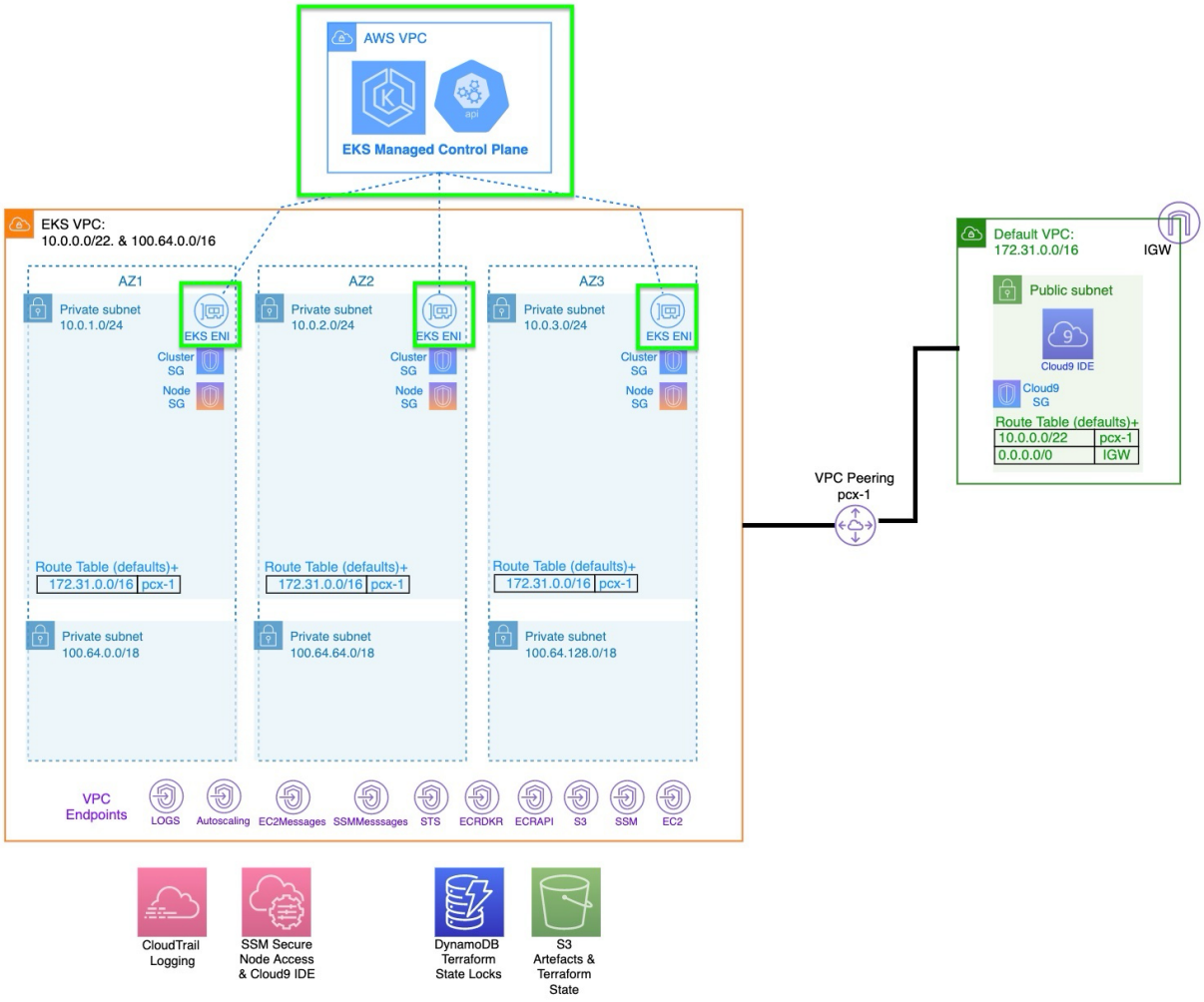
Terraform files explanation

Create the EKS Cluster

Diagram of the components, with the EKS cluster parts we are creating highlighted.



Terraform files and explanation

backend-cluster.tf, vars-main.tf

These files have been pre-created from the gen-backend.sh script in the tf-setup stage. The contents have been explained in previous sections.

aws_eks_cluster__cluster.tf

Create the EKS cluster, note the use of various ssm parameter values.

```
resource "aws_eks_cluster" "cluster" {
 enabled_cluster_log_types = [
  "api",
  "audit",
  "authenticator",
  "controllerManager",
  "scheduler",
 ]
 name = data.aws_ssm_parameter.tf-eks-cluster-name.value

 role_arn = data.aws_ssm_parameter.cluster_service_role_arn.value
 tags     = {}
 version  = var.eks_version

 timeouts {}

 vpc_config {
  endpoint_private_access = true
  endpoint_public_access  = false
  public_access_cidrs = [
   "0.0.0.0/0",
  ]
  security_group_ids = [
   data.aws_ssm_parameter.net-cluster-sg.value,
  ]
  subnet_ids = [
   data.aws_ssm_parameter.sub-priv1.value,
   data.aws_ssm_parameter.sub-priv2.value,
   data.aws_ssm_parameter.sub-priv3.value,
  ]
 }
```

```
  encryption_config {
    provider {
      key_arn = data.aws_kms_key.ekskey.arn
    }
    resources = ["secrets"]
  }
  provisioner "local-exec" {
    command     = "until curl --output /dev/null --insecure --silent ${self.endpoint}/healthz; do sleep 2; done"
    working_dir = path.module
  }

}
```

aws_eks-addons.tf

Cluster Add Ons

```
locals {
  cni_config = file("${path.module}/cni.json")
}


resource "aws_eks_addon" "vpc-cni" {
  depends_on = [aws_eks_cluster.cluster]
  #depends_on     = [null_resource.gen_cluster_auth]
  cluster_name      = data.aws_ssm_parameter.tf-eks-cluster-name.value
  addon_name        = "vpc-cni"
  resolve_conflicts = "OVERWRITE"

  configuration_values = local.cni_config
  addon_version        = "v1.12.1-eksbuild.1"

  preserve = true

}
```

aws_eks_idp.tf

Add the open connect identity provider to the EKS cluster

```
resource "aws_eks_identity_provider_config" "oidc" {
 cluster_name = aws_eks_cluster.cluster.name


 oidc {
  client_id                     = "sts.amazonaws.com"
  identity_provider_config_name = aws_eks_cluster.cluster.name
  issuer_url                    = aws_eks_cluster.cluster.identity.0.oidc.0.issuer
 }
}
```

null_resource.tf

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

```
resource "null_resource" "gen_backend" {
triggers = {
  always_run = "${timestamp()}"
}
depends_on = [aws_eks_cluster.cluster]
provisioner "local-exec" {
  on_failure  = fail
  interpreter = ["/bin/bash", "-c"]
  command     = <<EOT
    echo -e "\x1B[31m Warning! Testing Network Connectivity ${aws_eks_cluster.cluster.name}...should see port 443/tcp open  https\x1B[0m"
    ./test.sh
    echo -e "\x1B[31m Warning! Checking Authorization ${aws_eks_cluster.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

```
rm -f ~/.kube/config
arn=$(aws sts get-caller-identity | jq -r .Arn)
aws eks update-kubeconfig --name $1
kubectx
echo "kubectl"
kubectl version
# pre-set some CNI options before nodes are created
kubectl set env ds aws-node -n kube-system AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
#kubectl set env ds aws-node -n kube-system AWS_VPC_K8S_CNI_EXTERNALSNAT=true
echo "CNI options"
kubectl describe daemonset aws-node -n kube-system | grep CNI_CUSTOM | tr -d ' '
kubectl describe daemonset aws-node -n kube-system | grep AWS_VPC_K8S_CNI_EXTERNALSNAT | tr -d ' '
```