CS Kubernetes Infrastructure

Pablo García Miranda

IT-CS-CT
• Kubernetes Architecture
• Workloads
  • Pods, ReplicaSets, Deployments
  • Secrets & Environment Variables
• Services: ClusterIP vs NodePort
• Load Balancing: Ingress Controller
• Scheduling
  • Clusters set-up
  • Assigning Pods to Nodes
• Milestones
Kubernetes Architecture

Ctrl Plane - 1,2,...n

kubelet

kubectrl

etcd
controller manager
scheduler

Node 1

Pods
Container Runtime
kubelet
System Services

Node 1

Pods
Container Runtime
kubelet
System Services

Cloud Provider Network Edge
Load Balancer
End Users

platform9.com
Workloads
Workloads: Pods, ReplicaSets, Deployments

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: network-microservice-infoream
spec:
  selector:
    matchLabels:
      app: network-microservice-infoream
  replicas: 2
template:
  metadata:
    labels:
      app: network-microservice-infoream
  spec:
    containers:
      - name: network-microservice-infoream
        image: gitlab-registry.cern.ch/network/monoland/microservices/infoream:master
        imagePullPolicy: Always
        imagePullSecrets:
          - name: gitlab-registry
```
**Workloads:** Pods, ReplicaSets, Deployments

```bash
garciam@aiadm05 ~/kubernetes kubectl apply -f ssl_enabled/infoream_basic.yaml
deployment.apps/network-microservice-infoream created
pgarciam@aiadm05 ~/kubernetes kubectl get all
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>pod/network-microservice-infoream-6bdcd8b9cd-ntcz6</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>7s</td>
</tr>
<tr>
<td>pod/network-microservice-infoream-6bdcd8b9cd-tw8gq</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>7s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
<th>PORT(S)</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>service/kubernetes</td>
<td>ClusterIP</td>
<td>10.254.0.1</td>
<td>&lt;none&gt;</td>
<td>443/TCP</td>
<td>29d</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESIRED</th>
<th>CURRENT</th>
<th>UP-TO-DATE</th>
<th>AVAILABLE</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployment.apps/network-microservice-infoream</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>7s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESIRED</th>
<th>CURRENT</th>
<th>READY</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>replicaset.apps/network-microservice-infoream-6bdcd8b9cd</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>7s</td>
</tr>
</tbody>
</table>
Workloads: Secrets & Environment Variables

```sh
$ kubectl create secret generic key-store \
   --from-file=host.jks=host.jks \
   --from-literal=key_store_password=changeit
```
Services: ClusterIP vs NodePort

```
apiVersion: v1
kind: Service
metadata:
  name: network-microservice-infoream
spec:
  selector:
    app: network-microservice-infoream
  ports:
    - name: https
      port: 443
      targetPort: 8080
  type: ClusterIP
```

```
apiVersion: v1
kind: Service
metadata:
  name: network-microservice-infoream
spec:
  selector:
    app: network-microservice-infoream
  ports:
    - name: https
      port: 443
      targetPort: 8080
      nodePort: 30000
  type: NodePort
```
Load Balancing: Ingress Controller
Load Balancing: Ingress Controller

```yaml
apiVersion: extensions/v1beta1
classkind: Ingress
metadata:
  name: nginx-ingress
  namespace: default
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/ssl-passthrough: "true"
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
spec:
rules:
- host: inforam-network-microservice.cern.ch
  http:
    paths:
    - path: /
      backend:
        serviceName: network-microservice-infoream
        servicePort: 443
```
Scheduling: Clusters set-up
Scheduling: Clusters set-up

```bash
$ kubectl get nodes
NAME                     STATUS    ROLES     AGE     VERSION
network-microservices-multinod-djb6qfro67s-master-0  Ready     master  9m       v1.11.6
network-microservices-multinod-djb6qfro67s-minion-1  Ready     <none>   9m       v1.11.6
network-microservices-multinod-djb6qfro67s-minion-2  Ready     <none>   9m       v1.11.6
network-microservices-multinod-djb6qfro67s-minion-3  Ready     <none>   9m       v1.11.6
network-microservices-multinod-djb6qfro67s-minion-4  Ready     <none>   9m       v1.11.6
network-microservices-multinod-djb6qfro67s-minion-5  Ready     <none>   9m       v1.11.6

$ kubectl label nodes network-microservices-multinod-djb6qfro67s-minion-0 network=IP1 zone=meyrin
$ kubectl label nodes network-microservices-multinod-djb6qfro67s-minion-1 network=IP1 zone=meyrin capability=sms
$ kubectl label nodes network-microservices-multinod-djb6qfro67s-minion-2 network=IP1 zone=prevessin
$ kubectl label nodes network-microservices-multinod-djb6qfro67s-minion-3 network=IP1 zone=prevessin capability=sms
$ kubectl label nodes network-microservices-multinod-djb6qfro67s-minion-4 network=ITS zone=cern-a
$ kubectl label nodes network-microservices-multinod-djb6qfro67s-minion-5 network=ITS zone=cern-b
```
Scheduling: Pod and Node Affinity
Scheduling: Pod and Node Affinity
Scheduling: Pod and Node Affinity

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: network-microservice-inforem
spec:
  selector:
    matchLabels:
      app: network-microservice-inforem
replicas: 2
template:
  metadata:
    labels:
      app: network-microservice-inforem
spec:
  containers:
  ...
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: network
            operator: In
            values:
            - ITS
      podAntiAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - network-microservice-csrest
topologyKey: zone
```
Milestones

1. Cluster creation and hostname-based routing with SSL pass-through
2. Assigning pods to nodes based on the requirements for each microservice
3. Demonstrate mounting/sharing of USB ports inside pods
4. Integration with Gitlab
5. Monitoring and alarms
Milestones

6. Secret management
7. High availability via DNS load-balancing
8. Dashboard with security
9. Migration of our microservices to the new infrastructure
10. Auto-scaling of deployments and of cluster*
11. Automatic host certificate renewal*

* Improvements
Questions