Project - 2 11/04/22

Micro service using & kubernetes and Docker

Infra structure :-

Jenkins

Sonar Qube

Kubernetes cluster

Automations :-

Jenkins pipeline

Docker file

Kubernetes manifest file

Project - 1

Checkout - Git

Unit Test cases - maven

code Quality - sonar

Artifact storage - Nexus

Deploy latest artifact - Ansible

publish reports - Junit

E-mail - E-mail

Git - repo

Docker file

From : tomcat : 8.0

copy : ./gameoflife-web/target
/gameoflife.war /usr/local
/tomcat/webapps

Jenkins file

65. /vamshi1227/

78. & master ip address

79.

80.

deploy - kube.yml

14. /vamshi1227/

step-1:

- added Jenkins user sudoers file
- added Jenkins to docker group
- changed permission of /var/run/docker.sock to 777
- login to docker hub

step-2:

Establish SSH communication b/w Jenkins & master

step-3:-

change IP of master node in Jenkins pipline in github-repo

step-4 :-

Copy K8S config file to the Jenkins user & provide permissions (PDF 45)

Sonar-project properties

'GOL-Micro-Deployment' = Pipeline name

Piplne

scm : GIT

repo : Past (code path)

scripth path : Jenkinsfile-docker-deploy

Note,
on Jenkins machine install docker and login

```
Sudo -i
su Jenkins
cd
pwd
Sudo Yum install docker -y
exit
Yum install docker -y
service docker start
```

⇒ If not running change it to (t2 medium).

```
su jenkins
cd
docker images
```

* Sudo usermod -aG docker $USER.

⇒ add user in sudoer file

exit

* visudo

⇒ add Jenkins (All) =(All) NO=PASSWORD=All.

* su jenkins.

* cd

* sudo usermod -aG docker $USER.

* docker images

* sudo docker images

} adding jenkins to docker

Now, provide 'sock' access.

cd /var/run

ls -l

sudo chmod 777 /var/run/docker.sock

sudo chown $USER /var/run/docker.sock

integration b/w docker and Jenkins

docker login

username:

Password:

NOW, SSH mechanism b/w master & Jenkins

cd .ssh/

ls

cat id-rsa.pub

in master, (Ubuntu)

adduser Jenkins

Pass: { Jenkins
Jenkins

Enter

Y

visudo

copy and past

Jenkins ALL=(ALL):ALL) NOPASSWORD=ALL

⇒ TO save in Nano editor

ctrl + X + Y

su Jenkins

cd

mkdir .ssh

cd .ssh

vi authorized_keys

Chmod 400 authorized_key

cd ..

Chmod 700 .ssh/

ls -la

on Jenkins

ssh @Jenkins@ Ip address privat

exit

Now,
Change Ip address in Jenkinspile -docke deploy

Now,
on master give Jenkins permission for admin file

mkdir -p $HOME/.KUBE

Sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

Sudo chown $(id-u):$(id-g) $HOME/.kube/config

Amazon cloud web service :- 12/04/22

* Physical
* Vertical
* Instance

Aws, GCP, Azur, IBM, Bluemix — cloud.

1. Choose AMI :-

* My AMI's :- Image.
  * we will get Entire set-up of that particular image. ex:- Tomcat

* Aws Marterplace :- Vendors.
  * Here we have to pay rent per hour.

* Community AMI's :- Instant store instance
  Here when we loss the instance data is also lost.

  EBS :-
  Elastic block storage

2. Choose an instance type :- There're different types of families present. (Types of CPU's)
   ex:- T series, E, D etc.,

* General Instance :-

* Compute instance :-

* Memory instance
* Storage instance
* G.P.U instance

→ Burstable performance instance :- (T_2 instance)

When the user is doubled then if not loose the users here not to impact the user increase to instance from (t_2 medium - t_2 larg), here (BPI) comes into the picture, here credits are add for the hours instance not running and this credits are used for increasing instance storage and this is applied for only that day.

3. Configure instance :-

Number of instance :-

purchasing option :-

EC2 on Demand :- pay for what you use
Has the highest cost but no upfront payment.

EC2 Reserve instance :- 75% discount
Reservation period 1 or 3.
if paid upfront more discount.

EC2 spot instance :- 90% discount, but if it taken for 1 hours after that someone bids for more that instance will go to that person.

EC2 Dedicated Hosts :- Same pricing as on Demand instance

Network :- default

Subnet :- region availability.

Hostname type :-

placement Group :- for clusters, spread, partition

Cluster	Spread	Partition
* Same availability Zone * Very fast * low latency.	* different availability Zone. * limited to 7 instance * Not sufficient for huge work	* up to 7 partitions * up 100 instance * for more effective * KAFKA, HBFC

Advanced Details :-

User data :- bootable scripts

4. Add storage :- We can change the size up to 50GB and we can had Volumes.

5. Add tags :- providing the 'Name'
ex:- nlbserver
Tomcat
SonarQube etc.,

6. Security Group:- We will provide the security ports for the access to the apps.

Ex:- 10.0.0.0/28 SSH

HTTP

(or)

(up front) Create the security Group and then configure with that security Group to the instances.

Here access is given for who can use (or) open the instance

13/04/22

Creating Templets (Reusability).

templet-name and other requirements upfront.

want to change instance type then 'stop' it and can change the type.

Similarly Termination protection if 'enable' then it can't be deleted to delete it we have to 'disable' it.

Similarly user data also stop and then change the the user data.

System logs also can be seen

Images can be created with whatever the particular image is having.

Elastic Block store :- (EBS)

Volumes :- (Backup) (8GB) is default volume for any instance. to modify this volume the action and modify the volume.

** root volumes can't be used straight forward we have run some command (growpart).

we can also create the volumes upfront on the Zone level and create volume.

Backup's (Snapshots) :- Backup of volume (data)

create snapshot :- Give name.

Lifecycle Manager :- for automating the snapshots.

Policy Schedule :-

name :-

frequency :- Daily/weekly.

Every :- Hours

Starting :-

Retention type :-

Retain :-

This lifecycle policy for important tool like Jenkins, sonar etc.,

We want to access the instance in another region

ex :- If in ohio then I want to use in mumbai.

two way either :- entire state (or) data

Entire state of instance :-

create a image once this is ready take to the other region (copy ami) destination region and copy. hence image created on other region, now create server with this image.

Only data :- then create a snapshot of that particular volume then select that snapshot and copy it there they will ask the region, give the region and hence it is copied to that region and creat a volume from the snapshot

and attach this volume to the server, hence you got the data from one region to another.

Encryption :-

We can encrypt the volume while creating the server, now if encrypt data after server creation then create the snapshot (Actions, copy the snapshot and encrypt it) we will get an encrypted snapshot, now create a volume with that snapshot and now we can create a server with encrypted volume.

Elastic Ip's :- static Ip (Allocate Elastic Ip) create and then associate this Ip by choosing an instance. there we will get an static Ip.
If we want to release then deassociate the Ip and release it.

Load Balancing :- Route traffic and check health of the server and SSL Termination

Creating Load Balancer :- restricting ports on LB level.

Classic Load Balancer :-

Name :-

Security :-

Advanced details :- it will tell if the server is running (or) not.

we can access with DNS name as well.

" This is 'round Robin Algorithm".

Port configuration :-

Stickiness :- seconds.

for the given time (sec) it will route to the only 1 server.

(ALB) 18/04/22

Application load Balancer :- Smart load balancing.

Load Balancer

create Load Balancer

Name :-

V.P.C

Target = Instance.

For micro server based application.

Network load Balancer :- Similar to classic load Balancer

* works on 'TCP' ports
* For processing millions of requests per second.

** * IAM :- Indentity Access Managment.

i) User Groups
ii) Users
iii) Roles
iv) Policies

User

Add user Groups

copy ~~Apply~~ same as existing user.

Policies
↓
Set of permissions

This is to restrict the users on user level.

User group

Create a group.

group name :- Tester.

add group level permissions and creat a group

***

MFA :- Multi factor Authentication.

↓

users:-

particular user

securing credentials.

MFA device :- Manage

Continue

Download app

Show QR code.

Scan code.

we will get 2 codes.

imp { MFA code 1
MFA code 2

Need this code to login along with user & password.

Roles:-

Managing users (or) managing serviceses.

TO create a Custom role

Image id (AWS) (1) page. → there you can find.

Instead of using the "Access key and secret keys" we use 'Roles' in AWS.

8:00.

Roles

Create role.

AWS service

EC2 → Next

⇒ Give access

- Permission policies.

} Custom role created.

Role name

ec2-s3

Now, Instance → Action

Instance setting

Attach/Replace IAM role

IAM role [ec2-s3]

Policies :-

Granual level of access.

This is to create our own policy.

List

Read

write

Tag

Permissions

while creating role we attach the IAM at stage (3).

19/04/22

Simple Notification Service :- (SNS)

First create a topic.

create Topic.

FIFO ⊙ Standard.

Name

Email

create topic.

⇒ Subscription :-

create sub

Topic ARN

Protocol : Email

Endpoint : email id

Create subscription.

⇒ Now on Email "mail" will be sent and click on link.

Cloud watch :- Monitoring Tool in AWS.

↓

Service.

For ex:- EC2 Service has monitoring service for every 5 min and it's free. (data points are collected).

Detail monitoring is cost based.

Cloud watch

⇒ Dashboard

create dashboard.

Name :

Select widget :

☑ Metrics ○ logs

Add metric graph

Browse.

Search for instance and add metric

create webget.

Dash boards are not reagion Level, so we can use dash boards Global level.

Alarms :- To keep an eye on servers we use alarms.

it can also take some actions as well.

⇒ All alarm.

→ Metrics.

→ EC2

→ instance level metric.

copy past the instance id.

it's a Graphical representation

Cloud watch :- Log streaming.

in ELK it will reformati the data but in cloud watch.

it will store in log group but not formati any data.

Integration b/w EC2 to cloud watch.

service to service communication.

using roles and Policies.

create the policy. (from Pdf).

Go to IAM

policies

create policy.

Json

⇒ Past code.

review.

Name - aws-log-group-policy

create policy.

Now, create the role using the policy.

IAM

Roles

create role.

EC2 (Next)

select EC2 policy we created.

Next

Role name:- aws-log-group-role

create role.

Now, Go to instance select server and actions

instance setting then Attach/replace the role

IAM role : aws-log-group-role.

apply.

close

Now, to push the logs from EC2-cloudwatch we need the agents for that install the agent in server.

cd Downloads/.

Yum install awslogs (awslogs is agent).

cd /etc/awslogs/.

ls.

vi awscli.conf

⇒ Change to region.

vi awslog.conf

⇒ remove all the entries.
and paste from (pdf).

file to be updated by us:

region =

file = /var/log/ (service name.log) → real time.

log_group_name = (service name.log) → real time.

Now, start the agent.

service awslogsd status.

service awslogsd start

service awslogsd status.

cd /var/log.

ls

touch application.log.

ls

echo "This is test data" >> application.log.
" " " " " " " "
" " " " " " " " "
} creating log.

Now, to check go to AWS

log groups and check.

Now, to get the alarms.

applications_logs.

⇒ metric pileter.

⇒ filler patter (which logs you want).

ex:- Error.

Next

metric details:-

Key word :- ERROR.

and give details.

and create the metric details and set alarm

Cloud watch = creating dash boards, alarams, log stream

- mng, scheduling triggering the events (Lamda).

with prior to monitoring

Lamda :- execution of scripts and automation

without servers is Lamda (Lambda).

* Server less Executions = Lambda

Lambda.

Functions.

Create Functions.

Scratch.

Function :- EC2-start.

Runtime :- Python 3.9.

Permissions :-

To authenticate create the policy. format.

IAM

Policy.

Jason

⇒ Paste the code (pdf).

Name :- Lamda policy.

[Create policy]

Role

Create role

Aws service.

Lambda.

☑ Lambda policy

[Next]

Role name :- Lambda role.

NOW, on Lambda

✓ We an existing role

→ Existing role

Lambda-role

| create function |

EC2-start

code Test monitor Conf

↓

Take code from pdf

place the instance id's in the script.

Events (cronjob)

Rules

Black to cloudwatch event

create rules

○ Event pattern

○ Service name - EC2

○ Event type - EBS

(or) ○ Schedule

○ fixed rate of

○ cron expression

⊕ Add target (many types)

Ex:- Lambda function

EC2-stop