2019
NT KONFERENCA
21. - 23. MAJ 2019

Azure Track v2.0

Containers adoption in Microsoft Azure

Matjaž Perpar
Cloud Solution Architect
Microsoft

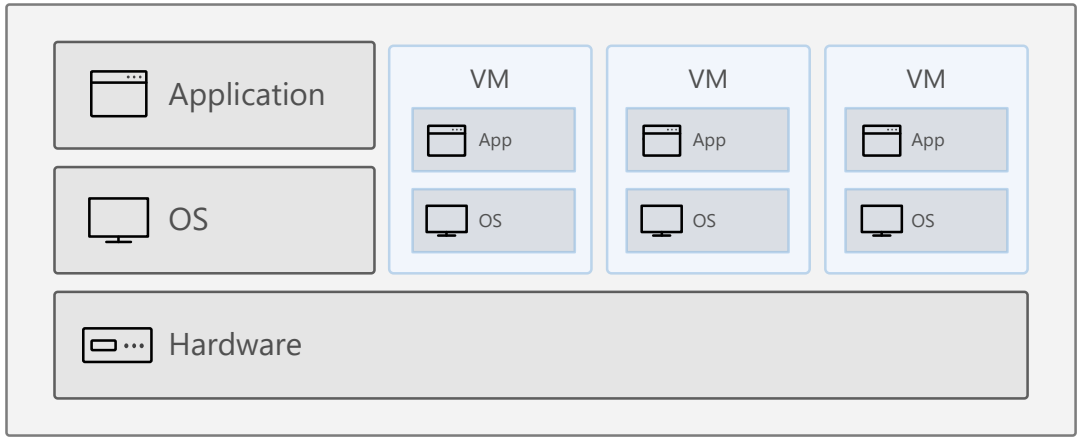
#ntk19

Azure Track v2.0

- Azure Track v2.0 – Moj Drugi PC je Azure
- Azure Track v2.0 – Azure Stack in »hiperkonvergirana« infrastruktura (HCI)
- **Azure Track v2.0 – Cloud-native pristopi s kontejnerji**
- Azure Track v2.0 – Migracija v Azure – kaj, kako, zakaj?
- Azure Track v2.0 – DevOps in Infrastructure as code
- Azure Track v2.0 – Cloud management in governance

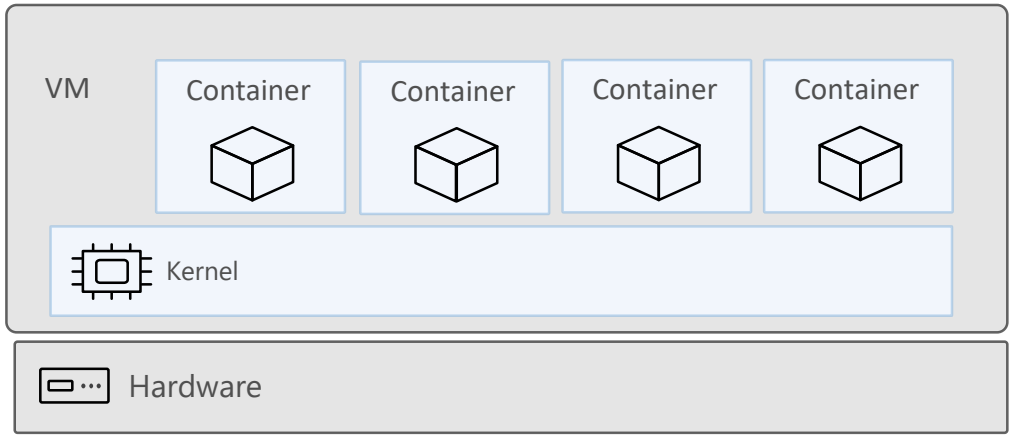
What is a container?

Traditional virtual machines = hardware virtualization



- Large footprint
- Slow to bootup
- Poor utilization of computing resource
- Ideal for long running tasks

Containers = operating system (Kernel) virtualization



- Small footprint
- Quick bring-up time
- Portable
- Better utilization of computing resource
- Agile – ideal for running short-lived/ephemeral tasks (think Azure Functions, AWS Lambda)

Average lifetime of a container = 2.5 days, VM = 23 days. Containers churn 9 times more than VMs

Containers: Fast facts

STANDARIZATION (Docker)

+

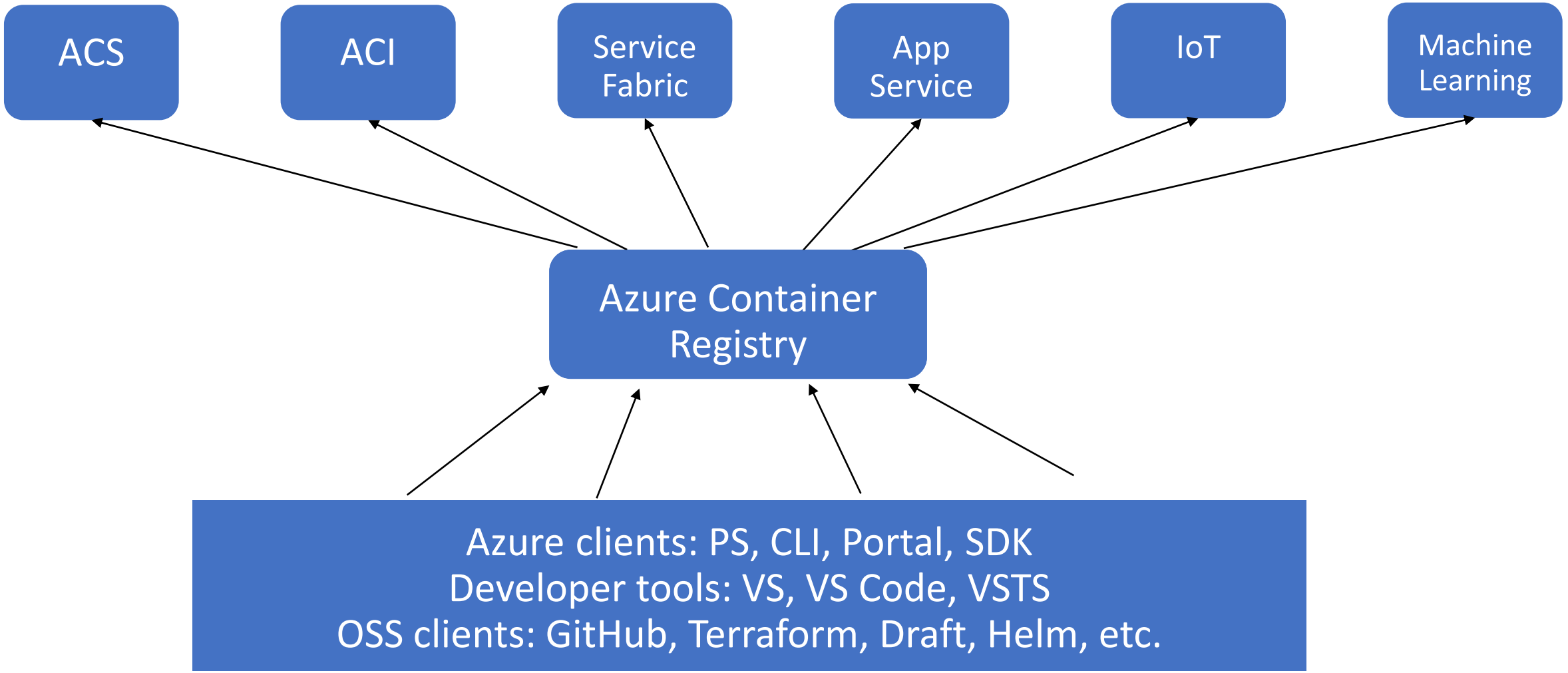
PORTABILTY + SPEED + CONFIGURATION

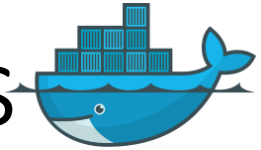
1. Containers allows you to MIMIC a Production Environment during Development, Testing and Pre-production
2. Containers and associated Tooling allows you develop and run solutions on our Laptop
3. Containers considerably reduce the time and cost of Engineering Operations

The benefits of using containers

Any OS	Anywhere	Any app	Any language
Linux	On-premises	Monolith	Java
Windows	Cloud	Microservice	.Net
			Python
			Node

Containers as the App Packaging Format



What is  docker?

An open source container runtime

Mac, Windows and Linux support

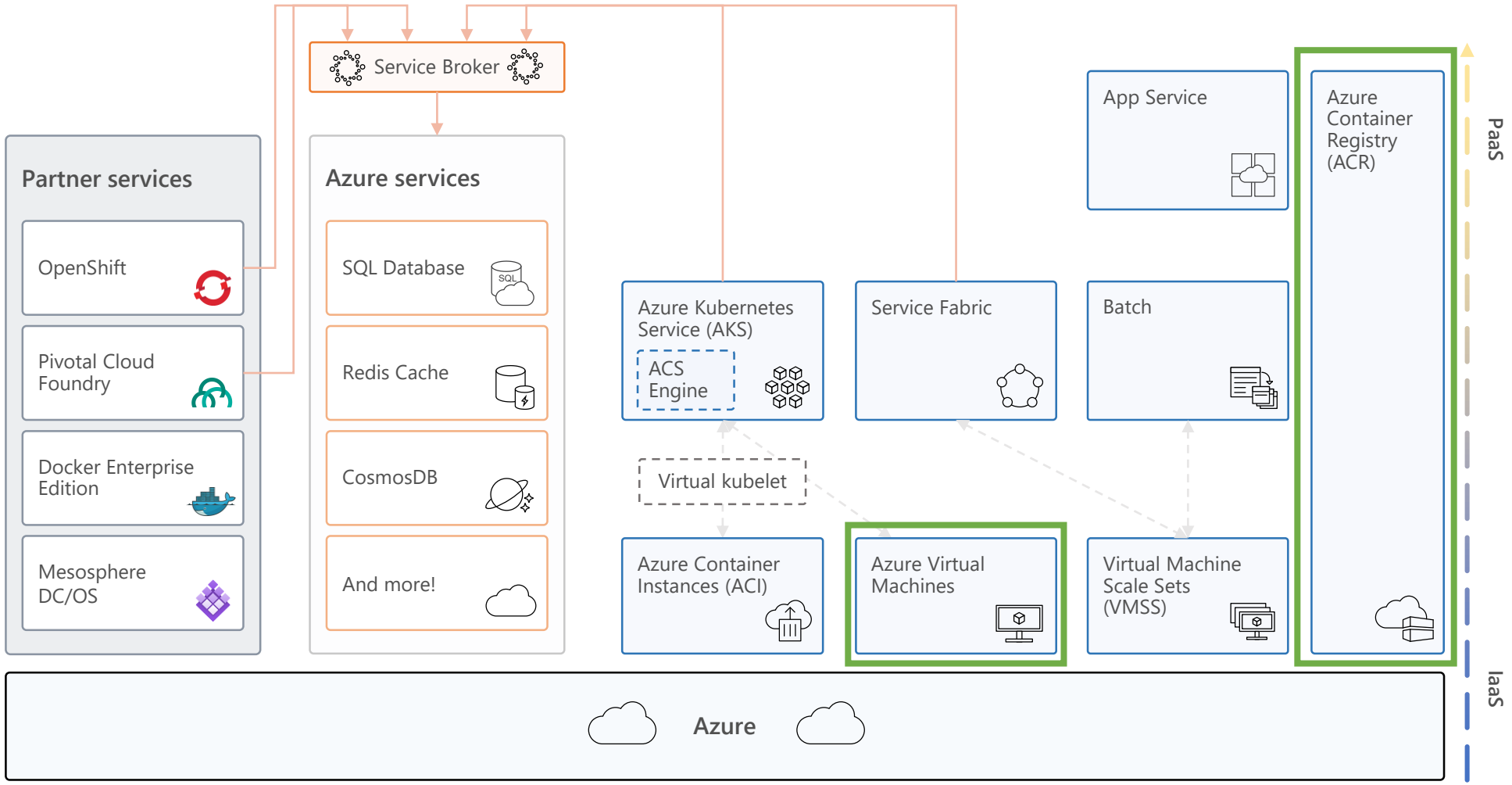
```
# The world’s simplest Dockerfile
$ cat Dockerfile
FROM scratch
COPY hello /
CMD ["/hello"]

# Let’s create a docker image “tagged” hello-world
$ docker build –t hello-world .

# And run it…
$ docker run hello-world
```

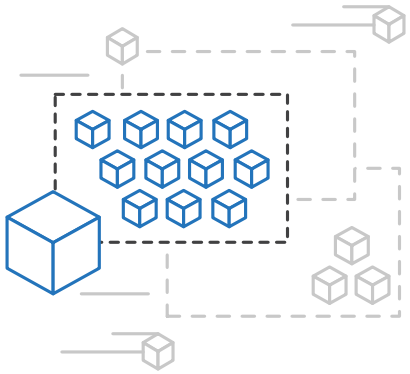

DOCKER
INTRO
#ntk19

Azure container ecosystem

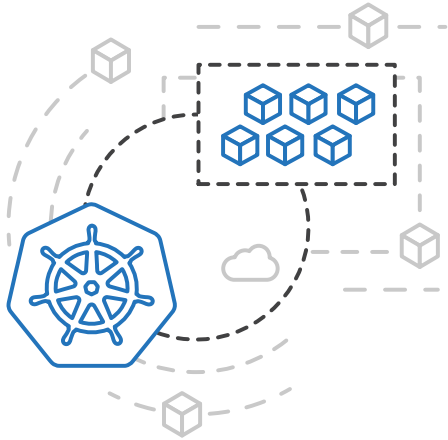


Azure Container Instances (ACI)

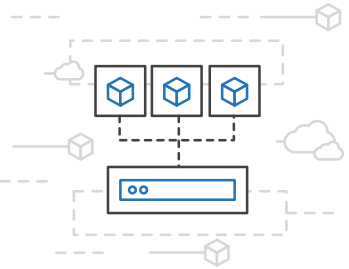
Easily run serverless containers



Run containers without managing servers

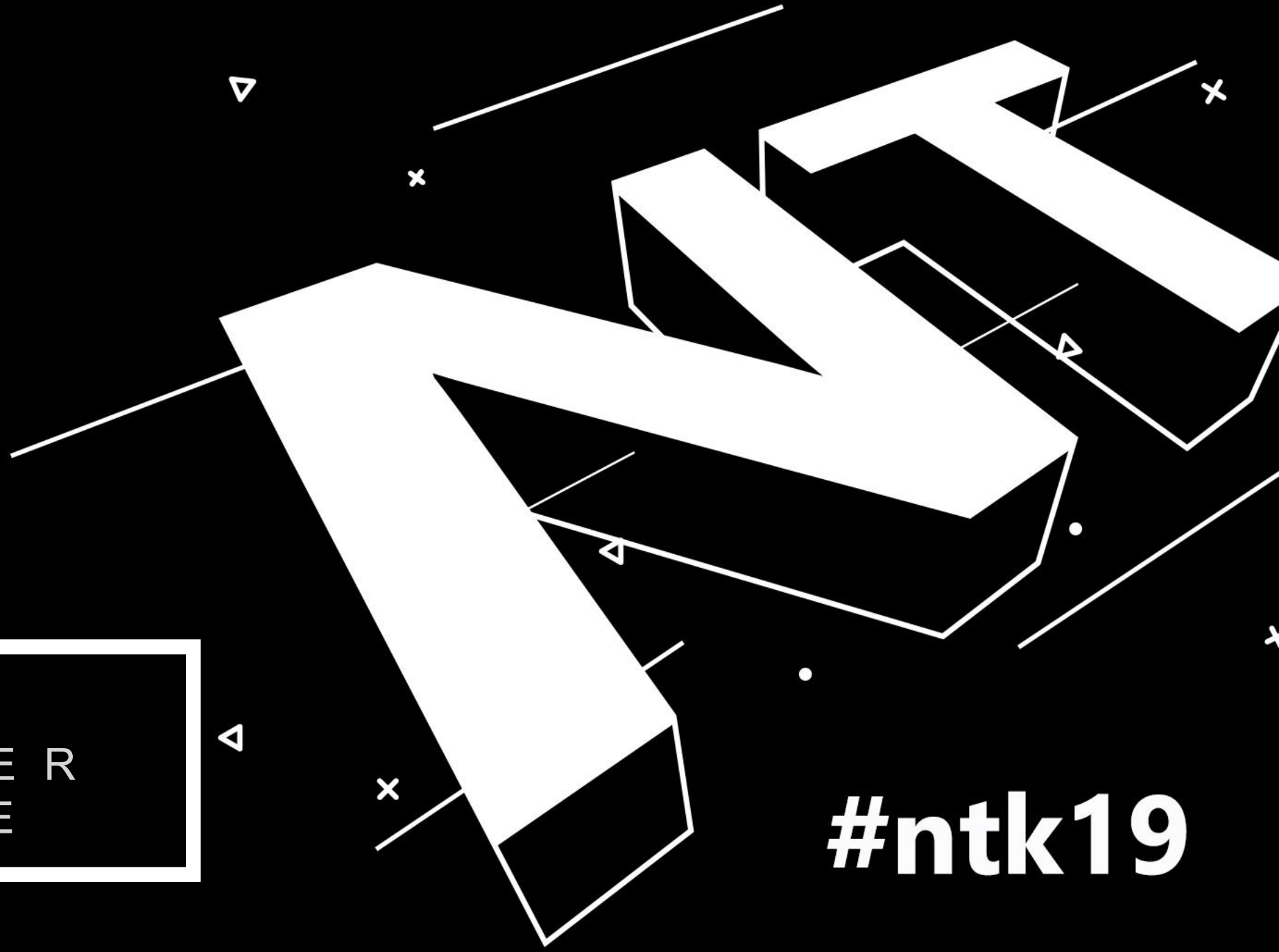


Containers as a primitive billed per second



Secure applications with hypervisor isolation

AZURE CONTAINER INSTANCE



#ntk19

The elements of **orchestration**

Scheduling

Affinity/anti-affinity

Health monitoring

Failover

Scaling

Networking

Service discovery

Coordinated app upgrades

Kubernetes: the de-facto orchestrator

Portable

Public, private, hybrid, multi-cloud

Extensible

Modular, pluggable, hookable, composable

Self-healing

Auto-placement, auto-restart, auto-replication, auto-scaling

Kubernetes: empowering you to do more

Deploy your applications quickly and predictably

Scale your applications on the fly

Roll out new features seamlessly

Limit hardware usage to required resources only

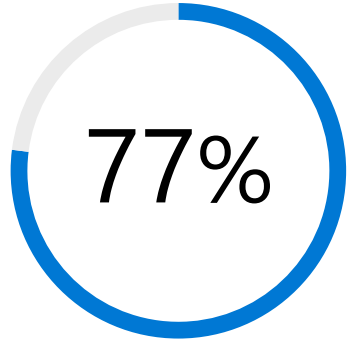
Kubernetes momentum

"By 2020, more than **50%** of enterprises will run **mission-critical, containerized cloud-native applications** in production."

Gartner

Larger companies are leading the adoption.

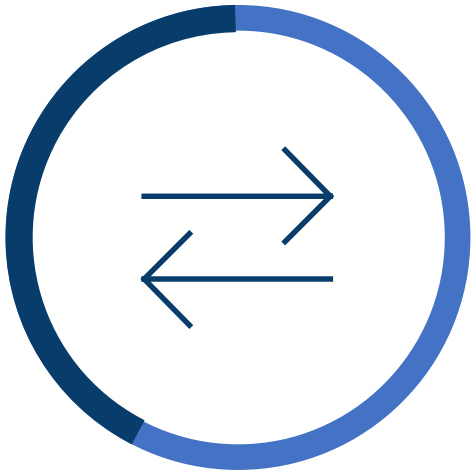
For the organizations running Kubernetes today, **77%**[1] of those with more than 1,000 developers are running it in production.



[1]Heptio: state of Kubernetes 2018

What's behind the growth?

Kubernetes: the leading orchestrator shaping the future app development and management



42%

portability

45%

scalability

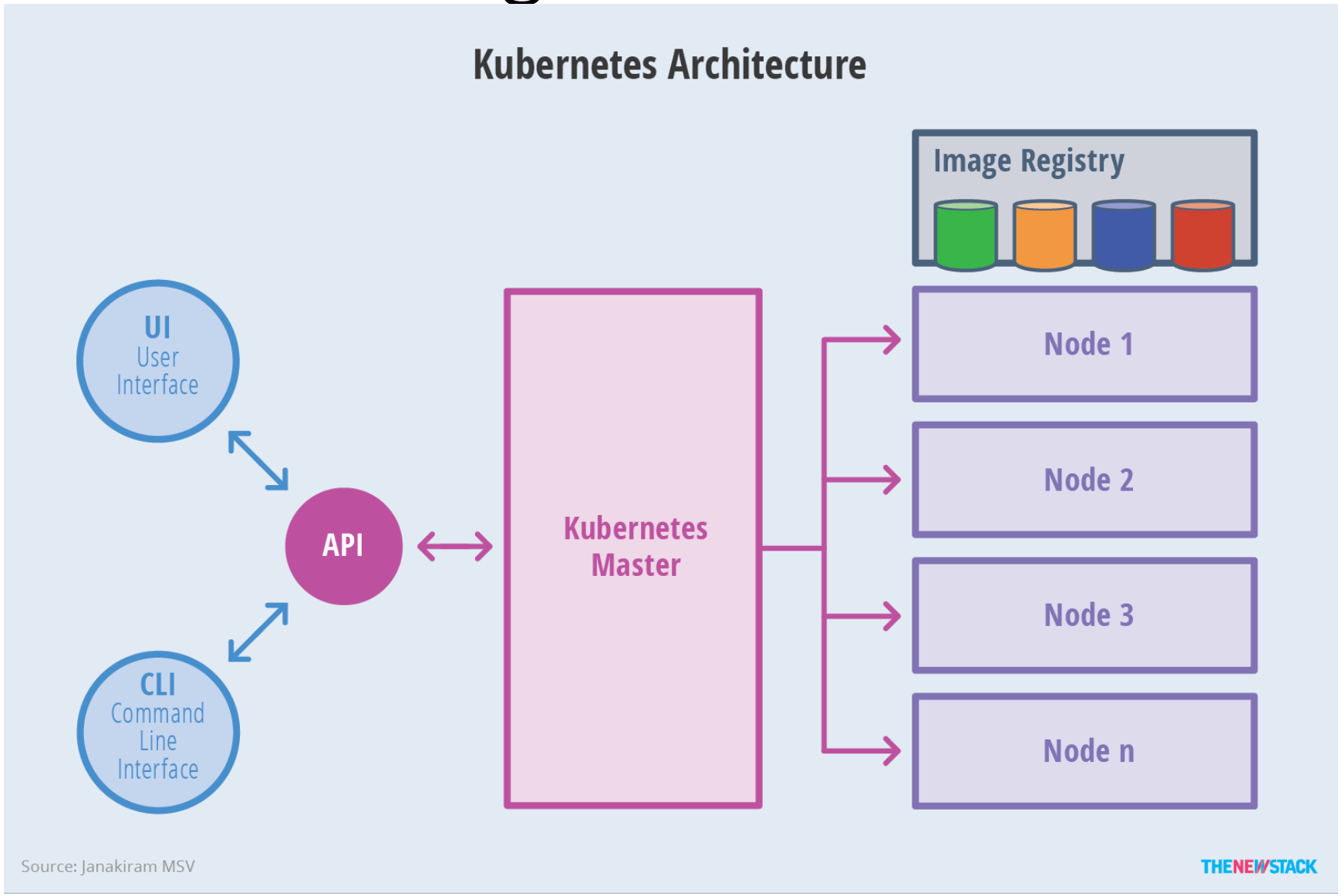
50%

agility

The perceived benefits of Kubernetes

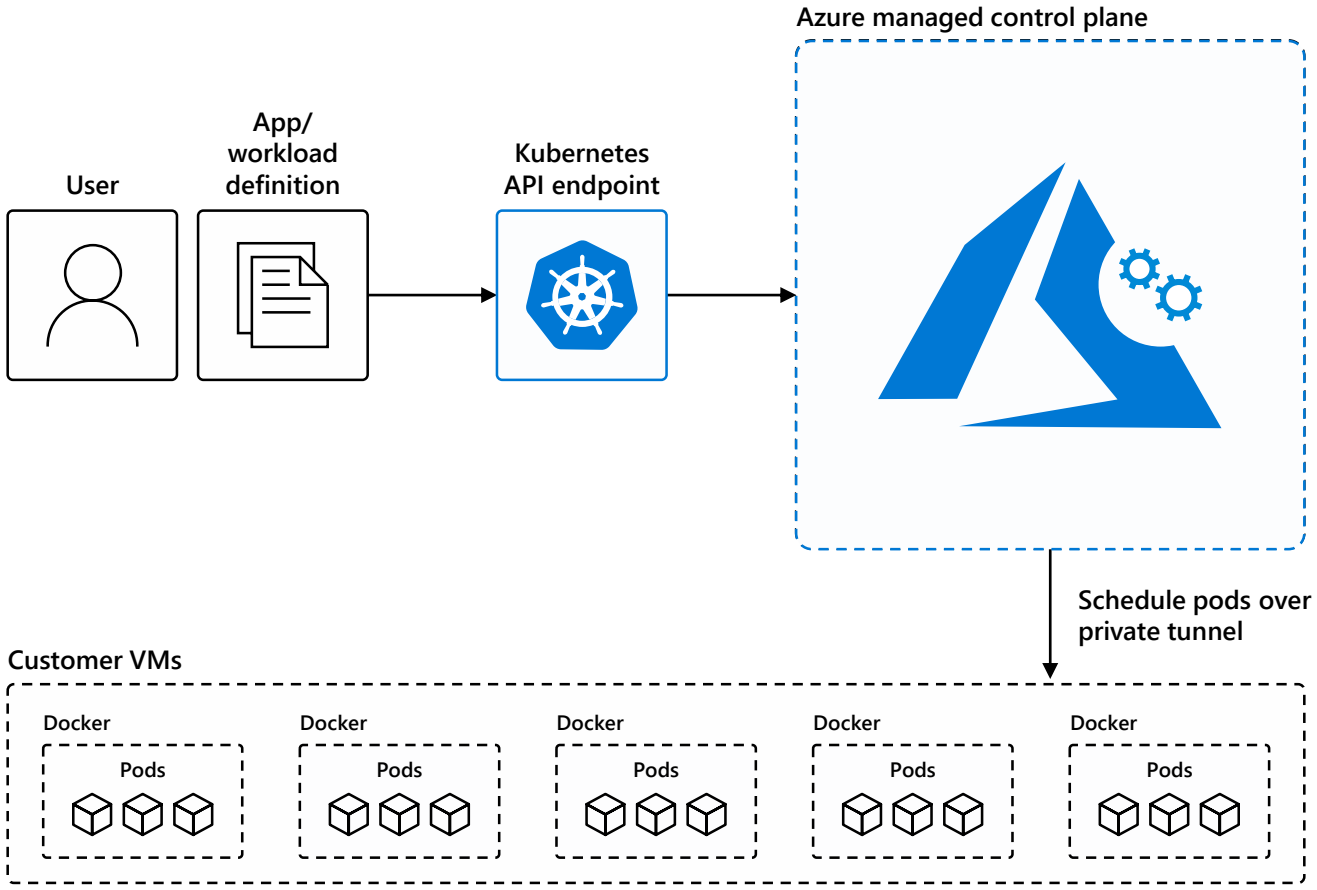
Source: Cloud Native Computing Foundation

Kubernetes: High Level Architecture

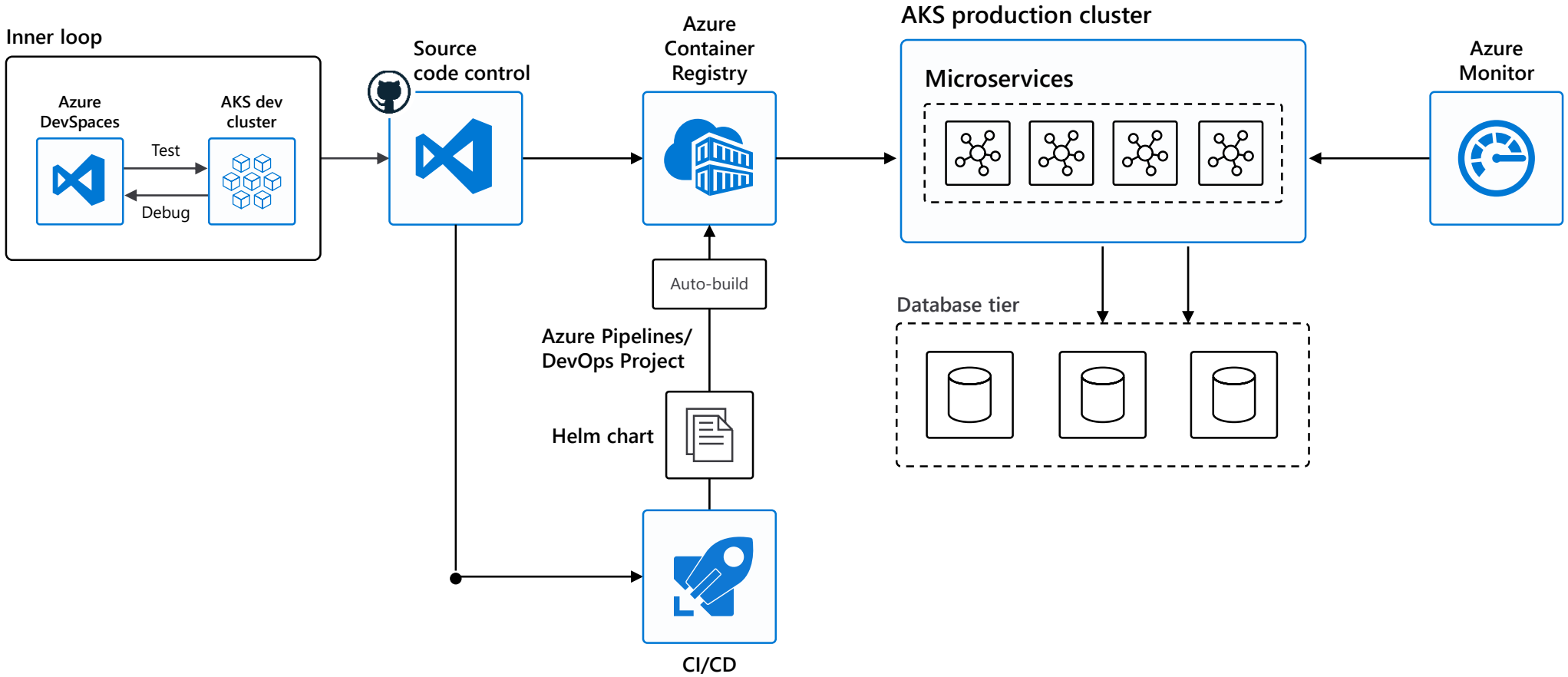


How managed Azure Kubernetes Service works

- Automated upgrades, patches
- High reliability, availability
- Easy, secure cluster scaling
- Self-healing
- API server monitoring
- At no charge



Integrated end-to-end Kubernetes experience



Create your first AKS Cluster

```
az aks create \
    --resource-group matper-aks-rg \
    --name aks01
    --node-count 3 \
    --enable-addons monitoring \
    --generate-ssh-keys \
    --disable-rbac \
    --node-vm-size Standard_B4ms
```


DEMO
#ntk19

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: azure-vote-front
spec:
  replicas: 1
  selector:
    matchLabels:
      app: azure-vote-front
  template:
    metadata:
      labels:
        app: azure-vote-front
    spec:
      containers:
      - name: azure-vote-front
        image: microsoft/azure-vote-front:v1
        resources:
          requests:
            cpu: 100m
            memory: 128Mi
          limits:
            cpu: 250m
            memory: 256Mi
        ports:
        - containerPort: 80
        env:
        - name: REDIS
          value: "azure-vote-back"
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: azure-vote-front
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: azure-vote-front
```

What about containers on Windows OS?

```
kubectl run windemo --image=microsoft/iis --replicas=2
```

```
# Sample Dockerfile
FROM microsoft/aspnet


# Creates an HTML file and adds content to this file.
RUN echo "Hello World - Dockerfile" > c:\inetpub\wwwroot\index.html

# Sets a command or process that will run each time a container is run from the new image.
CMD [ "cmd" ]


```

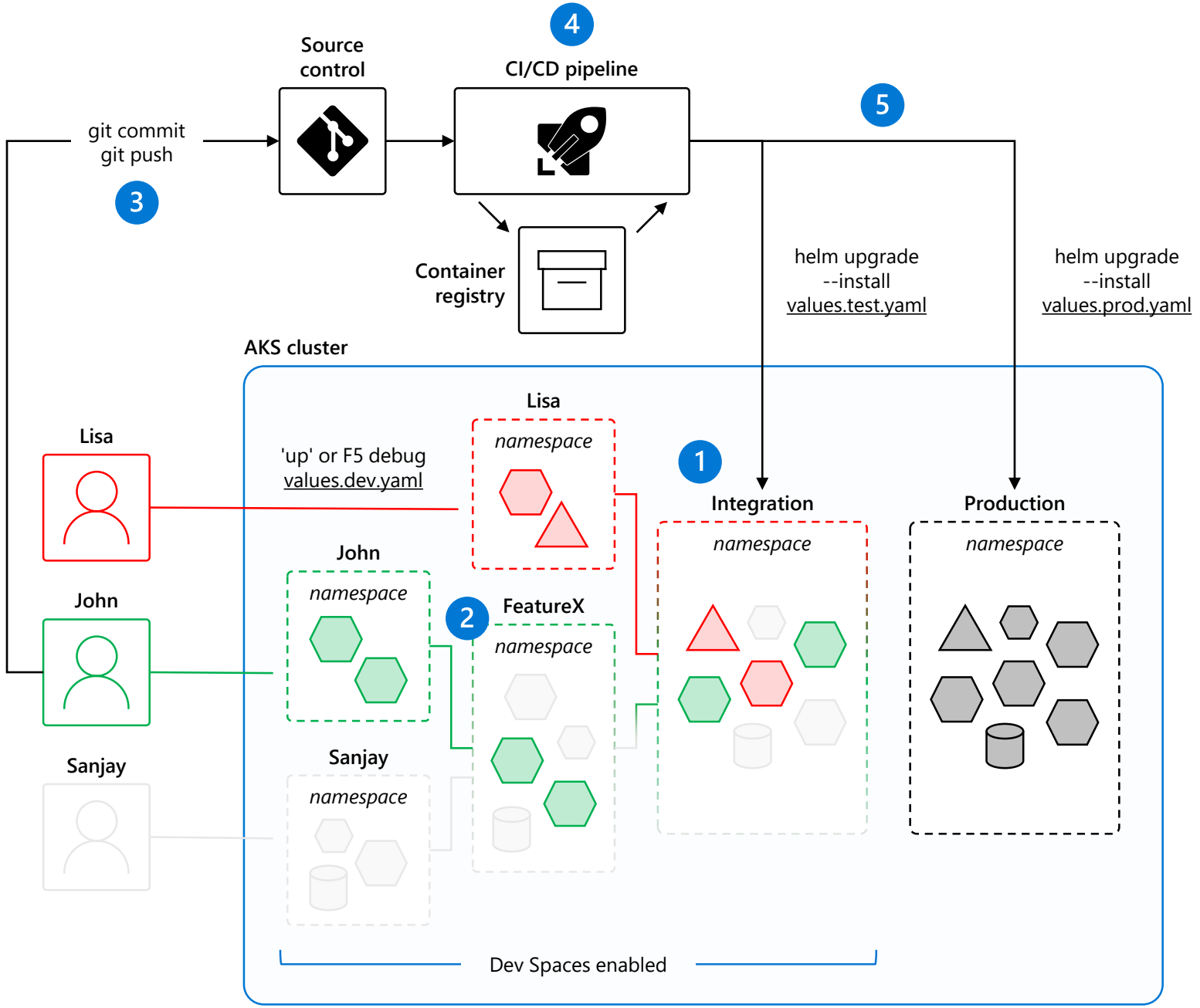
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample
  labels:
    app: sample
spec:
  replicas: 1
  template:
    metadata:
      name: sample
      labels:
        app: sample
    spec:
      nodeSelector:
        "beta.kubernetes.io/os": windows
      containers:
      - name: sample
        image: mcr.microsoft.com/dotnet/framework/samples:aspnetapp
        resources:
          limits:
            cpu: 1
            memory: 800m
          requests:
            cpu: .1
            memory: 300m
        ports:
          - containerPort: 80
  selector:
    matchLabels:
      app: sample
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: sample
spec:
  type: LoadBalancer
  ports:
  - protocol: TCP
    port: 80
  selector:
    app: sample
```

Dev Spaces

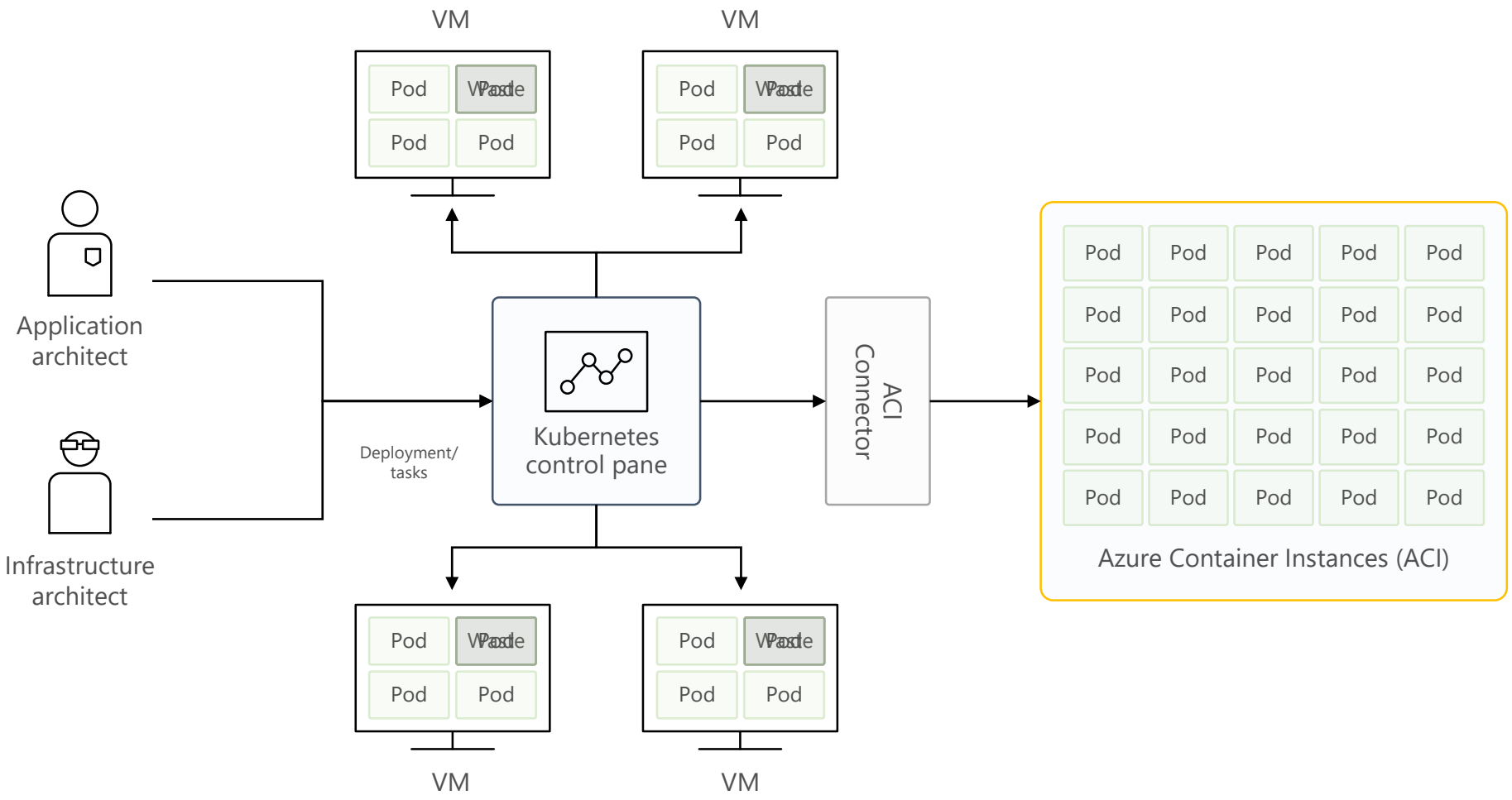
1. The “Integration” dev space is running a full baseline version of the entire application
2. John and Sanjay are collaborating on FeatureX; it is setup as a dev space and running all the modified services required to implement a feature
3. Code is committed to the master source control
4. A CI/CD pipeline can be triggered to deploy into “Integration,” which updates the team's baseline
5. The same Helm assets used during development are used in later environments by the CD system

*Dev Spaces is enabled per Kubernetes namespaces and can be defined as anything. Any namespace in which Dev Spaces is NOT enabled runs *unaffected*.*



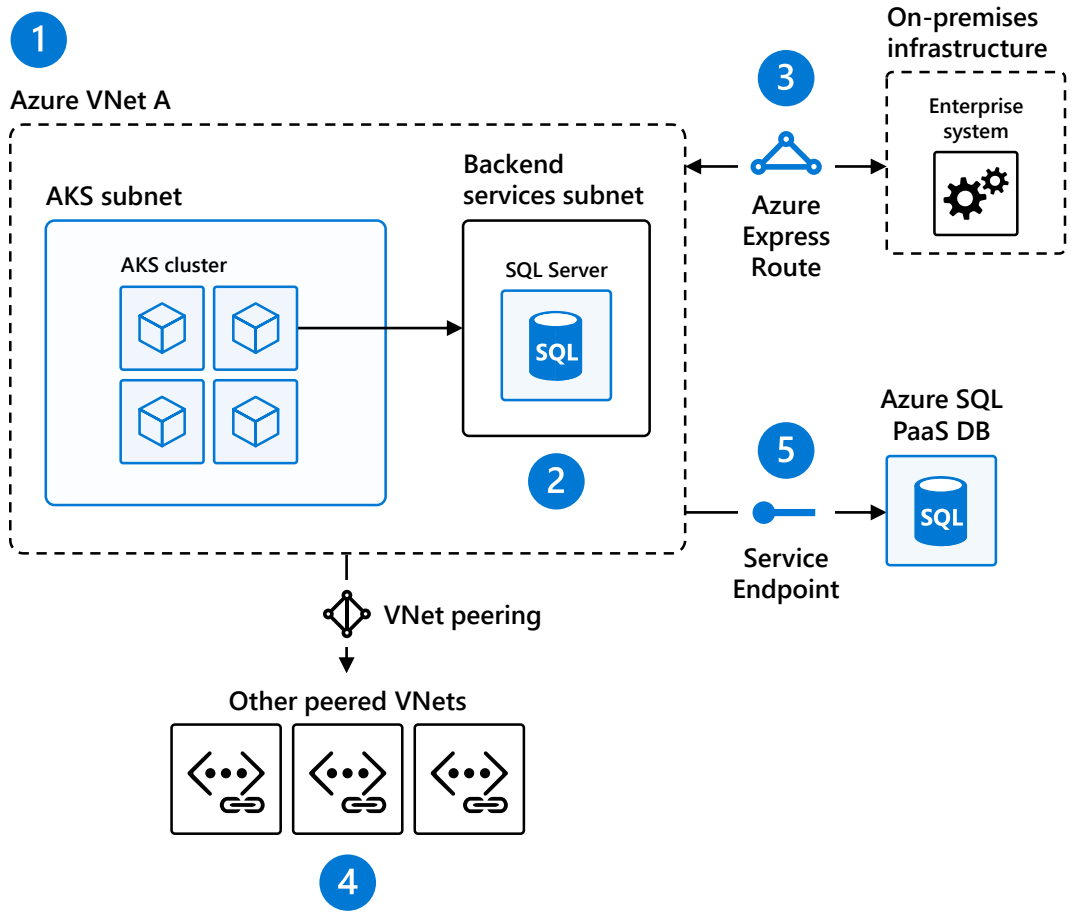

DEMO
#ntk19

Bursting with the ACI Connector



Secure network communications with VNET and CNI

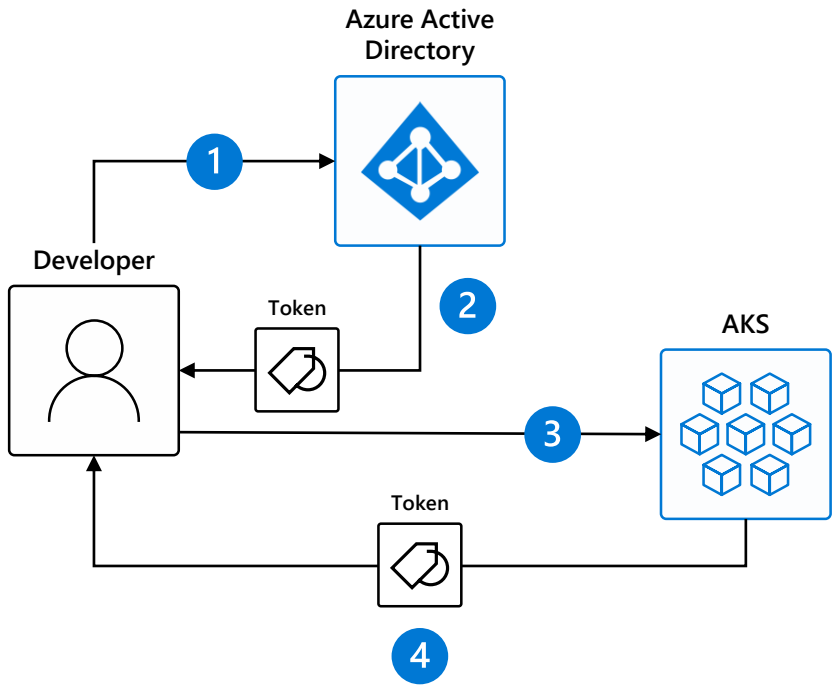
1. Uses Azure subnet for both your containers and cluster VMs
2. Allows for connectivity to existing Azure services in the same VNet
3. Use Express Route to connect to on-premises infrastructure
4. Use VNet peering to connect to other VNets
5. Connect AKS cluster securely and privately to other Azure resources using VNet endpoints



AKS VNet integration works seamlessly with your existing network infrastructure

Identity and access management through AAD and RBAC

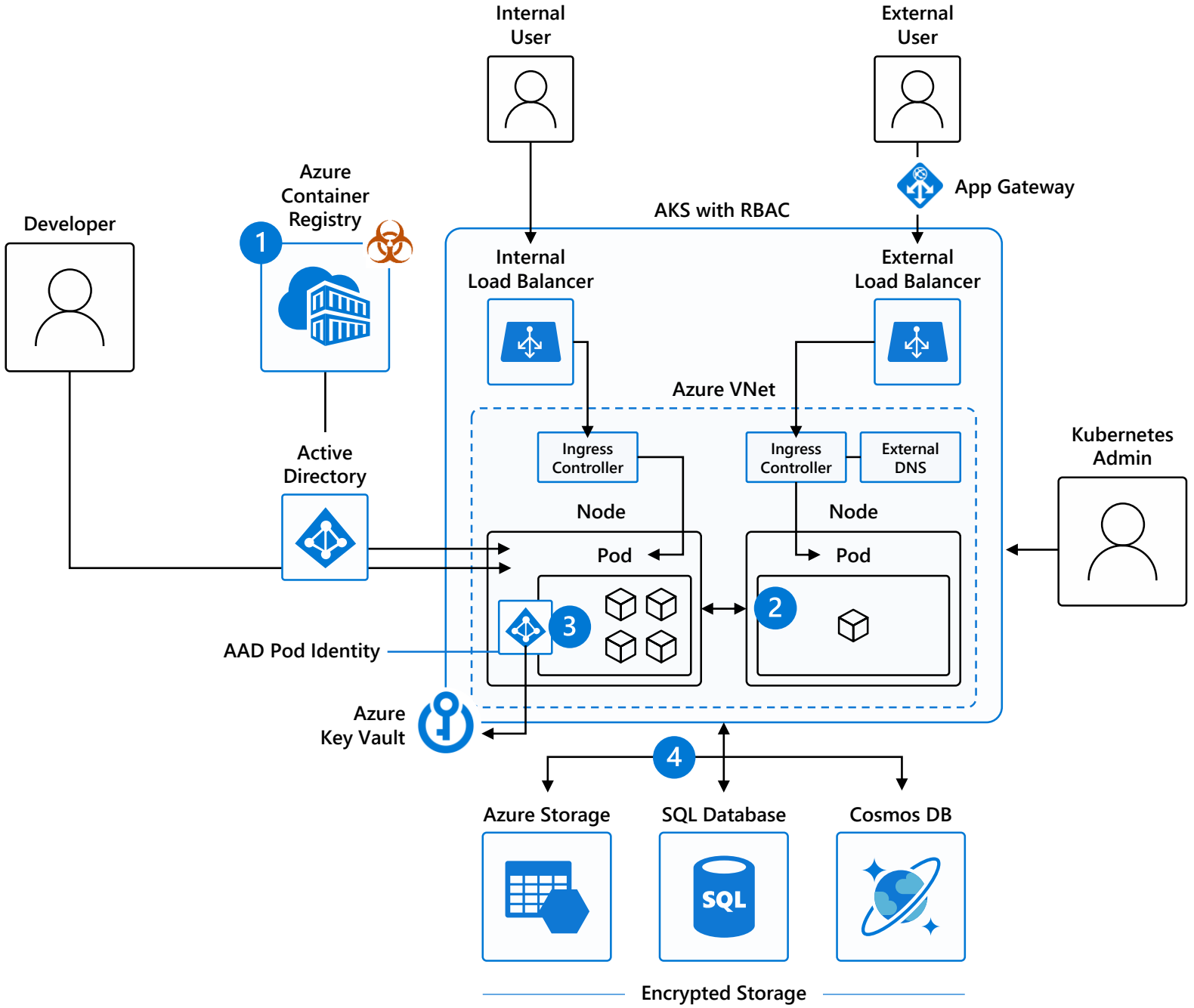
1. A developer authenticates to the AAD token issuance endpoint and requests an access token
2. The AAD token issuance endpoint issues the access token
3. The access token is used to authenticate to the secured resource
4. Data from the secured resource is returned to the web application



Azure delivers a streamlined identity and access management solution with Azure Active Directory (AAD) and Azure Kubernetes Services (AKS)

Security overview

1. Image and container level security
   - AAD authenticated Container registry access
   - ACR image scanning and content trust for image validation
2. Node and cluster level security
   - Automatic security patching nightly
   - Nodes deployed in private virtual network subnet w/o public addresses
   - Network policy to secure communication paths between namespaces (and nodes)
   - Pod Security Policies
   - K8s RBAC and AAD for authentication
3. Pod level security
   - Pod level control using AAD Pod Identity
   - Pod Security Context
4. Workload level security
   - Azure Role-based Access Control (RBAC) & security policy groups
   - Secure access to resources & services (e.g. Azure Key Vault) via Pod Identity
   - Storage Encryption
   - App Gateway with WAF to protect against threats and intrusions



Azure container summary