**DEVOPS**

\*DAY1

What is devops? It is a culture that improve the organizational’s ability to deliver their application. Devops is a process of increasing the ability to deliver your applications in the

-Improve the delivery

-improve the automation

-improve the quality of the application & testing

-monitor the product

🡪it is a process of improving the application delivery quicker by ensuring that there is a proper automation, proper quality is in place, proper continuous monitoring and a continuous testing.

For example if an application is deliver in 2 weeks as devops engineer, you have reduce that to 1 week or couple day by ensuring that you have automation, quality continuous monitoring and testing in place.

🡪as devops engineer I have the capacity to adapt myself to a new tools

In the past I am coming from system admin to system engineer and couple years from now I have transitioned devops engineer. I was very passionate learning about those tools and it was not very difficult for me the fact that I was used to them as a system engineer such as automated system configuration, system infrastructure. In current organization I take care of automation. I ensure that quality is maintain for the application, I ensured that I have setup continuous monitoring and I have automated testing process into devops lifecycle. For the CI/CD I am using Jenkins and Github action, for the deployment of application into container orchestration I am using Kubernetes, for system configuration I am using ansible and terraform for building infrastructure automation.

\*DAY 2

Software Development Lifecycle (SDLC): is a standard follow by an organization, it is a process or culture used by the software industry to design, develop, and test in order to deliver high quality product. Example of E-Commerce

Planning/requirements🡪defining🡪designing(High Level Design/Low Level Design)=system has to be scalable, HA🡪Building🡪testing🡪Deploy.

As devops engineer your main focus will be improving and automating “Building, testing and deployment” of the application :

Building: developer write the application code based on their favourite language using the designing document then push the code to common location = Source Code Repository (Github)

🡪Testing: Application stored in Git is deployed in the server. We have Quality Assurance engineer take care of testing the software

🡪Deployment: After testing, application is promoted to production for customer

Devops engineer will ensure all those three process is completed quickly in automated way.

In order to complete the project we will use Agile module by taking in short sprint. Each sprint is a completion of each steps of process.

Day3: VM

Day4: AWS & Azure

Day5: AWS cli (download iTerm) aws cdk , aws api, terraform

In order to connect to aws server, you must have yr key value-pair that you have created on aws.

ssh -i /location\_of\_key\_pair\_file username@server\_pub\_ip.

For the first time you will get an error “permission 0644…….pem are too open

#chmod 600 /pem\_file

Try again to login

\*download aws cli in your machine; create access key which will allow you lo get access to aws console form aws cli; login into it and type: aws configure. This help you get access aws resource from cli

🡪enter both key, region as well.

CloudFormation

Day6: Linux & shell

Linux: secure OS; free open source; fast

Linux: OS🡪Kernel🡪System Library🡪compilers;Users processor;system software

Day-9: Git and Github: Version control system

\*Sharing code

\*versioning

🡪Previous Version control system such as SVN & CVS are Centralized version control but Git is Distributed version control system. For example on centralized version control, we have a central server from which version member access and share their code. Here we have single point of failure. Compare to Distributed system, each member can create a copy of the distributed system = original source which is called fork. So even though the distributed system goes down, each member still have a copy of the source code.

Diff between Git and Github:

Git is an opensource. You can install it on the server and allow every developer to commit the changes to the git server.

Github use same concept as Git and building solution such as (issue, commenting, reviewing project management

1-create a dir example.com

2-make it as git repo: **git init** ; ls (To verify). You will see a dir called “.git” which is created to track the entire repo. In that dir, you will see “HEAD, config hooks objects refs”.in the “**objects**” it is the where the tracking will take place. **Hooks** help to prevent unintentionally passwd or api. **Config** for configuring git creds

3-git lifecycle: **git status** tell you what file to track or unstage a tracking file. In case you have modified the adding file, use **git diff** to see the exact changes you have modified

git add ; git commit -m “comment” then use **git log** to verify or track yr commit; in case you want to rollback on previous version on whatever you have committed (go back to previous changes), use

git reset --hard <commit\_id>

When you decide to share your code with other you have to either Github or self hosted or bitbucket. First create repository in github. It does not have to have same name with yr local repo.

What is Git Branching Strategy? Instead of modifying your main branch for bug fix or new feature, you clone the main branch which will create a new branch, do you modification, test if it is good then you are going to merge the change to the main branch. Before merging you create a release branch which you share with the customer. After approval, you go ahead and merge it to Master branch. The change has to be review by 2 or 3 reviewers then finally merge to the main branch.

**Release branch** is created in order to deploy the new apps version to the customer. No changes will be merge to that branch. The fact that it is for testing. Master branch is usually for active development.

**Hot fix or feature branch** is a quick branch created to fix an issue on the apps. After the fix, the change will be reviewed and merge to master.

**Master branch**🡪feature/bug\_fix branch🡪Release Branch🡪Merge\_to\_Master branch

Day 11: Git commands & Interview

After doing git add and git commit, do git push.it will not work because the repo is created to the CLI and there is no remote reference locally (‘git remote -v’ to check). In order to add a remote repo do: git remote add “url/repo\_nameWhereas you should clone (download) the repo from github by using **git clone url….**for the url you can use https (on which you use username/passwd) for ssh (use public key).

To create a branch and switch to it: **git checkout -b branch\_name** (git branch to verify).

To switch to another branch name ‘division”: **git checkout division ;**

There is 3 ways to merge: git merge; git rebase; git cherry-pick

🡪git cherry-pick (you will choose what “commit” you want to merge). Do **git log branch\_name** or **git checkout branch\_name && git log** in order to see all the commit in the branch, then copy the Commit\_ID and use **git cherry-pick commit\_ID**

**🡪**in case you have multiple commit, use **git merge or git rebase**.

Difference between both is with **git merge there is not linear fashion of the changes**. With **git rebase there is a linear way, you will get much cleaner project history**. When you do git log, the new update will come after the old commit. Git rebase will help you track which commit come after which commit.

Day 13: could service to know: EC2 ; VPC; IAM; EBC-S3-EFS ; Cloudwatch (Monitoring); Lambda (send mail notification,serverless function), cloud build service, (aws configuration to define and keep track of configuration in aws), aws kms (Key management service for managing key, certificate, encryption), CloudTrail (Fo keep track of what is going on in aws) , AWS EKS , Fargate (container solution), ELK ( Elasticsearch,logsearch,kibana it is logging search mechanism))

Day 14 & 15: Configuration Management Ansible

Day 16 & 17: Terraform ofr

\*managing yr infrastructure,

\* track yr infrastructure with tf.state file

\*automate changes

\*standardize configurations.

Terraform lifecycle:

1-write yr terraform configuration file (harshicorp terraform aws to find info about aws infrastructure creation) . the config file will create an API for the target cloud provider; 2-Plan review the changes terraform will make to your infrastructure. In case for any error it will make a notice; 3-Apply==Terraform provisions your infrastructure and update the state file = terraform.tfstate

NB: make sure all the variable are input variables.tf file mostly. So in case you want to modify anything in config file, you do it there.

Before running terraform, you must set up the authentication. For aws, install awscli, then run aws configure using Access/Secret key in order to authenticate to aws. For azure use azure service principal.

For good practice always store your state file = terraform.tfstate (used to track whatever you are creating) remotely in remote backend = remote storage services (S3). Do not manipulate it locally.Make sure the permission of the file is read only.

Devops engineer write Terraform file and store it in github repo. Then the from jjenkins it will be executed to create infrastructure on AWS. The terraform state file generated will be store in Amazon S3 and integrate them with proper lock ing solutions such as DynamoDB.

We use terraform with S3 bucket as remote storage and DynamoDB for locking. The purpose of thoseS3 &Dydb for example during a case of parallel execution, the state has to be used only by a single user so by using dydb, you can introduce a mechanism of locking that’s how you can make only one person to use terraform.

Problems with terraform:

State file is single source of truth;

Manual changes to the cloud provider can not be identified and auto-corrected;

Trying to be used as configuration management.

Day18: What is CICD

Unit testing

Static code analysis = to check the syntax, apps is well formatted, unnecessary variable;

Code quality / Vulnerability

Automation = testing the functionality of that specific function; end to end testing

Reports= having the reports of all testing

Deployment.

Developer🡪store code in Github🡪Jenkins (orchestrate all tools in order to build the pipeline)

Tools are: Maven (for java code); Sonarqube; Nexus; K8S/Docker

Jenkins will deploy the apps in different ENV (Dev🡪QA🡪STG🡪PROD) After manual or automated approval. For example on Dev you might have 1M,1WN; STG 3M,5WN; PROD 3M/13WN

Day 19: Jenkins https://www.jenkins.io/doc/book/installing/linux/#red-hat-centos

1-install java