Microsoft

Microsoft Cloud Workshop

Containers and DevOps

Ben Griffin
Azure Technical Specialist, OSS
https://www.linkedin.com/in/bengriffin1/
ben.griffin@microsoft.com
@benitogriffin

Agenda

Time	Topic
9:00 – 10:00	Topic Introduction – Containers & DevOps
10:00 – 10:30	Architectural Design Session – Case Study
10:30 – 10:45	Morning Tea
10:45 – 12:30	Architectural Design Session – Continue & Present
12:30 – 1:00	Lunch
1:00 – 1:30	Kiubernetes Primer
1:30 – 2:30	Hands On Lab
2:30 – 2:45	Working - Afternoon Tea
4:00	Share Learnings
4:30	Q&A, wrap-up, and next steps

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

In reality, they shouldn’t…

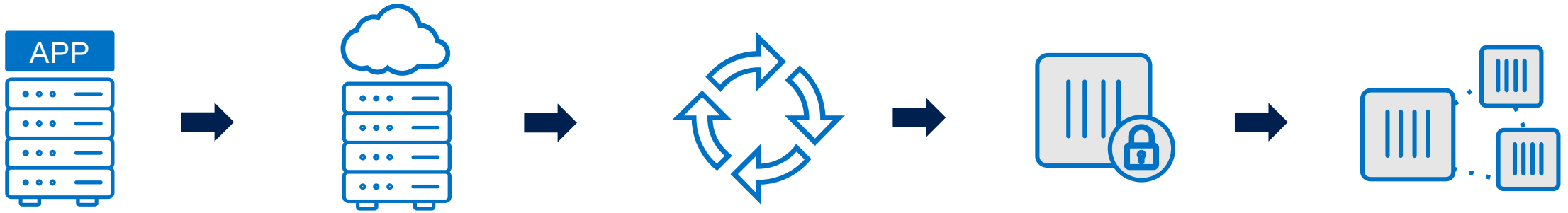
They do care about cloud native applications

"Unlimited" Scale

Global reach

Rapid innovation -> time to market

From traditional app to modern app



Existing Application

Modern Infrastructure

Move to the cloud as VMs or Containers or refresh HW.

Modern Methodologies

Implement CI/CD and automation.

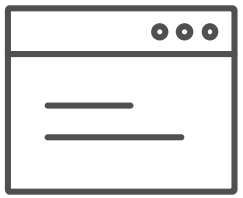
Containerize Applications

Re-architect apps for scale with containers.

Modern Microservices

Add new services or start peeling off services from monolithic code.

What we hear from **developers**



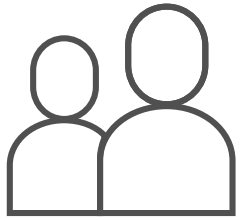
I need to create applications at a competitive rate without worrying about IT

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

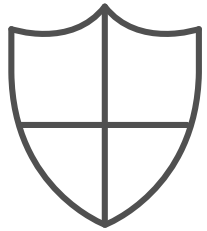
My productivity and application innovation become suspended when I have to wait on IT

What we hear from **IT**

I need to manage servers and maintain compliance with little disruption



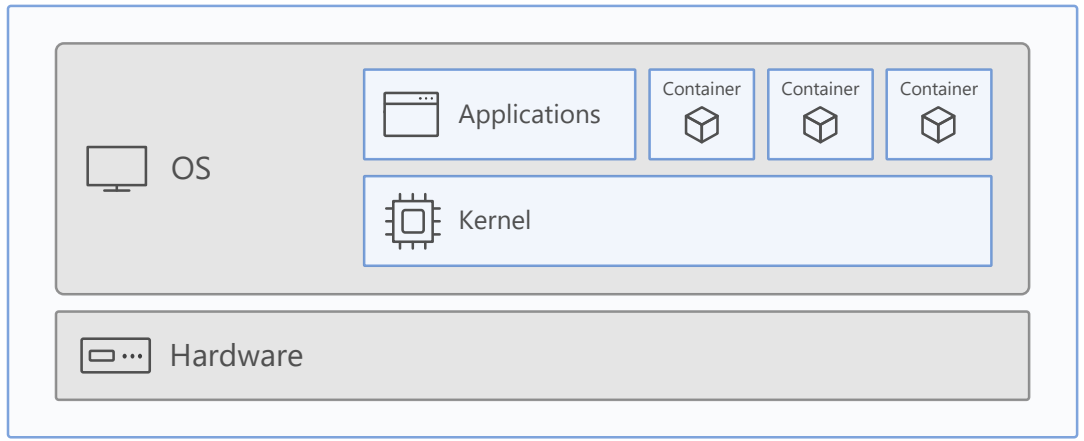
I'm unsure of how to integrate unfamiliar applications, and I require help from developers



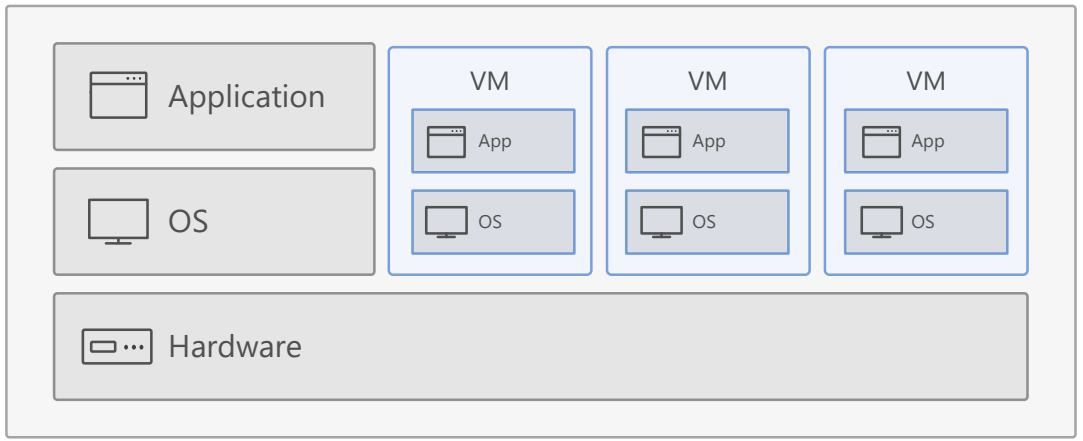
I'm unable to focus on both server protection and application compliance

What is a **container**?

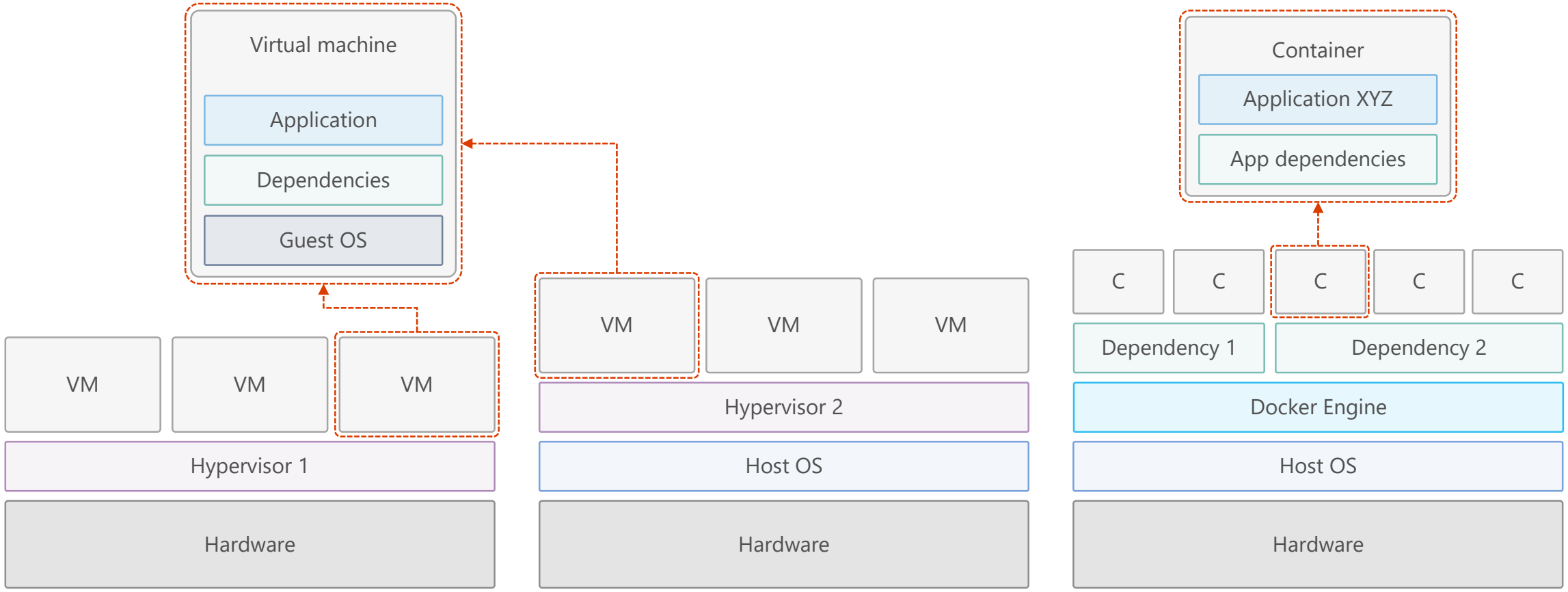
Containers = operating system virtualization



Traditional virtual machines = hardware virtualization



Virtualization versus **containerization**

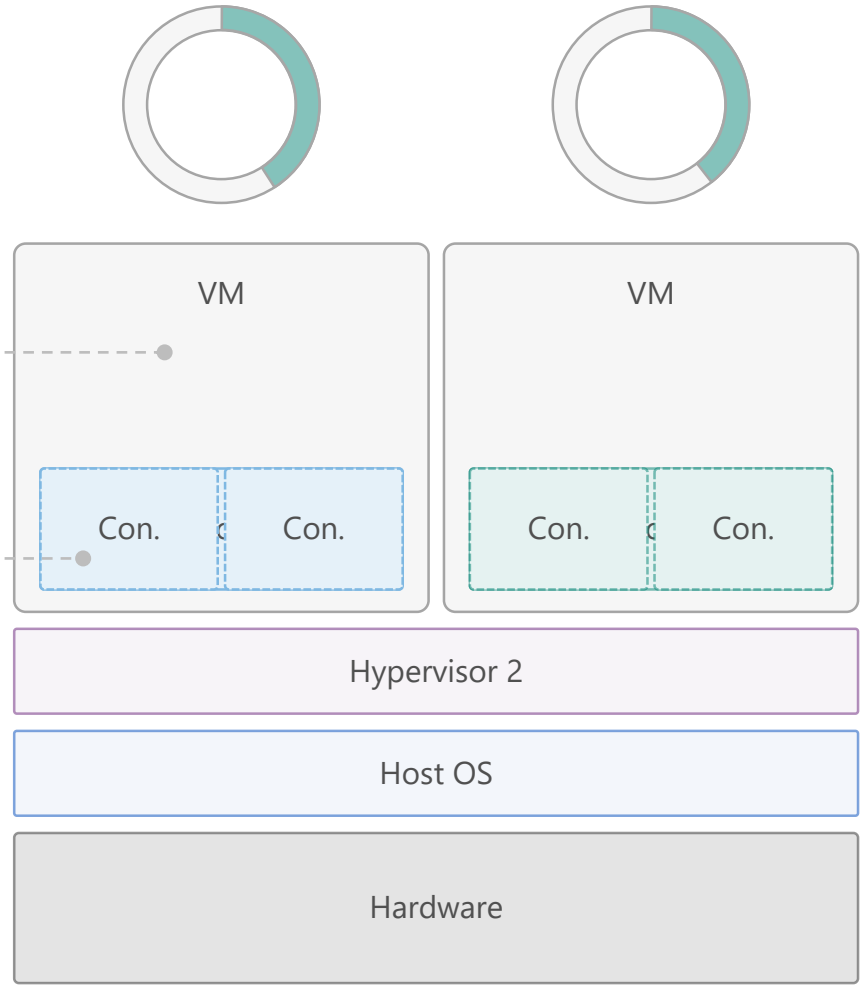


The container **advantage**

Traditional virtualized environment

Low utilization of container resources

Containerization of applications and their dependencies

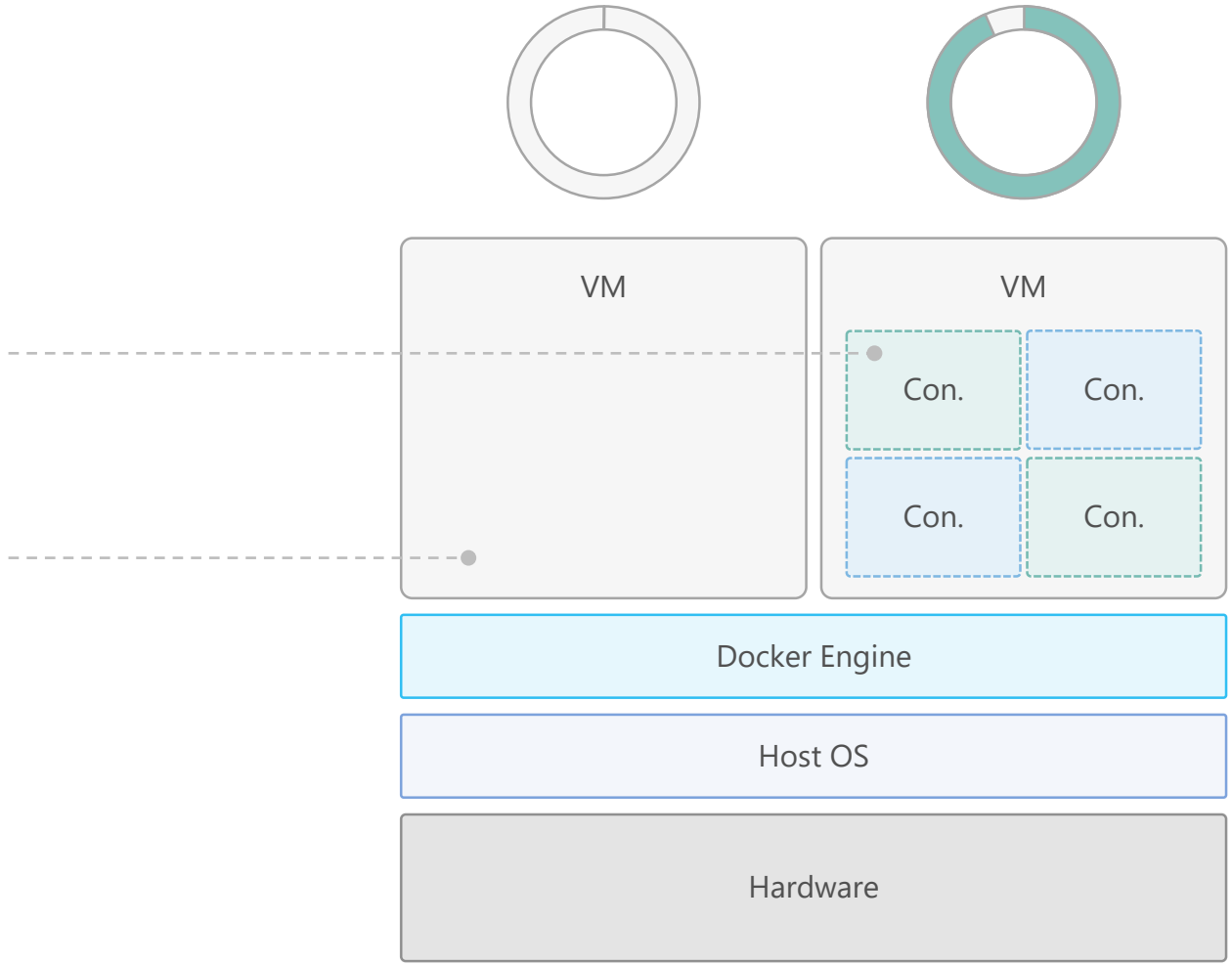


The container **advantage**

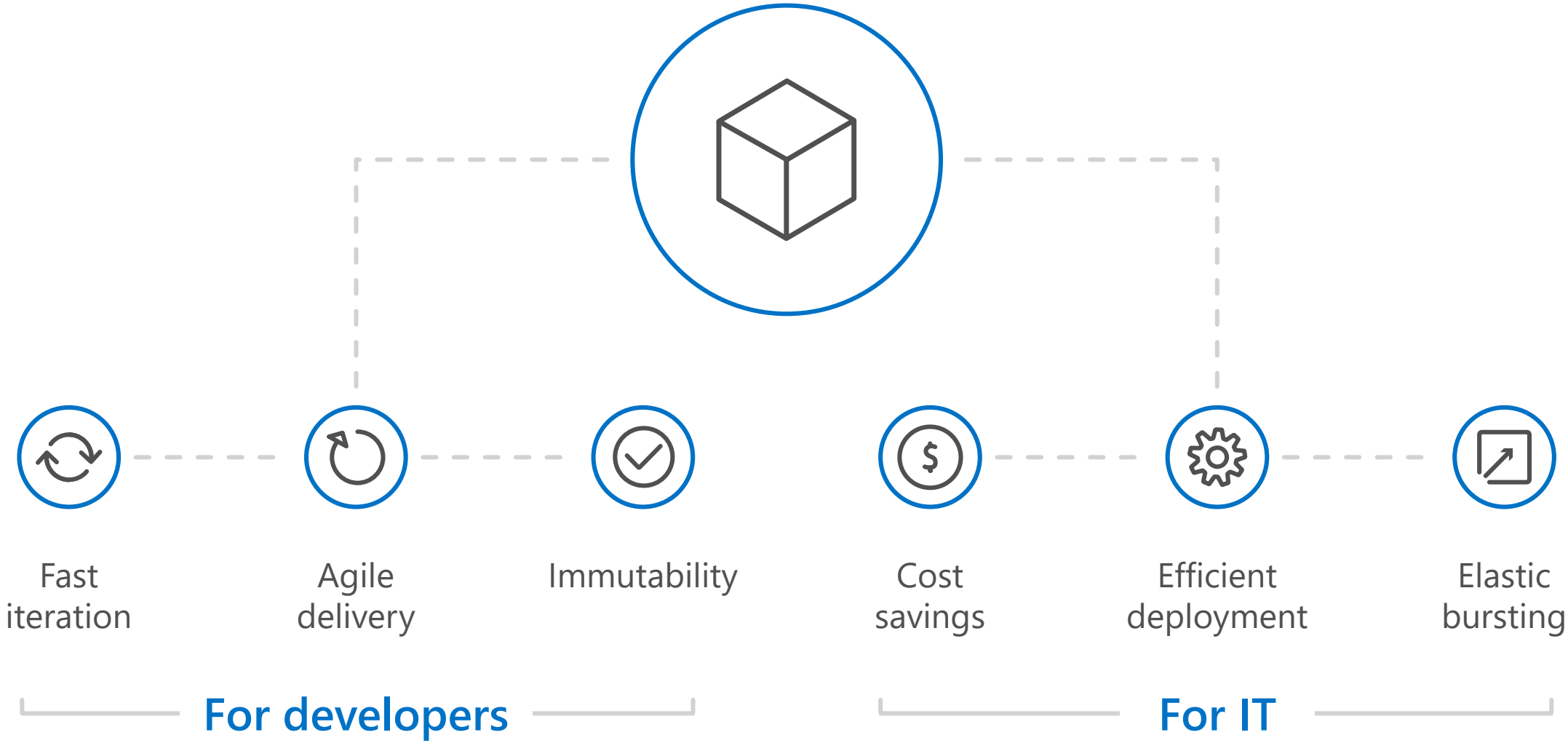
Containerized environment

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

Decommission unused resources for efficiency gains and cost savings



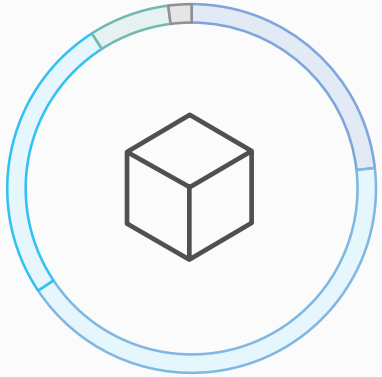
The container **advantage**



Containers are gaining **momentum**

Does your organization currently use container technologies?[1]

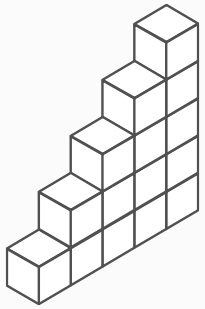
- 23% My org. is **evaluating** container technologies
- 42% **Yes**, my org. currently uses container technologies
- 25% **No**, my org. is not using container technologies
- 7% **Not sure**
- 2% **Not applicable**



Larger companies are leading adoption.[2]

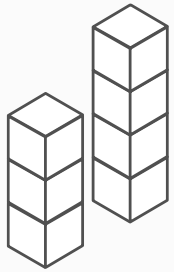
Nearly **60% percent** of organizations running 500 or more hosts are classified as **container dabblers** or adopters.

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



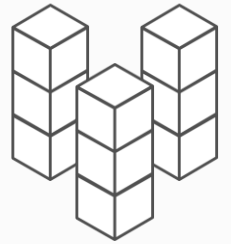
5X

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



7X

Containers churn 9 times FASTER than VMs.[1]

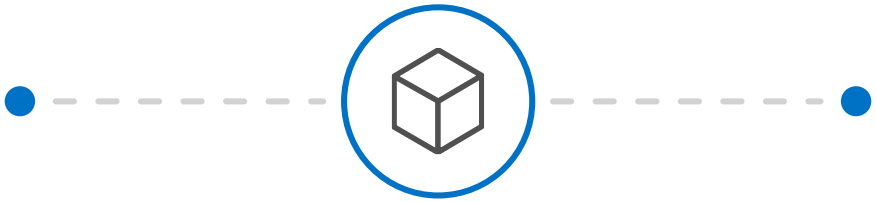


9X

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

Kubernetes: the de-facto orchestrator

Portable

Public, private, hybrid, multi-cloud

Extensible

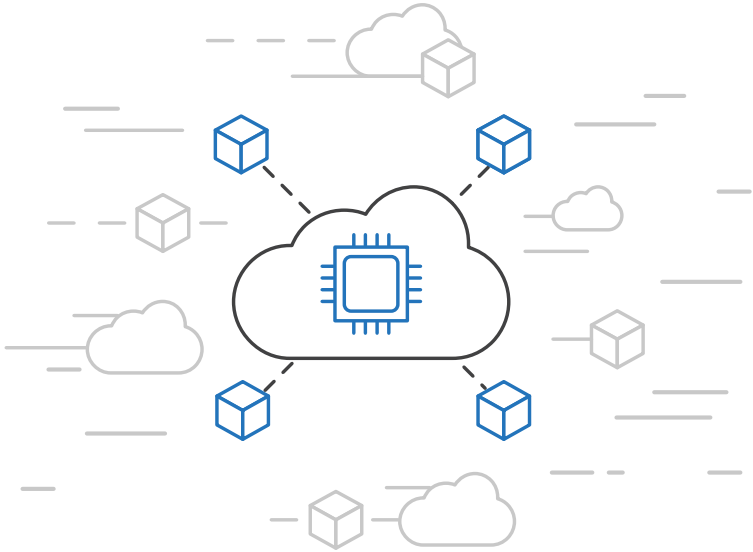
Modular, pluggable, hookable, composable

Self-healing

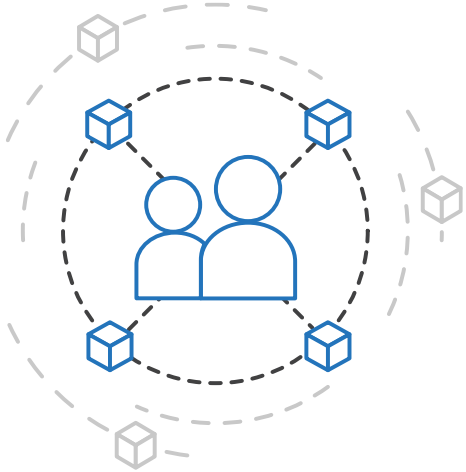
Auto-placement, auto-restart, auto-replication, auto-scaling

Azure container **strategy**

Embrace containers as ubiquitous

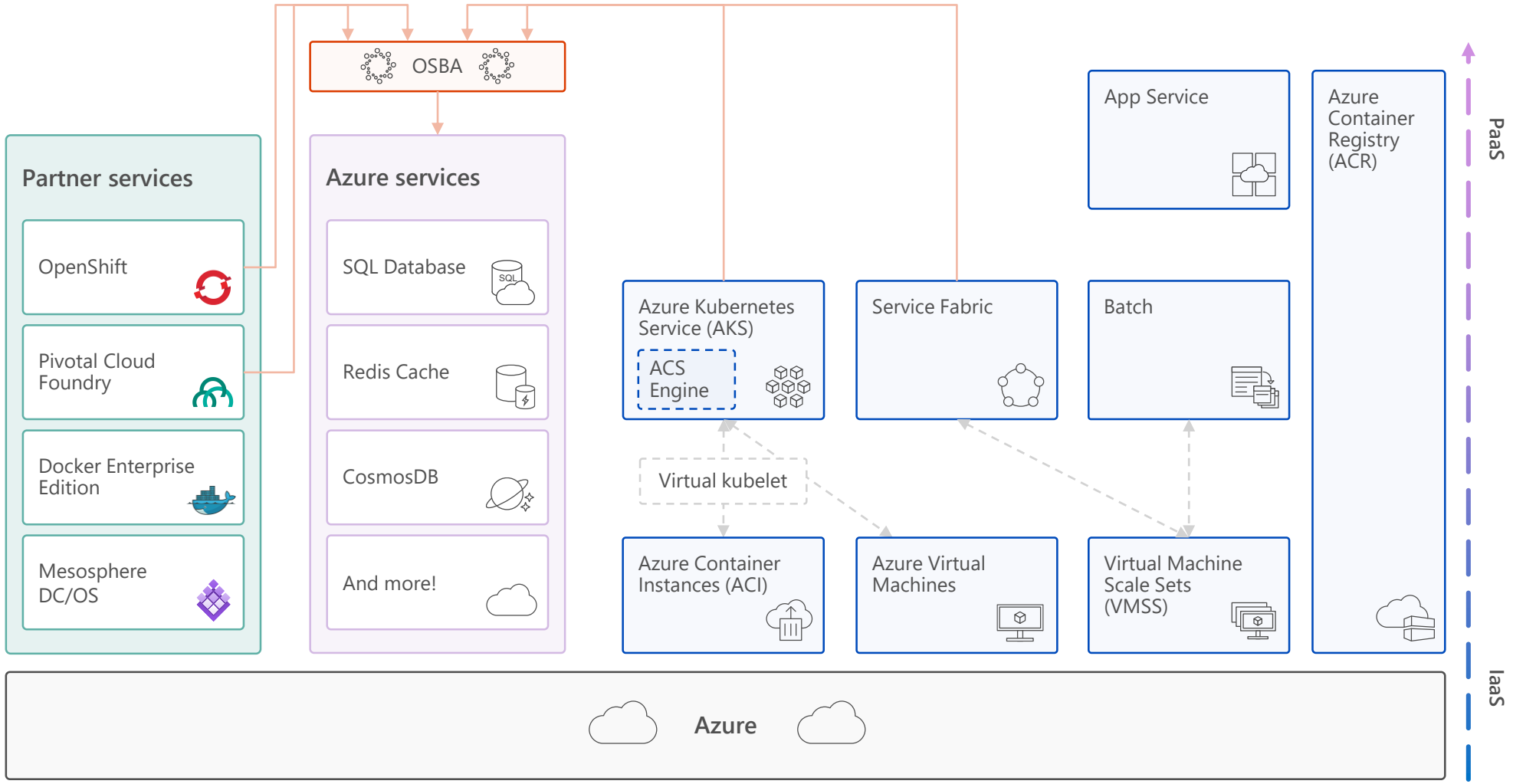


Support containers across the compute portfolio



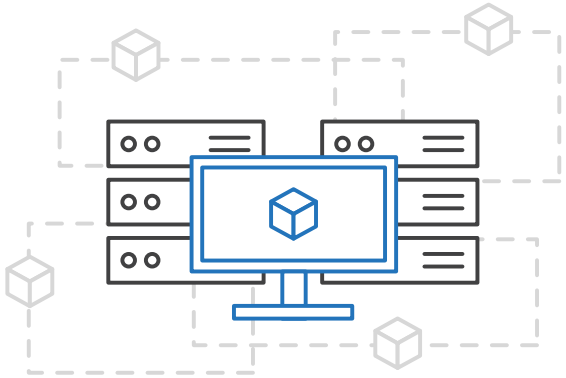
Democratize container technology

Azure container **ecosystem**



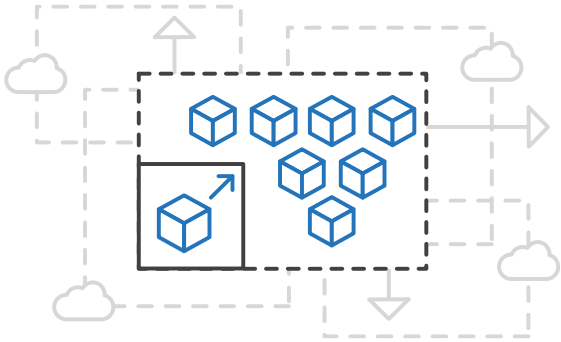
Azure Kubernetes Service (AKS)

Simplify the deployment, management, and operations of Kubernetes



Focus on your containers not the infrastructure

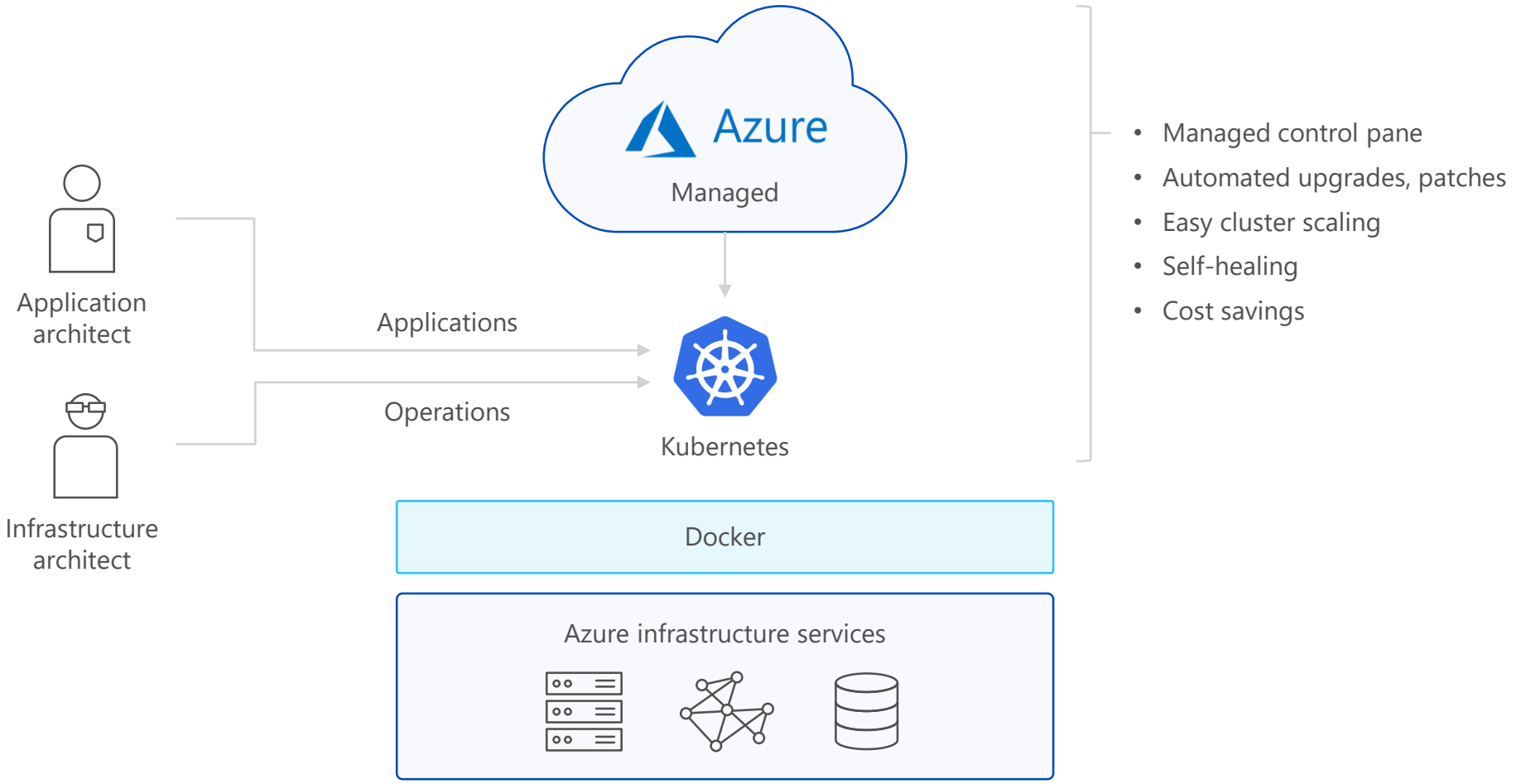
Work how you want with open-source APIs



Scale and run applications with confidence

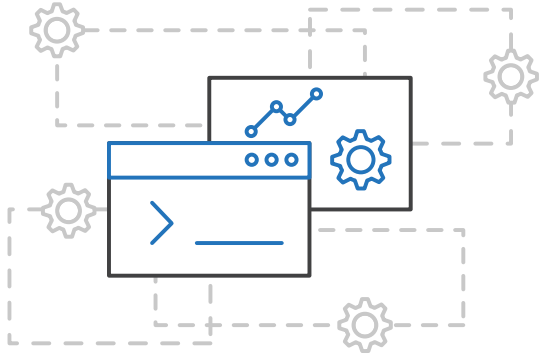
Azure Kubernetes Service (AKS)

A fully managed Kubernetes cluster

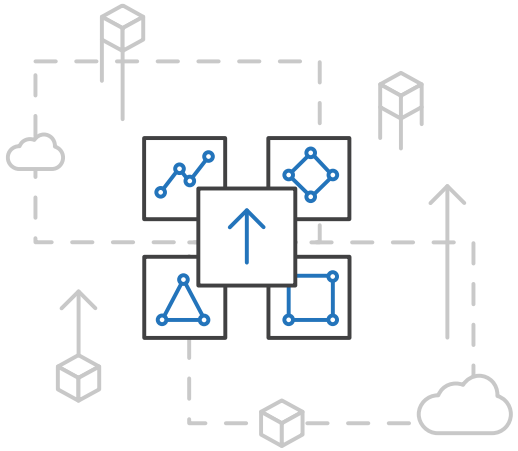


ACS Engine

Customized Kubernetes on Azure



A proving ground
for new features

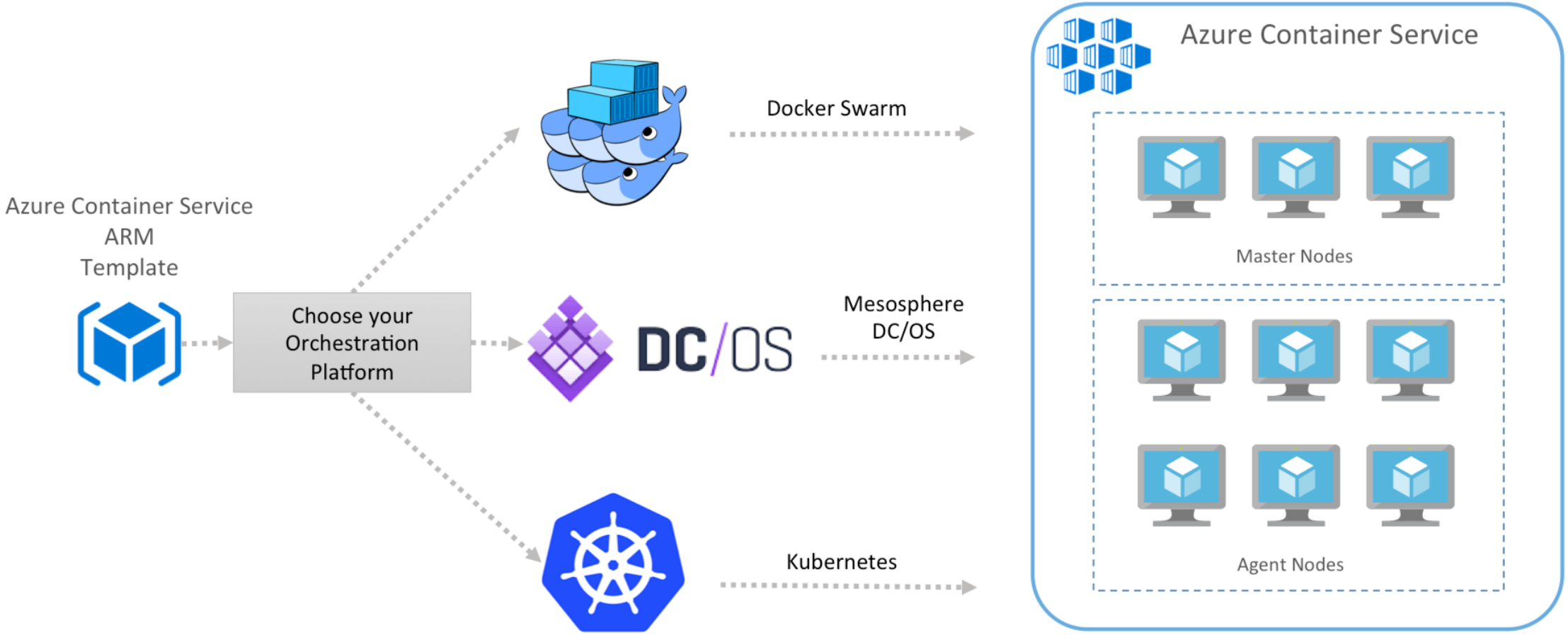


Enables custom
deployments

Available
on GitHub

Azure Container Service

Freedom to choose your orchestrator on Azure



Azure Container Instance (ACI)

Containers as a core Azure resource

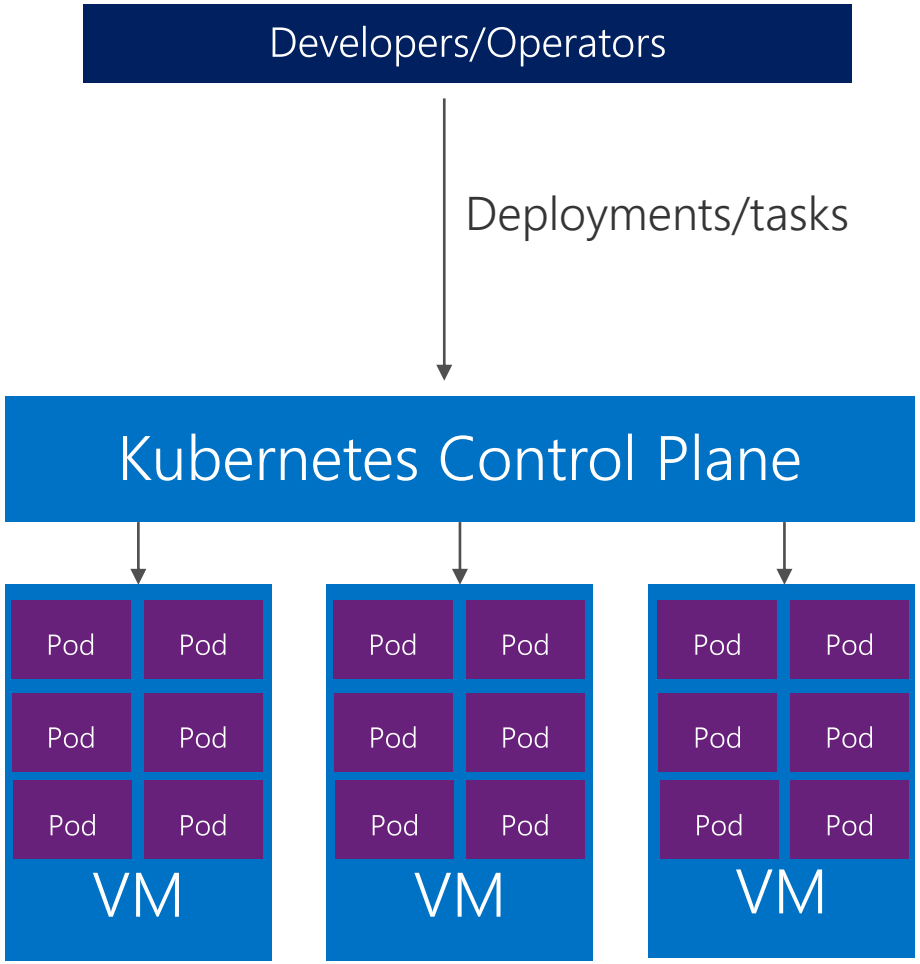
Fastest and easiest way to run a container in the cloud

No VM management

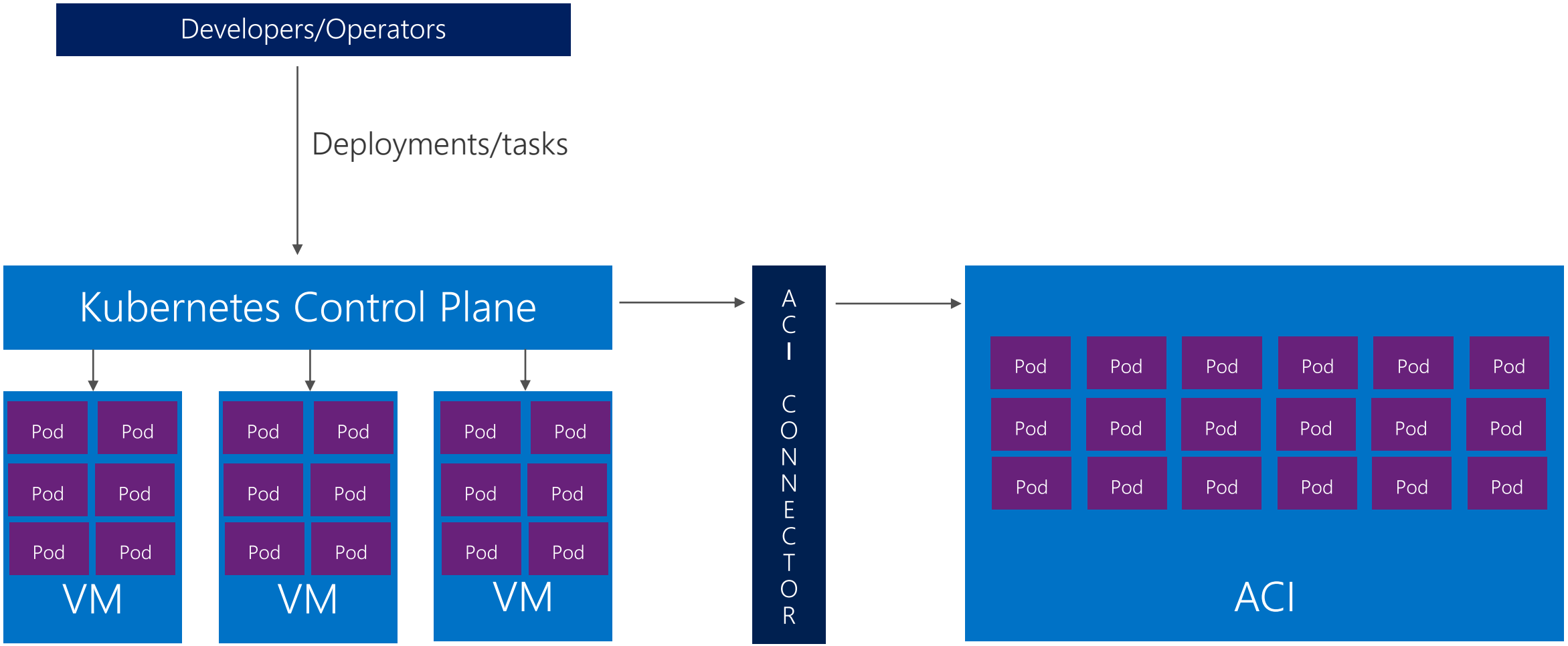
Per-second billing based on resource requirements (CPU + Memory)

Deploy images from DockerHub, Azure Container Registry, or any other Docker registry

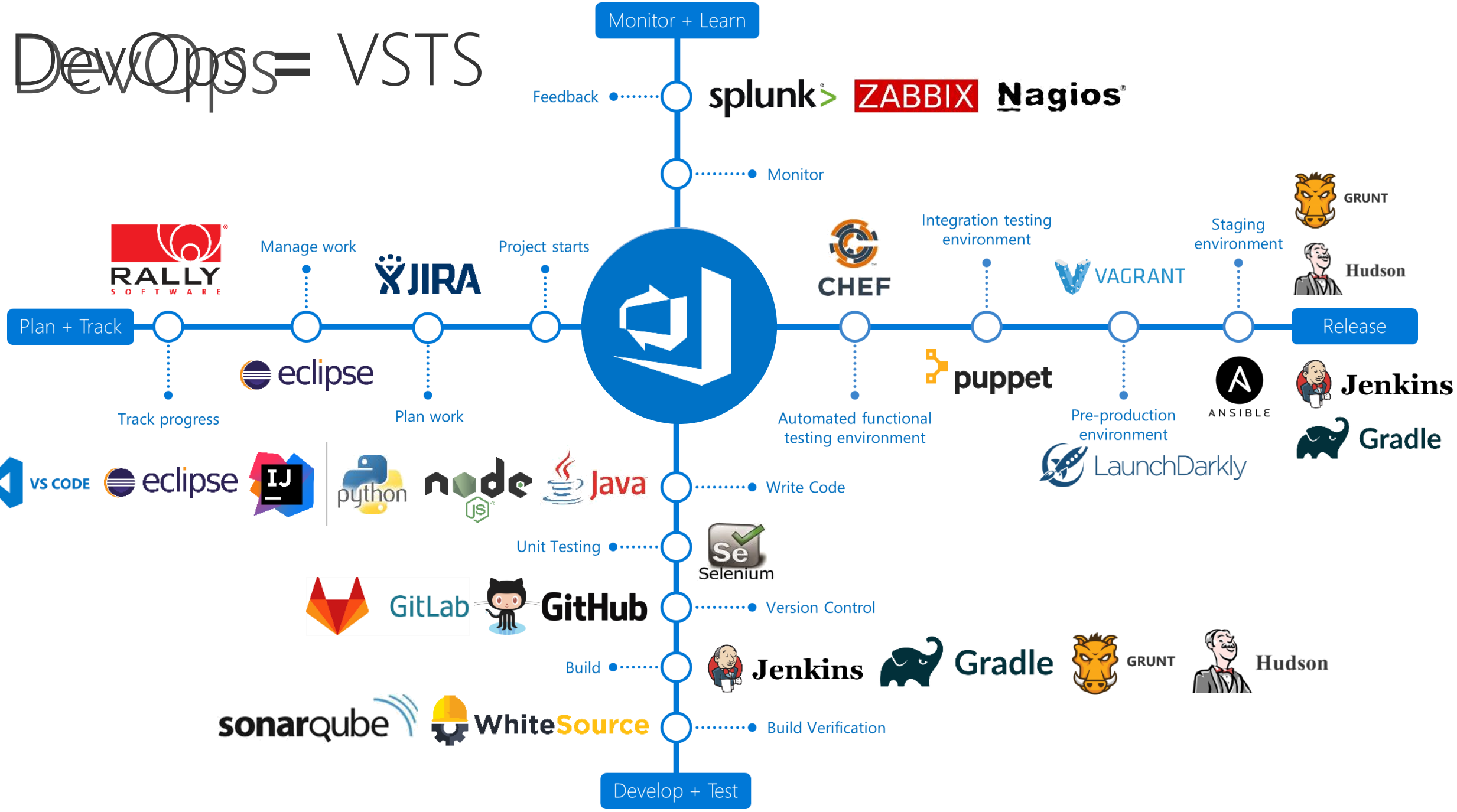
Bursting with the ACI Connector



Bursting with the ACI Connector

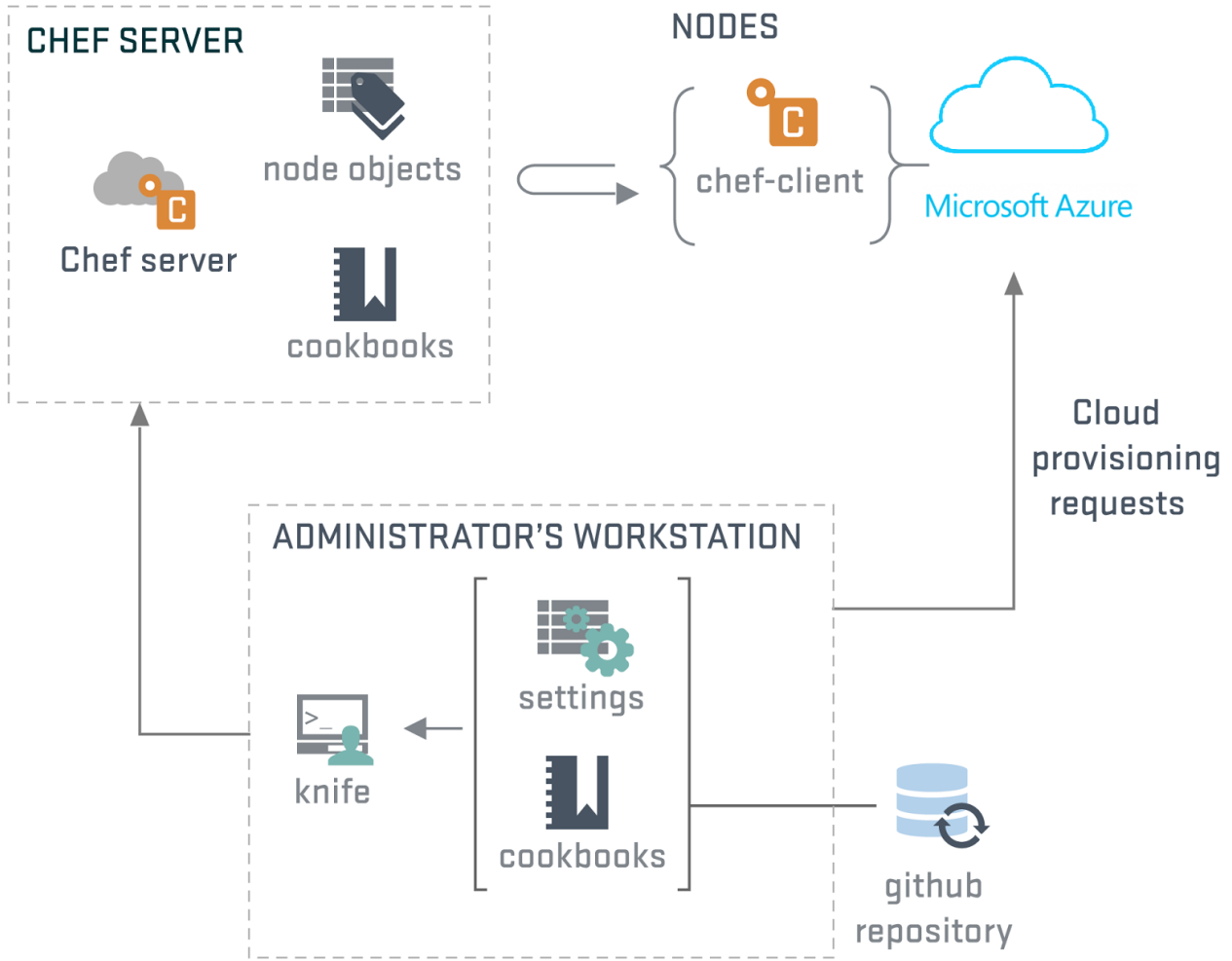


DevOps = VSTS


Monitor + Learn
Feedback
splunk>
ZABBIX
Nagios
Monitor
RALLY SOFTWARE
Manage work
JIRA
Project starts
CHEF
Integration testing environment
VAGRANT
Staging environment
GRUNT
Hudson
Plan + Track
Release
eclipse
puppet
ANSIBLE
Jenkins
Track progress
Plan work
Automated functional testing environment
Pre-production environment
LaunchDarkly
Gradle
VS CODE
eclipse
IJ
python
node
Java
Write Code
Unit Testing
Selenium
GitLab
GitHub
Version Control
Build
Jenkins
Gradle
GRUNT
Hudson
sonarqube
WhiteSource
Build Verification
Develop + Test

Common Scenarios

Chef Automation on Azure



Step 1: Review the customer case study

Outcome

Analyze your customer needs

Timeframe

15 minutes

Customer situation

Fabrikam Medical Conferences provides conference website services tailored to the medical community.

After starting with a few small conferences, they now have evolved into a well-known brand and handle over 100 conferences per year and growing.

Customer situation

- Each conference site has a limited budget, but the conference owners have significant customization and change demands.

- These changes can impact various aspects of the system from UI through to backend including conference registration and payment terms.

Customer situation

- 12 developers handle
  - Development
  - Testing
  - Deployment
  - Operational management of all customer sites

- Due to customer demands, they have issues with the efficiency and reliability of their development and DevOps workflows.

Customer situation

- The technology used is the MEAN stack
  - Mongo, Express, Angular, Node.js

  - Conference sites are currently hosted in Azure
  - Web applications and APIs hosted in Azure App Services
  - MongoDB is a managed service provided by MongoLabs on Azure

Customer situation

- Fabrikam considers conference owners (“customers”) as “tenants,” and treats each tenant as a unique deployment including:
  - A database in the MongoDB cluster with its own collections
  - A copy of the most recent functional conference code base is taken and configured to point at the tenant database
- They make modifications to support the customer’s needs.
  - Fabrikam deploys the tenant’s code to the App Service Plan (VM)
  - Once the conference site is live, the inevitable requests for customizations to the deployment begins

Customer situation

- They are looking to achieve the following:
  - Reduce potential regressions introduced to functional tenant code resulting from changes
  - Ideally, changes to individual areas should not require a full regression test of the site functionality
  - Reduce the time to onboard new tenants
  - Reduce overhead managing changes, and related deployments
  - Improve ability to roll back and recover post change
  - Increase visibility into system operations and health

Customer situation

These are the key challenges we want to resolve:

- Reduce potential for regressions when making changes to each tenant code base.
- Reduce the coverage required as new features are rolled out in different areas.
- Reduce the time to onboard new tenants.
- Reduce overhead managing changes and related deployments.
- Improve ability to roll back or forward quickly.
- Improve visibility into overall operations and health.

Customer needs

- Simplify new tenant deployment.
- Improve reliability of tenant updates.
- Choose a suitable Docker container strategy on Azure.
- Continue to use MongoDB for data storage.

Customer needs

- Continue to use Git repositories for source control.
- Look at Chef as the CICD tool of choice.
- Use tools for deployment, CICD integration, container scheduling, orchestration, monitoring, and alerts.
- They wish to complete an implementation of the proposed solution for a single tenant to train the team and perfect the process.

Customer objections

- With so many platforms and tools for Docker and container orchestration, how should we choose an option for Azure?
- What is the simplest way to move containers on Azure, based on our PaaS experience, while at the same time considering our scale and growth requirements?

Step 2: Design the solution

Outcome

Design a solution and prepare to present the solution to the target customer audience in a 15-minute chalk-talk format.

Timeframe

60 minutes

Business needs (10 minutes)	• Respond to questions outlined in your guide and list the answers on a flipchart.
Design (35 minutes)	• Design a solution for as many of the stated requirements as time allows. Show the solution on a flipchart.
Prepare (15 minutes)	• Identify any customer needs that are not addressed with the proposed solution. • Identify the benefits of your solution. • Determine how you will respond to the customer's objections. • Prepare for a 15-minute presentation to the customer.

Step 3: Present the solution

Outcome

Present a solution to the target customer in a 15-minute chalk-talk format

Timeframe

30 minutes (15 minutes for each team to present and receive feedback)

Directions

- Pair with another table.
- One table is the Microsoft team and the other table is the customer.
- The Microsoft team presents their proposed solution to the customer.
- The customer asks one of the objections from the list of objections in the case study.
- The Microsoft team responds to the objection.
- The customer team gives feedback to the Microsoft team.

Wrap-up

Outcome

- Identify the preferred solution for the case study.
- Identify solutions designed by other teams.

Timeframe

15 minutes

Preferred target audience

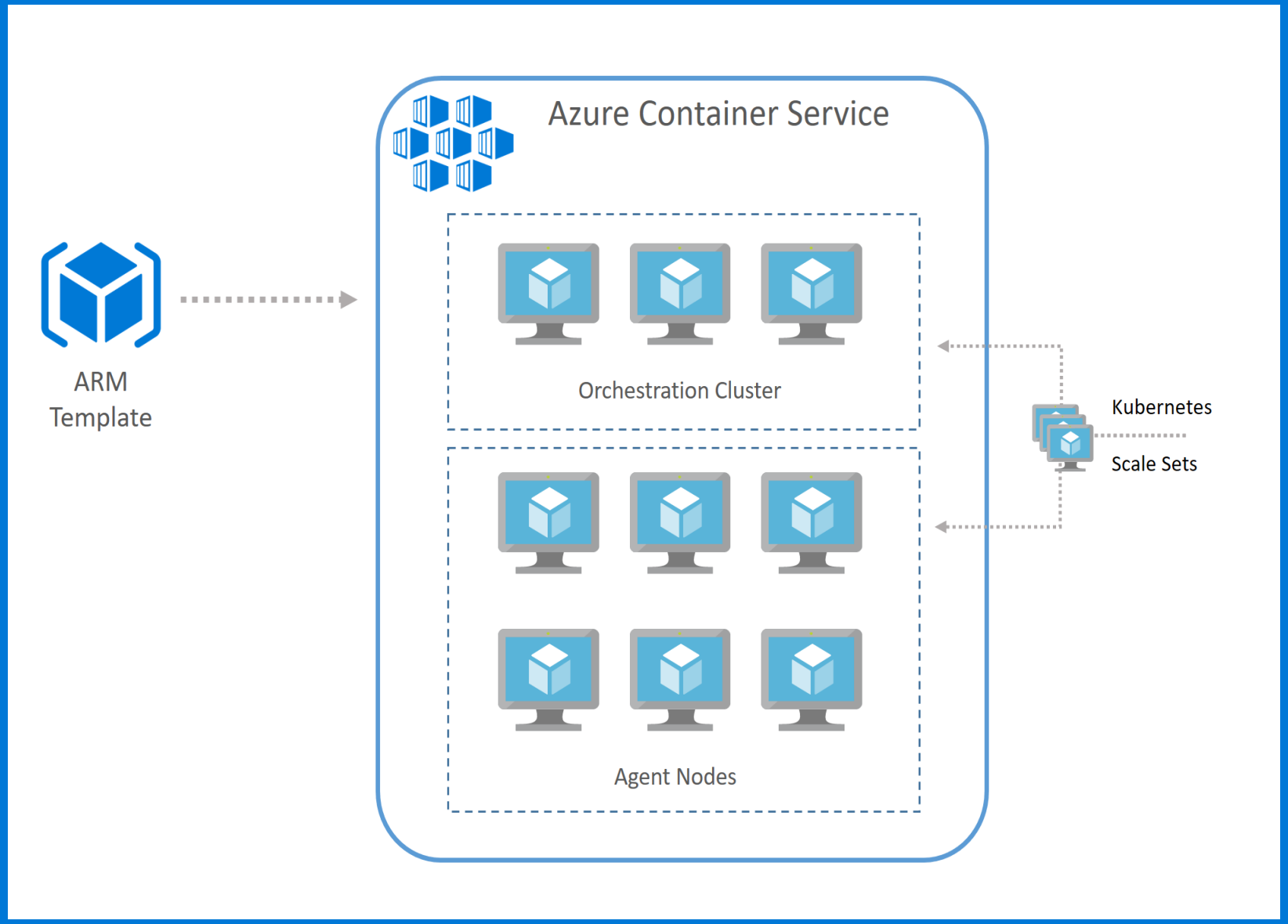
- Arthur Block, VP Engineering at Fabrikam Medical Conferences
- The primary audience is the technical strategic decision-maker with influential solution architects, or lead technical personnel in development or operations.
- Usually we talk to the key architects, developers, and infrastructure managers who report to the CIO or equivalent, or to key solution sponsors or those who represent the business unit IT or developers who report to those sponsors.

Preferred solution

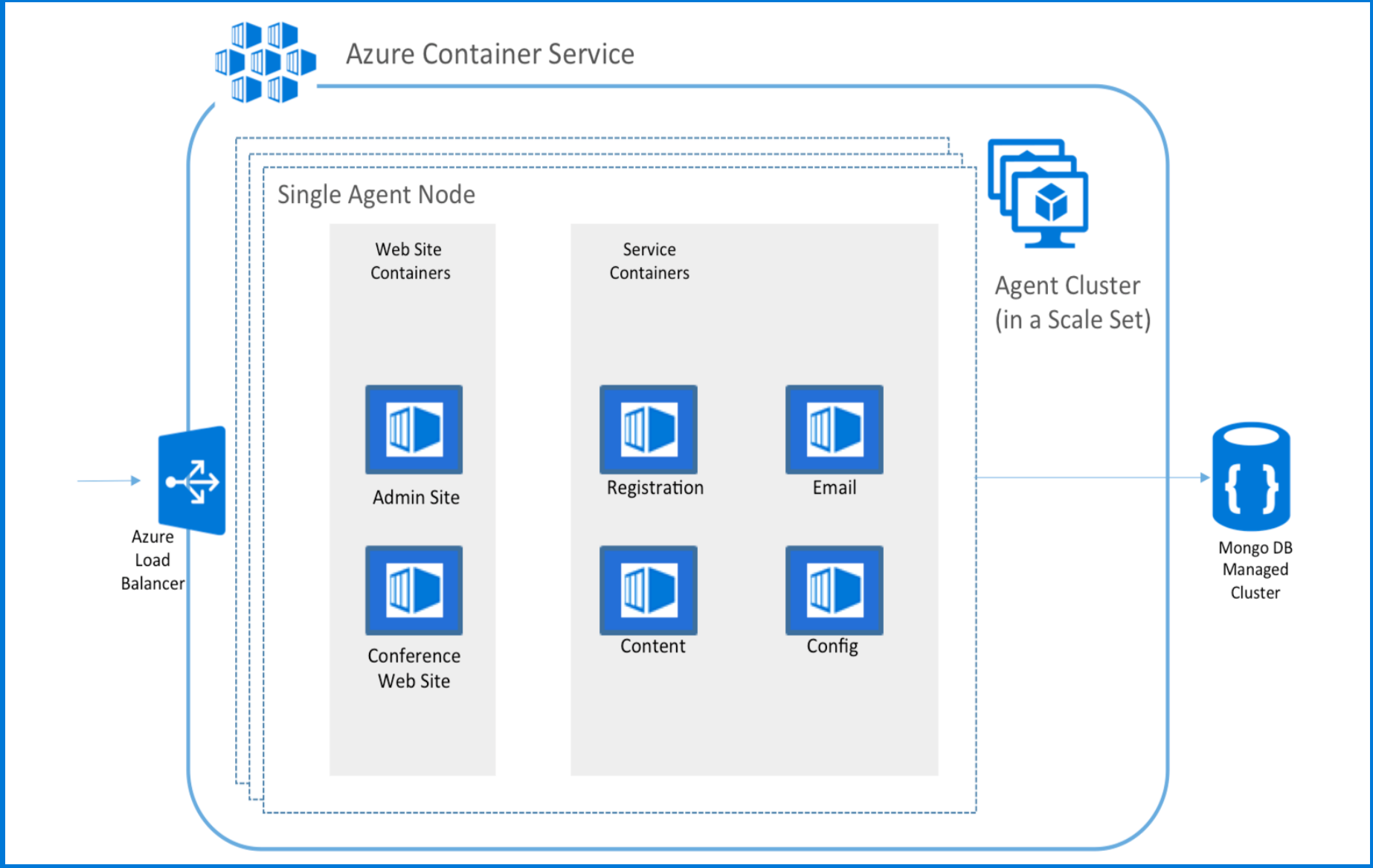
- After evaluating the options for container platforms on Azure, and discussing the investment in Azure Kubernetes Service (AKS) with the team at Microsoft, Fabrikam Medical Conferences decided to move forward with Azure Container Service with Kubernetes as a transitional step toward AKS.

- They also decided to move forward with Chef for infrastructure and container DevOps workflows.

Preferred solution



Preferred solution



Preferred objections handling

- *What is the simplest way to move to containers on Azure, based on our PaaS experience, while at the same time considering our scale and growth requirements?*

  - App Service for Containers – simple, PaaS without robust orchestration platform management tooling
  - Azure Container Instances – simple, isolated, without management tooling
  - Azure Kubernetes Service (AKS) – the ideal solution when out of preview, fully managed
  - Azure Container Service with Kubernetes – the best choice to later transition to AKS

Preferred objections handling

- *With so many platforms and tools for Docker and container orchestration, how should we choose an option for Azure?*

  - The best option is to go with a managed cluster such as AKS, native to Azure
  - Azure Container Service with Kubernetes will support a smooth transition to AKS while providing the needed features near term.

Customer quote

"With Azure Container Service and Kubernetes we feel confident we can make the move to a container-based platform with the right DevOps support in place to be successful with a small team.

In addition, it is an excellent way to transition our operations to Kubernetes on Azure with the long-term goal being managed container orchestration with AKS."

Arthur Block, VP of Engineering at Fabrikam Medical Conferences

Microsoft

Kubernetes Concepts

Kubernetes Architecture Components

api-server

etcd

controller-manager

scheduler

master components

* Azure Managed in AKS (Preview)

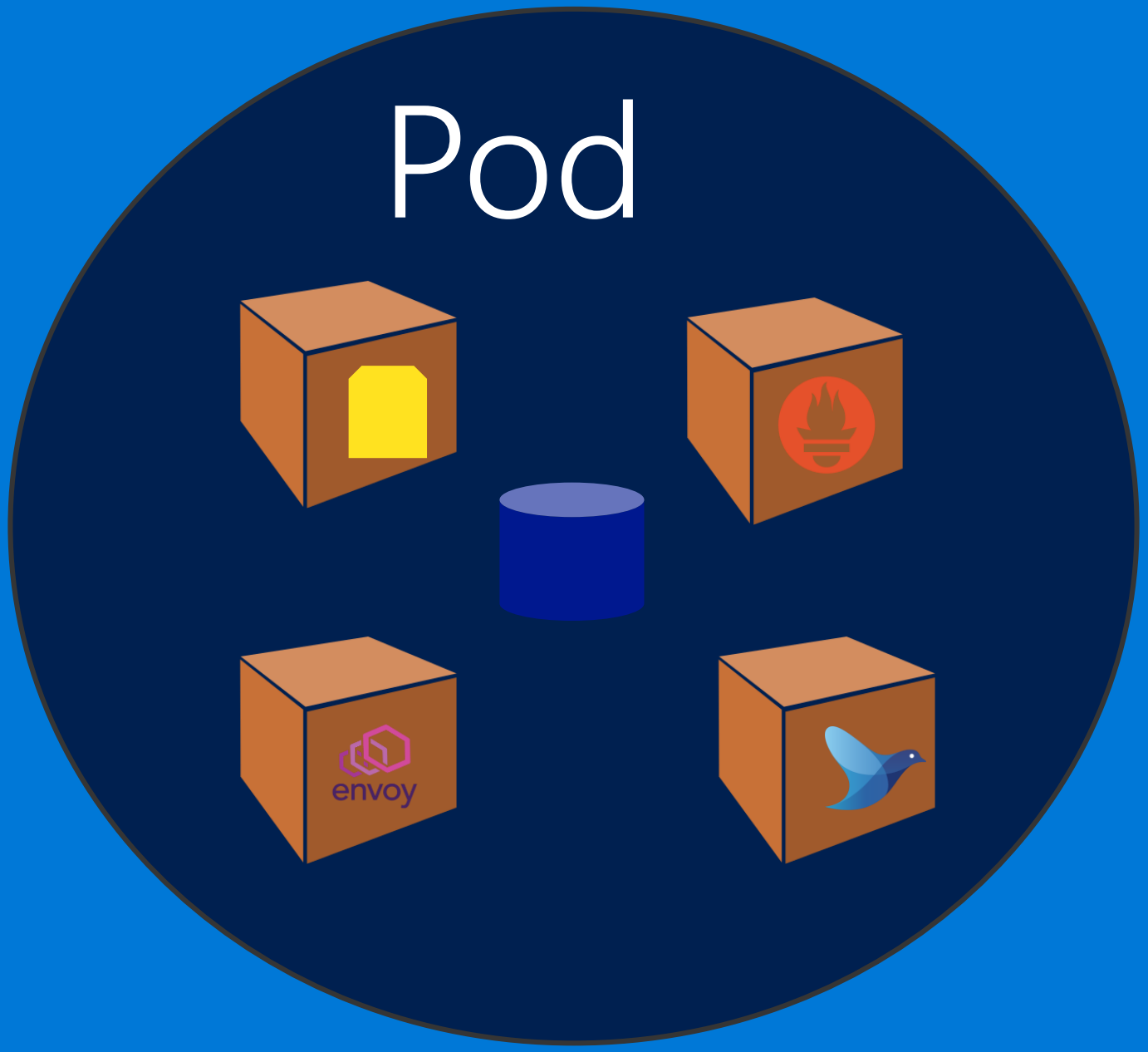
kubelet

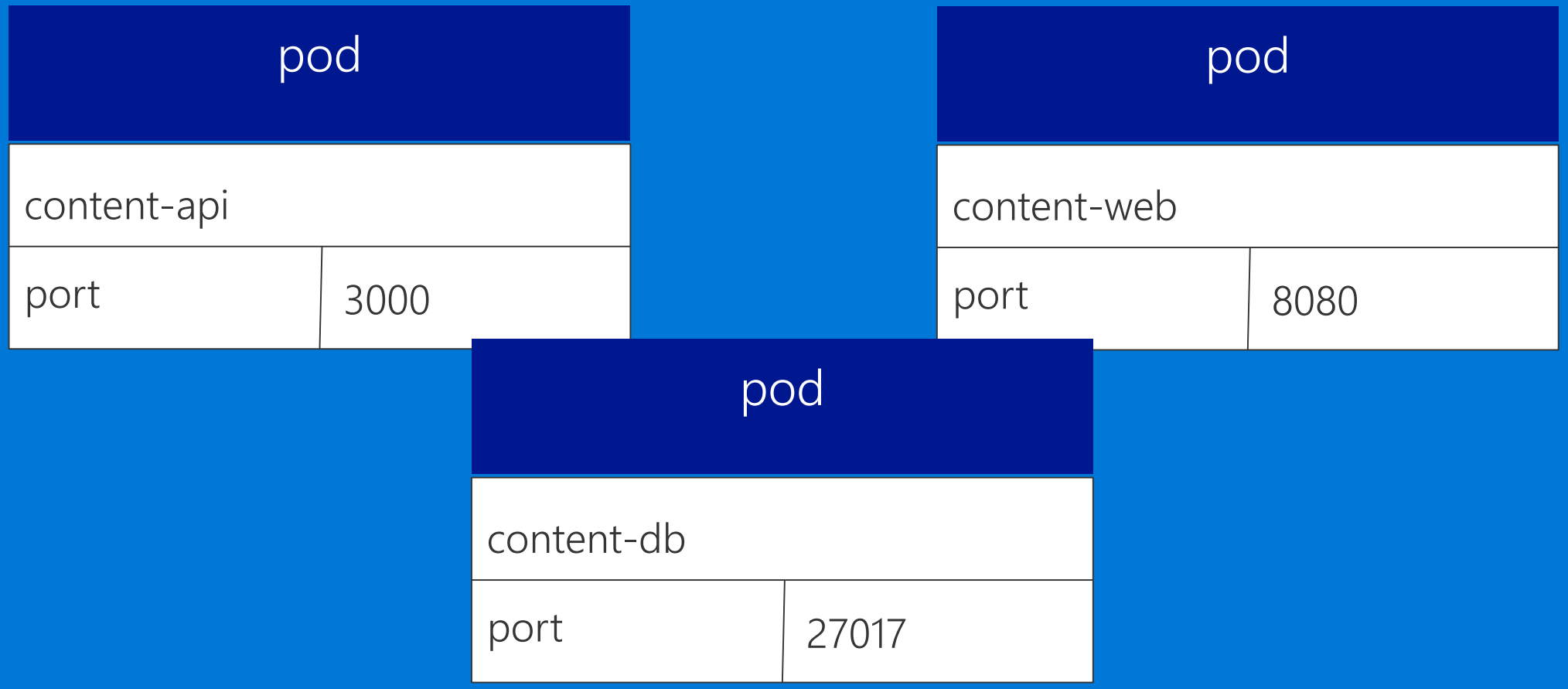
kube-proxy

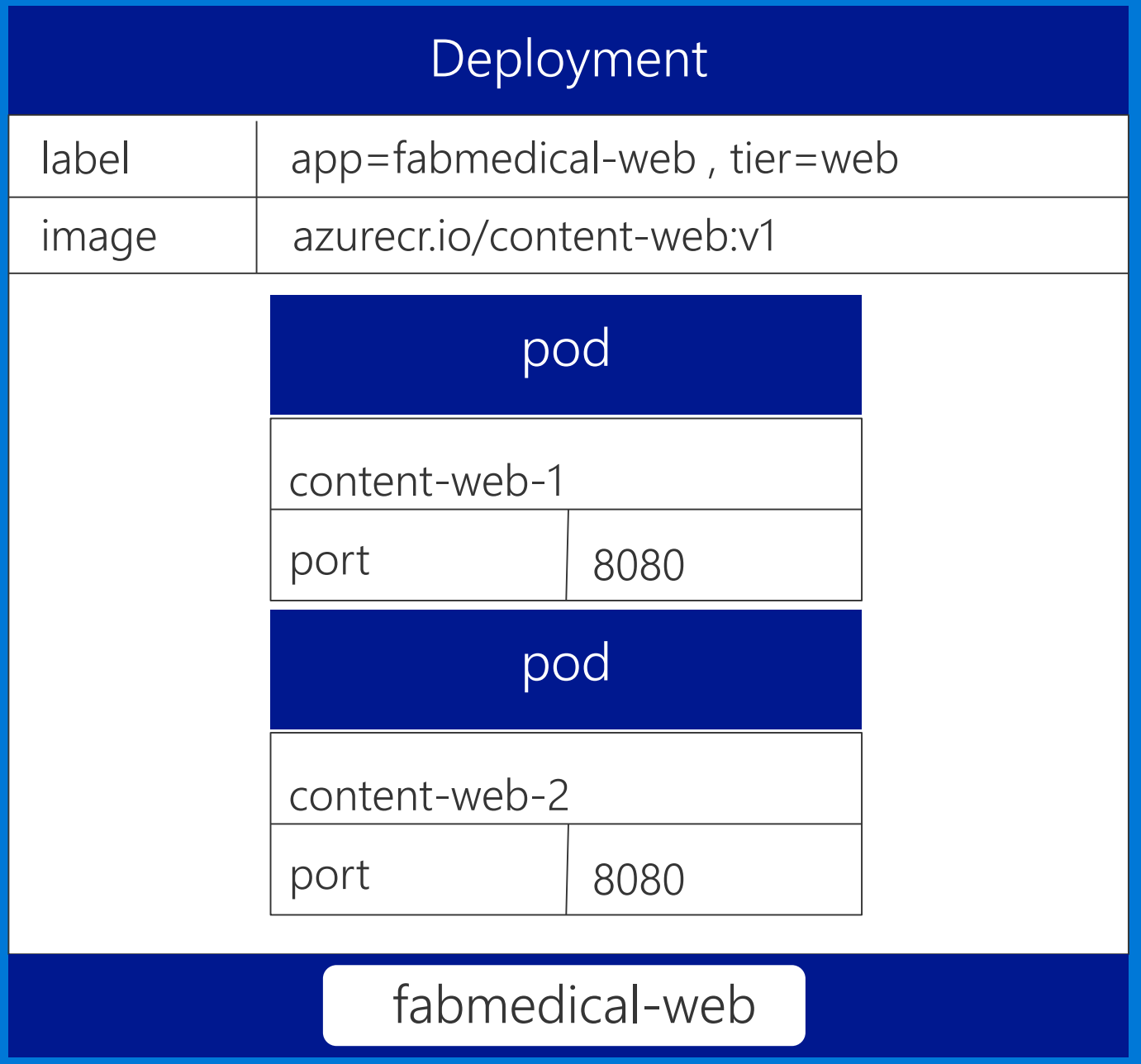
docker

dns

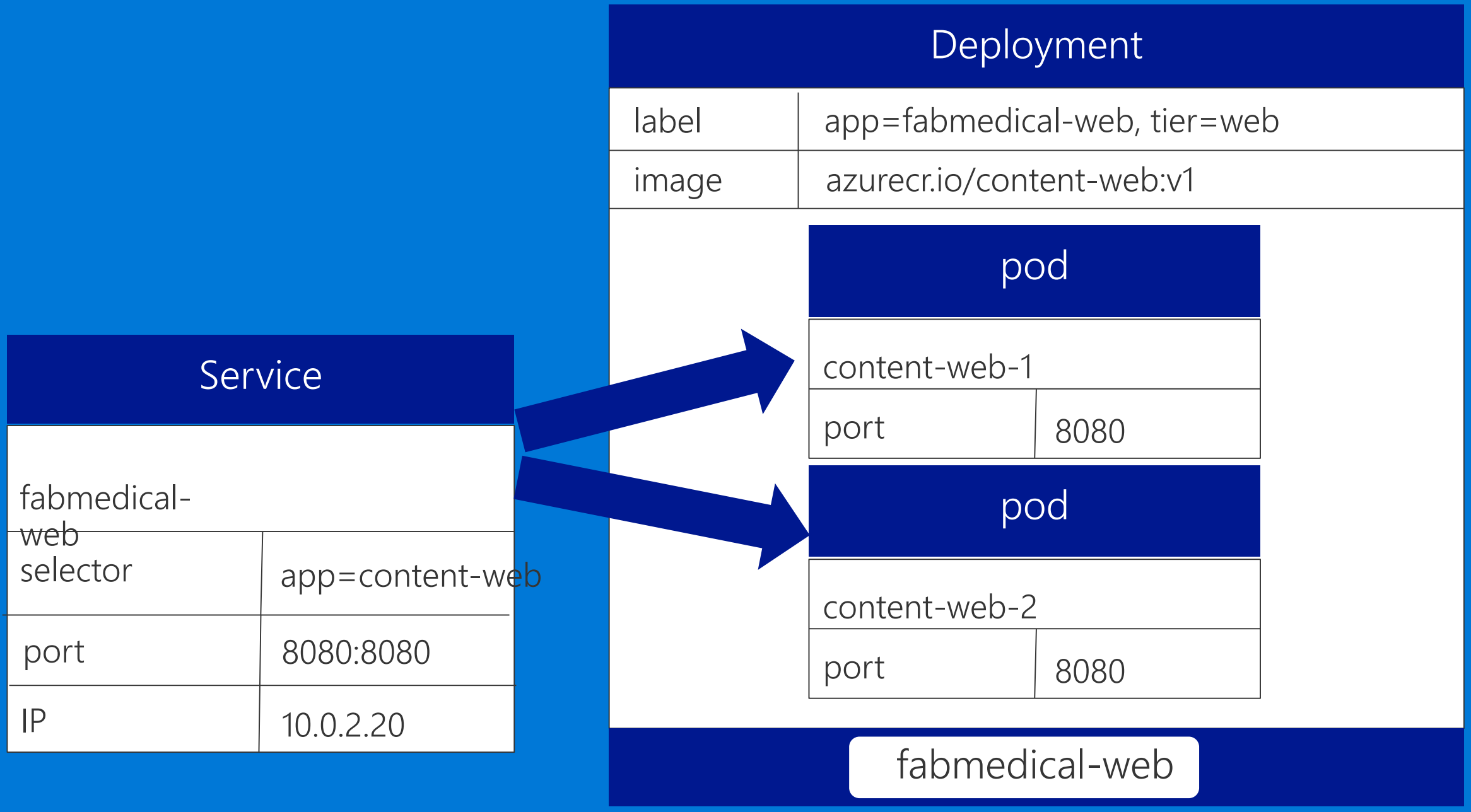
node components


Pod
envoy


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


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

port


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

Kubernetes: what is it?

- Kubernetes technology has been used at Google for 10+ years (Borg Scheduler) and is more mature than Docker Swarm and Mesos
- Built-in service discovery
- Kubernetes is now an open source project from July 2015 with many maintainers, including Microsoft
- kubectl is the primary way to interact with Kubernetes but GUI exists
- No use of Docker tooling, all managed through Kubernetes
- Can run on bare metal and in the cloud
- According to Redmonk, **54 percent of Fortune 100 companies are running Kubernetes** in some form with adoption coming from every sector across the world

Kubernetes: some terminology?

Kubernetes manages applications and not machines

"**Control Plane**" (Master Nodes)

"**Nodes**" runs the containers

"**Pods**" are one or more containers + how to run, shared storage, network

"**Deployments**" control rollout and rollback of stateless pods

"**StatefulSets**" control rollout and rollback of stateful pods

"**Service**" is a pod exposed via a load balancer which serves traffic to load balanced containers(pods)

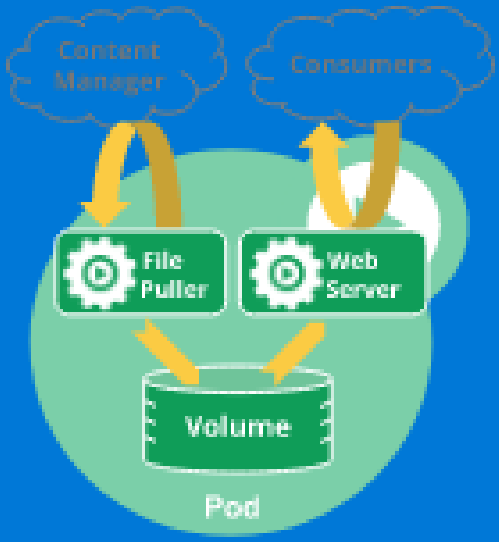
"**Replication controllers / Replica Sets**" handle the start / stop / scalability of stateless pods

Kubernetes – Key Features

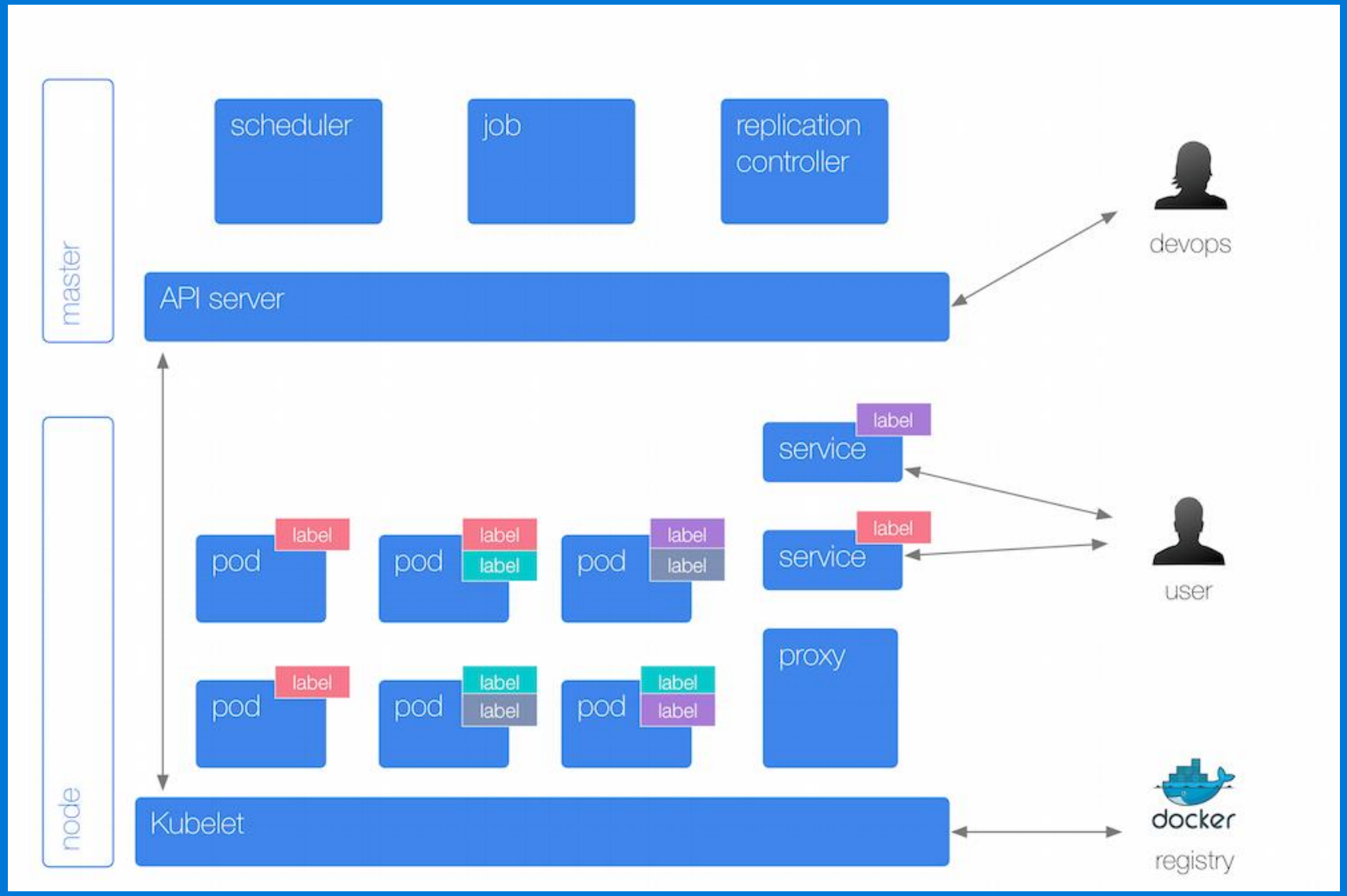
- Key features
  - Declarative infrastructure
  - Self-healing
  - Horizontal scaling
  - Automated rollouts and rollbacks
  - Service discovery and load balancing
  - Automatic bin packing
  - Storage orchestration
  - Secret and configuration management
  - Not a PaaS platform

What is a pod?

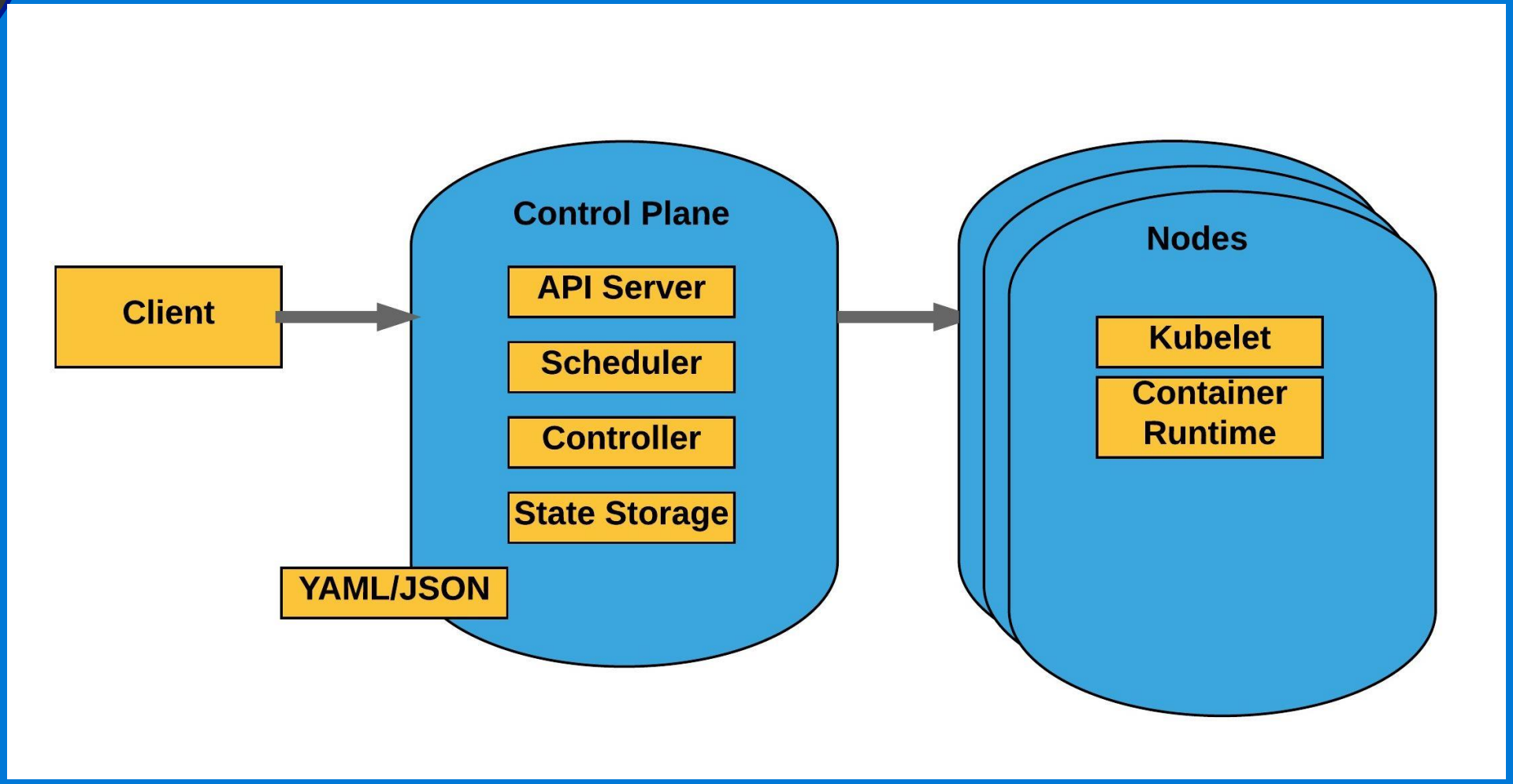
- Pod is the basic building block in kubernetes
- Pods are how containers are delivered
- Can be multiple containers (eg - side car)
- Encapsulates container(s), storage, network IP, and options on how to run
- Use Deployment resources to deploy
  - ReplicaSet
  - StatefulSet
  - DaemonSet
  - Job
  - InitContainer



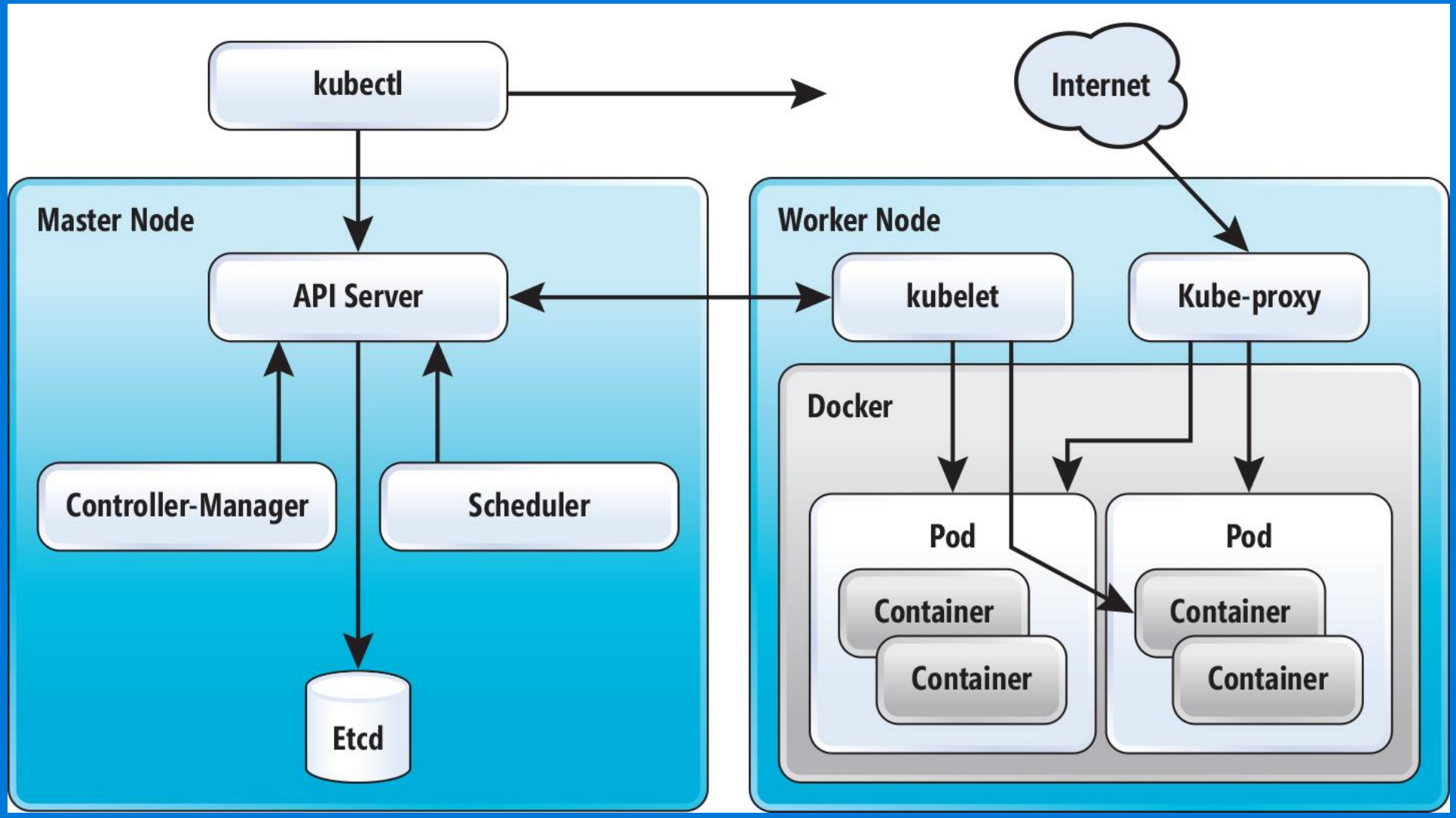
Kubernetes: the visual



Kubernetes: the architecture

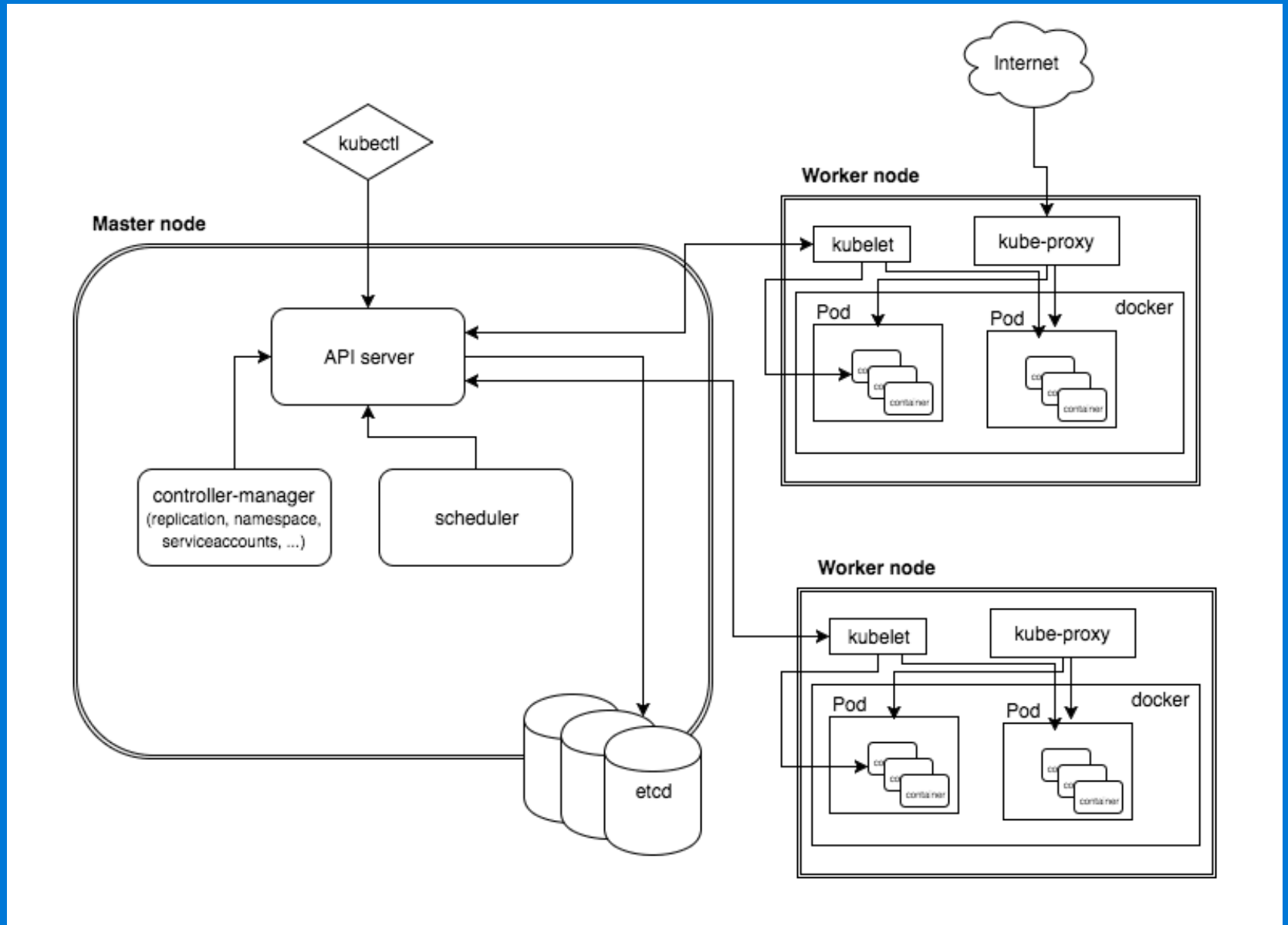


Kubernetes: architecture – Master / Worker

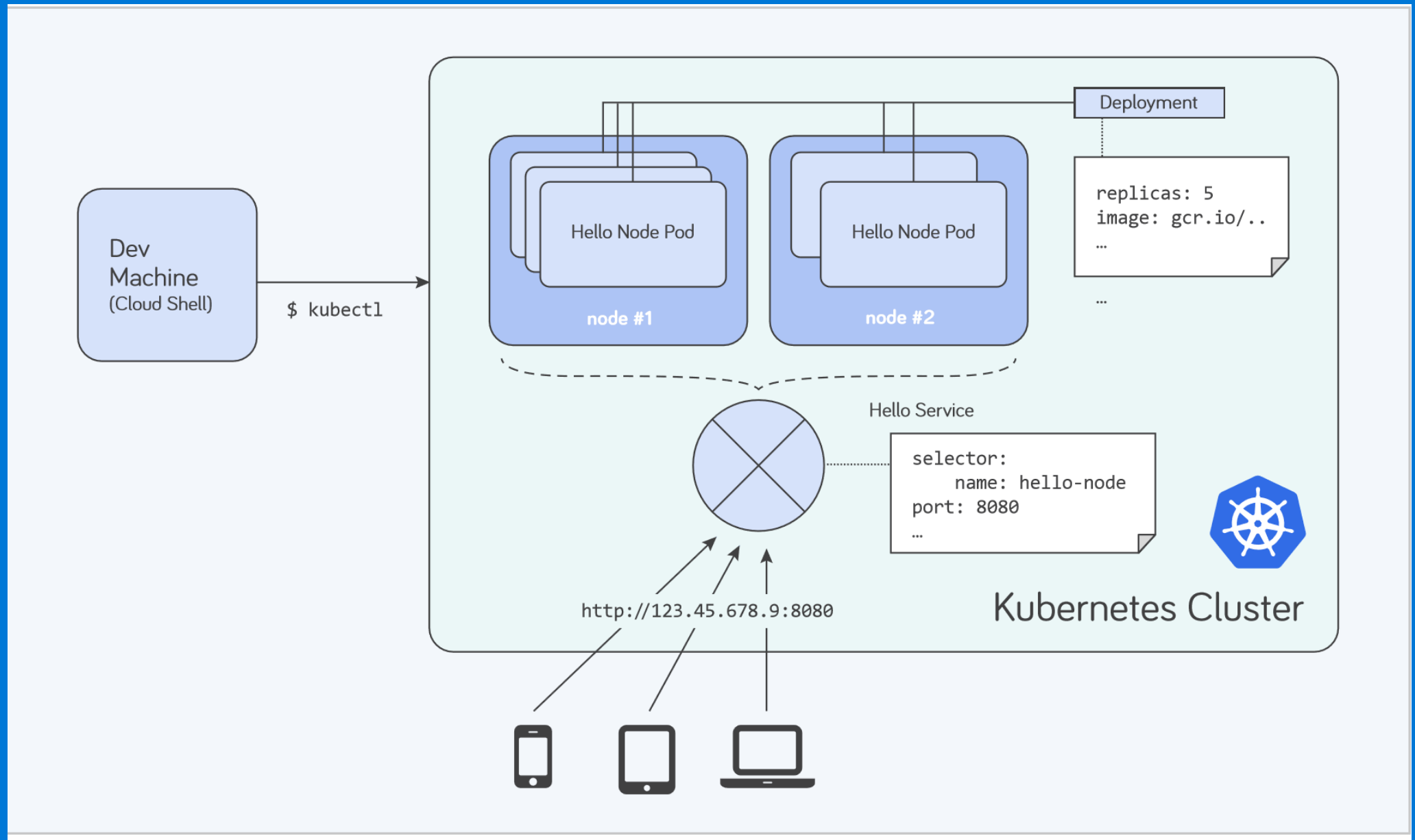


Kubernetes Architecture Components

Kubernetes Architecture



Kubernetes Cluster



Kubernetes Cluster

Here's what this diagram is showing:

- A Cluster is a collection of physical and/or virtual machines called Nodes.
- Each Node is responsible for running a set of Pods.
- A Pod is a group of networked Docker-based containers.
- Outside of the parent-child chain are Deployments and Services.
- Services are logical sets of Pods with a defined policy by which to access them (read: microservice).
- A service can span multiple Nodes within a Kubernetes Cluster.

Kubernetes Resources

pod

deployment

service

replica set

ingress

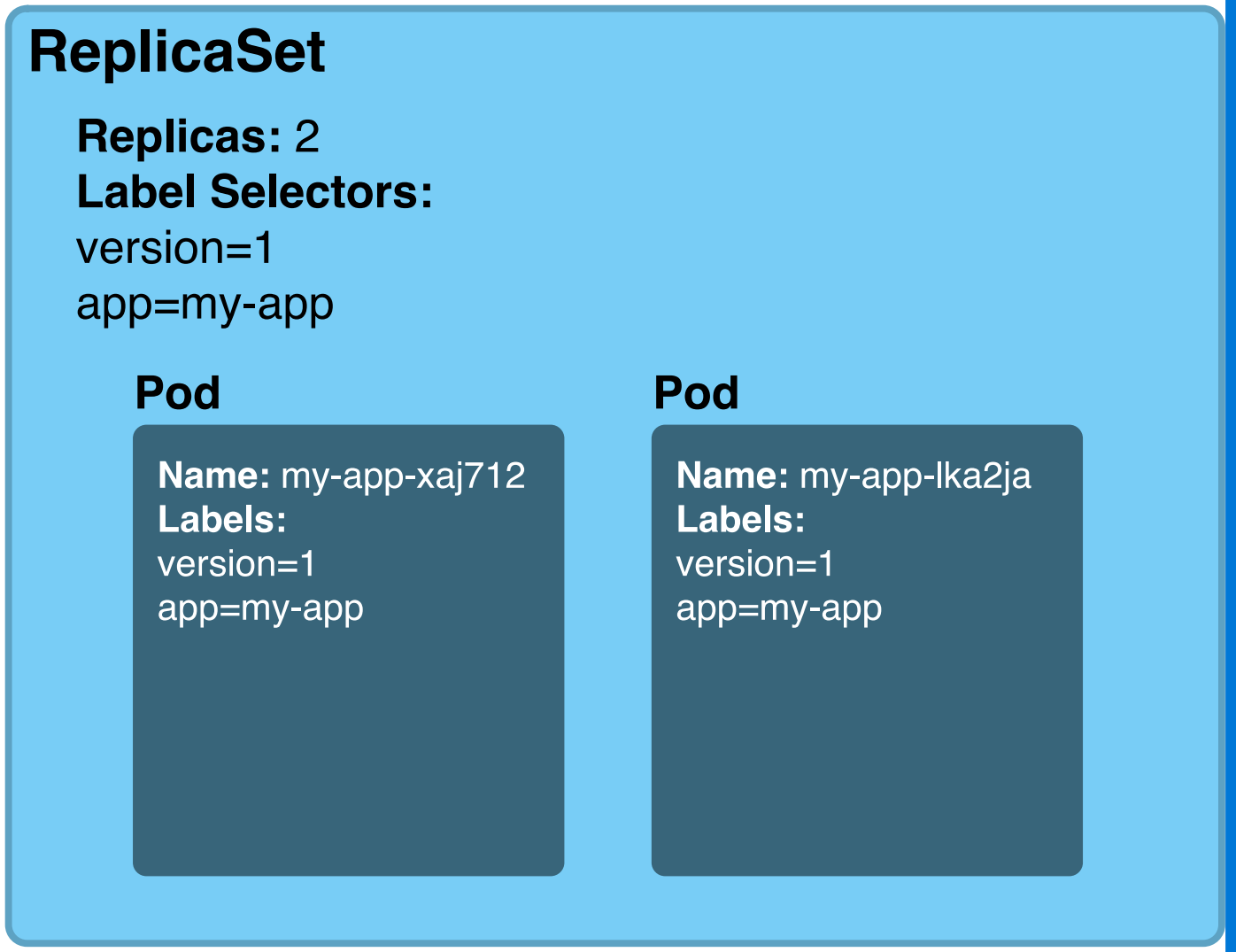
daemon set, job

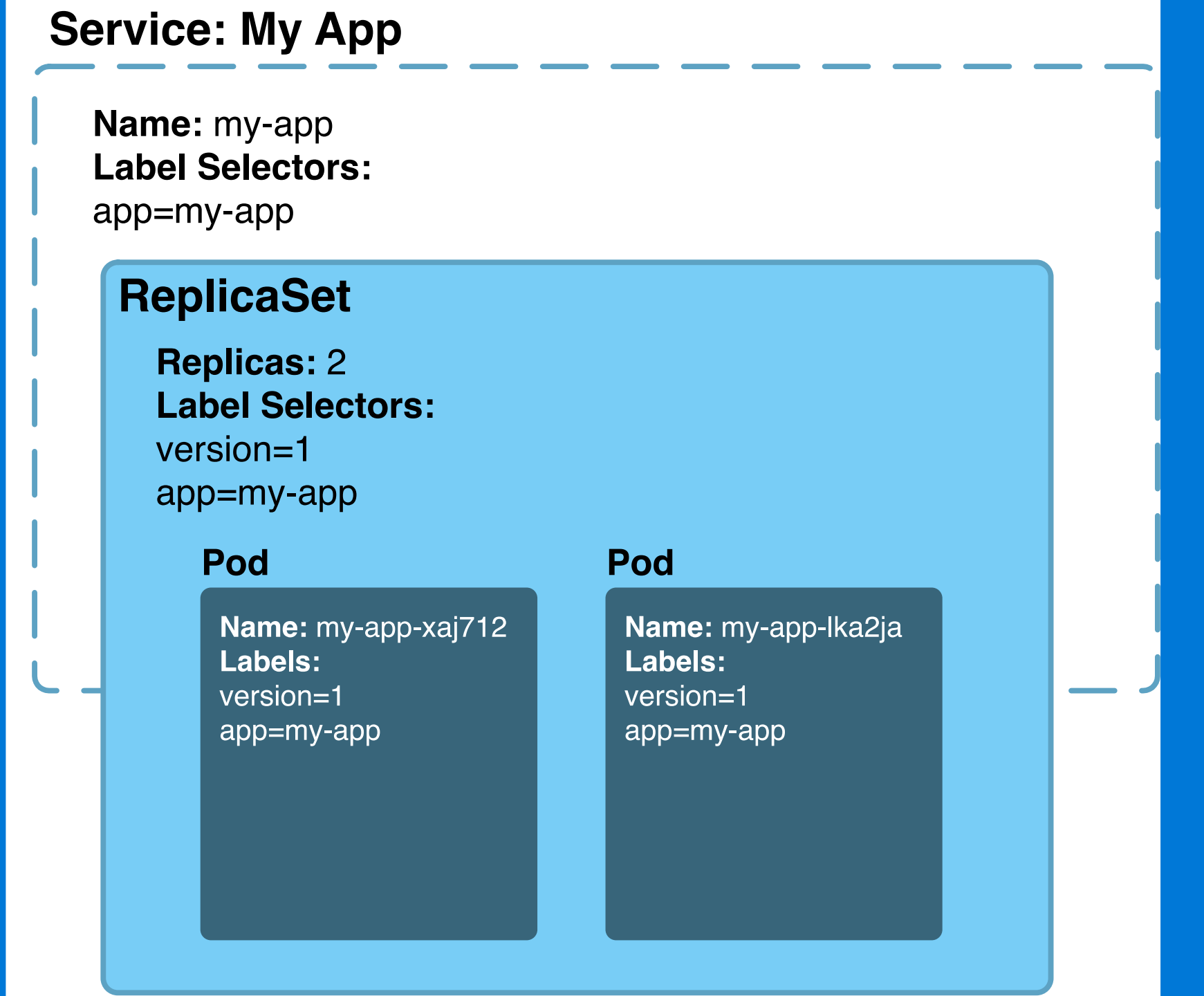
namespace

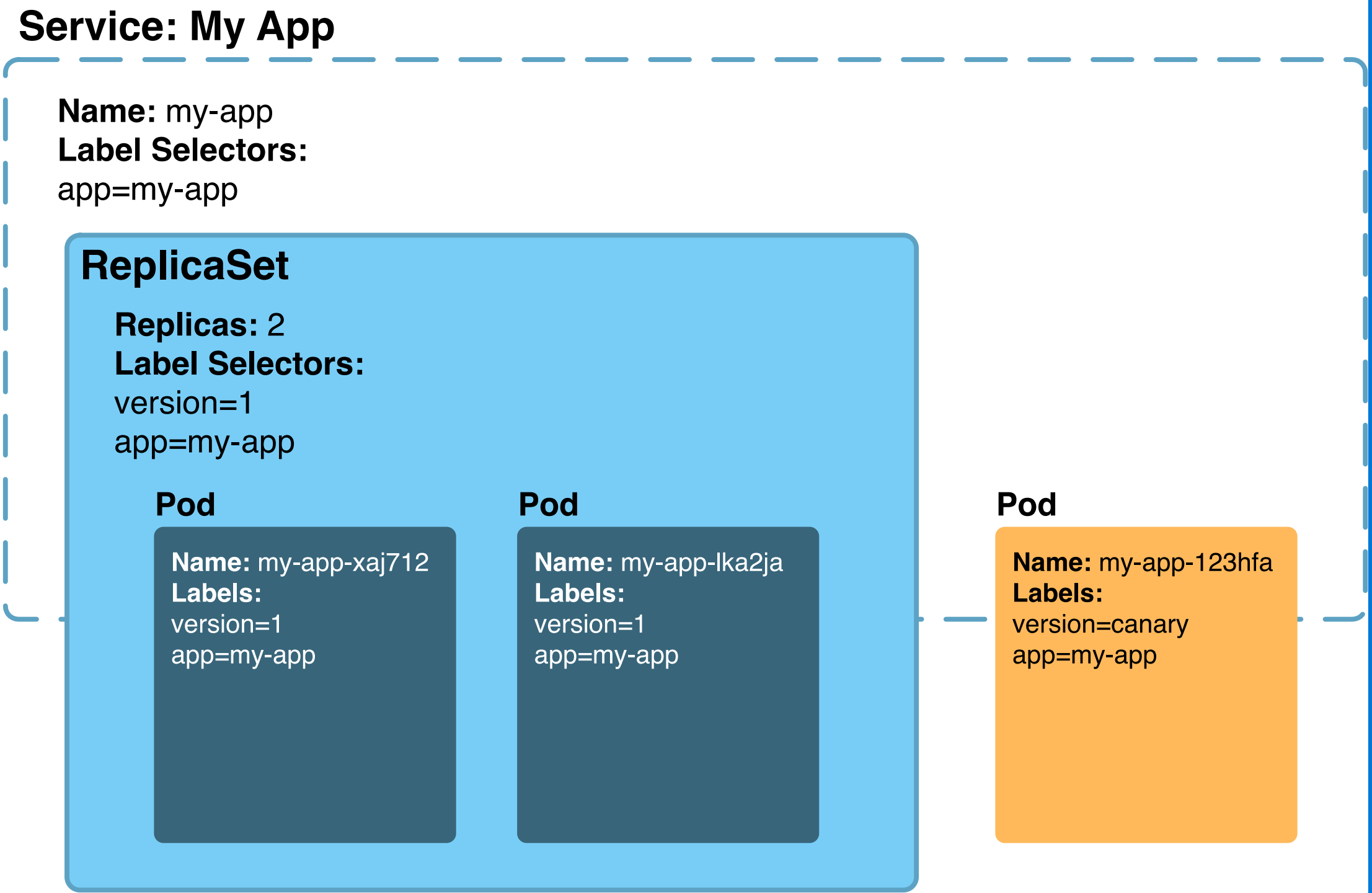
secret, config-map

Pod

Name: my-app-xaj712
Labels:
version=1
app=my-app


ReplicaSet
Replicas: 2
Label Selectors:
version=1
app=my-app
Pod
Name: my-app-xaj712
Labels:
version=1
app=my-app
Pod
Name: my-app-lka2ja
Labels:
version=1
app=my-app


Service: My App
Name: my-app
Label Selectors:
app=my-app
ReplicaSet
Replicas: 2
Label Selectors:
version=1
app=my-app
Pod
Name: my-app-xaj712
Labels:
version=1
app=my-app
Pod
Name: my-app-lka2ja
Labels:
version=1
app=my-app


Service: My App
Name: my-app
Label Selectors:
app=my-app
ReplicaSet
Replicas: 2
Label Selectors:
version=1
app=my-app
Pod
Name: my-app-xaj712
Labels:
version=1
app=my-app
Pod
Name: my-app-lka2ja
Labels:
version=1
app=my-app
Pod
Name: my-app-123hfa
Labels:
version=canary
app=my-app

Service: My App

Name: my-app
Label Selectors:
app=my-app

ReplicaSet

Replicas: 2
Label Selectors:
version=1
app=my-app

Pod

Name: my-app-xaj712
Labels:
version=1
app=my-app

Pod

Name: my-app-lka2ja
Labels:
version=1
app=my-app

ReplicaSet

Replicas: 1
Label Selectors:
version=2
app=my-app

Service: My App

Name: my-app
Label Selectors:
app=my-app

ReplicaSet

Replicas: 2
Label Selectors:
version=1
app=my-app

Pod

Name: my-app-xaj712
Labels:
version=1
app=my-app

Pod

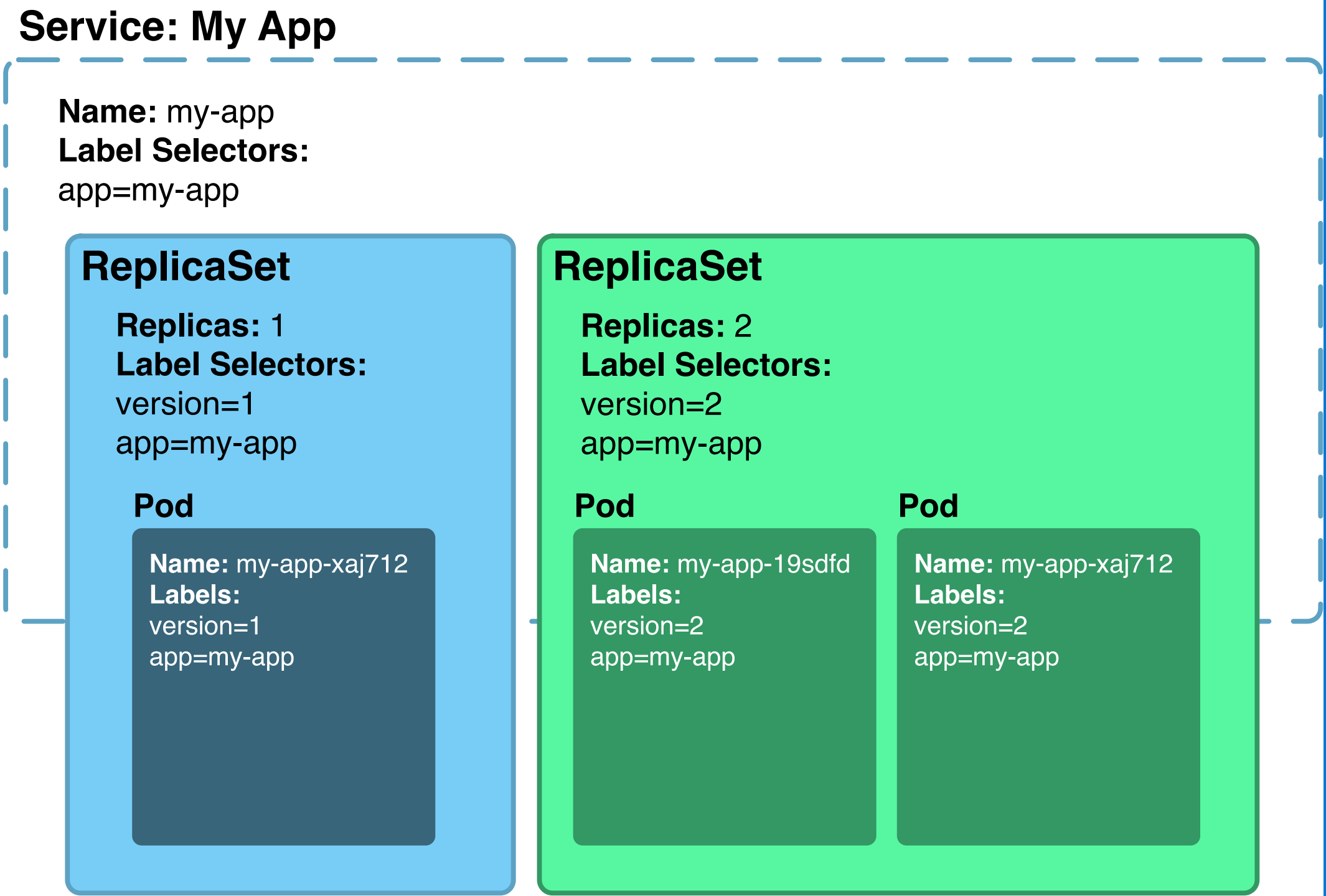
Name: my-app-lka2ja
Labels:
version=1
app=my-app

ReplicaSet

Replicas: 1
Label Selectors:
version=2
app=my-app

Pod

Name: my-app-19sdfd
Labels:
version=2
app=my-app


Service: My App
Name: my-app
Label Selectors:
app=my-app
ReplicaSet
Replicas: 1
Label Selectors:
version=1
app=my-app
Pod
Name: my-app-xaj712
Labels:
version=1
app=my-app
ReplicaSet
Replicas: 2
Label Selectors:
version=2
app=my-app
Pod
Name: my-app-19sdfd
Labels:
version=2
app=my-app
Pod
Name: my-app-xaj712
Labels:
version=2
app=my-app

Service: My App

Name: my-app
Label Selectors:
app=my-app

ReplicaSet

Replicas: 0
Label Selectors:
version=1
app=my-app

ReplicaSet

Replicas: 2
Label Selectors:
version=2
app=my-app

Pod

Name: my-app-19sdfd
Labels:
version=2
app=my-app

Pod

Name: my-app-0q2a87
Labels:
version=2
app=my-app

Microsoft

Resource	Description
http://aka.ms/container-workshop	Github repo of the lab documents
https://rhelblog.redhat.com/2015/07/29/architecting-containers-part-1-user-space-vs-kernel-space/ https://rhelblog.redhat.com/2015/09/17/architecting-containers-part-2-why-the-user-space-matters-2/ https://rhelblog.redhat.com/2015/11/10/architecting-containers-part-3-how-the-user-space-affects-your-application/	Containers 101 Series - Redhat
https://kubernetes.io/docs/home/?path=users&persona=app-developer&level=foundational	Kubernetes Training (official site)
https://vimeo.com/245778144/4d1d597c5e	Kubernetes Deconstructed – KubeCon 2017 (un-abridged version)

Microsoft

http://bit.ly/devops-containers-melbourne

Microsoft

Exercise 3, Step 8

SSH -L 8001:127.0.0.1:8001 -i .ssh/<key>@<ip address>