EXPERIMENT - 1

SDLC Implementation

Aim : To understand the Software development significance in Software Project Development

Software: Software is a set of computer programs and associated documentation and data.

Software Project: Software is a set of computer programs and associated documentation and data.

Software Development Life Cycle:

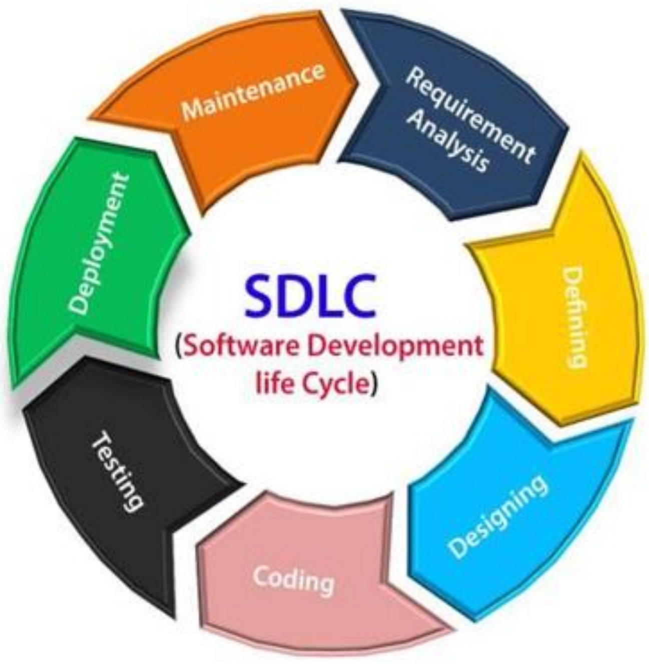
• A collection of phases in a sequence

• SDLC describes project planning, development, refine, maintain and deployment phases

• SDLC consists of 6 stages namely

PLANNING DEFINING, DESIGNING BUILDING

TESTING and DEPLOYMENT



Planning:

Basing on the client needs preliminary surveys and analysis conducted in this phase to initiate Project.

• Feasibility Study

• Domain Study

• Requirement Analysis

• Technical Resources Analysis

• Risk associations Identification

Defining:

The SRS (Software Requirement Specification) document is prepared in this phase by Software Engineer with assistance of System Analyst.

• Requirement Report Verification

• Market Analysis

• Product Requirements

• Resource Requirements

• Cost Estimation

• Staff Estimation

• Build Assets Requirements

Designing:

Based on SRS Designing process modularizes the project based on several constraints. Design Document

Specification generated at the end of this phase.

• Best Architecture Recognition

• Stakeholders Parameters

• Product Robustness

• Design Modularity

• Budget Constraints

• Data Flow among Modules

• Collaboration of Modules

• Module dependencies

Developing:

In this stage of SDLC the actual development starts and the product is built. The Developing Product

Specification generated at the end of phase

• Different HLL software used by developers

• Coding developed under guidelines of System Analyst

• Team Leaders manages individual modules with collaboration to other teams

• Organization programming tools used for code generation

• Code Optimization conducted

• Necessary Compilers, IDEs, Languages and Packages are employed in code generation

Testing:

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC.

• Unit testing

• Integration Testing

• System Testing

• UAT - User Acceptance Testing

Deployment & Maintenance:

In this phase product is released into market with maintenance and help desk support

services. The maintenance is done by customer.

• Alpha Release (Within R & D environments and developing zones)

• Beta Release (Inside the territory with survey of feedback from customers)

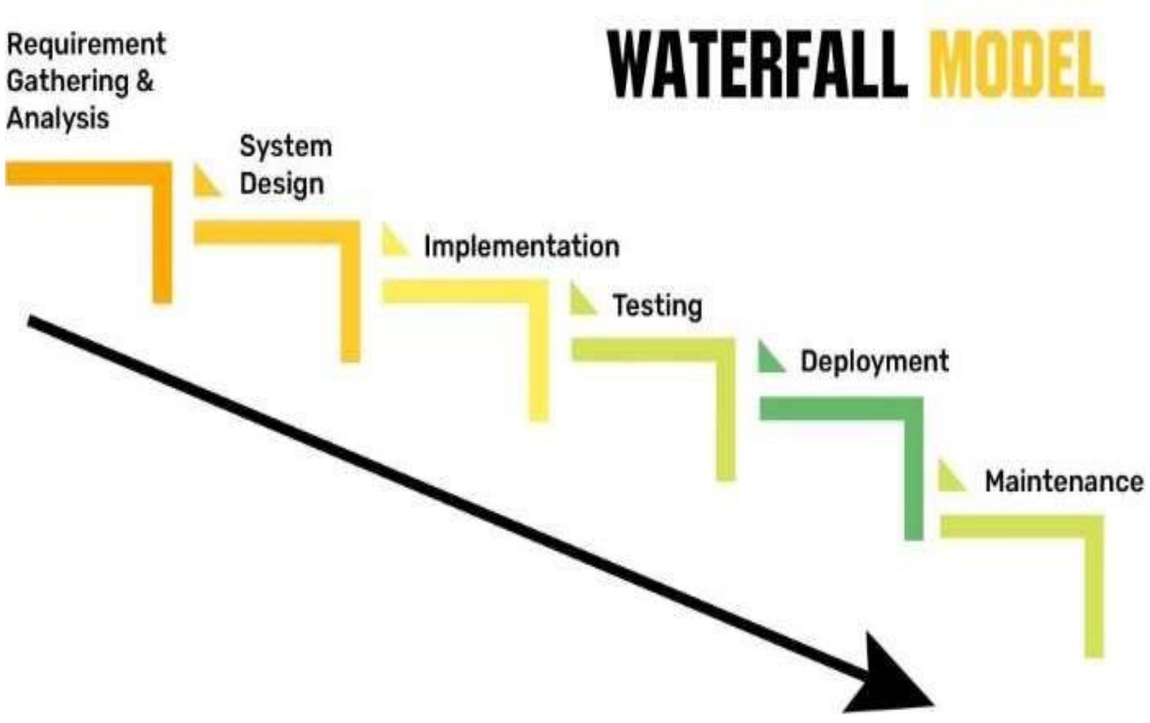
• Customer Assistance (After Global release)

• Help Desk (Suggestions and improvement strategies)

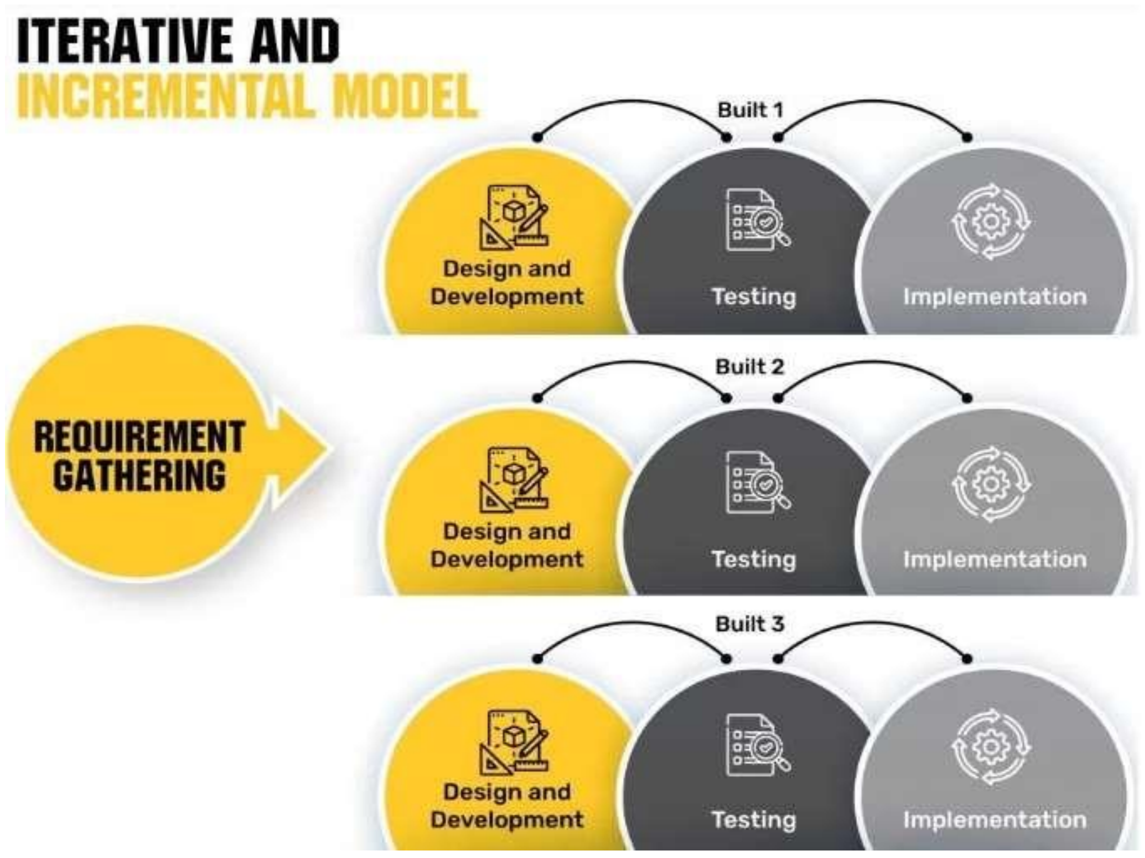
• Target Market strategies (market analysis of product usage)

• Real business environment (Product Working environments)

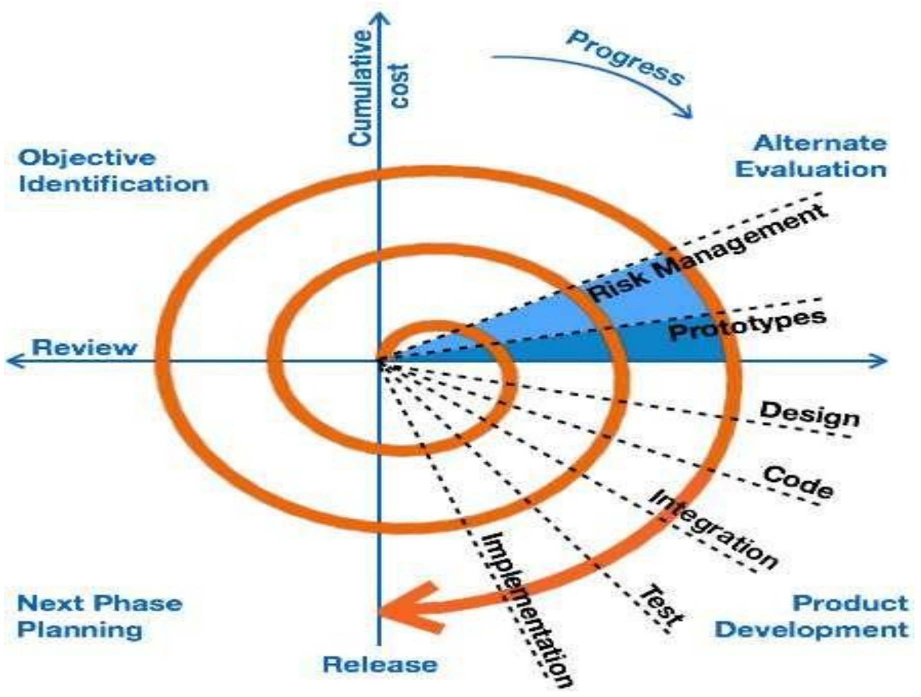
SDLC Models: WATER-FALL model



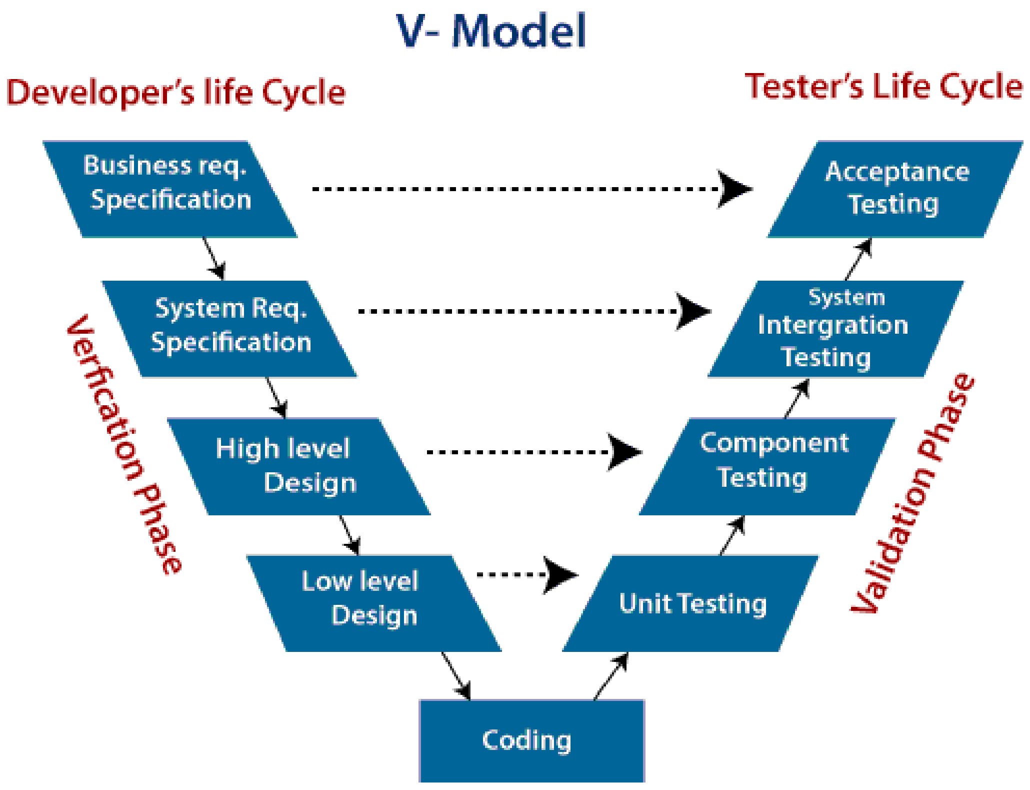
SDLC Models: ITERATIVE and INCREMENTAL model



SDLC Models: The SPIRAL MODEL

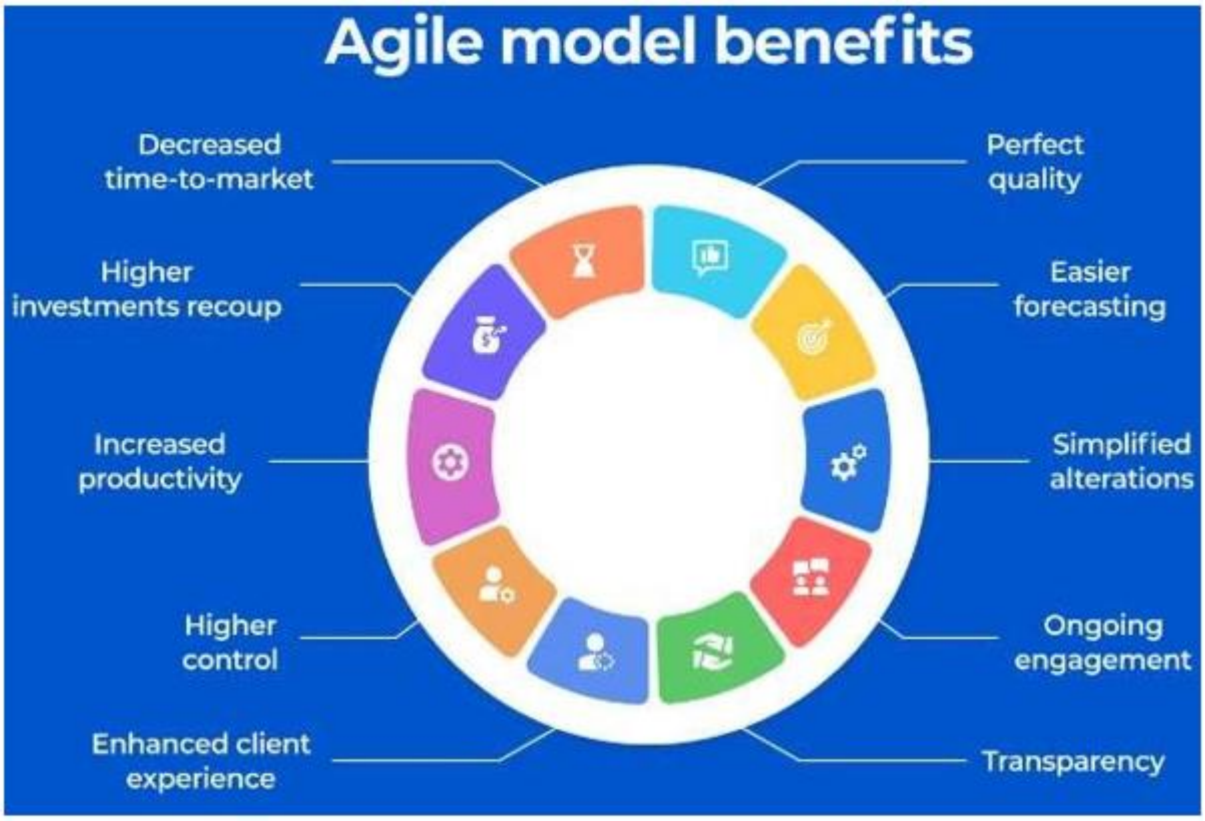


SDLC Models: The V-MODEL



SDLC Models: The AGILE MODEL

- Combination of Iterative and Incremental process models
- Process adaptability high
- Customer satisfaction with rapid delivery
- Increment builds based
- Minimizes Failures
- High Quality modules
- Low Cost and Risk
- Continuous Adaptability to Technologies

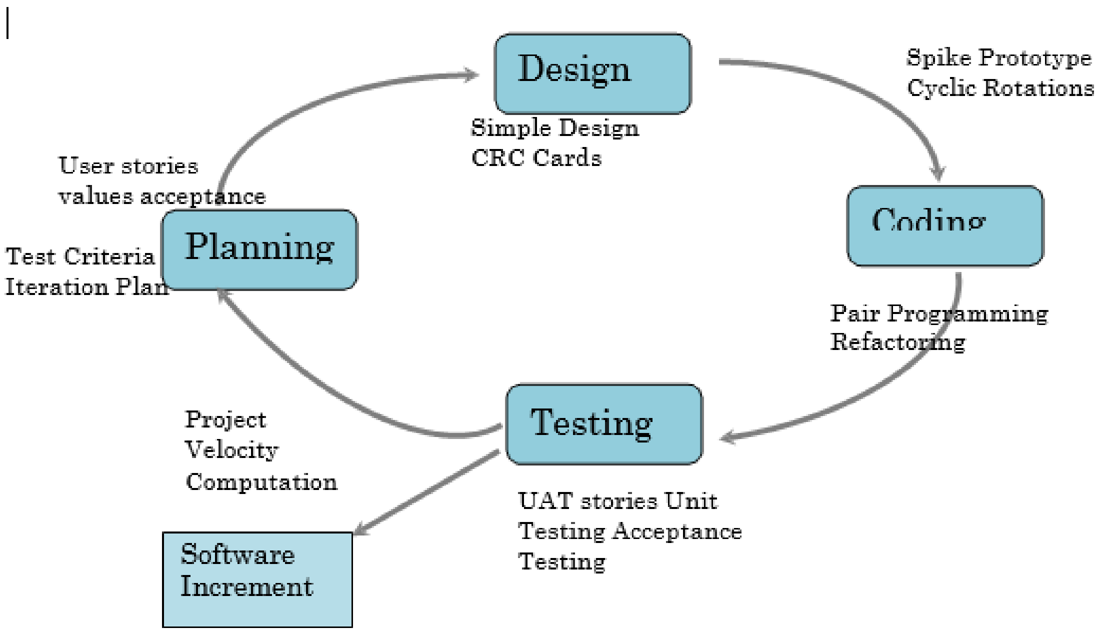


EXPERIMENT - 2

Aim: Understanding Extreme Programming concepts in Agile Programming

Description:

- Most widely used approach in Agile software development
- Depends on Communication, Simplicity, Feedback, Courage and Respect Values
- Simplicity achieved through design based on immediate needs with refactoring later mode
- Feedback derived from customer, teams and designers. functionality based testing implemented
- Establishes discipline among groups of project with strict practices on demand
- Design today if necessary modify tomorrow mode is applied
- XP performs various acceptance tests such as customer tests, stakeholder's tests and Functionality tests.
- Build stories based on software implementation groups to generate acceptance test cases.



XP programming Values:

Communication	Informal Collaboration, metaphors, Documentation
Simplicity	Needs base, design simplicity, code compactness
Feedback	Customer, Designer, stakeholder, Project leaders
Courage	Commitment, Discipline, recognition
Respect	Equity, Trust between stakeholders, customers, teams

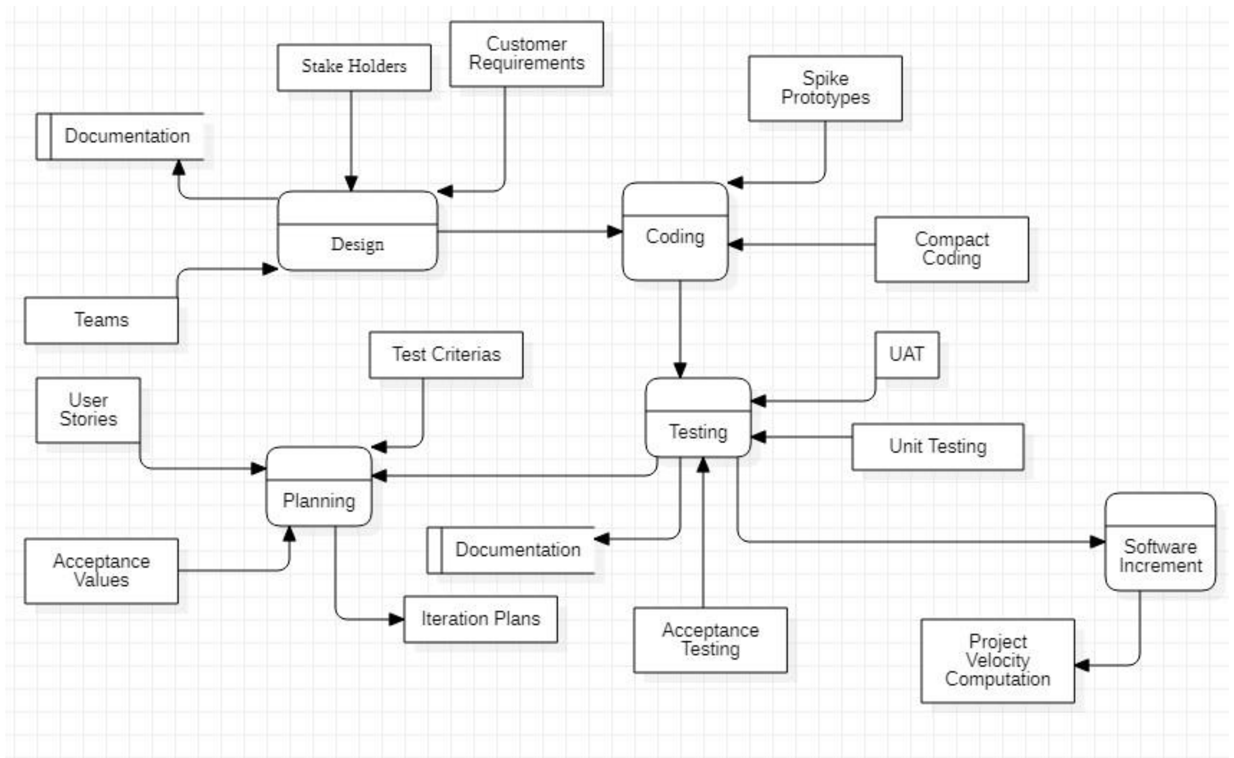
Test First Programming:

Developing code → Write Test strategies → Implement Test Scenarios → Run tests → Repeat

Refactoring in Agile:

Refactoring is the practice of continuously improving the design of existing code, without changing the fundamental behavior. In Agile, teams maintain and enhance their code on an incremental basis from Sprint to Sprint. If code is not refactored in an Agile project, it will result in poor code quality, such as unhealthy dependencies between classes or packages, improper class responsibility allocation, too many responsibilities per method or class, duplicate code, and a variety of other types of confusion and clutter. Refactoring helps to remove this chaos and simplifies the unclear and complex code.

EXtreme Programming using Data Flow Diagram:



DATA FLOW DIAGRAM

EXPERIMENT - 3

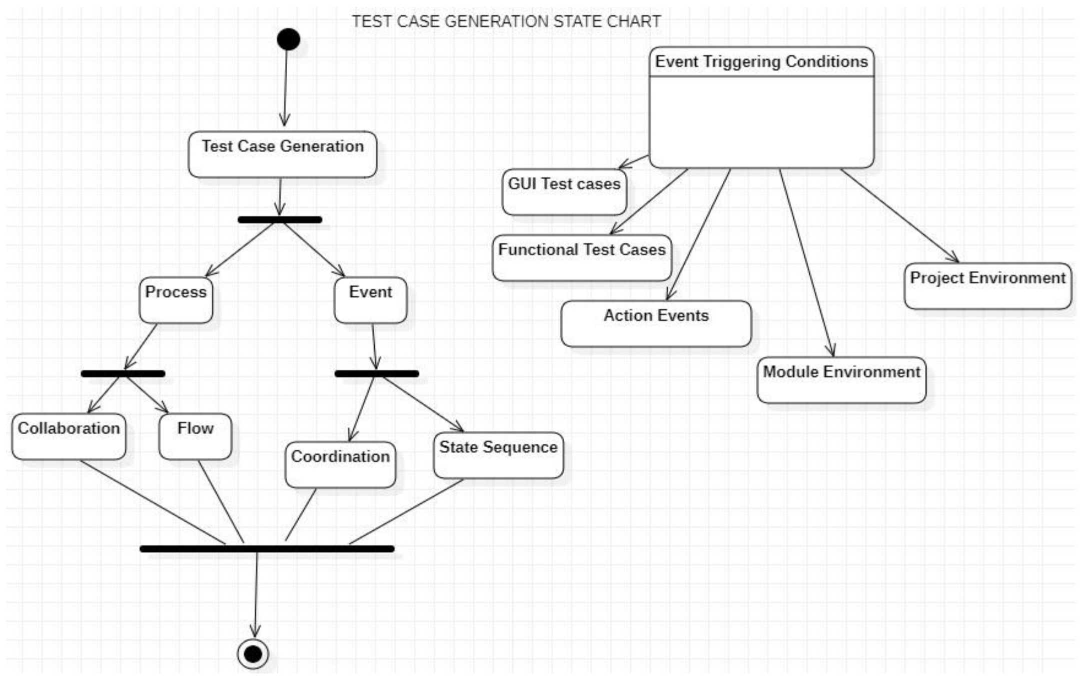
Aim: Understand Test Case Auto Generation Logic in Agile Programming

Description:

- Agile programming Test case scenarios are auto generated
- Test First Program philosophy implemented
- Testing environment is highly automated and less human intervention
- Adopts scenarios by learning methods using AI methods
- Test cases generation is implemented over Process and Event categories
- Events are generated with user interaction with system through services of software
- GUI based, Functional based, Action triggered, Module generated and finally Project environment generated events are considered
- In agile programming collaboration and control flow given preference to improve process quality
- In communication perspective event coordination and sequence of transitions among communication channel must be optimized
- Test case scenarios are implemented in planning phase in agile refactoring process purely generated basing on stories generated by teams

Automated Testing:

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.



Test Case Automation Steps:

- Decide what Test Cases to Automate
- Select the Right Automated Testing Tool
- Divide your Automated Testing Efforts
- Create Good, Quality Test Data
- Create Automated Tests that are Resistant to Changes in the UI

EXPERIMENT – 4

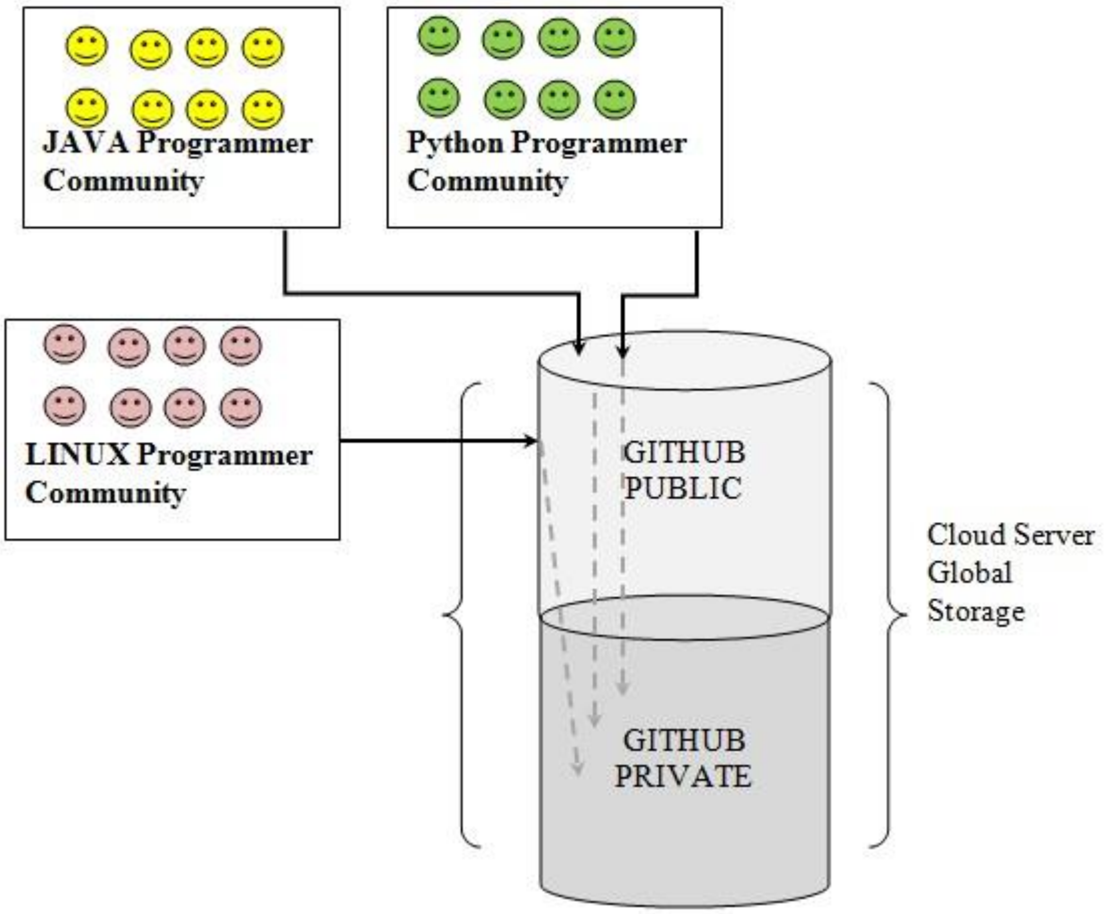
Aim : Creating and managing GIT-HUB account for initializing Community Programming

Description :

Community Programming

A moderate programming approach groups of citizens, academicians, programmers across the globe share their unique experiences and perspectives from various programming environments across communities. Improve knowledge and debugging abilities of programmers. Considered as one of the Agile programming supportive tool.

- Provides solutions to high end programs quickly
- Improves Libraries and services of applications
- Supports GNU licensing
- Connects academicians, programmers and designers in one platform
- Share knowledge and coding techniques across the globe
- Supported by many Cloud service Giants like AWS (Amazon Web Service), CISCO Cloud Service, ORACLE Cloud and IBM Cloud services
- Highly used by Students and teaching communities in the world
- Essential for Dev-Ops projects


JAVA Programmer Community
Python Programmer Community
LINUX Programmer Community
GITHUB PUBLIC
GITHUB PRIVATE
Cloud Server Global Storage

GIT-HUB

Create a Local repository. Folder in your local disk for repository purpose

Keep all the project files in local repository

Creating GIT-HUB repository

1. Log into GIT-Hub administrative console
2. Move to GIT-Hub repositories page
3. Click on green button "Create Repository"
4. Opens a wizard for creation
5. Enter name of the repository
6. Include description
7. Choose to make use by public or private
8. Add Read-me option
9. Include a .gitignore file for your development framework
10. Choose fair use license
11. Click finally green button

Load Eclipse software and Project to GIT-Hub:

- First place your project/file in local repository
- Go to File menu select import
- Form the available select "Existing Maven Projects"
- It opens Maven projects root directory to load project from local
- Open share project window select "project team share"
- Login to GIT-Hub before sharing
- Create GIT-Repository with local repository selection
- After finish select package explorer and apply team commit
- Now Open your GIT-Hub account
- Select all un staged changes and apply index
- Project explorer → Team Push → Branch Master
- Provide authentication preview two times
- Now your project appear in GIT-Hub repository

EXPERIMENT – 5

Aim: Using Git-Bash environment to create repository and stage a file with performing commits

Description:

1. Installing Git in Local System?

   Visit the web site https://git-scm.com/download/win choose 64-bit Git for Windows setup

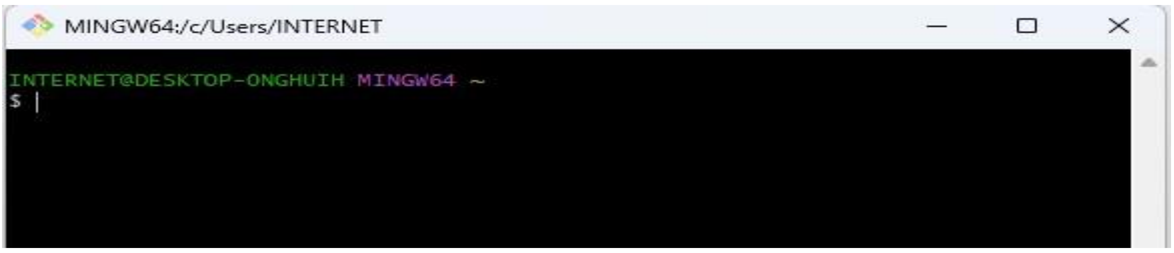
   Run the setup file

   Select next until entire setup process completes

   If you need startup icon or runtime icon you can select checkboxes else left them unselect

   After completion of installation

   Git BASH console opened on screen as follows



```
MINGW64:/c/Users/INTERNET
INTERNET@DESKTOP-ONGHUIH MINGW64 ~
$ |
```

2. Creating Environment for Git-Local?

   set environmental path for GIT

   type advanced system settings in search box select <Environment Variables> button select <New> button

   Give a Name like "GITBash" in Variable Name set Variable path as given below "C:\Users\INTERNET\AppData\Local\Programs\Git\git-bash.exe"

3. Creating your Identity in GIT?

```
$ git config --global user.name "Dr BVRK"
$ git config --global user.password "123342"
$ git config --global user.email “bhvram78@gmail.com”
$ git config user.name
Dr BVRK
$ git config user.email bhvram78@gmail.com
$ git config user.password
123342
```

4. Create a new local repository in any folder?

```
$ git init
Initialized empty Git repository in E:/MYGitFiles/.git/
```

5. Stage files into repository created?

```
$ git add .
SamhiSesi@DESKTOP-F7HHS75 MINGW64 /e/mygitfiles (master)
$ git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   ReadMe.txt new file:   bvr1.java
```

6. Perform commit over staged documents

```
$ git commit -m "Performing Commit 002"
[master (root-commit) 28d6b76] Performing Commit 002
 2 files changed, 16 insertions(+) create mode 100644 ReadMe.txt create mode 100644
 bvr1.java
```

7. Performing un commit last operations

```
$ git reset --soft HEAD
            SamhiSesi@DESKTOP-F7HHS75 MINGW64 /e/mygitfiles (master)
$ git status
On branch master
nothing to commit, working tree clean
            SamhiSesi@DESKTOP-F7HHS75 MINGW64 /e/mygitfiles (master)
$ git reset HEAD
            SamhiSesi@DESKTOP-F7HHS75 MINGW64 /e/mygitfiles (master)
$ git status
On branch master
nothing to commit, working tree clean
            SamhiSesi@DESKTOP-F7HHS75 MINGW64 /e/mygitfiles (master)
$ ls
ReadMe.txt  bvr1.java
```

EXPERIMENT – 6

Aim: Using JUNIT for conducting unit test over GIT-HUB

Description:

Visual Studio Code:

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and Mac-OS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded GIT.

- Visual Studio Code features a lightning fast source code editor, perfect for day-to-day use. With support for hundreds of languages,
- VS Code helps you be instantly productive with syntax highlighting, bracket-matching, auto- indentation, box-selection, snippets, and more.
- Intuitive keyboard shortcuts, easy customization and community-contributed keyboard shortcut mappings let you navigate your code with ease.
- Customize every feature to your liking and install any number of third-party extensions.
- VS Code includes enriched built-in support for Node.js development with JavaScript and Type Script, powered by the same underlying technologies that drive Visual Studio.
- VS Code also includes great tooling for web technologies such as JSX/React, HTML, CSS, SCSS,, and JSON
- Architecturally, Visual Studio Code combines the best of web, native, and language-specific technologies.
- Visual Studio Code includes a public extensibility model that lets developers build and use extensions, and richly customize their edit-build-debug experience.

Installation of Visual Studio Code:

Step 1: Visit the official website of the Visual Studio Code using any web browser like Google Chrome, Microsoft Edge, etc.

Step 2: Press the “Download for Windows” button on the website to start the download of the Visual Studio Code Application.

Step 3: When the download finishes, then the Visual Studio Code icon appears in the downloads folder.

Step 4: Click on the installer icon to start the installation process of the Visual Studio Code.

Step 5: After the Installer opens, it will ask you for accepting the terms and conditions of the Visual Studio Code. Click on I accept the agreement and then click the Next button.

Step 6: Choose the location data for running the Visual Studio Code. It will then ask you for browsing the location. Then click on Next button.

Step 7: Then it will ask for beginning the installing setup. Click on the Install button.

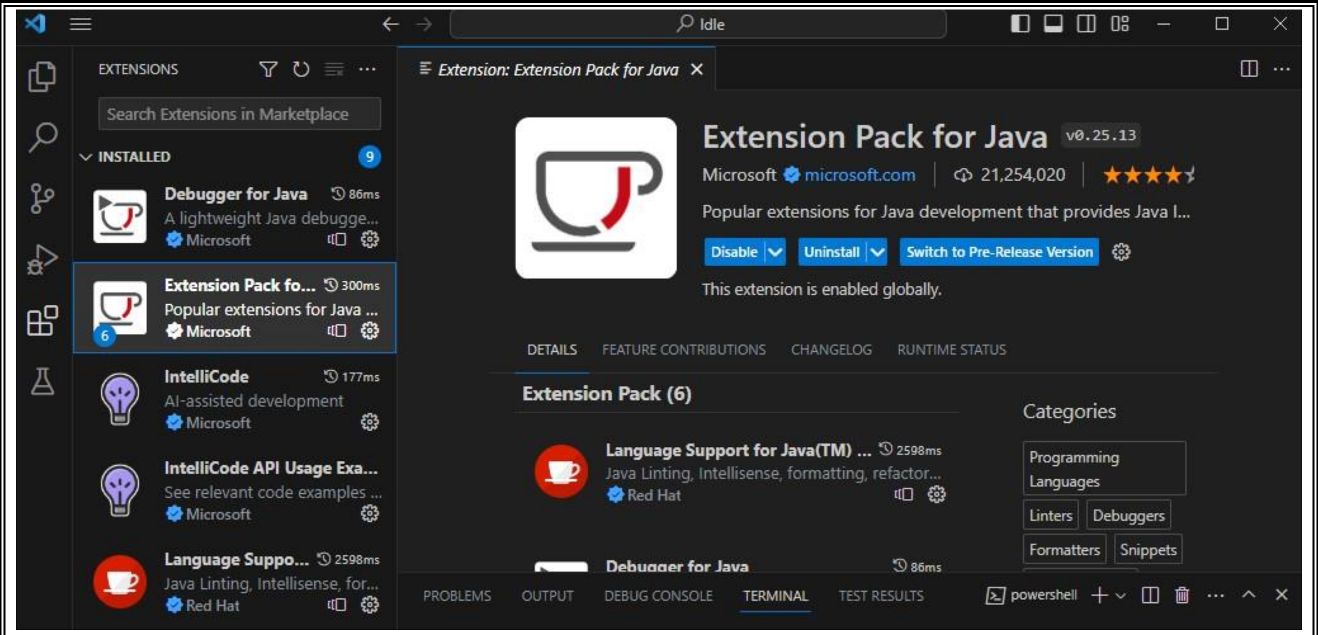
Step 8: After clicking on Install, it will take about 1 minute to install the Visual Studio Code on your device.

Step 9: After the Installation setup for Visual Studio Code is finished, it will show a window like this below. Tick the "Launch Visual Studio Code" checkbox and then click Next.

Step 10: After the previous step, the Visual Studio Code window opens successfully. Now you can create a new file in the Visual Studio Code window.

Performing Unit Test using JUNIT in VS-Code:

- Open Visual Studio Code editor
- Select Extensions icon from left tool bar or use short cut key (Ctrl + Shift + X)
- A Tab opened after tool bar displaying Extensions Installed button, Recommended button
- Also text bar for searching installed extensions with filter and refresh options
- Select "Extension Pack for Java" from recommended list
- Select "Install" button to download extension pack as built in library for VS-Code
- The pack includes features as
    - Extension pack includes
    - Java Language Library
    - Java Debugger
    - Test Runner for JUNIT/TestNG tool
    - Maven for Java
    - Project Manager for Java
    - IntelliCode-AI assisted


Idle
EXTENSIONS
Search Extensions in Marketplace
INSTALLED 9
Debugger for Java 86ms
A lightweight Java debugge...
Microsoft
Extension Pack fo... 300ms
Popular extensions for Java ...
Microsoft
IntelliCode 177ms
AI-assisted development
Microsoft
IntelliCode API Usage Exa...
See relevant code examples ...
Microsoft
Language Suppo... 2598ms
Java Linting, Intellisense, for...
Red Hat
Extension: Extension Pack for Java
Extension Pack for Java v0.25.13
Microsoft microsoft.com 21,254,020
Popular extensions for Java development that provides Java I...
Disable
Uninstall
Switch to Pre-Release Version
This extension is enabled globally.
DETAILS
FEATURE CONTRIBUTIONS
CHANGELOG
RUNTIME STATUS
Extension Pack (6)
Language Support for Java(TM) ... 2598ms
Java Linting, Intellisense, formatting, refactor...
Red Hat
Debugger for Java 86ms
Categories
Programming Languages
Linters
Debuggers
Formatters
Snippets
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
TEST RESULTS
powershell

Testing using JUNIT:

Create a folder on desktop
named "IDLE" open
explorer with IDLE folder
selction

use popup window to add new file with name "Myprg.java"

enter the following code
in Myprg.java public
class bvr1

```
{
    public String pword()
    {        return ("Hello");}
}
```

save the file

select testing tool in tool bar ...currently no testing
tool associated select enable java tests button

The installation of JUNIT package begins automatically

wait until installation finished

Now you can watch LIB folder under your directory"IDLE" with
library added junit-platform-console-standalone-1.10.0.jar

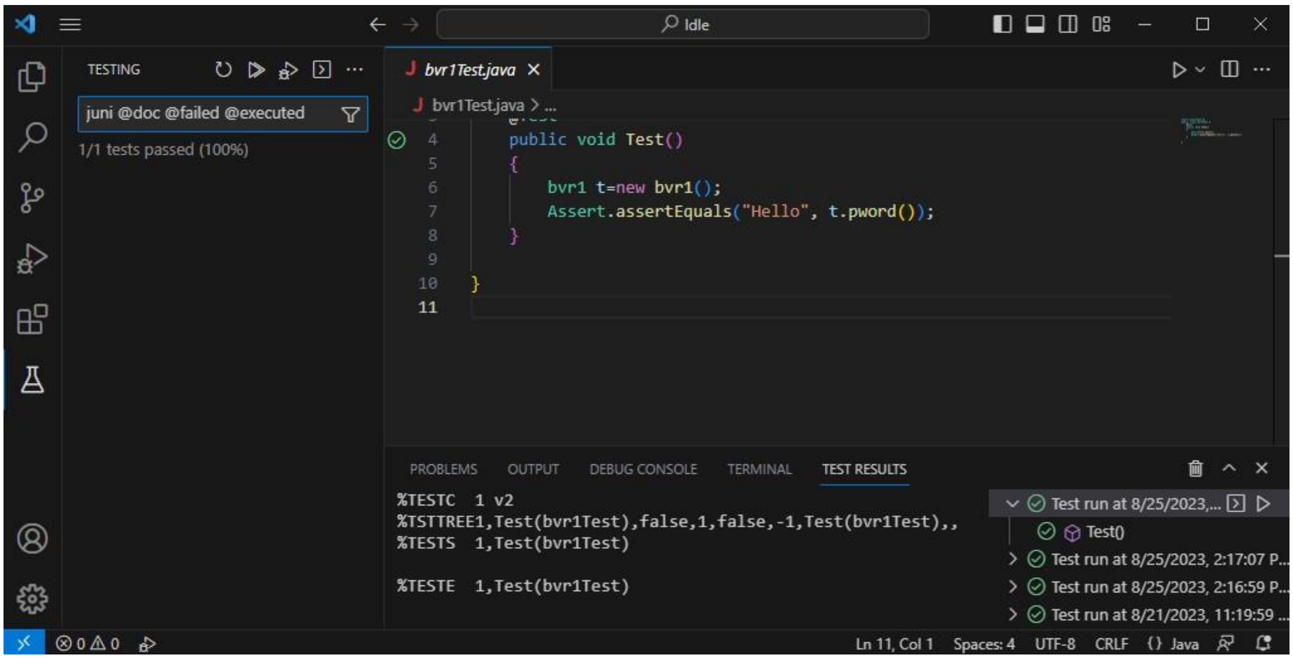
Now create a file named "MyprgTest.java" begin typing code as

```
import org.junit.*;

public class bvr1Test {

   @Test

   public void Test()

   {

      bvr1 t=new bvr1();
      Assert.assertEquals("Hell
      o", t.pword());

   }


}
```

VSCode assists you with opening methods automatically

Select testing tool in toolbar markers appear in your MyTest.java file

Choose only Test() method block and click process begins and reports status of results



```
    public void Test()
    {
        bvr1 t=new bvr1();
        Assert.assertEquals("Hello", t.pword());
    }

}
```

```
%TESTC  1 v2
%TSTTREE1,Test(bvr1Test),false,1,false,-1,Test(bvr1Test),,
%TESTS  1,Test(bvr1Test)

%TESTE  1,Test(bvr1Test)
```

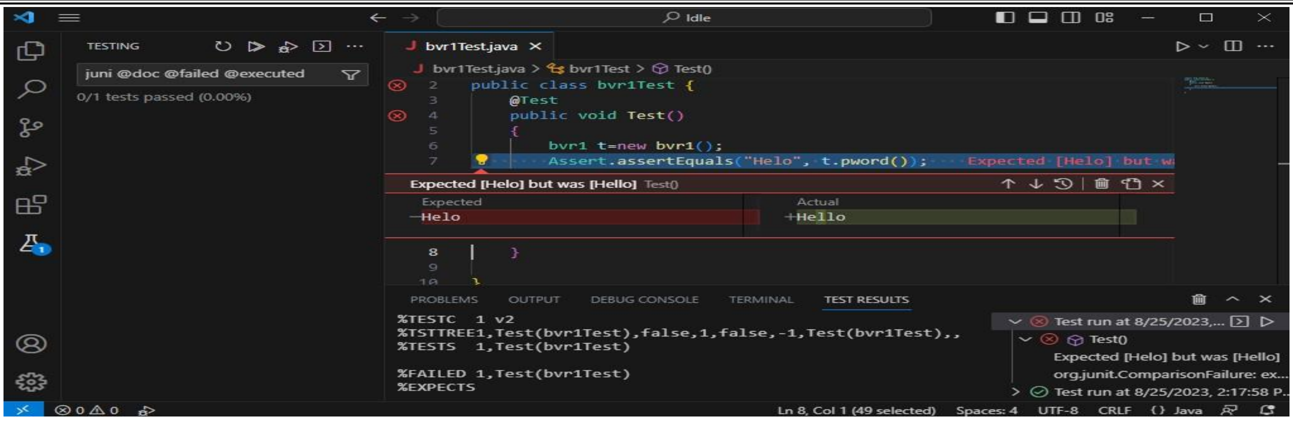
Idle
TESTING
juni @doc @failed @executed
1/1 tests passed (100%)
bvr1Test.java
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
TEST RESULTS
Test run at 8/25/2023,...
Test()
Test run at 8/25/2023, 2:17:07 P...
Test run at 8/25/2023, 2:16:59 P...
Test run at 8/21/2023, 11:19:59 ...
Ln 11, Col 1 Spaces: 4 UTF-8 CRLF {} Java

Now go to MyTest.java file and change argument value “Hello” to “Helo”

Perform testing again JUNIT detects and reports errors in code as follows

Generating Test failed report at console.

Similarly we can check any java project module in this environment effectively.


Idle
TESTING
juni @doc @failed @executed
0/1 tests passed (0.00%)
bvr1Test.java
bvr1Test.java > bvr1Test > Test()

```java
public class bvr1Test {
    @Test
    public void Test()
    {
        bvr1 t=new bvr1();
        Assert.assertEquals("Helo", t.pword());    Expected [Helo] but wa
```

Expected [Helo] but was [Hello] Test()

```
Expected                         Actual
-Helo                            +Hello
```

```java
    }

}
```

```
%TESTC  1 v2
%TSTTREE1,Test(bvr1Test),false,1,false,-1,Test(bvr1Test),,
%TESTS  1,Test(bvr1Test)

%FAILED 1,Test(bvr1Test)
%EXPECTS
```

Test run at 8/25/2023,...
Test()
Expected [Helo] but was [Hello]
org.junit.ComparisonFailure: ex...
Test run at 8/25/2023, 2:17:58 P...

Ln 8, Col 1 (49 selected) Spaces: 4 UTF-8 CRLF {} Java

EXPERIMENT – 7

Aim: Using GRADDLE for CICD Pipeline

Description:

GRADLE BUILD TOOL:

Gradle is Build Tool for Java artifacts construction with many features supporting enterprise solutions

- Incremental Builds
- Build Caching
- Subtask strategies
- Annotation processing
- Compilation of Artifacts
- Continuous Build with Dry Runs Dependency Management in Gradle: Substitution of

Compatible Libraries:

Use dependency substitution rules to identify that dependency should be treated as similar. Tell Gradle that only one should be selected and use Gradle conflict resolution to pick the newest version from both of them. Similar use cases are situations where you have libraries like spring-all and spring-core in dependency graph. Without properly modeling this the proper behavior of your application depends on the very fragile order in your class path.

Enhanced Metadata Resolution Support:

Dependency metadata can be modified after repository metadata is downloading but before it is chosen by Gradle as the final resolved version. This allows the creation of custom rules to do things like declare modules as changing go on.

Replacement of external and project dependencies:

Dynamically replace external dependencies for project dependencies and vice versa. Especially helpful when only a subset of your modules are checked out locally.

Resolved dependency versions can be dynamically:

Gradle supports the Maven snapshot mechanism but is more powerful than that. You can declare a dependency on the latest release, most current development version, or even the latest 5.X build.

Dynamic Dependency Locking:

Allow builds to remain deterministic and reproducible when using dynamic dependency versions.

Dynamic Dependencies Selection Rules:

Define custom rules to select a specific version when a dynamic dependency is declared. The rules can be based on names and version but also extended metadata like branch or status. The rules can also differ based on the environment the build is happening, e.g. local or CI.

Dependency Version Alignment:

Dependency alignment allows different modules in a logical group. Like JSON Modules aligned by versions.

Maven and Ivy Repository Compatible:

Gradle is compatible with the POM & IVY Metadata formats and can retrieve dependencies from any Maven or IVY compatible repository.

Native BOM support:

Platform definitions, aka Maven BOM dependencies are natively supported, allowing importing things like the Spring Boot platform definition without using an external plug-in.

Installing Gradle:

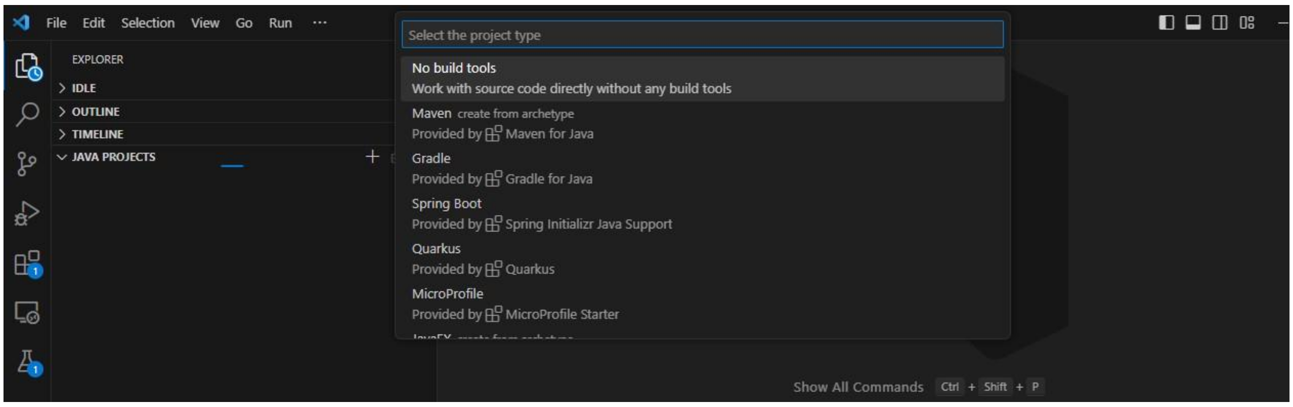
Open Visual Studio Code

Select Extensions Tab and type 'Gradle' in search bar

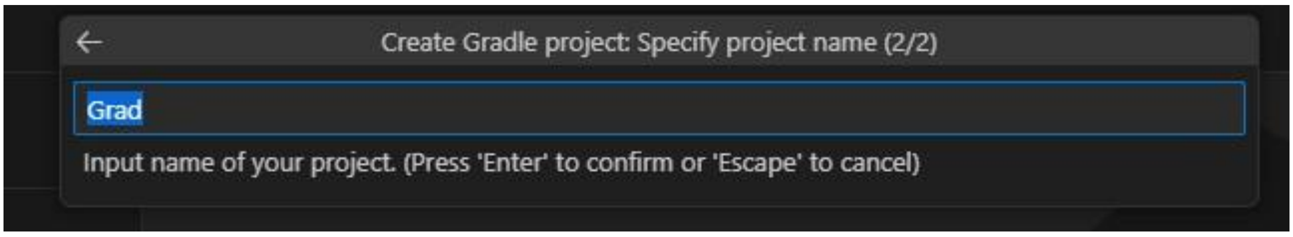
List of recommendations appear choose the 'Gradle Extension pack'

Press <Install> to your VS-Code

Now Go to JAVA PROJECTS Tab to verify Gradle Installation


File Edit Selection View Go Run ···
EXPLORER
IDLE
OUTLINE
TIMELINE
JAVA PROJECTS
Select the project type
No build tools
Work with source code directly without any build tools
Maven create from archetype
Provided by Maven for Java
Gradle
Provided by Gradle for Java
Spring Boot
Provided by Spring Initializr Java Support
Quarkus
Provided by Quarkus
MicroProfile
Provided by MicroProfile Starter
Show All Commands Ctrl + Shift + P

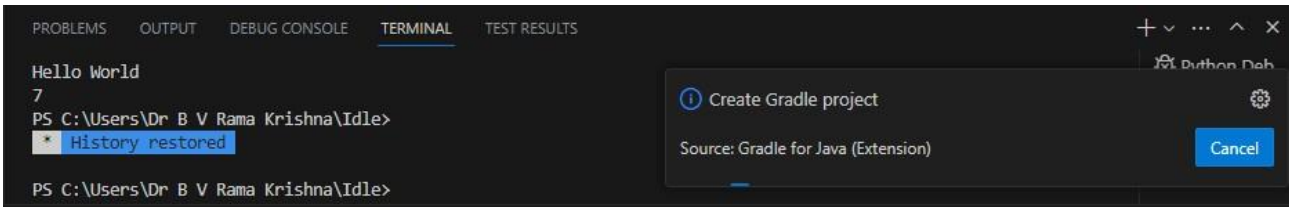
Looks Gradle installed in VS Code press Gradle and choose local directory JAVA Class Path must be correct in windows to work Gradle correctly.


Create Gradle project: Specify project name (2/2)

Grad

Input name of your project. (Press 'Enter' to confirm or 'Escape' to cancel)

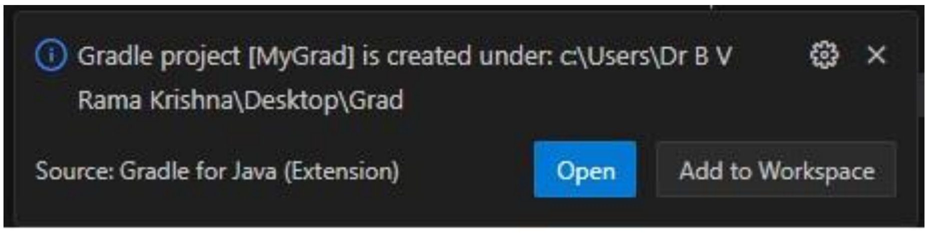
Select ‘Groovy’ build approach and press <Enter>



```
Hello World
7
PS C:\Users\Dr B V Rama Krishna\Idle>
 *  History restored

PS C:\Users\Dr B V Rama Krishna\Idle>
```

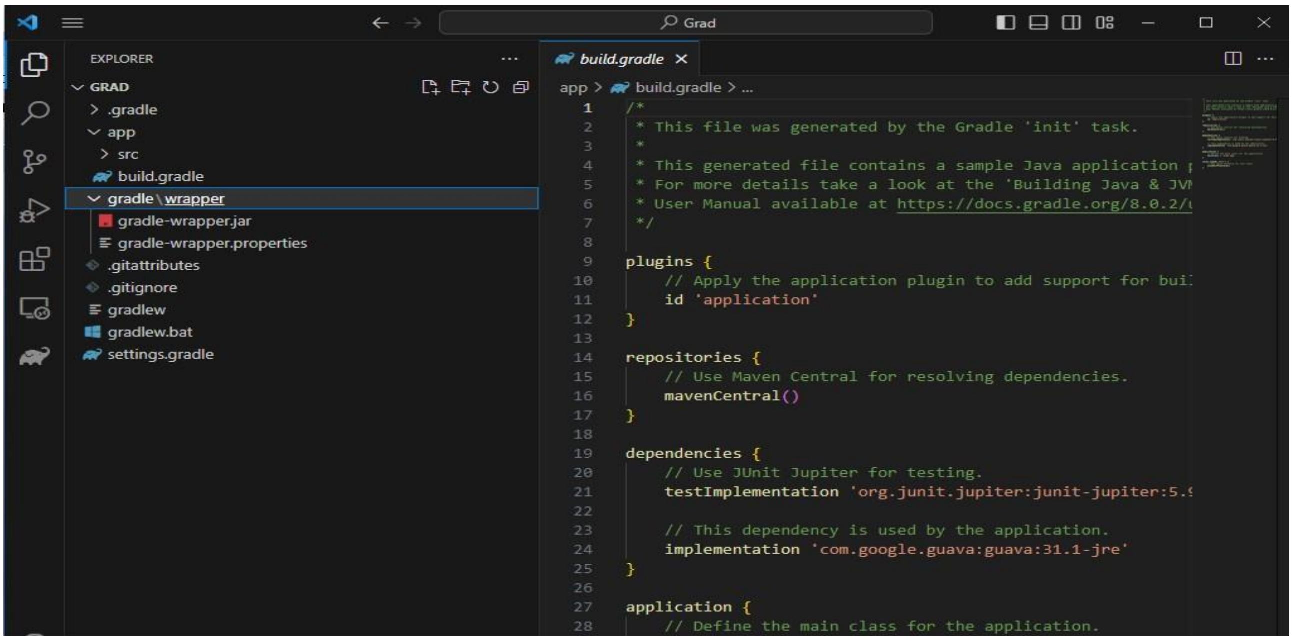
Create Gradle project
Source: Gradle for Java (Extension)

GRADLE Project construction begins for JAVA



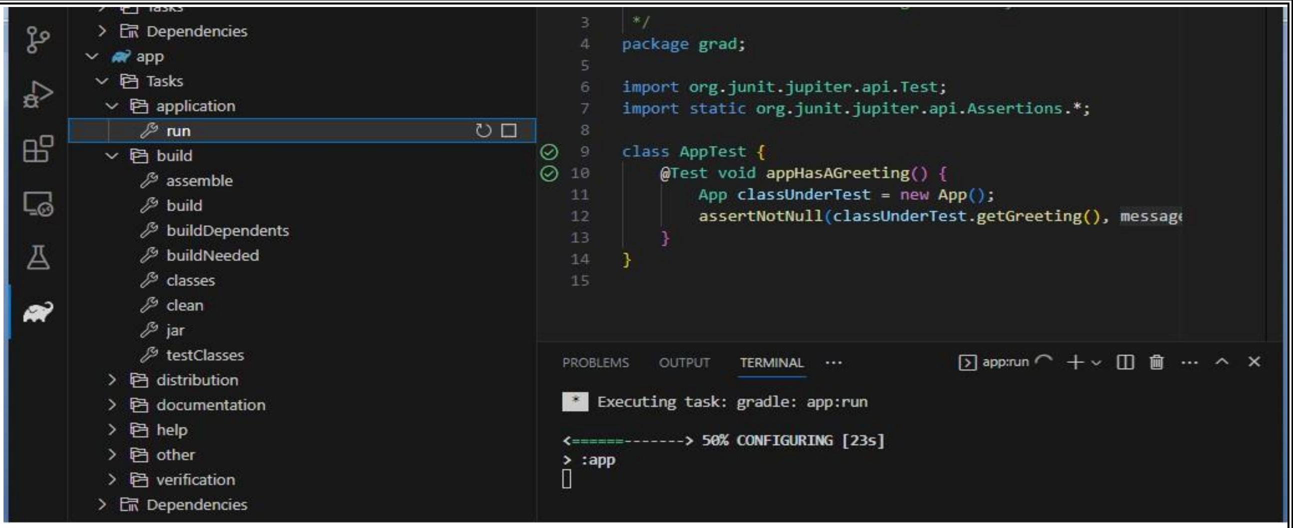
Gradle project [MyGrad] is created under: c:\Users\Dr B V Rama Krishna\Desktop\Grad
Source: Gradle for Java (Extension)

Project created successfully under workspace with dependencies…Press Open button to open workspace Now you can see all dependency , main, src, wrapper and libraries are added to workspace for JRE support artifact construction



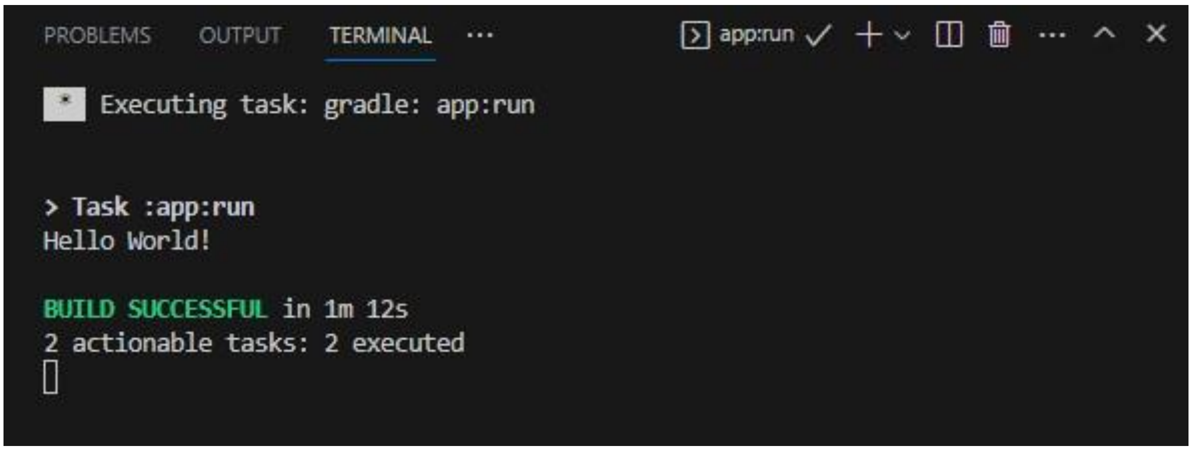
```
/*
 * This file was generated by the Gradle 'init' task.
 *
 * This generated file contains a sample Java application p
 * For more details take a look at the 'Building Java & JVM
 * User Manual available at https://docs.gradle.org/8.0.2/u
 */

plugins {
    // Apply the application plugin to add support for buil
    id 'application'
}

repositories {
    // Use Maven Central for resolving dependencies.
    mavenCentral()
}

dependencies {
    // Use JUnit Jupiter for testing.
    testImplementation 'org.junit.jupiter:junit-jupiter:5.9
    
    // This dependency is used by the application.
    implementation 'com.google.guava:guava:31.1-jre'
}

application {
    // Define the main class for the application.
```

Working with Build in App tab press ‘Run’ to build artifact



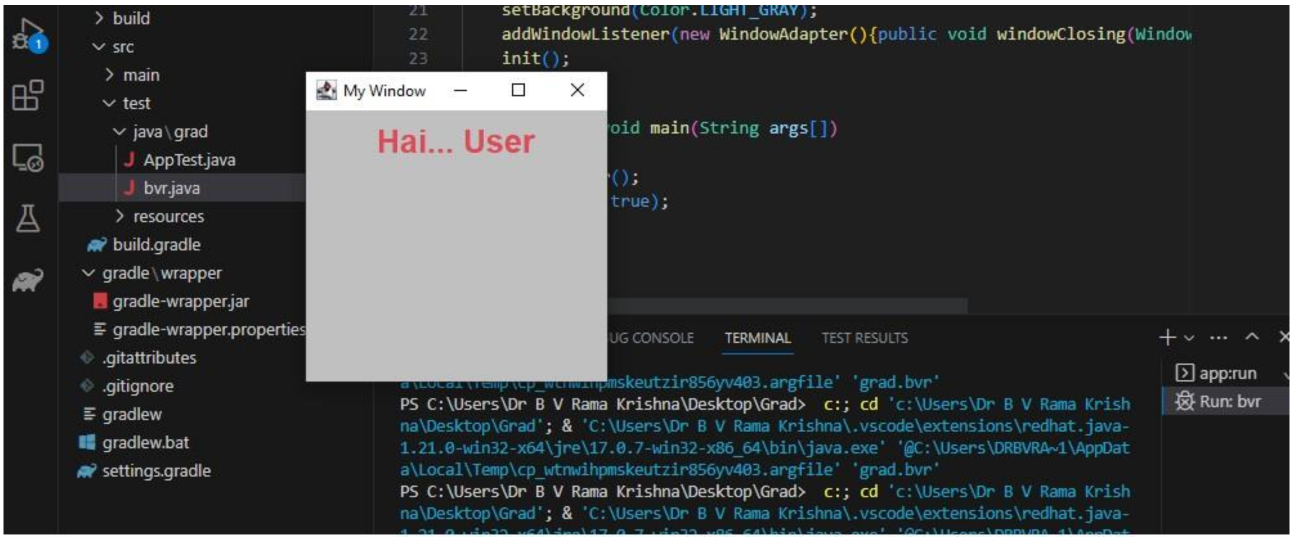
```
*/
package grad;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class AppTest {
    @Test void appHasAGreeting() {
        App classUnderTest = new App();
        assertNotNull(classUnderTest.getGreeting(), messag
    }
}
```

```
Executing task: gradle: app:run

<============-> 50% CONFIGURING [23s]
> :app
```



```
Executing task: gradle: app:run


> Task :app:run
Hello World!

BUILD SUCCESSFUL in 1m 12s
2 actionable tasks: 2 executed
```

GRADLE running another app to wish through Http local server using builded artifacts



```
setBackground(Color.LIGHT_GRAY);
addWindowListener(new WindowAdapter(){public void windowClosing(Window
init();
```

```
a\Local\Temp\cp_wtnwihpmskeutzir856yv403.argfile' 'grad.bvr'
PS C:\Users\Dr B V Rama Krishna\Desktop\Grad>  c:; cd 'c:\Users\Dr B V Rama Krish
na\Desktop\Grad'; & 'C:\Users\Dr B V Rama Krishna\.vscode\extensions\redhat.java-
1.21.0-win32-x64\jre\17.0.7-win32-x86_64\bin\java.exe' '@C:\Users\DRBVRA~1\AppDat
a\Local\Temp\cp_wtnwihpmskeutzir856yv403.argfile' 'grad.bvr'
PS C:\Users\Dr B V Rama Krishna\Desktop\Grad>  c:; cd 'c:\Users\Dr B V Rama Krish
na\Desktop\Grad'; & 'C:\Users\Dr B V Rama Krishna\.vscode\extensions\redhat.java-
```

GRADLE also capable to run environments using built in JAVA dependencies.

EXPERIMENT – 8

Aim: Creating Branches and handling PUSH and PULL requests in GIT-HUB

Description:

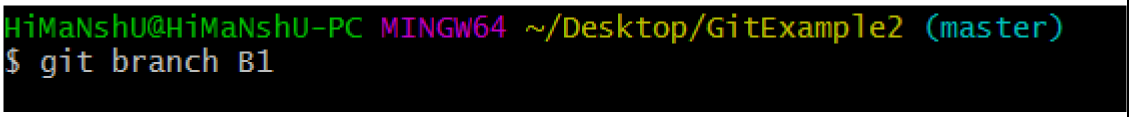
A branch is a version of the repository that diverges from the main working project. It is a feature available in most modern version control systems. A GIT project can have more than one branch. These

branches are a pointer to a snapshot of your changes. The master branch is a default branch in GIT. It is instantiated when first commit made on the project. When you make the first commit, you're given a master branch to the starting commit point.

The git branch command allows you to create, list, rename and delete branches. Many operations on branches are applied by git checkout and git merge command. So, the git branch is tightly integrated with the git checkout and git merge commands.

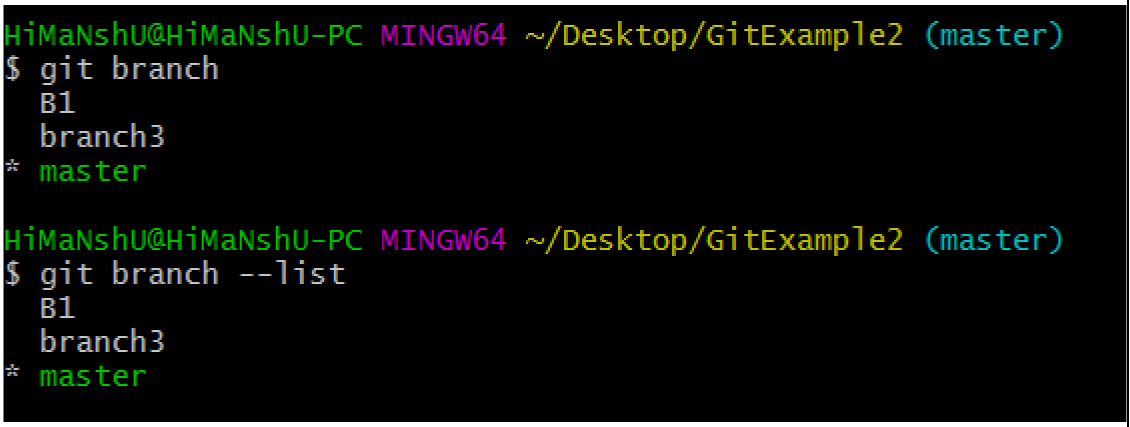
Commands:

1. git branch <branch name> //Creates a branch under master



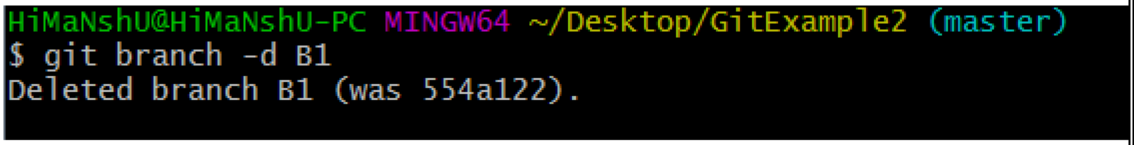
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch B1
```

2. git branch --list (or) git branch // shows all the branches under master



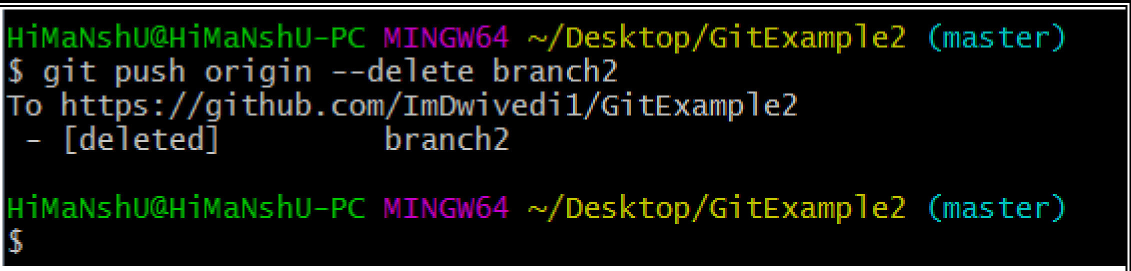
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch
  B1
  branch3
* master

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch --list
  B1
  branch3
* master
```

3. git branch -d<branch name> // used to delete specific branch



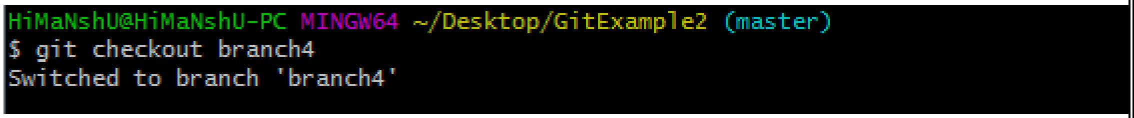
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch -d B1
Deleted branch B1 (was 554a122).
```

4. git push origin -delete <branch name> // to delete a remote branch in GIT-HUB from desktop



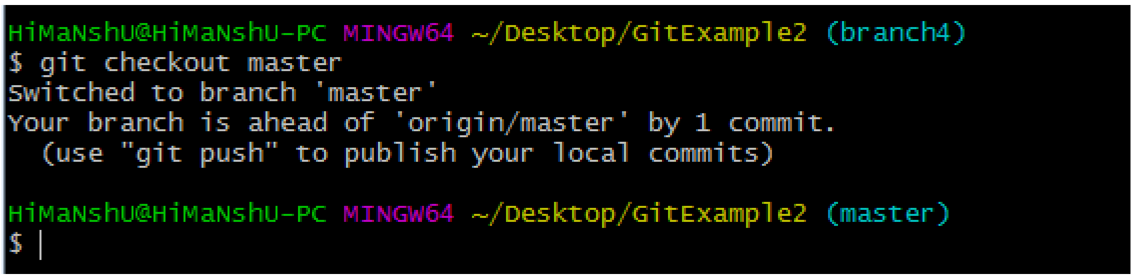
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git push origin --delete branch2
To https://github.com/ImDwivedi1/GitExample2
 - [deleted]         branch2

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$
```

5. git checkout<branch name> // switching between branches from master branch



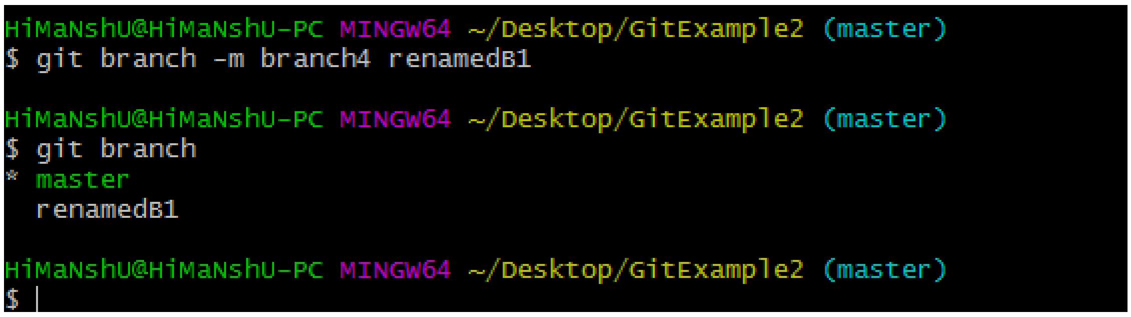
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git checkout branch4
Switched to branch 'branch4'
```

6. git branch -m master // to switch to branch master from any level of branch



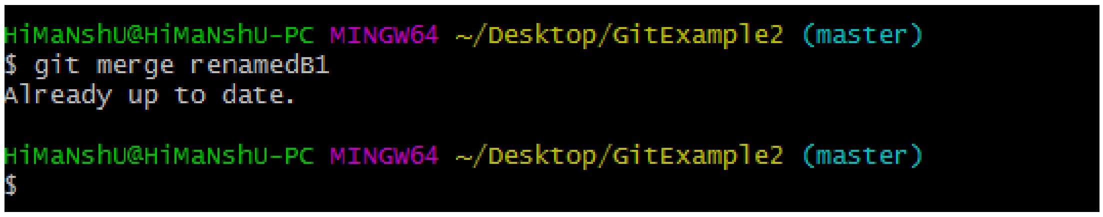
```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (branch4)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ |
```

7. git branch -m <old branch name><new branch name> // renaming the specific branch



```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch -m branch4 renamedB1

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git branch
* master
  renamedB1

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ |
```

8. git merge <branch name> // merging two branches



```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$ git merge renamedB1
Already up to date.

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
$
```

EXPERIMENT – 9

Aim: Jenkins installation and environment creation of CICD pipe line

Description:

Jenkins is an open-source automation server that automates the repetitive technical tasks involved in the continuous integration and delivery of software. Jenkins is Java-based, installed from Ubuntu packages.

Jenkins Environment:

1. Installing Jenkins

   Ubuntu default Jenkins package is always behind latest version so install Jenkins from project- maintained packages.

2. create first repository to download Jenkins package

3. $ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key |sudo gpg --dearmor -o /usr/share/

   keyrings /jenkins.gpg

4. Now Debian package repository must be initiated and appended to package list

```
$ sudo sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins.gpg]
http://pkg.jenkins.io/debian- stable binary/ > /etc/apt/sources.list.d /jenkins.list'
```

5. Now run following command to place up to date Jenkins version into repository

```
$ sudo apt update

$ sudo apt install Jenkins

 . . . . . . Installing package components
```

6. Starting Jenkins

   Now that Jenkins is installed, start it by using systemctl command

```
$ sudo systemctl start jenkins.service

$ sudo systemctl status jenkins
```

7. After successful installation following Jenkins start progress appear

   Output

```
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2022-04-18 16:07:28 UTC; 2min 3s ago

 Main PID: 88180 (java) Tasks: 42 (limit: 4665) Memory: 1.1G

                                  CPU: 46.997s
```

```
        CGroup: /system.slice/jenkins.service
              └─88180 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war
     --webroot=/var/cache/jenkins/war --httpPort=8080
```

8. Now Jenkins is running time to establish firewall for smooth execution of Jenkins

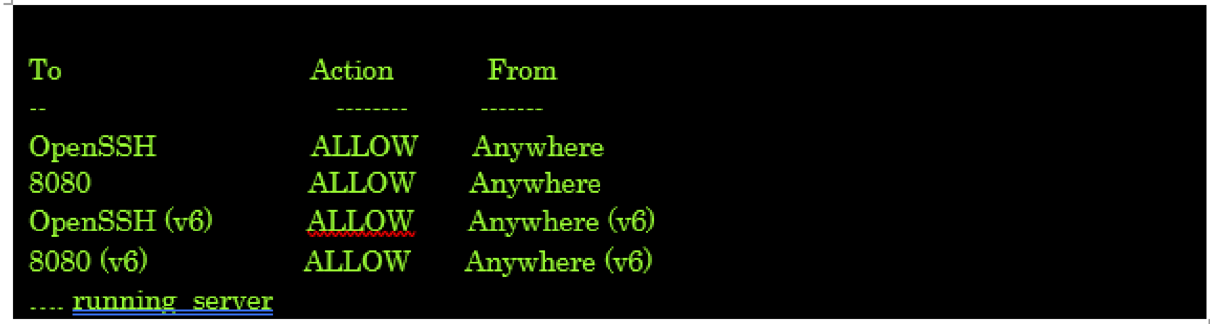
```
$ sudo ufw allow 8080
$ sudo ufw allow OpenSSH
$ sudo ufw enable
```

9. Now 8080 port is ready with Jenkins with following status

```
$ sudo ufw status
```

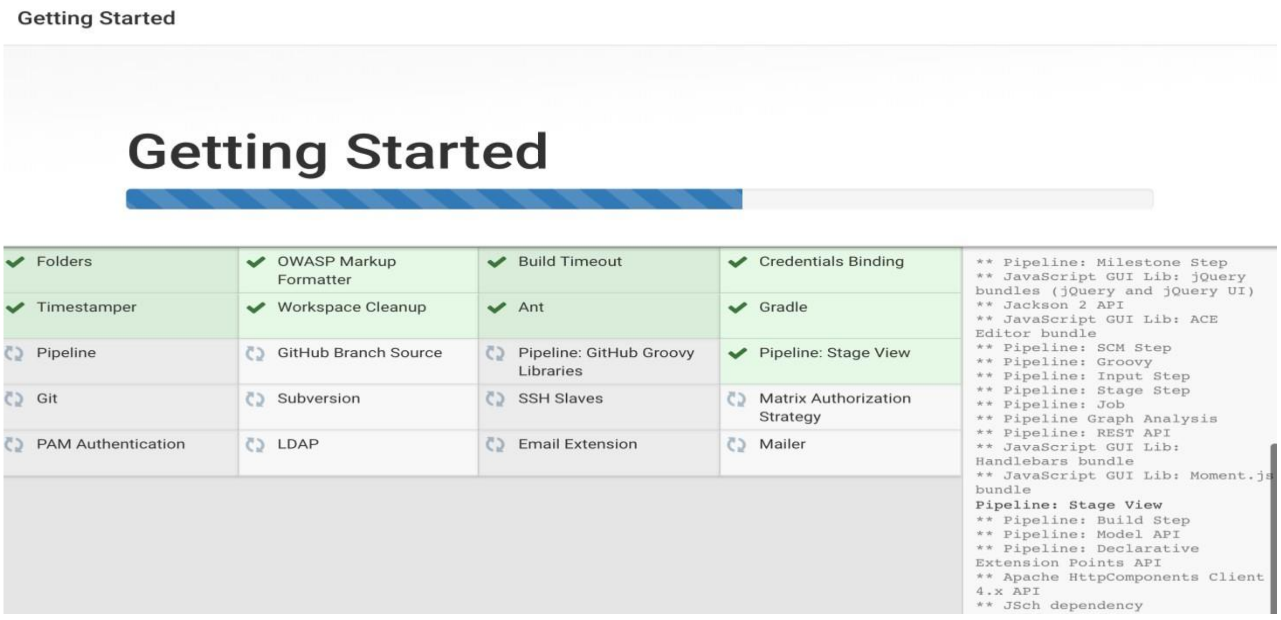
Output

Status: active



```
To                      Action        From
--                        --------      -------
OpenSSH                 ALLOW       Anywhere
8080                    ALLOW       Anywhere
OpenSSH (v6)            ALLOW       Anywhere (v6)
8080 (v6)               ALLOW       Anywhere (v6)
.... running server
```

10. Now Jenkins tuning begins with window opened for installing necessary plug-ins, leave all selected by default as it is click 'OK'



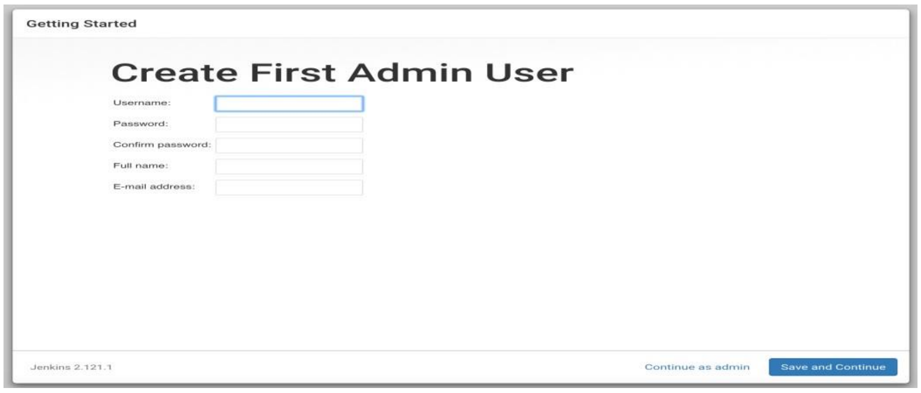
Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	
✓ Timestamper	✓ Workspace Cleanup	✓ Ant	✓ Gradle	
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	✓ Pipeline: Stage View	
Git	Subversion	SSH Slaves	Matrix Authorization Strategy	
PAM Authentication	LDAP	Email Extension	Mailer	

```
** Pipeline: Milestone Step
** JavaScript GUI Lib: jQuery
bundles (jQuery and jQuery UI)
** Jackson 2 API
** JavaScript GUI Lib: ACE
Editor bundle
** Pipeline: SCM Step
** Pipeline: Groovy
** Pipeline: Input Step
** Pipeline: Stage Step
** Pipeline: Job
** Pipeline Graph Analysis
** Pipeline: REST API
** JavaScript GUI Lib:
Handlebars bundle
** JavaScript GUI Lib: Moment.js
bundle
Pipeline: Stage View
** Pipeline: Build Step
** Pipeline: Model API
** Pipeline: Declarative
Extension Points API
** Apache HttpComponents Client
4.x API
** JSch dependency
```

11. Create user credentials as Jenkins Admin



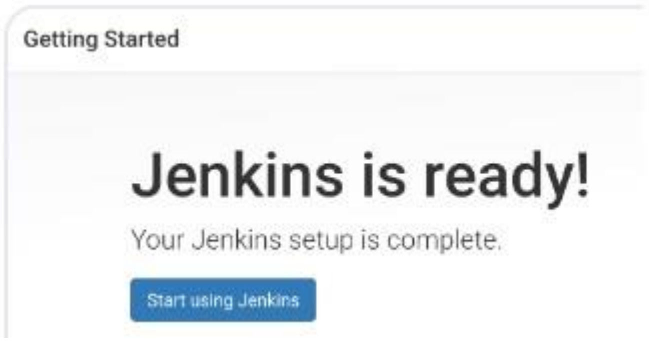
Getting Started

Create First Admin User

Username:
Password:
Confirm password:
Full name:
E-mail address:

Jenkins 2.121.1

12. In instance configuration type Jenkins URL with your local 8080 port address

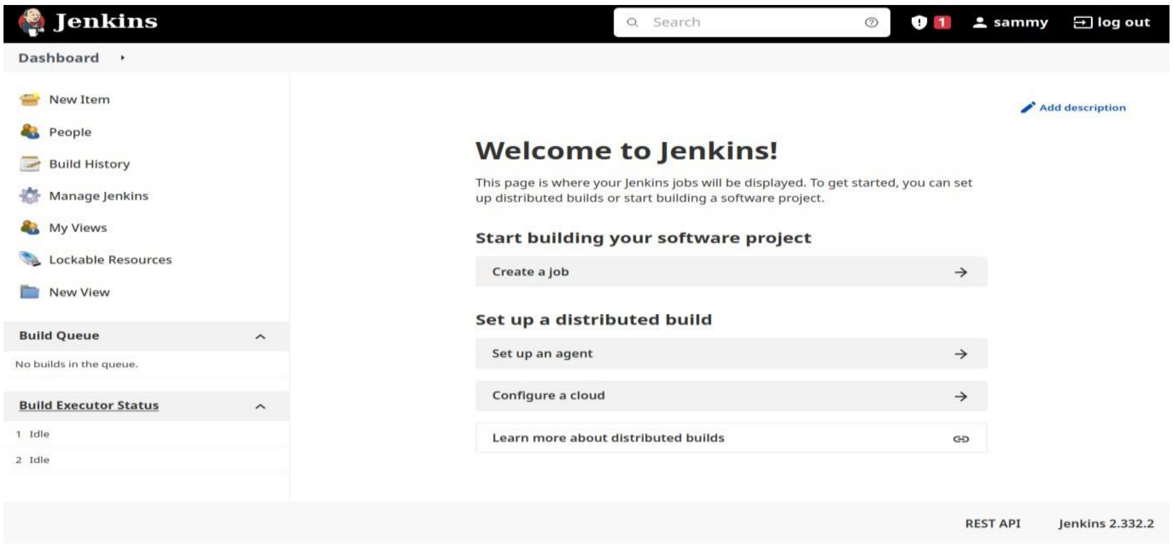


Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

13. Click Start using Jenkins



Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

REST API Jenkins 2.332.2

EXPERIMENT – 10

Nexus Repository

Aim: To implement Nexus repository for storing artifacts

Procedure:

Installation:

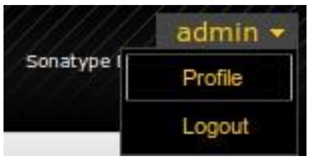
- Extract nexus.TGZ zipped file from web source
- Create NEXUS_FOLDER in your local machine
- Place the unzipped content to NEXUS_FOLDER

Initializing:

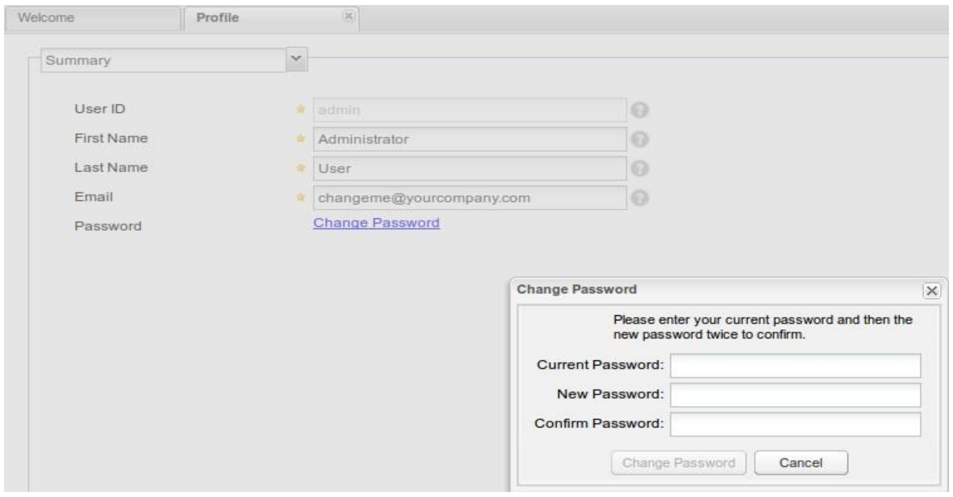
- cd /NEXUS_FOLDER/nexus/bin
- Give command nexus start // Nexus repository initialized on your machine
- If you want to stop give command < nexus stop >

Managing:

- Once Nexus manager started access its web interface
- Default user is [admin] with password [admin123]
- Click the [LOGIN] button on top corner of web page



- After login you can configure NEXUS using [Profile] option in menu
- Finally after setting and tuning user credentials re-login the Nexus

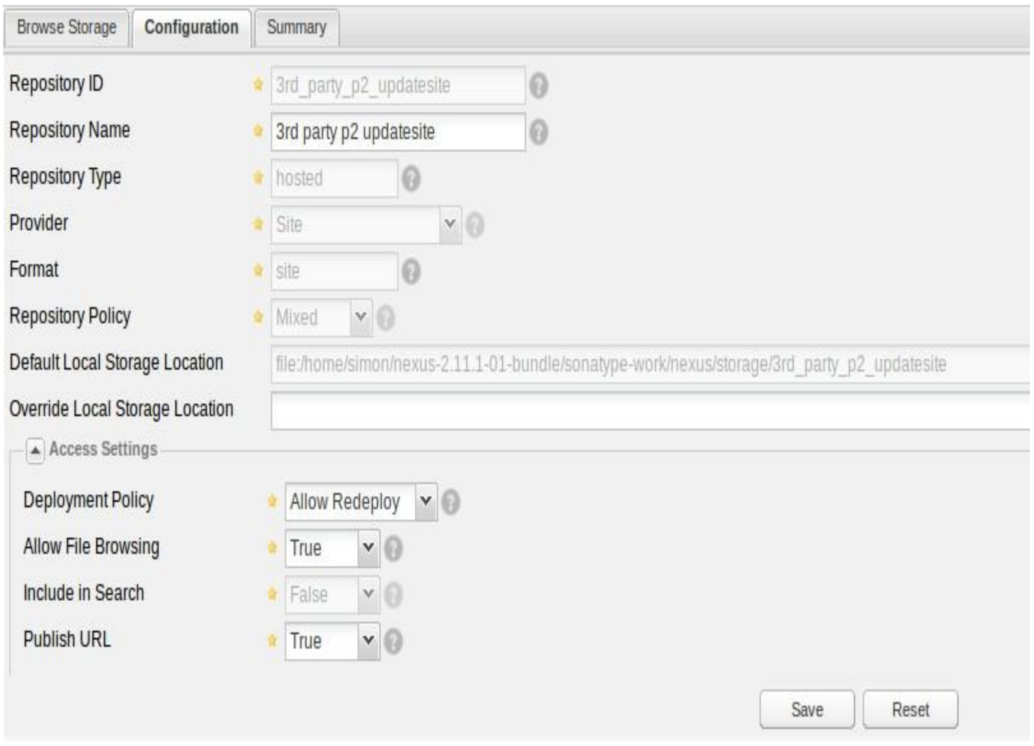


User ID admin
First Name Administrator
Last Name User
Email changeme@yourcompany.com
Password Change Password

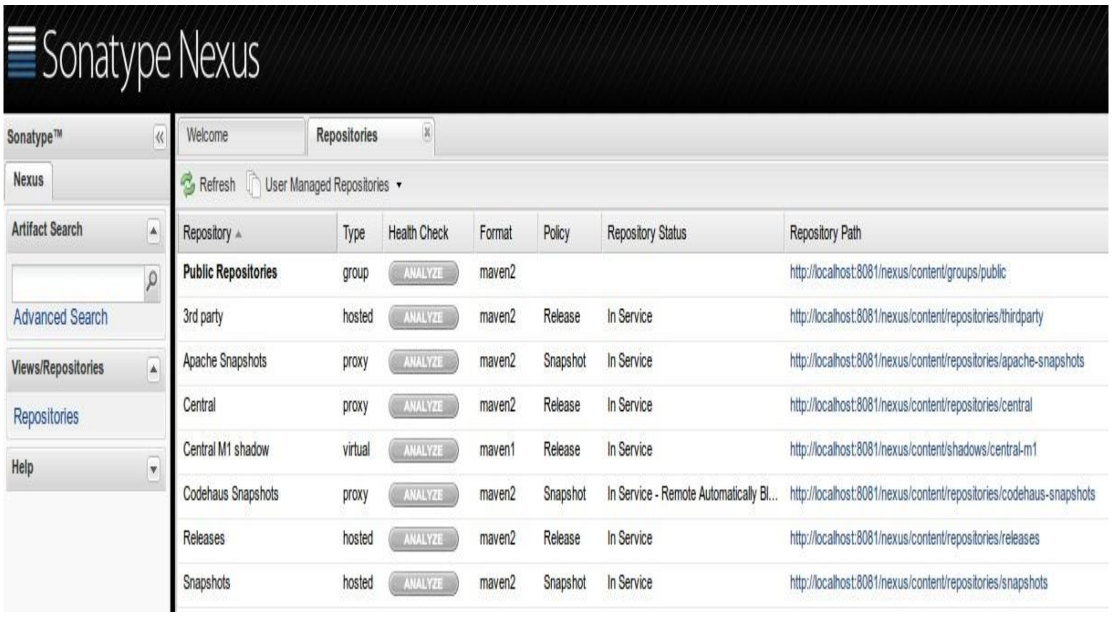
Change Password
Please enter your current password and then the new password twice to confirm.
Current Password:
New Password:
Confirm Password:

Repository:

- Create your repository click Repositories→Add→Host repository
- Configure page opens configure repository carefully
- Root repository created now



Repository ID	3rd_party_p2_updatesite
Repository Name	3rd party p2 updatesite
Repository Type	hosted
Provider	Site
Format	site
Repository Policy	Mixed
Default Local Storage Location	file:/home/simon/nexus-2.11.1-01-bundle/sonatype-work/nexus/storage/3rd_party_p2_updatesite
Override Local Storage Location	
Deployment Policy	Allow Redeploy
Allow File Browsing	True
Include in Search	False
Publish URL	True

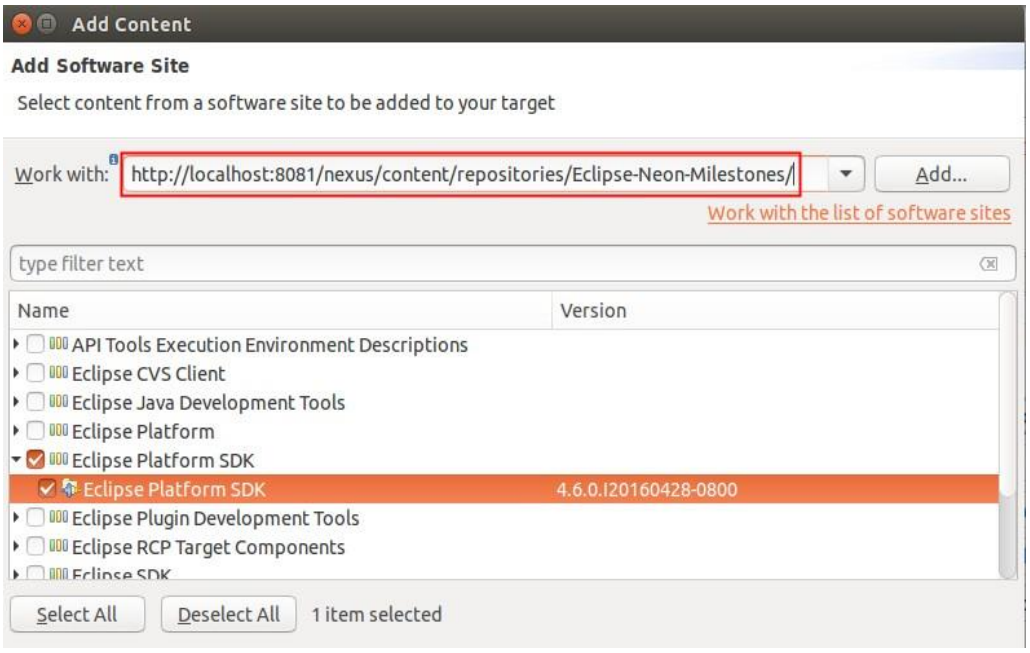


Repository	Type	Health Check	Format	Policy	Repository Status	Repository Path
Public Repositories	group	ANALYZE	maven2			http://localhost:8081/nexus/content/groups/public
3rd party	hosted	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/thirdparty
Apache Snapshots	proxy	ANALYZE	maven2	Snapshot	In Service	http://localhost:8081/nexus/content/repositories/apache-snapshots
Central	proxy	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/central
Central M1 shadow	virtual	ANALYZE	maven1	Release	In Service	http://localhost:8081/nexus/content/shadows/central-m1
Codehaus Snapshots	proxy	ANALYZE	maven2	Snapshot	In Service - Remote Automatically Bl...	http://localhost:8081/nexus/content/repositories/codehaus-snapshots
Releases	hosted	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/releases
Snapshots	hosted	ANALYZE	maven2	Snapshot	In Service	http://localhost:8081/nexus/content/repositories/snapshots

Proxy:

- Create Proxy repository for server side data handling
- First you need to install plug-in " Nexus P2"
- Go to .bin/nexus type command < install bundle.zip>
- After installation completes restart nexus <nexus stop> and <nexus start>
- open Nexus web page on 8081 port
- Watch the Repositories menu you can see newly "Proxy Repository" option added

- Check proxy repository with Git-Hub and Maven interfaces inclusion
- Go to http://localhost:8081/nexus/content/repositories/Eclipse-Neon-Milestones
- Put Check √ mark to option platform SDK inclusion



Add Content

Add Software Site

Select content from a software site to be added to your target

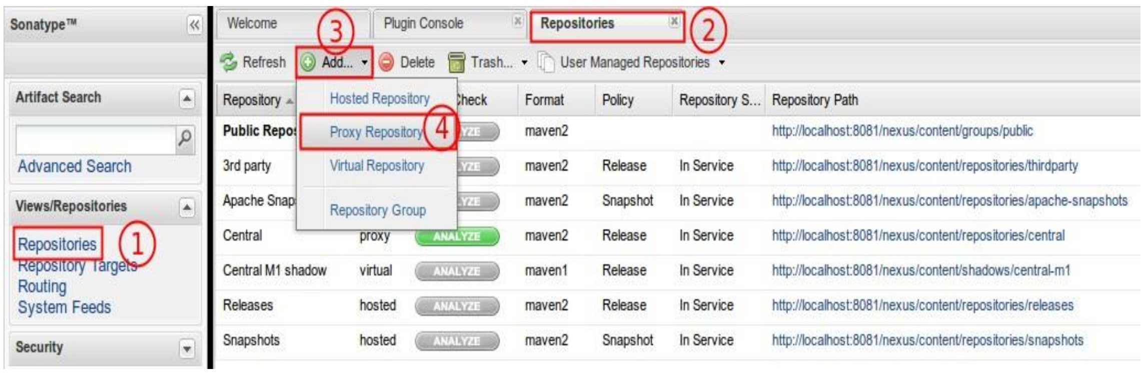
Work with: http://localhost:8081/nexus/content/repositories/Eclipse-Neon-Milestones/

Work with the list of software sites

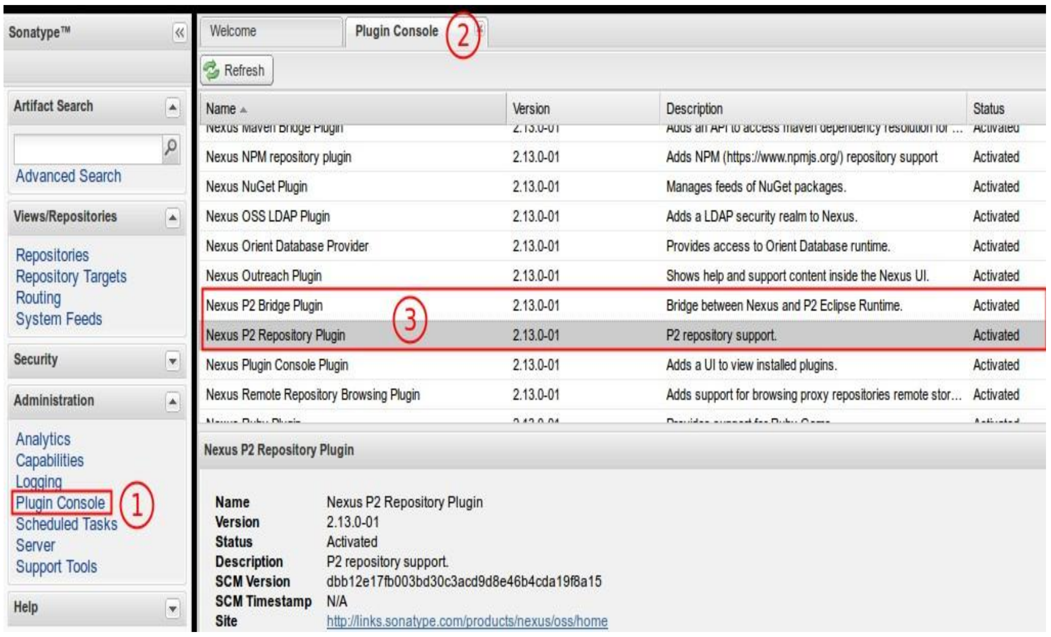
type filter text

Name	Version
☐ API Tools Execution Environment Descriptions	
☐ Eclipse CVS Client	
☐ Eclipse Java Development Tools	
☐ Eclipse Platform	
✓ Eclipse Platform SDK	
✓ Eclipse Platform SDK	4.6.0.I20160428-0800
☐ Eclipse Plugin Development Tools	
☐ Eclipse RCP Target Components	
☐ Eclipse SDK	

1 item selected



Repository		Check	Format	Policy	Repository S...	Repository Path
Public Repo			maven2			http://localhost:8081/nexus/content/groups/public
3rd party			maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/thirdparty
Apache Snap			maven2	Snapshot	In Service	http://localhost:8081/nexus/content/repositories/apache-snapshots
Central	proxy	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/central
Central M1 shadow	virtual	ANALYZE	maven1	Release	In Service	http://localhost:8081/nexus/content/shadows/central-m1
Releases	hosted	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/releases
Snapshots	hosted	ANALYZE	maven2	Snapshot	In Service	http://localhost:8081/nexus/content/repositories/snapshots



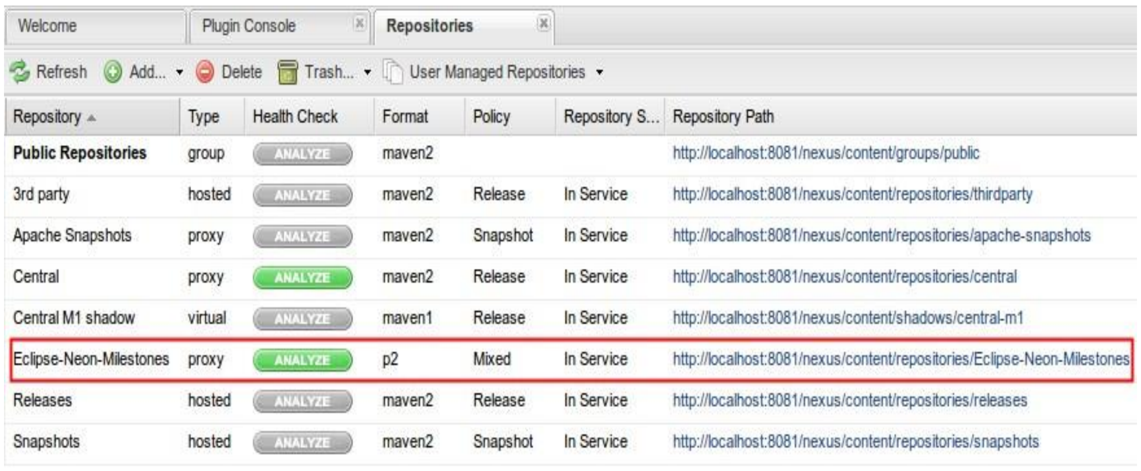
Name	Version	Description	Status
Nexus Maven Bridge Plugin	2.13.0-01	Adds an API to access maven dependency resolution for ...	Activated
Nexus NPM repository plugin	2.13.0-01	Adds NPM (https://www.npmjs.org/) repository support	Activated
Nexus NuGet Plugin	2.13.0-01	Manages feeds of NuGet packages.	Activated
Nexus OSS LDAP Plugin	2.13.0-01	Adds a LDAP security realm to Nexus.	Activated
Nexus Orient Database Provider	2.13.0-01	Provides access to Orient Database runtime.	Activated
Nexus Outreach Plugin	2.13.0-01	Shows help and support content inside the Nexus UI.	Activated
Nexus P2 Bridge Plugin	2.13.0-01	Bridge between Nexus and P2 Eclipse Runtime.	Activated
Nexus P2 Repository Plugin	2.13.0-01	P2 repository support.	Activated
Nexus Plugin Console Plugin	2.13.0-01	Adds a UI to view installed plugins.	Activated
Nexus Remote Repository Browsing Plugin	2.13.0-01	Adds support for browsing proxy repositories remote stor...	Activated

Nexus P2 Repository Plugin

Name	Nexus P2 Repository Plugin
Version	2.13.0-01
Status	Activated
Description	P2 repository support.
SCM Version	dbb12e17fb003bd30c3acd9d8e46b4cda19f8a15
SCM Timestamp	N/A
Site	http://links.sonatype.com/products/nexus/oss/home

Ready:

- Now Nexus shows repositories updated with MAVEN, GITHUB
- All stages are appeared with time stamps



Repository	Type	Health Check	Format	Policy	Repository S...	Repository Path
Public Repositories	group	ANALYZE	maven2			http://localhost:8081/nexus/content/groups/public
3rd party	hosted	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/thirdparty
Apache Snapshots	proxy	ANALYZE	maven2	Snapshot	In Service	http://localhost:8081/nexus/content/repositories/apache-snapshots
Central	proxy	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/central
Central M1 shadow	virtual	ANALYZE	maven1	Release	In Service	http://localhost:8081/nexus/content/shadows/central-m1
Eclipse-Neon-Milestones	proxy	ANALYZE	p2	Mixed	In Service	http://localhost:8081/nexus/content/repositories/Eclipse-Neon-Milestones
Releases	hosted	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/releases
Snapshots	hosted	ANALYZE	maven2	Snapshot	In Service	http://localhost:8081/nexus/content/repositories/snapshots

EXPERIMENT – 11

Aim: To implement Sonar Cube server on windows platform

Procedure:

Description:

Sonar-Qube is a Code Quality Assurance tool that collects and analyzes source code, and provides reports for the code quality of your project.

Sonar-Qube Architecture can be classified in four components

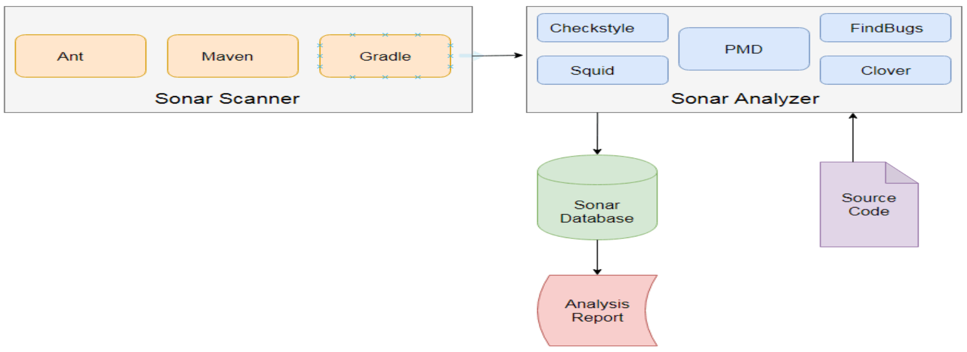
1. Sonar Scanner - Previously known as sonar runner. Ideally you need to use any of

   the built-in tools such as ant, Maven or Gradle that invokes the Sonar Scanner to fetch the source code. Sonar Qube cube supports plug-ins like SVN and some of the other version control system

2. Source Code - The source code is the code written by developer or manager. The code is then pushed to repository.

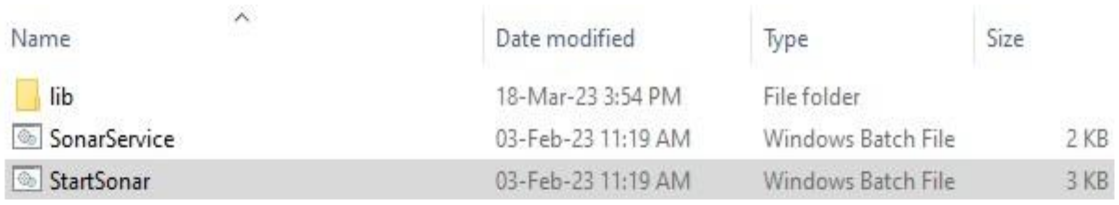
3. Sonar Analyzer - Sonar analyzer takes the source code you would like to analyze and go through all the code and gives you technical problems

4. Sonar Qube Database - The reports generated by Sonar Cube will be sent to the database. There is a caching server which is used to hold the temporary reports in the cache, Sonar Qube will have a default database You can integrate your own database our to your Sonar Qube Sonar Qube supports various databases like a ms SQL Server, Oracle etc.



Requirements: Require Java 17 Version minimum, 1GB of free RAM in system

Installation: 1. Go to C:\sonarqube-9.9.0.65466\bin\windows-x86-64 path, there is Start Sonar window batch file right click on run as administrator.



Name	Date modified	Type	Size
lib	18-Mar-23 3:54 PM	File folder	
SonarService	03-Feb-23 11:19 AM	Windows Batch File	2 KB
StartSonar	03-Feb-23 11:19 AM	Windows Batch File	3 KB

2. Set the java path environmental variable

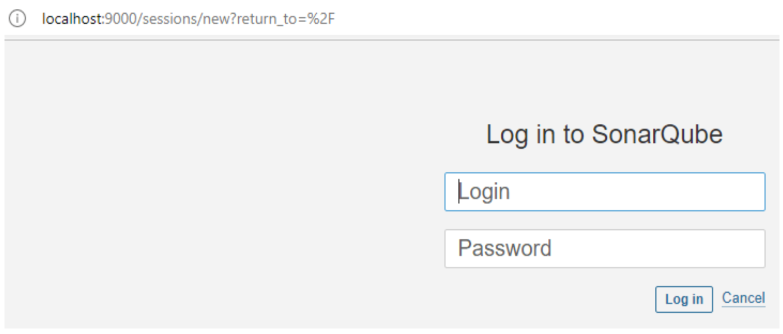
   setx SONAR_JAVA_PATH "C:\Program Files\java_home\bin\java.exe"

3. Open command prompt Navigate to below path and run StartSonar.bat file

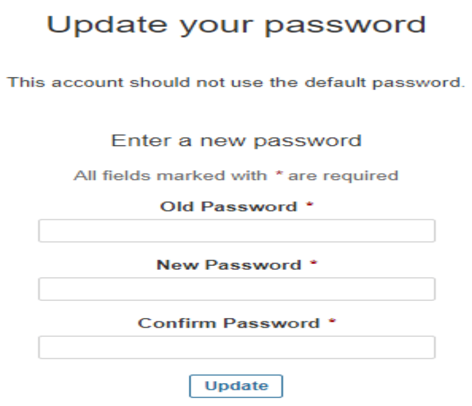
   cd C:\sonarqube-9.9.0.65466\bin\windows-x86-64

   <SONARQUBE_HOME>/bin/windows-x86-64/StartSonar. Bat

4. Now Access Sonar Qube from Windows using localhost:9000



5. Sonar Qube default username and password, login: admin and password: admin and click on login button. Now you can update password/username.


Update your password

This account should not use the default password.

Enter a new password

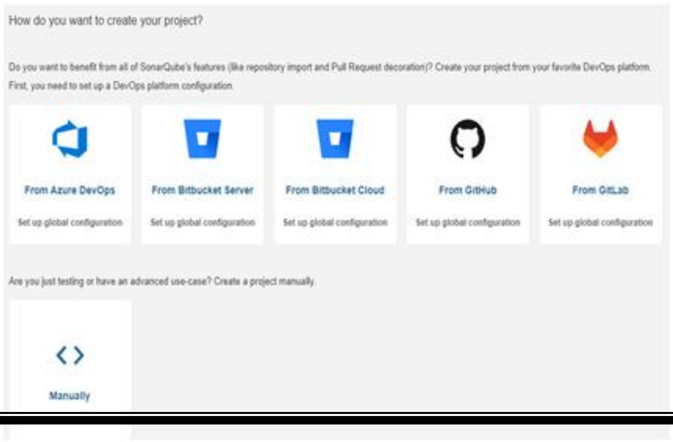
All fields marked with * are required

Old Password *

New Password *

Confirm Password *

6. Once set navigate to Sonar Qube Dashboard


How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.

First, you need to set up a DevOps platform configuration.

From Azure DevOps
Set up global configuration
From Bitbucket Server
Set up global configuration
From Bitbucket Cloud
Set up global configuration
From GitHub
Set up global configuration
From GitLab
Set up global configuration

Are you just testing or have an advanced use-case? Create a project manually.

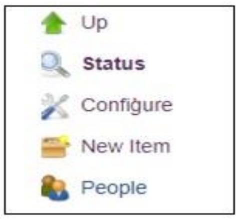
7.

Managing: Login to Jenkins configuration domain with

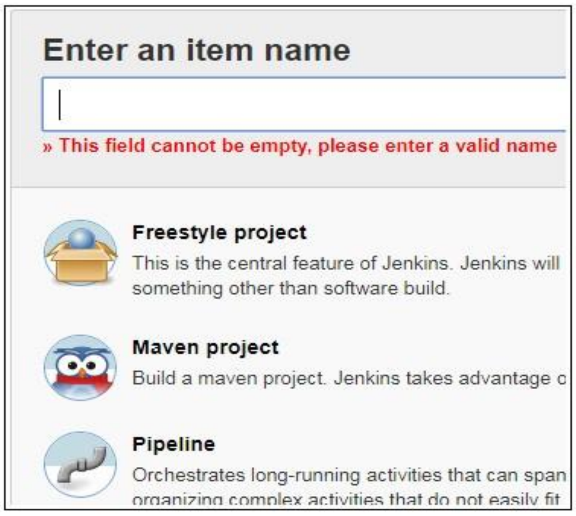
https://jenkins.domain.com

Choose specific profile

Configure Sonar job



Click the pipeline option from items panel then select 'Maven'



Enter an item name

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

Freestyle project
This is the central feature of Jenkins. Jenkins will
something other than software build.

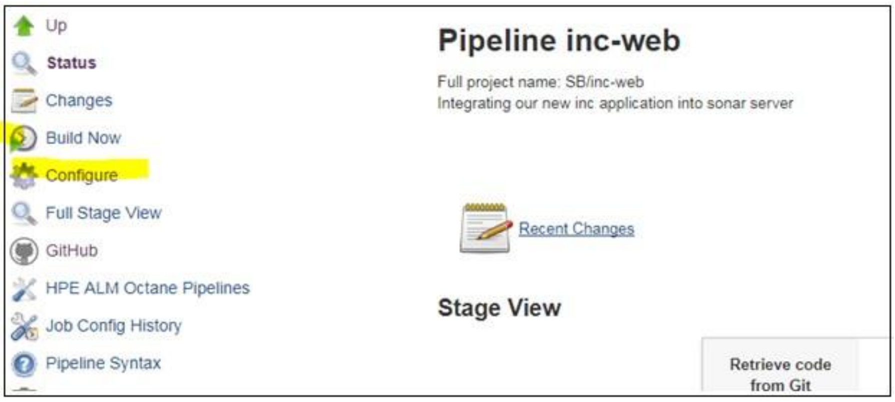
Maven project
Build a maven project. Jenkins takes advantage o

Pipeline
Orchestrates long-running activities that can span
organizing complex activities that do not easily fit

Configure 'Changes' option to Enable

For Build option choose 'Jenkins'

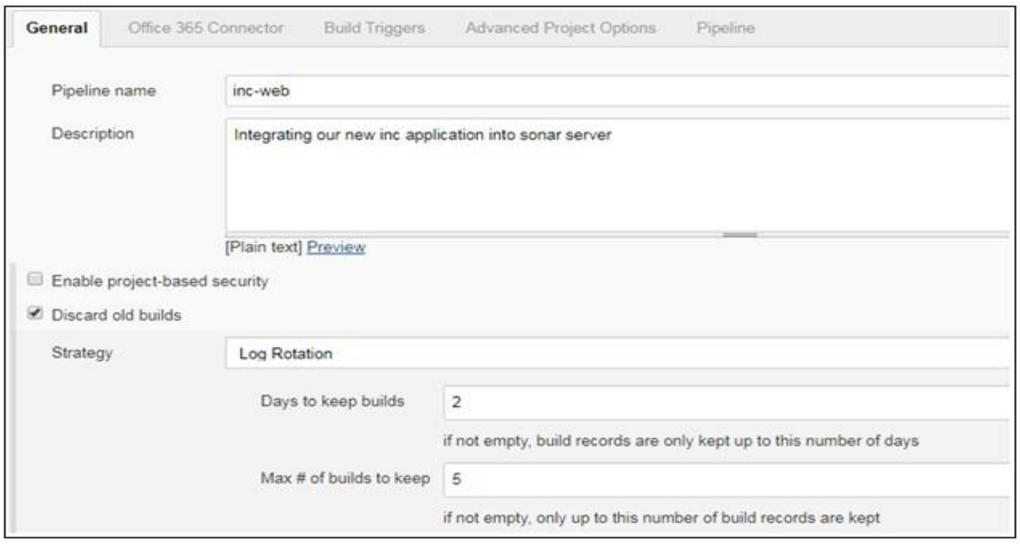
Configure source code read from 'GitHub' or 'Git/SVN'



Pipeline inc-web

Full project name: SB/inc-web
Integrating our new inc application into sonar server

Now tune up Sonar Qube integrations as



Pipeline name: inc-web

Description: Integrating our new inc application into sonar server

[Plain text] Preview

☐ Enable project-based security

☑ Discard old builds

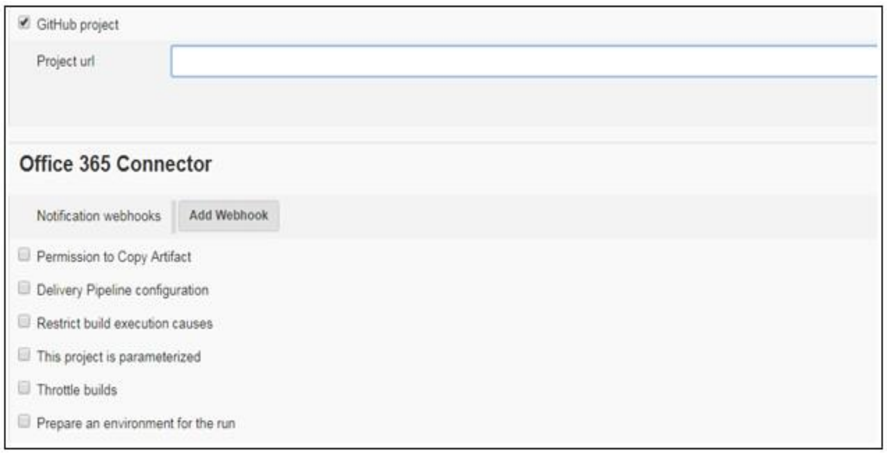
Strategy: Log Rotation

Days to keep builds: 2

if not empty, build records are only kept up to this number of days

Max # of builds to keep: 5

if not empty, only up to this number of build records are kept


☑ GitHub project

Project url:

Office 365 Connector

Notification webhooks

☐ Permission to Copy Artifact

☐ Delivery Pipeline configuration

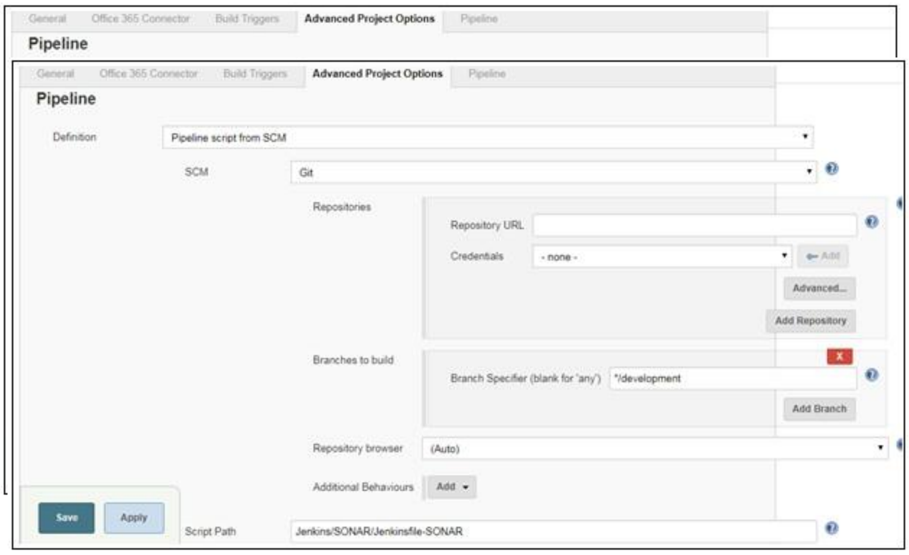
☐ Restrict build execution causes

☐ This project is parameterized

☐ Throttle builds

☐ Prepare an environment for the run

Select the pipeline



Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL:

Credentials: - none -

Branches to build

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

Repository browser: (Auto)

Additional Behaviours

Script Path: Jenkins/SONAR/Jenkinsfile-SONAR

Finally add plugin code and prepare Jenkin file code

[java]try {

stage("Building SONAR …") {

sh './Mavenw clean sonarqube'

}

} catch (e) {emailext attachLog: true, body: ‘See attached log’, subject: ‘BUSINESS

Build Failure’, to: ‘abc@gmail.com’

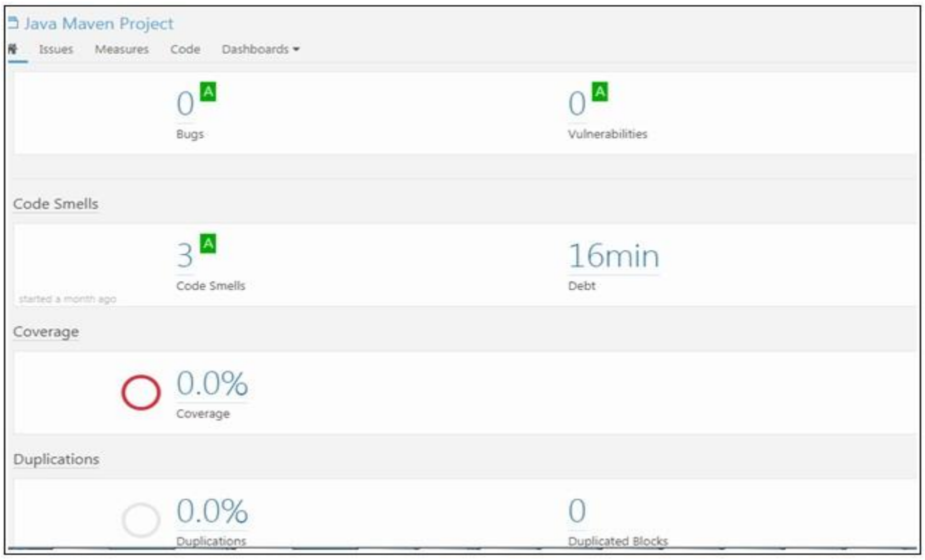
step([$class: ‘WsCleanup’])

return

}

[/java]

Now run the pipeline results the code execution from Sonar Qube as



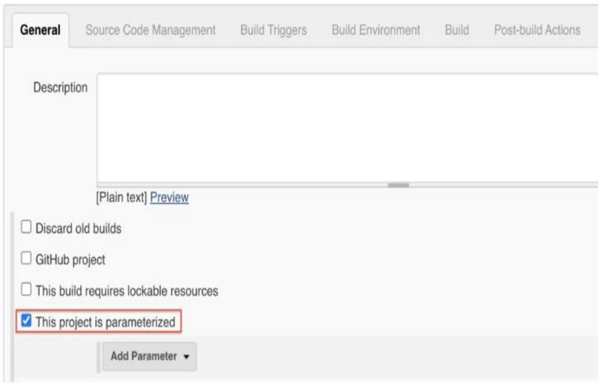
EXPERIMENT – 12

AIM: Implementing parameterized Pipeline project using JENKINS

Description:

A parameterized build allows us to pass data into Jenkins Jobs. Any Jenkin job or pipeline can be parameterized.

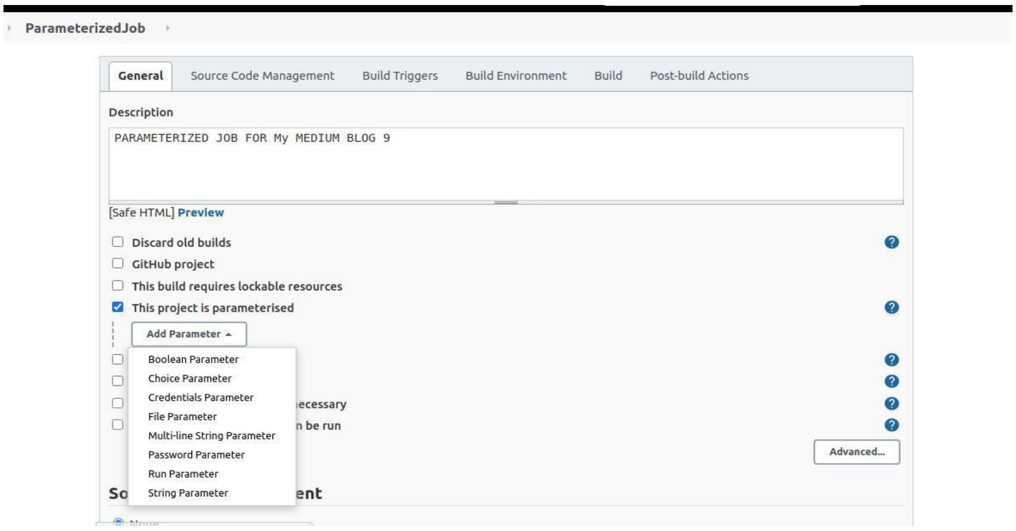
Implementation: open the Jenkin Manager



Description

[Plain text] Preview

☐ Discard old builds
☐ GitHub project
☐ This build requires lockable resources
☑ This project is parameterized



Description

```
PARAMETERIZED JOB FOR My MEDIUM BLOG 9
```

[Safe HTML] Preview

☐ Discard old builds
☐ GitHub project
☐ This build requires lockable resources
☑ This project is parameterised

select new project and check the box (This project is parameterized) Click the Add Parameter button

Each parameter can be defined with characteristics {Type, Name, Default Value, Description}

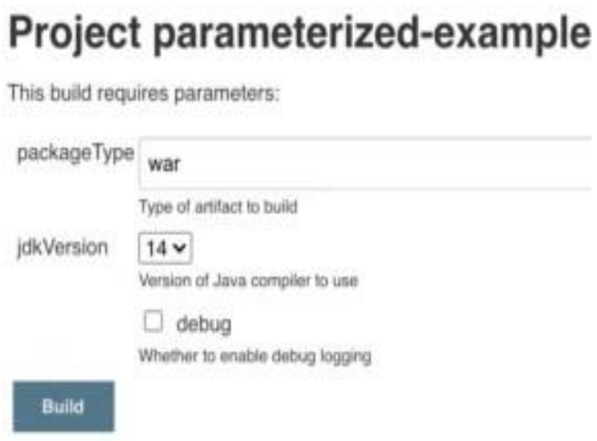
The Jenkin supported project parameters are String, Choice, credentials, File, Multi-line string, Password and Run

Defining build in JENKIN

```
${package Type}
%package Type%
pipeline{
agent any
stages{
   stage('Build'){
                  when{
                 expression{ params.jdkVersion == "14"}
                      }
                    }
          }
```

```
pipeline{agent any
 stages{
   stage('Build'){
             steps {
                    echo "${packagingType}"
                   }
                      }
  }
                            }
```

Finally we have to build parameters based jobs using 'Build with Parameters' link



Project parameterized-example

This build requires parameters:

packageType: war
Type of artifact to build

jdkVersion: 14
Version of Java compiler to use

☐ debug
Whether to enable debug logging