2. Fargate Application (Optional)

Run an application on fargate

```
cd ~/environment/tfekscode/extra/fargateapp
```

Initialize Terraform:

```
terraform init
Initializing the backend...

** OUTPUT TRUNCATED FOR BREVITY  as similar to previous examples **
```

Validate the Terraform code

```
terraform validate
Success! The configuration is valid.
```

Plan the deployment:

```
terraform plan -out tfplan
data.aws_caller_identity.current: Reading...
data.aws_availability_zones.az: Reading...
data.aws_region.current: Reading...
data.aws_region.current: Read complete after 0s [id=eu-west-1]
data.aws_availability_zones.az: 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_config_map.aws-observability__aws-logging will be created
+ resource "kubernetes_config_map" "aws-observability__aws-logging" {
  + data = {
    + "output.conf" = <<-EOT
        [OUTPUT]
          Name cloudwatch
          Match *
          region eu-west-1
          log_group_name fluent-bit-eks-fargate
          log_stream_prefix fargate1-
          auto_create_group true
          sts_endpoint https://sts.eu-west-1.amazonaws.com
          endpoint https://logs.eu-west-1.amazonaws.com
      EOT
   }
  + id   = (known after apply)

  + metadata {
    + generation       = (known after apply)
    + name             = "aws-logging"
    + namespace        = (known after apply)
    + resource_version = (known after apply)
    + uid              = (known after apply)
   }
 }

# kubernetes_deployment.fargate1__logging_server will be created
+ resource "kubernetes_deployment" "fargate1__logging_server" {
  + id               = (known after apply)
  + wait_for_rollout = true

  + metadata {
    + generation       = (known after apply)
    + name             = "logging-server"
    + namespace        = (known after apply)
    + resource_version = (known after apply)
    + uid              = (known after apply)
  }

+ spec {
    + min_ready_seconds         = 0
    + paused                    = false
    + progress_deadline_seconds = 600
    + replicas                  = "2"
    + revision_history_limit    = 10

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

    + strategy {
        + type = "RollingUpdate"

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

    + template {
        + metadata {
            + generation       = (known after apply)
            + labels           = {
                + "app.kubernetes.io/name" = "logging-server"
              }
            + 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/nginx/nginx"
                + image_pull_policy          = "Always"
                + name                       = "nginx"
                + 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)
                 }
              }
          }
      }
```

```
    }

   + timeouts {
      + create = "3m"
    }
  }

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

   + metadata {
      + generation       = (known after apply)
      + labels           = {
         + "aws-observability" = "enabled"
        }
      + name             = "aws-observability"
      + resource_version = (known after apply)
      + uid              = (known after apply)
    }

   + timeouts {}
  }

# kubernetes_namespace.fargate1 will be created
+ resource "kubernetes_namespace" "fargate1" {
   + id = (known after apply)

   + metadata {
      + generation       = (known after apply)
      + name             = "fargate1"
      + resource_version = (known after apply)
      + uid              = (known after apply)
    }

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

# kubernetes_service.fargate1__service-logger will be created
+ resource "kubernetes_service" "fargate1__service-logger" {
    + id                    = (known after apply)
    + status                = (known after apply)
    + wait_for_load_balancer = true

    + metadata {
        + generation       = (known after apply)
        + name             = "service-logging"
        + namespace        = "fargate1"
        + 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" = "logging-server"
          }
        + session_affinity                  = "None"
        + type                              = "NodePort"

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

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

Build the environment:

```
1
terraform apply tfplan
kubernetes_service.fargate1__service-logger: Creating...
kubernetes_namespace.fargate1: Creating...
kubernetes_namespace.aws-observability: Creating...
kubernetes_namespace.aws-observability: Creation complete after 2s [id=aws-observability]
kubernetes_namespace.fargate1: Creation complete after 2s [id=fargate1]
kubernetes_config_map.aws-observability__aws-logging: Creating...
kubernetes_deployment.fargate1__logging_server: Creating...
kubernetes_service.fargate1__service-logger: Creation complete after 2s [id=fargate1/service-logging]
kubernetes_config_map.aws-observability__aws-logging: Creation complete after 0s [id=aws-
observability/aws-logging]
kubernetes_deployment.fargate1__logging_server: Still creating... [10s elapsed]
....
kubernetes_deployment.fargate1__logging_server: Still creating... [1m0s elapsed]
kubernetes_deployment.fargate1__logging_server: Creation complete after 1m5s [id=fargate1/logging-server]

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

Check for resources running on the Fargare namespace `fargate1`

```
1
kubectl get all -n fargate1
NAME                               READY   STATUS    RESTARTS   AGE
pod/logging-server-5ccb5956b6-kbl66   1/1     Running   0          4m3s
pod/logging-server-5ccb5956b6-psvgc   1/1     Running   0          4m3s
```

```
NAME                    TYPE      CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
service/service-logging   NodePort   172.20.129.2   <none>        80:30017/TCP   4m3s

NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/logging-server   2/2     2            2           4m3s

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/logging-server-5ccb5956b6   2         2         2       4m3s
```

Look for the Fargate nodes:

```
kubectl get nodes
```

Note how we have two fargate nodes in the cluster:

```
NAME                                                    STATUS   ROLES    AGE    VERSION
fargate-ip-100-64-101-77.eu-west-1.compute.internal   Ready    <none>   87s    v1.24.9-eks-300e41d
fargate-ip-100-64-58-186.eu-west-1.compute.internal   Ready    <none>   90s    v1.24.9-eks-300e41d
ip-10-0-1-19.eu-west-1.compute.internal                 Ready    <none>   3h3m   v1.24.11-eks-a59e1f0
ip-10-0-1-248.eu-west-1.compute.internal                Ready    <none>   40m    v1.24.11-eks-a59e1f0
ip-10-0-2-153.eu-west-1.compute.internal                Ready    <none>   3h3m   v1.24.11-eks-a59e1f0
ip-10-0-2-235.eu-west-1.compute.internal                Ready    <none>   40m    v1.24.11-eks-a59e1f0
```

Test the Fargate application logging

```
kubectl port-forward service/service-logging 8080:80 -n fargate1
```

browse away localhost:8080 or:

```
curl localhost:8080
```

Look in CloudWatch logs

CloudWatch > `Log groups` > `fluent-bit-eks-fargate`

Look for the lastest log group you you should see entries like this:

```
{
   "log": "2023-04-16T17:45:00.387598272Z stdout F 127.0.0.1 - - [16/Apr/2023:17:45:00 +0000] \"GET / HTTP/1.1\" 200 615 \"-\" \"curl/7.61.1\" \"-\""
}
```

etc.