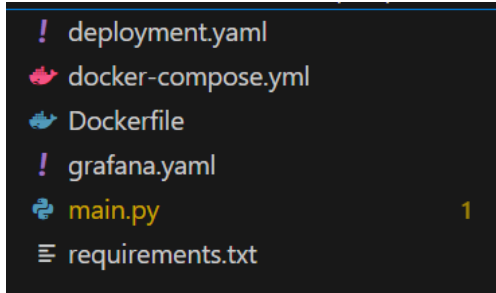


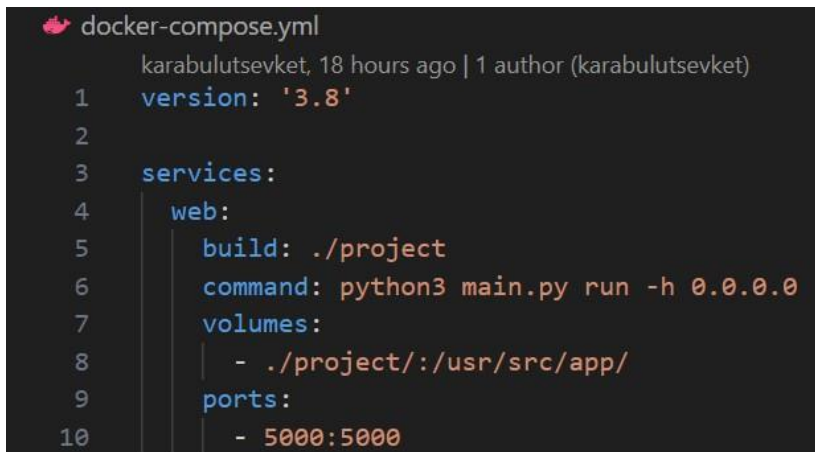
DETAYLI ANLATIM

Tamamladığım CI/CD tool'u nasıl yapıldığını anlatıyorum:

İlk olarak Visual Studio Code IDE'si ile gerekli kodları yazalım:



docker-compose.yml



deployment.yaml

```
! deployment.yaml
karabulutsevkett, 18 hours ago | 1 author (karabulutsevkett)
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: app
5    namespace: default
6    labels:
7      app: app
8  spec:
9    replicas: 3
10   selector:
11     matchLabels:
12       app: app
13   template:
14     metadata:
15       labels:
16         app: app
17     spec:
18       containers:
19         - name: app
20           image: mirrotech/main:latest
21           ports:
22             - containerPort: 5000
23
24   ---
25   apiVersion: v1
26   kind: Service
27   metadata:
28     name: nobel-api-service
29     namespace: default
30   spec:
31     selector:
32       app: app
33     ports:
34       - protocol: TCP
35         port: 5000
36         targetPort: 5000
37     type: LoadBalancer
```

Dockerfile

```
Dockerfile > FROM
karabulutsevket, 18 hours ago | 2 authors (karabulutsevket and others)
1 FROM python:3.11.3-slim-buster karabulutsevket, 18
2
3 # set work directory
4 WORKDIR /usr/src/app
5
6 # set environment variables
7 ENV PYTHONDONTWRITEBYTECODE 1
8 ENV PYTHONUNBUFFERED 1
9
10 # install dependencies
11 RUN pip install --upgrade pip
12 COPY ./requirements.txt /usr/src/app/requirements.txt
13 RUN pip install -r requirements.txt
14
15 # copy project
16 COPY . /usr/src/app/
17
18 CMD ["python", "main.py"]
```

main.py

```
main.py > ...
karabulutsevket, 19 hours ago | 1 author (karabulutsevket)
1 from flask import Flask karabulutsevket, 19 hours ago • Update app.py
2
3 app = Flask(__name__)
4
5
6 @app.route("/")
7 def hello():
8     return "<h1 style='color:black'>Merhaba ben Sevket Karabulut! Bu bir Flask web app.</h1>"
9
10
11 if __name__ == "__main__":
12     app.run(host='0.0.0.0', port=5000)
```

requirements.txt

```
requirements.txt
karabulutsevket, 19 hours ago | 1 author (karabulutsevket)
1 flask karabulutsevket, 19 hours ago • Update requirements.txt
2 Flask-RESTful
```

grafana.yaml

```
! grafana.yaml
karabulutsevket, 2 hours ago | 1 author (karabulutsevket)
1  grafana:      karabulutsevket, 2 hours ago • Create grafana.yaml
2    adminUser:  admin
3    adminPassword: password
4
5    ingress:
6      enabled: true
7      annotations:
8        kubernetes.io/ingress.class: "nginx"
9        nginx.ingress.kubernetes.io/rewrite-target: /$1
10       nginx.ingress.kubernetes.io/use-regex: "true"
11      path: /grafana/(?.*)
12      hosts:
13        - k8s.cluster.local
14
15    plugins:
16      - digrich-bubblechart-panel
17      - grafana-clock-panel
18      - grafana-piechart-panel
19
20    grafana.ini:
21      server:
22        root_url: http://localhost:3000/grafana
23      smtp:
24        enabled: true
25        host: smtp.gmail.com:587
26        user: "sevket.karabulut110@gmail.com"
27        password: "admin"
28        skip_verify: true
```

Ardından kodları GitHub'a ekleyelim.

Google VM instances kısmına gidip herhangi bi isimde VM instance oluřturalım:

VM instances									
CREATE INSTANCE IMPORT VM REFRESH									
INSTANCES OBSERVABILITY INSTANCE SCHEDULES									
VM instances									
Filter Enter property name or value									
<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect	
<input type="checkbox"/>	✓	vmmirror	europa-west3-c			10.156.0.2 (nic0)	34.159.39.165 (nic0)	SSH	⋮
Related actions									

vmmirror ierięi řu řekildeydi:

DETAILS		OBSERVABILITY	OS INFO	SCREENSHOT
Logs				
Logging				
Serial port 1 (console)				
SHOW MORE				
Basic information				
Name	vmmirror			
Instance id	1757107132938074886			
Description	None			
Type	Instance			
Status	✓ Running			
Creation time	Jan 31, 2024, 2:27:41 AM UTC+03:00			
Zone	europa-west3-c			
Instance template	None			
In use by	None			
Reservations	Automatically choose			
Labels	None			
Tags ?	-			
Deletion protection	Disabled			
Confidential VM service ?	Disabled			
Preserved state size	0 GB			

Wind
Window

Machine configuration

Machine type	e2-small
CPU platform	Intel Broadwell
Minimum CPU platform	None
Architecture	x86/64
vCPUs to core ratio [?]	—
Custom visible cores [?]	—
Display device	Disabled Enable to use screen capturing and recording tools
GPUs	None
Resource policies	

Networking

Public DNS PTR Record	None
Total egress bandwidth tier	—
NIC type	—

[→ VIEW IN NETWORK TOPOLOGY](#)

Firewalls

HTTP traffic	On
HTTPS traffic	On

Firewalls

HTTP traffic	On
HTTPS traffic	On
Allow Load Balancer Health checks	Off

Network tags

[http-server](#) [https-server](#)

Network interfaces

Name [↑]	Network	Subnetwork	Primary internal IP address	Alias IP ranges	IP stack type	External IP address	Network
nic0	default	default	10.156.0.2		IPv4	34.159.39.165 (Ephemeral)	Premium

Storage

Boot disk

Name [↑]	Image	Interface type	Size (GB)	Device name	Type	Architecture	Encryption	Mode	Whisker
vmmirror	debian-11-bullseye	SCSI	100	vmmirror	Balanced persistent	x86/64	Google-managed	Boot, read/write	Default

Local disks

None

Additional disks

None

Security and access

Shielded VM [?](#)

Secure Boot ?	Off
vTPM ?	On
Integrity Monitoring ?	On

SSH Keys

SSH keys

Username	Key	
sevket_karabulut110	ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBHV4/oJbdyK63iHWqR2y73FqSFQSI...	▼
sevket_karabulut110	ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQComFzjLXFudvUwewExHSX2X7ocvfUWJsXX/WRVVEwfhNXGMTrCj+dAe09n...	▼

API and identity management

Service account	123560372333-compute@developer.gserviceaccount.com
Cloud API access scopes	Allow default access

[▼ SHOW DETAILS](#)

Management

Data Encryption

Key ID	—
Key name	—

Availability policies

VM provisioning model ?	Standard
Max duration ?	None
Preemptibility	Off (Recommended)
On VM termination ?	—
On host maintenance	Migrate VM instance (Recommended)
Automatic restart	On (Recommended)
Customer Managed Encryption Key (CMEK) revocation policy	Do nothing

Ardından oluşturduğumuz vm instances'in Connect sütunundaki SSH kısmına tıklayalım.

INSTANCES								
OBSERVABILITY								
INSTANCE SCHEDULES								
VM instances								
<div>Filter Enter property name or value</div>								
<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	✓	vmmirror	europa-west3-c			10.156.0.2 (nic0)	34.159.39.165 (nic0)	SSH

Açılan yerde(SSH-in-browser) sırasıyla komutları girelim:

```
$ sudo apt update
```

```
$ sudo apt install apt-transport-https ca-certificates curl  
gnupg2 software-properties-common
```

```
$ curl -fsSL
```

```
https://download.docker.com/linux/debian/gpg | sudo  
apt-key add -
```

```
$ sudo add-apt-repository "deb [arch=amd64]
```

```
https://download.docker.com/linux/debian
```

```
$(lsb_release -cs) stable"
```

```
$ sudo apt update
```

```
$ apt-cache policy docker-ce
```

```
$ sudo apt install docker-ce
```

```
$ sudo systemctl status docker
```

```
$ sudo usermod -aG docker ${USER}
```

```
$ id -nG
```

```
$ sudo apt install python3-pip
```



```
$ sudo apt update
```

```
$ java -version
```

```
$ sudo apt install default-jre
```

```
$ java -version
```

```
$ sudo apt install default-jdk
```

```
$ javac -version
```

```
$ wget -q -O - https://pkg.jenkins.io/debian-  
stable/jenkins.io.key | sudo gpg --dearmor -o  
/usr/share/keyrings/jenkins.gpg
```

```
$ sudo sh -c 'echo deb [signed-  
by=/usr/share/keyrings/jenkins.gpg]  
http://pkg.jenkins.io/debian-stable binary/ >  
/etc/apt/sources.list.d/jenkins.list'
```

```
$ sudo apt update
```

```
$ sudo apt install jenkins
```

```
$ sudo wget -O /usr/share/keyrings/jenkins-keyring.asc  
https://pkg.jenkins.io/debian-stable/jenkins.io-  
2023.key
```

```
$ echo deb [signed-by=/usr/share/keyrings/jenkins-  
keyring.asc] https://pkg.jenkins.io/debian-stable  
binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list >  
/dev/null
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install jenkins
```

```
$ sudo systemctl start jenkins.service
```

```
$ sudo systemctl status jenkins
```

```
$ sudo cat
```

```
/var/lib/jenkins/secrets/initialAdminPassword
```

```
$ sudo apt-get install wget apt-transport-https gnupg  
lsb-release
```

```
$ wget -qO - https://aquasecurity.github.io/trivy-  
repo/deb/public.key | gpg --dearmor | sudo tee  
/usr/share/keyrings/trivy.gpg > /dev/null
```

```
$ echo "deb [signed-by=/usr/share/keyrings/trivy.gpg]  
https://aquasecurity.github.io/trivy-repo/deb  
$(lsb_release -sc) main" | sudo tee -a  
/etc/apt/sources.list.d/trivy.list
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install trivy
```

```
$ sudo usermod -aG docker jenkins
```

```
$ sudo systemctl restart Jenkins
```

Artık vmmirror'u kapatıp açtıktan sonra Jenkins üzerinde:

<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	✓	vmmirror	europa-west3-c			10.156.0.2 (nic0)	34.159.39.165 (nic0)	SSH ▾ ⋮

External IP kullanılabilir olacaktır.(34.159.39.165:8080)

Jenkins'e giriş yaptıktan sonra yeni bir iş oluşturalım.

Serbest still proje seçelim ve gerekli işlemleri yapalım:

Git ?

Repositories ?

Repository URL ?

https://github.com/karabulutsevkett/test3.git

Credentials ?

- none - ▾

+ Add ▾

Gelişmiş ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

Shell çalıştır ?

Komut

Tüm ortam değişkenlerini görmek için tıklayınız

```
pip install -r requirements.txt
docker build -t main:latest .
trivy image --scanners vuln main
docker login -u mirrotech -p 895668806
docker tag main:latest mirrotech/main:latest
docker push mirrotech/main:latest
docker run -p 5000:5000 main:latest
```

Gelişmiş ▾

Vulnerability scan with gype ?

Scan destination ?

dir:/var/lib/jenkins/workspace/test

Name of resut report

gypeReport_\${test}_\${latest}.txt

☒ Download and install gype automatically ?

Oluşturduğumuz test adlı projeyi Şimdi Yapılandır seçeneği ile çalıştıralım.

- Durum

</> Değişiklikler


Çalışma Alanı

Şimdi Yapılandır

Konfigürasyonu Düzenle

Projeyi Sil

GitHub Hook Log

Adını Değiştir
-  Yapılandırma Geçmişi

Eğilim ▾

Ardından şu çıktıyı alıyoruz:

```
f1f336716730: Preparing
17a2e74fff66: Preparing
b814cfcb4046: Preparing
ad383980bcf6: Preparing
37b14643f733: Preparing
d85b356ec3b5: Preparing
17a2e74fff66: Waiting
b814cfcb4046: Waiting
ad383980bcf6: Waiting
37b14643f733: Waiting
d85b356ec3b5: Waiting
4e78fdc158cb: Layer already exists
c502675a7ac8: Layer already exists
17d7b7f15de4: Layer already exists
f1f336716730: Layer already exists
17a2e74fff66: Layer already exists
b814cfcb4046: Layer already exists
664dd204acf8: Layer already exists
37b14643f733: Layer already exists
d85b356ec3b5: Layer already exists
ad383980bcf6: Layer already exists
latest: digest: sha256:52bfb6c7dc9391016a6fdff72465e6c9ba26d7664518185494f1e21fb2d49586 size: 2419
+ docker run -p 5000:5000 main:latest
* Serving Flask app 'main'
* Debug mode: off
-[31m-[1mWARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.-[0m
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
-[33mPress CTRL+C to quit-[0m
```

Ardından Jenkins'te CI işlemini tamamlamış olduk. ArgoCD'ye giriş yapmak için firewall ve Kubernetes Cluster oluşturalım:

<input type="checkbox"/>	<input checked="" type="checkbox"/>	mirror	europa-central2	Autopilot	5	20 GB	-	⋮
--------------------------	-------------------------------------	------------------------	-----------------	-----------	---	-------	---	---

Ardından active Shell kısmında komutları sırasıyla yazalım:

```
$ gcloud container clusters get-credentials
CLUSTER_NAME --zone YOUR_ZONE
```

```
$ kubectl create namespace argocd
```

```
$ kubectl apply -n argocd -f  
https://raw.githubusercontent.com/argoproj/argocd/stable/manifests/install.yaml
```

```
$ kubectl get ns
```

```
$ kubectl get pods -n argocd
```

```
$ kubectl get svc -n argocd
```

```
$ kubectl get pods -n argocd
```

```
$ kubectl get svc -n argocd
```

```
$ kubectl patch svc argocd-server -n argocd -p '{"spec":  
{"type": "LoadBalancer"}}'
```

```
$ kubectl get svc -n argocd
```

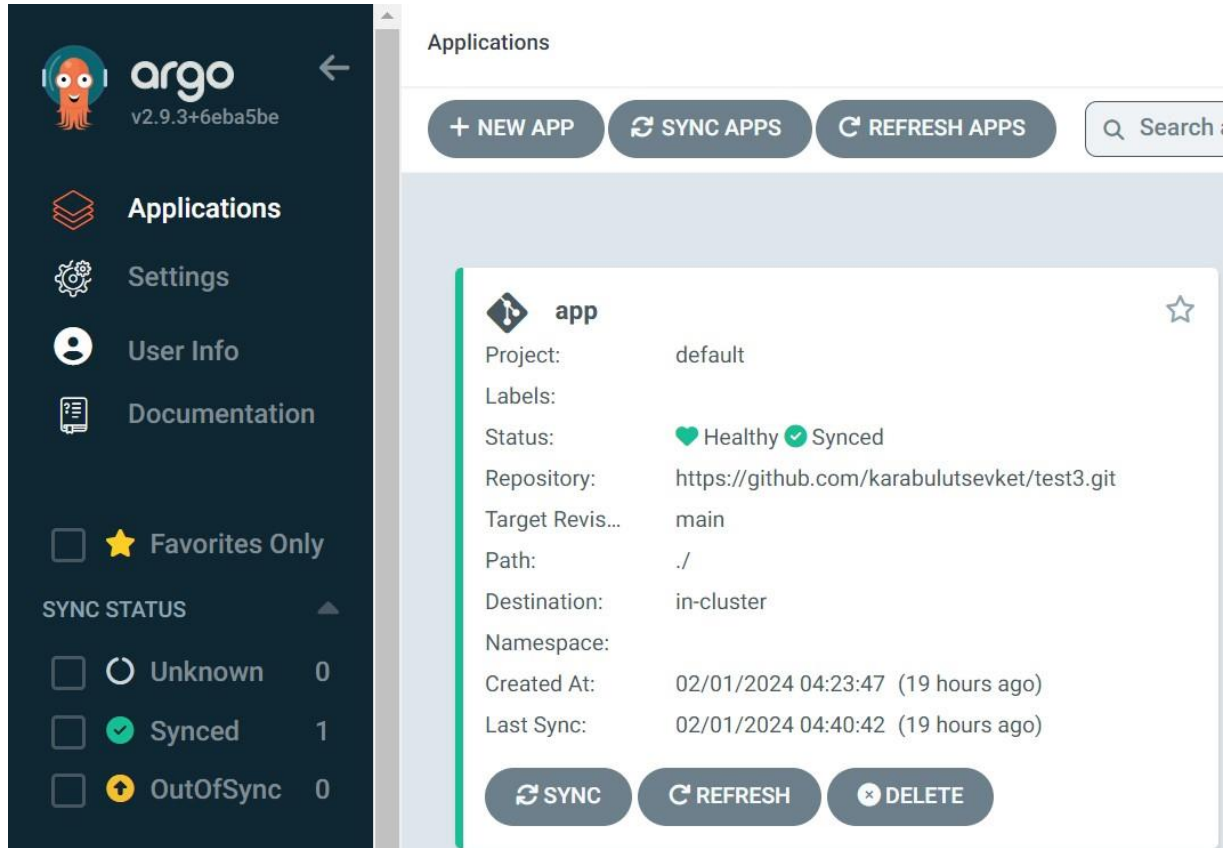
```
$ kubectl get svc -n argocd
```

```
$ kubectl get secret -n argocd
```

```
$ kubectl get secret argocd-initial-admin-secret -n  
argocd -o json
```

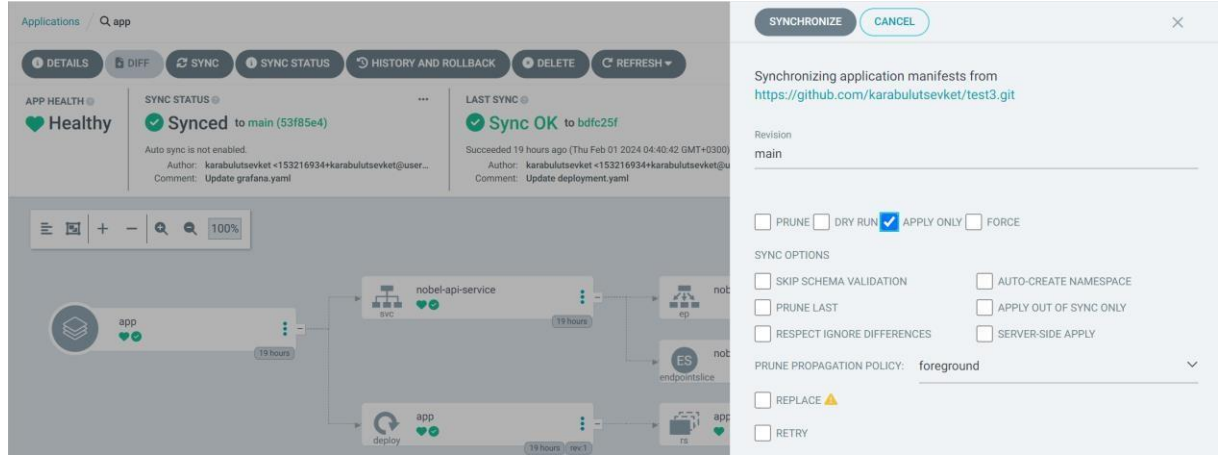
```
$ kubectl get secret argocd-initial-admin-secret -n  
argocd -o json | jq .data.password -r | base64 -d
```

Komutlarımızı ekledikten sonra ArgoCD ye giriş için verilen ip'mizi alıyoruz. Şifremizi ise (<https://www.base64decode.org/>) sitesinden decode yardımıyla asıl şifremize ulaşıyoruz. Ardından ip adresimize girip admin kullanıcı adı ve asıl şifremizle giriş yapıyoruz. Applications kısmına giriyoruz ve ekleme yapıyoruz:



+NEW APP kısmına tıklayıp projemizin adını deployment.yaml da yazdığımız koda göre veriyoruz. Ardından GitHub linki, Branchi ve Path yolunu giriyoruz. Cluster kısmını seçip Namespace kısmını boş bırakıyoruz(Default da yazılabilir). Ardından SYNC ye

basarak Apply Only seçeneğini tik atıp SYNCHRONIZE kısmına tıklıyoruz ve sonuçların çıkmasını bekliyoruz:



Sonuç: CI/CD Tools tamamlandı ve Kubernetes dağıtımı yapıldı.

