Deploy the sample app to EKS using CICD

In this chapter you will deploy a sample application using CodeCommit, CodePipeline & CodeBuild

```
cd ~/environment/tfekscode/sampleapp
```

Be sure you have run the `terraform destroy -auto-approve` command in the previous step before proceeding with the steps below.

Before we start check CodeBuild is authorized to access the EKS cluster ok

```
kubectl get -n kube-system configmap/aws-auth -o yaml | grep -i codebuild
    - rolearn: arn:aws:iam:xxxxxxxxxxxx:role/codebuild-eks-cicd-build-app-service-role
```

If you don't see a line of output similar to the line above - run this command:

```
~/environment/tfekscode/nodeg/auth-cicd.sh
```

Create a service credential to use with our CodeCommit git repo:

```
usercred=$(aws iam create-service-specific-credential --user-name git-user --service-name codecommit.amazonaws.com)
GIT_USERNAME=$(echo $usercred | jq -r '.ServiceSpecificCredential.ServiceUserName')
GIT_PASSWORD=$(echo $usercred | jq -r '.ServiceSpecificCredential.ServicePassword')
CREDENTIAL_ID=$(echo $usercred| jq -r '.ServiceSpecificCredential.ServiceSpecificCredentialId')
test -n "$GIT_USERNAME" && echo GIT_USERNAME is "$GIT_USERNAME" || "echo GIT_USERNAME is not set"
```

Clone the (empty) repo:

```
test -n "$AWS_REGION" && echo AWS_REGION is "$AWS_REGION" || "echo AWS_REGION is not set"
wsid=$(aws ssm get-parameter --name /workshop/tf-eks/id --query Parameter.Value --output text)
git clone codecommit::$AWS_REGION://eksworkshop-app-${wsid}
Cloning into 'eksworkshop-app-xxxxxxxxxxxxxxxx'...

'Namespace' object has no attribute 'cli_binary_format'
warning: You appear to have cloned an empty repository.
```

Populate with our source files - including the special file **buildspec.yaml** which has the steps CodeBuild will follow.

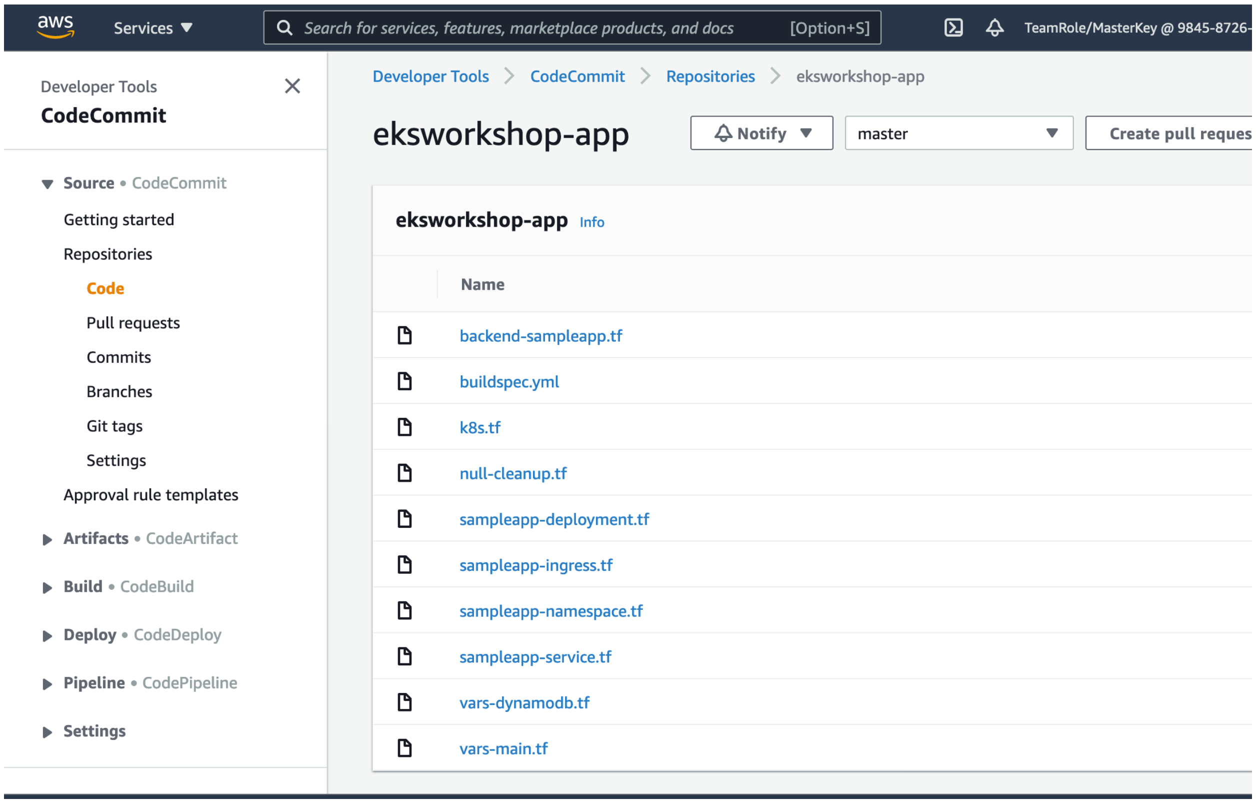
```
cd eksworkshop-app-${wsid}
cp ../buildspec.yml .
cp ../*.tf .
cp ../cleanup.sh .
cp ../../tf-setup/generated/backend-k8scicd.tf backend-sampleapp.tf
```

Add files to git, commit and push

```
git add --all
git commit -m "Initial commit."
git push
```

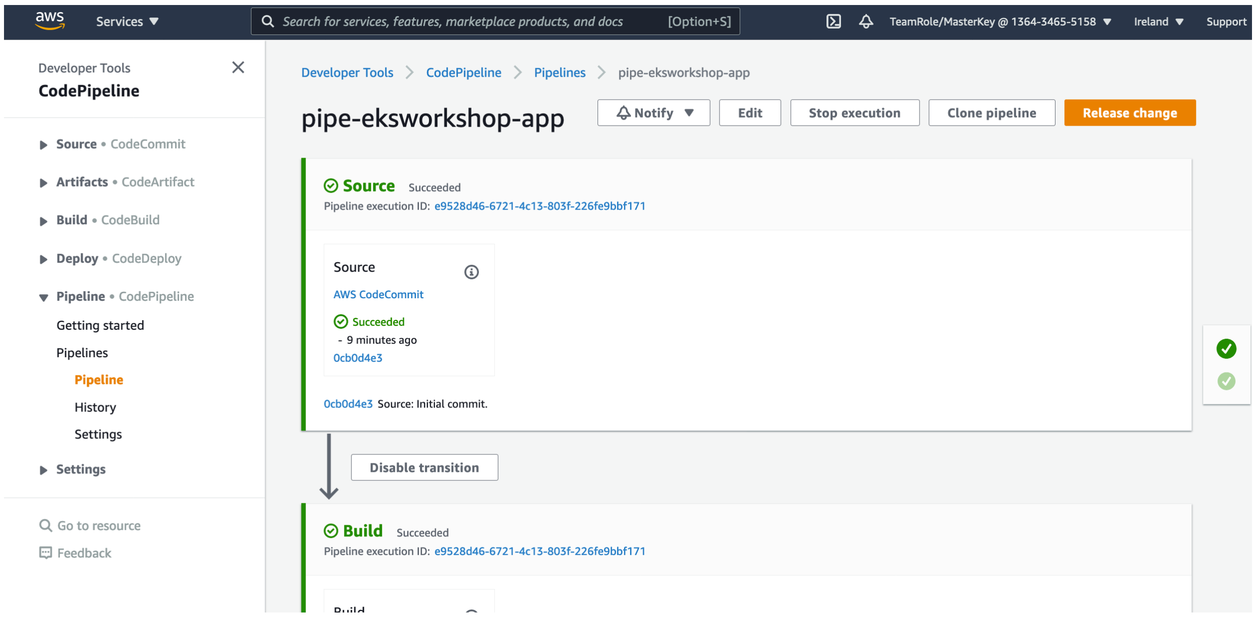
This should now trigger a few activities

Check you can see your code in CodeCommit - navigate to your repository in the console and confirm you can see the files:



Name
backend-sampleapp.tf
buildspec.yml
k8s.tf
null-cleanup.tf
sampleapp-deployment.tf
sampleapp-ingress.tf
sampleapp-namespace.tf
sampleapp-service.tf
vars-dynamodb.tf
vars-main.tf

Next check if the CodePipeline is running - navigate to it in the console



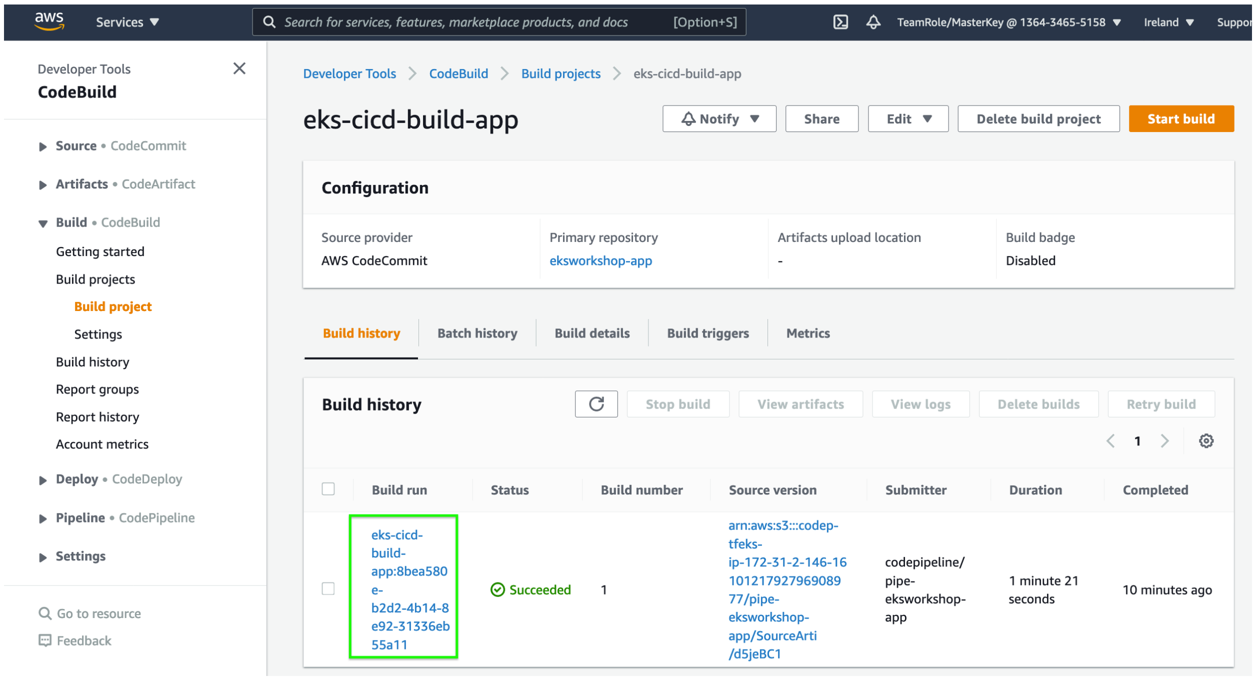
Source Succeeded
Pipeline execution ID: e9528d46-6721-4c13-803f-226fe9bbf171

Source
AWS CodeCommit
Succeeded
- 9 minutes ago
0cb0d4e3

0cb0d4e3 Source: Initial commit.

Build Succeeded
Pipeline execution ID: e9528d46-6721-4c13-803f-226fe9bbf171

You can also link through to the CodeBuild project:



Developer Tools > CodeBuild > Build projects > eks-cicd-build-app

eks-cicd-build-app

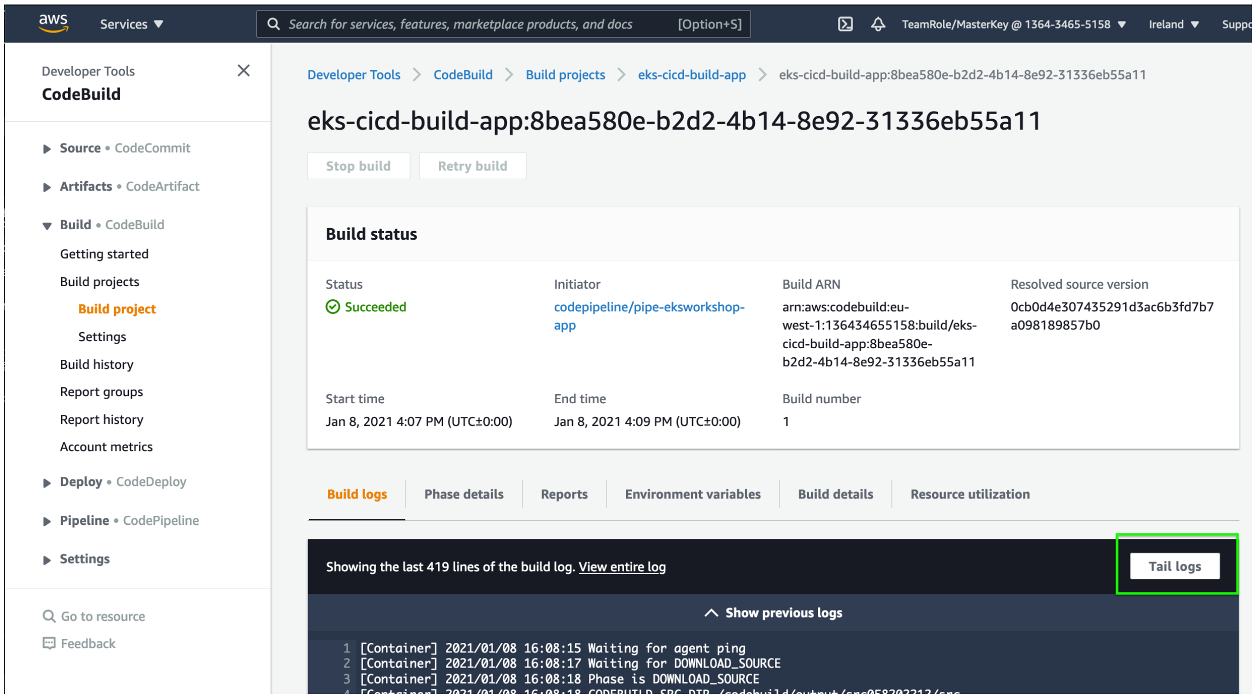
Configuration

Source provider	Primary repository	Artifacts upload location	Build badge
AWS CodeCommit	eksworkshop-app	-	Disabled

Build history

Build run	Status	Build number	Source version	Submitter	Duration	Completed
eks-cicd-build-app:8bea580e-b2d2-4b14-8e92-31336eb55a11	Succeeded	1	arn:aws:s3:::codep-tfeks-ip-172-31-2-146-16101217927969089 77/pipe-eksworkshop-app/SourceArti /d5jeBC1	codepipeline/pipe-eksworkshop-app	1 minute 21 seconds	10 minutes ago

And tail to logs of the build job (scroll the window or use the `Tail Logs` button):



Developer Tools > CodeBuild > Build projects > eks-cicd-build-app > eks-cicd-build-app:8bea580e-b2d2-4b14-8e92-31336eb55a11

eks-cicd-build-app:8bea580e-b2d2-4b14-8e92-31336eb55a11

Build status

Status	Initiator	Build ARN	Resolved source version
Succeeded	codepipeline/pipe-eksworkshop-app	arn:aws:codebuild:eu-west-1:136434655158:build/eks-cicd-build-app:8bea580e-b2d2-4b14-8e92-31336eb55a11	0cb0d4e307435291d3ac6b3fd7b7a098189857b0

Start time	End time	Build number
Jan 8, 2021 4:07 PM (UTC±0:00)	Jan 8, 2021 4:09 PM (UTC±0:00)	1

Showing the last 419 lines of the build log. View entire log

```
[Container] 2021/01/08 16:08:15 Waiting for agent ping
[Container] 2021/01/08 16:08:17 Waiting for DOWNLOAD_SOURCE
[Container] 2021/01/08 16:08:18 Phase is DOWNLOAD_SOURCE
```

Check everything is running ?

```
kubectl get pods,svc,deployment -n game-2048 -o wide

NAME                        READY   STATUS    RESTARTS   AGE   IP            NODE
NOMINATED NODE   READINESS GATES

pod/deployment-2048-76d4bff958-5w94k   1/1     Running   0          55s   100.64.143.56   ip-10-0-3-166.eu-
west-1.compute.internal   <none>           <none>

pod/deployment-2048-76d4bff958-r4jhb   1/1     Running   0          55s   100.64.24.3     ip-10-0-1-228.eu-west-
1.compute.internal   <none>           <none>


NAME                   TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE   SELECTOR

service/service-2048   NodePort   172.20.162.86   <none>        80:32624/TCP   20s
app.kubernetes.io/name=app-2048


NAME                              READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES
SELECTOR

deployment.apps/deployment-2048   2/2     2            2           55s   app-2048     123456789012.dkr.ecr.eu-west-
1.amazonaws.com/sample-app   app.kubernetes.io/name=app-2048
```

Note that:

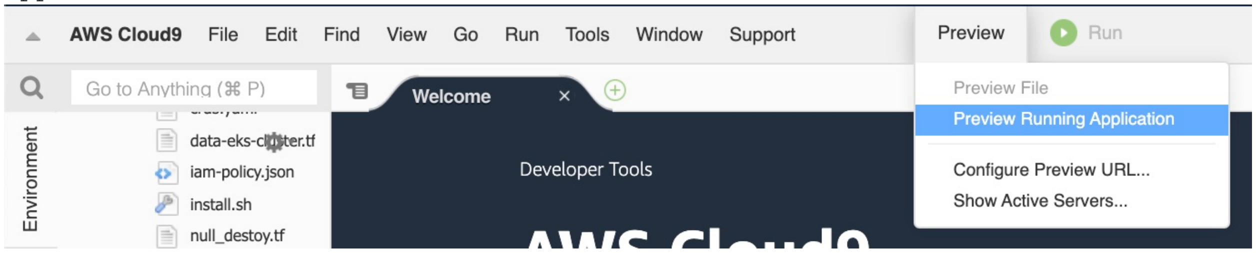
- The pods are deployed to a 100.64.x.x. address.
- The service is exposing port 80.
- The deployment is referencing a private ECR repository belonging to your account. (see the IMAGES section of the deployment output)

Enable port forwarding so we can see the application in our Cloud9 IDE

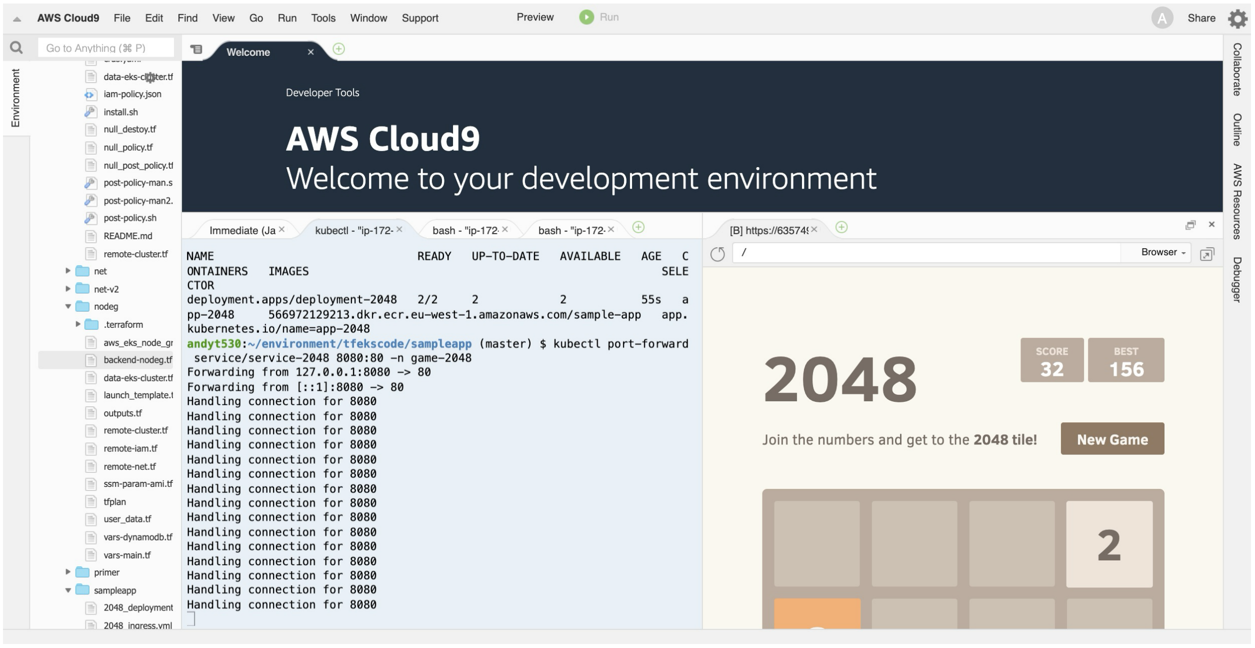
```
kubectl port-forward service/service-2048 8080:80 -n game-2048
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
```

Preview the running (port-forwarded service) application from the cloud 9 IDE

`Preview -> Preview Running Application`



You should then see the app running in the browser


AWS Cloud9
Welcome to your development environment
Developer Tools
2048
SCORE 32
BEST 156
Join the numbers and get to the 2048 tile!
New Game
2

```
NAME                            READY   UP-TO-DATE   AVAILABLE   AGE   C
ONTAINERS   IMAGES                                                      SELE
CTOR
deployment.apps/deployment-2048   2/2     2            2           55s   a
pp-2048     566972129213.dkr.ecr.eu-west-1.amazonaws.com/sample-app   app.
kubernetes.io/name=app-2048
andyt530:~/environment/tfekscode/sampleapp (master) $ kubectl port-forward
 service/service-2048 8080:80 -n game-2048
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
```

Finding the Internal Load Balancer

As before with the CLI the CI/CD pipeline has also deployed a Load Balancer.

The load balancer will take about 8 minutes to provision and come online

Check how long it has bene provisioning by using the command:

```
1
kubectl get ingress -n game-2048
NAME        CLASS   HOSTS  ADDRESS  PORTS  AGE
```

```
ingress-2048  <none>  *            80    5m27s
```

Watching the aws-load-balancer-controller - open another terminal and use this command to watch the logs:

```
kubectl logs `kubectl get pods -n kube-system | grep aws-load-balancer-controller | awk '{print $1}'` -n kube-system --follow
```

After 8 minutes have elapsed

Check the `targetbindings` have populated. This is the new CRD type that was created as part of the load balancer controller installation.

```
kubectl get targetgroupbindings -A
NAMESPACE   NAME                           SERVICE-NAME   SERVICE-PORT   TARGET-TYPE   AGE
game-2048   k8s-game2048-service2-11af83fe8f   service-2048   80           ip           82s
```

Then obtain the internal DNS name of the load balancer using and check valid HTML is returned with curl

```
ALB=$(aws elbv2 describe-load-balancers --query 'LoadBalancers[*].DNSName' | jq -r .[])
curl $ALB:8080
<!DOCTYPE html>`
<html>
<head>
  <meta charset="utf-8">
  <title>2048</title>

** Output truncated for brevity **

  <script src="js/application.js"></script>
</body>
</html>
```

Cleanup

Interrupt the port forwarding with ctrl-c if necessary.

The following destroy operation make take up to 12 minutes as it deletes the ingress and ALB

```
cd ~/environment/tfekscode/sampleapp
terraform destroy -auto-approve
null_resource.cleanup: Destroying... [id=9012327125218962041]
null_resource.cleanup: Provisioning with 'local-exec'...
null_resource.cleanup (local-exec): Executing: ["/bin/bash" "-c" "       echo \"remote git credentials &\" sample app\n       ./cleanup.sh\n       echo \"********************************************************************************\"\n"]
null_resource.cleanup (local-exec): remote git credentials & sample app
kubernetes_namespace.game-2048: Destroying... [id=game-2048]
kubernetes_service.game-2048__service-2048: Destroying... [id=game-2048/service-2048]
kubernetes_ingress.game-2048__ingress-2048: Destroying... [id=game-2048/ingress-2048]
kubernetes_deployment.game-2048__deployment-2048: Destroying... [id=game-2048/deployment-2048]
kubernetes_ingress.game-2048__ingress-2048: Destruction complete after 2s
kubernetes_service.game-2048__service-2048: Destruction complete after 2s
kubernetes_deployment.game-2048__deployment-2048: Destruction complete after 2s
null_resource.cleanup (local-exec): ********************************************************************************
null_resource.cleanup: Destruction complete after 3s
kubernetes_namespace.game-2048: Still destroying... [id=game-2048, 10s elapsed]
kubernetes_namespace.game-2048: Still destroying... [id=game-2048, 20s elapsed]

...

kubernetes_ingress_v1.game-2048__ingress-2048: Still destroying... [id=game-2048/ingress-2048, 7m50s elapsed]
kubernetes_namespace.game-2048: Still destroying... [id=game-2048, 7m50s elapsed]
kubernetes_ingress_v1.game-2048__ingress-2048: Destruction complete after 7m57s
kubernetes_namespace.game-2048: Still destroying... [id=game-2048, 8m0s elapsed]
kubernetes_namespace.game-2048: Destruction complete after 8m4s
```

```
Destroy complete! Resources: 5 destroyed.
```

Note: it's only possible to delete the application from the command line like this because we are using S3 for the Terraform backend state files.