Terraform files explanation

aws.tf

This specifies the Terraform version requirements, the AWS region and profile from variables, and the AWS credentials from a local file if present.

Look at the contents of the file aws.tf - this file is specifying to Terraform:

- Which version of Terraform should be used **required_version = "~> 1.4.2"**.
- Where the AWS, null and external "providers" come from and the version to use.
- And for the AWS provider itself, which region to use and where to get the AWS login credentials.

```
terraform {
 # specify minimum version of Terraform
 required_version = "~> 1.4.2"
 required_providers {
  aws = {
   source = "hashicorp/aws"
   #  Lock version to prevent unexpected problems
   version = "4.63.0"
  }
  null = {
   source  = "hashicorp/null"
   version = "~> 3.1.0"
  }
  external = {
   source  = "hashicorp/external"
   version = "~> 2.1.0"
  }
  kubernetes = {
   source  = "hashicorp/kubernetes"
   version = "2.17.0"
  }
```

```
    helm = {
      source  = "hashicorp/helm"
      version = "~> 2.4.1"
    }
    local = {
      source  = "hashicorp/local"
      version = "~> 2.1.0"
    }

  }
}

# specify local directory for AWS credentials
provider "aws" {
  region                  = var.region
  shared_credentials_files = ["~/.aws/credentials"]
  profile                 = var.profile
}
provider "null" {}
provider "external" {}
```

---

rand.tf

```
# generate a random id used thoughout this build
resource "random_id" "id1" {
  byte_length = 8
}

output "tfid" {
  value = random_id.id1.hex
}
```

---

vars-dynamodb.tf

This file defines some variables with values for the 7x DynamoDB tables. We use 7 tables to help lock the 7x state files that are used for the different stages/sections of our infrastructure build and a string map of the 7 stages in the build.

```
variable "table_name_net" {
 description = "The name of the DynamoDB table. Must be unique in this AWS account."
 type       = string
 default    = "terraform_locks_net"
}

variable "table_name_iam" {
 description = "The name of the DynamoDB table. Must be unique in this AWS account."
 type       = string
 default    = "terraform_locks_iam"
}

variable "table_name_c9net" {
 description = "The name of the DynamoDB table. Must be unique in this AWS account."
 type       = string
 default    = "terraform_locks_c9net"
}

variable "table_name_cicd" {
 description = "The name of the DynamoDB table. Must be unique in this AWS account."
 type       = string
 default    = "terraform_locks_cicd"
}

variable "table_name_cluster" {
 description = "The name of the DynamoDB table. Must be unique in this AWS account."
 type       = string
 default    = "terraform_locks_cluster"
}

variable "table_name_nodeg" {
 description = "The name of the DynamoDB table. Must be unique in this AWS account."
 type       = string
```

```
  default     = "terraform_locks_nodeg"
}

variable "table_name_fargate" {
  description = "The name of the DynamoDB table. Must be unique in this AWS account."
  type        = string
  default     = "terraform_locks_fargate"
}

variable "table_name_sampleapp" {
  description = "The name of the DynamoDB table. Must be unique in this AWS account."
  type        = string
  default     = "terraform_locks_sampleapp"
}

variable "table_name_tf-setup" {
  description = "The name of the DynamoDB table. Must be unique in this AWS account."
  type        = string
  default     = "terraform_locks_tf-setup"
}

variable "stages" {
  type    = list(string)
  default = ["tf-setup", "net", "iam", "c9net", "cluster", "nodeg", "cicd", "sampleapp", "fargate" ]
}

variable "stagecount" {
  type    = number
  default = 9
}
```

---

vars-main.tf

Some other general variables are set in this file: the region, default AWS profile name, the EKS cluster name

```
# TF_VAR_region
variable "region" {
 description = "The name of the AWS Region"
 type       = string
 default    = "eu-west-1"
}

variable "profile" {
 description = "The name of the AWS profile in the credentials file"
 type       = string
 default    = "default"
}

variable "cluster-name" {
 description = "The name of the EKS Cluster"
 type       = string
 default    = "mycluster1"
}

variable "eks_version" {
 type    = string
 default = "1.23"
}

variable "no-output" {
 description = "The name of the EKS Cluster"
 type       = string
 default    = "secret"
 sensitive   = true
}
```

---

kms.tf

This creates a KMS key which is used to encrypt various recources created by Terraform

```
resource "aws_kms_key" "ekskey" {
```

```
 description = format("EKS KMS Key 2 %s", var.cluster-name)
}

output "keyid" {
 value = aws_kms_key.ekskey.key_id
}
```

---

dynamodb-tables.tf

This file specifies that Terraform should create the five dynamoDB tables used to hold the locks for accessing the Terraform state files we will create later in the S3 bucket, Note the **depends_on** statement to ensure the S3 bucket gets created before the DynamoDB table.

Note how this uses the special terraform "count" capability to create (9x) **var.stagecount** different DynamoDB tables. Each table's name is constructed by assembling a string **format(** that contains a fixed value **terraform_locks** and a string value from our string array **%s",var.stages** - indexed by the **[count.index]**

```
resource "aws_dynamodb_table" "terraform_locks" {
 count       = var.stagecount
 depends_on   = [aws_s3_bucket.terraform_state]
 name        = format("terraform_locks_%s", var.stages[count.index])
 billing_mode = "PAY_PER_REQUEST"
 hash_key     = "LockID"
 server_side_encryption {
  enabled     = true
  kms_key_arn = aws_kms_key.ekskey.arn
 }
 attribute {
  name = "LockID"
  type = "S"
 }
 point_in_time_recovery {
  enabled = true
 }
```

```
}
```

---

s3-bucket.tf

This creates an s3 bucket and various bucket options. This bucket is used to store the Terraform state files.

```
resource "aws_s3_bucket" "terraform_state" {

 bucket = format("tf-state-workshop-%s", random_id.id1.hex)

 // This is only here so we can destroy the bucket as part of automated tests. You should not copy this for production
 // usage
 force_destroy = true

 lifecycle {
  ignore_changes = [bucket]
 }

}

resource "aws_s3_bucket_server_side_encryption_configuration" "terraform_state" {
 bucket = aws_s3_bucket.terraform_state.id

 rule {
  bucket_key_enabled = false

  apply_server_side_encryption_by_default {
   sse_algorithm     = "aws:kms"
   kms_master_key_id = aws_kms_key.ekskey.key_id
  }
 }
}
```

```
resource "aws_s3_bucket_versioning" "terraform_state" {
 # Enable versioning so we can see the full revision history of our
 # state files
 bucket = aws_s3_bucket.terraform_state.id
 versioning_configuration {
  status = "Enabled"
 }
}

resource "aws_s3_bucket_public_access_block" "pub_block_state" {
 bucket = aws_s3_bucket.terraform_state.id

 restrict_public_buckets = true
 block_public_acls      = true
 block_public_policy    = true
 ignore_public_acls     = true
}
```

---

null_resource.tf

The **null_resource** type allow us to run local (or remote) commands as part of an infrastructure build. The resource contains a `depends_on` statement which helps both sequence script invocation and prevents any race conditions (the sleep 6) so that resources are created properly before they are used.

This null_resource then calls the **gen-backend.sh** script at the right time.

```
resource "null_resource" "gen_backend" {
 triggers = {
  always_run = timestamp()
 }
 depends_on =
[aws_dynamodb_table.terraform_locks,aws_s3_bucket_server_side_encryption_configuration.terraform_state]
 provisioner "local-exec" {
```

```
    when    = create
    command = = <<EOT
      sleep 6
      ./gen-backend.sh
    EOT
  }
}
```

---

gen-backend.sh

This script generates these terraform files for use in the other 7 sections of the Terraform build of our EKS infrastructure:

- generated/backend-{section}.tf *(For each section this defines where our Terraform state file and DynamoDB lock table is located)*
- generated/remote-{section}.tf *(This allows us to access output variables from other sections, helping to ensure 7x independent infrastructure build tasks can be performed)*
- For the sample application and the optional extra activities - a local state file is configured see **aws.tf** this is very similar to what was used for the Terraform primer lab.

```
#!/bin/bash
cp dot-terraform.rc $HOME/.terraformrc
d=`pwd`
sleep 5
reg=`terraform output -json region | jq -r .[]`
#reg=$(echo "var.region" | terraform console 2> /dev/null | jq -r .)
if [[ -z ${reg} ]] ; then
   echo "no terraform output variables - exiting ....."
   echo "run terraform init/plan/apply in the the init directory first"
else
   echo "region=$reg"
fi
   s3b=`terraform output -json s3_bucket | jq -r .[]`
#
## using terragrunt for the DRY code might be a better approach than the below -
```

```
#
s3b=$(echo "aws_s3_bucket.terraform_state.id" | terraform console 2> /dev/null | jq -r .)

echo $s3b > tmp-buck.txt
echo $reg
mkdir -p generated

#default=["net","iam","c9net","cluster","nodeg","cicd","eks-cidr"]
SECTIONS=('tf-setup' 'net' 'iam' 'c9net' 'cicd' 'cluster' 'nodeg' 'sampleapp' 'fargate')

for section in "${SECTIONS[@]}"
do

  #tabn=`terraform output dynamodb_table_name_$section | tr -d '"'`
  tabn=$(printf "terraform_locks_%s" $section)
  s3b=`terraform output -json s3_bucket | jq -r .[]`
  echo $s3b $tabn

  cd $d

  of=`echo "generated/backend-${section}.tf"`
  #vf=`echo "generated/vars-${section}.tf"`

  # write out the backend config
  printf "" > $of
  printf "terraform {\n" >> $of
  printf "required_version = \"~> 1.3.7\"\n" >> $of
  printf "required_providers {\n" >> $of
  printf "  aws = {\n" >> $of
  printf "   source = \"hashicorp/aws\"\n" >> $of
  printf "#  Lock version to avoid unexpected problems\n" >> $of
  printf "   version = \"4.52.0\"\n" >> $of
  printf "  }\n" >> $of
  printf "  kubernetes = {\n" >> $of
  printf "   source = \"hashicorp/kubernetes\"\n" >> $of
  printf "   version = \"2.17.0\"\n" >> $of
```

```
    printf "  }\n" >> $of
    printf " }\n" >> $of
    printf "backend \"s3\" {\n" >> $of
    printf "bucket = \"%s\"\n"  $s3b >> $of
    printf "key = \"terraform/%s.tfstate\"\n"  $tabn >> $of
    printf "region = \"%s\"\n"  $reg >> $of
    printf "dynamodb_table = \"%s\"\n"  $tabn >> $of
    printf "encrypt = \"true\"\n"   >> $of
    printf "}\n" >> $of
    printf "}\n" >> $of
    ##
    printf "provider \"aws\" {\n" >> $of
    printf "region = var.region\n"  >> $of
    printf "shared_credentials_files = [\"~/.aws/credentials\"]\n" >> $of
    printf "profile = var.profile\n" >> $of
    printf "}\n" >> $of

done
# just for cicd k8s
section="sampleapp"
tabn=$(printf "terraform_locks_sampleapp" $section)
s3b=`terraform output -json s3_bucket | jq -r .[]`
echo $s3b $tabn
cd $d
of=`echo "generated/backend-k8scicd.tf"`
    # write out the backend config
    printf "" > $of
    printf "terraform {\n" >> $of
    printf "required_version = \"~> 1.3.7\"\n" >> $of
    printf "required_providers {\n" >> $of
    printf "  kubernetes = {\n" >> $of
    printf "   source = \"hashicorp/kubernetes\"\n" >> $of
    printf "   version = \"2.17.0\"\n" >> $of
    printf "  }\n" >> $of
    printf " }\n" >> $of
    printf "backend \"s3\" {\n" >> $of
```

```
  printf "bucket = \"%s\"\n"  $s3b >> $of
  printf "key = \"terraform/%s.tfstate\"\n"  $tabn >> $of
  printf "region = \"%s\"\n"  $reg >> $of
  printf "dynamodb_table = \"%s\"\n"  $tabn >> $of
  printf "encrypt = \"true\"\n"   >> $of
  printf "}\n" >> $of
  printf "}\n" >> $of
  ##

cd $d

exit 0
```

---

ssm-params.tf

This is a crucial step for the rest of the build as key variables are stored in the SSM parameter store with the prefic "/workshop/tf-eks".

- The unique ID for our build - see `rand.tf` above.
- The Key id of the KMS key we use to encrypt various resouces - see `kms.tf`
- The Key Arn of the KMS key we use to encrypt various resouces - see `kms.tf`
- The Region we are working in - see `vars-main.tf`
- Our chosen EKS cluster name - see `vars-main.tf`

Sharing key parameter this way will ease the build in the separate stages still to come.

```
resource "aws_ssm_parameter" "tf-eks-id" {
 name       = "/workshop/tf-eks/id"
 description = "The unique id for the workshop"
 type       = "String"
 value      = random_id.id1.hex

 tags = {
  workshop = "tf-eks-workshop"
 }
```

```
}

resource "aws_ssm_parameter" "tf-eks-keyid" {
 name       = "/workshop/tf-eks/keyid"
 description = "The keyid for the workshop"
 type       = "String"
 value      = aws_kms_key.ekskey.key_id

 tags = {
  workshop = "tf-eks-workshop"
 }
}

resource "aws_ssm_parameter" "tf-eks-keyarn" {
 name       = "/workshop/tf-eks/keyarn"
 description = "The keyid for the workshop"
 type       = "String"
 value      = aws_kms_key.ekskey.arn

 tags = {
  workshop = "tf-eks-workshop"
 }
}

resource "aws_ssm_parameter" "tf-eks-region" {
 name       = "/workshop/tf-eks/region"
 description = "The region for the workshop"
 type       = "String"
 value      = var.region

 tags = {
  workshop = "tf-eks-workshop"
 }
}

resource "aws_ssm_parameter" "tf-eks-cluster-name" {
```

```
  name        = "/workshop/tf-eks/cluster-name"
  description = "The EKS cluster name for the workshop"
  type        = "String"
  value       = var.cluster-name

  tags = {
    workshop = "tf-eks-workshop"
  }
}
```