Proje sırasında ara ara developer veya QA tester şapkası takacağız.

Proje sırasınca uygulamayı aplikasyonda görmek değil pipeline yazıp, success almak, taskleri deneyerek gitmek asıl hedefimiz.

Bir task aldığımızda önce localde deneriz, burada da önce docker-compose ile çalıştığını gördük, bundan sonra Kubernetese aktaracaz.

petclinic-app 
GitHub 
unit tests ve jacoco report 
Devops 
dev-server 
devops 
14:54 
I. ci-job 
every commit 
functional tests 
12. pipeline—nightly 
every night 
for QA testers 
13. pipeline-weekly 
every Sunday 
for staging 
| 4. pipeline-staging 
every Sunday 
production 
| 5. pipeline—prod 
every commit (master 

Developerlar kodu yazıyor ve bu kodlar unit testlerden geçiyor, biz de bunun için CI job yazacaz. Jacoco test sonucunda rapor hazırlıyor bizim DevOpsçu olarak görevimiz bu raporu sunmak.

Developerlar feedback doğrultusunda yeni bir özellik eklediğinde ya da kod yazmaya devam ettiğinde test kodlarını da yazacak. Bizim yazacağımız job da her commit de çalışarak unit testlerini yapacak.

Geçiyorsai kalıyorsai yöneticinin istediği doğrultuda başarılı ise elimizde daha güzel bir kod oluşmuş olacak. Unit testi biz maven la yapıyoruz. Makine ayağa kaldırıp uygulama ayağa kaldırmaya gerek yok bunun için.

Mvn clean test komutu yeterli oluyor. Github a her commit ve push olduğunda unit test olacak.

Daha sonra fonksiyonel testler olacak. Otomatize şekilde bir uygulamanın testleri yapılır. Örneğin logine tıkladığımızda o sayfanın açılmasıdır. Fonksiyon testlerin yapılması için uygulama ayağa kalkar ve üzerinde fonksiyonel testler yapılır. Bunun için de Selenium tool kullanılıyor.

Pipeline ile bir cluster oluşturacaz, terraform ile 3 tane node oluşacak ve kubernetes cluster ı oluşturularak uygulama deploy edilip üzerinde fonksiyonel testler yapıldıktan sonra her şeyi destroy edecek. Bu pipeline her gece çalışacak. Çünkü kodlar gündüz yazılacak ve unit testler de kodu daha güzel hale getiriyor. Geceleyin de fonksiyonel testleri yapılıyor.

Tabi biz simule edecez, pipeline yazdığımızda bu işi bitirmiş olacak.

Testler burada da bitmiyor QA testerların manuel yaptığı testler var. Bunun için de uygulamanın yağa kalkması gerekiyor. Burada yazacağımız pipeline kendini termianate etmeyecek çünkü üzerinde QA testerler hafta boyunca çalışacak ve rapor hazırlayacak. Developerlara feedback yapacak. Her seferinde yeni clusterlar açmıyorlar, kodları GitHub tan çekip testlerini yapıyorlar.

Daha sonra production öncesi son kabul testlerinin yaptığı staging aşaması olacak. Bizim örneğimizde böyle bazı firmalarda direk production a da çıkabilir. Bu pipeline da da bir cluster ayağa kalkacak. Staging için ayrı bir ekip oluyor. Burada elip hafta boyu çalışıp developerlara feedback yapacak. Developer da sürekli çalışıyor.

En son uygulamanın son haliyle production a çıkıyor ve müşterinin önüne geliyor. Burada da cluster olacak. Feedbackler olacak. Bu artık main branchta olacak. Her commit yapıldığında production da uygulamanın son hali deploy edilmiş olacak.

1. **Adıma geçiyoruz:**

buradaki görevimiz unit trst. bu developerların işi ancak biz de burada örnek olsun diye yapacaz.

git checkout dev

git branch feature/msp-11

git checkout feature/msp-11

v spring-petclinic-con... 
v src 
v main 
> java 
> resources 
v test/java.org/s... 
J 
PetclinicConfigS... 

server ların içinde bu şekilde test kodları da yazılmış.

Biz pet-clinic-customers-service içine test kodu ekleyecez.

v spring-petclinic-customers-service 
v src 
v java.org/springframework/samples/petclinic/customers 
> model 
> web 
J 
CustomersSewiceApplication.java 
> resources 
v test 
v java.org/springframework/samples/petclinic/customers/web 
J 
PetResourceTestjava 
> resources 
Dockerfile 
pomxml 

burada main ve test paralel gidiyor ancak baktığımız zaman test altında model .

klasörü yok. buraya model klasörü ekleyip PetTest.java dosyasını oluşturuyoruz.

Buradaki klasör-dosya yapısını herkes aynı dizinlerde çalışsın diye oluşturulur. Yani main ve test altında aynı isimle klasör ve dosyalar var örneğin. Biz bunu maven ile oluşturmuştuk. Ancak editörler de bunu yapabilir.

Unit test her bir birimin testi demektir. Bütün bir dosya da olabilir ya da her bir metodun testi de olabilir. Buna developerlar yöneticilerle karar verir.

@Test

public void testGetName(){

//Arrange

Pet pet = new Pet();

//Act

pet.setName("Fluffy");

//Assert

assertEquals("Fluffy", pet.getName());

}

bu modeldir.

Biizm yapıştırdığımız test dosyasında üç tane test var. uygulamanın customer sayfasında bir kişi ismini girdiğinde doğru bir şekilde giriyor mu bunu test ediyor örneğin.

setName karşılığında Fluffy demiş, aiaüıda da assertEquals da dönüşte de Fluffy döndürüyorsa bu başarılıdır demektir.

Junit bu testleri yazmakta kullanılan bir frameworktür.

Şimdi testi çalıştıracaz.

(Bu test kısımlarını şu an genel kültür olarak öğreniyoruz, ileride çalıştığımız zaman şirket içinde öğreniriz bunları.)

petclinic-customer-service klasörü altına gidelim.

ec2-user@jenkins-server# spring-petclinic-cu 
• stomers-service:( feature/msp-ll)$ Is 
Dockerfile pom.xml src target 
ec2-user@jenkins-server# spring-petclinic-cu 
o stomers-service:( feature/msp-ll)$ 

../mvnw clean test

komutuyla testi çaıştıracaz. clean maven ın daha önce oluşturduğu dosyaları siler. target folderdaki dosyaları silecek.

burada .. ile bir üst klasörü neden verdik çünkü pom.xml burada.

[INFO] 
[INFO 
] Results: 
[INFO] 
[INFO] Tests run: 4, 
Skipped: 
[INFO] 
[INFO] 
Failures: e, 
Errors 
[INFO] BUILD SUCCESS 
[INFO] 
[INFO] Total time: 23.655 s 
[INFO] Finished at: 
[INFO] 

BUILD SUCCESS aldık.

test te hata alırsa şöyle bir çıktı alırız:

(INFO] Total time: 19.621 s 
[INFO] Finished at: 
[INFO] 
(ERROR] Failed to execute goal 
(default-test) on project spring-pet 
re are 
[ERROR) 
(ERROR] 
"Its. 
[ERROR) 
(ERROR) 
[ERROR] 
[ERROR] 
[ERROR] 
[ERROR) 
(ERROR) 
[ERROR I 
Please refer to 
Please refer to files (if any exist) [date) -jvrRun(N) .d'—, (date] .dw.stre— and 
[Help 1] 
To see the full stack trace of the errors, re-run Maven with the -e switch. 
Re-run haven using the -X switch to enable full debug logging. 
For more information about the errors and possible solutions, please read the following articles: 
[Help I) http;//cwiki.apache.org/confIuence/display/MVEN/MojoFaiIureException 
rtenkins-server* spring-petclinic-custmers-service: (feature/msp-ll) 
O ec2-use 

(Burada hata almak için hoca Fluffy nin y sini sildi.)

Hata aldıktan sonra mvn clean package dersek paketi üretir mi üretmez? bu işlemller developerlar içindir. Hata alan developer kodunu düzeltir.

şimdi petclinic-microservice-with-db ana dizine gidelim.

ve yaptığımız değişiklikleri commit leyelim.

Jacoco Java code coverage java için rapo hazırlayan bir plugin. Bizim işimiz DevOps çu olarak bu raporları hazırlamak. Bu plugini kullanma, bununla ilgili pom.xml e ekleme yapma yapmamız gerekiyor. Jenkins te job u çalıştırdığımızda jacoco plugini bize raporu verecek ancak,

Pluginin çalışması için pom.xml e

<plugin>

<groupId>org.jacoco</groupId>

<artifactId>jacoco-maven-plugin</artifactId>

<version>0.8.10</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>

bu kısmı ekliyoruz.

bunu eklememiz gerektiğini mavnrepository docs tan alıyoruz. bu pluginin nasıl kullanılacağı ile ilgili orada örnekler ve bilgiler mevcut.

pom dosyası xml formatında girintiler önemli değil aç-kapalar yeterli. plugins altına pluginler açılıp kapanmış biz de plugins içerisine jacoco pluginini yapıştırdık. İstersek aynı hizaya getirebiliriz.

jacoco ile oluşturulan raporlar neden önemli? bunu managerlar yada senior developerlar isteyebiliyor. bazen bütün kodların unit test yapılmasını istemiyorlar.

şimdi cutomer-service klasörü içinde

../mvnw test

komutunu tekrar çalıştıracaz.

BUILD SUCCESS aldık. ana dizine gidip commit ve push edebiliriz.

şimdi cusotmer service altında raporu görmek için

python3 -m http.server

komutunu çalıştıralım. Ana dizinde çalıştırısak tüm uyuglamanıın raporunu getirir.

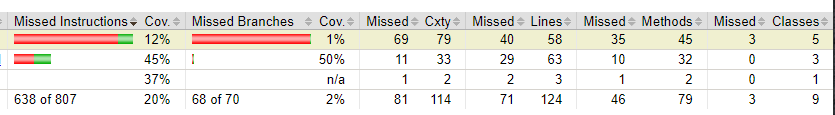
8000 portunda target/site/jacoco altında bu raporu görebiliyoruz.

spring-petclinic-customers-service 
spring-petclinic-customers-service 
Missed • Cxty$ 
Missed • Lines' 
Element 
EB Qlgszjngframework etclinic customers.web 
EB Qlgszjngframework etclinic customers.model 
EB 
Total 
Missed Instructions' 
638 of 807 
Cov. A 
12% 
37% 
20% 
Missed Branches 
68 of 70 
A Cov.$ 
1% 
n/a 
2% 
69 
11 
79 
33 
114 
40 
29 
58 
63 
124 
Missed" Methods A 
35 
45 
10 
32 
46 
79 
Missed A 
Classes e 

Manager sınıfı developerlardan yazdığın unit test tüm uygulamanın %70 ini kapsasın diyor ya da %50 ile %70 arasında olsun. Zaten bütün kodlara unit test yapılması da mükemmel olduğu anlamına gelmiyor. Bu nedenle genelde böyle oran veriliyor. Unit testlerden geçiyor ancak bütün kodlara unit test yapılmamış oluyor.

org.springframework.samples.petclinic.customers.model 
Element 
@ Owner 
Total 
Missed Instructions' Cov. A 
= 34% 
53% 
100% 
104 of 192 
Missed Branches $ Cov.$ Missed: 
City $ Missed: Lines $ Missed $ 
M ethodse 
16 
12 
32 
Missed 
Classes A 
1 of2 
10 
11 
17 
12 
33 
29 
34 
24 
63 
10 
created with Jacoco 1020230424095 

Satır bazında, method bazındai class bazında vs. ne kadarını kapsadığını gösteriyor. bazen if-else li cümleler den bazılarının testi yazılıp bazılarının yazılmıyor. Onları da gösteriyor.



Missed olanlar eksik kalan kısmı.

Raporda PEt sekmesnin içen geldik:

public void setBirthDate(finaI Date birthDate 
this. birthDate = birthDate; 
public Pet Type getType() { 
return type; 
public void setType (final Pet Type type) { 
this . type 
type; 
public m•ner getmner() { 
return owner; 
public void setOwner(finaI Owner owner) { 
this. owner = 
owner ; 
Override 
ublic String toString 
return new ToStringCreator t 1 
. append( , 
this . getld 
. append( "name" 
this . getuame 
0) 
. append( "birthDate'% this . getBirt 
. append( "type" 
this . get Type() 
. getName() 
. append( " 
this . . getFirstN 
. a p pen d ( " o I 'In erLastname%this.ge t t Last-Name()) 

Test yazılan kodlar yeşil, yazılmaan kodlar kırmızı ile gösteriliyor.

Devops çunun işi raporu hazırlamak, developerın işi kapsamayı istenen seviyeye getirmek.

1. adıma geçiyoruz:

Buradaki görevimiz manuel olarak hazırladığımız raporu şimdi jenkins te job olarak yazacaz.

projenin ana dizininde jenkins isimli bir klasö olutşururyoruz.

Browserda da jenkins e gidiyoruz.

- job name: petclinic-ci-job

- job type: Freestyle project

- GitHub project: [https://github.com/[your-github-account]/petclinic-microservices](https://github.com/%5Byour-github-account%5D/petclinic-microservices)

- Source Code Management: Git

Repository URL: [https://github.com/[your-github-account]/petclinic-microservices.git](https://github.com/%5Byour-github-account%5D/petclinic-microservices.git)

- Branches to build:

Branch Specifier (blank for 'any'): - \*/dev

- \*/feature\*\*

- \*/bugfix\*\*

- Build triggers: GitHub hook trigger for GITScm polling

- Build Environment: Add timestamps to the Console Output

-Post-build Actions:

Add post-build action: Record jacoco coverage report

- Build:

Add build step: Execute Shell

Command:

bu bilgilerle bir job oluşturacaz.

Ayrıca makine dolmasın diye

Discard old builds 
Strategy 
Log Rotation 
Days to keep builds 
if not empty, build records are only kept up to this number of days 
3 
Max # of builds to keep 
if not empty, only up to this number of build records are kept 

3 gün ve son 3 buildi tutsun diye conf yapalım. Biz kendimiz çalışma amaçalı ayrı bir branch ta çalışırsak(proje branchları dışında) jenkins altındaki workspace ten o workspace leri silelim.

bizde şu an bugfix branchı yok ancak gerçek hayatı simule ettiğimiz için ekledik.

her commit te unit test in raporunu görmesi lazım bunun iiçn trigger ekliyoruz.

Add post-build action: Record jacoco coverage report

Gerçek hayatta birden fazla jacoco exec dosyası olabilir.

Record JaCoCo coverage report ? 
Exclusions (e.g.: 
Path to exec files (e.g.: 
exec 
Inclusions (e.g 

buradaki yıldız bütün exec dosyalarını alması için.

inclusion bunları da ekle, exclusion da bunları hariç tut demektir.

Path to class directories (e.g.: H/target/cIessDir, 
y classes) 

class dosyalarının pathi. biz default tutuyoruz.

Path to source directories (e.g.: 
Inclusions (e.g.: 
Exclusions (e.g.: 

source kodlarının pathi ve dahil et hariç tut kısmı.

Always run coverage collection, even if build is FAILED or ABORTED ? 
Instruction 
% Branch 
% Complexity % Line 
70 
Fail the build if ccwerage degrades more than the delta thresholds 
Instruction 
% Branch 
% Method 
% Method 
% Class 
% Class 
% Complexity % Line 
10 

bu kısmı da developerlar isterse dolduruyoruz. Örneğin burada satırlar da %40 ile %70 arasını hedefledik.

Alt kısımdaki %10 ise şu demek:

built işlemi her gün yapılıyor örneğin ve ikinci builtte ilke builte göre oran %10 dan aşağı düşerse fail ver demektir.

verdiğimiz orandan daha çok düşerse fail yani.

Build stepse de Execute shell olarak şu kodu ekliyoruz.

echo 'Running Unit Tests on Petclinic Application'

docker run --rm -v $HOME/.m2:/root/.m2 -v `pwd`:/app -w /app maven:3.8-openjdk-11 mvn clean test

amacımız mvn test yaptırmak ancak biz bunu docker a yaptırıyoruz. piyasada makineyi şişirmemek için yaygın bir kullanım bu.

CI/CD toollarında docker ile çalıştırmak her zaman daha efektif olabiliyor.

docker run maven:3.8-openjdk-11 mvn clean test

burada kontainer ı bu image ile çalıştırıyoruz ve konteynır çalışır çalışmaz bir komutun çalışmasını istiyorsak komutu böyle yazdığımızda çalışıyor.

ancak bu halile çalışmaz çünkü image içinde bizim dosyalarımız yok pom.xml yok.

bunu nasıl yapıyoruz.

docker run -v `pwd`:/app maven:3.8-openjdk-11 mvn clean test

`pwd`

backtick ile yazılan bir şeyi $ ile yazmak gibi komut olarak algılar. Linux konusu bu. bi burada lokalimdeki dosyaları yani pwd yi konteynır içinde app klasörü aç oraya kopyala dedik.

ancak halen çalışmaz çünkü app klasörüne geçmeli.

docker run -v `pwd`:/app -w /app maven:3.8-openjdk-11 mvn clean test

- w app ile o klasöre geçmeyi de ekledik kodumuza.

bu komut da konteynır içine .m2 folder ı oluşturup oraya indirecek dependencies i. her seferinde yapacak bunu. o nedenle biz lokaldeki .m2 folder ını volume olarak bağlayacaz ve lokaldeki.m2 konteynırdaki .m2 yi ezecek. içindeki dosyalarla gelecek. Bu şekilde işlem hızlı olacak ve makineyi yormayacak.

docker run -v $HOME/.m2:/root/.m2 -v `pwd`:/app -w /app maven:3.8-openjdk-11 mvn clean test

--rm i de işi bitince kotenyırı siliyor. komutun son hali:

docker run --rm -v $HOME/.m2:/root/.m2 -v `pwd`:/app -w /app maven:3.8-openjdk-11 mvn clean test

jenkins folder altında jenkins-petclinic-ci-job.sh

dosyasını oluşturalım. bu dosya içine de docker run komutumuzu girecez.

bunu neden yapıyoruz; job da çalıştırdığımız komut dosyalarımızın arasında da yer alsın diye yapıyoruz. çalışma mantığımız bu olacak. Havada hiçbir şey kalmayacak, yaptığımız işlemler kayıt olarak kalsın diye saklayacaz.

bazen de scripti çalıştıracaz.

şimdi Github repomuza gidip webhook conf ediyoruz.

Webhooks / Add webhook 
We'll send a POST request to the LJRL below with details of any subscribed events. You can 
also specify which data format you'd like to receive (JSON, etc). More 
information can be found in our developer documentation. 
Payload URL 
http://100.24.124.194:8080/github -webhaok,d 
Content type 
application/x-wvm-form-urlencoded 
Secret 
Which events would you like to trigger this webhook? 
@ Just the push event. 
C) Send me everything. 
C) Let me select individual events. 
Active 
We will deliver event details when this hook is triggered. 
Add webhook 

Biz start stop yapınca jenkins server ı bu webhook çalışmayacak çünkü public ip değişmiş olacak. bunu düzeltmeye çalışmayalım.

Gerçek hayatta jenkins server kapanmayacağı için bir sorun olmayacak.

git add .

git commit -m 'added Jenkins Job for CI pipeline'

git push --set-upstream origin feature/msp-12

git checkout dev

git merge feature/msp-12

git push origin dev

komutlarıyla değişiklikleri push ve merge ettik. webhook çalıştı ve trigger çalıştı:

Build History 
Q Filter builds... 
(pending—guild is already in progress (ETA: N/A)) 
Jun 3, 2023, 7:08 AM 
Atom feed for all 
trend v 
Atom feed for failures 

job çalıştığında jenkins şu an central repodan dosyaları çekiyor? neden biz .m2 volume ünü bağlamıştık oysa. Çünkü henüz konteynırda çalışmıyor, jenkins user içindeki workspace e indirecek bu dependencies i.

jenkins user altında .m2 folder oluştu yani.

jenkins user shell ine gidip bakalım. Ancak jenkins user ın bir shell i yoktu. Çünkü regular bir user değil, system user ı jenkins user. bunun için bir komudmuz vardı:

sudo usermod -s /bin/bash jenkins

sudo su - jenkins

üstteki komut ile jenkins user a shell verdik alttaki komut ile jenkins user ın shell ine geçtik.

Job SUCCES oldu. 2. builde baktığımızda coverage reportu görebiliyoruz:

Coverage Re port 
previous Build 
Next Build 
Build #2 (Jun 3, 2023, AM) 
No changes. 

Coverage report a tıklarsak raporu görürüz:

Overall Coverage Summary 
name 
all classes 
instruction 
M: 1779 C: 501 
branch 
complexity 
M: 265 C: 88 
branch 
100% 
100% 
100% 
100% 
M: 158 c: o 
100% 
line 
method 
M: 262 C: 6 
Coverage Breakdown by Package 
name 
org.springframework.samples.petclinic.admin 
org.springframework.samples.petclinic.api 
org.springframework.samples.petclinic.api.application 
org.springframework.samples.petclinic.api.boundary.web 
org.springframework.samples.petclinic.api.dto 
org.springframework.samples.petclinic.config 
org.springframework.samples.petclinic.customers 
5% 
M: 136 C: 110 
complexity 
M: IC: 2 
M: IC:5 
M: 127 C: 5 
4% 
M: IC: 
M: IC: 
M: 9 c: 25 
100% 
100% 
100% 
100% 
cl ass 
class 
instruction 
M: sc: 7 
58% 
M: 14 C: 52 
M: 36 C: 9 
20% 
M: 819 c: 30 
M: 132 C: 87 
line 
M: 1 C: 10 
M: 21 C: 7 
method 
M: 48 C: 5 

line da %40 ile %70 arasında olsun dediğimiz için başarılı geldi %45 ile.

bu raporu ileterek biz de görevimizi yapmış olduk.

JaCoCo Coverage Report 
a 
Download iacoco . exec 

bu raporun dosyasını buradan indirebiliyoruz.

MSP 13 ten 18 in sonuna kadar petclinic-line jenkins file yazmamız gerekiyor bu dosyayı yazmak için de bütün stage leri adım adım deneyecez.