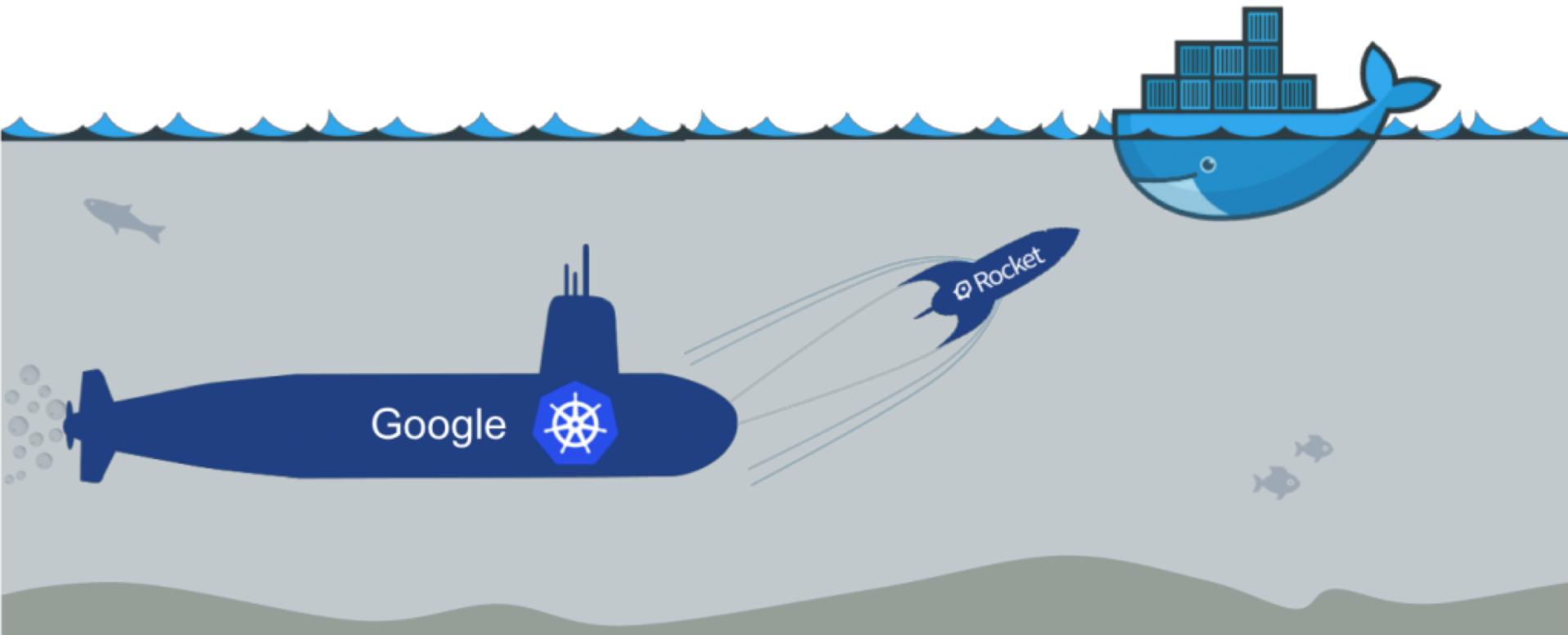




Service Orchestration

BigData from testing to production

PhD. Ronald Muresano

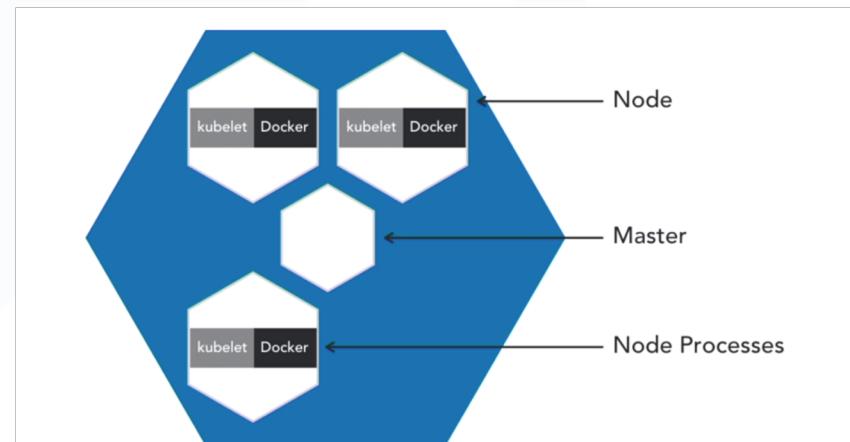
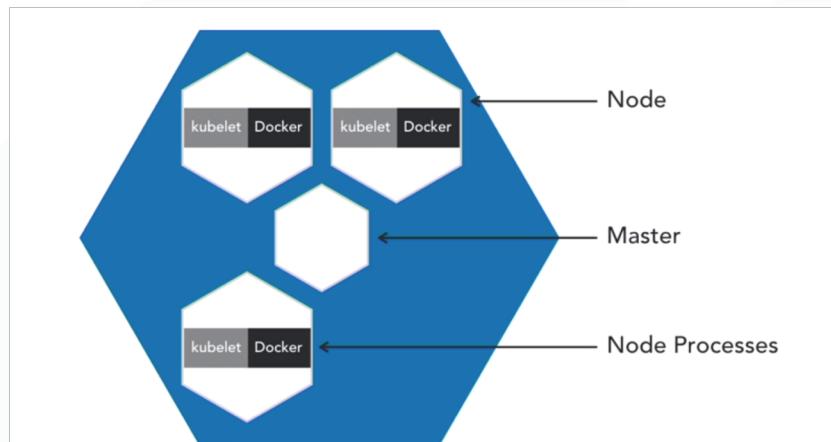


Kubernetes (Multi-container Services) Architecture

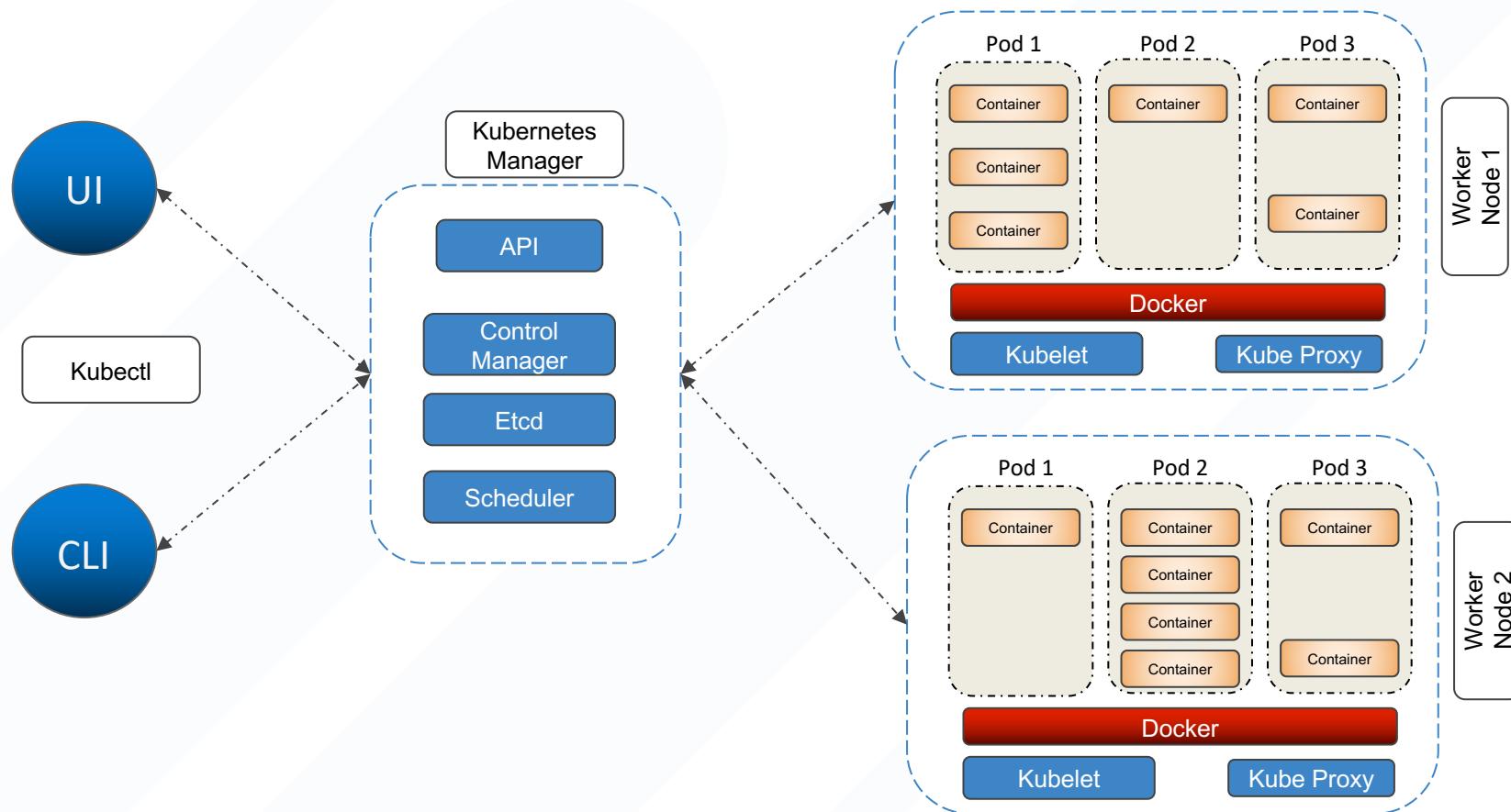
Kubernetes is **an open-source system** for managing containerized applications across multiple hosts in a cluster. Kubernetes provides mechanisms for **horizontally scale**, application deployment, **scheduling**, updating, maintenance, **scaling**, run stateless and stateful application and **ideal to execute micro-services**.

It is a declarative language for launching containers

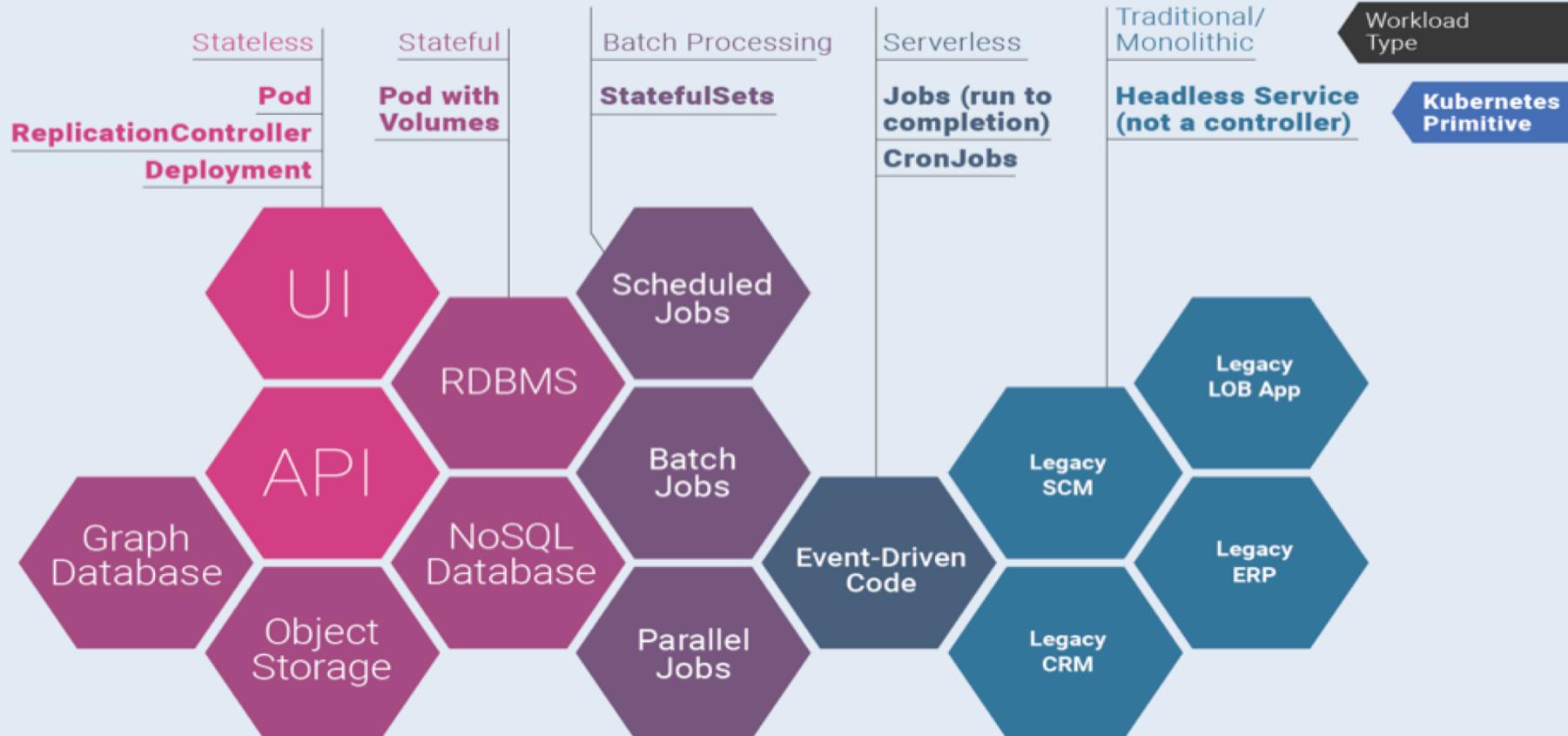
The state of the entities in the system at any given point of time is represented by Kubernetes Objects. Kubernetes Objects also act as an additional layer of abstraction over the container interface.



Kubernetes (Multi-container Services) Architecture



Mapping Workloads to Kubernetes Primitives



Kubernetes (Multi-container Services) Cluster Example

Creating Kubernetes
Cluster

Pre-requisites

- [Vagrant 2.1.4+](#)
- [Virtualbox 5.2.18+](#)



Vagrant up

Vagrant File

<https://github.com/mureron/bdaasmicroservices/blob/master/demos/kubernetes/cluster/Vagrantfile>

Kubernetes (Multi-container Services) Cluster Example

Getting the kubernetes
Cluster Info

```
[vagrant@k8-manager-bdaas:~/spark$ kubectl get nodes
NAME           STATUS  ROLES   AGE      VERSION
k8-manager-bdaas  Ready   master   2d11h   v1.15.0
k8-worker-bdaas-1  Ready   <none>  2d11h   v1.15.0
k8-worker-bdaas-2  Ready   <none>  2d11h   v1.15.0
[vagrant@k8-manager-bdaas:~/spark$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
jupyterspark-notebook-5566f85d95-qzk5z  1/1    Running   0          19h
spark-master-86d944466b-nnwbs          1/1    Running   0          19h
spark-worker-cfbb8dfdc-q9fbf          1/1    Running   0          19h
[vagrant@k8-manager-bdaas:~/spark$ kubectl get deployments
NAME            READY  UP-TO-DATE  AVAILABLE  AGE
jupyterspark-notebook  1/1     1          1          19h
spark-master      1/1     1          1          19h
spark-worker       1/1     1          1          19h
[vagrant@k8-manager-bdaas:~/spark$ ]
```

Main Kubernetes objects (Namespaces)

They are virtual clusters backed by the physical cluster. It is great concept to use for large enterprises where there are many users and teams and you want to give access to different teams but at the same time have a rough idea of who owns what in the Kubernetes environment.

namespace.yaml

- A great way to divide cluster resources between multiple users and this can be done using resource quotas.
- Names of resources, like deployments and pods, must be unique within the namespace, but not necessarily across separate namespaces.
- There is a **default** namespace where all our objects get placed

```
1  apiVersion: v1
2  kind: Namespace
3  metadata:
4    name: big-data-service
```

```
[vagrant@k8-manager-bdaas: $ kubectl get namespaces
NAME          STATUS   AGE
big-data-service  Active  10s
default        Active  13d
kube-node-lease  Active  13d
kube-public     Active  13d
kube-system     Active  13d
```

Main Kubernetes objects (Pods)

Pods: is the smallest deployable unit on a Node. It's a group of containers which must run together. Quite often, but not necessarily, a Pod usually contains one container.



Pod States

- Pending
- Running
- Succeeded
- Failed
- CrashLoopBackOff

Image
Ports
Volume

pod.yaml

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: gluster-pod1
5    labels:
6      name: gluster-pod1
7  spec:
8    containers:
9      - name: gluster-pod1
10     image: gcr.io/google_containers/nginx-slim:0.8
11     ports:
12       - name: web
13         containerPort: 80
14         securityContext:
15           privileged: true
16         volumeMounts:
17           - name: gluster-vol1
18             mountPath: /usr/share/nginx/html
19         volumes:
20           - name: gluster-vol1
21             persistentVolumeClaim:
22               claimName: gluster1
```

Main Kubernetes objects (Services)

Service is used to define a logical set of Pods and related policies used to access them.

Internal services, where an IP is only reachable from within the cluster.

External services where services running web servers, or publicly accessible pods, are exposed through an external endpoint. These endpoints are available on each node through a specific port. This is called a NodePort

Load balancer. This is for use cases when you want to expose your application to the public internet.

service.yaml

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: jupyter-notebook
5    labels:
6      name: jupyter-notebook
7  spec:
8    ports:
9      - protocol: TCP
10     port: 8888 #
11     targetPort: 8888
12   selector:
13     name: jupyter-notebook
```

```
[vagrant@k8-manager-bdaas:~/examples/namespaces]$ kubectl get services -o wide
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE   SELECTOR
jupyter-notebook  ClusterIP  10.110.120.30 <none>        8888/TCP      42m   name=jupyter-notebook
kubernetes      ClusterIP  10.96.0.1    <none>        443/TCP       13d   <none>
```

Main Kubernetes objects (Volume)

It is essentially a directory accessible to all containers running in a Pod. Lifetimes are not managed and until very recently there were only local-disk-backed volumes

storage-class.yaml

```
1 kind: StorageClass
2 apiVersion: storage.k8s.io/v1
3 metadata:
4   name: local-storage
5 provisioner: kubernetes.io/no-provisioner
6 volumeBindingMode: WaitForFirstConsumer
```

Persistent-Volume-Claim.yaml

```
13  apiVersion: v1
14  kind: PersistentVolumeClaim
15  metadata:
16    name: jupyter-pv-claim
17    labels:
18      app: jupyter
19  spec:
20    storageClassName: manual
21    accessModes:
22      - ReadWriteOnce
23    resources:
24      requests:
25        storage: 2Gi
```

```
38  template:
39    metadata:
40      labels:
41        app: jupyter-notebook
42    spec:
43      containers:
44        - name: rmuresano-jupyter
45          image: rmuresano/bdaasjupyter:0_0_4
46          env:
47            - name: STANDALONE
48              value: "YES"
49            - name: CASSANDRA
50              value: "NO"
51            - name: HDFS
52              value: "NO"
53            - name: JUPYTERPORT
54              value: "8888"
55          ports:
56            - containerPort: 8888
57          volumeMounts:
58            - name: jupyter-storage
59              mountPath: /data/jupyter
60        volumes:
61          - name: jupyter-storage
62            persistentVolumeClaim:
63              claimName: jupyter-pv-claim
```

Kubernetes Controllers Objects

These are used to help us in the following tasks **Application Reliability, Scaling and Load Balancing.**

There are a number of Controllers provided by Kubernetes.

- **Replication Controller / Replica Set** ensures that a specified number of Pod replicas are running at any given time.
- **Deployment** is used to change the current state to the desired state.
- **StatefulSet** is used to ensure control over the deployment ordering and access to volumes, etc.
- **Job** is used to perform some task and exit after successfully completing their work or after a given period of time.
- **DaemonSet** is used to run a copy of a Pod on all the nodes of a cluster or on specified nodes.

Replication Controllers

- A *ReplicationController* ensures that a specified number of pod replicas are running at any one time.
- It makes sure that a pod or a homogeneous set of pods is always up and available.
- In case of a Pod Fail it automatically starts a new Pod.

Service.yaml

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: jupyter-notebook
5    labels:
6      name: jupyter-notebook
7  spec:
8    ports:
9      - protocol: TCP
10     port: 8888 #
11     targetPort: 8888
12   selector:
13     name: jupyter-notebook
```

ReplicationController.yaml

```
1  apiVersion: v1
2  kind: ReplicationController
3  metadata:
4    name: jupyter-notebook
5    labels:
6      name: jupyter-notebook
7  spec:
8    replicas: 1
9    selector:
10   name: jupyter-notebook
11  template:
12    metadata:
13      labels:
14        name: jupyter-notebook
15    spec:
16      containers:
17        - name: rmuresano-jupyter
18          image: rmuresano/bdaasjupyter:0_0_5
19          env:
20            - name: STANDALONE
21              value: "YES"
22            - name: CASSANDRA
23              value: "NO"
24            - name: HDFS
25              value: "NO"
26            - name: JUPYTERPORT
27              value: "8888"
28          ports:
29            - containerPort: 8888
```

Replication Controllers

```
vagrant@k8-manager-bdaas:~/examples/replicationController$ kubectl get rc,pods,services
NAME                                DESIRED   CURRENT   READY   AGE
replicationcontroller/jupyter-notebook   1         1         0       87s
```

```
NAME          READY   STATUS            RESTARTS   AGE
pod/jupyter-notebook-gjpj6   0/1    ContainerCreating   0         87s
pod/pod-env-var      1/1    Running           0         50m
```

```
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/jupyter-notebook   ClusterIP   10.110.120.30 <none>        8888/TCP   6m52s
service/kubernetes        ClusterIP   10.96.0.1     <none>        443/TCP    13d
```

```
vagrant@k8-manager-bdaas:~/examples/replicationController$ kubectl get rc,pods,services
NAME                                DESIRED   CURRENT   READY   AGE
replicationcontroller/jupyter-notebook   1         1         1       5m16s
```

```
NAME          READY   STATUS            RESTARTS   AGE
pod/jupyter-notebook-gjpj6   1/1    Running           0         5m16s
pod/pod-env-var      1/1    Running           0         54m
```

```
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/jupyter-notebook   ClusterIP   10.110.120.30 <none>        8888/TCP   10m
service/kubernetes        ClusterIP   10.96.0.1     <none>        443/TCP    13d
```

```
vagrant@k8-manager-bdaas:~/examples/replicationController$ kubectl delete pods/jupyter-notebook-gjpj6
pod "jupyter-notebook-gjpj6" deleted
```

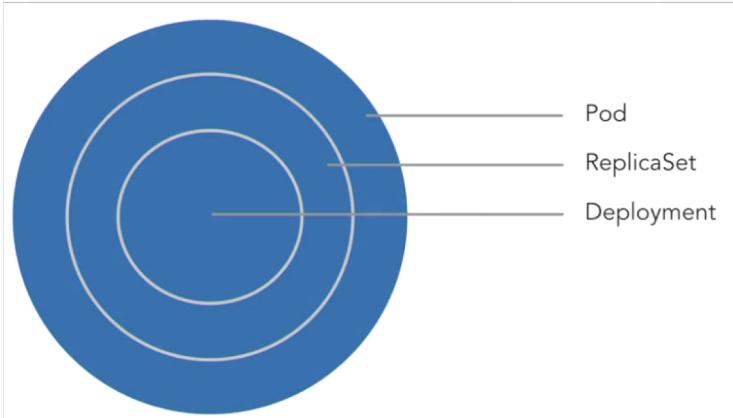
```
vagrant@k8-manager-bdaas:~/examples/replicationController$ kubectl get pods
NAME          READY   STATUS            RESTARTS   AGE
jupyter-notebook-d5hx4   1/1    Running           0         57s
pod-env-var      1/1    Running           0         55m
```

Deployments

A *Deployment* controller provides declarative updates for Pods and ReplicaSets. It can be defined to create new ReplicaSets or replace existing ones with new ones.

They are higher-level constructs that were introduced to solve specific issues.

- Pod management.
- Scaling a ReplicaSet
- Pod updates and role-backs



Deployment definition

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jupyter-notebook
  labels:
    app: jupyter-notebook
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jupyter-notebook
  template:
    metadata:
      labels:
        app: jupyter-notebook
    spec:
      containers:
        - name: rmuresano-jupyter
          image: rmuresano/bdaasjupyter:0_0_2
          env:
            - name: STANDALONE
              value: "YES"
            - name: CASSANDRA
              value: "NO"
            - name: HDFS
              value: "NO"
            - name: JUPYTERPORT
              value: "8888"
          ports:
            - containerPort: 8888
          volumeMounts:
            - name: jupyter-storage
              mountPath: /data/jupyter
```

Kubernetes (Multi-container Services) Architecture

Deployment definition

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jupyter-notebook
  labels:
    app: jupyter-notebook
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jupyter-notebook
  template:
    metadata:
      labels:
        app: jupyter-notebook
    spec:
      containers:
        - name: rmuresano-jupyter
          image: rmuresano/bdaasjupyter:0_0_2
          env:
            - name: STANDALONE
              value: "YES"
            - name: CASSANDRA
              value: "NO"
            - name: HDFS
              value: "NO"
            - name: JUPYTERPORT
              value: "8888"
          ports:
            - containerPort: 8888
          volumeMounts:
            - name: jupyter-storage
              mountPath: /data/jupyter
```

Volumen definition

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: task-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/data/jupyter"
```

Example:

<https://github.com/mureron/bdaasmicroservices/blob/master/demos/kubernetes/Configuration/services/jupyter/jupyterServiceDepl.yaml>

Stateful

Manages the deployment and scaling of a set of Pods, *and* provides guarantees about the ordering and uniqueness of these Pods.

A StatefulSet maintains a sticky identity for each of their Pods.

These pods are created from the same spec, but are not interchangeable: each has a persistent identifier that it maintains across any rescheduling.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx # has to match .spec.template.metadata.labels
  serviceName: "nginx"
  replicas: 3 # by default is 1
  template:
    metadata:
      labels:
        app: nginx # has to match .spec.selector.matchLabels
  spec:
    terminationGracePeriodSeconds: 10
    containers:
      - name: nginx
        image: k8s.gcr.io/nginx-slim:0.8
        ports:
          - containerPort: 80
            name: web
        volumeMounts:
          - name: www
            mountPath: /usr/share/nginx/html
    volumeClaimTemplates:
      - metadata:
          name: www
        spec:
          accessModes: [ "ReadWriteOnce" ]
          storageClassName: "my-storage-class"
          resources:
            requests:
              storage: 1Gi
```

Jobs

Jobs are a construct that run a pod once and then stop. By itself, jobs aren't really that useful. But a lot of times people end up using CronJobs which are just like jobs, but they run periodically.

jobs.yaml

```
1 apiVersion: batch/v1
2 kind: Job
3 metadata:
4   name: finalcountdown
5 spec:
6   template:
7     metadata:
8       name: finalcountdown
9     spec:
10    containers:
11      - name: counter
12        image: busybox
13        command:
14          - bin/sh
15          - --c
16          - "for i in 9 8 7 6 5 4 3 2 1 ; do echo $i ; done"
17    restartPolicy: Never
```

```
vagrant@k8-manager-bdaas:~/examples/jobs$ kubectl apply -f jobs.yaml
job.batch/finalcountdown unchanged
vagrant@k8-manager-bdaas:~/examples/jobs$ kubectl get jobs -o wide
NAME           COMPLETIONS  DURATION   AGE   CONTAINERS   IMAGES   SELECTOR
finalcountdown  1/1          5s         85s   counter     busybox   controller-uid=55b
vagrant@k8-manager-bdaas:~/examples/jobs$ kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
finalcountdown-m5dgx  0/1    Completed   0          91s
jupyter-notebook-d5hx4 1/1    Running    0          151m
pod-env-var      1/1    Running    0          3h26m
vagrant@k8-manager-bdaas:~/examples/jobs$ kubectl logs finalcountdown-m5dgx
9
8
7
6
5
4
3
2
1
```

ConfigMaps

- It allows to decouple configuration artifacts from image content to keep containerized applications portable.
- One of the benefits it brings ConfigMap is that containers can be portable
- It can be used to pass ENV variables values to a Pod.

```
1 apiVersion: v1
2 kind: ConfigMap
3 metadata:
4   name: example-configmap
5 data:
6   # Configuration values can be set as key-value properties
7   database: mongodb
8   database_uri: mongodb://localhost:27017
9   TEST_VARIABLE: HOLA
10  # Or set as complete file contents (even JSON!)
11  keys: |
12    image.public.key=771
13    rsa.public.key=42
14
```

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: pod-env-var
5 spec:
6   containers:
7     - name: env-var-configmap
8       image: nginx:1.7.9
9       envFrom:
10      - configMapRef:
11        name: example-configmap
```

ConfigMaps

```
[vagrant@k8-manager-bdaas:~/examples/configmap]$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
pod-env-var   1/1     Running   0          6m48s
[vagrant@k8-manager-bdaas:~/examples/configmap]$ kubectl exec -it pod-env-var bash
[root@pod-env-var:/# env
HOSTNAME=pod-env-var
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT=tcp://10.96.0.1:443
TERM=xterm
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_HOST=10.96.0.1
TEST_VARIABLE=HOLA
database_uri=mongodb://localhost:27017
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
database=mongodb
PWD=/
NGINX_VERSION=1.7.9-1~wheezy
keys=image.public.key=771
rsa.public.key=42
SHLVL=1
HOME=/root
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
_=/usr/bin/env
```

Playing with Kubernetes

Exercise 1: Deploy a Jupyter using the volumen claim and services. Time 10-15 min

Steps to follow:

- 1) **Git clone the class repo in case you don't have it**

```
git clone git@github.com:mureron/bdaasmicroservices.git
```

- 2) **Enter in the folder Kubernetes/Services/kubclt/jupyter**
- 3) kubectl apply -f StorageClass.yaml
- 4) kubectl apply -f jupyterServiceDepl.yaml (This file has the Volume Claim, Service and Deployment)
- 5) Make the port Forwarding : kubectl port-forward service/<name> 8003:8888
- 6) Create a ssh Tunnel using in a new terminal ssh -L 8003:localhost:8003 vagrant@192.168.205.10
password : vagrant
- 7) Enter in the service using localhost:8002

Helm as a Kubernetes manager application

It's a tool that coordinates the installation and management of Kubernetes applications.

Helm manages charts, which are packages of preconfigured Kubernetes resources. It uses a client server architecture where the Helm client runs on your local machine and the server component, called Tiller, runs in your Kubernetes cluster as a deployment in its own namespace.



Helm architecture



A Helm chart must contain:

Chart.yaml: contains high-level metadata information about your chart.

Templates: The template file contains your Kubernetes resources and the components that will make up your application

Values.yaml: a file that contains the deployment parameters values.

Helm Chart (Zeppelin-Spark Example)

Chart.yaml

```
1 apiVersion: v1
2 name: spark
3 version: 1.0.0
4 appVersion: 1.5.1
5 description: Fast and general-purpose cluster computing system.
6 home: http://spark.apache.org
7 icon: http://spark.apache.org/images/spark-logo-trademark.png
8 sources:
9   - https://github.com/kubernetes/kubernetes/tree/master/examples/spark
10  - https://github.com/apache/spark
11 maintainers:
12   - name: lachie83
13     email: lachlan.evenson@gmail.com
```

Helm Chart (Zeppelin-Spark Example)

values.yaml

```
1 # Default values for spark.
2 # This is a YAML-formatted file.
3 # Declare name/value pairs to be passed
4 # name: value
5
6 Spark:
7   Path: "/opt/spark"
8
9 Master:
10  Name: master
11  Image: "k8s.gcr.io/spark"
12  ImageTag: "1.5.1_v3"
13  Replicas: 1
14  Component: "spark-master"
15  Cpu: "100m"
16  Memory: "512Mi"
17  ServicePort: 7077
18  ContainerPort: 7077
19  # Set Master JVM memory. Default 1g
20  # DaemonMemory: 1g
21  ServiceType: LoadBalancer
```

```
23 WebUi:
24   Name: webui
25   ServicePort: 8080
26   ContainerPort: 8080
27
28 Worker:
29   Name: worker
30   Image: "k8s.gcr.io/spark"
31   ImageTag: "1.5.1_v3"
32   Replicas: 3
33   Component: "spark-worker"
34   Cpu: "100m"
35   Memory: "512Mi"
36   ContainerPort: 8081
37   # Set Worker JVM memory. Default 1g
38   # DaemonMemory: 1g
39   # Set how much total memory workers have to give executors
40   # ExecutorMemory: 1g
41   Autoscaling:
42     Enabled: false
43     ReplicasMax: 10
44     CpuTargetPercentage: 50
45
46 Zeppelin:
47   Name: zeppelin
```

template/spark-master-deployment.yaml

```
17 apiVersion: v1
18 kind: Service
19 metadata:
20   name: {{ template "webuifullname" . }}
21   labels:
22     heritage: {{ .Release.Service | quote }}
23     release: {{ .Release.Name | quote }}
24     chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
25     component: "{{ .Release.Name }}-{{ .Values.Master.Component }}"
26 spec:
27   ports:
28     - port: {{ .Values.WebUi.ServicePort }}
29       targetPort: {{ .Values.WebUi.ContainerPort }}
30   selector:
31     component: "{{ .Release.Name }}-{{ .Values.Master.Component }}"
32   type: {{ .Values.Master.ServiceType }}
```

Helm Chart (Zeppelin-Spark Example)

Template Folder

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: {{ template "masterfullname" . }}
5   labels:
6     heritage: {{ .Release.Service | quote }}
7     release: {{ .Release.Name | quote }}
8     chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"
9     component: "{{ .Release.Name }}-{{ .Values.Master.Component }}"
10 spec:
11   ports:
12     - port: {{ .Values.Master.ServicePort }}
13       targetPort: {{ .Values.Master.ContainerPort }}
14   selector:
15     component: "{{ .Release.Name }}-{{ .Values.Master.Component }}"
16 ---
17 apiVersion: v1
18 kind: Service
19 metadata:
20   name: {{ template "webuifullname" . }}
```

- [!\[\]\(072b85bca7e3c3f0df32e9cb1d6321d2_img.jpg\) NOTES.txt](#)
- [!\[\]\(c08b2954ced52bd12b7d0886396aa8f5_img.jpg\) _helpers.tpl](#)
- [!\[\]\(2518f1e19e7fb7ef181a867081a31458_img.jpg\) spark-master-deployment.yaml](#)
- [!\[\]\(729e10e84db853a809370fe7baefddc2_img.jpg\) spark-sql-test.yaml](#)
- [!\[\]\(62c45eda4bc23efb369f0ef570de9340_img.jpg\) spark-worker-deployment.yaml](#)
- [!\[\]\(9999676de84988a10f0d1ae0bb783cb4_img.jpg\) spark-worker-hpa.yaml](#)
- [!\[\]\(c33649b1877d776dbaf177a59eb9d8fe_img.jpg\) spark-zeppelin-config-pvc.yaml](#)
- [!\[\]\(346f0340ac100ce7a1e0ff90f4c69fab_img.jpg\) spark-zeppelin-deployment.yaml](#)
- [!\[\]\(9f7fa8e0473d2b280bf380ec0ff46368_img.jpg\) spark-zeppelin-ingress.yaml](#)
- [!\[\]\(b19d745f8f5cf014ed4f33df12703bfb_img.jpg\) spark-zeppelin-notebook-pvc.yaml](#)

Installing Helm Client

Installation procedure:https://helm.sh/docs/using_helm/#installing-helm

From Binaries

Download your [desired version](#)

1. Unpack it (`tar -zxvf helm-v2.0.0-linux-amd64.tgz`)
2. Find the `helm` binary in the unpacked directory, and move it to its desired destination (`mv linux-amd64/helm /usr/local/bin/helm`)

From Snap (Linux)

```
$ sudo apt update  
$ sudo apt install snapd
```

The Snap package for Helm is maintained by [Snapcrafters](#).

```
$sudo snap install helm --classic
```

From Homebrew (macOS)

```
brew install kubernetes-helm
```

Set up the RBAC in kubernetes cluster for tiller Helm

- Tiller runs inside your Kubernetes cluster, and manages releases (installations) of your charts.
- Tiller needs access to the Kubernetes API and the RBAC (Role Base Access Control) block this operations.
- To able this option, we need to do the following steps:
 - Create a Service Account *tiller* for the Tiller server (in the kube-system namespace).
 - Bind the *cluster-admin* ClusterRole to this Service Account. We use ClusterRoleBindings to apply to all name-spaces.
 - Update the existing Tiller deployment

The ***cluster-admin* ClusterRole** exists by default in your Kubernetes cluster, and allows superuser operations in all of the cluster resources. The reason for binding this role is because with Helm charts, you can have deployments consisting of a wide variety of Kubernetes resources



Set up the RBAC in kubernetes cluster for tiller Helm

Step 1: Create The Tiller Service Account

- Create a *tiller-serviceaccount.yaml* file using *kubectl*:

```
kubectl create serviceaccount tiller --namespace kube-system
```

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: tiller-clusterrolebinding
subjects:
- kind: ServiceAccount
  name: tiller
  namespace: kube-system
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: ""
```

Step 2 : Bind The Tiller Service Account To The *Cluster-Admin* Role

- Create a *clusterroleBinding.yaml*
- Deploy it

```
kubectl create -f tiller-clusterrolebinding.yaml
```

Set up the RBAC in kubernetes cluster for tiller Helm

Step 3: Update The Existing Tiller Deployment

- Update the existing *tiller-deploy* deployment with the Service Account you created earlier:

```
helm init --service-account tiller --upgrade
```

| NAMESPACE | NAME | READY | STATUS | RESTARTS | AGE |
|-------------|-----------------------------------------|-------|---------|----------|-----|
| default | zeppelinspark-master-64dc78f778-tln6b | 1/1 | Running | 0 | 37m |
| default | zeppelinspark-worker-79bc8df6b4-84d57 | 1/1 | Running | 0 | 31m |
| default | zeppelinspark-zeppelin-68458f76c6-b2pb4 | 0/1 | Evicted | 0 | 8m |
| default | zeppelinspark-zeppelin-68458f76c6-jvt2l | 0/1 | Evicted | 0 | 19m |
| default | zeppelinspark-zeppelin-68458f76c6-lvj92 | 0/1 | Evicted | 0 | 10m |
| default | zeppelinspark-zeppelin-68458f76c6-n6dc5 | 0/1 | Evicted | 0 | 5m |
| default | zeppelinspark-zeppelin-68458f76c6-rfw25 | 1/1 | Running | 0 | 49s |
| default | zeppelinspark-zeppelin-68458f76c6-spwmc | 0/1 | Evicted | 0 | 42m |
| default | zeppelinspark-zeppelin-68458f76c6-xklgm | 0/1 | Evicted | 0 | 12m |
| kube-system | prometheus-to-sd-fmmcg | 1/1 | Running | 0 | 27m |
| kube-system | prometheus-to-sd-jprkz | 1/1 | Running | 0 | 43m |
| kube-system | prometheus-to-sd-qrw94 | 1/1 | Running | 0 | 47m |
| kube-system | prometheus-to-sd-v7t78 | 1/1 | Running | 0 | 47m |
| kube-system | tiller-deploy-7b659b7fdbd-jc7hh | 1/1 | Running | 0 | 31m |

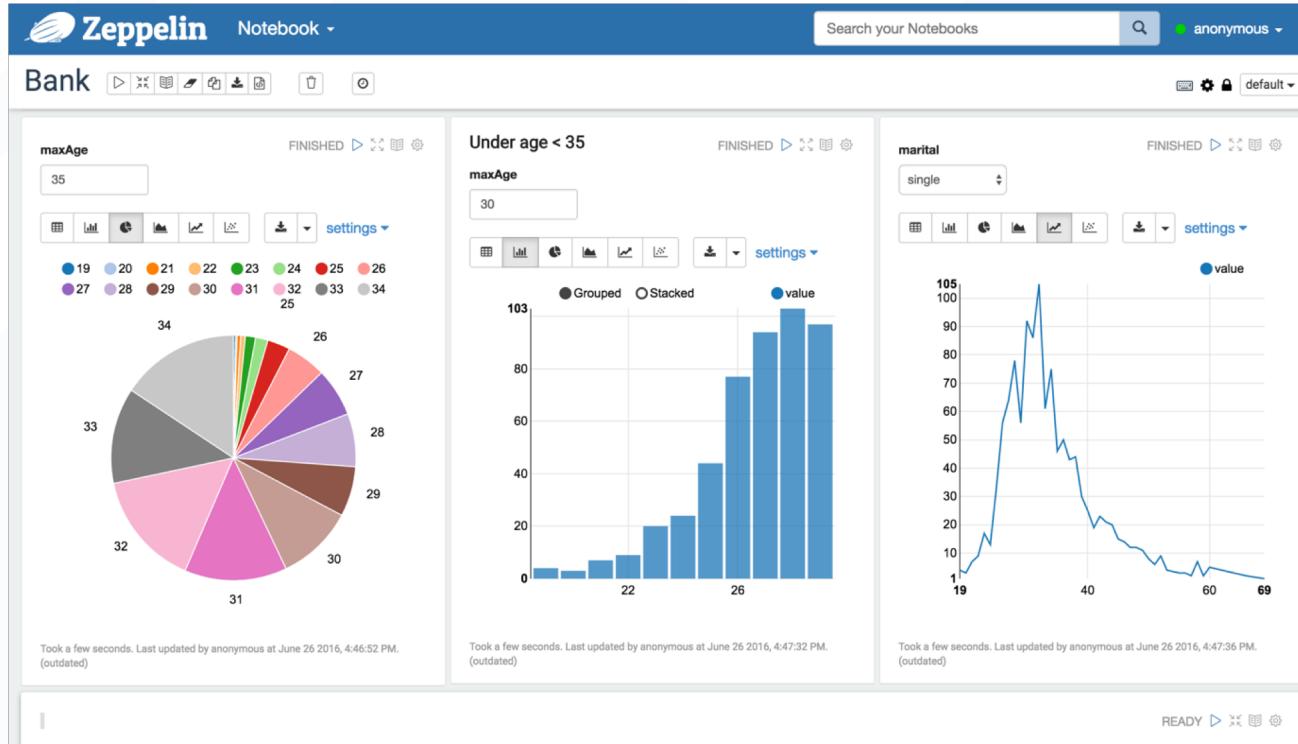
Preparing Helm

Exercise Installation procedure. Estimated time 10 minutes



Using Helm

Example running Zeppelin and Spark



Using Helm

Steps to follow:

- 1) Git clone the class repo in case you don't have it

```
git clone git@github.com:mureron/bdaasmicroservices.git
```

- 2) Enter in the folder Kubernetes/Services/helm

- 3) helm install --name zeppelinspark -f values.yaml stable/spark
(wait while the service is completed deployed)

- 4) kubectl get pods,deployment,services

- 5) Make the port Forwarding : kubectl port-forward service/<name> 8002:8888

- 6) Create a ssh Tunnel using in a new terminal ssh -L 8002:localhost:8002 vagrant@192.168.205.10
password : vagrant
- 7) Enter in the service using localhost:8002

Zeppelin Spark Helm Deployment on Dashboard

 **kubernetes** Search + CREATE | 

☰ Overview

Persistent Volumes
Roles
Storage Classes

Namespace
default ▾

Overview

Workloads

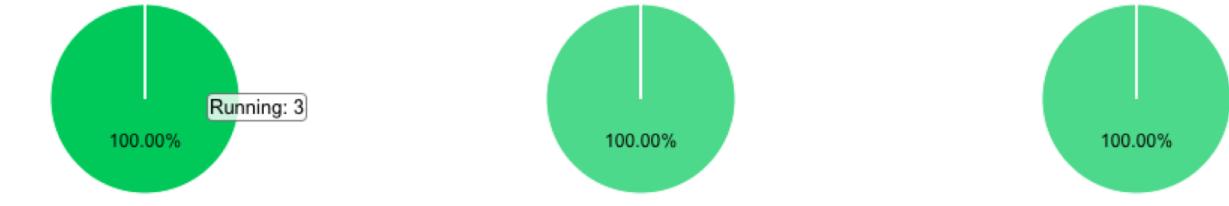
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets

Deployments

| Name | Labels | Pods | Age | Images | ⋮ |
|----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|-------|------------|---------------------------|---|
|  zep-master | chart: spark-1.0.0 component: zep-spark-m... heritage: Tiller release: zep chart: spark-1.0.0 | 1 / 1 | 36 seconds | k8s.gcr.io/spark:1.5.1_v3 | ⋮ |

Pods

Replica Sets



A close-up photograph of a bald man's face. He is wearing dark sunglasses and a dark, possibly black or dark green, military-style beret. His eyes are wide and focused directly at the viewer with a serious, almost intense expression. The background is blurred, showing what appears to be a natural outdoor setting with greenery.

**WELCOME
TO THE
HELM WORLD**

Playing with kubernetes and Helm

Exercise: Deploy Grafana and Node Red using Helm. Time 15 minutes

The values.yaml file are in the folder and play with the parameters and the dashboard



Course Roadmap

Orchestration for developing

Docker and Docker-Compose.

Big Data Challenges

An overview applied to the SMEs.
Orchestration Philosophy

Docker Swarm

Orchestration in a real private cluster (Deploying and Creating services)



Kubernetes

Production Orchestration.
Automating deployment,
scaling, and management of
containerized applications

Cloud Infrastructure

Google Cloud and AWS
deployments

Texting in real Cloud Environment

Access to all Cloud Platform Products

Get everything you need to build and run your apps, websites and services, including Firebase and the Google Maps API.

\$300 credit for free

Sign up and get \$300 to spend on Google Cloud Platform over the next 12 months.

No autocharge after free trial ends

We ask you for your credit card to make sure you are not a robot. You won't be charged unless you manually upgrade to a paid account.



Filter by: [Clear all](#)

▼ Tier Type

- Featured
- 12 Months Free
- Always Free
- Trials

▼ Product Categories

- Analytics
- Application Integration
- AR & VR
- Business Productivity
- Compute
- Customer Engagement

COMPUTE

Free Tier 12 MONTHS FREE

[Amazon EC2](#)

750 Hours

per month

Resizable compute capacity in the Cloud

750 hours per month of Linux, RHEL, or SLES

STORAGE

Free Tier 12 MONTHS FREE

[Amazon S3](#)

5 GB

of standard storage

Secure, durable, and scalable object storage infrastructure

5 GB of Standard Storage

DATABASE

Free Tier 12 MONTHS FREE

[Amazon RDS](#)

750 Hours

per month of db.t2.micro database usage (applicable DB engines)

Managed Relational Database Service for MySQL, PostgreSQL, MariaDB, Oracle BYOL, or SQL Server



Create a Kubernetes Cluster

← → C https://console.cloud.google.com/kubernetes/list?project=curious-mender-246305&folder=true&organizationId=true

Aplicaciones Suggested Sites http://www.patric... Wolfram|Alpha Wi... https://itaca.edu.g... Libros Viajes Mac developemnt GPU OpenMp

Google Cloud Platform My First Project

Kubernetes Engine

Clusters

A Kubernetes cluster is a managed group of VM instances for running containerized applications. [Learn more](#)

Filter by label or name

| <input type="checkbox"/> Name ^ | Location | Cluster size | Total cores | Total memory | Notifications | Labels |
|--------------------------------------------------------|---------------|--------------|-------------|--------------|---------------|-------------------------|
| <input checked="" type="checkbox"/> standard-cluster-1 | us-central1-a | 4 | 4 vCPUs | 15.00 GB | | Connect |

Workloads Services & Ingress Applications Configuration Storage Marketplace

Create a Kubernetes Cluster

Google Cloud Platform My First Project ▾

Search icon

Notifications icon

Profile icon

⋮

← Create a Kubernetes cluster

• Your first cluster

Experimenting with Kubernetes Engine, deploying your first application. Affordable choice to get started.

>

○ CPU intensive applications

Web crawling or anything else that requires more CPU.

○ Memory intensive applications

Databases, analytics, things like Hadoop, Spark, ETL or anything else that requires more memory.

○ GPU Accelerated Computing

Machine learning, video transcoding,

'Your first cluster' template

Experimenting with Kubernetes Engine, deploying your first application. Affordable choice to get started.

Some fields can't be changed after the cluster is created. Hover over the help icons to learn more. Dismiss

Name ? seminario-test

Location type ?

Zonal

Regional

Zone ?

Create Cancel Equivalent REST or command line

Key fields for this cluster template

| | |
|----------------------------------|-------------------|
| Cluster version | 1.13.7-g (latest) |
| Machine type | g1-smal |
| Autoscaling | Disabled |
| Stackdriver Logging & Monitoring | Disabled |
| Boot disk size | 30GB |

You will be billed for the 1 node (VM inst in your cluster. [Compute Engine pricing](#)

Create a Kubernetes Cluster (Ready to Use)

Google Cloud Platform My First Project ▾

Clusters

EDIT DELETE ADD NODE POOL DEPLOY CONNECT

| Name | Status | CPU requested | CPU allocatable | Memory requested | Memory allocatable | Storage requested | Storage allocatable |
|---------------------------------------------------|--------|---------------|-----------------|------------------|--------------------|-------------------|---------------------|
| gke-standard-cluster-1-default-pool-3f1d2461-h2d9 | Ready | 464 mCPU | 940 mCPU | 988.78 MB | 2.77 GB | 0 B | 0 B |
| gke-standard-cluster-1-default-pool-3f1d2461-mw6z | Ready | 491 mCPU | 940 mCPU | 377.49 MB | 2.77 GB | 0 B | 0 B |
| gke-standard-cluster-1-default-pool-3f1d2461-tv83 | Ready | 549 mCPU | 940 mCPU | 877.66 MB | 2.77 GB | 0 B | 0 B |
| gke-standard-cluster-1-default-pool-3f1d2461-wxc3 | Ready | 461 mCPU | 940 mCPU | 346.03 MB | 2.77 GB | 0 B | 0 B |

Installing the Gcloud command line tool (Ubuntu)

Crea una variable de entorno para lograr la distribución correcta:

```
export CLOUD_SDK_REPO="cloud-sdk-$(lsb_release -c -s)"
```

Agrega el URI de distribución del SDK de Cloud como una fuente de paquete:

```
echo "deb http://packages.cloud.google.com/apt $CLOUD_SDK_REPO main" | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list
```

Nota: Si tienes instalado [apt-transport-https](#), puedes usar “https” en lugar de “http” en este paso.

Importa la clave pública de Google Cloud:

```
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
```

Sugerencia para la solución de problemas: Si no puedes obtener las últimas actualizaciones debido a una clave vencida, [obtiene el archivo de claves apt-get.gpg más reciente](#).

Actualiza el SDK de Cloud y también instálalo:

```
sudo apt-get update && sudo apt-get install google-cloud-sdk
```

Installing the Gcloud command line tool (mac)

1. Crea un proyecto de Google Cloud Platform, si todavía no tienes uno.
2. Asegúrate de que Python 2.7 esté instalado en tu sistema: `$ python -v`
3. Download the package: `$ wget google-cloud-sdk-231.0.0-darwin-x86_64.tar.gz`
4. Descomprimir el fichero `$ tar -xvf google-cloud-sdk-231.0.0-darwin-x86_64.tar.gz`
5. `./google-cloud-sdk/install.sh`

Playing with Kubernetes and Google Cloud

Exercise: Estimated time 30-40 min

Install the Helm

 Deploy Jupyter (Kubectl)

 Deploy the Zeppelin Spark (using Helm)

 Deploy the Grafana (using Helm)

 Deploy service de grafana (kubectl)

Deployment and Services on Google Cloud

| MacBook-Pro-de-Ronal:zeppelinspark rmuresano\$ kubectl get services,pods,deployments -o wide | | | | | | |
|----------------------------------------------------------------------------------------------|--------------|--------------|----------------|----------------|----------|---------------------------------------------------|
| NAME | TYPE | CLUSTER-IP | EXTERNAL-IP | PORT(S) | AGE | SELECTOR |
| service/grafana | ClusterIP | 10.12.13.88 | <none> | 80/TCP | 2h | app=grafana,release=grafana |
| service/grafana-service-deployment | LoadBalancer | 10.12.10.115 | 35.239.124.107 | 3000:31889/TCP | 2h | app=grafana |
| service/jupyter-service-deployment | LoadBalancer | 10.12.7.79 | 35.232.42.194 | 8888:31454/TCP | 2h | app=jupyter-notebook |
| service/kubernetes | ClusterIP | 10.12.0.1 | <none> | 443/TCP | 2h | <none> |
| service/zeppelinspark-master | ClusterIP | 10.12.14.53 | <none> | 7077/TCP | 1h | component=zeppelinspark-spark-master |
| service/zeppelinspark-webui | LoadBalancer | 10.12.0.109 | 34.67.56.201 | 8080:31303/TCP | 1h | component=zeppelinspark-spark-master |
| service/zeppelinspark-zeppelin | LoadBalancer | 10.12.6.57 | 35.232.19.39 | 8080:31617/TCP | 1h | component=zeppelinspark-zeppelin |
| NAME | READY | STATUS | RESTARTS | AGE | IP | NODE |
| pod/grafana-7fb95c4d47-2z4mn | 1/1 | Running | 0 | 2h | 10.8.3.4 | gke-standard-cluster-1-default-pool-3f1d2461-tv83 |
| pod/jupyter-notebook-865f46bdc6-kx5ff | 1/1 | Running | 0 | 2h | 10.8.2.3 | gke-standard-cluster-1-default-pool-3f1d2461-h2d9 |
| pod/zeppelinspark-master-86457dfbf8-x62g2 | 1/1 | Running | 0 | 1h | 10.8.2.4 | gke-standard-cluster-1-default-pool-3f1d2461-h2d9 |
| pod/zeppelinspark-worker-856486cb64-zhdkz | 1/1 | Running | 0 | 1h | 10.8.3.6 | gke-standard-cluster-1-default-pool-3f1d2461-tv83 |
| pod/zeppelinspark-zeppelin-566947c958-756zb | 1/1 | Running | 0 | 1h | 10.8.3.5 | gke-standard-cluster-1-default-pool-3f1d2461-tv83 |

Kubernetes and Google Cloud

The screenshot shows the Google Cloud Platform Workloads interface. The top navigation bar includes the Google Cloud Platform logo, the project name "My First Project", a search bar, and a menu icon. On the left, a sidebar lists various cloud services with corresponding icons. The main area is titled "Workloads" and contains a brief description: "Workloads are deployable units of computing that can be created and managed in a cluster." Below this is a filter bar with the condition "Is system object : False" and a "Filter workloads" input field. A "Columns" dropdown is also present. The main table displays six entries, each representing a Deployment:

| <input type="checkbox"/> | Name | Status | Type | Pods | Namespace | Cluster |
|--------------------------|------------------------|--------|------------|------|-----------|--------------------|
| <input type="checkbox"/> | grafana | ✓ OK | Deployment | 1/1 | default | standard-cluster-1 |
| <input type="checkbox"/> | jupyter-notebook | ✓ OK | Deployment | 1/1 | default | standard-cluster-1 |
| <input type="checkbox"/> | zeppelinspark-master | ✓ OK | Deployment | 1/1 | default | standard-cluster-1 |
| <input type="checkbox"/> | zeppelinspark-worker | ✓ OK | Deployment | 1/1 | default | standard-cluster-1 |
| <input type="checkbox"/> | zeppelinspark-zeppelin | ✓ OK | Deployment | 1/1 | default | standard-cluster-1 |

Radiatus (A Big Data Analytics Platform)

It is a BDaaS platform over a PaaS which allows data scientist to setup and deploy Big Data Infrastructure

Create an efficient Big Data ecosystem to be executed on Cloud.

Simplify the service creation by automating the technological stack integration.

Focused on the Data Scientist to simplify the deployment procedure.

Data Analytics Architecture





Radiatus (A Big Data Analytics Platform)

The Radiatus platform interface is shown, featuring a dark sidebar on the left and a main dashboard area.

Left Sidebar:

- Main
- Dashboard
- Projects (selected)
- Volumes
- Packages
- Settings
- Users
- Organizations
- Roles
- Components
- Mandatory libraries
- Technologies
- Services
- Scenarios
- Environment vars
- Mail

Main Dashboard Area:

Projects

| Name | Scenario | Service | Users | Actions | | |
|---------------------|----------------------------|---------------------|--------|---------|--|--|
| Jupyter | Exploratory Data Analytics | Jupyter | Ferran | | | |
| Jupyter | Exploratory Data Analytics | Jupyter | Ferran | | | |
| Jupyter Spark Kafka | Real Time Data Analytics | Jupyter Spark Kafka | Ferran | | | |

Buttons: + NEW PROJECT, DELETE PROJECT

Row selection: Jupyter

Bottom navigation: Rows per page: 5, 1-3 of 3, < >

Bottom Left Window: A screenshot of a Jupyter Notebook interface showing a map of New York City.

Bottom Right Window: A screenshot of a Jupyter Notebook interface showing a satellite image of Spain.

Demo

Radiatus (A Big Data Analytics Platform)

Big Data as a Service: A new approach for the SMEs

Muchas gracias por su atención

Ronal Muresano

rmuresano@gmail.com

rmuresano@iti.es



Windows Installation

Install docker windows

<https://github.com/docker/toolbox/releases/tag/v18.09.3>