Deploy the sample app to two node groups

In this chapter you will deploy two sample applications to separate node groups

```
1
cd ~/environment/tfekscode/extra/sampleapp2
```

Initialize Terraform - note in the output that the **kubernetes** provider is also installed

```
1
terraform init
```

```
1
terraform plan -out tfplan
```

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.game1-2048__deployment1-2048 will be created
  + resource "kubernetes_deployment" "game1-2048__deployment1-2048" {
      + id            = (known after apply)
      + wait_for_rollout = true

      + metadata {
          + generation      = (known after apply)
          + name            = "deployment1-2048"
          + namespace       = "game1-2048"
          + 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" = "app1-2048"
    }
 }

+ strategy {
   + type = "RollingUpdate"

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

+ template {
   + metadata {
      + generation      = (known after apply)
      + labels          = {
         + "app.kubernetes.io/name" = "app1-2048"
       }
      + 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)
            + node_selector                = {
              + "eks/nodegroup-name" = "ng1-mycluster1"
             }
            + restart_policy               = "Always"
            + service_account_name         = (known after apply)
            + share_process_namespace      = false
            + termination_grace_period_seconds = 30

            + container {
              + image                      = "666763910423.dkr.ecr.eu-west-1.amazonaws.com/aws/awsandy/docker-
2048"
              + image_pull_policy          = "Always"
              + name                       = "app1-2048"
              + 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_deployment.game2-2048__deployment2-2048 will be created
 + resource "kubernetes_deployment" "game2-2048__deployment2-2048" {
```

```
+ id              = (known after apply)
+ wait_for_rollout = true

+ metadata {
   + generation      = (known after apply)
   + name            = "deployment2-2048"
   + namespace       = "game2-2048"
   + 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" = "app2-2048"
       }
    }

   + strategy {
      + type = "RollingUpdate"

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

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

```
      + labels         = {
        + "app.kubernetes.io/name" = "app2-2048"
       }
      + 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)
      + node_selector                    = {
        + "eks/nodegroup-name" = "ng2-mycluster1"
       }
      + restart_policy                   = "Always"
      + service_account_name             = (known after apply)
      + share_process_namespace          = false
      + termination_grace_period_seconds = 30

      + container {
        + image                    = "666763910423.dkr.ecr.eu-west-1.amazonaws.com/aws/awsandy/docker-
2048"
        + image_pull_policy        = "Always"
        + name                     = "app2-2048"
        + 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.game1-2048__ingress1-2048 will be created
+ resource "kubernetes_ingress_v1" "game1-2048__ingress1-2048" {
  + id     = (known after apply)
  + status = (known after apply)

  + metadata {
    + annotations     = {
      + "alb.ingress.kubernetes.io/listen-ports" = jsonencode(
          [
            + {
              + HTTP = 8081
            },
          ]
        )
      + "alb.ingress.kubernetes.io/scheme"      = "internal"
      + "alb.ingress.kubernetes.io/target-type"  = "ip"
      }
    + generation      = (known after apply)
    + name            = "ingress1-2048"
    + namespace       = "game1-2048"
    + resource_version = (known after apply)
    + uid             = (known after apply)
  }
```

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

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

          + backend {
            + service {
              + name = "service1-2048"

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

# kubernetes_ingress_v1.game2-2048__ingress2-2048 will be created
+ resource "kubernetes_ingress_v1" "game2-2048__ingress2-2048" {
  + id     = (known after apply)
  + status = (known after apply)

  + metadata {
    + annotations     = {
      + "alb.ingress.kubernetes.io/listen-ports" = jsonencode(
          [
          + {
            + HTTP = 8082
```

```
            },
          ]
        )
      + "alb.ingress.kubernetes.io/scheme"      = "internal"
      + "alb.ingress.kubernetes.io/target-type"  = "ip"
    }
  + generation      = (known after apply)
  + name            = "ingress2-2048"
  + namespace       = "game2-2048"
  + resource_version = (known after apply)
  + uid             = (known after apply)
 }

+ spec {
  + ingress_class_name = "alb"

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

        + backend {
          + service {
            + name = "service2-2048"

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

```
# kubernetes_namespace.game1-2048 will be created
+ resource "kubernetes_namespace" "game1-2048" {
  + id = (known after apply)

  + metadata {
    + generation      = (known after apply)
    + name            = "game1-2048"
    + resource_version = (known after apply)
    + uid             = (known after apply)
   }

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

# kubernetes_namespace.game2-2048 will be created
+ resource "kubernetes_namespace" "game2-2048" {
  + id = (known after apply)

  + metadata {
    + generation      = (known after apply)
    + name            = "game2-2048"
    + resource_version = (known after apply)
    + uid             = (known after apply)
   }

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

# kubernetes_service.game1-2048__service1-2048 will be created
+ resource "kubernetes_service" "game1-2048__service1-2048" {
  + id                   = (known after apply)
```

```
  + status                = (known after apply)
  + wait_for_load_balancer = true

  + metadata {
      + generation      = (known after apply)
      + name            = "service1-2048"
      + namespace       = "game1-2048"
      + 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" = "app1-2048"
        }
      + session_affinity            = "None"
      + type                        = "NodePort"

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

```
# kubernetes_service.game2-2048__service2-2048 will be created
+ resource "kubernetes_service" "game2-2048__service2-2048" {
    + id                   = (known after apply)
    + status               = (known after apply)
    + wait_for_load_balancer = true

    + metadata {
        + generation      = (known after apply)
        + name            = "service2-2048"
        + namespace       = "game2-2048"
        + 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" = "app2-2048"
          }
        + session_affinity            = "None"
        + type                        = "NodePort"

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

```
    }
  }
```

Plan: 8 to add, 0 to change, 0 to destroy.

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.game1-2048__deployment1-2048 will be created
 + resource "kubernetes_deployment" "game1-2048__deployment1-2048" {
    + id              = (known after apply)
    + wait_for_rollout = true

    + metadata {
       + generation      = (known after apply)
       + name            = "deployment1-2048"
       + namespace       = "game1-2048"
       + 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" = "app1-2048"
           }
```

```
 }

+ strategy {
   + type = "RollingUpdate"

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

+ template {
   + metadata {
      + generation      = (known after apply)
      + labels          = {
         + "app.kubernetes.io/name" = "app1-2048"
       }
      + 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)
      + node_selector                    = {
         + "eks/nodegroup-name" = "ng1-mycluster1"
       }
      + restart_policy                   = "Always"
      + service_account_name             = (known after apply)
      + share_process_namespace          = false
```

```
            + termination_grace_period_seconds = 30

            + container {
                + image                    = "666763910423.dkr.ecr.eu-west-1.amazonaws.com/aws/awsandy/docker-
2048"
                + image_pull_policy        = "Always"
                + name                     = "app1-2048"
                + 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_deployment.game2-2048__deployment2-2048 will be created
  + resource "kubernetes_deployment" "game2-2048__deployment2-2048" {
    + id             = (known after apply)
    + wait_for_rollout = true

    + metadata {
        + generation      = (known after apply)
        + name            = "deployment2-2048"
        + namespace       = "game2-2048"
```

```
    + 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" = "app2-2048"
          }
      }

    + strategy {
        + type = "RollingUpdate"

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

    + template {
        + metadata {
            + generation      = (known after apply)
            + labels          = {
                + "app.kubernetes.io/name" = "app2-2048"
              }
            + 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)
          + node_selector                    = {
              + "eks/nodegroup-name" = "ng2-mycluster1"
            }
          + restart_policy                   = "Always"
          + service_account_name             = (known after apply)
          + share_process_namespace          = false
          + termination_grace_period_seconds = 30

          + container {
              + image                    = "666763910423.dkr.ecr.eu-west-1.amazonaws.com/aws/awsandy/docker-
2048"
              + image_pull_policy        = "Always"
              + name                     = "app2-2048"
              + 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.game1-2048__ingress1-2048 will be created
+ resource "kubernetes_ingress_v1" "game1-2048__ingress1-2048" {
    + id     = (known after apply)
    + status = (known after apply)

    + metadata {
        + annotations     = {
            + "alb.ingress.kubernetes.io/listen-ports" = jsonencode(
                [
                  + {
                      + HTTP = 8081
                    },
                ]
            )
            + "alb.ingress.kubernetes.io/scheme"      = "internal"
            + "alb.ingress.kubernetes.io/target-type"  = "ip"
          }
        + generation      = (known after apply)
        + name            = "ingress1-2048"
        + namespace       = "game1-2048"
        + resource_version = (known after apply)
        + uid             = (known after apply)
      }

    + spec {
        + ingress_class_name = "alb"

        + rule {
            + http {
                + path {
```

```
            + path      = "/"
            + path_type = "ImplementationSpecific"

            + backend {
              + service {
                + name = "service1-2048"

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

# kubernetes_ingress_v1.game2-2048__ingress2-2048 will be created
+ resource "kubernetes_ingress_v1" "game2-2048__ingress2-2048" {
   + id     = (known after apply)
   + status = (known after apply)

   + metadata {
      + annotations     = {
        + "alb.ingress.kubernetes.io/listen-ports" = jsonencode(
            [
             + {
               + HTTP = 8082
              },
            ]
         )
        + "alb.ingress.kubernetes.io/scheme"      = "internal"
        + "alb.ingress.kubernetes.io/target-type"  = "ip"
       }
      + generation      = (known after apply)
```

```
      + name            = "ingress2-2048"
      + namespace       = "game2-2048"
      + resource_version = (known after apply)
      + uid             = (known after apply)
    }

  + spec {
      + ingress_class_name = "alb"

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

                  + backend {
                      + service {
                          + name = "service2-2048"

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

# kubernetes_namespace.game1-2048 will be created
+ resource "kubernetes_namespace" "game1-2048" {
  + id = (known after apply)

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

```
      + name           = "game1-2048"
      + resource_version = (known after apply)
      + uid            = (known after apply)
    }

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

# kubernetes_namespace.game2-2048 will be created
+ resource "kubernetes_namespace" "game2-2048" {
  + id = (known after apply)

  + metadata {
      + generation      = (known after apply)
      + name           = "game2-2048"
      + resource_version = (known after apply)
      + uid            = (known after apply)
    }

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

# kubernetes_service.game1-2048__service1-2048 will be created
+ resource "kubernetes_service" "game1-2048__service1-2048" {
  + id                  = (known after apply)
  + status              = (known after apply)
  + wait_for_load_balancer = true

  + metadata {
      + generation      = (known after apply)
      + name           = "service1-2048"
      + namespace       = "game1-2048"
```

```
      + 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" = "app1-2048"
        }
      + session_affinity                  = "None"
      + type                              = "NodePort"

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

# kubernetes_service.game2-2048__service2-2048 will be created
+ resource "kubernetes_service" "game2-2048__service2-2048" {
    + id                   = (known after apply)
    + status               = (known after apply)
    + wait_for_load_balancer = true

    + metadata {
```

```
      + generation       = (known after apply)
      + name             = "service2-2048"
      + namespace        = "game2-2048"
      + 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" = "app2-2048"
        }
      + session_affinity                  = "None"
      + type                              = "NodePort"

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

Plan: 8 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------
```

This plan was saved to: tfplan

To perform exactly these actions, run the following command to apply:

terraform apply "tfplan"

Deploy the Kubernetes application:

1

terraform apply tfplan

kubernetes_ingress_v1.game1-2048__ingress1-2048: Creating...

kubernetes_service.game1-2048__service1-2048: Creating...

kubernetes_namespace.game1-2048: Creating...

kubernetes_service.game2-2048__service2-2048: Creating...

kubernetes_ingress_v1.game2-2048__ingress2-2048: Creating...

kubernetes_namespace.game2-2048: Creating...

kubernetes_deployment.game2-2048__deployment2-2048: Creating...

kubernetes_deployment.game1-2048__deployment1-2048: Creating...

kubernetes_namespace.game1-2048: Creation complete after 2s [id=game1-2048]

kubernetes_namespace.game2-2048: Creation complete after 2s [id=game2-2048]

kubernetes_ingress_v1.game2-2048__ingress2-2048: Creation complete after 2s [id=game2-2048/ingress2-2048]

kubernetes_ingress_v1.game1-2048__ingress1-2048: Creation complete after 2s [id=game1-2048/ingress1-2048]

kubernetes_service.game2-2048__service2-2048: Creation complete after 2s [id=game2-2048/service2-2048]

kubernetes_service.game1-2048__service1-2048: Creation complete after 2s [id=game1-2048/service1-2048]

kubernetes_deployment.game2-2048__deployment2-2048: Creation complete after 4s [id=game2-2048/deployment2-2048]

kubernetes_deployment.game1-2048__deployment1-2048: Creation complete after 4s [id=game1-2048/deployment1-2048]

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Check everything is running ?

1

kubectl get pods,svc,deployment -A -o wide | grep game

```
game1-2048   pod/deployment-2048-ng1-788c7f7874-mccgw        1/1   Running   0       2m8s
100.64.100.188   ip-10-0-2-179.eu-west-1.compute.internal   <none>        <none>

game1-2048   pod/deployment-2048-ng1-788c7f7874-nvlqq        1/1   Running   0       2m8s
100.64.42.14    ip-10-0-1-25.eu-west-1.compute.internal    <none>        <none>

game2-2048   pod/deployment-2048-ng2-74bbf67dc5-w9sbh        1/1   Running   0       2m7s
10.0.2.166      ip-10-0-2-71.eu-west-1.compute.internal    <none>        <none>

game2-2048   pod/deployment-2048-ng2-74bbf67dc5-zq7p6        1/1   Running   0       2m7s   10.0.1.180
ip-10-0-1-231.eu-west-1.compute.internal   <none>        <none>

game1-2048   service/service1-2048              NodePort   172.20.87.238   <none>      80:32481/TCP
2m6s   app.kubernetes.io/name=app1-2048

game1-2048   service/service2-2048              NodePort   172.20.206.182   <none>      80:30243/TCP
2m5s   app.kubernetes.io/name=app2-2048

game1-2048   deployment.apps/deployment-2048-ng1        2/2   2        2       2m8s   app1-2048
136434655158.dkr.ecr.eu-west-1.amazonaws.com/sample-app
app.kubernetes.io/name=app1-2048

game2-2048   deployment.apps/deployment-2048-ng2        2/2   2        2       2m7s   app2-2048
136434655158.dkr.ecr.eu-west-1.amazonaws.com/sample-app
app.kubernetes.io/name=app2-2048
```

Note from the output that:

- The pods are deployed to 110.64.x.x (node group 1) and 10.0.x.x addresses (node group 2).
- The services are exposing port 80.
- The deployment is referencing a private ECR repository belonging to your account.

If you see pods apparently stuck in "ContainerCreating" mode for a minute or more try the following:

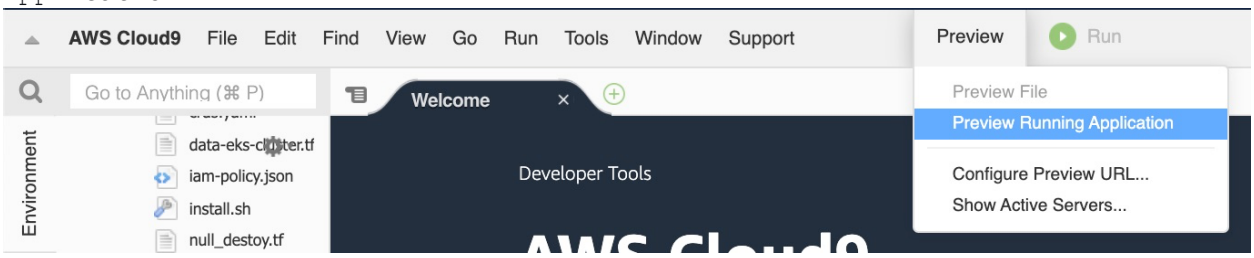
Expand here to see the fix

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

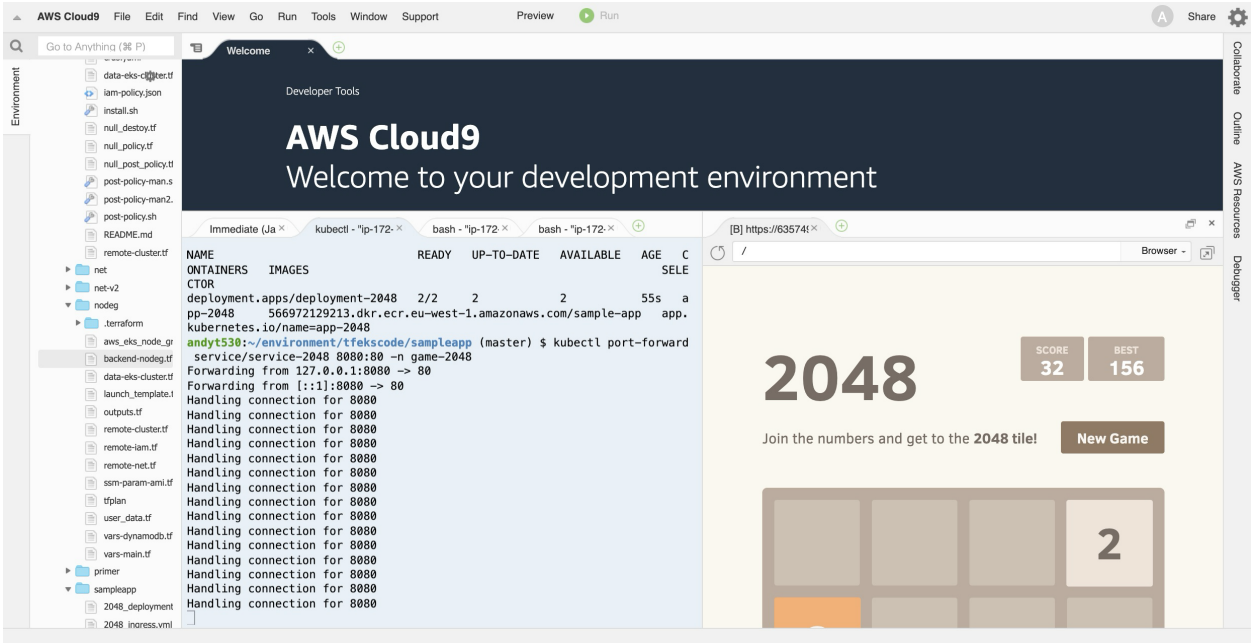
```
1
kubectl port-forward service/service2-2048 8081:80 -n game2-2048
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
Handling connection for 8081
Handling connection for 8081
Handling connection for 8081
```

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



As the Terraform files are similar to the previous section they are not explained here.

Cleanup

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

Then use Terraform to delete the Kubernetes resources:

1

terraform destroy -auto-approve