CI/CD Pipeline
on AWS infrastructure using Jenkins

EPAM DEVOPS ONLINE WINTER 2022

Oleksandr Makarov

DevOps is a software development approach to increase the efficiency, speed and security of development life cycle.

It involves:

- continuous development
- continuous testing
- continuous integration
- continuous deployment
- continuous monitoring

As a result, develop high-quality software and shorter development life cycle are competitive advantage for business and their customers.

Main goal of the project

I was interested to improve my skills and get new knowledge. Also I want to actualize my knowledge to nowadays reality in IT sphere.

I commenced from simple project (upload static web-site from GitHub to AWS S3 bucket using Jenkins). During this stage I was interested in infrastructure as code and decided to improve my project.

I can allocate next goals for me in this project:

- Practice in using AWS tools
- Learn Terraform (syntax, using variables)
- Bash scripting as a part of automatization for set-up Jenkins
- Using Jenkins file for create a pipeline
- Docker as a part of environment

Technology stack and tools are used in the project

I. GitHub (Source control of project, includes: node.js site; terraform configs and scripts; docker configs; jenkinsfile)
II. OS
   - Ubuntu 22.04 for EC2 instances
   - Alpine 3.16 for docker images
III. AWS
   - EC2 Instances (Jenkins server – t2.small, Production server – t2.micro)
   - ECR (keep docker images for product, stage and test containers)
   - Secret Manager (store public and private keys for GitHub)
   - S3 Bucket (keep logs and test results, terraform backend - .tfstate, Jenkins server config scripts)
   - IAM (manage roles for aws resources)
   - VPC (define isolated network for instances and attach elastic IP)

Technology stack and tools are used in the project (continue)

IV. Docker (create images for test environment, stage and product environments)

V. Terraform providers
   - aws (create aws infrastructure)
   - local (is used to manage local resources, such as files)
   - tls (create a key pairs)

VI. Jenkins (Multibranch CI/CD pipeline)
   - BlueOcean (convenient visualization of pipeline process)
   - Docker pipeline plugin

Project plan

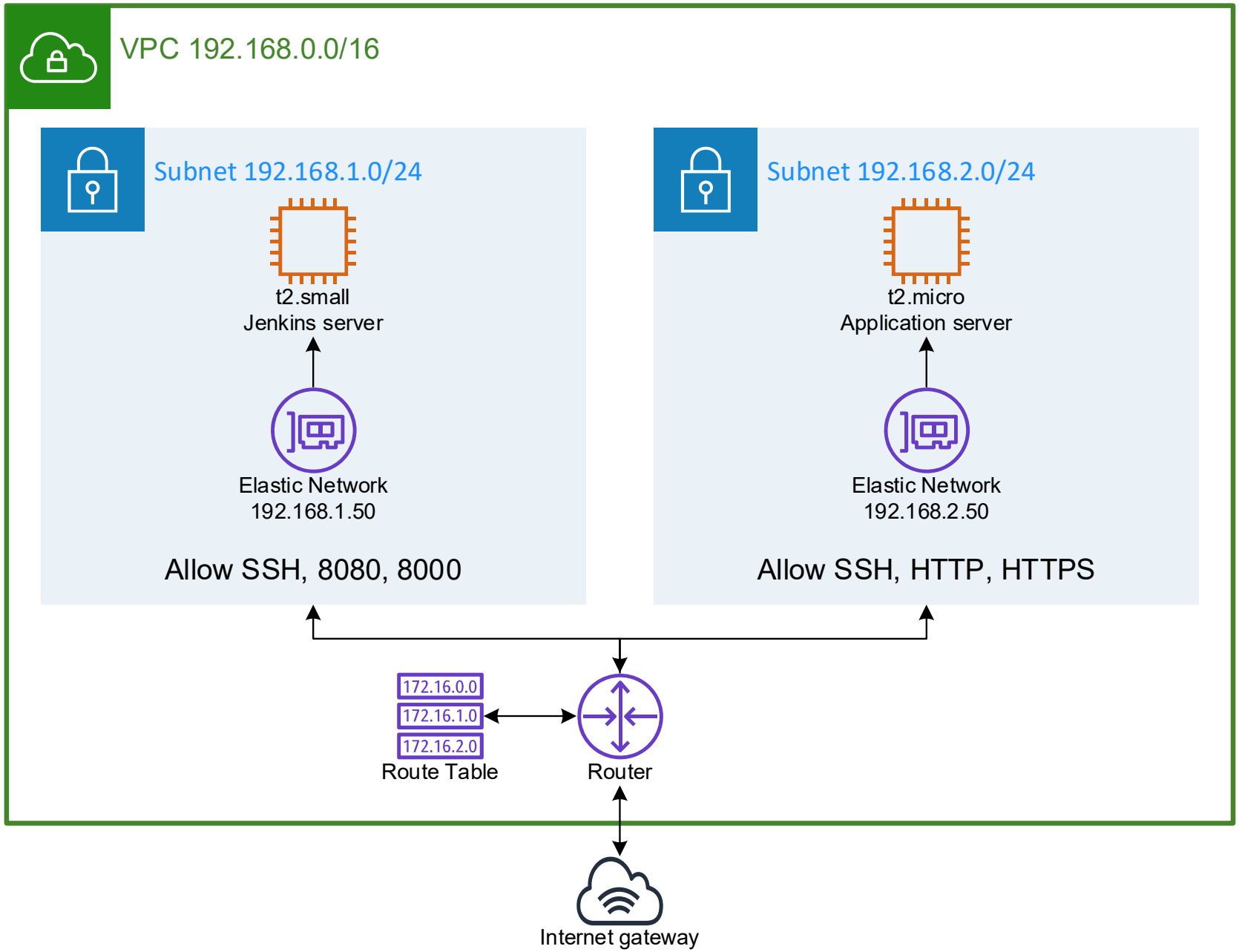
I. Configure local environment and deploy an application on virtual machine for testing the concept

II. Create AWS infrastructure using Terraform
   - Terraform backend on s3 bucket
   - VPC infrastructure
   - IAM roles
   - EC2 instances
   - Keys for instances and GitHub
   - S3 buckets
   - ECR repositories

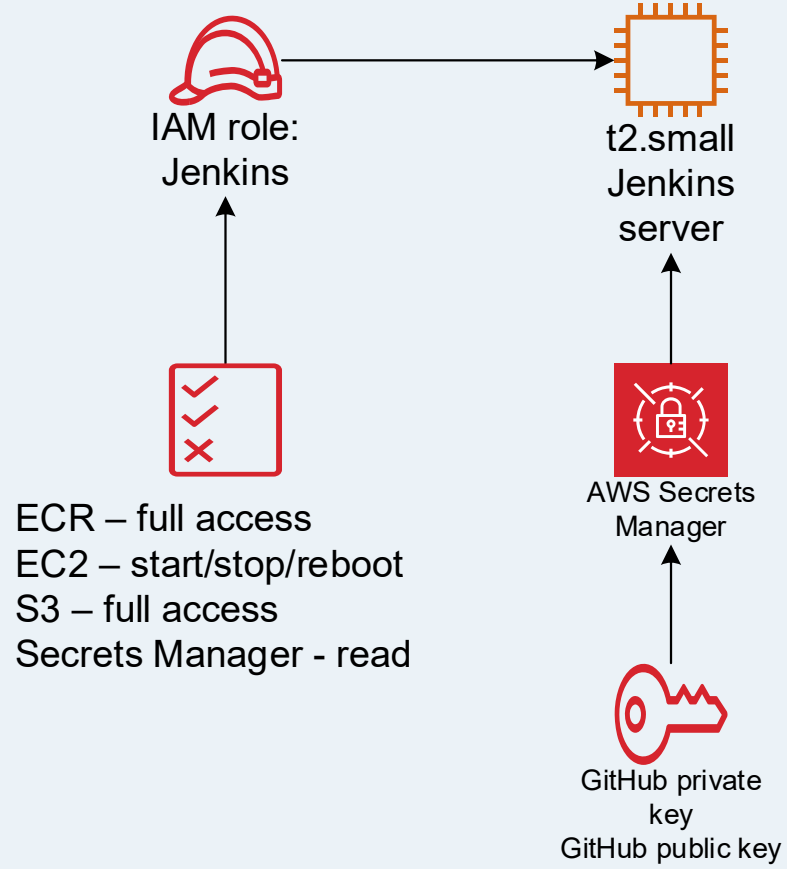
III. Create Docker files with instructions for test and production images

IV. Configure Jenkins and production server
   - user_data scripts
   - shell scripts for user_data (store on s3 bucket)
   - initial Jenkins set up using curl

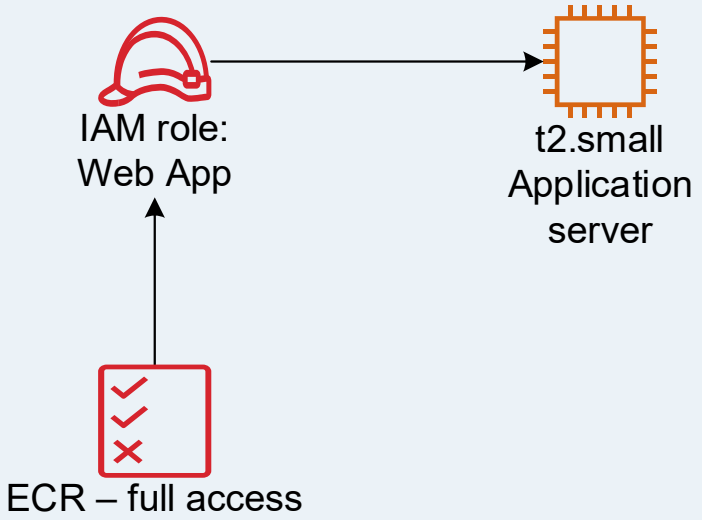
V. CI/CD Pipeline
   - create jenkisfile

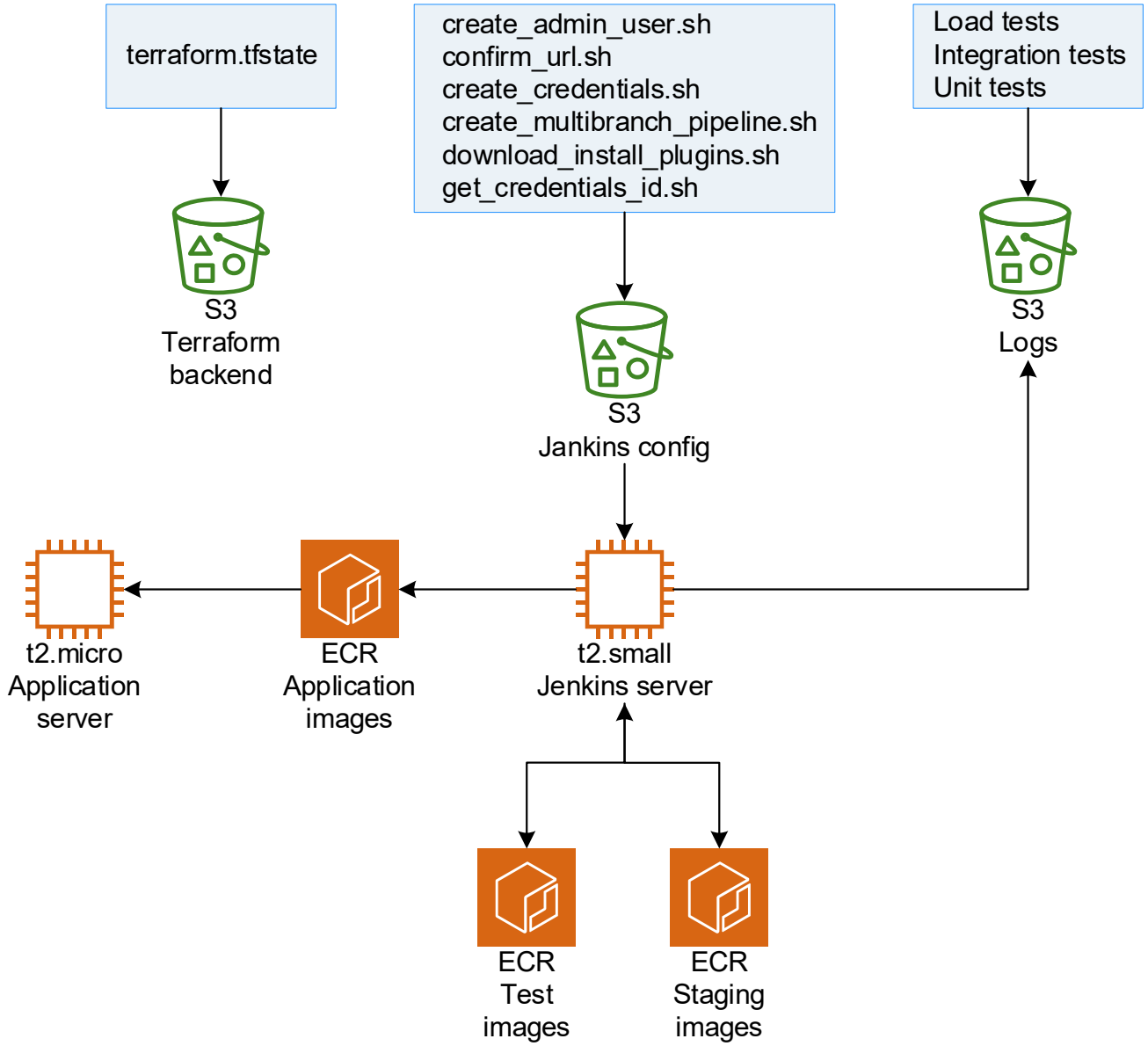

VPC 192.168.0.0/16
Subnet 192.168.1.0/24
t2.small
Jenkins server
Elastic Network
192.168.1.50
Allow SSH, 8080, 8000
Subnet 192.168.2.0/24
t2.micro
Application server
Elastic Network
192.168.2.50
Allow SSH, HTTP, HTTPS
172.16.0.0
172.16.1.0
172.16.2.0
Route Table
Router
Internet gateway

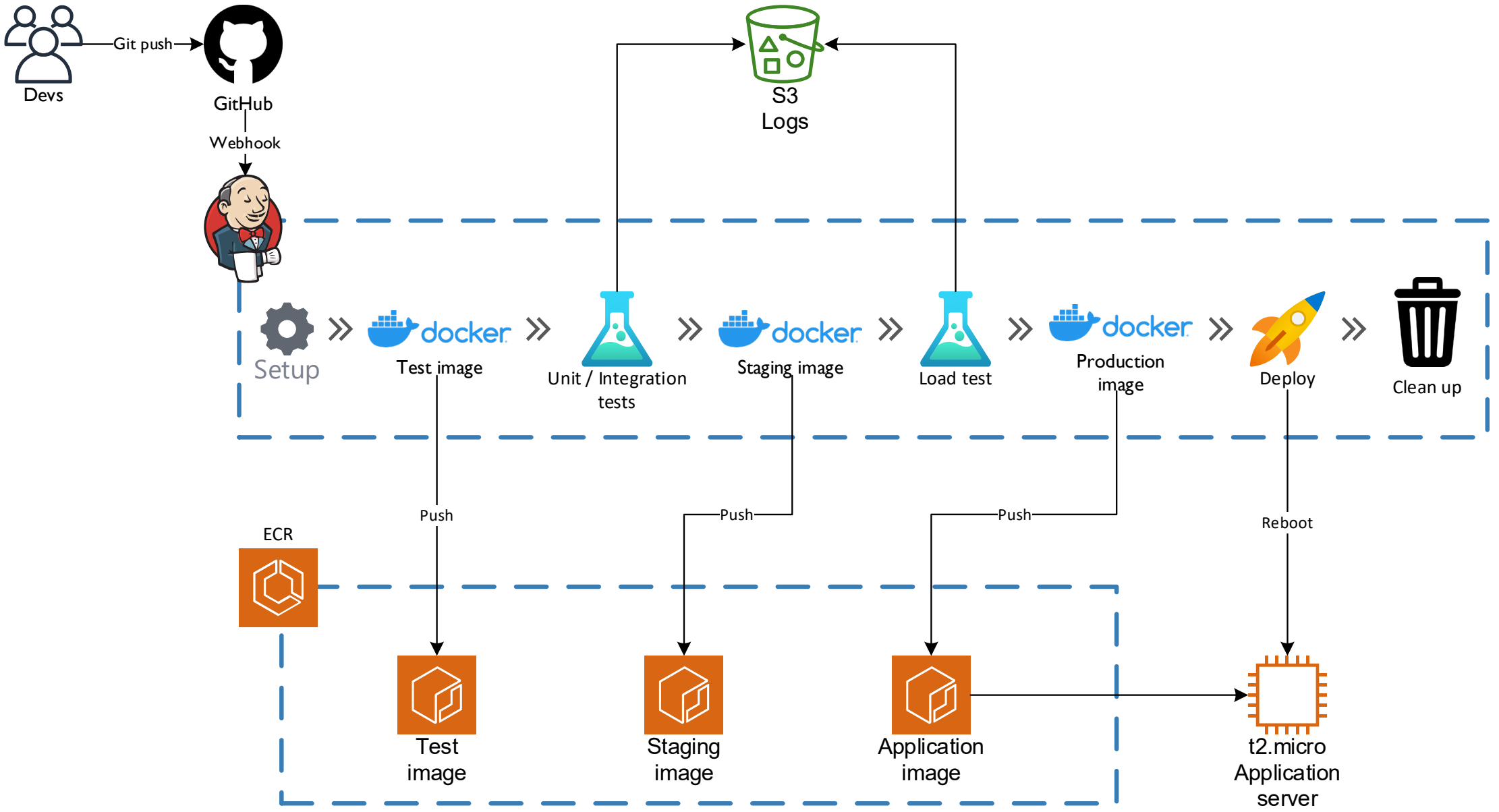
Jenkins permissions



Application permission




terraform.tfstate
create_admin_user.sh
confirm_url.sh
create_credentials.sh
create_multibranch_pipeline.sh
download_install_plugins.sh
get_credentials_id.sh
Load tests
Integration tests
Unit tests
S3
Terraform
backend
S3
Jankins config
S3
Logs
t2.micro
Application
server
ECR
Application
images
t2.small
Jenkins server
ECR
Test
images
ECR
Staging
images


Devs
Git push
GitHub
Webhook
S3
Logs
Setup
Test image
Unit / Integration tests
Staging image
Load test
Production image
Deploy
Clean up
Push
Push
Push
Reboot
ECR
Test image
Staging image
Application image
t2.micro Application server

Conclusion

The goals set in this project have been achieved.

Ideas for improvement:

- integrate Ansible as a tool for configuring servers
- create and administrate database server for storing data
- adjust autoscaling