Containerizing .NET Core Applications with Microsoft Azure and AKS

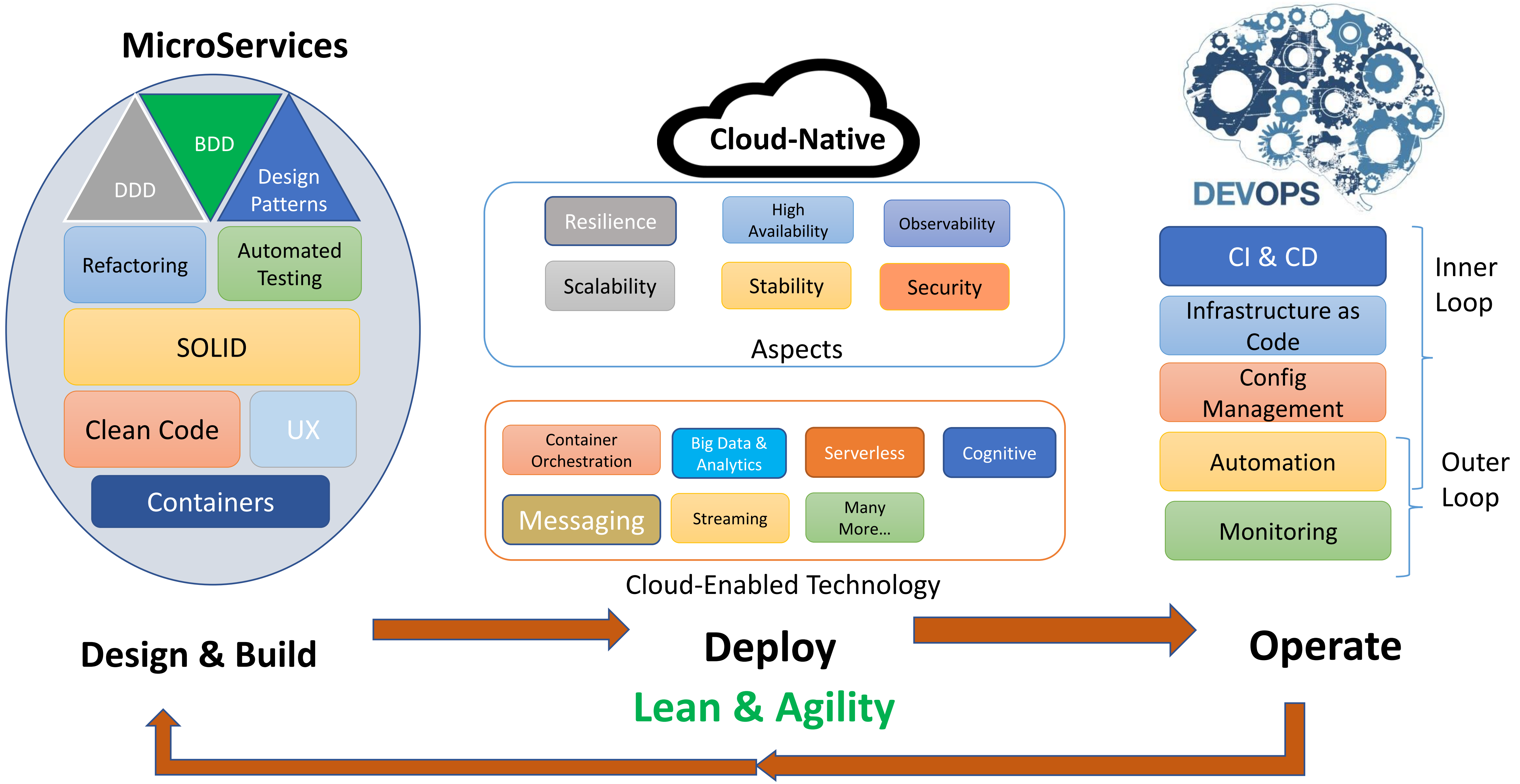
Manoj Ganapathi

Chief Architect, CodeOps

About Me

- Manoj is a seasoned IT professional with more than 20 years of experience. He has extensive experience in enterprise & solution architecture, design and implementation of large & complex enterprise systems. As an architect and technology consultant, he has consulted with several large, fortune 500 enterprises and worked with ISVs and startups. In his career, he has worked in multiple technology-oriented and leadership roles across all phases of software development life cycle. He is experienced in building and running technical communities and has been a speaker in several technology conferences.
- Over the last decade, he has worked extensively on consulting, architecture and implementation of Cloud-based solutions, specializing on building highly scalable, resilient systems and DevOps practices.
- Currently, he is the Chief Architect at CodeOps Technologies (http://codeops.tech/) and a Digital Technology Consultant.
- LinkedIn profile: https://www.linkedin.com/in/manojg
- @manojgr, manoj@codeops.tech

Perspective on Modern Software Delivery



MicroServices

DDD

BDD

Design Patterns

Refactoring

Automated Testing

SOLID

Clean Code

UX

Containers

Design & Build

Cloud-Native

Resilience

High Availability

Observability

Scalability

Stability

Security

Aspects

Container Orchestration

Big Data & Analytics

Serverless

Cognitive

Messaging

Streaming

Many More…

Cloud-Enabled Technology

Deploy

DEVOPS

CI & CD

Infrastructure as Code

Config Management

Automation

Monitoring

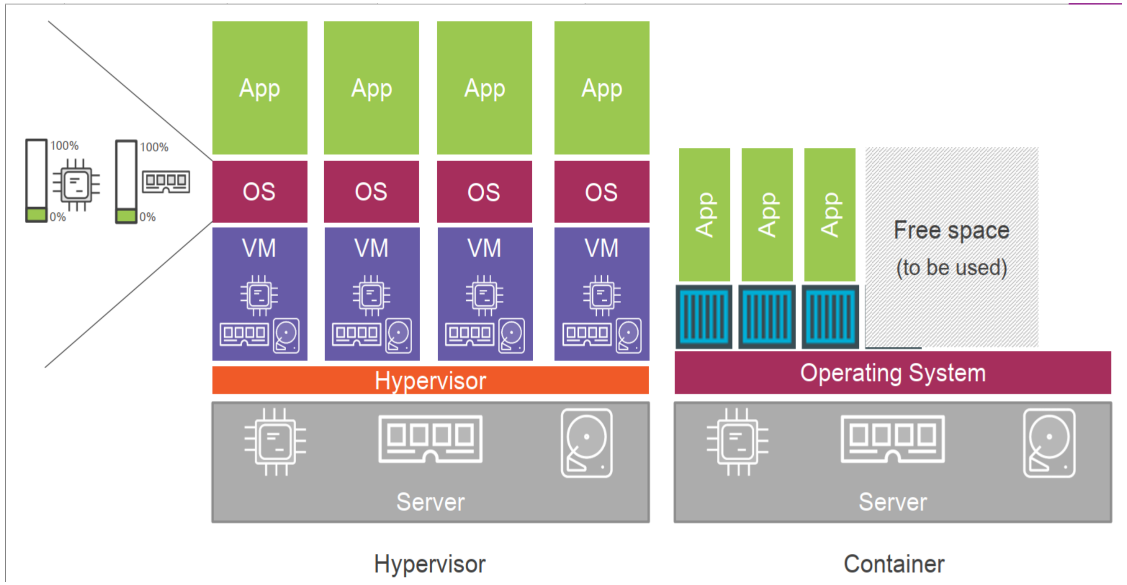
Inner Loop

Outer Loop

Operate

Lean & Agility

Containers v/s VMs

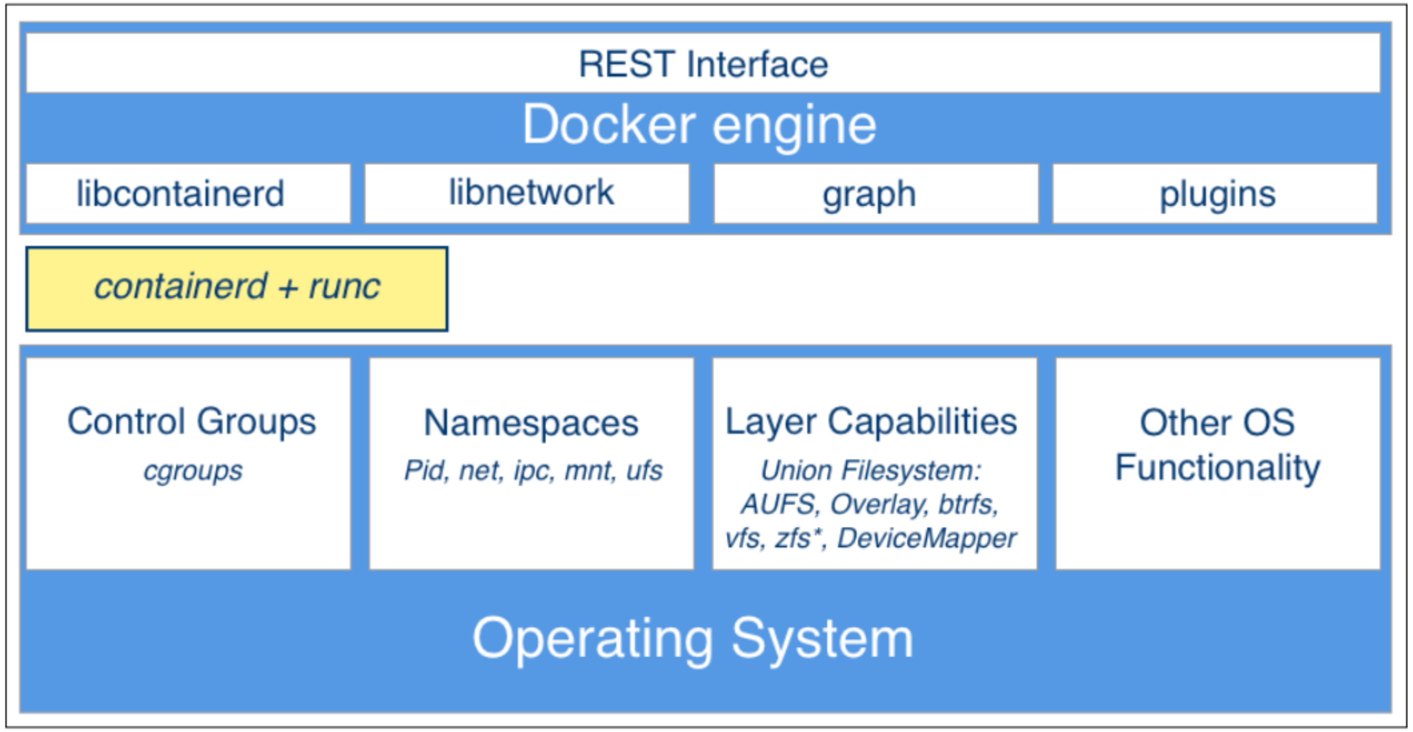


- Image Ref: Nigel Poulton

Benefits of Containers

- Lightweight/Lower Overhead
- ✓ Consistent Environment
- Run Anywhere

Docker Architecture



REST Interface

Docker engine

libcontainerd

libnetwork

graph

plugins

containerd + runc

Control Groups

cgroups

Namespaces

Pid, net, ipc, mnt, ufs

Layer Capabilities

Union Filesystem: AUFS, Overlay, btrfs, vfs, zfs, DeviceMapper*

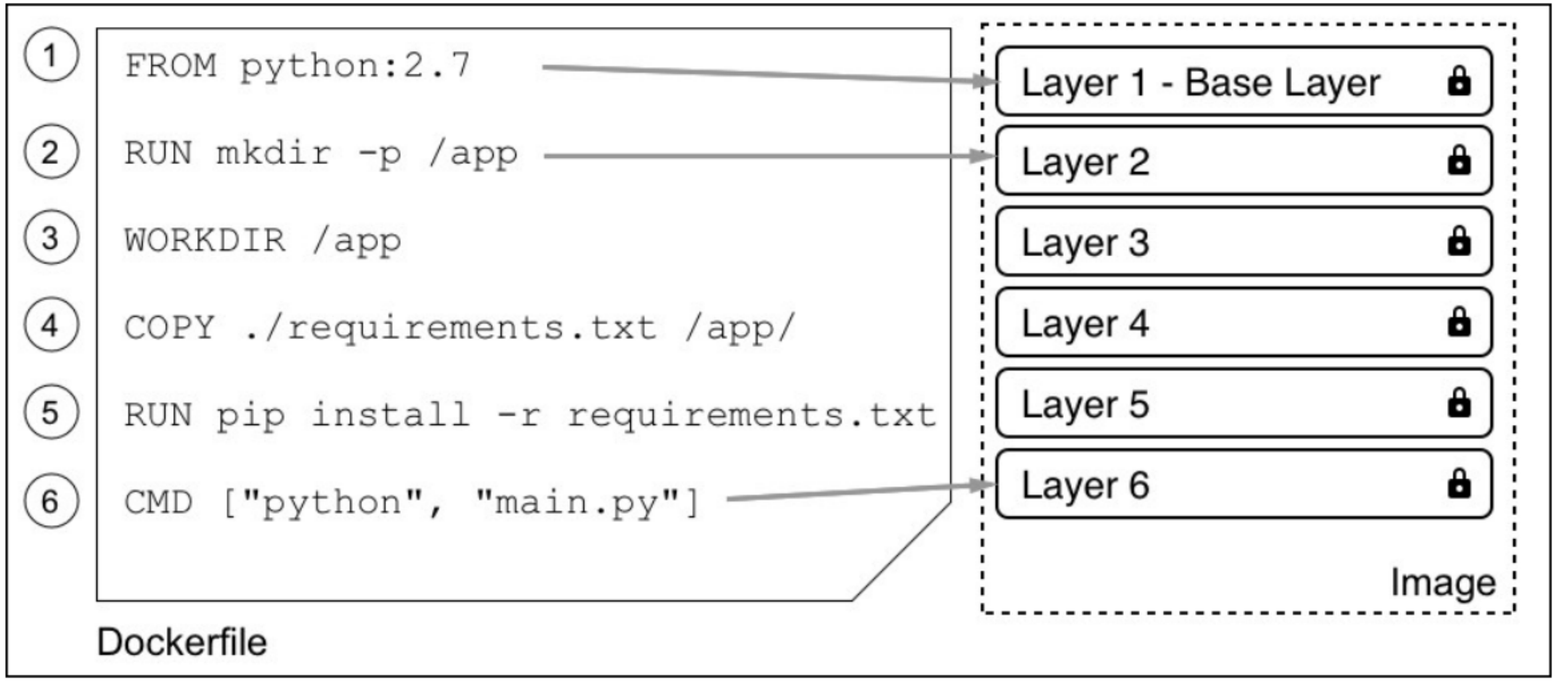
Other OS Functionality

Operating System

- Containers are encapsulated, secure processes running on the host platform
- Benefits
  - Security
  - Isolation
  - Standardized Infra

Ref: [Containerize your apps with Docker & Kubernetes]() - book

Container Images & Dockerfile



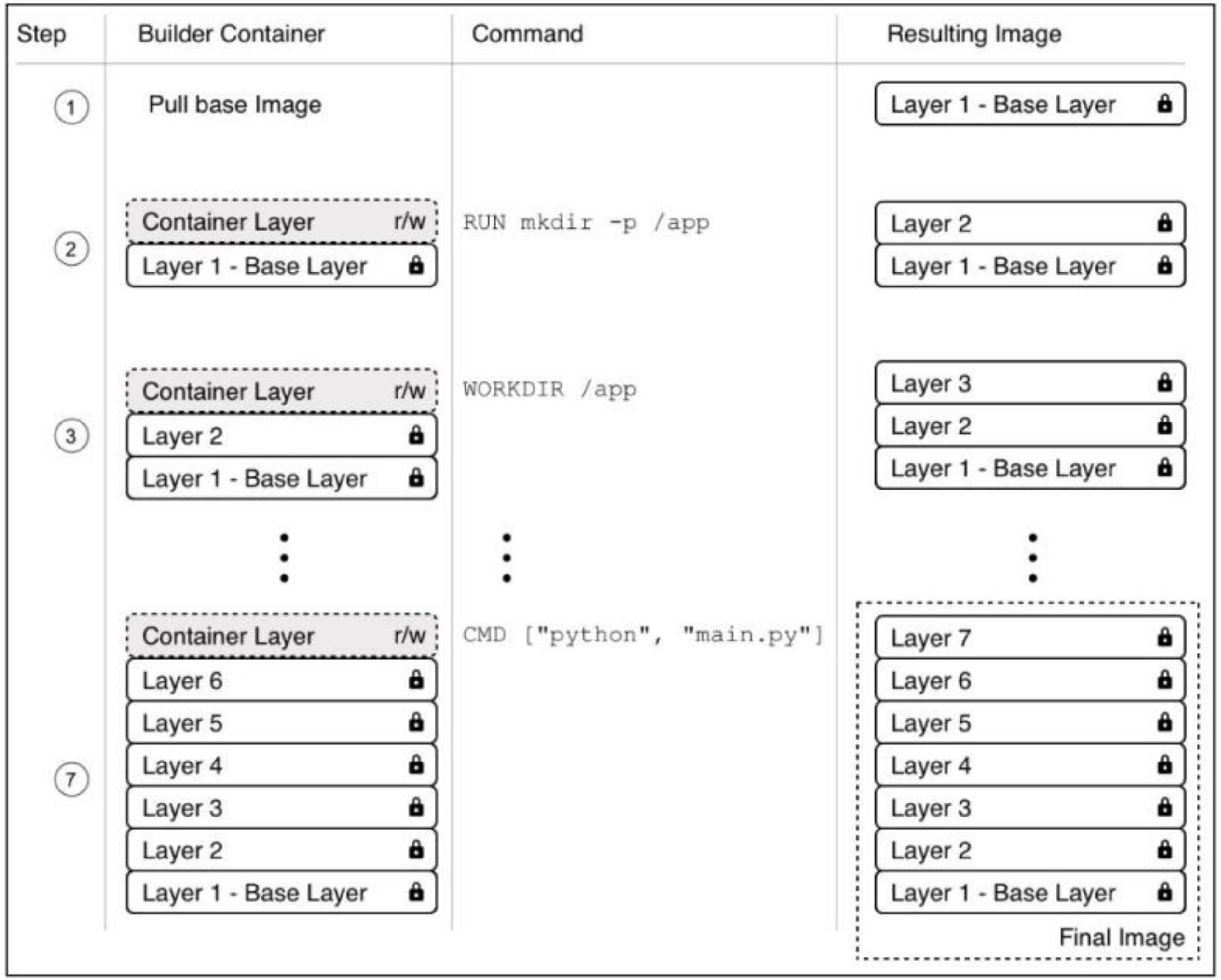
```
FROM python:2.7
RUN mkdir -p /app
WORKDIR /app
COPY ./requirements.txt /app/
RUN pip install -r requirements.txt
CMD ["python", "main.py"]
```

The relation of Dockerfile and layers in an image

- Templates for Containers
- Starts with a base layer, usually the OS
- All layers immutable, only top layer is writable

Ref: Containerize your apps with Docker & Kubernetes - book

Best Practices



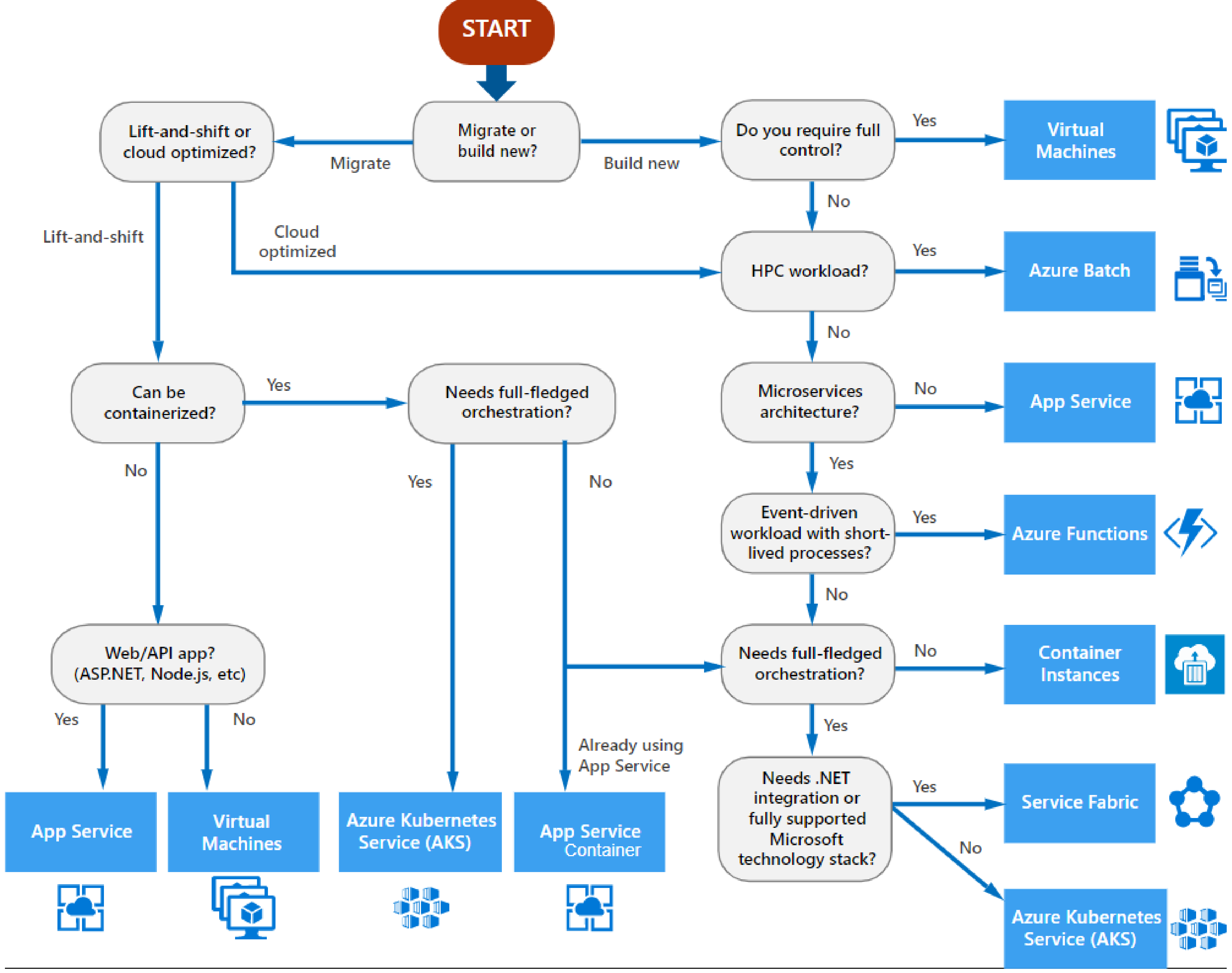
Step	Builder Container	Command	Resulting Image
1	Pull base Image		Layer 1 - Base Layer 🔒
2	Container Layer r/w Layer 1 - Base Layer 🔒	RUN mkdir -p /app	Layer 2 🔒 Layer 1 - Base Layer 🔒
3	Container Layer r/w Layer 2 🔒 Layer 1 - Base Layer 🔒	WORKDIR /app	Layer 3 🔒 Layer 2 🔒 Layer 1 - Base Layer 🔒
	⋮	⋮	⋮
7	Container Layer r/w Layer 6 🔒 Layer 5 🔒 Layer 4 🔒 Layer 3 🔒 Layer 2 🔒 Layer 1 - Base Layer 🔒	CMD ["python", "main.py"]	Layer 7 🔒 Layer 6 🔒 Layer 5 🔒 Layer 4 🔒 Layer 3 🔒 Layer 2 🔒 Layer 1 - Base Layer 🔒 Final Image

The image build process visualized

- Keep Containers ephemeral
- Order commands to leverage caching
- Avoid installing multiple packages
- Use .dockerignore
- Use multi-stage builds

Ref: Containerize your apps with Docker & Kubernetes - book

Choosing the right compute option


START
Migrate or build new?
Migrate
Build new
Lift-and-shift or cloud optimized?
Lift-and-shift
Cloud optimized
Do you require full control?
Yes
Virtual Machines
No
HPC workload?
Yes
Azure Batch
No
Microservices architecture?
No
App Service
Yes
Event-driven workload with short-lived processes?
Yes
Azure Functions
No
Needs full-fledged orchestration?
No
Container Instances
Yes
Needs .NET integration or fully supported Microsoft technology stack?
Yes
Service Fabric
No
Azure Kubernetes Service (AKS)
Can be containerized?
Yes
Needs full-fledged orchestration?
Yes
No
Already using App Service
No
Web/API app? (ASP.NET, Node.js, etc)
Yes
No
App Service
Virtual Machines
Azure Kubernetes Service (AKS)
App Service Container

https://docs.microsoft.com/en-us/azure/architecture/guide/technology-choices/compute-decision-tree

Need for Container Orchestrators

- VELOCITY
- SCALING (SOFTWARE & TEAMS)
- ABSTRACTING INFRASTRUCTURE
- EFFICIENCY

Achieving Velocity

IMMUTABILITY

DECLARATIVE CONFIGURATION

SELF-HEALING

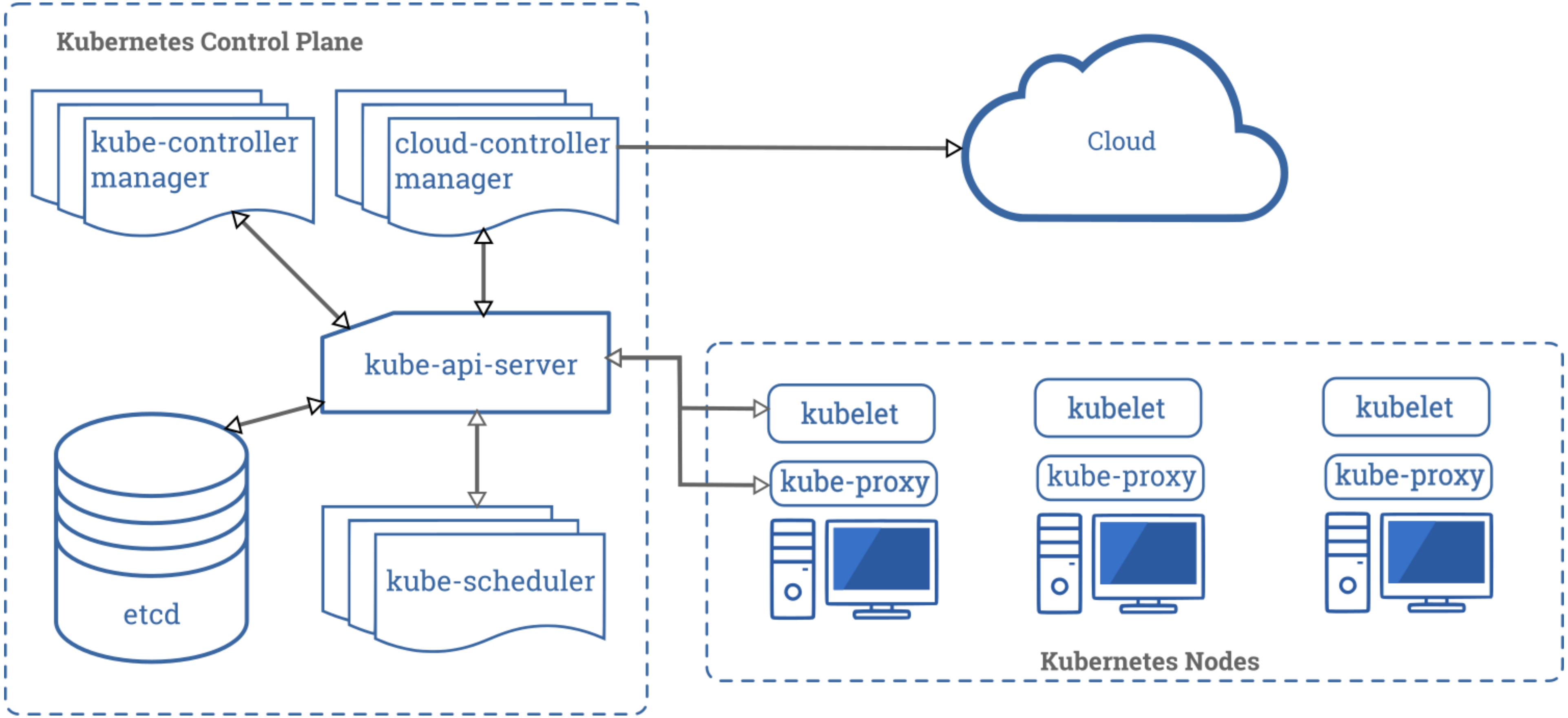
Other benefits

DECOUPLED ARCHITECTURE

HIGH UTILIZATION

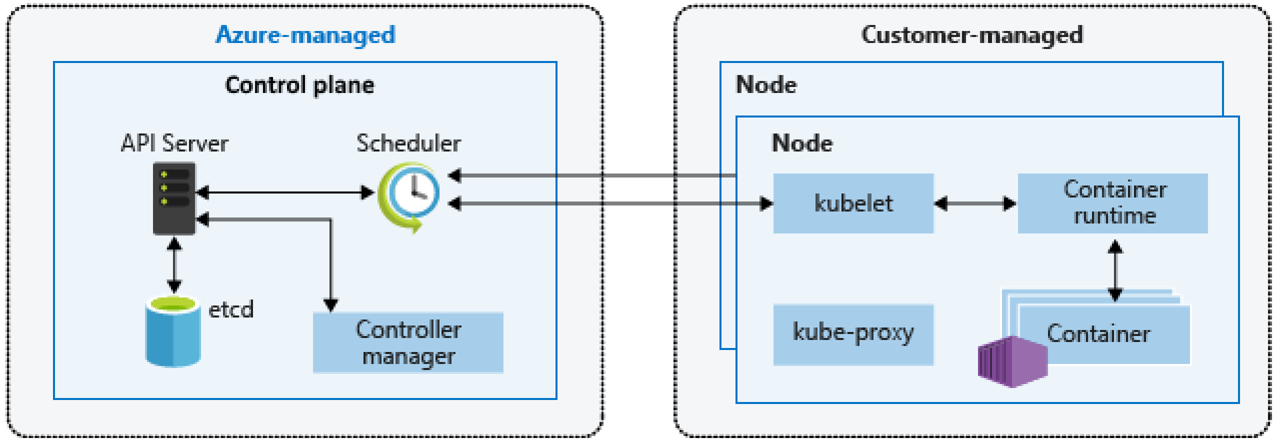
EFFICIENCY

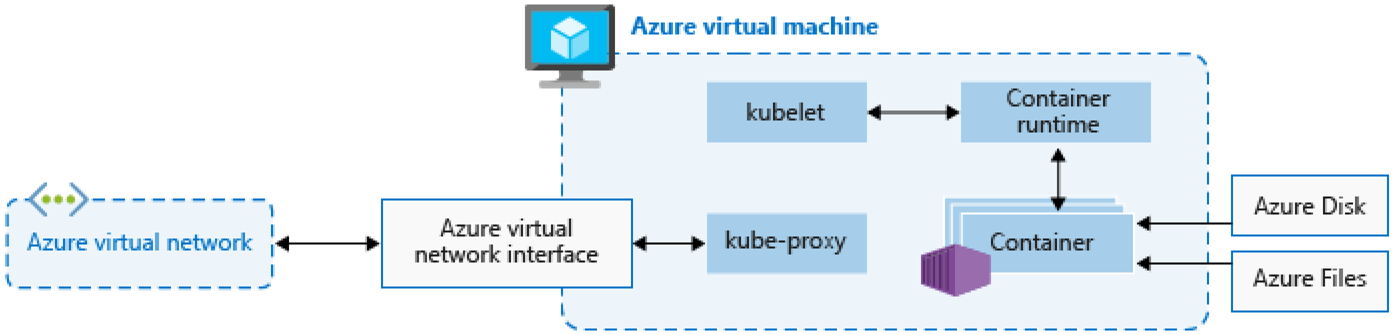
K8S Architecture



https://kubernetes.io/docs/concepts/overview/components/

AKS Deployment Model





[Concepts - Kubernetes basics for Azure Kubernetes Services (AKS) - Azure Kubernetes Service | Microsoft Docs]

K8S objects

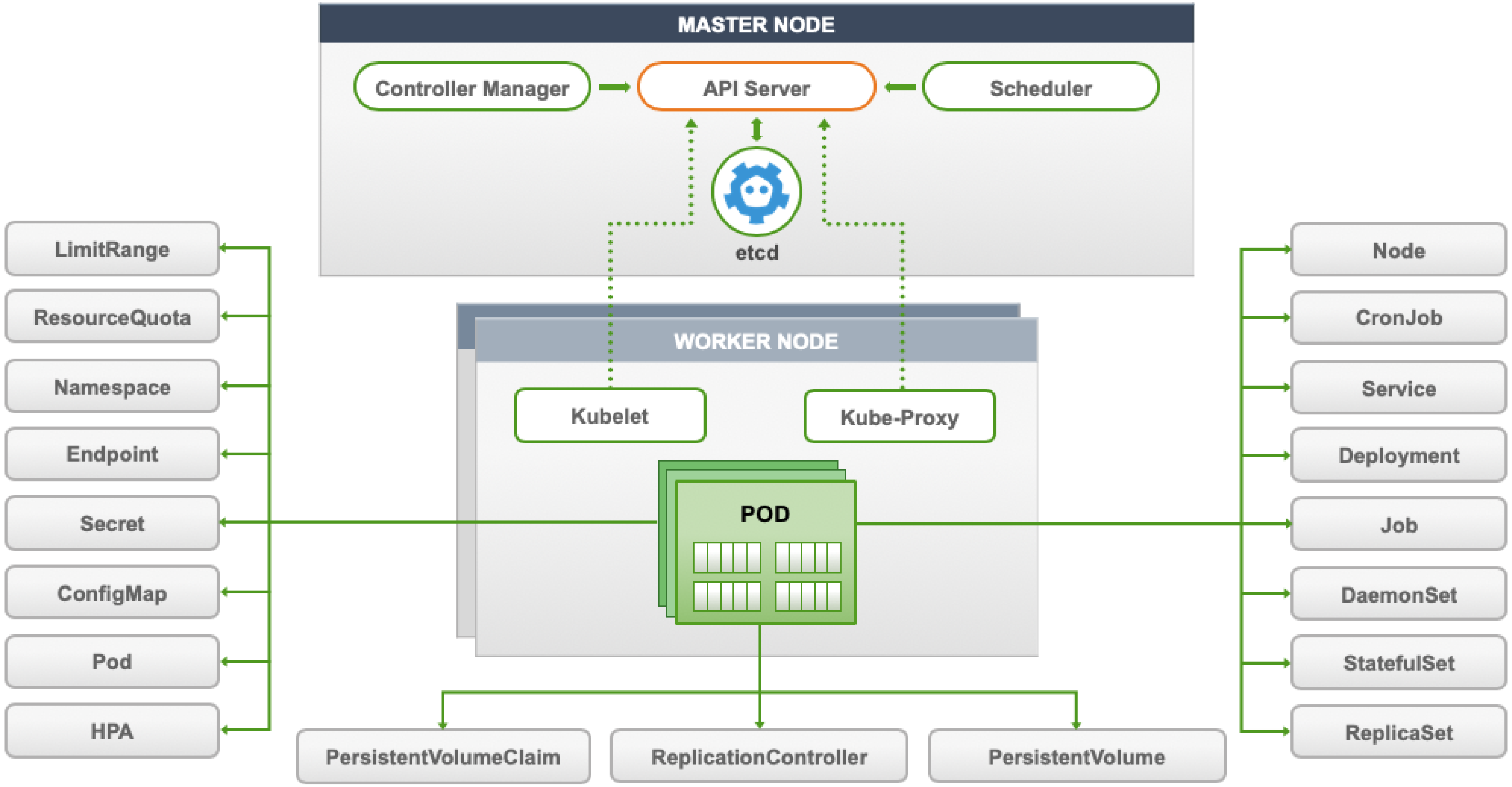


Image Ref: https://www.splunk.com/en_us/blog/it/monitoring-kubernetes.html

Pod Lifecycle

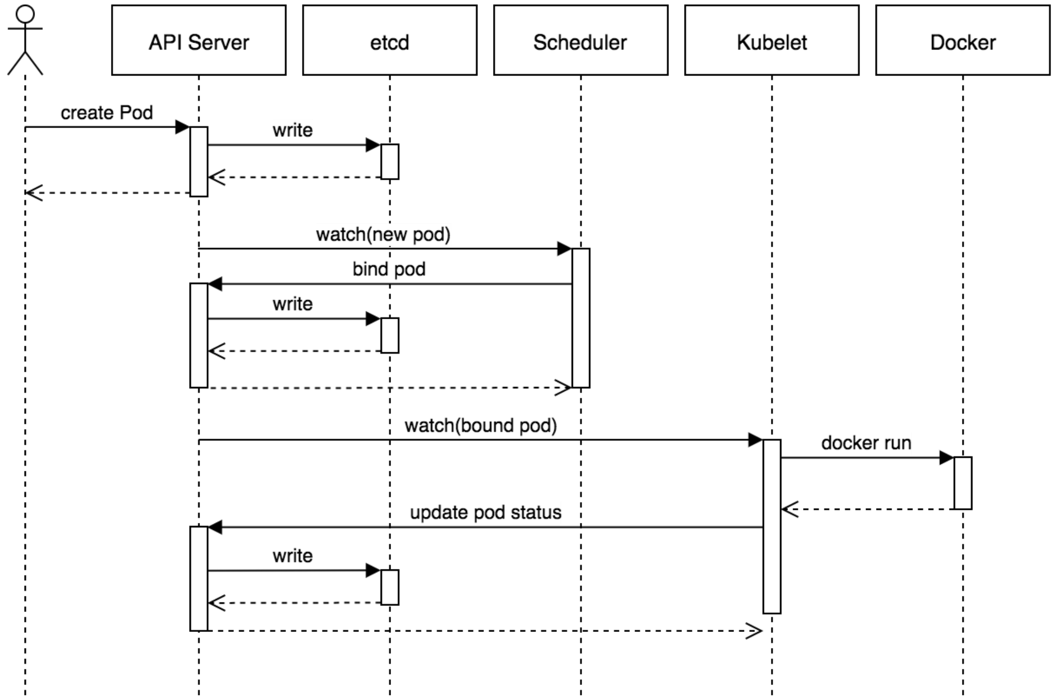


Image Ref: Joe Beda’s Blog

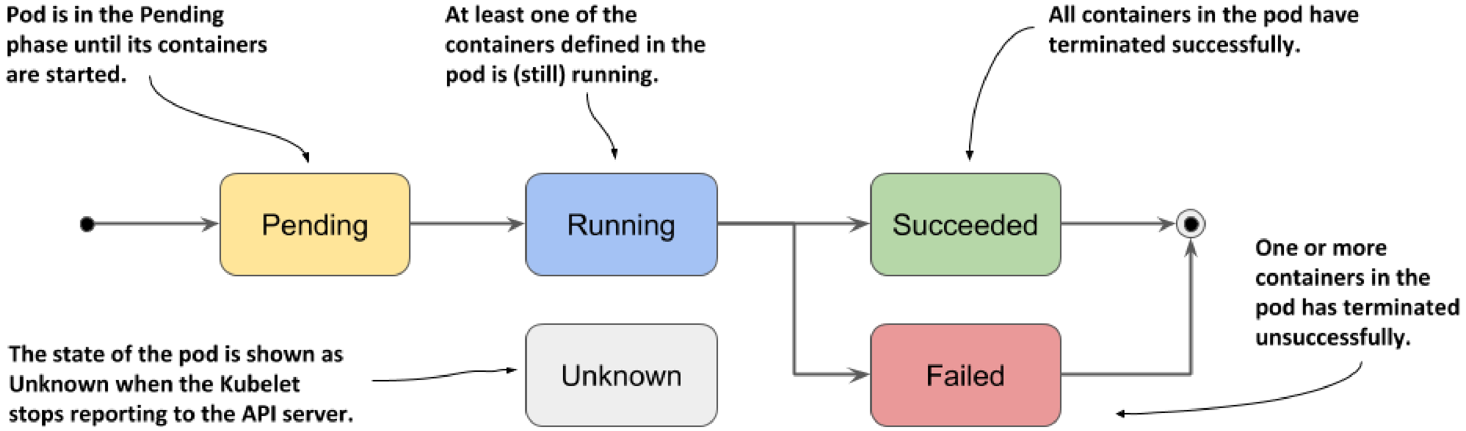

Pod is in the Pending phase until its containers are started.
At least one of the containers defined in the pod is (still) running.
All containers in the pod have terminated successfully.
Pending
Running
Succeeded
Unknown
Failed
One or more containers in the pod has terminated unsuccessfully.
The state of the pod is shown as Unknown when the Kubelet stops reporting to the API server.

Image Ref: Manning Books

K8S Extensibility

K8s is highly modular

A resource is an endpoint in the Kubernetes API that stores a collection of API objects of a certain kind; for example, the built-in pods resource contains a collection of Pod objects.

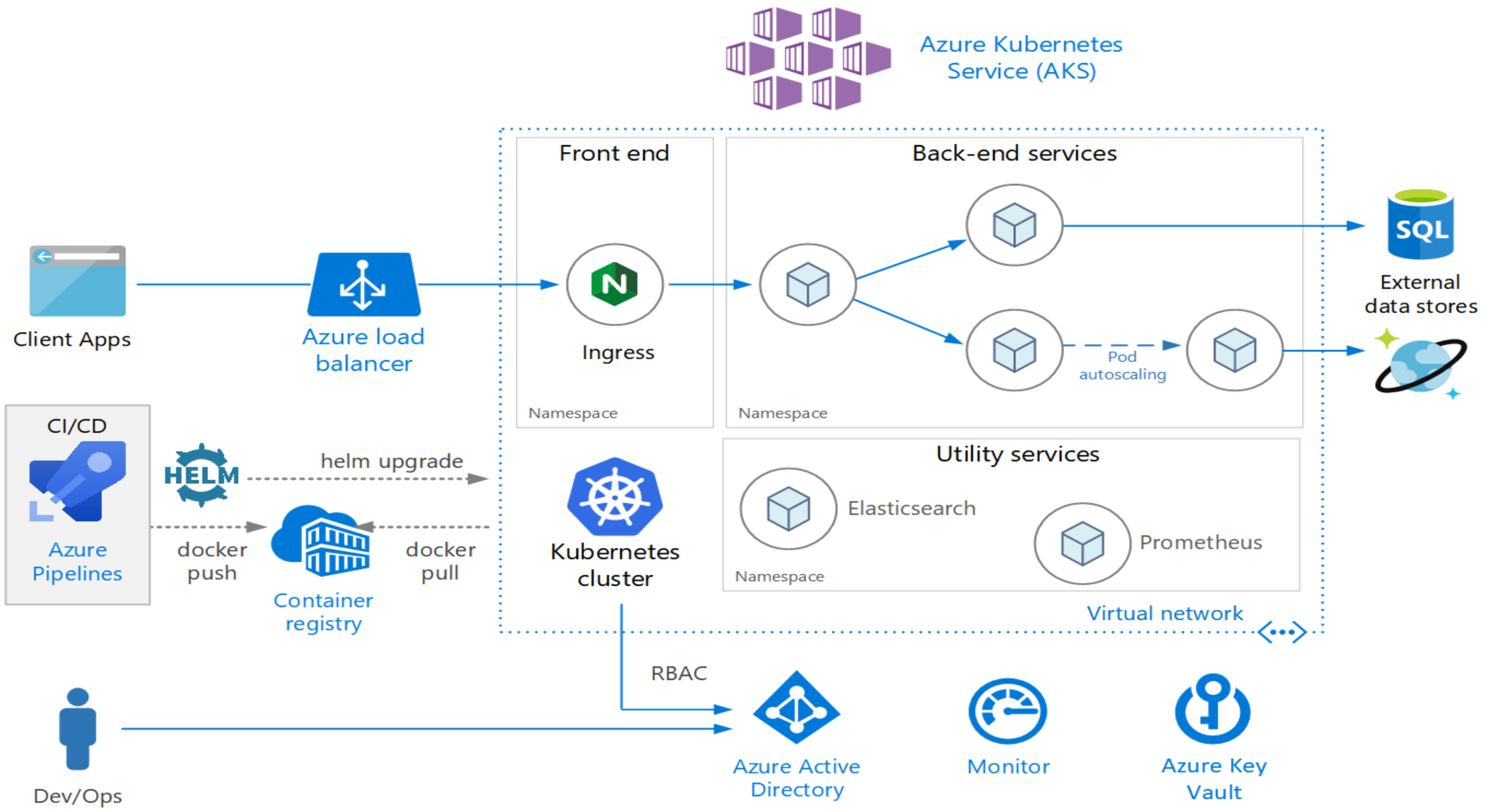
A custom resource is an extension of the Kubernetes API that is not necessarily available in a default Kubernetes installation - It represents a customization.

Operators are software extensions to Kubernetes that make use of custom resources to manage applications and their components. Operators follow Kubernetes principles, notably the “control loop”

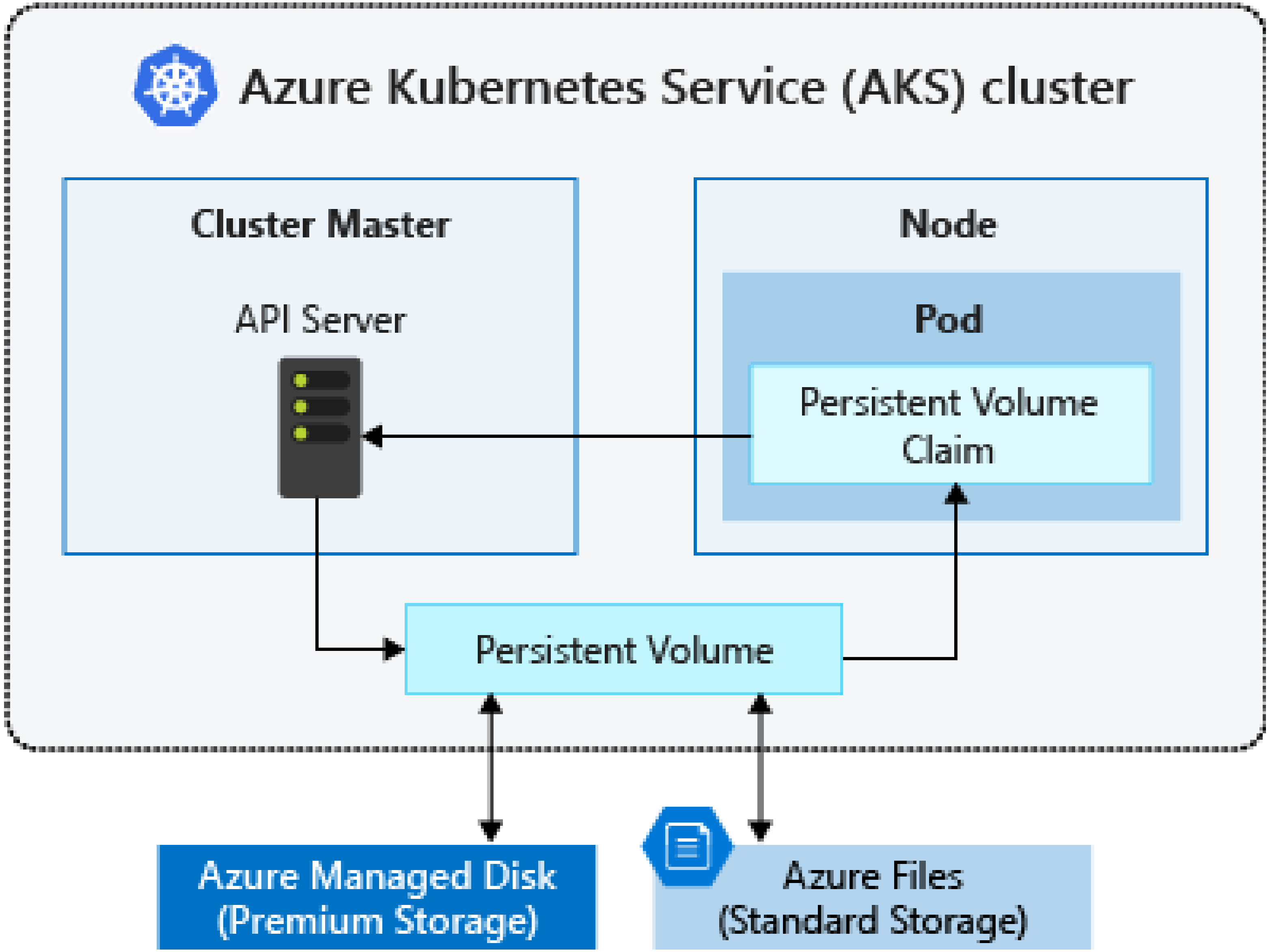
Understanding Kubernetes Objects | Kubernetes

Operator pattern | Kubernetes

Microservices architecture on Azure Kubernetes Service (AKS)



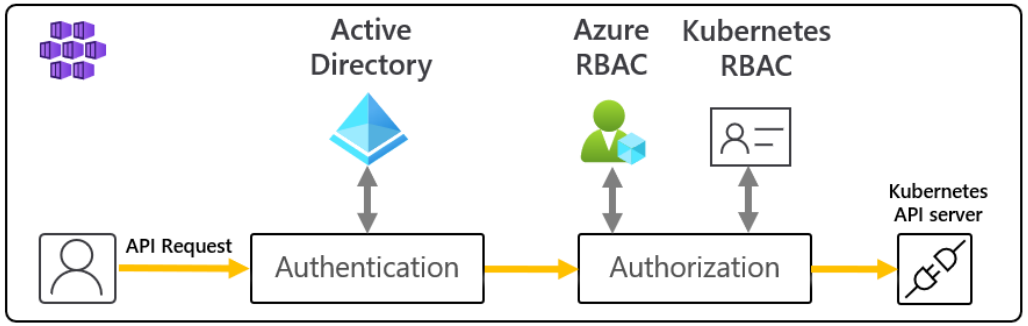
https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/containers/aks-microservices/aks-microservices

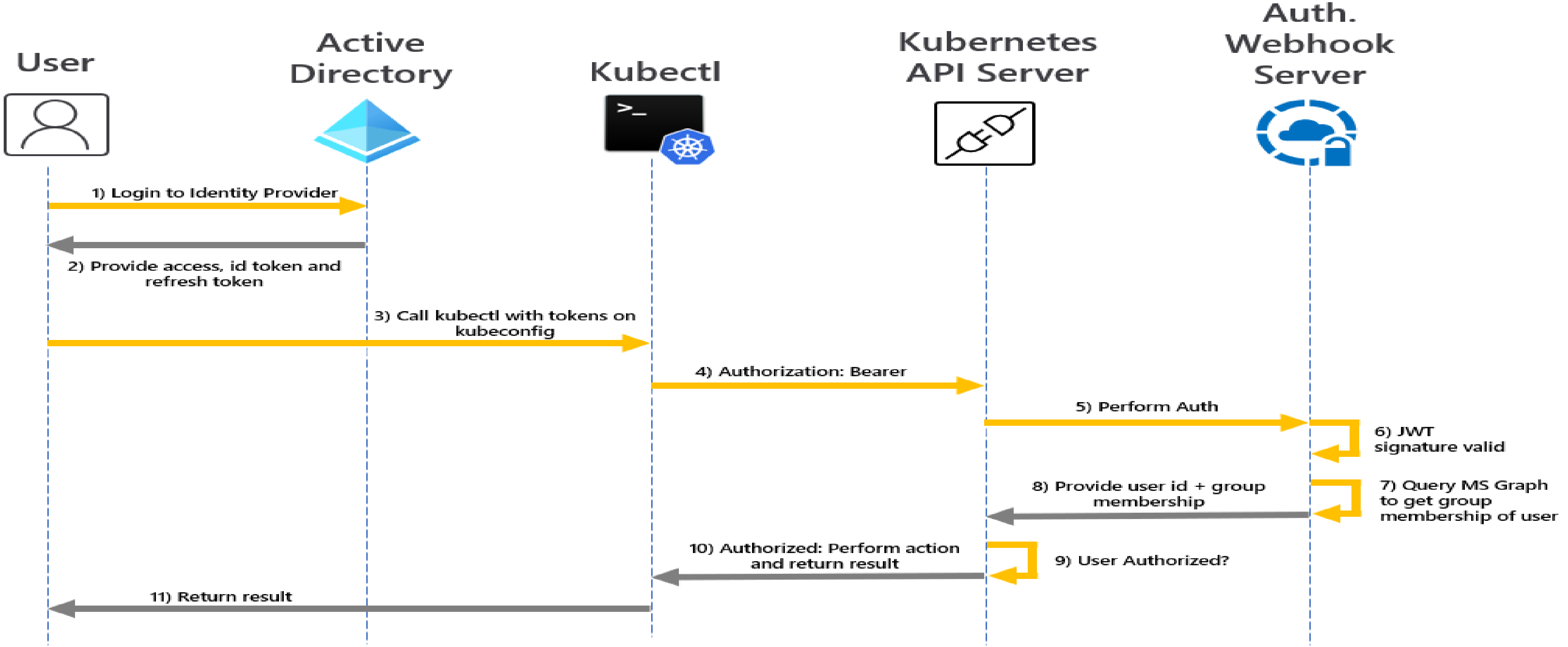

Azure Kubernetes Service (AKS) cluster
Cluster Master
API Server
Node
Pod
Persistent Volume Claim
Persistent Volume
Azure Managed Disk (Premium Storage)
Azure Files (Standard Storage)

Storage in K8S/AKS

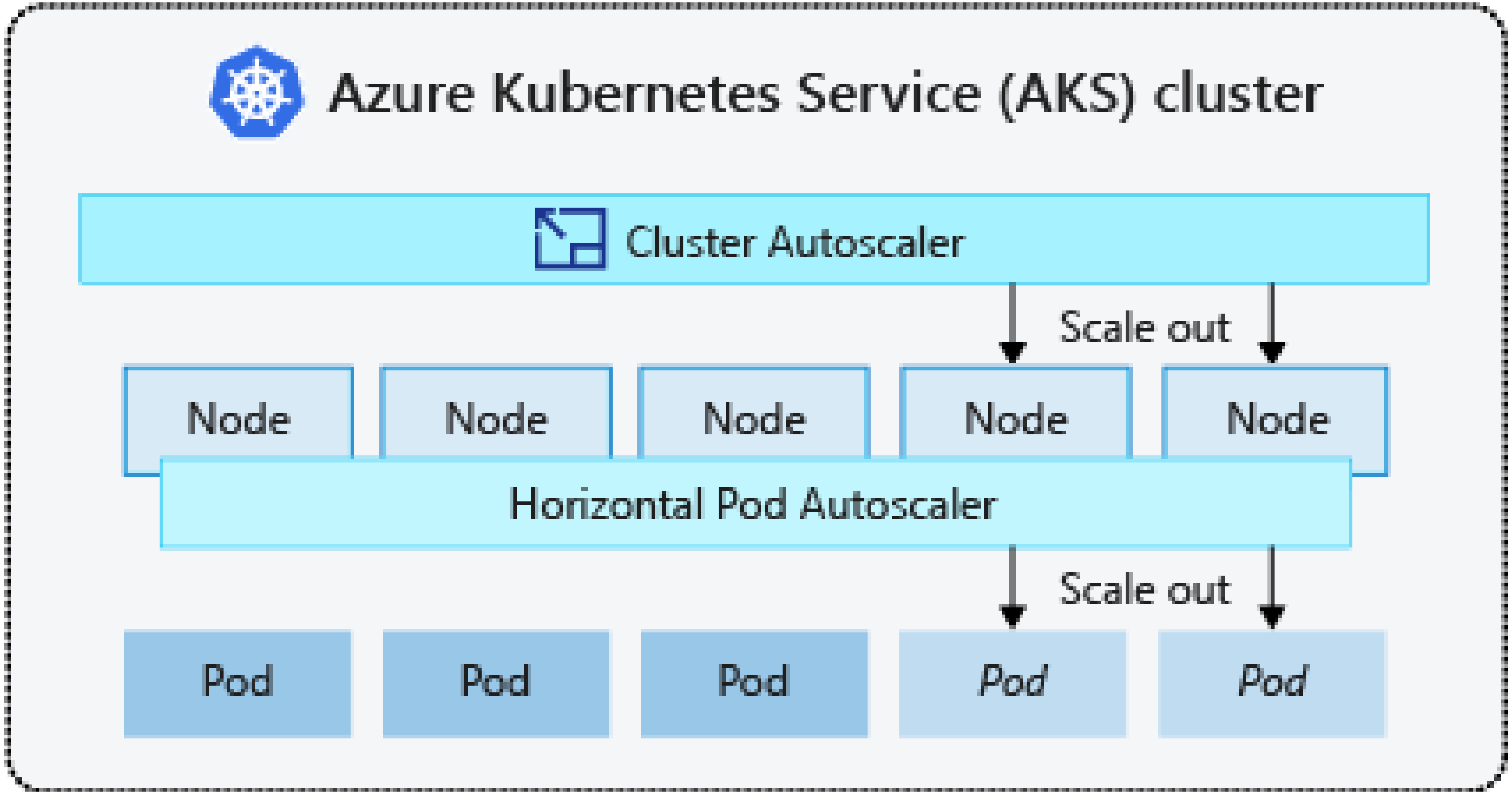
https://docs.microsoft.com/en-us/azure/aks/concepts-storage

Authentication & Authorization




User
Active Directory
Kubectl
Kubernetes API Server
Auth. Webhook Server
1) Login to Identity Provider
2) Provide access, id token and refresh token
3) Call kubectl with tokens on kubeconfig
4) Authorization: Bearer
5) Perform Auth
6) JWT signature valid
7) Query MS Graph to get group membership of user
8) Provide user id + group membership
9) User Authorized?
10) Authorized: Perform action and return result
11) Return result

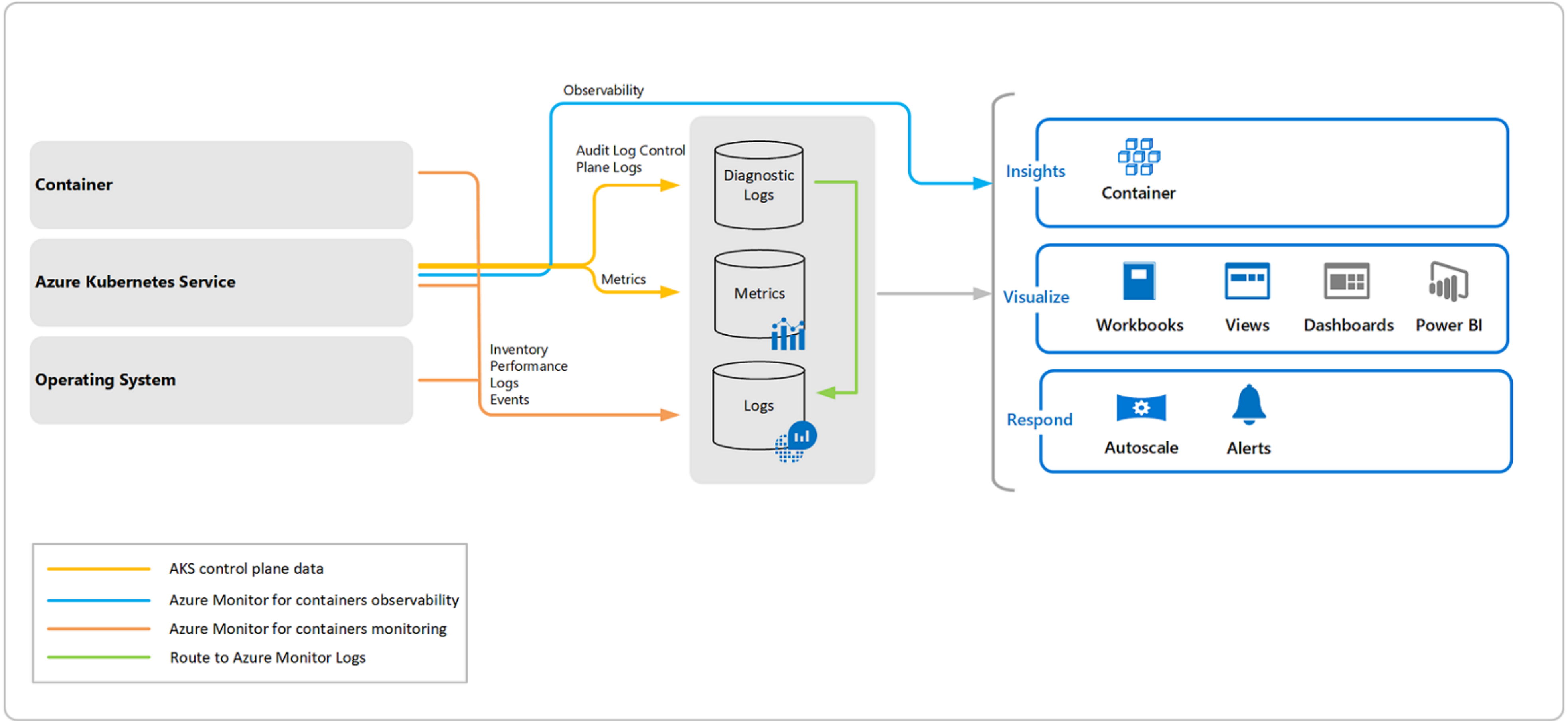
https://docs.microsoft.com/en-us/azure/aks/concepts-identity



Scaling options for applications in Azure Kubernetes Service (AKS)

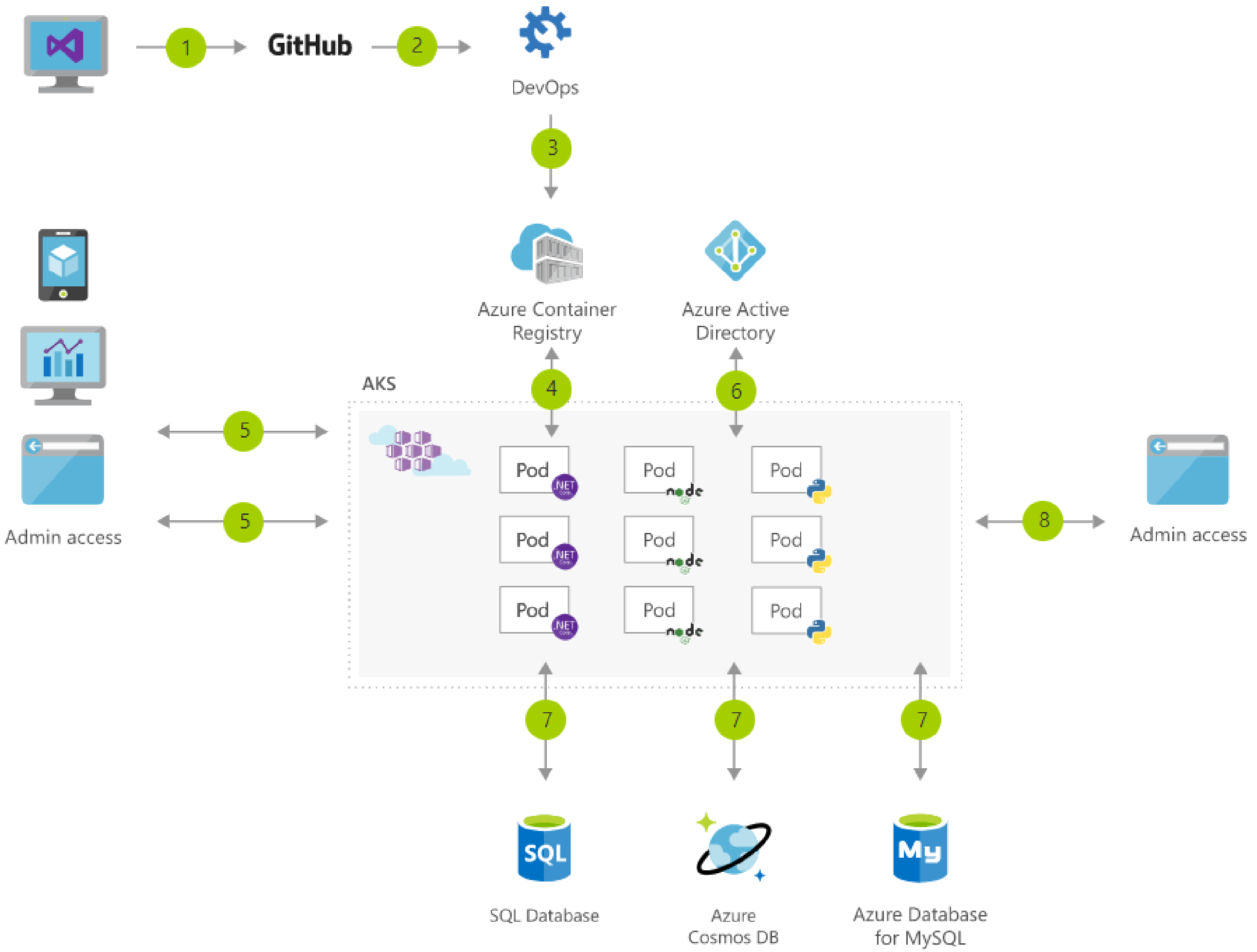
https://docs.microsoft.com/en-us/azure/aks/concepts-scale

Monitoring in AKS (Azure Monitor)



https://docs.microsoft.com/en-us/azure/azure-monitor/insights/container-insights-overview

Deploying with AKS (DevOps)



1. Use an IDE, such as Visual Studio, to commit changes to GitHub.
2. GitHub triggers a new build on Azure DevOps
3. Azure DevOps packages microservices as containers and pushes them to the Azure Container Registry
4. Containers are deployed to AKS cluster
5. Azure Active Directory is used to secure access to the resources
6. Users access services via apps and websites
7. Administrators access the apps via a separate admin portal
8. Microservices use databases to store and retrieve information

https://azure.microsoft.com/en-in/solutions/architecture/microservices-with-aks/

A note about Service Mesh

- A service mesh provides capabilities like traffic management, resiliency, policy, security, strong identity, and observability to your workloads.
- Your application is decoupled from these operational capabilities and the service mesh moves them out of the application layer, and down to the infrastructure layer.
- Examples: Istio, Consul, Linkerd

- About service meshes - Azure Kubernetes Service | Microsoft Docs

Resources

- K8s Learning Path
- Docker on Azure -
- Containerize your apps with Docker & Kubernetes
- Learning path for Containers & Kubernetes on MS Learn
- Git Hub Repo (Labs)
- K8S Cheat Sheet
- AKS Best Practices
- Kubernetes in the Cloud Adoption Framework - Cloud Adoption Framework