Deploy a Flask Application with ML Model on AWS EC2 Using CI/CD Pipeline

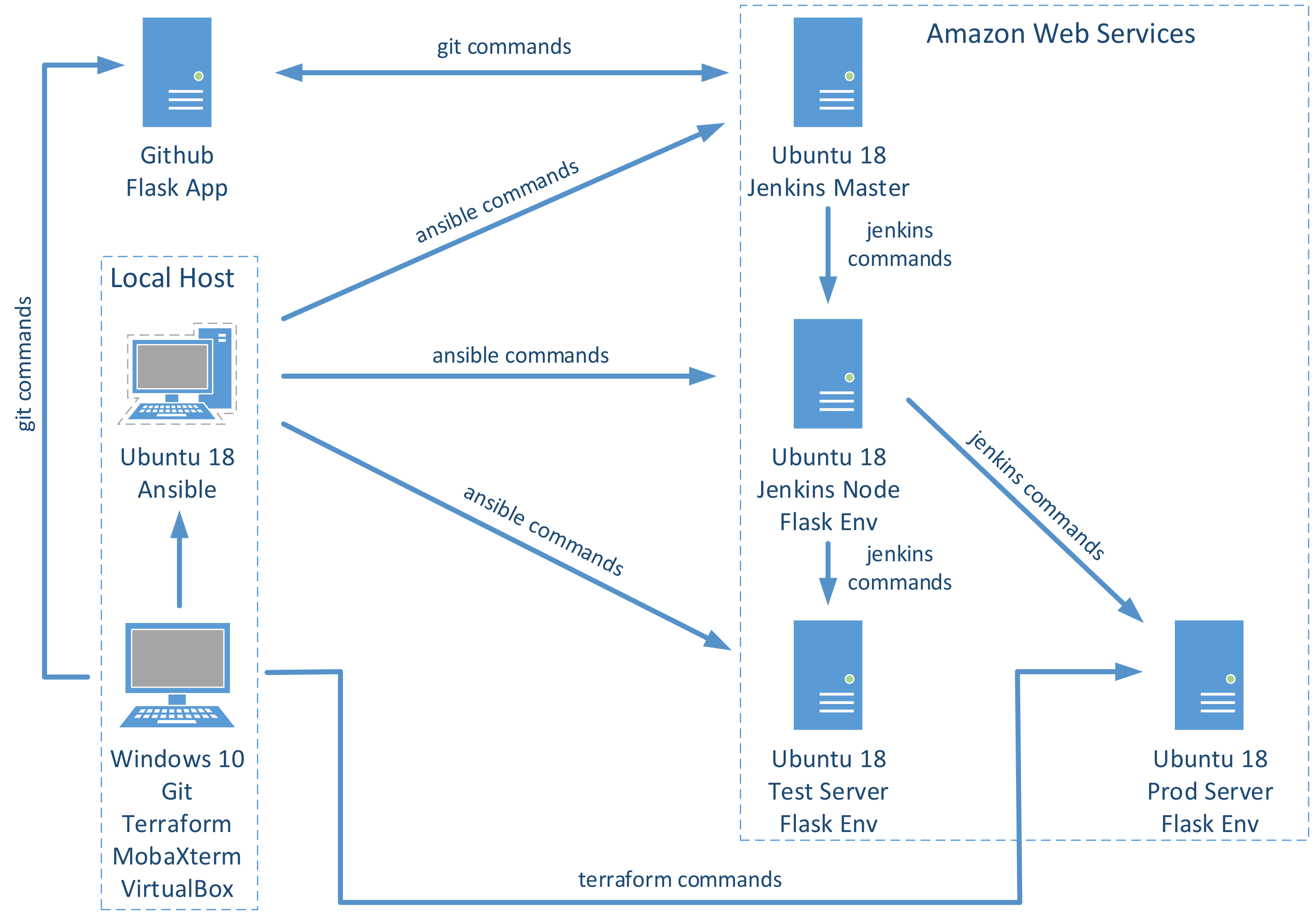
Mykola Shtompel

Motivation, relevance, goal, tasks

- Motivation – interest in machine learning technologies and the specifics of their implementation.
- Relevance – need to provide web access to software that uses machine learning models.
- Goal – ensure the deployment of the flask application with machine learning model on the cloud servers in a reasonable time.
- Tasks:
  - Development of the project structure and selection of software and technologies
  - Implementation of the application deployment procedure in the cloud infrastructure

Design - Software and technologies

- Git
- Ansible
- Terraform
- Jenkins
- Github
- AWS



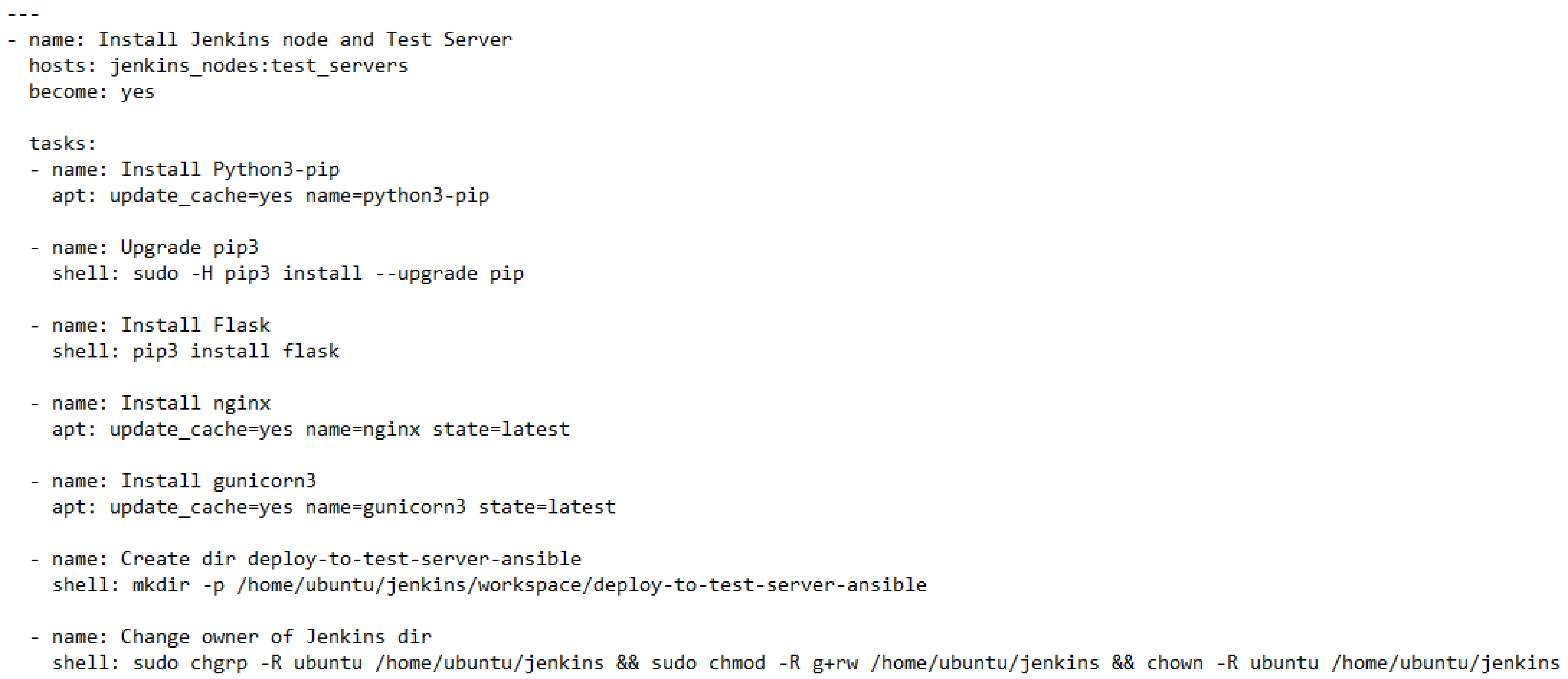
Implementation – Main Steps. Part 1

- Creating Github repository and installing Git on Local Host
- Adding Flask App on Github repository using Git from Local Host
- Creating and configuring Jenkins, Jenkins Node and Test Server instances on Amazon Web Services from Local Host (manually)
- Installing Ansible on VirtualBox Machine from Local Host (manually)
- Creating basic network settings and testing connection to servers using Ansible (settings files, playbook) from VirtualBox Machine
- Configuring Jenkins Master using Ansible (playbook) from VirtualBox Machine with additional manually configuring from Local Host

Implementation – Main Steps. Part 2

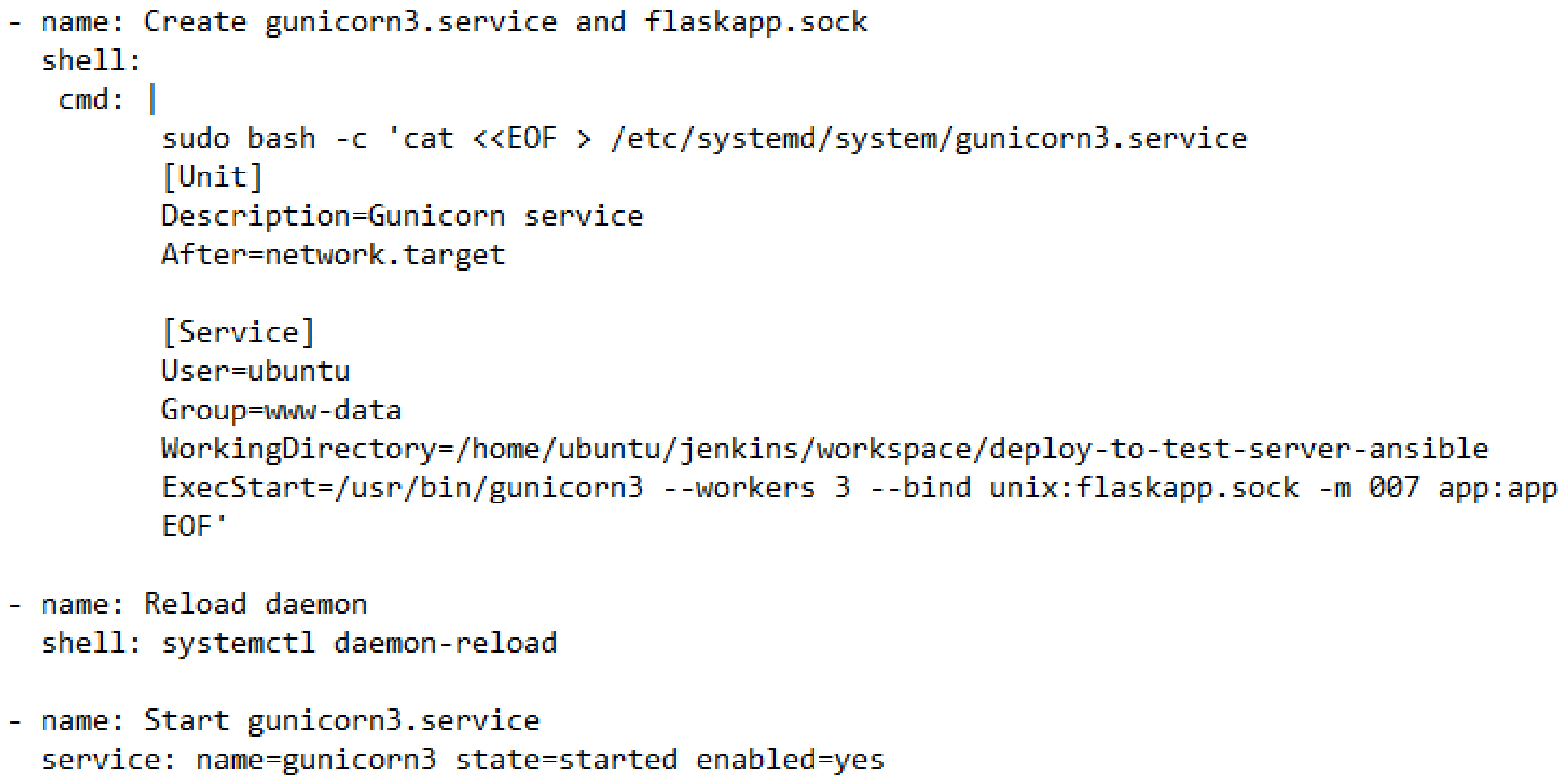
- Configuring Jenkins Node and Test Server using Ansible (playbooks) from VirtualBox Machine
- Installing Terraform and Atom on Local Host
- Creating and configuring Prod Server instance on Amazon Web Services using Terraform (tf, user data) from Local Host
- Creating CI/CD pipeline using Jenkins to deploy Flask App on Test Server and Prod Server instances (jobs) on Amazon Web Services
- Changing Flask App on Github repository using Git from Local Host and checking results

Implementation – Screenshots. Ansible. Part 1



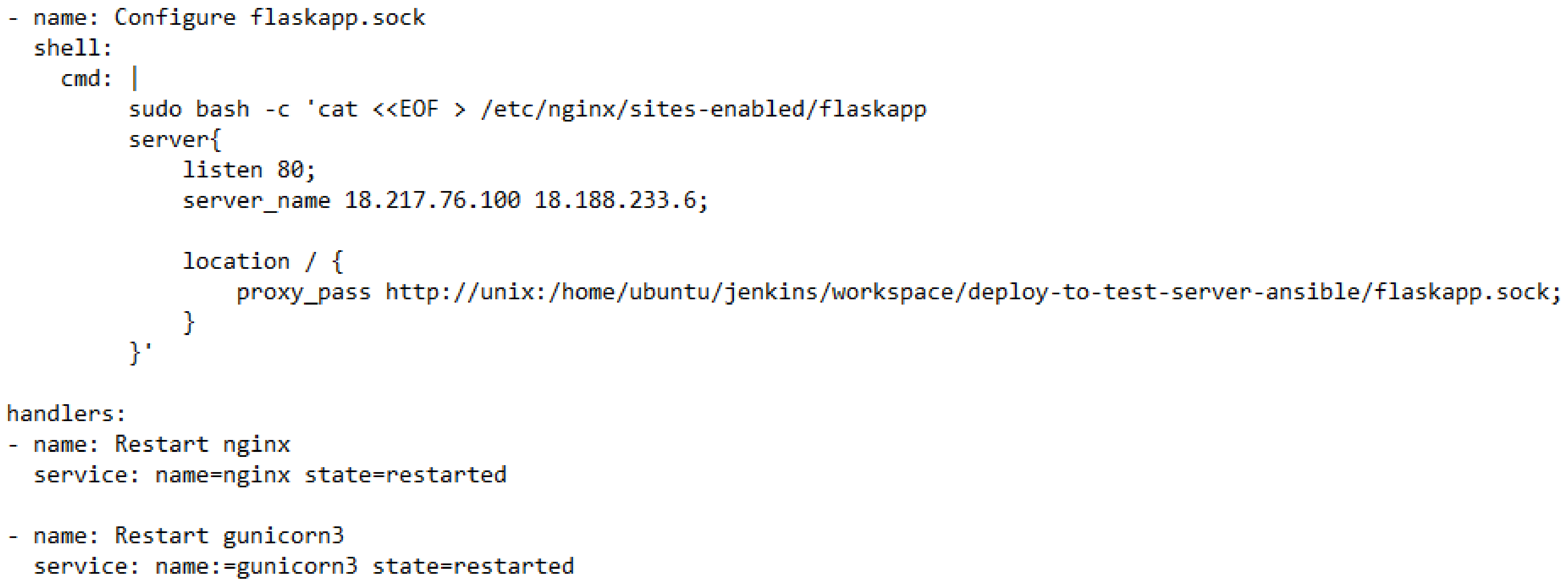
```
---
- name: Install Jenkins node and Test Server
  hosts: jenkins_nodes:test_servers
  become: yes

  tasks:
  - name: Install Python3-pip
    apt: update_cache=yes name=python3-pip

  - name: Upgrade pip3
    shell: sudo -H pip3 install --upgrade pip

  - name: Install Flask
    shell: pip3 install flask

  - name: Install nginx
    apt: update_cache=yes name=nginx state=latest

  - name: Install gunicorn3
    apt: update_cache=yes name=gunicorn3 state=latest

  - name: Create dir deploy-to-test-server-ansible
    shell: mkdir -p /home/ubuntu/jenkins/workspace/deploy-to-test-server-ansible

  - name: Change owner of Jenkins dir
    shell: sudo chgrp -R ubuntu /home/ubuntu/jenkins && sudo chmod -R g+rw /home/ubuntu/jenkins && chown -R ubuntu /home/ubuntu/jenkins
```

Implementation – Screenshots. Ansible. Part 2



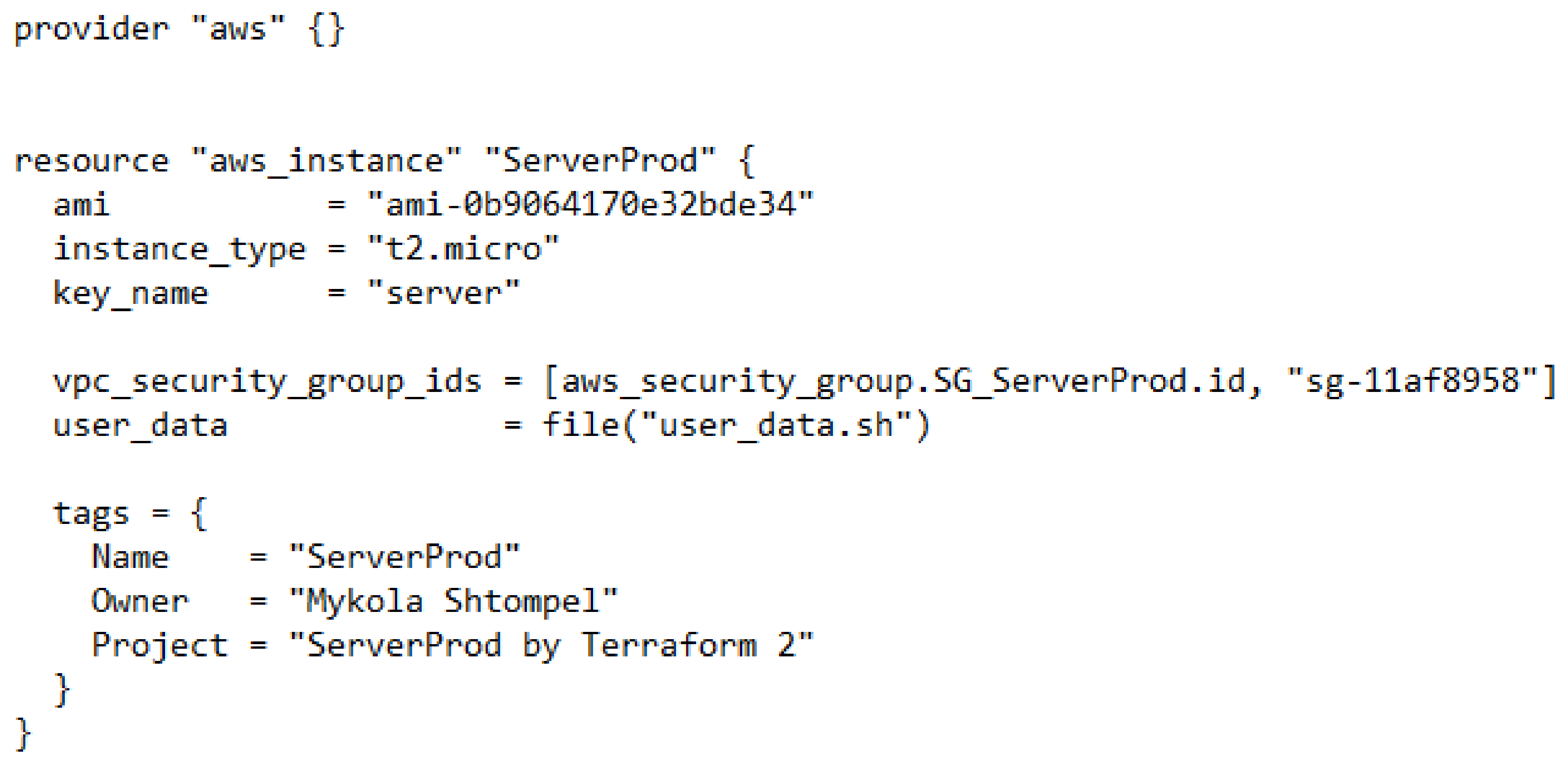
```
- name: Create gunicorn3.service and flaskapp.sock
  shell:
   cmd: |
         sudo bash -c 'cat <<EOF > /etc/systemd/system/gunicorn3.service
         [Unit]
         Description=Gunicorn service
         After=network.target

         [Service]
         User=ubuntu
         Group=www-data
         WorkingDirectory=/home/ubuntu/jenkins/workspace/deploy-to-test-server-ansible
         ExecStart=/usr/bin/gunicorn3 --workers 3 --bind unix:flaskapp.sock -m 007 app:app
         EOF'

- name: Reload daemon
  shell: systemctl daemon-reload

- name: Start gunicorn3.service
  service: name=gunicorn3 state=started enabled=yes
```

Implementation – Screenshots. Ansible. Part 3



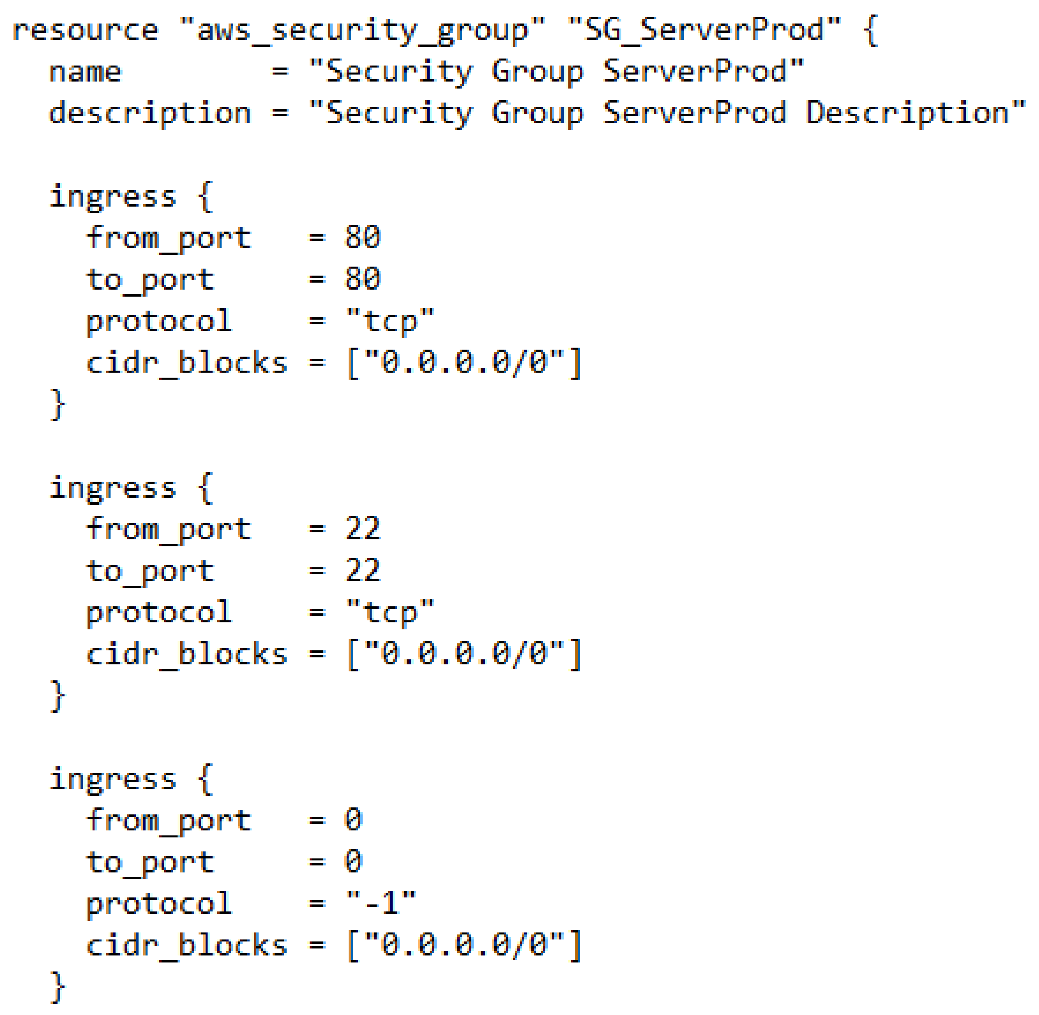
```
- name: Configure flaskapp.sock
  shell:
    cmd: |
         sudo bash -c 'cat <<EOF > /etc/nginx/sites-enabled/flaskapp
         server{
             listen 80;
             server_name 18.217.76.100 18.188.233.6;

             location / {
                 proxy_pass http://unix:/home/ubuntu/jenkins/workspace/deploy-to-test-server-ansible/flaskapp.sock;
             }
         }'

handlers:
- name: Restart nginx
  service: name=nginx state=restarted

- name: Restart gunicorn3
  service: name:=gunicorn3 state=restarted
```

Implementation – Screenshots. Terraform. Part 1

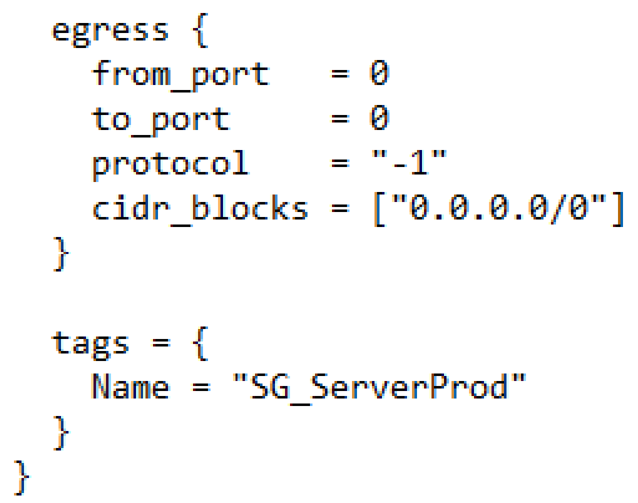


```
provider "aws" {}


resource "aws_instance" "ServerProd" {
  ami           = "ami-0b9064170e32bde34"
  instance_type = "t2.micro"
  key_name      = "server"

  vpc_security_group_ids = [aws_security_group.SG_ServerProd.id, "sg-11af8958"]
  user_data              = file("user_data.sh")

  tags = {
    Name    = "ServerProd"
    Owner   = "Mykola Shtompel"
    Project = "ServerProd by Terraform 2"
  }
}
```

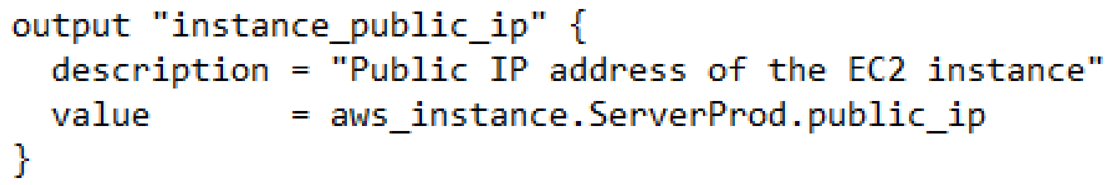
Implementation – Screenshots. Terraform. Part 2



```
resource "aws_security_group" "SG_ServerProd" {
  name        = "Security Group ServerProd"
  description = "Security Group ServerProd Description"

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
```



```
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "SG_ServerProd"
  }
}
```



```
output "instance_public_ip" {
  description = "Public IP address of the EC2 instance"
  value       = aws_instance.ServerProd.public_ip
}
```

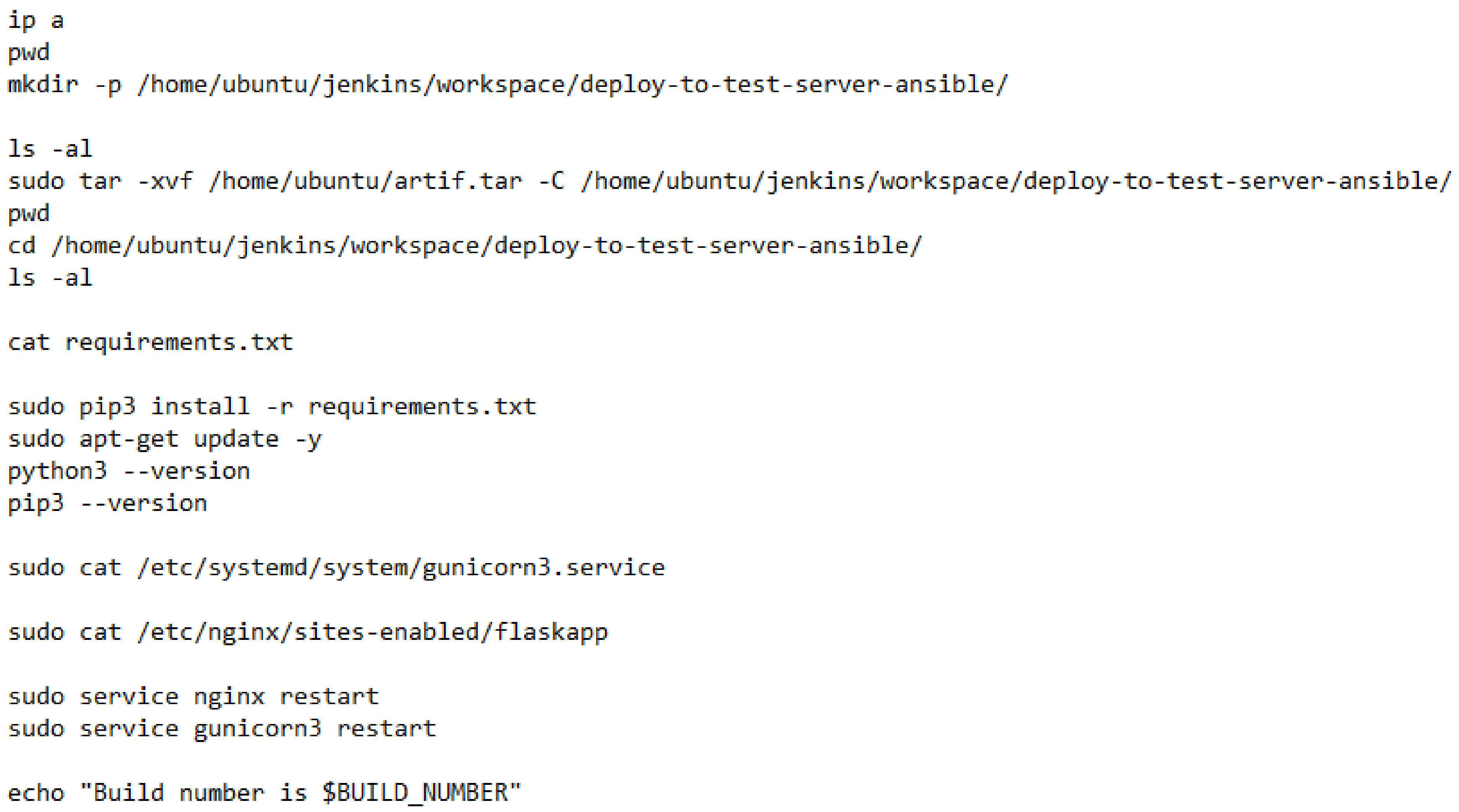
Implementation – Screenshots. Jenkins. Part 1

```
echo "--Build--"
echo "Build is starting"
ip a
ls -al

cat requirements.txt

sudo pip3 install -r requirements.txt
sudo apt-get update -y
python3 --version
pip3 --version

sudo cat /etc/systemd/system/gunicorn3.service

sudo cat /etc/nginx/sites-enabled/flaskapp

sudo service nginx restart
sudo service gunicorn3 restart

echo "Build number is $BUILD_NUMBER"

echo "Build is finishing"
echo "--Build--"
```

```
echo "--Test--"
echo "Test is starting"

sleep 10
response=$(curl -s -o /dev/null -w "%{http_code}\n" 18.217.76.100)
if [ "$response" != "200" ]
then
 exit 1
fi

echo "Test is finishing"
echo "--Test--"
```

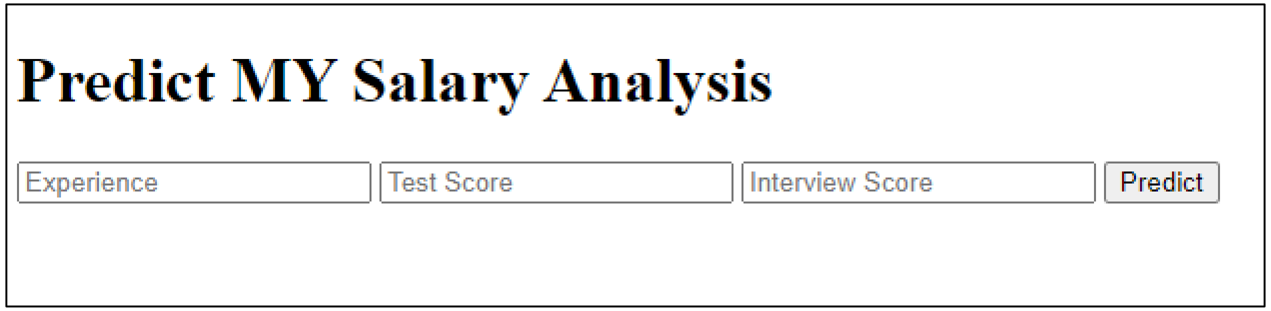
```
tar -cvf /home/ubuntu/jenkins/workspace/deploy-to-test-server-ansible/artif.tar .
cd /home/ubuntu/jenkins/workspace/deploy-to-test-server-ansible
ls -al
```

Implementation – Screenshots. Jenkins. Part 2

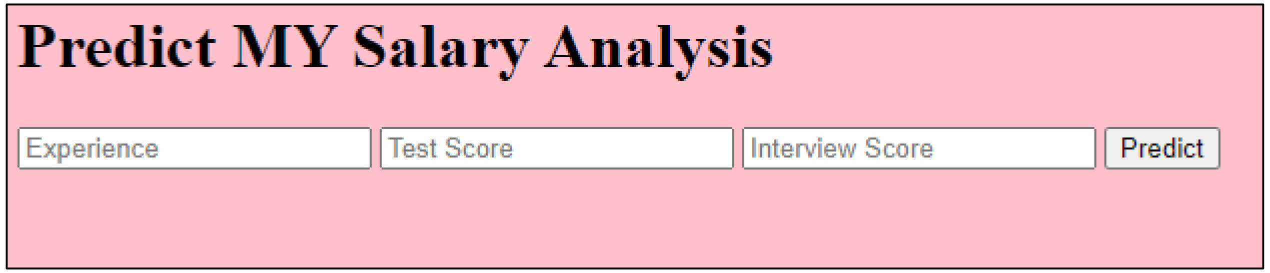


```
ip a
pwd
mkdir -p /home/ubuntu/jenkins/workspace/deploy-to-test-server-ansible/

ls -al
sudo tar -xvf /home/ubuntu/artif.tar -C /home/ubuntu/jenkins/workspace/deploy-to-test-server-ansible/
pwd
cd /home/ubuntu/jenkins/workspace/deploy-to-test-server-ansible/
ls -al

cat requirements.txt

sudo pip3 install -r requirements.txt
sudo apt-get update -y
python3 --version
pip3 --version

sudo cat /etc/systemd/system/gunicorn3.service

sudo cat /etc/nginx/sites-enabled/flaskapp

sudo service nginx restart
sudo service gunicorn3 restart

echo "Build number is $BUILD_NUMBER"
```

Demo - Results



Predict MY Salary Analysis



Predict MY Salary Analysis

Conclusion

- If necessary, the rapid deployment of a web application that uses machine learning models, it is advisable to apply to implement the CI/CD pipeline with the use of such software and technologies:
  - For web access – Flask, Gunicorn, Nginx, Amazon Web Services
  - For configuration management – Ansible
  - For creating and modification cloud infrastructure – Terraform
  - For automate the development process – Jenkins

Thank You

Github repositories:

https://github.com/mykshtompel/project

https://github.com/mykshtompel/Presentation-and-files