Deployment 5

Brayan Molina

November 13, 2022

Table of Contents

Objectives:

This deployment will build on the information used in previous deployments of the flask application, Url-shortner. The objective of this deployment will be the following:

- Demonstrate ability to deploy a containerized application.

Documentation:

Jenkins Server Configuration and Installation:

- EC2 instance on AWS configured on default VPC with the following:
  - Port 22 – SSH
  - Port 80 – HTTP
  - Port 8080 – Jenkins
  - Jenkins script used for default installation
- Details for user configured on Amazon's IAM will be provided to Jenkins through Jenkins credentials manager. This will allow Jenkins to recognize the AWS access key and secret key needed to deploy our application using Terraform and the terraform agent.

Terraform and Docker Installation & Agent Profiles:

Installation:

- Two EC2 instances will be configured to serve as a terraform and docker agent.

- Terraform installed on an EC2 instance through the following command:

```
wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor |
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-
keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -
cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update && sudo apt install terraform
```

- Docker installed on EC2 instance through the following command:

```
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-compose-plugin
```

Agent Profiles:

Docker agent:

- Agent name: dockerage
- Create a root directory called /home/ubuntu/dockerf
- Usage: Only build jobs with label expressions matching this node. This will be used by our Jenkinsfile to redirect the steps of creating the image and pushing the image to Dockerhub to the Docker agent.
- Launch agents via SSH: This will be used by Jenkins to launch our agent using the private key associated with the server and the I.P address of the Docker EC2 server.
- Host key verification strategy: Non verifying Verification Strategy

Terraform Agent:

- Agent name: terrage
- Create a root directory called /home/ubuntu/terraf
- Usage: Only build jobs with label expressions matching this node. This will be used by our Jenkinsfile to redirect the steps required to provision our infrastructure using the Terraform agent.
- Launch agents via SSH: This will be used by Jenkins to launch our agent using the private key associated with the server and the I.P address of the Terraform EC2 server.
- Host key verification strategy: Non verifying Verification Strategy

Pipeline Build in Jenkins:

Jenkinsfile:

Build and Test:

- Build and Test stages are configuring virtual environment and running flask application, as well as testing function found inside the application source code.
- The build and test stages will be done by the Jenkins controller. This was configured because the Jenkins controller had the necessary dependencies installed, default-jre and python3.10-venv as defined by the Jenkinsfile.

Image and Push:

- Addition of dockerfile was added to the repository to build python image with our application requirements.
- User must be added to docker group to have permissions to run dockerfile.
- According to documentation, the newgrp command is used to change the current GID (group ID) during a login session. This is used to avoid restarting the server to pass the docker group changes.
- Credentials need to be added to Jenkins credential manager as username and password. These credentials will be necessary to push our image to Docker hub.

- Using withDockerRegistry() allows us to use the credentials we saved into our Jenkins to run the commands in the push stage.
- The push stage will tag the local image made by our dockerfile and then push to Docker hub.
- **NOTE: Repository that will contain our image in Docker Hub must be made manually by user before running the pipeline. Image will then be stored in the Docker repository.**

Terraform steps(init,plan,apply,destroy):

- Credentials provided to Jenkins will be stored as environment variables active within the scope of the step.
- Dir step will be used to change to Terraform directory containing configuration for our resources.
- Terraform will be used to deploy our application using ECS and Fargate.
- Terraform agent will be needed for the Init, Plan, Apply, and Destroy stages in the Jenkinsfile.
- Line 33 will contain the tag of the image pushed to Docker Hub in previous step
- Line 55 and 56 will have the ARN of the role configured to have ECS execution permissions to build our containers on ECS.

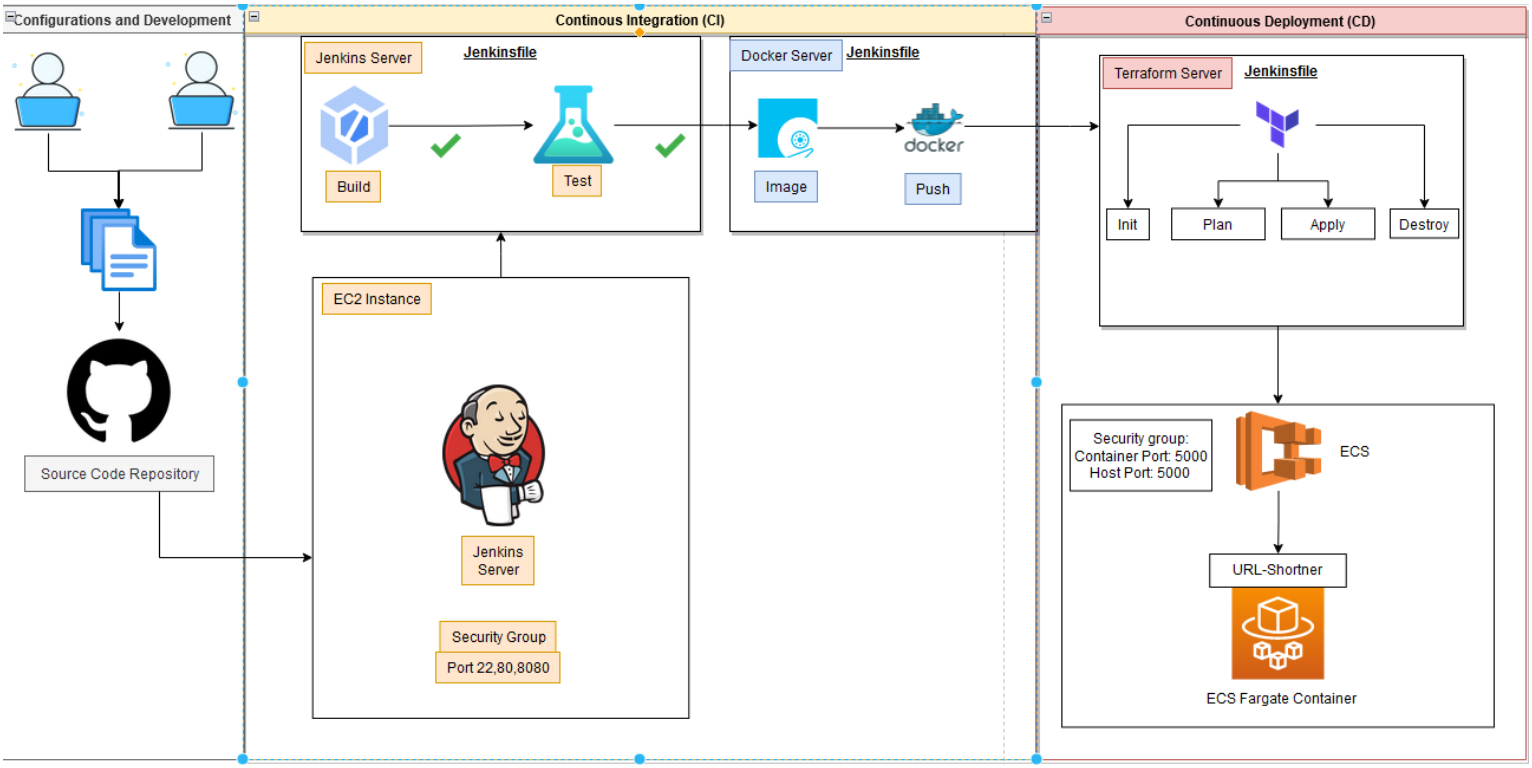
Observations and Errors:

Error	*Cause*	*Solution*
The virtual environment was not created successfully because ensurepip is not available. On Debian/Ubuntu systems, you need to install the python3-venv package using the following command.	Virtual Environment command not installed in the Jenkins server	Run apt install python3.10-venv to install venv into your server.
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock:	Docker account not connected to Jenkins	Add Docker credentials to Jenkins credential manager
test3/bin/activate: No such file or directory py.test: command not found	Ensure that the correct agent is being used for these steps. Check that being installed	Configure Jenkinsfile to use agent that has correct dependencies. Default-JRE installation.
An image does not exist locally with the tag: bmol5/flask	Incorrect tag being used	Verify that Docker is using the correct tag when pushing to Docker Hub

Improvements and Notes:

- Improvements can be made by substituting Fargate with an EC2 if you want to control your resources.
- Increasing the interval time in Terraform will delay the time health checks are performed and provide ECS the time the container needs to start the application.
- Docker Hub has a cache system that may result in your image not updating when using Jenkins as a CI tool. Delete the repository on Docker Hub to 'update' your image.

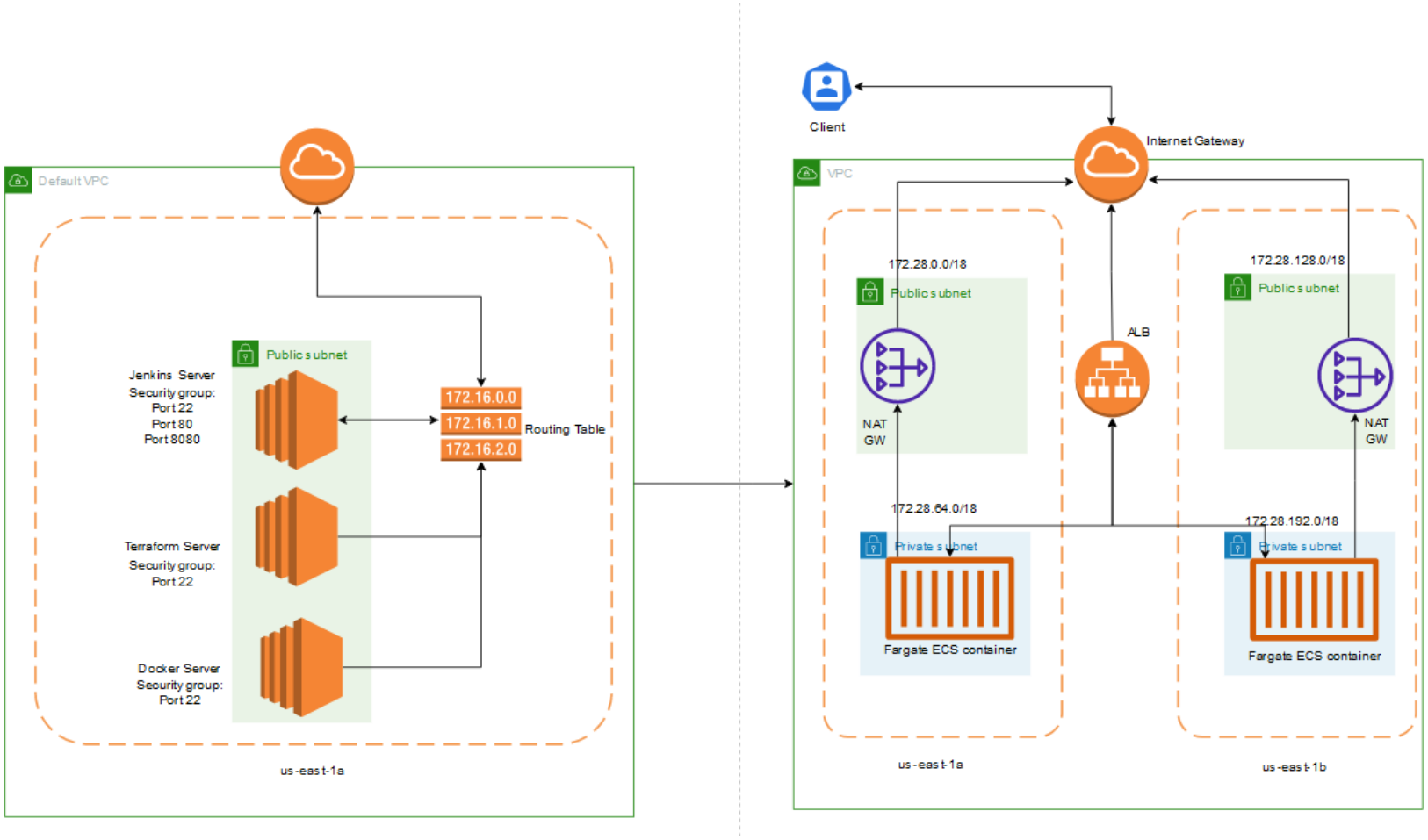
CI/CD Pipeline:



Key takeaways:

- Docker server will create the image using a Dockerfile and push to Docker Hub. Credentials must be added to Jenkins manager for this step to successfully be made.
- The terraform configuration files will be used to create task and cluster in ECS and use Fargate to deploy our application. An application load balancer will also be configured to monitor health checks and route to our containerized application.

VPC Diagram:



Key Takeaways:

- The ALB exists in both public subnets and can retrieve information from the Fargate serverless infrastructure deploying the ECS container.
- The ALB will provide health checks to the container. If health checks fail, Fargate will deploy another task(container)
- Terraform will provision the entire infrastructure on AWS from the Terraform Server.
- Port 5000 will be open on the container and the host machine for the container.

Images:

Interface Type	Description	Instance ID	Status	Primary private IPv4 address
nat_gateway	Interface for NAT Gatew...	–	In-use	172.28.53.132
Elastic network interface	ELB app/url-lb/c53a937...	–	In-use	172.28.22.135
Elastic network interface	ELB app/url-lb/c53a937...	–	In-use	172.28.131.131

Figure 1. I.P addresses demonstrating load balancer can be found public subnets

Registered targets (3)

Filter resources by property or value

IP address	Port	Zone	Health status	Health status details
172.28.249.199	5000	us-east-1b	draining	Target deregistration is in progress
172.28.85.241	5000	us-east-1a	initial	Target registration is in progress
172.28.251.16	5000	us-east-1b	draining	Target deregistration is in progress

Figure 2. Container I.P addresses being started by Fargate after failing health checks.

Cluster	urlapp-cluster
Launch type	FARGATE
Platform version	1.4.0
Task definition	url-task:11
Group	service:url-ecs-service
Task role	ecstaskEX
Last status	STOPPED
Desired status	STOPPED
Created at	2022-11-13 19:38:29 -0500
Started at	2022-11-13 19:39:01 -0500
Stopped at	2022-11-13 19:41:40 -0500
Stopped reason	Task failed ELB health checks in (target-group arn:aws:elasticloadbalancing:us-east-1:665532315280:targetgroup/url-app/b0ba7d5825f00f9a)

Figure 3. Task failed in ECS due to failing health checks.

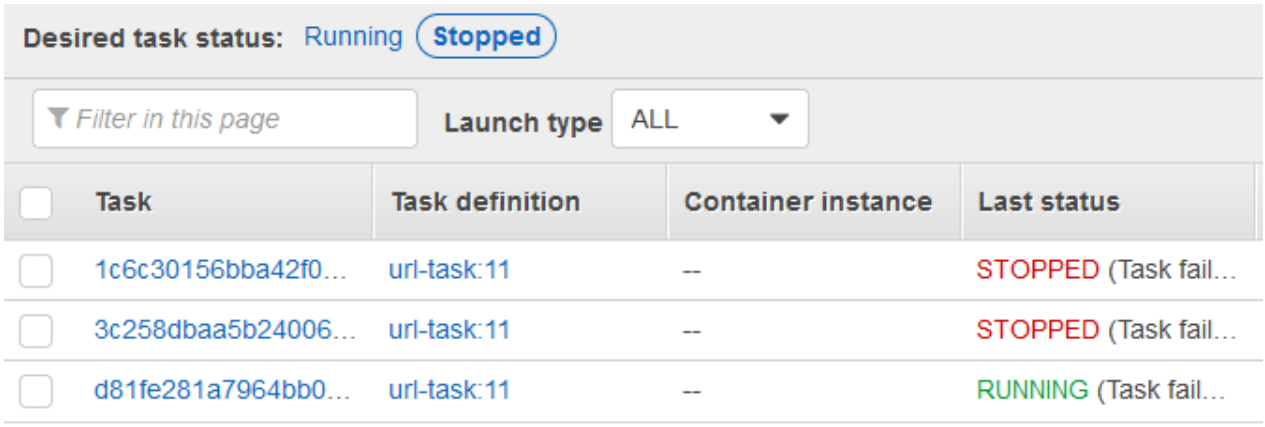


Figure 4. New tasks being formed by Fargate after health check fails

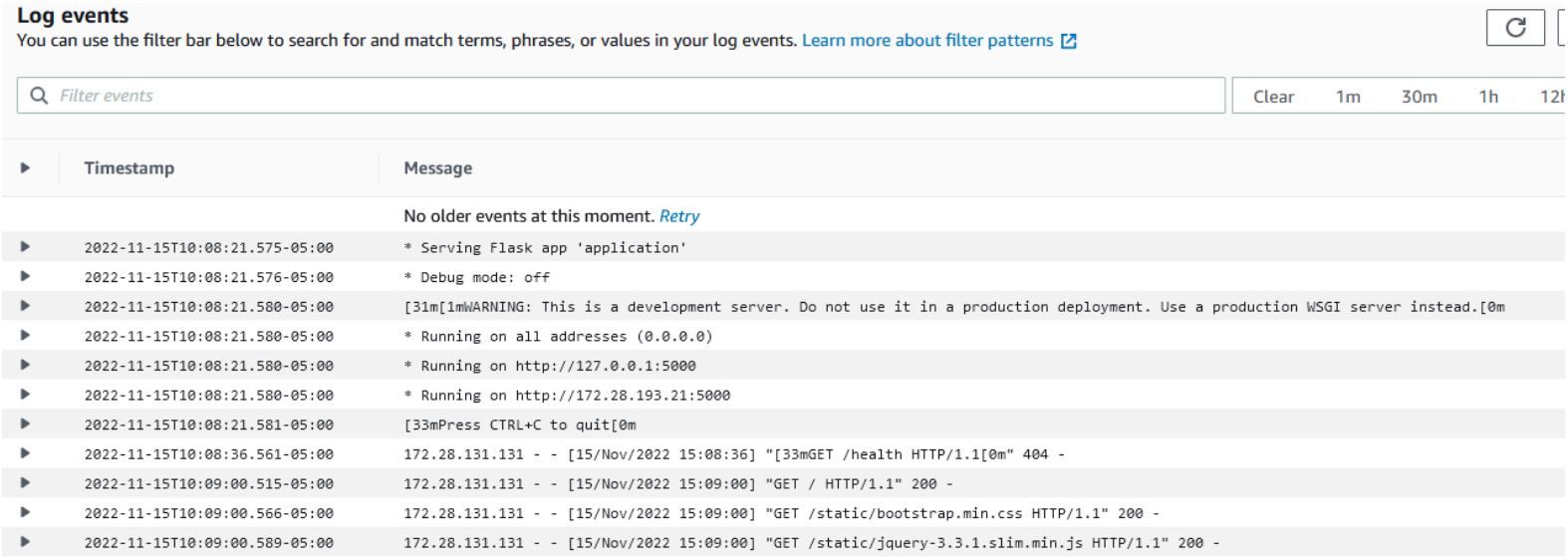


Figure 5. Successful deployment of containerized application

Sources:

1. https://www.terraform.io/downloads
2. https://docs.docker.com/engine/install/ubuntu/
3. https://www.jenkins.io/doc/pipeline/steps/credentials-binding/
4. https://www.jenkins.io/doc/pipeline/steps/docker-workflow/#withdockerregistry-sets-up-docker-registry-endpoint
5. https://stackoverflow.com/questions/54503360/aws-ecs-error-task-failed-elb-health-checks-in-target-group
6. https://docs.oracle.com/en-us/iaas/Content/Balance/Concepts/load_balancer_types.htm