| Concepts | Technologies | Application |
|---|---|---|
| Stateless & Stateful | Kubernetes | |
| Virtualization v/s Containers | | |
| Orchestration | | |
| DevOps | | |
| CI/CD | | |

Docker

Dockerfile

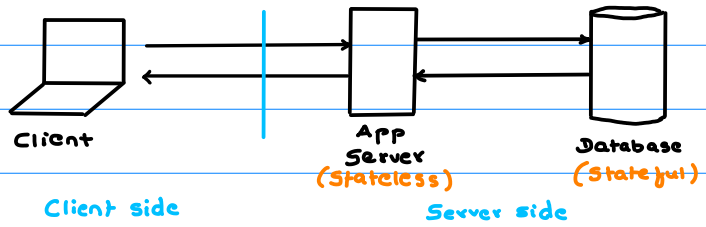
Docker registry

Multi-host

Cross-host networking

State/Volume

1. Image building
2. Container ↔ Container networking
3. Volume management

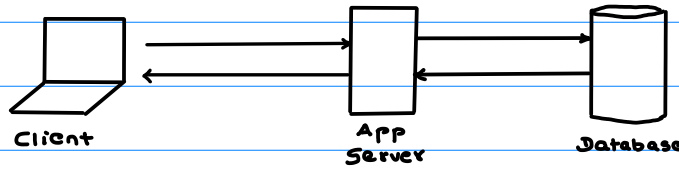



Anytime, anywhere, any device

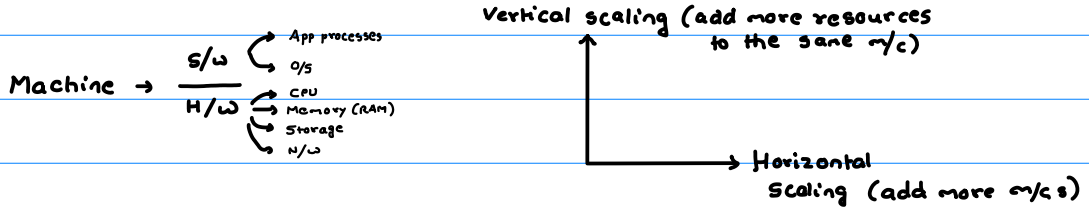
State, Stateless, Stateful

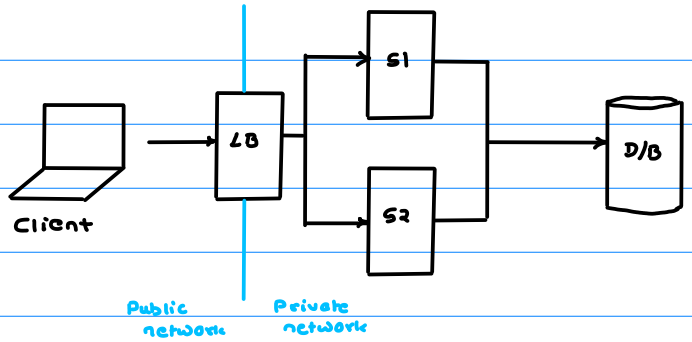

| | Scalability | Failure Handling |
|---|---|---|
| Stateless | LB + Replicas | Replace replica |
| Stateful | Sharding | Replication |

## Stateless Scalability




100 Req/s



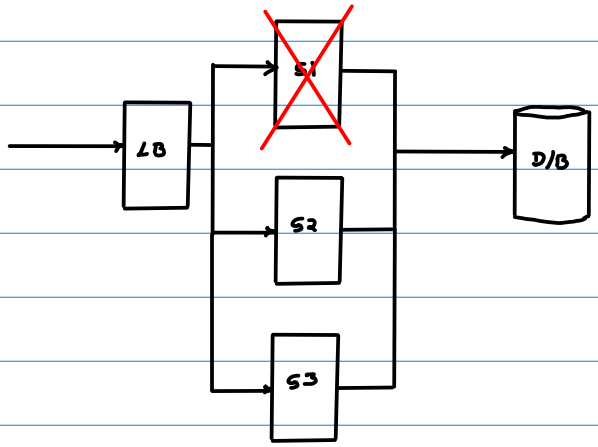




## Load balancer

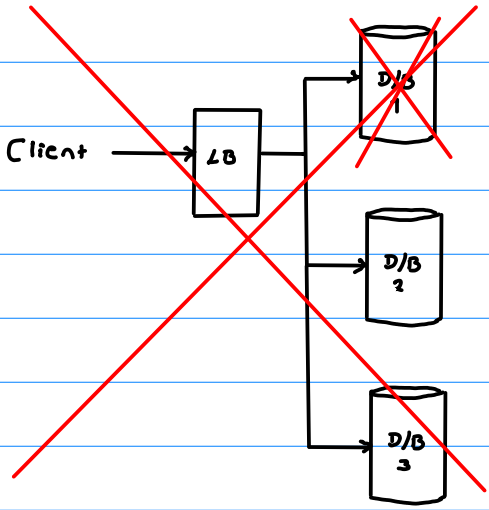
1. Ingress
2. Load balancing / Round Robin / Reverse Proxy
3. Failure management

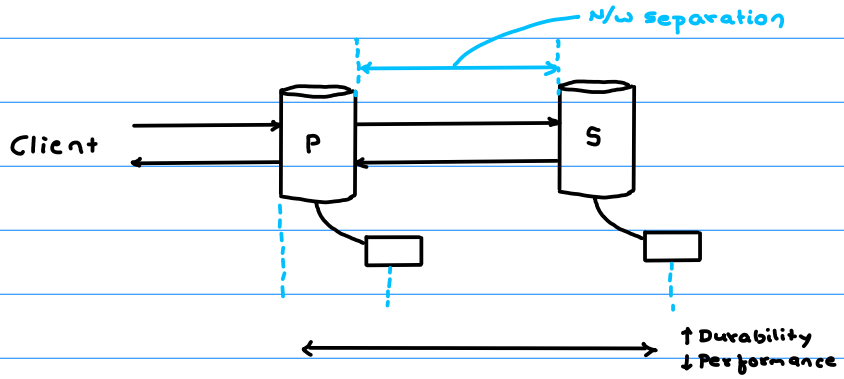
Health check endpoint

/ping → 200 OK



## Stateful scalability

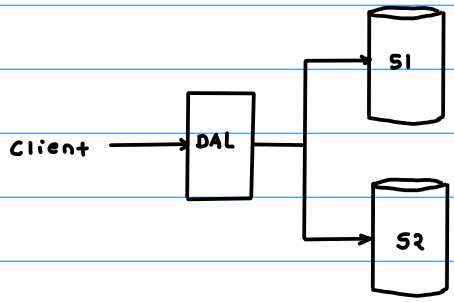



CRUD
- Read
- Write

### Types of failure

1. M/c failure → S/w, H/w → Multiple m/c
2. Power, Fire → Multiple A/z
3. Cyclone, Earthquake → Multiple region



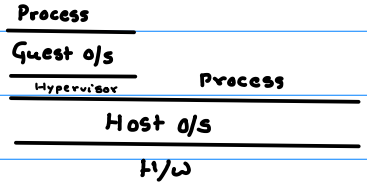

Shard key → Consistent Hashing → 0/1

## Virtualization

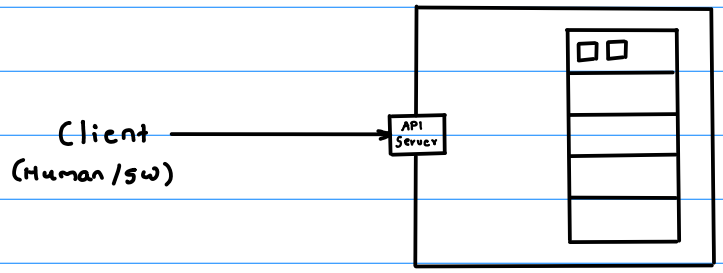
Virtual Machine → S/W (Process, O/S) / H/W (CPU, RAM, Storage, N/W)

↳ Not real

Real Machine




Isolation of resources




AWS → 2006

Google Cloud → 2008

ELB, RDS → 2009

Cloudformation → 2010

Azure → 2010

IAAS

PAAS

IAC

Reduced admin

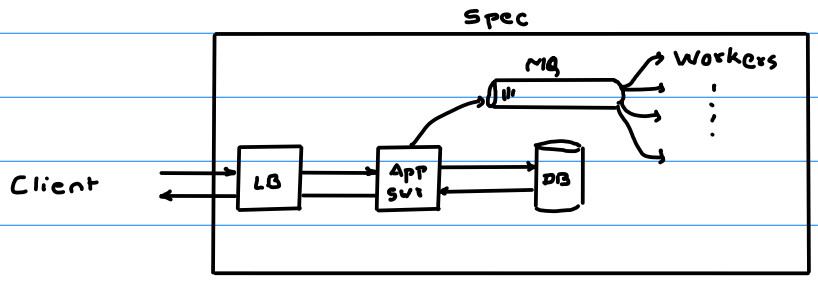
Increased automation

Public

Private

Hybrid

Multi-cloud




App Servers

Databases

Load balancers

Caching Servers

Message queues

(IAC) Spec → Orchestrator → Create / Operate

Vendor lock-in

Kubernetes → FOSS

↳ Container based orchestration

↳ Google Kubernetes Engine

Amazon Kubernetes Service

Azure —"— —"—

Docker

RedHat / IBM

VMware / Pivotal

## Containers

Process

O/S

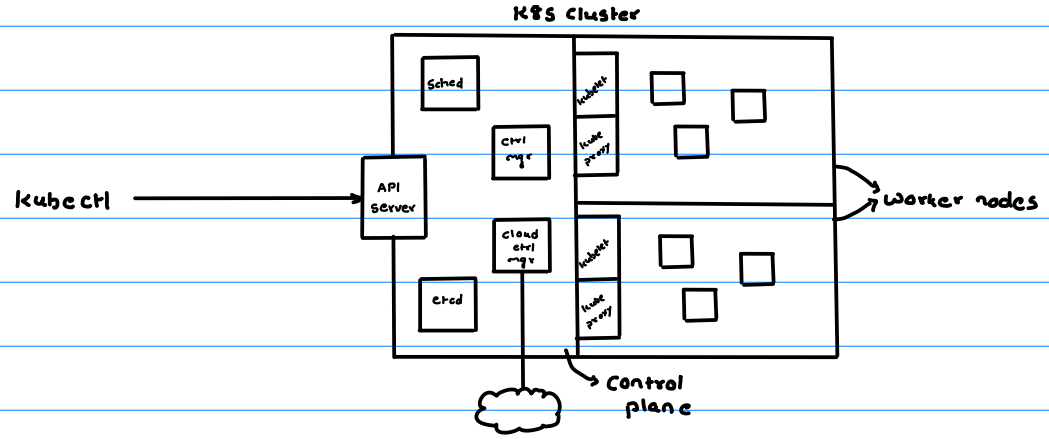
H/W

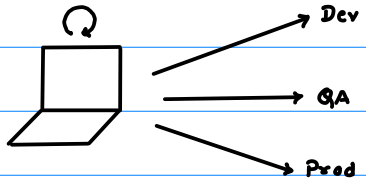
cgroups
namespaces
chroot

## Container v/s VM

1. Resource isolation
2. VMs need more resources
   Containers are light-weight
3. VMs are more secure
   Containers are not as secure

## Kubernetes






kubectl get nodes

KUBECONFIG / Context

1. Cluster (IP:Port)
2. Auth info
3. Namespace

Resources

| | | |
|---|---|---|
| Pods | Persistent Volumes | Configmaps |
| Replicasets | Persistent Volume Claims | Secrets |
| Deployment | Statefulsets | Namespaces |
| Services | | |

Declarative v/s Imperative Spec

deploy.sh

```
#!/bin/bash
kubectl run ----
kubectl run ----
---
---
```

$ ./deploy.sh

* What to do
* When to do
* When not to do

Declarative

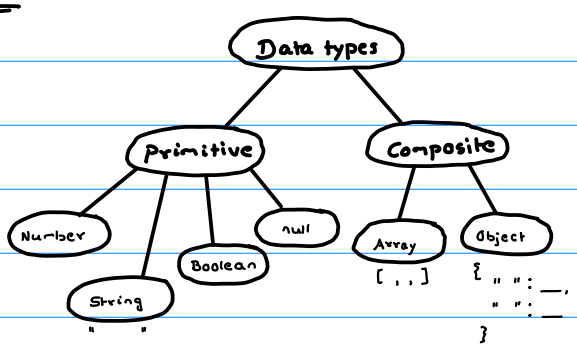
What we need → Orchestrator → Env

(Desired state) (Current state)

```
Pods:
1. nginx → image=nginx
2. mysql → - - -
- --
- --
```

YAML

## YAML




Array of arrays → [ [ ], [ ] ]

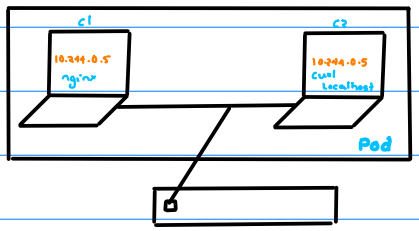
Array of objects → [ { }, { } ]

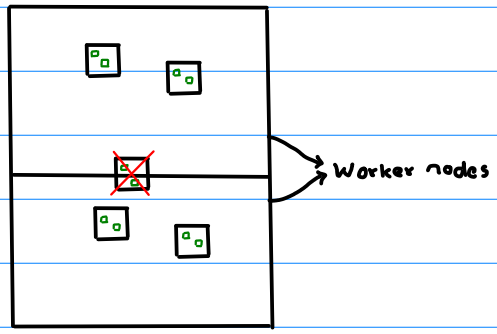
Object containing arrays → { " " : [ ] }

Object containing objects → { " " : { } }

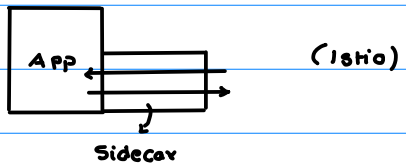
## Pods & containers

1. All containers of a pod have the same IP
2. All containers of a pod have to be healthy
3. All containers of a pod will be in the same node

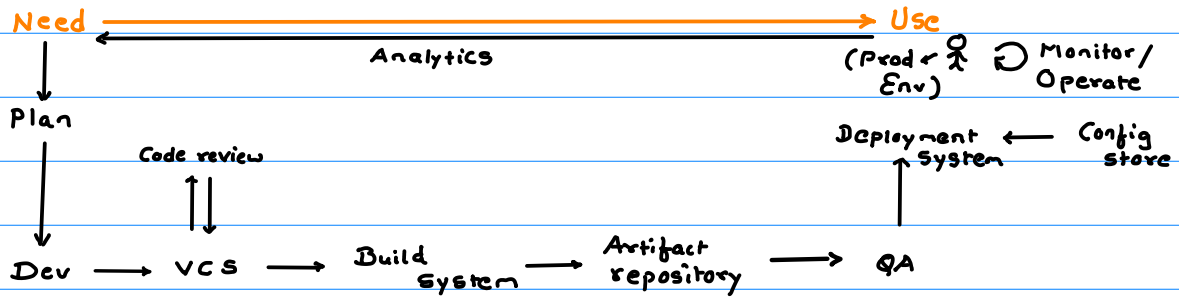



## Sidecar containers



DevOps



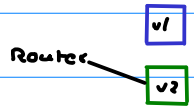

Provisioning

Deployment

v1 v1 v1 → v2 v2 v2

1. Bring down v1
   Start all v2 (Recreate)

2. Start all v2
   Switch traffic
   Bring down v1




3. Incremental deployment
   (Rolling update)

Current → Desired

v1 v1 v1 → $T_0$ → v2 v2 v2

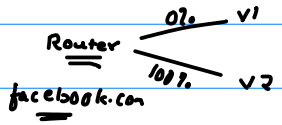
~~v1~~ v1 v1 v2 → $T_1$

~~v1~~ v1 v2 v2 → $T_2$

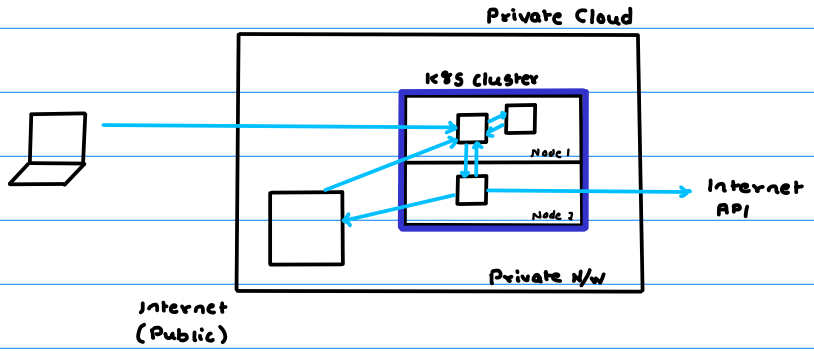
~~v1~~ v2 v2 v2 → $T_3$

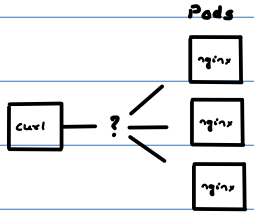
v2 v2 v2 → $T_4$

4. Canary deployment



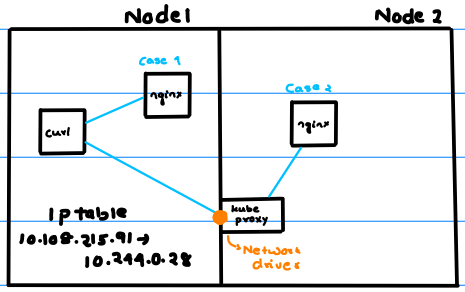
# Services



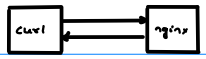

Service name → Cluster IP → Pod IP

nginx → 10.108.215.91 → 10.244.0.28, 10.244.0.29, 10.244.0.30



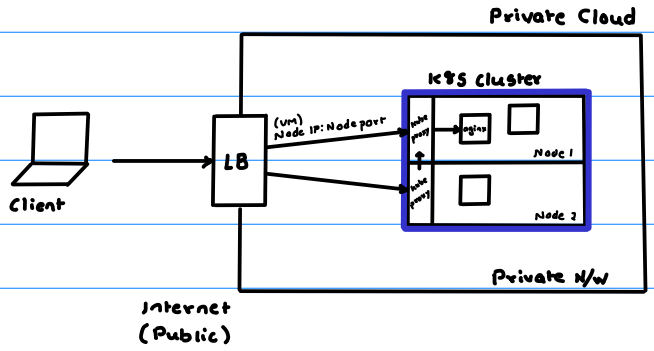

[Pod IP ⟷ Pod IP] → K8S N/W

VM ⟷ VM → VM N/W




Curl nginx

1. Service
2. Kube DNS  nginx → 10.108.215.91
3. Network driver  Node ⟷ Node
4. kube-proxy
5. endpoint-controller




Client → LB → Node's kube-proxy → Pod (Ingress-controller) → Service Pod

/users → users:80

/payments → payments:80

↳ Cluster IP : Port

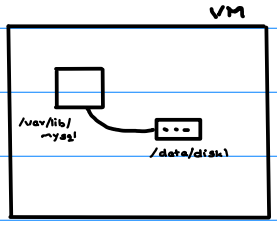
## Volume/Storage Management

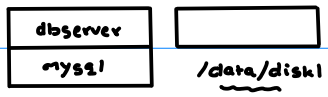
MySQL → DDL & DML

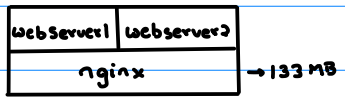
↓

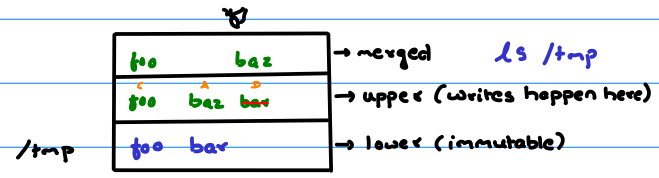
Data directory

(/var/lib/mysql)



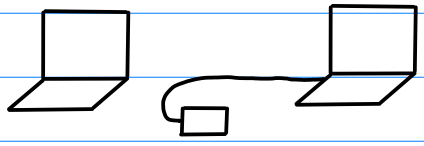



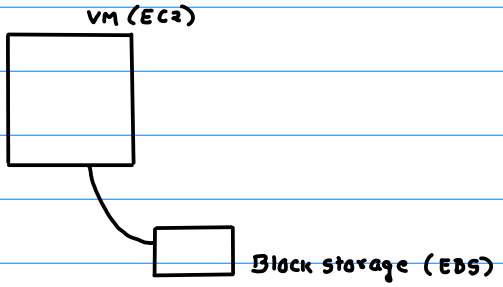

docker run -v /data/disk1 : /var/lib/mysql

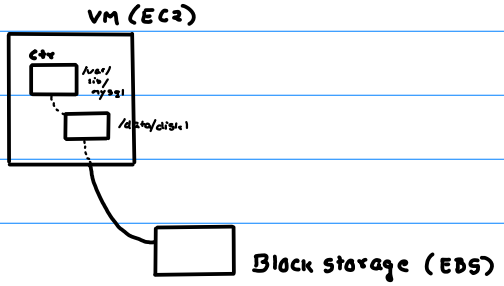
Host path — Container path

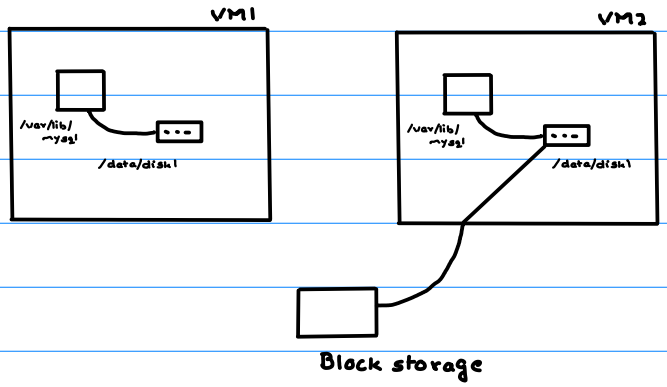
write → Read

Read ← write







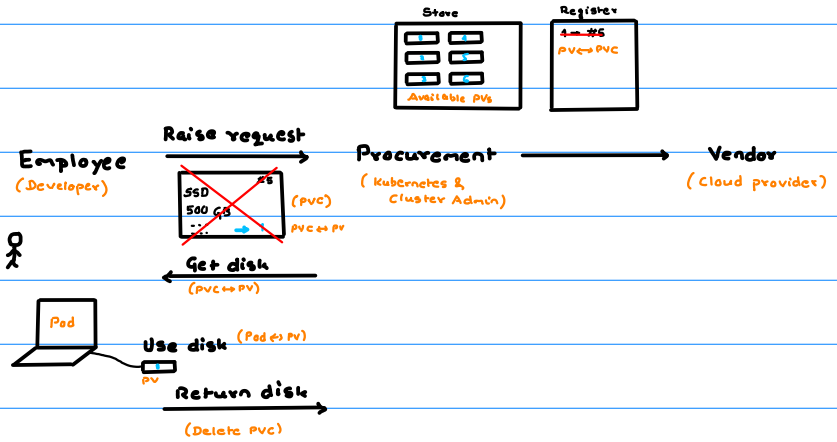



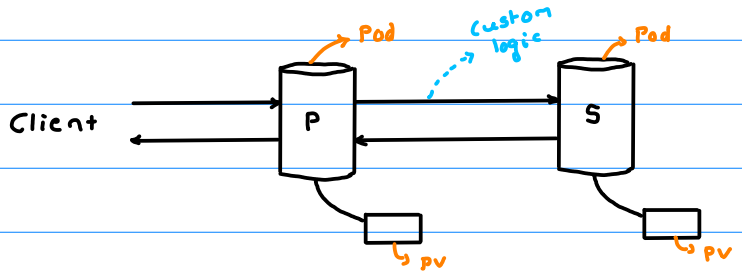

Cloud APIs

O/S

1. Create Volume
2. Attach volume → /dev/sdf
3. Partition volume → fdisk → /dev/sdf1
   /dev/sdf2
4. Format partition → mkfs.*
5. Mount partition → mount /dev/sdf1 /mnt/disk1
6. Read/write file/directories → cat, ls, rm, mkdir ...
7. Umount → umount
8. Detach volume

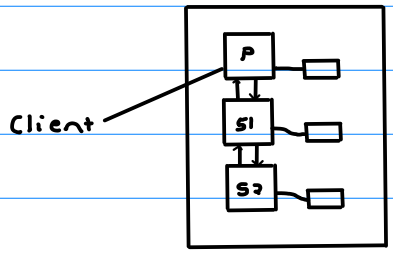
## Persistent Volume (PV) & Persistent Volume Claim (PVC)




| Primary | Secondary | Secondary |
|---|---|---|
| Pod | Pod | Pod |
| PVC | PVC | PVC |
| PV | PV | PV |
| Service | Service | Service |

## Service discovery in stateful components




Client needs to know the specific pod

Pods need to communicate with each other

**ReplicaSet**

replicas: 3

template:
pod spec

↓

Pod names were not predictable

**StatefulSet**

replicas: 3

template:
pod spec

vctemplate:
pvc spec

1. Pod names are predictable
2. Pods start in a certain order
3. Pods & corresponding PVCs are auto-managed
4. Every pod can be addressed

Resources + Verbs → Rules ↔ Role ↔ Users

pods - read

RoleBinding

ClusterRole (non-namespaced resources)

Users → ClusterRoleBinding