



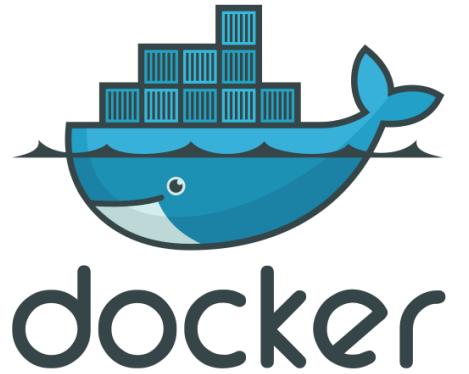
Azure Container Hackfest

*Delivering modern cloud native applications with
open source technologies on Azure*

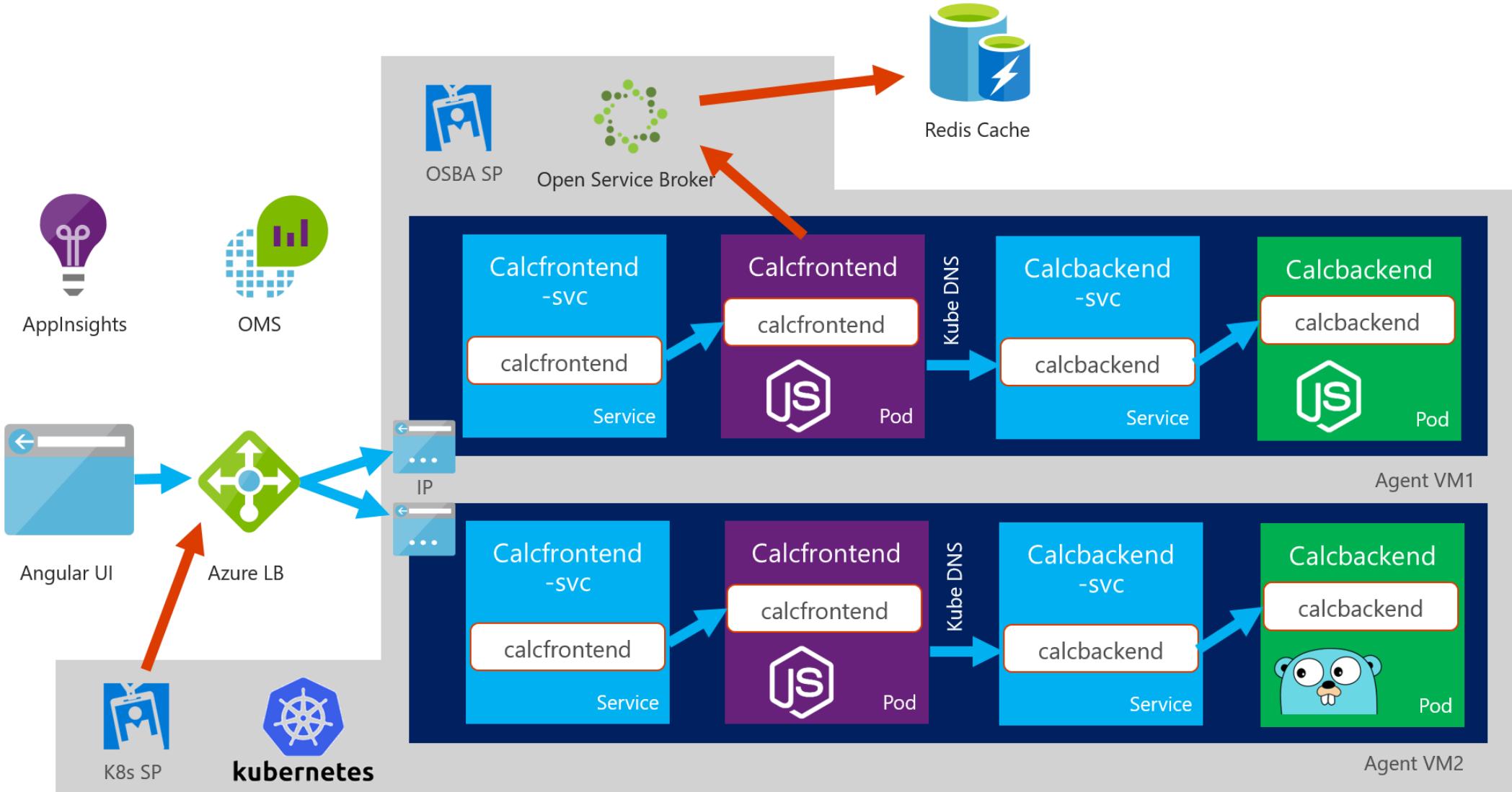


What we are going to learn today...

- Local container development / Docker 101
- Single container applications (on Azure)
- Multi container applications / Orchestration with Kubernetes (AKS)
- DevOps: Continuous Integration & Continuous Deployment with VSTS
- Networking / Ingress / Monitoring
- Application Packaging / Helm
- Open Service Broker for Azure



What we are going to build today...



Where to find everything...?

<https://github.com/CSA-OCP-GER/phoenix>

Screenshot of the GitHub repository page for 'CSA-OCP-GER / phoenix'. The repository has 134 commits, 1 branch, 0 releases, and 6 contributors. It is forked from 'denniszieke/phoenix'. The README.md file contains the following text:

```
Containerize your enterprise - tutorials and resources for learning Kubernetes hands on!
```

The repository has a list of files and their commit history:

- about
- apps
- hints
- img
- .deployment
- .gitignore
- ProjectPhoenix.pdf
- README.md
- about.md
- challenges.0.md
- challenges.1.md
- challenges.2.md
- challenges.3.md

Screenshot of the GitHub repository page for 'CSA-OCP-GER / phoenix' showing the README.md file content. The README.md file contains the following text:

```
Project Phoenix - Containerize your enterprise

Welcome to project Phoenix - a hands on workshop to practice container technology in the enterprise.

What do we want to achieve?

After the workshop you should have hands on experience with:

1. Defining Application and system architectures for containers.
2. Defining, configuring up and maintaining runtime environments for containers in Azure.
3. Configuration options for CI/CD pipeline for containers in Azure.
4. Understanding of Kubernetes objects (Pods, Services, Deployments, Secrets) and their usage for multi container applications.
5. Logging, scaling and monitoring of container runtimes.
```

A diagram illustrating the CI/CD pipeline for the Project Phoenix workshop:

```
graph LR; A[Source Code Control (SCC)] --> B[Build/CI, Integrate, Test]; B --> C[CD, Deploy]; C --> D[Run, Manage, Integrate]; E[Azure Container Registry] --> B; E --> C; E --> D;
```

The diagram shows a flow from Source Code Control (SCC) through Build/CI, Integrate, Test, CD, Deploy, and finally Run, Manage, Integrate. The Azure Container Registry is connected to all stages of the pipeline.

Challenges

- Setup your system
- Single Loop Challenge
- Kubernetes 101
- Kubernetes Multi-Container
- Operations and Application Insights
- Integration

Search or jump to... / Pull requests Issues Marketplace Explore

Unwatch ▾

CSA-OCP-GER / phoenix
forked from denniszielke/phoenix

Code Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master phoenix / challenges.3.md

cdennig Cleanup

2 contributors

83 lines (67 sloc) | 5.3 KB

Raw

Kubernetes Multicontainer Challenge

Need help? Check hints here !

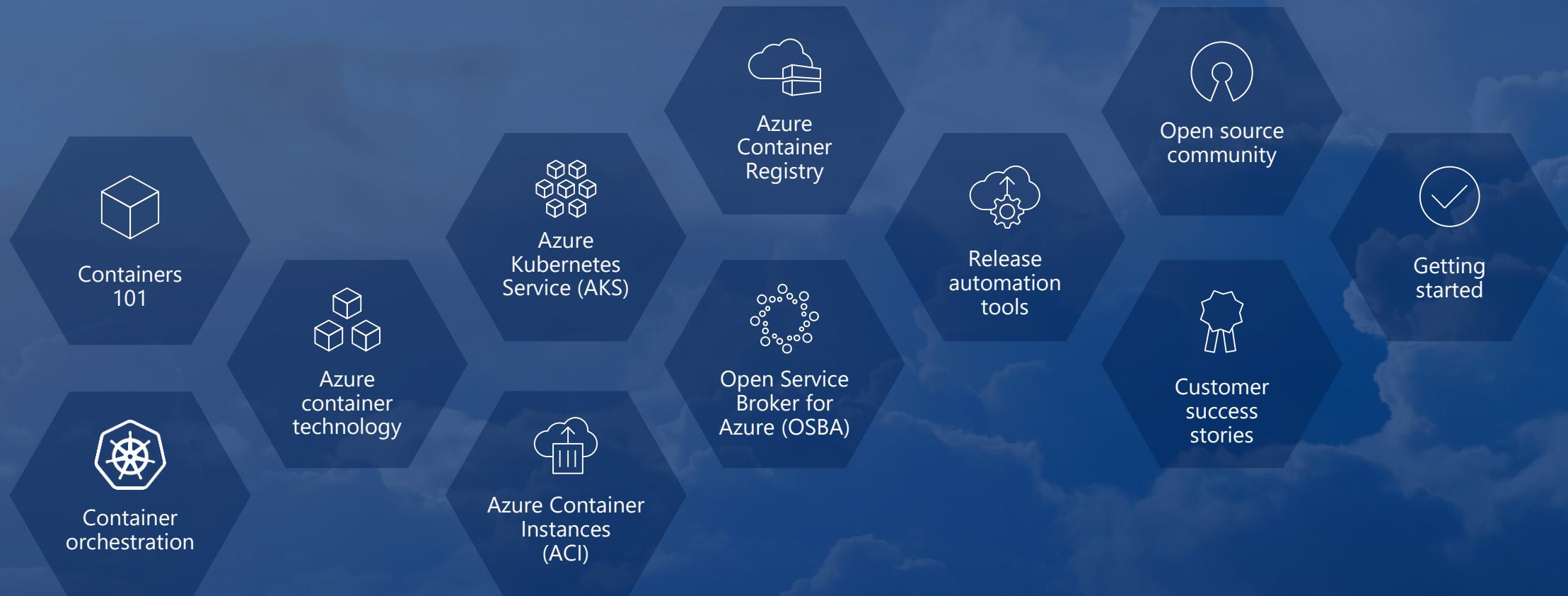
In this chapter you will create a multi-container application in Kubernetes. This is more close to real-life administration a little more challenging. In reality you might want to be able to specify that multiple instances can talk to each other in a defined way. You might want to make sure certain parts of your application have multiple instances to cover load. You might want to be able to monitor performance of your application to make sure that your system is self-healing so that faulty components are replaced automatically. You might want to make sure to have zero downtime of your application. We are going to configure all of this.

Here's what you'll learn:

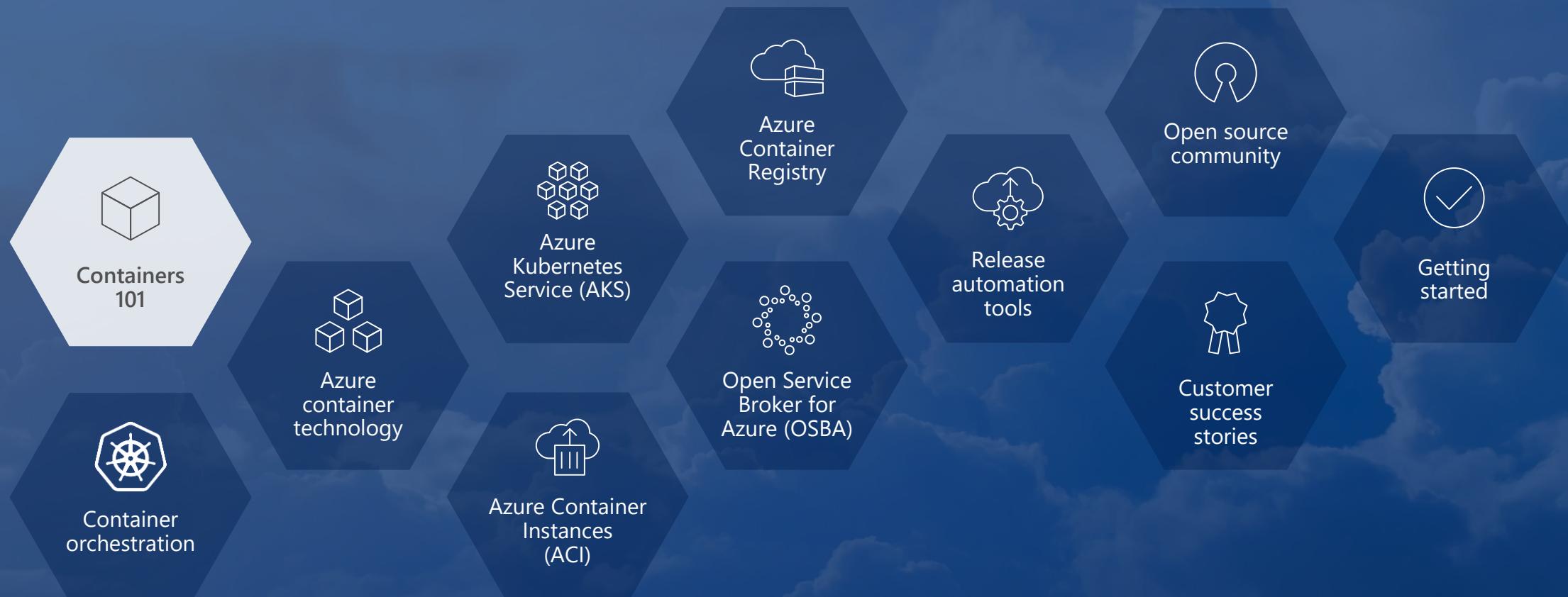
- How to write Yaml files to specify a desired state of a Kubernetes object
- How to use the Azure Portal to view Application performance
- How to store secrets in Kubernetes
- How to configure your Kubernetes instance to ensure a certain number of pods is always running
- How to define rolling upgrades to avoid downtime during an application update

Learn to install what you've learned into an on-premises Docker environment via VSTS

Table of contents



Containers 101



Why should customers care about
containers and microservices?

In reality, they shouldn't...

They do care about cloud native applications



"Unlimited" Scale

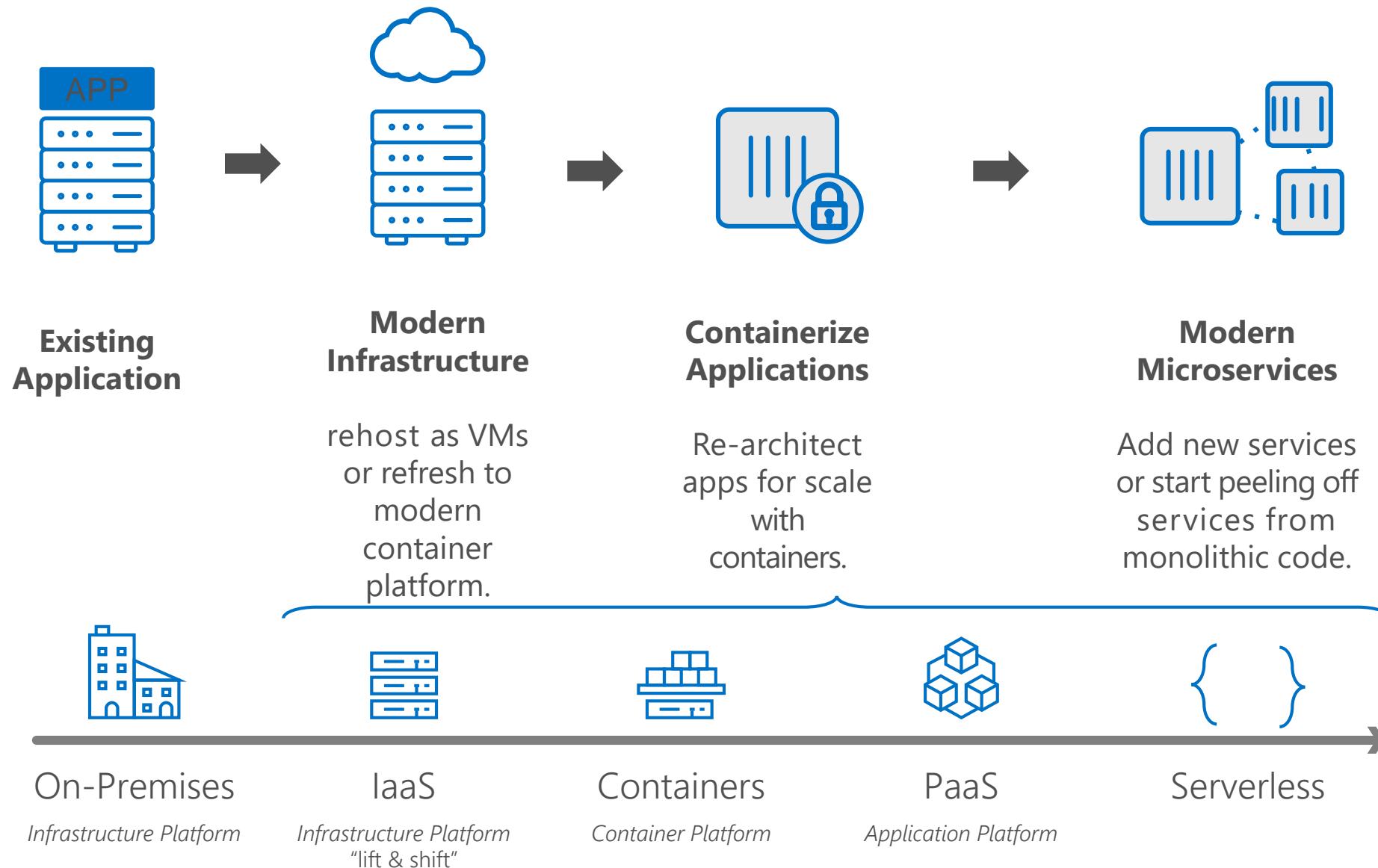


Global reach



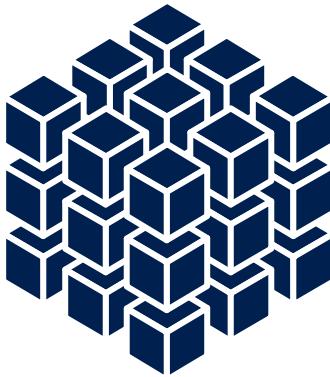
Rapid innovation -
> time to market

From traditional app to modern app



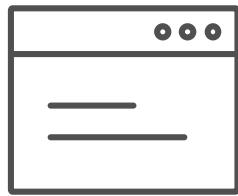
Microservices ≠ Containers

“Microservices” is an **architectural design approach**

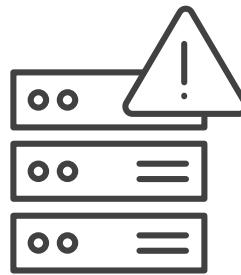


Containers are an **implementation detail** that often helps

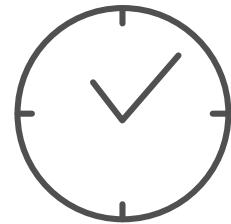
What we hear from **developers**



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



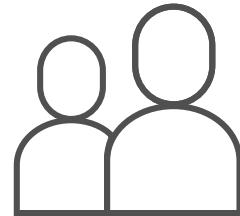
My productivity and application innovation become suspended when I have to wait on IT



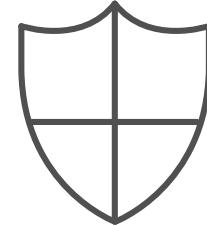
What we hear from **IT**



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

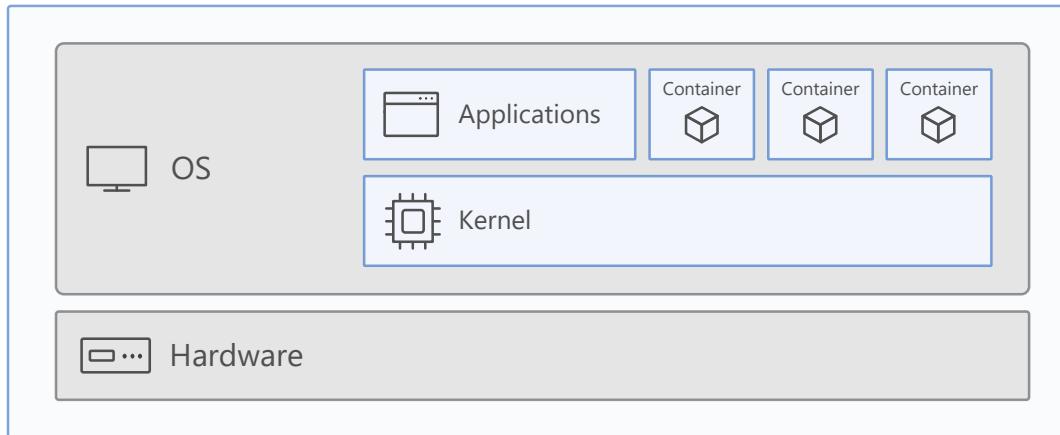


I'm unable to focus on both
server protection and
application compliance

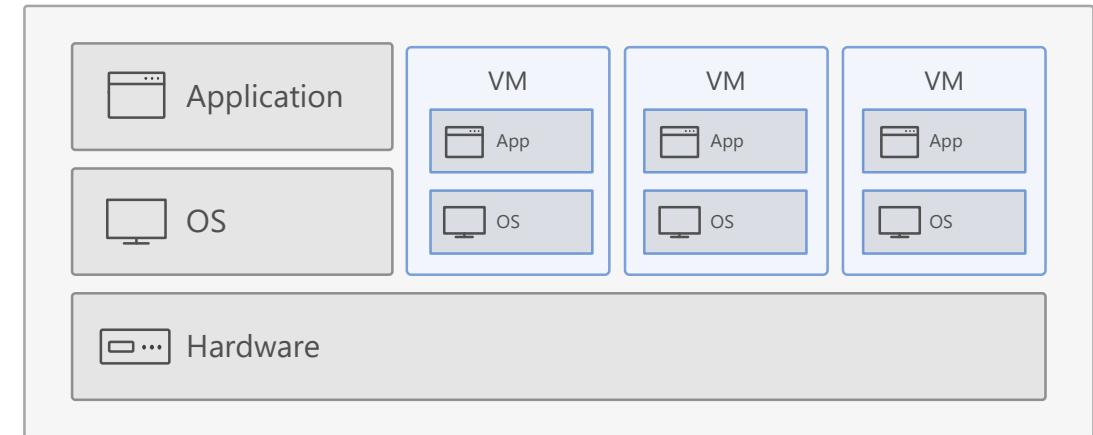


What is a **container**?

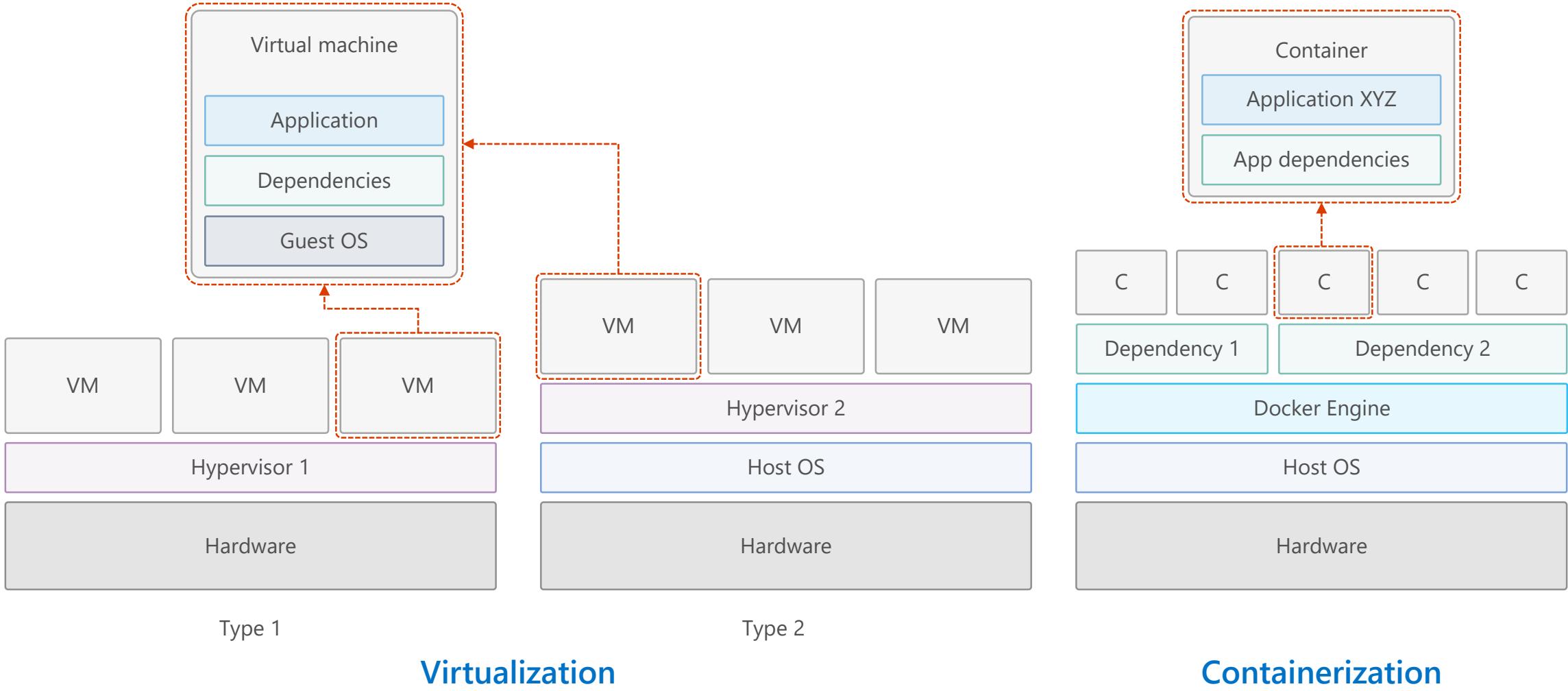
Containers = operating system virtualization



Traditional virtual machines = hardware virtualization



Virtualization versus **containerization**

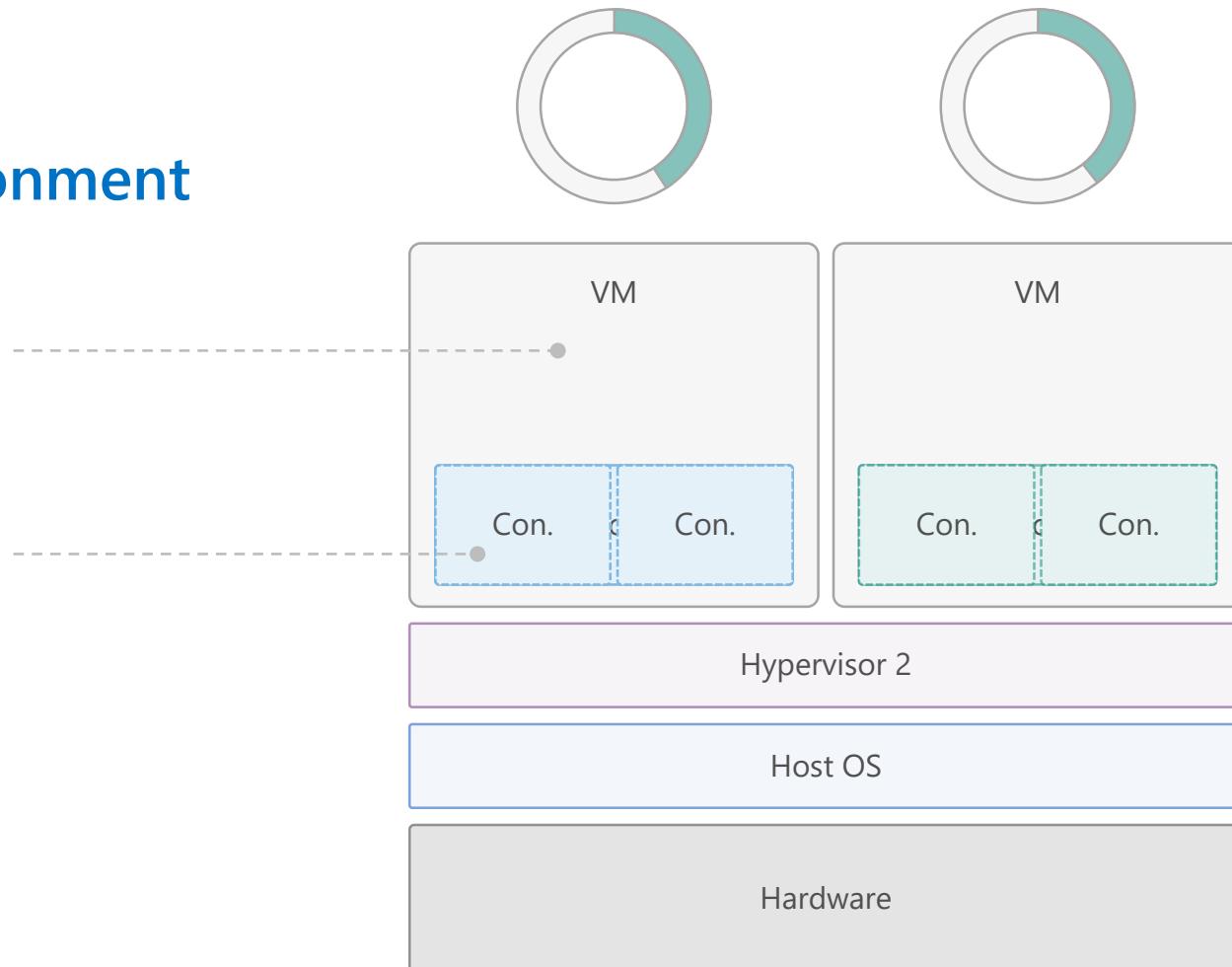


The container **advantage**

Traditional virtualized environment

Low utilization of container resources

Containerization of applications and their dependencies

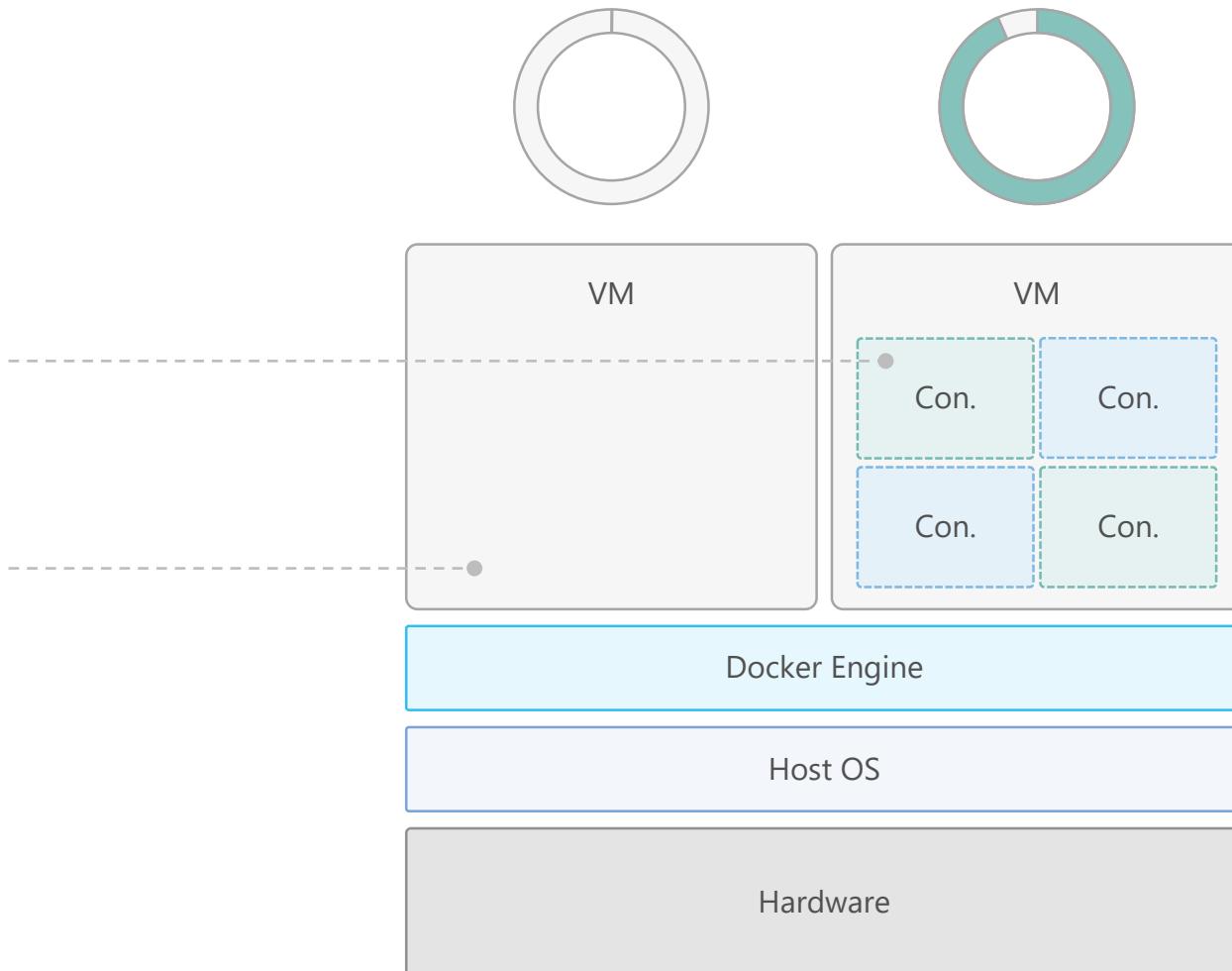


The container **advantage**

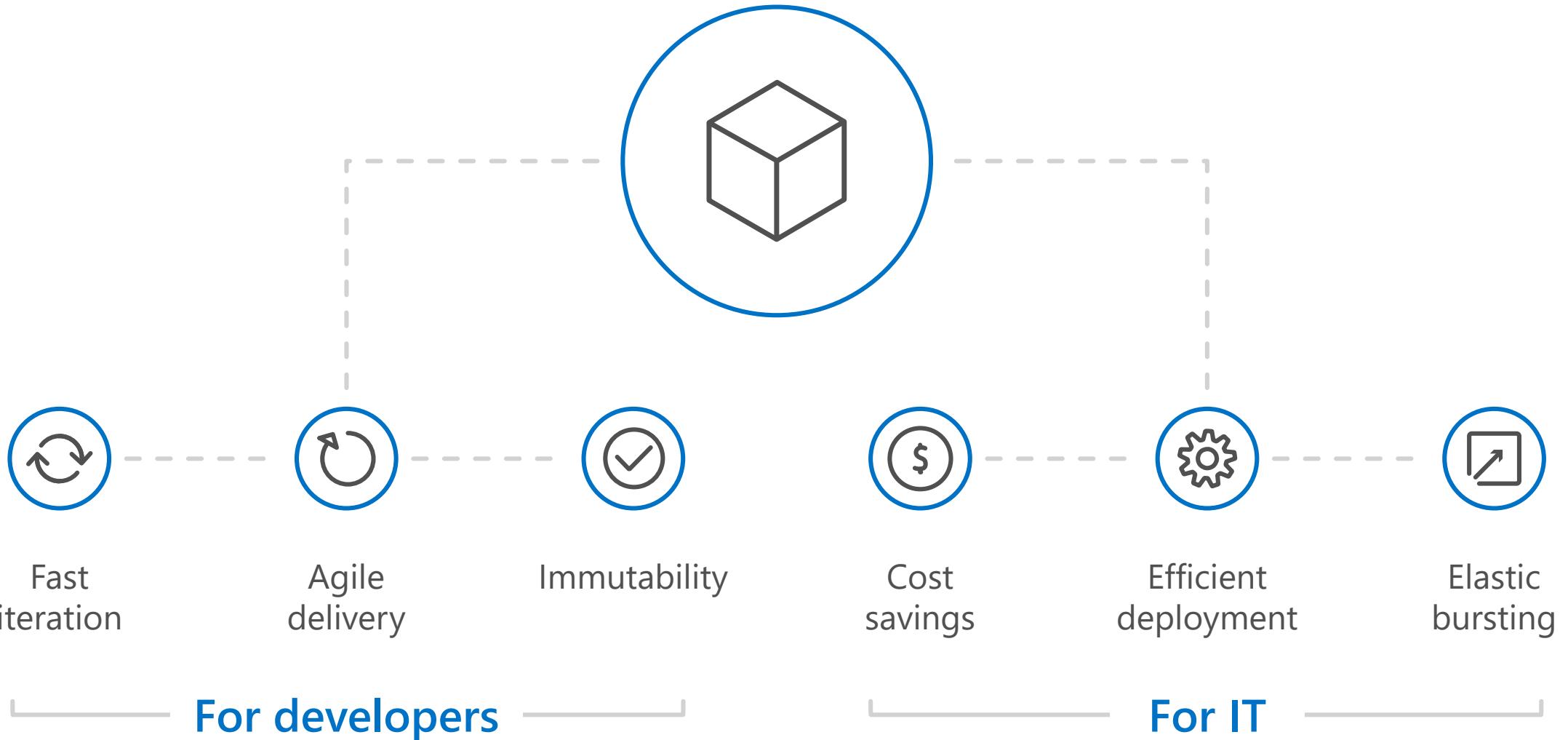
Containerized environment

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

Decommission unused resources for efficiency gains and cost savings



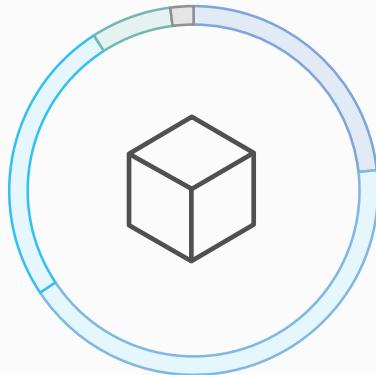
The container **advantage**



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

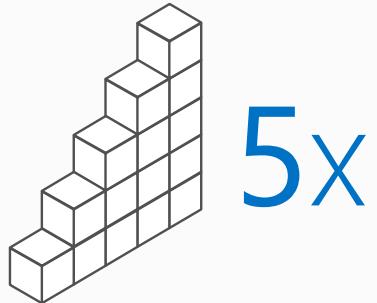


Larger companies are leading adoption.²

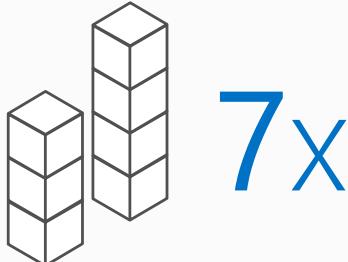
Nearly 60% percent of organizations running 500 or more hosts are classified as container dabblers or adopters.



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



Container hosts often run **SEVEN** containers at a time.¹



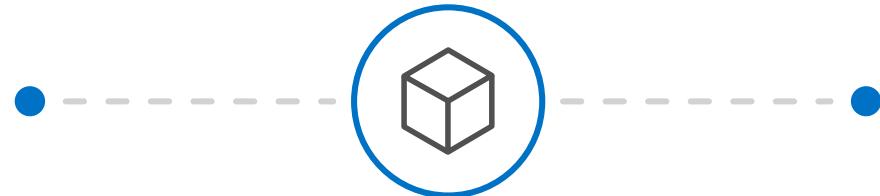
Containers churn 9 times **FASTER** than VMs.¹



Source:

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

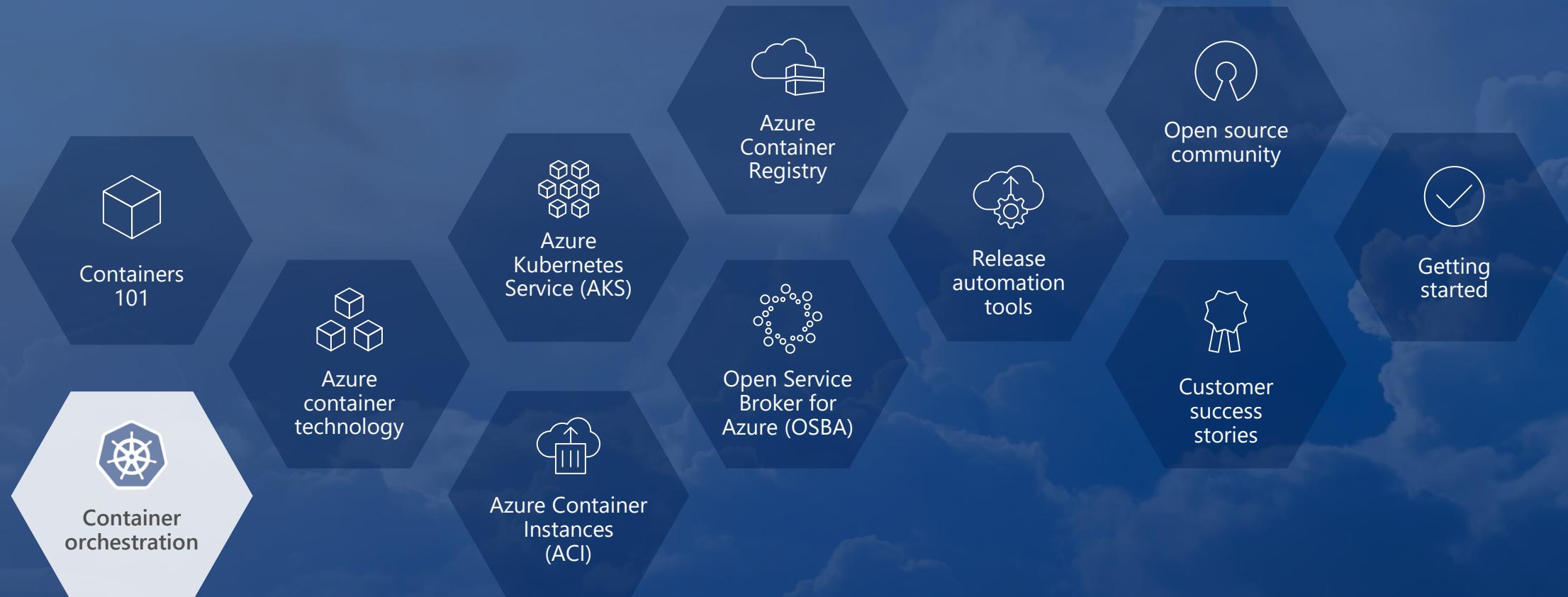
Industry analysts **agree**



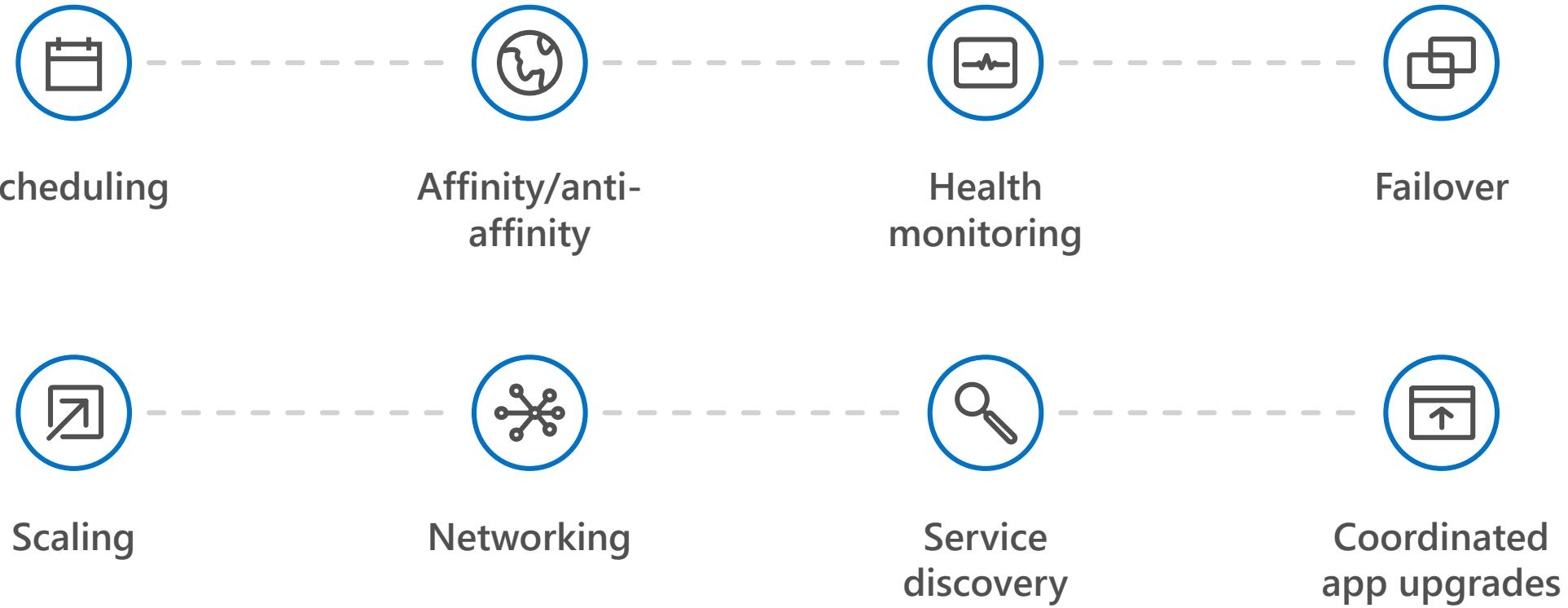
"By 2020, more than 50% of enterprises will run mission-critical, containerized cloud-native applications in production, up from less than 5% today."

Gartner[®]

Container orchestration



The elements of **orchestration**



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

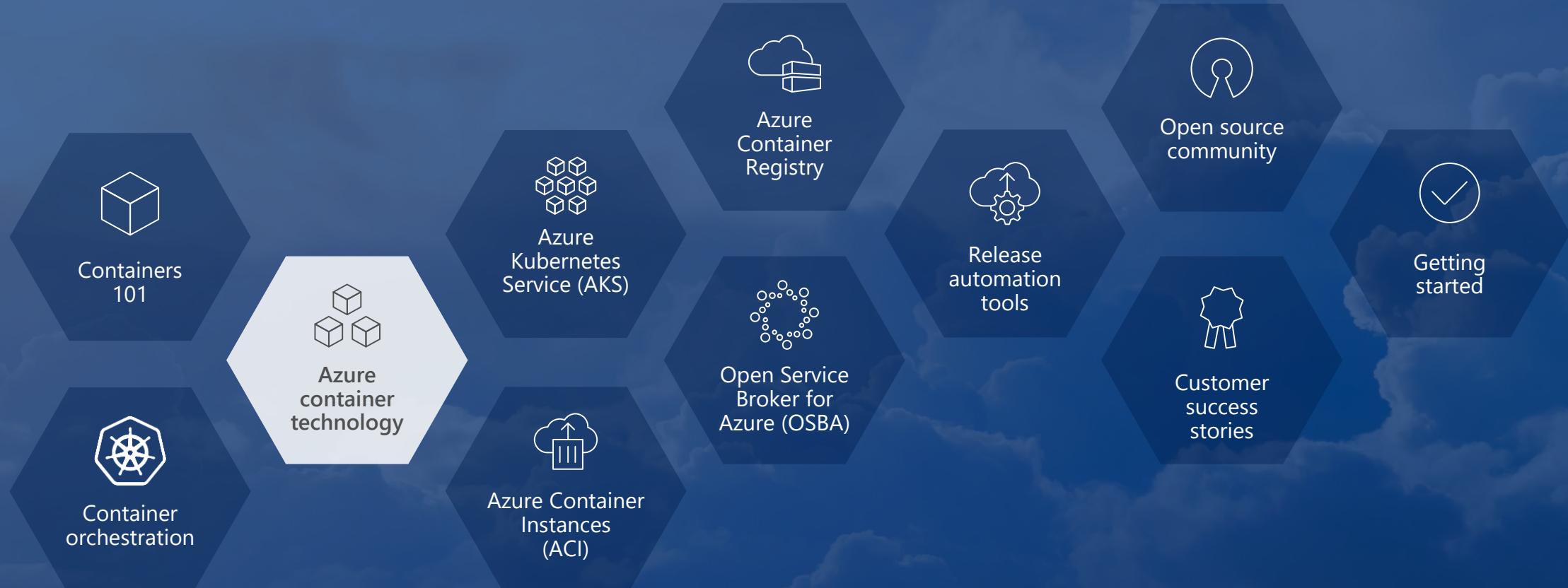
Roll out
new features
seamlessly

Limit hardware
usage to required
resources only



"Distributed apps are sufficiently complicated that they need to be flown by the instruments"

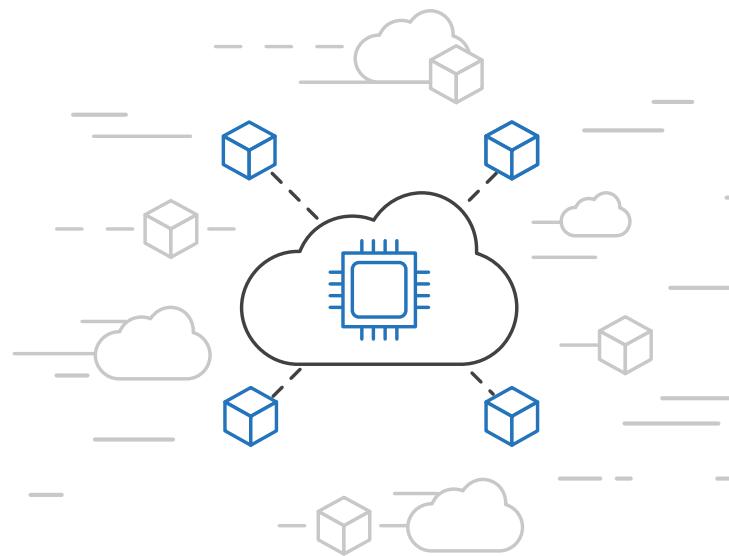
Azure container technology



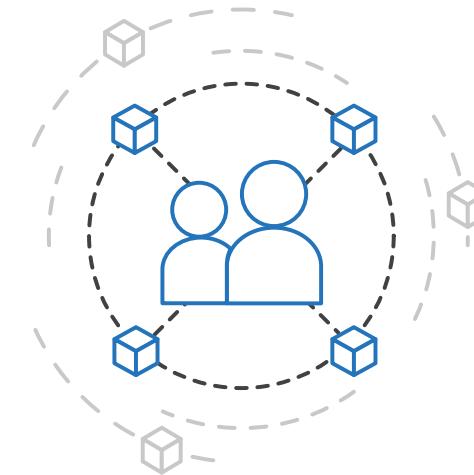
Azure container **strategy**



Embrace containers
as ubiquitous

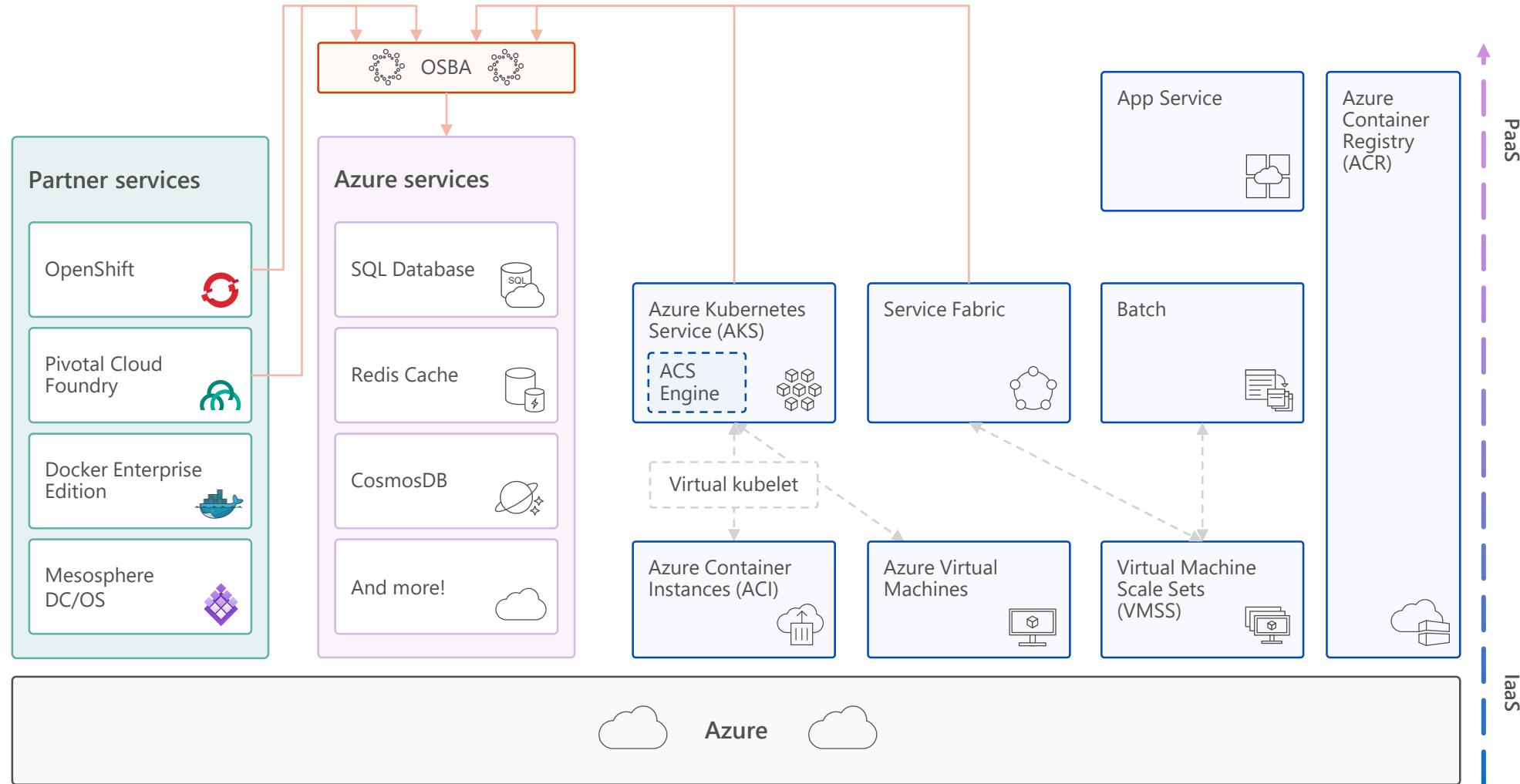


Support containers
across the compute
portfolio

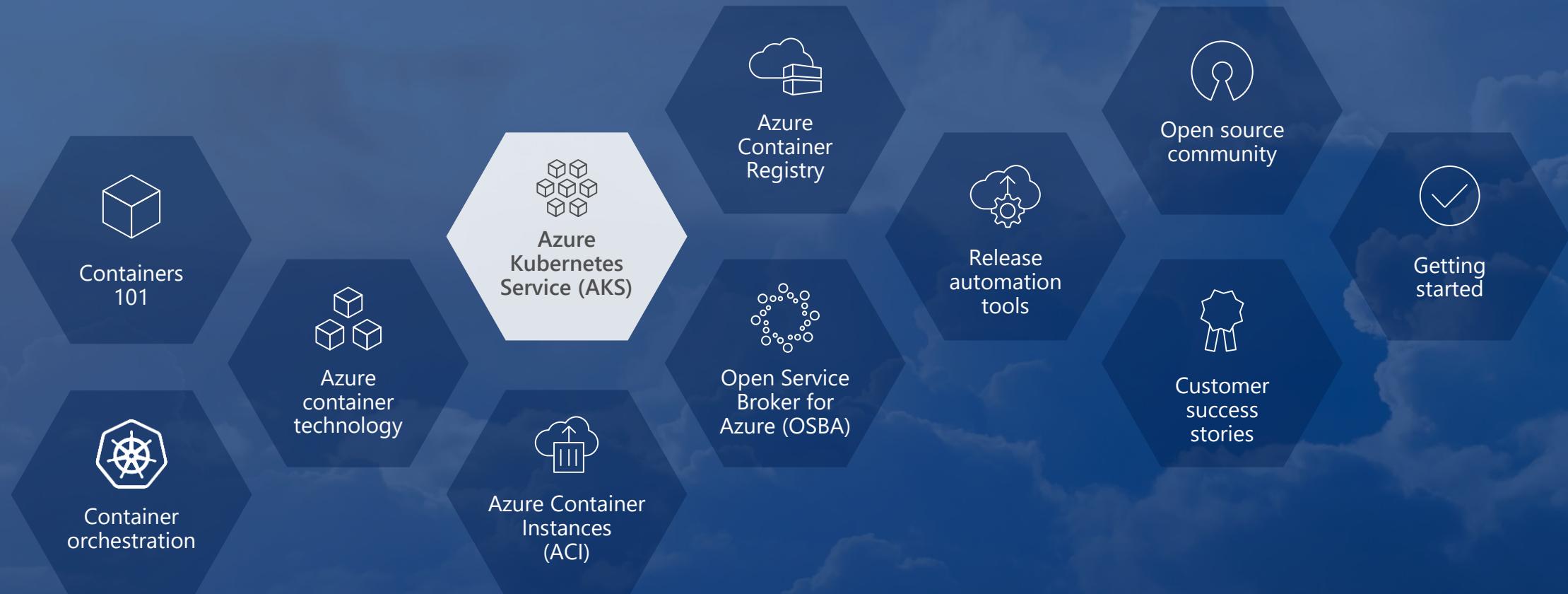


Democratize
container technology

Azure container ecosystem



Azure Kubernetes Service (AKS)





Azure Kubernetes Service (AKS)



Azure Container Instances (ACI)



Azure Container Registry

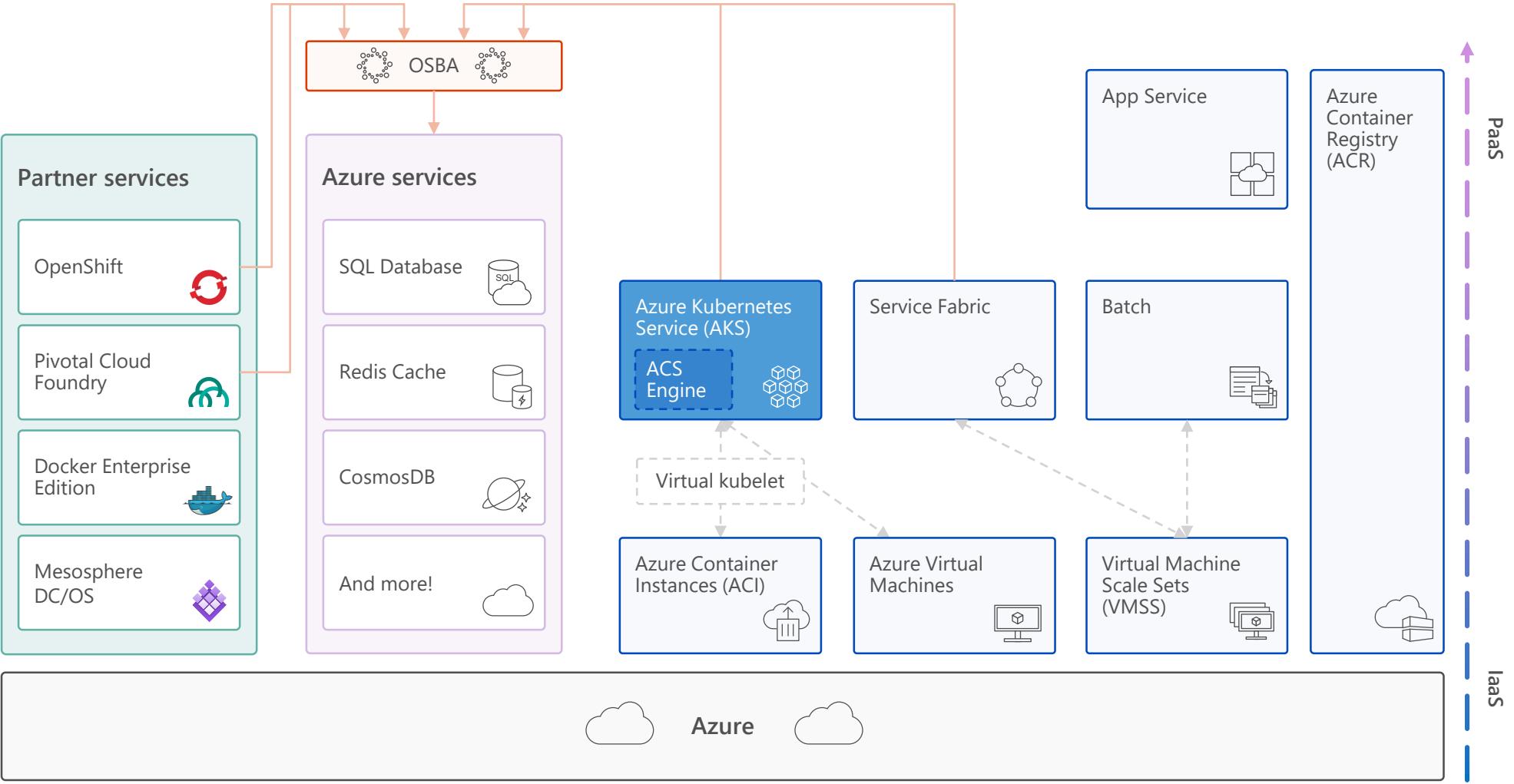


Open Service Broker API (OSBA)



Release Automation Tools

Azure Kubernetes Service (AKS)





Azure Kubernetes
Service (AKS)



Azure Container
Instances (ACI)



Azure Container
Registry



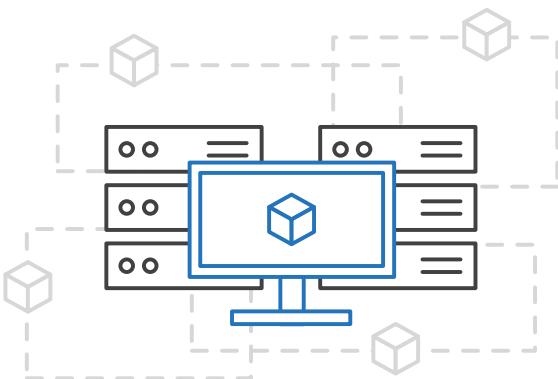
Open Service
Broker API (OSBA)



Release
Automation Tools

Azure Kubernetes Service (AKS)

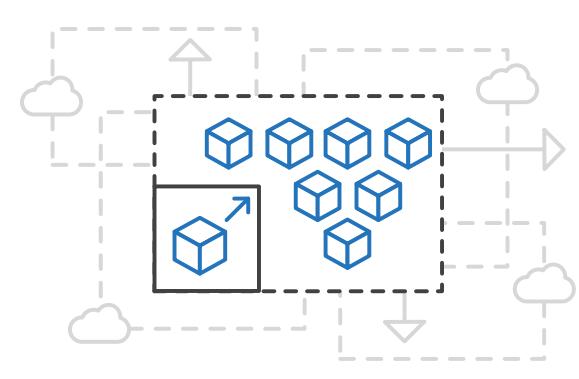
Simplify the deployment, management, and operations of Kubernetes



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



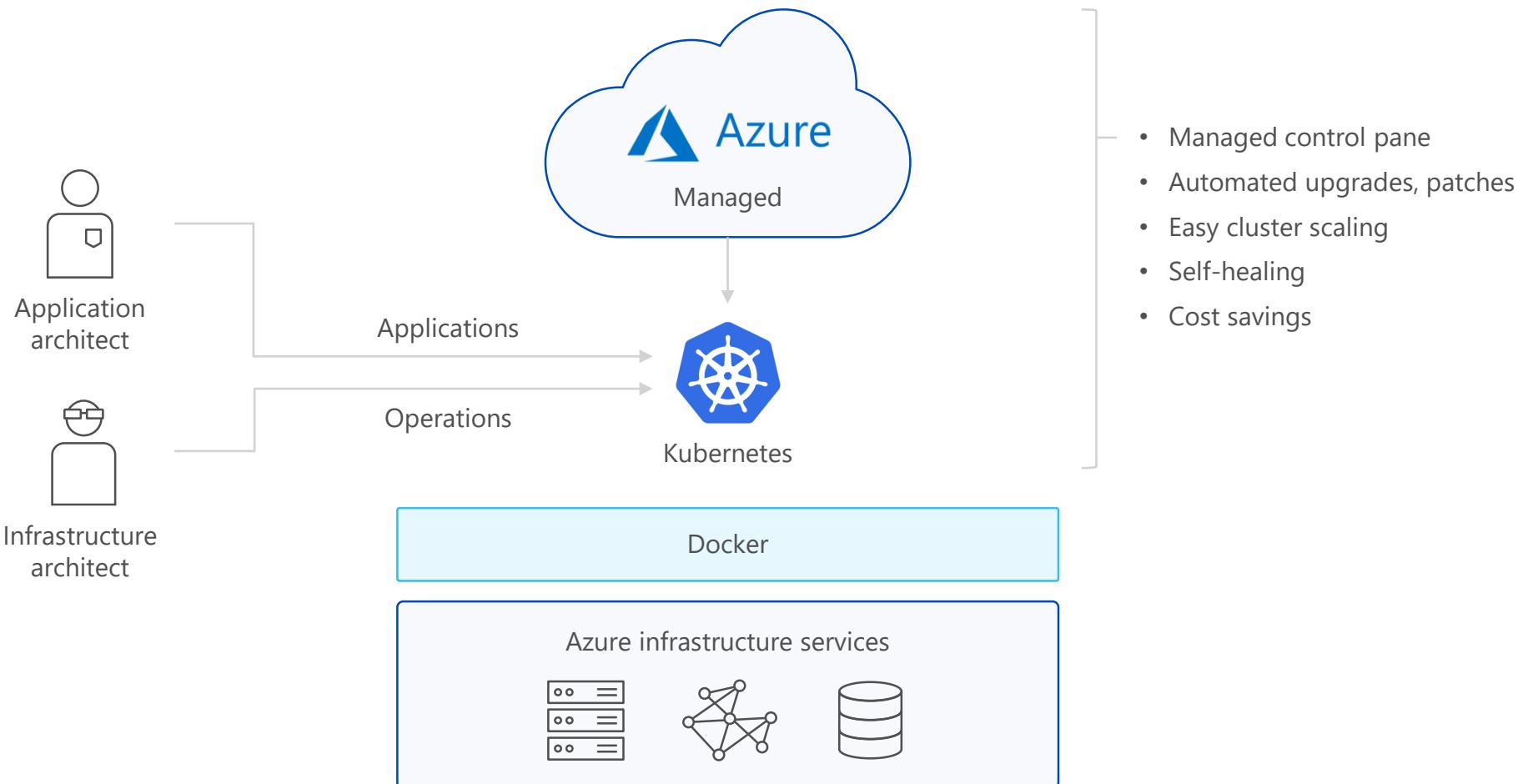
Open Service Broker API (OSBA)



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

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation menu includes options like 'Create a resource', 'All services', 'Dashboard', 'All resources', 'Resource groups', 'App Services', 'SQL databases', and 'Azure Cosmos DB'. The main content area displays a terminal window with the following command history:

```
Last login: Wed Oct 25 11:53:45 on ttys002
chzbrgr71 az group create --name myResourceGroup --location westus2
{
  "id": "/subscriptions/471d33fd-a776-405b-947c-467c291dc741/resourceGroups/myResourceGroup",
  "location": "westus2",
  "managedBy": null,
  "name": "myResourceGroup",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
chzbrgr71 az aks create --resource-group myResourceGroup --name myK8sCluster --agent-count 1 --generate-ssh-keys
```



Azure Kubernetes
Service (AKS)



Azure Container
Instances (ACI)



Azure Container
Registry



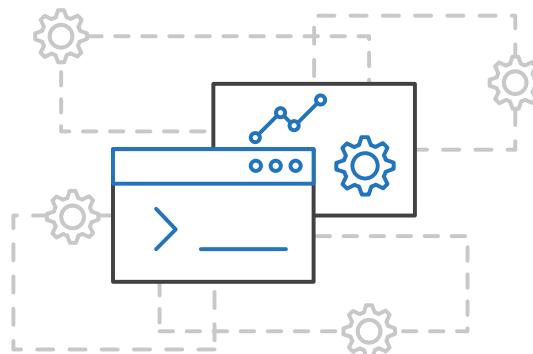
Open Service
Broker API (OSBA)



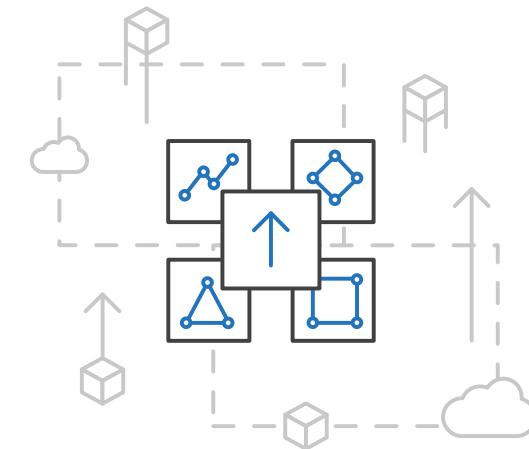
Release
Automation Tools

ACS Engine

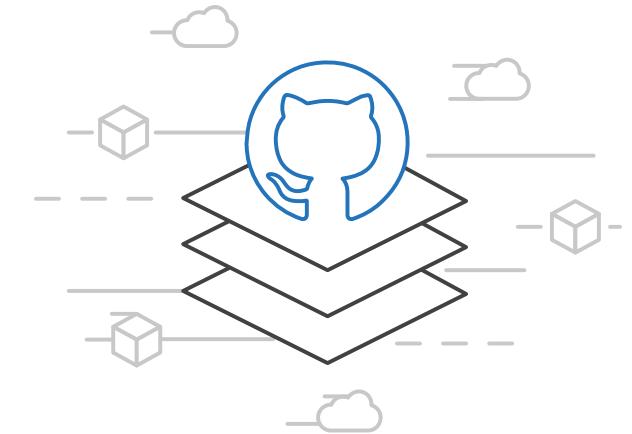
Customized Kubernetes on Azure



A proving ground
for new features



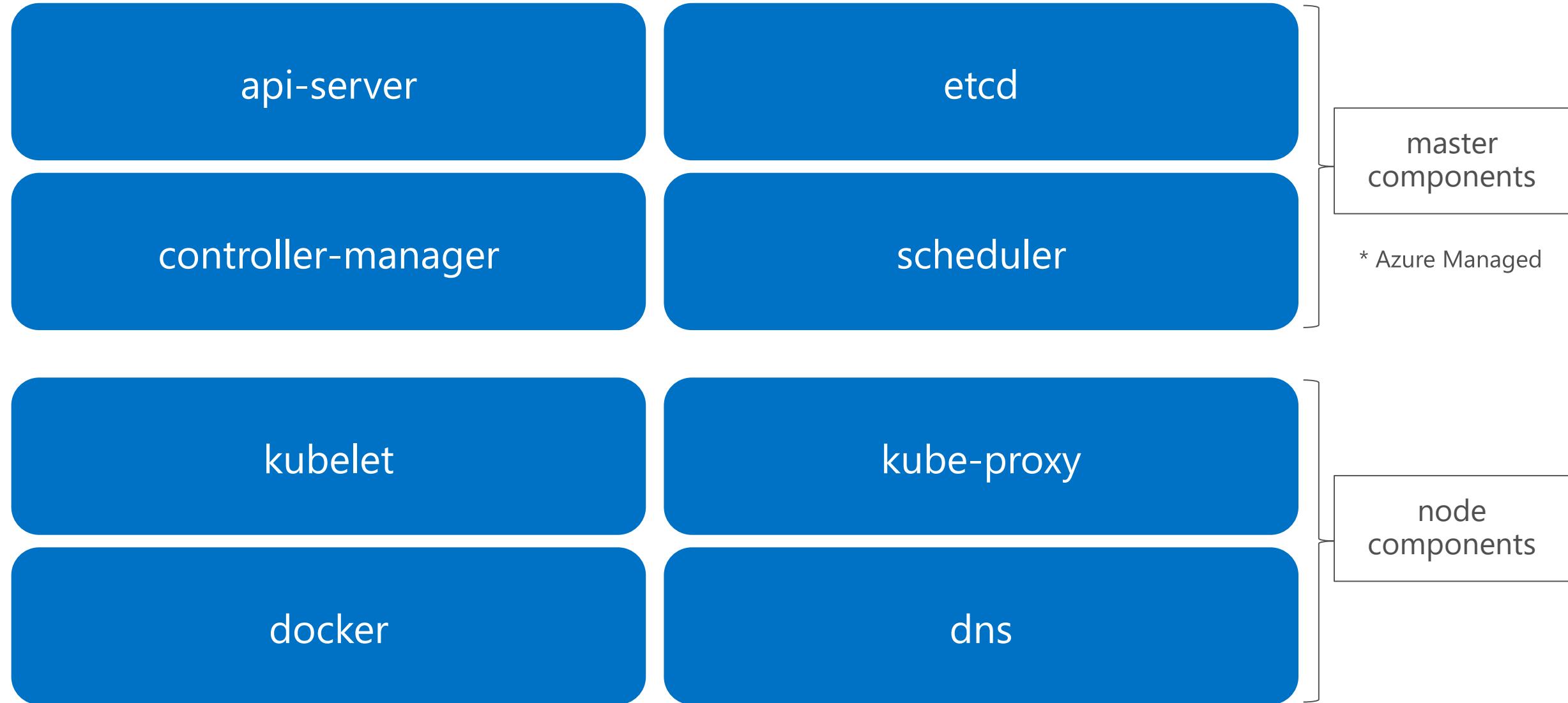
Enables custom
deployments



Available
on GitHub

Kubernetes Concepts

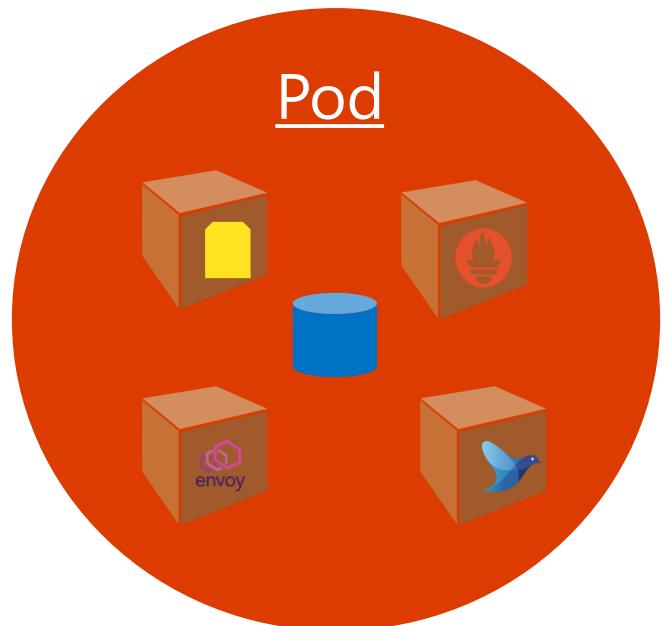
Kubernetes Architecture Components



Pods

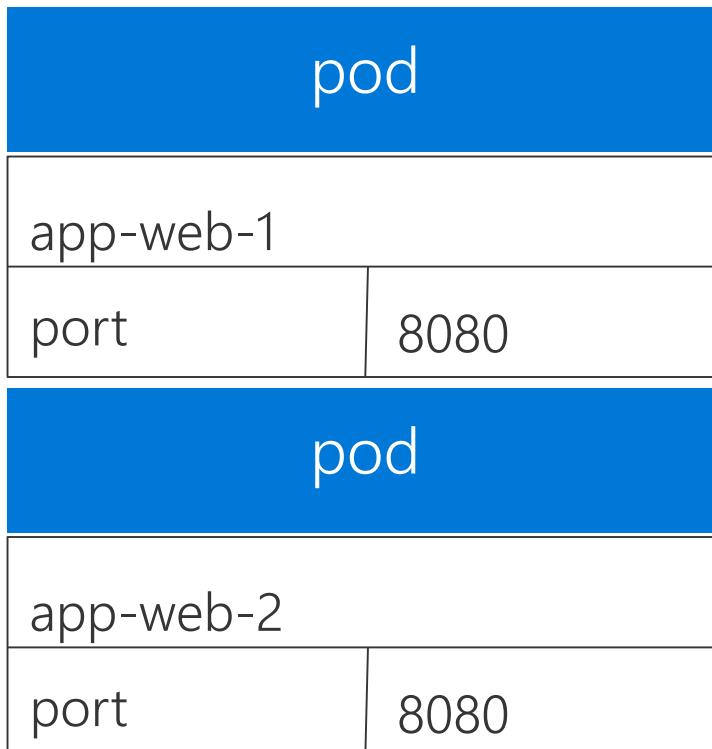
Pod is a set of containers that can run on a host

- Pod represents a logical construct to bundle one or more applications together
 - Sidecar Model
 - Pod resources deployed to a single agent
- An application-specific “logical host”
- Share:
 - Storage resources
 - Unique network IP
- Options to control how container(s) should run
- Most commonly Docker, but other runtimes are supported



Deployment

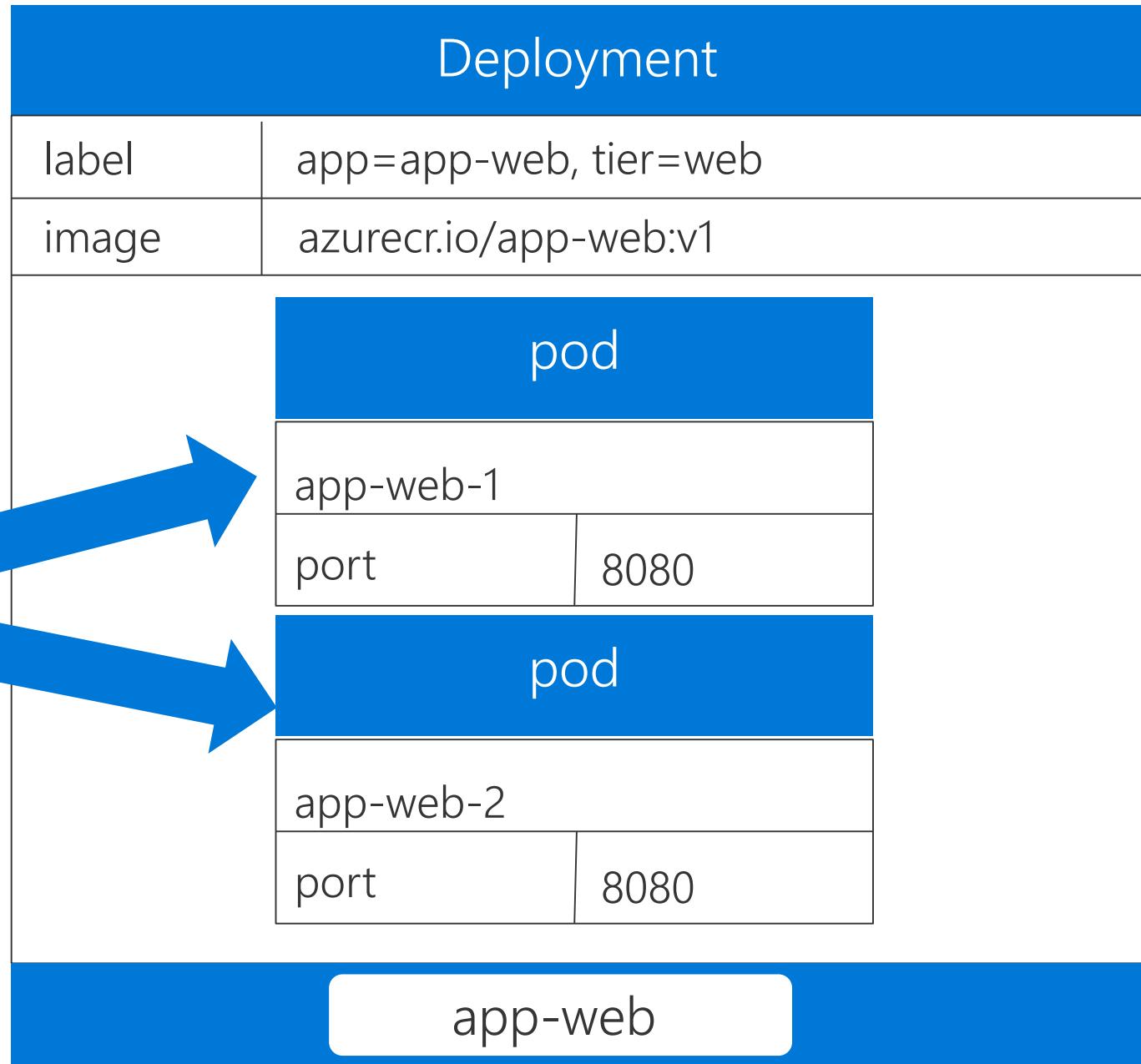
label	app=app-web , tier=web
image	azurecr.io/app-web:v1



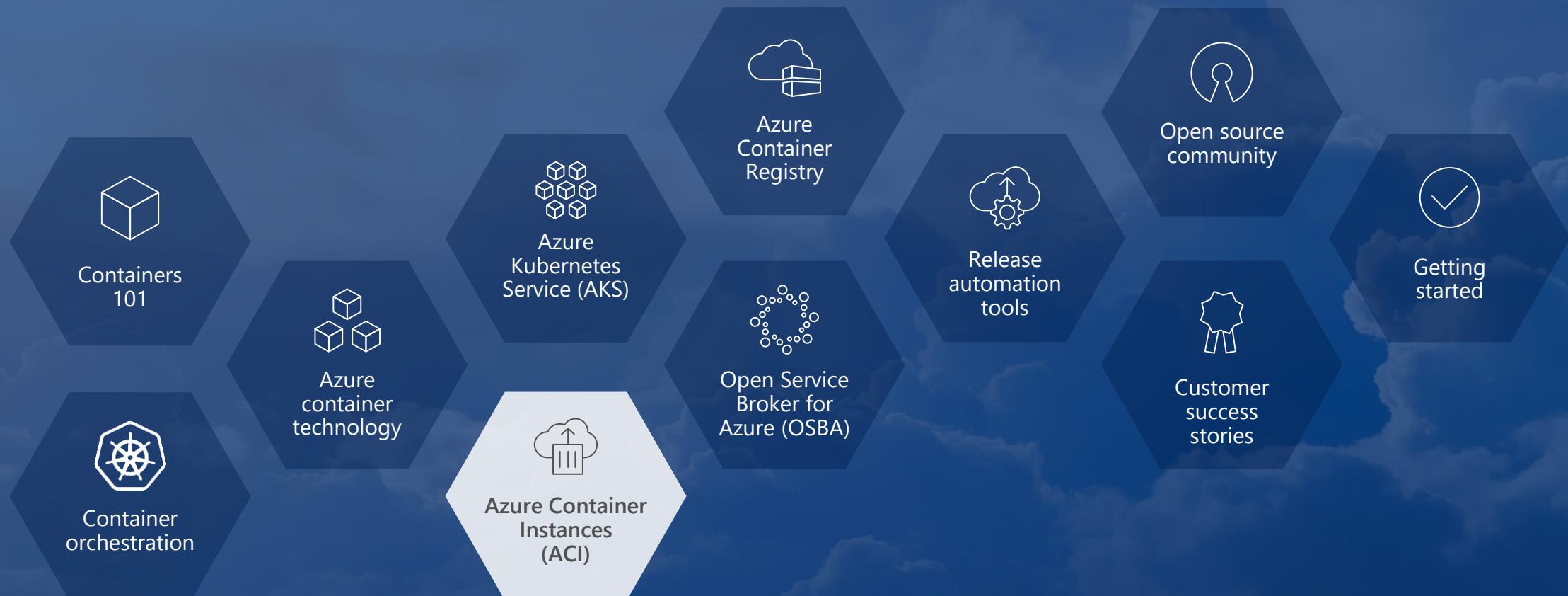
port

app-web

Service	
app-web	
selector	app=app-web
port	8080:8080
IP	10.0.2.20



Azure Container Instances (ACI)





Azure Kubernetes Service (AKS)



Azure Container Instances (ACI)



Azure Container Registry

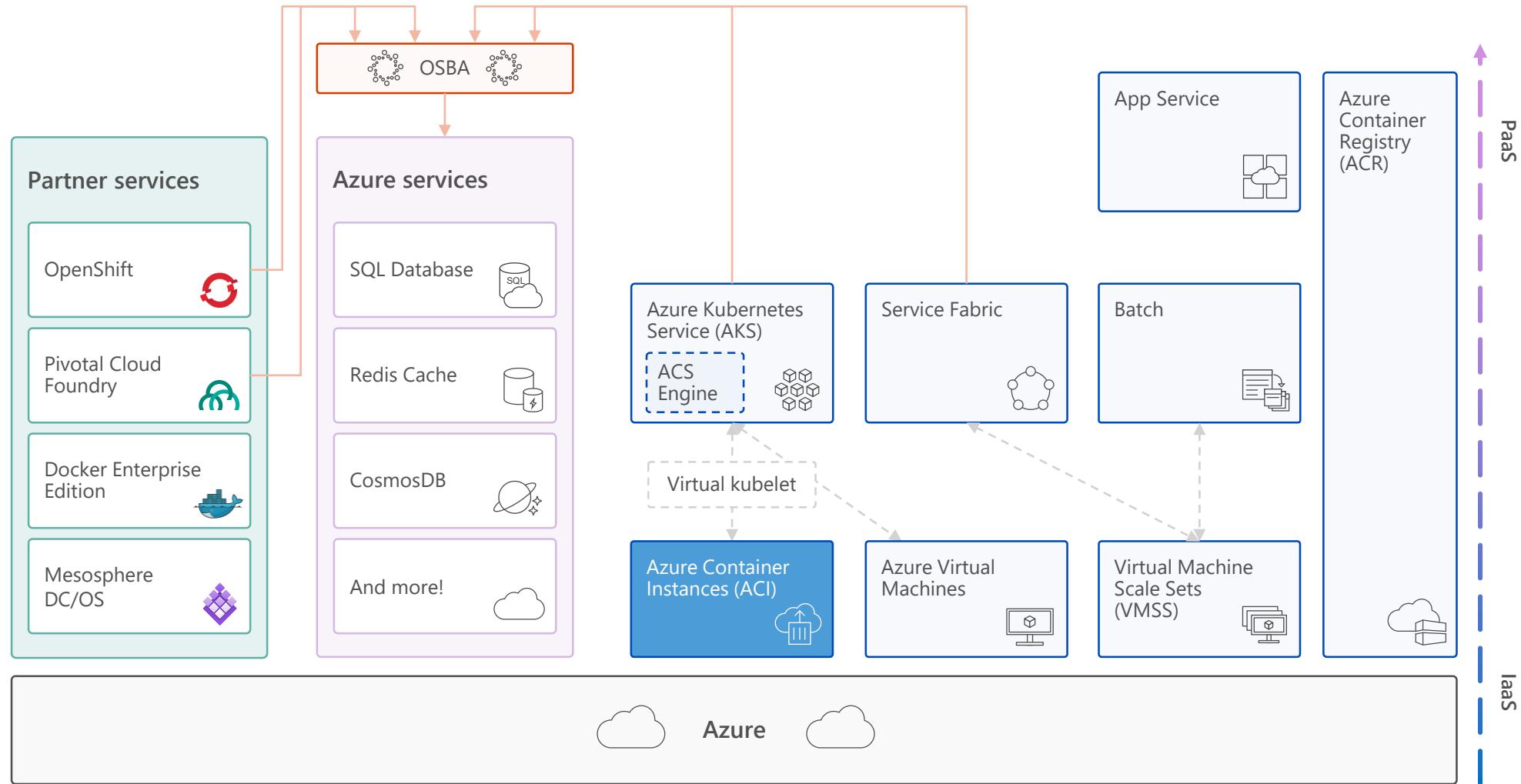


Open Service Broker API (OSBA)



Release Automation Tools

Azure Container Instances (ACI) PREVIEW





Azure Kubernetes
Service (AKS)



Azure Container
Instances (ACI)



Azure Container
Registry



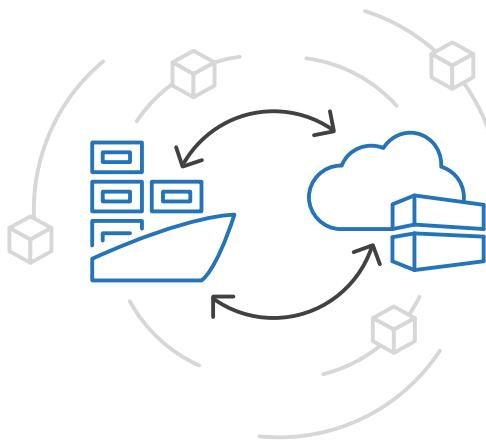
Open Service
Broker API (OSBA)



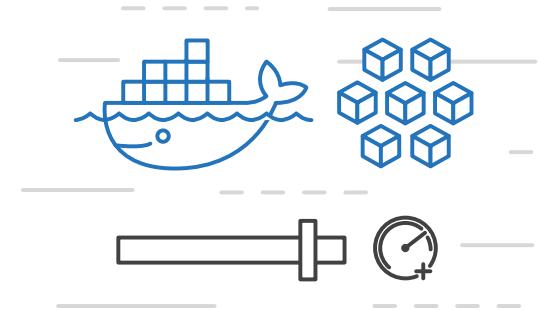
Release
Automation Tools

Azure Container Instances (ACI) PREVIEW

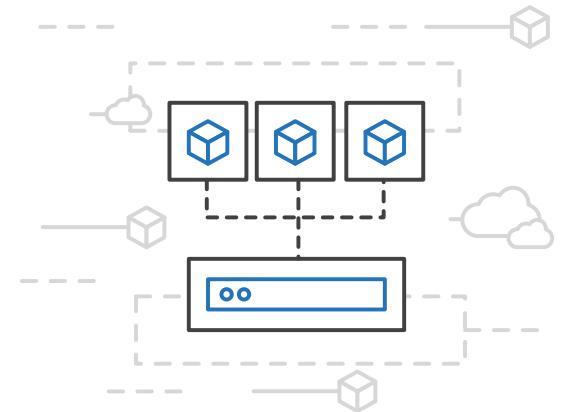
Easily run containers on Azure with a single command



Start using
containers right away



Cloud-scale
container capacity



Hyper-visror
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

The screenshot shows the Microsoft Azure portal's 'New' blade. On the left, there's a sidebar with 'Create a resource' and 'All services' options. Below that is a 'FAVORITES' section with links to Dashboard, All resources, Resource groups, App Services, SQL databases, and Azure Cosmos DB. The main area has tabs for 'Azure Marketplace' (selected) and 'Featured'. Under 'Azure Marketplace', there are links for 'Get started', 'Recently created', 'Compute', 'Networking', 'Storage', 'Web + Mobile', 'Containers' (which is highlighted with a dashed blue border), and 'Databases'. Under 'Featured', there are cards for 'Azure Container Service - AKS (preview)', 'Azure Container Service', and 'Azure Container Instances (preview)' (which is also highlighted with a dashed blue border). Each card includes a 'Learn more' link.

Microsoft Azure

Report a bug

Search resources, services ar...

Home > New

New

+

Create a resource

All services

★ FAVORITES

Dashboard

All resources

Resource groups

App Services

SQL databases

Azure Cosmos DB

Azure Marketplace

See all

Get started

Recently created

Compute

Networking

Storage

Web + Mobile

Containers

Databases

Containers

Azure Container Instances (preview)

Azure Container Service

Azure Container Registry

PREVIEW

Learn more

Learn more

Learn more



Azure Kubernetes
Service (AKS)



Azure Container
Instances (ACI)



Azure Container
Registry



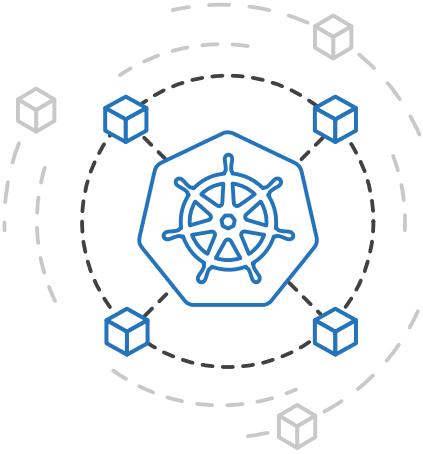
Open Service
Broker API (OSBA)



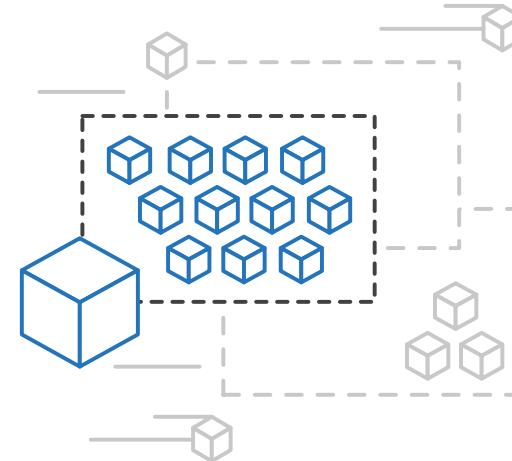
Release
Automation Tools

Azure Container Instances (ACI) PREVIEW

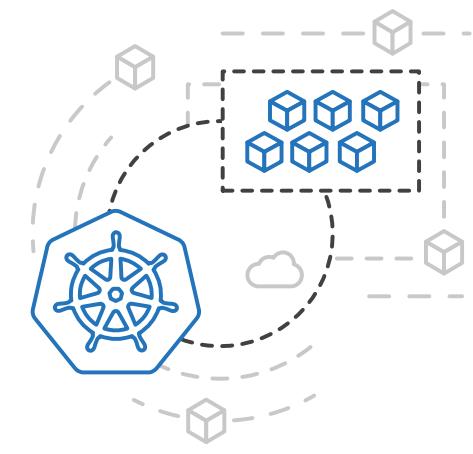
ACI Connector for Kubernetes



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



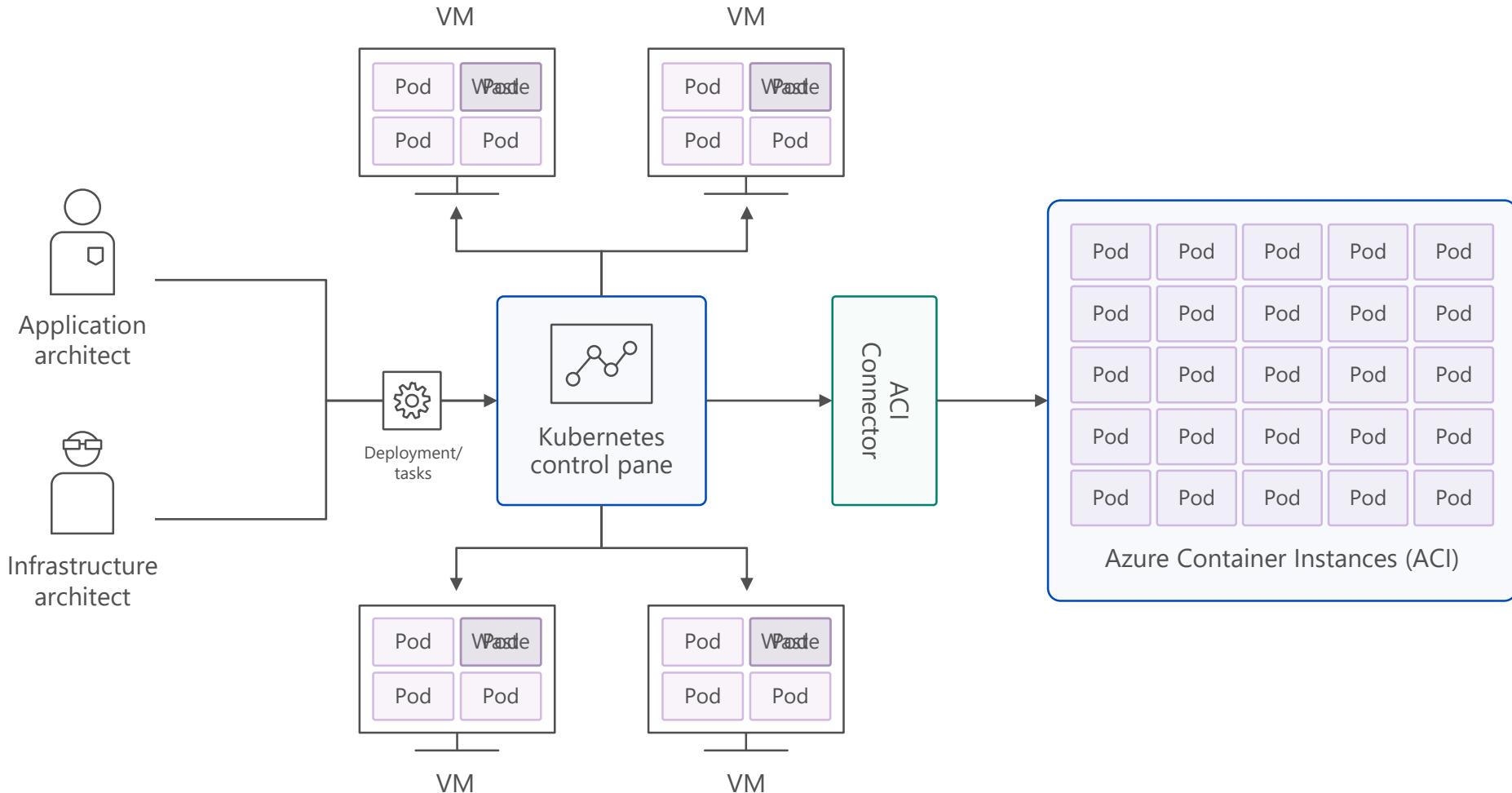
Open Service Broker API (OSBA)



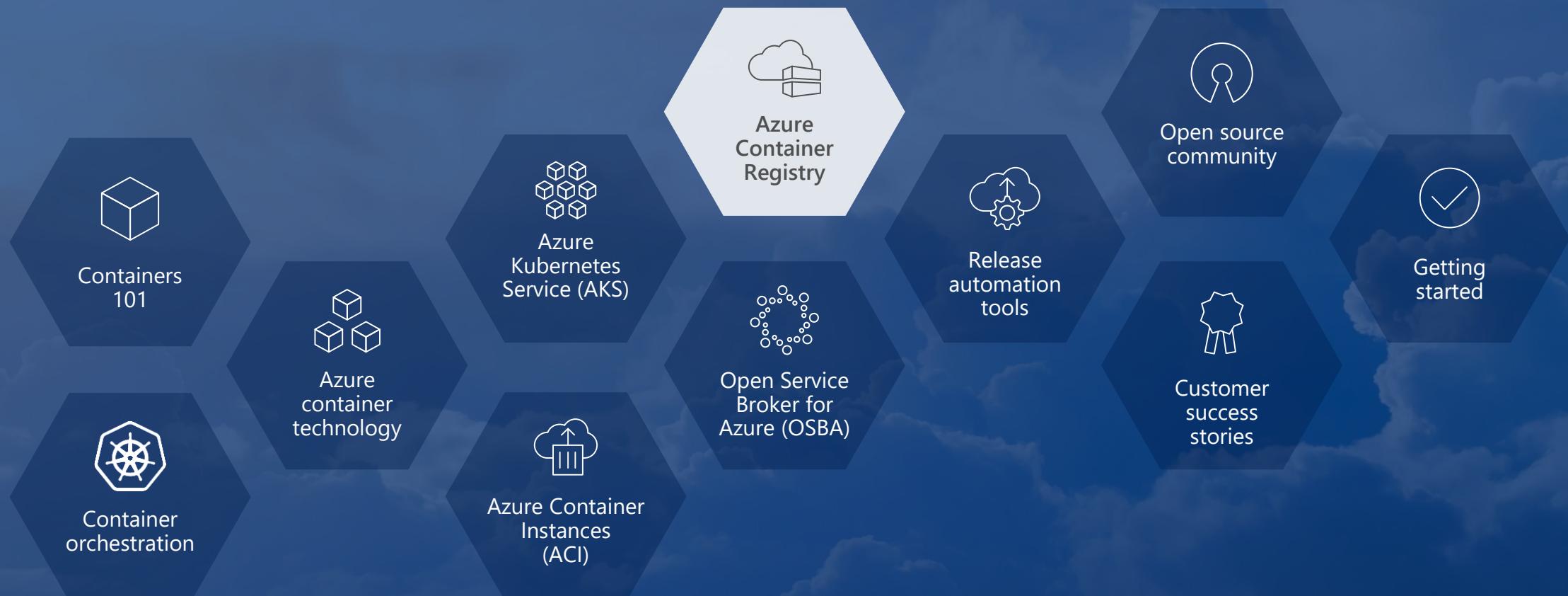
Release Automation Tools

Azure Container Instances (ACI) PREVIEW

Bursting with the ACI Connector



Azure Container Registry





Azure Kubernetes Service (AKS)



Azure Container Instances (ACI)



Azure Container Registry

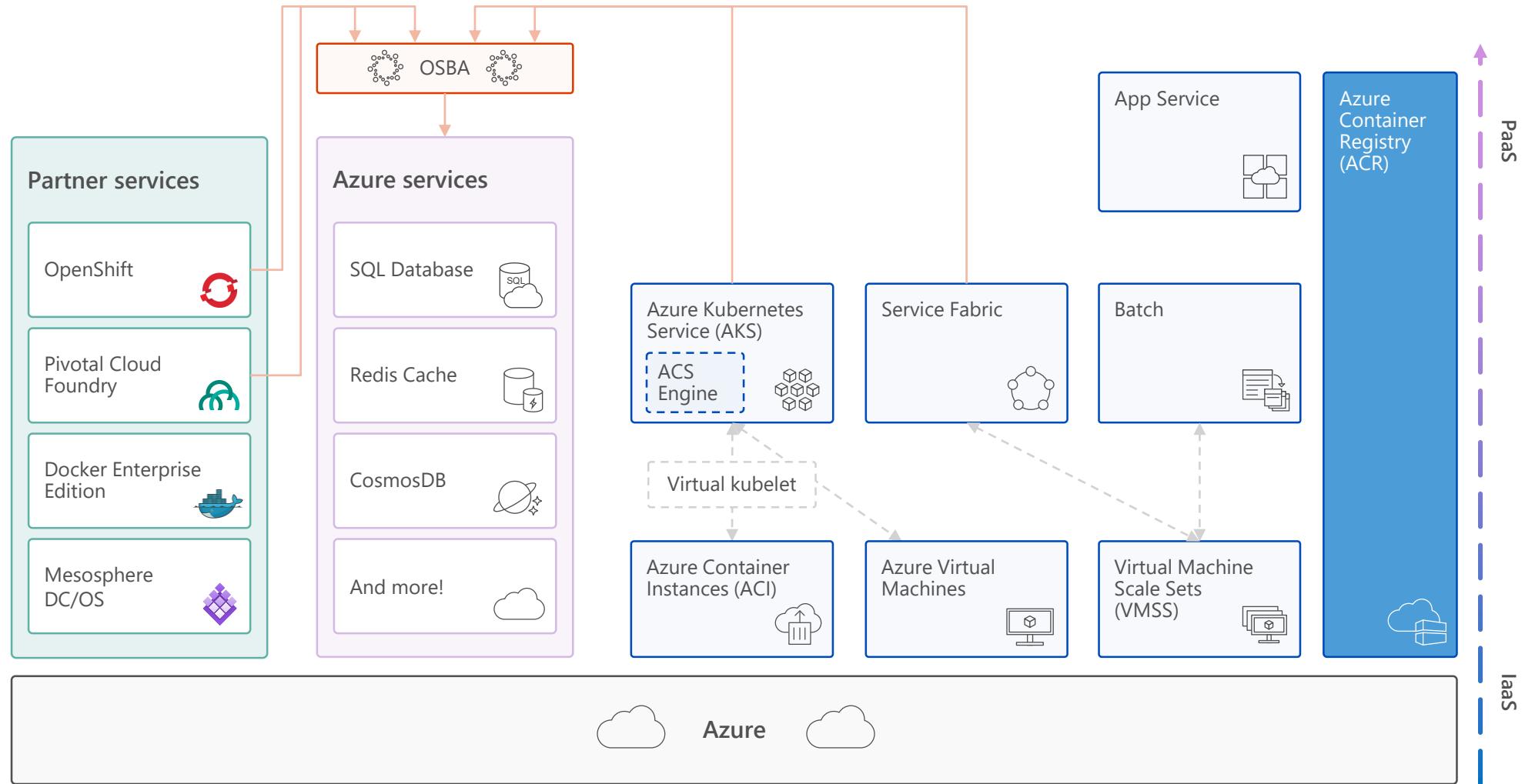


Open Service Broker API (OSBA)



Release Automation Tools

Azure Container Registry





Azure Kubernetes
Service (AKS)



Azure Container
Instances (ACI)



Azure Container
Registry



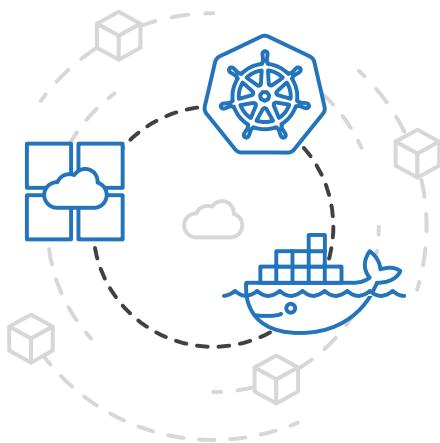
Open Service
Broker API (OSBA)



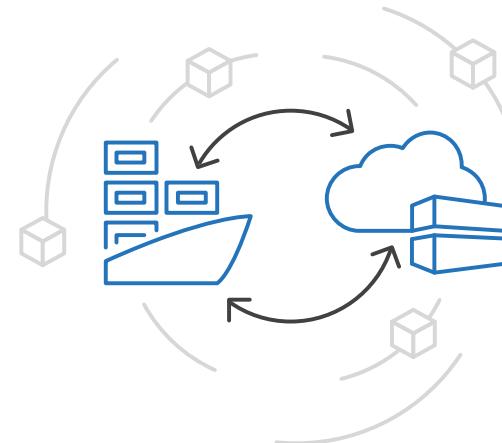
Release
Automation Tools

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)



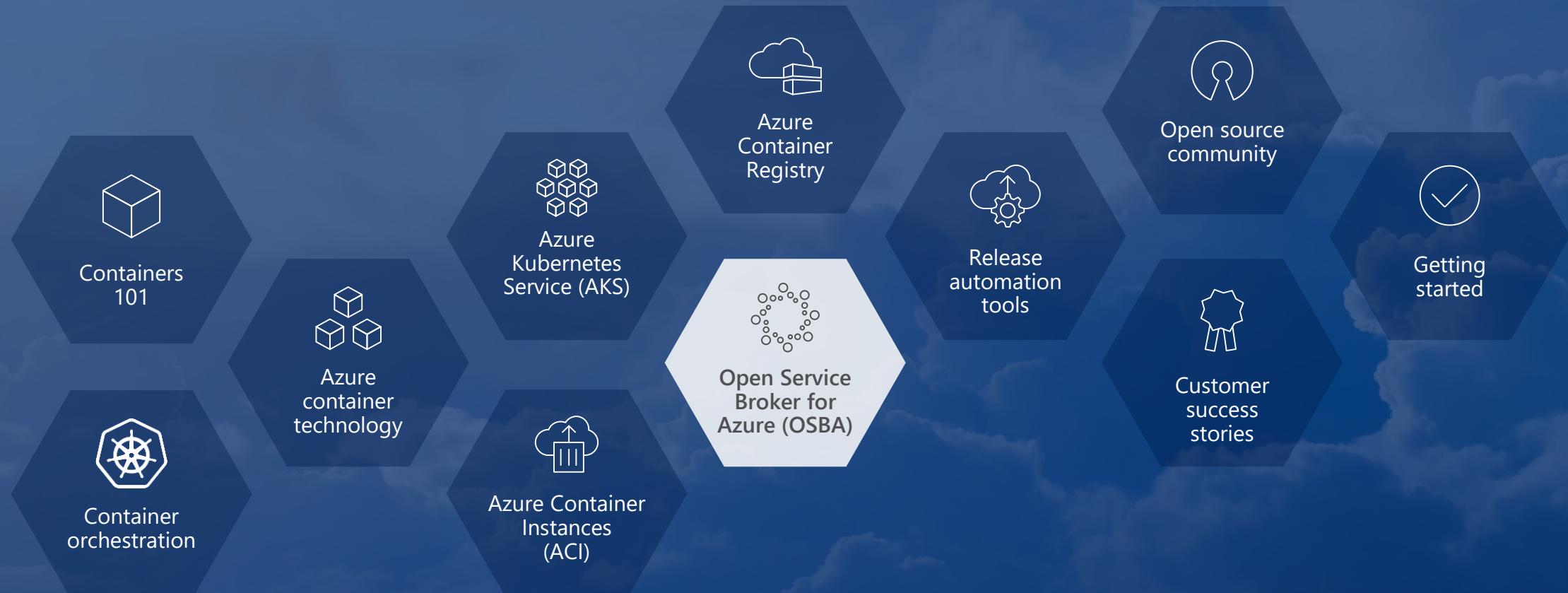
Release
Automation Tools

Azure Container Registry

Create a container in the Registry quickly

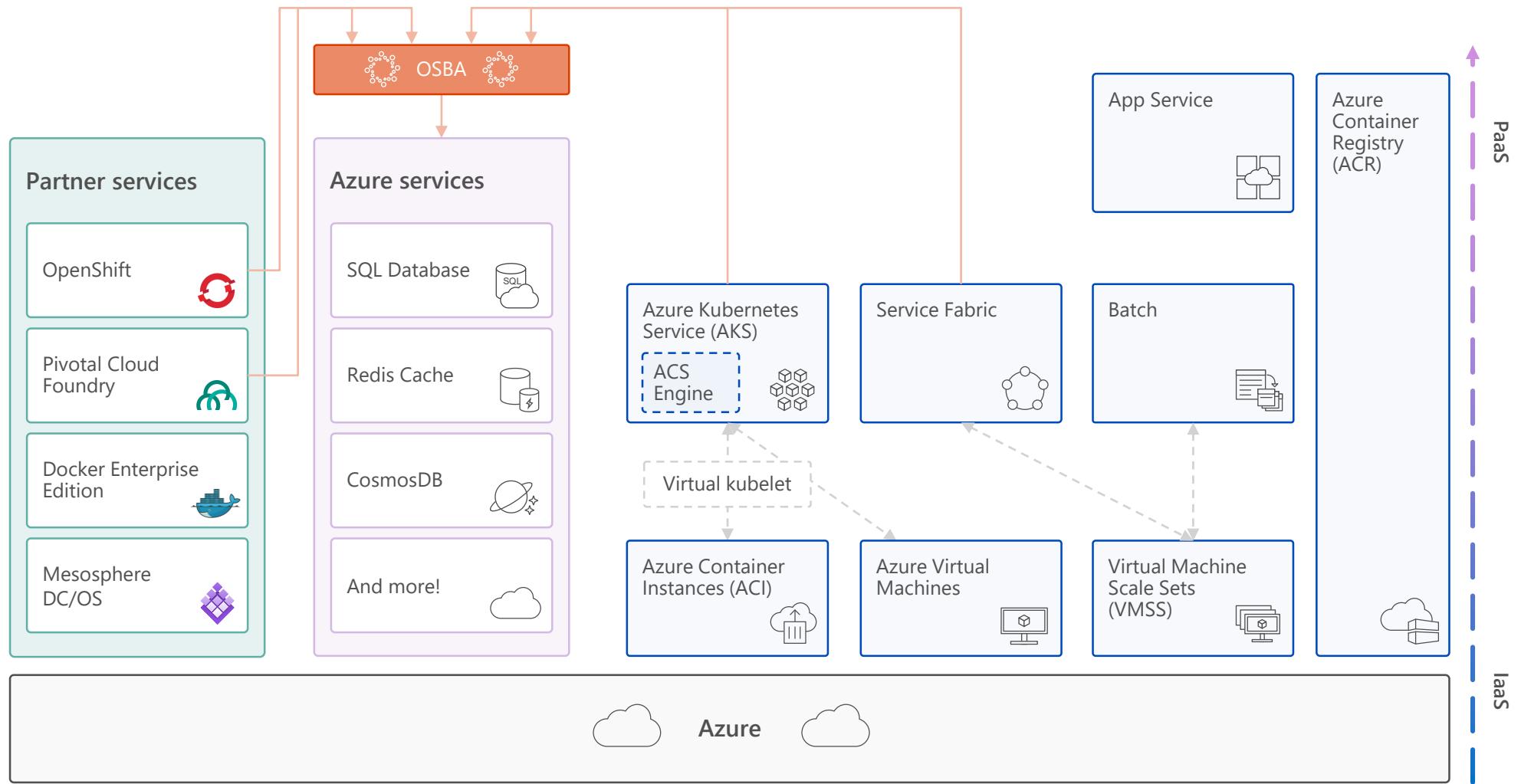
The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with various service icons and links like 'Create a resource', 'All services', 'Dashboard', etc. The main area is titled 'New' and features sections for 'Azure Marketplace' and 'Featured'. Under 'Featured', several services are listed: 'Azure Container Service - AKS (preview)' (with a 'PREVIEW' badge), 'Azure Container Service' (with a 'Learn more' link), 'Azure Container Instances (preview)' (with a 'Learn more' link), and 'Azure Container Registry' (which is highlighted with a dashed blue rectangle). Other options like 'Compute', 'Networking', 'Storage', 'Web + Mobile', and 'Databases' are also visible.

Open Service Broker for Azure





Open Service Broker for Azure (OSBA)





Azure Kubernetes
Service (AKS)



Azure Container
Instances (ACI)



Azure Container
Registry



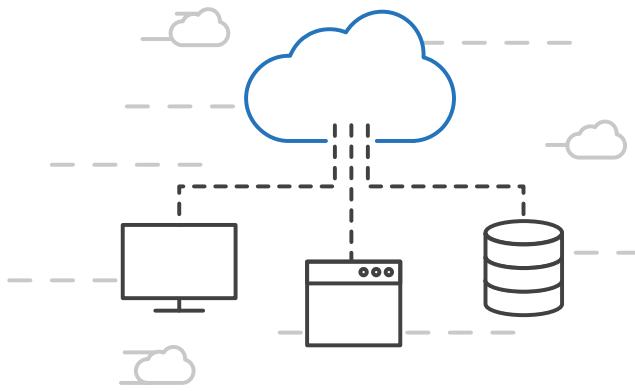
Open Service
Broker API (OSBA)



Release
Automation Tools

Open Service Broker for Azure (OSBA)

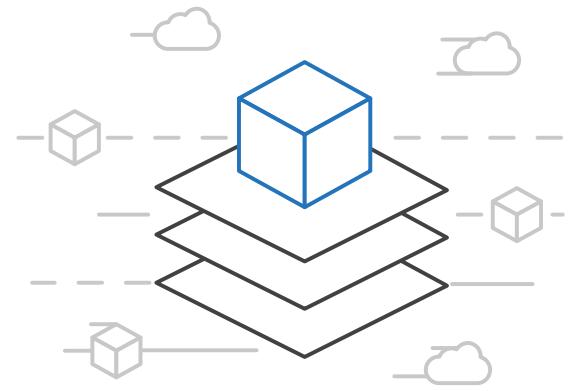
Connecting containers to Azure services and platforms



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 SQL Database



Redis Cache



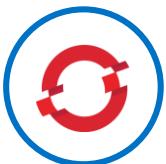
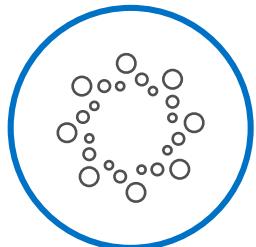
CosmosDB



And more!



Open Service Broker
for Azure (OSBA)



OpenShift



Cloud Foundry



Service Fabric
(Coming soon)



Kubernetes
(AKS)

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

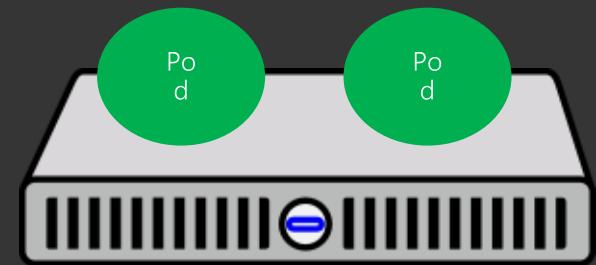
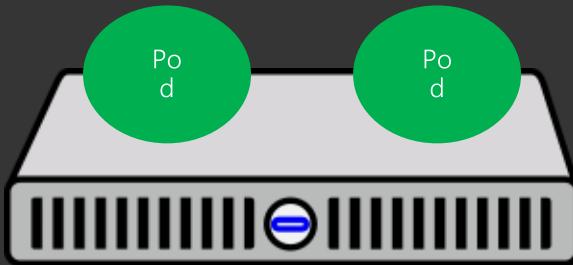
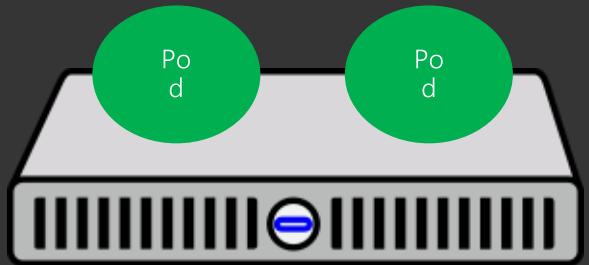
Increase/decrease the number of pods for a given service

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

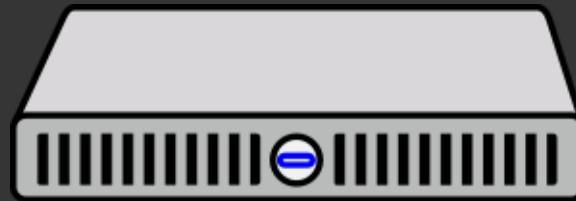
 Copy

 Try It

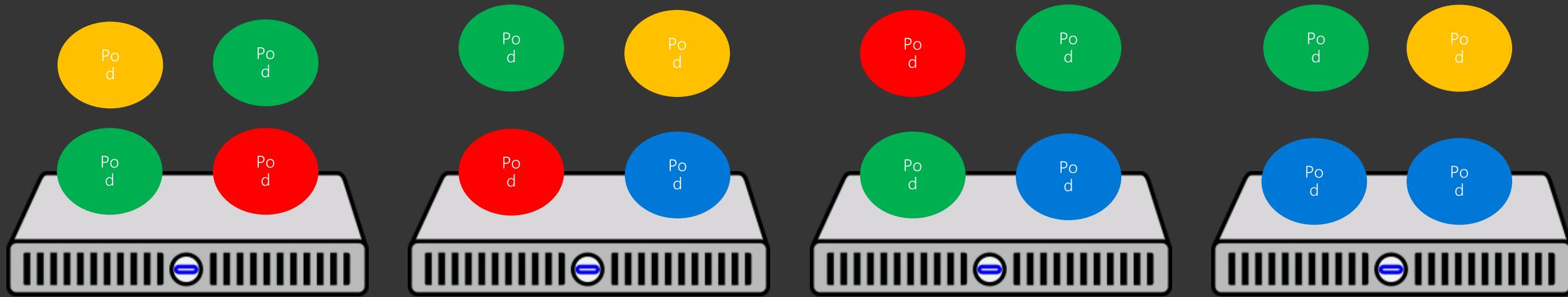
```
az aks scale --name myK8sCluster --resource-group myResourceGroup --agent-count 1
```

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
```

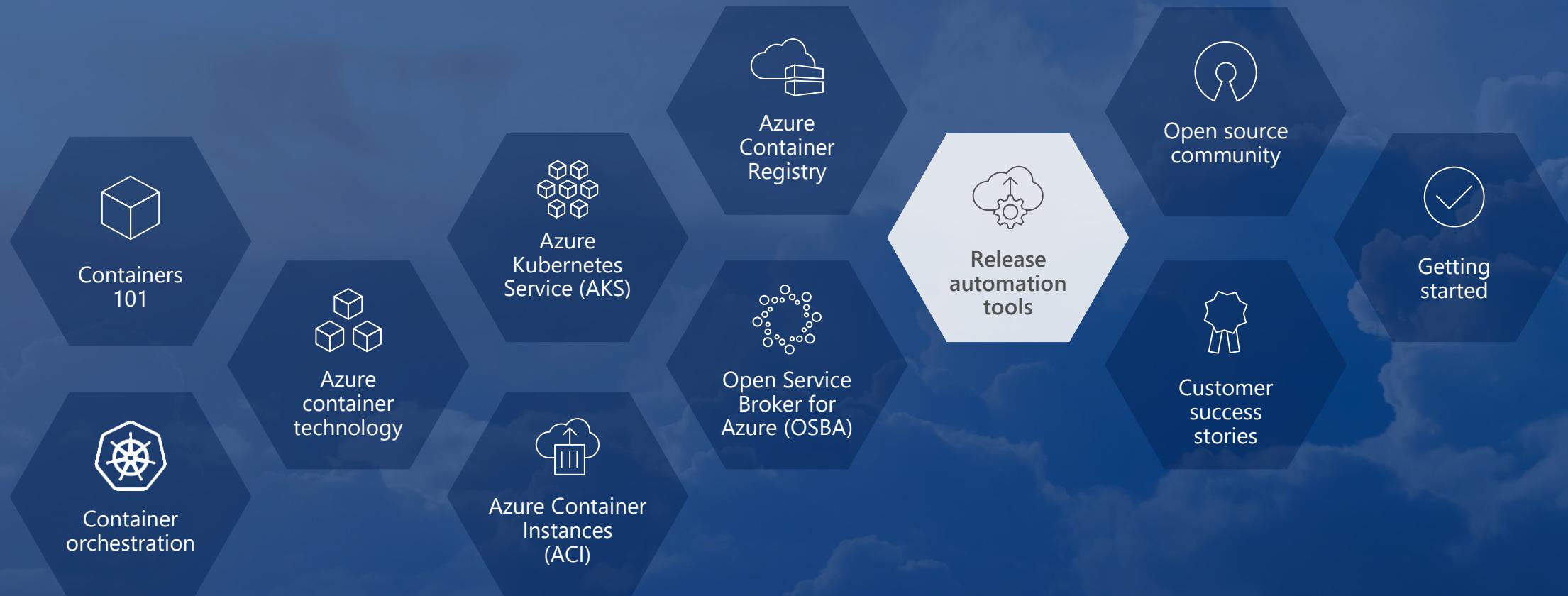
Azure CLI

 Copy

 Try It

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

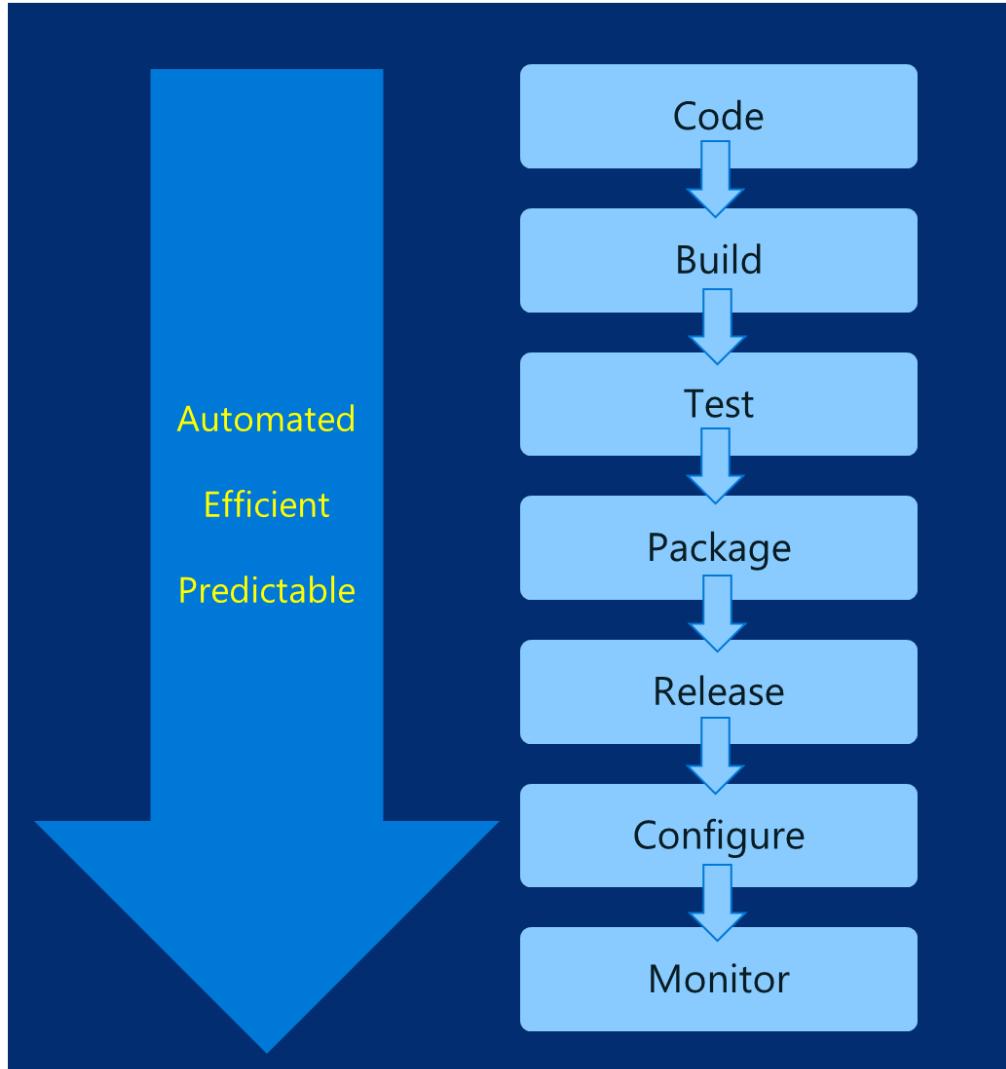
Release automation tools



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

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



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

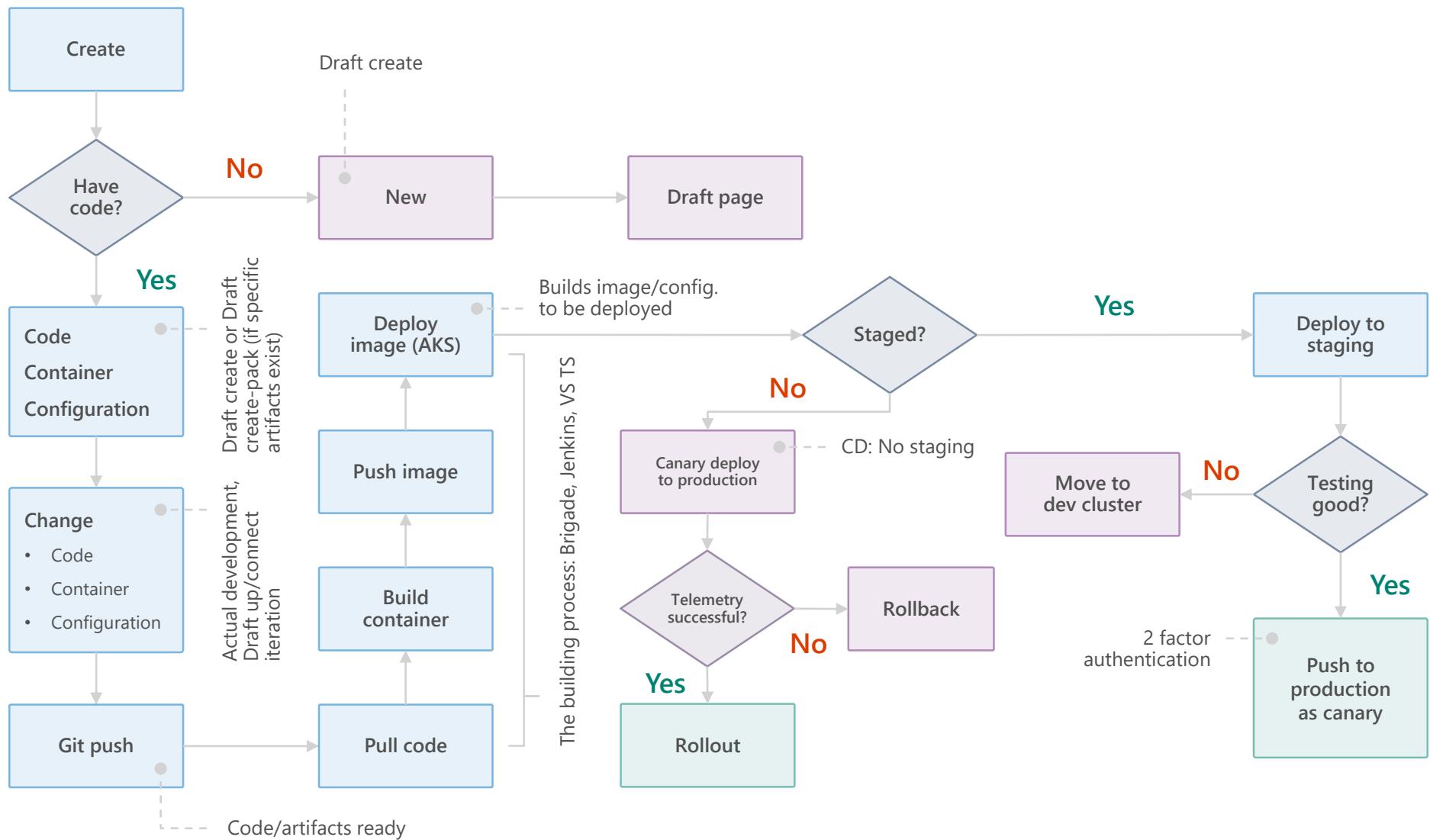


Open Service
Broker API (OSBA)



Release Automation Tools

Release automation workflow





Azure Kubernetes
Service (AKS)



Azure Container
Instances (ACI)



Azure Container
Registry



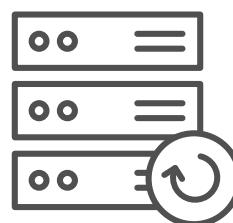
Open Service
Broker API (OSBA)



Release
Automation Tools

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



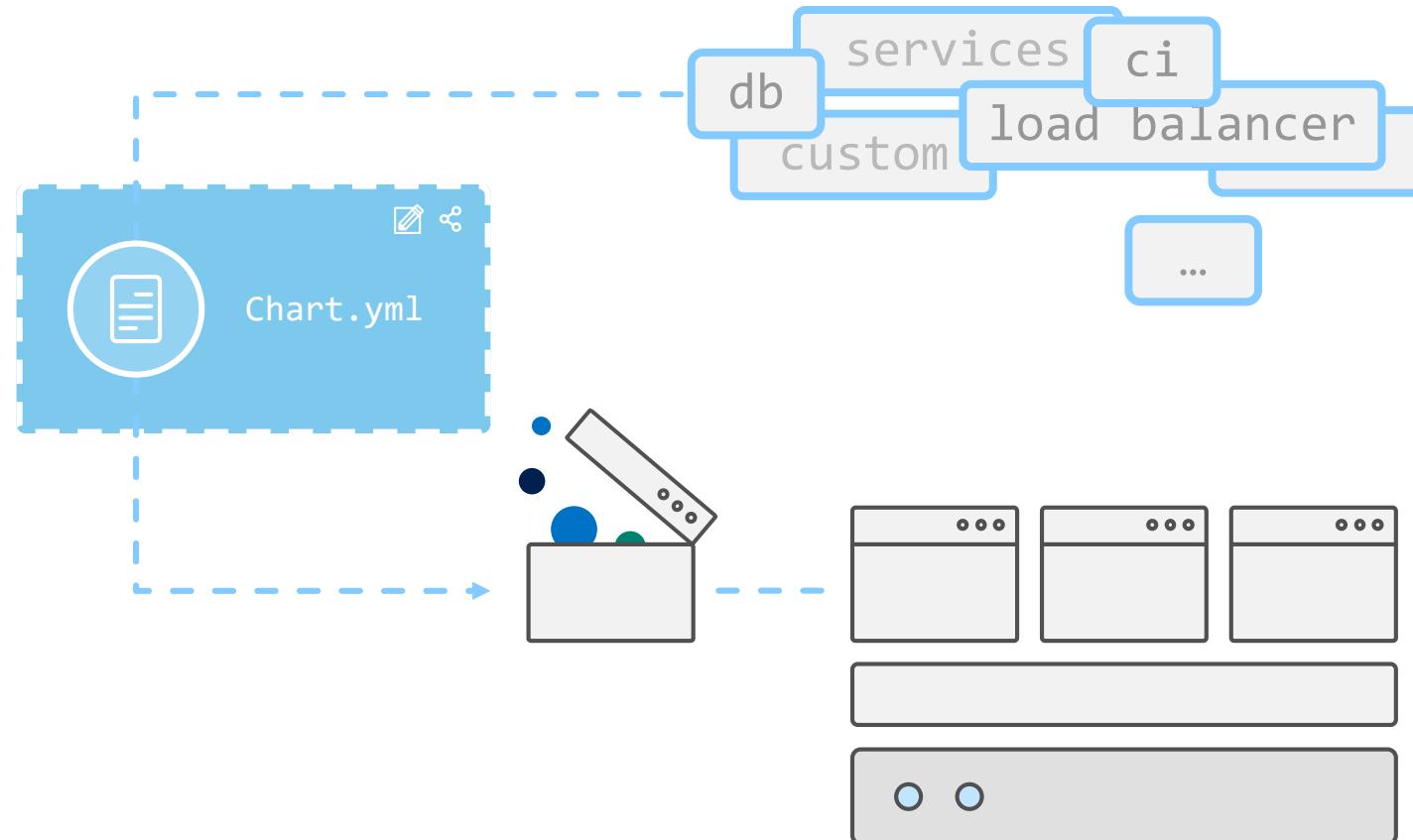
Open Service
Broker API (OSBA)



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 (server 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



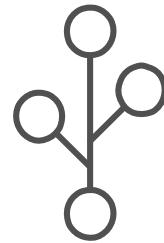
Open Service
Broker API (OSBA)



Release
Automation Tools

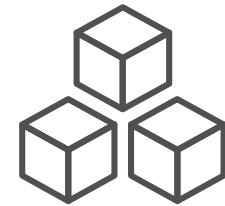
Brigade

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





Azure Kubernetes Service (AKS)



Azure Container Instances (ACI)



Azure Container Registry



Open Service Broker API (OSBA)



Release Automation Tools

Brigade

Brigade in action

The screenshot shows a code editor interface with the following details:

- EXPLORER:** Shows a file tree with the following files:
 - OPEN EDITORS: Welcome, README.md, JS brigade.js (selected)
 - RATING-DB:
 - .gitignore
 - brig-project-update.yaml
 - JS brigade.js (selected)
 - db.yaml
 - heroes.json
 - import.sh
 - ratings.json
 - README.md
 - sites.json
 - DOCKER
 - AZURE STORAGE
 - AZURE COSMOS DB
 - KUBERNETES
- CODE EDITOR:** The JS brigade.js file is open, displaying the following code:

```
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("dbACRImage")}`)
  brigConfig.set("dbACRImage", `${brigConfig.get("acrServer")}/${brigConfig.get("acrUsername")}`)

  console.log(`=> GitHub webhook ${brigConfig.get("branch")}) with commit ${gitPayload.sha}`)

  // 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("dbImage")}`)
  var pipeline = new Group()
  pipeline.add(docker)
  pipeline.add(helm)
})
```

At the bottom of the editor, there are status indicators: draft-pack-version+, 0/0, Azure: rasquill@microsoft.com, Ln 78, Col 2, Spaces: 4, UTF-8, LF, JavaScript, and a smiley face icon.



Azure Kubernetes
Service (AKS)



Azure Container
Instances (ACI)



Azure Container
Registry



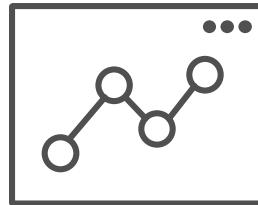
Open Service
Broker API (OSBA)



Release
Automation Tools

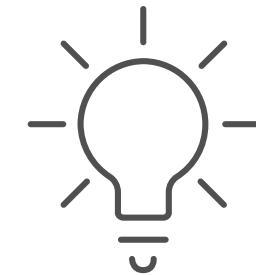
Kashti

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

[Build #01C0HX5S1GH7A0TZBJA2EYG7R1](#)

Started at 2017-12-23T07:31:49Z | Finished at 2017-12-23T07:36:44Z | Passed

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

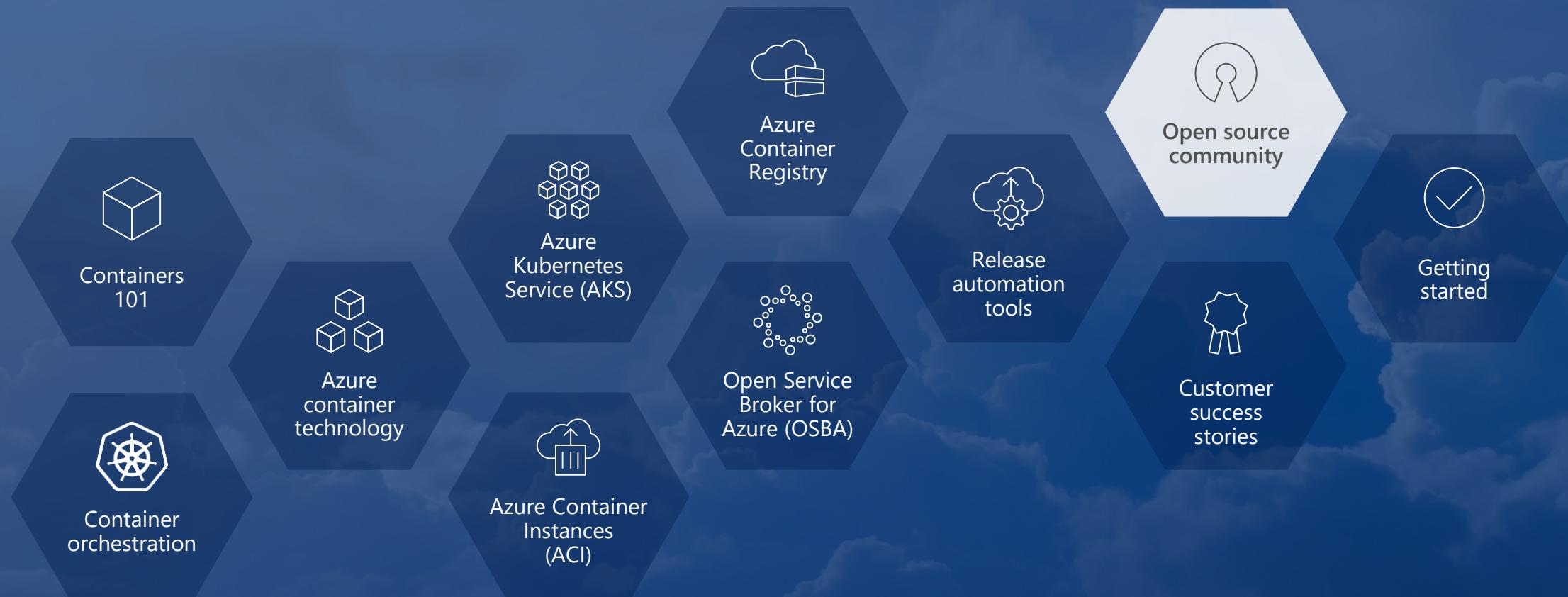
Job Status	Task	Branch	ID	Timestamp	Duration	Details
Passed	brigade-cli	master	#01c0hx5s1gh7a0tzbj...eyg7r1	Succeeded a month ago.	Ran for 295 seconds.	Details >
Failed	brigade-cli	master	#01c0hx0yxqa15t7fdcqxe7ernw	Failed a month ago.	Ran for 268 seconds.	
Failed	brigade-cli	master	#01c0hmmh4h41x0mq7z9sr794cr	Failed a month ago.	Ran for 11 seconds.	
Failed	brigade-cli	master	#01c0hmj9cfxr6eb6cq818yac5n	Failed a month ago.	Ran for 35 seconds.	
Failed	brigade-cli	master	#01c0hmawqnw9332yg1wd4jqgah	Failed a month ago.	Ran for 68 seconds.	
Failed	brigade-cli	master	#01c0hm8ekar4vbybpd93d475mt	Failed a month ago.	Ran for 49 seconds.	
Passed	brigade-cli	master	#01c0hm249jgz2jcje4wj2cf0sn	Succeeded a month ago.	Ran for 106 seconds.	
Failed	brigade-cli	master	#01c0hkb91dkxzshp58tx18qcf	Failed a month ago.	Ran for 70 seconds.	
Failed	brigade-cli	master	#01c0hkwhmegs7ywvjjzbngcbhb	Failed a month ago.	Ran for 8 seconds.	

Events log

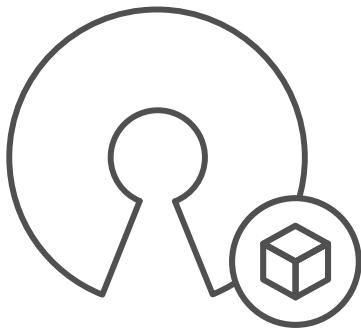
Questions?



Open source community



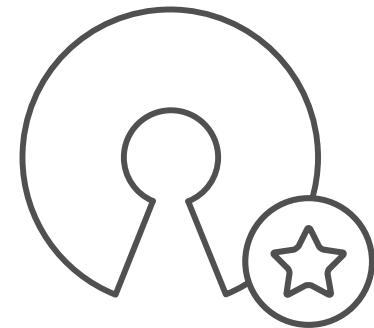
Community culture



Open source container
code contributions



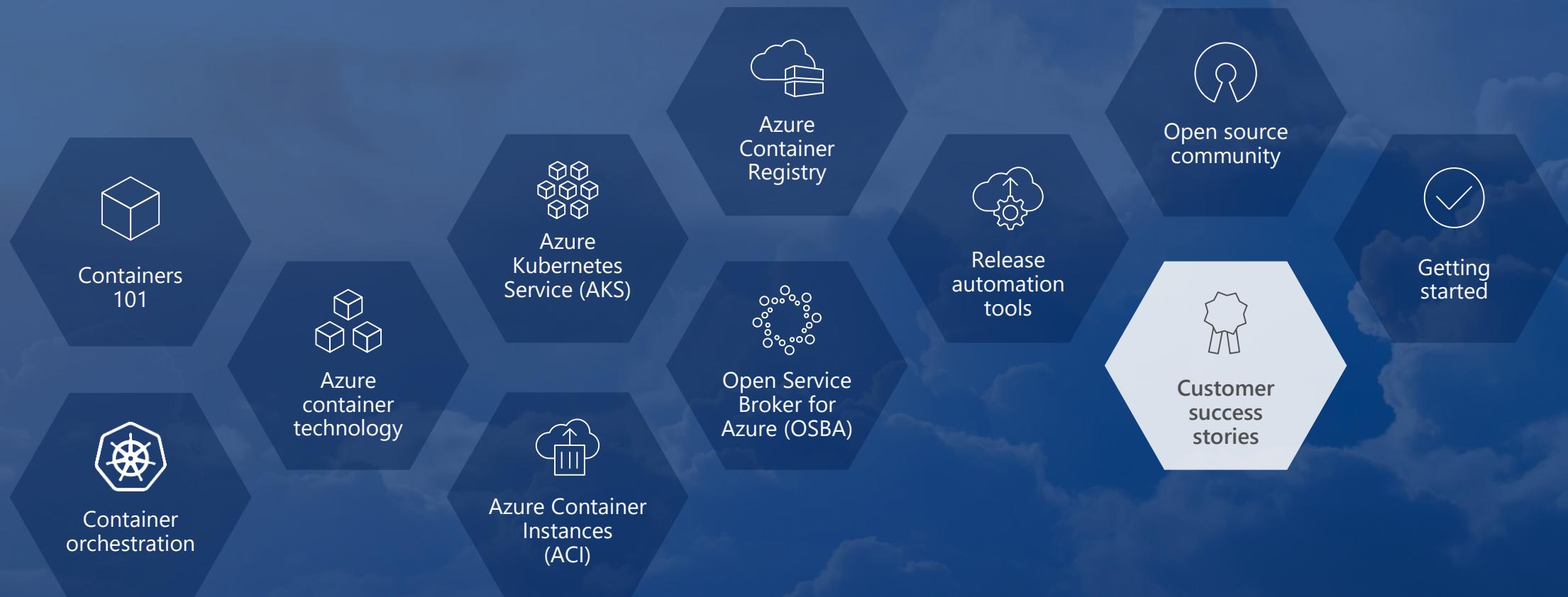
Numerous open source
project builds



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.

Products and services

Microsoft Azure Container Service

Organization size

1,000 employees

Industry

Power and utilities

Country

United States

Business need

Optimize operational efficiency





Azure



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

Microsoft Azure Container Service

Organization size

100,000+ employees

Industry

Healthcare

Country

Germany

Business need

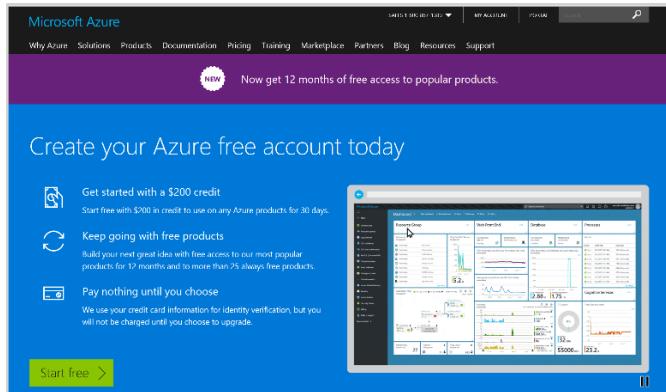
Faster application development



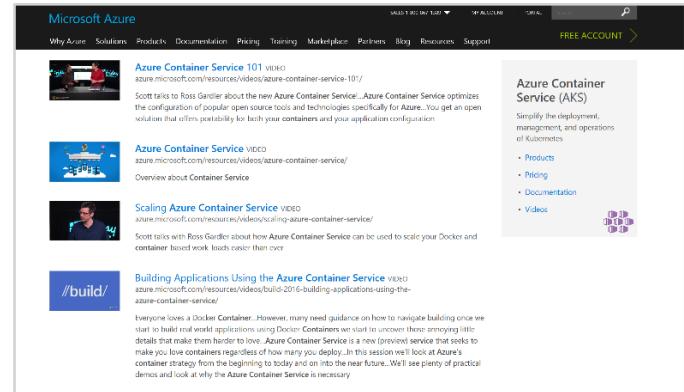
Azure container **resources**

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

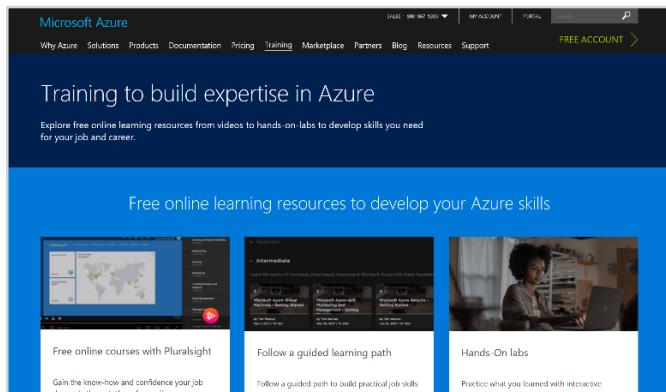
Sign up for a free Azure account



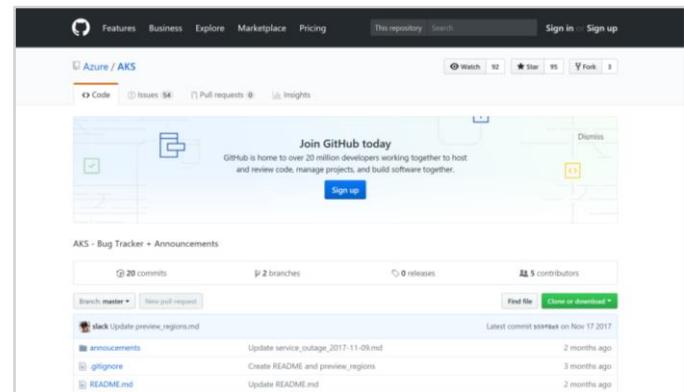
Check out the Azure container videos page



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

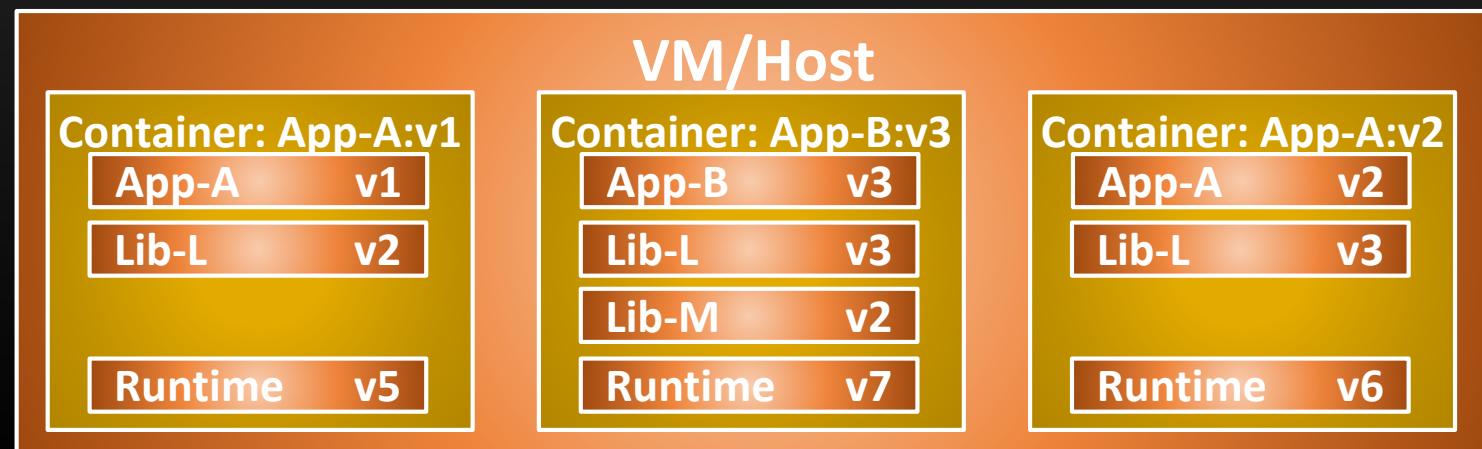
- it's own file system

- 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





/lib



/src

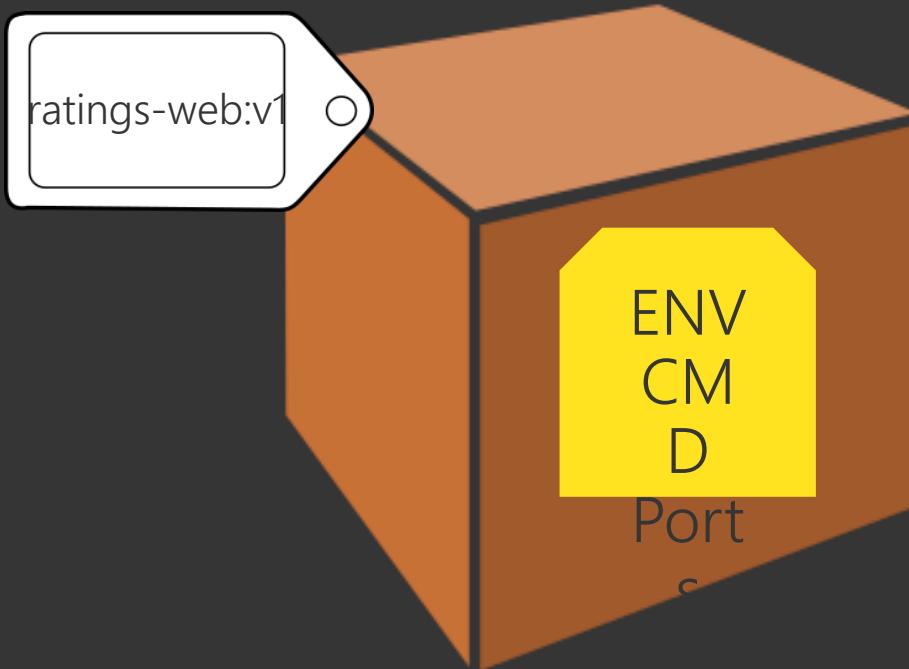


/lib2

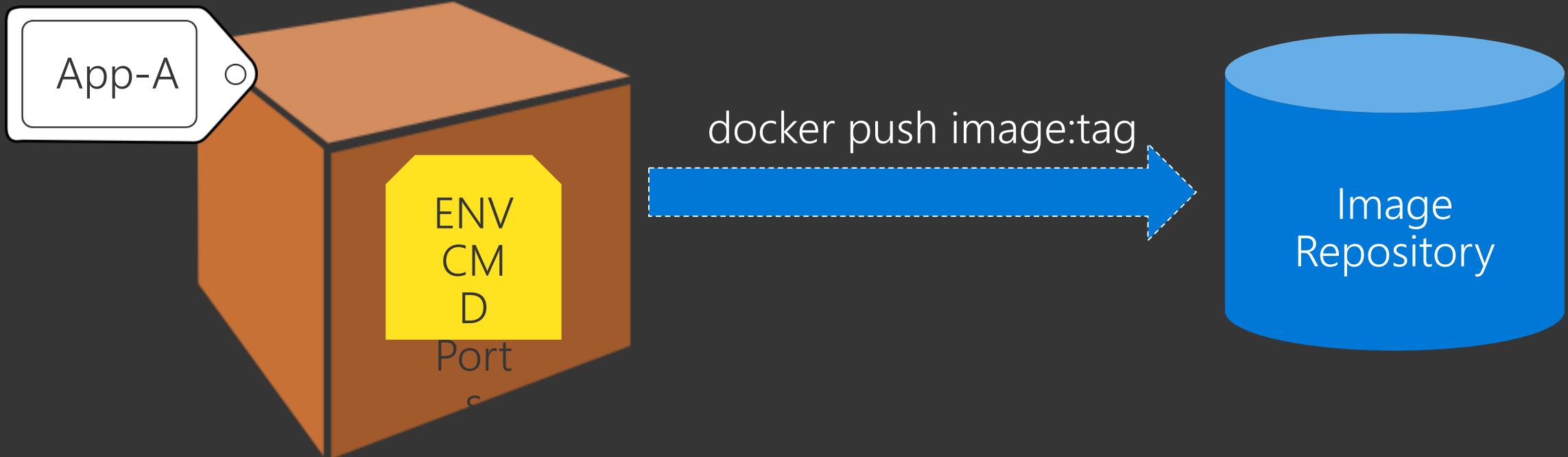


/config

Package Container



Push Container Image



Azure Container Support

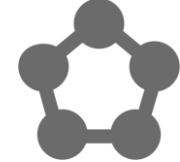
Pivotal
Cloud Foundry



Azure Container Instance



Azure Kubernetes Service
(AKS)



Service Fabric



Web Apps



Batch



Azure Kubernetes
Service (AKS)



Azure Container
Instances (ACI)



Azure Container
Registry



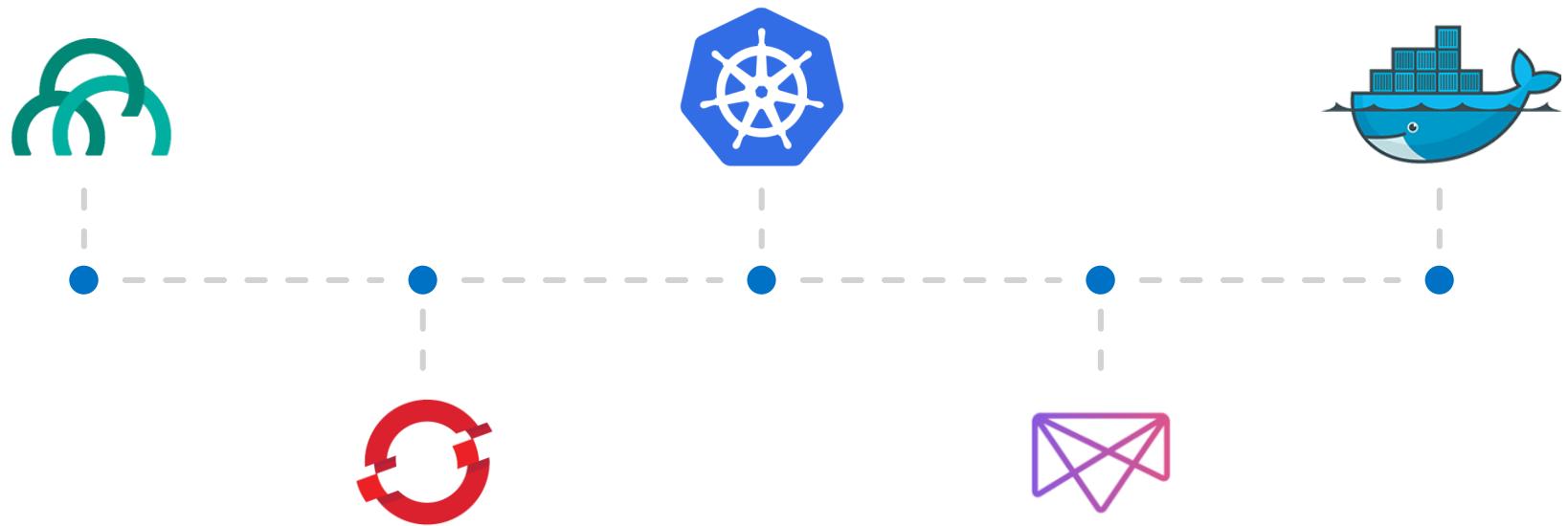
Open Service
Broker API (OSBA)



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

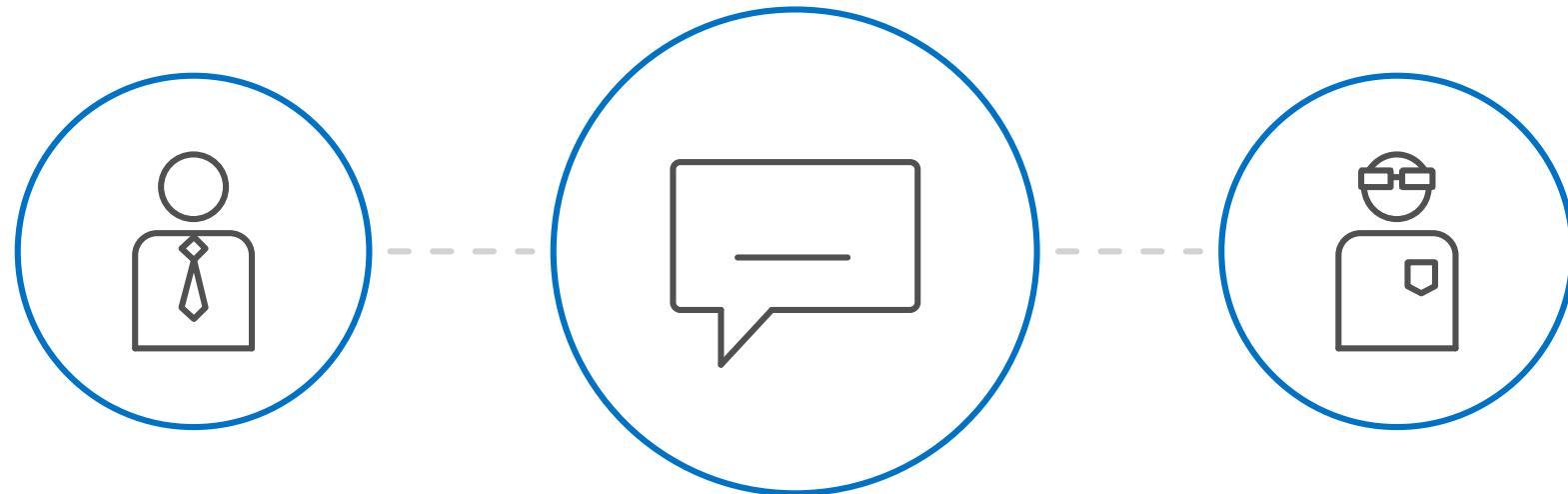


Open Service
Broker API (OSBA)



Release
Automation Tools

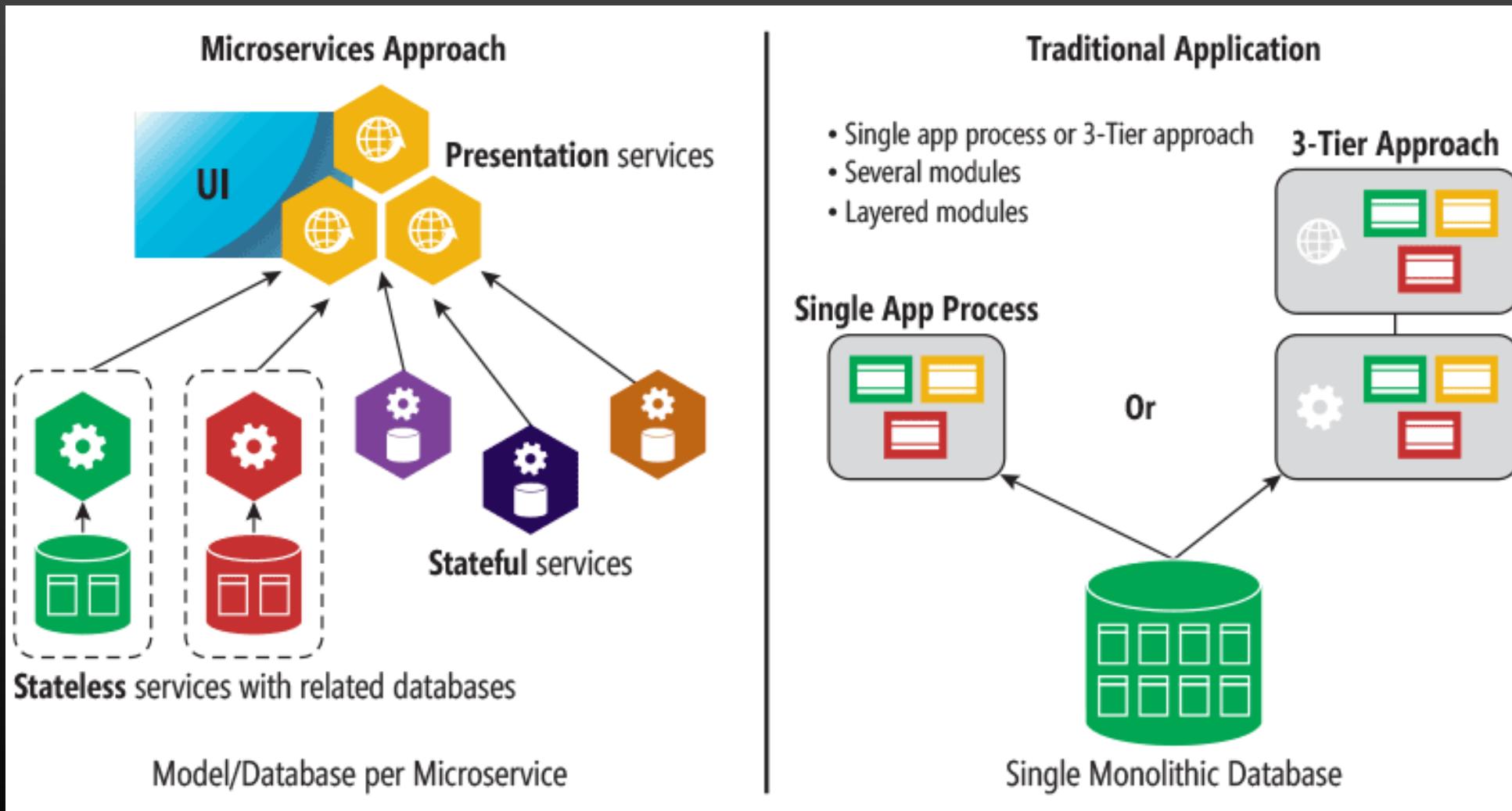
For customers without a preferred container platform...



**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

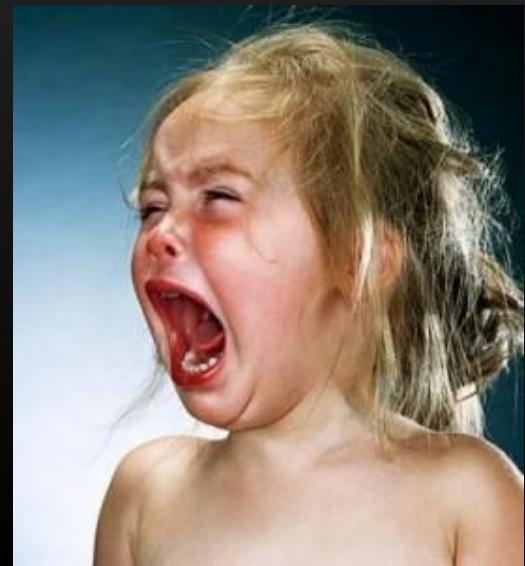
containers are an implementation
detail that often helps

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



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 Releaseld (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

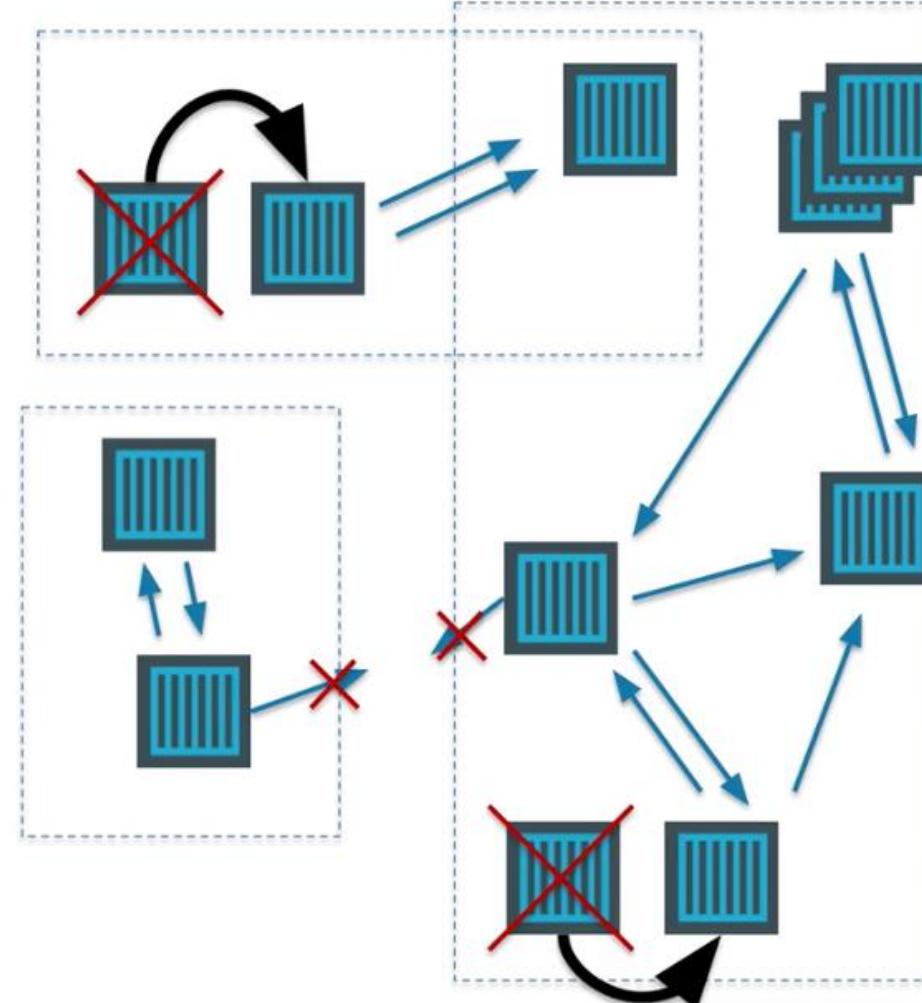
- Allow for multiple virtual clusters backed by the same physical cluster
- Logical separation
- Namespace used in FQDN of Kubernetes services
 - Eg - <service-name>.<namespace-name>.svc.cluster.local
- Every Kubernetes resource type is scoped to a namespace (except for nodes, persistentVolumes, etc.)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

Role Based Access Control

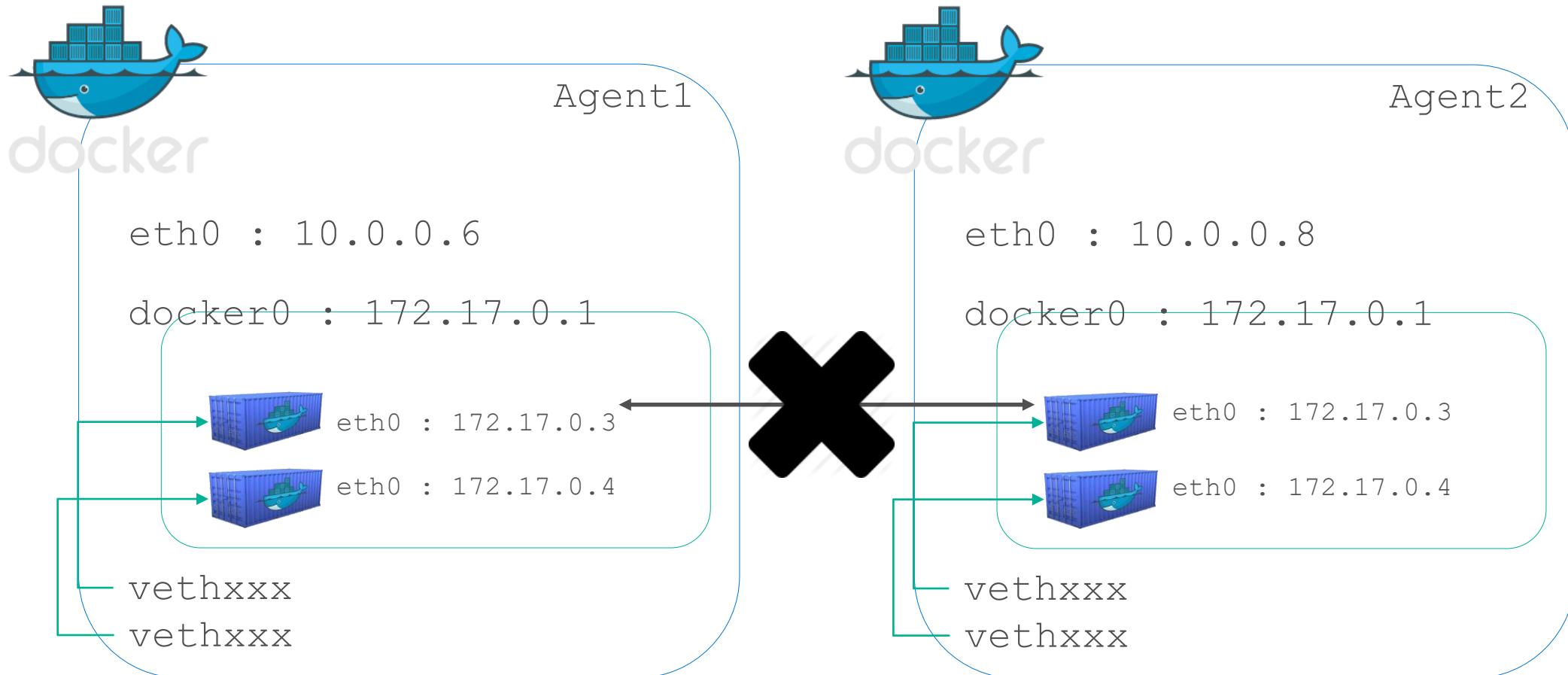
- Allows for dynamic policies against k8s API for authZ
- In version 1.6, this is a beta feature
- Can create both "Roles" and "ClusterRoles"
- Can configure api-server and kubectl to use Azure AD for access
- To grant permissions:
 - Use "RoleBindings" for within namespaces
 - Use "ClusterRoleBindings" 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" ]
```

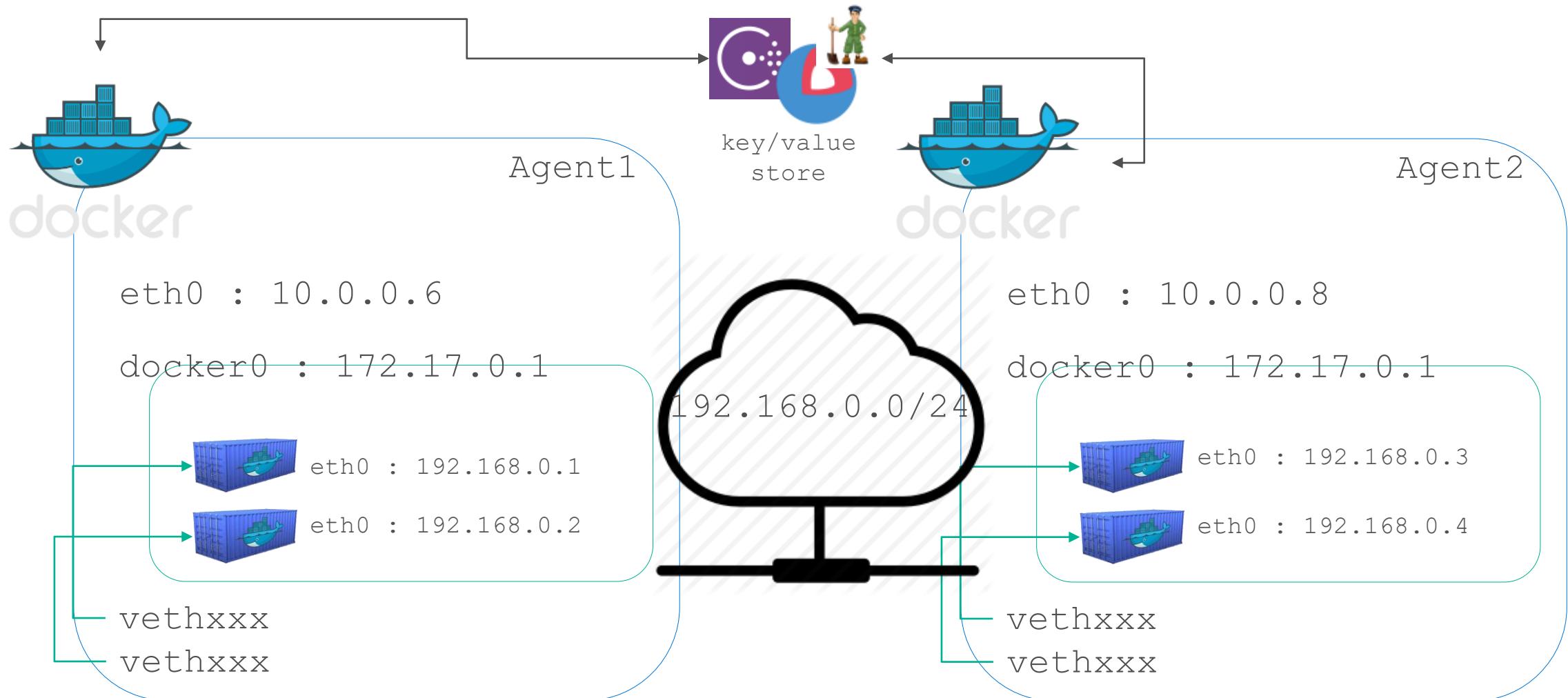
Networking in the Container World



Multi-Host Networking



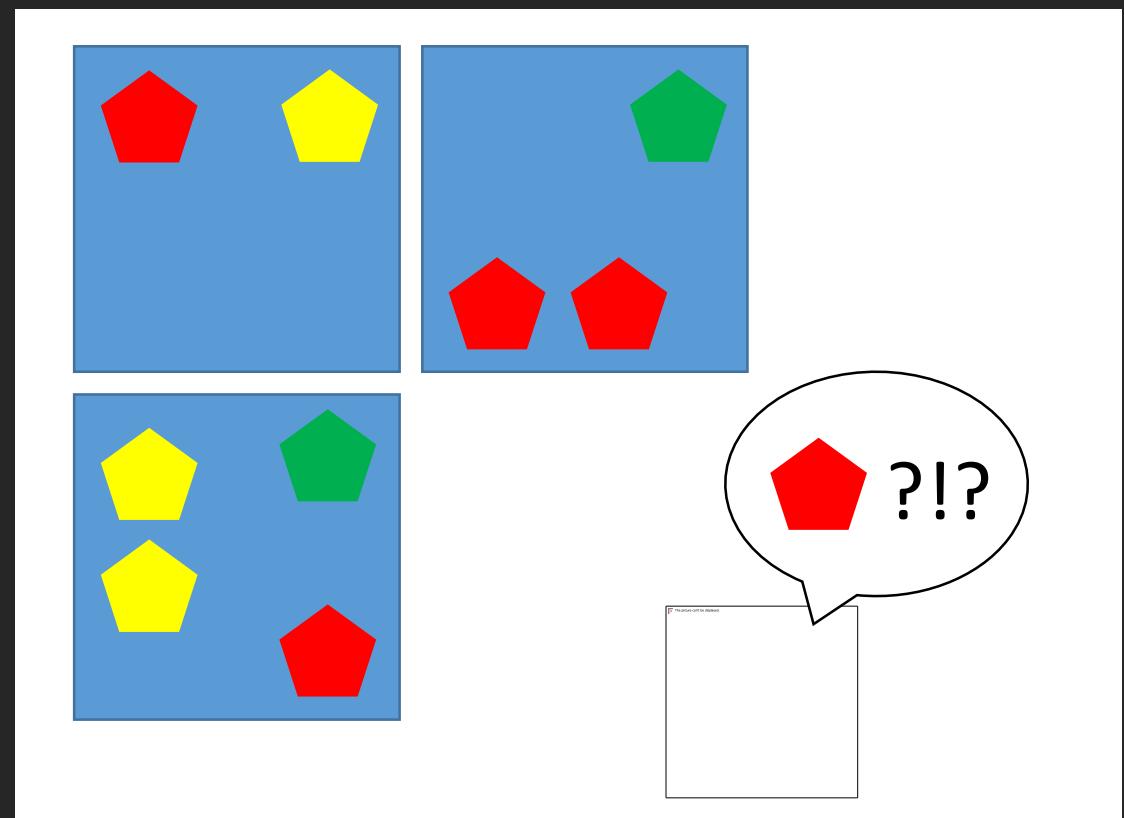
Multi-Host Networking



What is service discovery?

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

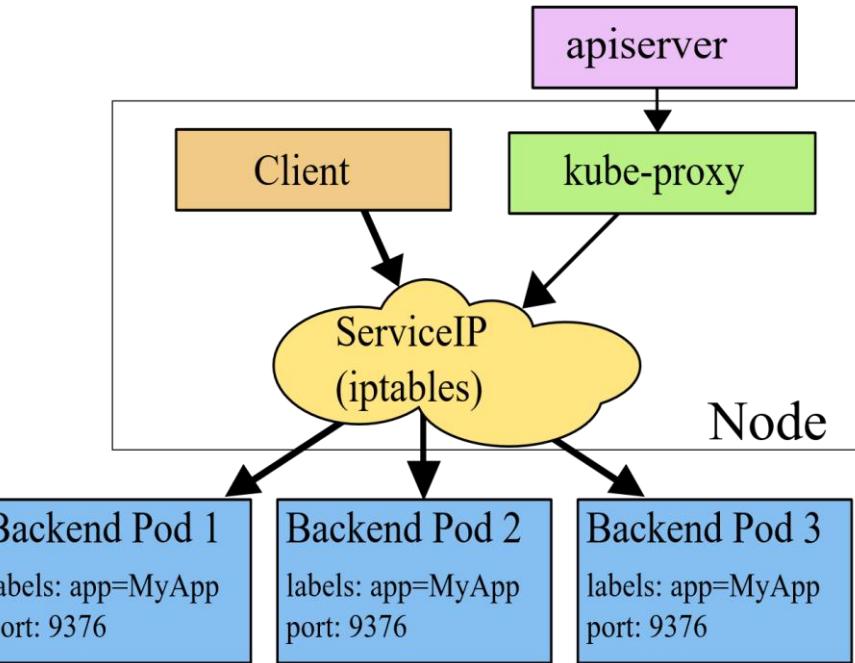
- Service Registry/Naming Service
- Service Announcement
- Lookup/Discovery
- Load Balancing



Networking in Kubernetes

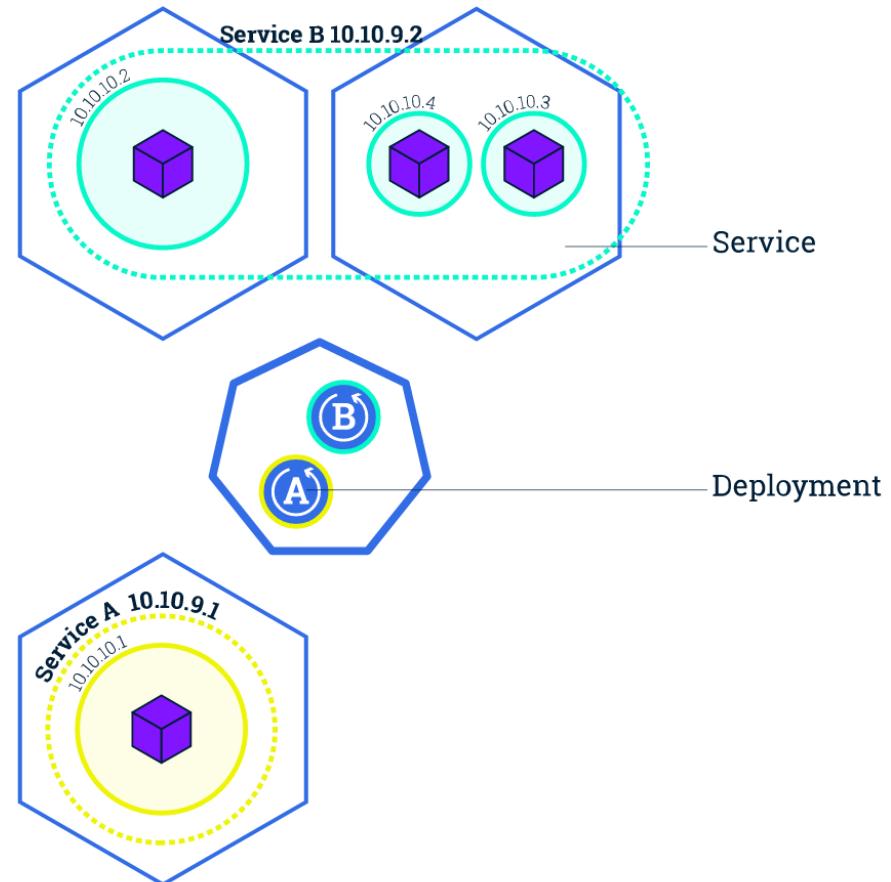
Kubernetes Networking model introduces 3 methods of communications:

- 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 kube-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