



PRIFYSGOL CYMRU
Y Drindod Dewi Sant
UNIVERSITY OF WALES
Trinity Saint David
SWANSEA - ABERTAWE

Applied Computing

Wales Institute of Science and Art (WISA)

Network Programmability (ACCB5029)

Lab 8: Building a CI/CD Pipeline using Jenkins.

ICONS Used in this document:



Do research and complete the task. Need referencing/ in-text citations. Shall be included as a part of your coursework.



Record your Reflection.

 **Networking**
CISCO Academy

Task 1:**Objectives**

1. Commit the Sample App to Git.
2. Modify and Push changes to Git.
3. Download and run the Jenkins Docker Image.
4. Configure Jenkins.
5. Use Jenkins to run a build of your application.
6. Use Jenkins to test a build.
7. Create a Pipeline in Jenkins.

Task 2: (Build a CI/CD Pipeline for the new application you created in a Docker Container in Task 2 of Lab 7)**Objectives**

1. Discuss with your group (the group you are working for Coursework 1) and propose a new pipeline for the application built in a docker container during your lab 7 – task 2 activity. You shall use any tools (including Jenkins). Document with adequate annotations and timestamped screenshots.
2. Individually, write a 250 words personal reflection on your understanding on CI/CD Pipeline.

Required Resources

Host computer with DEVASC VM.

Note: You must have completed Lab 7 to continue with this Lab. This lab will not have step by step instruction. Commands used in the previous labs are not stated here. This is to help you to complete the task with adequate understanding.

Instructions

Launch the DEVASC VM

1. Commit the Sample App to Git

- 1.1. Login to GitHub and create a new repository with the following details.
 - + Repository name: my-sample-app
 - + Description: Lab 8 Explore CI/CD with GitHub and Jenkins
 - + Public/Private: Private
 - + Select: Create repository
- 1.2. Configure your Git credentials locally in the VM. Recall the commands used in Version Control Lab. You must configure username and email.
- 1.3. Initialize a directory as the Git repository.
 - 1.3.1. You will use the sample-app files you created in a previous lab. Navigate to the sample-app directory you created during your lab 7 and initialize it as a Git repository. Recall the steps used to initialize from our earlier labs.
- 1.4. Point Git repository to GitHub repository.
 - 1.4.1. Use the git remote add command to add a Git URL with a remote alias of "origin" and point to the newly created repository on GitHub. *(Use the URL of the Git repository you created earlier. Note that your GitHub username is case-sensitive.)*
 - 1.4.2. Use appropriate command to check if the Git URL with a remote alias is added.
- 1.5. Stage, commit, and push the sample-app files to the GitHub repository.
 - 1.5.1. Use the git add command to stage the files in the sample-app directory. Use the asterisk (*) argument to stage all files in the current directory.
 - 1.5.2. Use the git status command to see the files and directories that are staged and ready to be committed to your GitHub repository.

```
Initialized empty Git repository in /home/devasc/labs/devnet-src/sample-app/.git/
```

```
origin https://github.com/accb6019/my-sample-app.git (fetch)
origin https://github.com/accb6019/my-sample-app.git (push)
```

```
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   copingtext
    new file:   running.txt
    new file:   sample-app.sh
    new file:   sample_app.py
    new file:   static/style.css
    new file:   tempdir/Dockerfile
    new file:   tempdir/sample_app.py
    new file:   tempdir/static/style.css
    new file:   tempdir/templates/index.html
    new file:   templates/index.html
    new file:   userdetails
```

- 1.5.3. Use the git commit command to commit the staged files and start tracking changes. Add a message of your choice or use the one provided here. Eg: "Committing my-sample-app files after Lab 7"

```
[master (root-commit) 2c26341] Committing my-sample-app files after Lab 7
11 files changed, 134 insertions(+)
create mode 100644 copingtext
create mode 100644 running.txt
create mode 100755 sample-app.sh
create mode 100644 sample_app.py
create mode 100644 static/style.css
create mode 100644 tempdir/Dockerfile
create mode 100644 tempdir/sample_app.py
create mode 100644 tempdir/static/style.css
create mode 100644 tempdir/templates/index.html
create mode 100644 templates/index.html
create mode 100755 userdetails
```

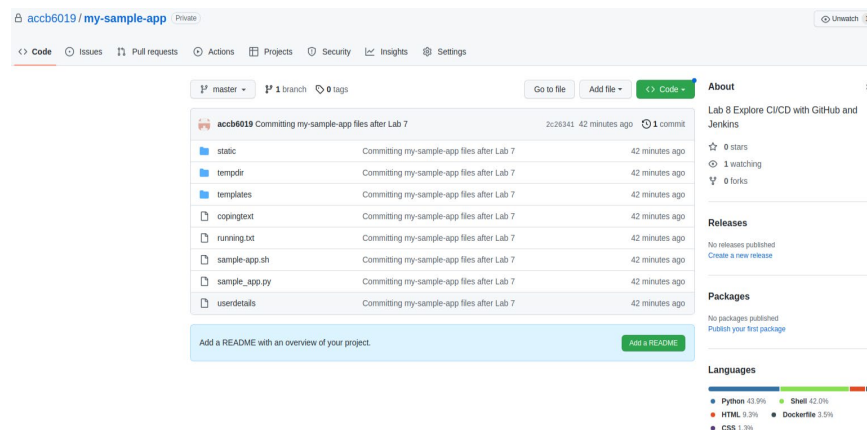
1.5.4. Use the git push command to push your local sample-app files to your GitHub repository.

- + It will ask for a username and Password.
- + Note that the password authentication is no more supported, recall the personal access token we created in our previous labs.
- + Use Personal Access Token to authenticate.

```
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 2 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (13/13), 2.81 KiB | 410.00 KiB/s, done.
Total 13 (delta 0), reused 0 (delta 0)
To https://github.com/accb6019/my-sample-app.git
* [new branch]      master -> master
```

(File numbers may vary depending on the files you created during Lab 7)

1.5.5. Take appropriate screenshot to show that the files are committed in the Github repository.



Note: If, instead of a request for your username, if you get other messages, then you misconfigured your GitHub credentials in previous steps. The URL must have the correct case-sensitive username and the name of the repository that you created earlier. To reverse your previous git add command, use the command git remote rm origin. Then return to appropriate previous steps making sure to enter the correct credentials and correct URL.

Note: If, after entering your username and Personal Access Token as password, you get a fatal error stating repository is not found, you most likely submitted an incorrect URL. You will need to reverse your git add command with the git remote rm origin command.

2. Modify the Sample App and Push Changes to Git

We will be installing a Jenkins Docker image that will use port 8080. *Recall that your sample-app files are also specifying port 8080. The Flask server and Jenkins server cannot both use*

8080 at the same time. To overcome this, we will change the port number used by the sample-app files, run the sample-app again to verify it works on the new port, and then push your changes to your GitHub repository.

- 2.1. Open the sample-app files.
Make sure you are still in the ~/labs/devnet-src/sample-app directory as these are the files that are associated with your GitHub repository. Open both sample_app.py and sample-app.sh for editing.
- 2.2. Edit the sample-app files accordingly to change the instance of port 8080 to 5050.

```
sample_app.py
home > devsec > labs > devnet-src > sample-app > sample_app.py > ...
4 from flask import request
5
6 # Importing render template method from the flask
7 # HTML file can be rendered in Flask automatic
8 from flask import render_template
9
10 sample = Flask(__name__)
11 # calls the page and renders - used in development
12 # the "/" (root directory) is bound to the main
13 # So, when the user goes to http://localhost:8080
14 # the output of the return statement will be displayed
15
16 @sample.route("/")
17 def main():
18     #request.remote_addr is a property in the Flask
19     # It asks the IP address of the client that
20     # This property returns a string representing
21
22     #return "You are calling me from " + request.remote_addr
23     # editing to the return function to use render template
24     return render_template("index.html")
25
26 #Use the function run in the main
27 if __name__ == "__main__":
28     sample.run(host="0.0.0.0", port=5050)
```

```
sample_app.py sample-app.sh X
home > devsec > labs > devnet-src > sample-app > sample-app.sh
12 echo "FROM python" > tempdir/Dockerfile
13 #Your sample_app.py script needs Flask, so add the Docker RUN
14 echo "RUN pip install flask" >> tempdir/Dockerfile
15
16 #Your container will need the website folders and the sample
17 #so add the Docker COPY commands to add them to a directory inside the
18 #Here, we use /home/myapp as the parent directory inside the
19 #Besides copying the sample_app.py file to the Dockerfile,
20 #we will also be copying the index.html file from the template
21 #the style.css file from the static directory.
22 echo "COPY ./static /home/myapp/static" >> tempdir/Dockerfile
23 echo "COPY ./templates /home/myapp/templates/" >> tempdir/Dockerfile
24 echo "COPY sample_app.py /home/myapp/" >> tempdir/Dockerfile
25 #Use the Docker EXPOSE command to expose port 5050 for use by
26 echo "EXPOSE 5050" >> tempdir/Dockerfile
27 #add the Docker CMD command to execute the Python script
28 echo "CMD python3 /home/myapp/sample_app.py" >> tempdir/Dockerfile
29 #Add the commands to the sample-app.sh file to switch to the
30 #The docker build command -t option allows you to specify the
31 #the trailing period (.) indicates that you want the container
32 cd tempdir
33 docker build -t sampleapp .
34 #Add the docker run command to the sample-app.sh file to start
35 docker run -t -d -p 5050:5050 --name samplerunning sampleapp
```

- 2.3. Build and verify the sample-app. Enter the bash command to build your app using the new port 5050.

```
Step 2/7 : RUN pip install flask
--> Using cache
--> d17b17f10990
Step 3/7 : COPY ./static /home/myapp/static/
--> Using cache
--> 920f3598d0ee
Step 4/7 : COPY ./templates /home/myapp/templates/
--> Using cache
--> 340177e7be85
Step 5/7 : COPY sample_app.py /home/myapp/
--> 3691ceb2c283
Step 6/7 : EXPOSE 5050
--> Running in 7b3f46842885
Removing intermediate container 7b3f46842885
--> cabb87ea800
Step 7/7 : CMD python3 /home/myapp/sample_app.py
--> Running in f12417d10588
Removing intermediate container f12417d10588
--> ec7b5fd5e658
Successfully built ec7b5fd5e658
Successfully tagged sampleapp:latest
c429fd6993e9cc15662c3093956872395764b03a3fa4b691f4bcab345f2770
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
c429fd6993e6   sampleapp "/bin/sh -c 'python3..." 2 seconds ago   Up less than a second   0.0.0.0:5050->5050/tcp              samplerunning
```

- 2.4. Open a browser tab and navigate to localhost:5050. What is the output?

My First Sample app x +

localhost:5050

You are calling me from 172.17.0.1

Created by - Nitheesh for accb5029

- 2.5. Shut down the server when you have verified that it is operating on port 5050. Return to the terminal window where the server is running and press CTRL+C to stop the server.
- 2.6. Push your changes to GitHub with a comment "Changed port from 8080 to 5050"

```
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   sample-app.sh
    modified:   sample_app.py
    modified:   tempdir/Dockerfile
    modified:   tempdir/sample_app.py
```

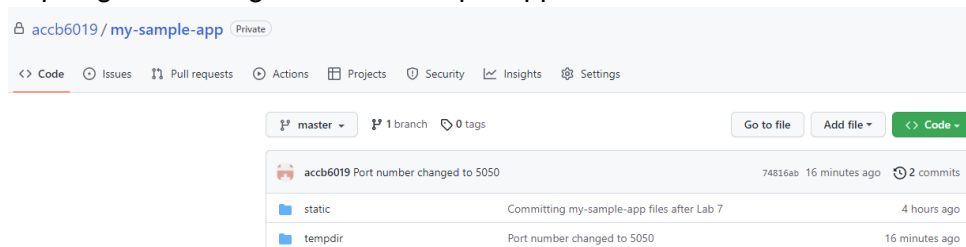
```

devasc@labvm:~/labs/devnet-src/sample-app$ git commit -m "Port number changed to 5050"
[master 74816ab] Port number changed to 5050
4 files changed, 6 insertions(+), 6 deletions(-)
devasc@labvm:~/labs/devnet-src/sample-app$ git push origin master
Username for 'https://github.com': accb6019
Password for 'https://accb6019@github.com':

Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 626 bytes | 626.00 KiB/s, done.
Total 6 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/accb6019/my-sample-app.git
2c26341..74816ab master -> master

```

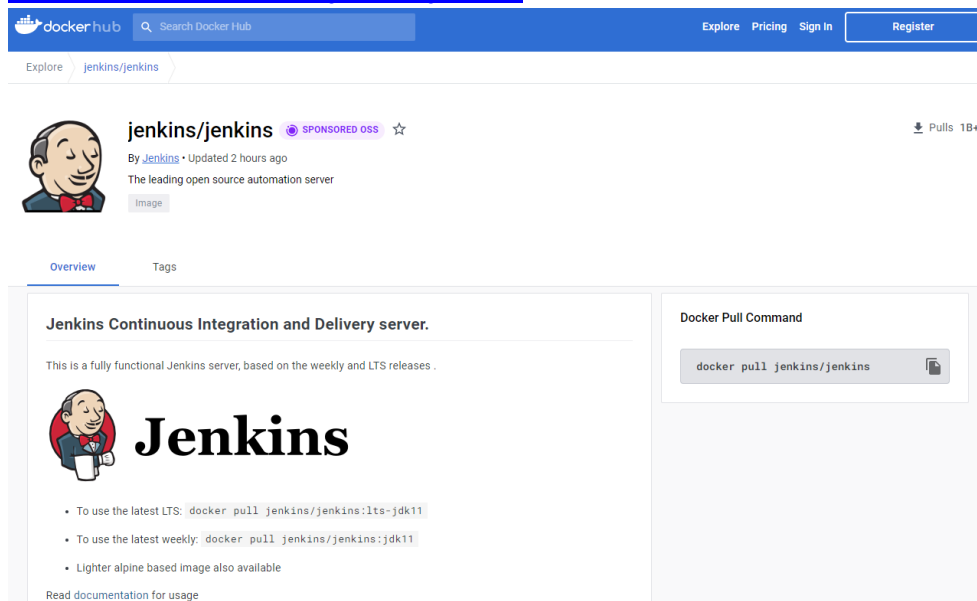
- 2.7. You can verify that your GitHub repository is updated by visiting <https://github.com/github-user/sample-app>.



3. Download and run the Jenkins Docker Image

- 3.1. The Jenkins Docker image is stored here:

<https://hub.docker.com/r/jenkins/jenkins>.



- 3.2. Pull Jenkins docker image

```

devasc@labvm:~/labs/devnet-src/sample-app$ docker pull jenkins/jenkins
Using default tag: latest
latest: Pulling from jenkins/jenkins
32fb02163b6b: Already exists
c09d5e9e1188: Downloading [=====>] 24.17MB/51.63MB
a56533012712: Download complete
7936e107ffe7: Download complete
3ca683058265: Download complete
9b9fbc92e59f: Downloading [=====>] 27.48MB/98.19MB
a34b17185d9d: Download complete
3d029f949ee6: Download complete
dadc99893dfc: Downloading [=>] 2.158MB/76.93MB
795d6802d2d2: Waiting
fb7363d543fb: Waiting
1d65c556d569: Waiting
a22ceeae8b5b: Waiting

```

3.2.1. Explore Jenkins Server

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
sampleapp	latest	ec7b5fd5e658	56 minutes ago	941MB
<none>	<none>	f8a55a3458f9	29 hours ago	941MB
<none>	<none>	8911fa3b09a3	4 days ago	941MB
jenkins/jenkins	lts	d5ed2ceef0ec	5 days ago	471MB
jenkins/jenkins	latest	f72705c021e2	5 days ago	471MB
python	latest	33ce09363487	12 days ago	925MB

devasc@labvm:~/labs/devnet-src/sample-app\$ docker ps						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
15556596d0de	jenkins/jenkins:lts	"/usr/bin/tini -- /u."	3 minutes ago	Up 3 minutes	0.0.0.0:8080->8080/tcp, 50000/tcp	jenkins_server


```
devasc@labvm:~/labs/devnet-src/sample-app$ docker exec -it jenkins_server /bin/bash
root@15556596d0de:/# cat /var/jenkins_home/secrets/initialAdminPassword
53db94249e5148db98e20898ccf8ccbb
```

3.3. Start the Jenkins Docker container.

Enter the following command on one line. This command will start the Jenkins Docker container and then allow Docker commands to be executed inside your Jenkins server.

```
devasc@labvm:~/labs/devnet-src/sample-app$ docker run --rm -u root -p 8080:8080 -v jenkins-data:/var/jenkins_home -v $(which docker):/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock -v "$HOME"/home --name jenkins_server jenkins/jenkins:lts
```

```
devasc@labvm:~/labs/devnet-src/sample-app$ docker run --rm -u root -p 8080:8080 -v jenkins-data:/var/jenkins_home -v $(which docker):/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock -v "$HOME"/home --name jenkins_server jenkins/jenkins:lts
Unable to find image 'jenkins/jenkins:lts' locally
lts: Pulling from jenkins/jenkins
```

The options used in this docker run command are as follows:

- rm - This option automatically removes the Docker container when you stop running it.
- u - This option specifies the user. You want this Docker container to run as root so that all Docker commands entered inside the Jenkins server are allowed.
- p - This option specifies the port the Jenkins server will run on locally.
- v - These options bind mount volumes needed for Jenkins and Docker.
 - The first -v specifies where Jenkins data will be stored.
 - The second -v specifies where to get Docker so that you can run Docker inside the Docker container that is running the Jenkins server.
 - The third -v specifies the PATH variable for the home directory.

3.4. Verify the Jenkins server is running.

The Jenkins server should now be running. Copy the admin password that displays in the output, as shown in the following.

Do not enter any commands in this server window. If you accidentally stop the Jenkins server, you will need to re-enter the docker run command. After the initial install, the admin password is displayed as shown below.

```
2023-03-14 11:35:26.918+0000 [id=30] INFO jenkins.install.SetupWizard#init:
*****
*****
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
53db94249e5148db98e20898ccf8ccbb
This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
```

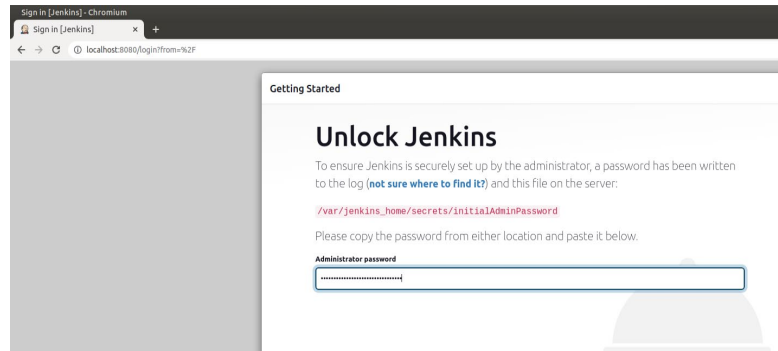
Note: If you lose the password, or it does not display as shown above, or you need to restart the Jenkins sever, you can always retrieve the password by accessing the command line of Jenkins Docker container.

Create a second terminal window in VS Code and enter the following commands so that you do not stop the Jenkins server.:

```
devasc@labvm:~/labs/devnet-src/sample-app$ docker exec -it jenkins_server /bin/bash
root@15556596d0de:/# cat /var/jenkins_home/secrets/initialAdminPassword
53db94249e5148db98e20898ccf8ccbb
```

4. Configure Jenkins

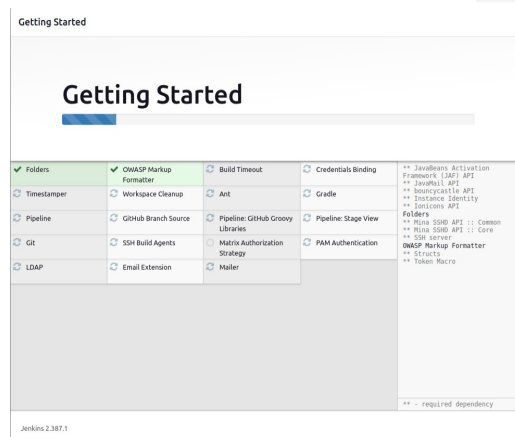
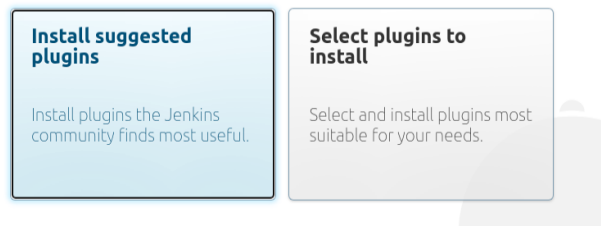
- 4.1. Open a web browser tab. Navigate to <http://localhost:8080/> and login in with your copied admin password.



- 4.2. Install the recommended Jenkins plugins.

Customize Jenkins

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



In the terminal window, you will see log messages as the installation proceeds. Be sure that you do not close this terminal window. You can open another terminal window for access to the command line.

- 4.3. After the installation finishes, you are presented with the Create First Admin User window. For now, click Skip and continue as admin at the bottom.

[Skip and continue as admin](#) [Save and Continue](#)

- 4.4. In the Instance Configuration window, do not change anything. Click Save and Finish at the bottom.

Jenkins 2.387.1

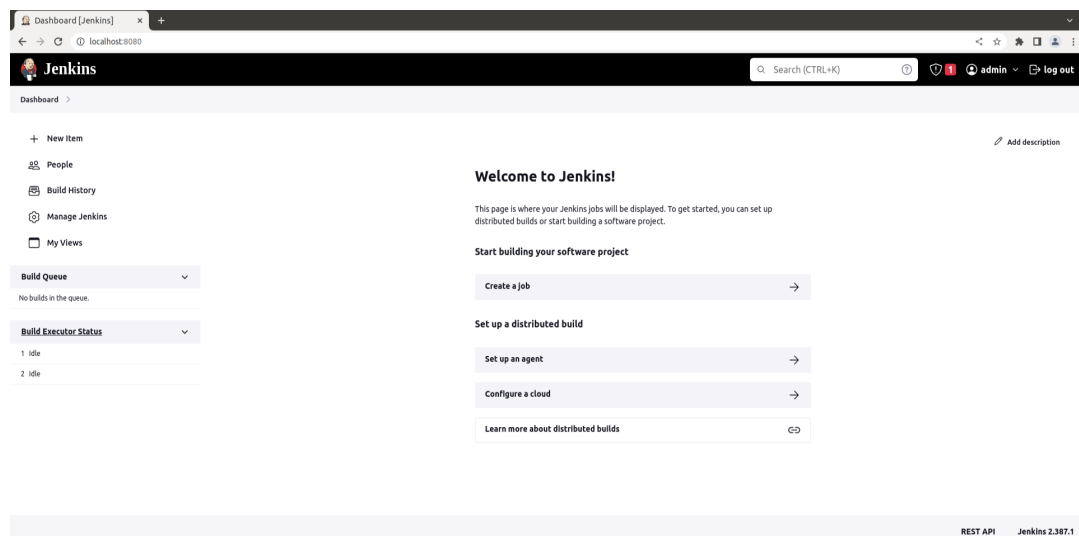
Not now

Save and Finish

To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

5. Start using Jenkins.

- 5.1. Click Start using Jenkins. You should now be on the main dashboard with a Welcome to Jenkins! message.



The fundamental unit of Jenkins is the job (also known as a project). You can create jobs that do a variety of tasks including the following:

- + Retrieve code from a source code management repository such as GitHub.
- + Build an application using a script or build tool.
- + Package an application and run it on a server.

- 5.2. Let's create a simple Jenkins job that retrieves the latest version of your sample-app from GitHub and runs the build script. In Jenkins, you can then test your app and add it to a development pipeline.

- 5.2.1. Create a new job: Click the Create a job link or click New Item in the menu on the left.

- 5.2.2. In the Enter an item name field, fill in the name MySamAppBld.

- 5.2.3. Click Freestyle project as the job type and click ok.

In the description, the SCM abbreviation stands for software configuration management, which is a classification of software that is responsible for tracking and controlling changes in software.

- 5.2.4. Configure the Jenkins MySamAppBld.

You are now in the configuration window where you can enter details about your job. The tabs across the top are just shortcuts to the sections below. Click through the tabs to explore the options you can configure. For this simple job, you only need to add a few configuration details.

5.2.4.1.

5.2.4.2.

5.2.4.3.

5.2.4.4.

5.2.4.5.

Dashboard > MySamAppBld > Configuration

Configure

General

- Source Code Management
- Build Triggers
- Build Environment

General

Description

This is my first Jenkins Job - I belong to the Group - Lecturer and I am Nitheesh

[Plain text] [Preview](#)

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

<https://github.com/accb6019/my-sample-app.git>

Failed to connect to repository: Command "git ls-remote -h - https://github.com/accb6019/my-sample-app.git HEAD" returned status code 128:
 stdout:
 stderr: remote: Support for password authentication was removed on August 13, 2021.
 remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
 fatal: Authentication failed for 'https://github.com/accb6019/my-sample-app.git/'

[+ Add](#)

Jenkins

Username ?

[accb6019](#)

☐ Treat username as secret ?

Password ?

.....

Credentials ?

- none -

- none -

[accb6019/*****](#)

Configure

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

Build Environment

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s) ?
- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published build scans
- ☐ Terminate a build if it's stuck
- ☐ With Ant ?

Build Steps

Add build step ^

Filter

- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script

5.2.4.6.

5.2.4.7. Add build step dropdown, choose Execute shell.

5.2.4.8. In the Command field, enter the command you use to run the build for my-sample-app.sh script.

Build Steps

Execute shell ?

Command

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

```
bash ./my-sample-app.sh
```

5.2.4.9. Click the Save button. You are returned to the Jenkins dashboard with the BuildAppJob selected.

- 5.3. On the left side, click Build Now to start the job. Jenkins will download your Git repository and execute the build command `bash ./my-sample-app.sh`. Your build should succeed because you have not changed anything in the code since you modified the code.

5.3.1. Did the build succeed? If no, like the one in right side, why? Click on the build to Troubleshoot. Record your observations. View the console output.


Build #1 (Mar 14, 2023, 1:54:05 PM)

 No changes.

 Started by user [admin](#)


Revision: 74816ab0c8b160477c2a2ba985bef9749de6820d
Repository: <https://github.com/accb6019/my-sample-app.git>
 • refs/remotes/origin/master

```

Started by user admin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/MySampleAppBld
The recommended git tool is: NONE
using credential 0380bad7-b2ea-4c26-8609-a74c1cfa14c7
Cloning the remote git repository
Cloning repository https://github.com/accb6019/my-sample-app.git
> git init /var/jenkins_home/workspace/MySampleAppBld # timeout=10
Fetching upstream changes from https://github.com/accb6019/my-sample-app.git
> git --version # timeout=10
> git --version # 'git version 2.39.2'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/accb6019/my-sample-app.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/accb6019/my-sample-app.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 74816ab0c8b160477c2a2ba985bef9749de6820d (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 74816ab0c8b160477c2a2ba985bef9749de6820d # timeout=10
Commit message: 'Part number changed to 5958'
First time build. Skipping changelog.
[MySampleAppBld] $ /bin/sh -xe /tmp/jenkins11693651569952839312.sh
+ bash ./my-sample-app.sh
bash: ./my-sample-app.sh: No such file or directory
Build step 'Execute shell' marked build as failure
Finished: FAILURE

```

Another issue to fix: See that the ports are not displayed for samplerunning even though the build is successful. Troubleshoot!!!

```

Step 1/7 : FROM python
--> 33ce09363487
Step 2/7 : RUN pip install flask
--> Using cache
--> d17b17f1d090
Step 3/7 : COPY ./static /home/myapp/static/
--> c513bafb2d4c
Step 4/7 : COPY ./templates /home/myapp/templates/
--> c2c6b2b019a
Step 5/7 : COPY sample_app.py /home/myapp/
--> a65a593578a5
Step 6/7 : EXPOSE 5050
--> Running in a441222dc1c7
Removing intermediate container a441222dc1c7
--> aaf9bc96379a
Step 7/7 : CMD python3 /home/myapp/sample_app.py
--> Running in f563620c21be
Removing intermediate container f563620c21be
--> d939dc8bdc2d
Successfully built d939dc8bdc2d
Successfully tagged sampleapp:latest
docker: Error response from daemon: Conflict. The container name '/samplerunning' is already in use by container 'c429f66993e69cc13662c3093956872395764083a3af40891f4b'. You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
155594060de	jenkins/jenkins:lts	"/usr/bin/tini -- /u..."	2 hours ago	Up 2 hours	0.0.0.0:8080->8080/tcp, 50000/tcp	jenkins_server
c429f66993e6	ec705f0e508	"/bin/sh -c 'python3..."	3 hours ago	Exited (137) 3 hours ago		samplerunning

Finished: SUCCESS

5.3.2. Make necessary changes to execute the build successfully. You must get an outcome like the below one. Check the console output as well.


Build #2

 Mar 14, 2023, 1:59 PM

 Started by user [admin](#)


Revision: 74816ab0c8b160477c2a2ba985bef9749de6820d
Repository: <https://github.com/accb6019/my-sample-app.git>
 • refs/remotes/origin/master

```

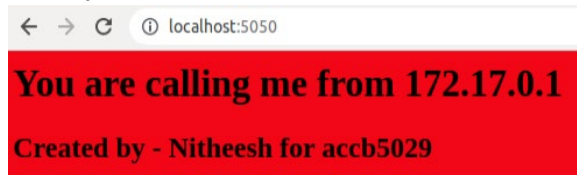
Step 1/7 : FROM python
--> 33ce09363487
Step 2/7 : RUN pip install flask
--> Using cache
--> d17b17f1d090
Step 3/7 : COPY ./static /home/myapp/static/
--> c513bafb2d4c
Step 4/7 : COPY ./templates /home/myapp/templates/
--> Using cache
--> c2c6b2b019a
Step 5/7 : COPY sample_app.py /home/myapp/
--> Using cache
--> a65a593578a5
Step 6/7 : EXPOSE 5050
--> Using cache
--> aaf9bc96379a
Step 7/7 : CMD python3 /home/myapp/sample_app.py
--> Running in d939dc8bdc2d
Successfully built d939dc8bdc2d
Successfully tagged sampleapp:latest

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
7c260e640376	sampleapp	"/bin/sh -c 'python3..."	4 seconds ago	Up Less than a second	0.0.0.0:5050->5050/tcp	samplerunning
be59460679e	jenkins/jenkins:lts	"/usr/bin/tini -- /u..."	15 minutes ago	Up 15 minutes	0.0.0.0:8080->8080/tcp, 50000/tcp	jenkins_server

Finished: SUCCESS

5.3.3. Open another web browser tab and verify sample app is running. Type in the local address, localhost:5050. You should see the content of your index.html displayed.



6. Use Jenkins to Test a Build

You will create a second job that tests the build to ensure that it is working properly. Note: You need to stop and remove the samplerunning docker container.

- 6.1. Start a new job for testing your sample-app.
 - 6.1.1. Return to the Jenkins web browser tab and to return to the main dashboard.
 - 6.1.2. Click the New Item link to create a new job.
 - 6.1.3. In the Enter an item name field, fill in the name TestAppJob.
 - 6.1.4. Click Freestyle project as the job type and click OK.

- 6.2. Configure the Jenkins TestAppJob.

General

Description

My First Jenkins Test with my group - Lecturer: I am Nitheesh

[Plain text] [Preview](#)

- ☐ Discard old builds ?
- ☐ GitHub project
- ☐ This project is parameterized ?
- ☐ Throttle builds ?
- ☐ Execute concurrent builds if necessary ?

Advanced ▾

Source Code Management

☒ None

Build Triggers

2

- ☐ Trigger builds remotely (e.g., from scripts) ?
- ☒ Build after other projects are built ?

For Projects to watch, fill in the name of the job that you need to test. In my case, it is MySamAppBld. Execute Shell and write the test script and save.

Build Steps

Execute shell ?

Command

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

```
if curl http://172.17.0.1:5050/ | grep "You are calling me from 172.17.0.1"; then
    exit 0
else
    exit 1
fi
```

- 6.3. Refresh the web page with the refresh button for your browser. You should now see your two jobs listed in a table.

Dashboard >

+ New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

Auto-Refresh

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	MySamAppBld	42 min #5	4 hr 23 min #1	5.7 sec
⌂	☀	TestAppJob	N/A	N/A	N/A

- 6.4. Build the MySamAppBld. If all goes well, you should see the timestamp for the Last Success column update for both MySamAppBld and TestAppJob. This means your code for both jobs ran without error. But you can also verify this for yourself.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	MySamAppBld	35 sec #6	4 hr 26 min #1	8.5 sec
✓	☀	TestAppJob	20 sec #2	49 sec #1	0.13 sec

Note: If timestamps do not update, make sure enable auto refresh is turned on.

- 6.5. Click the Link for TestAppJob and check the Console Output.

✓ Console Output

```
Started by upstream project "MySamAppBld" build number 6
originally caused by:
  Started by user admin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/TestAppJob
[TestAppJob] $ /bin/sh -xe /tmp/jenkins16242969549805420081.sh
+ grep You are calling me from 172.17.0.1
+ curl http://172.17.0.1:5050/
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed

  0     0    0     0    0     0      0      0  --:--:-- --:--:-- --:--:--    0
100  236 100  236    0     0 11800    0  --:--:-- --:--:-- --:--:-- 12421
  <h1>You are calling me from 172.17.0.1</h1>
+ exit 0
Finished: SUCCESS
```

It is not necessary to verify your sample app is running because the TestAppJob already did this for you. However, you can open a browser tab for 172.17.0.1:5050 to see that it is indeed running.

← → ↻ ⚠ Not secure | 172.17.0.1:5050

You are calling me from 172.17.0.1

Created by - Nitheesh for accb5029

7. Create a Pipeline in Jenkins

Although you can currently run your two jobs by simply clicking the Build Now button for the MySamAppBld, software development projects are typically much more complex. These projects can benefit greatly from automating builds for continuous integration of code changes and continuously creating development builds that are ready to deploy. This is the essence of CI/CD. **A pipeline can be automated to run based on a variety of triggers including**

periodically, based on a GitHub poll for changes, or from a script run remotely. However, in this part you will script a pipeline in Jenkins to run your two apps whenever you click the pipeline Build Now button.

- 7.1. Create a Pipeline job.
 - 7.1.1. Click the Jenkins link in the top left, and then New Item and enter FirstPipeline.
 - 7.1.2. Select Pipeline as the job type and click OK.
- 7.2. Configure the FirstPipeline job.
 - 7.2.1. Along the top, click the tabs and investigate each section of the configuration page. Notice that there are several different ways to trigger a build. For the FirstPipeline job, you will trigger it manually.
 - 7.2.2. In the Pipeline section, add the following script.

Pipeline

Definition

Pipeline script

Script ?

```

1 node {
2   stage('Preparation') {
3     catchError(buildResult: 'SUCCESS') {
4       sh 'docker stop samplerunning'
5       sh 'docker rm samplerunning'
6     }
7   }
8   stage('Build') {
9     build 'BuildAppJob'
10  }
11  stage('Results') {
12    build 'TestAppJob'
13  }
14 }

```

This script does the following:

- + It creates a single node build as opposed to a distributed or multi node. Distributed or multi node configurations are for larger pipelines than the one you are building in this lab.
 - + In the Preparation stage, the FirstPipeline will first make sure that any previous instances of the MySamAppBld docker container are stopped and removed. But if there is not yet a running container you will get an error. Therefore, you use the catchError function to catch any errors and return a "SUCCESS" value. This will ensure that pipeline continues to the next stage.
 - + In the Build stage, the FirstPipeline will build your MySamAppBld.
 - + In the Results stage, the FirstPipeline will build your TestAppJob.
- 7.2.3. Click Save and you will be returned to the Jenkins dashboard for the FirstPipeline job. Run the FirstPipeline.
 - 7.2.4. On the left, click Build Now to run the FirstPipeline job. Record your outcome. (If you coded your Pipeline script without error, then the Stage View should show three green boxes with number of seconds each stage took to build. If not, click Configure on the left to return to the FirstPipeline configuration and check your Pipeline script.)

Dashboard > FirstPipeline >

Pipeline FirstPipeline

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

Build History trend

Filter builds...

#2 Mar 14, 2023, 7:03 PM

#1 Mar 14, 2023, 6:41 PM

Stage View

Average stage times:
(Average full run time: ~19s)

	Preparation	Build	Results
#2	650ms	11s	6s
#1	13s	197ms failed	

Permalinks

Examples of two builds. One with error and other is successful.

7.2.5. Verify the FirstPipeline Console Output. You should see output like the following:

```
Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/FirstPipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Preparation)
[Pipeline] catchError
[Pipeline] {
[Pipeline] sh
+ docker stop samplerunning
Error response from daemon: No such container: samplerunning
[Pipeline] }
ERROR: script returned exit code 1
[Pipeline] // catchError
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] build (Building MySamAppBld)
Scheduling project: MySamAppBld
Starting building: MySamAppBld #7
Build MySamAppBld #7 completed: SUCCESS
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Results)
[Pipeline] build (Building TestAppJob)
Scheduling project: TestAppJob
Starting building: TestAppJob #3
Build TestAppJob #3 completed: SUCCESS
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Task 2: (Build a CI/CD Pipeline for the new application you created in a Docker Container in Task 2 of Lab 7)

Objectives

1. Discuss with your group (the group you are working for Coursework 1) and propose a new pipeline for the application built in a docker container during your lab 7 – task 2 activity. You shall use any tools (including Jenkins). Document with adequate annotations and timestamped screenshots.
2. Individually, write a 250 words personal reflection on your understanding on CI/CD Pipeline.

End of document.