

Java Petclinic Uygulamasını Jenkins ile AWS EKS Üzerine Otomatik Dağıtma: Adım Adım Kılavuz

Proje Gereksinimleri:

Bu proje, Java tabanlı Petclinic uygulamasını Jenkins kullanarak AWS EKS cluster'ına otomatik olarak dağıtmayı hedefliyor. İşlemlerin sağlıklı bir şekilde yürütülebilmesi için aşağıdaki gereksinimler önemlidir:

- **AWS CLI ve eksctl:** AWS ile Kubernetes (EKS) cluster yönetimi için gereklidir.
- **Docker:** Docker imajlarının oluşturulması ve Docker Hub'a gönderilmesi için gereklidir.
- **Jenkins (min 4GB RAM):** Jenkins, CI/CD işlemlerini yönetecektir. SonarQube ile aynı sunucu üzerinde çalışıyorsa, en az 4 GB RAM'e sahip bir makine önerilir.
- **SonarQube:** Kod kalitesini analiz eden bir araçtır ve Jenkins ile entegre edilecektir.
- **kubectll:** Kubernetes cluster üzerinde işlem yapmak için kullanılır.
- **Git:** Kaynak kodunu almak için kullanılır.
- **Trivy:** Docker imajlarını güvenlik açıklarına karşı taramak için gereklidir.
- **AWS Hesabı:** AWS üzerinde EKS (Elastic Kubernetes Service) kullanmak için gerekli kimlik bilgileri (Access Key ve Secret Key) oluşturulmuş olmalıdır.

"Bu makalede bahsedilen tüm adımlara ve projeye aşağıdaki GitHub reposundan ulaşabilirsiniz: [Petclinic Java Uygulaması - GitHub](#)

Gereksinimlerin Kurulumu:

Jenkins Kurulumu: Jenkins'i kurmak için aşağıdaki adımları izleyin:

```
curl -s  
https://raw.githubusercontent.com/hakanbayraktar/ibb-tech/refs/heads/main/devops/jenkins/install/jenkins-install.sh | sudo bash
```

```
=====
Jenkins Kurulumu Tamamlandı!
=====

Jenkins Dashboard'a erişmek için:

http://146.148.90.85:8080

Giriş bilgileri:

Parola: 0ca0e2ffe8224d7ba5adc690e1becdb2

ubuntu@jenkins:~$
```

Jenkins arayüzüne erişmek için web tarayıcınızda <http://146.148.90.85:8080> adresini açın. İlk kurulum sırasında sizden admin şifresi istenecektir. Yukarıda Parola kısmındaki şifreyi ekrana kopyalayarak kurulumu tamamlayın.

Gerekli Plugin'leri Kurun:

Pipeline: CI/CD pipeline'ını oluşturmak için gerekli.

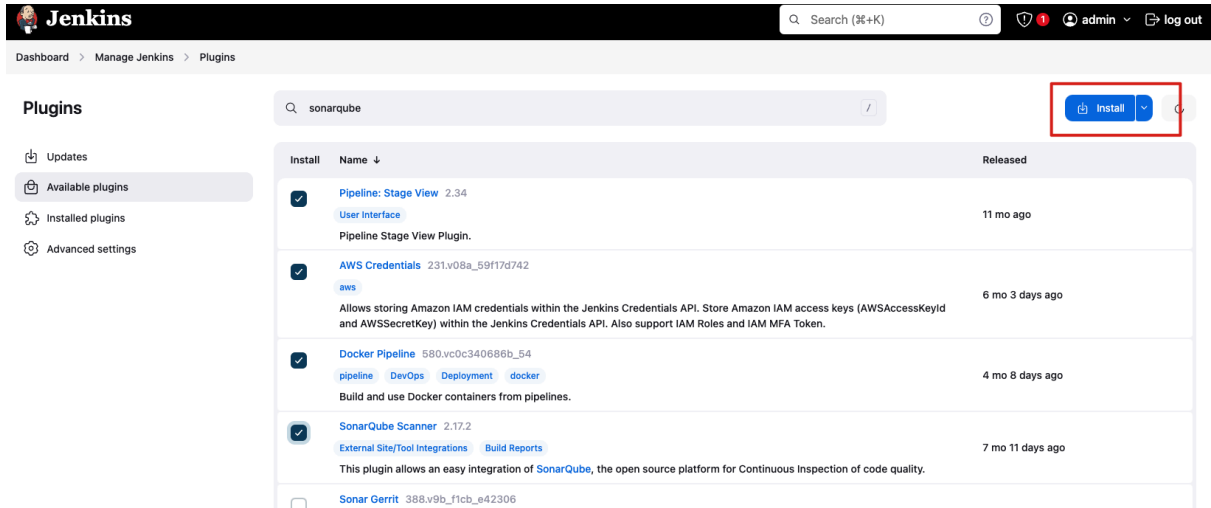
AWS Credentials: AWS'ye bağlanmak için gerekli kimlik bilgilerini yönetir.

Docker Pipeline: Docker işlemlerini Jenkins pipeline içinde kullanmak için gereklidir.

SonarQube: Kod kalitesi analizi için SonarQube'u Jenkins ile entegre eder.

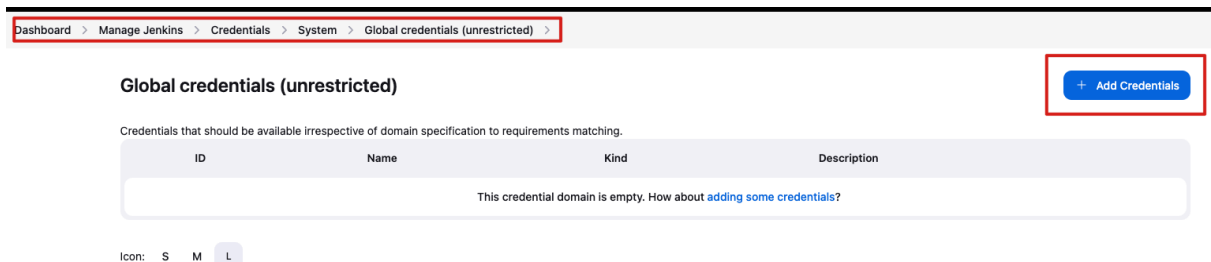
Jenkins pluginlerini kurmak için:

Manage Jenkins > Manage Plugins bölümünden available plugins seçerek ilgili plugini arayarak kurulum yapabilirsiniz.



Credentials Tanımlama:

Dashboard > Manage Jenkins > Credentials > System > Global credentials bölümünden **Add credentials** butonuna tıklayarak Jenkinsfile'da tanımlanan credential'ları ekleyebilirsiniz



AWS Credentials için:

- Kind kısmında **AWS Credentials** seçin.
- Access Key ve Secret Key bilgilerini girin. ID kısmına **aws-key** yazın

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

AWS Credentials

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

ID ?

aws-key

Description ?

Access Key ID ?

AKIAIZLDTJDXMYSVAFT

Secret Access Key

.....

! Please specify the Secret Access Key

IAM Role Support

Advanced ▾

Create

Docker Credentials için:

- Kind kısmında **Username with password** seçin.
- Username ve Password bilgilerini girdikten sonra ID kısmına **docker-cred** yazın

New credentials

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

hbayraktar

☐ Treat username as secret ?

Password ?

.....

ID ?

docker-cred

Description ?

Create

Docker Kurulumu:

Docker, Jenkins üzerinde çalışacak ve imajları oluşturup push etmek için gereklidir:

```
curl -s  
https://raw.githubusercontent.com/hakanbayraktar/ibb-tech/refs/heads/main/docker/ubuntu-24-docker-install.sh | sudo bash
```

```
sudo chmod 666 /var/run/docker.sock  
sudo usermod -aG docker jenkins  
sudo usermod -aG docker ubuntu
```

AWS CLI ve kubectl Kurulumu:

AWS CLI ile AWS kaynaklarını yönetebilir ve **kubectl** ile Kubernetes cluster'ınızı yönlendirebilirsiniz:

```
sudo apt install curl unzip -y  
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
```

```
"awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

kubectI Kurulumu:

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

Trivy Kurulumu:

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

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

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

sudo apt-get update

sudo apt-get install trivy -y
```

SonarQube Kurulumu:

SonarQube ile kod kalitesini analiz edeceğiz. SonarQube Jenkins ile aynı makinede çalışacaksa, sisteminizin en az 4 GB RAM'e sahip olduğundan emin olun.

SonarQube Docker ile Kurulumu:


```
docker run -d --name sonarqube \
-p 9000:9000 \
sonarqube
```


SonarQube Token Oluşturma:

SonarQube arayüzüne <http://server-IP:9000> adresinden erişin.

- Kullanıcı adı: admin
- Parola: admin

→ ↻ ⚠ Not Secure http://146.148.90.85:9000/sessions/new?return_to=%2F





Log in to SonarQube

Login *


Password *

[Go back](#) [Log in](#)

default şifreyi değiştirin

↻ ⚠ Not Secure http://146.148.90.85:9000/account/reset_password

Update your password

 This account should not use the default password.

Enter a new password

All fields marked with * are required

Old Password *

New Password *

Confirm Password *

[Update](#)

Kendi kullanıcı profilinize gidin ve **Security > Tokens** bölümüne tıklayın.

A Administrator

Profile **Security** Notifications Projects

Security

If you want to enforce security by not providing credentials of a real SonarQube user to run your code This will increase the security of your installation by not letting your analysis user's password going th

Generate Tokens

Name	Type	Expires in	
<input type="text" value="jenkins"/>	<input type="text" value="Global Analysis Token"/>	<input type="text" value="30 days"/>	<input type="button" value="Generate"/>

Yeni bir token oluşturun ve bu token'ı Jenkins pipeline'ında kullanın.

AWS IAM Kullanıcı ve Erişim Anahtarları Oluşturma:

Jenkins'in AWS kaynaklarına erişimi için bir IAM kullanıcısı ve ilgili erişim anahtarlarını oluşturmanız gerekmektedir:

IAM Kullanıcı Oluşturma:

- AWS Management Console'a giriş yapın.
- **IAM > Users > Add User** adımlarını takip edin.
- Kullanıcıya **programmatic access** verin ve gerekli politika izinlerini tanımlayın (örn. EKS, EC2, S3 gibi servisler için).
- Kullanıcının **Access Key ID** ve **Secret Access Key** bilgilerini kaydedin.

Jenkins Üzerinde AWS Credentials Tanımlama:

- **Manage Jenkins > Manage Credentials** bölümünden yeni bir AWS credential tanımlayın.
- AWS Access Key ID ve Secret Access Key bilgilerini buraya girin.

AWS EKS Cluster Oluşturma:

AWS CLI veya eksctl kullanarak bilgisayarınızdan veya jenkins serverden EKS cluster oluşturabilirsiniz. Burada eksctl kullanarak bir cluster oluşturma adımları verilmiştir:

aws configure ile access key secret key ve region bilgilerini girin

```
(base) ibb-tech-projects ➜ aws configure
AWS Access Key ID [*****UHW6]: AKIAI44QH8DHBEXAMPLE
AWS Secret Access Key [*****W10Z]: OLJVMQKQM7UWJZEXAMPLE40as9bXPk
Default region name [us-east-1]:
Default output format [None]:
```

```
eksctl create cluster \
  --name my-eks-cluster \
  --region us-east-1 \
  --nodegroup-name standard-workers \
  --node-type t3.medium \
  --nodes 2
```

```
2024-09-28 01:14:43 [i] waiting for CloudFormation stack "eksctl-my-eks-cluster-cluster"
2024-09-28 01:15:43 [i] waiting for CloudFormation stack "eksctl-my-eks-cluster-cluster"
2024-09-28 01:16:44 [i] waiting for CloudFormation stack "eksctl-my-eks-cluster-cluster"
2024-09-28 01:16:48 [!] recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on the cluster, eksctl
not configure the requested permissions; the recommended way to provide IAM permissions for "vpc-cni" addon is via pod identity
ociations; after addon creation is completed, add all recommended policies to the config file, under `addon.PodIdentityAssoci
`, and run `eksctl update addon`
2024-09-28 01:16:48 [i] creating addon
2024-09-28 01:16:48 [i] successfully created addon
2024-09-28 01:16:49 [i] creating addon
2024-09-28 01:16:49 [i] successfully created addon
2024-09-28 01:16:50 [i] creating addon
2024-09-28 01:16:50 [i] successfully created addon
2024-09-28 01:18:53 [i] building managed nodegroup stack "eksctl-my-eks-cluster-nodegroup-standard-workers"
2024-09-28 01:18:55 [i] deploying stack "eksctl-my-eks-cluster-nodegroup-standard-workers"
2024-09-28 01:18:55 [i] waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-standard-workers"
2024-09-28 01:19:26 [i] waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-standard-workers"
2024-09-28 01:20:06 [i] waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-standard-workers"
2024-09-28 01:20:39 [i] waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-standard-workers"
2024-09-28 01:21:24 [i] waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-standard-workers"
2024-09-28 01:21:24 [i] waiting for the control plane to become ready
2024-09-28 01:21:25 [✓] saved kubeconfig as "/Users/hakan/.kube/config"
2024-09-28 01:21:25 [i] no tasks
2024-09-28 01:21:25 [✓] all EKS cluster resources for "my-eks-cluster" have been created
2024-09-28 01:21:25 [✓] created 0 nodegroup(s) in cluster "my-eks-cluster"
2024-09-28 01:21:26 [i] nodegroup "standard-workers" has 2 node(s)
2024-09-28 01:21:26 [i] node "ip-192-168-20-26.ec2.internal" is ready
2024-09-28 01:21:26 [i] node "ip-192-168-53-83.ec2.internal" is ready
2024-09-28 01:21:26 [i] waiting for at least 2 node(s) to become ready in "standard-workers"
2024-09-28 01:21:26 [i] nodegroup "standard-workers" has 2 node(s)
2024-09-28 01:21:26 [i] node "ip-192-168-20-26.ec2.internal" is ready
2024-09-28 01:21:26 [i] node "ip-192-168-53-83.ec2.internal" is ready
2024-09-28 01:21:26 [✓] created 1 managed nodegroup(s) in cluster "my-eks-cluster"
2024-09-28 01:21:28 [i] kubectl command should work with "/Users/hakan/.kube/config", try 'kubectl get nodes'
2024-09-28 01:21:28 [✓] EKS cluster "my-eks-cluster" in "us-east-1" region is ready
```

Amazon Elastic
Kubernetes Service

Clusters

Amazon EKS Anywhere

Enterprise Subscriptions [New](#)

Related services

Amazon ECR

AWS Batch

Console settings

Documentation [↗](#)

Submit feedback

Extended support for Kubernetes versions pricing

New prices for extended support started in the April billing cycle. For more information, see the [blog post](#) [↗](#)

Notifications 🔔 0 🔔 0 🔔 0 🔔 4 🔔 0

EKS > Clusters

Clusters (1) [Info](#)

Filter clusters

< 1 >

Cluster name	Status	Kubernetes version	Support period	Upgrade policy	Created	Provider
my-eks-cluster	Active	1.30 Upgrade now	Standard support until July 28, 2025	Extended	22 minutes ago	EKS

Kubeconfig Güncelleme:

```
aws eks --region us-east-1 update-kubeconfig --name my-eks-cluster
```

IAM OIDC Sağlayıcı Oluşturma:

```
eksctl utils associate-iam-oidc-provider \
  --region us-east-1 \
  --cluster my-eks-cluster \
  --approve
```

Load Balancer Denetleyici İçin IAM Politikası Oluşturma:

```
curl -o iam-policy.json
https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.2.1/docs/install/iam_policy.json
```

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerIAMPolicy \
  --policy-document file://iam-policy.json
```

Döndürülen politika ARN'sini not edin.

```
(base) eks ➜ aws iam create-policy \
  --policy-name AWSLoadBalancerControllerIAMPolicy \
  --policy-document file://iam-policy.json
{
  "Policy": {
    "PolicyName": "AWSLoadBalancerControllerIAMPolicy",
    "PolicyId": "ANPA54T5LD3FIY25UFT43",
    "Arn": "arn:aws:iam:954819354314:policy/AWSLoadBalancerControllerIAMPolicy",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2024-09-13T06:45:46+00:00",
    "UpdateDate": "2024-09-13T06:45:46+00:00"
  }
}
```

```
eksctl create iamserviceaccount \
  --cluster=my-eks-cluster \
  --namespace=kube-system \
```

```
--name=aws-load-balancer-controller \
--attach-policy-arn=arn:aws:iam::637423295047:policy/AWSLoadBalancerControllerIAMPolicy \
--override-existing-serviceaccounts \
--approve
```

Pipeline Tanımı (Jenkinsfile):

Aşağıda Jenkins pipeline tanımı bulunmaktadır:

```
pipeline {
    agent any
    environment {
        cred = credentials('aws-key')
        dockerhub_cred = credentials('docker-cred')
        DOCKER_IMAGE = "hbayraktar/petclinic"
        DOCKER_TAG = "${BUILD_NUMBER}"
        SONARQUBE_URL = 'http://localhost:9000'
        SONAR_TOKEN = credentials('SONAR_TOKEN')
    }
    stages {
        stage("Git Checkout") {
            steps {
                git branch: 'main', changelog: false, poll: false, url:
                'https://github.com/hakanbayraktar/petclinic-java.git'
            }
        }
        stage('SonarQube Analysis') {
            steps {
                sh """
                    mvn sonar:sonar \
                    -Dsonar.projectKey=petclinic-java \
                    -Dsonar.host.url=${SONARQUBE_URL} \
                    -Dsonar.login=${SONAR_TOKEN}
                """
            }
        }
        stage("MVN Build") {
            steps {
                sh "mvn clean install -Dmaven.test.skip=true"
            }
        }
        stage("Docker Build & Push") {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker-cred', toolName:
                    'docker') {
                        sh "docker build -t ${DOCKER_IMAGE}:${DOCKER_TAG} ."
                        sh "docker push ${DOCKER_IMAGE}:${DOCKER_TAG}"
                    }
                }
            }
        }
    }
}
```

```

    stage("Update Kubernetes Manifest") {
        steps {
            sh "sed -i
's|hbayraktar/petclinic:latest|${DOCKER_IMAGE}:${DOCKER_TAG}|'
manifest/deployment.yaml"
        }
    }
    stage("TRIVY") {
        steps {
            sh "trivy image ${DOCKER_IMAGE}:${DOCKER_TAG}"
        }
    }
    stage("Deploy To EKS") {
        steps {
            sh 'aws eks update-kubeconfig --region us-east-1 --name
my-eks-cluster'
            sh 'kubectl apply -f manifest/deployment.yaml'
        }
    }
}
post {
    always {
        echo "Job is completed"
    }
    success {
        echo "Deployment successful!"
    }
    failure {
        echo "Job failed."
    }
}
}

```

New Item kısmından pipeline türünde petclinic adında bir job oluşturun

New Item

Enter an item name

petclinic

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and for organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.



Organization Folder

Creates a set of multibranch project subfolders by scanning for repositories.

OK

General bölümünden github project seçip github url adresimizi kopyalayalım

Dashboard > petclinic > Configuration

Configure

General

Advanced Project Options

Pipeline

General

Enabled

Description

Plain text [Preview](#)

☐ Discard old builds ?

☐ Do not allow concurrent builds

☐ Do not allow the pipeline to resume if the controller restarts

☒ GitHub project

Project url ?

<https://github.com/hakanbayraktar/petclinic-java.git>

Advanced

☐ Pipeline speed/durability override ?

☐ Preserve stashes from completed builds ?

Save Apply

Pipeline kısmında definition altında pipeline script from SCM SCM bölümünde Git seçtikten sonra Repository URL kısmına github url adresimizi kopyalayalım

Dashboard > petclinic > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

☐ Poll SCM ?

☐ Quiet period ?

☐ Trigger builds remotely (e.g., from scripts) ?

Advanced Project Options

Advanced

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/hakanbayraktar/petclinic-java.git

Credentials

- none -

Branch Specifier (blank for 'any') altındaki master yerine main yazdıktan sonra Jenkinsfile yazdıktan sonra save deyip job'ı oluşturalım

← → ↻ Not Secure http://146.148.90.85:8080/job/petclinic/configure

Dashboard > petclinic > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/main

Add Branch

Repository browser

(Auto)

Additional Behaviours

Add

Script Path


Jenkinsfile

☒ Lightweight checkout ?

Pipeline Syntax

Save Apply

petclinic job içerisindeyken build now diyerek petclinic job'ın çalışmasını sağlayabiliriz

**Jenkins**

Dashboard > petclinic >

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

GitHub

Stages

Rename

Pipeline Syntax

petclinic

Stage View

	Declarative: Checkout SCM	Declarative: Post Actions
Average stage times:	491ms	95ms
#1 Sep 28 03:09 No Changes	491ms	95ms

Permalinks

Build History trend

Filter...

#1

Sep 28, 2024, 12:09 AM

Atom feed for all

Atom feed for failures

build number #1 seçtikten sonra sol menüden console output seçerek job ile ilgili logları inceleyerek job ile ilgili bilgi alabilir varsa hataları logları okuyarak çözebiliriz.

Dashboard > petclinic > #1

Status

Changes

Console Output

Edit Build Information

Delete build '#1'

Timings

Git Build Data

Pipeline Overview

Pipeline Console

Restart from Stage

Replay

Pipeline Steps

Workspaces

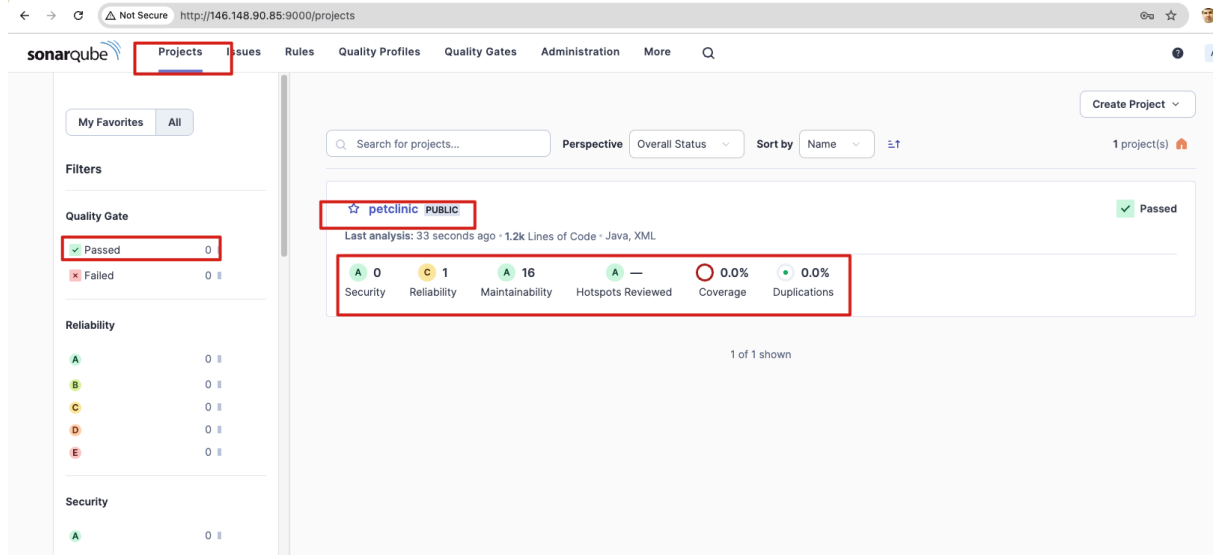
Console Output

Download

Cop

```
Started by user admin
Obtained Jenkinsfile from git https://github.com/hakanbayraktar/petclinic-java.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/petclinic
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/hakanbayraktar/petclinic-java.git
> git init /var/lib/jenkins/workspace/petclinic # timeout=10
> git init /var/lib/jenkins/workspace/petclinic # timeout=10
Fetching upstream changes from https://github.com/hakanbayraktar/petclinic-java.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/hakanbayraktar/petclinic-java.git +refs/heads/*:refs/remotes/origin/* #
> git config remote.origin.url https://github.com/hakanbayraktar/petclinic-java.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 21032591e88e86ba4a2c9f7361ec06eef42250e3 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 21032591e88e86ba4a2c9f7361ec06eef42250e3 # timeout=10
Commit message: "Update Jenkinsfile"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
```

işlem başarılı ile çalıştıktan sonra loglardan ayrıntılı bilgi alabilirsiniz. sonarqube analizi için sonarqube server dashboard unu inceleyebilirsiniz.



Sonuç:

Bu makalede, Jenkins'i kullanarak Java Petclinic uygulamasının AWS EKS üzerinde nasıl otomatik olarak dağıtılacağı adım adım anlatılmıştır. Bu süreçte AWS, Docker, Kubernetes ve SonarQube gibi araçlarla entegrasyon sağlanmış ve güvenlik açıkları, kod kalitesi analizleri gibi konular ele alınmıştır. Ayrıca, uygulamanın güvenliğini sağlamak için container image taramaları yapılmış, sürekli entegrasyon ve sürekli dağıtım (CI/CD) süreçleri detaylandırılmıştır.