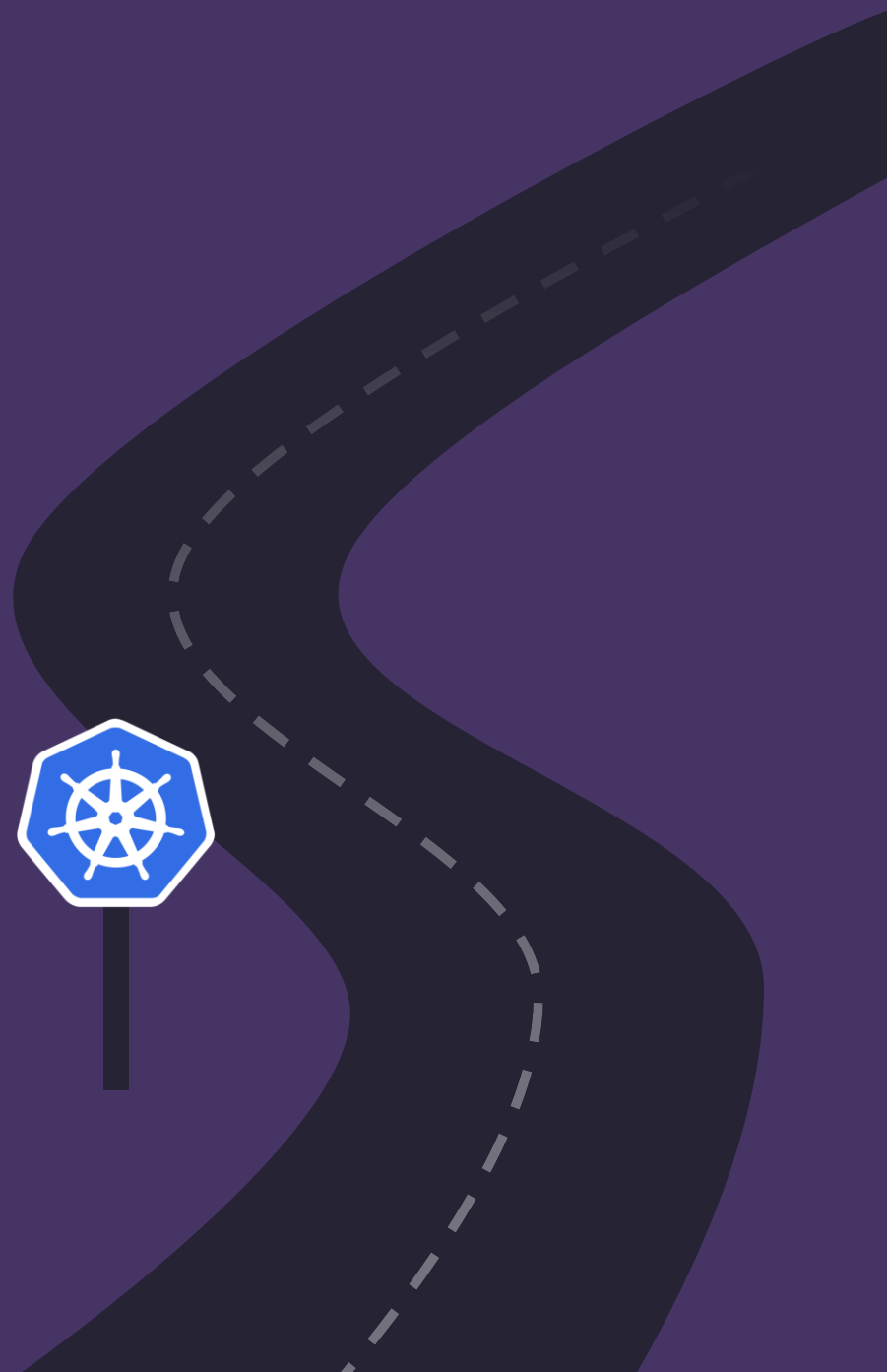
Kubernetes

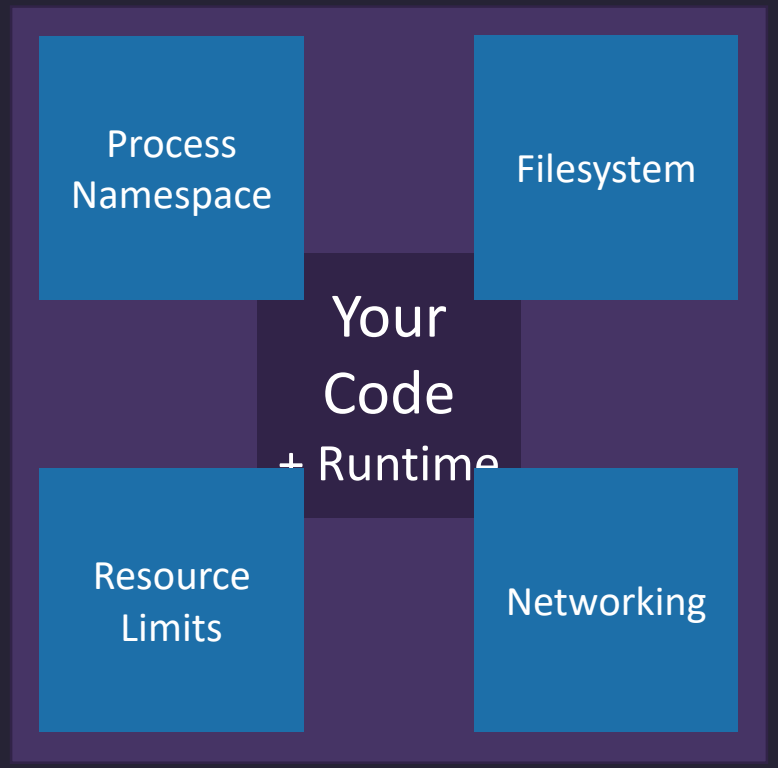
Going 0 to 100



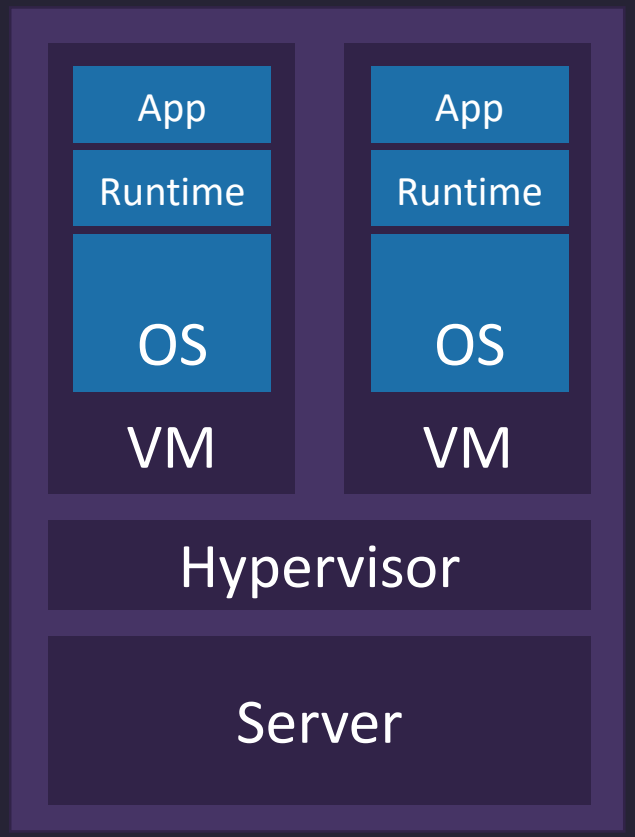
Scott Holden

Let's Revisit Containers

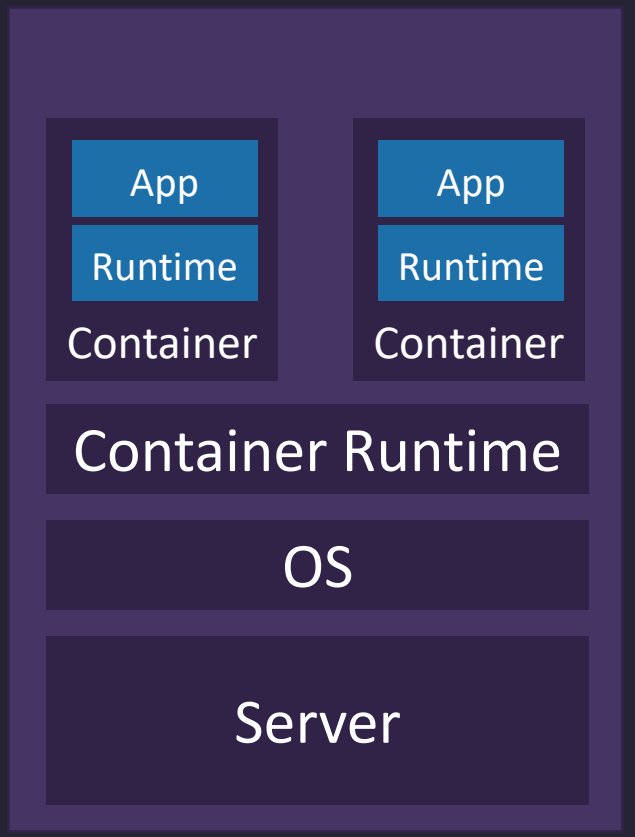
Sandbox



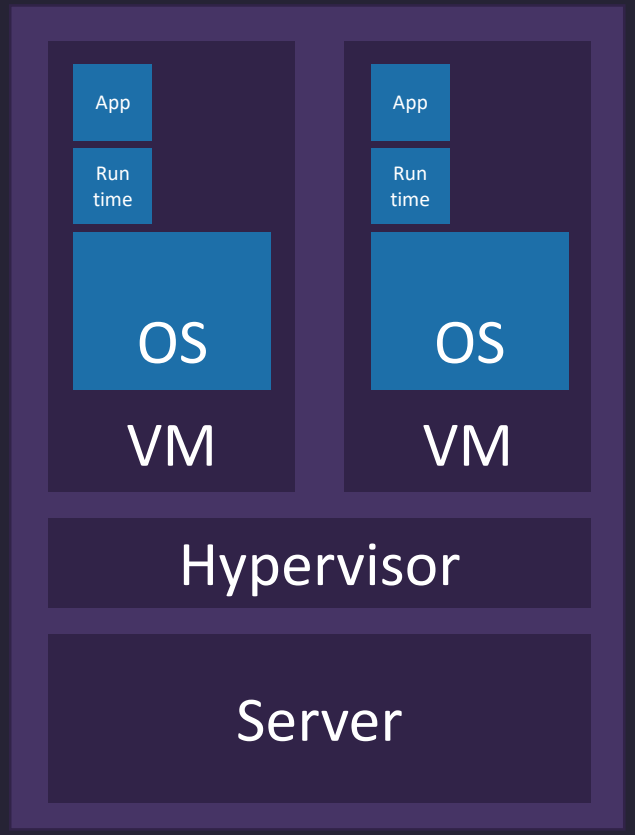
Container



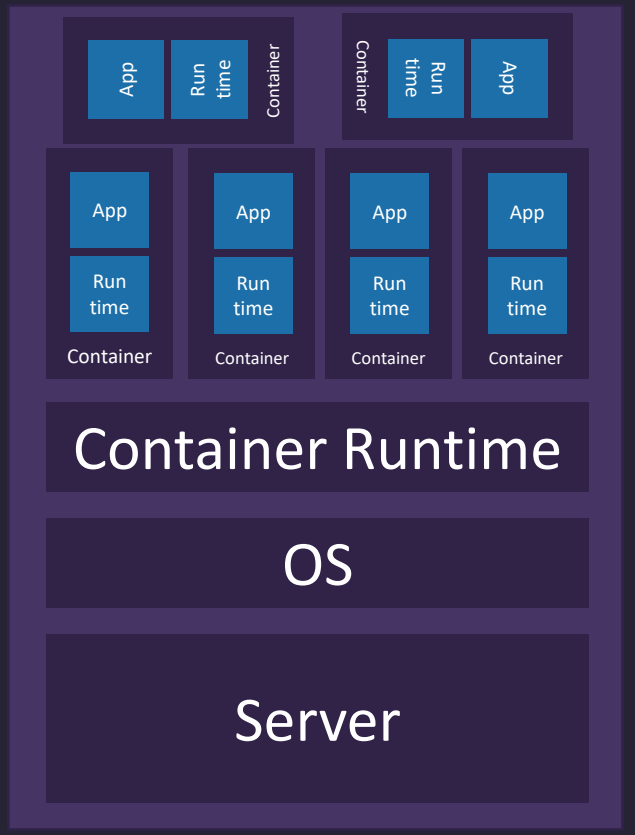
Virtual Machine



Container



Virtual Machine



Container

Container Image

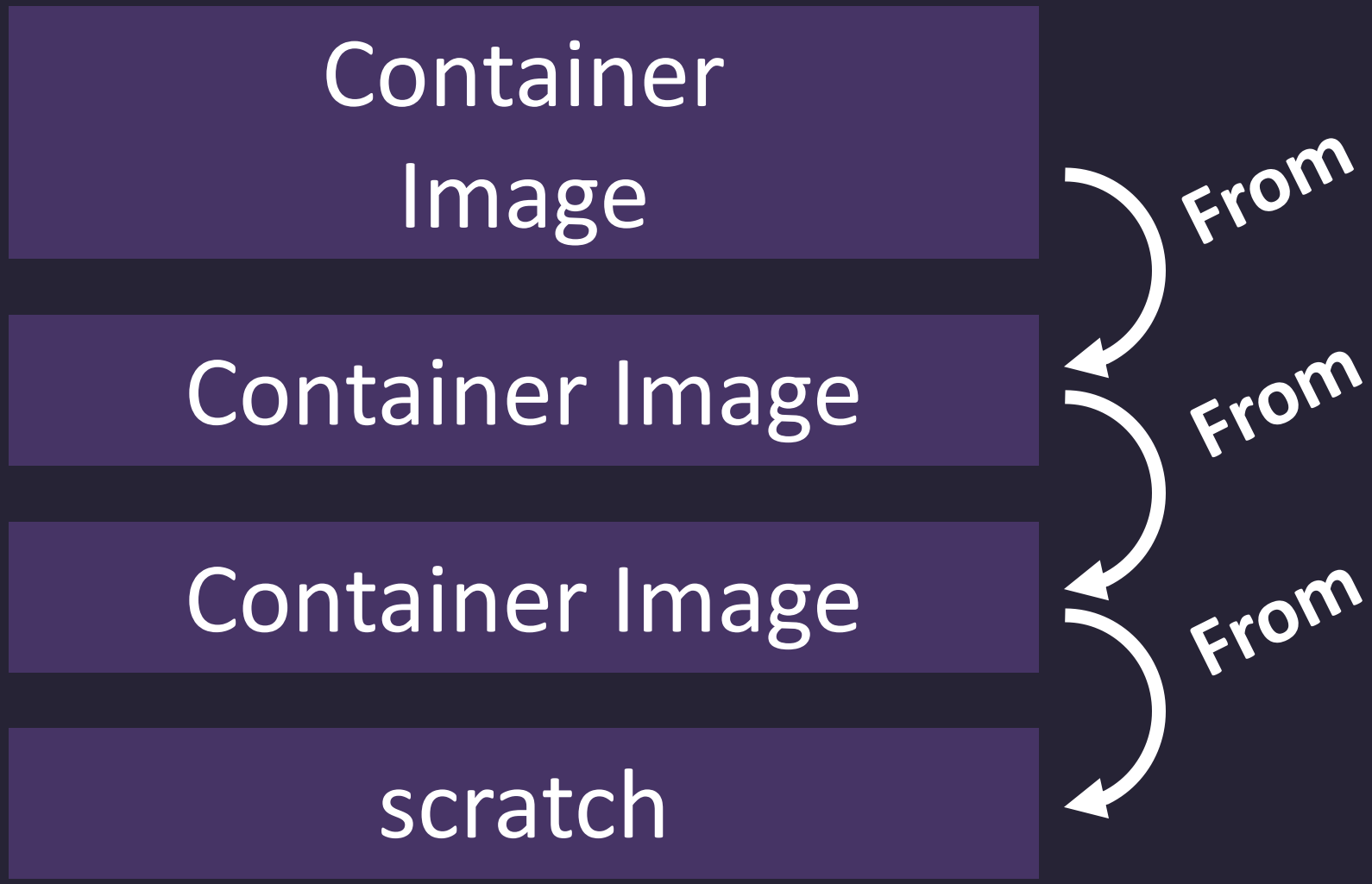
+ Configuration

Container Runtime

Running Container

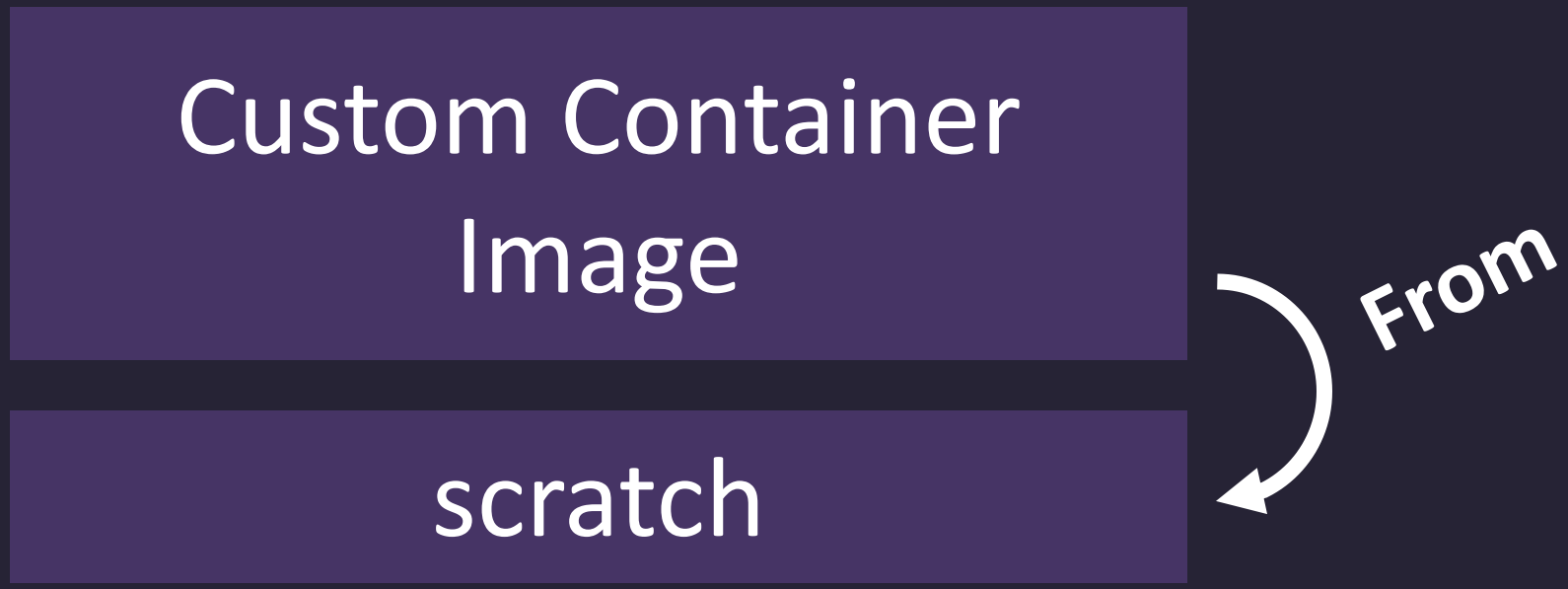
Demo:
Running A Simple Container

Container Image

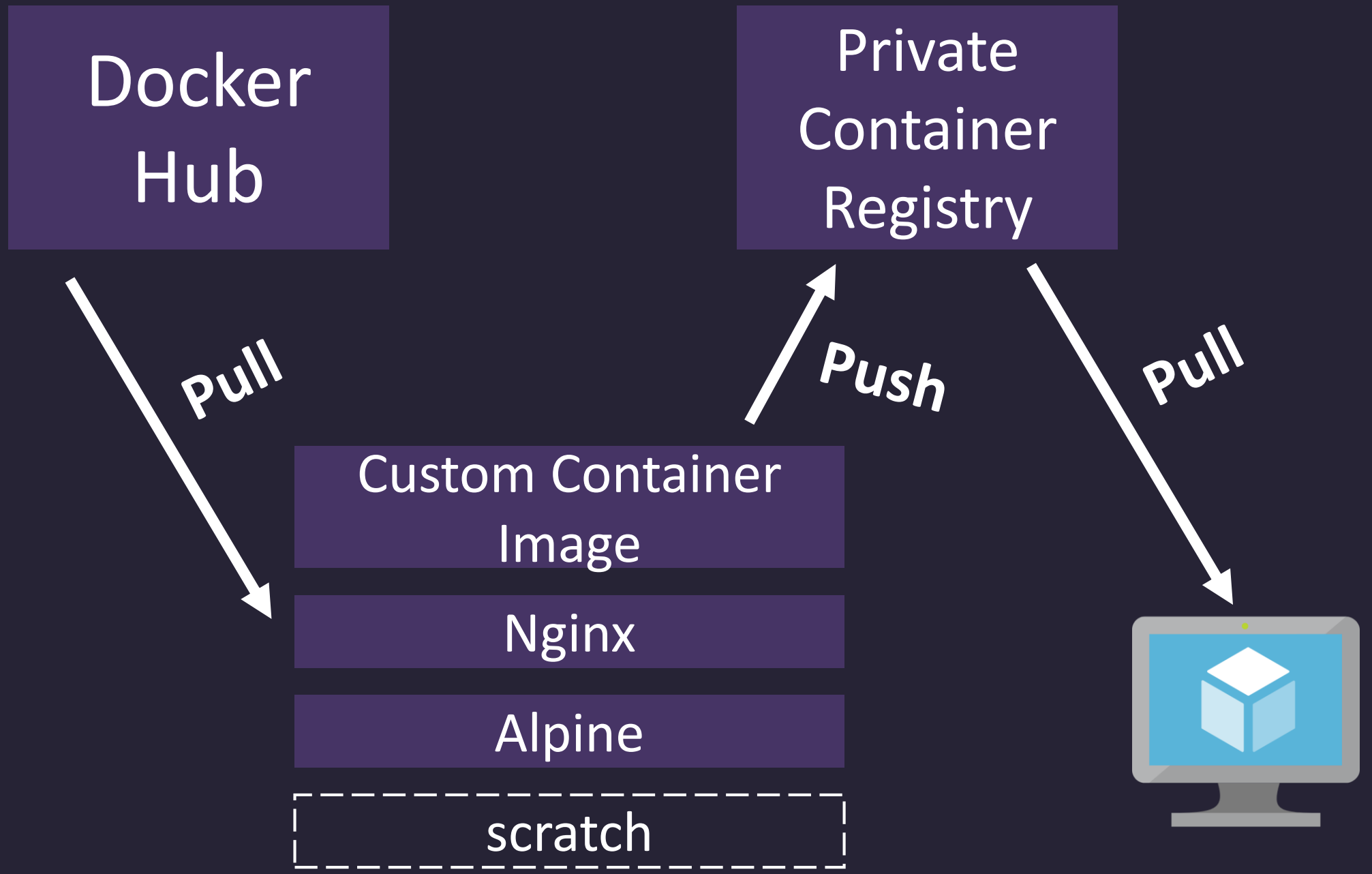

Container Image
Container Image
Container Image
scratch
From
From
From


Custom Container Image
Nginx
Debian
scratch
+1KB
+56MB
+69MB
Nothing
From
From
From


Other Container Image
Custom Container Image
From
From
From
From
Nginx
Alpine
scratch


Custom Container Image
From
scratch

Demo:
Making A Container Image


Docker Hub
Private Container Registry
Pull
Push
Pull
Custom Container Image
Nginx
Alpine
scratch

What's The Challenge?

One container, one host

Manageable

A few containers, a few hosts

Getting tricky

Lots of containers, many hosts

More than a full time job!

One host fails

  Move the container

Containers move

  Service discovery

Is this container healthy?

  Monitoring & recovery

- Scaling in & out
  - Which host?
- Deploying new versions
  - How to update?
- Required configuration
  - Check every host

Introducing Kubernetes

Scheduling

Affinity/anti-affinity

Health monitoring

Failover

Scaling

Networking

Service discovery

Coordinated app upgrades

Kubernetes is a Scheduler*

*In terms of running containers

Kubernetes
Masters & Nodes

Master (Node)

Runs Kubernetes
Made up of a few key services
Talks with the worker nodes
Normally a cluster

100+

Master Node

kube-apiserver
(Front end, what kubectl talks to)

kube-scheduler
(Decides what node gets what)

etcd*
(Internal backing store)

kube-controller-manager
(Runs the internal controllers)

cloud-controller-manager
(Underlying Cloud provider comms)

(Worker) Node

Some form of compute
Runs your workloads
Container runtime lives here

100+

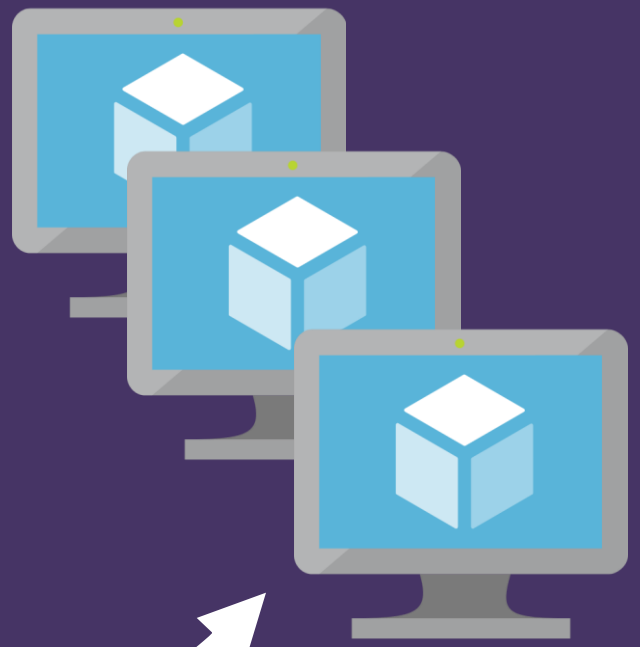
kubelet
(Controls what containers to run)

kube-proxy
(Sets up and maintains networking)

Container Runtime
(Docker, etc, This is what runs the containers)

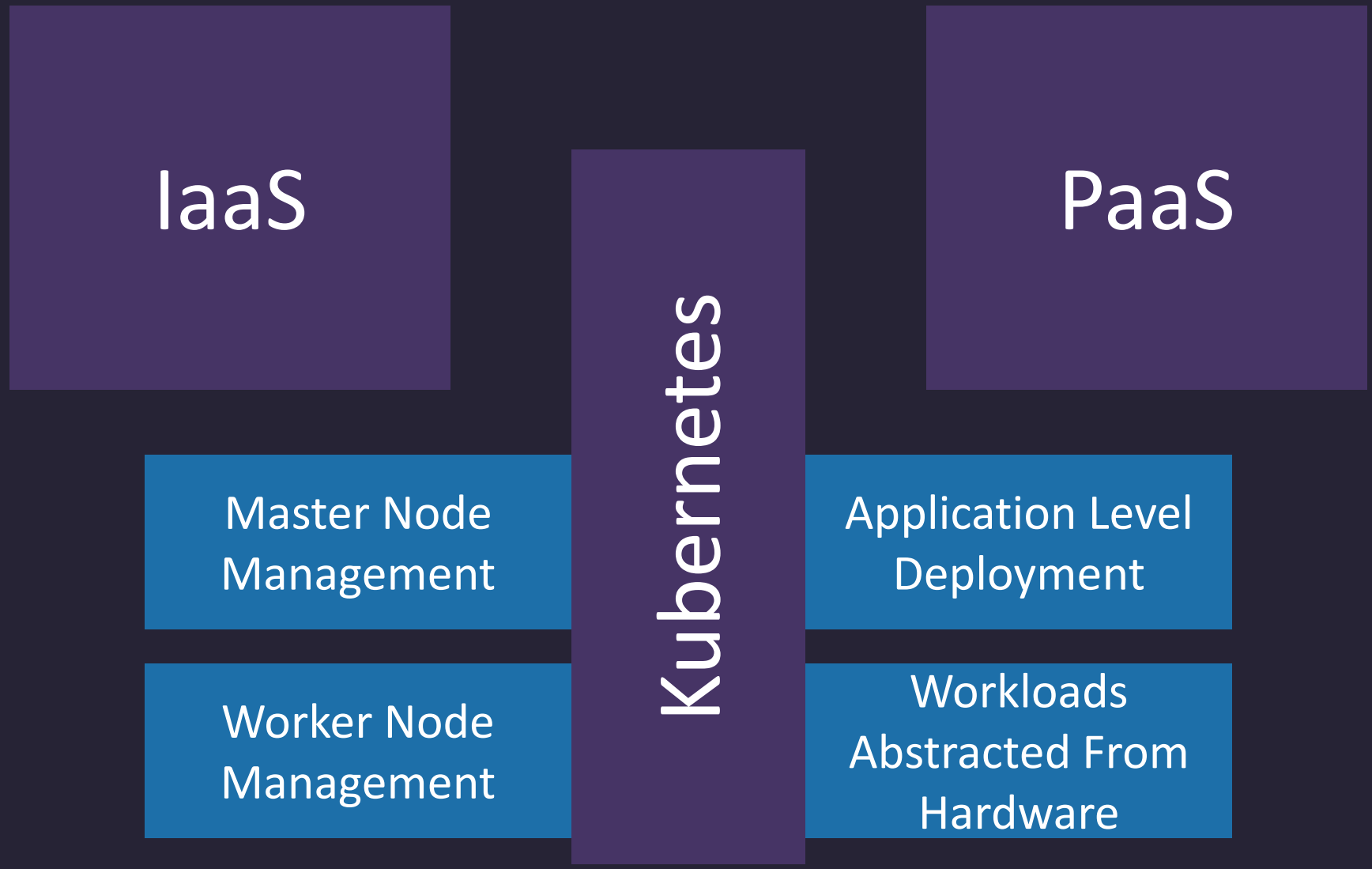
Worker Node

Master Nodes



Worker Nodes

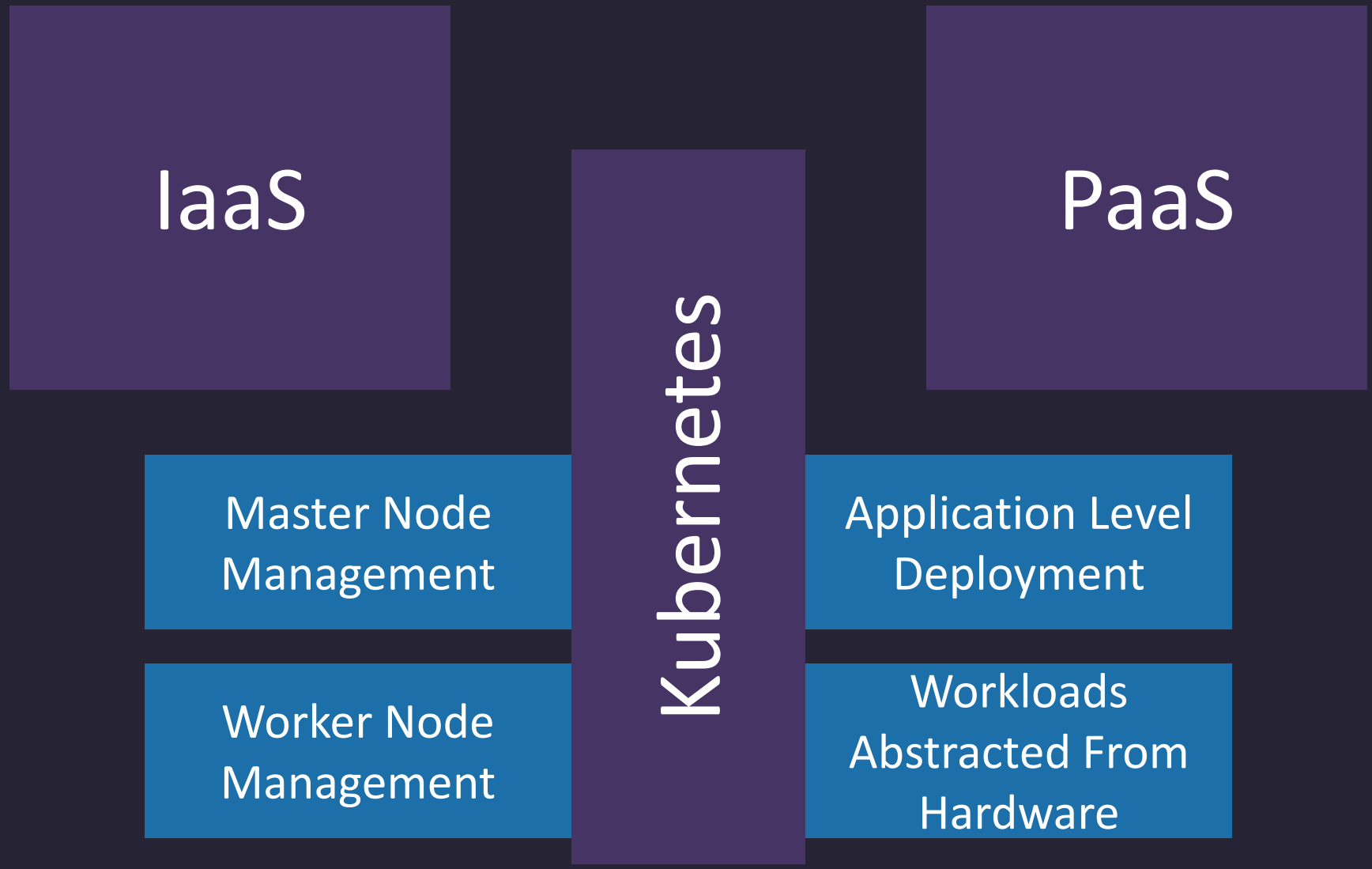
Control Plane

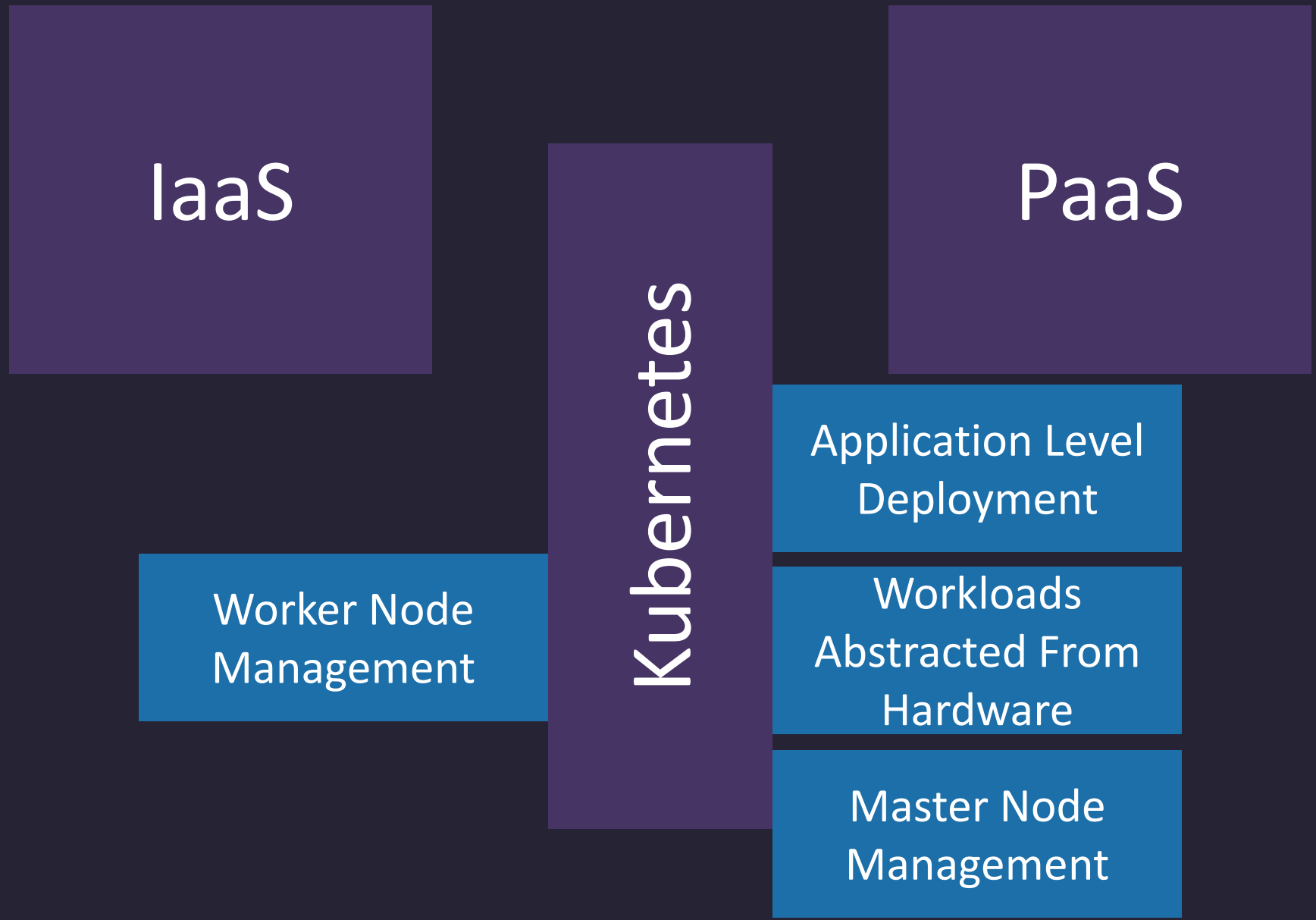

IaaS
PaaS
Kubernetes
Master Node Management
Worker Node Management
Application Level Deployment
Workloads Abstracted From Hardware

Cloud Managed Masters

Master Nodes ‘as a Service’

Azure Kubernetes Service
Google Container Engine

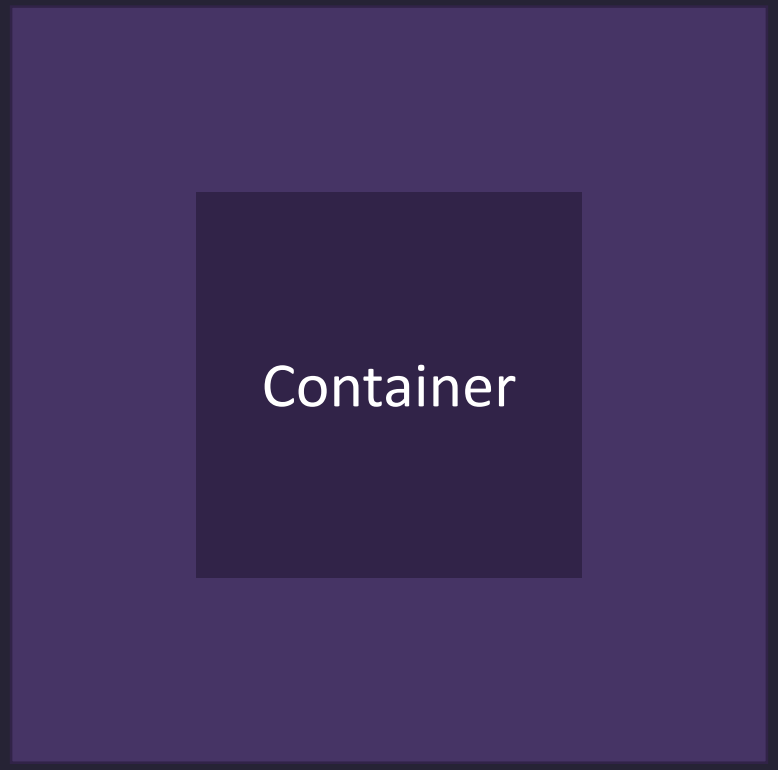

IaaS
PaaS
Kubernetes
Master Node Management
Worker Node Management
Application Level Deployment
Workloads Abstracted From Hardware


IaaS
Kubernetes
PaaS
Application Level Deployment
Workloads Abstracted From Hardware
Master Node Management
Worker Node Management

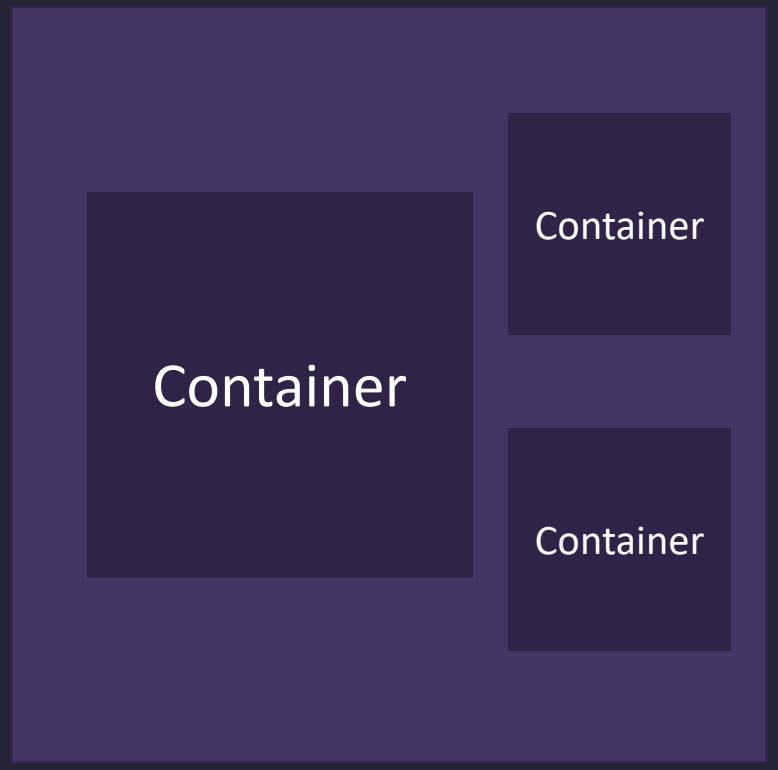
Kubernetes Nuts & Bolts

Pod

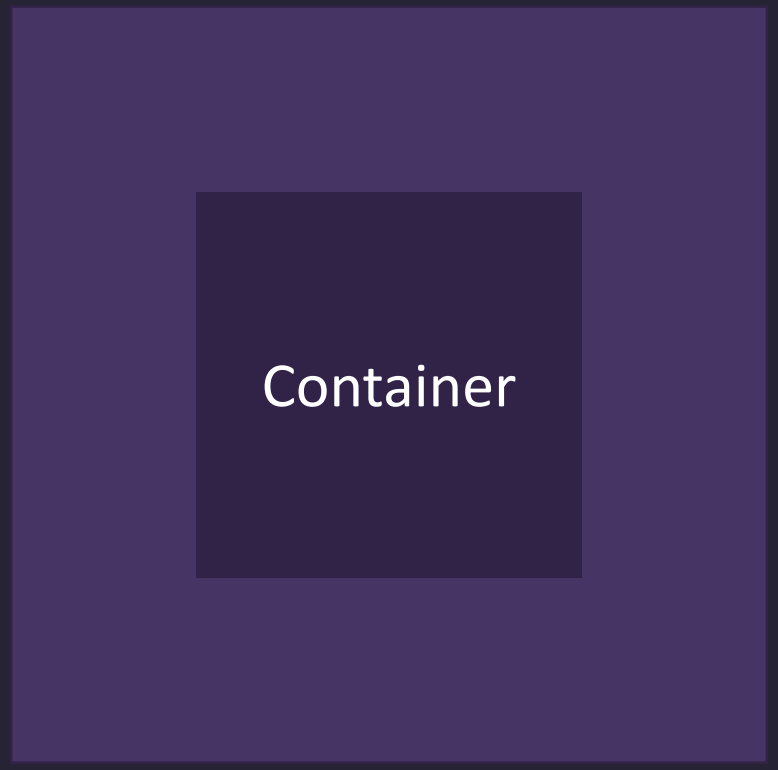
Lowest component
Made up of one or more containers
Single instance



Pod



Pod



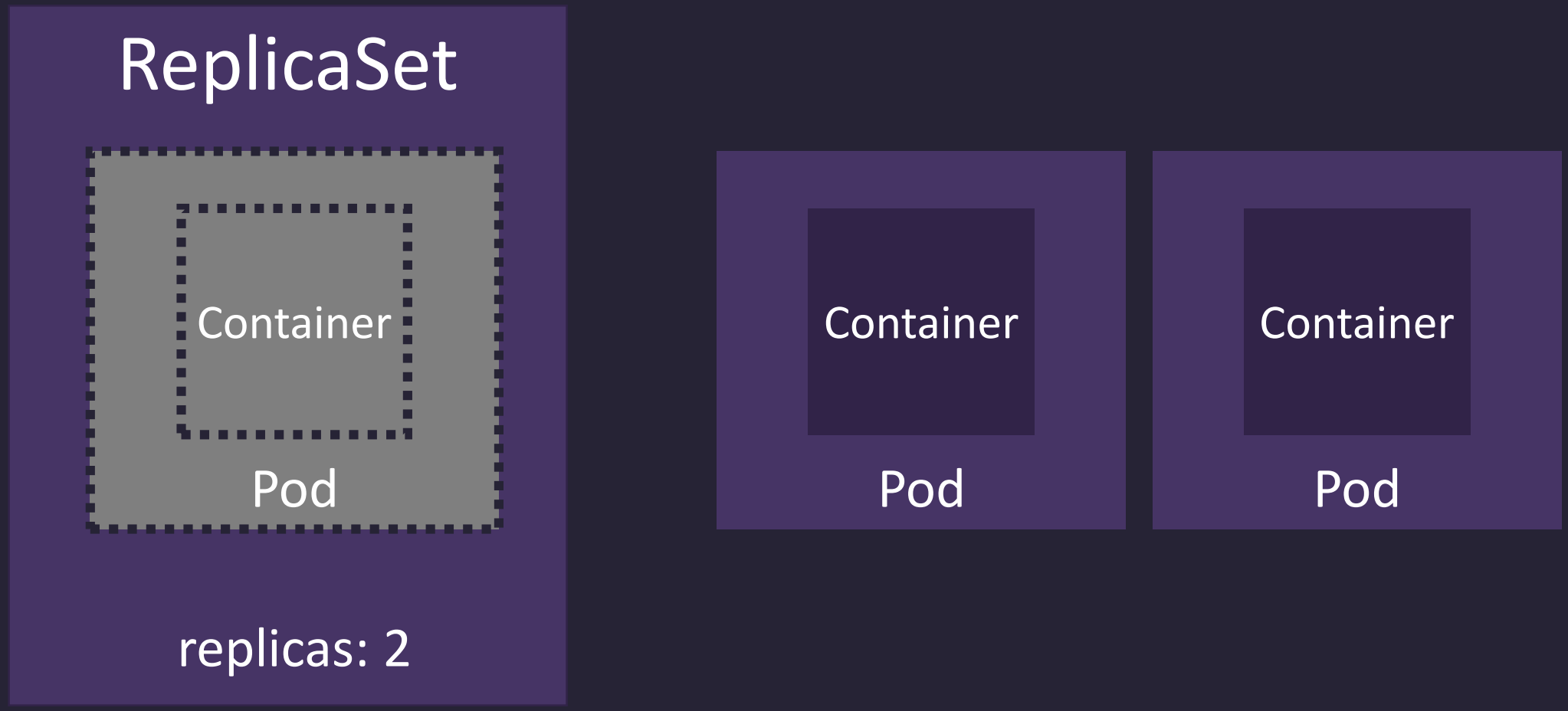
Pod

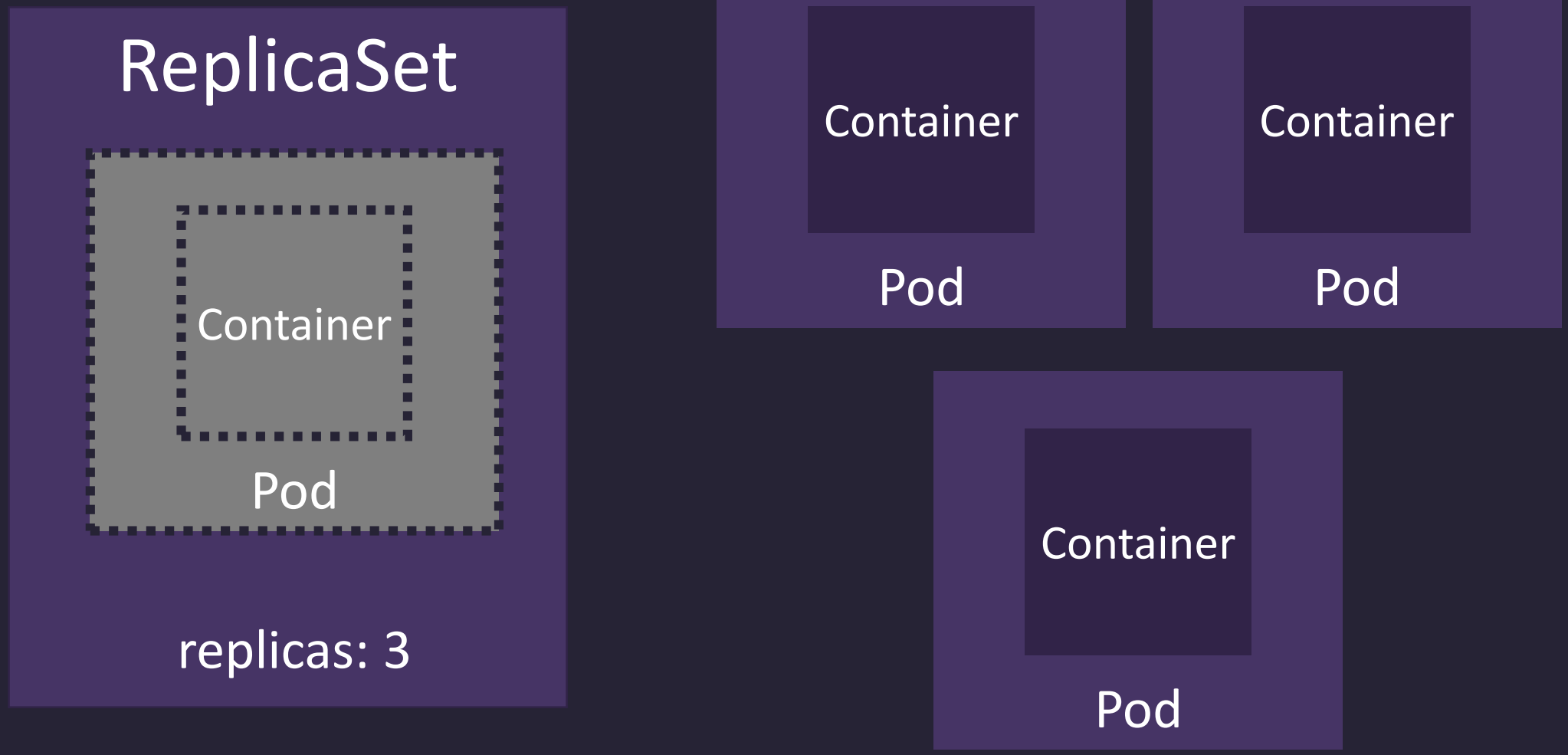
Demo:
Single Pod

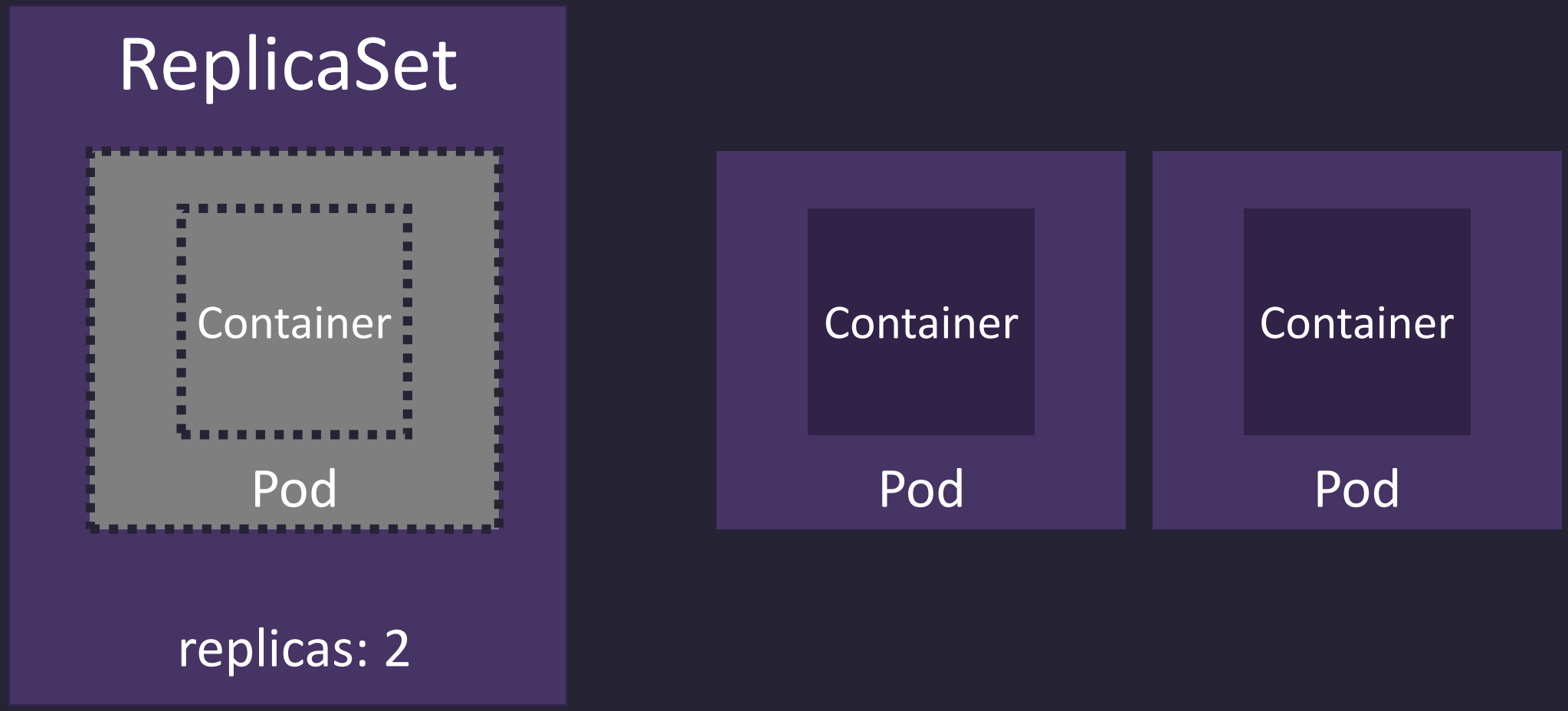
ReplicaSet

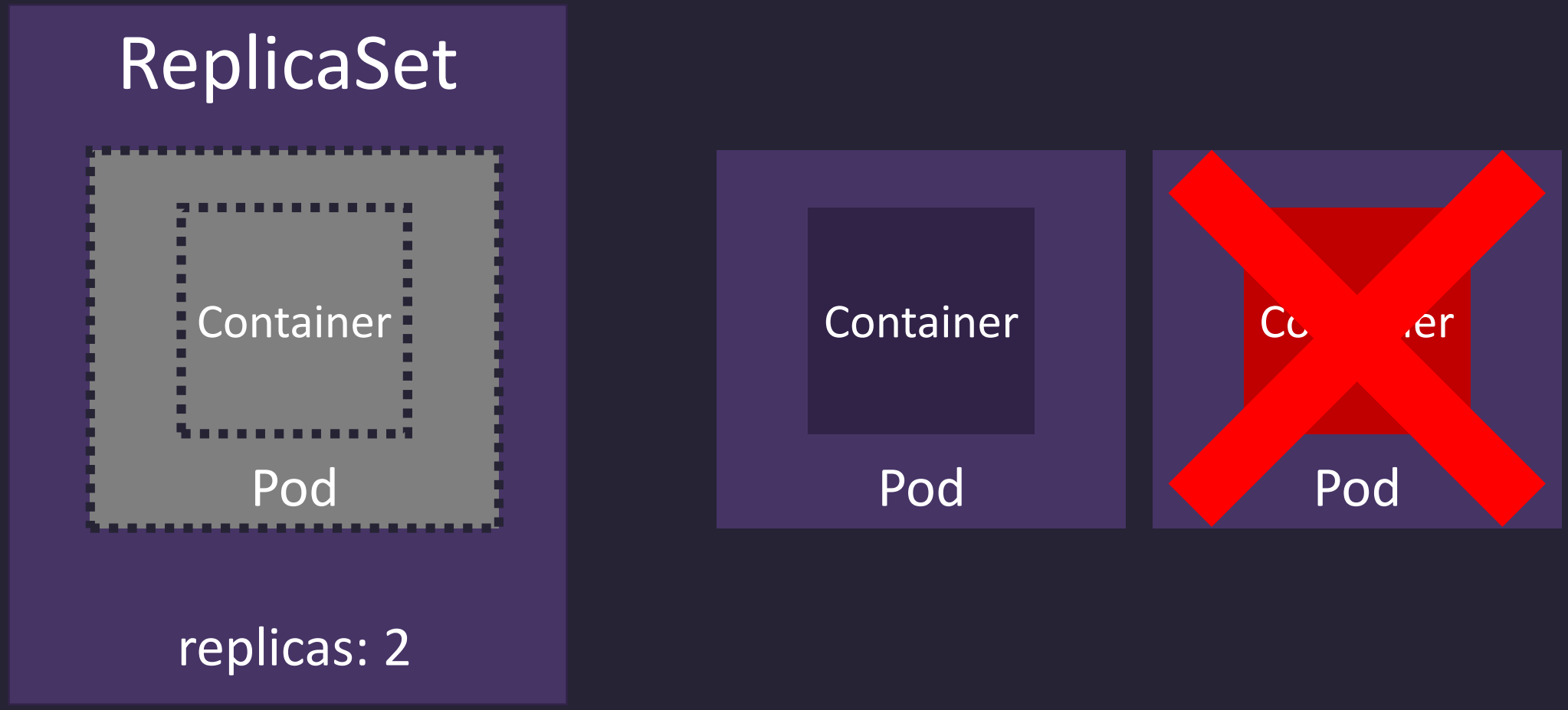
Sole purpose to maintain a set of pods

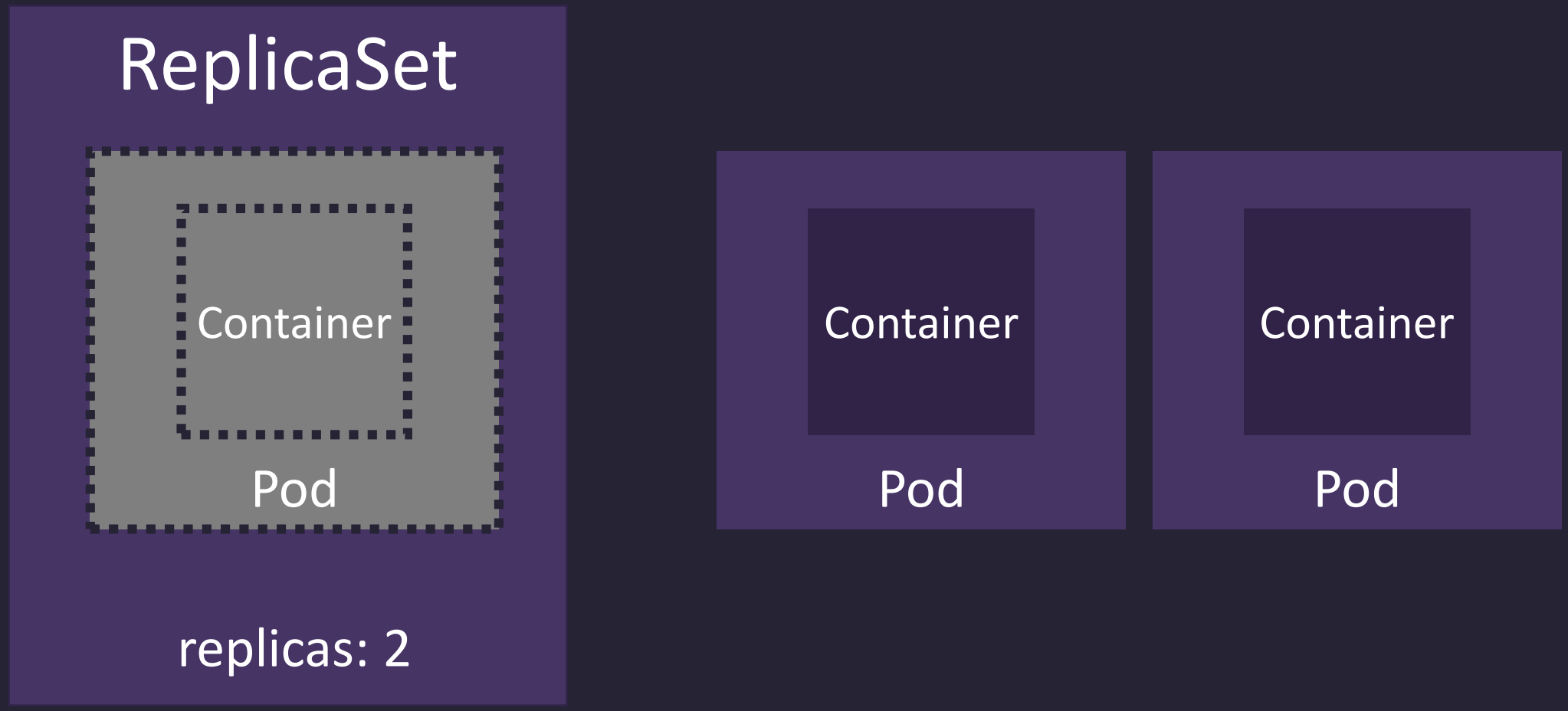
Templated definition of a pod

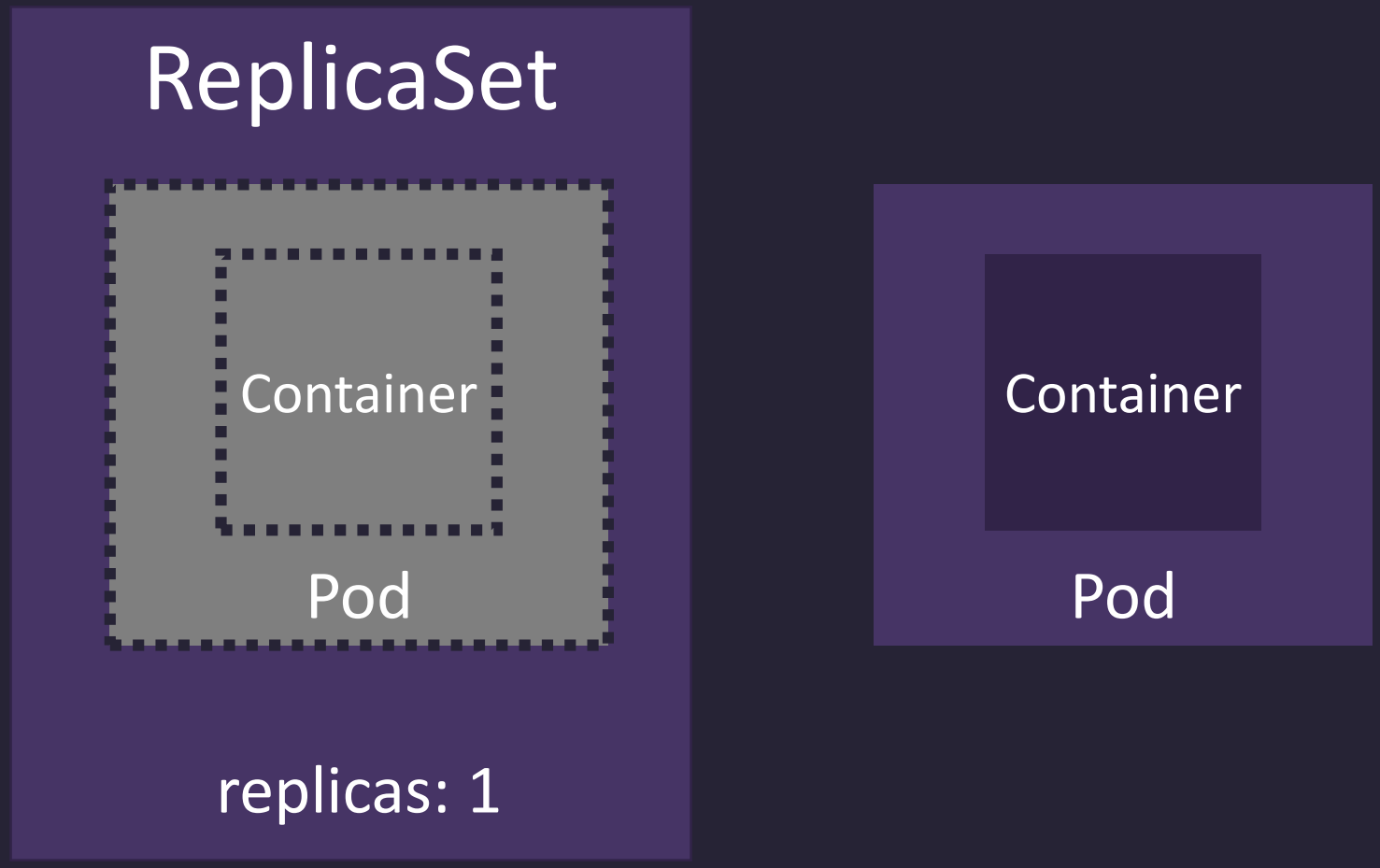

ReplicaSet
Container
Pod
replicas: 2
Container
Pod
Container
Pod


ReplicaSet
Container
Pod
replicas: 3
Container
Pod
Container
Pod
Container
Pod


ReplicaSet
Container
Pod
replicas: 2
Container
Pod
Container
Pod


ReplicaSet
Container
Pod
replicas: 2
Container
Pod
Pod


ReplicaSet
Container
Pod
replicas: 2
Container
Pod
Container
Pod


ReplicaSet
Container
Pod
replicas: 1
Container
Pod

Demo:
ReplicaSet

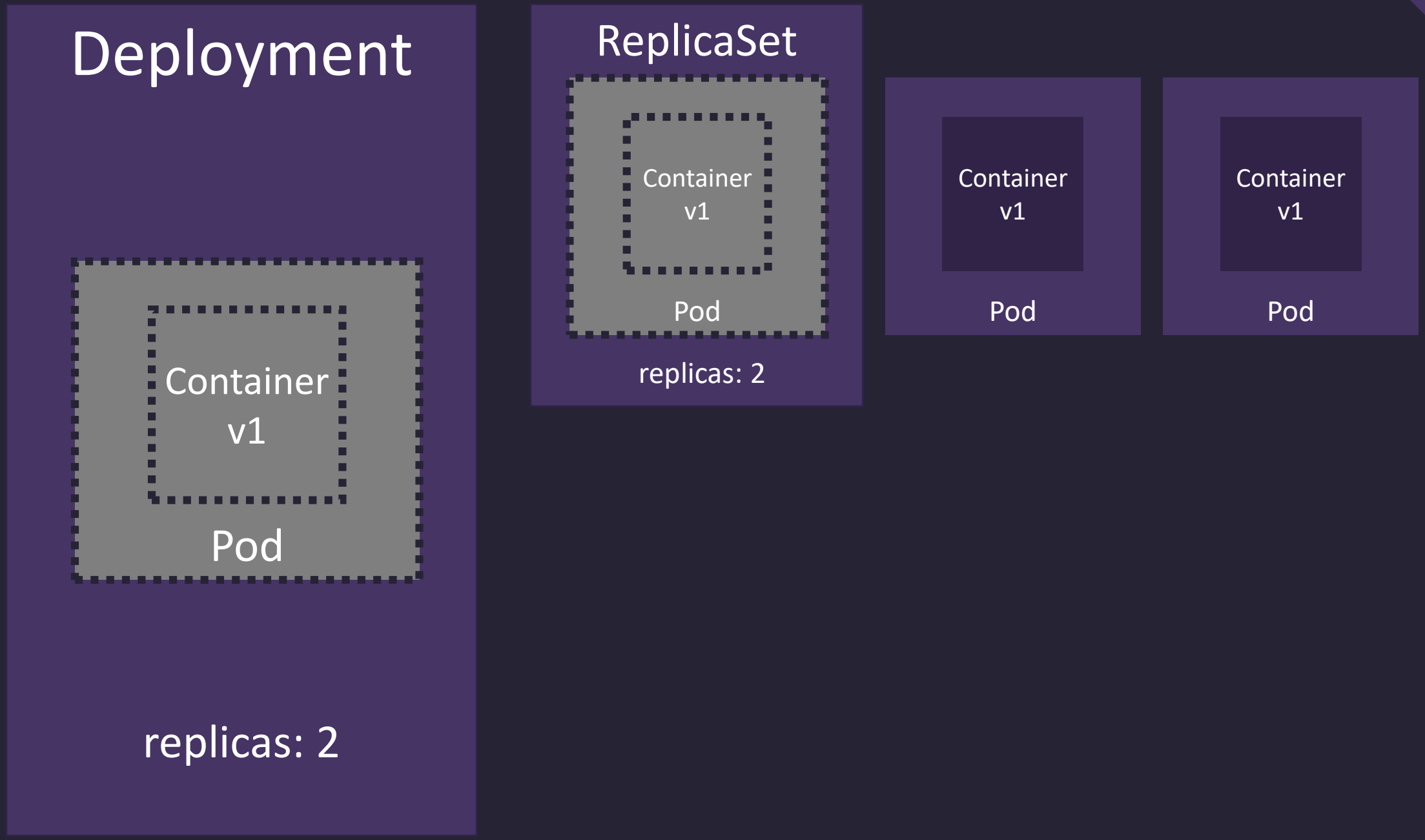
100+

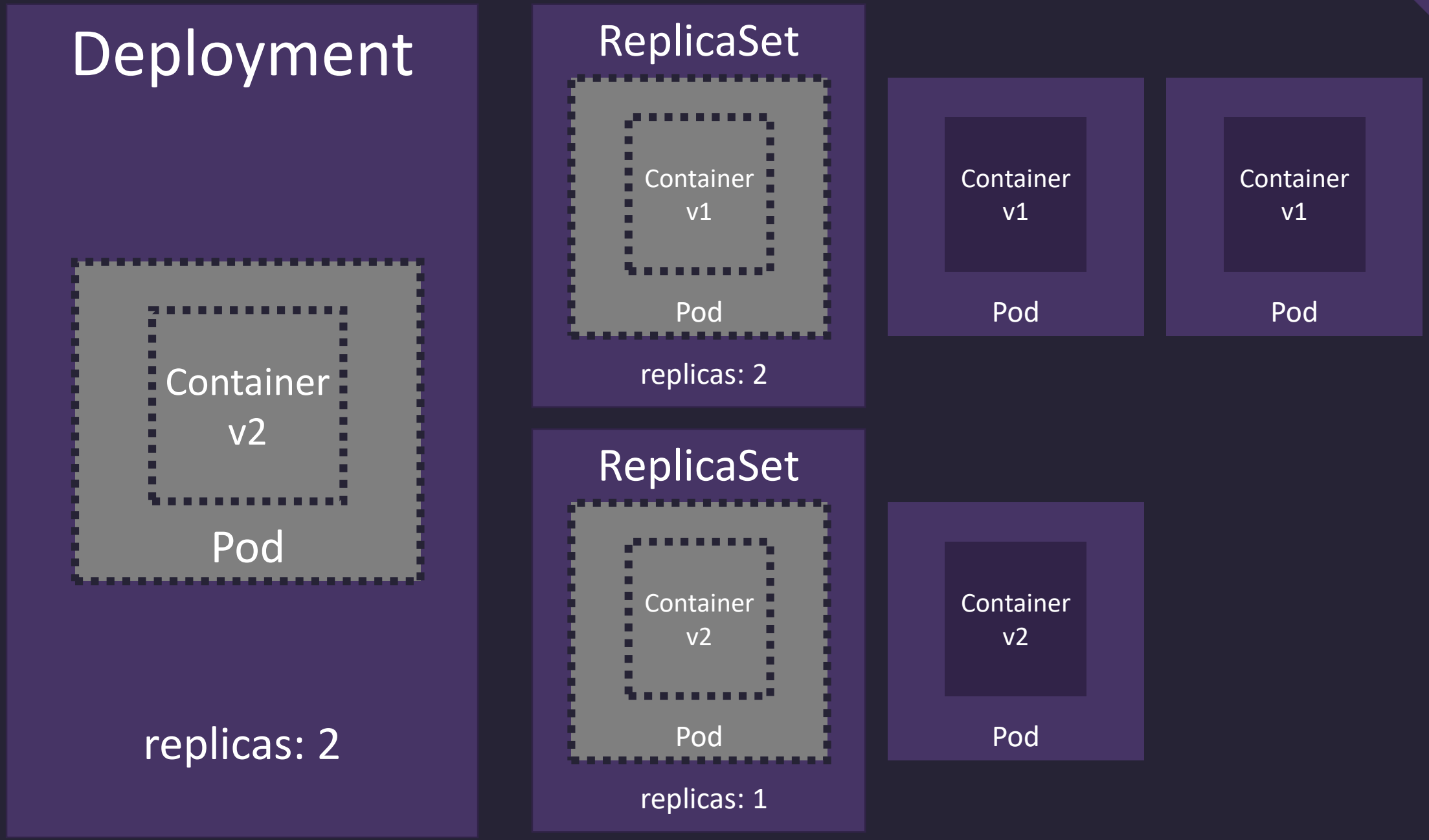
Other Types

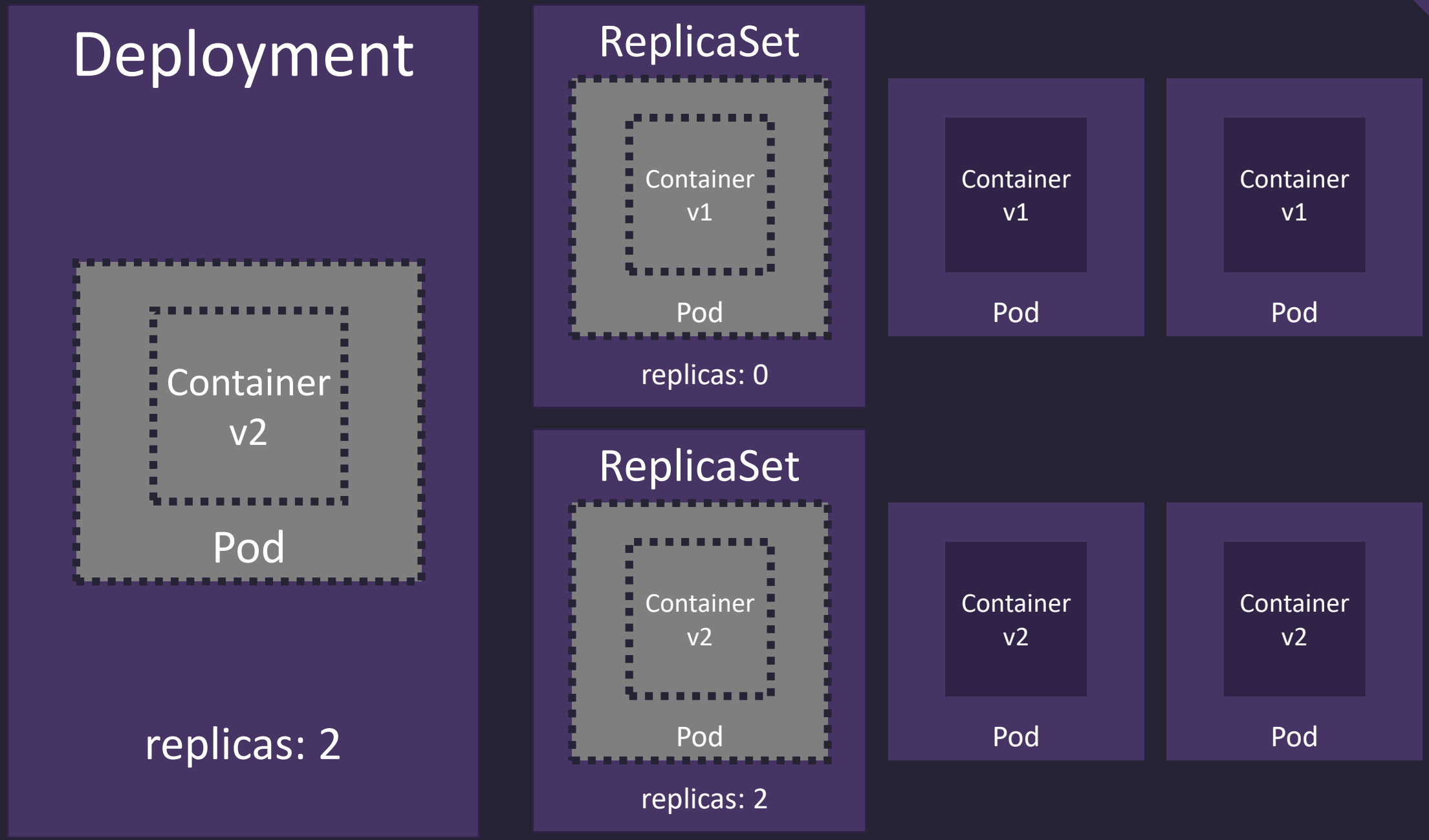
ReplicaSet – Replicas of a Pod
Job – Run & terminate
StatefulSet – Sticky identity for Pods
DaemonSet – Linked to Node lifecycle

Deployment

Level above a ReplicaSet
Designed to handle updates
  via multiple ReplicaSet’s
Deploy & Rollback


Deployment
Container
v1
Pod
replicas: 2
ReplicaSet
Container
v1
Pod
replicas: 2
Container
v1
Pod
Container
v1
Pod

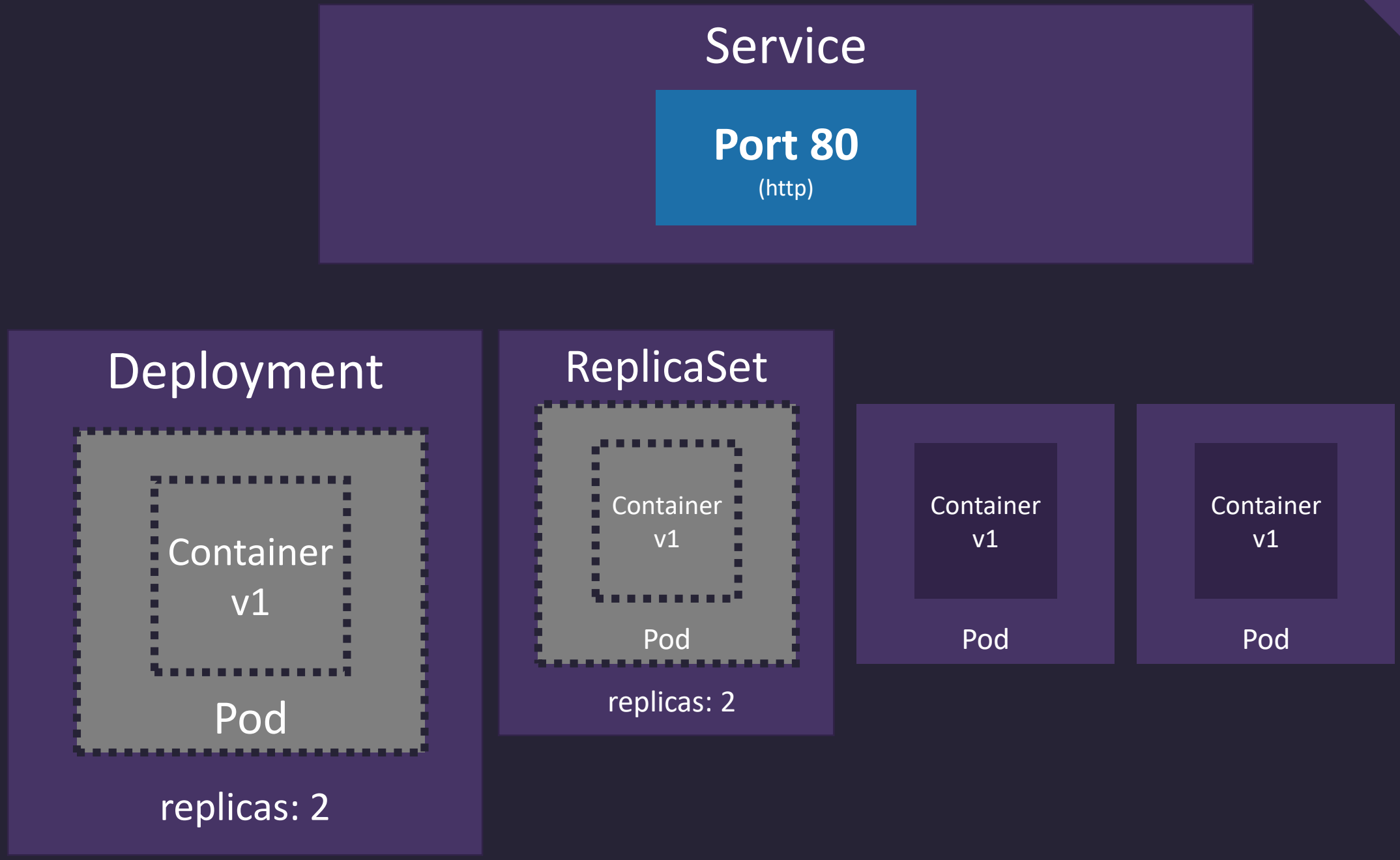

Deployment
Container
v2
Pod
replicas: 2
ReplicaSet
Container
v1
Pod
replicas: 2
Container
v1
Pod
Container
v1
Pod
ReplicaSet
Container
v2
Pod
replicas: 1
Container
v2
Pod


Deployment
Container
v2
Pod
replicas: 2
ReplicaSet
Container
v1
Pod
replicas: 0
ReplicaSet
Container
v2
Pod
replicas: 2
Container
v1
Pod
Container
v1
Pod
Container
v2
Pod
Container
v2
Pod

Demo:
Deployment

Service

Expose Pod’s as a network service
Simple DNS names
Uses an underlying kube-proxy*


Service
Port 80
(http)
Deployment
Container
v1
Pod
replicas: 2
ReplicaSet
Container
v1
Pod
replicas: 2
Container
v1
Pod
Container
v1
Pod

Service Publishing

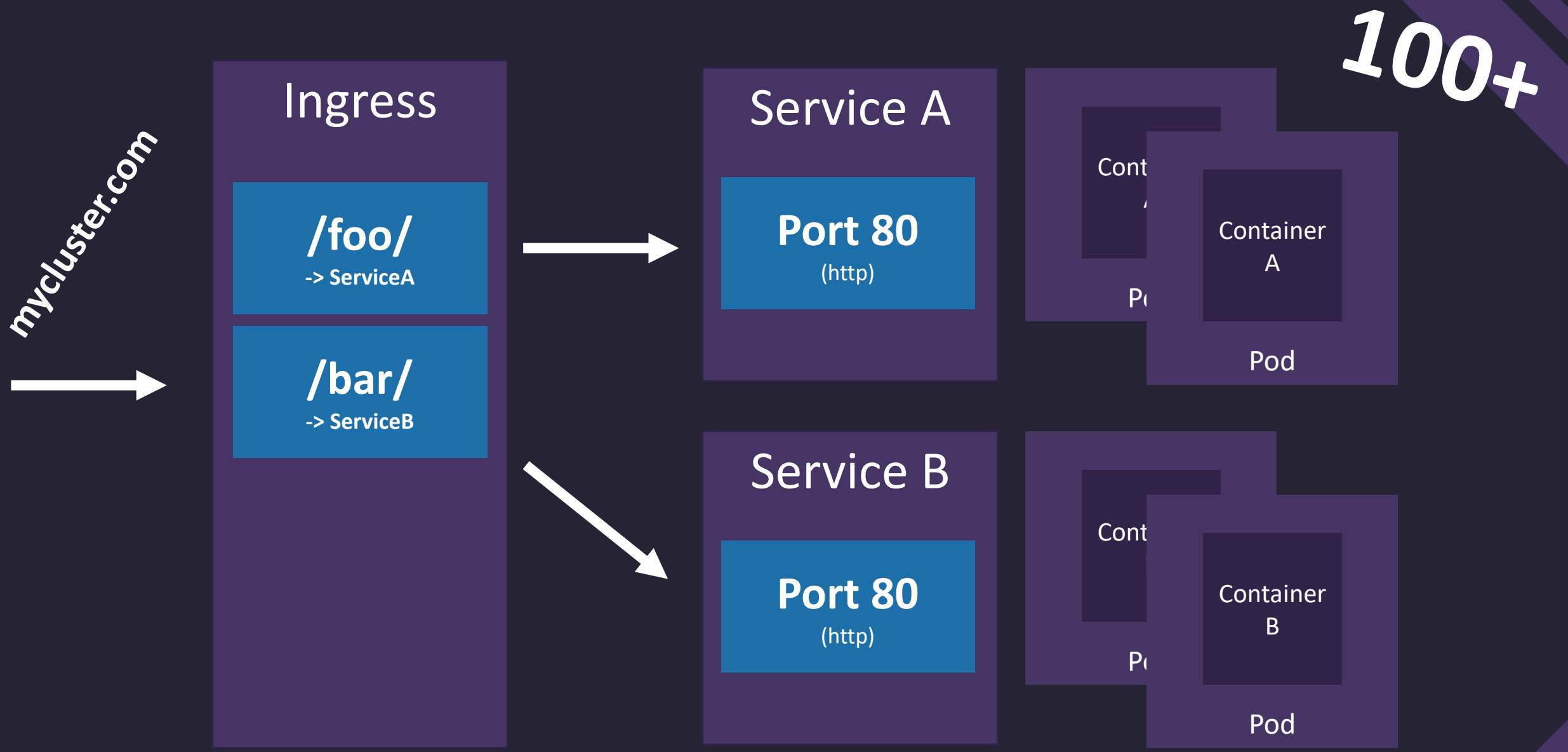
ClusterIP (Default) – Internal only
NodePort – Expose a port on every node
LoadBalancer – Use an external LB

Demo:
Service & Publishing

100+

Ingress

Exposes HTTP & HTTPS routes
Reverse Proxy
Single IP, multiple services
Requires an Ingress Controller


mycluster.com
Ingress
/foo/
-> ServiceA
/bar/
-> ServiceB
Service A
Port 80
(http)
Service B
Port 80
(http)
Container
A
Pod
Container
B
Pod
100+

Demo:
Ingress via Nginx

Kubernetes
Extras & More

100+

Autoscaling

Scale Pods via Horizontal Pod Autoscaler

Control Replicas via Metrics

Scaling Nodes is provider specific

Volumes

100+

Directory on a Disk or another Container

Lots of backing providers!

PersistentVolume’s give more control around lifecycle.

Namespaces

100+

Logical separation of a cluster
Namespace per team
Some objects (nodes) aren’t namespace’d

100+

Affinity and anti-affinity

Greater control over node mapping
Via *nodeSelector*

Even inter-pod affinity!

100+

Taints & Tolerations

Great for multiple worker node pools
Allows for ‘soft’ or ‘hard’ requirement

Dedicated nodes or specialised hardware

100+

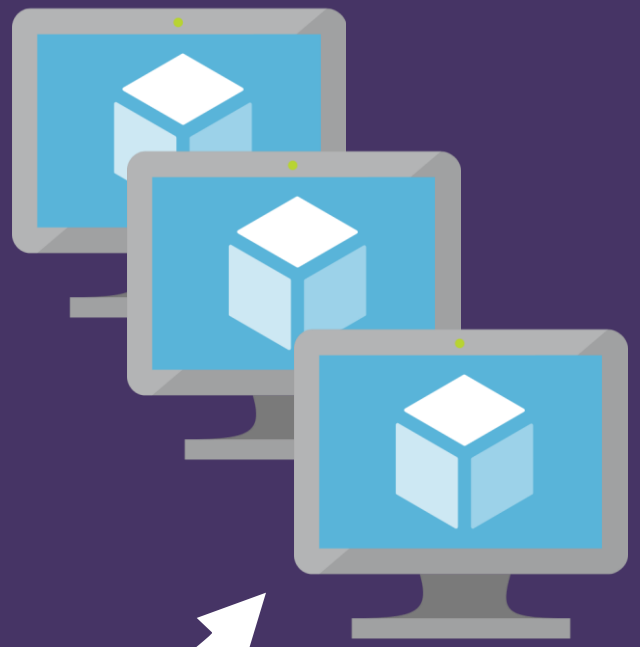
Resource Quotas

Memory, CPU, Storage
Limits & Requests

Request 1gb, Limit 2gb

Azure Kubernetes Service

Master Nodes



Worker Nodes

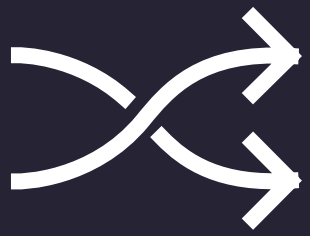
Control Plane

Master Nodes

Control Plane

Worker Nodes

Built-in auto scaling



Elastically burst using ACI

Global data centers

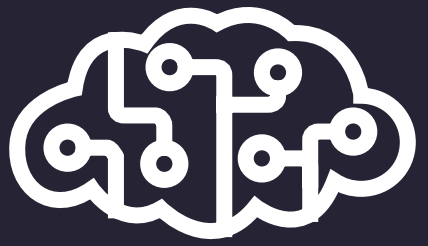
Geo-replicated container registry

Control access through AAD and RBAC

Safeguard keys and secrets with Key Vault

Advanced networking via Azure CNI

SOC, HIPAA, PCI & more



Tips & Tricks

Keep it simple!

Start simple

Don’t be too verbose

Easy to read = easy to review

Group where appropriate

Deployment & Service?

Single file!

100+ deployments, services, and more?

Multiple files

Single Pod or ReplicaSet?

No one likes a naked pod…
ReplicaSet with ‘replicas: 1’
Even better, go for a deployment!

Avoid host bound networking

HostNetwork, HostIP, HostPort

Prefer Services and/or Ingress

Use labels!

Name is a great start
Version is helpful
Part-Of for grouping
Component for identification

Container image tags

Don’t use :latest

Be specific with versions

Image pull behaviour is impacted by this

Right size node

Tiny nodes are great for dev…

But not so great for prod!

Pod scaling requires space (deployment!)

You can horizontally scale nodes!

Kubectl

Learn the CLI

kubectl --help is your friend

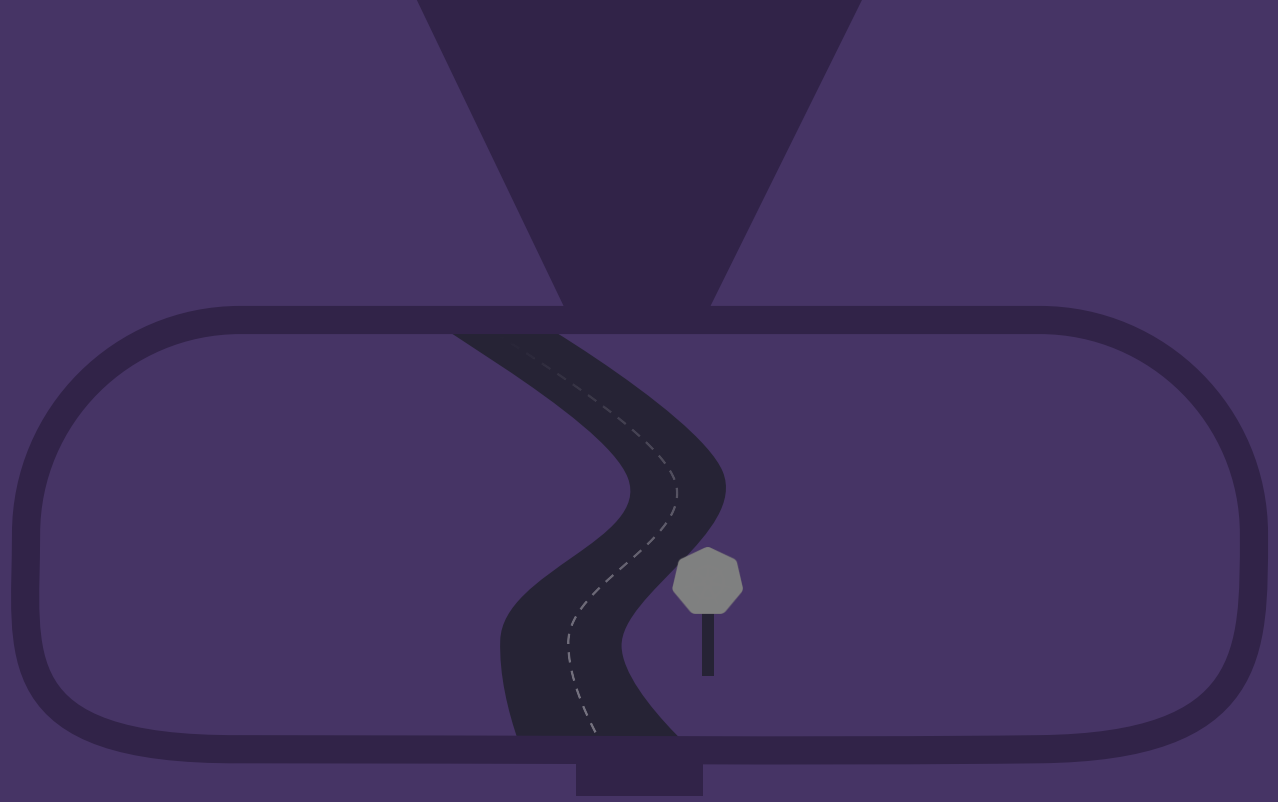
Apply everything in a folder

kubectl apply -f

Start stateless

External state is simple

Volumes, while incredibly powerful, require planning



Thank You!

Questions?