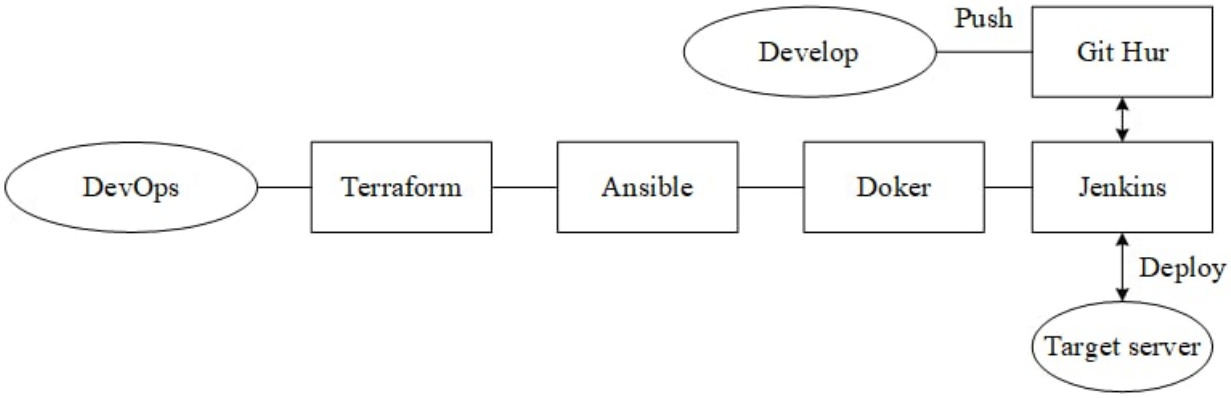
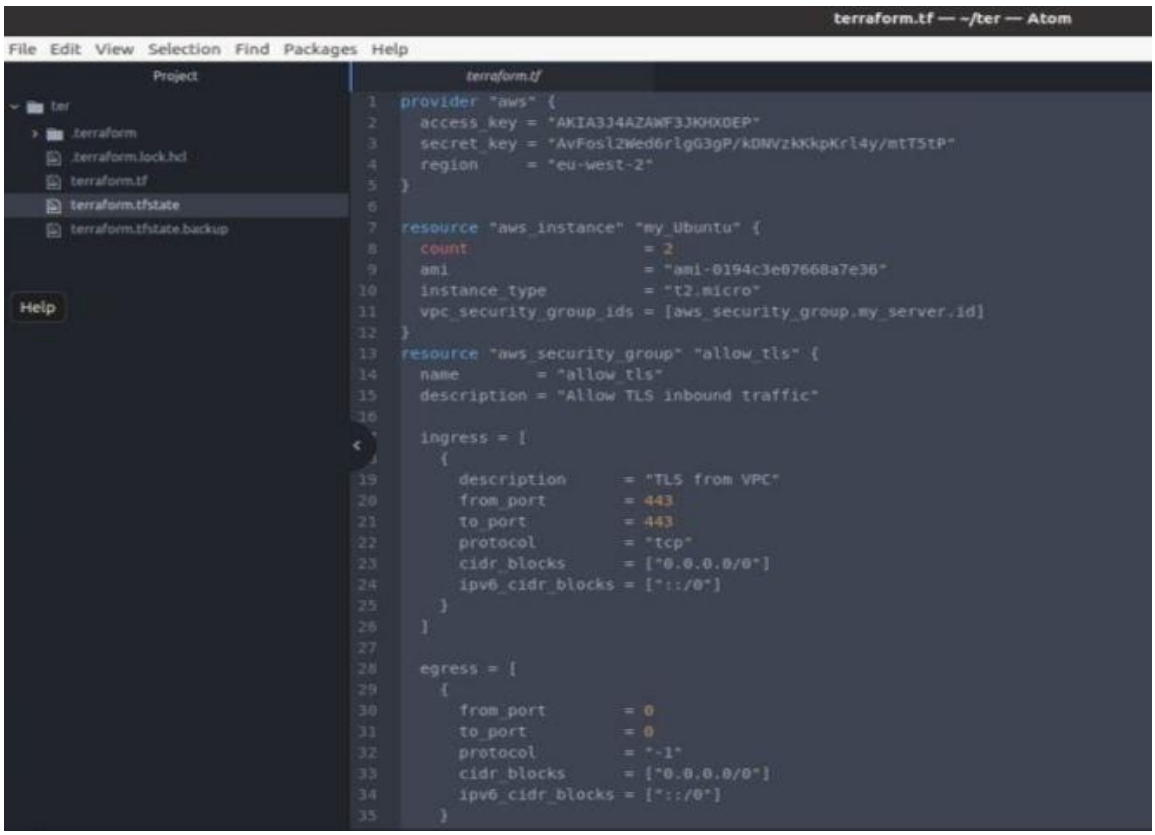


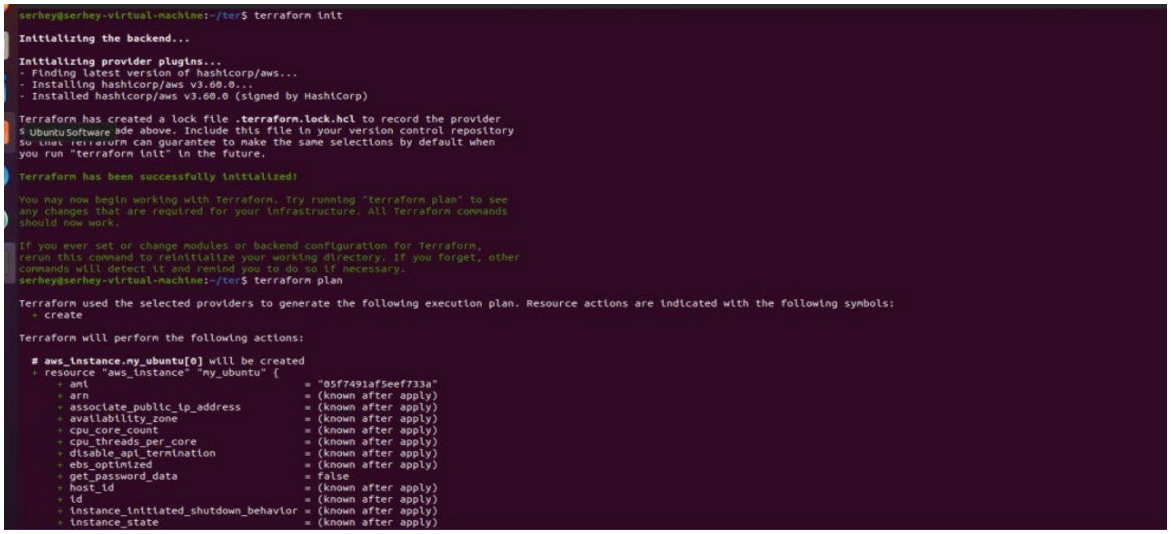
Step 1 - Launching Machines with Terraform

Terraform maps the resources described in the config file to the corresponding resources of the cloud provider. This mapping is called state, it's a giant JSON file. When terraform apply runs, Terraform updates the state by forwarding the appropriate request to the cloud provider. It then compares the returned assets with the information recorded in your Terraform configuration. If any difference is found, then a plan is created, in essence - a list of changes that need to be made to the resources of the cloud provider so that the actual configuration matches the one specified in your configuration. Finally, Terraform applies these changes by making the appropriate calls to the cloud provider.

On the Host machine, create a file with which we will create machines in AWS with which we will work further, the file must be in the format ".tf".



The terraform init command is used to initialize the working directory containing the Terraform configuration files. This is the first command to run after writing a new Terraform configuration or cloning an existing one from source control.

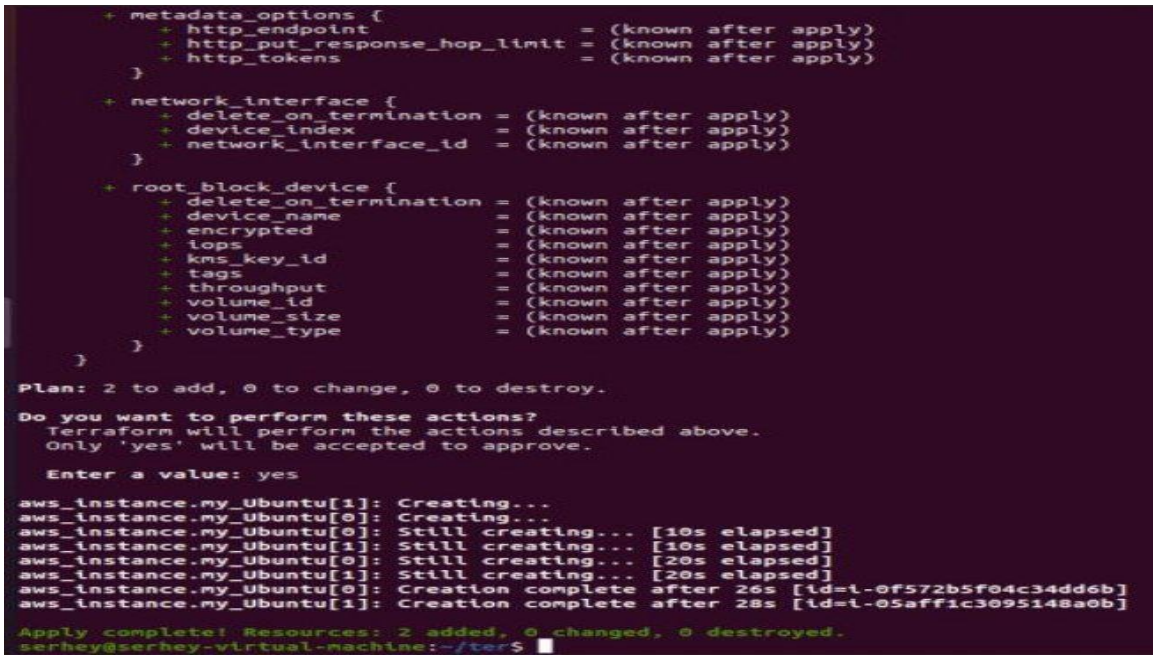


The terraform plan command creates an execution plan. By default, plan creation consists of:

• Read the current state of any pre-existing remote objects to ensure that the state of Terraform is up to date.

• Compare the current configuration with the previous state and identify any differences.

• Proposing a set of change actions that should, if applied, bring the remote objects into line with the configuration.

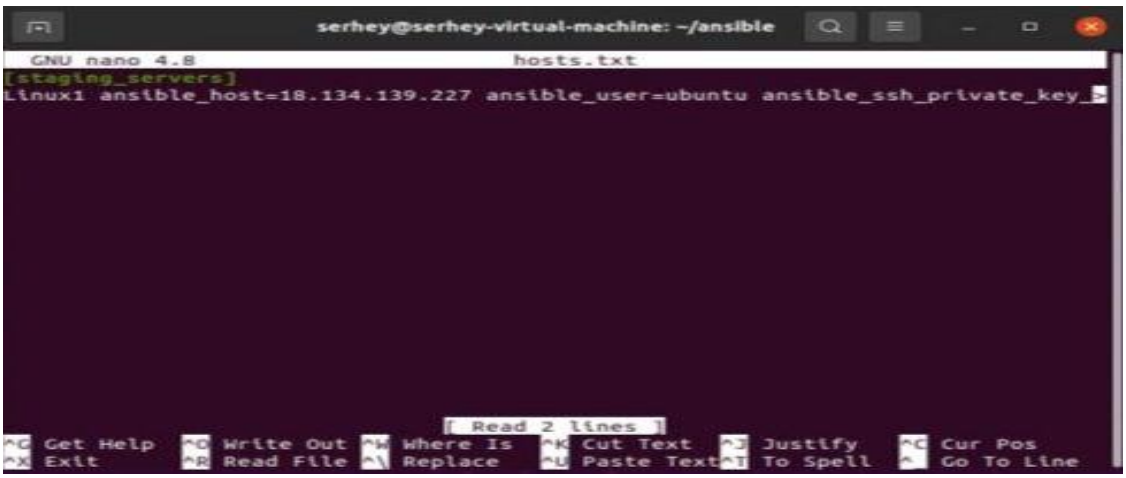


The terraform apply command performs the actions suggested in the Terraform plan. In our case, it creates virtual machines and a security system .

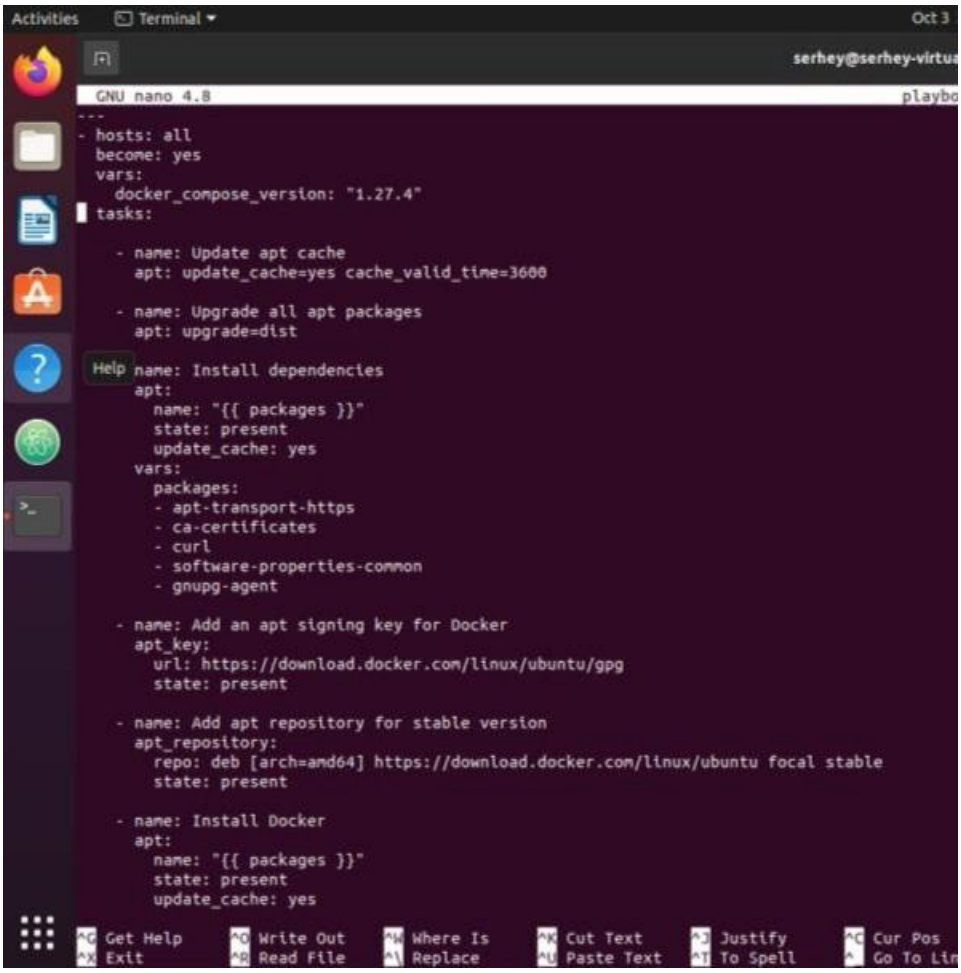
Step 2. Initial setup of the project environment. Ansible Installing Docker

Ansible is a simple configuration management tool that automates I resource management, application deployment or cloud provisioning, and more.

Create a file in which we write the data of the machine on which we will configure

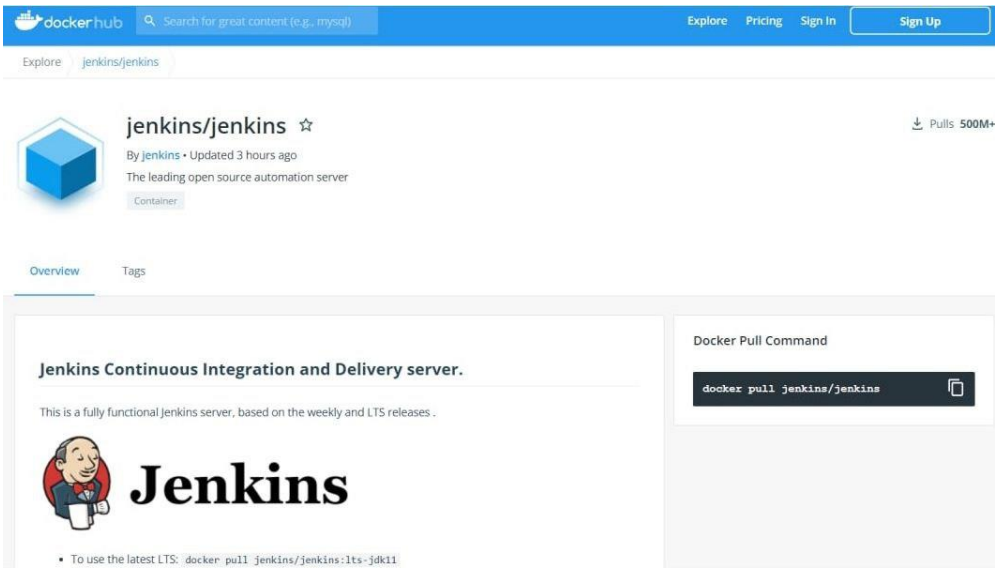


Ansible playbooks are a way to send commands to remote computers using scripts. Instead of individually using commands to remotely configure computers from the command line, we set up the entire installation process by passing a script to one or more systems.

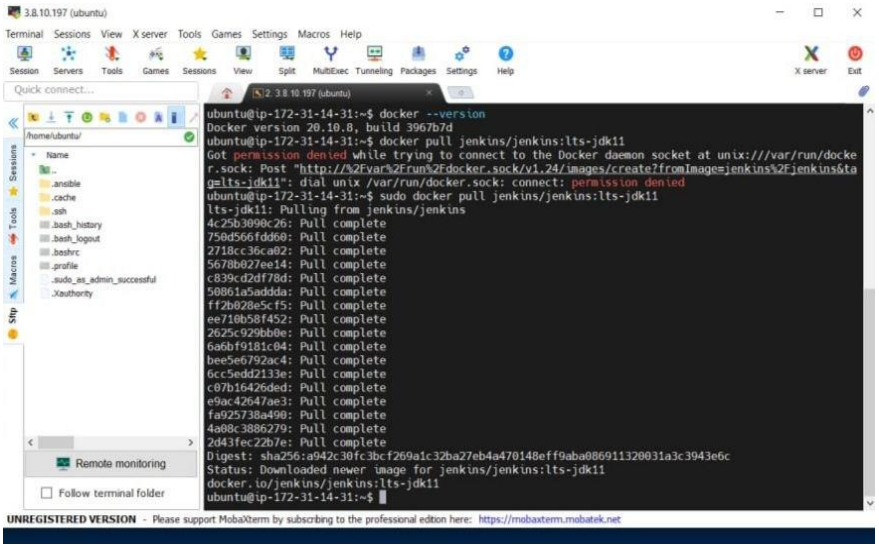


Step 3 - Running an isolated system with Docker

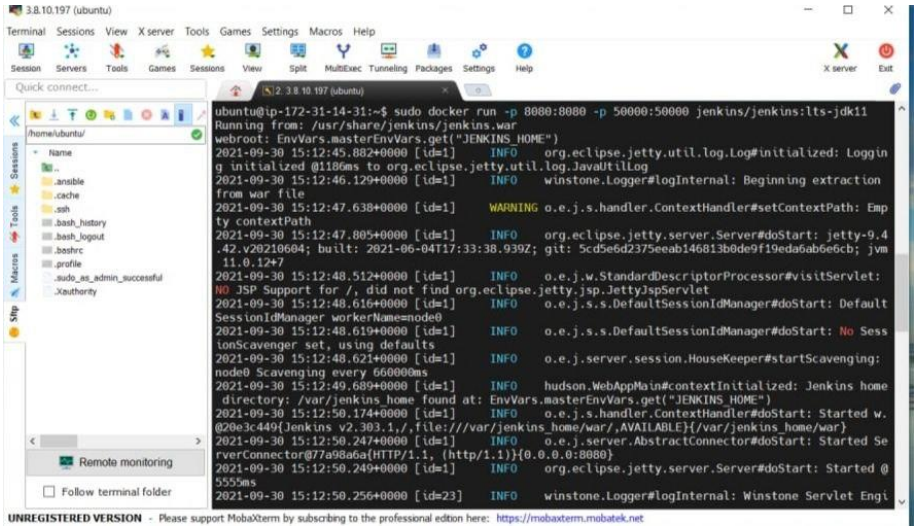
Docker is a software platform for rapidly developing, testing, and deploying applications. Docker packages software into standardized blocks called containers. Each container contains everything you need to run your application: libraries, system tools, code, and the runtime environment. With Docker, you can quickly deploy and scale applications in any environment, and remain confident that your code will work.



Docker Hub is a public registry maintained by Docker, Inc. It contains images that you can download and use to create containers. And also provides authentication, workgroup structure, workflow tools. Find the container we need and launch it.



Run the command that will automatically create the jenkins_home docker volume on the host machine. Docker volumes retain their contents even when the container is stopped, started, or deleted.



Step 4.Jenkins

Jenkins is an open source Java software system designed to provide a continuous software integration process.

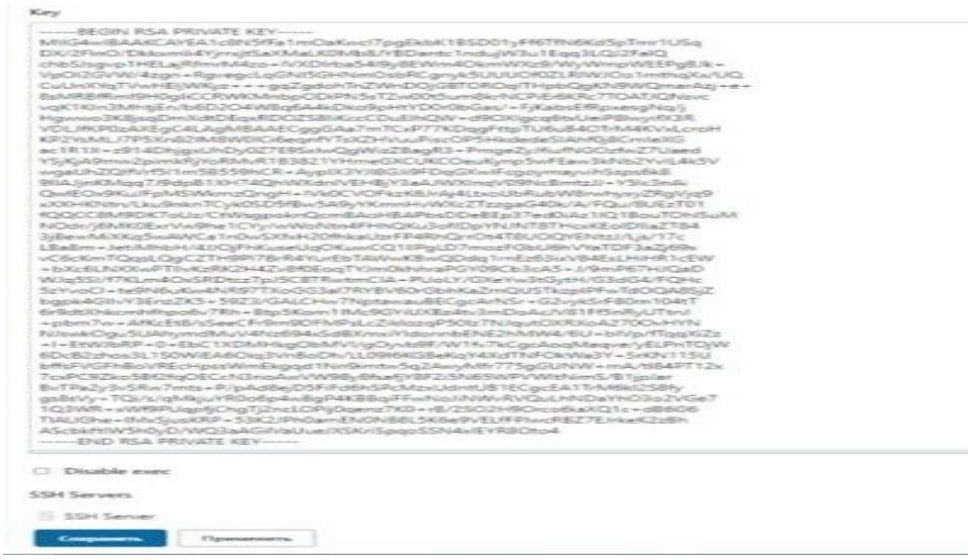
The main purpose of jenkis in our project is to pull the updated project from the hub and build it, and then deploy it on our server

We connect via ssh to the server to which the Deploy will be performed

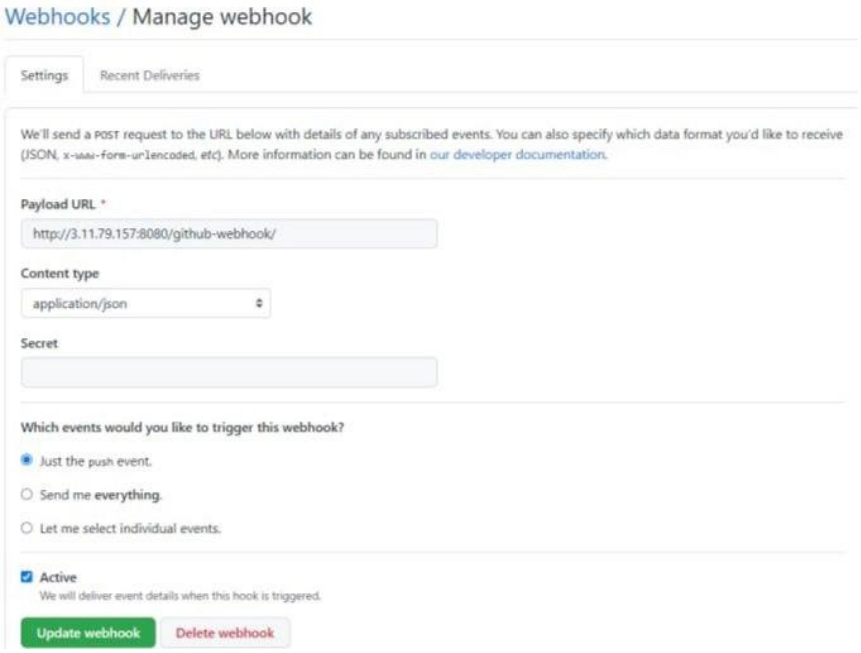
```
  Memory usage: 23%               IPv4 address for eth0: 172.31.2.63
  Swap usage:   0%


1 update can be applied immediately.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

$  sudo apt update
[sudo] password for student:
student is not in the sudoers file.  This incident will be reported.
$ exit
Connection to 3.9.188.145 closed.
ubuntu@ip-172-31-14-31:~$ ssh-copy-id student@3.9.188.145
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ubuntu/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are a
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to in
student@3.9.188.145's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'student@3.9.188.145'"
and check to make sure that only the key(s) you wanted were added.
```

Create a job in Jenkins and bind it to the cloud repository with GitHubWebHook

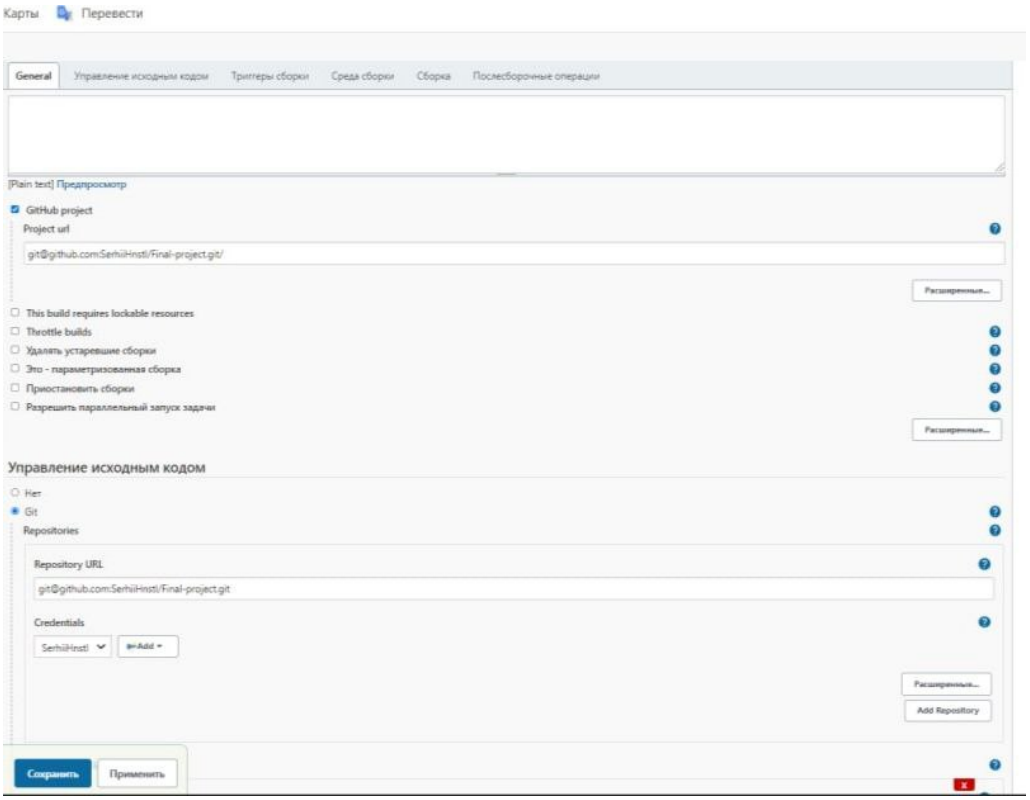
Copy the private key and check the connection



Configuring GitHub to connect to Jenkins



Set up the job so that it looks at the github and does a deploy



Initial view of our site:

Making changes and pushing:

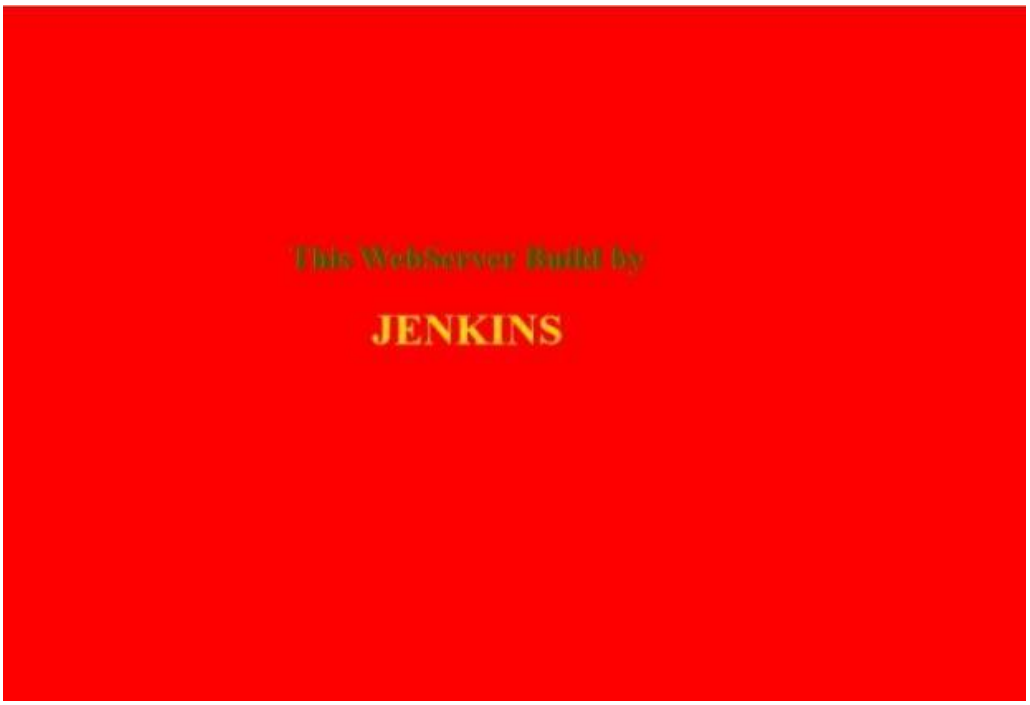
```
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\Иисус\Desktop\Final-project>git add .

C:\Users\Иисус\Desktop\Final-project>git commit -m "Commit1"
[main a9c58c0] Commit1
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\Иисус\Desktop\Final-project>git push origin
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 308 bytes | 308.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Heads up! The branch 'main' that you pushed to was renamed to 'master'.
remote:
To github.com:SerhiiHnstl/Final-project.git
   dc5263b..a9c58c0  main -> main

C:\Users\Иисус\Desktop\Final-project>
```

Jenkins starts automatically:

Вывод на консоль

```
Started by user Serhii Hons
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/github
The recommended git tool is: NONE
using credential github
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/github/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url git@github.com:SerhiiHnstl/Final-project.git # timeout=10
Fetching upstream changes from git@github.com:SerhiiHnstl/Final-project.git
 > git --version # timeout=10
 > git --version # 'git version 2.25.1'
using GIT_SSH to set credentials
 > git fetch --tags --force --progress -- git@github.com:SerhiiHnstl/Final-project.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 920501a1e44e81cdc08df317167503c6317de644 (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 920501a1e44e81cdc08df317167503c6317de644 # timeout=10
Commit message: "final"
 > git rev-list --no-walk 920501a1e44e81cdc08df317167503c6317de644 # timeout=10
[github] $ /bin/sh -xe /tmp/jenkins16827737627288S9911.sh
SSH: Connecting from host [ip-172-31-14-31]
SSH: Connecting with configuration [apache] ...
SSH: Disconnecting configuration [apache] ...
SSH: Transferred 2 file(s)
Finished: SUCCESS
```

Site after changes:



Thank you for the attention!