

DevOps on cloud TCS 651

Unit-01

An introduction to Software Engineering, SDLC, Agile Framework, An introduction to DevOps, Gain insights of the DevOps environment, DevOps Vs Agile, DevOps Ecosystem.

Software Engineering

- Software engineering is an engineering approach for software development.
- We can alternatively view it as a systematic collection of past experience. The experience is arranged in the form of methodologies and guidelines.
- A small program can be written without using software engineering principles.
- But if one wants to develop a large software product, then software engineering principles are indispensable to achieve a good quality software cost effectively.
- Without using software engineering principles it would be difficult to develop large programs.
- In industry it is usually needed to develop large programs to accommodate multiple functions.
- A problem with developing such large commercial programs is that the complexity and difficulty levels of the programs increase exponentially with their sizes.
- For example, a program of size 1,000 lines of code has some complexity. But a program with 10,000 LOC is not just 10 times more difficult to develop,
- but may as well turn out to be 100 times more difficult unless software engineering principles are used.

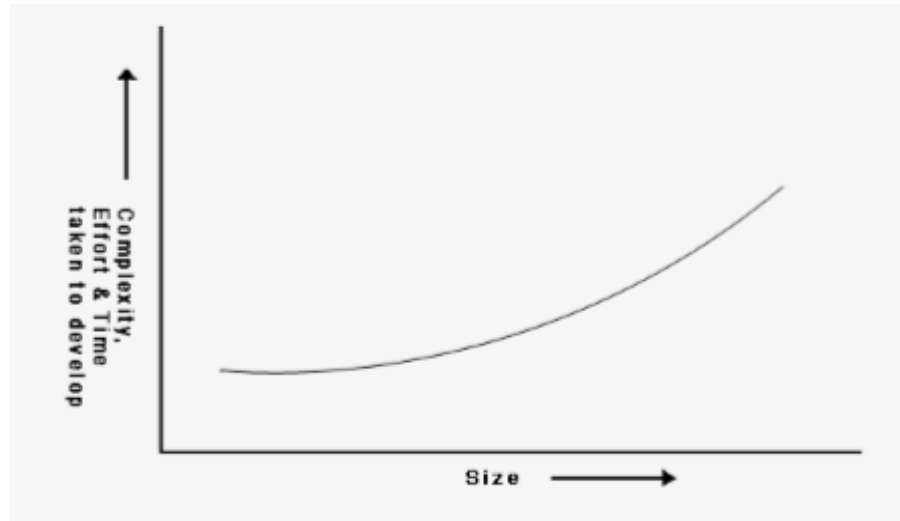


Fig. : Increase in development time and effort with problem size

Program Vs Software Product

- ▶ Programs are developed by individuals for their personal use. They are therefore, small in size and have limited functionality but software products are extremely large.
- ▶ In case of a program, the programmer himself is the sole user but on the other hand, in case of a software product, most users are not involved with the development.
- ▶ In case of a program, a single developer is involved but in case of a software product, a large number of developers are involved. For a program, the user interface may not be very important, because the programmer is the sole user.
- ▶ On the other hand, for a software product, user interface must be carefully designed and implemented because developers of that product and users of that product are totally different.

Life Cycle Model

- A software life cycle model (also called process model) is a descriptive and diagrammatic representation of the software life cycle.
- A life cycle model represents all the activities required to make a software product transit through its life cycle phases.
- It also captures the order in which these activities are to be undertaken.
- In other words, a life cycle model maps the different activities performed on a software product from its inception to retirement.

The need for a software life cycle model

- The development team must identify a suitable life cycle model for the particular project and then adhere to it.
- Without using of a particular life cycle model the development of a software product would not be in a systematic and disciplined manner.
- A software life cycle model defines entry and exit criteria for every phase. A phase can start only if its phase-entry criteria have been satisfied.
- So without software life cycle model the entry and exit criteria for a phase cannot be recognized.
- Without software life cycle models (such as classical waterfall model, iterative waterfall model, prototyping model, evolutionary model, spiral model etc.) it becomes difficult for software project managers to monitor the progress of the project.

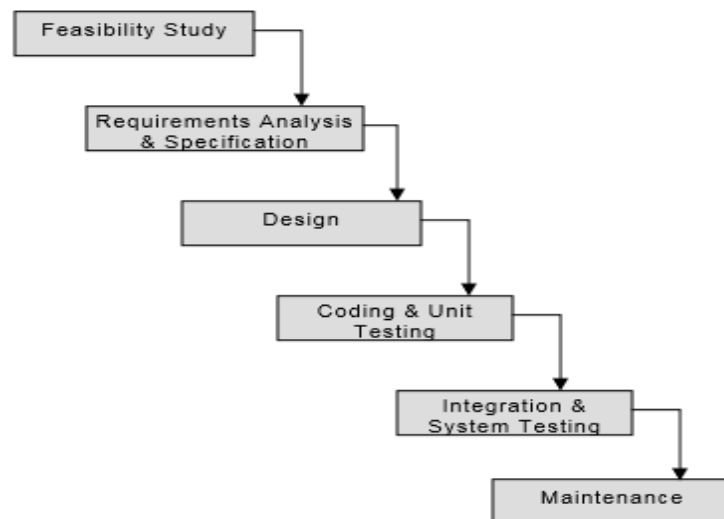
Different life cycle models

Many life cycle models have been proposed so far. Each of them has some advantages as well as some disadvantages. A few important and commonly used life cycle models are as follows:

- Classical Waterfall Model
- Iterative Waterfall Model
- Prototyping Model
- Evolutionary Model
- Spiral Model

Classical Waterfall Model

- The classical waterfall model is intuitively the most obvious way to develop software. Though the classical waterfall model is elegant and intuitively obvious, it is not a practical model in the sense that it cannot be used in actual software development projects.
- Thus, this model can be considered to be a theoretical way of developing software.
- Classical waterfall model divides the life cycle into the following phases:
 - Feasibility Study
 - Requirements Analysis and Specification
 - Design
 - Coding and Unit Testing
 - Integration and System Testing
 - Maintenance



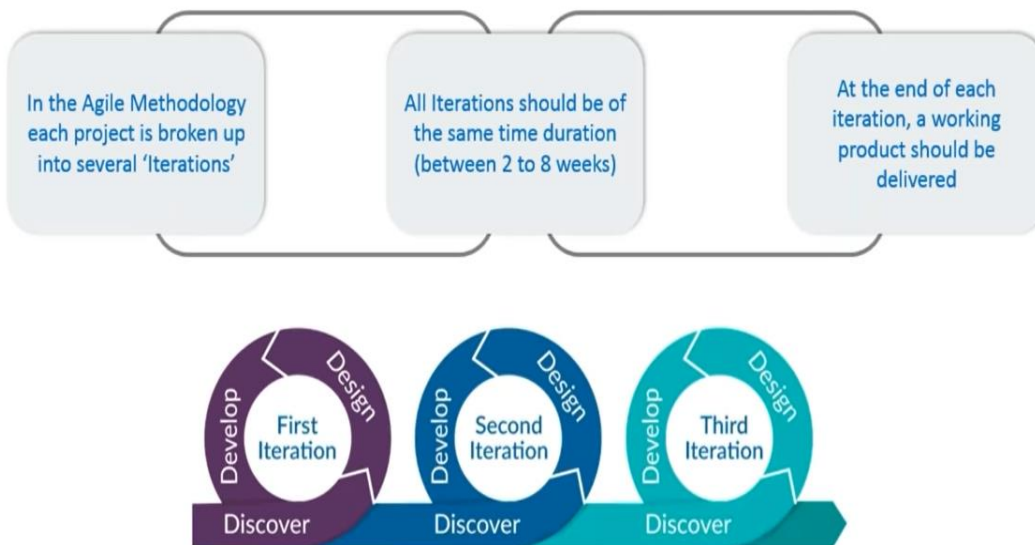
Classical Waterfall Model

Shortcomings of Classical Waterfall Model

- The classical waterfall model is an idealistic one since it assumes that no development error is ever committed by the engineers during any of the life cycle phases.
- However, in practical development environments, the engineers do commit a large number of errors in almost every phase of the life cycle.
- The source of the defects can be many: oversight, wrong assumptions, use of inappropriate technology, communication gap among the project engineers, etc.

- ▶ These defects usually get detected much later in the life cycle. For example, a design defect might go unnoticed till we reach the coding or testing phase.
- ▶ Once a defect is detected, the engineers need to go back to the phase where the defect had occurred and redo some of the work done during that phase and the subsequent phases to correct the defect and its effect on the later phases.
- ▶ Therefore, in any practical software development work, it is not possible to strictly follow the classical waterfall model.

What is Agile Methodology?

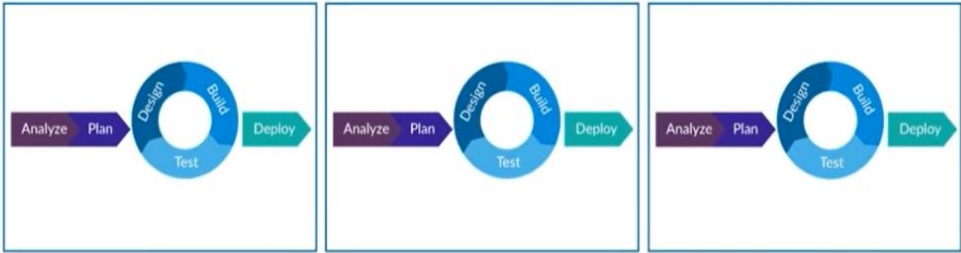


Waterfall vs Agile

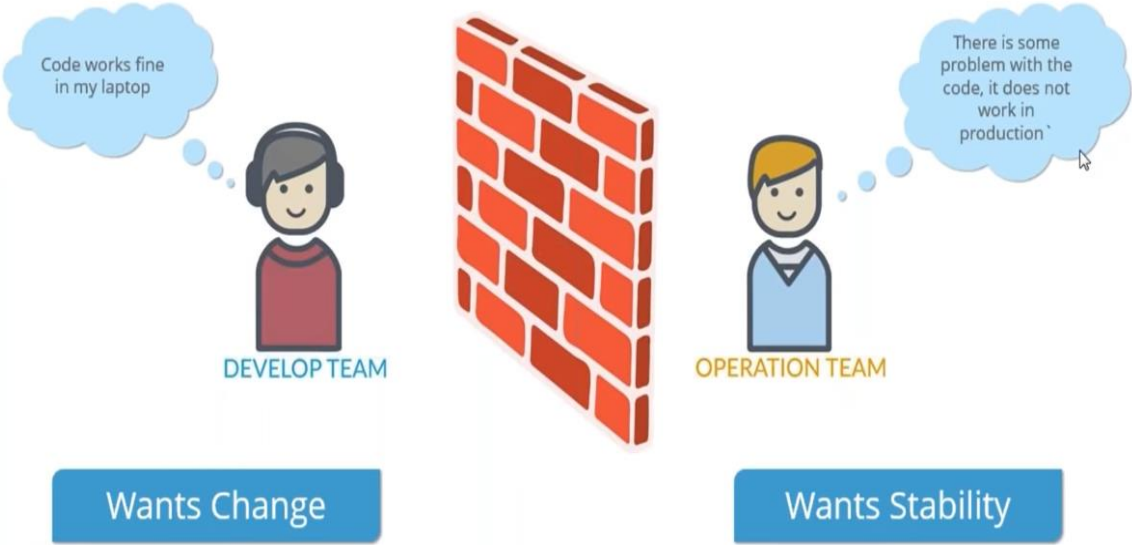
Waterfall



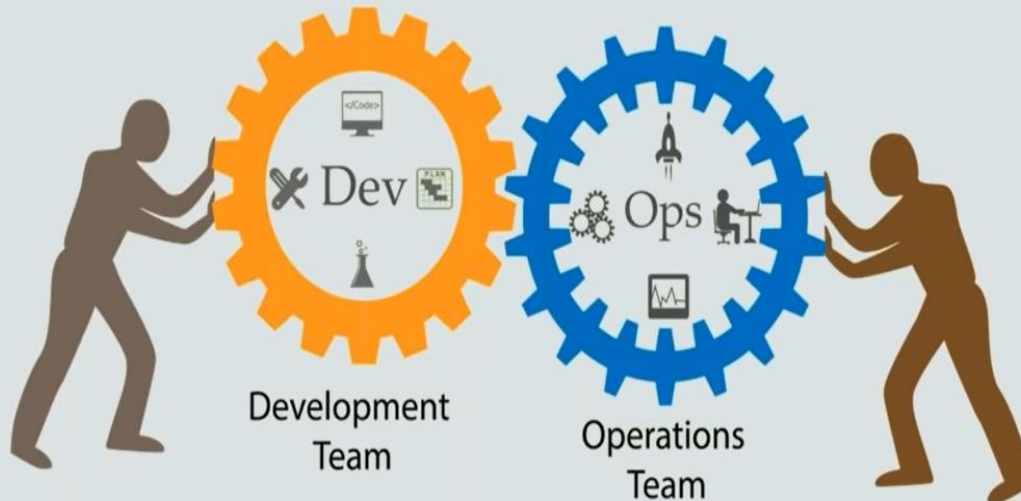
Agile



Limitations of Agile



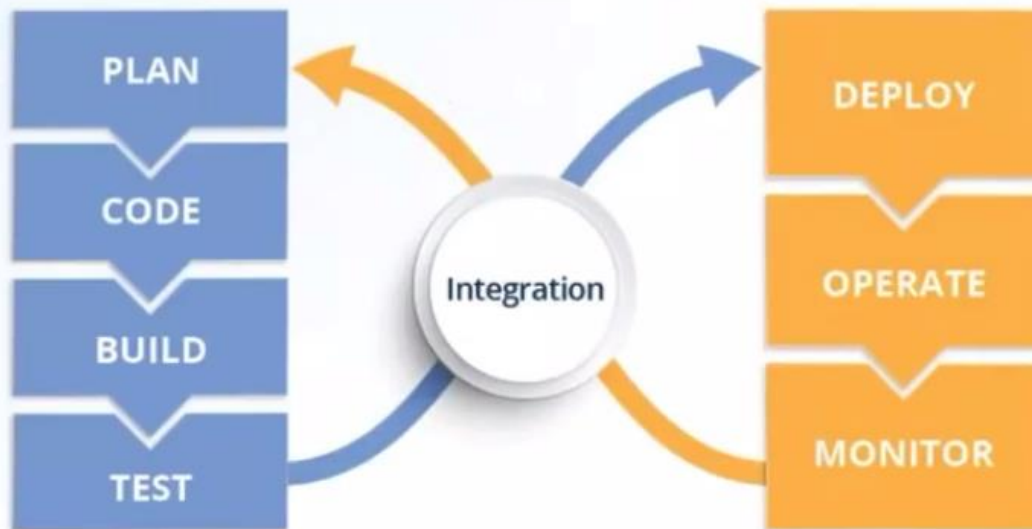
Solution is DevOps



What is DevOps?

- DevOps is an approach to bridge the gap between software development and operations
 - ➔ It is a response to the interdependence of software development and IT operations
 - ➔ DevOps is not a technology problem, but a business and cultural problem
 - ➔ Stresses collaboration between development and operations
 - ➔ Aims to deliver software and services quicker

DevOps is a Software Development approach which involves Continuous Development, Continuous Testing, Continuous Integration, Continuous Deployment and Continuous Monitoring of the software throughout its development lifecycle



What is DevOps and why it is gaining popularity?

- DevOps (Development and Operations) is a culture that promotes collaboration between the development and operations team to increase the organization's ability to deliver products or applications in an automated and repeatable way. Today an increasing number of organizations are implementing DevOps, fueled by reports of the benefits of DevOps which includes less time consumption to deliver things to market, reduced costs, increased security, and higher quality of products. So, it can be defined as a type of agile relationship between development and IT operations with a basic goal to improve the relationship between these two units.

Why DevOps is Gaining Popularity?

Minimal Time Accelerated Results

With DevOps, where operations and development teams are integrated, applications can be developed and deployed much more rapidly. As per business today, the success factor hinges largely on an organization's ability to deliver outcomes faster than the competition. Since changesets are smaller, problems tend to be less complex.

Collaboration Reduces Differences

DevOps make sure that the difference between operations and developments is eliminated by building a bridge to make them work together in a better way. The software development culture then continuously focuses on combined achievement rather than individual goals. The development environment becomes progressively more seamless as all team members work toward shared goals as it's no longer a matter of tossing the application code over the wall and hoping for the best.

Reduced Differences Increase Efficiency

Improved Collaboration increases efficiency. DevOps focuses on uniting the efforts of both of these departments as well as other departments such as testing, product management, and others in the team. Fast software delivery is crucial in today's digital age and a DevOps culture is a crucial aspect of this process.

Reduced Errors

DevOps reduces the chances of human errors during development and operations process. It lowers the application failure rate with multiple deployments in the process in a defined timeline by deploying frequent iterations. The shorter development cycles associated with a strong DevOps approach promotes more frequent code releases.

Stable operating environment

For any business or software platform, stability is the key. DevOps is established to bring stability with reliability. Organizations with DevOps get their deployment 30 times faster than their rivals with 50% lesser chances of failure.

Traditional Waterfall Model

Best suited when:

- Complete Requirements are clear and fixed
- Product definition is stable

Agile Development

Best suited when:

- Requirements change frequently
- Development needs to be fast

DevOps Approach

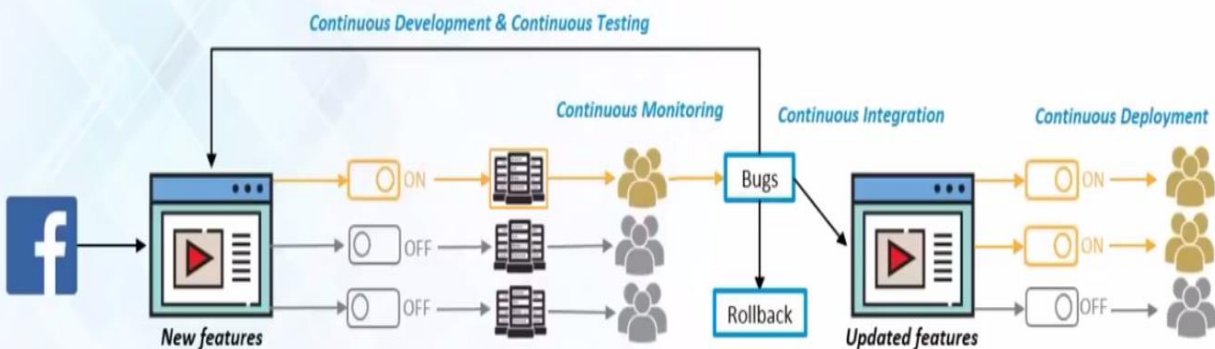
Best suited when:

- Requirements change frequently
- **Development** needs to be Agile
- **Operations** needs to be Agile

Dark Launching Technique

According to Dark Launching Technique:

- The new features are first deployed on a smaller & specific user base.
- They are continuously monitored and the feedbacks are continuously developed and tested.
- Once the features are stable, they are deployed on other user bases in multiple releases.



To Implement Dark launching, the below activities are fundamental as they lie at the heart of the DevOps lifecycle:

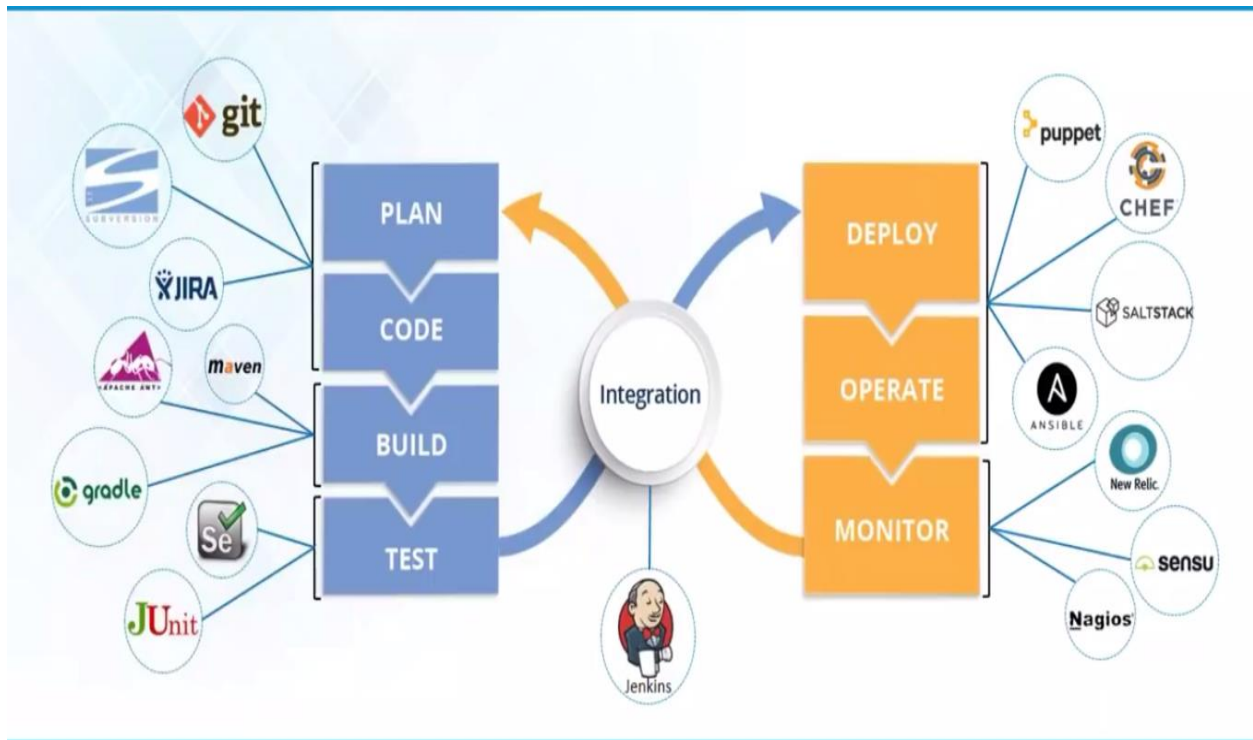
- Continuous Development
- Continuous Testing
- Continuous Integration
- Continuous Deployment
- Continuous Monitoring

Features	DevOps	Agile
Agility	Agility in both Development & Operations	Agility in only Development
Processes/ Practices	Involves processes such as CI, CD, CT, etc.	Involves practices such as Agile Scrum, Agile Kanban, etc.
Key Focus Area	Timeliness & quality have equal priority	Timeliness is the main priority
Release Cycles/ Development Sprints	Smaller release cycles with immediate feedback	Smaller release cycles
Source of Feedback	Feedback is from self (Monitoring tools)	Feedback is from customers
Scope of Work	Agility & need for Automation	Agility only

The most popular DevOps tools are mentioned below:

- Git : Version Control System tool
- Jenkins : Continuous Integration tool
- Selenium : Continuous Testing tool
- Puppet, Chef, Ansible : Configuration Management and Deployment tools
- Nagios : Continuous Monitoring tool
- Docker : Containerization tool

1. Developers develop the code and this source code is managed by Version Control System tools like Git etc.
2. Developers send this code to the Git repository and any changes made in the code is committed to this Repository.
3. Jenkins pulls this code from the repository using the Git plugin and build it using tools like Ant or Maven.
4. Configuration management tools like puppet deploys & provisions testing environment and then Jenkins releases this code on the test environment on which testing is done using tools like selenium.
5. Once the code is tested, Jenkins send it for deployment on the production server (even production server is provisioned & maintained by tools like puppet).
6. After deployment It is continuously monitored by tools like Nagios.
7. Docker containers provides testing environment to test the build features.



Different phases in DevOps methodology.



The various phases of the DevOps lifecycle are as follows:

- **Plan** – In this stage, all the requirements of the project and everything regarding the project like time for each stage, cost, etc are discussed. This will help everyone in the team to get a brief idea about the project.
- **Code** – The code is written over here according to the client’s requirements. Here codes are written in the form of small codes called units.
- **Build** – Building of the units is done in this step.
- **Test** – Testing is done in this stage and if there are mistakes found it is returned for re-build.
- **Integrate** – All the units of the codes are integrated into this step.
- **Deploy** – codeDevOpsNow is deployed in this step on the client’s environment.
- **Operate** – Operations are performed on the code if required.
- **Monitor** – Monitoring of the application is done over here in the client’s environment.

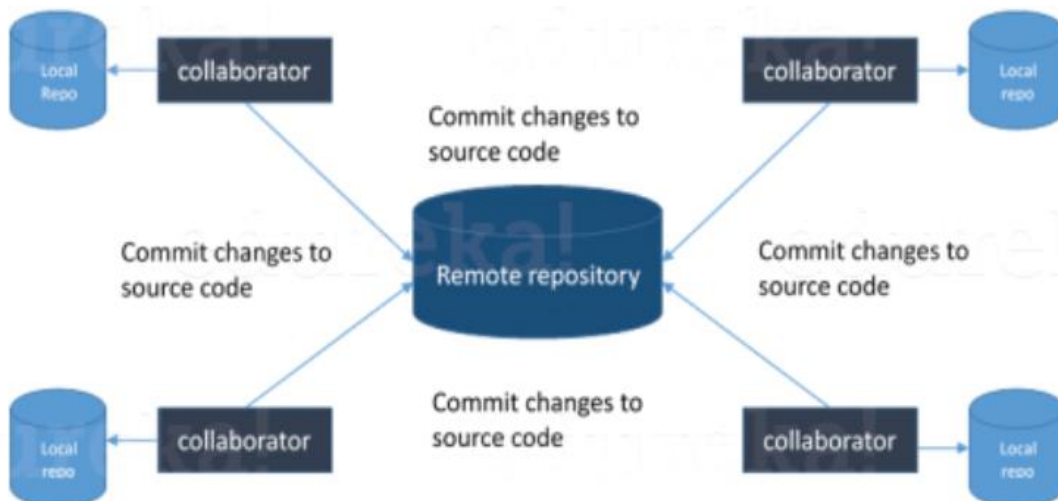
Unit-02

Version Control with Git, Install GIT and work with remote repositories, GIT workflows, Branching and Merging in Git. Understand the importance of Continuous Integration, Introduction to Jenkins, Jenkins management. Build and automation of Test using Jenkins and Maven.



What is Git?

- Git is a Distributed Version Control system (DVCS). It can track changes to a file and allows you to revert back to any particular change.
- Its distributed architecture provides many advantages over other Version Control Systems (VCS) like SVN one major advantage is that it does not rely on a central server to store all the versions of a project's files. Instead, every developer "clones" a copy of a repository I have shown in the diagram below with "Local repository" and has the full history of the project on his hard drive so that when there is a server outage, all you need for recovery is one of your teammate's local Git repository.
- There is a central cloud repository as well where developers can commit changes and share it with other teammates as you can see in the diagram where all collaborators are committing changes "Remote repository".



Some basic Git commands:

Command	Function
<code>git config --global user.name "name"</code> <code>git config --global user.email "E-mail"</code>	Configure the author name and email address to be used with your commits
<code>Git init</code>	Create a new local local repository
<code>Git clone /path/to/repository</code>	Create a working copy of a local repository
<code>git clone username@host:/path/to/repository</code>	For a remote server, use
<code>git add <Filename.></code> <code>git add *</code>	Add one or more file to staging
<code>git commit -m "Commit message"</code>	Commit changes to head
<code>git commit -a</code>	Commit any files you've added with git add, and also commit any files you've changed since then
<code>git push origin master</code>	Send changes to the master branch of your remote repository
<code>git status</code>	List the files you've changed and those you still need to add or commit
<code>git remote add origin <server></code>	If you haven't connected your local repository to a remote server, add the server to be able to push to it

What is Version control?

It is a system that records changes to a file or set of files over time so that you can recall specific versions later. Version control systems consist of a central shared repository where teammates can commit changes to a file or set of file. Then you can mention the uses of version control.

Version control allows you to:

- Revert files back to a previous state.
- Revert the entire project back to a previous state.
- Compare changes over time.
- See who last modified something that might be causing a problem.
- Who introduced an issue and when.

What are the benefits of using version control?

1. With Version Control System (VCS), all the team members are allowed to work freely on any file at any time. VCS will later allow you to merge all the changes into a common version.
2. All the past versions and variants are neatly packed up inside the VCS. When you need it, you can request any version at any time and you'll have a snapshot of the complete project right at hand.
3. Every time you save a new version of your project, your VCS requires you to provide a short description of what was changed. Additionally, you can see what exactly was changed in the file's content. This allows you to know who has made what change in the project.
4. A distributed VCS like Git allows all the team members to have complete history of the project so if there is a breakdown in the central server you can use any of your teammate's local Git repository.

What is Git rebase and how can it be used to resolve conflicts in a feature branch before merge?

- Git rebase is a command which will merge another branch into the branch where you are currently working, and move all of the local commits that are ahead of the rebased branch to the top of the history on that branch.
- Now once you have defined Git rebase time for an example to show how it can be used to resolve conflicts in a feature branch before merge, if a feature branch was created from master, and since then the master branch has received new commits, Git rebase can be used to move the feature branch to the tip of master.
- The command effectively will replay the changes made in the feature branch at the tip of master, allowing conflicts to be resolved in the process. When done with care, this will allow the feature branch to be merged into master with relative ease and sometimes as a simple fast-forward operation.

Explain the difference between a centralized and distributed version control system (VCS).?

Centralized Version Control	Distributed Version control
In this, we will not be having a copy of the main repository on the local repository.	In this, all the developers will be having a copy of the main repository on their local repository
There is a need for the internet for accessing the main repository data because we will not be having another copy of the server on the local repository	There is no need for the internet for accessing the main repository data because we will be having another copy of the server on the local repository.
If the main server crashes then there will be a problem in accessing the server for the developers.	If there is a crash on the main server there will be no problem faced regarding the availability of the server.

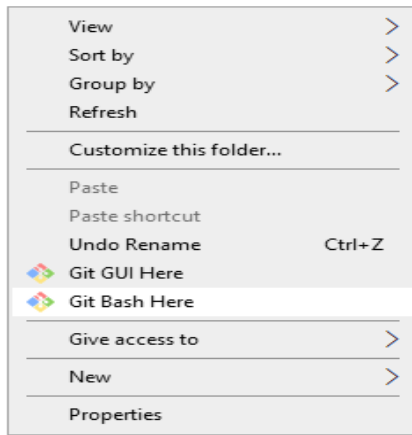
Git Lab

Install the git in windows follow the link <https://gitforwindows.org/>

#####CREATING REPOSITORY#####

1. Create a folder in any drive name "yourname_git_demo"
2. Navigate into the created folder and right_click
3. Choose the option "Git Bash Here"
4. In the Git Bash terminal type "git init", it will initialize empty git repository

(Stay in the same folder)



```
MINGW64:/c/DevOpsLab/GitDemo
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo
$ git init
Initialized empty Git repository in C:/DevOpsLab/GitDemo/.git/

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo
(master)
$ |
```

GitHub interface showing the 'New' button circled in red. The page includes a search bar, navigation links (Pull requests, Issues, Marketplace, Explore), a list of repositories, and a 'Learn Git and GitHub' section with a 'Read the guide' button.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner: / Repository name *:

Great repository names are short and memorable. Need inspiration? How about **stunning-parakeet**?

Description (optional):

Public
Anyone can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: | Add a license: ⓘ

[Create repository](#)

ofcoursevijay / sample_rep Unwatch 1 Star 0 Fork 0

[Code](#) [Issues 0](#) [Pull requests 0](#) [Actions](#) [Projects 0](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

No description, website, or topics provided. [Edit](#)

[Manage topics](#)

1 commit | 1 branch | 0 packages | 0 releases | 1 contributor

Branch: master | [New pull request](#) | [Create new file](#) | [Upload files](#) | [Find file](#) | [Clone or download](#)

ofcoursevijay Initial commit

[README.md](#) Initial commit

[README.md](#)

Clone with HTTPS ⓘ [Use SSH](#)

Use Git or checkout with SVN using the web URL.

https://github.com/ofcoursevijay/sample_

[Open in Desktop](#) | [Download ZIP](#)

#linking remote repository

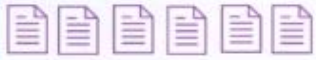


git remote add origin "copy and paste the link from your git-hub repository, if you have already created (Otherwise create a new repository using your git account)"

git pull origin master

```
MINGW64:/c/DevOpsLab/GitDemo
Dell@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git remote add origin "https://github.com/ofcoursevijay/sample_rep.git"
Dell@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git pull origin master
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/ofcoursevijay/sample_rep
 * branch          master      -> FETCH_HEAD
 * [new branch]    master      -> origin/master
Dell@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$
```

This PC > Local Disk (C:) > DevOpsLab > GitDemo

Name	Date modified	Type	Size
.git	29-01-2020 11:08	File folder	
README.md	29-01-2020 11:08	MD File	1 KB

git status		➤ Tells you which files are added to index and are ready to commit.
git add		➤ Lets you add files to your index.
git commit	<p>Commit</p> 	<ul style="list-style-type: none"> ➤ It refers to recording snapshots of the repository at a given time. ➤ Committed snapshots will never change unless done explicitly.

Create some text file and write anything into it

This PC > Local Disk (C:) > DevOpsLab > GitDemo

Name	Date modified	Type	Size
.git	29-01-2020 11:08	File folder	
README.md	29-01-2020 11:08	MD File	1 KB
textfile1.txt	29-01-2020 11:25	Text Document	1 KB



```
MINGW64:/c/DevOpsLab/GitDemo
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        textfile1.txt

nothing added to commit but untracked files present (use "git add" to track
)
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ |
```

```
MINGW64:/c/DevOpsLab/GitDemo
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git add textfile1.txt
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   textfile1.txt

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$
```

Navigate into the created folder and right click->new->Text document

Name the file as text1 and write something into the file and save it.

```
git status
```

```
git add textfile1.txt
```

```
git status
```

```
git commit -m "first commit"
```

```
## changes finally committed in local repository
```

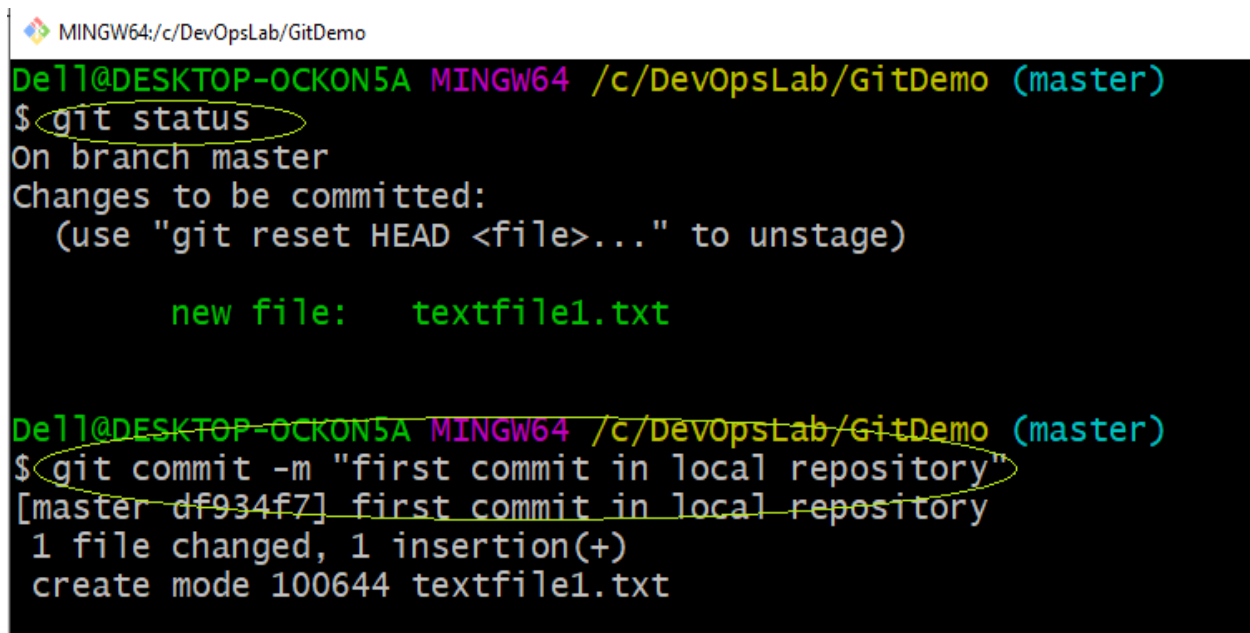
```
## create more files in the same folder as text2.txt, text3.txt and repeat and understand the above procedure.
```

```
## Explore what is tracked file, untracked file and modified file.
```

```
## git add -A // is used to add all files at once.
```

```
git status
```

```
git commit -a -m "adding all files together"
```



```
MINGW64:/c/DevOpsLab/GitDemo
Dell@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

       new file:   textfile1.txt

Dell@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git commit -m "first commit in local repository"
[master df934f7] first commit in local repository
 1 file changed, 1 insertion(+)
 create mode 100644 textfile1.txt
```

```
## to check how git stores all the commits, use git log
```

```
MINGW64; c/DevOpsLab/GitDemo
Dell@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git log
commit df934f793ddc72c376496d466c78039a3248f8b6 (HEAD -> master)
Author: Vijay Singh <vijaysingh_agra@hotmail.com>
Date: Wed Jan 29 11:42:28 2020 +0530

    first commit in local repository

commit b99c2f98bb59b18a46fb9a431d08b2e23c52616f (origin/master)
Author: ofcoursevijay <37365613+ofcoursevijay@users.noreply.github.com>
Date: Wed Jan 29 11:03:03 2020 +0530

    Initial commit

Dell@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$
```

#####PARALLEL DEVELOPMENT#####

Branching

- Branches are pointers to a specific commit.
- Branches are of two types:
 - Local branches
 - Remote-tracking branches



```
MINGW64:/c/DevOpsLab/GitDemo
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git branch firstbranch
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git checkout firstbranch ←
Switched to branch 'firstbranch'
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$
```

Create some text file in the same folder and write something into it

> This PC > Local Disk (C:) > DevOpsLab > GitDemo >

Name	Date modified	Type	Size
.git	29-01-2020 12:05	File folder	
README.md	29-01-2020 11:08	MD File	1 KB
textfile1.txt	29-01-2020 11:25	Text Document	1 KB
textfile2.txt	29-01-2020 12:10	Text Document	1 KB

textfile2.txt - Notepad

File Edit Format View Help

firstbranch first file

```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ git add textfile2.txt
```

```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ git commit -m "making changes in firstbranch"
[firstbranch a4f37f1] making changes in firstbranch
1 file changed, 1 insertion(+)
create mode 100644 textfile2.txt
```

```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$
```

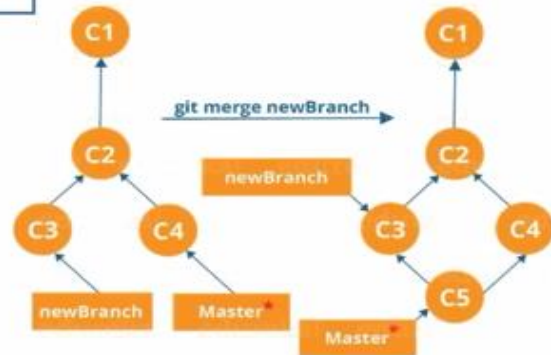
```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ ls
README.md  textfile1.txt  textfile2.txt
```

```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ git checkout master
Switched to branch 'master'
```

```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ ls
README.md  textfile1.txt
```

```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$
```

- It is a way to combine the work of different branches together.
- Allows to branch off, develop a new feature & combine it back in.



```

MINGW64:/c/DevOpsLab/GitDemo
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ ls
README.md  textfile1.txt

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git merge firstbranch
Updating df934f7..a4f37f1
Fast-forward
 textfile2.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 textfile2.txt

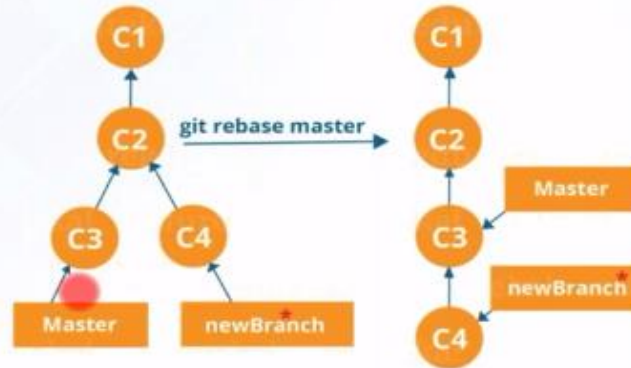
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ ls
README.md  textfile1.txt  textfile2.txt

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ |

```

Rebasing

- This is also a way of combining the work between different branches.
- It can be used to make a linear sequence of commits.



Create some text files as textfile3.txt and textfile4.txt and write something into it

› This PC › Local Disk (C:) › DevOpsLab › GitDemo ›

Name	Date modified	Type	Size
.git	29-01-2020 13:20	File folder	
README.md	29-01-2020 11:08	MD File	1 KB
testfile3.txt	29-01-2020 13:21	Text Document	1 KB
testfile1.txt	29-01-2020 11:25	Text Document	1 KB
testfile2.txt	29-01-2020 13:07	Text Document	1 KB
testfile4.txt	29-01-2020 13:21	Text Document	1 KB

MINGW64:/c/DevOpsLab/GitDemo

```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git checkout firstbranch
Switched to branch 'firstbranch'

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ git add -A

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ git commit -a -m "adding files for rebasing demo"
[firstbranch 2c75a32] adding files for rebasing demo
 2 files changed, 2 insertions(+)
 create mode 100644 testfile3.txt
 create mode 100644 textfile4.txt

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ |
```

MINGW64:/c/DevOpsLab/GitDemo

```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ ls
README.md  testfile3.txt  textfile1.txt  textfile2.txt  textfile4.txt

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ git checkout master
Switched to branch 'master'

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ ls
README.md  textfile1.txt  textfile2.txt

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ |
```

```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git checkout firstbranch
Switched to branch 'firstbranch'

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ git rebase master
Current branch firstbranch is up to date.

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ |
```

```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ git checkout master
Switched to branch 'master'

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git rebase firstbranch
First, rewinding head to replay your work on top of it...
Fast-forwarded master to firstbranch.

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ ls
README.md  testfile3.txt  testfile1.txt  testfile2.txt  testfile4.txt

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$
```

```
P Dell@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$
P
A Dell@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ ssh-keygen
S Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Dell/.ssh/id_rsa):
S /c/Users/Dell/.ssh/id_rsa already exists.
E Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
N Enter same passphrase again:
B Your identification has been saved in /c/Users/Dell/.ssh/id_rsa.
Your public key has been saved in /c/Users/Dell/.ssh/id_rsa.pub.
S The key fingerprint is:
SHA256:/SIksth/9gLWqy7ru6SHpiOH5pwyK6jQa6VP0GM329I Dell@DESKTOP-OCKON5A
The key's randomart image is:
B +----[RSA 2048]-----+
R |
C |
| . * = S .
| . o . * O o . .
| o . . o o . + . E . . .
| X . * o o o o . o + .
| O X o = o o = * o o . o .
+----[SHA256]-----+
Dell@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ cat /c/Users/Dell/.ssh/id_rsa.pub
```

ofcoursevijay / sample_rep

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

No description, website, or topics provided. Edit

Manage topics

1 commit 1 branch 0 packages 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

ofcoursevijay Initial commit

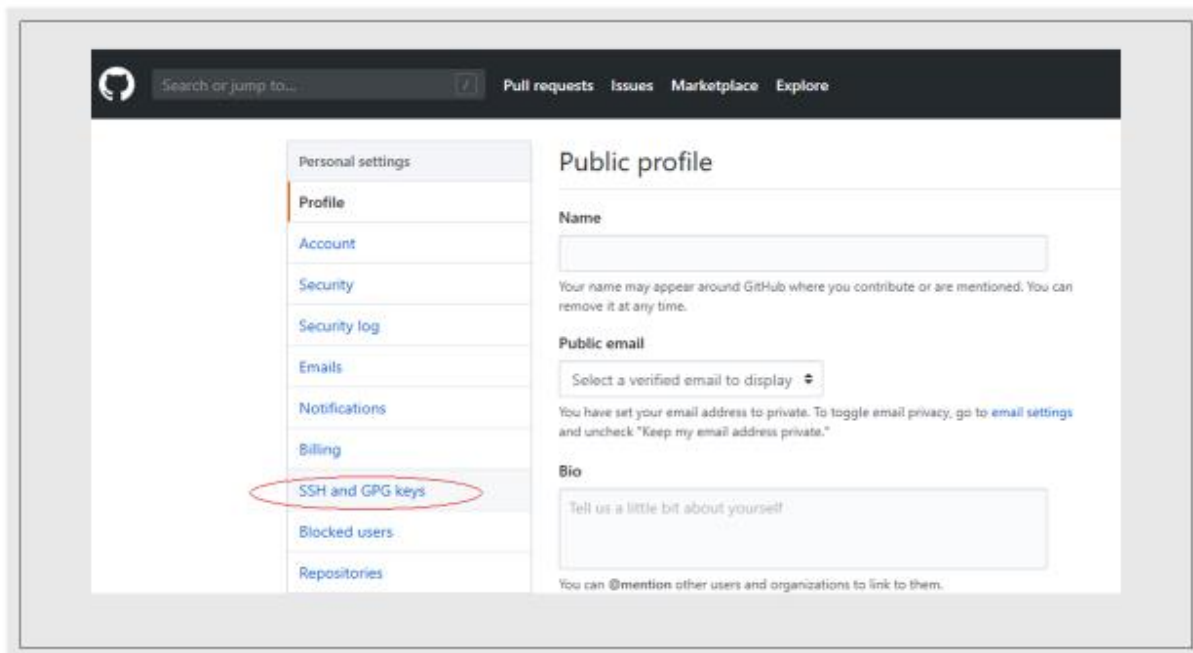
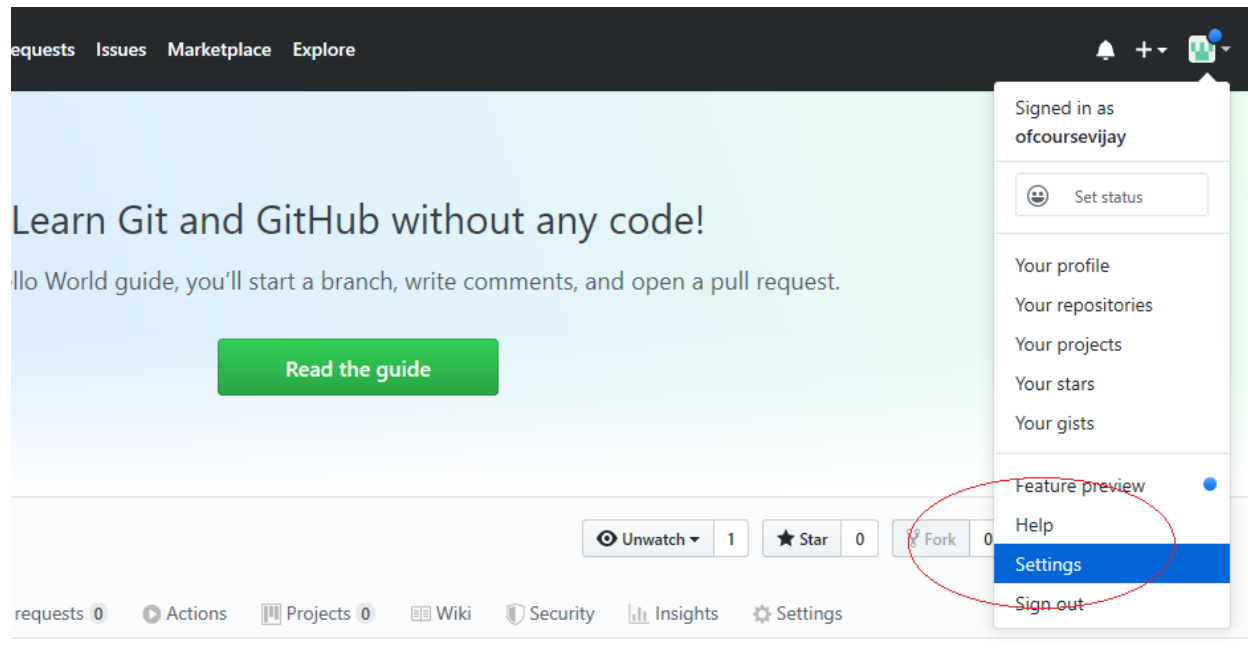
README.md Initial commit

Clone with HTTPS Use Git or checkout with SVN using the web URL. Use SSH

https://github.com/ofcoursevijay/sample_

Open in Desktop Download ZIP


sample_rep



SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

 **ssh1**
40:53:f1:15:20:c2:1e:ec:e7:e4:75:94:c0:ed:a7:3b
Added on 8 Feb 2019
Never used — Read/write

[Delete](#)

Check out our guide to [generating SSH keys](#) or [troubleshoot common SSH Problems](#).

GPG keys

[New GPG key](#)

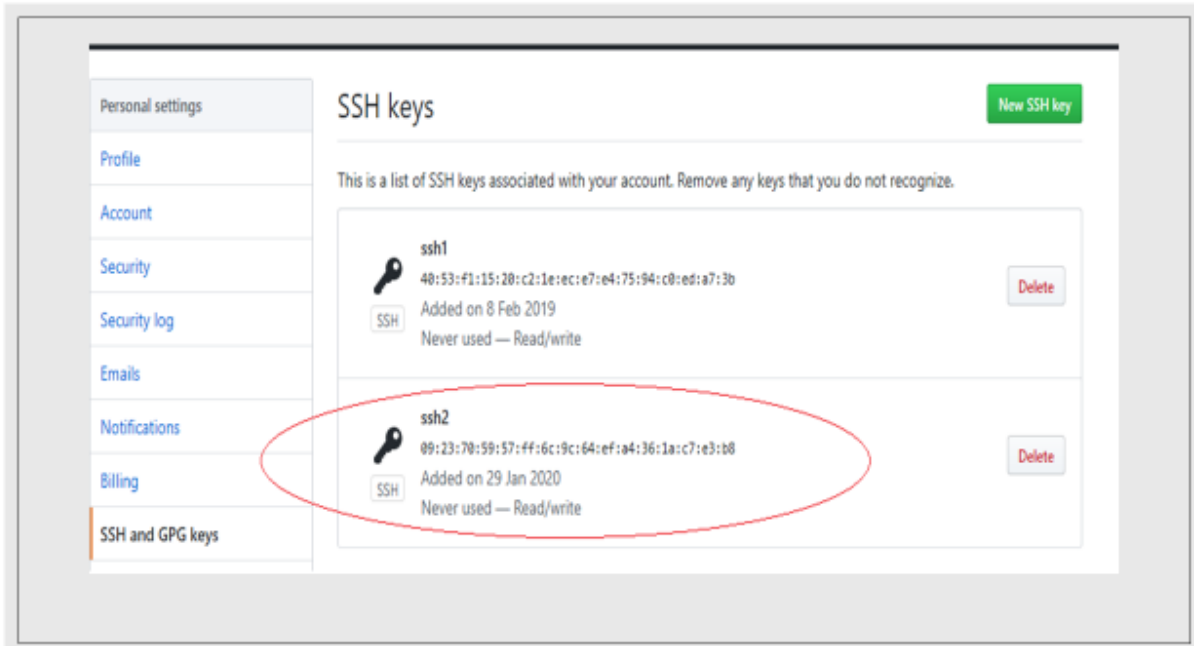
SSH keys / Add new

Title
ss2

Key

```
mOUonUb17zHUf0cSDqqy1/q8wZEf99K0MSKQ7Y4yQj8dVEp+wWRbQUaYGF3Y0v  
QgdXO897W0mage/AJoMge5mcpJEpUVEjxiWhMCgYrBmNz7aniCVHPVymbO2dD0  
gnby/8KrYImcleQiqGoQt3RuYeAH1j76gAgCoN/H7hHtEggzKAY+hmiyy89iL81e8  
Wj/RGOlcG7daN8GdgwJbMRQ3aQ8Wp2iBfoQJaSJaImLDyh3to6XUuRVZuGgG7E  
p2JbOoDxHz38X-HIOj9qWQK8gFWs5R/mLH4ch7ZB+yW8b1cla9s89QPyZ2/9lcy  
Au7hdmtwptFaQdGivn3i+Aj0ITBRz03+EsulIGA/hqzC9N8MECv8JectHU5Ieztl  
8hsZpcm5ZUCp3/7umH3L2r3/W4DwqybcgffaPgFuT2Pxbi9TXimzulCIR3Pxt/R  
RHUSAoGBAMJkjZUNRXDILKoRM3MjP+OOhLCqrW+ePox7qr0HF1MvmO+OvWkbsH0  
Ztu7SU05gcuOQGGV9kAlaKk5FaGaq7YHb/K3J5H0Xq8BpmuWjd94t7Gs9QYvIsDo  
PtrVvCkLr1zWTwPP9gBIE32Kuvd1+TdBuisDvHR7H10cLv55KDO
```

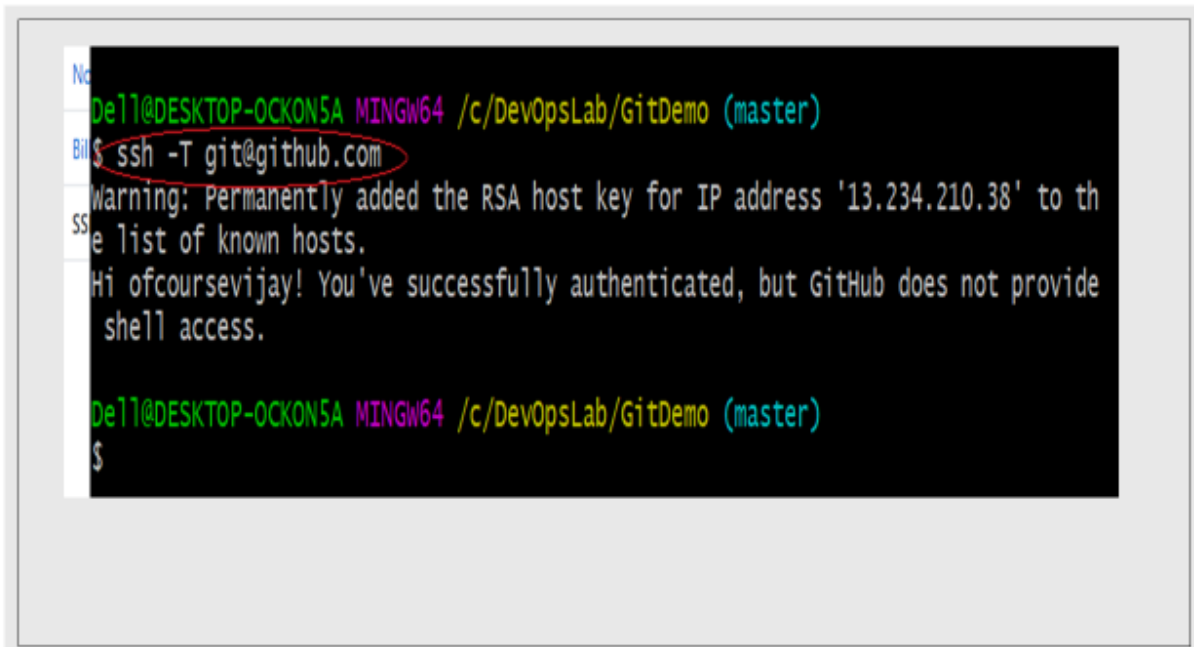
[Add SSH key](#)



The screenshot shows the GitHub 'SSH keys' page. On the left is a sidebar with navigation links: Personal settings, Profile, Account, Security, Security log, Emails, Notifications, Billing, and SSH and GPG keys. The main content area is titled 'SSH keys' and includes a 'New SSH key' button. Below the title is a message: 'This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.' There are two SSH keys listed:

- ssh1**: 48:53:f1:15:28:c2:1e:ec:e7:e4:75:94:c8:ed:a7:3b. Added on 8 Feb 2019. Never used — Read/write. A 'Delete' button is to its right.
- ssh2**: 09:23:70:59:57:ff:6c:9c:64:ef:a4:36:1a:c7:e3:b8. Added on 29 Jan 2020. Never used — Read/write. A 'Delete' button is to its right.

The 'ssh2' key entry is circled in red.



```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ ssh -T git@github.com
Warning: Permanently added the RSA host key for IP address '13.234.210.38' to the list of known hosts.
Hi ofcoursevijay! You've successfully authenticated, but GitHub does not provide shell access.
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$
```

```
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git checkout firstbranch
Switched to branch 'firstbranch'

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ git push origin firstbranch
Counting objects: 10, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (10/10), 864 bytes | 432.00 KiB/s, done.
Total 10 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'firstbranch' on GitHub by visiting:
remote:   https://github.com/ofcoursevijay/sample_rep/pull/new/firstbranch
remote:
To https://github.com/ofcoursevijay/sample_rep.git
 * [new branch]      firstbranch -> firstbranch

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$
```

No description, website, or topics provided.

Edit

[Manage topics](#)

1 commit 2 branches 0 packages 0 releases 1 contributor

Your recently pushed branches:

firstbranch (1 minute ago) [Compare & pull request](#)

Branch: master [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

ofcoursevijay Initial commit Latest commit b99c2f9 3 hours ago

README.md Initial commit 3 hours ago


README.md







sample_rep

Branch: firstbranch ▾ New pull request

Create new file Upload files Find file Clone or download ▾

This branch is 3 commits ahead of master. [Pull request](#) [Compare](#)

 ofcoursevijay adding files for rebasing demo Latest commit 2c75a32 1 hour ago

 README.md	Initial commit	3 hours ago
 testfile3.txt	adding files for rebasing demo	1 hour ago
 testfile1.txt	first commit in local repository	3 hours ago
 testfile2.txt	making changes in firstbranch	2 hours ago
 testfile4.txt	adding files for rebasing demo	1 hour ago
 README.md		Edit

```
MINGW64:/c/DevOpsLab/GitDemo
De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (firstbranch)
$ git checkout master
Switched to branch 'master'

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$ git push origin master
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/ofcoursevijay/sample_rep.git
 b99c2f9..2c75a32  master -> master

De11@DESKTOP-OCKON5A MINGW64 /c/DevOpsLab/GitDemo (master)
$
```

Branch: firstbranch ▾ New pull request

Create new file Upload files Find file

Switch branches/tags Pull

Find or create a branch...

Branches Tags

- master default
- ✓ firstbranch

textfile2.txt	making changes in firstbranch
textfile4.txt	adding files for rebasing demo

README.md

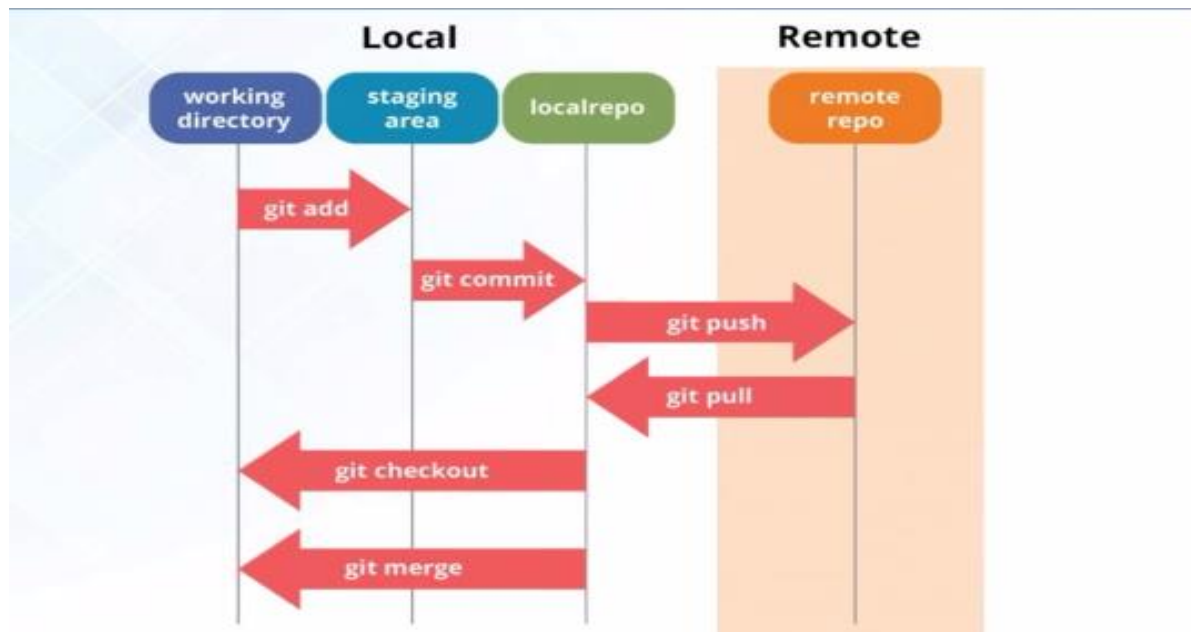
Branch: master ▾ New pull request

Create new file

ofcoursevijay adding files for rebasing demo

README.md	Initial commit
testfile3.txt	adding files for rebasing demo
textfile1.txt	first commit in local repository
textfile2.txt	making changes in firstbranch
textfile4.txt	adding files for rebasing demo

README.md



What is continuous integration?

Continuous integration is a software development process where developers integrate the new code they've written more frequently throughout the development cycle, adding it to the code base at least once a day. Automated testing is done against each iteration of the build to identify integration issues earlier, when they are easier to fix, which also helps avoid problems at the final merge for the release. Overall, continuous integration helps streamline the build process, resulting in higher-quality software and more predictable delivery schedules.

Continuous integration and DevOps

In the DevOps framework, continuous integration sits at the beginning of the software development process, where you check in your code at least once a day to keep your local copies from drifting too far away from the main branch of the code build. This helps you avoid disastrous merge conflicts that could "break" the build and take the team hours or days to resolve.

Continuous integration serves as a prerequisite for the testing, deployment and release stages of continuous delivery. The entire development team will know within minutes of check-in whether you've created bad code, as the continuous integration service automatically builds and tests your code changes for any errors.

Continuous integration (CI) vs. continuous delivery (CD) vs. continuous deployment

Continuous delivery and continuous deployment follow continuous integration in the DevOps cycle.

Continuous delivery (CD) picks up where continuous integration ends, automating the delivery of applications to selected infrastructure environments. CD focuses on delivering any validated changes to the code base—updates, bug fixes, even new features—to users as quickly and safely as possible. It ensures the automation of pushing code changes to different environments, such as development, testing and production.

In **continuous deployment**, the code changes to an application are released automatically into the production environment. This automation is driven by a series of predefined tests. Once new updates pass those tests, the system pushes the updates directly to the software's users.

Benefits of continuous integration

Commonly cited benefit of continuous integration include:

- Early and improved error detection and metrics that let you address errors early—sometimes within minutes of check-in
- Continuous and demonstrated progress for improved feedback
- Improved team collaboration; everyone on the team can change the code, integrate the system and quickly determine conflicts with other parts of the software
- Improved system integration, which reduces surprises at the end of the software development lifecycle
- Fewer parallel changes for merging and testing

- Reduced number of errors during system testing
- Constantly updated systems to test against

Open source continuous integration tools

Popular open source continuous integration tools include:

- **Jenkins:** A widely used open source continuous integration tool, Jenkins allows developers to automatically build, integrate and test code as soon as they commit it to the source repository, making it easier for developers to catch bugs early and deploy software faster. The Docker plug-in is available on Jenkins.
- **Buildbot:** Buildbot can automate all aspects of the software development cycle. As a job scheduling system, it queues and executes jobs, and reports results.
- **Go:** What makes Go stand out from the crowd is the concept of pipelines, which makes the modeling of complex build workflows easy.
- **Travis CI:** One of the oldest and most-trusted hosted solutions, it is also available in an on-premises version for the enterprise.
- **GitLab CI:** An integral part of the open source Rails project, GitLab CI is a free hosted service that provides detailed git repository management with features like access control, issue tracking, code reviews and more.

Explain some common practices of CI/CD.

Continuous Integration (CI) and Continuous Deployment/Delivery (CD) are software development practices that aim to automate and improve the software development process. Some common CI/CD practices include:

1. **Automated building and testing of code changes:** Every code change is automatically built and tested to ensure that it does not break the existing functionality.
2. **Code reviews and collaboration:** Teams review code changes and provide feedback before they are merged into the main codebase.
3. **Automated deployment:** The process of deploying code changes to a production environment is automated and can be triggered by a successful build and test.

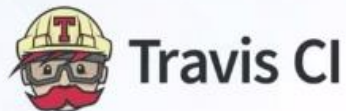
4. **Infrastructure as Code (IaC):** The infrastructure that supports the application is managed as code and versioned, allowing for easier and more consistent deployment.
5. **Continuous monitoring and logging:** The application is continuously monitored for performance and errors, and logs are automatically collected and analyzed.
6. **Rollback capabilities:** The ability to quickly and easily roll back to a previous version of the application in case of failure.
7. **Security scans:** Security scans are performed on the code and infrastructure to identify vulnerabilities and security risks.



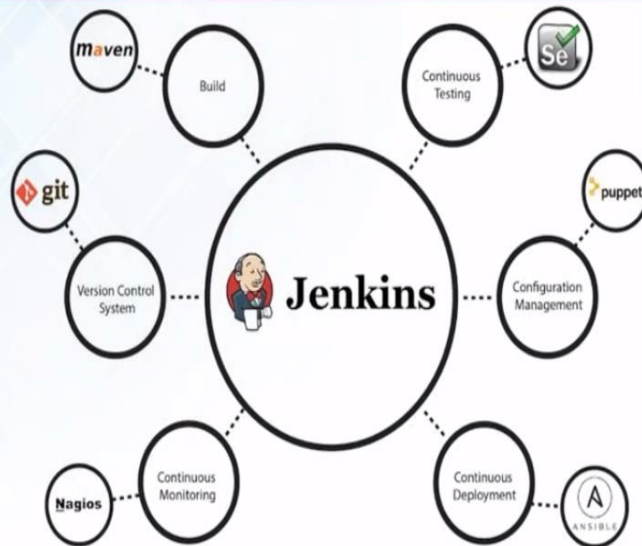
Jenkins

The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project. Jenkins offers a simple way to set up a continuous integration and continuous delivery environment for almost any combination of languages and source code repositories. While Jenkins doesn't eliminate the need to create scripts for individual steps, it does give you a faster and more robust way to integrate your entire chain of build, test, and deployment tools than you can easily build yourself.

Continuous Integration Tools

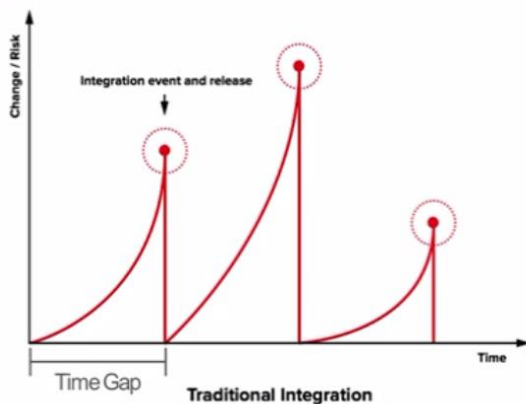


Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins allows integration of various DevOps stages.



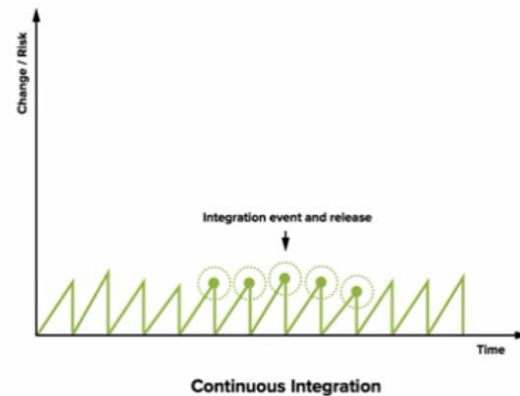
Traditional Integration Vs Continuous Integration

Traditional Integration



- Greater time gap in case of Traditional Integration
- Relatively greater risk or change in conflicts

Continuous Integration



- Integration time gap is relatively much lesser
- Relatively lesser risk or change in conflicts

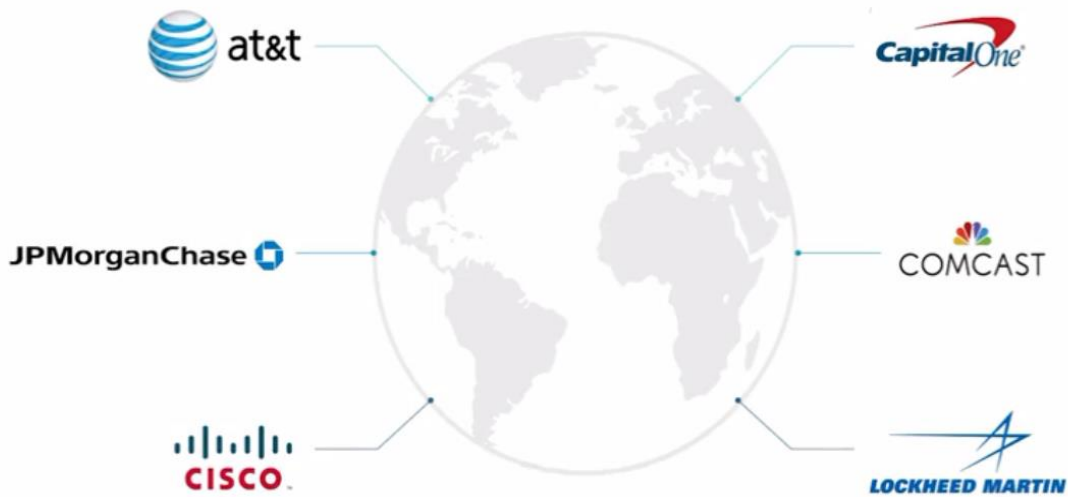
Importance Of Continuous Integration



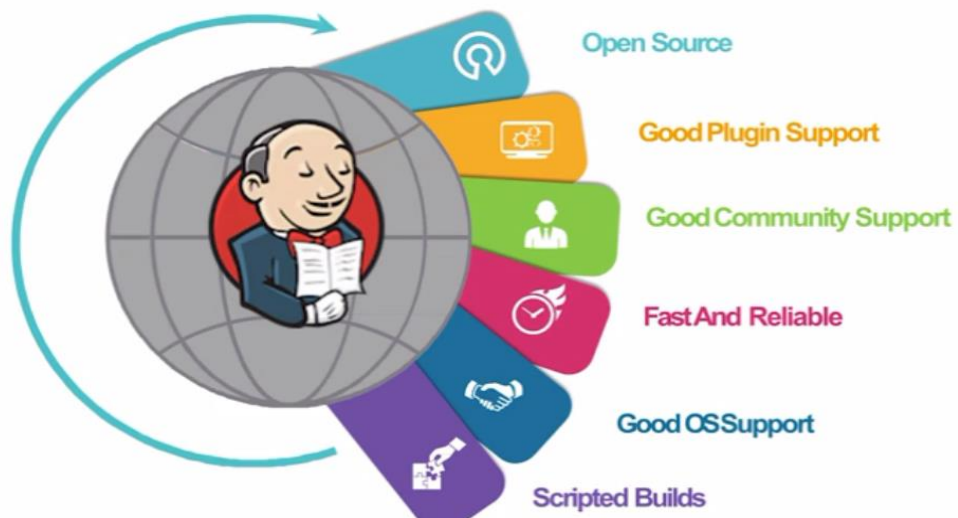
Popular Continuous Integration Tools



Companies Using Jenkins



Why Jenkins Is So Popular?



Features Of Jenkins



Jenkins Lab

Download Jenkins for windows

<https://jenkins.io/download/>

https://jenkins.io/download/

Blog Documentation Plugins Community Subprojects




Jenkins




Build great things at any scale

The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project.

Documentation Download

Getting started with Jenkins

Gentoo 
macOS 
OpenBSD 
openSUSE
Red Hat/Fedora/CentOS
Ubuntu/Debian
Windows
Generic Java package (.war)

macOS 
OpenBSD 
openSUSE
Red Hat/Fedora/CentOS
Ubuntu/Debian
OpenIndiana Hipster 
Windows
Generic Java package (.war)

localhost:8080/login?from=%2F

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword

Please copy the password from either location and paste it below.

Administrator password

[Continue](#)

his PC > Local Disk (C:) > Program Files (x86) > Jenkins > secrets >

Name	Date modified	Type	Size
filepath-filters.d	06-02-2019 16:34	File folder	
whitelisted-callables.d	06-02-2019 16:34	File folder	
initialAdminPassword	06-02-2019 16:34	File	1 KB
jenkins.model.Jenkins.crumbSalt	06-02-2019 16:34	CRUMBSALT File	1 KB
master.key	06-02-2019 16:34	Keynote document	1 KB
org.jenkinsci.main.modules.instance_ide...	06-02-2019 16:34	Keynote document	1 KB
slave-to-master-security-kill-switch	06-02-2019 16:34	File	1 KB

localhost:8080/

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Getting Started

✗ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✗ Credentials Binding	4.x API
✓ Timestamper	✗ Workspace Cleanup	✓ Ant	✓ Gradle	** Display URL API
🔄 Pipeline	🔄 GitHub Branch Source	🔄 Pipeline: GitHub Groovy Libraries	✓ Pipeline: Stage View	Mailer
🔄 Git	🔄 Subversion	🔄 SSH Slaves	🔄 Matrix Authorization Strategy	** Pipeline: Basic St
🔄 PAM Authentication	🔄 LDAP	🔄 Email Extension	✓ Mailer	Gradle

** Pipeline: Milestone
** Jackson 2 API
** Pipeline: Input St
** Pipeline: Stage St
** Pipeline Graph Anal
** Pipeline: REST API
** JavaScript GUI Lib:
bundle
** JavaScript GUI Lib:
bundle
Pipeline: Stage View
** Pipeline: Build St
** Pipeline: Model API
** Pipeline: Declarati

localhost:8080/

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address: ×

Jenkins 2.204.2 Continue as admin [Save and Continue](#)

Getting Started

Instance Configuration

Jenkins URL:

✕

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.204.2

Not now

[Save and Finish](#)




localhost:8080/

Getting Started

Jenkins is almost ready!

Your Jenkins setup is complete, but some plugins require Jenkins to be restarted.

[Restart](#)

-  New Item
-  People
-  Build History
-  Manage Jenkins
-  Lockable Resources
-  Credentials
-  New View

Welcome to Jenkins!

Please [create new jobs](#) to get started.

Build Queue







No builds in the queue.

Build Executor Status

- 1 Idle
- 2 Idle

Enter an item name

» This field cannot be empty, please enter a valid name

-  **Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
-  **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
-  **Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
-  **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
-  **GitHub Organization**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.
-  **Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

Build Environment

- Use secret text(s) or file(s) ?
- Abort the build if it's stuck
- Add timestamps to the Console Output
- Inspect build log for published Gradle build scans
- With Ant ?

Build

Add build step ▾

- Execute Windows batch command**
- Execute shell
- GitHub PR: set 'pending' status
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit

Jenkins

Jenkins > **job1** >

- Back to Dashboard
- Status**
- Changes
- Workspace
- Build Now**
- Delete Project
- Configure
- Rename

Project job1

- [Workspace](#)
- [Recent Changes](#)

Permalinks

Build History [trend](#) ▾

find

- #1** Feb 5, 2020 10:24 AM

Task 1#####

1. Create new job or item
2. Enter an item name :Job1
3. Choose freestyle project
4. OK
5. configure the Job1
6. Choose Build section
7. Add build step
8. Execute shell or Execute windows batch command(on case of windows)
9. In command field type: echo "My first task in Jenkins"
10. Save
11. For executing the Job1, go to Job1 and click on "Build Now"

//Here Build the Job means executing the Job

12. Check the Out using Build History.

Task 2#####

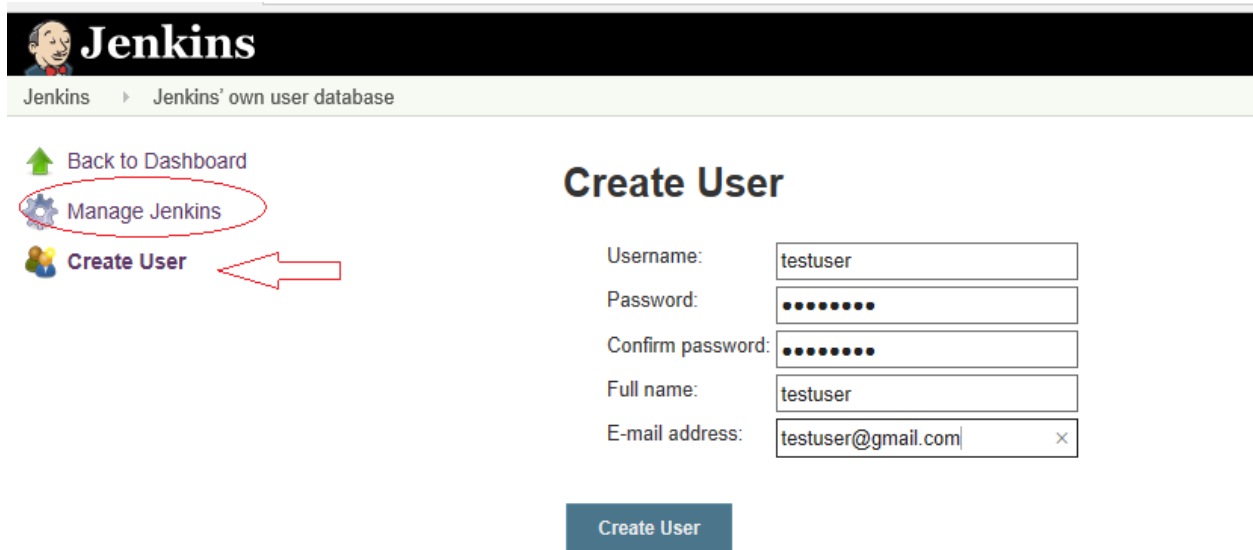
1. Create Job2
2. Freestyle Project
3. OK
4. Configure
5. Build
6. Add build step
7. Execute shell
8. Command: date>> /tmp/date.txt
9. Save
10. Build Now

// check the date.txt in tmp

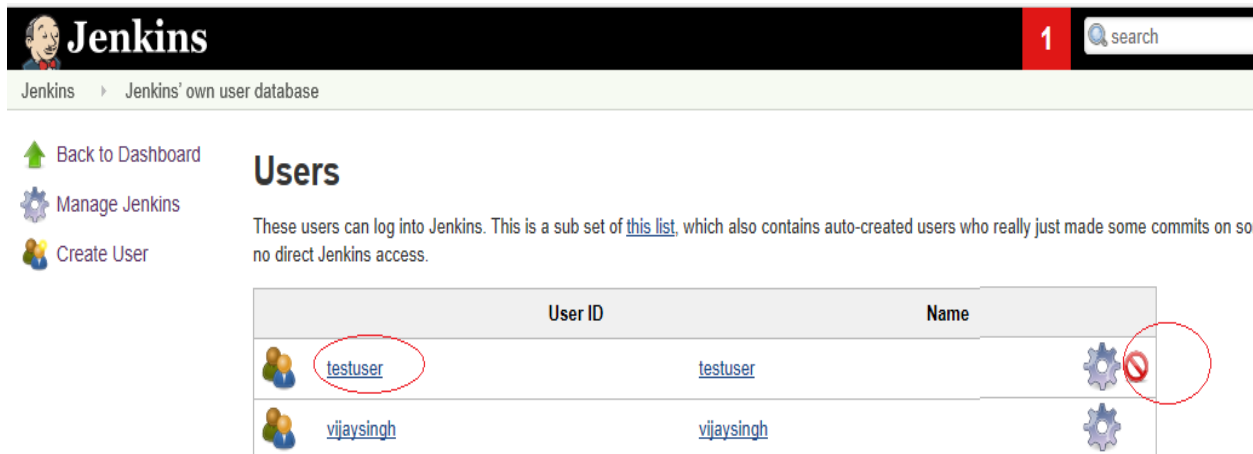
Task 3

//Create User






1. Manage Jenkins
2. Manage Users
3. Create User (create some users)



The screenshot shows the Jenkins 'Create User' page. The header includes the Jenkins logo and the text 'Jenkins' and 'Jenkins' own user database'. On the left sidebar, there are three navigation items: 'Back to Dashboard', 'Manage Jenkins' (circled in red), and 'Create User' (with a red arrow pointing to it). The main content area is titled 'Create User' and contains a form with the following fields: Username (testuser), Password (masked with dots), Confirm password (masked with dots), Full name (testuser), and E-mail address (testuser@gmail.com). A 'Create User' button is located below the form.



The screenshot shows the Jenkins 'Users' page. The header includes the Jenkins logo, the text 'Jenkins' and 'Jenkins' own user database', a red tab with the number '1', and a search bar. On the left sidebar, there are three navigation items: 'Back to Dashboard', 'Manage Jenkins', and 'Create User'. The main content area is titled 'Users' and contains a paragraph: 'These users can log into Jenkins. This is a sub set of [this list](#), which also contains auto-created users who really just made some commits on so no direct Jenkins access.' Below the text is a table with two columns: 'User ID' and 'Name'. The table contains two rows: one for 'testuser' and one for 'vijaysingh'. The 'testuser' row has a gear icon circled in red with a red 'X' over it, indicating that the user is disabled. The 'vijaysingh' row has a gear icon.

User ID	Name	
 testuser	testuser	 
 vijaysingh	vijaysingh	

Task 4

//security in Jenkins

1. Manage Jenkins
2. Configure Global Security
3. Authorization
4. Matrix-based Security or Project-based security
5. Add user if you required
6. Assign the privileges
7. Logout and log-in again with different user and explore.

Jenkins > Configure Global Security

Configure Global Security

Enable security Disable remember me

Access Control

Security Realm

Delegate to servlet container
 Jenkins' own user database
 Allow users to sign up
 LDAP

Authorization

Anyone can do anything
 Legacy mode
 Logged-in users can do anything
 Matrix-based security

User/group	Overall	Credentials		Agent			Job		Run		View		SCMLockable Resources									
	Administrator	Read	Delete	Manage Domains	View	Update	Build	Configure	Connect	Create	Delete	Disconnect	Build	Cancel	Configure	Create	Delete	Read	Tag	Reserve	Unlock	View
Anonymous Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

5. once it is installed go to Manage Jenkins
 6. Configure Systems
 7. Global properties->Audit trail
 8. Add Logger->log file
 9. Add Log Location: /tmp/jenkinsAudir.log
 10. Log file sizeMB: 25
 11. Log file count: 5
 12. Save
- // After executing any job you can check the log file /tmp/jenkinsAudit.log

The screenshot shows the Jenkins web interface. At the top left is the Jenkins logo. Below it, a breadcrumb trail shows 'Jenkins' and 'Plugin Manager', with 'Plugin Manager' circled in red. On the left sidebar, there are three links: 'Back to Dashboard' with a green arrow icon, 'Manage Jenkins' with a gear icon and circled in red, and 'Update Center' with a green puzzle piece icon. The main content area has four tabs: 'Updates', 'Available', 'Installed', and 'Advanced'. Below the tabs is a table with columns 'Install' and 'Name'. The 'Install' column has a dropdown arrow and a checked checkbox. The 'Name' column contains the text 'Audit Trail', which is circled in red. Below the table, there are two buttons: 'Install without restart' (circled in red) and 'Download now and install after restart'.

Jenkins **2**

Jenkins > Plugin Manager

Back to Dashboard

Manage Jenkins

Update Center

Updates Available Installed Advanced

Install ↓	Name
<input checked="" type="checkbox"/>	Audit Trail <small>Keep a log of who performed particular Jenkins operations, such as co</small>

Install without restart

Download now and install after restart

Add ▾

List of default health metrics for Folders

Audit Trail

Loggers

Log file

Log Location

Log File Size MB

Log File Count

Delete

Add Logger ▾

Advanced...

Log file

Log Location

Log File Size MB

Log File Count

Delete

Add Logger ▾

Task 6

//Notifications

1. Manage Jenkins
2. Configure Systems
3. Extended E-mail Notifications
4. SMTP server: smtp.gmail.com
5. Advanced
6. Check Use SMTP Authentication

- 7. Username : abc@gmail.com
- 8. Password : *****
- 9. Use SSL check
- 10. SMTP port: 465
- 11. Default recipient : abc@gmail.com
- 12. Save

Extended E-mail Notification

SMTP server

Default user E-mail suffix

Use SMTP Authentication

Advanced Email Properties

Use SSL

SMTP port

Charset

Additional accounts

Default Content Type

Use List-ID E-mail Header

Add 'Precedence: bulk' E-mail Header

Default Recipients

Task 7

// Notification for a Job

- 1. Select a Job
- 2. Configure
- 3. Post-build Actions
- 4. Editable Email Notification
- 5. Project recipient list: abc@gmail.com
- 6. Advanced
- 7. Trigger
- 8. Add a trigger(recipient list)
- 9. Save



Back to Dashboard

Status

Changes

Workspace

Build Now

Delete Project

Configure

Rename

Project job1

[Workspace](#)

[Recent Changes](#)

Permalinks

General Source Code Management Build Triggers Build Environment Build **Post-build Actions**

Execute Windows batch command

Command `dir`

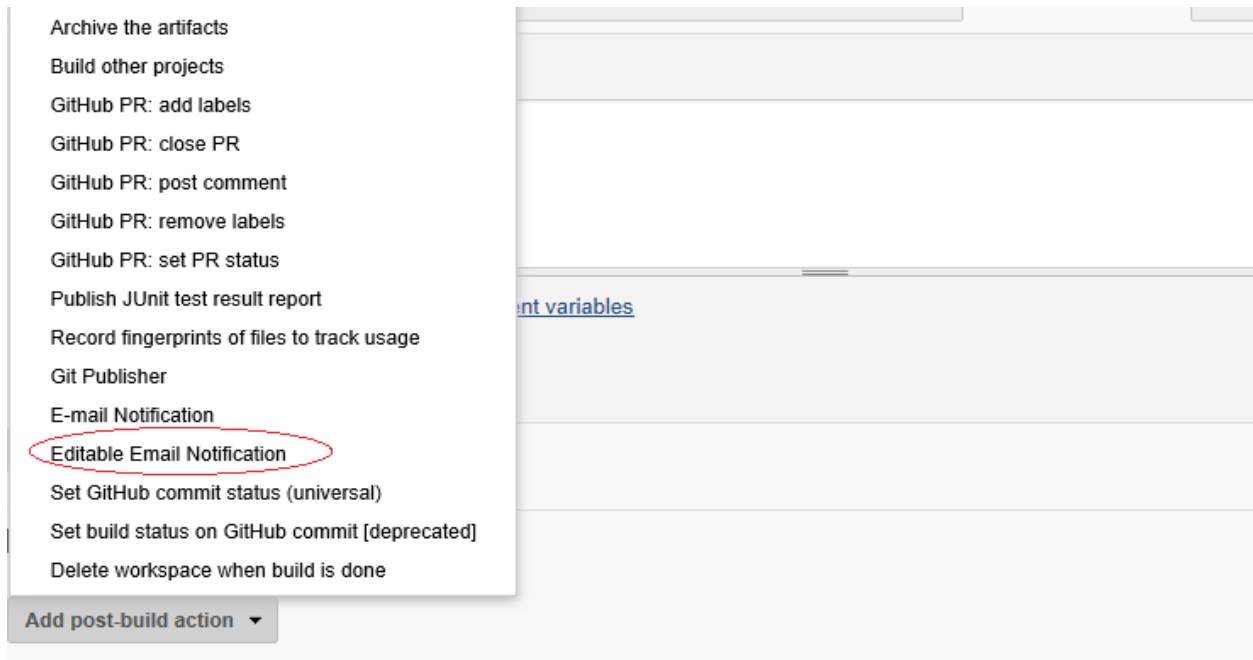
See [the list of available environment variables](#)

Advanced...

Add build step ▾

Post-build Actions

Add post-build action ▾



//Scheduling the Jobs(Three ways)

- a. Timer
- b. Poll SCM
- c. Pipeline

Task 8

//Timer

1. Choose any Job
2. Configure
3. Build Trigger
4. Build periodically

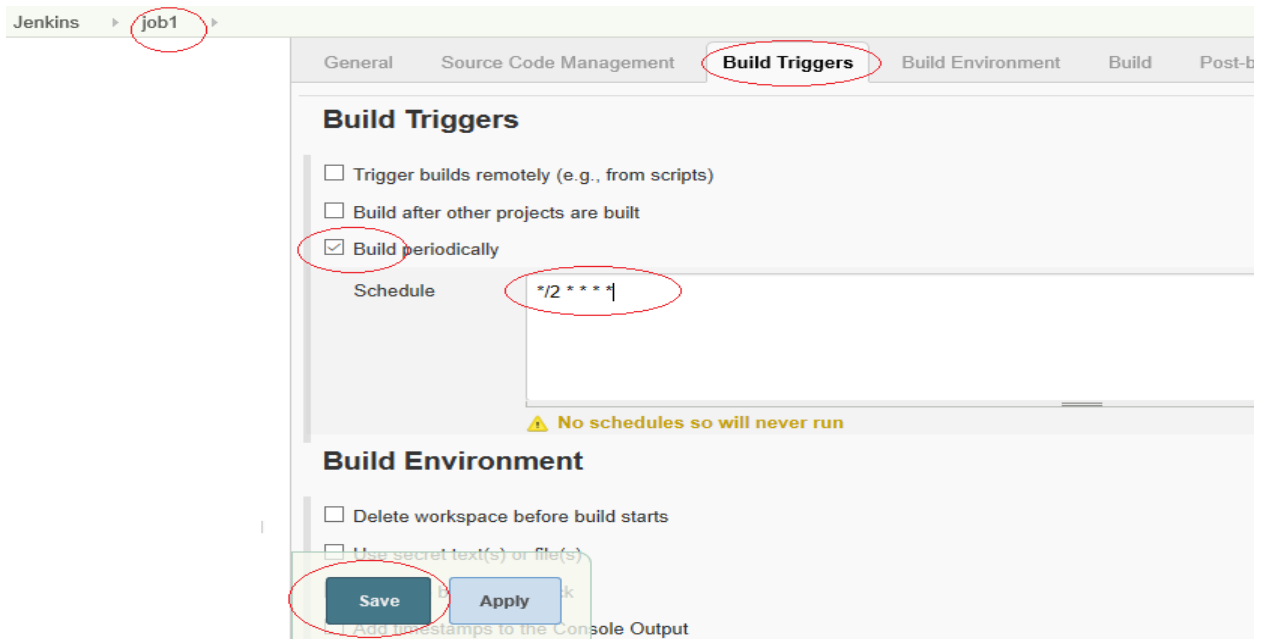
// little bit Learn about, what is cron job?

5. Type in schedule: */2 * * * *

// every two minutes it will execute the job

6. Save

7. Check the build history, in output console check that the job is started by timer, not by the user.



Task 9

//Poll SCM, pull the code from github and compile every two minutes or any changes occur then only it will compile.

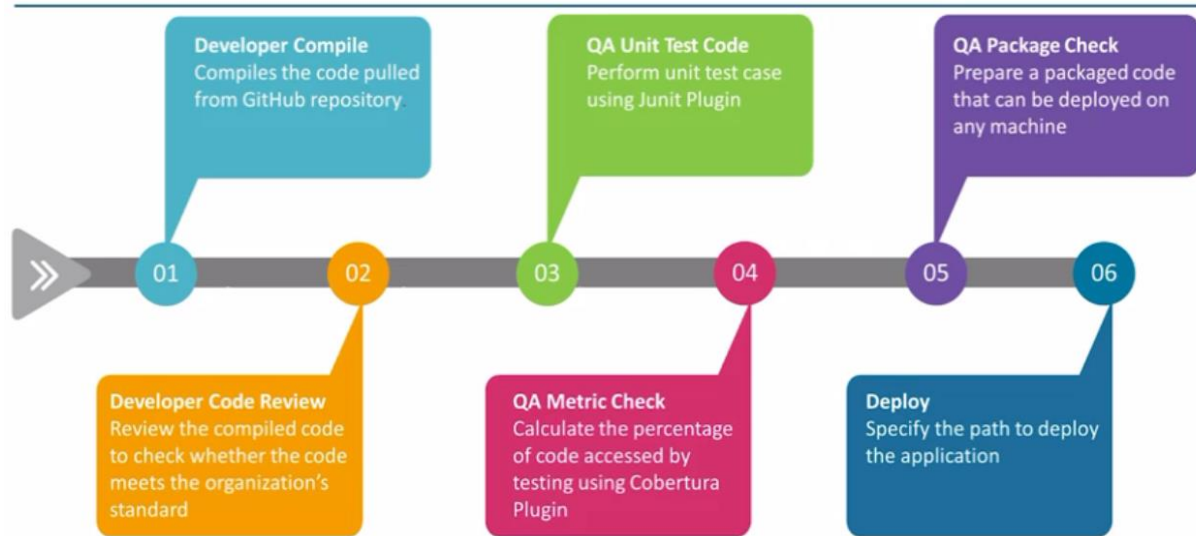
1. Create new Job (job_pollSCM_Demo)
2. Freestyle project
3. OK
4. configure
5. Source code management
6. git
7. Repositories
8. Repository URL: copy and paste Git Repository URL
- // any commits happens, then only it will build
9. Save
10. Configure
11. Build Trigger
12. check Poll SCM
13. Schedule: */2****
14. Save

//perform some commit on Github and check the reflection on jenkins

//check the workspace at /var/lib/jenkins/workspace/

cd /var/lib/jenkins/workspace/

Delivery Pipeline



Audit Trail Plugin

Keep a log of who performed particular Jenkins operations, such as configuring jobs. This plugin adds an `Audit Trail` section in the main Jenkins configuration page. Here you can configure log location and settings (file size and number of rotating log files), and a URI pattern for requests to be logged. The default options select most actions with significant effect such as creating/configuring/deleting jobs and views or delete/save-forever/start a build. The log is written to disk as configured and recent entries can also be viewed in the Manage / System Log section.

What is PMD?

It is a static rule-set based Java source code analyser that identifies potential problems in your source code. There are other static analysers like FindBugs.

What Is Static Analysis?

Static analysis is a method of debugging that is done by automatically examining the source code without having to execute the program. This provides developers with an understanding of their code base and helps ensure that it is compliant, safe, and secure.

What Is Static Code Analysis?

Static code analysis and static analysis are often used interchangeably, along with source code analysis.

Static code analysis addresses weaknesses in source code that might lead to vulnerabilities. Of course, this may also be achieved through manual source code reviews. But using automated tools is much more effective.

What Is Cobertura?

Cobertura is an open source tool that measures test coverage by instrumenting a code base and tracking the percentage of code that are executed as the test suite runs. It is based on jcoverage. In addition to identifying untested code and locating bugs, Cobertura can optimize code by flagging dead and unreachable code.

Cobertura monitors tests by instrumenting the bytecode with extra statements to log which lines are and are not being reached as the test suite executes. It then produces a report in HTML or XML that shows exactly which packages, classes, methods, and individual lines of code are not being tested.

What is a POM?

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects. Examples for this is the build directory, which is target; the source directory, which is src/main/java; the test source directory, which is src/test/java; and so on. When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal.

Some of the configuration that can be specified in the POM are the project dependencies, the plugins or goals that can be executed, the build profiles, and so on. Other information such as the project version, description, developers, mailing lists and such can also be specified.

Task 10

Delivery Pipeline for the project

#####

01(Compile)->02(Code Review)->03(Unit Test Code)->04(Metric Check)->05 (Package Check)->06(Deploy)

#####

01=>Compile: Compiles the code pulled from github repository.

02=>Code Review: Review the compiled code to check whether the code meets the organization's standard

03=> Unit Test Code: Perform unit test case using

Junit Plugin

04=> Metric Check: Calculate the percentage of code

accessed by testing using Cobertura Plugin

05=>Package Check: Prepare a packaged code that

can be deployed on any machine.

06=>Deploy: Specify the path to deploy the application.

Use the link: [edureka-git/DevOpsClassCodes](https://github.com/edureka-git/DevOpsClassCodes)

Maven(build automation tool)

Task 10.01

//compile job

1. Create Job "DevCompile"

2. Freestyle project.

3. ok

4. configure.

5. Source Code management.

6. git

7. Repository URL: <https://github.com/edureka-git/DevOpsClassCodes.git>

8. save

9. jenkins->manage jenkins->Global Tool Configuration

10.JDK

11.Add JDK

12.Name: myjava

13.Install automatically

14.version: java SE Development kit 8u152

15.check I agree

// you have to give oracle account details, if you don't have just create one.

16. Maven

17. Add Maven

18. Name: mymaven->check Install Automatically

19. Save

20. configure DevCompile job

21. Build->Add build step->invoke top-level Maven targets.

22. maven version->mymaven

23. Goals->compile.

24. Save.

25. Build Now.

26. Check the Output console.

Task 10.02

//Code review

1. create job "CodeReview".
2. Freestyle project.
3. ok.
4. configure->source code management.
5. git
6. repository.
7. Repository URL:<https://github.com/edureka-git/DevOpsClassCodes.git>
8. Build->add build step.
9. Invoke top-level Maven targets.
- 10.Maven Version: mymaven
- 11.Goals: -P metrics pmd;pmd
//learn about PMD plugin(program and mistake detector)
- 12.Save.
- 13.Build Now.
- 14.Check pmd.xml under workspace/target.

// xml report is not user friendly, to generate more user friendly trend reports follow the steps:
15. Manage jenkins->manage plugins.
16. Search plugin: pmd.
17. PMD install without restart.
18. configure CodeReview job.
19. Post build actions.
20. choose publish pmd analysis results.

//provide the path of xml file

21. PMD results: target/pmd.xml

22. save and build now.

23. check the analysis under PMD warnings.

Task 10.03

//Job for testing

1. Create ne job "UnitTest"

2. Freestyle project.

3. ok.

4. configure->source code management->git

5. Repository URL: <https://github.com/edureka-git/DevOpsClassCodes.git>

6. Build->Add build step->top-level maven targets.

7. Maven version: mymaven

8. Goals:test

9. save->Build Now

10.Build History->Check the console output.

11.Check the detailed reports at workspace->target->surefire-reports(in the xml files)

//for generating trend reports

12. configure UnitTest job

13. Post-build Actions->Publish JUnit test result reports

//if the option is not present in your Post-build Action, just install JUnit Plugin.

14. Test report XMLs: target/surefire-reports/*.xml

// provide the path of XML files.

15. Save->Build Now

16. Check the Test Results.

Task 10.04

1. Create a new job "MetricCheck"

2. Source code management->git

3. Repository URL: <https://github.com/edureka-git/DevOpsClassCodes.git>

4. Build->Add build step

5. Invoke top-level maven targets.

6. Maven Version: mymaven

7. Goals= cobertura:cobertura -Dcobertura.report.format=xml

8. Save

9. Build Now

10. check workspace->target->site->cobertura->coverage.xml

// Cobertura tool against your compiled classes to help you determine how well the unit testing and integration testing efforts have been, and can then be used to identify which parts of your Java program are lacking test coverage.

11. for generate trend reports

//jenkins->manage jenkins->plugin manager->search cobertura plugin-> check cobertura plugin->install without restart

12. Configure MetricCheck

13. Post-build Actions

14. Publish Cobertura Coverage Reports

15. Cobertura xml report pattern: target/site/cobertura/coverage.xml

16. Save

17. Build Now

18. Check the Coverage Reports in MetricCheck Job

Task 10.05

1. Create a Job "Package"
2. Freestyle project->Ok
3. Configure->Source Code Management->git
4. Repository URL: <https://github.com/edureka-git/DevOpsClassCodes.git>
5. Build->Add build step->Top level Maven targets
6. Maven version : mymaven
7. Goals:package
8. Save
9. Build Now
- 10.//var file created under /var/lib/jenkins/workspace/Package/Target/

Task 10.06

//Create pipe-line

1. Go to Task 10.01 Compile Job.
2. Configure.
3. Post-build Actions->Add post-build actions.
4. Build other projects.
5. Project to Build: CodeReview.
6. Save.
7. Go to CodeReview project->configure.
8. Build Triggers.

9. check: Build after other projects are build.
 10. Project to watch : DevCompile.
 11. Post-build actions.
 12. Add post-build actions.
 13. Build other projects.
 14. projects to build: UnitTest
 15. Save.
 16. Go to UnitTest.
 17. Configure.
 18. Build Trigger-> Build after other projects are build.
 19. Project to Watch: CodeReview
 20. Post-Build Actions.
 21. Build Other Projects.
 22. MetricCheck.
 23. Save.
 24. Go to MetricCheck->Configure.
 25. Post-Build Actions.
 26. Projects to build: Package
 27. Build Triggers
 28. Build after other projects are build.
 29. Project to watch: MetricCheck.
 30. //No post-build.
- // Start first Job.

// To install pipeline plugin

//manage jenkins->manage plugins.

//search build pipeline plugin

//check build pipeline, install without restart.

//after installation click new view at the top beside All, choose Build pipeline view.give the name dev_pipeline

//select initial job: DevCompile

Apache Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

Common Maven Commands

Here is a list of common Maven commands plus a description of what they do. Please note, that even if a Maven command is shown on multiple lines in the table low, it is to be considered a single command line when typed into a windows command line or linux shell.

Maven Command	Description
mvn --version	Prints out the version of Maven you are running.
mvn clean	Clears the target directory into which Maven normally builds your project.
mvn package	Builds the project and packages the resulting JAR file into the target directory.
mvn package -Dmaven.test.skip=true	Builds the project and packages the resulting JAR file into the target directory - without running the unit tests during the build.
mvn clean package	Clears the target directory and Builds the project and packages the resulting JAR file into the target directory.

<code>mvn clean package -Dmaven.test.skip=true</code>	Clears the target directory and builds the project and packages the resulting JAR file into the target directory - without running the unit tests during the build.
<code>mvn verify</code>	Runs all integration tests found in the project.
<code>mvn clean verify</code>	Cleans the target directory, and runs all integration tests found in the project.
<code>mvn install</code>	Builds the project described by your Maven POM file and installs the resulting artifact (JAR) into your local Maven repository
<code>mvn install -Dmaven.test.skip=true</code>	Builds the project described by your Maven POM file without running unit tests, and installs the resulting artifact (JAR) into your local Maven repository
<code>mvn clean install</code>	Clears the target directory and builds the project described by your Maven POM file and installs the resulting artifact (JAR) into your local Maven repository
<code>mvn clean install -Dmaven.test.skip=true</code>	Clears the target directory and builds the project described by your Maven POM file without running unit tests, and installs the resulting artifact (JAR) into your local Maven repository
<code>mvn dependency:copy-dependencies</code>	Copies dependencies from remote Maven repositories to your local Maven repository.
<code>mvn clean dependency:copy-dependencies</code>	Cleans project and copies dependencies from remote Maven repositories to your local Maven repository.
<code>mvn clean dependency:copy-dependencies package</code>	Cleans project, copies dependencies from remote Maven repositories to your local Maven repository and packages your project.
<code>mvn dependency:tree</code>	Prints out the dependency tree for your project - based on the dependencies configured in the pom.xml file.
<code>mvn dependency:tree -Dverbose</code>	Prints out the dependency tree for your project - based on the dependencies configured in the pom.xml file. Includes repeated, transitive dependencies.
<code>mvn dependency:tree -Dincludes=com.fasterxml.jackson.core</code>	Prints out the dependencies from your project which depend on the com.fasterxml.jackson.core artifact.
<code>mvn dependency:tree -Dverbose -Dincludes=com.fasterxml.jackson.core</code>	Prints out the dependencies from your project which depend on the com.fasterxml.jackson.core artifact. Includes repeated, transitive dependencies.
<code>mvn dependency:build-classpath</code>	Prints out the classpath needed to run your project (application) based on the dependencies configured in the pom.xml file.

Unit-03

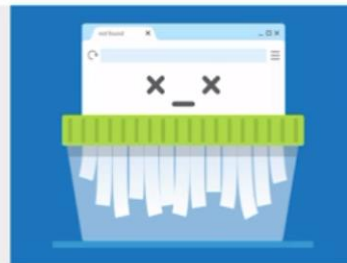
Continuous Testing, learn and Install Selenium, create test cases in Selenium, Integrate Selenium with Jenkins, Continuous Deployment, Install and configure puppet, understand master-slave architecture of puppet.

What Is Testing?



“ Identifying the mistakes or bugs in the software or an application ” .

“ Verifying whether the application works as expected or not ” .

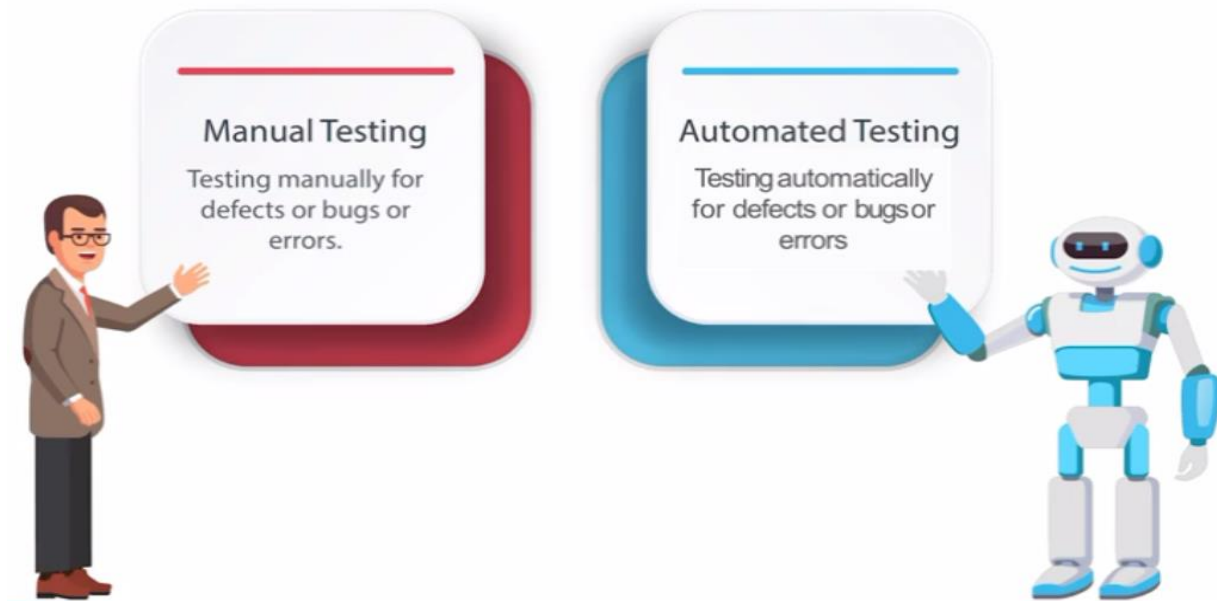


Why Testing?

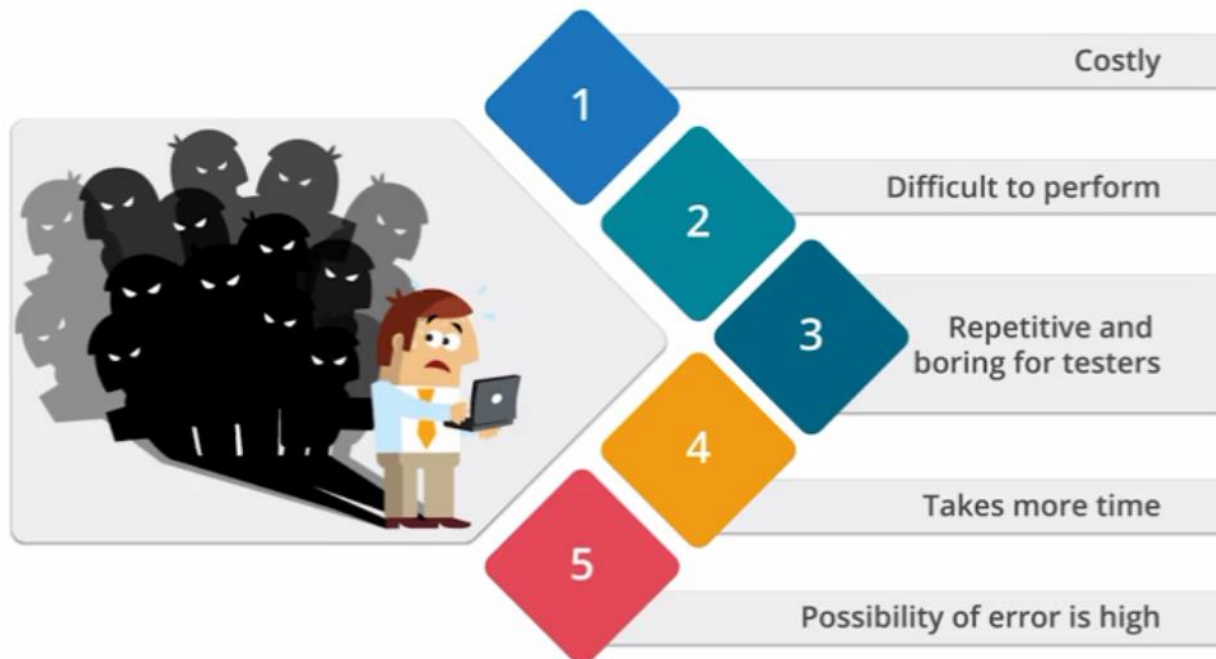
- Cost of fixing the bug is larger if testing is not done in early stage
- To produce good quality product
- To make software application defect free
- To check if behaviour of application is same in development and production environment



Types of Testing



Problems with manual testing



Automated Testing

“It is performed by using automation tools to run test cases and like Regression testing it is also used to check an application from Load, Performance and Stress point of view”.



Benefits of Automation Testing







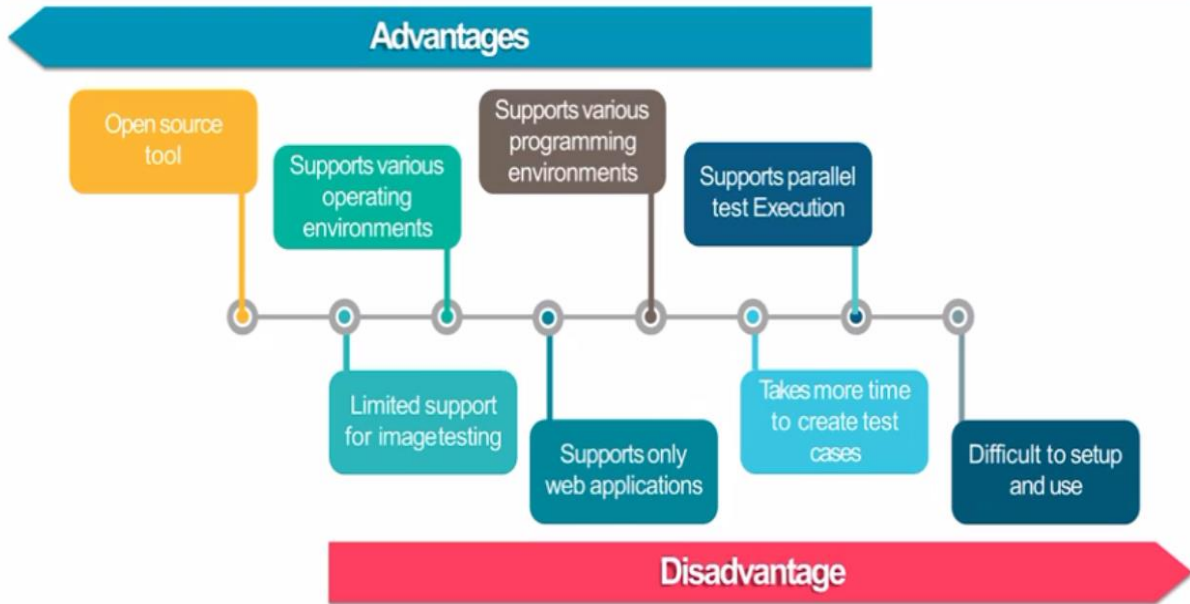
What is Selenium?

- **Selenium** is a tool which is used to automate web applications across different platforms using different programming languages
- It is **open source** and mainly used for functional and regression testing
- Known for better performance and execution speed
- Best tool for testing web applications

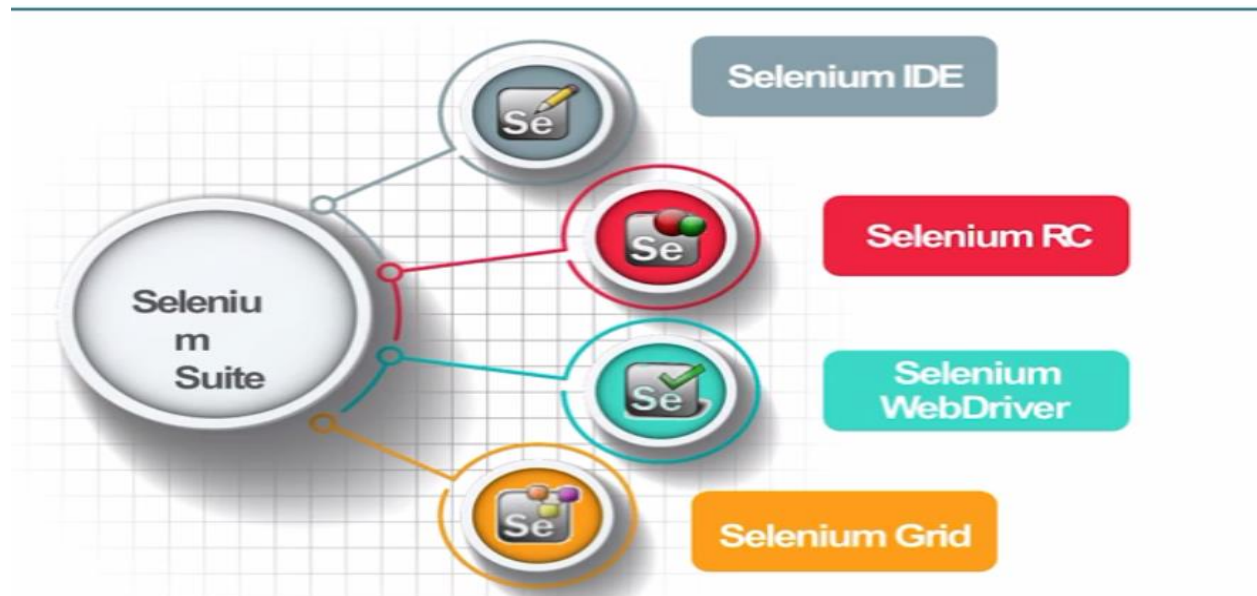
Selenium support the following:

Programming Languages	Operating Systems	Browsers
 Java	 Windows	 Internet Explorer
 python	 Linux	 Firefox
 JavaScript	 Android	 Google chrome
 php	 Apple	 Safari
 Perl	 iOS	 Opera
 Ruby		

Advantages & Disadvantages Of Selenium

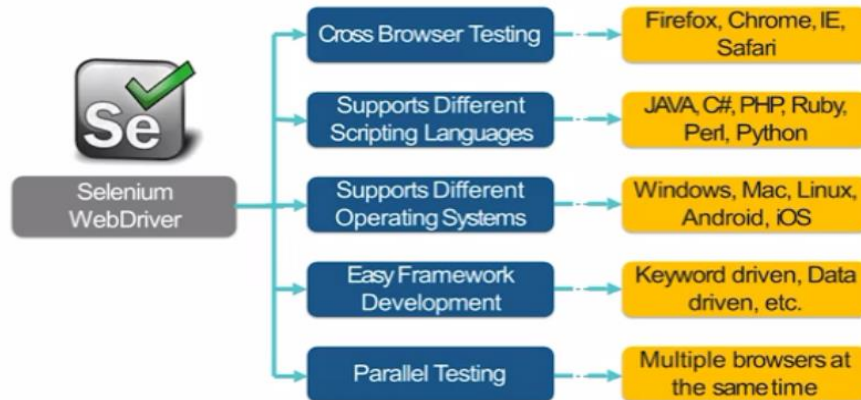


Components of Selenium Suite



Selenium WebDriver

- **Selenium WebDriver** is a tool used to automate testing for web application
- It allows us to execute tests against different browsers like Firefox, Chrome, IE & Safari
- **Selenium WebDriver** eliminated the use of Selenium Server thus making it to work faster than RC



Functional Testing

Testing each and every function inside the application to make sure that it is working as per the requirement



Regression Testing

Re-testing all the functionality when a new feature is added in the application to make sure all functions work good with the new feature





Selenium

- Open Source
- Hardware resource consumption is low
- Community Support
- Supports Windows, Linux, Mac



Test Complete

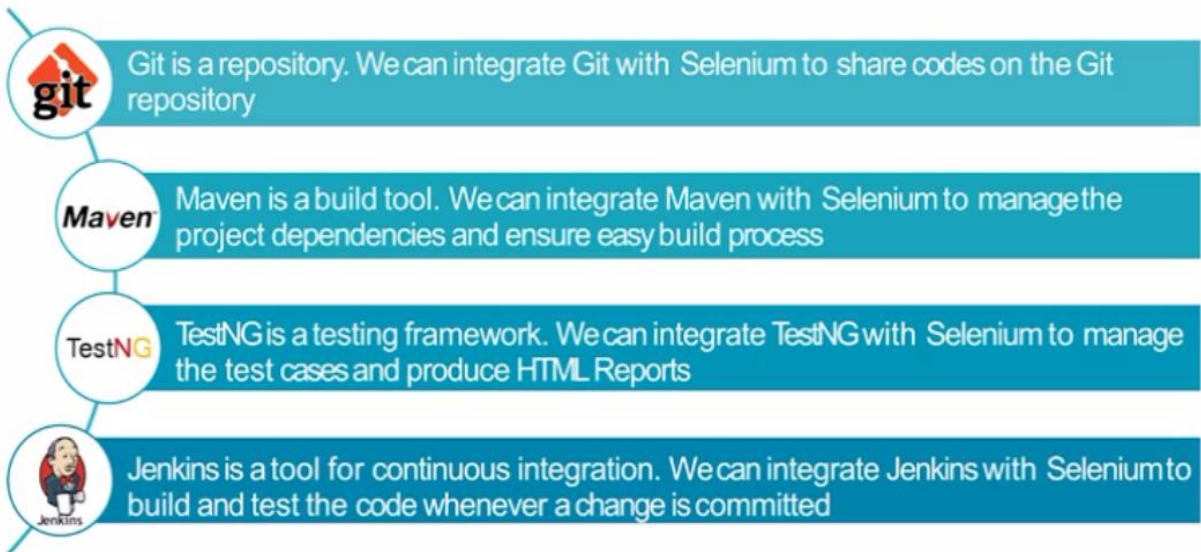
- License Required
- Hardware resource consumption is high
- Small User Community
- Supports Windows 7, Windows Vista, Windows Server 2008 or later OS

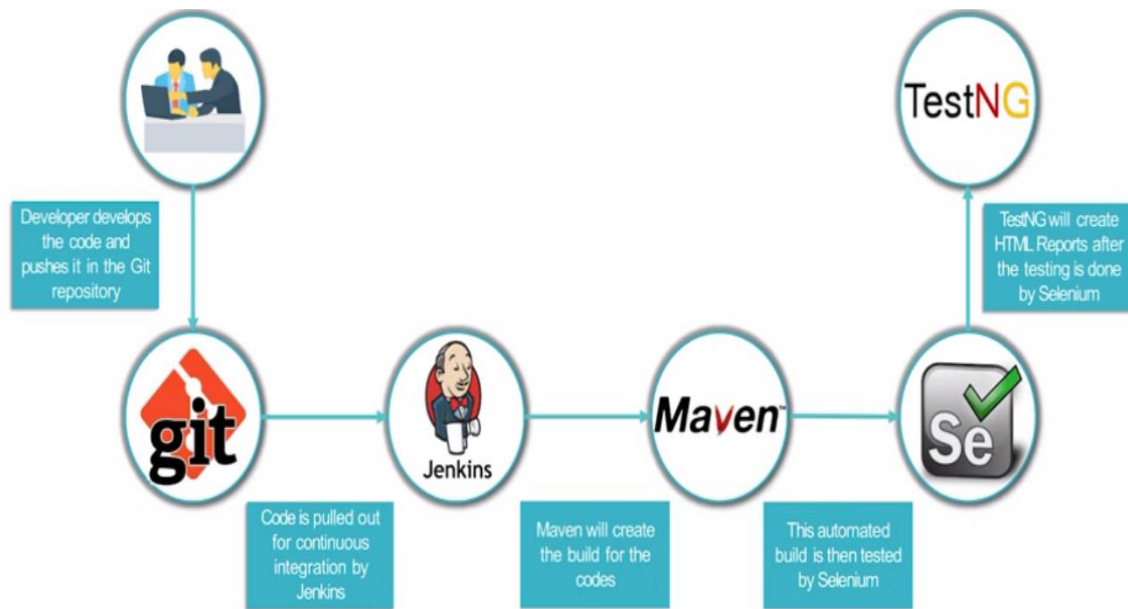


IBM RFT






- License Required
- Hardware resource consumption is high
- Dedicated IBM Support
- Supports Only Windows

Tools that can be integrated with Selenium

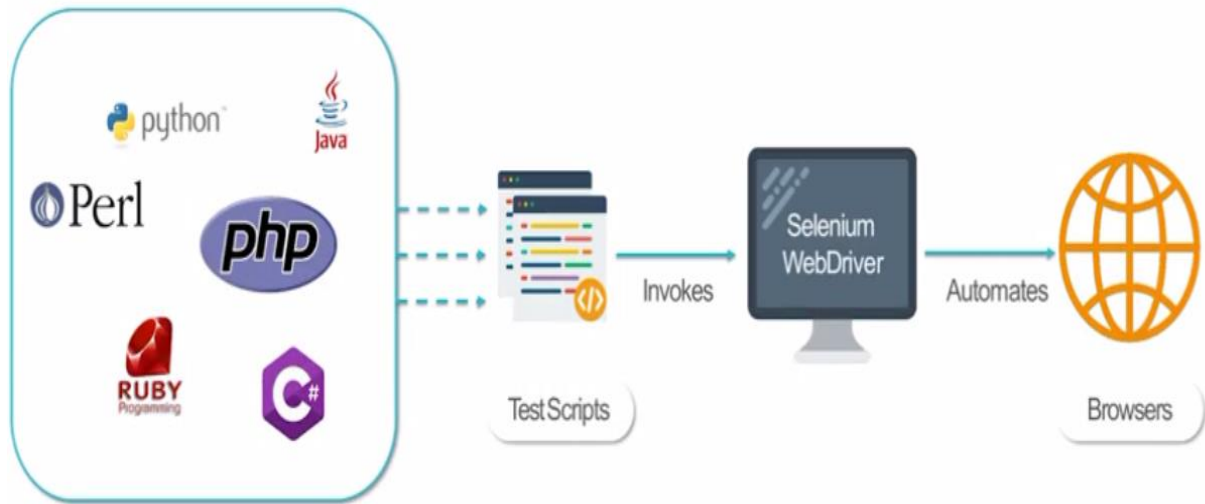




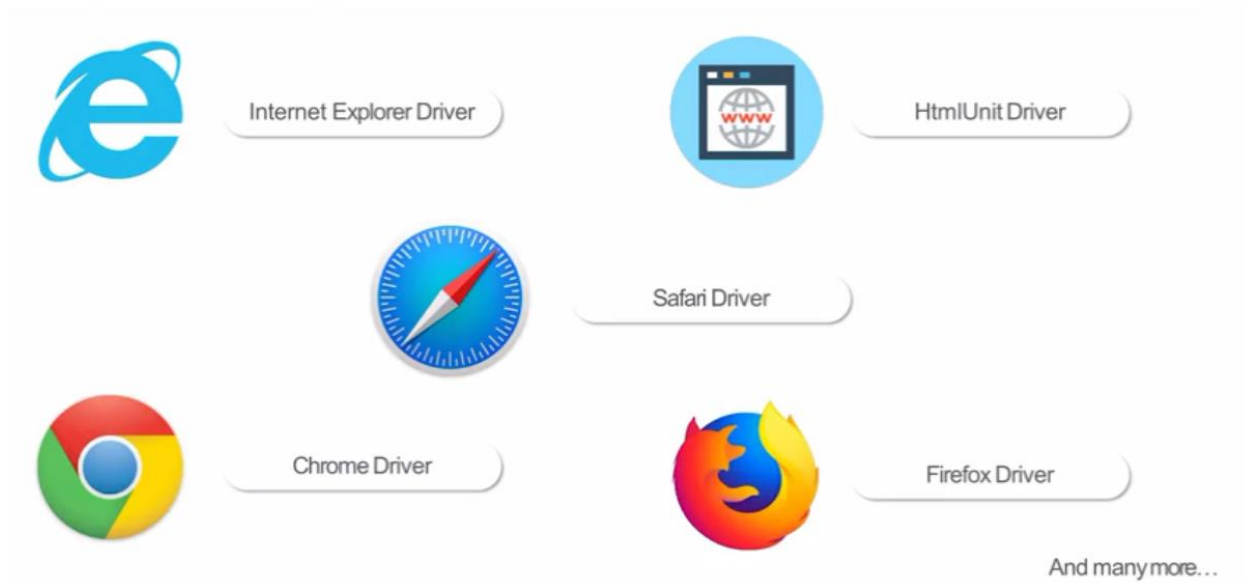
Selenium WebDriver

-  Selenium WebDriver is a programming interface to create and execute testcases
-  Testcases are created and executed using Elements locators/ Object locators/ WebDriver methods
-  Selenium WebDriver has only a programming interface; not IDE
-  Fast as it interacts with browser directly; RCneeds RCserver to interact with browser
-  Each browser has its own browser driver on which the application under the test runs. Selenium WebDriver makes direct calls to the browser

Working of Selenium WebDriver



Types of Selenium WebDriver



What is TestNG in Selenium?

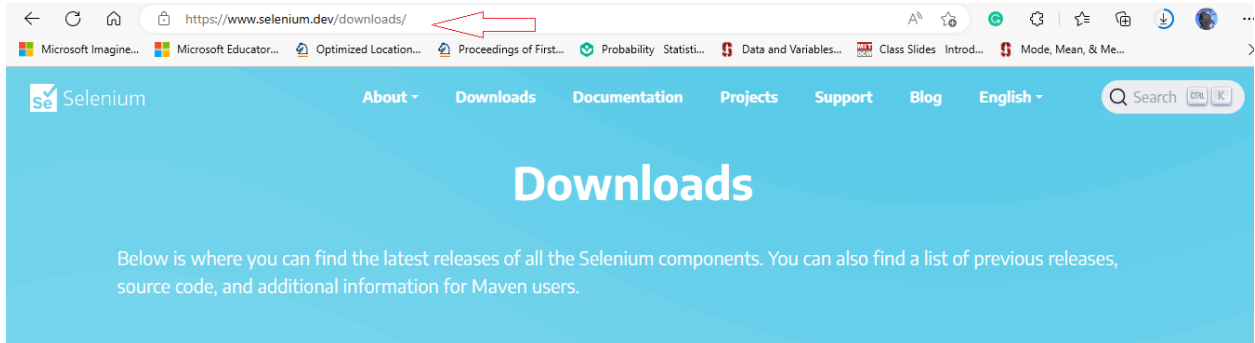
TestNG is an open-source testing framework where NG stands for 'Next Generation.' It is architected to simplify a broad range of testing needs starting from unit testing to integrated system testing. Initially, both JUnit and TestNG were designed solely for unit testing. TestNG is inspired by JUnit Java platform and NUnit .NET platform, and some new functionalities were introduced in TestNG, making it more powerful and easy to use than the JUnit testing framework.

Advantages of TestNG over Junit

- TestNG Annotations are used to create test cases easily.
- Test cases can be 'grouped,' 'prioritized,' and 'executed' more efficiently.
- It supports parameterization.
- It supports data-driven testing using Data Providers.
- It can generate HTML test reports of the results representing: the number of test cases runs, the number of test cases failed, the number of test cases skipped
- It effortlessly supports integration with various other tools and plugins like Eclipse IDE and built automation tools like Ant and Maven.
- It supports parallel execution.
- Logs can be generated.
- In TestNG, there is no need to state `@AfterClass` and `@BeforeClass` in a project, which is present in JUnit.
- You can specify any test method name in TestNG as method's name constraint is not present in TestNG like it is in JUnit.

Selenium

1. Download and install eclipse IDE for java developer.
2. Download selenium server standalone jar file and keep in a folder.

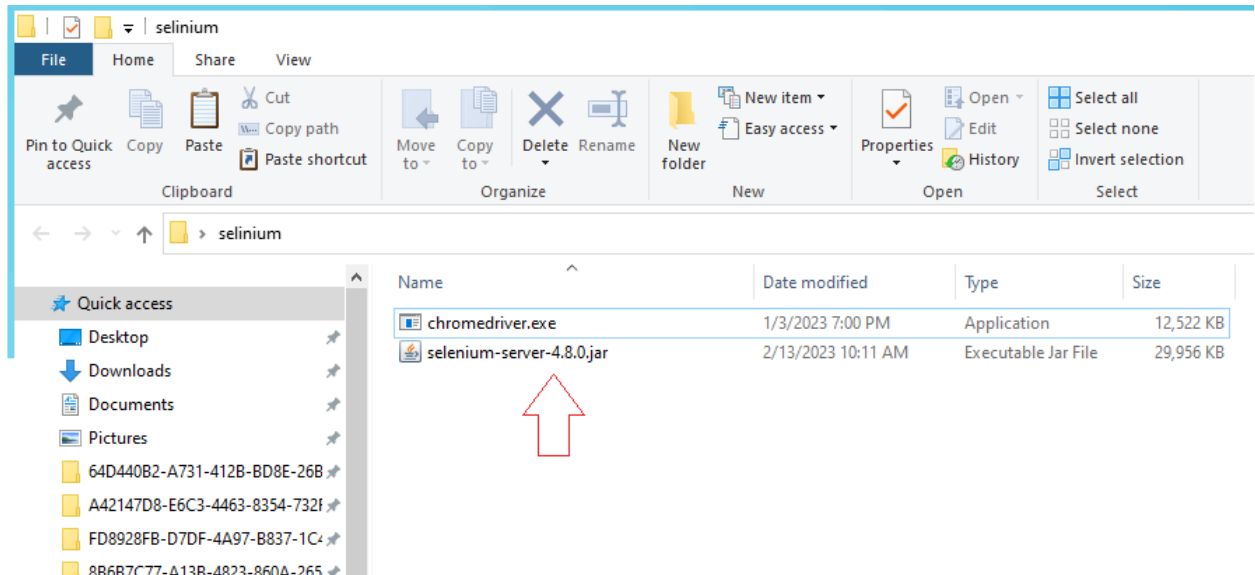


Selenium Server (Grid)

The Selenium Server is needed in order to run Remote Selenium WebDriver (Grid).

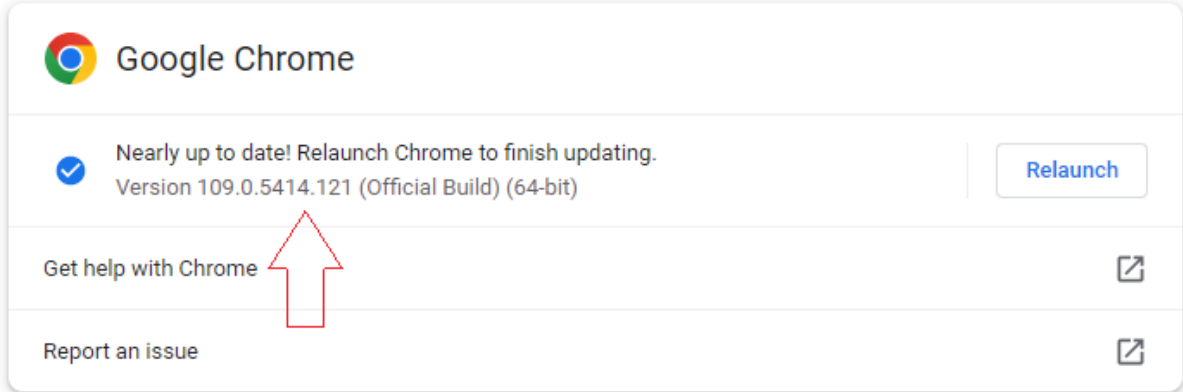
Latest stable version [4.8.0](#)

To use the Selenium Server in a Grid configuration see the [documentation](#).




3. Download chromedriver.exe and keep in the same folder.(before downloading driver check chrome version using chrome help->about google chrome)


About Chrome

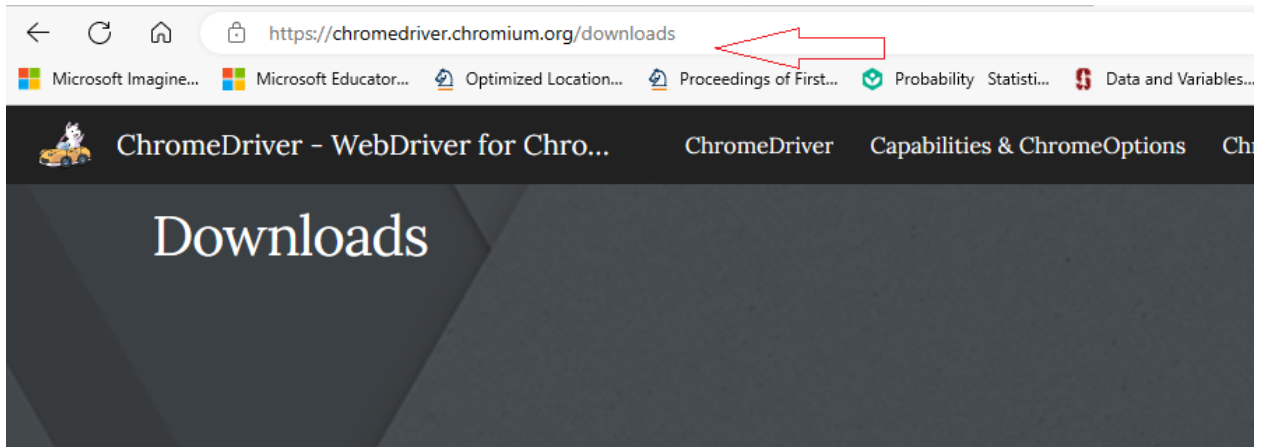


Google Chrome

Nearly up to date! Relaunch Chrome to finish updating.
Version 109.0.5414.121 (Official Build) (64-bit) [Relaunch](#)

[Get help with Chrome](#) 

[Report an issue](#) 



← ↻ 🏠 <https://chromedriver.chromium.org/downloads>

Microsoft Imagine... Microsoft Educator... Optimized Location... Proceedings of First... Probability Statisti... Data and Variables...

ChromeDriver - WebDriver for Chrome... ChromeDriver Capabilities & ChromeOptions Ch...

Downloads

Current Releases

- If you are using Chrome version 111, please download [ChromeDriver 111.0.5563.19](#)
- If you are using Chrome version 110, please download [ChromeDriver 110.0.5481.77](#)
- If you are using Chrome version 109, please download [ChromeDriver 109.0.5414.74](#)
- For older version of Chrome, please see below for the version of ChromeDriver that supports it.

If you are using Chrome from Dev or Canary channel, please following instructions on the [ChromeDriver Canary](#) page.

For more information on selecting the right version of ChromeDriver, please see the [Version Selection](#) page.

ChromeDriver 111.0.5563.19

4. Open eclipse then file->new->java project (name the project as SeleniumDemo) ->Finish.
5. Right click on SeleniumDemo->new->Package (name the package as seleniumScript)
6. Right click on seleniumScript->new->class (name the class as WorkingWithChrome)

```

package seleniumScript;

import org.openqa.selenium.chrome.ChromeDriver;

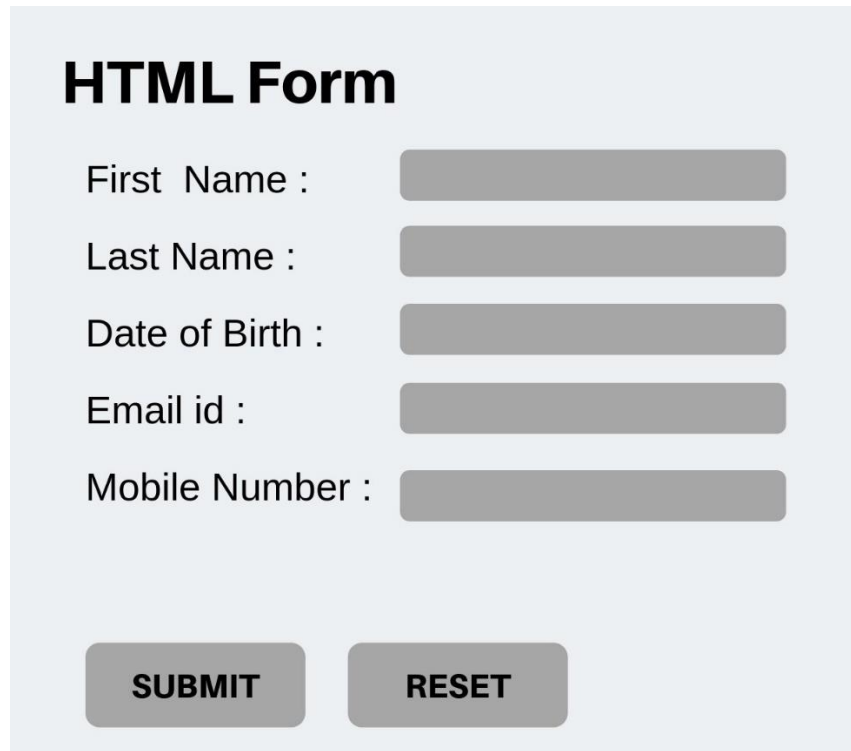
public class WorkingWithChrome {
    ChromeDriver driver;
    String url="https://www.google.com/";
    public void invokeBrowser(){

        System.setProperty("webdriver.chrome.driver", "C:\\Users\\Dell\\Desktop\\
        \\selenium\\chromedriver.exe");
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get(url);
    }
    public void getTitle(){
        String titleofthepage=driver.getTitle();
        System.out.println("Title of the page::::"+titleofthepage);
    }
    public void closeBrowser(){
        //driver.close();
        driver.quit();
    }
    public static void main(String[] args) {
        WorkingWithChrome wc=new WorkingWithChrome();
        wc.invokeBrowser();
        wc.getTitle();
        wc.closeBrowser();
    }
}

```

- Right click on code and run as java application.

1. Create the following form in HTML using id or name attribute.



HTML Form

First Name :

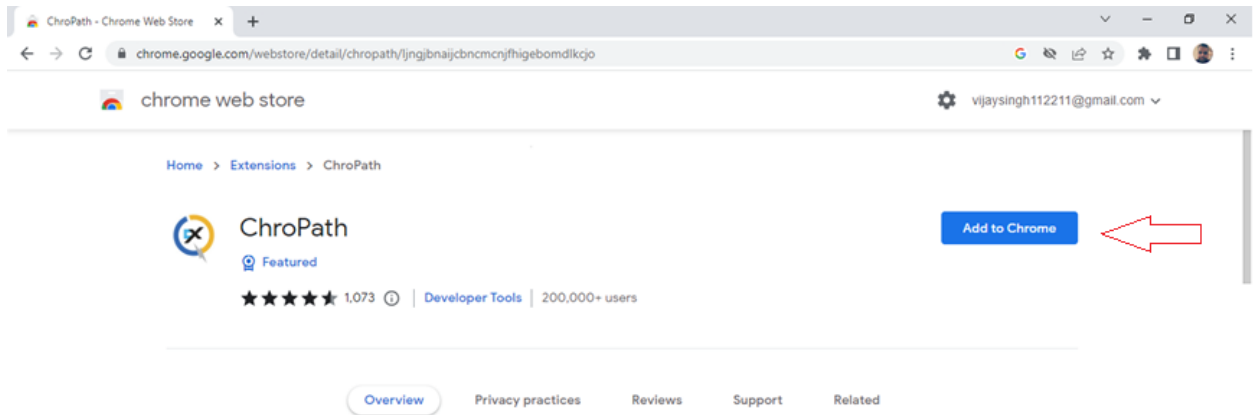
Last Name :

Date of Birth :

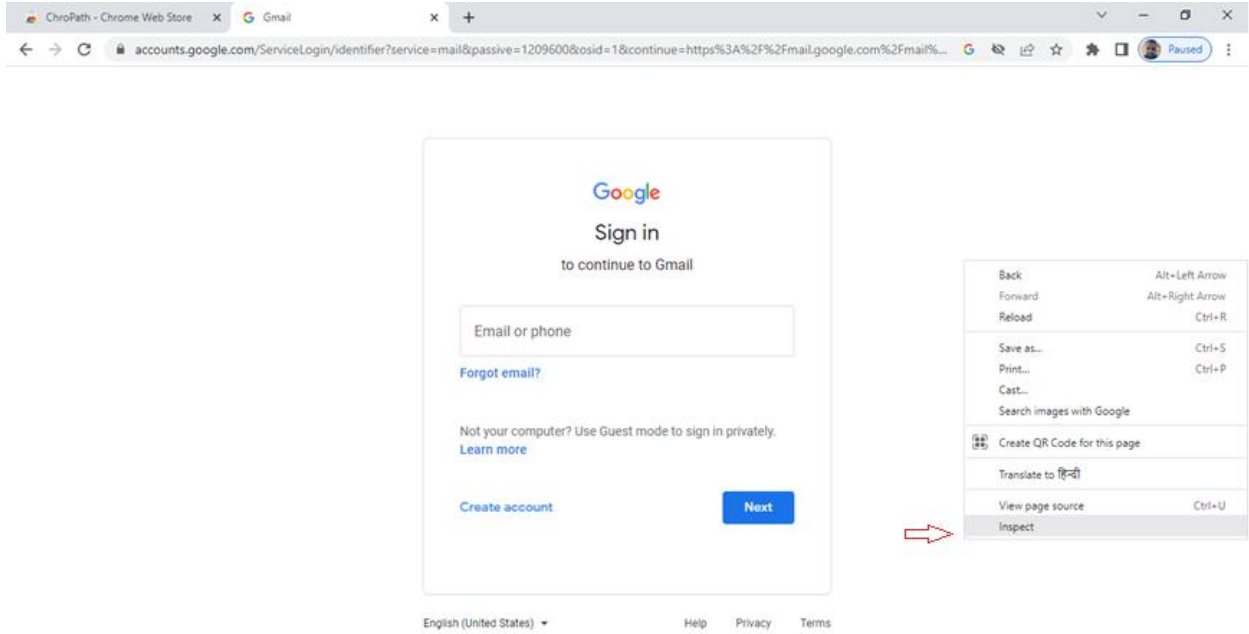
Email id :

Mobile Number :

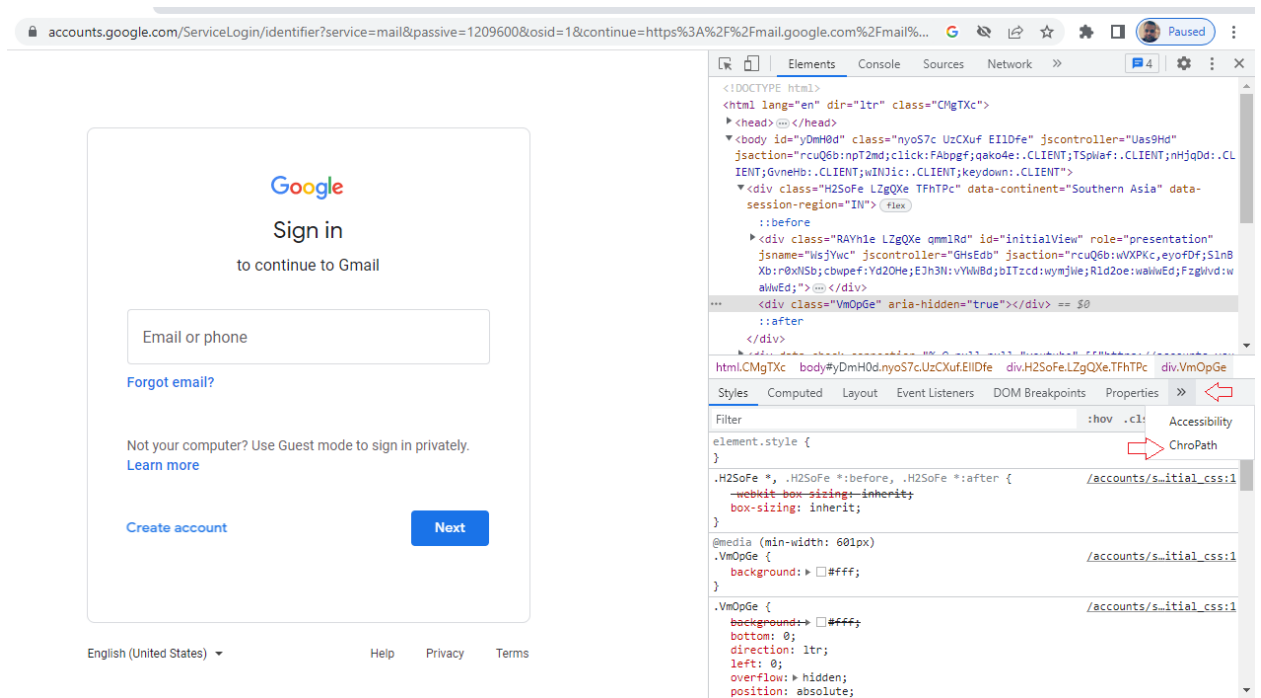
2. Write a selenium code using java or python to fill all the details in the form and click the submit button.
3. Download and install “ChroPath” chrome extension using chrome web store.

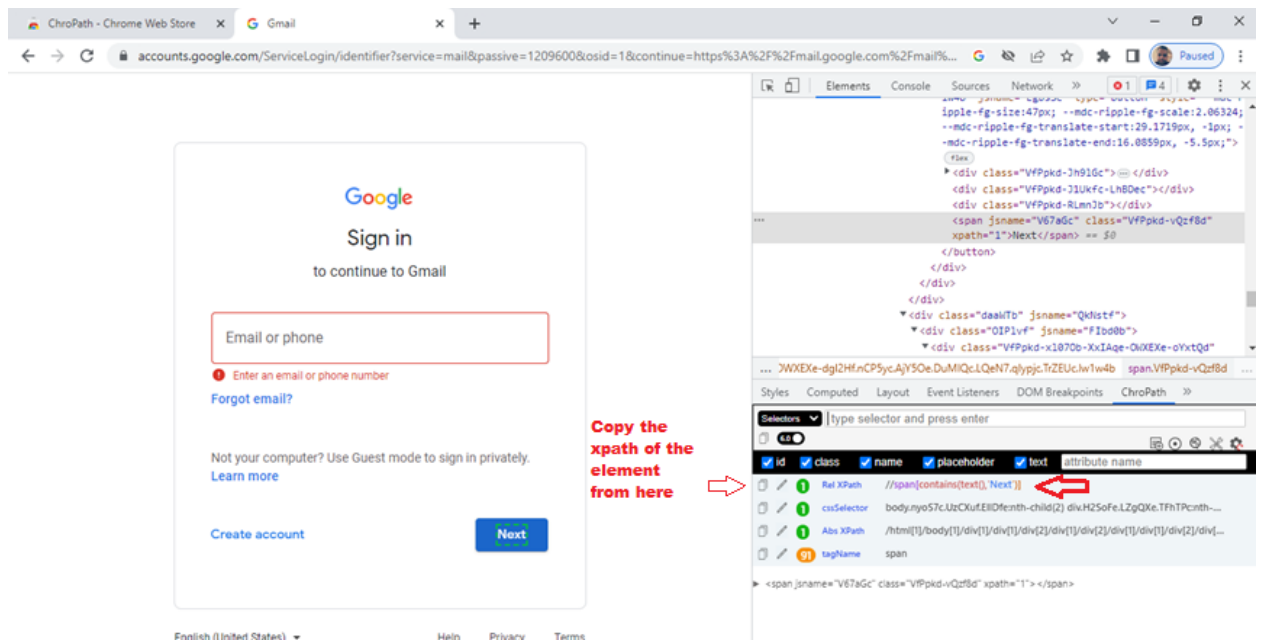
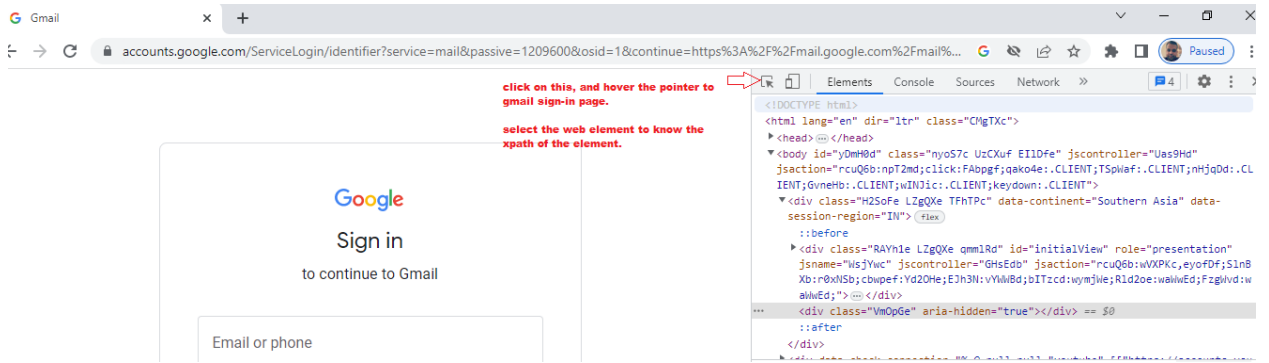


4. Open “gmail” login page then right click and inspect.



5. Use the “chroPATH” tab in the following way:





Example of using Xpath

```
driver.findElement(By.xpath("//input[@id='username-firstName']")).sendKeys("Your-Name");
```

- Demonstrate the gmail login using xpath.

Creation of TestNG test case in Selenium steps:

- Open eclipse->help->eclipse marketplace->find (TestNG)->install. (if already installed TestNG in eclipse ignore this step)
- Create a folder "TestNG_demo".
- Create a xml file with the following contents, and keep in the Test_NG folder.

```
<?xml version="1.0" encoding="UTF-8"?>

<suite name="Demo suite 1" verbose="1" >

  <test name="Demo test 1" >

    <classes>

      <class name="com.techbeamers.TestA"/>

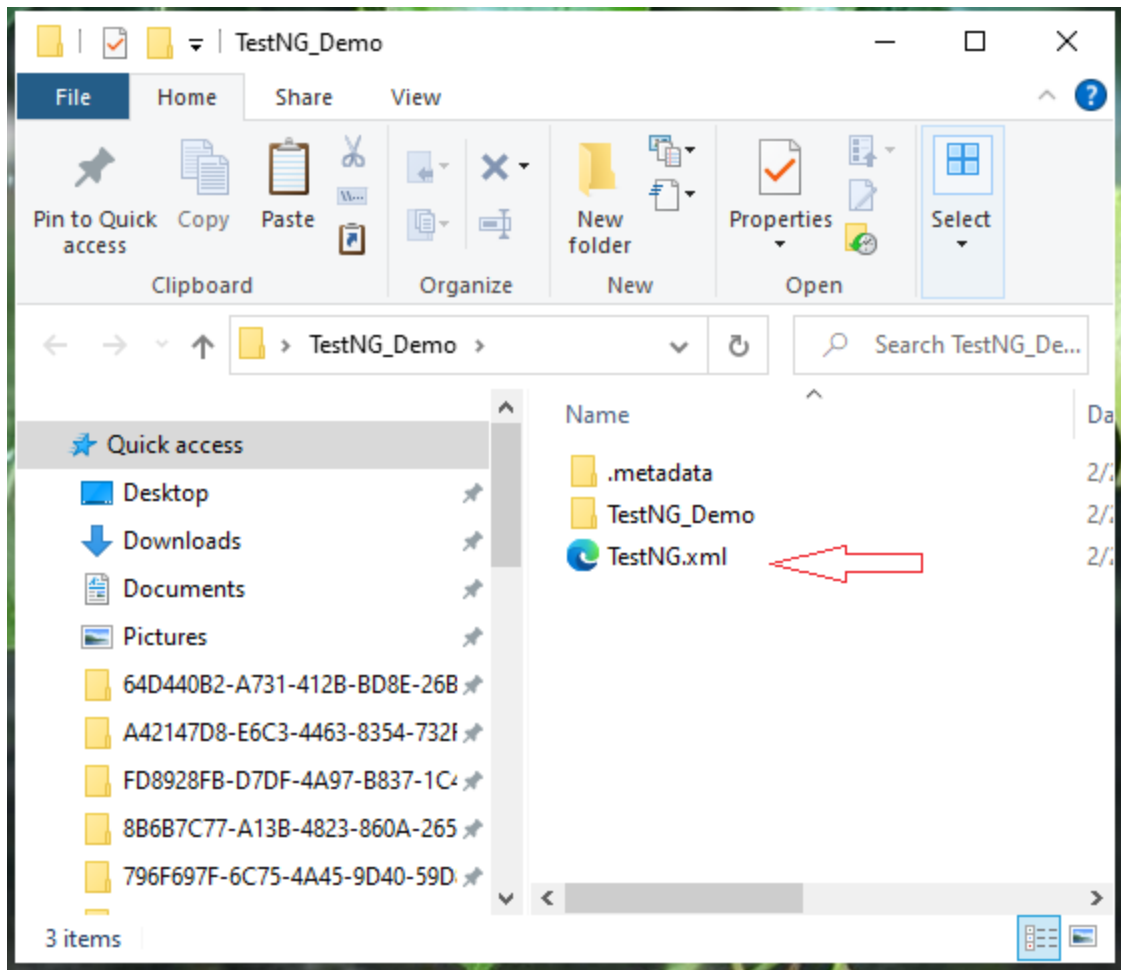
      <class name="com.techbeamers.TestB"/>

      <class name="com.techbeamers.TestC"/>

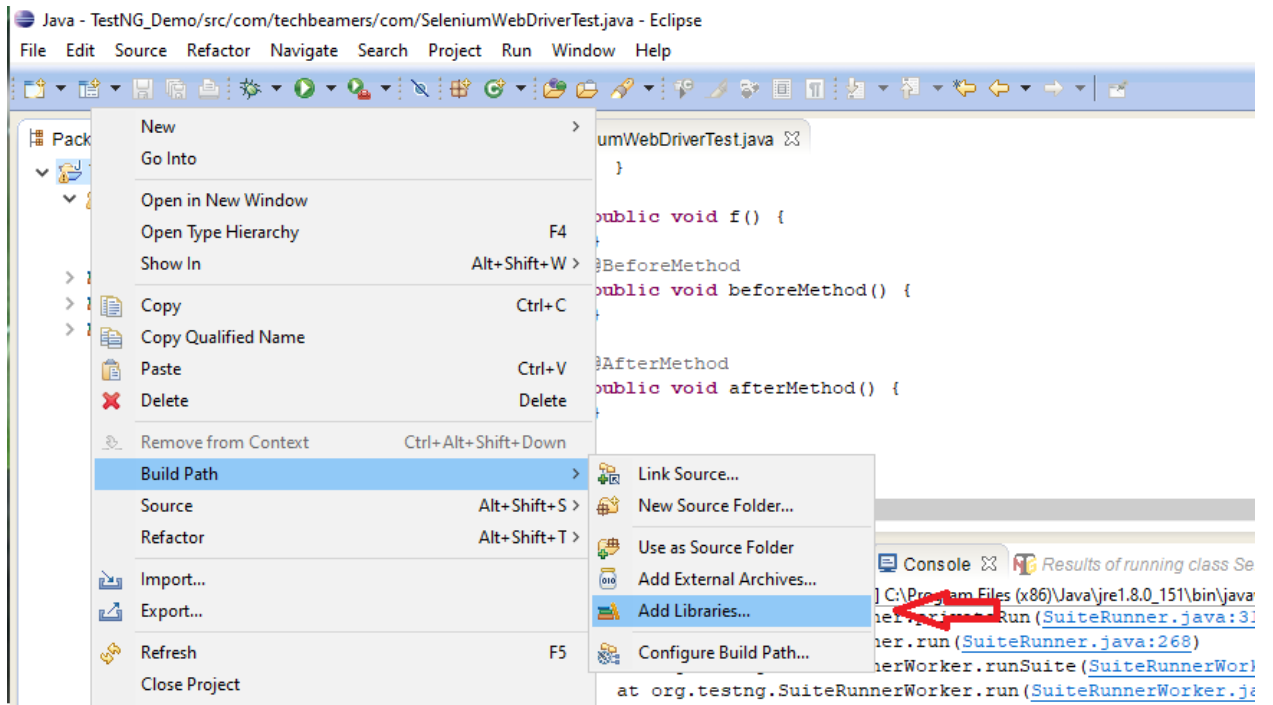
    </classes>

  </test>

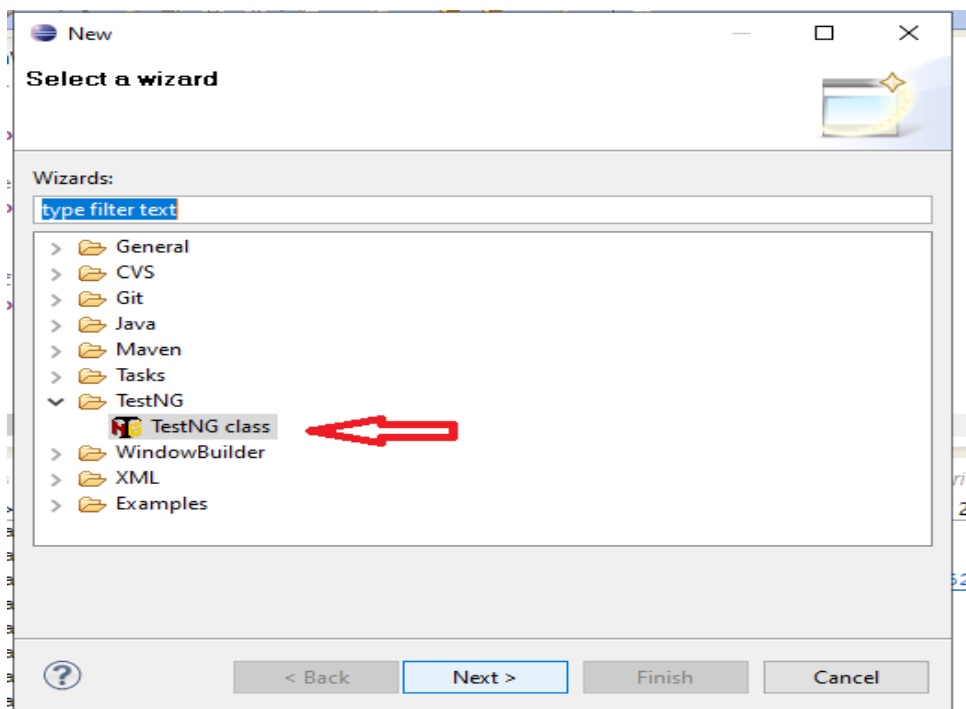
</suite>
```

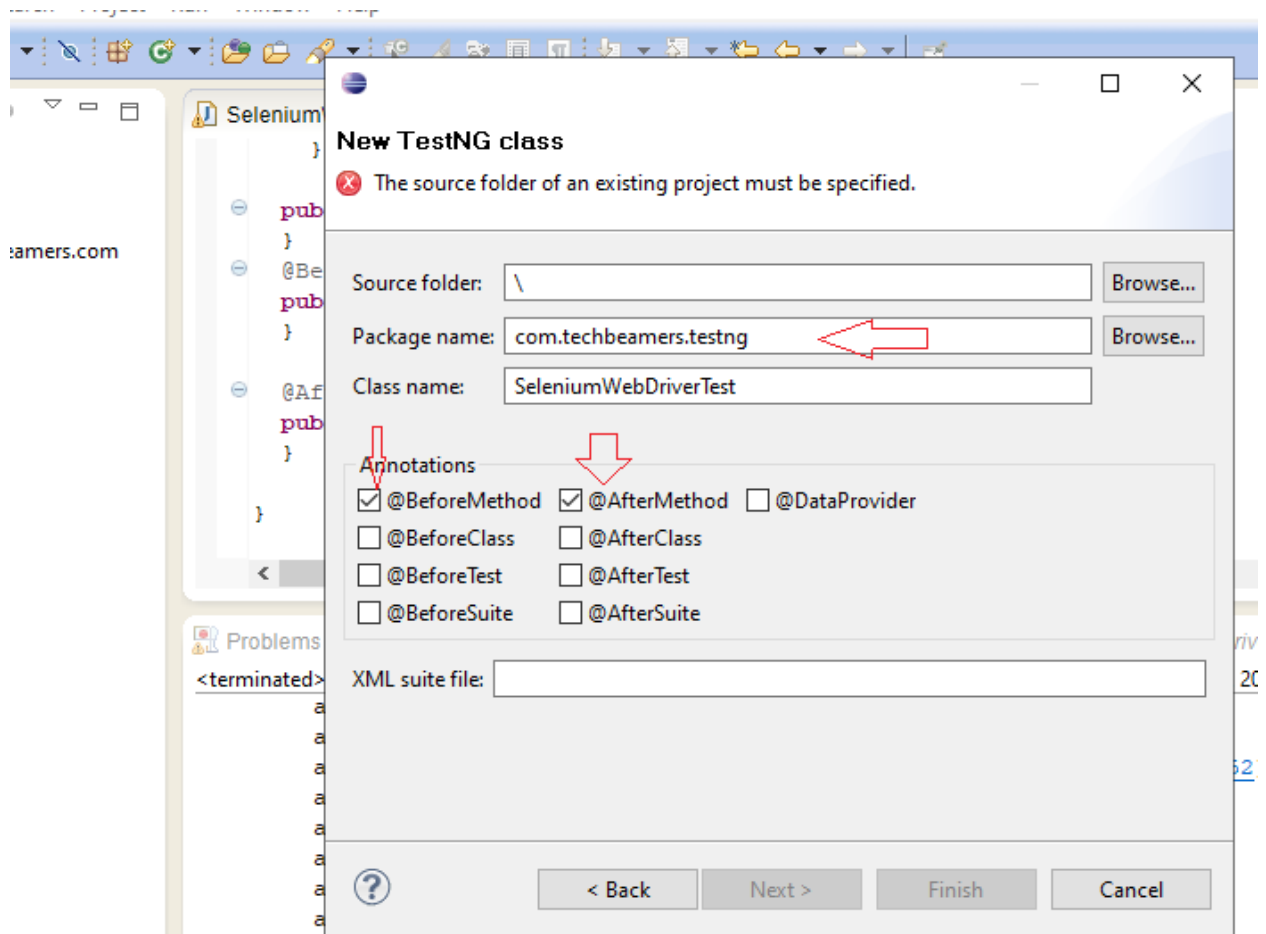



- Create a new java project in eclipse("TestNG_Demo")
- Add the TestNG library to your java project(right click on project "TestNG_Demo"->Build Path->Add Libraries)



- Choose TestNG and Finish.
- Right Click on the Package "TestNG_Demo" ->New-> others.





- Add the selenium jar file in your package->right click on the project->build path->configuration->libraries ->add external jar.
- Keep the chromedriver.exe in the same folder "TestNG_Demo" and provide the path.
- Provide the xml file to the project->right click on the project "testNG_Demo"->import->xml->xml catalog->import the xml file where you have stored (TestNG_Demo folder)
- Code

```

package com.techbeamers.com;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

public class SeleniumWebDriverTest {

    WebDriver driver = new ChromeDriver();

    @Test
    public void MyFirstTestNGTestCase() throws InterruptedException {
        String title = driver.getTitle();
        System.out.print("Current page title is : " + title);
    }
}

```

```

WebElement user = driver.findElement(By.name("userName"));
user.sendKeys("test");
WebElement pwd = driver.findElement(By.name("password"));
pwd.sendKeys("test");
WebElement signin = driver.findElement(By.name("login"));
signin.click();

Thread.sleep(1000);

        System.out.print("\n'SUCCESSFUL EXECUTION!!!");
    }
    public void invokeBrowser() {

        System.setProperty("webdriver.chrome.driver", "C:\\\\Users\\De
11\\Desktop\\JAVACODE\\chromedriver.exe");
        driver.manage().window().maximize();
        driver.get("http://newtours.demoaut.com/");
    }
    public void cleaupProc() {
        System.out.print("\nBrowser close");
        driver.quit();
    }

    public void f() {
    }
    @BeforeMethod
    public void beforeMethod() {
    }

    @AfterMethod
    public void afterMethod() {
    }
}

```

- Run the TestNG test case
- Right click on the project->run as->testNG test
- Find the defaulttest.html file in your TestDemo folder
C:\\Users\\Dell\\Desktop\\TestNG_Demo\\TestNG_Demo\\test-output\\Default suite
- Run on browser

Default test

Tests passed/Failed/Skipped:	0/1/0
Started on:	Tue Feb 21 13:54:28 IST 2023
Total time:	0 seconds (103 ms)
Included groups:	
Excluded groups:	

tover the method name to see the test class name)

test method	Exception
	<pre>org.openqa.selenium.NoSuchElementException: no such element: Unable to locate element: {"method":"css selector","selector":"*[name='userName']"} (Session info: chrome-110.0.5481.97) For documentation on this error, please visit: https://selenium.dev/exceptions/#no_such_element Build info: version: '4.8.0', revision: '267090adea' System info: os.name: 'Windows 10', os.arch: 'x86', os.version: '10.0', java.version: '1.8.0_151' Driver info: org.openqa.selenium.chrome.ChromeDriver Command: [89f5f9f75ce866915243f55db03e1e93, findElement {using=name, value=userName}] Capabilities {acceptInsecureCerts: false, browserName: chrome, browserVersion: 110.0.5481.97, chrome: {chromedriverVersion: 110.0.5481.77 (65e Session ID: 89f5f9f75ce866915243f55db03e1e93 at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method) at sun.reflect.NativeConstructorAccessorImpl.newInstance(Unknown Source) at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(Unknown Source) at java.lang.reflect.Constructor.newInstance(Unknown Source) at org.openqa.selenium.remote.codec.w3c.W3CHttpResponseCodec.createException(W3CHttpResponseCodec.java:200) at org.openqa.selenium.remote.codec.w3c.W3CHttpResponseCodec.decode(W3CHttpResponseCodec.java:133) at org.openqa.selenium.remote.codec.w3c.W3CHttpResponseCodec.decode(W3CHttpResponseCodec.java:53) at org.openqa.selenium.remote.HttpCommandExecutor.execute(HttpCommandExecutor.java:184) at org.openqa.selenium.remote.service.DriverCommandExecutor.invokeExecute(DriverCommandExecutor.java:167) at org.openqa.selenium.remote.service.DriverCommandExecutor.execute(DriverCommandExecutor.java:142) at org.openqa.selenium.remote.RemoteWebDriver.execute(RemoteWebDriver.java:943)</pre>

What does Selenium software do?

Below are a few of the most intriguing uses of Selenium software:

1. **Automated Testing:** Automated Testing comes in handy in larger projects where if not for Selenium, the tester would have to manually test each and every created functionality. With Selenium, all of the manual tasks are automated, thereby reducing the burden and stress on the testers.
2. **Cross Browsers Compatibility:** Selenium supports a wide range of browsers such as Chrome, Mozilla Firefox, Internet Explorer, Safari, and Opera.
3. **Increases Test Coverage:** With the automation of tests, the overall testing time gets reduced which results in freeing up time for the tester to perform more testing on different test scenarios in the same time.
4. **Reduces Test Execution Time:** Since Selenium supports parallel test execution, it greatly helps in reducing parallel test execution time.
5. **Multiple OS Support:** Selenium WebDriver provides support across multiple Operating Systems like Windows, Linux, UNIX, Mac, etc. With Selenium WebDriver you can create a test case on Windows OS and execute it on Mac OS.

What Are Browser Elements?

Elements are the different components that are present on web pages. The most common elements we notice while browsing are:

- Text boxes
- CTA Buttons
- Images
- Hyperlinks
- Radio buttons/ Checkboxes
- Text area/ Error messages
- Drop down box/ List box/ Combo box
- Web Table/ HTML Table
- Frame

Locating Browser Elements Present On The Web Page

Every element on a web page will have attributes (properties). Elements can have more than one attribute and most of these attributes will be unique for different elements. For example, consider a page having two elements: an image and a text box. Both these elements have a 'Name' attribute and an 'ID' attribute. These attribute values need to be unique for each element. In other words, two elements cannot have the same attribute value.

Since the elements are located using these attributes, we refer to them as 'Locators'. The locators are:

- **By.id**
Syntax: `driver.findElement(By.id("xyz"));`
- **By.name**
Syntax: `driver.findElement(By.name("xyz"));`
- **By.className**
Syntax: `driver.findElement(By.className("xyz"));`
- **By.tagName**
Syntax: `driver.findElement(By.tagName("xyz"));`
- **By.linkText**
Syntax: `driver.findElement(By.linkText("xyz"));`
- **By.partialLinkText**
Syntax: `driver.findElement(By.partialLinkText("xyz"));`
- **By.css**
Syntax: `driver.findElement(By.css("xyz"));`
- **By.xpath**
Syntax: `driver.findElement(By.xpath("xyz"));`

What are the main differences between Automation Testing & Manual Testing?

Manual Testing	Automation Testing
In manual testing, the accuracy and reliability of test cases are low, as manual tests are more prone to human error.	On the other hand, automated testing is more reliable as tools and scripts are used to perform tests.
The time required for manual testing is high as human resources perform all the tasks.	The time required is comparatively low as software tools execute the tests
In manual testing, the investment cost is low, but Return of Investment(ROI) is also low.	In automation testing, investment cost and Return on Investment are both high.
Manual testing is preferred when the test cases are run once or twice. Also suitable for Exploratory, Usability, and Adhoc Testing.	You can use test automation for Regression Testing, Performance Testing, Load Testing, or highly repeatable functional test cases.
Allows for human observation to find out any glitches. Therefore manual testing helps in improving the customer experience.	As no human observation is involved, there is no guarantee of a positive customer experience.

Can automation testing replace manual testing?

Automation testing isn't a replacement for manual testing. No matter how good automated tests are, you cannot automate everything. Manual tests play an important role in software development and come in handy whenever you have a case where you cannot use automation. Automated and manual testing each has its own strengths and weaknesses. Manual testing helps us understand the entire problem and more flexibly explore other angles of tests. On the other hand, automated testing helps save time in the long run by accomplishing a large number of surface-level tests in a short time.

What are the advantages of using an Automation Framework?

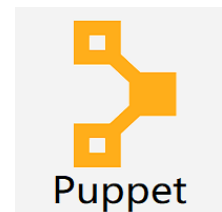
The advantages of using a test automation framework are:

- Re-usability of code
- Reliable recovery scenarios
- Maximum test coverage
- Low maintenance cost
- High Return of Investment(ROI) in the long run
- Minimal manual intervention
- Easy reporting capabilities

What are the advantages of TestNG over JUnit?

In TestNG, testing is based on JUnit, but it is designed to overcome the limitations of JUnit. Some advantages of TestNG over JUnit are:

- Annotations are easier to understand in TestNG
- In TestNG, test cases can be grouped easily
- TestNG supports parallel testing, unlike in JUnit



What is Puppet?

- Puppet is a **DevOps configuration management tool**. This is developed by Puppet Labs and is available for both open-source and enterprise versions. It is used to centralize and automate the procedure of configuration management.
- This tool is developed using Ruby DSL (domain-specific language), which allows you to change a complete infrastructure in code format and can be easily managed and configured.
- Puppet tool deploys, configures, and manages the servers. This is used particularly for the automation of hybrid infrastructure delivery and management.
- With the help of automation, Puppet enables system administrators to operate easier and faster.
- Puppet can also be used as a deployment tool as it can deploy software on the system automatically. Puppet implements infrastructure as a code, which means that you can test the environment for accurate deployment.
- Puppet supports many platforms such as Microsoft Windows, Debian/Ubuntu, Red Hat/CentOS/Fedora, MacOS X, etc.

- Puppet uses the client-server paradigm, where one system in any cluster works as the server, called the puppet master, and other works as a client on nodes called a slave.

What Is Static Code Analysis?

Static code analysis refers to the operation performed by a static analysis tool, which is the analysis of a set of code against a set (or multiple sets) of coding rules.

Static code analysis and static analysis are often used interchangeably, along with source code analysis.

Static code analysis addresses weaknesses in source code that might lead to vulnerabilities. Of course, this may also be achieved through manual source code reviews. But using automated tools is much more effective.

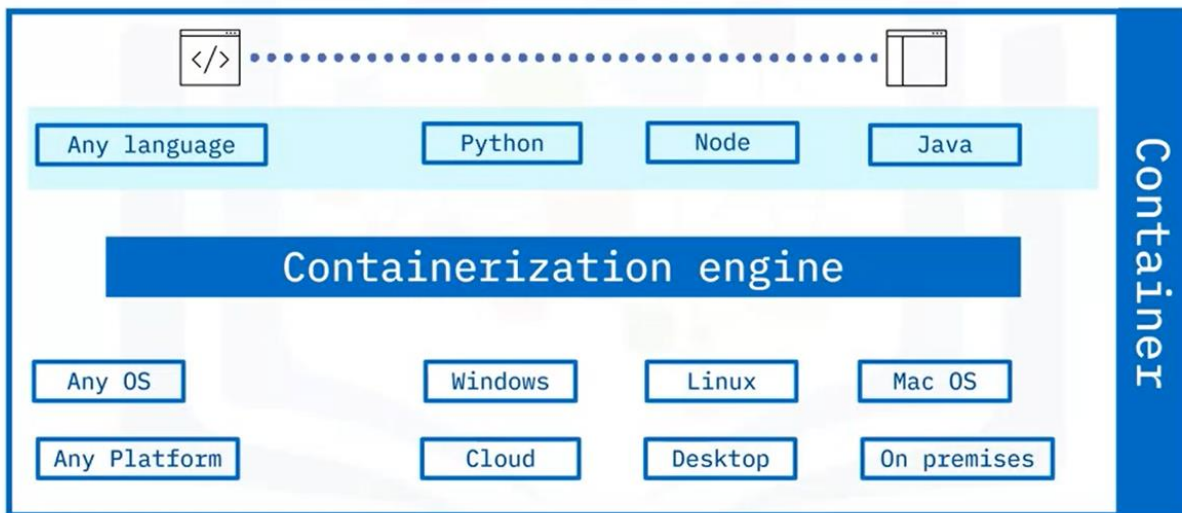
Static analysis is commonly used to comply with coding guidelines — such as MISRA. And it's often used for complying with industry standards — such as ISO 26262.

Docker

Why use containers?

Isolation and Allocation	No way to define resource boundaries for apps in a physical server
Server Utilization	Not optimal because servers tend to be either over-utilized or under-utilized
Provisioning and Costs	Requires long periods for provisioning resources and expensive maintenance costs
Performance	Constrained during peak workloads
Portability	Applications are not portable across multiple environments and operating systems
Resiliency	Complex, time-consuming and expensive
Scalability	Limited scalability and resiliency
Automation	Difficult to implement for multiple platforms

Containers offer easy portability



Container benefits

- Containers enable organizations to:
- Quickly create applications using automation
 - Lower deployment time and costs
 - Improve resource utilization (CPU, memory)
 - Port across different environments
 - Support next-gen applications (microservices)

Container challenges

- Security impacted if operating system affected
- Difficult to manage thousands of containers
- Complex to migrate legacy projects to container technology
- Difficult to right-size containers for specific scenarios

Container vendors

Docker

- Robust and most popular container platform today

Podman

- Daemon-less architecture providing more security than Docker containers

LXC

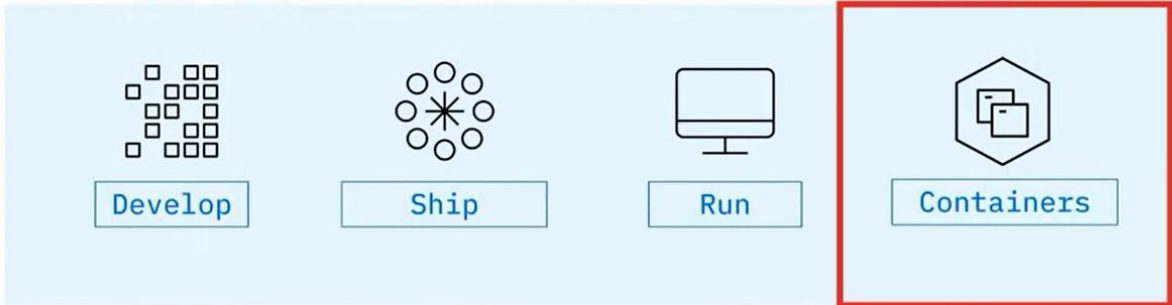
- Preferred for data-intensive apps and ops

Vagrant

- Offers highest levels of isolation on the running physical machine

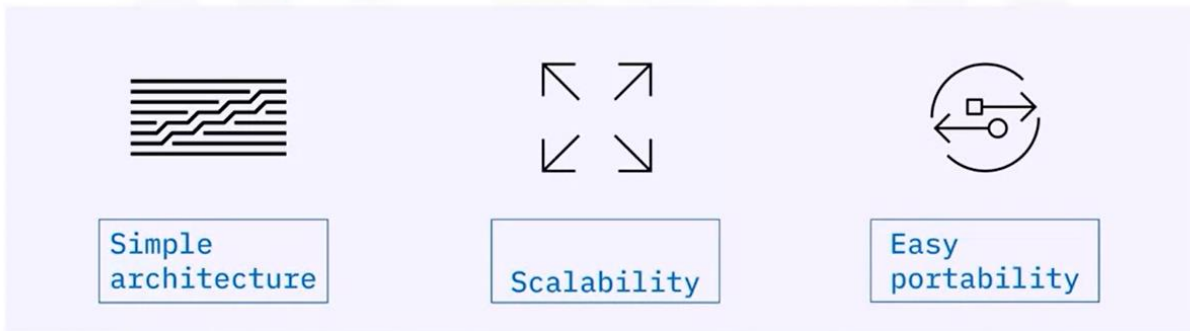
Docker defined

Available since 2013, Docker is an open platform, or engine, where programmers can:

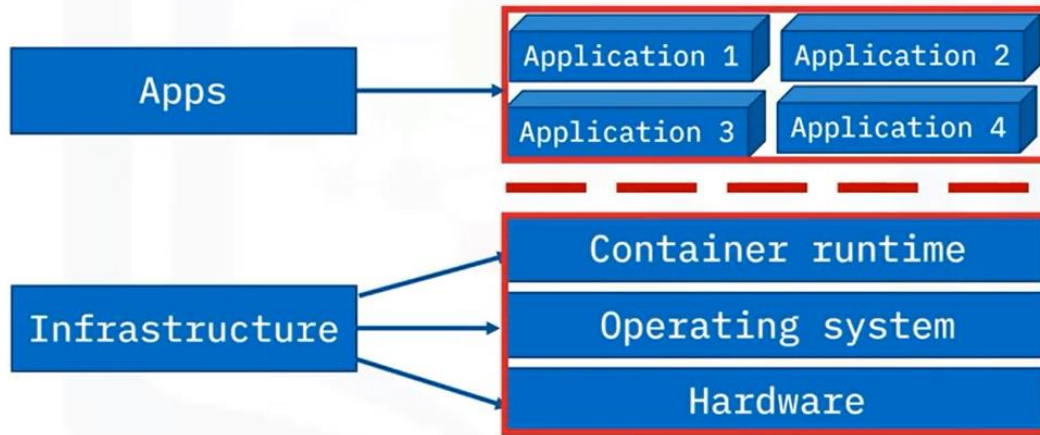


Docker becomes popular

Docker became popular due to:



The Docker process



Docker's underlying technology

- Written in Go programming language
- Uses Linux kernel's features to deliver functionality
- Uses the namespaces technology to provide an isolated workspace called "container"
- Creates a set of namespaces for every container and each aspect runs in a separate namespace with access limited to that namespace



Docker benefits

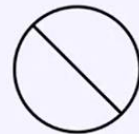
- ✓Consistent and isolated environments
- ✓Fast deployment
- ✓Repeatability and automation
- ✓Supports Agile and CI/CD DevOps practices
- ✓Versioning for easy testing, rollbacks, and redeployments
- ✓Collaboration, modularity, and scaling
- ✓Easy portability and flexibility



Challenging use cases

Docker is not a good fit for applications:

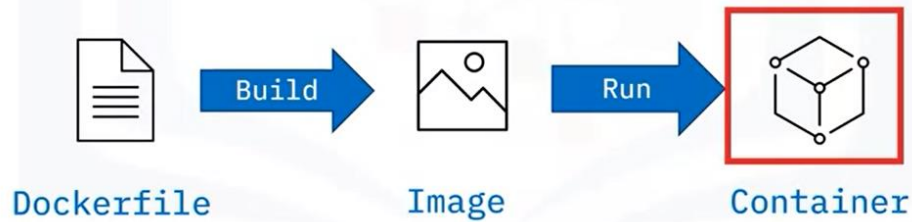
- Requiring high performance or security
- Based on Monolith architecture
- Using rich GUI features
- Performing standard desktop or limited functions



Docker container creation process

Steps to create and run containers:

1. Create a Dockerfile
2. Use the Dockerfile to create a container image
3. Use the container image to create a running container



Dockerfile example

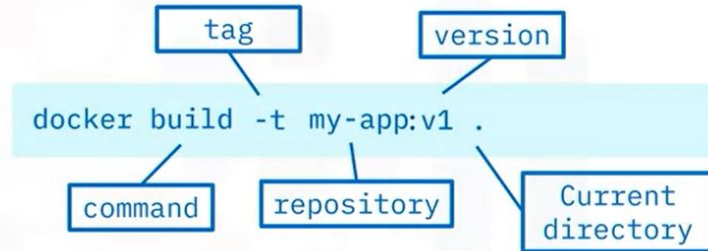
Use a Dockerfile to create a running container:

- This sample Dockerfile has the commands FROM and CMD
- FROM: Defines the base image
- CMD: Prints the words "Hello World!" on the terminal

```
FROM alpine
CMD ["echo",
    "Hello World!"]
```

Docker build command

Create a container image using the build command:



Output:

```
Sending build context to Docker daemon
.
.
Successfully built <image id>
Successfully tagged my-app:v1
```

Docker image verification



To verify the creation of the image, run the docker images command:

```
$ docker images
```

Output:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
my-app	v1	b8b15c59b352	2 minutes ago	78.2MB

Docker run command



Create the container using the run command:

```
$ docker run my-app:v1
```

```
Hello, world!
```

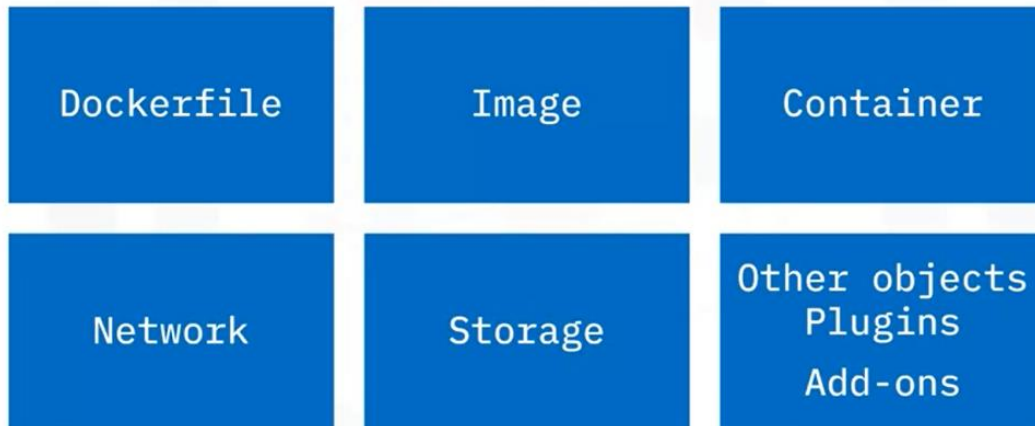
To verify the details of the container created, run the docker ps -a command:

```
$ docker ps -a
```

Docker commands

Docker command	Purpose	Example
build	Creates container images from a Dockerfile	docker build -t my-app:v1
images	Lists all images, repositories, tags, and sizes	docker images
run	Creates a container from an image	docker run -p 8080:80 nginx
push	Stores images in a configured registry	docker push my-app:v1
pull	Retrieves images from a configured registry	docker pull nginx

Docker objects

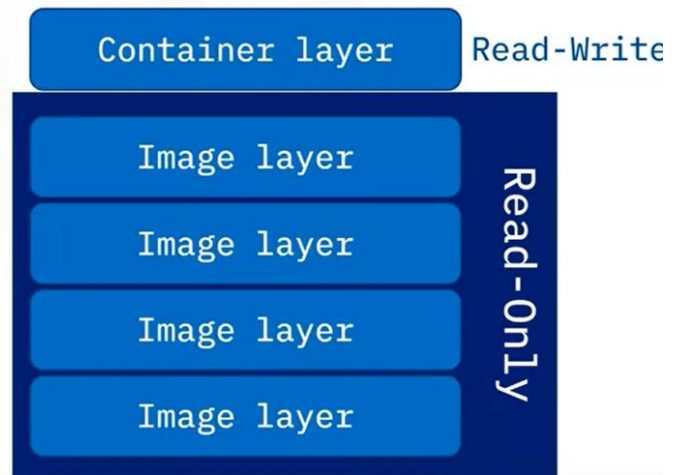


Dockerfile

- A Dockerfile is a text file that contains instructions needed to create an image
- The Dockerfile is created using any editor from the console or terminal
- Let's go over a few instructions that Docker provides
 - FROM
 - Defines base image
 - RUN
 - Executes arbitrary commands
 - CMD
 - Defines default command for container execution

Docker image

- Read-only template with instructions for creating Docker container
- Built using instructions in a Dockerfile; a new read-only image layer is created for each instruction
- A writeable layer is added when an image is run as a container
- Layers can be shared between images, which saves disk space and network bandwidth



Docker containers

A Docker container

- Is a runnable instance of an image
- Can be created, stopped, started or deleted using the Docker API or CLI
- Can connect to multiple networks, attach storage, or create a new image based on its current state
- Is well isolated from other containers and its host machine



Docker networks, storage & plugins

Networks Networks are used for the isolated container communication

Storage Docker uses volumes and bind mounts to persist data even after a container stops

Plugins Storage plugins provide the ability to connect to external storage platforms

Docker architecture



- Based on client-server architecture
- Provides a complete application environment
- Includes the client, the host, and the registry components

Docker process overview

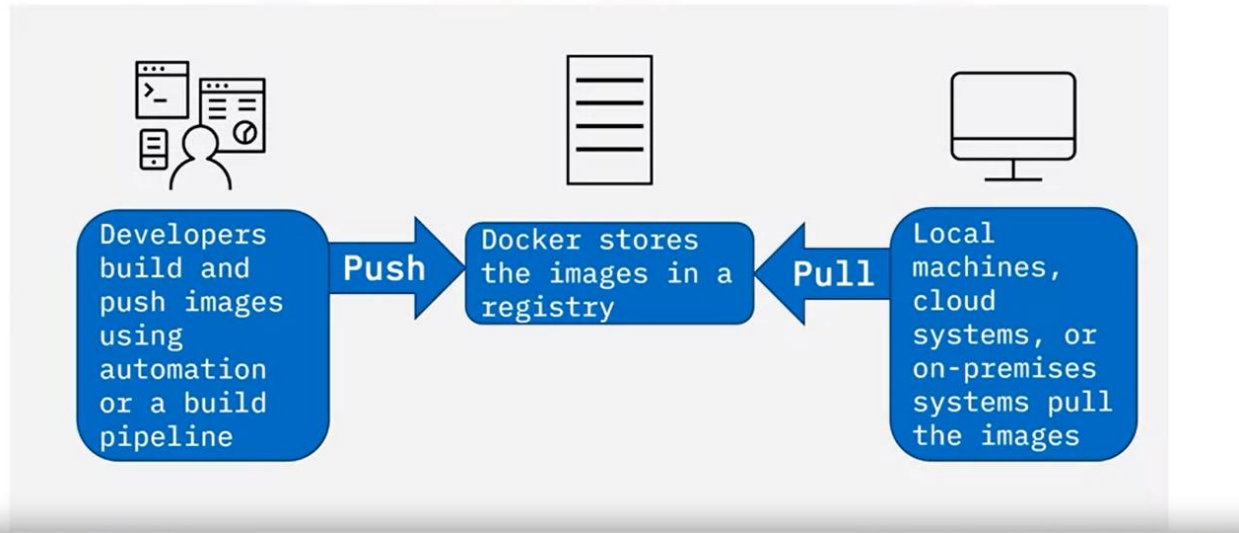
1. Using either the Docker command line interface (CLI) or REST APIs, the Docker client sends instructions or commands to the Docker host server
2. The Docker host server, referred to as the host, includes the Docker daemon known as dockerd
3. The daemon listens for Docker API requests or commands such as "docker run" and processes those commands
4. The daemon builds, runs, and distributes containers to the registry
5. The registry stores the images

Registry

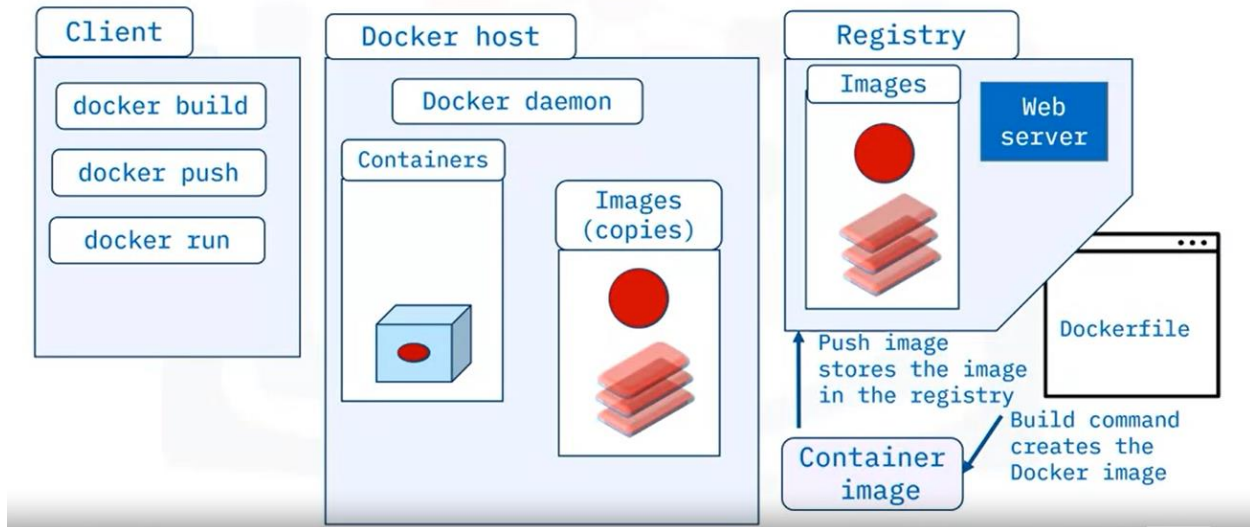
- Stores and distributes images
- Access is public or private
 - Public such as Docker Hub – Everyone can access
 - Private – Implemented for security
- Registry locations are either hosted or self-hosted



Registry access



Docker architecture



Command	Description
<code>curl localhost</code>	Pings the application.
<code>docker build</code>	Builds an image from a Dockerfile.
<code>docker build . -t</code>	Builds the image and tags the image id.
<code>docker CLI</code>	Start the Docker command line interface.
<code>docker container rm</code>	Removes a container.
<code>docker images</code>	Lists the images.

docker ps	Lists the containers.
docker ps -a	Lists the containers that ran and exited successfully.
docker pull	Pulls the latest image or repository from a registry.
docker push	Pushes an image or a repository to a registry.
docker run	Runs a command in a new container.
docker run -p	Runs the container by publishing the ports.
docker stop	Stops one or more running containers.
docker stop \$(docker ps -q)	Stops all running containers.
docker tag	Creates a tag for a target image that refers to a source image.
docker --version	Displays the version of the Docker CLI.
exit	Closes the terminal session.
export MY_NAMESPACE	Exports a namespace as an environment variable.
git clone	Clones the git repository that contains the artifacts needed.
ibmcloud cr images	Lists images in the IBM Cloud Container Registry.
ibmcloud cr login	Logs your local Docker daemon into IBM Cloud Container Registry.
ibmcloud cr namespaces	Views the namespaces you have access to.
ibmcloud cr region-set	Ensures that you are targeting the region appropriate to your cloud account.
ibmcloud target	Provides information about the account you're targeting.
ibmcloud version	Displays the version of the IBM Cloud CLI.
ls	Lists the contents of this directory to see the artifacts.

Term	Definition
Agile	is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer issues.
Client-server architecture	is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.

A container	powered by the containerization engine, is a standard unit of software that encapsulates the application code, runtime, system tools, system libraries, and settings necessary for programmers to efficiently build, ship and run applications.
Container Registry	Used for the storage and distribution of named container images. While many features can be built on top of a registry, its most basic functions are to store images and retrieve them.
CI/CD pipelines	A continuous integration and continuous deployment (CI/CD) pipeline is a series of steps that must be performed in order to deliver a new version of software. CI/CD pipelines are a practice focused on improving software delivery throughout the software development life cycle via automation.
Cloud native	A cloud-native application is a program that is designed for a cloud computing architecture. These applications are run and hosted in the cloud and are designed to capitalize on the inherent characteristics of a cloud computing software delivery model.
Daemon-less	A container runtime that does not run any specific program (daemon) to create objects, such as images, containers, networks, and volumes.
DevOps	is a set of practices, tools, and a cultural philosophy that automate and integrate the processes between software development and IT teams.
Docker	An open container platform for developing, shipping and running applications in containers.
A Dockerfile	is a text document that contains all the commands you would normally execute manually in order to build a Docker image. Docker can build images automatically by reading the instructions from a Dockerfile.
Docker client	is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.
Docker Command Line Interface (CLI)	The Docker client provides a command line interface (CLI) that allows you to issue build, run, and stop application commands to a Docker daemon.
Docker daemon (dockerd)	creates and manages Docker objects, such as images, containers, networks, and volumes.
Docker Hub	is the world's easiest way to create, manage, and deliver your team's container applications.
Docker localhost	Docker provides a host network which lets containers share your host's networking stack. This approach means that a localhost in a container resolves to the physical host, instead of the container itself.

Docker remote host	A remote Docker host is a machine, inside or outside our local network which is running a Docker Engine and has ports exposed for querying the Engine API.
Docker networks	help isolate container communications.
Docker plugins	such as a storage plugin, provides the ability to connect external storage platforms.
Docker storage	uses volumes and bind mounts to persist data even after a running container is stopped.
LXC	Linux Containers is a OS-level virtualization technology that allows creation and running of multiple isolated Linux virtual environments (VE) on a single control host.
IBM Cloud Container Registry	stores and distributes container images in a fully managed private registry.
Image	An immutable file that contains the source code, libraries, and dependencies that are necessary for an application to run. Images are templates or blueprints for a container.
Immutability	Images are read-only; if you change an image, you create a new image.
Microservices	are a cloud-native architectural approach in which a single application contains many loosely coupled and independently deployable smaller components or services.
Namespace	A Linux namespace is a Linux kernel feature that isolates and virtualizes system resources. Processes which are restricted to a namespace can only interact with resources or processes that are part of the same namespace. Namespaces are an important part of Docker's isolation model. Namespaces exist for each type of resource, including networking, storage, processes, hostname control and others.
Operating System Virtualization	OS-level virtualization is an operating system paradigm in which the kernel allows the existence of multiple isolated user space instances, called containers, zones, virtual private servers, partitions, virtual environments, virtual kernels, or jails.
Private Registry	Restricts access to images so that only authorized users can view and use them.
REST API	A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services.
Registry	is a hosted service containing repositories of images which responds to the Registry API.
Repository	is a set of Docker images. A repository can be shared by pushing it to a registry server. The different images in the repository can be labelled using tags.

Server Virtualization

Server virtualization is the process of dividing a physical server into multiple unique and isolated virtual servers by means of a software application. Each virtual server can run its own operating systems independently.

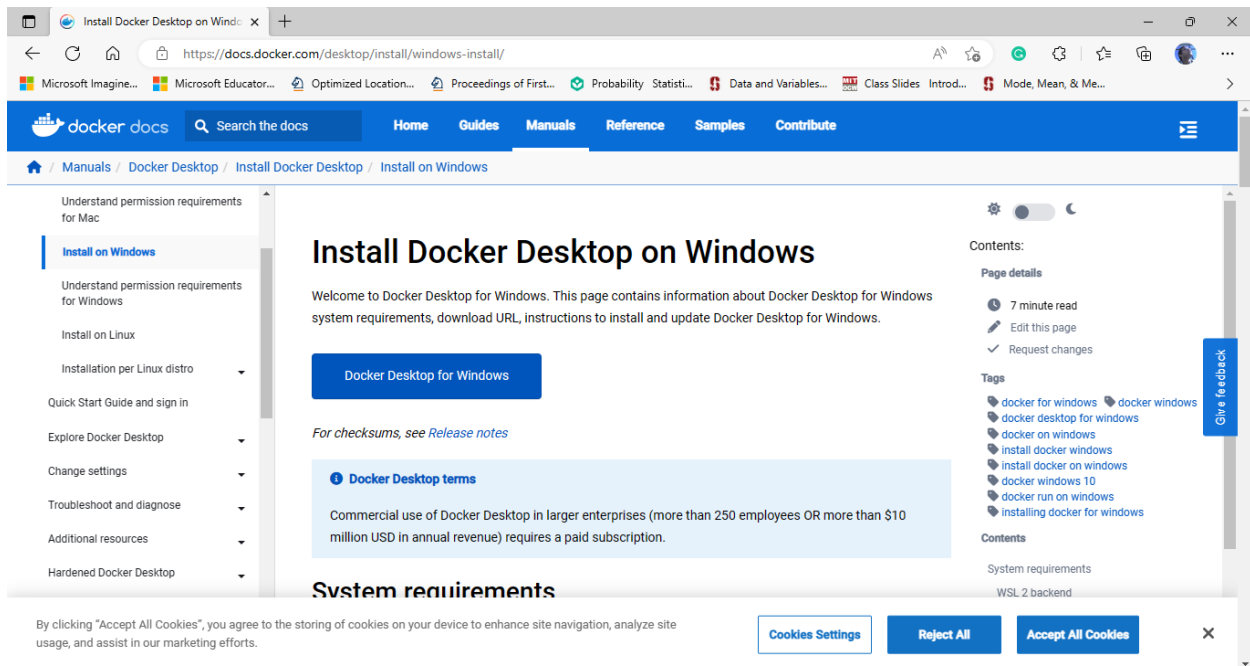
Serverless

is a cloud-native development model that allows developers to build and run applications without having to manage servers.

Tag

A tag is a label applied to a Docker image in a repository. Tags are how various images in a repository are distinguished from each other.

- Download and install Docker on your system.



The screenshot shows a web browser window displaying the Docker Docs page for installing Docker Desktop on Windows. The page title is "Install Docker Desktop on Windows". The main content area includes a welcome message, a "Docker Desktop for Windows" button, and a section for "System requirements". A sidebar on the left lists navigation options like "Understand permission requirements for Mac", "Install on Linux", and "Explore Docker Desktop". The right sidebar shows "Page details" (7 minute read), "Tags" (docker for windows, docker desktop for windows, etc.), and "Contents" (System requirements, WSL 2 backend). A cookie consent banner is visible at the bottom of the page.

Create an Login in your DockerHub account

Open PWD platform on your browser (link : <https://labs.play-with-docker.com/>)

Click on Add New Instance on the left side of the screen to bring up Alpine OS instance on the right side

Task #1

Creating Docker File

Command: vim Dockerfile

Write the following content into the docker file

```
FROM alpine:3.5
```

```
RUN apk update
```

```
RUN apk add git
```

then Esc : wq

Build Docker Image

```
docker build https://github.com/alpine-docker/git
```

Tagging image as myimage_1

```
docker tag https://github.com/alpine-docker/git myimage_1
```

Verify the Images

```
docker images
```

Create a container

```
docker run -itd alpine
```

```
docker ps
```

Pushing it to DockerHub

```
docker login
# enter your username and password
docker push <image name>
```

Create Ubuntu Container

```
docker pull ubuntu
docker run -dit ubuntu
```

Accessing the container shell

```
docker exec -ti <container-id> bash
####for container id use docker ps command
```

Display the docker host information

```
docker info
```

Stop the container

```
docker stop <container name>
```

Task #2

Running Single Node WordPress Example

```
git clone https://github.com/collabnix/dockerlabs
cd dockerlabs/intermediate/workshop/DockerCompose/
cd wordpress
```

Bringup the containers

```
docker-compose up -d
```

Checking container status

```
docker-compose ps
```

Listout the services

```
docker-compose ps --services
```

Stop the container of a single service

```
docker-compose stop webservice
```

Checking container status

```
docker-compose ps
```

```
docker-compose start webservice
```

```
docker-compose start webservice
```

```
#####
```

What is the difference between Docker and Docker Compose?

The key difference between docker run versus docker-compose is that docker run is entirely command line based, while docker-compose reads configuration data from a YAML file.

#####

Create an Login in your DockerHub account

Open PWD platform on your browser (link : <https://labs.play-with-docker.com/>)

Click on Add New Instance on the left side of the screen to bring up Alpine OS instance on the right side

Task #1

Creating Docker File

Command: vim Dockerfile

Write the following content into the docker file

FROM alpine:3.5

RUN apk update

RUN apk add git

then Esc : wq

Build Docker Image

docker build <https://github.com/alpine-docker/git>

Tagging image as myimage_1

docker tag <https://github.com/alpine-docker/git> myimage_1

Verify the Images

```
docker images
```

Create a container

```
docker run -itd alpine
```

```
docker ps
```

Pushing it to DockerHub

```
docker login
```

```
# enter your username and password
```

```
# tag your image with your id as
```

```
docker tag alpine <docker id>/myimage_1
```

```
docker push <image name>
```

Create Ubuntu Container

```
docker pull ubuntu
```

```
docker run -dit ubuntu
```

Accessing the container shell

```
docker exec -ti <container-id> bash
```

```
###for container id use docker ps command
```

Display the docker host information

```
docker info
```

Stop the container

```
docker stop <container name>
```

Task #2

Running Single Node WordPress Example

```
git clone https://github.com/collabnix/dockerlabs  
cd dockerlabs/intermediate/workshop/DockerCompose/  
cd wordpress
```

Bringup the containers

```
docker-compose up -d
```

Checking container status

```
docker-compose ps
```

Listout the services

```
docker-compose ps --services
```

Stop the container of a single service

```
docker-compose stop webservice
```

Checking container status

docker-compose ps

docker-compose start webserver

docker-compose start webserver

#####

What is the difference between Docker and Docker Compose?

The key difference between docker run versus docker-compose is that docker run is entirely command line based, while docker-compose reads configuration data from a YAML file.

#####

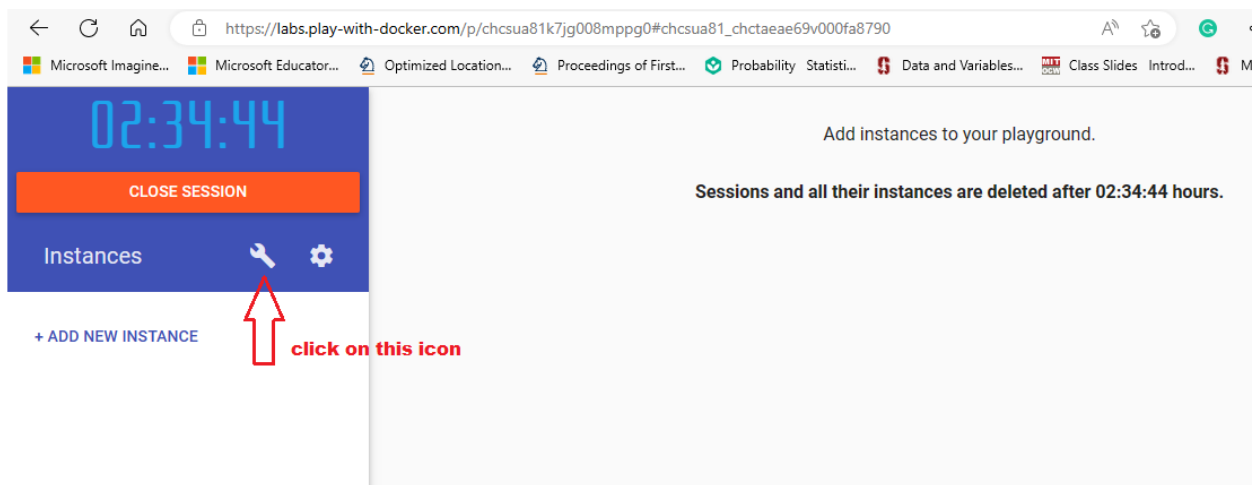
Docker Swarm

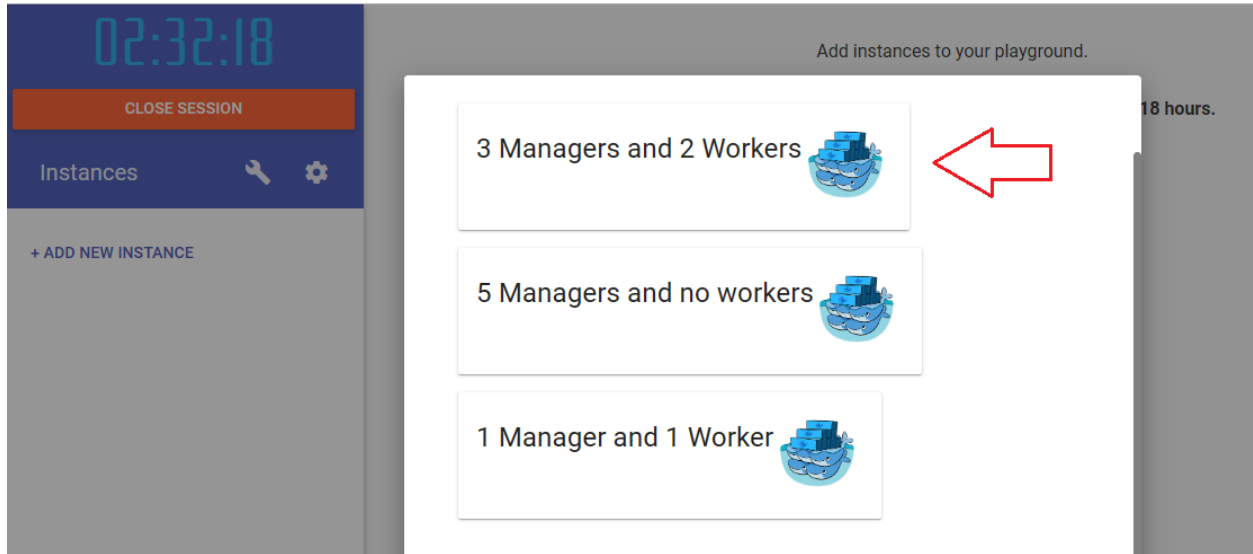
Swarm is the platform for managing "Dockerized" containers and is a native mode of Docker. A Swarm cluster (set of computers that operate as a single system) consists of: Docker Engine-deployed Swarm manager nodes that manage the cluster.

#####

Task # 03

Open PWD platform in your browser





Clone the Repository

```
git clone https://github.com/dockersamples/docker-swarm-visualizer
cd docker-swarm-visualizer
docker-compose up -d
```

To run in a docker swarm:

```
docker service create \
  --name=viz \
  --publish=8080:8080/tcp \
  --constraint=node.role==manager \
  --mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
  dockersamples/visualizer
```

```
docker run -it -d -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock dockersamples/visualizer
```

If port 8080 is already in use on your host, you can specify e.g. -p [YOURPORT]:8080

Task # 4

Docker Network

Display all the existent networks in the host:

```
docker network ls
```

##The docker network command is the main command for configuring and managing container networks.

##docker network inspect to view configuration details of the container networks on your Docker host.

```
docker network inspect bridge
```

##Run a docker info command on any of your Docker hosts and locate the list of network plugins.

```
docker info
```

List the bridges on your Docker host

```
brctl show
```

##The output above shows a single Linux bridge called docker0. This is the bridge that was automatically created for the bridge network.

##use the ip command to view details of the docker0 bridge.

Connect a container

##The bridge network is the default network for new containers. This means that unless you specify a different network, all new containers will be connected to the bridge network.

Create a new container.

```
docker run -dt ubuntu sleep infinity
```

##This command will create a new container based on the ubuntu:latest image and will run the sleep command to keep the container running in the background. As no network was specified on the docker run command, the container will be added to the bridge network.

```
brctl show
```

##docker0 bridge now has an interface connected. This interface connects the docker0 bridge to the new container just created.

##Inspect the bridge network

```
docker network inspect bridge
```

Test network connectivity

The output to the previous docker network inspect command shows the IP address of the new container. In the previous example it is "172.17.0.2" but yours might be different.

##Ping the IP address of the container from the shell prompt of your Docker host.

```
ping 172.17.0.2
```

##Press Ctrl-C to stop the ping.

```
docker ps
```

Exec into the container

```
docker exec -it <container id> /bin/bash
```

Update APT package lists and install the iputils-ping package

```
apt-get update
```

Start a new container based off the official NGINX image.

NGINX is open source software for web serving, reverse proxying, caching, load balancing, media streaming, and more. It started out as a web server designed for maximum performance and stability. In addition to its HTTP server capabilities, NGINX can also function as a proxy server for email (IMAP, POP3, and SMTP) and a reverse proxy and load balancer for HTTP, TCP, and UDP servers.

```
docker run --name web1 -d -p 8080:80 nginx
```

##Check that the container is running and view the port mapping.

```
curl <ip>:8080
```



Kubernetes

Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.



Differences Between Docker Swarm and Kubernetes

Kubernetes and Docker Swarm are both effective solutions for:

- Massive scale application deployment
- Implementation
- Management

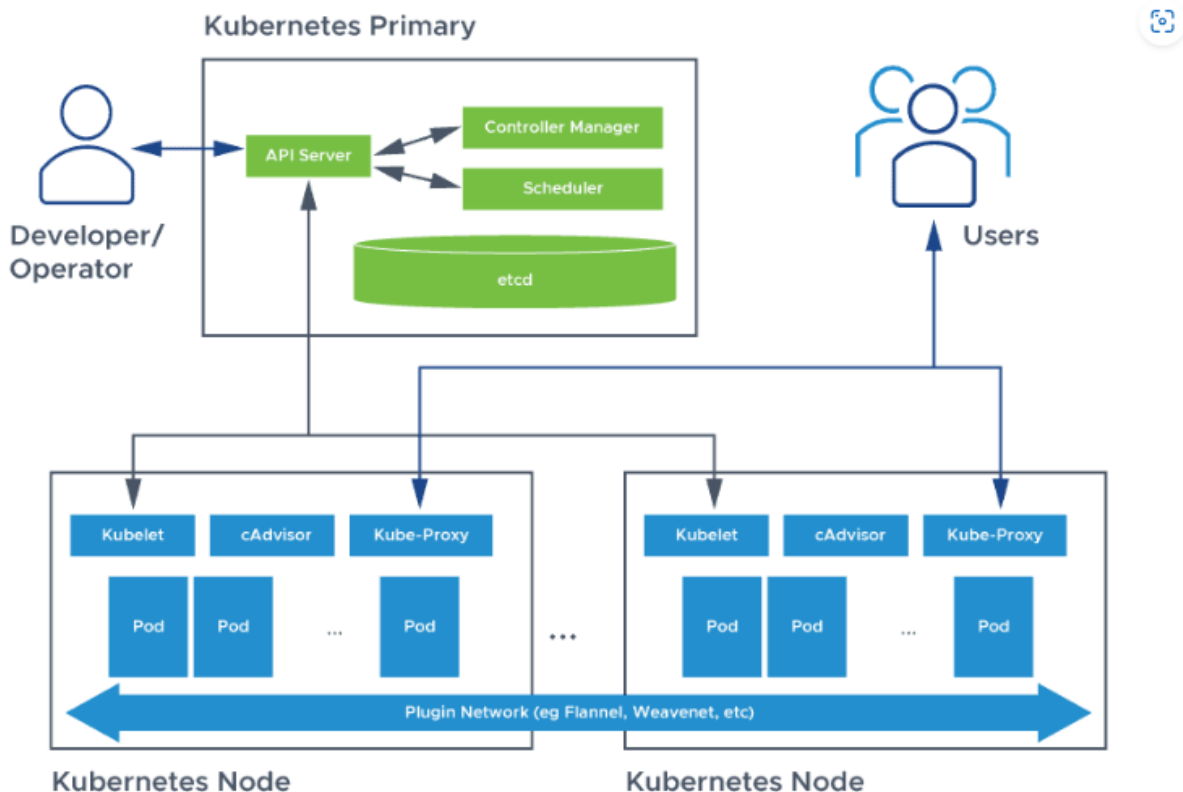
The Core Differences

The main difference is that Kubernetes is a container orchestration system that manages multiple containers. Docker Swarm does not manage any containers but instead is a cluster manager for Docker containers. Kubernetes also has built-in support for stateful applications, whereas Docker Swarm does not.

- Kubernetes is designed to work with any programming language and framework, while Docker Swarm only works with the Docker Engine API.

- Kubernetes runs on top of Linux Containers while Swarm runs inside Docker Containers.
- Kubernetes uses a master-slave architecture where one master instance controls multiple worker instances; Kubernetes uses an active/passive model where each worker instance is controlled by its own master instance.

KubernetesArchitecture



A Kubernetes cluster is a form of Kubernetes deployment architecture. Basic Kubernetes architecture exists in two parts: the control plane and the nodes or compute machines. Each node could be either a physical or virtual machine and is its own Linux environment. Every node also runs pods, which are composed of containers.

Kubernetes architecture components or K8s components include the Kubernetes control plane and the nodes in the cluster. The control plane machine components include the Kubernetes API server, Kubernetes scheduler, Kubernetes controller manager, and etcd.

Kubernetes node components include a container runtime engine or docker, a Kubelet service, and a Kubernetes proxy service.

minikube start

minikube is local Kubernetes, focusing on making it easy to learn and develop for Kubernetes.

Installation

To install the latest minikube stable release on x86-64 Windows using Windows Package Manager:

If the Windows Package Manager is installed, use the following command to install minikube:

```
winget install minikube
```

OR

If the Chocolatey Package Manager is installed, use the following command:

```
choco install minikube
```

```
Command Prompt
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell>winget install minikube
The 'msstore' source requires that you view the following agreements before using.
Terms of Transaction: https://aka.ms/microsoft-store-terms-of-transaction
The source requires the current machine's 2-letter geographic region to be sent to the backend service to function properly (ex. "US").

Do you agree to all the source agreements terms?
[Y] Yes [N] No: y
Found Kubernetes - Minikube - A Local Kubernetes Development Environment [Kubernetes.minikube] Version 1.30.1
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.
Downloading https://github.com/kubernetes/minikube/releases/download/v1.30.1/minikube-installer.exe
34.2 MB / 34.2 MB
Successfully verified installer hash
Starting package install...
Successfully installed

C:\Users\Dell>
```

Start your cluster

From a command prompt with administrator access; run

```
minikube start
```

```

Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>minikube start
* minikube v1.30.1 on Microsoft Windows 10 Pro 10.0.19045.2965 Build 19045.2965
* Automatically selected the hyperv driver
* Downloading VM boot image ...
  > minikube-v1.30.1-amd64.iso....: 65 B / 65 B [-----] 100.00% ? p/s 0s
  > minikube-v1.30.1-amd64.iso: 282.84 MiB / 282.84 MiB 100.00% 52.65 MiB p
* Starting control plane node minikube in cluster minikube
* Downloading Kubernetes v1.26.3 preload ...
  > preloaded-images-k8s-v18-v1...: 397.02 MiB / 397.02 MiB 100.00% 48.92 M
* Creating hyperv VM (CPUs=2, Memory=2200MB, Disk=20000MB) ...
* Preparing Kubernetes v1.26.3 on Docker 20.10.23 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

C:\WINDOWS\system32>

```

Interact with your cluster

If you already have kubectl installed, you can now use it to access your shiny new cluster:

```
kubectl get po -A
```

OR

```
minikube kubectl -- get po -A
```

```

C:\WINDOWS\system32>kubectl get po -A
'kubectl' is not recognized as an internal or external command,
operable program or batch file.

C:\WINDOWS\system32>minikube kubectl -- get po -A
  > kubectl.exe.sha256: 64 B / 64 B [-----] 100.00% ? p/s 0s
  > kubectl.exe: 46.49 MiB / 46.49 MiB [-----] 100.00% 69.80 MiB p/s 900ms

```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-787d4945fb-snnkg	1/1	Running	0	2m36s
kube-system	etcd-minikube	1/1	Running	0	2m49s
kube-system	kube-apiserver-minikube	1/1	Running	0	2m49s
kube-system	kube-controller-manager-minikube	1/1	Running	0	2m49s
kube-system	kube-proxy-f6zmq	1/1	Running	0	2m36s
kube-system	kube-scheduler-minikube	1/1	Running	0	2m51s
kube-system	storage-provisioner	1/1	Running	1 (2m4s ago)	2m43s

```

C:\WINDOWS\system32>

```

minikube bundles the Kubernetes Dashboard, allowing you to get easily acclimated to your new environment:

```
minikube dashboard
```



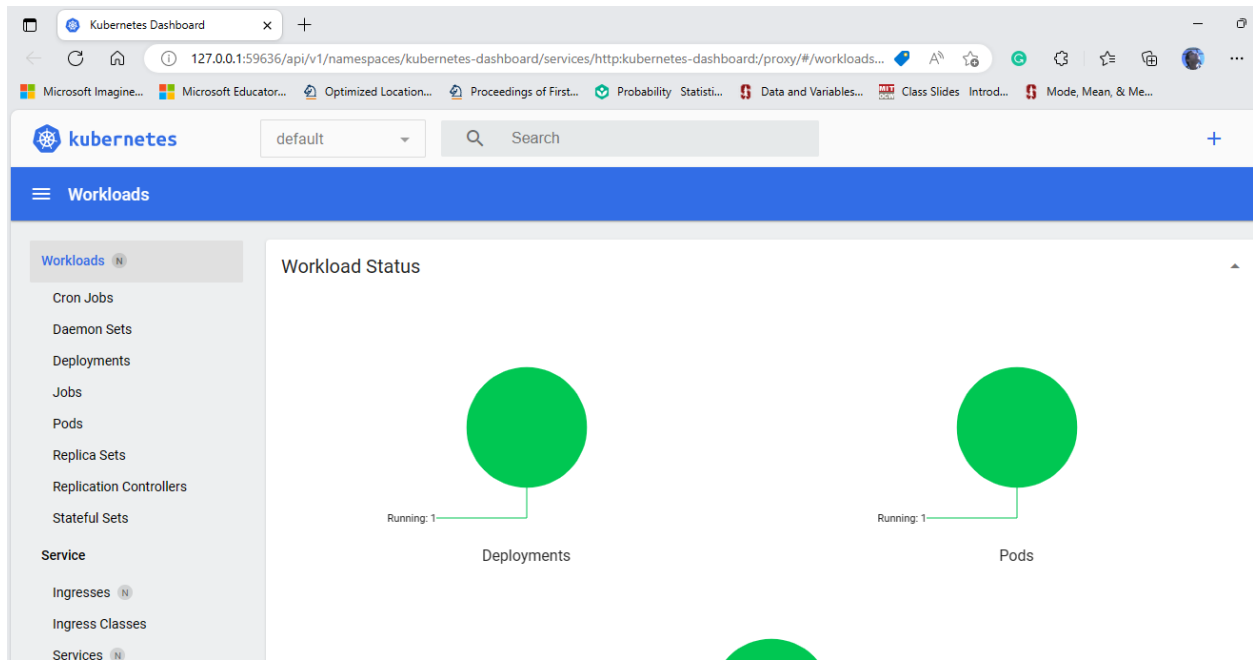
```

C:\WINDOWS\system32>minikube dashboard
* Enabling dashboard ...
  - Using image docker.io/kubernetes/dashboard:v2.7.0
  - Using image docker.io/kubernetes/metrics-scraper:v1.0.8
* Some dashboard features require the metrics-server addon. To enable all features please run:

    minikube addons enable metrics-server

* Verifying dashboard health ...
* Launching proxy ...
* Verifying proxy health ...
* Opening http://127.0.0.1:59363/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...

```



Ctrl+c for exit

Deploy applications

```
minikube kubectl -- create deployment hello-minikube --image=kicbase/echo-server:1.0
```

```
minikube kubectl -- expose deployment hello-minikube --type=NodePort --port=8080
```

```

C:\WINDOWS\system32>minikube kubectl -- create deployment hello-minikube --image=kicbase/echo-server:1.0
deployment.apps/hello-minikube created

C:\WINDOWS\system32>minikube kubectl -- expose deployment hello-minikube --type=NodePort --port=8080
service/hello-minikube exposed

```

```
minikube kubectl -- get services hello-minikube
```

```
C:\WINDOWS\system32>minikube kubectl -- get services hello-minikube
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
hello-minikube     NodePort    10.99.18.149  <none>         8080:32460/TCP  86s
```

minikube service hello-minikube

```
C:\WINDOWS\system32>minikube service hello-minikube
-----
| NAMESPACE | NAME           | TARGET PORT | URL                               |
|-----|-----|-----|-----|
| default   | hello-minikube | 8080        | http://172.31.27.249:32460      |
|-----|-----|-----|-----|
* Opening service default/hello-minikube in default browser...
```



minikube kubectl -- port-forward service/hello-minikube 7080:8080

```
C:\WINDOWS\system32>minikube kubectl -- port-forward service/hello-minikube 7080:8080
Forwarding from 127.0.0.1:7080 -> 8080
Forwarding from [::1]:7080 -> 8080
Handling connection for 7080
Handling connection for 7080
Handling connection for 7080
Handling connection for 7080
Handling connection for 7080
Handling connection for 7080
```

Your application is now available at <http://localhost:7080/>

Ctrl+c for exit

Manage your cluster

Pause Kubernetes without impacting deployed applications:

```
minikube pause
```

Unpause a paused instance:

```
minikube unpause
```

Halt the cluster:

```
minikube stop
```

Delete all of the minikube clusters:

```
minikube delete --all
```

What are hypervisors?

Virtualization requires the use of a hypervisor, which was originally called a virtual machine monitor or VMM. A hypervisor abstracts operating systems and applications from their underlying hardware. The physical hardware that a hypervisor runs on is typically referred to as a host machine, whereas the VMs the hypervisor creates and supports are collectively called guest machines.

Type 1 hypervisors

A Type 1 hypervisor runs directly on the host machine's physical hardware, and it's referred to as a bare-metal hypervisor. The Type 1 hypervisor doesn't have to load an underlying OS. With direct access to the underlying hardware and no other software -- such as OSes and device drivers -- to contend with for virtualization, Type 1 hypervisors are regarded as the most efficient and best-performing hypervisors available for enterprise computing.

Type 2 hypervisors

A Type 2 hypervisor is typically installed on top of an existing OS. It is sometimes called a hosted hypervisor because it relies on the host machine's preexisting OS to manage calls to CPU, memory, storage and network resources.

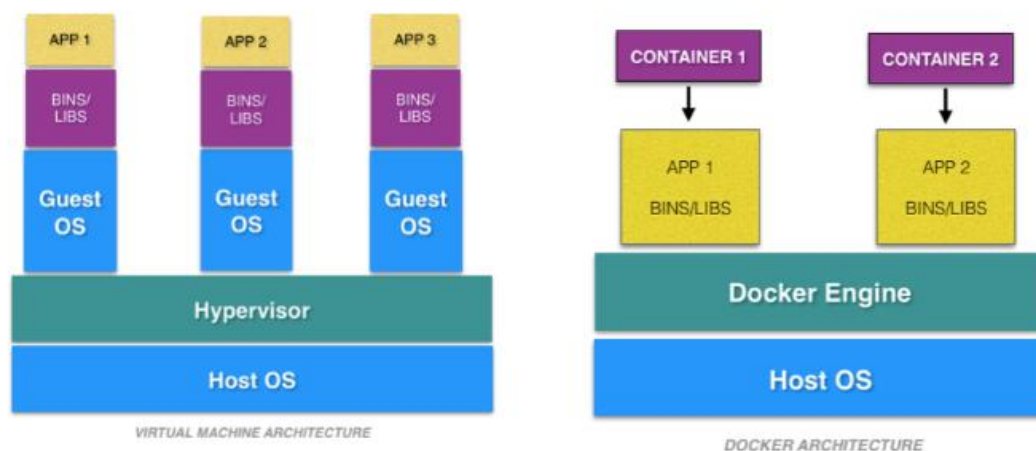
Type 2 hypervisors trace their roots back to the early days of x86 virtualization when the hypervisor was added above the existing systems' OSes. Although the purpose and goals of Type 1 and Type 2 hypervisors are identical, the presence of an underlying OS with Type 2 hypervisors introduces unavoidable latency; all of the hypervisor's activities and the work of every VM has to pass through the host OS. Also, any security flaws or vulnerabilities in the host OS could potentially compromise all of the VMs running above it.

Docker vs Virtual Machine: main differences

The following are the significant differences between Docker and virtual machines.

OS Support and Architecture

The main difference between Docker and VMs lies in their architecture, demonstrated below.



VMs have the host OS and guest OS inside each VM. A guest OS can be any OS, like Linux or Windows, irrespective of the host OS. In contrast, Docker containers host on a single physical server with a host OS, which shares among them. Sharing the host OS between containers makes them light and increases the boot time. Docker containers are considered suitable to run multiple applications over a single OS kernel; whereas, virtual machines are needed if the applications or services required to run on different OS.

Kubernetes vs. Docker Swarm – A comparison

The major difference between the platforms is based on complexity. Kubernetes is well suited for complex applications. On the other hand, Docker Swarm is designed for ease of use, making it a preferable choice for simple applications.

- Kubernetes: Depending on the operating system, manual installation can differ for each OS. If you are using services from a cloud provider, installation is not required.
- Docker Swarm: Docker instances are typically consistent across operating systems and thus fairly simple to set up.
- Kubernetes: Services are made discoverable through a single DNS name. Kubernetes accesses container applications through an IP address or HTTP route.
- Swarm: Comes with internal load balancers.
- Kubernetes: Kubernetes has built-in monitoring along with third-party monitoring tools integration support.
- Docker Swarm: In contrast, there are no in-built monitoring mechanisms in Docker Swarm. However, Docker Swarm supports monitoring through third-party applications.

Continuous Delivery vs. Deployment – What's the Difference?

Continuous delivery automates deployment of a release to an environment for staging or testing. **Continuous deployment** automatically deploys every release through your pipeline (including testing) and to production.

What Continuous Delivery Does

- Continuous delivery **automatically deploys releases to a testing or staging environment.**
- Continuous delivery **does require human intervention to deploy a release from staging to production.**
- Continuous delivery **does not automatically deploy code changes to production.**

What Continuous Deployment Does

- Continuous deployment **automatically deploys releases** from building through testing and **into production.**
- Continuous deployment **doesn't require human intervention.**
- Continuous deployment **doesn't ensure your testing culture and protocol is up to snuff.** It simply looks for a checked box and, if it finds the box is checked, deploys the release to production and beyond.

Microservices

A microservices architecture is a cloud-native approach to building software in a way that allows for each core function within an application to exist independently.

Monolithic architecture vs microservices architecture

The traditional approach to building applications has focused on the monolith. In a monolithic architecture, all the functions and services within an application are locked together, operating as a single unit. When the application is added to or improved upon in any way, the architecture grows more complex. This makes it more difficult to optimize any singular function within the application without taking the entire application apart. This also means that if one process within the application needs to be scaled, the entire application must be scaled as well.

In microservices architectures, applications are built so that each core function within the app runs independently. This allows development teams to build and update new components to meet changing business needs without disrupting the application as a whole.

Automation vs Orchestration: The Main Difference

Automation refers to automating a single process or a small number of related tasks (e.g., deploying an app). Orchestration refers to managing multiple automated tasks to create a dynamic workflow (e.g., deploying an app, connecting it to a network, and integrating it with other systems).

Whereas automation is a simple "if this, then that" process, orchestration has many moving parts and requires advanced logic that can:

Make decisions based on an output from an automated task.

Branch out into different steps and actions.

Adapt to changing circumstances and conditions.

Coordinate multiple tasks at the same time.

The line between the two concepts can be blurry. For example, you can have a process with 100 steps and automate all of them, and you would still rely on automation. You only start orchestrating when you introduce some coordination and decision-making to the process.

Technically, automation is a subset of orchestration as you cannot orchestrate manual, non-automated tasks.

Automation and orchestration do not aim to replace people fully. As a company becomes more reliant on automated tasks, human involvement becomes less frequent but more valuable. The IT staff gets to focus on problem-solving and innovation rather than mundane, day-to-day tasks.

Differences Between Virtualization and Containerization

1. Isolation

Virtualization results in a fully isolated OS and VM instance, while containerization isolates the host operating system machine and containers from one another. However, all containers are at risk if an attacker controls the host.

2. Different Operating Systems

Virtualization can host more than one complete operating system, each with its own kernel, whereas containerization runs all containers via user mode on one OS.

3. Guest Support

Virtualization allows for a range of operating systems to be used on the same server or machine. On the other hand, containerization is reliant on the host OS, meaning Linux containers cannot be run on Windows and vice-versa.

4. Deployment

Virtualization means each virtual machine has its own hypervisor. With containerization, either Docker is used to deploy an individual container, or Kubernetes is used to orchestrate multiple containers across multiple systems.

5. Persistent Virtual Storage

Virtualization assigns a virtual hard disk (VHD) to each individual virtual machine, or a server message block (SMB) if shared storage is used across multiple servers. With containerization, the local hard disk is used for storage per node, with SMB for shared storage across multiple nodes.

6. Virtual Load Balancing

Virtualization means failover clusters are used to run VMs with load balancing support. Since containerization uses orchestration via Docker or Kubernetes to start and stop containers, it maximizes resource utilization. However, decommissioning for load balancing with containerization occurs when limits on available resources are reached.

7. Virtualized Networking

Virtualization uses virtual network adaptors (VNA) to facilitate networking, running through a master network interface card (NIC). With containerization, the VNA is split into multiple isolated views for lightweight network virtualization.

What are the requirements to become a DevOps Engineer?

When looking to fill out DevOps roles, organizations look for a clear set of skills. The most important of these are:

Experience with infrastructure automation tools like Chef, Puppet, Ansible, SaltStack or Windows PowerShell DSC.

Fluency in web languages like Ruby, Python, PHP or Java.

Interpersonal skills that help you communicate and collaborate across teams and roles.

Mention some of the core benefits of DevOps?

Faster development of software and quick deliveries.

DevOps methodology is flexible and adaptable to changes easily.

Compared to the previous software development models, confusion about the project is decreased due to increased product quality.

The gap between the development team and operation team is bridged. i.e, the communication between the teams has been increased.

Efficiency is increased by the addition of automation of continuous integration and continuous deployment.

Customer satisfaction is enhanced.

What is the difference between Git Merge and Git Rebase?

Here, both are merging mechanisms but the difference between the Git Merge and Git Rebase is, in Git Merge logs will be showing the complete history of commits.

However, when one does Git Rebase, the logs are rearranged. The rearrangement is done to make the logs look linear and simple to understand. This is also a drawback since other team members will not understand how the different commits were merged into one another.

What are the benefits of Automation Testing?

Supports execution of repeated test cases

Aids in testing a large test matrix

Enables parallel execution

Encourages unattended execution

Improves accuracy thereby reducing human generated errors

Saves time and money

What is Docker image?

Docker image is the source of Docker container. In other words, Docker images are used to create containers. Images are created with the build command, and they'll produce a container when started with run. Images are stored in a Docker registry such as registry.hub.docker.com because they can become quite large, images are designed to be composed of layers of other images, allowing a minimal amount of data to be sent when transferring images over the network.

References

[DevOps Ecosystem \(winsurtech.com\)](https://winsurtech.com)

[Top 120+ DevOps Interview Questions and Answers For 2023 | Edureka](#)

[Git \(git-scm.com\)](https://git-scm.com)

[What is continuous integration? | IBM](#)

[Jenkins](#)

[What is Jenkins? The CI server explained | InfoWorld](#)

[What Is Static Analysis? Static Code Analysis Overview | Perforce](#)

[Top 50 Cobertura interview questions and answers - DevOpsSchool.com](#)

[Top Test Automation Interview Questions and Answers in 2023 | Edureka](#)

[Puppet Tutorial - javatpoint](#)