

Cem Anaral
+90 553 346 90 42
cemanaral@hotmail.com

IAM Read Only User Credentials

region: us-east-1 (N. Virginia)

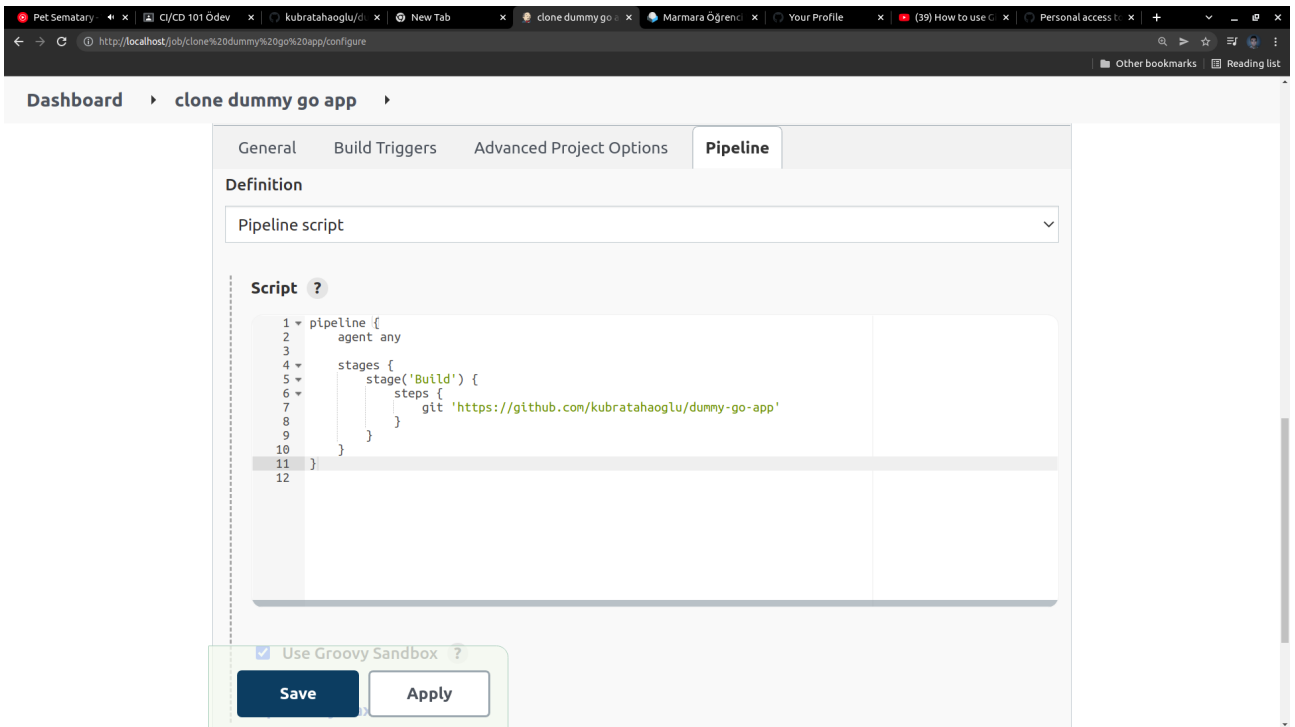
link: <https://476300129823.signin.aws.amazon.com/console>

username: aws-odev-user

password: {MQ{34VrZqSQzL-

Repo Klonlama

Aşağıdaki scripte sahip “clone dummy go app” isimli bir pipeline oluşturdum.



Çalışmadı. Aslında içinde gelen sample koda göre yazmıştım.

Sonrasında pipeline tarafındaki sağ alttaki “Pipeline Syntax” butonuna tıkladım. Pipeline scripti yaratan bir toolmuş. Aşağıdaki gibi pipeline scriptini yarattım.

The screenshot shows the 'Pipeline Syntax' tool interface. It has a sidebar with navigation links: Back, Snippet Generator, Declarative Directive Generator, Steps Reference, Global Variables Reference, Online Documentation, Examples Reference, and IntelliJ IDEA GDLS. The main area is titled 'Overview' and contains a 'Steps' section. Under 'Steps', there is a 'Sample Step' dropdown set to 'git: Git'. Below this, there are input fields for 'Repository URL' (https://github.com/kubratahaoglu/dummy-go-app), 'Branch' (main), and 'Credentials' (none). There are checkboxes for 'Include in polling?' and 'Include in changelog?'. A 'Generate Pipeline Script' button is at the bottom. Below the button, the generated script is shown: 'git branch: 'main', url: 'https://github.com/kubratahaoglu/dummy-go-app''.

The screenshot shows the Jenkins Pipeline configuration page. The 'Pipeline' tab is selected. The 'Script' section contains a Groovy script for a pipeline. The script is as follows:

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Clone git repo') {
6       steps {
7         git branch: 'main', url: 'https://github.com/kubratahaoglu/dummy-go-app'
8       }
9     }
10  }
11 }
12
```

At the bottom, there is a checkbox for 'Use Groovy Sandbox' which is checked. Below it are 'Save' and 'Apply' buttons.

Bu sefer hata vermedi.

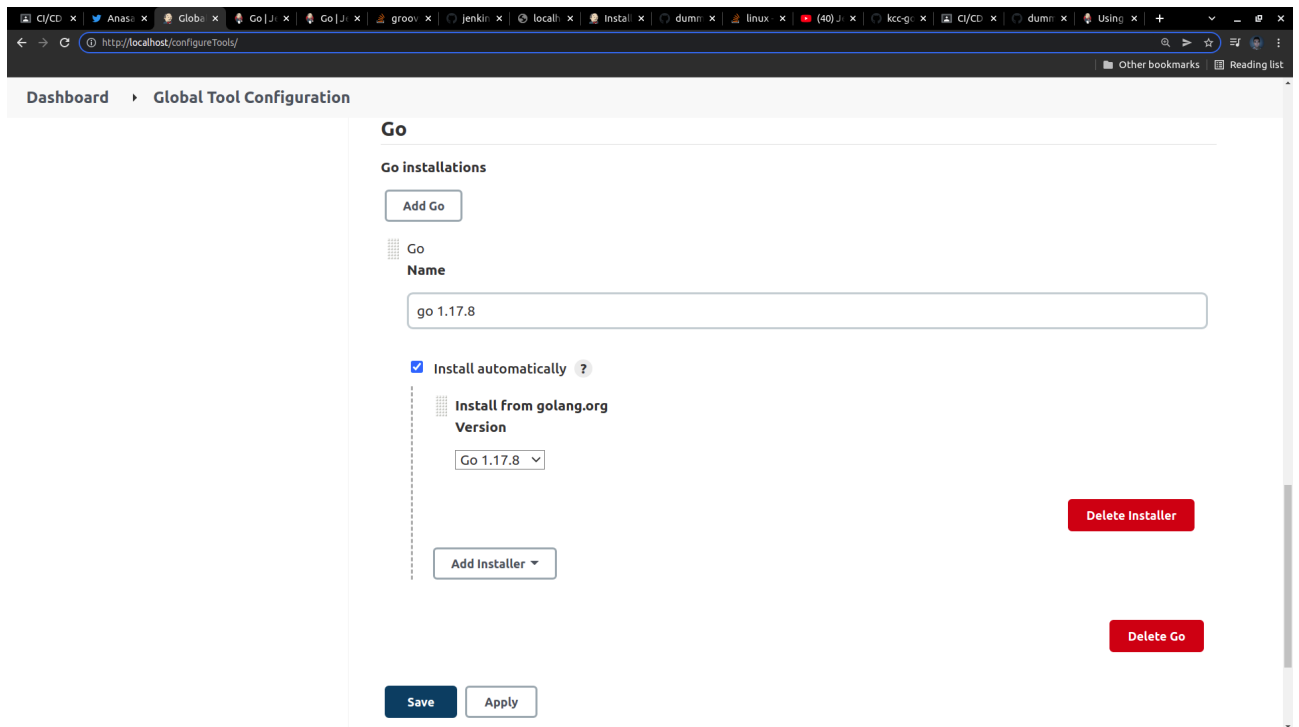
Gerçekten klonlanmış mı diye docker containerıma bağlandım (jenkinsi docker ile çalıştırıyorum).

“/var/jenkins_home/workspace/clone dummy go app” directorysine gittiğimde clonlandığını gördüm.

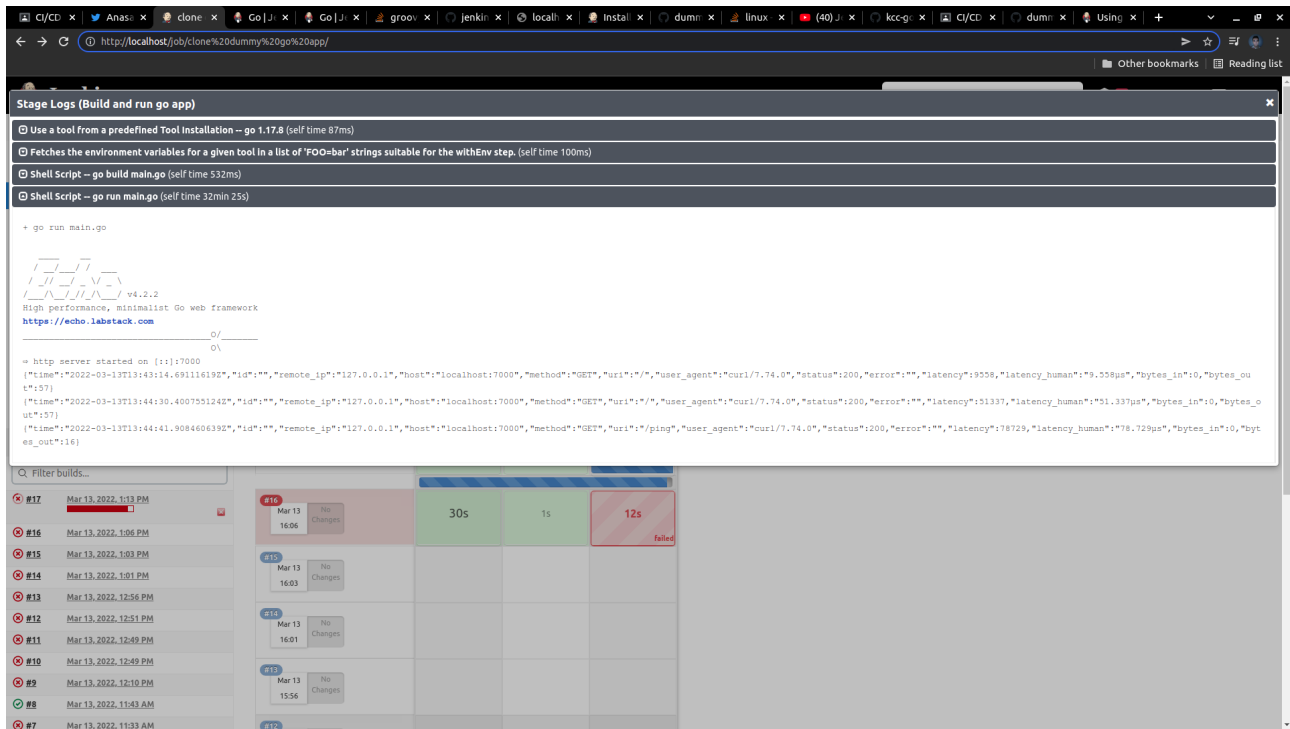
```
jenkins@5fbecbfd8426:~/workspace/clone dummy go app$ pwd
/var/jenkins_home/workspace/clone dummy go app
jenkins@5fbecbfd8426:~/workspace/clone dummy go app$ ls -la
total 56
drwxr-xr-x 3 jenkins jenkins 4096 Mar 13 11:43 .
drwxr-xr-x 4 jenkins jenkins 4096 Mar 13 11:29 ..
drwxr-xr-x 8 jenkins jenkins 4096 Mar 13 11:43 .git
-rw-r--r-- 1 jenkins jenkins  95 Mar 13 11:43 README.md
-rw-r--r-- 1 jenkins jenkins 1030 Mar 13 11:43 go.mod
-rw-r--r-- 1 jenkins jenkins 29478 Mar 13 11:43 go.sum
-rw-r--r-- 1 jenkins jenkins  591 Mar 13 11:43 main.go
jenkins@5fbecbfd8426:~/workspace/clone dummy go app$
```

Go build alma ve çalıştırma

İlk önce go pluginini yükledim ve tool olarak ayarlarını yaptım.



Docker containerımda jenkins 8080 portundan çalıştığı için hata aldım. Main.go isimli script HTTP_PORT environment variable ile port u çektiği için bir 7000 portunu HTTP_PORT environment variable'ı olarak ayarladım.



```

pipeline {
    agent any

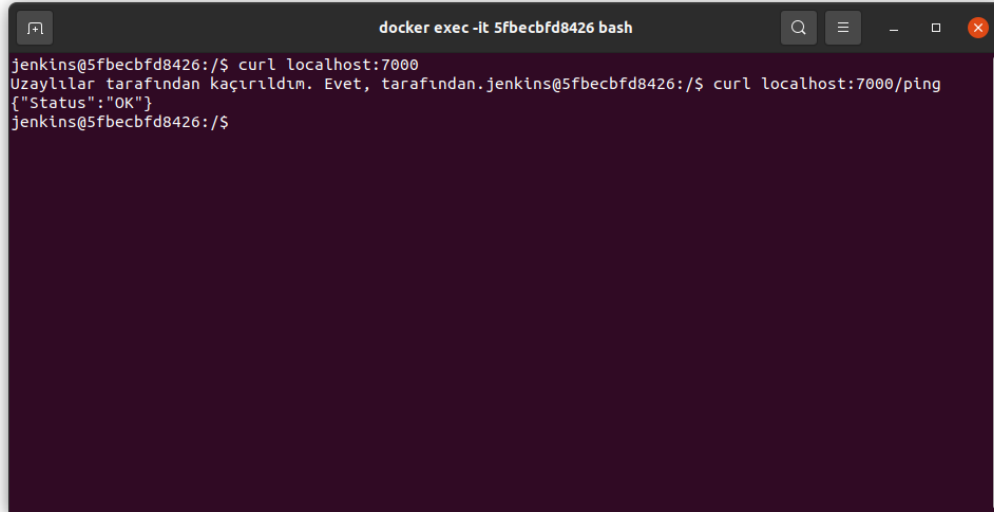
    tools {
        go 'go 1.17.8'
    }

    environment {
        GO111MODULE = 'on'
        HTTP_PORT = 7000
    }

    stages {
        stage('Clone git repo') {
            steps {
                git branch: 'main', url: 'https://github.com/kubratahaoglu/dummy-go-app'
            }
        }
        stage('Build and run go app') {
            steps {
                sh "go build main.go"
                sh "go run main.go"
            }
        }
    }
}

```

Jenkinsin çalıştığı containera bağlanıp 7000 portuna curl ile request attığımda çıktıya ulaştım (direkt ana bilgisayarımda çıktıyı görebilmek için 7000 portundan containera port binding yapmam gerekiyordu)

A terminal window titled "docker exec -it 5fbecbfd8426 bash" with a dark purple background. The terminal shows the following commands and output:

```
jenkins@5fbecbfd8426:/$ curl localhost:7000
Uzaylılar tarafından kaçırıldım. Evet, tarafından.jenkins@5fbecbfd8426:/$ curl localhost:7000/ping
{"Status":"OK"}
jenkins@5fbecbfd8426:/$
```

Codebuild ile repo çekme ve build etme

Codebuild projesine “cicd-odevi” adını verdim.

Github hesabımdan bir access token oluşturdum ve yeni açtığım codebuild projesine bu tokeni verdim.

Buildspec kısmında Insert build commands butonuna tıklayıp editoru açtım. Başlangıç için aşağıdaki scripti yazdım.

version: 0.2

phases:

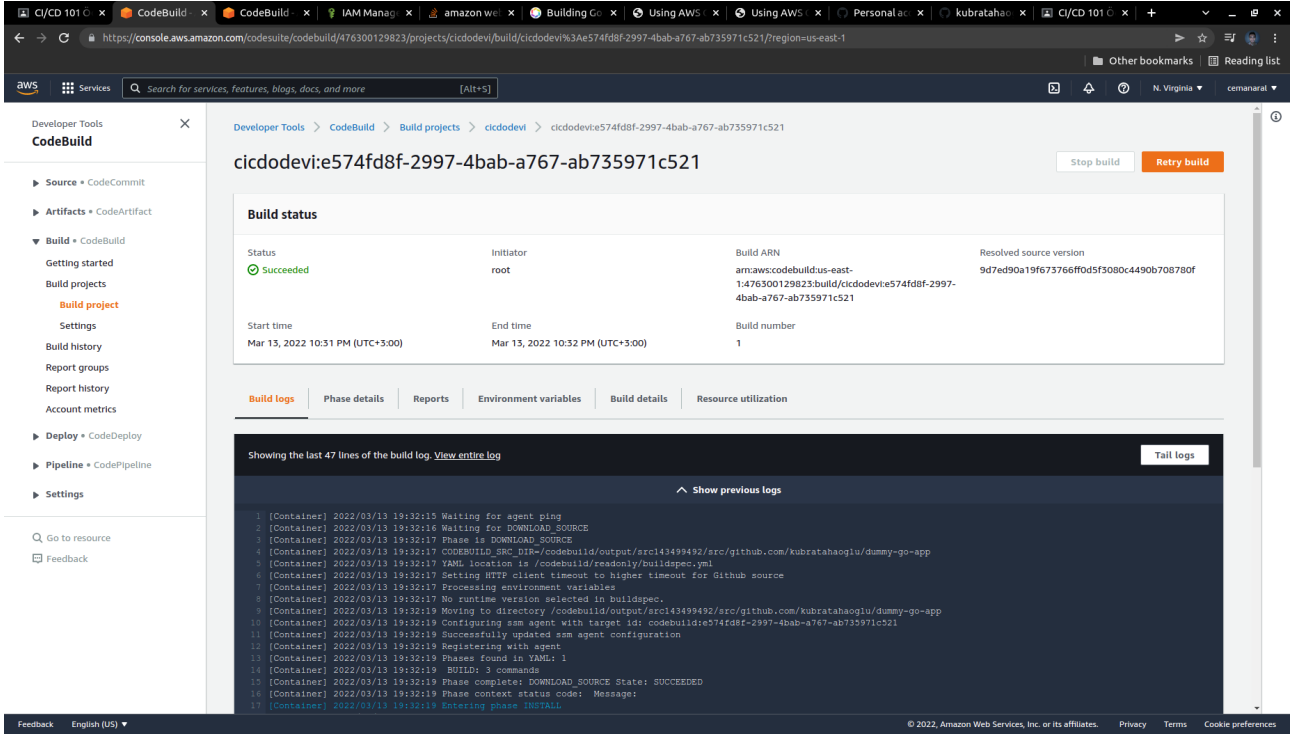
build:

commands:

- echo Build is starting...
- cd \$CODEBUILD_SRC_DIR
- go build main.go

Policy tarafında kendisi açtığı halde zaten bu isimli bir policy var diye hata veriyordu. Ben de kendi açtığı policynin arn numarasını bulup ilgili kutucuğa koydum. Projeyi oluşturabildim.

Build ettiğimde hata almadım.



ECR Pushlama

İlk olarak goapp adında public bir ECR repositorysi açtım (public.ecr.aws/h5r6z1s3/goapp).

Sanırım bir Dockerfile oluşturup onu build edip öyle ECR'a pushlamam gerekiyor. Sonuçta adı üstünde Elastic **Container** Registry.

Ancak github reposunda Dockerfile olmadığı için git reposunu forklayıp kendim Dockerfile'ı ekleyeceğim.

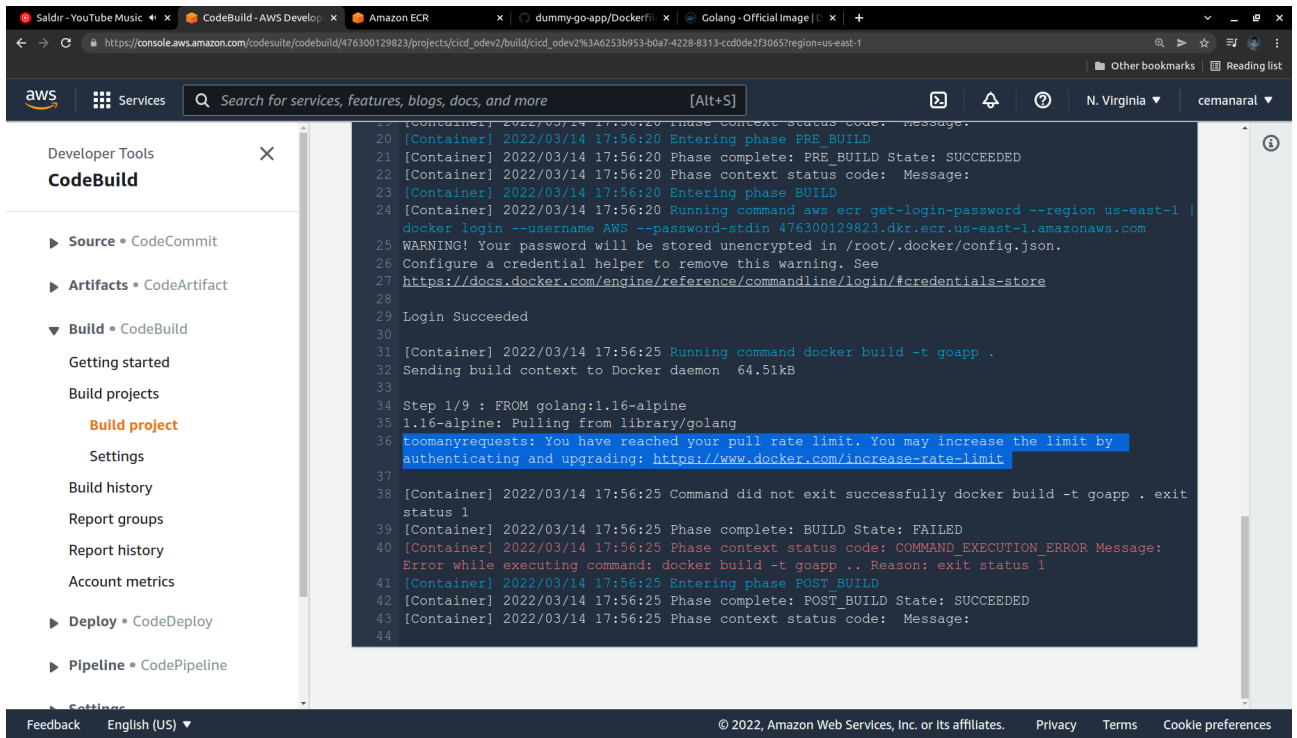
Galiba önceki aşamayı da o şekilde build etmem gerekiyordu. Yanlış yapmış olabilirim.

Linkteki gibi Dockerfile'ımı oluşturdum.:

<https://github.com/cemanaral/dummy-go-app/blob/main/Dockerfile>

Önceden oluşturduğum codebuild projesindeki source'u kendi forkumla değiştirdim.

CodeBuild'de docker imajının buildini alırken DockerHub'ın pull rate limitine takıldığı için golang base imajını ECR Public Gallery'den alacak şekilde değiştirdim.



The screenshot shows the AWS CodeBuild console interface. On the left, there's a sidebar with navigation options: Developer Tools, CodeBuild, Source (CodeCommit), Artifacts (CodeArtifact), Build (CodeBuild), and Pipeline (CodePipeline). The main area displays a build log for a project named 'dummy-go-app/docker'. The log shows the build process starting with 'Entering phase PRE_BUILD', followed by 'Phase complete: PRE_BUILD State: SUCCEEDED'. Then, it enters the 'BUILD' phase and runs the command 'aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 476300129823.dkr.ecr.us-east-1.amazonaws.com'. A warning message appears: 'WARNING! Your password will be stored unencrypted in /root/.docker/config.json. Configure a credential helper to remove this warning. See https://docs.docker.com/engine/reference/commandline/login/#credentials-store'. The login is successful, and then the build runs 'docker build -t goapp .'. The log shows 'Sending build context to Docker daemon 64.51kB' and 'Step 1/9 : FROM golang:1.16-alpine'. The error occurs at line 36: '1.16-alpine: Pulling from library/golang' with the message 'toomanyrequests: You have reached your pull rate limit. You may increase the limit by authenticating and upgrading: https://www.docker.com/increase-rate-limit'. The build then enters the 'POST_BUILD' phase and completes with 'Phase complete: POST_BUILD State: SUCCEEDED'.

```
[Container] 2022/03/14 17:56:20 Phase context status code: Message:
20 [Container] 2022/03/14 17:56:20 Entering phase PRE_BUILD
21 [Container] 2022/03/14 17:56:20 Phase complete: PRE_BUILD State: SUCCEEDED
22 [Container] 2022/03/14 17:56:20 Phase context status code: Message:
23 [Container] 2022/03/14 17:56:20 Entering phase BUILD
24 [Container] 2022/03/14 17:56:20 Running command aws ecr get-login-password --region us-east-1 |
docker login --username AWS --password-stdin 476300129823.dkr.ecr.us-east-1.amazonaws.com
25 WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
26 Configure a credential helper to remove this warning. See
27 https://docs.docker.com/engine/reference/commandline/login/#credentials-store
28
29 Login Succeeded
30
31 [Container] 2022/03/14 17:56:25 Running command docker build -t goapp .
32 Sending build context to Docker daemon 64.51kB
33
34 Step 1/9 : FROM golang:1.16-alpine
35 1.16-alpine: Pulling from library/golang
36 toomanyrequests: You have reached your pull rate limit. You may increase the limit by
authenticating and upgrading: https://www.docker.com/increase-rate-limit
37
38 [Container] 2022/03/14 17:56:25 Command did not exit successfully docker build -t goapp . exit
status 1
39 [Container] 2022/03/14 17:56:25 Phase complete: BUILD State: FAILED
40 [Container] 2022/03/14 17:56:25 Phase context status code: COMMAND_EXECUTION_ERROR Message:
Error while executing command: docker build -t goapp .. Reason: exit status 1
41 [Container] 2022/03/14 17:56:25 Entering phase POST_BUILD
42 [Container] 2022/03/14 17:56:25 Phase complete: POST_BUILD State: SUCCEEDED
43 [Container] 2022/03/14 17:56:25 Phase context status code: Message:
44
```

Public ECR’da pushlarken authorization sorunu alıyordum. “goapp” adında yeni bir private repository açtım. Aşağıdaki Buildspec ile bir problem yaşamadan ECR’a pushladım.

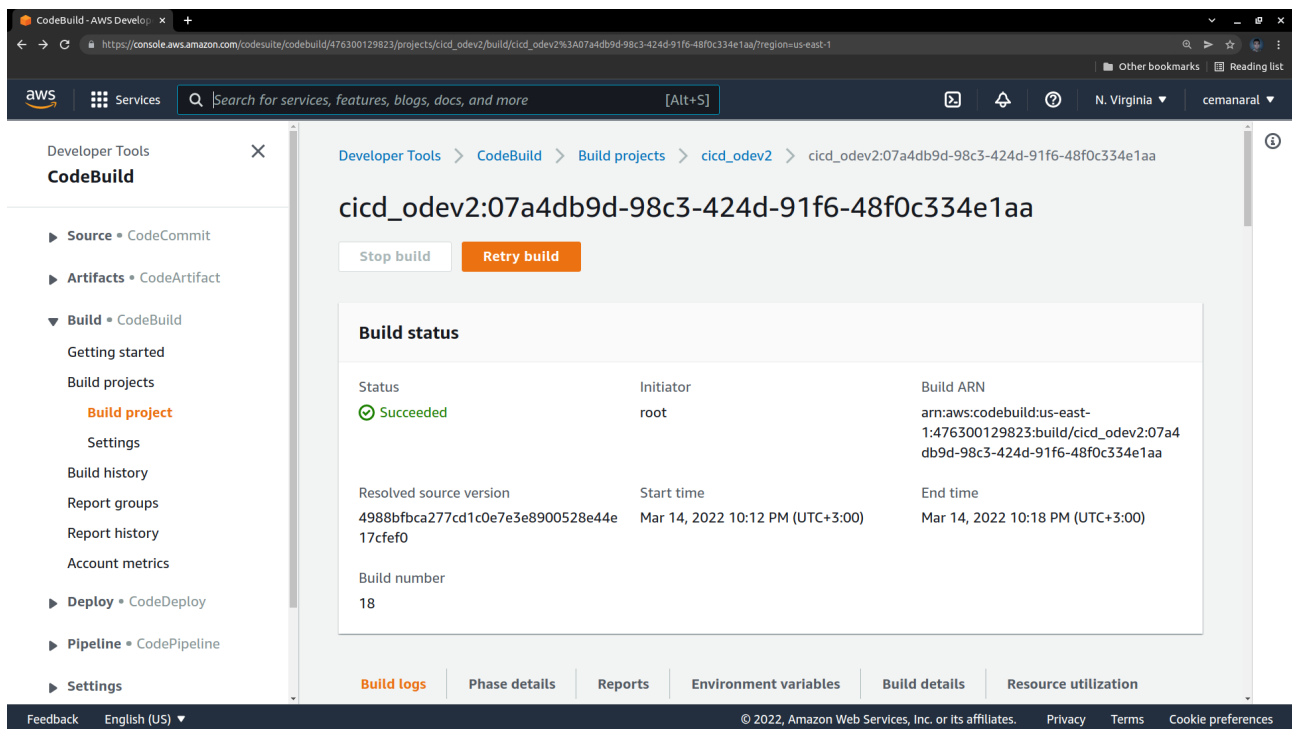
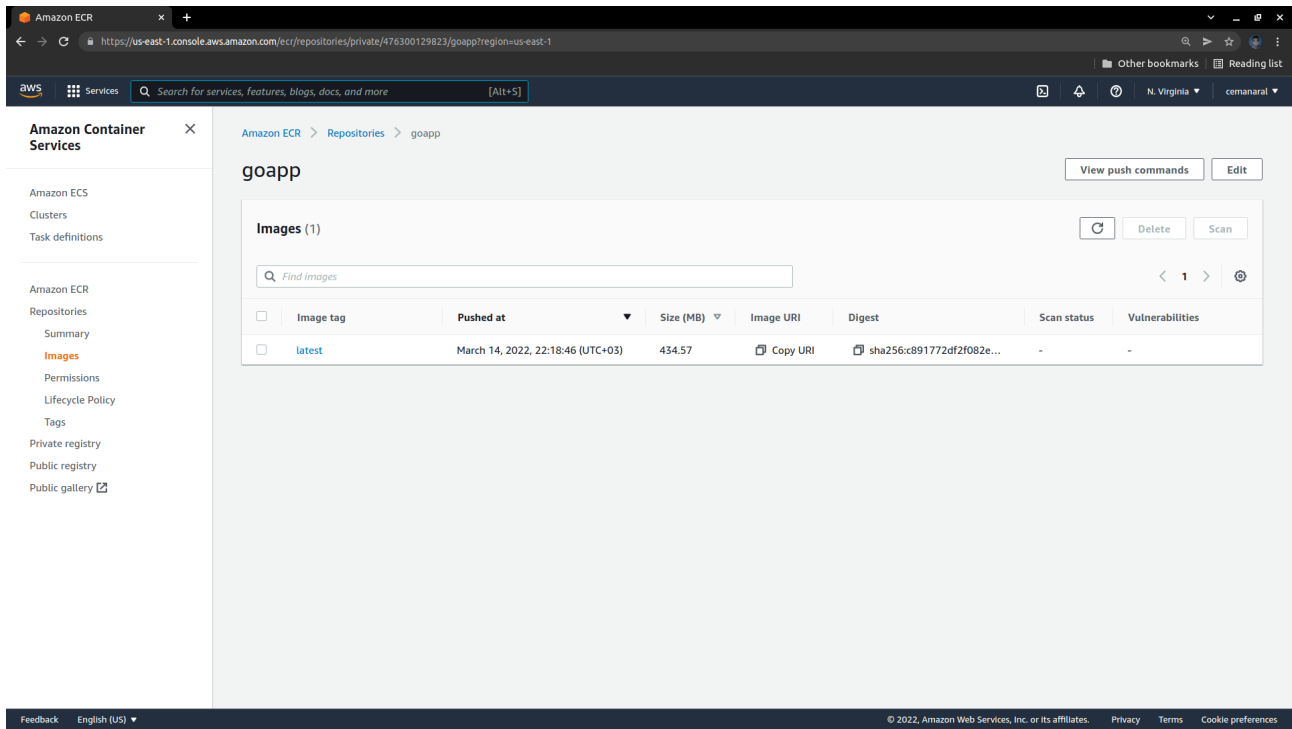
version: 0.2

phases:

build:

commands:

- aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 476300129823.dkr.ecr.us-east-1.amazonaws.com
- docker build -t goapp .
- docker tag goapp:latest 476300129823.dkr.ecr.us-east-1.amazonaws.com/goapp:latest
- docker push 476300129823.dkr.ecr.us-east-1.amazonaws.com/goapp:latest



Düzeltilme – Jenkins’de Docker build alma

Not: Dockerfile’ı git reposunu forklayarak kendim ekledim.

Pipeline:

The screenshot shows the Jenkins Pipeline configuration page for a job named 'go-docker-build'. The 'Pipeline' tab is selected, and the 'Definition' section is visible. The 'Pipeline script' dropdown is set to 'Script'. The script is a Groovy pipeline with the following stages:

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Clone git repo') {
6       steps {
7         git branch: 'main', url: 'https://github.com/cemnaral/dummy-go-app/'
8       }
9     }
10    stage('Build with docker') {
11      steps {
12        sh "docker build -t goapp:latest ."
13      }
14    }
15    stage('Run with docker') {
16      steps {
17        sh "docker run -d -p80:8080 goapp:latest"
18      }
19    }
20  }
21 }
```

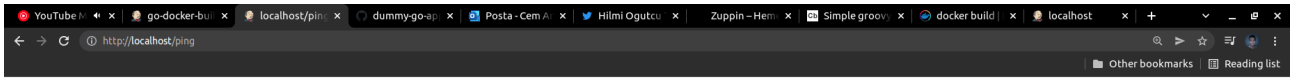
Below the script, the 'Use Groovy Sandbox' checkbox is checked. At the bottom, there are 'Save' and 'Apply' buttons. The bottom right corner of the page shows 'REST API' and 'Jenkins 2.332.1'.

The screenshot shows the Jenkins console output for the 'go-docker-build' job. The output displays the execution of the pipeline stages:

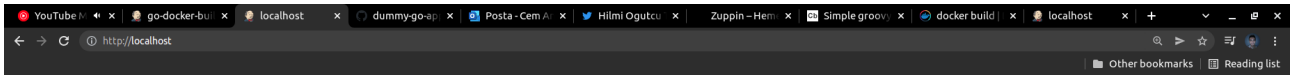
```
[Pipeline] { (Build with docker)
[Pipeline] sh
+ docker build -t goapp:latest .
Sending build context to Docker daemon 119.8kB

Step 1/9 : FROM public.ecr.aws/docker/library/golang:latest
--> e76246e054d0
Step 2/9 : WORKDIR /app
--> Using cache
--> 222c9487fb4d
Step 3/9 : COPY go.mod ./
--> Using cache
--> 5a094fc3837c
Step 4/9 : COPY go.sum ./
--> Using cache
--> 6987f6cf870b
Step 5/9 : RUN go mod download
--> Using cache
--> 23918e1bc97e
Step 6/9 : COPY *.go ./
--> Using cache
--> 051d429f54ff
Step 7/9 : RUN go build
--> Using cache
--> 968925f8be49
Step 8/9 : EXPOSE 8080
--> Using cache
--> b07557bebe67
Step 9/9 : CMD ["go", "run", "main.go"]
--> Using cache
--> f2010c9f1f78
Successfully built f2010c9f1f78
Successfully tagged goapp:latest
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] sh
+ docker run -d -p80:8080 goapp:latest
25772ab0ec0671550b318a505781e850da550b3eb99a904cae0adfd0af190ea7
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

The bottom right corner of the page shows 'REST API' and 'Jenkins 2.332.1'.



```
{"Status": "OK"}
```



Uzaylılar tarafından kaçırıldım. Evet, tarafından.