DevSecOps Final Project

1

Personal details:

Author: Shira Shani

Email: shira.shani@grunitech.com

Date: 08/11/2023

Platform: Google Cloud Platform (GCP)

reference to my jenkins freestyle project step by step

Table of Contents:

Introduction to the project:

In this DevSecOps project, we will be implementing a secure and efficient development pipeline for a web application called "DevConnect." Our journey will include key stages such as **Dockerization**, GitHub repository setup, **deployment** automation, Kubernetes cluster creation, bug fixing, and the establishment of a **CI/CD** pipeline using Jenkins. The goal is to ensure continuous integration, testing, and deployment of the application while maintaining security and reliability throughout the development process. This guide will walk you through each step

DevSecOps Final Project

2

Let’s get started!!

Dockerization:

Dockerize the application

First I dockerized the application, for that i added 2 files: dockerfile and requirements.txt:

The dockerfile:

```
FROM python:latest

# Set the working directory to /app
WORKDIR /app

# Install required Python packages from requirements.txt
RUN pip install -r requirements.txt

# Copy the Django project files into the container
COPY . .

# Expose the Django development port
EXPOSE 8000

# Start the Django development server
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

The requirements.txt:

```
Django==4.2.7
Pillow==10.1.0
django-crispy-forms==2.1
```

DevSecOps Final Project

3

TemplateDoesNotExist at /login/

bootstrap4/uni_form.html

Request Method:	GET
Request URL:	http://localhost:8000/login/
Django Version:	4.2.7
Exception Type:	TemplateDoesNotExist
Exception Value:	bootstrap4/uni_form.html
Exception Location:	/usr/local/lib/python3.12/site-packages/django/template/backends/django.py, line 84, in reraise
Raised during:	django.contrib.auth.views.LoginView
Python Executable:	/usr/local/bin/python
Python Version:	3.12.0
Python Path:	['/app', '/usr/local/lib/python312.zip', '/usr/local/lib/python3.12', '/usr/local/lib/python3.12/lib-dynload', '/usr/local/lib/python3.12/site-packages']
Server time:	Sun, 12 Nov 2023 07:41:30 +0000

I added to the requirements.txt
crispy-bootstrap4==2023.1

```
Django==4.2.7
Pillow==10.1.0
django-crispy-forms==2.0.0
crispy-bootstrap4==2023.1
```

And added to the instslled_apps list `crispy_bootstrap4`

```
INSTALLED_APPS = [
    'blog.apps.BlogConfig',
    'users.apps.UsersConfig',
    'crispy_forms',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'crispy_bootstrap4'
]
```

DevSecOps Final Project

Reference to stackoverflow where I found the solution to this err
https://stackoverflow.com/questions/75495403/django-returns-templatedoesnotexist-when-using-crispy-forms

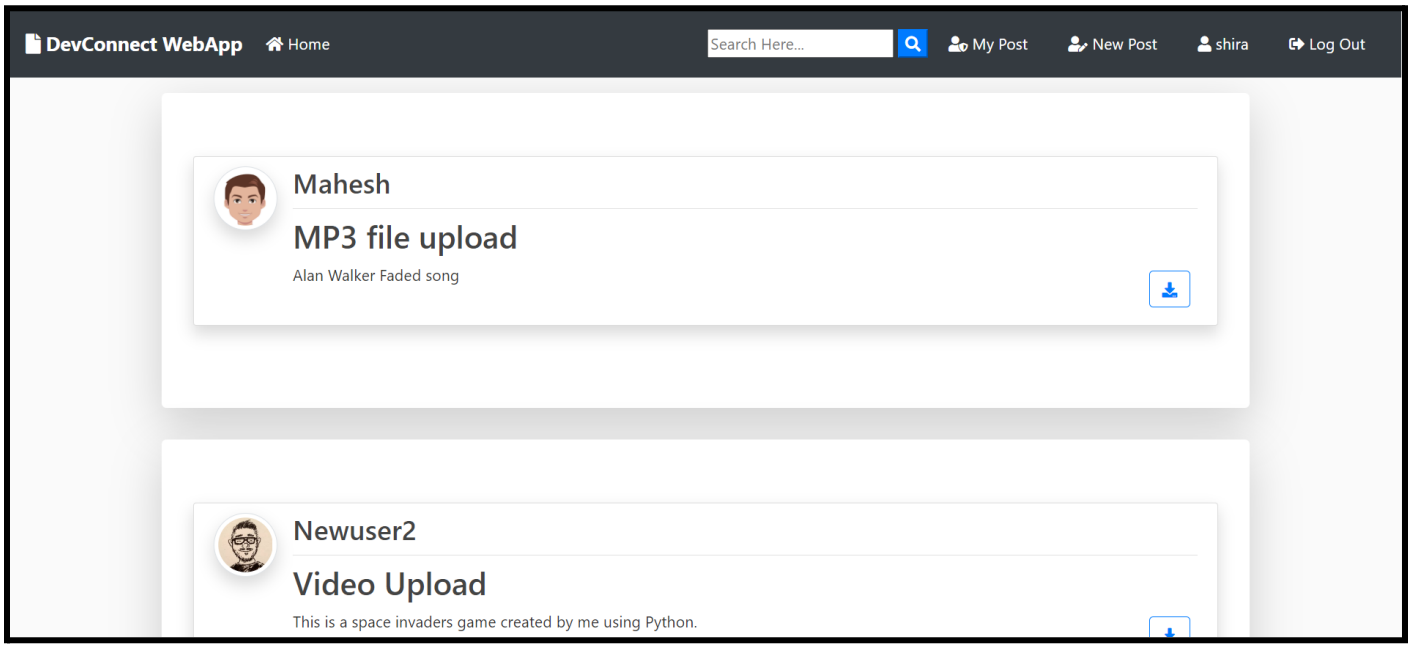
DevSecOps Final Project

5

Run the application and make sure everything is working.

Now I built the image with the command: `docker build -t my-python-app .`
And then run it: `docker run -p 8000:8000 my-python-app`

And the app is up and running locally in port 8000: http://localhost:8000/



DevSecOps Final Project

6

Create init.sh and delete.sh for automation.

init.sh:

```
#!/bin/bash
# Build the Docker image
docker build -t my-python-app .
# Run the Docker container, mapping port 8000
docker run -d -p 8000:8000 my-python-app
# Display container ID for reference
container_id=$(docker ps -q --filter "ancestor=my-python-app")
echo "Docker container is running with ID: $container_id"
```

Delete.sh:

```
#!/bin/bash
# Stop the Docker container
container_id=$(docker ps -q --filter "ancestor=my-python-app")
if [ -n "$container_id" ]; then
    docker stop "$container_id"
    echo "Docker container with ID $container_id has been stopped."
else
    echo "No running Docker container found."
fi
# Remove the Docker container
container_id=$(docker ps -aq --filter "ancestor=my-python-app")
if [ -n "$container_id" ]; then
    docker rm "$container_id"
    echo "Docker container with ID $container_id has been removed."
else
    echo "No Docker container found for removal."
```

DevSecOps Final Project

7

Run the application with volume and make it persistent -

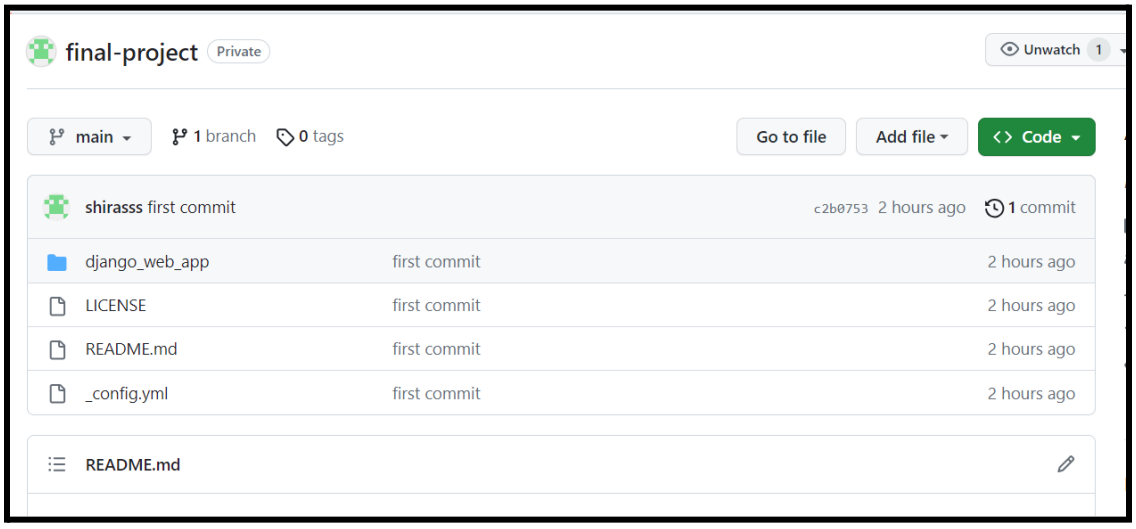
check by signing up to the app, delete the container and then start the container and log in without signing up,

I added a docker-compose.yaml file:

```yaml
version: '3'
services:
  web:
    build: .
    ports:
      - "8000:8000"
    volumes:
      - my-django-data:/app/media
      - ./db.sqlite3:/app/db.sqlite3
    environment:
      - DJANGO_DB_HOST=db
volumes:
  My-django-data:
```

Now I ran : docker-compose up then signed in, deleted the container and ran again docker-compose up and logged in →it logged me in!-> it is persistent ✅

Create a new Private Github repository and push your code to it.

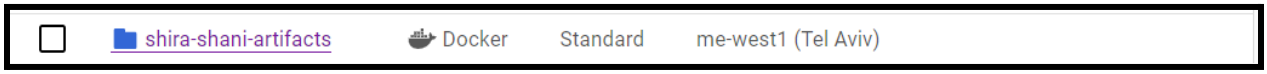


DevSecOps Final Project

8

Deployment

Create an artifact repository called <your-name>-artifacts in the me-west1 region and automate a deployment of the web app image to it.

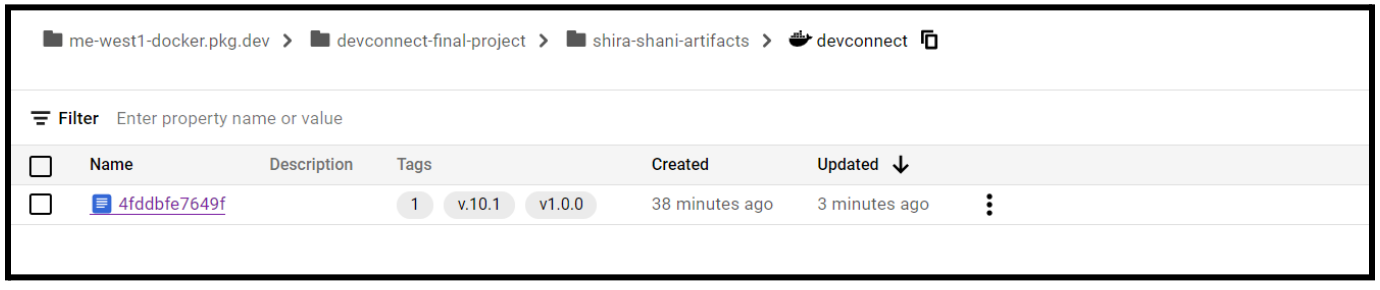


I created a deploy.sh file in a scripts folder in the project:\ for deployment automation

```
image_name=devconnect
echo "enter the version"
read version
local_image_name=django_web_app-web:latest
gcloud auth login
artifact_registry_image=me-west1-docker.pkg.dev/devconnect-final-project/
shira-shani-artifacts/${image_name}:${version}
docker tag ${local_image_name} ${artifact_registry_image}
docker push ${artifact_registry_image}
```

I pushed the **`django_web_app-web:latest`** image I have locally to the artifact registry with the deploy.sh file:
Inside the repository shira-shani-artifacts:

And inside that I have the image tags I pushed:



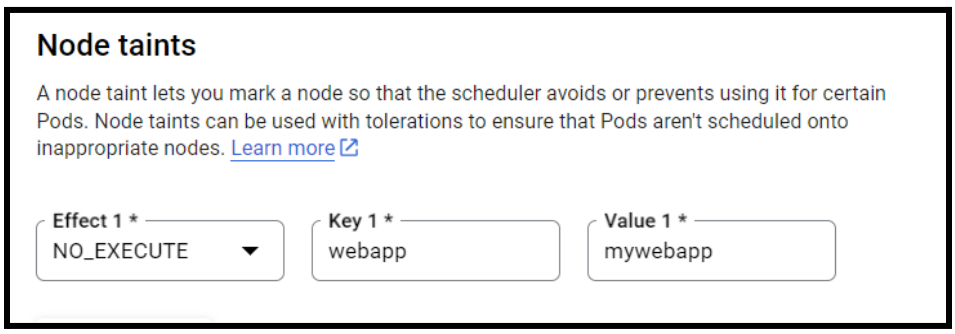
DevSecOps Final Project

9

Deploy a zonal GKE standart cluster called <your-name>-

cluster with the following specifications:
IF NOT MENTIONED LEAVE AT DEFAULT
Zone: me-west1-a/b/c
Node pool 1:
a. name - devconnect-app
b. nodes - 1
c. machine type - e2-micro (2 vCPU, 1 core, 1 GB memory)
d. service account - assign DevOps-sa
e. boot disk - 10 gb.
f. node taints - NO_EXECUTE, key=webapp, value=mywebapp.(read and understand taints)
In the cloud shell I ran the following command:

```
gcloud container clusters create shira-shani-k-cluster \
  --zone=me-west1-a \
  --node-locations=me-west1-a,me-west1-b,me-west1-c \
  --node-pool=devconnect-app:1 \
  --machine-type=e2-micro \
  --num-nodes=1 \
  --service-account=DevOps-sa \
  --boot-disk-size=10GB
```



DevSecOps Final Project

10

In a namespace called production, create 1 replica deployment to the app.

Create the namespace production :

kubectl create namespace production

configure kubectl, to use a particular Google Kubernetes Engine (GKE) cluster named "shira-shani-cluster" in the "us-central1" region

gcloud container clusters get-credentials shira-shani-cluster --region us-central1

I created a deployment.yaml file by nano deployment.yaml and copied this content to it :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: "shira-shani-app-deployment"
  namespace: production
spec:
  replicas: 1
  selector:
    matchLabels:
      app: "app-deployment"
  template:
    metadata:
      labels:
        app: "app-deployment"
    spec:
      tolerations:
        - key: "webapp"
          operator: "Equal"
          value: "mywebapp"
          effect: "NoExecute"
      containers:
        - name: "devconnect"
          image:
"me-west1-docker.pkg.dev/devconnect-project/shira-shani-artifacts/devconnect:v1.0.0"
```

DevSecOps Final Project

And ran in the cloud shell: kubectl apply -f deployment.yaml
And the deployment is succeeded: ✅

Reference to Control scheduling with node taints that helped me solve the err in the deployment:ERROR:“Does not have minimum availability”
https://cloud.google.com/kubernetes-engine/docs/how-to/node-taints

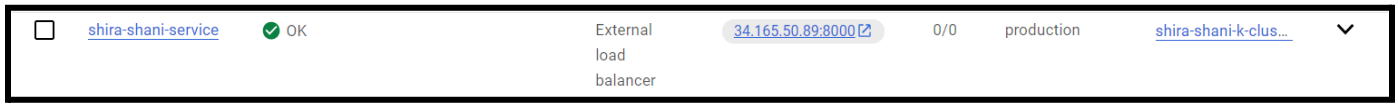
Expose it using load balancer service and access it through a browser:

I created a service.yaml file:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: shira-shani-service
  namespace: production
spec:
  selector:
    app: "app-deployment"
  ports:
    - protocol: TCP
      port: 8000
      targetPort: 8000
  type: LoadBalancer
```

Under the selector I wrote: app: "django-app" which matches the value in the deployment.yaml file under the selector

And ran the command: kubectl apply -f service.yaml

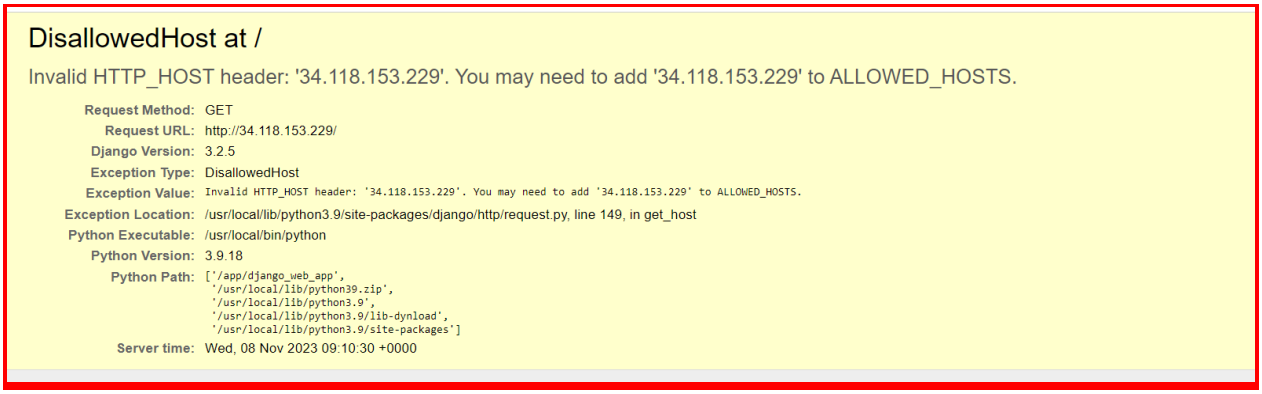


DevSecOps Final Project

12

Fix the bug and upload to the artifact repository a new version with the corrected bugfix.

The bug that I need to fix:



In order to fix this bug I changed in the settings.py the ALLOWED_HOSTS from being empty to: **`ALLOWED_HOSTS = ['*']`**

Rollout the new version deployment:

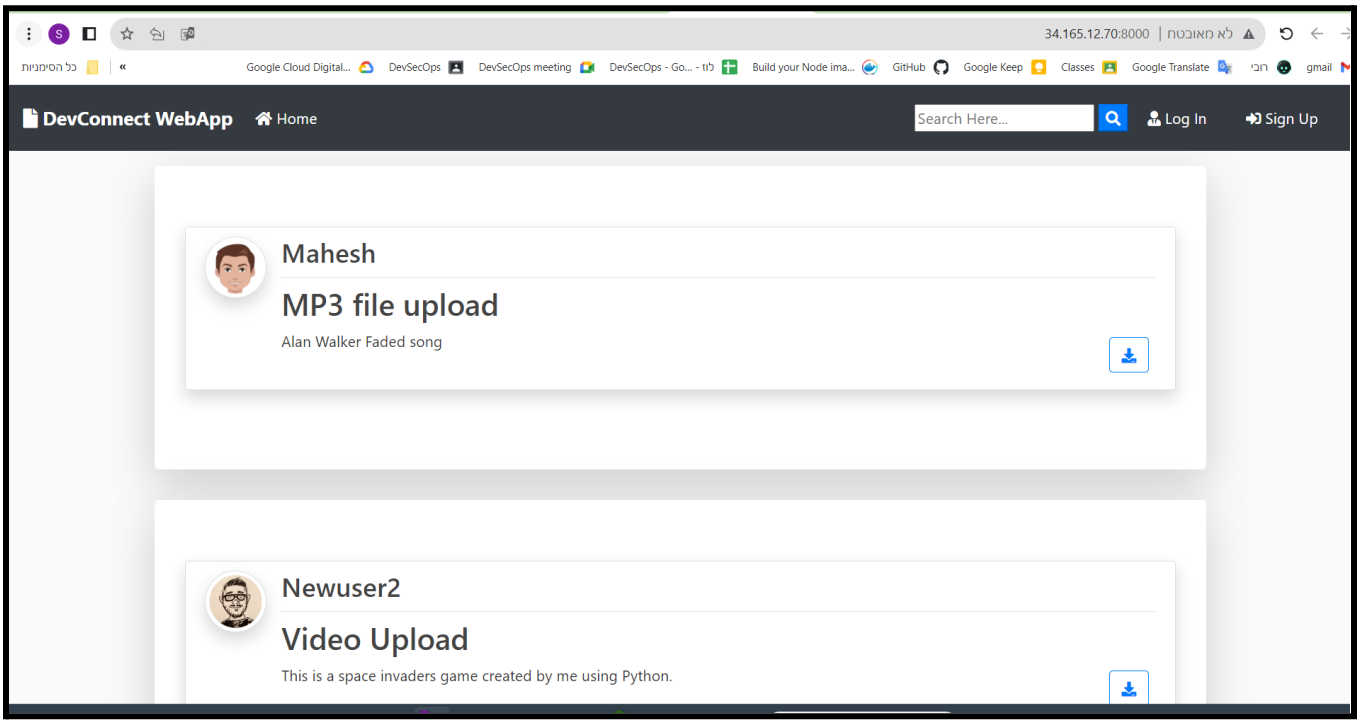
Now I built the app again by init.sh file and pushed the updated version to the artifact- v3.0.0, by the deploy.sh file

And updated the deployment image version to the fixed one

DevSecOps Final Project

13

And the application is accessed through the browser:in the address: http://35.224.16.151:8000/ 👏👏👏



DevSecOps Final Project

14

CI/CD:

Create a Compute engine instance with the following specs:

a. Name - <your-name>-jenkins.
b. Region - me-west1(Tel-Aviv)
c. Machine type - e2-medium (2 vCPU, 1 core, 4 GB memory)
d. service account - assign DevOps-sa
e. boot disk - 10 gb.
f. Automation - install docker engine.

Automation:

```
#!/bin/bash
sudo apt-get update
sudo apt-get -y install docker.io
```

DevSecOps Final Project

15

Create a new local repository called jenkins_lab and use it to create an automation deployment from your local laptop that builds your jenkins image from freestyle project, uploads it to the artifact registry and runs it inside the compute engine instance, make sure to run with volume for persistence.
Make sure the container can use docker!

A dockerfile for building the jenkins image :

```
FROM jenkins/jenkins:lts-jdk17
USER root
RUN groupadd -g 997 docker
RUN gpasswd -a jenkins docker

RUN apt-get update && apt-get install -y docker.io
RUN curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-compose
-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose --insecure
RUN chmod +x /usr/local/bin/docker-compose
# Install Docker CLI
RUN apt-get update && \
    apt-get install -y apt-transport-https ca-certificates curl
software-properties-common && \
    curl -fsSL https://get.docker.com | sh && \
    apt-get clean
RUN apt-get update && apt-get install -y curl gnupg
RUN echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg]
http://packages.cloud.google.com/apt cloud-sdk main" | tee -a
/etc/apt/sources.list.d/google-cloud-sdk.list
RUN curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key
--keyring /usr/share/keyrings/cloud.google.gpg add -
RUN apt-get update && apt-get install -y google-cloud-sdk
RUN usermod -aG docker jenkins
EXPOSE 8080
USER jenkins
```

DevSecOps Final Project

16

I ran docker build -t jenkins_image.
And created a deploy.sh file for pushing the jenkins image

```
image_name=jenkins
local_image_name=jenkins_push
echo "enter the version"
read version
artifact_registry_image=me-west1-docker.pkg.dev/devconnect-project/
shira-shani-artifacts/${image_name}:${version}

gcloud auth login
docker tag ${local_image_name} ${artifact_registry_image}
docker push ${artifact_registry_image}
```

After the image was pushed to the artifact by the deploy.sh file with the version I entered: v1.0.0:

I ssh into the vm, then I tried to write there docker commands but got error:

```
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docke
r.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/json": dial unix /var/run/dock
er.sock: connect: permission denied
```

So I ran : sudo chmod 666 /var/run/docker.sock
To change the permissions on the Docker socket file to allow the containerized Docker client to communicate with the host's Docker daemon, This is commonly done when running Docker commands inside a container,
I ran it as a sudo -(it was not allowed not as the sudo)
And the problem was solved!

DevSecOps Final Project

I ran the following commands:

configure the Docker CLI to authenticate to the Google Container Registry (GCR) in the us-west1 region to be able to push and pull Docker images to and from GCR:

```
gcloud auth configure-docker me-west1-docker.pkg.dev
```

Pull the jenkins image I just pushed:

```
docker pull
me-west1-docker.pkg.dev/devconnect-project/shira-shani-artifacts/jenkins:v1.0.0
```

I created volume named jenkins_home by the command:

```
docker volume create jenkins_home
```

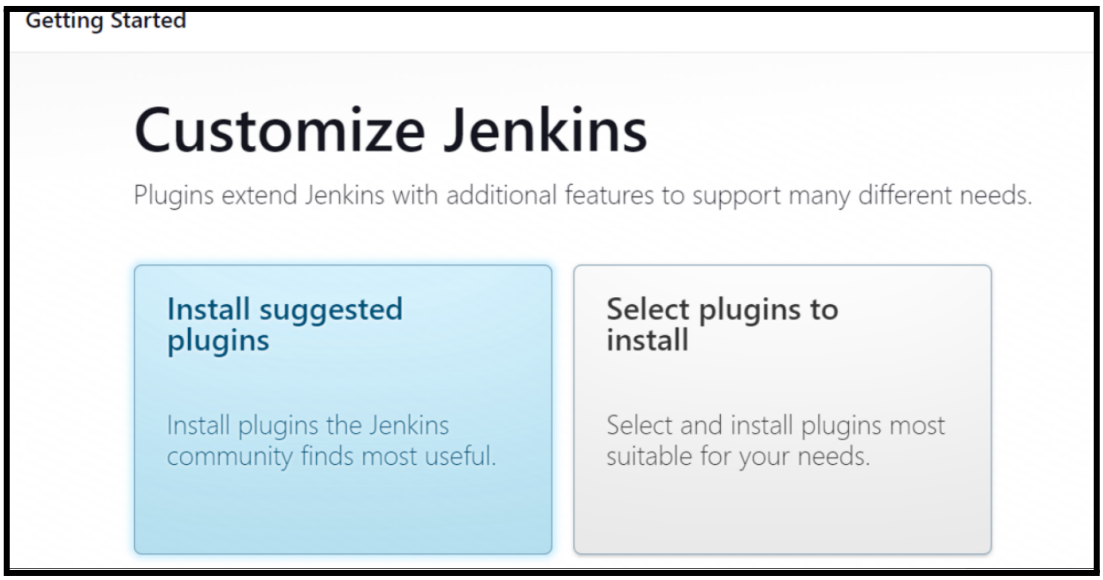
Then ran the image I pulled with the volume for persistence

```
docker run -d -p 8080:8080
 -v /etc/ssl/certs:/etc/ssl/certs
-v jenkins_home:/var/jenkins_home
-v /var/run/docker.sock:/var/run/docker.sock
--name jenkins --restart=on-failure
me-west1-docker.pkg.dev/devconnect-final-project/shira-shani-artifacts/devconnect:v1.0.0
```

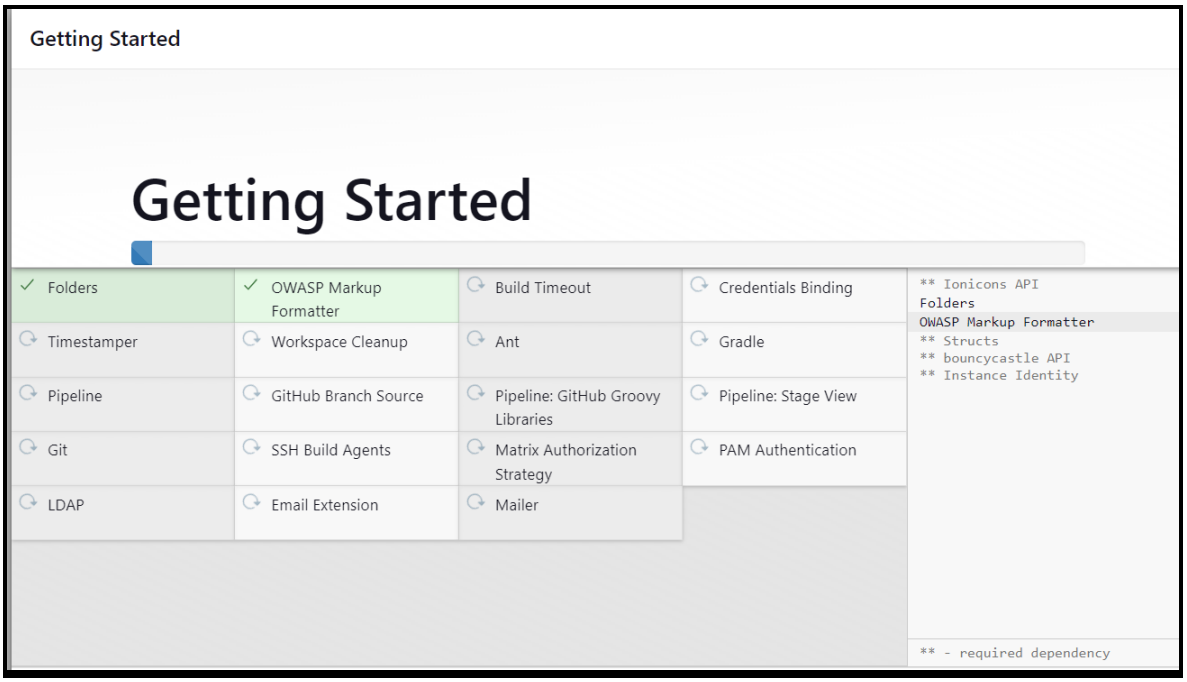
DevSecOps Final Project

Access jenkins through the web and configure it(Install suggested plugins, create user, etc…)

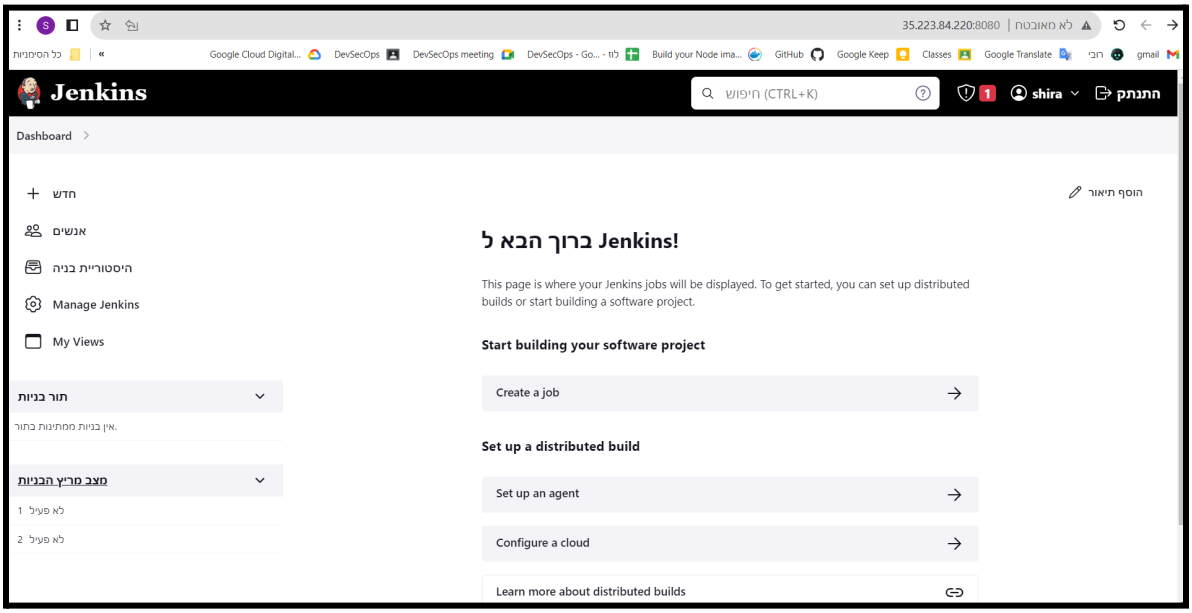
Now the jenkins server is up and running in the address http://35.222.0.182:8080/ where 35.222.0.182 is the compute engine external IP address, first I clicked on install suggested plugins:



DevSecOps Final Project



Now the jenkins server is available with the suggested plugins installed



DevSecOps Final Project

20

Create a CI/CD pipeline(jenkinsfile) that do the following:

- Build is triggered by checking if change(push) has been made every 10 seconds.
- Build the application
- Test - run django tests and check for 200(OK) response when trying to access the app.
If build succeeded:
- Push the image new version to artifact registry repository(The version must be the commit message)
- BONUS: Deploy the updated app to production cluster
If build failed:
- print “the pipeline failed :(“.
DevSecOps Final Project -
DevConnect
- Always Clean up all resources and workspace when you are done.

Now I want to create a pipeline project and connect it to the repo in github.
For that I first need to generate an ssh key for allowing this connection.

Reference to setup-ssh-between-jenkins-and-github:
https://levelup.gitconnected.com/setup-ssh-between-jenkins-and-github-e4d7d226b271

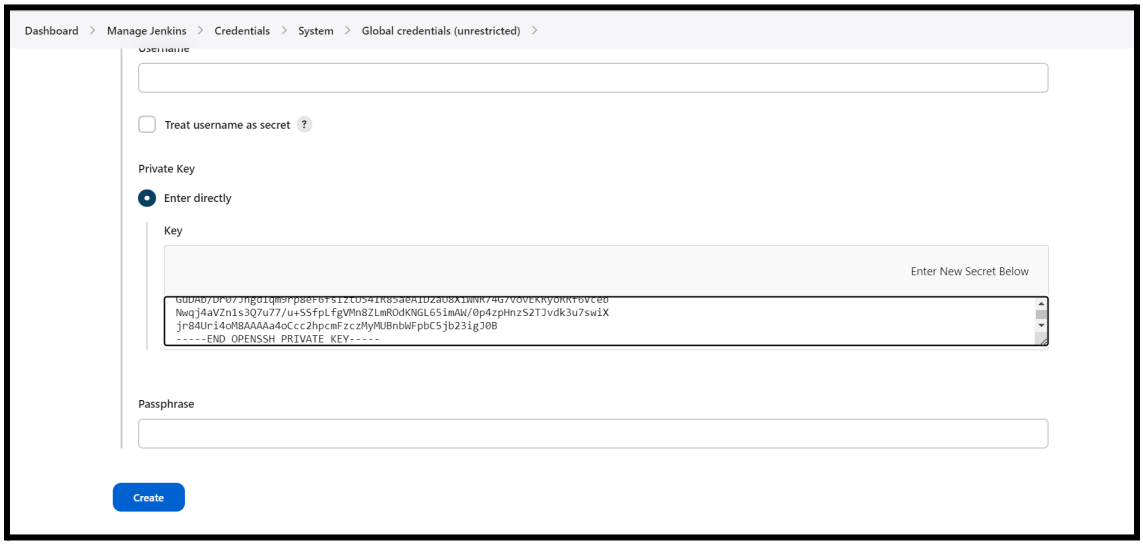
DevSecOps Final Project

In the compute engine ssh I get into the jenkins container:
docker exec -it <container_id> bash, in my case:
docker exec -it jenkins bash

Now generate the key:
ssh-keygen -t rsa -b 4096 -C "shirass321@gmail.com"

Enter the **public** key to github as a new ssh key:

Enter the **private** key as a new credential in the jenkins server:



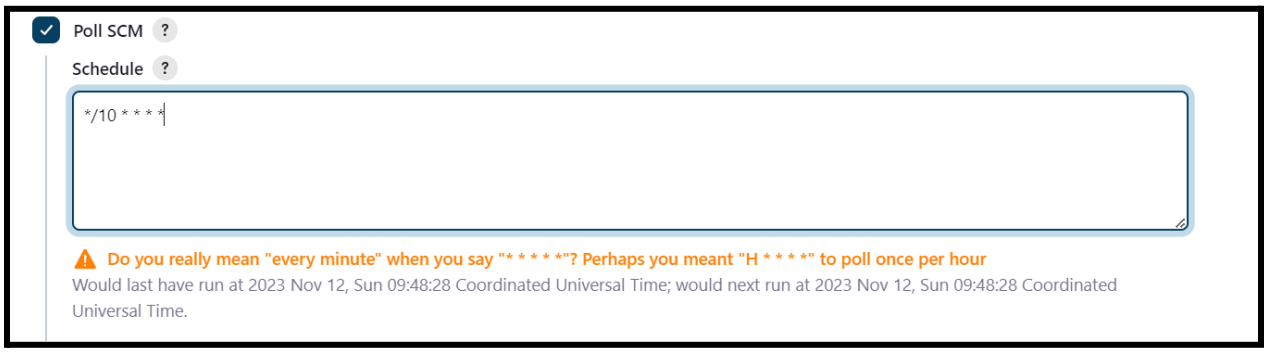
DevSecOps Final Project

After doing the above I ran inside the container :

`git ls-remote -h -- git@github.com:shirasss/final-project.git HEAD`
entered yes and the connection is complete and I can use the added credential with the connection to GitHub!

```
jenkins@59d5923565f4:/$ git ls-remote -h -- git@github.com:shirasss/final-project.git HEAD
The authenticity of host 'github.com (140.82.114.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
jenkins@59d5923565f4:/$
```

Then I created a pipeline project :
Build is triggered by checking if change(push) has been made every 10 seconds.



DevSecOps Final Project

My jenkinsfile:

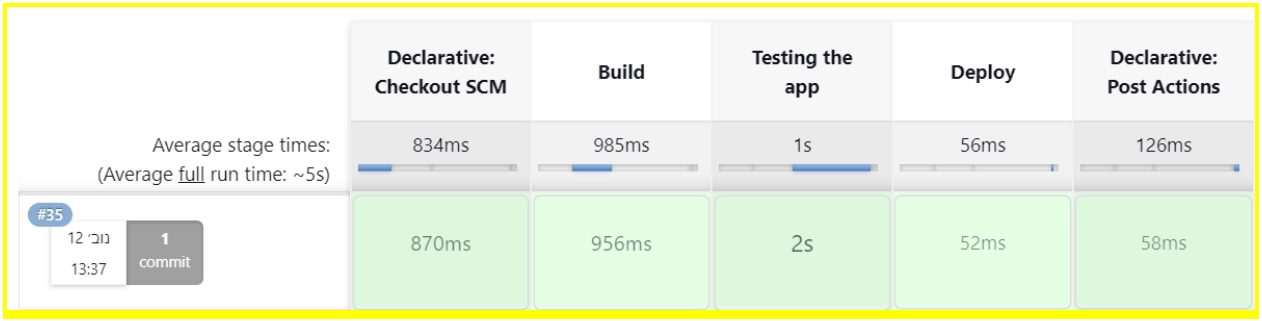
```
pipeline {
    agent any
    environment {
        dockerImageName = 'django_from_jenkins2'
        artifactRegistryImage =
"me-west1-docker.pkg.dev/devconnect-project/shira-shani-artifacts/devconne
ct"
        containerName = "django_container"
    }
    stages {
        stage('Build') {
            steps {
                echo 'Building the Docker image'
                dir('django_web_app') {
                    script {
                        // Extract the commit hash
                        def commitHash = sh(script: 'git rev-parse HEAD',
returnStdout: true).trim()
                        sh "docker build -t $dockerImageName:$commitHash
."
                        sh "docker run --name $containerName -d -p
5050:8000 $dockerImageName:$commitHash"
                    }
                }
            }
        }
        stage('Testing the app') {
            steps {
                echo 'Running Django tests'
                script {
                    // Run the Django tests
                        sh "docker exec $containerName python manage.py
test"
                }
            }
        }
```

```
        stage('Push to Artifact Registry') {
            steps {
                echo 'Pushing the Docker image to Artifact Registry'
                script {
                    // Use the commit hash as the version tag
                    def commitHash = sh(script: 'git rev-parse HEAD',
returnStdout: true).trim()

                    sh "docker tag $dockerImageName:$commitHash
$artifactRegistryImage:$commitHash"
                    sh "docker push $artifactRegistryImage:$commitHash"
                }
            }
        }

        stage('Deploy') {
            steps {
                echo 'Deploying...'
                // Add deployment steps here
            }
        }
    }
    post {
        success {
            echo 'Pipeline succeeded!'
        }
        failure {
            echo 'Pipeline failed!'
        }
        always {
            sh "docker stop $containerName || true"
            sh "docker rm $containerName || true"
        }
    }
}
```

DevSecOps Final Project

25

The build without the push to artifact stage:



	Declarative: Checkout SCM	Build	Testing the app	Deploy	Declarative: Post Actions
Average stage times: (Average full run time: ~5s)	834ms	985ms	1s	56ms	126ms
#35 12 נוב׳ 13:37 — 1 commit	870ms	956ms	2s	52ms	58ms

And after adding the push to artifact stage there was a permission denied to the artifact error so I ran inside the compute engine the command:

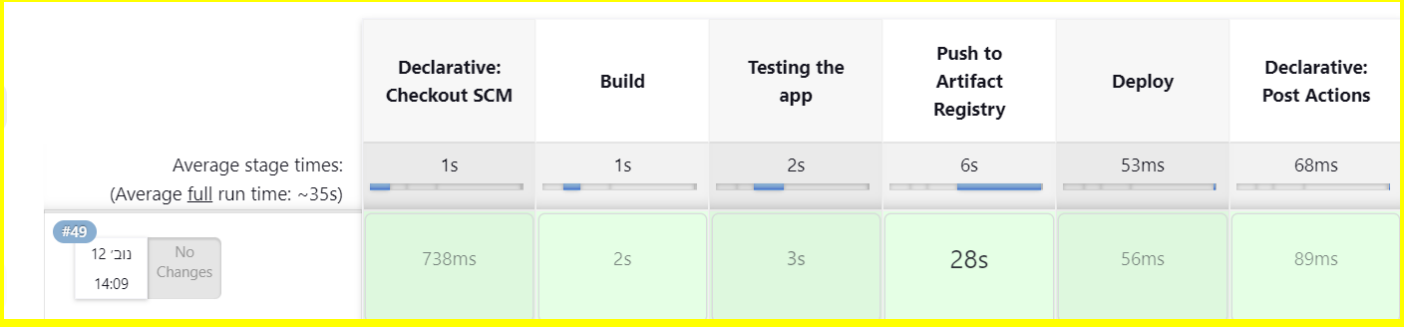
gcloud auth login to log in as shira.shani@grunitech.com

```
shira_shani@shira-shani-jenkins:~$ gcloud auth login

You are running on a Google Compute Engine virtual machine.
It is recommended that you use service accounts for authentication.

You can run:

  $ gcloud config set account `ACCOUNT`

to switch accounts if necessary.

Your credentials may be visible to others with access to this
virtual machine. Are you sure you want to authenticate with
your personal account?

Do you want to continue (Y/n)?  y

Go to the following link in your browser:

    https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.googleusercontent.com&redirect_uri=https%3A%2F%2Fsdk.cloud.google.com%2Fauthcode.html&sc
ope=openid+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.adm
in+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fsqlservice.login+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts.reauth&state=qqtE
Zhio7RSQmefSbhUZgF7BYEV6E7&prompt=consent&access_type=offline&code_challenge=wlOXg-sqsZgTyxybYuNHdzfEWYxeKfzV9jRCzmNANu0&code_challenge_method=S256

Enter authorization code: 4/0AfJohXn9oBRJdFN8txYPny9rlMTg4J1WU5OV5HFysDuGKxG7TM7Jp3c-wbgcd_PvV2PvsA

You are now logged in as [shira.shani@grunitech.com].
Your current project is [devconnect-project].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID
```

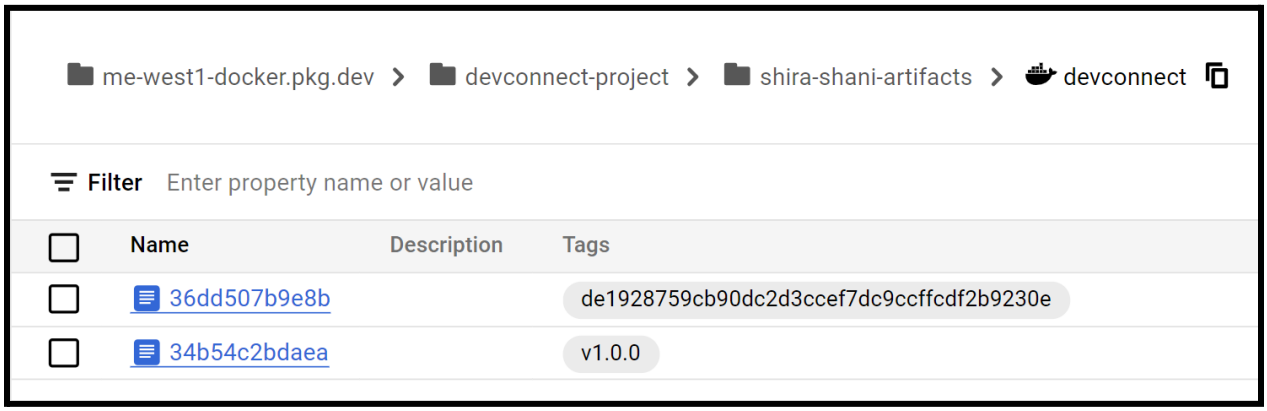
And also ran **gcloud auth configure-docker me-west1-docker.pkg.dev** to authenticate to the Google Container Registry (GCR) in the us-west1 region

DevSecOps Final Project

After doing the above the push to artifact registry stage worked and the image was pushed successfully!!!!👏👏👏



After the above built : in the artifact registry devconnect image the new tag is pushed:



And the tag is the commit hash.

Whenever there is a new push the the github repository the pipeline starts the build and the new version image is being built, tested and pushed to the artifact registry

DevSecOps Final Project

27

Sketch an architecture of the DevConnect project using google

architecture tool: