DevOps roles and responsibilities:

Hi Iam Manisha Alivel. I am having 4 years of experience in the IT industry as a DevOps engineer .

Currently working in Claro software solutions pvt.ltd

Presently, I am working on a learning domain. We have a team for 5 including TL

So, I was assigned with an AWS team and the DevOps team.I am a part of the DevOps team, We give 24/7 support for these clients.

As part of my role ,I am responsible for setting up & managing DevOps CI/CD pipelines.

I have used various DevOps tools like Git (for version control system), Jenkins (for continuous integration AND continuous delivery), Maven (as a build tool), SonarQube (for code analysis), Nexus(for artefact repository) ,Ansible(for Continuous deployement and Configuration management ) Docker (for containerization) and Kubernates (for Container Orchestration).

We use Jira for ticketing. Kanban is our dashboard. so we start our day with stand up calls and work depending on the tickets.

Created and Maintained Github repositories for various projects like frontend projects like ( Angular, React ,Java etc ) backend projects like ( PHP,Laravel,Symfony , Ruby etc), Creating various branches ( like prelive , live , QA branch , UAT) and tags for multiple projects.

We use Jenkins for creating jobs.I use Jenkins for different Java projects. We have around 100+ jobs running currently in our company.

It helps in the Continuous Integration tool (written in Java) and automation server to continuously monitor the large code base in real-time. .

It enables Developers to find bugs in their code and fix them.

Email notifications are made to the developers regarding their check-ins as a post-build action

To create a job we have 3 types like using plugins, scripted and declarative. We use declaratives in our organisation.

Jenkins Declarative Pipeline we will Automate all the jobs in a single scripted file for saving time and reduce repetitive tasks

pipeline

agent

stages

steps

action1

action2

pipeline {

agent any

stages {

stage('Welcome Step') {

steps {

echo 'Welcome to LambdaTest'

}

}

}

}

---------------------------

pipeline {

agent any

stages {

stage('Fetch code') {

steps {

git branch: 'name', url: 'url link'

}

}

stage('build'){

steps {

sh 'mvn install'

}

}

stage('Test') {

steps {

sh 'mvn test'

}

}

}

}

DOCKER

--we used docker for containerization. Like to create a image we used to write a docker file. With the image we create a container which has all applications and dependencies in it.Using containers, developers can easily (re)deploy an image to any OS. Just install Docker, execute a command, and your application is up and running.

Dockerfile For Nginx

FROM ubuntu:latest

WORKDIR /tmp

RUN apt-get update

RUN apt-get install nginx -y

COPY index.html /tmp

ADD testfile.tar.gz /tmp

EXPOSE 80

CMD ["nginx","-g","daemon off;"]

What is the difference between ADD and COPY?

COPY and ADD are both Dockerfile instructions that serve similar purposes.

They let you copy files from a specific location into a Docker image.

COPY command will copy files/directories from your host machine to your image

ADD command will copy files/dir but can extract compress file to destination

CMD command it will execute when u start the container

Docker Commands:

docker ps List running containers

docker ps -a List all the Containers start and

stopped also.

docker exec -it <container name> /bin/sh SSH into container

docker restart <container name> Restart a container

docker stats Show running container stats

docker system df Check docker daemon disk space usage

docker system prune -af Remove images, networks, containers, and volumes

docker ps -s List all the Running Containers with the File Size

docker ps -q List the IDs of the Running Containers

docker start <container\_id or container\_name> Start a Docker Container

docker stop <container\_id or container\_name> Stop a running Docker Container

docker restart <container\_id or container\_name> Restart a Docker container

docker pause <container\_id or container\_name> Pause a running Container

docker --version

service docker start

service docker status

docker pull image name

docker push image name with tag

docker images shows images

vi dockerfile creating dockerfile

docker build -t imagename . creating image from dockerfile

docker run -it --name contname imagename /bin/bash creating container from image

--I also have good knowledge on Ansible and terraform also but i did not get a chance to work on it. Like in Ansible, if we want to install the OS in 50 systems previously we need to do it manually in every system, this is a time taking process. But with Ansible we can install os in all the systems by writing a playbook. This will reduce the time. Like we give HOST, BECOME, TASK. We can execute playbooks with commands like ansible-playbook manisha.yml.

The Ansible inventory file defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate

TO INSTALL APACHE

---

- hosts: web servers

become: true

tasks:

- name: Install Package

yum: name=httpd state=present

- name: Start httpd service

service: name=httpd state=started

--In terraform also we can create the resources automatically by executing the user data script.Like in user data script we give PROVIDER, REGION, ACCESS AND SECRET KEYS, NUMBER OF INSTANCES TO CREATE AND VPCS, SUBNET, ROUTE TABLE.

--we need to create a .tf file and in that file we need to provide the user data script then we need to execute with different commands like terraform INIT( it'll download the required plugins), Next, terraform format( It'll format the structure), Nxt terraform VALIDATE( it'll validate the terraform code), then terraform PLAN and last is Terraform APPLY( it'll execute the .TF file) and we have a command like Terraform DESTROY to delete all the resources

provider "aws" {

region = "us-west-2"

}

resource "aws\_instance" "instance" {

ami = "ami-0ddb956ac6be95761"

instance\_type = "t2.small"

key\_name = aws\_key\_pair.my-pub-key.id

vpc\_security\_group\_ids = [aws\_security\_group.sg.id]

subnet\_id = "subnet-xxxxxxx"

associate\_public\_ip\_address = true

root\_block\_device {

volume\_size = 50

delete\_on\_termination = true

}

tags = {

Name = "Citizix-Debian-Server"

}

}

resource "aws\_security\_group" "sg" {

name = "Citrix-Server-SG"

description = "Restrictions for Citizix server"

ingress {

from\_port = 22

to\_port = 22

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"]

}

vpc\_id = "vpc-xxxxxx"

tags = {

Name = "Citrix-Server-SG"

}

}

output "instance-private-ip" {

value = aws\_instance.instance.private\_ip

}

output "instance-public-ip" {

value = aws\_instance.instance.public\_ip

}

--Kubernetes--

In our company, we use Kubernetes to run containers, manage them, automate deployments, and scale deployments.

we do this by AWS EKS.we use EKS to run KUBERNETES on AWS without needing to to install and maintain

own Kubernetes control plane or nodes.

Which helped us for Scaling the K8s For ensuring High availability across multiple AZs.

Automatically scales control plane based on load, detects and replaces unhealthy control plane instances.

It is integrated with other AWS services for Scalability and Security.

ECR for container images

ELB for load distribution

IAM for authentication

Get node resource usage = Kubectl top node

Get pod resource usage = Kubectl top pod

Get resource usage for a given pod = Kubectl top <podname> --containers

List resource utilisation for all containers = Kubectl top pod --all-namespaces --

containers=true

Delete pod = Kubectl delete pod/<pod-name> -n <my-namespace>

List all pods = Kubectl get pods

List pods for all namespace = Kubectl get pods -all-namespaces

For logs = Kubectl logs my-pod -n namespace

To check linux OS = uname -a or cat /etc/os-release

Get pod info = Kubectl describe pod <podname>

List running pods = Kubectl get pods –field-selector=status.phase=Running

Scale out = Kubectl scale --replicas=3 deployment/nginx-app

Check update status = Kubectl rollout status deployment/nginx-app

DATADOG

I am using a data dog as a monitoring tool for my organisation.

With the data dog that I have attached to my AWS account with the data dog with which I can monitor all the EC2 instances have created various dashboards, in which I can see the server status, CPU utilisation memory and all the optimization of loading of various servers. I can manage the logs for various front end and back end applications.

SECURITY

One more important thing is making all the repositories private in GitHub, and keeping the Jenkins jobs secure, and keeping the access on the production servers to only limited users like only the users who have got good understanding and working on production servers. only get access to the production instances.

--AT one time. The customer's base was increasing and the traffic was increasing on the different servers and it became mandatory to scale up.

So, along with a team I started writing Docker files for different applications. Like Front end We had Angular and react.js and backend as Node js, php .we were facing frequent challenges like server down every time we had to increase the type of instances.

So, we have migrated all these applications or back end applications also to the Kubernetes. So I got a very good understanding of application end to end deployment lifecycle, like how to create automatic triggers from github, then how to deploy it to the service using Ansible and then one up scanning this by using tools like sonar cube.

----

Our team has saved almost $25,000 for the company by moving all the on-demand instances and RDS to the reserve instances. We have brought many reserve instances like many of our front end and back end applications

When I joined the company there were only Pre live and live environments. But coming today we have got four environments. Now we have Pre live, live we have a QA environment and we have UAT environment

COMING TO AWS:

I also have got a chance to work on various AWS services like ec2, s3, Vpc, IAM roles (Like we used to give the permissions for the developers to access a few services like s3 buckets), Autoscaling and LB.

--Like in autoscaling if the traffic is increasing it'll create the servers automatically. here we have two types of scaling: HORIZONTAL and VERTICAL SCALING, horizontal scaling means we can increase the servers and in Vertical scaling we can increase the power of machines like RAM and CPU...

--Like In LB it will distribute the traffic to different servers and reduce the latency. We have 3 types of LB they are CLASSIC, APPLICATION AND NETWORK LOAD BALANCERS..

--Classic Load Balancers (CLB): Provides the basic level of load balancing across several EC2 instances.

--Application Load Balancers (ALB): Pushes the traffic across multiple registered targets within different Availability Zones. It is used for Microservices

--Network Load Balancers (NLB): Uses TCP/IP protocol for distribution of traffic across several Servers. It is used to reduce the latency and can handle millions of requests for second.

If asked (E2E deployment)

We use github, we use Jenkins, we use sonarQube, we use Ansible for deployment and finally we get notification in Slack. So, for example if I want to say end to end deployment of the CI CD pipeline.

Initially, we start with Jenkins and in Jenkins we give credentials of my github and then add the main branch after giving the branch then we SSH into the server. Then the build command will run. Then the deployment happens by using Ansible runs the different build commands in various servers which are connected by using Ansible inventory files and in playbooks we have the various commands to run in different applications.

So, once deployment is done like we have Kubernetes manager with which it goes it runs commands like Kubectl apply -f. It deploys the application and the application will be up and running. This is the entire deployment lifecycle.

Roles in Devops:

1.PLAN 2. CODE 3.BUILD 4.TEST 5.RELEASE 6.DEPLOY 7.OPERATION 8.MONITOR

End to End project flow:

First thing developer makes the code change once he have continuous pipeline setup then the entire process will execute automatically so developer makes the code change into a remote repository we are going to use github as soon as there is new commit in github

Jenkins server fetches the latest changes automatically,build it,run some tests,unit test,code analysis,all the kinds of test and give the information to developer on slack for notification,then the next job the pipeline will be executed which will going to package artifact

Version it,store it some remote software repository like nexus once that is done the notification will sent then the next job then trigger automatically which is going to deploy the artifact to staging environment once it will deploy to staging environment they will few more test conducted integration test,load test,any software testing so there will software testers

Test the scripts we are going to execute from a windows server which is going to checkout staging environment for any bugs,any errors if every thing is good then again the notification will be set the artifacts then we store some were else for production deployment like s3 or we can keep it in nexus repository also and this will be very will tested artifact or same aritifact we can deploy to production we have till here continuos delivery we wait for the approva

Ansible playbook

Ansible playbooks are written using with YAML So create .yaml extension

Sudo vi nginx-install.yml

—

* name: nginx install & start services

hosts: all

become: true

tasks:

* name: install nginx

yum:

name: nginx

State: latest

* name: start nginx

Service:

name: nginx

start: started

Integrating sonarqube with jenkins using maven

Click on manage jenkins,manage plugins,sonarqube for scanner for jenkins plugin install ,