##EC2 Comands

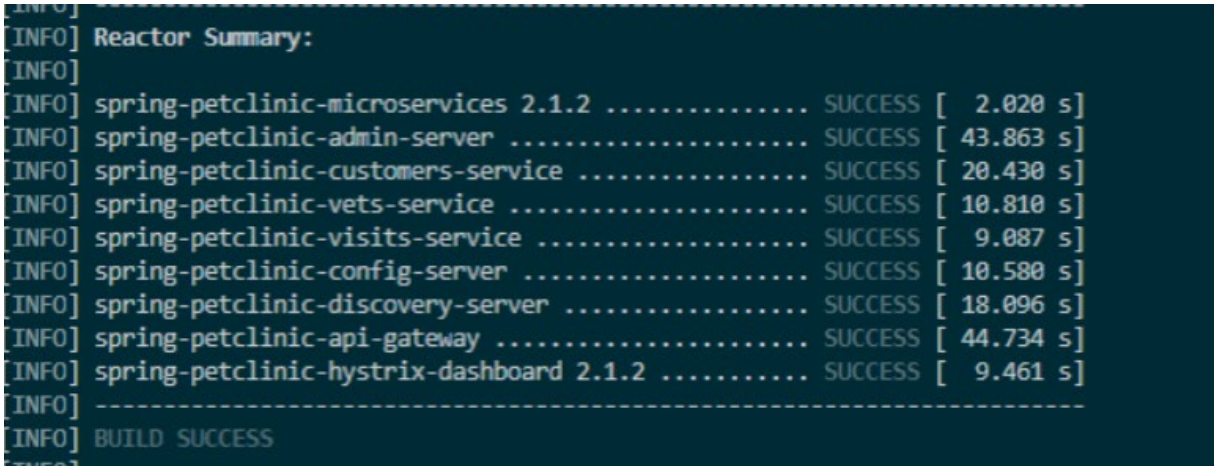
1. sudo yum update -y # Ec2 update edin
2. sudo hostnamectl set-hostname petclinic-dev-server # Ec2 adini petclinic-dev-server olarak degistirin
3. bash # Petclinic-dev-server ismini gorelim
4. sudo amazon-linux-extras install docker -y # Docker kuralim
5. sudo systemctl start docker # Docker aktive edelim
6. sudo systemctl enable docker
7. sudo usermod -a -G docker ec2-user
8. sudo curl -L "https://github.com/docker/compose/releases/download/1.26.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
9. sudo chmod +x /usr/local/bin/docker-compose
10. sudo yum install git -y # Ec2 icine git yukleyelim
11. sudo yum install java-11-amazon-corretto -y # Ec2 icine maven icin java kuralim
12. git clone https://github.com/clarusway/petclinic-microservices.git # Ec2 icine proje dosyasini cekelim
13. cd petclinic-microservices/
14. ls -a
15. rm -rf .git # Eski .git dosyasini silelim
16. ls -a
17. git init # Tekrardan git init baslatalim. Burdaki amac ec2 ile repomuzu birbirine baglamak ve icindekileri kendi repomuza push edebilmek.
18. ls -a
19. git add .
20. git commit -m "first commit"
21. git config --global user.name davidclarusway # Git hesabimizi baglayalim
22. git config --global user.email david@clarusway
23. git config --global credential.helper store # Her seferinde sifre girmemizi onleyelim
24. git commit -m "first commit"
25. git remote add origin https://github.com/Gokay2705/petclinic-microservices.git
26. git push -f origin master
27. git checkout master # Master bransindamiyiz kontrol edelim
28. git branch dev # Dev bransi olusturalim
29. git checkout dev
30. git push -u origin dev
31. git checkout master
32. git branch release # Release bransi olusturalim
33. git checkout release
34. git push -u origin release
35. git branch

36. git branch -a # Tum branslari gorelim

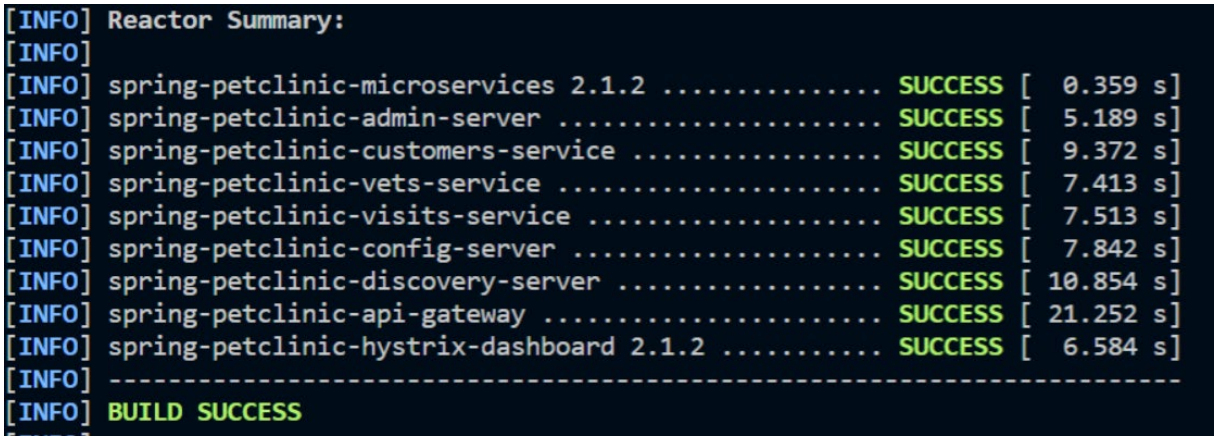
    Bu kodu girince çıktısı aşağıda olacaktır

    dev
    master
    * release
    remotes/origin/dev
    remotes/origin/master
    remotes/origin/release

37. ./mvnw clean test # Mvn dosyainda hata var mi test edelim




38. sudo chmod +x mvnw # Mvn dosyasina “execution” yetkisi verelim
39. ./mvnw clean package



40. ./mvnw clean install # Mvn dosyasini yukleyelim. Burada tum microservices'ler tek tek yuklenecek.

```
[INFO] Reactor Summary:
[INFO]
[INFO] spring-petclinic-microservices 2.1.2 ..... SUCCESS [  1.827 s]
[INFO] spring-petclinic-admin-server ..... SUCCESS [  3.928 s]
[INFO] spring-petclinic-customers-service ..... SUCCESS [  9.551 s]
[INFO] spring-petclinic-vets-service ..... SUCCESS [  7.721 s]
[INFO] spring-petclinic-visits-service ..... SUCCESS [  8.198 s]
[INFO] spring-petclinic-config-server ..... SUCCESS [  8.208 s]
[INFO] spring-petclinic-discovery-server ..... SUCCESS [ 13.171 s]
[INFO] spring-petclinic-api-gateway ..... SUCCESS [ 21.992 s]
[INFO] spring-petclinic-hystrix-dashboard 2.1.2 ..... SUCCESS [  6.673 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

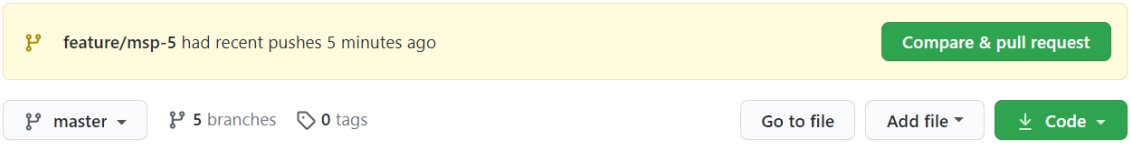
41. git checkout dev
42. git branch feature/msp-4 # Feature/msp-4  bransi olusturalim
43. git checkout feature/msp-4
44. git add .
45. git commit -m "added mvn package script"
46. git push -u origin feature/msp-4
47. git checkout dev # Dev bransina « merge » etmek icin geciyoruz.
48. git merge feature/msp-4 # "merge" ediyoruz.
49. git push -u origin dev
50. git branch feature/msp-5 # Feature/msp-5  bransi olusturalim
51. git checkout feature/msp-5
52. "infrastructure" isimli klasor olusrutup icine "dev-server-for-petclinic-cf-template.yml" isimli yaml dosyasi olustur.


```
AWSTemplateFormatVersion: 2010-09-09
Description: >
  This cloudformation template prepares development environment for petclinic microservices app
  User needs to select appropiate key name when launching the template.
Parameters:
  KeyPairName:
    Description: Enter the name of your Key Pair for SSH connection
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: Must be one of the existing EC2 keypair
Resources:
  PetclinicSG:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable HTTP and SSH for petclinic microservices
```

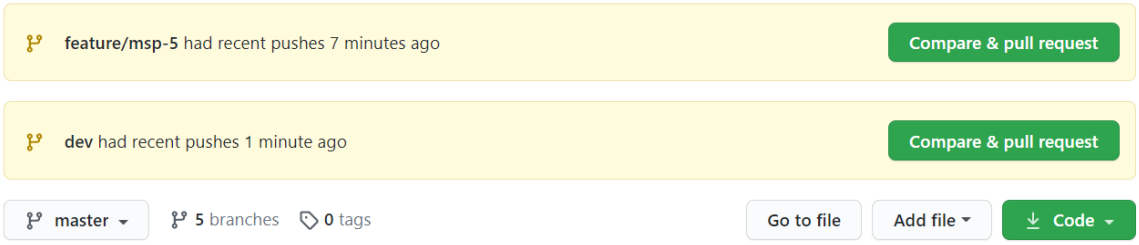
```
SecurityGroupIngress:
  - IpProtocol: tcp
    FromPort: 22
    ToPort: 22
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 80
    ToPort: 80
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 8888
    ToPort: 8888
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 8761
    ToPort: 8761
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 8081
    ToPort: 8081
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 8082
    ToPort: 8082
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 8083
    ToPort: 8083
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 8080
    ToPort: 8080
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 9411
    ToPort: 9411
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 9090
    ToPort: 9090
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 7979
    ToPort: 7979
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 3000
    ToPort: 3000
```

```
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 9091
        ToPort: 9091
        CidrIp: 0.0.0.0/0
  PetClinicLT:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId: ami-04d29b6f966df1537
        InstanceType: t2.medium
        KeyName: !Ref KeyPairName
        SecurityGroupIds:
          - !GetAtt PetclinicSG.GroupId
        UserData:
          Fn::Base64: |
            #! /bin/bash
            yum update -y
            hostnamectl set-hostname petclinic-dev-server
            amazon-linux-extras install docker -y
            systemctl start docker
            systemctl enable docker
            usermod -a -G docker ec2-user
            curl -L "https://github.com/docker/compose/releases/download/1.26.2/docker-compose-
$(uname -s)-$(uname -m)" \
            -o /usr/local/bin/docker-compose
            chmod +x /usr/local/bin/docker-compose
            yum install git -y
            yum install java-11-amazon-corretto -y
            git clone https://github.com/Gokay2705/petclinic-microservices.git # Insert the link to your
own repo
            cd petclinic-microservices/
            git fetch
            git checkout dev
  PetClinicServer:
    Type: AWS::EC2::Instance
    Properties:
      LaunchTemplate:
        LaunchTemplateId: !Ref PetClinicLT
        Version: !GetAtt PetClinicLT.LatestVersionNumber
      Tags:
        - Key: Name
          Value: !Sub Petclinic App Dev Server ${AWS::StackName}
Output:
  PetClinicDNSName:
    Description: Petclinic App Url
    Value: !GetAtt PetClinicServer.PublicDnsName
```

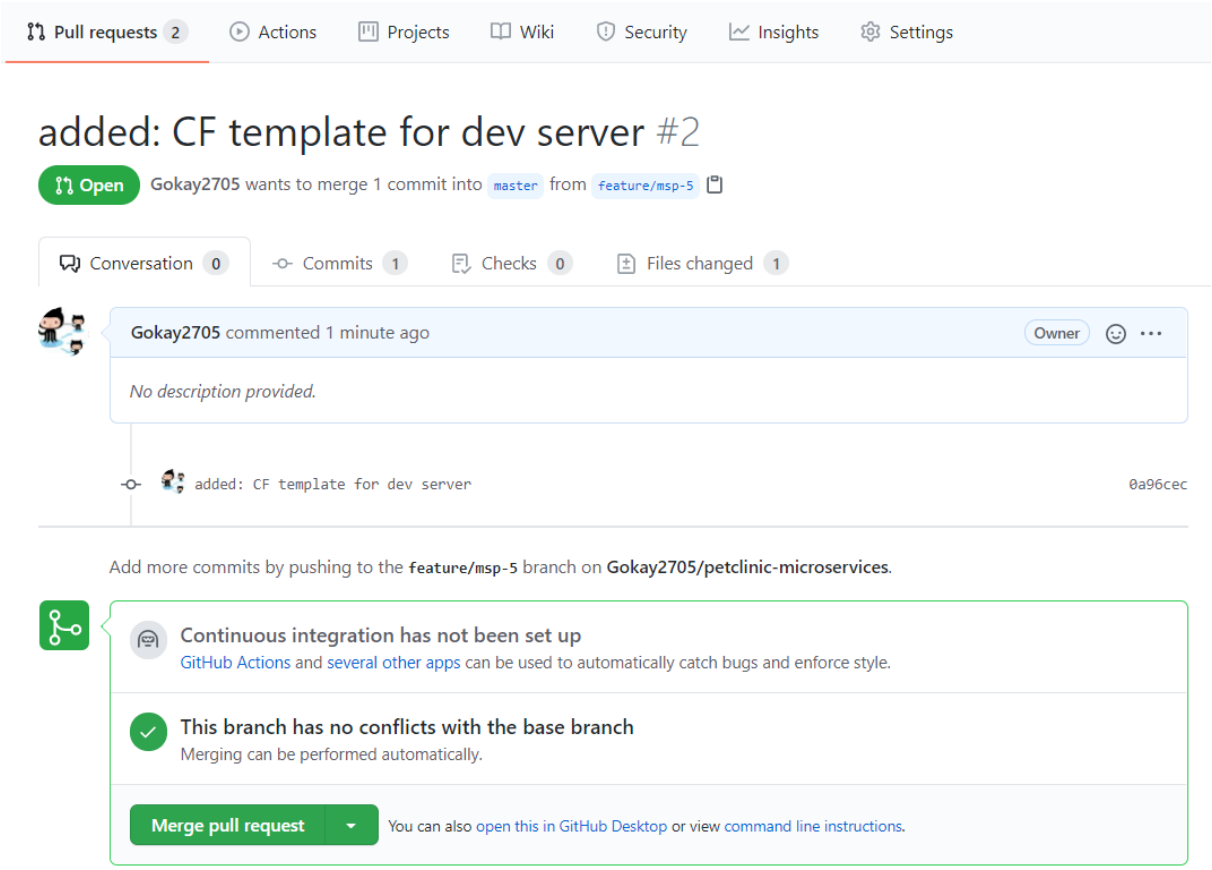
53. git checkout feature/msp-5
54. git add .
55. git commit -m "added CF template for dev server"
56. git push -u origin feature/msp-5 # Yaptigimiz degisikleri push ettik
57. git checkout dev # Yapilan degisikleri merge etmek icin dev gectik
58. git merge feature/msp-5 # Merge ediyoruz



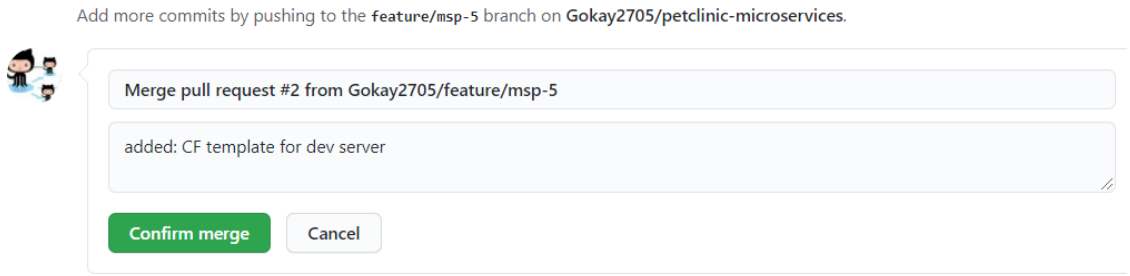

59. git push -u origin dev #  merge ettigimiz dosyayi push ediyoruz.




60. Github hesabina giderek yapilan “pull request”leri “merge” edin.



61. Yapilan “merge pull reguest” talebini onaylayin.

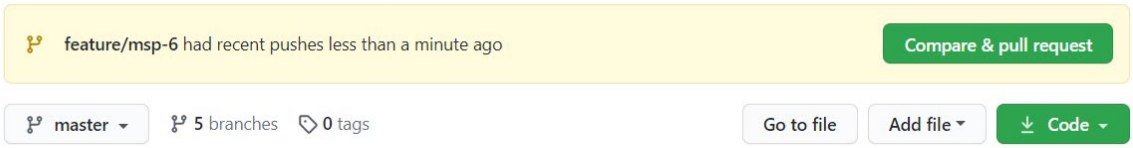



**PART-II**

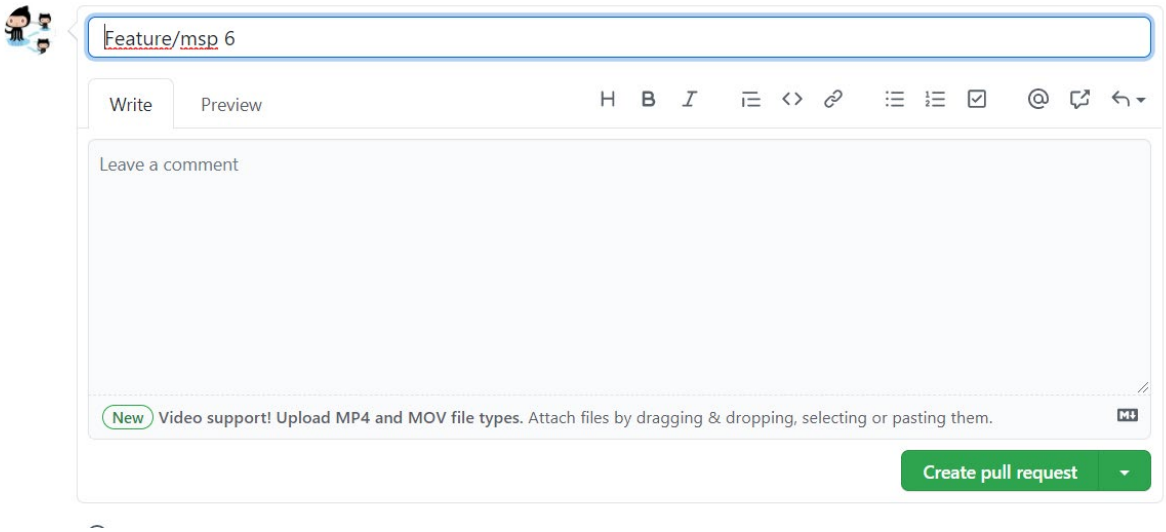
62. Git checkout dev
63. git branch feature/msp-6 # git checkout -b feature/msp-6
64. git checkout feature/msp-6
65. “admin server directory” klasoru icinde “Dockerfile” olustirarak imaj olusturacagiz.

    FROM openjdk:11-jre
    ARG DOKERSIZE:VERSION=V0.6.1
    ARG EXPOSED_PORT=9090
    ENV SPRING_PROFILES_ACTIVE docker
    ADD
    https://github.com/jwilder/dockerize/releases/download/${DOCKERIZE_VERSION}/dockerize-alpine-linux-amd64-${DOCKERIZE_VERSION}.tar.gz dockerize.tar.gz
    RUN tar -xzf dockerize.tar.gz
    RUN chmod +x dockerize
    ADD ./target/*.jar /app.jar # target klasoru icindeki tum jar uzantili dosyalari al app.jar olarak image icine ekle
    EXPOSE ${EXPOSED_PORT}
    ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"] # illa bu komutun calismasini istiyorsak CMD yerine kullaniriz. Java kodunun durmaini engelleyen kod “urandom”.
66. “spring-petclinic-api-gateway” klasoru icine ayni Dockerfile dosyasini kopyala EXPOSED_PORT=8080 yap.
67. “spring-petclinic-config-server” klasoru icine ayni Dockerfile dosyasini kopyala EXPOSED_PORT=8888 yap.
68. “spring-petclinic-customer-service” klasoru icine ayni Dockerfile dosyasini kopyala EXPOSED_PORT=8081 yap.
69. “spring-petclinic-discovery-server” klasoru icine ayni Dockerfile dosyasini kopyala EXPOSED_PORT=8761 yap.
70. “spring-petclinic-hystrix-dashboard” klasoru icine ayni Dockerfile dosyasini kopyala EXPOSED_PORT=7979 yap.
71. “spring-petclinic-vets-service” klasoru icine ayni Dockerfile dosyasini kopyala EXPOSED_PORT=8083 yap.
72. “spring-petclinic-visits-service” klasoru icine ayni Dockerfile dosyasini kopyala EXPOSED_PORT=8082 yap.
73. git add .
74. git commit -m 'added Dockerfiles for microservices'

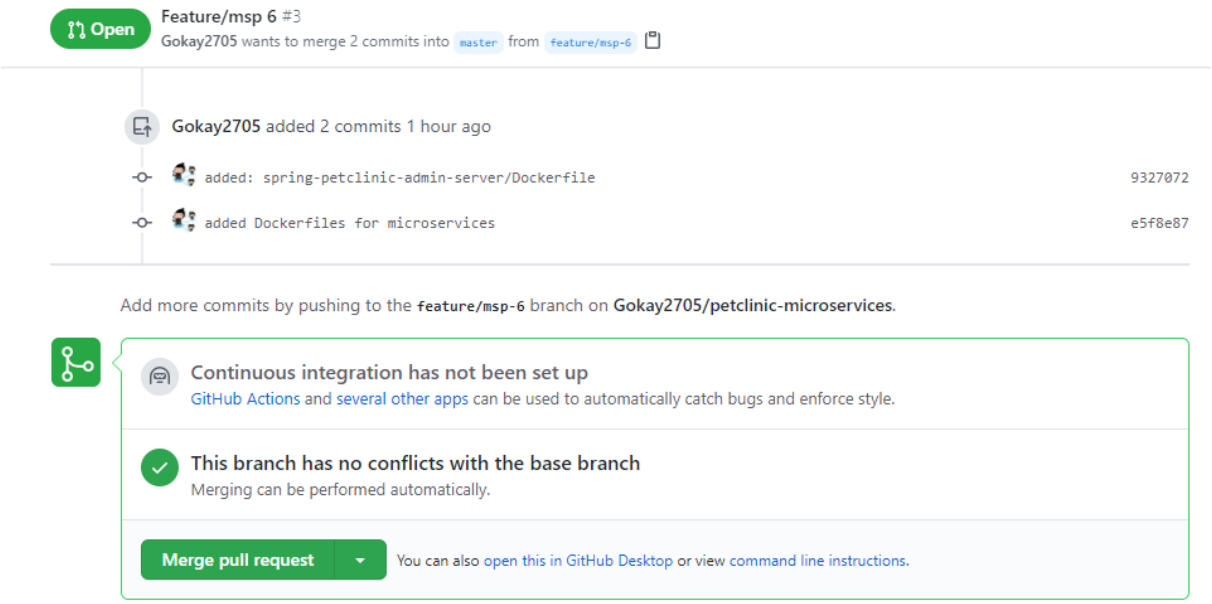
75. git push --set-upstream origin feature/msp-6
76. git checkout dev
77. git merge feature/msp-6
78. git push origin dev



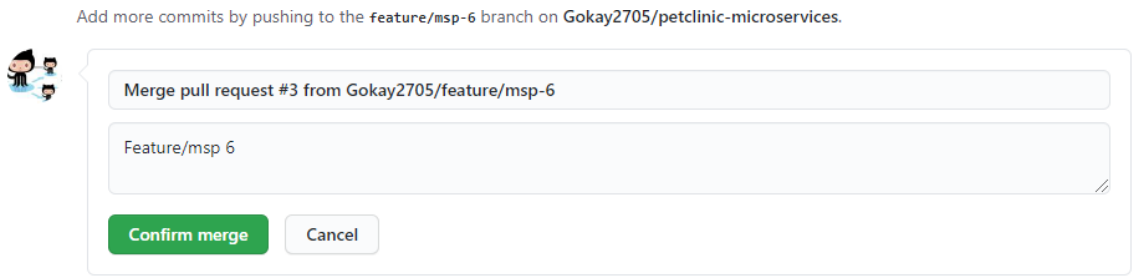

79. Compare & pull request




80. Create pull request




81. Merge pull request




82. Confirm merge ile feature/msp-6 yapilan degisikleri merge etmis olduk.
83. git checkout dev
84. git branch feature/msp-7 # feature/msp-7 bransi olusturuyoruz.
85. git checkout feature/msp-7
86. git branch

```
[ec2-user@petclinic-dev-server petclinic-microservices]$ git branch
  dev
  feature/msp-4
  feature/msp-5
  feature/msp-6
* feature/msp-7
  master
  release
```

87. Tum Dockerfile calistirarak image olusturacak “build-dev-docker-images.sh” isimli komut yazalim.

```
./mvnw clean package # klasor icindeki tum target klasorlerini silecek
docker build --force-rm -t "petclinic-admin-server:dev" ./spring-
petclinic-admin-server
docker build --force-rm -t "petclinic-api-gateway:dev" ./spring-
petclinic-api-gateway
docker build --force-rm -t "petclinic-config-server:dev" ./spring-
petclinic-config-server
docker build --force-rm -t "petclinic-customers-service:dev" ./spring-
petclinic-customers-service
docker build --force-rm -t "petclinic-discovery-server:dev" ./spring-
petclinic-discovery-server
docker build --force-rm -t "petclinic-hystrix-dashboard:dev" ./spring-
petclinic-hystrix-dashboard
docker build --force-rm -t "petclinic-vets-service:dev" ./spring-
petclinic-vets-service
docker build --force-rm -t "petclinic-visits-service:dev" ./spring-
petclinic-visits-service
docker build --force-rm -t "petclinic-grafana-
server:dev" ./docker/grafana
docker build --force-rm -t "petclinic-prometheus-
server:dev" ./docker/prometheus
```

88. git add .
89. git commit -m 'added script for building docker images'
90. git push --set-upstream origin feature/msp-7
91. git checkout dev
92. git merge feature/msp-7
93. git push origin dev
94. 79. Adimdan sonrasini tekrarla.


95. git checkout dev
96. git branch feature/msp-8
97. git checkout feature/msp-8
98. Aplikasyonu deloy etmek icin “docker-compose-local.yml” dosyasini olusturalim.

```
version: '2'

services:
  config-server:
    image: petclinic-config-server:dev
    container_name: config-server
    mem_limit: 512M # Ec2 memorysini yememesi icin sinir belirliyoruz.
    ports:
     - 8888:8888

  discovery-server:
    image: petclinic-discovery-server:dev
    container_name: discovery-server
    mem_limit: 512M
    depends_on:
    - config-server
    entrypoint: ["./dockerize","-wait=tcp://config-server:8888","-
timeout=60s","--","java", "-Djava.security.egd=file:/dev/./urandom","-
jar","/app.jar"] # -- dockerize’dan java’ya geciyoruz.Bu maksatla
kullanilir.
    ports:
     - 8761:8761

  customers-service:
    image: petclinic-customers-service:dev
    container_name: customers-service
    mem_limit: 512M
    depends_on:
    - config-server
    - discovery-server
```

```
    entrypoint: ["./dockerize","-wait=tcp://discovery-server:8761","-
timeout=60s","--","java", "-Djava.security.egd=file:/dev/./urandom","-
jar","/app.jar"]
    ports:
    - 8081:8081

  visits-service:
    image: petclinic-visits-service:dev
    container_name: visits-service
    mem_limit: 512M
    depends_on:
     - config-server
     - discovery-server
    entrypoint: ["./dockerize","-wait=tcp://discovery-server:8761","-
timeout=60s","--","java", "-Djava.security.egd=file:/dev/./urandom","-
jar","/app.jar"]
    ports:
     - 8082:8082

  vets-service:
    image: petclinic-vets-service:dev
    container_name: vets-service
    mem_limit: 512M
    depends_on:
     - config-server
     - discovery-server
    entrypoint: ["./dockerize","-wait=tcp://discovery-server:8761","-
timeout=60s","--","java", "-Djava.security.egd=file:/dev/./urandom","-
jar","/app.jar"]
    ports:
     - 8083:8083

  api-gateway:
    image: petclinic-api-gateway:dev
    container_name: api-gateway
    mem_limit: 512M
    depends_on:
     - config-server
     - discovery-server
    entrypoint: ["./dockerize","-wait=tcp://discovery-server:8761","-
timeout=60s","--","java", "-Djava.security.egd=file:/dev/./urandom","-
jar","/app.jar"]
    ports:
     - 8080:8080

  tracing-server:
    image: openzipkin/zipkin
    container_name: tracing-server
    mem_limit: 512M
```

```
    environment:
    - JAVA_OPTS=-XX:+UnlockExperimentalVMOptions -
Djava.security.egd=file:/dev/./urandom # Java 8-9 da memory problemini
cozmek icin kullanilir.
    ports:
     - 9411:9411

  admin-server:
    image: petclinic-admin-server:dev
    container_name: admin-server
    mem_limit: 512M
    depends_on:
     - config-server
     - discovery-server
    entrypoint: ["./dockerize","-wait=tcp://discovery-server:8761","-
timeout=60s","--","java", "-Djava.security.egd=file:/dev/./urandom","-
jar","/app.jar"]
    ports:
     - 9090:9090

  hystrix-dashboard:
    image: petclinic-hystrix-dashboard:dev
    container_name: hystrix-dashboard
    mem_limit: 512M
    depends_on:
     - config-server
     - discovery-server
    entrypoint: ["./dockerize","-wait=tcp://discovery-server:8761","-
timeout=60s","--","java", "-Djava.security.egd=file:/dev/./urandom","-
jar","/app.jar"]
    ports:
     - 7979:7979

  ## Grafana / Prometheus

  grafana-server:
    image: petclinic-grafana-server:dev
    container_name: grafana-server
    mem_limit: 256M
    ports:
    - 3000:3000

  prometheus-server:
    image: petclinic-prometheus-server:dev
    container_name: prometheus-server
    mem_limit: 256M
    ports:
    - 9091:9090
```

99. chmod +x build-dev-docker-images.sh
100. ./build-dev-docker-images.sh
101. docker images

```
[ec2-user@jenkins-server petclinic-microservices]$ docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
petclinic-prometheus-server   dev                 87782e3a813c        2 minutes ago       110MB
petclinic-grafana-server      dev                 b6a4218f00a1        2 minutes ago       245MB
petclinic-visits-service      dev                 a9c01f73eb08        2 minutes ago       368MB
petclinic-vets-service        dev                 c6d8785140e7        2 minutes ago       369MB
petclinic-hystrix-dashboard   dev                 1b3cfa3951ee        2 minutes ago       339MB
petclinic-discovery-server    dev                 f7f536fce025        2 minutes ago       352MB
petclinic-customers-service   dev                 e61a8e15ef00        2 minutes ago       368MB
petclinic-config-server       dev                 3d5de1d9b34a        2 minutes ago       335MB
petclinic-api-gateway         dev                 da229e3e018b        2 minutes ago       356MB
petclinic-admin-server        dev                 a2e62b23e58c        2 minutes ago       351MB
node                          14-alpine           51d926a5599d        3 days ago          116MB
openjdk                       11-jre              94321aa03ce0        3 days ago          286MB
prom/prometheus               v2.4.2              4149712a7a11        2 years ago         110MB
grafana/grafana               5.2.4               920eb69ade2a        2 years ago         245MB
```

102. « test-local-deployment.sh » dosyasi lusturun.

```
docker-compose -f docker-compose-local.yml up
```

103. chmod +x ./test-local-deployment.sh
104. ./test-local-deployment.sh

```
Creating tracing-server    ... done
Creating grafana-server    ... done
Creating prometheus-server ... done
Creating config-server     ... done
Creating discovery-server  ... done
Creating admin-server      ...
Creating vets-service      ...
Creating customers-service ...
Creating hystrix-dashboard ...
Creating admin-server      ... done
Creating vets-service      ... done
Creating customers-service ... done
Creating hystrix-dashboard ... done
Creating visits-service    ... done
24d48fa841034899ac91c6a1e939483a38bcc6c0): Error starting userland proxy: listen tcp 0.0.0.0:8080: bind: address already in use
```

105. git checkout dev
106. git branch feature/msp-9
107. git checkout feature/msp-9
108. ./spring-petclinic-customers-service/src/test/java/org/springframework/samples/petclinic/customers/model/ altina “PetTest.java” dosyasi olusturun.

```
      package org.springframework.samples.petclinic.customers.model;


import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.Test;

public class PetTest {
```

```
    @Test
    public void testGetName(){
        //Arrange
        Pet pet = new Pet();
        //Act
        pet.setName("Fluffy");
        //Assert
        assertEquals("Fluffy", pet.getName());
    }

    @Test
    public void testGetOwner(){
        //Arrange
        Pet pet = new Pet();
        Owner owner = new Owner();
        owner.setFirstName("Call");
        //Act
        pet.setOwner(owner);
        //Assert
        assertEquals("Call", pet.getOwner().getFirstName());
    }

    @Test
    public void testBirthDate(){
        //Arrange
        Pet pet = new Pet();
        Date bd = new Date();
        //Act
        pet.setBirthDate(bd);
        //Assert
        assertEquals(bd,pet.getBirthDate());
    }
}
```

109. git checkout dev
110. git branch feature/msp-9
111. git checkout feature/msp-9
112. Github hesabindan degisimleri merge edin.
113. git add .
114. git commit -m 'added 3 UTs for customer-service'
115. git push --set-upstream origin feature/msp-9
116. cd spring-petclinic-customers-service/
117. ../mvnw clean test

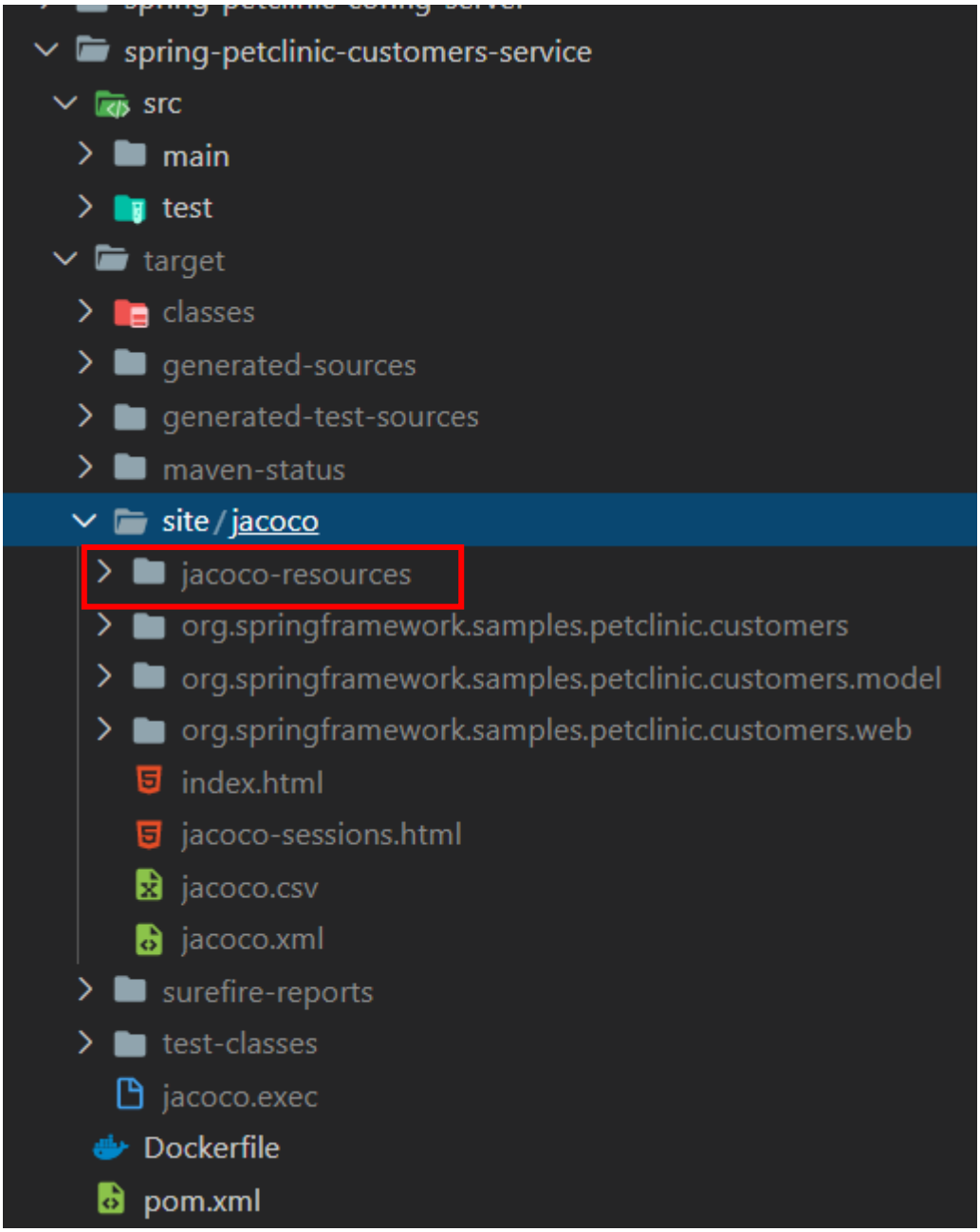
```
[INFO] Results:
[INFO]
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

118. Ana klasor altindaki pom dosyasinin icindeki plugins bolumunun sonuna asagidaki kodu yerlestirin.

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.2</version>
  <executions>
    <execution>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
    </execution>
    <!-- attached to Maven test phase -->
    <execution>
      <id>report</id>
      <phase>test</phase>
      <goals>
        <goal>report</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

119. git add .
120. git commit -m 'updated POM with Jacoco plugin'
121. git push
122. git checkout dev
123. git merge feature/msp-9
124. git push origin dev
125. Github hesabindan degisikleri merge edin.
126. cd petclinic-microservices/spring-petclinic-customers-service
127. ../mvnw test

```
NFO] Loading execution data file /home/ec2-user/petclinic-microservi
NFO] Analyzed bundle 'spring-petclinic-customers-service' with 9 cla
NFO] ------------------------------------------------------------------------
NFO] BUILD SUCCESS
NFO] ------------------------------------------------------------------------
NFO] Total time: 19.215 s
NFO] Finished at: 2020-12-19T18:32:23Z
```




Burada jococo calismis ve site/jacoco klasoru olusturdu.

128. cd petclinic-microservices/spring-petclinic-customers-service/target/site/jacoco
129. python -m SimpleHTTPServer # for python 2.7
130. python3 -m http.server # for python 3+

**spring-petclinic-customers-service**

| Element | Missed Instructions | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines | Missed | Methods | Missed | Classes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| org.springframework.samples.petclinic.customers.web | | 12% | | 1% | 69 | 79 | 40 | 58 | 35 | 45 | 3 | 5 |
| org.springframework.samples.petclinic.customers.model | | 45% | | 50% | 11 | 33 | 29 | 63 | 10 | 32 | 0 | 3 |
| org.springframework.samples.petclinic.customers | | 37% | | n/a | 1 | 2 | 2 | 3 | 1 | 2 | 0 | 1 |
| Total | 638 of 807 | 20% | 68 of 70 | 2% | 81 | 114 | 71 | 124 | 46 | 79 | 3 | 9 |

131. git checkout dev
132. git branch feature/msp-10 # uniq test icin brans olusturduk
133. git checkout feature/msp-10
134. cd petclinic-microservices/
135. mkdir selenium-jobs



test_owners_all_headless.py test_owners_register_headless.py test_veterinarians_headless.py

136.

137. test_veterinarians_headless.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from time import sleep
import os

# Set chrome options for working with headless mode (no screen)
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("headless")
chrome_options.add_argument("no-sandbox")
chrome_options.add_argument("disable-dev-shm-usage")

# Update webdriver instance of chrome-driver with adding chrome options
driver = webdriver.Chrome(options=chrome_options)

# Connect to the application
APP_IP = os.environ['MASTER_PUBLIC_IP']
url = "http://"+APP_IP.strip()+":8080/"
print(url)
driver.get(url)
vet_link = driver.find_element_by_link_text("VETERINARIANS")
vet_link.click()

# Verify that table loaded
sleep(1)
verify_table = WebDriverWait(driver,
10).until(EC.presence_of_element_located((By.TAG_NAME, "table")))

print("Table loaded")

driver.quit()
```

138. test_owners_register_headless.py

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from time import sleep
import random
import os
# Set chrome options for working with headless mode (no screen)
chrome_options = webdriver.ChromeOptions()
```

```
chrome_options.add_argument("headless")
chrome_options.add_argument("no-sandbox")
chrome_options.add_argument("disable-dev-shm-usage")

# Update webdriver instance of chrome-driver with adding chrome options
driver = webdriver.Chrome(options=chrome_options)

# Connect to the application
APP_IP = os.environ['MASTER_PUBLIC_IP']
url = "http://"+APP_IP.strip()+":8080/"
print(url)
driver.get(url)
owners_link = driver.find_element_by_link_text("OWNERS")
owners_link.click()
sleep(2)
all_link = driver.find_element_by_link_text("REGISTER")
all_link.click()
sleep(2)
# Register new Owner to Petclinic App
fn_field = driver.find_element_by_name('firstName')
fn = 'Callahan' + str(random.randint(0, 100))
fn_field.send_keys(fn)
sleep(1)
fn_field = driver.find_element_by_name('lastName')
fn_field.send_keys('Clarusway')
sleep(1)
fn_field = driver.find_element_by_name('address')
fn_field.send_keys('Ridge Corp. Street')
sleep(1)
fn_field = driver.find_element_by_name('city')
fn_field.send_keys('McLean')
sleep(1)
fn_field = driver.find_element_by_name('telephone')
fn_field.send_keys('+1230576803')
sleep(1)
fn_field.send_keys(Keys.ENTER)
sleep(1)
# Wait 2 second to get updated Owner List
sleep(2)
# Verify that new user is added to Owner List
if fn in driver.page_source:
   print(fn, 'is added and found in the Owners Table')
   print("Test Passed")
else:
   print(fn, 'is not found in the Owners Table')
   print("Test Failed")
driver.quit()
```

139. test_owners_all_headless.py

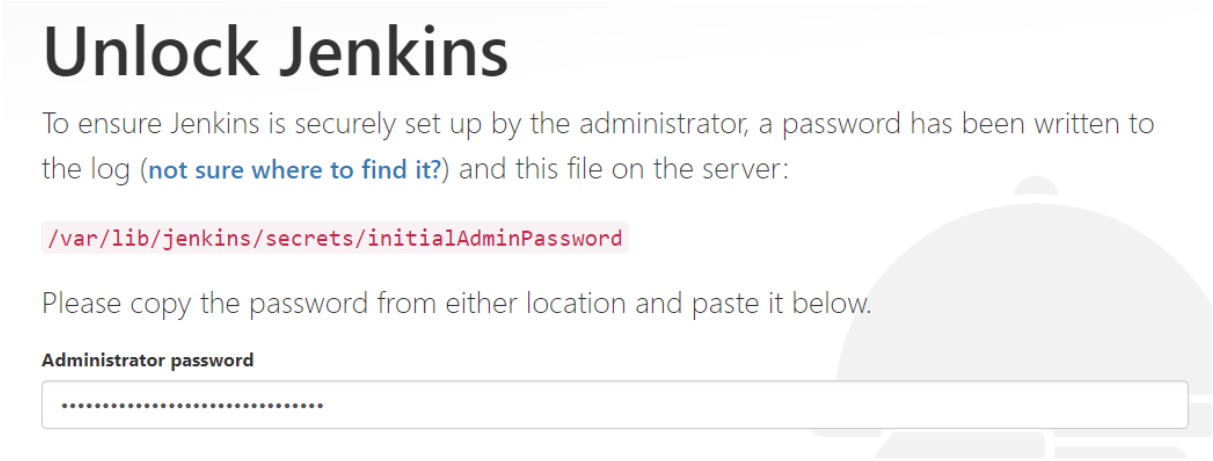
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from time import sleep
import os

# Set chrome options for working with headless mode (no screen)
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument("headless")
chrome_options.add_argument("no-sandbox")
chrome_options.add_argument("disable-dev-shm-usage")

# Update webdriver instance of chrome-driver with adding chrome options
driver = webdriver.Chrome(options=chrome_options)

# Connect to the application
APP_IP = os.environ['MASTER_PUBLIC_IP']
url = "http://"+APP_IP.strip()+":8080/"
print(url)
driver.get(url)
owners_link = driver.find_element_by_link_text("OWNERS")
owners_link.click()
sleep(2)
all_link = driver.find_element_by_link_text("ALL")
all_link.click()
sleep(2)

# Verify that table loaded
sleep(1)
verify_table = WebDriverWait(driver,
10).until(EC.presence_of_element_located((By.TAG_NAME, "table")))

print("Table loaded")

driver.quit()
```

140. git add .
141. git commit -m 'added selenium jobs written in python'
142. git push --set-upstream origin feature/msp-10
143. git checkout dev
144. git merge feature/msp-10
145. git push origin dev
146. Github hesanindan merge yapiniz.

III. Day

147. Template uzerinden ec2 ayaga kaldiracagiz. Amac ec2 icine jenkins kurmak.

148. git clone https://github.com/Gokay2705/petclinic-microservices.git #repoyu ec2 icine kopyaliyoruz.
149. Public IPv4 address :8080 adresinden jenkins aciniz.




150. sudo cat /var/lib/jenkins/secrets/initialAdminPassword # cikan sifreyi giriniz.
151. git checkout dev
152. git branch feature/msp-11
153. git checkout feature/msp-11
154. infrastructure klasoru altina “jenkins-server-cfn-template.yml” dosyasini olusturun.

```
AWSTemplateFormatVersion: 2010-09-09


Description: >
  This Cloudformation Template creates a Jenkins Server using JDK 11 on EC2 Instance.
  Jenkins Server is enabled with Git, Docker and Docker Compose,
  AWS CLI Version 2, Python 3, Ansible, and Boto3.
  Jenkins Server will run on Amazon Linux 2 EC2 Instance with
  custom security group allowing HTTP(80, 8080) and SSH (22) connections from anywhere.

Parameters:
  KeyPairName:
    Description: Enter the name of your Key Pair for SSH connections.
    Type: AWS::EC2::KeyPair::KeyName
    ConstraintDescription: Must one of the existing EC2 KeyPair
Resources:
  EmpoweringRoleforJenkinsServer:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service:
              - ec2.amazonaws.com
            Action:
```

```
          - 'sts:AssumeRole'
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryFullAccess
      - arn:aws:iam::aws:policy/AWSCloudFormationFullAccess
      - arn:aws:iam::aws:policy/AdministratorAccess
JenkinsServerEC2Profile:
  Type: "AWS::IAM::InstanceProfile"
  Properties:
    Roles: #required
      - !Ref EmpoweringRoleforJenkinsServer
JenkinsServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable SSH and HTTP for Jenkins Server
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 8080
        ToPort: 8080
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0
JenkinsServer:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: ami-0947d2ba12ee1ff75
    InstanceType: t2.medium
    KeyName: !Ref KeyPairName
    IamInstanceProfile: !Ref JenkinsServerEC2Profile
    SecurityGroupIds:
      - !GetAtt JenkinsServerSecurityGroup.GroupId
    Tags:
      - Key: Name
        Value: !Sub Jenkins Server of ${AWS::StackName}
      - Key: server
        Value: jenkins
    UserData:
      Fn::Base64: |
        #! /bin/bash
        # update os
        yum update -y
        # set server hostname as jenkins-server
        hostnamectl set-hostname jenkins-server
```

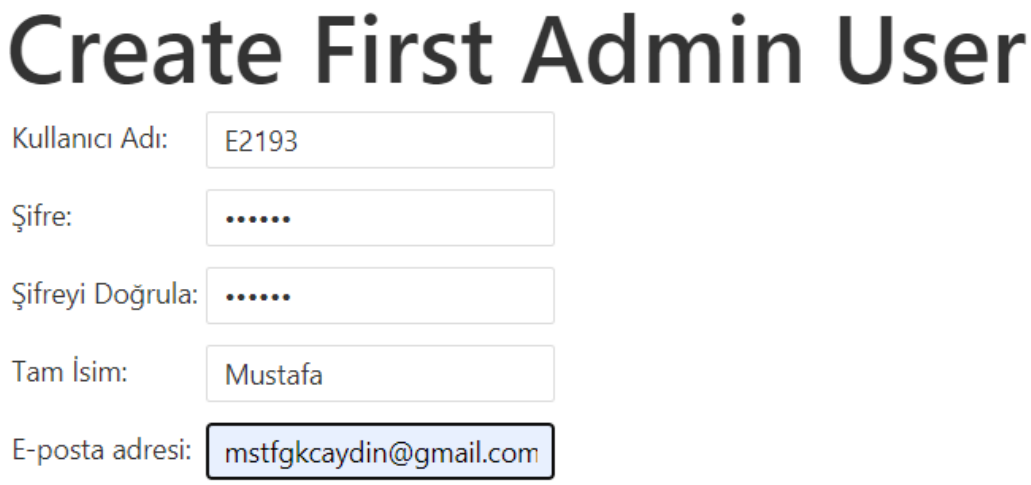
```
        # install git
        yum install git -y
        # install java 11
        yum install java-11-amazon-corretto -y
        # install jenkins
        wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat/jenkins.repo
        rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
        yum install jenkins -y
        systemctl start jenkins
        systemctl enable jenkins
        # install docker
        amazon-linux-extras install docker -y
        systemctl start docker
        systemctl enable docker
        usermod -a -G docker ec2-user
        usermod -a -G docker jenkins
        # configure docker as cloud agent for jenkins
        cp /lib/systemd/system/docker.service /lib/systemd/system/docker.service.bak
        sed -i 's/^ExecStart=.*/ExecStart=\/usr\/bin\/dockerd -H tcp:\/\/127.0.0.1:2375 -H
unix:\/\/\/var\/run\/docker.sock/g' /lib/systemd/system/docker.service
        systemctl daemon-reload
        systemctl restart docker
        systemctl restart jenkins
        # install docker compose
        curl -L "https://github.com/docker/compose/releases/download/1.26.2/docker-compose-
$(uname -s)-$(uname -m)" \
        -o /usr/local/bin/docker-compose
        chmod +x /usr/local/bin/docker-compose
        # uninstall aws cli version 1
        rm -rf /bin/aws
        # install aws cli version 2
        curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
        unzip awscliv2.zip
        ./aws/install
        # install python 3
        yum install python3 -y
        # install ansible
        pip3 install ansible
        # install boto3
        pip3 install boto3

Outputs:
  JenkinsDNS:
    Description: Jenkins Server DNS Name
    Value: !Sub
      - ${PublicAddress}
      - PublicAddress: !GetAtt JenkinsServer.PublicDnsName
  JenkinsURL:
```

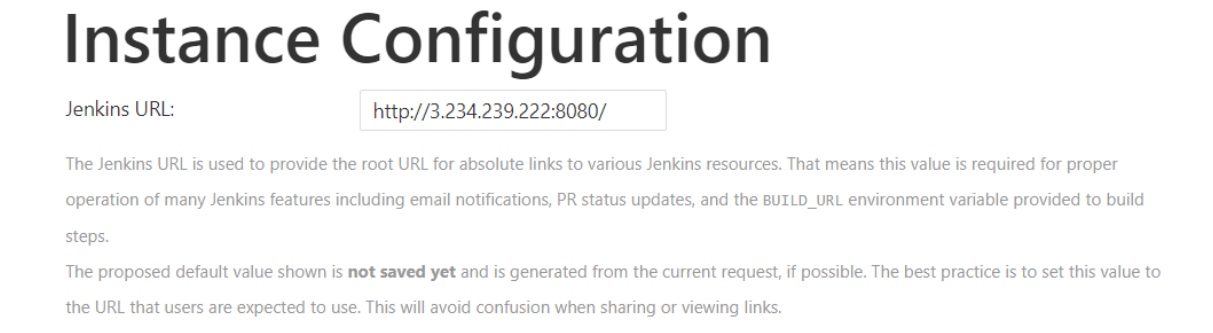
Description: Jenkins Server URL
Value: !Sub
- http://${PublicAddress}:8080
- PublicAddress: !GetAtt JenkinsServer.PublicDnsName

155. git add .
156. git config --global user.email mstfgkcaydin@gmail.com
157. git config --global user.name "Gokay2705"
158. git config --global credential.helper store
159. git commit -m 'added jenkins server cfn template'
160. git push --set-upstream origin feature/msp-11
161. git checkout dev
162. git merge feature/msp-11
163. git push origin dev
164. git branch -a

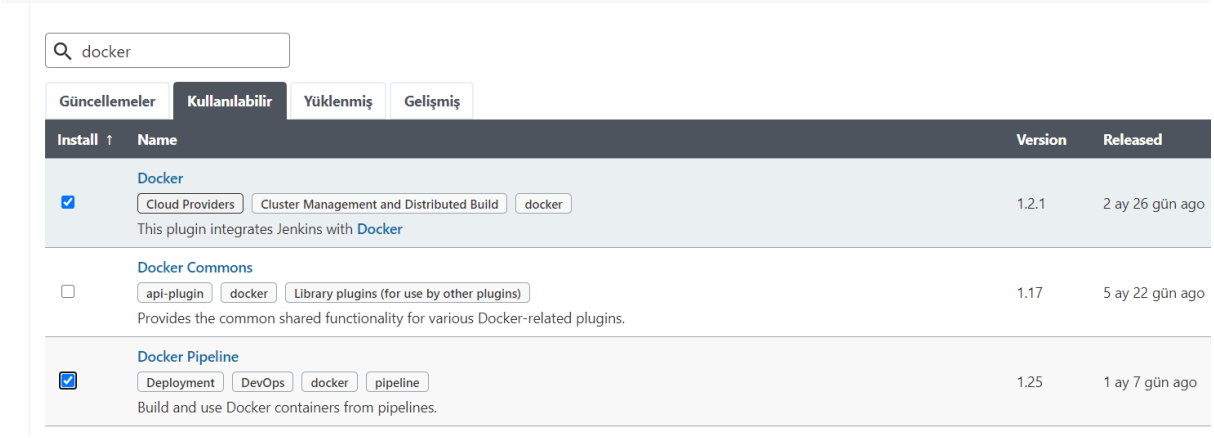
```
[ec2-user@jenkins-server petclinic-microservices]$ git branch -a
* dev
  master
  remotes/origin/HEAD -> origin/master
  remotes/origin/dev
  remotes/origin/feature/msp-10
  remotes/origin/feature/msp-4
  remotes/origin/feature/msp-5
  remotes/origin/feature/msp-6
  remotes/origin/feature/msp-7
  remotes/origin/feature/msp-9
  remotes/origin/master
  remotes/origin/release
[ec2-user@jenkins-server petclinic-microservices]$
```

165. Jenkins baglandiktan sonra :






166. Docker ve Docker Pipline Jenkins plugins'den yukle.

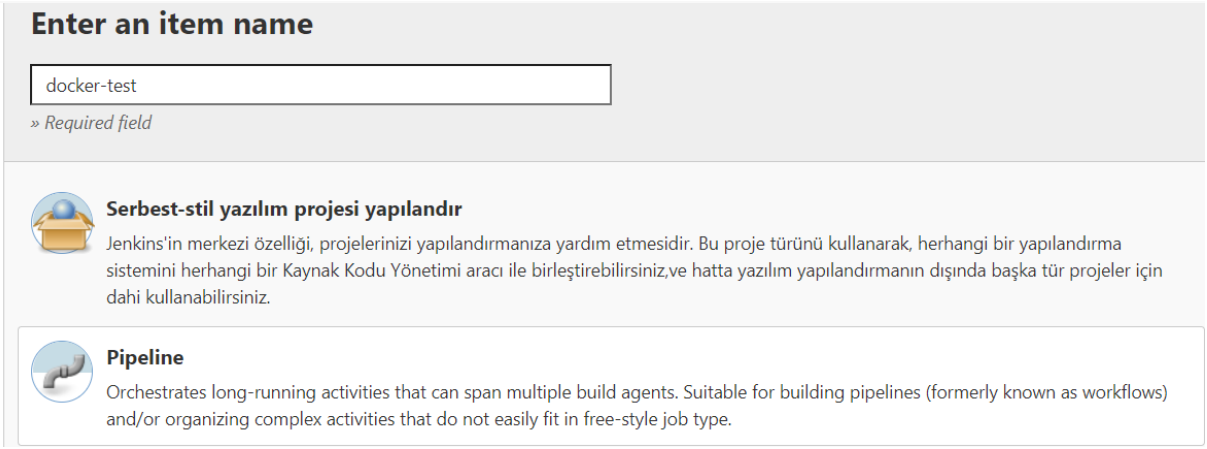



167. Github integration plugin yukleyin.
168. Jacoco plugin yukleyin.
169. Manage jenkins/manage nods and clouds/configure clouds tiklayin.


Test connection tikla.


170. Bu islemi Jenkins dockerdan gelen komutlari buradan dinleyecek. Read me 893 satirda belittik.

171. Docker-test piple olusturalim.




172. Acilan menuye asagidaki kodu girin.

```
pipeline {
    agent {
        docker { image 'node:14-alpine' }
    }
    stages {
        stage('Test') {
            steps {
                sh 'node --version'
            }
        }
    }
}
```

173. New items/master-test adini girin, pipeline secin.

```
pipeline {
  agent {
    label 'master'
  }
  stages {
    stage('Test') {
      steps {
        sh 'docker run node:14-alpine  node --version'
      }
    }
  }
}
```

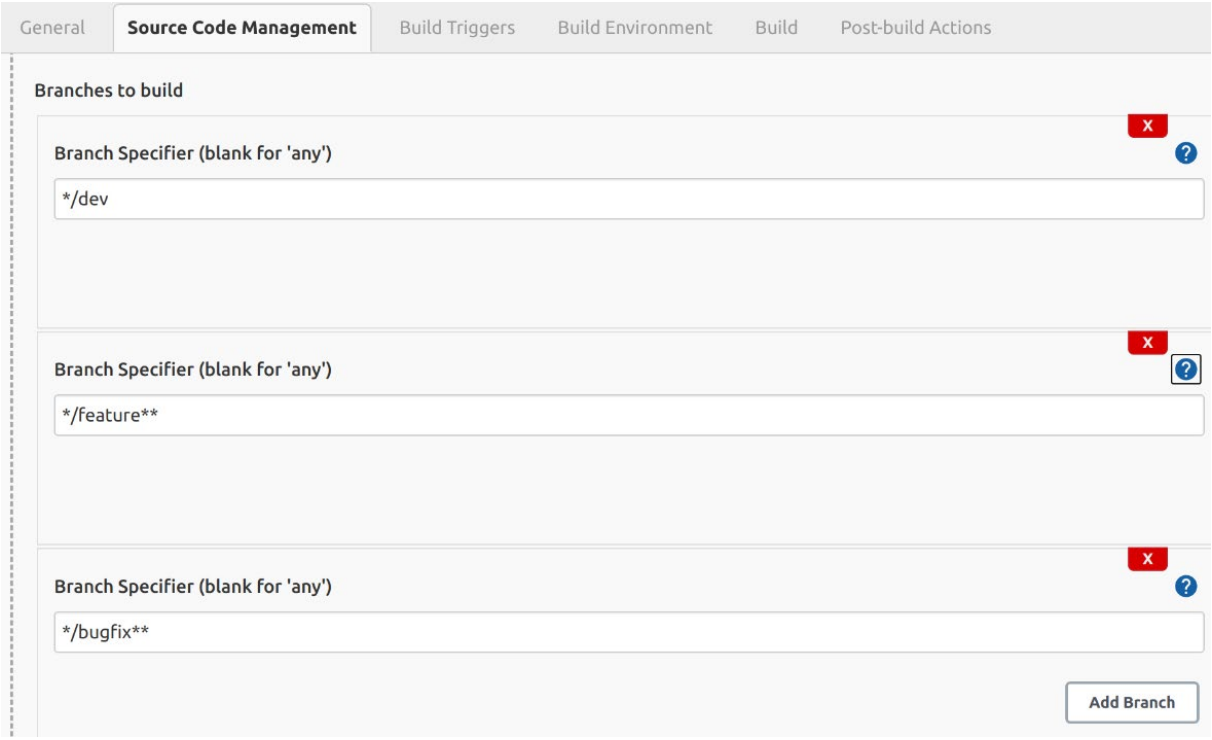
174. git checkout
175. git checkout dev
176. git branch feature/msp-13
177. git checkout feature/msp-13
178. mkdir Jenkins # ana klasor altinda olustur.
179. Jenkins server uzerinden New item/ petclinic-ci-job adini verin/ Freestyle project secin.

180. Git hub project secenegini secin.
https://github.com/Gokay2705/petclinic-microservices


181. Source Code Management
https://github.com/Gokay2705/petclinic-microservices.git


182. Brans olarak dev yazin.


183. Yukaridaki gibi */feature** ve */bugfix** branslarini da ekleyin.



184. Buidnow deyin.

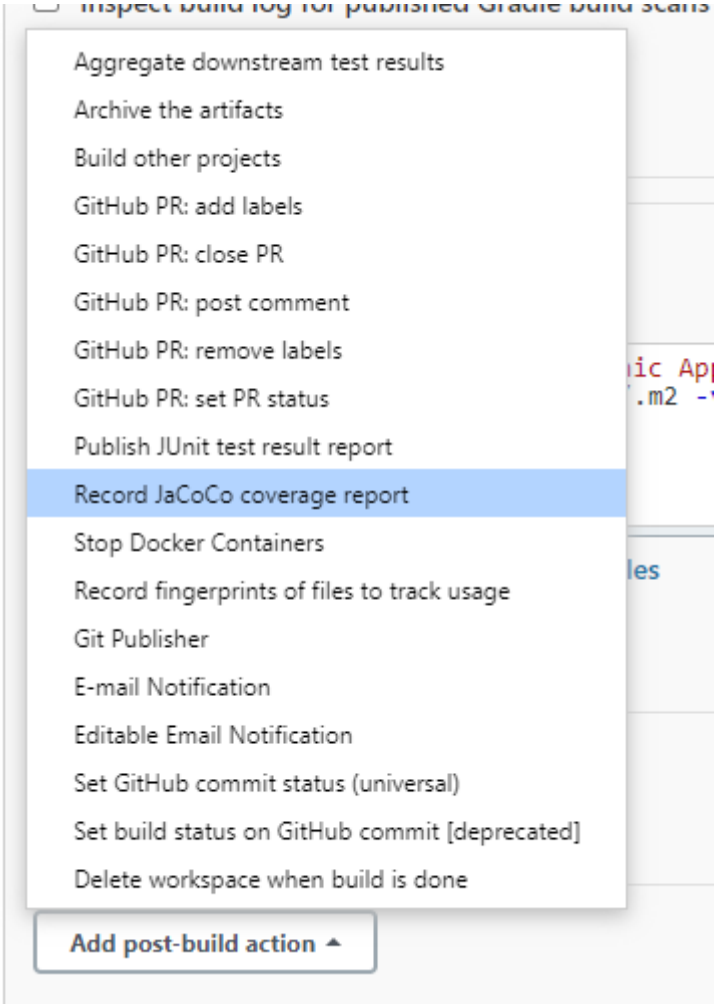
## Console Output

```
Started by an SCM change
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/petclinic-ci-job
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/Gokay2705/petclinic-microservices.git # timeout=10
Fetching upstream changes from https://github.com/Gokay2705/petclinic-microservices.git
 > git --version # timeout=10
 > git --version # 'git version 2.23.3'
 > git fetch --tags --force --progress -- https://github.com/Gokay2705/petclinic-microservices.git +refs/heads/*:refs/remotes/origin/* # timeout=10
Seen branch in repository origin/dev
Seen branch in repository origin/feature/msp-10
Seen branch in repository origin/feature/msp-4
Seen branch in repository origin/feature/msp-5
Seen branch in repository origin/feature/msp-6
Seen branch in repository origin/feature/msp-7
Seen branch in repository origin/feature/msp-9
Seen branch in repository origin/master
Seen branch in repository origin/release
Seen 9 remote branches
 > git show-ref --tags -d # timeout=10
Checking out Revision 3c9cc229eabc283b397345a7b155c4ec53ec065a (origin/dev)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 3c9cc229eabc283b397345a7b155c4ec53ec065a # timeout=10
Commit message: "added jenkins server cfn template"
First time build. Skipping changelog.
Finished: SUCCESS
```

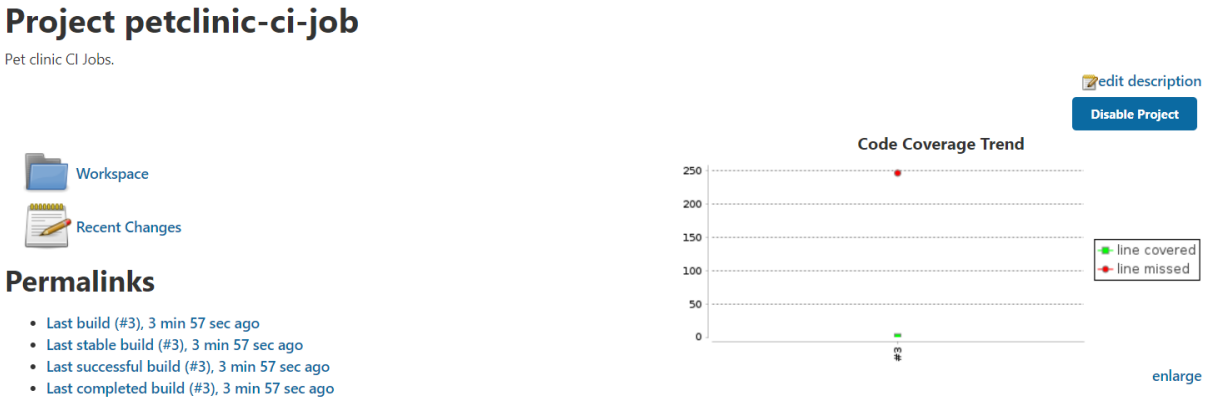
185. Buid triger ve Buid Envirenment asagidaki secenekleri secip Execute shell'e asagidaki kodu girin.

echo 'Running Unit Tests on Petclinic Application'
docker run --rm -v $HOME/.m2:/root/.m2 -v `pwd`:/app -w /app maven:3.6-openjdk-11 mvn clean test

186. Post-build Actions JaCoco kayitlarini olusturmayi secin.

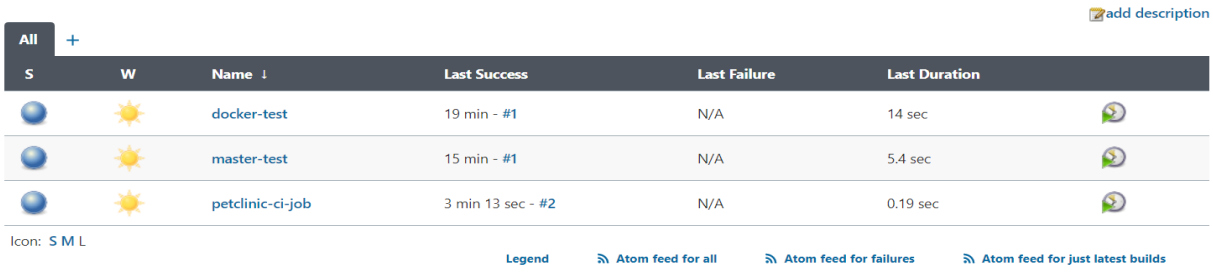



187. Buidnow.



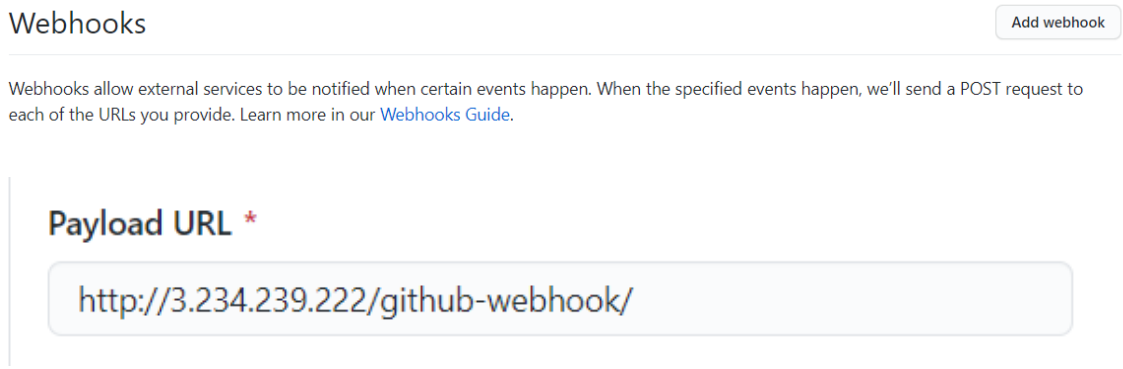

188. Server üzerinden yaptigimiz test islemlerini simdi java containerine yapiyoruz.

189. Genel durum.



190. Webhook ekleyin.

```
191. http://[jenkins-server-hostname]:8080/github-webhook/
```




Public IPv4:8080 address adresini giriyoruz.

192. Git branch

```
[ec2-user@jenkins-server petclinic-microservices]$ git branch
  dev
* feature/msp-13
  master
```

193. Jenkins klasoru icine" jenkins-petclinic-ci-job.sh" dosyasini olusturun.

```
echo 'Running Unit Tests on Petclinic Application'
docker run --rm -v $HOME/.m2:/root/.m2 -v `pwd`:/app -w /app maven:3.6-
openjdk-11 mvn clean test
```

194. git add .
195. git commit -m 'added Jenkins Job for CI pipeline'
196. git push --set-upstream origin feature/msp-13
197. Github icindeki webhook sayfasina girelim.


198. git checkout dev
199. git merge feature/msp-13
200. git push origin dev

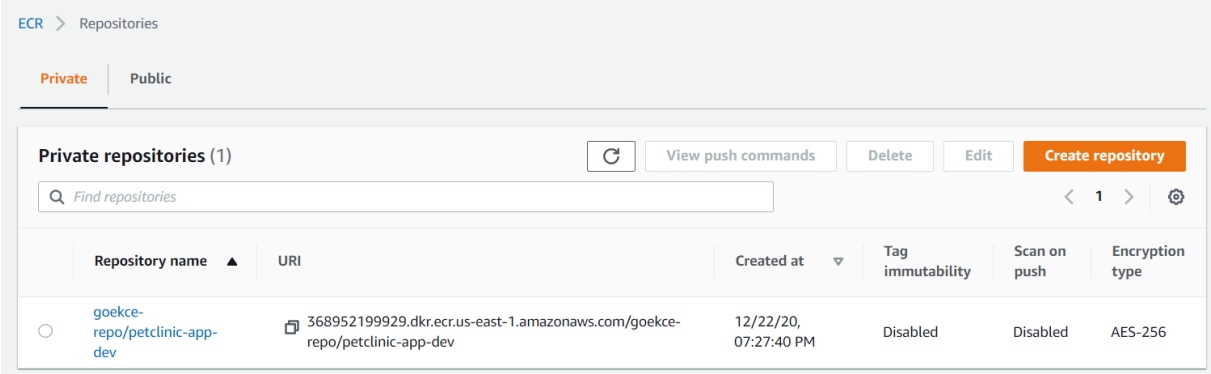
4.DAY

201. Jenkins'de `create-ecr-docker-registry-for-dev` adinda free style new item olusturun.
202. Build kisminda execute shell seceneginin secip, asagidaki kodu girin.

```
PATH="$PATH:/usr/local/bin"
APP_REPO_NAME="clarusway-repo/petclinic-app-dev"
AWS_REGION="us-east-1"
```

aws ecr create-repository \
--repository-name ${APP_REPO_NAME} \
--image-scanning-configuration scanOnPush=false \
--image-tag-mutability MUTABLE \
--region ${AWS_REGION}

203. Build now secin.
204. Yukaridaki kod ile aws uzerinden us-east-1'de repo olusturacagiz.




205. git checkout dev
206. git branch feature/msp-15
207. git checkout feature/msp-15
208. git branch

```
[ec2-user@jenkins-server petclinic-microservices]$ git branch
  dev
  feature/msp-13
* feature/msp-15
  feature/msp-9
  master
```

209. infrastructure klasoru altinda "create-ecr-docker-registry-for-dev.sh" isimli dosya olusturun.
210. Asagidaki kodu girin.
211.

```
PATH="$PATH:/usr/local/bin"
APP_REPO_NAME="goekce-repo/petclinic-app-dev"
AWS_REGION="us-east-1"

aws ecr create-repository \
  --repository-name ${APP_REPO_NAME} \
  --image-scanning-configuration scanOnPush=false \
  --image-tag-mutability MUTABLE \
  --region ${AWS_REGION}
```

212. git add .
213. git commit -m 'added script for creating ECR registry for dev'
214. git push --set-upstream origin feature/msp-15


Burada petclinic-ci-job triger edecek.

215. git checkout dev
216. git branch feature/msp-16
217. git checkout feature/msp-16
218. Fuctional test asamasindayiz.

# Pipelines to be Configured

| Name | Branch | Trigger | Environment / Test Type | Tools |
|---|---|---|---|---|
| petclinic-ci-job | dev<br>feature**<br>bugfix** | **Webhook**<br>on each commit | Unit Test | jenkins, maven, git, github, jacoco |
| petclinic-nightly | dev | **Cronjob**<br>every night 11.59pm | Functional Tests | jenkins, git, github, docker, docker-compose, docker swarm, ansible, maven, selenium with python, bash scripting, aws cli / ecr / cloudformation |
| petclinic-weekly | release | **Cronjob**<br>every sunday 11.59pm | Manual QA | jenkins, git, github, docker, docker-compose, docker swarm, ansible, maven, bash scripting, aws cli / ecr / cloudformation |
| petclinic-staging | release | **Cronjob**<br>every sunday 11.59pm | Staging Env. | jenkins, git, github, docker, rancher, kubernetes, maven, bash scripting, aws cli / ecr / cloudformation |

219. Docker Swarm Altyapısı için 3 Yönetici, 2 İşçi Örneğinden oluşan bir Cloudformation şablonu hazırlayın ve bunu "altyapı" klasörünün altına "docker-swarm-altyapı-cfn-template.yml" olarak kaydedin.

```
AWSTemplateFormatVersion: 2010-09-09

Description: >
  This Cloudformation Template creates an infrastructure for Docker Swar
m
  with five EC2 Instances with Amazon Linux 2. Instances are configured
  with custom security group allowing SSH (22), HTTP (80) UDP (4789, 794
6),
  and TCP(2377, 7946, 8080) connections from anywhere.
  User needs to select appropriate key name when launching the template.

Parameters:
  KeyPairName:
    Description: Enter the name of your Key Pair for SSH connections.
    Type: AWS::EC2::KeyPair::KeyName
```

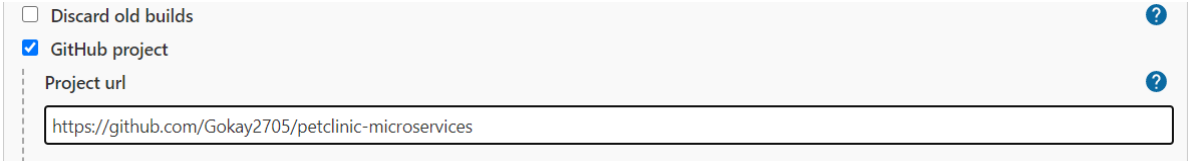
```
    ConstraintDescription: Must one of the existing EC2 KeyPair

Resources:
  RoleEnablingEC2forECR:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service:
              - ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryFullAccess
  EC2Profile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles: #required
        - !Ref RoleEnablingEC2forECR
  DockerMachinesSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable SSH and HTTP for Docker Machines
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 2377
          ToPort: 2377
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 7946
          ToPort: 7946
          CidrIp: 0.0.0.0/0
        - IpProtocol: udp
          FromPort: 7946
          ToPort: 7946
          CidrIp: 0.0.0.0/0
        - IpProtocol: udp
          FromPort: 4789
          ToPort: 4789
```

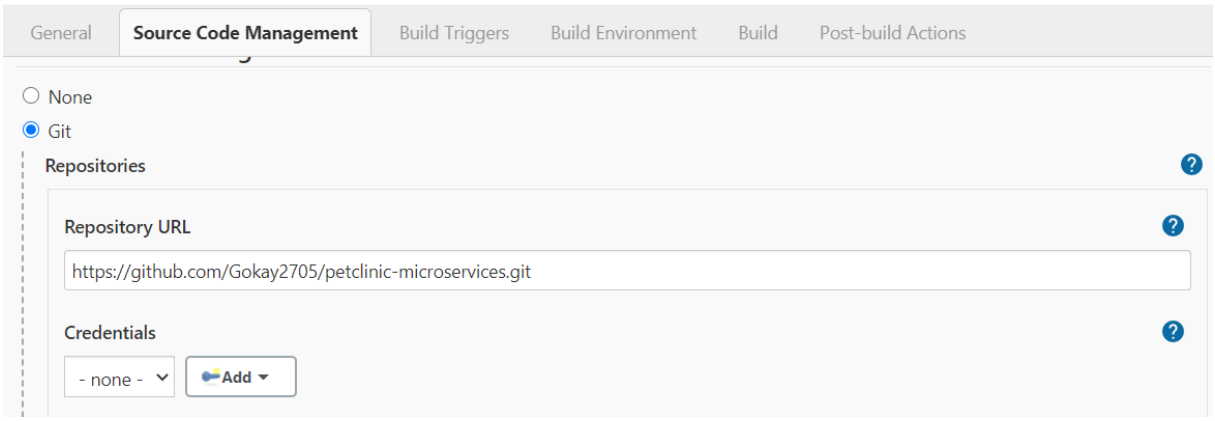
```
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 8080
        ToPort: 8080
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 8088
        ToPort: 8088
        CidrIp: 0.0.0.0/0
DockerMachineLT:
  Type: "AWS::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateData:
      ImageId: ami-0947d2ba12ee1ff75
      InstanceType: t2.medium
      KeyName: !Ref KeyPairName
      IamInstanceProfile:
        Arn: !GetAtt EC2Profile.Arn
      SecurityGroupIds:
        - !GetAtt DockerMachinesSecurityGroup.GroupId
      TagSpecifications:
        - ResourceType: instance
          Tags:
            - Key: app-stack-name
              Value: !Sub ${AWS::StackName}
            - Key: environment
              Value: dev
DockerInstance1:
  Type: AWS::EC2::Instance
  DependsOn:
      - "DockerInstance2"
  Properties:
    LaunchTemplate:
      LaunchTemplateId: !Ref DockerMachineLT
      Version: !GetAtt DockerMachineLT.LatestVersionNumber
    Tags:
      - Key: server
        Value: docker-instance-1
      - Key: swarm-role
        Value: grand-master
      - Key: Name
        Value: !Sub ${AWS::StackName} Docker Machine 1st
DockerInstance2:
  Type: AWS::EC2::Instance
  Properties:
    LaunchTemplate:
      LaunchTemplateId: !Ref DockerMachineLT
      Version: !GetAtt DockerMachineLT.LatestVersionNumber
    Tags:
```

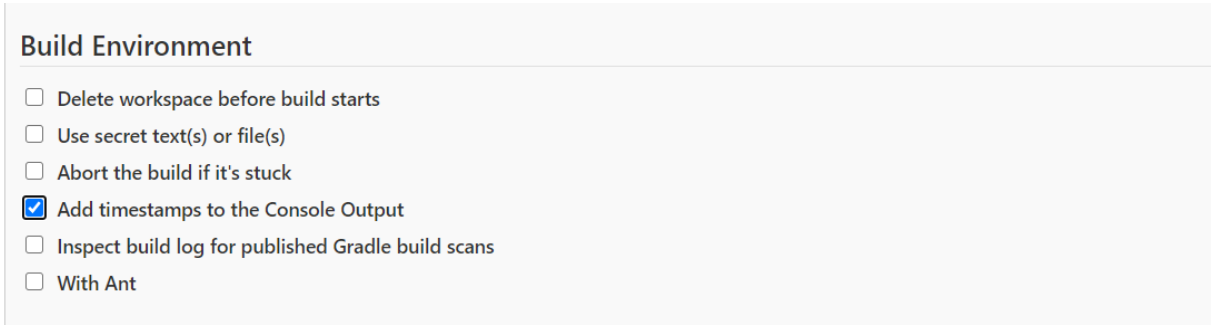
```
        - Key: server
          Value: docker-instance-2
        - Key: swarm-role
          Value: manager
        - Key: Name
          Value: !Sub ${AWS::StackName} Docker Machine 2nd
  DockerInstance3:
    Type: AWS::EC2::Instance
    Properties:
      LaunchTemplate:
        LaunchTemplateId: !Ref DockerMachineLT
        Version: !GetAtt DockerMachineLT.LatestVersionNumber
      Tags:
        - Key: server
          Value: docker-instance-3
        - Key: swarm-role
          Value: manager
        - Key: Name
          Value: !Sub ${AWS::StackName} Docker Machine 3rd
  DockerInstance4:
    Type: AWS::EC2::Instance
    Properties:
      LaunchTemplate:
        LaunchTemplateId: !Ref DockerMachineLT
        Version: !GetAtt DockerMachineLT.LatestVersionNumber
      Tags:
        - Key: server
          Value: docker-instance-4
        - Key: swarm-role
          Value: worker
        - Key: Name
          Value: !Sub ${AWS::StackName} Docker Machine 4th
  DockerInstance5:
    Type: AWS::EC2::Instance
    Properties:
      LaunchTemplate:
        LaunchTemplateId: !Ref DockerMachineLT
        Version: !GetAtt DockerMachineLT.LatestVersionNumber
      Tags:
        - Key: server
          Value: docker-instance-5
        - Key: swarm-role
          Value: worker
        - Key: Name
          Value: !Sub ${AWS::StackName} Docker Machine 5th
Outputs:
  1stDockerInstanceDNSName:
    Description: 1st Docker Instance DNS Name
    Value: !Sub
```

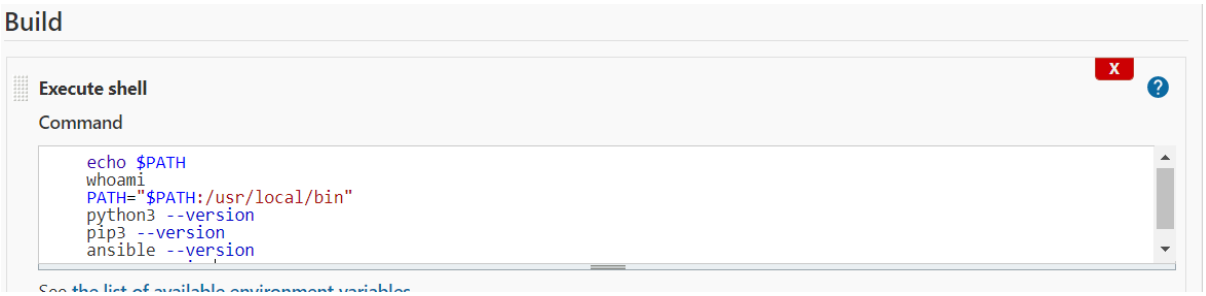
```
      - ${PublicAddress}
      - PublicAddress: !GetAtt DockerInstance1.PublicDnsName
  2ndDockerInstanceDNSName:
    Description: 2nd Docker Instance DNS Name
    Value: !Sub
      - ${PublicAddress}
      - PublicAddress: !GetAtt DockerInstance2.PublicDnsName
  3rdDockerInstanceDNSName:
    Description: 3rd Docker Instance DNS Name
    Value: !Sub
      - ${PublicAddress}
      - PublicAddress: !GetAtt DockerInstance3.PublicDnsName
  4thDockerInstanceDNSName:
    Description: 4th Docker Instance DNS Name
    Value: !Sub
      - ${PublicAddress}
      - PublicAddress: !GetAtt DockerInstance4.PublicDnsName
  5thDockerInstanceDNSName:
    Description: 5th Docker Instance DNS Name
    Value: !Sub
      - ${PublicAddress}
      - PublicAddress: !GetAtt DockerInstance5.PublicDnsName
```

220. git add .
221. git commit -m 'added cloudformation template for Docker Swarm infrastructure'
222. git push --set-upstream origin feature/msp-16
223. Bir Jenkins İşi oluşturun ve bunu manuel olarak "dev" için qa Otomasyon Altyapısı oluşturan "bash" komut dosyalarını test etmek için "test-oluşturma-qa-otomasyon-altyapısı" olarak adlandırın.




echo $PATH
whoami
PATH="$PATH:/usr/local/bin"
python3 --version
pip3 --version
ansible --version
aws –version

224. Build now.

225. Sonuc basarili ise shell kismini guncelleyin. Burdaki amac ihtiyac duydugum tollar calisiyor mu? Onu tespit etmek idi.Simdiki amac test creating key pair test etmek.

PATH="$PATH:/usr/local/bin"
CFN_KEYPAIR="goekce-ansible-test-dev.key"
AWS_REGION="us-east-1"
aws ec2 create-key-pair --region ${AWS_REGION} --key-name ${CFN_KEYPAIR} --query "KeyMaterial" --output text > ${CFN_KEYPAIR}
chmod 400 ${CFN_KEYPAIR}

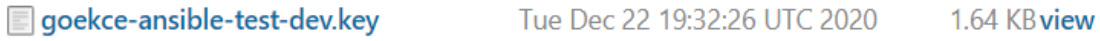
226. Docker Swarm infrastructure ile AWS Cloudformation oluturulmasini control etmek icin shelli guncelleyin.

```
PATH="$PATH:/usr/local/bin"
APP_NAME="Petclinic"
APP_STACK_NAME="Goekce-$APP_NAME-App-${BUILD_NUMBER}"
CFN_KEYPAIR="goekce-ansible-test-dev.key"
CFN_TEMPLATE="./infrastructure/docker-swarm-infrastructure-cfn-template.yml"
AWS_REGION="us-east-1"
aws cloudformation create-stack --region ${AWS_REGION} --stack-name ${APP_STACK_NAME} --capabilities CAPABILITY_IAM --template-body file://${CFN_TEMPLATE} --parameters ParameterKey=KeyPairName,ParameterValue=${CFN_KEYPAIR}
```

| Name | Instance ID | Instance state | Instance type | Status check |
|---|---|---|---|---|
| Jenkins Server of petclinic-jenkins-server | i-0d1a8c412ba9ca5c0 | Running | t2.medium | 2/2 checks ... |
| Goekce-Petclinic-App-3 Docker Machine 4th | i-06b330300c60863df | Running | t2.medium | 2/2 checks ... |
| Goekce-Petclinic-App-3 Docker Machine 2nd | i-00e413df378e21dba | Running | t2.medium | 2/2 checks ... |
| Goekce-Petclinic-App-3 Docker Machine 3rd | i-0fe0286df54c66b60 | Running | t2.medium | 2/2 checks ... |
| Goekce-Petclinic-App-3 Docker Machine 1st | i-00a3094b3f00fa648 | Running | t2.medium | 2/2 checks ... |
| Goekce-Petclinic-App-3 Docker Machine 5th | i-02f38c8b4699a0c0f | Running | t2.medium | 2/2 checks ... |

227. Kod calitiktan sonra asagidaki ciktilari aliriz.

Jenkis/workspace




AWS

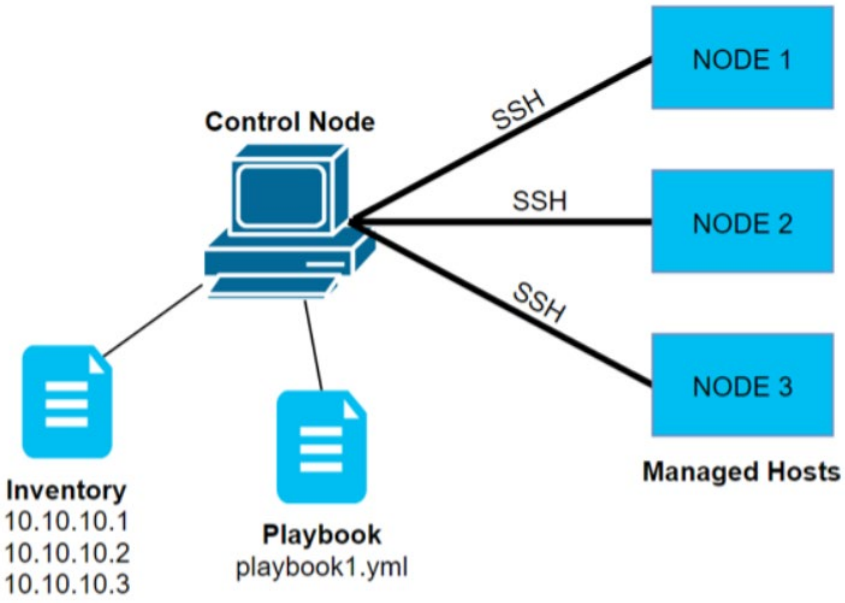

228. Docker ve SHH baglantisini control etmek icin:

```
CFN_KEYPAIR="goekce-ansible-test-dev.key"
ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i ${WORKSPACE}/${CFN_KEYPAIR} ec2-user@Private IPv4 addresses hostname
```

```
1:18:36 [test-creating-qa-automation-infrastructure] $ /bin/sh -xe /tmp/jenkins16819538465223416865.sh
1:18:36 + CFN_KEYPAIR=goekce-ansible-test-dev.key
1:18:36 + ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -i /var/lib/jenkins/workspace/test-creating-qa-automation-infrastructure/goekce-
nsible-test-dev.key ec2-user@172.31.40.96 hostname
1:18:36 Warning: Permanently added '172.31.40.96' (ECDSA) to the list of known hosts.
1:18:37 ip-172-31-40-96.ec2.internal
1:18:37 Finished: SUCCESS
```

229. Ana klasor altina `ansible/inventory` klasoru olusturup icine `hosts.ini` dosyasi olusturun. Burada private IP'leri yaziniz.

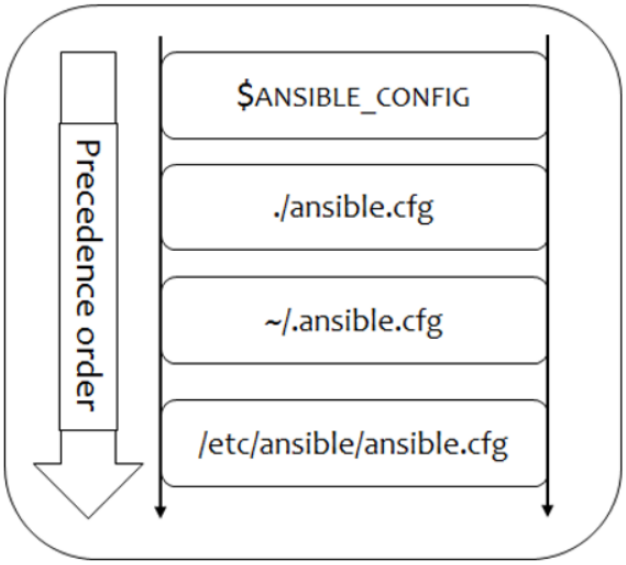
172.31.40.96 ansible_user=ec2-user
172.31.44.111 ansible_user=ec2-user
172.31.38.53 ansible_user=ec2-user
172.31.47.12 ansible_user=ec2-user
172.31.37.171 ansible_user=ec2-user




230. git add .
231. git commit -m 'added ansible static inventory host.ini for testing'
232. git push
233. Jenkins icindeki `test-creating-qa-automation-infrastructure` itemin shel kismini ec2'lere ping atttigini control etmek icin guncelleyelim.

```
PATH="$PATH:/usr/local/bin"
CFN_KEYPAIR="goekce-ansible-test-dev.key"
export ANSIBLE_INVENTORY="${WORKSPACE}/ansible/inventory/hosts.ini"
export ANSIBLE_PRIVATE_KEY_FILE="${WORKSPACE}/${CFN_KEYPAIR}"
export ANSIBLE_HOST_KEY_CHECKING=False
ansible all -m ping
```

```
21:37:22 172.31.38.53 | SUCCESS => {
21:37:22     "ansible_facts": {
21:37:22         "discovered_interpreter_python": "/usr/bin/python"
21:37:22     },
21:37:22     "changed": false,
21:37:22     "ping": "pong"
21:37:22 }
21:37:22 Finished: SUCCESS
```

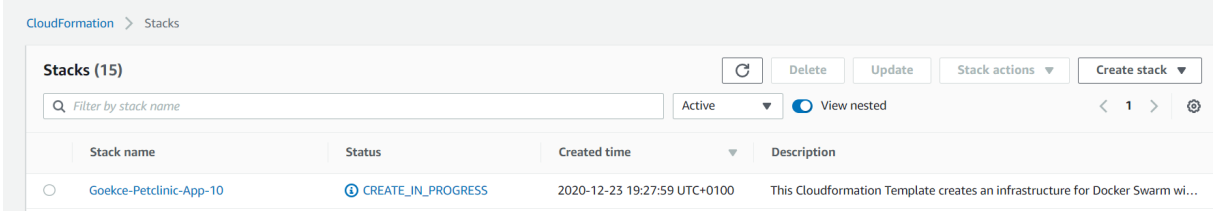



# 5. DAY

234. EC2 sildigimiz icin tekrar EC2 ayaga kaldiracagiz.

Jenkins server uzerinden “test-creating-qa-automation-infrastructure” piplineda Execute shell kismina asagidaki kodu girelim.

```
PATH="$PATH:/usr/local/bin"
APP_NAME="Petclinic"
APP_STACK_NAME="Goekce-$APP_NAME-App-${BUILD_NUMBER}"
CFN_KEYPAIR="goekce-ansible-test-dev.key"
CFN_TEMPLATE="./infrastructure/docker-swarm-infrastructure-cfn-template.yml"
AWS_REGION="us-east-1"
aws cloudformation create-stack --region ${AWS_REGION} --stack-name ${APP_STACK_NAME} --capabilities CAPABILITY_IAM --template-body file://${CFN_TEMPLATE} --parameters ParameterKey=KeyPairName,ParameterValue=${CFN_KEYPAIR}
```




235. Docker makinelerinin özel IP adreslerini kullanarak Ansible için "ansible / envanter" klasörü altında "dev_stack_dynamic_inventory_aws_ec2.yaml" adlı dinamik envanter dosyası hazırlayın.

```
plugin: aws_ec2
regions:
 - "us-east-1"
filters:
 tag:app-stack-name: APP_STACK_NAME
 tag:environment: dev
keyed_groups:
 - key: tags['app-stack-name']
   prefix: 'app_stack_'
   separator: ''
 - key: tags['swarm-role']
   prefix: 'role_'
   separator: ''
 - key: tags['environment']
   prefix: 'env_'
   separator: ''
 - key: tags['server']
   separator: ''
hostnames:
 - "private-ip-address"
compose:
 ansible_user: "'ec2-user'"
```

236. Ansible için "dev_stack_swarm_grand master aws_ec2.yml" adıyla dinamik envanter dosyası ve Docker makinelerinin özel IP adreslerini kullanarak "ansible /inventory" klasörü hazırlayın.

```
plugin: aws_ec2
regions:
 - "us-east-1"
filters:
 tag:app-stack-name: APP_STACK_NAME
 tag:environment: dev
 tag:swarm-role: grand-master
hostnames:
 - "private-ip-address"
compose:
 ansible_user: "'ec2-user'"
```

237. Ayni klasor icinde “dev_stack_swarm_managers_aws_ec2.yaml” dosyasini olusturalim.

```
plugin: aws_ec2
regions:
```

```
 - "us-east-1"
filters:
 tag:app-stack-name: APP_STACK_NAME
 tag:environment: dev
 tag:swarm-role: manager
hostnames:
 - "private-ip-address"
compose:
 ansible_user: "'ec2-user'"
```

238. Ayni klasor icinde “dev_stack_swarm_workers_aws_ec2.yaml” dosyasini olusturalim.

```
plugin: aws_ec2
regions:
 - "us-east-1"
filters:
 tag:app-stack-name: APP_STACK_NAME
 tag:environment: dev
 tag:swarm-role: worker
hostnames:
 - "private-ip-address"
compose:
 ansible_user: "'ec2-user'"
```

239. Feature/msp-16’da olalim.

```
[ec2-user@jenkins-server ~]$ cd petclinic-microservices/
[ec2-user@jenkins-server petclinic-microservices]$ git branch
  dev
  feature/msp-13
  feature/msp-15
* feature/msp-16
  feature/msp-9
  master
```

240. git add .
241. git commit -m 'added ansible dynamic inventory files for dev environment'
242. git push
243. Shel mismini guncelleyelim.

```
APP_NAME="Petclinic"
CFN_KEYPAIR="call-ansible-test-dev.key"
PATH="$PATH:/usr/local/bin"
export ANSIBLE_PRIVATE_KEY_FILE="${WORKSPACE}/${CFN_KEYPAIR}"
export ANSIBLE_HOST_KEY_CHECKING=False
export APP_STACK_NAME="Call-$APP_NAME-App-${BUILD_NUMBER}"
# Dev Stack
sed -i "s/APP_STACK_NAME/$APP_STACK_NAME/"
./ansible/inventory/dev_stack_dynamic_inventory_aws_ec2.yaml
cat ./ansible/inventory/dev_stack_dynamic_inventory_aws_ec2.yaml
```

ansible-inventory -v -i ./ansible/inventory/dev_stack_dynamic_inventory_aws_ec2.yaml --graph
# Dev Stack Grand Master
sed -i "s/APP_STACK_NAME/$APP_STACK_NAME/" ./ansible/inventory/dev_stack_swarm_grand_master_aws_ec2.yaml
cat ./ansible/inventory/dev_stack_swarm_grand_master_aws_ec2.yaml
ansible-inventory -v -i ./ansible/inventory/dev_stack_swarm_grand_master_aws_ec2.yaml --graph
# Dev Stack Managers
sed -i "s/APP_STACK_NAME/$APP_STACK_NAME/" ./ansible/inventory/dev_stack_swarm_managers_aws_ec2.yaml
cat ./ansible/inventory/dev_stack_swarm_managers_aws_ec2.yaml
ansible-inventory -v -i ./ansible/inventory/dev_stack_swarm_managers_aws_ec2.yaml --graph
# Dev Stack Workers
sed -i "s/APP_STACK_NAME/$APP_STACK_NAME/" ./ansible/inventory/dev_stack_swarm_workers_aws_ec2.yaml
cat ./ansible/inventory/dev_stack_swarm_workers_aws_ec2.yaml
ansible-inventory -v -i ./ansible/inventory/dev_stack_swarm_workers_aws_ec2.yaml –graph

244. Yukarıdaki işi çalıştırdıktan sonra, statik ana bilgisayarlara ping atarak dev dinamik envanterindeki tüm örnekleri test etmek için komut dosyasını aşağıdaki komutla değiştirin.

```
APP_NAME="Petclinic"
CFN_KEYPAIR="goekce-ansible-test-dev.key"
PATH="$PATH:/usr/local/bin"
export ANSIBLE_PRIVATE_KEY_FILE="${WORKSPACE}/${CFN_KEYPAIR}"
export ANSIBLE_HOST_KEY_CHECKING=False
export APP_STACK_NAME="Goekce-$APP_NAME-App-10”
sed -
i "s/APP_STACK_NAME/$APP_STACK_NAME/" ./ansible/inventory/dev_stack_dyna
mic_inventory_aws_ec2.yaml
ansible -
i ./ansible/inventory/dev_stack_dynamic_inventory_aws_ec2.yaml all -
m ping
```

Yukarida CFN_KEYPAIR kismina keypair’I ve alltaki APP_SATACK_NAME kisminda cloud formationda olan template ismini ve numarasinin yazmayi unutmayalim.

```
13:02:10 172.31.41.92 | SUCCESS => {
13:02:10     "ansible_facts": {
13:02:10         "discovered_interpreter_python": "/usr/bin/python"
13:02:10     },
13:02:10     "changed": false,
13:02:10     "ping": "pong"
13:02:10 }
13:02:10 Finished: SUCCESS
```

245. Ansible altina playbooks klasoru olusturup 4 tane yaml dosayasi olusturun.

246. `pb_setup_for_all_docker_swarm_instances.yaml`

```
- hosts: all
  tasks:
  - name: update os
    yum:
      name: '*'
      state: present
  - name: install docker
    command: amazon-linux-extras install docker=latest -y
  - name: start docker
    service:
      name: docker
      state: started
      enabled: yes
  - name: add ec2-user to docker group
    shell: "usermod -a -G docker ec2-user"
  - name: install docker compose.
    get_url:
      url: https://github.com/docker/compose/releases/download/1.2
6.2/docker-compose-Linux-x86_64
      dest: /usr/local/bin/docker-compose
      mode: 0755
  - name: uninstall aws cli v1
    file:
      path: /bin/aws
      state: absent
  - name: download awscliv2 installer
    unarchive:
      src: https://awscli.amazonaws.com/awscli-exe-linux-
x86_64.zip
      dest: /tmp
      remote_src: yes
      creates: /tmp/aws
      mode: 0755
  - name: run the installer
    command:
    args:
      cmd: "/tmp/aws/install"
      creates: /usr/local/bin/aws
```

247. `pb_initialize_docker_swarm.yaml`

```
- hosts: role_grand_master
  tasks:
  - name: initialize docker swarm
    shell: docker swarm init
  - name: install git
    yum:
```

```
      name: git
      state: present
  - name: run the visualizer app for docker swarm
    shell: |
      docker service create \
        --name=viz \
        --publish=8088:8080/tcp \
        --constraint=node.role==manager \
        --
mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock
\
        dockersamples/visualizer
```

248. `pb_join_docker_swarm_managers.yaml`

```
- hosts: role_grand_master
  tasks:
  - name: Get swarm join-token for managers
    shell: docker swarm join-token manager | grep -i 'docker'
    register: join_command_for_managers

  - debug: msg='{{ join_command_for_managers.stdout.strip() }}'

  - name: register grand_master with variable
    add_host:
      name: "grand_master"
      manager_join: "{{ join_command_for_managers.stdout.strip() }
}"

- hosts: role_manager
  tasks:
  - name: Join managers to swarm
    shell: "{{ hostvars['grand_master']['manager_join'] }}"
    register: result_of_joining

  - debug: msg='{{ result_of_joining.stdout }}'
```

249. `pb_join_docker_swarm_workers.yaml`

```
- hosts: role_grand_master
  tasks:
  - name: Get swarm join-token for workers
    shell: docker swarm join-token worker | grep -i 'docker'
    register: join_command_for_workers

  - debug: msg='{{ join_command_for_workers.stdout.strip() }}'

  - name: register grand_master with variable
    add_host:
      name: "grand_master"
      worker_join: "{{ join_command_for_workers.stdout.strip() }}"
```

```
- hosts: role_worker
  tasks:
  - name: Join workers to swarm
    shell: "{{ hostvars['grand_master']['worker_join'] }}"
    register: result_of_joining

  - debug: msg='{{ result_of_joining.stdout }}'
```

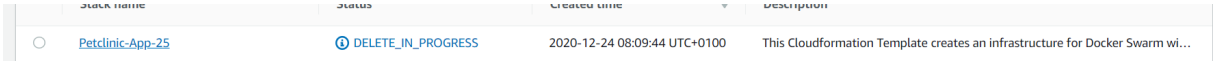
250. git add .
251. git commit -m 'added ansible playbooks for dev environment'
252. git push
253. `test-creating-qa-automation-infrastructure' shel kismina:

```
APP_NAME="Petclinic"
CFN_KEYPAIR="goekce-ansible-test-dev.key"
PATH="$PATH:/usr/local/bin"
export ANSIBLE_PRIVATE_KEY_FILE="${WORKSPACE}/${CFN_KEYPAIR}"
export ANSIBLE_HOST_KEY_CHECKING=False
export APP_STACK_NAME="Goekce-$APP_NAME-App-10"
sed -i "s/APP_STACK_NAME/$APP_STACK_NAME/" ./ansible/inventory/dev_stack_dynamic_inventory_aws_ec2.yaml
# Swarm Setup for all nodes (instances)
ansible-playbook -i ./ansible/inventory/dev_stack_dynamic_inventory_aws_ec2.yaml -b ./ansible/playbooks/pb_setup_for_all_docker_swarm_instances.yaml
# Swarm Setup for Grand Master node
ansible-playbook -i ./ansible/inventory/dev_stack_dynamic_inventory_aws_ec2.yaml -b ./ansible/playbooks/pb_initialize_docker_swarm.yaml
# Swarm Setup for Other Managers nodes
ansible-playbook -i ./ansible/inventory/dev_stack_dynamic_inventory_aws_ec2.yaml -b ./ansible/playbooks/pb_join_docker_swarm_managers.yaml
# Swarm Setup for Workers nodes
ansible-playbook -i ./ansible/inventory/dev_stack_dynamic_inventory_aws_ec2.yaml -b ./ansible/playbooks/pb_join_docker_swarm_workers.yaml
```

```
13:07:32 PLAY RECAP *********************************************************************
13:07:32 172.31.37.180              : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
13:07:32 172.31.37.207              : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
13:07:32 172.31.41.92               : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
13:07:32
13:07:32 Finished: SUCCESS
```

254. Docker swamlari kapatalim.

```
PATH="$PATH:/usr/local/bin"
APP_NAME="Petclinic"
AWS_STACK_NAME="Goekce-$APP_NAME-App-10”
AWS_REGION="us-east-1"
aws cloudformation delete-stack --region ${AWS_REGION} --stack-name ${AWS_STACK_NAME}
```




Bu kod ile coloudformation'daki template silindi.Template silinice de ec2 instance'lar da silindi.


255. Keypair silelim.

```
PATH="$PATH:/usr/local/bin"
CFN_KEYPAIR="goekce-ansible-test-dev.key"
AWS_REGION="us-east-1"
aws ec2 delete-key-pair --region ${AWS_REGION} --key-
name ${CFN_KEYPAIR}
rm -rf ${CFN_KEYPAIR}
```

```
13:08:53 + PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/local/bin
13:08:53 + APP_NAME=Petclinic
13:08:53 + AWS_STACK_NAME=Petclinic-App-25
13:08:53 + AWS_REGION=us-east-1
13:08:53 + aws cloudformation delete-stack --region us-east-1 --stack-name Petclinic-App-25
13:08:55 Finished: SUCCESS
```

256.