redhat.

Containers orchestration with Kubernetes and OpenShift

Paolo Patierno
Principal Software Engineer @ Red Hat
28/03/2018

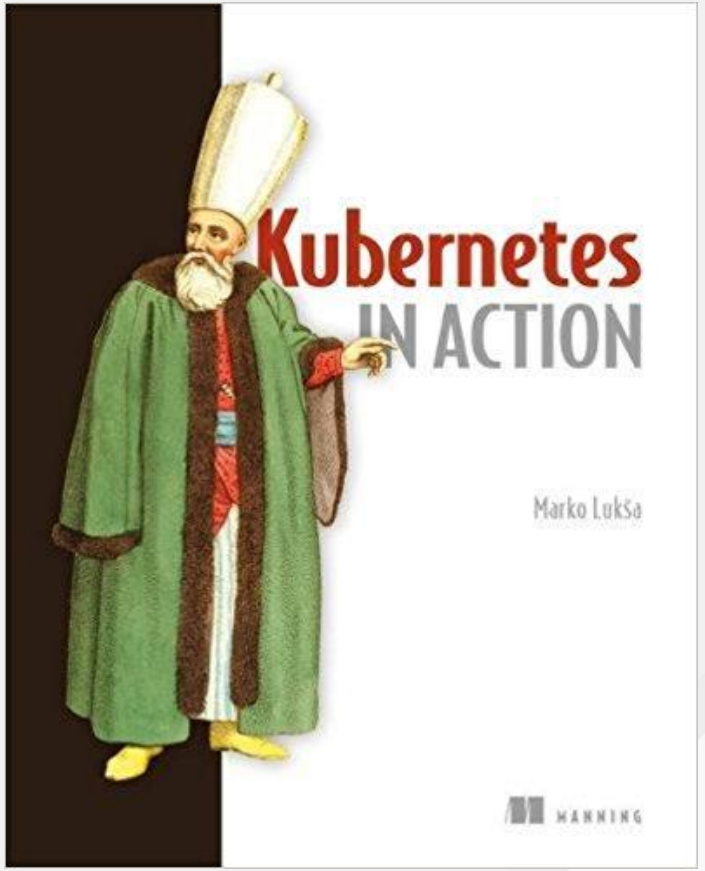
Who am I ?

@ppatierno

- Principal Software Engineer @ Red Hat
  - Messaging & IoT team
- Lead/Committer @ Eclipse Foundation
  - Hono, Paho and Vert.x projects
- Microsoft MVP Azure/IoT
- Hacking low constrained devices in spare time (from previous life)
- Valentino Rossi's fan !
- Try to be a runner ...

Kubernetes in Action

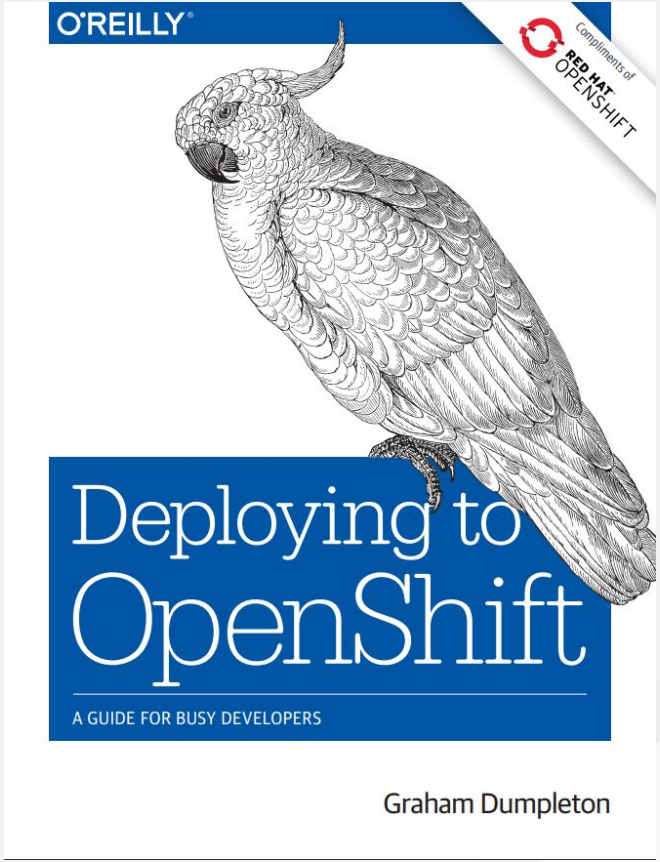
- Marko Luksa (I know this guy ! ;))
- To become a Kubernetes guru
- Discount code for 40 % : **ctwluksa18**



Deploying to OpenShift

A guide for busy developers

- Graham Dumpleton
- It’s free
- https://www.openshift.com/promotions/deploying-to-openshift.html



What is Kubernetes ?

- A system for ...
  - ... **automating deployment** ...
  - ... **scaling** ...
  - ... **management** ...
- ... of **containerized applications**
- Comes from Google experience with project “Borg”
- **Open source**
- Written in **Go**
- It’s name comes from Greek for “helmsman”

What is Kubernetes ?

- It’s like the **Linux kernel** …
- … but for a **distributed system**
- Abstract the underlying hardware in terms of “nodes”
- On the nodes a set of different “resources” can be deployed and handled
- Containerized applications are deployed, using and sharing “resources”

What is Kubernetes ?

Main features

- **Automatic binpacking**
  - Schedules containers according to their requirements in terms of resources
- **Self healing**
  - Restarts containers that fail, reschedules containers when nodes die
  - Kills containers that don’t respond (to health checks)
- **Horizontal scaling**
  - Scale applications up and down
- **Service discovery and load balancing**
  - Applications can be discovered through DNS names
  - Load balance across set of containers

What is Kubernetes ?

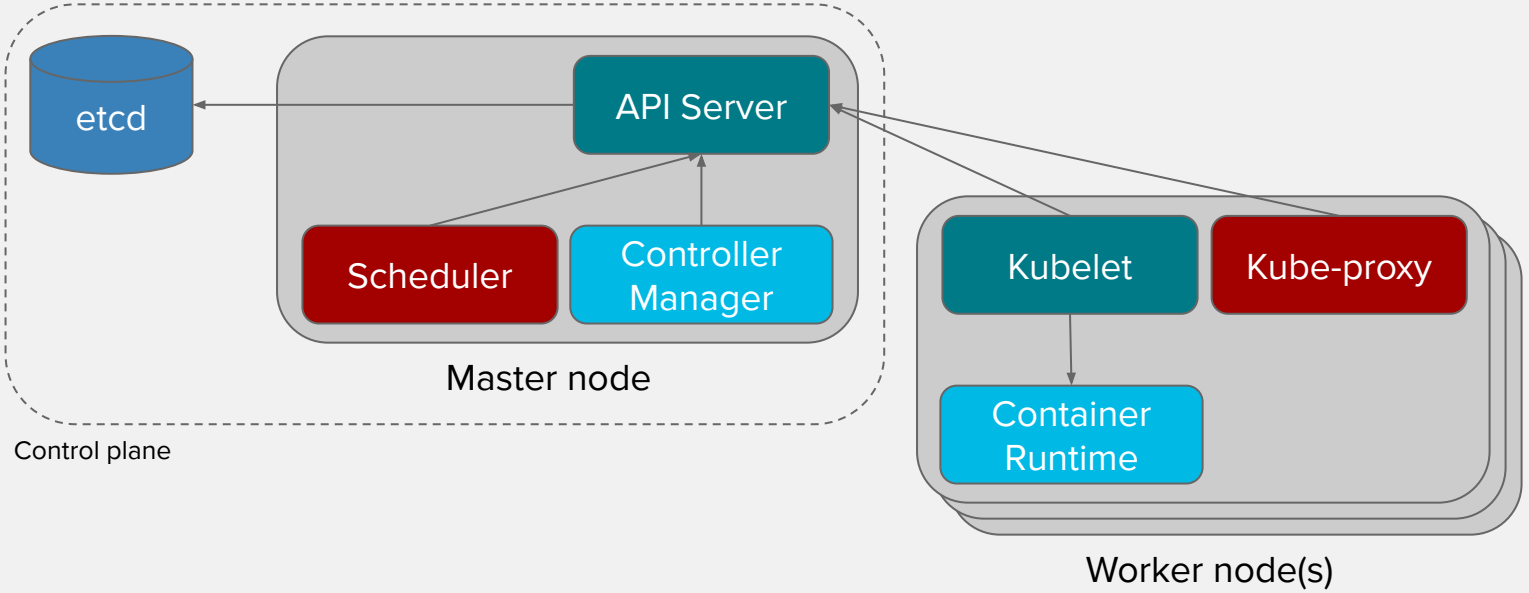
Main features

- **Automated rollouts and rollbacks**
  - Rollouts change to the applications
  - Automatically rollback if something goes wrong
- **Secret and configuration management**
  - Deploy and update secrets and application configuration
- **Storage orchestration**
  - Automatically mount the storage system of your choice (local, network, AWS, ...)
- **Batch execution**
  - Support for running batch jobs (other than long running services)

Why should I used it ?

- **Better hardware utilization** at lower cost
- **Infrastructure abstraction**
  - Applications don’t care if the cluster runs on-premise or with a cloud provider
- Make developing and running **“cloud native” applications** easier
- Support for many different type of workloads
  - Long running services
  - Batch jobs
  - ...

Architecture



Architecture

Master node(s)

- Runs the **control plane** for the cluster
- **API server**
  - Exposes HTTP REST API for accessing Kubernetes resources
- **Scheduler**
  - Defines the worker nodes to schedule containers
- **Control Manager**
  - Performs cluster level functions (components replication, tracking worker nodes, ..)
- Single instance or multiple instance (HA) with elected leader

Architecture

ETCD

- Reliable distributed key-value store
- **Store the cluster state** (and all related resources)
- More instances :
  - Data are replicated
  - Leader is elected
- Can run on the Master or on separate hosts
- **Accessible only by API server**

Architecture

Worker node(s)

- Used to run applications
- **Kubelet**
  - Manages containers on the node
- **Kube-proxy**
  - Manage network traffic between containers
- **Container runtime**
  - Runs containers
  - Docker, cri-o, rkt or others

Kubernetes API

- **HTTP REST API**
  - Used to access Kubernetes resources
- Split into **API groups**
  - Core API, i.e. api/v1
  - Beta or Alpha extensions, i.e. api/extensions/v1beta1
- Each resource is described with :
  - API version, Kind, Metadata, Spec
- Can be accessed via
  - Directly (an HTTP client)
  - Provided tools (kubectl)

Kubectl

- **CLI tool** for interacting with Kubernetes API
  - Used for create, list, update and delete resources
  - Communicate with API server, HTTP REST API
- Advanced usage
  - **Proxy connections** between cluster and localhost
  - **Get logs** from containers
  - Running **commands into containers**
  - Connecting into running containers
- Local configuration
  - API endpoint and user credentials

Distributions

- Kubernetes is **open source**
- Companies offer a Kubernetes distribution adding more on top of it
  - Red Hat OpenShift (OCP)
  - CoreOS Tectonic (now in Red Hat)
  - Canonical
  - VMWare
- Public cloud providers
  - Microsoft Azure Container Service (AKS)
  - Google Kubernetes Engine (GKE)
  - Amazon Elastic Container Service (EKS)

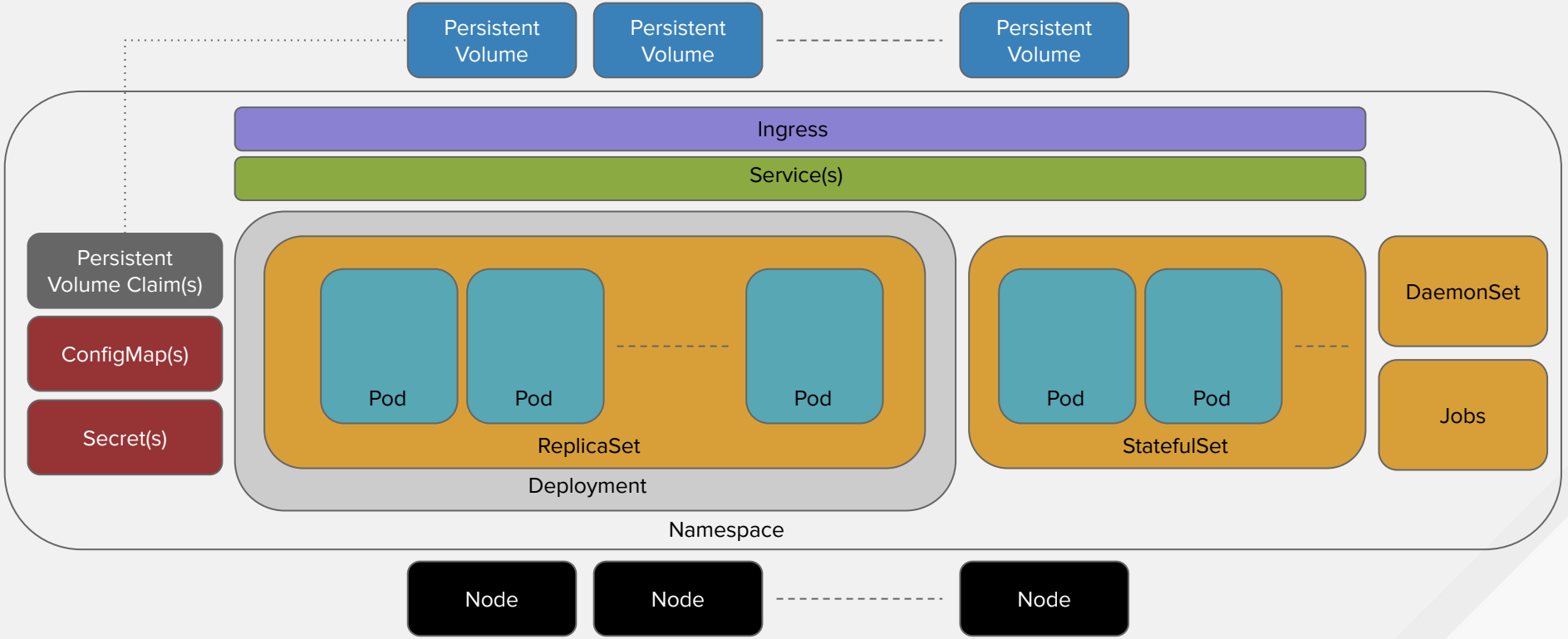
Development

- **Minikube**
  - Single node Kubernetes cluster
  - Runs in a VM
- **Minishift**
  - The OpenShift “incarnation” of Minikube
- **OpenShift Origin**
  - The open source OpenShift project
  - Derived from Kubernetes
  - Single node on local PC ➜ `oc cluster up` !
  - Multi node installation

Resources

- Kubernetes API provides a REST interface so ....
- ... we handle resources !
- There are different types of resources
- More than one resources are needed for deploying an application
- Each resource has its specific purpose
- They have **labels** :
  - For identification
  - For filtering, querying ...
- Are described by YAML or JSON

Resources



Namespaces

- Most of the **resources (not all) are grouped** into namespaces
  - Avoiding name conflicts
  - Applications isolation

Pods

- **Smallest computing unit**
- Every Pod has its own IP address (not exposed outside the cluster)
  - Such address is **not stable** or Pod re-creation
- A Pod is **scheduled on a node**
- Contains **one or more containers** :
  - They share resources and can easily share data
  - They **share the Pod's IP address and the ports space**
  - Unhealthy **containers are automatically restarted**
- Pods are not automatically restarted
- **Healthy and readiness probes** (HTTP GET, TCP socket and exec)

ReplicaSets

- Ensure that Pods are running in a desired number (replicas)
- Use **selector(s)** on labels for identifying Pods to handle
  - Starts new Pods if needed
  - Kills running Pods if needed
- The ReplicaSets description contains a **template** for spinning new Pod replicas
- Usually not used directly but “under” Deployments

Services

- **Pods have unstable IP address**
  - Cannot be used for access containers from other applications
  - Not exposed outside the cluster
- A Service targets one or more Pods ... using **selector(s)** on lables
- Pod/Container accessible through Service ports
- **Stable IP address and DNS name**
- **Load balancing** traffic on target Pods
- Services hosts and ports are exposed through environment variables

Services

Types

- **ClusterIP**
  - Assigns internal IP address / DNS name
  - Accessible only within the cluster
- **NodePort**
  - The Service is directly accessible through a specific port on a Node
  - Used for debugging purposes
- **LoadBalancer**
  - Creates a load balancer which got an IP address accessible from outside
  - Mostly used on cloud provider (AWS, Azure, GCP, ...)
- **None**
  - So called “headless” services used with StatefulSets

Volumes

- **Map disks** to Pods
- Are **mounted into containers**
  - One volume can be mounted into different containers in same Pod (sharing data)
  - Can be mounted on different mount paths
- Different types :
  - emptyDir
  - hostPath
  - NFS
  - AWS/GCP/Azure disks
  - PersistantVolumeClaims

ConfigMaps

- The way for **configuring applications**
- Used to set values or files/directories content
- Their **values can be mapped as environment variables**
  - Directly accessible by containers
- They **can be mounted as volumes**
  - Containers access to files

Secrets

- Similar to ConfigMaps but for store … **secrets**
  - Passwords
  - Private keys

Deployments

- Used to .... **deploy** an application
- Instead of using Pods or ReplicaSets directly
- Create ReplicaSets which then create Pods
- **Support for rolling updates** and canary deployments

Persistent Volumes and Claims

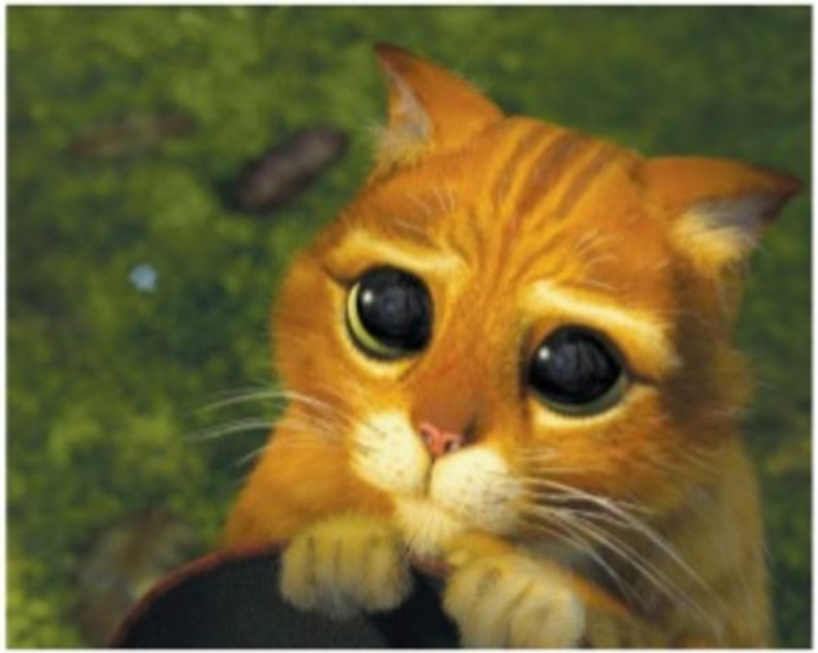
- **Storage abstraction** for persistent disks
- The developer doesn’t need to know the underlying storage type (NFS, GlusterFS, ...)
- Mostly used to map volumes to disks from cloud provider
- Can be provisioned ...
    - Statically by the cluster administrator
    - Dynamically using Storage class and related provisioner
- A Pod can **claim** a Persistent Volume

Ingress

- Another way to expose services outside
- Service LoadBalancer needs its own IP address (can be expensive on a cloud provider)
- An ingress …
  - Requires only one IP
  - Provides access to multiple services
  - The host and path in the request determines the service to which forward the request itself
- An ingress controller is needed in the cluster (i.e. it’s a minikube addon)
- Only HTTP and HTTPS supported today

StatefulSets

Pets vs Cattle



StatefulSets

- Stateless application are like cattle
  - You can easily replace them
  - No need for addressing individually (in a cluster, i.e. Kafka, Zookeeper, ...)
- Stateful application are like pets
  - They have an identity and you take care of them
  - Need to address replicas
- StatefulSets helps for ...
  - Having unique identity using an index in the name
  - Having fixed IP addresses

… and many more !

- Jobs
- DaemonSets
- Custom Resources

What is OpenShift ?

- An enterprise grade Kubernetes version
- Built on top of Kubernetes but adding ...
  - Template (vs using “Helm”)
  - Source-2-Image for building images from source code
  - Routes (vs Ingress)
  - Projects (as Namespaces)
  - A really great dashboard !
  - ... and more ...

Resources

- **Demo :** https://github.com/ppatierno/devday-kubernetes-openshift
- **Kubernetes** : https://kubernetes.io/
- **OpenShift (Origin)** : https://www.openshift.org/
- **Minikube** : https://github.com/kubernetes/minikube
- **Minishift** : https://github.com/minishift/minishift
- **OpenShift (OCP)** : https://www.openshift.com/
- **Azure Container Service** : https://azure.microsoft.com/en-us/services/container-service/
- **Google Kubernetes Engine** : https://cloud.google.com/kubernetes-engine/
- **Amazon Elastic Container Service** : https://aws.amazon.com/eks/
- **Docker** : https://www.docker.com/
- **Cri-o** : http://cri-o.io/
- **Rkt** : https://coreos.com/rkt/

Thank you ! Questions ?

@ppatierno