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.

```
#### User data for worker launch

resource "aws_launch_template" "lt-ng1" {
 depends_on = [null_resource.auth_cluster]
 instance_type        = "t3.small"
 key_name             = "eksworkshop"
 name                 = format("at-lt-%s-ng1", 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-ng1", data.aws_eks_cluster.eks_cluster.name)
    }
```

```
  }
 lifecycle {
  create_before_destroy=true
 }
}
```

aws_eks_node_group__manamieksp_ng1.tf

```
resource "aws_eks_node_group" "ng1" {
 #ami_type      = "AL2_x86_64"
 depends_on     = [aws_launch_template.lt-ng1]
 cluster_name   = data.aws_eks_cluster.eks_cluster.name
 disk_size      = 0
 instance_types = []
 labels = {
  "alpha.eksctl.io/cluster-name"   = data.aws_eks_cluster.eks_cluster.name
  "alpha.eksctl.io/nodegroup-name" = format("ng1-%s", data.aws_eks_cluster.eks_cluster.name)
 }
 node_group_name = format("ng1-%s", data.aws_eks_cluster.eks_cluster.name)
 node_role_arn   = data.terraform_remote_state.iam.outputs.nodegroup_role_arn
 #release_version = "1.17.11-20201007"
 subnet_ids = [
    data.terraform_remote_state.net.outputs.sub-priv1,
    data.terraform_remote_state.net.outputs.sub-priv2,
    data.terraform_remote_state.net.outputs.sub-priv3,
 ]
 tags = {
  "alpha.eksctl.io/cluster-name"             = data.aws_eks_cluster.eks_cluster.name
  "alpha.eksctl.io/eksctl-version"           = "0.29.2"
  "alpha.eksctl.io/nodegroup-name"           = format("ng1-%s", data.aws_eks_cluster.eks_cluster.name)
  "alpha.eksctl.io/nodegroup-type"           = "managed"
  "eksctl.cluster.k8s.io/v1alpha1/cluster-name" = data.aws_eks_cluster.eks_cluster.name
 }
 #version = "1.17"
```

```
 launch_template {
  name    = aws_launch_template.lt-ng1.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" "gen_cluster_auth" {
 triggers = {
  always_run = timestamp()
 }
 depends_on = [aws_eks_node_group.ng1]
 provisioner "local-exec" {
  on_failure  = fail
  when        = create
  interpreter = ["/bin/bash", "-c"]
  command     = <<EOT
    ./c9-auth.sh
    ./auth-cicd.sh
```

```
    echo "********************************************************************************"
  EOT
 }
}
```

c9-auth.sh

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

```
test -n "$C9_PID" && echo C9_PID is "$C9_PID" || "echo C9_PID is not set && exit"
echo "local auth"
sleep 5
c9builder=$(aws cloud9 describe-environment-memberships --environment-id=$C9_PID | jq -r
'.memberships[].userArn')
if echo ${c9builder} | grep -q user; then
    rolearn=${c9builder}
    echo Role ARN: ${rolearn}
elif echo ${c9builder} | grep -q assumed-role; then
    assumedrolename=$(echo ${c9builder} | awk -F/ '{print $(NF-1)}')
    rolearn=$(aws iam get-role --role-name ${assumedrolename} --query Role.Arn --output text)
    echo Role ARN: ${rolearn}
fi
## need to Terraform this ?
cat << EOF > patch.yaml
data:
 mapUsers: |
  - userarn: ${rolearn}
    username: admin
    groups:
     - system:masters
EOF
kubectl get configmap -n kube-system aws-auth -o yaml > aws-auth.yaml
cat patch.yaml >> aws-auth.yaml
kubectl apply -f aws-auth.yaml
```

```
#eksctl create iamidentitymapping --cluster mycluster1 --arn ${rolearn} --group system:masters --username admin
```

auth-cicd.sh

This script authorizes the CodeBuild role to access the EKS cluster by patching the aws-auth configmap

The CodeBuild role **codebuild-eks-cicd-build-app-service-role** is added to the `system:masters` kubernetes group which gives full admin rights to the cluster.

In production environments you would want to scope this down to perhaps a specific namespace using Kubernetes RBAC.

```
test -n "$ACCOUNT_ID" && echo ACCOUNT_ID is "$ACCOUNT_ID" || "echo ACCOUNT_ID is not set && exit"
ROLE="    - rolearn: arn:aws:iam::$ACCOUNT_ID:role/codebuild-eks-cicd-build-app-service-role\n  username: build\n    groups:\n      - system:masters"
#
kubectl get -n kube-system configmap/aws-auth -o yaml | awk "/mapRoles: \|/{print;print \"$ROLE\";next}1" > /tmp/aws-auth-patch.yml
#
kubectl patch configmap/aws-auth -n kube-system --patch "$(cat /tmp/aws-auth-patch.yml)"
```

output.tf

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

Expand to view Terraform code

```
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"
}
```