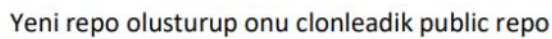
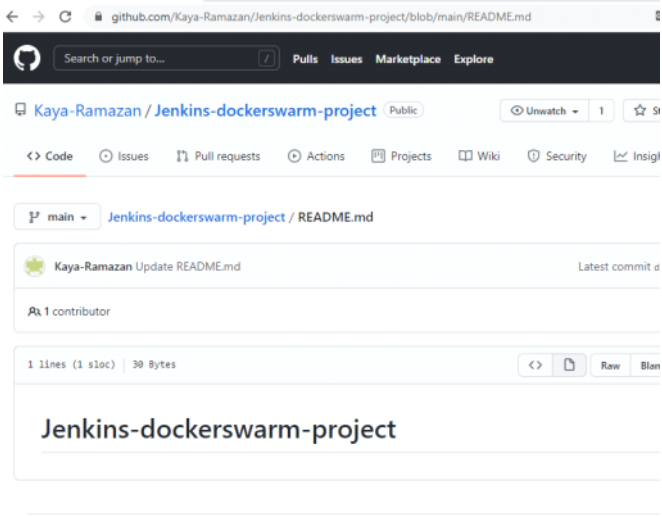


Saturday, 9 October 2021 0.27

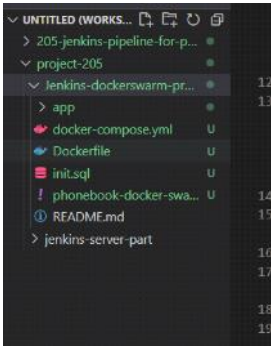
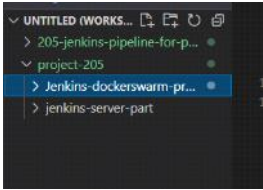
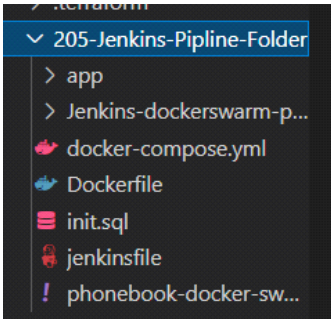


Jengins File in bunu yapabilmesi için icerisinde;

- ## Amacimiz Jenkins pipeline ile daha önce yaptigimiz islemi yapabilmek



Ve olusturdugumuzu dosyayi kendi publigimizde clonaladik



Bu sekilde bir dosya yapisi olusturduk kendi lokalimizde olusturdugumuz dosya da yani Jenkins-docker-Swarm -Project de;

App dosyamiz

Docker compose yaml dosyamiz

Docker file dosyamiz

Init.sql dosyamiz

Ve phone book docker swarm yml imiz oldu digerlerini git hub dosyamizda cok yer kaplamasin diye ust klasörler koyduk

Docker server part i ile de jenkinsi kolayca ayaga kaldiracagiz

Simdi Tf dosyamiza göz atalim

provider "aws" { # aws installla Terraform dedigimizde bu kısmi direk

```

alabiliyorduk
region = "us-east-1"
profile = "tyler"
access_key = ""
secret_key = "" #c bu sekilde aws e kolayca baglanabiliyoruz

```

From <<https://app.slack.com/client/T0227UVRJ8/D02DU06A00/thread/C021WSETWBB-1633805656.264700>>

```

}

data "aws_ami" "tf-ami" {
  # ami yi resource kullanarak cekiyoruz elle
  # de yazabilirdik

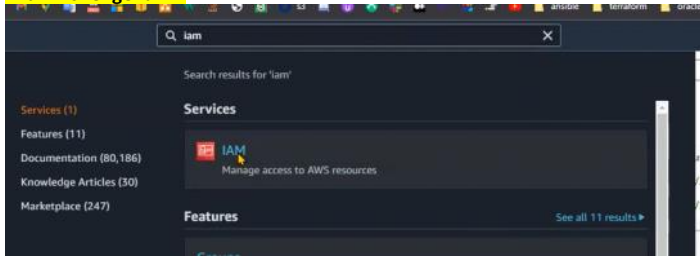
  owners      = ["amazon"]
  most_recent = true
  filter {
    name     = "name"
    values   = ["amzn2-ami-hvm*"] #filter kullanarak waws den ami cekecek
  }
}

data "aws_caller_identity" "current" {} # bir digetr data resource muz bu da
# bizim account numaramizi cekiyor

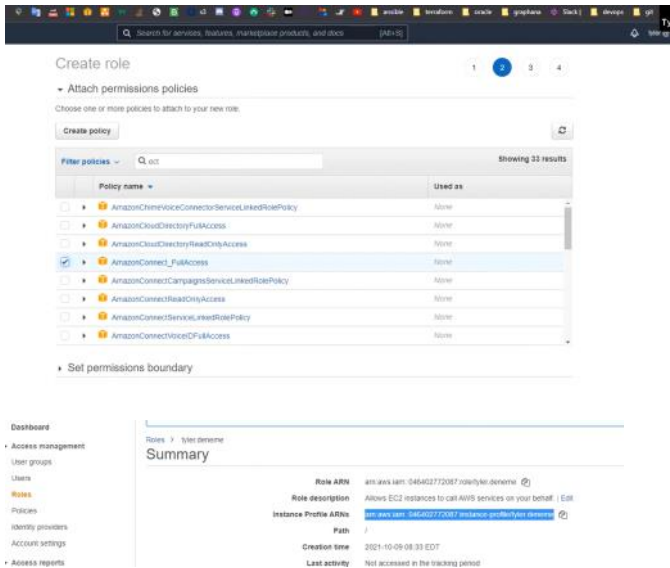
data "aws_region" "current" {
  name = "us-east-1"
}

```

I am role geldik



Rolde ec2-full-access sectik ve bir role olusturdugumuzu farz edersek



Role de kullandigimiz alan burasidir

```

resource "aws_iam_role" "aws_access" {
  # bu rdaki önemli konulardan biri rol
  # olusturmak roller bizim adimiza istedigimiz islemleri yaoacak bunun iki yolu var
  # biri iam role olusturup ec2 ya attach etmek digeride tf dosyasi ile bunu yapmak

  name = "awsrole"
  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = "sts:AssumeRole"
        Effect = "Allow"
        Sid    = ""
        Principal = {
          Service = "ec2.amazonaws.com"
        }
      }
    ]
  })
}

```

```

    })
    inline_policy {
        # normalde instancelarin msh komutu ile git kullanmadan
        name = "my_inline_policy" instancelaruin briur biryle irtibat kurmasini
        sagliyor
        policy = jsonencode({
            Version = "2012-10-17"
            Statement = [
                {
                    "Effect" : "Allow",
                    "Action" : "ec2-instance-connect:SendSSHPublicKey",
                    "Resource" : "arn:aws:ec2:${data.aws_region.current.name}:
${data.aws_caller_identity.current.account_id}:instance/*",
                    "Condition" : {
                        "StringEquals" : {
                            "ec2:osuser" : "ec2-user"
                        }
                    }
                },
                {
                    "Effect" : "Allow",
                    "Action" : "ec2:DescribeInstances",
                    "Resource" : "*"
                }
            ]
        })
    }
    managed_policy_arns =
    ["arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryFullAccess",
    "arn:aws:iam::aws:policy/AmazonEC2FullAccess",
    "arn:aws:iam::aws:policy/IAMFullAccess",
    "arn:aws:iam::aws:policy/AWSCloudFormationFullAccess"]
}

```

The screenshot shows the AWS IAM console interface. On the left, there's a navigation menu with options like Dashboard, Access management, User groups, Users, Roles, Policies, Identity providers, Account settings, Access reports, Access analyzer, Archive rules, Analyzers, Settings, Credential report, Organization activity, and Service control policies (SCPs). The main content area displays the details of a role named 'tyler-deneme'. The role's ARN is 'arn:aws:iam::046402772087:role/tyler-deneme'. The role description is 'Allows EC2 instances to call AWS services on your behalf'. The instance profile ARN is 'arn:aws:iam::046402772087:instance-profile/tyler-deneme'. The path is '/'. The creation time is '2021-10-09 08:33 EDT'. The last activity is 'Not accessed in the tracking period'. The maximum session duration is '1 hour'. The permissions tab is selected, showing 'Permissions policies (1 policy applied)'. The attached policy is 'AmazonConnect_FullAccess'.

Aws d bir role olusturdugumuz farz edersek instance profile ARN sini kopyalayip nmanagepolisy kismina yapistiriyoruz diyebiliriz devaminda da Ecr full access ac2 full access

Bunlari ec2 ya direk attach edemiyoruz bunu n icin debir role profile olusturup ona attach ediyoruz

resource "aws iam instance profile" "ec2-profile" { # bunlari ec2 lara attach etmek icin kullaniyoruz

```

    name = "jenkins-profile"
    role = aws_iam_role.aws_access.name
}
resource "aws_instance" "tf-jenkins-server" { # instance olusturuyoruz
    ami = data.aws_ami.tf-ami.id
    instance_type = "t2.micro"
    key_name = "tyler-team"
    // Write your pem file name
    security_groups = ["jenkins-server-sec-gr"]
    iam_instance_profile = aws_iam_instance_profile.ec2-profile.name # profile ile
    rolu bagliyoruz
    tags = {
        Name = "Jenkins_Server"
    }
    user_data = file("install-jenkins.sh") # user datayi sh dposyamizdan cekiyor
}
resource "aws_security_group" "tf-jenkins-sec-gr" {
    name = "jenkins-server-sec-gr"
    tags = {
        Name = "jenkins-server-sec-group"
    }
    ingress {

```

```

    from_port = 80
    protocol = "tcp"
    to_port = 80
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port = 22
    protocol = "tcp"
    to_port = 22
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port = 8080
    protocol = "tcp"
    to_port = 8080
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    from_port = 0
    protocol = "-1"
    to_port = 0
    cidr_blocks = ["0.0.0.0/0"]
  }
}
}

```

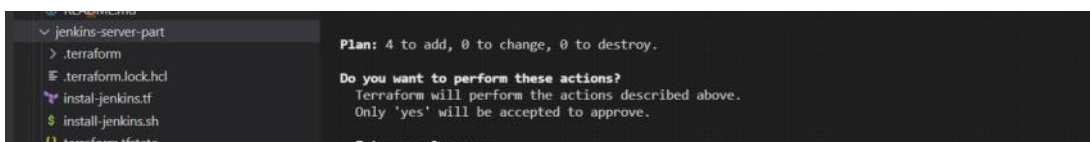
Simdi de `install.jenkins.sh` dosyamiza göz atalım;

```

#!/bin/bash
sudo yum update -y
hostnamectl set-hostname jenkins-server # burda isim verdik
yum install git -y # git i yukledik
sudo amazon-linux-extras install java-openjdk11 -y # jenkins yukleniyor
sudo wget -O /etc/yum.repos.d/jenkins.repo
https://pkg.jenkins.io/redhat/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
sudo amazon-linux-extras install epel -y
sudo yum install jenkins -y
sudo systemctl start jenkins # start enable -status yapıyoruz
sudo systemctl enable jenkins
sudo systemctl status jenkins
amazon-linux-extras install docker -y # docker yukleniyor
systemctl start docker
systemctl enable docker
usermod -a -G docker ec2-user # guruplari ekliyoruz
usermod -a -G docker jenkins
cp /lib/systemd/system/docker.service /lib/systemd/system/docker.service.bak #
docker in server daki bir konfigurASYONU
sed -i 's/^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
curl -L "https://github.com/docker/compose/releases/download/1.26.2/docker-
compose-$(uname -s)-$(uname -m)" \ # docker compose install ediliyor
-o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
# uninstall aws cli version 1
rm -rf /bin/aws # önce cli version 1 varsa siliyor version 2 yi install ediyoruz
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
./aws/install
yum install python3 -y
amazon-linux-extras install epel -y
yum install python-pip -y
pip install ec2instanceconnectcli
yum install amazon-ecr-credential-helper -y
mkdir -p /home/jenkins/.docker
cd /home/jenkins/.docker
echo '{"credsStore": "ecr-login"}' > config.json
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo
https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
sudo yum -y install terraform # terraform install eddiyoruz

```

Jenkins server part dosyamizin icerisn de terraform dosyamizi init ediyoruz



```
jenkins-server-part
├── .terraform
├── .terraform.lock.hcl
├── install-jenkins.tf
├── install-jenkins.sh
└── terraform.tfstate

Plan: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_iam_role.aws_access: Creating...
aws_security_group.tf-jenkins-sec-gr: Creating...
aws_iam_role.aws_access: Creation complete after 6s [id=awsrole]
aws_security_group.tf-jenkins-sec-gr: Creation complete after 6s [id=sg-055e36e7a5a4cf4ca]
aws_iam_instance_profile.ec2-profile: Creating...
aws_iam_instance_profile.ec2-profile: Creation complete after 1s [id=jenkins-profile]
aws_instance.tf-jenkins-server: Creating...
aws_instance.tf-jenkins-server: Still creating... [10s elapsed]
aws_instance.tf-jenkins-server: Still creating... [20s elapsed]
aws_instance.tf-jenkins-server: Still creating... [30s elapsed]
aws_instance.tf-jenkins-server: Still creating... [40s elapsed]
aws_instance.tf-jenkins-server: Still creating... [50s elapsed]
aws_instance.tf-jenkins-server: Creation complete after 58s [id=i-08305ba77bd0d557a]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

35846@LAPTOP-NRE8DIO6 MINGW64 ~/Desktop/Projets/project-205/jenkins-server-part
```

Ve daha sonra acilan ec2 muzun public ip sinin : 8080 ile browser da acip jenkins e baglaniyoruz

Ve
Setting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Password almak icin ec2 ya baglanip

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

Komutunu uyguluyoruz

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

[ec2-user@jenkins-server ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
0946e644a8094b68beee3216a1656064
[ec2-user@jenkins-server ~]$
```

Enter an item name

Project-pipeline

* Required field

Freestyle project
This is the central feature of Jenkins. It

Pipeline
Orchestrates long-running activities th

Multi-configuration project
Suitable for projects that need a large

Folder
Creates a container that stores nested
as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects accor

Organization Folder
Creates a set of multibranch project as

OK

Ve ilk pipline olusturuyoruz

Github url miizi kopyaliyoruz

Kaya-Ramazan / Jenkins-dockerswarm-project Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Kaya-Ramazan Update README.md

README.md Update README.md

README.md

Jenkins-dockerswarm-project

Go to file Add file Code

Clone

HTTPS SSH GitHub CLI

https://github.com/Kaya-Ramazan/Jenkins-di

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Abonelikler

No i

prov

Rele

No re

Creat

Jenkins de Terraform u kullanmak icin managementta dan Terraform un plugin inin yukluyoruz

Jenkins

Manage Jenkins

Building on this build will maintain the repository history. You should not copy-paste content from this build.

System Configuration

Configure System

Global Tool Configuration

Manage Plugins

Manage Nodes and Clouds

Security

Configure Global Security

Manage Credentials

Configure Credential Provider

Manage Users

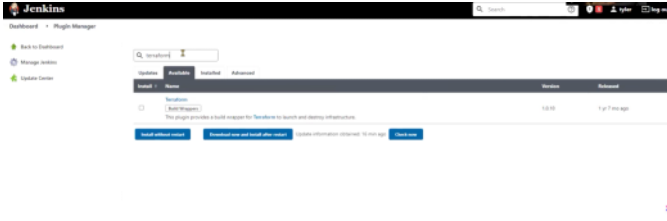
Status Information

System Information

System Log

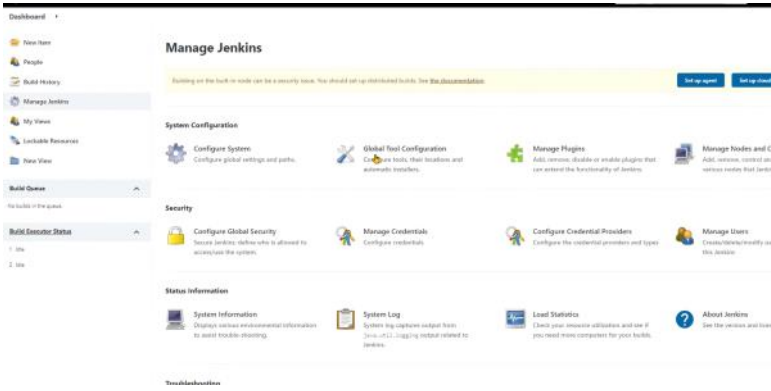
Local Feedback

About Jenkins



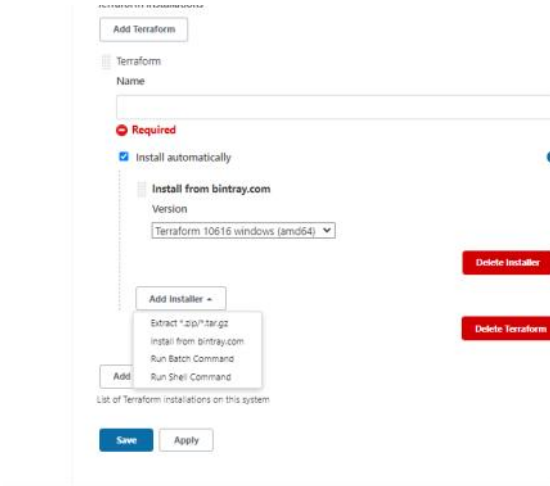
Bir sey adajha yapacagiz indirdigimiz Terraform n versiyonunu secmek
Bunun icin global tool managed,ent a godiyoruz ve eb altta terrafom u yaziyoruz

Vscode dan terraform version dedigimizde cikan cversionu buraya ekliyoruz



```
15846@LAPTOP-NREBDIO6 MINGW64 ~/Desktop/Projets/project-205/jenkins-server-part
$ terraform version
Terraform v1.0.6
on windows_amd64
+ provider registry.terraform.io/hashicorp/aws v3.62.0

Your version of Terraform is out of date! The latest version
is 1.0.8. You can update by downloading from https://www.terraform.io/downloads.html
```



Artik jenkins server uzerinden Terraform calistirabiliriz

Ve bir job olusturuyoruz

General Build Triggers Advanced Project Options Pipeline

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://ghp_5zkSIFyUL08Jk3Acu5s9GGN8d7NH039nYHf @github.com/Kaya-Ramazan/jenkins-dockerswarm-project.git`

☐ Pipeline speed/durability override

☐ Preserve stashes from completed builds

☐ This project is parameterized

Url adresimizi yazarken http// token adresimiz @ github adresimiz seklinde yapıyoruz

General Build Triggers Advanced Project Options Pipeline

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

`https://ghp_5zkSIFyUL08Jk3Acu5s9GGN8d7NH039nYHf @github.com/Kaya-Ramazan/jenkins-dockerswarm-project.git`

Credentials

- none - Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/main

Add Branch

Repository browser

(Auto)

Additional Behaviours

Add

Script Path

Jenkinsfile

Save Apply

Ve pipeline olustu

Lokalde olusturdugumuz dosyaya bu filelarimizi kopyalıyoruz diger dosyalari bu folderin disinda olusturacagiz Github a daha büyük boyutta dosya göndermemek için bu yöntemi uyguluyoruz

Jenkins part isimli dosyada da gereksinimlerimiz var onlari kullnacagiz Bir önceki derste terrafrm ile 5 node dan oluşan dcker swarm çalışan bir inf kurmustuk aynisi cfn ile ve jenkins ekleyerek yapıyoruz

Jenkins server ayaga kaldiracagiz Jenkins de baya bir seyler olmalı python docker vs Ecr a yetki vermemiz lazim Imaji gonderip imaj cekecegiz jenkins vasitasiyla Tf file sayesinde apply ile inf kuruldu

Ecr reposu create edildi imaj build edildi sonra docker swrm cluster ayaga kaldkti

Bunu tek bir dosya üzerinden yapmistim bunu burda pipeline ile farkli stagelere bolecegiz

Bu uygulamanin pipeline da nasıl gorundugunu gormek amacimiz

Project-205: Jenkins Pipeline for Dockerized Phonebook Application (Python Flask & MySQL) Deployed on Docker Swarm

Description

This project aims to create a Jenkins pipeline to deploy the Phonebook Application web application with Docker Swarm on Elastic Compute Cloud (EC2) Instances by pulling the app images from the AWS Elastic Container Registry (ECR) repository.

Problem Statement

![[Project_205](Project_205.png)]

- Your company has recently started a project that aims to serve as phonebook web application. Your teammates have started to work on the project and developed the UI and backend part of the project and they need your help to deploy the app in development environment.
- You are, as a cloud engineer, requested to create a Jenkins pipeline to deploy the Phonebook Application in the development environment on Docker Swarm on AWS EC2 Instances using AWS Cloudformation Service to showcase the project.
- To prepare the application for deployment, you need to;
 - Create a new public repository for the project on GitHub.
 - Create docker image using the `Dockerfile` from the base image of `python:alpine`.
 - Deploy the app on swarm using `docker compose`. To do so on the `Compose` file;
 - Create a MySQL database service with one replica using the image of `mysql:5.7`;
 - attach a named volume to persist the data of database server.
 - attach `init.sql` file to initialize the database using `configs`.
 - Configure the app service to;
 - pull the image of the app from the AWS ECR repository.
 - deploy the one app for each swarm nodes using `global` mode.
 - run the app on `port 80`.
 - Use a custom network for the services.
- You are also requested; to use AWS ECR as image repository, to create Docker Swarm with 3 manager and 2 worker node instances, to automate the process of Docker Swarm initialization through Cloudformation in the development environment. So, to prepare the infrastructure, you can configure Cloudformation template using the followings;
 - The application stack should be created with new AWS resources.
 - The application should run on Amazon Linux 2 EC2 Instance
 - EC2 Instance type can be configured as `t2.micro`.
 - To use the AWS ECR as image repository;
 - Enable the swarm node instances with IAM Role allowing them to work with ECR repos using the instance profile.
 - Install AWS CLI `Version 2` on swarm node instances to use `aws ecr` commands.
 - Use Amazon ECR Credential Helper to allow Docker to interact with ECR easily.
 - To automate the process of Docker Swarm initialization;
 - Install the docker and docker-compose on all nodes (instances) using the `user-data` bash script.
 - Create a separate instance with `instance profile` to be first manager node of the swarm. Within the `user-data` script;
 - Set the first manager node hostname as `Grand-Master`.
 - Initialize Docker swarm.
 - Create a docker service named `viz` on the manager node on port `8080` using the `dockersamples/visualizer` image, to monitor the swarm nodes easily.
 - Download `docker-compose.yml` file from the repo and deploy application stack on Docker Swarm.
 - Create a launch template with `instance profile` for Manager Nodes. Within the `user-data` script;
 - Install the python `ec2instanceconnectcli` package for `mssh` command.
 - Connect from manager node to the `Grand-Master` to get the `join-token` and join the swarm as manager node using `mssh` command.
 - Create two manager node instances using the `Manager Launch Template`.
 - Create a launch template with `instance profile` for Worker Nodes. Within the `user-data` script;
 - Install the python `ec2instanceconnectcli` package to use `mssh` command.
 - Connect from worker node to the `Grand-Master` to get the `join-token` and join the swarm as worker node using `mssh` command.
 - Create two worker node instance using the `Worker Launch Template`.
 - Create a single security group for all swarm nodes and open necessary ports for the app and swarm services.
 - Tag the swarm node instances appropriately as `Docker Manager/Worker`
 - `<Number> of <StackName>` to discern them from AWS Management Console.
 - The Web Application should be accessible via web browser from anywhere.
 - Phonebook App Website URL, Visualization App Website URL should be given as output by Cloudformation Service, after the stack created.
 - To create a Jenkins Pipelines, you need to launch a Jenkins Server with security group allowing SSH (port 22) and HTTP (ports 80, 8080) connections. For this purpose, you can use pre-configured `[*Cloudformation Template for Jenkins Server enabled with Git, Docker, Docker Compose and also configured to work with AWS ECR using IAM role*]`(`/clarusway-jenkins-with-git-docker-ecr-cfn.yml`).

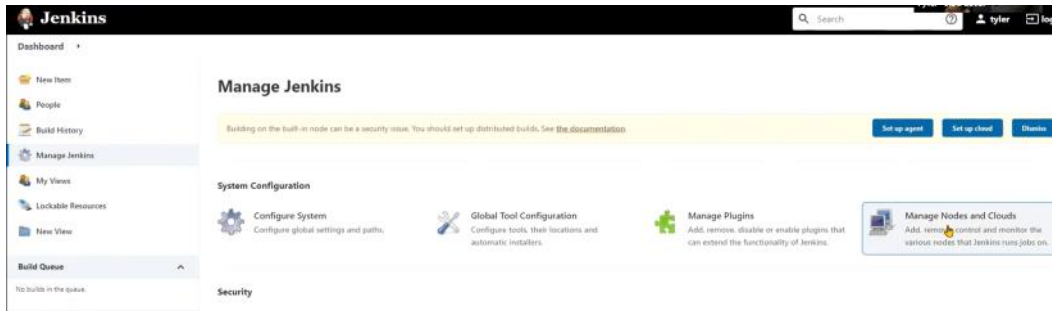
Istrelerimiz bunlardi

- Create a Jenkins Pipeline with following configuration;
- Create an image repository on ECR for the app. # bir ECR reposu olusturacak
- Build the application Docker image and push it to the ECR repository.
- Build the infrastructure for the app on EC2 instances using Cloudformation template.
- Deploy the application on Docker Swarm

Project Skeleton

```
```text
205-jenkins-pipeline-for-phonebook-app-on-docker-swarm (folder)
|
|----readme.md # Given to the students (Definition of the project)
|----phonebook-cfn-template.yml # To be delivered by students (Cloudformation
template)
|----Dockerfile # To be delivered by students
|----docker-compose.yml # To be delivered by students
|----init.sql # Given to the students (SQL statements to initialize db)
|----jenkins-cfn-template.yml # Given to the students (Cloudformation template for
Jenkins Server)
|----Jenkinsfile # To be delivered by students
|----app
| |----phonebook-app.py # Given to the students (Python Flask Web
Application)
| |----requirements.txt # Given to the students (List of Flask
Modules/Packages)
| |----templates
| |----index.html # Given to the students (HTML template)
| |----add-update.html # Given to the students (HTML template)
| |----delete.html # Given to the students (HTML template)
```
```

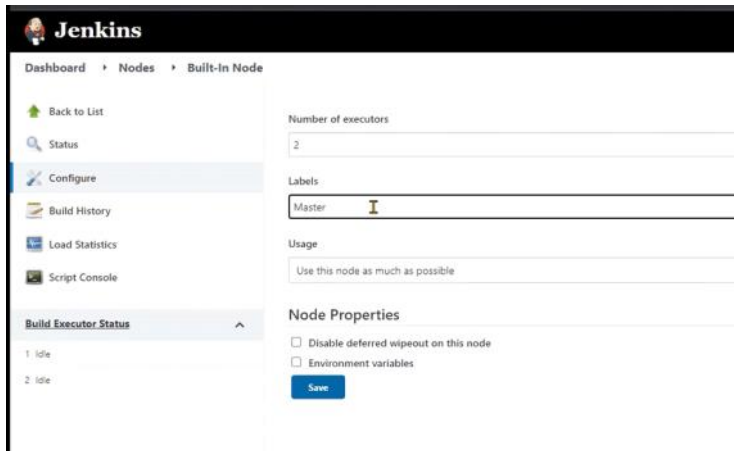
**** Simdi jenkins file olusturacagiz



Eger ki manage jenkins de manage nodes da label olarak master diye belirtirsek agent da master demmeiz gerekir ki byu su anlama gelir master nodes haricinde calisma

```
portfolio-jenkins > jenkins-dockerswarm-project > Jenkinsfile

1 Pipeline {
2   agent ("master")
3   environment {
4     PATH=sh(script:"echo $PATH:/usr/local/bin", return
5     AWS_REGION = "us-east-1"
```



```

pipeline {
    #
    agent any
    environment {
        PATH=sh(script:'echo $PATH:/usr/local/bin', returnStdout:true).trim() #
        AWS komutlarinin calismasi icin Jenkins in hangi ped e bakmasi gerektigini bilmesi
        gerekiyor
        AWS_REGION = "us-east-1"
        AWS_ACCOUNT_ID=sh(script:'export PATH="$PATH:/usr/local/bin" && aws sts
        get-caller-identity --query Account --output text', returnStdout:true).trim() #
        account id yi almaizi sagliyo
        ECR_REGISTRY="${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com"
        // ECR_REGISTRY = "046402772087.dkr.ecr.us-east-1.amazonaws.com"
        APP_REPO_NAME = "clarusway-repo/phonebook-app"
        APP_NAME = "phonebook"
        AWS_STACK_NAME = "Call-Phonebook-App-${BUILD_NUMBER}"
        CFN_TEMPLATE="phonebook-docker-swarm-cfn-template.yml"
        CFN_KEYPAIR="tyler-team"
        HOME_FOLDER = "/home/ec2-user"
        GIT_FOLDER = sh(script:'echo ${GIT_URL} | sed "s/.*\\///;s/.git$//"',
        returnStdout:true).trim()
    }
    stages {
        # stagerimizi bizden isterler göre yaziyoruz
        stage('Create ECR Repo') { # ECR repo create ediyoruz
            steps {
                echo 'Creating ECR Repo for App' # cli komutlari ile ecr i
                jenkins e create ettirecegiz
                sh """
                aws ecr create-repository \
                --repository-name ${APP_REPO_NAME} \
                --image-scanning-configuration scanOnPush=false \
                --image-tag-mutability MUTABLE \
                --region ${AWS_REGION}
                """
            }
        }
        stage('Build App Docker Image') {
            steps {
                echo 'Building App Image'
                sh 'docker build --force-rm -t "${ECR_REGISTRY}/
                $APP_REPO_NAME:latest" .'
                sh 'docker image ls'
            }
        }
        stage('Push Image to ECR Repo') { # ERC repoyu Push ediyoruz
            steps {
                echo 'Pushing App Image to ECR Repo'
                sh 'aws ecr get-login-password --region ${AWS_REGION} | docker
                login --username AWS --password-stdin "${ECR_REGISTRY}"'
                sh 'docker push "${ECR_REGISTRY}/${APP_REPO_NAME:latest}"'
            }
        }
        stage('Create Infrastructure for the App') { # Infrastructure create
        ediyoruz
            steps {
                echo 'Creating Infrastructure for the App on AWS Cloud'
                sh "aws cloudformation create-stack --region ${AWS_REGION} --
                stack-name ${AWS_STACK_NAME} --capabilities CAPABILITY_IAM --template-body file://
                ${CFN_TEMPLATE} --parameters ParameterKey=KeyName,ParameterValue=
                ${CFN_KEYPAIR}"
                script {
                    while(true) { # while döngüsü ile infstructure calisiyor mu
                    calismiyormu onu kontroledegiz
                        sleep(10)
                        echo "Docker Grand Master is not UP and running yet. Will
                        try to reach again after 10 seconds..."
                        ip = sh(script:'aws ec2 describe-instances --region
                        ${AWS_REGION} --filters Name=tag-value,Values=docker-grand-master Name=tag-
                        value,Values=${AWS_STACK_NAME} --query
                        Reservations[*].Instances[*].[PublicIpAddress] --output text | sed "s/\\s*None
                        \\s*/g"', returnStdout:true).trim() # aws insstancelarinin calistigi region a
                        göre bulup bunlardan filter ile suzup aradigimiz valudegeri grand master olanlari
                        bulup getiriyor
                        if (ip.length() >= 7) { # ip 7 karakterden fazladir ve
                        eger ki 7 karakterden fazla ise olustugu nu dusunebiliriz
                            echo "Docker Grand Master Public Ip Address Found:
                            $ip"
                            env.MASTER_INSTANCE_PUBLIC_IP = "$ip"
                            break
                        }
                    }
                }
            }
        }
        stage('Test the Infrastructure') { # Infstructure i test ediyoruz

```

```

    steps {
        echo "Testing if the Docker Swarm is ready or not, by checking Viz
App on Grand Master with Public Ip Address: ${MASTER_INSTANCE_PUBLIC_IP}:8080"
        script {
            while(true) { # curl komutu kullanarak master instance public
ipsine baglanmaya calisiypruz burda da amac docker swarm ayaga kalkti mi
vizulation calisiyor mu onu kontrol etmek
                try {
                    sh "curl -s --connect-timeout 60
${MASTER_INSTANCE_PUBLIC_IP}:8080" # 60 saniye icinde baglanamiyorsa 5 saniye
daha bekliyor ve
                    echo "Successfully connected to Viz App."
                    break
                }
                catch(Exception) {
                    echo 'Could not connect Viz App'
                    sleep(5)
                }
            }
        }
    }
}

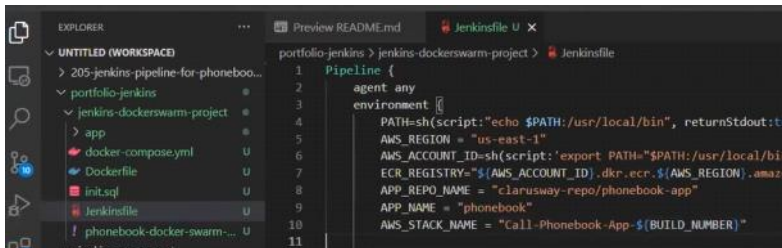
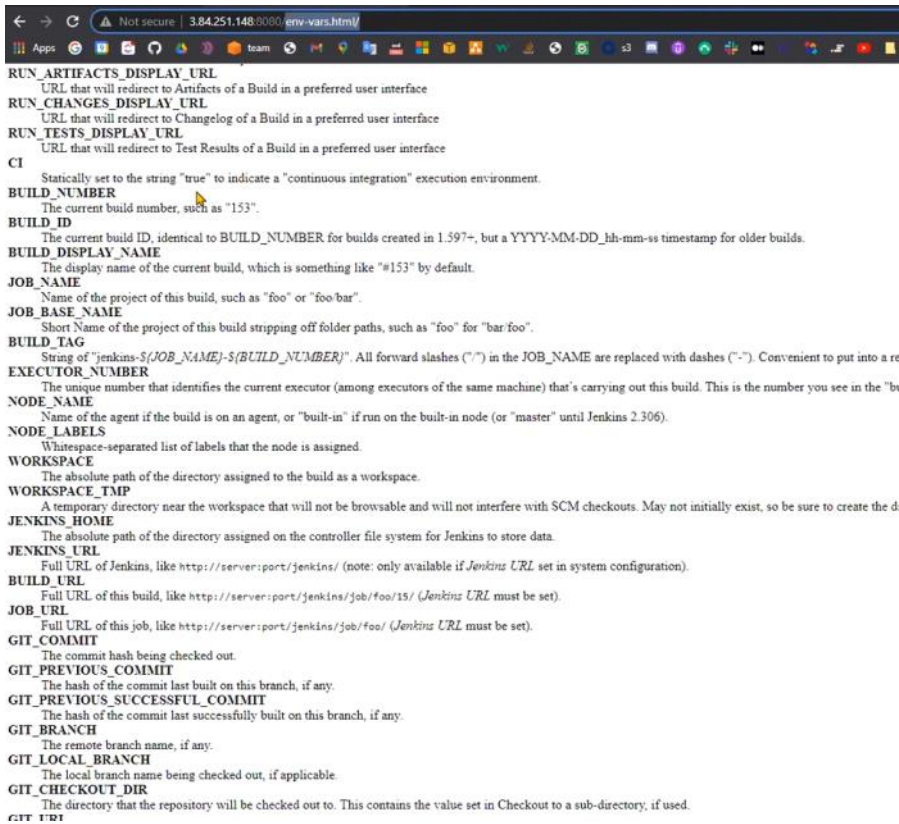
stage('Deploy App on Docker Swarm'){ # uygulama yi daha sonra docker
swarm uzerinde n deploy edcegiz
    environment {
        MASTER_INSTANCE_ID=sh(script:'aws ec2 describe-instances --region
${AWS_REGION} --filters Name=tag-value,Values=docker-grand-master Name=tag-
value,Values=${AWS_STACK_NAME} --query
Reservations[*].Instances[*].[InstanceId] --output text',
returnStdout:true).trim()
    }
    steps {

        echo "Cloning and Deploying App on Swarm using Grand Master with
Instance Id: $MASTER_INSTANCE_ID"
        sh 'mssh -o UserKnownHostsFile=/dev/null -o
StrictHostKeyChecking=no --region ${AWS_REGION} ${MASTER_INSTANCE_ID} git clone
${GIT_URL}' # master instance baglan ve git clone komutunu calistir anlamina
geliyor repository den dosyalariizi cekmeizi saglayacak
        sleep(10)
        sh 'mssh -o UserKnownHostsFile=/dev/null -o
StrictHostKeyChecking=no --region ${AWS_REGION} ${MASTER_INSTANCE_ID} docker stack
deploy --with-registry-auth -c ${HOME_FOLDER}/${GIT_FOLDER}/docker-compose.yml
${APP_NAME}'
    }
}

post {
    # ECR calissada calismasa da lokaldeki imajlari
silebiliyoruz bu kisim pipline in sionunda yapmak istedigimiz seyler icin
    always {
        echo 'Deleting all local images'
        sh 'docker image prune -af'
    }
    failure { # eger failure cikarsa ECR reposunu siliyoruz cunku silmez isek
bize ayni isim de ECR oldugunu soykleyecek
        echo 'Delete the Image Repository on ECR due to the Failure'
        sh """
        aws ecr delete-repository \
            --repository-name ${APP_REPO_NAME} \
            --region ${AWS_REGION} \
            --force
        """
        echo 'Deleting Cloudformation Stack due to the Failure' # CLI ile
resource lari tekrardan siliyoruz
        sh 'aws cloudformation delete-stack --region ${AWS_REGION} --
stack-name ${AWS_STACK_NAME}'
    }
}
}

```

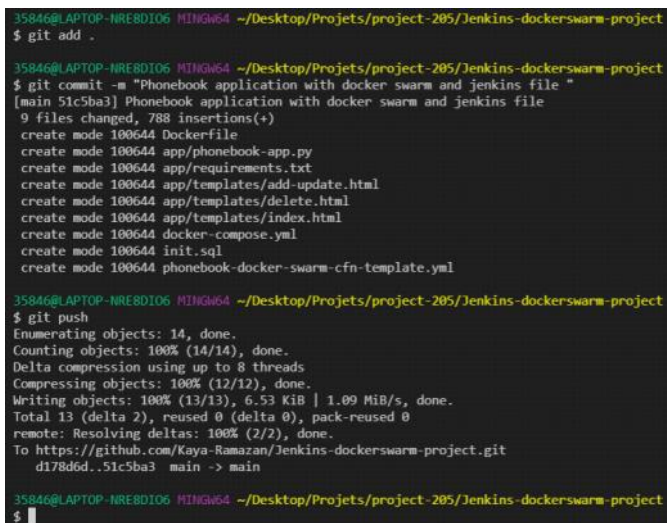
Ayrıca jenkins'e bağlandığımız konsola gidip : [public ip : 8080 / env-vars.html/](http://public_ip:8080/env-vars.html/) de
dığımızız



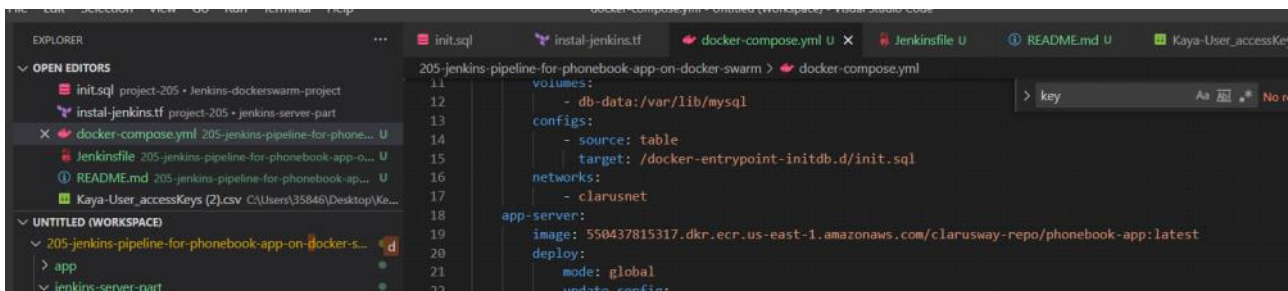
Eger ki

Jenkins file mizda hazir olduguna göre artik dosyalarimizi push eedbiliriz

Git hub a göndermek istedigimiz dosyolari docker swarm klasörü icerisinde bulunduruyorduk



Awx consolda ecr create ettigimizde bulunan image numberimiz ile docker compose yaml da bulunan image numaralarimizi ayni olmasi gerekiyor



Expected Outcome

![[Phonebook App Search Page](./search-snapshot.png)]

1-26

At the end of the project, following topics are to be covered;

- Jenkins Pipeline Configuration
- Docker Swarm Deployment
- Web App and MySQL Database Configuration in Docker Swarm
- Bash scripting
- AWS ECR as Image Repository
- AWS IAM Policy and Role Configuration
- AWS EC2 Launch Template Configuration
- AWS EC2 Configuration
- AWS EC2 Security Group Configuration
- AWS Cloudformation Service
- AWS Cloudformation Template Design
- Git & Github for Version Control System

At the end of the project, students will be able to;

- demonstrate how to configure Jenkins pipeline to deploy app on Docker Swarm together with Cloudformation Template.
- demonstrate how to configure Dockerfile and docker-compose files.
- set up a Docker Swarm cluster to work with AWS ECR using Cloudformation.
- deploy an application stack on Docker Swarm.
- create and configure AWS ECR from the AWS CLI.
- use Docker commands effectively to tag, push, and pull images to/from ECR.
- demonstrate bash scripting skills using 'user data' section in Cloudformation to install and setup environment for Docker Swarm on EC2 Instances.
- demonstrate their configuration skills of AWS EC2, Launch Templates, IAM Policy, Role, Instance Profile, and Security Group.
- configure Cloudformation template to use AWS Resources.
- show how to launch AWS Cloudformation Templates from AWS CLI.
- apply git commands (push, pull, commit, add etc.) and Github as Version Control System.

Resources

- [AWS Cloudformation User Guide](<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html>)
- [AWS CLI Command Reference](<https://docs.aws.amazon.com/cli/latest/index.html>)
- [AWS ECR Credential Helper](<https://github.com/aws/aws-ecr-credential-helper>)
- [Authenticating Amazon ECR Repositories for Docker CLI with Credential Helper](<https://aws.amazon.com/blogs/compute/authenticating-amazon-ecr-repositories-for-docker-cli-with-credential-helper/>)
- [Docker Compose File Reference](<https://docs.docker.com/compose/compose-file/>)
- [Docker Reference Page](<https://docs.docker.com/reference/>)
- [EC2 Instance Connect](<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Connect-using-EC2-Instance-Connect.html>)
- [Jenkins Handbook](<https://www.jenkins.io/doc/book/>)

Sonucta ECR repomuzada stack olustu

aws Services Search for services, features, marketplace products, and docs [Alt+S] Kaya-User @ 5504-378

Amazon Container Services

Amazon ECS
Clusters
Task definitions

Amazon ECR
Clusters

Amazon ECR
Repositories
Private registry
Public registry
Public gallery

ECR > Repositories

Private Public

Private repositories (2)

Find repositories

| | Repository name | URI | Created at | Tag immutability | Sc pu |
|-----------------------|------------------------------|---|---------------------------------|------------------|-------|
| <input type="radio"/> | clarusway-repo/phonebook-app | 550437815317.dkr.ecr.us-east-1.amazonaws.com/clarusway-repo/phonebook-app | 10 Ekim 2021, 22:04:35 (UTC+03) | Disabled | Dis |
| <input type="radio"/> | student1-team/todo-app | 550437815317.dkr.ecr.us-east-1.amazonaws.com/student1-team/todo-app | 09 Ekim 2021, 11:25:52 (UTC+03) | Disabled | Dis |

Cloudformation da stack de olusmus durumda burdaki output ile konsolda ciktimizi aliyoruz

aws Services Search for services, features, marketplace products, and docs [Alt+S] Kaya-User @ 5504-3781-5317 N. Virginia

CloudFormation

Stacks
Stack details
Drifts
StackSets
Exports

Designer

▼ Registry
Public extensions
Activated extensions
Publisher

Feedback

CloudFormation > Stacks > Call-Phonebook-App-5

Stacks (1)

Filter by stack name

Active View nested < 1 >

Call-Phonebook-App-5
2021-10-10 22:05:04 UTC+0300
CREATE_COMPLETE

Call-Phonebook-App-5

Delete Update Stack actions Create

Stack info Events Resources **Outputs** Parameters Template Change sets

Outputs (6)

Search outputs

| Key | Value | Description | Export name |
|---------------------------|--|-----------------------------|-------------|
| 1stDockerManagerDNSName | ec2-54-210-181-242.compute-1.amazonaws.com | Docker Manager 1st DNS Name | - |
| 1stDockerManagerDashboard | http://ec2-54-210-181-242.compute-1.amazonaws.com:8080 | Dashboard for Docker Swarm | - |
| 1stDockerWorkerDNSName | ec2-54-91-207-163.compute-1.amazonaws.com | Docker Worker 1st DNS Name | - |
| 2ndDockerManagerDNSName | ec2-18-233-171-158.compute-1.amazonaws.com | Docker Manager 2nd DNS Name | - |
| 2ndDockerWorkerDNSName | ec2-35-174-115-3.compute-1.amazonaws.com | Docker Worker 2nd DNS Name | - |
| 3rdDockerManagerDNSName | ec2-18-233-224-174.compute-1.amazonaws.com | Docker Manager 3rd DNS Name | - |

Ve son olarak Docker instanc larda olusan Docker Manager 1 in public ip ile : 8080 den baglandigimizda

aws Services ecr Kaya-User @ 5504-3781-5317 N. Virginia Support

New EC2 Experience Tell us what you think

EC2 Dashboard
EC2 Global View
Events
Tags
Limits

Instances
Instances **New**
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances **New**
Dedicated Hosts
Scheduled Instances
Capacity Reservations

Images
AMIs

Elastic Block Store

Network & Security
Security Groups
Elastic IPs
Placement Groups

Instances (1/6) Info

Filter instances

Instance state: running X Clear filters

| Name | Instance ID | Instance state | Instance type |
|---|---------------------------|----------------|------------------|
| Jenkins_Server | i-08305ba77bd0d557a | Running | t2.micro |
| Docker Worker 1st of Call-Phonebook-App-5 | i-0dbd2aa5cd94dd5 | Running | t2.micro |
| Docker Manager 3rd of Call-Phonebook-App-5 | i-0946dd570935fc568 | Running | t2.micro |
| Docker Worker 2nd of Call-Phonebook-App-5 | i-0b082e7970be497c3 | Running | t2.micro |
| Docker Manager 2nd of Call-Phonebook-App-5 | i-095609a4db983e4e6 | Running | t2.micro |
| Docker Manager 1st of Call-Phonebook-App-5 | i-07c5988367b90df2 | Running | t2.medium |

Instance: i-07c5988367b90df2 (Docker Manager 1st of Call-Phonebook-App-5)

Details Security **Networking** Storage Status checks Monitoring Tags

You can now check network connectivity with Reachability Analyzer. Run Reachability Analyzer X

Networking details info

Public IPv4 address
Public IPv4 address copied
ec2-54-210-181-242.compute-1.amazonaws.com | open address

Private IPv4 addresses
172.31.22.107
Private IPv4 DNS
ip-172-31-22-107.ec2.internal

VPC ID
vpc-05fd07f13680c6059 (default-vpc)

Subnet ID
subnet-011de771b8c36e167

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

← → Not secure | 54.210.181.242:8080

filter containers

grand-master manager 3.850G RAM x86_64/linux

ip-172-31-83-115.ec2.internal worker 0.960G RAM x86_64/linux

phonebook_app-server phonebook_database

ip-172-31-91-106.ec2.internal manager 0.960G RAM x86_64/linux

ip-172-31-93-131.ec2.internal worker 0.960G RAM x86_64/linux

phonebook_app-server

File olusturukun degistirmeyi unutmamiz gereken husular

```
1 version: "3.8"
2
3 services:
4   database:
5     image: mysql:5.7
6     environment:
7       MYSQL_ROOT_PASSWORD: P123456p
8       MYSQL_DATABASE: phonebook_db
9       MYSQL_USER: admin
10      MYSQL_PASSWORD: Clarusway_1
11     volumes:
12       - db-data:/var/lib/mysql
13     configs:
14       - source: table
15         target: /docker-entrypoint-initdb.d/init.sql
16     networks:
17       - clarusnet
18   app-server:
19     image: 046482772087.dkr.ecr.us-east-1.amazonaws.com/clarusway-repo/phonebook-app:latest
20     deploy:
21       mode: global
22       update_config:
23         parallelism: 2
24         delay: 5s
25         order: start-first
26     ports:
27       - "80:80"
28     networks:
29       - clarusnet
30   networks:
31     clarusnet:
32       driver: overlay
33
34 volumes:
35   db-data:
36
37 configs:
38   table:
39     file: ./init.sql
```

main clarusway-devops-8-21 / projects / 205-jenkins-pipeline-for-phonebook-app-on-docker-swarm / Jenkinsfile

TylerCounter added jenkins file to project-205 Latest commit 720cads 1 minute ago History

1 contributor

124 lines (109 sloc) 5.44 KB

```
1 pipeline {
2   agent any
3   environment {
4     PATH=sh(script:'echo $PATH:/usr/local/bin', returnStdout:true).trim()
5     AWS_REGION = "us-east-1"
6     AWS_ACCOUNT_ID=sh(script:'export PATH="/usr/local/bin" && aws sts get-caller-identity --query Account --output text', returnStdout:true).trim()
7     ECR_REGISTRY="${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com"
8     // ECR_REGISTRY = "046482772087.dkr.ecr.us-east-1.amazonaws.com"
9     APP_REPO_NAME = "clarusway-repo/phonebook-app"
10    APP_NAME = "phonebook"
11    AWS_STACK_NAME = "Call-Phonebook-App-${BUILD_NUMBER}"
12    CFN_TEMPLATE="phonebook-docker-swarm-cfn-template.yml"
13    CFN_KEYPAIR="Tyler-team"
14    HOME_FOLDER = "/home/ec2-user"
15    GIT_FOLDER = sh(script:'echo ${GIT_URL} | sed "s/.*\\/.*\\.git$/", returnStdout:true).trim()
16  }
17  stages {
18    stage('Create ECR Repo') {
19      steps {
20        echo 'Creating ECR Repo for App'
21        sh """
22        aws ecr create-repository \
23          --repository-name ${APP_REPO_NAME} \
```

