**What is DevOps?**

**DevOps is a series of processes and tools, as well as a cultural practice within a**

**company, that encourages collaboration between development (developers,**

**QA, project management, and other associated teams) and operations (systems**

**administrators, systems architects, security, and related). It is often**

**characterized by its use of fast release cycles, common tools, and overall**

**Tell me about your self**

……………………….

COMING TO THE CLOUD

AWS I have knowledge like few services I worked EC2 (elastic compute cloud) TOOL and S3(simple storage service)  AND ELASTIC CONTAINER SERVICE I worked in AWS. this is myself

workflow

1)The developer write their code into local machine (local repo) then pushed them into GIT Hub(remote repo)  
Note: if any one changed or updated code into master repository then The job will be executed (POll SCM)  
2)The Jenkins will fetch the code into workspace then compile and build project as packages (based on life cycle commands we mentioned in job)  
  
3)The Jenkins will communicate to SonarQube for code quality and security i.e suppose if report is green it will go for further process

if not ,then send email to team for status of your application code

4)The Jenkins will send WAR file to nexus repository for sharability  
5)if code quality checks were passed then Jenkins will deploy WAR file into Tomcat container  
6)suppoose if build fails ,then send email notification to team as Buil failed with error stacktrace  
  
  
Note : we are using Node JS application for UI

**Work flow**

Developer once the implementation for there JIRA story is done, they will be pushing the code to the GITHUB.

i will be creating Jenkins pipeline to run the selenium test on the latest code which is available in the GITHUB. If any changes are required for the exiting Jenkins job I will updating the jobs

Once the code is committed, the Jenkins job's will be triggered and the Jenkins job will run the selenium test's. If the Jenkins job is completed without any issue, I will be deploying the package s to the tomcat server using the Jenkins pipeline and this server will be used by development team

To perform there testing or debugging the issues

Another job will be triggered to deploy the packagers to the testing servers which will be used

By the testing team and qa team

**We have Jenkins pipeline scripts**

Once the code is committed, the Jenkins pipeline's will be triggered to build the code using the Maven tool and the build package will be deployed on the the testing server and the selenium test's will run on the server.

Once the testing is completed and passed, an another jenkins job will be triggered to create the docker images using the build packages. Once the docker image is created successfully the docker images will be deployed to the DEV server.

While creating the Jenkins job, maven script will be used to download the required binaries to build the code.

<http://10.134.55.90:8080/view/Mozart/job/hip-server-master-mozart/>

/repo/vnext/repo/common/10.9.9.9\_Mozart/server

Developer pushes a code to Design server in git hub auto deploys to 10.134.37.100

Check Each day selenium Runs If everything fine we deploy design server manually on 10.134.37.56

Connector tester takes latest stable build, which is available on 56 servers and deploys on 10.134.37.53 server (separate builds of design server and rest not with the installer) Not only for QA for Developers for debugging.

We takes same builds and deploys on 108 also shares to unica team.

Future process will be different for connectors testing as we will have separate QA own setups.

**Roles and responsibileties** :

1. **Roles and responsibileties intailly day start from DSM(daily standup meeting)**
2. **We are building and deploying java application tool which I am working GIT, GITHUB, Jenkins, nexus artifactory , MAVEN, DOCKER,LINUX,Sonarqube,Core qube**
3. Creating and maintaining branching strategies in github
4. Creating Merge Request once the developer push there code into respective branches.
5. Check Jenkins job got faild and debugging the issue
6. Integrating Ui()and MW() changes
7. Creating tasks in service NOW on sprint bases
8. Following agaile metholody along with team effectively
9. Using Jenkins CICD to deploy our code into different environment.

**Daily activites:**

1)checking email notification regul

2)create github project repsority if required

3)giving the access to the developers for that repositorys

4)createing branchs for devlopers

5)jenkins build monitor

6)new projects onboarding.

7)deploy into multiple environemnts

8)server monitoring

9)applicaiton monitoring

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

config changes

instllation

configuriaon

deployment

monitoring

Git Interview Quasation.

**Branching statagy.**

**We are following 4 branching statagys one is master, feature branch, release branch ,bug fix,**

**Here developer develop there code in Feature branch once developer complete there code in Feature branch then we have to create another release branch if we get any isuues in in release branch then we have to create another bug-fix branch and here fix the issues in bug-fix branch then we have to raise merge request to master**

1) master branch - default branch (production branch)

2) development branch

3) Storybranch /feature branch

feature1 mani

feature2 kishore

feature3 murali

feature4 ajay

4) hot fix branch

We have to flow 4 brancing strategy

Developers will develop in some features like feature1, feature2, feature3, feature4

* master — This branch contains production code. All development code is merged into master.
* development — This branch contains pre-production code. When the features are finished then they are merged into development branch.
* feature-\* — feature branches are used to develop new features for the upcoming releases. May branch off from develop and must merge into develop.
* hotfix-\* — hotfix branches are necessary to act immediately upon an undesired status of master. May branch off from master and must merge into master and development.
* release-\* — release branches support preparation of a new production release. They allow many minor bug to be fixed and preparation of meta-data for a release. May branch off from develop and must merge into master and development.

**s**

Git clone

To download entire exiting repo from central repo to local repo then use below commands.

Git clone <remote\_repo> (all branchs came)

git clone -b <branch> <remote\_repo>

**Git add .**

Git commit -m “commit message”

**Git pull**

Pulling the updated code from master repo to local repo then once use pull command that files are came to working area

Git pull --rebase

Git pull origin master

Git pull =git fetch + git merge

**Git push**

**Push the code from local repo to master repo**

**Git push**

Git fetch:

fetch the code from master repo to local repo then once use fetch command that files are did not come to working area use git merge then comes files to working area.

Git fetch

**Banching stastgy:**

1)developer push the code into git hub repo then we hate have create one feature branch beacus of to fix the issue in particular feature in this feature branch then merge from feature branch to dev branch again another feature got an issues I have to fallow same process after successfully fix the issue then feature branch was delete

2)suppose if there any emergency issues will come then create another branch is call hot fix branch from the master branch then resolved the issues in hot fix branch then merge back to master and dev branches

3)another branch we have that is release branch it is fork from dev branch then in this branch fix the issues the release gets merged into master and dev tagged with a version number

Tagged:

**which may have progressed since the release was initiated**

**git clone : to download the entire repo from remote to local**

**git pull : to download updated code from remote repo to local repo here files are available on working directory**

**git fetch: to download updated code from remote repo to local repo here files are not available on working directory when will you git merge the code. then that files are available in working area**

**git push: push the change on your local repo to remote repo**

**git push origin branch\_name**

**or**

git push --set-upstream origin dev

**ex: local dev branch**

**git pull origin master**

**remote master branch**

**how to delete files and branchs in remote repo**

**files : first delete file in your local machine then add and commit it then push your branch where you file is deleated in branch**

**git push origin <branch-name>**

**branch: 1.git push origin –delete <branch-name> (delete branch on remote repo and push any branch)**

**2.git branch -D -f <branch-name> (first switch to other branch and delete your select (or) required branch)**

How to resolve conflits:

Conflicts generally arise when two people have changed the same lines in a file with two different branches

**Git rebase and git merge**

**Both integrate changes from one branch into another**

**Git rebase moves a feature branch into a master**

**Git merge adds a new commit**

**Or**

• git merge applies all unique commits from branch A into branch B in one commit with finial result

• git rebase gets all unique commits from both branches and applies them one by one.

• git merge doesn’t rewrite commit history, just adds one new commit

• git rebase rewrites commit history but doesn’t create extra commit for merging

**Continues** integration: continuesly compiling the source so that we can fix the bugs day to day and we can release the product on time

**Continues deployment** : CD is nothing but continuesly releasing the packages into diffrent environment like UAT QA and test

Continues delivery : Continuesly deploying packages into production server

**Master-slave arche structure:** Whatever the slave you will create it will connect with master and up and run Master is default node So creating slave and connecting with master is called matter slave configure

**(Or)**

Jenkins uses a master-slave-archetecture to manage distributed builds. In this archestructure master and slave communication through tcp/ip protocal. your main Jenkins server is master; a master instance of Jenkins can also execute buils jobs directoly

🡪no need to install Jenkins in your slave machine

🡪wich software you want use you can install lik git java in your slave machine

🡪ip address

🡪adminuser and password

🡪remote root directory

🡪label name

**Jenkins Plugin**

Maven integration

Green ball

deploy to container

copy atrifacts

quality gates plugin

soner quabe scanner

nexus artifacts uploader

Matrix based

Maven 2 project

Amazon EC2

HTML publisher

Join

Green Balls

Git plugin

Multijob plugin

Test Results Analyzer

Metrics

**Security**

1)matrix-based- security: to grant specific permissions to users and groups

* [Overall](https://wiki.jenkins.io/display/jenkins/matrix-based+security#Matrix-basedsecurity-Overall)
* [Slave](https://wiki.jenkins.io/display/jenkins/matrix-based+security#Matrix-basedsecurity-Slave)
* [**Job**](https://wiki.jenkins.io/display/jenkins/matrix-based+security#Matrix-basedsecurity-Job)
* [**Run**](https://wiki.jenkins.io/display/jenkins/matrix-based+security#Matrix-basedsecurity-Run)
* [**View**](https://wiki.jenkins.io/display/jenkins/matrix-based+security#Matrix-basedsecurity-View)
* [**SCM**](https://wiki.jenkins.io/display/jenkins/matrix-based+security#Matrix-basedsecurity-SCM)

**Project based matrix authoriation strategy**:It is an extension of Matrix-based security . It allows an access control list matrix to be defined for each project. when we want to give access to specific jobs to specific users so that the security of Jenkins is not compromised.

Step 1. Go to Jenkins Dashboard -> Manage Jenkins -> Configure Global Security -> Check Enable security -> Select Project-based Matrix Authorization Strategy

**Role based authoriation strategy:**

The Role Strategy plugin is meant to be used from [Jenkins](https://jenkins.io/) to add a new role-based mechanism to manage users' permissions. Supported features

* Creating **global roles**, such as admin, job creator, anonymous, etc., allowing to set Overall, Agent, Job, Run, View and SCM permissions on a global basis.
* Creating **project roles**, allowing to set only Job and Run permissions on a project basis.
* Creating **agent roles**, allowing to set node-related permissions.
* Assigning these roles to users and user groups

Build periodically:

We will make a build based on a specific time format

It will be in cron format

When will we enter the time job will start at that time

here

How do you do troubleshooting of build failures ?

Ans : common issues i faced was:

1. When we did not install dependency plugin then it will show the error : Build failed

:Required goal not found

TS: install the dependency plugin and configure it and then we trigger a build then

build will be successful.

2. Missing artifact: build failed ....it will show the error like failed to resolve artifact...one

required artifact is missing...

such type of issue will come while we removed artifact from the cache(it happens every few

months)--->

TS: rebuild that versiion of that module to get it reinstalled.

3. ....it's not a file : The error shows like : build error: failed to create assembly

It happens when the

when a module is build but one of its dependencies is built at the same time

(so its jar is already 'deleted') -> waiting until the dependency build had

finished, then re-triggering the module build fixed it

4. Invalid java version:

It happens when we didnt configure correct version we are usually using then

this error willl occur

TS:

Each hudson job has a jdk to be used - if this is e.g. set to 1.5 but the build

would actually require 1.6 you'll see the above error -> get the job properly

configured

5. Mail notification failure: if any password changes this type of issue will

come....

If the issue is related to me ....i will resolve it by my side and if not i will send mail

attaching the printscreen to perticular team / person.....

**ING work Flow**

1)The developer write their code into local machine (local repo) then pushed them into GIT Hub(remote repo)  
Note: if any one changed or updated code into master repository then The job will be executed (POll SCM)  
2)The Jenkins will fetch the code into workspace then compile and build project as packages (based on life cycle commands we mentioned in job)  
  
3)The Jenkins will communicate to SonarQube for code quality and security i.e suppose if report is green it will go for further process

if not ,then send email to team for status of your application code

4)The Jenkins will send WAR file to nexus repository for sharability  
5)if code quality checks were passed then Jenkins will deploy WAR file into Tomcat container  
6)suppoose if build fails ,then send email notification to team as Buil failed with error stacktrace  
  
  
Note : we are using Node JS application for UI

SonareQube:  
=========  
It is the tool for using code quality and security .Below were the key points will take care  
1)Reliabity : to check code for bugs are available or not  
2)Security :to check any security Vulnerabilities  
3)Maintainability : to check quality of the code like technical Debts (like variable declarations and exception handling )

There are three types of issues: Bug – A coding error that will break your code and needs to be fixed immediately. Vulnerability – A point in your code that's open to attack. Code Smell – A maintainability issue that makes your code confusing and difficult to maintain.  
  
Jenkins :  
-------  
Jenkins is an open source automation server written in Java.  
Jenkins helps to  automate the non-human part of software development process, with continuous  integration and facilitates technical aspects of continuous delivery  
  
Maven :  
======  
It is an Java based tool for compile and build your projects.  
validate - validates the project  
compile - compiles the source code of the project  
test - tests the compiled source code using a suitable unit testing framework.  
package - take the compiled code and package it in a distributable format  
verify – run integration tests  
install - install the package into the local repository (may be for dependencies)  
deploy - copies the final package to the remote repository for sharing with others  
  
  
Nexus :  
----  
Nexus is an repository where we can push your code artifacts ( like War and Jar files )  
  
  
GitHub:  
It is an cloud based repository where we can keep application coding files

Groovy script:

**Declarative pipeline**

The Declarative pipeline is a new feature that is added to create the pipeline. This is basically written in a Jenkinsfile which can be stored into a source code management system such as Git. Declarative pipelines are an ideal solution for the simple continuous delivery pipeline as it has very limited and pre-defined structure.

**scripted pipeline**

The scripted pipeline is a traditional way of writing the Jenkins pipeline as code. Ideally, a Scripted pipeline is written in the Jenkins file on the web UI of Jenkins. Unlike the Declarative pipeline, the scripted pipeline strictly uses groovy based syntax. Since this, The scripted pipeline provides huge control over the script and can manipulate the flow of the script extensively. This helps developers to develop advance and complex pipelines as code.

https://github.com/jyotheesh/INGFavBank.git

node {

   def mvnHome

   stage('Preparation') { // for display purposes

      // Get some code from a GitHub repository

      git 'https://github.com/knaveen1803/hello-world.git'

      // Get the Maven tool.

      // \*\* NOTE: This 'M3' Maven tool must be configured

      // \*\*       in the global configuration.

      mvnHome = tool 'mvn'

   }

   stage('Build') {

      // Run the maven build

      withEnv(["MVN\_HOME=$mvnHome"]) {

         if (isUnix()) {

            sh '"$MVN\_HOME/bin/mvn" -Dmaven.test.failure.ignore clean package'

         } else {

            bat(/"%MVN\_HOME%\bin\mvn" -Dmaven.test.failure.ignore clean package/)

         }

      }

   }

   stage('sonar test') {

      withEnv(["MVN\_HOME=$mvnHome"]) {

      sh '"$MVN\_HOME/bin/mvn" -Dmaven.test.failure.ignore clean package sonar:sonar'

      }

   }

   stage('deploy to nexus') {

    withEnv(["MVN\_HOME=$mvnHome"]) {

    sh '"$MVN\_HOME/bin/mvn" -Dmaven.test.failure.ignore clean deploy'

    }

   }

   stage('Tomcat Deploy') {

       deploy adapters: [tomcat9(credentialsId: 'c9ce8716-48ec-49e9-a62e-4b78922b570e', path: '', url: 'http://ec2-54-202-252-89.us-west-2.compute.amazonaws.com:8888')], contextPath: 'nk', war: '\*\*/\*.war'

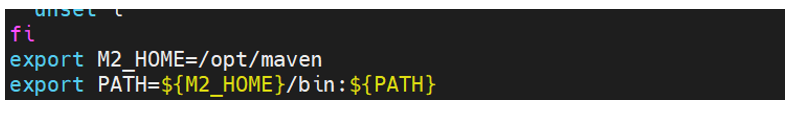
   }

}

Maven:

1.tar

2.vi etc /profile



3. Source /etc/profile

4. mvn –version

Docker:

A Docker image is **a file used to execute code in a Docker container**. ... Docker is used to create, run and deploy applications in containers. A Docker image contains application code, libraries, tools, dependencies and other files needed to make an application run.

1.docker info ---------🡪(view system wide information)

2. Docker search jenkins -🡪(show all alphine images names)

3. **docker –version**

**4.** **docker pull <image name> ---🡪** pull images from the **docker repository**

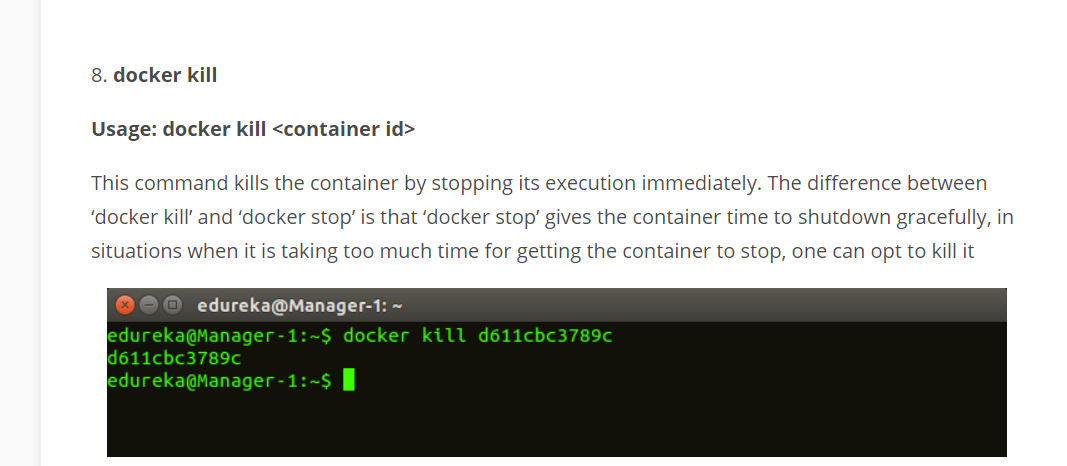
**5.** **docker run -it -d <image name>-----------🡪**create a container from an image

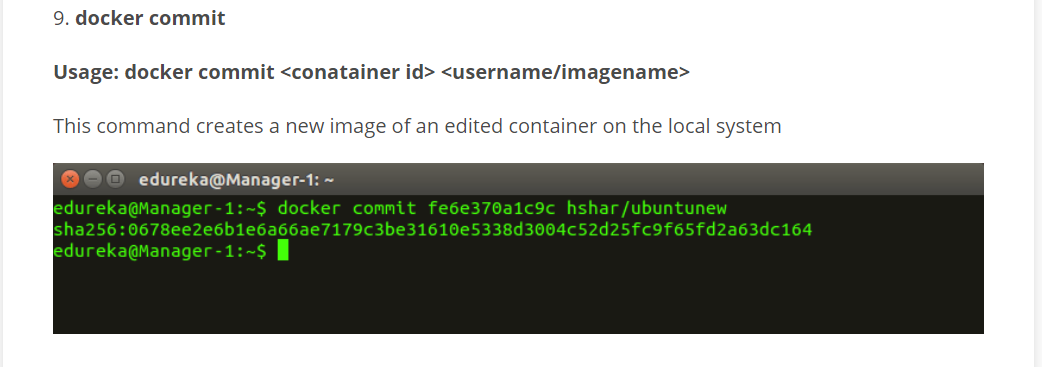
6.docker ps ------🡪(list the running containers)

7.docker ps -a --------🡪show all the running and exited containers

8. **docker exec -it <container id> bash-------🡪**access the running container

9. **docker stop (or) start (or) restart <container id>--------🡪stop a running container**

**10.** 

11. 

12.docker login -----🡪login to the docker hub repository

13. **docker push**

**Usage: docker push <username/image name>**

This command is used to push an image to the docker hub repository

14.docker images

15.docker rm <containerID>---🡪delete a stopped container

Docker rm -f <cid> <cid> <cid>

Docker rm -f $)docker ps -a)

16. **docker rmi <image-id>------🡪**delete an image from local storage

Docker rmi <image-id> <image-id>

Docker rmi -f $(docker images -q)

17. **docker build <path to docker file>----------🡪**build an image from a specified docker file

18.docker restart <container id>-------🡪restart the container

19. Copy a file from a docker container to the local system.

In this example, I am copying httpd.pid file inside a docker container with id 09ca6feb6efc to /home/geekflare/

ANS: sudo docker cp 09ca6feb6efc:/usr/local/apache2/logs/httpd.pid /home/geekflare/

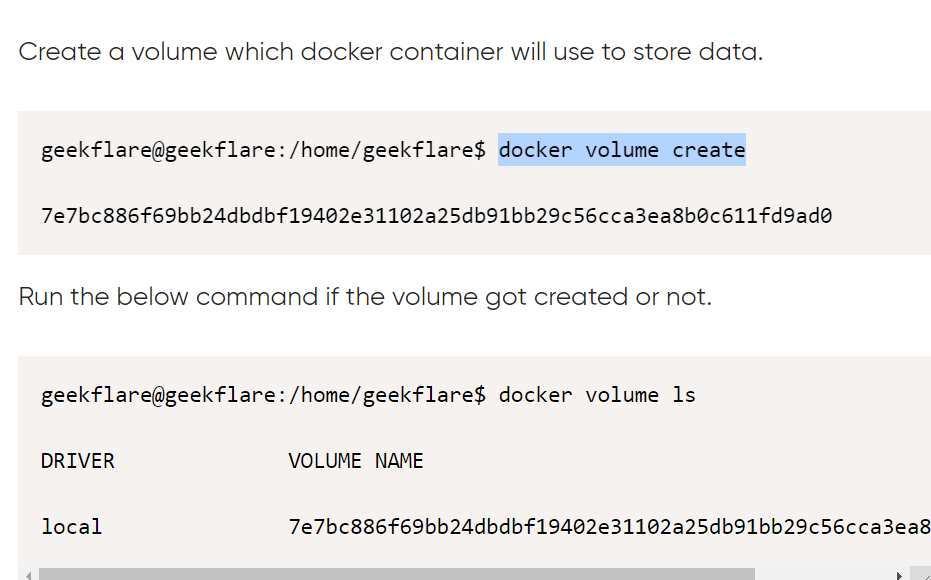
20. docker history <Jenkins image>

Shows the history of a docker image

21. docker logs 09ca6feb6efc(container id)

22. docker update -c 1 2f6fb3381078(cid)

update the CPU configuration of docker container with container id

23. 

24. docker plugin install vieux/sshfs DEBUG=1

25.docker logout---🡪logout from the docker hub

26.docker inspect cid

27.create tag for images:

Docker tag <image\_id> <imagename:tagname>

Docker push image\_name:tagname

28.change image name:

Docker tag “old-image-name” “newimage -name”

29.showing container ip address

Docker inspect cid

Docker inspect cid | grep ipa

30.docker run -it -d -u root –name Jenkins -p 9080:8080 -v jenkinsvol(host-machine-path) :/var/Jenkins- home/workspace(cid-path) Jenkins:latest

31.change container name

Docker rename ocid new-cid

32.showing all networks

Docker network ls

33.showing specific network information

Docker network inspect network-id

34.create new network

Docker network create –subnet 10.1.0.1/24 --gatway 10.1.0.1 <network-name>

(Or)

Docker network create –subnet 10.1.0.0/16 --gatway 10.10.0.1 --ip-range 10.1.5.0/24 -- driver=bridge –label=network-new <new network name>

35.add network to container

Docker run -it -d –net <network-name> ,<image-name>

(or)

Doker run -it –net <network-name> --ip 10.1.5.55 <image-name>

## What is a Dockerfile?

* A Dockerfile is a text configuration file written using a [special syntax](https://docs.docker.com/engine/reference/builder/)
* It describes step-by-step instructions of all the commands you need to run to assemble a Docker Image.
* The docker build command processes this file generating a Docker Image in your Local Image Cache, which you can then start-up using the docker run command, or push to a permanent Image Repository.

<https://codefresh.io/docker-tutorial/build-docker-image-dockerfiles/>

ADD commad is use to copy war file from a spectifc URL to a directory in the image.

COPY command is use to copy a file from local to a image.

CMD can be override by passing the command while running the image.

ENTRYPOINT can't be override by passing the command while running the image.

Need to use dos2unix for the sh script while copying the flies from windows to centos or linux.

Aws:

Load balancer: to distribute the traffic to health instances

Ex: suppose we have use 4 instances two instances are stopped load balancer will distributer traffic only running instances

AMI: to create a virtural machine using exe

**Maven:**

**1.Generate project structure**

Mvn archetype: generate

**2.Structure number you have to give like :1591**

**Asking**

**group id, artifact id, version 1.0 snapshot:, package**

**by default get number is jar**

**by default number (1591 +5)= war**

### **How is DevOps different from agile methodology?**

[DevOps](https://www.simplilearn.com/tutorials/devops-tutorial/what-is-devops) is a culture that allows the development and the operations team to work together. This results in continuous development, testing, integration, deployment, and monitoring of the software throughout the lifecycle.

[Agile](https://www.simplilearn.com/tutorials/agile-scrum-tutorial/what-is-agile) is a software development methodology that focuses on iterative, incremental, small, and rapid releases of software, along with customer feedback. It addresses gaps and conflicts between the customer and developers.

### **What are the different phases in DevOps?**

The various phases of the DevOps lifecycle are as follows:

* **Plan** - Initially, there should be a plan for the type of application that needs to be developed. Getting a rough picture of the development process is always a good idea.
* **Code** - The application is coded as per the end-user requirements.
* **Build** - Build the application by integrating various codes formed in the previous steps.
* **Test** - This is the most crucial step of the application development. Test the application and rebuild, if necessary.
* **Integrate** - Multiple codes from different programmers are integrated into one.
* **Deploy** - Code is deployed into a cloud environment for further usage. It is ensured that any new changes do not affect the functioning of a high traffic website.
* **Operate** - Operations are performed on the code if required.
* **Monitor** - Application performance is monitored. Changes are made to meet the end-user requirements.

### **How will you approach a project that needs to implement DevOps?**

The following standard approaches can be used to implement DevOps in a specific project:

**Stage 1**

Discussion about process and implementation for about two to three weeks to identify areas of improvement so that the team can create a road map (model) for the implementation.

**Stage 2**

Create a proof of concept (PoC). Once it is accepted and approved, the team can start on the actual implementation and roll-out of the project plan.

**Stage 3**

The project is now ready for implementing DevOps by using version control/integration/testing/deployment/delivery and monitoring followed step by step.

By following the proper steps for version control, integration, testing, deployment, delivery, and monitoring, the project is now ready for DevOps implementation.

### **What is the difference between continuous delivery and continuous deployment?**

|  |  |
| --- | --- |
| **Continuous Delivery** | **Continuous Deployment** |
| Ensures code can be safely deployed on to production | Every change that passes the automated tests is deployed to production automatically  (every change in the code to be deployed for testing) |

### **What is the role of configuration management in DevOps?**

* It helps with the administration and management of multiple servers and maintains the integrity of the entire infrastructure.

### **How does continuous monitoring help you maintain the entire architecture of the system?**

Continuous monitoring in DevOps is a process of detecting, identifying, and reporting any faults or threats in the entire infrastructure of the system.

### **What is the role of AWS in DevOps?**

AWS has the following role in DevOps:

* **Flexible services -** Provides ready-to-use, flexible services without the need to install or set up the software.
* **Built for scale -** You can manage a single instance or scale to thousands using AWS services.
* **Automation -** AWS lets you automate tasks and processes, giving you more time to innovate
* **Secure -**Using AWS Identity and Access Management (IAM), you can set user permissions and policies.
* **Large partner ecosystem -** AWS supports a large ecosystem of partners that integrate with and extend AWS services.

### **Name three important DevOps KPIs. (key performance indicator)**

The three important KPIs are as follows:

* **Meantime to failure recovery -**This is the average time taken to recover from a failure.
* **Deployment frequency -** The frequency in which the deployment occurs.
* **Percentage of failed deployments -** The number of times the deployment fails.

### **Explain the term "Infrastructure as Code" (IaC) as it relates to configuration management.**

* Administering cloud computing environments, also known as infrastructure as a service (IaaS).
* Writing code to manage configuration, deployment, and automatic provisioning