Topic:: Create CI/CD for front-end vue.js project in Ubuntu 18.04.6

- **First of all, add Script folder with including its scripts inside of it and appspec.yml file in project file's root, for that there is already one doc. Share in drive just follow it.**
- **Step1:** Create ec2 instance having OS ubuntu version 18.04.6, and instance type is t2.micro, also while creating ec2 instance, in advanced details select IAM role which we created for ec2 in previous file link: https://drive.google.com/file/d/1HAHRM-xYz0-4oBAzKro73FrAfRPrHm0y/view?usp=share_link
- Also in advanced settings > "User data" add bin/bash data

```
#!/bin/bash


sudo apt-get -y update

sudo apt-get -y install ruby-full

cd /var/www/html/

sudo apt-get install wget

wget https://aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.com/latest/install

sudo chmod +x ./install

sudo ./install auto

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash

. ~/.nvm/nvm.sh

nvm install 16

npm i @vue/cli-service --legacy-peer-deps

sudo npm install -g aws-cli
```

And launce ec2.

- On other side go to Code Deploy service and select Create application in Applications.

- **Step2:**Give name it and Select “EC2/On-premises” in compute platform and create application
- **Step3:**Now Select “Create deployment group” in created application, give name to deployment group, select role which we created for code deploy in previously whose link is available in step 1:
- In **Environment configuration** select Amazon EC2 instances, in “key” select “name", in value option select ec2 instance name on which you want to live project.
- In “load balancer” option untick “Enable load balancing” and select “Create deployment group”
- Now Goto Create pipeline, give name pipeline click next, in source add GitHub (Version2), if doing this first time then select “connect to GitHub", give name to connection and select “connect to GitHub", then select “Install New app” in that select “**All repositories**” if you want to add all repo in connection or select “**Only select repositories**” if want to add specific repo. In connection. Then save it
- Now, in repository name select the repo you want to connect and in branch name, name of the branch you want to add, click Next.
- Skip the build stage.
- Deploy stage select “AWS CodeDeploy” in Application name of the app. We previously create in step 2&3., click next. Click “Create Pipeline”.
- In Pipeline Deploy section go to details > Events > View Events.
- If all events show pending then we have to manually add code deploy agent ec2 first for that connect to ec2 terminal either direct or using PuTTY (for how connect see any YouTube vid.)
- Fire following cmds for enable codedeployagent:

```
sudo apt-get -y update
```

```
sudo apt-get -y install ruby-full
```

- Note:: if came any err during above two cmd just reapeat it again and again and err remove or in err show some package is missiong and also mention cmd in that for download it so just download it.

Enter cmd “ruby –v” if show version of ruby means above two cmds works proprly

```
sudo apt-get install wget
```

```
wget https://aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.com/latest/install
```

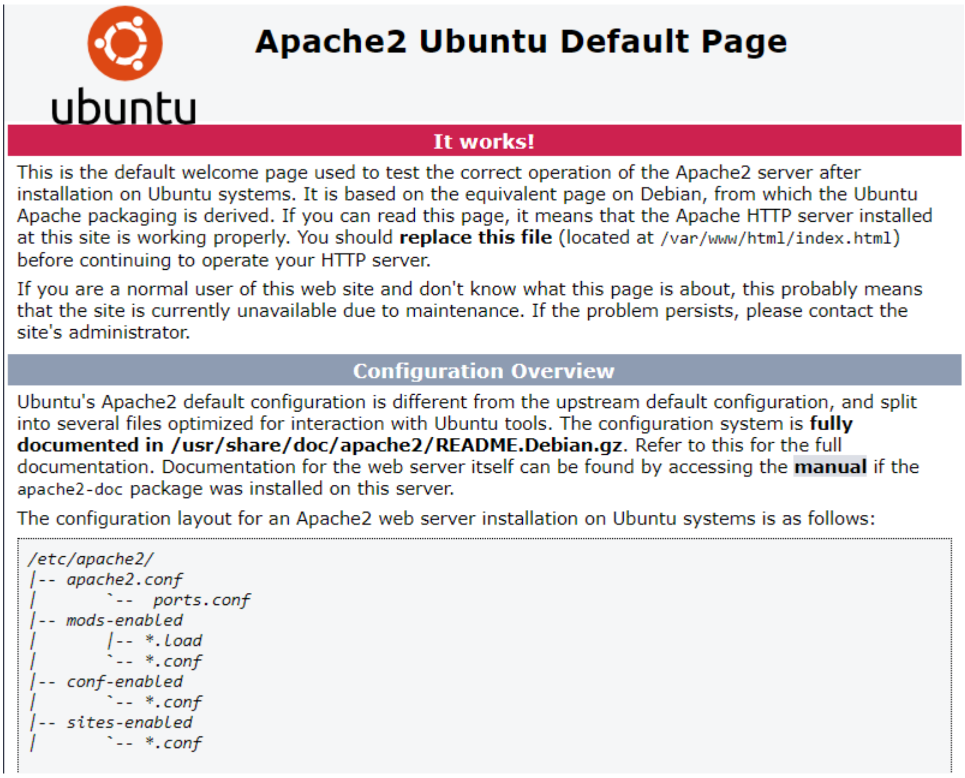
```
sudo chmod +x ./install
```

```
sudo ./install auto
```

```
sudo service codedeploy-agent start
```

```
sudo service codedeploy-agent status
```

- After last cmd if Active: active(running) show it means codedeploy-agent works proprely.
- Now re-run codepipeline and check in details > events > status,now show Succeeded.
- Run ip of ec2 if gives ,below ss then in terminal of ubuntu do some changes as given below.



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- Fire cmd :

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

- add path in “DocumentRoot” /var/www/html/ to /var/www/html/dist
- add given below data just above the </VirtualHost> as we can see in fig.

```
<Directory “/var/www/html”>
```

```
Options Indexes FollowSymLinks MultiViews
AllowOverride All
Require all granted
</Directory>
```

- Then restart apache2.

```
sudo service apache2 restart
```

- Add .htaccess file and fire cmd:

```
sudo a2enmod rewrite
sudo nano /etc/apache2/apache2.conf
```

- Using second cmd apache2.conf open in very bottem add

```
<Directory "/var/www/html">
AllowOverride All
</Directory>
```

- Now re-run ip add. It will starts work fine.