Kubernetes

28 November 2017

Giacomo Tartari
Senior Engineer, UiT-ITA-HPC

What is Kubernetes

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

in other words *The Cloud*™

or at least one of the building blocks for PaaS

What Kubernetes isn't

- Silver bullet
- Panacea
- Solution to all the problems afflicting IT

Why?

Because we should stop running servers and start running services

Because servers are pets and we need cattle

We need a tool to mass-manage and streamline operations/development

Kubernetes appers to be the winner of the container orchestration

Intermezzo - containers

It's like a VM but better... and worse

Virtualize the OS not the hardware

Cgroups, namespaces, and beyond: what are containers made from? (https://youtu.be/sK5i-N34im8)

Run containers on bare metal already! (https://www.youtube.com/watch?v=coFIEH3vXPw)

Simmetry

Kubernetes/Docker for **compute resources**

NAS, file servers, Software Defined Storage for **storage**

Configuration

Kubernetes run on clusters in master-worker configurations

High Availability with at least 3 master

Masters are API endpoints

Scales to ~5000 nodes

Yaml files

Pods

Set of containers

Docker containers <-> processes

Pods are set of process running in a virtualized OS

Objects/maniest files

```
apiVersion: v1
kind: Pod
metadata:
  name: constraintpod
  labels:
  env: production
spec:
  containers:
  - name: sise
    image: mhausenblas/simpleservice:0.5.0
    ports:
    - containerPort: 9876
    resources:
      limits:
        memory: "64Mi"
        cpu: "500m"
  livenessProbe:
    initialDelaySeconds: 2
    periodSeconds: 5
    httpGet:
    path: /health
    port: 9876
```

file from here (http://kubernetesbyexample.com/)

ReplicationControlloer

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: rcex
spec:
  replicas: 1
  selector:
    app: sise
  template: # pod starts here
    metadata:
      name: somename
      labels:
        app: sise
    spec:
      containers:
      - name: sise
        image: mhausenblas/simpleservice:0.5.0
        ports:
        - containerPort: 9876
```

Pod supervisor

Deployment

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: sise-deploy
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: sise
    spec:
      containers:
      - name: sise
        image: mhausenblas/simpleservice:0.5.0
        ports:
        - containerPort: 9876
        env:
        - name: SIMPLE_SERVICE_VERSION
          value: "0.9"
```

ReplicaSet supervisor

Deployments

Handles the rollout of new versions

Automatic rollback in case somethig is wrong

Keep history of deployments

Cloud @home

CI/CD pipeline

Git repository

Docker registry (private)

Kubernetes

(Storage)

What about the rest?

I need moar

My pods need to have credentials/config safely stored

My pods need persistent storage

My pods need to talk to many pods but not all the pods

and DNS
and load balancing

multitenancy?
access control?

ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: myapp-config
data:
  key: value
  newKey: |
  very long value that
  can span multiple lines
  such as s config file
```

Keys can be mounted in containers as env variables or volumes

Secrets

```
apiVersion: v1
kind: Secret
metadata:
  name: mariadb-secrets
type: Opaque
data:
  root: cG9ydHVz
```

Secrets are like ConfigMaps but opaque

They need special permission to be read

In action

```
---
...
  containers:
  - name: mariadb
    image: mariadb:10.0.23
    env:
    - name: MYSQL_DATABASE
      value: portus_production
    - name: MYSQL_ROOT_PASSWORD
      valueFrom:
        secretKeyRef:
          name: mariadb-secrets
          key: root
    ports:
    - containerPort: 3306
```

The secret is available through the MYSQL_ROOT_PASSWORD env variable

Volumes

A directory accessible to all containers running in a pod

The data in volumes is preserved across container restarts

Different backing: NFS, Ceph, azureDisk, glusterfs, gitRepo

Local storage on the node: emptyDir, hostPath

And values from ConfigMaps and Secrets

In action

```
containers:
- name: registry
  image: registry:2.6
  volumeMounts: # container
  - name: registry-config
    mountPath: /etc/docker/registry
    readOnly: true
  - name: registry-storage
    mountPath: /registry-data
volumes: # pod
- name: registry-config
  configMap:
    name: portus-registry
    items:
    - key: config
      path: config.yml
- name: registry-storage
  emptyDir: {} # test or temp files only!!!
```

Service

```
apiVersion: v1
kind: Service
metadata:
  name: registry
  labels:
  app: registry
spec:
  type: NodePort
  # type: LoadBalancer
  ports:
  - port: 5000
  targetPort: 5000
  selector:
  app: registry
  tier: registry
```

Network policy

```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
  name: portus-registry
  namespace: scratch
spec:
  podSelector:
  matchLabels:
    app: registry
    tier: registry
  ingress:
  - from:
    - namespaceSelector:
      matchLabels:
      name: kube-ingress
    - podSelector:
      matchLabels:
      app: portus
      tier: registry
    ports:
    - protocol: TCP
      port: 5000
    - protocol: TCP
      port: 5001
```

Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: registry
  namespace: scratch
  annotations:
    kubernetes.io/tls-acme: "true"
spec:
  tls:
  - hosts:
    - registry.scratch.ioudaas.no
    secretName: registry-tls
  rules:
  - host: registry.scratch.ioudaas.no
    http:
      paths:
      - path: /
        backend:
          serviceName: registry
          servicePort: 5000
```

Job API

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi-with-timeout
spec:
  backoffLimit: 5
  activeDeadlineSeconds: 100
  template:
  metadata:
    name: pi
  spec:
    containers:
    - name: pi
    image: perl
    command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
    restartPolicy: Never
```

Job API CronJobs

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
  spec:
    template:
    spec:
      containers:
      - name: hello
      image: busybox
      args:
      - /bin/sh
      - -c
      - date; echo Hello from the Kubernetes cluster
      restartPolicy: OnFailure
```

Namespaces

Kubernetes supports multiple virtual clusters backed by the same physical cluster.
These virtual clusters are called namespaces.

A mechanism to attach authorization, policy and constraints to a subsection of the cluster

(https://github.com/kubernetes/community/blob/master/contributors/design-proposals/architecture/namespaces.md)

RBAC

In the RBAC API, a role contains rules that represent a set of permissions.

A role can be defined within a namespace with a Role, or cluster-wide with a ClusterRole.

A Role can only be used to grant access to resources within a single namespace.

More info here (https://kubernetes.io/docs/admin/authorization/rbac/)

Demo

Environment

Helm/tiller Helm (https://helm.sh/)

Kompose (http://kompose.io/)

Draft (https://draft.sh/)

conformace (https://www.cncf.io/certification/software-conformance/)

Who else is using it?

RedHat (Openshift) (https://www.openshift.com/)

dave.cheney.net/2017/09/06/why-i-joined-heptio (https://dave.cheney.net/2017/09/06/why-i-joined-heptio)

DockerSwarm (https://techcrunch.com/2017/10/17/docker-gives-into-invevitable-and-offers-native-kubernetes-support/)

Rancher

mesos (https://www.prnewswire.com/news-releases/mesosphere-launches-kubernetes-beta-on-new-dcos-110-empowers-developers-with-freedom-of-choice-across-container-orchestration-and-data-services-300514972.html)

Azure

Google

Windows support?

is coming (https://kubernetes.io/docs/getting-started-guides/windows/)

How do I try it?

On your laptop

- minikube (https://github.com/kubernetes/minikube)

or go to the Cloud

- turnkey-cloud-solutions (https://kubernetes.io/docs/setup/pick-right-solution/#turnkey-cloud-solutions)
- kops (AWS) (https://github.com/kubernetes/kops)
- kubernetes-the-hard-way (https://github.com/kelseyhightower/kubernetes-the-hard-way)

Thank you

Giacomo Tartari
Senior Engineer, UiT-ITA-HPC
giacomo.tartari@uit.no (mailto:giacomo.tartari@uit.no)