

# Software Testing & Integration

---

## - Vineela

# **Session 1**

- **Introduction to software testing**
- **Why testing code is important**
- **Verification and validation**
- **Quality Assurance vs Quality Control vs Testing**
- **Principles of software testing**

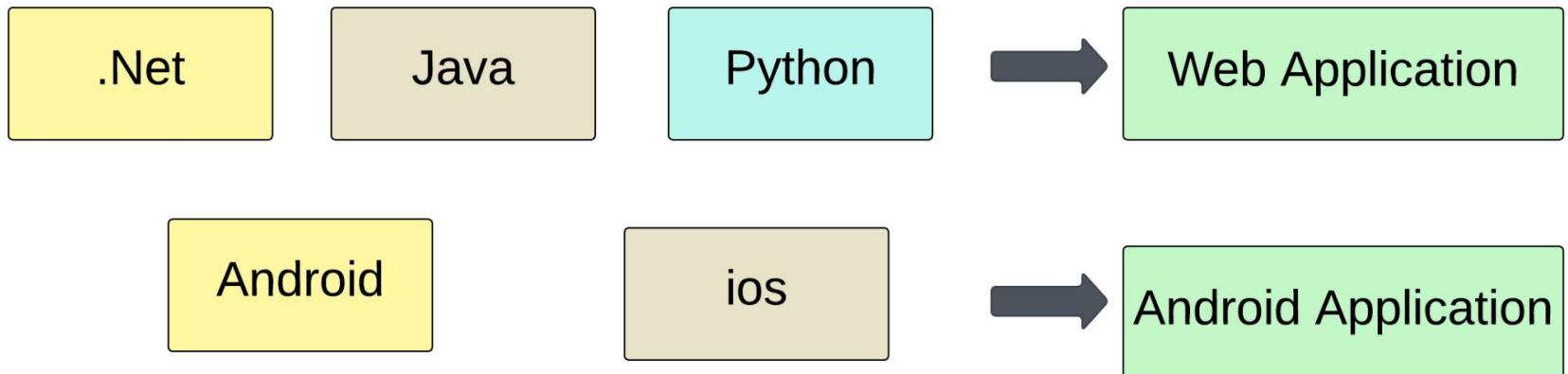
# Introduction to software testing

## What is Software Testing?

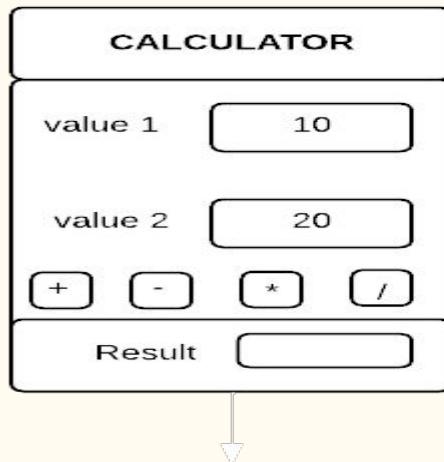
- It is the process of validating and checking of the developed software, whether its meeting the end-user customer requirement or not.
- It is the process of Identifying bugs in software.
- It involves running a program or system to identify and fix any errors or bugs



- Every application need to be tested irrespective of Technology and Business Domain needed to built up

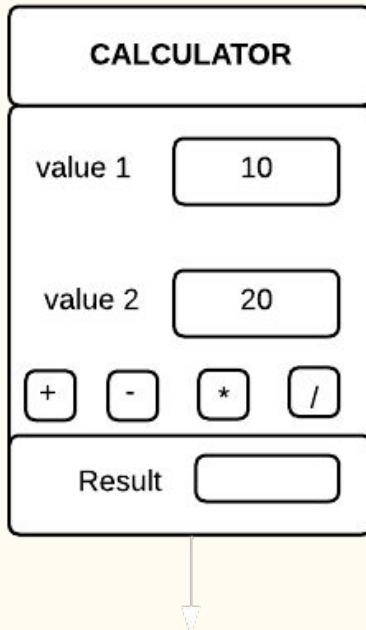


- **Error** - Any incorrect human action that produces a problem or wrong output in System
- **Defect/Bug** - Deviation of Actual result from the expected result
- **Failure** - Incorrect/Unexpected behaviour or output of an application



### **Sum Program**

```
int a,b,c,result;  
a = value1;  
b = value2;  
c = a*b;  
result = c;
```



## SUM Program

```
int a,b,c,result;  
a = value1;  
b = value2;  
c = a * b; * Error  
result = c;
```

Expected = 30  
Actual = 200

Defect

## Defects

### Major Defects

- Wrong Requirements by Business Analyst
- Wrong Design by Architect
- Wrong Coding by Developer

➤ Then What's Better Testing Approach?

**The Better Approach is to Test at every stage of Development**

## **Why testing code is important?**

- To find out defects and bugs at early stage of development
- Ensure Correctness and Quality
- Uncover security vulnerabilities
- To meet the compliance standards
- Consistent and Efficient user experience
- Immediate feedback addresses the issues and improve developers code

## Why Software Application have Defects?

- Complexity of Software
- Changing Requirements
- Human Error
- Environment Differences
- Insufficient Testing
- Integration Issues
- Poor Documentation and Communication
- Performance Issues
- Inadequate Requirements

## What to Test?



## What testing achieves?

- It ensures that the software is robust, secure, and user-friendly

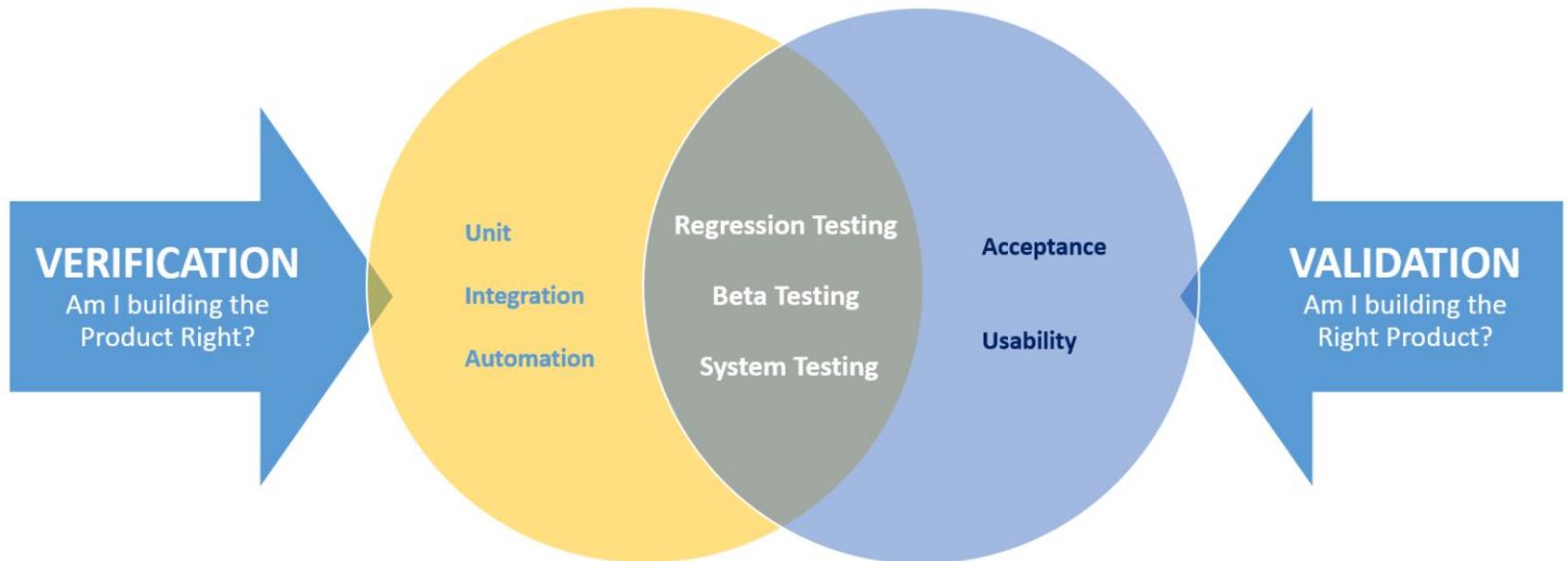
## **Verification and Validation:**

**Verification:** It involves a static analysis method (review) done without executing code.

It is the process of evaluation of the product development process to find whether specified requirements meet.

**Validation:** It involves dynamic analysis method (functional, non-functional), testing is done by executing code.

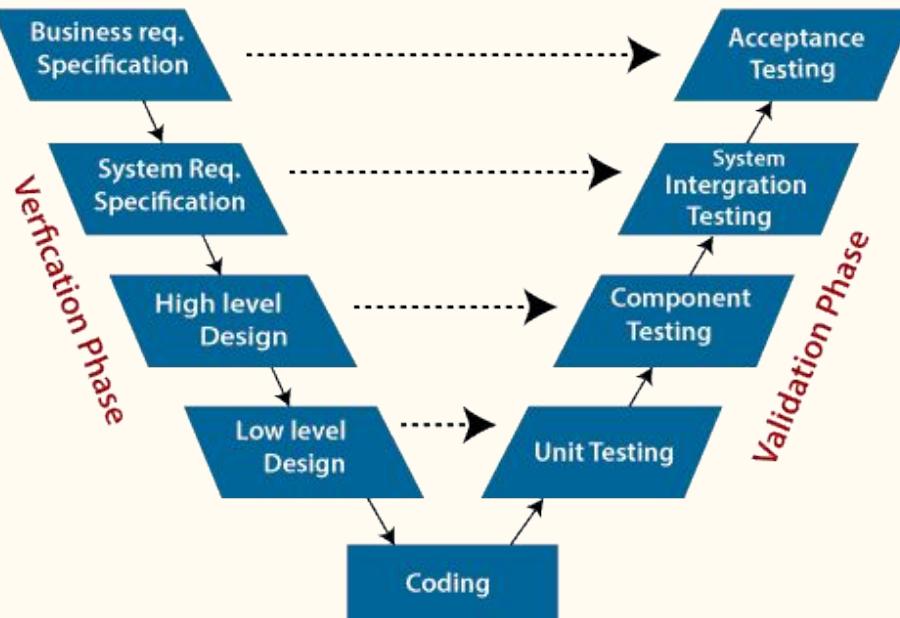
It is the process to classify the software after the completion of the development process to determine whether the software meets the customer expectations and requirements.



## V- Model

Developer's life Cycle

Tester's Life Cycle



# Quality Assurance vs Quality Control vs Testing

Aspect	Quality Assurance (QA)	Quality Control (QC)	Testing
Definition	Proactive process focused on improving processes to prevent defects.	Reactive process focused on detecting and fixing defects in the product.	The act of executing tests to find defects in the software.
Focus	Process and methodology improvement to ensure quality through audits, continuous improvement.	Identifying defects in the finished product through Inspections, reviews, defect management, testing	Detecting bugs through test cases through Manual and automated testing, performance testing, security testing

# Quality Assurance vs Quality Control vs Testing

Aspect	Quality Assurance (QA)	Quality Control (QC)	Testing
Activities	Process reviews, audits, creating standards, defining practices, training.	Product inspections, defect tracking, validating compliance.	Writing test cases, executing test scripts, reporting bugs.
Goal	Preventing defects by optimizing processes and standards.	Identifying and fixing defects in the product.	Verifying the correctness and functionality of the software.
Timing	Happens throughout the development lifecycle.	Occurs after the product has been built, before release.	Occurs during the development cycle, often after coding.

# 01

TESTING SHOWS THE  
PRESENCE OF DEFECTS,  
NOT THEIR ABSENCE

# 07

ABSENCE OF  
ERRORS FALLACY

# 02

EXHAUSTIVE  
TESTING IS  
IMPOSSIBLE



# 03

EARLY TESTING  
SAVES TIME AND  
MONEY



# 06

TESTING IS  
CONTEXT-DEPENDENT

# 05

BEWARE OF THE  
PESTICIDE PARADOX

## PRINCIPLES OF SOFTWARE TESTING

DEFECT  
CLUSTERING

# 04

## 7 Principles of Software Testing

# Session 2

- Introduction to STLC and V Model
- Types of testing: manual and automation
- Tools used for automation testing
- Introduction to testing methods: white-box, black-box and grey-box
- Introduction to functional testing
- Introduction to non-functional testing

# 6 Phases of the Software Development Life Cycle



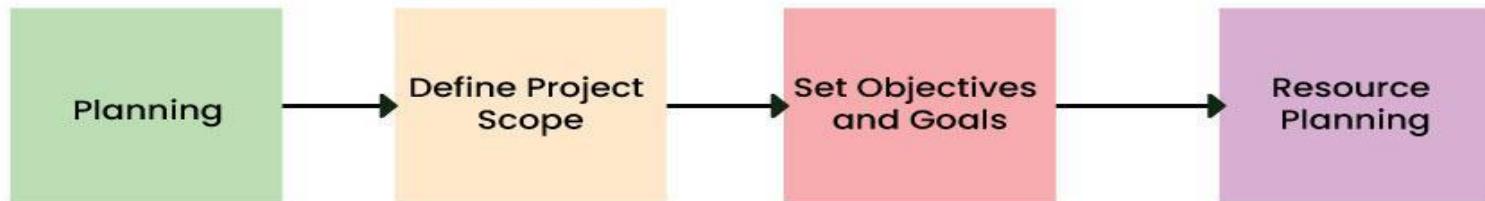
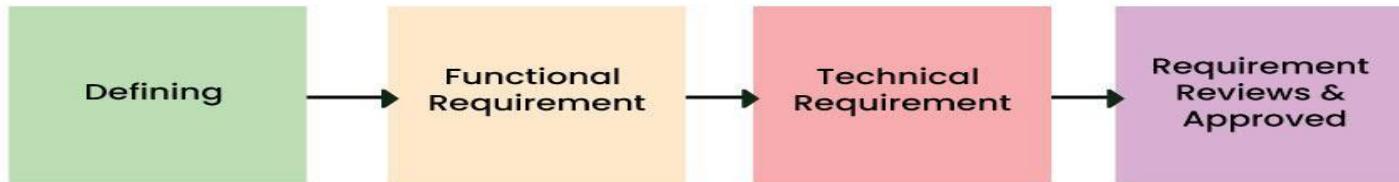
## **Phase-1: Requirement and Planning Analysis**

- This is the initial phase which defines project's scope, objectives and requirements
- Conduct interviews, surveys, and workshops to gather requirements.
- Understand and document the software requirements from stakeholders.

### **Key activities:**

- Requirement gathering
- Feasibility analysis
- Setting project timelines and budget
- Risk assessment

# Requirements and Planning Analysis



## **Phase-2: Design**

- This phase involves creating the system architecture, technical specifications and design documents based on the requirements gathered.

### **Key activities:**

- High-level system design (overall architecture)
- Low-level design (detailed design of modules, database schema, etc.)
- Design validation and reviews

# Design



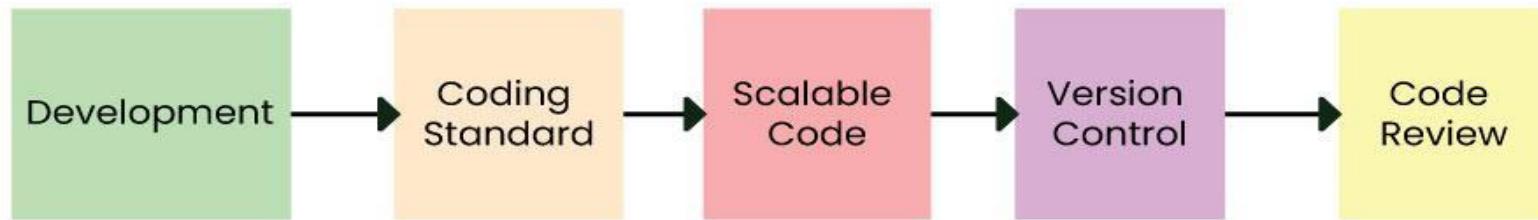
## **Phase-3: Development**

- During this phase, the actual code is written according to the design documents.
- It involves developers writing the source code and unit testing the individual components.

### **Key activities:**

- Coding according to design specifications
- Unit testing
- Integration with other modules

# Development



## **Phase-4: Testing**

- In this phase, the software is rigorously tested to identify and fix any bugs, errors, or issues.

### **Key activities:**

- Functional testing (does it meet the requirements?)
- Performance testing (how does it perform under stress?)
- Security testing
- User acceptance testing (UAT)
- Bug fixing and quality assurance

# Testing



## **Phase-5: Deployment**

- After testing is complete and the software is validated, it is deployed to the production environment where it will be available to users.

### **Key activities:**

- Installing the software on the production servers
- Ensuring that everything works as expected in the live environment

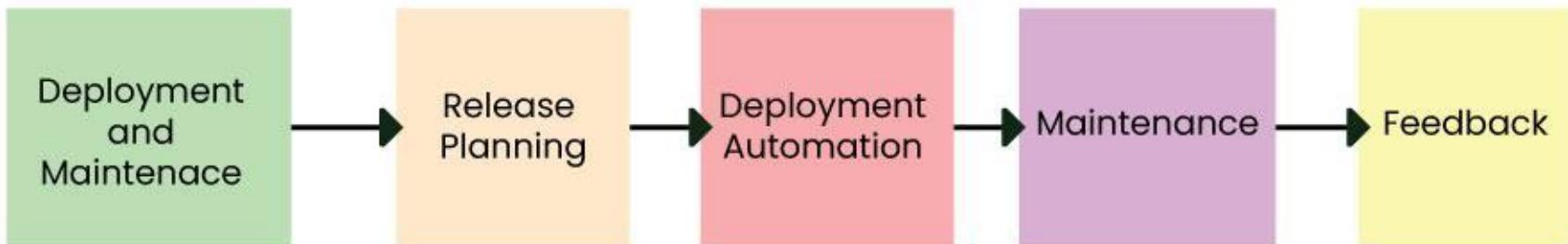
## **Phase-6: Maintenance**

- This phase involves ongoing support, updates, and modifications to the software post-deployment. Maintenance ensures the system remains functional and relevant over time.

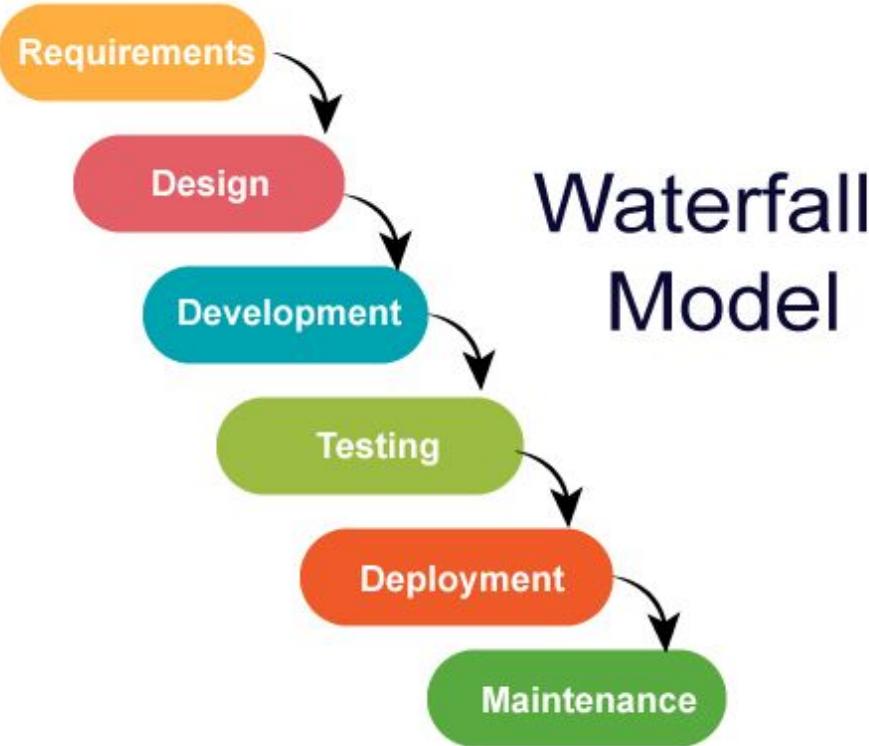
### **Key activities:**

- Bug fixes
- Regular updates
- Performance monitoring
- Handling user feedback and issues

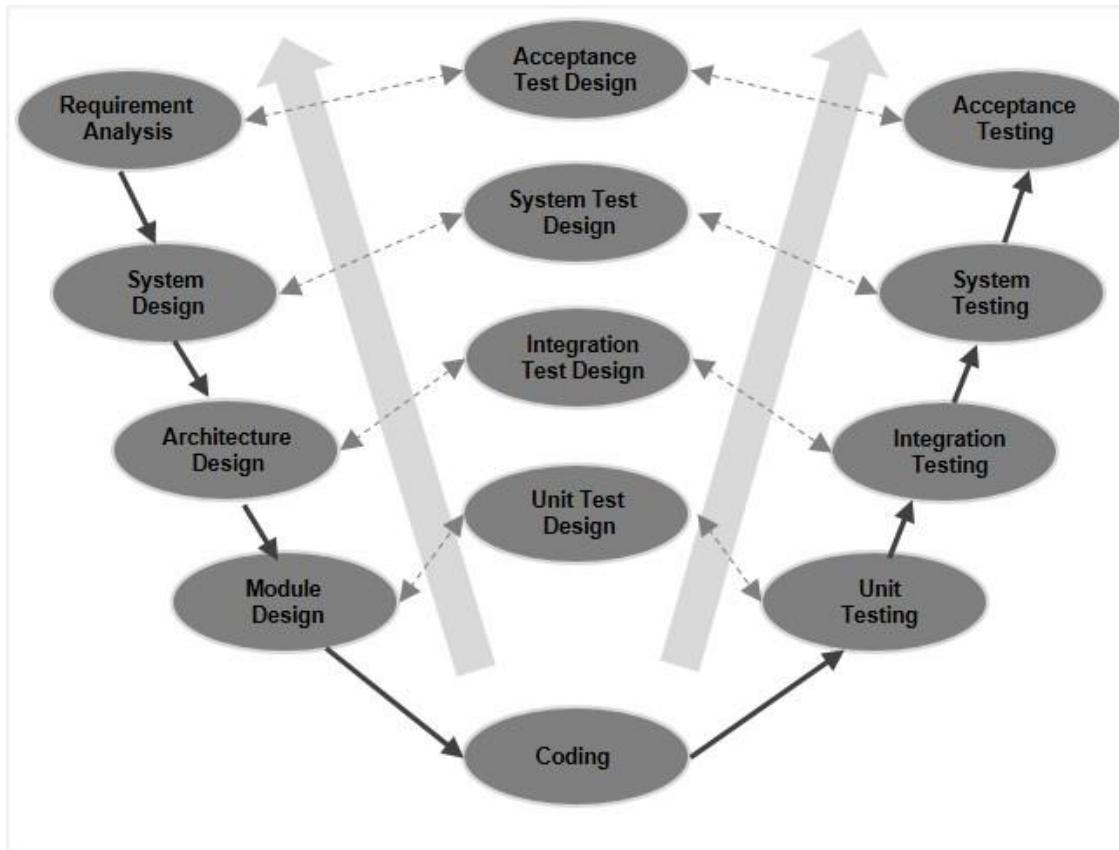
# Deployment and Maintenance



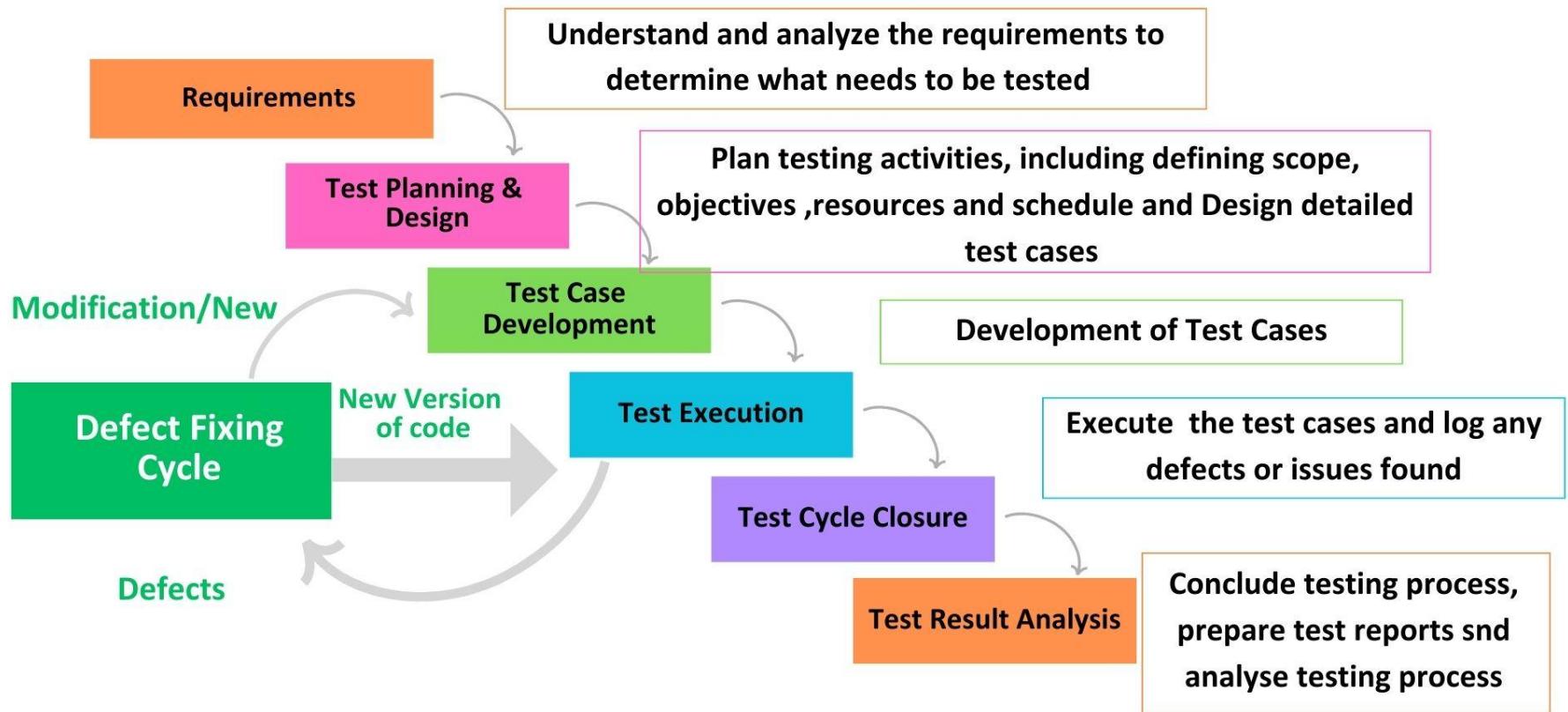
# Software Development Life Cycle (SDLC) Model

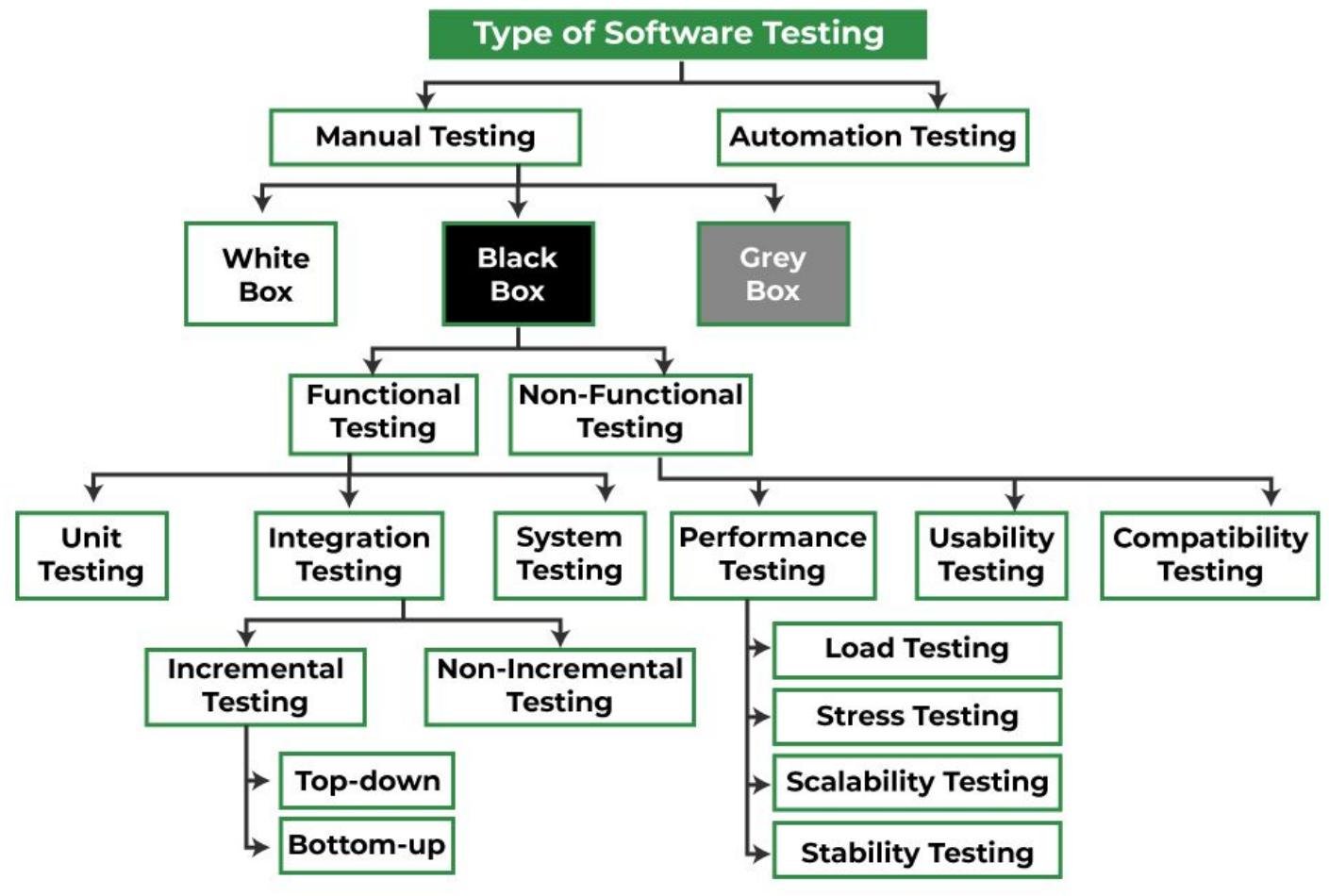


# Software Development Life Cycle (SDLC) V Model



# Software Testing Life Cycle





## **Types of Testing: manual and automation**

- **Manual testing** is the process of testing software applications manually and executing test cases without the use of automation tools.
- Testers perform test cases, identify bugs, and ensure that the software works as expected by following a manual, step-by-step process.

### **Tools used for Manual Testing:**

Test Link, Bugzilla, Jira ,LoadRunner, Apache J Meter, RedMine, FogBugz

## **Disadvantages of Manual Testing**

- Time consuming
- High changes of human error
- Less accuracy

## **Advantages of Manual Testing**

- No need for programming language
- Less Expensive

## Different Types of manual testing

### Black Box Testing

- Evaluate the performance and behaviour of software from user's point of view.
- Tester only has visibility into the inputs and outputs, not the internal code
- Also known as functional testing
- Includes System Testing, User Acceptance Testing

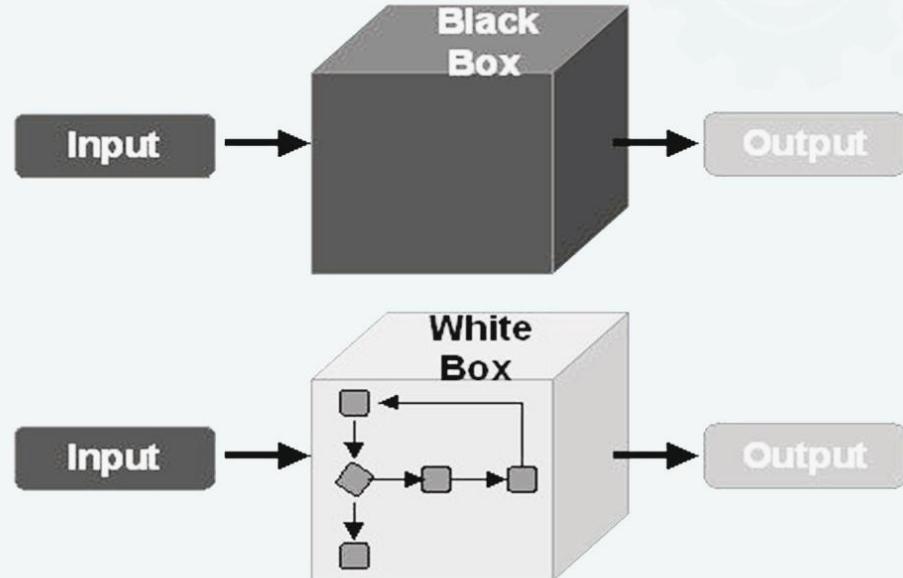
## **Different Types of manual testing**

### **White Box Testing**

- Evaluate the security and functionality of the product in more detail.
- This is performed by a developer or QA professional with an understanding of the internal code behind the software.
- Also known as structural testing or clear-box testing
- Includes Unit Testing, Integration Testing

## Introduction to testing methods: white-box, black-box and grey-box

- Tester provides input to the system and observes whether the output matches the expected results.
- Test cases are derived from the requirements and use cases, ensuring that the software meets the functional requirements



# Introduction to testing methods: white-box, black-box and grey-box

## Black Box Testing Coverage:

- **Equivalence Partitioning:** This technique divides the input data into partitions (or equivalence classes) where test cases are selected from each partition. The idea is that all values within a partition will be treated similarly by the system, so testing one value from each partition is sufficient.
- **Boundary Value Analysis:** Focuses on testing values at the boundaries of input ranges.  
For example, if an input field accepts values between 1 and 10, boundary values would include 1, 10, and values just above or below these limits (e.g., 0 and 11).

# Introduction to testing methods: white-box, black-box and grey-box

## White Box Testing Coverage:

- **Statement Coverage:** Ensures that every line of code in the program is executed at least once during testing.
- **Branch Coverage:** Tests all possible branches (e.g., if and else conditions) of the code, ensuring that both true and false branches are executed.
- **Path Coverage:** Focuses on testing all possible paths that can be taken in the code, which can be significantly more complex than branch or statement coverage, especially in large programs.

# Introduction to testing methods: white-box, black-box and grey-box

## White Box Testing Coverage:

- **Condition Coverage:** Ensures that every boolean expression in the program is tested for both true and false values.
- **Function Coverage:** Ensures that every function in the program is called at least once during testing.
- **Loop Coverage:** Ensures that loops are tested for different scenarios, such as zero iterations, one iteration, and multiple iterations.

# Introduction to testing methods: white-box, black-box and grey-box

## Advantages of Black Box Testing:

- **Non-technical Focus:** Testers do not need to have programming knowledge, making it accessible for non-technical personnel or end-users to conduct testing.
- **User Perspective:** Since the testing is based on the system's functionality and user requirements, it simulates how real users will interact with the system.
- **Effective for Large Systems:** Black box testing is particularly useful for large applications, as it allows testing of complex systems without requiring knowledge of the intricate details of the code.

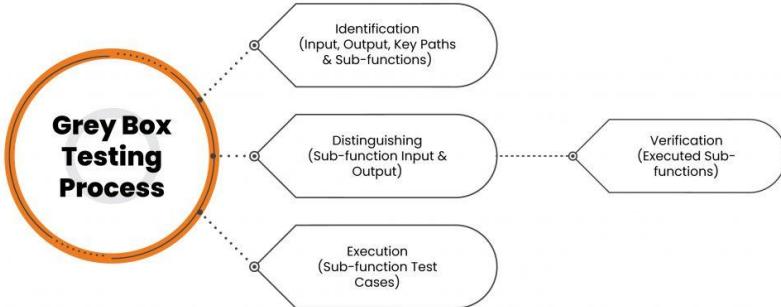
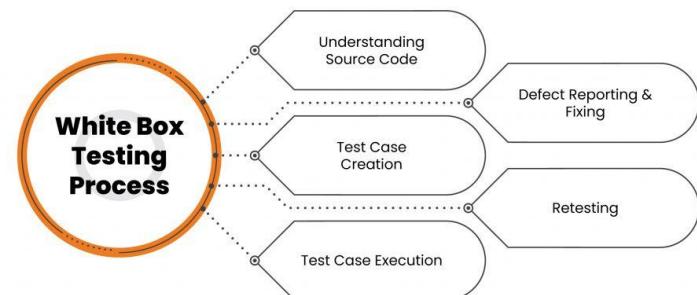
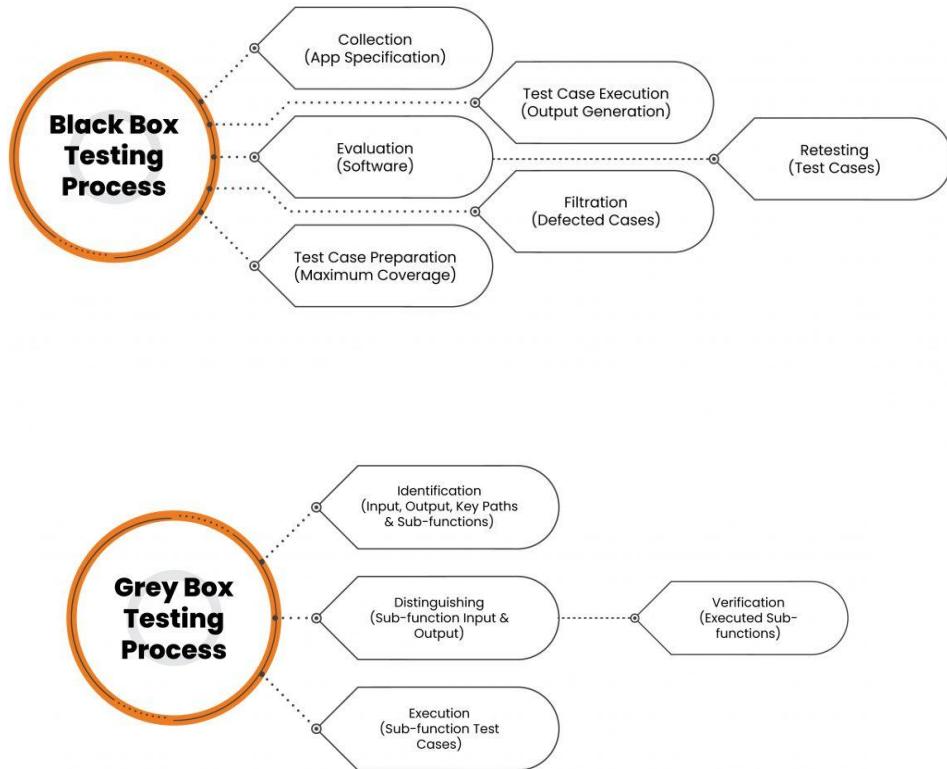
# Introduction to testing methods: white-box, black-box and grey-box

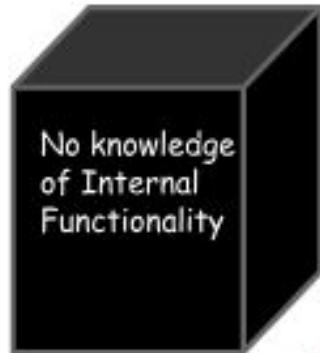
## Disadvantages of Black Box Testing:

- **Limited Code Coverage:** Black box testing does not guarantee that all code paths are covered. It may miss issues related to internal system logic, unused code, or paths that are not exercised by the inputs.
- **Redundancy:** There may be overlapping or redundant test cases, especially when multiple test cases produce similar outputs for different inputs.
- **Dependence on Requirements:** The quality of testing depends heavily on the completeness and clarity of the functional requirements. If the requirements are incomplete or unclear, the tests may not be comprehensive.

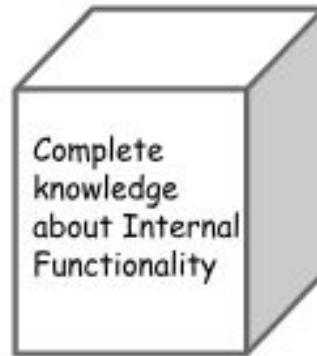
## Testing levels

- **Unit Testing:** Testing individual components or units of code to ensure that they function correctly. Typically done by developers.
- **Integration Testing:** Checking the interaction between multiple units or systems to ensure they work together as expected.
- **System Testing:** Testing the entire system as a whole to validate its functionality against the requirements.
- **User Acceptance Testing:** Ensuring that the software meets the acceptance criteria or business requirements, often performed by end users or clients.
- **Regression Testing:** Automating tests to verify that new code changes haven't broken existing functionality.

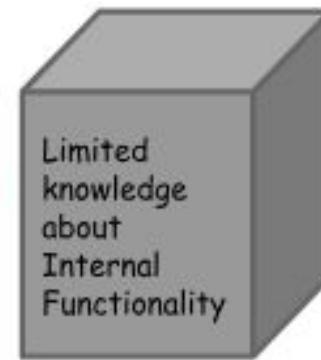




*Black Box*



*White Box*



*Grey Box( or Gray  
box)*

# Functional Testing

- Functional testing is a type of software testing that validates the software system against the functional requirements or specifications.
- They are performed both manually and using automation tools

## Types of functional testing

### Unit Testing

- To Test individual components or units of the software to ensure they work as intended
- It is performed by developers or QA professionals
- **Ex** - Testing a function in a payment processing system to ensure it calculates correct total amount
- **Tools :**            

# Types of Functional Testing

## Integration Testing

- Tests the interactions between integrated units/modules to ensure they function together correctly.

### Types

- **Big Bang** : All modules are integrated and tested at once
- **Top-Down** : All modules are integrated from top to bottom
- **Bottom-Up** : Modules are integrated from bottom to top

**Ex -** Testing the interaction between the login module and the user dashboard in a web application

- Tools

JUnit



TestNG

# Types of Functional Testing

## System Testing

- Tests the complete and integrated software system to verify that it meets the specified requirements.

**Ex -** Testing the entire online banking system to ensure all modules (login,fund transfer)

### Tools:



# **Types of Functional Testing**

## **Acceptance Testing**

- Validates the software against user requirements and checks if it is ready for deployment.
- This is done right before deploying the product

### **Types:**

**Alpha Testing** - Conducted by the developers at the development site

**Beta Testing** - Conducted by end-users at the client site

# Types of Functional Testing

## Acceptance Testing

### User Acceptance Testing (UAT)

- Customer performs test randomly and accepts/rejects the product

## Reliability Testing

- The behaviour of the application is tested under abnormal conditions such as “How does the system behave when system is switched off?”

**Ex -** Testing a new feature in a mobile app by end-users before releasing to the public

**Tools :**



## **Types of Functional Testing**

### **Smoke Testing**

- To verify the most critical functions of the software work properly

**Ex -** Performing a quick check on a new build of a mobile app to ensure it launches and the main features are accessible

**Tools:**



## Types of Functional Testing

### Sanity Testing

- To verify that new functionality or bug fixes work as expected

**Ex** - Testing a recently fixed bug in a customer service application to ensure the issue is resolved

Tools:



## **Functional Testing - E-Commerce Website**

**Unit Testing :** Testing the function that calculates discounts on products.

**Integration Testing :** verifying the interaction between the shopping cart and payment gateway modules.

**System Testing :** Ensuring that the entire checkout process works correctly.

**Acceptance testing :** validating the new search feature with end-users before deployment.

**Regression Testing :** Checking that existing payment methods still function after adding a new payment option

## **Non-functional Testing**

- It is the testing of non-functional aspects of an application such as performance , reliability,usability,security and so on
- Generally , tests are executed using tools
- They are performed after performing functional tests
- It Improves the quality of the software
- It allows you to polish the software

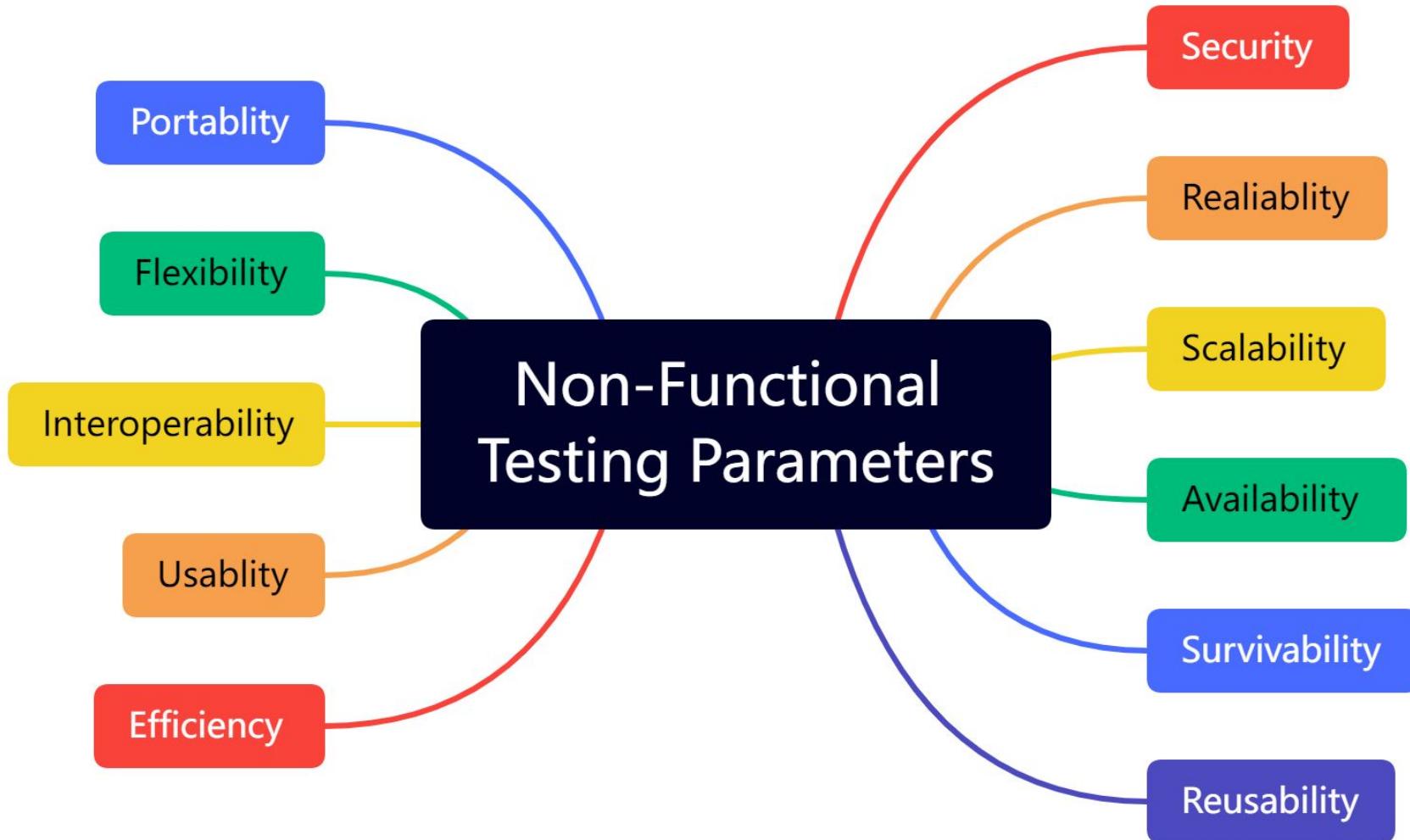
## Types of Non-functional testing

**Performance Testing** - Ensure the software performs well under expected workloads

- **Load Testing** - Evaluate the software's behaviour under expected load conditions
  - Ex** - Testing an ecommerce website response time when 1000 users browse products simultaneously
  
- **Stress Testing** - Determines the software's stability under extreme conditions beyond normal operational capacity
  - Ex** - Testing a banking application to see how it handles 10000 simultaneous transactions

## Types of Non-functional testing

- **Security Testing** - To identify vulnerabilities and ensure the software is protected against attacks
- **Usability Testing** - To ensure the software is user-friendly and provides a good user experience
- **Reliability Testing** - To ensure the software performs consistently under specific conditions
- **Compatibility Testing** - To ensure the software works well across different environments
- **Scalability Testing** - To determine the software's ability to scale up or down based on the demand



## **Non Functional Testing - E-Commerce Website**

**Performance Testing :** Assessing the website's response time during Sale Season

**Security Testing :** Conducting penetration tests to identify vulnerabilities in payment processing

**Usability Testing :** Ensuring the checkout process is quick for users

## Automation

Automated testing involves the use of specialized software tools to run test cases, which can significantly speed up the testing process and improve the accuracy and consistency of tests.

## Types of Automated Testing

- **Unit Testing:** Automation of unit tests to check individual code components.
- **Regression Testing:** Automating tests to verify that new code changes haven't broken existing functionality.
- **Load Testing:** Simulating a large number of users to check how the application behaves under stress or heavy load.
- **Functional Testing:** Ensuring that specific features of the application function correctly, based on pre-defined inputs and outputs.

## Automation Tools :

Lambda test, TestProject,Katalon Studio, Test IM, Ranorex Studio,Rapise,Squish



Aspect	Manual Testing	Automation Testing
<b>Execution approach</b>	Testers execute test cases manually.	Test cases are executed using automated tools.
<b>Human involvement</b>	Requires human intervention for each step.	Minimal human intervention once scripts are set up.
<b>Test case creation</b>	Test cases are created manually.	Test cases are scripted and automated.
<b>Speed and efficiency</b>	Relatively slower due to manual execution.	Faster and more efficient, especially for repetitive tests.
<b>Test data handling</b>	Test data is entered manually by testers.	Test data can be generated and managed automatically.
<b>Test coverage</b>	Subject to human bias and limitations.	Can achieve extensive test coverage with consistent precision.
<b>Scalability</b>	Less scalable for large-scale projects.	Highly scalable for large and complex projects.
<b>Human observation</b>	Enables human intuition and exploratory testing.	Lacks human intuition but excels in repetitive tasks.
<b>Suitable scenarios</b>	Exploratory testing, usability testing, ad-hoc testing.	Regression testing, load testing and data-driven testing.
<b>Initial investment</b>	Lower initial investment, minimal tool requirement.	Higher initial investment for selecting and setting up tools.
<b>Maintenance effort</b>	Moderate maintenance effort for test case updates.	Regular maintenance needed for test script updates.

## Test Case

- A test case is standard format adhered to in testing to check if the software is working as per the requirements.
- It comprises a collection of conditions that are required to be verified to validate if the actual outcomes on the software are matching with the expected ones.

The various sections of a test case are listed below –

- **Functionality Name** – It points to the functionality which is verified.
- **Test Case Identifier** – It points to a unique id assigned to each test case.
- **Name of Tester** – The name of the tester who will carry out the testing.
- **Test Scenario Name** – The name of the scenario which will be converted into a test case.

## Test Case

- **Test Case Description** – The requirement or the criteria which is going to be verified in the test case.
- **Test Steps** – The actions to be performed on the software to verify the test conditions.
- **Preconditions** – The prerequisites that need to be satisfied before starting with the test steps.
- **Test Priority** – The test priority is assigned to the test case to decide the order in which it needs to be executed.
- **Test Data** – The inputs and data which are needed to complete the test.
- **Expected Outcome** – The expected results as per the requirements.

## Test Case

- **Test Settings** – The settings and parameters that are needed to be set before running the test.
- **Actual Outcome** – The actual result produced on the software.
- **Environment** – The configuration of the environment on which the test is to be performed including the operating system, configurations, security etc.
- **Test Status** – The status of the test should reflect as pass, fail, not executed and blocked.
- **Comments** – The comments added against each test case.

New Microsoft Excel Worksheet - Microsoft Excel

1	<b>Test case template</b>					
2	test case name	Delta-3.0-ICICI-Login				
3	test case type	Functional test case				
4	requirement no		1			
5	module	login				
6	status	XXX				
7	severity	critical				
8	release	Delta				
9	version		3			
10	pre-condition	required one login				
11	test data	username-abc, password-123				
12	summary	to check the functionality of login				
13	<b>Steps no</b>	<b>Description</b>	<b>Inputs</b>	<b>Expected result</b>	<b>Actual results</b>	<b>Status</b>
14	1	open "Browser" and enter the "Url"	https://QA/Main//	Login page must be display	As Expected	pass
15	2	enter the following values for "Username":				XXX
16		Valid(abc)	abc	Accpect	Login page must be disp	pass
17		Invalid		555 Error message "invalid login"	not as expected	fail
18		Blank	Null	Error message username cannot	be not as expected	fail
19		Symbols	2 alphabet	Error message invalid login	not as expected	fail
20	3	enter the following values for "Password":				
21		valid		123 Accpect	Login page must be disp	pass
22		invalid	xy3	Error message invalid login	not as expected	fail
23		Blank		Error message password cannot	be not as expected	fail
24		enter the valid username and password and				
25	4	click on "OK" button	abc,123	"home Pag " must be displayed	home page is displayin	pass
26		enter the valid username and password and				
27	5	click on "Cancel" button	abc,123	all field must be cleared	the entered data is clea	pass
28	author	test engineer name				XXX
29	date		1/4/2020			
30	reviewed by	ryan				
31	appproved by	jessica				
32						

# Session 3

- **Introduction to Selenium (use Eclipse IDE)**
- **Load web driver**
- **Create selense commands: locators: by ID, name, class, tag name, XPath**
- **Add interactions: text box, radio button selection, check box selection, drop down item selection, keyboard actions, mouse actions, multi select**

# Introduction to Selenium (use Eclipse IDE)

- Selenium is a popular open-source web-based functional automation tool
- Provides a single interface that lets you to write test scripts in programming languages like Ruby,java,Nodejs,C# and others
- Using Selenium framework, testers are able to deliver test cycles faster by automating reported test cases
- Selenium Webdriver allows you to directly interact with the browsers through automation test scripts
- Selenium Webdriver supports operating systems- Windows, Mac OS,iOS and Solaris
- Selenium Webdriver also known as Selenium 2

## Getting Started



### Selenium WebDriver

If you want to create robust, browser-based regression automation suites and tests, scale and distribute scripts across many environments, then you want to use Selenium WebDriver, a collection of language specific bindings to drive a browser - the way it is meant to be driven.

[READ MORE ▶](#)

### Selenium IDE

If you want to create quick bug reproduction scripts, create scripts to aid in automation-aided exploratory testing, then you want to use Selenium IDE; a Chrome, Firefox and Edge add-on that will do simple record-and-playback of interactions with the browser.

[READ MORE ▶](#)

### Selenium Grid

If you want to scale by distributing and running tests on several machines and manage multiple environments from a central point, making it easy to run the tests against a vast combination of browsers/OS, then you want to use Selenium Grid.

[READ MORE ▶](#)

# Selenium Architecture

There are four components of Selenium Architecture:

## **Selenium Client Library**

- Selenium supports multiple libraries such as Java, Ruby, Python, etc.,
- Selenium Developers have developed language bindings to allow Selenium to support multiple languages.

## **JSON Wire Protocol over HTTP**

- JSON is used to transfer data between a server and a client on the web.
- JSON Wire Protocol is a REST API that transfers the information between HTTP server.
- Each WebDriver (such as FirefoxDriver, ChromeDriver etc.,) has its own HTTP server.

# Selenium Architecture

## Browser Drivers -

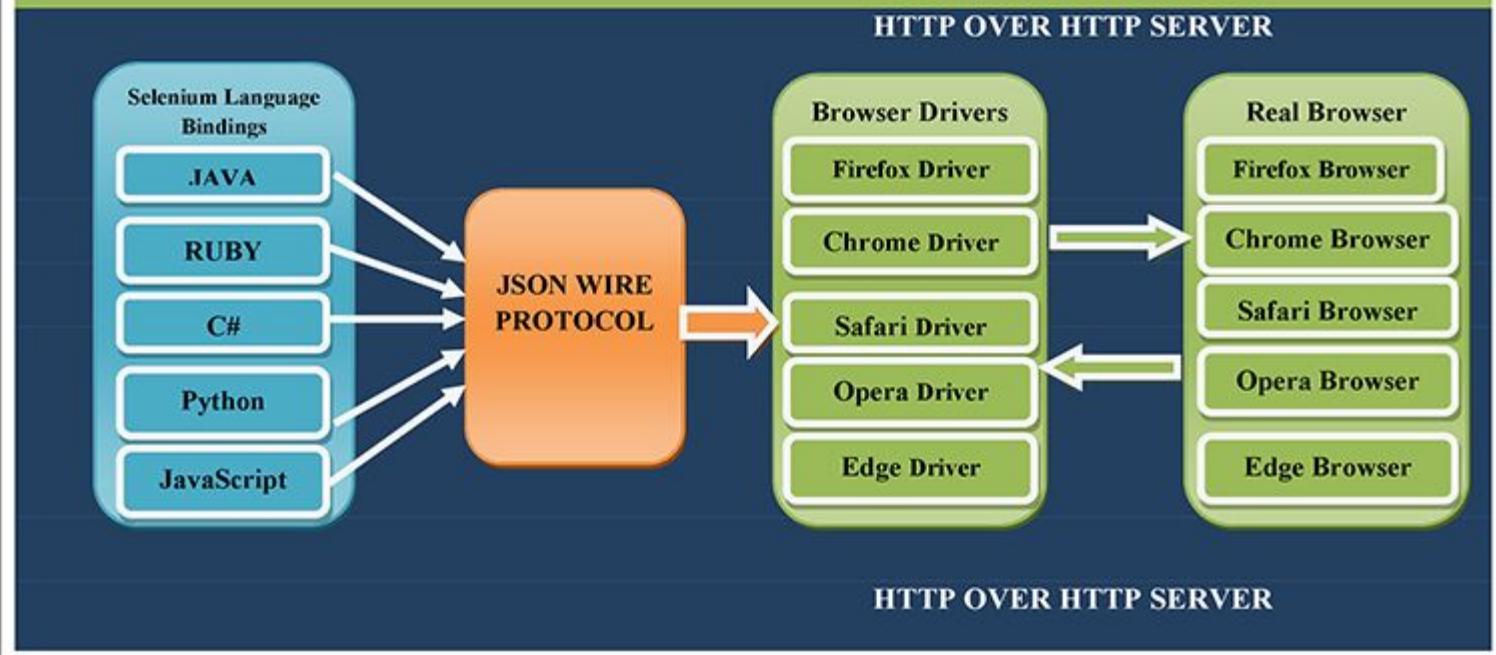
- Each browser contains separate browser driver. Browser drivers communicate with respective browser without revealing the internal logic of browser's functionality.
- When a browser driver is received any command then that command will be executed on the respective browser and the response will go back in the form of HTTP response.

## Browsers

- Selenium supports multiple browsers such as Firefox, Chrome, IE, Safari etc.,

**Example:**    WebDriver driver = new FirefoxDriver();  
                  driver.get(<https://www.google.com>)

# Selenium Web Driver Architecture



## Selenium Setup Steps

- 1) Install Java - <https://www.oracle.com/in/java/technologies/downloads/>
- 2) Install Eclipse IDE - <https://www.eclipse.org/downloads/>
- 3) Any supported browser and install drivers
- 4) Associate Selenium to your java project

<https://www.selenium.dev/downloads/>

## **Advantages of Selenium**

- Free and Open Source
- Supports multiple languages
- Supports multiple browsers
- Supports almost every OS
- Integrated with CI/CD
- Run Tests Concurrently

## Disadvantages of Selenium

- We can't perform regression testing and Re-testing
- Static actions like GUI testing and availability of fields is not possible
- Selenium has limited support for desktop applications, making it more suitable for web application testing.
- Maintenance efforts are required to update test scripts as web applications evolve and change.
- Dependency on browser updates can cause compatibility issues and require regular updates of Selenium drivers to ensure compatibility with the latest browser versions.
- It is not providing detailed test report to analyse bugs

## + - Browsers

**Firefox**

GeckoDriver is implemented and supported by Mozilla, refer to their [documentation](#) or supported versions.

**Internet Explorer**

Only version 11 is supported, and it requires additional [configuration](#).

**Safari**

SafariDriver is supported directly by Apple, for more information, check their [documentation](#)

**Opera**

OperaDriver is supported by Opera Software, refer to their [documentation](#) for supported versions.

**Chrome**

ChromeDriver is supported by the Chromium project, please refer to their [documentation](#) for any compatibility information.

**Edge**

Microsoft is implementing and maintaining the Microsoft Edge WebDriver, please refer to their [documentation](#) for any compatibility information.

When writing driver scripts, you need to make use of language-specific client drivers.

While language bindings for [other languages exist](#), these are the core ones that are supported by the main project hosted on GitHub.



.NET/C#

Stable: [4.27.0 \(November 25, 2024\)](#)

[Changelog](#)

[API Docs](#)

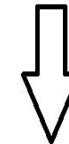


Ruby

Stable: [4.27.0 \(November 25, 2024\)](#)

[Changelog](#)

[API Docs](#)



Java

Stable: [4.27.0 \(November 25, 2024\)](#)

[Changelog](#)

[API Docs](#)



## Load web driver

- Meaning of driver is page where driver focus is available during runtime
- Selenium provided
  - firefox driver
  - chrome driver
  - safari driver
- All these classes are the implementation of interface/super class i.e.,web driver
- This web driver interface provided the common operations such as accessing web URL,navigating the browser, closing the browser and so on

```
// Set the path for the ChromeDriver  
  
System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");  
  
// Create a new instance of the Chrome driver  
  
WebDriver driver = new ChromeDriver();
```

## Create selense commands: locators: by ID, name, class, tag name, XPath

### Locator by ID

- This locator is used to locate an element by its **id** attribute
- The id is **unique** for each element on a webpage

**Syntax** - WebElement elementById = driver.findElement(By.id("elementId"));

### Locator by Name

- This locator is used to find an element by its **name** attribute
- The name attribute can be shared by multiple elements, but it's often used for form fields

**Syntax** - WebElement elementByName =  
driver.findElement(By.id("elementName"));

# Create selense commands: locators: by ID, name, class, tag name, XPath

## Locator by Class Name

- This locator is used to find elements by their **class** attribute
- This can be useful when you want to select elements with a specific class, but be aware that multiple elements can share the same class.

**Syntax** - WebElement elementByClassName =

```
driver.findElement(By.className("elementClass"));
```

## Create selense commands: locators: by ID, name, class, tag name, XPath

### Locator by Tag Name

- This locator is used to locate an element by its **tag** name (such as <div>, <button>, <input>, etc.)

### Syntax

```
WebElement elementByTagName = driver.findElement(By.tagName("div"));
```

## Create selense commands: locators: by ID, name, class, tag name, XPath

### Locator by XPath

- The locator is used to locate an element using an **XPath** expression.
- XPath allows for complex queries and is very flexible, but may be slower compared to other locators if not written optimally.
- This is faster in identifying element since we can use any property of element in xpath to make selenium to identify any element even without class, name or id attributes.

**Syntax** - WebElement elementByXPath =  
driver.findElement(By.xpath("//div[@class='example']"));

## HTML Code Example

Html

```
<!DOCTYPE html>
<html>
<head>
    <title>Sample Page</title>
</head>
<body>
    <h1 id="header">Welcome to My Page</h1>
    <input type="text" name="username" class="input-field" />
    <button class="btn-submit">Submit</button>
    <div class="container">
        <p>Sample Text</p>
        <a href="#" id="link">Click Me</a>
    </div>
</body>
</html>
```

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class SeleniumLocatorsExample{
    public static void main(String[] args) {
        // Set the path for the ChromeDriver
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

        // Create a new instance of the Chrome driver
        WebDriver driver = new ChromeDriver();

        // Launch the URL of the HTML file
        driver.get("path/to/your/htmlfile.html");

        // Locate elements using different strategies
        WebElement header = driver.findElement(By.id("header"));
        WebElement username = driver.findElement(By.name("username"));
        WebElement inputField = driver.findElement(By.className("input-field"));
        WebElement button = driver.findElement(By.tagName("button"));
        WebElement link = driver.findElement(By.xpath("//a[@id='link']"));

        // Perform actions on the located elements (e.g., click, send keys, etc.)
        username.sendKeys("myusername");
        button.click();

        // Close the browser
        driver.quit();
    }
}
```

# Add interactions: text box, radio button selection, check box selection, drop down item selection, keyboard actions, mouse actions, multi select

## Text Box Interaction (Entering Text)

- You can send text to a text box using the sendKeys() method.

```
WebElement textBox = driver.findElement(By.id("text-box-id"));  
textBox.sendKeys("Hello, Selenium!");
```

## Radio Button Selection

- To select a radio button, find the radio button using a locator and click it.

```
WebElement radioButton = driver.findElement(By.id("radio-button-id"));  
radioButton.click();
```

# Add interactions: text box, radio button selection, check box selection, drop down item selection, keyboard actions, mouse actions, multi select

## Checkbox Selection

- Similarly to radio buttons, checkboxes can be clicked to select or unselect.

```
WebElement checkBox = driver.findElement(By.id("checkbox-id"));
checkBox.click();
```
- If you need to check whether a checkbox is already selected, you can use the `isSelected()` method.

## Usage of isSelected()

```
if (!checkBox.isSelected()) {  
    checkBox.click(); // Click to select it  
}  
  
if (!checkBox.isSelected()) {  
    checkBox.click(); // Click to select it  
}
```

**Add interactions: text box, radio button selection, check box selection, drop down item selection, keyboard actions, mouse actions, multi select**

## **Dropdown Item Selection**

- For dropdowns, you typically use the Select class provided by Selenium, which allows you to select options based on visible text, value, or index.

```
import org.openqa.selenium.support.ui.Select;  
  
import org.openqa.selenium.WebElement;  
  
WebElement dropdown = driver.findElement(By.id("dropdown-id"));  
  
Select select = new Select(dropdown);  
  
// Select by visible text  
  
select.selectByVisibleText("Option 1");  
  
// Select by value  
  
select.selectByValue("option1");  
  
// Select by index  
  
select.selectByIndex(1);
```

# Add interactions: text box, radio button selection, check box selection, drop down item selection, keyboard actions, mouse actions, multi select

## Keyboard Actions

- Selenium provides the Actions class and the Keys class for keyboard-related interactions, such as pressing keys or performing key combinations.

### Pressing a Single Key:

```
import org.openqa.selenium.Keys;
```

```
import org.openqa.selenium.interactions.Actions;
```

```
Actions actions = new Actions(driver);
```

```
actions.sendKeys(Keys.ENTER).perform();
```

**Add interactions: text box, radio button selection, check box selection, drop down item selection, keyboard actions, mouse actions, multi select**

## **Keyboard Actions**

- **Typing Text (Simulating a KeyPress):**

```
WebElement inputField = driver.findElement(By.id("input-field-id"));
```

```
inputField.sendKeys("Some text to type" + Keys.ENTER);
```

- **Pressing a Combination of Keys (e.g., Ctrl+C for Copy):**

```
actions.keyDown(Keys.CONTROL).sendKeys("c").keyUp(Keys.CONTROL).perform();
```

# Add interactions: text box, radio button selection, check box selection, drop down item selection, keyboard actions, mouse actions, multi select

## Mouse Actions (Click, Move, Hover, Drag & Drop)

- Selenium's Actions class provides methods for mouse actions like click, move, hover, and drag-and-drop.

### Click on an Element:

```
WebElement element = driver.findElement(By.id("button-id"));
```

```
Actions actions = new Actions(driver);
```

```
actions.click(element).perform();
```

# **Mouse Actions (Click, Move, Hover, Drag & Drop)**

## **Hover over an Element (Mouseover):**

```
WebElement element = driver.findElement(By.id("hover-element-id"));
```

```
Actions actions = new Actions(driver);
```

```
actions.moveToElement(element).perform();
```

## **Drag and Drop:**

```
WebElement source = driver.findElement(By.id("source-id"));
```

```
WebElement target = driver.findElement(By.id("target-id"));
```

```
Actions actions = new Actions(driver);
```

```
actions.dragAndDrop(source, target).perform();
```

# Mouse Actions (Click, Move, Hover, Drag & Drop)

## Multi-Select Dropdown (Handling Multiple Selections)

For multi-select dropdowns, you can use the `Select` class to select multiple options.

```
WebElement multiSelectDropdown =  
    driver.findElement(By.id("multi-select-dropdown-id"));  
  
Select select = new Select(multiSelectDropdown);  
  
// Check if it supports multiple selections  
  
if (select.isMultiple()) {  
  
    select.selectByIndex(0);  
  
    select.selectByVisibleText("Option 2");  
  
    select.selectByValue("option3"); }
```

```
import org.openqa.selenium.*;
import org.openqa.selenium.support.ui.*;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class SeleniumInteractions {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();

        driver.get("http://example.com");

        // Text Box Interaction
        WebElement textBox = driver.findElement(By.id("text-box-id"));
        textBox.sendKeys("Hello, Selenium!");

        // Radio Button Interaction
        WebElement radioButton = driver.findElement(By.id("radio-button-id"));
        radioButton.click();

        // Checkbox Interaction
        WebElement checkBox = driver.findElement(By.id("checkbox-id"));
        if (!checkBox.isSelected()) {
            checkBox.click();
        }

        // Dropdown Interaction
        WebElement dropdown = driver.findElement(By.id("dropdown-id"));
        Select select = new Select(dropdown);
        select.selectByVisibleText("Option 1");

        // Keyboard Action (e.g., Enter key)
        WebElement inputField = driver.findElement(By.id("input-field-id"));
        inputField.sendKeys("Some text" + Keys.ENTER);

        // Mouse Hover Action
        WebElement hoverElement = driver.findElement(By.id("hover-element-id"));
        Actions actions = new Actions(driver);
        actions.moveToElement(hoverElement).perform();

        // Multi-select Dropdown Interaction
        WebElement multiSelectDropdown = driver.findElement(By.id("multi-select-dropdown-id"));
        Select multiSelect = new Select(multiSelectDropdown);
        if (multiSelect.isMultiple()) {
            multiSelect.selectByIndex(1);
            multiSelect.selectByValue("option2");
        }

        driver.quit();
    }
}
```

## Summary of the Methods:

- **Text Box:** sendKeys()
- **Radio Button:** click()
- **Checkbox:** click() and isSelected()
- **Dropdown:** Select class (with methods like selectByVisibleText(), selectByIndex(), etc.)
- **Keyboard:** sendKeys() and Actions class (keyDown(), keyUp(), sendKeys() with Keys)
- **Mouse Actions:** Actions class (click(), moveToElement(), dragAndDrop())
- **Multi-select Dropdown:** Select class with isMultiple() and selectBy\*() methods

# Session 3

- **Introduction to delivery pipeline**
- **Introduction to Jenkins**
- **Jenkins management**
- **Adding slave node to Jenkins**
- **Building a delivery pipeline**
- **Selenium integration with Jenkins**

## Introduction to Jenkins

- Jenkins is an open source automation server which enables developers around the world to reliably build, test, and deploy their software.
- Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.
- It provides a streamlined, efficient, and consistent way to deliver software from development to production, ensuring higher quality, faster delivery, and easier collaboration among teams.

## **Introduction to Jenkins**

- It runs test and build like JAR, Trigger the job automatically
- Its primary focus is to keep track of the version control system and initiate and monitor a build system if there are any changes

**Example** - If your team is developing a project, Jenkins will continuously test your project builds and show you the errors in early stages of your development

## **Advantages of Jenkins**

### **Continuous Integration and Continuous Delivery**

As an extensible automation server, Jenkins can be used as a simple CI server or turned into the continuous delivery hub for any project.

### **Easy installation**

Jenkins is a self-contained Java-based program, ready to run out-of-the-box, with packages for Windows, Linux, macOS and other Unix-like operating systems.

# **Advantages of Jenkins**

## **Easy configuration**

Jenkins can be easily set up and configured via its web interface, which includes on-the-fly error checks and built-in help

## **Plugins**

With hundreds of plugins in the Update Center, Jenkins integrates with practically every tool in the continuous integration and continuous delivery toolchain.

## **Advantages of Jenkins**

### **Extensible**

Jenkins can be extended via its plugin architecture, providing nearly infinite possibilities for what Jenkins can do.

### **Distributed**

Jenkins can easily distribute work across multiple machines, helping drive builds, tests and deployments across multiple platforms faster.

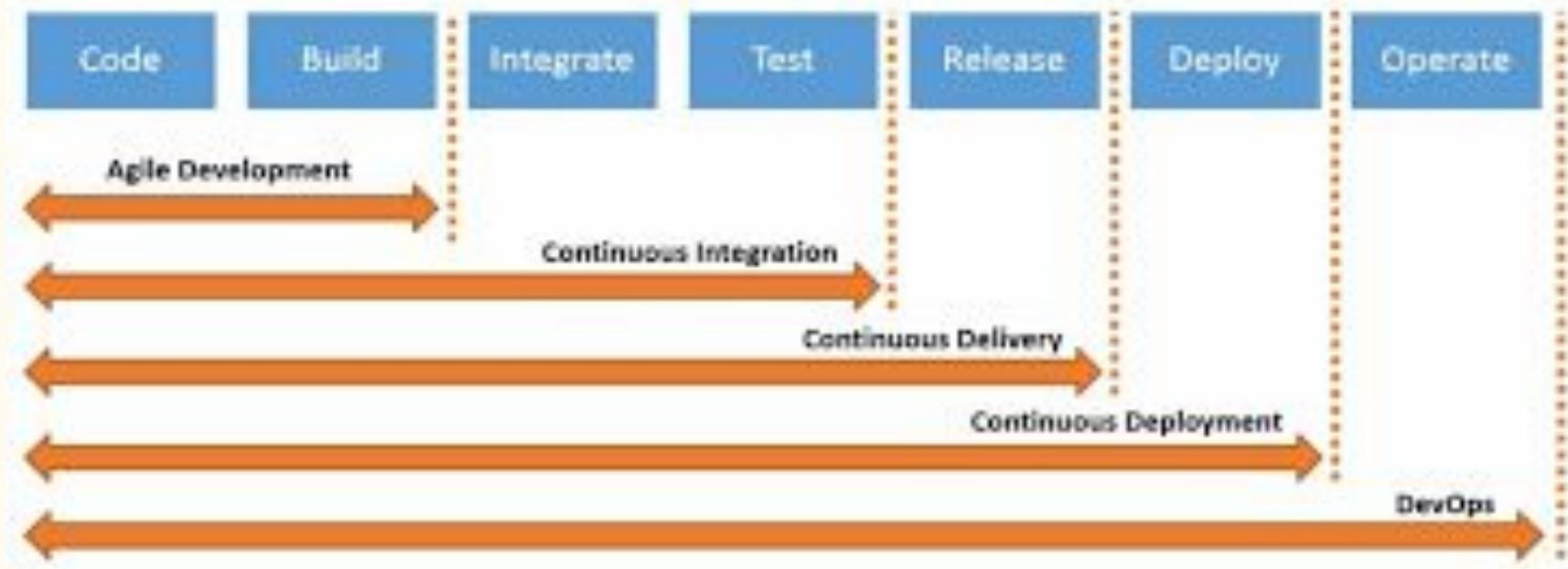
## Introduction to delivery pipeline

- A delivery pipeline is a key component of **Continuous Integration (CI)** and **Continuous Delivery (CD)**, which are practices within the **DevOps** culture aimed at automating software delivery and improving operational efficiency
- This process involves building the software in a reliable and repeatable manner, as well as progressing the built software (called a "build") through multiple stages of testing and deployment.
- **Jenkins Pipeline** (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins.
- **Tools** - Bamboo, TeamCity

## Features of Pipeline:

- **Code:** Pipelines are implemented in code and typically checked into source control, giving teams the ability to edit, review, and iterate upon their delivery pipeline.
- **Durable:** Pipelines can survive both planned and unplanned restarts of the Jenkins controller.
- **Pausable:** Pipelines can optionally stop and wait for human input or approval before continuing the Pipeline run.
- **Versatile:** Pipelines support complex real-world CD requirements, including the ability to fork/join, loop, and perform work in parallel.
- **Extensible:** The Pipeline plugin supports custom extensions to its DSL [1] and multiple options for integration with other plugins.

# Introduction to delivery pipeline



# **Continuous Integration (CI) and Continuous Deployment**

## **Continuous Integration**

- Continuous Integration is about how developers integrate code using a shared repository multiple times a day with the help of automation.

## **Continuous Delivery**

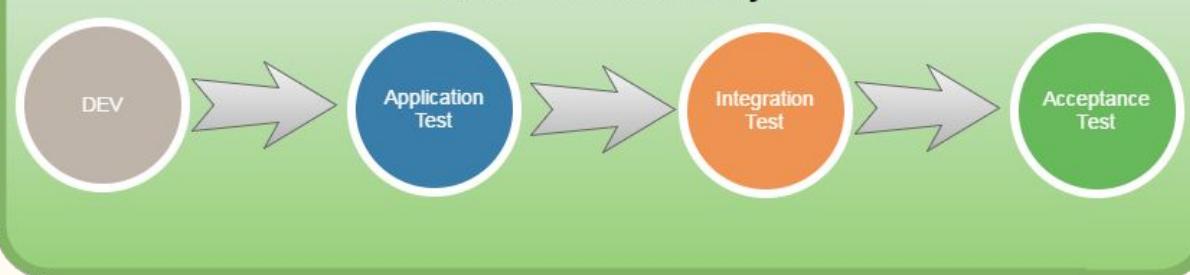
- It is about automatically releasing software to the test or produce environment

## Continuous Integration

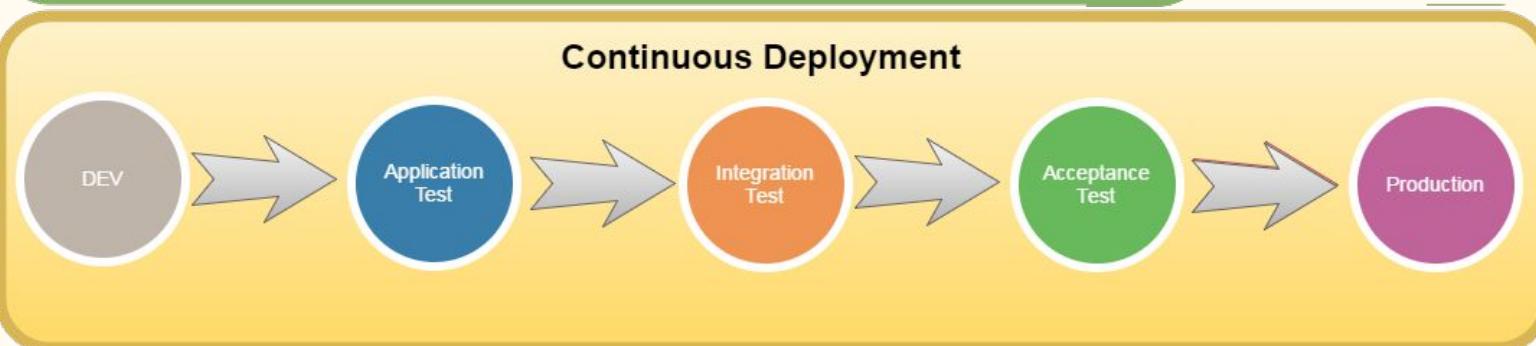


Automatic trigger

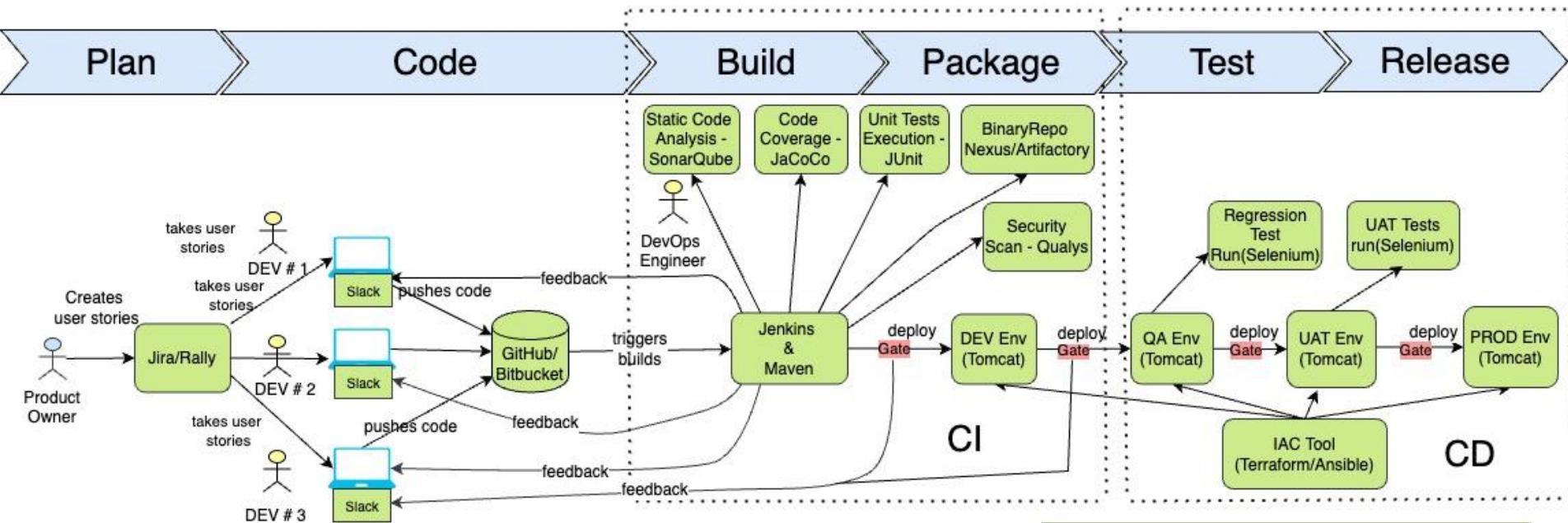
## Continuous Delivery



## Continuous Deployment



## Agile Development with DevOps - CICD Process Flow Diagram



### Continuous integration?

1. Continuously integrate and merge code changes frequently
2. Find out bugs early in software development stages
3. Make frequent releases possible
4. Make Continuous testing possible
5. Encourage good coding practices

### Sample Gating Criteria for DEV/QA/UAT/PROD deployment

1. 100% unit test executions pass
2. At least 80% code coverage
3. No critical defects/vulnerabilities in code

### Milestones:

1. Automated builds
2. Automated unit tests execution
3. Automated code quality checks
4. Automated code coverage checks
5. Automated security scanning
6. Automated deployments with gating
7. Automated feedback to prod teams
8. Automated Binary storage into Repo manager
9. Automated Infrastructure setup

## **Installation of Jenkins**

**Java** - <https://www.oracle.com/in/java/technologies/downloads/>

**Jenkins** - <https://www.jenkins.io/download/>

<https://www.jenkins.io/doc/pipeline/tour/hello-world/>

<https://www.javatpoint.com/installing-jenkins-on-windows>

1. Before downloading, please take a moment to review the [Hardware and Software requirements](#) section of the User Handbook.
2. Select one of the packages below and follow the download instructions.
3. Once a Jenkins package has been downloaded, proceed to the [Installing Jenkins](#) section of the User Handbook.
4. You may also want to verify the package you downloaded. [Learn more about verifying Jenkins downloads.](#)

Download Jenkins 2.479.2 LTS for:



	<b>Generic Java package (.war)</b>	
	SHA-256: 177c2c033f0d3ae4148e601d0fdada60112d83f250521f3a0a0fd97ccb138dbd	
	Docker	
	Kubernetes	
	Ubuntu/Debian	
	Red Hat Enterprise Linux and derivatives	
	Fedora	
	Windows	

Download Jenkins 2.491 for:

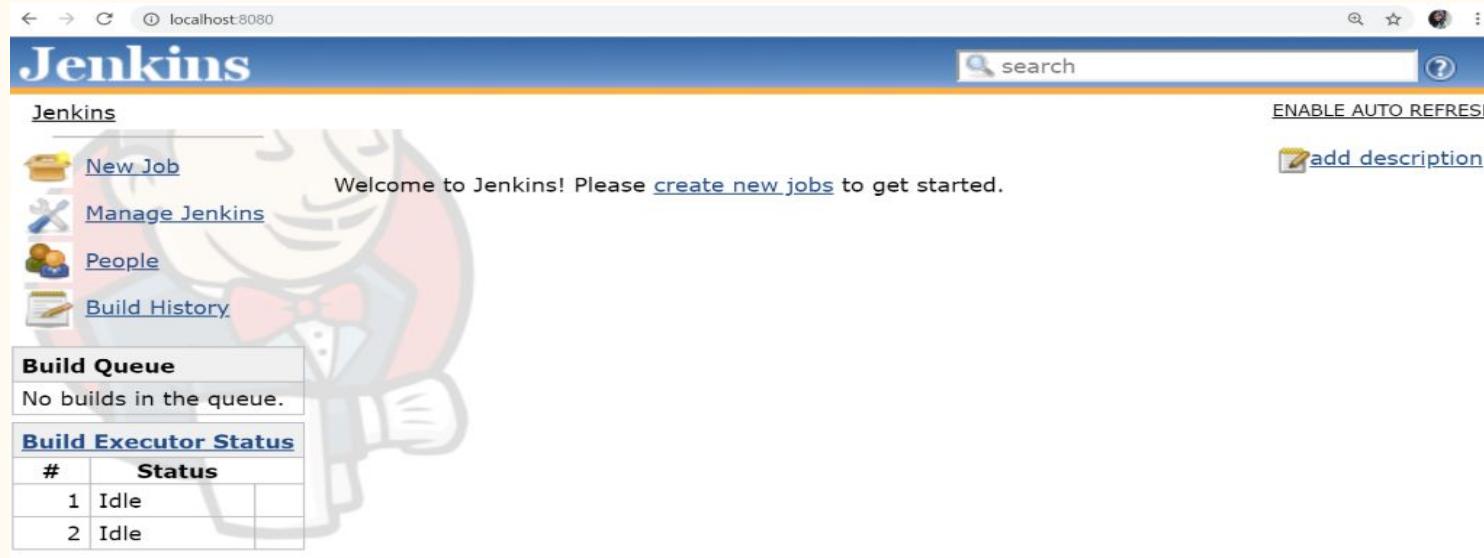
	<b>Generic Java package (.war)</b>	
	SHA-256: ccef73536436ced77776c994fc86897d6c3899efe8904def444036663111c9b	
	Docker	
	Ubuntu/Debian	
	Red Hat Enterprise Linux and derivatives	
	Fedora	
	Windows	
	openSUSE	

## Starting Jenkins

Open the command prompt and go to the directory where the Jenkins.war file is located. And then run the following command:

```
C:/Java -jar Jenkins.war
```

When you run this command, various tasks will run, one of which is the extraction of the war file which is done by an embedded webserver called winstone.



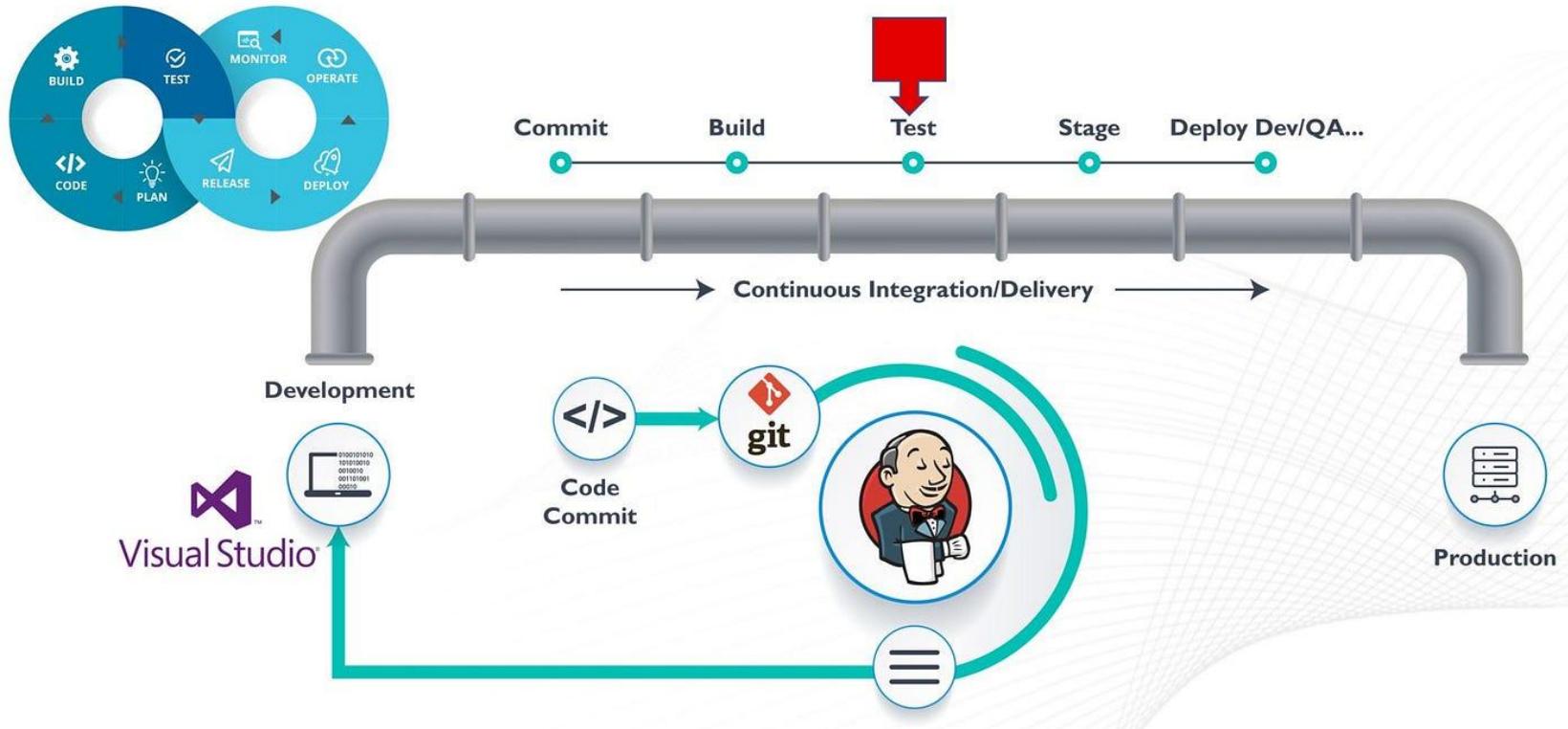
Now you can access the Jenkins. Open your browser and type the following url on your browser:

```
http://localhost:8080
```

## How does Jenkins work?

Jenkins Continuous Integration Pipeline is a powerful instrument that consists of a set of tools designed to **host**, **monitor**, **compile** and **test** code, or code changes, like:

- **Continuous Integration Server** (Jenkins, Bamboo, CruiseControl, TeamCity, and others)
- **Source Control Tool** (e.g., CVS, SVN, GIT, Mercurial, Perforce, ClearCase and others)
- **Build tool** (Make, ANT, Maven, Ivy, Gradle, and others)
- **Automation testing framework** (Selenium, Appium, TestComplete, UFT, and others)



## Jenkins Plugins

- By default, Jenkins comes with a limited set of features.
- If you want to integrate your Jenkins installation with version control tools like Git, then you need to install plugins related to Git.
- In fact, for integration with tools like Maven, Amazon EC2, you need to install respective plugins in your Jenkins.

# Jenkins Management

Jenkins

search [?](#)

Jenkins ➔ [DISABLE AUTO REFRESH](#)

[New Item](#) [add description](#)

[People](#)

[Build History](#)

[Manage Jenkins](#)

[Credentials](#)

[New View](#)

[Build Queue](#) -

No builds in the queue.

[Build Executor Status](#) -

1 Idle  
2 Idle

All [+](#)

S	W	Name ↓	Last Success	Last Failure	Last Duration
		<a href="#">JavaTpoint Maven</a>	N/A	N/A	N/A

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

To manage Jenkins, click on the "Manage Jenkins" option from the left hand of the Jenkins Dashboard page.

 New Item People Build History Manage Jenkins Credentials New View**Build Queue**

No builds in the queue.

**Build Executor Status**

1 Idle

2 Idle

## Manage Jenkins

**Configure System**

Configure global settings and paths.

**Configure Global Security**

Secure Jenkins; define who is allowed to access/use the system.

**Configure Credentials**

Configure the credential providers and types

**Global Tool Configuration**

Configure tools, their locations and automatic installers.

**Reload Configuration from Disk**

Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

**Manage Plugins**

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

 There are updates available**System Information**

Displays various environmental information to assist trouble-shooting.

When you click on the Manage Jenkins, you will get the various options to manage the Jenkins

## Configure System

In this, we can manage paths to the various tools to use in builds, such as the versions of Ant and Maven, as well as security options, email servers, and other system-wide configuration details. Jenkins will add the required configuration fields dynamically when new plugins are installed.

The screenshot shows the Jenkins configuration interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'Credentials', and 'New View'. Below that are two collapsed sections: 'Build Queue' (no builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is titled 'Maven Project Configuration'. It includes fields for 'Global MAVEN\_OPTS' (a dropdown menu), 'Local Maven Repository' (set to 'Default (~/.m2/repository)'), '# of executors' (set to 2), 'Labels' (an empty input field), 'Usage' (set to 'Use this node as much as possible'), and 'Quiet period' (an empty input field). At the bottom are 'Save' and 'Apply' buttons.

Jenkins configuration screen showing the Maven Project Configuration section.

**Maven Project Configuration**

Global MAVEN\_OPTS

Local Maven Repository

# of executors

Labels

Usage

Quiet period

Save      Apply

## Configure Global Security

Configure Global Security option provides the ability to set up users and their relevant permissions on the Jenkins instance. By default, you will not want everyone to be able to define builds or other administrative tasks in Jenkins. So Jenkins provides the ability to have a security configuration in place.

The screenshot shows the Jenkins global security configuration interface. At the top, there's a navigation bar with the Jenkins logo, a search bar, and a help icon. Below the header, the page title is "Configure Global Security" with a lock icon. There are several configuration sections:

- Markup Formatter:** A dropdown menu set to "Plain text".
  - Help icon (question mark)
- Agents:** Settings for TCP port for inbound agents.
  - "TCP port for inbound agents": "Fixed" (radio button selected), "Random", "Disable".
  - "Agent protocols..." button.
  - Help icon (question mark)
- CSRF Protection:**
  - "Prevent Cross Site Request Forgery exploits" checkbox.
  - Help icon (question mark)
- Plugin Manager:**
  - "Use browser for metadata download" checkbox.
  - Help icon (question mark)
- Hidden security warnings:** A section where users can review and manage hidden security issues.

At the bottom, there are two buttons: "Save" (dark blue) and "Apply" (light gray).

## **Reload Configuration from Disk**

- Jenkins stores all its system and job configuration details as XML files and all XML files are stored in the Jenkins home directory.
- Jenkins also stores all of the build histories.
- If you are moving build jobs from one Jenkins instance to another or archiving old build jobs, you will have to insert or remove the corresponding build job directories to Jenkins's builds directory.
- You do not have to take Jenkins offline to do this?you can simply use the "Reload Configuration from Disk" option to reload the Jenkins system and build job configurations directly.

## Manage Plugin

Here you can install a wide or different variety of third-party plugins from different Source code management tools such as Git, ClearCase or Mercurial, to code quality and code coverage metrics reporting. We can download, install, update, or remove the plugins from the Manage Plugins screen.

The screenshot shows the Jenkins Manage Plugins interface. At the top, there's a navigation bar with the Jenkins logo, a search bar, and a help icon. Below the navigation bar, the page title is "Plugin Manager". On the left, there are links to "Back to Dashboard" and "Manage Jenkins". The main content area has a header with tabs: "Updates" (which is selected), "Available", "Installed", and "Advanced". Below the tabs is a table with columns: "Install", "Name", "Version", and "Installed". The table lists several plugins:

Install	Name	Version	Installed
<a href="#">Branch API</a>	This plugin provides an API for multiple branch based projects.	2.5.3	2.3.0
<a href="#">Credentials</a>	This plugin allows you to store credentials in Jenkins.	2.2.0	2.1.19
<a href="#">Credentials Binding</a>	Allows credentials to be bound to environment variables for use from miscellaneous build steps.	1.19	1.18
<a href="#">Docker Commons</a>	Provides the common shared functionality for various Docker-related plugins.	1.15	1.14
<a href="#">Folders</a>	This plugin allows users to create "folders" to organize jobs. Users can define custom taxonomies (like by project type, organization type etc). Folders are nestable and you can define views within folders. Maintained by CloudBees, Inc.	6.9	6.5.1
<a href="#">JUnit</a>	Allows JUnit-format test results to be published.	1.28	1.27
<a href="#">Pipeline</a>			

At the bottom of the table, there are two buttons: "Download now and install after restart" and "Check now". A status message says "Update information obtained: 21 hr ago".

## System Information

This option displays a list of all the current Java system properties and system environment variables. Here you can check what version of Java is currently running in, what user it is running under, and so forth.

[ENABLE AUTO REFRESH](#)

## System Properties

Name ↓	Value
awt.toolkit	sun.awt.windows.WToolkit
catalina.base	C:\apache-tomcat-9.0.21
catalina.home	C:\apache-tomcat-9.0.21
catalina.useNaming	true
common.loader	"\${catalina.base}/lib","\${catalina.base}/lib/*.jar","\${catalina.home}/lib","\${catalina.home}/lib/*.jar"
file.encoding	Cp1252
file.encoding.pkg	sun.io
file.separator	\
ignore.endorsed.dirs	
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment
java.awt.printerjob	sun.awt.windows.WPrinterJob
java.class.path	c:\apache-tomcat-9.0.21\bin\bootstrap.jar;c:\apache-tomcat-9.0.21\bin\tomcat-juli.jar

## System Log

The System Log option is used to view the Jenkins log files in real time. As well as, the main use of this option is for troubleshooting.

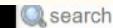
The screenshot shows the Jenkins interface with the following details:

- Header:** The Jenkins logo, a search bar, and a "ENABLE AUTO REFRESH" link.
- Breadcrumbs:** Jenkins > log
- Left Sidebar:** A vertical menu with icons and links:
  - Back to Dashboard
  - Manage Jenkins
  - Logger List
  - All Logs
  - New Log Recorder
  - Log Levels
- Page Content:** The "Log Recorders" page, featuring a title "Log Recorders" with a help icon, a table header with columns "S" and "Name ↓", and a single entry "All Jenkins Logs" with a clipboard icon. Below the table is a blue button labeled "Add new log recorder".

To see the logs click on the 'All Jenkins Logs'.



Jenkins > log >



ENABLE AUTO REFRESH

Back to Dashboard

Manage Jenkins

Logger List

All Logs

New Log Recorder

Log Levels

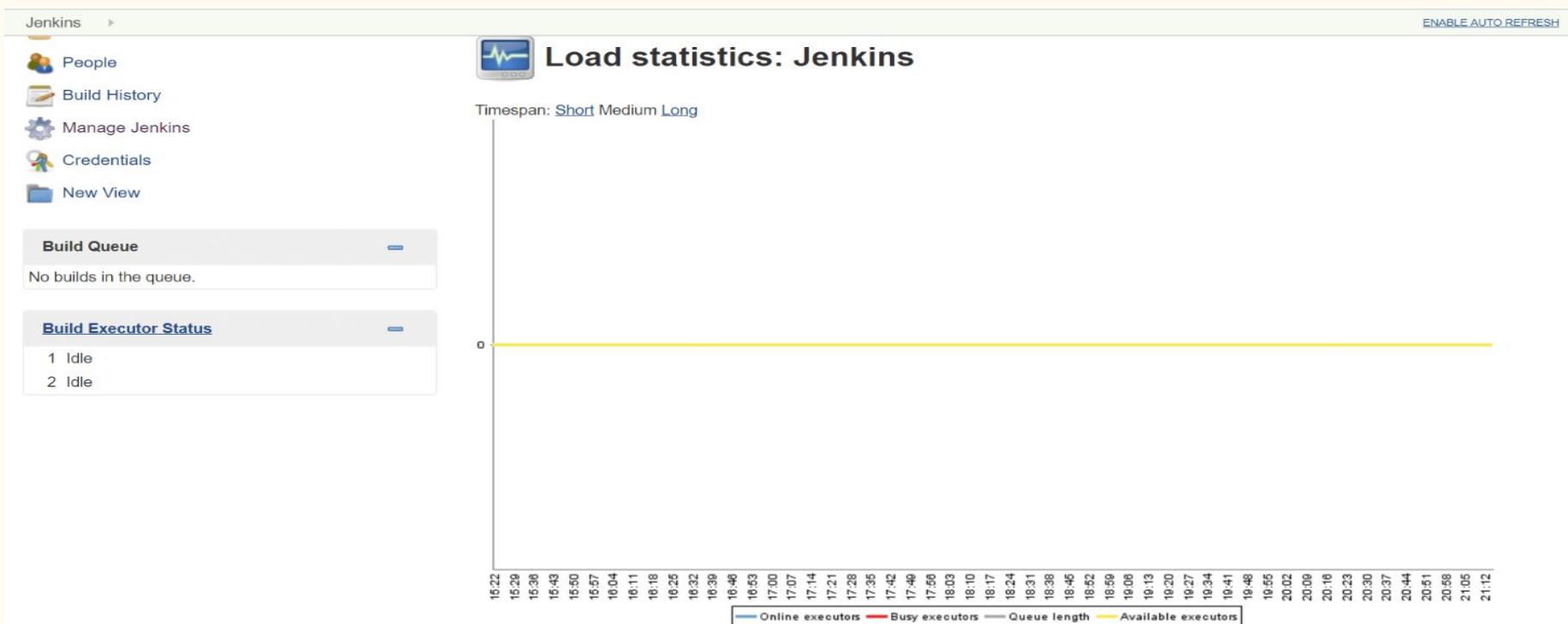


## Jenkins Log

```
Jul 13, 2019 3:20:29 PM INFO jenkins.InitReactorRunner$1 onAttained
Started initialization
Jul 13, 2019 3:20:32 PM INFO org.kohsuke.stapler.localeDrivenResourceProvider getLocaleDrivenResourceProviders
Registered LocaleDrivenResourceProvider: jenkins.MetaLocaleDrivenResourceProvider@21b8fdd7
Jul 13, 2019 3:20:40 PM INFO hudson.ClassicPluginStrategy createPluginWrapper
Plugin github-branch-source.jpi is disabled
Jul 13, 2019 3:21:37 PM INFO jenkins.InitReactorRunner$1 onAttained
Listed all plugins
Jul 13, 2019 3:21:53 PM INFO jenkins.InitReactorRunner$1 onAttained
Prepared all plugins
Jul 13, 2019 3:21:53 PM INFO jenkins.InitReactorRunner$1 onAttained
Started all plugins
Jul 13, 2019 3:21:57 PM INFO jenkins.InitReactorRunner$1 onAttained
Augmented all extensions
Jul 13, 2019 3:21:57 PM INFO jenkins.InitReactorRunner$1 onAttained
Loaded all jobs
Jul 13, 2019 3:21:57 PM INFO hudson.model.AsyncPeriodicWork$1 run
Started Download metadata
Jul 13, 2019 3:21:57 PM INFO hudson.model.AsyncPeriodicWork$1 run
Finished Download metadata. 22 ms
Jul 13, 2019 3:21:57 PM INFO jenkins.InitReactorRunner$1 onAttained
```

## Load Statistics

This option is used to see the graphical data on how busy the Jenkins instance is in terms of the number of concurrent builds and the length of the build queue which provides an idea of how long your builds need to wait before being executed. These statistics can show you a good idea of whether extra capacity or extra build nodes are required from an infrastructure perspective.



## Jenkins CLI

Using this option, you can access various features in Jenkins through a command-line. To run the Jenkins through cli, first you have to download the Jenkins-cli.jar file and run it on your command prompt as follows:

```
java -jar jenkins-cli.jar -s http://localhost:8080/jenkins/
```

This page gives the list of available commands with their description.

The screenshot shows the Jenkins CLI interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'Credentials', and 'New View'. Below these are two collapsed sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area has a title 'Jenkins CLI' with a terminal icon. It contains a message about using the CLI, a command line input field with the command 'java -jar jenkins-cli.jar -s http://localhost:8080/jenkins/ help', and a table of available commands with their descriptions.

Available Commands	
<a href="#"><u>add-job-to-view</u></a>	Adds jobs to view.
<a href="#"><u>build</u></a>	Builds a job, and optionally waits until its completion.
<a href="#"><u>cancel-quiet-down</u></a>	Cancel the effect of the "quiet-down" command.
<a href="#"><u>clear-queue</u></a>	Clears the build queue.
<a href="#"><u>connect-node</u></a>	Reconnect to a node(s)
<a href="#"><u>console</u></a>	Retrieves console output of a build.
<a href="#"><u>copy-job</u></a>	Copies a job.
<a href="#"><u>create-credentials-by-xml</u></a>	Create Credential by XML
<a href="#"><u>create-credentials-domain-by-xml</u></a>	Create Credentials Domain by XML
<a href="#"><u>create-job</u></a>	Creates a new job by reading stdin as a configuration XML file.
<a href="#"><u>create-node</u></a>	Creates a new node by reading stdin as a XML configuration.
<a href="#"><u>create-view</u></a>	Creates a new view by reading stdin as a XML configuration.

## Script Console

This option allows you to run Groovy scripts on the server. This option is very useful for advanced troubleshooting since it requires a strong knowledge of the internal Jenkins architecture.

The screenshot shows the Jenkins interface with the 'Script Console' page selected. The left sidebar contains links for Jenkins, People, Build History, Manage Jenkins, Credentials, and New View. Below these are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is titled 'Script Console' and contains instructions for running Groovy scripts. It includes an example code snippet: `println(Jenkins.instance.pluginManager.plugins)`. A note states that all plugin classes are pre-imported. At the bottom right of the main area is a 'Run' button.

Jenkins

People

Build History

Manage Jenkins

Credentials

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

## Script Console

Type in an arbitrary [Groovy script](#) and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use `System.out`, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. `jenkins.*`, `jenkins.model.*`, `hudson.*`, and `hudson.model.*` are pre-imported.

1

Run

## Manage nodes

Jenkins can handle parallel and distributed builds. In this page, you can configure how many builds you want. Jenkins runs concurrently, and, if you are using distributed builds, set up build nodes. A build node (slave) is another machine that Jenkins can use to execute its builds.

The screenshot shows the Jenkins interface for managing nodes. On the left, there's a sidebar with links: Back to Dashboard, Manage Jenkins, New Node, and Configure. Below that are sections for Build Queue (empty) and Build Executor Status (1 Idle, 2 Idle). The main content area is titled "Nodes" and contains a table with one row for the "master" node. The table columns are S, Name (sorted), Response Time, Free Disk Space, Free Temp Space, Clock Difference, Free Swap Space, and Architecture. The master node has 0ms response time, 178.86 GB free disk space, 178.86 GB free temp space, is in sync, has 2.93 GB free swap space, and is running Windows 10 (amd64). A "Refresh status" button is at the bottom right of the table. The top right of the screen has a search bar and a help icon.

S	Name ↓	Response Time	Free Disk Space	Free Temp Space	Clock Difference	Free Swap Space	Architecture
	master	0ms	178.86 GB	178.86 GB	In sync	2.93 GB	Windows 10 (amd64)
	Data obtained	3 min 34 sec	3 min 34 sec	3 min 33 sec	3 min 33 sec	3 min 33 sec	3 min 33 sec

## Create and Manage Users in Jenkins

**Step 1:** Open your Jenkins dashboard by visiting <http://localhost:8080/jenkins>

**Step 2:** Click on 'Manage Jenkins' and select the 'Available' tab.

The screenshot shows the Jenkins Plugin Manager interface. At the top, there is a navigation bar with the Jenkins logo, a search bar, and a help icon. Below the navigation bar, the 'Plugin Manager' page is displayed. On the left, there are links to 'Back to Dashboard' and 'Manage Jenkins'. The main area has tabs for 'Updates', 'Available' (which is highlighted with a green box), 'Installed', and 'Advanced'. A 'Filter:' input field is located at the top right of the plugin list. The plugin list table has columns for 'Install', 'Name', and 'Version'. The table contains several entries under the '.NET Development' category, each with a brief description and a checkbox for installation. The plugins listed are: CCM (version 3.2), change-assembly-version-plugin (version 1.10), FxCop Runner (version 1.1), MSBuild (version 1.29), and MSTest (version 1.0.0).

Install	Name	Version
<u>.NET Development</u>		
<input type="checkbox"/>	<a href="#">CCM</a>	3.2
<input type="checkbox"/>	This plug-in collects the <a href="#">CCM</a> analysis results of the project modules and visualizes the found warnings.	
<input type="checkbox"/>	<a href="#">change-assembly-version-plugin</a>	1.10
<input type="checkbox"/>	<a href="#">FxCop Runner</a>	1.1
<input type="checkbox"/>	FxCopCmd.exe support plugin.	
<u>MSBuild</u>		
<input type="checkbox"/>	This plugin makes it possible to build a Visual Studio project (.proj) and solution files (.sln).	1.29
<u>MSTest</u>		
<input type="checkbox"/>	This plugin converts <a href="#">MSTest</a> TRX test reports into JUnit XML reports so it can be integrated with Hudson's JUnit features.	1.0.0

**Step 3:** On the filter option, type "role-based" and press Enter.

The screenshot shows the Jenkins Plugin Manager interface. At the top, there's a navigation bar with the Jenkins logo, a search bar containing the text "search", and a help icon. Below the navigation bar, the page title is "Plugin Manager". Underneath the title, there are two links: "Back to Dashboard" and "Manage Jenkins".

The main content area has a search bar at the top right with the placeholder "Filter:" and a search icon. The search term "role-based" is entered into the field. Below the search bar, there are four tabs: "Updates", "Available" (which is selected and highlighted in red), "Installed", and "Advanced".

The main table displays a single plugin entry:

Install ↓	Name	Version
<a href="#">Role-based Authorization Strategy</a>		2.13

The plugin details are shown in a box:  
Description: Enables user authorization using a Role-Based strategy. Roles can be defined globally or for particular jobs or nodes selected by regular expressions.

At the bottom of the table, there are two buttons: "Install without restart" (highlighted with a green border) and "Download now and install after restart". To the right, a message says "Update information obtained: 1 hr".

**Step 4:** Now, select the plugin and click on 'Install without restart' button.

Jenkins > Update Center

ENABLE AUTO REFRESH

 Manage Jenkins

 Manage Plugins

## Installing Plugins/Upgrades

### Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

WMI Windows Agents



Success

Ant



Success

Matrix Authorization Strategy



Success

jQuery



Success

OWASP Markup Formatter



Success

PAM Authentication



Success

LDAP



Success

External Monitor Job Type



Success

jQuery UI



Success

Role-based Authorization Strategy



Success

 [Go back to the top page](#)

(you can start using the installed plugins right away)

 Restart Jenkins when installation is complete and no jobs are running

**Step 5:** Click on 'Go back to the top page'.

## Enable Role-Based Strategy on Jenkins

**Step 1:** After Plugin installation, go to the 'Manage Jenkins' and then click on 'Configure Global Security'.

The screenshot shows the Jenkins Manage Jenkins interface. On the left is a sidebar with links: New Item, People, Build History, Manage Jenkins (which is selected and highlighted in purple), Credentials, and New View. Below these are two collapsed sections: Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle). The main content area is titled "Manage Jenkins". It contains several configuration options: "Configure System" (gear icon), "Configure Global Security" (padlock icon, highlighted with a green box), "Configure Credentials" (key and gear icon), "Global Tool Configuration" (wrench and wrench icon), and "Reload Configuration from Disk" (refresh icon). At the bottom is a "Manage Plugins" section (green icon).

Jenkins

New Item

People

Build History

Manage Jenkins

Credentials

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Manage Jenkins

Configure System

Configure global settings and paths.

Configure Global Security

Secure Jenkins; define who is allowed to access/use the system.

Configure Credentials

Configure the credential providers and types

Global Tool Configuration

Configure tools, their locations and automatic installers.

Reload Configuration from Disk

Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

Manage Plugins

localhost:8080/jenkins/configureSecurity

When you click the Configure Global Security option then you will see the following page:

The screenshot shows the Jenkins global security configuration page. At the top left is the Jenkins logo. The title 'Configure Global Security' is centered above a yellow padlock icon. A green rectangular box highlights the 'Enable security' checkbox. Below it is a 'Markup Formatter' section with a 'Plain text' dropdown set to 'Plain text'. A note below says 'Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.' Under 'Agents', there are options for 'TCP port for inbound agents': 'Fixed' (radio button selected), 'Random' (radio button), and 'Disable' (radio button). A 'Agent protocols...' button is present. The 'CSRF Protection' section contains a 'Prevent Cross Site Request Forgery exploits' checkbox. The 'Plugin Manager' section has a note: 'This feature can be disabled via the global configuration screen.' At the bottom are 'Save' and 'Apply' buttons.

Enable security

Plain text

Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.

Agent protocols...

Prevent Cross Site Request Forgery exploits

Save Apply

Check on Enable security option.

**Step 2:** On the Security Realm section, select 'jenkins' own user database'.

On Authorization section, select 'Role-Based Strategy'.

The screenshot shows the Jenkins 'Configure Global Security' configuration page. At the top left, there is a yellow padlock icon. The main title is 'Configure Global Security'. Below the title, there is a section titled 'Access Control' with a checked checkbox for 'Enable security' and an unchecked checkbox for 'Disable remember me'. To the right of this is a 'Security Realm' section containing three radio button options: 'Delegate to servlet container' (unchecked), 'Jenkins' own user database' (checked and highlighted with a green box), and 'LDAP' (unchecked). Below the 'Security Realm' section is an 'Authorization' section with five radio button options: 'Anyone can do anything' (unchecked), 'Legacy mode' (unchecked), 'Logged-in users can do anything' (unchecked), 'Matrix-based security' (unchecked), and 'Project-based Matrix Authorization Strategy' (unchecked). The 'Role-Based Strategy' option is highlighted with a green box. At the bottom of the page are two buttons: a large blue 'Save' button on the left and a smaller grey 'Apply' button on the right. Both buttons are also highlighted with a green box.

Jenkins ➔ Configure Global Security

## Configure Global Security

Enable security

Disable remember me

Access Control

Security Realm

- Delegate to servlet container
- Jenkins' own user database
- LDAP

Authorization

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

Role-Based Strategy

**Save** **Apply**

Click on **Save**.

**Step 3:** You will be prompted to add your first user. Here, we are setting up an admin user for the system.

Jenkins

2 search log in

Jenkins > Jenkins' own user database

Back to Dashboard Manage Jenkins Create User

## Create First Admin User

Username:	admin
Password:	.....
Confirm password:	.....
Full name:	Administrator
E-mail address:	jnlp://localhost:8080/jenkins

Create First Admin User

Click on Create First Admin User.

The screenshot shows the Jenkins dashboard. At the top, there is a navigation bar with the Jenkins logo, the word "Jenkins" in large letters, a red notification badge with the number "2", a search icon, and a user menu for "Administrator". Below the navigation bar, the URL "Jenkins > Jenkins' own user database" is visible. On the left side, there is a sidebar with four items: "New Item" (with a plus sign icon), "People" (with a group icon), "Build History" (with a clipboard icon), and "Manage Jenkins" (with a gear icon). The main content area displays a large "Success" message in bold, centered text. Below it, a smaller message says "You are now logged in. Go back to [the top page](#)".

Jenkins

2

search

Administrator | log out

Jenkins > Jenkins' own user database

New Item

People

Build History

Manage Jenkins

Success

You are now logged in. Go back to [the top page](#).

## Creating User on Jenkins

**Step 1:** Now it's time to setup your users in the system. When you go to manage Jenkins and scroll down, you will see the 'Manage Users' option. Click on this option.

Jenkins ➔ ENABLE AUTO REFRESH

 Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

 **Manage and Assign Roles**  
Handle permissions by creating roles and assigning them to users/groups

 **About Jenkins**  
See the version and license information.

 **Manage Old Data**  
Scrub configuration files to remove remnants from old plugins and earlier versions.

 **Manage Users**  
Create/delete/modify users that can log in to this Jenkins

 **Prepare for Shutdown**  
Stops executing new builds, so that the system can be eventually shut down safely.

[Manage Users](#)

**Step 2:** Just Like, you defined the admin user, and start creating other users for the system. Let's create another user:

- To create another user click on 'Create User' option on the left hand side of the Manage Users page.

Jenkins

search

Administrator | log out

Jenkins > Jenkins' own user database

ENABLE AUTO REFRESH

Back to Dashboard

Manage Jenkins

Create User

## Users

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 some projects and have no direct Jenkins access.

User ID	Name	
 admin	Administrator	

Click on Create User button.

The screenshot shows the Jenkins web interface. At the top, there is a navigation bar with the Jenkins logo, a search bar, and links for 'Administrator | log out'. Below the navigation bar, the URL 'Jenkins › Jenkins' own user database' is visible. On the left side, there is a sidebar with three items: 'Back to Dashboard', 'Manage Jenkins', and 'Create User' (which is highlighted with a blue background). The main content area is titled 'Create User'. It contains five input fields: 'Username' (Nikita), 'Password' (\*\*\*\*\*), 'Confirm password' (\*\*\*\*\*), 'Full name' (Nikita), and 'E-mail address' (nikita@gmail.com). At the bottom of the form is a blue 'Create User' button.

Jenkins

search

Administrator | log out

Jenkins › Jenkins' own user database

Back to Dashboard

Manage Jenkins

Create User

Create User

Username: Nikita

Password: \*\*\*\*\*

Confirm password: \*\*\*\*\*

Full name: Nikita

E-mail address: nikita@gmail.com

## Managing User Roles on Jenkins

### Step 1:

- Click on Manage Jenkins.
- Select Manage and Assign Roles. Note that, Manage and Assign Role will only be visible if you have installed the Role strategy plugin.

Jenkins ➔ ENABLE AUTO REFRESH

 **Manage Nodes**  
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

 **Manage and Assign Roles**  
Handle permissions by creating roles and assigning them to users/groups Manage and Assign Role

 **About Jenkins**  
See the version and license information.

 **Manage Old Data**  
Scrub configuration files to remove remnants from old plugins and earlier versions.

 **Manage Users**  
Create/delete/modify users that can log in to this Jenkins

 **Prepare for Shutdown**  
Stops executing new builds, so that the system can be eventually shut down safely.

**Step 2:** Click on Manage Roles option to add new roles.

 Jenkins

search  Administrator | log out

Jenkins > Manage and Assign Roles

New Item

People

Build History

Manage Jenkins

My Views

Credentials

New View

 Manage and Assign Roles

 [Manage Roles](#)  
Manage Roles

 [Assign Roles](#)  
Assign Roles

 [Role Strategy Macros](#)  
Provides info about macro usage and available macros

**Build Queue**   
No builds in the queue.

**Build Executor Status**   
1 Idle  
2 Idle

**Step 2:** To create a new role called "developer".

- Type "developer" in the Role to add option.
- Click on Add to create a new role.
- Now, select the appropriate permissions that you want to assign to the developer role.
- Then Click on Save button.

The screenshot shows the Jenkins 'Manage Roles' interface. At the top, there's a navigation bar with a search icon, the word 'Administrator', and a 'log out' link. Below it, the path 'Jenkins > Manage and Assign Roles' is visible. The main title 'Manage Roles' is centered above the 'Global roles' section. This section contains a table with three columns: 'Role', 'Overall', and 'Agent'. Under 'Overall', there are 16 permission checkboxes. The 'admin' row has all checkboxes checked. The 'developer' row has checkboxes for 'Read', 'Create', 'Delete', 'View', 'Build', 'Configure', and 'Connect' checked. A green box highlights the 'Role to add' input field at the bottom left, which contains the text 'developer'. An 'Add' button is located below the input field. The 'Project roles' section is partially visible at the bottom.

Role	Overall	Credentials	Agent
admin	<input checked="" type="checkbox"/> Administer <input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Create <input checked="" type="checkbox"/> Delete <input checked="" type="checkbox"/> ManageDomains <input checked="" type="checkbox"/> Update <input checked="" type="checkbox"/> View <input checked="" type="checkbox"/> Build <input checked="" type="checkbox"/> Configure <input checked="" type="checkbox"/> Connect <input checked="" type="checkbox"/> Create <input checked="" type="checkbox"/> Delete <input checked="" type="checkbox"/> Disconnect <input checked="" type="checkbox"/> Provision <input checked="" type="checkbox"/> Build <input checked="" type="checkbox"/> Cancel <input checked="" type="checkbox"/> Configure <input checked="" type="checkbox"/> Create		
developer	<input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Create <input checked="" type="checkbox"/> Delete <input type="checkbox"/> <input checked="" type="checkbox"/> View <input checked="" type="checkbox"/> Build <input checked="" type="checkbox"/> Configure <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>		

Role to add

developer

Add

Role Pattern	Credentials	Job	Run	SCM
Create Delete ManageDomains Update View Build Cancel Configure Create Delete Disconnect Move Read Write Delete Review Update Test				

## Assigning User Roles on Jenkins

**Step 1:** Now, you have created roles, let's assign them to specific users.

Click on Manage Jenkins.

Then select Manage and Assign Roles.

Click on Assign Roles.

The screenshot shows the Jenkins Manage and Assign Roles interface. At the top, there is a navigation bar with a Jenkins logo, a search bar, and a user account labeled "Administrator | log out". Below the navigation bar, the page title is "Manage and Assign Roles". On the left side, there is a sidebar with various Jenkins management links: "New Item", "People", "Build History", "Manage Jenkins", "My Views", "Credentials", and "New View". Below this, there are two sections: "Build Queue" (which says "No builds in the queue.") and "Build Executor Status" (which shows "1 Idle" and "2 Idle"). The main content area is titled "Manage and Assign Roles". It contains three main items: "Manage Roles" (with a lock icon), "Assign Roles" (with a people icon, which is highlighted with a green box), and "Role Strategy Macros" (with a puzzle piece icon). The "Assign Roles" link leads to the current page shown in the screenshot.

Jenkins

search

Administrator | log out

Jenkins > Manage and Assign Roles

New Item

People

Build History

Manage Jenkins

My Views

Credentials

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

## Manage and Assign Roles

[Manage Roles](#)  
Manage Roles

[Assign Roles](#)  
Assign Roles

[Role Strategy Macros](#)  
Provides info about macro usage and available macros

**Step 2:** Let's add new role "developer" to user.

- Add the User name on User/group to add option.
- Click on Add button.

**Jenkins** search Administrator | log out

Jenkins > Manage and Assign Roles

New Item People Build History Manage Jenkins My Views Credentials New View

Build Queue No builds in the queue.

Build Executor Status 1 Idle 2 Idle

## Assign Roles

Global roles

User/group	admin	developer
Anonymous	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Nikita	<input type="checkbox"/>	<input type="checkbox"/>

User/group to add  Add

Item roles

User/group
Anonymous <input checked="" type="checkbox"/>

Select "developer" role check box.



# Jenkins

search



Administrator

| log out

Jenkins > Manage and Assign Roles

New Item

People

Build History

Manage Jenkins

My Views

Credentials



## Assign Roles

### Global roles

User/group	admin	developer
Anonymous	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Nikita	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

- You can assign any role to any user as per your requirement.
- Then click on **Save**.

# Adding slave node to Jenkins

- Jenkins master can allocate slave nodes, schedule jobs, and distribute builds to multiple slaves for execution.
- Additionally, it can monitor the slave node's state, get build results from slave nodes and aid in the parallel execution on multiple nodes.

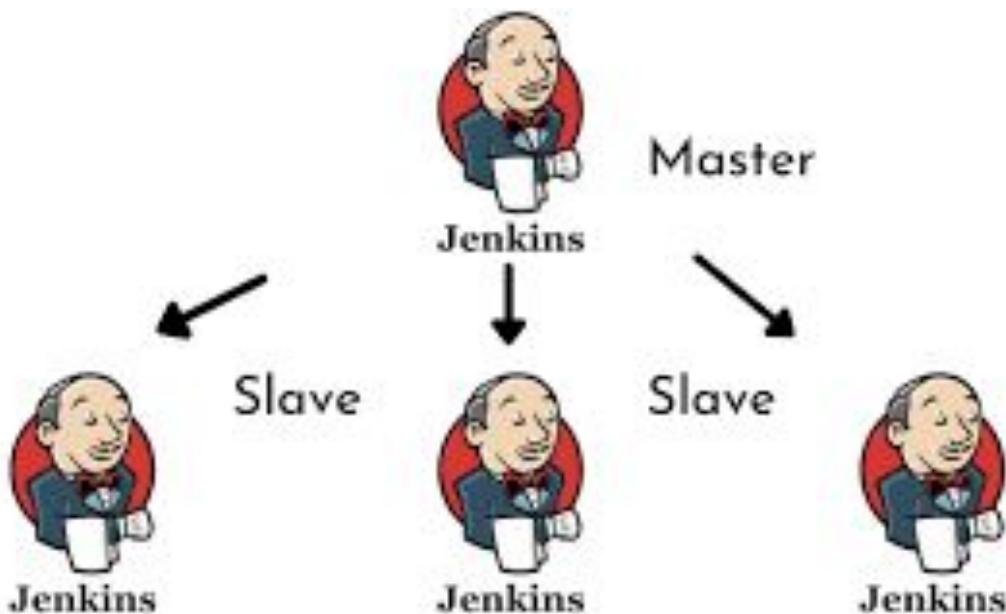
## Steps to be performed on slave node(agent) system:

- Open the URL localhost:8080 in browser and log in to Jenkins.
- In the Dashboard, click on System under System Configuration. Under the Jenkins URL tab, replace localhost by the IP address of the master system. Save the changes.

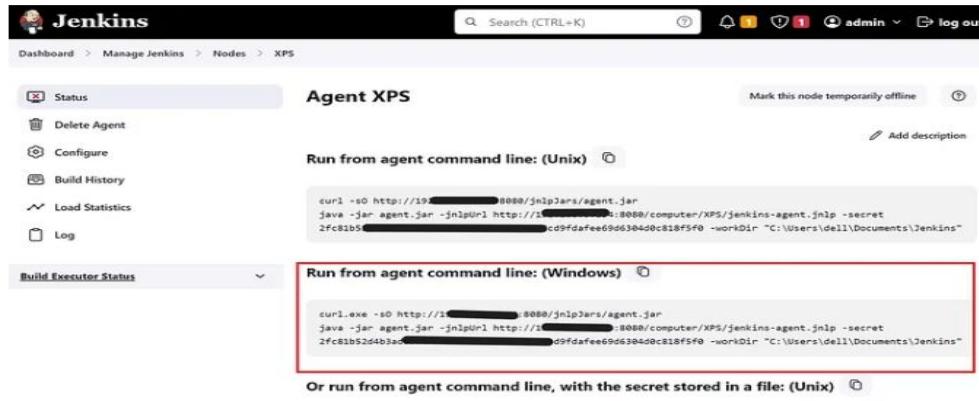
The screenshot shows the Jenkins System Configuration page. The navigation path is: Dashboard > Manage Jenkins > System. The page displays various configuration settings:

- Quiet period:** A text input field containing the value "5".
- SCM checkout retry count:** A text input field containing the value "0".
- Restrict project naming:** A checkbox that is currently unchecked.
- Jenkins Location:** A section containing the **Jenkins URL** field.
- Jenkins URL:** A text input field containing the URL "http://192.168.1.11:8080/". This field is highlighted with a red rectangular border.

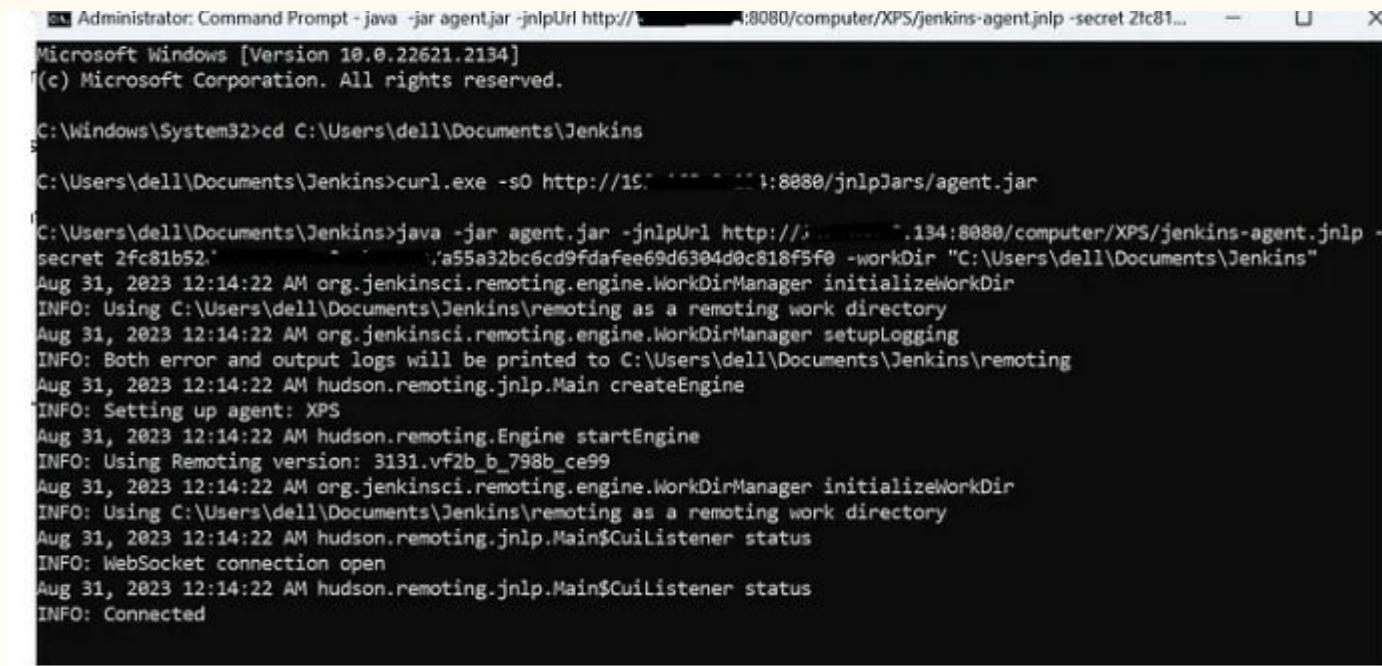
## Adding slave node to Jenkins



- Open the Jenkins dashboard in the browser using the above configured IP address (Ex:<IP of master>:8080).
- In the Dashboard, click on Nodes under System Configuration and click on New Node.
- Enter the name for the node and select the type as permanent agent.
- Click on Create. Add the description if required.
- Now on your slave machine, create an empty folder in any desired location (Ex: C:\Users\dell\Documents\Jenkins).
- Enter the path of this newly created directory in the remote root directory field.
- Enter the label if required. This is useful if you want to execute the job on multiple nodes without changing the configuration.
- In the Launch method, select the option ‘Launch agent by connecting it to controller’.
- Copy the same path entered in the ‘Remote root directory field’ above into the ‘Custom WorkDir path’ field. Select the option ‘Use WebSocket’ checkbox.
- In the availability section, choose the option ‘Keep this agent online as much as possible’ and save the changes.
- The newly created agent shows a red cross mark indicating that the agent is offline. Click on this node. There are 2 commands displayed under this section ‘Run from agent command line: (Windows)’



- Open command prompt as an admin and cd to the path that was entered previously in the ‘remote root directory field’ while creating the node.
- Copy paste both these commands in the cmd prompt and they should execute without any error. Once this is successful, your previously empty Jenkins folder will have a remoting folder and an agent.jar file.



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". The window displays the following terminal session:

```
Administrator: Command Prompt - java -jar agent.jar -jnlpUrl http://[REDACTED]:8080/computer/XPS/jenkins-agent.jnlp -secret 2fc81b52
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Users\dell\Documents\Jenkins

C:\Users\dell\Documents\Jenkins>curl.exe -s0 http://[REDACTED]:8080/jnlpJars/agent.jar

C:\Users\dell\Documents\Jenkins>java -jar agent.jar -jnlpUrl http://[REDACTED]:8080/computer/XPS/jenkins-agent.jnlp -secret 2fc81b52
Aug 31, 2023 12:14:22 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using C:\Users\dell\Documents\Jenkins\remoting as a remoting work directory
Aug 31, 2023 12:14:22 AM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to C:\Users\dell\Documents\Jenkins\remoting
Aug 31, 2023 12:14:22 AM hudson.remoting.jnlp.Main createEngine
INFO: Setting up agent: XPS
Aug 31, 2023 12:14:22 AM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3131.vf2b_b_798b_ce99
Aug 31, 2023 12:14:22 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using C:\Users\dell\Documents\Jenkins\remoting as a remoting work directory
Aug 31, 2023 12:14:22 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: WebSocket connection open
Aug 31, 2023 12:14:22 AM hudson.remoting.jnlp.Main$CuiListener status
INFO: Connected
```

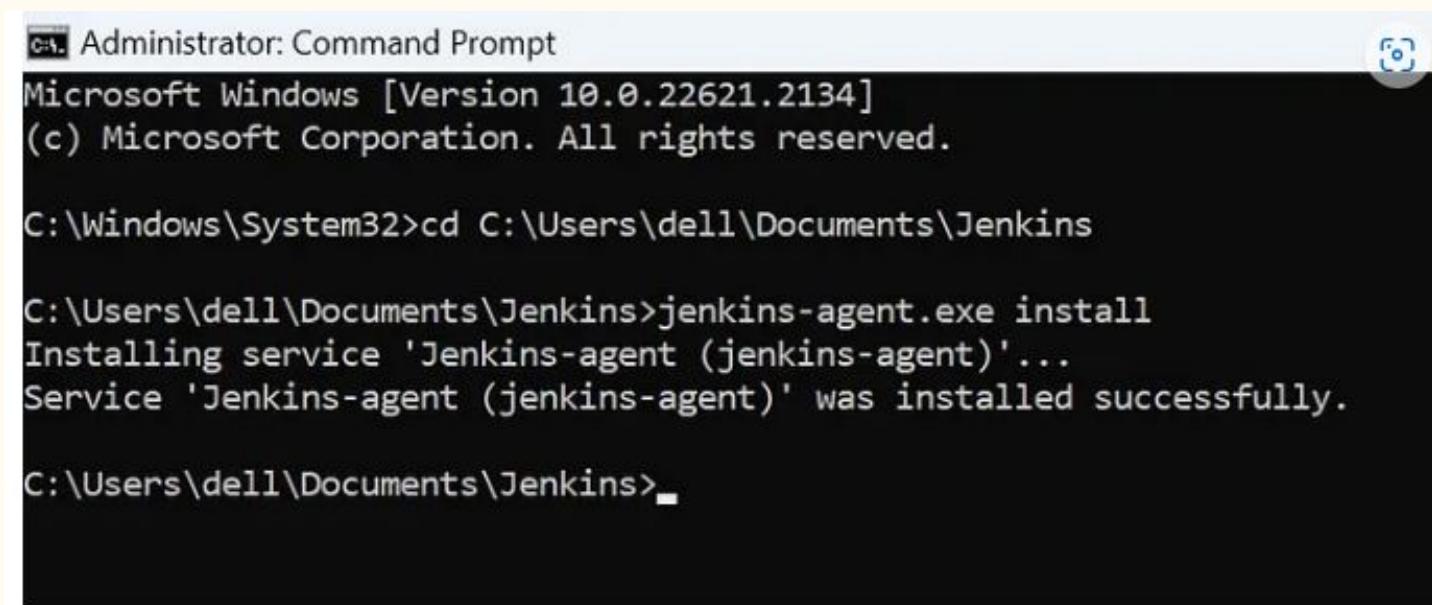
- Now when you refresh your Jenkins UI, it shows that the node is connected and online. But the limitation is that this node gets disconnected as soon as this command prompt window is closed.
- In order to keep the slave node online as much as possible, we will have to set it up as a service.
- For this purpose, we will be using the Windows Service Wrapper(winsw).
- Download the latest WinSW.exe from [Releases · winsw/winsw](#)
- Next, download the jenkins.xml file
- Note that the name of the exe and xml file should be the same.
- Now open the jenkins-agent.xml in edit mode. Change the value of id and name tag to jenkins-agent. Delete the value of arguments tag and replace the it with the command that is displayed in your node page,just remove the first word java and paste everything else as it is)



The screenshot shows a code editor window titled "jenkins-agent.xml". The window has a standard OS X-style title bar with a close button ("x") and a plus sign button ("+"). Below the title bar is a menu bar with "File", "Edit", and "View" options. The main content area contains XML code for a Windows service configuration:

```
<service>
  <id>jenkins-agent</id>
  <name>Jenkins-agent</name>
  <description>This service runs Jenkins continuous integration system.</description>
  <env name="JENKINS_HOME" value="%BASE%" />
  <executable>java</executable>
  <arguments>-jar agent.jar -jnlpUrl http://127.0.0.1:8080/computer/XPS/jenkins-agent.jnlp -secret
|2fc81b[REDACTED]7a55a32bc6cd9fdaf69d6304d0c818f5f0 -workDir "C:\Users\dell\Documents\Jenkins"</arguments>
  <log mode="roll" />
  <onfailure action="restart" />
</service>
```

Now open cmd prompt as an admin, cd to your Jenkins folder path and enter the command: jenkins-agent.exe install. Now the Jenkins-agent is installed as a service successfully.



The screenshot shows an Administrator Command Prompt window on a Windows 10 system. The title bar reads "Administrator: Command Prompt". The window displays the following text:

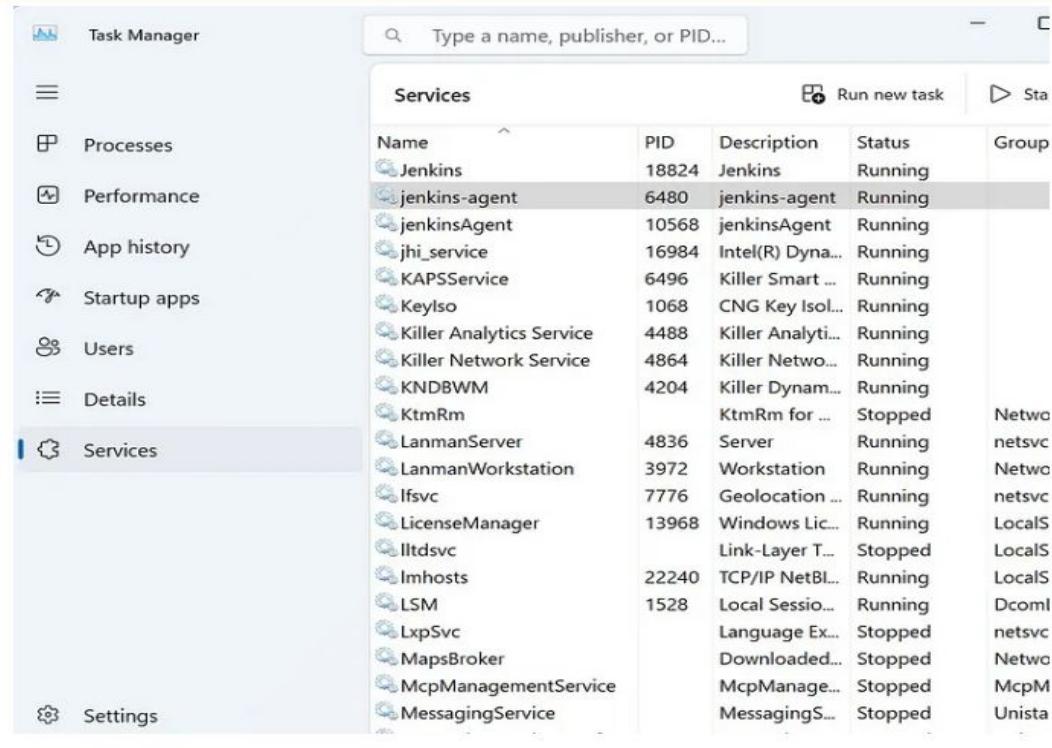
```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Users\dell\Documents\Jenkins

C:\Users\dell\Documents\Jenkins>jenkins-agent.exe install
Installing service 'Jenkins-agent (jenkins-agent)'...
Service 'Jenkins-agent (jenkins-agent)' was installed successfully.

C:\Users\dell\Documents\Jenkins>
```

- Now if you open the Jenkins UI, it's seen the agent is still not connected. This is because the agent service is not yet started.
- Open services tab in task manager, search for the jenkins-agent service, right click on it and click start.



Once this agent is up and running, you can refresh the Jenkins UI and tadaaaaaaa! the slave is connected.

The screenshot shows the Jenkins UI for managing nodes. The top navigation bar includes 'Dashboard', 'Manage Jenkins', 'Nodes', and 'XPS'. On the left, a sidebar lists various options: 'Status' (selected), 'Delete Agent', 'Configure', 'Build History', 'Load Statistics', 'Script Console', 'Log', 'System Information', and 'Disconnect'. The main content area is titled 'Agent XPS' and displays the message 'Agent is connected.' Below this, it says 'Projects tied to XPS' and shows 'None'. The entire interface has a light gray background with blue links and a white header bar.

You can now create multiple slave nodes in this way and assign builds to all the slave nodes from the master node and perform parallel execution.

# Selenium integration with Jenkins

## Step 1: Install Jenkins Plugins

The functionalities of Jenkins are enhanced using numerous plugins. To work with Selenium, you'll need to install a couple of plugins:

Selenium Plugin: Go to Jenkins Plugin Manager and install “Selenium Plugin”. Use this plugin for running Selenium tests on a Jenkins pipeline.

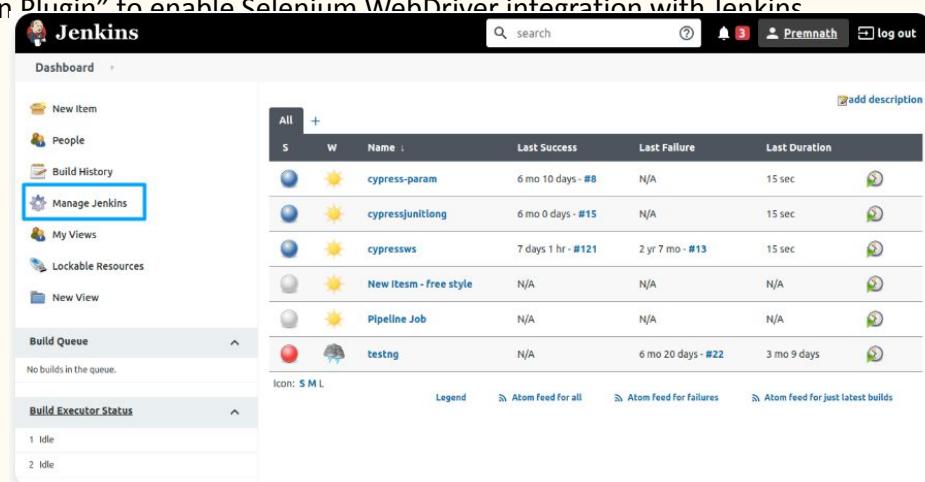
WebDriver Automation Plugin: Install the “WebDriver Automation Plugin” to enable Selenium WebDriver integration with Jenkins

## Step 2: Configure Jenkins

Once the plugins are installed, you need to configure Jenkins:

Open Jenkins in your web browser.

On the left sidebar, click “Manage Jenkins”.



The screenshot shows the Jenkins dashboard. On the left, there is a sidebar with the following options: New Item, People, Build History, Manage Jenkins (which is highlighted with a blue box), My Views, Lockable Resources, and New View. Below the sidebar are sections for Build Queue (No builds in the queue) and Build Executor Status (2 Idle). The main area displays a table of Jenkins jobs. The columns are: All, S, W, Name, Last Success, Last Failure, and Last Duration. The jobs listed are: cypress-param, cypressjunitlong, cypressws, New items - free style, Pipeline Job, and testng. Each job entry includes a green circular icon with a white sun, indicating they are currently successful. The last success column shows dates ranging from 6 months ago to N/A. The last failure column shows dates ranging from 2 years ago to N/A. The last duration column shows times ranging from 15 seconds to 20 days. At the bottom of the table, there are links for Atom feed for all, Atom feed for failures, and Atom feed for just latest builds. A legend at the bottom indicates that green means successful and red means failed.

All	S	W	Name	Last Success	Last Failure	Last Duration
			cypress-param	6 mo 10 days - #8	N/A	15 sec
			cypressjunitlong	6 mo 0 days - #15	N/A	15 sec
			cypressws	7 days 1 hr - #121	2 yr 7 mo - #13	15 sec
			New items - free style	N/A	N/A	N/A
			Pipeline Job	N/A	N/A	N/A
			testng	N/A	6 mo 20 days - #22	3 mo 9 days

Navigate to “Global Tool Configuration.”

The screenshot shows the Jenkins Global Tool Configuration interface. At the top, there is a navigation bar with links for 'Dashboard' and 'Global Tool Configuration'. The 'Global Tool Configuration' link is highlighted with a blue box. On the left, there are two navigation links: 'Back to Dashboard' and 'Manage Jenkins'. The main content area has a title 'Global Tool Configuration' with a wrench and screwdriver icon. It contains three sections: 'Maven Configuration', 'JDK', and 'Git'. The 'Maven Configuration' section has dropdown menus for 'Default settings provider' (set to 'Use default maven settings') and 'Default global settings provider' (set to 'Use default maven global settings'). The 'JDK' section has a 'JDK installations' list with an 'Add JDK' button. The 'Git' section has a 'Git installations' list with a 'Git' icon. At the bottom, there are 'Save' and 'Apply' buttons.

Jenkins

Dashboard > Global Tool Configuration

Global Tool Configuration

Maven Configuration

Default settings provider: Use default maven settings

Default global settings provider: Use default maven global settings

JDK

JDK installations: Add JDK

List of JDK installations on this system

Git

Git installations: Git

Save Apply

Under “Selenium WebDriver,” add the path to your Selenium WebDriver executable.

Save the configuration.

### Step 3: Create a Jenkins Job

To automate Selenium tests using Jenkins, you need to create a Jenkins job:

Navigate to the ‘New Item’ option on the Jenkins dashboard.

Give your job a name and select “Freestyle project.”

Under the “Build” section, click on “Add build step” and select “SeleniumHQ htmlSuite Run.”

Configure your Selenium tests by providing the path to your test suite, browser settings,

and other relevant details.

Save the job configuration.

The image consists of two side-by-side screenshots of the Jenkins web interface. The top screenshot shows the Jenkins dashboard with a modal window for creating a new item. The 'Freestyle project' option is selected, and a validation error message 'This field cannot be empty, please enter a valid name' is displayed below the input field. The bottom screenshot shows the configuration page for a 'Free Style Project'. It includes tabs for General, Source Code Management, Build Triggers, Build Environment, Build, and Post-build Actions. Under the General tab, there is a 'Description' text area and a checkbox for 'Discard old builds'. At the bottom of the page are 'Save', 'Apply', and 'Advanced...' buttons.

## Step 4: Configure Jenkins Build Triggers

To trigger your Selenium tests automatically, you can set up build triggers. The most common trigger is to run the tests when code is pushed to a specific branch in your version control system (e.g., GitHub). To set up GitHub webhook triggers:

In your Jenkins job configuration, go to the “Build Triggers” section.

Check the option “GitHub hook trigger for GITScm polling.”

Save your job configuration.

The screenshot shows the Jenkins configuration interface for a "Free Style Project". The "Build Triggers" tab is active. In the "Build Triggers" section, the "GitHub hook trigger for GITScm polling" checkbox is checked, while others like "Trigger builds remotely" and "Poll SCM" are unchecked. The "Build Environment" section contains several other configuration options. At the bottom, there are "Save" and "Apply" buttons.

Build Trigger	Status
Trigger builds remotely (e.g., from scripts)	<input type="checkbox"/>
Build after other projects are built	<input type="checkbox"/>
Build periodically	<input type="checkbox"/>
GitHub hook trigger for GITScm polling	<input checked="" type="checkbox"/>
Poll SCM	<input type="checkbox"/>

Build Environment Option	Status
Delete workspace before build starts	<input type="checkbox"/>
Use secret text(s) or file(s)	<input type="checkbox"/>
Abort the build if it's stuck	<input type="checkbox"/>
Add timestamps to the Console Output	<input type="checkbox"/>
Inspect build log for published Gradle build scans	<input type="checkbox"/>
With Ant	<input type="checkbox"/>

**Build Triggers**

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

**Build Environment**

- Delete workspace before build starts ?
- 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 ?

**Buttons**

Save    Apply

# Step 5: Build and Test

With everything set up, you can now build and test your Selenium integration with Jenkins:

Click on your Jenkins job from the dashboard.

Click “Build Now” to start a manual build, or wait for automatic triggers to initiate the tests.

Jenkins will execute your Selenium tests and display the results in the job console output.

