BVRIT HYDERABAD
College of Engineering for Women

Department of Computer Science & Engineering

DevOps

Lab Manual

III B.TECH - I SEM

REGULATION : R 22

BVRIT HYDERABAD
College of Engineering for Women
Rajiv Gandhi Nagar, Bachupally, Hyderabad -90

Department of Computer Science & Engineering

BVRIT HYDERABAD

College of Engineering for Women

Rajiv Gandhi Nagar, Bachupally, Hyderabad -90

DevOps Lab

III Year – I Semester

INSTITUTE VISION & MISSION

VISION

To emerge as the best among the institutes of technology and research in the country dedicated to the cause of promoting quality technical education.

MISSION

At BVRITH, we strive to

- Achieve academic excellence through innovative learning practices.
- Enhance intellectual ability and technical competency for a successful career.
- Encourage research and innovation.
- Nurture students towards holistic development with emphasis on leadership skills. life skills and human values.

DEPARTMENT VISION & MISSION

VISION

Develop women as technocrats, researchers and entrepreneurs in the field of computer science and engineering.

MISSION

M1: To impart quality education in Computer Science and Engineering by means of learning techniques and value-added courses.

M2: To inculcate professional excellence and research culture by encouraging projects in cutting-edge technologies through industry interactions.

M3: To build leadership skills, ethical values and teamwork among the students.
M4: To strengthen the collaboration of department and industry through internships, mentorships and professional body activities.

Program Educational Objectives (PEOs)

PEO-1: Adapt emerging technologies to contribute to the technical innovations for the progressive development in their respective fields.

PEO-2: Productively engage in multidisciplinary research areas by applying the basic principles of engineering sciences.

PEO-3: Demonstrate strong technical skills to bring out novel designs/products to address social & environmental issues.

PEO-4: Exhibit professional attitude, teamwork and practice code of ethics.

Program Specific Objectives (PSOs)

PSO 1: Ability to apply learned skills to build optimized solutions pertaining to Computer & Communication Systems, Data Processing, and Artificial Intelligence.

PSO 2: Employ standard strategies and practices in project development using FOSS (Free & Open-Source Software).

Program Outcomes

PO1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriateconsideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Course Outcomes

C503.1: Understand components of DevOps environment.

C503.2: Apply different project management, integration, testing and code deployment tools

C503.3: Assess various DevOps practices

C503.4: Investigate different DevOps Software development models

CO-PO Mapping:

CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
C218.1	3	2	1	-	1	-	-	-	-	-	-	-
C218.2	3	3	3	2	3	-	-	-	-	-	1	-
C218.3	3	2	3	1	3	-	-	-	-	-	-	-
C218.4	3	1	1	-	3	-	-	-	-	-	-	-
Mean	3	2	1	-	1	-	-	-	-	-	-	-

PSO s AND CO s MAPPING:

CO/PSO	PSO1	PSO2
C218.1	2	1
C218.2	3	2
C218.3	2	1
C218.4	2	1
Mean	2.25	1.25

CONTENTS

Syllabus as per JNTU Hyderabad

List of Experiments

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

B.Tech. III Year I Sem

L	T	P	C
0	0	4	2

List of Experiments

1. Write code for a simple user registration form for an event.
2. Explore Git and GitHub commands.
3. Practice Source code management on GitHub. Experiment with the source code in exercise 1.
4. Jenkins installation and setup, explore the environment.
5. Demonstrate continuous integration and development using Jenkins.
6. Explore Docker commands for content management.
7. Develop a simple containerized application using Docker.
8. Integrate Kubernetes and Docker
9. Automate the process of running containerized applications for exercise 7 using Kubernetes.
10. Install and Explore Selenium for automated testing.
11. Write a simple program in JavaScript and perform testing using Selenium.
12. Develop test cases for the above containerized application using selenium.

BVRIT HYDERABAD
College of Engineering for Women
Rajiv Gandhi Nagar, Bachupally, Hyderabad -90

DO'S

- On entering the lab fill the details in log book.
- Shutdown the system properly while leaving the lab.
- Keep the Computer lab premises clean and tidy.
- Contact the System Administrator if you notice any kind of machine malfunction.

DONT'S

- Don't install any application software.
- Don't disconnect or modify any machine, either PC or peripheral equipment.
- Don't connect your personal laptop machine in places other than designated.
- Don't eat or drink in the computer lab.
- Don't carry your mobile phones.
- Don't connect USB storage Devices to PC's.

BVRIT HYDERABAD
College of Engineering for Women
Rajiv Gandhi Nagar, Bachupally, Hyderabad -90

Department of Computer Science & Engineering
DevOPs

III Year – I Semester, CSE

INDEX

INSTRUCTIONS TO STUDENTS

(1) After entering the lab, make an entry in the log book providing the specified details.

(2) Observation book and Lab Record should be duly signed by the concerned faculty after the completion of each experiment, failing which marks will not be awarded. All experiments must be completed as per the schedule.

(3) After the completion of lab, properly shutdown the system.

GRADING:

- The overall lab evaluation is for **100** marks. In that **60** marks for Semester End Examination and remaining **35** marks will be awarded based on **internal evaluation** and **5** marks for project
- Internal Evaluation consists of:
  - Lab internal examination: There will be two internal examinations for 40 marks each. The final marks are awarded as the average of two internal examination marks.
  - Continuation evaluation is for 40 marks and distributed for each experiment as:

A.	A write-up on day-to-day experiment in the laboratory (in terms of aim, components / procedure, expected outcome)	10 Marks
B.	viva-voce (or) tutorial (or) case study (or) application (or) poster presentation of the course concerned.	10 Marks
C.	Internal practical examination	10 Marks
D.	Laboratory Report / Project and Presentation, which consists of the Design (or) Software / Hardware Model Presentation (or) App Development (or) Prototype Presentation	10 Marks

Program No.	List of Programs
1	Write code for a simple user registration form for an event.
2	Explore Git and GitHub commands.
3	Practice Source code management on GitHub. Experiment with the source code in exercise 1.
4	Jenkins installation and setup, explore the environment.
5	Demonstrate continuous integration and development using Jenkins.
6	Explore Docker commands for content management.
7	Develop a simple containerized application using Docker.
8	Integrate Kubernetes and Docker
9	Automate the process of running containerized applications for exercise 7 using Kubernetes.
10	Install and Explore Selenium for automated testing.
11	Write a simple program in JavaScript and perform testing using Selenium.
12	Develop test cases for the above containerized application using selenium.

Lab Cycle I

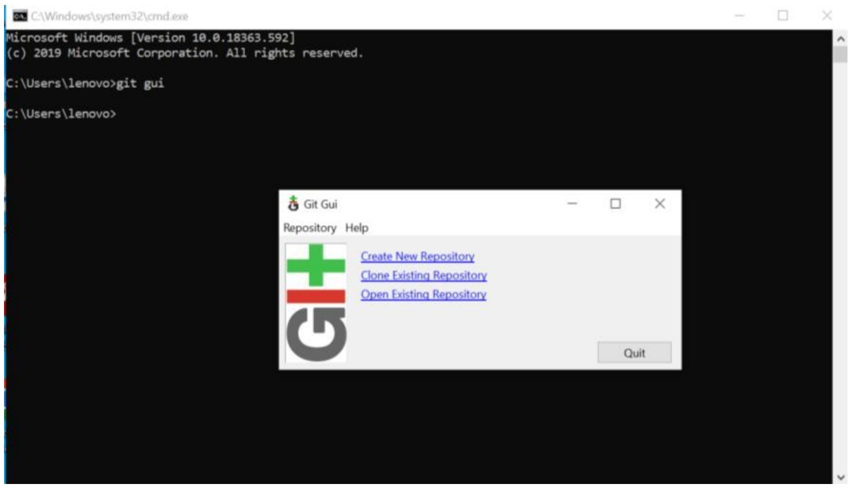
Programs

Program 1: Write code for a simple user registration for an event.
Make a user registration form in html. Open notepad, write the program, save as registration.htm', open the file with web browser.

```
<Html>
<head>
<title>
Registration Page for Science Fair
</title>
</head>
<body bgcolor="Lightskyblue">
<br>
<br>
<form>
<label> Firstname </label>
<input type="text" name="firstname" size="15"/> <br> <br>
<label> Middlename: </label>
<input type="text" name="middlename" size="15"/> <br> <br>
<label> Lastname: </label>
<input type="text" name="lastname" size="15"/> <br> <br>
<br>
<br>
<label>
Gender :
</label><br>
<input type="radio" name="male"/> Male <br>
<input type="radio" name="female"/> Female <br>
<input type="radio" name="other"/> Other
<br>
<br>
<label>
Phone :
</label>
<input type="text" name="country code" value="+91" size="2"/>
<input type="text" name="phone" size="10"/> <br> <br>
Address
<br>
    <textarea cols="80" rows="5" value="address">
```

Program2: Explore Git and Git Hub commands.

- Prerequisites:-a command line interface, a text editor, a Git Hub account.
- Git is a version control system which helps you track changes made to files overtime. Git maintains a local repository, where you make commit changes to the project before pushing them to the central repository on Git Hub.
- Github hosts Git repositories and provides developers with tools to s
- Install Git from- https://git-scm.com/downloads
- Launch Git GUI- open Windows Start menu, type git gui and press Enter (or click the application icon)



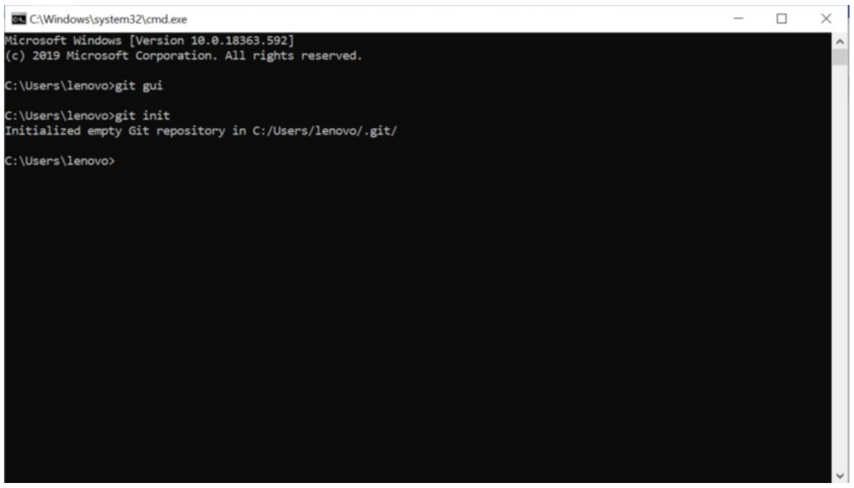
```
Microsoft Windows [Version 10.0.18363.592]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\lenovo>git gui

C:\Users\lenovo>
```

Git Commands: working with local repositories

- **git init –** the command git init is used to create an empty Git repository. After the git init

command is used, a .git folder is created in the directory with some jsubdirectories. Once the repository is initialized, the process of creating other files begins.

git init



```
Microsoft Windows [Version 10.0.18363.592]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\lenovo>git gui

C:\Users\lenovo>git init
Initialized empty Git repository in C:/Users/lenovo/.git/

C:\Users\lenovo>
```

- **git add** – add command is used after checking the status of the files, to add those files to th staging area. Before running commit command, "git add" is used to add any new or modifie files.

git add

- **git commit** – the commit command makes sure that the changes are saved to the loca repository. The command "git commit –m <message>" allows you to describe everyone an help them understand what has happened.

git commit –m "commit message"

- **git status-** the git status command tells the current state of the repository. The command, i will be shown by the git status. Also, if there are no changes, it will show the message n changes to commit, working directory clean.

- **git config-** the git config command is used to configure the user.name and user.email. thi specifies what email id and username will be used from a local repository. When git config i used with –global flag, it writes the settings to all repositories on the computer.

git config –global user.name "any username"
git config –global user.email<email id>

- **git branch** – the git branch command is used to determine what branch the local repositor is on. The command enables adding and deleting a branch.

```
#create a new branch
git branch <branch_name>
#list all remote or local branches
git branch –a
#delete a branch
git branch –d <branch_name>
```

- **git checkout**- the git checkout command is used to switch branches, whenever the work is to be started on a different branch. The command works on three separate entities: files, commits and branches.

```
#checkout an existing branch
 git checkout <branch_name>
#checkout and create a new branch with that name
git checkout –b<new_branch>
```

- **git merge**- the git merge command is used to integrate the branches together. The command combines the changes from branch to another branch. It is used to merge the changes in the staging branch to the stable branch.

```
git merge<branch_name>
```

Activity diagrams co

Git commands : working with remote repositories

- **git remote**- the git remote command is used to create, view and delete connections to other repositories. The connections here are not like direct links into other repositories, but as bookmarks that serve as convenient names to be as a reference.
  git remote add origin<address>
- **git clone-** the git clone command is used to create a local working copy of an existing remote repository.The command downloads the remote repository to the computer.It is equivalent to the git init command when working with a remote repository.
  git clone <remote_URL>
- **git pull-** the git pull command is used to fetch and merge changes from the remote repository to the local repository.
  git pull<branch_name><remoteURL>
- **git push-** the command git push is used to transfer the commits or pushing the content from the local repository to the remote repository. The command is used after a local repository has been modified, and the modifications are to be shared with the remote team members.
  git push –u origin master

- **git stash-** the git stash takes your modified tracked files and saves it on a pile of in complete changes that you can reapply at any time. To go back to work, you can use the stash pop. The git stash command will help a developer switch branches to work on something else without committing to in complete work.
  #store current work with untracked files
  Gitstash–u
  #bring stashed work back to the working directory
  Gitstash pop

- **git log**- the git log command shows the order of the commit history for a repository. The command helps in understanding the state of the current branch by showing the commits that lead to this state.
  git log

Program3: Practice Source code management on GitHub. Experiment with the source code written inexercise1.

- GitHub is an online hosting service for Git repositories. We push local repository in GitHub.
- Create GitHub account on https://github.com/
- create a repository in profile.
  open command prompt. Use following commands.
- cd<localdir> #to the location where the file of exercise1 is stored.
- git init #initializes empty git repository on the location.
- git add . #adds all the files to the staging area
- git commit –m “initial commit”
- git remote add origin <git hub repository url>
- git push –u origin main / git push origin HEADl:master

output : the exercise 1 file (registration.htm) will be pushed to your github repository

Program 4: Jenkins installation and setup, explore the environment.

Hardware requirement: RAM4 GB, Hard disk space more than 50 GB.
Software requirement: Java Development kit (JDK) or Java Runtime Environment (JRE), Web Browser (Google Chrome, Mozilla Firefox, Microsoft Edge), Operating system (Ubuntu 18.04 server installed with a non-root sudo user and firewall).

install Jenkins:

• Add the framework repository key:

```
$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add
```

```
ljs@lenovo:~$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key |
sudo apt-key add
OK
```

• Next, link the repository of the Debian packages to the sources.list of the server: $ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

• When both are in place, upgrade to apt to use the new repository:

```
$ sudo apt update
```

• Install Jenkins:

```
$ sudo apt install jenkins
```

Start the Jenkins server, as Jenkins and its dependencies are in place.

Start Jenkins:

Start Jenkins using systemctl:

```
$ sudo systemctl start jenkins
```

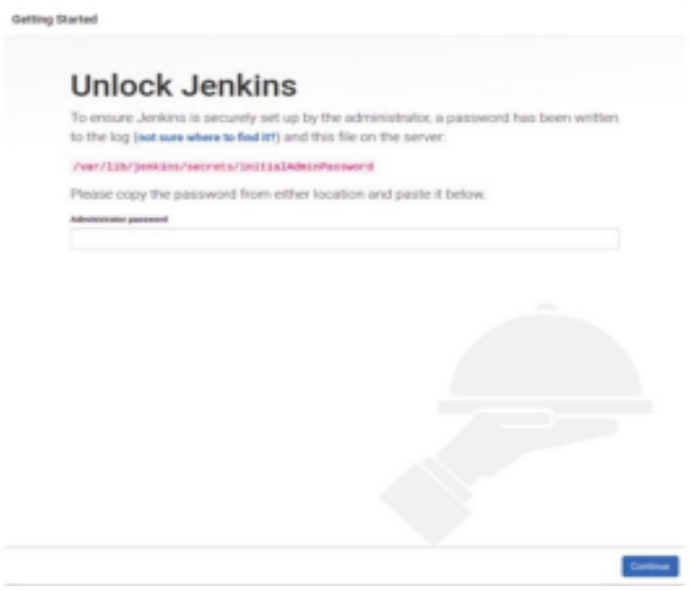
• As systemctl doesnot display performance, you can use the status command to check that Jenkins has successfully launched:

```
$ sudo systemctl status Jenkins
```

Output: jenkins.service - LSB: Start Jenkins at boot time Loaded:
loaded (/etc/init.d/jenkins; generated)
Active: active (exited) since Sat 2021-04-17 00:34:17 IST; 26s ago Docs:
man:systemd-sysv-generator(8)
Process: 17609 ExecStart=/etc/init.d/jenkins start(code=exited,status=0/SUCC

Setting up Jenkins:

- To set up installation, visit Jenkins on its default 8080 port with your server domain name or IP address: http://your_server_ip_or_domain:8080



Getting Started

Unlock Jenkins

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

/var/lib/jenkins/secrets/initialAdminPassword

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

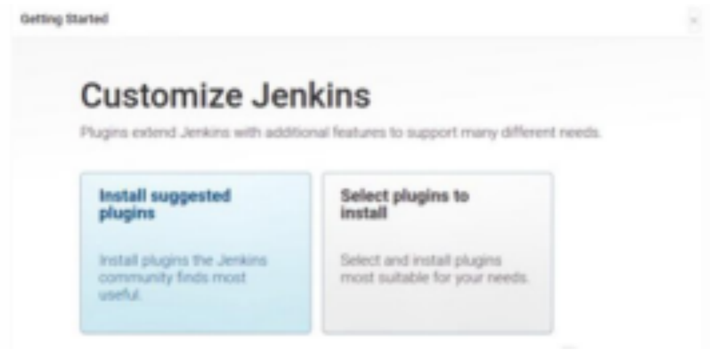
Administrator password

use cat command to display the password

```
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

• copy the alphanumeric terminal 32-character password and paste into Admin password field, then click continue.

```
Output: 0aaaf00d9afe48e5b7f2a494d1881326
```



Getting Started

Customize Jenkins

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

Install suggested plugins

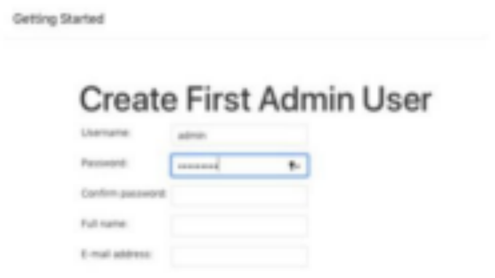
Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Click install suggested plugins to start installation.

• Once installation is done, the first admin user will be prompted. Save this step and use initial password to continue as an Admin.



Getting Started

Create First Admin User

Username: admin

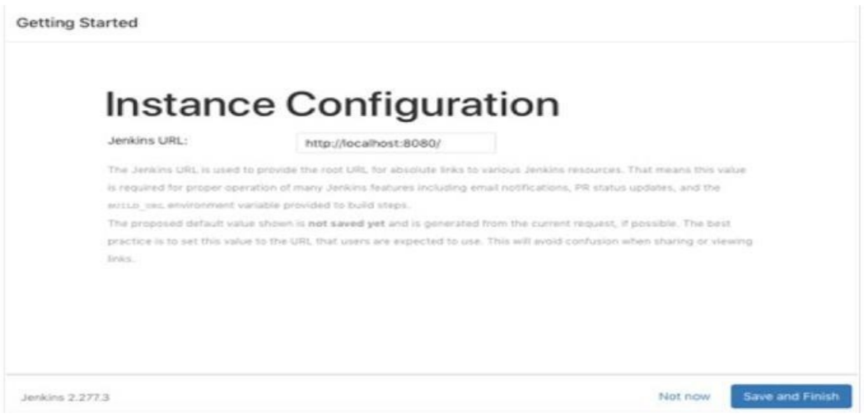
Password:

Confirm password:

Full name:

E-mail address:

- On configuration page, confirm either your server's domain name or the IP address of your server.



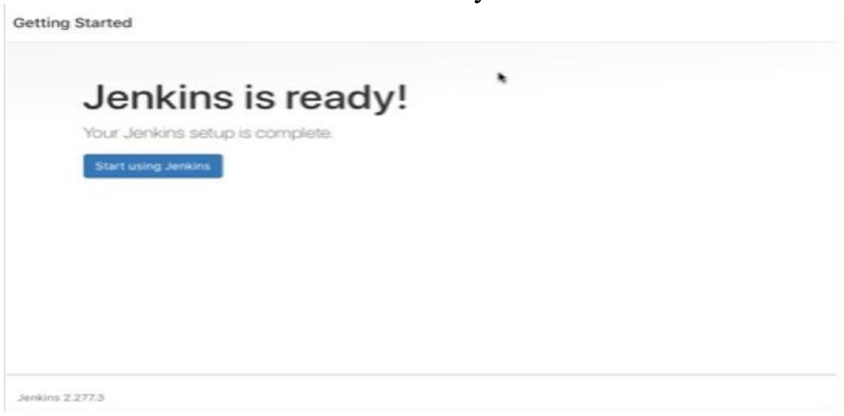
Getting Started

Instance Configuration

Jenkins URL: http://localhost:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.
The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

- Click save and Finish. Jenkins is ready

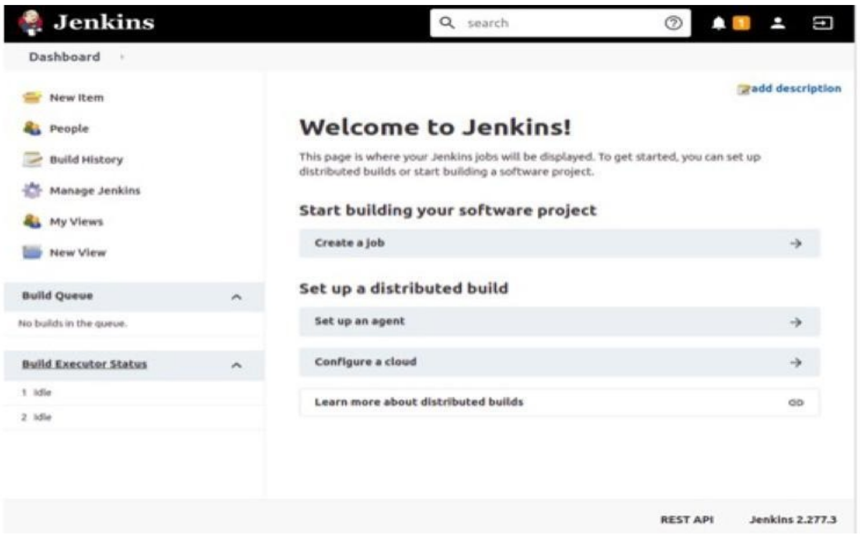


Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

- Hit start using Jenkins button. Jenkins dashboard appears.



No builds in the queue.

1 Idle
2 Idle

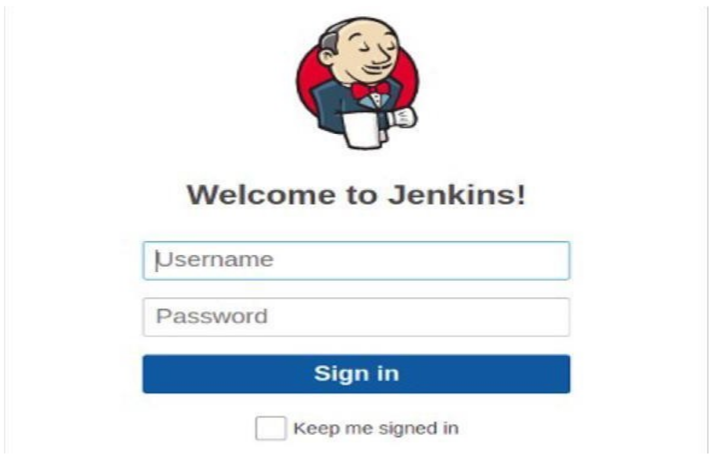
Welcome to Jenkins!

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.

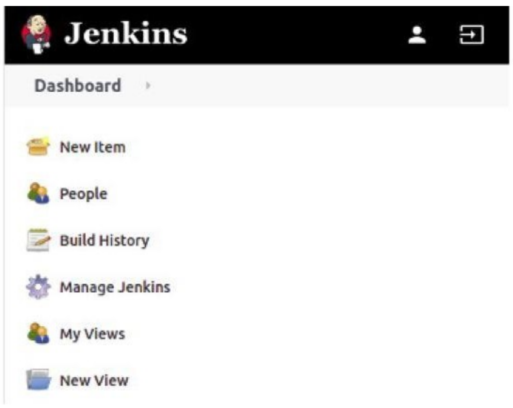
Start building your software project

Set up a distributed build

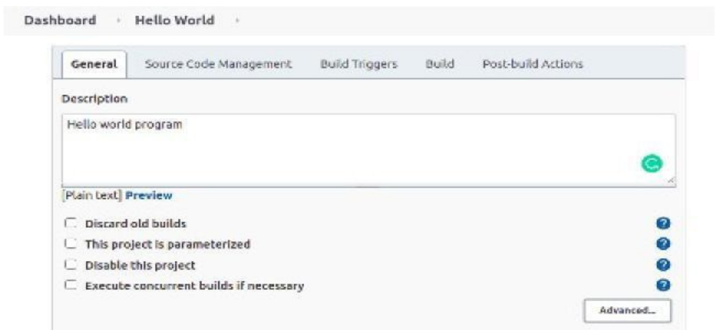
- **Creation of new build jobs in Jenkins:**
  Login page



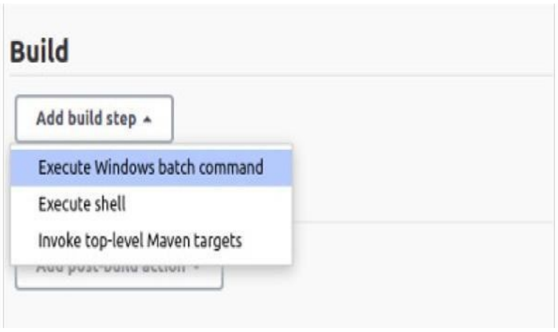
- Create 'new item': click on New item.



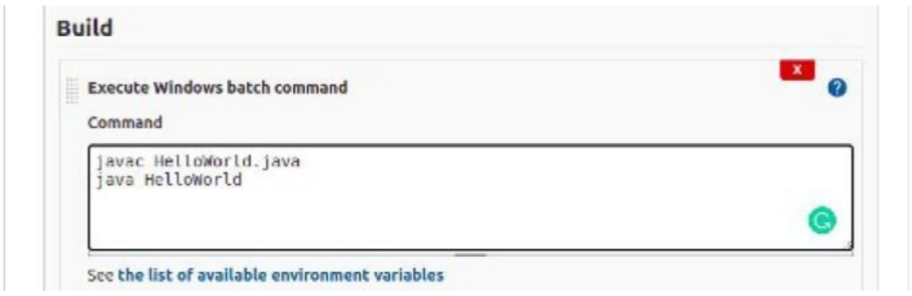
- Fill the project description: enter details for the job as per requirement.



- Source code management:
  Under source code management, enter the repository URL or use local repository.
  Build environment:
  Click on the "Add build Setup" and select "Execute Windows Batch Command"



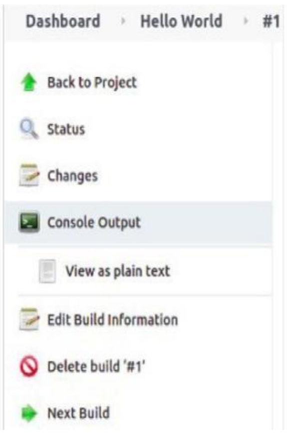
- Now, add the java commands, e.g javac, HelloWorld.java, java HelloWorld.



```
javac HelloWorld.java
java HelloWorld
```

See the list of available environment variables

- Save the project: click apply and save the project.
- Build source code and check its status: click on "Build Now" on the left-hand side of the screen to create the source code.
- Console output: select the build number and click on "Console Output" to check the status of the build run.

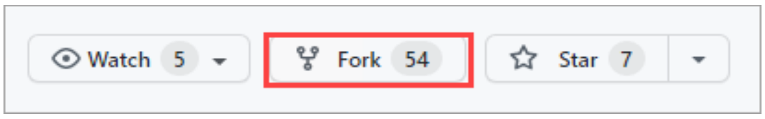


This screen means the HelloWorld program is successfully executed from the GitHub repository.
In case of failure, check the job logs by clicking on failure icon and debug the root cause.

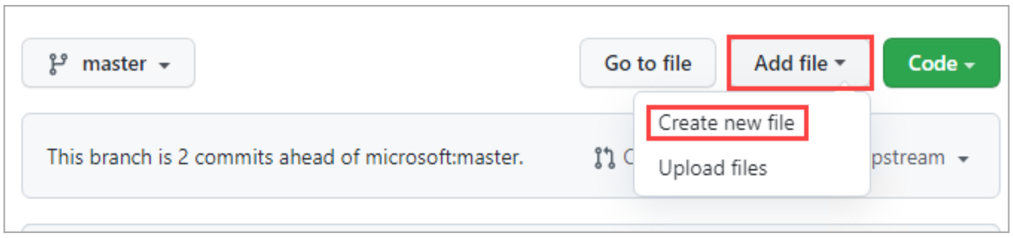
Program 5: Demonstrate continuous integration and development using Jenkins.

CI/CD pipeline with a GitHub repository:

- Login to the GitHub account.
- Next, navigate to the repository url, suppose MyShuttle2. Once you are on the repo page, click the Fork button.

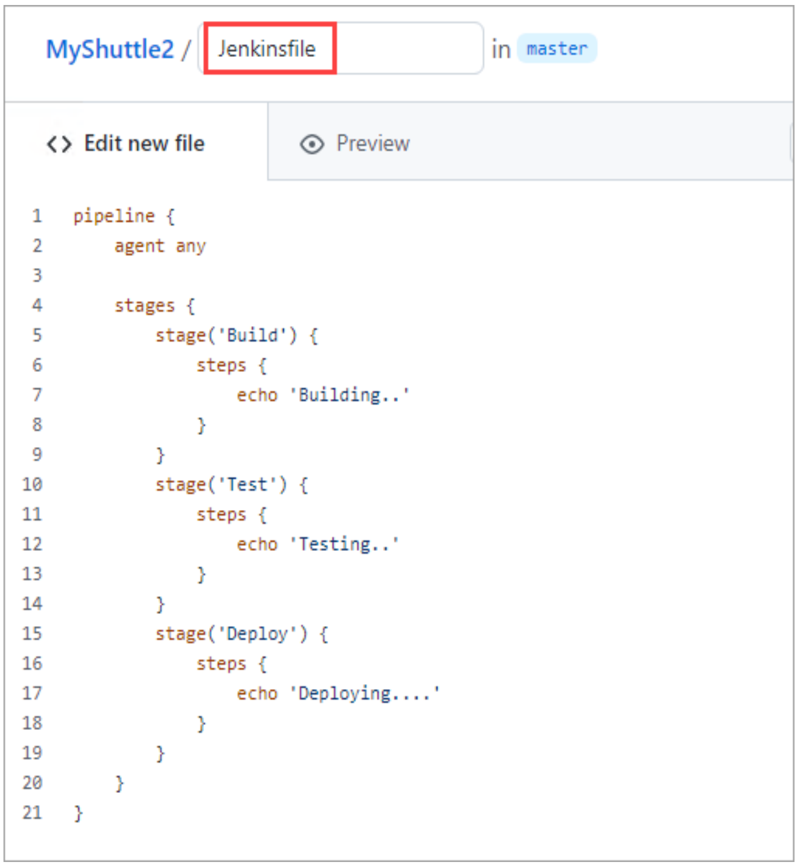


- Add a new file to the repository named Jenkinsfile. A Jenkinsfile is a text file that contains the definition of a Jenkins Pipeline. This file should be inside the source code repository.



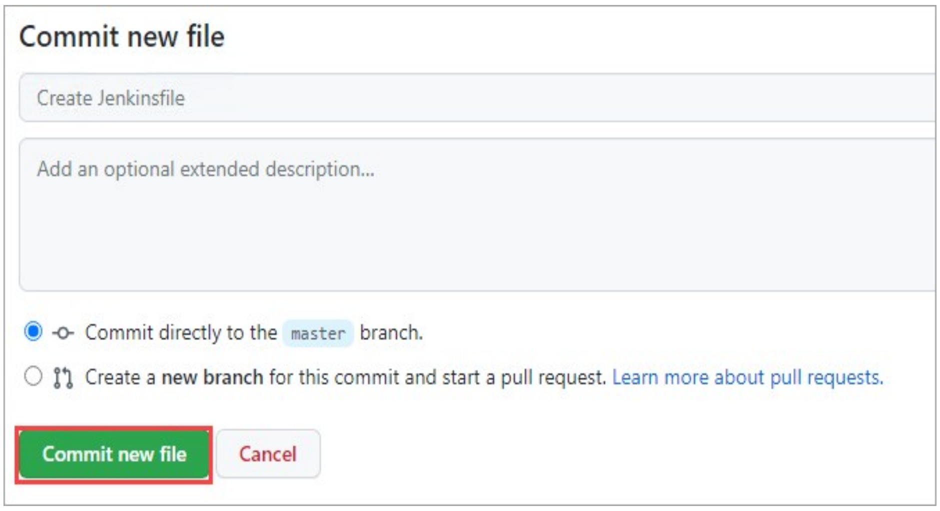
This branch is 2 commits ahead of microsoft:master.

- Type the pipeline script in Jenkinsfile as the name of the new file.



```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo 'Building..'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying....'
            }
        }
    }
}
```

- Press commit new file, at the bottom of the page.



Commit new file

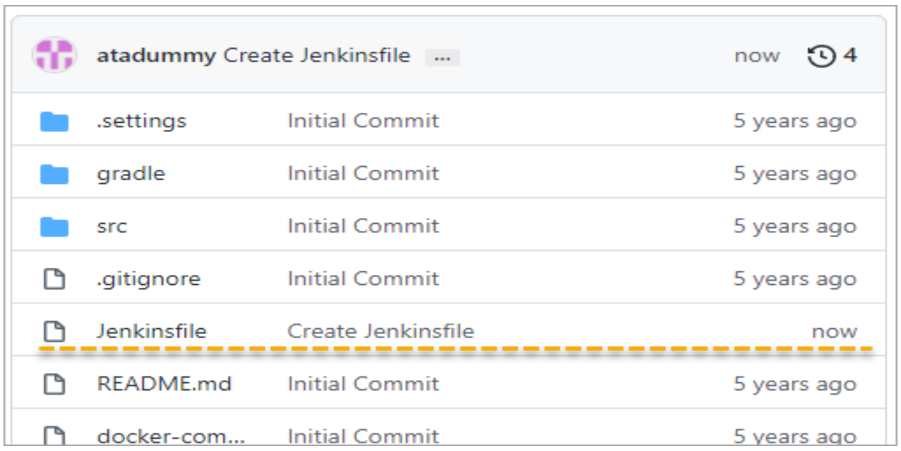
Create Jenkinsfile

Add an optional extended description...

[✓ Commit directly to the master branch.]

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

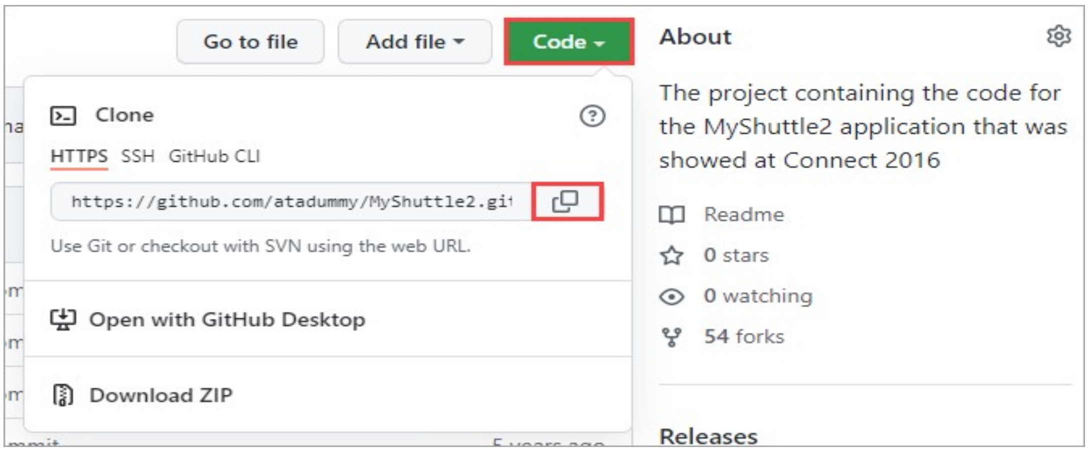
- This will show the jenkinsfile in your repository.



atadummy Create Jenkinsfile ... now 4

.settings	Initial Commit	5 years ago
gradle	Initial Commit	5 years ago
src	Initial Commit	5 years ago
.gitignore	Initial Commit	5 years ago
Jenkinsfile	Create Jenkinsfile	now
README.md	Initial Commit	5 years ago
docker-com...	Initial Commit	5 years ago

- As you will need the resulting URL during the Jenkins CI CD pipeline creation, copy the repository URL.



```
https://github.com/atadummy/MyShuttle2.git
```

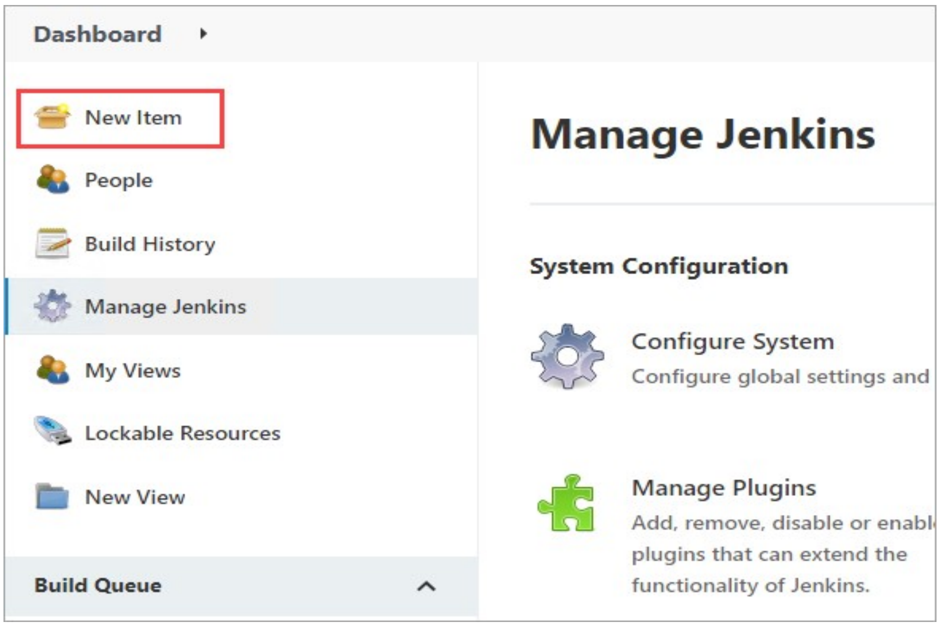
Use Git or checkout with SVN using the web URL.

About

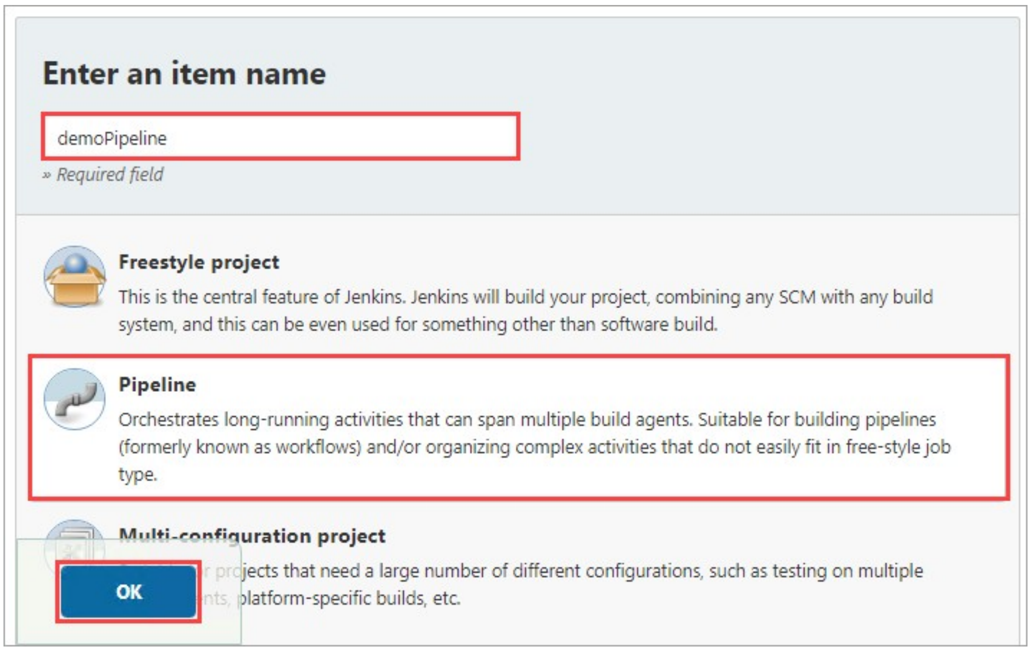
The project containing the code for the MyShuttle2 application that was showed at Connect 2016

Creating Jenkins CI CD Pipeline:

- Login to the Jenkins URL
- On the Jenkins dashboard, click the New Item option.



- Enter the name of the new pipeline. E.g. demoPipeline and select Pipeline template, click OK to finish.

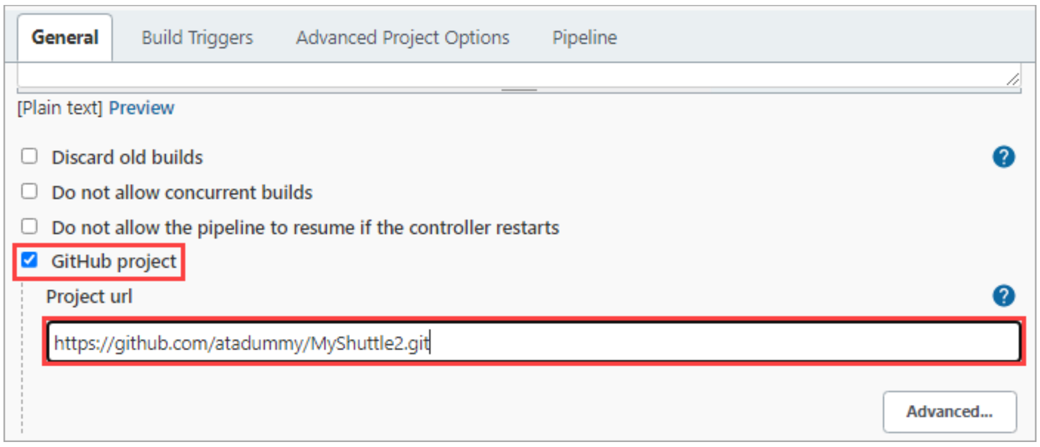


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

projects that need a large number of different configurations, such as testing on multiple platform-specific builds, etc.

- Check the GitHub box under the General tab on the configuration page. Next, paste the repository URL in the Project URL box.


General | Build Triggers | Advanced Project Options | Pipeline

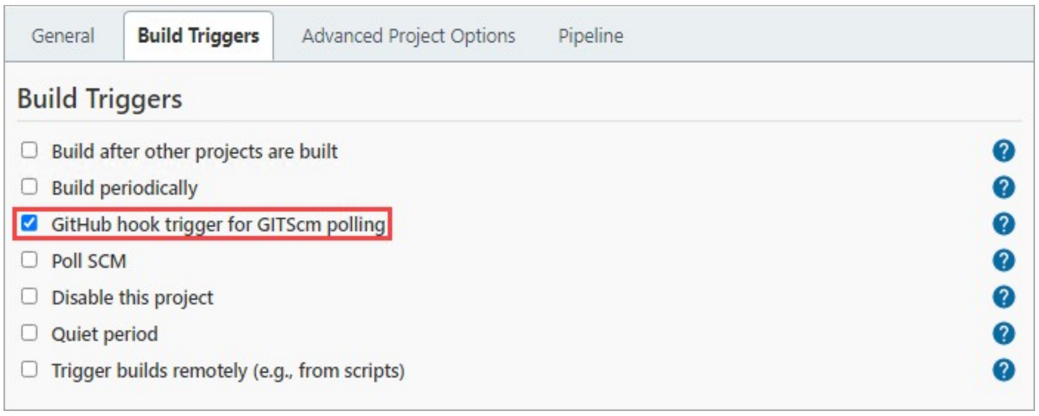
[Plain text] Preview

- [ ] Discard old builds
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [x] GitHub project

Project url

https://github.com/atadummy/MyShuttle2.git

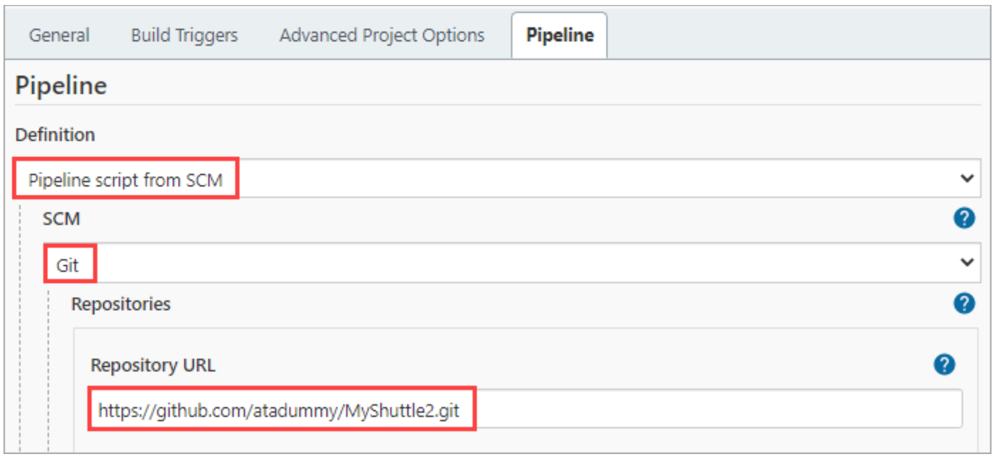
- Scroll down to the Build Triggers section and check the GitHub hook trigger for GITScm polling box.


General | Build Triggers | Advanced Project Options | Pipeline

Build Triggers

- [ ] Build after other projects are built
- [ ] Build periodically
- [x] GitHub hook trigger for GITScm polling
- [ ] Poll SCM
- [ ] Disable this project
- [ ] Quiet period
- [ ] Trigger builds remotely (e.g., from scripts)

- Scroll further down to the Pipeline section and select/specify the following:
  ✓ Definition: pipeline script from SCM.
  ✓ SCM: Git
  ✓ Repository URL: your repository URL.


General | Build Triggers | Advanced Project Options | Pipeline

Pipeline

Definition

Pipeline script from SCM

SCM

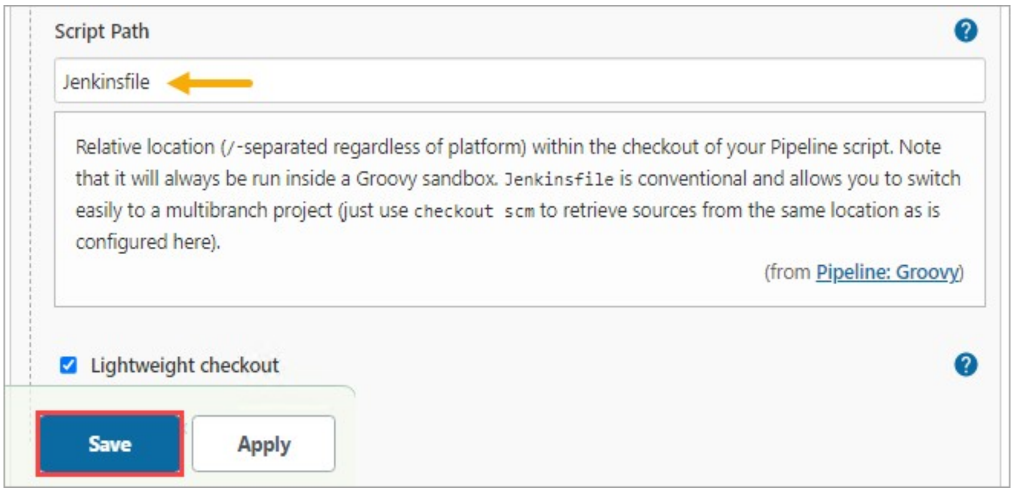
Git

Repositories

Repository URL

https://github.com/atadummy/MyShuttle2.git

- Confirm that the Script Path value is Jenkinsfile. This value refers the Jenkinsfile youcreated earlier, click Save.



Script Path

Jenkinsfile

Relative location (/-separated regardless of platform) within the checkout of your Pipeline script. Note that it will always be run inside a Groovy sandbox. `Jenkinsfile` is conventional and allows you to switch easily to a multibranch project (just use `checkout scm` to retrieve sources from the same location as is configured here).

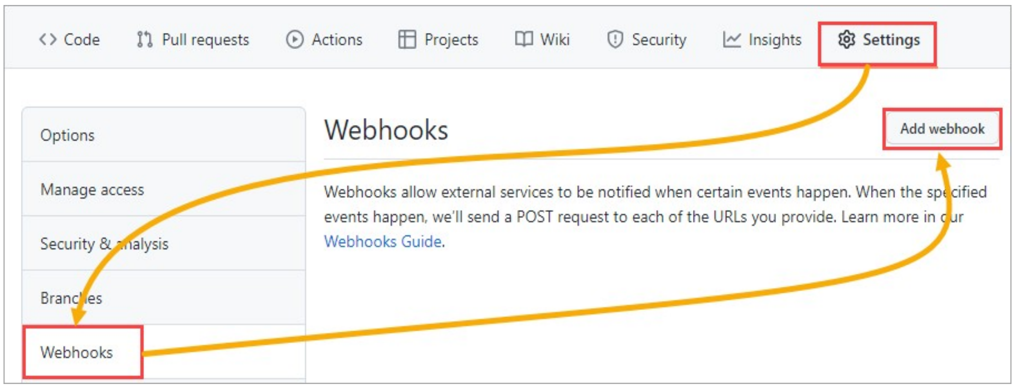
(from Pipeline: Groovy)

☑ Lightweight checkout

Configuring a Webhook in GitHub: for Jenkins to run a new job, you must create a webhook

in the GitHub repository. This webhook will notify Jenkins as soon as a new push occurs in the repository.

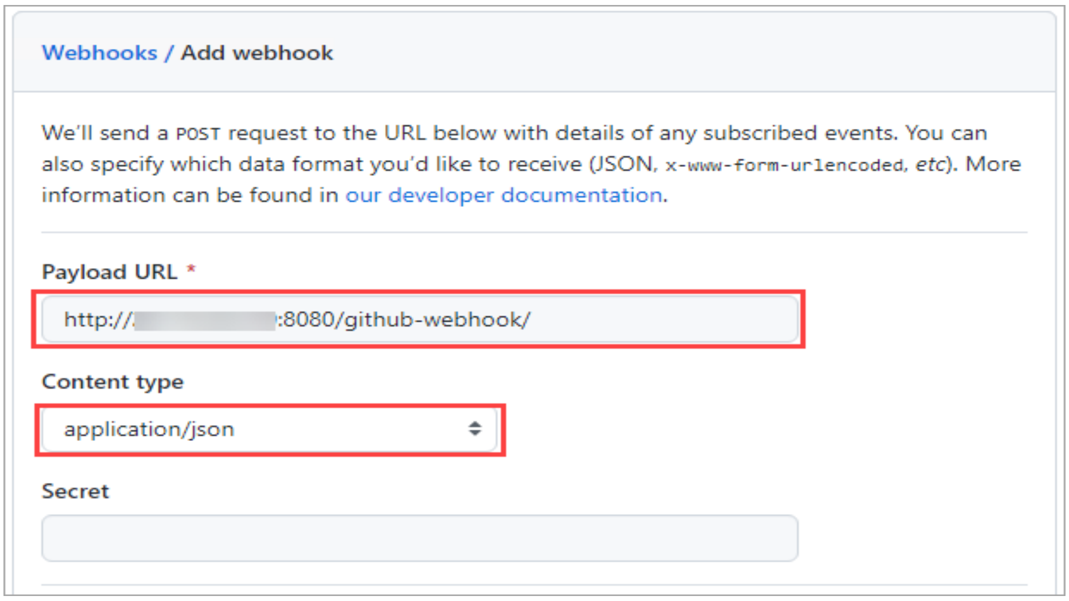
• In your GitHub repository, navigate to the settings page and click the Webhooks tab ,click Add webhook.



Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

- Specify your Jenkins URL and append /github-webhook/in the Payload URL field. Change the Content type value to application/json.



Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, *etc*). More information can be found in our developer documentation.

Payload URL *

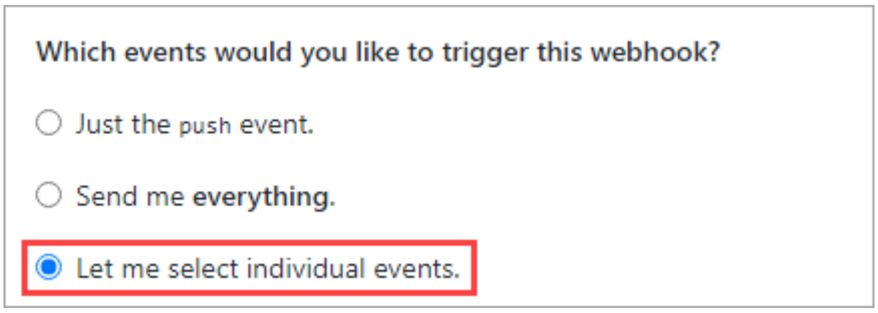
http://[redacted]:8080/github-webhook/

Content type

application/json

Secret

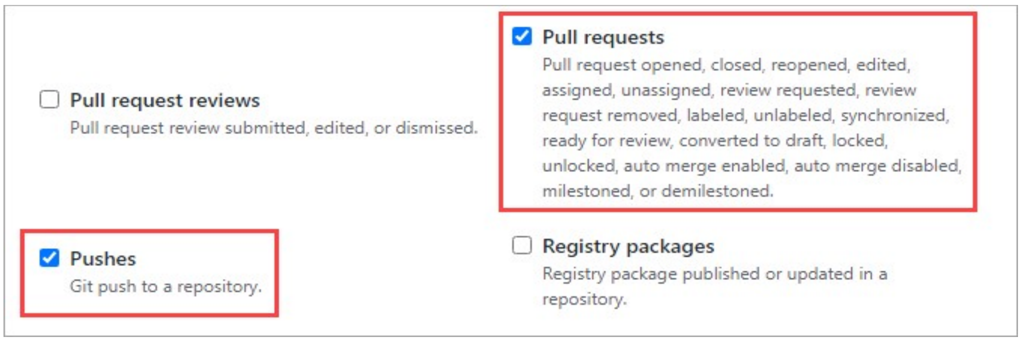
- Select the events when u like to trigger this webhook.



Which events would you like to trigger this webhook?

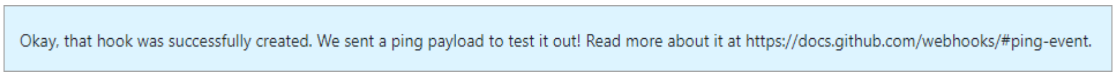
- [ ] Just the push event.
- [ ] Send me everything.
- [✓] Let me select individual events.

- Check the Pushes and Pull requests checkboxes.



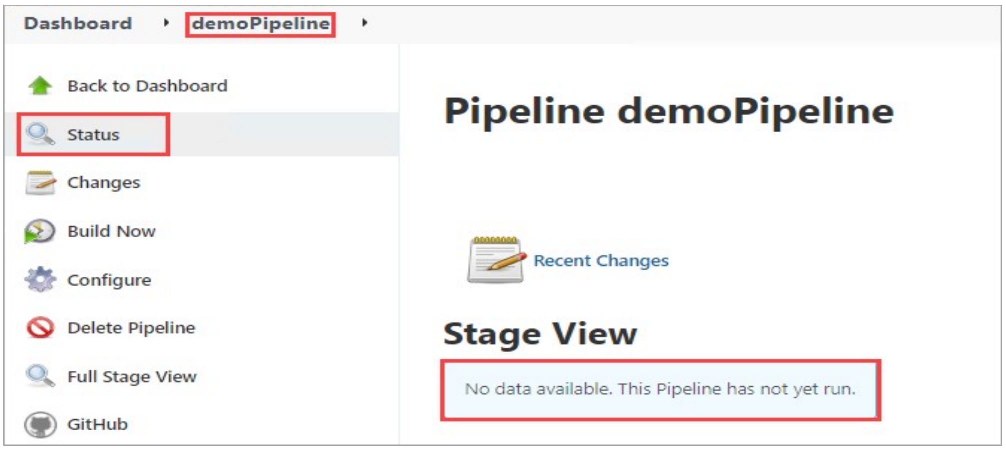
- [ ] **Pull request reviews** Pull request review submitted, edited, or dismissed.
- [✓] **Pull requests** Pull request opened, closed, reopened, edited, assigned, unassigned, review requested, review request removed, labeled, unlabeled, synchronized, ready for review, converted to draft, locked, unlocked, auto merge enabled, auto merge disabled, milestoned, or demilestoned.
- [✓] **Pushes** Git push to a repository.
- [ ] **Registry packages** Registry package published or updated in a repository.

- Validate the webhook by clicking the Add webhook button at the bottom.



Okay, that hook was successfully created. We sent a ping payload to test it out! Read more about it at https://docs.github.com/webhooks/#ping-event.

- Open the pipeline in Jenkins and click on Status. Here you see Stage View, shows No data available as there hasn't been an event that would trigger the pipeline.

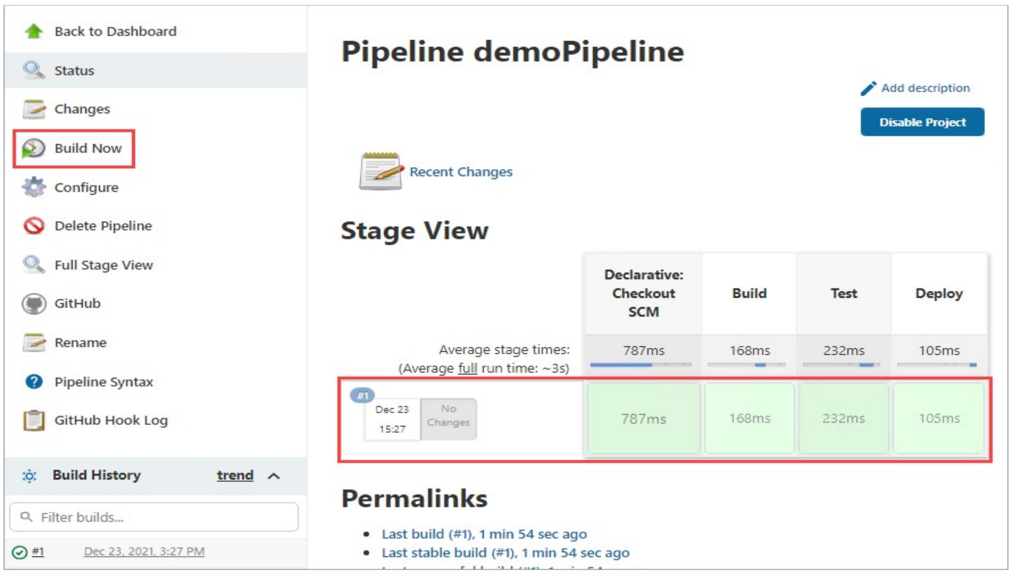


Pipeline demoPipeline

Stage View

No data available. This Pipeline has not yet run.

- To generate initial build data, click Build Now



Pipeline demoPipeline

Stage View

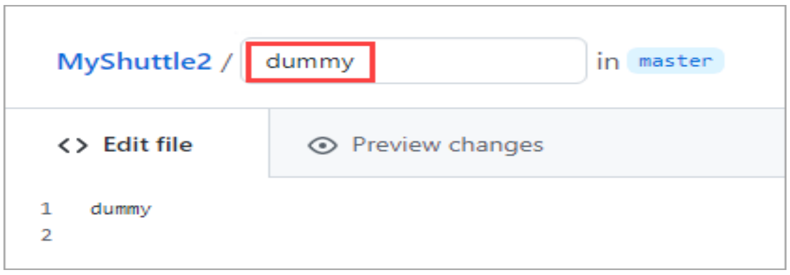
	Declarative: Checkout SCM	Build	Test	Deploy
Average stage times: (Average full run time: ~3s)	787ms	168ms	232ms	105ms
#1 Dec 23 15:27 No Changes	787ms	168ms	232ms	105ms

#1 Dec 23, 2021, 3:27 PM

Permalinks

- Last build (#1), 1 min 54 sec ago
- Last stable build (#1), 1 min 54 sec ago

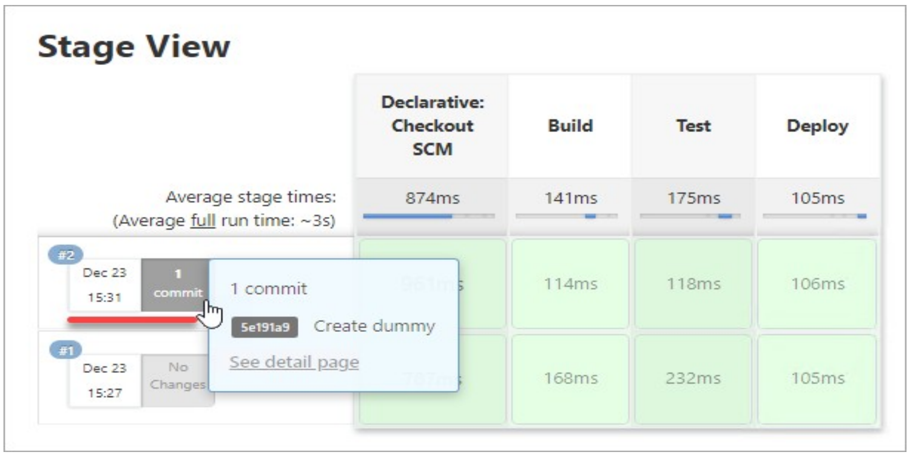
- To test the pipeline, add a new dummy file in the GitHub repository.



```
dummy
```

Click Commit new file at the bottom of the page.

- Go back to your Jenkins pipeline status page, you will see a new build entry with one commit.



Stage View

	Declarative: Checkout SCM	Build	Test	Deploy
Average stage times: (Average full run time: ~3s)	874ms	141ms	175ms	105ms
#2 Dec 23 15:31 1 commit	[illegible]	114ms	118ms	106ms
#1 Dec 23 15:27 No Changes	[illegible]	168ms	232ms	105ms

1 commit

5e191a9 Create dummy

See detail page

Program 6 : Explore Docker commands for content management.

• Run command: this is used to run a container from an image by specifying the Image ID or the Repository and/or Tag name.

$ docker run {image}

e.g $ docker run nginx

if it already exists, the command runs an instance <span class="NormalTextRun SpellingErrorV2 SCXW251451022 BCX0">nginx</span>

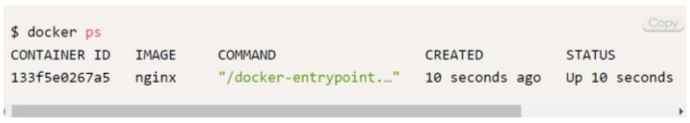
if it does not exist, it goes out to the docker hub (by default) and pulls the image down.

• ps command: this command lists all running containers and some basic information about them.

$ docker ps [option]

In option you can use various flags. To get information about the flags

$docker ps –help



```
$ docker ps
CONTAINER ID   IMAGE    COMMAND                  CREATED          STATUS
133f5e0267a5   nginx    "/docker-entrypoint.…"   10 seconds ago   Up 10 seconds
```

- ls command: like ps command, ls command can also be used for listing containers. –a flag can be used to list all containers.

  $ docker container ls
- stop command: this command is used to stop a running container.

  $ docker stop {container-id}

  After giving stop command can check with ps command, whether the container has stopped.
- rm command: this command removes a stopped or exited container.

  $ docker rm {CONTAINER NAME or ID}
- exec command: this command is used to go inside a running container. This is useful to debug running containers or do other work within a container.

  $ docker exec –it {container} {command}
- logs command:in case acontainer is launched in detached mode and you want to see its logs, you can use logs command.

  $ docker logs {CONTAINER NAME or ID}
- cp command: to copy files between a container and localhost filesystem, this command is used.

  $ docker container cp {CONTAINER NAME or ID:SRC_PATH} {DEST_PATH}|-
- export command: this command exports the filesystem of a container as a TAR file.

```
$ docker container export {CONTAINER NAME or ID}
```

- inspect command: this command gives detailed information about a container.

  ```
  $ docker inspect {CONTAINER NAME or ID}
  ```

  or $ docker container inspect {CONTAINER NAME or ID}
- kill command: this command kills a running container with an option –signal or –s flag. Multiple containers can also be specified to be killed in one go.

  ```
  $ docker kill {CONTAINER NAME or ID} [--signal VAL]
  ```
- stats command: to display a live stream of a container's resource usage, this command can be used.

  ```
  $ docker container stats {CONTAINER NAME or ID}
  ```

Lab Cycle II

Program 7: Develop a simple containerized application using Docker.

Choose any application on which you want to make a container using Docker. Here the file is 'package.json'.

- Create a file named dockerfile in the same folder as the file 'package.json' with the followin contents.

```
# syntax=docker/dockerfile:1
FROM node:12-alpine
RUN apk add --no-cache python2 g++ make
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

- Open a terminal and go to the 'app' directory with the 'Dockerfile'. Now build the containe image using the 'docker build' command.

```
$ docker build -t getting-started
```

This command used the Dockerfile to build a new container image. You might have noticed that a lot of "layers" were downloaded. This is because we instructed the builder that we wanted to start from the node:12-alpine image. But, since we didn't have that on our machine, that image needed to be downloaded.

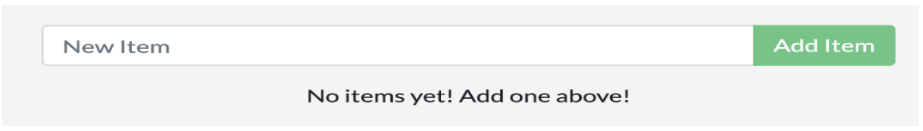
After the image was downloaded, we copied in our application and used yarn to install our application's dependencies. The CMD directive specifies the default command to run when starting a container from this image.

Finally, the -t flag tags our image. Think of this simply as a human-readable name for the final image. Since we named the image getting-started, we can refer to that image when we run a container.

The '.' at the end of the docker build command tells Docker that it should look for the Dockerfile in the current directory.

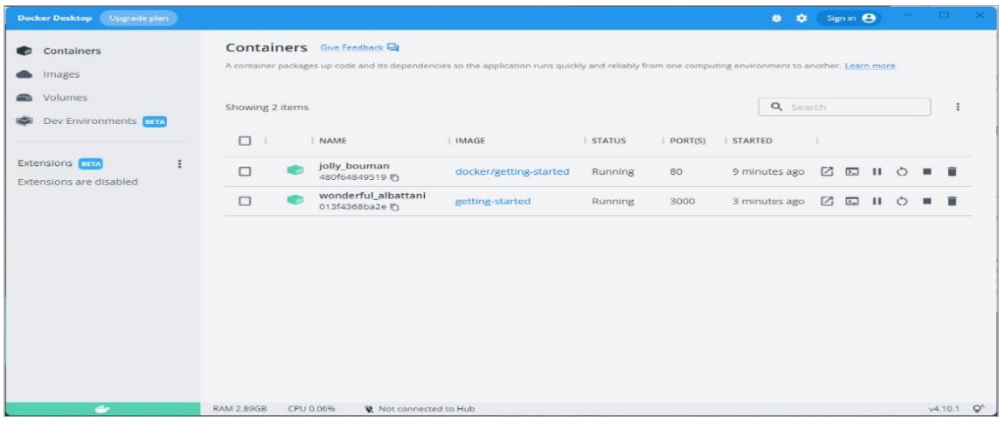
- Start your container using the docker run command and specify the name of the image we just created:
```
$ docker run -dp 3000:3000 getting-started
```
Remember the -d and -p flags? We're running the new container in "detached" mode (in the background) and creating a mapping between the host's port 3000 to the container's port 3000.Without the port mapping, we wouldn't be able to access the application.

- After a few seconds, open your web browser to http://localhost:3000. You should see our app.



No items yet! Add one above!

Go ahead and add an item or two and see that it works as you expect. You can mark items as complete and remove items. Your frontend is successfully storing items in the backend. Now the Docker dashboard will show two containers running.



Extensions are disabled

Containers

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Showing 2 items

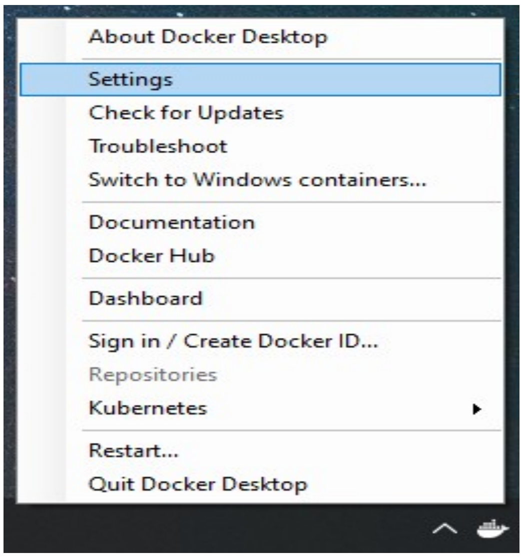
NAME	IMAGE	STATUS	PORT(S)	STARTED
jolly_bouman 480f64849519	docker/getting-started	Running	80	9 minutes ago
wonderful_albattani 013f4368ba2e	getting-started	Running	3000	3 minutes ago

RAM 2.89GB CPU 0.06% Not connected to Hub v4.10.1

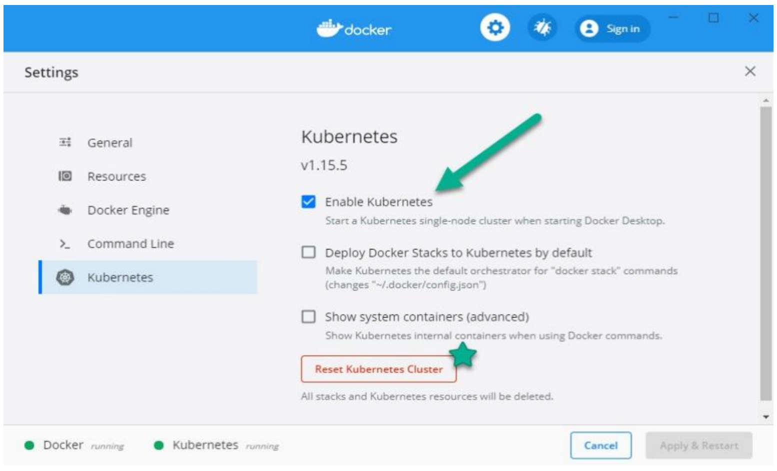
Program 8: Integrate kubernetes and Docker.

Install Docker desktop, enable kubernetes. Kubernetes itself runs in containers. When you deploya Kubenetes cluster you first install Docker (or another container runtime like containerd) and thenuse tools like kubeadm which starts all the Kubernetes components in containers. Docker Desktopdoes all that for you.

Make sure you have Docker Desktop running - in the taskbar in Windows and the menu bar on theMac you'll see Docker's whale logo. Click the whale and select Settings:



Click on Kubernetes and check the Enable Kubernetes checkbox:



Kubernetes
v1.15.5

[✓] Enable Kubernetes
Start a Kubernetes single-node cluster when starting Docker Desktop.

[ ] Deploy Docker Stacks to Kubernetes by default
Make Kubernetes the default orchestrator for "docker stack" commands (changes "~/.docker/config.json")

[ ] Show system containers (advanced)
Show Kubernetes internal containers when using Docker commands.

All stacks and Kubernetes resources will be deleted.

● Docker running ● Kubernetes running

Verify your Kubernetes cluster: like Docker uses 'docker' and 'docker-compose' commands tomanage containers, kubernetes uses tool 'kubect1' to manage apps. Docker desktop installskubect1 too.

Check the state of Docker desktop cluster:

```
kubectl get nodes
```

Program 9: Automate the process of running containerized application developed in exercise 7 using Kubernetes.

Steps to be done :

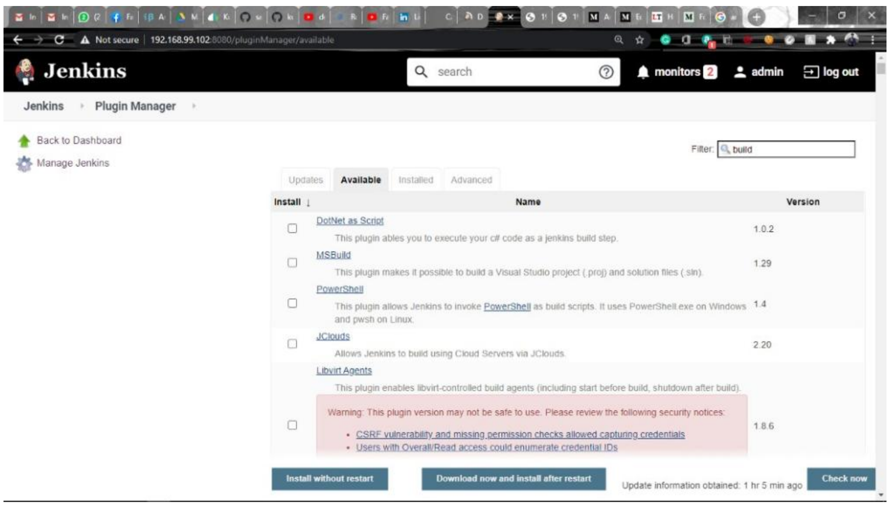
✓ Create a job chain of Job1, Job2, Job3, Job4 using build pipeline plugin in Jenkins.

✓ Job1: pull the github repo automatically when some developers push the repo to GitHub.

✓ Job2: automatically Jenkins should start respective language interpreter installed image container to deploy code on top of Kubernetes; expose your pod so the testing can be done; make the data to remain persistent.

✓ Test your app if it is working.

- Load your Jenkins server and restart the services.
  $ systemctl restart Jenkins
- Creating jobs pipeline. Install Build plugin in Jenkins by going to Manage Jenkins → Manage plugins → Available → Search for build plugin, install and restart.



Not secure | 192.168.99.102:8080/pluginManager/available

Jenkins › Plugin Manager

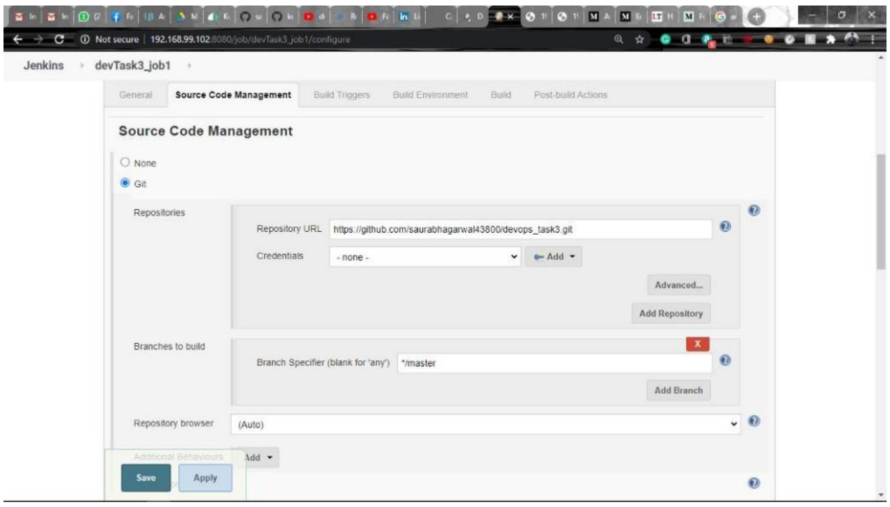
Filter: build

Updates | Available | Installed | Advanced

Install	Name	Version
☐	DotNet as Script — This plugin ables you to execute your c# code as a jenkins build step.	1.0.2
☐	MSBuild — This plugin makes it possible to build a Visual Studio project (.proj) and solution files (.sln).	1.29
☐	PowerShell — This plugin allows Jenkins to invoke PowerShell as build scripts. It uses PowerShell.exe on Windows and pwsh on Linux.	1.4
☐	JClouds — Allows Jenkins to build using Cloud Servers via JClouds.	2.20
☐	Libvirt Agents — This plugin enables libvirt-controlled build agents (including start before build, shutdown after build). Warning: This plugin version may not be safe to use. Please review the following security notices: • CSRF vulnerability and missing permission checks allowed capturing credentials • Users with Overall/Read access could enumerate credential IDs	1.8.6

Update information obtained: 1 hr 5 min ago

- Create job 1 to pull the GitHub repo automatically. Select SCM as Git and provide the URL of GitHub repo which will help this job to pull the code from GitHub repo.



Not secure | 192.168.99.102:8080/job/devTask3_job1/configure

Jenkins › devTask3_job1

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

Source Code Management

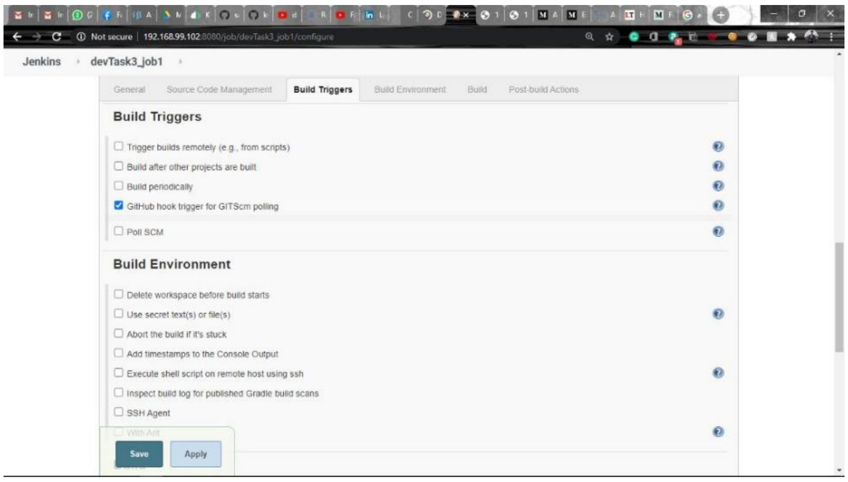
None [✓ Git]

Repositories
Repository URL: https://github.com/saurabhagarwal43800/devops_task3.git
Credentials: - none -

Branches to build
Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

- In Build Trigger, select GitHub hook trigger which makes the job pull the code only when there are some changes made in GitHub repo.



Jenkins › devTask3_job1

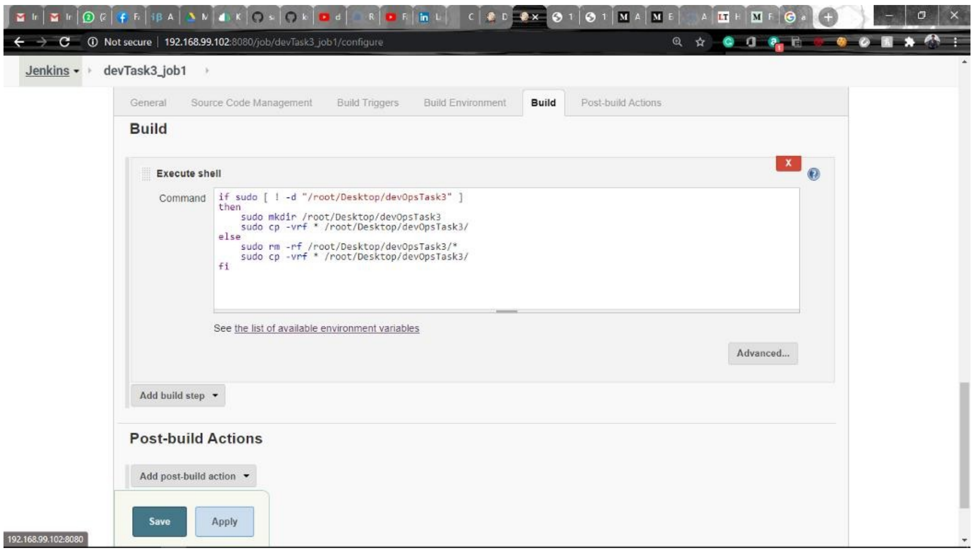
Build Triggers

- ☐ 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
- ☐ Execute shell script on remote host using ssh
- ☐ Inspect build log for published Gradle build scans
- ☐ SSH Agent

- In Build, select Execute shell to perform following commands which will create a directory in the container and copy all the files from the GitHub repo to this directory.



Jenkins › devTask3_job1

Build

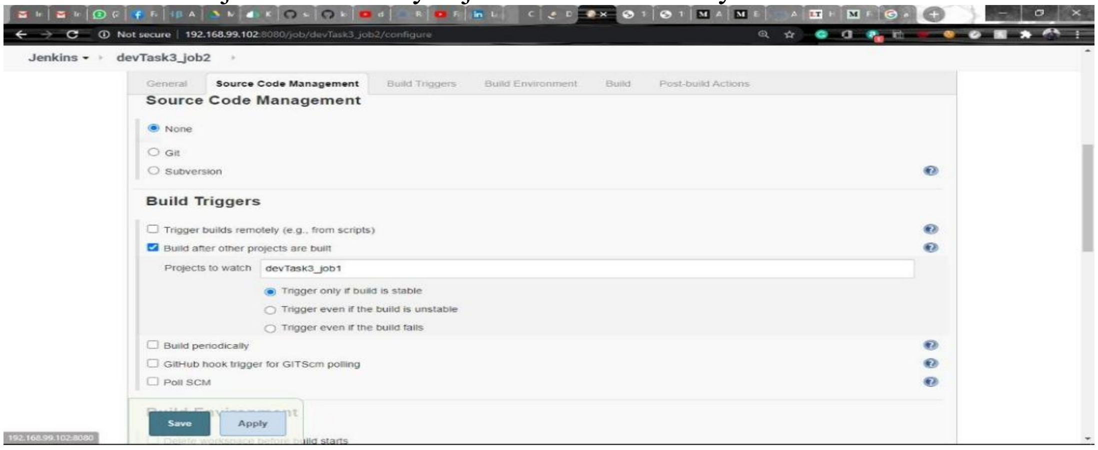
Execute shell

Command

```
if sudo [ ! -d "/root/Desktop/devOpsTask3" ]
then
    sudo mkdir /root/Desktop/devOpsTask3
    sudo cp -vrf * /root/Desktop/devOpsTask3/
else
    sudo rm -rf /root/Desktop/devOpsTask3/*
    sudo cp -vrf * /root/Desktop/devOpsTask3/
fi
```

See the list of available environment variables

- Create job 2 i.e. by looking at the program, the respective interpreter should automatically start. In Build Triggers, select Build after other projects are built and write the name of job1 which makes the job2 to run only if job1 ran successfully.



Jenkins › devTask3_job2

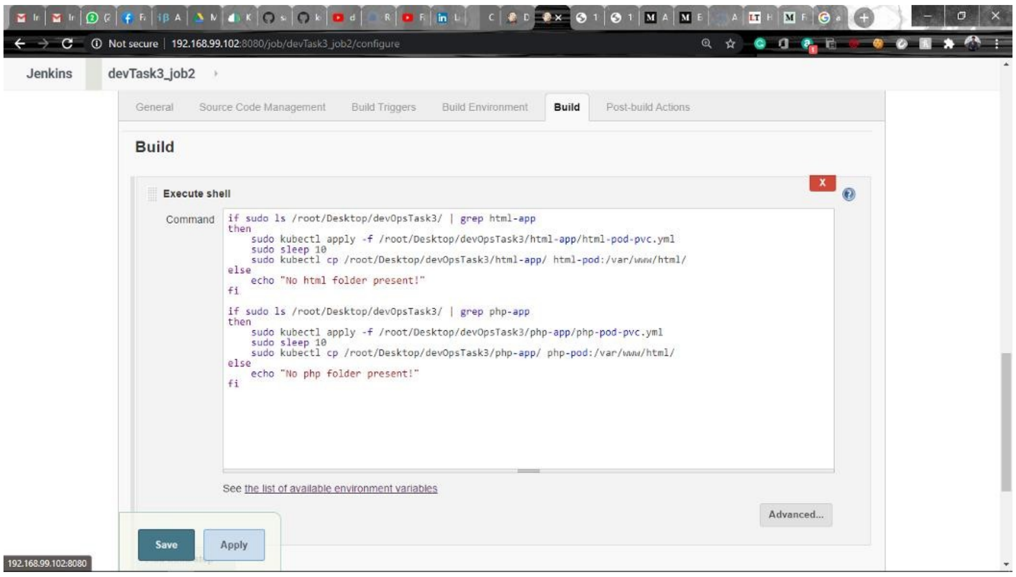
Source Code Management

- ● None
- ○ Git
- ○ Subversion

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☑ Build after other projects are built
  - Projects to watch: devTask3_job1
  - ● Trigger only if build is stable
  - ○ Trigger even if the build is unstable
  - ○ Trigger even if the build fails
- ☐ Build periodically
- ☐ GitHub hook trigger for GITScm polling
- ☐ Poll SCM

- In Build, select Execute shell for performing this script which will automatically start the respective language interpreter installed image container to deploy code on top of Kubernetes with persistent storage and also exposing the pod for testing.



Not secure | 192.168.99.102:8080/job/devTask3_job2/configure

Jenkins › devTask3_job2 ›

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

Build

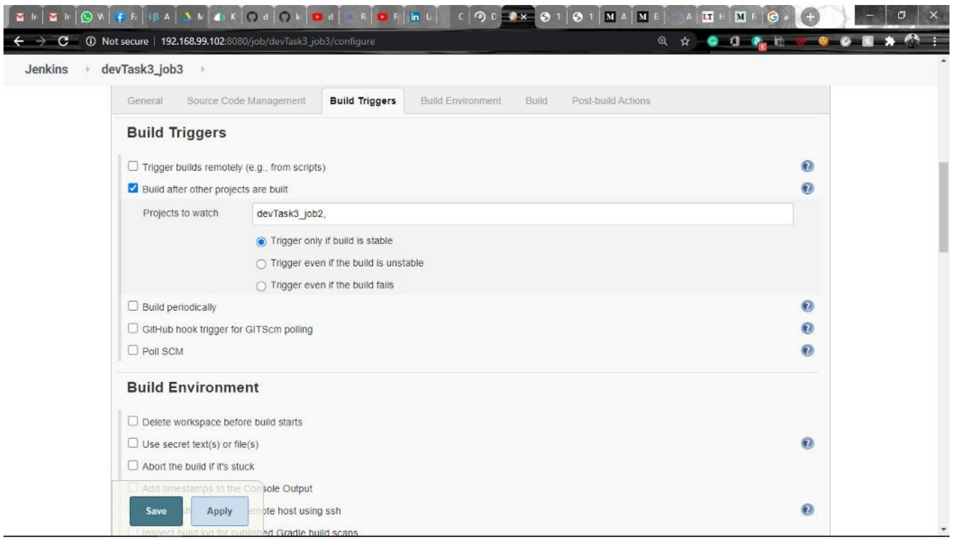
Execute shell

Command

```
if sudo ls /root/Desktop/devOpsTask3/ | grep html-app
then
    sudo kubectl apply -f /root/Desktop/devOpsTask3/html-app/html-pod-pvc.yml
    sudo sleep 10
    sudo kubectl cp /root/Desktop/devOpsTask3/html-app/ html-pod:/var/www/html/
else
    echo "No html folder present!"
fi

if sudo ls /root/Desktop/devOpsTask3/ | grep php-app
then
    sudo kubectl apply -f /root/Desktop/devOpsTask3/php-app/php-pod-pvc.yml
    sudo sleep 10
    sudo kubectl cp /root/Desktop/devOpsTask3/php-app/ php-pod:/var/www/html/
else
    echo "No php folder present!"
fi
```

See the list of available environment variables

- Create job3 to test the app, if the app is not working send email to the developer with error message.
- In Build Triggers, select Build after other projects are built and write the name of job2 which makes the job3 to run only if job2 ran successfully.



Not secure | 192.168.99.102:8080/job/devTask3_job3/configure

Jenkins › devTask3_job3 ›

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

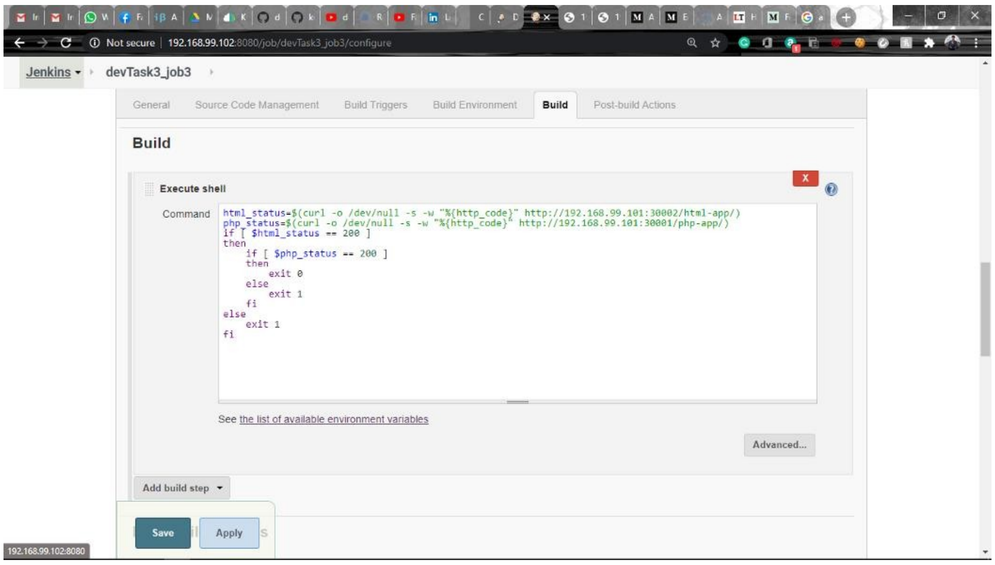
Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [x] Build after other projects are built
  - Projects to watch: devTask3_job2,
  - (●) Trigger only if build is stable
  - ( ) Trigger even if the build is unstable
  - ( ) Trigger even if the build fails
- [ ] 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
- [ ] ...te host using ssh
- [ ] Inspect build log for published Gradle build scans

- In Build, select Execute shell for performing this script which will test the websites that they Are running properly or not.



192.168.99.102:8080/job/devTask3_job3/configure

Jenkins › devTask3_job3

Build

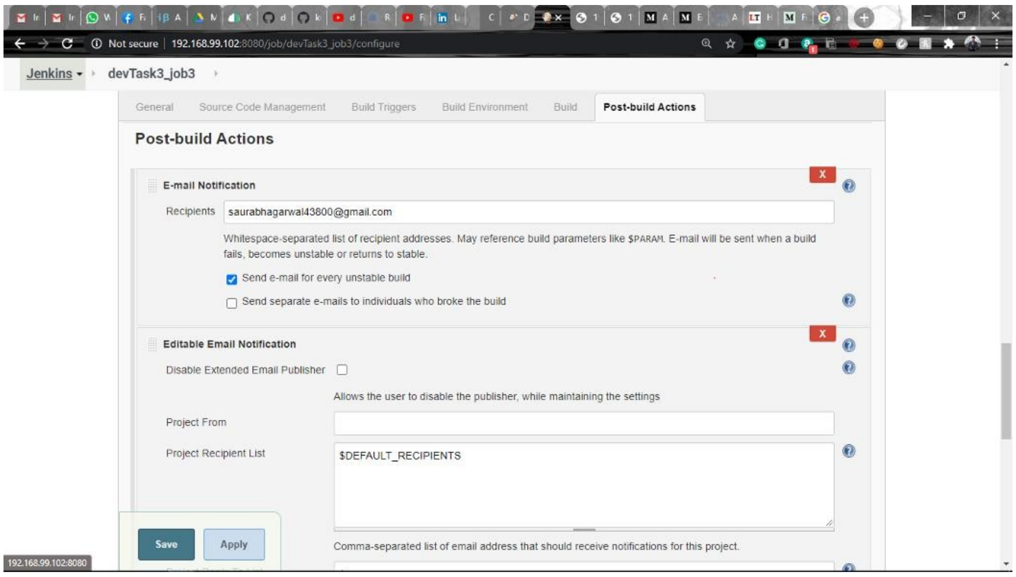
Execute shell

Command

```
html_status=$(curl -o /dev/null -s -w "%{http_code}" http://192.168.99.101:30002/html-app/)
php_status=$(curl -o /dev/null -s -w "%{http_code}" http://192.168.99.101:30001/php-app/)
if [ $html_status == 200 ]
then
    if [ $php_status == 200 ]
    then
        exit 0
    else
        exit 1
    fi
else
    exit 1
fi
```

See the list of available environment variables

- In Post-build Actions, select E-mail notifications and write e-mail address, to notify the failure.



192.168.99.102:8080/job/devTask3_job3/configure

Jenkins › devTask3_job3

Post-build Actions

E-mail Notification

Recipients: saurabhagarwal43800@gmail.com

Whitespace-separated list of recipient addresses. May reference build parameters like $PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.

[✓] Send e-mail for every unstable build

[ ] Send separate e-mails to individuals who broke the build

Editable Email Notification

Disable Extended Email Publisher [ ]

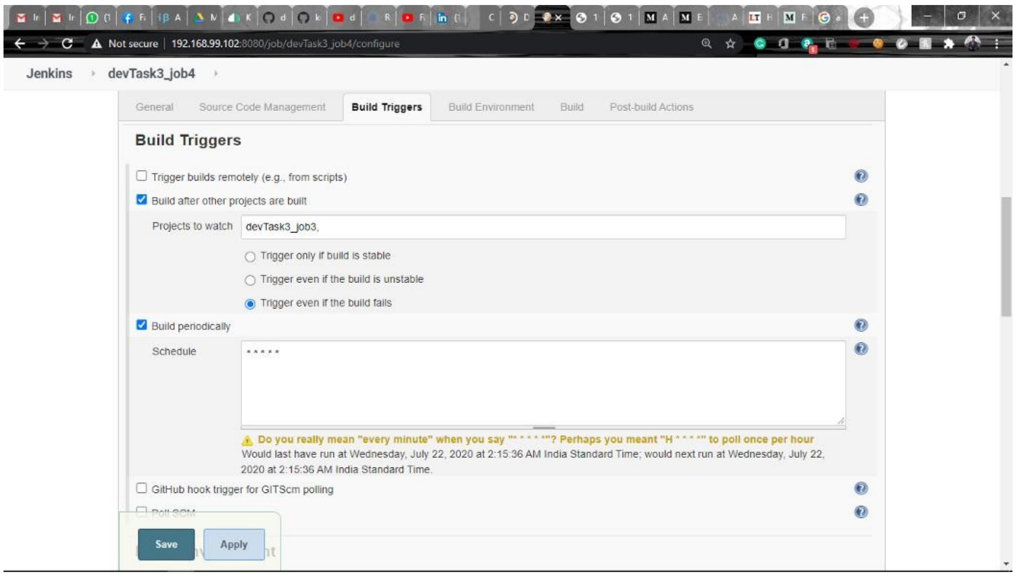
Allows the user to disable the publisher, while maintaining the settings

Project From:

Project Recipient List: $DEFAULT_RECIPIENTS

Comma-separated list of email address that should receive notifications for this project.

- Create job 4 which will monitor and redeploy the application, after the code is being edited by the developer if the application fails. In Build Triggers, select build after other projects are built so that this will run after job3 when it is successfully built and also select Build periodically so that it monitors the application every minute.



Not secure | 192.168.99.102:8080/job/devTask3_job4/configure

Jenkins › devTask3_job4 ›

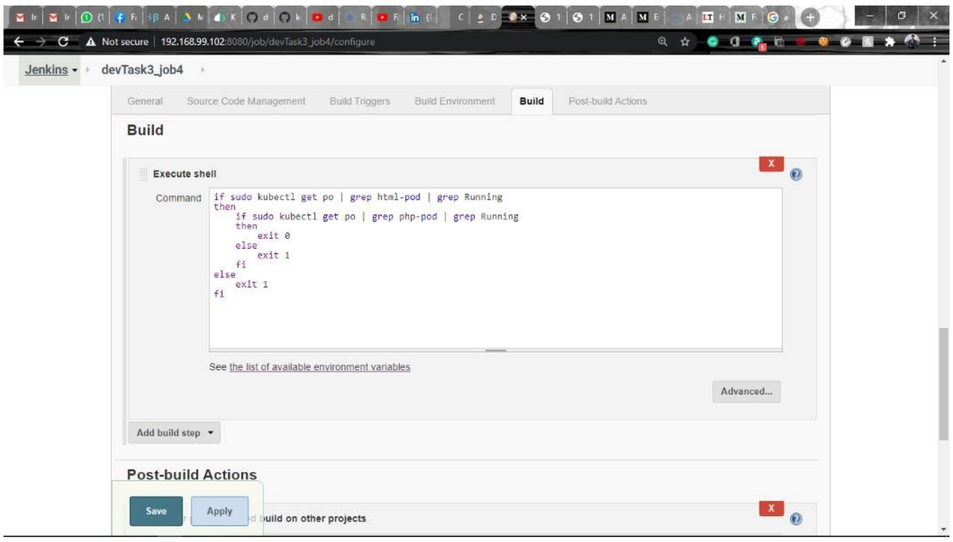
General | Source Code Management | **Build Triggers** | Build Environment | Build | Post-build Actions

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☑ Build after other projects are built
  - Projects to watch: devTask3_job3,
  - ○ Trigger only if build is stable
  - ○ Trigger even if the build is unstable
  - ● Trigger even if the build fails
- ☑ Build periodically
  - Schedule: `* * * * *`
  - ⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour
  - Would last have run at Wednesday, July 22, 2020 at 2:15:36 AM India Standard Time; would next run at Wednesday, July 22, 2020 at 2:15:36 AM India Standard Time.
- ☐ GitHub hook trigger for GITScm polling
- ☐ Poll SCM

Save | Apply

- In Build, select Execute shell to perform the following script so that it will continuously monitor if the pods are running or not and if fails it will make the job state as a failure.



Not secure | 192.168.99.102:8080/job/devTask3_job4/configure

Jenkins › devTask3_job4 ›

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

Build

Execute shell

Command

```
if sudo kubectl get po | grep html-pod | grep Running
then
    if sudo kubectl get po | grep php-pod | grep Running
    then
        exit 0
    else
        exit 1
    fi
else
    exit 1
fi
```

See the list of available environment variables

Advanced...

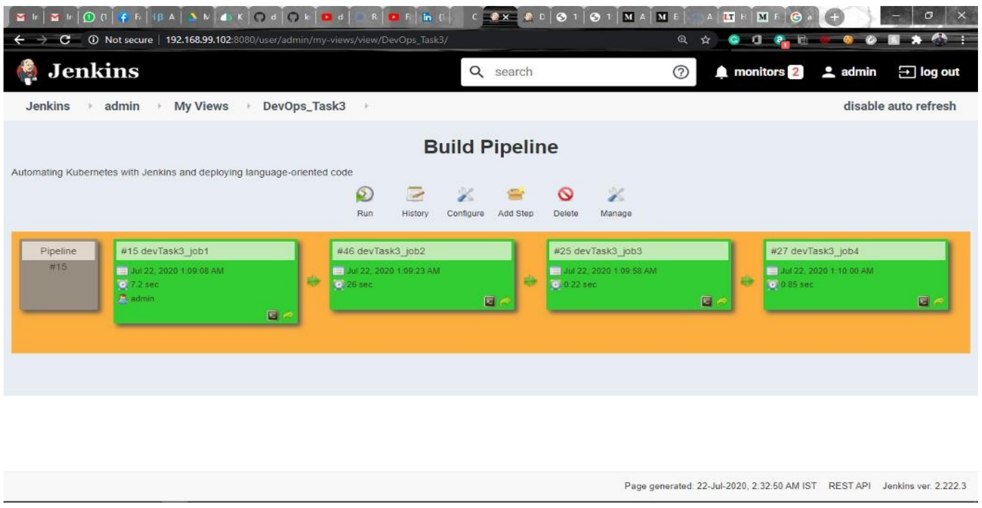
Add build step

Post-build Actions

build on other projects

Save | Apply

- In Post-build Actions, select Trigger parameterized build on other projects so that if the current job fails or unstable then it will automatically run the job2 to redeploy the application.


Not secure | 192.168.99.102:8080/user/admin/my-views/view/DevOps_Task3/

Build Pipeline

Automating Kubernetes with Jenkins and deploying language-oriented code

Pipeline #15

#15 devTask3_job1
Jul 22, 2020 1:09:08 AM
7.2 sec
admin

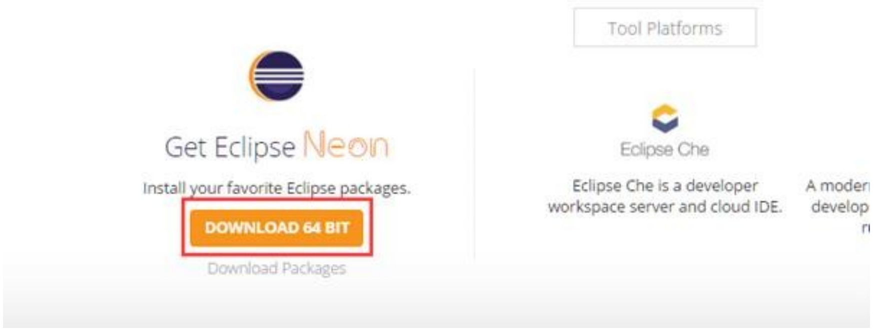
#46 devTask3_job2
Jul 22, 2020 1:09:23 AM
26 sec

#25 devTask3_job3
Jul 22, 2020 1:09:58 AM
0.22 sec

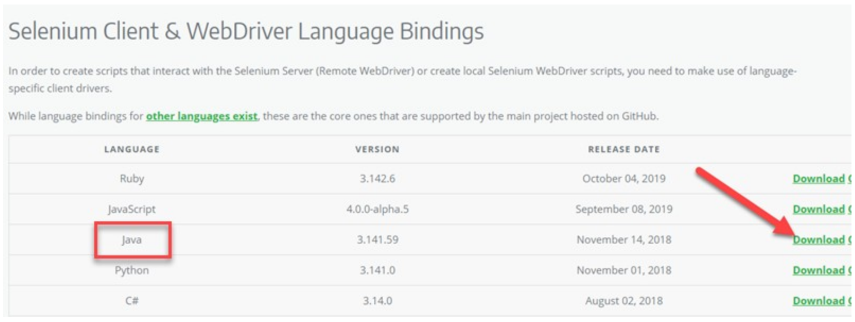
#27 devTask3_job4
Jul 22, 2020 1:10:00 AM
0.85 sec

Page generated: 22-Jul-2020, 2:32:50 AM IST REST API Jenkins ver. 2.222.3

Program 10: Install and explore Selenium for automated testing.
Prerequisites: Java development kit.
Install Eclipse IDE for java developers. https://www.eclipse.org/downloads/


Tool Platforms
Get Eclipse Neon
Install your favorite Eclipse packages.
DOWNLOAD 64 BIT
Download Packages
Eclipse Che
Eclipse Che is a developer workspace server and cloud IDE.
A modern develop

Download the selenium java client driver. https://www.selenium.dev/downloads/



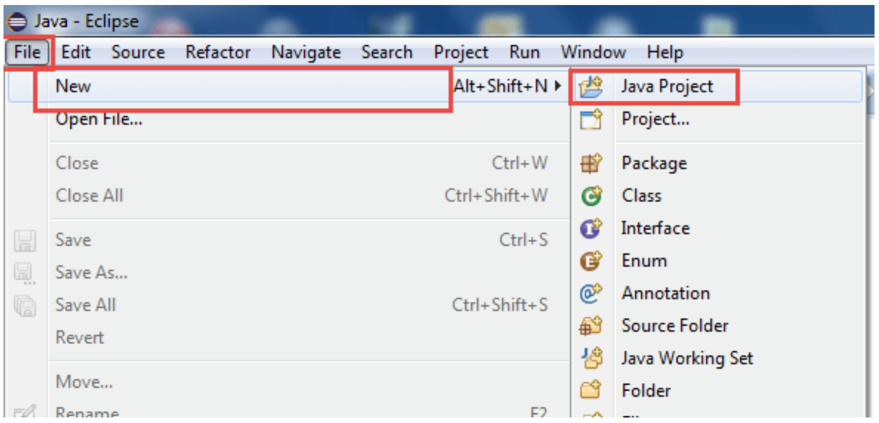
Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers.

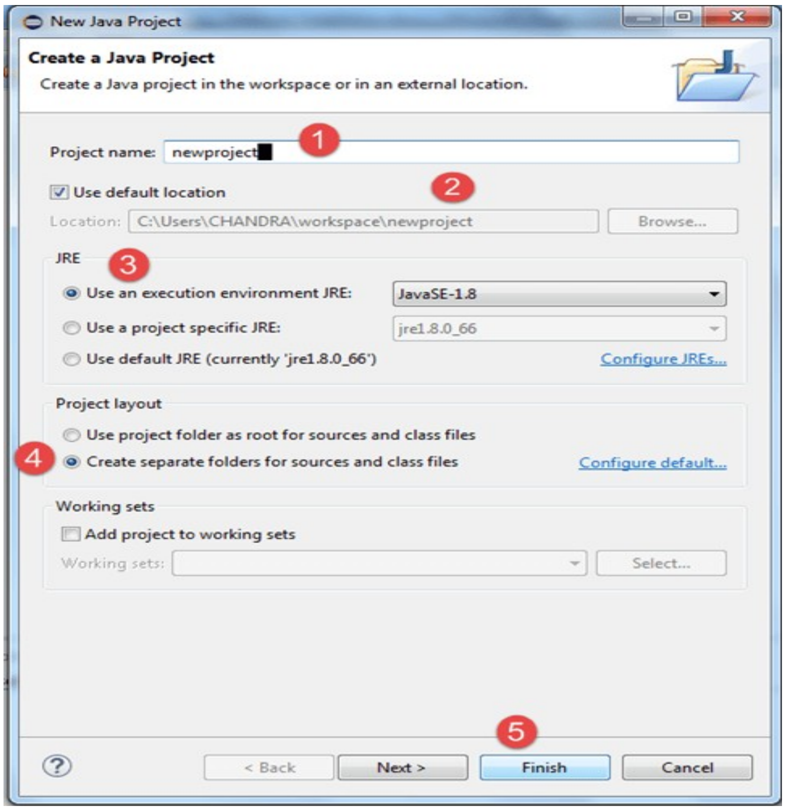
While language bindings for other languages exist, these are the core ones that are supported by the main project hosted on GitHub.

LANGUAGE	VERSION	RELEASE DATE	
Ruby	3.142.6	October 04, 2019	Download
JavaScript	4.0.0-alpha.5	September 08, 2019	Download
Java	3.141.59	November 14, 2018	Download
Python	3.141.0	November 01, 2018	Download
C#	3.14.0	August 02, 2018	Download

Configure Eclipse IDE with WebDriver.
Launch eclipse.exe, create a new project through File>New>Java Project. Name the project as "newproject".



Enter project name, location to save project, select an execution JRE, select layout project option, click on Finish button.


New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: newproject

☑ Use default location

Location: C:\Users\CHANDRA\workspace\newproject

JRE

● Use an execution environment JRE: JavaSE-1.8

○ Use a project specific JRE: jre1.8.0_66

○ Use default JRE (currently 'jre1.8.0_66')

Project layout

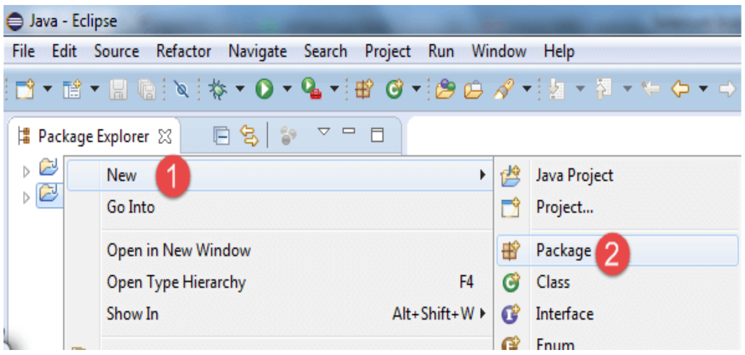
○ Use project folder as root for sources and class files

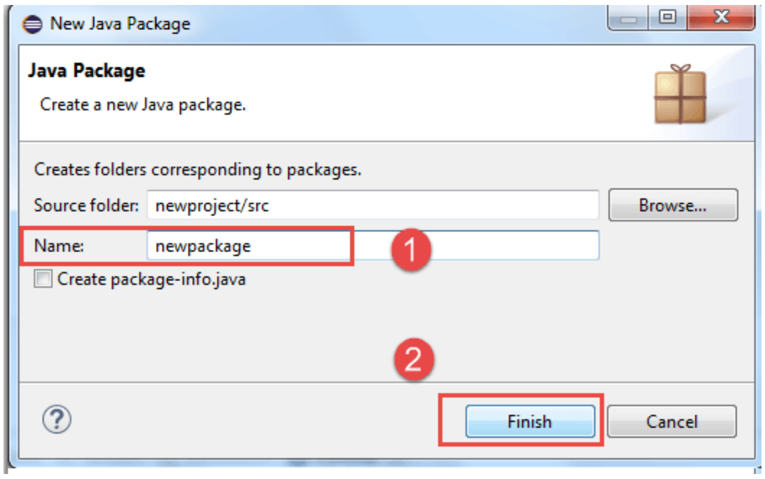
● Create separate folders for sources and class files

Working sets

☐ Add project to working sets

Make new package as “newpackage”.




New Java Package

Java Package

Create a new Java package.

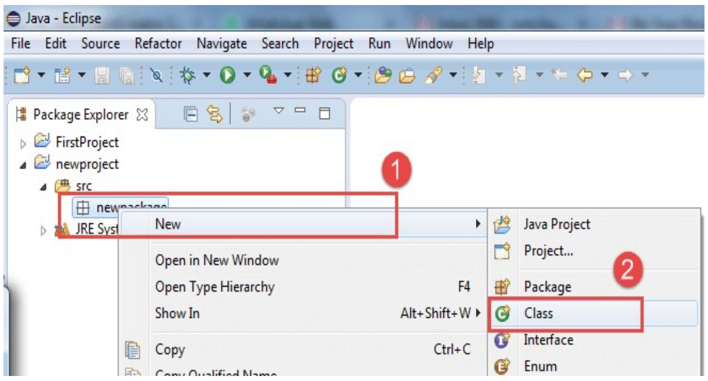
Creates folders corresponding to packages.

Source folder: newproject/src

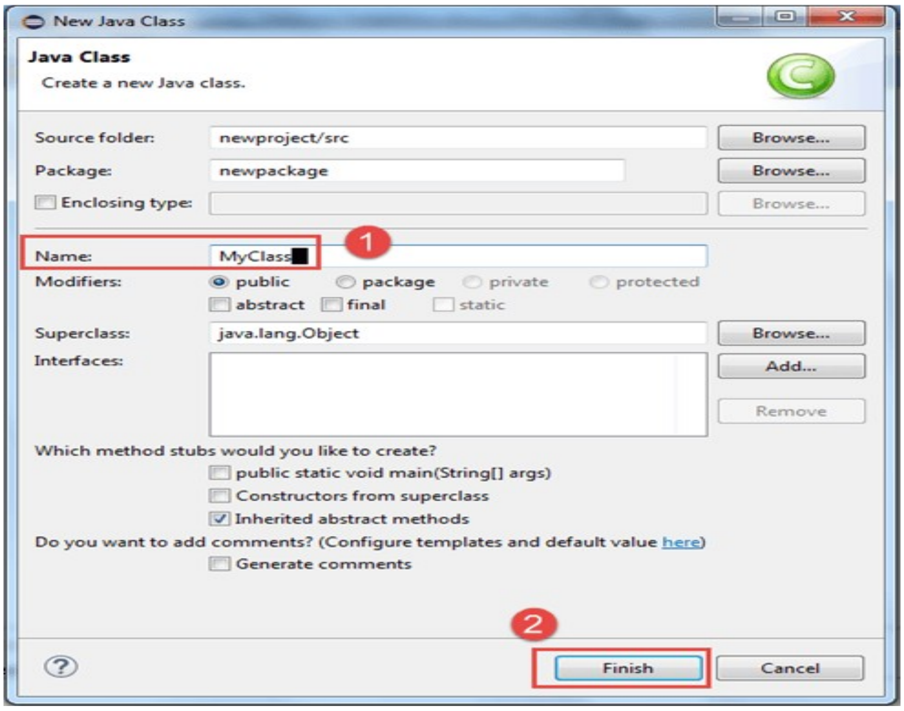
Name: newpackage

☐ Create package-info.java

- Create a new Java class under newpackage by right-clicking on it and then selecting- New > Class,and then name it as “MyClass”. Your Eclipse IDE should look like the image below.



Give name of the class, click on finish button.



Source folder: newproject/src

Package: newpackage

Enclosing type: [ ]

Name: MyClass

Modifiers: [✓ public] package private protected

[ ] abstract [ ] final [ ] static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

[ ] public static void main(String[] args)

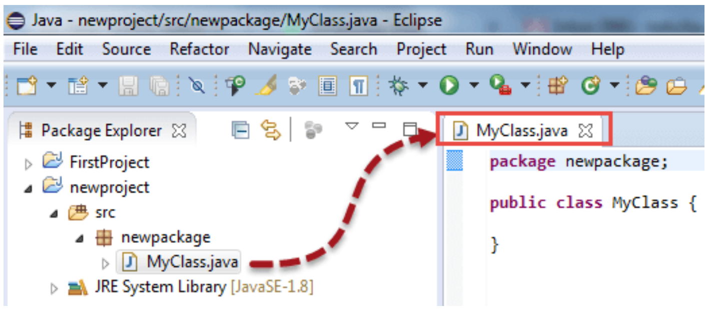
[ ] Constructors from superclass

[✓] Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

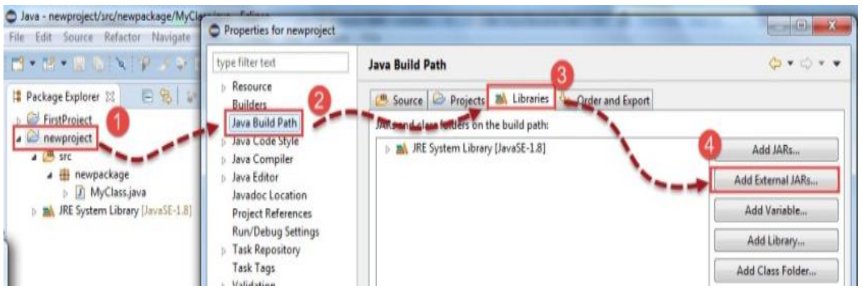
[ ] Generate comments

After creating the class:

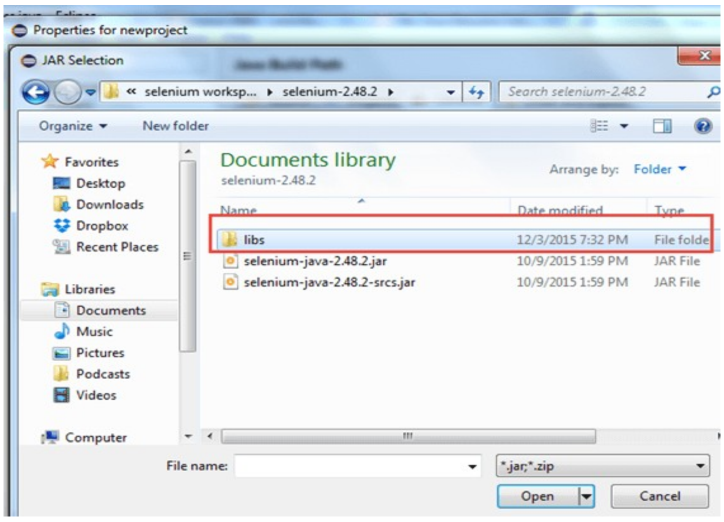

Java - newproject/src/newpackage/MyClass.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
FirstProject
newproject
src
newpackage
MyClass.java
JRE System Library [JavaSE-1.8]
MyClass.java

```
package newpackage;

public class MyClass {

}
```

Now selenium webdriver's into Java Build Path.

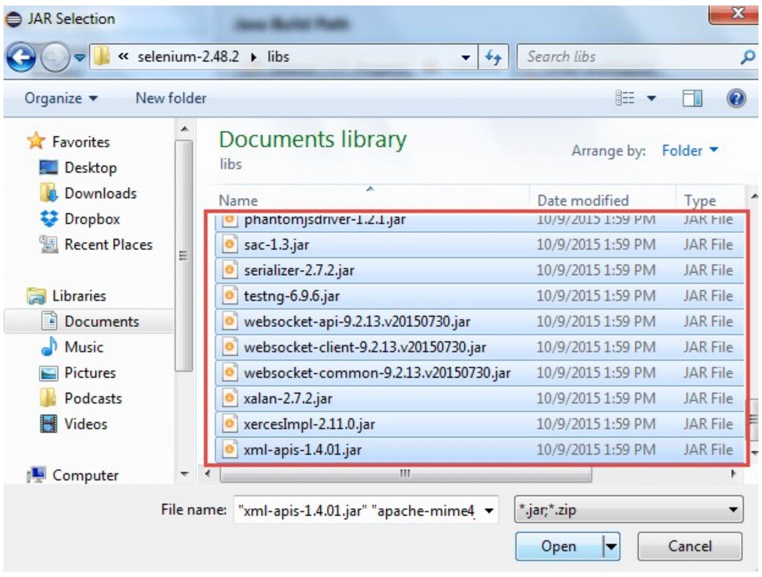


- When you click on "Add External JARs.." It will open a pop-up window. Select the JAR files you want to add.



Name	Date modified	Type
libs	12/3/2015 7:32 PM	File folde
selenium-java-2.48.2.jar	10/9/2015 1:59 PM	JAR File
selenium-java-2.48.2-srcs.jar	10/9/2015 1:59 PM	JAR File

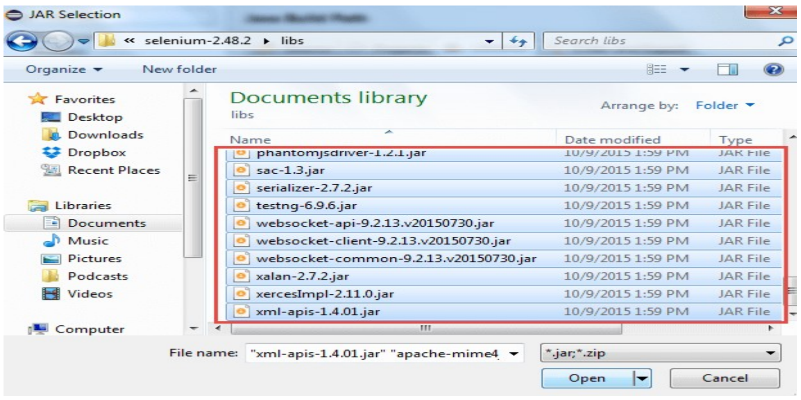
Select all files inside the lib folder.



Name	Date modified	Type
phantomjsdriver-1.2.1.jar	10/9/2015 1:59 PM	JAR File
sac-1.3.jar	10/9/2015 1:59 PM	JAR File
serializer-2.7.2.jar	10/9/2015 1:59 PM	JAR File
testng-6.9.6.jar	10/9/2015 1:59 PM	JAR File
websocket-api-9.2.13.v20150730.jar	10/9/2015 1:59 PM	JAR File
websocket-client-9.2.13.v20150730.jar	10/9/2015 1:59 PM	JAR File
websocket-common-9.2.13.v20150730.jar	10/9/2015 1:59 PM	JAR File
xalan-2.7.2.jar	10/9/2015 1:59 PM	JAR File
xercesImpl-2.11.0.jar	10/9/2015 1:59 PM	JAR File
xml-apis-1.4.01.jar	10/9/2015 1:59 PM	JAR File

File name: "xml-apis-1.4.01.jar" "apache-mime4
.jar;.zip

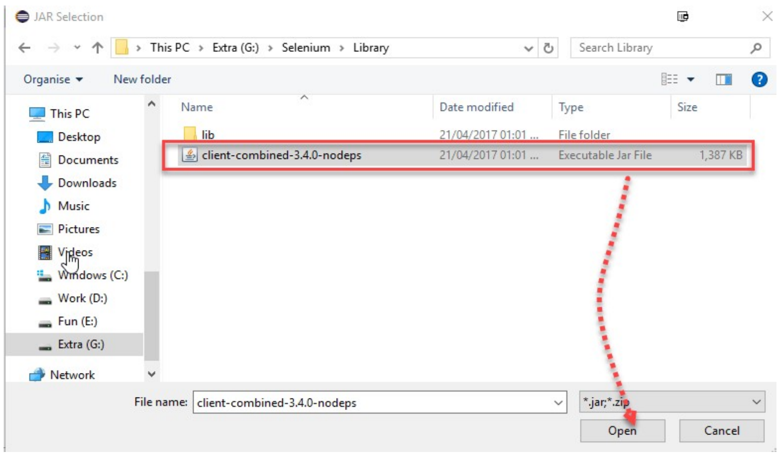
Select all files inside the lib folder.



Name	Date modified	Type
phantomjsdriver-1.2.1.jar	10/9/2015 1:59 PM	JAR File
sac-1.3.jar	10/9/2015 1:59 PM	JAR File
serializer-2.7.2.jar	10/9/2015 1:59 PM	JAR File
testng-6.9.6.jar	10/9/2015 1:59 PM	JAR File
websocket-api-9.2.13.v20150730.jar	10/9/2015 1:59 PM	JAR File
websocket-client-9.2.13.v20150730.jar	10/9/2015 1:59 PM	JAR File
websocket-common-9.2.13.v20150730.jar	10/9/2015 1:59 PM	JAR File
xalan-2.7.2.jar	10/9/2015 1:59 PM	JAR File
xercesImpl-2.11.0.jar	10/9/2015 1:59 PM	JAR File
xml-apis-1.4.01.jar	10/9/2015 1:59 PM	JAR File

File name: "xml-apis-1.4.01.jar" "apache-mime4
.jar;.zip

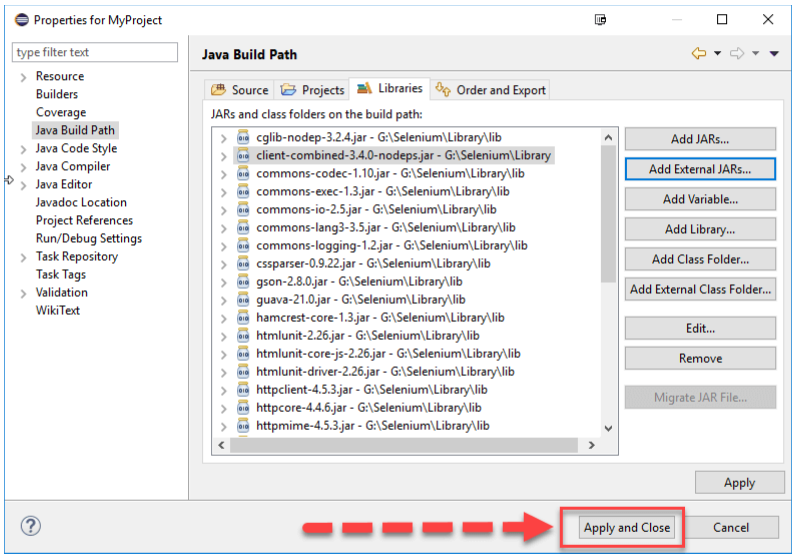
Select files outside lib folder.



Name	Date modified	Type	Size
lib	21/04/2017 01:01 ...	File folder	
client-combined-3.4.0-nodeps	21/04/2017 01:01 ...	Executable Jar File	1,387 KB

File name: client-combined-3.4.0-nodeps
.jar;.zip

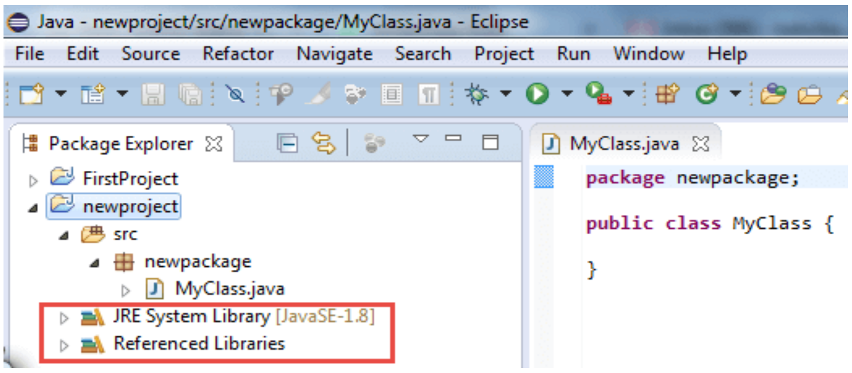
Once done, click "Apply and Close" button.



JARs and class folders on the build path:

```
cglib-nodep-3.2.4.jar - G:\Selenium\Library\lib
client-combined-3.4.0-nodeps.jar - G:\Selenium\Library
commons-codec-1.10.jar - G:\Selenium\Library\lib
commons-exec-1.3.jar - G:\Selenium\Library\lib
commons-io-2.5.jar - G:\Selenium\Library\lib
commons-lang3-3.5.jar - G:\Selenium\Library\lib
commons-logging-1.2.jar - G:\Selenium\Library\lib
cssparser-0.9.22.jar - G:\Selenium\Library\lib
gson-2.8.0.jar - G:\Selenium\Library\lib
guava-21.0.jar - G:\Selenium\Library\lib
hamcrest-core-1.3.jar - G:\Selenium\Library\lib
htmlunit-2.26.jar - G:\Selenium\Library\lib
htmlunit-core-js-2.26.jar - G:\Selenium\Library\lib
htmlunit-driver-2.26.jar - G:\Selenium\Library\lib
httpclient-4.5.3.jar - G:\Selenium\Library\lib
httpcore-4.4.6.jar - G:\Selenium\Library\lib
httpmime-4.5.3.jar - G:\Selenium\Library\lib
```

Add all the JAR files inside and outside the “libs” folder. Your Properties dialog should now looksimilar to the image below.


Java - newproject/src/newpackage/MyClass.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
FirstProject
newproject
src
newpackage
MyClass.java
JRE System Library [JavaSE-1.8]
Referenced Libraries
MyClass.java

```
package newpackage;

public class MyClass {

}
```

Finally, click OK and we are done importing Selenium libraries into our project.

Program 11: Write a simple program in Javascript and perform testing using Selenium.

Program for Calculator in javascript:

```
"use strict";
var input = document.getElementById('input'), // input/output button
number = document.querySelectorAll('.numbers div'), // number buttons
operator = document.querySelectorAll('.operators div'), // operator buttons
result = document.getElementById('result'), // equal button
clear = document.getElementById('clear'), // clear button
resultDisplayed = false; // flag to keep an eye on what output is displayed
// adding click handlers to number buttons
for (var i = 0; i < number.length; i++) {
number[i].addEventListener("click", function(e) {
// storing current input string and its last character in variables - used later
var currentString = input.innerHTML;
var lastChar = currentString[currentString.length - 1];
// if result is not diplayed, just keep adding
if (resultDisplayed === false) {
input.innerHTML += e.target.innerHTML;
} else if (resultDisplayed === true && lastChar === "+" || lastChar === "-" || lastChar ===
"×" || lastChar === "÷") {
// if result is currently displayed and user pressed an operator
// we need to keep on adding to the string for next operation
resultDisplayed = false;
input.innerHTML += e.target.innerHTML;
} else {
// if result is currently displayed and user pressed a number
// we need clear the input string and add the new input to start the new opration
resultDisplayed = false;
input.innerHTML = "";
input.innerHTML += e.target.innerHTML;
}
});
}
// adding click handlers to number buttons
for (var i = 0; i < operator.length; i++) {
operator[i].addEventListener("click", function(e) {
// storing current input string and its last character in variables - used later
var currentString = input.innerHTML;
var lastChar = currentString[currentString.length - 1];
// if last character entered is an operator, replace it with the currently pressed one
if (lastChar === "+" || lastChar === "-" || lastChar === "×" || lastChar === "÷") {
var newString = currentString.substring(0, currentString.length - 1) + e.target.innerHTML;
input.innerHTML = newString;
} else if (currentString.length == 0) {
// if first key pressed is an opearator, don't do anything
console.log("enter a number first");
} else {
// else just add the operator pressed to the input
input.innerHTML += e.target.innerHTML;
}
});
}
// on click of 'equal' button
result.addEventListener("click", function() {
// this is the string that we will be processing eg. -10+26+33-56*34/23
var inputString = input.innerHTML;
// forming an array of numbers. eg for above string it will be: numbers = ["10", "26", "33",
"56", "34", "23"]
var numbers = inputString.split(/\+|\-|\×|\÷/g);
// forming an array of operators. for above string it will be: operators = ["+", "+", "-", "*", "/"]
// first we replace all the numbers and dot with empty string and then split
var operators = inputString.replace(/[0-9]|\./g, "").split("");
console.log(inputString);
console.log(operators);
console.log(numbers);
console.log("----------------------------");
// now we are looping through the array and doing one operation at a time.

// first divide, then multiply, then subtraction and then addition
// as we move we are alterning the original numbers and operators array
// the final element remaining in the array will be the output
var divide = operators.indexOf("÷");
while (divide != -1) {
numbers.splice(divide, 2, numbers[divide] / numbers[divide + 1]);
operators.splice(divide, 1);
divide = operators.indexOf("÷");
}
var multiply = operators.indexOf("×");
while (multiply != -1) {
numbers.splice(multiply, 2, numbers[multiply] * numbers[multiply + 1]);
operators.splice(multiply, 1);
multiply = operators.indexOf("×");
}
```

```
var subtract = operators.indexOf("-");
while (subtract != -1) {
numbers.splice(subtract, 2, numbers[subtract] - numbers[subtract + 1]);
operators.splice(subtract, 1);
subtract = operators.indexOf("-");
}
var add = operators.indexOf("+");
while (add != -1) {
// using parseFloat is necessary, otherwise it will result in string concatenation :)
numbers.splice(add, 2, parseFloat(numbers[add]) + parseFloat(numbers[add + 1]));
operators.splice(add, 1);
add = operators.indexOf("+");
}
input.innerHTML = numbers[0]; // displaying the output
resultDisplayed = true; // turning flag if result is displayed
});
// clearing the input on press of clear
clear.addEventListener("click", function() {
input.innerHTML = "";
})
```

Save as calculator.js
Then, write the test case to check the basic functionality of addition, subtraction, division and multiplication.

```
from selenium import webdriver
import unittest
class CalculatorTest(unittest.TestCase):
def setUp(self):
#create a headless Chrome browser
op = webdriver.ChromeOptions()
op.add_argument('headless')
self.driver =
webdriver.Chrome("/Users/nabeel/Documents/selenium/chromedriver73", options=op)
def test_addition(self):
self.driver.get("file:///Users/nabeel/Documents/selenium/cal/calculator.html")
self.driver.find_element_by_name("5").click()
self.driver.find_element_by_name("+").click()
self.driver.find_element_by_name("8").click()
self.driver.find_element_by_name("=").click()
```

```
value = self.driver.find_element_by_id("result").get_attribute("value")
self.assertEqual('13', value)

def test_substraction(self):
self.driver.get("file:///Users/nabeel/Documents/selenium/cal/calculator.html")
self.driver.find_element_by_name("5").click()
self.driver.find_element_by_name("-").click()
self.driver.find_element_by_name("8").click()
self.driver.find_element_by_name("=").click()
value = self.driver.find_element_by_id("result").get_attribute("value")
self.assertEqual('-3', value)
def test_division(self):

self.driver.get("file:///Users/nabeel/Documents/selenium/cal/calculator.html")
self.driver.find_element_by_name("8").click()
self.driver.find_element_by_name("/").click()
self.driver.find_element_by_name("2").click()
self.driver.find_element_by_name("=").click()
value = self.driver.find_element_by_id("result").get_attribute("value")
self.assertEqual('4', value)
def test_multiplication(self):
self.driver.get("file:///Users/nabeel/Documents/selenium/cal/calculator.html")
self.driver.find_element_by_name("5").click()
self.driver.find_element_by_name("*").click()
self.driver.find_element_by_name("8").click()
self.driver.find_element_by_name("=").click()
value = self.driver.find_element_by_id("result").get_attribute("value")
self.assertEqual('40', value)
def tearDown(self):
#close the browser window
self.driver.quit()


if __name__ == "__main__":
unittest.main(verbosity=2)
```

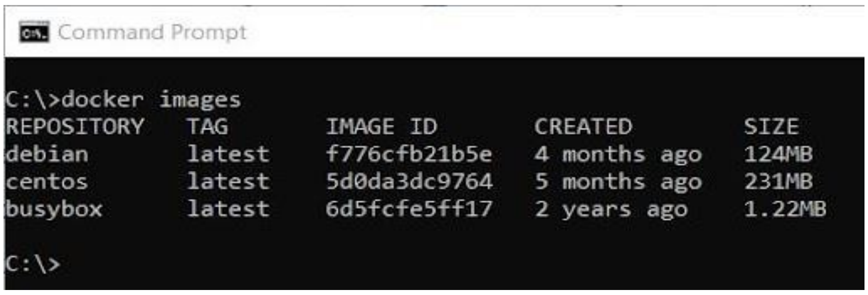
Save the file as ‘calculator_test_case.py’. run your test case in the command line as:

```
python calculator_test_case.py
```

output:

Program 12: Develop test cases for the containerized application using Selenium.

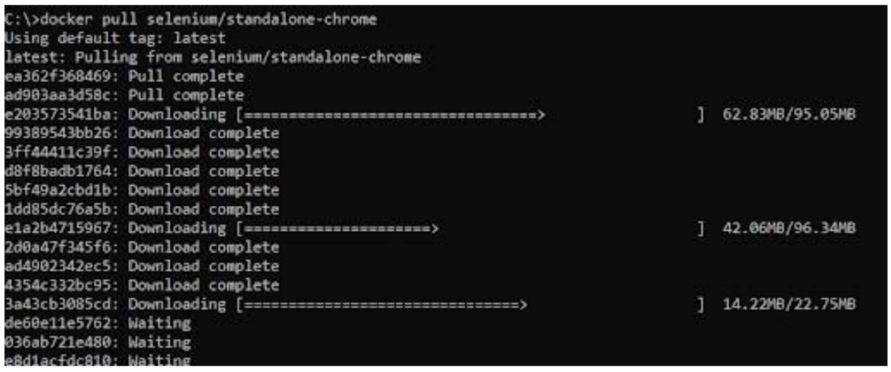
Running selenium tests on docker images.
Pull the docker image: in command prompt type ‘docker images’



Command Prompt

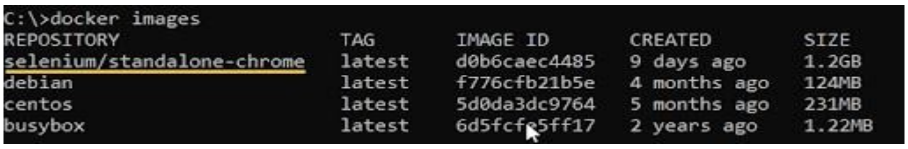
```
C:\>docker images
REPOSITORY   TAG      IMAGE ID       CREATED        SIZE
debian       latest   f776cfb21b5e   4 months ago   124MB
centos       latest   5d0da3dc9764   5 months ago   231MB
busybox      latest   6d5fcfe5ff17   2 years ago    1.22MB

C:\>
```

If you don’t have selenium standalone chrome docker image, run the following command to download a copy of image into the system
docker pull selenium/standalone-chrome



```
C:\>docker pull selenium/standalone-chrome
Using default tag: latest
latest: Pulling from selenium/standalone-chrome
ea362f368469: Pull complete
ad903aa3d58c: Pull complete
e203573541ba: Downloading [=================================>                ]  62.83MB/95.05MB
99389543bb26: Download complete
3ff44411c39f: Download complete
d8f8badb1764: Download complete
5bf49a2cbd1b: Download complete
1dd85dc76a5b: Download complete
e1a2b4715967: Downloading [====================>                             ]  42.06MB/96.34MB
2d0a47f345f6: Download complete
ad4902342ec5: Download complete
4354c332bc95: Download complete
3a43cb3085cd: Downloading [===============================>                  ]  14.22MB/22.75MB
de60e11e5762: Waiting
036ab721e480: Waiting
e8d1acfdc810: Waiting
```

Upon rerunning the command docker images, selenium/standalone-chrome image appears in th
list.

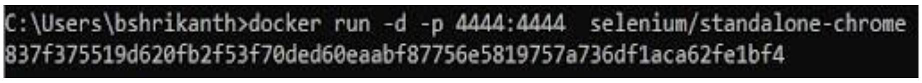


```
C:\>docker images
REPOSITORY                   TAG      IMAGE ID       CREATED        SIZE
selenium/standalone-chrome   latest   d0b6caec4485   9 days ago     1.2GB
debian                       latest   f776cfb21b5e   4 months ago   124MB
centos                       latest   5d0da3dc9764   5 months ago   231MB
busybox                      latest   6d5fcfe5ff17   2 years ago    1.22MB
```

Running the selenium web driver docker container.

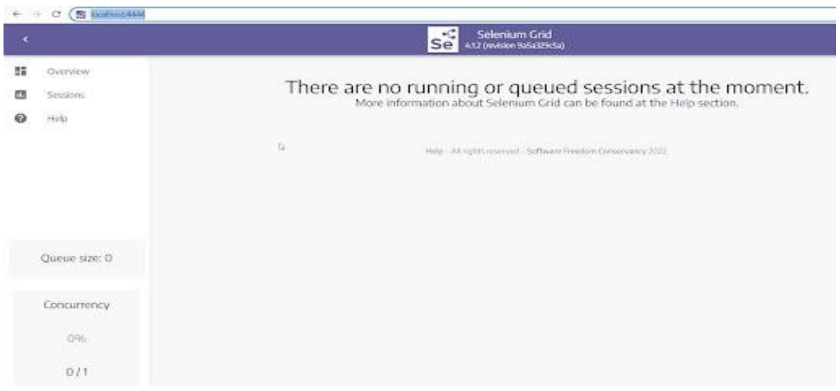
Upon pulling the selenium/standalone chrome image onto the system, start the container by running the following command.

docker run -d -p 4444:4444 -v /dev/shm:/dev/shm selenium/standalone-chrome



```
C:\Users\bshrikanth>docker run -d -p 4444:4444  selenium/standalone-chrome
837f375519d620fb2f53f70ded60eaabf87756e5819757a736df1aca62fe1bf4
```

The command after execution will return the containerID. Open the browser and navigate to http://localhost:4444/. It reflects Selenium Grid UI, as shown:



There are no running or queued sessions at the moment.
More information about Selenium Grid can be found at the Help section.

Help - All rights reserved - Software Freedom Conservancy 2022.

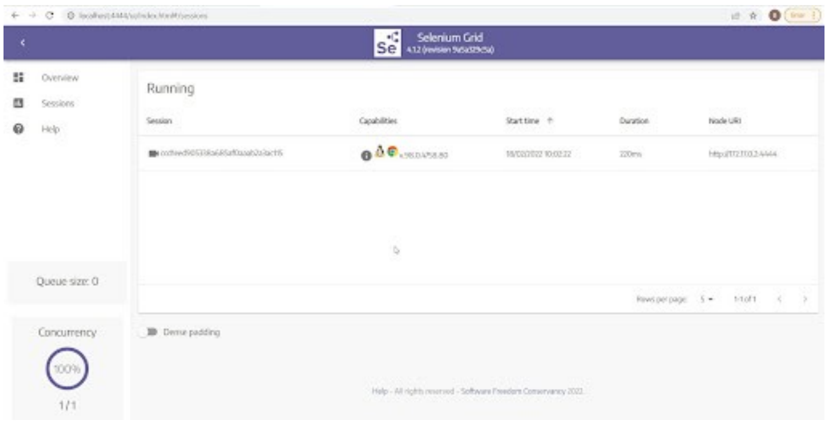
Creating a sample test file:

```
from selenium import webdriver
options = webdriver.ChromeOptions()
options.add_argument('--ignore-ssl-errors=yes')
options.add_argument('--ignore-certificate-errors')
driver = webdriver.Remote(
command_executor='http://localhost:4444/wd/hub',
options=options
)
driver.get("https://www.browserstack.com/")
driver.find_element_by_link_text("Get started free").click()
```

seleniumDockerTest.py

```
from selenium import webdriver
import time
print("Test Execution Started")
options = webdriver.ChromeOptions()
options.add_argument('--ignore-ssl-errors=yes')
options.add_argument('--ignore-certificate-errors')

driver = webdriver.Remote(
command_executor='http://localhost:4444/wd/hub',
options=options
)
#maximize the window size
driver.maximize_window()
time.sleep(10)
#navigate to browserstack.com
driver.get("https://www.browserstack.com/")
time.sleep(10)
#click on the Get started for free button
driver.find_element_by_link_text("Get started free").click()
time.sleep(10)
#close the browser
driver.close()
driver.quit()
```

print("Test Execution Successfully Completed!")
save the test case file with '.py' extension.
Executing the test case
python <filename>



Running

Session	Capabilities	Start time	Duration	Node URI
[illegible]	98.0.4758.80	18/02/2022 10:02:22	220ms	http://172.17.0.2:4444

Viva Questions

1. What is DevOps? Why it's needed?

Ans: the DevOpsis a combination ofsoftware development (Dev) and operations(Ops). It allows a single team to handle the entire application lifecycle, from development to testing, deployment and operations.

2. How is DevOps different from traditional IT??

Ans : In traditional methods, there are lot of huge moving parts to a development cycle, making scheduling quite a challenging task. On the contrary, DevOps is built on continuous smaller releases and automation from a dedicated team, making it much easier to schedule.

3.Explain DevOps Lifecycle.

Ans: DevOps lifecycle is defined as a combination of different phases of continuous software development, integration, testing, deployment and monitoring. A competent DevOps lifecycle is capable of building superior quality software through the system.

4.What is the DevOps pipeline?

Ans: The well-built pipeline is – 1. Continous integration/continuous delivery(CI/CD), 2. Continous Testing/Continous Deployment (CT/CD), 3.Continous Monitoring, 4.Continous Feedback, 5.continous Operations, 6. Tools and Control Environment, 7. Build Server and Automation, 8. DevOps Pipeline Deployment.

5.What is continuous integration?

Ans: Continous Integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. It's a primary DevOps best practice, allowing developers to frequently merge code changes into a central repository where builds and tests then run.

6.How is DevOps different from Agile?

Ans: Agile is a philosophy about how to develop and deliver software, while DevOps describes how to continuously deploy code through the use of modern tools and automated processes.

7.What are the principles of DevOps?

Ans: 1. Create a collaborative environment. 2. Automation. 3. Monitor the process continuously. 4. Implement end-to-end responsibility. 5. Foster continuous improvement.

8.What is the role, responsibility and skill of a DevOps Engineer?

Ans: The typical responsibilities for DevOps engineers are – building and setting up new development tools and infrastructure, understanding the needs of stakeholders and conveying this to developers and working on ways to automate and improve development and release processes.

9.Name some DevOps Automation tools.
Ans: Some of the DevOps automation tools are – Docker, Kubernetes (K8s), Raygun, Splunk, Git, Ansible, Jenkins, Bamboo, BitBucket, GitHub.

10. Explain the use of Selenium?
Ans: Selenium WebDriver is a collection of open source APIs which are used to automate the testing of a web application. It supports many browsers such as Firefox, Chrome, IE and Safari.

11. What is Jenkins? What is Jenkinsfile?
Ans: Jenkins is an open source automation server. It helps automate the parts of the software development related to building, testing and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat. A Jenkins file is a text file that contains the definition of a Jenkins pipeline and is checked into source control.

12. What is Git /GitHub?
Ans: Git is a software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

13. Name some basic Git commands.
Ans: git config, git init, git clone, git add, git commit, git diff, git reset, git log, git branch, git checkout, git push.

14. What is the difference between a centralized and distributed version control system (VCS)?
Ans: The main difference between centralized and distributed version control is that, in centralized version control, the versions are saved in the remote repository, while in distributed version control, versions can be saved in the remote repository as well as in local repositories of the local machines.

15. Explain Docker. What are the advantages of Docker over virtual machines?
Ans: Docker is a software platform for building applications based on containers-small and lightweight execution environments that make shared use of the operating system kernel but otherwise run in isolation from one another. Docker containers are process-isolated and don't require a hardware hypervisor. This means docker containers are much smaller and require far fewer resources than a VM. Hence Docker is very fast.

16. What is the difference between a registry and a repository?
Ans: Generally, a registry simply records official information that relates to an asset, whereas the repository stores the assets themselves.

17. What is a merge conflict in Git, how can it be resolved?
Ans: A merge conflict is an event that occurs when Git is unable to automatically resolve differences in code between two commits. To resolve the issue use following commands-

18. git log –merge [this command helps to produce the list of commits that are causing the conflict.git diff. [this command helps to identify the differences between the states repositories or files.]

19. git checkout [this command is used to undo the changes made to the file or for changing branches.]git reset –mixed [this command is used to undo changes directory and staging area.]

20. git merge –abort [this command helps in exiting the merge process and returning back to the state before the merging began.]

21. git reset [this command is used at the time of merge conflict to reset the conflicted files to their original state.]

22. Explain the concept of branching.
Ans: Branching is used in version control and software management to maintain stability while isolated changes are made to code. Branching facilitatesthe development of bug fixes, the addition of new capabilities and the integration of new versions after they have been tested in isolation.

23. Name some cloud platforms used for DevOps implementation.
Ans- some cloud computing platform used for DevOps implementation, are- Google Cloud, Amazon Web Services (AWS), Microsoft Azure.

24. What is Component Based Development (CBD)?
Ans- Component based development is way to approach product development. In this method, developers always look for existing well defined, tested and verified components to compose and assemble them to a product instead of developing from scratch.