An Introduction to Kubernetes

Premys Kafka
premysl.kafka@hpe.com  kafkapre  https://github.com/kafkapre
History

- ???? - Virtual Machines
- 2008 - Linux containers (LXC)
- 2013 - Docker
- 2013 - CoreOS stack
- 2014 - Kubernetes
Distributed systems

• Collection of computers that act, work, and appear as one large computer

• Advantages:
  • scalable horizontally, reliable, extensible, cheaper computers, geographic distribution, ...

• Problems:
  • network can fail, bandwidth, consistency, workload allocation, security issues, ...
Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

Name Kubernetes is from Greek word kybernētēs which means "helmsman" or "governor“

Firstly announced by Google in 2014

Influenced by Google's Borg

Many contributors (Google, RedHat, IBM, Cisco, VMware)
Overview

• Commonly referred to as "k8s"
• Apache License 2.0
• Written in Go
• Cross-platform (POC for Windows)
• Kubernetes v1.0 was released on July 21, 2015
• Google Container Engine runs k8s
{ Features }

• Automatic binpacking
• Horizontal scaling
• Self-healing
• Storage orchestration
• Service discovery and load balancing
• Secret and configuration management
• A group of one or more containers with shared storage
• Containers also share an IP address and port space, and can find each other via localhost
• Treated as one logical object
• Deployed on one machine together

• Life cycle
  • Bound to a node, restart in place
  • Can die, cannot be reborn with same ID
Replication Controller

- Creates and destroys Pods dynamically
- Ensures that a pod or particular set of pods are always up and available
  - pods are automatically replaced if they fail
- Scaling Pods
- Recommend to use Replication Controller even when you want to create one Pod
Service

• An abstraction which defines a logical set of Pods and a policy by which to access them
• Pods are targeted and determined by a Label Selector
**Service Types**

- **ExternalName**: map the service to the contents of the `externalName` field e.g. example.com
- **ClusterIP**: use a cluster-internal IP only
- **NodePort**: on top of having a cluster-internal IP, expose the service on a port on each node of the cluster
- **LoadBalancer**: on top of having a cluster-internal IP and exposing service on a NodePort also, ask the cloud provider for a load balancer which forwards to the Service
Labels

• Key/value pairs that are attached to objects, such as pods
• Can be used to organize and to select subsets of objects
• Each Key must be unique for a given object
• Loose coupling
• Can be added dynamically
• Two types of nodes
  • Master node: Master components (APIs, scheduler, etc) are there
  • Worker node: Kubelet and containers are there

• Kubelet: takes a set of PodSpecs and ensures that the containers described are running and healthy

• Kube-proxy: reflects services on each node and can do simple TCP,UDP stream or round robin TCP,UDP forwarding across a set of backends

• API Server: Do business logic implemented in separate components or in plug-ins. Also processes REST operations, validates them, and updates the corresponding objects in persistence (Etcd)
{ Architecture }

• **Scheduler**: binds unscheduled pods to node

• **Control Manager**: processes controllers like DaemonSet Controller, Replication Controller

• **Etcd**: distributed storage

• **Kubectl**: command line client
{ Architecture }

- Kubernetes (user commands)
- Authentication
- REST (pods, services, etc. controllers)
- Scheduler
- Controller manager (replication controller, etc.)
- Distributed Watchable Storage (implemented via etcd)
- Node
- Pod
- Container
- Docker
- Internet
- Firewall
{ Service Schema }

Diagram:
- Client
- kube-proxy
- ServiceIP (iptables)
- Pod 1: port 9376
- Pod 2: port 9376
- Pod 3: port 9376
- apiserver

Node
{ Other Features }

- Volumes (Git, NFS, Flocker, some cloud provider storage)
- DNS
- Namespaces
- DaemonSet
- Jobs
- Secrets
- ConfigMaps
- Autoscaling
- Resource limitation
{ Try Kubernetes }

• Install it
  • Minikube – runs virtual machine with k8s
  • Kubeadm – installs k8s on your machine
• Model your application in k8s yamls
• Deploy and manage your application
• Mesos
• Docker Swarm
K8S DEMO
see: https://github.com/kafkapre/linuxdays2016-kubernetes-example
Thank you!
Q & A

premysl.kafka@hpe.com  kafkapre  https://github.com/kafkapre