TABLE OF CONTENTS

INTRODUCTION

Scope:

Provides end to end continuous Integration and continuous deployment by using DevOps Tools.

Application defines student to provide entire information and which stores in DB for better analysis of data.

Provides security using microservice architecture by queueing the load in the web server queue when the server is bombarded with load.

Objective:

Deploying application with microservices and bring the application to live using cloud infrastructure

Jenkins CI CD tool for end-to-end automation and any build failures will be detect at the starting stages

Assigning the DNS to the Load Balancer IP with the Freenom subscription.

For https security to the domain created open ssl certificates to make the domain https.

Analysing the logs at the Load balancer level, web server level and app server level for better understanding the application flow.

ABSTRACT

The main aim of our project is to build and deploy application in microservice architecture. Our main mottto of this project is to develop an application with full end-to-end automation using ci/cd build process and deploying war file into tomcat servers using DevOps tools.

DevOps is a conceptual framework for reintegrating and development and operations of informations systems. This allows a single team to handle the entire application lifecycle, from development to **testing, deployment**, and **operations**. DevOps helps you to reduce the disconnection between software developers, quality assurance (QA) engineers, and system administrators.

As the project based on automation, it shows less errors and reduces time compared to manual process.

SYSTEM ANALYSIS

EXISTING SYSTEM

- The current system of the project is manually deploying the builds to DEV, QA, PROD environments.
- Because of manual deployments there is a chance for human errors.
- The manual process will associated with a lot of problems such as missing student details in our student registration form, Time wastage etc.

PROPOSED SYSTEM

- The proposed system is end-to-end automation with DevOps tools.
- This system makes the user to work with agile methodology more efficiently.
- It also reduces the work of developers to troubleshoot the deployment issues and focus on the code.
- The main aim of this project is to build and deploy application in microservice architecture with automatic CI/CD build process.

SYSTEM REQUIREMENTS

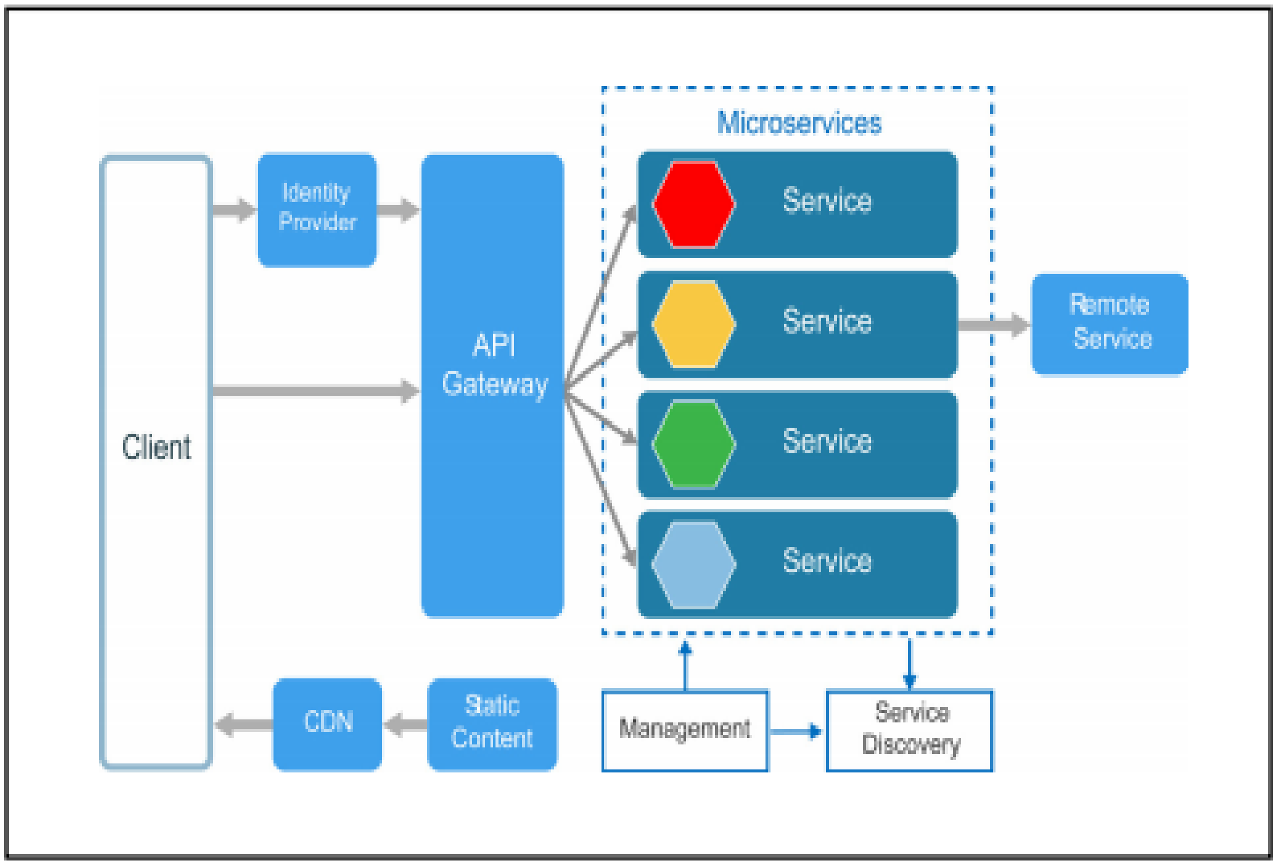
HARDWARE REQUIREMENTS:

• PROCESSOR	: 2VCU'S
• RAM	: 4GB
• HDD	: 10GB

SYSTEM REQUIREMENTS:

• OPERATING SYSTEM	: LINUX/WINDOWS
• FRONTEND CODE	: HTML
• BACKEND CODE	: JAVA
• SERVER	: APACHE TOMCAT 8
• DATABASE	: MARIADB

MICROSERVICE ARCHITECTURE



Microservices:

Microservices architecture is an approach in which a single application is composed of many loosely coupled and independently deployable smaller services. These services typically

- have their own technology stack, inclusive of the database and data management model;
- communicate with one another over a combination of REST APIs, event streaming, and message brokers; and
- are organized by business capability, with the line separating services often referred to as a bounded context.

- Code can be updated more easily - new features or functionality can be added without touching the entire application
- Teams can use different stacks and different programming languages for different components.
- Components can be scaled independently of one another, reducing the waste and cost associated with having to scale entire applications because a single feature might be facing too much load.

Microservice is a service-based application development methodology. In this methodology, big applications will be divided into smallest independent service units. Microservice is the process of implementing Service-oriented Architecture (SOA) by dividing the entire application as a collection of interconnected services, where each service will serve only one business need.

Microservice Over SOA

The following table lists certain features of SOA and Microservice, bringing out the importance of using microservice over SOA.

Service-oriented architecture (SOA) is an enterprise-wide approach to software development of application components that takes advantage of reusable software components, or services. In SOA software architecture, each service is comprised of the code and data integrations required to execute a specific business function — for example, checking a customer's credit, signing into a website or processing a mortgage application.

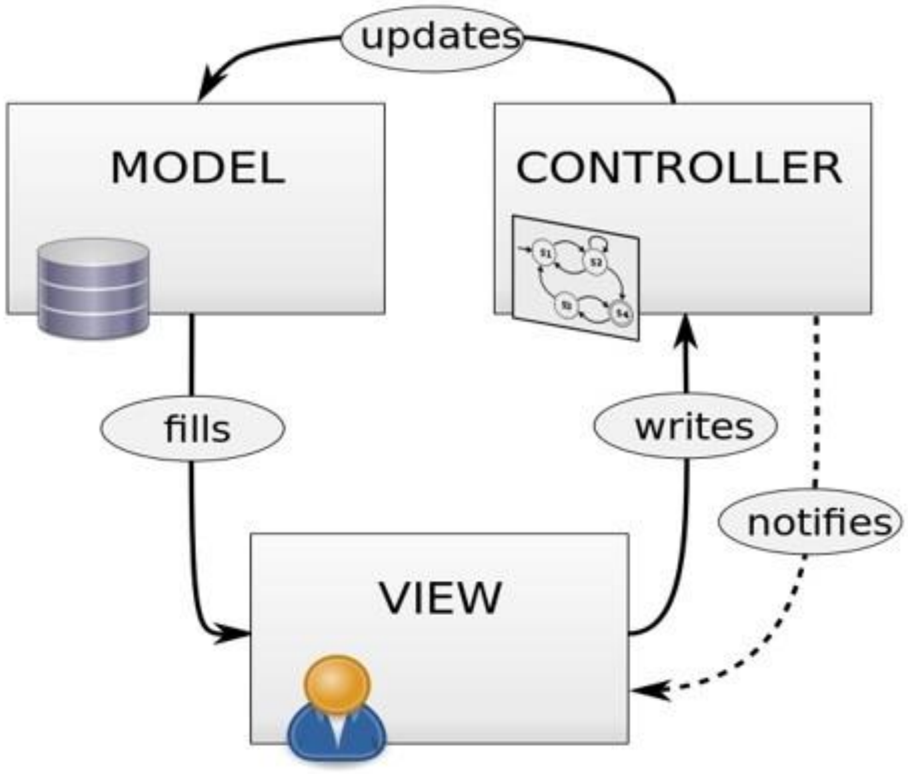
Component	SOA	Microservice
Design pattern	SOA is a design paradigm for computer software, where software components are exposed to the outer world for usage in the form of services.	Micro Service is a part of SOA. It is a specialized implementation of SOA.
Dependency	Business units are dependent on each other.	All business units are independent of each other.
Size	Software size is bigger than the conventional software.	Software size is small.
Technology	Technology stack is less than Microservice.	Microservice is heterogeneous in nature as exact technologies are used to perform a specific task. Microservices can be considered as a conglomerate of many technologies.
Autonomous and Focus	SOA applications are built to perform multiple business tasks.	Microservice applications are built to perform a single business task.
Nature	Monolithic in nature.	Full stack in nature.
Deployment	Deployment is time-consuming.	Deployment is very easy. Hence, it will be less time-consuming.
Cost-effectiveness	More cost-effective.	Less cost-effective.
Scalability	Less compared to Microservices.	Fully scaled.
Example	Let us consider one online CAB booking application. If we want to build that application using SOA, then its software units will be – ▫ GetPayments And DriverInformation And MappingDataAPI ▫ AuthenticateUsersAnd DriversAPI	If the same application is built using microservice architecture, then its APIs will be – ▫ SubmitPaymentsService ▫ GetDriverInfoService ▫ GetMappingDataService ▫ AuthenticateUserService ▫ AuthenticateDriverService

Scaling:

Scaling is a process of breaking down a software in different units. Scaling also defines in terms of scalability. Scalability is the potential to implement more advance features of the application. It helps to improve security, durability, and maintainability of the application. We have three types of scaling procedures that is followed in the industries. Following are the different scaling methodologies along with the corresponding real-life examples.

X-Axis Scaling:

X-axis scaling is also called as horizontal scaling. In this procedure, the entire application is sub-divided into different horizontal parts. Normally, any web server application can have this type of scaling. Consider a normal MVC architecture that follows horizontal scaling as shown in the following figure.



Y-Axis Scaling:

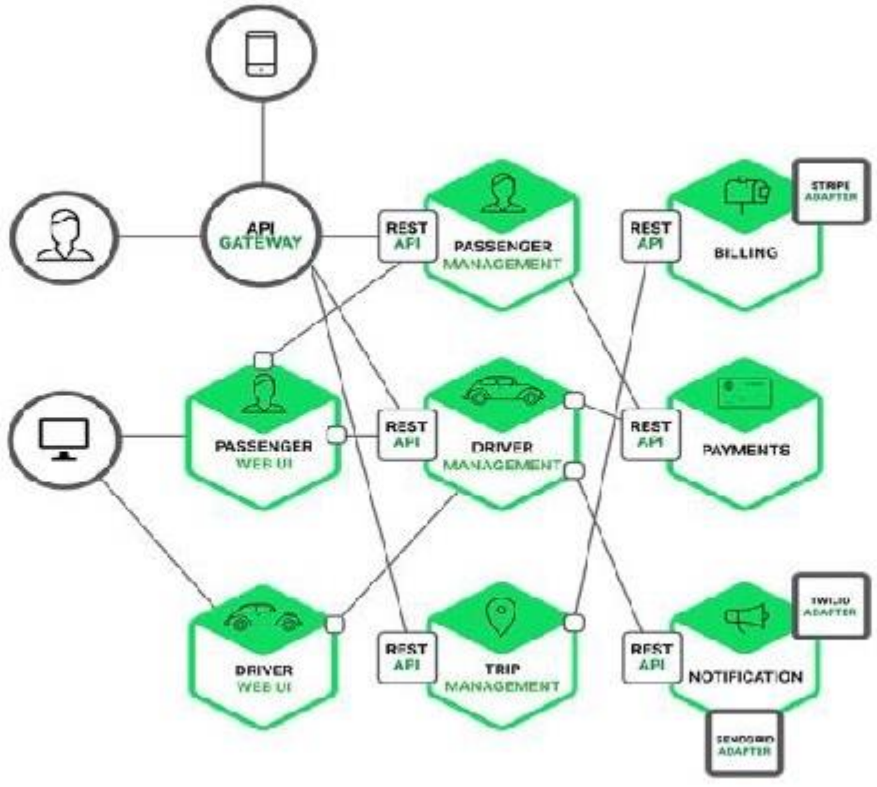
Y-axis scaling is also called as a vertical scaling that includes any resource level scaling. Any DBaaS or Hadoop system can be considered to be Y-axis scaled. In this type of scaling, the users request is redirected and restricted by implementing some logic.

Let us consider Facebook as an example. Facebook needs to handle 1.79 million users in every second; hence, controlling the traffic is a huge responsibility of Facebook network engineers. To overcome from any hazard, they follow Y-axis scaling which includes running multiple servers with the same application at the same time. Now in order to control this huge level of traffic, Facebook redirects all the traffic from one region to a specific server, as depicted in the image. This transferring of traffic based on the region is called load balancing in architectural language.

This method of breaking down resources into small independent business units is known as Y-Axis scaling.

Z-Axis Scaling:

X- and Y-axis scaling is pretty much easier to understand. However, one application can also be scaled at the business level, which is called as Z-axis scaling. Following is an example of scaling a cab service application in the different verticals of business units.



Advantages of Scaling:

- **Cost** – Proper scaling of a software will reduce the cost for maintenance.
- **Performance** – Due to loose coupling, the performance of a properly scaled software is always better than a non-scaled software.
- **Load distribution** – Using different technologies, we can easily maintain our server load.
- **Reuse** – Scalability of a software also increases the usability of the software.

How microservices works:

Microservices are a set of services that act together to make a whole application operate. This architecture utilizes APIs to pass information, such as user queries or a data stream, from one service to another.
How the underlying software works, or which hardware the service is built upon, depends solely on the team who built the service. This makes both communicating between teams and upgrading services very dynamic—even reactive—allowing a software company or team to be more resilient in its development.
Kubernetes has helped advance the cause of microservices, though it not a necessary building block. The rise of cloud computing and networked computers has done two things:

- Removed the responsibility from the user needing to have a powerful computer to run all the necessary operations.
- Places the responsibility on the company to use individual servers to run its service each time a user runs the application.

In the case of microservices, the user's machine may be responsible for basic processing, but it is mostly responsible for sending and receiving network calls to other computers.

Whenever you use an application, it's reasonable to assume that there are five other computers, give or take, that just turned on in order to power your experience. In the case of something like Facebook or Uber, it may be more reasonable to expect another 10,000 computers are actively processing information to enhance the user experience.

Microservices are often considered a logical evolution of Service Oriented Architecture (SOA), but there are clear differences between the two.

Examples of microservices architecture:

As Martin Fowler points out, many large companies now utilize microservices within their architecture. Netflix is one of the earlier, most well-known adopters. Some other well-known examples are:

- eBay
- Amazon
- Twitter
- PayPal
- SoundCloud
- Gilt
- The Guardian

Advantages of microservices:

The advantages of microservices seem strong enough to have convinced some big enterprise players such as Amazon, Netflix, and eBay to adopt the methodology. Compared to more monolithic design structures, microservices offer:

- **Improved fault isolation**: Larger applications can remain mostly unaffected by the failure of a single module.
- **Eliminate vendor or technology lock-in**: Microservices provide the flexibility to try out a new technology stack on an individual service as needed. There won't be as many dependency concerns and rolling back changes becomes much easier. With less code in play, there is more flexibility.
- **Ease of understanding:** With added simplicity, developers can better understand the functionality of a service.
- **Smaller and faster deployments**: Smaller codebases and scope = quicker deployments, which also allow you to start to explore the benefits of Continuous Deployment.

- **Scalability**: Since your services are separate, you can more easily scale the most needed ones at the appropriate times, as opposed to the whole application. When done correctly, this can impact cost savings.

Disadvantages of microservices:

Microservices may be a hot trend, but the architecture does have drawbacks. In general, the main negative of microservices is the complexity that any distributed system has.
Here's a list of some potential pain areas and other cons associated with microservices designs:

- **Communication between services is complex**: Since everything is now an independent service, you have to carefully handle requests traveling between your modules. In one such scenario, developers may be forced to write extra code to avoid disruption. Over time, complications will arise when remote calls experience latency.
- **More services equals more resources**: Multiple databases and transaction management can be painful.
- **Global testing is difficult**: Testing a microservices-based application can be cumbersome. In a monolithic approach, we would just need to launch our WAR on an application server and ensure its connectivity with the underlying database. With microservices, each dependent service needs to be confirmed before testing can occur.
- **Debugging problems can be harder**: Each service has its own set of logs to go through. Log, logs, and more logs.
- **Deployment challengers**: The product may need coordination among multiple services, which may not be as straightforward as deploying a WAR in a container.

- **Large vs small product companies**: Microservices are great for large companies, but can be slower to implement and too complicated for small companies who need to create and iterate quickly, and don’t want to get bogged down in complex orchestration.

Of course, with the right kind of automation and tools and the properly trained staff, all the above drawbacks can be addressed.

Deployment of microservices:

Now that we understand microservices, how are they deployed? The best way to deploy microservices-based applications is within containers, which are complete virtual operating system environments that provide processes with isolation and dedicated access to underlying hardware resources. One of the biggest names in container solutions right now is Docker, which you can learn more about in our Getting Started Course.

Virtual machines from infrastructure providers like Amazon Web Services (AWS) can also work well for microservices deployments, but relatively lightweight microservices packages may not leverage the whole virtual machine, potentially reducing their cost-effectiveness.

Code deployments can also be completed using an Open Service Gateway Initiative (OSGI) bundle. In this use case, all application services will be running under one Java virtual machine, but this method comes with a management and isolation tradeoff.

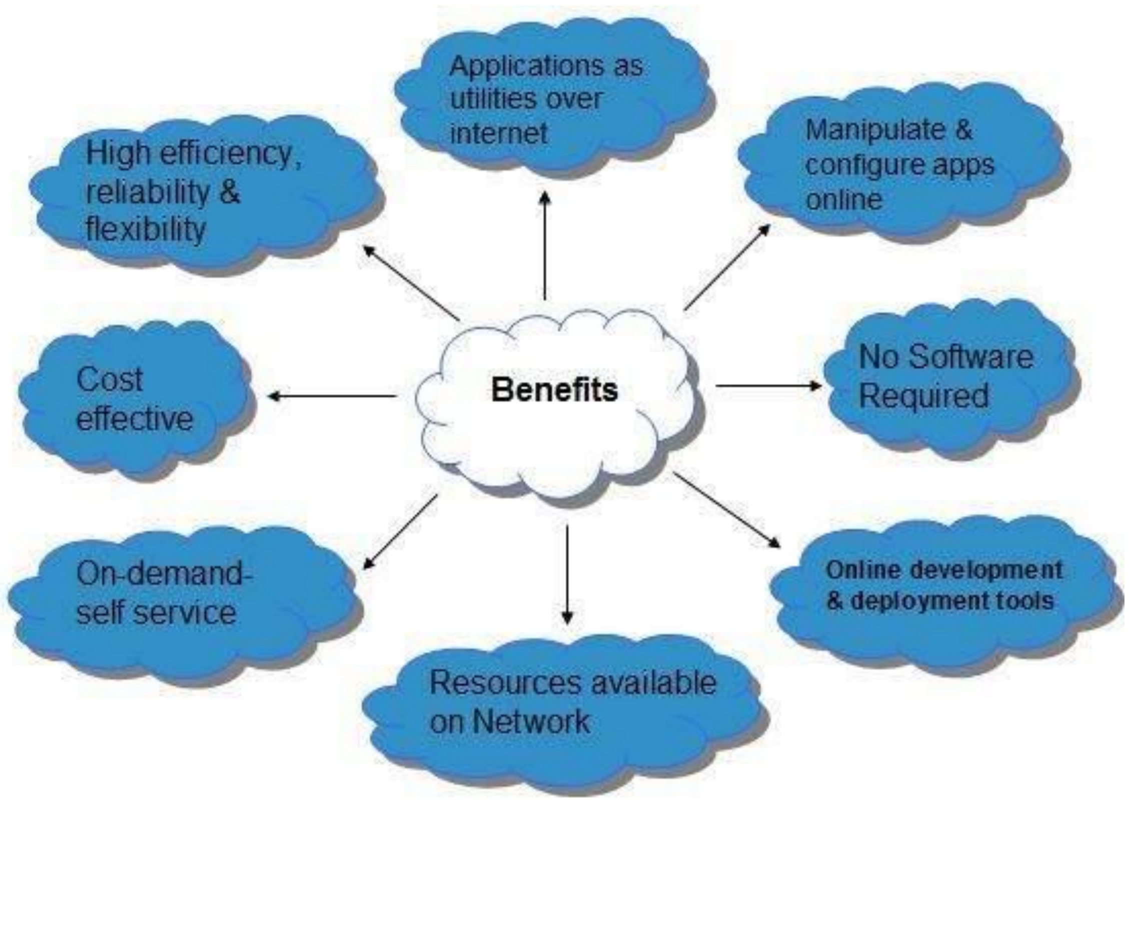
Cloud:

The term **Cloud** refers to a **Network** or **Internet.** In other words, we can say that Cloud is something, which is present at remote location. Cloud can provide services over public and private networks, i.e., WAN, LAN or VPN.

Cloud Computing provides us means of accessing the applications as utilities over the Internet. It allows us to create, configure, and customize the applications online.

Cloud Computing refers to **manipulating, configuring,** and **accessing** the hardware and software resources remotely. It offers online data storage, infrastructure, and application.

Benefits:


Benefits
Applications as utilities over internet
Manipulate & configure apps online
No Software Required
Online development & deployment tools
Resources available on Network
On-demand-self service
Cost effective
High efficiency, reliability & flexibility

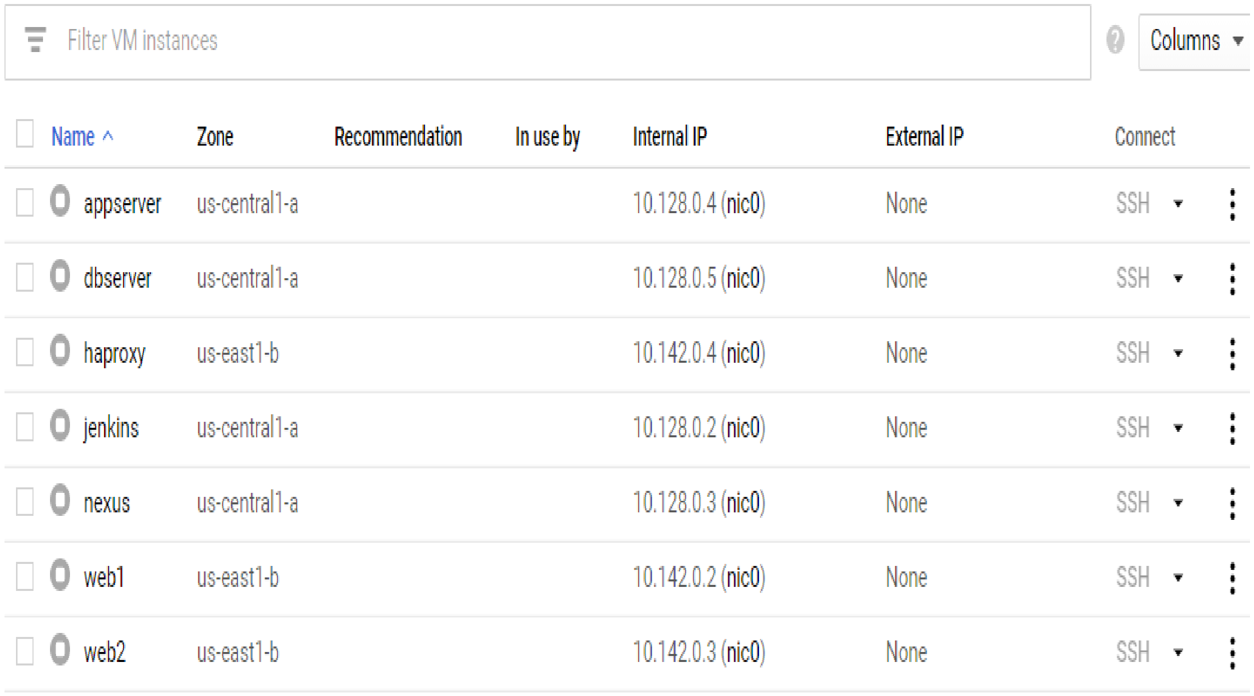
All though there are so many cloud based technologies, these 3 are widely used clouds. They are:

- AWS (Amazon Web Services)
- Microsoft Azure
- Google Cloud Platform (GCP)

Note:

In our project we have used Google Cloud Platform.

GCP servers:



Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
appserver	us-central1-a			10.128.0.4 (nic0)	None	SSH
dbserver	us-central1-a			10.128.0.5 (nic0)	None	SSH
haproxy	us-east1-b			10.142.0.4 (nic0)	None	SSH
jenkins	us-central1-a			10.128.0.2 (nic0)	None	SSH
nexus	us-central1-a			10.128.0.3 (nic0)	None	SSH
web1	us-east1-b			10.142.0.2 (nic0)	None	SSH
web2	us-east1-b			10.142.0.3 (nic0)	None	SSH

TOOLS

GIT

Git is an **open-source distributed version control system**. It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

Git was created by **Linus Torvalds** in **2005** to develop Linux Kernel. It is also used as an important distributed version-control tool for **the DevOps**.

Git is easy to learn, and has fast performance. It is superior to other **Source Code Management** tools like Subversion, CVS, Perforce, and ClearCase.

Git is foundation of many services like **GitHub** and **GitLab**, but we can use Git without using any other Git services. Git can be used **privately** and **publicly**.

GITHUB

GitHub is a Git repository hosting service. GitHub also facilitates with many of its features, such as access control and collaboration. It provides a Web-based graphical interface.

GitHub is an American company. It hosts source code of your project in the form of different programming languages and keeps track of the various changes made by programmers.

It offers both **distributed version control and source code management (SCM)** functionality of Git. It also facilitates with some collaboration features such as bug tracking, feature requests, task management for every project.

Benefits of GitHub:

GitHub service includes access controls as well as collaboration features like task management, repository hosting, and team management.

The key benefits of GitHub are as follows.

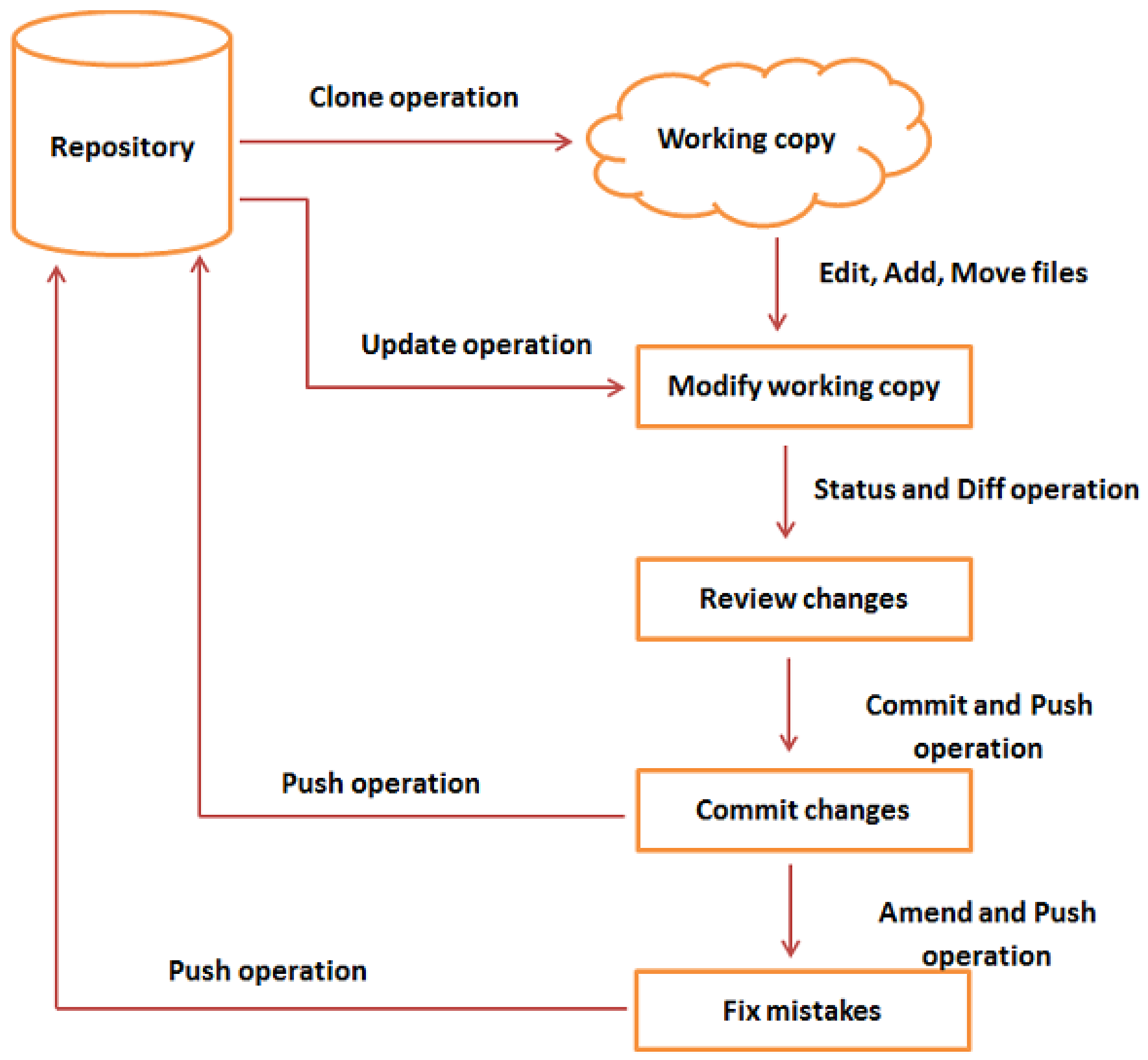
- It is easy to contribute to open-source projects via GitHub.
- It helps to create an excellent document.
- It allows your work to get out there in front of the public.
- You can track changes in your code across versions.

Git - Life Cycle:

General workflow is as follows −

- You clone the Git repository as a working copy.
- You modify the working copy by adding/editing files.
- If necessary, you also update the working copy by taking other developer's changes.
- You review the changes before commit.
- You commit changes. If everything is fine, then you push the changes to the repository.
- After committing, if you realize something is wrong, then you correct the last commit and push the changes to the repository.

Shown below is the pictorial representation of the work-flow.



Git-Commands:

Git Pull:

The term pull is used to receive data from GitHub. It fetches and merges changes from the remote server to your working directory. The **git pull command** is used to pull a repository.

Syntax: git pull **<**option**>** [**<**repository URL**><**refspec**>**...]

Git Push:

The "git push" command is used to push into the repository. The push command can be considered as a tool to transfer commits between local and remote repositories.

If we do not specify the location of a repository, then it will push to default location at **origin master**.

The basic syntax is given below:

git push<option>[<Remote URL><branch name><refspec>]

Syntax: git push origin master

Git Fetch:

The "**git fetch**" **command** is used to pull the updates from remote-tracking branches.

Syntax: git fetch< repository URL>

Git Init:

The git init command is used to create a new blank repository. It is used to make an existing project as a Git project. Several Git commands run inside the repository, but init command can be run outside of the repository.

Syntax: git init

Git Add:

The git add command is used to add file contents to the Index (Staging Area).

Syntax: git add <File name>

Git Commit:

It is used to record the changes in the repository. It is the next command after the git add.

Syntax: git commit -am "Commit message."

Git Clone:

The **git clone** is a command-line utility which is used to make a local copy of a remote repository. It accesses the repository through a remote URL.

Syntax: git clone **<**repository URL**>**

Git Status:

The git status command is used to display the state of the repository and staging area.

Syntax: git status

Git log:

Git log is a utility tool to review and read a history of everything that happens to a repository.

Syntax: git log

Git Reset:

The term reset stands for undoing changes. The git reset command is used to reset the changes.

Syntax:

git reset --hard

MAVEN

Maven is a project management and comprehension tool that provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.

In case of multiple development teams environment, Maven can set-up the way to work as per standards in a very short time. As most of the project setups are simple and reusable, Maven makes life of developer easy while creating reports, checks, build and testing automation setups.

Maven provides developers ways to manage the following −

- Builds
- Documentation
- Reporting
- Dependencies
- SCMs
- Releases
- Distribution
- Mailing list

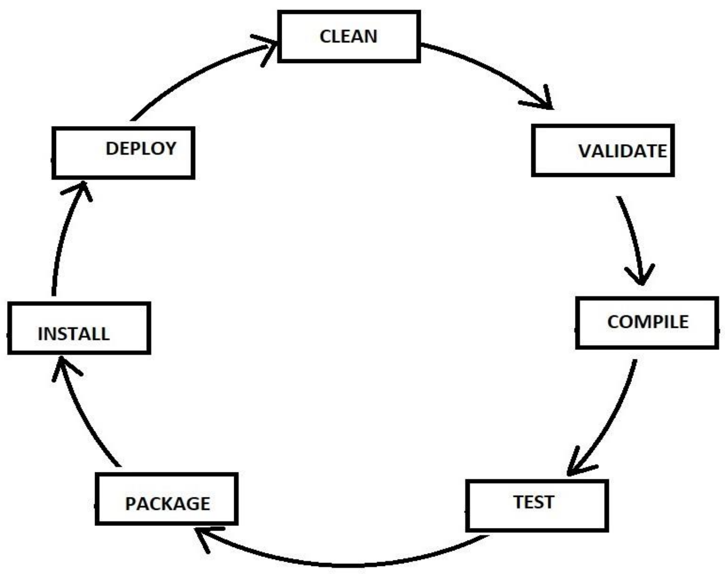
Objective:

The primary goal of Maven is to provide developer with the following −

- A comprehensive model for projects, which is reusable, maintainable, and easier to comprehend.
- Plugins or tools that interact with this declarative model.

Maven project structure and contents are declared in an xml file, pom.xml, referred as Project Object Model (POM), which is the fundamental unit of the entire Maven system.

Maven - Build Life Cycle:



MAVEN COMMANDS:

CLEAN:

Clears the target directory into which maven normally builds your project.

Syntax: mvn clean

Validate:

Validates if the project is correct and if all necessary information is available.

Syntax: mvn validate

Compile:

Source code compilation is done in this phase.

Syntax: mvn compile

Test:

Tests the compiled source code suitable for testing framework.

Syntax: mvn test

Package:

This phase creates the JAR/WAR package as mentioned in the packaging in POM.xml.

Syntax: mvn package

Install:

This phase installs the package in local/remote maven repository.

Syntax: mvn install

Deploy: Copies the final package to the remote repository.

Syntax: mvn deploy

Maven pom.xml file:

POM stands for Project Object Model. It is fundamental unit of work in Maven. It is an XML file that resides in the base directory of the project as pom.xml.

The POM contains information about the project and various configuration detail used by Maven to build the project(s).

POM also contains the goals and plugins. While executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, and then executes the goal. Some of the configuration that can be specified in the POM are following −project dependencies, plugins, goals, build profiles, project version, developers, mailing list.

POM.XML FILE:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.jdevs</groupId>
  <artifactId>studentapp</artifactId>
  <version>2.5-SNAPSHOT</version>
  <packaging>war</packaging>
  <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.3</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
      <plugin>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.6</version>
        <configuration>

<warSourceDirectory>WebContent</warSourceDirectory>
          <failOnMissingWebXml>false</failOnMissingWebXml>
        </configuration>
      </plugin></plugins></build>
   <distributionManagement>
        <repository>
            <id>deployment</id>
            <name>Internal Releases</name>
            <url>http://10.128.0.10:8081/repository/student-
rel/</url>
        </repository>
        <snapshotRepository>
            <id>deployment</id>
            <name>Internal Snapshot Releases</name>
            <url>http://10.128.0.10:8081/repository/student-
snap/</url>
        </snapshotRepository>
    </distributionManagement>
  <dependencies>
       <dependency>
              <groupId>javax.servlet</groupId>
              <artifactId>javax.servlet-api</artifactId>
              <version>3.0.1</version>
       </dependency>
  </dependencies>
</project>
```

JENKINS

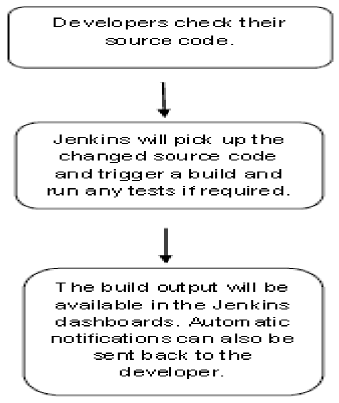
Jenkins is an open source automation tool written in Java programming language that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment technologies.

Jenkins **builds** and **tests** our software projects which continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.It also allows us to continuously **deliver** our software by integrating with a large number of testing and deployment technologies.

Jenkins achieves CI (Continuous Integration) with the help of plugins. Plugins is used to allow the integration of various DevOps stages. If you want to integrate a particular tool, you have to install the plugins for that tool. For example: Maven 2 Project, Git, HTML Publisher, Amazon EC2, etc.

For example: If any organization is developing a project, then **Jenkins** will continuously test your project builds and show you the errors in early stages of your development.

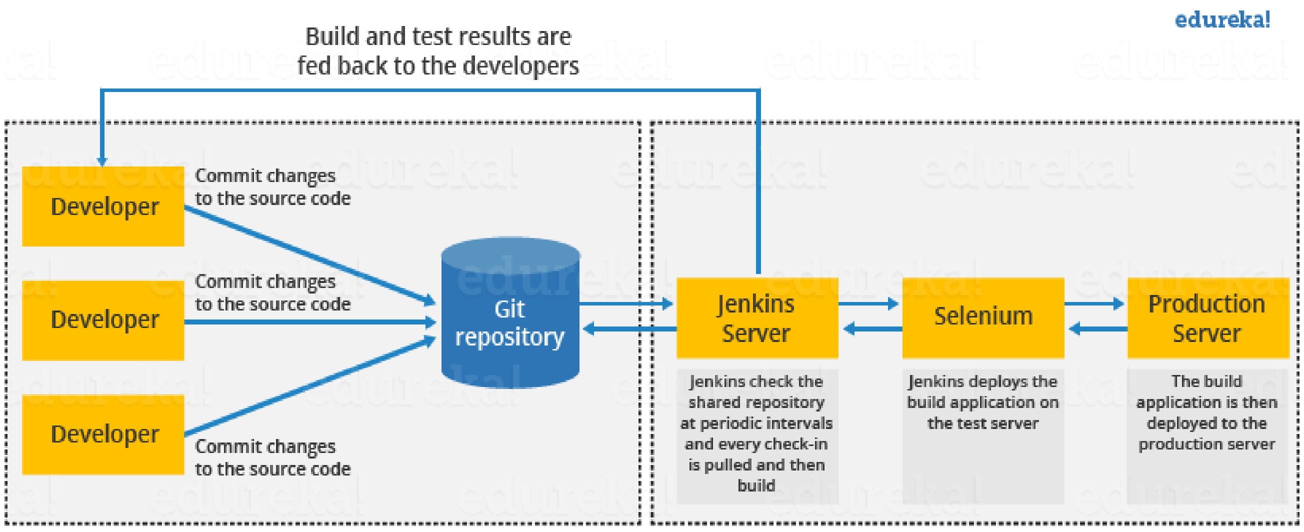
Work Flow:


Developers check their source code.
Jenkins will pick up the changed source code and trigger a build and run any tests if required.
The build output will be available in the Jenkins dashboards. Automatic notifications can also be sent back to the developer.

Continuous Integration:

Continuous Integration *(CI)* is a development practice in which the developers are needs to commit changes to the source code in a shared repository at regular intervals. Every commit made in the repository is then built. This allows the development teams to detect the problems early. Continuous integration requires the developers to have regular builds. The general practice is that whenever a code commit occurs, a build should be triggered.

Generic flow diagram of Continuous Integration with Jenkins:

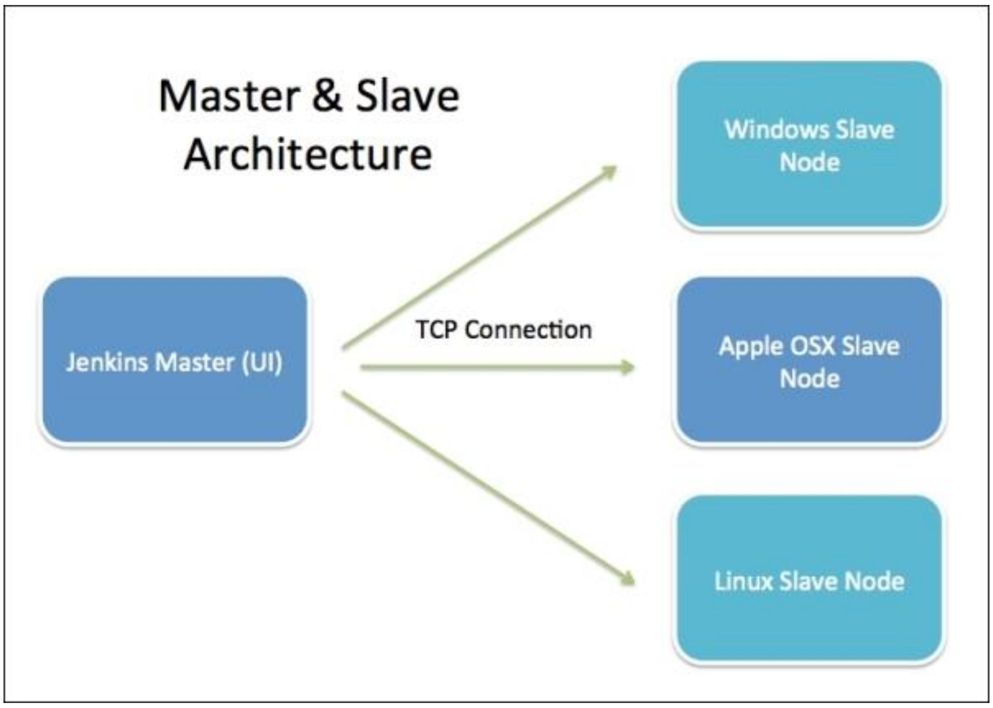

edureka!
Build and test results are fed back to the developers
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Developer
Commit changes to the source code
Git repository
Jenkins Server
Jenkins check the shared repository at periodic intervals and every check-in is pulled and then build
Selenium
Jenkins deploys the build application on the test server
Production Server
The build application is then deployed to the production server

Jenkins Architecture:

Jenkins follows Master-Slave architecture to manage distributed builds. In this architecture, slave and master communicate through TCP/IP protocol.

Jenkins architecture has two components:

- Jenkins Master/Server
- Jenkins Slave/Node/Build Server



Jenkins Master:

The main server of Jenkins is the Jenkins Master. It is a web dashboard which is nothing but powered from a war file. By default it runs on 8080 port. With the help of Dashboard, we can configure the jobs/projects but the build takes place in Nodes/Slave. By default one node (slave) is configured and running in Jenkins server. We can add more nodes using IP address, user name and password using the ssh, jnlp or webstart methods.

Jenkins Slave:

Jenkins slave is used to execute the build jobs dispatched by the master. We can configure a project to always run on a particular slave machine, or particular type of slave machine, or simple let the Jenkins to pick the next available slave/node.

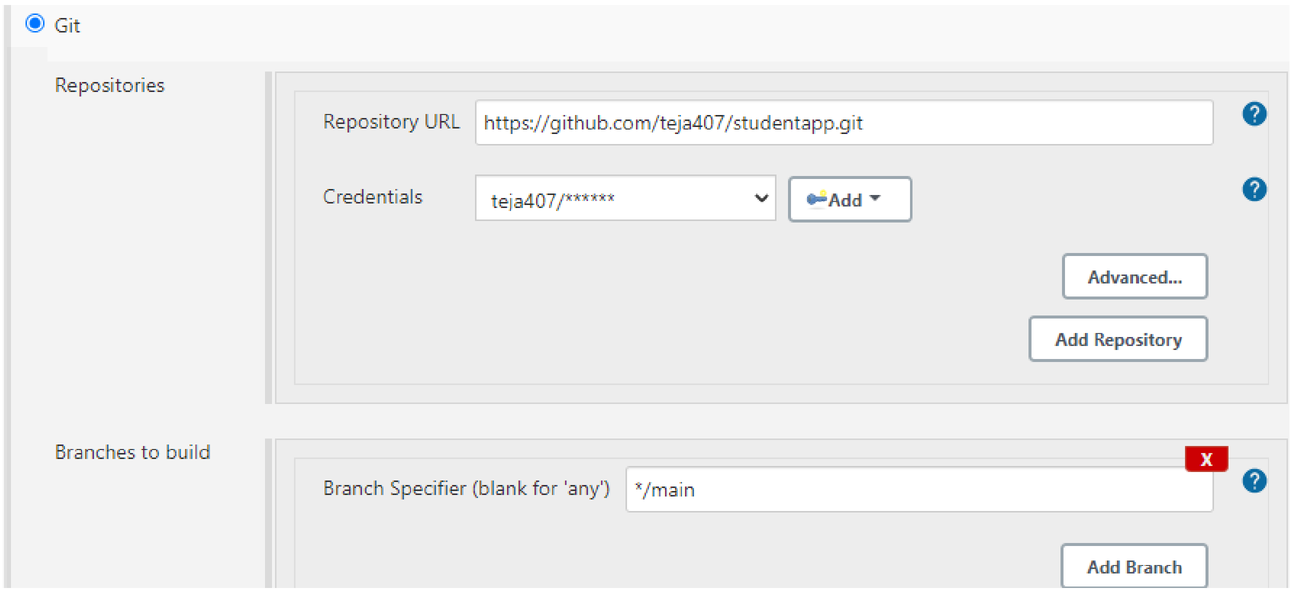
As we know Jenkins is developed using Java is platform independent thus Jenkins Master/Servers and Slave/nodes can be configured in any servers including Linux, Windows, and Mac.

GitHub Setup for Jenkins:

Jenkins is a CI (Continuous Integration) server and this means that it needs to check out source code from a source code repository and build code. Jenkins has outstanding support for various source code management systems like Subversion, CVS etc. To do the GitHub setup, first we need to install "Git Plugin".

Integrating Jenkins with GitHub:

- First create a new job in Jenkins, open the Jenkins Dashboard and click on "create new jobs".
- Now enter the item name and select the job type. For example, item name is **"test**" and job type is "**Freestyle project**". Click on **OK**.
- Once you click OK, the page will be redirected to its project configuration.
- Now, under the "Source Code Management" you will see the Git option, if your **Git** plugin has been installed in Jenkins.
- Enter the Git repository URL on the "Repository URL" option to pull the code from GitHub.


Git

Repositories

Repository URL: https://github.com/teja407/studentapp.git

Credentials: teja407/******

Branches to build

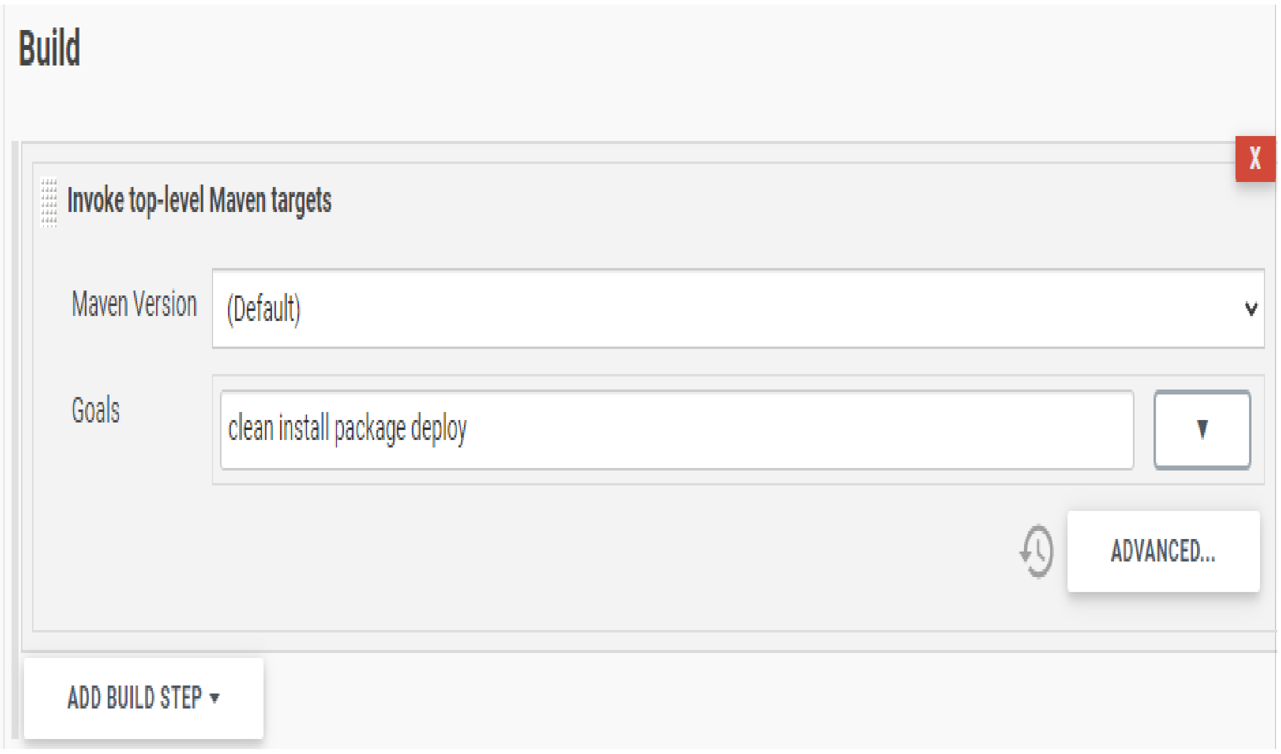
Branch Specifier (blank for 'any'): */main

Maven Setup for Jenkins:

To do the Maven setup, first we need to install "Maven Plugin".

Integrating Jenkins with Maven:

- Open already created “Test” project, then click on configure option.
- Now, below the "Source Code Management" you will see the Build option, In the Build Triggers section, there are multiple options, select the Maven one, If your Maven plugin has installed.


Build

Invoke top-level Maven targets

Maven Version: (Default)

Goals: clean install package deploy

KUBERNETES

Kubernetes – Node:

A node is a working machine in Kubernetes cluster which is also known as a minion. They are working units which can be physical, VM, or a cloud instance.

Kubernetes – Pod:

A pod is a collection of containers and its storage inside a node of a Kubernetes cluster. It is possible to create a pod with multiple containers inside it. For example, keeping a database container and data container in the same pod.

Types of Pod:

There are two types of Pods –

- Single container pod
- Multi container pod

Single Container Pod:

They can be simply created with the kubctl run command, where you have a defined image on the Docker registry which we will pull while creating a pod.

Multi Container Pod:

Multi container pods are created using **yaml mail** with the definition of the containers.

Kubernetes - Replica Sets:

Replica Set ensures how many replica of pod should be running. It can be considered as a replacement of replication controller. The key difference between the replica set and the replication controller is, the replication controller only supports equality-based selector whereas the replica set supports set-based selector.

Kubernetes - Replication Controller:

Replication Controller is one of the key features of Kubernetes, which is responsible for managing the pod lifecycle. It is responsible for making sure that the specified number of pod replicas are running at any point of time. It is used in time when one wants to make sure that the specified number of pod or at least one pod is running. It has the capability to bring up or down the specified no of pod.

Kubernetes – API:
Kubectl command-line tool can be used to create, update, delete, and get API object. Kubernetes API acts a communicator among different components of Kubernetes.

Kubernetes - App Deployment:
Deployment is a method of converting images to containers and then allocating those images to pods in the Kubernetes cluster. This also helps in setting up the application cluster which includes deployment of service, pod, replication controller and replica set. The cluster can be set up in such a way that the applications deployed on the pod can communicate with each other.

Kubernetes – Autoscaling:
Autoscaling is one of the key features in Kubernetes cluster. It is a feature in which the cluster is capable of increasing the number of nodes as the demand for service response increases and decrease the number of nodes as the requirement decreases.

Kubernetes – Kubectl:
Kubectl is the command line utility to interact with Kubernetes API. It is an interface which is used to communicate and manage pods in Kubernetes cluster.

Kubectl Commands:

kubectl cluster-info − It displays the cluster Info.

Syntax: kubectl cluster-info

kubectl config − Modifies the kubeconfig file.

Syntax:
```
kubectl config <SUBCOMMAD>
kubectl config –-kubeconfig <String of File name>
```

kubectl config delete-cluster – Deletes the specified cluster from kubeconfig.

Syntax: kubectl config delete-cluster <Cluster Name>

kubectl config get-clusters – Displays cluster defined in the kubeconfig.

Syntax: kubectl config get-cluster
kubectl config get-cluster <Cluser Name>

kubectl cp – Copy files and directories to and from containers.

Syntax: kubectl cp <Files from source> <Files to Destinatiion>

kubectl create – To create resource by filename of or stdin. To do this, JSON or YAML formats are accepted.

Syntax: kubectl create –f <File Name>

kubectl delete – Deletes resources by file name, stdin, resource and names.

Syntax: kubectl delete –f ([-f FILENAME] | TYPE [(NAME | -l label | --all)])

Kubectl describe – Describes any particular resource in kubernetes. Shows details of resource or a group of resources.

Syntax: kubectl describe <type> <type name>

kubectl edit – It is used to end the resources on the server. This allows to directly edit a resource which one can receive via the command line tool.

Syntax: kubectl edit <Resource/Name | File Name)

kubectl exec – This helps to execute a command in the container.

Syntax: kubectl exec POD <-c CONTAINER > -- COMMAND < args...>

Docker

Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere. Docker is a platform for packaging, deploying, and running applications. Docker applications run in containers that can be used on any system: a developer's laptop, systems on premises, or in the cloud.

Containerization is a technology that's been around for a long time, but it's seen new life with Docker. It packages applications as images that contain everything needed to run them: code, runtime environment, libraries, and configuration. Images run in containers, which are discrete processes that take up only as many resources as any other executable.

Features of Docker:

- Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
- With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
- You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
- Since Docker containers are pretty lightweight, they are very easily scalable.

Docker – Images:

In Docker, everything is based on Images. An image is a combination of a file system and parameters. Let's take an example of the following command in Docker.

```
docker run hello-world
```

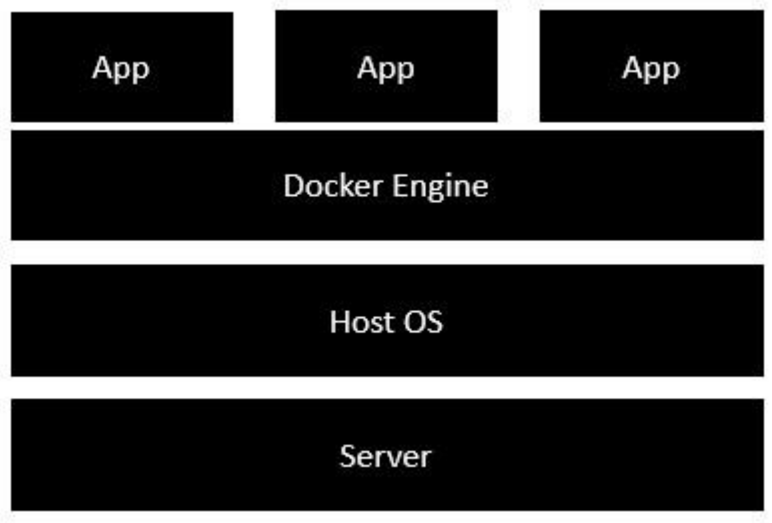
- The Docker command is specific and tells the Docker program on the Operating System that something needs to be done.

- Finally, "hello-world" represents the image from which the container is made.

Docker – Containers:

Containers are instances of Docker images that can be run using the Docker run command. The basic purpose of Docker is to run containers. The **run** command is used to mention that we want to create an instance of an image, which is then called a **container**.

Docker – Architecture:



Docker – Cloud:

The Docker Cloud is a service provided by Docker in which you can carry out the following operations −

- **Nodes** − You can connect the Docker Cloud to your existing cloud providers such as Azure and AWS to spin up containers on these environments.
- **Cloud Repository** − Provides a place where you can store your own repositories.
- **Continuous Integration** − Connect with **Github** and build a continuous integration pipeline.
- **Application Deployment** − Deploy and scale infrastructure and containers.
- **Continuous Deployment** −Can automate deployments.

CODING

HOME.JSP

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Display</title>
<style>
table#nat{
	width: 50%;
	background-color: #c48ec5;
}
</style>
</head>
<body>
<% String name =  request.getParameter("fullname");
	String Addr = request.getParameter("address");
	String age = request.getParameter("age");
	String Qual = request.getParameter("qual");
	String Persent = request.getParameter("percent");
	String Year = request.getParameter("yop"); %>
<table id ="nat">
<tr>
<td>Full Name</td>
```

```
<td><%= name %></td>
</tr>
<tr>
      <td>Address</td>
      <td><%= Addr %></td>
</tr>
<tr>
      <td>Age</td>
      <td><%= age %></td>
</tr>
<tr>
      <td>Qualification</td>
      <td><%= Qual %></td>
</tr>
<tr>
      <td>Percentage</td>
      <td><%= Percent %></td>
</tr>
<tr>
      <td>Year of Passout</td>
      <td><%= Year %></td>
</tr>
</table>
</body>
</html>
```

INDEX.JSP

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>User Data</title>
</head>
<style>
div.ex {
	text-align: right width:300px;
	padding: 10px;
	border: 5px solid grey;
	margin: 0px
}
</style><body>
	<h1>Student Registration Form</h1>
	<div class="ex">
	<form action="registrationController" method="post">
			<table style="with: 50%"><tr>
			<td>Student Id</td>
			<td><input type="text" name="id"/></td>
			</tr>
		<tr><td>Student Full Name</td>
```

```
            <td><input type="text” name="fullname"/></td>
            </tr>
            <tr>
              <td>Student Permanent Address</td>
              <td><input type="text" name="address"/></td>
            </tr>
            <tr>
               <td>Student Age</td>
               <td><input type="text" name="age"/></td>
            </tr>
            <tr>
               <td>Student Qualification</td>
               <td><input type="text" name="qual"/></td>
            </tr>
            <tr>
               <td>Student Percentage</td>
                <td><input type="text" name="percent"/></td>
            </tr>
            <tr>
               <td>Year Passed</td>
                <td><input type="text" name="yop"/></td>
            </tr>
/table>
            <input type="submit" value="register"/>
            </form></div>
</body></html>
```

NEWFILE.JSP

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
</body>
</html>
```

WEB.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
id="WebApp_ID" version="3.0">
  <display-name>CustomWebApp</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
 </web-app>
```

StudentDAO.java

```
package com.srk.dao;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.sql.DataSource;
import vo.Student;
public class StudentDAO {
        /*public static Connection getConnection(){
         Connection con=null;
         try{
             Class.forName("com.mysql.jdbc.Driver");
con=DriverManager.getConnection("jdbc:mysql://localhost:330
6/test","root","root");
         }catch(Exception e){System.out.println(e);}
         return con;
    }*/
        public static Connection getConnection() throws Exception
{
              Connection conn=null;
              Context initContext = new InitialContext()
             Context envContext = (Context)
initContext.lookup("java:comp/env");
DataSource ds = (DataSource)
envContext.lookup("jdbc/TestDB");
              conn = ds.getConnection();
              return conn;
        }
        public static void main(String[] args) {


        }
    public static int saveStudent(Student std){
         int status=0;
         try{
             Connection con=StudentDAO.getConnection();
             PreparedStatement ps=con.prepareStatement("insert
into
Students(student_name,student_addr,student_age,student_qu
al,student_percent,student_year_passed) values
(?,?,?,?,?,?)");
             ps.setString(1,std.getStudentName());
             ps.setString(2,std.getStudentAddr());
             ps.setString(3,std.getAge());
             ps.setString(4,std.getQualification());
             ps.setString(5,std.getPercentage());
             ps.setString(6,std.getYearPassed());
             status=ps.executeUpdate();
             con.close();
         }catch(Exception ex){ex.printStackTrace();}
         return status;
    }
```

```
    public static int updateStudent(Student std){
        int status=0;
        try{
            Connection con=StudentDAO.getConnection();
            PreparedStatement ps=con.prepareStatement("update
Students set
student_name=?,student_addr=?,student_age=?,student_qual
=?,student_percent=?,student_year_passed=? where
student_id=?");
            ps.setString(1,std.getStudentName());
            ps.setString(2,std.getStudentAddr());
            ps.setString(3,std.getAge());
            ps.setString(4,std.getQualification());
            ps.setString(5,std.getPercentage());
            ps.setString(6,std.getYearPassed());
            ps.setInt(7, std.getStudentId());
            status=ps.executeUpdate();
            con.close();
        }catch(Exception ex){ex.printStackTrace();}
        return status;
    }
    public static int deleteStudent(int stdId){
        int status=0;
        try{
            Connection con=StudentDAO.getConnection();
            PreparedStatement ps=con.prepareStatement("delete
from Students where student_id=?");
            ps.setInt(1,stdId);
            status=ps.executeUpdate();
            con.close();
        }catch(Exception e){e.printStackTrace();}
        return status;
    }
    public static Student getStudentById(int StdId){
       Student student=new Student();
        try{
            Connection con=StudentDAO.getConnection();
            PreparedStatement ps=con.prepareStatement("select
* from Students where student_id=?");
            ps.setInt(1,StdId);
            ResultSet rs=ps.executeQuery();
            if(rs.next()){
              student.setStudentId(rs.getInt(1));
              student.setStudentName(rs.getString(2));
              student.setStudentAddr(rs.getString(3));
              student.setAge(rs.getString(4));
              student.setQualification(rs.getString(5));
              student.setPercentage(rs.getString(6));
              student.setYearPassed(rs.getString(7));
            }
            con.close();
        }catch(Exception ex){ex.printStackTrace();}
        return student;
    }
    public static List<Student> getAllStudents(){
        List<Student> students=new ArrayList<Student>();
        try{
            Connection con=StudentDAO.getConnection();
            PreparedStatement ps=con.prepareStatement("select
* from Students");
            ResultSet rs=ps.executeQuery();
            while(rs.next()){
              Student student=new Student();
              student.setStudentId(rs.getInt(1));
              student.setStudentName(rs.getString(2));
              student.setStudentAddr(rs.getString(3));
              student.setAge(rs.getString(4));
              student.setQualification(rs.getString(5));
              student.setPercentage(rs.getString(6));
              student.setYearPassed(rs.getString(7));
              students.add(student);
            }
            con.close();
        }catch(Exception e){e.printStackTrace();}
        return students;
    }
}
```

DeleteStudent.java

```
package com.srk.servlet;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.srk.dao.StudentDAO;
@WebServlet("/deleteStudent")
public class DeleteStudent extends HttpServlet {
	protected void doGet(HttpServletRequest request,
HttpServletResponse response)
			throws ServletException, IOException {
	String sid=request.getParameter("stdId");
	int id=Integer.parseInt(sid);
	StudentDAO.deleteStudent(id);
	response.sendRedirect("viewStudents");
  }
}
```

EditStudent.java

```
package com.srk.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.srk.dao.StudentDAO;
import vo.Student;
@WebServlet("/editStudent")
public class EditStudent extends HttpServlet{
	protected void doGet(HttpServletRequest request,
HttpServletResponse response)
			throws ServletException, IOException {
		response.setContentType("text/html");
		PrintWriter out=response.getWriter();
		out.println("<h1>Update Student</h1>");
		String sid=request.getParameter("stdId");
		int stdId=Integer.parseInt(sid);
	Student student=StudentDAO.getStudentById(stdId);
		out.print("<form action='editStudent2'
method='post'>");
		out.print("<table>");
```

```
            out.print("<tr><td></td><td><input type='hidden'
name='stdId'
value='"+student.getStudentId()+"'/></td></tr>");

            out.print("<tr><td>Full Name :</td><td><input
type='text' name='stdname'
value='"+student.getStudentName()+"'/></td></tr>");

            out.print("<tr><td>Address :</td><td><input
type='text' name='stdaddrs'
value='"+student.getStudentAddr()+"'/></td></tr>");

            out.print("<tr><td>Age :</td><td><input
type='text' name='stdage'
value='"+student.getAge()+"'/></td></tr>");

            out.print("<tr><td>Qualification :</td><td><input
type='text' name='stdqual'
value='"+student.getQualification()+"'/></td></tr>");

            out.print("<tr><td>Percentage :</td><td><input
type='text' name='stdpercent'
value='"+student.getPercentage()+"'/></td></tr>");

            out.print("<tr><td>Year of Passout
:</td><td><input type='text' name='stdyearpass'
value='"+student.getYearPassed()+"'/></td></tr>");

            out.print("<tr><td colspan='2'><input
type='submit' value='Edit & Save '/></td></tr>");

            out.print("</table>");

            out.print("</form>");

            out.close();

        }

}
```

RegistrationController.java

```
package com.srk.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.srk.dao.StudentDAO;
import vo.Student;
@WebServlet("/registrationController")
public class RegistrationController extends HttpServlet {
	protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
		// TODO Auto-generated method stub
		response.setContentType("text/html");
		PrintWriter out = response.getWriter();
		String name = request.getParameter("fullname");
		String Addr = request.getParameter("address");
		String age = request.getParameter("age");
		String Qual = request.getParameter("qual");
		String Persent = request.getParameter("percent");
		String Year = request.getParameter("yop");
	if(name.isEmpty()||Addr.isEmpty()||age.isEmpty()||Qual.i
sEmpty()||Persent.isEmpty()||Year.isEmpty())
		{
			RequestDispatcher rd =
request.getRequestDispatcher("index.jsp");
			out.println("<font color=red>Please fill all the
fields</font>");
			rd.include(request, response);
		}
		else
		{
			Student student = new Student();
			student.setStudentName(name);
		student.setStudentAddr(Addr);
		student.setAge(age);
		student.setQualification(Qual);
		student.setPercentage(Persent);
		student.setYearPassed(Year);
			int status=StudentDAO.saveStudent(student);
		 if(status>0){
			response.sendRedirect("viewStudents");
		 }else{
		    out.println("Sorry! unable to save record");
		 }
		 out.close();
		}
	}}
```

SaveEditedStudent.java

```
package com.srk.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.srk.dao.StudentDAO;
import vo.Student;
@WebServlet("/editStudent2")
public class SaveEditedStudent extends HttpServlet{
        protected void doPost(HttpServletRequest request,
HttpServletResponse response)
                throws ServletException, IOException {
                response.setContentType("text/html");
                PrintWriter out=response.getWriter();
                String sid=request.getParameter("stdId");
                int studentId=Integer.parseInt(sid);
          String studentName = request.getParameter("stdname");
         String studentAddrs = request.getParameter("stdaddrs");
          String studentAge = request.getParameter("stdage");
        String studentQual = request.getParameter("stdqual");
 String studentPercent = request.getParameter("stdpercent");
String studentYearPass=request.getParameter("stdyearpass");
```

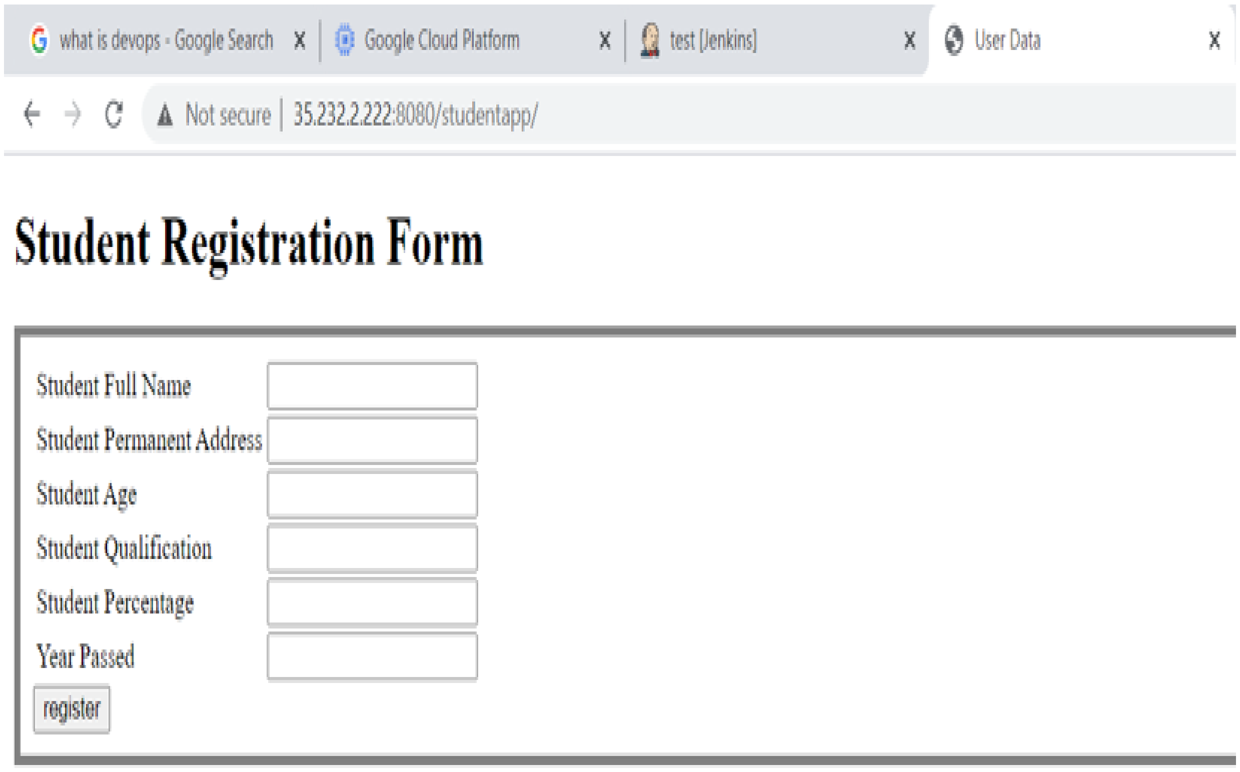
```
                Student  student = new Student();
                 student.setStudentId(studentId);
                 student.setStudentName(studentName);
                student.setStudentAddr(studentAddrs);
                student.setAge(studentAge);
                student.setQualification(studentQual);
                student.setPercentage(studentPercent);
                student.setYearPassed(studentYearPass);
                 int status=StudentDAO.updateStudent(student);
                 if(status>0){
                     response.sendRedirect("viewStudents");
                 }else{
                     out.println("Sorry! unable to update record");
                 }
                 out.close();
            }
    }
```

ViewStudents.java

```
package com.srk.servlet;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.srk.dao.StudentDAO;
import vo.Student;
@WebServlet("/viewStudents")
public class ViewStudents extends HttpServlet {
	protected void doGet(HttpServletRequest request,
HttpServletResponse response)
			throws ServletException, IOException {
	response.setContentType("text/html");
	PrintWriter out=response.getWriter();
	out.println("<a href='index.jsp'>Register Student</a>");
	out.println("<h1>Students List</h1>");
	List<Student> list=StudentDAO.getAllStudents();
	out.print("<table border='1' width='100%'");
	out.print("<tr><th>Student
ID</th><th>StudentName</th><th>Student
Addrs</th><th>Student Age</th><th>Student
Qualification</th><th>Student Percentage</th><th>Student
Year Passed</th><th>Edit</th><th>Delete</th></tr>
for(Student student : list){
out.print("<tr><td>"+student.getStudentId()+"</td><td>"+s
tudent.getStudentName()+"</td><td>"+student.getStudentA
ddr()+"</td><td>"+student.getAge()+"</td><td>"+student.
getQualification()+"</td><td>"+student.getPercentage()+"</
td><td>"+student.getYearPassed()+"</td><td><a
href='editStudent?stdId="+student.getStudentId()+"'>edit</a
></td><td><a
href='deleteStudent?stdId="+student.getStudentId()+"'>delet
e</a></td></tr>");

    }

    out.print("</table>");

    out.close();

 }

}
```

SCREENSHOTS

Student registration form



what is devops - Google Search | Google Cloud Platform | test [Jenkins] | User Data

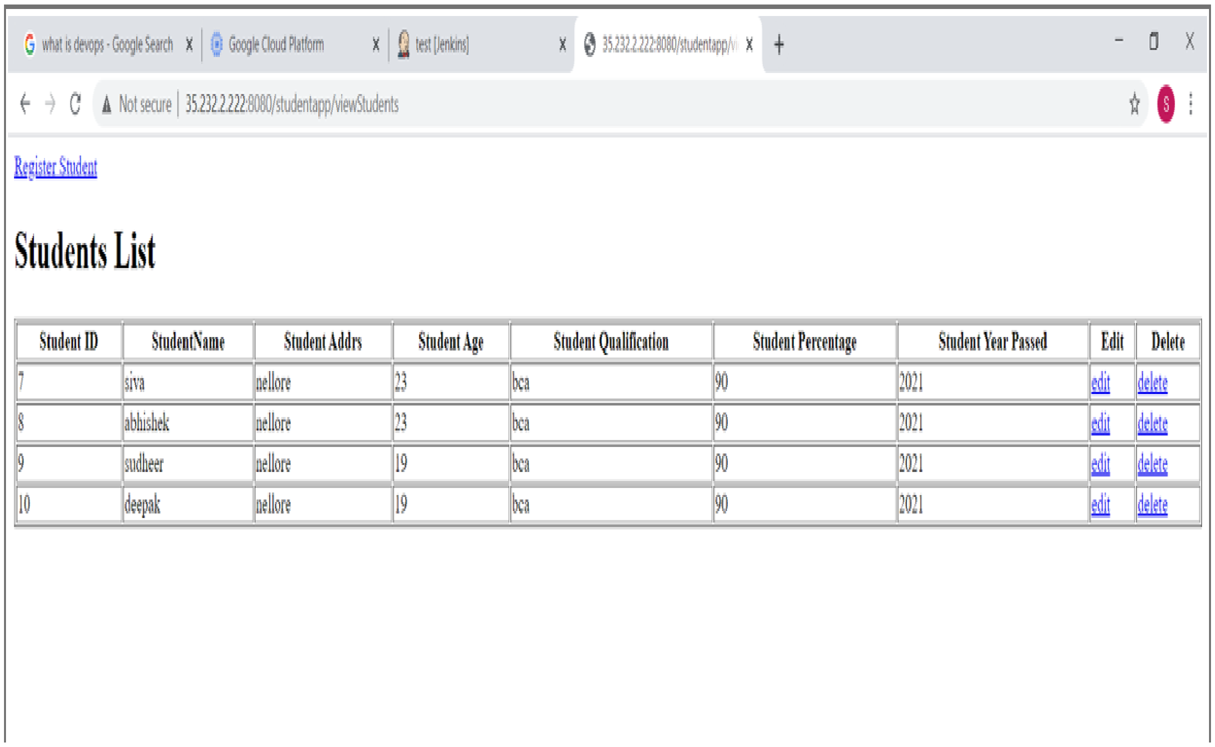
Not secure | 35.232.2.222:8080/studentapp/

Student Registration Form

Student Full Name	
Student Permanent Address	
Student Age	
Student Qualification	
Student Percentage	
Year Passed	

register

output



what is devops - Google Search | Google Cloud Platform | test [Jenkins] | 35.232.2.222:8080/studentapp/v

Not secure | 35.232.2.222:8080/studentapp/viewStudents

Register Student

Students List

Student ID	StudentName	Student Addrs	Student Age	Student Qualification	Student Percentage	Student Year Passed	Edit	Delete
7	siva	nellore	23	bca	90	2021	edit	delete
8	abhishek	nellore	23	bca	90	2021	edit	delete
9	sudheer	nellore	19	bca	90	2021	edit	delete
10	deepak	nellore	19	bca	90	2021	edit	delete

CONCLUSION

- DevOps is helping business in a tremendous way.
- It is bridging the gap between developers and operation team and resist change which create a smooth path for continuous development and continuous integration.
- The point is to start somewhere and make small, meaningfull changes. Measure the results and keep iterating. Take advantage of small tools and workflows that devs and ops are using.

BIBLOGRAPHY

- Javatpoint.com
- Tutorialspoint.com