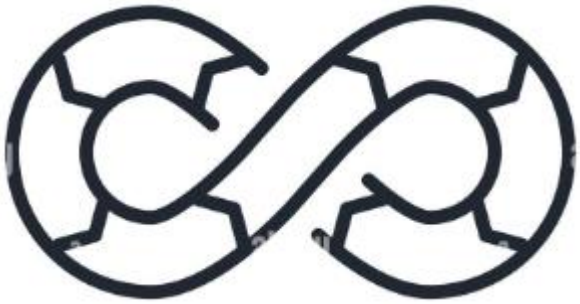


DevOps Assignment

Arash Foroughi

In this documentation, you will find a procedure for setting up a CI/CD pipeline, deploying a Dockerized application on Kubernetes, and ensuring its successful operation on some servers.

Document Statistics

Type of Information	Document Data
Title	DevOps-Assignment-Arash-Foroughi
Document Revision #	1.0
Last Date Document Update	09-06-2023
Total Number of Pages	10
Document Filename	DevOps_Assignment_Arash_Foroughi_V1.0.docx
Document Author	Arash Foroughi
Revision Author	-
Change Records	-

1. Setup a High Available Kubernetes Cluster Using Kubeadm:

Following this procedure, we are going to deploy and setup a High Available Kubernetes Cluster using kubeadm on Ubuntu Server 22.04 LTS.

We will consider you are going to deploy kubernetes cluster on some servers or EC2 instances and not using EKS service which it will getting up the cluster very easily.

So, for this scenario we need 4 servers in the minimum, one of the instances is Control-Plane and the three others are working as Worker nodes. For the HA Clustering, later we can define two of the workers as control-plane role also. Number of control-plane nodes in Kubernetes must be odd, because of the 2N+1 clustering type which designed on Kubernetes.

Role	FQDN	IP	OS	CPU	RAM
Master-1	kmaster1.example.com	10.0.2.4	Ubuntu 22.04	4	4G
Worker-1	kworker1.example.com	10.0.2.5	Ubuntu 22.04	2	2G
Worker-2	kworker2.example.com	10.0.2.6	Ubuntu 22.04	2	2G
Worker-3	kworker3.example.com	10.0.2.7	Ubuntu 22.04	2	2G

2. Setup Load Balancer Node (Needed in Multi-Master Architecture):

If you configured Kubernetes in Multi-Master design, it needs another VM except above ones to install HAproxy on it:

```
root@loadbalancer:~# apt update
root@loadbalancer:~# apt install haproxy -y
```

For configuring HAproxy, append below lines to /etc/haproxy/haproxy.cfg:

```
root@loadbalancer:~# vi /etc/haproxy/haproxt.cfg
```

```
frontend kubernetes-frontend
bind 192.168.44.153:6443
mode tcp
option tcplog
default_backend kubernetes-backend
backend kubernetes-backend
mode tcp
option tcp-check
balance roundrobin
server kmaster1 192.168.44.154:6443 check fall 3 rise 2
```

```
server kmaster2 192.168.44.155:6443 check fall 3 rise 2
```

Restart HAproxy service:

```
root@loadbalancer:~# ssh-copy-id ubuntu@10.0.2.4
root@loadbalancer:~# systemctl restart haproxy
```

HAproxy load balancer server has been configured successfully, now in next step needs to setup Kubernetes on all Master/Worker nodes.

3. Setup Ansible Host Inventory and Its Configurations:

For installing Kubernetes, there are some ways to set it up. We decided to use Ansible and the playbooks which attached to this GitHub to install and initiate Kubernetes on our nodes.

We consider Kmaster-1 as **Ansible Controller** node also and then install Ansible:

```
ubuntu@kmaster-1:~$ sudo apt install ansible -y
ubuntu@kmaster-1:~$ ansible --version
```

Then start to send the public key of Ansible Controller to all nodes:

```
ubuntu@kmaster-1:~$ ssh-keygen
ubuntu@kmaster-1:~$ ssh-copy-id ubuntu@10.0.2.4
ubuntu@kmaster-1:~$ ssh-copy-id ubuntu@10.0.2.5
ubuntu@kmaster-1:~$ ssh-copy-id ubuntu@10.0.2.6
ubuntu@kmaster-1:~$ ssh-copy-id ubuntu@10.0.2.7
```

Prepare the Ansible Host Inventory file per below information:

```
ubuntu@kmaster-1:~# sudo mkdir /etc/ansible
ubuntu@kmaster-1:~# sudo vim /etc/ansible/hosts
[all]
kmaster1 ansible_host=10.0.2.4 ansible_connection=ssh ansible_user=ubuntu
kmaster2 ansible_host=10.0.2.5 ansible_connection=ssh ansible_user=ubuntu
kworker1 ansible_host=10.0.2.6 ansible_connection=ssh ansible_user=ubuntu
kworker2 ansible_host=10.0.2.7 ansible_connection=ssh ansible_user=ubuntu
[masters]
kmaster1 ansible_host=10.0.2.4
[workers]
kworker1 ansible_host=10.0.2.4 ansible_connection=ssh ansible_user=ubuntu
kworker2 ansible_host=10.0.2.5 ansible_connection=ssh ansible_user=ubuntu
```

```
kworker3 ansible_host=10.0.2.6 ansible_connection=ssh ansible_user=ububue
```

For verification which the Ansible controller is working properly, use Ping Module to test the connectivity between nodes:

```
ubuntu@kmaster-1:~# ansible all -m ping
ubuntu@kmaster-1:~# ansible masters -m ping
ubuntu@kmaster-1:~# ansible workers -m ping
```

4. Install and Configuration of Kubernetes Cluster:

Login to kmaster-1 node, find Setup_K8s_Cluster_Playbook.yaml from the package and run it with below command:

```
ubuntu@kmaster-1:~# ansible-playbook K8s_Installation_Playbook.yaml
```

After finishing the previous Ansible playbook, it means you have successfully installed Docker and

Kubernetes prerequisites on all Master/Worker nodes successfully.

Now it's time to setup your kubernetes cluster and specify role of each node in the cluster with running below playbook again from kmaster-1 node which has also Ansible_Controller role in this scenario:

```
ubuntu@kmaster-1:~# ansible-playbook K8s_Initiation_Playbook.yaml
```

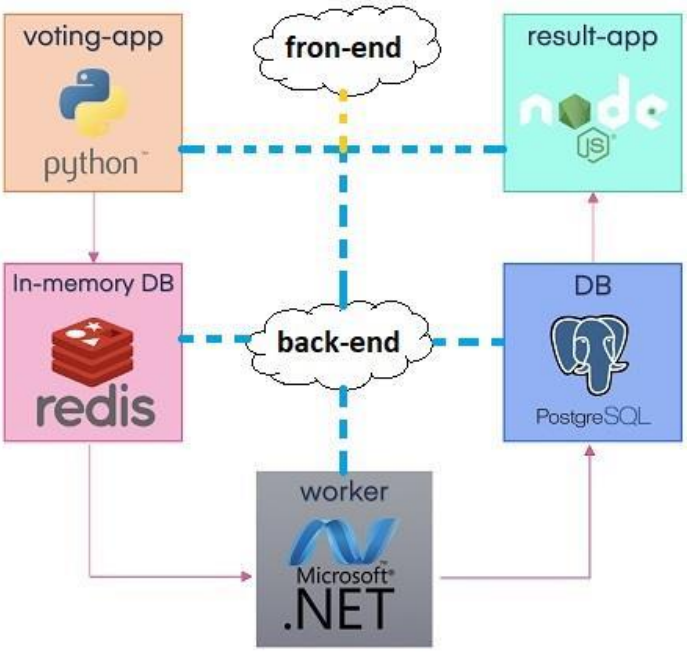
The Kubernetes cluster now has been created successfully. You can always check the status of nodes and pods with below commands:

```
ubuntu@kmaster-1:~# kubectl get nodes -o wide
ubuntu@kmaster-1:~# kubectl get pods -A -o wide
```

The Kubernetes cluster now has been created successfully. You can always check the status of nodes and pods with below commands:

```
ubuntu@kmaster-1:~# kubectl get nodes -o wide
ubuntu@kmaster-1:~# kubectl get pods -A -o wide
```

5. Setup a Voting Application as sample scenario:



Now it's time to deploy a scenario as pods on our Kubernetes cluster with below procedure. In the picture, you can find a sample voting app, which it contains 2 web services, one for voting and one for checking the result. In this scenario when someone votes, the vote will be written in a Redis DB and a .Net worker as backend will write it in a physical DB which it is Postgresql and at the end, result application will show read from this DB and show the result.

For starting up these services, you just need to run the pipeline from the GitLab repository, all of the procedures for building and running microservices will be done automatically by running the pipeline!

6. Adding Infrastructure & Services to the Monitoring System:

Now after running services on Kubernetes cluster, it's time to monitor services and their environment to prevent emergency issues.

For monitoring solution, we recommend to use Prometheus & Grafana which they have many exporters and agents for monitoring of nodes, databases, pods and Kubernetes clusters.

1. First, it needs to run Prometheus stack on Kubernetes cluster. For this we will go on kmaster-1 node, install **helm** and run Prometheus and all of its dependencies using **helm** charts from **Artifacthub** website.

   Run below commands to install helm on kubernetes master:

```
# curl https://baltocdn.com/helm/signing.asc | apt-key add -
# apt install apt-transport-https -y
```

```
# echo "deb https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list

# apt update; apt install helm -y
```

2. Then, needs to add Prometheus-stack repo to helm and install them using helm on kubernetes cluster:

```
# helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

# helm repo update

# helm install prometheus prometheus-community/kube-prometheus-stack
```

And now we have everything needed for Prometheus and Grafana on the Kubernetes cluster. If we check list of pods, deployments and daemonsets there are some objects which needed for monitoring.

In Prometheus.yaml file we need to add some scrape configurations for introducing exporters to our Prometheus. The template format of the scrape config is like below and we need to add below exporters on its configuration:

```
- job_name: 'kube-state-metrics'
  static_configs:
    - targets: [10.0.2.4:8080']


- job_name: 'node-exporter-kmaster-1'
  static_configs:
    - targets: [10.0.2.4:9100']


- job_name: 'node-exporter-kworker-1'
  static_configs:
    - targets: [10.0.2.5:9100]


- job_name: 'node-exporter-kworker-2'
  static_configs:
    - targets: [10.0.2.6:9100]


- job_name: 'node-exporter-kworker-3'
  static_configs:
    - targets: [10.0.2.7:9100]
```

```
- job_name: 'postgres-exporter'
  static_configs:
    - targets: [10.0.2.4:8080']

- job_name: 'redis-exporter'
  static_configs:
    - targets: [10.0.2.4:8080']

- job_name: 'blackbox-exporter'
  static_configs:
    - targets: [10.0.2.4:8080']
```

After adding above parameters to Prometheus configuration, it's time to integrate Prometheus with Grafana as Data Source and add some dashboards to it.

1) Click on the "**cogwheel**" in the sidebar to open the Configuration menu.
2) Click on "**Data Sources**".
3) Click on "**Add data source**".
4) Select "**Prometheus**" as the type.
5) Set the appropriate Prometheus server URL (for example, **http://localhost:9090/**)
6) Adjust other data source settings as desired (for example, choosing **right Access** method).
7) Click "**Save & Test**" to save the new data source.

Then, for adding some dashboards need to do as below procedure:

A. **Grafana.com** maintains a collection of shared dashboards which can be downloaded and used with standalone instances of Grafana. Use the **Grafana.com "Filter"** option to browse dashboards for the "Prometheus" data source only.

B. You must currently manually edit the downloaded **JSON** files and correct the data source: **entries to reflect the Grafana data source name which you chose for your Prometheus server**. Use the "Dashboards" → "Home" → "Import" option to import the edited dashboard file into your Grafana install.

After selecting some fit dashboards for our services and infrastructure, we can go and check the monitoring dashboards on Grafana!

For setting of alert manager, I recommend to use Grafana Alert Manager and for each dashboards we need alerting we can select its panel > edit and then write a alert rule for that panel.

And also, another which should be considered is about the Alert Manager Contact Points. There are so many ways to send the alerts to the people like sending Emails, sending messages in Slack, Telegram, PagerDuty and other messaging tools.

In my best practices, companies prefer to use Emails and Slack for getting notifications together. For setting up emails, first need to go to /etc/grafana/grafana.ini file and set the smtp configuration for sending emails from Grafana Notification Center.

And for setting up Grafana with any messaging tools like Slack, needs to get the webhook from that software and set that to Grafana contact point and then it can send alerts to that messenger software.

Thanks for Your Attention!

Please let me know in case of any questions with below Email:

arashforoughi@outlook.com