

[illegible]

<b>Name</b>	Jenkins: Basics
<b>URL</b>	<a href="https://attackdefense.com/challengedetails?cid=1718">https://attackdefense.com/challengedetails?cid=1718</a>
<b>Type</b>	DevSecOps : CI/CD Tools

**Important Note:** This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

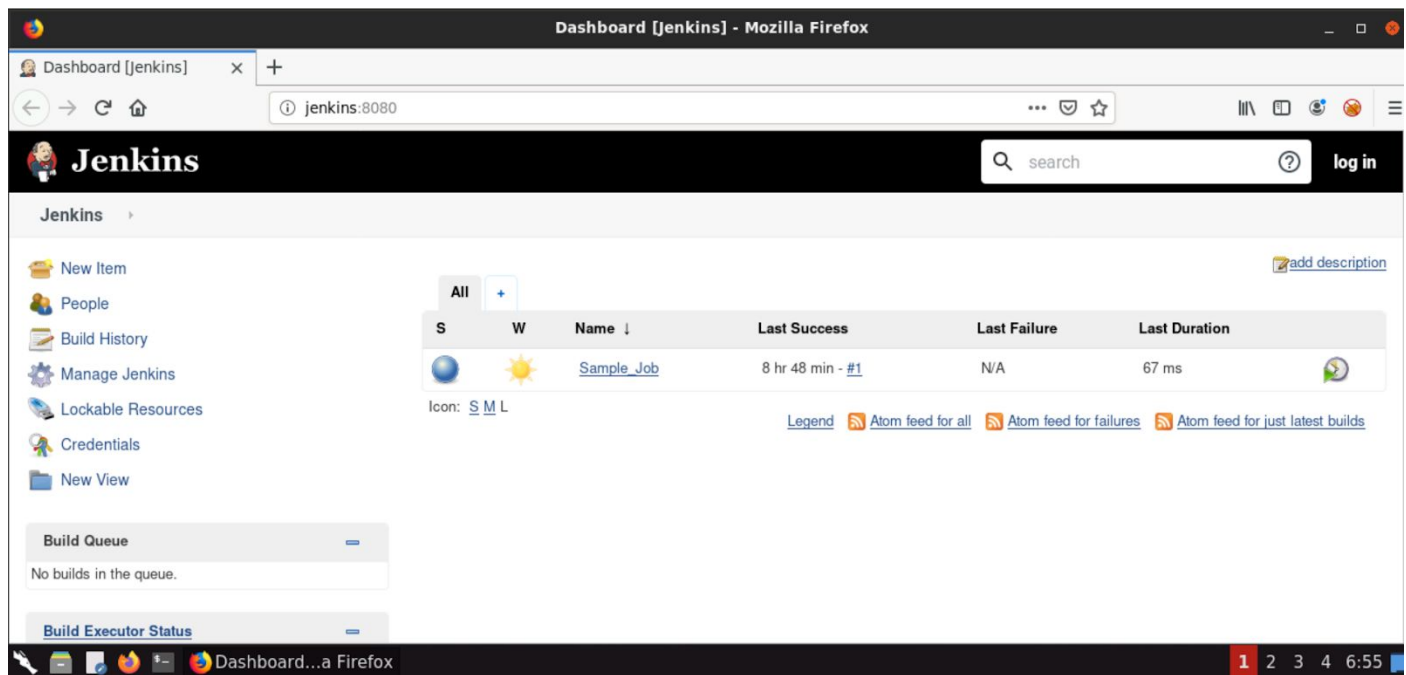
**Objective:** Perform the following activities on the given setup:

- Check the settings/build instructions for project Sample\_Job, fire a build and check the build logs
- Create a building project for a sample-web-server project hosted on local Gitlab instance

**Task I: Check the settings/build instructions for project Sample\_Job, fire a build and check the build logs**

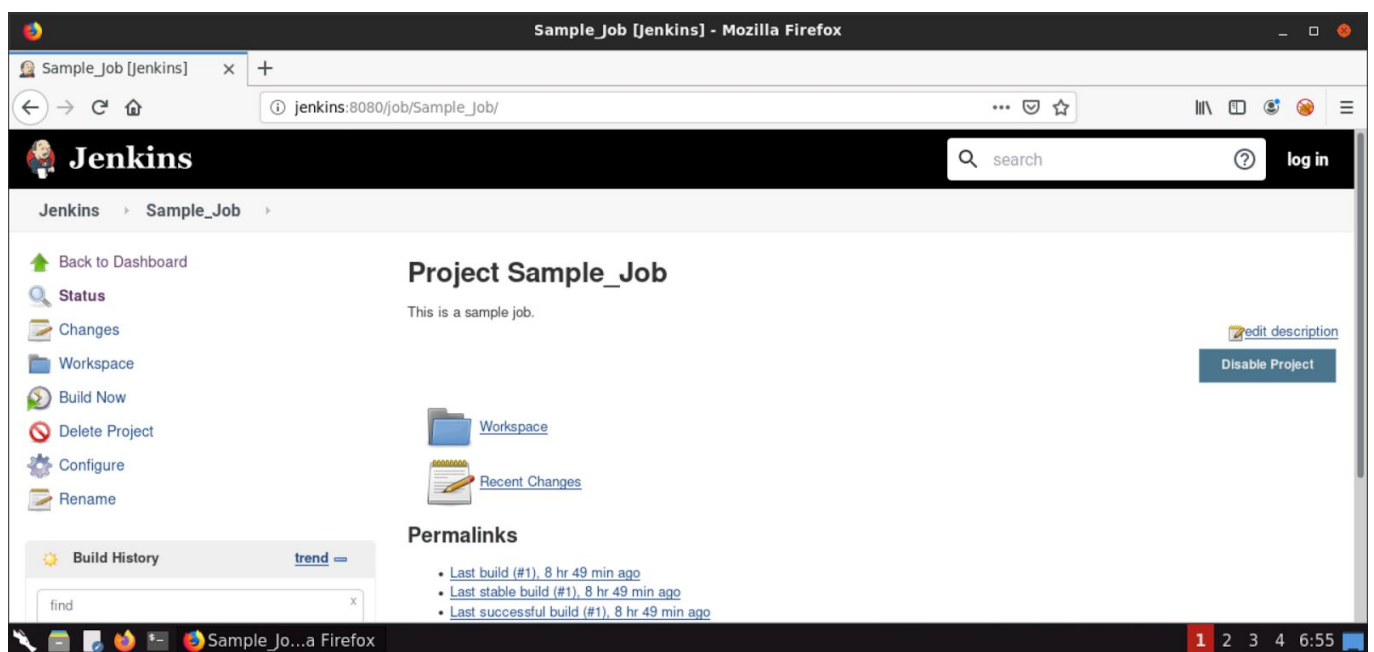
**Step 1:** Open the web browser and navigate to jenkins UI.

**URL:** <http://jenkins:8080>

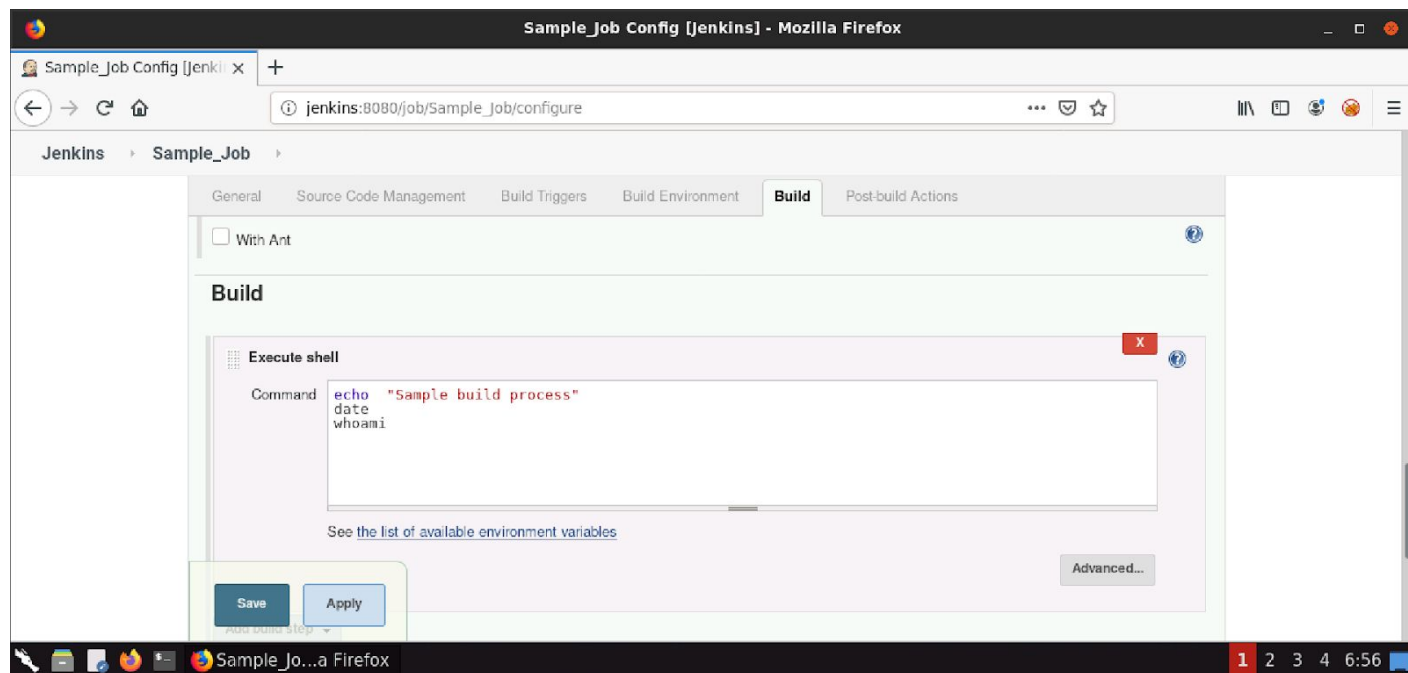


There is one project listed on the dashboard.

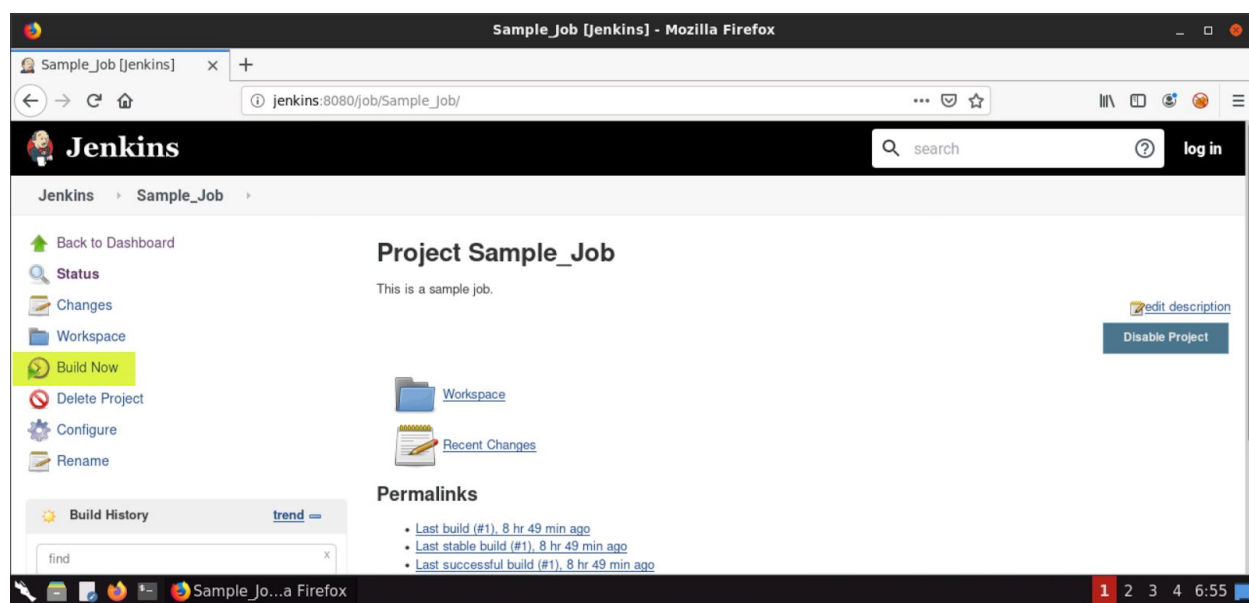
**Step 2:** Select the project to check the project details.



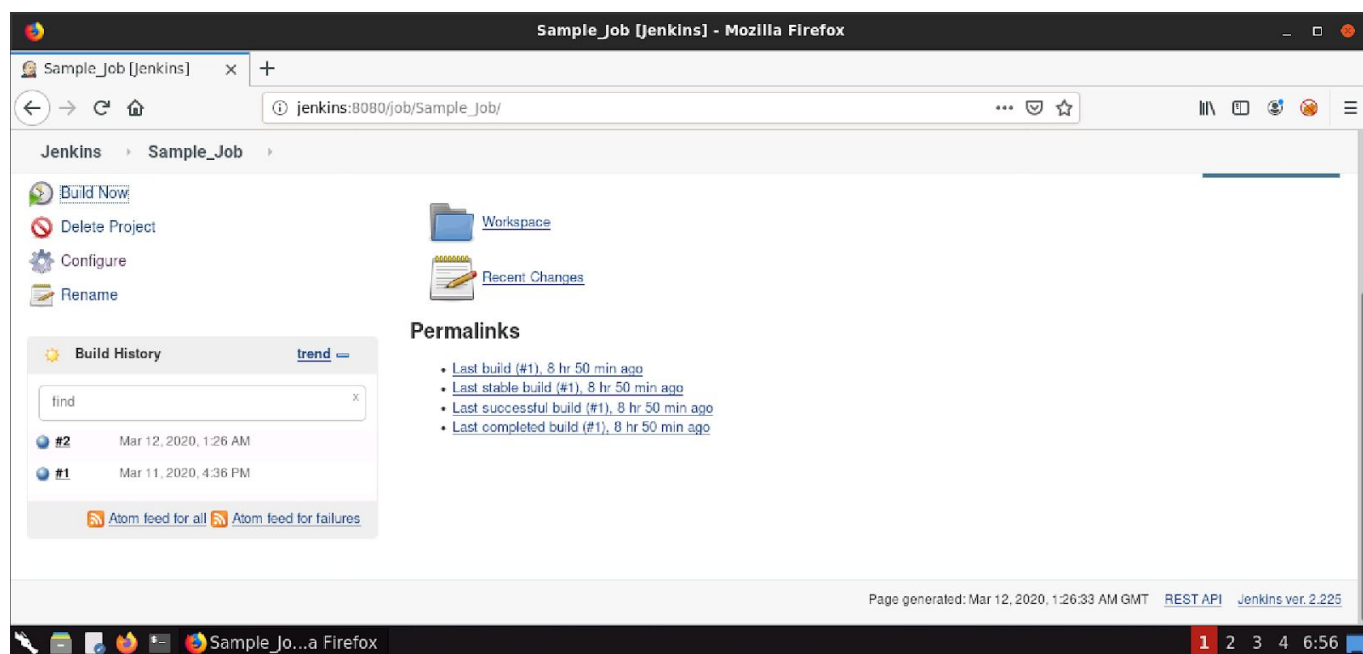
**Step 3:** Check the build instructions for the project.



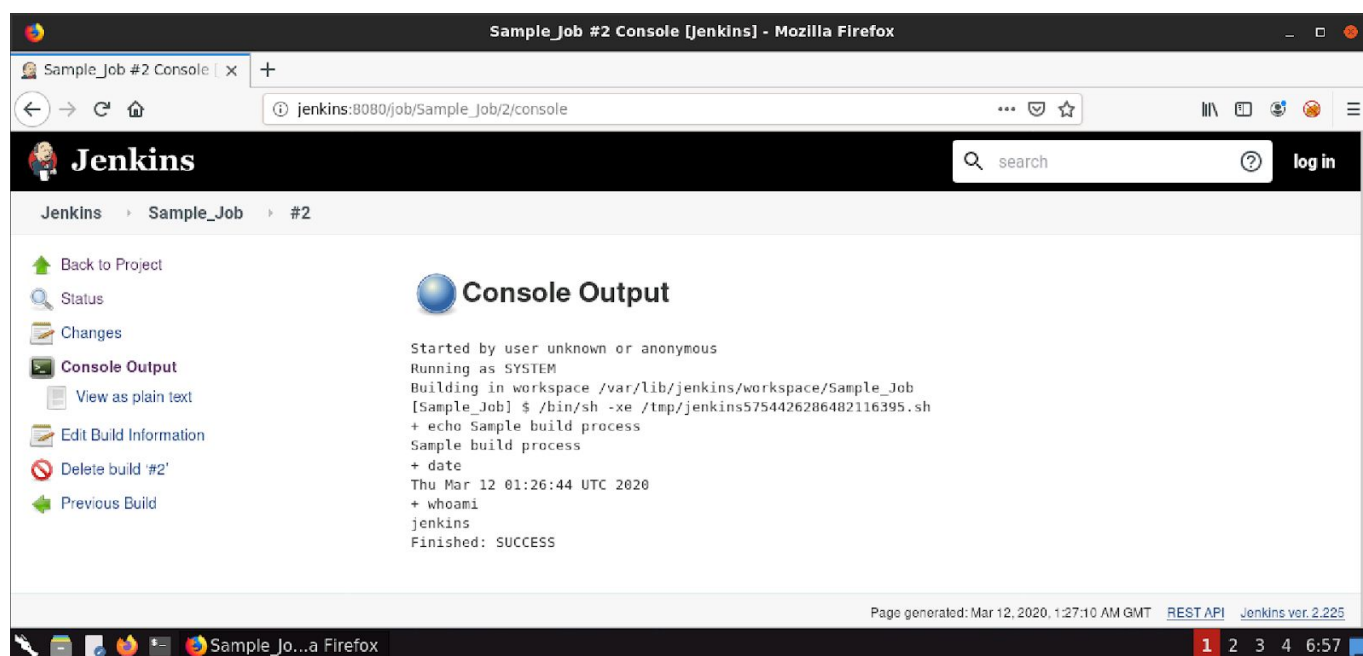
**Step 4:** Return to the project page and fire a new build by clicking the highlighted button..



The Build #2 is complete.



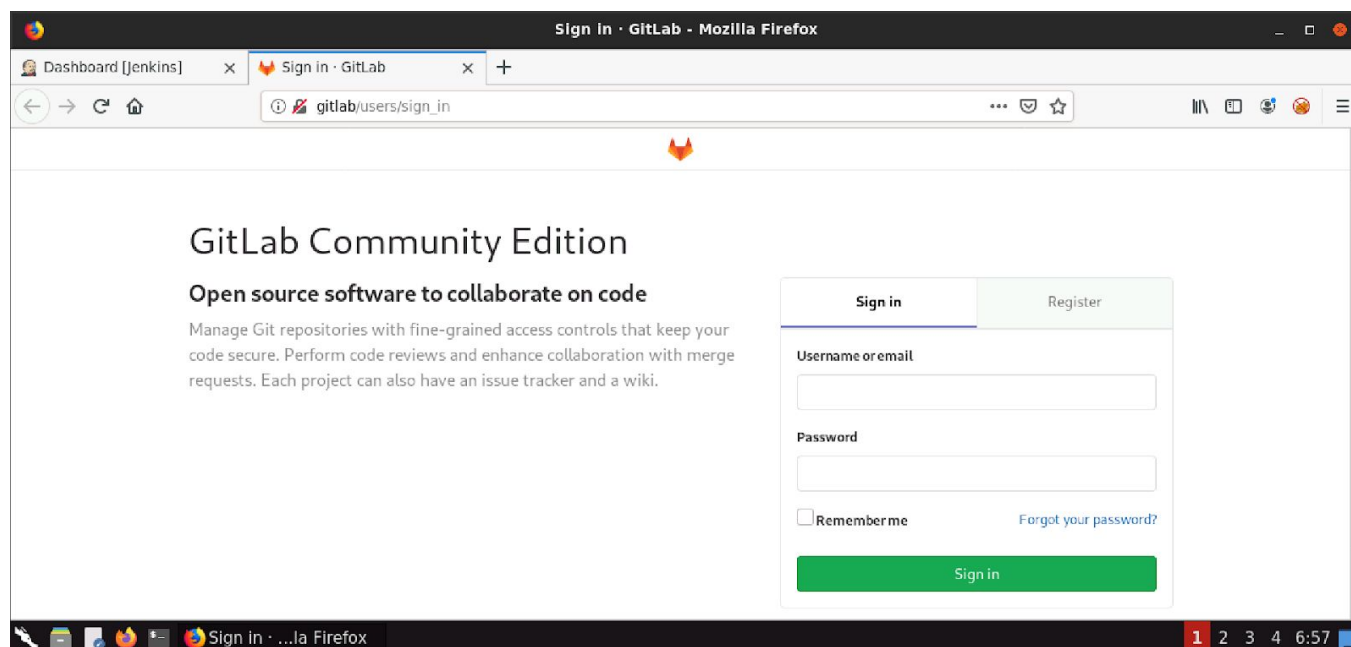
Click on #2 build entry to view the build page. Click on "Console Output" to view build logs.



## Task II: Create a building project for a sample-web-server project hosted on local Gitlab instance

**Step 5:** Open a new tab in Firefox and browse to gitlab

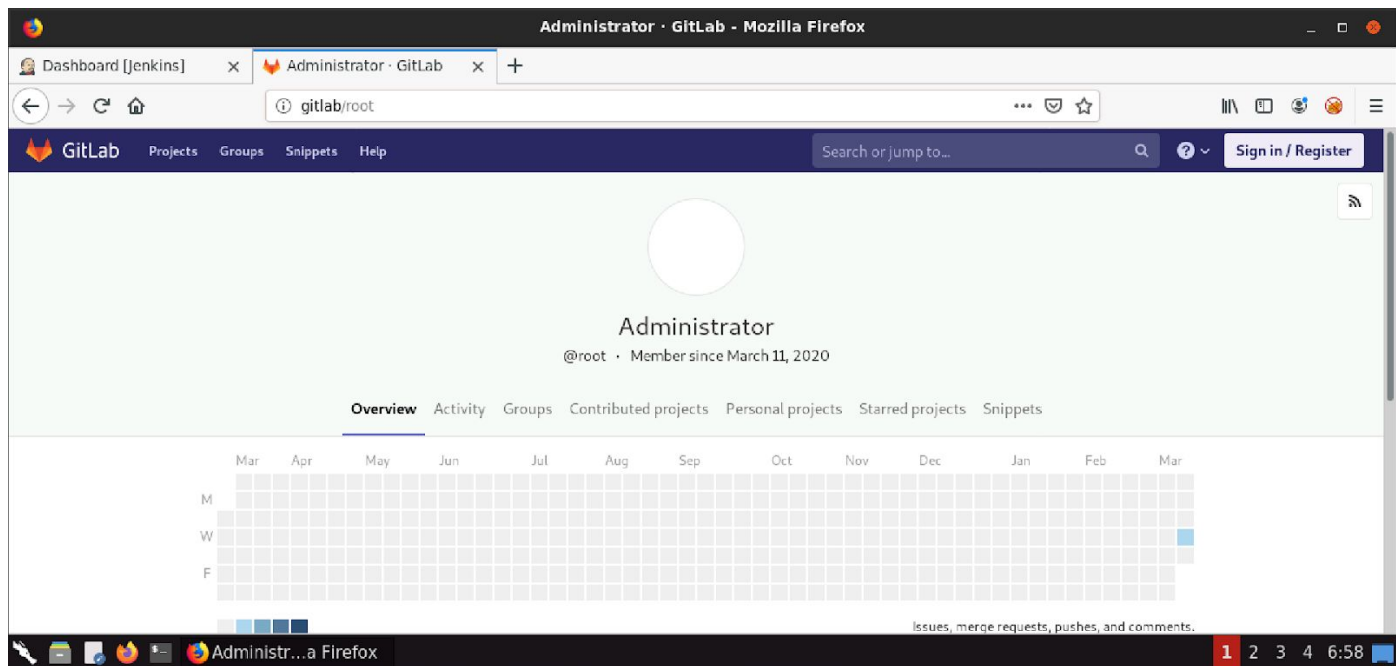
**URL:** <http://gitlab/>



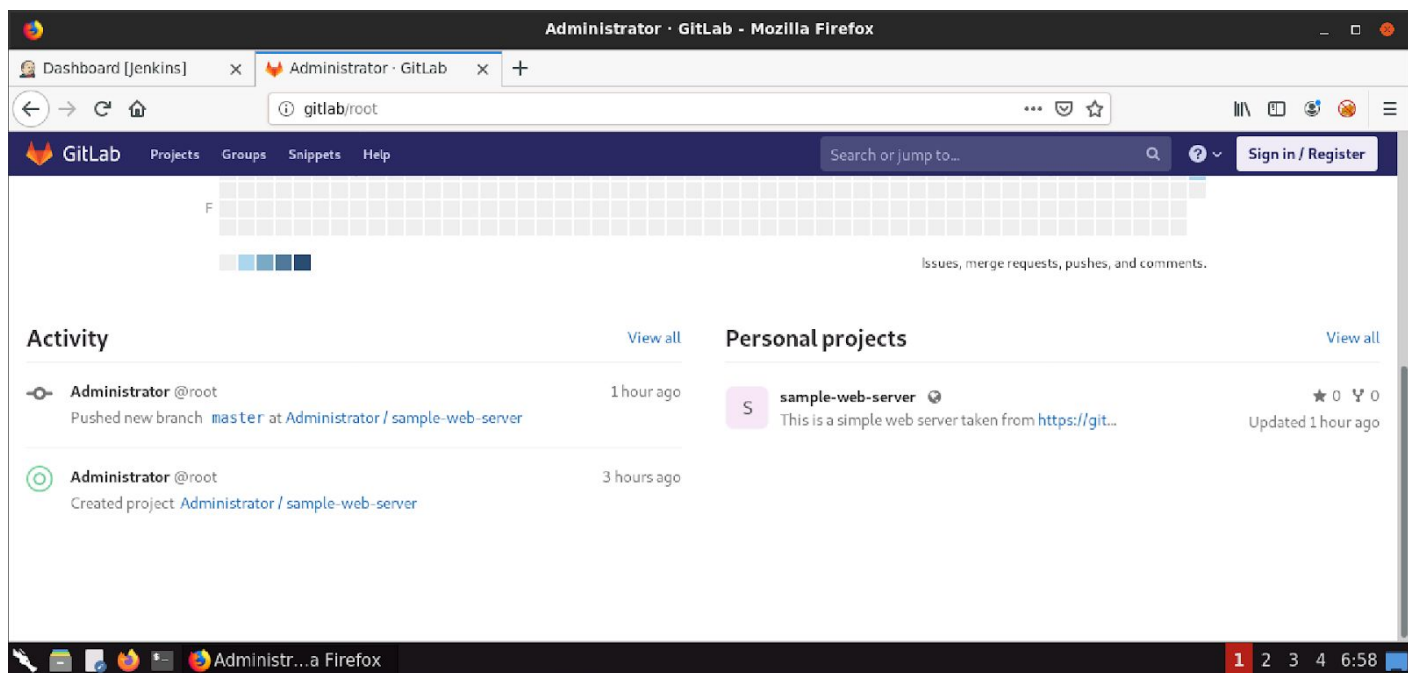
**Step 6:** As mentioned in the guidelines, the project belongs to user “root”. Check the profile page of the root user.

**URL:** <http://gitlab/root>

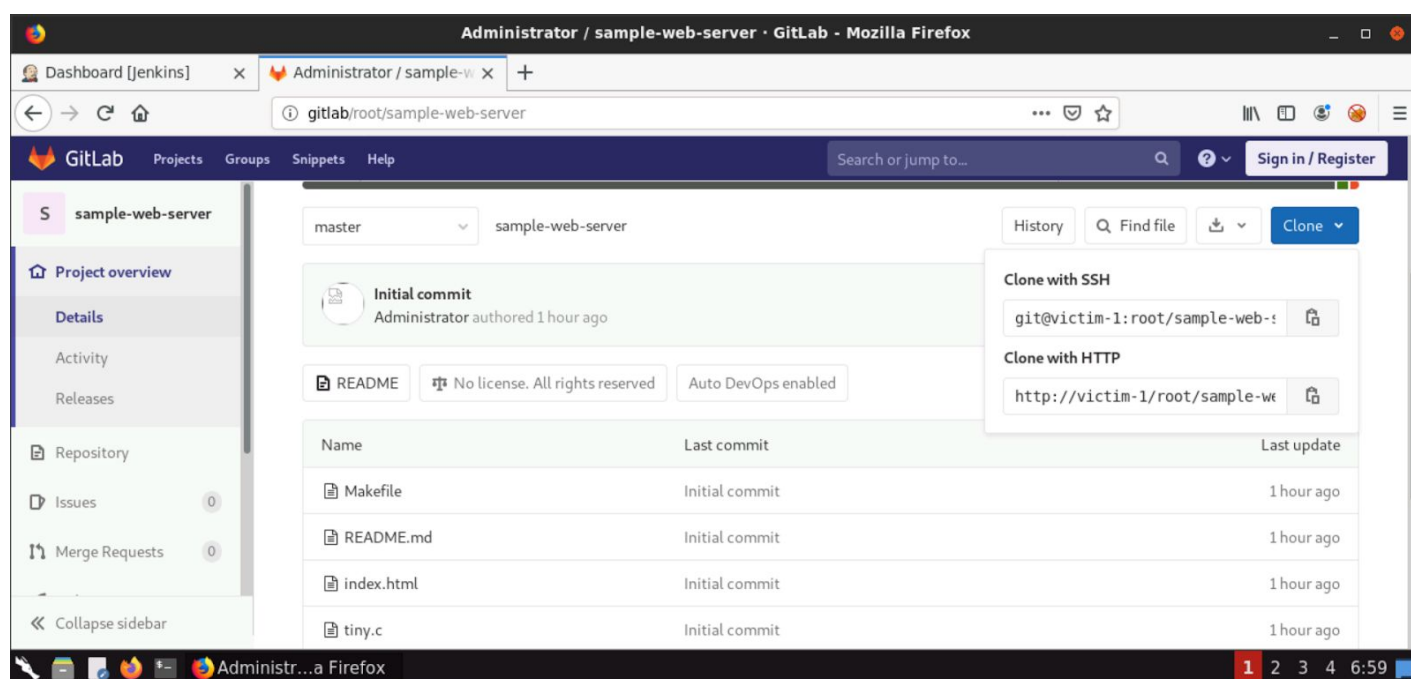
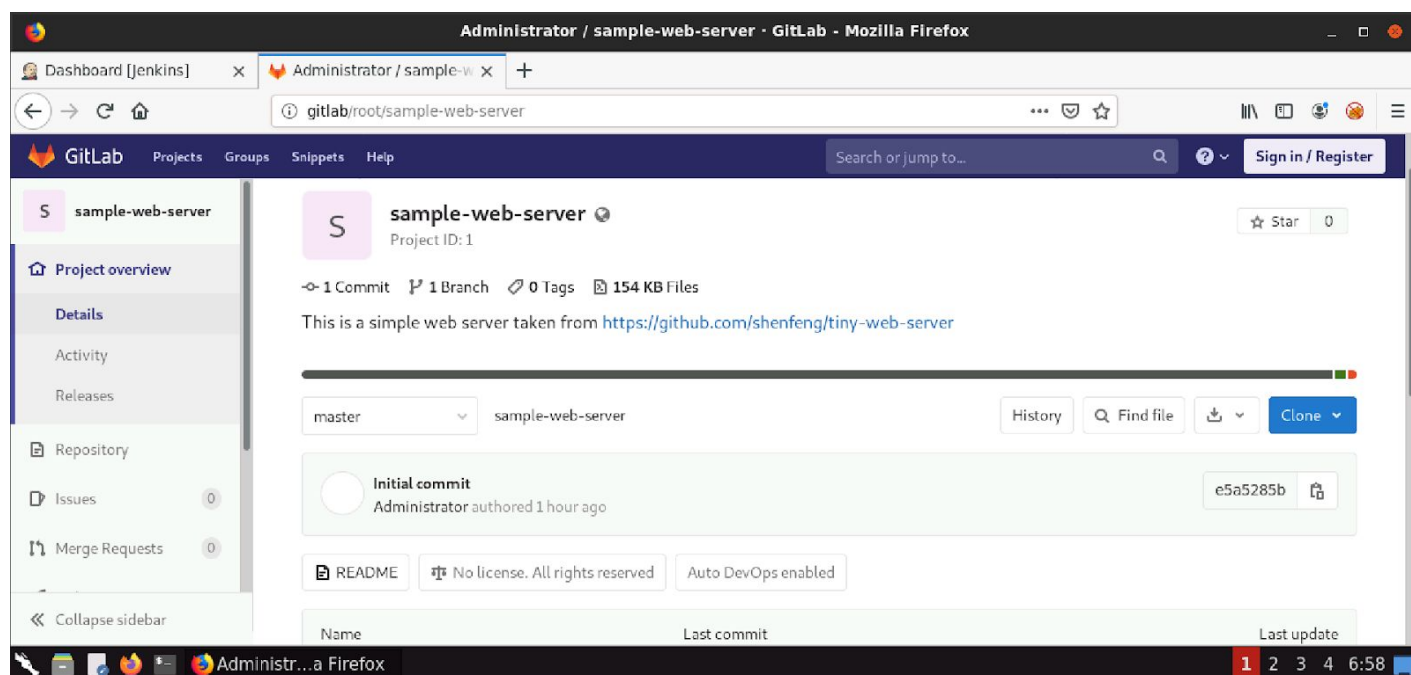




The project is also listed on the same page.

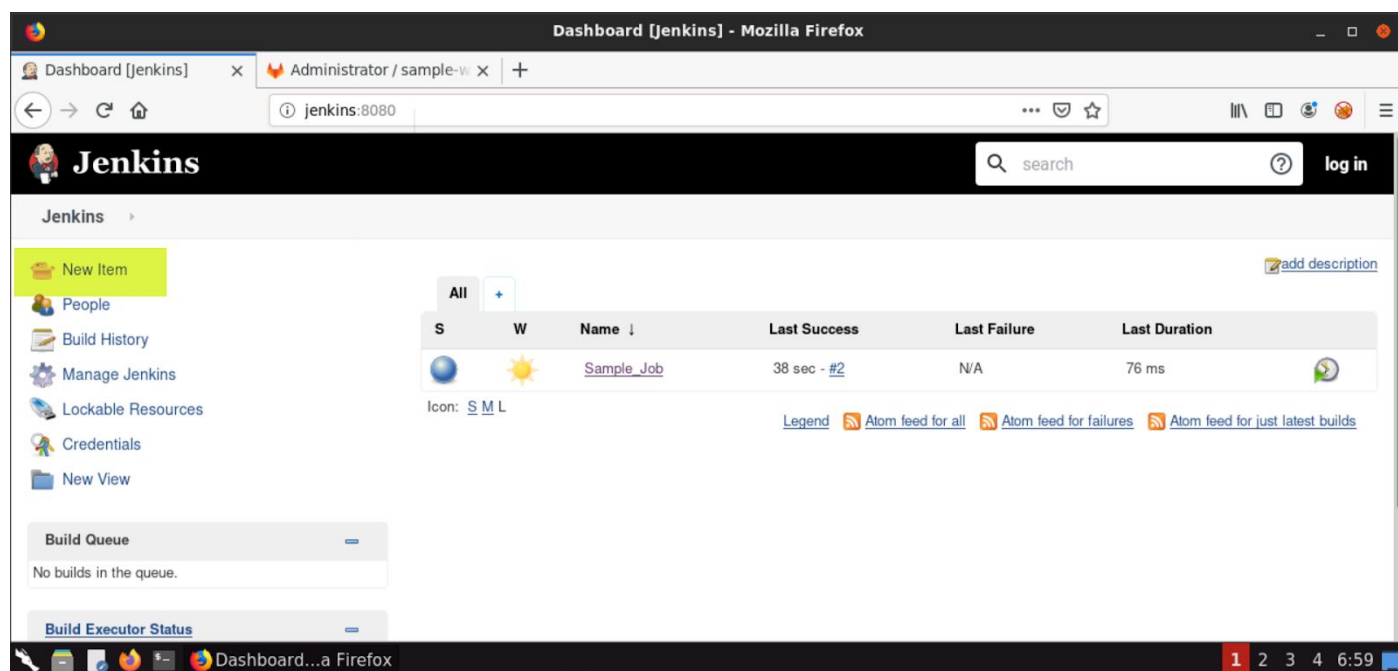


**Step 7:** Open the gitlab project in the browser and check the links for cloning this repository.

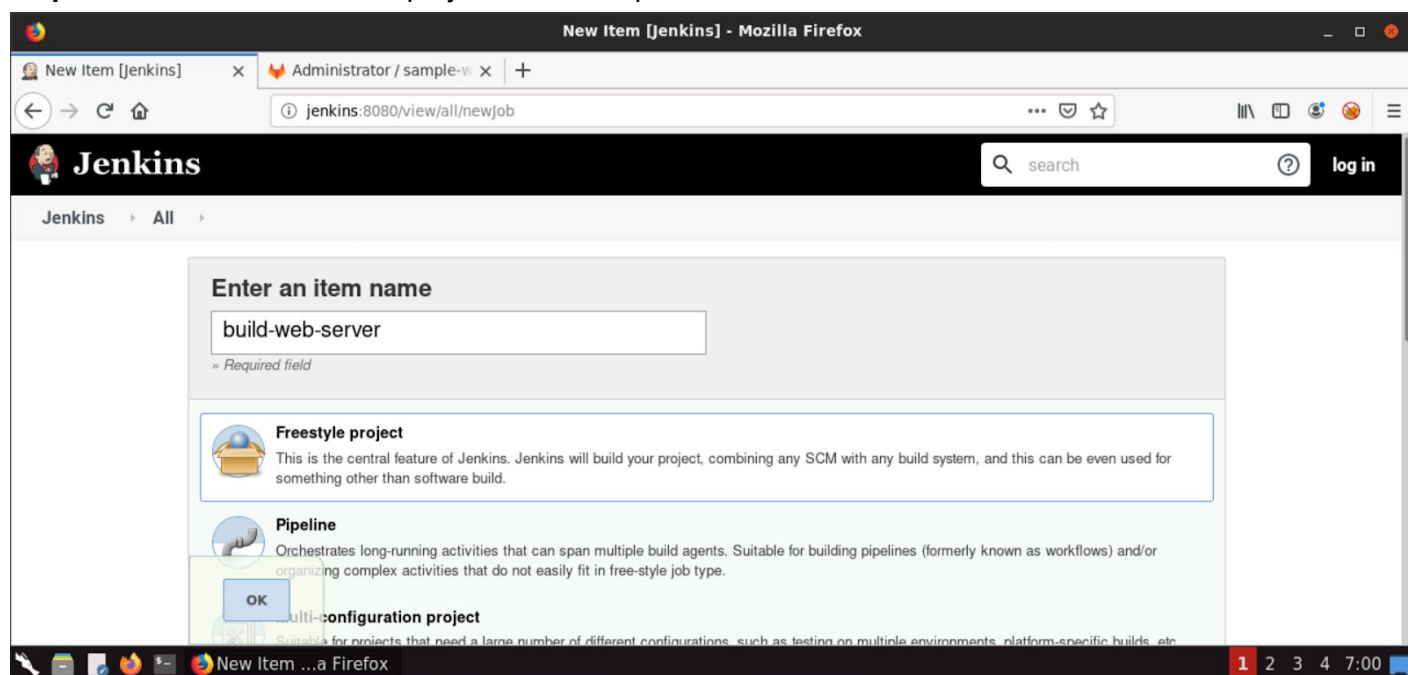




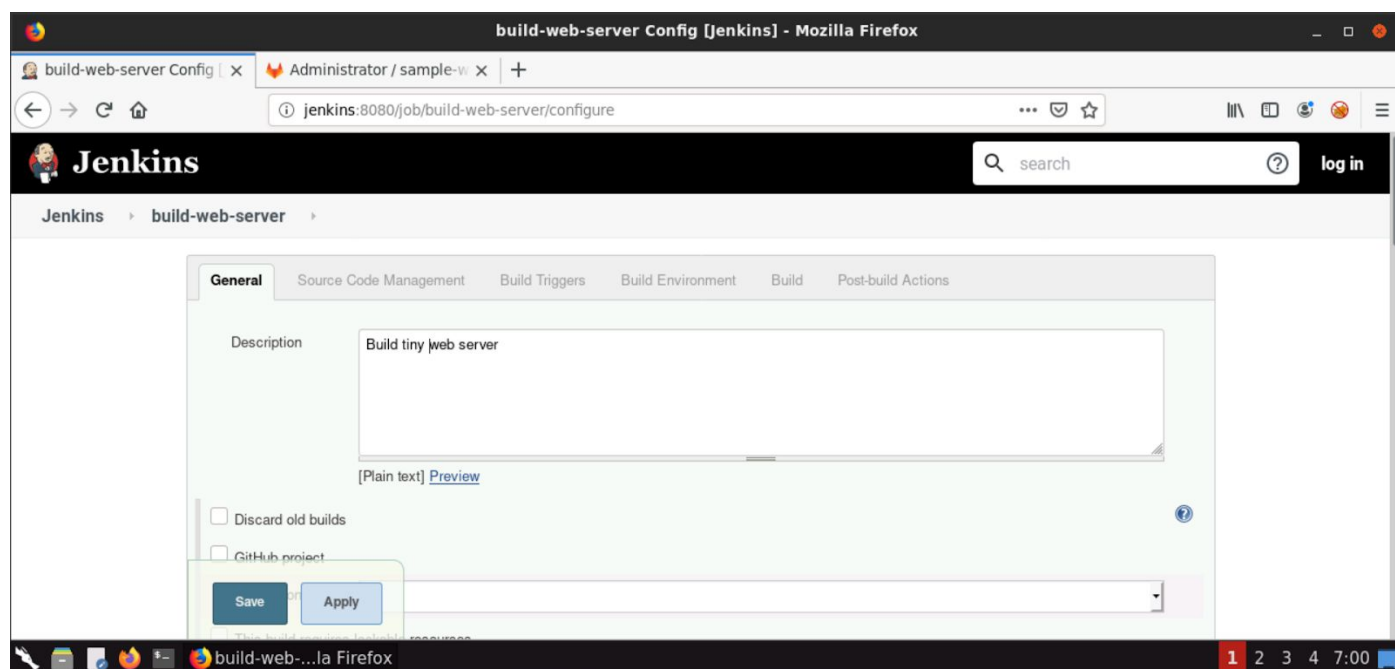
**Step 8:** Add a new project by clicking on “New Item” option in left hand menu.



**Step 9:** Provide a name to the project. For example: build-web-server.



**Step 10:** Add description to the project.

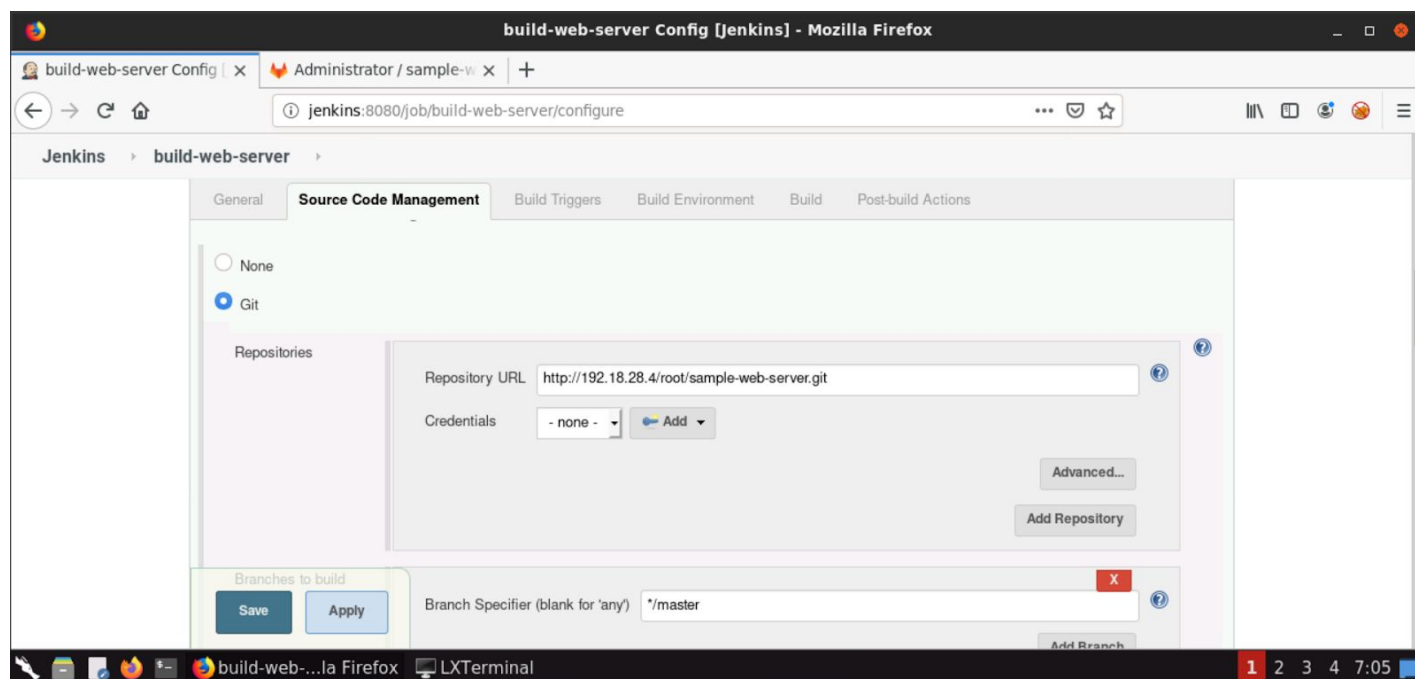


**Step 11:** The domain based git clone is not reliable in this setup. So, check the IP address for that gitlab domain. This IP can be then used to clone the git repo by Jenkins.

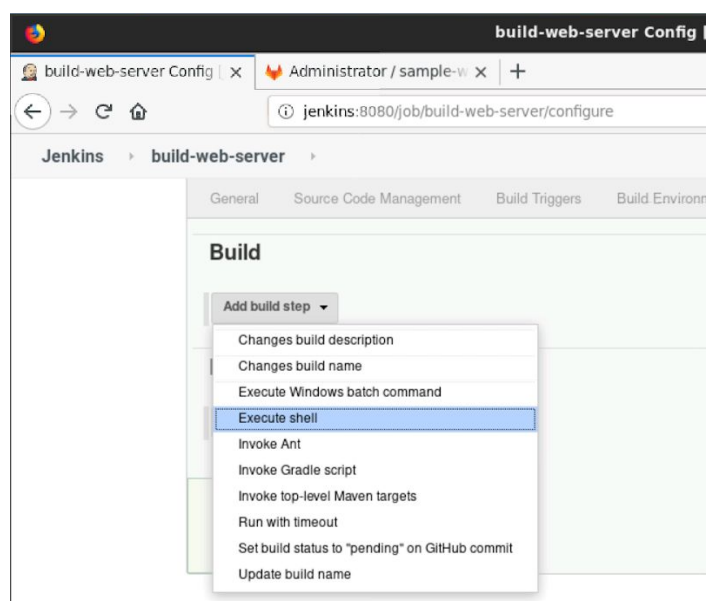
**Command:** ping gitlab

```
root@attackdefense:~# ping gitlab
PING gitlab (192.18.28.4) 56(84) bytes of data.
64 bytes from target-2 (192.18.28.4): icmp_seq=1 ttl=64 time=0.074 ms
64 bytes from target-2 (192.18.28.4): icmp_seq=2 ttl=64 time=0.053 ms
64 bytes from target-2 (192.18.28.4): icmp_seq=3 ttl=64 time=0.068 ms
^C
--- gitlab ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2051ms
rtt min/avg/max/mdev = 0.053/0.065/0.074/0.008 ms
root@attackdefense:~#
```

**Step 12:** Define the HTTP URL to gitlab repo. Please remember to use IP address instead of the “gitlab” domain.



**Step 13:** Add build steps to the project. The Jenkins is running on Linux OS, hence the shell commands need to be user.



**Step 14:** Add the following build steps to the project.

**Build instructions:**

make

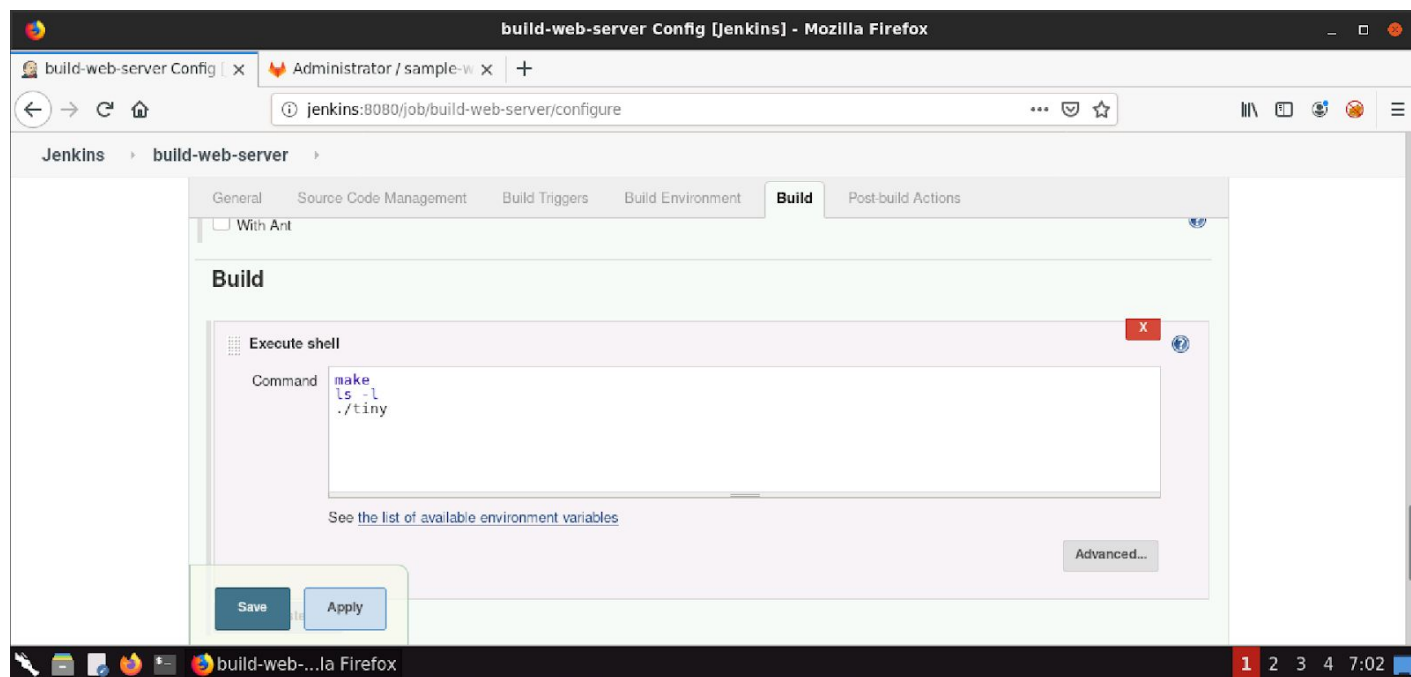
ls -l

./tiny

make instructions will build the C project and generate “tiny” binary.

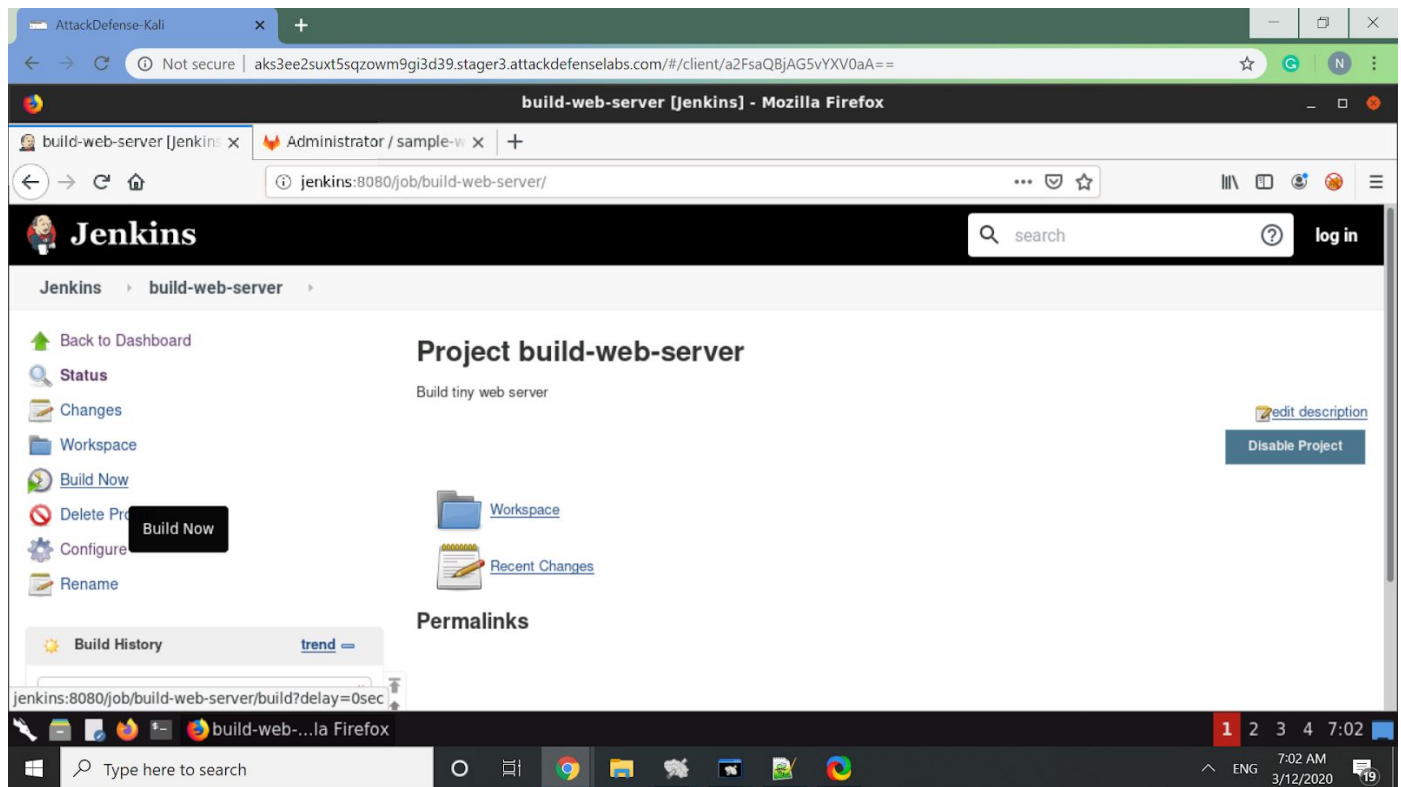
ls -l instructions will list the contents of the directory so one can verify the creation of the binary.

./tiny executes the newly generated binary

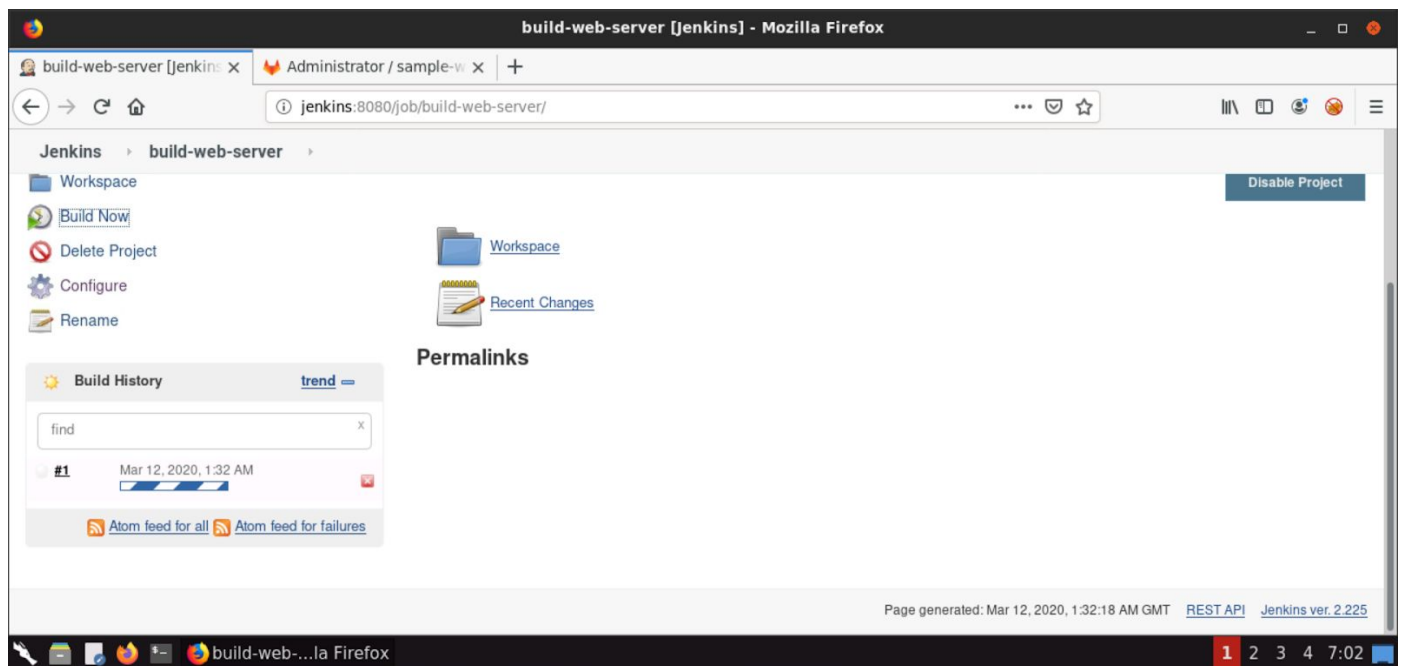


**Step 15:** Go to the project page and fire a build.



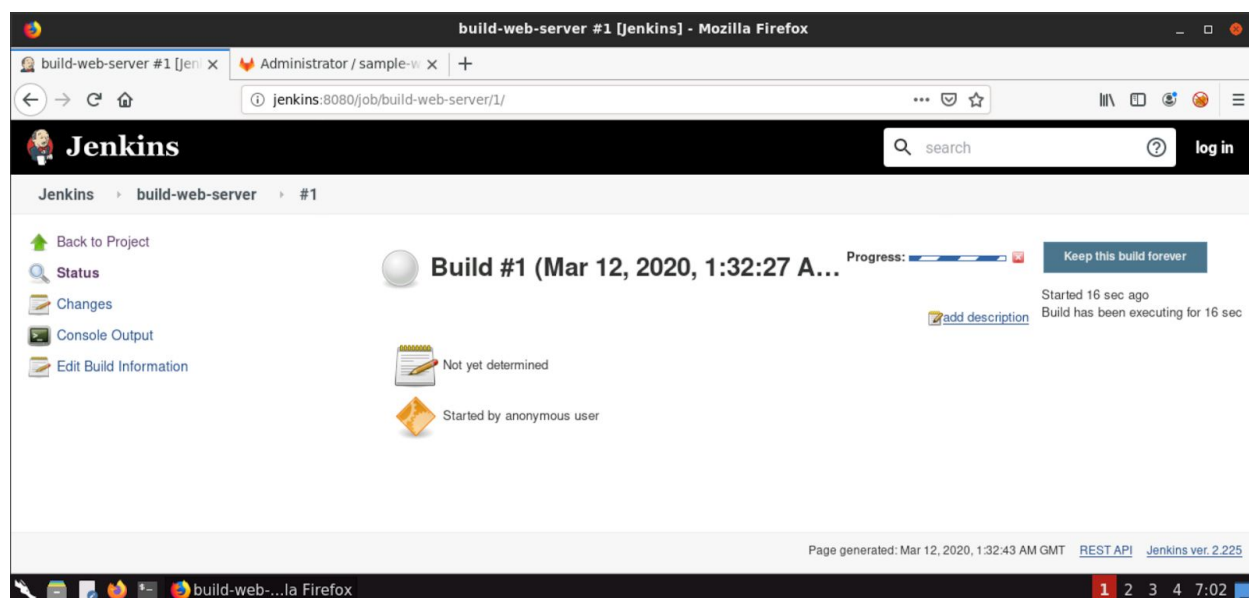


The running build job should be visible under Build History.

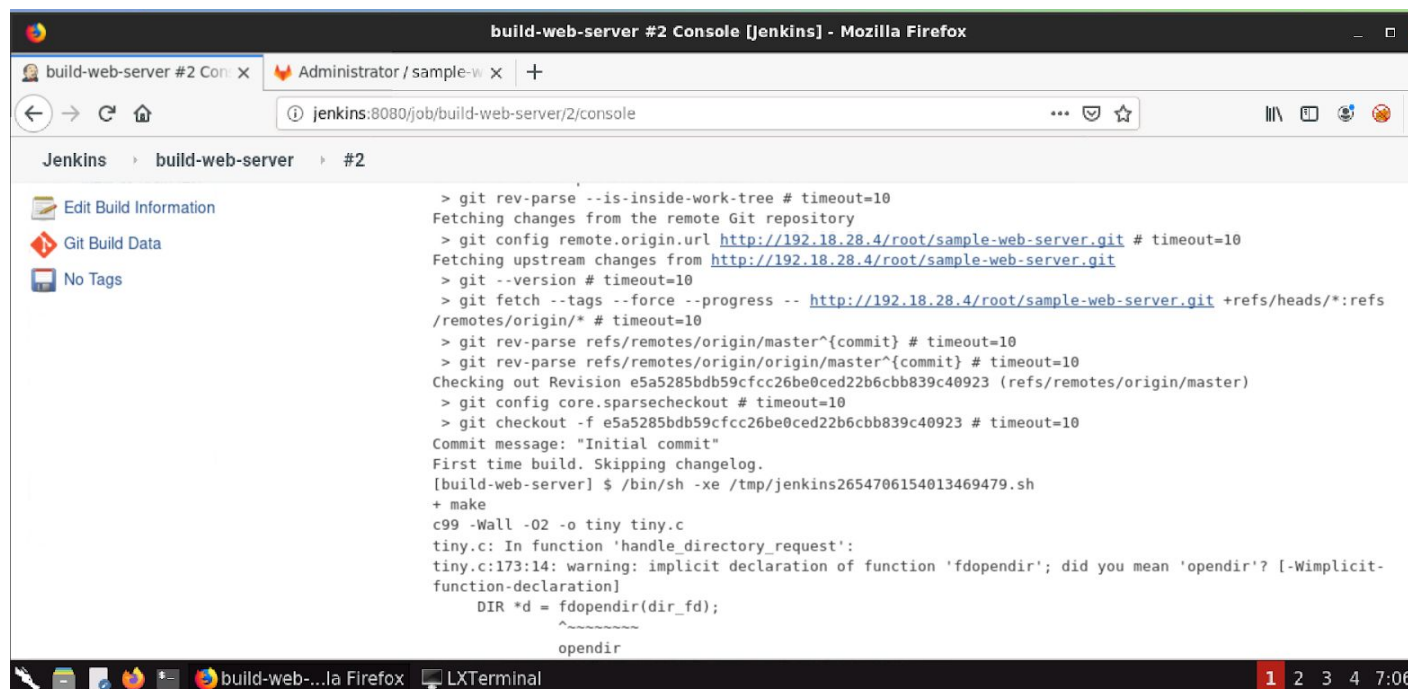


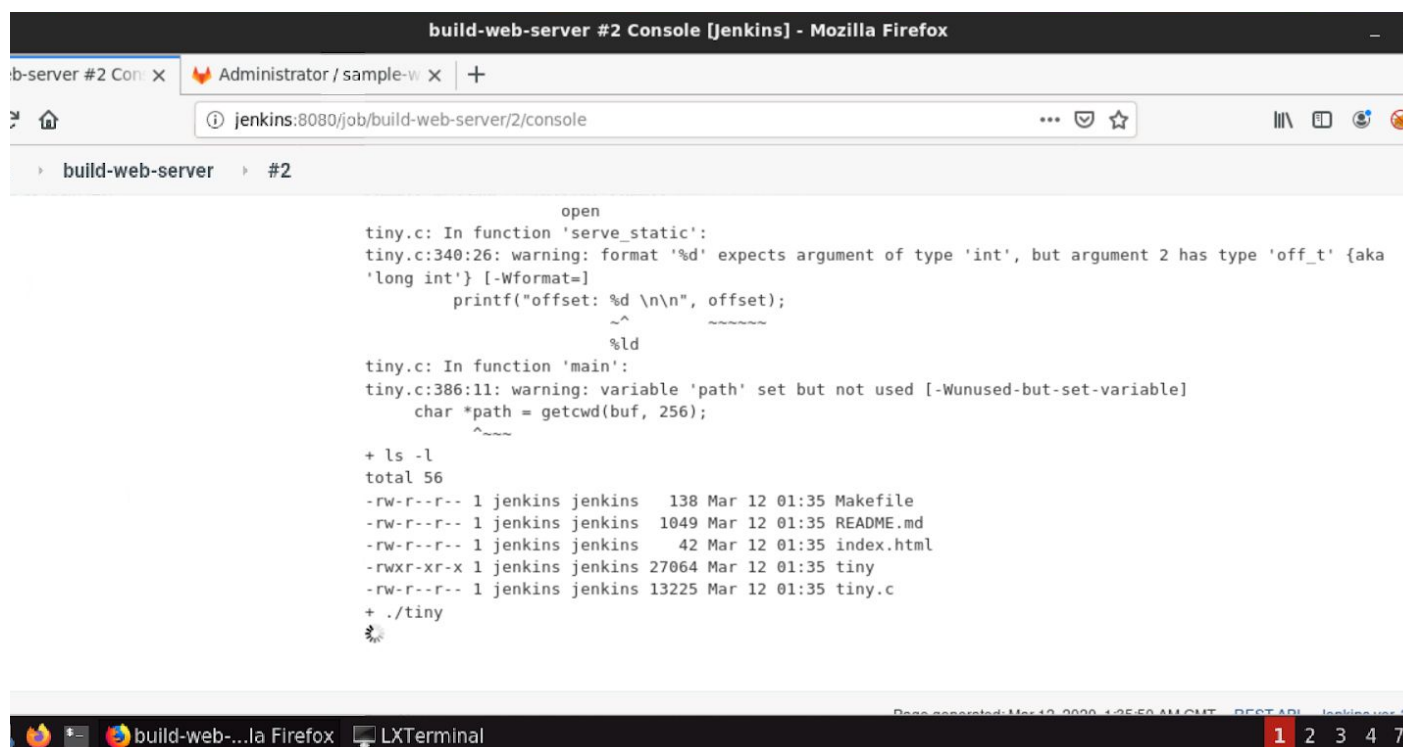


**Step 16:** Click on the build #1 to view the build page.



Click on "Console output" to check the build logs.





The image shows two screenshots. The top screenshot is a browser window titled 'build-web-server #2 Console [Jenkins] - Mozilla Firefox' displaying the Jenkins console output for job 'build-web-server' build #2. The output shows compilation warnings for 'tiny.c' and the execution of 'ls -l' command. The bottom screenshot shows a terminal window with tabs for 'build-web-server #2 Console' and 'LXTerminal'. The terminal shows the execution of the 'tiny' binary.

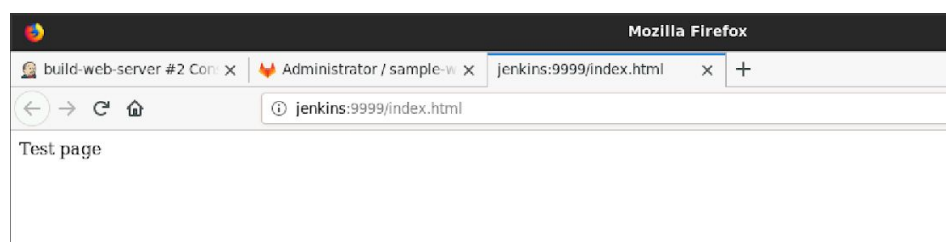
```
open
tiny.c: In function 'serve_static':
tiny.c:340:26: warning: format '%d' expects argument of type 'int', but argument 2 has type 'off_t' {aka
'long int'} [-Wformat=]
    printf("offset: %d \n\n", offset);
                        ^~
                        %ld
tiny.c: In function 'main':
tiny.c:386:11: warning: variable 'path' set but not used [-Wunused-but-set-variable]
    char *path = getcwd(buf, 256);
            ^~~~


+ ls -l
total 56
-rw-r--r-- 1 jenkins jenkins 138 Mar 12 01:35 Makefile
-rw-r--r-- 1 jenkins jenkins 1049 Mar 12 01:35 README.md
-rw-r--r-- 1 jenkins jenkins 42 Mar 12 01:35 index.html
-rwxr-xr-x 1 jenkins jenkins 27064 Mar 12 01:35 tiny
-rw-r--r-- 1 jenkins jenkins 13225 Mar 12 01:35 tiny.c
+ ./tiny
```

It is clear from the build logs that the code was successfully pulled from the Gitlab server and the tiny binary is generated by make command. The last instruction of the build process has also executed the binary.

The tiny binary is a web server which listens on port 9999 by default. And, an index.html in the code directory of this web server. So, one can access the index.html on port 9999 of the Jenkins machine.

**URL:** <http://jenkins:9999/index.html>





In this manner, the Jenkins and Gitlab combo can be used to create automated build systems.

**References:**

1. Jenkins Documentation (<https://jenkins.io/doc/>)
2. Gitlab Documentation (<https://docs.gitlab.com/>)