##EC2 Comands

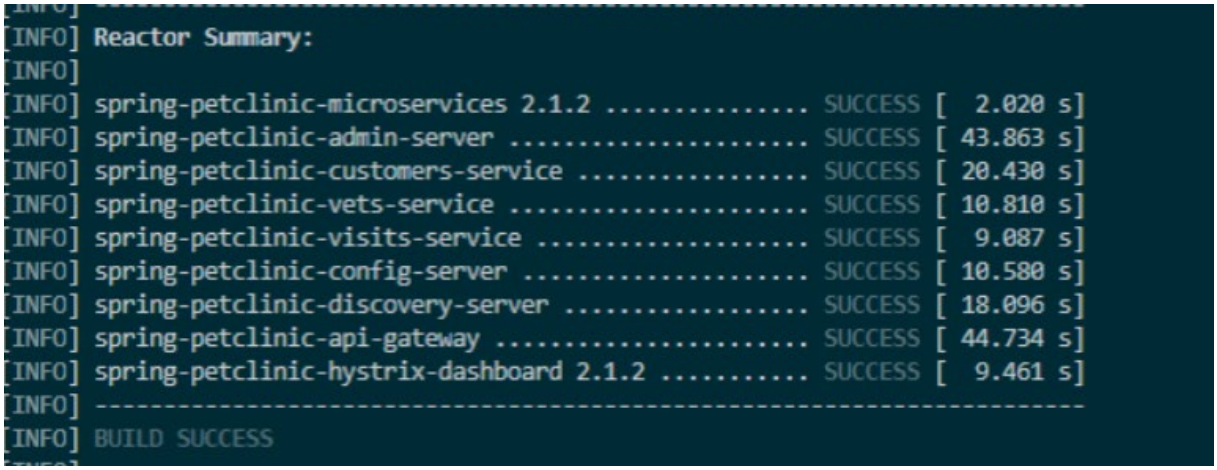
1. sudo yum update -y # Ec2 update edin
2. sudo hostnamectl set-hostname petclinic-dev-server # Ec2 adini petclinic-dev-server olarak degistirin
3. bash # Petclinic-dev-server ismini gorelim
4. sudo amazon-linux-extras install docker -y # Docker kuralim
5. sudo systemctl start docker # Docker aktive 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

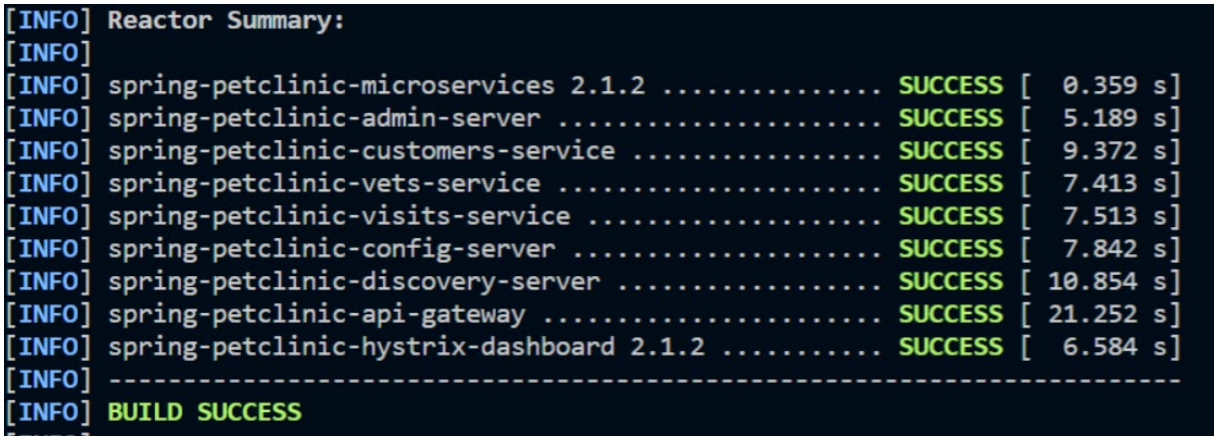
    Bu kodu girince çıktısı aşağıda olacaktır

    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
```

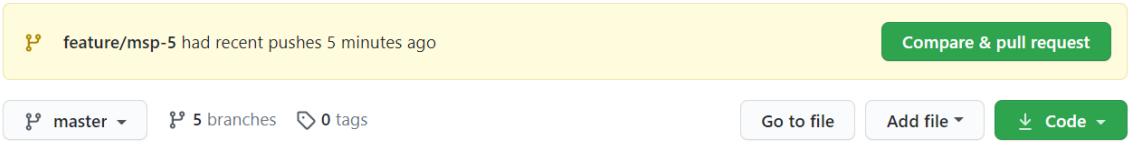
41. git checkout dev
42. git branch feature/msp-4 # Feature/msp-4 bransi olusturalim
43. git checkout feature/msp-4
44. git add .
45. git commit -m "added mvn package script"
46. git push -u origin feature/msp-4
47. git checkout dev # Dev bransina « merge » etmek icin geciyoruz.
48. git merge feature/msp-4 # "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
```

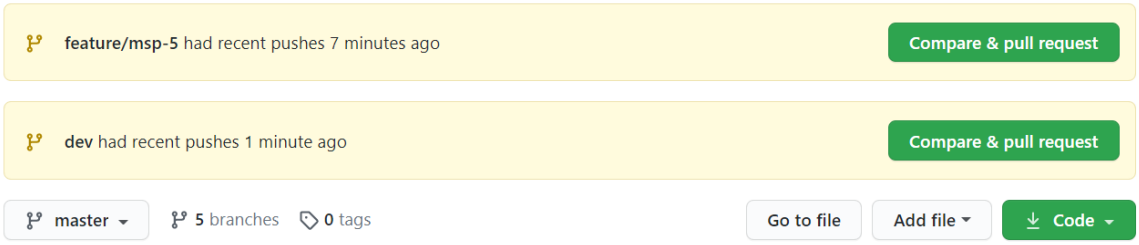
```
SecurityGroupIngress:
  - IpProtocol: tcp
    FromPort: 22
    ToPort: 22
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 80
    ToPort: 80
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 8888
    ToPort: 8888
    CidrIp: 0.0.0.0/0
  - IpProtocol: tcp
    FromPort: 8761
    ToPort: 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
```

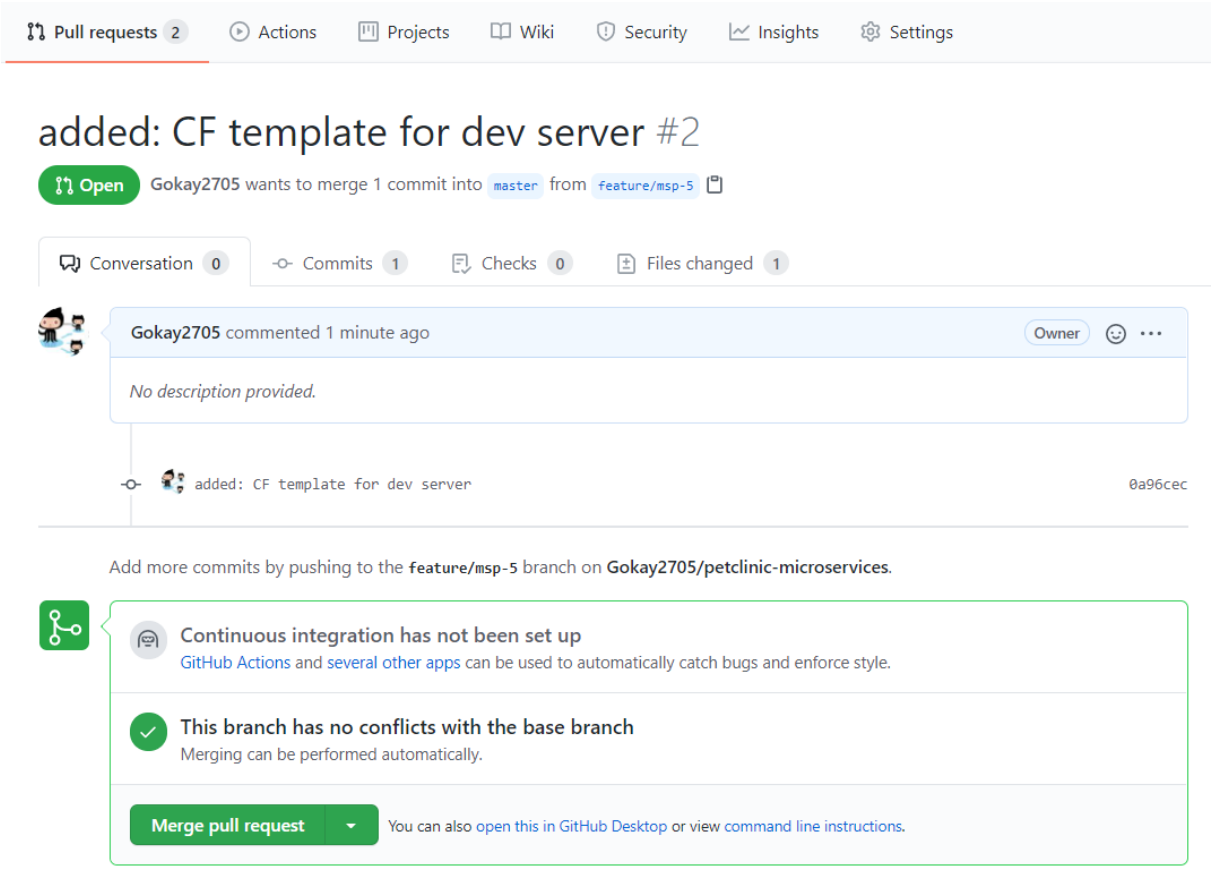
53. git checkout feature/msp-5
54. git add .
55. git commit -m "added CF template for dev server"
56. git push -u origin feature/msp-5 # Yaptigimiz degisikleri push ettik
57. git checkout dev # Yapilan degisikleri merge etmek icin dev gectik
58. git merge feature/msp-5 # Merge ediyoruz



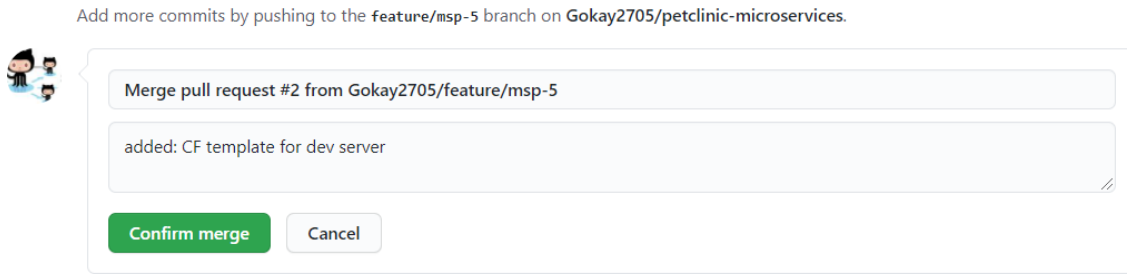
59. git push -u origin dev # merge ettigimiz dosyayi push ediyoruz.



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



61. Yapilan “merge pull reguest” talebini onaylayin.

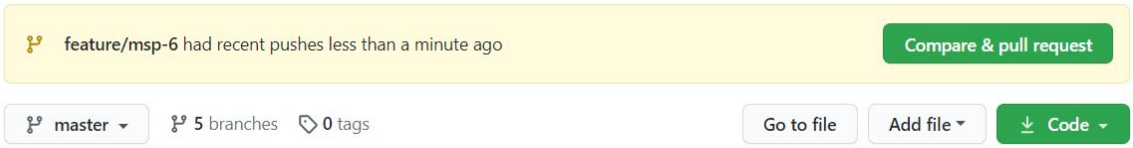


PART-II

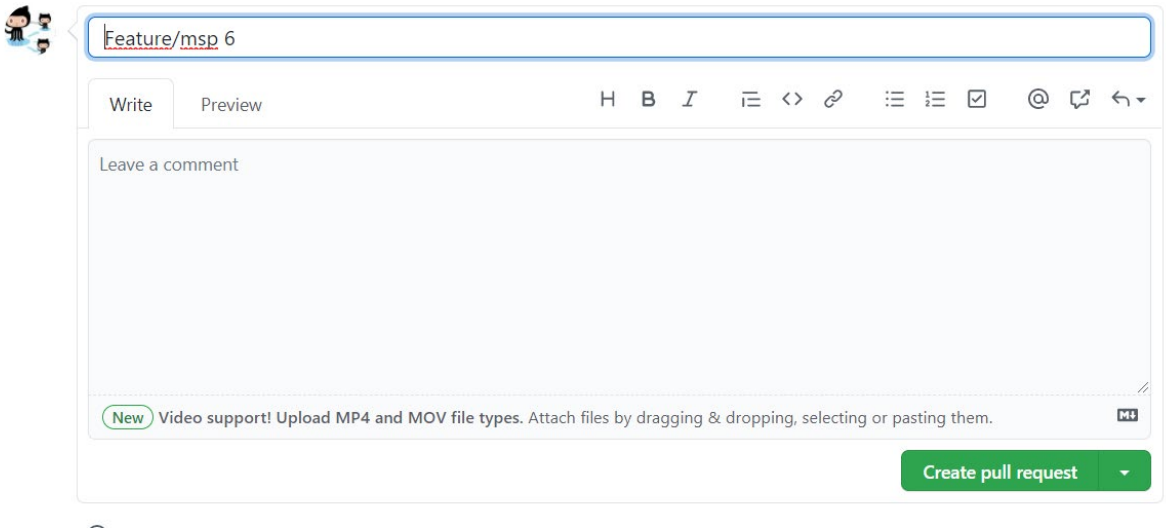
62. Git checkout dev
63. git branch feature/msp-6 # git checkout -b feature/msp-6
64. git checkout feature/msp-6
65. “admin server directory” klasoru icinde “Dockerfile” olustirarak imaj olusturacagiz.

    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'

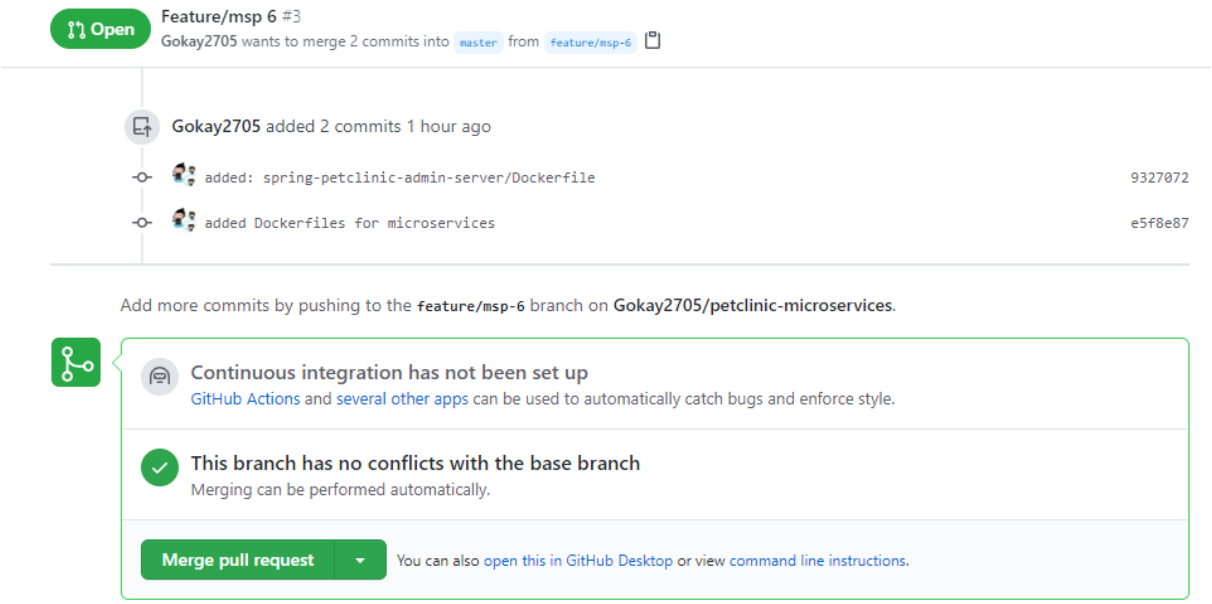
75. git push --set-upstream origin feature/msp-6
76. git checkout dev
77. git merge feature/msp-6
78. git push origin dev



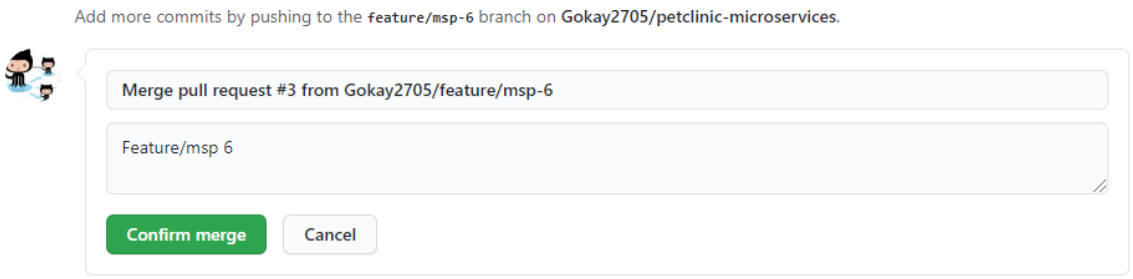
79. Compare & pull request



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@petclinic-dev-server petclinic-microservices]$ docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
petclinic-prometheus-server   dev                 5e21af1f194d        13 seconds ago      110MB
petclinic-grafana-server      dev                 9bd28389b3ef        16 seconds ago      245MB
<none>                        <none>              58c1ef6a08df        24 seconds ago      286MB
<none>                        <none>              35cf70e72453        25 seconds ago      286MB
<none>                        <none>              4e7703f849a9        25 seconds ago      286MB
<none>                        <none>              e1e3541a98a4        26 seconds ago      286MB
<none>                        <none>              bf58e3f20a65        27 seconds ago      286MB
<none>                        <none>              6a8167f889aa        28 seconds ago      286MB
<none>                        <none>              e301c59249bf        29 seconds ago      286MB
<none>                        <none>              1f4115073896        30 seconds ago      286MB
openjdk                       11-jre              94321aa03ce0        36 hours 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. s
105. hatttaa aldim yapamadim
106. s
107. git checkout dev
108. git branch feature/msp-9
109. git checkout feature/msp-9
110. ./spring-petclinic-customers-service/src/test/java/org/springframework/samples/petclinic/customers/model/ altina “PetTest.java” dosyasi olusturun.

```java
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());
    }
}
```

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

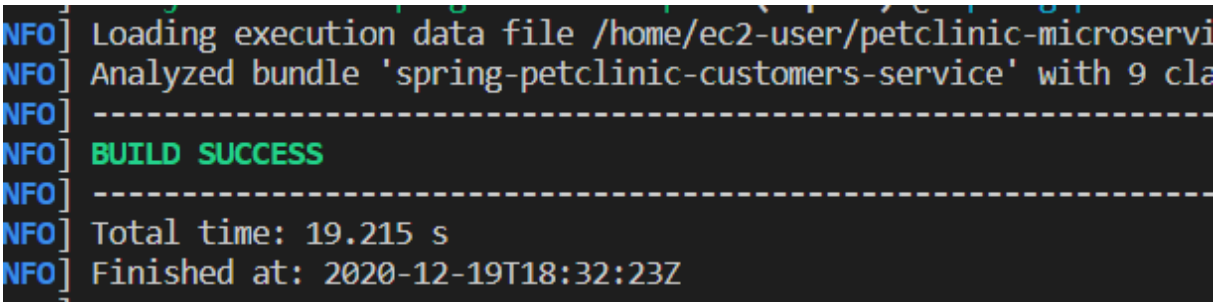
```
[INFO] Results:
[INFO]
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

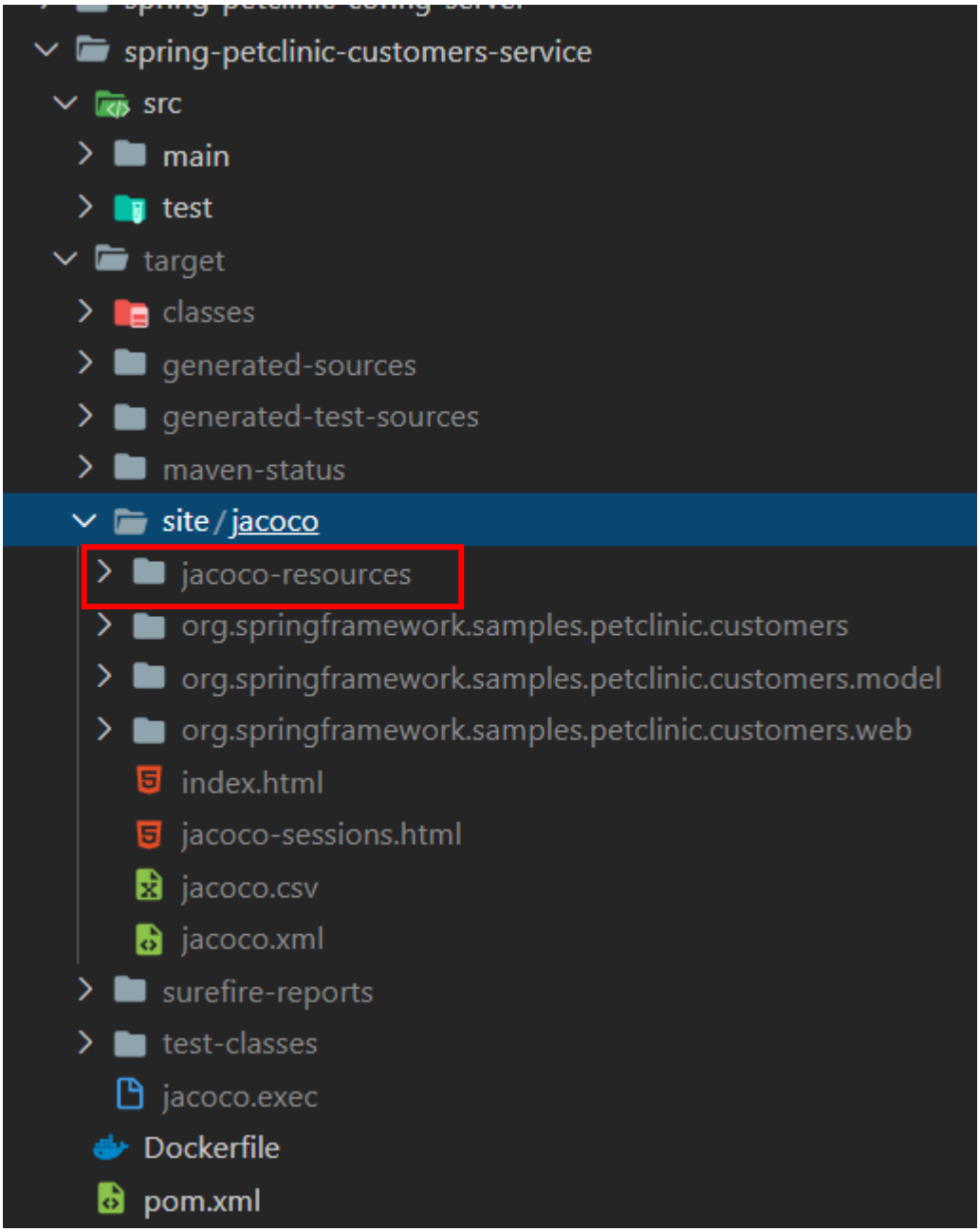
120. Ana klasor altindaki pom dosyasinin icindeki plugins bolumunun sonuna asagidaki kodu yerlestirin.

```
<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>
```

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





Burada jococo calismis ve site/jacoco klasoru olusturdu.

130. cd petclinic-microservices/spring-petclinic-customers-service/target/site/jacoco
131. python -m SimpleHTTPServer # for python 2.7
132. 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

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



test_owners_all_headless.py test_owners_register_headless.py test_veterinarians_headless.py

138.

139. 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()
```

140. 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()
```

141. test_owners_all_headless.py

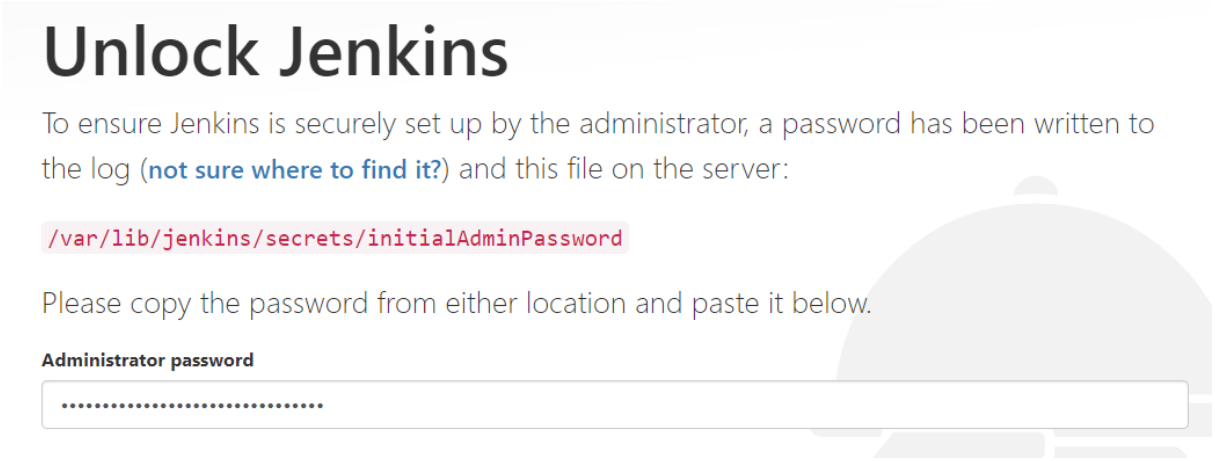
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from time import sleep
import os

# 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()
```

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

III. Day

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

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



152. sudo cat /var/lib/jenkins/secrets/initialAdminPassword # cikan sifreyi giriniz.
153. git checkout dev
154. git branch feature/msp-11
155. git checkout feature/msp-11
156. 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
```

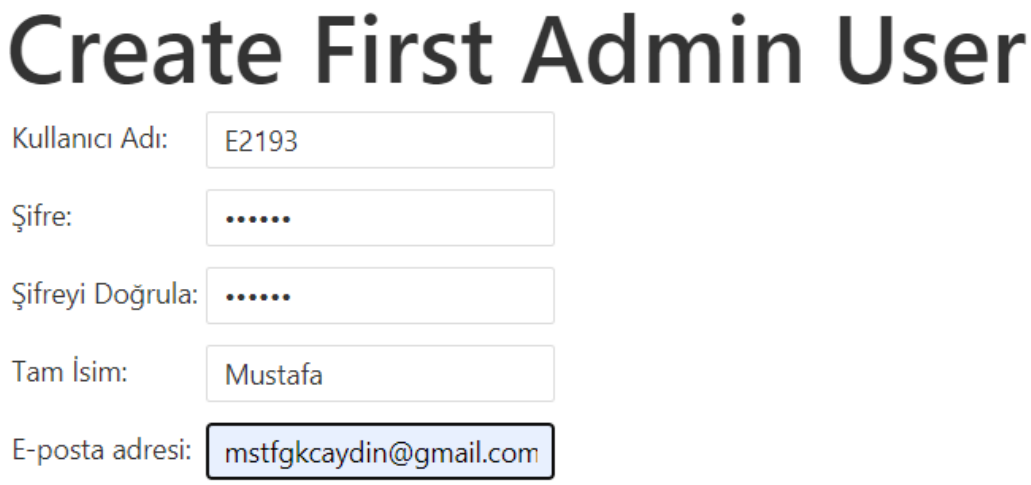
```
        # install git
        yum install git -y
        # install java 11
        yum install java-11-amazon-corretto -y
        # install jenkins
        wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat/jenkins.repo
        rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
        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:
```

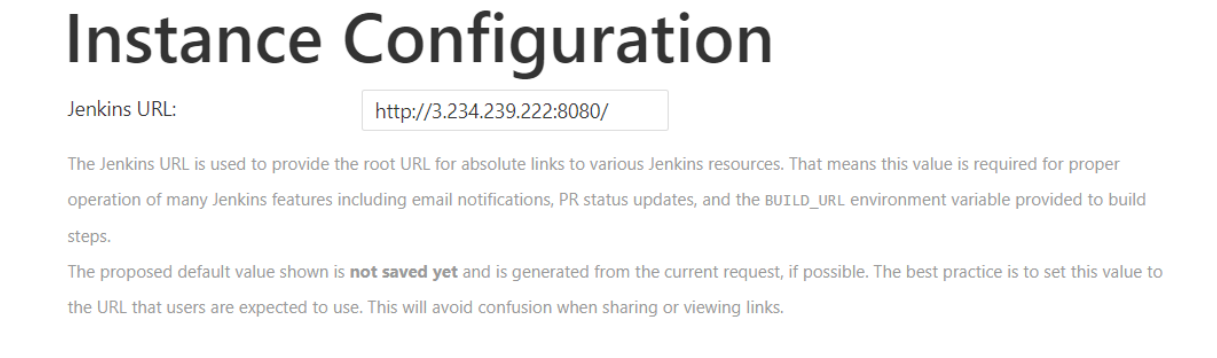
Description: Jenkins Server URL
Value: !Sub
- http://${PublicAddress}:8080
- PublicAddress: !GetAtt JenkinsServer.PublicDnsName

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

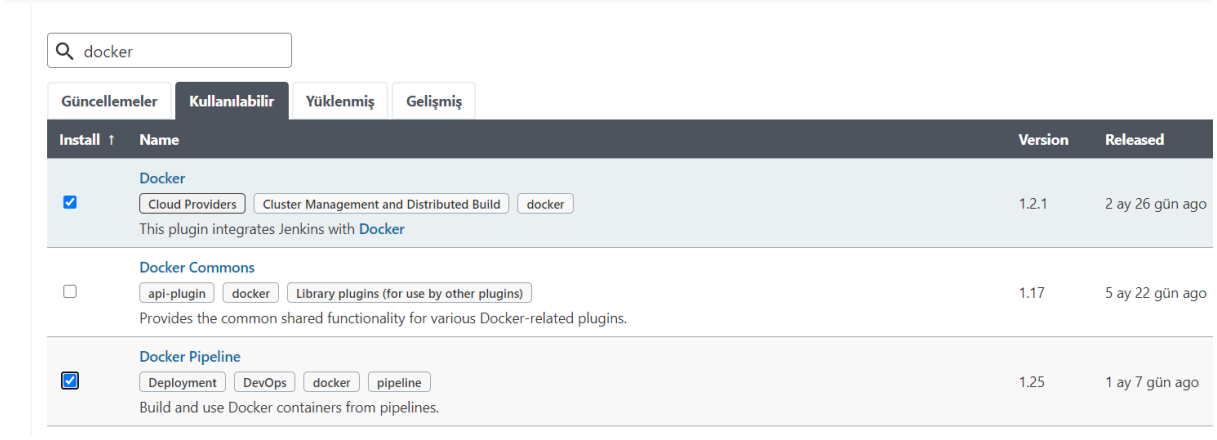
```
[ec2-user@jenkins-server petclinic-microservices]$ git branch -a
* dev
  master
  remotes/origin/HEAD -> origin/master
  remotes/origin/dev
  remotes/origin/feature/msp-10
  remotes/origin/feature/msp-4
  remotes/origin/feature/msp-5
  remotes/origin/feature/msp-6
  remotes/origin/feature/msp-7
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

167. Jenkins baglandiktan sonra :





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

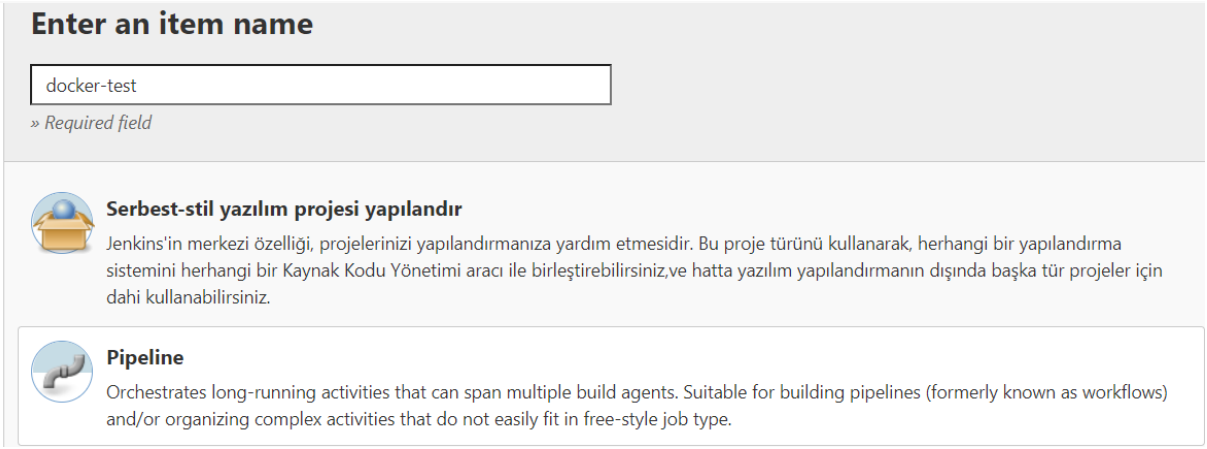


169. Github integration plugin yukleyin.
170. Jacoco plugin yukleyin.
171. Manage jenkins/manage nods and clouds/configure clouds tiklayin.

Test connection tikla.

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

173. Docker-test piple olusturalim.



174. Acilan menuye asagidaki kodu girin.

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

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

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

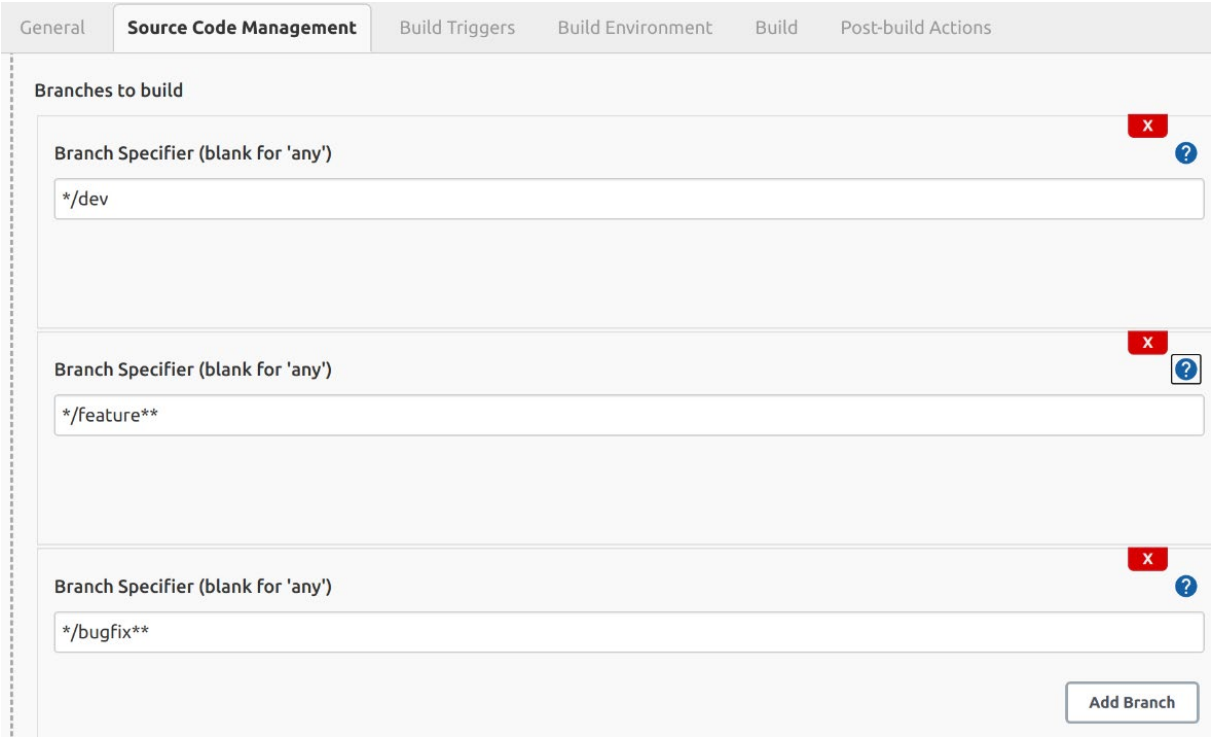
176. git checkout
177. git checkout dev
178. git branch feature/msp-13
179. git checkout feature/msp-13
180. mkdir Jenkins # ana klasor altinda olustur.
181. Jenkins server uzerinden New item/ petclinic-ci-job adini verin/ Freestyle project secin.

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

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

184. Brans olarak dev yazin.

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



186. Buidnow deyin.

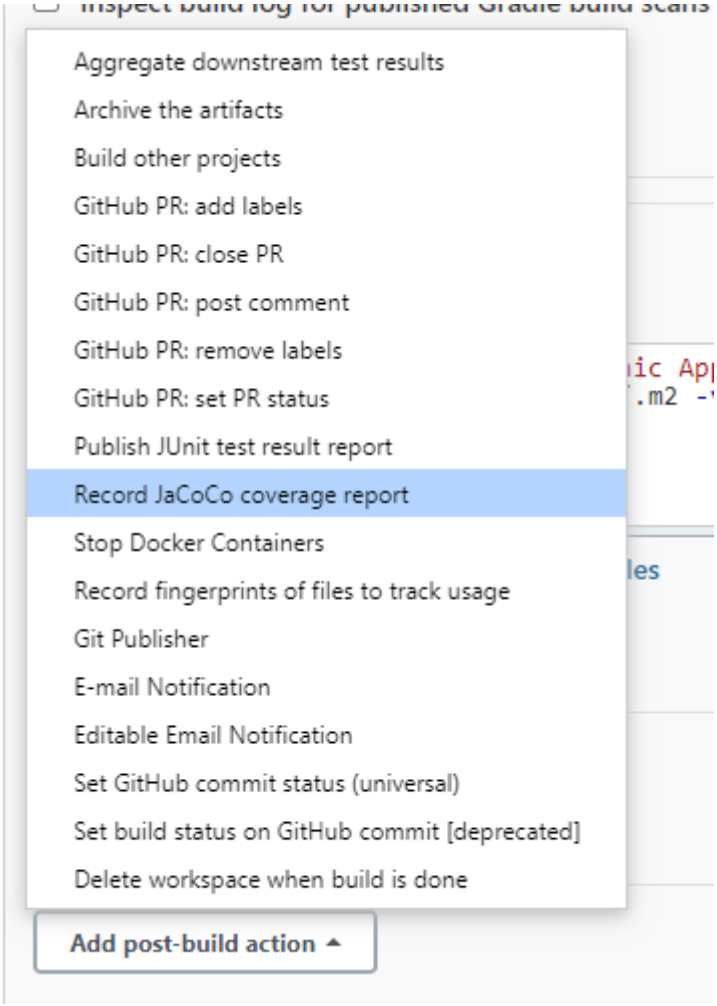
Console Output

```
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
```

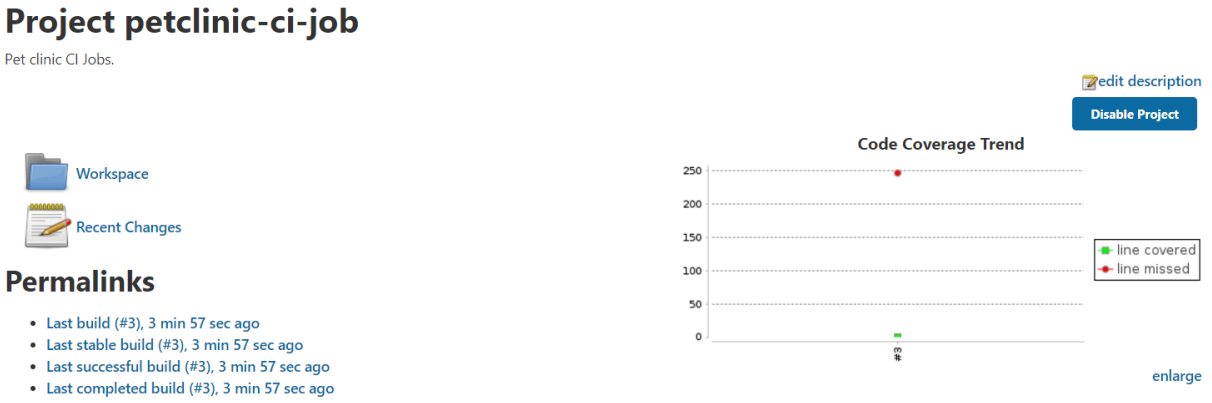
187. Buid triger ve Buid Envirenment asagidaki secenekleri secip Execute shell'e asagidaki kodu girin.

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

188. Post-build Actions JaCoco kayitlarini olusturmayi secin.

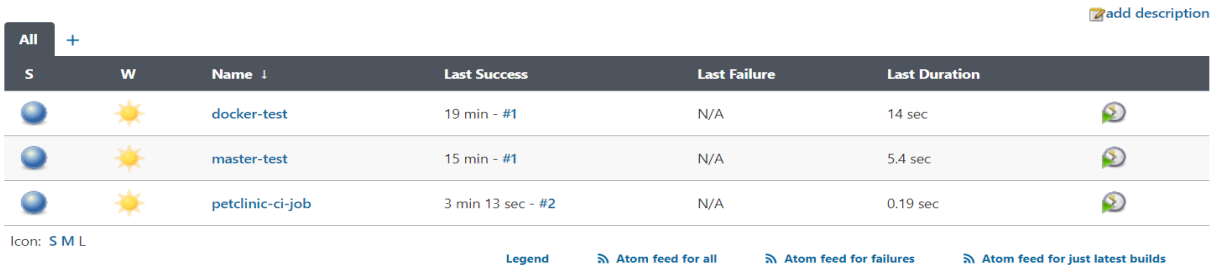


189. Buidnow.



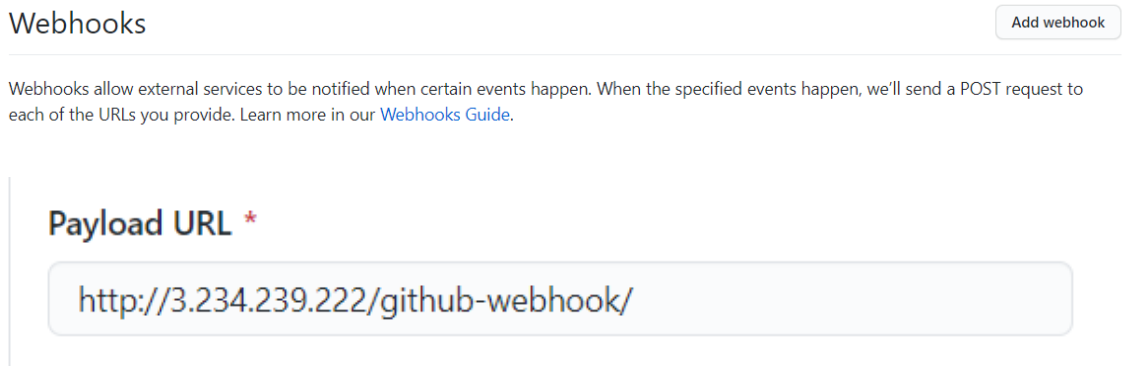
190. Server üzerinden yaptigimiz test islemlerini simdi java containerine yapiyoruz.

191. Genel durum.



192. Webhook ekleyin.

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



Public IPv4:8080 address adresini giriyoruz.

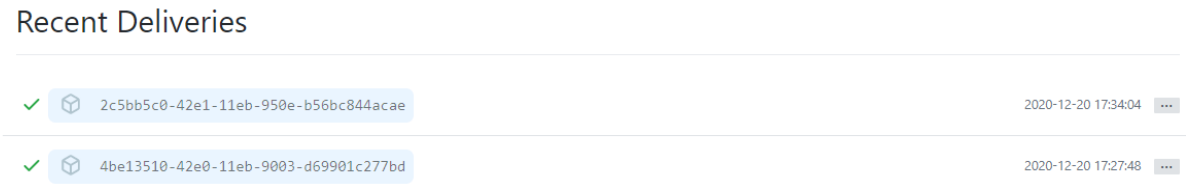
194. Git branch

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

195. 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
```

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



200. git checkout dev
201. git merge feature/msp-13
202. git push origin dev
203.