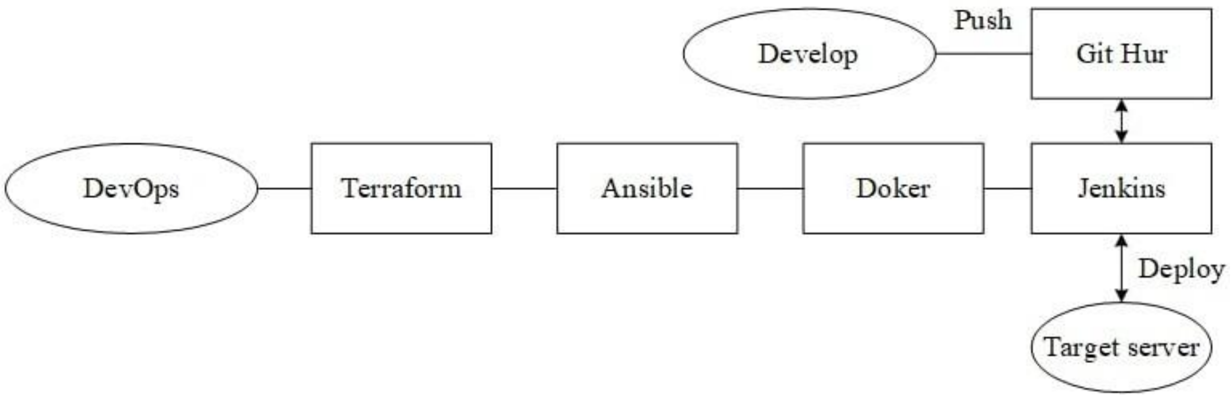
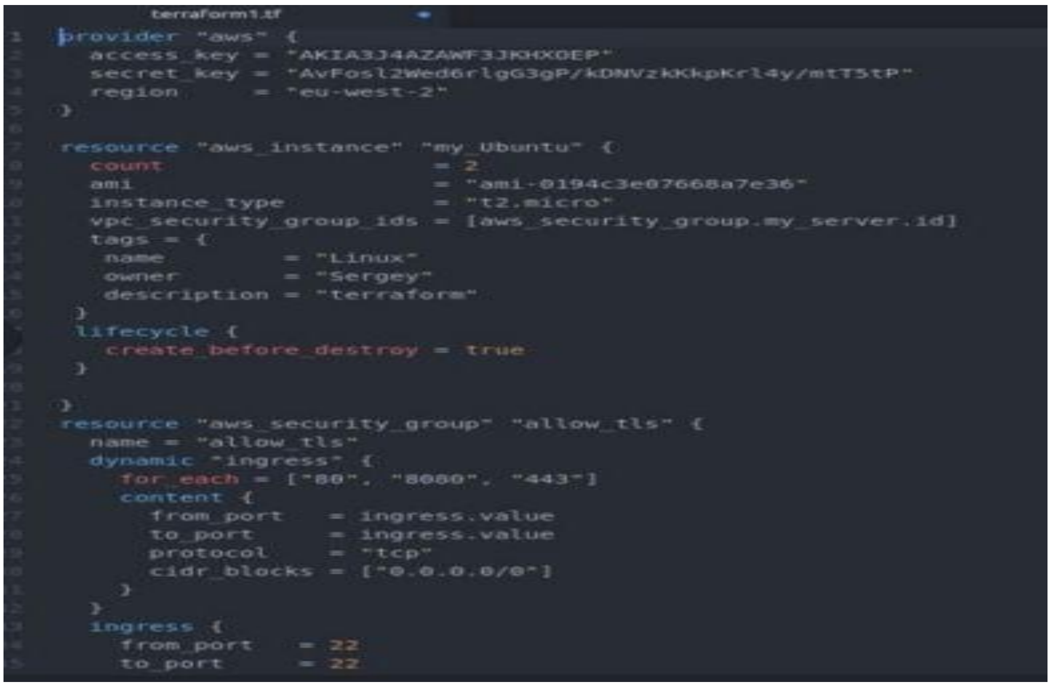


Step 1 - Launching Machines with Terraform

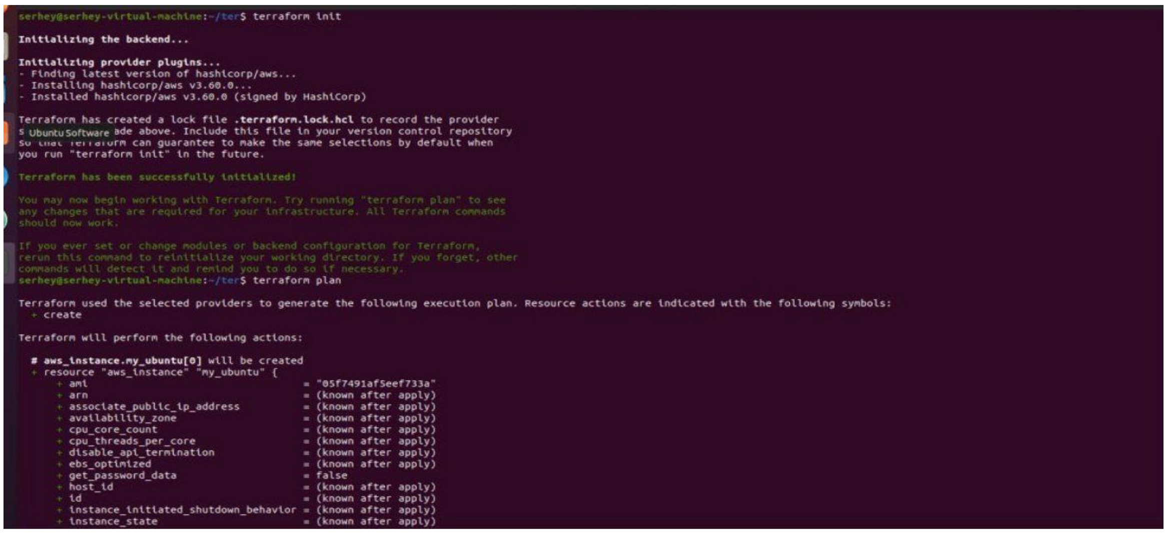
Terraform maps the resources described in the config file to the corresponding resources of the cloud provider. This mapping is called state, it's a giant JSON file. When terraform apply runs, Terraform updates the state by forwarding the appropriate request to the cloud provider. It then compares the 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".



```
terraform1.tf
provider "aws" {
  access_key = "AKIA3J4AZAWF3JKHXOEP"
  secret_key = "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"
  region     = "eu-west-2"
}

resource "aws_instance" "my_Ubuntu" {
  count                  = 2
  ami                    = "ami-0194c3e07668a7e36"
  instance_type          = "t2.micro"
  vpc_security_group_ids = [aws_security_group.my_server.id]
  tags = {
   name        = "Linux"
   owner       = "Sergey"
   description = "terraform"
 }
 lifecycle {
   create_before_destroy = true
 }

}
resource "aws_security_group" "allow_tls" {
  name = "allow_tls"
  dynamic "ingress" {
    for_each = ["80", "8080", "443"]
    content {
      from_port   = ingress.value
      to_port     = ingress.value
      protocol    = "tcp"
      cidr_blocks = ["0.0.0.0/0"]
    }
  }
  ingress {
    from_port   = 22
    to_port     = 22
```

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.



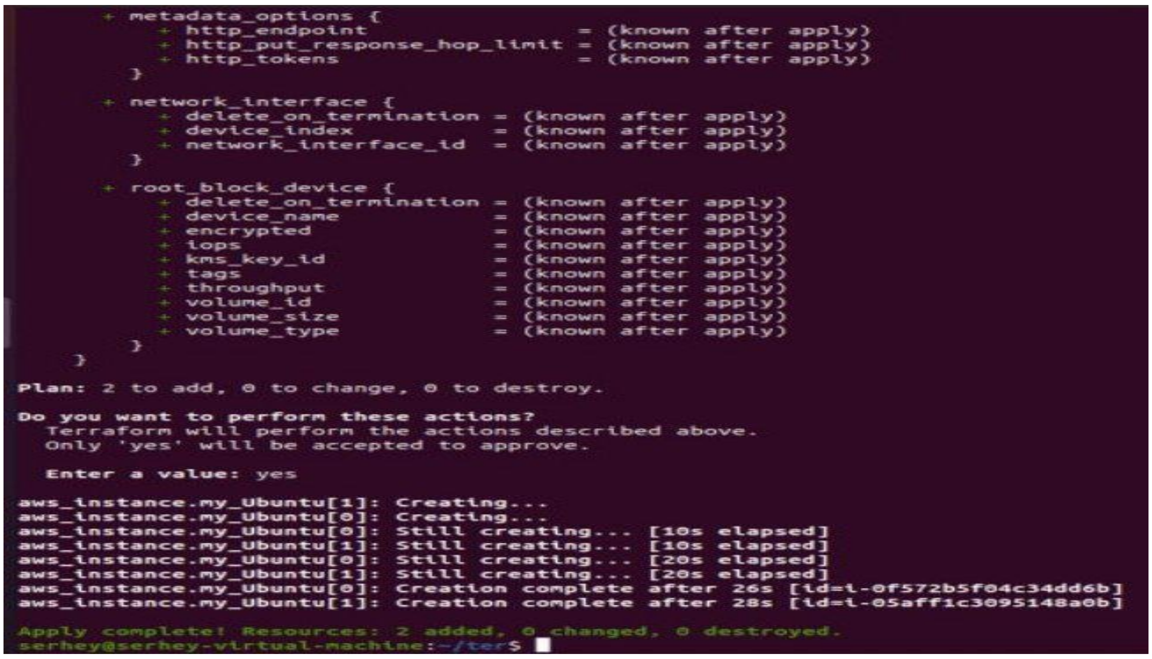
```
serhey@serhey-virtual-machine:~/ter$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.60.0...
- Installed hashicorp/aws v3.60.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
s[illegible]ade above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
serhey@serhey-virtual-machine:~/ter$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.my_ubuntu[0] will be created
  + resource "aws_instance" "my_ubuntu" {
      + ami                                  = "05f7491af5eef733a"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
```

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.



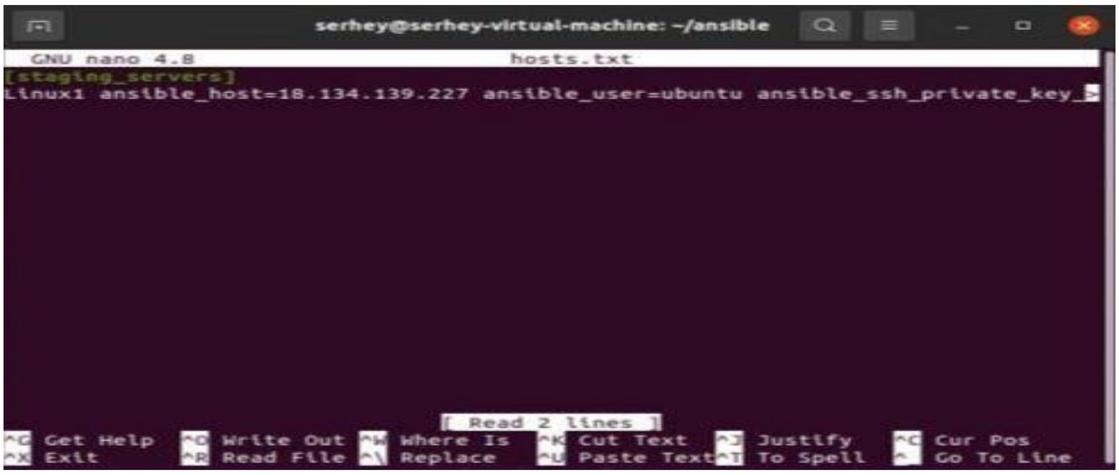
```
      + metadata_options {
          + http_endpoint               = (known after apply)
          + http_put_response_hop_limit = (known after apply)
          + http_tokens                 = (known after apply)
        }

      + network_interface {
          + delete_on_termination = (known after apply)
          + device_index          = (known after apply)
          + network_interface_id  = (known after apply)
        }

      + root_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + kms_key_id            = (known after apply)
          + tags                  = (known after apply)
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.my_Ubuntu[1]: Creating...
aws_instance.my_Ubuntu[0]: Creating...
aws_instance.my_Ubuntu[0]: Still creating... [10s elapsed]
aws_instance.my_Ubuntu[1]: Still creating... [10s elapsed]
aws_instance.my_Ubuntu[0]: Still creating... [20s elapsed]
aws_instance.my_Ubuntu[1]: Still creating... [20s elapsed]
aws_instance.my_Ubuntu[0]: Creation complete after 26s [id=i-0f572b5f04c34dd6b]
aws_instance.my_Ubuntu[1]: Creation complete after 28s [id=i-05aff1c3095148a0b]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
serhey@serhey-virtual-machine:~/ter$
```

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

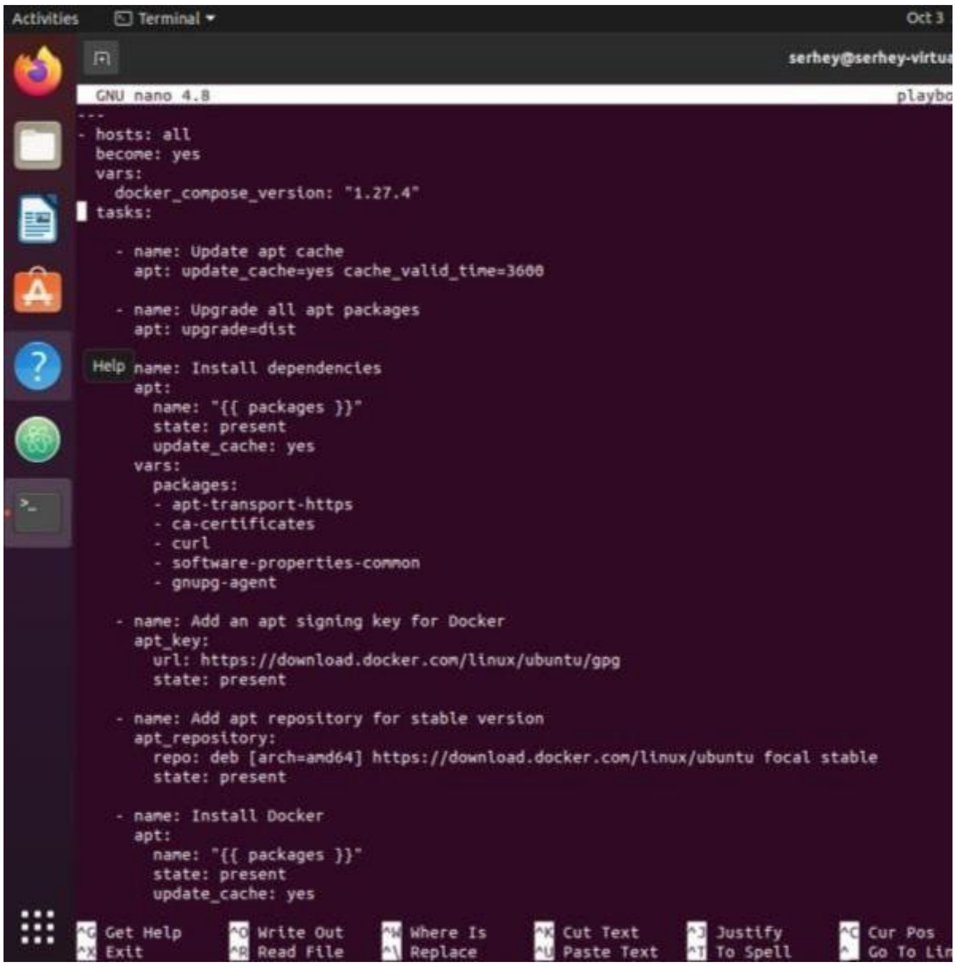
Ansible is a simple configuration management tool that automates I resource management, application deployment or cloud provisioning, and more.

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



```
serhey@serhey-virtual-machine: ~/ansible
GNU nano 4.8                         hosts.txt
[staging_servers]
Linux1 ansible_host=18.134.139.227 ansible_user=ubuntu ansible_ssh_private_key_>

                              [ Read 2 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

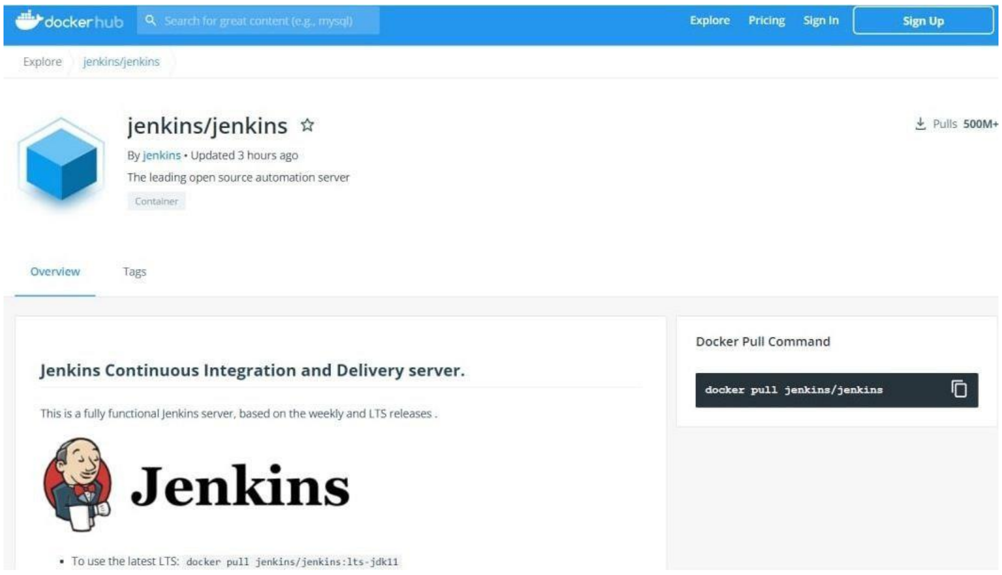
Ansible playbooks are a way to send commands to remote computers using scripts. Instead of individually using commands to remotely configure computers from the command line, we set up the entire installation process by passing a script to one or more systems.



```
GNU nano 4.8                                                    playbo
---
- hosts: all
  become: yes
  vars:
    docker_compose_version: "1.27.4"
  tasks:

    - name: Update apt cache
      apt: update_cache=yes cache_valid_time=3600

    - name: Upgrade all apt packages
      apt: upgrade=dist

    - name: Install dependencies
      apt:
        name: "{{ packages }}"
        state: present
        update_cache: yes
      vars:
        packages:
        - apt-transport-https
        - ca-certificates
        - curl
        - software-properties-common
        - gnupg-agent

    - name: Add an apt signing key for Docker
      apt_key:
        url: https://download.docker.com/linux/ubuntu/gpg
        state: present

    - name: Add apt repository for stable version
      apt_repository:
        repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable
        state: present

    - name: Install Docker
      apt:
        name: "{{ packages }}"
        state: present
        update_cache: yes
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Lin
```

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.



jenkins/jenkins ☆

By jenkins • Updated 3 hours ago

The leading open source automation server

Jenkins Continuous Integration and Delivery server.

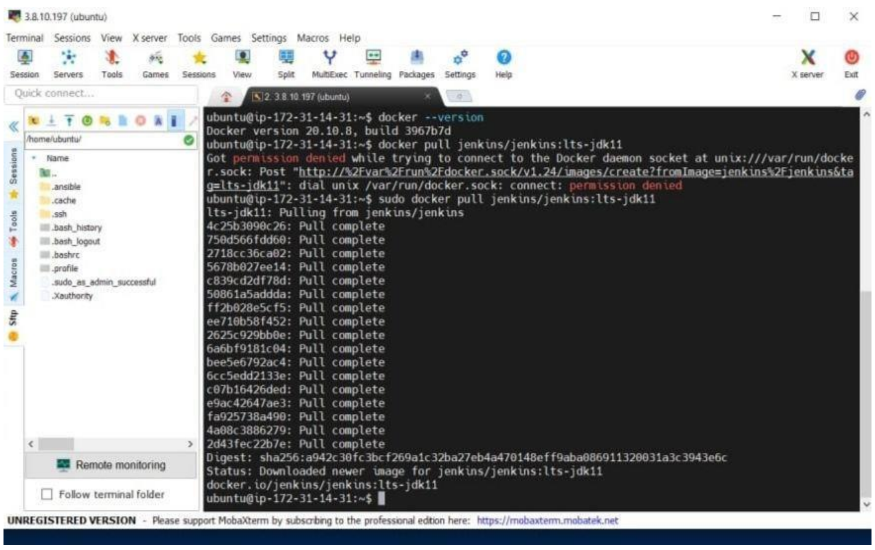
This is a fully functional Jenkins server, based on the weekly and LTS releases .

- To use the latest LTS: `docker pull jenkins/jenkins:lts-jdk11`

Docker Pull Command

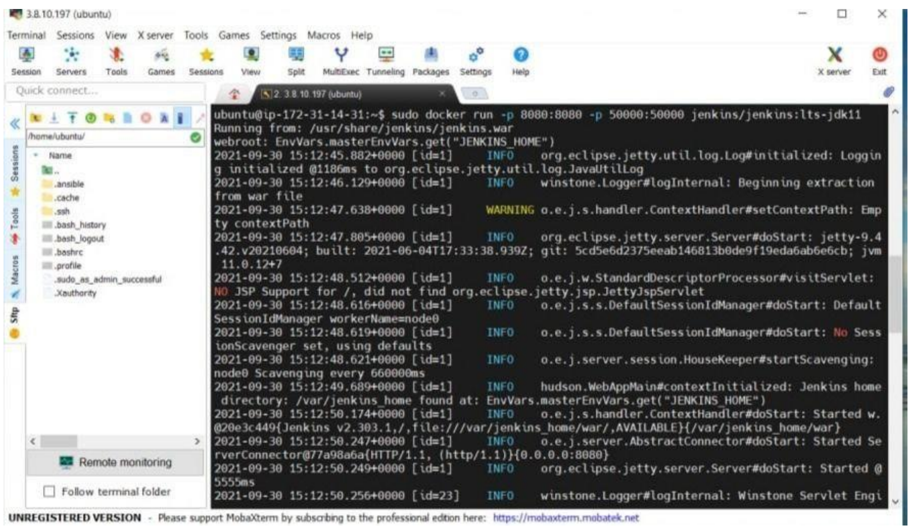
```
docker pull jenkins/jenkins
```

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.



```
ubuntu@ip-172-31-14-31:~$ docker --version
Docker version 20.10.8, build 3967b7d
ubuntu@ip-172-31-14-31:~$ docker pull jenkins/jenkins:lts-jdk11
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docke
r.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images/create?fromImage=jenkins%2Fjenkins&ta
g=lts-jdk11": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-14-31:~$ sudo docker pull jenkins/jenkins:lts-jdk11
lts-jdk11: Pulling from jenkins/jenkins
4c25b3090c26: Pull complete
750d566fdd60: Pull complete
2718cc36ca02: Pull complete
5678b027ee14: Pull complete
c839cd2df78d: Pull complete
50861a5addda: Pull complete
ff2b028e5cf5: Pull complete
ee710b58f452: Pull complete
2625c929bb0e: Pull complete
6a6bf9181c04: Pull complete
bee5e6792ac4: Pull complete
6cc5edd2133e: Pull complete
c07b16426ded: Pull complete
e9ac42647ae3: Pull complete
fa925738a490: Pull complete
4a08c3886279: Pull complete
2d43fec22b7e: Pull complete
Digest: sha256:a942c30fc3bcf269a1c32ba27eb4a470148eff9aba086911320031a3c3943e6c
Status: Downloaded newer image for jenkins/jenkins:lts-jdk11
docker.io/jenkins/jenkins:lts-jdk11
ubuntu@ip-172-31-14-31:~$
```

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.



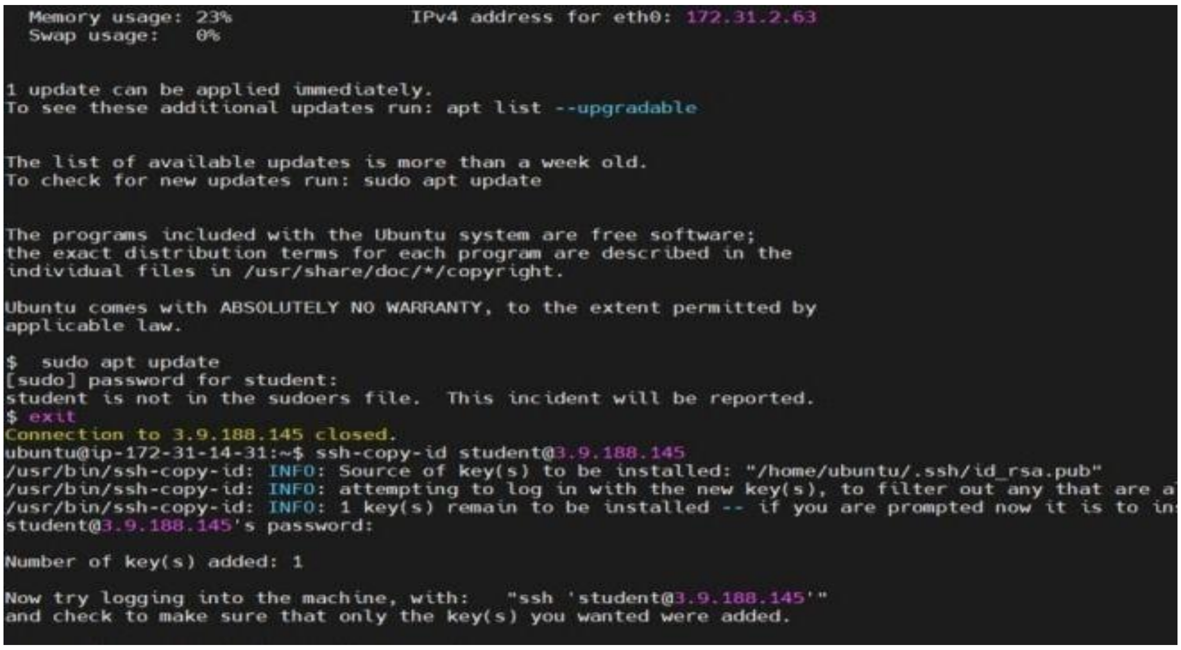
```
ubuntu@ip-172-31-14-31:~$ sudo docker run -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts-jdk11
Running from: /usr/share/jenkins/jenkins.war
webroot: EnvVars.masterEnvVars.get("JENKINS_HOME")
2021-09-30 15:12:45.882+0000 [id=1]     INFO    org.eclipse.jetty.util.log.Log#initialized: Loggin
g initialized @1186ms to org.eclipse.jetty.util.log.JavaUtilLog
2021-09-30 15:12:46.129+0000 [id=1]     INFO    winstone.Logger#logInternal: Beginning extraction
from war file
2021-09-30 15:12:47.638+0000 [id=1]     WARNING o.e.j.s.handler.ContextHandler#setContextPath: Emp
ty contextPath
2021-09-30 15:12:47.805+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: jetty-9.4
.42.v20210604; built: 2021-06-04T17:33:38.939Z; git: 5cd5e6d2375eeab146813b0de9f19eda6ab6e6cb; jvm
 11.0.12+7
2021-09-30 15:12:48.512+0000 [id=1]     INFO    o.e.j.w.StandardDescriptorProcessor#visitServlet:
NO JSP Support for /, did not find org.eclipse.jetty.jsp.JettyJspServlet
2021-09-30 15:12:48.616+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: Default
SessionIdManager workerName=node0
2021-09-30 15:12:48.619+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: No Sess
ionScavenger set, using defaults
2021-09-30 15:12:48.621+0000 [id=1]     INFO    o.e.j.server.session.HouseKeeper#startScavenging:
node0 Scavenging every 660000ms
2021-09-30 15:12:49.689+0000 [id=1]     INFO    hudson.WebAppMain#contextInitialized: Jenkins home
 directory: /var/jenkins_home found at: EnvVars.masterEnvVars.get("JENKINS_HOME")
2021-09-30 15:12:50.174+0000 [id=1]     INFO    o.e.j.s.handler.ContextHandler#doStart: Started w.
@20e3c449{Jenkins v2.303.1,/,file:///var/jenkins_home/war/,AVAILABLE}{/var/jenkins_home/war}
2021-09-30 15:12:50.247+0000 [id=1]     INFO    o.e.j.server.AbstractConnector#doStart: Started Se
rverConnector@77a98a6a{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}
2021-09-30 15:12:50.249+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: Started @
5555ms
2021-09-30 15:12:50.256+0000 [id=23]    INFO    winstone.Logger#logInternal: Winstone Servlet Engi
```

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

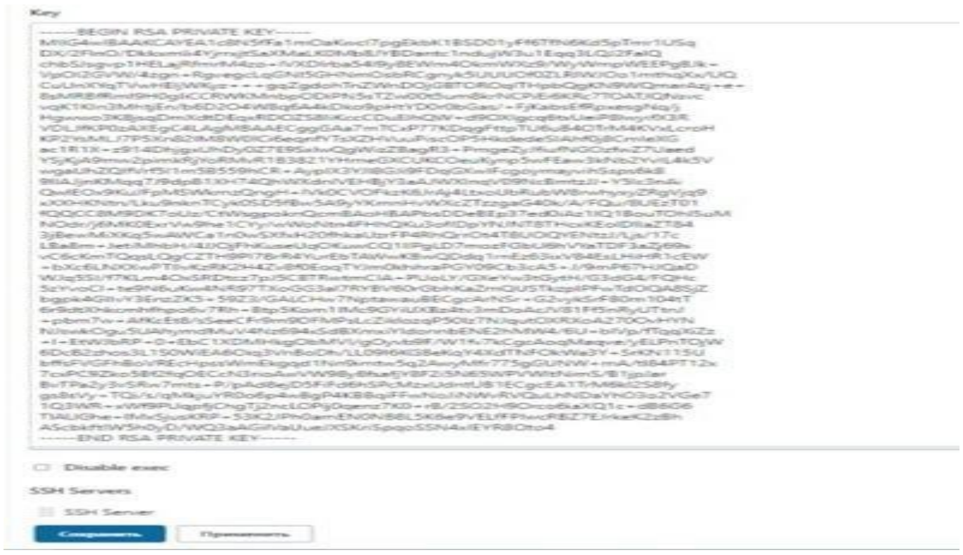
We connect via ssh to the server to which the Deploy will be performed



```
  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

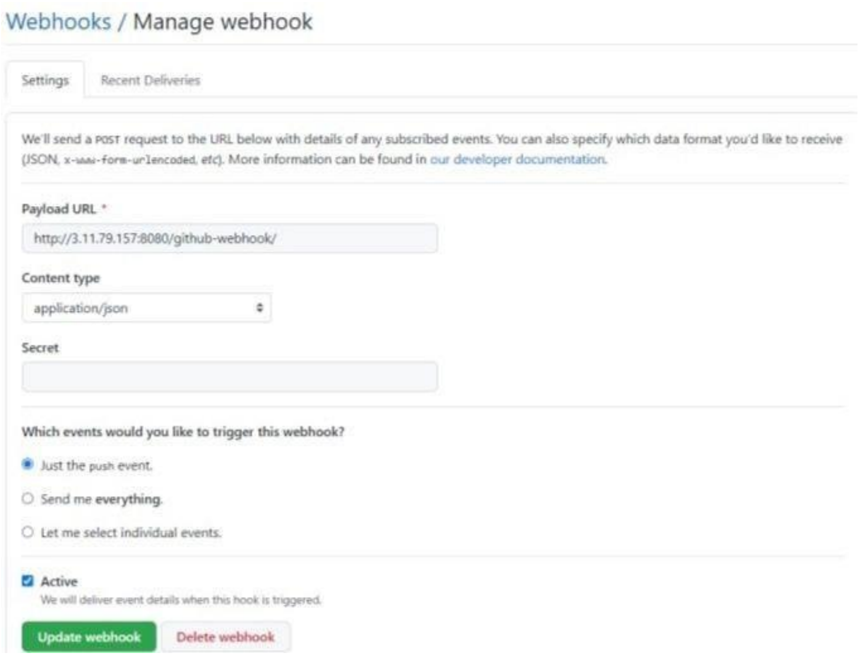
Copy the private key and check the connection



Key

```
-----BEGIN RSA PRIVATE KEY-----
[illegible]
-----END RSA PRIVATE KEY-----
```

Configuring GitHub to connect to Jenkins


Webhooks / Manage webhook

Settings | Recent Deliveries

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://3.11.79.157:8080/github-webhook/

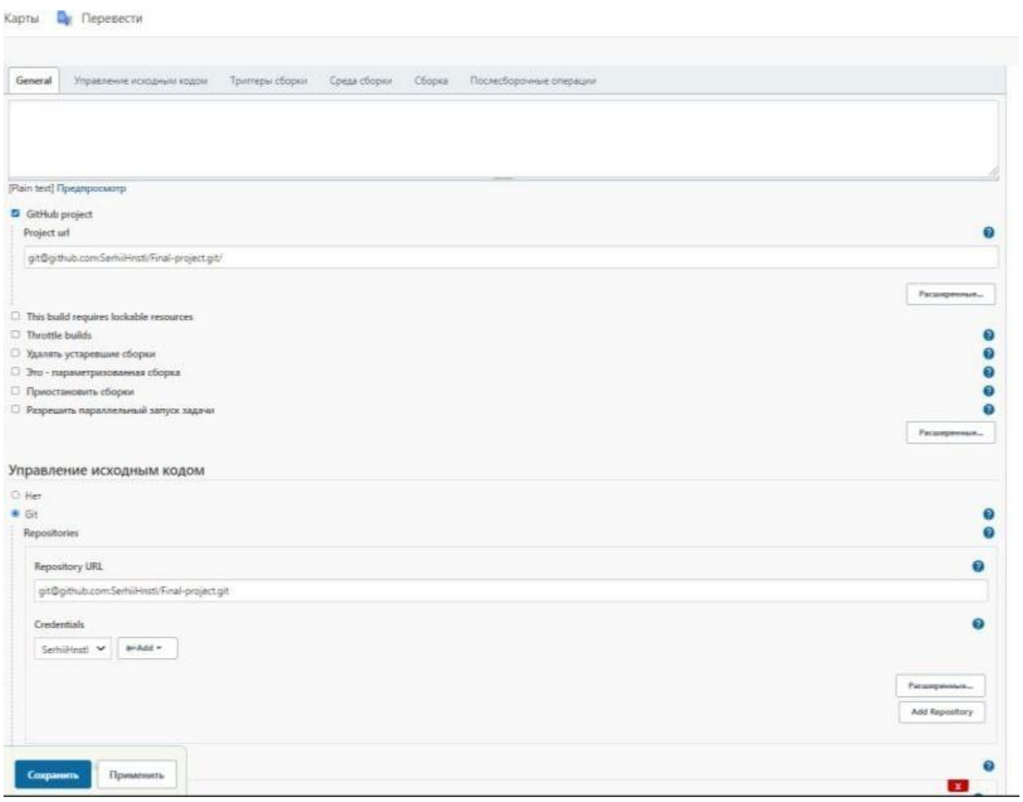
Content type
application/json

Secret

Which events would you like to trigger this webhook?
- [✓ Just the push event.]
- Send me **everything**.
- Let me select individual events.

[✓] **Active**
We will deliver event details when this hook is triggered.

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



[Plain text] Предпросмотр

[✓] GitHub project
Project url
git@github.com:SerhiiHnsti/Final-project.git/

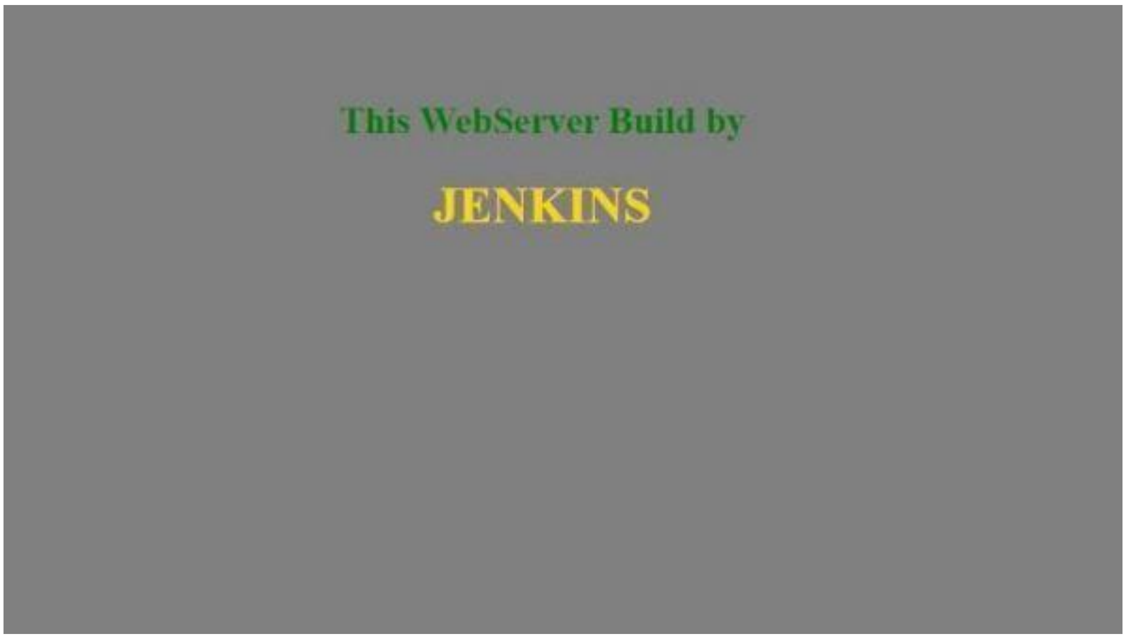
- [ ] This build requires lockable resources
- [ ] Throttle builds
- [ ] Удалять устаревшие сборки
- [ ] Это - параметризованная сборка
- [ ] Приостановить сборки
- [ ] Разрешить параллельный запуск задачи

Управление исходным кодом
- Нет
- [✓ Git]

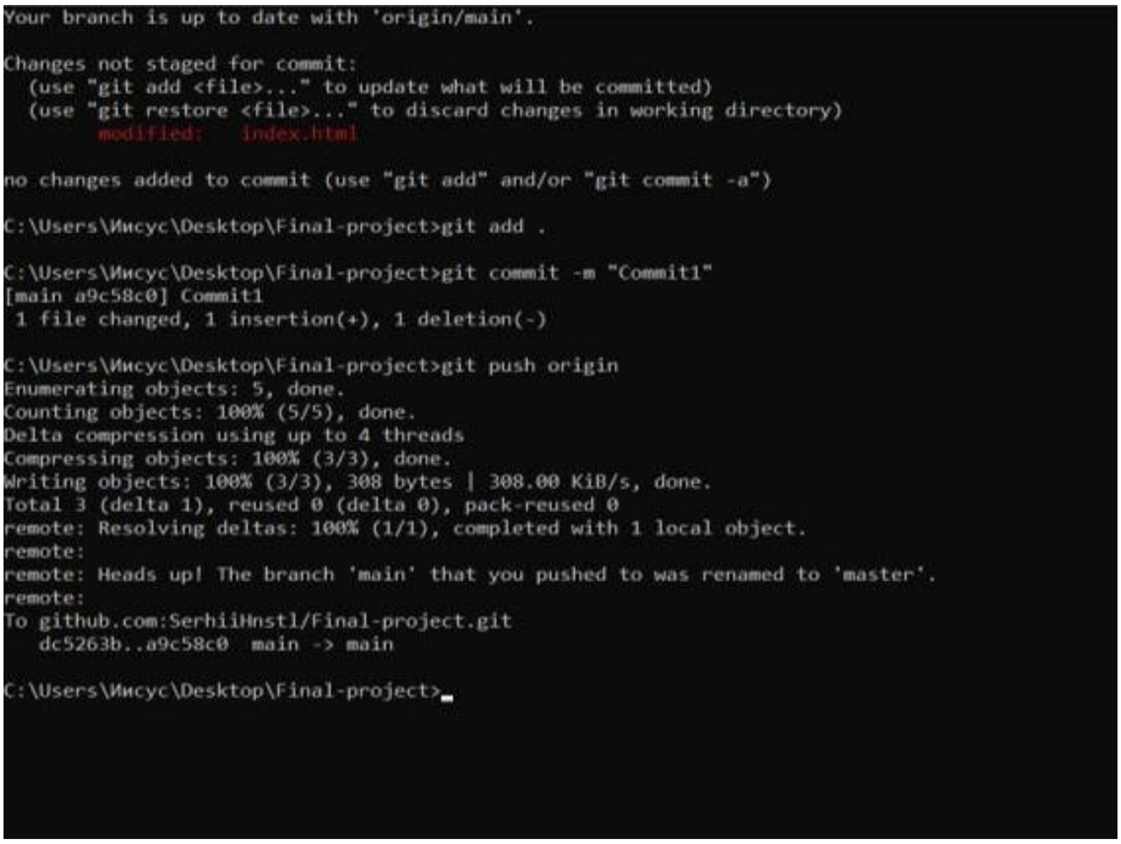
Repositories
Repository URL
git@github.com:SerhiiHnsti/Final-project.git

Credentials
SerhiiHnsti

Initial view of our site:

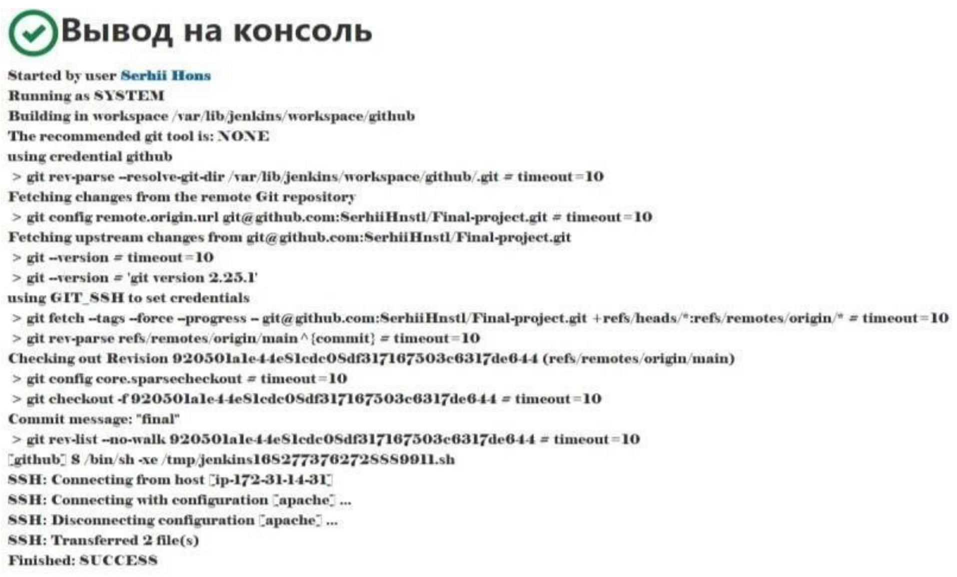


Making changes and pushing:



```
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>
```

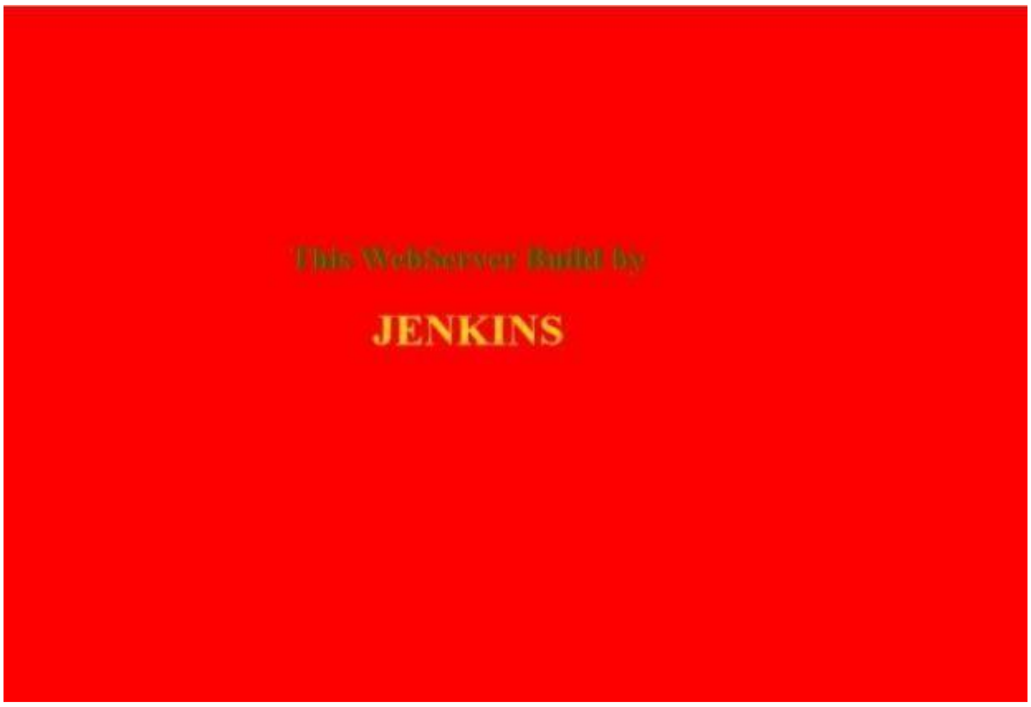
Jenkins starts automatically:



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/jenkins1682773762728889911.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!