[] | *image:assets/images/logo.png*

Jarvis

Nicolas Lamirault

Version 0.3.0, 12/20/2017

Table of contents

1. Identification 1

2. Introduction 2

2.1. Objet du document 2

2.2. Licence 2

3. System 3

3.1. Operating System 3

3.2. Cluster 3

3.3. Components 4

3.4. Administration 6

4. Services 8

4.1. Namespaces 8

4.2. Storage 8

4.3. Hypriot 11

4.4. Monitoring 11

4.5. Metrics 12

4.6. DNS 12

Annexes 13

Bibliographie 13

Figures 13

Glossaire 13

Chapter 1. Identification

Table 1. Identification

Référence	FR/BDX/Lam
Date	12/20/2017
Version	0.3.0
Classification	Open Source

Table 2. Suivi de révisions

Version	Date	Auteur	Modification
0.1.0	12/20/2017	Nicolas Lamirault <nicolas.lamirault@gmail.com>	Initialize

Chapter 2. Introduction

2.1. Objet du document

This is the user guide for Jarvis.

ℹ the last version of this document is on Github : https://github.com/zeiot/jarvis/docs

2.2. Licence

Chapter 3. System

3.1. Operating System

Install HypriotOS onto the sdcard:

```
$ sdcard/jarvis_os_2.sh jarvis myssid mywifipassword Linux
```

Generate a new machineid on each host, see : https://github.com/hypriot/image-builder-rpi/issues/167

3.2. Cluster

Ansible is used to configure the cluster.

Go into the **ansible** directory to manage the cluster.

3.2.1. Creation

Create an **inventory** file like that:

```
[master]
<master_ip_address>     ansible_connection=ssh     ansible_user=pirate

[nodes]
<node1_ip_address>      ansible_connection=ssh     ansible_user=pirate
<node2_ip_address>      ansible_connection=ssh     ansible_user=pirate
<node3_ip_address>      ansible_connection=ssh     ansible_user=pirate
```

You could now check communications with hosts:

```
$ ansible all -m ping -i inventory
```

Display some informations :

```
$ ansible-playbook -i inventory debug.yml
```

Initialize hosts

```
$ ansible-playbook -i inventory bootstrap.yml
```

Then setup the cluster :

```
$ ansible-playbook -i inventory site.yml
```

After that, you could check Kubernetes cluster status :

```
$ ansible-playbook -i inventory k8s.yml
```

3.2.2. Update

```
$ ansible-playbook -i inventory update.yml
```

3.2.3. Destruction

```
$ ansible-playbook -i inventory destroy.yml
```

3.3. Components

3.3.1. Kubernetes Dashboard

You could install the official Kubernetes Dashboard :

```
$ kubectl apply -f k8s/dashboard –record
$ kubectl describe services kubernetes-dashboard –namespace=kube-system
```

3.3.2. DNS

You could replace the kube-dns default installation with CoreDNS :

```
$ kubectl apply -f k8s/coredns –record
$ kubectl describe services kube-dns –namespace=kube-system
Name:              kube-dns
Namespace:         kube-system
Labels:            k8s-app=coredns
                   kubernetes.io/cluster-service=true
                   kubernetes.io/name=CoreDNS
Annotations:       kubectl.kubernetes.io/last-applied-
configuration={"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"labels":{"k8s-
app":"coredns","kubernetes.io/cluster-service":"true","kubernetes.io/na...
Selector:          k8s-app=coredns
Type:              ClusterIP
IP:                10.96.0.10
Port:              dns  53/UDP
TargetPort:        53/UDP
Endpoints:         10.36.0.5:53,10.44.0.2:53
Port:              dns-tcp  53/TCP
TargetPort:        53/TCP
Endpoints:         10.36.0.5:53,10.44.0.2:53
Port:              metrics  9153/TCP
TargetPort:        9153/TCP
Endpoints:         10.36.0.5:9153,10.44.0.2:9153
Session Affinity:  None
Events:            <none>
```

3.3.3. Heapster

Heapster enables Container Cluster Monitoring and Performance Analysis for Kubernetes :

```
$ kubectl apply -f k8s/heapster –record
```

3.3.4. Ingress Controllers

Nginx is used as the default Ingress Controller :

```
$ kubectl apply  -f ingress/ingress-controller-rbac.yaml --record
$ kubectl apply  -f ingress/ingress-default-backend.yaml --record
$ kubectl apply  -f ingress/nginx/ --record
```

3.3.5. Status

After a few minutes, check the cluster informations :

```
$ kubectl cluster-info
Kubernetes master is running at https://192.168.1.36:6443
Heapster is running at https://192.168.1.36:6443/api/v1/namespaces/kube-system/services/heapster/proxy
CoreDNS is running at https://192.168.1.36:6443/api/v1/namespaces/kube-system/services/kube-dns/proxy
```

How cluster’s nodes are :

```
$ kubectl get nodes
NAME          STATUS    ROLES     AGE       VERSION
jarvis-master   Ready     master    3h        v1.8.5
jarvis-node1    Ready     <none>    3h        v1.8.5
jarvis-node2    Ready     <none>    3h        v1.8.5
```

You could see also nodes metrics (with heapster) :

```
$ kubectl top nodes
NAME          CPU(cores)   CPU%      MEMORY(bytes)   MEMORY%
jarvis-master   631m         15%       639Mi           83%
jarvis-node2    216m         5%        485Mi           63%
jarvis-node1    254m         6%        531Mi           69%
```

3.4. Administration

3.4.1. Security

TODO

3.4.2. Quotas

TODO

3.4.3. Backup

TODO

3.4.4. Validation

TODO

Chapter 4. Services

4.1. Namespaces

Create the necessary namespaces :

```
$ $ kubectl apply -f k8s/namespaces/ --record
namespace "logs" created
namespace "metrics" created
namespace "monitoring" created
namespace "hypriot" created
$ kubectl get ns
NAME          STATUS    AGE
default       Active    54d
hypriot       Active    54d
kube-public   Active    54d
kube-system   Active    54d
logs          Active    48s
metrics       Active    47s
monitoring    Active    47s
```

4.2. Storage

We use a NFS server for storage. So we will use the NFS client provisionner :

```
$ kubectl create -f k8s/nfs-client-provisioner/
```

After that, creates a POD to check the NFS access :

```
$ kubectl create -f k8s/nfs-client-provisioner/nfs-pvc-test.yaml
$ kubectl create -f k8s/nfs-client-provisioner/nfs-pod-test.yaml
$ kubectl get pods
NAME                                       READY     STATUS              RESTARTS   AGE
hypriot-2682716425-bldnh                   1/1       Running             0          28d
nfs-client-provisioner-3721834868-6phzj    1/1       Running             0          1h
test-pod                                   0/1       ContainerCreating   0          4s
```

```
$ kubectl describe pod test-pod
Name:           test-pod
Namespace:      default
Node:           jarvis-node2/192.168.1.26
Start Time:     Sat, 23 Sep 2017 13:28:09 +0200
Labels:         <none>
Annotations:    <none>
Status:         Succeeded
IP:             10.36.0.2
Containers:
  test-pod:
    Container ID:
docker://351a925cad33164929975010fc128f5a2590f7d941849170caef0ffa6f56ef7c
    Image:              hypriot/armhf-busybox:1.24
    Image ID:           docker-pullable://hypriot/armhf-
busybox@sha256:746423cb45f66db032f2138f7459a26051dadb1c5101727bd8abb847c6f90b7f
    Port:               <none>
    Command:
      /bin/sh
    Args:
      -c
      touch /mnt/SUCCESS && exit 0 || exit 1
    State:              Terminated
      Reason:           Completed
      Exit Code:        0
      Started:          Sat, 23 Sep 2017 13:28:19 +0200
      Finished:         Sat, 23 Sep 2017 13:28:19 +0200
    Ready:              False
    Restart Count:      0
    Environment:        <none>
    Mounts:
      /mnt from nfs-pvc (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-5chdh (ro)
Conditions:
  Type          Status
  Initialized   True
  Ready         False
  PodScheduled  True
Volumes:
  nfs-pvc:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same
namespace)
    ClaimName:  test-claim
    ReadOnly:   false
  default-token-5chdh:
    Type:       Secret (a volume populated by a Secret)
    SecretName: default-token-5chdh
    Optional:   false
QoS Class:      BestEffort
Node-Selectors: <none>
Tolerations:    node.alpha.kubernetes.io/notReady:NoExecute for 300s
                node.alpha.kubernetes.io/unreachable:NoExecute for 300s
vents:
  FirstSeen     LastSeen        Count   From                    SubObjectPath                   Type            Reason
Message
  ---------     --------        -----   ----                    -------------                   --------        ------
-------
  14s           14s             1       default-scheduler                                       Normal          Scheduled
Successfully assigned t
est-pod to jarvis-node2
  14s           14s             1       kubelet, jarvis-node2                                   Normal
SuccessfulMountVolume   MountVolume.SetUp succe
eded for volume "default-token-5chdh"
  14s           14s             1       kubelet, jarvis-node2                                   Normal
SuccessfulMountVolume   MountVolume.SetUp succe
eded for volume "pvc-26194824-a052-11e7-9c1e-b827eb13a985"
  12s           12s             1       kubelet, jarvis-node2   spec.containers{test-pod}       Normal          Pulling
pulling image "hypriot/
armhf-busybox:1.24"
  5s            5s              1       kubelet, jarvis-node2   spec.containers{test-pod}       Normal          Pulled
Successfully pulled ima
ge "hypriot/armhf-busybox:1.24"
  4s            4s              1       kubelet, jarvis-node2   spec.containers{test-pod}       Normal          Created
Created container
  4s            4s              1       kubelet, jarvis-node2   spec.containers{test-pod}       Normal          Started
```

Check on the NFS server, if some data are written.

4.3. Hypriot

A namespae **hypriot** exists is present. A single service is available which display a logo of the Hypriot project.

```
$ kubectl apply -f k8s/hypriot -n hypriot
```

You could also create an user **hypriot** to manage this namespace :

```
$ ./k8s/scripts/kube-add-user.sh 192.168.1.36 hypriot hypriot

Kubernetes master: 192.168.1.36
Generating RSA private key, 2048 bit long modulus
.......................................................................+++
....+++
e is 65537 (0x10001)
Signature ok
subject=/CN=hypriot/O=jarvis
Getting CA Private Key
Cluster "admin” set.
User "hypriot” set.
Context "hypriot" modified.
rolebinding "hypriot-admin” created
rolebinding "hypriot-admin” labeled
Switched to context "hypriot”.

$ kubectl –kubeconfig=/tmp/hypriot-config get deployment
NAME     DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
hypriot   1        1        1          1         27m

$ kubectl –kubeconfig=/tmp/hypriot-config get ingress
NAME            HOSTS          ADDRESS          PORTS     AGE
hypriot-ingress   hypriot.jarvis   192.168.1.20,...   80       27m
```

4.4. Monitoring

A namespace **monitoring** is present for monitoring tools. You could install Prometheus and some exporters

```
$ kubectl apply -f k8s/prometheus/ -n monitoring --record
```

The kube state metrics component :

```
$ kubectl apply -f k8s/kube-state-metrics/ -n monitoring --record
```

4.5. Metrics

For some metrics, you could also install InfluxDB:

```
$ kubectl apply -f k8s/influxdb/ -n metrics --record
```

4.6. DNS

Annexes

Bibliographie

- [Ansible_doc] http://docs.ansible.com/ansible/latest/index.html

Figures

Glossaire

- CNCF: Cloud Native Computing Foundation
- Kubernetes : Containeurs orchestrator (manage by the CNCF)