

AWS & DEVOPS

DAY -1

CLOUD TECHNOLOGY;;;

LIVE COMPANY : HCL COMPANY ,, SHOLLINGANALLUR (ON -PREMISES)

PRICE : 1 CRORE -- OWN ASSET

- 1. SERVER ----PHYSICAL DEV**
- 2.STORAGE -- DATA STORAGE ,, HARDDISK**
- 3.NETWORK -- CONNECTION ,CABLE**
- 4.DATABASE -- TYPE OF STORAGE**
- 5.SECURITY --- SAFETY**
- 6.APPLICATION -- APP**

MAINTENANCE:

- 1.BUILDING RENTAL**
- 2.EB**
- 3.EMPLOYEE SALARY**
- 4. REPAIR**

CLOUD TECHNOLOGY: RENTAL PURPOSE..

RIR

- 1.REMOTE ACCESS**

2.INTERNET CONNECTION

3.RENTAL

WE CAN ACCESS REMOTELY WITH THE HELP OF INTERNET CONNECTION ON RENTAL BASIS...

WE CAN ACCESS

1.SERVER

2.STORAGE

3.NETWORK

4.DB

5.SECURITY

6.APP

ON RENTAL BASIS..... PAY AS YOU GO METHOD...

NEW BUSINESS: EX : SWIGGY

CLOUD SERVICE PROVIDERS:

1.AMAZON - AWS (AMAZON WEB SERVICE)

2.MICROSOFT -- AZURE

3.ORACLE

4.IBM

5.GOOGLE

AWS:

CURRENT TRENDING TECHNOLOGY

NON IT TO IT

REAL TIME PROJECTS ,,MOST OF THE COMPANIES USED AMAZON SERVERS

2006 LAUNCHED

IN INDIA 2016

HIGH SECURITY

30 + COUNTRIES

200+SERVICES

AWS ACCOUNT CREATION,,,

1 YEAR FREE ACCOUNT..

LEARNING PURPOSE...

SERVICE:

1. EC2 - ELASTIC COMPUTE CLOUD

USING THIS EC2 SERVICE WE CAN LAUNCH OUR SERVERS OR INSTANCES.

1.WINDOWS SERVER

2.LINUX SERVER

EC2 IS A REGIONAL BASED SERVICE...

WINDOWS SERVER LAUNCH

LOGIN AS : ROOT USER

USERNAME:

PASSWORD:

SERVICES --> COMPUTE --> EC2

LAUNCH INSTANCE

OPT OUT TO THE OLD EXPERIENCE

Step 1: Choose an Amazon Machine Image (AMI) -- WINDOWS

Step 2: Choose an Instance Type - t2.micro

Step 3: Configure Instance Details - no of instances : 1

Step 4: Add Storage - 30gb

Step 5: Add Tags -- CLICK TO ADD A NAME TAG --> DHARAKA WINDOWS SERVER

Step 6: Configure Security Group : RDP (REMOTE DESKTOP PROTOCOL) 3389

STEP 7 : REVIEW & LAUNCH

CREATE A NEW KEYPAIR

KEYPAIR NAME : DHARAKAKEY -->DOWNLOAD

DHARAKAKEY.PEM ,,, PEM KEY,, PEM (PRIVATE ENHANCED MAIL)

SO WE SUCCESFULLY LAUNCHED SERVER

NOW THE INSTANCE STATE IS " RUNNING "

STATUS CHECK : 2/2 CHECK PASS (INSTALLED & CONFIGURED)

NOW WE LOGIN THE "DHARAKA WINDOWS SERVER "

TO LOGIN WE NEED

1. USERNAME: Administrator

2.PASSWORD: IV*?FrpYdbJX-iaDhwW%5zAq6zMbX*xw

3.PUBLIC IP: 13.211.33.74

TO LOGIN WE NEED A TOOL: REMOTE DESKTOP CONNECTION

2. LINUX SERVER

Step 4: Add Storage -- 8 GB

Step 6: Configure Security Group - SSH (SECURE SHELL) - PORT NO: 22

CHOOSE AN EXISTING KEYPAIR

SELECT OUR PEMKEY (DHARAKAKEY)

LOGIN:

USERNAME : ec2-user

NO PASSWORD

PUBLIC IP : 13.239.137.94

PPK (PUTTY PRIVATE KEY)

I HAVE NO PPK KEY NOW... WE HAVE ONLY PEM KEY..

WE CONVERT PEM INTO PPK

FOR CONVERSION WE HAVE A TOOL : PUTTY GEN (TOOL)

OPEN PUTTY GEN -->LOAD -->SELECT OUR PEM KEY-->OK--> SAVE PRIVATE KEY--->DHARAKAKEY

NOW WE LOGIN THE LINUX SERVER:

PUTTY:

IP:

SSH -->AUTH -->PPK

login as : ec2-user

To Become a root user

sudo su -

To check the storage

Lsblk

ADVANCE EC2...

WINDOWS -- 30 GB

LINUX -- 8 GB

HOW TO ADD EXTRA VOLUME (STORAGE) ... EBS VOLUME- ELASTIC BLOCK STORAGE

1. ADD A NEW VOLUME WHEN WE LAUNCH A INSTANCE.

STEP 4 : ADD NEW VOLUME

30 GB

LOGIN :

USERNAME : Administrator

PASSWORD : &RnT)bJyWjuV)N.f.jyA82Z;3eGJcF?s

PUBLIC IP : 3.144.29.129

LOGIN

diskmanagement

offline -->INITIALIZE DISK-->online -->new simple volume --->ntfs-->ok

NTFS ---NEW TECHNOLOGY FILE SYSTEM

2.ADD EBS VOLUME IN A RUNNING INSTANCE.

ELASTIC BLOCK STORAGE -->VOLUME -->CREATE VOLUME --->SIZE 30 GB

NOW THE STATE IS AVAILAVLE

ATTACH THE NEW VOLUME IN THE INSTANCE

NOW THE STATE IS INUSE

3. SNAPSHOT --- VOLUME BACKUP

SNAPID snap-09728596cf12fbb72

LINUX SERVER LAUNCH

STEP 4 : ADD STORAGE : 8GB

LOGIN

1.USERNAME

2. NO PASSWORD ----> BUT WE NEED PPK (PUTTY PRIVATE KEY)

3. PUBLIC IP

WE HAVE ONLY PEM KEY..... WE CONVERT PEM INTO PPK...

TOOL: PUTTY GEN

PUTTY GEN IS USED TO CONVERT PEM INTO PPK

LOGIN : PUTTY TOOL

S3 - SIMPLE STORAGE SERVICE

- REMOTE STORAGE

-2006 AWS LAUNCHED THE FIRST SERVICE (S3)

-S3 COMES UNDER STORAGE ENGINEERING

SERVICE SET MODELS : IAAS (INFRA)

EC2 LAUNCH --- WINDOWS -30GB

LINUX - 8GB

STORAGE IN THE EC2,,, INSIDE THE SERVER

S3 IS A REMOTE STORAGE...

WE CAN ACCESS FROM ANYWHERE AT ANY TIME

UNLIMITED STORAGE...

FREE TIER - 8GB

HIGH SECURITY

HIGH AVAILABILITY

HIGH DURABILITY.

REAL TIME...

DEVELOPERS SAVES THERE SOURCE CODE IN S3...

ALSO STATIC WEBSITE HOSTING IS POSSIBLE

S3 IS A REPOSITORY ..

ONLINE REPOSITORY

WE CAN STORE OUR DATA IN DIFFERENT VERSIONS.

LOW COST...

EASY ACCESSABLE SERVICE.

S3 IS A GLOBAL SERVICE.

PRACTICALS :

SERVICE ---> STORAGE --> S3

1. BUCKET (NAME MUST BE UNIQUE)

CREATE BUCKET--->BUCKET NAME (016dharakabucket)--->AWS REGION (SYDNEY)-->OBJECT OWNERSHIP (ACL ENABLE)

--->UNBLOCK THE (BLOCK ALL PUBLIC ACCESS) --> I ACKNOWLEDGE -->CREATE BUCKET

2. OBJECT

CLICK THE BUCKET --->UPLOAD --->ADD FILE -->SELECT ANY FILE -->PERMISSION-->GRANT PUBLIC READ ACCESS --> I UNDERSTAND --->UPLOAD

WE GET ONE OBJECT URL:

<https://016dharakabucket.s3.ap-southeast-2.amazonaws.com/FULLSTACK.txt>

3. BUCKET VERSIONING

BUCKET PROPERTIES --->BUCKET VERSIONING -->EDIT ----->ENABLE

4. STATIC WEBSITE HOSTING

upload one html file

BUCKET PROPERTIES---->STATIC WEBSITE HOSTING --->ENABLE ->HOSTING TYPE -->HOST A STATIC WEBSITE-->

http://016dharakabucket.s3-website-ap-southeast-2.amazonaws.com -> URL

BUCKET PROPERTIES--->STATIC WEBSITE HOSTING --->ENABLE ->HOSTING TYPE -->REDIRECT REQUEST FOR AN OBJECT-->

HOST NAME ->WWW.FLIPKART.COM-->PROTOCOL (HTTP) -->SAVE CHANGES

5. STORAGE CLASS

OBJECT -->PROPERTIES

1. S3 Standard -

General purpose storage for any type of data, typically used for frequently accessed data

First 50 TB / Month \$0.023 per GB

Next 450 TB / Month \$0.022 per GB

Over 500 TB / Month \$0.021 per GB

2 .S3 Intelligent - Tiering* -

Automatic cost savings for data with unknown or changing access patterns

Monitoring and Automation, All Storage / Month (Objects > 128 KB) \$0.0025 per 1,000 objects

Frequent Access Tier, First 50 TB / Month \$0.023 per GB

Frequent Access Tier, Next 450 TB / Month \$0.022 per GB

Frequent Access Tier, Over 500 TB / Month \$0.021 per GB

Infrequent Access Tier, All Storage / Month \$0.0125 per GB

Archive Instant Access Tier, All Storage / Month \$0.004 per GB

S3 Intelligent - Tiering* - Optional asynchronous Archive Access tiers

Archive Access Tier, All Storage / Month \$0.0036 per GB

Deep Archive Access Tier, All Storage / Month \$0.00099 per GB

3.S3 Standard - Infrequent Access - For long lived but infrequently accessed data that needs millisecond access**

All Storage / Month \$0.0125 per GB

S3 One Zone - Infrequent Access - For re-createable infrequently accessed data that needs millisecond access**

All Storage / Month \$0.01 per GB

4. S3 Glacier Instant Retrieval***

- For long-lived archive data accessed once a quarter with instant retrieval in milliseconds

All Storage / Month \$0.004 per GB

5.S3 Glacier Flexible Retrieval (Formerly S3 Glacier)***

- For long-term backups and archives with retrieval option from 1 minute to 12 hours

All Storage / Month \$0.0036 per GB

6.S3 Glacier Deep Archive***

- For long-term data archiving that is accessed once or twice in a year and can be restored within 12 hours

All Storage / Month \$0.00099 per GB

ELB->ELASTIC LOADBALANCER

- 1.ELB GATHER & DISTRIBUTE THE REQUEST(TRAFFIC) FROM REMOTE USER (FLIPKART SERVER)**
- 2.PARELLEL PROCESSING IS DONE (2 PROCESS---GATHER&DISTRIBUTE ,,,, HEALTH CHECK))**
- 3.GATHER FROM REMOTE USER AND DISTRIBUTE TO SERVER(FLIPKART SERVER)**
- 4.BEFORE DISTRIBUTION ITS HEALTH CHECK THE SERVER**
- 5.IF THE SERVER IS NOT HEALTHY,ELB INFORM TO AUTOSCALING...AUTOSCALING IS A MASTER...**
- 6.AUTOSCALING DO SCALEUP(NEW SERVER CREATION) AND SCALEDOWN ...**

ELB Definition - Distributes the traffics to "available resources".

Enables parallel processing

Assures fastest system performance

"Customer Satisfaction" and "No Downtime"

"Customer request - Response fast"

NETWORK ENGINEERING.

SERVICE SET MODEL-->IAAS.

4TYPES OF LOAD BALANCERS.

CLASSIC LOADBALANCER

APPLICATION LOADBALANCER

NETWORK LOADBALANCER

GATEWAY LOADBALANCER

PRACTICALS.

CLASSIC LOAD BALANCER CREATION & ENABLE AUTOSCALING

- 1.CLASSIC LB CREATION-INPUT1-HEALTHCHECK**
- 2.LAUNCH CONFIGURATION-BASIC INSTANCE CREATION**
- 3.AUTOSCALING CREATION AND CONFIGURATION**

1.CLASSIC LB CREATION

SERVICE--->EC2--->LOAD BALANCERS-->CREATE LOADBALANCERS-->CREATE CLASSIC LOADBALANCER

STEP1:

LOADBALANCER NAME: CLASSICLB *

CREATE LB INSIDE : DEFAULT VPC

PROTOCOL ->HTTP PORTNO-->80

STEP2: ASSIGN SECURITY GROUP

CREATE A NEW SEURITY GROUP----->SEC GROUP NAME: CLASSIC LBSECGRP *

TYPE: ALL TCP *

STEP 3: CONFIGURE SECURITY SETTINGS

STEP 4: CONFIGURE HEALTH CHECK

PING PROTOCOL -->HTTP

PING PORT --> 80

PING PATH ---> /INDEX.HTML (APP HOMEPAGE)

RESPONSE TIMEOUT--> 5 SECONDS

INTERVAL --> 30 seconds

UNHEALTHY ---> 2 times

HEALTHY -----> 10 times acknowledgement they calculate as healthy

EVERY 30 SECONDS IT HITS THE INDEX.HTML (flipkart server)

WITHIN 5 SECONDS RESPONSE WILL BE COME

INCASE THERE IS NO RESPONSE FOR 2 TIMES...SO IT IS UNHEALTHY

IT WILL BE HEALTHY THRESHOLD WHEN CONSECUTIVE SUCCESS AFTER 10 TIMES.

STEP 5: ADD EC2 INSTANCE (SKIP)

STEP 6: ADD TAGS

STEP 7: REVIEW

CREATE

2.LAUNCH CONFIGURATION: BASIC INSTANCE CREATION

**SELECT LAUNCH CONFIGURATION(INSIDE AG)--->CREATE LAUNCH CONFIGURATION-->NAME
(MYLAUNCH22)-->**

**AMI (COPY THE AMI ID FROM ANY INSTANCE) -->INSTANCE TYPE (T2.MICRO)---> ami-
059af0b76ba105e7e**

ADVANCE DETAILS (BOOTSTRAP)

```
#!/bin/bash
```

```
yum install httpd -y
```

```
service httpd start
```

```
echo "This is my Classic Load Balancer Application" > /var/www/html/index.html
```

--->SECURITY GROUP (SELECT AN EXISTING SECURITYGROUP)---->KEYPAIR--->CREATE LAUNCH CONFIG

3.AUTOSCALING CREATION & CONFIGURATION

EC2->AUTOSCALING GROUP--->CREATE AUTOSCALING GROUP

**NAME: MYAUTOSCALING--->SWITCH TO LAUNCH CONFIG--->(MYLAUNCH22)--->NEXT-->DEFAULT VPC-
->**

**SELECT ALL AVAILABILITY ZONES-->ATTACH THE EXISTING LOADBALANCER->CHOOSE CLASSIC LB-
->**

SELECT LB-->NEXT

CONFIGURE GROUPSIZE & SCALING POLICIES-->GROUP SIZE

DESIRED CAPACITY: 2

MINIMUM CAPACITY: 1

MAXIMUM CAPACITY : 4

SCALING POLICIES

SCALING POLICY NAME : TARGET TRACKING POLICY

METRIC TYPE : AVERAGE CPU UTILIZATION

TARGET VALUE : 50

Condition:-

Scale-up:

If (Cpu utilization >=50% in vm1 and vm2) then

create vm3

if (Cpu utilization >=50% in vm1 and vm2 and vm3) then

create vm4

scale-down:

if (Cpu utilization <=50% in vm1 and vm2 and vm3) then

remove vm4

if (Cpu utilization <=50% in vm1 and vm2) then

remove vm3

This is not possible:

if (Cpu utilization <=50% in vm1) then

remove vm2

--->CREATE AUTO SCALING GROUP

GO TO EC2 AND CHECK,,,2 INSTANCE AUTOMATICALLY CREATED...

RENAME THE INSTANCE

COPY THESE PUBLIC IP AND CHECK IT IN BROWSER

SELECT OUR CREATED LOADBALANCER-->DESCRIPTION,,INSTANCE

INSERVICE:

APPLICATION IS RUNNING THROUGH ELB.

INTERFACE IS AVAILABLE BETWEEN BROWSER AND APPLICATION. (ie)LB AND ASG IS AVAILABLE

OUTSERVICE:

APPLICATION IS NOT RUNNING THROUGH ELB

-->DESCRIPTION..>CLB URL-->COPY AND PASTE IT IN BROWSER...

TASK: SELECT 2 VM'S AND TERMINATE...WHAT HAPPENED.

BY SCALEUP PROCESS WE GET 1 NINSTANCE INITIALLY(MIN CAPACITY IS 1),,,WITHIN A FEW MINUTES

WE GET A ANOTHER INSTANCE (DESIRED CAPACITY 2)

->COPY THE PUBLIC IP AND CHECK..PUBLIC IP IS DIFFERENT, BUT WE GET AN OUTPUT WITH THE HELP OF

LOADBALANCER.

THEORY :

CLASSIC LOAD BALANCER

PROTOCOL: HTTP,HTTPS,TCP

OSI MODEL: TRANSPORT LAYER & APPLICATION LAYER

MECHANISAM : 1 ELB TO 1 APPLICATION

METHOD: GENERAL

APPLICATION TYPE : MONOLYTHIC

MONOLYTHIC ---SINGLE APPLICATION

MICROSERVICES---

CLOUD FRONT

FLIPKART SERVER LOCATED IN AUSTRALIA..

FLIPKART IS A GLOBAL APP...

INDIA ---CHENNAI ---> BOOK ANY ITEM

AUSTRALIA -----> BOOK ANY ITEM

AUSTRALIAN PEOPLE RECEIVE QUICK RESPONSE (NEAR BY LOCATION)

INDIAN PEOPLE RECEIVE SLOW RESPONSE (DISTANCE)

CLOUD FRONT...

AWS PROVIDES BUFFER LOCATION OR EDGE LOCATION

121 EDGE LOCATION

aws.amazon.com/products

**TO SEE THE EDGE LOCATION: NORTH AMERICA,SOUTH AMERICA,EUROPE/MIDDLE EAST,ASIA
PACIFC,CHINA**

TOTAL 121 EDGE LOCATIONS (IQ)

4 PART-----121 EDGE LOCATION...

North America has 25 Availability Zones within seven geographic Regions, with 44 Edge Network locations

South America has 3 Availability Zones within one geographic Region, with 4 Edge Network locations

Europe, Middle East and Africa has 24 Availability Zones within eight geographic Regions, with 39 Edge Network locations

Asia Pacific and China has 29 Availability Zones within 9 geographic Regions, with 34 Edge Network locations

1ST TIME ---> REQUEST GOES TO FLIPKART SERVER (AUSTRALIA)

2ND TIME --->REQUEST GOES TO INDIA (CHENNAI LOCATION)

PRACTICALS:

1. S3 BUCKET CREATION

OBJECT : index.html

OBJECT URL : <https://cloudfrontbucket100.s3.amazonaws.com/index.html>

2.CLOUD FRONT

SERVICES --> NETWORK & CONTENT DELIVERY --> CLOUD FRONT

CREATE CLOUD FRONT DISTRIBUTION -->ORIGIN DOMAIN (mys3bucket)

<https://d28a77hnl9899b.cloudfront.net> (DISTRIBUTION NAME)

WE MANUALLY CREATE ONE CLOUD FRONT URL:

<https://d28a77hnl9899b.cloudfront.net/index.html> -->CLOUD FRONT URL

EBS -- ELASTIC BEAN STALK

DEVELOPER'S SERVICE....

USING EBS SERVICE WE DEPLOY ,TEST ,,ALSO CHECK THE LOGS..

WE HAVE CHECK OUR SOURCE CODE WITH THE EBS SERVICE

SERVICES -->COMPUTE -->EBS

CREATE APPLICATION -->

APPLICATION NAME : webapplication_test

ENVIRONMENT NAME:

PLATFORM : tomcat

PLATFORM BRANCH: tomcat 8.5

PLATFORM VERSION : 4.2.9

.SELECT SAMPLE APPLICATION---->CREATE ENVIRONMENT

GO TO EC2 CONSOLE AND CHECK

GO TO EBS CONSOLE:

RUNNING VERSION-->UPLOAD & DEPLOY--->CHOOSE FILE--->SAMPLE.WAR

COPY THE URL AND CHECK IT IN BROWSER.

CHECK THE LOGS.

CLOUD FORMATION

TO BUILD AN INFRA

IaaS - Infrastructure as a Code

AWS Specific

SERVICES --> MANAGEMENT & GOVERNANCE ->CLOUD FORMATION

USING TEMPLATE WE CREATE AN INFRA

FIRST CREATE A STACK -->PREPARE TEMPLATE -->USE A SAMPLE TEMPLATE

SAMPLE TEMPLATE -->WORDPRESS BLOG --->NEXT --.

STACK NAME : WEB APPLICATION

PARAMETERS

DB NAME : WORDPRESS DB

DB PASSWORD : admin123

DBROOT PASSWORD -- admin123

DB USER : admin

instance type : t2.micro

keyname :

ssh location : 0.0.0.0/0 --->NEXT

CREATE STACK

EFS - ELASTIC FILE SYSTEM

SERVICE SET MODELS : IAAS (INFRA STRUCTURE AS A SERVICE)

STORAGE ENGINEERING

100 SERVERS :

- 1. 1ST TASK -- CREATE ONE DIRECTORY (DIRNAME : USHADIR)**
- 2. 2ND TASK --- FILES CREATION IN THE DIRECTORY**

WE HAVE TO DO THE ACTION IN 1 SERVER----ITS AUTOMATICALLY CREATED IN 99 SERVER..

USING " SHARED VOLUME" WE ACHEIVE THE TASK..

EFS IS A PAID SERVICE...

CONFIGURED ONLY IN LINUX ENVIRONMENT.

NFS PROTOCOL (NETWORK FILE SHARING PROTOCOL)

PRACTICALS :

2 LINUX SERVER'S LAUNCH \$ LOGIN

SECURITY GROUP :

ADD RULE: ENABLE NFS

3.EFS

**CREATE FILE SYSTEM (NAME)-->REGIONAL -->GENERAL PURPOSE -->BURSTING -->DEFAULT VPC ---
>SECURITY GRP :**

MYNFSSECGRP --> CREATE

4. LOGIN MACHINE1:

mkdir new

LOGIN MACHINE2

`mkdir test`

5. EFS -->SELECT THE FILESYSTEM -->ATTACH -->USING NFS CLIENT

COPY

```
sudo mount -t nfs4 -o  
nfsvers=4.1,rsiz=1048576,wsiz=1048576,hard,timeo=600,retrans=2,noresvport fs-  
0268c3b33af2e62bc.efs.ap-southeast-2.amazonaws.com:/ efs
```

6. GO TO MACHINE 1

`cd new`

```
new ] sudo mount -t nfs4 -o  
nfsvers=4.1,rsiz=1048576,wsiz=1048576,hard,timeo=600,retrans=2,noresvport fs-  
0268c3b33af2e62bc.efs.ap-southeast-2.amazonaws.com:/ new
```

7. GO TO MACHINE 2

`cd test`

```
test ] sudo mount -t nfs4 -o  
nfsvers=4.1,rsiz=1048576,wsiz=1048576,hard,timeo=600,retrans=2,noresvport fs-  
0268c3b33af2e62bc.efs.ap-southeast-2.amazonaws.com:/ test
```

GO TO MACHINE 1

`new] mkdir rubydir`

GO TO MACHINE 2

`test] ls`

OUTPUT : rubydir

`df -h -----> TO VIEW THE FILE SYSTEM`

df -->disk fragmentation

h -> human readable

TO REMOVE THE FILE

rm -f filename

IAM - IDENTITY & ACCESS MANAGEMENT..

SECURITY ENGINEERING

SERVICE SET MODELS : PAAS

SERVICES -->SECURITY , IDENTITY & COMPLIANCE ---> IAM

ROOT USER HAS ALL PRIVILEGES TO ACCESS AWS ACCOUNT....

PRACTICALS;

1. IAM URL

Sign-in URL for IAM users in this account

<https://060591852750.signin.aws.amazon.com/console>

ACCOUNT ID : 060591852750 --- 12 DIGIT NUMBER

2. HOW TO SET AN ALIAS NAME :

INSTEAD OF ACCOUNT ID,,WE SET 1 ALIAS NAME

ACCOUNT ALIAS --> CREATE -->PREFERRED ALIAS --> dharaka

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

3. HOW TO CREATE A GROUP

USERS GROUP -->CREATE GROUP --->GROUPNAME (019dharakaec2grp) -->ATTACH PERMISSION POLICIES ---> SELECT

AMAZON EC2 FULL ACCESS (AWS MANAGED POLICY)-->CREATE GROUP

4. HOW TO CREATE A USER

USERS-->ADD USER --> USERNAME -->ENABLE ACCESS KEY-->ENABLE PASSWORD -->CUSTOM PASSWORD-->DISABLE REQ PASSWORD RESET -->NEXT PERMISSION--->SET PERMISSIONS-->ADD USER TO THE GROUP-->SELECT 019dharakaec2grp -->

Next

DOWNLOAD.CSV

LOGIN AS A IAM USER

RAM USER ACCESS ONLY EC2...NOT S3

5. HOW TO ADD EXTRA PERMISSION TO USER "RAM"

WE ADD S3 ACCESS POLICY TO RAM

POLICY : CUSTOMIZE POLICY OR AWS MANAGED POLICY (AMAZON S3 FULL ACCESS)

HOW TO CREATE A NEW POLICY

POLICY -->CREATE POLICY ->SERVICES (S3) -->ACTIONS (ALL S3 ACTIONS) -->RESOURCES (ALL)-->NAME (019dharakas3policy) -->CREATE POLICY

6.ATTACH THE POLICY TO THE USER " RAM "

SELECT & OPEN THE USER --->ADD PERMISSION-->ATTACH EXISTING POLICY DIRECTLY --->SELECT OUR 019dharakas3policy->

NEXT-->ADD PERMISSIONS

LOGIN AS IAM USER

7 .ROLES :

LAUNCH 2 EC2 SERVERS.

login:

aws s3 ls

HOW TO CREATE A ROLES

ROLES ---> CREATE ROLE --->COMMON USE CASE --->SELECT EC2 -->SELECT S3 POLICY (019dharakas3policy)--> ROLE NAME -> MYNEWROLE

8. ATTACH THE ROLE TO THE EC2 MACHINE

SELECT THE WITHROLE SERVER -->ACTION-->SECURITY -->MODIFY IAM ROLE -->SELECT THE ROLE

LAMBDA

->ITS A ENHANCEMENT SERVICE

PRIMARY SERVICES (INFRA)->EC2,STORAGE,NETWORK

->LAMBDA IS A ADDITIONAL SERVICE AND A OPTIONAL SERVICE

->IN REAL TIME DEVELOPERS USED THIS SERVICE.
->LAMBDA DESIGNED MAINLY FOR DEVELOPERS
->AS A CLOUD ENGINEER,WE KNOW THE CONCEPT
->ITS A SERVERLESS TOOL.

NOTED: NO HANDSON EXPERIENCE ON LAMBDA
KNOWLEDGE ON LAMBDA

EX: WHAT'S UP:

PRIMARY WORK -->CHATING

ADDITIONAL WORK -->STATUS,DP,VIDEO SHARING

HOW MANY CODES RUNNING IN WHAT'S UP ?????

276 CODES

WHEN WE PUSH 276 CODES IN A SINGLE EC2 SERVER,WHAT HAPPENED??

SLOW, LOW PERFORMANCE,,, CRASHED

EX: YOU HAVE IMAGE,ITS SIZE IS 7MB,,,WHEN YOU UPLOAD TO WHAT'S UP,IT WILL BE COMPRESSED IN KB

THEY USE IMAGE FLATTENING CODE USED TO COMPRESS.

-->LAMBDA IS A INTELLIGENT SERVICE

-->IT DO AUTOSCALLING AUTOMATICALLY,,WHEN WE UPLOAD MORE NUMBER OF IMAGES IN A FESTIVAL TIME...

-->IN REALTIME,, INTEGRATE CLOUDWATCH AND LAMBDA...

WHEN ANYONE UPLOAD JPEG OR ANY IMAGE RELATED FILE,,SUDDENLY CLOUDWATCH TRIGGER THE LAMDBA TO

DO (RUN THE CODING)THE WORK.

-->LAMBDA IS NOT A FREE SERVICE.

IN REAL TIME,,THE ADDITIONAL CODES ARE RUN IN LAMBDA..

LET'S YOU RUN CODE WITHOUT THINKING ABOUT SERVERS.

PRACTICALS:

SERVICES-->COMPUTE-->LAMBDA-->CREATE FUNCTION

1.AUTO FROM SCRATCH (DEVELOPER'S OWN CODE + OWN EXECUTE)

2.USE A BLUEPRINT (EXISTING CODES BY AWS)

3.CONTAINER IMAGES(DEVOPS)

4.BROWSE SERVERLESS APP REPOSITORY (PUBLIC IMAGES OR PUBLIC CODES) 769 CODES

STEP1:

CREATE ROLE IN IAM

SERVICE-->SECURITY-->IAM-->ROLES-->CREATE ROLE:

COMMON USE CASE : SELECT LAMBDA-->NEXT PERMISSION-->SELECT

1.AMAZON VPC FULL ACCESS,2.CLOUD WATCH FULL ACCESS,3.AMAZON EC2 FULL ACCESS

ROLE NAME-->CREATE ROLE.

STEP2:

GO TO LAMBDA CONSOLE---CREATE FUNCTION-->SELECT AUTO FROM SCRATCH-->BASIC INFO:

FUNCTION NAME: myfunction

RUN TIME : python 3.9

PERMISSION :

CHANGE DEFAULT EXECUTION ROLE

USE AN EXISTING ROLE : ROLENAM ---->CREATE FUNCTION

SELECT THE CODE MENU

COPY YOUR NEW CODE

code1:

```
import json
```

```
def lambda_handler(event, context):
```

```
    # TODO implement
```

```
    return "Hello Lambda"
```

FOR SAVING THE CODE CLICK "DEPLOY".

OUTPUT: SUCCESSFULLY UPDATE THE FUNCTION " FUNCTION NAME"

FOR EXECUTE : CLICK "TEST"

EVENT NAME: mytestcode -->create-->TEST

WE GET THE NEW LAMBDA FUNCTION AND EXECUTION RESULT.

REAL TIME SCENARIO:

SPRINT CLEARANCE: DAILY SPRINT, WEEKLY SPRINT, MONTHLY SPRINT.

MANUAL PROCESS DONE BY DEVELOPERS.

NOW A DAYS THE SPRINT CLEARANCE WORKS GOING ON IN LAMDA.

THEY CREATE A CODING FOR SPRINT CLEARANCE IN LAMDA.

EXAMPLE: DELETE THE LOADBALANCERS AUTOMATICALLY WITH THE HELP OF LAMDA SERVICE.

PRACTICAL 2: DELETE THE LOADBALANCER AUTOMATICALLY WITH THE HELP OF LAMDA

1.CREATE ONE LOADBALANCER

2.CREATE A CODE FOR DELETE THE LOADBALANCER

3.DEPLOY THE CODE AND EXECUTE.. (CROSS CHECK THE CONFIGURATION SETTING) TIMEOUT FOR RUNNING THE CODE

<https://crontab.guru/> ->cronjob assign

code:2

```
import json
```

```
def lambda_handler(event, context):  
    # TODO implement  
    return {  
        'statusCode': 200,  
        'body': json.dumps('Hello from Lambda!')  
    }
```

RDS (RELATIONAL DATABASE SERVICE)

6 TYPES OF ENG.

DATABASE ENGINEERING.

IAAS (NETWORK , STORAGE ,, SERVER)

PAAS(PLATFORM AS A SERVICE) --- SECURITY & DATABASE

SAAS (APPLICATION)

1.DBMS (DATABASE MANAGEMENT SYSTEMS)

2.PURPOSE: HANDLING THE SYSTEM DATA

3.OPERATIONS: READ , WRITE

4.DATABASE TYPES

STRUCTURED - RDS

SEMI STRUCTURED - DYNAMO DB

UNSTRUCTURED - REDSHIFT

5.CAN WE RUN ONE SINGLE DB AS A INSTANCE????

NO

6.SUBNET GROUPING ????

7.PRIMARY END POINT : INTERFACE BETWEEN DB AND EC2 INSTANCE.

PRACTICAL:

STEP 1 : CREATE SUBNET GROUPING

ENABLES MULTI-AVAILABILITY ZONE DEPLOYMENT OF OUR DB

SUBNET GROUPING IS MAINLY FOR ENABLE PRIMARY AND REPLICA DATABASE.

**SERVICE-->DATABASE-->RDS-->SUBNET GROUPS-->CREATE DB SUBNET GROUP-->NAME :
mysubnetgroup -->**

DESCRIPTION : mysubnetgroup-->VPC : default vpc-->ADDSUBNETS (SELECT ALL AVAILABILITY ZONE & ALLSUBNETS)-->CREATE

STEP 2 : DATABASE CREATION

DATABASES-->CREATE DATABASE-->SWITCH TO YOUR ORIGINAL INTERFACE(NEW VIEW)--->SELECT ENGINE-->MYSQL-->

NEXT--->USE CASE (SELECT DEV/TEST/MYSQL)-->NEXT-->

SPECIFY DB DETAILS:

LICENSE MODEL : general-public-license

DB ENGINE : mysql-community engine

DB ENGINE VERSION: mysql 8.0.27

DB INSTANCE CLASS : db.t2.micro

MULTI-AZ DEPLOYMENT : no

STORAGE TYPE : general purpose ssd (20)

STORAGE AUTOSCALE : disable

SETTING:

DB INSTANCE IDENTIFIER : hemanth

MATER USERNAME: admin

MASTER PASSWORD: admin123

CONFIGURE ADVANCE SETTING:

NETWORK & SECURITY

VPC : DEFAULT VPC

SUBNET GROUP : MYSUBNETGROUP (FOR ENABLE PRIMARY & REPLICA)

PUBLIC ACCESS : NO

AVAILABILITY ZONE : 1A

VPC SECURITY GROUP : create secgrp (all tcp)

DATABASE OPTIONS

DATABASE NAME : hemanthdb

PORT : 3306 (MYSQL PORTNO)-->CREATE DATABASE

STEP 3: CREATE EC2 AMAZON LINUX SERVER

EC2-->LAUNCH INSTANCE--->TAGS: RDS SERVER-->SEC GROUP (ALL TCP)-->REVIEW & LAUNCH

STEP 4: INSTALL MYSQL LISTENER

PUTTY LOGIN--->LOGIN AS : ec2-user

sudo su -

mysql

mysql --version

yum install mysql -y

mysql output is can't connect

STEP 5: DATABASE CATALOG USING ENDPOINT

OPEN THE DATABASE-->COPY THE ENDPOINT

mathan.ctu18ygpftpq.ap-southeast-2.rds.amazonaws.com

GO TO LINUX SERVER TERMINAL

root] mysql -h mathan.ctu18ygpftpq.ap-southeast-2.rds.amazonaws.com -P 3306 -u admin -p

ENTER PASSWORD : admin123

MYSQL IS A FRAMEWORK

SQL (NONE)>


```
SQL>connect databasename
SQL(DATABASENAME) > EXIT
root]
```

STEP 6: CHECK DDL & DML (Tables)

EX:

```
create database,
show database,
use databasename,
create table,
insert rows,
select * from database,
alter table,
update table
```

STEP 7: MASTER & REPLICA

SELECT DATABASE -->ACTIONS-->CREATE READ REPLICA

REPLICA SOURCE : hemanth

DB INSTANCE IDENTIFIER: hemanthreplica

AWS REGION : 1B

CREATE READ REPLICA

SELECT READ REPLICA AND COPY THE ENDPOINT.

hemanthreplica.cmkuskdau15z.ap-south-1.rds.amazonaws.com

mysql -h hemanthreplica.cmkuskdau15z.ap-south-1.rds.amazonaws.com -P 3306 -u admin -p

DELETE THE PRIMARY AND CHECK IT.

delete me

GO TO TERMINAL AND CHECK THE REPLICA DB

1.show databases;

2.use authours;

create database authours;

show databases;

describe databases;

use authours;

create table authour (id int,name varchar(25),email varchar (26));

insert into authour (id,name,email) values('100',"GOPI","awskarthik7@gmail.com");

insert into authour (id,name,email) values('200',"RAM","Damokrishnan@gmail.com");

insert into authour (id,name,email) values('300',"SUDHAKAR","trmohan79@gmail.com");

select * from authour;

describe tables;

select * from users where id='333';

alter table authour rename users;

select * from users;

alter table users add column address varchar(100);

update users set id='777' where id='111';

delete from users where id='777';

create database new;

show databases;

drop database new;

www.w3school.com

DDL - DATA DEFINITION LANGUAGE

CREATE; ALTER; DROP - DROP TABLE USERS; DROP DATABASE AUTHOURS;

DML - DATA MANIPULATION LANGUAGE

INSERT; SELECT; UPDATE; DELETE

ROUTE 53

DOMAIN NAME SERVICE.. (DNS)

WE ASSIGN A DOMAIN FOR THE PARTICULAR IP ADDRESS --- WE SET A DOMAIN FOR YOUR IP ADDRESSES

ROUTE 53

ROUTE MEANS DNS

53 -- PORT NUMBER OF DNS

-->NETWORK ENGINEERING

-->SERVICE SET MODELS : IAAS (INFRA STRUCTURE AS A SERVICE)

WWW.WIPRO.COM

WIPRO -->DOMAIN NAME

WWW. GOOGLE.COM

GOOGLE -->DOMMain NAME

WWW.VELS.UNIV

VELS -->DOMAIN NAME

.COM , .IN , .UNIV , .GOV , .CO.IN =====> ZONE'S

WWW == WORLD WIDE WEB (RECORD)

FULLY QUALIFIED DOMAIN NAME (FQDN) ==> WWW.FLIPKART.COM

1. PURCHASE 1 NEW DOMAIN FROM GODADDY

greenscloud.in

2. HOST THE DOMAIN IN AMAZON WEBSERVICE..

WE GET 2 RECORDS

NS RECORD (NAME SERVER RECORD) - 4 NAME SERVER'S

SOA RECORD (START OF AUTHORITY RECORD)

3. UPDATE THE NAME SERVERS IN GODADDY

4. SERVER LAUNCH ---WE GET 1 PUBLIC IP ADDRESS

step 3 :

```
#!/bin/bash
```

```
yum install httpd -y
```

```
service httpd start
```

```
echo "This is my Facebook Application" > /var/www/html/index.html
```

5. CREATE 1 MANUALL RECORD... SET THE IP ADDRESS INTO THE DOMAIN

CREATE RECORD--> RECORD NAME (www)--> VALUE (IP ADDRESS) -->CREATE RECORD

SQS & SNS

SQS = SIMPLE QUEUE SERVICE

SNS = SIMPLE NOTIFICATION SERVICE

APPLICATION ENGINEERING

SERVICE SET MODELS = SAAS

COMMUNICATION SERVICES..

TYPES OF COMMUNICATION...

1.SYNCRONISED : BOTH SENDER & RECEIVER WILL BE ACTIVE (MOBILE COMMUNICATION)

2. ASYNCRONISED : ANY ONE BE ACTIVE (EX: WHAT'SUP)

ASYNCRONISED

1. ONE TO ONE COMMUNICATION === SQS

2.ONE TO MANY COMMUNICATION == SNS

SQS

- 1. SENDER**
- 2. RECEIVER**
- 3. QUEUE**
- 4. DLQ**

SERVICES ---> APPLICATION INTEGRATION -->SQS

CREATE QUEUE -->STANDARD --> NAME (dharakaqueue) -->CREATE QUEUE

SEND & RECEIVE MESSAGE

SENDER

MESSAGE BODY

HI AWS

RECEIVER

POLL FOR MESSAGES

PURGE --> MESSAGE DELETED

DELETE --> QUEUE DELETED

SNS : SIMPLE NOTIFICATION SERVICE

SENDER - PUBLISHER

RECEIVER - SUBSCRIBER

TOPIC

SERVICES --> APPLICATION INTEGRATION --> SIMPLE NOTIFICATION SERVICE

CREATE TOPIC ---> STANDARD-->CREATE

ADD SUBSCRIBERS (RECEIVERS)

**SUBSCRIPTION-->CREATE SUBSCRIPTION-->TOPIC ARN : -->PROTOCOL (EMAIL)--->ENDPOINT
(EMAILID)--->CREATE SUBSCRIPTION**

@gmail.com

SUBSCRIPTION IS STILL PENDING:

INTEGRATION

GO TO GMAIL-->CONFIRM SUBSCRIPTION.

PUBLISH THE MESSAGE

SNS-->TOPIC-->SELECT TOPIC-->PUBLISH MESSAGE-->SUBJECT-->MESSAGE BODY--->PUBLISH

CHECK IN GMAIL

CLOUD WATCH

SERVER MONITORING TOOL.... IT IS A SERVICE USED FOR MONITORING PURPOSE...

MONITORS 21 SERVER METRICS

SERVICES-->MANAGEMENT & GOVERNANCE-->CLOUD WATCH (ALARMS,,LOGS,,METRICS)

METRICS-->ALL METRICS--->INSTANCE ID-->SEE 17 METRICS-->SELECT CPU TILIZATION

GRAPHED METRICS

ACTIONS->CREATE ALARM

Configure actions

CLOUD FRONT...

OBJECT URL : <https://30mycloudfrontbucket.s3.amazonaws.com/index.html>

WE NEED CLOUD FRONT URL :

Distribution domain name : <https://d1yixmmiiwcrfq.cloudfront.net>

CLOUD FRONT URL : <https://d1yixmmiiwcrfq.cloudfront.net/index.html>

VPC :

VIRTUAL PRIVATE CLOUD

VPC COMES UNDER NETWORK ENGINEERING

SERVICE SET MODELS : IAAS (INFRA STRUCTURE AS A SERVICE)

IP ADDRESS

54.221.1.189 -----> IP CONTAINS 4 PARTS

IP DECIMAL FORMAT BETWEEN (0 TO 255)

VALID IP : 100.50.1.100 200.25.20.254 50.0.0.100

INVALID IP : 300.100.11.1 2.5.5.256

IP CLASSIFICATIONS

CLASS A - N H H H

1 NETWORK PART - 8 NETWORK BITS /8 TOTAL IPS = 16 MILLION IP'S

3 HOST PART - 24 HOSTBITS

CLASS B N N H H

2 NETWORK PART - 16 NETWORK BITS / 16 TOTAL IP'S = 65,536 IP'S

2 HOST PART - 16 HOSTBITS

CLASS C N N N H

3 NETWORK PART - 24 NETWORK BITS /24 TOTAL IP'S = 256

1 HOST PART - 8 HOSTBITS

CLASS D

CLASS E

CIDR VALUES

1. /8 = 16 MILLION IP'S

2. /16 = 65,536 IP'S

3. /24 = 256 IP'S

10.0.0.0 /8 = 16 MILLION IP

10.0.0.0 /16 = 65,536 IP'S

10.0.0.0 /24 = 256 IP'S

PUBLIC IP : ONLY FOR REMOTE ACCESS ... OR REMOTE COMMUNICATION.. MUST INTERNET CONNECTION..

PRIVATE IP : FOR LOCAL COMMUNICATION

VPC --YOU GET AN INTERNET CONNECTION WITH THE HELP OF VPC (DEFAULT VPC)

VPC IS ALSO CALLED AS A NETWORK

VPC IS ACTS AS A DATACENTER..

WITHOUT VPC WE CANNOT GET AN INTERNET CONNECTION....

WE CAN CREATE OUR VPC...AWS CREATE A VPC FOR THE CLIENT (COMPANY)

EX : HDFCBANK.COM

PUBLIC SUBNET : WEBSERVER (EVERYONE ACCESS THE WEBSERVER)

PRIVATE SUBNET : APPSERVER ++ DATABASE (ONLY USERID AND PASSWORD - PREVILEGED USERS ONLY ACCESS)

PRACTICALS :

1.VPC CREATION

2.SUBNET CREATION

3. INTERNET GATEWAY CREATION

ATTACH TO VPC

4.ROUTE TABLE CREATION

4.5 SUBNET ASSOCIATION

5. ROUTES

6. CREATE SECURITY GROUP

7.WEBSERVER CREATION

8.APPSERVER CREATION

9. NAT

10 .NAT ROUTES

Scope of the project.

Restrict Remote user access the private subnet directly

Benifits

Eliminates unwanted traffics hitting on the private subnet

Eliminates to learn about Business using public subnet

Customer Satisfation level always good..

AWS CLOUD SERVICE PROVIDER

1. VPC CREATION---->MY-VPC 10.0.0.0/16; 65536 IP'S FOR WIPRO

IN REAL TIME VPC CREATED BY AWS ADMIN FOR ANY CLIENT.Assume WIPRO is now as a client.

Go to Services--->Network & Content delivery--->vpc-->your vpc--->create vpc-->ipv4 cidr (10.0.0.0/16)-->tenancy (default)-->create vpc

2.CREATE SUBNETS CREATION

WIPRO company allocates 2 subnets for a HDFC as per as global architecture(3 tier)

PUBLIC SUBNET CREATION:-->10.0.1.0/24---256 PRIVATE IP'S

select subnets-->create subnet-->vpc id-->subnet name--(public subnet)-->Avai zn--Cidr (10.0.1.0/24)

PRIVATE SUBNET CREATION:-->10.0.2.0/24---256 PRIVATE IP'S

select subnets-->create subnet-->vpc id-->subnet name--(private sub)-->Avai zn--Cidr (10.0.2.0/24)

We provide internet for WIPRO vpc(WIPRO public internet).We are going to supply internet to vpc through INTERNET GATEWAY and Attch to our VPC

3.INTERNET GATEWAY CREATION-->PUBLIC INTERNET

Select Internet Gateways-->create internet gateway-->name(My-internet-gateway)

INTERNET GATEWAY ATTACHED TO VPC

Go to Actions--->Attach to vpc-->Available vpc--->Attach internet gateway

This is the public internet for CTS vpc

We get an Internet connection for Public and Private subnet.We implement a Router for public subnet & private subnet

4.PUBLIC ROUTE TABLE CREATION

Go to RouteTables-->Create Route table---public Route table

PRIVATE ROUTE TABLE CREATION

Go to RouteTables-->Create Route table---private Route table

ex:meterbox

4.5 PUBLIC ROUTE TABLE TO PUBLIC SUBNET ASSOCIATIONS

Select Public Route table--->SubnetAssociations-->Edit subnetAssosia--->public subnet-->save assosia

PRIVATE ROUTE TABLE TO PRIVATE SUBNET ASSOCIATIONS

Select Private Route table--->SubnetAssociations-->Edit subnetAssosia--->private subnet-->save asso

ex:meterbox to homeline

5. ROUTES INTERNET FROM IGW TO PUBLIC ROUTE

Select Public Route table-->Routes-->Edit Routes-->Add Route--->0.0.0.0/0->Target (Internet Gateway)
mainline to meterbox

Restrict IGW to supply internet to private subnet

6. SECURITY GROUP CREATION BEFORE INSTANCE LAUNCH

PUBLIC SECURITY GROUP CREATION (RDP,HTTP,HTTPS)source--0.0.0.0/0

Go to Security Groups---->create sec group-->name--public security group>select vpc-->InboundRules->Addrule-->Rdp,http,https--create--->Security group id shown(note it)

sg-00c9589e704559a69

COMMON INTERFACE OPEN BETWEEN PUBLIC AND PRIVATE VIA SECURITY GROUP.

.PRIVATE SECURITY GROUP CREATION (All tcp)source--public security groupid (so common gateway will be open between public and private)

Go to Security Groups---->create sec group-->name--private>select vpc-->InboundRules-All tcp--source(copy and paste the security group id)

7. WEBSERVER CREATION

Launch Ec2 instance--public ip=yes,private ip=yes

Step3(Configure instance details)---select vpc-->myvpc,subnet (public subnet),Auto Assign public ip--enable

security group--->public sec group

Check the public ip and private ip.

APPSERVER CREATION

Launch EC2--public ip=no,private ip=yes

Step3(Configure instance details)---select vpc->myvpc,subnet(private subnet),Auto Assign public ip-Disable

security group-->private sec group

check the public ip and private ip

We successfully disabled Autoassigned publicip from global network to private subnet

8.Check internet access in the webserver.Login the web server.

Remote desktop connection----->>>> copy public ip ---->we get a internet connection in windows server.....

WEBSERVER PASSWORD : lqBp4.O3&Y\$mCm\$bJfd\$U-g=xzWg6x**

Check internet access in the Appserver.Login the App server.

Remote desktop connection----->>>>copy private ip--->we cannot get a internet connection in windows server....

appserver password: K(l=)hm-Q2BpZ.9kQNYDDM9XttXjim39

Go to window server 2019 and login once again as a Remote desktop connection and check ????still no internet....

9.NAT GATEWAYS CREATION for internet connectivity to private subnet from public subnet

Go to NAT Gateways-->create Nat Gateway->name(my-nat)--subnet(public subnet)--->connectivity type (public)Allocate elastic ip--create Natgateway

INTRANET ->SUPPLYING INTRANET TO PRIVATE SUBNET FROM PUBLIC SUBNET

CHECK INTERNET OF THE APPSERVER....?? no

10.ROUTE THE INTRANET FROM PUBLIC SUBNET TO PRIVATE SUBNET

ROUTES FROM NAT TO PRIVATE ROUTETABLE

Go to Route table--->select private route table--->Routes--->Edit routes-->Addrule-->Target->Nat gateway->Destination (0.0.0.0/0)-->save

NOW,CHECK THE INTERNET IN THE APPSERVER

.....We completed our project.....

...Finally delete one by one...

DYNAMO DB:

DATABASE ENGINEERING

PAAS (PLATFORM)

SEMI STRUCTURED DATABASE

SERVICES -->DATABASES -->DYANAMO DB

CREATE TABLE -->TABLE NAME (HDFC) -->PARTITION KEY (EMP_ID) ->CREATE TABLE

SELECT THE TABLE -->CREATE ITEM -->ADD NEW ATTRIBUTE -->CREATE ITEM

AWS --- MAIN TOPICS

1. EC2

2. APP HOSTING

3.S3

4.IAM

5.VPC

6.ROUTE 53

7.CLOUD FRONT

8.CLOUD WATCH

9.SNS & SQS

10. ELASTIC LOADBALANCER

11.AUTOSCALING

12. RDS

DOCKER -1

DEPLOYMENT TOOL

GIT (CONTINUOUS SOURCE CODE MANAGEMENT TOOL OR VERSION CONTROL)

FOR CONTINUOUS DEPLOYMENT TOOL..

DEPLOYMENT :

WINDOWS --IIS ----> C: inetpub:wwwroot

LINUX ----- APACHE --->/var/www/html

NOTE: USING DOCKER,,WE CAN INITIATE OR DEPLOY MANY APPLICATIONS

USING SINGLE MIDDLEWARE ,, WE CAN DEPLOY MULTIPLE APP

PRACTICALS :

LAUNCH UBUNTU SERVER.

LOGIN AS : ubuntu

sudo su -

1. FIRST WE NEED TO UPDATE THE SERVER..

] apt-get update LINUX : yum update

2. HOW TO INSTALL DOCKER

]apt install docker.io -y LINUX: yum install docker -y

3.HOW TO CHECK THE VERSION

]docker version

4. CREATE DOCKER HUB ACCOUNT

<https://hub.docker.com/>

5. HOW TO INSTALL HTTPD (APACHE) FROM DOCKER HUB

searchbox --> httpd

]docker pull httpd

]docker images

OUTPUT : To view the image with image id

6.WE CONVERT IMAGE INTO THE CONTAINER

```
] docker run -itd --name dharakacont1 -p "8090:80" httpd
```

7.HOW TO VIEW or LIST OUT THE CONTAINER DETAILS

```
]docker ps
```

HOW TO ACCESS IN BROWSER..

PUBLICIP:8090

OUTPUT: IT WORKS

8. ADD A NEW CONAINER

```
] docker run -itd --name agilacont1 -p "9090:80" httpd
```

publicip:9090

9. HOW TO LOGIN THE CONTAINER..

NOW I LOGIN THE CONTAINER 2 (agilacont1)

```
]docker exec -it container2id /bin/bash
```

bac0eed8d94c --- my container2 id

]ls

]cd htdocs

]ls

]vi index.html

]apt-get update

]apt install vim -y

]vi index.html

IT'S A NEW CONTAINER

:wq!

PUBLICIP:9090

WE GET A NEW OUTPUT

10. HOW TO LOGOUT FROM CONTAINER

]exit

11. HOW TO STOP THE PARTICULAR CONTAINER

]docker stop containerid

12. HOW TO START THE CONTAINER

]docker start containerid

13. HOW TO KILL THE CONTAINER (TEMPORARY REMOVAL)

]docker kill containerid

]docker ps -a

]docker start containerid

14. HOW TO REMOVE

]docker rm containerid

**output : You cannot remove a running container
bac0eed8d94ccfbfbfeabe5f4e60994f551efa228ea888f95fb5ad6725db7528. Stop the container before
attempting removal or force remove**

] docker stop containerid

]docker rm containerid

15. HOW TO REMOVE THE RUNNING CONTAINER

]docker rm -f containerid

HOW TO CHECK THE CONTAINER DETAILS:

]docker stats

ctrl c

]docker logs containerid

]docker top

]docker info

]docker inspect

docker run -itd --name sundarcon1 -p "7090:80" httpd

DOCKER -2

]docker images

]docker run -itd --name dharakacont1 -p "9090:80" httpd

]docker exec -it ce1d6eb7d33c /bin/bash

]wget

OUTPUT: COMMAND NOT FOUND

SO WE INSTALL "wget"

]vim

OUTPUT: COMMAND NOT FOUND

INSTALL WGET AND VIM IN THE CONTAINER

```
]apt install wget vim -y
```

EXIT FROM CONTAINER

```
]exit
```

SO WGET ,VIM ARE NOT IN IMAGE...SO WE COMMIT

```
]docker commit containerid myimage
```

```
]docker images
```

HOW TO BACKUP THE IMAGE

```
]docker save -o /root/myimagebackup.tar myimage
```

HOW TO REMOVE THE IMAGE

```
]docker rmi imageid
```

```
]docker images
```

HOW TO RECOVER THE IMAGE

```
]docker load -i myimagebackup.tar
```

HOW TO TAKE A IMAGE AS A CLOUD BACKUP

TASK: PUSH THE IMAGE FROM DOCKER SERVER TO DOCKER HUB,,,SO WE HAVE TO TAGGING

]docker login

username: (dockerhub username)

passowrd (dockerhub password)

output: login succed

]docker tag myimage gopal2427/01novmyimage (01novmyimage is a new image name)

]docker push gopal2427/01novmyimage

GO TO DOCKER HUB AND CHECKED...

1. HOW TO CREATE A DOCKER FILE

]mkdir dharakadir

]cd dharakadir

vi Dockerfile

FROM ubuntu:18.04

MAINTAINER siva

RUN apt update && apt install -y vim wget

:wq!

cat Dockerfile

2. NOW WE CONVERT DOCKER FILE INTO IMAGE

```
]docker build -t vimwgetimage .
```

```
]docker images
```

3. RUN AS A CONTAINER

```
]docker run -itd --name dharakacont2 -p "8090:80" vimwgetimage
```

```
]docker ps
```

4.LOGIN AS A CONTAINER

```
]docker exec -it containerid /bin/bash
```

```
]exit
```

HOW TO INSTALL GIT THROUGH DOCKER FILE

IMAGENAME -->vimwgetimage

```
]mkdir ram
```

```
]cd ram
```

```
]vi Dockerfile
```

FROM vimwgetimage

MAINTAINER dharaka

RUN apt install git -y

```
docker build -t vimwgetimage .
```



```
docker run -itd --name dharakacont3 -p "7090:80" vimwgetimage
```

```
docker exec -it 889d7884787e /bin/bash
```

```
git --version
```

GIT

->EASIEST & IMPORTANT TOOL.

->SOURCE CODE MANAGEMENT TOOL.

-->NON EXECUTABLE CODE TO EXECUTABLE CODE (JAR--JAVA ARCHEIVE,,WAR--WIN ARCHIVE OR WEB ARCHIVE)

->ITS ACT AS A VERSION CONTROL SYSTEM.(VERSION1-----etc)COMMIT ID.

DEVOPS STAGES:INTERVIEW QUESTION

1.CONTINUOUS DEVELOPEMENT--developer. Developer uses GIT(PACKAGE) AND GITHUB(PUBLIC or ONLINE REPOSITORY)

2.CONTINUOUS VERSION CONTROLLING SYSTEM (CVCS)versionwise....GIT

3.CONTINUOUS TESTING

4.CONTINUOUS MONITORING

5.CONTINUOUS INTEGRATION

6.CONTINUOUS DEPLOYMENT

TRADITIONAL MECHANISAM: SOURCE CODE MANAGED BY THERE OWN LOCAL MACHINES...

1.ITS A LESS SECURITY.

2.SLOW PROCESS

3.DIFFICULT TO MAINTAIN

4.AGILE AND WATERFALL MODEL.

NEW MECHANISAM: VCS

WHAT IS VERSION CONTROL SYSTEM?

RECORDS THE CHANGES VERSION WISE..ANY FAILURE WE RECOVER FROM OUR PREVIOUS VERSIONS.

TYPES OF VCS:

1: LOCAL VERSION CONTROL SYSTEM

SAME AS TRADITIONAL

2:CENTRALISED VERSION CONTROL SYSTEM

SINGLE SERVER IS USED,,SAVE IT IN VERSIONWISE

3.DISTRIBUTED VERSION CONTROL SYSTEM:

HIGHLEVEL SECURITY & MAXIMUM ERRORS ELIMINATION EX: GIT

GIT:

->LAUNCHED IN 2005 BY LINUS TORVALDS ,,,HE DEVELOPED LINUX IN 1991...OPEN SOURCE..

->FREE & OPEN SOURCE DISTRIBUTED VCS

->GIT IS A MASTER OF SOURCE CODE MANAGEMENT.

DIFFERENCE BETWEEN CENTRALISED VCS VS DISTRIBUTED VCS

CENTRALISED VCS

1.EVERYTHING IS STORED IN ONE CENTRALISED REPOSITORY

2.NO LOCAL REPOSITORY

DISTRIBUTED VCS

STAGING AREA (TEMPORARY)

LOCAL REPOSITORY

OS SUPPORTS

1.LINUX (DEBIAN)

2.LINUX (FEDORA)

3.LINUX (CENTOS)++REDHAT

4.MICROSOFT

5.MAC

GIT PLATFORM INDEPENDENT & SOURCE CODE INDEPENDENT

USERS: THEY ALL MANAGED THERE SOURCE CODES IN GIT

1.MICROSOFT

2.FACEBOOK

3.AMAZON

4.LINKED IN

5.YAHOO

6.ACCENTURE

GIT ARCHITECTURE:

MOTTO: SOURCE CODE TO APPLICATION PROJECT IS A PRIMARY MOTTO

1.DEVELOPER DETAILS:

PROJECT : LOAN

DOMAIN : BANKING

MODULE : LOAN BOOKING

DEVLOPER :AZHAR

MVC MODEL : JAVA FRAMEWORK

PROGRAM NAME: LOAN BOOKING SESSION BEAN.JAVA (SOURCE CODE)

EDITOR TOOL : VISUAL STUDIO CODE

IN DEVELOPER'S AZHAR'S LOCAL MACHINE

2.GIT SERVER OR REMOTE SERVER:

EC2

DEVELOPER DIRECTORY (AZHARDIR)

LOCAL REPOSITORY (RECORDS THE CHANGES)

COMMIT: IS NOTHING BUT SAVE THE CHANGES

3.REMOTE REPOSITORY (GIT HUB)

GIT STAGES;

**WORKING DIRECTORY ----->STAGING AREA----->LOCAL REPO----->REMOTE
REPO---->Take a clone cp
(git add) (commit) (git push)**

GIT HUB IS A PUBLIC REPOSITORY ---open source

NEXUS IS A PRIVATE REPOSITORY ---paid tool,separate tool

DIFFERENCE BETWEEN GIT AND GITHUB (INTERVIEW QUESTION)

GIT

- 1. IT IS A SOFTWARE (INSTALLATION)**
- 2.IT IS INSTALLED LOCALLY ON SYSTEM**
- 3.CLI TOOL**
- 4.MASTER COPY**

GITHUB

- IT IS A SERVICE,,CREATE MANUALLY**
- IT IS A HOSTED ON A WEB**
- GRAPHICAL INTERFACE**
- REPLICA**

PRACTICALS:

- 1.NEED A SOURCE CODE FROM DEVELOPER PREM**
- 2.UPLOAD THE SOURCE CODE TO EC2 INSTANCE**

STEP1:

LAUNCH LINUX EC2 MACHINE

BOOTSTRAP: LAUCH+INSTALL+LOGIN VS NORMAL INSTALLATION : LAUNCH + LOGIN + INSTALL

#!/bin/bash

yum install git -y

TAG : GIT SERVER

SECURITY GROUP : ALL TCP (NAME:GIT-SECGRP)

REVIEW & LAUNCH

PUTTY LOGIN

]sudo su -

root]git --version

STEP2:

CREATING DIRECTORY FOR THE DEVLOPER..BY GIT ADMIN

]mkdir dharakadir

]cd dharakadir

]

STEP3:

WE MUST INITIATE A LOCAL REPOSITORY.

]git init ---->not now..

BEFORE THIS COMMAND,,WE WILL CREATE A INTERLINK BETWEEN PRIVATE(DEVELOPER CREDENTIAL) AND REMOTE

REPOSITORY.

WE MUST INTEGRATE GIT SERVER<----->GIT HUB

PREREQUEST: OPEN THE GITHUB ACCOUNT(REMOTE REPO)

www.github.com

dharakadir]

git config --global user.name "Dharakasundar"

git config --global user.email "Dharakasundar1@gmail.com"

git config --global core.editor vim

git config --global core.compression 2

git config --global diff.tool vim.diff

FOR CHECK:

ramdir]git config -l (small L)

TO INITIATE A LOCAL REPOSITORY

ramdir]git init

OUTPUT: initialized empty git repository in /root/ramdir/.git/

SOURCE CODE'S COMMITWISE RECORDING WILL BE TAKING PLACE OVER LOCAL REPOSITORY..

]ls -la ->to view the hiddenfiles,,,git is a hidden file.

]cd .git/

.git]ls

OUTPUT: branches config descrip etc

BARE REPOSITORY: LOCAL REPO

NON BARE REPOSITORY : IS A WORKING DIRECTORY

]cd ..

dharakadir]ls

STEP4:UPLOAD THE SOURCE CODE BY WINSCP

COPY THE SOURCE CODE PUSH TO /home/ec2-user

```
]cd /home/ec2-user
```

```
ec2-user ]ls
```

OUTPUT: loanbooking sessionbean.java

STEP 5:

COPY THE SOURCE CODE FROM /home/ec2-user to our WORKING DIRECTORY dineshdir)

```
]cd - [previous working directory]
```

```
dharakadir]ls
```

output:

```
dharakadir]cp /home/ec2-user/LoanBookingSessionBean.java .
```

```
ls
```

output: LoanBookingSessionBean.java

NOW THE SOURCE CODE ARE IN WORKING DIRECTORY...(dharakaworking directory)

GIT 2

NOW THE SOURCE CODE ARE IN DEVELOPER WORKING DIRECTORY.

STEP1: STAGING AREA

```
dharakadir]#git status
```

output: untracked files,no commit

```
ramdir]#git add LoanBookingSessionBean.java
```

```
ramdir]#git status
```

"git add" command helps to move working dir to staging area

output:no commits yet

STEP2:LOCAL REPOSITORY

dharakadir]#git commit -m "dharaka45 First Commit-Loanbooking"

Staging area to Local repository

ramdir]#git log --oneline

output: To view the commit id and also we have 1 master added automatically.

ramdir]#git show #####(commitid) ->It shows the source code,,commit wise info

ramdir]#git status

output: onbranch master,nothing to commit

ramdir]#git branch

output: *master (green color)

Now our source code are in master branch local repository

STEP3:CREATE A REMOTE OR ONLINE REPO URL

OPEN THE GITHUB REMOTE REPOSITORY

www.github.com

Go to your repositories-->new-->Repository name:dharaka -->public->url

https://github.com/gopalgurukirupa/dharaka.git

STEP4:CREATE A FEATURE BRANCH.

Because developer commit the source codes only in feature branch

ramdir]#git branch dev

ramdir]#git branch

output: dev

***master(greencolor)**

Now we are in master branch

STEP5: SWITCH OVER FROM ONE BRANCH TO ANOTHER BRANCH (ie master branch to feature branch)

```
ramdir]#git checkout dev
```

output: switched to branch 'dev'

```
ramdir]#git branch
```

output: *dev(greencolor)

master

Now we are in feature branch "dev"

```
ramdir]#git log --oneline
```

output:commit id comes from master..

STEP6: DO CHANGES IN MY SOURCE CODE.EDIT THE SOURCE CODE.

```
import java.io.outputprinter
```

UPLOAD THE NEWSOURCE CODE THROUGH WINS CP ONCE AGAIN.

```
ramdir]#cd /home/ec2-user
```

```
ec2-user]#ls
```

```
ec2-user]#vi Loanbooking sessionbean.java
```

```
ec2-user]#cd -
```

```
dharakadir]#cp /home/ec2-user/Loanbookingsessionbean.java .
```

```
ramdir]#git status
```

STEP7:COMMIT TO LOCAL REPOSITORY

" New change now in working directory". Now we commit directly to Local repo.

```
dharakadir]#git branch
```

output: *dev

```
dharakadir]#git commit -am "DHARAKA-Second Commit-Loanbooking"
```

```
ramdir]#git log --oneline
```

output: 2 commits in feature branch 'dev'

STEP8: BUT IN MASTER BRANCH WE HAVE ONLY ONE COMMIT.

HOW TO CHECK;

```
ramdir]#git checkout master
```

```
ramdir]#git log --oneline
```

output: only one commit.

NOTED:BUT PUSH OPERATIONS FROM LOCAL TO REMOTE REPOSITORY TAKES PLACE ONLY IN MASTER BRANCH.

NOW FEATURE BRANCH HAS 2 COMMITS AND MASTER HAS ONLY ONE COMMIT

STEP9:WE USE MERGE COMMAND FOR COMMIT SYNC BETWEEN MASTER AND FEATURE...

MERGE COMMAND USE ONLY IN MASTER BRANCH

```
ramdir]#git merge dev
```

```
ramdir]#git log --oneline
```

output: 2 commits (master)

STEP10: PUSH THE SOURCE CODE TO REMOTE REPOSITORY

OPEN THE GITHUB:

```
git remote add origin https://github.com/gopalgurukirupa/dharaka.git
```

```
https://github.com/gopalgurukirupa/dharaka.git
```

WE MUST SET THE REMOTE VARIABLE FOR THE ABOVE URL.

SET ANY VARIABLE: 5pmdharakabatch

```
dharakadir]#git remote add origin https://github.com/Dharakasundar/Dharaka66.git
```

ramdir]#git push origin master

output:

username:

password:

NOTE:USE ONLY PERSONAL ACCESS TOKEN

STEP11:

**GO TO GITHUB-->SETTINGS-->DEVELOPER SETTING-->PERSONAL ACCESS TOKEN-->GENERATE NEW
TOKEN-->PASSWORD**

-->NOTE:MYTOKEN->SELECT ALL SCOPES->GENERATE TOKEN-->COPY THE TOKEN

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

dharakadir]#password : paste the token

CHECK IT IN GIT HUB :GO TO YOUR REPOSITORY-->

login as : ec2-user

sudo su -

]mkdir gayathridir

]cd gayathridir

]git init

1.PULL

HOW TO PULL THE CODE FROM GITHUB (REMOTE REPOSITORY) TO THE SERVER

]git pull https://github.com/gopalgurukirupa/dharaka.git (select code --->)

2.CLONE

HOW TO CLONE(MIRRORING OR COPY) THE REMOTE REPO

```
]cd
```

```
]git clone https://github.com/gopalgurukirupa/dharaka.git
```

```
]ls
```

3.REBASE

HOW TO "REBASE" FROM FEATURE BRANCH

```
]mkdir newdir
```

```
]cd newdir
```

```
]git init
```

```
]vi file1
```

1

press insertkey -->esc+shift :wq!

```
]git add file1
```

```
]git commit -m "first commit"
```

```
]git log --oneline
```

```
]git branch
```

output: we get a master branch

```
]git branch devbranch ( newbranch)
```

```
]git log --oneline
```

```
]vi file1
```

2

:wq!

```
]git commit -am "second commit"
```

```
]git log --oneline
```

```
]vi file1
```

3

:wq!

]git commit -am "third commit"

]git log --oneline

output: 3 commit id in master branch

NOW LOGIN IN FEATURE BRANCH

]git checkout devbranch

]git log --oneline

output: only 1 commit id

]git rebase master

output: we get all commit id's

4.CHERRY PICK

]mkdir cherdir

]cd cherdir

]git init

]vi cherfile

1

:wq!

]git add cherfile

]git commit -m "first commit"

]git branch cherbranch

```
]vi cherfile
```

```
2
```

```
:wq!
```

```
]git commit -am "second commit"
```

```
]vi cherfile
```

```
3
```

```
:wq!
```

```
]git commit -am "third commit"
```

```
]git log --oneline
```

output: 3 commits in master

```
]git checkout cherbranch
```

```
]git log --oneline
```

```
]git cherry-pick 479fa05 (commitid )
```

```
]git rebase master
```

NOTE : REBASE & CHERRYPICK ARE RUNNING IN FEATURE BRANCH

JENKINS INSTALLATION.

STEP 1: LAUNCH EC2 INSTANCE + UBUNTU INSTALLATION

SELECT UBUNTU SERVER 16.0.4 LTS

ADD TAG : JENKINS MASTER

SECURITY GROUP: ALL TCP (OPEN ALL)

REVIEW AND LAUNCH

PUTTY LOGIN

LOGIN NAME: ubuntu

SUDO SU -

STEP 2: JAVA INSTALLATION:

A. UPDATE ALL SOFTWARE PACKAGES TO UBUNTU SERVER

sudo apt-get update (FOR REFRESHMENT)

sudo apt-get upgrade -y (FOR NEW VERSION ENABLE)

B. INSTALL JAVA ON UBUNTU SERVER

sudo apt-get install default-jdk -y

java -version

STEP 3: INSTALL JENKINS ON UBUNTU SERVER

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

**sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > **
/etc/apt/sources.list.d/jenkins.list'

sudo apt-get update

sudo apt-get install jenkins -y

STEP 4:VERIFY

ps -ef | grep jenkins

OUTPUT: PROCESS ID WITH httpport

PORT NO OF JENKINS IS 8080

DEFAULT FILE PATH : /etc/default

```
cd /etc/default
```

OUTPUT: jenkins

```
vi jenkins
```

```
cd ..
```

```
cd ..
```

JENKINS IS A DAEMON PROCESS (Background process)

Daemon path : /etc/init.d

```
cd /etc/init.d
```

```
ls
```

output: jenkins

Check under /etc/init.d/jenkins

STEP 5: LAUNCH THE JENKINS URL IN THE BROWSER

<ubuntu_ip_address>:8080

65.0.3.162:8080

OUTPUT: UNLOCK JENKINS

/var/lib/jenkins/secrets/initialAdminPassword

STEP 6: GET THE PASSWORD FROM THE PATH : /var/lib/jenkins/secrets/initialAdminPassword

```
root@] cat /var/lib/jenkins/secrets/initialAdminPassword
```

01610cf4d3e945cbb4299a679c3ae902

STEP 7: CLICK "Install Suggested Plugins" - Then the installation gets started.

STEP 8: CREATE A NEW USER OF YOUR OWN BY GIVING (Name, Email-id, username, password).

NAME: admin

PASSWORD: admin123

CONFIRM PASSWORD: admin123

FULL NAME : JENKINS

EMAIL ADDRESS : kgopaldba@gmail.com

SAVE AND CONTINUE

COPY THE "JENKINS URL" : <http://18.139.208.235:8080/>

<http://65.0.3.162:8080/>

NOW CHECK IN

`]cat /var/lib/jenkins/secrets/initialAdminPassword`

OUTPUT: NO ADMIN PASSWORD

NOTE: WHEN WE SET A NEW CREDENTIALS THE INITIAL ADMIN PASSWORD IS AUTOMATICALLY ERASED..

SAVE AND FINISH

STEP 9: THEN LOGIN INTO THE CONSOLE & EXPLORE JENKINS.

JOBS: HOW TO CREATE A NEW SIMPLE JOBS:

JENKINS_URL/job/MY%20NEW%20PARAMETER%20JOB/build?token=TOKEN_NAME

<http://3.25.120.2:8080/job/MY%20NEW%20PARAMETER%20JOB/build?token=abcd>

BASIC JOBS

1.HOW TO CREATE A SIMPLE JOB.

GO TO JENKINS ADMIN CONSOLE

Task: Create 3 jobs

New item---->Enter a item name (Build project)-->select a Free style project-->ok

Select Build-->Add Build step-->Select Execute shell--->write your own script (pwd)-->save

Back to Dashboard

New item---->Enter a item name (Test project)-->select a Free style project-->ok

Select Build-->Add Build step-->Select Execute shell--->write your own script (uname -a)-->save

Back to Dashboard

New item---->Enter a item name (Deploy project)-->select a Free style project-->ok

Select Build-->Add Build step-->Select Execute shell--->write your own script (hostname)-->save

Back to Dashboard

NOTE:DEFAULT JOB LANDING PATH??? INTERVIEW QUESTION.

/var/lib/jenkins

cd /var/lib/jenkins

ls

Output: Till now no workspace created.Because we cannot run any jobs.

RUN THE JOB AND CHECK

ls

We get a Workspace directory.

2.SEMI AUTOMATION (JOB ORDERING)

Task: when we run a Build project.... Test project and Deploy project will be auto run.By JOB ORDERING.

Select Build project-->select drop down cursor-->Configure--->Post Build Actions-->Add PostBuild Actions-->

Build other project-->Project to build : TestProject --->Save

Select Test project-->select drop down cursor-->Configure--->Post Build Actions-->Add PostBuild Actions-->

Build other project-->Project to build : DeployProject --->Save

Back to Dashboard: Now Run the BuildProject....Test and Deploy will Autorun.

Its a 50 % Manual and 50 % Automation

3.FULLY AUTOMATION (PERIODICALLY TRIGGERING)

Select the First job(Buildproject)-->Configure-->BuildTriggers-->BuildPeriodicslly---> ? -->

Schedule * * * * * (For every 1 minute)

4.REMOTE RUN (REMOTELY TRIGGERING)

Select First job-->Configure-->BuildTriggers-->TriggerBuildsRemotely--->Authentication Token-->1234

COPY THE URL :

<http://3.110.169.190:8080/job/Newbuildproject/build?token=110011>

JENKINS_URL/job/Newbuildproject/build?token=TOKEN_NAME

REPLACE THE JENKINS URL: <https://18.141.205.81:8080/job/mymonitorproject/build?token=555>

HIT IT IN THE BROWSER...JOB RUNS AUTOMATICALLY.

5.PARAMETERIZED JOB:

New item-->Enter a item name: parametrised job-->Free style project-->ok

Build-->Execute shell-->command

echo "This is my parametrised job run by \$name"

General-->This project is parametrised-->Add parameter--->String parameter-->Name(Name)-->Save

Run--->

Name: >>>> Build

Back to Dashboard

Select the parametrized job-->Console output

"This is the parametrized job run by " "

SCRIPTED PIPELINE

Newitem-->Enter an item name: scripted pipeline-->select (pipeline)-->ok

Pipeline-->Pipeline script

copy the script

node

```
{
  stage('Commit')
  {
    echo "This is Code Download from GIT Project Repository....."
  }
  stage('Build')
  {
    echo "This is Build project using maven....."
  }
  stage('Test')
  {
    echo "This is Test project Implementation using Selenium....."
  }
  stage('Release')
```

```

{
    echo "This is Delivery using Dcoker....."
}
stage('Monitor')
{
    echo "This is Application Logs Monitorinig using tool Splunk....."
}
}

```

Before Running the script,,we add the appropriate plugins such as "git-github","maven","selenium","docker","splunk"

Now we any one of the plugins.

Manage jenkins-->Manage plugins-->Available-->docker-->Install without restart

Now we run the script....we get a view.

A.DELIVERY PIPELINE VIEW:

ONE TYPE OF VIEW,,,FOR CRYSTAL VIEW REPORT,,EX: FOR MANAGERS

WE MUST INSTALL A PLUGIN FOR DELIVERY PIPELINE VIEW.

Dashboard-->Manage jenkins-->Manage plugins-->Available-->Delivery pipeline-->Install without Restart

To check,,press +

Select Delivery Pipeline view--->viewname (Manager view)--->ok-->Pipelines-->Component name (Select Build project)

->ok

For Editing: Edit view---> select 1.enable manual triggers,2.enable rebuild, 3.enable start of newpipeline build

Google search: plugins.jenkins.io

B.BUILD PIPELINE VIEW:

ONE TYPE OF VIEW,,,FOR ASSOCIATES

WE MUST INSTALL A PLUGIN FOR BUILD PIPELINE

Dashboard-->Manage jenkins-->Manage plugins-->Available-->BuildPipeline-->Install without Restart

+

Select BuildPipeline view-->viewname (Associate view)-->select initial job (Build project)

C.BUILD MONITOR VIEW

Specific Job Monitoring view for Troubleshooting purpose.To focus a Particular job.

For ex: we change the shell script (`uname -b`) in the "TEST PROJECT"

Select Test project-->configure-->Build-->execute shell--->(uname -b)-->save

Go to Dashboard-->Manage jenkins-->Manage plugins-->Available-->Build Monitorview-->Install without Restart

+

Select Build Monitorview-->View name (Particular job monitoring)-->Jobs (select Test project)-->ok

Go and Run..we get a new console view.

MASTER & SLAVE CONCEPT:

REAL TIME: MASTER CONNECT THE SLAVES & RUN THE JOBS.

IN MASTER:

1.EC2 UBUNTU SERVER-MASTER

2.JAVA INSTALLATION

3.JENKINS INSTALLATION

4.ADMIN CONSOLE

5.JOB CREATION

JOB LANDING PATH: /var/lib/jenkins/workspace

IN SLAVES.

NO NEED TO INSTALL JENKINS..INSTALL ONLY JAVA

STEP 1:EC2 UBUNTU SERVER-SLAVE

ADD TAG: HDFC SLAVE

SECURITY GROUP:OPEN ALL PORTS

LAUNCH & LOGIN

sudo su -

STEP 2:INSTALL JAVA

A. UPDATE ALL SOFTWARE PACKAGES TO UBUNTU SERVER

sudo apt-get update (FOR REFRESHMENT)

sudo apt-get upgrade -y (FOR NEW VERSION ENABLE)

B. INSTALL JAVA ON UBUNTU SERVER

sudo apt-get install default-jdk -y

java -version

STEP 3:GO TO MASTER & GENERATE KEY IN DEFAULT USER "UBUNTU"

ubuntu]pwd

cd /home/ubuntu

OUTPUT: /home/ubuntu

HOW TO CHECK A HOSTNAME

ubuntu]hostname

OUTPUT:

ubuntu]ls -lart (to view the hidden files)

OUTPUT: .ssh

ubuntu]cd .ssh (extra privilege for ubuntu user for Key generation,connection between Master & slave)

.ssh\$]ls -lart (Master can take over slaves activity)

OUTPUT: WE GET ONLY ONE KEY CALLED "AUTHORIZED KEY"..IS A PUBLIC KEY

.ssh\$]ssh (hostname) (Check the ubuntu user can connect same host or not ?)

OUTPUT: PERMISSION DENIED,ubuntu not able to connect "ownself"

.ssh\$]ls -lart

OUTPUT: WE GET AN ADDITIONAL FILE "known_hosts"

.ssh\$]ls -lrt

.ssh\$]touch id_rsa (Standard format for private key)

.ssh\$]touch id_rsa.pub (Standard format for public key)

.ssh\$]ls -lrt

Now we copy the content from the authorized key & paste it in a "public key file"

.ssh\$]cp authorized_keys id_rsa.pub

.ssh\$]ls -lrt

OUTPUT: CHECK THE SIZE

.ssh\$]vi id_rsa

Now we copy the "pem file" and paste it in "id_rsa" file

:wq!

.ssh\$]ls -lrt

OUTPUT: WE GET 4 FILES,,BUT THE PERMISSIONS ARE DIFFERENT

.ssh\$]chmod 400 *

.ssh\$]ls -lrt

.ssh\$]ssh localhost

OUTPUT: NOW ITS CONNECT

STEP 4: AGENTING:

JENKINS DEFAULT JOB LANDING PATH: /var/lib/jenkins/workspace

GO TO SLAVE MACHINE

```
]cd /var/lib
```

```
]ls
```

OUTPUT: NO JENKINS AND NO WORKSPACE

NOW WE CREATE A CUSTOMIZED JOB LANDING PATH IN "SLAVE MACHINE"

```
]exit
```

```
]pwd
```

OUTPUT : /home/ubuntu

```
ubuntu]mkdir jenkins
```

```
ubuntu]cd jenkins
```

```
]pwd
```

OUTPUT : /home/ubuntu/jenkins

Now its a customized job landing path.

```
]ls -lrt
```

OUTPUT:

TILL NOW THERE IS NO CONNECTION BETWEEN MASTER & SLAVE.

NOW WE INTEGRATE MASTER & SLAVE WITH THE HELP OF "AGENTING"

GO TO JENKINS MASTER CONSOLE:

MANAGE JENKINS-->MANAGE NODES & CLOUDS-->NEW NODE-->NODE NAME (AGENT 1)

SELECT .PERMANENT AGENT--->OK

NUMBER OF EXECUTORS : 2 (FOR SPEED)CPU PROCESSOR

REMOTE ROOT DIRECTORY : /home/ubuntu/jenkins [customized job landing path]

LABELS : hdfc

LAUNCH METHOD : LAUNCH AGENTS VIA SSH

HOST : "SLAVES PUBLIC IP"

CREDENTIALS : ADD (JENKINS)

ADD CREDENTIALS

DOMAIN

KIND: SSH USERNAME WITH PRIVATE KEY

ID : ubuntu

USER : ubuntu

PRIVATE KEY

.ENTER DIRECTLY ADD

PASTE THE PEM KEY CONTENT

-->ADD

CREDENTILAS (SELECT UBUNTU)

HOST KEY VERIFICATION STRATEGY : "manually trusted key verification strategy"

SELECT-->SAVE

"AGENT 1" IS NOT IN SYNCHED STATE. CLICK :REFRESH STATUS

OPEN AGENT1 & RELAUNCH AGENT

"AGENT 1" IS SYNCHED NOW

STEP 5: CREATE A NEW JOB

NEW ITEM--->ENTER AN ITEM NAME (hdfc_slave_jobs)-->SELECT FREE STYLE PROJECT

BUILD->ADD BUILD STEP-->EXECUTE SHELL (pwd)

GENERAL

SELECT .RESTRICT WHERE THIS PROJECT CAN BE RUN

LABEL EXPRESSION : SELECT hdfc

CLICK APPLY & SAVE

STEP 6:GO TO SLAVE CONSOLE

]ls

OUTPUT : remoting.jar (It means master and slave are integrated)

But no workspace

STEP 7: GO TO JENKINS CONSOLE

RUN THE JOB

STEP 8: GO TO JENKINS CONSOLE

]ls -lrt

OUTPUT: workspace

]cd workspace

]ls

STEP 9: GO TO JENKINS CONSOLE

SELECT THE JOB--->CONSOLE OUTPUT

PROMETHEUS & GRAFANA

PROMETHEUS:

1. MONITORING TOOL

2. SERVER INFRASTRUCTURE MONITORING

3. IT MONITORS MORE THAN 800 + METRICS

4. IT MONITORS EVERY 10 TO 15 SECONDS (TIME INTERVAL OR SCRAP TIME)

5. IT'S A GENERIC MONITORING TOOL.. (USE IN ALL PROVIDERS ie AWS,GCP,AZURE,ORACLE)

6. IT'S A PROACTIVE MONITORING TOOL..

AWS- MONITORING TOOL

CLOUD WATCH

1. ONLY 21 METRICS

2. TIME INTERVAL IS MORE THAN 1 MIN

EX: 100 SERVERS,,,

1 SERVER (MASTER) ,, 99 SERVERS ARE SLAVES

MASTER & SLAVE CONCEPT...

MASTER -->COMMANDER

SLAVE -----> EXECUTOR

PRACTICAL:

LAUNCH 2 LINUX SERVER

1 MASTER

1 SLAVE

LOGIN:MASTER

1. ec2-user

2.sudo su -

3.yum update -y

GO TO GOOGLE -->PROMETHEUD DOWNLOAD

<https://prometheus.io/download/>

`https://github.com/prometheus/prometheus/releases/download/v2.39.1/prometheus-2.39.1.linux-amd64.tar.gz`

`]wget https://github.com/prometheus/prometheus/releases/download/v2.39.1/prometheus-2.39.1.linux-amd64.tar.gz`

`]ls -lrt`

`prometheus-2.39.1.linux-amd64.tar.gz`

`]tar -xzf prometheus-2.39.1.linux-amd64.tar.gz`

`]ls -lrt`

`prometheus-2.39.1.linux-amd64`

`]cd prometheus-2.39.1.linux-amd64`

`ls -lrt`

`prometheus`

`promtool`

`]cp prometheus /usr/local/bin`

`]cp promtool /usr/local/bin`

`]./prometheus --config.file=prometheus.yml &`

`portno is 9090`

`publicip:9090`

`34.239.121.249:9090`

WE GET A NEW PROMETHEUS CONSOLE:

STATUS -->TARGET

<http://34.239.121.249:9090/metrics>

COPY THE METRICS IN EXCEL SHEET

2. LOGIN SLAVE MACHINE

1.ec2-user

2.sudo su -

3.yum update -y

4. <https://prometheus.io/download/>

https://github.com/prometheus/node_exporter/releases/download/v1.4.0/node_exporter-1.4.0.linux-amd64.tar.gz

`]wget https://github.com/prometheus/node_exporter/releases/download/v1.4.0/node_exporter-1.4.0.linux-amd64.tar.gz`

`]ls -lrt`

`node_exporter-1.4.0.linux-amd64.tar.gz`

`]tar -xzf node_exporter-1.4.0.linux-amd64.tar.gz`

```
]ls -lrt
```

```
node_exporter-1.4.0.linux-amd64
```

```
]cd node_exporter-1.4.0.linux-amd64
```

```
ls -lrt
```

```
node_exporter
```

```
]./node_exporter &
```

```
port no of slave is 9100
```

3. TILL NOW THERE IS NO COMMUNICATION BETWEEN MASTER & SLAVE

NOW WE INTEGRATE MASTER & SLAVE

GO TO MASTER MACHINE

EDIT THE "prometheus.yml"

```
slaveipaddress: 54.227.149.226
```

```
vi prometheus.yml
```

```
- job_name: "node"
```

metrics_path defaults to '/metrics'

scheme defaults to 'http'.

static_configs:

- targets: ["13.211.188.120:9100"]

:wq!

ps -ef | grep prometheus

kill -9 6337

./prometheus --config.file=prometheus.yml &

GO TO BROWSER & CHECK

WE GET

1.NODE (SLAVE) - UP

2. PROMETHEUS - UP

GRAFANA:

REPORTING TOOL.

WE INSTALL GRAFANA IN MASTER

<https://grafana.com/grafana/download?edition=oss>

]cd ..

]wget <https://dl.grafana.com/oss/release/grafana-9.2.3.linux-amd64.tar.gz>

]ls -lrt

grafana-9.2.3.linux-amd64.tar.gz

]tar -xvzf grafana-9.2.3.linux-amd64.tar.gz

]cd grafana-9.2.3

]ls -lrt

]cd bin

]./grafana-server &

port no: 3000

3.25.117.68:3000

username: admin

password : admin

**CONFIGURATIONS --->DATASOURCE-->ADD DATA SOURCE-->SELECT PROMETRICS-->URL :
MASTERPUBLICIP:9090 -->SAVE & TEST (DATA SOURCE UPDATE SUCCEEDED)**

+DASHBOARD -->NEW DASH BOARD -->ADD AN PANEL -->METRICS (SELECT ANY CPU)-->APPLY

FOR IMPORT

+ -->import -->discard

GO TO GRAFANA.COM

DASHBOARD --><https://grafana.com/grafana/dashboards/3662-prometheus-2-0-overview/>

Copy the CLIP-Borad

GO TO GRAFANA

IMPORT VIA GRAFANA.COM

COPY & PASTE THE ID -->LOAD -->IMPORT

WE GET A GRAPH

SALT STACK.

-->CONFIGURATION MANAGEMENT TOOL (CM TOOL)

-->OTHER CM TOOL

1. ANSIBLE , 2. CHEF 3. PUPPET

-->MASTER & SLAVE CONCEPT

-> OPEN SOURCE TOOL

IN 1000 SERVER, THEY GIVE 10 TASK

1. GIT INSTALL

2. APACHE

3.JENKIN

4. JAVA

5.PYTHON

6.DIR CREATE

7. FILE CREATE

8. PROCESS KILL

9.PACKAGE

10.SERVICE RESTART

MASTER : 1 ,, SLAVE : 999

ADVANTAGE:

1. TIME SAVE

2. ERROR AVOID

3. CUT THE RESOURCE

4. REDUCE THE COST

SALT STACK IS A AGENT BASED TOOL. NO PAGENT CONFIGURATION.

AGENT BASED -- EX: WHAT'S UP STATUS

IN MASTER MACHINE CONTAINS SLAVE MACHINE'S IP ADDRESS

IN SLAVE MACHINE CONTAINS MASTER IP ADDRESS

MECHANISAM:

SALT STACK : PUSH & PULL

ANSIBLE : PUSH

CHEF : PULL

PUPPET : PULL

MASTER : CONTROL NODE

SLAVE : MINION NODE / MANAGED NODE

MASTER : ONLY LINUX INSTALLATION

PRACTICALS :

EC2 LAUNCH --- 3 INSTANCES

MASTER (1) , SLAVES (2)

ALL TCP

1. LOGIN MASTER

```
curl -L https://bootstrap.saltstack.com -o install_salt.sh
```

```
ls -lrt
```

```
sh install_salt.sh -P -M
```

```
service salt-master status
```

2. LOGIN SLAVE 1

```
curl -L https://bootstrap.saltstack.com -o install_salt.sh
```

```
ls -lrt
```

```
sh install_salt.sh -P
```

```
service salt-minion status
```

3. LOGIN SLAVE 2

```
curl -L https://bootstrap.saltstack.com -o install_salt.sh
```

```
ls -lrt
```

```
sh install_salt.sh -P
```

service salt-minion status

4. IN SLAVE MACHINE... UPDATE THE MASTER INFO (ip-172-31-86-222.ec2.internal) --- GO TO MASTER: hostname

NOW IAM IN SLAVE 1 MACHINE

]cd /etc/salt

]vi minion

master: ip-172-31-86-222.ec2.internal

:wq!

service salt-minion restart

5. GO TO SLAVE 2 MACHINE

]cd /etc/salt

]vi minion

master: ip-172-31-86-222.ec2.internal

:wq!

service salt-minion restart

6. GO TO MASTER

]salt-key

```
]salt-key -a ip-172-31-93-249.ec2.internal
```

]n/y - press y to process for each slave's

```
]salt-key
```

NOW CHECK THE CONNECTIVITY BETWEEN MASTER & SLAVE

```
]salt '*' test.ping
```

TO VIEW THE FUNCTIONS

```
]salt '*' sys.list_functions
```

TO VIEW THE PACKAGE INFO

```
]salt '*' sys.list_functions pkg
```

TO INSTALL ANY PACKAGE

NOW IAM GOING TO INSTALL GIT IN SLAVE

```
]salt '*' pkg.install git
```

NOW IAM GOING TO INSTALL MYSQL IN SLAVE

```
]salt '*' pkg.install mysql
```

NOW IAM GOING TO INSTALL PHP IN SLAVE

```
]salt '*' pkg.install php
```

SPLUNK

->MONITORING TOOL

->ITS A ENTERPRISE TOOL.. (NOT AN OPEN SOURCE TOOL)

->FOR LEARNING FREE TRIAL FOR 60 DAYS

-> IT MONITOR'S THE LOGS.....SO SPLUNK IS A "LOG MONITORING TOOL"

->RCA (ROOT CAUSE ANALYSIS)

-> MASTER & SLAVE CONCEPT

--> TYPES OF LOGS:

SYSTEM LOGS , ERROR LOGS , APP LOGS , EVENT LOGS , QUERY LOGS ,, GENERAL LOGS

--> PUSHING MECHANISAM..

SLAVE PUSH THE LOGS TO THE MASTER.

SLAVE HAS FORWARDERS TO DO THE ACTION.

PRACTICALS:

1.<https://www.splunk.com>

signup: username & password

https://www.splunk.com/en_us/download.html

2. LAUNCH 2 EC2 INSTANCE

1. LOGIN MASTER MACHINE

1.ec2-user

2.sudo su -

3. cd /opt

opt] yum update

opt]wget -O splunk-9.0.2-17e00c557dc1-Linux-x86_64.tgz
"https://download.splunk.com/products/splunk/releases/9.0.2/linux/splunk-9.0.2-17e00c557dc1-Linux-x86_64.tgz"

opt]ls

output: splunk-9.0.2-17e00c557dc1-Linux-x86_64.tgz

opt]tar -xzf splunk-9.0.2-17e00c557dc1-Linux-x86_64.tgz

opt]ls

output: splunk

]cd splunk

]ls

]cd bin

ls

]./splunk start --accept-license

username:admin

password: Admin123

portno:8000

publicip:8000

output: SPLUNK ENTERPRISE

SEARCH & REPORTING

2. LOGIN SLAVE MACHINE

1.ec2-user

2.sudo su -

3. cd /opt

opt] yum update -y

opt]wget -O splunkforwarder-9.0.2-17e00c557dc1-Linux-x86_64.tgz
"https://download.splunk.com/products/universalforwarder/releases/9.0.2/linux/splunkforwarder-9.0.2-17e00c557dc1-Linux-x86_64.tgz"

]ls -lrt

output :splunkforwarder-9.0.2-17e00c557dc1-Linux-x86_64.tgz

]tar -xzf splunkforwarder-9.0.2-17e00c557dc1-Linux-x86_64.tgz

]cd splunkforwarder

]ls

cd bin

]/splunk start --accept-license

username: admin

password: Admin123

port no: 8089

]/splunk add forward-server 3.85.131.44:9997 (masterip address)

OPEN WINSCP

COPY SLAVE IP ADDRESS

COPY THE SYSLOG FILE

bin]cd /home/ec2-user

ec2-user]ls

output: syslog

ec2-user]cd -

bin]cd /var/log

log]cp /home/ec2-user/syslog .

log]ls

output: syslog

log]cd -

bin]/splunk add monitor /var/log/syslog -index main -sourcetype slave1logs

output: ADDED MONITOR OF /var/log/syslog

GO TO MASTER MACHINE

bin] ./splunk enable listen 9997

username: admin

password: Admin123

output : Listening for Splunk data on TCP port 9997.

GO TO BROWSER,,MASTER SPLUNK ENTERPRISE

search

index="main"

output: error (space issue)

bin]cd /opt/splunk/etc/system/default

ls

output: server.conf

vi server.conf

copy

disk usage processor settings

[diskUsage]

minFreeSpace = 5000

pollingFrequency = 100000

pollingTimerFrequency = 10

default]cd ..

system]ls

output:local

]cd local

ls

output: server.conf

vi server.conf

disk usage processor settings

[diskUsage]

minFreeSpace = 50

pollingFrequency = 100000

pollingTimerFrequency = 10

:wq!

local]cd /opt/splunk/bin

bin]../splunk restart

GO TO GRAFICAL WEBSITE AND CHECK

WE GET A LOGS

TERRAFORM

1. ITS A GENERIC DEVOPS TOOL

AWS

IAAS (INFRASTRUCTURE AS A SERVICE) -- SERVER (EC2) ,STORAGE , NETWORK (VPC)

FOR VPC CREATION IN AWS (20 MINS)

1.VPC

2.2 SUBNETS

3.INTERNET GATEWAY

4.ROUTE TABLE

4.5 SUBNET ASSOCIATION

5.ROUTES

6.SECURITY GROUP

7.WEBSERVER

8.APP SERVER

9.NAT

10.ROUTES

TERRAFORM

1. INFRA BUILD TOOL

2. ENV BUILD TOOL

3. IAAC (INFRA AS A CODE)

4. WE INSTALL TERRAFORM IN ALL CLOUD (AWS ,AZURE ,ORACLE ,GCP---)

5. WE BUILD AN INFRA IN 2 MINS

6.TERRAFORM IS THE TOP TOOL IN ENV BUILD

7. DEVELOPED BY HASHI CORP COMPANY

8. HCL CODE IS USED (HASHICORP CONFIGURATION LANGUAGE)

9. TERRAFORM WORKS ONLY IN CLOUD ENV

PRACTICALS :

1. EC2 SERVER LAUNCH & LOGIN

LOGIN AS :ec2-user

sudo su -

2. GO TO TERRAFORM WEBSITE

<https://www.terraform.io/>

opensource -->download -->linux -->Releases -->select 1.0.11 -->linux Amd64-->right click -->copy link address

https://releases.hashicorp.com/terraform/1.0.11/terraform_1.0.11_linux_amd64.zip

3.GO TO SERVER

```
]yum update
```

```
]wget https://releases.hashicorp.com/terraform/1.0.11/terraform_1.0.11_linux_amd64.zip
```

```
]ls -lrt
```

output : terraform_1.0.11_linux_amd64.zip

```
]unzip terraform_1.0.11_linux_amd64.zip
```

```
]ls -lrt
```

output : terraform

```
]mv terraform /usr/local/bin
```

CHECK WHETHER THE TERRAFORM SERVER IS ACCESS THE OTHER SERVICE

```
]aws s3 ls
```

output : Unable to locate credentials. You can configure credentials by running "aws configure".

NOW WE CREATE 1 IAM ROLE WITH ADMIN ACCESS

GO TO IAM CONSOLE

IAM -->ROLES -->SELECT EC2--->ADMIN FULL ACCESS POLICY -->ROLE NAME --->CREATE ROLE

GO TO EC2 AND ATTACH THE ROLE

ACTION -->SECURITY --->MODIFY IAM ROLE --->SELECT OUR IAM ROLE

DOWNLOAD "VISUAL STUDIOCODE " (google --->download visual studio code)

select the EXTENSION -->HASHI CORP HCL --->INSTALL

MY PLAN IS TO CREATE VPC USING TERRAFORM CODE (HCL)

GO TO TERRAFORM WEBSITE:

terraform.io -->registry-->browse providers--->aws -->version 3.50.0 -->documentation

<https://registry.terraform.io/providers/hashicorp/aws/3.50.0/docs>

```
terraform {  
  required_providers {  
    aws = {  
      source = "hashicorp/aws"  
      version = "~> 3.0"  
    }  
  }  
}
```

Configure the AWS Provider

```
provider "aws" {  
  region = "us-east-1"  ----->NORTH VIRGINIA
```



```
}
```

VPC CREATIONS STEPS :

SELECT VPC

1. aws_vpc
- 2.aws_subnet
- 3.aws_internet_gateway
- 4.aws_route_table
- 5.aws_route_table-association
- 6.aws_nat_gateway
- 7.ec2 --> aws_eip
- 8.ec2_security_group

```
vi main.tf
```

```
terraform {  
  required_providers {  
    aws = {  
      source = "hashicorp/aws"  
      version = "~> 3.0"  
    }  
  }  
}
```

```
# Configure the AWS Provider
```

```
provider "aws" {  
  region = "us-east-1"
```

```
}
```

```
resource "aws_vpc" "dharakavpc" {
```

```
  cidr_block    = "10.0.0.0/16"
```

```
  instance_tenancy = "default"
```

```
  tags = {
```

```
    Name = "CTS-VPC"
```

```
  }
```

```
}
```

```
resource "aws_subnet" "pubsub" {
```

```
  vpc_id = aws_vpc.dharakavpc.id
```

```
  cidr_block = "10.0.1.0/24"
```

```
  availability_zone="us-east-1a"
```

```
  tags = {
```

```
    Name = "PUBLIC SUBNET"
```

```
  }
```

```
}
```

```
resource "aws_subnet" "prisub" {
```

```
  vpc_id = aws_vpc.dharakavpc.id
```

```
  cidr_block = "10.0.2.0/24"
```

```
  availability_zone="us-east-1a"
```

```
  tags = {
```

```
    Name = "PRIVATE SUBNET"
```

```
  }
```

```
}
```

```
resource "aws_internet_gateway" "tigw" {
```

```
  vpc_id = aws_vpc.dharakavpc.id
```

```
  tags = {
```

```
    Name = "INTERNET GATEWAY"
```

```
  }
```

```
}
```

```
resource "aws_route_table" "pubrt" {
```

```
  vpc_id = aws_vpc.dharakavpc.id
```

```
  route {
```

```
    cidr_block = "0.0.0.0/0"
```

```
    gateway_id = aws_internet_gateway.tigw.id
```

```
  }
```

```
  tags = {
```

```
    Name = "PUBLIC ROUTE TABLE"
```

```
  }
```

```
}
```

```
resource "aws_route_table_association" "pubsubassociation" {
```

```
  subnet_id    = aws_subnet.pubsub.id
```

```
  route_table_id = aws_route_table.pubrt.id
```

```
}
```

```
resource "aws_eip" "teip" {
```

```
    vpc    = true
}
```

```
resource "aws_nat_gateway" "tnat" {
  allocation_id = aws_eip.teip.id
  subnet_id    = aws_subnet.pubsub.id
```

```
  tags = {
    Name = "NAT-GATEWAY"
  }
}
```

```
resource "aws_route_table" "prirt" {
  vpc_id = aws_vpc.dharakavpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_nat_gateway.tnat.id
  }
}
```

```
  tags = {
    Name = "PRIVATE ROUTE TABLE"
  }
}
```

```
resource "aws_route_table_association" "prisubassociation" {
  subnet_id    = aws_subnet.prisub.id
  route_table_id = aws_route_table.prirt.id
```

```
}
```

```
resource "aws_security_group" "pubsg" {  
  name      = "pubsg"  
  description = "Allow TLS inbound traffic"  
  vpc_id    = aws_vpc.dharakavpc.id
```

```
  ingress {  
    description = "TLS from VPC"  
    from_port   = 0  
    to_port     = 65535  
    protocol    = "tcp"  
    cidr_blocks = ["0.0.0.0/0"]  
  }  
}
```

```
  egress {  
    from_port   = 0  
    to_port     = 0  
    protocol    = "-1"  
    cidr_blocks = ["0.0.0.0/0"]  
  }  
}
```

```
tags = {  
  Name = "PUBLIC SECURITY GROUP"  
}  
}
```

```
resource "aws_security_group" "prisg" {  
  name      = "prisg"
```

description = "Allow TLS inbound traffic from Publis Subnet"

vpc_id = aws_vpc.dharakavpc.id

ingress {

description = "TLS from VPC"

from_port = 0

to_port = 65535

protocol = "tcp"

cidr_blocks = ["10.0.1.0/24"]

}

egress {

from_port = 0

to_port = 0

protocol = "-1"

cidr_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "PRIVATE SECURITY GROUP"

}

}

resource "aws_instance" "pub_instance" {

ami = "ami-09d3b3274b6c5d4aa"

instance_type = "t2.micro"

availability_zone = "us-east-1a"

associate_public_ip_address = "true"

vpc_security_group_ids = [aws_security_group.pubsg.id]

```
subnet_id          = aws_subnet.pubsub.id
key_name            = "Dharaka"
```

```
tags = {
  Name = "DHARAKA WEBSERVER"
}
}
```

```
resource "aws_instance" "pri_instance" {
  ami                = "ami-09d3b3274b6c5d4aa"
  instance_type      = "t2.micro"
  availability_zone   = "us-east-1a"
  associate_public_ip_address = "false"
  vpc_security_group_ids = [aws_security_group.prisg.id]
  subnet_id          = aws_subnet.prisub.id
  key_name            = "Dharaka"
```

```
tags = {
  Name = "DHARAKA APPSERVER"
}
}
```

```
]terraform init
]terraform plan
]terraform apply
```

You should change :

-> Current 2 ami id's

-> Current 2 keypair's

-> Any 2 tag's Name's and

-> Change your 8 VPC Name's