DIHSÜD
Digital Innovation Hub

DIGITALISIERUNG FÜR KMU MÖGLICH MACHEN

DER DIGITAL INNOVATION HUB SÜD ALS KOSTENLOSES SERVICE FÜR KMU

Der DIH SÜD
wird unterstützt von:

FFG Forschung wirkt.
Das Land Steiermark
LAND KÄRNTEN

DIHSÜD
Digital Innovation Hub

FH | JOANNEUM
University of Applied Sciences

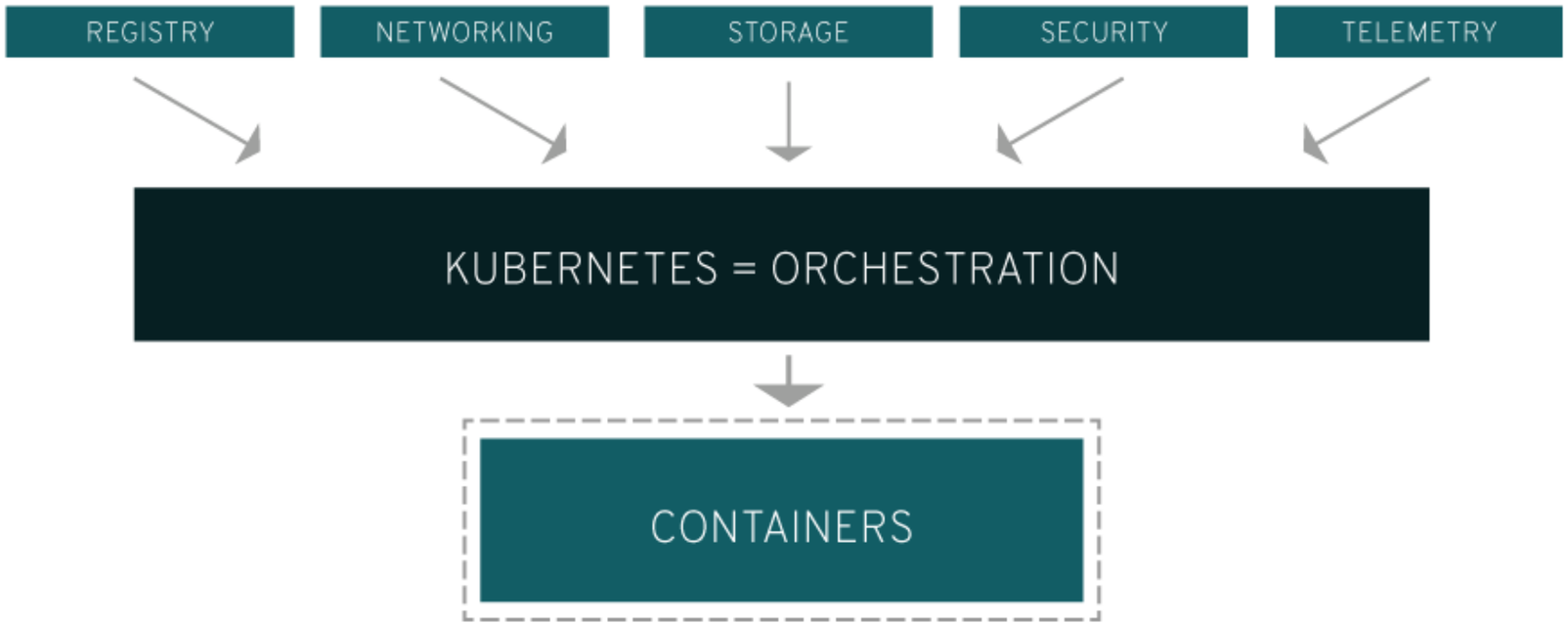
Application Container Orchestration
Kubernetes

Kubernetes (k8s)

Platform for deployment and management of containerized applications and services

- Automatic rollouts and rollbacks
- Secret (e.g. credentials) and configuration management
- Automatic scaling and load balancing
- Automatic restart of failing containers
- Management of node resources (CPU, memory)

Overview



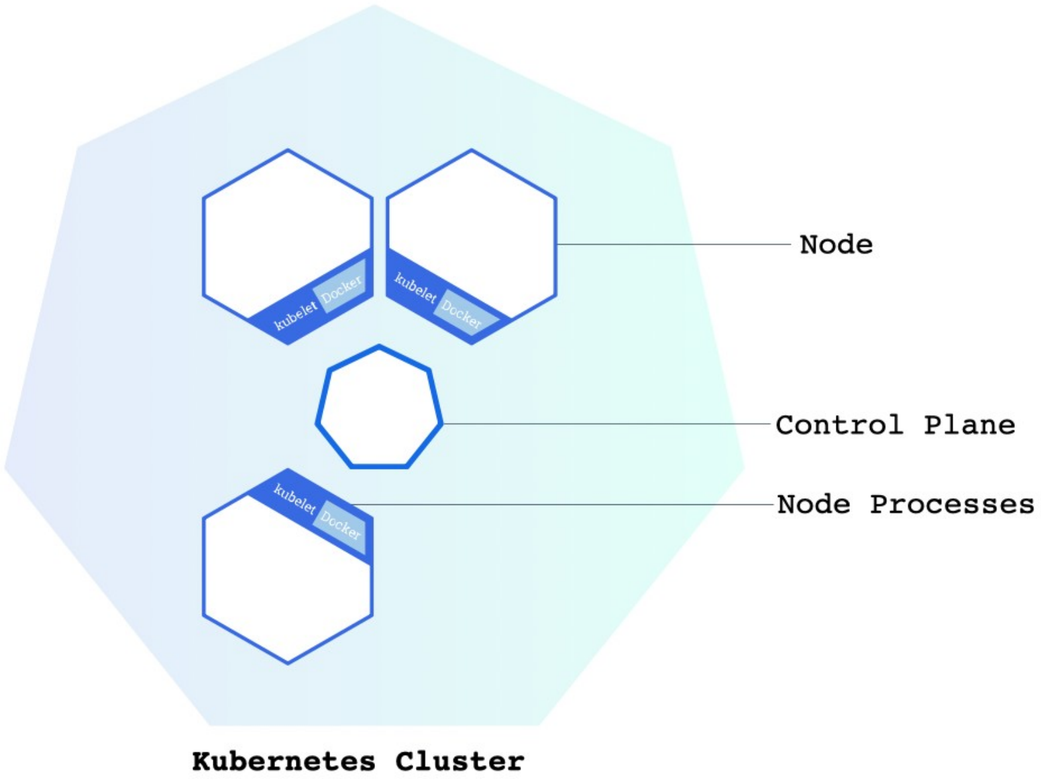
Source: https://www.redhat.com/en/topics/containers/what-is-kubernetes

Kubernetes (k8s)

Software for container orchestration across several hosts

- **Container Runtime Interface (CRI)**
  - Supported by several container runtimes, e.g. containerd, CRI-O
- **Container Network Interface (CNI)**
  - Provides plugin-support for cluster networking
  - E.g. Flannel, Calico, AWS VPC
- **Container Storage Interface (CSI)**
  - Provides plugin-support for exposing of block and filesystem storage to containers
  - E.g. AWS EBS, azureDisk, cephfs, cinder, iSCSI, nfs

Kubernetes Concepts - Cluster



Node runs applications

- Provides the Kubernetes runtime environment
- Maintains running pods

Control Plane coordinates the cluster

- Scheduling and scaling of applications, rolling out updates, ...
- Per default does not run applications

Source: https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/

Control Plane Components

Kube-apiserver

- Exposes the front-end for the Kubernetes control plane (Kubernetes API)

Etcd

- Key value store as backing store for all cluster data

Kube-scheduler

- Watches newly created pods that have no node assigned, and selects a node for them to run on

Kube-controller-manager

- Runs controllers (node, replication, endpoints, service account & token)

Cloud-controller-manager

- Runs controllers that interact with the underlying cloud providers

Node Components

Kubelet

- Agent that ensures that containers are running in a pod

Kube-proxy

- Network proxy that maintains network rules on nodes

Container runtime

- Software that is responsible for running containers
  e.g. containerd, cri-o

Kubernetes Addons

DNS

- DNS server, which serves DNS records for Kubernetes services

Web-UI (Dashboard)

- General purpose, web-based UI for Kubernetes clusters

Container resource monitoring

- Records generic time-series metrics about containers in a central database, and provides a UI for browsing that data

Cluster-level logging

- Responsible for saving container logs to a central log store with search/browsing interface

Kubernetes Objects

Persistent entities, describing …

- what containerized applications are running (and on which nodes)
- what resources are available to those applications
- the policies defining how those applications have to behave

“**record of intent**” describing the desired state of the cluster

- The Control Plane tries to match the cluster reality to defined objects

Required information in **YAML/JSON** format

- `ApiVersion`
- `kind (e.g. pod, deployment, service)`
- `metadata (e.g. name, UID, namespace)`
- `spec (depending on chosen object kind)`

Pod Objects

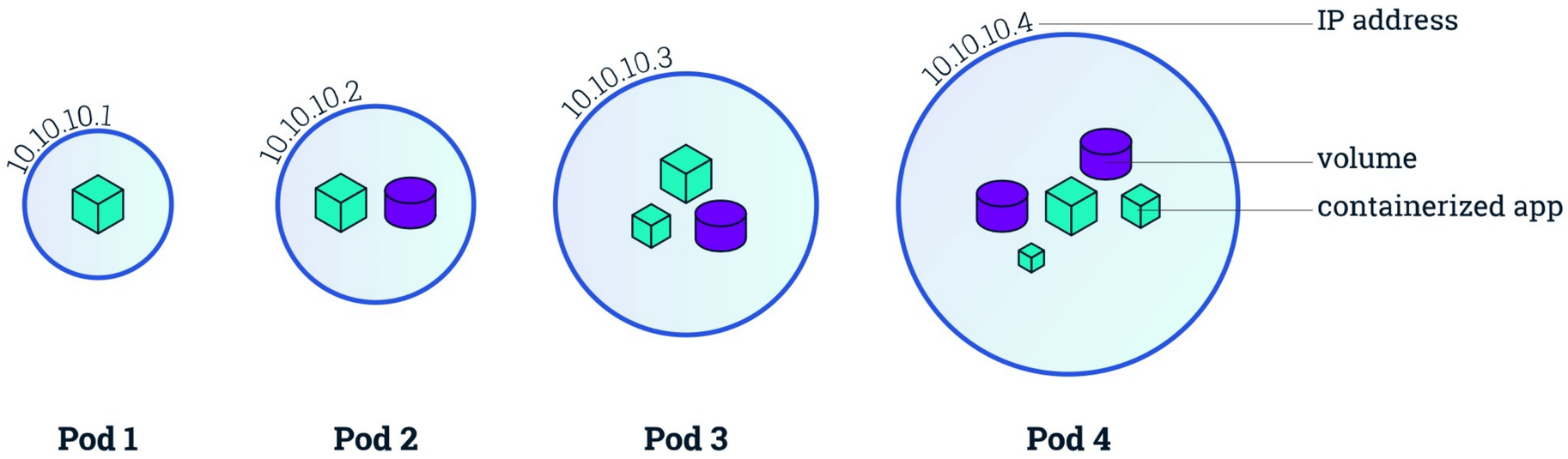
Encapsulates an application's

- Container(s)
- Storage resources (volumes shared between the containers)
- Network identity (IP address shared between the containers)

All containers of a Pod run on the same Node

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: static-web
  labels:
    app: mywebapp
spec:
  containers:
    - name: web
      image: nginx
      ports:
        - name: web
          containerPort: 80
          protocol: TCP
```

Pod Concept



https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/

Pod InitContainer

A Pod can also have one or more init containers, which are run before the app containers are started

Init containers always run to completion before the regular (main) containers are started

Each init container must complete before the next one starts

InitContainer Example Use Cases

Waiting for a service to be created

Delay the start of the app container

Clone a GIT repository into a volume

Initialize the content of a database

Register the Pod with a remote server

Generate the configuration for the main app container

InitContainer Example

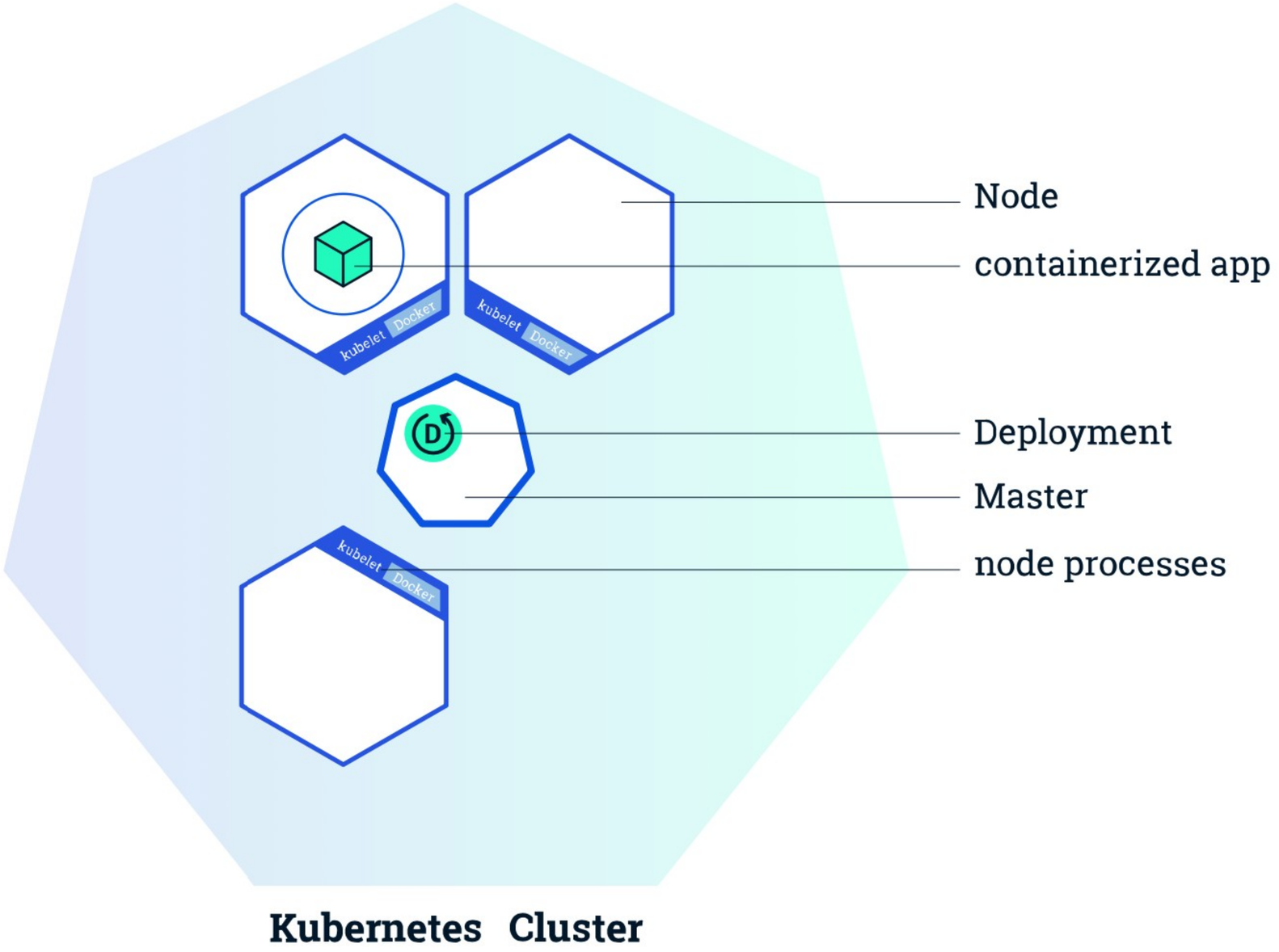
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app.kubernetes.io/name: MyApp
spec:
  containers:
  - name: myapp-container
    image: busybox:1.28
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
  initContainers:
  - name: init-myservice
    image: busybox:1.28
    command: ['sh', '-c', "until nslookup myservice.svc.cluster.local; do echo
waiting for myservice; sleep 2; done"]
```

Controller Objects

- Creates and manages multiple Pods
- Handles replication and rollout
- Provides self-healing at cluster-scope
- Types of controllers include “Deployment”, “DaemonSet”, “Jobs”

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
...
```

```yaml
...
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```



Deployment Concept

https://kubernetes.io/docs/tutorials/kubernetes-basics/deploy-app/deploy-intro/

Service Objects

Exposes an application running on a set of Pods as a network service

Abstraction which defines a logical set of Pods and a policy by which to access them

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

ClusterIP (default type)

- Exposes the service on a cluster-internal IP
- Service is only reachable from within the cluster

NodePort

- Exposes the service on each cluster node's IP

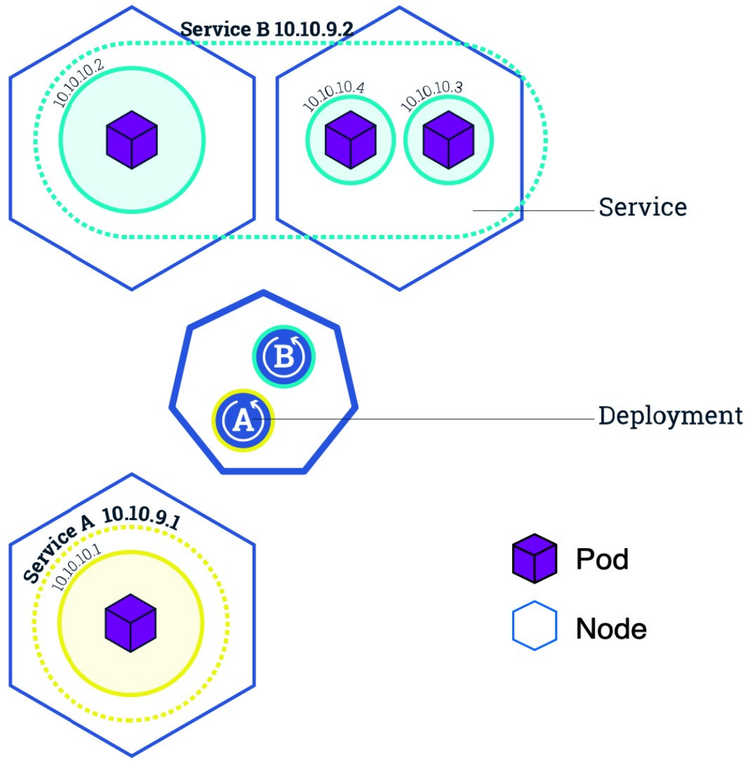
LoadBalancer

- Exposes the service externally using a cloud provider's load balancer

ExternalName

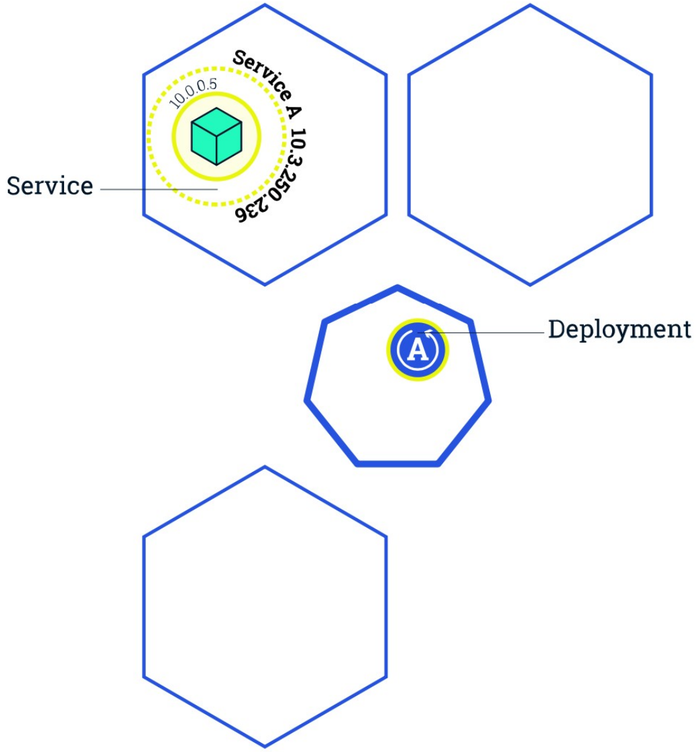
- Maps an internal cluster DNS name to an external service name using a `CNAME` record

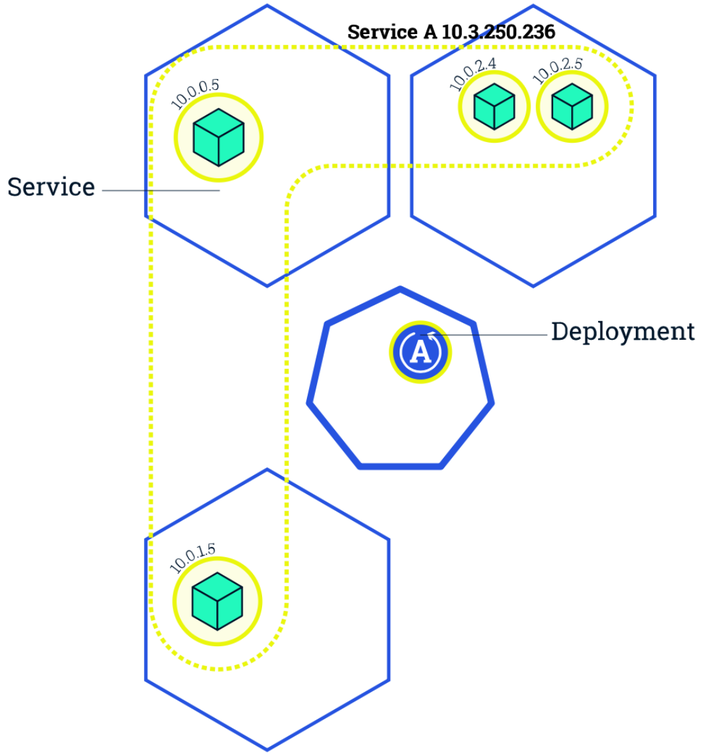
Service Concept



https://kubernetes.io/docs/tutorials/kubernetes-basics/expose/expose-intro/

Scaling Concept





https://kubernetes.io/docs/tutorials/kubernetes-basics/scale/scale-intro/

Ingress

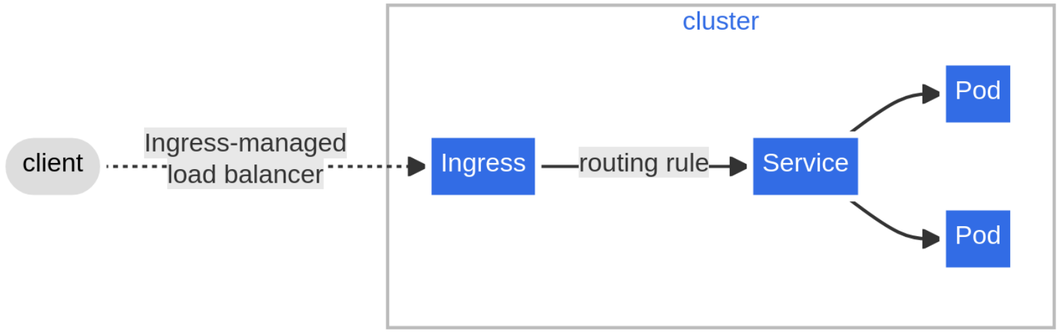
Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster

- Traffic routing is controlled by rules defined on the Ingress resource
- Does not expose arbitrary ports or protocols

Implemented by an Ingress Controller

- Several implementations are available, e.g. “ingress-nginx”
- Must be explicitly deployed

API is frozen, new features are being added to the Gateway API



https://kubernetes.io/docs/concepts/services-networking/ingress/

Ingress Resource Example

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx-example
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```

Name: must be a valid DNS subdomain name

Annotations: used to configure options depending on the Ingress controller

- Different controllers support different annotations
- E.g. “nginx.ingress.kubernetes.io/rewrite-target” specifies the target URI where the traffic must be redirected

Spec: provides the information to configure a load balancer or proxy server

- A reference to the **IngressClass** resource provides additional configuration including the name of the controller
- Contains a list of rules matched against all incoming requests

Kubernetes Storage (1)

Container filesystem

- Exists until container lifetime ends (e.g. crash)

Ephemeral Volume

- Exists until pod lifetime ends
- Can be shared between containers in pod
- Various types (emptyDir, configMap, secret, ...)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: redis
spec:
  containers:
  - name: redis
    image: redis
    volumeMounts:
    - name: redis-storage
      mountPath: /data/redis
  volumes:
  - name: redis-storage
    emptyDir: {}
```

Kubernetes Storage (2)

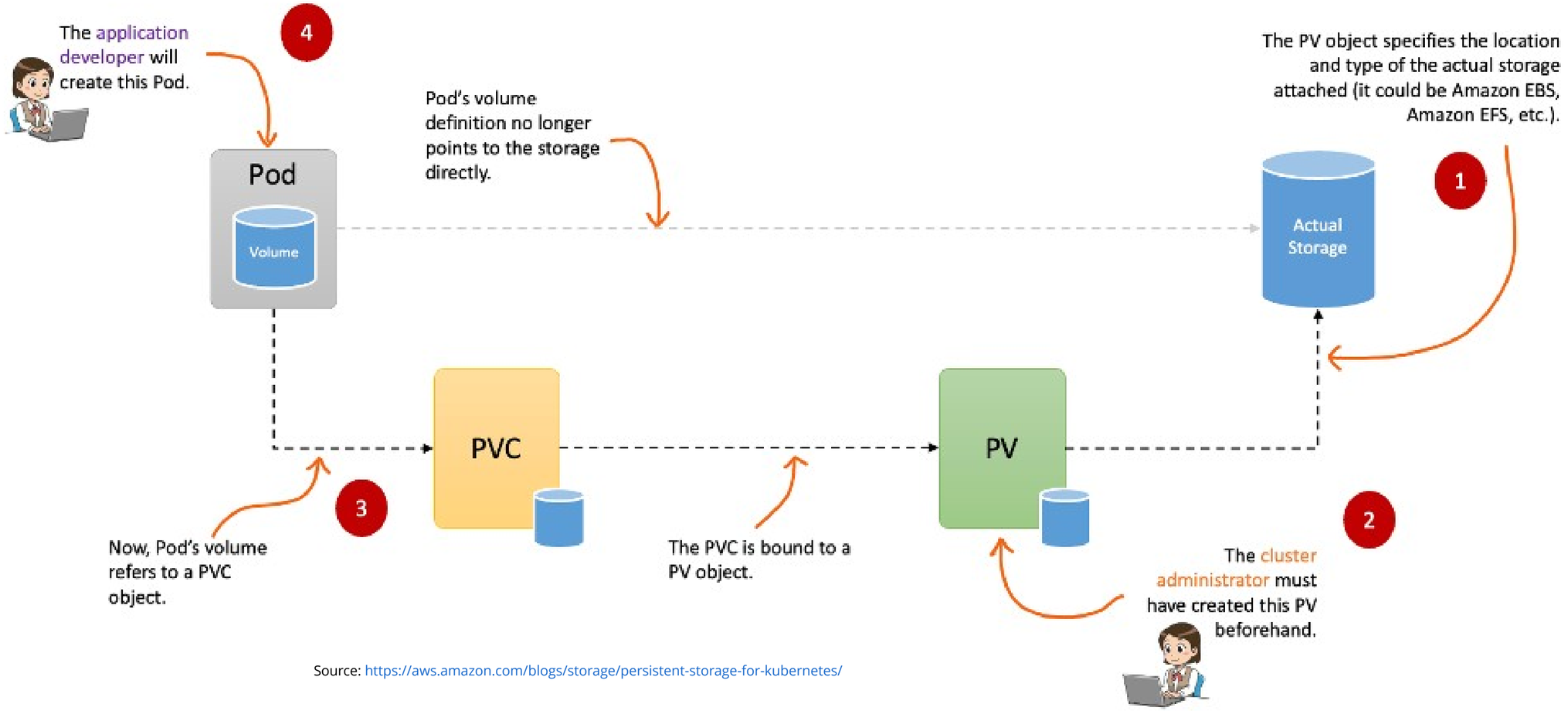
Persistent volume (PV)

- Exists until cluster lifetime ends
- Pre-configured by the administrator or dynamically provisioned by a CSI driver

Persistent volume claim (PVC)

- Abstracts details of how storage is provided from how it is consumed
- A Pod can not mount a PV object directly, it needs to ask for it
- That asking action is achieved by creating a PVC object and attaching it to the Pod (PVCs and PVs have a one-to-one mapping)

Kubernetes Storage (3)



Source: https://aws.amazon.com/blogs/storage/persistent-storage-for-kubernetes/

Kubernetes Storage (4)

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfsvol1
  labels:
    type: nfs
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: 192.168.49.1
    path: /srv/nfsvol1
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: dbvol
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 1Gi
  selector:
    matchLabels:
      type: nfs
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mydb
 spec:
  Template:
      volumes:
      - name: data-vol
        persistentVolumeClaim:
          claimName: dbvol
      containers:
      - name: db
…
        volumeMounts:
        - name: data-vol
          mountPath: /var/lib/mysql
```

ConfigMap

Used to store non-confidential data for pods in key-value pairs

- “data” field contains UTF-8 strings
- “binaryData” field contains binary data as base64-encoded strings

Pods can us ConfigMaps as

- Environment variables
- Command-line arguments
- Configuration files in a volume

The name of a ConfigMap must be a valid DNS subdomain name

ConfigMap Definition Example

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: game-demo
data:
  # property-like keys; each key maps to a simple value
  player_initial_lives: "3"
  ui_properties_file_name: "user-interface.properties"

  # file-like keys
  game.properties: |
    enemy.types=aliens,monsters
    player.maximum-lives=5
  user-interface.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
```

ConfigMap Usage Example

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: configmap-demo-pod
spec:
  containers:
    - name: demo
      image: alpine
      command: ["sleep", "3600"]
      env:
        - name: PLAYER_INITIAL_LIVES
          valueFrom:
            configMapKeyRef:
              name: game-demo
              key: player_initial_lives
```

Secrets

Similar to ConfigMaps but specifically intended to hold confidential data:

- Passwords, tokens or keys
- Allows to pull container images from private registries

There are different types of secrets for different use cases

- Opaque, kubernetes.io/dockerconfigjson, kubernetes.io/ssh-auth, ...
- See https://kubernetes.io/docs/concepts/configuration/secret/#secret-types

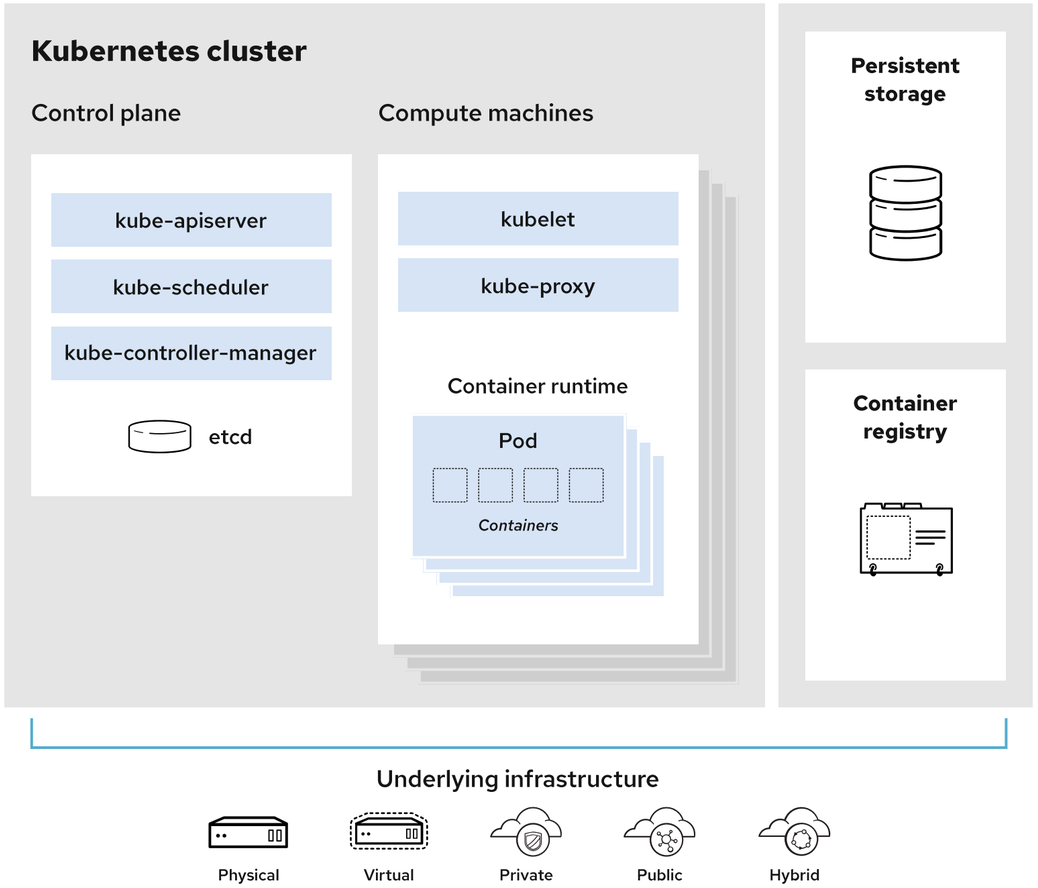
Caution: Secrets are, by default, stored unencrypted

- Anyone with API access can retrieve or modify a Secret
- See https://kubernetes.io/docs/concepts/security/secrets-good-practices/

Secret Example

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: secret-ssh-auth
type: kubernetes.io/ssh-auth
data:
  # the data is abbreviated in this example
  ssh-privatekey: |
    UG91cmluZzYlRW1vdGljb24lU2N1YmE=
```

Summary



https://www.redhat.com/en/topics/containers/what-is-kubernetes

Any Questions ?

Further Reading

https://kubernetes.io