# Deploy the sample app to EKS using Terraform

In this chapter you will deploy a sample application using Terraform.

The biggest benefit when using Terraform to maintain Kubernetes resources is integration into the Terraform plan/apply life-cycle. So you can review planned changes before applying them.

Also, using kubectl, purging of resources from the cluster is not trivial without manual intervention. Terraform does this reliably.

For a discussion of other benefits see here:

---

```
1
cd ~/environment/tfekscode/sampleapp
```

Initialize Terraform:

```
1
terraform init
```

Plan the deployment:

```
1
terraform plan -out tfplan
data.aws_caller_identity.current: Reading...
data.aws_region.current: Reading...
data.aws_region.current: Read complete after 0s [id=eu-west-1]
data.aws_caller_identity.current: Read complete after 1s [id=440018911661]


Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  + create


Terraform will perform the following actions:


  # kubernetes_deployment.game-2048__deployment-2048 will be created
  + resource "kubernetes_deployment" "game-2048__deployment-2048" {
      + id               = (known after apply)
```

```
+ wait_for_rollout = true

+ metadata {
   + generation      = (known after apply)
   + name            = "deployment-2048"
   + namespace       = "game-2048"
   + resource_version = (known after apply)
   + uid             = (known after apply)
 }

+ spec {
   + min_ready_seconds        = 0
   + paused                   = false
   + progress_deadline_seconds = 600
   + replicas                 = "4"
   + revision_history_limit    = 10

   + selector {
      + match_labels = {
         + "app.kubernetes.io/name" = "app-2048"
       }
    }

   + strategy {
      + type = "RollingUpdate"

      + rolling_update {
         + max_surge      = "25%"
         + max_unavailable = "25%"
       }
    }

   + template {
      + metadata {
         + generation      = (known after apply)
         + labels          = {
```

```
      + "app.kubernetes.io/name" = "app-2048"
    }
  + name            = (known after apply)
  + resource_version = (known after apply)
  + uid             = (known after apply)
 }
+ spec {
  + automount_service_account_token  = true
  + dns_policy                       = "ClusterFirst"
  + enable_service_links             = true
  + host_ipc                         = false
  + host_network                     = false
  + host_pid                         = false
  + hostname                         = (known after apply)
  + node_name                        = (known after apply)
  + restart_policy                   = "Always"
  + service_account_name             = (known after apply)
  + share_process_namespace          = false
  + termination_grace_period_seconds = 30

  + container {
    + image                    = "440018911661.dkr.ecr.eu-west-1.amazonaws.com/aws/awsandy/docker-
2048"
    + image_pull_policy        = "Always"
    + name                     = "app-2048"
    + stdin                    = false
    + stdin_once               = false
    + termination_message_path   = "/dev/termination-log"
    + termination_message_policy = (known after apply)
    + tty                      = false

    + port {
      + container_port = 80
      + protocol       = "TCP"
    }

    + resources {
```

```
            + limits   = (known after apply)
            + requests = (known after apply)
          }
        }
      }
    }
  }

# kubernetes_ingress_v1.game-2048__ingress-2048 will be created
+ resource "kubernetes_ingress_v1" "game-2048__ingress-2048" {
  + id     = (known after apply)
  + status = (known after apply)

  + metadata {
    + annotations     = {
      + "alb.ingress.kubernetes.io/listen-ports" = jsonencode(
          [
            + {
              + HTTP = 8080
            },
          ]
        )
      + "alb.ingress.kubernetes.io/scheme"      = "internal"
      + "alb.ingress.kubernetes.io/target-type"  = "ip"
      }
    + generation      = (known after apply)
    + name            = "ingress-2048"
    + namespace       = "game-2048"
    + resource_version = (known after apply)
    + uid             = (known after apply)
  }

  + spec {
    + ingress_class_name = "alb"
```

```
      + rule {
        + http {
          + path {
            + path     = "/"
            + path_type = "ImplementationSpecific"

            + backend {
              + service {
                + name = "service-2048"

                + port {
                  + number = 80
                }
              }
            }
          }
        }
      }
    }
  }

# kubernetes_namespace.game-2048 will be created
+ resource "kubernetes_namespace" "game-2048" {
    + id = (known after apply)

    + metadata {
      + generation      = (known after apply)
      + name            = "game-2048"
      + resource_version = (known after apply)
      + uid             = (known after apply)
    }

    + timeouts {
      + delete = "20m"
    }
  }
```

```
# kubernetes_service.game-2048__service-2048 will be created
+ resource "kubernetes_service" "game-2048__service-2048" {
    + id                   = (known after apply)
    + status               = (known after apply)
    + wait_for_load_balancer = true

    + metadata {
        + generation      = (known after apply)
        + name            = "service-2048"
        + namespace       = "game-2048"
        + resource_version = (known after apply)
        + uid             = (known after apply)
      }

    + spec {
        + allocate_load_balancer_node_ports = true
        + cluster_ip                  = (known after apply)
        + cluster_ips                 = (known after apply)
        + external_traffic_policy     = (known after apply)
        + health_check_node_port      = (known after apply)
        + internal_traffic_policy     = (known after apply)
        + ip_families                 = (known after apply)
        + ip_family_policy            = (known after apply)
        + publish_not_ready_addresses = false
        + selector                    = {
            + "app.kubernetes.io/name" = "app-2048"
          }
        + session_affinity            = "None"
        + type                        = "NodePort"

        + port {
            + node_port   = (known after apply)
            + port        = 80
            + protocol    = "TCP"
            + target_port = "80"
```

```
      }
    }
  }

 # null_resource.cleanup will be created
 + resource "null_resource" "cleanup" {
    + id       = (known after apply)
    + triggers = {}
  }

Plan: 5 to add, 0 to change, 0 to destroy.
```

## Deploy the sample app:

```
1
terraform apply tfplan
null_resource.cleanup: Creating...
null_resource.cleanup: Creation complete after 0s [id=7523966713166509195]
kubernetes_service.game-2048__service-2048: Creating...
kubernetes_ingress_v1.game-2048__ingress-2048: Creating...
kubernetes_namespace.game-2048: Creating...
kubernetes_deployment.game-2048__deployment-2048: Creating...
kubernetes_namespace.game-2048: Creation complete after 2s [id=game-2048]
kubernetes_ingress_v1.game-2048__ingress-2048: Creation complete after 2s [id=game-2048/ingress-2048]
kubernetes_service.game-2048__service-2048: Creation complete after 2s [id=game-2048/service-2048]
kubernetes_deployment.game-2048__deployment-2048: Creation complete after 4s [id=game-2048/deployment-2048]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
```

## Check everything is running ?

```
1
kubectl get pods,svc,deployment -n game-2048
NAMESPACE   NAME                                    READY   STATUS    RESTARTS   AGE
game-2048   pod/deployment-2048-d6457c6fb-6sx2x     1/1     Running   0          52s
game-2048   pod/deployment-2048-d6457c6fb-gfgsd     1/1     Running   0          52s
game-2048   pod/deployment-2048-d6457c6fb-q5n6c     1/1     Running   0          52s
```

```
game-2048   pod/deployment-2048-d6457c6fb-vk6nv          1/1   Running  0      52s

NAMESPACE   NAME                       TYPE      CLUSTER-IP      EXTERNAL-IP  PORT(S)      AGE
game-2048   service/service-2048             NodePort   172.20.190.226  <none>     80:32360/TCP   52s

NAMESPACE   NAME                          READY  UP-TO-DATE  AVAILABLE  AGE
game-2048   deployment.apps/deployment-2048         4/4   4       4       52s
```

Note that:

- The pods are deployed to 100.64.x.x addresses
- The service is exposing port 80
- The deployment is referencing a private ECR repository belonging to your account

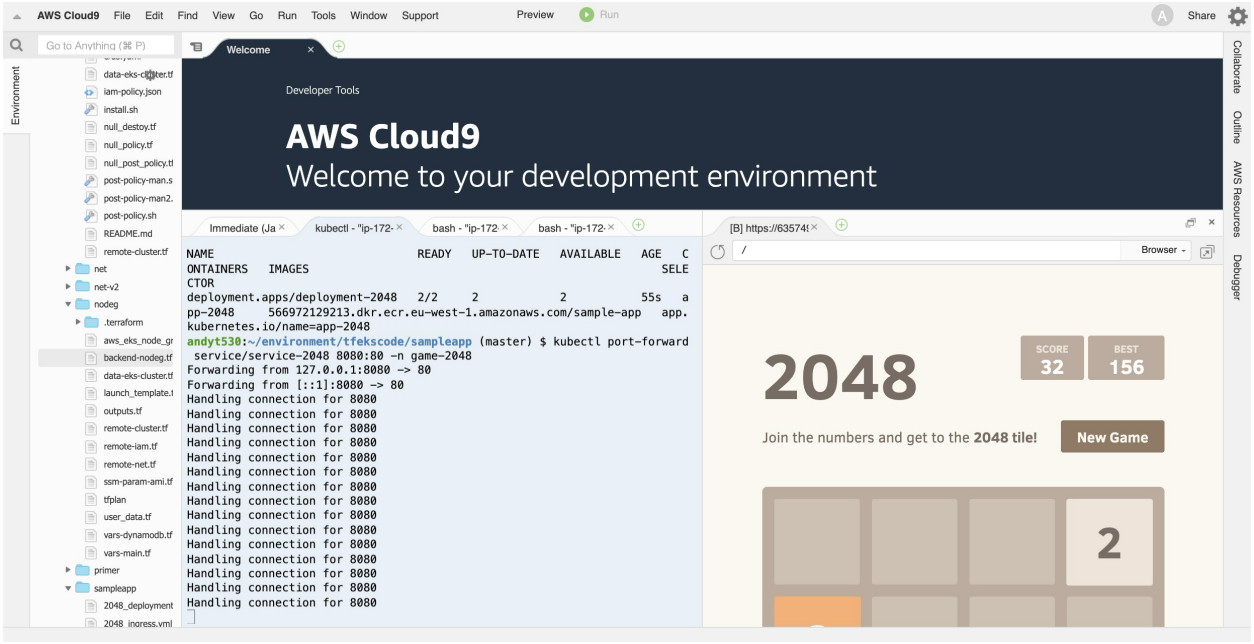
Enable port forwarding so we can see the application in out Cloud9 IDE:

```
1
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




Interrupt the port forwarding with **ctrl-C**

# Finding the Internal Load Balancer

As part of the build above we 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:

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

### After 7 to 12 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.

```
1
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

```
1
2
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

Use terraform to delete our sample application:

```
1
terraform destroy -auto-approve
```

Note the namespace takes several minutes (up to 20 mins) to delete as it waits for the ingress resource to be deleted.

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, 17m0s elapsed]

kubernetes_ingress_v1.game-2048__ingress-2048: Still destroying... [id=game-2048/ingress-2048, 17m10s elapsed]

kubernetes_ingress_v1.game-2048__ingress-2048: Destruction complete after 17m19s

kubernetes_namespace.game-2048: Destroying... [id=game-2048]

kubernetes_namespace.game-2048: Destruction complete after 6s

Destroy complete! Resources: 5 destroyed.