DevSecOps solution

In this project we will combine all of the knowledge you’ve learned to create a full automation and deployment for an app!

workplan:

# Dockerize the app -

* For the packages, use all the latest versions.
* Run the application and make sure everything is working.

Create requirements.txt with latest versions:

Django>=4.2

django-crispy-forms>=2.1

Pillow>=10.1.0

crispy-bootstrap4==2023.1

Create Dockerfile

FROM python:3.9.18-slim as builder

WORKDIR /app

COPY ./django\_web\_app/requirements.txt .

RUN pip install -r requirements.txt

FROM python:3.9.18-slim as run

COPY --from=builder /usr/local/lib/python3.9/site-packages /usr/local/lib/python3.9/site-packages

WORKDIR /django\_web\_app

COPY ./django\_web\_app .

RUN python manage.py makemigrations

RUN python manage.py migrate

ENTRYPOINT [ "python", "manage.py" , "runserver", "0.0.0.0:8000" ]

:

**First problem:**  Add to settings.py:

DEFAULT\_AUTO\_FIELD = 'django.db.models.AutoField'

<https://stackoverflow.com/questions/66971594/auto-create-primary-key-used-when-not-defining-a-primary-key-type-warning-in-dja>

Warning:

learning\_logs.Entry: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.models.AutoField'.

HINT: Configure the DEFAULT\_AUTO\_FIELD setting or the LearningLogsConfig.default\_auto\_field attribute to point to a subclass of AutoField, e.g. 'django.db.models.BigAutoField'.

10%

**Second problem:**runserver on 0.0.0.0:8000 instead of localhost:8000 or 127.0.0.1:8000

<https://stackoverflow.com/questions/61863806/stuck-in-watching-for-file-changes-with-statreloader>

10%

**Third problem:** change base.html first line to {% load static %} instead of {% load staticfiles %}

<https://stackoverflow.com/questions/55929472/django-templatesyntaxerror-staticfiles-is-not-a-registered-tag-library>

40%

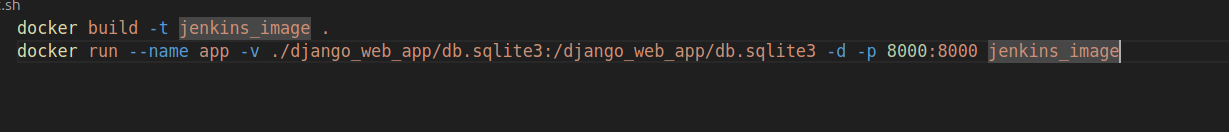
**Forth problem:** **Error:** form not working - You will need to pip install crispy-bootstrap4 and add crispy\_bootstrap4 to your list of INSTALLED\_APPS

<https://stackoverflow.com/questions/75495403/django-returns-templatedoesnotexist-when-using-crispy-forms>

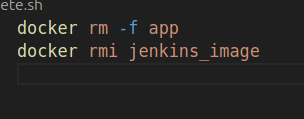
40%

* Create init.sh and delete.sh for automation.
* 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.

**Init.sh:**



**delete.sh:**



# Deployment -

* Create an artifact repository called <your-name>-artifacts

and automate a deployment of the web app image to it.

Create a repo

configure docker

**deploy.sh:**

#!/bin/bash

deploy(){

VERSION=$1

# Error handling must get 2 inputs from user

if [ $# -lt 1 ]

then

echo "USAGE: $0 '<version>' "

exit 1

fi

docker build -t me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/app:$VERSION .

# Error handling: Checks if build is completed succefuly

if [ $? -eq 0 ]

then

echo

echo "Build Succeeded/Exists"

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

docker push me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/app:$VERSION

else

echo "Error: Build failed, Fix Dockerfile and try again"

fi

}

deploy $1

* 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:**

1. name - devconnect-app
2. nodes - 1
3. machine type - e2-micro (2 vCPU, 1 core, 1 GB memory)
4. service account - assign DevOps-sa
5. boot disk - 12 gb.
6. node taints - NO\_EXECUTE, key=webapp, value=mywebapp.(read and understand taints)

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

kubectl create namespace production

kubectl create deployment devconnect-app --image me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/app:1.0.0 --namespace production

* Add the appropriate tolerations to deployment.yaml file.

<https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/>

kubectl edit deployment/django-app

Add the following:  
 tolerations:

- effect: NoExecute

key: webapp

operator: Equal

value: mywebapp

* Expose it using load balancer service and access it through a browser.

kubectl expose deployment devconnect-app --type="LoadBalancer" --port=8000 --namespace=production

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

kubectl edit deployment/devconnect-app & change version manually.

OR

kubectl set image deployment/devconnect-app app=me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/app:1.0.1 --namespace=production.

* Rollout the new version deployment.

**kubectl rollout status deployment/devconnect-app --namespace=production**

# CI/CD -

* Create a Compute engine instance with the following specs:

1. Name - <your-name>-jenkins.
2. Region - me-west1(Tel-Aviv){% load static %}
3. Machine type - e2-medium (2 vCPU, 1 core, 4 GB memory)
4. service account - assign DevOps-sa
5. boot disk - 20 gb.
6. FireWall - Allow HTTP traffic.
7. Automation - install docker engine.

**Docker automation:**

#!/bin/bash

sudo apt-get update -y

sudo apt-get install ca-certificates curl gnupg -y

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \

"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \

"$(. /etc/os-release && echo "$VERSION\_CODENAME")" stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update -y

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

sudo groupadd docker

sudo usermod -aG docker $USER

newgrp docker

* 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.

ssh and configure docker on the machine:

gcloud auth configure-docker \

me-west1-docker.pkg.dev

**deploy.sh:**

#!/bin/bash

choice\_func(){

read -r -n 1 -p "Enter Y|y or N|n: " choice

while [[ $choice != 1 && $choice != 0 ]]

do

if [[ $choice == [Yy] ]]; then

choice=0

elif [[ $choice == [Nn] ]]; then

choice=1

else

echo "Invalid input, Try again."

fi

done

}

deploy(){

VERSION=$1

# Error handling must get 2 inputs from user

if [ $# -lt 1 ]

then

echo "USAGE: $0 '<version>' "

exit 1

fi

choice=0

# Tag & Build thin.dockerfile if not exists

if docker images -a | grep -E "^me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/jenkins:$VERSION$" &> /dev/null

then

echo "Image me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/jenkins:$VERSION already exists."

echo

echo "Use the existing image - [N|n] or rebuild and delete the existing one - [Y|y]"

choice\_func

fi

if [ $choice = 0 ]

then

docker rmi -f me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/jenkins:$VERSION

docker build -t me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/jenkins:$VERSION .

fi

# Error handling: Checks if build is completed succefuly

if [ $? -eq 0 ]

then

echo

echo "Build Succeeded/Exists"

echo "Push to GCR?"

choice\_func

if [ $choice = 0 ]

then

docker push me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/jenkins:$VERSION

fi

gcloud compute ssh --zone "me-west1-a" "test-jenkins" --project "devconnect-final-project" --command "docker run -p 8080:8080 -v jenkins\_home:/var/jenkins\_home -v /var/run/docker.sock:/var/run/docker.sock -d --name jenkins me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/jenkins:$VERSION"

else

echo "Error: Build failed, Fix Dockerfile and try again"

fi

}

deploy $1

**Dockerfile:**

FROM jenkins/jenkins:2.414.3-jdk17

USER root

RUN apt-get update && apt-get install -y ca-certificates curl gnupg

RUN install -m 0755 -d /etc/apt/keyrings

RUN curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg

RUN chmod a+r /etc/apt/keyrings/docker.gpg

RUN echo \

"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \

"$(. /etc/os-release && echo "$VERSION\_CODENAME")" stable" | \

tee /etc/apt/sources.list.d/docker.list > /dev/null

RUN apt-get update && apt-get install -y docker-ce-cli

RUN groupadd docker

RUN touch /var/run/docker.sock

RUN chmod 666 /var/run/docker.sock

RUN usermod -aG docker jenkins

# Downloading gcloud package

RUN curl -O https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-cli-453.0.0-linux-x86\_64.tar.gz

RUN tar -xf google-cloud-cli-453.0.0-linux-x86\_64.tar.gz

RUN ./google-cloud-sdk/install.sh

RUN rm -rf google-cloud-cli-453.0.0-linux-x86\_64.tar.gz

USER jenkins

Make sure the container can use docker!

sudo chmod 666 /var/run/docker.sock

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

Create jenkins-github connection.

* Create a CI/CD pipeline(jenkinsfile) that do the following:
* Build is triggered by checking if change(push) has been made every 10 seconds.

poll scm \*/6 \* \* \* \*

* **BONUS:** Build is triggered everytime a commit is pushed.
* 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 :(“.
* Always Clean up all resources and workspace when you're done.

**deploy.sh:**

#!/bin/bash

deploy(){

VERSION=$1

# Error handling must get 2 inputs from user

if [ $# -lt 1 ]

then

echo "USAGE: $0 '<version>' "

exit 1

fi

docker build -t me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/app:$VERSION .

# Error handling: Checks if build is completed succefuly

if [ $? -eq 0 ]

then

echo

echo "Build Succeeded/Exists"

source '/google-cloud-sdk/path.bash.inc'

source '/google-cloud-sdk/completion.bash.inc'

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

docker push me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/app:$VERSION

else

echo "Error: Build failed, Fix Dockerfile and try again"

fi

}

deploy $1

**JenkinsFile:**

pipeline {

agent any

stages {

stage('Build') {

steps {

sh 'chmod +x init.sh'

sh './init.sh'

}

}

stage('Test') {

steps {

sh 'docker exec app python manage.py test'

sh 'sleep 4'

sh "curl 10.208.0.7:8000"

sh 'status=$?'

echo '$status'

}

}

stage('Deploy') {

steps {

echo 'Deploying....'

}

}

}

post {

success {

sh 'version=$(git log -n 1 --format="%s" HEAD)'

sh 'echo $(git log -n 1 --format="%s" HEAD)'

sh 'chmod +x deploy.sh'

sh './deploy.sh $(git log -n 1 --format="%s" HEAD)'

sh 'rm -rf \*'

}

failure {

echo 'The Pipeline failed :('

sh 'rm -rf \*'

}

always {

sh 'echo build ended, deleting all resources...'

sh 'chmod +x ./delete.sh'

sh './delete.sh'

}

}

}

Gcloud solution:

<https://stackoverflow.com/questions/31037279/gcloud-command-not-found-while-installing-google-cloud-sdk>

# Scoring:

מחושב לפי -

**חלק ראשון - דוקריזציה - 45% מהציון הסופי**

6 שאלות בחלק זה, על כל שאלה ניתן ניקוד שווה → 100/6 = 16.6 נק׳ לכל סעיף

***( ז״א שאם ירד 50 אחוז על הסעיף = ירדו 8.35 נקודות < לדוגמה על גרסאות לא נכונות >) וזה מהווה 45% מהציון הסופי → 8.35\*0.45 = מכאן שירד בפועל 3.75 נק׳ )***

**חלק שני - דיפלויימנט - 45% מהציון הסופי**

7.7 נק׳ לכל סעיף → בפועל מהציון הסופי - 3.4 נק׳ לסעיף

ציון בוחן - 10% מהציון הסופי

ביצוע Ci/Cd - 5 נק׳ בונוס לציון הסופי

עריכת קובץ וסידורו - 5 נק׳ לציון הסופי

**סה״כ ניתן לקבל 10 נק׳ בונוס**

## Dockerization

| Dockerize the app | 1.1  Create Dockerfile - best practice | 1.2  Multistaging and Chaching - saved packages | 1.3  Create a new volume | 1.4  T-shoot - 4 problems | 1.5  **init.sh** and **delete.sh** automation | 1.6  App is running |
| --- | --- | --- | --- | --- | --- | --- |
|  | use : COPY ./django\_web\_app/requirements.txt .  RUN pip install -r requirements.txt  instead of just :  RUN pip install -r requirements.txt | FROM python:3.9.18-**slim** as run  and  COPY --from=builder /usr/local/lib/**python3.9**/site-packages /usr/local/lib/python3.9/site-packages | VOLUME ./db.sqlite3  docker run…  -v ./db.sqlite3:/db.sqlite3 | 1.4  Use latest version of Django (4.2), django-crispy-forms (2.1),pillow (10.1.0)  with Use | docker build  docker run  docker rm  docker rmi  ניתן להוסיף נקודות על Error checking | הוכחה להפעלת האפליקציה בצורה לוקאלית |

## Deployment

| Deployment | 2.1  Create an artifact repository called <your-name>-artifacts | 2.2  Automate a deployment of the web app image to it  ( [deploy.sh](http://deploy.sh/) ) | 2.2.1  Automate a deployment of the web app image to it  ( [deploy.sh](http://deploy.sh/) ) | 2.3  Deploy a zonal GKE standart cluster called <your-name>-cluster with the following specifications  ( deploy.sh ) | 2.3.1  Taints Explained | 2.4  In a namespace called production, create 1 replica deployment to the app. |
| --- | --- | --- | --- | --- | --- | --- |
|  | gcloud **auth configure-docker** me-west1-docker.pkg.de  gcloud **auth login** | docker **build** -t me-west1-docker.pkg.dev/devconnect-final-project/test-**artifacts/app**:$VERSION | docker **push** me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/app:$VERSION  else  echo "Error: Build failed, Fix Dockerfile and try again"  fi  }  deploy $1 | **Zone:** me-west1-a/b/c  **Node pool 1**:  **name** - devconnect-app  **nodes** - 1  **machine type** - e2-micro (2 vCPU, 1 core, 1 GB memory)  **service account** - assign DevOps-sa  **boot disk** - 12 gb.  **node taints** - NO\_EXECUTE, key=webapp, value=mywebapp | **Node taints** - NO\_EXECUTE, key=webapp, value=mywebapp.(read and understand taints) | kubectl create namespace production  kubectl create deployment devconnect-app --image me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/app:1.0.0 --namespace production |

| 2.5  Deploy 1 replica of the DevConnect app in this namespace by applying my deployment.yaml manifest to the production namespace | 2.6  Add the appropriate tolerations to deployment.yaml file. | 2.7  Expose it using load balancer service and access it through a browser. | 2.8  Apply the service | 2.9  Get service External IP | 2.a  Fix the bug and **upload** to the artifact repository a new version with the corrected bugfix. | 2.b  Rollout the new version deployment |
| --- | --- | --- | --- | --- | --- | --- |
| kubectl apply -f deployment.yaml | kubectl edit deployment/django-app  Add the following:  tolerations:  **- effect: NoExecute**  **key: webapp**  **operator: Equal**  **value: mywebapp** | kubectl **expose deployment devconnect-app --type="LoadBalancer" --port=8000 --namespace=production** | kubectl apply -f service.yaml | kubectl get all --namespace production | kubectl edit deployment/devconnect-app & change version manually.  OR  kubectl set image deployment/devconnect-app app=me-west1-docker.pkg.dev/devconnect-final-project/test-artifacts/app:1.0.1 --namespace=production. | kubectl rollout status deployment/devconnect-app --namespace=productionCI/CD - |

## CI/CD

| 3.1  Create a Compute engine instance with the following specs | 3.2  Docker automation for creating the vm | 3.3  Create a new local repository called jenkins\_lab | 3.4  deploy.sh | 3.5  Dockerfile - best practice | 3.6  Make sure the container can use docker |
| --- | --- | --- | --- | --- | --- |
| Name - <your-name>-jenkins.  Region - me-west1(Tel-Aviv){% load static %}  Machine type - e2-medium (2 vCPU, 1 core, 4 GB memory)  service account - assign DevOps-sa  boot disk - 20 gb.  FireWall - Allow HTTP traffic.  Automation - install docker engine.  יכול להתבצע גם דרך SDK | Use startup script for installing Docker on a Compute Engine instance:  #!/bin/bash  # Install Docker  apt-get update  apt-get -y install [docker.io](http://docker.io/) | gcloud auth configure-docker \  me-west1-docker.pkg.dev | choice\_func(){  read -r -n 1 -p "Enter Y|y or N|n: " choice  while [[ $choice != 1 && $choice != 0 ]]  do  if [[ $choice == [Yy] ]]; then  choice=0  elif [[ $choice == [Nn] ]]; then  choice=1  else  echo "Invalid input, Try again."  ......................... |  | Make sure the container can use docker:  sudo chmod 666 /var/run/docker.sock  now i have my Jenkins container running with cloud SDK and docker installed,  I accessed my Jenkins server from my browser and used the UI to configure Jenkins, including installing suggested plugins and creating a user |

| 3.7  Access jenkins through the web and configure it(Install suggested plugins, create user, etc…) | 3.7.1  SSH  or  Gcloud SDK | 3.8  **SCM Checkout**:  Build is triggered by checking if change(push) has been made every 10 seconds | 3.a  Test - run django tests and check for 200(OK) response when trying to access the app. | 3.b  Push the image new version to artifact registry repository(The version must be the commit message) | 3.c  Print “the pipeline failed :(“. |
| --- | --- | --- | --- | --- | --- |
| Create jenkins-github connection | create a connection to github repo using ssh.  Or using the gcloud sdk | add  poll scm \*/6 \* \* \* \*  in the pipeline configuration | Screenshot | Screenshot | יכולות להשתמש בקובץ הקיים - [deploy.sh](http://deploy.sh/) |

Rational Table:

| Subject | Objectives | Rational |
| --- | --- | --- |
| 1. Dockerization | Use latest version of Django (4.2), django-crispy-forms (2.1),pillow (10.1.0)  with Use |  |
| 1.1 Dockerfile | best practice  use : **COPY ./django\_web\_app/requirements.txt .**  **RUN pip install -r requirements.txt**  instead of just :  RUN pip install -r requirements.txt | **נכון:**להעתיק את קובץ requirements.txt בנפרד ואז לעשות pip install. **לא נכון:** להעתיק את כל האפליקציה(התיקייה) ואז לעשות pip install.  שכן אחרת- לא משתמשים ב Caching של דוקר ומורידים את החבילות בכל פעם מחדש במקום שע״י שימוש ב caching החבילות ישמרו מראש |
| 1.2 | Create multi-stage dockerfile  **FROM python:3.9.18-slim as run**  and  **COPY --from=builder /usr/local/lib/python3.9/site-packages /usr/local/lib/python3.9/site-packages**  using Python Slim OR alpine  Instead of using  **FROM python:latest** | **נכון:** להשתמש ב multi stage  **לא נכון:** לעשות את הכל בimage אחד.  **נכון:** להשתמש ב python slim עם מספר גרסה.  **לא נכון:** להשתמש רק ב python וגרסא latest.  מה יקרה אם נפתח את האפליקציה בעתיד? המון דברים לא יתמכו |
| 1.3 | create a new volume  VOLUME ./db.sqlite3  docker run…  -v ./db.sqlite3:/db.sqlite3 | ביצוע Mount רק לקובץ הספציפי db.sqlite 3 - הקובץ הבינארי בו יושב הדאטאבייס של האפליקציה.  תשובה מתקבלת אך מופחתת נקודות- העתקת האפליקציה - VOLUME /app |
| 1.4 T-shoot | סה״כ 4. תקלות להציג + פתרון. | הורדת נקודות על פחות מ 4 |
| 1.5 | init.sh and delete.sh automation | נכון:  docker build  docker run  docker rm  docker rmi  ניתן להוסיף נקודות על Error checking |
| 1.6 | צילום מסך להוכחת תקינות | הוכחה להפעלת האפליקציה בצורה לוקאלית |
| 2.Deployment | Deploy.sh   Build and push the Docker image:  Use $VERSION for push  use gcloud auth configure docker | חשוב מאוד- gcloud auth configure docker אחרת לא ניתן לבצע push ( מי שהצליחה, כנראה ביצעה בצורה ידנית או שזה כבר קונפג בעבר בפרויקט הקודם) - להוריד על זה המון נקודות |
| deploy cluster | Create a zonal GKE standard cluster |  |
| deploy cluster | Create a pool with the values according to the requirements |  |
| deploy cluster | Create a namespace for the application |  |
| YAML | Create a deployment.yaml file | הוכחה שעובד |
| Load balancer service create | Expose it using load balancer service and access it through a browser | יצירת Load balancer Service |
| תקלה יזומה - Taints | Error - website not accessible  **ERROR:“Does not have minimum availability”** | Taints - הסבר + הוכחה לפתרון |
| תקלה לאחר Taints | ALLOWED\_HOSTS = []  התיקון - best practice  Put the IP of the service you created  גם שינוי כלשהו - מזכה בנקודות | מאשרים את ההוסט לחשיפת האפליקציה, באם נשאר ריק - האפליקציה לא ״תעלה״ |
| Artifact Registry | Upload to the artifact repository a new version with the corrected bugfix. | use deploy.sh  use versioning and bugFixes  הורדת נקודות אם לא התבצעה העלאת באג פיקס |
| Rollout - בונוס | Rollout the new version deployment  באמצעות  kubectl rollout status deployment/devconnect-app --namespace=productionCI/CD - | מי שכן ביצעה, לתת נקודות נוספות  כיוון וגם פקודת kubectl set img עושה save ל-deployment |
| 3.1 Compute engine | Create instance docker installation automation. | startup script for installing Docker on a Compute Engine instance:  #!/bin/bash  # Install Docker  apt-get update  apt-get -y install docker.io  הסקריפט המלא נמצא ב תשובות בצבע אדום תחת Docker automation |
|  | create local folder named “ jenkins lab” |  |
| Jenkins image  בשלב זה יכולים לבצע כבר את השלבים שבהמשך: gcloud  Artifact registry - upload jenkins  Artifact registry - upload jenkins | Build jenkins image with docker cli installation on it. | docker cli installed in the dockerfile. |
| Pushing to Artifact registry | upload the new img to the artifact registry | deploy.sh in jenkins\_lab folder. |
| Compute engine | pull the jenkins img from artifact registry and deploy in the instance | Create a Jenkins instance  יכול להתבצע גם דרך SDK |
| Expose Jenkins | Finding the EXTERNAL IP and opening it in bowser | ניקוד נמוך - זניח |
| Docker in Docker | make sure you can run docker commands inside the jenkins container | בפועל מה שקורה:  docker cli installed on jenkins container and integrated with the instance’s docker demon through docker.sock file that mounted. |
| תקלה - Docker in docker | We will not succeed, so we will connect as ROOT and then change the permissions - change permissions to socket | נבצע שינוי הרשאות לקובץ -docker.sock file דרך root או דרך ההוסט ( נכון יותר דרך ההוסט |
| gcloud | Add a gcloud installation to the jenkins image(Dockerfile) | Edit the Dockerfile and add a gcloud installation. |
| Artifact registry - upload jenkins | Upload the new image to the artifact registry  use semantic versioning.  יכול להתבצע בפעם אחת על ידי התקנה על ההתחלה של SDK | push the new image use semVer  1.0.1 |
| Artifact registry - upload jenkins | deploy the new version on the compute engine instance. | run deploy.sh of the jenkins\_lab using the new version. |
| Create a CI/CD pipeline(jenkinsfile) | create a connection to github repo using ssh. | create jenkins credentials and ssh key in the jenkins container. |
| Pipeline | Build is triggered by checking if change(push) has been made every 10 seconds. | add  poll scm \*/6 \* \* \* \*  in the pipeline configuration |
| Pipeline BONUS | Build is triggered every time a commit is pushed. | בכל שינוי שנעשה -> גיטהאב ישלח לג׳נקינס טריגר ל-build |
| Pipeline | Build the application. docker build the application and run it(run init.sh) | Build and run the app inside the pipeline. |
| Pipeline Tests | Run django tests and check for 200(OK) responses when trying to access the app. | run “python manage.py test”  inside the container  and curl the app to check if you get OK(200) response |
| Pipeline | If build succeeded:  Push the image new version to artifact registry repository(**The version must be the commit message**) | Use Gcloud to push the new version to the artifact repository in case the build is successful.  Extract and use the new version from the commit message. |
| Pipeline BONUS | Deploy the updated app to production cluster | Integrate jenkins and kubernetes and roll out the new version to production if the build is successful. |
| Pipeline | If build failed:  print “the pipeline failed :(“. | echo “the pipeline failed :(“ when build failed. |
| Pipeline | Always Clean up all resources and workspace when you're done. | In the post - always stage clean all resources:  docker rm  docke rmi  or delete.sh  and  delete all workspace folder content:  rm -rf \* |
| Final Product | Change Application environment to production push the new version, wait for CI to automate deployment and rollout the new version manually | פלואו שלם: 1. לשנות את סביבת העבודה של האפליקציה לproduction.  2. לחכות שהCI/CD BUILD יופעל ,יעבור את כל הבדיקות וידחוף את הגרסא החדשה לartifact repo.  3. לעשות rollout בצורה ידנית לגרסא החדשה שנוצרה. |
| Final Product for those who made the Bonus | Change Application environment to production push the new version,CI/CD is triggered instantly.  build deploys and rolls out the new version automatically. | פלואו שלם: 1. לשנות את סביבת העבודה של האפליקציה לproduction.  2. הCI/CD BUILD יופעל מיד עם דחיפת השינוי(בונוס 1)  3.הbuild יעבור את כל הבדיקות ,ידחוף את הגרסא החדשה לartifact repo ויעשה rollout בצורה אוטומטית לגרסא החדשה שנוצרה. |