TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Macroarea di Ingegneria
Dipartimento di Ingegneria Civile e Ingegneria Informatica

Kubernetes

Corso di Sistemi Distribuiti e Cloud Computing
A.A. 2021/22

Valeria Cardellini

Laurea Magistrale in Ingegneria Informatica

Container orchestration

- Platforms for managing the deployment of multi-container packaged applications in large-scale clusters
- Allow to configure, provision, deploy, monitor, and dynamically control containerized apps
  - Used to integrate and manage *containers at scale*
- Examples
  - **Docker Swarm**
  - **Kubernetes**
  - Amazon Elastic Container Service
  - Google Kubernetes Engine
  - Marathon
  - Nomad (container orchestration platform for Mesos)

Fully managed Cloud services

Container management systems at Google

- Application-oriented shift

  "Containerization transforms the data center from being machine-oriented to being application-oriented"

- Goal: let container technology operate at Google scale
  - Everything at Google runs as a container
  - Google launches more than 2 billion containers per week

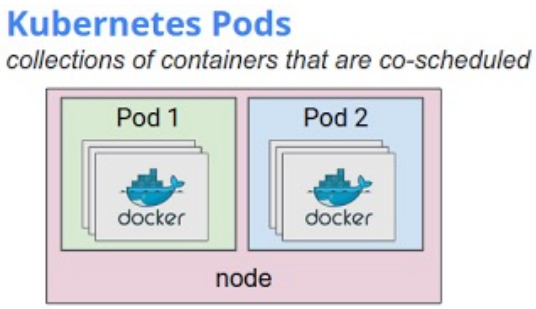
- Borg -> Omega -> Kubernetes
  - Borg and Omega: purely Google-internal systems, precede Kubernetes
  - Kubernetes: open-source

Kubernetes

- Google's open-source platform for automating deployment, scaling, and management of containerized apps across clusters of hosts
  http://kubernetes.io
- Features:
  - Portable: public, private, hybrid, multi-cloud
  - Extensible: modular, pluggable, hookable, composable
  - Self-healing: auto-placement, auto-restart, auto-replication, auto-scaling of containers
- Can run on public or private cloud platforms (AWS, Azure, OpenStack, Apache Mesos), and also on bare metal machines
- Offered as Cloud service by main providers
  - Kubernetes management and deployment on underlying infrastructure is up to Cloud provider

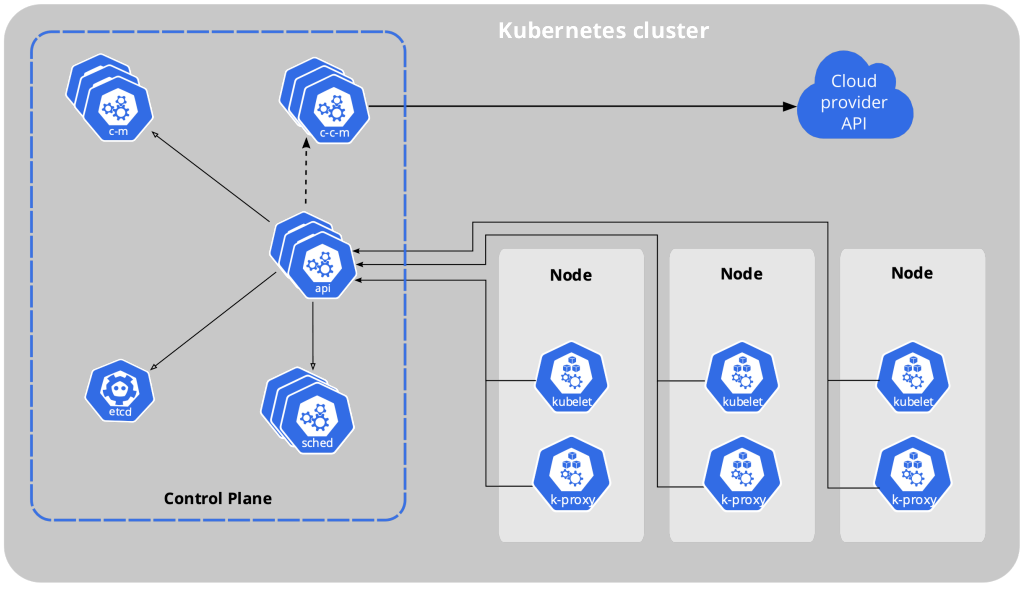
Kubernetes: pod

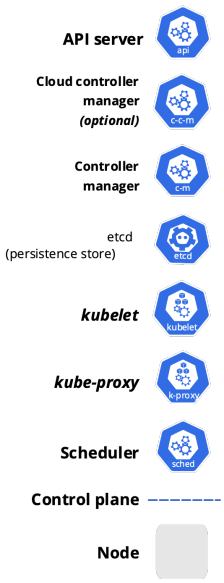
- **Pod**: the smallest deployable compute object in Kubernetes
  - Set of (tightly coupled) containers with shared storage/network, and a specification for how to run the containers
  - Pod containers are bundled and scheduled together, and run in a shared context



  - Kubernetes gives pods their own IP addresses and a single DNS name for a set of pods, and can load-balance across them
- Users organize pods using labels
  - Label: arbitrary key/value pair attached to pod
  - E.g., role=frontend and stage=production

Kubernetes: architecture





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

Architecture: control plane

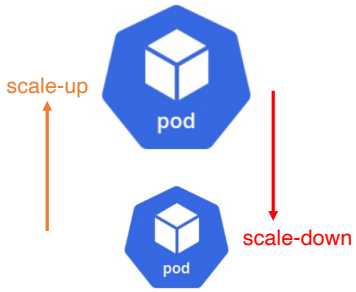
- Organized according to master-worker pattern
- **Kubernetes control plane**: cluster's master, takes global decision about the cluster (e.g., scheduling) as well as detecting and responding to cluster events (e.g., starting up a new pod)
  - Multiple master nodes can be set to provide a cluster with failover and high availability
- Main components of control plane
  - ***kube-apiserver***: API server that exposes Kubernetes API, it is the front end for Kubernetes' control plane
  - ***etcd***: highly available distributed key-value store, used as Kubernetes' backing store for all cluster data
  - ***kube-scheduler***: decides how to assign pods to nodes (placement)

Architecture: nodes

- **Kubernetes nodes**: worker nodes (can be VM or physical machines) that maintain multiple running pods and provide Kubernetes' runtime environment
- Main components on worker nodes
  - ***kubelet***: agent ensuring that pods running on node are healthy and running
  - ***kube-proxy***: network proxy that maintains network rules on nodes

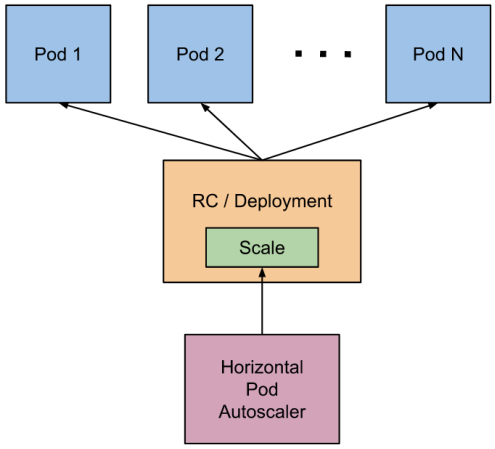
Kubernetes: Auto-scaling

- Multiple auto-scalers at different control layers
  - Cluster auto-scaling with node granularity
  - Horizontal and vertical auto-scaling with pod granularity
- Cluster Autoscaler: adjusts size of Kubernetes cluster
- Vertical Pod Autoscaler (VPA): scales amount of pod resources (CPU, memory)
  - Based on historical resource usage of pods: decaying histogram of weighted CPU and memory usage
  - Requires restarting pods: this disrupts the continuity of applications and services



Auto-scaling: HPA

- Horizontal Pod Autoscaler (HPA): scales number of pods in a deployment, replica set or stateful set
  https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/
  - Based on observed CPU utilization (or, with custom metrics support, on some other application-provided metrics
  - Creates new pods without affecting existing ones

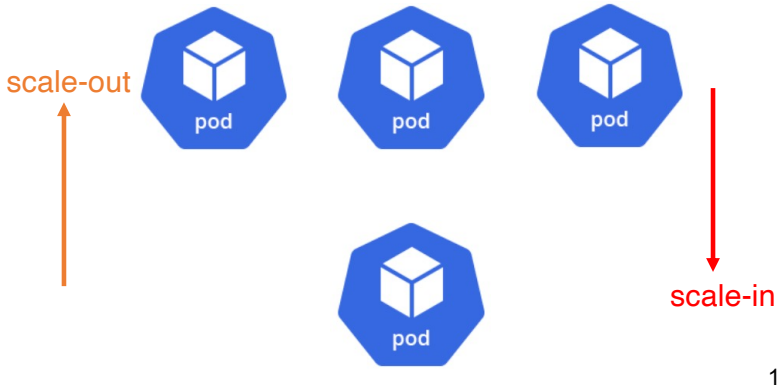


Auto-scaling: HPA

- HPA policy: heuristic, variant of threshold-based policy
  - Scales number of pods according to ratio between observed value and target value

$$desiredReplicas = \left\lceil currentReplicas \frac{currentMetricValue}{desiredMetricValue} \right\rceil$$

- Stabilization time window to limit fluctuations in replicas number



Kubernetes distributions

- Multiple options

1. Install Kubernetes from source code
2. Pure distributions: pre-built Kubernetes
   - E.g., Charmed Kubernetes
3. Plus distributions: platforms that integrate Kubernetes with other specific technologies (e.g., container runtimes, host OSes or control-plane add-ons)
   - E.g., Red Hat OpenShift
4. Kubernetes-as-a-service: Kubernetes in the Cloud
   - E.g, Azure AKS, AWS EKS, Google GKE
5. Limited-purpose distributions: intended for a specific and limited purpose (e.g., single-node, DevOps, edge & IoT)
   - E.g., kind, minikube, MicroK8s, K3S

Kubernetes tools

- Some useful tools
- kubectl: Kubernetes command-line tool to run commands against Kubernetes cluster
  - Can use kubectl to deploy applications, inspect and manage cluster resources, and view logs
- Metrics Server: scalable, efficient source of container resource metrics
  - Collects resource metrics from Kubelets and exposes them in Kubernetes apiserver through Metrics API for use by Horizontal Pod Autoscaler and Vertical Pod Autoscaler