

CAPSTONE PROJECT (1)

Bölüm-1 BUILD TEST (MSP-1-1o)

Stage Enviroment, perdele misliyice bir sunum yapılırsaca ayaga takılırır.

Production Env, TÜLÜYUZ ayakta olması gerek en önceli environment

Artifact'ler ayrı bir depoda saklanır, Jenkins Server veya SCM'de saklanır.

Server Kurulum Sırasonası

- 1- DevOps Server (ilk hizmetler, pipeline geçiş)
- 2- Jenkins Server (chart'lar chi kurulacak)
- 3- Nexus Server (Pipeline'lar "

Müsteri her zaman UI+API Gateway (Port:8080) üzerinden erişmeye başlacak.

Toplam 8 microservice var

- Customer-service REST-API (random port)
- vets-service " (")
- visits-service " (")
- Spring Cloud Config Server (Github Repoja Bağlı)
- Eureka Service Discovery
- Spring Boot Admin Server
- Zulu Distributed Logging Server
- UI+API Gateway (Port: 8080)

REST API

Rest (Representational State Transfer), Server (Sunucu) ve Client (İstemeç) arasında veri alışverişi sağlayan bir teknoloji modelidir.

Rest API'de Rest uygulamaları, web hizmetleri arasında veri alış-verisi yapmak üzere uygulama ana birimidir.

API (Application Programming Interface) bir modelidir. API sayesinde programlar, kendi içinde olmaları ve farklı programlarda mevcut olan bazı işlev ve kabiliyetlerin, gerekli kendi bünyelerinde sunulabiliyor.

JSON (JavaScript Object Notation) tüm sunucular arasında haberleşme için kullanılan ve verilen XML'e göre daha az kompleks, daha küçük boyutta ve daha kolay taşınan bir protokoldür. Rest API, JSON formatı kullanır.

Rest API nasıl çalışır?

Rest, HTTP protokolü kullanarak, url adresler üzerinde veri ve obya alışverişi yapan bir yapının REST API ise Rest uygulamayı yapabilmek için kullanmış olduğu yerlerdir. Bu API yardımcıca REST client'ları ve veri alış-verisi yapabiliyor.

Uygulamada Config Server ayıga kalleneleri dğer
haberinin ayıga kallenemasi lessiz, Kallenelerin ^{2. seviye Service Discovery}
snack dikkatlerini bir orlansı yolu. ^{ayıga belli olur.}

Ayıga dikkatler kendisi Service Discovery'ye rapor edip.
Service Discovery, herhangi bir service görenek hemen
yeni tabloya. Dikkatli sıraklı 2. lüye.

Uygulama 2. tone veri tabanı kullanıyor. "Me" on the fly
değişimiz uygulama ile ayıga kallen ve onları yolu olur
selüle. 2. aşamada MySQL'ı eruer kullanacağız. Daha
sonra son aşamada da MySQL'ı RDS'ye baplayacağız.

Ziplik surekli logları tutacaksı

Ayrıca prometheus ve grafana da kullanacağız.

Spring boot framework kullanılıcak, Java ile geliştirilen
peki çok uygulamada kullanılır.

Microservice yapısı kullanımlı bir uygulama için okupps
environment kurulum.

Project 5 tame şablonu Kursopz.

Gitflow

master: Uygulamanın son versiyonu / çalışan uygulama
buanda durur. Hıç kimse master/mah branch'le
çalışmasın.

dev : Developers'lar master branch'i üzerinde çalışırken
develop ve kod geliştirmeye devam ederler. Bu branch'le
herde bir sonraki aşamaya hazır stable ~~ürün~~ için dorus
feature: Her bir özellik eklemek istendipinde dev
branch'i üzerinde açılır. (FEATURE/F1, FEATURE/F2...)

Her feature aynı anda bir branch açılır. İki bitince
dev branch'i ile merge ederse. (integrate after
the feature is stable and tested)

release/V0.1.0 : Kodlar geliştirildi ve testler geçti. Artık
production'a ready. Dev'lerin versionu buuya kaynır.
Burada bulunan bug'lar düzelttilir ve bugfix1, bugfix2
diye commit edilir. Bug'in çözüldüğü commit'te developer'e
merge eder.

Bütün hatalar giderildikten sonra uygulama V0.1.0
olarak master'a merge edilir.

hotfix: production'da bulunan (master branch'le) uygunla
şanlı hatalar için hotfix (hotfix/hft) kılınır.

Cihaz uygulama üzerinde yapılan düzeltmelerdir. Sonunda
yeni version olur (V0.1.1)

PIPELINES

GİPTA

Build Pipeline: En temel ve herkesh kendisi pipeline.

Cadece uygulamayı compile edip, unit testleri yapıyor.
ve build ediyor

Unit test: Uygulama içindeki fonksiyonların işlediğimiz gibi çalışıp çalışmadığını test eder. Syntax'a bakar.

Her bir fonksiyonun başından önce test edilmesi (terap makinesinde $+/-\times/\div$ islemlerinin ayrı ayrı test edilmesi)

Functional Test: Tüm fonksiyonları beraber test edilmesi.

Build testi developer tarafından Assertor metodları, unit test'lerde kullanılır. Bonus olarak da gibtir. Daha sonra class test geçer. Selenium kullanılarak Automation kullanılır.

jacoco: Bu bir code coverage tool. Her proje de mutlaka kullanılır. Kodumuzun ne kadar stable olduğunu ölçer. Maven'da jacoco pluginini yüklemeniz gereklidir.

Functional Test için Environment ve çalışan bir uygulama şartı. Test'ler gece yapılır. İşi bitince infrastructure silinecektir. Functional Test'lerde uygulamanın en güncel halini almak istedigimiz için gece yapılmalıdır.

Note: Bir Source code doryası, bir unit olasık kabul edilir.

= Unit Test'lerin developer tarafından yazması.

3. Pipeline (Manuel DA)

İnsan müdahalesi gereken testlerde okunur.

Uygulamayı release branch'ine getirip ve testleri erken yapmayı.

Hepsinin sadece belirli günlerde yapılıyor (Paser. 23.5g)

4. Staging Environment

Müşteri / Kullanıcıya sunulur hizmet

Rancher Uygulaması (Tool)

Cluster Orchestration Tool

Cluster'ın mange edilebilirliği (Scale up - down)

K8S'ın kullanımına çok daha kolay.

Semantic Versioning

Major Minor Patch

2 . 0 . 1

minim (crapp) çakışır java versiyonları
kendisi olmasa genellikle Maven'in light versiyonu

* GIT

CAPSTONE PROJECT (2r)

→ Her bir mikroservis için dockerfile oluşturun,
FROM openjdk:11-jre (War file'lar hâlinde olmalıdır
JRE yoksa)

ARG DOCKERIZE-VERSION=v0.6.1

→ ARG = Variable, Argument

↳ docker build --build-arg VERSION=2 -t arg4.

↳ Dockerfile içinde ARG VERSION=1

--build-arg = set build time variables

ENV şeklinde girdiğiniz ENV.var, instance

çapında bir değişken oluştururken, ARG
sadece Dockerfile içinde geçerli ve sadece
imagine build ederken kullanılır. ARG için
default değer tanımlanır (VERSION=1) zorunlu.

ENV.var'ı developer'dan alıyoruz.

ARG'i aşağıda açıkladığımız docker programı
için oluşturduk. İhtiyaç dursa kullanılsın.

ENV SPRING_PROFILES_ACTIVE docker,mysql

Dosya: customer-service → src → main → resources → bootstrap

↳ profiles: docker olursa config-server'dan al

Ben docker,mysql diyeck burada konteynır oluştur-
ucum ve mysql kullanacağımı belirtmiş oldum.

7

ADD

ADD komutu URL adresinden kopyalama veya zip'li dosyaları kopyalama yapabilmeni COPY sebeceyle de kopyalama Local'deki bir dosyayı girdiğinde direkt unzip yapmayı

RUN tar -xzf clockwise, tar.gz (unzip)

ADD ./target/*.jar /app.jar

EXPOSE \${EXPOSED_PORTS}

Buuya sebce bilgi amaci koruyor. Kerteynırıza gidiştiğim portu etkilemes.

ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/app.jar"]

Aslında komut ["java", "-jar", "/app.jar"]

Bu komutu normalde Developer'lar bize verecek

* **Build Docker Images (build-dev-docker-image.sh)**

./mvnw clean package

Anada kodu içinde değişiklik yapılmış olabilir.

Her seferinde uygulamanın en son halini almak istiyorum.

Jar file'in en son halini alıyorum.

docker build --force-rm -t "petclinic-admin-server" ./spring-petclinic-admin-server

Image'in adı petclinic-admin-server'den
image oluşturmak için ./spring-petclinic-admin-server
içindeki Dockerfile'i kullan.

--force-rm Always remove intermediate containers
(Arası konteynları her zaman sil)

Docker çalışma mantığı

1- Base image'den bir konteynır oluşturur.

2- 2. adımı ekle image sonra konteynır oluşturur.

3- 3. adımı " " " "

Arada çakışan konteynır hata alırsa Docker bunu
silir. Bu konteynırı silmemesi istiyorsak

--force-rm ile ne olursa olsun silinebiliriz,

-rm'ı sadece -rm olursa hata olmadığı sürece
siler

Pipeline kullanacağınız için kodlu konteynirları
kalmasını isteniyorsunuz. O zaman -force-rm

-Djava, security, egd=/tmp/dev/. /urandom

⑨

Normale default'lı random, O yüzden you may want. Koch
şifrelerken random ve urandom device'larını kullanıyorsun.
Random entomik belirsiz hanekelelerden veri topları ve
led üreteç. Ancak konteynerde kloibüsünde entomik
entropi olmadığını için urandom kullanıyorsun. Yolcu entomik kiliflerde

MSP-8

* Docker - Compose

Birler - birde container'ı içeren bir şekilde göstermek için kullanılır. Docker Compose, container gösterimini declarative versiyonu.

* Depend-On

ilk ve 2. sırada çalışması gereken container'ı depends-on kullanarak ayarlayabiliriz.

Ana depend-on konteynırının çalıştığını kontrol etmet.

442-) Services (konteynır) --memory 512 r

docker run -p 8888:8888 petclinic-config-server:dev kontenru docker-compose service'ı olarak yazısını services:

and config-server:

image: petclinic-config-server:dev (image name)

container-name: config-server

mem_limit: 512M (memory limit)

ports:

- 8888:8888 (ports)

440-) version:2

Memory limit version:2'de var 3.4k yok. O yüzden version:1 kullanılır.

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

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

456) depends-on:

- Config-server

(Config-server'in sunucusunu
çalıştırın.
config-server'in up olması
gerekli.)

entrypoint: ["./dockerize", "-wait=tcp://
config-server:8888", "-timeout=160s", "--",
"java", "-Djava.security.egd=file:/dev/.turandom",
"-jar", "/app.jar"]

↳ "wait=tcp://config-server:8888"

Config-server'in hazır olduğunu/calıştığını kontrol et.
değilse bekle. Haarsa çalışmaya başla. Haarsa "--"
sonrasındaki komutu ("java") çalıştır.

"-timeout=160s"

config-server'in çalışmamasına rağmen 160sn bekle

Note: Dockerize tool'u bir konteynür'in çalışmıyor-calısmadığını
kontrol eder.

Note: docker-compose'daki entrypoint, Dockerfile'daki
entrypoint'in üstüne yasır. Önceliği vardır.

Note: "--" Docker ve Java komutlarını ayırsın.

526-) circuit-breaker (sigorta)

Birim mikroservisindeki servislerden herhangi biri çalışmıyorsa request'ları bu service gitmesini engelleyerek bir sigorta'ya ihtiyacımız var.

Bu sistem tracino-server oblayınya 2 tane yapıyor.
→ Requestler ulaşmasa sistem komple çökebilir.

Health Check: Konteynır çalışmıyorsa "unhealthy" durum
yeine yerisini değiştiriyor.

CAPSTONE PROJECT (3)

MSP-3

Unit Test'leri oluşturmak ve eklemek Developpenin görevleridir.

Maven'da microservice'in her bir ~~class~~ klasörünün altında main ve test isiminde 2 parallel klasör blınları. Maven'ın işlevlərinin bu klasörə səməsi yad kırıcıdır.

Assertion: Təhlükədə bulunan (Unit testte bu kullanılır) Funksiyalar sonucu bir dəqən olaraq gəriyir. Səmə beklerənə eftə Success, digərse Fail verir.

* **Plugin'ları** pom.xml'in Springboot kusmına ekliyiriz.

Plugins'ın altında hər hansı bir yere ekleyebiliriz. Unit Test yapabilməndiz üçün Maven JUnit Coco plugin'ını ekleməmiz yekli.

* Her bir microservice'in ana pom.xml'de **script** edilmiş keçidin "əzələr" pom.xml'i var.

* Mvn ~~script~~ plugin'ının script'i bir üst dəriyənə olursa (~~script~~)

> ..\mvnw clean test

komutunu çalışdırırız.

* **Büyük Test Jurnalında** Project'in altında site klasörleri düşer.

Bu sayfalarımızda web server olmazsa işin; sayfanın çalıştırılarak istenileninde simple http server modül üzerinde browser'un çalıştuğu raporu oluşturmak.

> `python3 -m http.server`

ve sonunu raporla

* **Code Coverage Reportu Hizlamlanmış** [↑] Bölm̄ Sınıflarına

⇒ **BÖLÜM 2) QA TEST (MSP-10-19) (Nightly Job)**

MSP-10

* **Functional Test (Automated)**

Bu testler test ekibinin sorumluluğunda. Bütün işler developer'ların.

QA Test = Quality Assurance Test

Selenium hem java ile hemde python ile çalışır.
⇒ Burada bizim sorumluluğumuz; Test tool'u olan Selenium'ı Devops environment'a entegre etmek.

* **Selenium Pool**

Selenium'in kümeleri işin gerekli olurlar;

> Chrome browser

> Chrome API driver

> Selenium (Python Selenium)

(Kutuphanesi)

Buradaki konuları:

exploitnak çok mantıklı

* TestBench metodu test otomasyonu service' e özel yazılmış
durum.

* Automated Functional Test'ler sayfa üzerinde
otomatik bir şekilde seranaya izinden yapsın.

* Selenium'u konteynır'da çalıştırın ve deki
zalimlikle oblige'ci bu testi eğlensin.

01.30 MSP-11

MSP-11

Infrastructure' ~~test~~ ^{ayrıca kılavuzda} Terraform ve ansible kılavuz
kubernetes ve Docker ile hazırlanıyor ^{ayrıca kılavuzda}

Bu lüsimde jenkins-server-cfn-template.yml adı
kullanarak cloudformation servisi üzerinden Jenkins Server'i
kurulur.

MSP-12

Jenkins Server'a plugin'ları yükles.

Plugin'ler;

- > Github Integration
- > Docker
- > Docker Pipeline
- > Jacoco

* Image'larındaki Jenkins'ın Docker'ı yönetmesini "Cloud Agent" olarak tanımlamaya çalışırsınız.
 Manage Jenkins → Manage Nodes and Clouds →
 Configure Clouds → Add a new cloud → Docker
 (Docker Plugin'ı sayesinde çalışır) → Docker Cloud Details →
 "tcp://localhost:2375" (cfg dosyasında docker'in
 servis dosyasında sed komutu ile değiştirilebilir ve
127.0.0.1:2375 şeklinde değiştirebilirsiniz.)

* GaloisTracepm'ız jenkins'ın app'leri build etmek için bir nodejs'e ihtiyaçımız var. Jenkins Server'da build edeceğimiz her uygulma için tool'ları
 yüklemekle uğraşmamak amacıyla bu tool'ları container'ı
 kerta çalıştırıyoruz.

Bu cloud yapısını bu şekilde yapılıyorsunuz.

Docker container'ı Jenkins Server üzerinde çalıştırıyor.
 Jenkins 2375 portundan Docker Daemon ile haberleşecek. Bu sonda da
 job'larımıza Docker Container'lar içinde çalıştırabileceğiz. Docker

* Sed komutu Cloud ismini verdığınız yerde çalıştırın.

```
> 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
```

* /lib/systemd/system/docker.service içindeki satırın
değeri yerine önceliği hali;

ExecStart=/usr/bin/dockerd -H fd:// --containerd=
/run/containerd/containerd.sock \$OPTIONS.

\$DOCKER_STORAGE_OPTIONS \$DOCKER_ADD_RUNTIMES

* Yerine yazdığınıınız satır;

ExecStart=/usr/bin/dockerd -H tcp://127.0.0.1:2375
-H unix:///var/run/docker.sock

MSP-13

Discard old build'si segmeyi olıskorlik haline
getirmek için Jenkins Server'in storage'ı doldurmak.

* /feature** feature ile başlayıp herapı bir branch

992-993. satırlar

* docker run --rm -v \${HOME}/.m2:/root/.m2
-r `pwd`:/app -w /app maven:3.8-openjdk-ll
mvn clean test

(A)

HOME: /var/lib/jenkins

pwd: /var/lib/jenkins/workspaces/petclinic-ci-jpb/

Amaçımız "mvn clean test" komutunu çalıştırarak tüm
ortakları rm kumak. mvn yüklenen yerine bu işlemi
mvn oben bir docker container'da yapmak daha mantıklı.

- rm ana konteynürden temizli olursa sil.
-v \$HOME/.m2;/root/.m2
mvn clean test hibernate sonrakıda m2 klasörü
oluşturdu ve buanda düşen hersey instance'ın
önemi yok. \$HOME=/var/lib/jenkins/.m2'ye
geçerseki, Püm artifact'ler instance geçerse.

-v 'pwd': /app
job çalışmaya başladığında github repo'sunu
tüm klasör ve dosyaları, pwd=/var/lib/jenkins/
workspace/petclinic-ci-jobs/ attıra download
edilecektir. Bu volume tanımı sayesinde bu
klasör ve dosyaların tamamı aynı zamanda
instance içindeki /app klasörine geçerse.
-w /app working directory olarak set et
mvn: 3-81-openjdk-11-image
mvn failed test working directory'de huru oluştur.

* Jacoco code coverage report

Inclusion = Code coverage report'a dahil edileceğiniz dğalar

Exclusion = " dahil edilmeyeceğiniz "

Change build status --- = Stable / Not stable durumu

always run coverage --- = Build fail olsaya veya yarında
kaçılık de report'u oluştur. Bunu segmentler içinde
Success'te oluşturular.

fail the build if coverage = Bir önceki build ile karşılaştır
ne belirtilen oranda düşüs varsa bu job'ı fail olarak
ipucu

* webhook oluşturmak için branch önemli değil,

Github trigger her push'u bibliotek anıks Jenkins
biçim belirttiğiniz branch'leri dikkate alacak.

Trend

* Jacoco Coverage Report 'da işin işin'si
sayı tarafı olusan grafiğe tıklayınız.

* stable - unstable

class'in dependency to-to sırası

0-50 arası fail

50-70 " unstable

70 < x stable olust kabul edils

MSP-14 (Creating Docker Registry for Dev Manually)

İlk kez Jenkins'ı yapacağınızda öncelikle image'ları
kendiniz oluşturacaksınız.

'pwd'= pwd komutunu çalıştığı yer. Bu job'un
/var/lib/jenkins/workspace/~~petclinic~~ petclinic-ci-job

* Create ECR Repo

PATH="\$PATH : /usr/local/bin" -----> whereis aws

APP_REPO_NAME = "clawway-repo/petclinic-app-dev"
^{local}

AWS_REGION = "us-east-1"

aws ecr create-repository \

--repository-name \${APP_REPO_NAME} \

--image-scanning-configuration scanOnPush=false \

--image-tag-mutability MUTABLE \ -----> overwrite
izin vermek

-region \${AWS_REGION}

MSP-15 (Create a SA Automation Env with EKS-Part-1)

Bir policy oluşturmak istiyorsak AWS IAM Visual Editor'ı kullanabiliriz.

EKS Master ve worker'ları ne yapmasını istiyorsak bu yetkileri policy olanak instance'lara vermemiz gereklidir.

Master'a verdığınız policy'ler, EKS Cluster Policy yapısına çok benzeyir.

* roles.tf

Bu dosyanın başında provider bilgisi yok cihku' bunu module olarak kullanacağımız.

```
policy = file("./modules/IAM/policy-for-master.json")
```

↳ Burada module'ü oluşturduğumuz path'i verdik
assume_role-policy bölümünden harpi service role
vermek istiyorsak onu yazıyoruz (service = "ec2.amazonaws.com"
aws_iam_policy_attachment Bir role ile bir policy'yi
birbirine bağlamak için kullanıyoruz. (role.name ve policy.name)
AWS'de terraform ile role'ü service'e direkt bağlayamıyoruz.
Bunu profile bağlanamız, service'de profile'i bağlamamız gerekir.

* main.tf

```
module "iam"
```

```
source = "./modules/IAM"
```

21

Module'ü daha sonra
kullanmak için tanımladık
ve path'ını portekdik. Path
merin'den bağlayarak
zehibe oluşturuyoruz.

K&S Cluster için Ubuntu Makineler araya kalkıyor. Daha sonra Rancher toolunu kullanıyor. burada port'ları ona göre belirliyoruz.

`iam-instance-profile=module.com.master-profile-name`

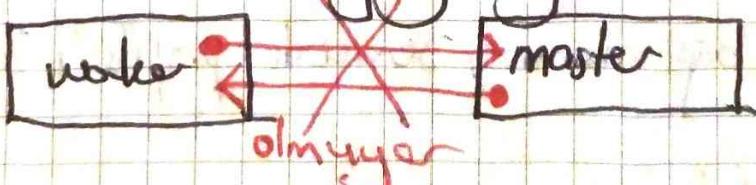
`subnet_id=default VPC'de belirttiğiniz AZ'in subnet'i olması gerekiyor. (Örneğin AZ'de us-east-1a ise onu seçelimiz) 1b/1c`

`key_name="mattkay"` Bu key adını ilerde değiştireceğiz, tag kısmını ansible'da kullanacağız.

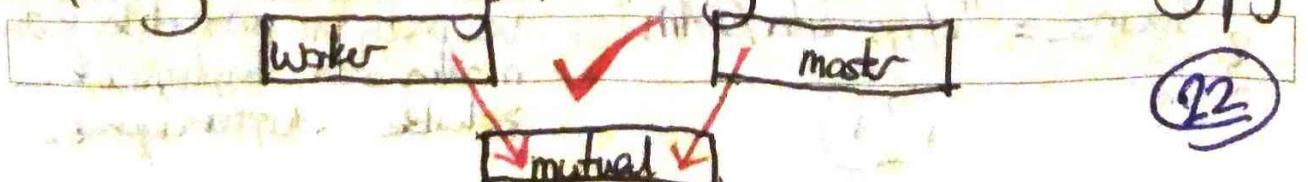
III. Security Group'lar Neden 3 tanı

Master'ın ve worker'ların belirtilen port'lardan sadece birbirinle konuşmasını istedığımızda aynı anda birbirine atayamıyoruz. Sistem kırıdalye giriyor.

Tek tanglı clavuz worker'in portuna sadece master'e gelis olsun veya master'a sadece worker'den giriş olsun diyebildik. Ancak aynı portları/sec.grp'ları karıştırıktan birbirine bağlayamıyoruz. Çünkü biri diğerinden önce olmuyor.



Bu sebeple mutual adını verdığınız 3. bit sec.grp oluşturuyoruz ve yönlendirmeyi onun içinden yapuyoruz.



MSP-16 (Create a SA Automation Env with K8S-Part-2)

Burada MSP-16 branch'ini push yapmasa da hata alırız.
AWS Console'da key,pem'leri silerek bile Jenkins WorkSpace'ın
icindeki silinmiyor.

Sed -i "/mattkey/\\$ANS-KEYPAIR/g" main.tf
main.tf dosyası içindeki tüm mattkey kelimeketi:
\$ANS-KEYPAIR ile kalıcı olarak değiştir.

* AWS CLI ile key pem oluşturma

```
aws ec2 create-key-pair --region ${AWS_REGION}  
--key-name ${ANS-KEYPAIR} --query "KeyMaterial"  
--output text > ${ANS-KEYPAIR}
```

* Jenkins User'ına bağlanma

sudo usermod -s /bin/bash jenkins

↳ Normalde Jenkins'in shell'i root.Bu sayede oluşturuyoruz.
Güvenlik Jenkins bir sistem user'i.

sudo su - jenkins

↳ Jenkins user'a git ve onun home'una git/alias

Jenkins HOME = /var/lib/jenkins

* DA Test Kapılamında MSP-18'e (Bekliden fazla elimizde)

- > 3 instance aynı anda çalışacak formatta deneyimler
- > Makineleri K8S Cluster'a dönüştürün ansible playbook
- > Uygulamalarınızı mikroservisi geliştirerek K8S manifestları olarak

> Burada jenkins testleri yorumlu ve kapatıyoruz.

* hostname konutunu deneme

AWS-KEYPAIR = "callable-test-dev.key"

Ssh -o UserKnownHostFile=/dev/null -o

StrictHostKeyChecking=no -i \${WORKSPACE}/ubuntu/appname-for
hostname

→ StrictHostKeyChecking=no (HostKeyChecking yapme)

→ UserKnownHostFile=/dev/null (Cinkey bilgiyi silme)

Bu kullanıysa known host
dosyasına kaydetme. Makine başlangıç yerlerini bu dosyaya
kaydetir. Bu da doryadır yorumlu yapar. /dev/null'i
topradanma gibi düşünebiliriz.

→ \${WORKSPACE} = workspace, Jenkins'in terminali
değerlerinde bin.

Burada /var/lib/jenkins/workspace/Host... = workspace

* Dynamic inventory ile ping etme

export ANSIBLE_INVENTORY = "\${WORKSPACE}/ansible/
inventory/hosts.ini

↳ ilk obrak ANSIBLE_INVENTORY dosyasını tamponuz.

Aynı şekilde ANSIBLE_PRIVATE_KEY-FILE ve ANSIBLE_HOST-
KEY-CHECKING'i de belirtmelisiniz.

* dynamic-inventory.yml

us-east-1 (oluşturduk) makinelerimiz.

→ filters:

- top : Project : k8s-kube-ars

- top : environment : dev

} Bu ixtiraklara fithele

↳ Bu top'leri mənənəf obyektib verməstik.

→ keyval groups:

- key : tags ['Project']
prefix : all_instances

ilki grup = aws_ec2

grubun adı = all_instances_k8s-kube-ars

- key : tags ['Role']
prefix : role

grubun adı = role-master
role master role-worker

* dynamic-inventory'yi graph file görme

ansible-inventory -v -i ./ansible/inventory/cluster-stack...-graph

↳ -v = verbose (hata olursa detaylı göstər)

* Cluster Configuration File (K8S Cluster Configuration)

Clusterlərin default sistemlərin düşündə ayar yapılmaları
buyle bir obyekti kullanıbiliyoruz. (Cluster Configuration Object)

Control Plane Endpoint otomatik olaraq sonradan pekələ
Master instance'in private ip'sini alıbæk. (Pipeline)

cloud-provider disardan bu hizmeti alıbmama, belidirness
Load Controller Manager'ın "gələcək durumda obyekti" kimi belidir.
podSubnet : 10.244.9.0/16 Flannel'i çalışdırmaq üçün
perdili olan CIDR bloğu

enable-aggregator-routing : "true" Birdeñ fəsək master
oldığında pelen trafiği route etməsi üçün enable edilir.

Init Configuration: Cluster'in runtime'ında depuHDD'nin
istemleri buraya yazıyor.

NodeRegistration: Cluster'in yeni node şenasi zamanı
ayarlannı tutulduğu yer.

KubeletConfiguration: Kubelet ayarlannı tanımladığınız
yer.

cGroupDriver: systemd LSS driver obruk depuH'da
cgroupfs'la kullanıyo araci kubeadm kullanımas
diumunda systemd driver seçilmesi zoruntu.

* Storage Class

Dinamic olarak volume/storage oları ayarlıyor.

volumeBindingMode: WaitForFirstConsumer

Bir pod oluştur ve bağlı olmalıdır mode bağları. Dope
seçenekle immediate.

allowedTopologies: Zone ile alakalı kısıtlıysa belirttiyors.
Bana ws-east-la da oluştur dedik.

* ansible playbook (k8s-setup.yaml)

~~-all-~~ swap off kubelet'in performansı artırmak için swap'ı
kapattırıyor.

change the Docker Group Docker'da sistem kullanıcıyı değiştir.

~~-master-~~

get gettest base envsubst konumunu tutanabilmek için
gerekli

→ envsubst (environment substitution)

export CONTROLPLANE_ENDPOINT=... master'in private IP'si gibi.

envsubst </home/ubuntu/clusterconfig-base.yml> /home/ubuntu/clusterconfig.yml

↳ Master node'daki clusterconfig-base.yml dosyasını, clusterconfig.yml'a eklerse ve bu sırada CONTROLPLANE-ENDPOINT'in değerini yerine yaz.

Burada toplamda 3 dosya var:

- 1- key ve value degerinin belirtilen clusterconfig-base.yml
- 2- envsubst komutunun uygulanması, playbook
- 3- Sonuç olarak clusterconfig.yml

→ kubeadm init --config /home/ubuntu/clusterconfig.yml

Cluster'ı bizardaki config ayarlarına göre başlat.

→ register join command for workers

Normalde master'da oluşan değişkeni worker'da kullanıyor.

Bunun için add-host modülü kullanılıyor.

kube-master'ı host olarak ^{ve} worker-join parametresini
değişken olarak eklediğim için artık sistem bunları püskür.
Artık hostvars'ta bunları kullanabiliyoruz.

-worker-

- name : join workers to cluster
- shell : "shoptuars [\"kubes-master\"] [\"worker-join\"] \{\}"
- register: result-of-joining

Bir önceki bölümde oluşturduğumuz depolarları kullanarak worker'ları master'a join ediyorsun.

kube-master host'a git ve worker-join depliği ol.

-master-

Patch the Instance

Cloud Control Manager'in instanceları bulabilmesi için yapınız.

Normalde komut şu gibi olur;

```
> kubectl patch node <node-name> -p "{'spec': {  
    "unschedulable": true}}"
```

↳ node'u describe ettiğimizde gördüğümüz "spec" depliği
değiştirmemiz gereklidir.

Ingress

Mükemmel yapısında, tek bir name ile birbirlerinden
seçmekte kullanılabiliriz. Aksi takdirde her biri içeri
Load Balancer kullanacaktır.

Deploy Aus CSI Driver

StorageClass volume'ların oluşturulması için bu driver'i
kullanıyoruz.

ALSP-17 (Prepare PetClinic Kubernetes manifest file)

Kompose dosyaları Docker-Composeyaml dosyaları.
K8S manifest yaml'larla dönüştürülüyor.

Kompose hizmeti servisiniyor. Bu komponenin oluşturularak
varsa bazı düzlemler yapmamız gerekmektedir.

* docker-compose.yaml

config-server:

image: "{{ .Values.image-Tag.confis-server }}"

↳ Helm values.yaml'dan belirtilen image'i çek.

regcred (register credentials)

K8S'de image'i Docker-Hub yeine farklı bir registry'den
ekluyorsak bize credentials lazım. Bize de ECR kullanıyoruz.

regcred esasen bir secret. Bunu oluşturup credentials'i
oradan almasını sağlayacağımız.

* helm create petclinic-chart

Bu komut ile K8S katalogu içinde helm chart oluşturuldu.

Default dosya ve değerleri silip kendimiz kendi yaramalıız.

Values.yaml dosyası value'ler için öndeki.

templates altındaki dosyaları sil ve

> kompose convert -f docker-compose.yaml -o petclinic-chart/
bu komut ile docker-compose.yaml dosyasında bulunan

bilgileri istenilen manifest yaml dosyalarını templates'in
altında oluşturur.

Note: docker-compose.yml'de sadece API-GATEWAY
altinda "kompose.service.expose": "80", "VALUES.DNSNAME": ""
kullandığında sadece API-GATEWAY için expose olmaktadır.
Yine sadece onda nodeport servisini açtı ("kompose.
service.type: "nodeport" sayesinde)

* InitContainers

Pod'un altinda yardımcı bir konteynır oluşturarak,
service'in çalışıp çalışmadığını görmek. Daha
service kalkışında içinde bulunduğu pod'a ek konteynır
kalkmasına izin veriyor.

Containers içerisinde resources: {} altta eklenir.
Containers ile aynı hizarda olacak.

image: busybox Bu işler için kullanılan bir image

command: ['sh', '-c', 'until nc -z config-server 8888;
do echo waiting for config-server; sleep 2; done; ']

sleep komutu gir

-c yanın komut diziini yap

nc (netcat-networkcat) TCP/UDP bağlantıları oluştur

göndermeden dinler. -z sadece dinlene yap

until nc -z ... Bu server'i bilsey gondermeden 8888
portundan dinle

Sonuç olarak "Waiting for config server" yarılır.

2 sn'ye bekle ve bu sürede config oturum kader dönüye devam et.

Daha sonra config oturum obsevatorie döngüden silinir.

spec:

IngressClassName: nginx

Burada nginx ingress controller kullanısmamız için ekliyorum.

rules:

-host: '{{.Values.DNS_NAME}}' Buyle hostlere
Olmalı.

→ values-template.yaml

Helm de value kullanımı için değerleri value-template.yaml
ficisine yazdırınalıyz.

→ Set up on Helm v3 chart

Chart'larımıza kayabilmek için bir S3 Bucket oluşturmalıyız.

→ Install Helm S3 Plugin

Helm S3'ün Jenkins üzerinde çalışabilmesi için
örade da yüklenen gerekliden yoksa Jenkins, S3'ler chart'ları
gerekliyor.

→ yapımladan önce

(3)

* # 2350. satırı 'ec2-user'a geri dönüştür

* AWS_REGION=us-east-1 helm s3 init S3://.../stable/nginx

Bu bucket'ı berim helm repository yap. Helm'ı S3'de indirip
edipuzz. initialize olunca libsovr içinde index.yaml oluyor.
Bir chart'ın olmasa, olmaması index.yaml ve spf13 chart'larıdır.

Notes Initialize ettiğimiz repo'yu eklenmesek client machine bunu görmezdir.
... helm repo add ... ile ekliyoruz.

* Chart versioning

chart.yaml dosyası içerisindeki version ve appVersion'u değiştirmiyorsa chart versiyonu ile etkileşime girmemeli. chart versiyonu ile etkileşime girmemek için version değiştirmek yeterli.

* helm package getlink=chart

chart dosyası dosyalarını aldığı yerde ^{k8s altında} yükleyen ^{appVersion} getlink=chart dosyalarının

~~getlink=chart~~ klasörünün içinde zip file oluşturdu.

* Store Helm packages into S3 Helm Repo

HELM_S3_NODE=3 AWS_REGION=us-east-1 helm s3 push...

HELM_S3_MODE=3 sonun olması durumunda force kullan.

* HELM_VERSION

Jenkinsfile bulundığında versiyonu manuel yazanızı ^{git commit} etmeniz lazımdır. (version: HELM_VERSION)

MSP-18 (Prepare a CI Automaton Pipeline for Nightly Build)

! docker run --rm ...

--rm ile container'a işin bitişe kadar devam eder.

! package-with-maven-container.sh

image'ı dinamik olarak almak için oluşturuluyor.

-> MVN-VERSION Bir değişken oluşturuyor.

! \${WORKSPACE} Default değişken. .cbb'ınca çalıştırılmıştır.

! \${ECR_REGISTRY}

! \${APP_REPO_NAME} > Bu değişkenler Jenkinsfile'da vereceğiz.

! \${BUILD_NUMBER} Default değişken. Build sayısı.

MVN-VERSION = \${WORKSPACE}/spring-petclinic-admin-server/target/maven-archiver/pom.properties && echo \$version

↳ "source" kodunun başka şekilde yazımı. Bu şekilde ./. pom.properties dosyasını çalıştırıyoruz.

! \${WORKSPACE}/spring.../pom.properties çalıştırılacak dosya

↳ target klasöründeki package obran sonra oluşuyor.

↳ pom.properties'in içinde groupId, artifactId, version var

echo \$version pom.properties içinde oku

Jenkinsfile, çalıştırınca bu değerler values.yml'a dinamik olarak gidecek. values.yml'den paketlenerek biner. Sonra helm'e push edilecek. Chart'ınız hazır. Çalıştırdığımızda pod, deploy, service vb. çalışacak.

* Create ansible playbook

aws ecr get-login-password --region \$AWS_REGION |

docker login --username AWS --password-stdin \$ECR_REGISTRY

↳ Bu komut gizli şifre \$AWS_HOME/.docker/config.json da saklı oluyor.

config.json da şifre okunuyor. Credential for oraya gelip ECR

* Deploy petclinic application

kubectl delete secret regcred -n petclinic || true

↳ regcred isimli bir secret varsa sil. Yada ejer istenene devam et (|| true)

secret'i komut satırından oluşturken şifre ekliyoruz.

--type=kubernetes.io/dockerconfigjson

↳ Bir container image'ine bağlanıprak amacıyla credential'ın

secret içinde depolamak istersen bu type'i kullanıyoruz.

--from-file=.dockercfg.json... tipi bunaşık yazıyoruz

helm upgrade --install hedef dosyadan içeriğin varsa

calıstır ipsa da obyaşrı okuttur ve calıstır chart

helm repo add S3 'i helm repos olarak ekliyor

helm install sadece repo içindeki chart'ı salistirıyor

* dummy-selenium-test.html.py

on google.com base url'de "I am Feeling Lucky" yazın

bulunsu success mesajı.

* py-run-dummy-selenium-jobs.yaml (fileglob)

fileglob bir ansible modülüdür.

{} Item 3) dediğimizde bu sefer argumentları sıralamıştıkda ... özyarısı ıgħabeli turn özyakta uggula demiş oluyoruz

Örn: with-fileglob:

- "/playbooks/files/fccapp/*"

(fccapp klasörü
alttakki tum
özyakta)

workspace: Birazdan ansible konutumda bussu tanıtacağız.

Bizim örnekimizde;

with-fileglob: "{{ workspace }}/selenium-jobs/dummy.py"

↳ burada dummy ile başlayıp tüm py dosyalarını galistir.

* run-dummy-selenium-jobs.sh

"workspace" {{ workspace }} normal workspace'ı değiştik
--extra-vars

* pb-run-selenium-jobs.yaml

--env MASTER-PUBLIC-IP = {{ master-public-ip }}

↳ testi yapabilmek için masternode'un 3001 portuna bağlanmas, lösüm. Testi belli dosyelerimizde yapaloqlı, dummy'ddu gibi google'da yapmayızaqlı içi henu vermelidir.
with-fileglob: " -- /test*.py" test ile başlayıp .py ile bitti bütün dosyalar galistir.

30001. definition public IP'ni'z buraya gelecek
ve bu port'ta yapacağınız testi

* Jenkinsfile-patchclinic-nightly

1- Create ECR Repo

2- Packaging App - create -jr file

3- Create TAAS

4- Build App Docker Images

5- Push Images to ECR Repo

6- Create Key Pair for Ansible

7- Create GVA Auto. Infrastructure 3 instance

8- K8S setup ile instances configure ediliyor.

helm'de kubernetes image olarak value'ye gönder

S3 bucket', helm repo olarak configla

helm package ile chart kindeli yaml'ları paketle

S3 push

ansible deploy yaml → yaml dosya ile deployment'la
service'le oluştur

9- Selenium test'leri yap

10- Terminate

* Pipeline

env.IMAGE-TAG - "...", "variable": "Groovy Dev")
L) env ile tüm instance'ta kulanılabilir. env olmasa
sadece obya içinde geçerlidir.

MVN-VERSION = 'o ----- && echo \$version'

L) pom.properties içinde version deyimi olsın.

↑ 'aws ec2 wait instance-status-ok --instance-ids \$id'

L) mühendislerin instance checkleri tamamlandıktan
sonra yap.

* Test The Application Deployment

> sh "curl -s \${MASTER-PUBLIC-IP};3001"

L) selenium testlerini yaparken sayfa gelip mi kontrol
edilmesi.

Her 5sn'de bir "Couldn't connect" oluyor.

* Run JTA Automation Test

⇒ --connection=local Bu playbook'u Jenkins Master'da çalıştır
--Inventory 127.0.0.1 Localhost'a ekləstir.

MASTER_PUBLIC-IP test dosyalarında testlerin master'ın
yanıbaşının belirtildi. O zaman master'in
IP'sini alırız.

* Destroy Image, ECR Repo, Cluster, Key,pem

MSP-20 (Prepare Build Scripts for Dev Env)

- MSP-20 ve 21'de Payalı olan cluster'ın no uygulanamayan sorunları, yükleyeceğiz.
- Genel manada bu bölümdeki çoğu işler petclinic-nightly ile aynı olacak.
- ECR oluşturma işlemini 1 kez yapacagınız işin manuel yapabiliriz. SA admalar için oluşturuyorsunuz.
- ⇒ deploy-app-on-qa-environment.sh
- > helm 3.3 push --force petclinic-chart --
↳ burada push, aynı isimde bir chart varsa üzerine yazabilisin anlamsız felicity. Bksa hata verir.

MSP-21 (Build and Deploy App on QA Env. Manually)

- ⇒ build-and-deploy-petclinic-on-qa-env-manually.sh
- ⇒ key.pem 'sinde "sudo ls /var/lib/jenkins/.ssh"' içinde olması gerekiyor
Jenkins Server bunu K8S master'a bağlanacak ve istenileni yapacak.

⇒ İşlemler

- 1) Packaging with maven
- 2) Preparing TAGS
- 3) Building Images

- 4) Pushing Images to ECR
- 5) Deploying App on K8S Cluster
- 6) Deleting All Local Images

★ config.json

★ config.json
ECEI login configuration (username, password), HOME folder
config.json allows ECEI's pull/push rich libraries.

★ AWS_REGION=\$AWS_REGION helm upgrade --install petclinic app-release stable-petclinic/petclinic-chart --version \${BUILD_NUMBER} --namespace petclinic-qa

"helm upgrade --install" komutu "petclinic-app-release"
İstediğiniz release versiyonunu update eder yada clusterin.
"helm -install" sadece oluşturur.

"stable-petclinic/petclinic - chart --version \${BUILD_NUMBER}"
dazu chart überinden releaseslug-für-local

> helm repo ls

> helm fs -n petclinic-qa release 'lex' poster.

MSP-22 (Prepare a CI Pipeline)

21. bölümde environment'ı kurduktan sonra testler ettiğiz.

Daha sonra herhangi bir docker environment'ı oluşturabilmemiz isteceğiz. Dödayanya bin bir docker otomatik yaratmamız gerekmeyecektir. Bu nedenle pipeline kuruyorum.

* ssh key.pem'ler nerede?

> sudo ls /var/lib/jenkins/.ssh/

* Internet sayfasını görmek için;

> sudo usermod -s /bin/bash jenkins (jenkins'in shell'i)

> sudo su - jenkins (jenkins user'ına ve dosyalarına genel erişim)

→ maskin "ssh -i" si ile master-node'a bağlanı

> kubectl get ns

> kubectl get ingress -n petclinic-ns

→ route-53 Hosted Zone'ı git. Vorhe/Rade Projekti

→ simple record → define simple record → Alias to Network

Load Balancer → Choose network load balancer kisminda

Address hedefiye onu Seçipus → create records

Gelmeyince K8S Cluster'da bir yer ekstik tekrarsta.

MSP-23 (Popore High Availability Role k8s Cluster on EC2)

Rancher Pod'ınıza güvenliköz

→ Rancher'a, diğer servisler üzerinde işler yapabilmek
policy ^{EC2 ya} tanımlayorsun.

Bu policy'leri bir role attach ediyorsun.

Role'ü de lisans ^{graba} veya malzeme atamıs. Buradan
Rancher Instance'ına atıyorsun.

- Rancher Instance'sıza ayrıca bir security group oluştur.
- Jenkins Server üzerinden (ssl) Rancher'a bağlantı yapmak
gerekli. Bunun içinde Rancher'a key-pair tanımlayıp olsun.

* Load Balancer SecGrp

Load Balancer SIGN 80, 443 anywhere yapuyorsun.

* Rancher SecGrp

* INBOUND

80 ALB ile konusmak

443 Rancher UI or API

22 Any Node IP

6443 Jenkins Server, k8s kullanmak için Docker Com.

* OUTBOUND

All top → ^{call} rke-cluster-09 Kendi kendiye olacak
seçimde bir sec.grp var.

Kullanımız cluster Sec.grp içinde bir den fork service
var. Buna göre bilinçle den network'lu tespit ediyorsun.

* GPG key'in olduğu dosyayı siler,

> cd /usr/share/keyrings/docker-archive-keyring.gpg

Note: Linux'te bilgisayarın IP'sey kırıltadan sonra oyun
por haliini alabilmek için fehér update etmelisin.
> sudo apt-get update

Note: User Name ALB anaçtpuyla (Port:443) roncher'a
parametri için, Listener, Target Group ve ALB oluşturur
mari streluyor.

Healthz

Bu bir API servisi. Aşağıdakilerde 200 kodunu
göndereyor.

* ALB

Web trafficinde hizmet verebilmesi için port 80
(HTTP)
hemde 443 portu.

HTTP'ın default action'ına bşşey yapanın enklu
HTTPS'e yönlendiriyor.

42

* Helm Uninstall

helm uninstall roncher -n <namespacenamı>

* mkdrr -p n/kube

Bu komutu şahitlerin varsa kube-config roncher
cluster.yaml dosyalarını workingdir'e getirir.

Bu bizim kubernetes'e panzer olmamızı sağlıyor.
kube-konfig dosyalarının altına fastidionda kesteli; kalkarlıyoruz.

* Installing Rancher

```
> helm install rancher rancher-latest/rancher  
  --namespace cattle-system  
  --set hostname=rancher.murataynali.com  
  --set tls=external  
  --set replicas=1
```

* Pod'ın log kontrolü

```
> kubectl logs -f rancher-... -n cattle-system  
    <pod-name>           <namespace>
```

* Rancher Password Reset *

```
> export KUBECONFIG=~/.kube/config  
> kubectl --kubeconfig $KUBECONFIG -n cattle-system  
exec $(kubectl --kubeconfig $KUBECONFIG -n cattle-system  
get pods -l app=rancher | grep '1/1' | head -1)  
awk '$2 print $1') --reset-password
```

* Rancher ilk şifreyi öğrenme

```
> kubectl get secret --namespace cattle-system  
  bootstrap-secret -o go-template='{{.data.bootstrap}}'  
  {{.data.password}}  
  {{.data.base64decode}}
```

MSP-216 (Prepare Nexus Server)

MSP-25 (Create Staging and Production Environment) with Rancher

Rancher'da cluster oluşturup uygulamamızı deploy edebileceğiz

* Nexus bir Artifact Repo. Docker Image'ları, Helm Chart'ları veya maven'in oluşturduğu jar file'ları tutabiliyor.

* Jenkins, Rancher, Rancher'da Cluster'ı yönetiyor.

* k8s container orchestration tool'ları, Rancher Cluster orchestration tool.

* Kuracağınız pipeline'ları

→ Helm chart'lar S3'te depolansalar,

→ Rancher cluster'(3 Node) kuracağım

→ Rancher uygulamamızı kodları, cluster'a yükleyeceğim.

→ Araç biz Artifact'ları Nexus'a gönderip, S3 Tek Repo yerine Nexus'tan almasını sağlayabileceğiz.

* Rancher di'yi Jenkins server'a kuracağım, Server'dan verdiğimiz komutlar (rancher kubeconfig...) ile Rancher'i görebileceğiz.

* nexus-server-terraform.yml) Cihaz

* Nexus'ta bizim oluşturuyoruz. Bu volume
oluşturuyoruz.

> docker ^{volume} Create --name nexus-data

* Nexus Server'i (container) run et

> docker run -d -p 8081:8081 --name nexus -v
nexus-data:nexus-data sonatype/nexus3

* Null Resource

Nexus'in boylanmak için bize bir initial password
gereklidir.

- o UserKnownHostsFile = /dev/null (Gelecek IP'yi ekleme)
- o StrictHostKeyChecking = no (Host Key Checking yapma)
- ec2-user@{:...} Instance'a bağlanmak için public IP'yi al.

⇒ 'docker copy nexus:/nexus-data/admin.password admin.password'

↳ nexus container'in içindeki nexus-data klasöründe bulunan
admin.password'ü, admin.password olarak kopyala

&& cat /home/ec2-user/admin.password'

(nexus server)

↳ instance içine alınan admin.password'ı oku

> initialpassword.txt"

↳ initialpassword.txt olacak. yosdr ve benzeyen
(jerkins server)

* Nexus Anonymous Access

Nexus'a peki bir kişi sign-in olmadan bilgileri
görebilen mi? (Double-clickenin)

* om2 local repo klasisini oluşturmak için:
> mvn clean tomcat salistirabilice.

* Nexus setting.xml

Buraya private ip'yi yazınak, birsey indirmek
istediğinde öncelikle nexus repo(om2) ye bakana
central reposu git denip obul.

* MVN pom.xml "distribution management"

MVN release ve snapshot reposunu burada tanıtmak.

Release repo, kodların son halinin saklanacağı yer.

Snapshot repo, developerların kullandığı ve ~~yer~~ paket
gönderildikçe versiyonlama yapılan yer.

* Nexus Repo kullanım

> curl -u admin:123 -L -X ~~POST~~ ^{repositoryler isinden al} <nexus-repo-name>/jar-
file-path > --output admin-service.jar 46

↳ Nexus'taki jar-fili admin-service.jar olarak indile

↳ browse → repo-name → package → admin → 2.1.2 → jar file
→ pack

Note: Nexus Repo'ya sadece 1 kez release gönderebilirsiniz. Ancak birden fazla gönderip üzerine yatanız istenmez. Server Administration and Con. → Repositories → Repo'yu tıkla. → Deployment-policy → Allow Re-dep by yourself.

MSP-25

* Rancher'in AWS ile konuşabilmesi için credential'ın eklemesi gereklidir.

→ Cluster Management → Cloud Credentials → Create
→ Amazon → Farklı bir isim → Access Key → Secret Key
→ us-east-1

* Rancher'da Template Oluşturma

→ Cluster Man. → RKE1 Configuration → Node Templates
→ Add Template → Amazon EC2 → Next → AZ → VPC → Next
Region + Credentials → Security Group Standard (Rancher yolu) → Network → Tom. None
Note: Template'ler sadece politikalardan node control plane ve worker node'ın konfigurasyonunu belirler.

MSP-2.7 (Prepare a Staging Pipeline)

* add cluster

* drain before delete

Node'ların bitti herhangi bir şekilde delete olursa
önce bunun içindeki pods, service vb. kaynakları baska
node aktarır sonra silip.

* Rancher'da Namespace Oluşturma

⇒ → petclinic-k8s-instance → Projects/Namespaces →
→ Create → Namespaces (project:Default)
→ Add name → Create

* Create Rancher API Key

⇒ Sürümle Avatar → Accounts API Keys → Create →
copy the credentials →
in a file

* Rancher API Key'i Jenkins Server'a Tanıma

⇒ Manage Jenkins → Manage Credentials → Jenkins →
→ Global Credentials → Add Credentials

* Jenkinsfile-petclinic-staging

* Rancher Context & Cluster ID

⇒ → petclinic-k8s-instance → Projects/Namespaces → petclinic-
staging-nr : → Edit YAML → 7.satır Project id'si al.

48

Rancher Context → Tamamı "c-d9enpn: p-8sf711"
CLUSTERID → : örexli kism → "c-d9enpn"

* Deploying App on Rancher K8s Cluster

* render login

--context olarak project id'yi farklı

--token olarak Bearer Token geliyor
↳ Access Key + Secret Key

* > rancher cluster & CLUSTERID > k8s/config

CLUSTERID'yi alıp config isminde bir file olarak
k8s klasörünün içine kayıyor. Bu sayede Jenkins'ten
donut webiliyoruz.

kubectl komutları, nasıl .kube'in içindeki
config dosyası sayesinde grabiliyorsak buda öyle.
Rancher komutları, şimdik içinde bu durum.

MSP-28 (Prepare a production pipeline)

1) feature /msp-28'e gel

2) Create k8s Cluster

Rancher UI → Cluster Manager → Clusters →
Create → Amazon ECR → PetClinic - K8S - instance

Note: Production ortamında normalle tüm worker
aynı instance'lar verilmesi etcd ve Control Plane bir gruba,
worker role ise başka bir gruba verilir.

Note: etcd neden 1,3,5 makine istiyor.

İki haneli rakamlar lider node'un seçilmesini
zorlaştırır. Tek hane olduğu zaman lider control
plane kolayca seçilebiliyor.

3) ECR oluştur (sononpush production'a true olur)

4) Amazon S3 oluştur

4-Jenkins Job'ı oluştur

* TAG oluşturma

Artık tag'ı her başında testing, staging vb ipuçları yok.

Note: Production ortamında data base cluster içinde
durmas. Bu neden için RDS kullanılır - Databaşka
bağlantısını yapmanıza gerek yok.



* Change .kube config file

Note - Halihazırda .kube altında bulunan config file, Rancher ile kendisine cluster'ı local olarak eklem yapmamızıza izin veriyor. (Sadece local cluster'a ait) config file'in içiniinden pod'a cluster'a hizmet edecek şekilde düzenleyebiliriz veya her bir cluster için ayrı config'ı oluşturabiliriz.

Bz. Şimdi yeni bir cluster(petclinic-prod) oluşturduk. Umarı böyle buna başlatabilmek için Rancher'dan local cluster'a ait .kube config'i alıp, halihazırda bulunan config'in içine dayanmış olursun.

Rancher → petclinic-cluster → sağ tarafta "Copy Kubeconfig"
* Create namespace > kubectl create ns petclinic-prod-ns
* MySQL deployment'si / Rancher'ten (Kontrol paneli)

Artık RDS kullanlığımız için mysql deployment'i q
üntüleme yok. Ayrıca mysql-service'inde ayarları RDS için değiştirmeliyiz.

RDS endpoint, cluster'in RDS'ye bağlanasını sağlar.

51

External

External Name Service'i, dışarıda bulunan bir bağlantı/endpoint'i cluster'a tanıtmaya yarayan

Bu service'i oluşturduğumuzda herhangi bir pod mysql-server'a gitmek istedğinde extname service'i秉ten, RDS endpoint'ına yönlendirilecek (Redirecting)

Note: Redirect vs Forward

Redirect bir bağlantı_digeste gönderir (Gibi)
Forward'ta bir forma olayı kişi konusufizik (Gibi)

* Jenkinsfile

Region / Rancher URL / Rancher Context / ClusterID
Rancher Context → Cluster → Namespace → petclinic-prod-ns:
→ Edit YAML → 7. satır

* Moving a namespace in project

: move → target project → default

* Deploy App on Petclinic K8S Cluster

+ Secret needed

Jenkins içindeki .dockers/config.json file'i alıp
bir secret oluşturmakla locum(petclinic-prod-ns)
K8S'in deploy sırasında docker ile bağlantıını(docker
daemon) bu yapılayacak.

* rm -f k8s/config /rancher cluster kf \$CLUSTERID >

Jenkins'in "Rancher kubernetes" ile bağlantı yapabilmesi
için config file'in güncellenmesi perakende Cluster'in
config file'ini buraya yapıştıracak.

> AWS_REGION=\$AWS_REGION helm upgrade --install petclinic-app-release stable-petclinic/petclinic-chart --version \${BUILD_NUMBER} --namespace petclinic-prod-ns --kubeconfig k8s/config

adıyla
petclinic-app-release'i, k8s/config file'ini kullanarak
petclinic-prod-ns namespace'si üzerinde, ... repor'ında,
helm upgrade --install oluşturular, stable-petclinic/
petclinic_chart --version \${BUILD_NUMBER} 'daki
rigid chart'ı alır, k8s cluster'ın içinde depoly et.

stable-petclinic/petclinic_chart --version \${BUILD_NUMBER}
daki 'Rigid chart', alır, petclinic-app-release adıyla,
petclinic-prod-ns üzerindeki cluster'a, k8s/config file'ni
kullanarak, deploy et.

5) What is DevOps

1- Pipeline

2- K8S Deployment

3- Terraform

MSP-29 (Setting Domain Name and TLS for Production Pipeline with Route 53)

- * instance'anda birki public ip'sini `node.muratgazis.com` a ver.
- * sayfayı kontrol et.

Note = gidişte erişim bir load balancer üzerinden yapılıyor,因此 sayesinde içeriği kırıcılar bloke forte endpoint'ı (service') expose edebiliyoruz.

* Subdomain ve domain name'in secure olmasının
symmetric encryption: Tek bir şifre ^{key} kullanılır. Bu nedenle
enkripsi hem de dekripsi kolaydır.

asymmetric encryption: Private key ve Public key
dimdeki üzere 2 tane varyant. Public key ile kriptolandırmak ve
private key ile dekript edilmesi fikri, yani hem de
bilgilimizi alabiliyoruz.

TLS/SSL Certificate: web sayfasi ekstra olarak kriptolene
taki tutulup ve certifikatlandırılıyor.

CSR = Certificate Signing Request

CA validate our request

* K8S Cert Manager Gereklilik

Sertifikaların oluşturulması

CA'ye bunların gönderilmesi

OA'dan gelen challenge'in doğru redetiminin yapılması

Sertifikaların głki alına yüklenmesi

* >kubectl api-resources

* Charge Api-gateway Ingress In Rancher

Rancher → public-cluster → ~~Service Discovery~~ Service Discovery →
Ingress → Api-gateway ~~annotation~~ : Edit YAML
annotation 5. satır \leftrightarrow (cert-manager.io/...)
spec 6. satır \leftrightarrow (TLS'ler sona kadar)

↳ same dğrxe heren bağlanıyor. kaybolduysa

* Prometheus ve Grafana'yı aranmak için; (80)

1) Services'in altında Prometheus ve Grafana'nın type'ini
"NodePort" yapıyor. Daha sonra verilen port ile web
sayfadan bağlanıyoruz

CAPSTONE PROJECT & TERRARI

Interview for Sahan:

Mikroservis yapısı kullanın bir uygunlama için Dug environment'i oluştur.

Build pipeline'ın production environment'a hizmet 5 tane pipeline ile build otur. Bupipeline her poekli oları environment'i oluştur.

* CI/CD - workflow

Dev Branch'inde bir sonrakı production'a ait en stabil ünitesi dener. Günlük 625 bananabli ünitenin alyp testlerini yapıyor.

* Build Pipeline

RC = Release Candidate

Her adım okunaklı en sonel pipeline. Uygunluk, compile, build ediyor ve Unit testlerini yapıyor.

* Devops → Automation

Note: En iyi galipacak pipeline, build pipeline. Hinde 6-8 kez çalışır.

(56)

Note: Unit Test, en basic ve mutlaka yapılma gereklisi.

* Functional Testler, Selenium gibi toolları otomatik yapılır. Bu testler hemen çalıştırılır. Dev Branch kullanılır.

* Manual QA Testler, hedefinin sadece belirli printlerde Release Branch kullanılır (Pazar 23.59) yapılır.

* Docker Compose

Bizim konteynır sayısına konutlarım olabiliyor.

```
> docker run --name config-server --memory 512  
  -p 8088:8080  
  petclinic/config-server:dev
```

* Unit Test

F = Fail

• = Success

* jacoco = java code coverage

Springboot kümnen altına düşüyoruz.

.. /mvnw test = ana pom.xml bir irtik oldugu için .. / yapayız. Sadece customer-service / test ediyorsun

* Jenkins Initial Password

```
> sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

* Runner obrak container kullanarak çok fazla