ADVANCE DEVOPS
LAB MANUAL

1. **Launch Windows instances of AWS EC2 . Get connected to instances using RDP client software.**

STEP 1:
- Go to AWS Management Console
- Click on Services and search 'EC2'

STEP 2:
- Click Launch instance.
- Give name to the instance
- Under OS Type, select Windows
- Under Key Pair, click on 'Create key pair'
  - ✓ Give name to the key pair
  - ✓ Click RSA
  - ✓ Select .pem
  - ✓ Create key pair.
- Under key pair, select the key pair just created.
- Under Network, open all ports (check all boxes)
- Launch Instance

STEP 3:
- Go on instances
- Status check -> 2/2 checks passed.
- Select Instance
- Go on Connect ->RDP Client.
- Download remote desktop file
- Get password ->Upload private key
- .pem file -> Decrypt password
- Copy password.
- Open RDP file → Connect
- Paste password.
- Ok then Yes
- Optional (Search any thing on Microsoft Edge)'

STEP 4:
- Terminate Instance

2. Launch ubuntu instances of AWS EC2 . Get connected to instances using MobaXterm client software.

STEP 1:

- ➢ Download MobaXterm (if already not downloaded)
  - ✓ Mobaxterm
  - ✓ Search Mobaxterm on Google.
  - ✓ Download Installer edition

STEP 2:

- ➢ Go to AWS Management Console
- ➢ Click on Services and search 'EC2'

STEP 3:

- ➢ Click Launch instance.
- ➢ Give name to the instance
- ➢ Under OS Type, select Amazon Linux
- ➢ Under Key Pair, click on 'Create key pair'
  - ✓ Give name to the key pair
  - ✓ Click RSA
  - ✓ Select .ppk
  - ✓ Create key pair.
- ➢ Under key pair, select the key pair just created.
- ➢ Under Network, open all ports (check all boxes)
- ➢ Launch Instance

STEP 4:

- ➢ Extract Mobaxtern
- ➢ Install Mobaxterm (Next...... Karte jao)

STEP 5:

- ➢ Go on instances
- ➢ Status check -> 2/2 checks passed.
- ➢ Select Instance
- ➢ Click on Name
- ➢ Copy IPv4 address

STEP 6:

- ➢ Search MobaXterm
- ➢ Open it
- ➢ Click Session
- ➢ Click SSH
- ➢ Paste IPv4 Remote Host.
- ➢ Advanced Setting
- ➢ Under use private key, attach ppk file
- ➢ Accept

STEP 7:

Enter the following:

  - ✓ ec 2-user
  - ✓ mkdir , Is, date, time, cal (or any of your choice)

3. Deployment of static website on AWS S3

STEP 1:

- Go to AWS Management Console
- Click on Services and search ‘Amazon S3 buckets’

STEP 2:

- Click on Create bucket
- Give name to bucket
- Under ‘**Block Public Access settings for this bucket’**
    - ✓ uncheck ‘**Block all public access’.**
    - ✓ Check the acknowlegment
- Click ‘Create Bucket’

STEP 3:

- Click on Bucket Name
- Click on ‘Permissions’
- Under Bucket Policy
    - ✓ Click ‘Edit’
    - ✓ Paste the following code **[Prefarably paste directly from GCR because here indents are not proper]**

```
{
"Version": "2008-10-17",
"Id": "PolicyForPublicWebsiteContent",
"Statement": [
{
"Sid": "PublicReadGetObject",
"Effect": "Allow",
"Principal": {
"AWS": "*"
},
"Action": "s3:GetObject",
"Resource": "arn:aws:s3:::bucketname/*"
}
]
}
```

- ✓ **<u>Note:</u> Change the highlighted part in the code (line 12) to the bucket name.**
- ✓ Save Changes

STEP 4:

- Go on ‘Objects’
- Upload any html file
    - ✓ Click ‘Upload’
    - ✓ Click ‘Add Files’
    - ✓ Click ‘Upload’

STEP 5:

- Go to Objects
- Click File name
- Copy Object URL
- Paste On Chrome

STEP 6:

- Delete Object.
- Delete Bucket

4. Run Node.js scripts in an AWS Cloud9 development environment.

STEP 1:

- ➢ Open the following link
  - ✓ https://docs.aws.amazon.com/cloud9/latest/user-guide/sample-nodejs.htmL

OR

  - ✓ Go to documentation, then user guide.

STEP 2:

- ➢ Go to AWS Management Console (In a new tab)
  - ✓ Click on Services and search 'Cloud9'

STEP 3:

- ➢ Click 'create environment'
- ➢ Give name to the environment
- ➢ Create environment
- ➢ Select environment and open in cloud9 [from here refer documentation opened in step 1]
- ➢ File -> new file ->go to documentation ->copy code ->paste in file
- ➢ Save file -> hello.js->run configurations->new run configuration
- ➢ To run:
  - ✓ Command: hello.js 3 4 (any two variables)
- ➢ Node.js:
- ➢ Window->new terminal
- ➢ npm install aws-sdk
- ➢ File -> new file ->go to documentation ->copy code (v2) ->paste in file-> Save file ->s3.js-> run configurations->new run configuration
- ➢ Window->new terminal
- ➢ npm install async
- ➢ Command -> s3.js my-test-bucket us-west-2

5. Run Python code in an AWS Cloud9 development environment.

STEP 1:

- ➢ Open the following link
  - ✓ https://docs.aws.amazon.com/cloud9/latest/user-guide/sample-python.html

OR

  - ✓ Go to documentation, then user guide.

STEP 2:

- ➢ Go to AWS Management Console (In a new tab)
  - ✓ Click on Services and search 'Cloud9'

STEP 3:

- ➢ Click 'create environment'
- ➢ Give name to the environment
- ➢ Create environment
- ➢ Select environment and open in cloud9 [from here refer documentation opened in step 1]
- ➢ File -> new file ->go to documentation ->copy code ->paste in file
- ➢ Save file -> hello.py->run configurations->new run configuration
- ➢ Command -> hello.py 3 4 (any two variables)
- ➢ Python (Boto3):
- ➢ Window->new terminal
- ➢ python -m pip show boto3 (if not found)
- ➢ then this : sudo python3 -m pip install boto3
- ➢ File -> new file ->go to documentation ->copy code ->paste in file-> Save file ->s3.py-> run configurations->new run configuration
- ➢ Command -> s3.py my-test-bucket us-west-2

6. create Lambda functions using Python / Java / Nodejs. Create AWs Lambda function and configure a trigger for Amazon Simple Storage Service (Amazon S3). The trigger invokes your Lambda function every time that you add an object to you Amazon S3 bucket. Allow AWS Lambda to access Amazon DynamoDB Table .Create IAM role that allows full access to DynamoDB Table

STEP 1:

- ➢ Go to AWS Console

STEP 2:

- ➢ Go to AWS service IAM
- ➢ Click on role (ie on the left side under 'access management')
- ➢ Click on create a role
- ➢ Select AWS service .
- ➢ For common use cases, select Lambda.
- ➢ For use case for other AWS service select DynamoDB
- ➢ click on next
- ➢ Now in permissions search for dynamo and you will see 4 options select all

  (ie . AmazonDynamoDBReadOnlyAccess
  AWSLambdaInvocation-DynamoDB
  AWSLambdaDynamoDBExecutionRole
  AmazonDynamoDBFullAccess)
- ➢ Click on next
- ➢ Give role name
- ➢ Click on create role

STEP 3:

- ➢ search for AWS service Lambda
- ➢ click on create function
- ➢ click on author from scratch
- ➢ give your function a name
- ➢ chose language as Python 3.11
- ➢ In architecture select x86_64
- ➢ Go to change default execution role and click on use an existing role, select your existing role
- ➢ Click on create function

STEP 4:

- ➢ Go to AWS service S3
- ➢ Create a bucket
- ➢ In general Configuration give your bucket a name rest everything remains the same
- ➢ In object ownership dont change the default setting [ie ACLS Disabled (recommended)]
- ➢ For Block public access setting, Uncheck block all public access
- ➢ Click on acknowledgement checkbox
- ➢ Rest don't change anything
- ➢ Click on create bucket

STEP 5:

- Go to AWS service Lambda
- Click on your function name
- Establish a connection between lambda and S3 bucket by clicking in add trigger Select S3
- Select your bucket
- check on acknowledgement checkbox and add
- (A connection will be formed between lambda and S3)
- In Lambda under function overview you will see code, test, monitor configuration, Aliases and version
- In there click on code and paste this **[Prefarably paste directly from GCR because here indents are not proper]**

```
import boto3
from uuid import uuid4
def lambda_handler(event, context):
        s3 = boto3.client("s3")
        dynamodb = boto3.resource('dynamodb')
         for record in event['Records']:
                bucket_name = record['s3']['bucket']['name']
                object_key = record['s3']['object']['key']
                size = record['s3']['object'].get('size', -1)
                event_name = record ['eventName']
                event_time = record['eventTime']
                dynamoTable = dynamodb.Table('newtable')
                dynamoTable.put_item(
                        Item={'unique': str(uuid4()), 'Bucket': bucket_name, 'Object': object_key,'Size': size,
                'Event': event_name, 'EventTime': event_time})
```

- Deploy the code

STEP 6:

- Go to AWS Service DynamoDB
- Click on create table
- give table name same as the name you put in the code pasted in lambda code(newtable)
- give partition key same as you put in the code pasted in lambda code(unique)
- Don't change anything and create a table

STEP 7:

- Go to AWS Service S3
- upload a jpg file in your bucket
- click on upload

STEP 8:

- Go to AWS Service DynamoDB
- click on your table name
- click on explore table items
- scroll down and you will see the output

STEP 9:

- delete the table
- delete objects from bucket and then the bucket
- delete function from lambda

7. To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.Use AWS Lambda blueprint.

STEP 1:

- ➢ Login to your account
- ➢ Go to AWS Console

STEP 2:

- ➢ Search for ‘S3’ (Buckets).
- ➢ Go to S3
- ➢ Click on ‘Create Bucket’ (Orange colour on the right hand side)
- ➢ Give a name to your bucket (Rules for bucket name is hyperlinked below the input box)
- ➢ Scroll down and click on ‘Create Bucket’
- ➢ Your bucket is created.

STEP 3:

- ➢ Go to AWS Console again (in a duplicate/new tab)

STEP 4:

- ➢ Search for AWS service Lambda
- ➢ Click on ‘Create Function’.
- ➢ Click on ‘Use a Blueprint’. (2nd from the 4 boxes)
- ➢ Under basic Functions
  - ✓ Select blueprint as ‘ get S3 object’ and language as python 3.10 (or any python version available)
  - ✓ Give your function a name
  - ✓ Under execution role select ‘Create a new role from AWS policy templates’
  - ✓ Give a role name
- ➢ Under S3 Trigger
  - ✓ Under bucket add your bucket created in the previous step.
  - ✓ Under event types select ‘all object create events’ only.
  - ✓ Check the box saying ‘I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.’
- ➢ Scroll down and click on ‘Create Function’
- ➢ Your function is created
- ➢ Your function should be connected to the bucket. (in the diagram type thing)

STEP 5:

- ➢ Scroll down and click ‘code’ on the left hand side
- ➢ Under ‘code’ add a print statement in the code

```
print(‘YOUR NAME’)
```

below the line

```
print("CONTENT TYPE: " + response['ContentType'])
```

- ➢ Deploy the code (click on the deploy button beside the test button)

STEP 6:

- Go to back to your buckets
- Click on the bucket created in the 2nd step
- Click on the upload button (towards the right)
- Upload any image using the add files button (in the centre)
- After uploading, select the image file, scroll down and then click on upload (orange button).

STEP 7:

- Go back to your functions
- Click on the function created in the 4th step
- Scroll down and click 'monitor'
- Under 'monitor' Click on 'View CloudWatch Logs' to check the logs when the function is updated.
- Scroll down and click the log under log stream
- Your name should be visible in the logs
- If your name is visible then your experiment has been executed and completed.

STEP 8:

- Delete objects from the bucket
- Delete objects
- Delete function from lambda
- Logout from account

8. Deploy a containerized web Application on AWS EC2 Linux .[install Docker ,pull nginx image and run it].Pull python images and run the command to list all the locally stored docker images .

STEP 1:

- Create a EC2 instance with followings specs
  - ✓ <u>OS</u>- Amazon Linux
  - ✓ Create a new <u>key pair</u> (RSA and PEM)
  - ✓ <u>Network settings</u>: Tick allow SSH traffic from, Allow HTTPS traffic from internet, Allow HTTPS traffic from internet

STEP 2:

- When instance is created select it and click on the connect button (first button after the reload button on top right corner) and copy its Public IPv4 DNS address.

STEP 3:

- Open MobaXterm start new session (Click on session button in top left corner)
- Select SHH (at top left, first button)
- Paste the Public IPv4 DNS address in remote user text bar
- Specify user as [ec2-user]
- In Advanced SHH settings, click on the check box for use private key and then add the key pair in that section.
- Then click on OK and then an alert box will appear click accept on it

STEP 4:

- Once the terminal opens up it asks 'login as' here type 'ec2-user'

STEP 5:

- Now you're logged in into the instance now execute the following commands sequentially.
  - ✓ sudo su
  - ✓ yum install docker -y
  - ✓ service docker start
  - ✓ docker pull nginx
  - ✓ docker images
  - ✓ docker ps
  - ✓ docker run -p 80:80 nginx

STEP 6:

- After execution of this commands, go to the instance networking details of the instance we created and from there copy the Public IPv4 address and paste it on the browser a webpage of nginx will appear and our exp is completed.

9. Install docker on AWS EC2 –Ubuntu by using curl
#curl -fsSLhttps://get.docker.com -o get-docker.sh
#sh get-docker.sh
Run hello-world from docker hub and explain the steps
Pull 3 or 4 images ,one of the python , run " Hello World " inside the container.

Step 1: Login to AWS Management Console Dashboard

Step 2: Search for EC2 and select it.

Step 3: Click on "Launch Instances"

Step 4: Give any name to your "Instance".

Step 5: Select UBUNTU and go with 20.04 LTS(HVM), SSD Volume Type

Step 6: Create a New Keypair

Step 7: Allow all the Traffic from the Internet and click on Launch Instances (In the network settings). Click on View Instances

Step 8: Connect the Instances.

Step 9: Go to SSH client and copy Public DNS

Step 10: Now paste the Public DNS in MobaXterm for that Go to MobaXterm → Session → SSH (paste the public DNS in remote host and write "ubuntu" in username)→Adv SSH Setting(use the private key installed).

Step 11: Your UBUNTU terminal is created, for Root access type "sudo su".

Step 12: Now use the command "curl -fsSL https://get.docker.com -o get-docker.sh" & "sh get-docker.sh" to pull Docker.

<u>RUN HELLO WORLD FROM DOCKER HUB AND EXPLAIN IT'S STEPS</u>

Step 13: Enter command 'docker -- version' to see the current docker version. Then run command 'docker images' to see installed images. Also run command 'docker run hello-world' which will run hello world from docker.hub.com and run it.

Step 14: Now run 'docker images' again, the repository will have an image named as 'hello-world'.

<u>Pull 3 or 4 images, one of the python, run " Hello World "inside container.</u>

Step 15: Run 'docker pull mysql' , 'docker pull nginx' , 'docker pull python'. These commands will pull the images from docker hub. To check the pulled images run command 'docker images' you will get image id as well.

Step 16: Now, to get access to the python container use command→ ' docker run -it (images id, of python image, starting 4 characters are enough) bash' and type 'python' to enter the shell and write any code.

Eg: docker run -it e285 bash

Step 17: write “print(‘HELLO WORLD!!’)” in the shell.

10. Install docker on AWS EC2 –Ubuntu by using curl
#curl -fsSLhttps://get.docker.com -o get-docker.sh
#sh get-docker.sh
Demonstrate any 5 docker command and explain its uses

Step 1: login to AWS Management Console Dashboard

Step 2: Search for EC2 and select it.

Step 3: Click on "Launch Instances"

Step 4: Give any name to your "Instance".

Step 5: Select UBUNTU and go with 20.04 LTS(HVM), SSD Volume Type

Step 6: Create a New Keypair

Step 7: Allow all the Traffic from the Internet and click on Launch Instances (In the network settings). Click on View Instances

Step 8: Connect the Instances.

Step 9: Go to SSH client and copy Public DNS

Step 10: Now paste the Public DNS in MobaXterm for that Go to MobaXterm → Session → SSH (paste the public DNS in remote host and write "ubuntu" in username)→Adv SSH Setting(use the private key installed).

Step 11: Your UBUNTU terminal is created, for Root access type "sudo su".

Step 12: Now use the command "curl -fsSL https://get.docker.com -o get-docker.sh" & "sh get-docker.sh" to pull Docker.

<u>DEMOSNTRATE 5 DOCKER COMMANDS:</u>

1) Enter command '**docker --version**' to see the current docker version.Then enter command 'docker python --version' to see the current python version(after running 'docker pull python').
2) For checking the status of the container use command '**docker ps -a**'.
3) Now run command '**docker history python**' to get the docker python history.
4) After running "**docker pull mysql**" Now run '**docker inspect mysq**l' to inspect mysql.
5) Run '**docker images** ' to get the list of images present.
6) Now stop the container by using command '**docker stop [container/image id]**' → and simultaniously kill the docker by 'docker kill'.

<u>Note</u>: you will get container id by running command 'docker images'.

11. Run a Flask Application inside a Docker Container and explain the steps.

- **Step 1**: login to AWS Management Console Dashboard
- **Step 2**: Search for EC2 and select it.
- **Step 3**: Click on “Launch Instances”
- **Step 4**: Give any name to your “Instance”.
- **Step 5**: Select UBUNTU and go with 20.04 LTS(HVM), SSD Volume Type
- **Step 6**: Create a New Keypair
- **Step 7**: Allow all the Traffic from the Internet and click on Launch Instances (In the network settings). Click on View Instances
- **Step 8**: Connect the Instances.
- **Step 9**: Go to SSH client and copy Public DNS
- **Step 10**: Now paste the Public DNS in MobaXterm for that Go to MobaXterm → Session → SSH (paste the public DNS in remote host and write “ubuntu” in username)→Adv SSH Setting(use the private key installed).
- **Step 11**: Your UBUNTU terminal is created, for Root access type “sudo su”.
- **Step 12**: Now use the command “curl -fsSL https://get.docker.com -o get-docker.sh” & “sh get-docker.sh” to pull Docker.
- **Step 13**: Run command “docker --version” then “docker images”.
- **Step 14**: Now create a directory [your own name], go inside it and create another directory name as “APP” and go inside it too.
  - ✓ mkdir yourname
  - ✓ cd yourname
  - ✓ mkdir app
  - ✓ cd app
- **Step 15**: Now open nano editor by [nano app.py] and add the code of flask given below

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello():
return "Demo Flask & Docker application is up and running!"
if __name__ == '__main__':
app.run(host="0.0.0.0", port=80)
```

- **Step 16**: Now open nano editor by [nano requirements.txt] and write the requirements of the file to be installed on other PC.
  - ✓ Write “flask” in the requirements.txt file

- **Step 17**: Now go back to your Parent Directory by [cd ..] and write [nano Dockerfile] and copy the code given below and Write code in Dockerfile.

```
FROM python
WORKDIR /opt/demo/
COPY /app .
RUN pip install -r requirements.txt
ENTRYPOINT python app.py
```

- **Step 18**: Now run the command “docker build -t [file name]:latest .”
- **Step 19**: Now for checking the images run command as “docker images”.
- **Step 20**: Run command “docker run -d -p 80:80 [directory name]”.
- **Step 21**: copy the Public IPv4 DNS. From the instances section in AWS
- **Step 22**: Paste it in the Address Bar in a new tab.
- **Step 23**: After performing now terminate the instance.

Summary of flow of commands in this exp chronologically

In MobaXterm

- Sudo su
- curl -fsSL https://get.docker.com -o get-docker.sh
- sh get-docker.sh
- docker –version
- docker images
- mkdir yourname
- cd yourname mkdir app
- cd app nano app.py
- nano requirements.txt
- cd ..-->nano Dockerfile
- docker build -t [file name]:latest .
- docker images
- docker run -d -p 80:80 [directory name]

12. Perform an experiment, to Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins on Linux Machine.Login to Nagios dashboard and just list any 5 services available of dashboard

Search ->Nagios Xl Online demo

Click ->
https://nagiosxi.demos.nagios.com/nagiosxi/login.php?redirect=/nagiosxi/index.php%3f&noauth=1

Click->Login as administrator

Click->login

Birdseye : red mei no functional machines , kuch to bhi problem huwa hai, green ones are working properly ,yellow : abhi to chalra hai but future mei it may give some problem

So nagios se apun apne pure network ka health moniter karte ho from the server ,its taking heartbeat of all clients machine

So nagio is one kimd of monitering service

Health monitor from server

Red - mar chuka

Yellow -takleef me

Green - fit

Console ,Home,Instance

Launch Instance.->Give name

Amazon linux->Key Pair

Saare Ports open ->edits

All traffic , Anywhere->Launch Instance

Connect <mobaxterm> (maam ne nhi kiya)

* learn Nagios , its features to be able to explain in orals (at least 5)

-pure network ka health monitor karta hai

Machine is ready

Sudo su

From manual commands

```
yum install httpd php
yum install gcc glibc glibc-common
yum install gd gd-devel
adduser -m nagios
passwd nagios
groupadd nagioscmd
usermod -a -G nagioscmd nagios
usermod -a -G nagioscmd apache
```

mkdir ~/downloads
cd ~/downloads

software : wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.4.14.tar.gz

plugins : wget http://nagios-plugins.org/download/nagios-plugins-2.4.6.tar.gz

unzip plugins : tar zxvf nagios-4.4.14.tar.gz

cd nagios-4.4.14
./configure --with-command-group=nagioscmd
sudo yum install openssl-devel
./configure --with-command-group=nagioscmd

make all

make install
make install-init
make install-config
make install-commandmode

make install-webconf

set the user as nagios admin:
htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

service httpd restart
cd ~/downloads
tar zxvf nagios-plugins-2.4.6.tar.gz
cd nagios-plugins-2.4.6

./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
make install

chkconfig --add nagios [not required]
chkconfig nagios on

/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

service nagios start
service httpd restart
come to console : copy public ipv4 address ->paste on crome

along with /nagios

tactical overview : 5 services explore for oral point of view

13. Perform an experiment, to understand Terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.Using Terraform , create an EC2 instance on AWS cloud
Run following Terraform commands
terraform init:
terraform plan:
terraform apply:
terraform destroy

Search IAM

Click on roles -> Click on create role

Select AWS services

Use Case->EC2

Go to next

Select EC2 again

Search EC2 full access

Click - › Next

Name, review, create->Give name -> **Create role**

Create sever->Create instance

Name it-> Select amazon linux

Create new Key Pairs

RSA .pem -> Create

Network settings-> Edit

Add security group rule

Type-> All traffic , Anywhere

Click Adv Details

IAM instance Profile Dropdown

<Terra form Role>

Launch Instance

Connect to Instance

EC2 instance Connect

Connect

Click Check box-> Click Connect

<Use Moba Xterm>

Sudo su ->Mkdir project-name->cd project-name

Nano variable.tf

Copy Yellow wala Code

Paste it

```
variable "security_group" {
   description = "Name of security group"
   default     = "my-jenkins-security-group1"
}
```

Changes:

Default “ “ (aws-region)

 Default “ ”(key name)

Default “ ”(ami_id)

Come to dash board

Select Ubuntu -> AMID. (Copy) *

Ls

Nonao main.tf

Copy yellow wala code-> Paste it

Save->Ctrl+O ENTER Ctrl+X

wget [paste from manual]

it is a zip file

Copy Software's name type-> unzip __________

cp terraform /bin/

terraform –version

terraform init

terraform plan

terraform apply

Enter a value : yes

Go to instance

14. Running the NGINX Server in a Docker Container and changing the content of the index.html page from the running container.

STEP 1:
Create an Ubuntu instance and get it connected to MobaXterm -

STEP 2:
COMMANDS :
1. sudo su
2. curl -fsSL https://get.docker.com -o get-docker.sh
3. sh get-docker.sh
4. docker --version
5. docker pull nginx
6. docker run -it -p 80:80 nginx bash
7. service nginx start

(Now copy ipv4 address and paste it in new browser) -
(Get out of the container on shell : Ctrl p + ctrl q) -

8. docker ps
9. docker exec -it <container ID> bash
10. cd /usr/share/nginx/html
11. ls
12. apt-get update
13. apt-get install vim (Enter Y)
14. vim index.html

(Enter i to insert in the editor to write [make any change to that html file] and to save to changes press "esc" and type ":wq" to exit) - (Now refresh the tab where you have pasted the ipv4 add [You will see the changes that u did in html file])

15. read escape sequence (come out of container : ctrl p + ctrl q)
16. docker ps
17. docker stop <container ID>