WORKSHOP 1:

SETTING UP PIPELINES WITH EKS

CREATE A WORKSPACE

The Cloud9 workspace should be built by an IAM user with Administrator privileges, not the root account user. Please ensure you are logged in as an IAM user, not the root account user.

Ad blockers, javascript disablers, and tracking blockers should be disabled for the cloud9 domain, or connecting to the workspace might be impacted. Cloud9 requires third-party-cookies.

Launch Cloud9 in your closest region:

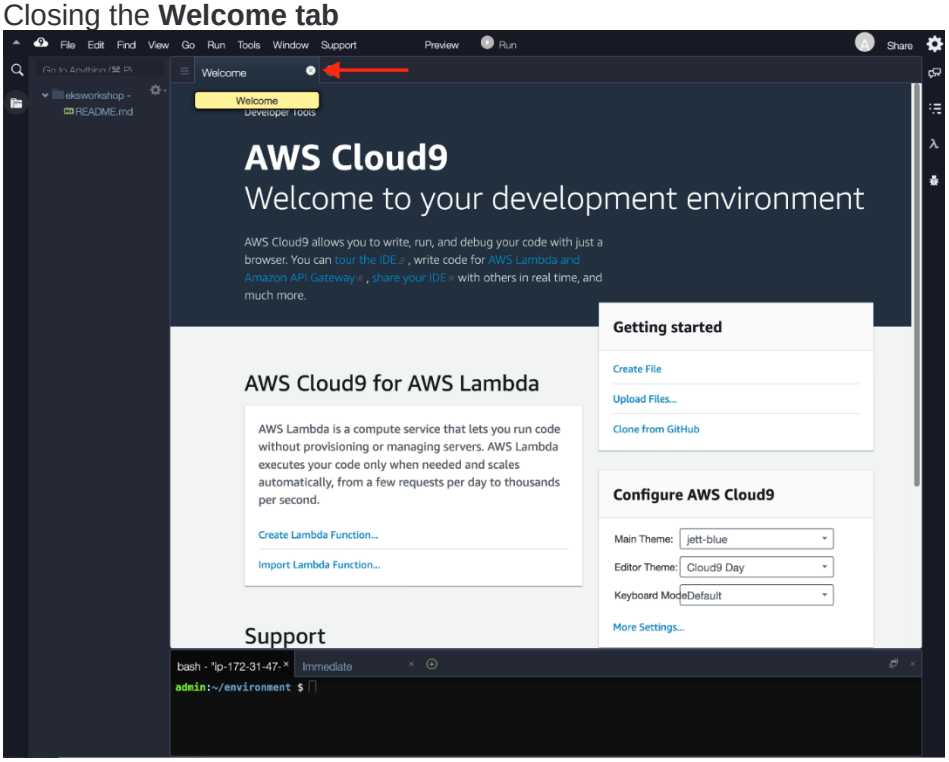
- Ireland
- Ohio
- Singapore

Create a Cloud9 Environment: https://us-east-2.console.aws.amazon.com/cloud9/home?region=us-east-2

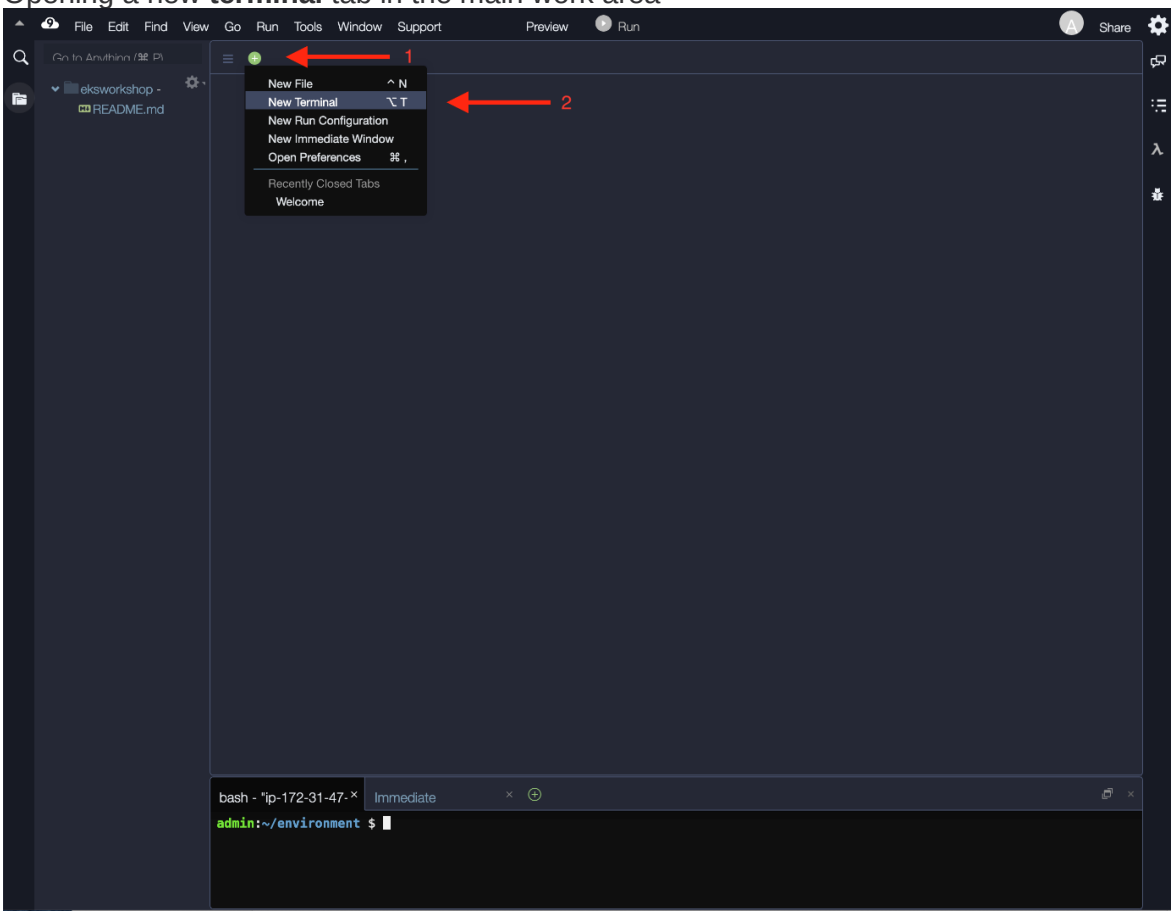
- Select **Create environment**
- Name it **eksworkshop**, click Next.
- Choose **t3.medium** for instance type, take all default values and click **Create environment**

When it comes up, customize the environment by:

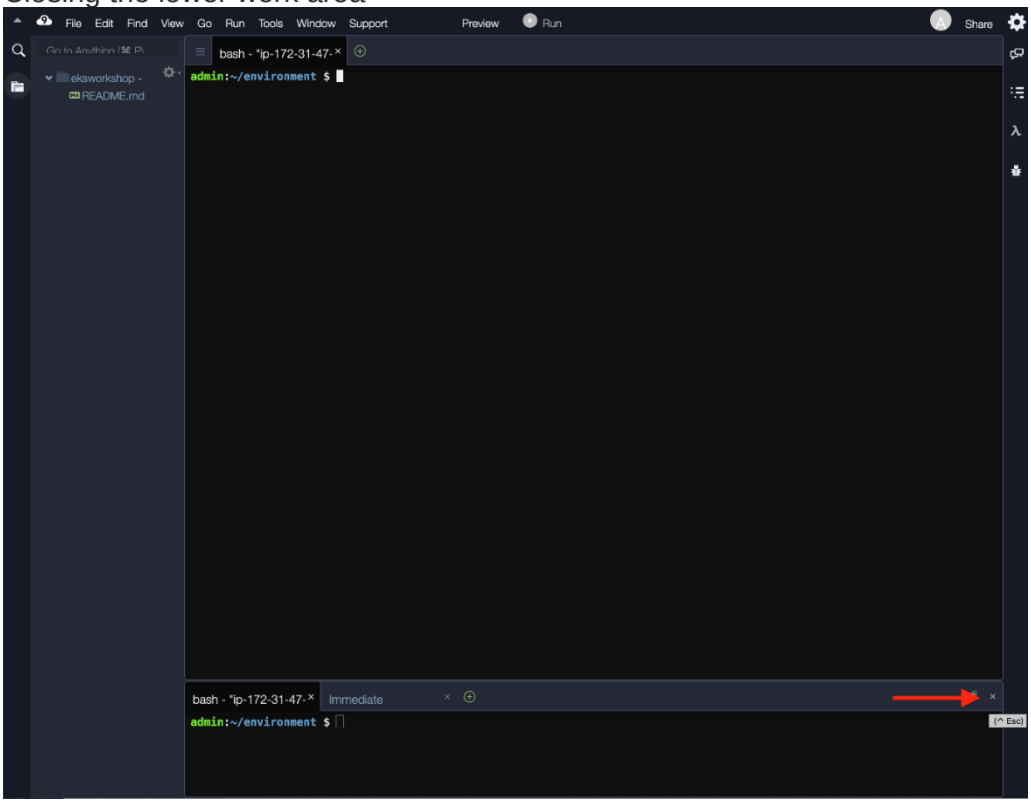
- Closing the **Welcome tab**



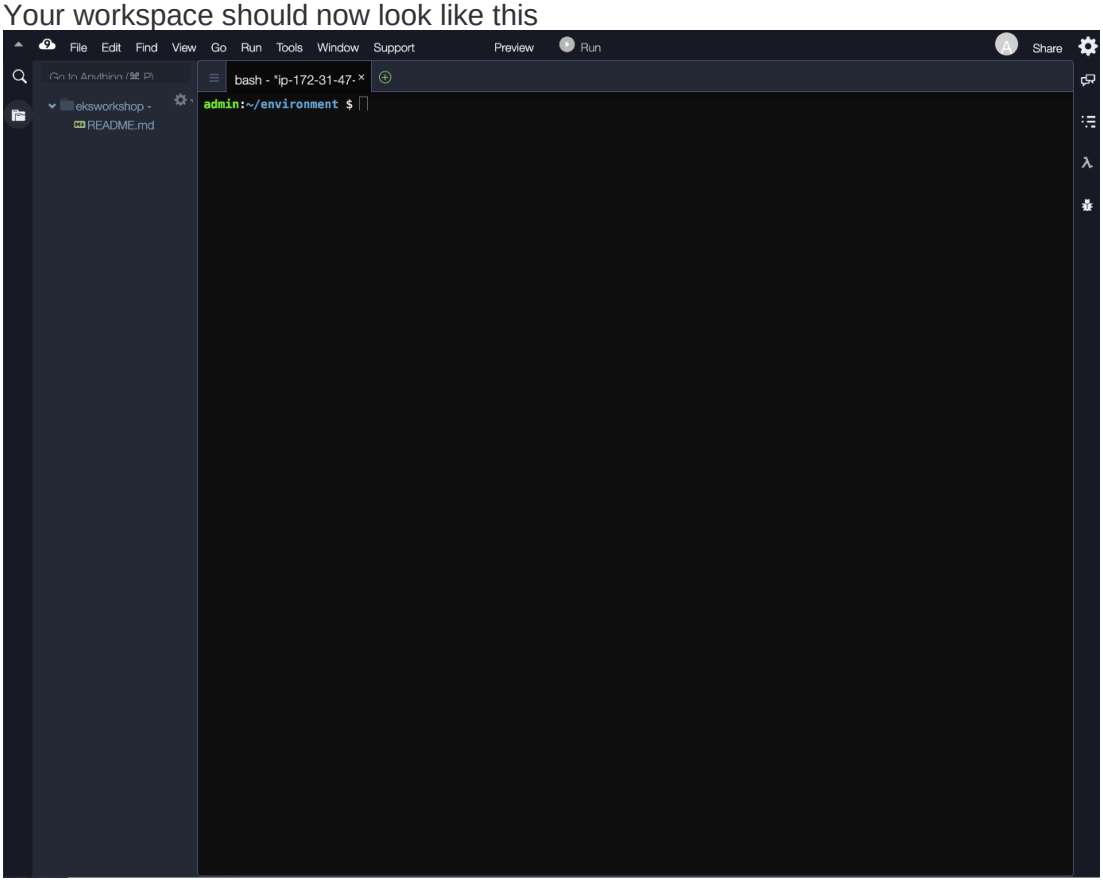
- Opening a new **terminal** tab in the main work area



- Closing the lower work area



- Your workspace should now look like this



If you intend to run all the sections s and tests.

Increase the disk size on the Cloud9 instance

The following command adds more disk space to the root volume of the EC2 instance that Cloud9 runs on. Once the command completes, we reboot the instance and it could take a minute or two for the IDE to come back online.

```
pip3 install --user --upgrade boto3
export instance_id=$(curl -s http://169.254.169.254/latest/meta-
data/instance-id)
python -c "import boto3
import os
from botocore.exceptions import ClientError
ec2 = boto3.client('ec2')
volume_info = ec2.describe_volumes(
    Filters=[
        {
            'Name': 'attachment.instance-id',
            'Values': [
                os.getenv('instance_id')
            ]
        }
    ]
)
volume_id = volume_info['Volumes'][0]['VolumeId']
try:
    resize = ec2.modify_volume(
            VolumeId=volume_id,
            Size=30
    )
    print(resize)
except ClientError as e:
    if e.response['Error']['Code'] == 'InvalidParameterValue':
        print('ERROR MESSAGE: {}'.format(e))"
if [ $? -eq 0 ]; then
    sudo reboot
fi
```

INSTALL KUBERNETES TOOLS

Amazon EKS clusters require kubectl and kubelet binaries and the aws-cli or aws-iam-authenticator binary to allow IAM authentication for your Kubernetes cluster.

In this workshop we will give you the commands to download the Linux binaries. If you are running Mac OSX / Windows, please see the official EKS docs for the download links.

Install kubectl

```
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
```

Update awscli

Upgrade AWS CLI according to guidance in AWS documentation.

```
sudo pip install --upgrade awscli && hash -r
```

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

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

Install yq for yaml processing

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

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
```

Enable kubectl bash_completion

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

```
. ~/.bash_completion
```

set the AWS Load Balancer Controller version

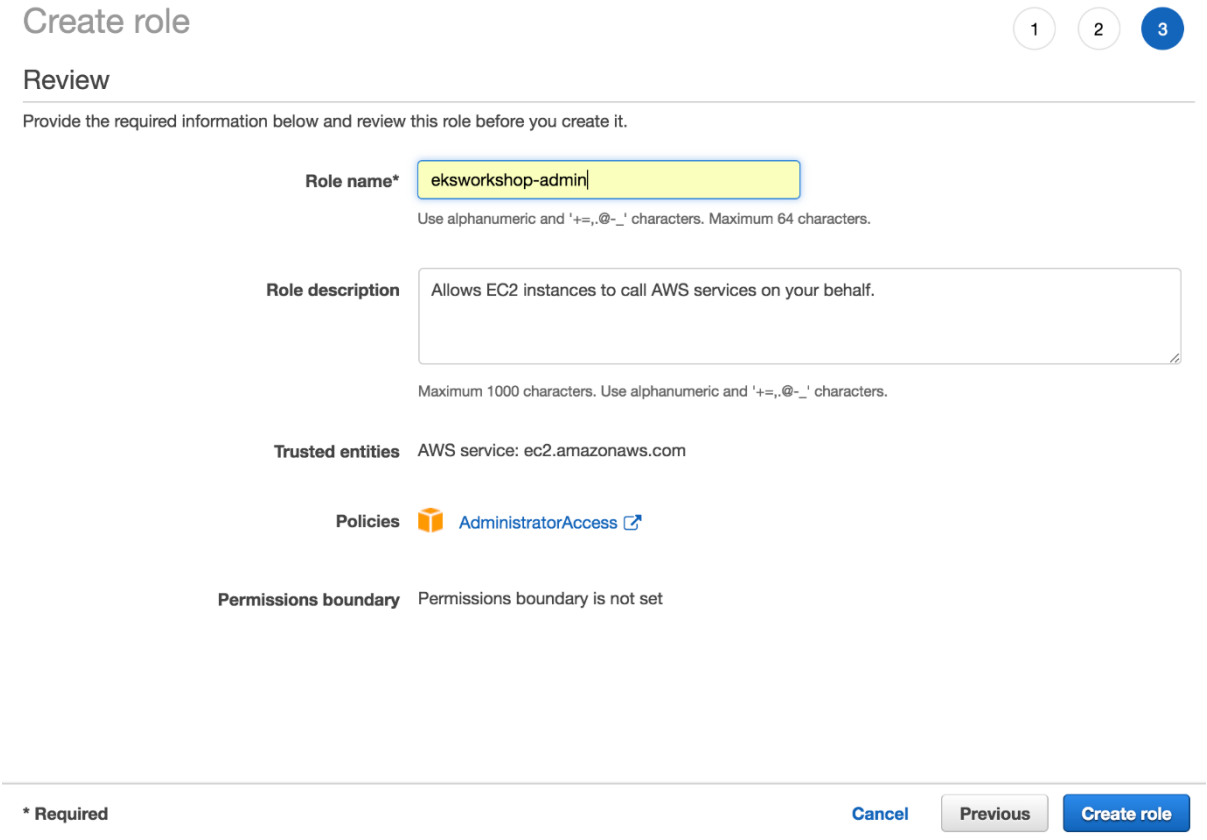
```
echo 'export LBC_VERSION="v2.2.0"' >>  ~/.bash_profile
.  ~/.bash_profile
```

- ------

CREATE AN IAM ROLE FOR YOUR WORKSPACE

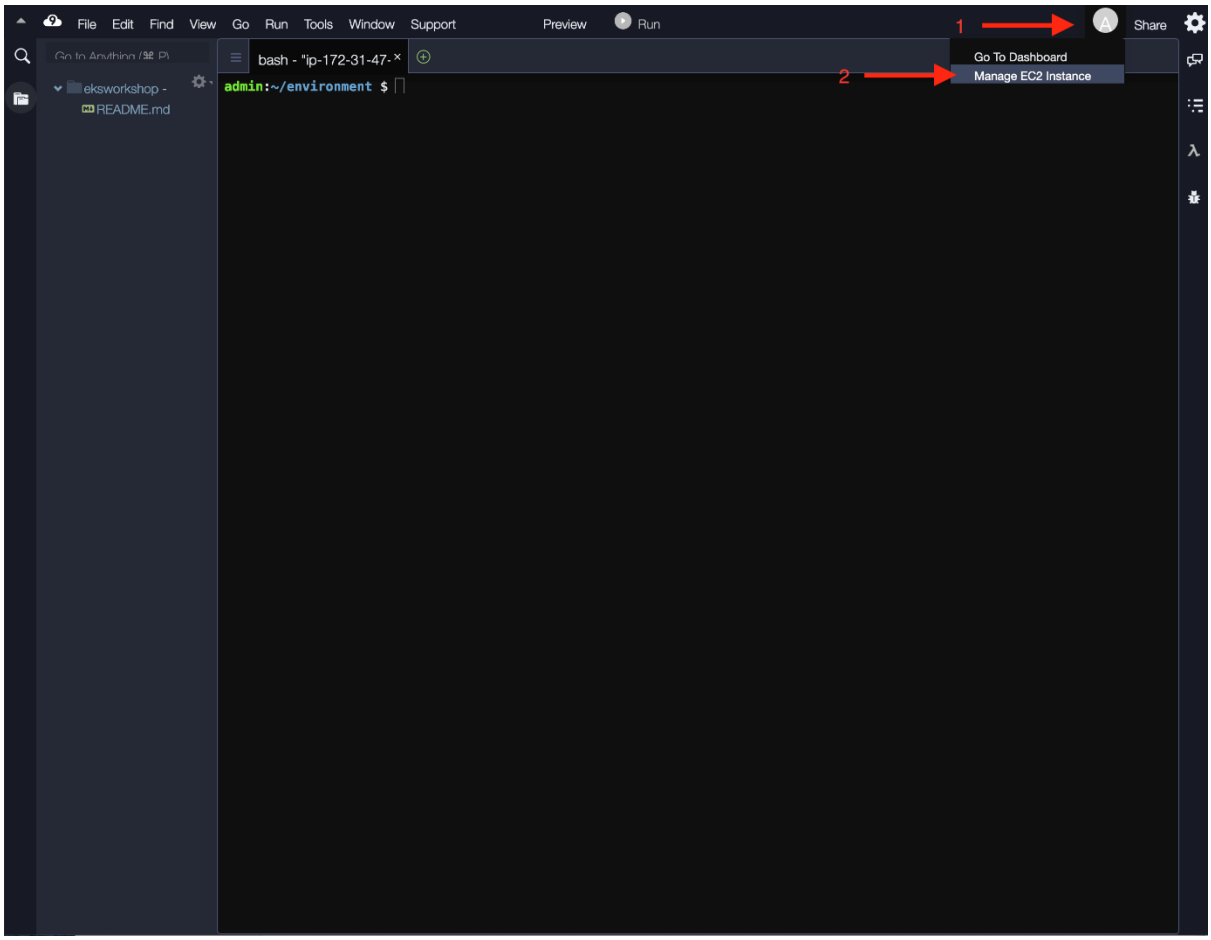
1. Follow this deep link to create an IAM role with Administrator access.
2. Confirm that **AWS service** and **EC2** are selected, then click **Next: Permissions** to view permissions.
3. Confirm that **AdministratorAccess** is checked, then click **Next: Tags** to assign tags.
4. Take the defaults, and click **Next: Review** to review.

Enter **eksworkshop-admin** for the Name, and click **Create role**.

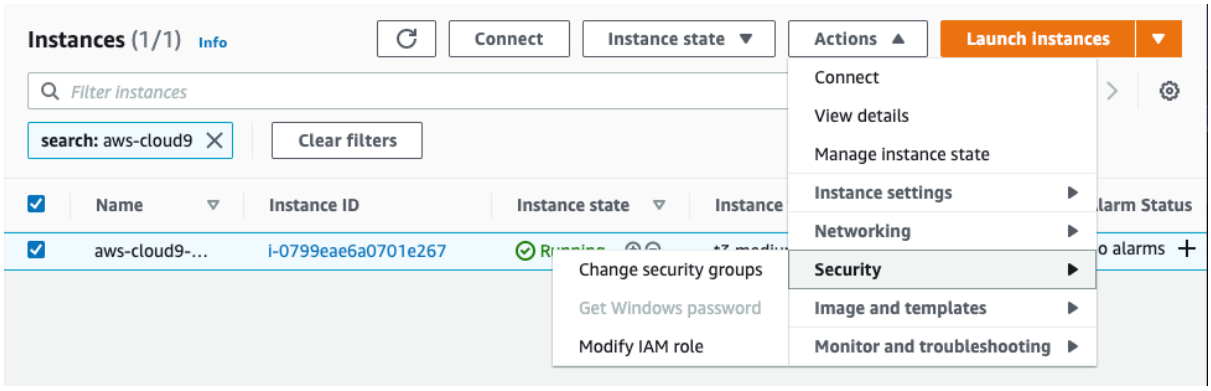


ATTACH THE IAM ROLE TO YOUR WORKSPACE

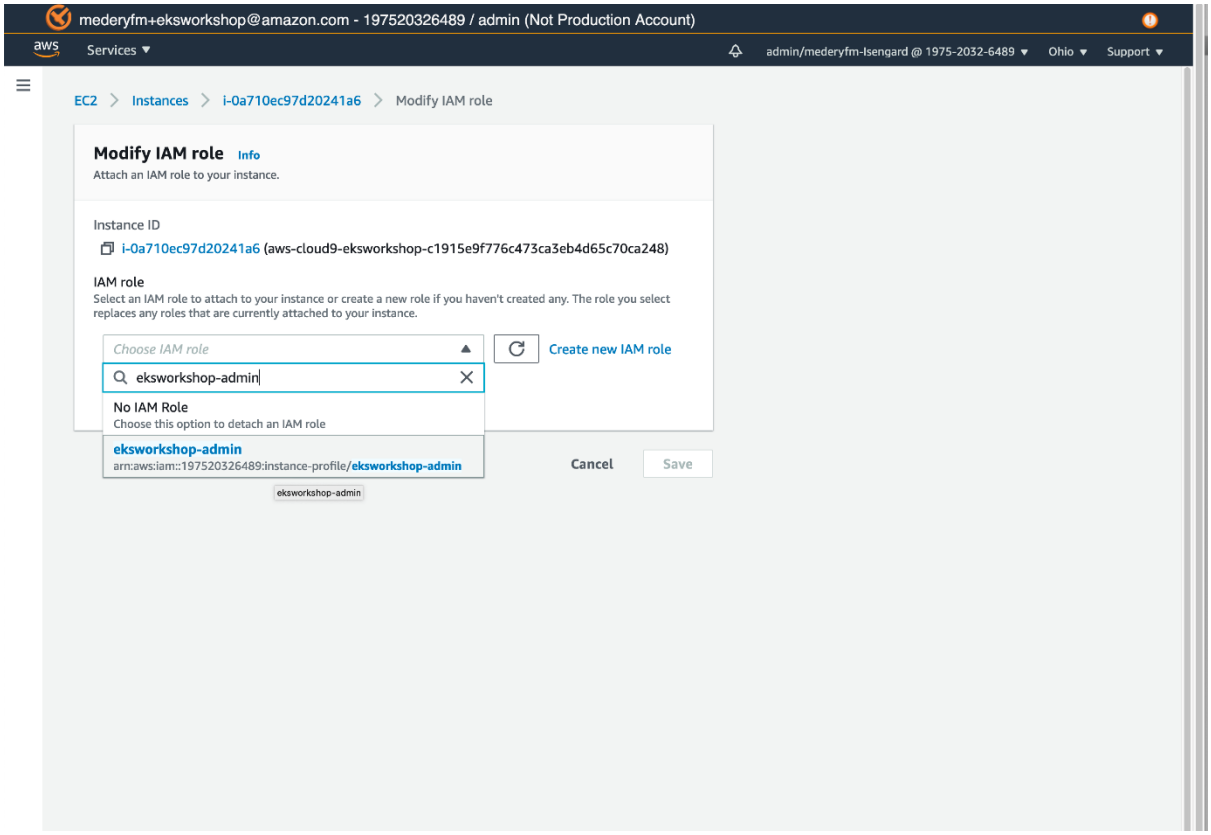
1. Click the grey circle button (in top right corner) and select **Manage EC2 Instance**



Select the instance, then choose **Actions / Security / Modify IAM Role**

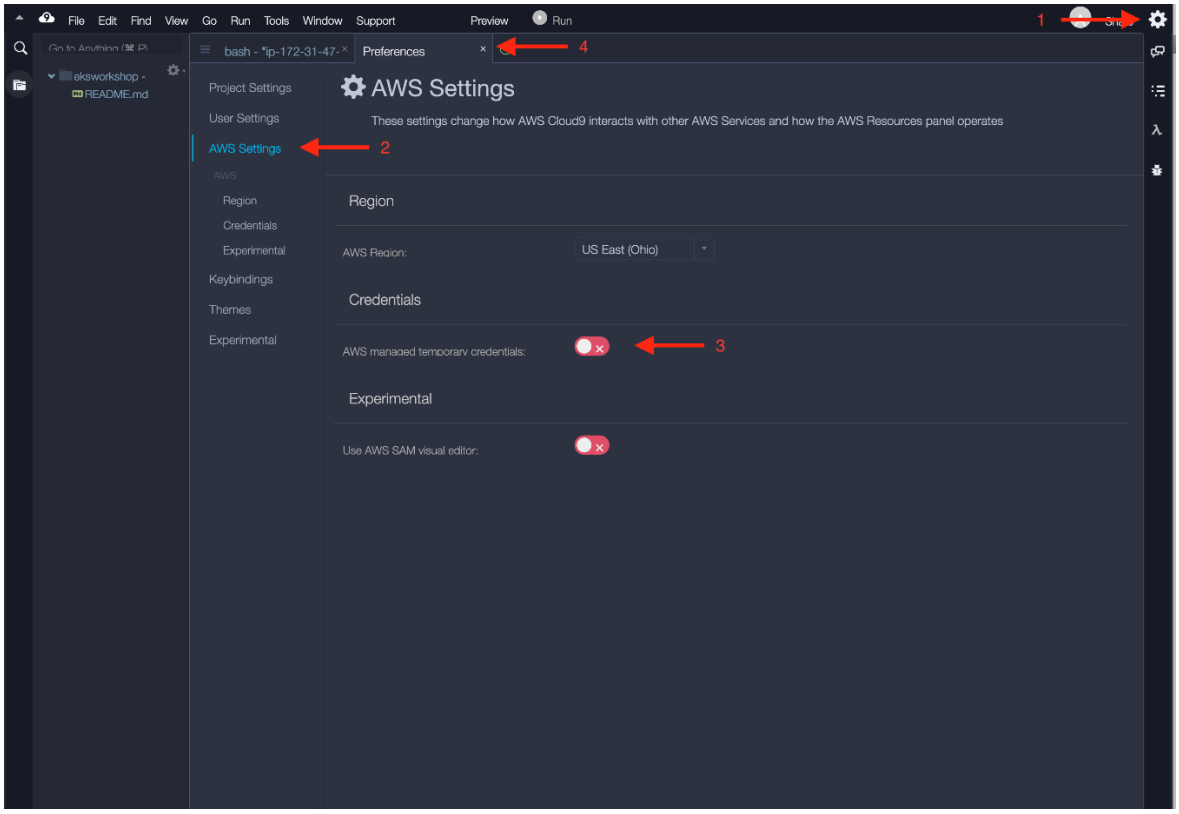


Choose **eksworkshop-admin** from the **IAM Role** drop down, and select **Save**



- Return to your Cloud9 workspace and click the gear icon (in top right corner)
- Select **AWS SETTINGS**
- Turn off **AWS managed temporary credentials**

- Close the Preferences tab



To ensure temporary credentials aren't already in place we will also remove any existing credentials file:

```
rm -vf ${HOME}/.aws/credentials
```

We should configure our aws cli with our current region as default.

If you are at an AWS event, ask your instructor which **AWS region** to use.

```
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))
```

Check if AWS_REGION is set to desired region

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

Let's save these into 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
```

Validate the IAM role

Use the GetCallerIdentity CLI command to validate that the Cloud9 IDE is using the correct IAM role.

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


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

If the IAM role is not valid, **DO NOT PROCEED**. Go back and confirm the steps on this page.

CLONE THE SERVICE REPOS

```
cd ~/environment
git clone https://github.com/brentley/ecsdemo-frontend.git
git clone https://github.com/brentley/ecsdemo-nodejs.git
git clone https://github.com/brentley/ecsdemo-crystal.git
```

CREATE AN AWS KMS CUSTOM MANAGED KEY (CMK)

- Create a CMK for the EKS cluster to use when encrypting your Kubernetes secrets:

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

- Let's retrieve the ARN of the CMK to input into the create cluster command.

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

- We set the MASTER_ARN environment variable to make it easier to refer to the KMS key later.

- Now, let's save the MASTER_ARN environment variable into the bash_profile

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

LAUNCH USING EKSCTL

PREREQUISITES

For this module, we need to download the eksctl binary:

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

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

Confirm the eksctl command works:

```
eksctl version
```

Enable eksctl bash-completion

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

LAUNCH EKS

DO NOT PROCEED with this step unless you have validated the IAM role in use by the Cloud9 IDE. You will not be able to run the necessary kubectl commands in the later modules unless the EKS cluster is built using the IAM role.

How do I check the IAM role on the workspace?

Expand here to see the solution

Run `aws sts get-caller-identity` and validate that your *Arn* contains `eksworkshop-admin` and an Instance Id.

```
{
```

```
    "Account": "123456789012",
    "UserId": "AROA1SAMPLEAWSIAMROLE:i-01234567890abcdef",
    "Arn": "arn:aws:sts::123456789012:assumed-role/eksworkshop-admin/i-
01234567890abcdef"
}
```

If you do not see the correct role, please go back and validate the IAM role for troubleshooting.

If you do see the correct role, proceed to next step to create an EKS cluster

Create an EKS cluster

`eksctl` version must be 0.38.0 or above to deploy EKS 1.19, click here to get the latest version.

Create an eksctl deployment file (eksworkshop.yaml) use in creating your cluster using the following syntax:

```
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
```

Next, use the file you created as the input for the eksctl cluster creation.

We are deliberatly launching at least one Kubernetes version behind the latest available on Amazon EKS. This allows you to perform the cluster upgrade lab.

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

Launching EKS and all the dependencies will take approximately 15 minutes

Take approximately 15 minutes

TEST THE CLUSTER

Test the cluster:

Confirm your nodes:

```
kubectl get nodes # if we see our 3 nodes, we know we have
authenticated correctly
```

Export the Worker Role Name for use throughout the workshop:

```
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
```

Congratulations!

You now have a fully working Amazon EKS Cluster that is ready to use! Before you move on to any other labs, make sure to complete the steps on the next page to update the EKS Console Credentials.

CONSOLE CREDENTIALS

- This step is optional, as nearly all of the workshop content is CLI-driven. But, if you'd like full access to your workshop cluster in the EKS console this step is recommended.
- The EKS console allows you to see not only the configuration aspects of your cluster, but also to view Kubernetes cluster objects such as Deployments, Pods, and Nodes. For this type of access, the console IAM User or Role needs to be granted permission within the cluster.
- By default, the credentials used to create the cluster are automatically granted these permissions. Following along in the workshop, you've created a cluster using temporary IAM credentials from within Cloud9. This means that you'll need to add your AWS Console credentials to the cluster.
- Import your EKS Console credentials to your new cluster:
- IAM Users and Roles are bound to an EKS Kubernetes cluster via a ConfigMap named `aws-auth`. We can use `eksctl` to do this with one command.
- You'll need to determine the correct credential to add for your AWS Console access. If you know this already, you can skip ahead to the `eksctl create iamidentitymapping` step below.
- If you've built your cluster from Cloud9 as part of this tutorial, invoke the following within your environment to determine your IAM Role or User ARN.

```
c9builder=$(aws cloud9 describe-environment-memberships --
environment-id=$C9_PID | jq -r '.memberships[].userArn')
if echo ${c9builder} | grep -q user; then
rolearn=${c9builder}
    echo Role ARN: ${rolearn}
elif echo ${c9builder} | grep -q assumed-role; then
assumedrolename=$(echo ${c9builder} | awk -F/ '{print $(NF-1)}')
rolearn=$(aws iam get-role --role-name ${assumedrolename} --query
Role.Arn --output text)
echo Role ARN: ${rolearn}
fi
```

With your ARN in hand, you can issue the command to create the identity mapping within the cluster.

```
eksctl create iamidentitymapping --cluster eksworkshop-eksctl --
arn ${rolearn} --group system:masters --username admin
```

Note that permissions can be restricted and granular but as this is a workshop cluster, you're adding your console credentials as administrator.

Now you can verify your entry in the AWS auth map within the console.

```
kubectl describe configmap -n kube-system aws-auth
```

Now you're all set to move on. For more information, check out the EKS documentation on this topic.

DEPLOY THE KUBERNETES DASHBOARD

DEPLOY THE OFFICIAL KUBERNETES DASHBOARD

The official Kubernetes dashboard is not deployed by default, but there are instructions in the official documentation

We can deploy the dashboard with the following command:

```
export DASHBOARD_VERSION="v2.0.0"

kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/${DASHBOARD_VERS
ION}/aio/deploy/recommended.yaml
```

Since this is deployed to our private cluster, we need to access it via a proxy. `kube-proxy` is available to proxy our requests to the dashboard service. In your workspace, run the following command:

```
kubectl proxy --port=8080 --address=0.0.0.0 --disable-filter=true &
```

This will start the proxy, listen on port 8080, listen on all interfaces, and will disable the filtering of non-localhost requests.

This command will continue to run in the background of the current terminal's session.

We are disabling request filtering, a security feature that guards against XSRF attacks. This isn't recommended for a production environment, but is useful for our dev environment.

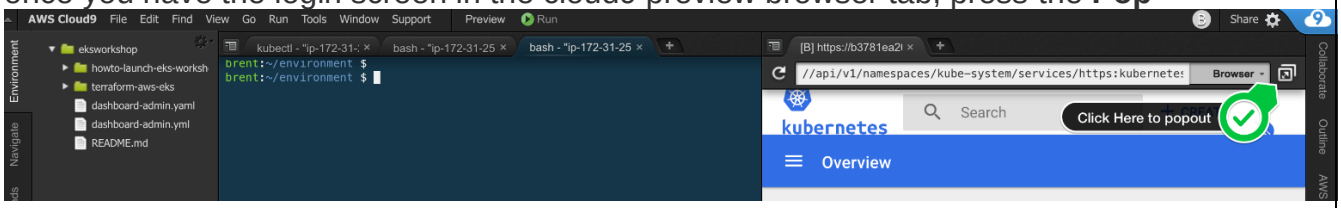
ACCESS THE DASHBOARD

Now we can access the Kubernetes Dashboard

1. In your Cloud9 environment, click **Tools / Preview / Preview Running Application**
2. Scroll to **the end of the URL** and append:

```
/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-
dashboard:/proxy/
```

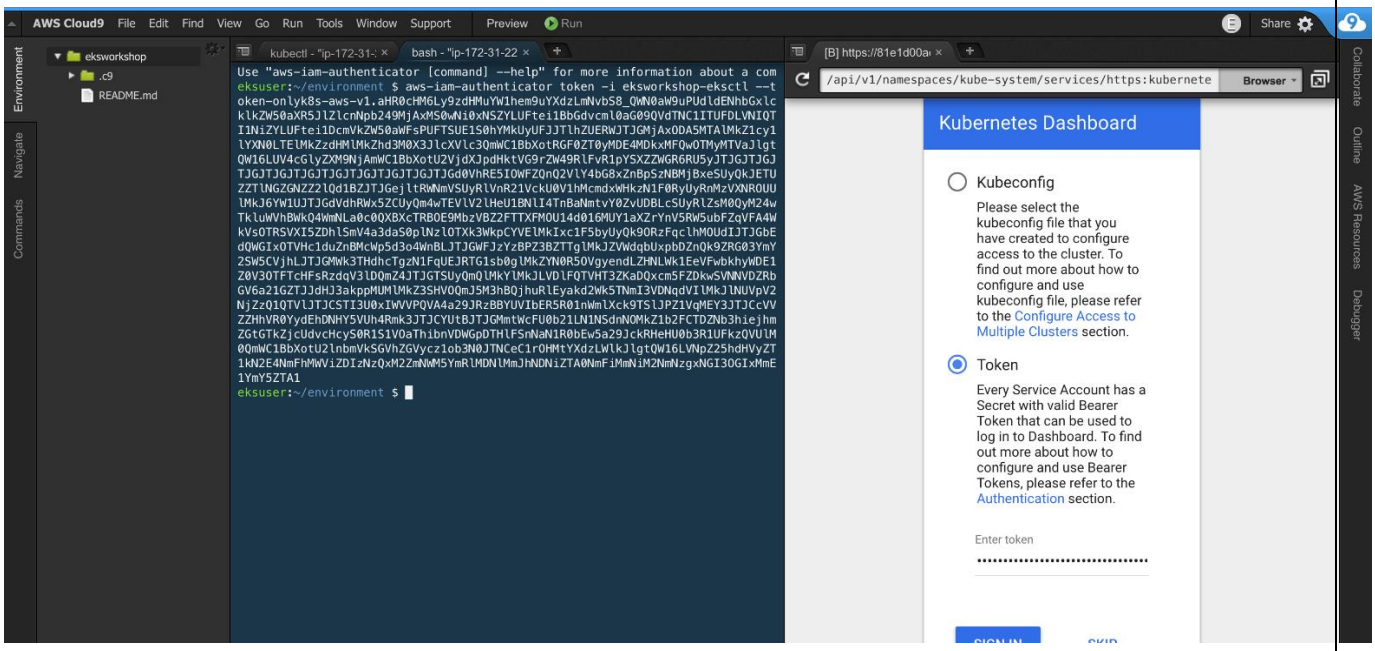
The Cloud9 Preview browser doesn't appear to support the token authentication, so once you have the login screen in the cloud9 preview browser tab, press the **Pop**



Open a New Terminal Tab and enter

```
aws eks get-token --cluster-name eksworkshop-eksctl | jq -r
'.status.token'
```

Copy the output of this command and then click the radio button next to *Token* then in the text field below paste the output from the last command.



Then press *Sign In*.

(Do following steps in case you don't want to proceed with Workshop 2.)

CLEANUP

Stop the proxy and delete the dashboard deployment

```
# kill proxy
pkill -f 'kubectl proxy --port=8080'

# delete dashboard
kubectl delete -f
https://raw.githubusercontent.com/kubernetes/dashboard/${DASHBOARD_VERS
ION}/aio/deploy/recommended.yaml

unset DASHBOARD_VERSION
```