Kubernetes Deployment On Bare-metal Ubuntu Nodes

This document describes how to deploy kubernetes v1.0.1 on ubuntu nodes for WSO2 App Factory deployment

1. Install Docker
   Please follow [1] to install docker based on your linux distribution

2. Install zip, unzip, tree and git
   ```
   > sudo apt-get install zip
   > sudo apt-get install unzip
   > sudo apt-get install tree
   > sudo apt-get install git
   ```

3. Deploy Kubernetes(k8s) 1.0.1. These are the basic steps to deploy k8s. You can find more information on [2]

   a. Get k8s
   ```
   > mkdir ~/Kubernetes
   > cd ~/Kubernetes
   > git clone https://github.com/kubernetes/kubernetes.git
   > cd kubernetes
   > git checkout v1.0.1
   > cd cluster/ubuntu
   ```

   b. Then run `build.sh`, this will download all the needed binaries into `./binaries`.
   ```
   > ./build.sh
   ```

   c. Then open the **config-default.sh** file and configure the kubernetes cluster based on your environment[3]. (Please replace "**<USER_NAME>** and **<ETH0_IP>** with a existing username and eth0 ip address

   ```
   export nodes=${nodes:-"<USER_NAME>@<ETH0_IP>"}
   roles=${roles:-"ai"}
   NUM_MINIONS=${NUM_MINIONS:-1}
   ```

Note:
In the POC setup(10.145.97.121), we have used a single machine which acts as both master and minion and node variable is as below.

export nodes=${nodes:-"**wso2@192.168.10.19**"}

Here 192.168.10.19 is the “eth0” address of the node 10.145.97.121 and wso2 is the user(During the deployment, I have created sudoer as a “wso2” and used that user in the Kubernetes node 10.145.97.121).

d. After all the above variable being set correctly. We can use below command in `cluster/` directory to bring up the whole cluster.

```
> cd ~/Kubernetes/kubernetes/cluster
> export KUBERNETES_PROVIDER=ubuntu
> ./kube-up.sh
```

If all things goes right, you will see the below message from console `Cluster validation succeeded` indicating the k8s is up.

Now you can access the k8s cluster dashboard using
http://<PUBLIC_IP>:8080/static/app/#/dashboard/
ex: http://10.145.97.121:8080/static/app/#/dashboard/

You can also use `kubectl` command to see if the newly created k8s is working correctly. The `kubectl` binary is under the `cluster/ubuntu/binaries` directory. You can move it into your PATH. Then you can use the below command smoothly.

For example, use `$ kubectl get nodes` to see if all your minion nodes are in ready status. It may take some time for the minions ready to use like below.

```
NAME            LABELS                                  STATUS
192.168.10.19   kubernetes.io/hostname=192.168.10.19   Ready
```

After the previous parts, you will have a working k8s cluster. Next you have to deploy addons like dns onto the existing cluster(The configuration of dns is configured in cluster/ubuntu/config-default.sh[4]).

After all the required variables have been set as in [4], just type the below command to deploy the add ons.

```
> cd ~/Kubernetes/kubernetes/cluster/ubuntu
> export KUBERNETES_PROVIDER=ubuntu
> ./deployAddons.sh
```

After some time, you can use `$ kubectl get pods --namespace=kube-system` to see the dns pod is running in the cluster.

```
NAME                READY     REASON    RESTARTS   AGE
kube-dns-v4-nsxmr   2/3       Running   0          2d
```

4. When setting up kubernetes cluster we need to load docker images specific to appfactory. To do that
    a. First get the required additional artifacts (I have created a zip file with these additional artifacts in 10.145.97.116 node and you can get and configure them using below commands)

    ```
    > cd ~
    > mkdir Sources
    > cd ~/Sources
    > wget http://10.145.97.116/af_artifacts.zip
    > unzip af_artifacts.zip -d .
    ```

    b. Get the WSO2 App Factory source code

    ```
    > cd ~/Sources
    > git clone https://github.com/wso2/product-af.git
    > cd product-af
    > git checkout 2.2.0-M5
    > mvn clean install -Dmaven.test.skip=true
    ```

c. Copy required jars to build docker images

```
> cd modules/cartridges/docker
```

Open **`setup_docker.sh`** script and Update the **CODE_PATH** and **AF_ARTIFACTS_HOME** variables

- **CODE_PATH**
  absolute path of the wso2appfactory source code
  ex: CODE_PATH=/home/wso2/Sources/product-af

- **AF_ARTIFACTS_HOME** :
  absolute path of the required additional artifacts directory
  ex: AF_ARTIFACTS_HOME=/home/wso2/Sources/af_artifacts

Execute setup_docker.sh tol copy all required jars

```
> sudo bash setup_docker.sh
```

e. Building the base docker image.
First go the `base-image` directory.

```
> cd ~/Sources/product-af/modules/cartridges/docker/
> cd base-image
```

Make sure following artifacts are there in the packages directory

```
> ls -1 packages/
```

apache-stratos-python-cartridge-agent-4.1.3.zip
jdk-7u60-linux-x64.tar.gz
wso2ppaas-configurator-4.1.3.zip

Run build.sh

```
> sudo ./build.sh
```

Now if you run docker images command, you can see wso2/base-image 4.1.3 image in the docker repository

```
> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
wso2/base-image	**4.1.3**	9cf18436e1a1	43 hours ago	703 MB
gcr.io/google_containers/kube2sky	1.1	aabf05df7856	8 months ago	19.15 MB
gcr.io/google_containers/etcd	2.0.9	d5d77b0a1030	10 months ago	12.82 MB
gcr.io/google_containers/pause	0.8.0	Bf595365a558	11 months ago	241.7 KB

gcr.io/google_containers/skydns	2015-03-11-001	2af62c60205d	11 months ago	8.814 MB
debian	7.7	6d129b37ec3b	14 months ago	84.99 MB

e. Building appfactory wso2as-5.2.1 docker image.
First go the `wso2as-5.2.1/templates-module` directory.

```
> cd ~/Sources/product-af/modules/cartridges/docker/
> cd wso2as-5.2.1/templates-module
```

Make sure required jars are there as in **'###App Factory' section** of the **README.md**.

```
> tree files/
```

build the templates-module

```
> mvn clean install
```

Copy the build artifact to the wso2as-5.2.1/packages directory

```
> cp target/wso2as-5.2.1-template-module-4.1.3.zip ../packages/
```

Make sure following artifacts are there in the packages directory of the wso2as-5.2.1

```
> cd ../
> ls -1 packages/
```

jdk-7u60-linux-x64.tar.gz
mysql-connector-java-5.1.27-bin.jar
wso2as-5.2.1-template-module-4.1.3.zip
wso2as-5.2.1.zip

Run build.sh

```
> sudo ./build.sh
```

Now if you run docker images command, you can see wso2/as 5.2.1 image in the docker repository

```
> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
wso2/as	**5.2.1**	fe9f02d29ed5	43 hours ago	1.558 GB
wso2/base-image	4.1.3	9cf18436e1a1	43 hours ago	703 MB
gcr.io/google_containers/kube2sky	1.1	aabf05df7856	8 months ago	19.15 MB
gcr.io/google_containers/etcd	2.0.9	d5d77b0a1030	10 months ago	12.82 MB
gcr.io/google_containers/pause	0.8.0	Bf595365a558	11 months ago	241.7 KB
gcr.io/google_containers/skydns	2015-03-11-001	2af62c60205d	11 months ago	8.814 MB
debian	7.7	6d129b37ec3b	14 months ago	84.99 MB

Now Kubernetes setup is done!. Next step is to deploy the WSO2 App Factory setup.

References

[1] https://docs.docker.com/engine/installation/linux/ubuntulinux/
[2] https://github.com/kubernetes/kubernetes/blob/v1.0.1/docs/getting-started-guides/ubuntu.md
[3] https://github.com/kubernetes/kubernetes/blob/v1.0.1/docs/getting-started-guides/ubuntu.md#user-content-configure-and-start-the-kubernetes-cluster
[4] https://github.com/kubernetes/kubernetes/blob/v1.0.1/docs/getting-started-guides/ubuntu.md#user-content-deploy-addons