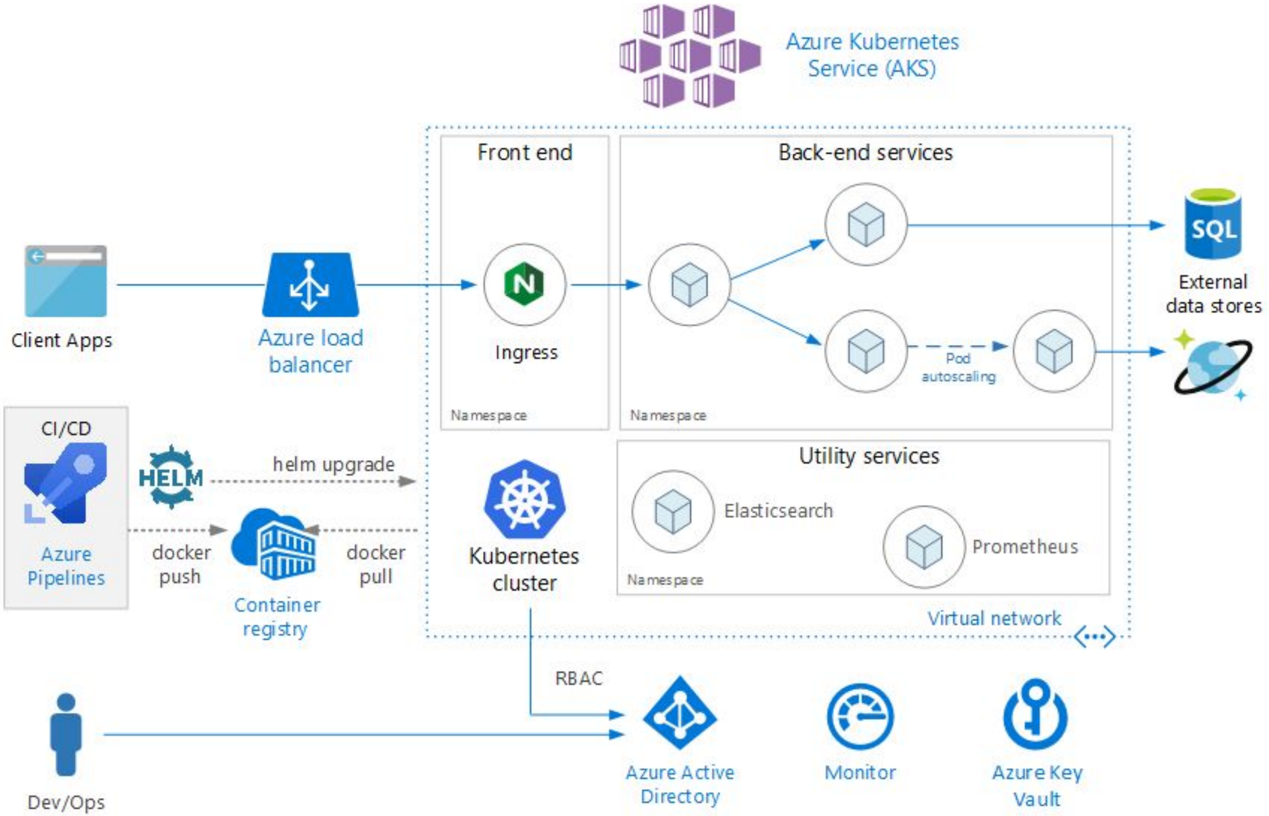
Korte intro over AKS - 1

Azure Kubernetes Service is een, bijna volledig, managed Kubernetes PaaS dienst ......

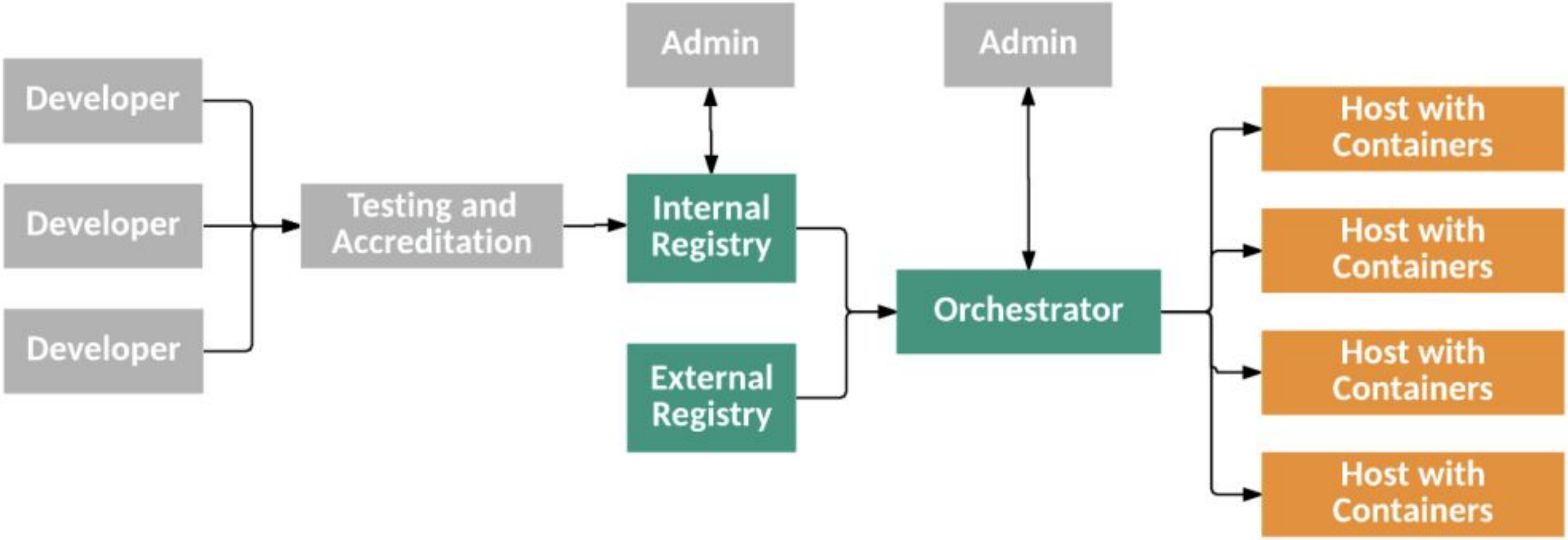


Korte intro over AKS - 2

…… en integreert met Azure/ Microsoft diensten

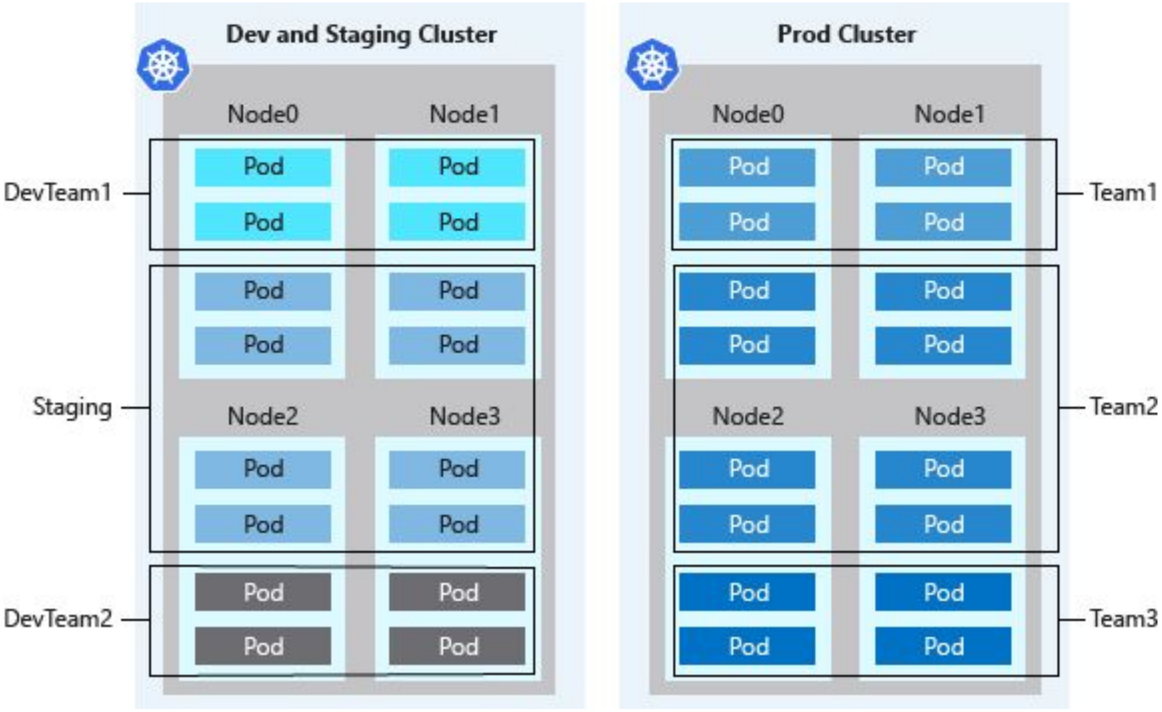


NIST 800-190: Application Container Security



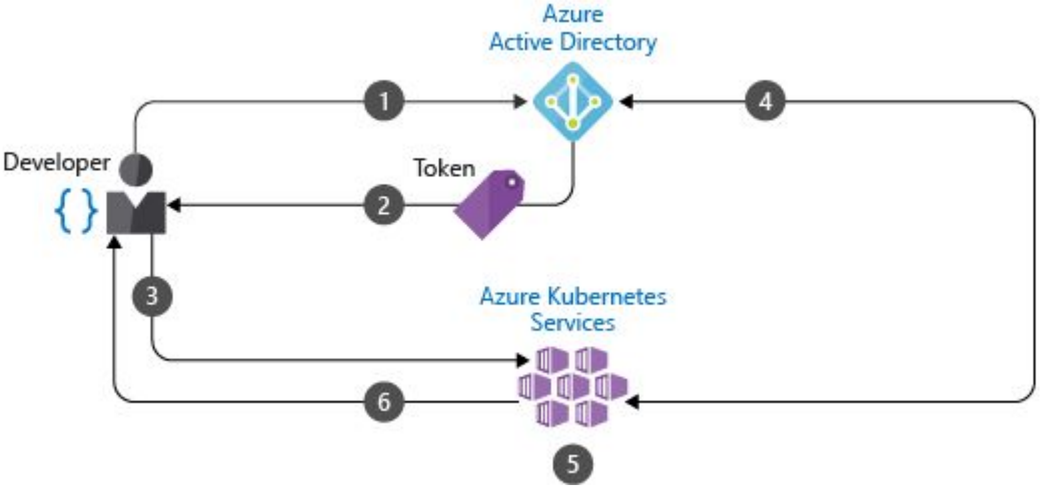
Best practices AKS & Security

Logische indeling/ isolatie van clusters en teams

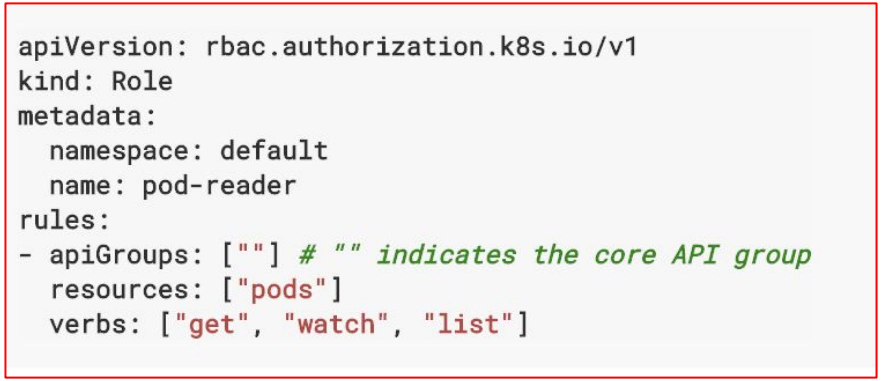


Role Based Access Control

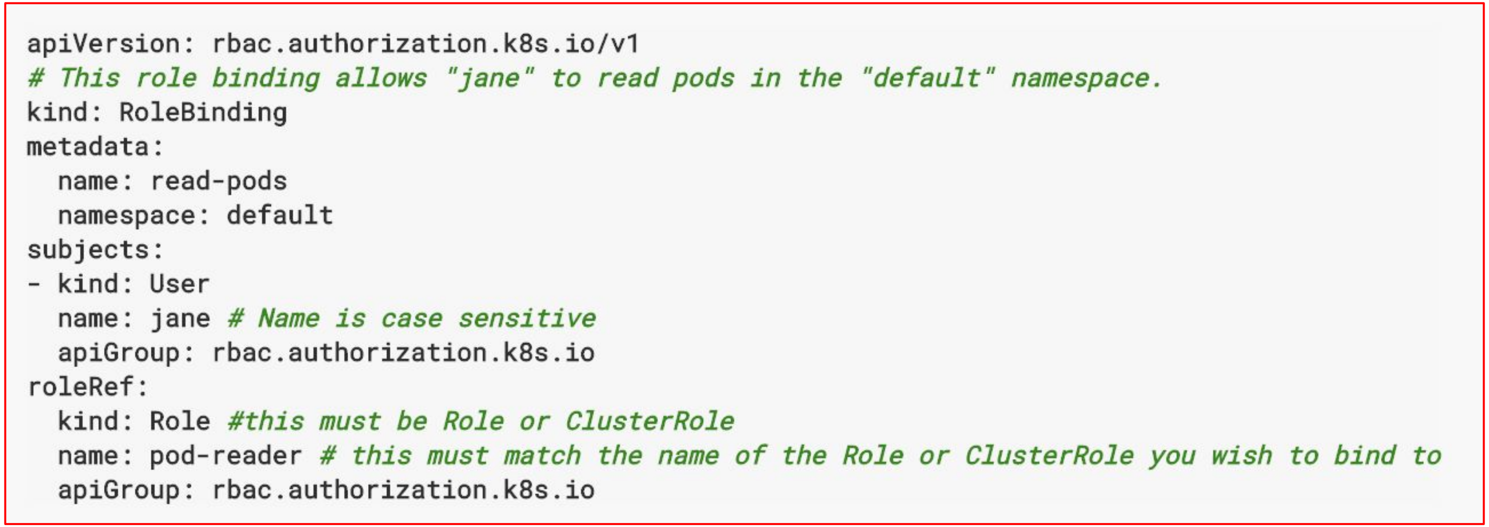
Integreer met Azure AD



Role Based Access Control

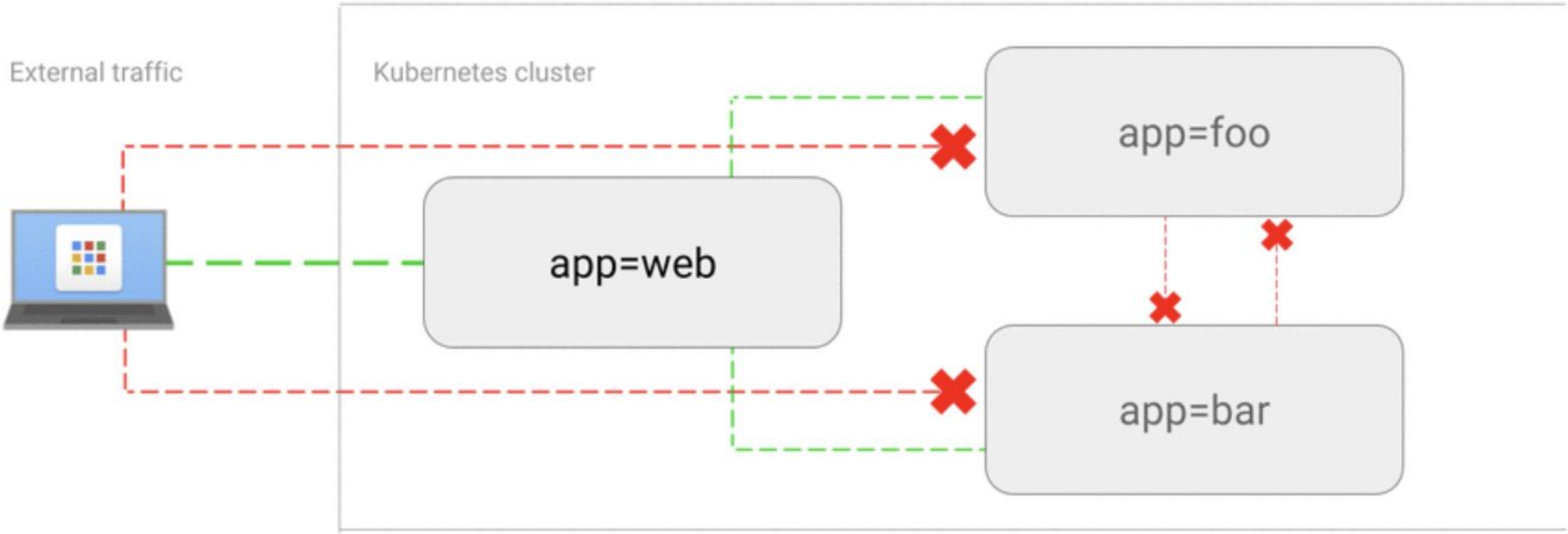


```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

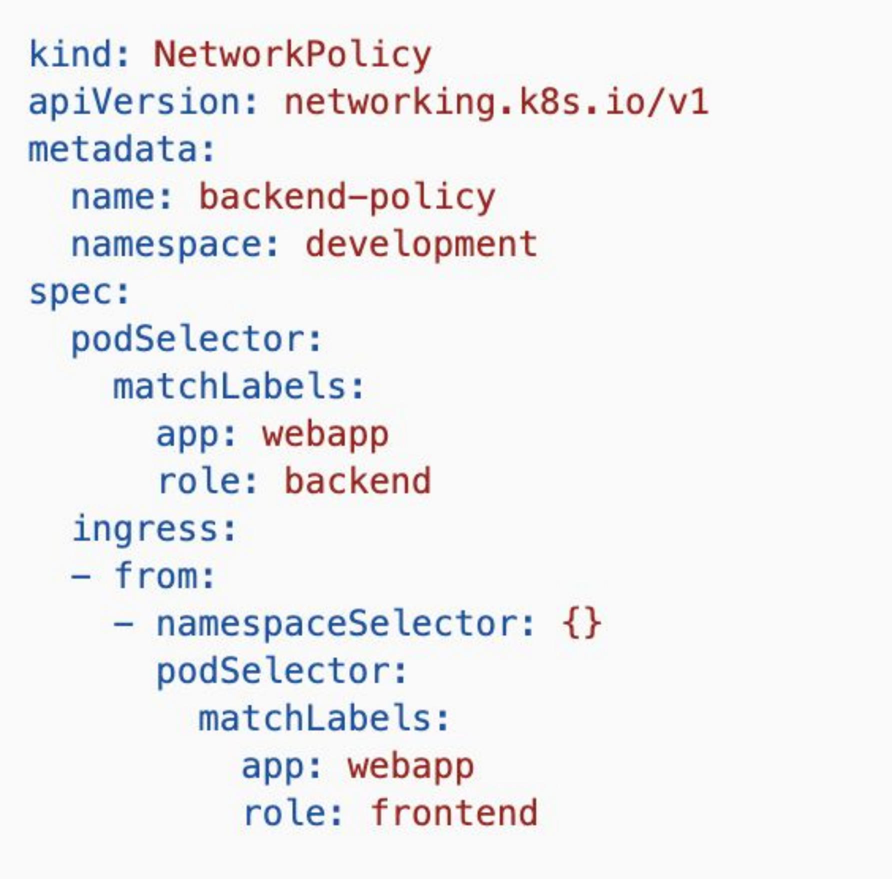


```
apiVersion: rbac.authorization.k8s.io/v1
# This role binding allows "jane" to read pods in the "default" namespace.
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role #this must be Role or ClusterRole
  name: pod-reader # this must match the name of the Role or ClusterRole you wish to bind to
  apiGroup: rbac.authorization.k8s.io
```

Netwerk policies



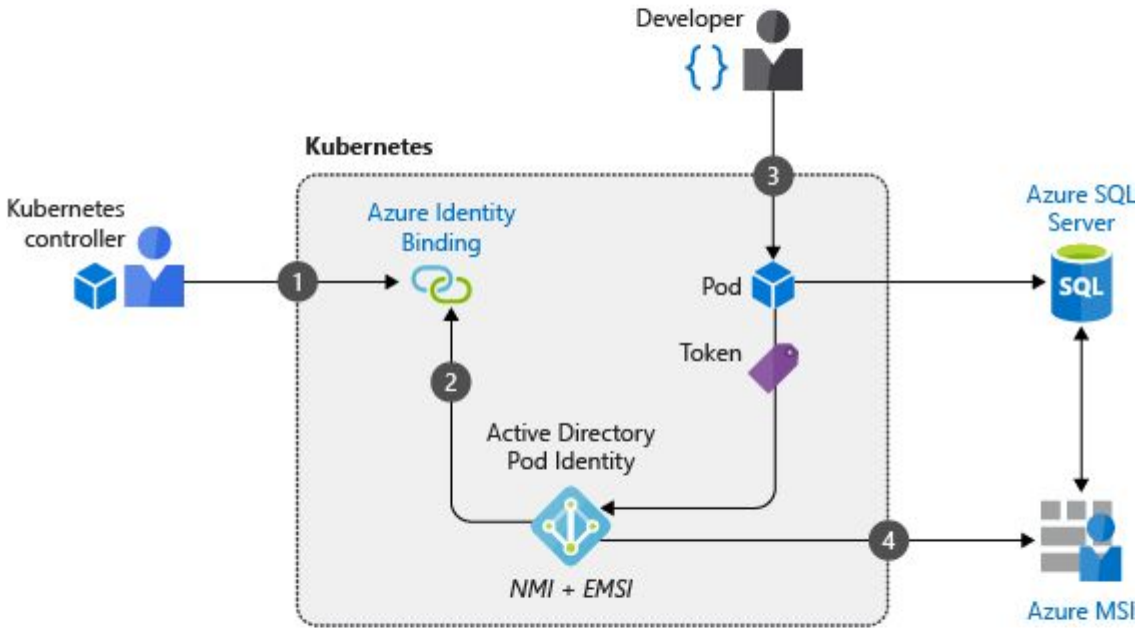
Netwerk policies



```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
```

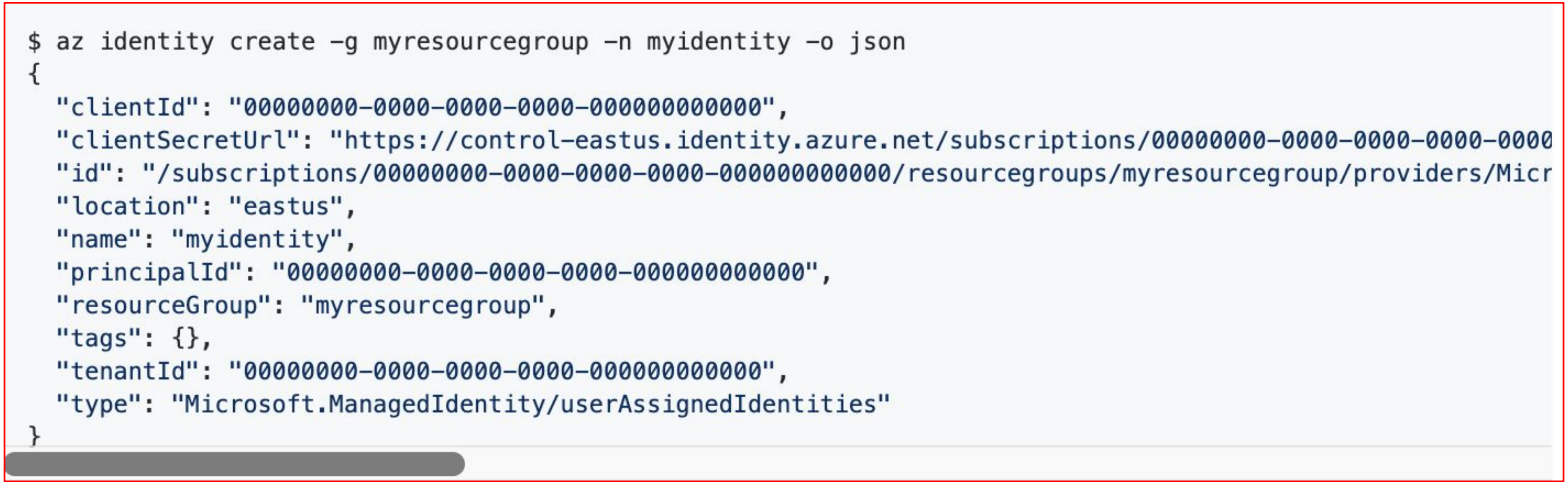
Pod identity

Als pods toegang nodig hebben tot andere Azure diensten

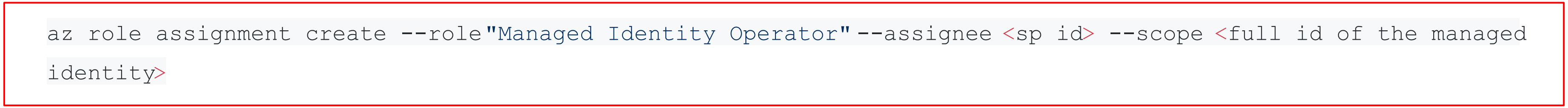


Pod identity

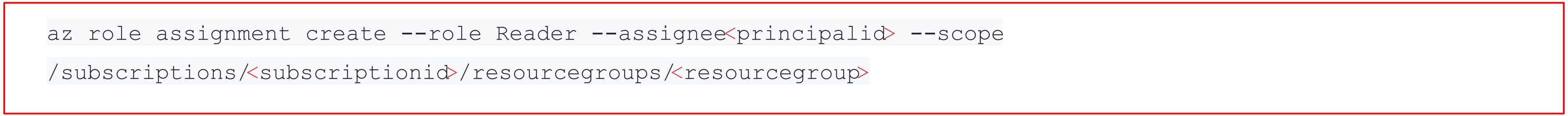
Aanmaken Managed Identity in Azure



```
$ az identity create -g myresourcegroup -n myidentity -o json
{
  "clientId": "00000000-0000-0000-0000-000000000000",
  "clientSecretUrl": "https://control-eastus.identity.azure.net/subscriptions/00000000-0000-0000-0000-0000
  "id": "/subscriptions/00000000-0000-0000-0000-000000000000/resourcegroups/myresourcegroup/providers/Micr
  "location": "eastus",
  "name": "myidentity",
  "principalId": "00000000-0000-0000-0000-000000000000",
  "resourceGroup": "myresourcegroup",
  "tags": {},
  "tenantId": "00000000-0000-0000-0000-000000000000",
  "type": "Microsoft.ManagedIdentity/userAssignedIdentities"
}
```



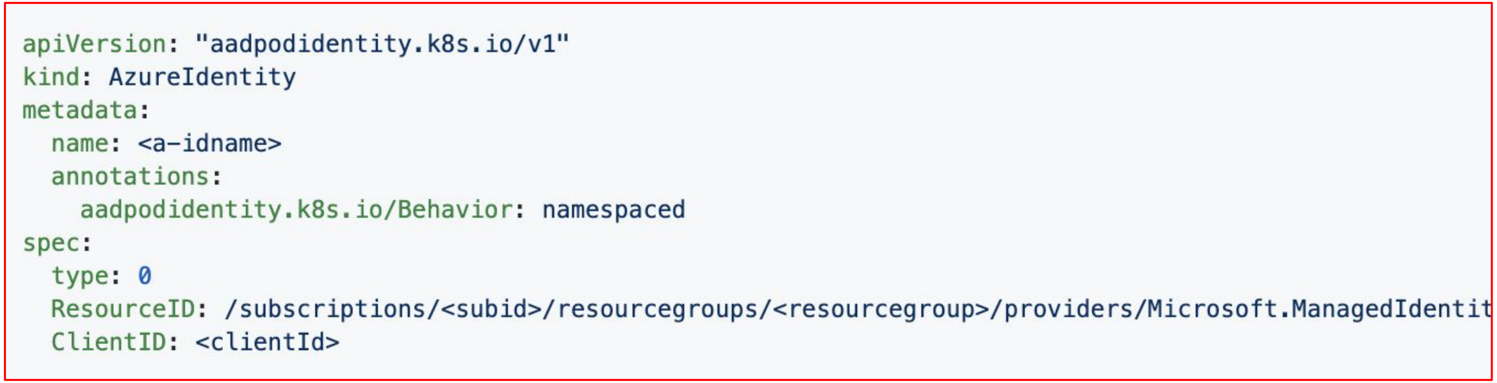
```
az role assignment create --role"Managed Identity Operator"--assignee <sp id> --scope <full id of the managed
identity>
```



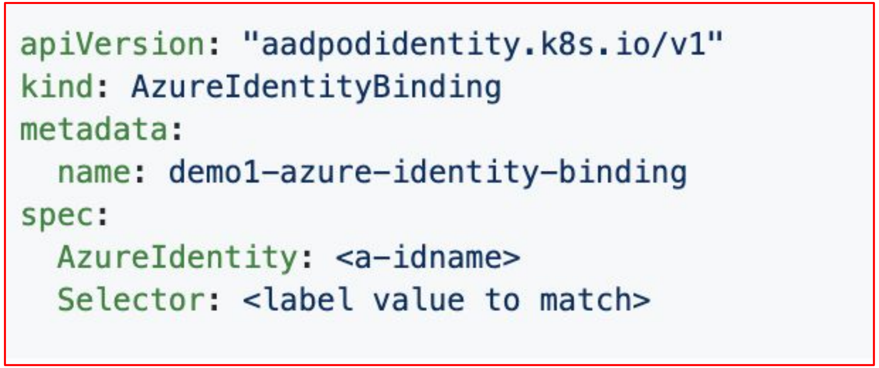
```
az role assignment create --role Reader --assignee<principalid> --scope
/subscriptions/<subscriptionid>/resourcegroups/<resourcegroup>
```

Pod identity

Aanmaken Managed Identity op je cluster



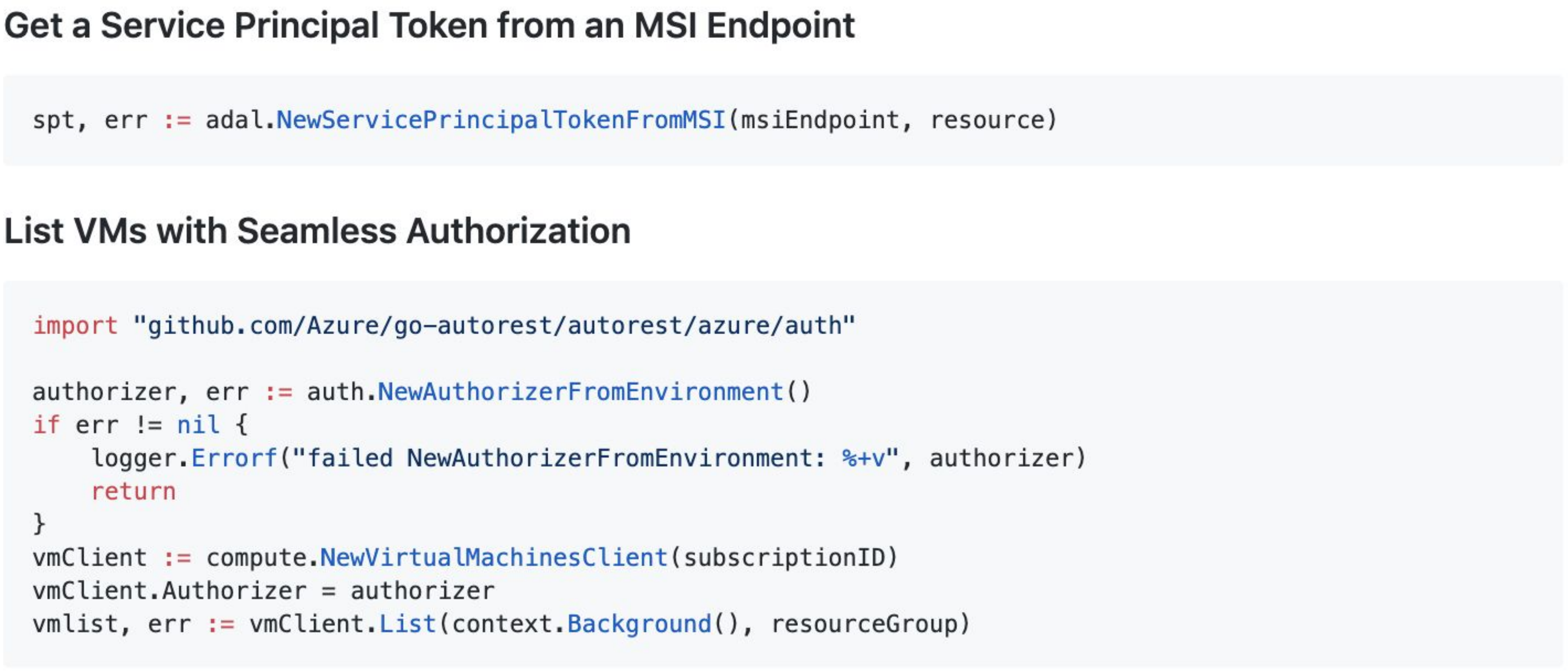
```
apiVersion: "aadpodidentity.k8s.io/v1"
kind: AzureIdentity
metadata:
  name: <a-idname>
  annotations:
    aadpodidentity.k8s.io/Behavior: namespaced
spec:
  type: 0
  ResourceID: /subscriptions/<subid>/resourcegroups/<resourcegroup>/providers/Microsoft.ManagedIdentit
  ClientID: <clientId>
```



```
apiVersion: "aadpodidentity.k8s.io/v1"
kind: AzureIdentityBinding
metadata:
  name: demo1-azure-identity-binding
spec:
  AzureIdentity: <a-idname>
  Selector: <label value to match>
```

Pod identity

Voorbeeld code



Get a Service Principal Token from an MSI Endpoint

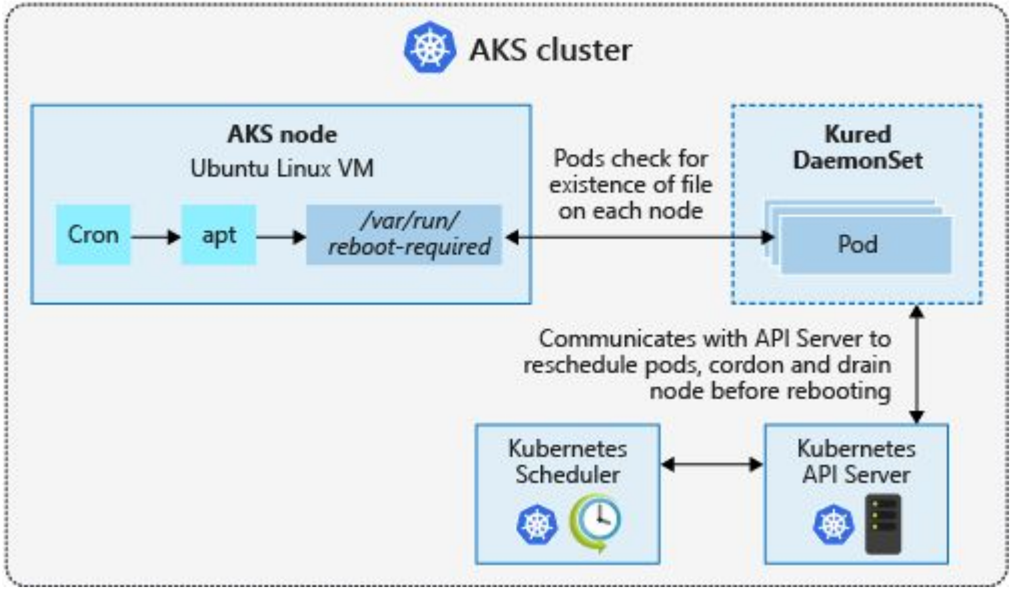
```
spt, err := adal.NewServicePrincipalTokenFromMSI(msiEndpoint, resource)
```

List VMs with Seamless Authorization

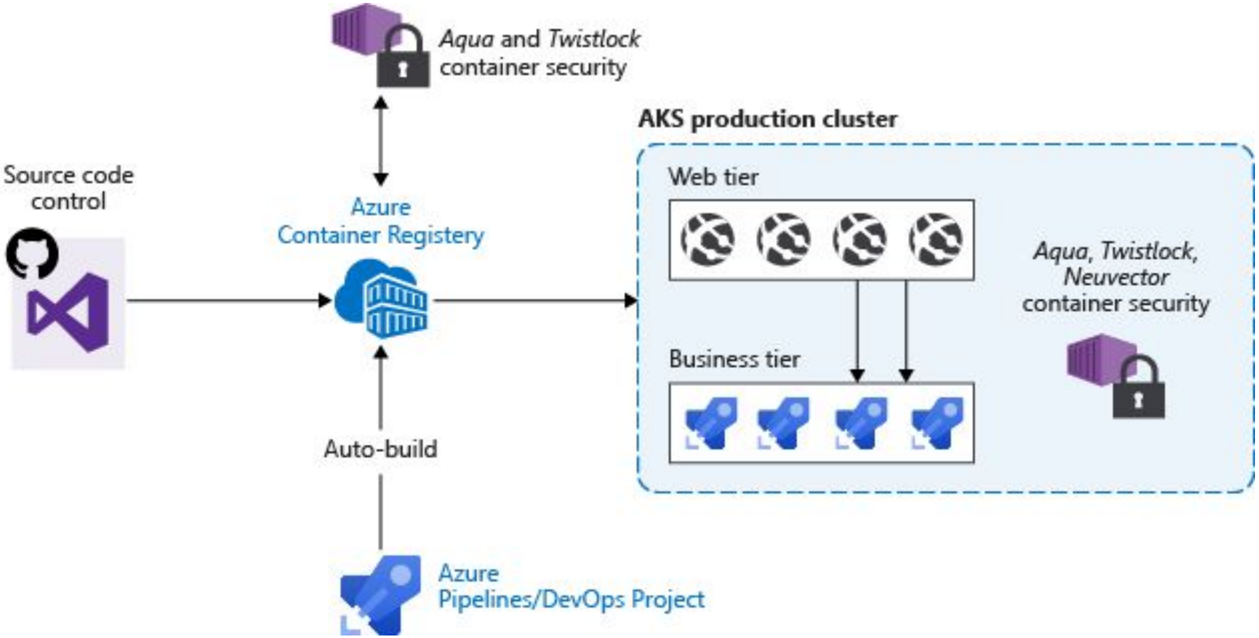
```
import "github.com/Azure/go-autorest/autorest/azure/auth"

authorizer, err := auth.NewAuthorizerFromEnvironment()
if err != nil {
    logger.Errorf("failed NewAuthorizerFromEnvironment: %+v", authorizer)
    return
}
vmClient := compute.NewVirtualMachinesClient(subscriptionID)
vmClient.Authorizer = authorizer
vmlist, err := vmClient.List(context.Background(), resourceGroup)
```

Kured

Automatische reboots van Linux vm's



Secure images en run time



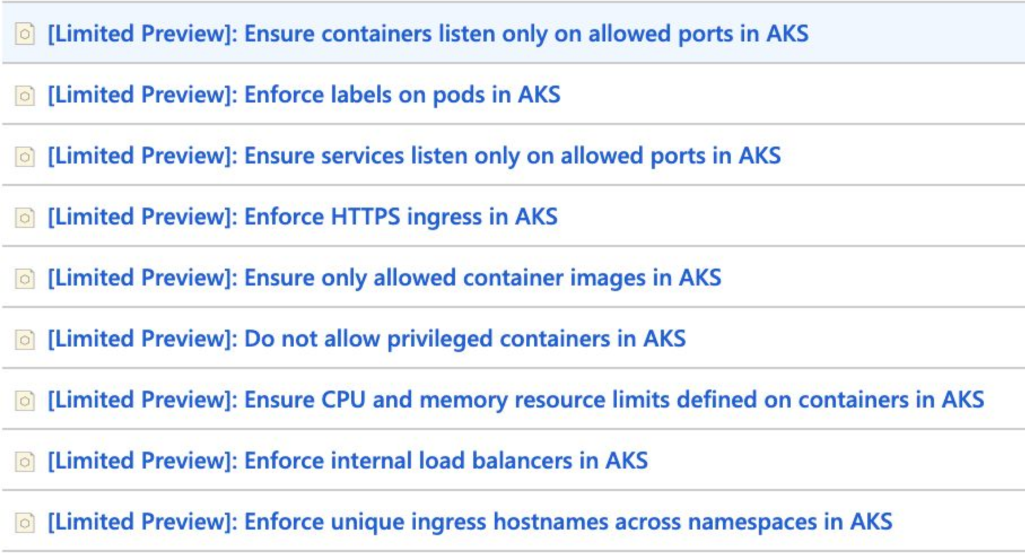
Best practices - Review

- Logische opdeling cluster door **Namespaces**
  - incl **RBAC**
  - incl **Netwerk Policies**
- **Pod Identity**
- **Kured** (herstarten vm’s ivm updates)

(Limited) Preview features - 1

Sorry… toch een paar bullets:

- Beveiligen van de API server dmv geautoriseerde IP ranges
- Pod security policy
- Azure Policy



[Limited Preview]: Ensure containers listen only on allowed ports in AKS

[Limited Preview]: Enforce labels on pods in AKS

[Limited Preview]: Ensure services listen only on allowed ports in AKS

[Limited Preview]: Enforce HTTPS ingress in AKS

[Limited Preview]: Ensure only allowed container images in AKS

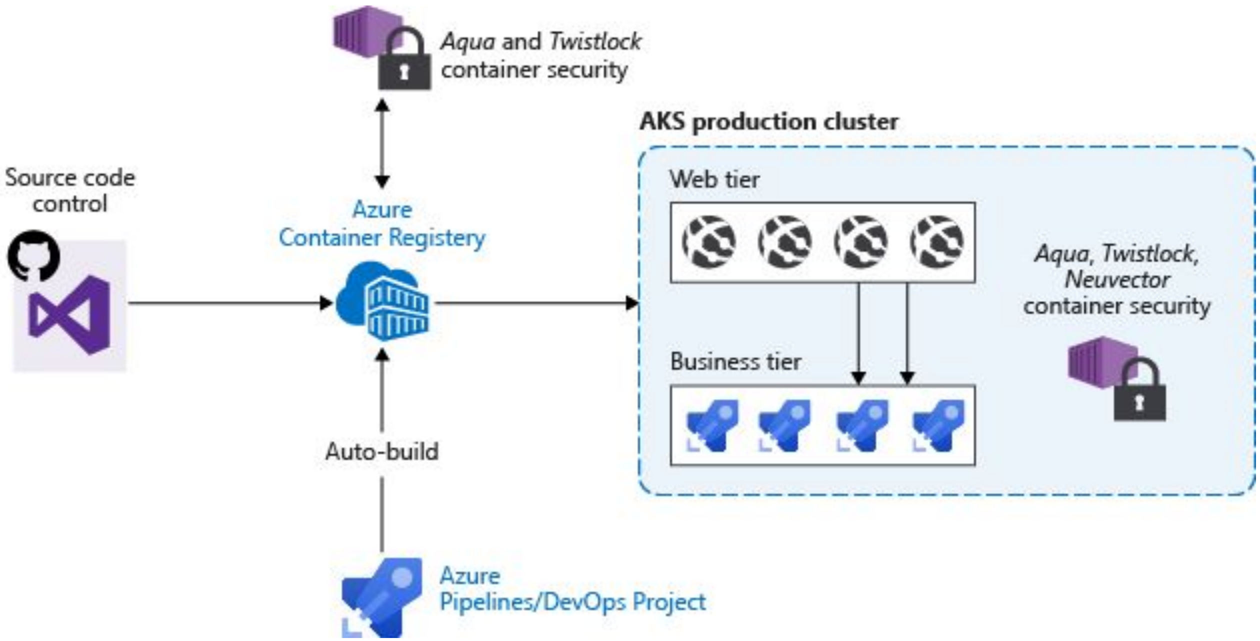
[Limited Preview]: Do not allow privileged containers in AKS

[Limited Preview]: Ensure CPU and memory resource limits defined on containers in AKS

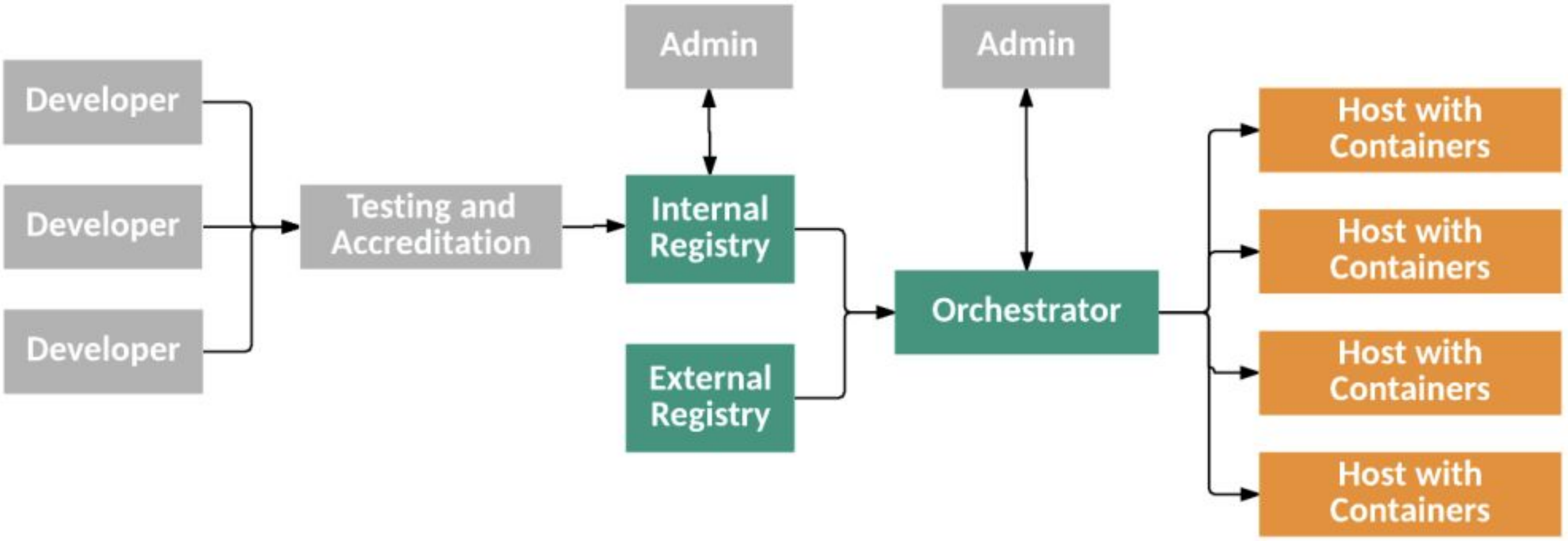
[Limited Preview]: Enforce internal load balancers in AKS

[Limited Preview]: Enforce unique ingress hostnames across namespaces in AKS

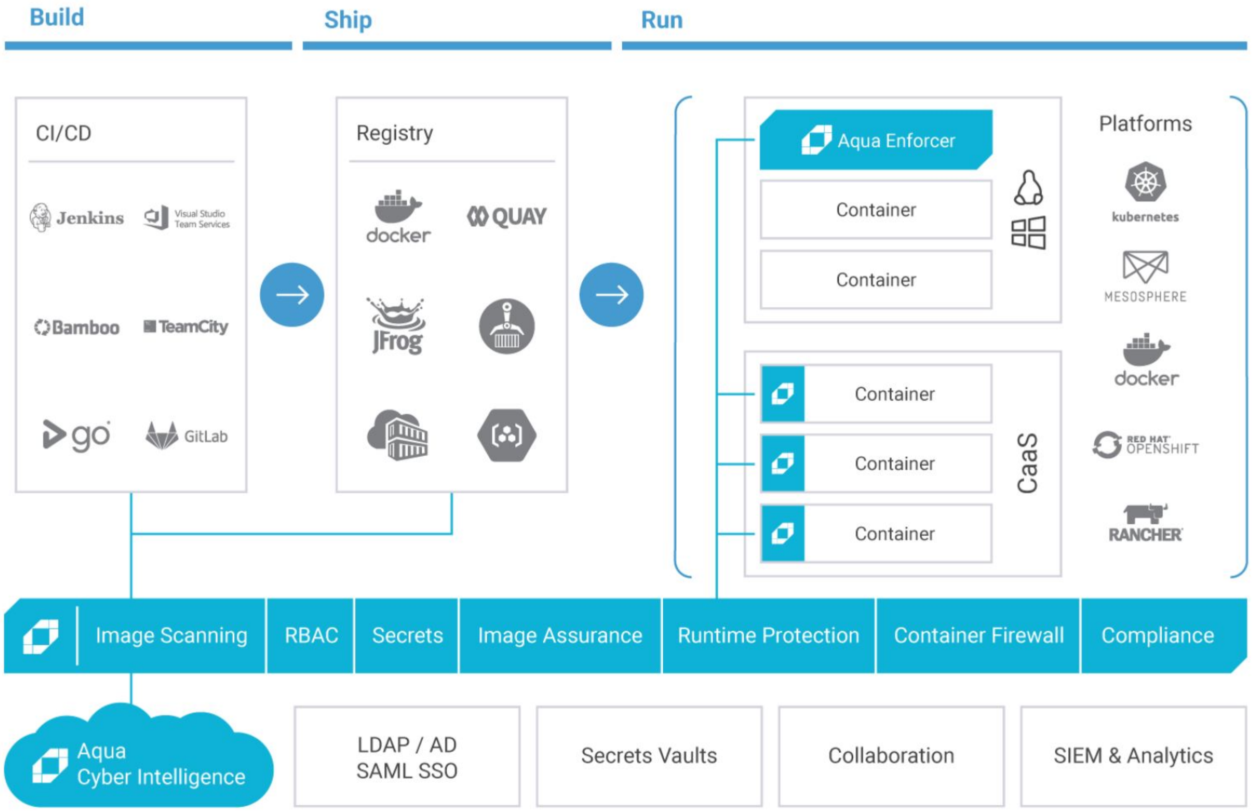
Secure images en run time



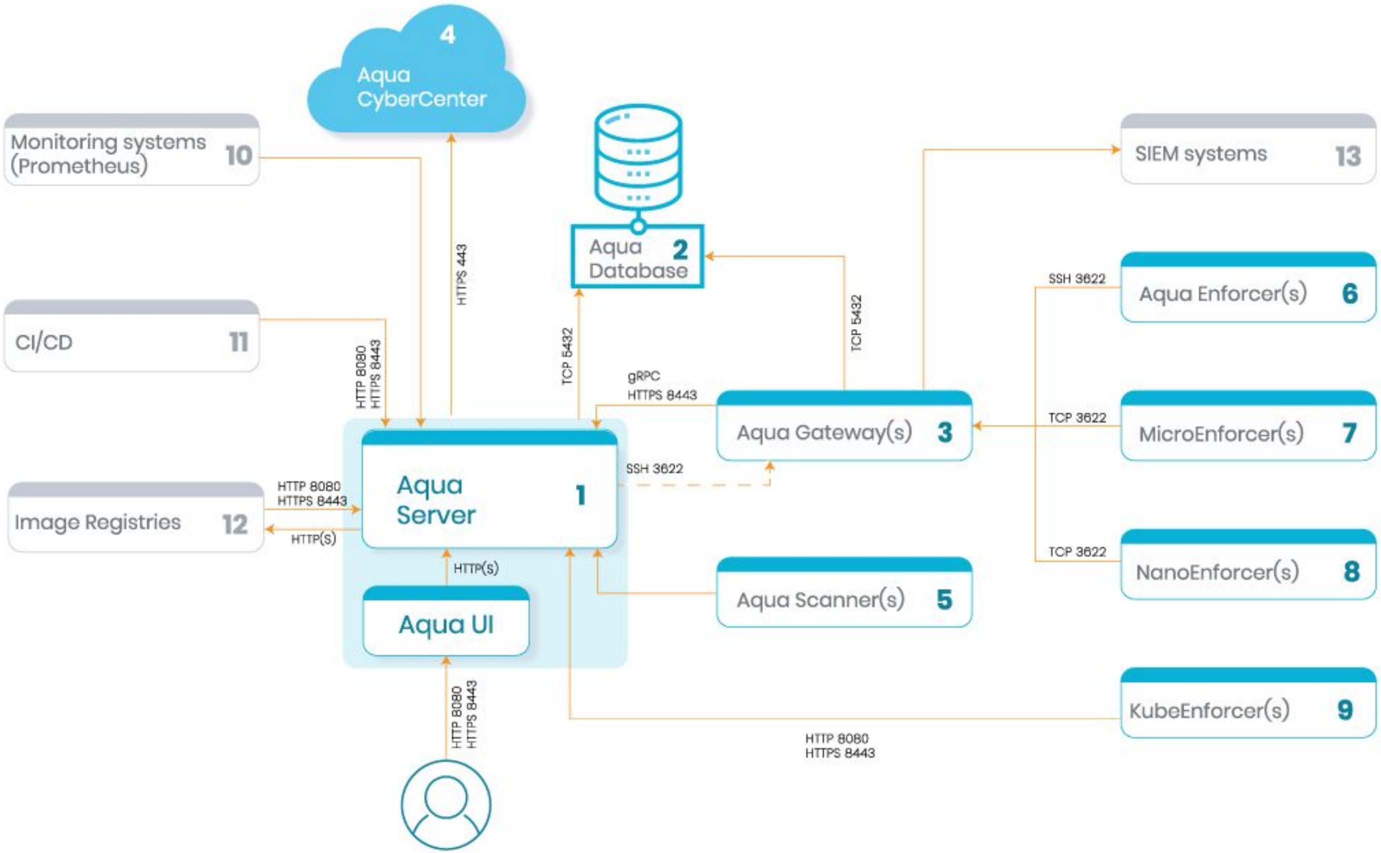
NIST 800-190: Application Container Security



Aqua


Build
Ship
Run
CI/CD
Jenkins
Visual Studio Team Services
Bamboo
TeamCity
go
GitLab
Registry
docker
QUAY
JFrog
Aqua Enforcer
Container
Container
Container
Container
Container
CaaS
Platforms
kubernetes
MESOSPHERE
docker
RED HAT OPENSHIFT
RANCHER
Image Scanning
RBAC
Secrets
Image Assurance
Runtime Protection
Container Firewall
Compliance
Aqua Cyber Intelligence
LDAP / AD SAML SSO
Secrets Vaults
Collaboration
SIEM & Analytics

Aqua CSP


4
Aqua CyberCenter
Monitoring systems (Prometheus) 10
HTTPS 443
Aqua Database 2
SIEM systems 13
CI/CD 11
HTTP 8080
HTTPS 8443
TCP 5432
TCP 5432
gRPC
HTTPS 8443
SSH 3622
Aqua Enforcer(s) 6
Aqua Gateway(s) 3
TCP 3622
MicroEnforcer(s) 7
SSH 3622
Aqua Server 1
Image Registries 12
HTTP 8080
HTTPS 8443
HTTP(S)
HTTP(S)
TCP 3622
NanoEnforcer(s) 8
Aqua Scanner(s) 5
Aqua UI
KubeEnforcer(s) 9
HTTP 8080
HTTPS 8443
HTTP 8080
HTTPS 8443

Caution! You are about to see a DEMO

Let’s hope not to upset the DEMO gods