Lesson iut.univ-paris8.fr 2018-05

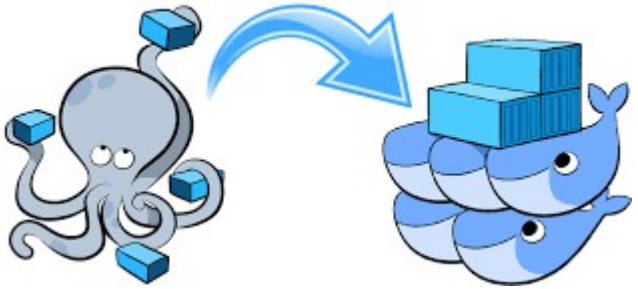
Introduction to Kubernetes
(Cluster Containers Orchestration)

This Document:
http://arnaud-nauwynck.github.io/docs/Intro-Kubernetes.pdf

arnaud.nauwynck@gmail.com

Comparison, Alternatives

Docker Compose & Docker Swarm



MESOS

MARATHON

Google

Google Stack Foundation: Borg → Omega (~ Kubernetes)

CamSaS

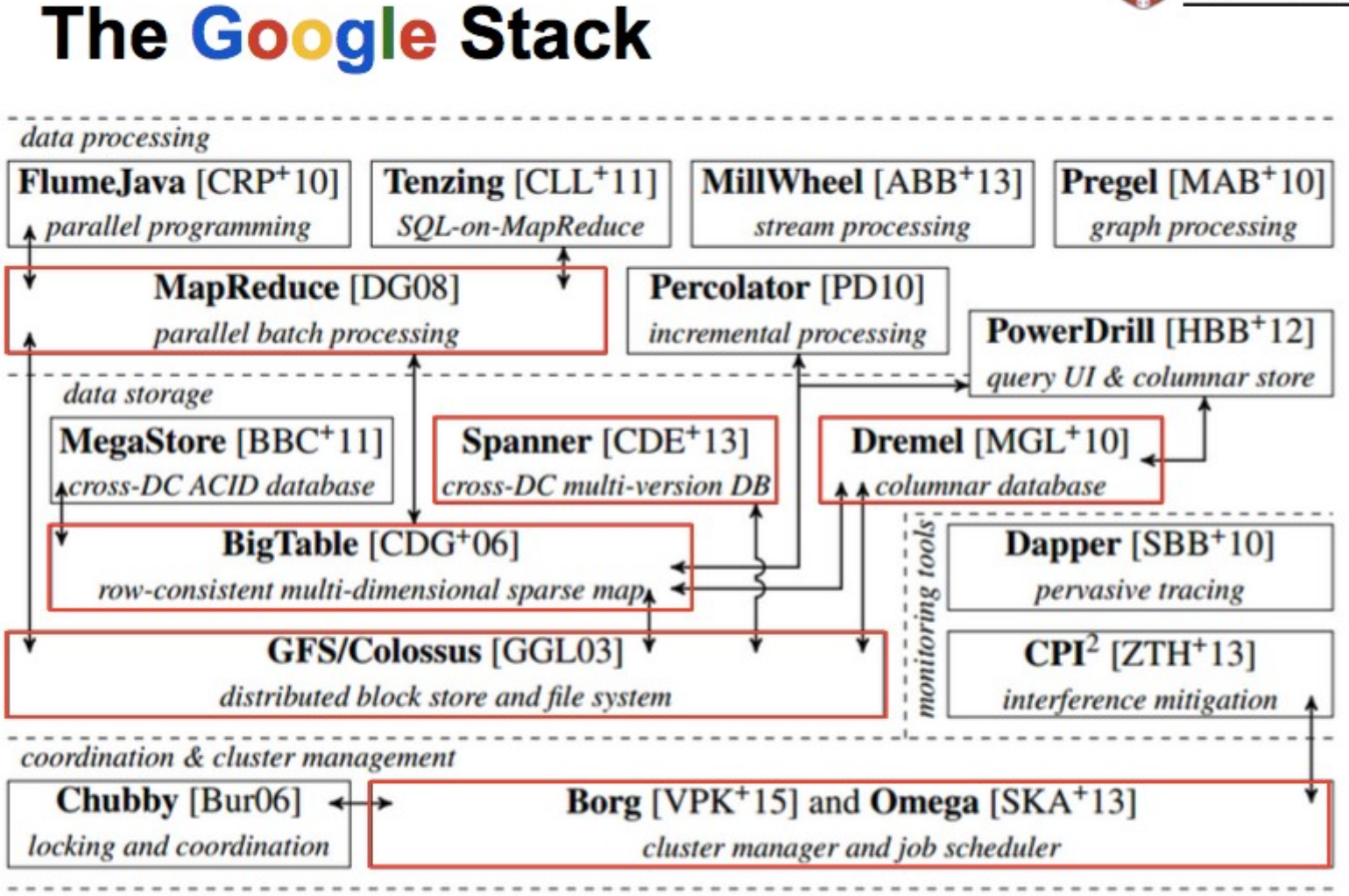
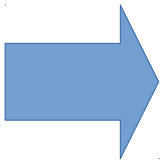


Figure from M. Schwarzkopf, "Operating system support for warehouse-scale computing", PhD thesis, University of Cambridge, 2015 (to appear).

Vision : DataCenter as a Computer

Programming Model Analogy

1 Process → N Threads		1 Service → N Containers
Schedule on CPU Cores (on GPU Processing Units)		Schedule on Nodes
Inter-Threads communications		Inter Containers Networks
Low Level Thread API High Level Concurrent Library		Low Level Docker API High Level Cloud-Native Library

https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events

Fallacies of distributed computing

From Wikipedia, the free encyclopedia

The **fallacies of distributed computing** are a set of assertions made by L Peter Deutsch and others at Sun Microsystems describing false assumptions that programmers new to distributed applications invariably make.

The fallacies [edit]

The fallacies are:[1]

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. Topology doesn't change.
6. There is one administrator.
7. Transport cost is zero.
8. The network is homogeneous.

Expect Chaos ..

Disk → Crashes

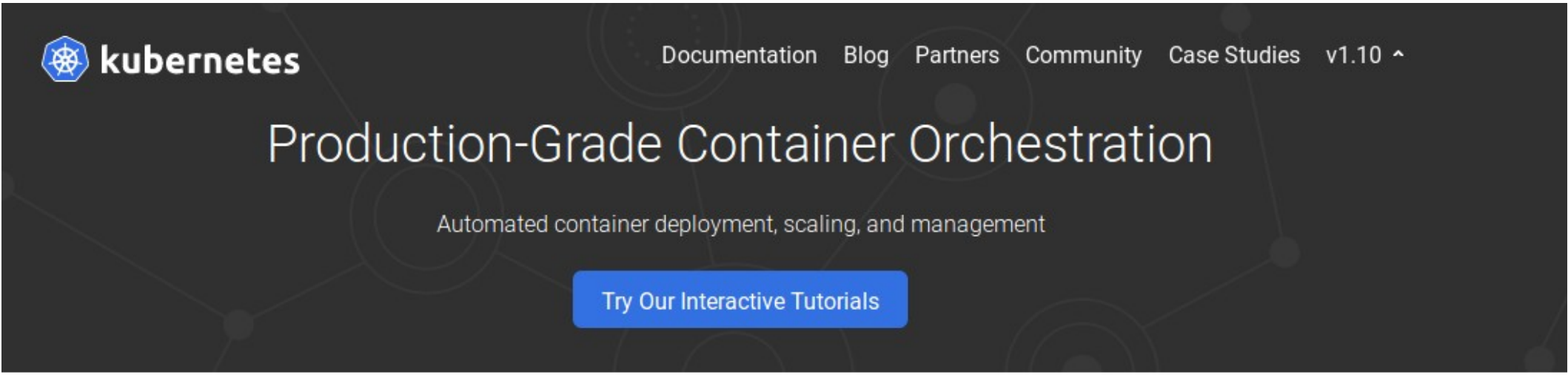
Electricity → Shutdown

CPU → Burn

Network → unplug / noise

Design for Resiliency .. Test

https://kubernetes.io/



Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

Kubernetes Name & Logo ...

What does *Kubernetes* mean? K8s?

The name **Kubernetes** originates from Greek, meaning *helmsman* or *pilot*, and is the root of *governor* and cybernetic. *K8s* is an abbreviation derived by replacing the 8 letters "ubernete" with "8".

https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Features

Automatic binpacking

Automatically places containers based on their resource requirements and other constraints, while not sacrificing availability. Mix critical and best-effort workloads in order to drive up utilization and save even more resources.

Horizontal scaling

Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.

Automated rollouts and rollbacks

Kubernetes progressively rolls out changes to your application or its configuration, while monitoring application health to ensure it doesn't kill all your instances at the same time. If something goes wrong, Kubernetes will rollback the change for you. Take advantage of a growing ecosystem of deployment solutions.

Storage orchestration

Automatically mount the storage system of your choice, whether from local storage, a public cloud provider such as GCP or AWS, or a network storage system such as NFS, iSCSI, Gluster, Ceph, Cinder, or Flocker.

Self-healing

Restarts containers that fail, replaces and reschedules containers when nodes die, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.

Service discovery and load balancing

No need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives containers their own IP addresses and a single DNS name for a set of containers, and can load-balance across them.

Secret and configuration management

Deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.

Batch execution

In addition to services, Kubernetes can manage your batch and CI workloads, replacing containers that fail, if desired.

https://kubernetes.io/docs/home/

Setup

01 - Downloading Kubernetes
02 - Independent Solutions
03 - Hosted Solutions
04 - Turn-key Cloud Solutions
05 - Custom Solutions
06 - User Journeys
07 - Installing Addons
08 - Configuring Kubernetes with Salt
09 - Building Large Clusters
10 - Running in Multiple Zones
11 - Building High-Availability Clusters

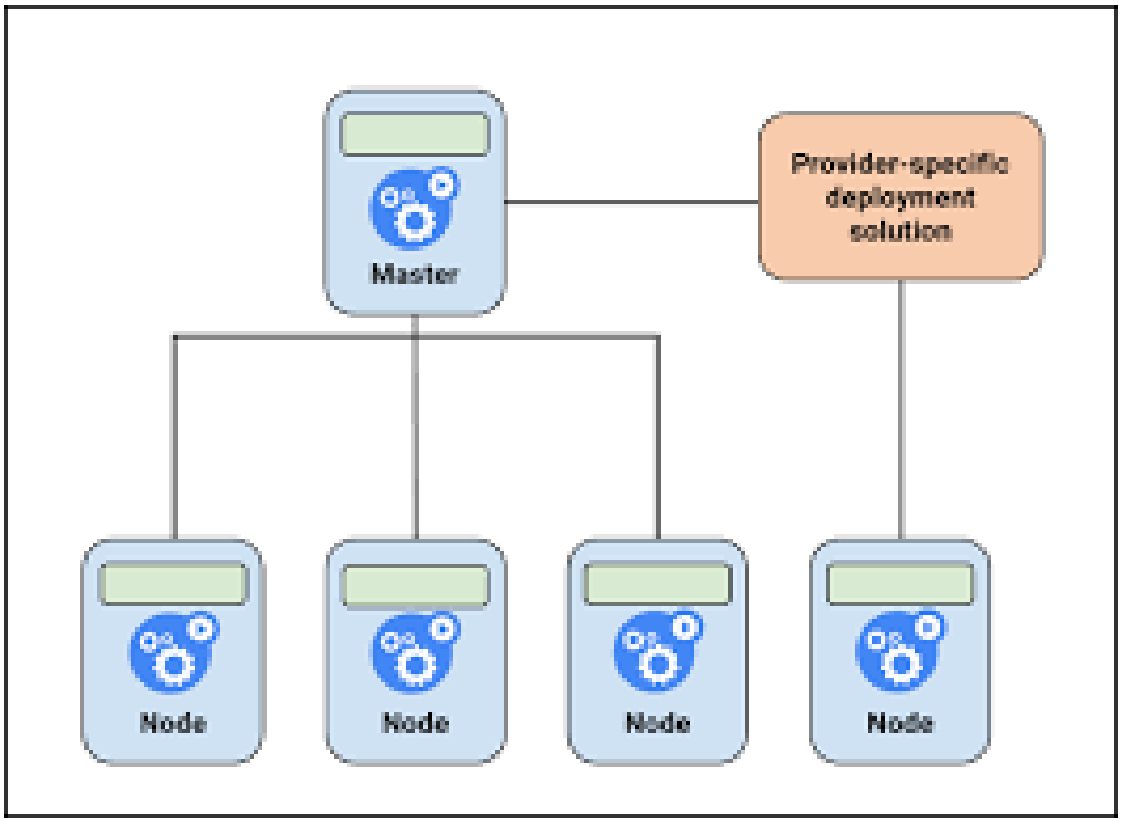
Concepts

01 - Overview
02 - Kubernetes Architecture
03 - Extending Kubernetes
04 - Containers
05 - Workloads
06 - Configuration
07 - Services, Load Balancing, and Networking
08 - Storage
09 - Cluster Administration

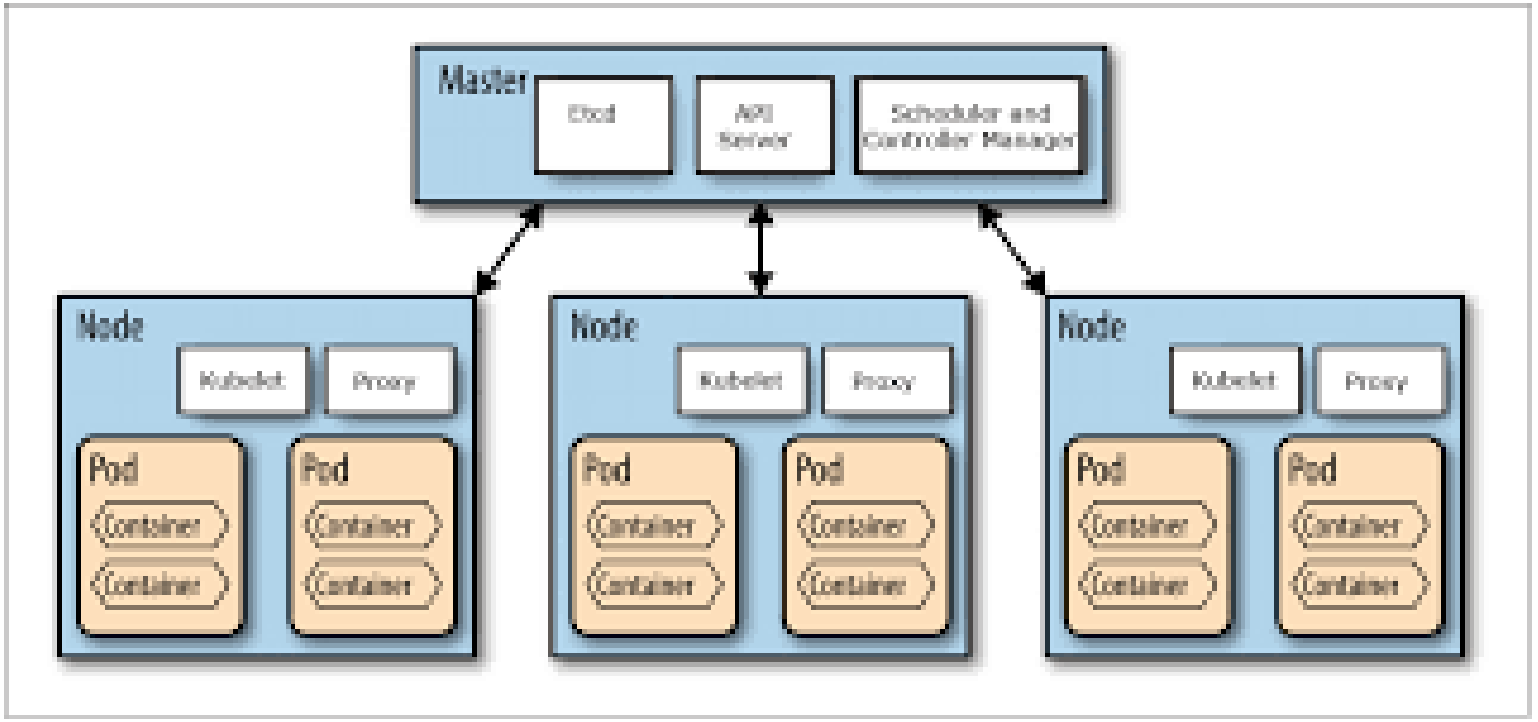
Tasks

01 - Install Tools
02 - Configure Pods and Containers
03 - Inject Data Into Applications
04 - Run Applications
05 - Run Jobs
06 - Access Applications in a Cluster
07 - Monitor, Log, and Debug
08 - Extend Kubernetes
09 - TLS
10 - Administer a Cluster
11 - Federation - Run an App on Multiple Clusters
12 - Manage Cluster Daemons
13 - Manage GPUs
14 - Manage HugePages
15 - Extend kubectl with plugins
16 - Troubleshooting

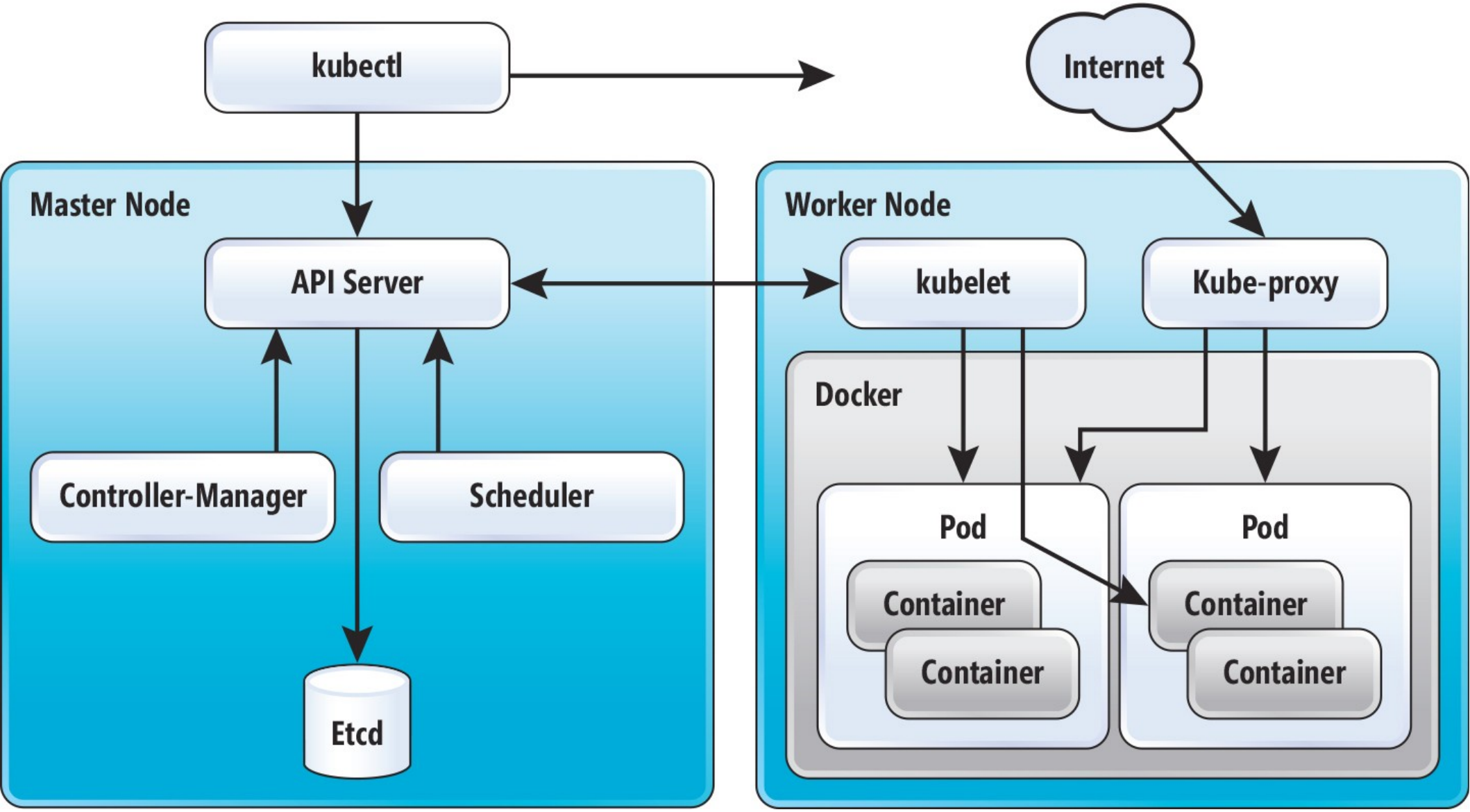
Master - Nodes



Master – Node – Pod - Container



Master Detail : kubectl,API,..
Node Detail :Kubelet, Kube-Proxy,..



Kubernetes = Docker + ...

Docker Already have (Toy) Docker Swarm
(Docker Swarm = same API as Docker… for Cluster)

So, what's more in Kubernetes compared to Docker ?

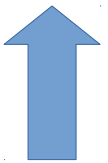
… different API ! New Higher-Level Concepts

Docker → Kubernetes
Imperatif → Declaratif

$ docker run image …
$ docker start/stop/rm/ ….

One-shot action execution
Command = verb to execute

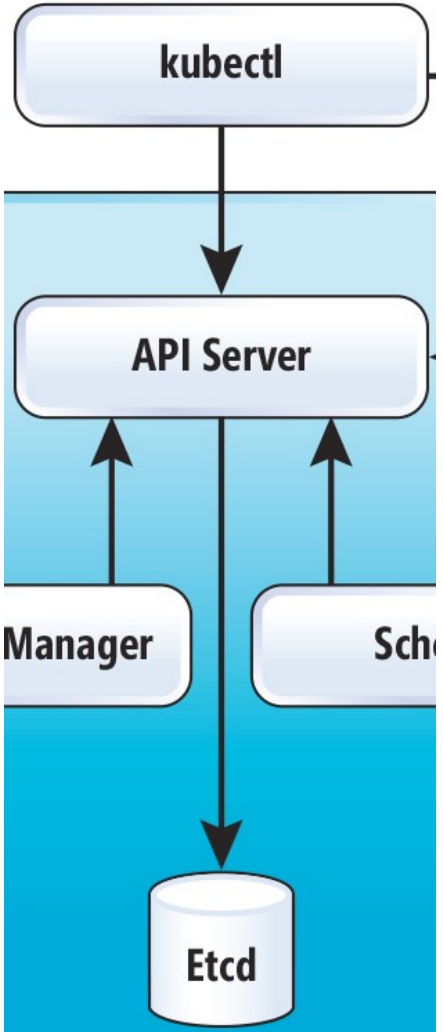
$ kubelet rs create ...
$ kubelet rs delete ...



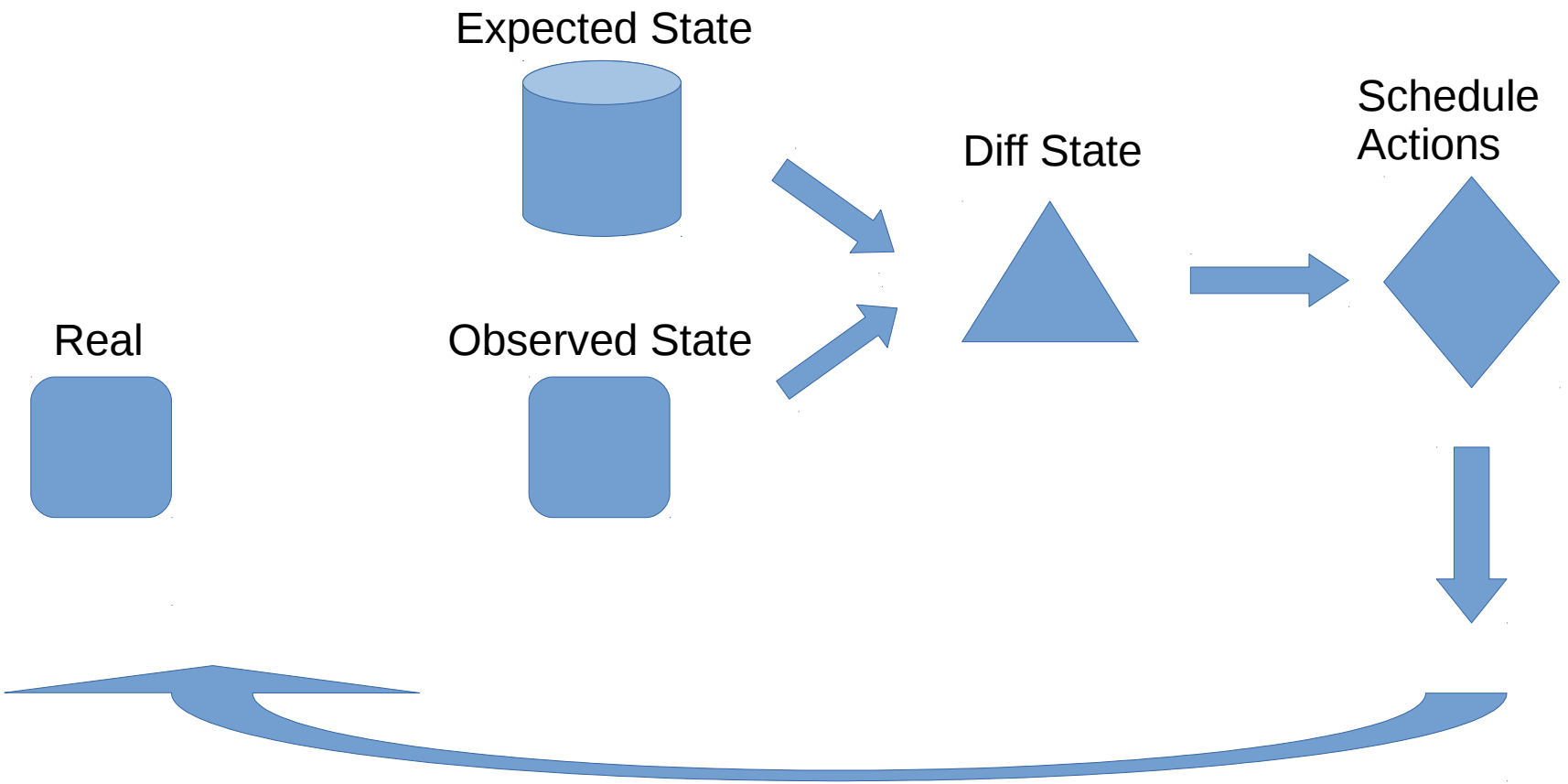
Manage persistent resource
Command = REST

Behind → Controller loop
to reconcile(start/stop/..)

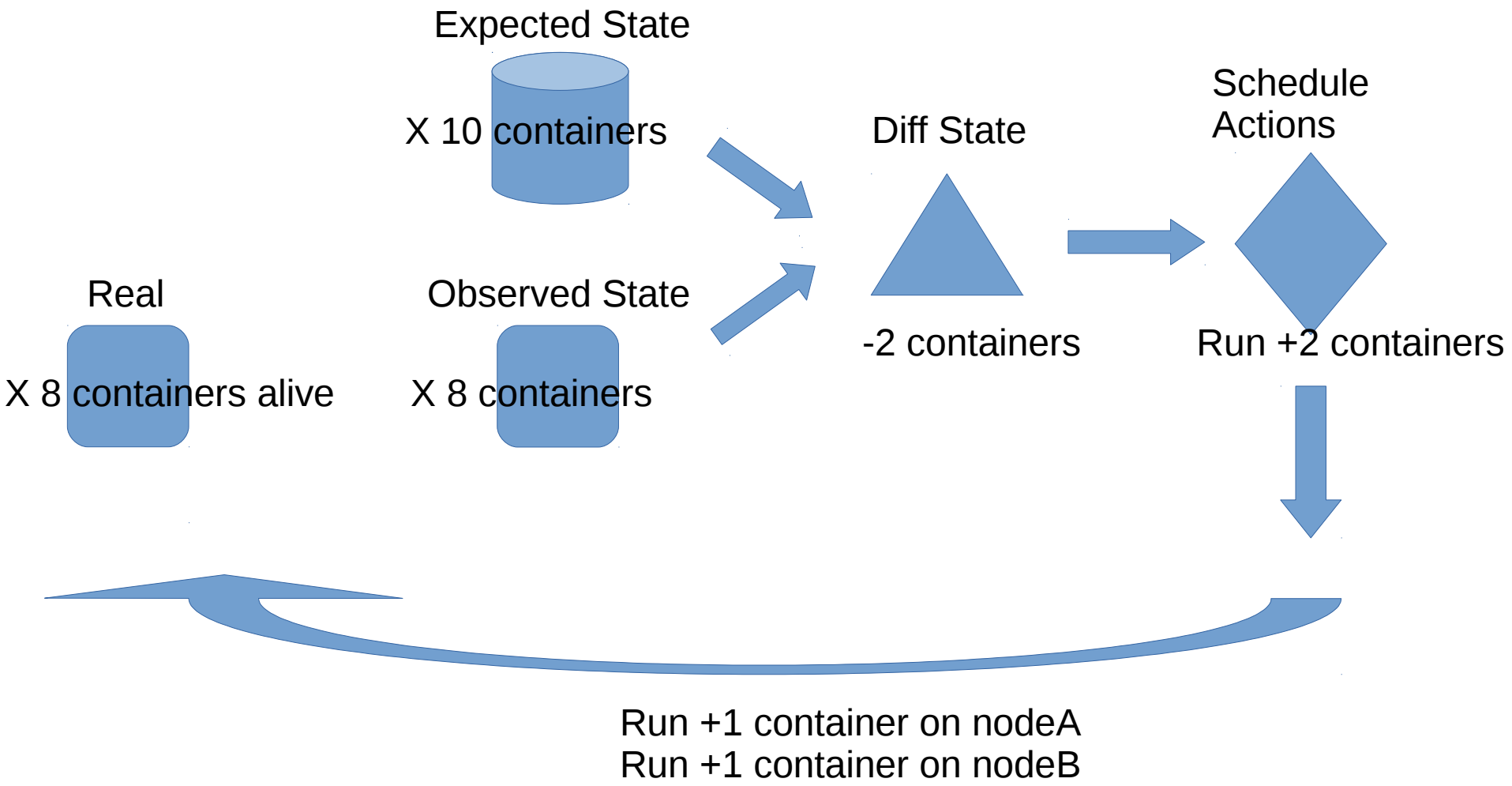
REST – Persistent Resources



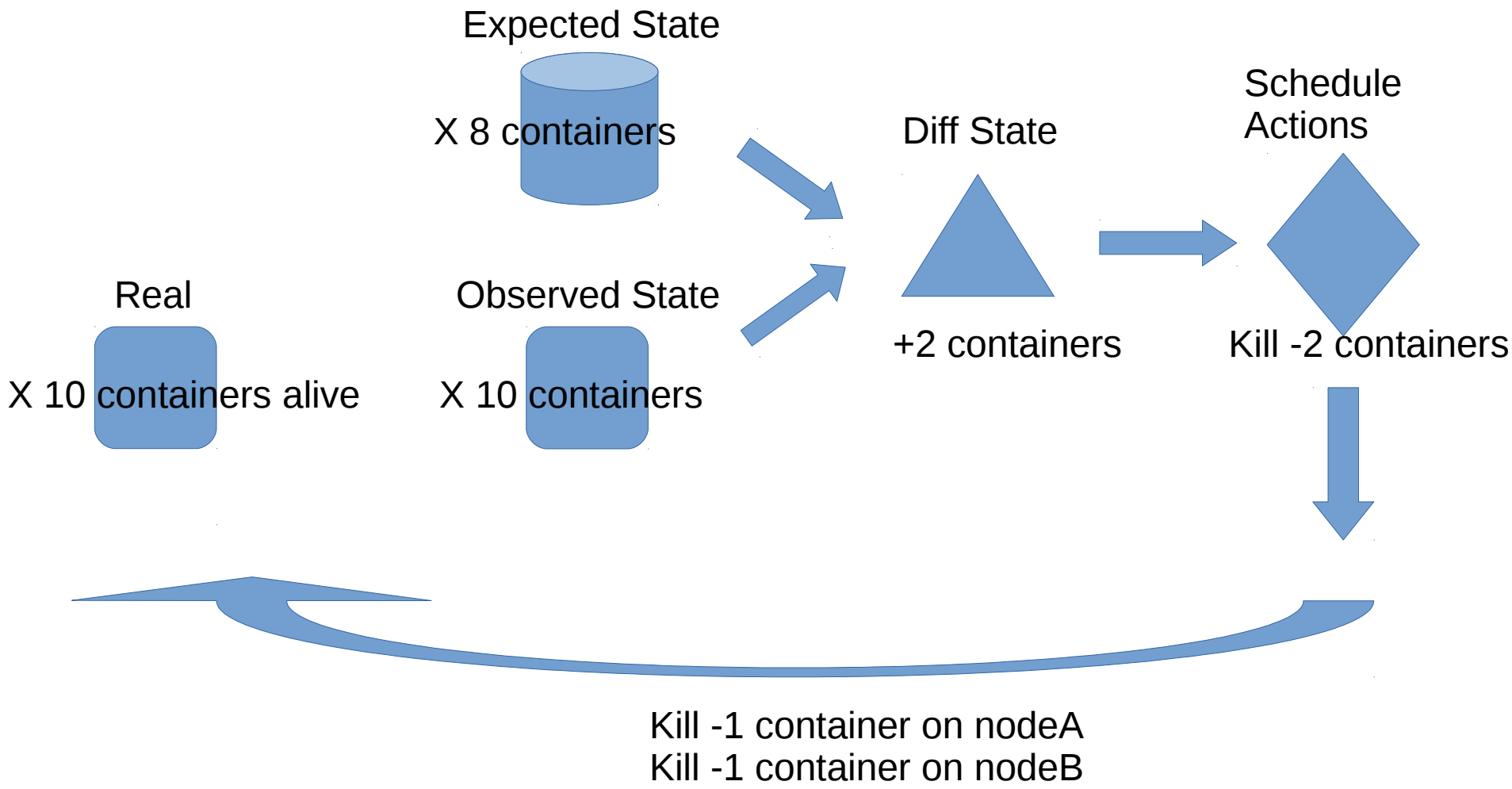
Controller = Reconcile
diff : Expected - Observed



Example: Scale UP => schedule run



Example: Scale DOWN => schedule rm



Kubernetes Objects

Kubernetes contains a number of abstractions that represent the state of your system: deployed containerized applications and workloads, their associated network and disk resources, and other information about what your cluster is doing. These abstractions are represented by objects in the Kubernetes API; see the Kubernetes Objects overview for more details.

The basic Kubernetes objects include:

- Pod
- Service
- Volume
- Namespace

In addition, Kubernetes contains a number of higher-level abstractions called Controllers. Controllers build upon the basic objects, and provide additional functionality and convenience features. They include:

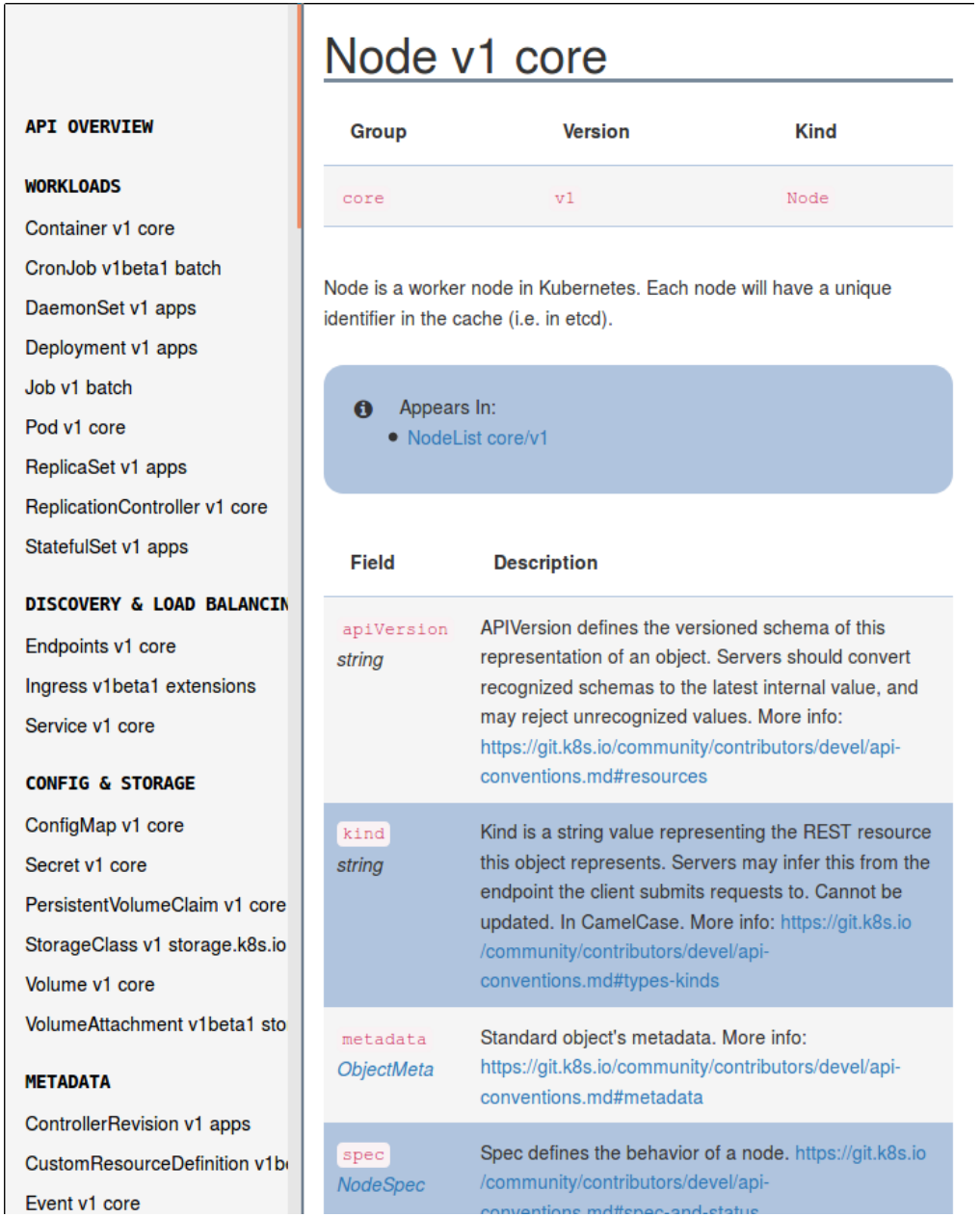
- ReplicaSet
- Deployment
- StatefulSet
- DaemonSet
- Job

Nodes

What is a node?

A `node` is a worker machine in Kubernetes, previously known as a `minion`. A node may be a VM or physical machine, depending on the cluster. Each node has the services necessary to run pods and is managed by the master components. The services on a node include Docker, kubelet and kube-proxy. S The Kubernetes Node section in the architecture design doc for more details.

Node API



API OVERVIEW

WORKLOADS

Container v1 core
CronJob v1beta1 batch
DaemonSet v1 apps
Deployment v1 apps
Job v1 batch
Pod v1 core
ReplicaSet v1 apps
ReplicationController v1 core
StatefulSet v1 apps

DISCOVERY & LOAD BALANCIN

Endpoints v1 core
Ingress v1beta1 extensions
Service v1 core

CONFIG & STORAGE

ConfigMap v1 core
Secret v1 core
PersistentVolumeClaim v1 core
StorageClass v1 storage.k8s.io
Volume v1 core
VolumeAttachment v1beta1 sto

METADATA

ControllerRevision v1 apps
CustomResourceDefinition v1b
Event v1 core

Node v1 core

Group	Version	Kind
core	v1	Node

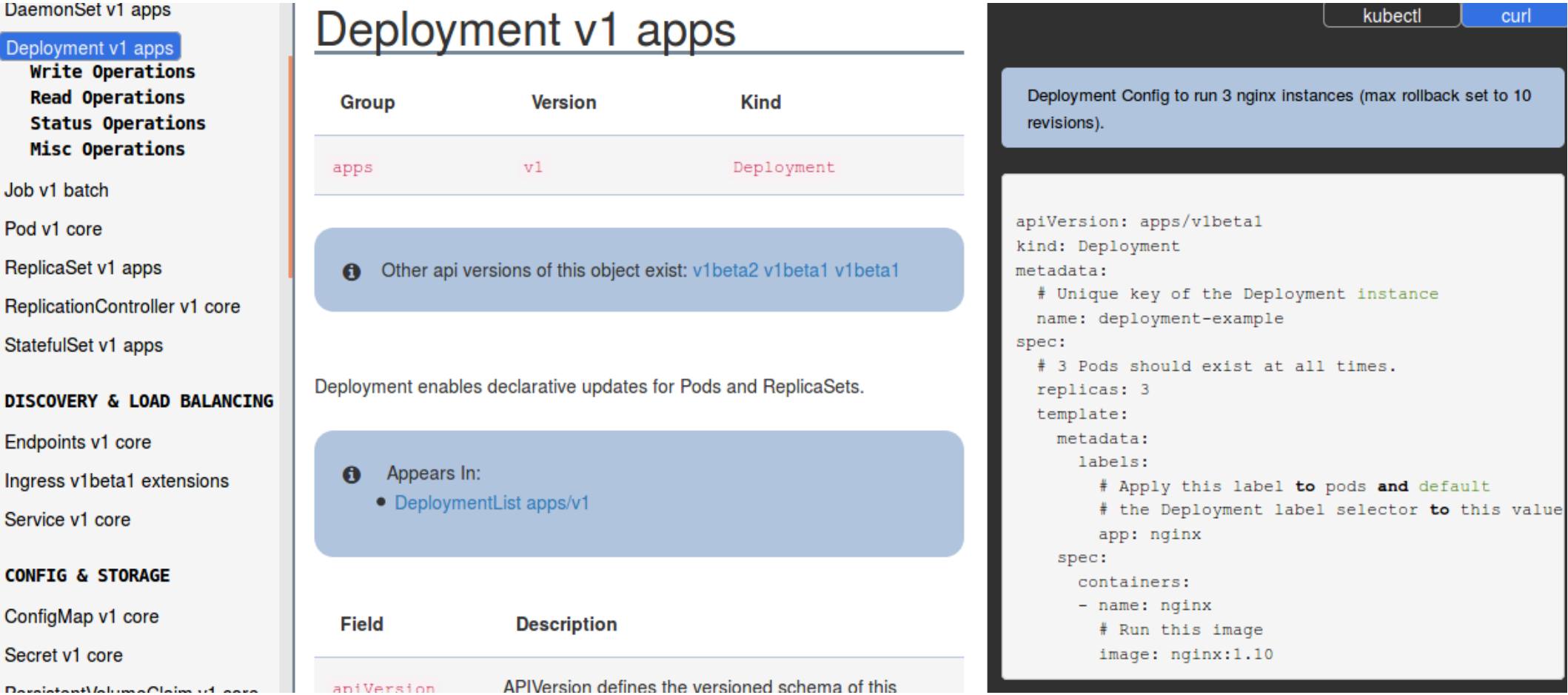
Node is a worker node in Kubernetes. Each node will have a unique identifier in the cache (i.e. in etcd).

Appears In:
- NodeList core/v1

Field	Description
apiVersion *string*	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/api-conventions.md#resources
kind *string*	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/api-conventions.md#types-kinds
metadata *ObjectMeta*	Standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/api-conventions.md#metadata
spec *NodeSpec*	Spec defines the behavior of a node. https://git.k8s.io/community/contributors/devel/api-

https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.10/#node-v1-core

Kubernetes API … Rest Json/Yaml



Http Rest Yaml → Json ...

`curl` Command (*requires* `kubectl proxy` *to be running*)

```
$ kubectl proxy
$ curl -X POST -H 'Content-Type: application/yaml' --
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: deployment-example
spec:
  replicas: 3
  revisionHistoryLimit: 10
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.10
        ports:
        - containerPort: 80
' http://127.0.0.1:8001/apis/apps/v1/namespaces/defau
```

$ curl -X POST .
or
$ kubelet ..

Response Body


```
{
  "kind": "Deployment",
  "apiVersion": "apps/v1beta1",
  "metadata": {
    "name": "deployment-example",
    "namespace": "default",
    "selfLink": "/apis/apps/v1beta1/namespaces/defaul
    "uid": "4ccca349-9cb1-11e6-9c54-42010a800148",
    "resourceVersion": "2118306",
    "generation": 1,
    "creationTimestamp": "2016-10-28T01:53:19Z",
    "labels": {
      "app": "nginx"
    }
  },
  "spec": {
    "replicas": 3,
    "selector": {
      "matchLabels": {
        "app": "nginx"
      }
    },
    "template": {
      "metadata": {
        "creationTimestamp": null,
        "labels": {
          "app": "nginx"
```

Kubectl Client

kubectl

kubectl controls the Kubernetes cluster manager

Synopsis

kubectl controls the Kubernetes cluster manager.

Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/

```
kubectl [flags]
```

Options

```
      --alsologtostderr                  log to standard error a
      --as string                        Username to impersonate
      --as-group stringArray             Group to impersonate fo
      --cache-dir string                 Default HTTP cache dire
      --certificate-authority string     Path to a cert file fo
      --client-certificate string        Path to a client certi
```

Kubetcl commands

GETTING STARTED

run

run-container

expose

APP MANAGEMENT

annotate

autoscale

convert

create

delete

edit

get

label

patch

replace

rolling-update

rollout

scale

set

DECLARATIVE APP MANAGEMENT

apply

WORKING WITH APPS

attach

auth

cp

describe

exec

logs

port-forward

proxy

top

CLUSTER MANAGEMENT

api-versions

certificate

cluster-info

cordon

drain

taint

uncordon

KUBECTL SETTINGS AND USAGE

alpha

completion

config

explain

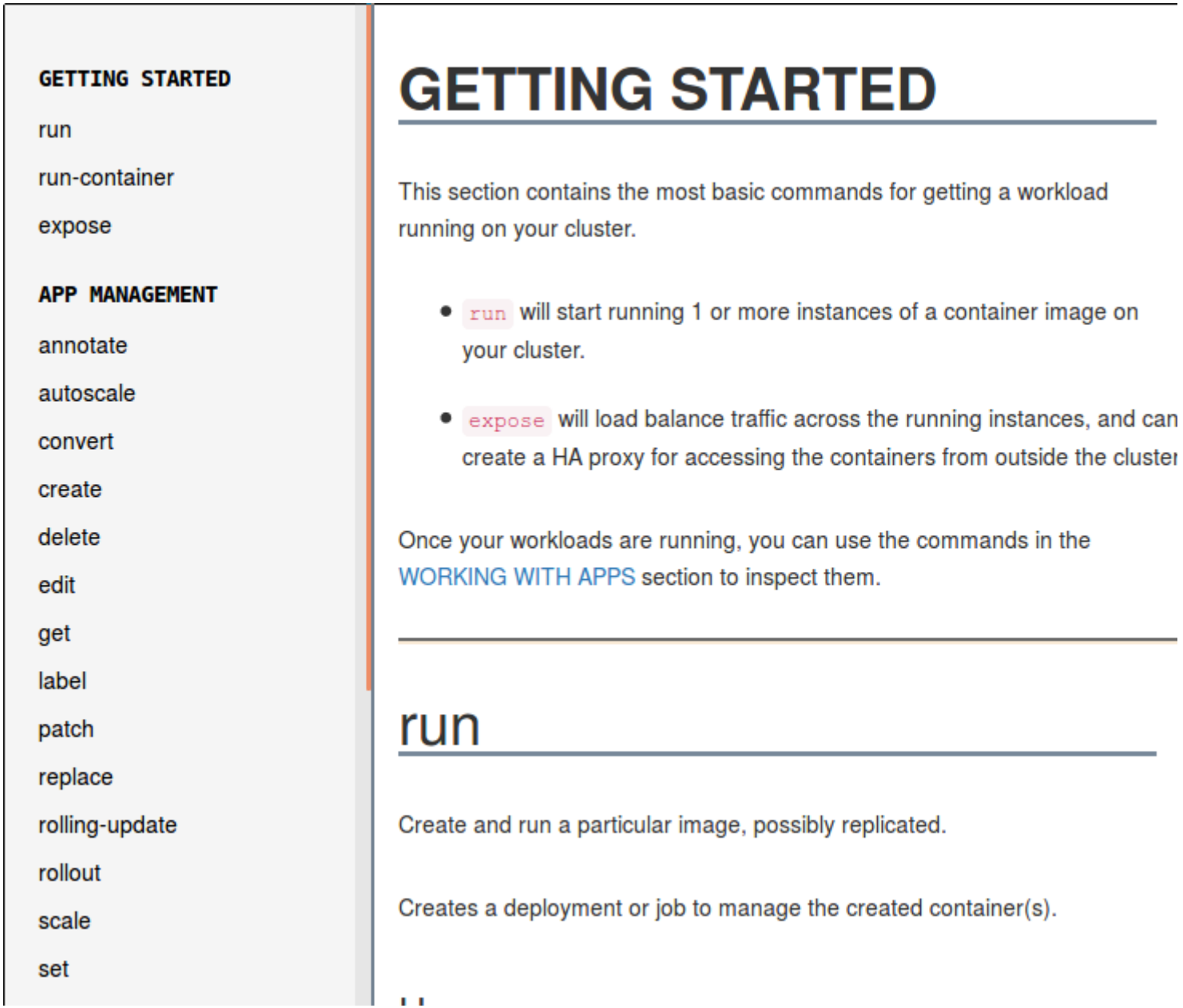
options

version

plugin

DEPRECATED COMMANDS

Kubectl commands details



GETTING STARTED

run

run-container

expose

APP MANAGEMENT

annotate

autoscale

convert

create

delete

edit

get

label

patch

replace

rolling-update

rollout

scale

set

GETTING STARTED

This section contains the most basic commands for getting a workload running on your cluster.

- `run` will start running 1 or more instances of a container image on your cluster.
- `expose` will load balance traffic across the running instances, and can create a HA proxy for accessing the containers from outside the cluster

Once your workloads are running, you can use the commands in the WORKING WITH APPS section to inspect them.

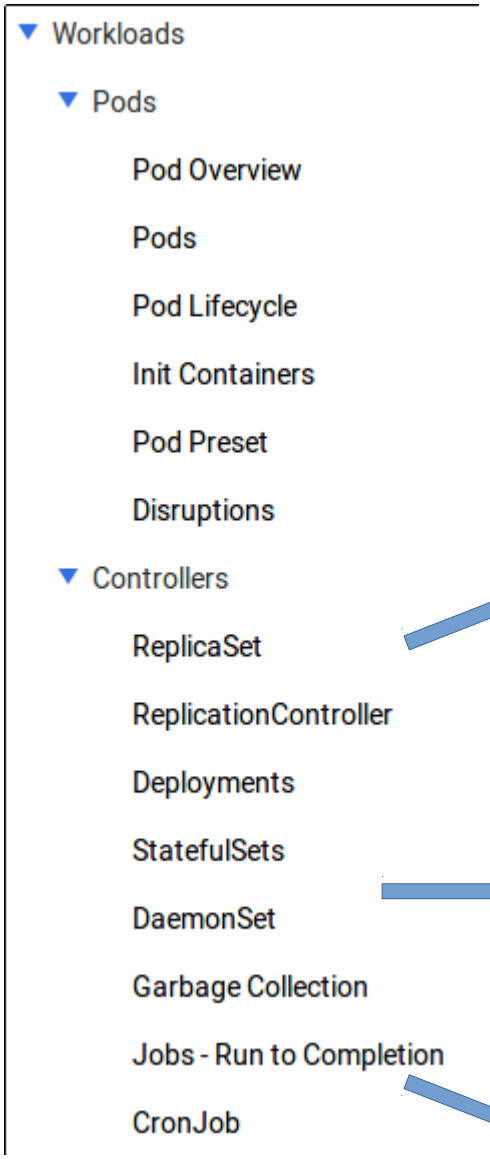
run

Create and run a particular image, possibly replicated.

Creates a deployment or job to manage the created container(s).

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands

Kubernetes Controllers: ReplicationSet, ..



scale=N times … image “schedule” on nodes

1 container per node (example: cAdvisor monitoring)

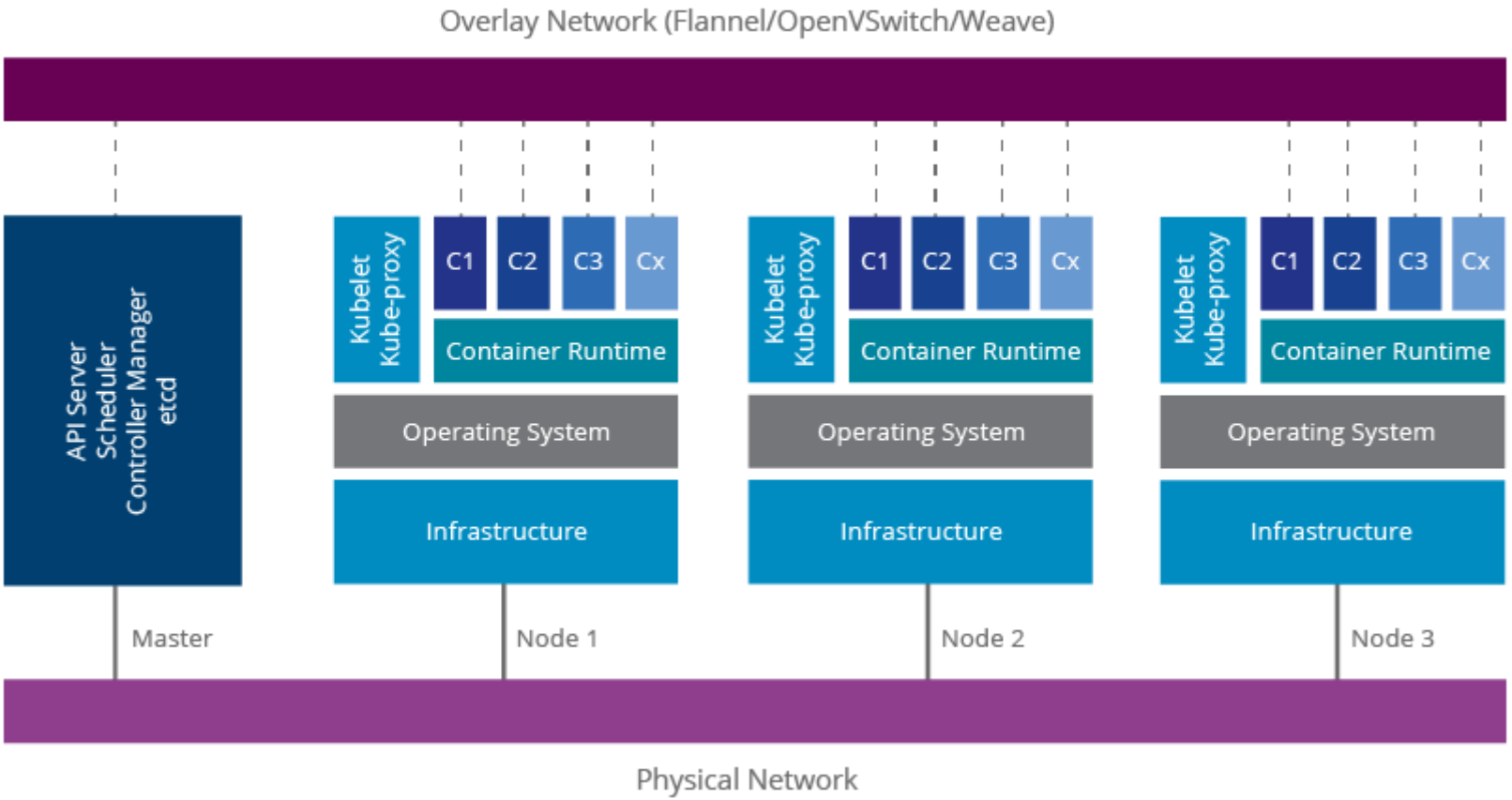
Launch Job (short tasks, batches)

Networks...

Port Forwarding

Level 7 Reverse Proxy, Ingress

Overlay Network



Volumes

Summary

Part 1: VM & Containers

Part 2 : Docker

Part 3: Kubernetes

(Cluster Orchestration)