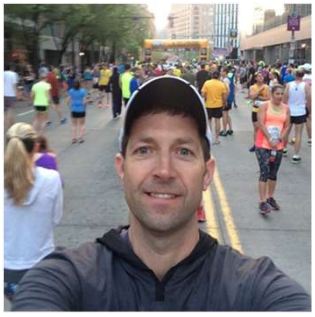
Who are we?



Brian Redmond
Microsoft
Azure Global Black Belt Team
@chzbrgr71

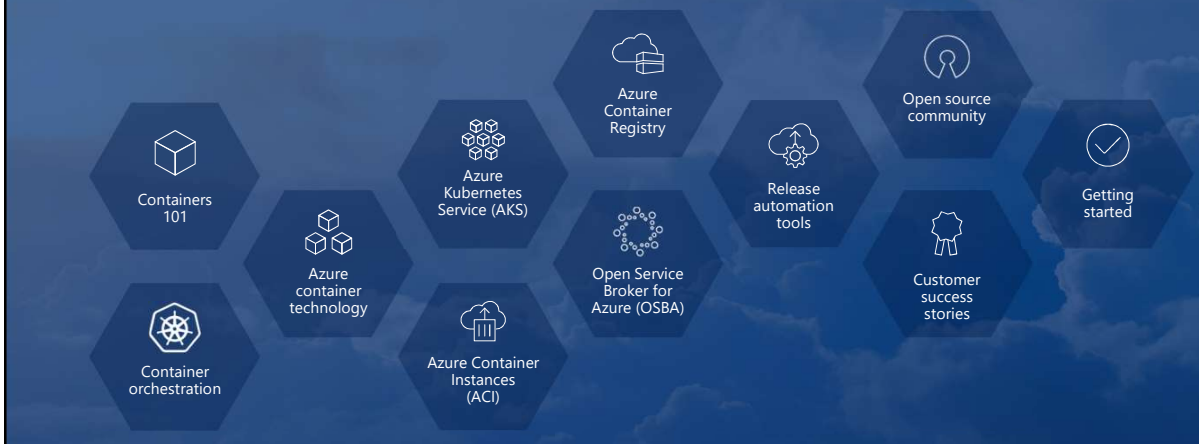
Joey Schluchter
Microsoft
Azure Global Black Belt Team
@sonofjorel

Partner Name
Company
Team Name
@twitterhandle

Agenda

Time	Topic
8:30 – 9:00	Breakfast and get settled
9:00 – 9:45	Cloud Native Applications, Containers, and Docker
9:45 – 11:00	**HANDS-ON: Labs 1, 2**
11:00 – 11:45	Kubernetes Overview, Azure Kubernetes Service, Kubernetes Concepts
11:45	**Working Lunch**
11:45 – 1:00	**HANDS-ON: Labs 3, 4, 5**
1:00 – 1:15	Azure Containers Deep Dive: ACI, Open Service Broker, and more. Operations / Management Concepts
1:15– 2:15	**HANDS-ON: Labs 6, 7**
2:15 – 2:45	Managing Storage and State, and utilizing Azure PaaS (CosmosDB)
2:45 – 4:00	**HANDS-ON: Labs 8, 9, 10**
2:15 – 2:45	Q&A, wrap-up, and next steps

Table of contents



Why should customers care about **containers** and **microservices**?

In reality, they shouldn't…

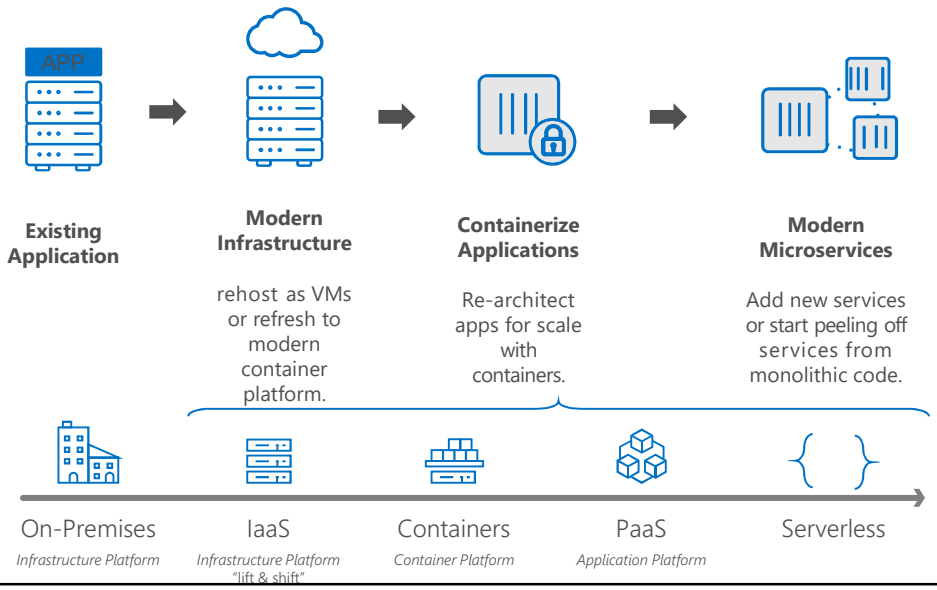
They do care about cloud native applications

"Unlimited" Scale

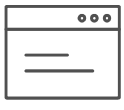
Global reach

Rapid innovation -
> time to market

From traditional app to modern app



What we hear from developers



I need to create applications at a competitive rate without worrying about IT

New applications run smoothly on my machine but malfunction on traditional IT servers

My productivity and application innovation become suspended when I have to wait on IT

What we hear from **IT**

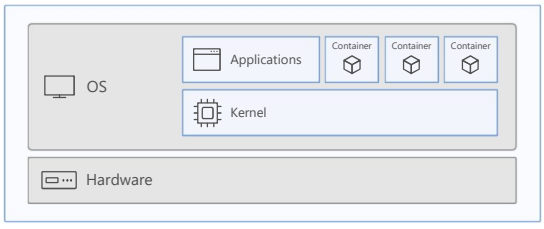
I need to manage servers and maintain compliance with little disruption

I'm unsure of how to integrate unfamiliar applications, and I require help from developers

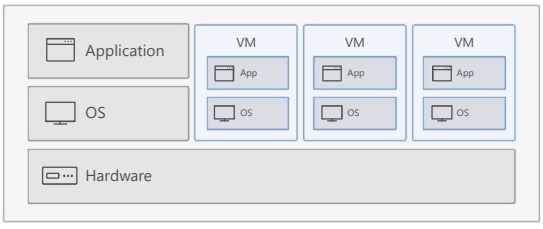
I'm unable to focus on both server protection and application compliance

What is a **container**?

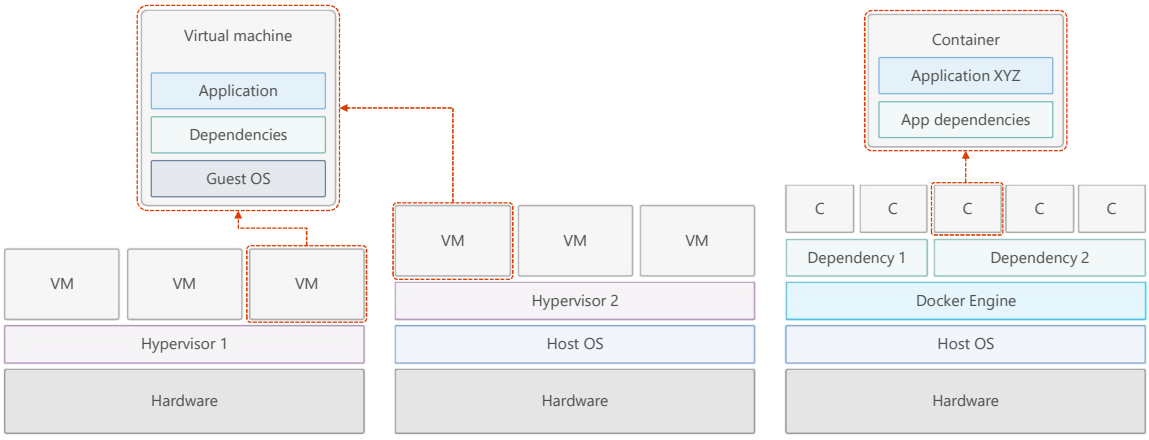
Containers = operating system virtualization



Traditional virtual machines = hardware virtualization



Virtualization versus **containerization**



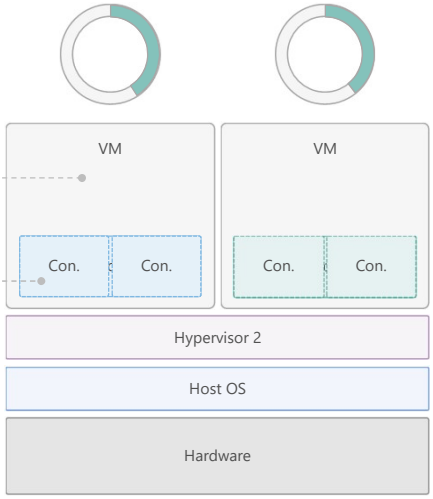
Virtualization **Containerization**

The container **advantage**

Traditional virtualized environment

Low utilization of container resources

Containerization of applications and their dependencies

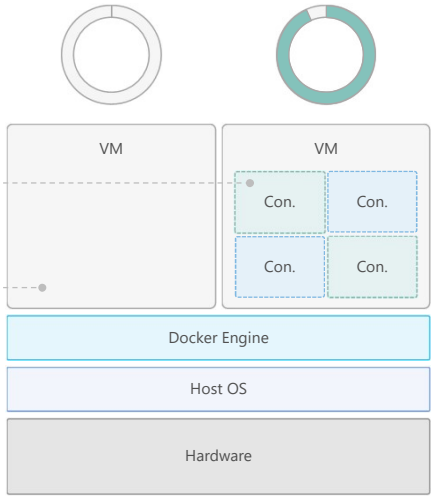


The container **advantage**

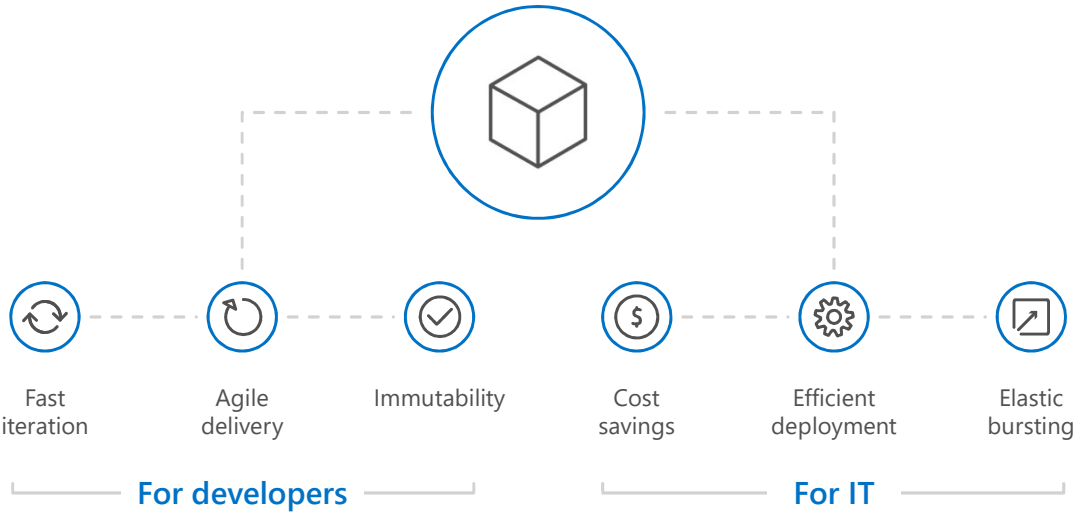
Containerized environment

Migrate containers and their dependencies to underutilized VMs for improved density and isolation

Decommission unused resources for efficiency gains and cost savings



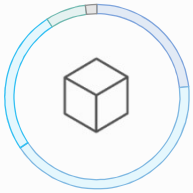
The container **advantage**



Containers are gaining **momentum**

Does your organization currently use container technologies?[1]

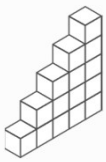
- 23% My org. is **evaluating** container technologies
- 42% **Yes**, my org. currently uses container technologies
- 25% **No**, my org. is not using container technologies
- 7% **Not sure**
- 2% **Not applicable**



Larger companies are leading adoption.[2]

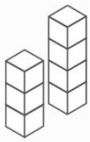
Nearly **60% percent** of organizations running 500 or more hosts are classified as **container dabblers** or adopters.

The average company QUINTUPLES its container usage within 9 months.[1]



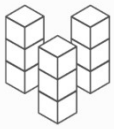
5X

Container hosts often run SEVEN containers at a time.[1]



7X

Containers churn 9 times FASTER than VMs.[1]



9X

Source:
1: Datadog: 8 Surprising Facts About Real Docker Adoption; 2: DZone: The DZone Guide to Deploying and Orchestrating Containers

Industry analysts **agree**

"By 2020, more than 50% of enterprises will run mission-critical, containerized cloud-native applications in production, up from less than 5% today."

Gartner

The elements of **orchestration**

- Scheduling
- Affinity/anti-affinity
- Health monitoring
- Failover
- Scaling
- Networking
- Service discovery
- Coordinated app upgrades

Container Management at Scale

Cluster Management: deploy and manage cluster resources	**Scheduling:** where containers run	**Lifecycle & Health:** keep containers running despite failure	**Service Discovery & Load Balancing:** where are my containers and how to connect	**Security:** control access to cluster resources and protect secrets
Scaling: make sets of containers elastic in number	**Image repository:** centralized, secure Docker container images	**Continuous Delivery:** CI/CD pipeline and workflow	**Logging & Monitoring:** track what's happening in containers and cluster	**Storage volumes:** persistent data for containers

Kubernetes: the de-facto orchestrator

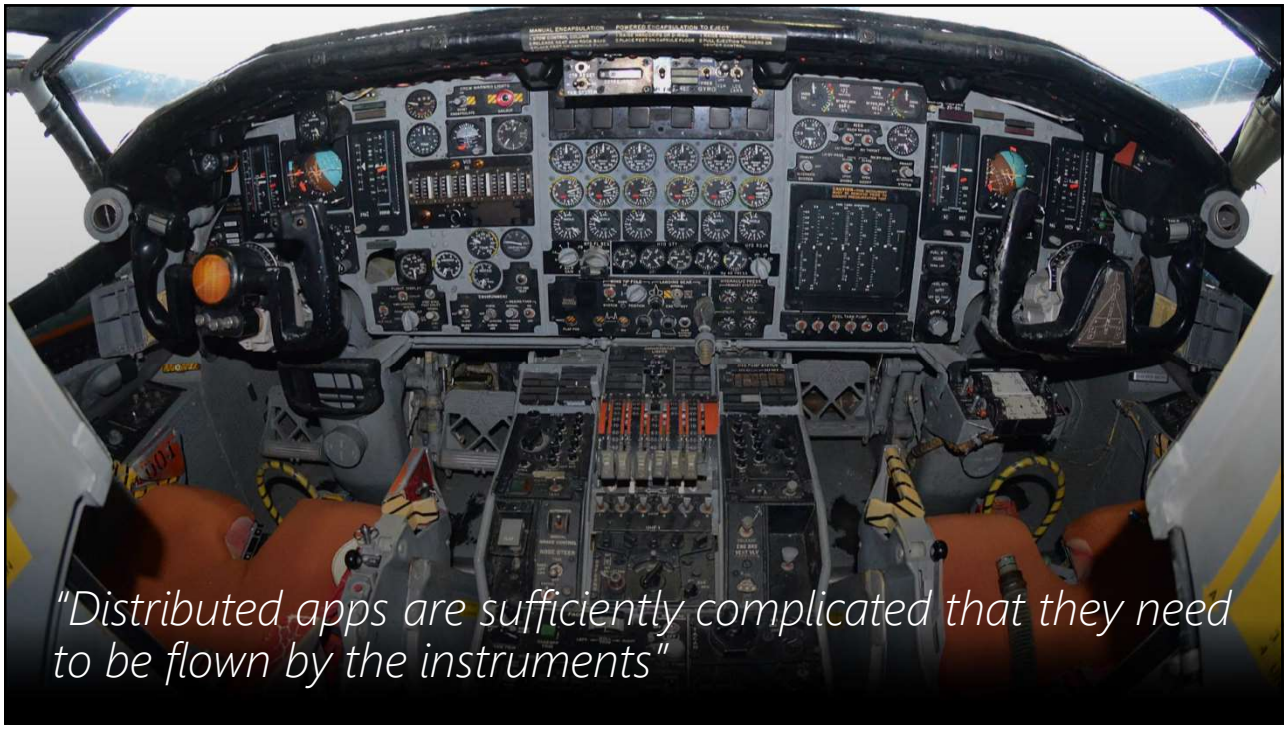
Portable
Public, private, hybrid, multi-cloud

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



Azure container technology

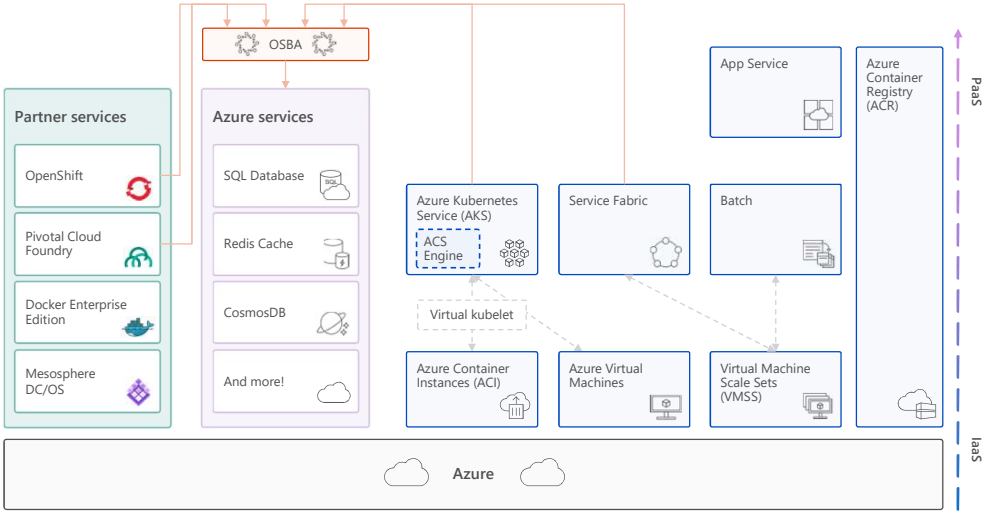
Azure container **strategy**

Embrace containers as ubiquitous

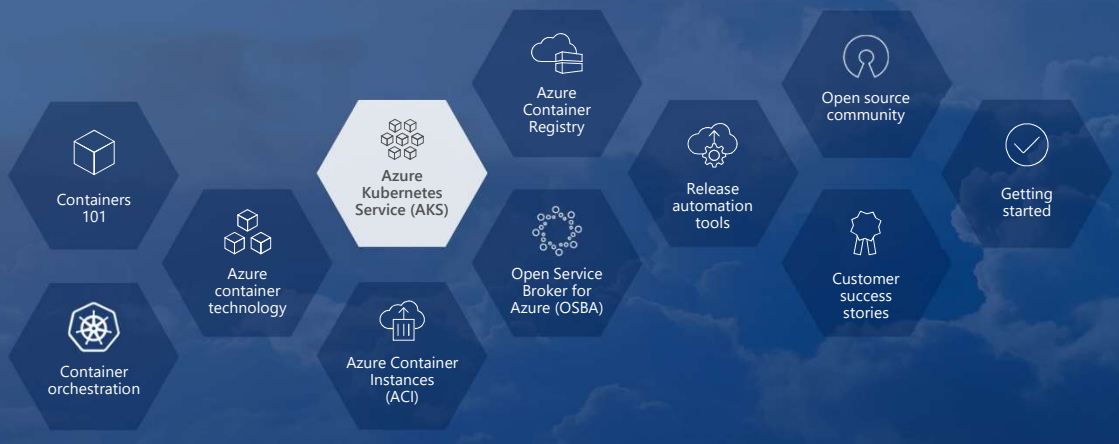
Support containers across the compute portfolio

Democratize container technology

Azure container **ecosystem**



Azure Kubernetes Service (AKS)



Azure Kubernetes Service (AKS)

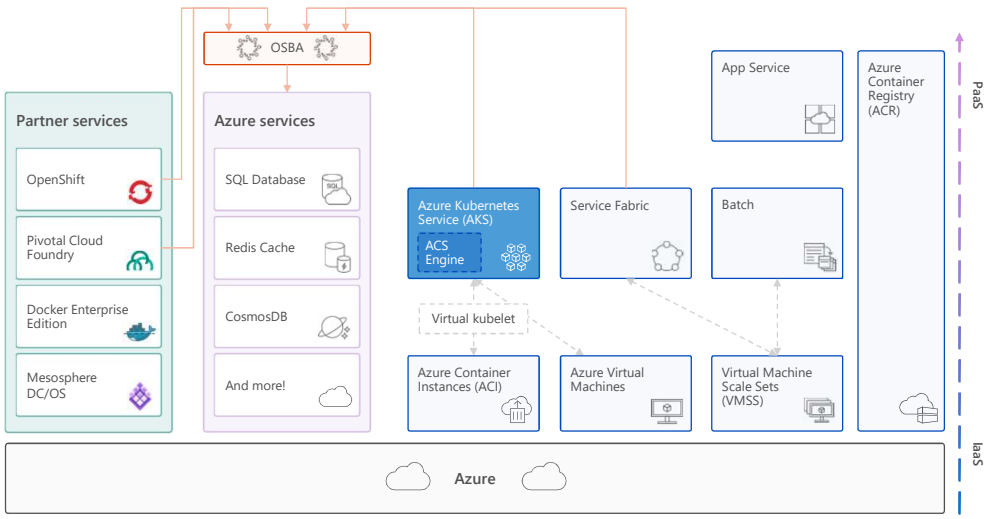
Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Azure Kubernetes Service (AKS)



Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

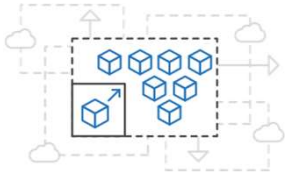
Release Automation Tools

Azure Kubernetes Service (AKS)

Simplify the deployment, management, and operations of Kubernetes

Focus on your containers not the infrastructure

Work how you want with open-source APIs



Scale and run applications with confidence

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

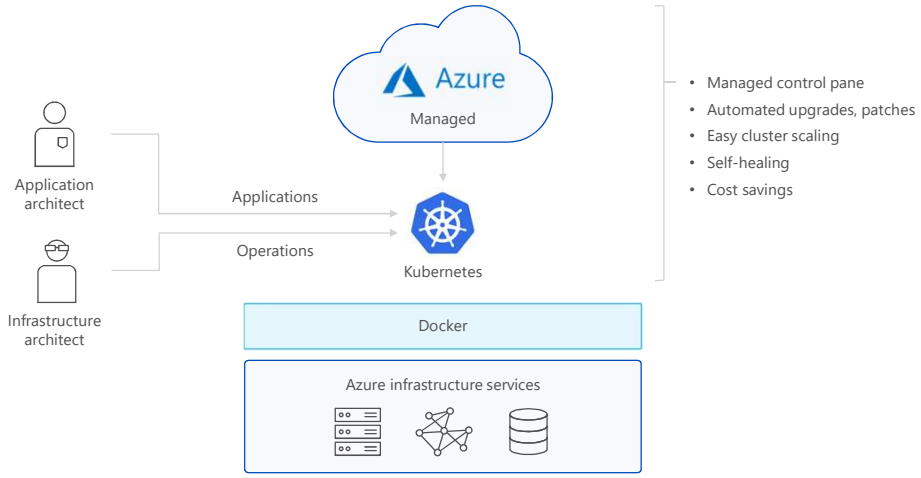
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Azure Kubernetes Service (AKS)

A fully managed Kubernetes cluster



Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Azure Kubernetes Service (AKS)

Get started easily

```
$ az aks create -g myResourceGroup -n myCluster --generate-ssh-keys
\ Running ..

$ az aks install-cli
Downloading client to /usr/local/bin/kubectl ..

$ az aks get-credentials -g myResourceGroup -n myCluster
Merged "myCluster" as current context ..

$ kubectl get nodes
NAME                       STATUS    AGE       VERSION
aks-mycluster-36851231-0   Ready     4m        v1.8.1
aks-mycluster-36851231-1   Ready     4m        v1.8.1
aks-mycluster-36851231-2   Ready     4m        v1.8.1
```

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Azure Kubernetes Service (AKS)

Manage an AKS cluster

```
$ az aks list -o table
Name              Location    ResourceGroup     KubernetesRelease  ProvisioningState
----------------  ----------  --------------  -------------------  -------------------
myCluster         westus2     myResourceGroup              1.7.7  Succeeded

$ az aks upgrade -g myResourceGroup -n myCluster --kubernetes-version 1.8.1
\ Running ..

$ kubectl get nodes
NAME                       STATUS    AGE       VERSION
aks-mycluster-36851231-0   Ready     12m       v1.8.1
aks-mycluster-36851231-1   Ready     8m        v1.8.1
aks-mycluster-36851231-2   Ready     3m        v1.8.1

$ az aks scale -g myResourceGroup -n myCluster --agent-count 10
\ Running ..
```

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

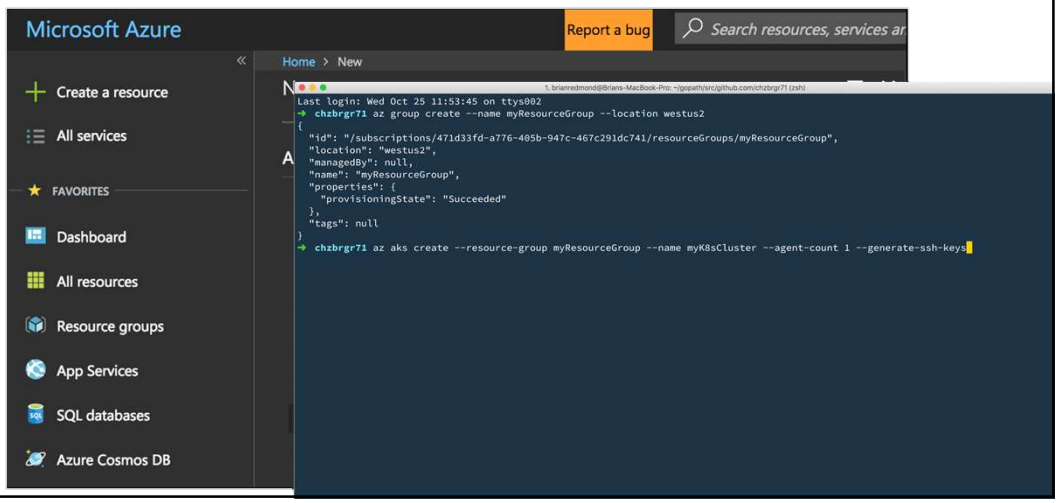
Azure Container Registry

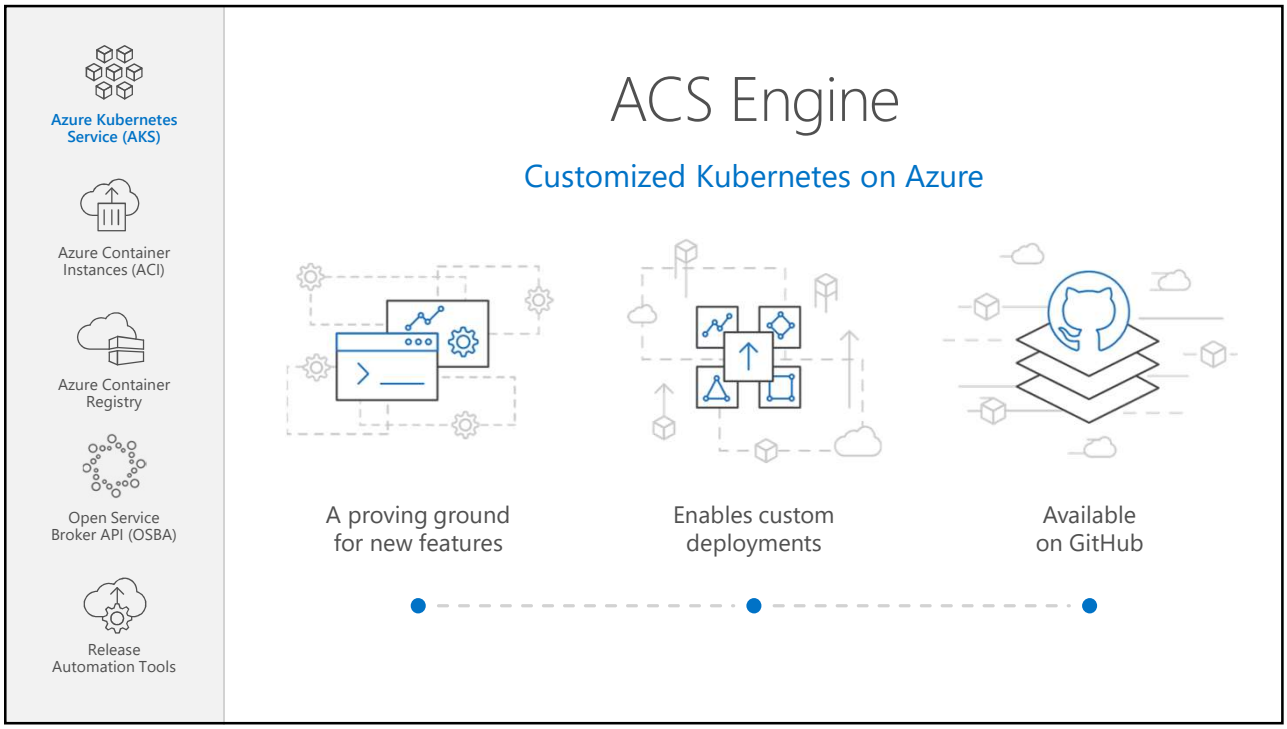
Open Service Broker API (OSBA)

Release Automation Tools

Azure Kubernetes Service (AKS)

Create AKS via Portal, Azure CLI, or ARM template





Kubernetes Concepts

Kubernetes Architecture Components

api-server

etcd

controller-manager

scheduler

master components

* Azure Managed

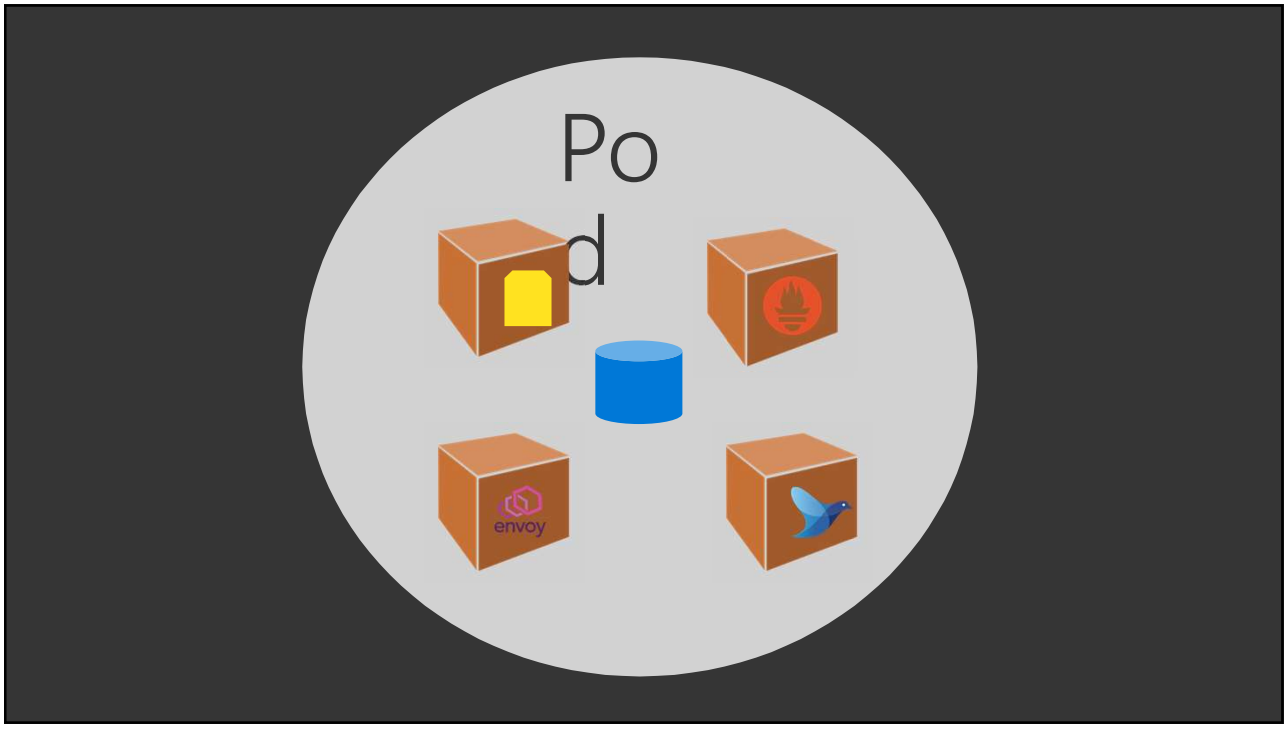
kubelet

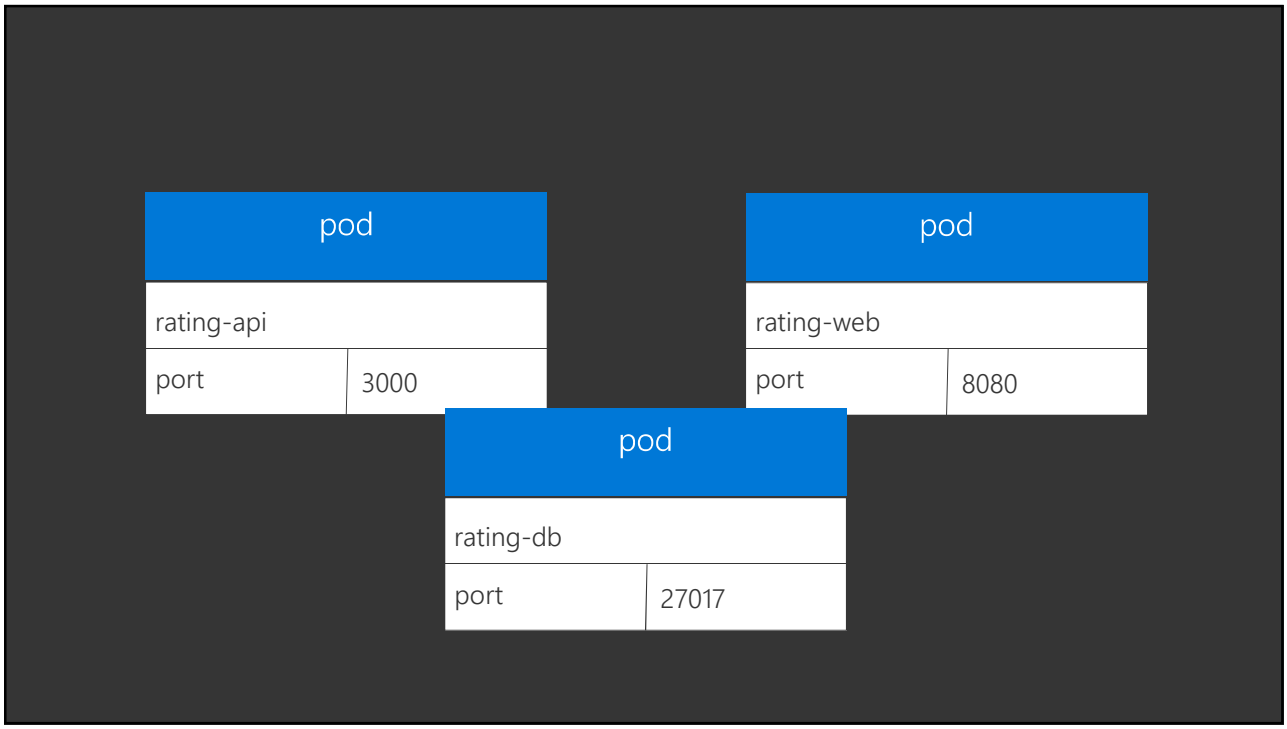
kube-proxy

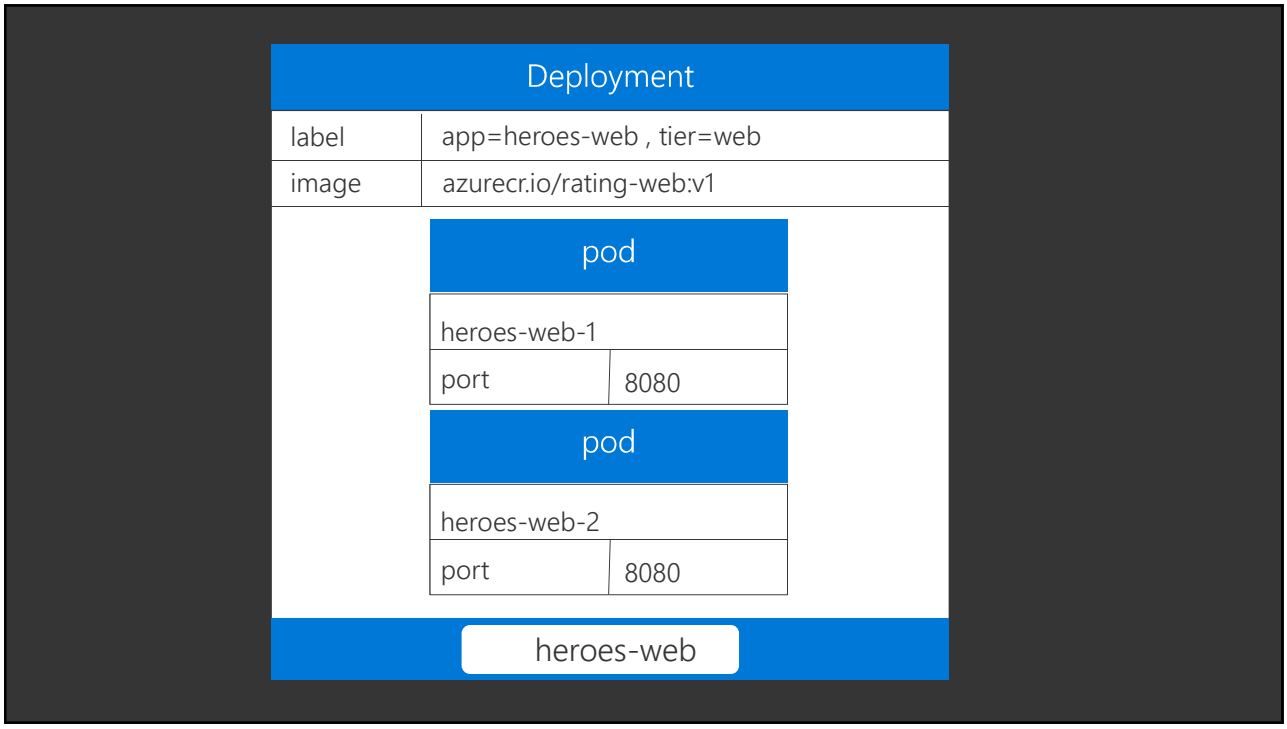
docker

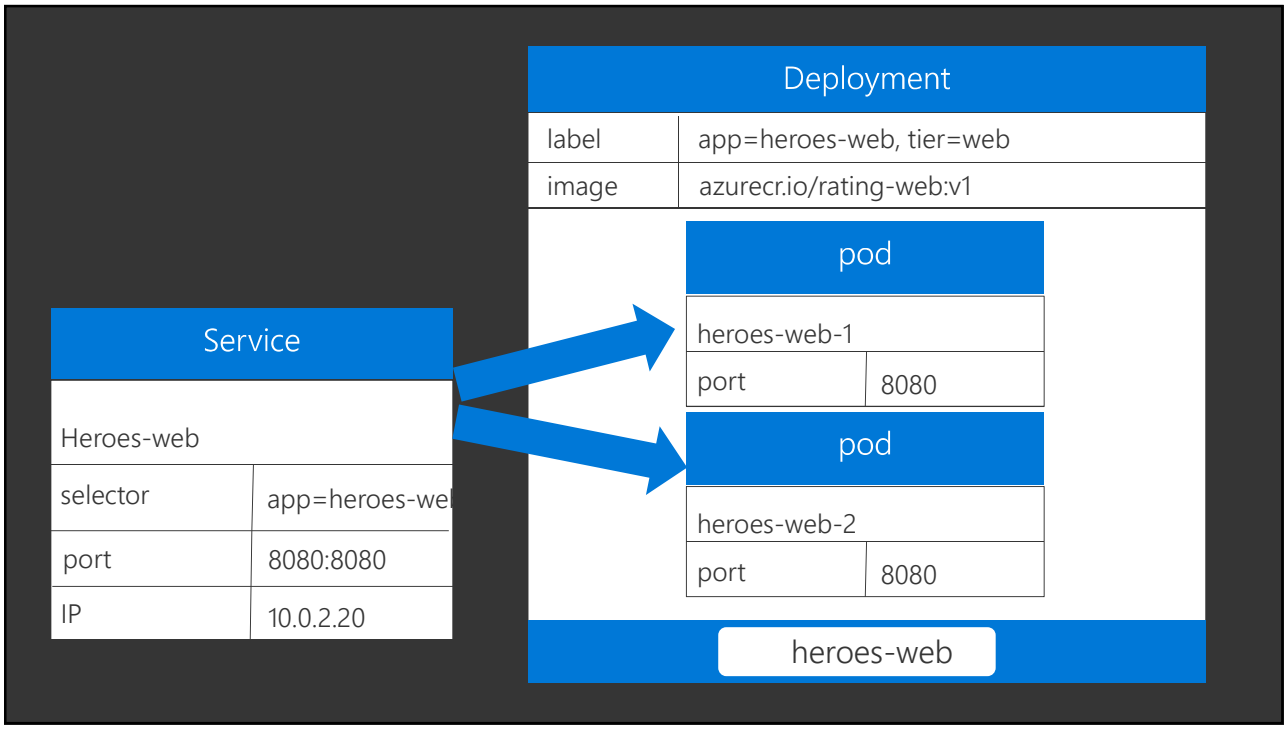
dns

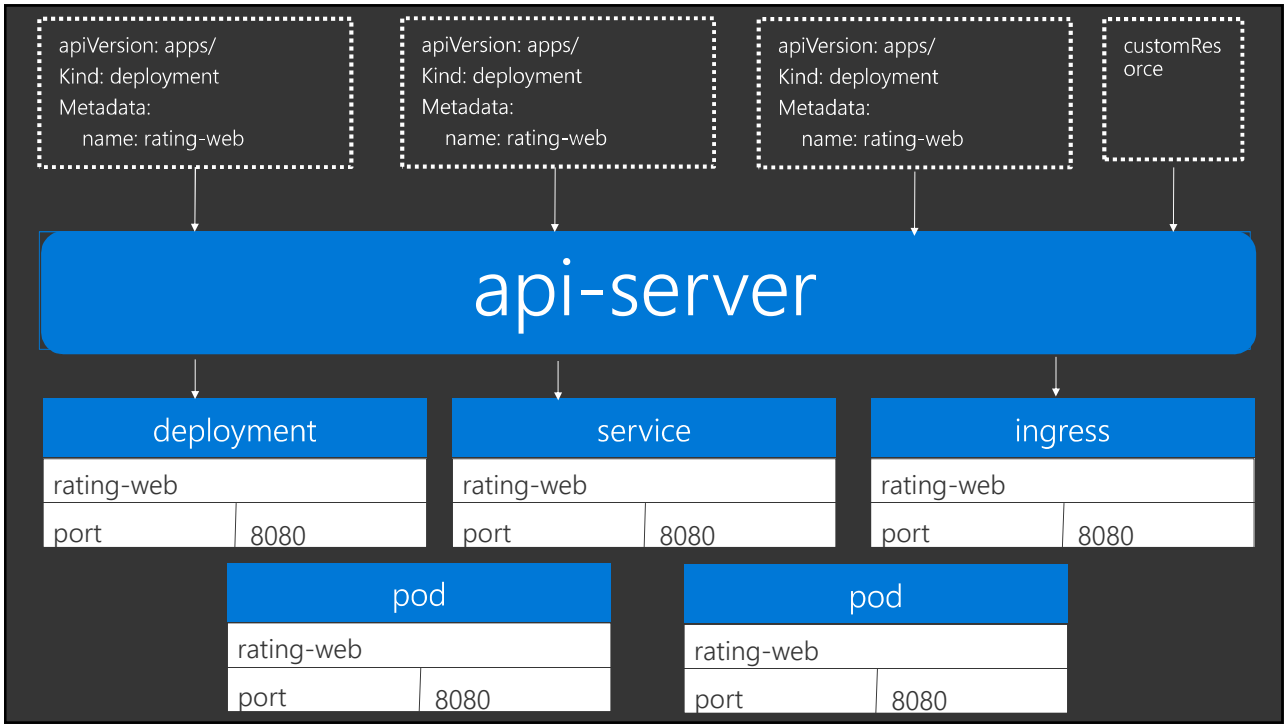
node components



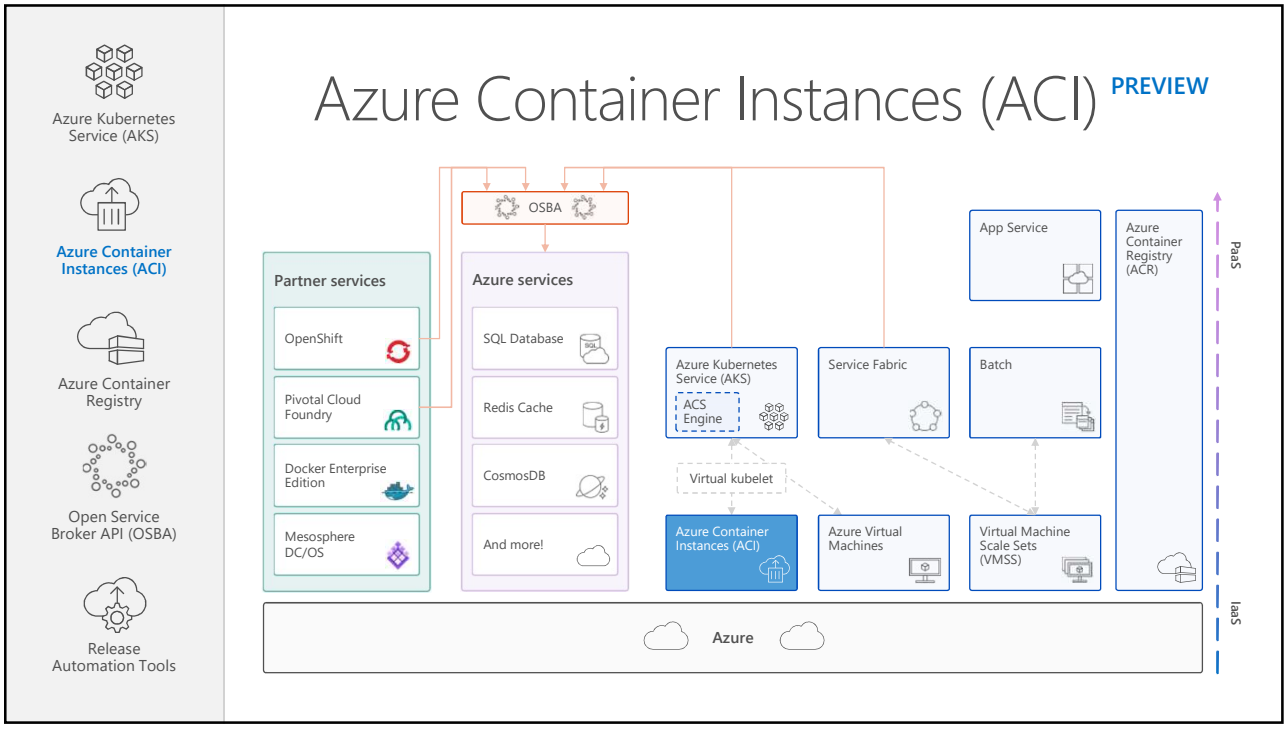

pod
rating-api
port
3000
pod
rating-web
port
8080
pod
rating-db
port
27017


Deployment
label
app=heroes-web , tier=web
image
azurecr.io/rating-web:v1
pod
heroes-web-1
port
8080
pod
heroes-web-2
port
8080
heroes-web


Deployment
label
app=heroes-web, tier=web
image
azurecr.io/rating-web:v1
pod
heroes-web-1
port
8080
pod
heroes-web-2
port
8080
heroes-web
Service
Heroes-web
selector
app=heroes-we
port
8080:8080
IP
10.0.2.20


apiVersion: apps/
Kind: deployment
Metadata:
name: rating-web
apiVersion: apps/
Kind: deployment
Metadata:
name: rating-web
apiVersion: apps/
Kind: deployment
Metadata:
name: rating-web
customRes
orce
api-server
deployment
rating-web
port
8080
service
rating-web
port
8080
ingress
rating-web
port
8080
pod
rating-web
port
8080
pod
rating-web
port
8080


Azure Container Instances (ACI)
Containers 101
Azure container technology
Container orchestration
Azure Kubernetes Service (AKS)
Azure Container Registry
Azure Container Instances (ACI)
Open Service Broker for Azure (OSBA)
Release automation tools
Open source community
Customer success stories
Getting started


Azure Kubernetes Service (AKS)
Azure Container Instances (ACI)
Azure Container Registry
Open Service Broker API (OSBA)
Release Automation Tools
Azure Container Instances (ACI) PREVIEW
OSBA
Partner services
OpenShift
Pivotal Cloud Foundry
Docker Enterprise Edition
Mesosphere DC/OS
Azure services
SQL Database
Redis Cache
CosmosDB
And more!
Azure Kubernetes Service (AKS)
ACS Engine
Service Fabric
Batch
App Service
Azure Container Registry (ACR)
Virtual kubelet
Azure Container Instances (ACI)
Azure Virtual Machines
Virtual Machine Scale Sets (VMSS)
Azure
PaaS
IaaS

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

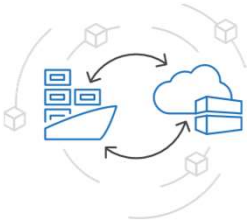
Azure Container Registry

Open Service Broker API (OSBA)

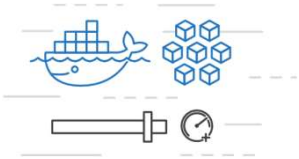
Release Automation Tools

Azure Container Instances (ACI) PREVIEW

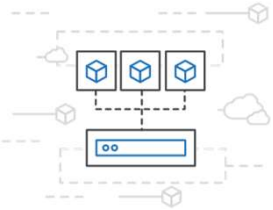
Easily run containers on Azure with a single command



Start using containers right away



Cloud-scale container capacity



Hyper-visor isolation

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Azure Container Instances (ACI) PREVIEW

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>
```

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

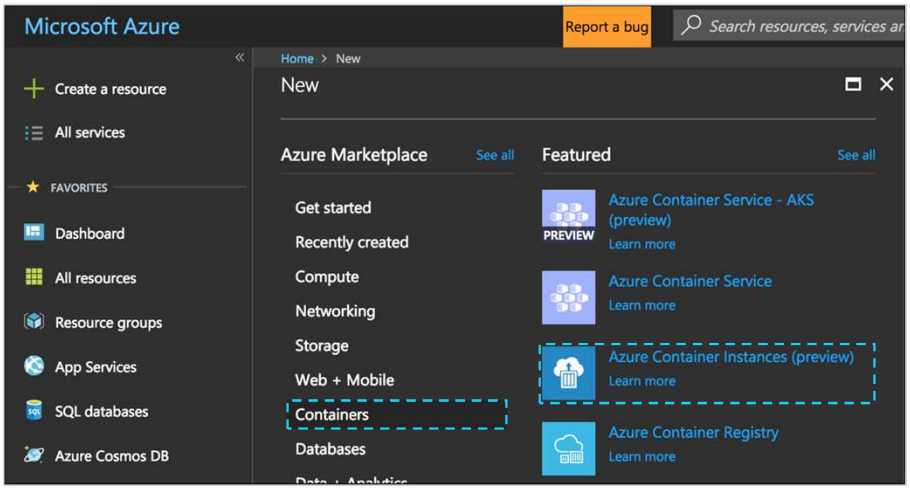
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Azure Container Instances (ACI) PREVIEW

Create an Azure Container Instance quickly



Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

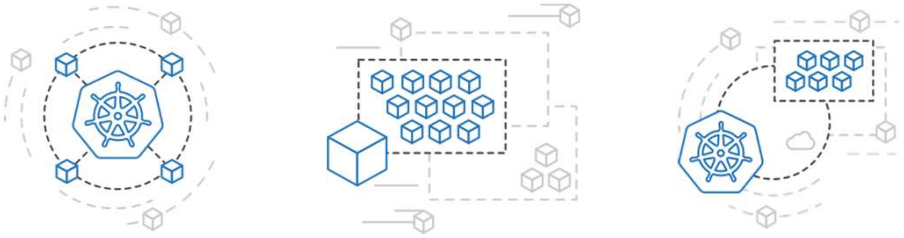
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Azure Container Instances (ACI) PREVIEW

ACI Connector for Kubernetes



Kubernetes provides rich orchestration capabilities

ACI provides infinite container-based scale

The ACI Connector for K8s brings them together

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

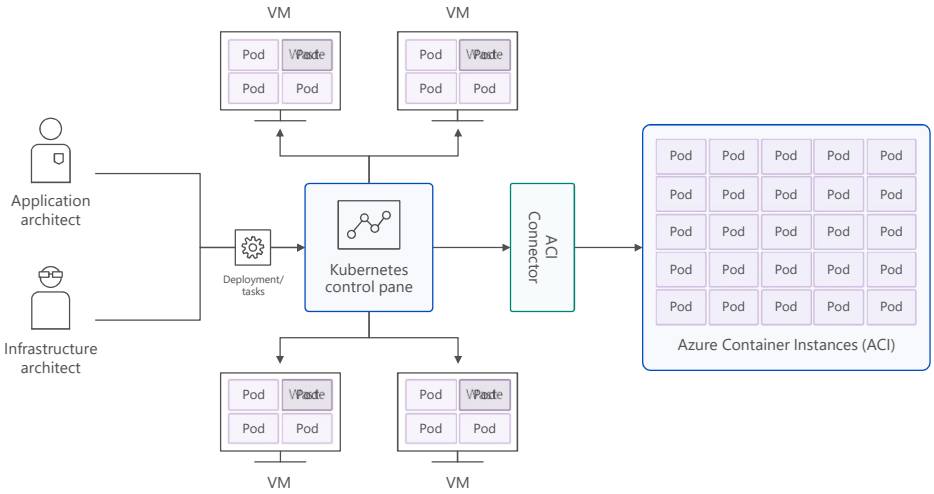
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Azure Container Instances (ACI) PREVIEW

Bursting with the ACI Connector



Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

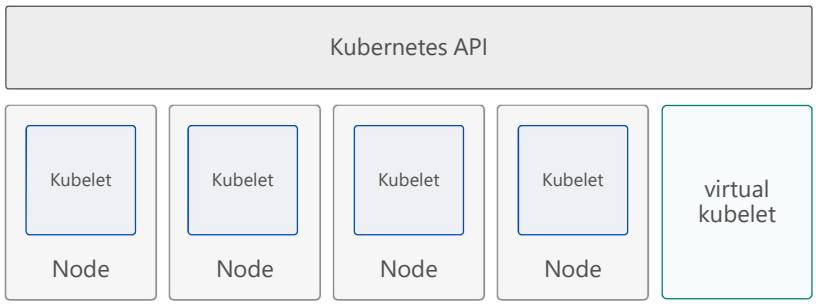
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Azure Container Instances (ACI) PREVIEW

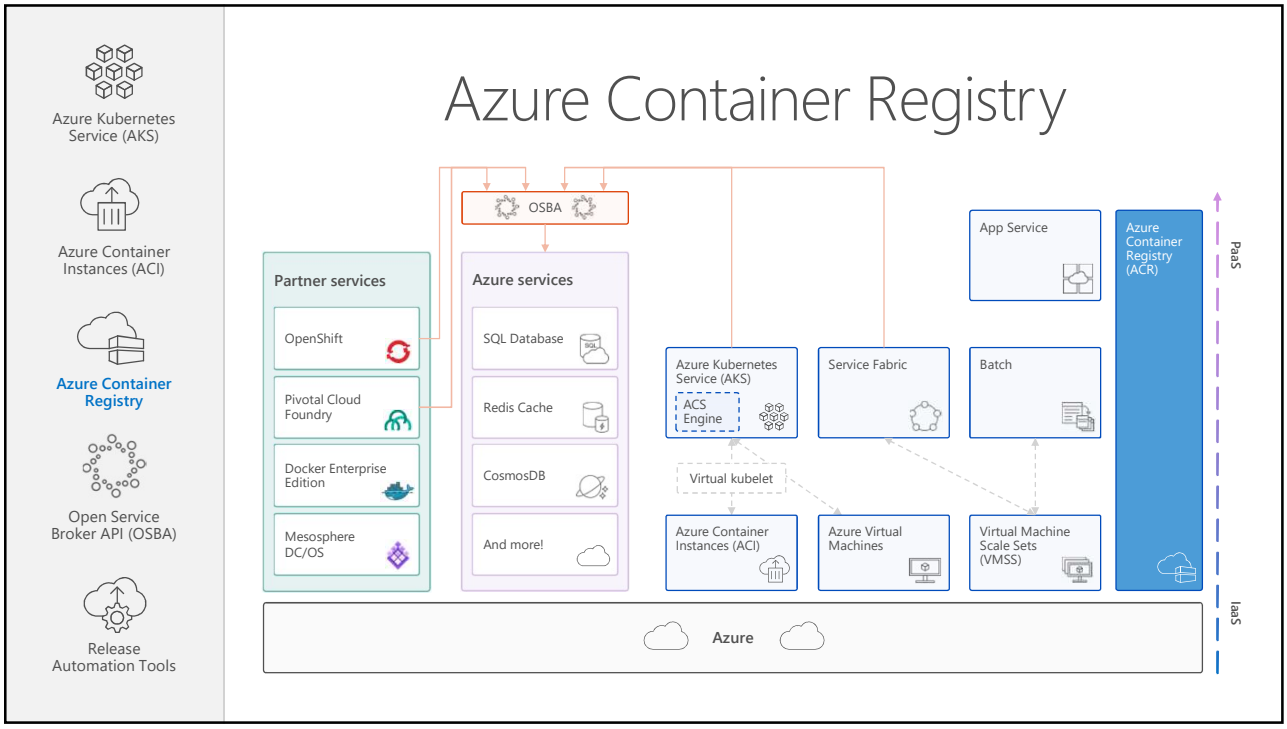
Virtual Kubelet



Typical kubelets implement the pod and container operations for each node as usual.

Virtual kublet registers itself as a "node" and allows developers to program their own behaviors for operations on pods and containers.


Azure Container Registry
Azure Container Registry
Containers 101
Azure Kubernetes Service (AKS)
Open source community
Azure container technology
Open Service Broker for Azure (OSBA)
Release automation tools
Getting started
Container orchestration
Azure Container Instances (ACI)
Customer success stories


Azure Kubernetes Service (AKS)
Azure Container Instances (ACI)
Azure Container Registry
Open Service Broker API (OSBA)
Release Automation Tools
Azure Container Registry
OSBA
Partner services
OpenShift
Pivotal Cloud Foundry
Docker Enterprise Edition
Mesosphere DC/OS
Azure services
SQL Database
Redis Cache
CosmosDB
And more!
Azure Kubernetes Service (AKS)
ACS Engine
Service Fabric
Batch
App Service
Azure Container Registry (ACR)
Virtual kubelet
Azure Container Instances (ACI)
Azure Virtual Machines
Virtual Machine Scale Sets (VMSS)
Azure
PaaS
IaaS

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

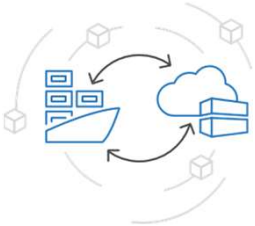
Open Service Broker API (OSBA)

Release Automation Tools

Azure Container Registry

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

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

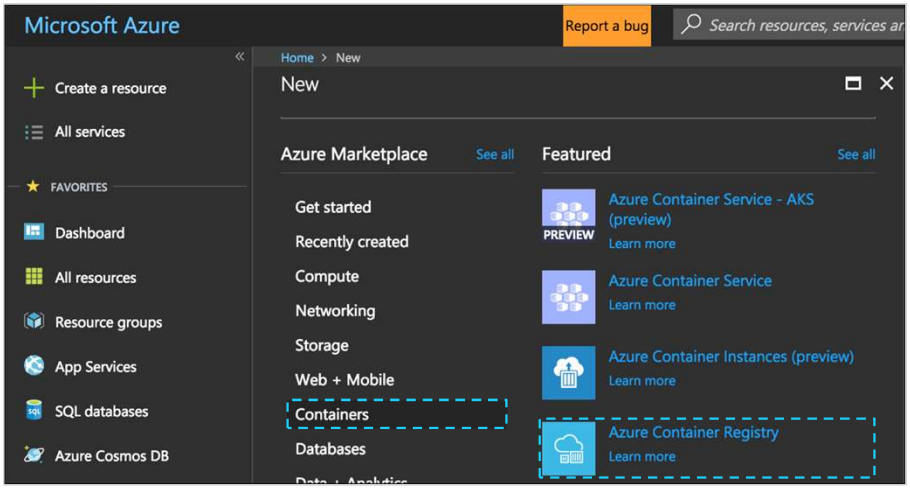
Azure Container Registry

Open Service Broker API (OSBA)

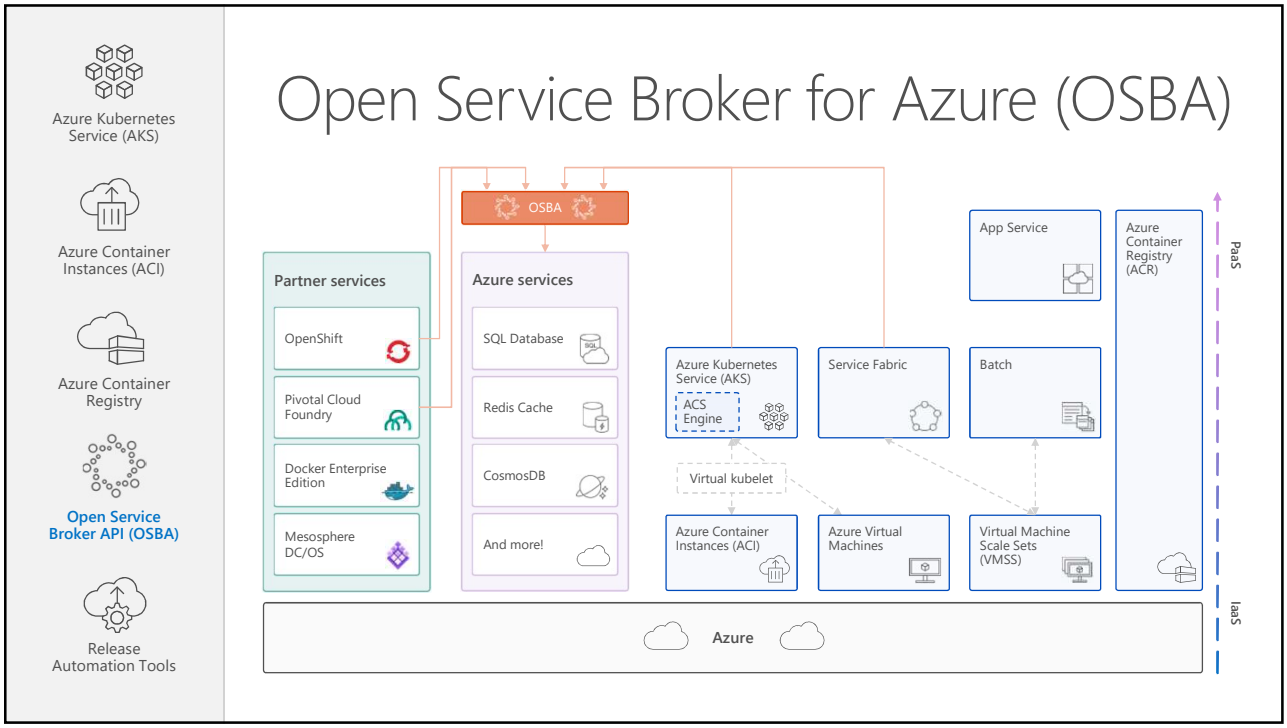
Release Automation Tools

Azure Container Registry

Create a container in the Registry quickly




Open Service Broker for Azure
Containers 101
Azure container technology
Container orchestration
Azure Kubernetes Service (AKS)
Azure Container Registry
Azure Container Instances (ACI)
Open Service Broker for Azure (OSBA)
Release automation tools
Open source community
Customer success stories
Getting started


Azure Kubernetes Service (AKS)
Azure Container Instances (ACI)
Azure Container Registry
Open Service Broker API (OSBA)
Release Automation Tools
Open Service Broker for Azure (OSBA)
OSBA
Partner services
OpenShift
Pivotal Cloud Foundry
Docker Enterprise Edition
Mesosphere DC/OS
Azure services
SQL Database
Redis Cache
CosmosDB
And more!
Azure Kubernetes Service (AKS)
ACS Engine
Service Fabric
Batch
Virtual kubelet
Azure Container Instances (ACI)
Azure Virtual Machines
Virtual Machine Scale Sets (VMSS)
App Service
Azure Container Registry (ACR)
PaaS
IaaS
Azure

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Open Service Broker for Azure (OSBA)

Connecting containers to Azure services and platforms

A standardized way to connect with Azure services

Simple and flexible service integration

Compatible across numerous platforms

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

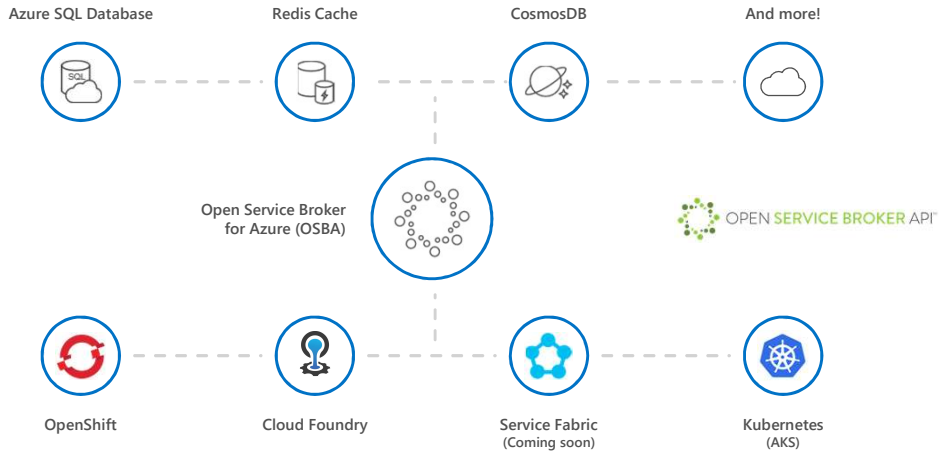
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Open Service Broker for Azure (OSBA)

An implementation of the Open Service Broker API



Azure Container Instances (ACI)

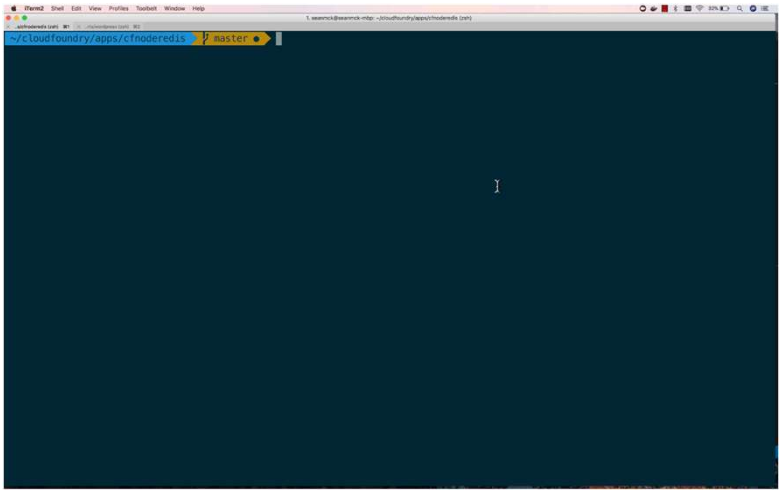
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Open Service Broker for Azure (OSBA)

OSBA in action



Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Open Service Broker for Azure (OSBA)

Getting started with ease

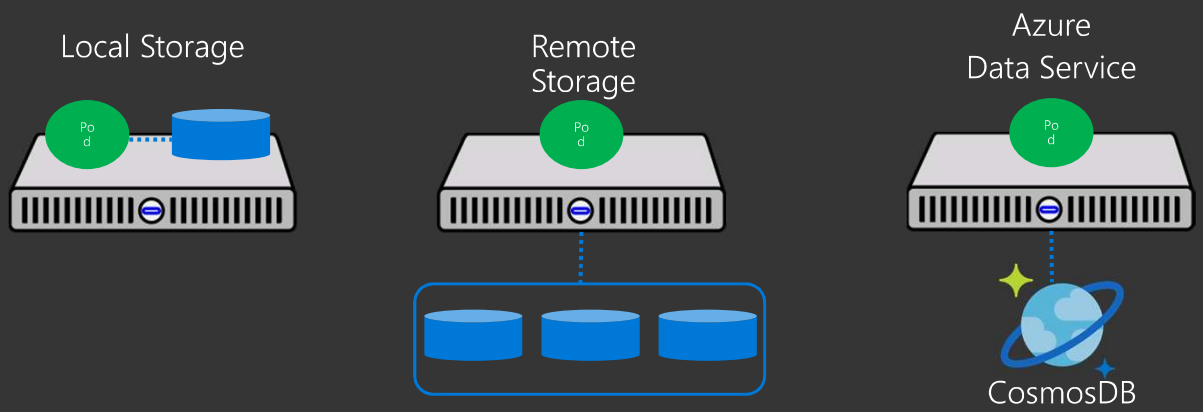
```
$ helm repo add azure Azure/helm-charts

$ helm install azure/service-broker

$ helm install azure/wordpress
```

Managing Storage and State

Managing State For Containers

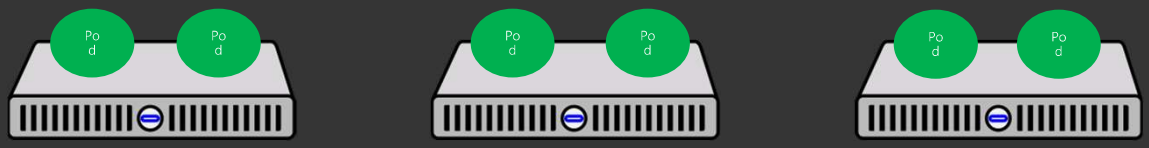


Scaling Applications

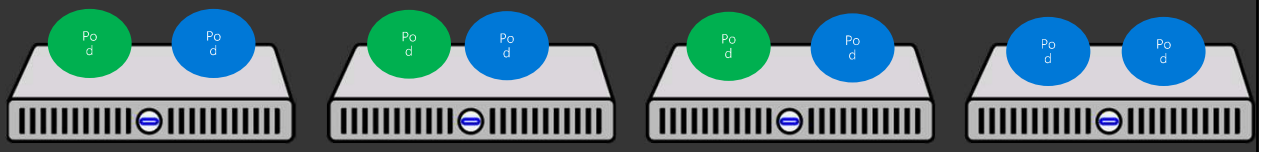
Application Scaling Strategies

- Infrastructure Level
  - Increase/decrease the number of VM's running in the cluster
  - Azure infrastructure function
  - via AKS API/CLI (utilizes Azure Availability Sets)
  - Auto-scaling coming to AKS soon
- Application Level
  - Increase/decrease the number of pods for a given service
  - Handled by the orchestrator or the application itself
  - Kubernetes Horizontal Pod Autoscaling
- These functions should be coordinated

Scaling Pods



Scaling Host



Scaling the ACS Cluster

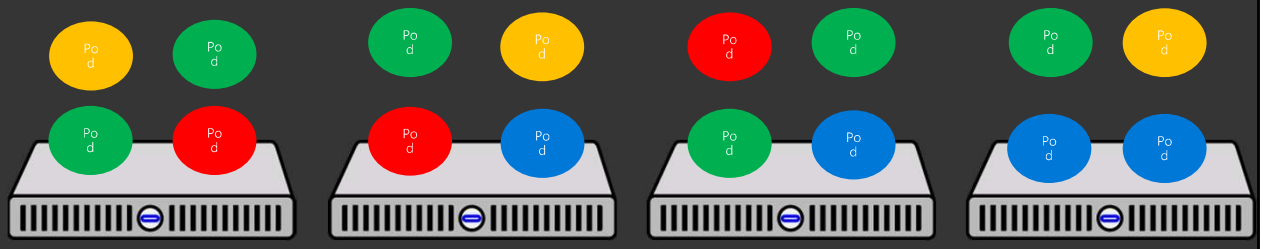
Azure CLI | Copy | Try It

```
az aks scale --name myK8sCluster --resource-group myResourceGroup --agent-count 1
```

Monitoring

Current monitoring is static in nature

Need to monitor for dynamic environments



Application Health and Readiness

- Kubelets on workers use liveness and readiness probes to know when to restart a container or start directing traffic
  - Liveness check: when to restart
  - Readiness check: when to forward Service traffic to a pod

Monitoring: What indicators matter in Kubernetes

- Cluster health: total number of nodes, pods, etc
- Node health: available compute resources, health
- Application health
- Tooling Examples
  - Datadog
  - Prometheus
  - InfluxData
  - Cloud-specific: OMS (Azure), CloudWatch (AWS)

Logging

- Pod logs: applications must log to standard out!
- Cluster-wide event logs
- Logging forwarder solutions
  - fluentd
  - logstash
- Log indexers
  - OMS Log Analytics
  - Splunk
  - ELK stack

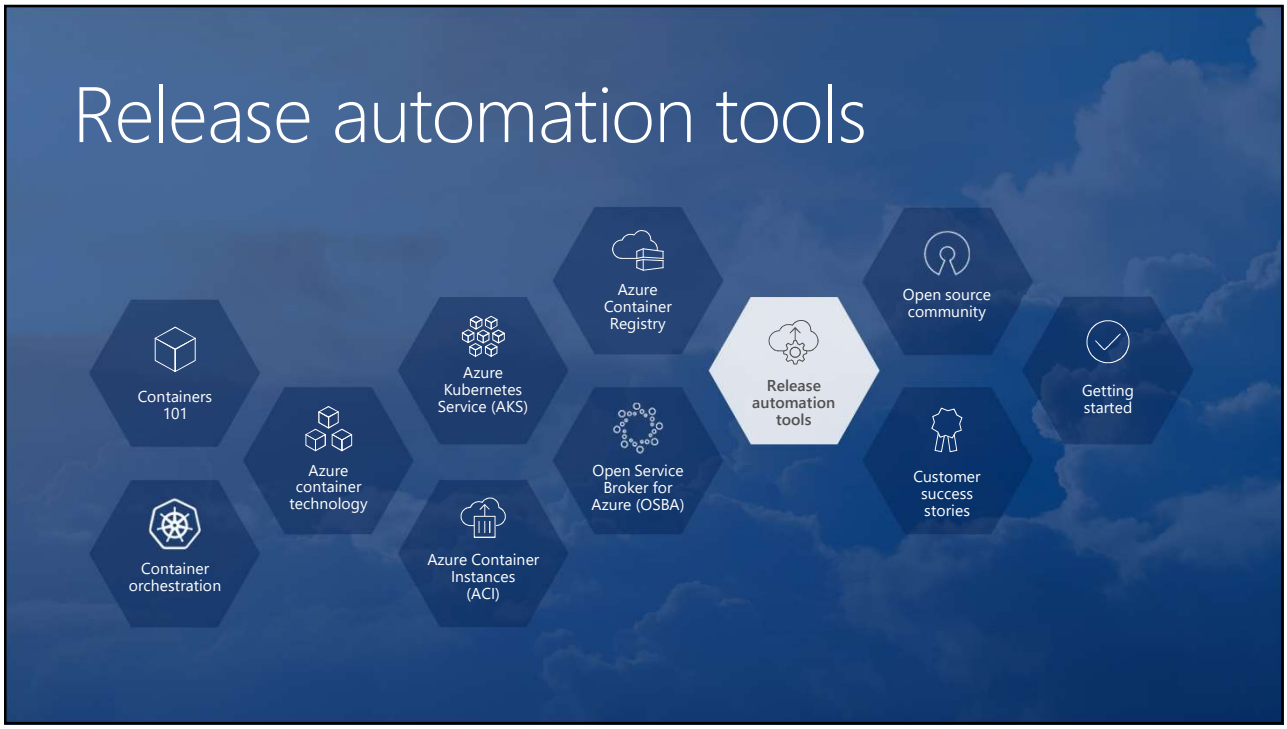
Upgrading Kubernetes in AKS

Azure CLI | Copy | Try It

```
az aks get-versions --name myK8sCluster --resource-group myResourceGroup --output table
```

Azure CLI | Copy | Try It

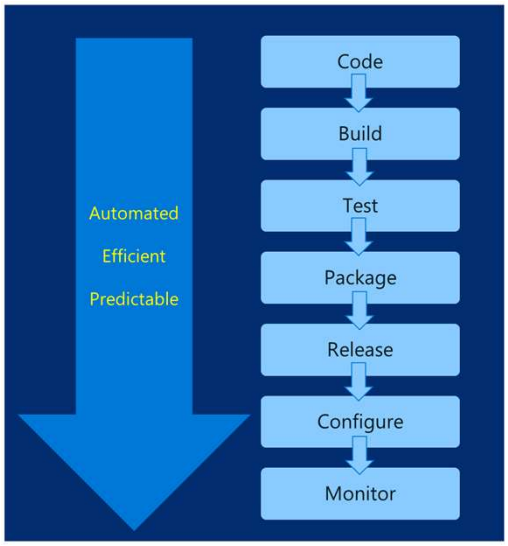
```
az aks upgrade --name myK8sCluster --resource-group myResourceGroup --kubernetes-version 1.8.1
```



DevOps Practices Arrive

- Developers: Test, Build, Code, Plan
- Operations: Monitor, Release, Deploy, Operate
- DevOps Features
  - Speed of application delivery/updates
  - Faster time to value
  - Repeatable/consistent
  - Automated testing
  - Traceability of process
  - Applying developer patterns to infrastructure
    - Infrastructure as code
    - Ops teams embracing source control
  - Centralized monitoring, logging, debugging
- CI / CD – key aspect of quality DevOps

DevOps – Why?



Deploy 200 times more frequently

Go from code check-in to production 2,555 times faster

Recover from failure 24 times faster

Spent 50% less time remediating security challenges

Spent 22% less time on unplanned work

2.2 times more likely to believe their places a great place to work

Source: https://puppet.com/resources/white-paper/2016-state-of-devops-report

Common Toolsets

- Jenkins
- Visual Studio Team Services
- Spinnaker and Netflix OSS
- Travis
- TeamCity
- CircleCI
- Additional utilities: code quality scanning, security, collaboration, etc.

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Release automation tools

Simplifying the Kubernetes experience

DRAFT	HELM	BRIGADE	KASHTI
Streamlined Kubernetes development	The package manager for Kubernetes	Event-driven scripting for Kubernetes	Visualization dashboard for Brigade

Azure Kubernetes Service (AKS)

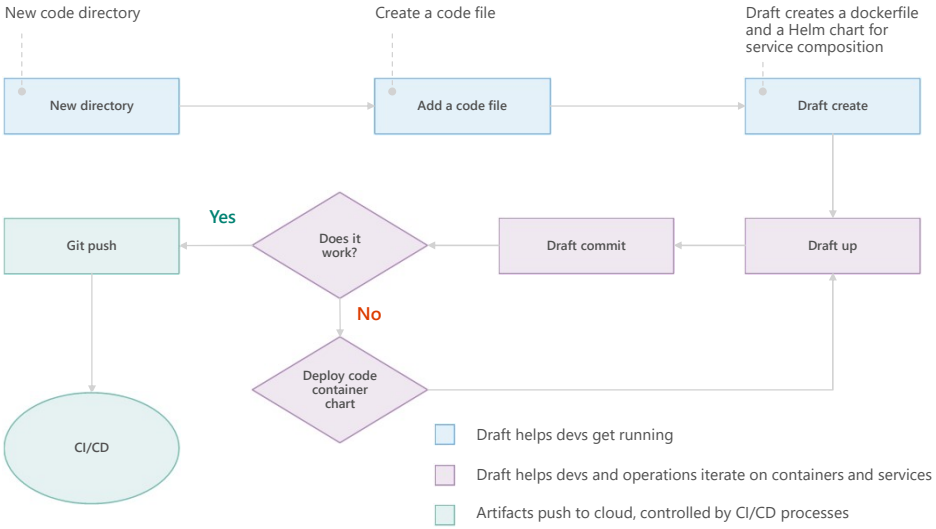
Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Release automation workflow



Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

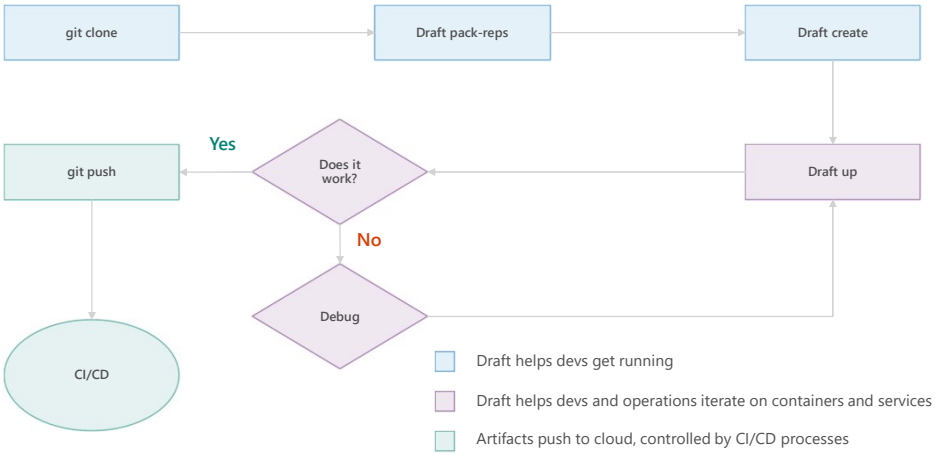
Azure Container Registry

Open Service Broker API (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



Azure Kubernetes Service (AKS)

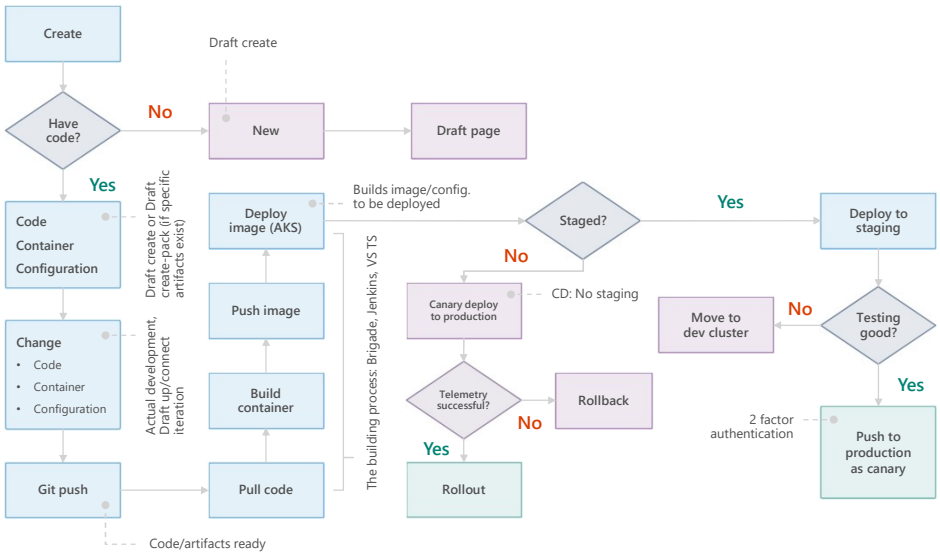
Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Release automation workflow



Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (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

Azure Container Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Draft

Draft in action

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (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 rollout` to roll back to an older version of a release with ease

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

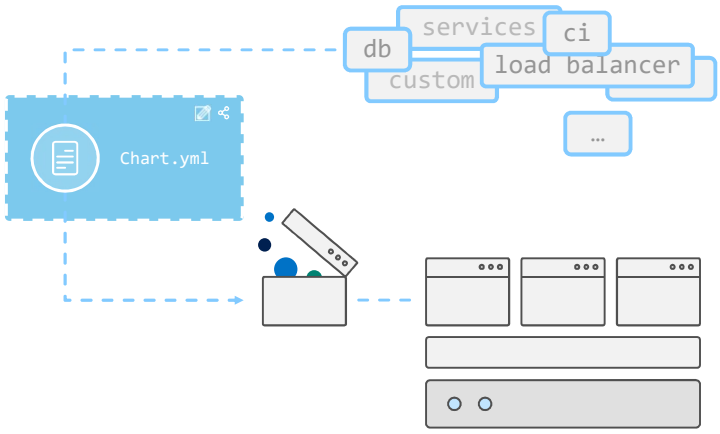
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Helm

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



Helm Charts

Application definition

Consists of:

- Metadata
- Kubernetes resource definitions
- Configuration
- Documentation

Stored in chart repository

- Any HTTP server that can house YAML/tar files (Azure, GitHub pages, etc.)
- Public repo with community supported charts (eg – Jenkins, Mongo, etc.)

Helm (CLI) + Tiller (sever side)

Release: Instance of chart + values -> Kubernetes

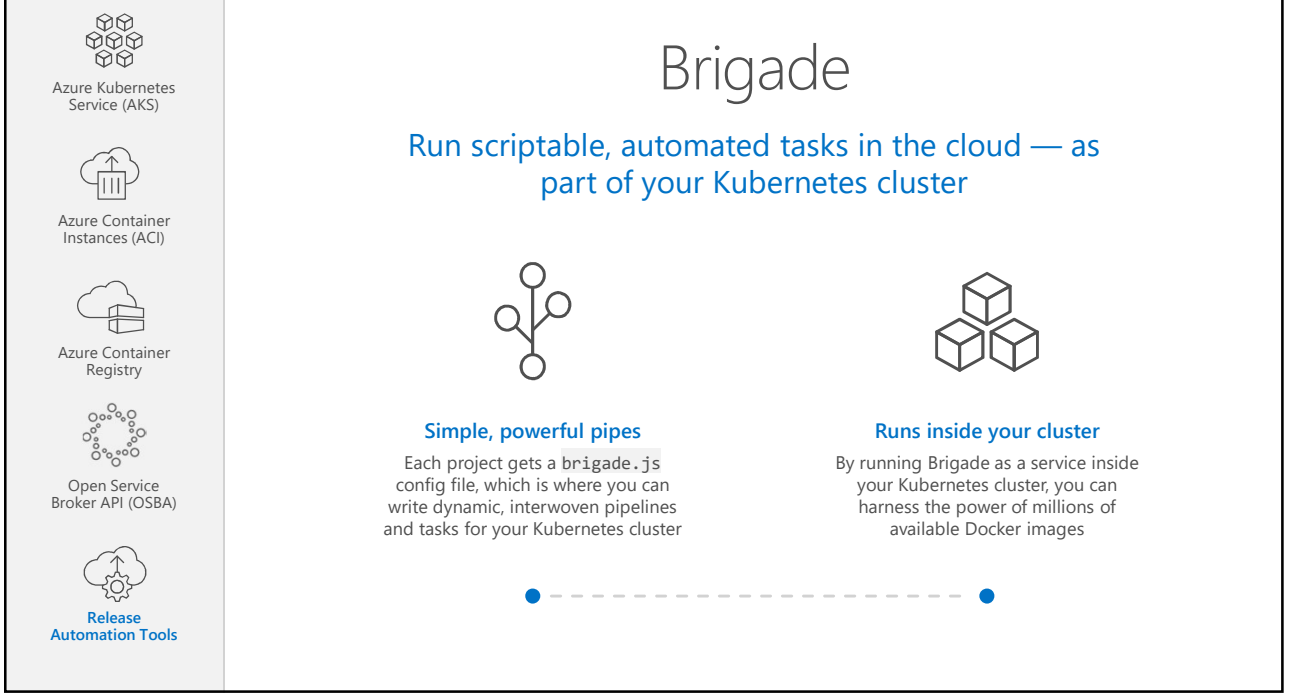
Chart structure

- Layout
  - Helm expects a strict chart structure

```
wordpress/
  Chart.yaml          # A YAML file containing information about the chart
  LICENSE             # OPTIONAL: A plain text file containing the license for the chart
  README.md           # OPTIONAL: A human-readable README file
  values.yaml         # The default configuration values for this chart
  charts/             # OPTIONAL: A directory containing any charts upon which this chart depends.
  templates/          # OPTIONAL: A directory of templates that, when combined with values,
                      # will generate valid Kubernetes manifest files.
  templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes
```

Helm values.yaml

- The knobs and dials:
  - A `values.yaml` file provided with the chart that contains **default** values
  - Use -f to provide your own values overrides
  - Use --set to override individual values



Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

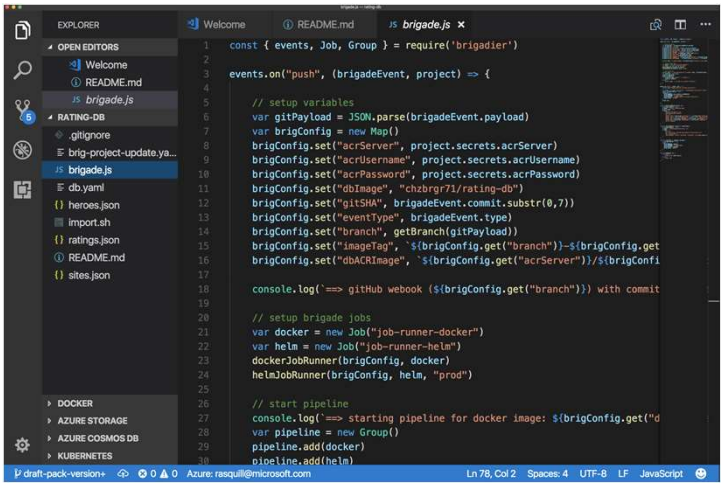
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Brigade

Brigade in action



Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

Kashti

A simple UI to display build results and logs

Simple visualizations

A web dashboard for Brigade, helping to easily visualize and inspect your Brigade builds

Driving deep insights

Make Brigade DevOps workflows—projects, scripts, and jobs—and their events visible instantly

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

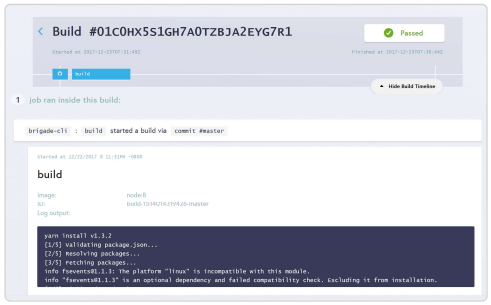
Azure Container Registry

Open Service Broker API (OSBA)

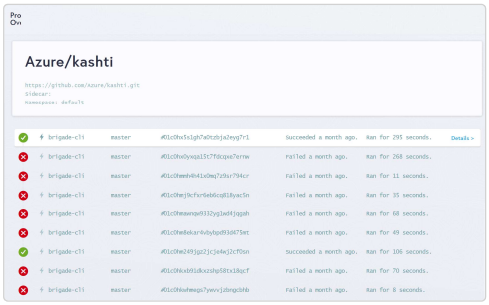
Release Automation Tools

Kashti

Dashboards for Brigade pipelines



Builds dashboard



Events log

Questions?

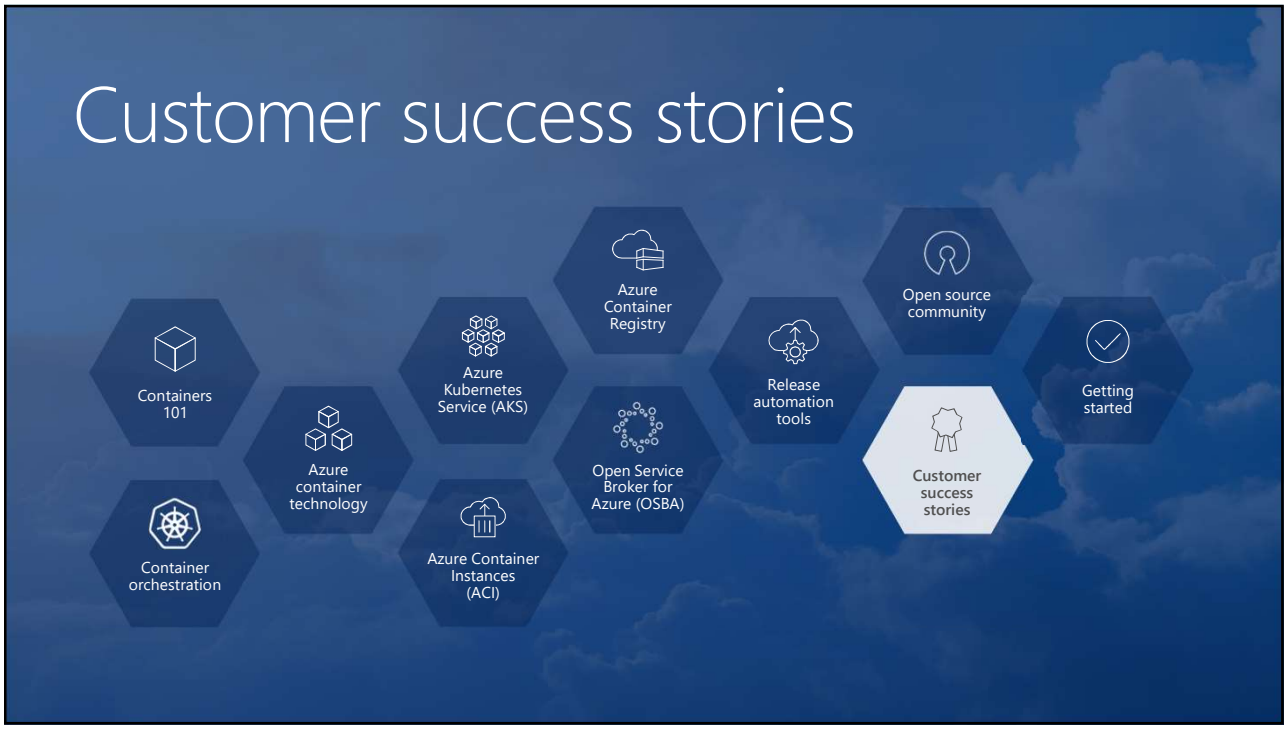
Community culture

Open source container code contributions

Numerous open source project builds

Open source community leadership

Customer success stories



Energy company electrifies pace of innovation and expansion

Ambit Energy provides electricity and natural-gas services in deregulated markets around the world. It uses technology as a competitive differentiator, employing microservices, DevOps, and continuous deployment to speed software development. To stand up infrastructure just as quickly, Ambit uses Microsoft Azure services such as Azure Container Service, together with infrastructure as code and open source technologies, to completely automate infrastructure provisioning. By implementing Azure, Ambit can move dramatically faster to enhance its services and enter new markets. Infrastructure redundancy is flexible and worry-free. And costs are 22 percent lower, which helps Ambit compete in the crowded electricity market. Because Ambit's cloud journey is gradual, it appreciates the fact that Azure is a great hybrid-cloud enabler, connecting easily to Ambit datacenters.

AMBIT ENERGY

Products and services	Organization size	Industry	Country	Business need
Microsoft Azure Container Service	1,000 employees	Power and utilities	United States	Optimize operational efficiency

Siemens Health leverages technology to connect medical devices to the cloud through AKS

Digitization and networking between healthcare providers and software development companies are essential to value-based care. Moving from the development of value-added services into becoming more of a platform provider, it became important for Siemens to adopt a microservices approach to application delivery. To that end, Siemens adopted Azure Container Service (AKS) to run their microservices-based apps. AKS puts Siemens in a position not only to deploy business logic in Docker containers—including the orchestration—but also enables them to use an applicant gateway and API management to manage exposure, control, and to meter the access continuously. With their cloud-based development approach, Siemens has driven newfound product development agility. This project is already having a positive impact within the healthcare industry.

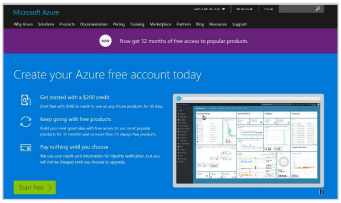
SIEMENS

Products and services	Organization size	Industry	Country	Business need
Microsoft Azure Container Service	100,000+ employees	Healthcare	Germany	Faster application development

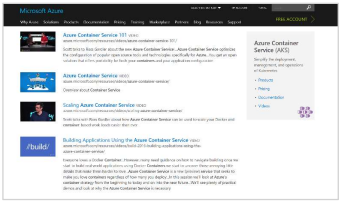
Azure container **resources**

- Azure Kubernetes Service (AKS)
- Azure Container Instances (ACI)
- Azure Container Registry
- OSBA announcement blog
- Draft webpage
- Helm webpage
- Brigade webpage
- Kashti announcement blog

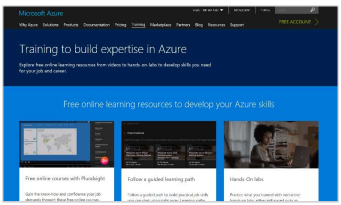
Sign up for a free Azure account



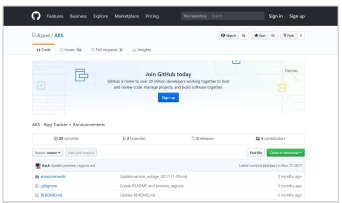
Check out the Azure container videos page



Hone your skills with Azure training



Get the code from GitHub



Appendix Sections

So many choices…

Service	Sweet spot
Azure Web Apps	Deploy scalable web apps and services using containers
Service Fabric	Build microservice applications packaged in containers
Azure Kubernetes Service	Use OSS orchestrators to manage containers across VMs
Azure Batch	Schedule large scale batch processes deployed in containers
Azure Container Instances	Run individual containers with no VM management

Why Change?

Aging infrastructure

- Aging hardware, operating systems, and business applications in the datacenter can impact:
- Operational costs, efficiency, and reliability
- Capital expenditure requirements
- Security, audit, and regulatory compliance

Lack of agility

- Deployment time of new services
- Operation is time (and budget) consuming
- Innovation is happening outside IT inside business areas

High Cost

- Longer release cycles, monolithic and highly coupled architecture
- Highly IT dependent
- Low application performance and time-to-market compromise business agility

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

Docker, Docker, Docker

Containers have been around for many years
Linux kernel: cgroups, namespaces

Docker Inc. did not invent them
They created open source software to build and manage containers

Docker makes containers easy
Super easy. Fast learning curve

Docker is a container format and a set of tools
Docker CLI, Docker Engine, Docker Swarm, Docker Compose, Docker Machine

What is a container?

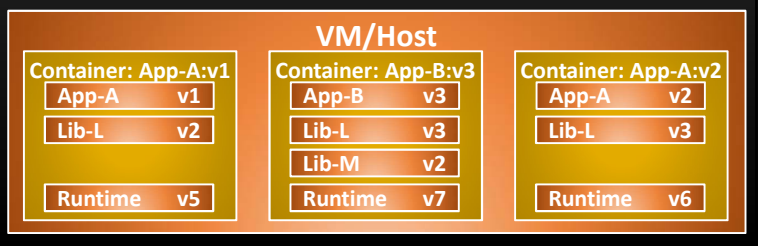
Slice up the OS to run multiple apps on a single VM

Every container has an isolated view and gets
- it's own file system
- it's own PID0 and eth0 network interface

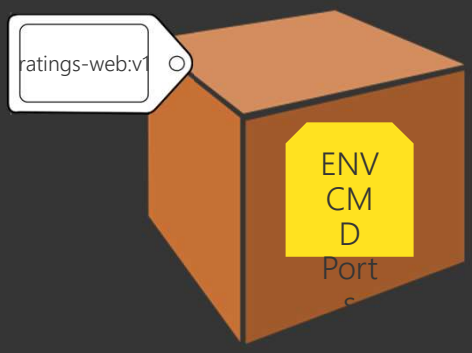
Container and apps share lifecycle

Shared kernel, very fast start-up, and repeatable execution

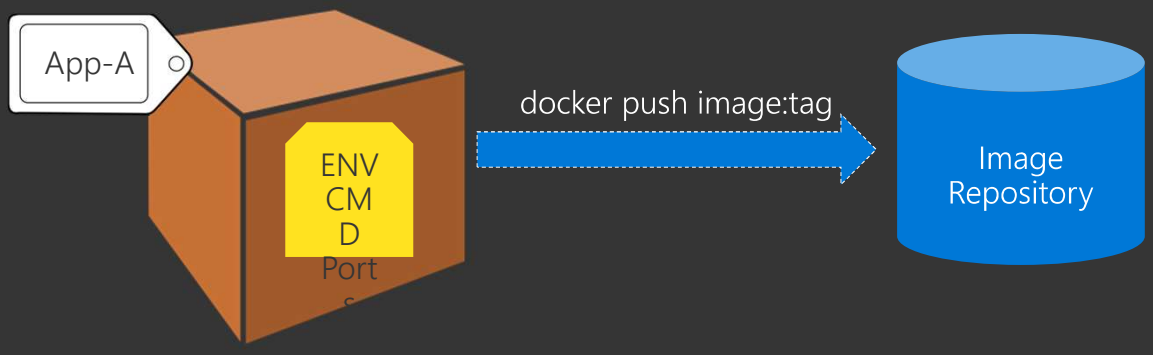
Cannot mix OS



Package Container



Push Container Image



Azure Container Support

Pivotal Cloud Foundry

RED HAT OPENSHIFT Container Platform

Mesos DC/OS

kubernetes

docker

Azure Container Instance

Azure Kubernetes Service (AKS)

Service Fabric

Web Apps

Batch

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

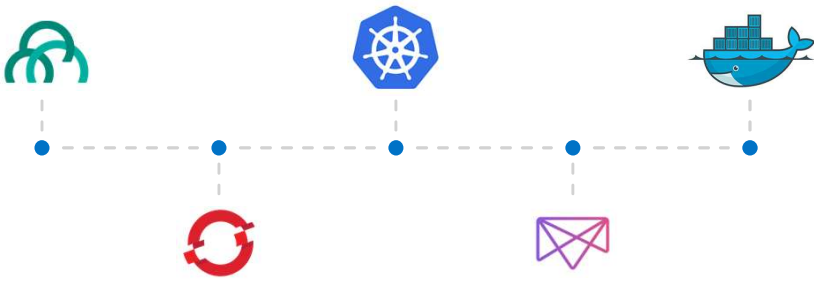
Azure Container Registry

Open Service Broker API (OSBA)

Release Automation Tools

For customers with a preferred container platform

Pivotal Cloud Foundry · Kubernetes · Docker Enterprise Edition
Red Hat OpenShift · Mesosphere DC/OS



Help them bring that platform to Azure

Azure Kubernetes Service (AKS)

Azure Container Instances (ACI)

Azure Container Registry

Open Service Broker API (OSBA)

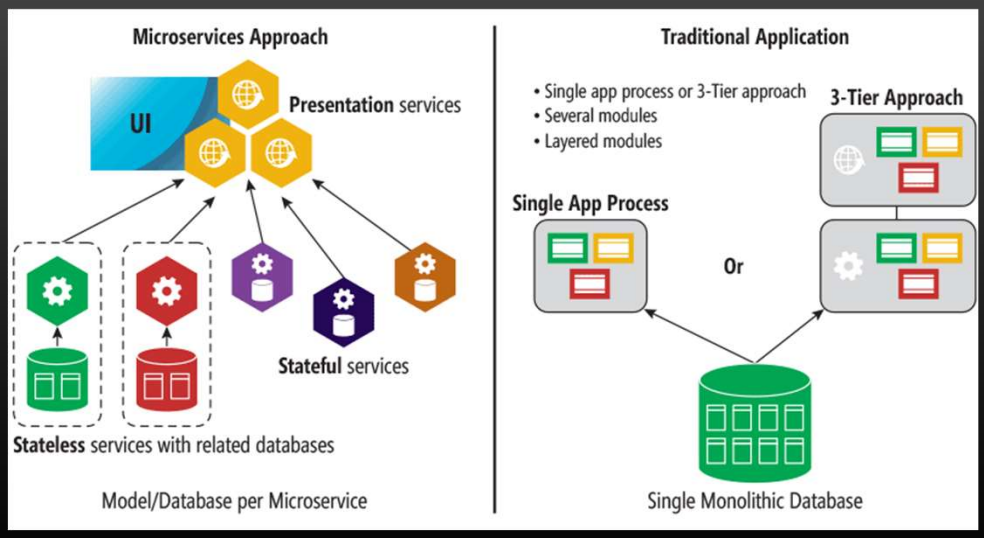
Release Automation Tools

For customers without a preferred container platform…

Lead with profiling questions, then make recommendations based on the customer profiles

For more details, refer to the container cheat sheet

Microservices



microservices ≠ containers

microservices is an architectural design approach

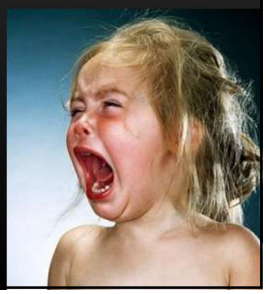
containers are an implementation detail that often helps

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

ConfigMaps and Secrets

- Flexible configuration model for Kubernetes
- Both Secrets and ConfigMaps can be used as ENV vars or files
- Secrets
  - Use Secrets for things which are actually secret like API keys, credentials, etc
  - Secret values are base64 encoded and automatically decoded for pods
- ConfigMaps
  - ConfigMaps for configuration that doesn't have sensitive information. However, there are pros/cons with each
  - ConfigMaps designed to more conveniently support working with strings

Namespaces

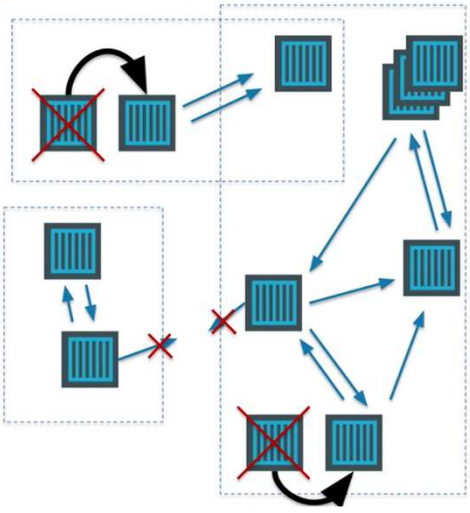
- Allow for multiple virtual clusters backed by the same physical cluster
- Logical separation
- Namespace used in FQDN of Kubernetes services
  - Eg - <service-name>.<namespace-name>.svc.cluster.local
- Every Kubernetes resource type is scoped to a namespace (except for nodes, persistentVolumes, etc.)
- Intended for environments with many users, teams, projects

Role Based Access Control

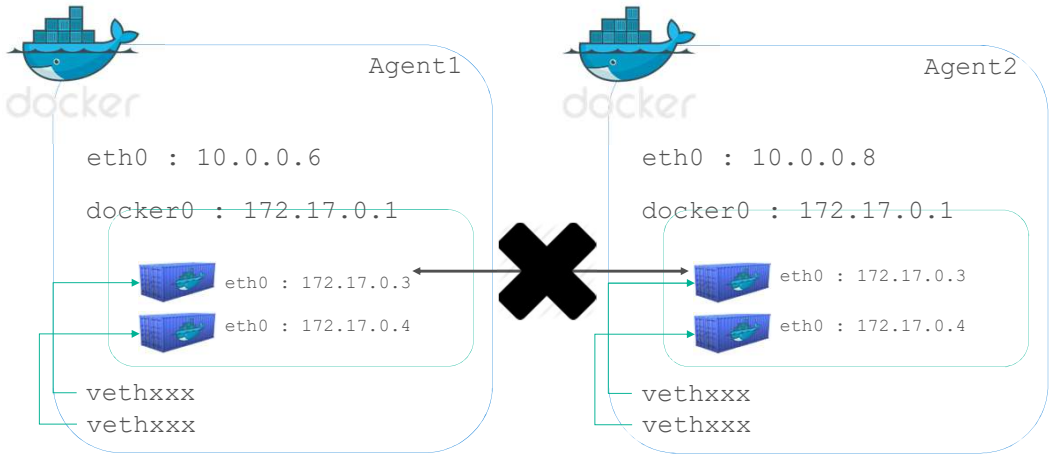
- Allows for dynamic policies against k8s API for authZ
- In version 1.6, this is a beta feature
- Can create both "Roles" and "ClusterRoles"
- Can configure api-server and kubectl to use Azure AD for access
- To grant permissions:
  - Use "RoleBindings" for within namespaces
  - Use "ClusterRoleBindings" for cluster-wide access
  - Can contain users, groups, service accounts
  - Control access to specific resources or API verbs

```
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["batch", "extensions"]
  resources: ["jobs"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

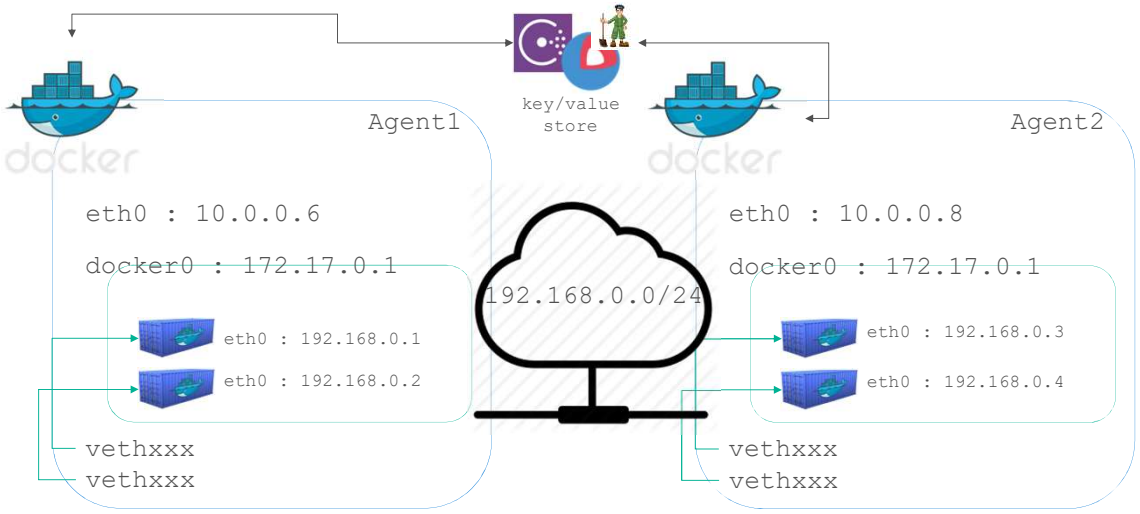
Networking in the Container World



Multi-Host Networking



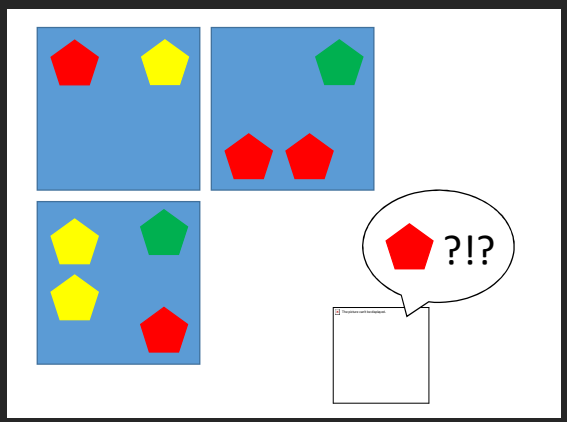
Multi-Host Networking



What is service discovery?

Problem: How service can find and communicate with another one running into the same cluster?

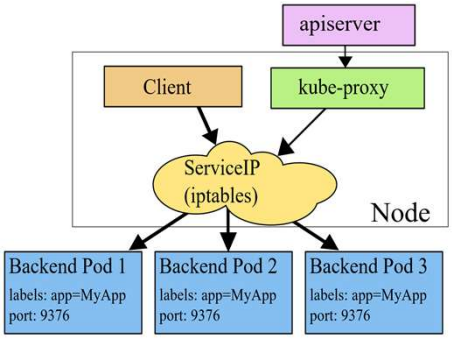
- Service Registry/Naming Service
- Service Announcement
- Lookup/Discovery
- Load Balancing



Networking in Kubernetes

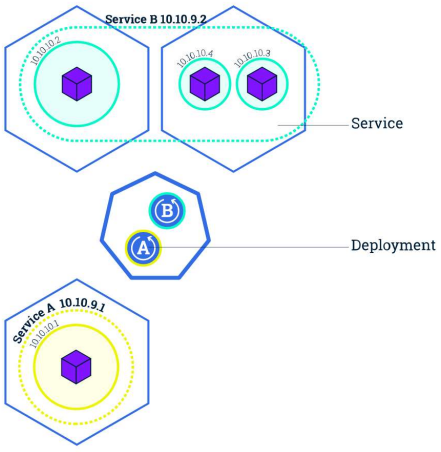
Kubernetes Networking model introduces 3 methods of communications:

- Pod-to-Pod communication directly by IP address. Kubernetes has a Pod-IP wide metric simplifying communication.
- Pod-to-Service Communication – Client traffic is directed to service virtual IP by iptables rules that are modified by the kub-proxy process (running on all hosts) and directed to the correct Pod.
- External-to-Internal Communication – external access is captured by an external load balancer which targets nodes in a cluster. The Pod-to-Service flow stays the same.



Service Discovery with Kubernetes

- Nodes, Deployments, Pods, Services
- Services
  - Expose the services
  - Works with platform Load Balancers to get public IPs
  - Internal load balancing between pods
- kube-dns (preferred)
  - Name service enabled in Kubernetes (Default in ACS)
- Environment Variables
  - Every Service is assigned environment variables with information of the service
  - REDIS_MASTER_SERVICE_HOST=10.0.0.11
    REDIS_MASTER_SERVICE_PORT=6379
    REDIS_MASTER_PORT=tcp://10.0.0.11:6379
    REDIS_MASTER_PORT_6379_TCP=tcp://10.0.0.11:6379
    REDIS_MASTER_PORT_6379_TCP_PROTO=tcp
    REDIS_MASTER_PORT_6379_TCP_PORT=6379
    REDIS_MASTER_PORT_6379_TCP_ADDR=10.0.0.11



Ingresses

- Typically, services and pods have IPs only routable by the cluster network
- All traffic that ends up at an edge router is either dropped or forwarded elsewhere
- Ingress is a collection of rules that allow inbound connections to reach the cluster services
- An Ingress controller is responsible for fulfilling the Ingress, usually with a loadbalancer, though it may also configure your edge router or additional frontends to help handle the traffic in an HA manner

Container Network Interface (CNI)

- CNI (a CNCF project) is a spec for plugins to configure Linux container network interfaces
- Wide range of support
  - Kubernetes, OpenShift, Cloud Foundry, Mesos, rkt, etc.
- Examples
  - Calico, Weave, Contiv, etc.
- Azure CNI plug-in integrates Azure VNET into kubernetes clusters
- Use acs-engine

Daemonsets

- Ensures that all (or some) nodes run a copy of a pod
- Doesn't care if a node is marked as unschedulable
- Can make pods even if the scheduler is not running

StatefulSets

- Used for workloads that require consistent, persistent hostnames e.g. etcd-01, etcd-02
- One PersistentVolume storage per pod
- StatefulSet example - zookeeper
- StatefulSet example - cockroachdb

Jobs

- Creates one or more pods and ensures that a specified number of them successfully terminate
- 3 types of jobs
  - Non-parallel Jobs - e.g. single pod done when exit is successful
  - Parallel Jobs with a fixed completion count - e.g. one successful pod for each value in the range 1 to .spec.completions
  - Parallel Jobs with a work queue that coordinate with each other and terminate when one pod exits successfully