CLOUD COMPUTING

CONTAINER ORCHESTRATION

Lorenzo Leonini
lorenzo.leonini@unine.ch

Dev vs Ops

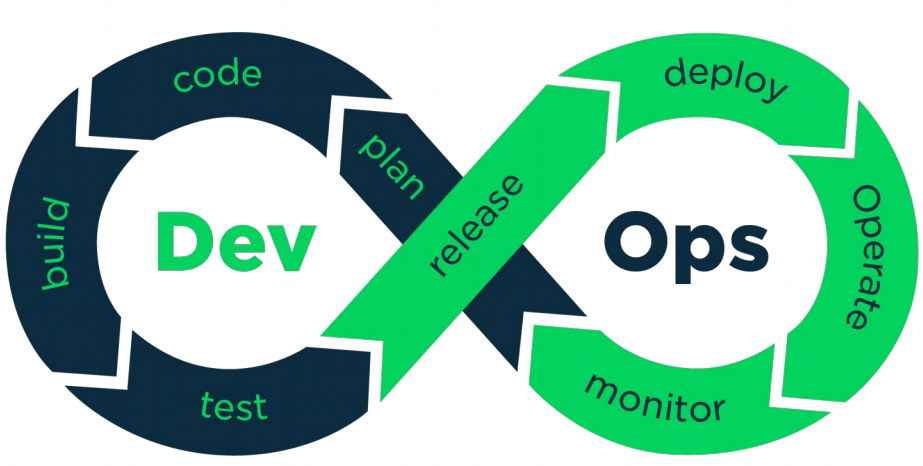
- Different teams with different responsibilities and goals

- **Development**
  - Create **changes**
    - Project mode
  - Manage software releases
  - As fast as possible

- **Operations**
  - Manage infrastructure, OS, runtime and software
  - Responsible to ensure the production work as expected
  - Changes = opportunities to break something

Devops

- Removing the barriers between development and operations
  - Continuous integration
  - Continuous delivery
  - Infrastructure as code
  - Monitoring
  - Microservices
  - Collaboration



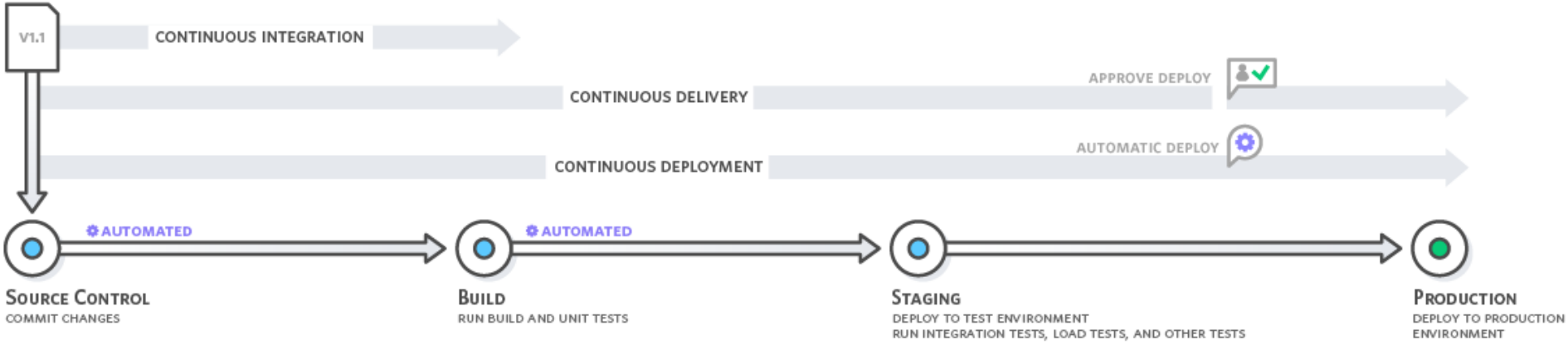
Devops - Infrastructure

- Classical approach
  - VMWare UI
    - Manual/API provisioning
  - SSH remote control
  - Scripts (shell)
  - Puppet / Chef / Ansible
  - Monitoring
    - HTTP GET
    - CPU/Memory/Disk

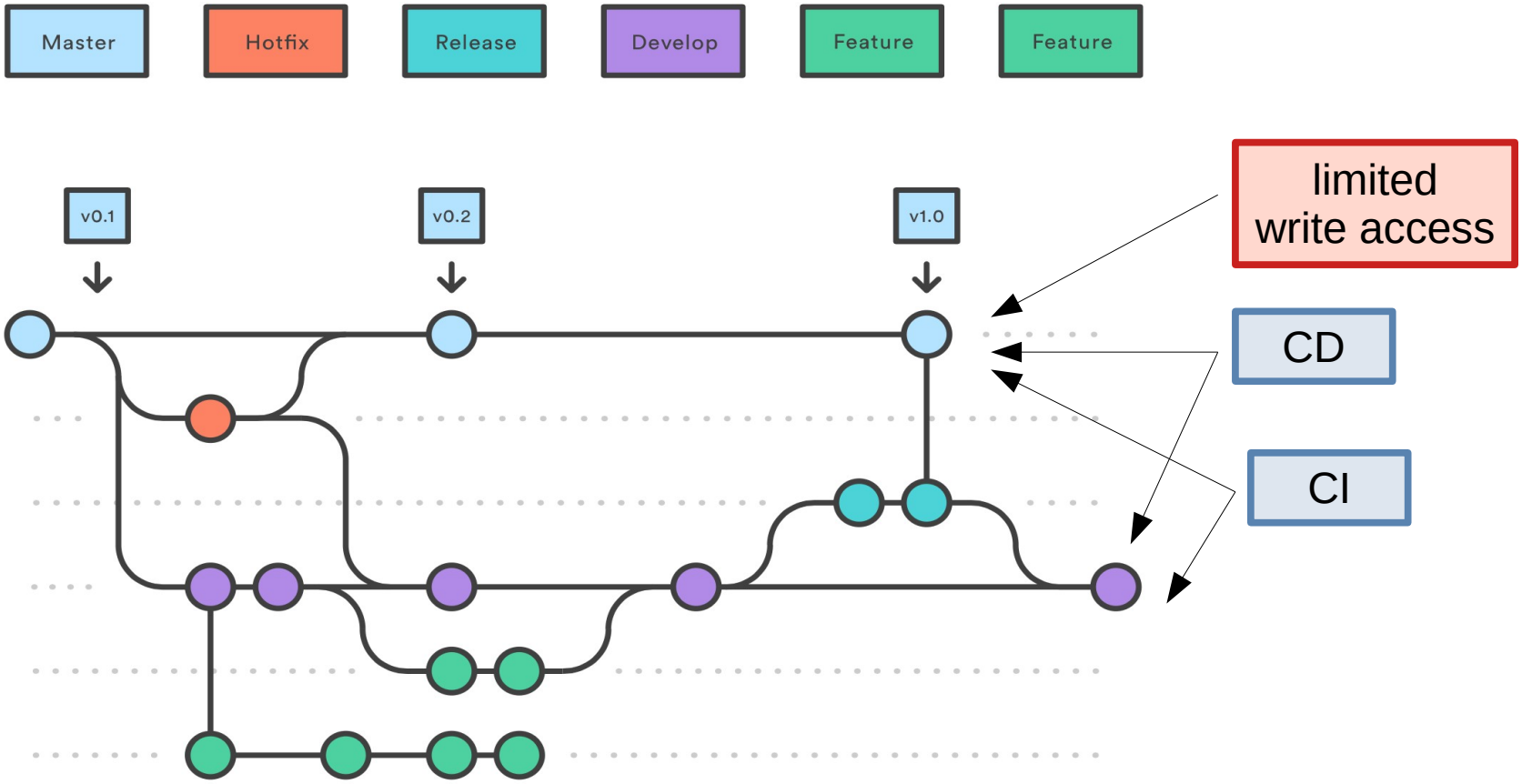
- Devops
  - Declarative architecture
    - Services
    - Network
    - E.g. Dockerfile, docker-compose.yml
  - Automation
    - Continuous Integration/Delivery
      - Problems => email / slack
  - Monitoring
    - Classical and custom metrics
    - => Autoscaling
      - Problems => email / slack

Devops – Continuous *

- Integration
  - Central repository
  - Automated builds
  - Automated tests
- Delivery
  - Deploy
    - Testing/Staging
    - Production



Devops - gitflow



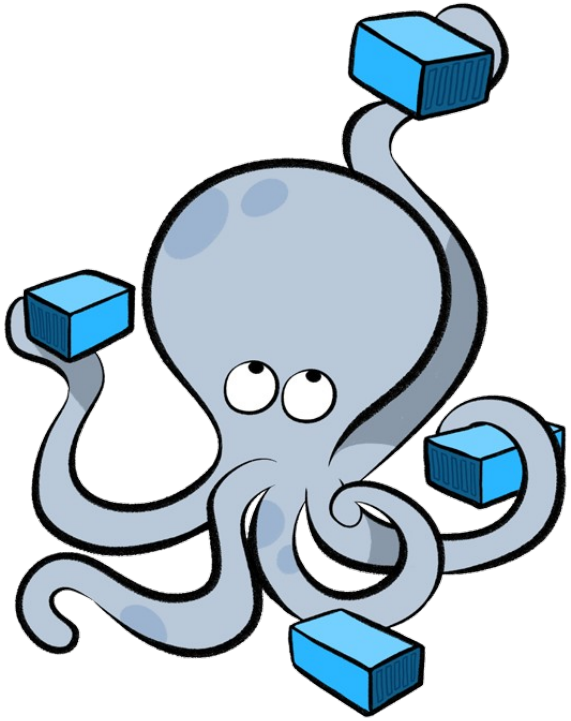

docker
SWARM

Docker

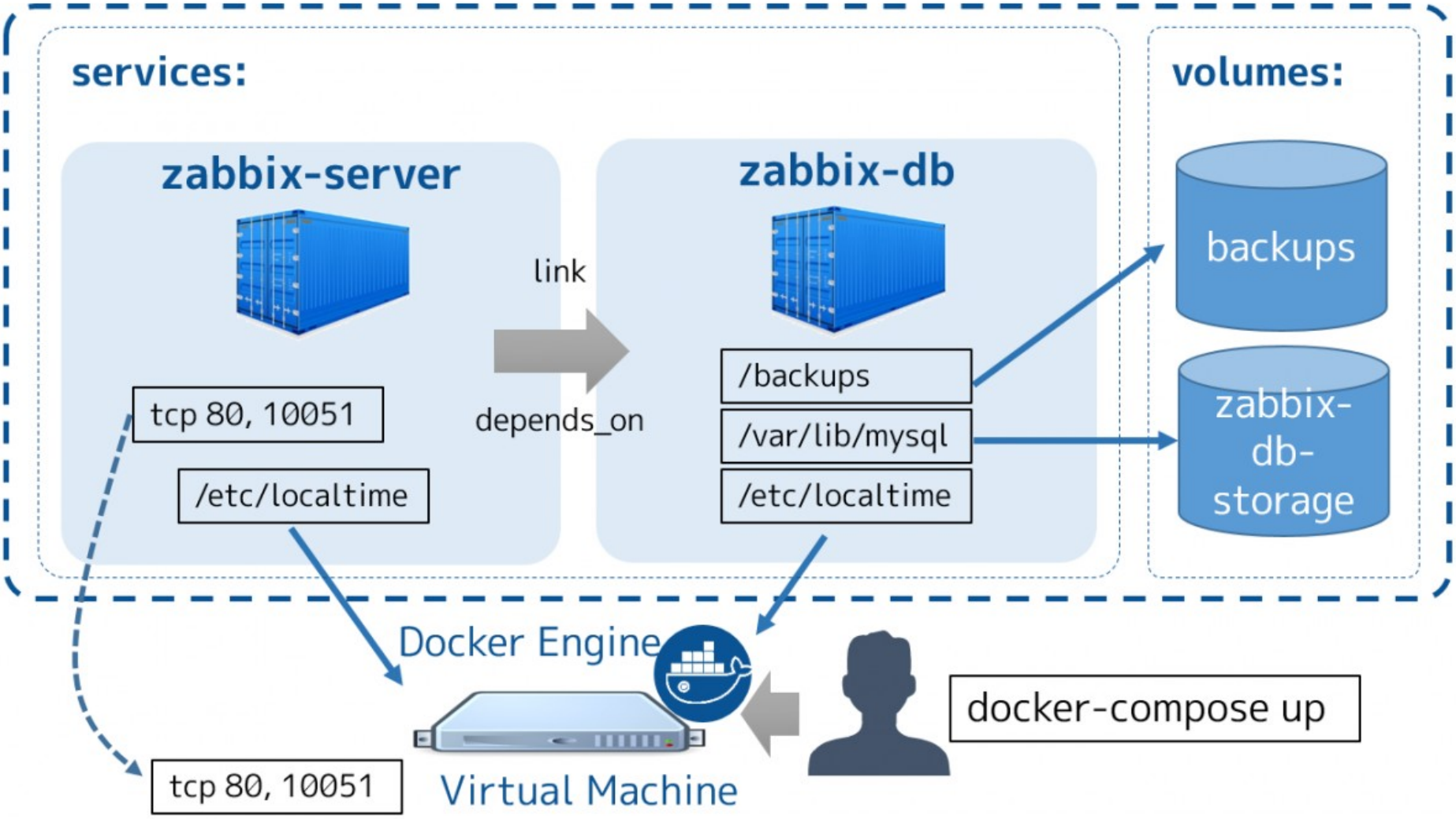
- Image
  - Application files and runtime dependencies
  - Config (and ENV vars support)
  - Runtime config
    - Executable to launch
    - Network ports
    - Volumes
- Container
  - Isolation
  - Resource limits
- Dockerfile
  - Declarative way to build an image

Docker compose

- “Service” description
  - How to compose individual images to run a service
  - Shared network
  - Startup dependencies
  - Re-launch behavior
  - Network ports
  - Volumes



docker-compose.yml (v2 format)



Docker

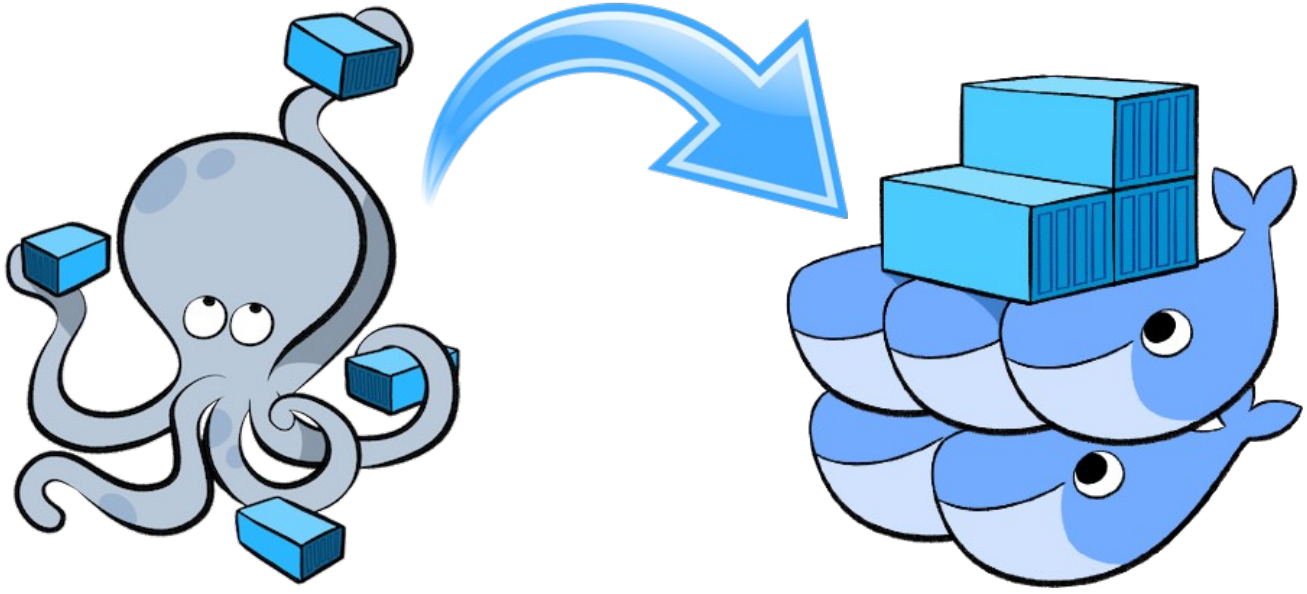
- How to deploy a “docker-compose” in the cloud ?
- Need a **cluster of nodes**
  - **VMs** or physical computer able to run containers
- Need **virtual networking**
  - Let containers on different hosts communicate together
  - **Service discovery**
- Need a **scheduler** will choose the most appropriate nodes where executing the service
  - Like OpenStack does when providing VMs in IaaS

What is orchestration ?

- **Provisioning** and **deployment** of containers
  - Allocation of resources between containers
  - External exposure of services running in a container
  - **Service discovery** between containers
- **Redundancy** and **availability** of containers
  - Health monitoring of containers and hosts
- **Scaling** up or removing containers
- Configuration of an application in relation to the containers running it

Docker Swarm

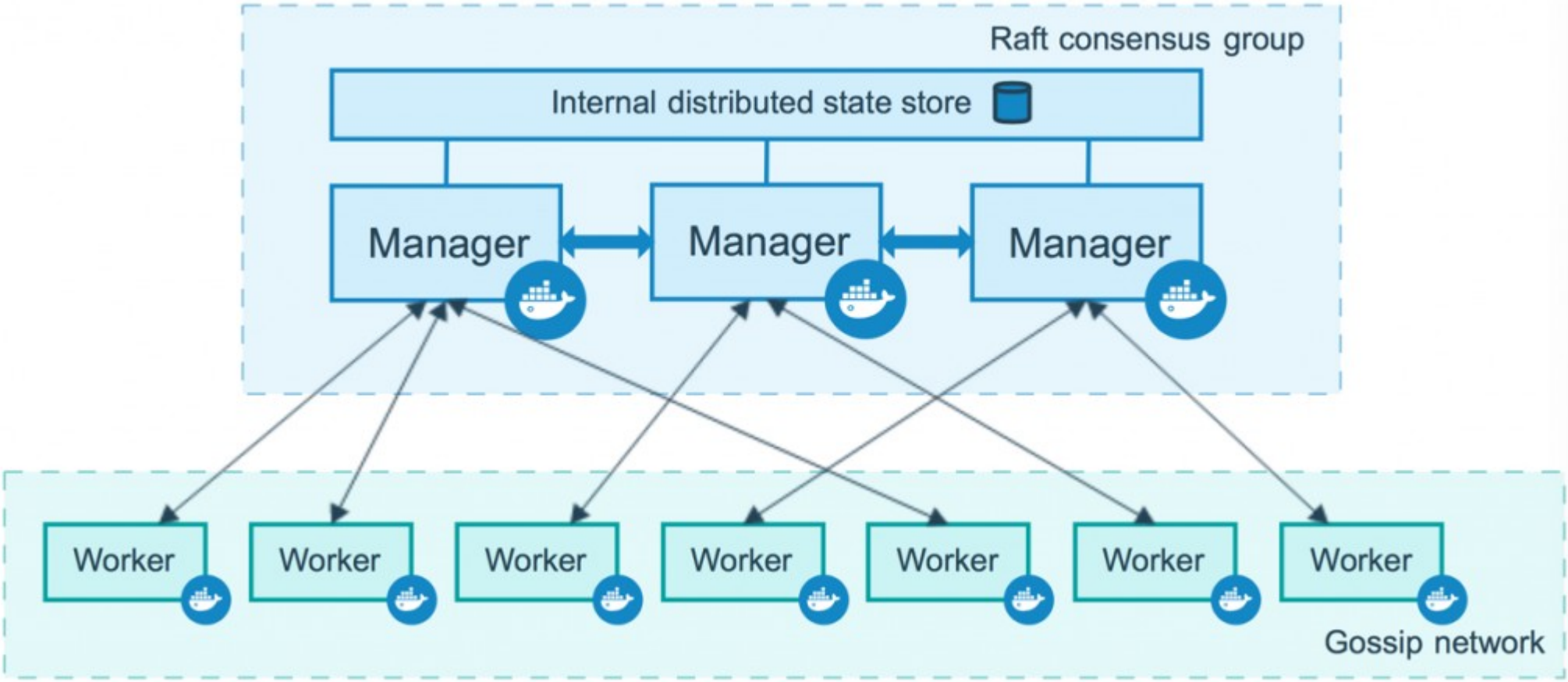
- Container Orchestration from Docker
- Require docker-compose v3



Docker Swarm - features

- Cluster management
- Decentralized design
- Scaling
- Declarative service model
- Desired state reconciliation
- Multi-host networking
- Service discovery
- Load balancing
- Secure by default
- Rolling updates

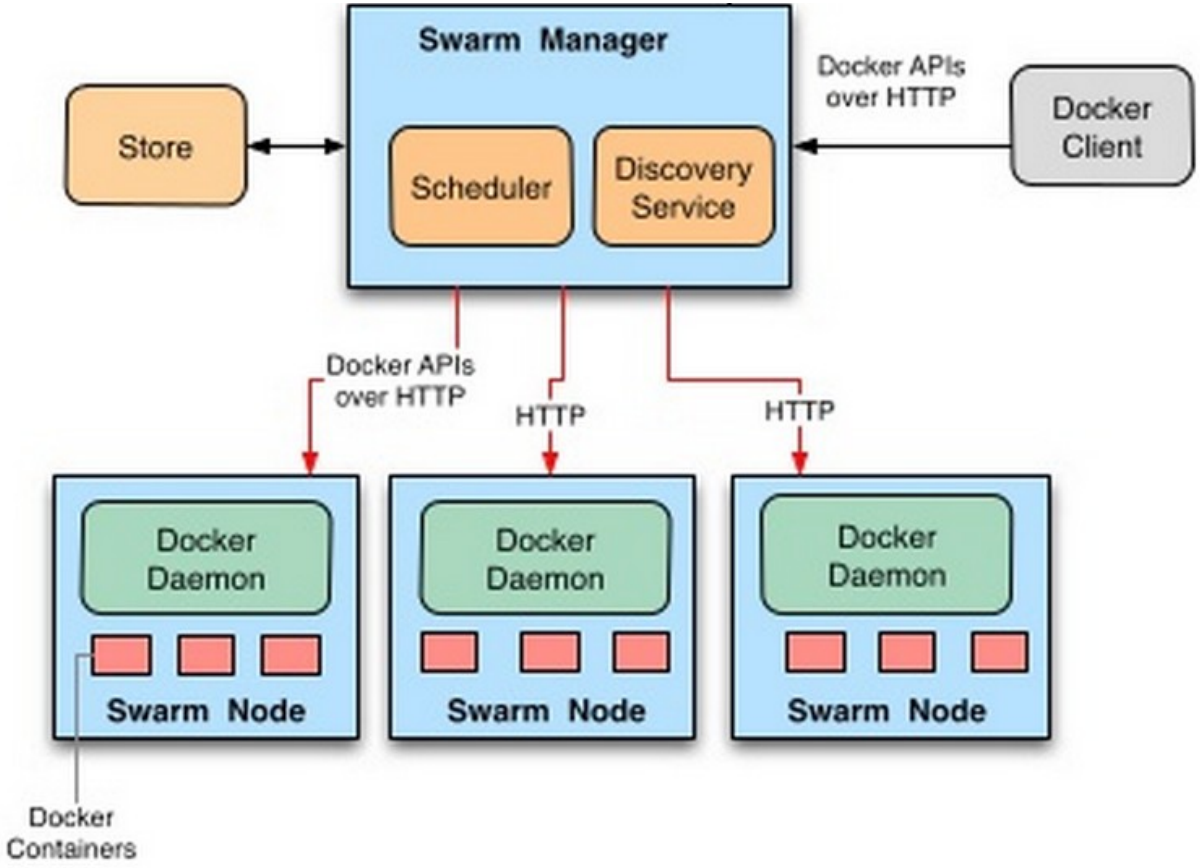
Docker Swarm - architecture



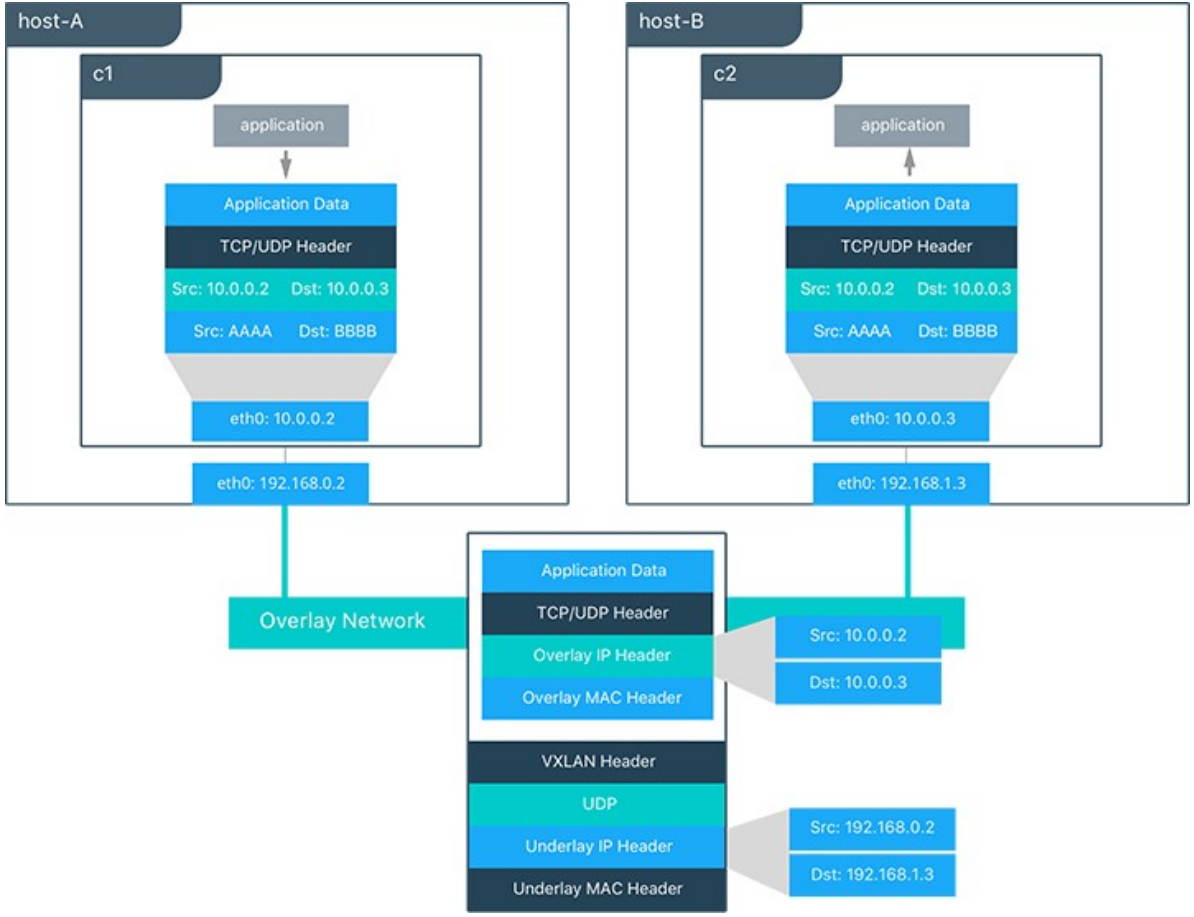
Docker Swarm - nodes

- Instances of the Docker engine participating in the swarm
- Manager node
  - Maintain the desired state of the swarm
  - Dispatch tasks to worker nodes
- Worker node
  - Execute assigned tasks
  - Report to the manager
  - By default manager modes can also act as workers

Docker Swarm - internals



Docker Swarm - network



Docker Swarm - scheduling strategies

- Spread (default)
  - Run on the node that as the least containers running
- Binpack
  - Try to fill nodes as much as possible
  - Avoid fragmentation
- Random
  - Assign to random nodes

Docker Swarm - filters

- Schedule based on specific **host properties**
  - `$ docker daemon --label storage=ssd`
  - On run: `-e constraint:storage==ssd`
  - On build: `$ docker build --build-arg=constraint:storage==disk`
- Force containers to run in a given location or sub-cluster partition
  - `region=us-east`
  - `environment=production`
- **Affinity**
  - Use an affinity filter to create "**attractions**" between containers to schedule it next to another container based on these affinities
    - `-e affinity:container==frontend`

Docker Swarm - commands

- $ docker swarm *
  - To initialize/join manager/worker nodes
- $ docker service *
  - To launch a service on the swarm (~ docker run)
- $ docker stack *
  - To launch a composed service on the swarm
  - -f docker-compose.yml (v3)

Docker Swarm - tutorial

kubernetes

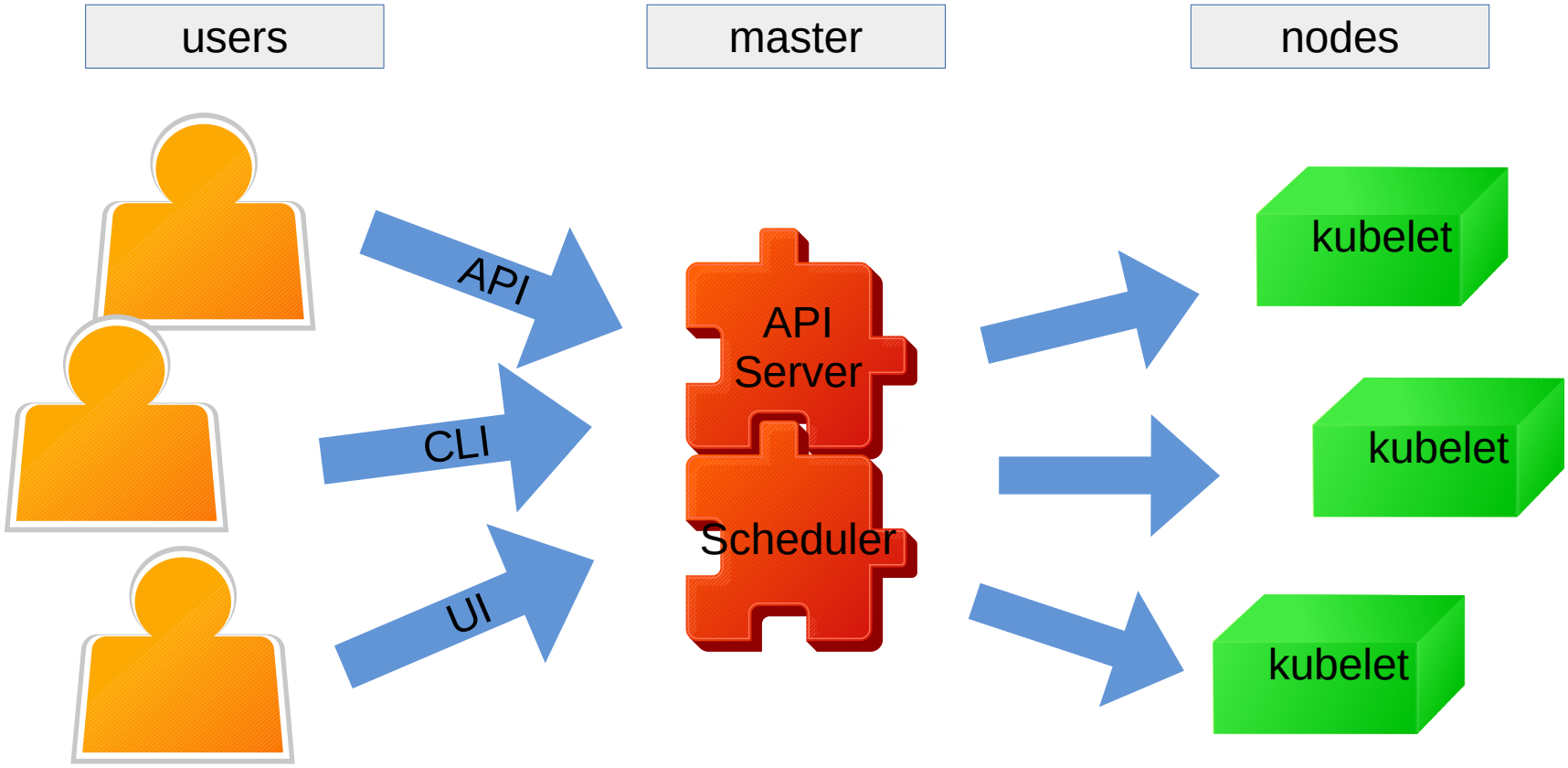
kubernetes

- Kubernetes is a **container orchestrator** (runs and manage containers)
- Google has been using Linux containers on a large scale for more than 10 years
- Large number of hosts and need for advanced features led to development of cluster management systems
  - Dynamic configuration, service discovery, auto-scaling, quota management, security management, etc.
- Three generations of systems
  - Borg (closed source)
  - Omega (closed source)
  - Kubernetes (open source since 2014)

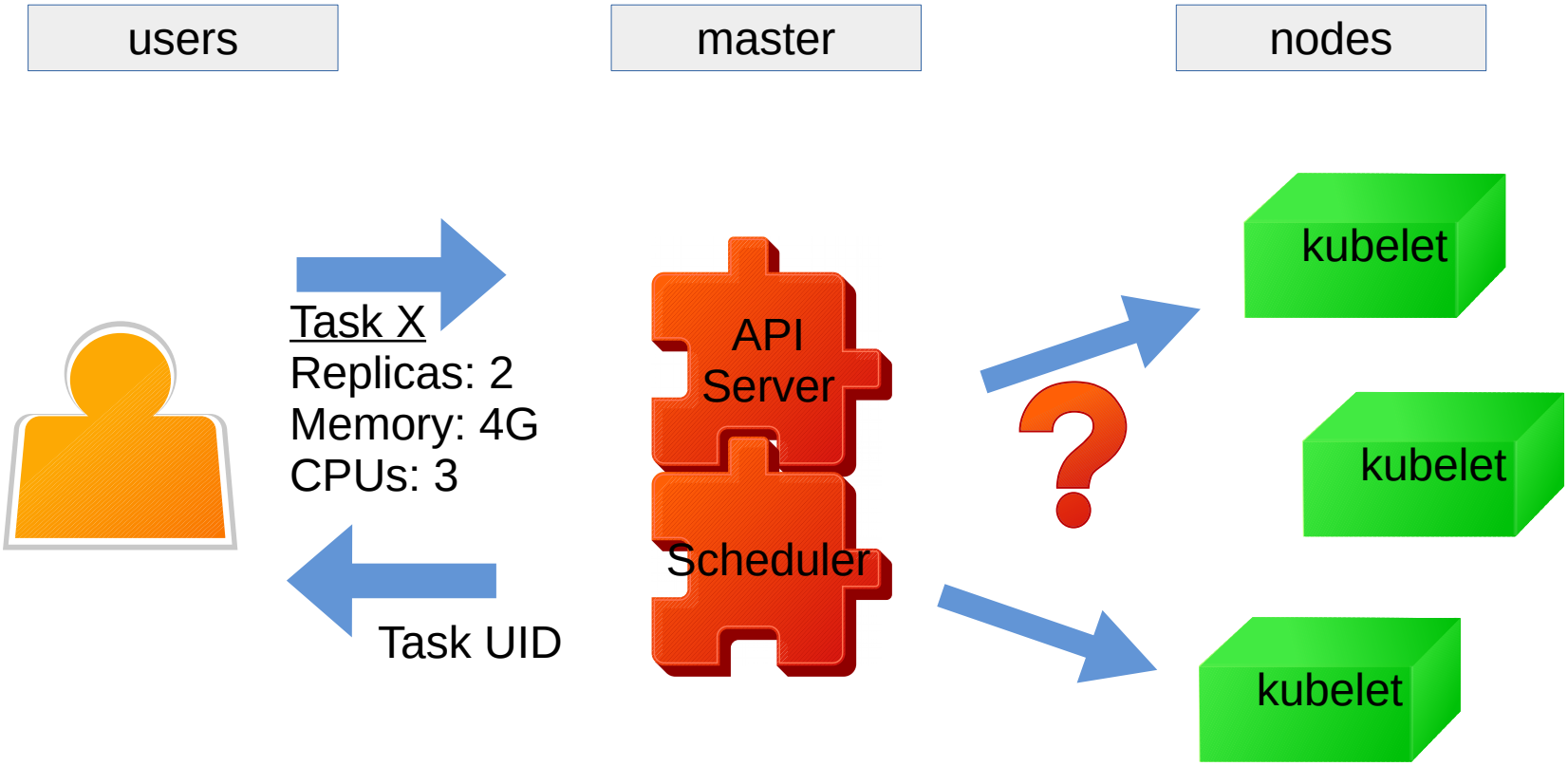
kubernetes

- Everything in Google works in containers:
  - Gmail, Search, Maps, ...
  - GFS, Mapreduce, batch, ...
- **Google launch 2 billion containers per week !**
  - Update
  - Service discovery
  - Scaling, replication, sets
- Support multiple clouds and bare-metal (OpenStack)
- Open source, written in **Go**
- Focus on **applications** and **jobs** not machines !
- Now used by many other companies (Core OS, Red Hat, Docker, ...)

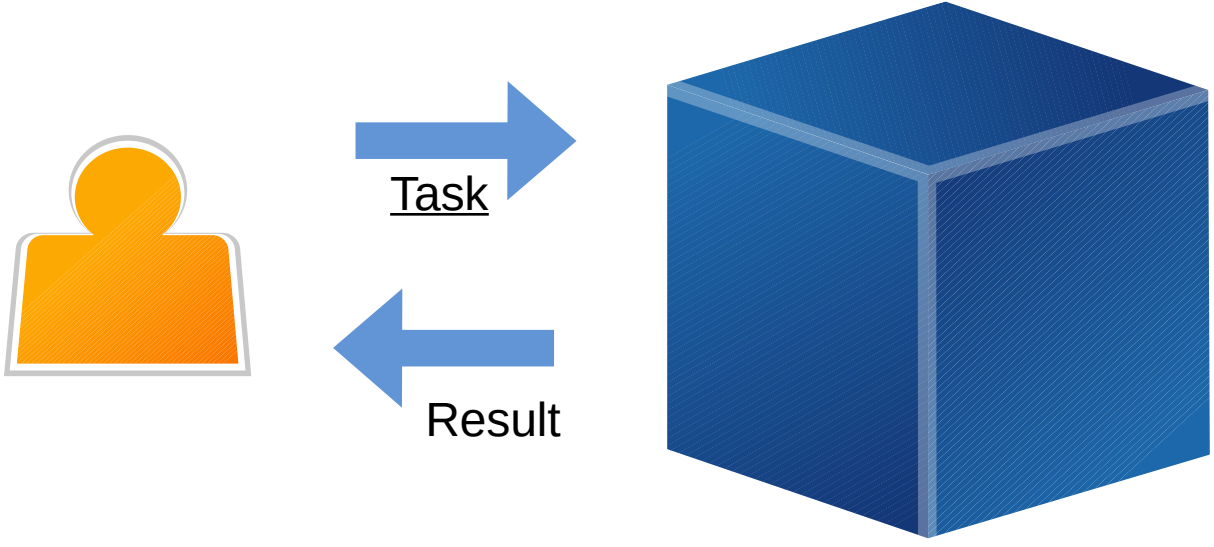
kubernetes - 1000 foot view



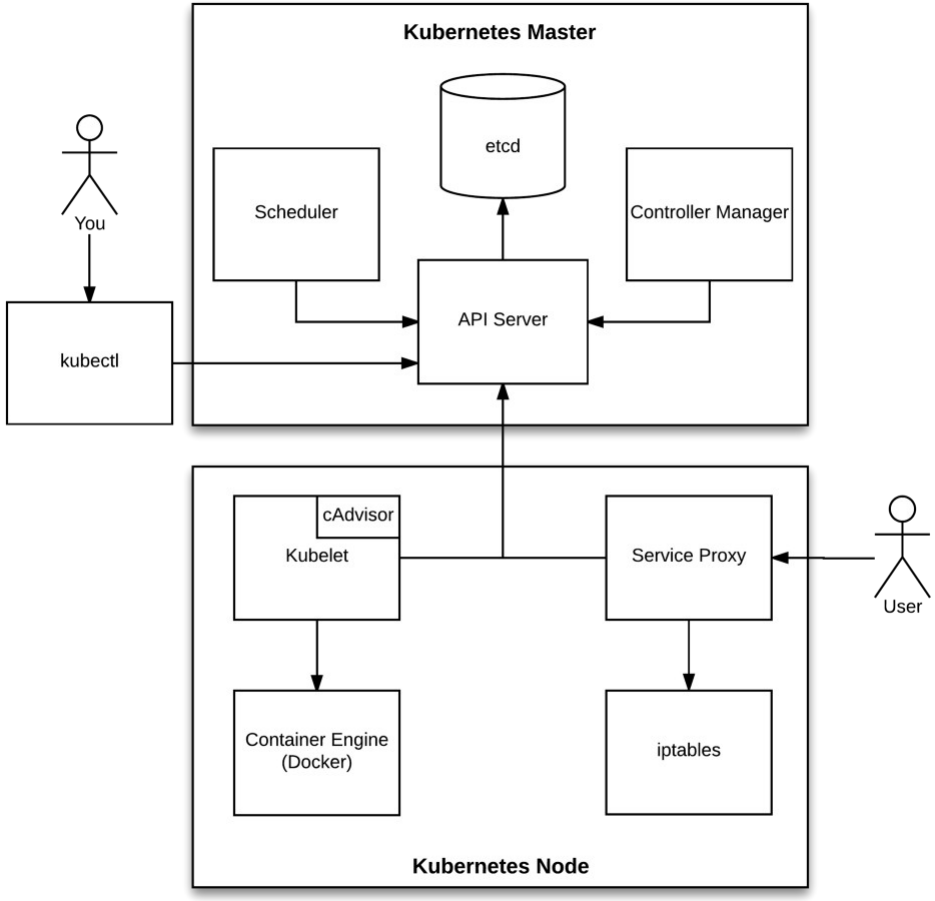
kubernetes - 1000 foot view



kubernetes - 1000 foot view



kubernetes - architecture



kubernetes - architecture

- Nodes
  - Master node
  - Worker node
- Objects
  - Pod
  - Service
  - Ingress
  - Volume
  - Namespace
- Controllers
  - ReplicaSet
  - Deployments
  - StatefulSets
  - DaemonSet
  - Job
  - CronJob
  - ...

kubernetes - objects

- Represent **states** and **intent** in the system
  - Application running
  - Resources used
  - Application policies: restart, upgrade and fault-tolerance
- Nested objects
  - spec
    - Desired state of the object
  - status
    - Actual state of the object

kubernetes - master processes

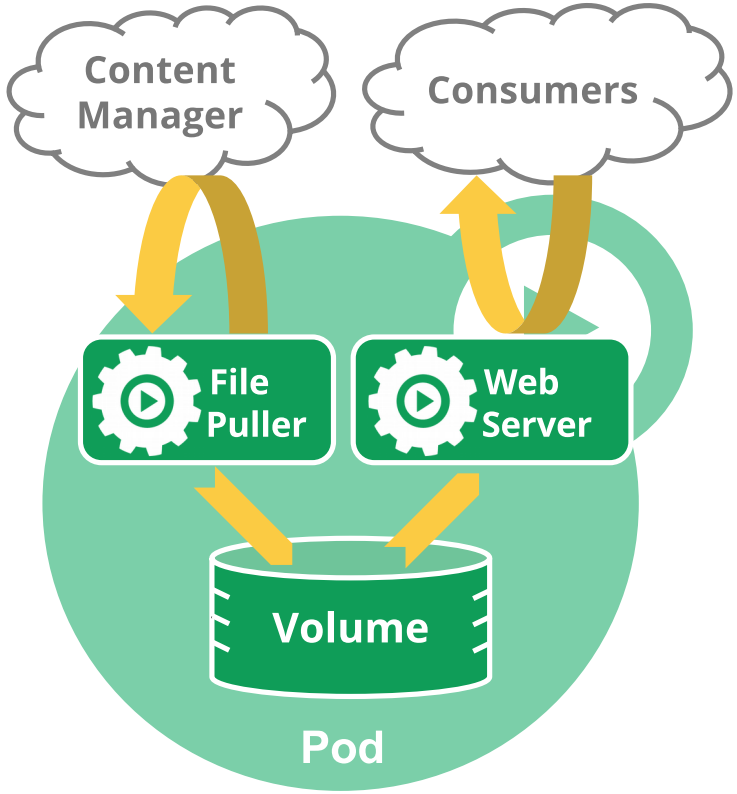
- kube-apiserver
  - REST cluster frontend
- kube-controller-manager
  - Non terminating control loop that regulates the system
  - Move state towards the desired state
- kube-scheduler
  - Impacts availability, performance, and capacity
  - Policy-rich
  - Topology-aware
- etcd
  - High availability DB

kubernetes - worker processes

- kubelet
  - Node "agent"
  - Receive PodSpec
    - YAML or JSON description of pods
  - Execute containers
    - Ensure that container are running and healthy (probe them)
    - Report status
      - cAdvisor: aggregate metrics on container resource usages
- kube-proxy
  - Watch API server for changes in services or pods definitions
  - Ensure that every nodes and containers can talk to each other

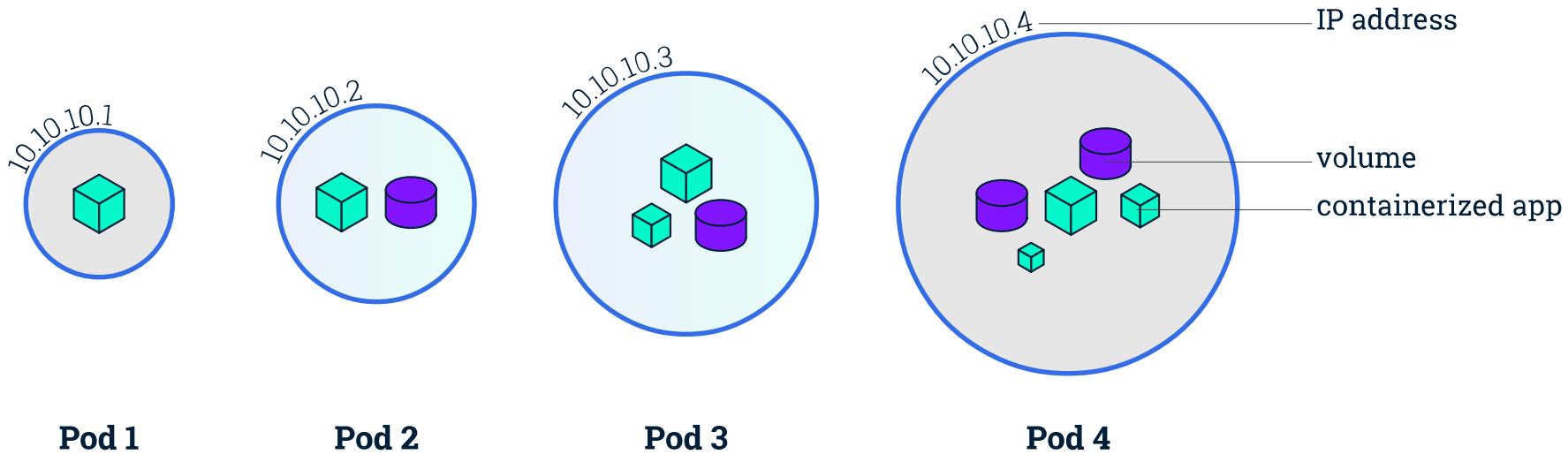
kubernetes - pods

- Basic building block of Kubernetes
  - Smallest and simplest **unit** in the Kubernetes object model
  - Represents a running process on your cluster
- One or **multiple containers** that are tightly coupled
  - Shared volumes
  - Shared memory
  - Unique IP address & localhost
  - IPC
- Mortal



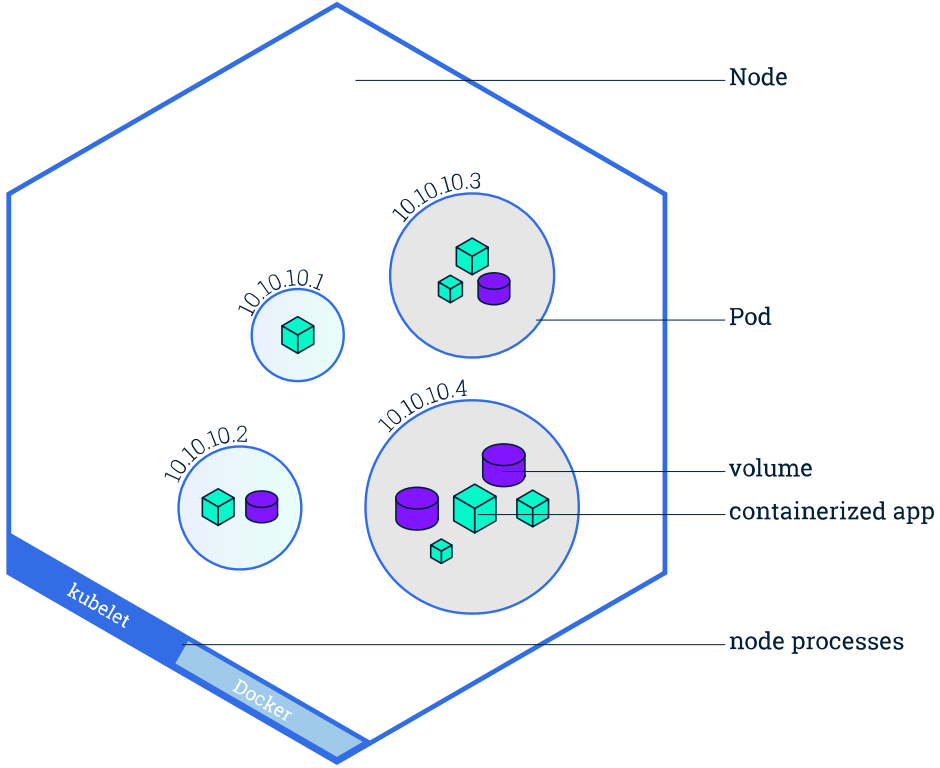
kubernetes - pods

- Pods can run individually or be managed by controlers



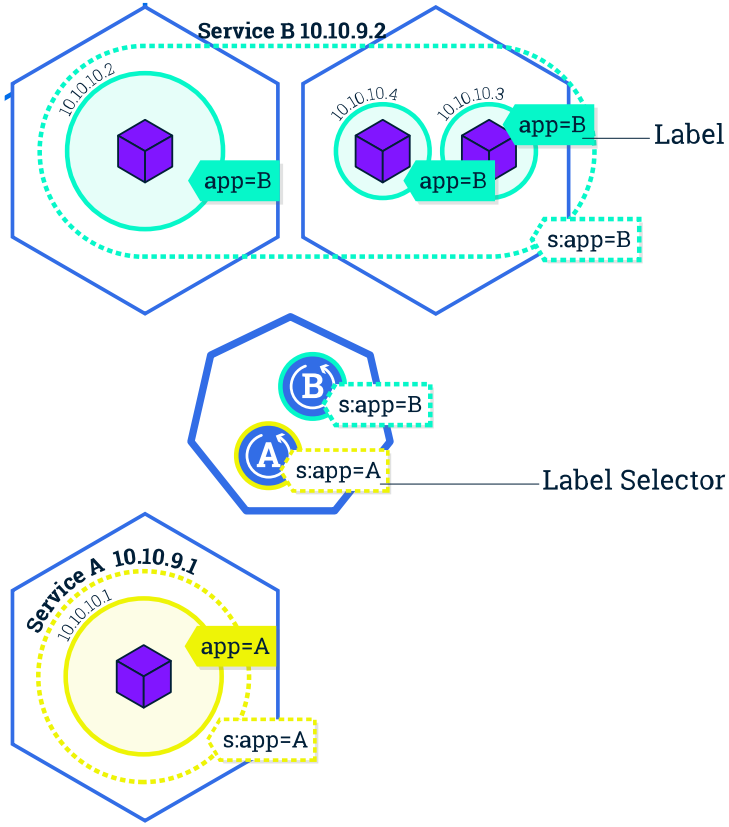
kubernetes - nodes

- Node
  - Running on virtual or bare metal machine
- kubelet
  - Run pods inside containers
    - Docker
    - Rocket
    - LXC



kubernetes - services

- Abstraction to a logical set of pods
  - Grouped by a selector
- Define access policy
- Get a stable virtual IP and port
  - Called a service portal
  - Name become a DNS
- VIP is captured by kube-proxy
  - Watches the services constituency
  - Update when backend changes
- Hides complexity - ideal for non-native apps



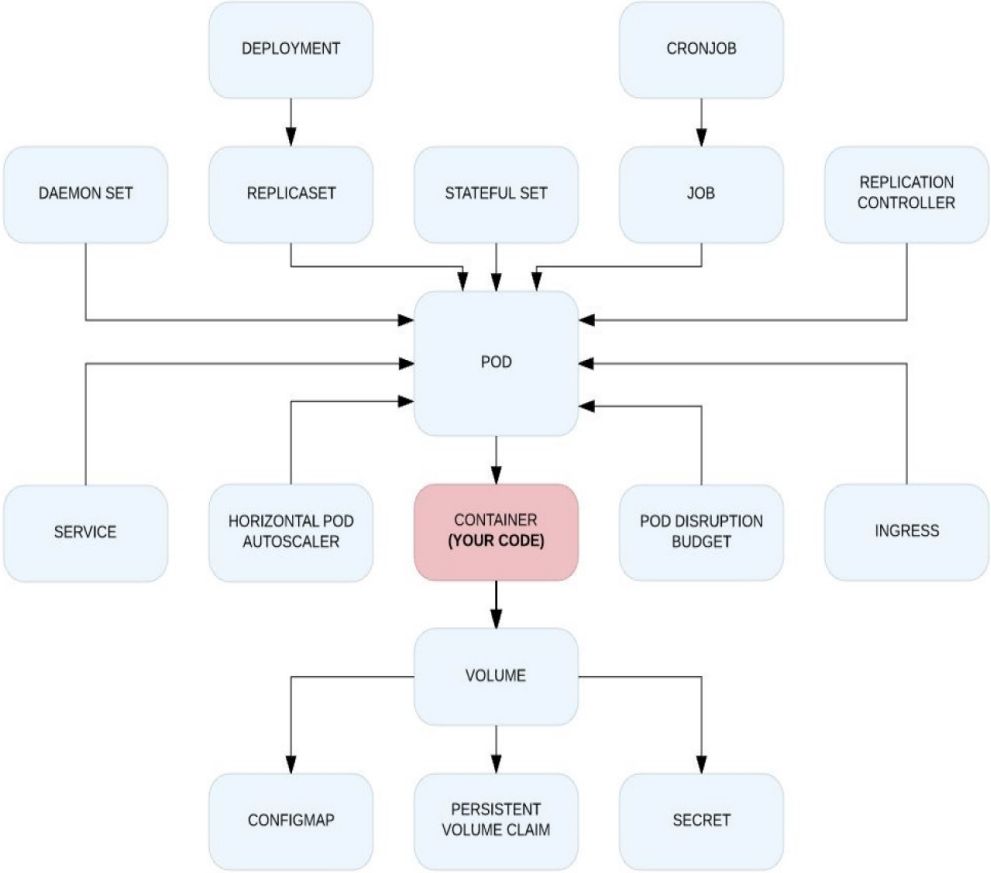
kubernetes - ingress

- The kubernetes cluster is firewalled from the internet
- Services and pods only have routable internal IPs
- Ingress is an object that manages external access to services (typically HTTP)
  - Load-balancing
  - SSL
  - Name-based virtual hosting
  - Path based redirections
    - /service/v1 => backend
    - /service/v2 => backend 2

kubernetes - volumes

- Lifetime equal to the pod (not the container)
- Pods can have many types of volumes simultaneously
  - Defined at pod level, mounted in containers
- Curently, 26 types of volumes are supported
  - AWS EBS, AzureFile, CEPHFS, Glusterfs, NFS, iSCSI, …
  - Host FS (hostpath)
    - */var/lib/docker*

kubernetes - objects/controllers



kubernetes - controllers

- Handle the lifetime of a set of pods, control placement and monitoring
  - Some similarities with Docker Swarm
- **Deployment** controller
  - Permits to update the description of Pods and ReplicaSets
    - Declarative way to deploy, imperative is: `kubectl rolling-update`
- **ReplicaSet** controller
  - Keep a number of copies of a pod/service across the cluster
  - Automatically replace failed copies
  - Can handle dynamic addition and removal of copies (elastic scaling)
- **DaemonSet** controller
  - Run a copy of a specific pod on each node of the cluster
- **Job** controller
  - Handle the execution of batch jobs

kubernetes - configMaps & secrets

- ENV vars are not enough for complex deployments
- Need for centralized configuration
  - ConfigMap
    - For non confidential data (e.g. ports, language, ..)
  - Secrets
    - For confidential data (e.g. API keys, …)
    - Currently just slightly obfuscated (base64)

Swarm VS kubernetes

ORCHESTRATOR FEATURE COMPARISON

Docker SWARM			
REST API	CLI	WebUI	Topology deployment orchestrator
"Management Node" Failover	"Compute Node" Failover	Container Replicas Failover	Cluster flapping prevention
Container Placement Management	Dynamic DNS Service	Container Auto-scaling	Load-balancing service
Multi-networks per container	App Networks	Distributed Storage Volume	Secret Management
Multi-tenancy and resource isolation	Tags selectors	Authentication Providers	ACL Management

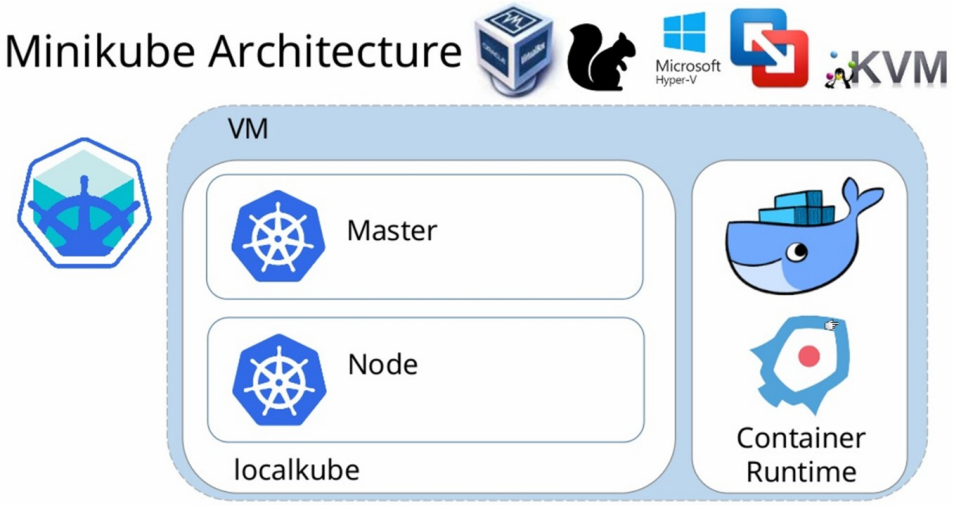
kubernetes by Google			
REST API	CLI	WebUI	Topology deployment orchestrator
"Management Node" Failover	"Compute Node" Failover	Container Replicas Failover	Cluster flapping prevention
Container Placement Management	Dynamic DNS Service	Container Auto-scaling	Load-balancing service
Multi-networks per container	App Networks	Distributed Storage Volume	Secret Management
Multi-tenancy and resource isolation	Tags selectors	Authentication Providers	ACL Management

kubernetes - conclusion

- Complex to setup
- Includes a lot of objects (we have only seen a few of them), still evolving
  - Intimidating at start
- Declarative architectures are very powerfull
- kubernetes is becoming the "standard" for container orchestration
  - CoreOS has replaced fleetd by kubernetes
  - All major clouds are providing kubernetes as a service

kubernetes - tutorial

- minikube
  - minikube launches a Master + Worker node inside a Virtualbox VM
  - https://kubernetes.io/docs/setup/minikube/



kubernetes - playgrounds

- Online playgrounds
  - https://www.katacoda.com/courses/kubernetes/playground
  - https://labs.play-with-k8s.com/tps://labs.play-with-k8s.com/