Azure

Kubernetes Hackfest

Kubernetes and Containers on Azure

Table of contents

Introduction

Azure Kubernetes Service Overview

Top scenarios

Customer stories

Open source culture

Resources

Product deep dive

Azure + open source momentum

Azure is a strong platform for Open Source

Linux VMs are growing at ~2 times Windows VMs today

Microsoft announced GitHub acquisition

1 in 3 VMs on Azure are Linux

~60% of 3rd party Azure Marketplace images are open source

redhat

suse

CLOUD FOUNDRY

docker

HashiCorp Terraform

Pivotal

Jenkins

Partnerships

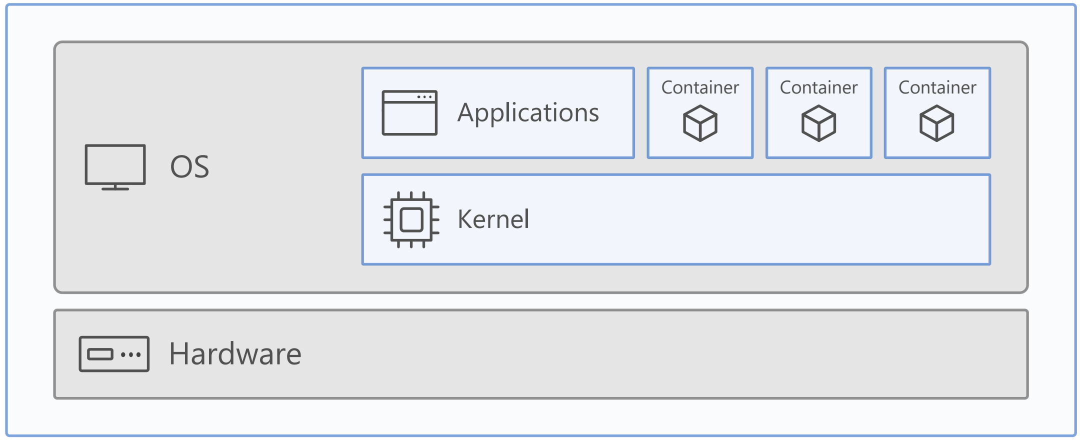
“Microsoft Joins Cloud Native Computing Foundation as Platinum Member”

Introduction

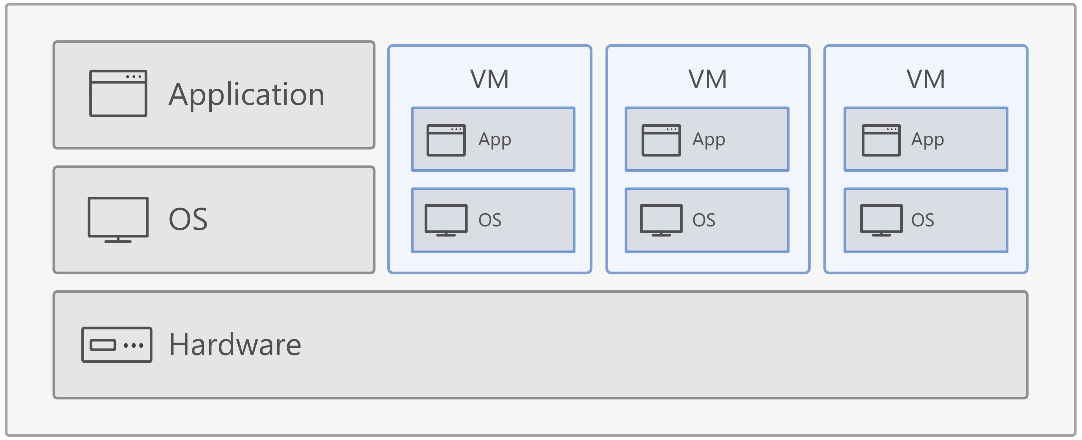
- Introduction
- Azure Kubernetes Service Overview
- Top scenarios
- Customer stories
- Open source culture
- Resources
- Product deep dive

What is a **container**?

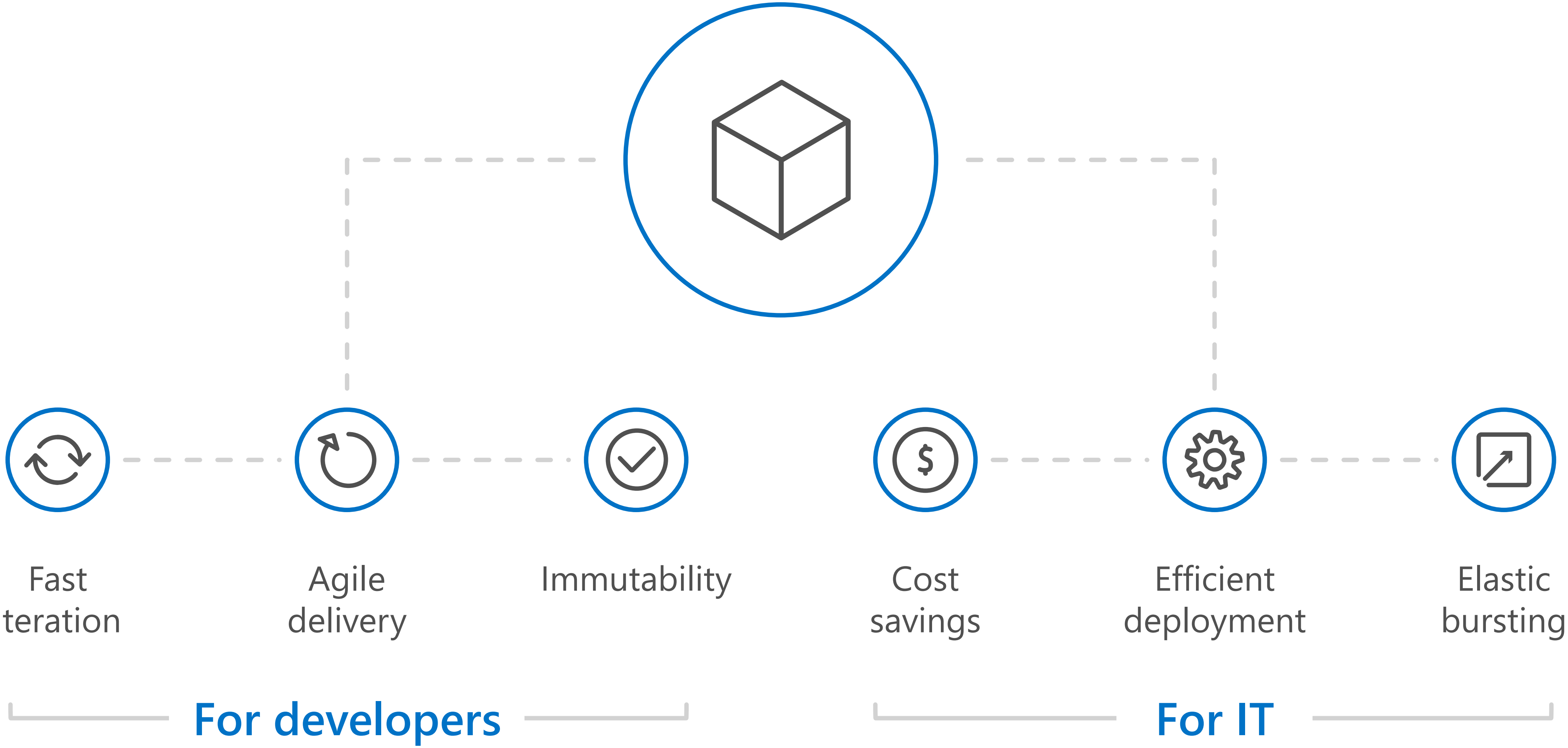
Containers = operating system virtualization



Traditional virtual machines = hardware virtualization



The container **advantage**



Fast iteration

Agile delivery

Immutability

Cost savings

Efficient deployment

Elastic bursting

For developers

For IT

Why **containers?**

- Repeatable execution
  - immutable environment
  - reusable and portable code (“Build, Ship, and Run”)
- Consistency across development, test, & production
- Fast & agile app deployment; instant startup
- Cloud portability
- Density, partitioning, scale
- Diverse developer framework support
- Promotes microservices

The elements of **orchestration**

Scheduling

Affinity/anti-affinity

Health monitoring

Failover

Scaling

Networking

Service discovery

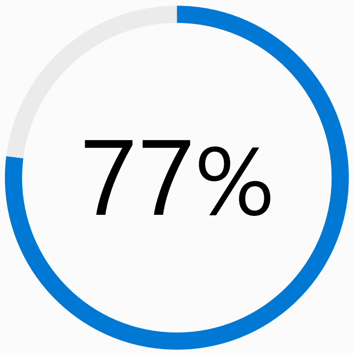
Coordinated app upgrades

Containers and Kubernetes **momentum**

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

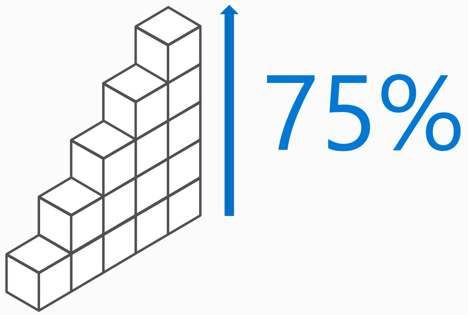
Gartner

Half of container environment is orchestrated.[1]

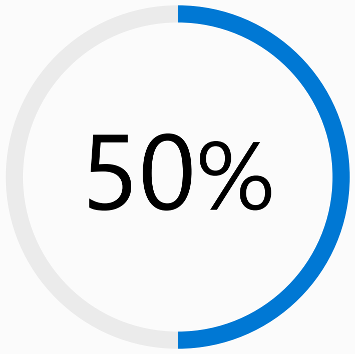


77% of companies[2] who use container orchestrators choose Kubernetes.

The average size of a container deployment has grown 75% in one year. [1]



Larger companies are leading the adoption.[1]

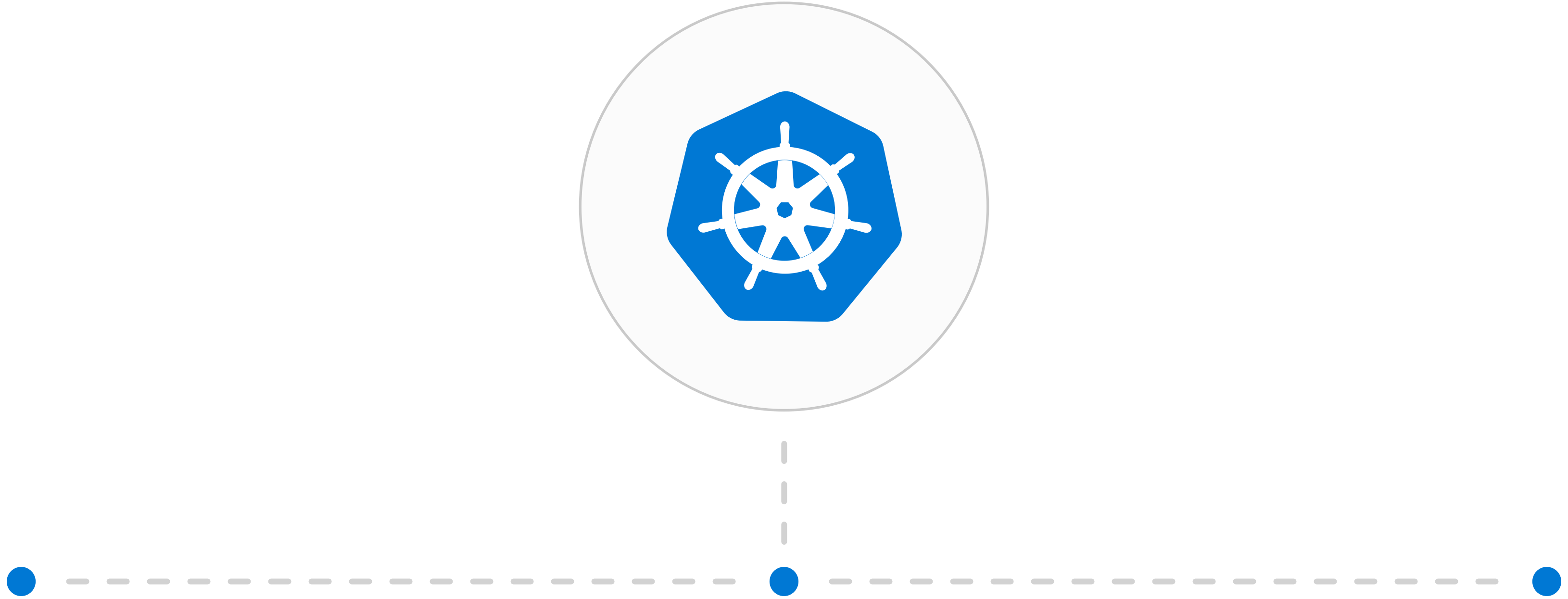


Nearly **50%** of organizations[1] running 1000 or more hosts have adopted containers.

[1] Datadog report: 8 Surprising Facts About Real Docker Adoption

[2] CNCF survey: cloud-native-technologies-scaling-production-applications

Kubernetes: the industry leading orchestrator



Portable

Public, private, hybrid,
multi-cloud

Extensible

Modular, pluggable,
hookable, composable

Self-healing

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

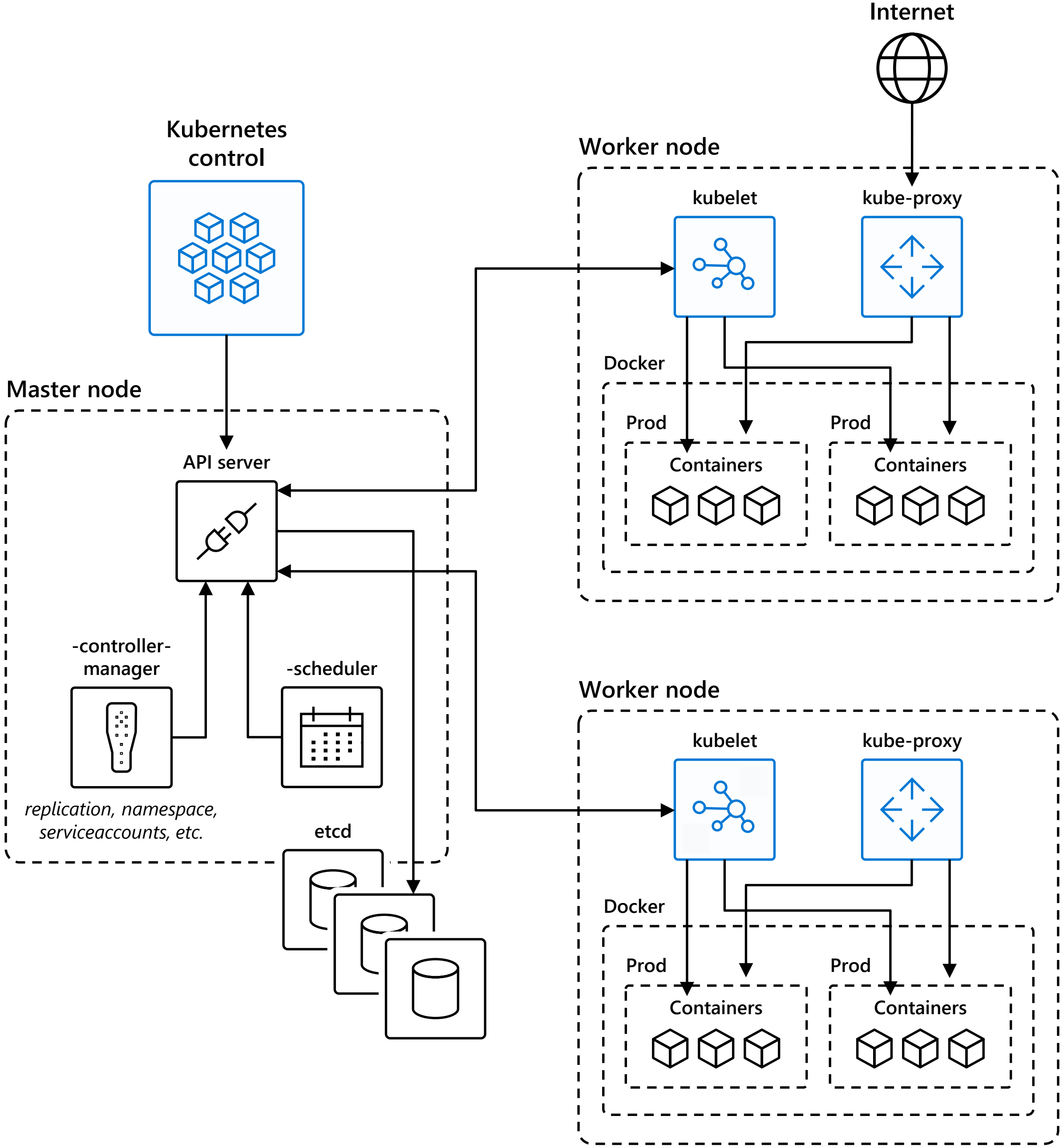
Kubernetes 101

1. Kubernetes users communicate with API server and apply desired state

2. Master nodes actively enforce desired state on worker nodes

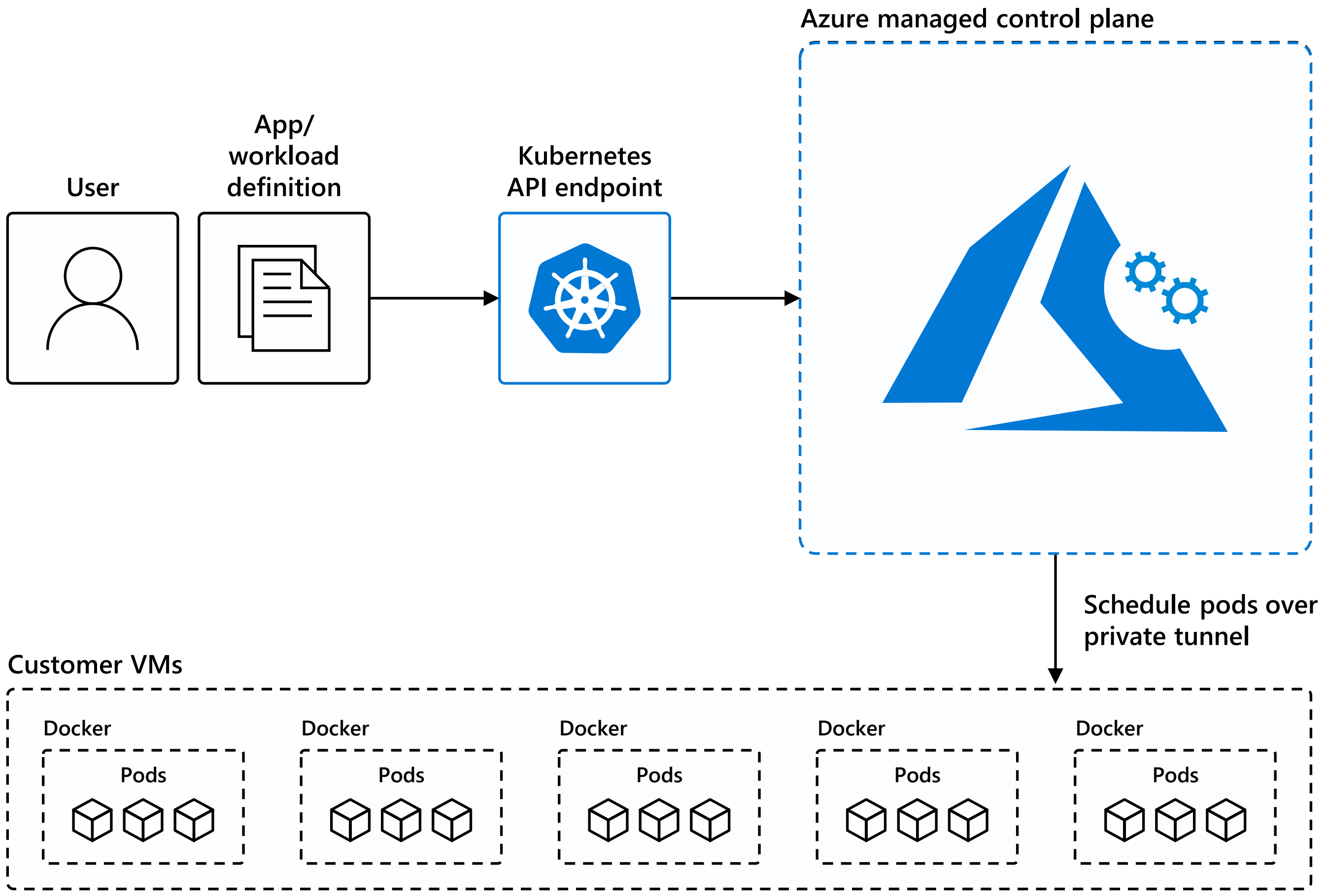
3. Worker nodes support communication between containers

4. Worker nodes support communication from the Internet



How managed Kubernetes on Azure works

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



From infrastructure to **innovation**

Managed Kubernetes empowers you to do more

Focus on your containers and code, not the plumbing of them

Responsibilities	DIY with Kubernetes	Managed Kubernetes on Azure
Containerization	⬜	◪
Application iteration, debugging	⬜	◪
CI/CD	⬜	◪
Cluster hosting	⬜	🟦
Cluster upgrade	⬜	🟦
Patching	⬜	🟦
Scaling	⬜	🟦
Monitoring and logging	⬜	🟦

⬜	Customer
🟦	Microsoft

Azure Kubernetes Service Overview

Introduction

Azure Kubernetes Service Overview

Top scenarios

Customer stories

Open source culture

Resources

Product deep dive

AKS: Simplify the deployment, management, and operations of Kubernetes

Deploy and manage Kubernetes with ease

Accelerate containerized application development

Set up CI/CD in a few clicks

Secure your Kubernetes environment

Scale and run applications with confidence

Work how you want with open-source tools & APIs

Azure Kubernetes Service (AKS)

Kubernetes made easy – get the most complete and simple end-to-end experience for seamless Kubernetes lifecycle management on Azure

Deploy and manage Kubernetes with ease • Free managed control plane for auto upgrades, patching and self healing • Provision with portal, CLI, ARM, or Terraform • Full visibility with integrated monitoring and logging	**Scale and run applications with confidence** • Built-in auto scaling • Global data center to boost performance and reach • Elastically burst from AKS cluster using ACI	**Secure your Kubernetes environment** • Control access through AAD and RBAC • Safeguard keys and secrets with Key Vault • Secure network communication with VNET and CNI
Accelerate containerized application development • Define, install and upgrade apps easily with Helm • Automatically scaffold, containerize and deploy with CLI or Visual Studio • Rapidly iterate, test and debug microservices using Dev Spaces	**Work how you want with open-source tools & APIs** • 100% open source Kubernetes • Take full advantage of services and tools in the ecosystem • Easily integrate with SLA-backed Azure services with OSBA	**Set up CI/CD in a few clicks** • Three steps away from a CI/CD pipeline with DevOps Project • Work with existing tools such as Jenkins • Geo-replicated container registry

Work how you want with opensource tools and APIs

	Development	DevOps	Monitoring	Networking	Storage	Security
Take advantage of services and tools in the Kubernetes ecosystem	HELM DRAFT	Jenkins Terraform BRIGADE JFrog CODESHIP HASHICORP	Prometheus fluentd Grafana OPENTRACING DATADOG JAEGER	CNI Networking TIGERA	MAPR portworx	Twistlock aqua heptio RBAC
OR, Leverage growing Azure support	VS Code	VSTS ARM	Azure Monitor	Azure VNET	Azure Storage	Azure Container Registry AAD Key Vault

Secure your Kubernetes environment

Control access through AAD and RBAC

Safeguard keys and secrets with Key Vault

Secure network communications with VNET and CNI

Compliant Kubernetes service with certifications covering SOC, HIPAA, and PCI

Scale and run applications with confidence

Built-in auto scaling

Global data center to boost performance and reach

Elastically burst from AKS cluster using ACI

Geo-replicated container registry

Top scenarios

Introduction

Azure Kubernetes Service Overview

Top scenarios

Customer stories

Open source culture

Resources

Product deep dive

Top scenarios for Kubernetes on Azure

Lift and shift to containers

Cost saving

without refactoring your app

Microservices

Agility

Faster application development

Machine learning

Performance

Low latency processing

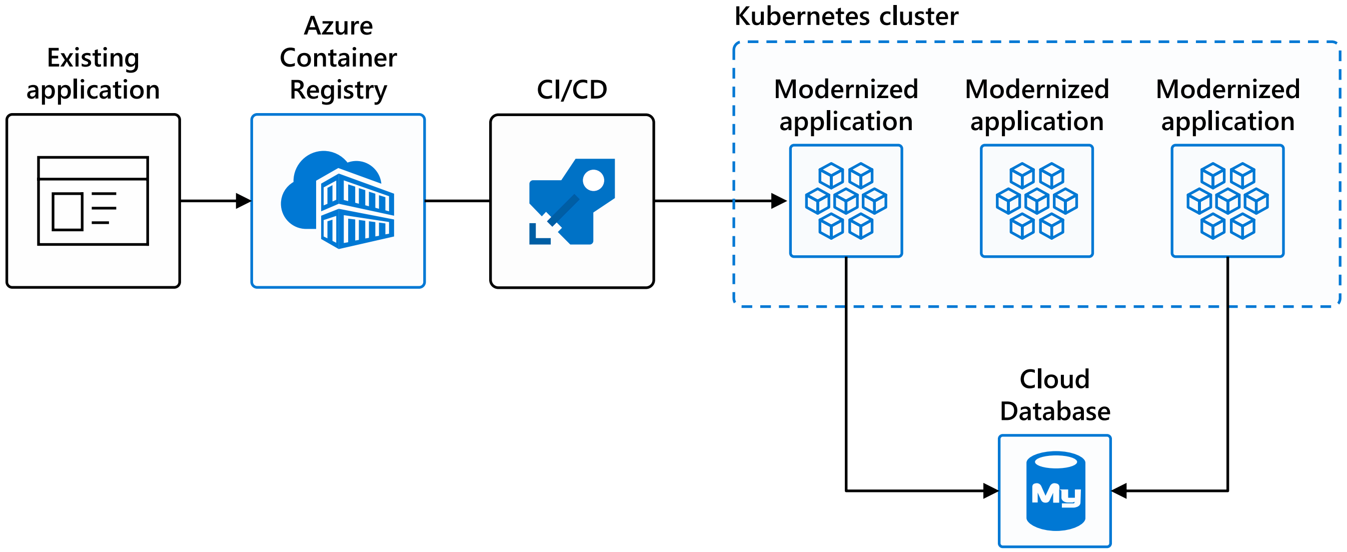
IoT

Portability

Build once, run anywhere

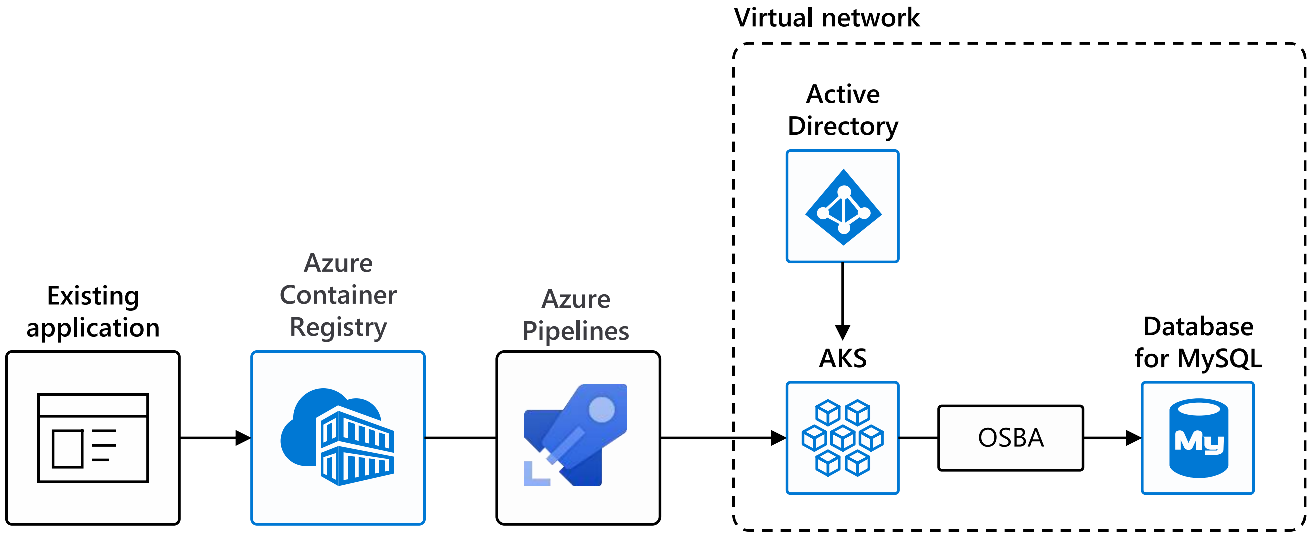
App modernization without code changes

- Speed application deployments by using container technology
- Defend against infrastructure failures with container orchestration
- Increase agility with continuous integration and continuous delivery



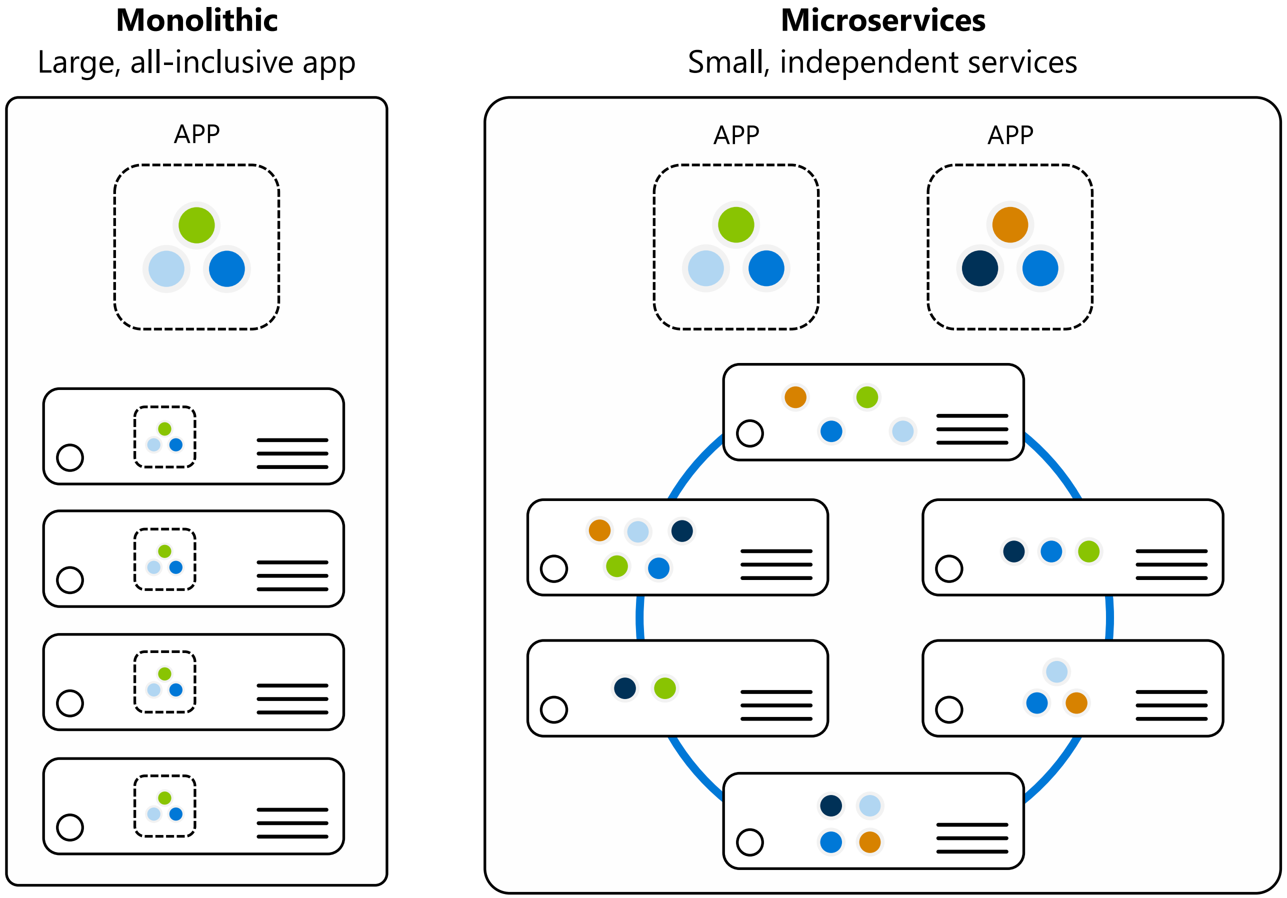
Capabilities

1. Use Azure Container Registry to store container images for your modernized applications, replicated globally with ACR geo-replication
2. Integrate AKS with Azure Pipelines to enable continuous integration/continuous delivery (CI/CD) using Helm or other Kubernetes ecosystem tooling
3. Enhance security with Azure Active Directory and RBAC to control access to AKS resources
4. Easily access to SLA-backed Azure Services such as Azure Database for MySQL using Open Service Broker for Azure (OSBA)



Microservices: for faster app development

- Independent deployments
- Improved scale and resource utilization per service
- Smaller, focused teams



Monolithic
Large, all-inclusive app

Microservices
Small, independent services

Lift and shift to containers

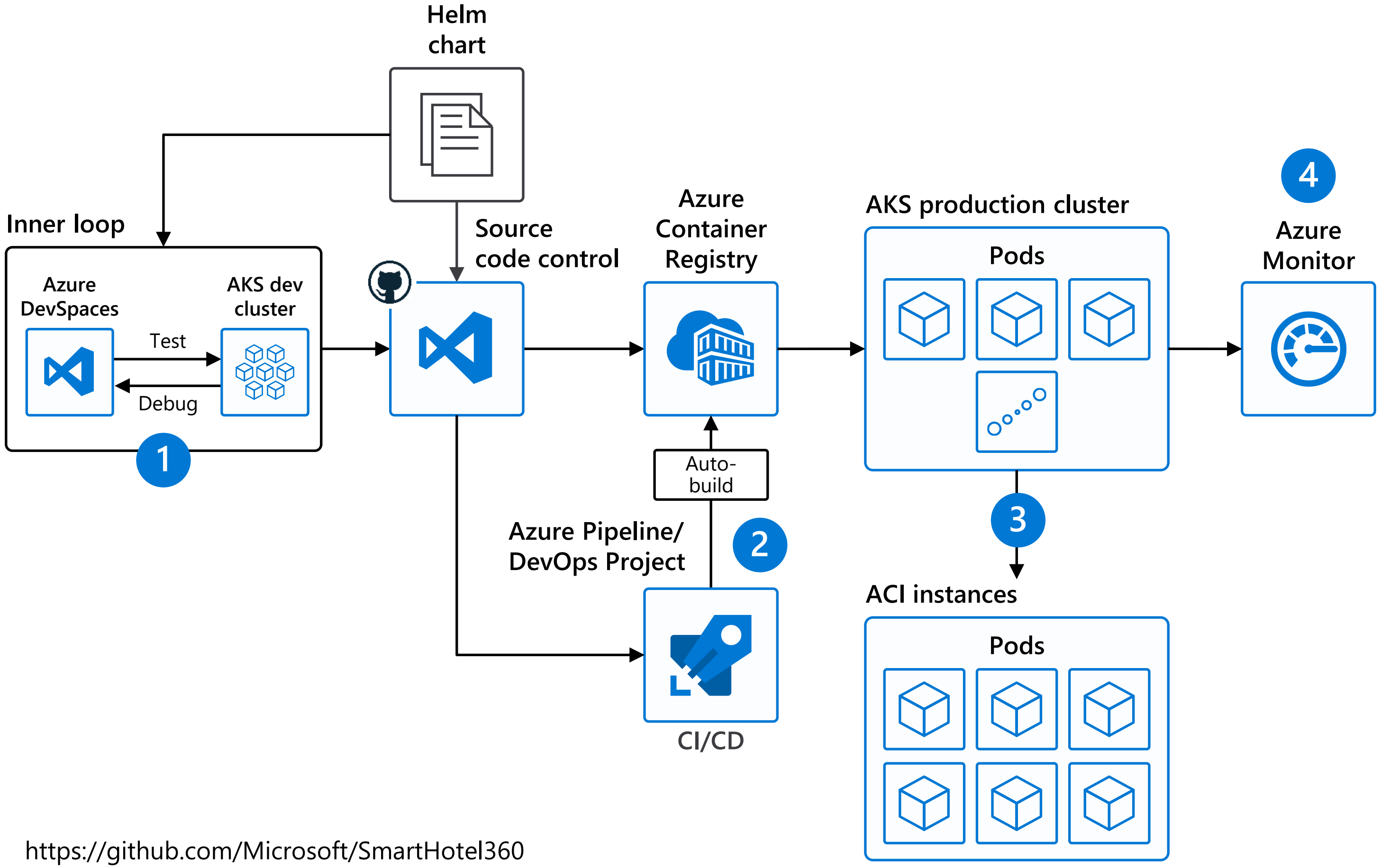
Microservices

Machine learning

IoT

Capabilities

1. Use Azure Dev Spaces to iteratively develop, test, and debug microservices targeted for AKS clusters.
2. Azure DevOps has native integration with Helm and helps simplifying continuous integration/continuous delivery (CI/CD)
3. Virtual node—a Virtual Kubelet implementation—allows fast scaling of services for unpredictable traffic using ACI.
4. Azure Monitor provides a single pane of glass for monitoring over app telemetry, cluster-to-container level health analytics.

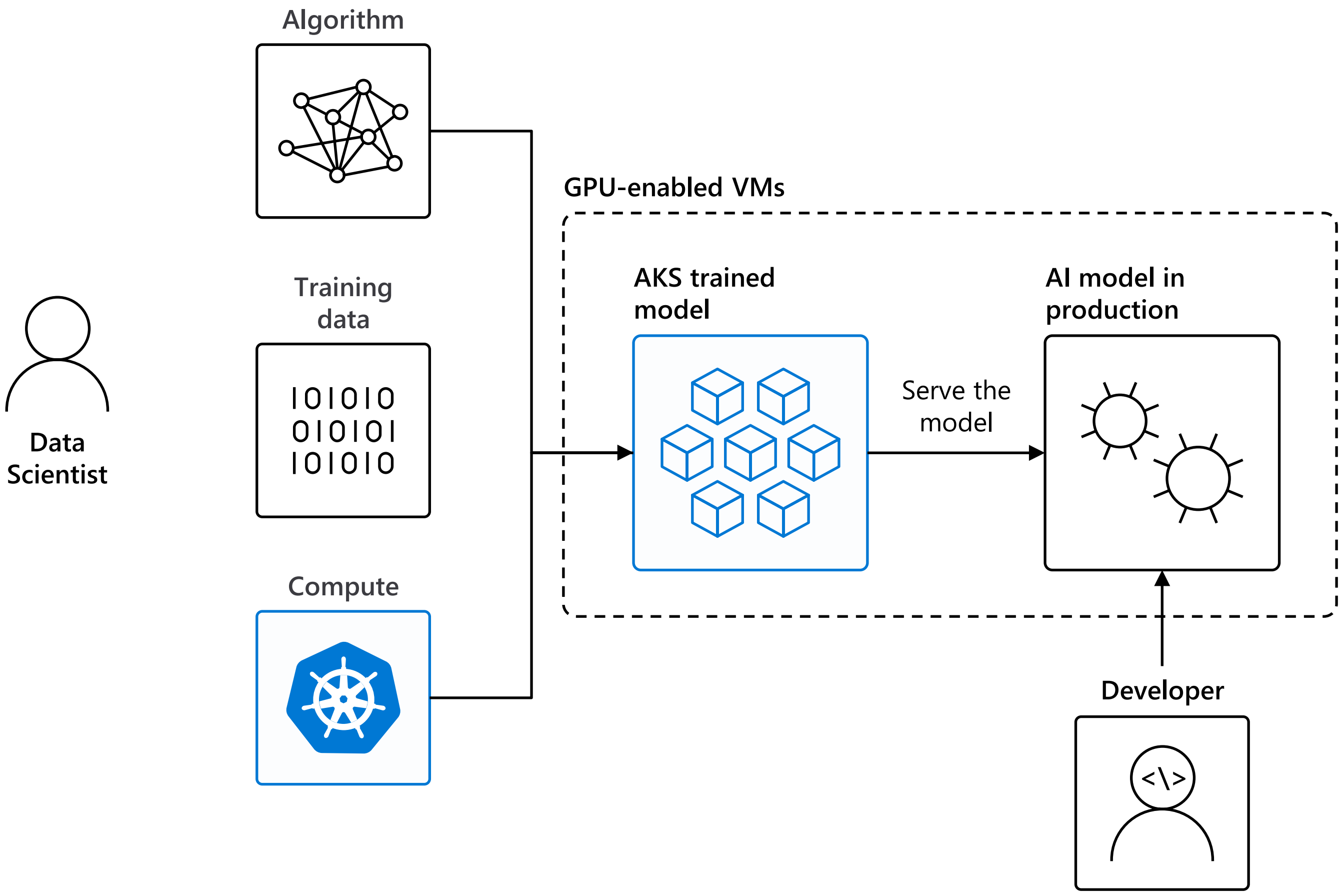


https://github.com/Microsoft/SmartHotel360-AKS-DevSpaces-Demo

Data science in a box

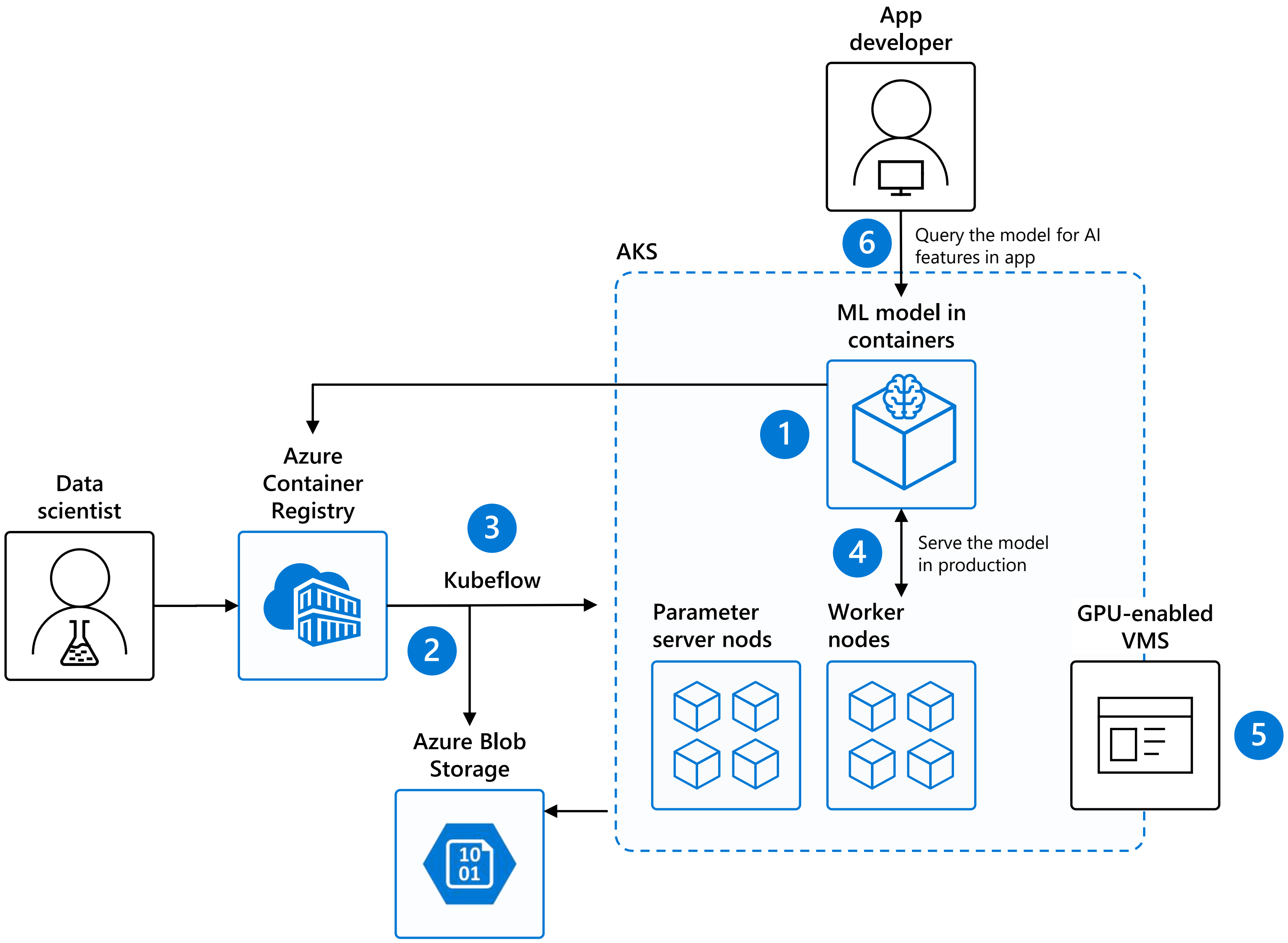
- Quick deployment and high availability
- Low latency data processing
- Consistent environment across test, control and production

https://github.com/Azure/kubeflow-labs



Capabilities

1. Package ML model into a container and publish to ACR
2. Azure Blob storage hosts training data sets and trained model
3. Use Kubeflow to deploy training job to AKS, distributed training job to AKS includes Parameter servers and Worker nodes…
4. Serve production model using Kubeflow, promoting a consistent environment across test, control and production
5. AKS supports GPU enabled VM
6. Developer can build features querying the model running in AKS cluster



https://github.com/Azure/kubeflow-labs

Lift and shift to containers

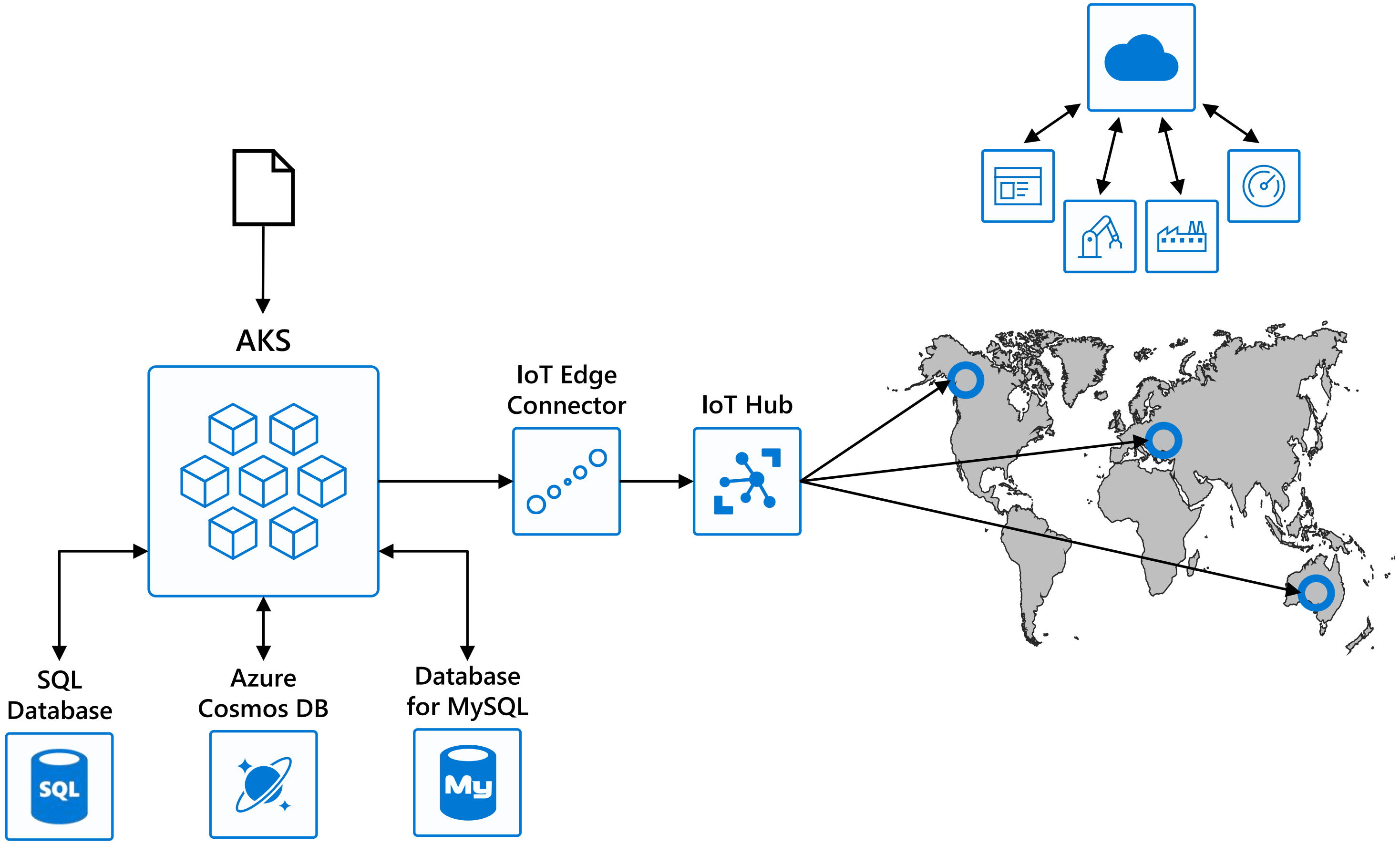
Microservices

Machine learning

IoT

Scalable Internet of Things solutions

- Portable code, runs anywhere
- Elastic scalability and manageability
- Quick deployment and high availability



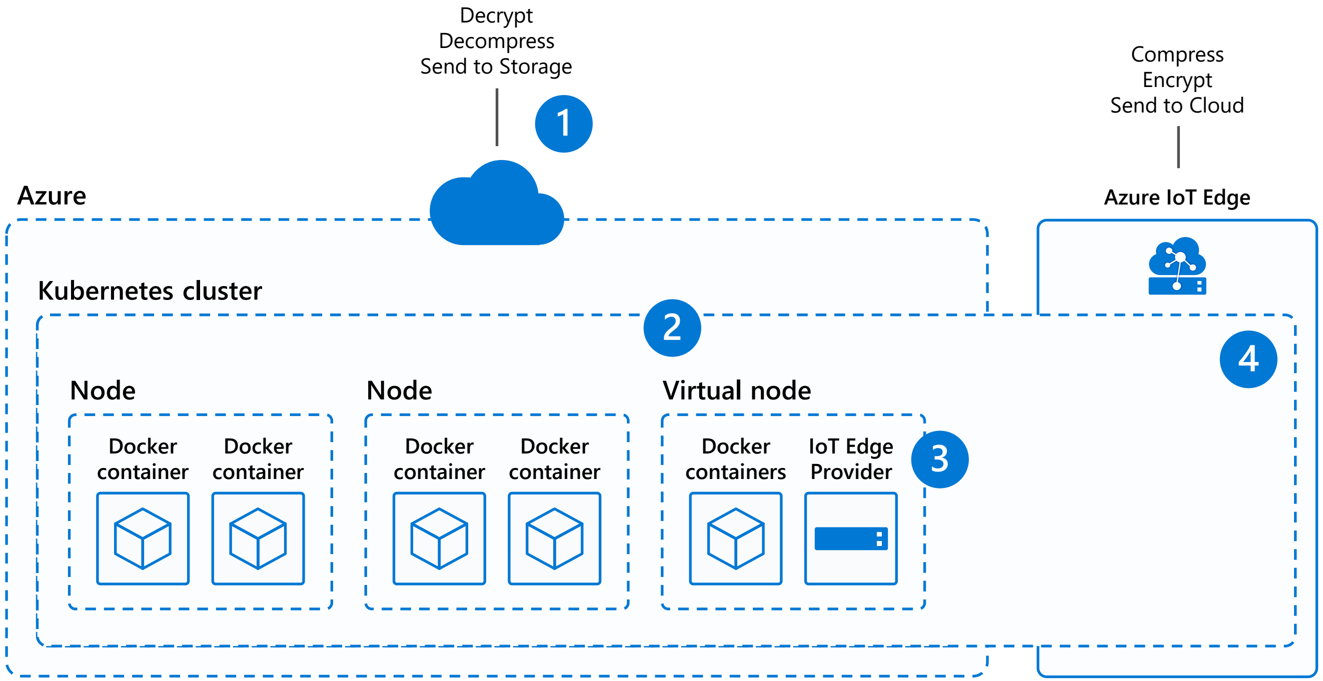
Capabilities

1. Azure IoT Edge encrypts data and send to Azure, which then decrypts the data and send to storage

2. Virtual node, an implementation of Virtual Kubelet, serves as the translator between cloud and edge

3. IoT Edge Provider in virtual node redirects containers to IoT Edge and extend AKS cluster to target millions of Edge devices

4. Consistent update, manage, and monitoring as one unit in AKS using single pod definition



Product deep dive

Introduction

Azure Kubernetes Service Overview

Top scenarios

Customer stories

Open source culture

Resources

Product deep dive

What is Kubernetes?

Background

- "Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications”
- Schedules and runs application containers across a cluster of machines
- Kubernetes v1.0 released on July 21, 2015. Joe Beda, Brendan Burns, & Craig McLuckie

Key features

- Declarative infrastructure
- Self-healing
- Horizontal scaling
- Automated rollouts and rollbacks
- Service discovery and load balancing
- Automatic bin packing
- Storage orchestration
- Secret and configuration management
- Not a PaaS platform

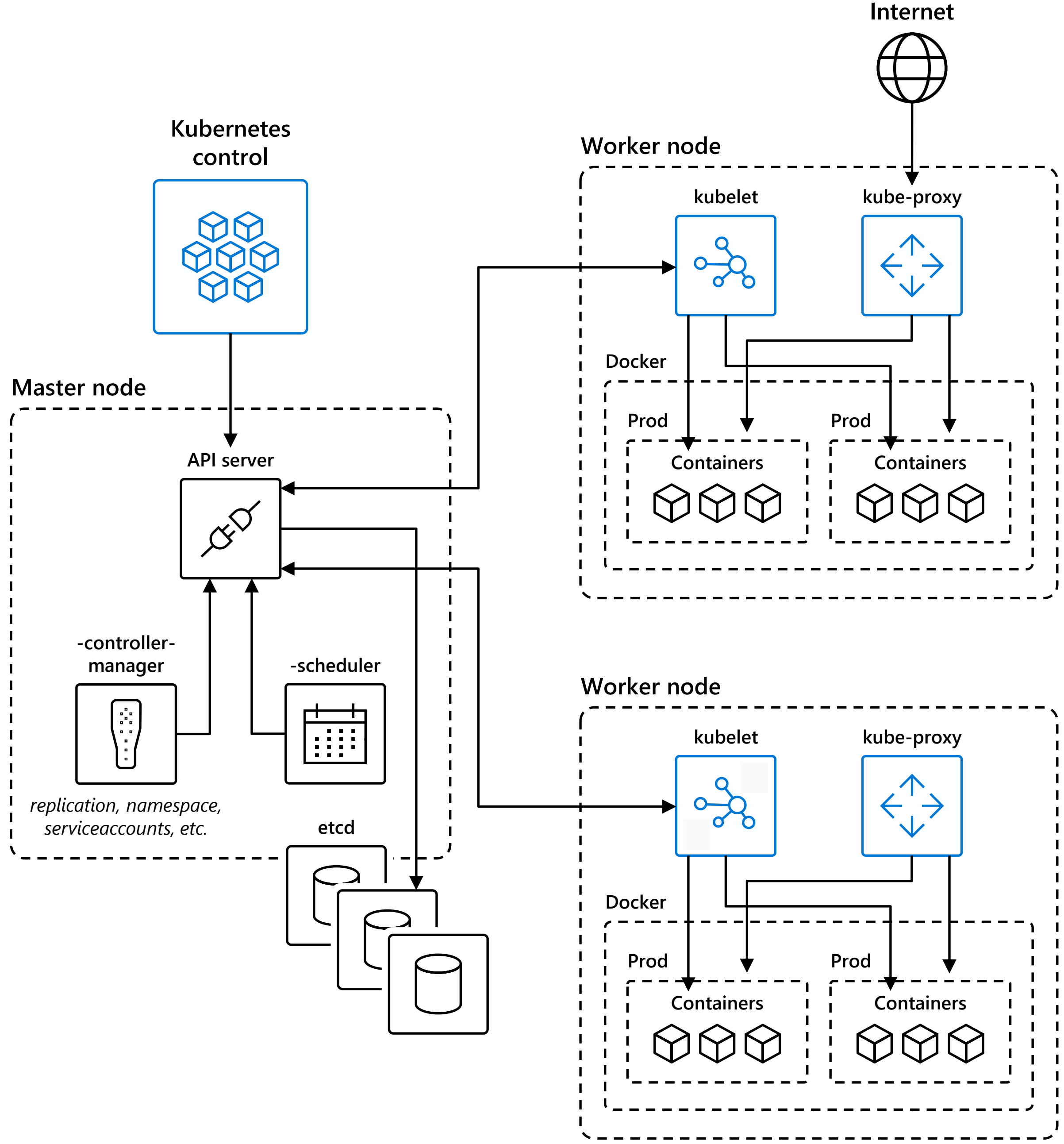
Kubernetes 101

1. Kubernetes users communicate with API server and apply desired state

2. Master nodes actively enforce desired state on worker nodes

3. Worker nodes support communication between containers

4. Worker nodes support communication from the Internet



Kubernetes Resources

pod	deployment
service	Replica set
ingress	Daemon set job
namespace	Secret Config-map

Kubernetes Resources: Namespaces

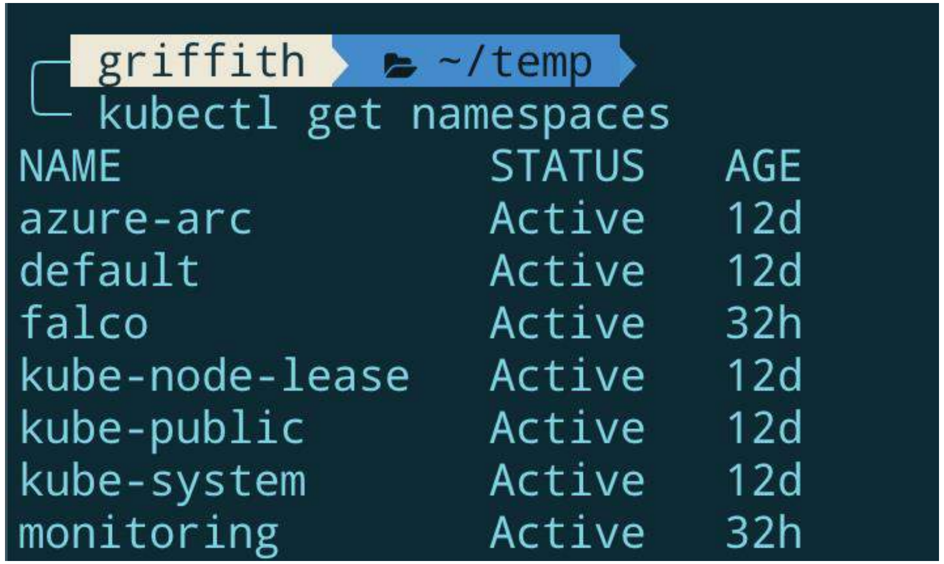
Provide grouping or Kubernetes resources

to enable:

- RBAC
- Affinity
- Quotas
- Policy (Cluster & Network)

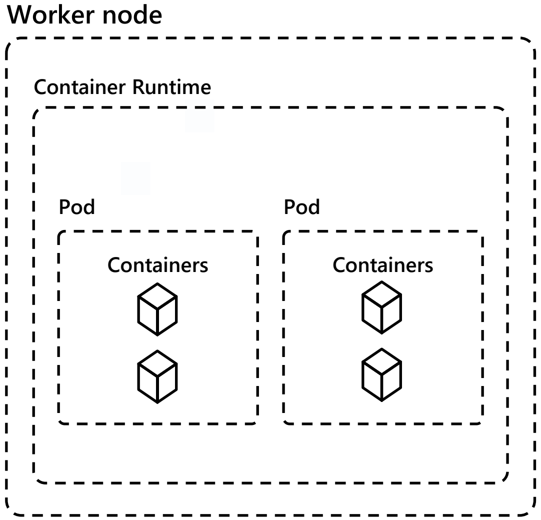
Default K8s Namespaces:

- default
- kube-node-lease
- kube-public
- kube-system



```
 griffith  ~/temp
 kubectl get namespaces
NAME              STATUS   AGE
azure-arc         Active   12d
default           Active   12d
falco             Active   32h
kube-node-lease   Active   12d
kube-public       Active   12d
kube-system       Active   12d
monitoring        Active   32h
```

Kubernetes Resources: Pods

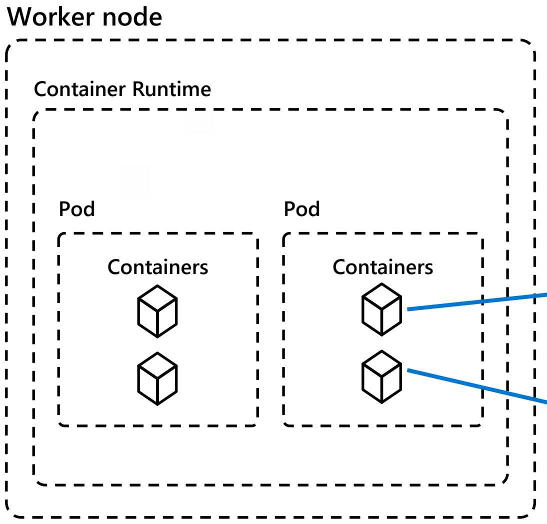


Zero to Many Containers Per Pod

Multi-Container Common Patterns:

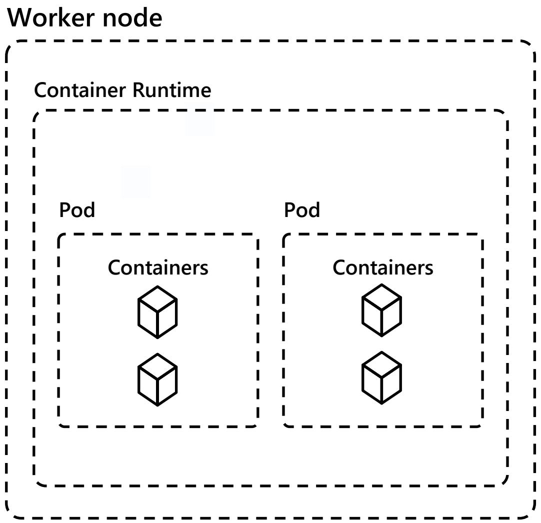
- Sidecar: Enhance or extend a container
- Ambassador: Proxy network calls
- Adapter: Transform output

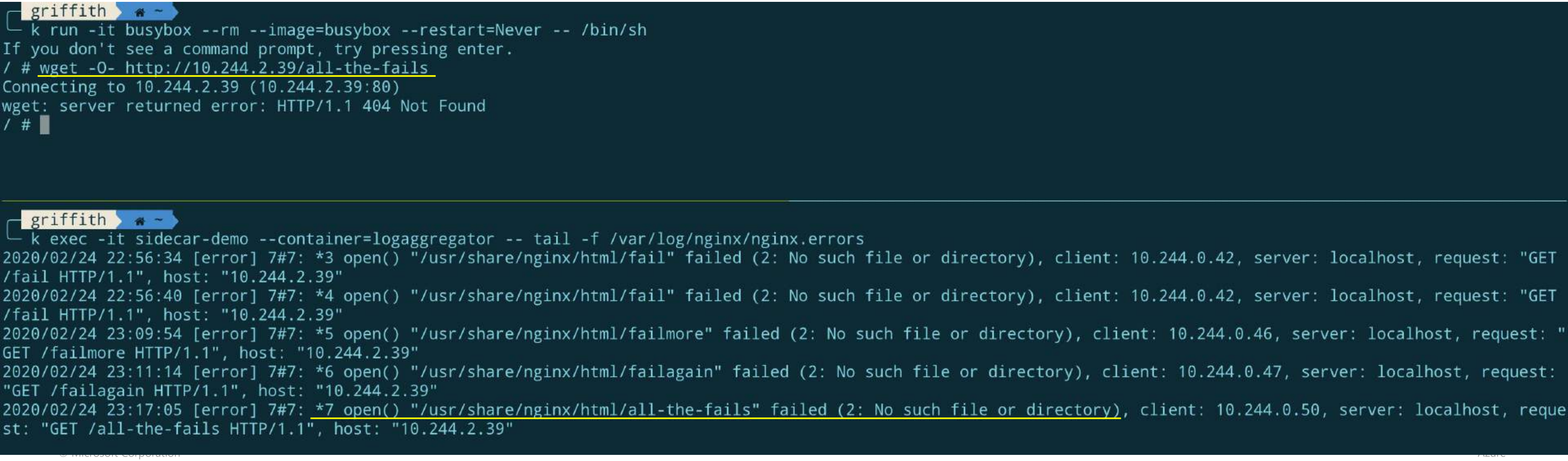
Kubernetes Resources: Pods - Adapter Example



```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: sidecar-demo
  name: sidecar-demo
spec:
  containers:
  - image: nginx
    name: webserver
    volumeMounts:
    - mountPath: /var/log/nginx/
      name: log-volume
    resources: {}
  - image: busybox
    name: logaggregator
    args: [/bin/sh, -c, 'tail -f
/var/log/nginx/error.log>/var/log/nginx/nginx.errors']
    volumeMounts:
    - mountPath: /var/log/nginx/
      name: log-volume
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Never
  volumes:
  - name: log-volume
    emptyDir: {}
```

Kubernetes Resources:
Pods – Adapter/Sidecar Example





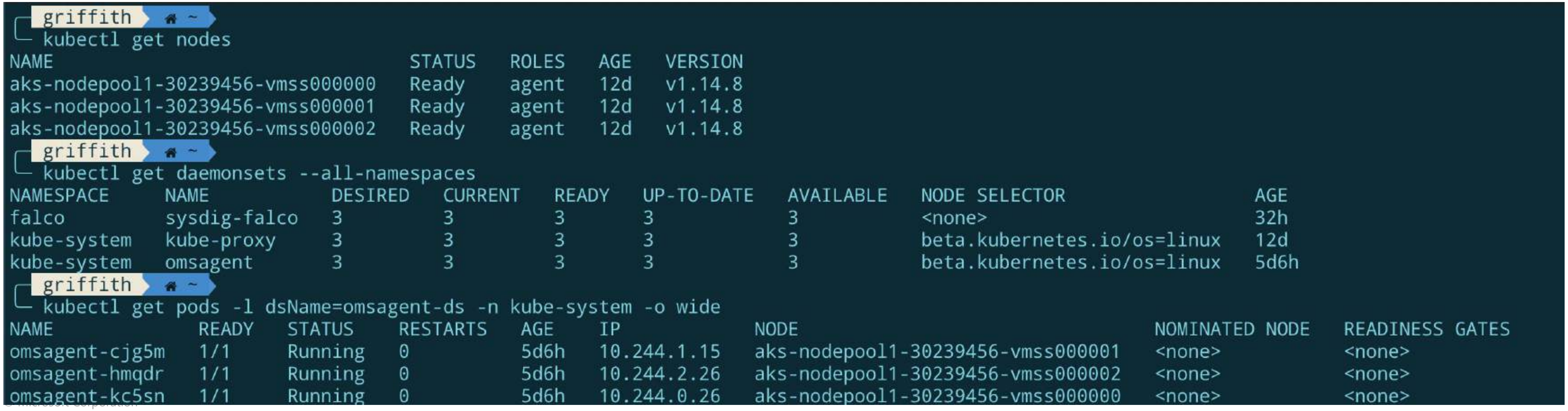
```
┌ griffith  ~
└ k run -it busybox --rm --image=busybox --restart=Never -- /bin/sh
If you don't see a command prompt, try pressing enter.
/ # wget -O- http://10.244.2.39/all-the-fails
Connecting to 10.244.2.39 (10.244.2.39:80)
wget: server returned error: HTTP/1.1 404 Not Found
/ #
```

```
┌ griffith  ~
└ k exec -it sidecar-demo --container=logaggregator -- tail -f /var/log/nginx/nginx.errors
2020/02/24 22:56:34 [error] 7#7: *3 open() "/usr/share/nginx/html/fail" failed (2: No such file or directory), client: 10.244.0.42, server: localhost, request: "GET
/fail HTTP/1.1", host: "10.244.2.39"
2020/02/24 22:56:40 [error] 7#7: *4 open() "/usr/share/nginx/html/fail" failed (2: No such file or directory), client: 10.244.0.42, server: localhost, request: "GET
/fail HTTP/1.1", host: "10.244.2.39"
2020/02/24 23:09:54 [error] 7#7: *5 open() "/usr/share/nginx/html/failmore" failed (2: No such file or directory), client: 10.244.0.46, server: localhost, request: "
GET /failmore HTTP/1.1", host: "10.244.2.39"
2020/02/24 23:11:14 [error] 7#7: *6 open() "/usr/share/nginx/html/failagain" failed (2: No such file or directory), client: 10.244.0.47, server: localhost, request:
"GET /failagain HTTP/1.1", host: "10.244.2.39"
2020/02/24 23:17:05 [error] 7#7: *7 open() "/usr/share/nginx/html/all-the-fails" failed (2: No such file or directory), client: 10.244.0.50, server: localhost, reque
st: "GET /all-the-fails HTTP/1.1", host: "10.244.2.39"
```

Kubernetes Resources: DaemonSet

Runs a given pod on every node

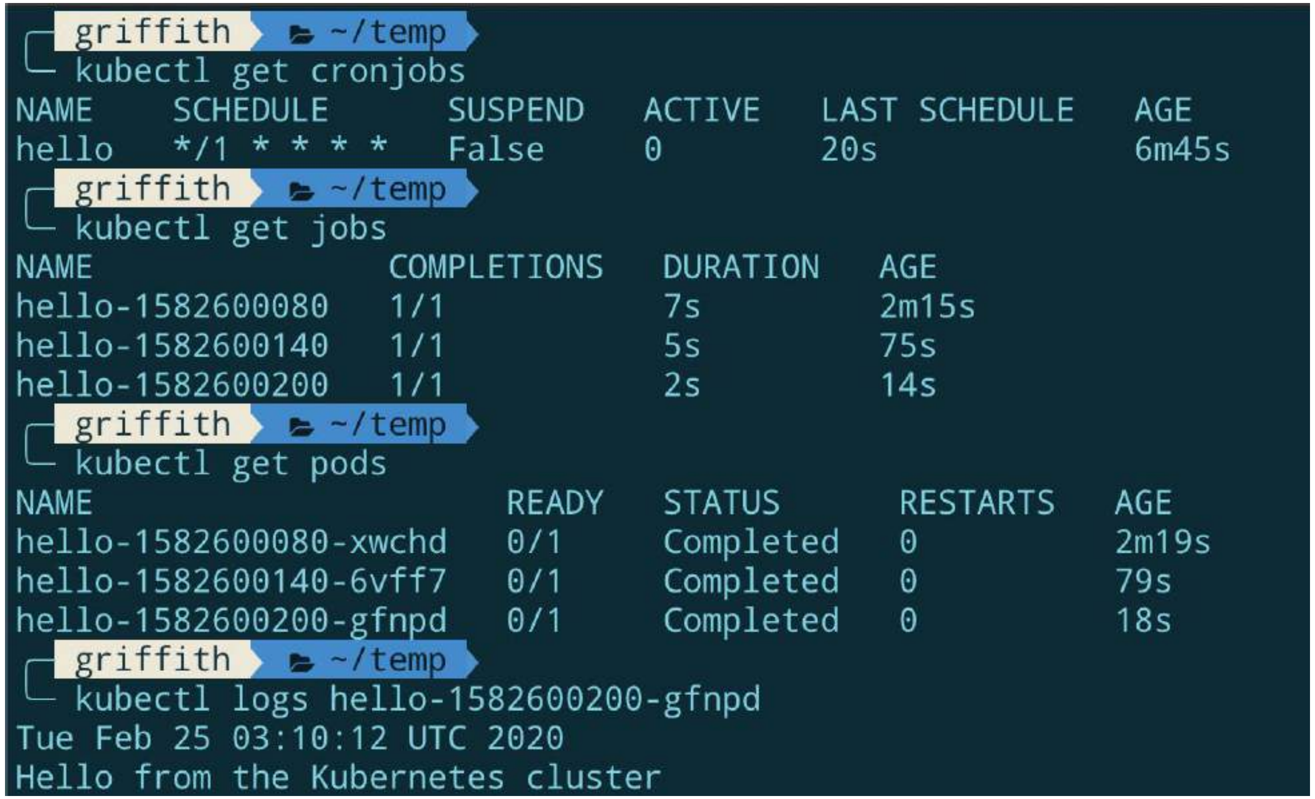
Typically used for monitoring agents or system processes you want running on every node



```
┌ griffith  ⌂ ~
└ kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-nodepool1-30239456-vmss000000   Ready    agent   12d   v1.14.8
aks-nodepool1-30239456-vmss000001   Ready    agent   12d   v1.14.8
aks-nodepool1-30239456-vmss000002   Ready    agent   12d   v1.14.8
┌ griffith  ⌂ ~
└ kubectl get daemonsets --all-namespaces
NAMESPACE     NAME           DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR                 AGE
falco         sysdig-falco   3         3         3       3            3           <none>                        32h
kube-system   kube-proxy     3         3         3       3            3           beta.kubernetes.io/os=linux   12d
kube-system   omsagent       3         3         3       3            3           beta.kubernetes.io/os=linux   5d6h
┌ griffith  ⌂ ~
└ kubectl get pods -l dsName=omsagent-ds -n kube-system -o wide
NAME             READY   STATUS    RESTARTS   AGE    IP            NODE                                NOMINATED NODE   READINESS GATES
omsagent-cjg5m   1/1     Running   0          5d6h   10.244.1.15   aks-nodepool1-30239456-vmss000001   <none>           <none>
omsagent-hmqdr   1/1     Running   0          5d6h   10.244.2.26   aks-nodepool1-30239456-vmss000002   <none>           <none>
omsagent-kc5sn   1/1     Running   0          5d6h   10.244.0.26   aks-nodepool1-30239456-vmss000000   <none>           <none>
```

Kubernetes Resources: Jobs/CronJobs

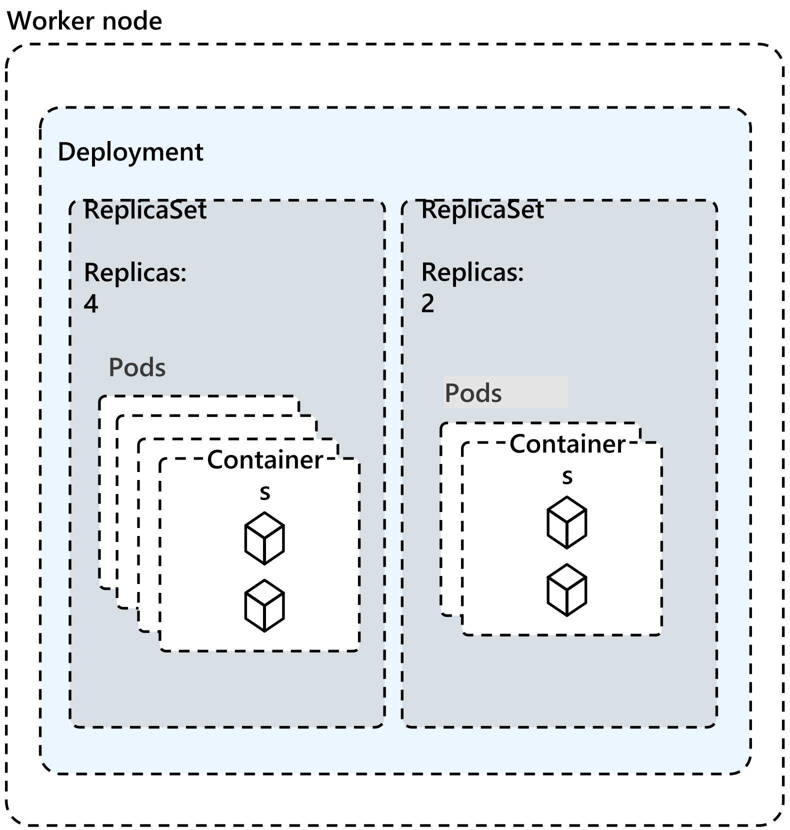
A task that runs to completion, possibly on a schedule via Cron



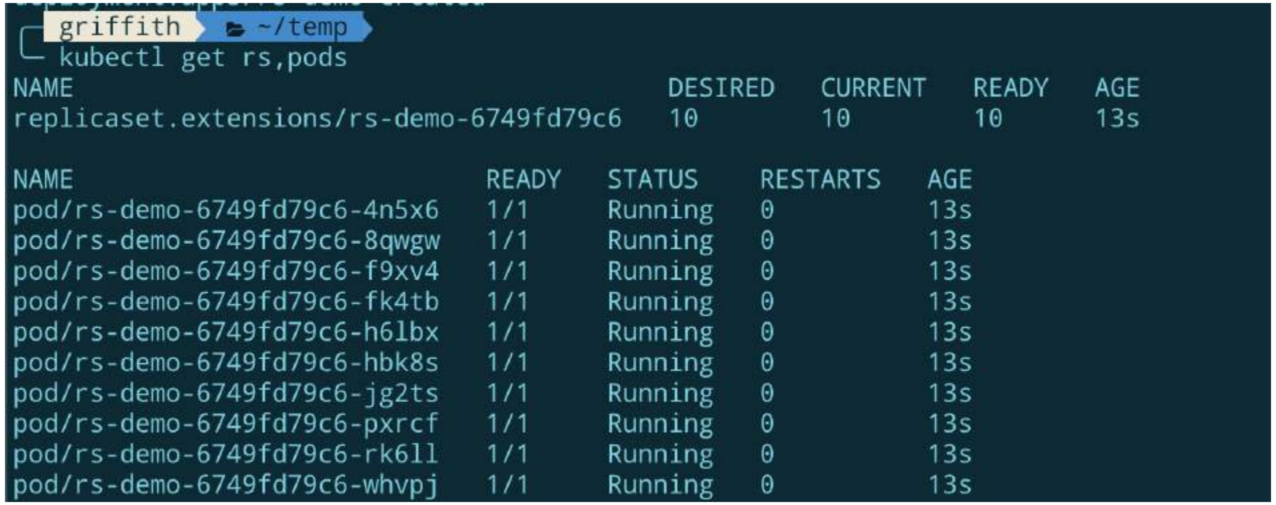
```
┌ griffith  ~/temp
└ kubectl get cronjobs
NAME    SCHEDULE      SUSPEND   ACTIVE   LAST SCHEDULE   AGE
hello   */1 * * * *   False     0        20s             6m45s
┌ griffith  ~/temp
└ kubectl get jobs
NAME                  COMPLETIONS   DURATION   AGE
hello-1582600080      1/1           7s         2m15s
hello-1582600140      1/1           5s         75s
hello-1582600200      1/1           2s         14s
┌ griffith  ~/temp
└ kubectl get pods
NAME                        READY   STATUS      RESTARTS   AGE
hello-1582600080-xwchd      0/1     Completed   0          2m19s
hello-1582600140-6vff7      0/1     Completed   0          79s
hello-1582600200-gfnpd      0/1     Completed   0          18s
┌ griffith  ~/temp
└ kubectl logs hello-1582600200-gfnpd
Tue Feb 25 03:10:12 UTC 2020
Hello from the Kubernetes cluster
```

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            args:
            - /bin/sh
            - -c
            - date; echo Hello from the Kubernetes cluster
          restartPolicy: OnFailure
```

Kubernetes Resources: Replica Sets



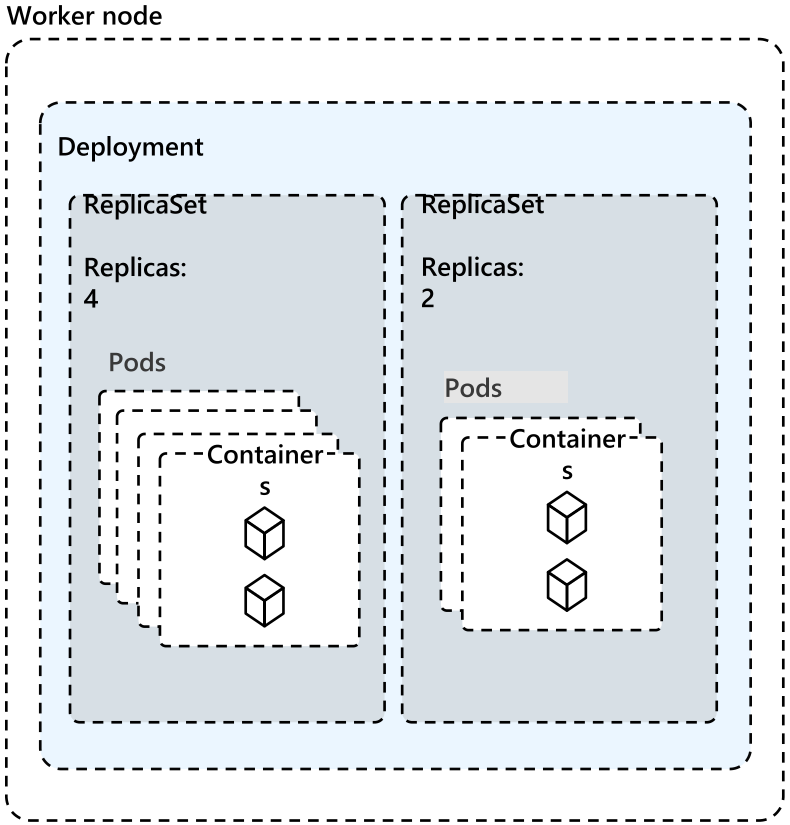
```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    run: rs-demo
  name: rs-demo
spec:
  replicas: 10
  selector:
    matchLabels:
      run: rs-demo
  strategy: {}
  template:
    metadata:
      labels:
        run: rs-demo
    spec:
      containers:
      - image: nginx
        name: rs-demo
        resources: {}
```



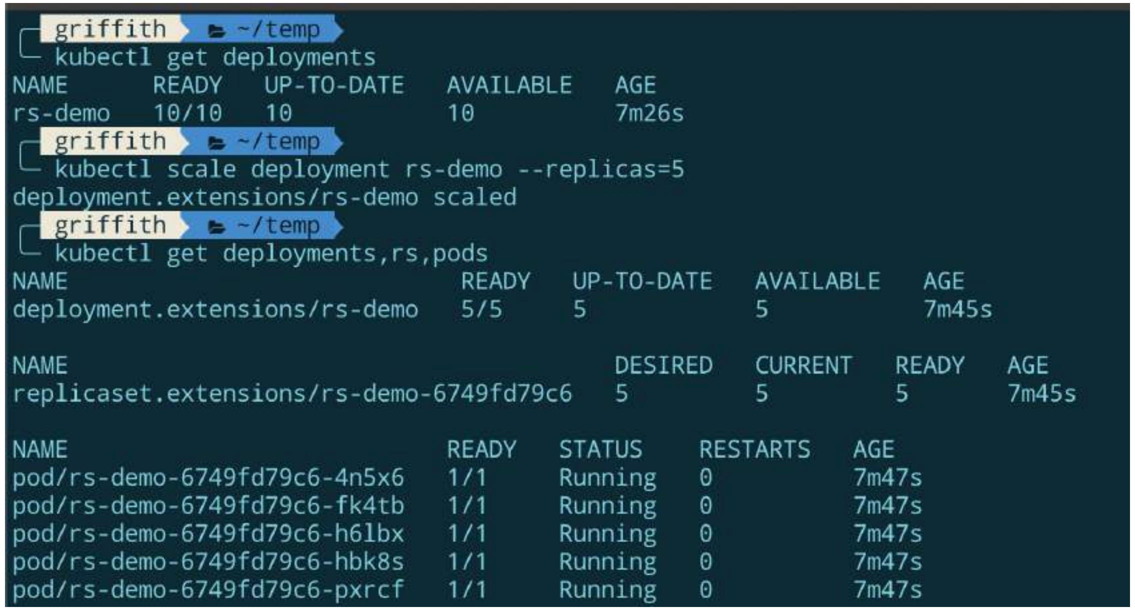
```
┌ griffith   ~/temp
└ kubectl get rs,pods
NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.extensions/rs-demo-6749fd79c6   10        10        10      13s

NAME                             READY   STATUS    RESTARTS   AGE
pod/rs-demo-6749fd79c6-4n5x6     1/1     Running   0          13s
pod/rs-demo-6749fd79c6-8qwgw     1/1     Running   0          13s
pod/rs-demo-6749fd79c6-f9xv4     1/1     Running   0          13s
pod/rs-demo-6749fd79c6-fk4tb     1/1     Running   0          13s
pod/rs-demo-6749fd79c6-h6lbx     1/1     Running   0          13s
pod/rs-demo-6749fd79c6-hbk8s     1/1     Running   0          13s
pod/rs-demo-6749fd79c6-jg2ts     1/1     Running   0          13s
pod/rs-demo-6749fd79c6-pxrcf     1/1     Running   0          13s
pod/rs-demo-6749fd79c6-rk6ll     1/1     Running   0          13s
pod/rs-demo-6749fd79c6-whvpj     1/1     Running   0          13s
```

Kubernetes Resources: Deployments



```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    run: rs-demo
  name: rs-demo
spec:
  replicas: 10
  selector:
    matchLabels:
      run: rs-demo
  strategy: {}
  template:
    metadata:
      labels:
        run: rs-demo
    spec:
      containers:
      - image: nginx
        name: rs-demo
        resources: {}
```

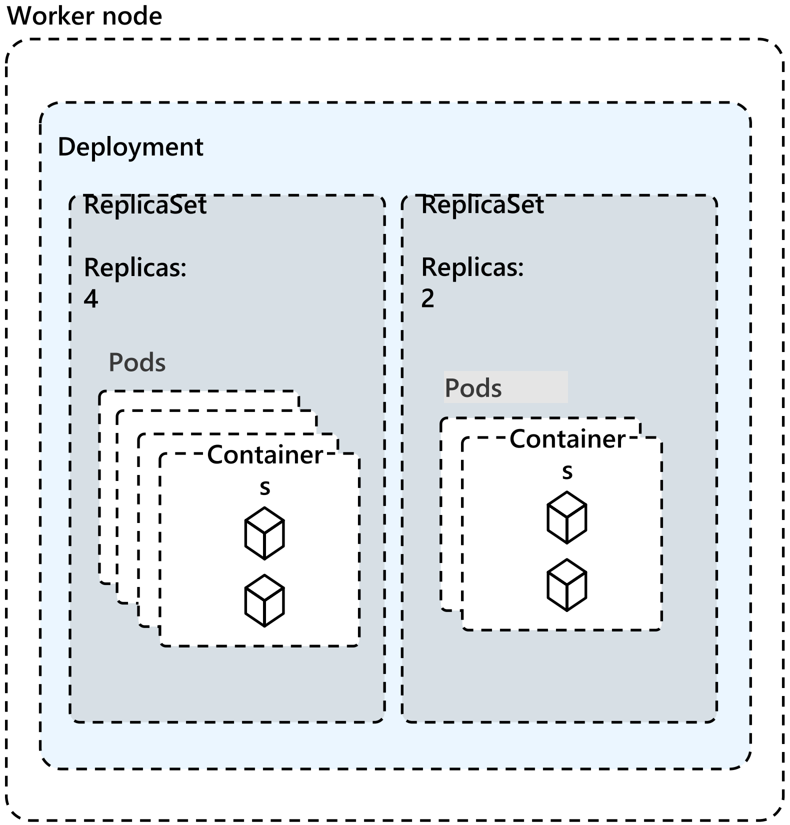


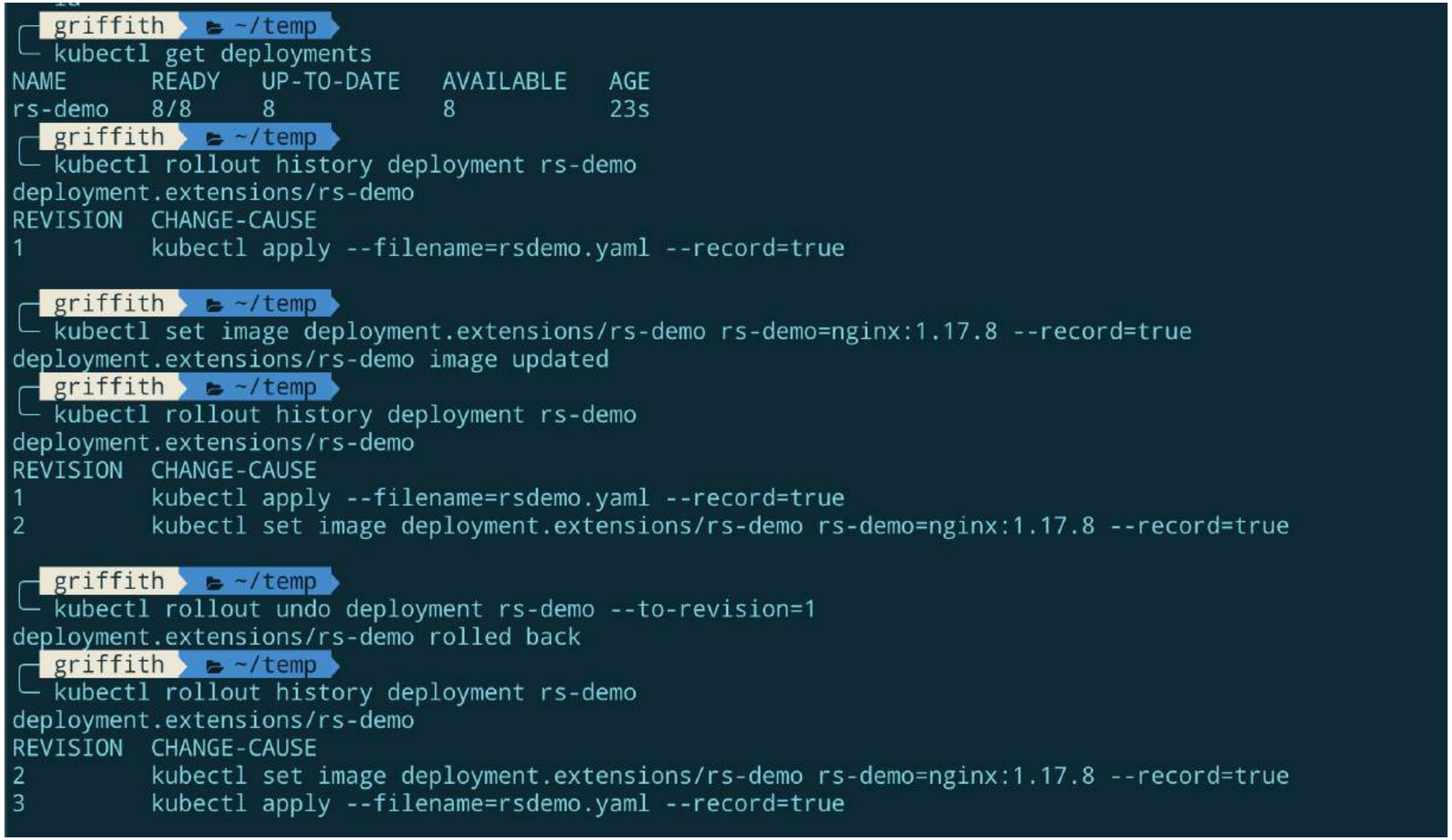
```
┌ griffith  ~/temp
└ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
rs-demo   10/10   10           10          7m26s
┌ griffith  ~/temp
└ kubectl scale deployment rs-demo --replicas=5
deployment.extensions/rs-demo scaled
┌ griffith  ~/temp
└ kubectl get deployments,rs,pods
NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.extensions/rs-demo   5/5     5            5           7m45s

NAME                                        DESIRED   CURRENT   READY   AGE
replicaset.extensions/rs-demo-6749fd79c6   5         5         5       7m45s

NAME                           READY   STATUS    RESTARTS   AGE
pod/rs-demo-6749fd79c6-4n5x6   1/1     Running   0          7m47s
pod/rs-demo-6749fd79c6-fk4tb   1/1     Running   0          7m47s
pod/rs-demo-6749fd79c6-h6lbx   1/1     Running   0          7m47s
pod/rs-demo-6749fd79c6-hbk8s   1/1     Running   0          7m47s
pod/rs-demo-6749fd79c6-pxrcf   1/1     Running   0          7m47s
```

Kubernetes Resources: Deployment History and Rollback



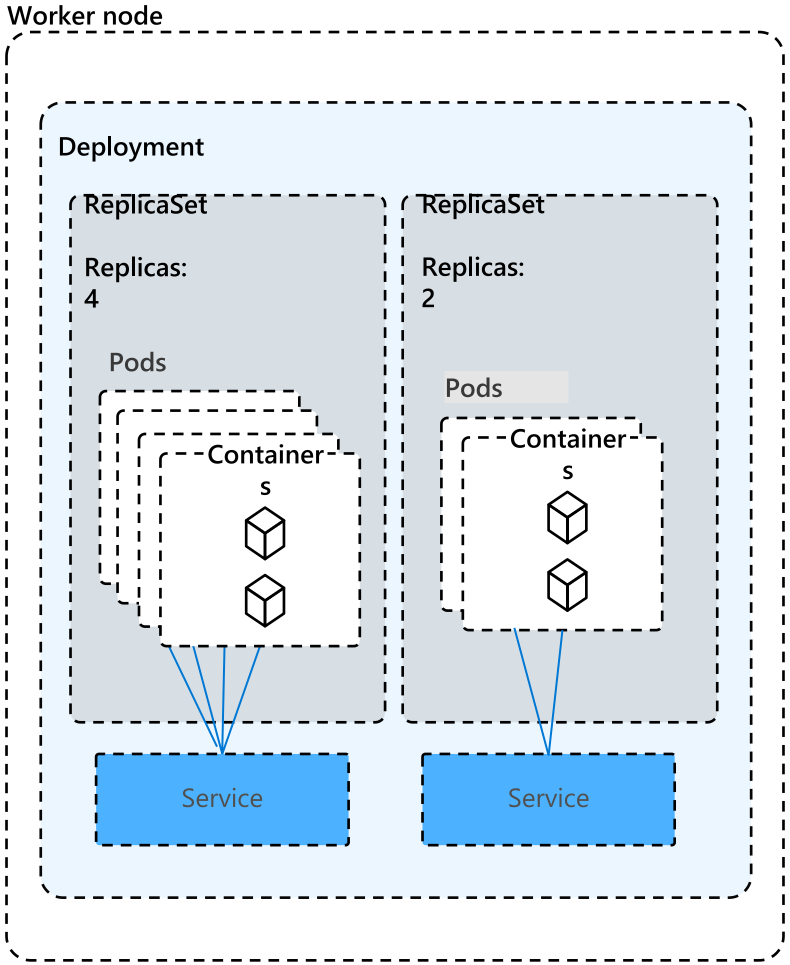


```
┌ griffith  ~/temp
└ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
rs-demo   8/8     8            8           23s
┌ griffith  ~/temp
└ kubectl rollout history deployment rs-demo
deployment.extensions/rs-demo
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=rsdemo.yaml --record=true

┌ griffith  ~/temp
└ kubectl set image deployment.extensions/rs-demo rs-demo=nginx:1.17.8 --record=true
deployment.extensions/rs-demo image updated
┌ griffith  ~/temp
└ kubectl rollout history deployment rs-demo
deployment.extensions/rs-demo
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=rsdemo.yaml --record=true
2         kubectl set image deployment.extensions/rs-demo rs-demo=nginx:1.17.8 --record=true

┌ griffith  ~/temp
└ kubectl rollout undo deployment rs-demo --to-revision=1
deployment.extensions/rs-demo rolled back
┌ griffith  ~/temp
└ kubectl rollout history deployment rs-demo
deployment.extensions/rs-demo
REVISION  CHANGE-CAUSE
2         kubectl set image deployment.extensions/rs-demo rs-demo=nginx:1.17.8 --record=true
3         kubectl apply --filename=rsdemo.yaml --record=true
```

Kubernetes Resources: Service

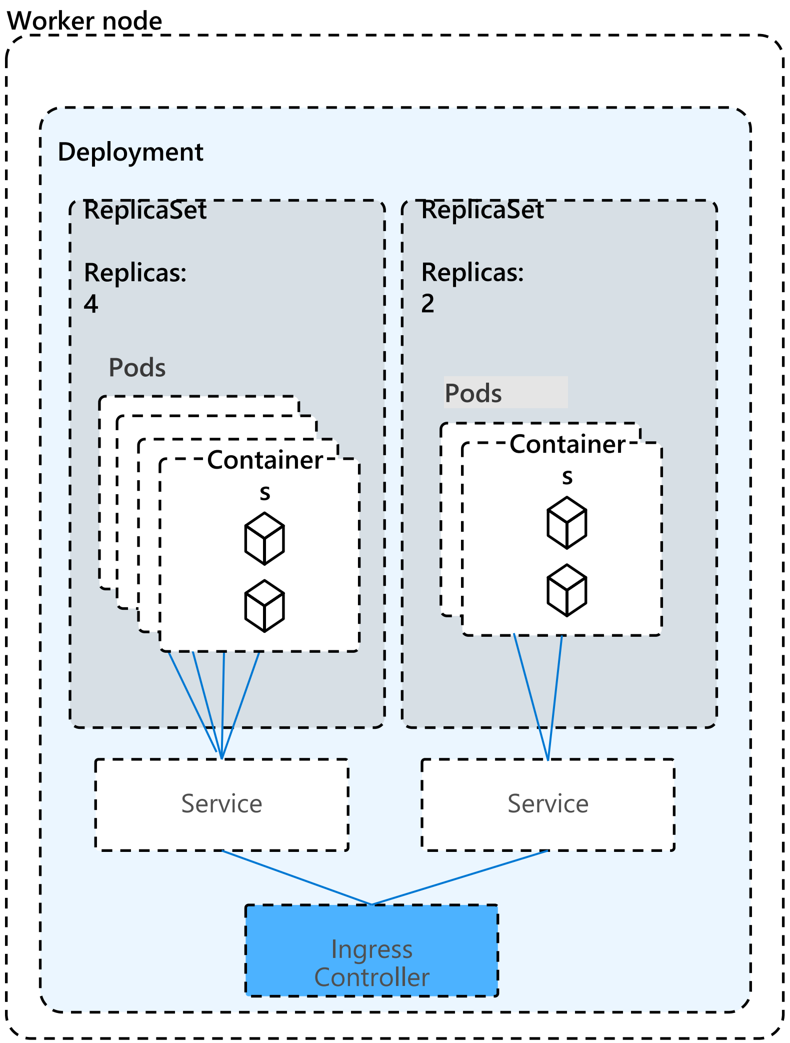


Provides Layer 4 Load Balancing

Types:

- **ClusterIP**: Service is provided an IP internal to the cluster
- **NodePort**: Allocates a port used to access the service across all nodes
- **LoadBalancer**: Exposes the service via a cloud provider loadbalancer (ex. Azure internal or external LB)
- **ExternalName**: Expose the service via cluster DNS mapping

Kubernetes Resources: Ingress



Provides Layer 7 Load Balancing

Kubernetes provides the basic API spec but third parties provide implementations, often adding features via Custom Resource Definitions

Common Ingress Controller Implementations

- Nginx
- Traefik
- Gloo
- Kong
- Azure App Gateway
- etc

Common Features:

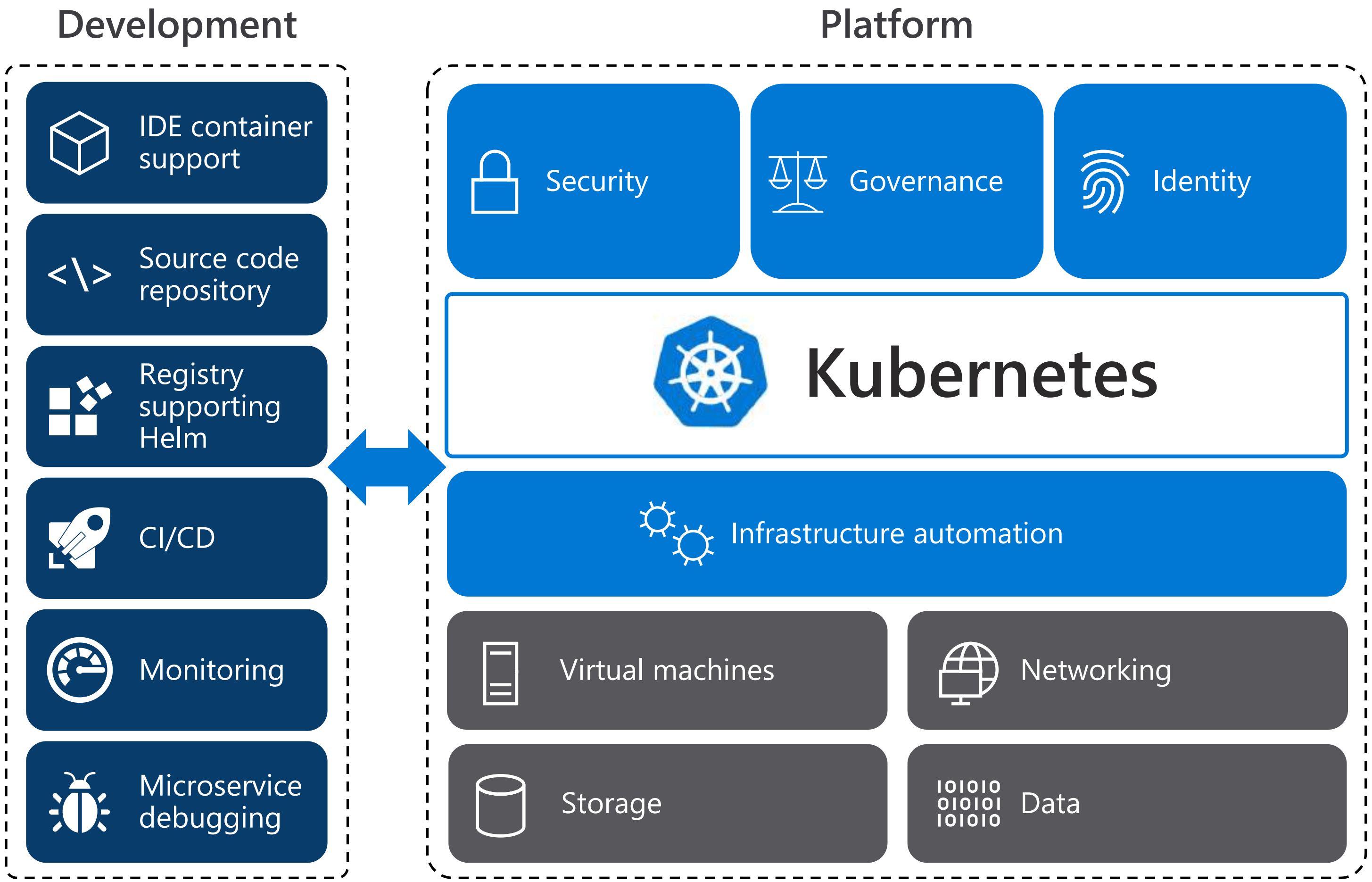
- SSL Offload
- Routing (including Canary & A/B)
- WAF

Kubernetes on its own is **not enough**

Save time from infrastructure management and roll out updates faster without compromising security

Unlock the agility for containerized applications using:

- **Infrastructure automation** that simplifies provisioning, patching, and upgrading
- Tools for **containerized app development and CI/CD workflows**
- Services that support **security, governance, and identity and access management**



Azure Container Registry – vulnerability scanning

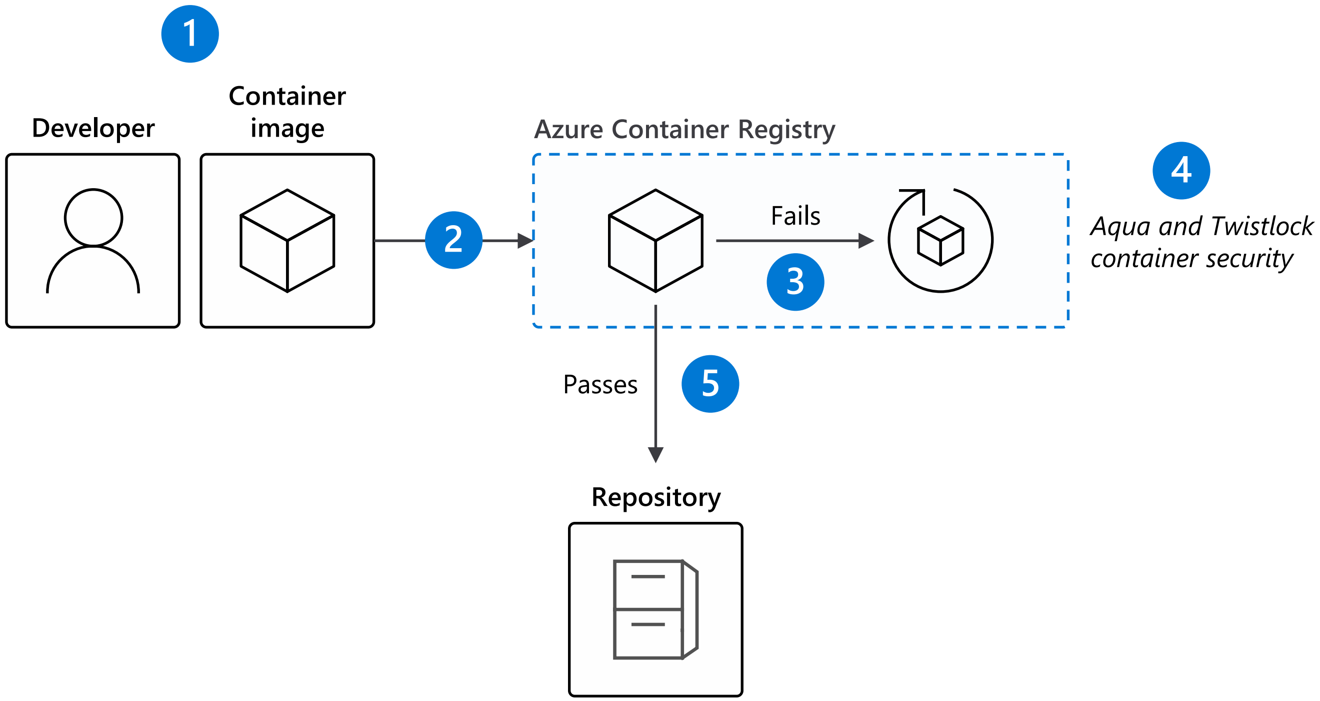
1. Developer/CI system builds container image

2. Image pushed to Azure Container Registry

3. Azure Container Registry quarantines image until scanning passes

4. Azure Container Registry scans content leveraging Aqua, Twistlock

5. Azure Container Registry publishes the image to the repository



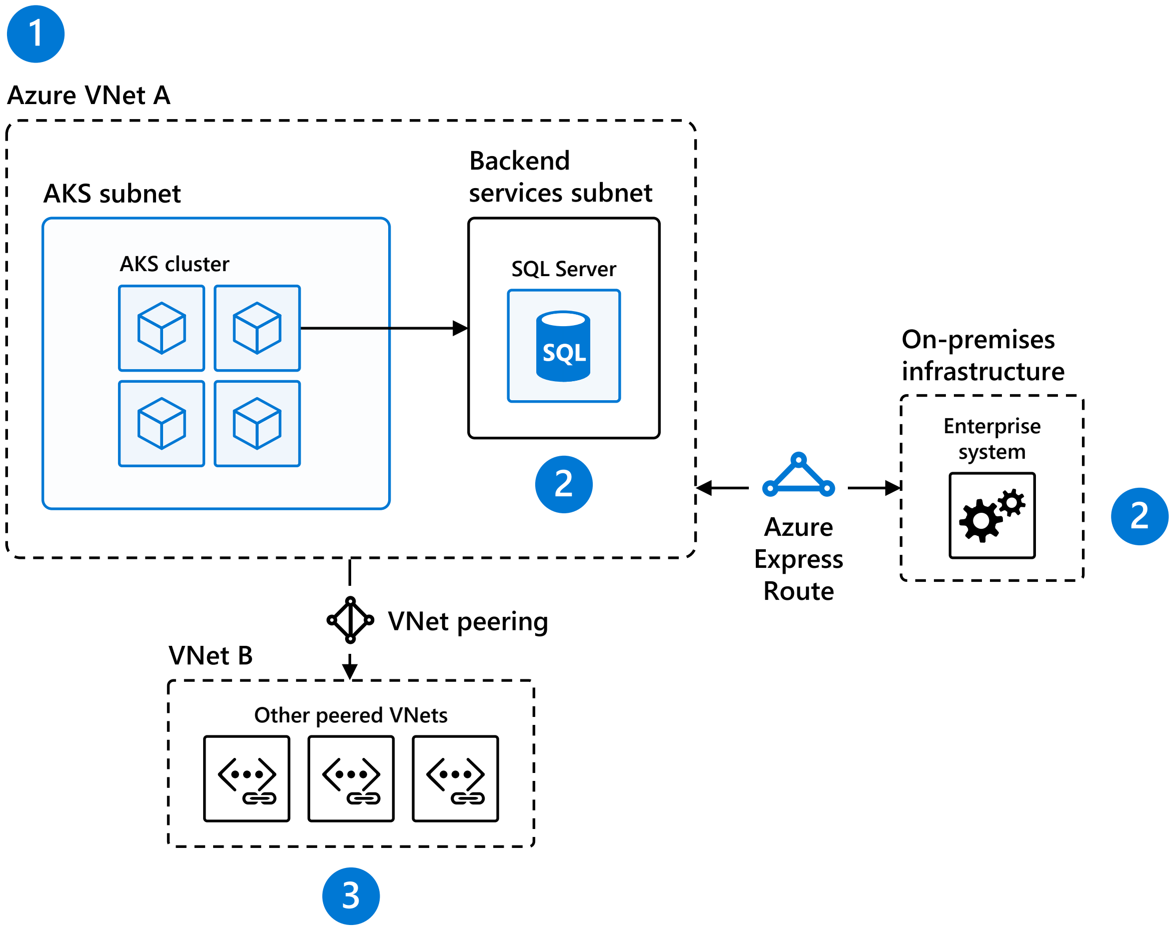
Secure network communications with VNET and CNI

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 VNET peering to other VNET

AKS VNet integration works seamlessly with your existing network infrastructure



Identity and access management through AAD and RBAC

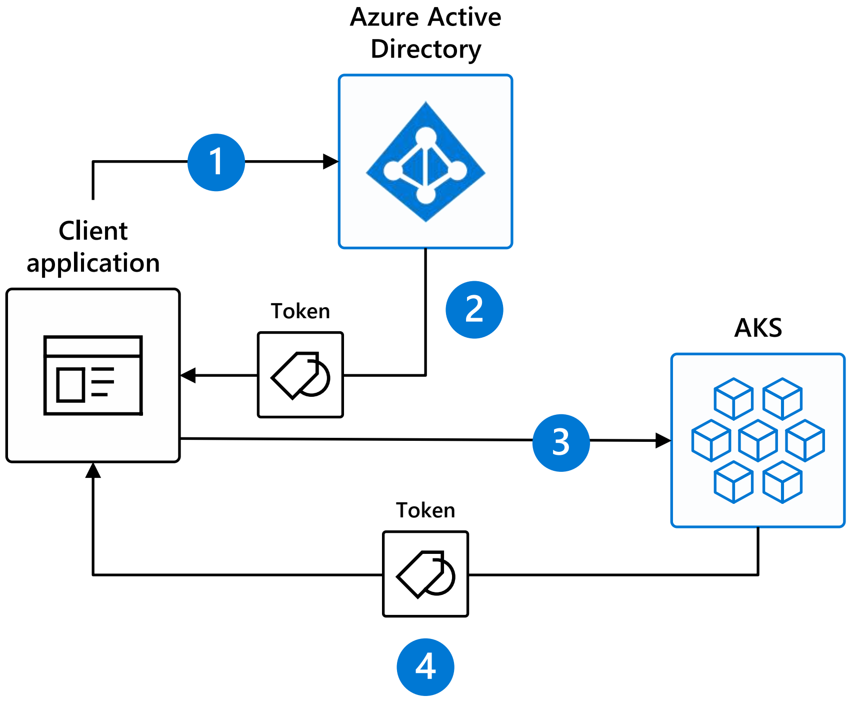
1. The client application authenticates to the AAD token issuance endpoint and requests an access token

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)



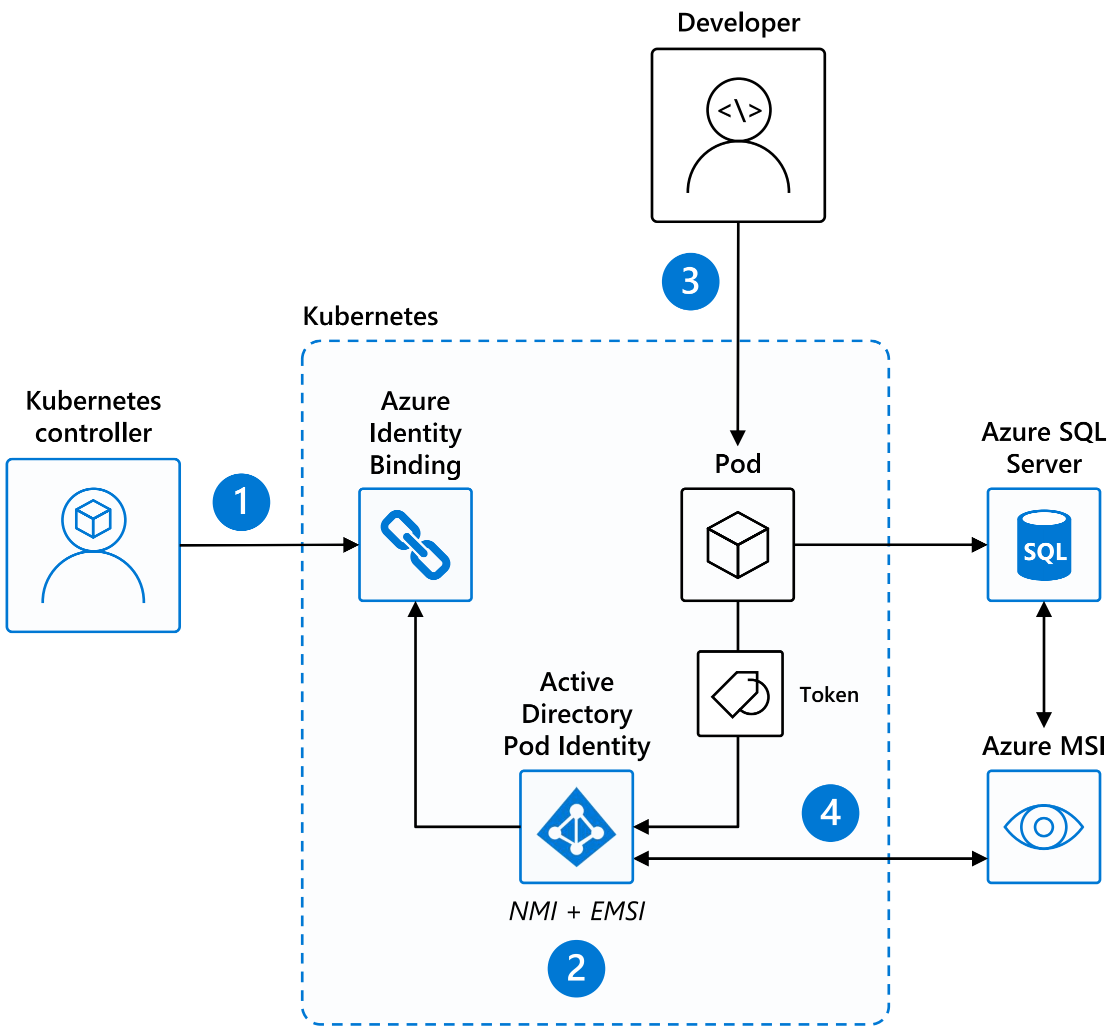
Pod identity

1. Kubernetes operator defines an identity map for K8s service accounts

2. Node Managed Identity (NMI) watches for mapping reaction and syncs to Managed Service Identify (MSI)

3. Developer creates a pod with a service account, and pod uses standard Azure SDK to fetch a token bound to MSI

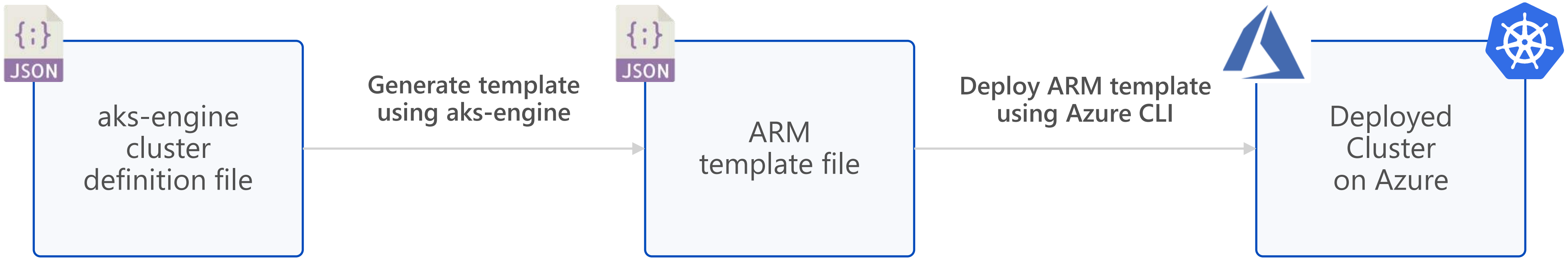
4. Pod uses access token to consume other Azure services; services validate token



AKS Engine

AKS engine takes the best practices developed for AKS and provides it as an OSS project for deploying unmanaged clusters.

- Build fully customized Kubernetes clusters
- Easily deploy Kubernetes on Azure Stack



JSON
aks-engine cluster definition file
Generate template using aks-engine
JSON
ARM template file
Deploy ARM template using Azure CLI
Deployed Cluster on Azure

ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker API (OSBA)

Release Automation Tools

Azure Container Instances (ACI)

Easily run containers on Azure without managing servers

Run containers without managing servers

Increase agility with containers on demand

Secure applications with hypervisor isolation

What can you build with ACI

Elastic bursting with AKS

Azure Container Service (AKS) can use the Virtual Kubelet to provision pods inside ACI that start in seconds. Then ACI provides fast, isolated compute to meet traffic that comes in spikes, without the need to manage servers.

Event-driven applications with Azure Logic Apps

Combine ACI with the ACI Logic Apps connector, Azure queues, and Azure Functions to build robust infrastructure that can elastically scale out containers on demand.

Data processing jobs

Use Azure Container Instances for data processing where source data is ingested, processed, and placed in a durable store such as Azure Blob storage. Achieve significant cost savings through per-second billing.

ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker API (OSBA)

Release Automation Tools

ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker API (OSBA)

Release Automation Tools

Azure Container Instances (ACI)

Get started easily

```
$ az container create --name mycontainer --image microsoft/aci-helloworld --resource-group myResourceGroup --ip-address public
  "ipAddress": {
    "ip": "52.168.86.133",
    "ports": [...]
  },
  "location": "eastus",
  "name": "mycontainer",
  "osType": "Linux",
  "provisioningState": "Succeeded",

$ curl 52.168.86.133
<html>
<head>
  <title>Welcome to Azure Container Instances!</title>
</head>
```

ACS Engine

Azure Container Instances (ACI)

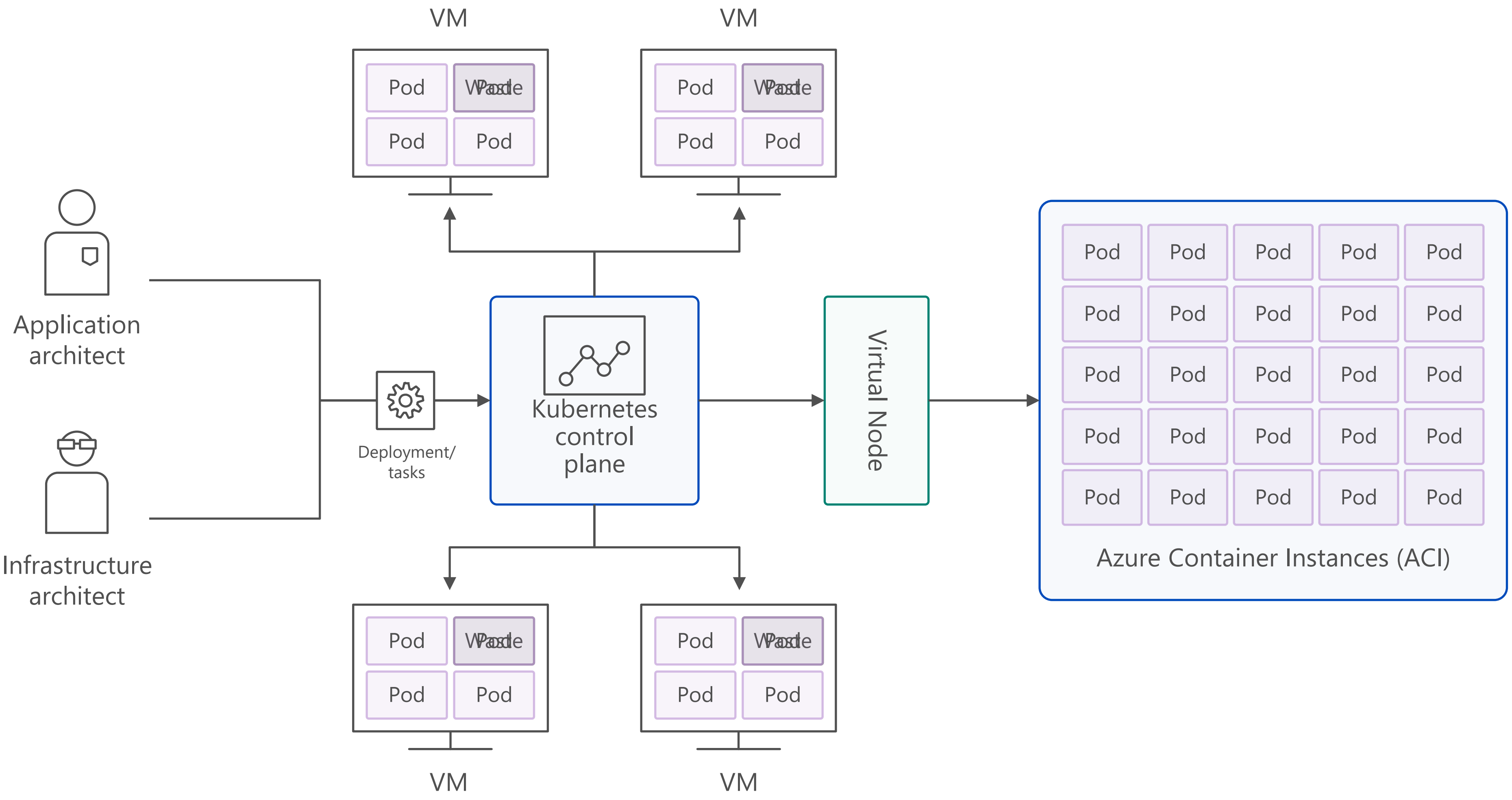
Azure Container Registry (ACR)

Open Service Broker API (OSBA)

Release Automation Tools

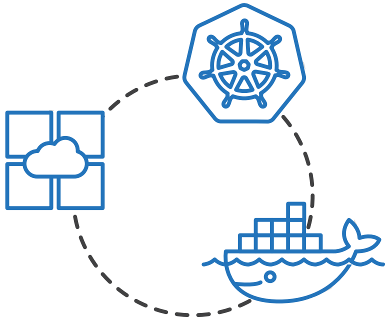
Azure Container Instances (ACI)

Bursting with Azure Virtual Nodes PREVIEW

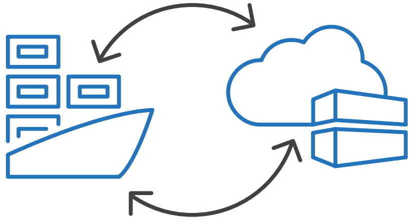


Azure Container Registry (ACR)

Manage a Docker private registry as a first-class Azure resource



Manage images for all types of containers



Use familiar, open-source Docker CLI tools

Azure Container Registry geo-replication

ACS Engine

Azure Container Instances (ACI)

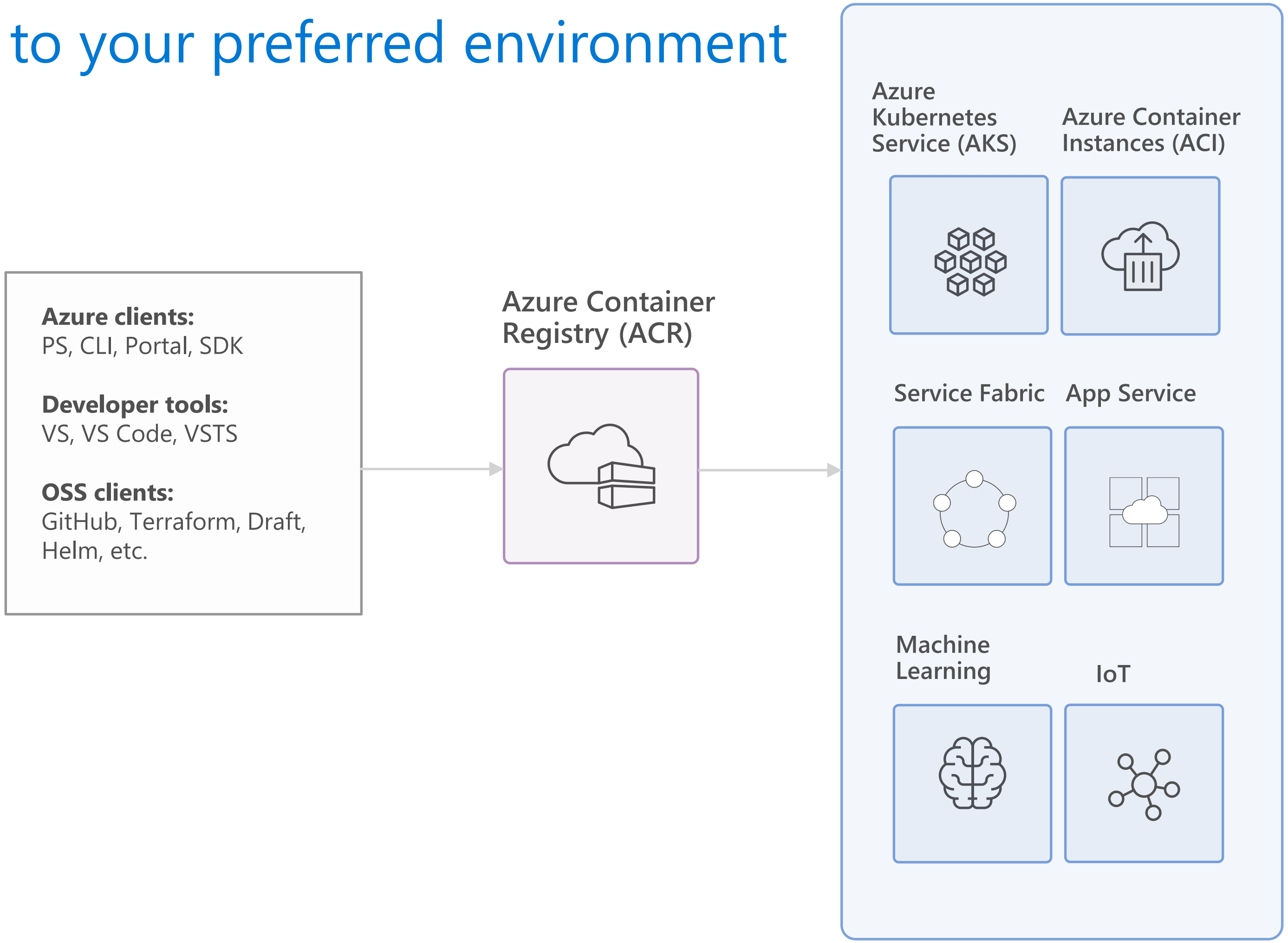
Azure Container Registry (ACR)

Open Service Broker API (OSBA)

Release Automation Tools

Containers as the App Packaging Format

Deploy to your preferred environment



Azure clients:
PS, CLI, Portal, SDK

Developer tools:
VS, VS Code, VSTS

OSS clients:
GitHub, Terraform, Draft, Helm, etc.

Azure Container Registry (ACR)
Azure Kubernetes Service (AKS)
Azure Container Instances (ACI)
Service Fabric
App Service
Machine Learning
IoT

ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

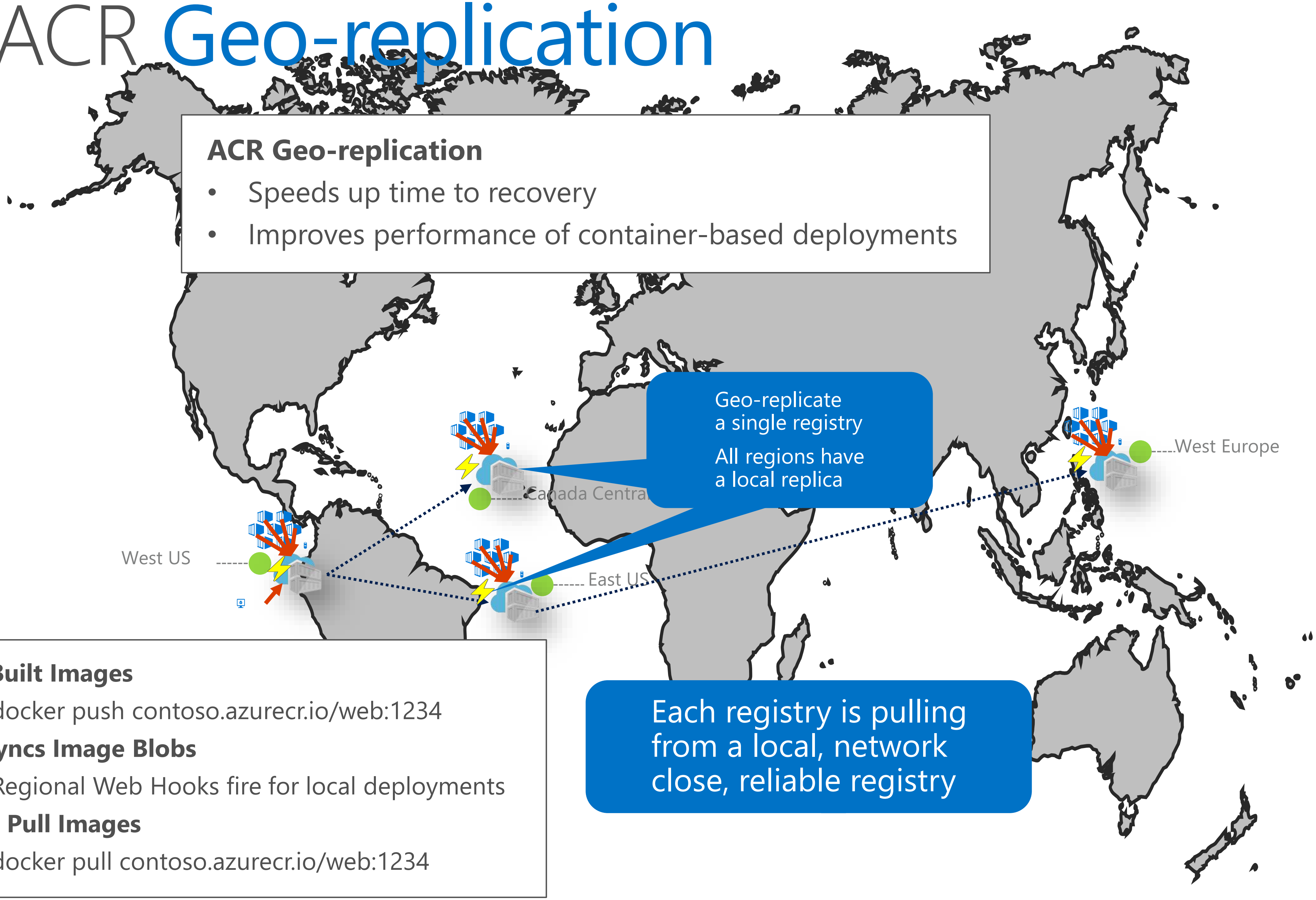
Open Service Broker for Azure (OSBA)

Release Automation Tools

ACR Geo-replication

ACR Geo-replication

- Speeds up time to recovery
- Improves performance of container-based deployments



Geo-replicate
a single registry

All regions have
a local replica

Each registry is pulling from a local, network close, reliable registry

Push Built Images

```
docker push contoso.azurecr.io/web:1234
```

ACR Syncs Image Blobs

Regional Web Hooks fire for local deployments

Nodes Pull Images

```
docker pull contoso.azurecr.io/web:1234
```

ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

Built-in security with ACR

Authenticate using Azure Active Directory Identity

Integrated OS & Framework Patching

Secure images by default with quarantine pattern

ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

ACR Tasks

Native Container Build Service in the cloud

Follows docker **build** semantics

```
docker build -t helloworld:v1 .
az acr build -t helloworld{{.Build.ID}} .
```

Trigger based builds (git commits, base image updates)

```
az acr build-task create
 --image           helloworld{{.Build.ID}}
 --name            myBuildTask
 --registry        jengademos
 --context         https://github.com/me/helloworld
 --branch          master
 --git-access-token $PAT
```

ACS Engine

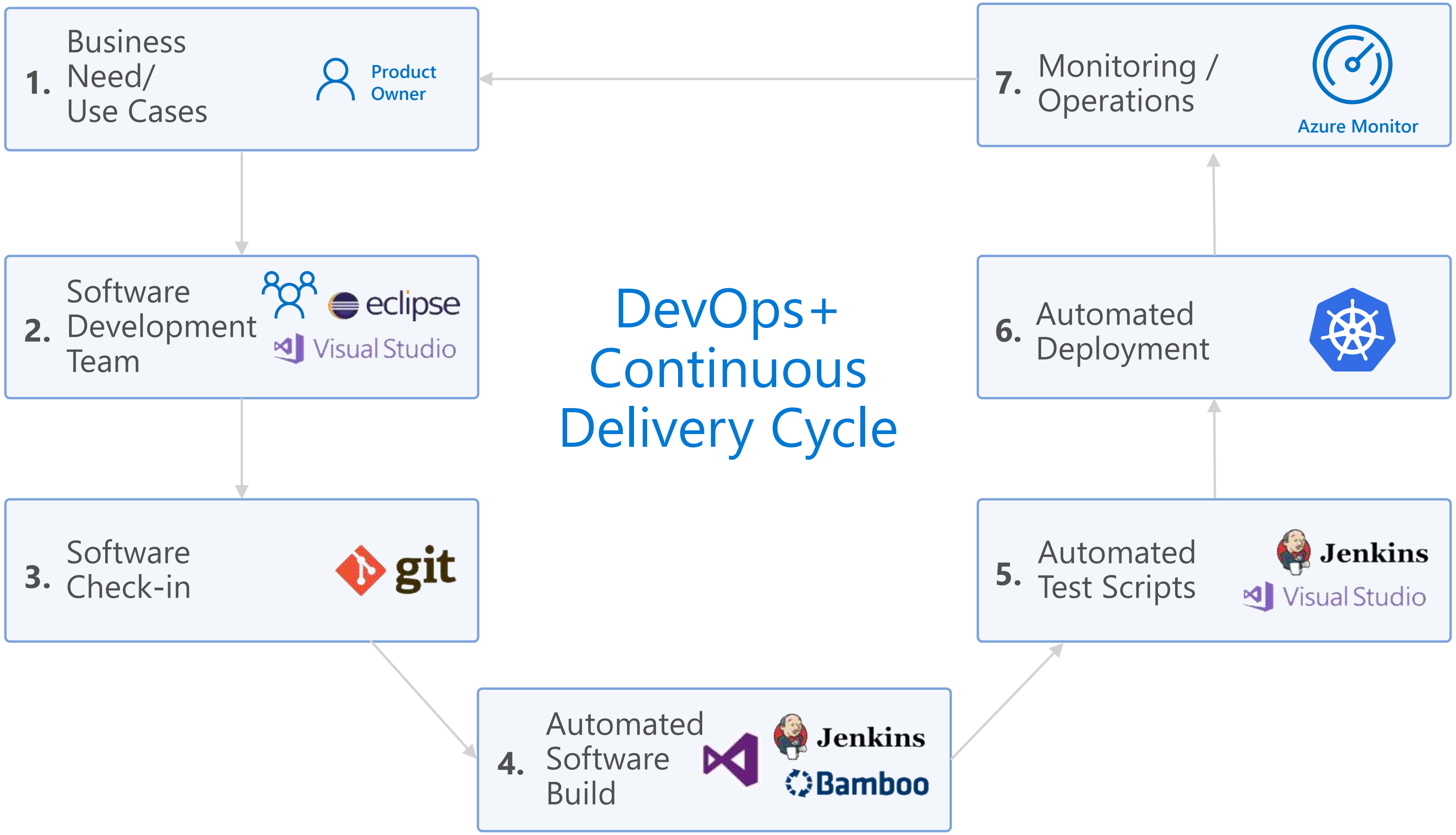
Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

DevOps Practices Arrive



1. Business Need/ Use Cases — Product Owner

2. Software Development Team — eclipse, Visual Studio

3. Software Check-in — git

4. Automated Software Build — Jenkins, Bamboo

5. Automated Test Scripts — Jenkins, Visual Studio

6. Automated Deployment

7. Monitoring / Operations — Azure Monitor

DevOps+ Continuous Delivery Cycle

ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

Why DevOps?

The benefits

46x
more frequent deployments

5x
lower change failure rate

440x
faster deployments

440x
shorter lead times

ACS Engine

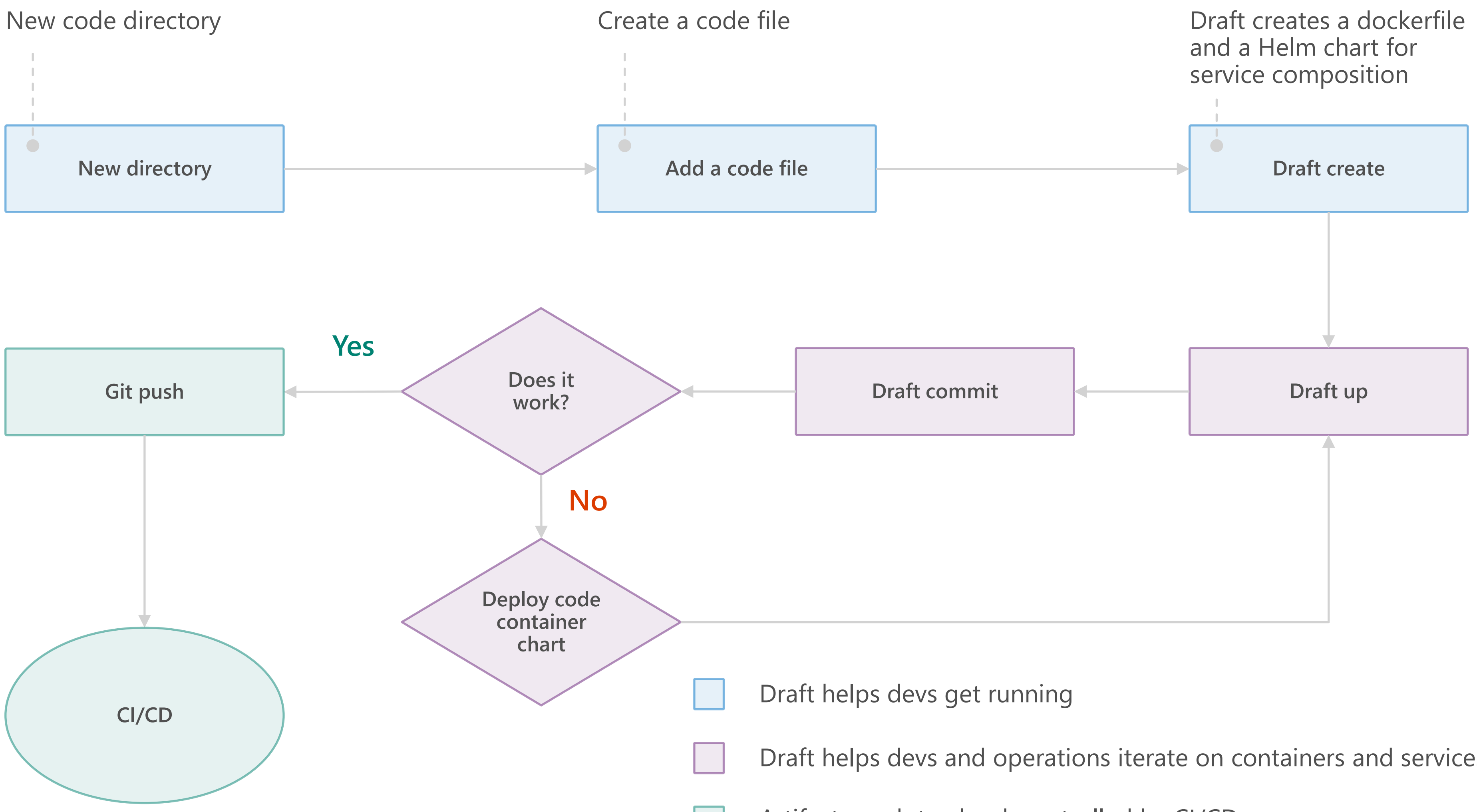
Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

Release automation workflow



New code directory

New directory

Create a code file

Add a code file

Draft creates a dockerfile and a Helm chart for service composition

Draft create

Draft up

Draft commit

Does it work?

Yes

Git push

CI/CD

No

Deploy code container chart

🟦 Draft helps devs get running

🟪 Draft helps devs and operations iterate on containers and services

🟩 Artifacts push to cloud, controlled by CI/CD processes

ACS Engine

Azure Container Instances (ACI)

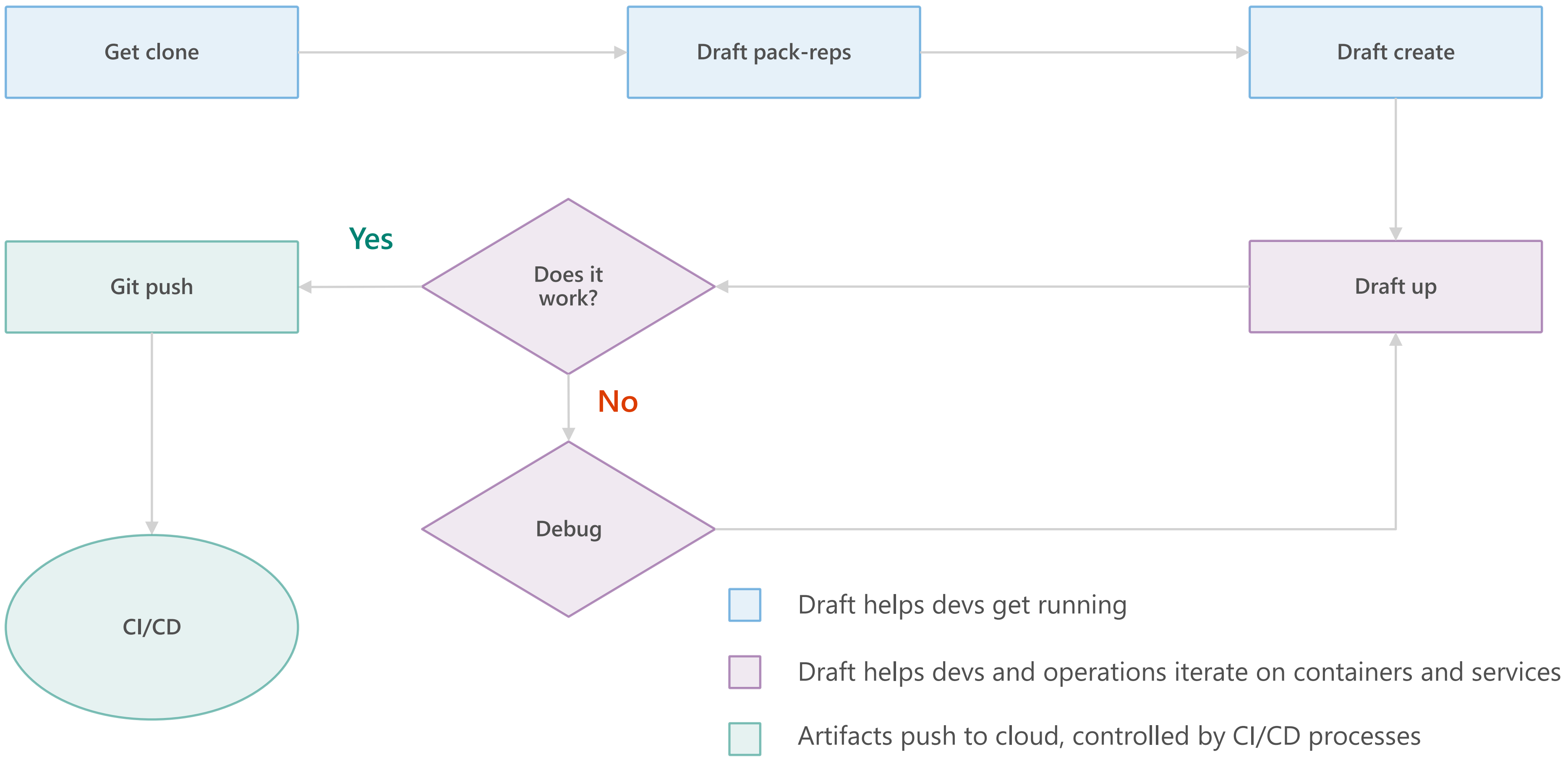
Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

Release automation workflow

Once developers are up and running—or working on a service that is in a complex system—Draft ALSO helps devs ignore artifacts and focus on code



Get clone

Draft pack-reps

Draft create

Draft up

Does it work?

Yes

No

Git push

Debug

CI/CD

🟦 Draft helps devs get running

🟪 Draft helps devs and operations iterate on containers and services

🟩 Artifacts push to cloud, controlled by CI/CD processes

ACS Engine

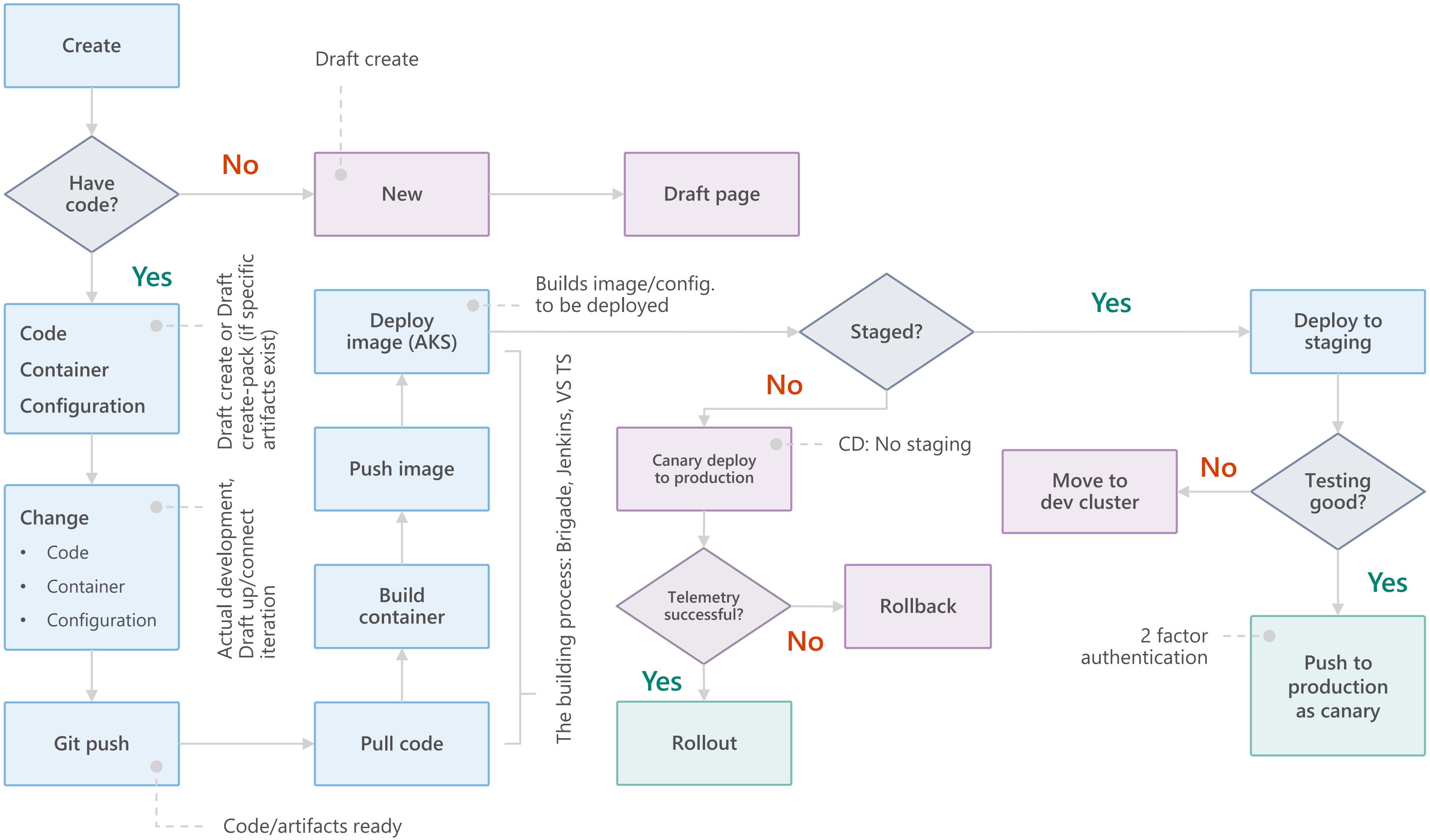
Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

Release automation workflow



Create

Have code?

No

New

Draft create

Draft page

Yes

Code

Container

Configuration

Draft create or Draft create-pack (if specific artifacts exist)

Change

- Code
- Container
- Configuration

Actual development, Draft up/connect iteration

Git push

Code/artifacts ready

Pull code

Build container

Push image

Deploy image (AKS)

Builds image/config. to be deployed

The building process: Brigade, Jenkins, VS TS

Staged?

Yes

Deploy to staging

No

Canary deploy to production

CD: No staging

Telemetry successful?

No

Rollback

Yes

Rollout

Testing good?

No

Move to dev cluster

Yes

Push to production as canary

2 factor authentication

ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

Release automation tools

Simplifying the Kubernetes experience

DRAFT

Streamlined Kubernetes development

HELM

The package manager for Kubernetes

BRIGADE

Event-driven scripting for Kubernetes

ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

Helm

The best way to find, share, and use software built for Kubernetes

Manage complexity

Charts can describe complex apps; provide repeatable app installs, and serve as a single point of authority

Easy updates

Take the pain out of updates with in-place upgrades and custom hooks

Simple sharing

Charts are easy to version, share, and host on public or private servers

Rollbacks

Use `helm rollback` to roll back to an older version of a release with ease

ACS Engine

Azure Container Instances (ACI)

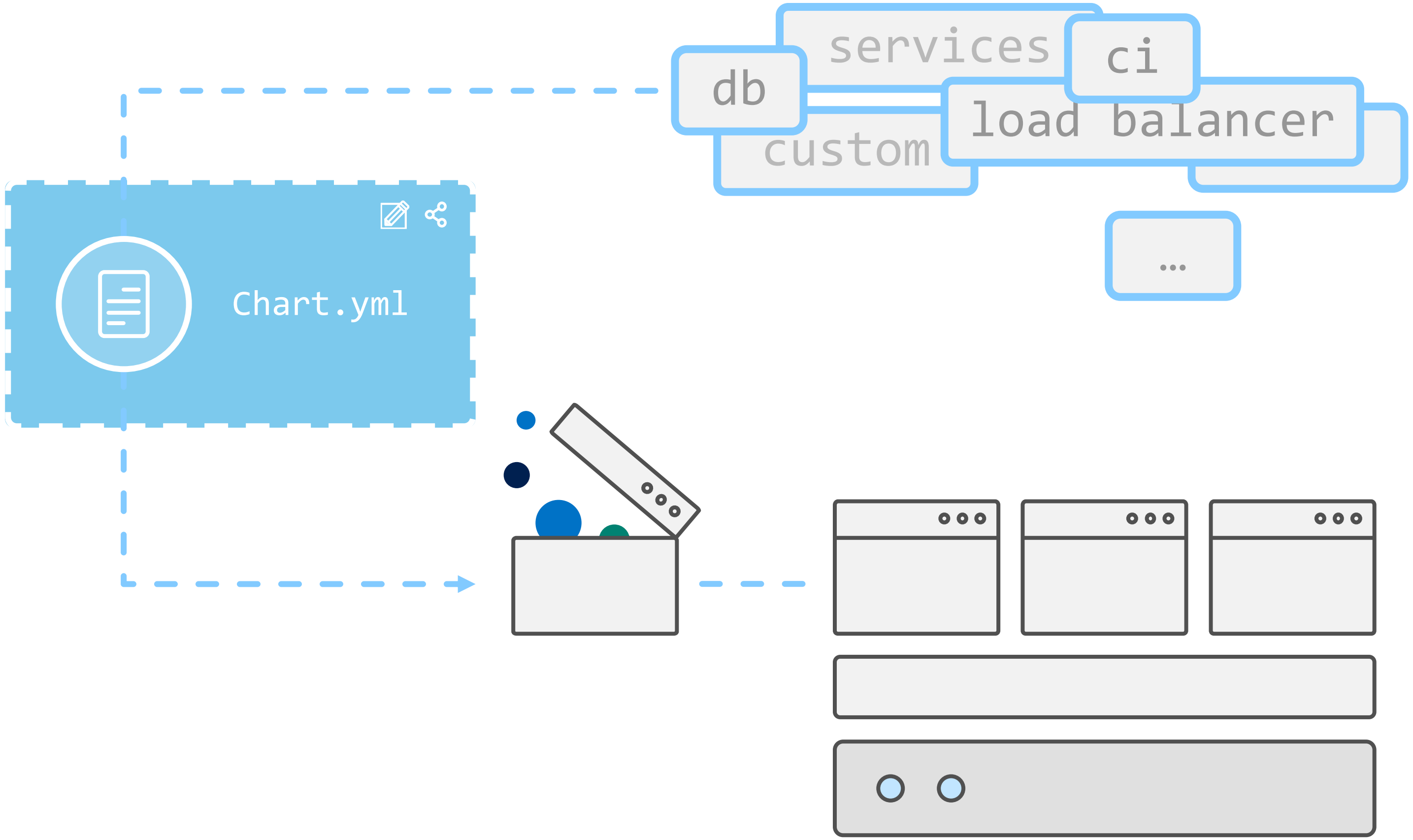
Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

Helm

Helm Charts helps you define, install, and upgrade even the most complex Kubernetes application



ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

Draft

Simple app development and deployment – into any Kubernetes cluster

Simplified development

Using two simple commands, developers can now begin hacking on container-based applications without requiring Docker or even installing Kubernetes themselves

Language support

Draft detects which language your app is written in, and then uses packs to generate a Dockerfile and Helm Chart with the best practices for that language

Draft

Draft in action

```
$ 
```

ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

Brigade

Run scriptable, automated tasks in the cloud — as part of your Kubernetes cluster

Simple, powerful pipes

Each project gets a `brigade.js` config file, which is where you can write dynamic, interwoven pipelines and tasks for your Kubernetes cluster

Runs inside your cluster

By running Brigade as a service inside your Kubernetes cluster, you can harness the power of millions of available Docker images

ACS Engine

Azure Container Instances (ACI)

Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

ACS Engine

Azure Container Instances (ACI)

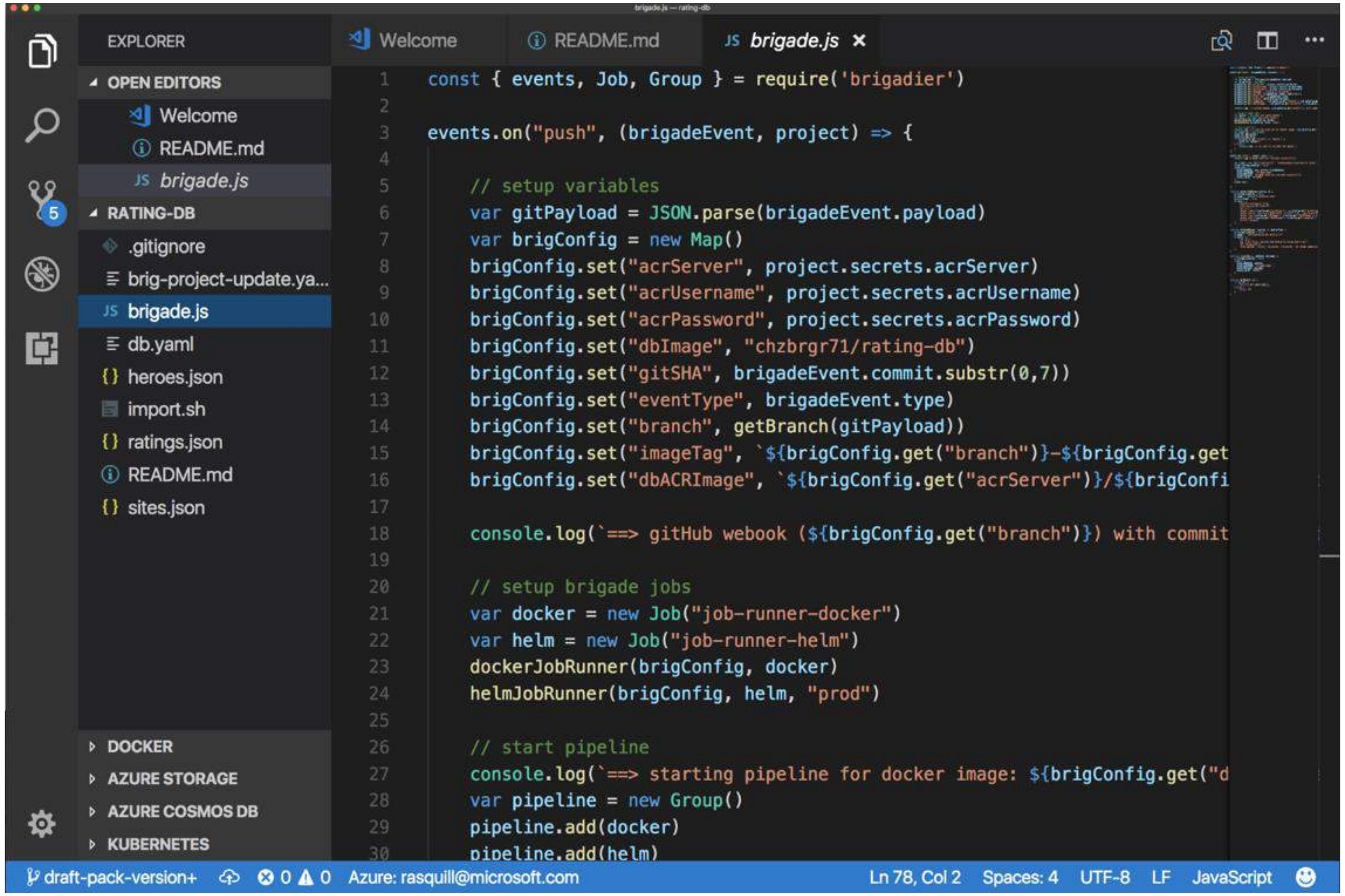
Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

Brigade

Brigade in action



```javascript
const { events, Job, Group } = require('brigadier')

events.on("push", (brigadeEvent, project) => {

    // setup variables
    var gitPayload = JSON.parse(brigadeEvent.payload)
    var brigConfig = new Map()
    brigConfig.set("acrServer", project.secrets.acrServer)
    brigConfig.set("acrUsername", project.secrets.acrUsername)
    brigConfig.set("acrPassword", project.secrets.acrPassword)
    brigConfig.set("dbImage", "chzbrgr71/rating-db")
    brigConfig.set("gitSHA", brigadeEvent.commit.substr(0,7))
    brigConfig.set("eventType", brigadeEvent.type)
    brigConfig.set("branch", getBranch(gitPayload))
    brigConfig.set("imageTag", `${brigConfig.get("branch")}-${brigConfig.get
    brigConfig.set("dbACRImage", `${brigConfig.get("acrServer")}/${brigConfi

    console.log(`==> gitHub webook (${brigConfig.get("branch")}) with commit

    // setup brigade jobs
    var docker = new Job("job-runner-docker")
    var helm = new Job("job-runner-helm")
    dockerJobRunner(brigConfig, docker)
    helmJobRunner(brigConfig, helm, "prod")

    // start pipeline
    console.log(`==> starting pipeline for docker image: ${brigConfig.get("d
    var pipeline = new Group()
    pipeline.add(docker)
    pipeline.add(helm)
```

ACS Engine

Azure Container Instances (ACI)

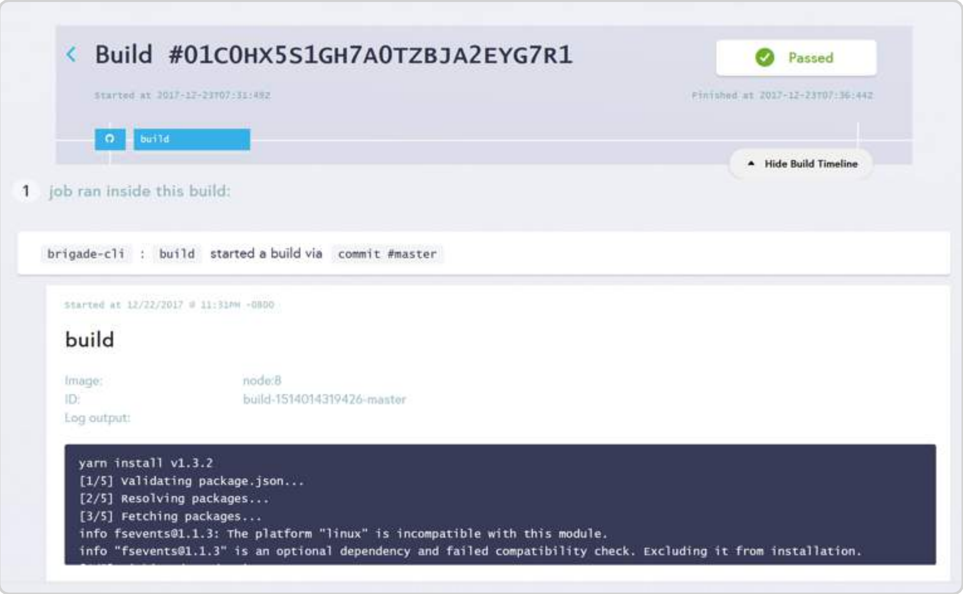
Azure Container Registry (ACR)

Open Service Broker for Azure (OSBA)

Release Automation Tools

Brigade UI

Dashboards for Brigade pipelines



Build #01C0HX5S1GH7A0TZBJA2EYG7R1

✓ Passed

Started at 2017-12-23T07:31:49Z

Finished at 2017-12-23T07:36:44Z

1 job ran inside this build:

`brigade-cli` : `build` started a build via `commit #master`

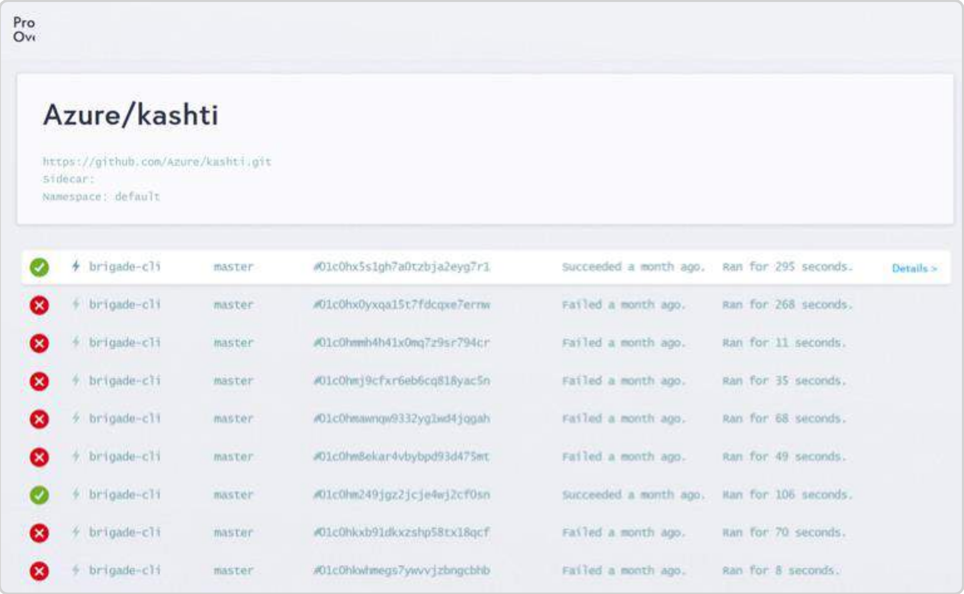
Started at 12/22/2017 @ 11:31PM -0800

build

Image: node:8
ID: build-1514014319426-master
Log output:

```
yarn install v1.3.2
[1/5] Validating package.json...
[2/5] Resolving packages...
[3/5] Fetching packages...
info fsevents@1.1.3: The platform "linux" is incompatible with this module.
info "fsevents@1.1.3" is an optional dependency and failed compatibility check. Excluding it from installation.
```

Builds dashboard



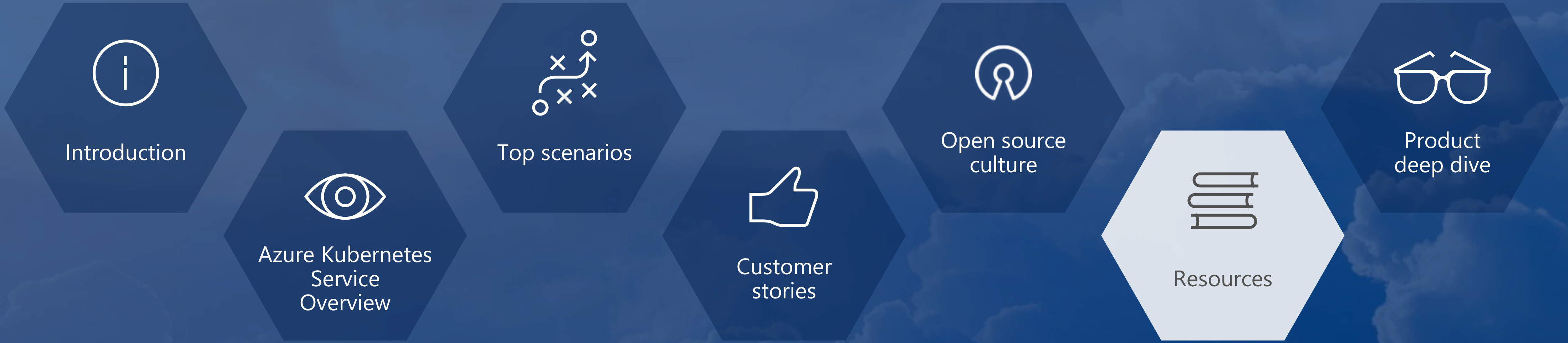
Azure/kashti

https://github.com/Azure/kashti.git
Sidecar:
Namespace: default

✓	brigade-cli	master	#01c0hx5s1gh7a0tzbja2eyg7r1	Succeeded a month ago.	Ran for 295 seconds.	Details >
✗	brigade-cli	master	#01c0hx0yxqa15t7fdcqxe7ernw	Failed a month ago.	Ran for 268 seconds.	
✗	brigade-cli	master	#01c0hmmh4h41x0mq7z9sr794cr	Failed a month ago.	Ran for 11 seconds.	
✗	brigade-cli	master	#01c0hmj9cfxr6eb6cq818yac5n	Failed a month ago.	Ran for 35 seconds.	
✗	brigade-cli	master	#01c0hmawnqw9332yglwd4jqgah	Failed a month ago.	Ran for 68 seconds.	
✗	brigade-cli	master	#01c0hm8ekar4vbybpd93d475mt	Failed a month ago.	Ran for 49 seconds.	
✓	brigade-cli	master	#01c0hm249jgz2jcje4wj2cf0sn	Succeeded a month ago.	Ran for 106 seconds.	
✗	brigade-cli	master	#01c0hkxb91dkxzshp58tx18qcf	Failed a month ago.	Ran for 70 seconds.	
✗	brigade-cli	master	#01c0hkwhmegs7ywvvjzbngcbhb	Failed a month ago.	Ran for 8 seconds.	

Events log

Resources



Introduction

Azure Kubernetes Service Overview

Top scenarios

Customer stories

Open source culture

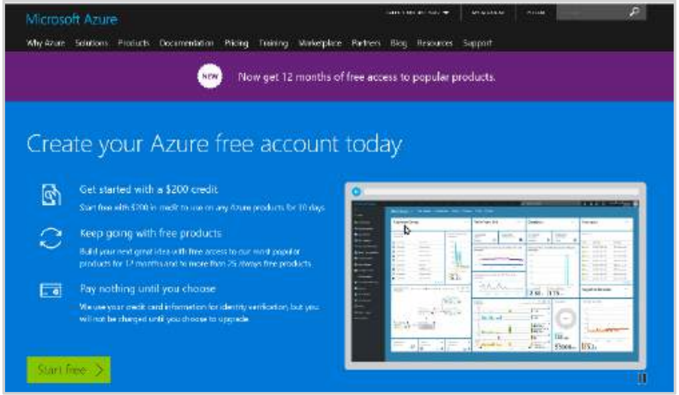
Resources

Product deep dive

AKS resources

- Azure Kubernetes Service (AKS)
- Containers on Azure pitch deck
- Smart Hotel 360 Demo
- Documentation resources
- Ebook for distributed systems
- Distributed system HoL
- AKS HoL

Sign up for a free Azure account



Now get 12 months of free access to popular products.

Create your Azure free account today

Get started with a $200 credit

Start free with $200 in credit to use on any Azure products for 30 days

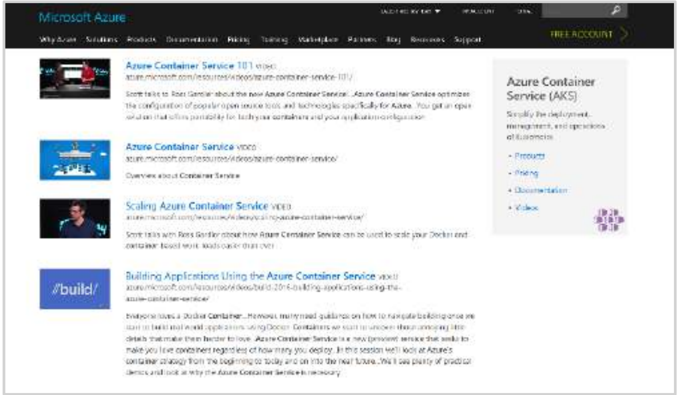
Keep going with free products

Build your next great idea with free access to our most popular products for 12 months and to more than 25 always free products

Pay nothing until you choose

We use your credit card information for identity verification, but you will not be charged until you choose to upgrade

Check out the Azure container videos page



Azure Container Service 101 video

azure.microsoft.com/resources/videos/azure-container-service-101/

Scott talks to Ross Gardler about the new Azure Container Service ...Azure Container Service optimizes the configuration of popular open source tools and technologies specifically for Azure. You get an open solution that offers portability for both your containers and your application configuration.

Azure Container Service video

azure.microsoft.com/resources/videos/azure-container-service/

Overview about Container Service

Scaling Azure Container Service video

azure.microsoft.com/resources/videos/scaling-azure-container-service/

Scott talks with Ross Gardler about how Azure Container Service can be used to scale your Docker and container based work loads easier than ever.

Building Applications Using the Azure Container Service video

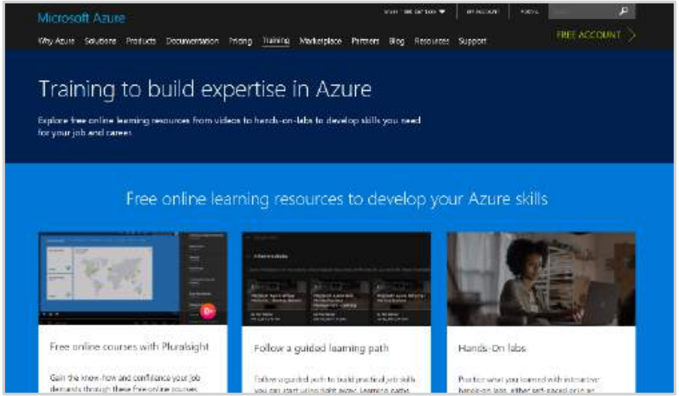
azure.microsoft.com/resources/videos/build-2016-building-applications-using-the-azure-container-service/

Everyone loves a Docker Container...However, many need guidance on how to navigate building once we start to build real world applications using Docker. Containers we start to uncover those annoying little details that make them harder to love. Azure Container Service is a new (preview) service that seeks to make you love containers regardless of how many you deploy. In this session we'll look at Azure's container strategy from the beginning to today and on into the near future...We'll see plenty of practical demos and look at why the Azure Container Service is necessary.

Azure Container Service (AKS)

Simplify the deployment, management, and operations of Kubernetes

Hone your skills with Azure training



Training to build expertise in Azure

Explore free online learning resources from videos to hands-on-labs to develop skills you need for your job and career.

Free online learning resources to develop your Azure skills

Free online courses with Pluralsight

Gain the know-how and confidence your job demands through these free online courses.

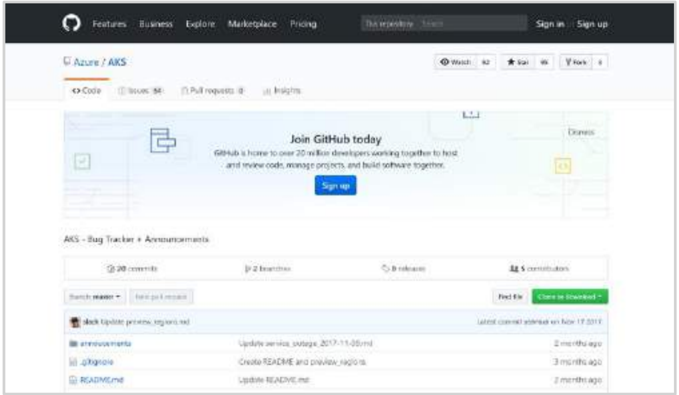
Follow a guided learning path

Follow a guided path to build practical job skills you can start using right away. Learning paths

Hands-On labs

Practice what you learned with interactive hands-on labs, either self-paced or in an

Get the code from GitHub



Azure / AKS

Join GitHub today

GitHub is home to over 20 million developers working together to host and review code, manage projects, and build software together.

AKS - Bug Tracker + Announcements

20 commits | 2 branches | 0 releases | 5 contributors

slack Update preview_regions.md — Latest commit on Nov 17 2017

announcements	Update service_outage_2017-11-05.md	2 months ago
.gitignore	Create README and preview_regions	3 months ago
README.md	Update README.md	2 months ago

Connect with us

Core team

PM: Gabe Monroy, @gabrtv

PM: Sean McKenna

PM: Jason Hansen

PMM: Stella Lin

CDA: Bryan Liston

Community

Brendan Burns, @brendandburns

Michelle Noorali

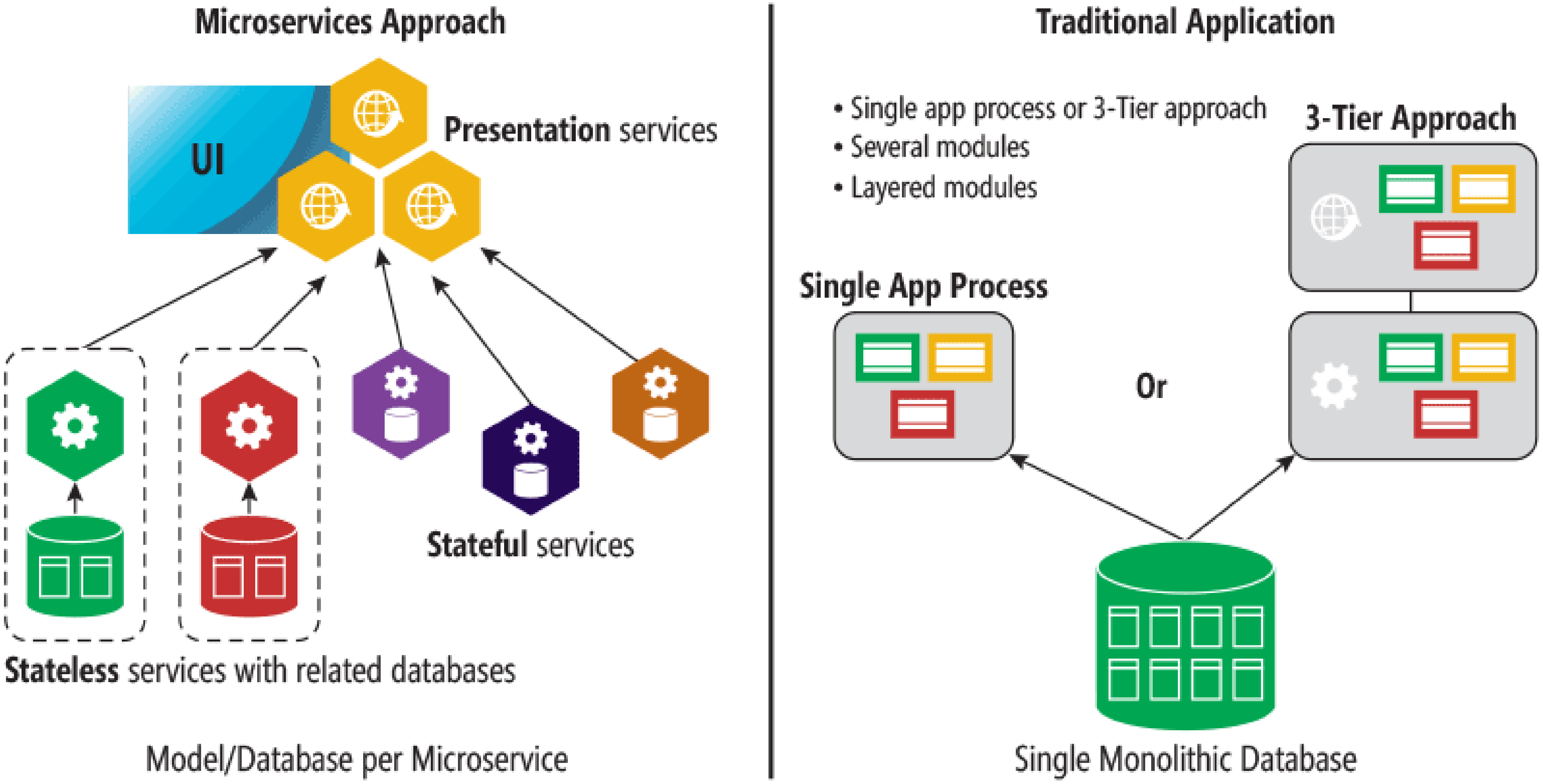
Partner team

Morgan Pettis

Leon Jones

Dan Sandlin

Microservices



microservices ≠ containers

microservices is an architectural design approach

containers are an implementation detail that often helps

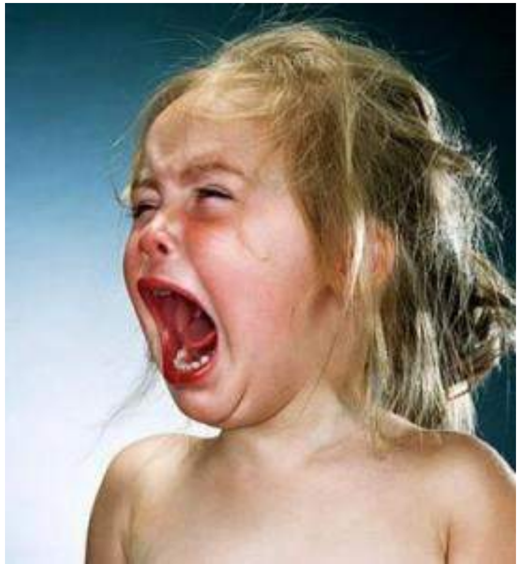
Microservices Benefits

- ✓ Independent deployments
- ✓ Enables continuous delivery
- ✓ No downtime upgrades
- ✓ Improved scale and resource utilization per service
- ✓ Fault isolation
- ✓ Security isolation
- ✓ Services can be distributed across multiple servers or environments
- ✓ Multiple languages / diversity
- ✓ Smaller, focused teams
- ✓ Code can be organized around business capabilities
- ✓ Autonomous developer teams

Microservices – The Hard Part

- ✓ Deployment is complex
- ✓ Testing is difficult
- ✓ Debugging is difficult
- ✓ Monitoring/Logging is difficult
- ✓ New service versions must support old/new API contracts
- ✓ Distributed databases make transactions hard
- ✓ Cluster and orchestration tools overhead

- ✓ Distributed services adds more network communication
  - ✓ Increased network hops
  - ✓ Requires failure/recovery code
  - ✓ Need service discovery solution
- ✓ Advanced DevOps capability: short-term pain for long-term gain



12-Factor Apps

http://12factor.net

12-Factor Apps (1-5)

1. Single root repo; don’t share code with another app
2. Deploy dependent libs with app
3. No config in code; read from environment vars
4. Handle unresponsive app dependencies robustly
5. Strictly separate build, release, & run steps
    - Build: Builds a version of the code repo & gathers dependencies
    - Release: Combines build with config ReleaseId (immutable)
    - Run: Runs app in execution environment

12-Factor Apps (6-12)

6. App executes as 1+ stateless process & shares nothing
7. App listens on ports; avoid using (web) host
8. Use processes for isolation; multiple for concurrency
9. Processes can crash/be killed quickly & start fast
10. Keep dev, staging, & prod environments similar
11. Log to stdout (dev=console; prod=file & archived)
12. Deploy & run admin tasks (scripts) as processes