Terraform files explanation

Terraform files and explanation

The first five files have been pre-created from the gen-backend.sh script in the tf-setup stage, The S3 bucket and DynamoDB tables were also pre-created in the tf-setup stage.

backend-cluster.tf & vars-main.tf

As described in previous sections.

---

data-eks-cluster.tf

Get a data resource ("read only") reference for the EKS cluster control plane. Note the use of **data.terraform_remote_state.cluster.xxx** variables.

```
data "aws_eks_cluster" "eks_cluster" {
  name = data.terraform_remote_state.cluster.outputs.cluster-name
}

output "endpoint" {
  value = data.aws_eks_cluster.eks_cluster.endpoint
}

output "ca" {
  value = data.aws_eks_cluster.eks_cluster.certificate_authority[0].data
}

# Only available on Kubernetes version 1.13 and 1.14 clusters created or upgraded on or after September 3, 2019.
output "identity-oidc-issuer" {
  value = data.aws_eks_cluster.eks_cluster.identity[0].oidc[0].issuer
}
```

```
output "cluster-name" {
  value = data.aws_eks_cluster.eks_cluster.name
}
```

user_data.tf

This file will be base64 encoded and passed into the launch template is will:

- Join this node to the cluster **sudo /etc/eks/bootstrap.sh**
  - Note how some parameters for this are passed via Terraform data resources eg. **'${data.aws_eks_cluster.eks_cluster.name}'**
- Install our custom software/configuration - in this case the SSM agent.

```
locals {
  eks-node-private-userdata = <<USERDATA
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash -xe
sudo /etc/eks/bootstrap.sh --apiserver-endpoint '${data.aws_eks_cluster.eks_cluster.endpoint}' --b64-
cluster-ca '${data.aws_eks_cluster.eks_cluster.certificate_authority[0].data}'
'${data.aws_eks_cluster.eks_cluster.name}'
echo "Running custom user data script" > /tmp/me.txt
yum install -y amazon-ssm-agent
echo "yum'd agent" >> /tmp/me.txt
systemctl enable amazon-ssm-agent && systemctl start amazon-ssm-agent
date >> /tmp/me.txt

--==MYBOUNDARY==--
USERDATA
}
```

ssm-param-ami.tf

This gets the latest Amazon Linux 2 AMI for EKS from Systems Manager parameter store.

```
data "aws_ssm_parameter" "eksami" {
 name=format("/aws/service/eks/optimized-ami/%s/amazon-linux-2/recommended/image_id",
data.aws_eks_cluster.eks_cluster.version)
}
```

launch_template.tf

The launch template to use with the EKS managed node, this references:

- Our choice of AMI: **image_id = data.aws_ssm_parameter.eksami.value**.
- Our base64 user data script **user_data = base64encode(local.eks-node-private-userdata)**.

The use of **create_before_destroy=true** is also important to allow us to create new versions of the launch template.

```
resource "aws_launch_template" "lt-ng2" {
 key_name              = "eksworkshop"
 name                  = format("at-lt-%s-ng2", data.aws_eks_cluster.eks_cluster.name)
 tags                  = {}
 image_id              = data.aws_ssm_parameter.eksami.value
 user_data             = base64encode(local.eks-node-private-userdata)
 vpc_security_group_ids  = [data.terraform_remote_state.net.outputs.allnodes-sg]
 tag_specifications {
    resource_type = "instance"
  tags = {
    Name = format("%s-ng2", data.aws_eks_cluster.eks_cluster.name)
    }
  }
 lifecycle {
  create_before_destroy=true
 }
}
```

aws_eks_node_group_ng2.tf

This file contains the options to setup the SPOT instance types.

```
# File generated by aws2tf see https://github.com/aws-samples/aws2tf

resource "aws_eks_node_group" "ng2" {
  #ami_type      = "AL2_x86_64"
  depends_on     = [aws_launch_template.lt-ng2]
  cluster_name   = data.aws_eks_cluster.eks_cluster.name
  disk_size      = 0
  capacity_type = "SPOT"
  instance_types = [
    "m5.large",
    "m5a.large",
    "m5d.large",
    "m5ad.large"
  ]
  labels = {
    "eks/cluster-name"   = data.aws_eks_cluster.eks_cluster.name
    "eks/nodegroup-name" = format("ng2-%s", data.aws_eks_cluster.eks_cluster.name)
  }
  node_group_name = format("ng2-%s", data.aws_eks_cluster.eks_cluster.name)
  node_role_arn   = data.aws_ssm_parameter.nodegroup_role_arn.value
  #release_version = "1.17.11-20201007"
  subnet_ids = [
      data.aws_ssm_parameter.sub-priv1.value,
      data.aws_ssm_parameter.sub-priv2.value,
      data.aws_ssm_parameter.sub-priv3.value
  ]
```

```
 tags = {
  "eks/cluster-name"           = data.aws_eks_cluster.eks_cluster.name
  "eks/nodegroup-name"         = format("ng2-%s", data.aws_eks_cluster.eks_cluster.name)
  "eks/nodegroup-type"         = "managed"
  "eksnet" = "net-main"
 }


 launch_template {
  name    = aws_launch_template.lt-ng2.name
  version = "1"
 }

 scaling_config {
  desired_size = 2
  max_size     = 3
  min_size     = 1
 }

 lifecycle {
  ignore_changes = [scaling_config[0].desired_size]
 }

 timeouts {}
}
```

null_resource.tf

The null resource runs the test.sh and auth.sh script after the creation of the cluster **depends_on = [aws_eks_cluster.cluster]**

```
resource "null_resource" "auth_cluster" {
triggers = {
  always_run = "${timestamp()}"
}
```

```
depends_on = [data.aws_eks_cluster.eks_cluster]
provisioner "local-exec" {
  on_failure  = fail
  interpreter = ["/bin/bash", "-c"]
  command     = <<EOT
    echo -e "\x1B[31m Warning! Checking Authorization ${data.aws_eks_cluster.eks_cluster.name}...should see Server Version: v1.17.xxx \x1B[0m"
    ./auth.sh
    echo "**************************************************************************************"
   EOT
}
}
```

auth.sh

Authorize the local user to the cluster via ~/.kube/config

```
echo "sleep 5 for sync"
sleep 5
rm -f ~/.kube/config
cn=`terraform output cluster-name`
arn=$(aws sts get-caller-identity | jq -r .Arn)
aws eks update-kubeconfig --name $cn
kubectx
echo "kubectl"
kubectl version --short
```

output.tf

Some output variables are defined,but they are not used in this workshop

```
locals {
 config-map-aws-auth = <<CONFIGMAPAWSAUTH
apiVersion: v1
```

```
kind: ConfigMap
metadata:
 name: aws-auth
 namespace: kube-system
data:
 mapRoles: |
  - rolearn: data.terraform_remote_state.iam.outputs.nodegroup_role_arn
   username: system:node:{{EC2PrivateDNSName}}
   groups:
    - system:bootstrappers
    - system:nodes
CONFIGMAPAWSAUTH

 kubeconfig = <<KUBECONFIG
apiVersion: v1
clusters:
- cluster:
  server: aws_eks_cluster.eks-cluster.endpoint
  certificate-authority-data: aws_eks_cluster.eks-cluster.certificate_authority.0.data
 name: kubernetes
contexts:
- context:
  cluster: kubernetes
  user: aws
 name: aws
current-context: aws
kind: Config
preferences: {}
users:
- name: aws
 user:
  exec:
   apiVersion: client.authentication.k8s.io/v1alpha1
   command: aws-iam-authenticator
   args:
    - "token"
```

```
        - "-i"
        - "aws_eks_cluster.eks-cluster.name"
KUBECONFIG
}

output "config-map-aws-auth" {
  value = "local.config-map-aws-auth"
}

output "kubeconfig" {
  value = "local.kubeconfig"
}
```