Jenkins and Maven

Jenkins

- Automation server written in Java which helps to automate the build process with continuous integration and facilitating technical aspects of continuous delivery.
- It can execute Apache Ant, Apache Maven based projects as well as arbitrary shell scripts and Windows batch commands.
- Builds can be triggered by various means such as commit or scheduling

Maven

- Maven was originally started to simplify the build processes
- It can be used for building and managing any Java-based project.
- Maven’s Objectives
  - Making the build process easy
  - Providing a uniform build system
  - Providing quality project information
  - Providing guidelines for best practices development
  - Allowing transparent migration to new features

Jenkins Installation

- Jenkins can be installed stand-alone or one can use a docker container that has devopstraining installation.
- We will use Docker based system
- devopstraining
    - Provides a good user experience of Jenkins to model and present the process of software delivery

Maven Installation(if not already in docker image)

- In terminal,
  - docker container ls -a
  - docker exec –it <<container id>> /bin/bash
  - Go to lib
  - Run **apk add maven**
  - mvn --version

Tomcat Installation(if not already in docker image)

- In terminal,
  - docker container ls -a
  - docker exec –it <<container id>> /bin/bash
    - wget http://mirrors.estointernet.in/apache/tomcat/tomcat-9/v9.0.20/bin/apache-tomcat-9.0.20.zip -O /tmp/apache-tomcat-9.0.20.zip

    Note: if above tomcat version version does not work, use another version from http://mirrors.estointernet.in/apache/tomcat/
  - Go to /tmp
    - unzip apache-tomcat-9.0.20.zip
  - Go to apache-tomcat-9.0.20/bin
  - Give access by below command:
    - chmod +x *.sh

Tomcat Installation(Cont…)

- In terminal,
  - Edit server.xml under conf:
    - vi server.xml
    - edit port 8080 to 8089
    - press esc > press shift key+colon > write wq > press enter
  - Go to /tmp/apache-tomcat-9.0.20/bin
  - To start server:
    - sh startup.sh
  - To stop server:
    - sh shutdown.sh

Jenkins Hands-on

- In next set of slides we will

1. Download a Docker image which has Jenkins, maven, tomcat installed
2. Create a pipeline to do following tasks in sequence
    1. Download sources from a GitHub repository and use maven to build it
    2. Perform PMD
    3. Code Compile
    4. Static Code Analysis
    5. Perform Static Code Analysis
    6. Run Junit based tests on it
    7. Perform Code coverage
    8. Make build
    9. Tomcat server up
    10. Deploy war on tomcat
    11. Run the system test

Docker Image with Preinstalled Jenkins

Run the following command

Note: Make sure you have created blueoceantest folder is created under C:\Users\<username>\

- Windows
  - *docker run --rm -u root -p 8089:8089 -p 8080:8080 -v jenkins-data:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -v c:\\"%HOMEPATH%"\blueoceantest:/home umangsaltuniv/devopstraining*

  (Note: command for jenkinsci/blueocean is mentioned below)

  - docker run --rm -u root -p *8089:8089* -p 8080:8080 -v jenkins-data:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -v c:\\"%HOMEPATH%"\blueoceantest:/home jenkinsci/blueocean

- Linux/Mac
  - *docker run --rm -u root -p 8089:8089 -p 8080:8080 -v jenkins-data:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -v $HOME:/home umangsaltuniv/devopstraining*

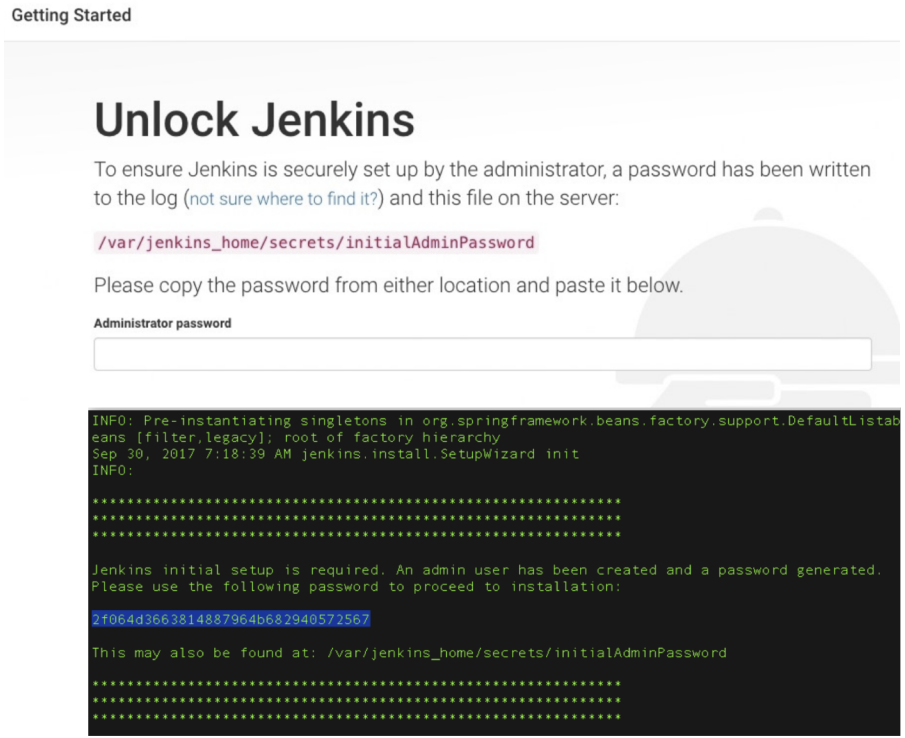
Understanding the Command

1. Automatically removes the Docker container when it is shut down
2. Maps port 8089 on the host machine to port 8089 of the **umangsaltuniv/devopstraining** container for tomcat server
3. Maps port 8080 on the host machine to port 8080 of the umangsaltuniv/devopstraining container for jenkins server
4. Let's not worry about it right now
5. (But highly recommended) Maps the /var/jenkins_home directory in the container to the Docker volume with the name jenkins-data (automatically created if non-existen). Makes Jenkins state to persist each time you restart Jenkins
6. Allows umangsaltuniv/devopstraining container to communicate with the Docker daemon when we use "agent" command in pipeline code
7. Maps the %HOMEPATH% directory on the host machine to the /home directory in the container. Mac has slightly different syntax

```
docker run \
        -u root \
        --rm \ (1)
        -p 8089:8089 \ (2)
         -p 8080:8080 \ (3)
        -v jenkins-data:/var/jenkins_home \ (4)
        -v /var/run/docker.sock:/var/run/docker.sock \ (5)
        -v c:\\"%HOMEPATH%" \ (6)
        blueoceantest:/home umangsaltuniv/devopstraining (7)
```

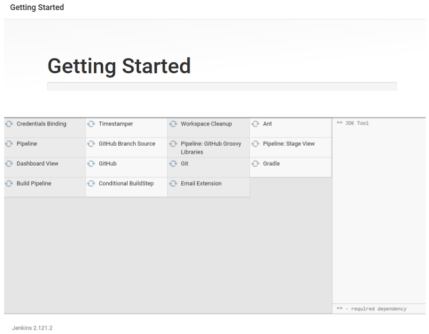
Unlocking Jenkis

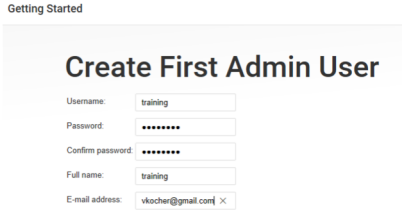
- After the 2 sets of asterisks appear in the terminal/command prompt window, browse to http://localhost:8080 and wait until the Unlock Jenkins page appears
- From the terminal/command prompt window again, copy the automatically-generated alphanumeric password (between the 2 sets of asterisks)
- On Customize Jenkins page click Install suggested plugins

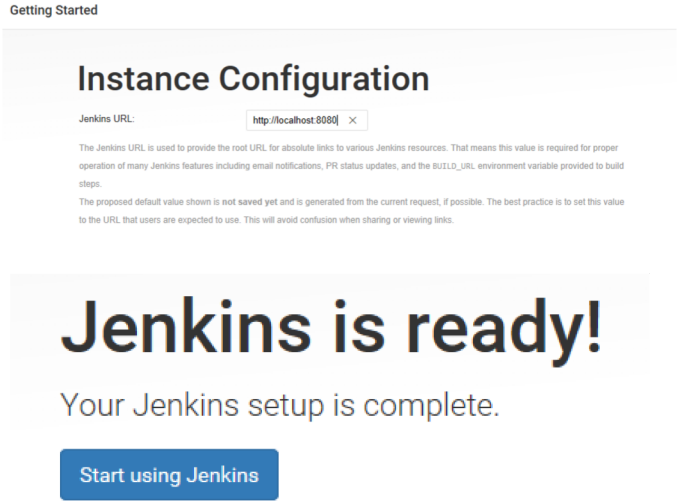


Getting Started

1. When the Create First Admin User page appears, specify your details in the respective fields and click Save and Finish.
2. When the Jenkins is ready page appears, click Start using Jenkins.
3. Please note
    - This page may indicate Jenkins is almost ready! instead and if so, click Restart.
    - If the page doesn't automatically refresh after a minute, use your web browser to refresh the page manually.
    - If required, log in to Jenkins with the credentials of the user you just created and you're ready to start using Jenkins!
4. you can stop the umangsaltuniv/devopstraining Docker container by typing `Ctrl-C`







Step 1 - Getting Sample Code(App & System Tests)

- Open the browser and go to https://github.com/login
- Login to your account
- Launch URL https://github.com/umangsaltuniv/verity-devops
- Click Fork at right top section
- "verity-devops" repository will be added on your GitHub account
- Launch URL https://github.com/umangsaltuniv/EMSystemTests.git
- Click Fork at right top section
- “EMSystemsTests” repository will be added on your GitHub account

  Note: verity-devops project has Expense Manager sample app & some unit tests those will run on the app to do unit testing of the app. EMSystemTests project has system test that will run on the app to do system testing of the app

Step 2 - Cloning Sample Code(App)

- After forking verity-devops repository on GitHub Web, Click “Clone or download”
- Click “Open in Desktop”
- Click “Open GitHubDesktop”
- In GitHub Desktop, before clicking Clone on the Clone a Repository dialog box, ensure Local Path is set to **C:\DO-United\GitHubRepo\verity-devops**
- Click Clone
- Note down the URL where you forked (https://github.com/<Your username>/verity-devops.git)
- GitHub Desktop will be opened in cloned repository

Step 2 - Cloning Sample Code(System Tests)

- After forking EMSystemTests on GitHub Web, Click “Clone or download”
- Click “Open in Desktop”
- Click “Open GitHubDesktop”
- In GitHub Desktop, before clicking Clone on the Clone a Repository dialog box, ensure Local Path is set to **C:\DO-United\GitHubRepo\EMSystemTests**
- Click Clone
- Note down the URL where you forked (https://github.com/<Your username>/EMSystemTests.git)
- GitHub Desktop will be opened in cloned repository

Step 3 - Create Pipeline Project

- Go to Jenkins Dashboard
- Click “New Item” at top left section
- Enter project name(**verity-devops**) > Select Pipeline > Click OK

  Note: Make sure jenkins pipeline project name & github repository name should be same(e.g verity-devops)
- Go to Pipeline section
- Choose the ”Pipeline script from SCM” option from the ”Definition” field
- Choose the ”Git” option from the ”SCM” field
- Enter your Repository URL(e.g. https://github.com/<Your username>/verity-devops.git)
- Enter “Jenkinsfile.txt” under “Script Path” section
- Click Apply > Save

Step 4 - Create Jenkinsfile

- Create and save new text file with the name Jenkinsfile.txt at the root of your local **verity-devops** Git repository
- Write the pipeline code (see next pages) into your empty Jenkinsfile stage by stage

Add Stage 1 – Clean

1. **Clean** defines a stage that appears on the Jenkins UI
2. **sh** runs the Maven command to cleanly target folder that includes code compiled classes, unit test reports, war files etc

```
pipeline {
    agent any
    stages {
                //Code starts for stage Clean
        stage('Clean') {
            steps {
                sh 'mvn clean'
            }
        }
                //Code ends for stage Clean
```

Step 5- Saving Jenkinsfile/Commit

- Save the edited Jenkinsfile and commit it to GitHub Web by following steps:
  - Open GitHub Desktop
  - JenkinsFile will be selected & displayed under Current repository "**verity-devops**"
  - Select Current branch as "master"
  - Enter Summary under "Summary" section
  - Click "Commit to master"
  - Click "Fetch origin"
  - Go to your repository on GitHub Web & Refresh the page
  - JenkinsFile will be displayed there

Step 6 - Running The Pipeline

- Go to Jenkins Dashboard
- Click “Open Blue Ocean” at left section
- Click your pipeline project
- Click Run
- Then quickly click the ”OPEN” link which appears at the lower right section to see running progress of your Pipeline project
- Jenkins Initially queues the project to be run on the agent

Add Stage 2 – PMD

1. **PMD** defines a stage that appears on the Jenkins UI
2. **sh** runs the Maven command to detect programming mistake detector

```
        //Code starts for stage PMD
        stage('PMD') {          (1)
    steps {
        sh 'mvn site'           (2)
    }
}
        //Code ends for stage PMD
```

Step 7- Saving Jenkinsfile/Commit

- Save the edited Jenkinsfile and commit it to GitHub Web by following steps:
  - Open GitHub Desktop
  - JenkinsFile will be selected & displayed under Current repository "**verity-devops**"
  - Select Current branch as "master"
  - Enter Summary under "Summary" section
  - Click "Commit to master"
  - Click "Fetch origin"
  - Go to your repository on GitHub Web & Refresh the page
  - JenkinsFile will be displayed there

Step 8 - Running The Pipeline

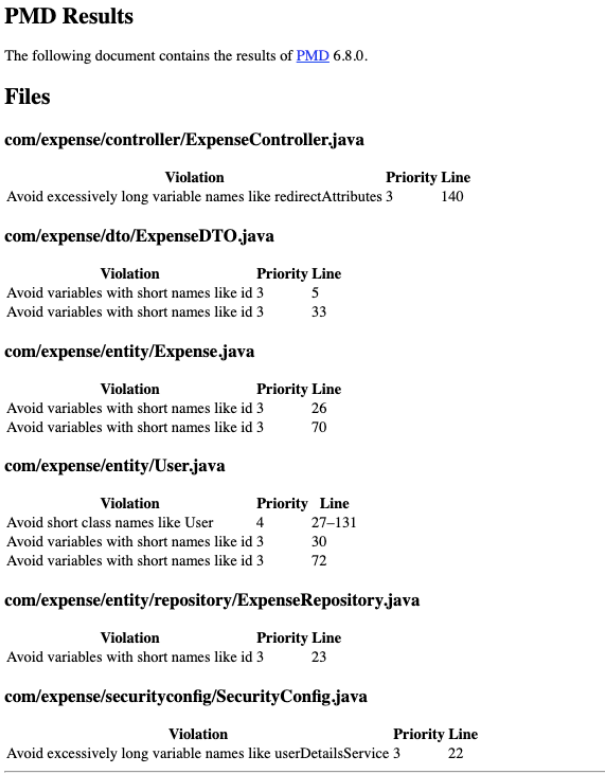
- Go to Jenkins Dashboard
- Click “Open Blue Ocean” at left section
- Click your pipeline project
- Click Run
- Then quickly click the ”OPEN” link which appears at the lower right section to see running progress of your Pipeline project
- Jenkins Initially queues the project to be run on the agent

Getting the PMD reports

- Launch below command in cmd
- docker container ls –a
- docker exec –it <<container id>> /bin/bash
- ls var/jenkins_home/workspace/ExpenseManager/target/site/
- pmd.html is stored in var/jenkins_home/workspace/ExpenseManager/target/site/
- The following command will download pmd.html to local machine
  - docker cp <<container id>>:var/jenkins_home/workspace/ExpenseManager/target/site/pmd.html D:\pmd.html

PMD Result

- You can see result in pmd.html



PMD Results

The following document contains the results of PMD 6.8.0.

Files

com/expense/controller/ExpenseController.java

Violation	Priority	Line
Avoid excessively long variable names like redirectAttributes	3	140

com/expense/dto/ExpenseDTO.java

Violation	Priority	Line
Avoid variables with short names like id	3	5
Avoid variables with short names like id	3	33

com/expense/entity/Expense.java

Violation	Priority	Line
Avoid variables with short names like id	3	26
Avoid variables with short names like id	3	70

com/expense/entity/User.java

Violation	Priority	Line
Avoid short class names like User	4	27–131
Avoid variables with short names like id	3	30
Avoid variables with short names like id	3	72

com/expense/entity/repository/ExpenseRepository.java

Violation	Priority	Line
Avoid variables with short names like id	3	23

com/expense/securityconfig/SecurityConfig.java

Violation	Priority	Line
Avoid excessively long variable names like userDetailsService	3	22

Add Stage 3 – Compile

1. **Compile** defines a stage that appears on the Jenkins UI
2. **sh** runs the Maven command to compile the source code

```
        //Code starts for stage Compile
        stage('Compile') {      (1)
    steps {
        sh 'mvn compile'        (2)
    }
}
        //Code Ends for stage Compile
```

Step 9- Saving Jenkinsfile/Commit

- Save the edited Jenkinsfile and commit it to GitHub Web by following steps:
  - Open GitHub Desktop
  - JenkinsFile will be selected & displayed under Current repository "**verity-devops**"
  - Select Current branch as "master"
  - Enter Summary under "Summary" section
  - Click "Commit to master"
  - Click "Fetch origin"
  - Go to your repository on GitHub Web & Refresh the page
  - JenkinsFile will be displayed there

Step 10 - Running The Pipeline

- Go to Jenkins Dashboard
- Click “Open Blue Ocean” at left section
- Click your pipeline project
- Click Run
- Then quickly click the ”OPEN” link which appears at the lower right section to see running progress of your Pipeline project
- Jenkins Initially queues the project to be run on the agent

Add Stage 4 – Static Code Analysis

1. **Static Code Analysis** defines a stage that appears on the Jenkins UI
2. Connects to SonarQube server
3. **sh** runs the sonar command to perform the static code analysis

```
        //Code starts for stage Static Code Analysis
        stage('Static Code Analysis') {         (1)
    steps {
                script {
  scannerHome = tool 'sonar-scanner'
}
                        withSonarQubeEnv('My SonarQube Server') (2)
                        {
                        sh "${scannerHome}/bin/sonar-scanner"   (3)
                        }
                }
        }
        //Code ends for stage Static Code Analysis
```

Step 11- Saving Jenkinsfile/Commit

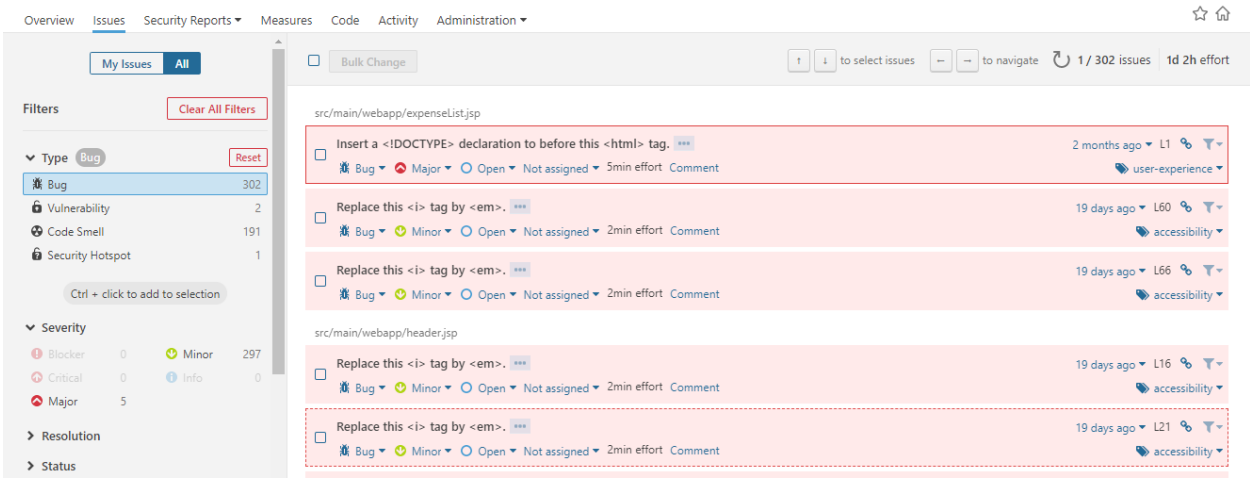
- Save the edited Jenkinsfile and commit it to GitHub Web by following steps:
  - Open GitHub Desktop
  - JenkinsFile will be selected & displayed under Current repository "**verity-devops**"
  - Select Current branch as "master"
  - Enter Summary under "Summary" section
  - Click "Commit to master"
  - Click "Fetch origin"
  - Go to your repository on GitHub Web & Refresh the page
  - JenkinsFile will be displayed there

Step 12 - Running The Pipeline

- Go to Jenkins Dashboard
- Click “Open Blue Ocean” at left section
- Click your pipeline project
- Click Run
- Then quickly click the ”OPEN” link which appears at the lower right section to see running progress of your Pipeline project
- Jenkins Initially queues the project to be run on the agent

Static Code Analysis Results

- Go to SonarQube web
- See the results



Add Stage 5 – Unit Test

1. **Unit Test** defines a stage that appears on the Jenkins UI
2. **sh** runs the Maven command to run the unit tests

Note: This **junit** command generates a JUnit XML report, which is saved to the target/surefire reports directory

```
        //Code starts for stage Unit Test
stage('Unit Test') {            (1)
    steps {
        sh 'mvn test'           (2)
    }
    post {
        always {
            junit 'target/surefire-reports/*.xml'
        }
    }
}
        //Code ends for stage Unit Test
```

Step 13- Saving Jenkinsfile/Commit

- Save the edited Jenkinsfile and commit it to GitHub Web by following steps:
  - Open GitHub Desktop
  - JenkinsFile will be selected & displayed under Current repository "**verity-devops**"
  - Select Current branch as "master"
  - Enter Summary under "Summary" section
  - Click "Commit to master"
  - Click "Fetch origin"
  - Go to your repository on GitHub Web & Refresh the page
  - JenkinsFile will be displayed there

Step 14 - Running The Pipeline

- Go to Jenkins Dashboard
- Click “Open Blue Ocean” at left section
- Click your pipeline project
- Click Run
- Then quickly click the ”OPEN” link which appears at the lower right section to see running progress of your Pipeline project
- Jenkins Initially queues the project to be run on the agent

Unit Test Result

- You can see result at below location in container:
- /var/jenkins_home/workspace/ExpenseManager/target/surefire-reports/

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 11, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 01:04 min
```

Add Stage 6 – JaCoCo

1. **JaCoCo** defines a stage that appears on the Jenkins UI
2. **sh** runs the Maven command to perform the code coverage

```
        //Code starts for stage JaCoCo
        stage('JaCoCo') {       (1)
    steps {
        sh 'mvn jacoco:report'  (2)
    }
}
        //Code ends for stage JaCoCo
```

Step 15- Saving Jenkinsfile/Commit

- Save the edited Jenkinsfile and commit it to GitHub Web by following steps:
  - Open GitHub Desktop
  - JenkinsFile will be selected & displayed under Current repository "**verity-devops**"
  - Select Current branch as "master"
  - Enter Summary under "Summary" section
  - Click "Commit to master"
  - Click "Fetch origin"
  - Go to your repository on GitHub Web & Refresh the page
  - JenkinsFile will be displayed there

Step 16 - Running The Pipeline

- Go to Jenkins Dashboard
- Click “Open Blue Ocean” at left section
- Click your pipeline project
- Click Run
- Then quickly click the ”OPEN” link which appears at the lower right section to see running progress of your Pipeline project
- Jenkins Initially queues the project to be run on the agent

Getting the JaCoCo reports

- Launch below command in cmd
- docker container ls –a
- docker exec –it <<container id>> /bin/bash
- ls var/jenkins_home/workspace/ExpenseManager/target/site/jacoco
- index.html is stored in var/jenkins_home/workspace/ExpenseManager/target/site/jacoco
- The following command will download index.html to local machine
  - docker cp <<container id>>:var/jenkins_home/workspace/ExpenseManager/target/site/jacoco/index.html D:\index.html

JaCoCo Result

- You can see result in index.html

ExpenseApp

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.expense.controller		42%		38%	10	15	54	83	6	11	0	2
com.expense.entity		67%		n/a	11	47	35	90	11	47	1	3
com.expense.dto		52%		n/a	3	22	5	33	3	22	0	1
com.expense.service.impl		70%		25%	7	18	10	35	5	16	0	3
com.expense.mapper		7%		n/a	6	7	4	5	6	7	0	1
com.expense		58%		n/a	1	3	2	4	1	3	0	1
com.expense.securityconfig		100%		n/a	0	5	0	24	0	5	0	1
Total	521 of 1,234	58%	8 of 12	33%	38	117	110	274	32	111	1	12

Add Stage 7 – Build

1. **Build** defines a stage that appears on the Jenkins UI
2. **sh** runs the Maven command to build the web application without running the unit tests

```
        //Code starts for stage Build
        stage('Build') {                (1)
    steps {
        sh 'mvn install -DskipTests'    (2)
    }
}
        //Code ends for stage Build
```

Step 17- Saving Jenkinsfile/Commit

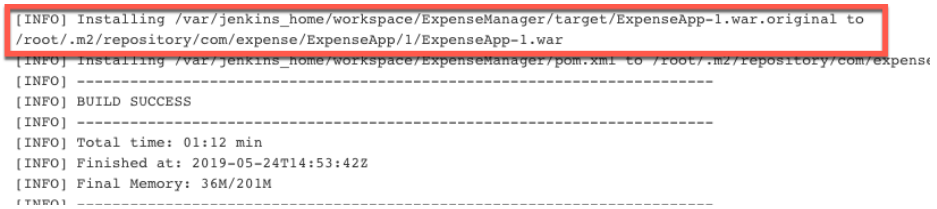
- Save the edited Jenkinsfile and commit it to GitHub Web by following steps:
  - Open GitHub Desktop
  - JenkinsFile will be selected & displayed under Current repository "**verity-devops**"
  - Select Current branch as "master"
  - Enter Summary under "Summary" section
  - Click "Commit to master"
  - Click "Fetch origin"
  - Go to your repository on GitHub Web & Refresh the page
  - JenkinsFile will be displayed there

Step 18 - Running The Pipeline

- Go to Jenkins Dashboard
- Click “Open Blue Ocean” at left section
- Click your pipeline project
- Click Run
- Then quickly click the ”OPEN” link which appears at the lower right section to see running progress of your Pipeline project
- Jenkins Initially queues the project to be run on the agent

Build Result

- The war file would be generated
- We can view the presence of the war file using
- You can find war at below location in container: /var/jenkins_home/workspace/ExpenseManager/target/



Add Stage 8 – Tomcat Server Up

1. **Tomcat Server Up** defines a stage that appears on the Jenkins UI
2. **sh** runs the command launch Tomcat server

```
        //Code starts for stage Tomcat Server Up
        stage('Tomcat Server Up') {                          (1)
    steps {
        sh '/tmp/apache-tomcat-9.0.20/bin/startup.sh'        (2)
    }
}
        //Code ends for stage Tomcat Server Up
```

Step 19- Saving Jenkinsfile/Commit

- Save the edited Jenkinsfile and commit it to GitHub Web by following steps:
  - Open GitHub Desktop
  - JenkinsFile will be selected & displayed under Current repository "**verity-devops**"
  - Select Current branch as "master"
  - Enter Summary under "Summary" section
  - Click "Commit to master"
  - Click "Fetch origin"
  - Go to your repository on GitHub Web & Refresh the page
  - JenkinsFile will be displayed there

Step 20 - Running The Pipeline

- Go to Jenkins Dashboard
- Click “Open Blue Ocean” at left section
- Click your pipeline project
- Click Run
- Then quickly click the ”OPEN” link which appears at the lower right section to see running progress of your Pipeline project
- Jenkins Initially queues the project to be run on the agent

Add Stage 9 – War Deployed on Tomcat Server

1. **War Deployed on Tomcat Server** defines a stage that appears on the Jenkins UI
2. **sh** runs the command to move war file from code project location to Tomcat's location

```
        //Code starts for stage War Deployed on Tomcat Server
        stage('War Deployed on Tomcat Server') {                (1)
    steps {
        sh 'cp /var/jenkins_home/workspace/verity-devops/target/ExpenseApp-1.war /tmp/apache-tomcat-9.0.20/webapps'      (2)
    }
}
        //Code ends for stage War Deployed on Tomcat Server
```

Step 21- Saving Jenkinsfile/Commit

- Save the edited Jenkinsfile and commit it to GitHub Web by following steps:
  - Open GitHub Desktop
  - JenkinsFile will be selected & displayed under Current repository "**verity-devops**"
  - Select Current branch as "master"
  - Enter Summary under "Summary" section
  - Click "Commit to master"
  - Click "Fetch origin"
  - Go to your repository on GitHub Web & Refresh the page
  - JenkinsFile will be displayed there

Step 22 - Running The Pipeline

- Go to Jenkins Dashboard
- Click “Open Blue Ocean” at left section
- Click your pipeline project
- Click Run
- Then quickly click the ”OPEN” link which appears at the lower right section to see running progress of your Pipeline project
- Jenkins Initially queues the project to be run on the agent

Add Stage 10 – System Test

1. **System Test** defines a stage that appears on the Jenkins UI
2. **sh** runs the Maven command to run the system tests

```
        //Code starts for stage System Test
        stage('System Test') {                                          (1)
    steps {
        //Change git url below as per your forked github repository URL
        git url: 'https://github.com/umangsaltuniv/EMSystemTests.git'
        sh 'mvn -Dtest=ExpenseManagerSystemTest test'                   (2)
    }
}
        //Code ends for stage System Test
}
}
```

Step 23- Saving Jenkinsfile/Commit

- Save the edited Jenkinsfile and commit it to GitHub Web by following steps:
  - Open GitHub Desktop
  - JenkinsFile will be selected & displayed under Current repository "**verity-devops**"
  - Select Current branch as "master"
  - Enter Summary under "Summary" section
  - Click "Commit to master"
  - Click "Fetch origin"
  - Go to your repository on GitHub Web & Refresh the page
  - JenkinsFile will be displayed there

Step 24 - Running The Pipeline

- Go to Jenkins Dashboard
- Click “Open Blue Ocean” at left section
- Click your pipeline project
- Click Run
- Then quickly click the ”OPEN” link which appears at the lower right section to see running progress of your Pipeline project
- Jenkins Initially queues the project to be run on the agent

Output

- Jenkins will show output of Deliver stage by showing execution results of your web application