Terraform files explanation

Terraform files and explanation

The first three files have been pre-created from the gen-backend.sh script in the tf-setup stage.

backend-sample.tf & vars-main.tf

As previously described.

---

k8s.tf

Include the kubernetes provider

```
provider "kubernetes" {}
```

Define the namespace using Terraform, notice the extended timeout period for deletion **delete = "20m"** to allow ingress resources time to delete.

sampleapp-namespace.tf

```
resource "kubernetes_namespace" "game-2048" {
  metadata {
    name = "game-2048"
  }

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

sampleapp-deployment.tf

Define the deployment. There are a few notable parts to this file:

Two data resources are specified that return the current AWS region and account id

data "aws_region" "current" {} data "aws_caller_identity" "current" {}

These data resources are then used to construct a string for the image location.

image = format("%s.dkr.ecr.%s.amazonaws.com/sample-app", data.aws_caller_identity.current.account_id, data.aws_region.current.name)

There's also a node selector to ensure the app runs on our specific node group

node_selector = { "alpha.eksctl.io/nodegroup-name" = "ng1-mycluster1" }

```
data "aws_region" "current" {}
data "aws_caller_identity" "current" {}

resource "kubernetes_deployment" "game-2048__deployment-2048" {

 metadata {
  name      = "deployment-2048"
  namespace = "game-2048"
 }

 spec {
  replicas = 4
  selector {
   match_labels = {
    "app.kubernetes.io/name" = "app-2048"
   }
  }
  strategy {
   type = "RollingUpdate"
```

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

  template {
    metadata {
      annotations = {}
      labels      = { "app.kubernetes.io/name" = "app-2048" }
    }

    spec {

      node_selector                 = { "alpha.eksctl.io/nodegroup-name" = "ng1-mycluster1" }
      restart_policy                = "Always"
      share_process_namespace       = false
      termination_grace_period_seconds = 30

      container {
        image             = format("%s.dkr.ecr.%s.amazonaws.com/sample-app",
data.aws_caller_identity.current.account_id, data.aws_region.current.name)
        image_pull_policy = "Always"
        name              = "app-2048"
        port {
          container_port = 80
          host_port      = 0
          protocol       = "TCP"
        }

        resources {
        }
      }
    }
  }
}
```

sampleapp-service.tf

Define the service:

```
resource "kubernetes_service" "game-2048_service-2048" {

 metadata {
  name      = "service-2048"
  namespace = "game-2048"
 }

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

  type = "NodePort"

  port {
   port       = 80
   protocol    = "TCP"
   target_port = "80"
  }
 }

}
```

sampleapp-ingress.tf

Define the ingress resource - this creates a AWS Application Load Balancer via the previously installed aws-load-balancer-controller.

Note how multiple annotations are passed, including the listener port 8080.

```
resource "kubernetes_ingress" "game-2048__ingress-2048" {
  metadata {
    annotations = {"kubernetes.io/ingress.class" = "alb", "alb.ingress.kubernetes.io/scheme" = "internal",
"alb.ingress.kubernetes.io/target-type" = "ip", "alb.ingress.kubernetes.io/listen-ports" = "[{\"HTTP\": 8080}]"
}
    name       = "ingress-2048"
    namespace   = "game-2048"
  }

  spec {

    rule {
      http {
        path {
          path = "/*"
          backend {
            service_name = "service-2048"
            service_port = "80"
          }
        }
      }
    }
  }
}
```