Kubernetes with CI/CD and Helm

Project introduction **2**
AWS services used in this project 2
Set up EKS and create a Kubernetes cluster in Cloud9 **2**
Part 1: set up your Cloud9 environment 2
Part 2: install Kubernetes tools in Cloud9 4
Part 3: set up your EKS cluster in Cloud9 7
Part 3.1: set up the environment to launch your EKS cluster 7
Part 3.2: create a deployment file 8
Part 3.3: create your EKS cluster 8
Part 3.4: test your cluster 9
Set up CodePipeline **10**
Part 1: create an IAM role and generate a GitHub personal Token 10
Part 2: set up CodePipeline 12
Part 2.1: cleanup 21
Introduction to Helm **22**
What is Helm? 22
Helm Charts 22
Section overview 22
Set up and use Helm 22
Prerequisites 22
Part 1: introduction 23
Part 1.1: install the Helm Command Line Interface (CLI) 23
Part 1.2: add your first Helm Chart repository 23
Part 2: deploy an Apache web server with Helm Charts 24
Part 2.1: update Charts list in Helm 24
Part 2.2: browse and find Helm Chart repositories 24
Part 2.3: install the Apache server Chart to your EKS cluster using Helm 25
Part 2.4: cleanup 26
Part 3: deploy a static web application with Helm Charts 27
Part 3.1: find a Git repository 27
Part 3.2: deploy from a Git repository using Bitnami’s Apache Helm Chart 27
Part 3.3: cleanup 29
Clean up project resources **29**
References **30**
Project introduction 30

Set up EKS and create a Kubernetes cluster in Cloud9 30
Set up CodePipeline 30
Introduction to Helm 30

Project introduction

The objective of this project is to help you get started with using Kubernetes. Kubernetes is a platform that manages and automates containerized services and applications. In Kubernetes, a cluster is a group of worker machines, also known as nodes, that are used for running containerized applications.

In this guide, there will be walkthroughs for how to set up Elastic Kubernetes Service (EKS) in a Cloud9 environment and how to create a Kubernetes cluster. There will also be a walkthrough on how to use AWS CodePipeline with Kubernetes. The guide will introduce you to how to use Helm, a package manager for Kubernetes, to deploy a standalone web server and a static web application as well.

AWS services used in this project

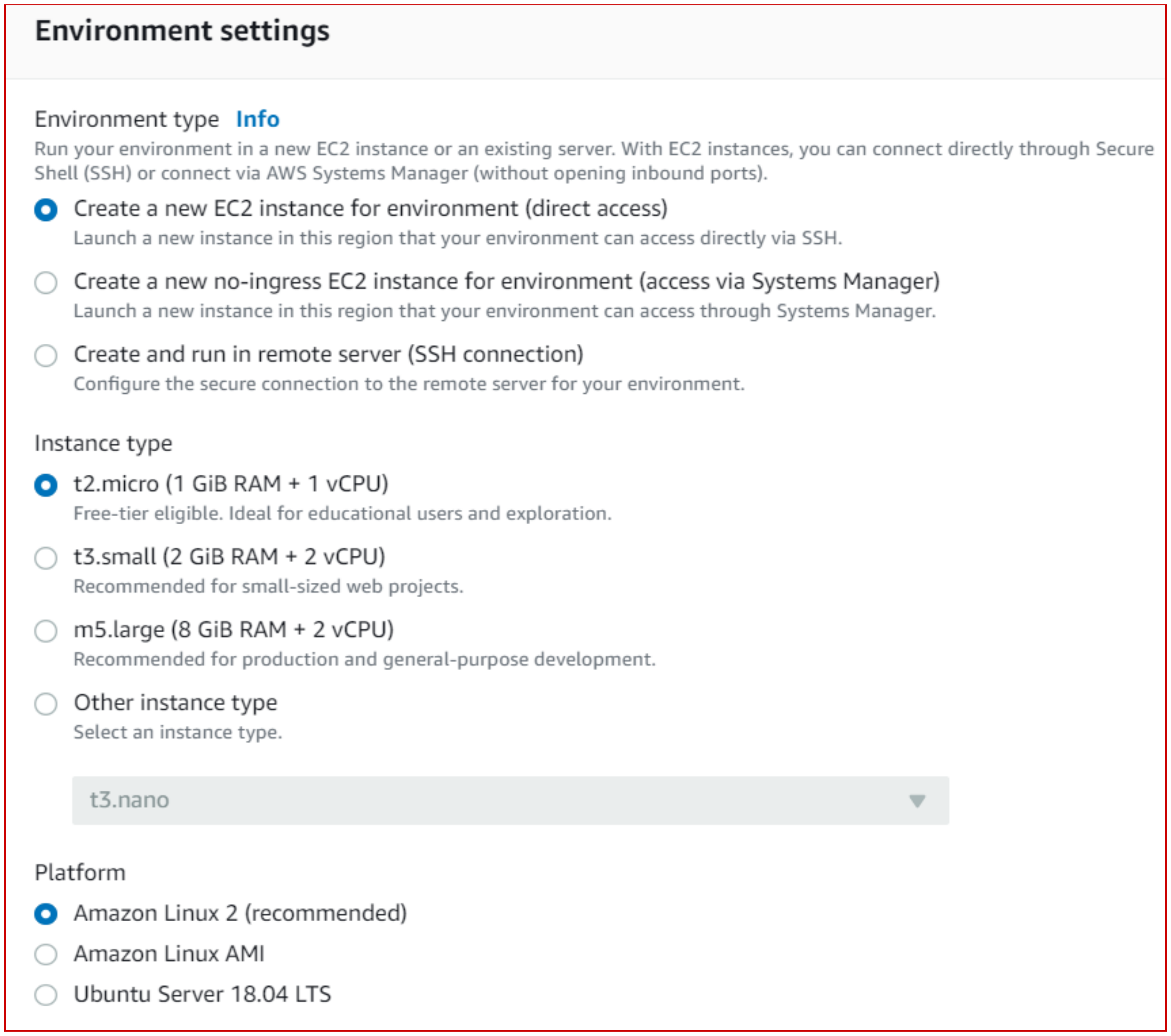
The AWS services used in this project are

- Cloud9
- Elastic Compute Cloud (EC2)
- Identity and Access Management (IAM)
- Amazon Elastic Kubernetes Service (EKS)
- CloudFormation
- Key Management Service (KMS)
- CodePipeline

Set up EKS and create a Kubernetes cluster in Cloud9

Part 1: set up your Cloud9 environment

1. Change AWS region to Oregon (us-west-2) beside your account name on your AWS console.
2. Click on the magnifying glass icon in the topbar and search “Cloud9”.
3. Select “Create environment” on the Cloud9 dashboard.
4. Enter any name such as “eks-lab”.
5. Accept default settings and click “Next step”.
    a. Make sure the instance type is “t2.micro” and the platform is “Amazon Linux 2 (recommended)”.



Environment settings

Environment type Info

Run your environment in a new EC2 instance or an existing server. With EC2 instances, you can connect directly through Secure Shell (SSH) or connect via AWS Systems Manager (without opening inbound ports).

- [✓ Create a new EC2 instance for environment (direct access)] Launch a new instance in this region that your environment can access directly via SSH.
- Create a new no-ingress EC2 instance for environment (access via Systems Manager) Launch a new instance in this region that your environment can access through Systems Manager.
- Create and run in remote server (SSH connection) Configure the secure connection to the remote server for your environment.

Instance type

- [✓ t2.micro (1 GiB RAM + 1 vCPU)] Free-tier eligible. Ideal for educational users and exploration.
- t3.small (2 GiB RAM + 2 vCPU) Recommended for small-sized web projects.
- m5.large (8 GiB RAM + 2 vCPU) Recommended for production and general-purpose development.
- Other instance type Select an instance type.

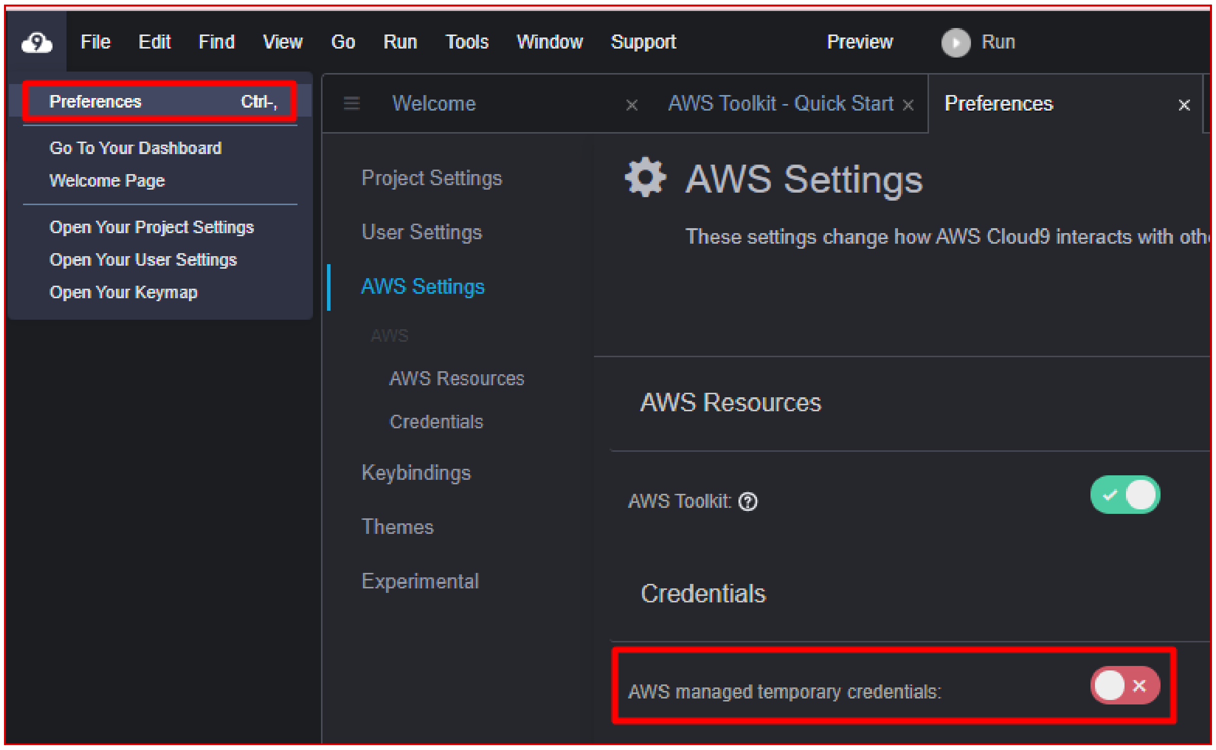
t3.nano

Platform

- [✓ Amazon Linux 2 (recommended)]
- Amazon Linux AMI
- Ubuntu Server 18.04 LTS

6. Review your settings, then select “Create environment”.
7. Once Cloud9 has started, select preferences by clicking on the Cloud9 icon on the top left.

8. Choose the “AWS Settings” tab on the left, and disable “AWS managed temporary credentials” by toggling it to the left.



9. In the terminal, enter `aws configure`.
10. Enter AWS Access Key ID and AWS Secret Access Secret Key (from your rootkey.csv file):
    a. The default region name is us-west-2.
    b. The default output format is json.

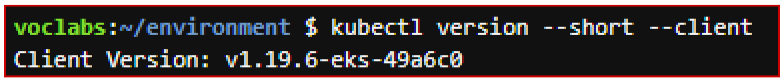
Part 2: install Kubernetes tools in Cloud9

1. Install kubectl. Kubectl is a command line tool that is used for talking with a cluster API server.

```
sudo curl --silent --location -o /usr/local/bin/kubectl \

https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl

sudo chmod +x /usr/local/bin/kubectl
```

In the terminal, you can verify your installation of this tool by entering:

```
kubectl version --short --client
```



```
voclabs:~/environment $ kubectl version --short --client
Client Version: v1.19.6-eks-49a6c0
```

2. Update awscli (AWS Command Line Interface) by running the following commands:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

3. Install jq, envsubst (from GNU gettext utilities) and bash-completion:

```
sudo yum -y install jq gettext bash-completion moreutils
```

4. Install yq for yaml processing:

```
echo 'yq() {
  docker run --rm -i -v "${PWD}":/workdir mikefarah/yq "$@"
}' | tee -a ~/.bashrc && source ~/.bashrc
```

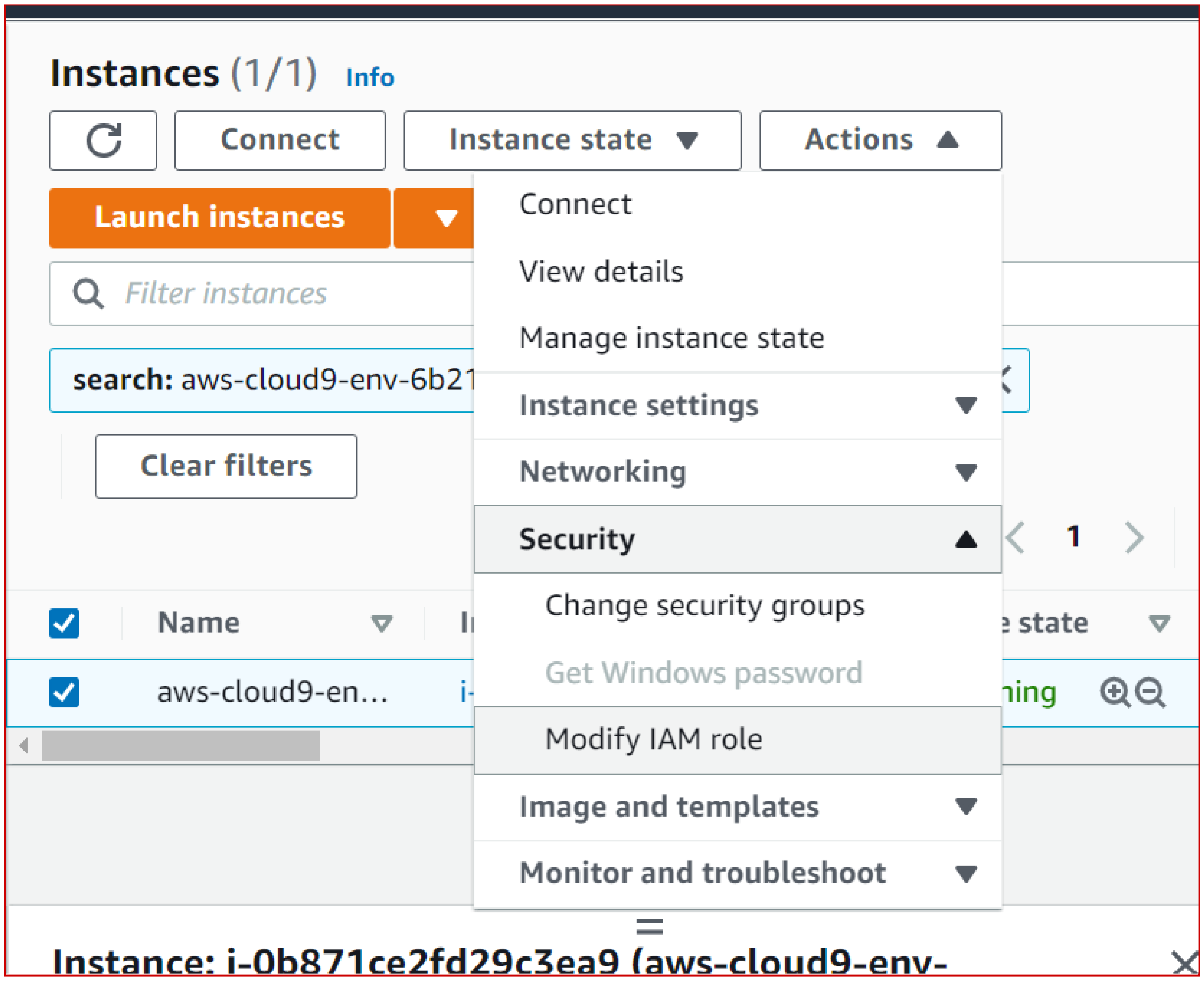
5. Verify the binaries are in the path and executable:

```
for command in kubectl jq envsubst aws
  do
    which $command &>/dev/null && echo "$command in path" || echo "$command
NOT FOUND"
  done
```

6. Enable kubectl bash_completion:

```
kubectl completion bash >>  ~/.bash_completion
. /etc/profile.d/bash_completion.sh
. ~/.bash_completion
```

7. Set the AWS Load Balancer Controller version:
8. Create an IAM role for your workspace. Follow this link and accept all default to create the role with admin access:
    a. https://console.aws.amazon.com/iam/home#/roles$new?step=type&commonUseCase=EC2%2BEC2&selectedUseCase=EC2&policies=arn:aws:iam::aws:policy%2FAdministratorAccess&roleName=eksworkshop-admin
9. Attach the IAM role to your workspace. Locate the Cloud9 instance in your EC2 console. Click “Actions” ⇒ “Security: ⇒ “Modify IAM role”, and click the role created in previous step called “eksworkshop-admin”:



10. Delete temporary credentials by running this command in your Cloud9 terminal:

```
aws cloud9 update-environment  --environment-id $C9_PID
--managed-credentials-action DISABLE
rm -vf ${HOME}/.aws/credentials
```

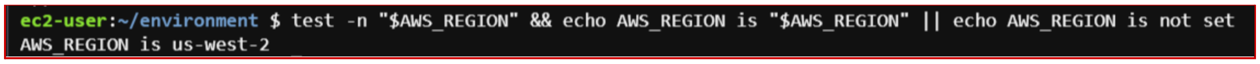
11. Configure AWS CLI with the current region:

```
export ACCOUNT_ID=$(aws sts get-caller-identity --output text --query
Account)
export AWS_REGION=$(curl -s
169.254.169.254/latest/dynamic/instance-identity/document | jq -r
'.region')
export AZS=($(aws ec2 describe-availability-zones --query
'AvailabilityZones[].ZoneName' --output text --region $AWS_REGION))
```

12. Check to make sure AWS_REGION is set to us-west-2:

```
test -n "$AWS_REGION" && echo AWS_REGION is "$AWS_REGION" || echo
AWS_REGION is not set
```

You should get the following output:



```
ec2-user:~/environment $ test -n "$AWS_REGION" && echo AWS_REGION is "$AWS_REGION" || echo AWS_REGION is not set
AWS_REGION is us-west-2
```

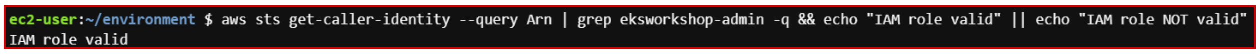
13. Save the above configure to bash_profile:

```
echo "export ACCOUNT_ID=${ACCOUNT_ID}" | tee -a ~/.bash_profile
echo "export AWS_REGION=${AWS_REGION}" | tee -a ~/.bash_profile
echo "export AZS=(${AZS[@]})" | tee -a ~/.bash_profile
aws configure set default.region ${AWS_REGION}
aws configure get default.region
```

14. Validate the IAM role in your terminal:

```
aws sts get-caller-identity --query Arn | grep eksworkshop-admin -q && echo
"IAM role valid" || echo "IAM role NOT valid"
```

You should get the following output in your Cloud9 terminal after running the above command:



```
ec2-user:~/environment $ aws sts get-caller-identity --query Arn | grep eksworkshop-admin -q && echo "IAM role valid" || echo "IAM role NOT valid"
IAM role valid
```

15. Create an AWS KMS Custom Managed Key (CMK).
    a. Create a KMS:

```
aws kms create-alias --alias-name alias/eksworkshop --target-key-id $(aws
kms create-key --query KeyMetadata.Arn --output text)
```

    b. Retrieve the Amazon Resource Name (ARN) of the KMS created in above step and input in to create cluster command:

```
export MASTER_ARN=$(aws kms describe-key --key-id alias/eksworkshop --query
KeyMetadata.Arn --output text)
```

    c. Save MASTER_ARN environment variable to bash_profile:

```
echo "export MASTER_ARN=${MASTER_ARN}" | tee -a ~/.bash_profile
```

Part 3: set up your EKS cluster in Cloud9

Part 3.1: set up the environment to launch your EKS cluster

1. Download eksctl binary by running these commands in your Cloud9 terminal:

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(una
me -s)_amd64.tar.gz" | tar xz -C /tmp

sudo mv -v /tmp/eksctl /usr/local/bin
```

2. Confirm that the eksctl command works:

```
eksctl version
```

You should see a version number as the output.

3. Enable eksctl bash-completion by running this command:

```
eksctl completion bash >> ~/.bash_completion
. /etc/profile.d/bash_completion.sh
. ~/.bash_completion
```

Part 3.2: create a deployment file

1. In your Cloud9 terminal, create a deployment file by running the following command. This is also included as a file called “eksworkshop.yaml” in our artifacts folder.

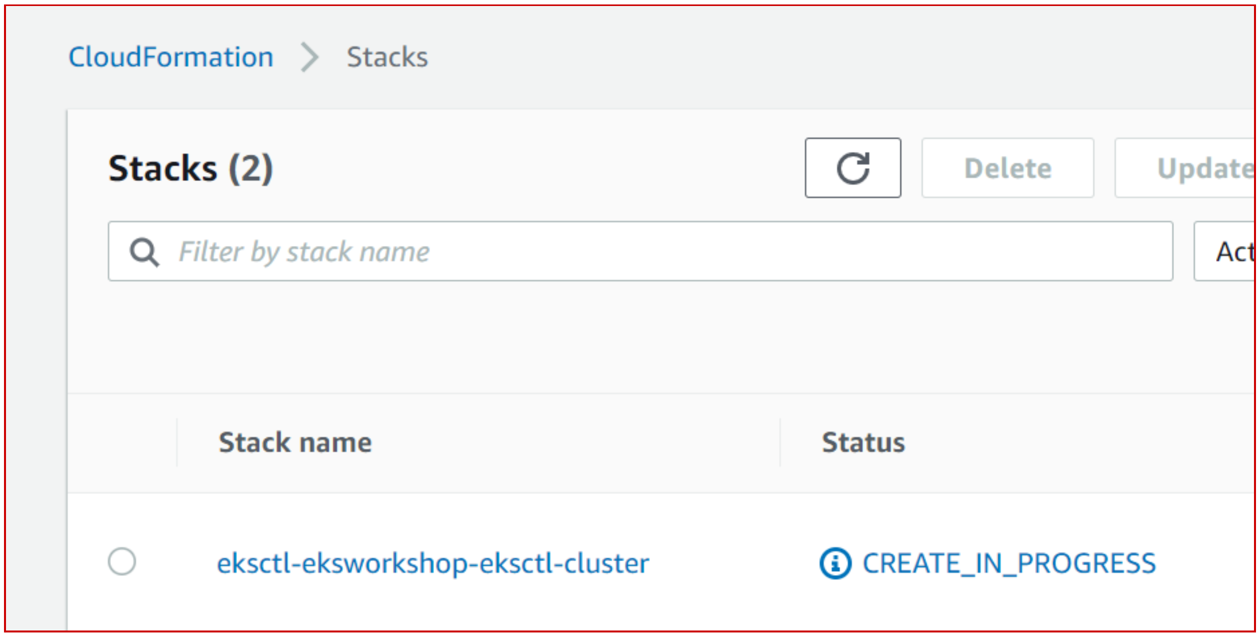
```
cat << EOF > eksworkshop.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: eksworkshop-eksctl
  region: ${AWS_REGION}
  version: "1.19"

availabilityZones: ["${AZS[0]}", "${AZS[1]}", "${AZS[2]}"]

managedNodeGroups:
- name: nodegroup
  desiredCapacity: 3
  instanceType: t3.small
  ssh:
    enableSsm: true

# To enable all of the control plane logs, uncomment below:
# cloudWatch:
#  clusterLogging:
#    enableTypes: ["*"]

secretsEncryption:
  keyARN: ${MASTER_ARN}
EOF
```

Part 3.3: create your EKS cluster

1. To create an EKS cluster, run this command. It should contain your deployment file name. In this example, the deployment file name is called “eksworkshop.yaml”:

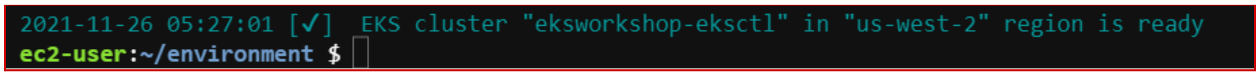
```
eksctl create cluster -f eksworkshop.yaml
```

Note that this step may take 15 to 20 minutes to complete. Most of this time is spent waiting for the CloudFormation stacks to be created. You can check the current status of the creation of the stacks by navigation to the CloudFormation page in your AWS management console:



Stack name	Status
eksctl-eksworkshop-eksctl-cluster	CREATE_IN_PROGRESS

Once cluster creation is completed, you should see a message similar to the following screenshot:

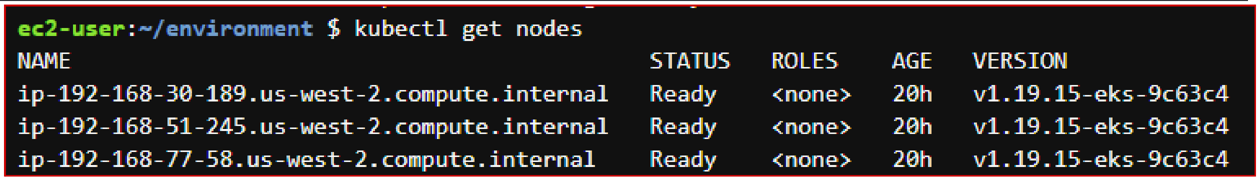


```
2021-11-26 05:27:01 [✓]  EKS cluster "eksworkshop-eksctl" in "us-west-2" region is ready
ec2-user:~/environment $ 
```

Part 3.4: test your cluster

1. Confirm that there are three nodes created by the previous step:

```
kubectl get nodes
```



```
ec2-user:~/environment $ kubectl get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
ip-192-168-30-189.us-west-2.compute.internal  Ready    <none>   20h   v1.19.15-eks-9c63c4
ip-192-168-51-245.us-west-2.compute.internal  Ready    <none>   20h   v1.19.15-eks-9c63c4
ip-192-168-77-58.us-west-2.compute.internal   Ready    <none>   20h   v1.19.15-eks-9c63c4
```

2. Export the Worker Role Name for later use:

```
STACK_NAME=$(eksctl get nodegroup --cluster eksworkshop-eksctl -o json | jq
-r '.[].StackName')
ROLE_NAME=$(aws cloudformation describe-stack-resources --stack-name
$STACK_NAME | jq -r '.StackResources[] |
select(.ResourceType=="AWS::IAM::Role") | .PhysicalResourceId')
echo "export ROLE_NAME=${ROLE_NAME}" | tee -a ~/.bash_profile
```

Set up CodePipeline

To set up a CodePipeline, CodeBuild will be used to deploy Kubernetes service. First, AWS CodeBuild requires us to set up an IAM role which allows us to interact with the EKS cluster. Then we have to add the IAM role to the aws-auth configmap.

Next, we will fork a github repository. By changing content inside repository, our builds will be triggered automatically

Part 1: create an IAM role and generate a GitHub personal Token

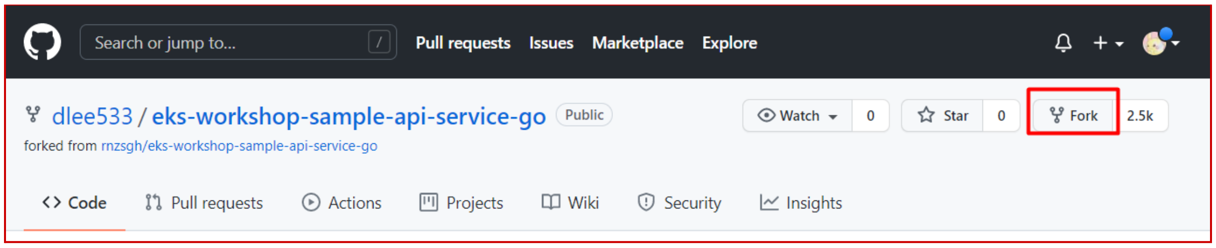
1. Create IAM assume role for root access (EksWorkshopCodeBuildKubectlRole):

```
cd ~/environment

TRUST="{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Effect\":
\"Allow\", \"Principal\": { \"AWS\": \"arn:aws:iam::${ACCOUNT_ID}:root\" },
\"Action\": \"sts:AssumeRole\" } ] }"

echo '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow",
"Action": "eks:Describe*", "Resource": "*" } ] }' > /tmp/iam-role-policy

aws iam create-role --role-name EksWorkshopCodeBuildKubectlRole
--assume-role-policy-document "$TRUST" --output text --query 'Role.Arn'

aws iam put-role-policy --role-name EksWorkshopCodeBuildKubectlRole
--policy-name eks-describe --policy-document file:///tmp/iam-role-policy
```

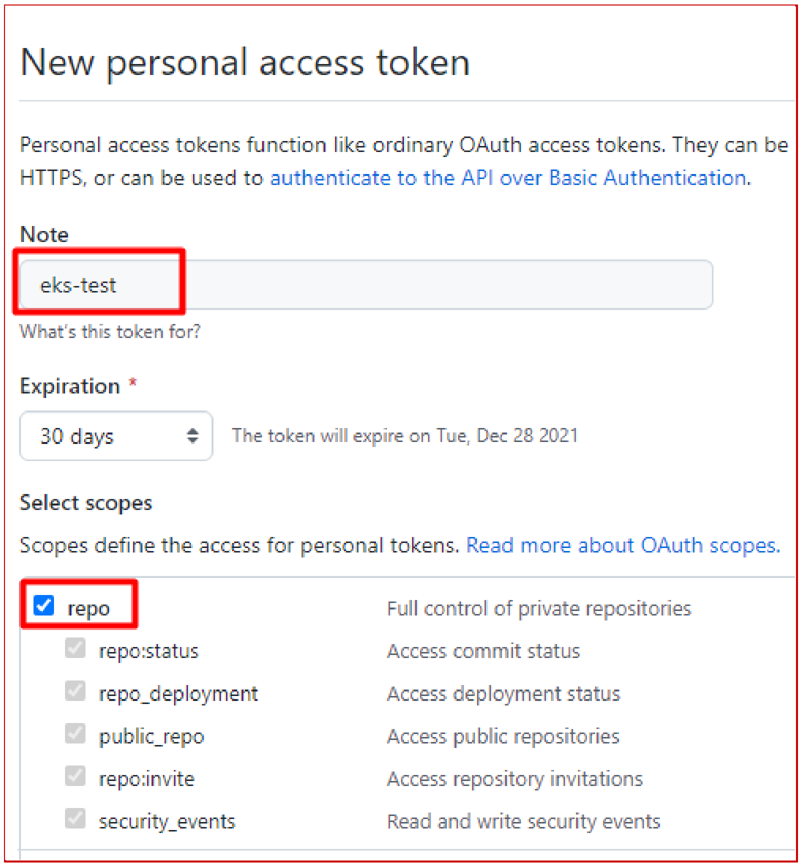
2. Modify aws-auth configmap by running the following commands:

```
ROLE="    - rolearn:
arn:aws:iam::${ACCOUNT_ID}:role/EksWorkshopCodeBuildKubectlRole\n
username: build\n      groups:\n        - system:masters"

kubectl get -n kube-system configmap/aws-auth -o yaml | awk "/mapRoles:
\|/{print;print \"$ROLE\";next}1" > /tmp/aws-auth-patch.yml

kubectl patch configmap/aws-auth -n kube-system --patch "$(cat
/tmp/aws-auth-patch.yml)"
```

3. Fork the sample Kubernete service by going here: https://github.com/dlee533/eks-workshop-sample-api-service-go



4. Generate a github token by going here: https://github.com/settings/tokens/new
   a. You will be prompted to enter your github password
5. Generate a new token by entering a token name under Notes and check repo under Select scopes. Then click 'Generate Token'



New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

eks-test

What's this token for?

Expiration *

30 days

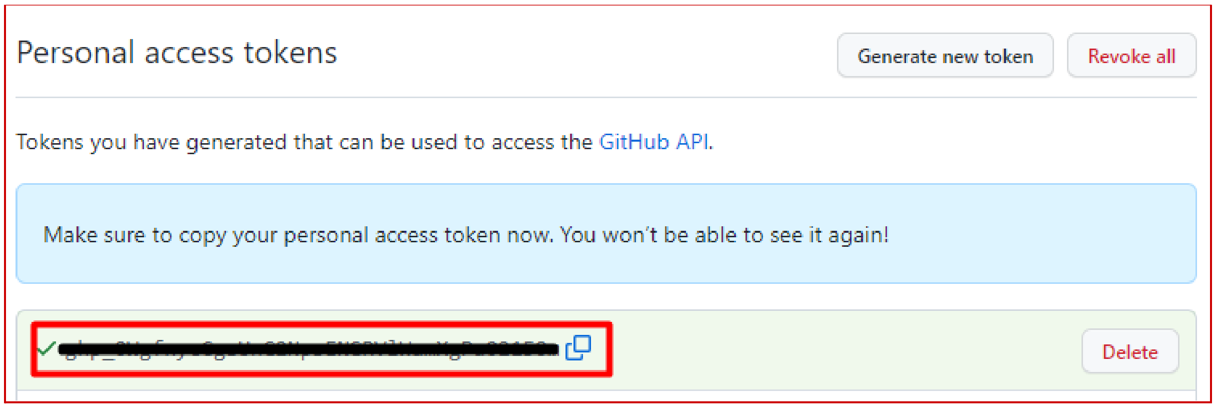
The token will expire on Tue, Dec 28 2021

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

☑ repo	Full control of private repositories
☑ repo:status	Access commit status
☑ repo_deployment	Access deployment status
☑ public_repo	Access public repositories
☑ repo:invite	Access repository invitations
☑ security_events	Read and write security events

6. Copy generated token onto a txt file and store it at a secure place


Personal access tokens

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your personal access token now. You won't be able to see it again!

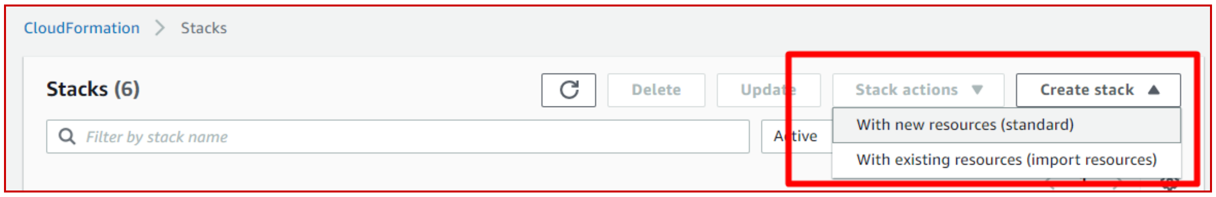
✓ [redacted]

Part 2: set up CodePipeline

CloudFormation templates contain information that are required to set up each EKS service/deployment. AWS Codepipeline is then created by CloudFormation stacks.

Each CodePipeline corresponds to one EKS service in a separate repository.

1. Navigate to the CloudFormation dashboard ⇒ Click on Create Stack ⇒ with new resources (standard) ⇒ In the Amazon S3 URL, enter: https://s3.amazonaws.com/eksworkshop.com/templates/main/ci-cd-codepipeline.cfn.yml

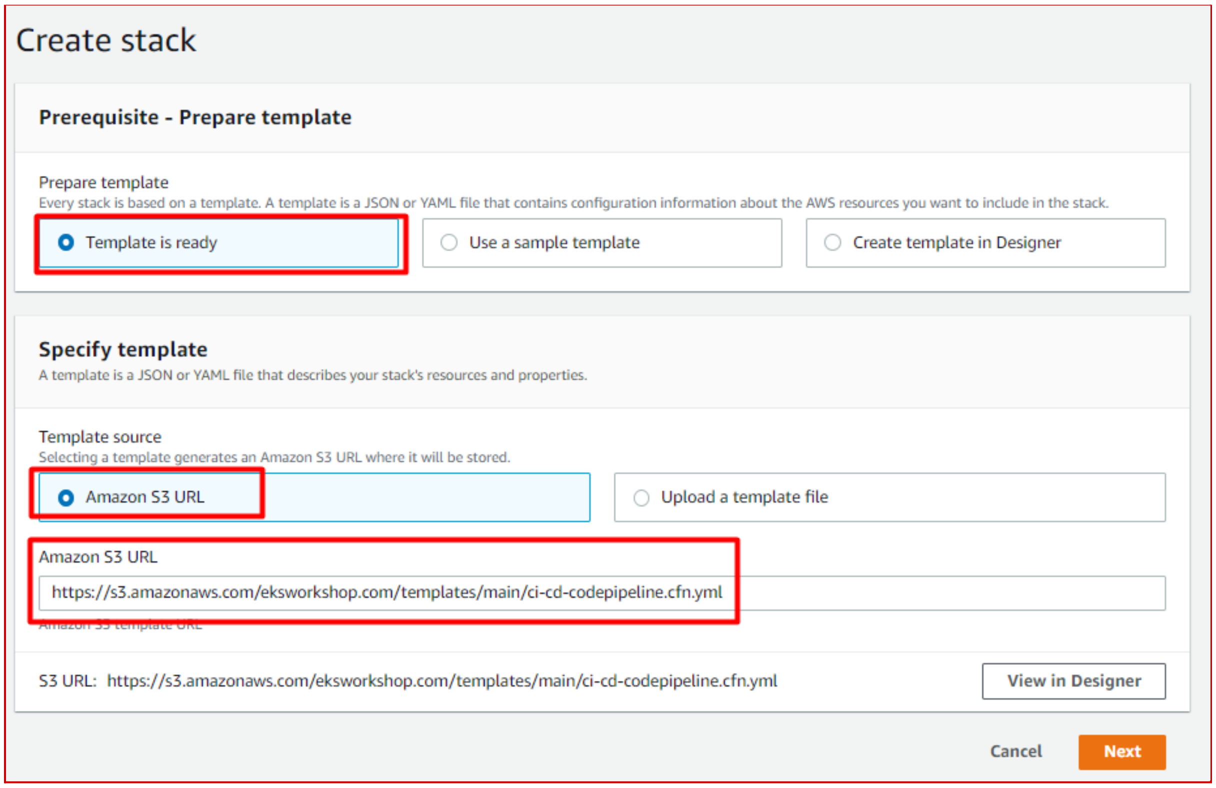

CloudFormation > Stacks

Stacks (6)

Filter by stack name

Active

Delete
Update
Stack actions
Create stack
With new resources (standard)
With existing resources (import resources)


Create stack

Prerequisite - Prepare template

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

[✓ Template is ready] Use a sample template Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

[✓ Amazon S3 URL] Upload a template file

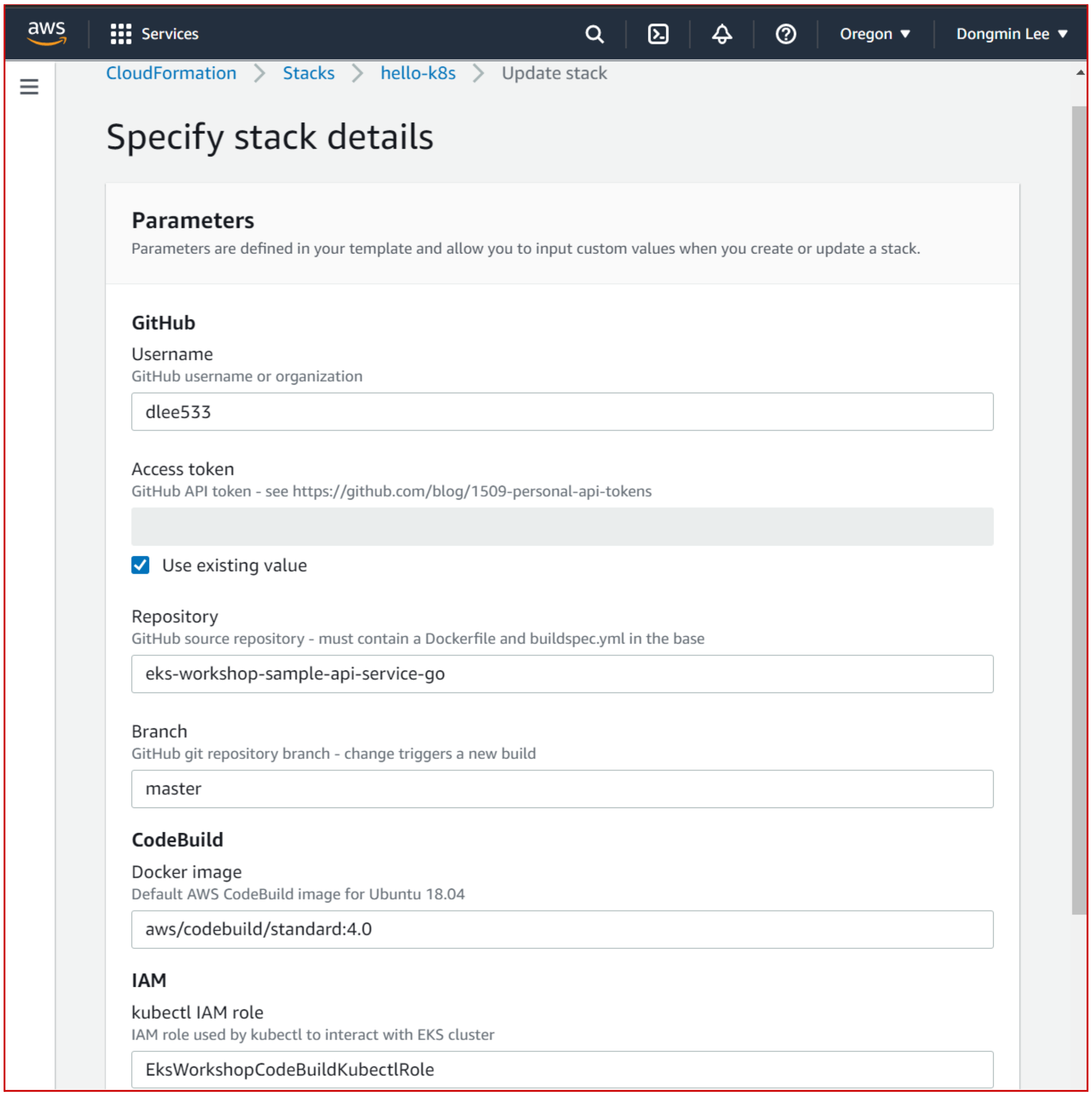
Amazon S3 URL

https://s3.amazonaws.com/eksworkshop.com/templates/main/ci-cd-codepipeline.cfn.yml

Amazon S3 template URL

S3 URL: https://s3.amazonaws.com/eksworkshop.com/templates/main/ci-cd-codepipeline.cfn.yml

2. In next page, edit to following values in the screenshot, then create stack



Specify stack details

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

GitHub

Username
GitHub username or organization
dlee533

Access token
GitHub API token - see https://github.com/blog/1509-personal-api-tokens

☑ Use existing value

Repository
GitHub source repository - must contain a Dockerfile and buildspec.yml in the base
eks-workshop-sample-api-service-go

Branch
GitHub git repository branch - change triggers a new build
master

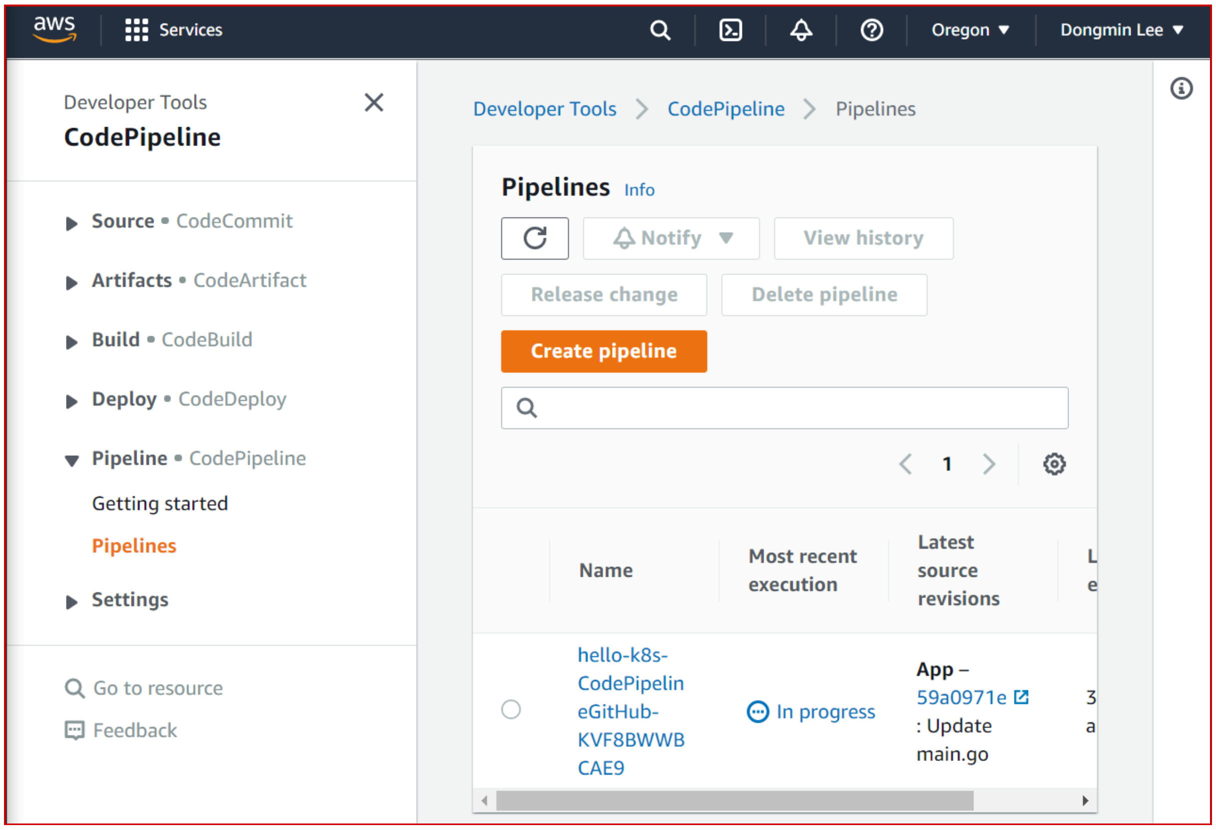
CodeBuild

Docker image
Default AWS CodeBuild image for Ubuntu 18.04
aws/codebuild/standard:4.0

IAM

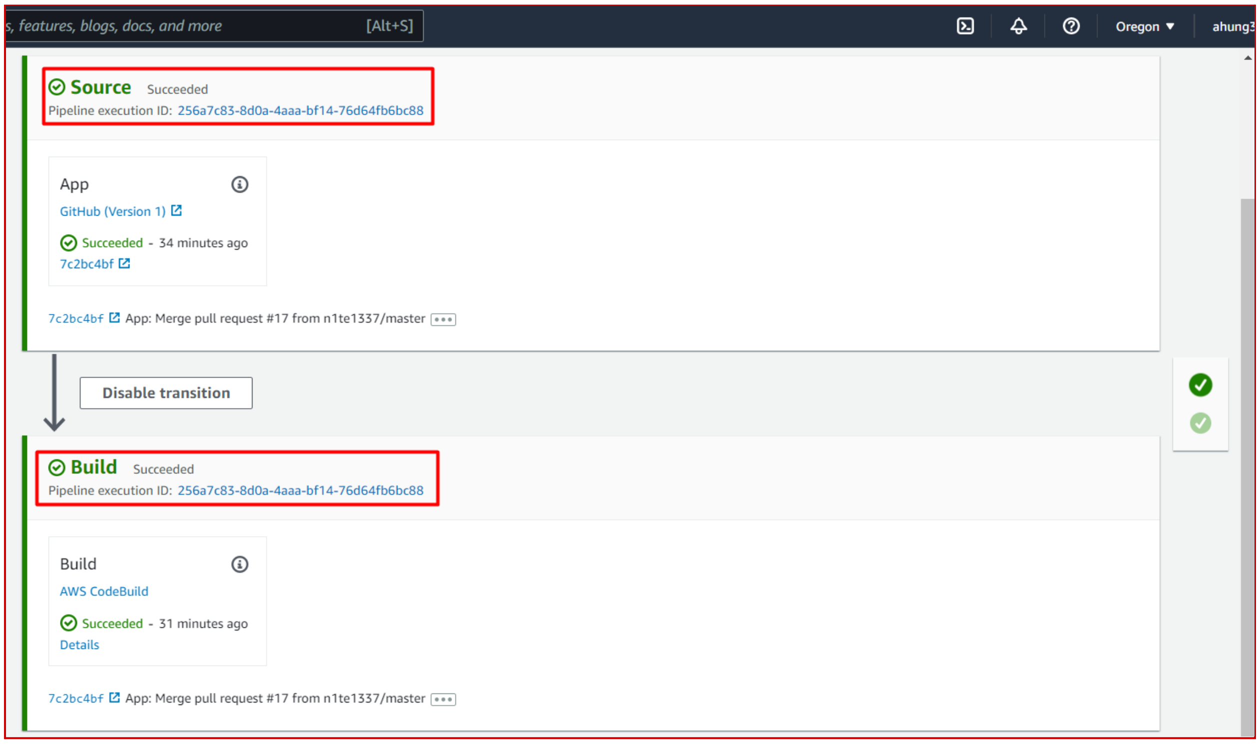
kubectl IAM role
IAM role used by kubectl to interact with EKS cluster
EksWorkshopCodeBuildKubectlRole

3. From the AWS Developer Console ⇒ CopePipeline ⇒ Pipeline and wait for the green check marks to appear beside Source and Build steps



Name	Most recent execution	Latest source revisions
hello-k8s-CodePipelineGitHub-KVF8BWWBCAE9	In progress	App – 59a0971e : Update main.go

4. You will see the output below when the build is succeeded



Source Succeeded
Pipeline execution ID: 256a7c83-8d0a-4aaa-bf14-76d64fb6bc88

App
GitHub (Version 1)
Succeeded - 34 minutes ago
7c2bc4bf

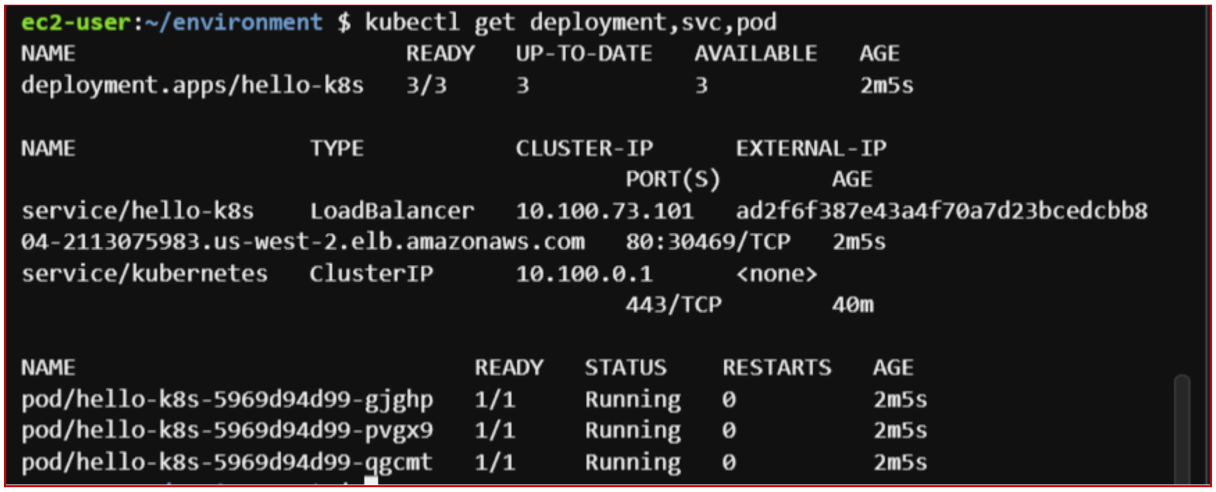
7c2bc4bf App: Merge pull request #17 from n1te1337/master

Build Succeeded
Pipeline execution ID: 256a7c83-8d0a-4aaa-bf14-76d64fb6bc88

Build
AWS CodeBuild
Succeeded - 31 minutes ago
Details

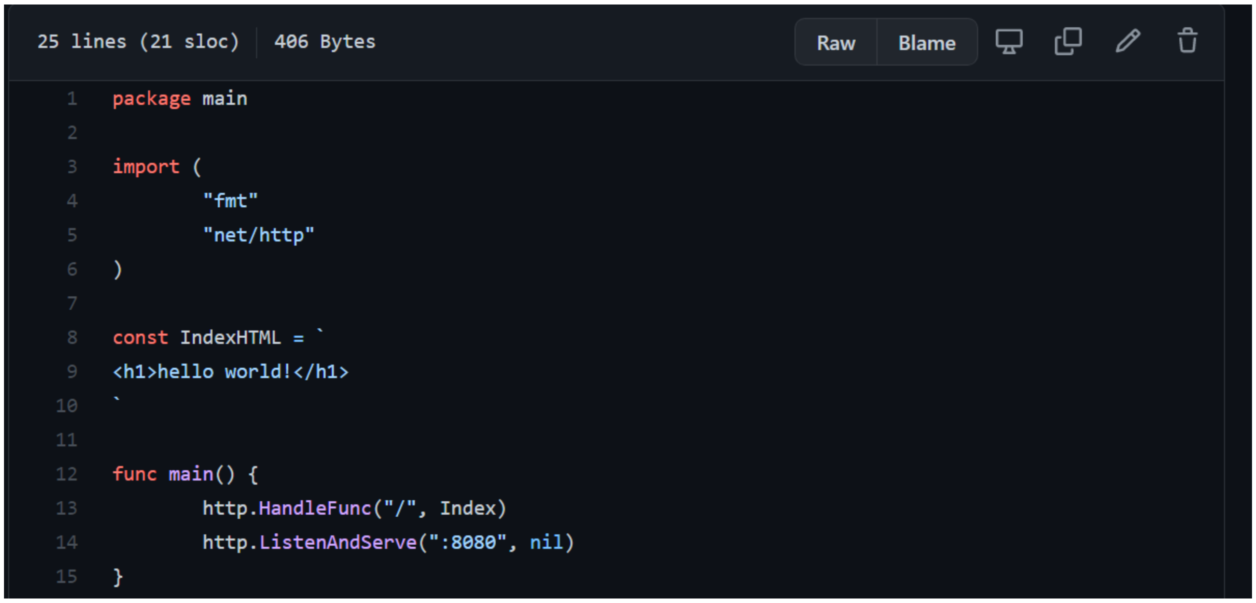
7c2bc4bf App: Merge pull request #17 from n1te1337/master

5. From your cloud9 environment, to check the running components, type in `kubectl get deploy,svc,pod`
6. Go to the external ip of the LoadBalancer to check the response from application deployed in kubernetes



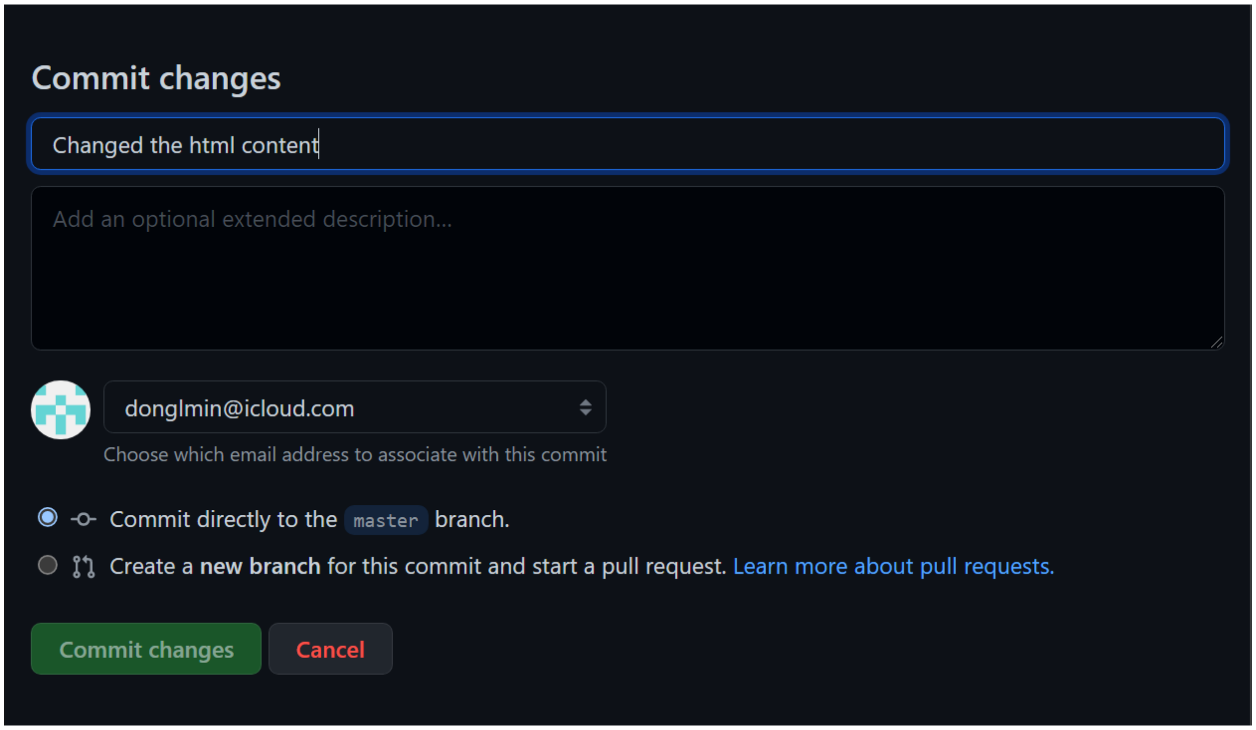
```
ec2-user:~/environment $ kubectl get deployment,svc,pod
NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hello-k8s   3/3     3            3           2m5s

NAME                   TYPE           CLUSTER-IP      EXTERNAL-IP
                                                PORT(S)        AGE
service/hello-k8s      LoadBalancer   10.100.73.101   ad2f6f387e43a4f70a7d23bcedcbb8
04-2113075983.us-west-2.elb.amazonaws.com   80:30469/TCP   2m5s
service/kubernetes     ClusterIP      10.100.0.1      <none>
                                                443/TCP        40m

NAME                             READY   STATUS    RESTARTS   AGE
pod/hello-k8s-5969d94d99-gjghp   1/1     Running   0          2m5s
pod/hello-k8s-5969d94d99-pvgx9   1/1     Running   0          2m5s
pod/hello-k8s-5969d94d99-qgcmt   1/1     Running   0          2m5s
```

7. Changing the source code in the forked github repository will prompt the pipeline to run
8. Navigate to the github repository and click the main.go file. We can try changing the message inside main.go



25 lines (21 sloc) | 406 Bytes

```go
package main

import (
        "fmt"
        "net/http"
)

const IndexHTML = `
<h1>hello world!</h1>
`

func main() {
        http.HandleFunc("/", Index)
        http.ListenAndServe(":8080", nil)
}
```



Commit changes

Changed the html content

Add an optional extended description...

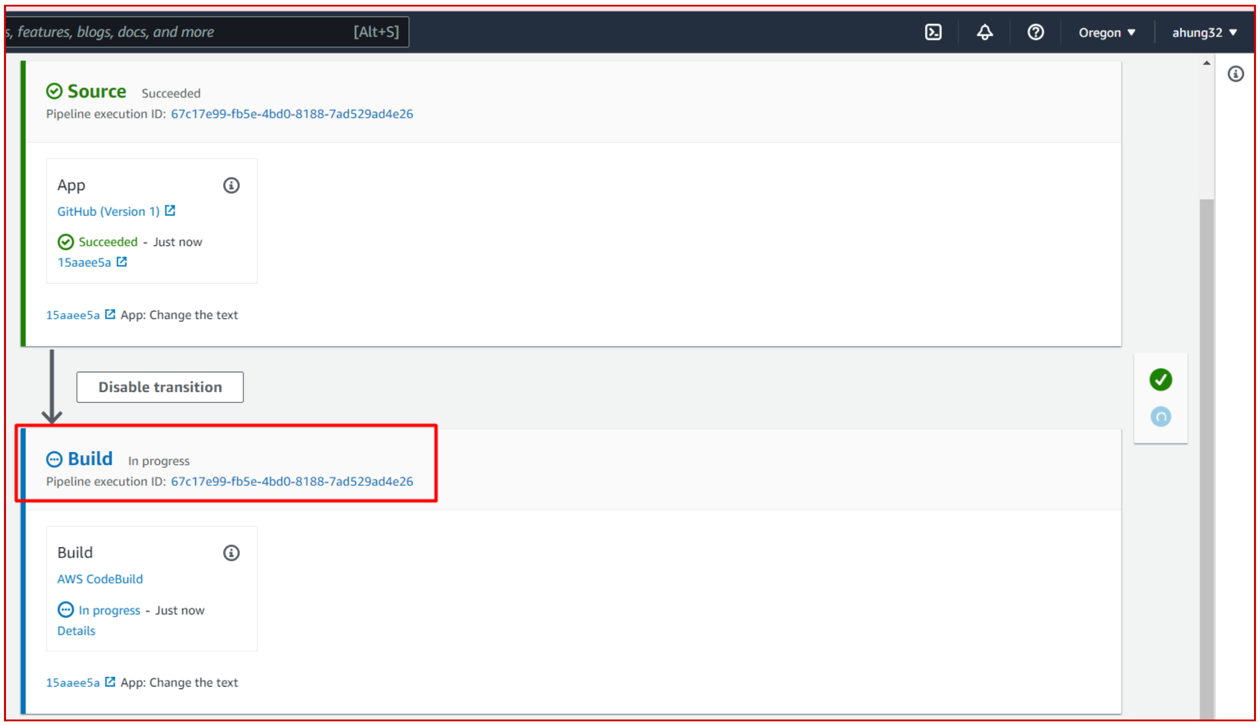
donglmin@icloud.com

Choose which email address to associate with this commit

[✓ Commit directly to the `master` branch.]

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

9. You should see that build is in progress



Source Succeeded
Pipeline execution ID: 67c17e99-fb5e-4bd0-8188-7ad529ad4e26

App
GitHub (Version 1)
Succeeded - Just now
15aaee5a

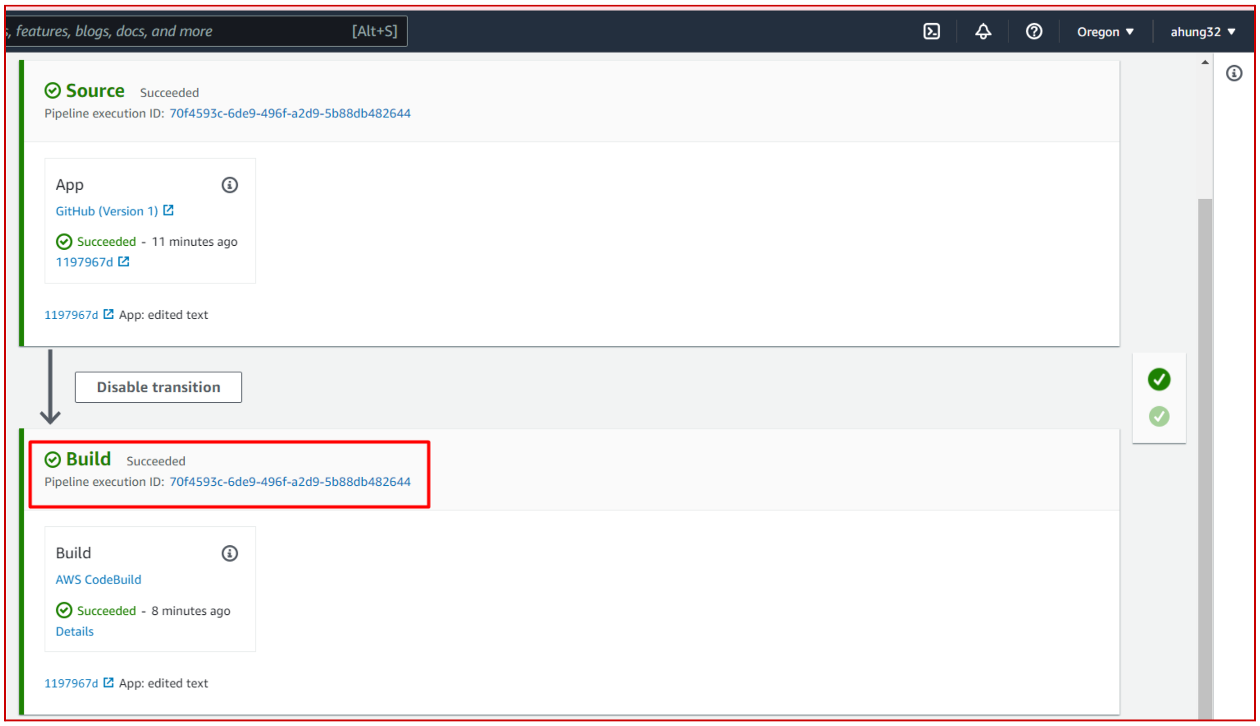
15aaee5a App: Change the text

Build In progress
Pipeline execution ID: 67c17e99-fb5e-4bd0-8188-7ad529ad4e26

Build
AWS CodeBuild
In progress - Just now
Details

15aaee5a App: Change the text

10. You will see the output below once the build has succeeded



Source Succeeded
Pipeline execution ID: 70f4593c-6de9-496f-a2d9-5b88db482644

App
GitHub (Version 1)
Succeeded - 11 minutes ago
1197967d

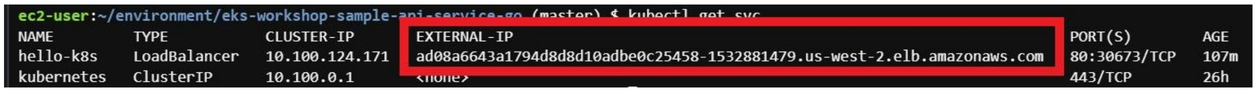
1197967d App: edited text

Build Succeeded
Pipeline execution ID: 70f4593c-6de9-496f-a2d9-5b88db482644

Build
AWS CodeBuild
Succeeded - 8 minutes ago
Details

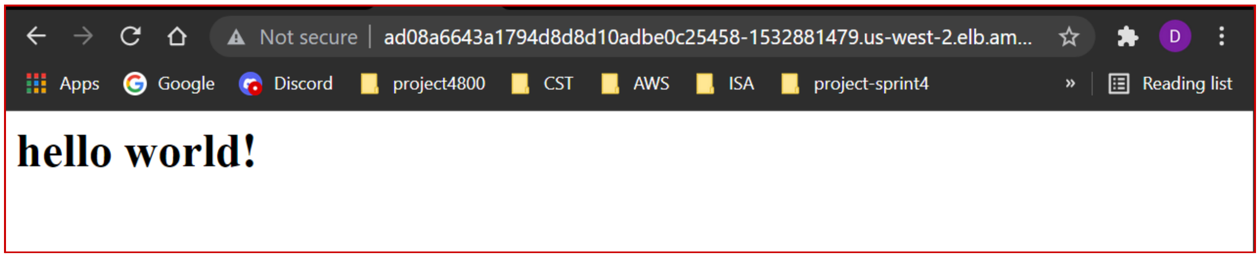
1197967d App: edited text

11. Check the external ip of the load balancer service by running `kubectl get svc`



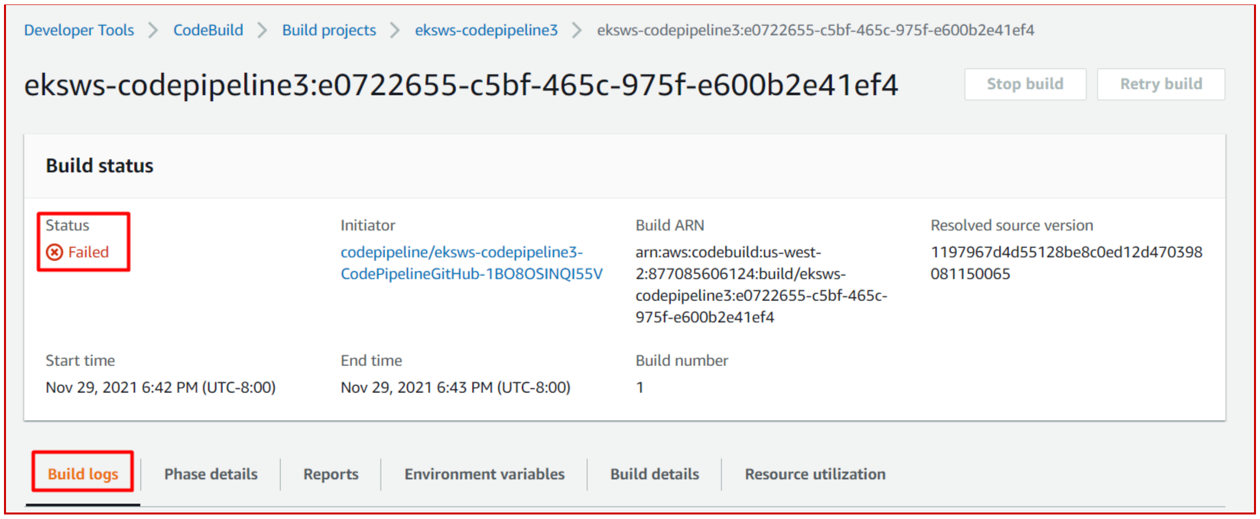
```
ec2-user:~/environment/eks-workshop-sample-api-service-go (master) $ kubectl get svc
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP                                                                 PORT(S)        AGE
hello-k8s    LoadBalancer   10.100.124.171   ad08a6643a1794d8d8d10adbe0c25458-1532881479.us-west-2.elb.amazonaws.com   80:30673/TCP   107m
kubernetes   ClusterIP      10.100.0.1       <none>                                                                      443/TCP        26h
```

12. Go to the external ip of the load balancer in the browser. Check to make sure that the page has the output as below



hello world!

13. If your build fails, please navigate to Build projects ⇒ eksws-codepipeline ⇒ Build logs tab. You may see the error where it says `docker build --tag $REPOSITORY_URI$TAG`

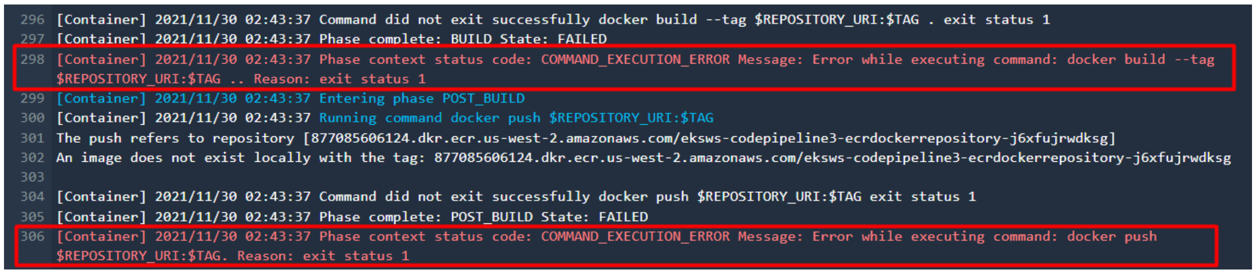

Developer Tools > CodeBuild > Build projects > eksws-codepipeline3 > eksws-codepipeline3:e0722655-c5bf-465c-975f-e600b2e41ef4
Stop build
Retry build
Build logs
Phase details
Reports
Environment variables
Build details
Resource utilization

eksws-codepipeline3:e0722655-c5bf-465c-975f-e600b2e41ef4

Build status

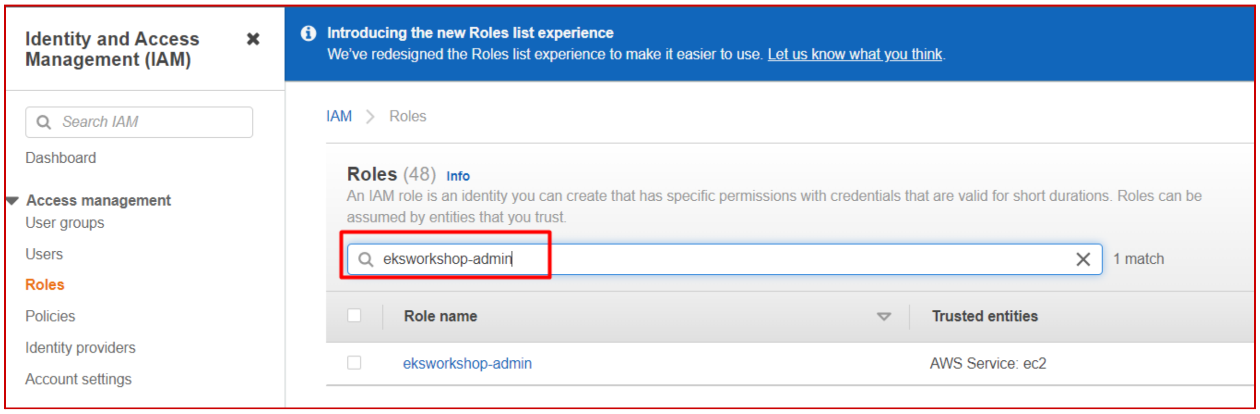
Status	Initiator	Build ARN	Resolved source version
Failed	codepipeline/eksws-codepipeline3-CodePipelineGitHub-1BO8OSINQI55V	arn:aws:codebuild:us-west-2:877085606124:build/eksws-codepipeline3:e0722655-c5bf-465c-975f-e600b2e41ef4	1197967d4d55128be8c0ed12d470398081150065

Start time	End time	Build number
Nov 29, 2021 6:42 PM (UTC-8:00)	Nov 29, 2021 6:43 PM (UTC-8:00)	1



```
[Container] 2021/11/30 02:43:37 Command did not exit successfully docker build --tag $REPOSITORY_URI:$TAG . exit status 1
[Container] 2021/11/30 02:43:37 Phase complete: BUILD State: FAILED
[Container] 2021/11/30 02:43:37 Phase context status code: COMMAND_EXECUTION_ERROR Message: Error while executing command: docker build --tag
$REPOSITORY_URI:$TAG .. Reason: exit status 1
[Container] 2021/11/30 02:43:37 Entering phase POST_BUILD
[Container] 2021/11/30 02:43:37 Running command docker push $REPOSITORY_URI:$TAG
The push refers to repository [877085606124.dkr.ecr.us-west-2.amazonaws.com/eksws-codepipeline3-ecrdockerrepository-j6xfujrwdksg]
An image does not exist locally with the tag: 877085606124.dkr.ecr.us-west-2.amazonaws.com/eksws-codepipeline3-ecrdockerrepository-j6xfujrwdksg

[Container] 2021/11/30 02:43:37 Command did not exit successfully docker push $REPOSITORY_URI:$TAG exit status 1
[Container] 2021/11/30 02:43:37 Phase complete: POST_BUILD State: FAILED
[Container] 2021/11/30 02:43:37 Phase context status code: COMMAND_EXECUTION_ERROR Message: Error while executing command: docker push
$REPOSITORY_URI:$TAG. Reason: exit status 1
```

14. If you have the same error as above screenshots, please navigate to the IAM role console to change your IAM role to include AmazonEC2ContainerRegistryFullAccess



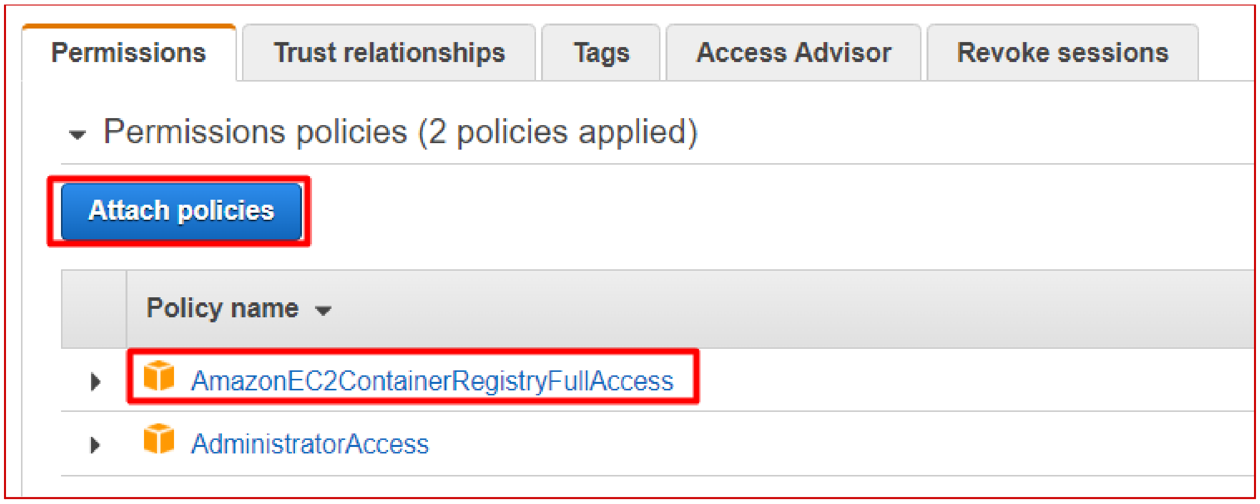
Introducing the new Roles list experience
We've redesigned the Roles list experience to make it easier to use. Let us know what you think.

Roles (48)
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

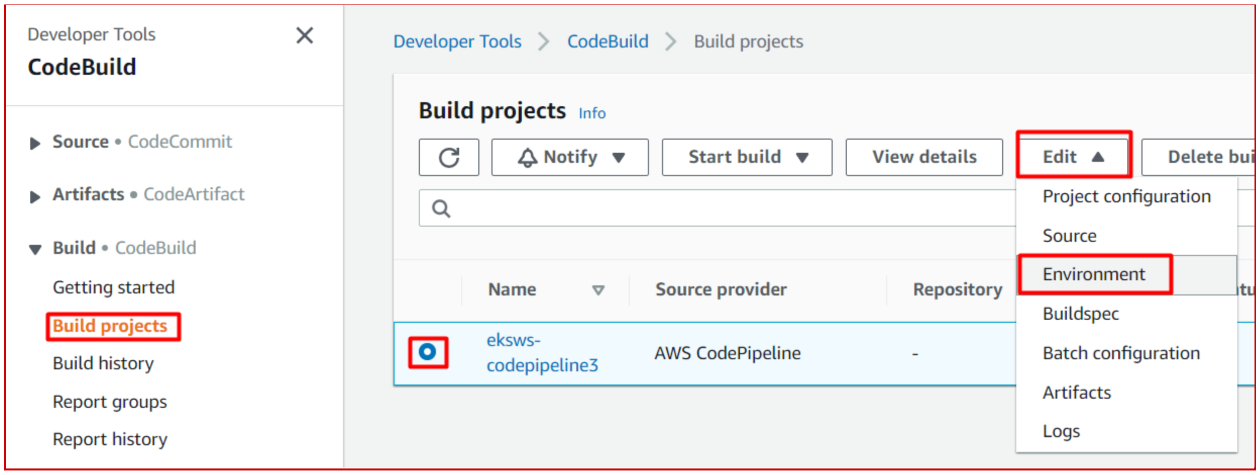
eksworkshop-admin — 1 match

Role name	Trusted entities
eksworkshop-admin	AWS Service: ec2

15. Select eksworkshop-admin role. Under Permissions, click Attach policies and enter AmazonEC2ContainerRegistryFullAccess in the search bar

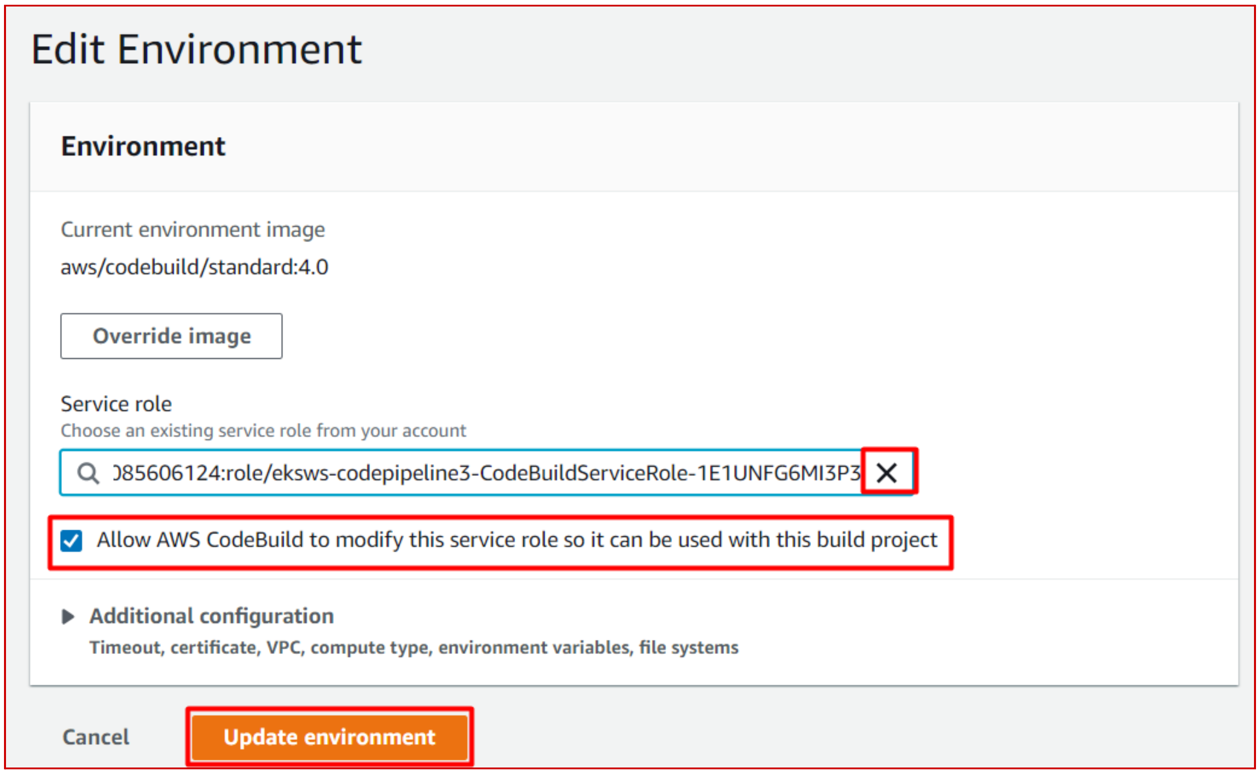


16. Navigate back to CodeBuild. On the side bar, select Build projects, then click on the radio button besides eks-codepipeline ⇒ select Edit ⇒ Environment



17. Clear IAM role by clicking ‘x’, then add the IAM role back in. Make sure to check the box where it says ‘Allow AWS CodeBuild to modify this service role so it can be used with

this building project'. Select update environment.



Edit Environment

Environment

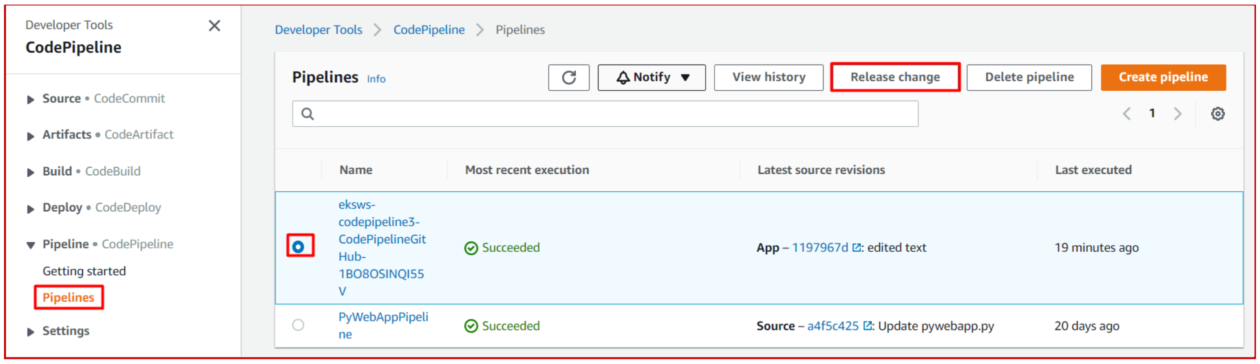
Current environment image
aws/codebuild/standard:4.0

Service role
Choose an existing service role from your account
)85606124:role/eksws-codepipeline3-CodeBuildServiceRole-1E1UNFG6MI3P3

☑ Allow AWS CodeBuild to modify this service role so it can be used with this build project

Additional configuration
Timeout, certificate, VPC, compute type, environment variables, file systems

18. Navigate to pipelines on the side bar and click on the radio button beside eksws-codepipeline ⇒ select Release change to rebuild the pipeline.



Name	Most recent execution	Latest source revisions	Last executed
eksws-codepipeline3-CodePipelineGitHub-1BO8OSINQI55V	Succeeded	App – 1197967d: edited text	19 minutes ago
PyWebAppPipeline	Succeeded	Source – a4f5c425: Update pywebapp.py	20 days ago

19. This will resolve the repository uri error.

Part 2.1: cleanup

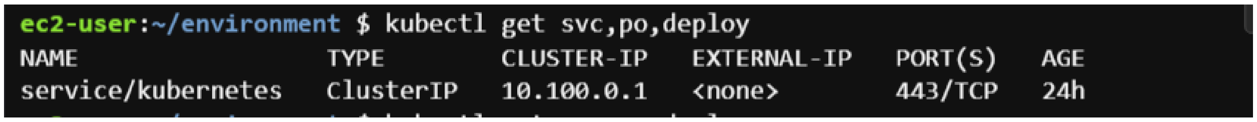
1. Run following codes in the Cloud9 environment to delete deployment and service

```
kubectl delete deployment hello-k8s
kubectl delete service hello-k8s
```

2. Make sure the deployment, service and pods are deleted by running following code

```
kubectl get deploy,svc,po
```

If successfully deleted, only the service/kubernetes will be remained



```
ec2-user:~/environment $ kubectl get svc,po,deploy
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   24h
```

Introduction to Helm

What is Helm?

Helm is a Kubernetes package manager that helps streamline the setup, configuration, and deployment of an application on Kubernetes clusters.

Helm Charts

A Chart is a packaging format in Helm. A Chart is a group of files that packages and defines sets of resources in Kubernetes. A Chart is helpful for deployment because you can simply run a command to install the dependencies of a project instead of manually installing various Kubernetes resources for your application yourself through `kubectl`.

Section overview

In this module, you will be introduced to working with Helm. The module includes a walkthrough on how to set up Helm on your AWS Cloud9 environment with the CLI and how to configure your first Helm Chart repository. This module also goes through how to carry out a Helm Chart deployment using bitnami/apache for a standalone Apache web server and for a static web application using its Git repository.

Set up and use Helm

Prerequisites

Prior to starting going through the following steps involving Helm, please ensure that you have launched an AWS Cloud9 environment in the Oregon (us-west-2) region.

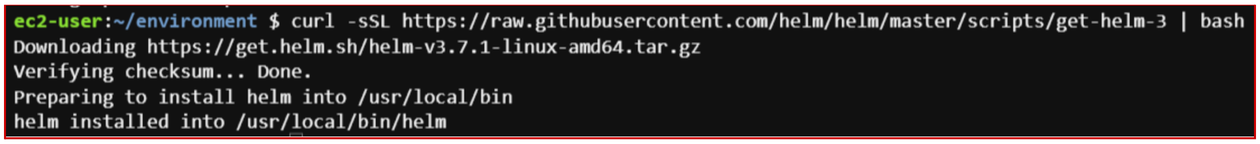
Additionally, please follow the steps from the “Setting up EKS and create a Kubernetes cluster in Cloud9” section of the project that goes through how to launch EKS and create a cluster using Cloud9.

Part 1: introduction

Part 1.1: install the Helm Command Line Interface (CLI)

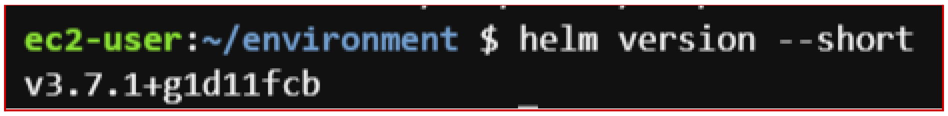
1. Install the Helm CLI by running the following command:

```
curl -sSL
https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 |
bash
```



```
ec2-user:~/environment $ curl -sSL https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
Downloading https://get.helm.sh/helm-v3.7.1-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
```

2. Check that Helm is installed by running `helm version --short`

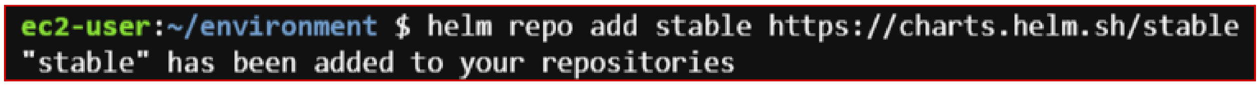


```
ec2-user:~/environment $ helm version --short
v3.7.1+g1d11fcb
```

Part 1.2: add your first Helm Chart repository

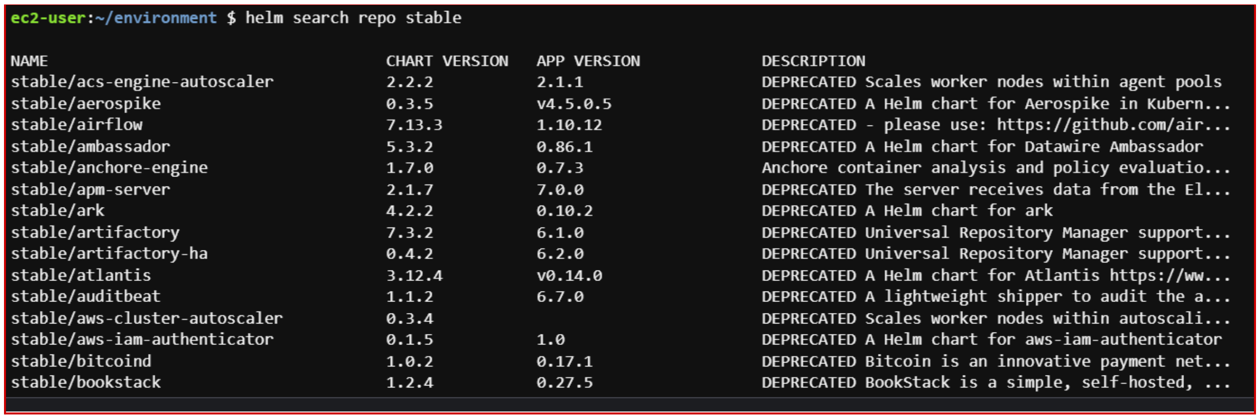
1. Start by downloading the `stable` repository. The `stable` repository is the default Chart repository in Helm:

```
helm repo add stable https://charts.helm.sh/stable
```



```
ec2-user:~/environment $ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
```

2. Run `helm search repo stable` to see a list of possible charts that you can install:



```
ec2-user:~/environment $ helm search repo stable

NAME                              CHART VERSION   APP VERSION         DESCRIPTION
stable/acs-engine-autoscaler      2.2.2           2.1.1               DEPRECATED Scales worker nodes within agent pools
stable/aerospike                  0.3.5           v4.5.0.5            DEPRECATED A Helm chart for Aerospike in Kubern...
stable/airflow                    7.13.3          1.10.12             DEPRECATED - please use: https://github.com/air...
stable/ambassador                 5.3.2           0.86.1              DEPRECATED A Helm chart for Datawire Ambassador
stable/anchore-engine             1.7.0           0.7.3               Anchore container analysis and policy evaluatio...
stable/apm-server                 2.1.7           7.0.0               DEPRECATED The server receives data from the El...
stable/ark                        4.2.2           0.10.2              DEPRECATED A Helm chart for ark
stable/artifactory                7.3.2           6.1.0               DEPRECATED Universal Repository Manager support...
stable/artifactory-ha             0.4.2           6.2.0               DEPRECATED Universal Repository Manager support...
stable/atlantis                   3.12.4          v0.14.0             DEPRECATED A Helm chart for Atlantis https://ww...
stable/auditbeat                  1.1.2           6.7.0               DEPRECATED A lightweight shipper to audit the a...
stable/aws-cluster-autoscaler     0.3.4                               DEPRECATED Scales worker nodes within autoscali...
stable/aws-iam-authenticator      0.1.5           1.0                 DEPRECATED A Helm chart for aws-iam-authenticator
stable/bitcoind                   1.0.2           0.17.1              DEPRECATED Bitcoin is an innovative payment net...
stable/bookstack                  1.2.4           0.27.5              DEPRECATED BookStack is a simple, self-hosted, ...
```

3. Run the following commands to execute the Bash completion configuration for the `helm` command:

```
helm completion bash >> ~/.bash_completion
. /etc/profile.d/bash_completion.sh
. ~/.bash_completion
source <(helm completion bash)
```

Part 2: deploy an Apache web server with Helm Charts

Part 2.1: update Charts list in Helm

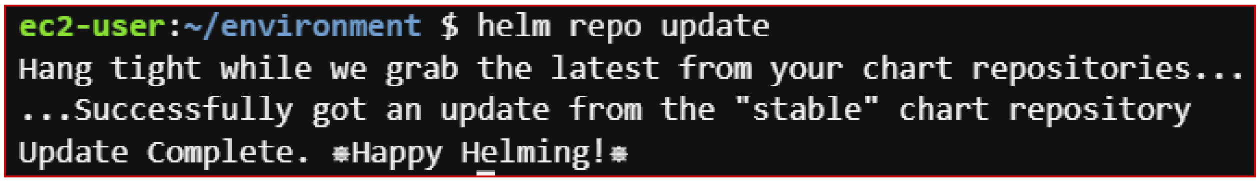
1. Charts often get updated, so it is important to run a repository update locally on your end to get the latest updated Charts list from time to time. Before running the update, run the following command to add the Helm default `stable` repository:

```
helm repo add stable https://charts.helm.sh/stable
```

2. Run the update:

```
helm repo update
```

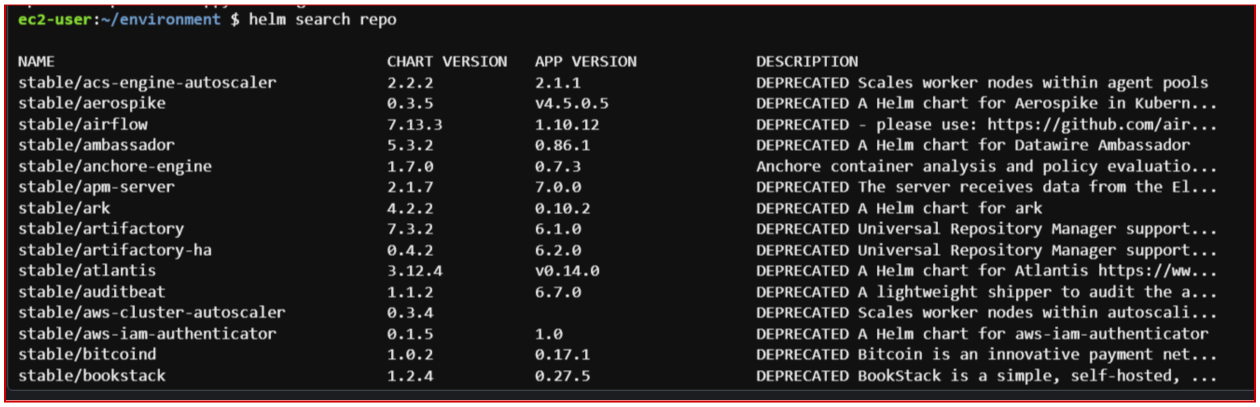
You should see the following result in your Cloud9 terminal:



```
ec2-user:~/environment $ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "stable" chart repository
Update Complete. ⎈Happy Helming!⎈
```

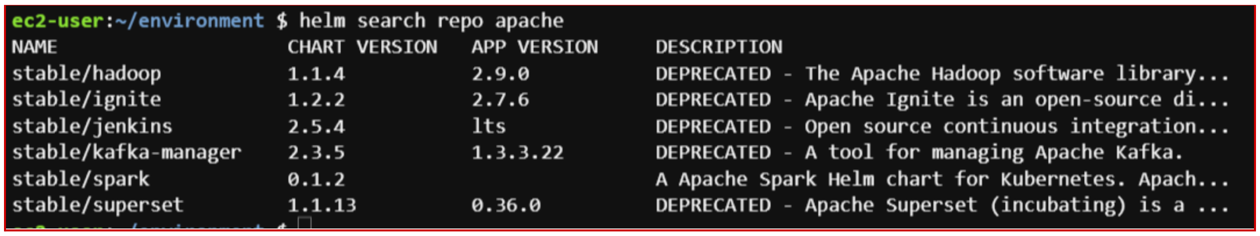
Part 2.2: browse and find Helm Chart repositories

1. Run `helm search repo` to output a list of all charts. The following example is just a part of the total output:



```
ec2-user:~/environment $ helm search repo

NAME                            CHART VERSION   APP VERSION     DESCRIPTION
stable/acs-engine-autoscaler    2.2.2           2.1.1           DEPRECATED Scales worker nodes within agent pools
stable/aerospike                0.3.5           v4.5.0.5        DEPRECATED A Helm chart for Aerospike in Kubern...
stable/airflow                  7.13.3          1.10.12         DEPRECATED - please use: https://github.com/air...
stable/ambassador               5.3.2           0.86.1          DEPRECATED A Helm chart for Datawire Ambassador
stable/anchore-engine           1.7.0           0.7.3           Anchore container analysis and policy evaluatio...
stable/apm-server               2.1.7           7.0.0           DEPRECATED The server receives data from the El...
stable/ark                      4.2.2           0.10.2          DEPRECATED A Helm chart for ark
stable/artifactory              7.3.2           6.1.0           DEPRECATED Universal Repository Manager support...
stable/artifactory-ha           0.4.2           6.2.0           DEPRECATED Universal Repository Manager support...
stable/atlantis                 3.12.4          v0.14.0         DEPRECATED A Helm chart for Atlantis https://ww...
stable/auditbeat                1.1.2           6.7.0           DEPRECATED A lightweight shipper to audit the a...
stable/aws-cluster-autoscaler   0.3.4                           DEPRECATED Scales worker nodes within autoscali...
stable/aws-iam-authenticator    0.1.5           1.0             DEPRECATED A Helm chart for aws-iam-authenticator
stable/bitcoind                 1.0.2           0.17.1          DEPRECATED Bitcoin is an innovative payment net...
stable/bookstack                1.2.4           0.27.5          DEPRECATED BookStack is a simple, self-hosted, ...
```

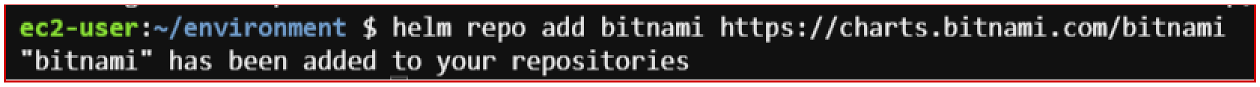
2. You can run a search with a specific keyword to narrow down your searches. For example, try running `helm search repo apache` to search for Apache-related charts:



```
ec2-user:~/environment $ helm search repo apache
NAME                CHART VERSION   APP VERSION     DESCRIPTION
stable/hadoop       1.1.4           2.9.0           DEPRECATED - The Apache Hadoop software library...
stable/ignite       1.2.2           2.7.6           DEPRECATED - Apache Ignite is an open-source di...
stable/jenkins      2.5.4           lts             DEPRECATED - Open source continuous integration...
stable/kafka-manager 2.3.5          1.3.3.22        DEPRECATED - A tool for managing Apache Kafka.
stable/spark        0.1.2                           A Apache Spark Helm chart for Kubernetes. Apach...
stable/superset     1.1.13          0.36.0          DEPRECATED - Apache Superset (incubating) is a ...
```

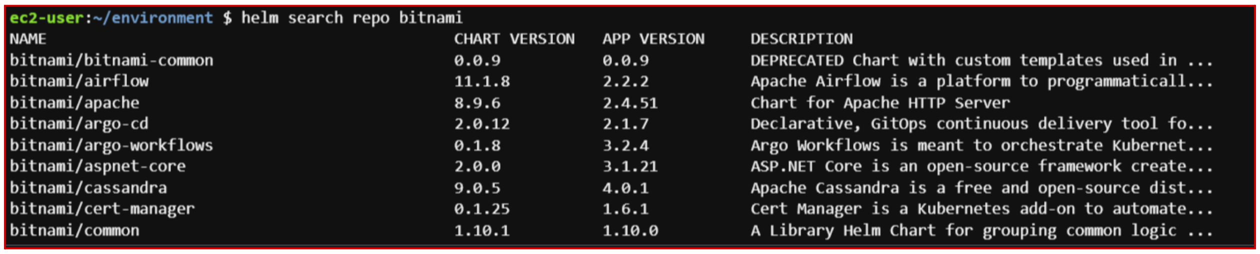
3. Note that a different repository, the Bitnami repository, has a standalone Apache HTTP server Chart: https://github.com/bitnami/charts/tree/master/bitnami/apache. The default `stable` Helm Chart repository does not have this. Thus, you should add the Bitnami Chart repository locally:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```



```
ec2-user:~/environment $ helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
```

4. You can now browse the list of Charts in the Bitnami repository. The following example is just a part of the total list output:



```
ec2-user:~/environment $ helm search repo bitnami
NAME                                            CHART VERSION   APP VERSION     DESCRIPTION
bitnami/bitnami-common                          0.0.9           0.0.9           DEPRECATED Chart with custom templates used in ...
bitnami/airflow                                 11.1.8          2.2.2           Apache Airflow is a platform to programmaticall...
bitnami/apache                                  8.9.6           2.4.51          Chart for Apache HTTP Server
bitnami/argo-cd                                 2.0.12          2.1.7           Declarative, GitOps continuous delivery tool fo...
bitnami/argo-workflows                          0.1.8           3.2.4           Argo Workflows is meant to orchestrate Kubernet...
bitnami/aspnet-core                             2.0.0           3.1.21          ASP.NET Core is an open-source framework create...
bitnami/cassandra                               9.0.5           4.0.1           Apache Cassandra is a free and open-source dist...
bitnami/cert-manager                            0.1.25          1.6.1           Cert Manager is a Kubernetes add-on to automate...
bitnami/common                                  1.10.1          1.10.0          A Library Helm Chart for grouping common logic ...
```

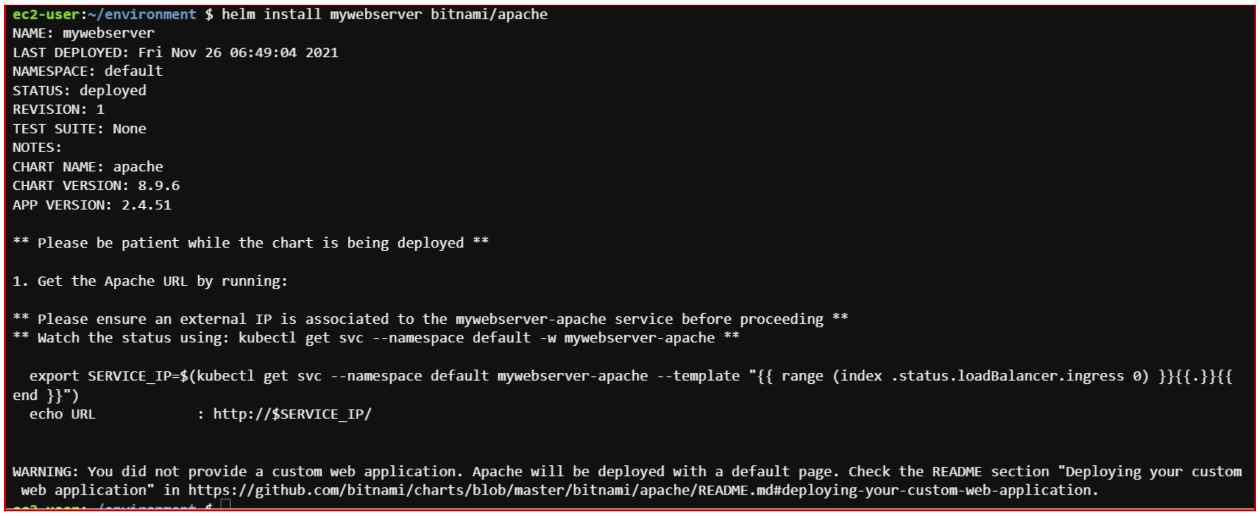
You can see that there is a Chart for the Apache HTTP server in the list.

Part 2.3: install the Apache server Chart to your EKS cluster using Helm

1. Run the following command to install and deploy the Bitnami Apache server Chart. The name used in the following command is `mywebserver`, but you can customize the name as you wish. Please note that you need to provide a unique name for the Chart you are installing. This is because you can install the same Chart multiple times in a Kubernetes cluster. A Chart can be installed multiple times as each Chart can be modified in different ways.

```
helm install mywebserver bitnami/apache
```

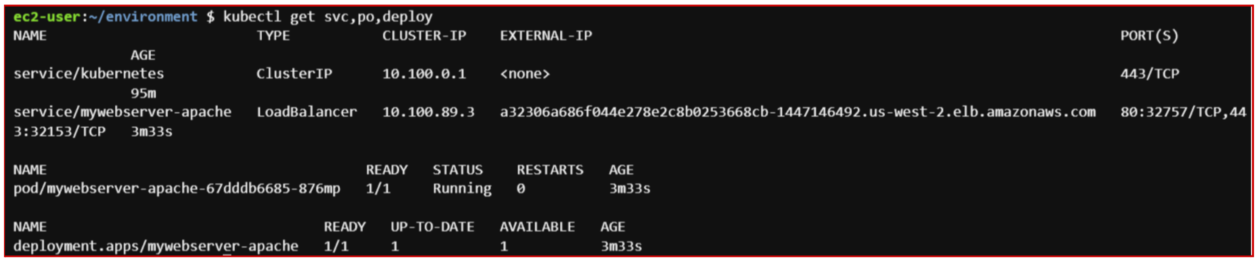
You should see a similar output to the one below that contains the deployment details:



```
ec2-user:~/environment $ helm install mywebserver bitnami/apache
NAME: mywebserver
LAST DEPLOYED: Fri Nov 26 06:49:04 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: apache
CHART VERSION: 8.9.6
APP VERSION: 2.4.51

** Please be patient while the chart is being deployed **

1. Get the Apache URL by running:

** Please ensure an external IP is associated to the mywebserver-apache service before proceeding **
** Watch the status using: kubectl get svc --namespace default -w mywebserver-apache **

  export SERVICE_IP=$(kubectl get svc --namespace default mywebserver-apache --template "{{ range (index .status.loadBalancer.ingress 0) }}{{.}}{{
end }}")
  echo URL            : http://$SERVICE_IP/


WARNING: You did not provide a custom web application. Apache will be deployed with a default page. Check the README section "Deploying your custom
 web application" in https://github.com/bitnami/charts/blob/master/bitnami/apache/README.md#deploying-your-custom-web-application.
```

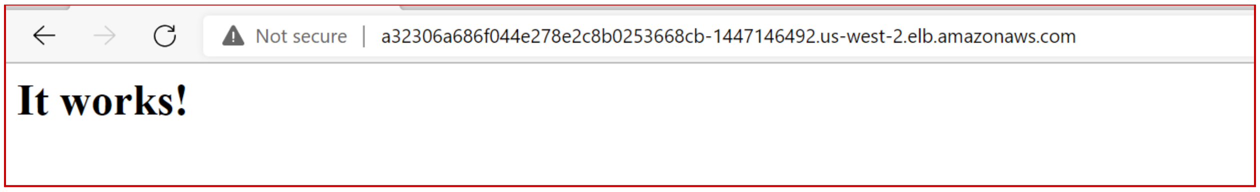
2. Run the following command to see the information for the Kubernetes Service (external IP address where you can see the web server's page), Pods (group of containers), and Deployments. These objects (Service, Pod, Deployment) are added by this Chart.

```
kubectl get svc,po,deploy
```



```
ec2-user:~/environment $ kubectl get svc,po,deploy
NAME                        TYPE           CLUSTER-IP     EXTERNAL-IP                                                                PORT(S)
                AGE
service/kubernetes          ClusterIP      10.100.0.1     <none>                                                                     443/TCP
                95m
service/mywebserver-apache  LoadBalancer   10.100.89.3    a32306a686f044e278e2c8b0253668cb-1447146492.us-west-2.elb.amazonaws.com   80:32757/TCP,44
3:32153/TCP    3m33s

NAME                                       READY   STATUS    RESTARTS   AGE
pod/mywebserver-apache-67dddb6685-876mp    1/1     Running   0          3m33s

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mywebserver-apache    1/1     1            1           3m33s
```

3. From the output above, you can see that there is a value under the "External-IP" column. Copy the external IP address from your own output and paste it in your browser to check if your Apache server is online. You may need to wait a few minutes before you see the webpage. You should see the default page:

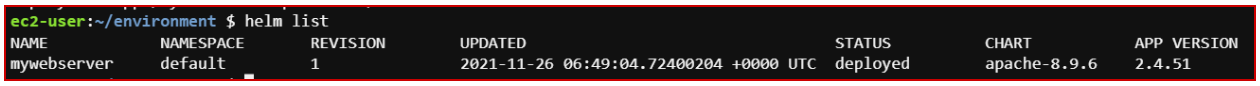


Not secure | a32306a686f044e278e2c8b0253668cb-1447146492.us-west-2.elb.amazonaws.com

It works!

Part 2.4: cleanup

1. Before removing the deployed web server, you can check what is currently running and being deployed by running the `helm list` command:

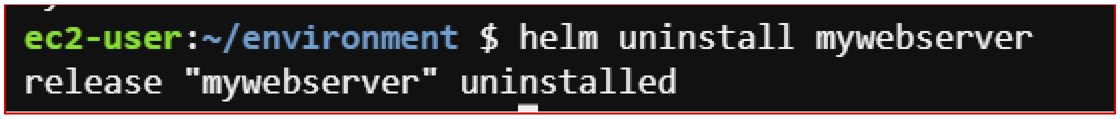


```
ec2-user:~/environment $ helm list
NAME          NAMESPACE   REVISION   UPDATED                                   STATUS     CHART          APP VERSION
mywebserver   default     1          2021-11-26 06:49:04.72400204 +0000 UTC    deployed   apache-8.9.6   2.4.51
```

2. To uninstall your deployed web server, run the following command. Replace `mywebserver` with your actual web server Chart's name:

```
helm uninstall mywebserver
```

You should see an output similar to the following:

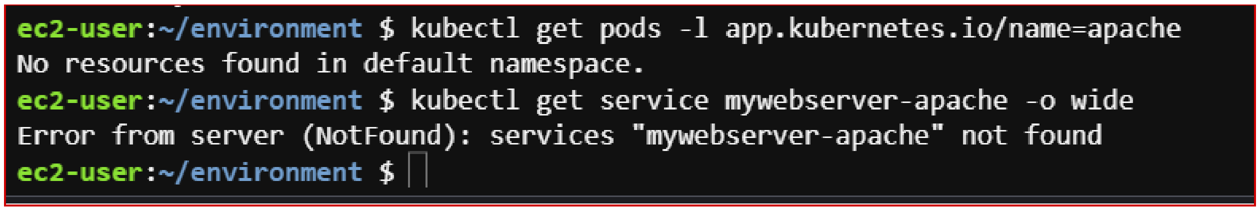


```
ec2-user:~/environment $ helm uninstall mywebserver
release "mywebserver" uninstalled
```

3. Recall that there were also Pod and Service objects that were created by the Chart. You can verify that the Pod and Service objects are now unavailable due to the uninstall by running the following commands:

```
kubectl get pods -l app.kubernetes.io/name=apache
kubectl get service mywebserver-apache -o wide
```

You should see similar output to the screenshot below:



```
ec2-user:~/environment $ kubectl get pods -l app.kubernetes.io/name=apache
No resources found in default namespace.
ec2-user:~/environment $ kubectl get service mywebserver-apache -o wide
Error from server (NotFound): services "mywebserver-apache" not found
ec2-user:~/environment $ 
```

4. You can also confirm your uninstallation was successful by running the external IP address that was outputted from an earlier step in this section in your browser. You should see that the web page is no longer available.

Part 3: deploy a static web application with Helm Charts

In this part, you will deploy a static web application using Bitnami's Apache Helm Chart. This will serve your application with Apache. An introduction to Bitnami's Apache Helm Chart was discussed in part 2 of this section.

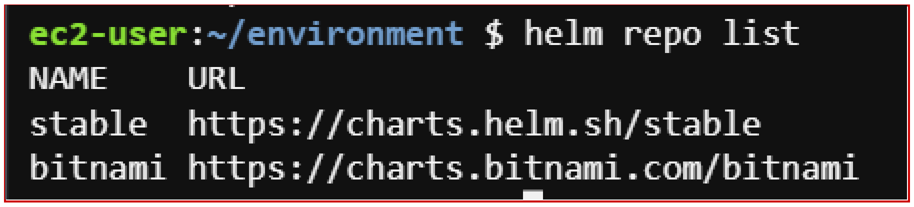
Using Bitnami's Apache Helm Chart, there is an efficient way to deploy a static website that involves cloning a Git repository.

Part 3.1: find a Git repository

1. Find a public Git repository that contains the static web content that you would like to deploy. The Git repository that will be used in this part as an example is the 2048 tile game: https://github.com/gabrielecirulli/2048.

Part 3.2: deploy from a Git repository using Bitnami's Apache Helm Chart

1. You have added the Bitnami Chart repository in part 2 of this section. To verify this, you can run `helm repo list` in your Cloud9 environment:



```
ec2-user:~/environment $ helm repo list
NAME    URL
stable  https://charts.helm.sh/stable
bitnami https://charts.bitnami.com/bitnami
```

If you do not have Bitnami in your list, add Bitnami using the following command:

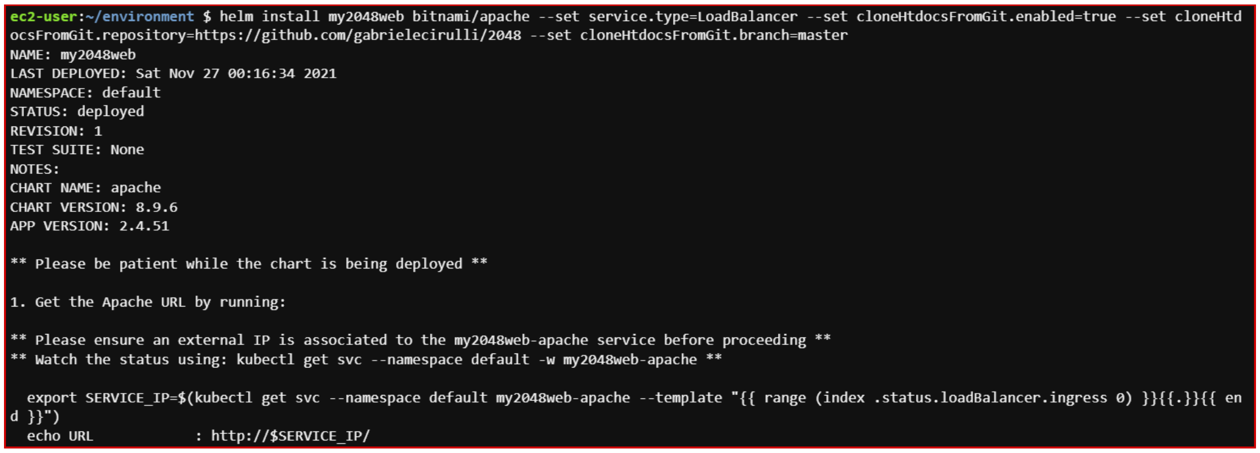
```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

2. To deploy the web application, you will use the same `helm install` command that was introduced in part 2. This time, you will add additional parameters that allow you to pass the Git repository URL and specify the branch you want to clone from. Here is an example template of the command:

```
helm install apache bitnami/apache --set service.type=LoadBalancer --set
cloneHtdocsFromGit.enabled=true --set
cloneHtdocsFromGit.repository=GIT-REPOSITORY-URL --set
cloneHtdocsFromGit.branch=GIT-REPOSITORY-BRANCH
```

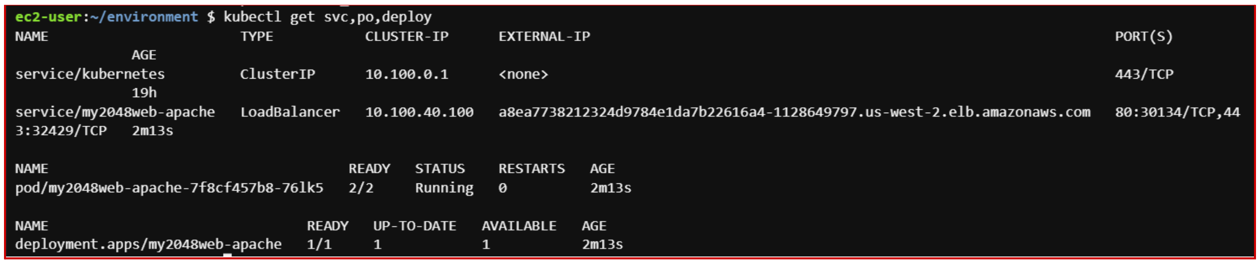
The following is the command with the Chart name, Git repository URL and Git repository branch fields updated with the 2048 web time game repository. For this example, the name that was added for the Chart is `my2048web`, but you may update it as you wish.

```
helm install my2048web bitnami/apache --set service.type=LoadBalancer --set
cloneHtdocsFromGit.enabled=true --set
cloneHtdocsFromGit.repository=https://github.com/gabrielecirulli/2048 --set
cloneHtdocsFromGit.branch=master
```



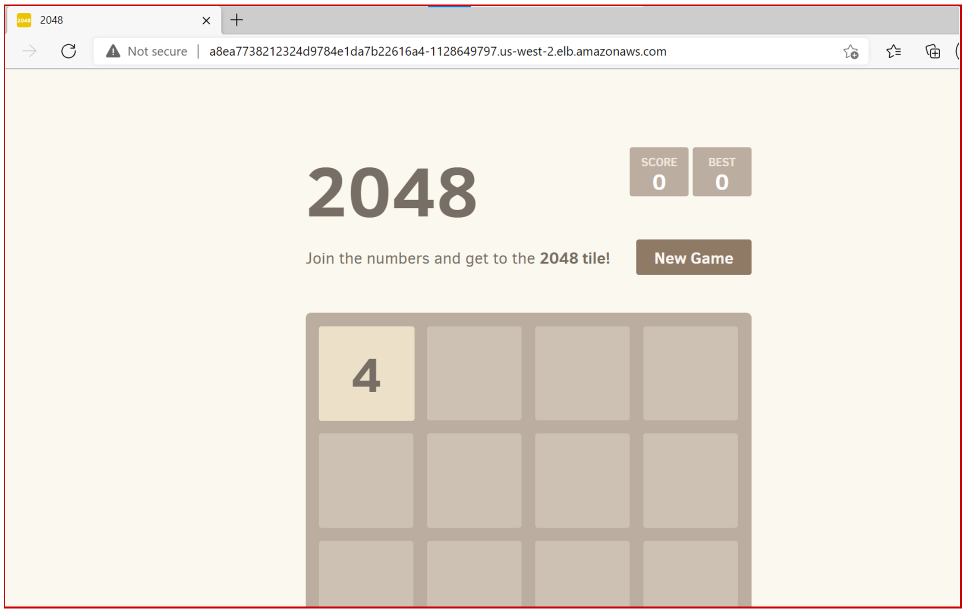
```
ec2-user:~/environment $ helm install my2048web bitnami/apache --set service.type=LoadBalancer --set cloneHtdocsFromGit.enabled=true --set cloneHtd
ocsFromGit.repository=https://github.com/gabrielecirulli/2048 --set cloneHtdocsFromGit.branch=master
NAME: my2048web
LAST DEPLOYED: Sat Nov 27 00:16:34 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: apache
CHART VERSION: 8.9.6
APP VERSION: 2.4.51

** Please be patient while the chart is being deployed **

1. Get the Apache URL by running:

** Please ensure an external IP is associated to the my2048web-apache service before proceeding **
** Watch the status using: kubectl get svc --namespace default -w my2048web-apache **

  export SERVICE_IP=$(kubectl get svc --namespace default my2048web-apache --template "{{ range (index .status.loadBalancer.ingress 0) }}{{.}}{{ en
d }}")
  echo URL            : http://$SERVICE_IP/
```

3. Run the `kubectl get svc,po,deploy` command to get the current status and external IP address of where the application is being deployed to:



```
ec2-user:~/environment $ kubectl get svc,po,deploy
NAME                         TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)
             AGE
service/kubernetes           ClusterIP      10.100.0.1      <none>                                                                     443/TCP
             19h
service/my2048web-apache     LoadBalancer   10.100.40.100   a8ea7738212324d9784e1da7b22616a4-1128649797.us-west-2.elb.amazonaws.com   80:30134/TCP,44
3:32429/TCP   2m13s

NAME                                  READY   STATUS    RESTARTS   AGE
pod/my2048web-apache-7f8cf457b8-76lk5   2/2     Running   0          2m13s

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my2048web-apache   1/1     1            1           2m13s
```

4. Enter the external IP address in your browser to check that your application is successfully deployed:


2048
Not secure | a8ea7738212324d9784e1da7b22616a4-1128649797.us-west-2.elb.amazonaws.com
2048
SCORE 0
BEST 0
Join the numbers and get to the 2048 tile!
New Game
4

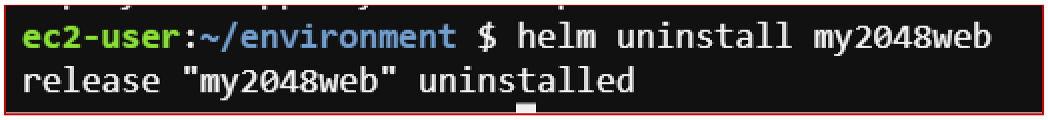
If you cannot access the page yet, please wait a few minutes and try again.

Part 3.3: cleanup

1. The cleanup for this part is the same as the cleanup for part 2. Run `helm uninstall` with the name of your Chart. The name in this example is `my2048web`:

```
helm uninstall my2048web
```

You should see the following output:



```
ec2-user:~/environment $ helm uninstall my2048web
release "my2048web" uninstalled
```

Clean up project resources

After finishing the project, make sure to delete/terminate your

- Cloud9 environment instance
- the rest of the EC2 instances that were created
- customer managed key in KMS
- CloudFormation stacks
- cluster in EKS
- pipeline in CodePipeline
- IAM roles

This cleanup can be done using your AWS Management Console.

References

Project introduction

https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
https://kubernetes.io/docs/concepts/overview/components/
https://helm.sh/docs/
https://www.densify.com/kubernetes-tools/helm-eks

Set up EKS and create a Kubernetes cluster in Cloud9

https://www.eksworkshop.com/030_eksctl/

Set up CodePipeline

https://www.eksworkshop.com/intermediate/220_codepipeline/
https://stackoverflow.com/questions/58758423/aws-ecs-codepipeline-build-error-repository-uri

Introduction to Helm

https://github.com/bitnami/charts/tree/master/bitnami/apache
https://www.eksworkshop.com/beginner/060_helm/
https://www.digitalocean.com/community/tutorials/an-introduction-to-helm-the-package-manager-for-kubernetes
https://docs.bitnami.com/tutorials/deploy-static-content-apache-kubernetes