DevSecOps Final Project

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

Let’s get started!!

Dockerization:

Dockerize the application

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

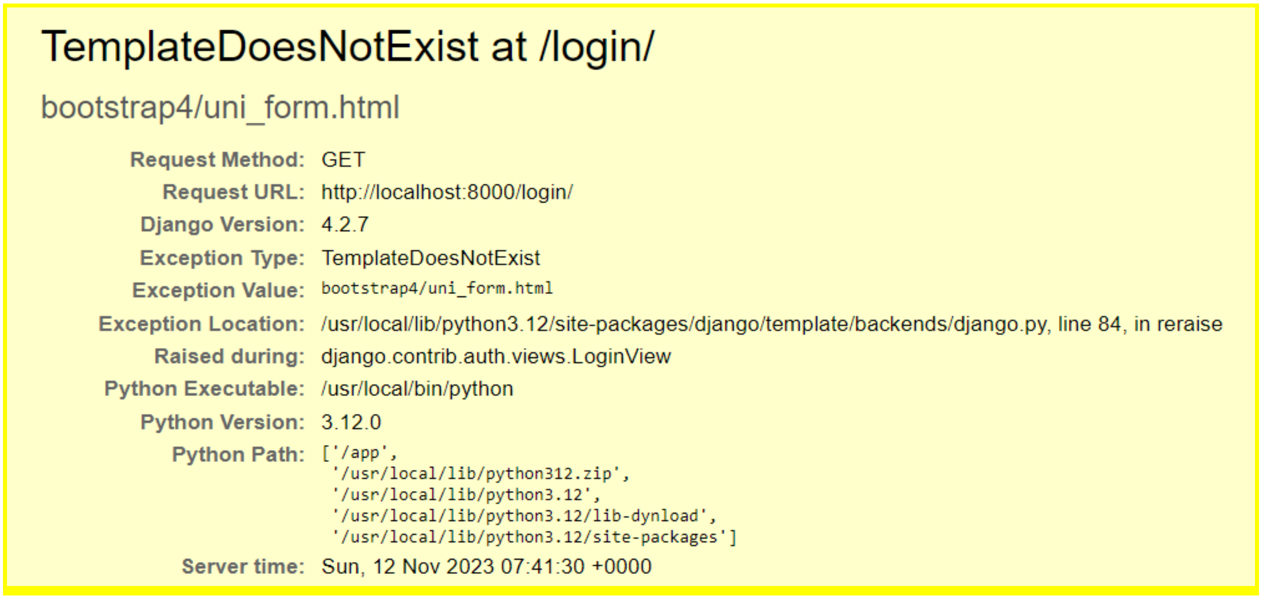
The dockerfile:

```
FROM python:latest

# Set the working directory to /app
WORKDIR /app

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

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

# 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==2.0.0
```

DevSecOps Final Project



TemplateDoesNotExist at /login/

bootstrap4/uni_form.html

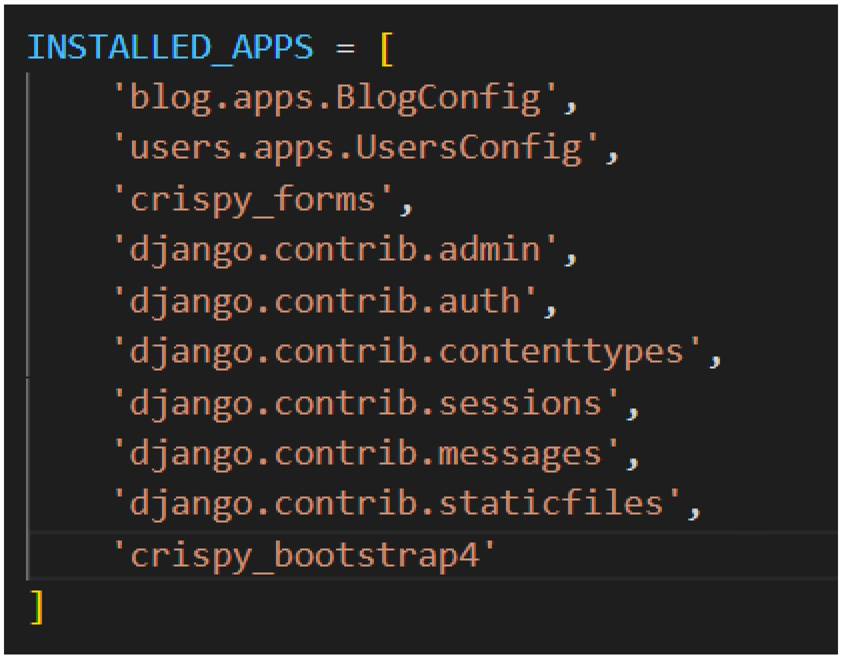
```
Request Method:      GET
Request URL:         http://localhost:8000/login/
Django Version:      4.2.7
Exception Type:      TemplateDoesNotExist
Exception Value:     bootstrap4/uni_form.html
Exception Location:  /usr/local/lib/python3.12/site-packages/django/template/backends/django.py, line 84, in 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`



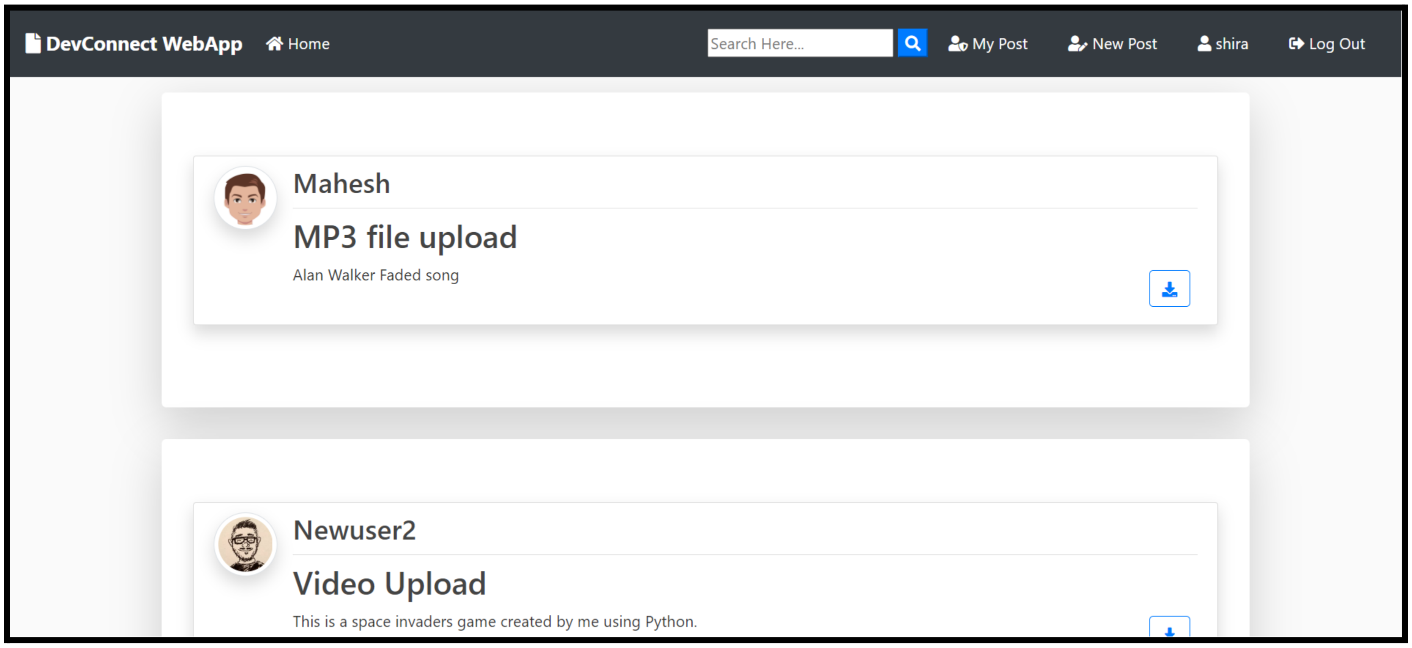
```
INSTALLED_APPS = [
    'blog.apps.BlogConfig',
    'users.apps.UsersConfig',
    'crispy_forms',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'crispy_bootstrap4'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

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

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/



Mahesh

MP3 file upload

Alan Walker Faded song

Newuser2

Video Upload

This is a space invaders game created by me using Python.

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."
```

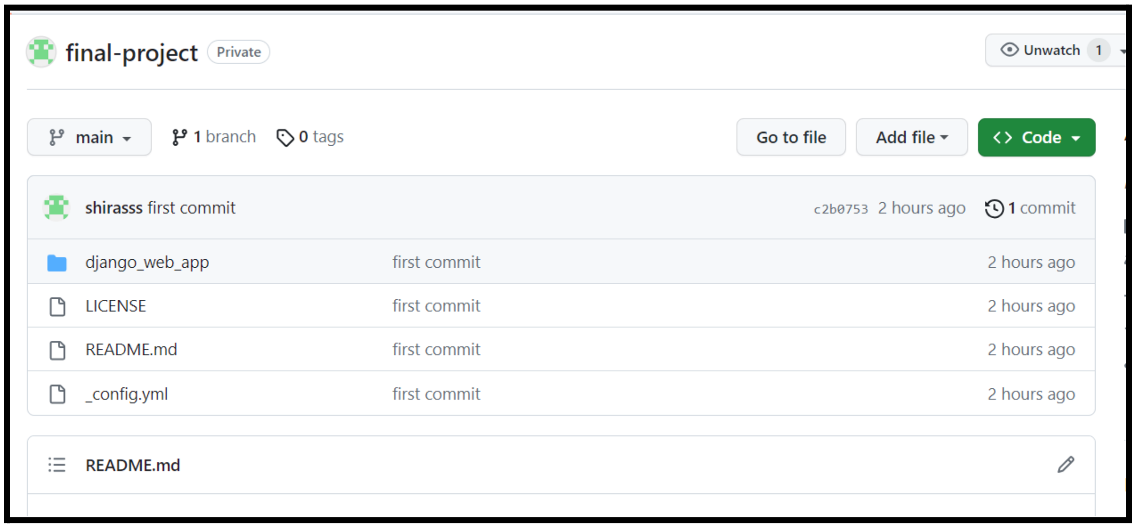
Run the application with volume and make it persistent - check by signing up to the app, delete the container and then start the container and log in without signing up,

I added a docker-compose.yaml file:

```
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.


final-project Private

main · 1 branch · 0 tags

shirasss first commit · c2b0753 2 hours ago · 1 commit

django_web_app	first commit	2 hours ago
LICENSE	first commit	2 hours ago
README.md	first commit	2 hours ago
_config.yml	first commit	2 hours ago

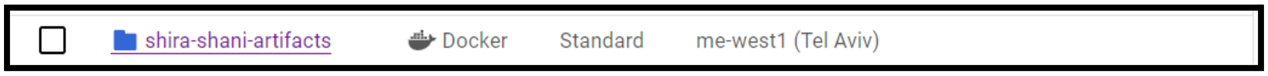
README.md

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.

DevSecOps Final Project

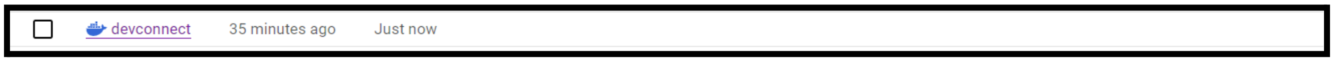


shira-shani-artifacts Docker Standard me-west1 (Tel Aviv)

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

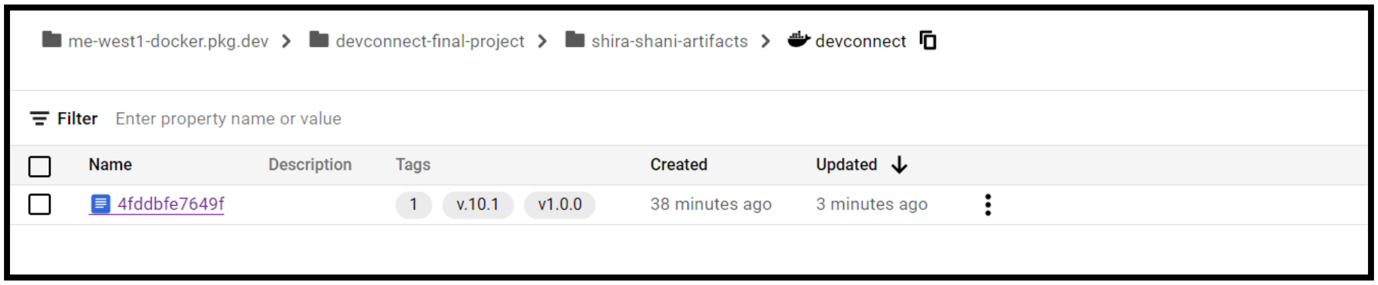
```
image_name=devconnect
echo "enter the version"
read version
local_image_name=django_web_app-web:latest
gcloud auth login
artifact_registry_image=me-west1-docker.pkg.dev/devconnect-final-project/
shira-shani-artifacts/${image_name}:${version}
docker tag ${local_image_name} ${artifact_registry_image}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



devconnect 35 minutes ago Just now

And inside that I have the image tags I pushed:



me-west1-docker.pkg.dev > devconnect-final-project > shira-shani-artifacts > devconnect

Filter Enter property name or value

Name	Description	Tags	Created	Updated
4fddbfe7649f		1 v.10.1 v1.0.0	38 minutes ago	3 minutes ago

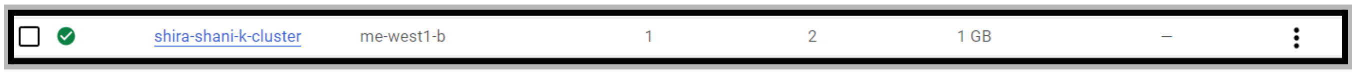
Deploy a zonal GKE standart cluster called <your-name>-

cluster with the following specifications:
IF NOT MENTIONED LEAVE AT DEFAULT
Zone: me-west1-a/b/c
Node pool 1:

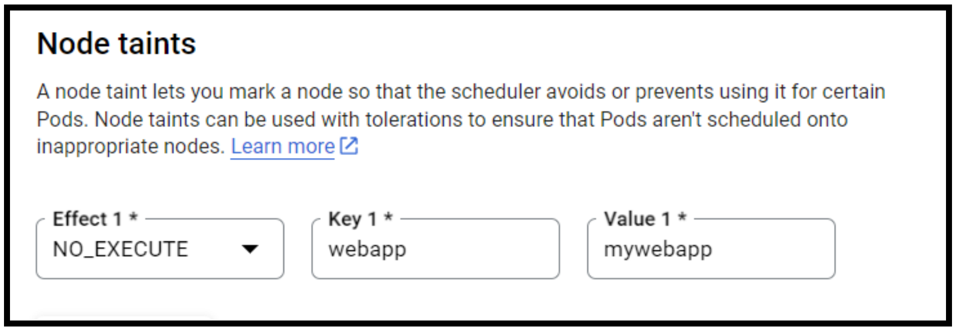
DevSecOps Final Project

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
```



shira-shani-k-cluster	me-west1-b	1	2	1 GB	–	⋮



Node taints

A node taint lets you mark a node so that the scheduler avoids or prevents using it for certain Pods. Node taints can be used with tolerations to ensure that Pods aren't scheduled onto inappropriate nodes. Learn more

Effect 1 *: NO_EXECUTE
Key 1 *: webapp
Value 1 *: mywebapp

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

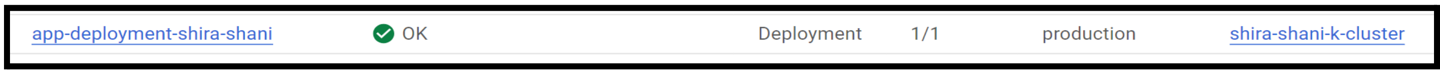
Create the namespace production :

```
kubectl create namespace production
```

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

```yaml
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"
```

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



```
app-deployment-shira-shani   ✓ OK   Deployment   1/1   production   shira-shani-k-cluster
```

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

DevSecOps Final Project

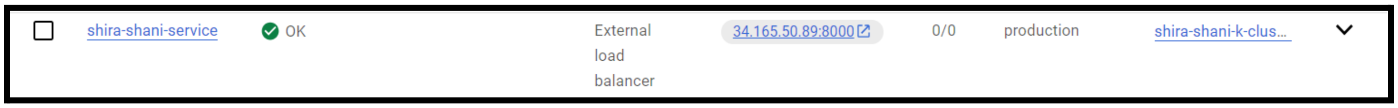
Expose it using load balancer service and access it through a browser:

I created a service.yaml file:

```
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

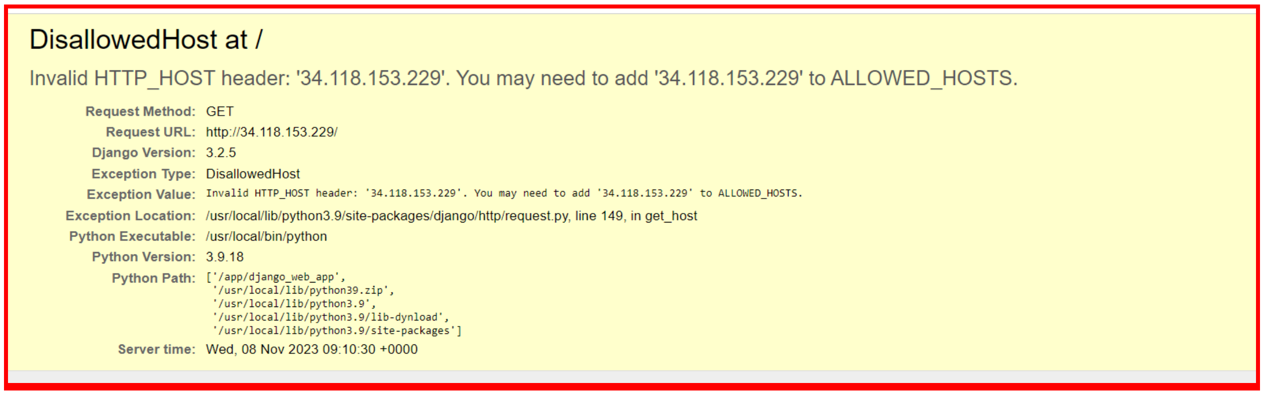


☐	shira-shani-service	✓ OK	External load balancer	34.165.50.89:8000	0/0	production	shira-shani-k-clus...

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

The bug that I need to fix:

DevSecOps Final Project

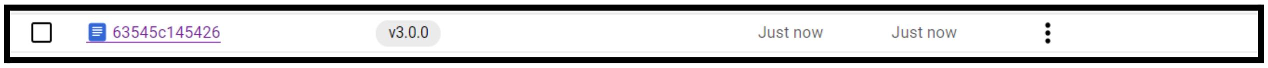


```
DisallowedHost at /
Invalid HTTP_HOST header: '34.118.153.229'. You may need to add '34.118.153.229' to ALLOWED_HOSTS.

     Request Method:  GET
        Request URL:  http://34.118.153.229/
     Django Version:  3.2.5
     Exception Type:  DisallowedHost
    Exception Value:  Invalid HTTP_HOST header: '34.118.153.229'. You may need to add '34.118.153.229' to ALLOWED_HOSTS.
 Exception Location:  /usr/local/lib/python3.9/site-packages/django/http/request.py, line 149, in get_host
  Python Executable:  /usr/local/bin/python
     Python Version:  3.9.18
        Python Path:  ['/app/django_web_app',
                       '/usr/local/lib/python39.zip',
                       '/usr/local/lib/python3.9',
                       '/usr/local/lib/python3.9/lib-dynload',
                       '/usr/local/lib/python3.9/site-packages']
        Server time:  Wed, 08 Nov 2023 09:10:30 +0000
```

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

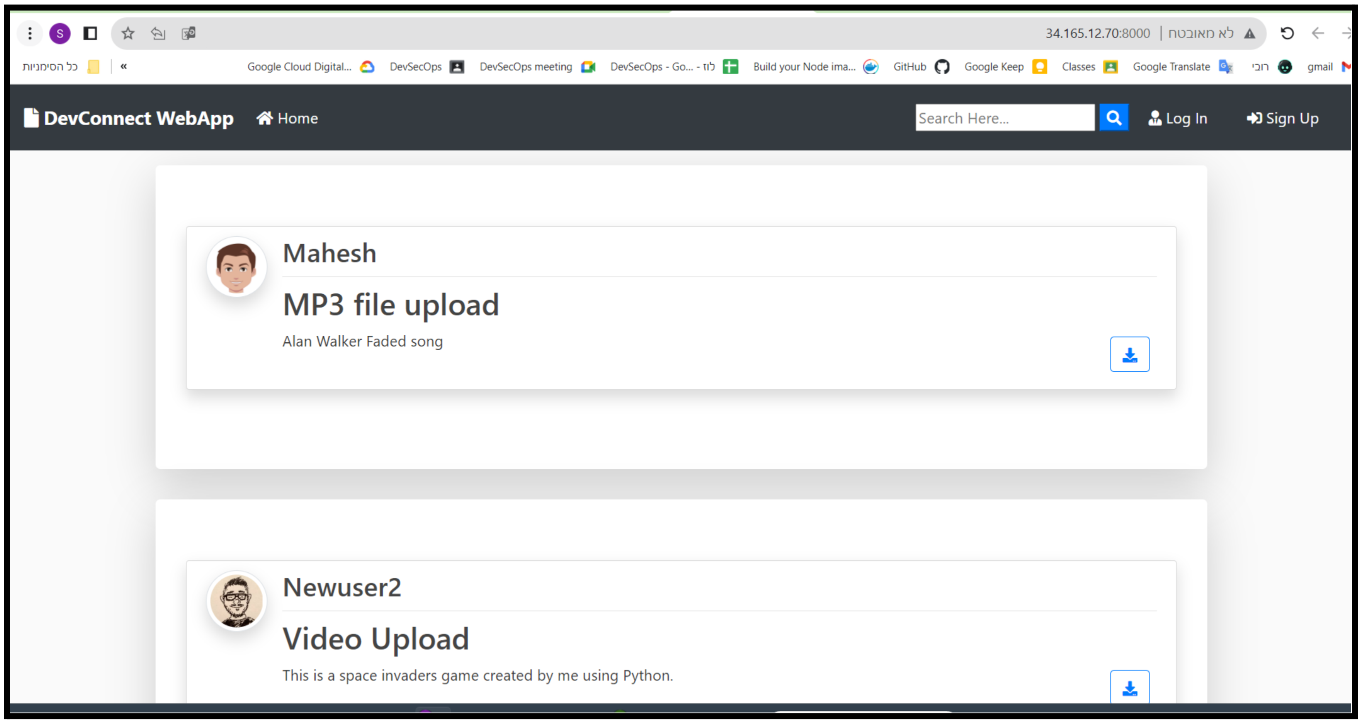


```
☐  63545c145426        v3.0.0        Just now        Just now     ⋮
```

And updated the deployment image version to the fixed one

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

DevSecOps Final Project



Mahesh
MP3 file upload
Alan Walker Faded song

Newuser2
Video Upload
This is a space invaders game created by me using Python.

CI/CD:

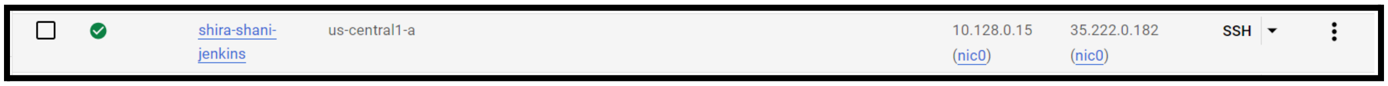
Create a Compute engine instance with the following specs:

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
```



☐	✓	shira-shani-jenkins	us-central1-a	10.128.0.15 (nic0)	35.222.0.182 (nic0)	SSH

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
```

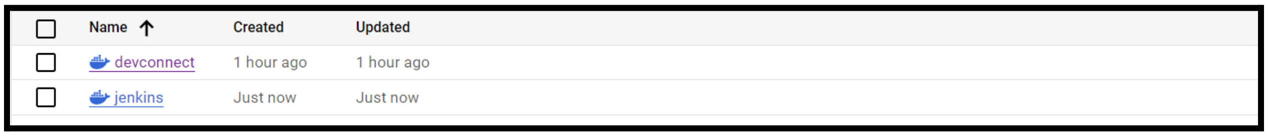
I ran `docker build -t jenkins_image.`

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:



	Name ↑	Created	Updated
☐	devconnect	1 hour ago	1 hour ago
☐	jenkins	Just now	Just now

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)
<u>And the problem was solved!</u>

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/jen
kins:v1.0.0
```

I created volume named jenkins_home by the command:

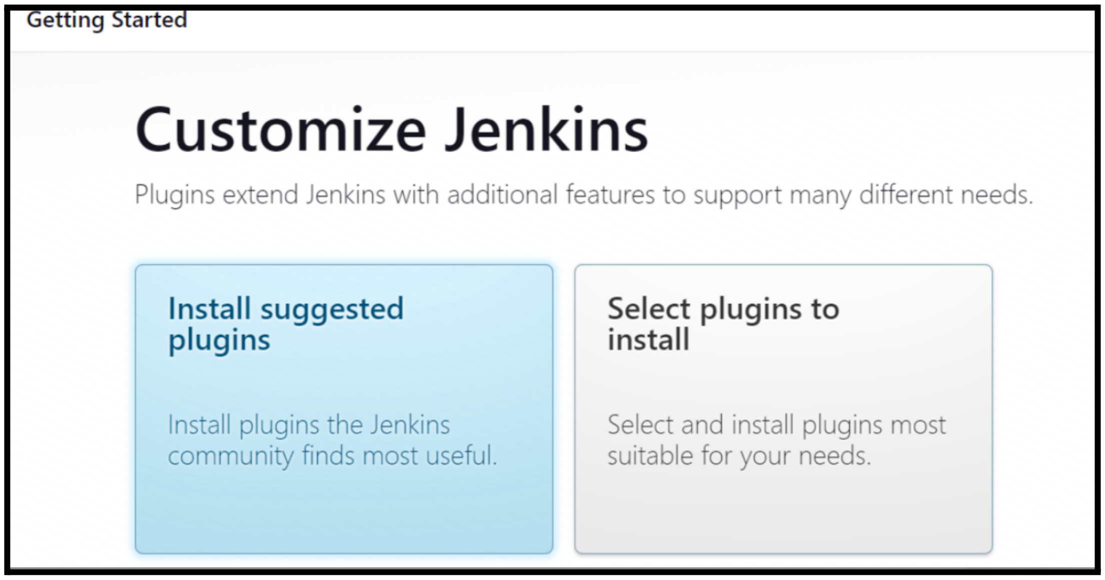
```
docker volume create jenkins_home
```

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-artifact
s/devconnect:v1.0.0
```

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

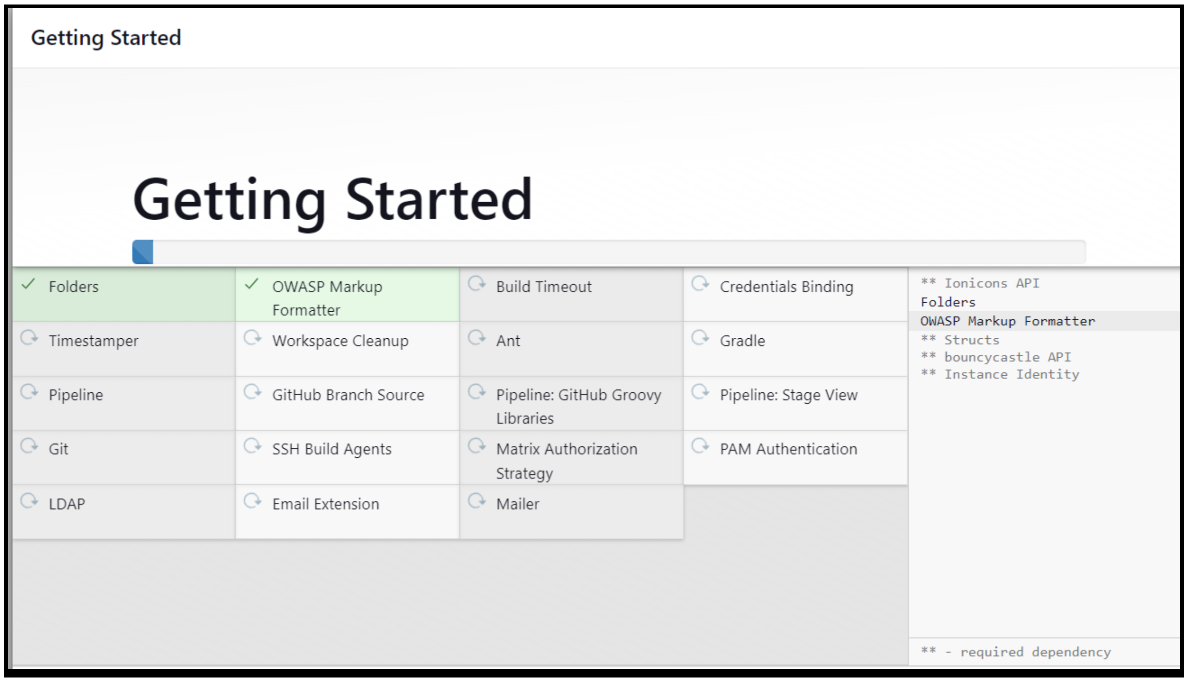
Now the jenkins server is up and running in the address http://35.222.0.182:8080/ where 35.222.0.182 is the compute engine external IP address, first I clicked on install suggested plugins:


Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.


Getting Started

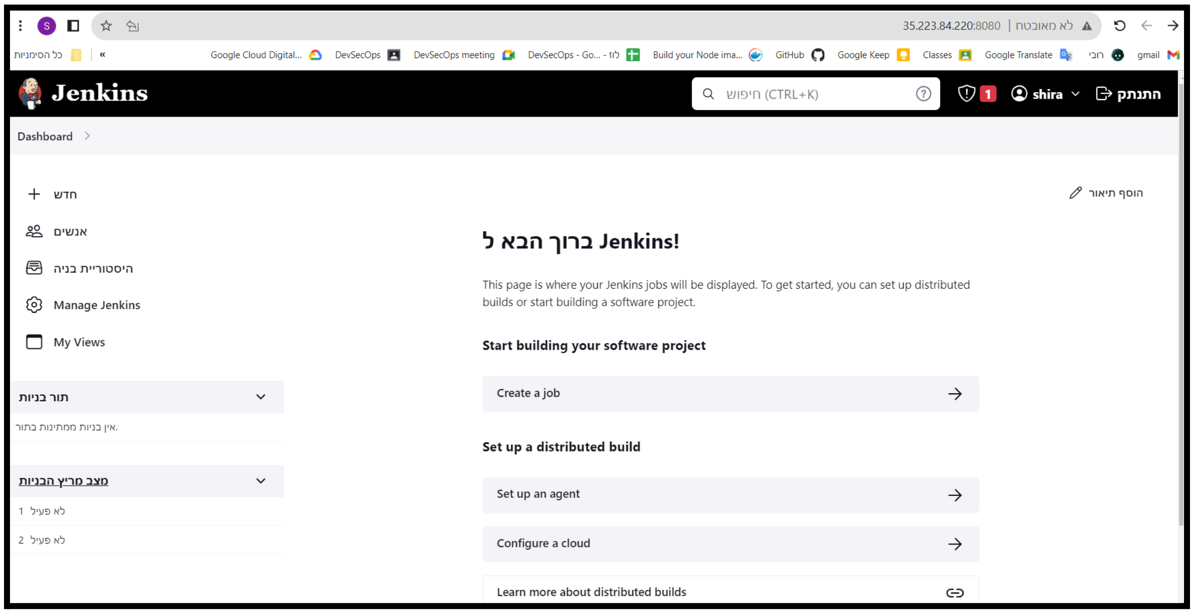
Getting Started

✓ Folders	✓ OWASP Markup Formatter	Build Timeout	Credentials Binding
Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication
LDAP	Email Extension	Mailer	

```
** Ionicons API
Folders
OWASP Markup Formatter
** Structs
** bouncycastle API
** Instance Identity
```

```
** - required dependency
```

Now the jenkins server is available with the suggested plugins installed



ברוך הבא ל Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Set up a distributed build

אין בניות ממתינות בתור.

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

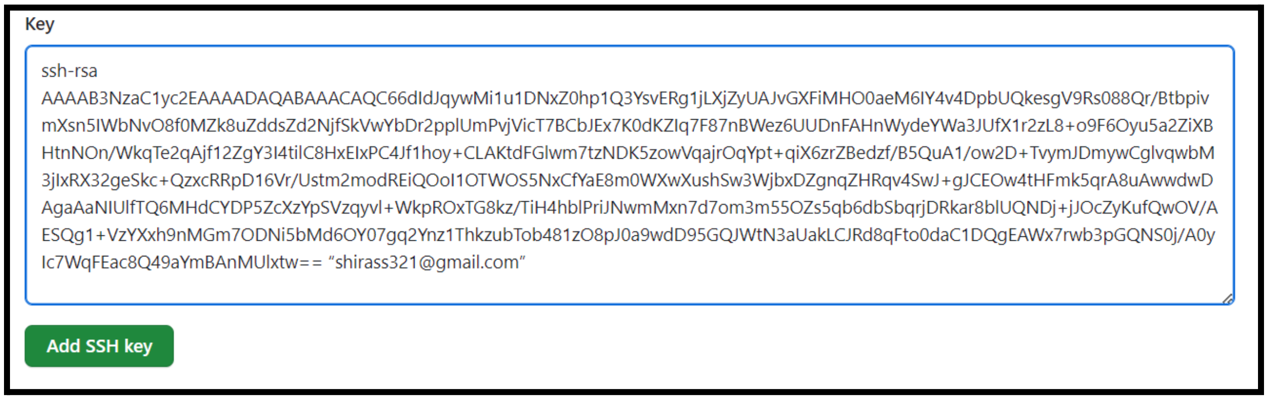
In the compute engine ssh I get into the jenkins container:
docker exec -it <container_id> bash, in my case:

```
docker exec -it jenkins bash
```

Now generate the key:

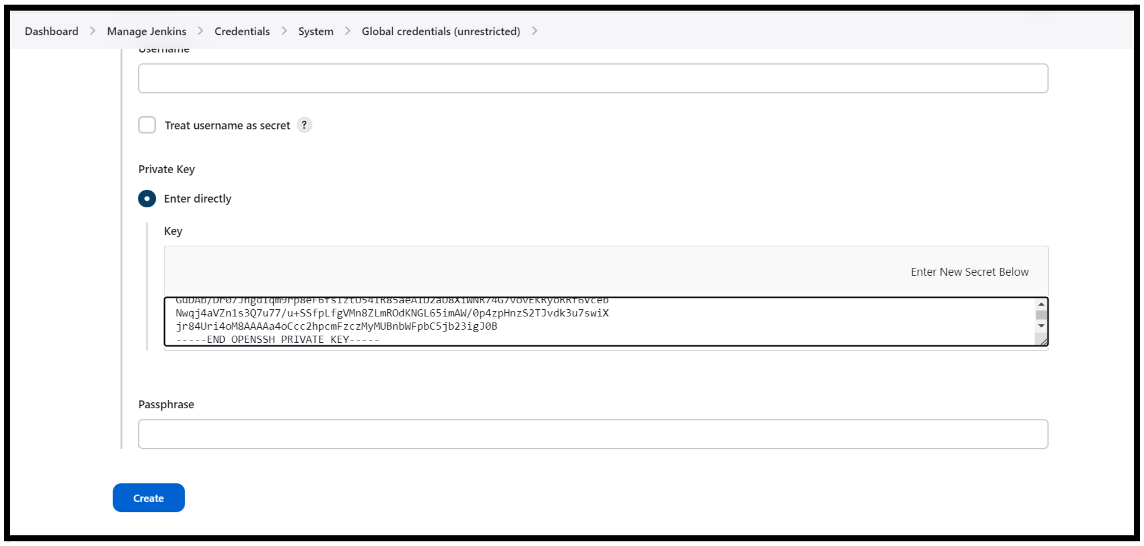
```
ssh-keygen -t rsa -b 4096 -C "shirass321@gmail.com"
```

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


Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQC66dIdJqywMi1u1DNxZ0hp1Q3YsvERg1jLXjZyUAJvGXFiMHO0aeM6IY4v4DpbUQkesgV9Rs088Qr/Btbpiv
mXsn5IWbNvO8f0MZk8uZddsZd2NjfSkVwYbDr2pplUmPvjVicT7BCbJEx7K0dKZIq7F87nBWez6UUDnFAHnWydeYWa3JUfX1r2zL8+o9F6Oyu5a2ZiXB
HtnNOn/WkqTe2qAjf12ZgY3I4tilC8HxElxPC4Jf1hoy+CLAKtdFGlwm7tzNDK5zowVqajrOqYpt+qiX6zrZBedzf/B5QuA1/ow2D+TvymJDmywCglvqwbM
3jIxRX32geSkc+QzxcRRpD16Vr/Ustm2modREiQOoI1OTWOS5NxCfYaE8m0WXwXushSw3WjbxDZgnqZHRqv4SwJ+gJCEOw4tHFmk5qrA8uAwwdwD
AgaAaNIUlfTQ6MHdCYDP5ZcXzYpSVzqyvl+WkpROxTG8kz/TiH4hblPriJNwmMxn7d7om3m55OZs5qb6dbSbqrjDRkar8blUQNDj+jJOcZyKufQwOV/A
ESQg1+VzYXxh9nMGm7ODNi5bMd6OY07gq2Ynz1ThkzubTob481zO8pJ0a9wdD95GQJWtN3aUakLCJRd8qFto0daC1DQgEAWx7rwb3pGQNS0j/A0y
Ic7WqFEac8Q49aYmBAnMUlxtw== "shirass321@gmail.com"
```

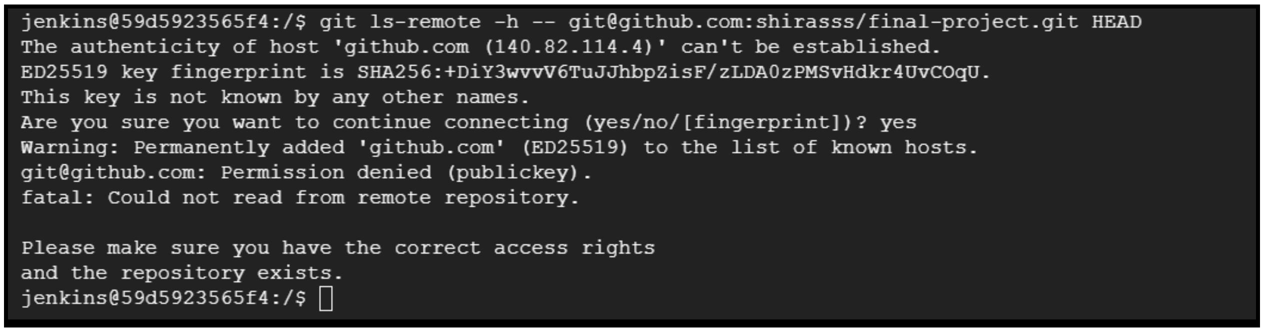
Enter the **private** key as a new credential in the jenkins server:



```
Nwqj4aVZn1s3Q7u77/u+SSfpLfgVMn8ZLmROdKNGL65imAW/0p4zpHnzS2TJvdk3u7swiX
jr84Uri4oM8AAAAa4oCcc2hpcmFzczMyMUBnbWFpbC5jb23igJ0B
-----END OPENSSH PRIVATE KEY-----
```

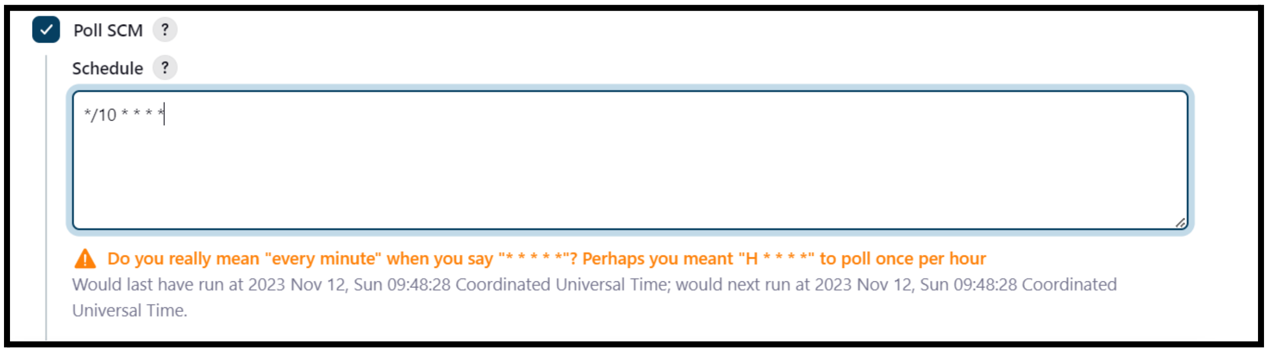
After doing the above I ran inside the container :

git ls-remote -h -- git@github.com:shirasss/final-project.git HEAD
entered yes and the connection is complete!



```
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.



Poll SCM

Schedule

```
*/10 * * * *
```

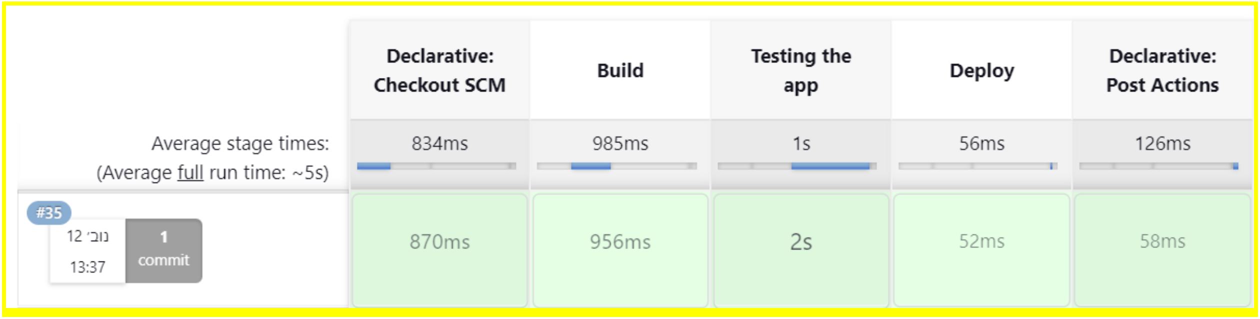
Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour
Would last have run at 2023 Nov 12, Sun 09:48:28 Coordinated Universal Time; would next run at 2023 Nov 12, Sun 09:48:28 Coordinated Universal Time.

My jenkinsfile:

```
pipeline {
    agent any
    environment {
        dockerImageName = 'django_from_jenkins'
        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 
."
                    }
                }
            }
        }
        stage('Testing the app') {
            steps {
                echo 'Running Django tests'
                script {
                    def commitHash = sh(script: 'git rev-parse HEAD', 
returnStdout: true).trim()
                    sh "docker stop $containerName || true"
                    sh "docker rm $containerName || true"
                    sh "docker run --name $containerName -d -p 5050:8000 
$dockerImageName:$commitHash"
                    // Run the Django tests
                    dir('django_web_app') {
                        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!'
        }
    }
}
```

The build without the push to artifact stage:

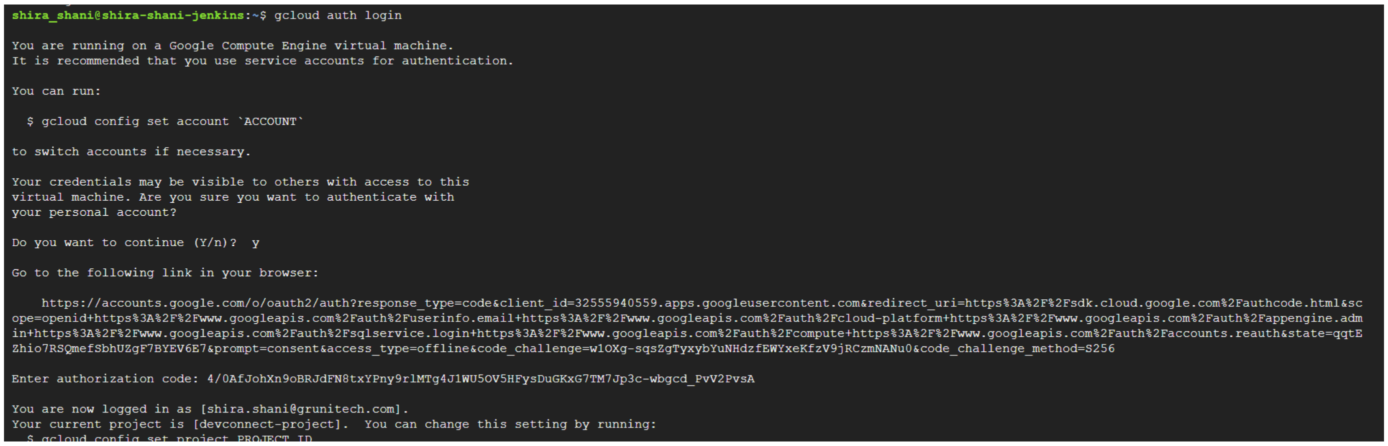


	Declarative: Checkout SCM	Build	Testing the app	Deploy	Declarative: Post Actions
Average stage times: (Average full run time: ~5s)	834ms	985ms	1s	56ms	126ms
#35 12 נוב׳ 13:37 1 commit	870ms	956ms	2s	52ms	58ms

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:

DevSecOps Final Project

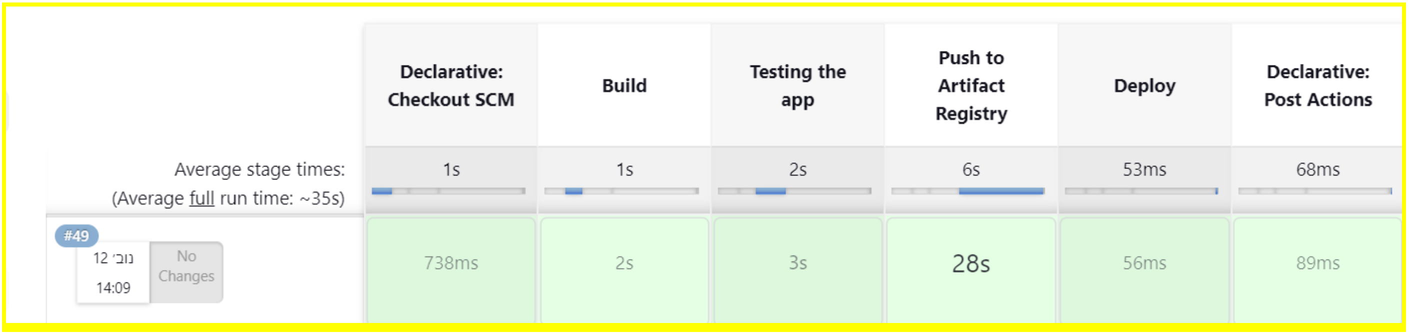
gcloud auth login to log in as shira.shani@grunitech.com



```
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=w1OXg-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

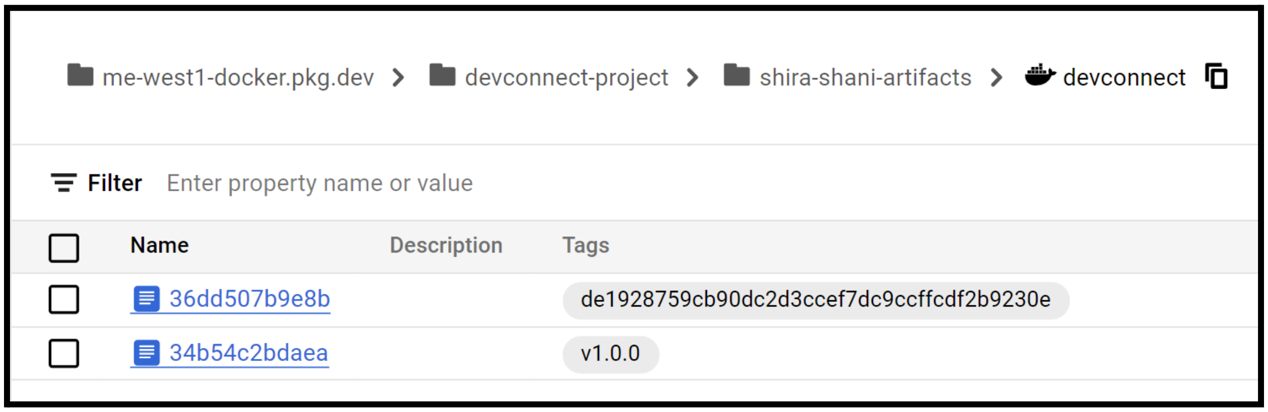
After doing the above the push to artifact registry stage worked and the image was pushed successfully!!!!👏👏👏



	Declarative: Checkout SCM	Build	Testing the app	Push to Artifact Registry	Deploy	Declarative: Post Actions
Average stage times: (Average full run time: ~35s)	1s	1s	2s	6s	53ms	68ms
#49 12 נוב׳ 14:09 No Changes	738ms	2s	3s	28s	56ms	89ms

DevSecOps Final Project

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



me-west1-docker.pkg.dev > devconnect-project > shira-shani-artifacts > devconnect

Filter Enter property name or value

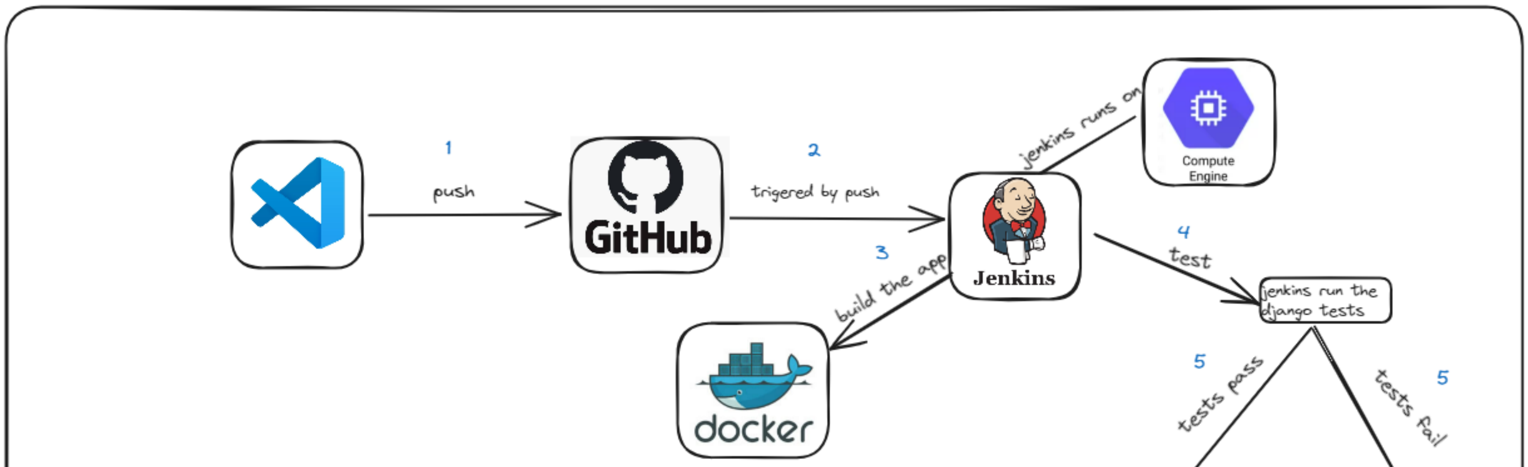
Name	Description	Tags
36dd507b9e8b		de1928759cb90dc2d3ccef7dc9ccffcdf2b9230e
34b54c2bdaea		v1.0.0

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

Sketch an architecture of the DevConnect project using google

architecture tool:


1
push
GitHub
2
trigered by push
jenkins runs on
Compute Engine
Jenkins
3
build the app
docker
4
test
jenkins run the django tests
5
tests pass
5
tests fail