EPAM Winter 2020-2021 Final Task

CREATING BASIC CI/CD CHAIN

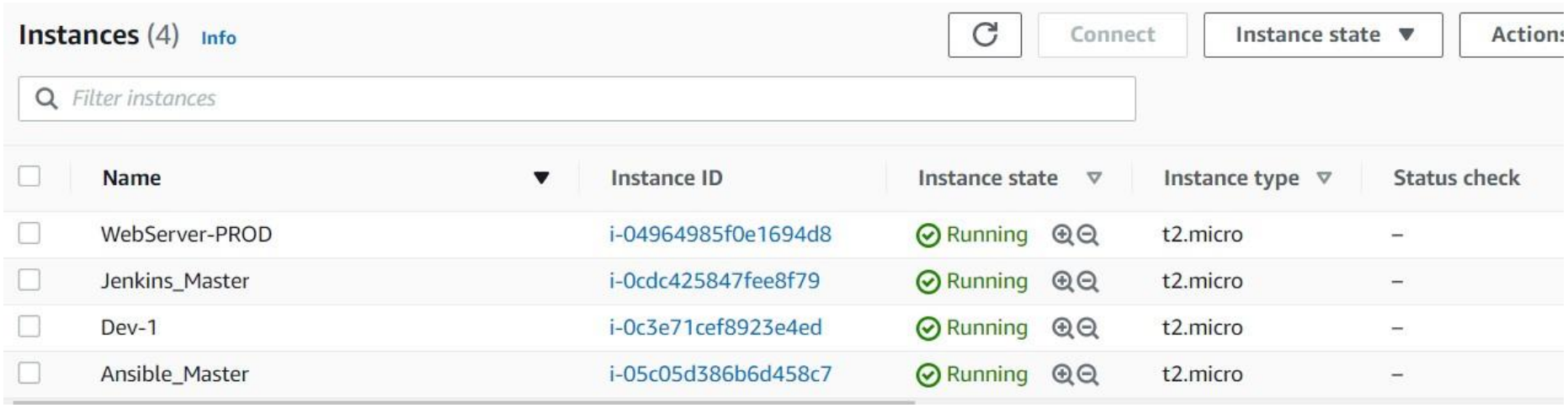
The main idea is

- The main idea of this project is to create a simple CI/CD chain. At the end of this project, we will have 1 Developer, 1 Web Server (PROD), 1 Jenkins-Master. I also decided to use Ansible and write a simple playbook in which I described the configuration of our Web Servera. So I'm going to add an instance for Ansible-Master as well.

Project Steps

1. Let's create four instances for our purpose.
2. Set up Git on Dev-workstation and connected it with GitHub repository
3. Install Ansible and write playbook to configure our Web Server.
4. Install Java and Jenkins. Connecting Jenkins and GitHub via Webhook.
5. Configure Build actions in Jenkins.

Step 1. Let's create four instances for our purpose.



Step 2. Set up Git on Dev-workstation and connected it with GitHub repository

- We going to use ec2 instances along this project. After we created instance for Developer we have to set up Git. Before that I created the repository to connect. For connecting to this repository I configured access via ssh.
- Let's check if it's works with push.

```
drwxrwxr-x 3 ubuntu ubuntu 4096 Mar 26 01:20 ./
drwxr-xr-x 7 ubuntu ubuntu 4096 Mar 26 01:14 ../
drwxrwxr-x 8 ubuntu ubuntu 4096 Mar 26 01:18 .git/
-rw-rw-r-- 1 ubuntu ubuntu    1 Mar 26 01:20 index.html
ubuntu@ip-172-31-32-158:~/WebServer$ git add .
ubuntu@ip-172-31-32-158:~/WebServer$ git commit -m "Add empty
[master 7e8f433] Add empty index.html
 1 file changed, 1 insertion(+)
 create mode 100644 index.html
ubuntu@ip-172-31-32-158:~/WebServer$ git push
Counting objects: 3, done.
Writing objects: 100% (3/3), 271 bytes | 271.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:eagle-vn/WebServer.git
   f7e92ed..7e8f433  master -> master
```

Step 3. Install Ansible and write playbook to configure our Web Server.

- Using build-in modules we configure the state of our Web Server.
- What this playbook is done? Install apache2, then configure it to be enabled with boot and change the owner of index.html so that we will be able to change it using Jenkins.

```
ubuntu@ip-172-31-45-191:~/ansible$ cat web.yml
---

- name: Install apache and change owner index.html file
  hosts: PROD_SERVERS
  become: yes

  tasks:
  - name: Install Apache WebServer
    apt: name=apache2 state=present

  - name: Start Apache and Enable it with boot
    service: name=apache2 state=started enabled=yes

  - name: Change owner index.html file
    file: path=/var/www/html/index.html owner=ubuntu
```

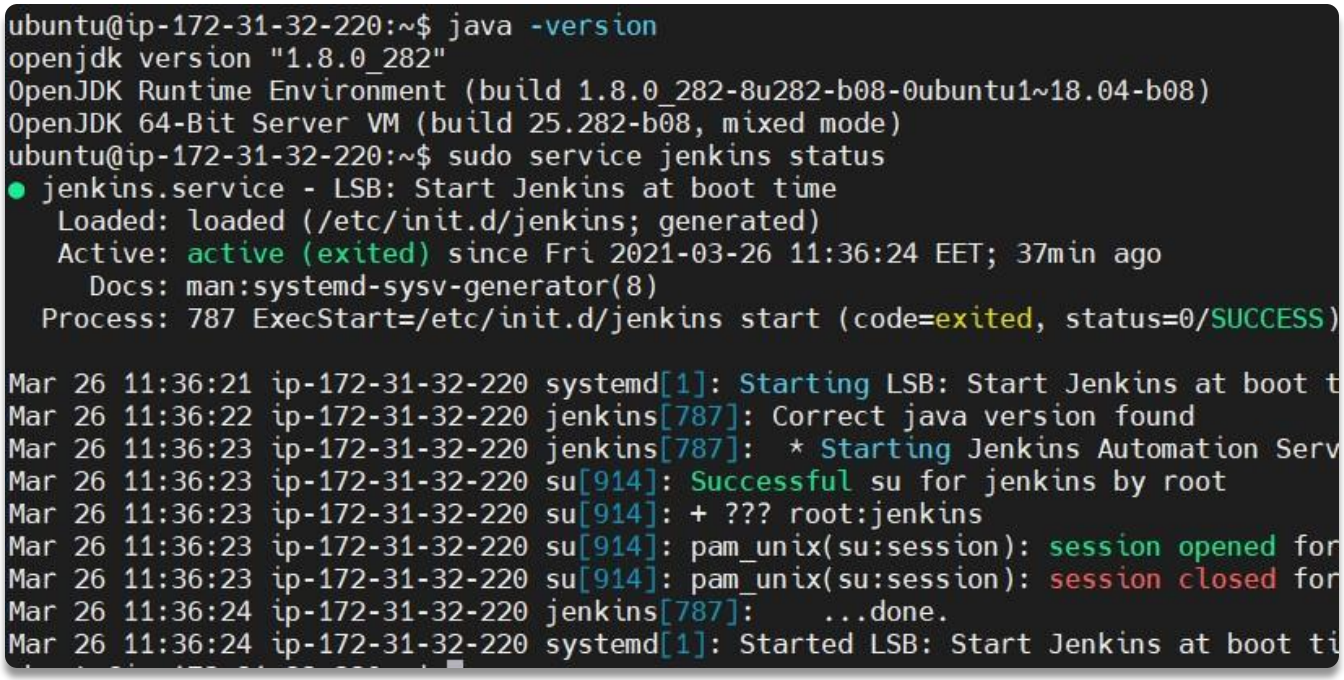
Check how it works.

- I have run this playbook before so nothing has changed. Everything is already in the right state. And that's cool.

```
ubuntu@ip-172-31-45-191:~/ansible$ ansible-playbook web.yml

PLAY [Install apache and change owner index.html file] ******************

TASK [Gathering Facts] ***************************************************
ok: [linux-1]

TASK [Install Apache WebServer] ******************************************
ok: [linux-1]

TASK [Start Apache and Enable it with boot] ******************************
ok: [linux-1]

TASK [Change owner index.html file] **************************************
ok: [linux-1]

PLAY RECAP ***************************************************************
linux-1                    : ok=4    changed=0    unreachable=0    failed
```

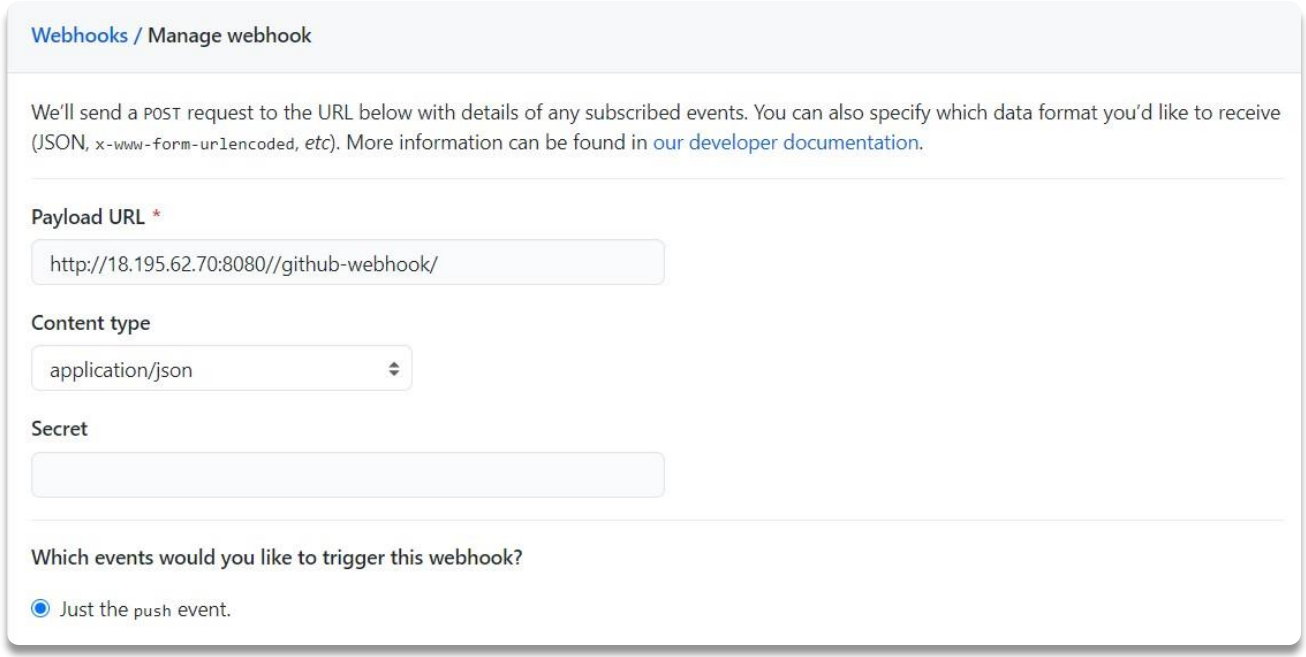
Step 4.Install Java and Jenkins. Connecting Jenkins and GitHub via Webhook.

- I have already done these steps earlier, so we only need to check if everything is up and running. And it does!



Configuring a trigger via a webhook

- When our developer pushes something to the repository it will send a POST request to our Jenkins-Master. After that, Jenkins will be triggered to look if something new in this repository appears?



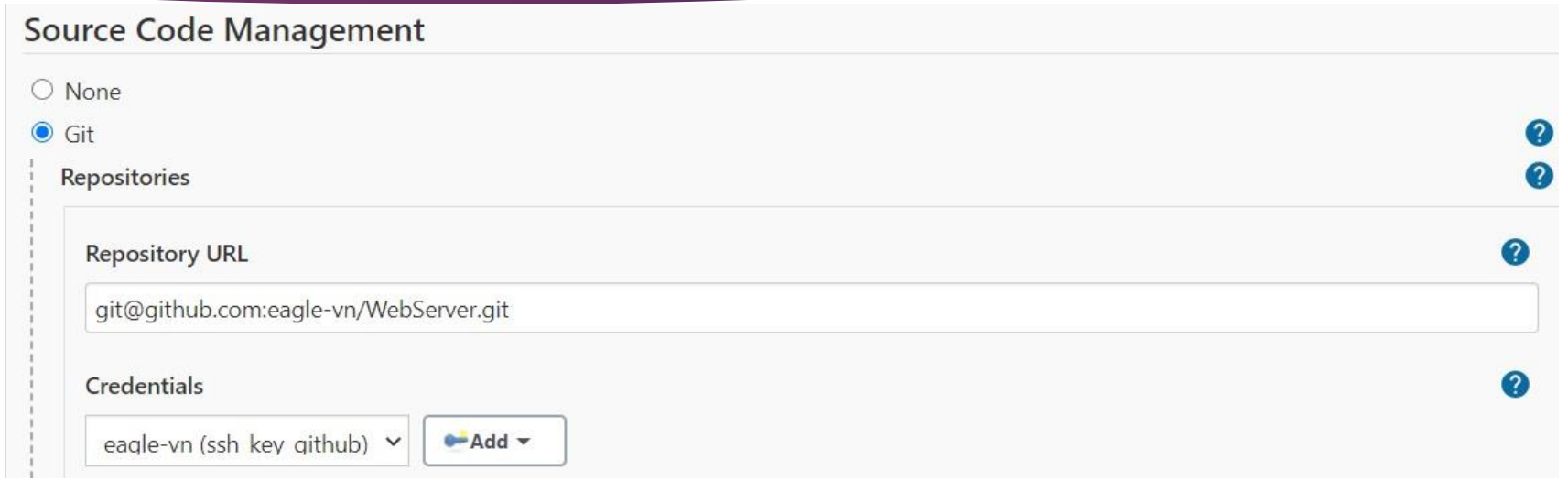
Step 5. Configure Build actions in Jenkins

- We gonna use the following plugins: GitHub, Git, Publish Over SSH, SSH Agend. Let's make it done :)
- So, here we put our GitHub repository to make the link between the following repo and Jenkins. We already placed this link in Step 4.

Pre-configuration for working with the remote repository.

Specify the repository to track. And also add the credentials for our repository.

Check the box to enable Triggers.



Add simple build actions, test and delivery to our Web Server.

Build

Execute shell

Command

```
echo "Build started"
ls -lah
echo "The name of this job is: $JOB_NAME and Build №$BUILD_ID" >> index.html
echo "Build finished"
```

Execute shell

Command

```
echo "Test started"
test=`grep "Hello" index.html | wc -l`
echo "$test"
if [ "$test" = "1" ]
then
        echo "Test passed"
else
        echo "Test failed"
        exit 1
fi
```

Post-build Actions

Post-build Actions

Send build artifacts over SSH

SSH Publishers

SSH Server

Name

WebServer-PROD

Let's check the result.

- I changed the background and pushed it to our repository. Then build started via Webhook, tested, and delivered to the Web Server.

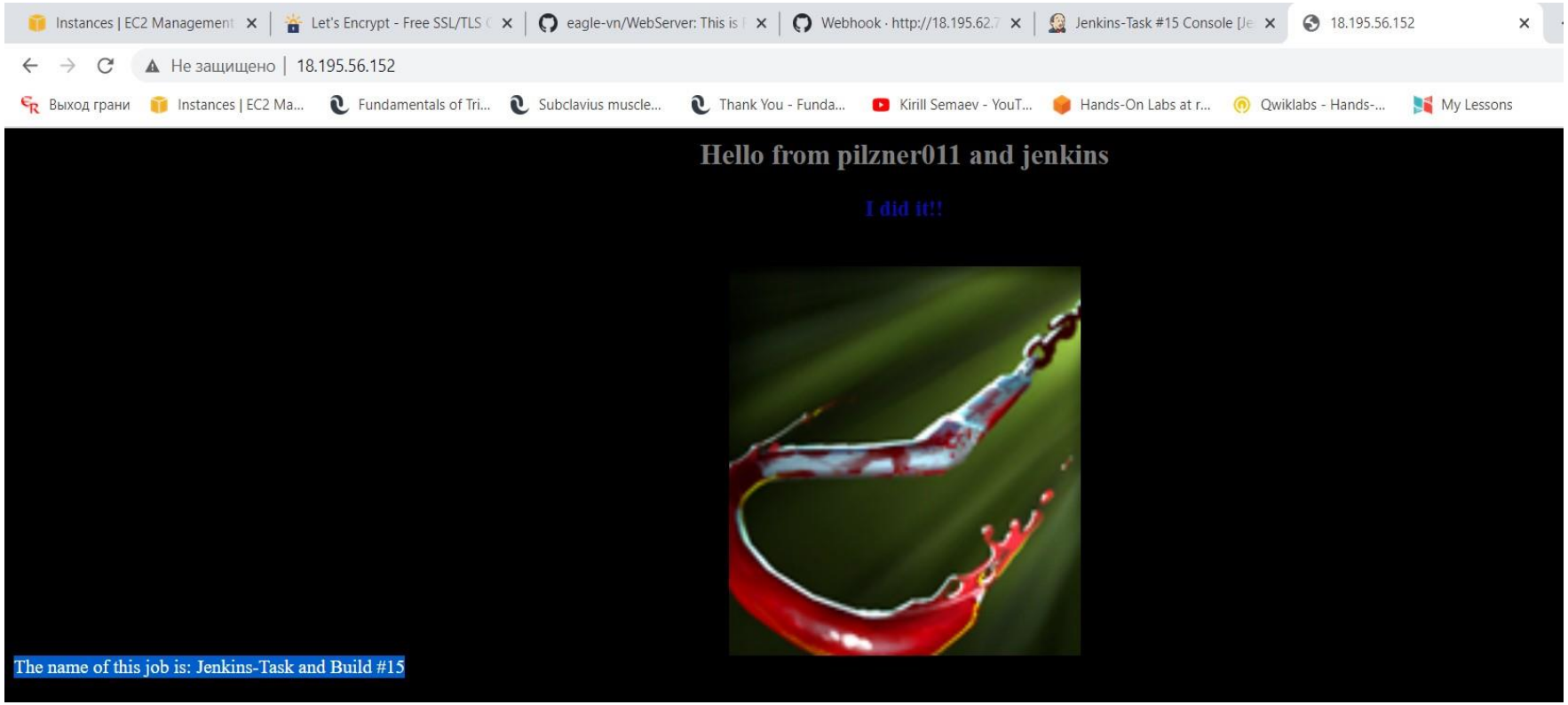
Recent Deliveries

b9dde732-8e28-11eb-94f3-d6b43a9bc1c0 2021-03-26 14:45:13 ...

Console Output

```
Started by GitHub push by eagle-vn
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Jenkins-Task
```

Thank you for attention!



Project by Volodymyr Sheshulkov