**Introduction:**

I have 11+ years of exp and

have good experience on devops tools like

bitbucket - git concepts

Maven for build

Jenkins for ci cd

Docker

Kubernetes

Ansible

Terraform

I am also good in cloud such as aws

Have worked on some of the services like

S3, ecs, eks, cloud formation, ec2, vpc, iam network level

Aws - keys in aws - public key n private key

**Day to Day activities:**

* I**n** Devops Work, i work on Several User stories, several tasks, on New AUtomations, CHanges to Pipelines, Executing & MOnitoring Monthly/Weekly/Daily Builds.
* I do the Release Management work using Jenkins.
* I am responsible for Ensuring that Monitoring is Done on Each Asset and Ensure we are getting the logs.
* 25% of my work is on exploring new Technologies and doing POC on new tickets and Provide the Feedback and Make Production Ready APplications.
* Day starts with Checking the tickets, Responding to them and Gatekeeping Production Environment.

# Devops Definitions & Path

What is Devops according to you ?

Answer:- Devops is filling a gap b/w development and operations etc", in My view Devops means its a Problem solving skill with Technology and Automation. With Devops Organizations can streamline thier whole IT Process using Automation and minimize the IT Efforts and staff by replacing them with Robots and Software. while there are pros and cons to this in my view its a problem solving skill.

In Reality we have different Phases during Software Development life cycle, like Plan, Code build , Package, Test, Deploy, Monitor and security lies at every stage in the whole Process, however we have certain important elements to practise these called CI/CD/CF

**What is CI (Continous integration)**

Lets say you have multiple developers in your team working on bugs and each person is contributing to the same application by pushing thier code on a version control system like github/svn. When a developer pushes a change to the Code to the Code Repository it is linked to a build Management System like Jenkins which is responsible for packing the code, eg:- In case of java, you can think of a jar file. Since Multiple Developers are working on same Application, we need to ensure that it builds correctly and does not introduce any new issues during build phase. "**This Process of Enabling Multiple Developers to work seemlessly on the same Application without stepping on each other's toes and ensuring that new changes integrate with Application without introducing any new issues is called continous integration**"

Eg of CI tools :- Jenkins ,Code Pipeline etc.

Eg of Builds :- For Java its Maven, Gradle, Ant

For Python its PIP , For Angular/Node js its NPM, For .net it is MS Build.

**What is CD (Continous Delivery)**

Once .the Buges are fix and the build is validated, its time for creating a shippable , ready product to be deployed wherever possible, Continous delivery is a form of a Creating a Software shippable product to be able to deploy on a Production Environment. When you say its continous , it means you are automating the Process by using a event based mechanism that can understand the Code Commits on a Release branch and run the delivery process. In Short "**Releasing a Software by an Automated Pipeline is called Continous Delivery**" . The Shippable products are typically placed in a Centralized Location called "Artifactory" and the shippable products or software are called "Artifacts" , sometimes if these are docker images, they can be placed in "Secure Private Docker Registry like ECR/ACR"

**What is CD (Continous Deployment)**

**"The Process of taking a shippable product or the software and deploying that on a Environment (Production/Non Production) which can be a either an On-Prem data center or a Public/Private Cloud using a Automated Pipeline is called Continous Deployment**

**CICD Pipeline tools:**

Maybe I can go through the pipeline that something I can discuss  
**Tools:** Jenkins was the heart of the orchestration integrating all these different tools so source control have used SVN in the past but now I am using GIT because of branching strategies all of that and the flexibility that we have. I have used maven and npm for builds I am used toffle for scanning the repositories for secrets , I have also used artifactory for pushing the artifacts

Using sonarqube for detecting the bugs , code smells and pretty much selenium and sahi for functional testing but I am not like pro in selenium and sahi but I work with testing teams to create those scripts and I am responsible for creating the frameworks so that everything is orchestrated right so that’s the CI pipeline and also CD pipeline ,

**CD:**  
 I work with different kinds of applications may be we can call it has Ec2 based, serverless and container based .for ec2 based I used packer and I building those ami’s and I used regular launch configurations , auto scaling groups

and I have used ansible playbooks to be able to copy the artifacts from s3 or artifactory depending on the project to these instances bootstrapping and I have created those bootstrap scripts as well and for serverless I have used API Gateway and Lambda , S3, Cloudfront and more or less I have used all of these services and I have used Cloud formation , SaM templates pretty much. using cloudformation stacks to be able to do the orchestration of serverless platforms in AWS and for container I have created builds using code build, code pipeline and for the deploying them in the cluster using code deploy. I have also used helm charts for container orchestration as well.

**CICD Process Technical- CI stages:**

I worked with creating a generic pipeline that is reusable of course every pipeline is can not be 100% reusable for different applications but I will explain you the pipeline that recently I have created so basically we have a git and gitlab enterprise here basically we have different branching strategies like you know we have all development going on you know on the development branch and any features are created in feature branches .

So the first thing what we do when we start the automation project is we able to orchestrate with Jenkins so the Jenkins will be looking at the branches and so there is any updates for the branches if there is any update basically it clones using plugin that available SCM plugin and we begin that CICD process so then we have like something called a complain check stage It’s a declarative pipelines that we have and here we have complain check stage where I am checking for secrets and I am checking for post received hooks you know sometimes after the commits

Here we want to do some checks to make sure for Ex: Dockerfile we want them to use the only basic details those kind of checks are done in the complain checks so that they are not using anything from dockerhub application teams are not using so things like that and also have this naming standards also the part of the complain checks because we have various apps when we go with the generic model and there is a share tendency model in the aws account so we want to make sure they are following the standard naming conversion

Then build stage depending on the project and I have worked with the java based applications using maven again maven clean build clean package to be able to build them and I have used npm to build nodejs projects and I also have angular projects not many of these angular projects , most recently I have been working with angular projects and you know that one thing I wanted to say that basically single page applications here been built there and pushed to s3 they all use different model

Builds are tested in the test stage I have unit test stage basically I am try to check running this and generate reports sometimes we have also like dbunit and I have used Liquibase as well basically to orchestrate the database side thing right like DDL’s and DML’s because that’s one part where everything has been manual and I am recently been working with Liquibase as well.

We used sonarqube and we also have quality gates but not everything ideally passes all these stages because some of them are old applications so we trying to set quality gates at 70%

If they pass then basically it goes to the next stage

We have Jfrog artifactory and S3 so depending on the kind of application we use different tools as well so whether it’s artifactory or S3.

**CICD problems faced situation:**

Each application is different Some applications are migrated from onprem to cloud They have been deployed by engineer Manually I am responsible for automation Bash scripts Python Etc

So when I create the jenkins pipeline, they are the step s that I have to automate Sometimes application teams dont give all steps So I have to add the steps if we find out the issues

These applications are old and unmaintained, so when we automate, we have to figure out what are all the moving pieces

**CICD Stages :**

Dceclarative pipeline

Various phases

Checkout

Check for secrets

Build

Scan for vulnerabilities

Sonarqube

Push to artifactory

--

Cd

Either serverless - sam templates, cloudformaiton

Docker - ECS

Ec2 - ansible playbook

Each cd step has predeploy

Deploy

Post deploy

**Shared library:**

It’s a collection of independent Groovy scripts which we pull into Jenkinsfile at runtime. The best part is, the library can be stored, like everything else, in a Git repository. This means you can version, tag, and do all the cool stuff you're used to with Git.

**Git revert:**

First use git logs command to check which commit you want to revert then use git revert hash code to revert that commit.

# Github

**Branching & Merging Strategy**

We use Git flow branching.

1. According to this Method, We create 3 different branches in each git repo called Development/QA & Release.

2. for us the flow is from Development --> QA --> Release Branches.

3. Development branch is used for daily and adhoc builds on Development.

4. QA Branch is for Collating all daily builds to Candidate or Production Builds. anything deployed in Prod is first deployed and tested in QA.

5. Basically when a developer want to develop any featuere , he will create his own branch which is a copy of dev branch and does the development

6. Developer then raises a Merge request with source as his branch and destination as Development branch.

7. Once the Code is in Dev Branch a merge request raised to QA and QA Deployment is done and here we do security testing and unit test and feedback.

8. Once everything is ok , QA --> Release, anathor merge request is raised and final build for prod is generated and deployed.

**Onprem- cloud:**

Yes I used cloud endure to migrate from vmware to aws ec2 ami is taht when we are onprem - we are stuck with vmware We can use ec2, we can stop, start, we only pay for the servers we use for that time If we use vmware, we are not able to use the full native connectivity with aws instrascutre

For example, we can sconnect to s3 bucket using instance profile That cannot be done when we connect from vmware We have to user iam users in that case That is not the best practice

So it is best to user ec2 and cloud native services when ever possible

**Team size:**

There is like 40 engineers the entire group including directors managers  
we have about 12-15 core engineers and I am the one of the core engineers, very hands with implementation.

There are applications are migrating from on prem to cloud so my role is to actually create those pipelines for automated for manual work that has been done and once I create that basically I create the pipeline and once it’s stable after a month I provide knowledge transfer to other engineers who is doing maintenance after doing the knowledge transfer then I will go back to the new application.

There is a rotational based and every three weeks I go for the production based support .

**How many applications you support?**

I migrated about 12 applications and I support about 5-6 applications I would say because once the model is successful there are not much you know everything is automated pretty much I do knowledge transfer to the offshore team. Basically that’s the model they have here but I am quite flexible to be able to do anything.

**Terraform locking:’**

Terraform will lock your state for all operations that could write state. This prevents others from acquiring the lock and potentially corrupting your state.

State locking happens automatically on all operations that could write state. You won't see any message that it is happening. If state locking fails, Terraform will not continue. You can disable state locking for most commands with the -lock flag but it is not recommended.

Locking helps make sure that only one team member runs terraform configuration. Locking helps us prevent conflicts, data loss and state file corruption due to multiple runs on same state file. **DynamoDB** can be used as a locking mechanism to remote storage backend S3 to store state files.

**How to recover state files in terraform:**

**The terraform state pull command and the terraform state push command** can directly read and write entire state files from and to the configured backend. You might need this for obtaining or restoring a state backup.

**Tell me How terraform works**

a. Basic Terraform code has 4 different sections, the typical extension of a terraform file is ".tf", In a terraform file "main.tf", You will see providers , resources, variables, outputs, datasources. A Provider is meant to me the infrastructure provider like AWS/Azure/GCP etc. Resource is meant to be the object thats created on the Cloud Platform. Datasource is the Properties of the Resource. Outputs is to print the created elements and its properties, eg:- printing IP of an EC2 after provisioning.

b. Terraform has 3 Basic states , i.e terraform init/plan & apply

**"terraform init"** --> This is to initialize the terraform, when you execute this command, this will create a .terraform folder in the folder where the main.tf file exists and it will download the provider specific binaries in the folder.

**"terraform plan"** --> This is to print a view of what objects will be created when you run the terraform code. this gives a snapshot of what is going to be created. However it wont create anything.

**"terraform apply" -**-> THis is the step where actual infrastructure is created, Once the terraform apply is done, it will create the infrastructure for you.

# Kubernetes

1. I have Hands on Kubernetes Creating Pods, Deployments, Replica Sets, Labels, Stateful sets, Daemon Sets, Cronjobs etc.
2. I have used Helm Charts for doing the Deployments.
3. i have Experience with On-Prem Kubernetes Creating Kubernetes cluster with Kubeadm and kubectl and also Hands on with Managed Services like AKS , EKS, GKE
4. I have hands on Kubernetes configs and secrets.
5. I also have experience using service mesh like ISTIO, ENVOY.
6. I have experience with Monitoring Kubernetes cluster with Prometheus, Grafana, Kubernetes CLuster State Monitoring.
7. i also have experience with Security on Kubernetes CLuster using Trend Micro, Twist lock and Azure Security Center.

​

**Kubernetes Architecture**

In Kuberentes Architecture, we have 3 Planes.

1. **Control Pane :-** This Components typically resides in the Kubernetes Master Node. Components that make global decisions on a cluster , for example scheduling , detecting and responding to server events in the K8s Cluster.

a. **etcd :-** Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

b. **api server:-** THis part exposes the API's of Kubernetes, THis is basically front end of kubernetes cluster.

c. **kube-schedular:-** This Component schedules all Pods and select and runs the pods where they have to run

d. **Control Manager:-** This is typically an important part and optional, To Reduce complexity on Kubernetes, Kubernetes has some inbuilt and customized controllers. these are all run and controlled by Control Manager. Each controller is a different Process.

* + **Node controller:** For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding
  + **Route controller:** For setting up routes in the underlying cloud infrastructure
  + **Service controller:** For creating, updating and deleting cloud provider load balancers

2. **Node Components:-** Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

**Kubelet :-** An Agent that runs on Each Node in the cluster , this makes sure that containers are running in a Pod.

**Kube-Proxy :-** Its a Network Proxy that runs on each Node in your cluster , Implementing Part of Kubernetes Service Concept.

**Container Runtime:-** Kubernetes follow Docker ,CRI-O, Containerd Run times, this runtime is required for running the Pods.

### 3. ADD ONS:-

Addons use Kubernetes resources ([DaemonSet](https://kubernetes.io/docs/concepts/workloads/controllers/daemonset" \t "_blank), [Deployment](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/), etc) to implement cluster features. Because these are providing cluster-level features, namespaced resources for addons belong within the kube-system namespace.

**kube-dns:-** Cluster DNS is a DNS server, in addition to the other DNS server(s) in your environment, which serves DNS records for Kubernetes services.

**web-UI(Dashboard) :-** [Dashboard](https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/) is a general purpose, web-based UI for Kubernetes clusters. It allows users to manage and troubleshoot applications running in the cluster, as well as the cluster itself.

**Container Resource Monitoring:-** [Container Resource Monitoring](https://kubernetes.io/docs/tasks/debug-application-cluster/resource-usage-monitoring/) records generic time-series metrics about containers in a central database, and provides a UI for browsing that data.

**CLuster Level Logging:-** A [cluster-level logging](https://kubernetes.io/docs/concepts/cluster-administration/logging/) mechanism is responsible for saving container logs to a central log store with search/browsing interface.

**Deployment Use cases kubernetes:**

Deployment Use Cases in Kubernetes are given below: Use Case 1- Create a Deployment: On the creation of deployment, Pods are created automatically by ReplicaSet in the background. Use Case 2- Update Deployment: Creation of new ReplicaSet happens and now the deployment is updated. Deployment revisions are updated through these new ReplicaSet. Use Case 3- Rollback Deployment: If the current deployment state is not steady, rollback of deployment happens. But we can see the container images are updated. Use Case 4- Scale a Deployment: Based on the requirement, scaling up or scaling down can be performed on each and every deployment. Use Case 5- Pause the Deployment: To apply various fixes, deployment can be paused and later resumed.

# Deployment Strategies

​

I have experience with multiple Deployment strategies

* **Recreate**: Version A is terminated then version B is rolled out.
* **Ramped** (also known as rolling-update or incremental): Version B is slowly rolled out and replacing version A.
* **Blue/Green**: Version B is released alongside version A, then the traffic is switched to version B.
* **Canary**: Version B is released to a subset of users, then proceed to a full rollout.
* **A/B testing**: Version B is released to a subset of users under specific condition.
* **Shadow**: Version B receives real-world traffic alongside version A and doesn’t impact the response.

In Our case, we are using Blue green Deployments in Kubernetes and other Environments.

# Docker:

1. Have hands on Writing Docker files and creating a Layered Approach for the Docker images which can be a combination of base images vs Application images.
2. Experience in Installing / Configuring & Securing Docker Daemon
3. Have experience and used Docker Compose i.e the Multi Container Technology to Spin up the Complete Environment Dev/Test/Prod/Sandbox.
4. Have experience installing and configuring the Docker Swarm.
5. Experience in Docker Security - Vulnerability assessment on Docker Registries using Twist lock, Trend micro, Azure Security Center etc and driving things to close.
6. Experience in yaml for Docker Compose and swarm Deployments.
7. I have also used Docker enterprise as well.