Intro to Kubernetes

WW Developer Advocacy Team

IBM Developer

But Wait? What About Production?

Automated scheduling and scaling

Zero downtime deployments

High availability and fault tolerance

A/B deployments

What is container orchestration?

Container orchestration

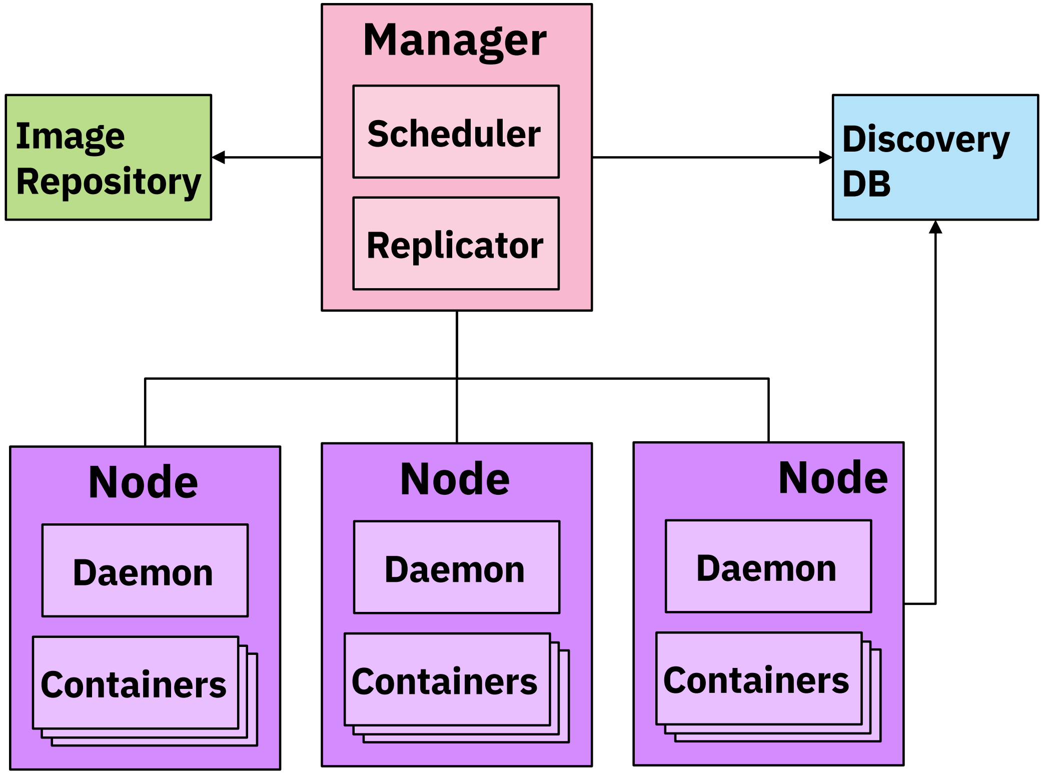
Cluster management

Scheduling

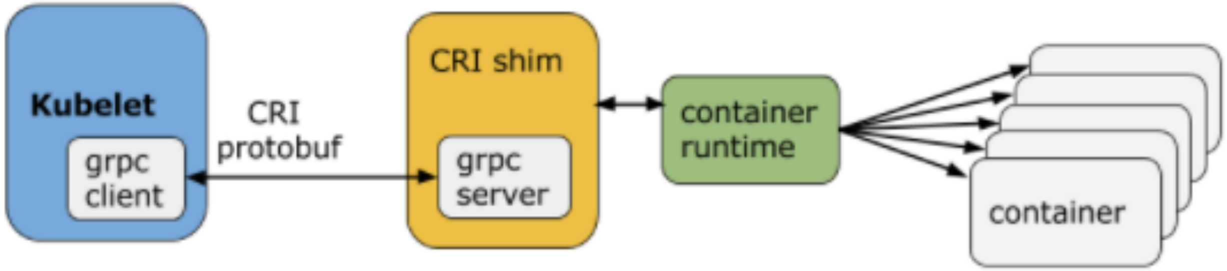
Service discovery

Replication

Health management



Container Runtime Interface



As at V1.18:

- Docker
- CRI-O
- Containerd
- frakti

What is Kubernetes?

Container Orchestrator

    - Provision, manage, scale applications

- Manage infrastructure resources needed by applications
    - Volumes
    - Networks
    - Secrets
    - And many many many more..

- What's in a name?
    - Kubernetes (K8s/Kube): "Helmsman" in ancient Greek

- Declarative model
    - Provide the "desired state" and Kubernetes will make it happen

How was Kubernetes created

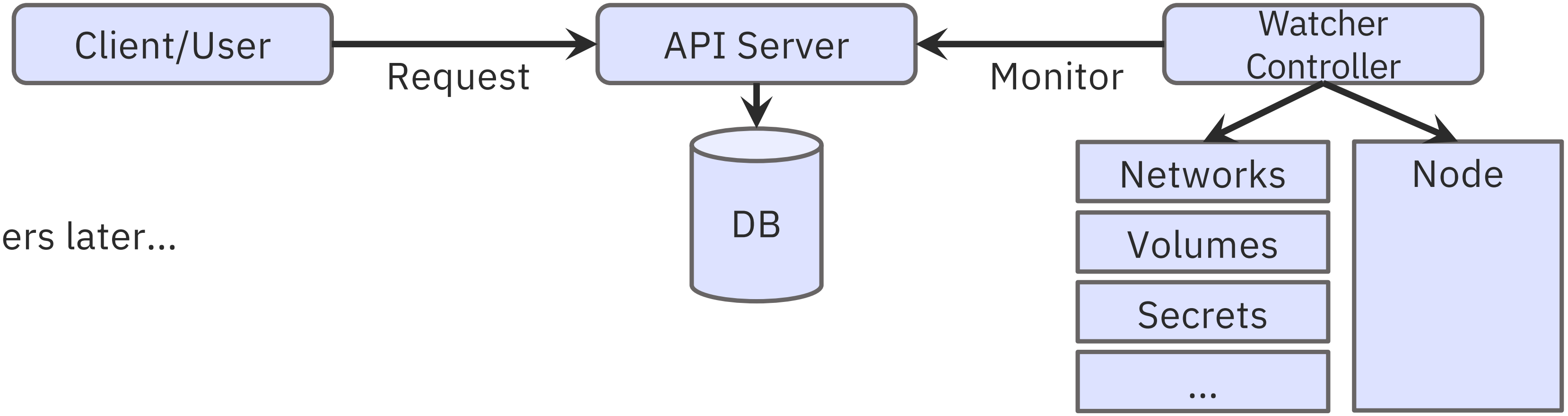
- Based on Google’s Borg & Omega
- Open Governance
  - Cloud Native Compute Foundation
- Adoption by Enterprise
  - RedHat, Microsoft, IBM and Amazon

Kubernetes Architecture

- At its core, Kubernetes is a database (etcd).
  With "watchers" & "controllers" that react to changes in the DB.
  The controllers are what make it Kubernetes.
  This pluggability and extensibility is part of its "secret sauce".

- DB represents the user's desired state
  - Watchers attempt to make reality match the desired state

"API Server" is the HTTP/REST
front-end to the DB



More on controllers later...

Kubernetes Resource Model

A resource for every purpose

- Config Maps
- Daemon Sets
- **Deployments**
- Events
- Endpoints
- Ingress
- Jobs
- Nodes
- Namespaces
- **Pods**
- Persistent Volumes
- Replica Sets
- Secrets
- Service Accounts
- **Services**
- Stateful Sets, and more…
- Kubernetes aims to have the building blocks on which you build a cloud native platform.

- Therefore, the internal resource model **is** the same as the end user resource model.

Key Resources

- Pod: set of co-located containers
  - Smallest unit of deployment
  - Several types of resources to help manage them
  - Replica Sets, Deployments, Stateful Sets, …

- Services
  - Define how to expose your app as a DNS entry
  - Query based selector to choose which pods apply

Kubernetes Client

CLI tool to interact with Kubernetes cluster

Platform specific binary available to download

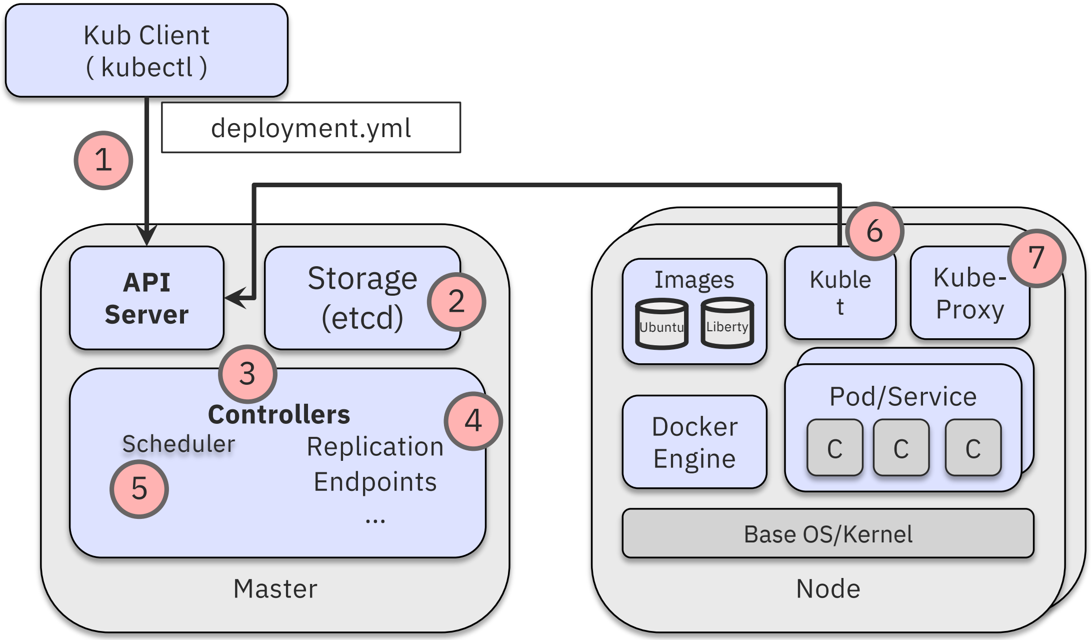
– https://kubernetes.io/docs/tasks/tools/install-kubectl

The user directly manipulates resources via json/yaml

```
$ kubectl (create|get|apply|delete) -f myResource.yaml
```

Kubernetes in Action!

1. User via "kubectl" deploys a new application
2. API server receives the request and stores it in the DB (etcd)
3. Watchers/controllers detect the resource changes and act upon it
4. ReplicaSet watcher/controller detects the new app and creates new pods to match the desired # of instances
5. Scheduler assigns new pods to a kubelet
6. Kubelet detects pods and deploys them via the container runtime (e.g. Docker)
7. Kubeproxy manages network traffic for the pods – including service discovery and load-balancing



IBM Developer

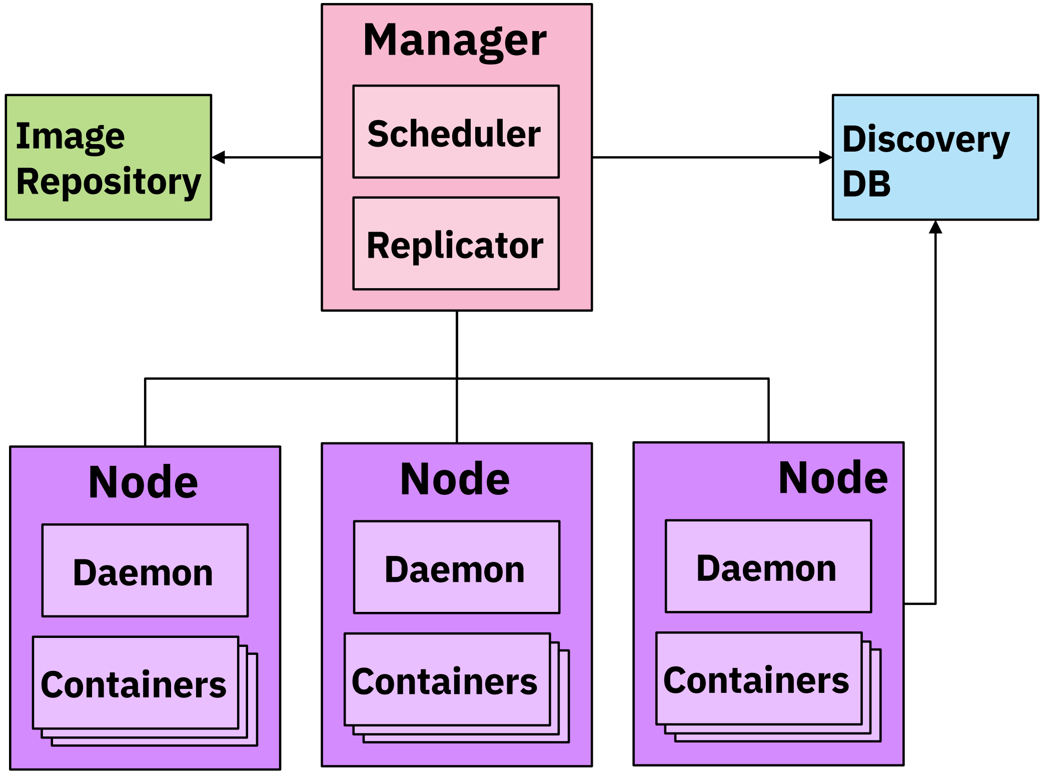
Benefits of Container Orchestration

Automated scheduling and scaling

Zero downtime deployments

High availability and fault tolerance

A/B deployments



But Wait? What About Production?

Kubernetes by itself is not enterprise-ready

Kubernetes must integrate with underlying platform to provide infrastructure, storage, etc.

Lacking in operational view + controls, pre-built catalogs

Kubernetes @ IBM

Offerings / Plans

- IKS - IBM Cloud Kubernetes Service
- ICP - IBM Cloud Private
- IBM Cloud service teams use Kubernetes to host

Key Development Activities

- Service Catalog (co-lead)
- Contributor Experience
- Networking & Istio (co-lead)
- ContainerD integration (co-lead)
- Storage
- Performance