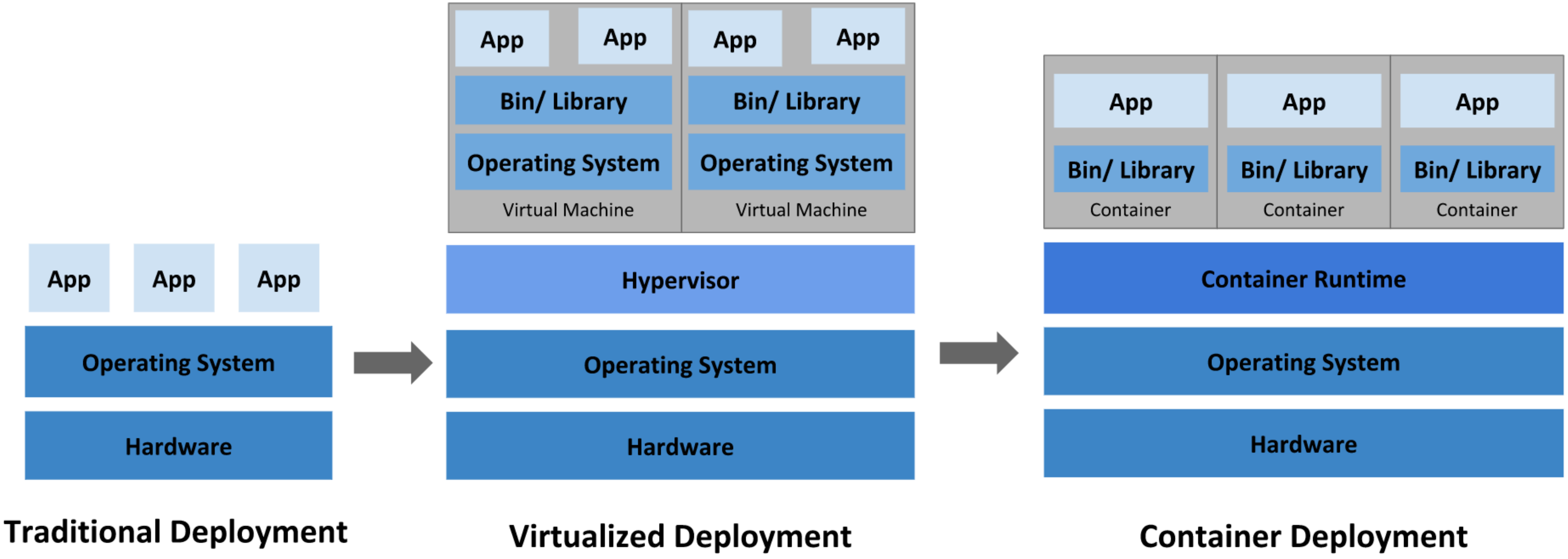
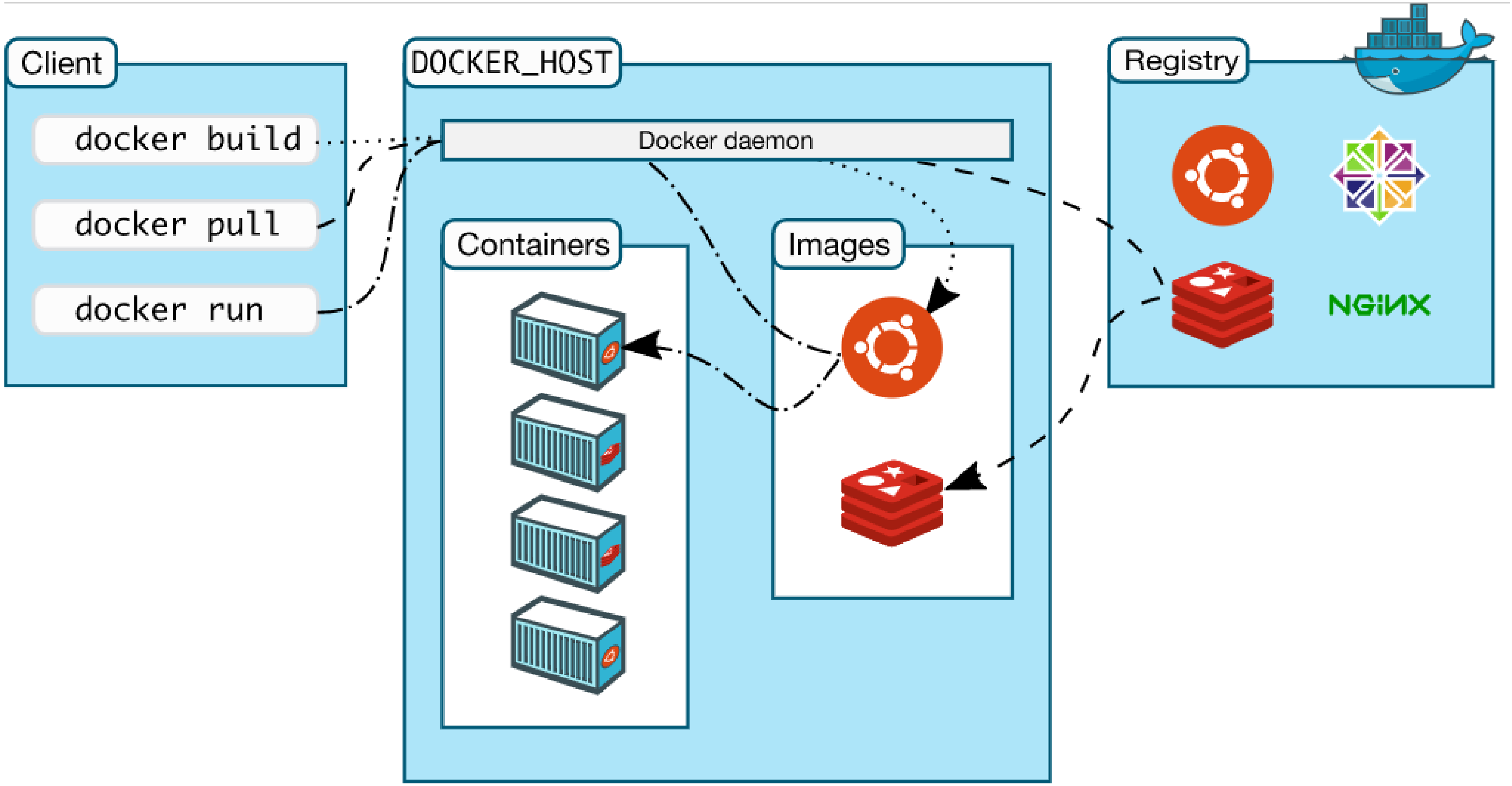
Introduction to Kubernetes

Part 2

Recap


App
App
App
Operating System
Hardware
Traditional Deployment
App
App
App
App
Bin/ Library
Bin/ Library
Operating System
Operating System
Virtual Machine
Virtual Machine
Hypervisor
Operating System
Hardware
Virtualized Deployment
App
App
App
Bin/ Library
Bin/ Library
Bin/ Library
Container
Container
Container
Container Runtime
Operating System
Hardware
Container Deployment

Recap

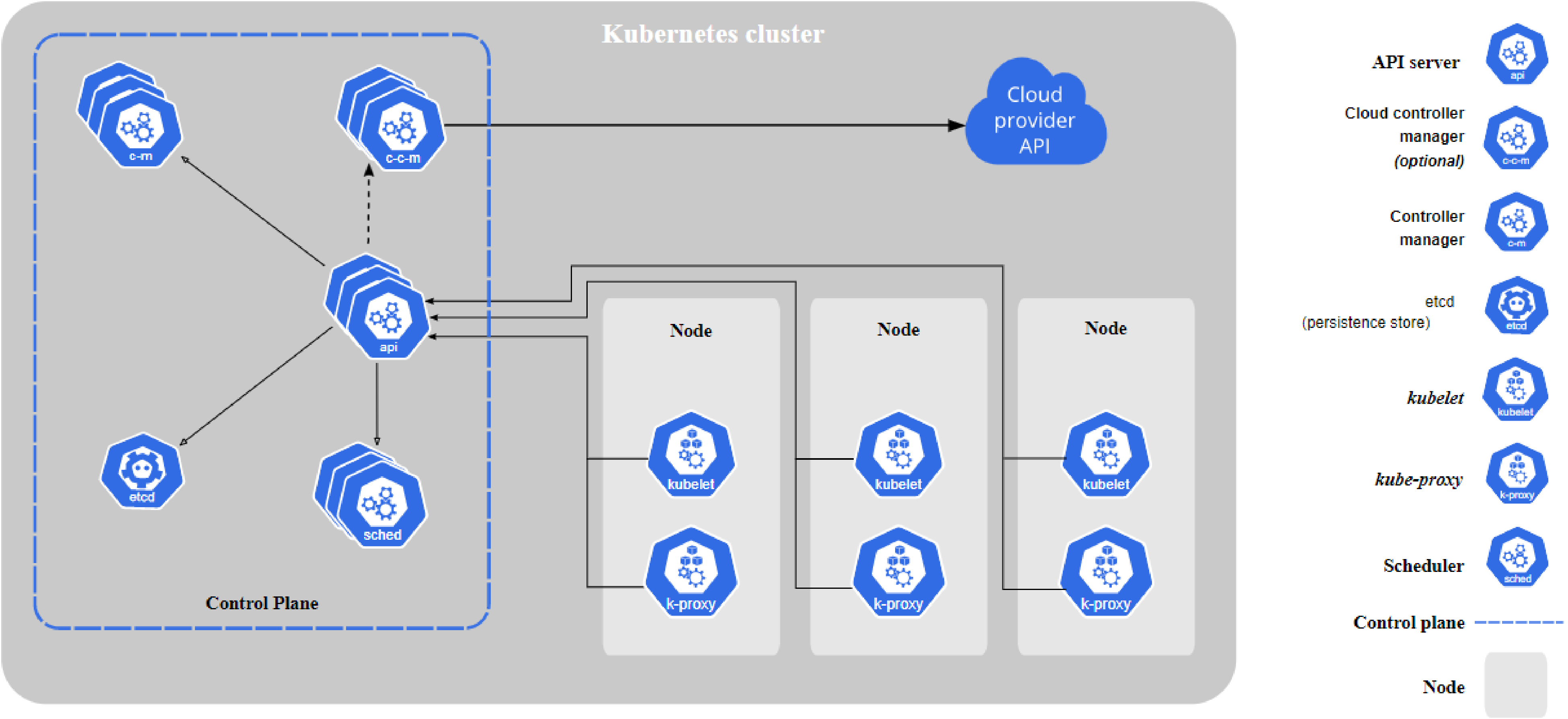

Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

Recap

Kubernetes

- Distributed system to orchestrate containers
- Declarative and based on control loops
- Core object is Pod, which wraps 1..N containers
- Namespace – virtual cluster inside a physical one
- Control and Data planes

Recap



Kubernetes cluster
c-m
c-c-m
Cloud provider API
api
etcd
sched
Control Plane
Node
kubelet
k-proxy
Node
kubelet
k-proxy
Node
kubelet
k-proxy
API server
api
Cloud controller manager *(optional)*
c-c-m
Controller manager
c-m
etcd (persistence store)
etcd
kubelet
kubelet
kube-proxy
k-proxy
Scheduler
sched
Control plane
Node

Course Plan

1. What is Kubernetes
2. What is Container
3. Kubernetes 10000-foot view
4. Kubernetes core objects
5. Daily interactions

Kubernetes Core Objects

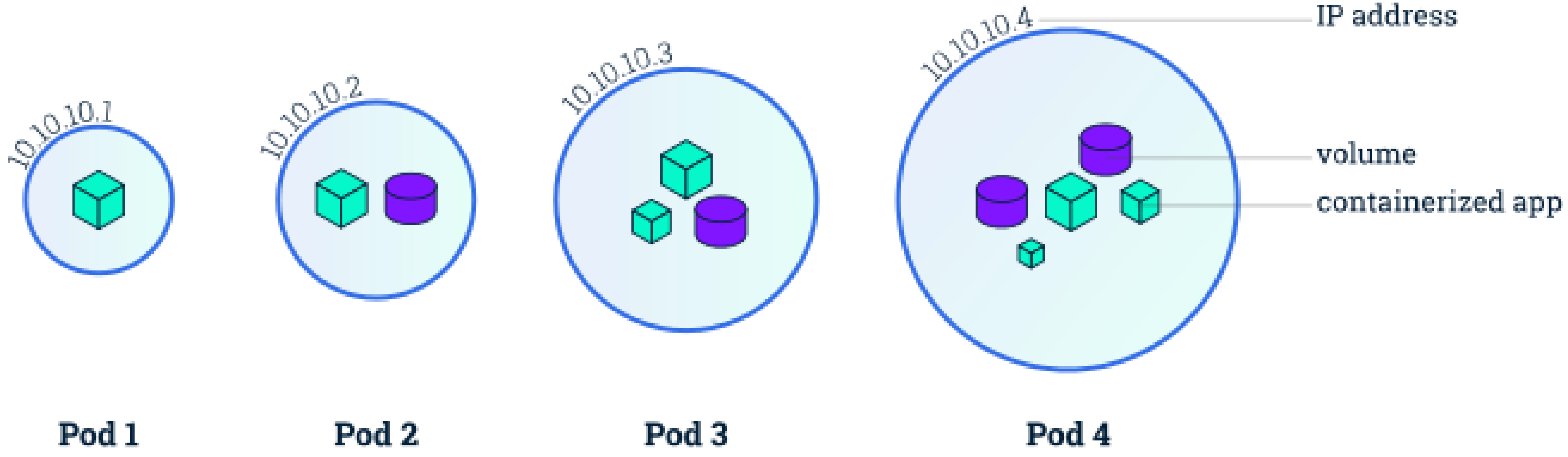
- Pod
- Deployment
- StatefulSet
- Storage
- DaemonSet
- Job and CronJob
- Services and Ingress
- ConfigMaps and Secrets
- Policies

Kubernetes Core Objects

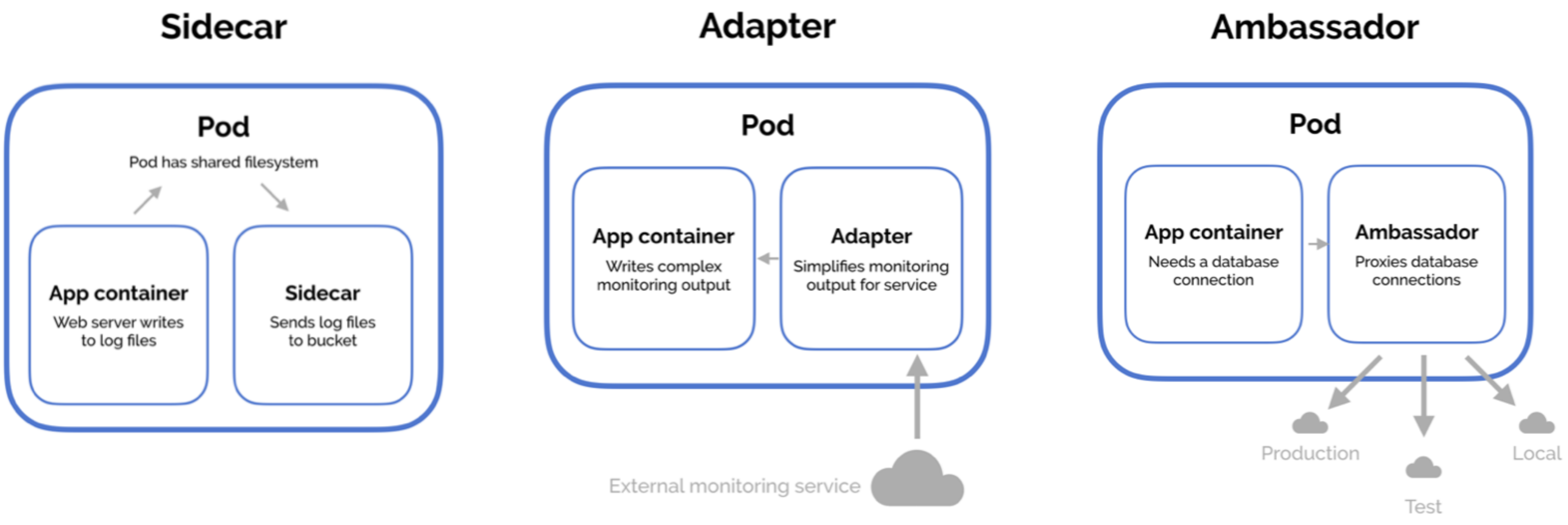
Pod:

- Smallest deployable unit
- Can have >=1 containers
- Containers could be init-only
- All containers in a pod shares the same storage and network
- has unique IP

Kubernetes Core Objects



Kubernetes Core Objects



Sidecar

Pod

Pod has shared filesystem

App container

Web server writes to log files

Sidecar

Sends log files to bucket

Adapter

Pod

App container

Writes complex monitoring output

Adapter

Simplifies monitoring output for service

External monitoring service

Ambassador

Pod

App container

Needs a database connection

Ambassador

Proxies database connections

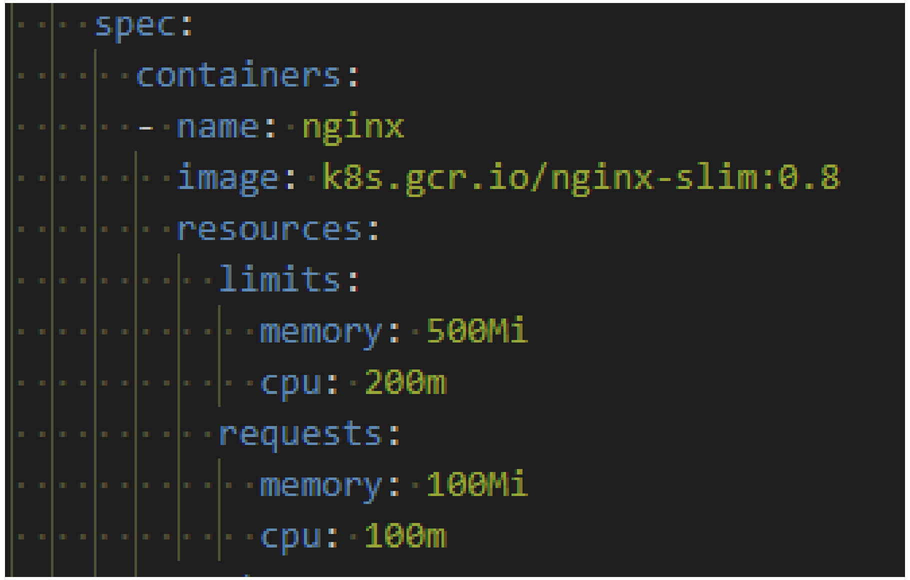
Production

Test

Local

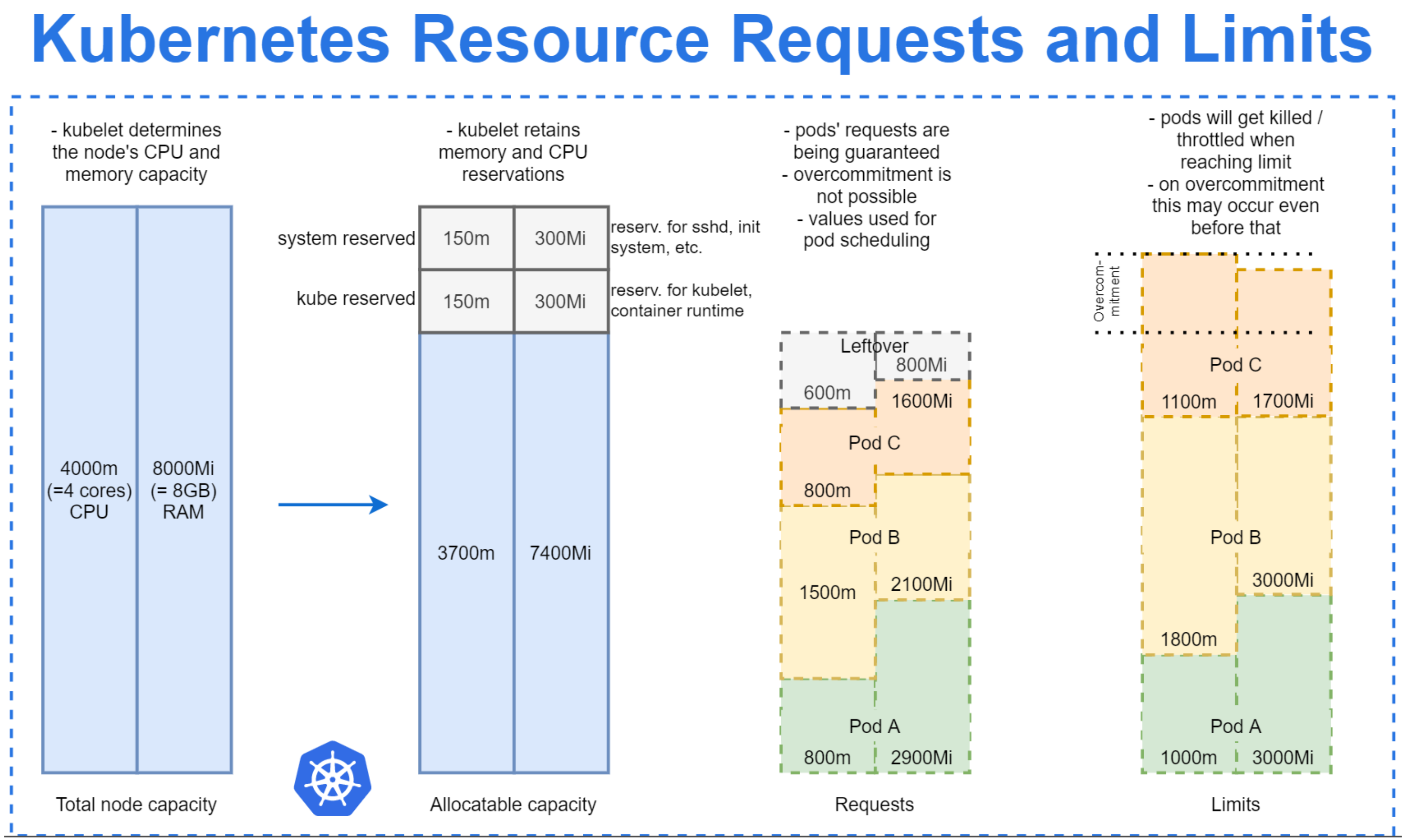
Kubernetes Core Objects

Resource-quotas: requests & limits



```
    spec:
      containers:
        - name: nginx
          image: k8s.gcr.io/nginx-slim:0.8
          resources:
            limits:
              memory: 500Mi
              cpu: 200m
            requests:
              memory: 100Mi
              cpu: 100m
```

Kubernetes Core Objects


Kubernetes Resource Requests and Limits
- kubelet determines the node's CPU and memory capacity
4000m (=4 cores) CPU
8000Mi (= 8GB) RAM
Total node capacity
- kubelet retains memory and CPU reservations
system reserved
150m
300Mi
reserv. for sshd, init system, etc.
kube reserved
150m
300Mi
reserv. for kubelet, container runtime
3700m
7400Mi
Allocatable capacity
- pods' requests are being guaranteed
- overcommitment is not possible
- values used for pod scheduling
Leftover
600m
800Mi
Pod C
800m
1600Mi
Pod B
1500m
2100Mi
Pod A
800m
2900Mi
Requests
- pods will get killed / throttled when reaching limit
- on overcommitment this may occur even before that
Overcom-mitment
Pod C
1100m
1700Mi
Pod B
1800m
3000Mi
Pod A
1000m
3000Mi
Limits

Kubernetes Core Objects

To pull image images from Container Registry Kubernetes uses

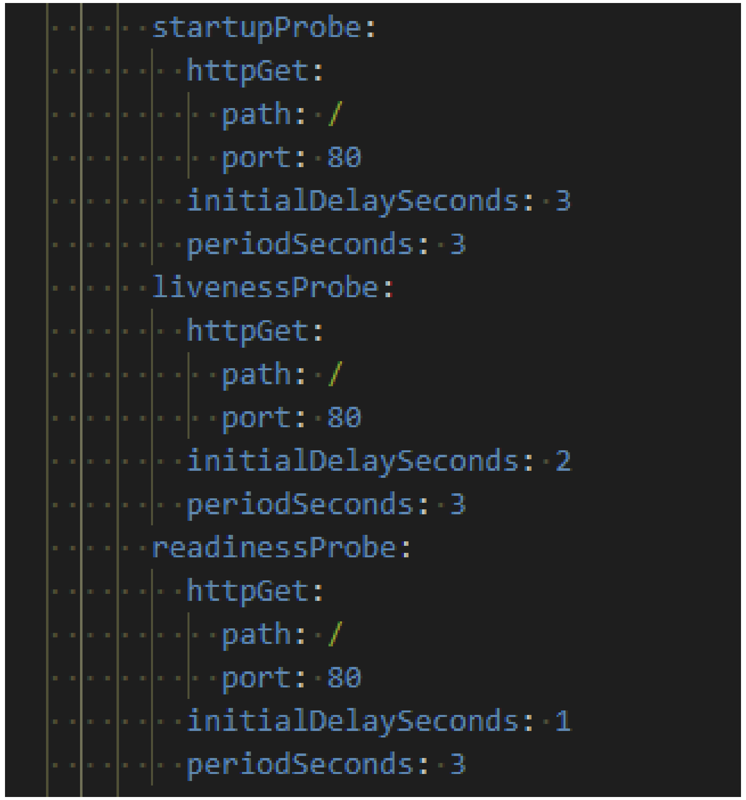
- Node configuration
- or imagePullSecret property

Kubernetes Core Objects

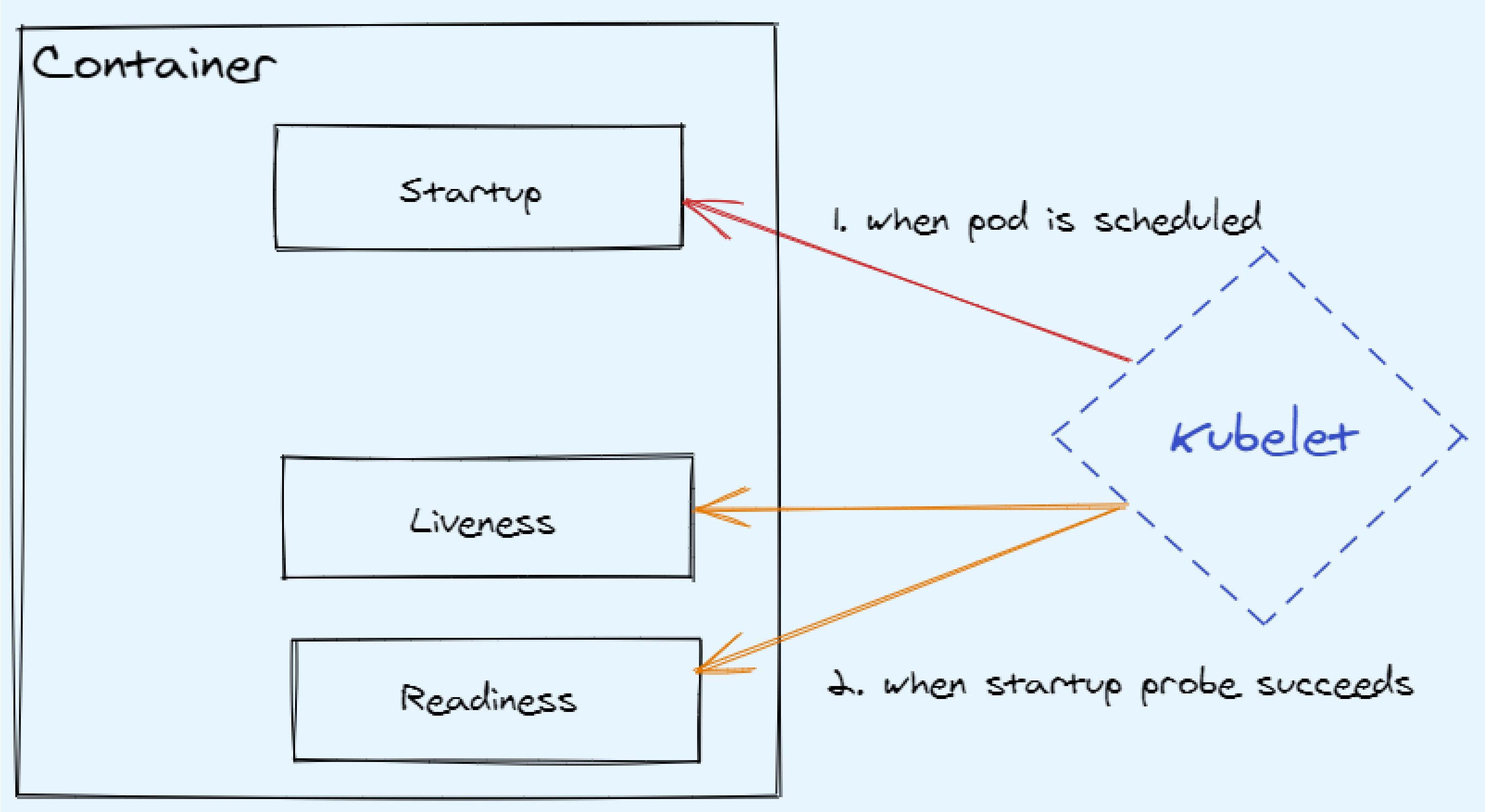
Pod Lifecycle:

- Get scheduled
- Remains on the node until completion, termination or deletion
- Do not self-heal (kubelet *heals*)
- Probes: startup, readiness, liveness

Kubernetes Core Objects



```
      startupProbe:
        httpGet:
          path: /
          port: 80
        initialDelaySeconds: 3
        periodSeconds: 3
      livenessProbe:
        httpGet:
          path: /
          port: 80
        initialDelaySeconds: 2
        periodSeconds: 3
      readinessProbe:
        httpGet:
          path: /
          port: 80
        initialDelaySeconds: 1
        periodSeconds: 3
```



Kubernetes Core Objects

Pod Scheduling:

- Available node resources (requests)
- Node selectors
- Node taints/toleration
- Affinity and anti-affinity

Kubernetes Core Objects

Deployment:

- a set of containers
- defines update-strategy
- scales container instances
- tools to watch the status and roll-back
- wraps ReplicaSets

Kubernetes Core Objects

ReplicaSet sets how many pods of exact specification Kubernetes should run (lower-level object)

Kubernetes Core Objects

Deployment Demo

Kubernetes Core Objects

StatefulSet: like Deployment, but with guarantees about ordering and naming (sticky identity)

Kubernetes Core Objects

Storage:

- Volumes - piece of storage available on the node, where pod is running. Volume is mounted into Pods
- Persistent Volume (PV) - a piece of storage in the cluster with reference to physical data location
- Persistent Volume Claim (PVC) - a storage request by a user. Often used to map StatefulSets to PVs
- StorageClass defines parameters of PV: provisioner (AzureDisk, AWSElasticBlockStore, etc), reclaim policy, allows resizing, etc

Kubernetes Core Objects

StatefulSet Demo

Kubernetes Core Objects

DaemonSet defines which Nodes should have an instance of given Pod

Kubernetes Core Objects

Jobs/CronJob run scheduled or one-time tasks

Kubernetes Core Objects

Service:

- not an application, but a record in etcd
- represents networking rules

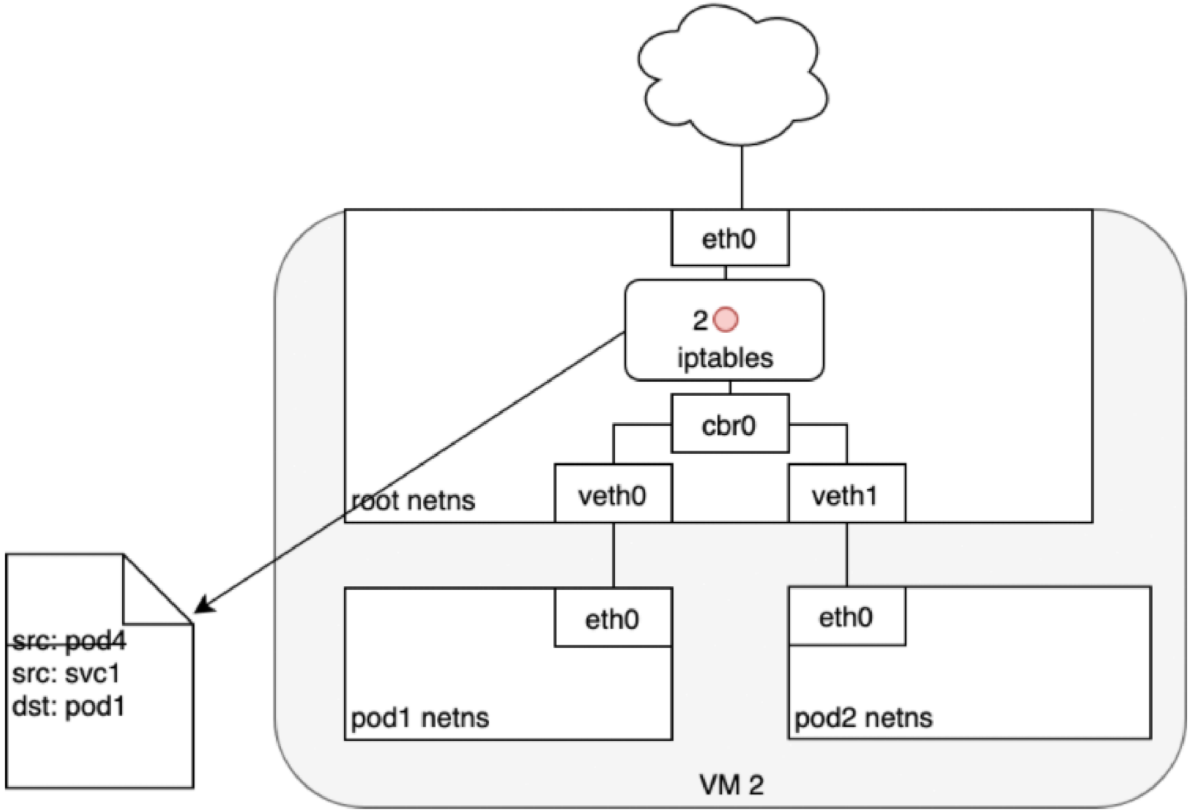
Kubernetes Core Objects

Service:

- pods are disposable, while a service is long-living
- service selects pods based on labels
- service might "route" traffic to pods
- headless-service - does not route traffic, but registers DNS identity

Kubernetes Core Objects

5.4 Life of a packet: Service to Pod



Kubernetes Core Objects

Service types:

- ClusterIP
- NodePort
- LoadBalancer

Kubernetes Core Objects

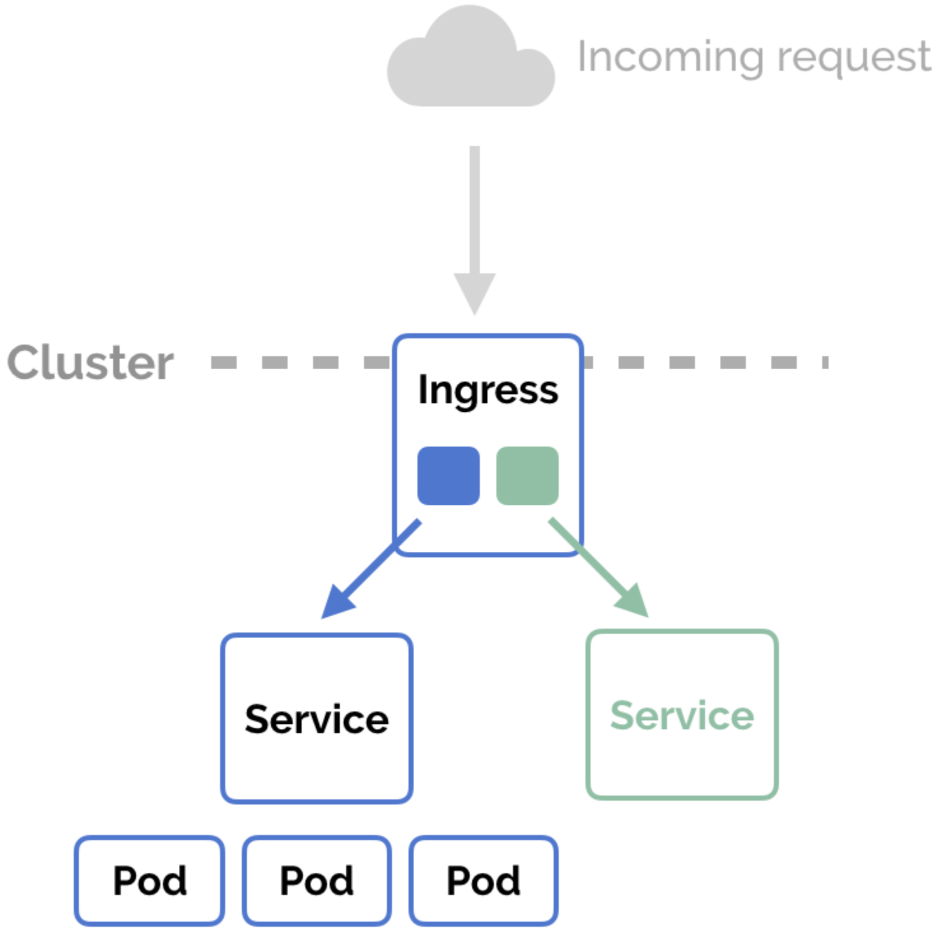
Services Demo

Kubernetes Core Objects

Ingress:

- API object (not a real application) to define rules
- Exposes HTTP/HTTPS routes from outside the cluster to Services
- Real work is done by ingress-controller (nginx, haproxy, envoy, etc)

Kubernetes Core Objects



Kubernetes Core Objects

Configuration:

- Decouple environment-specific information from application
- via environment variables
- via ConfigMap (CM) - API object to store non-confidential data. CM could be mapped as env-vars or volumes
- via Secret - API object to store confidential data
- Secrets are not encrypted and are stored in etcd(!); secrets are encoded in base64, stored in tmpfs (RAM) instead of node-fs, could be encrypted at-rest and in-transit.

Kubernetes Core Objects

Policies:

- LimitRanges: min, max, default values for cpu/memory
- Resource Quotas: how many cpu, memory, storage could be used in a namespace
- Pod Security Policy: enforces pod configuration
- NetworkPolicies: enforces network boundaries (who can talk to whom); implemented by addons
- RBAC: defines identities and their permissions (who can do what)

Kubernetes Core Objects

- Requests and Limits
- 12-factor applications and Sidecar-containers patterns
- Stateful Applications in Kubernetes article
- Kubernetes Deconstructed video
- Understanding Kubernetes Networking article

- Official Pods, Workloads, Networking, Storage, Configuration, Scheduling documentation

Course Plan

1. What is Kubernetes
2. What is Container
3. Kubernetes 10000-foot view
4. Kubernetes core objects
5. Daily interactions

Daily Interactions

Navigate clusters and namespaces:

- `kubectl config get-contexts`
- `kubectl config set-context --current --namespace=kube-system`
- `kubectl config use-context dev-cluster`
- `kubectl get pods -n default`
- `kubectl get pods -n default --context second`

If you have a lot of clusters and namespaces: kubectx + kubens tools

Daily Interactions

Basic operations:

- get help: `kubectl --help` and `kubectl explain …` and `kubectl api-resources`
- apply: `kubectl apply -f sample_deployment.yaml`
- get & watch: `kubectl get pods -o wide -w`
- describe: `kubectl describe deployment nginx-deployment`
- scale: `kubectl scale deployment nginx-deployment --replicas=1`
- delete: `kubectl delete -f sample_deployment.yaml` or `kubectl delete deployment sample`

Daily Interactions

Troubleshooting:

- get or describe an object
- get logs: `*kubectl logs pod_name container_name --tail=100*`
- get events: `*kubectl get events -n default*`
- port-forward to pod/service: `*kubectl port-forward svc/service1 port*`
- run shell in a container: `*kubectl exec pod-name -i -t -- bash*`

If it is too simple, then welcome to complete troubleshooting deployments guide

Daily Interactions

Sharing, versioning, releasing Kubernetes Objects:

- Plain yaml files
- kustomize
- helm
- Argo CD and Flux (GitOps)

Daily Interactions

Additional resources:

- Troubleshooting VMware course
- Kubernetes Best Practices articles and videos by Google
- Tasks section at official docs

Alternative learning materials

- Official documentation is very well-written: https://kubernetes.io/docs/concepts/
- Full list of VMware courses: https://kube.academy/courses
- Azure learn-k8s materials: azure-website + github-repo
- Kubernetes Up and Running book. VMware gives it in exchange for your personal data
- Designing Distributed Systems book. Microsoft gives it in exchange for your personal data
- Learnk8s blog: https://learnk8s.io/blog