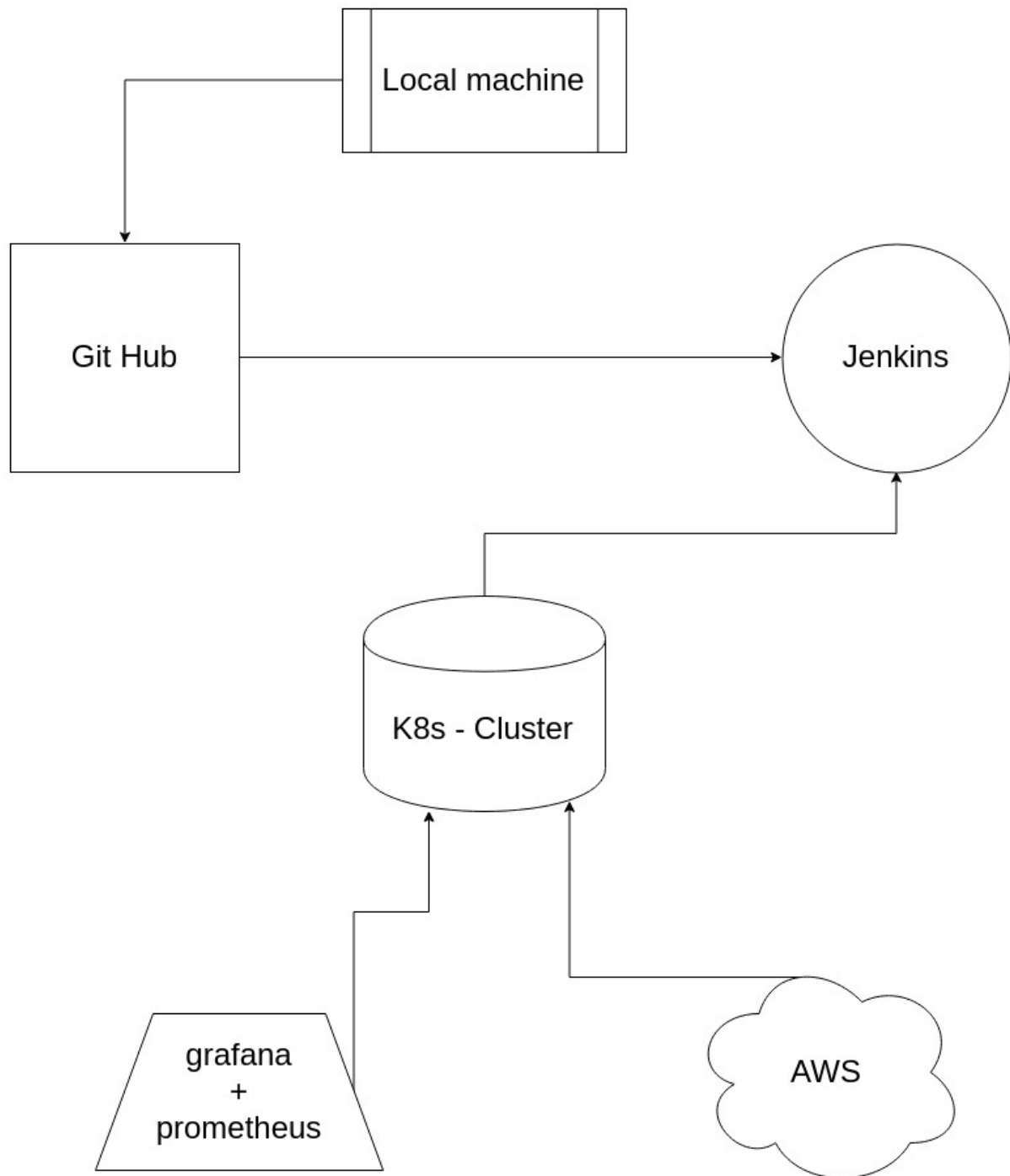


Implement CI/CD

Diagram



Tools

In This task i will use the following tools.

- Git
- GitHub
- Jenkins
- Docker
- Kubernetes
- Grafana and Prometheus
- AWS S3 Service.

Scenario

1. In Local machine I will install Git client and i will add my repo in GitHub.

So i can update my code in my local machine and push it to my GitHub account and it's refer to (SCM).

2.I will install Jenkins server to trigger the updates in My GitHub repo by "Web Hook" and Deploy the updates to containers and it's refer to (CI/CD).

3.I will install Kubernetes Cluster as a platform to deploy containers and orchestrate my action in these containers, and it's refer to Containerization.

4.I will Install Grafana and Prometheus to monitor the Kubernetes cluster and containers that running on it, and it's refer to monitoring.

5.I will Use my AWS account to create S3 bucket that will contain APP backup.

Note : I will user Wordpress App to deploy the CI/CD cycle.

Requirements

1. Your Local machine, any OS.
2. Github Free Account
3. Three public servers one for Jenkins and two for k8s Cluster (master - node)
4. Docker Hub Free Account
5. AWS Free Tier Account.

Now i will give you brief about Tools that we will use.

1. Source Control Management (Git)

SCM is a vital part of any CI/CD pipeline, and in many ways it is the first step.

Anyone who wants to build or maintain a CI/CD pipeline needs a basic understanding of SCM.

Git is the most widely used SCM today.

Before you can begin using git, you need to install and configure it.

Git installation instructions: <https://git-scm.com/downloads>

GitHub ssh authentication setup documentation:

<https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/>

<https://help.github.com/articles/adding-a-new-ssh-key-to-your-github-account/>

2. Continuous Integration / Continuous Delivery (Jenkins)

Continuous Integration is an essential component of a full CI/CD Pipeline.

I used in this task: Jenkins to implement CI/CD using Jenkins.

For more information on Jenkins, check out the official Jenkins site:

<https://jenkins.io/>

3.Containerization (Docker and Kubernetes)

To start thinking about deploying frequently, we need to move to the simplest scalable way for that and it's a containers.

Docker is the most usable engine to deploy the containers.

But in order to provide stability in the context of this high rate of change, we need robust tools for managing our infrastructure.

This introduces orchestration and how we can use it in the context of a CI/CD Pipeline.

It also about Kubernetes, the orchestration tool we will be using.

4. Monitoring (Prometheus and Grafana)

In order to manage the complexity of a system that is able to support a CI/CD process, it is critical to have visibility into that system. Monitoring tools provide access to important data that can help you maintain stability in the context of a high rate of change.

I will use Prometheus and Grafana, the two monitoring tools

For more information about these monitoring tools, check out their official sites:

Prometheus - <https://prometheus.io/>

Grafana - <https://grafana.com/>

4. Backup Solution (AWS S3 Service)

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance.

This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics.

1. Install Git in your Local Machine

- Installing Git :

Git installation instructions for your system :

<https://git-scm.com/downloads>

For example On a Linux Redhat distro that uses rpm :

sudo yum -y install git

- Configuring Your Name and Email :

After installing git, you need to set the name and email that will be associated with your commits.

When using GitHub, it is best to use an email address that is associated with your GitHub account.

git config --global user.name "<your name>"

git config --global user.email <your email>

- Create Free GitHub Account :

<https://github.com>

kops-karimfadiConfiguring Your Name and Email :

An easy way to authenticate with a remote git server like GitHub is to use an ssh key pair.

On a CentOS environment, you can generate an ssh key pair like this:
`ssh-keygen -t rsa -b 4096`

This command will prompt you for several things.

These can be left as their defaults, though it is good practice to enter a passphrase. If you do use a passphrase, make sure you remember what it is.

After generating the key pair, copy the contents of `~/.ssh/id_rsa.pub`. On GitHub.com, click your profile image at the top right, then click "settings," "SSH and GPG Keys." Click "new ssh key," enter a name and paste the contents of `id_rsa.pub` into the key field, then submit the form.

- Fork My Repo to your Account :

My Repo Link : <https://github.com/karimfadi/implement-wordpress-cicd>

Now you have a copy from my repo to your account with your permission.

2. Launch Your Jenkins Server and Integrate it with Your GitHub Repo.

- Create Your public VM.

You Can use any Public Cloud like “AWS” or “Digital Ocean” to deploy your server and access to it through ssh.

in this Demo i will use Centos 7 server to install Jenkins.

- Install Jenkins.

You Can use check this Article to install Jenkins on Centos 7:

<https://linuxize.com/post/how-to-install-jenkins-on-centos-7/>

- Install Docker and Git on Jenkins Server.

We need to Install Docker in Jenkins Server because when we will deploy our pipeline one of the stages be for “Build and Push Docker image to Docker hub”.

```
sudo yum -y install docker git
sudo systemctl start docker
sudo systemctl enable docker
sudo groupadd docker
sudo usermod -aG docker jenkins
sudo systemctl restart jenkins
sudo systemctl restart docker
```


3. Create Kubernetes Cluster

In this step we will create K8s Cluster that container two servers one “master” and the another “worker” so you need to launch two public VMs.

- Install Docker and Kubernetes on All servers.

The first thing that we are going to do is use SSH to log in to all machines.

Once we have logged in, we need to elevate privileges using sudo.

```
sudo su
```

Disable SELinux.

```
setenforce 0
```

```
sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
```

Enable the br_netfilter module for cluster communication.

```
modprobe br_netfilter  
echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

Ensure that the Docker dependencies are satisfied.

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

Add the Docker repo and install Docker.

```
yum-config-manager --add-repo  
https://download.docker.com/linux/centos/docker-ce.repo  
yum install -y docker-ce
```

Set the cgroup driver for Docker to systemd, then reload systemd, enable and start Docker

```
sed -i '/^ExecStart/ s/$/ --exec-opt native.cgroupdriver=systemd/'  
/usr/lib/systemd/system/docker.service  
systemctl daemon-reload  
systemctl enable docker --now
```

Add the repo for Kubernetes.

```
cat << EOF > /etc/yum.repos.d/kubernetes.repo  
[kubernetes]  
name=Kubernetes  
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x  
86_64  
enabled=1  
gpgcheck=0  
repo_gpgcheck=0  
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg  
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg  
EOF
```

Install Kubernetes.

```
yum install -y kubelet kubeadm kubectl
```

Enable the kubelet service.

The kubelet service will fail to start until the cluster is initialized, this is expected.

```
systemctl enable kubelet
```

-*Note: Complete the following section on the MASTER ONLY!

Initialize the cluster using the IP range for Flannel.

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

Copy the kubeadm join command that is in the output.

We will need this later.

Exit sudo and copy the admin.conf to your home directory and take ownership.

```
mkdir -p $HOME/.kube  
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Deploy Flannel.

```
kubectl apply -f  
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/  
kube-flannel.yml
```

Check the cluster state.

```
kubectl get pods --all-namespaces
```

-*Note: Complete the following steps on the NODES ONLY!

Run the join command that you copied earlier, this requires running the command as sudo on the nodes.

Then check your nodes from the master.

kubectl get nodes

cloud_user@kareemfadl102c:~\$ kubectl get nodes

NAME	STATUS	ROLES	AGE	VERSION
kareemfadl102c.mylabserver.com	Ready	master	3d2h	v1.13.4
kareemfadl103c.mylabserver.com	Ready	<none>	3d2h	v1.13.4

Note : One of Most features in k8s is Autoscaling (HPA):

<https://kubernetes.io/docs/tasks/federation/administer-federation/hpa/>

In Our application i deployed the HPA for Autoscaling.

But it's depend on getting merics like CPU from K8s cluster.

To install server-metrics into our k8s cluster follow this link :

<https://medium.com/@cagri.ersen/kubernetes-metrics-server-installation-d93380de008>

4. Install Prometheus and Grafana in Kubernetes Cluster

In this step we will install prometheus and grafana for monitoring kubernetes infrastructures and containers that running on this cluster

The first step toward using Prometheus and Grafana to gather metrics within Kubernetes is to install them.

```
curl
https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get >
/tmp/get_helm.sh
```

```
chmod 700 /tmp/get_helm.sh
DESIRED_VERSION=v2.8.2 /tmp/get_helm.sh
helm init --wait
```

```
kubectl --namespace=kube-system create clusterrolebinding
add-on-cluster-admin --clusterrole=cluster-admin
--serviceaccount=kube-system:default
```

```
helm ls
cd ~/
git clone https://github.com/kubernetes/charts
cd charts
git checkout efdcffe0b6973111ec6e5e83136ea74cdbe6527d
cd ../
```

vim prometheus-values.yml

prometheus-values.yml:

alertmanager:

persistentVolume:

enabled: false

server:

persistentVolume:

enabled: false

Then run:

*helm install -f prometheus-values.yml charts/stable/prometheus --name
prometheus --namespace prometheus*

vim grafana-values.yml

grafana-values.yml:

adminPassword: password

Then run:

*helm install -f grafana-values.yml charts/stable/grafana/ --name grafana
--namespace grafana*

```
vim grafana-ext.yml
```

```
grafana-ext.yml:
```

```
-----  
kind: Service  
apiVersion: v1  
metadata:  
  namespace: grafana  
  name: grafana-ext  
spec:  
  type: NodePort  
  selector:  
    app: grafana  
  ports:  
    - protocol: TCP  
      port: 3000  
      nodePort: 8080  
-----
```

Then run:

```
kubectl apply -f grafana-ext.yml
```

You can check on the status of the prometheus and grafana pods with these commands:

```
kubectl get pods -n prometheus
```

```
kubectl get pods -n grafana
```

You can access your Grafana URL through this link:

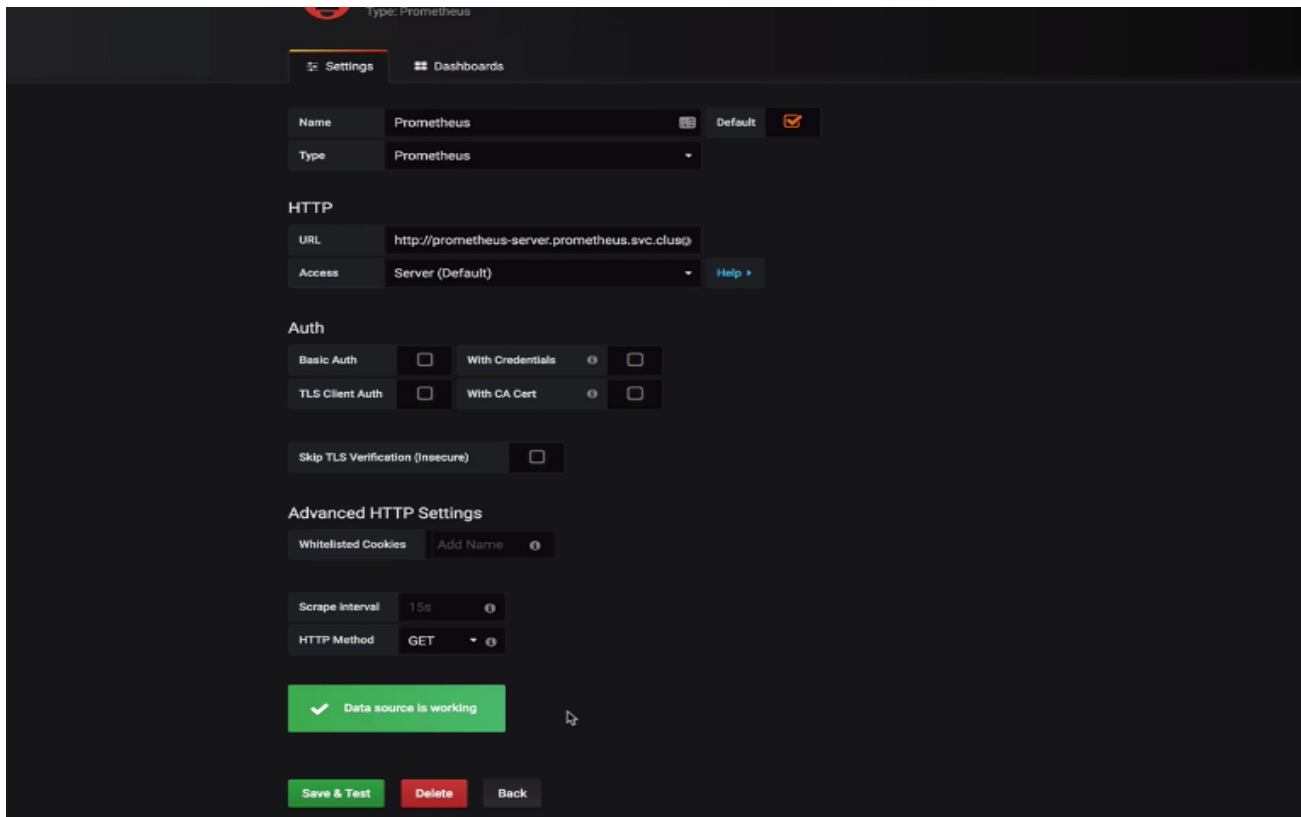
<http://k8s-master-IP:8080>

User: admin

password: password

setting up your datasource in grafana, use this url:

`http://prometheus-server.prometheus.svc.cluster.local`



The screenshot shows the Grafana web interface for configuring a Prometheus data source. The top navigation bar has 'Settings' and 'Dashboards' tabs. The main content area is titled 'Prometheus' and includes the following sections:

- Name:** Prometheus (with a 'Default' label and a checkmark icon).
- Type:** Prometheus (selected from a dropdown menu).
- HTTP:**
 - URL:** `http://prometheus-server.prometheus.svc.cluster.local`
 - Access:** Server (Default) (with a 'Help' link).
- Auth:**
 - Basic Auth:** ☐ **With Credentials:** ☐
 - TLS Client Auth:** ☐ **With CA Cert:** ☐
 - Skip TLS Verification (Insecure):** ☐
- Advanced HTTP Settings:**
 - Whitelisted Cookies:** Add Name (with an info icon)
 - Scrape Interval:** 15s (with an info icon)
 - HTTP Method:** GET (with an info icon)

A green status bar at the bottom of the configuration area says 'Data source is working' with a checkmark icon. At the very bottom, there are three buttons: 'Save & Test' (green), 'Delete' (red), and 'Back' (grey).

One of the important ways that monitoring can help you maintain stability in the context of Kubernetes is monitoring of the Kubernetes cluster itself.

Fortunately, there is a ready-made community dashboard that can be easily imported into your grafana installation to quickly get some insight into the performance and stability of your cluster.

You can import the Kubernetes All Nodes community dashboard in order to set up cluster monitoring for your Kubernetes cluster.

You can find additional community dashboards here:

<https://grafana.com/dashboards>

Check here for more info on the Kubernetes All Nodes dashboard:

<https://grafana.com/dashboards/3131>

Check here for more info on the Kubernetes All Pods dashboard:

<https://grafana.com/dashboards/3146>

Note : in case you not show cpu or metics edit to graph and add the following : <https://github.com/vegasbrianc/prometheus/issues/109>

You can Create an Alert through grafana Dashboard, for example :
Choose Kubernetes All Nodes dashboard >> CPU section >> Edit >>
choose “Alert” >> Enter your condition “Threshold” and save it.

You can create a notification also through notification section and attach it to your alert in the Dashboard.

But you will need to configure mail server for that.

5. Create Your Docker Hub account from here :

To Push your image versions : <https://hub.docker.com>

6. Create Your AWS account from here :

To Use S3 service and send backup for our DB of app to S3 service :

<https://console.aws.amazon.com/>

7. Integrate Jenkins with GitHub , Docker Hub and Kubernetes Cluster.

Now We have a Fresh Installation for Jenkins, to Apply the CICD Pipeline we need authentication between Jenkins and Github, Docker Hub and Kubernetes Cluster.

Before we start to do that we need to install the following plugin for k8s integration :

<https://wiki.jenkins.io/display/JENKINS/Kubernetes+Continuous+Deploy+Plugin>

Jenkins with Github , Docker Hub , Kubernetes Cluser

Access to Jenkins Url with your permission and new add credentials, with will add new credentials.

-For GitHub >> “Username with password Type” add your GitHub user and password with ID called “github_token”

-For Docker Hub >> “Username with password Type” add your Docker Hub user and password with ID called “docker_hub_login”

-For K8s >> “Kubernetes login Type” with ID called “kubeconfig” and Chose “Enter Directely” , add the content of your kubernetes cluster you can find it in the following path in k8s master node :
`cloud_user@kareemfadl102c:~$ cat ~/.kube/config`



Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

	Name	Kind	Description	
	karimfadl/***** (github_token)	Username with password	github_token	
	karimfadl/***** (docker_hub_login)	Username with password	docker_hub_login	
	kubeconfig (kubeconfig)	Kubernetes configuration (kubeconfig)	kubeconfig	

Icon: [S](#) [M](#) [L](#)

8. Integrate GitHub With Jenkins

In This case you need to trigger any update happen in your Github repo to deploy your action.

That's mean when you make a comment in your code and push the updates to your Github Repo the Jenkins trigger that and deploy your container directly.

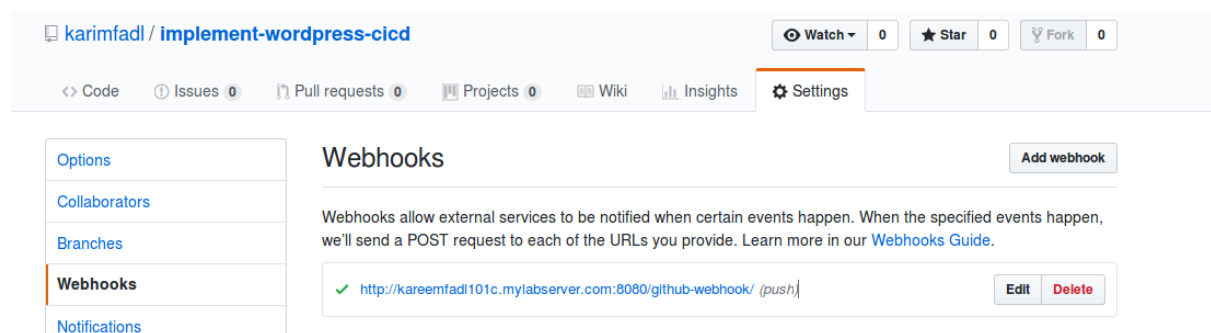
We will use Webhook tool in Github Account by the following:

-Access your GitHub account at in your repo >> setting >> webhook >> add new webhook.

-At Payload URL add :

<http://Jenkins-Public-IP/github-webhook/>

-At Content type : chose application/json
and Add your webhook



8. Deploy Mysql DB in Kubernetes Cluster on master node.

Before we create Mysql DB we need to specify some vars like Database password, Access key and secret of AWS account to send our backup to S3 bucket.

Kubernetes use secrets method to save that.

<https://kubernetes.io/docs/concepts/configuration/secret/>

-Create AWS account with admin access to s3 service, save access and secret key

-My database password that i chosen is "password" for "wordpress" user that have full privileges in "wordpress" database

Clone Your Repo in k8s master node and start to deploy Mysql DB.

for example :

```
git clone https://github.com/karimfadl/implement-wordpress-cicd
cd implement-wordpress-cicd
```

To convert my "DB password" "password" to add it in secret.yml file :

```
karim@karim ~ $ echo -n 'password' | base64
cGFzc3dvcmQ=
```

Do the same command for Access and secret key and add all of them to secret file.

Deploy secret and mysql yml files.

```
kubectl create -f secret.yml
```

```
kubectl create -f configmap-wordpress.yml
```

```
kubectl create -f mysql.yml
```

Note : if you face an issue because of mysql service rang port deploy

Follow this link :

<http://www.thinkcode.se/blog/2019/02/20/kubernetes-service-node-port-range>

Deploy HPA yml files.

```
kubectrl create -f hpa-wordpress.yml
```

Check mysql pods , services, hpa, secret

```
kubectrl get pods,svc,hpa,secret
```

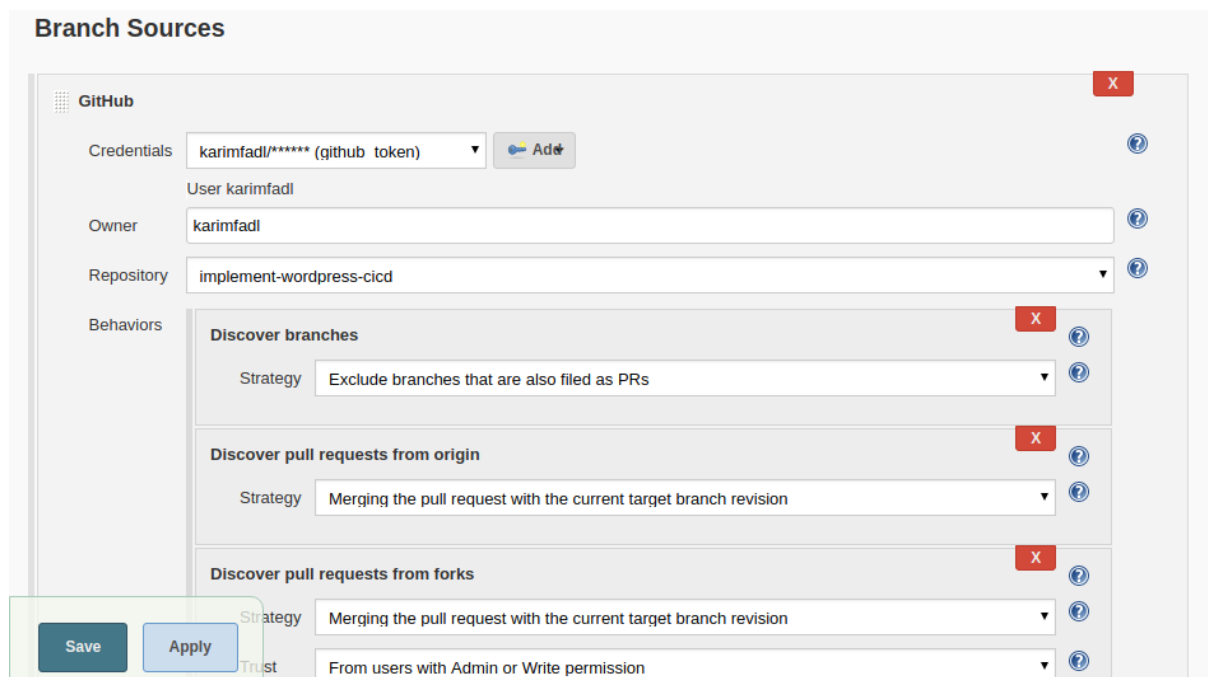
Note in case you need to change the App variables like database name, database user, database password, or Root Database password

You need to change in “secret.yml” and “configmap-wordpress.yml” files.

10. Create your CICD Pipeline in Jenkins

Create Jenkins new item with name for example “CICD-wordpress” and “multibranch pipeline”

-In GitHub section add your credentials - Owner “Github user” - You Repo that Forked from My Repo.

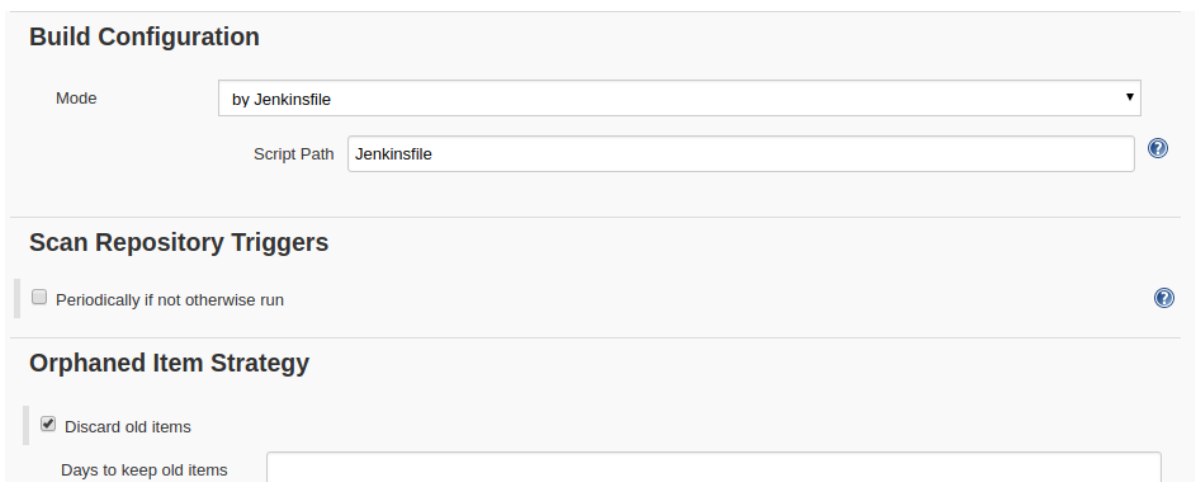


The screenshot shows the 'Branch Sources' configuration page in Jenkins. The 'GitHub' section is expanded, showing the following fields:

- Credentials:** A dropdown menu showing 'karimfadi/***** (github token)' with an 'Add' button.
- User:** A text field containing 'karimfadi'.
- Owner:** A text field containing 'karimfadi'.
- Repository:** A dropdown menu showing 'implement-wordpress-cicd'.
- Behaviors:** A section with three expandable items:
 - Discover branches:** Strategy: 'Exclude branches that are also filed as PRs'.
 - Discover pull requests from origin:** Strategy: 'Merging the pull request with the current target branch revision'.
 - Discover pull requests from forks:** Strategy: 'Merging the pull request with the current target branch revision'.

At the bottom left, there are 'Save' and 'Apply' buttons. On the right side of each behavior section, there is a red 'X' icon and a help icon (question mark).

-In Build Configuration chose Jenkinsfile “It’s the default”



The screenshot shows the 'Build Configuration' page in Jenkins. The 'Mode' dropdown is set to 'by Jenkinsfile'. The 'Script Path' text field contains 'Jenkinsfile'. Below this, the 'Scan Repository Triggers' section has a checkbox for 'Periodically if not otherwise run' which is unchecked. The 'Orphaned Item Strategy' section has a checkbox for 'Discard old items' which is checked. Below this, there is a text field for 'Days to keep old items' which is currently empty.

For testing App before deploy it in production i am using Canary.

<https://hackernoon.com/canary-release-with-kubernetes-1b732f2832ac>

Now we need to change in Jenkinsfile the following :

```
DOCKER_IMAGE_NAME = "karimfadi/wordpress"
```

to

```
DOCKER_IMAGE_NAME = "You_dockerhub_user/wordpress"
```

Commit the Changes and push the updates.

Check the pipeline steps You will find the Code deployed first in Canary pods to test it and you can check it through the browser by accessing this link : http://k8s_node_ip:8082

If it Ok back to pipeline and click to proceed

It should start to Deploy your wordpress App with stateful Database

Open your browser and access this link : http://k8s_node_ip:8081

and start your Wordpress App.

11. Create Backup for your Database

-Access to AWS console with your Account permission and create S3 bucket called for example "yourname-db-backup" >> make it public and chose the region that you need for example "ohio >> us-east-2"
Also Create folder called for example "database" inside the bucket.

in k8s master node Edit and start to deploy Mysql DB backup yml file.

for example :

```
vim backup-mysql.yml
```

Change the following :

- name: AWS_DEFAULT_REGION
value: "us-east-2"
- name: AWS_BUCKET_NAME
value: "karimfadi-db"
- name: AWS_BUCKET_BACKUP_PATH
value: "/database"

Replace Value of Region , Bucket name and Path.

Choose the desired schedule, for example :



```
schedule: "*/2 * * * *"
```

now i get back up every 2 mins.

Now you can launch the yml file :

```
kubectl create -f backup-mysql.yml
```


Wait for 2 mins and check your bucket you will see like that :

Viewing 1 to 2			
<input type="checkbox"/> Name ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>  wordpressMay-17-2019-12-12.sql	May 17, 2019 2:12:09 PM GMT+0200	514.6 KB	Standard
<input type="checkbox"/>  wordpressMay-17-2019-12-14.sql	May 17, 2019 2:14:08 PM GMT+0200	514.6 KB	Standard

Thanks for Reading, I am still enhance in this Repo appreciate your advices and you can contact with me through my Linkedin Profile :

<https://www.linkedin.com/in/karim-fadl/>