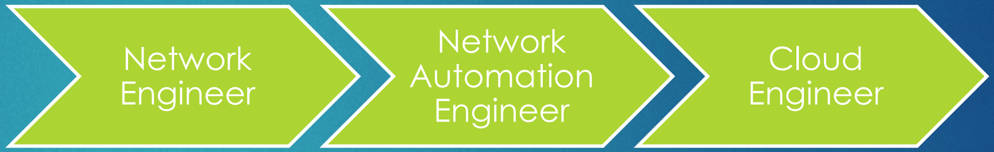
Deploying AWS Infrastructure with Terragrunt

UROŠ BAJŽELJ

About me

Uroš Bajželj
uros@bajzelj.com



Network automation platform

- Components:
  - Reliable single source of truth
  - Workflow executor
  - Monitoring

Nautobot

AWX

Grafana

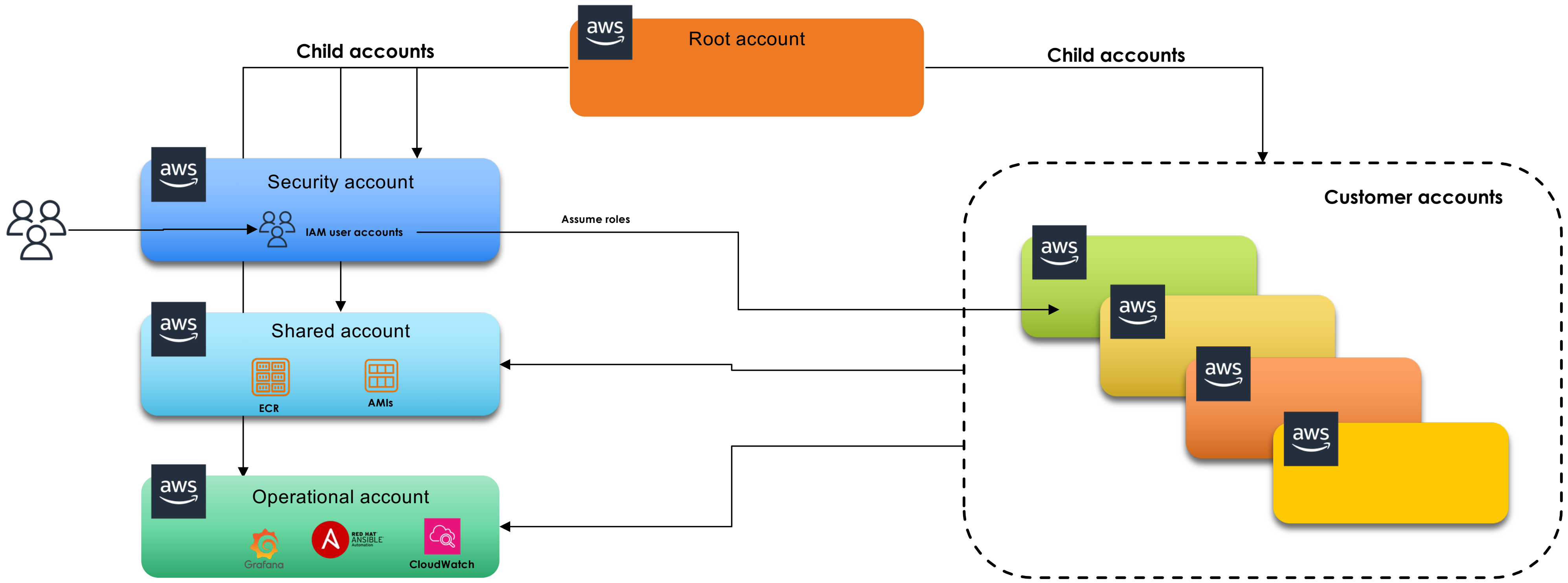
dex

HashiCorp Vault

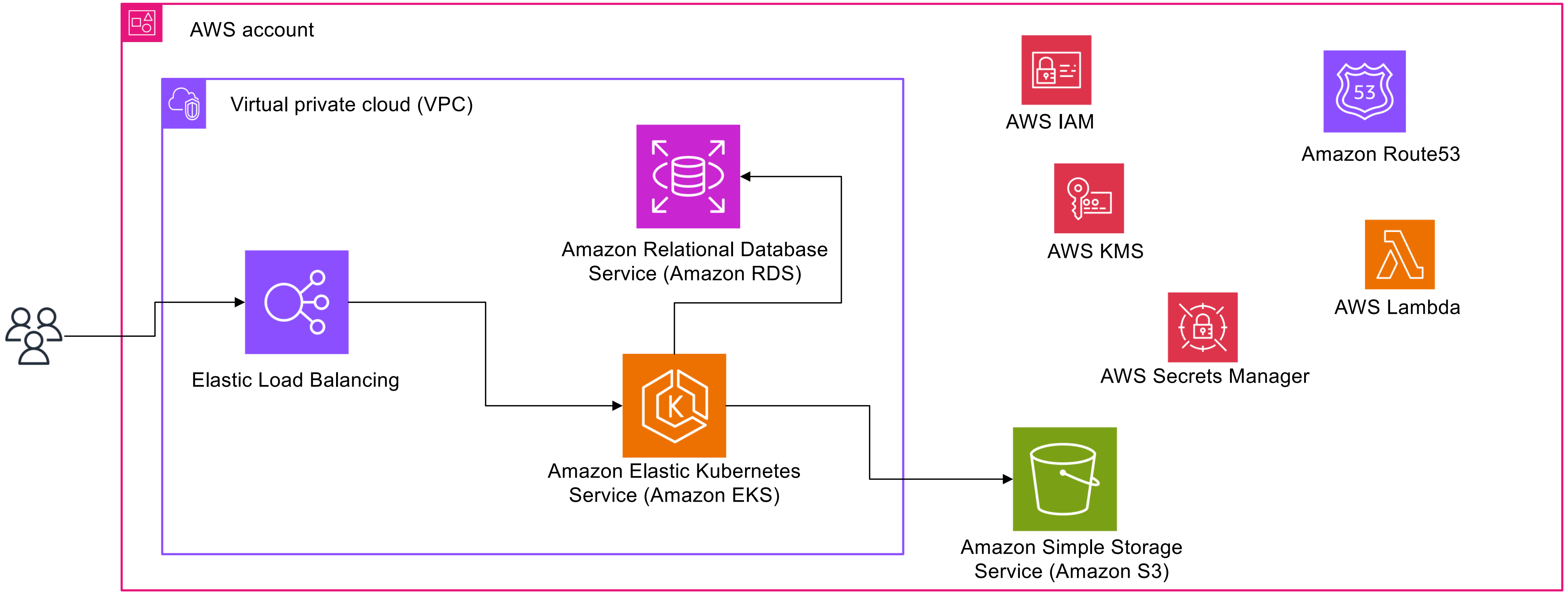
redis

træfik

Reference Architecture



Customer Account


AWS account
Virtual private cloud (VPC)
Amazon Relational Database Service (Amazon RDS)
Elastic Load Balancing
Amazon Elastic Kubernetes Service (Amazon EKS)
AWS IAM
AWS KMS
AWS Secrets Manager
Amazon Route53
AWS Lambda
Amazon Simple Storage Service (Amazon S3)

Key Requirements for Infrastructure Management

- Infrastructure management must support “Infrastructure as a Code”
  - Everything should be stored in Git
  - We want to review changes before deploying them
- The infrastructure must be defined declaratively
  - We want to have the target state defined
- Infrastructure management must be scalable and maintainable
  - Easy to replicate infrastructure for another customer
  - Single location for all infrastructure parameters

What to use for infrastructure management?

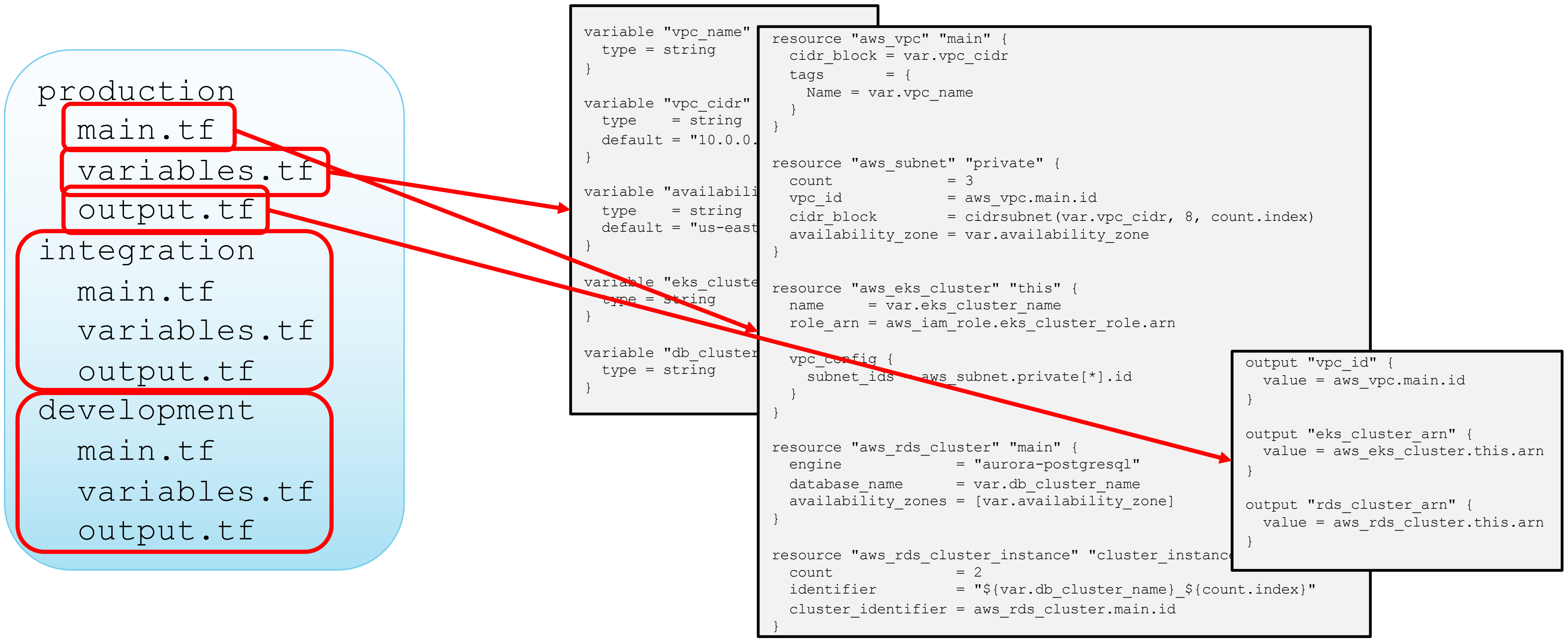
But…

Terragrunt

What is Terragrunt?

- Open-source tool developed by Gruntwork
- A thin wrapper on top of Terraform
- Developed around the DRY (Do Not Repeat Yourself) concept
- Designed to handle complex infrastructure deployments
  - e.g. production, qa, dev
- Split Terraform state into multiple isolated units
- Dependency management

Deploy Infrastructure with Terraform



```
production
  main.tf
  variables.tf
  output.tf
integration
  main.tf
  variables.tf
  output.tf
development
  main.tf
  variables.tf
  output.tf
```

```
variable "vpc_name"
  type = string
}

variable "vpc_cidr"
  type    = string
  default = "10.0.0.
}

variable "availabili
  type    = string
  default = "us-east
}

variable "eks_cluste
  type = string
}

variable "db_cluster
  type = string
}
```

```
resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr
  tags       = {
    Name = var.vpc_name
  }
}

resource "aws_subnet" "private" {
  count             = 3
  vpc_id            = aws_vpc.main.id
  cidr_block        = cidrsubnet(var.vpc_cidr, 8, count.index)
  availability_zone = var.availability_zone
}

resource "aws_eks_cluster" "this" {
  name     = var.eks_cluster_name
  role_arn = aws_iam_role.eks_cluster_role.arn

  vpc_config {
    subnet_ids = aws_subnet.private[*].id
  }
}

resource "aws_rds_cluster" "main" {
  engine             = "aurora-postgresql"
  database_name      = var.db_cluster_name
  availability_zones = [var.availability_zone]
}

resource "aws_rds_cluster_instance" "cluster_instanc
  count              = 2
  identifier         = "${var.db_cluster_name}_${count.index}"
  cluster_identifier = aws_rds_cluster.main.id
}
```

```
output "vpc_id" {
  value = aws_vpc.main.id
}

output "eks_cluster_arn" {
  value = aws_eks_cluster.this.arn
}

output "rds_cluster_arn" {
  value = aws_rds_cluster.this.arn
}
```

Assign Variables

```
production
  main.tf
  variables.tf
  output.tf
  terraform.tfvars
integration
  main.tf
  variables.tf
  output.tf
  terraform.tfvars
development
  main.tf
  variables.tf
  output.tf
  terraform.tfvars
```

```
vpc_name                = "production"
vpc_cidr                = "10.10.0.0/16"
availability_zone       = "us-east-1a"
eks_cluster_name        = "k8s_production"
db_cluster_name         = "db_prod"
```

```
vpc_name                = "integration"
vpc_cidr                = "10.20.0.0/16"
availability_zone       = "us-east-1a"
eks_cluster_name        = "k8s_integration"
db_cluster_name         = "db_integration"
```

```
vpc_name                = "development"
vpc_cidr                = "10.30.0.0/16"
availability_zone       = "us-east-1a"
eks_cluster_name        = "k8s_development"
db_cluster_name         = "db_dev"
```

```
variable "vpc_name" {
  type = string
}

variable "vpc_cidr" {
  type    = string
  default = "10.0.0.0/16"
}

variable "availability_zone" {
  type    = string
  default = "us-east-1a"
}

variable "eks_cluster_name" {
  type = string
}

variable "db_cluster_name" {
  type = string
}
```

Split Terraform Code into Modules

```
resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr
  tags       = {
    Name = var.vpc_name
  }
}

resource "aws_subnet" "private" {
  count             = 3
  vpc_id            = aws_vpc.main.id
  cidr_block        = cidrsubnet(var.vpc_cidr, 8, count.index)
  availability_zone = var.availability_zone
}
resource "aws_eks_cluster" "this" {
  name     = var.eks_cluster_name
  role_arn = aws_iam_role.eks_cluster_role.arn

  vpc_config {
    subnet_ids = aws_subnet.private[*].id
  }
}
resource "aws_rds_cluster" "main" {
  engine             = "aurora-postgresql"
  database_name      = var.db_cluster_name
  availability_zones = [var.availability_zone]
}

resource "aws_rds_cluster_instance" "cluster_instances" {
  count              = 2
  identifier         = "${var.db_cluster_name}_${count.index}"
  cluster_identifier = aws_rds_cluster.main.id
}
```

./modules/vpc

```
resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr
  tags       = {
    Name = var.vpc_name
  }
}

resource "aws_subnet" "private" {
  count             = 3
  vpc_id            = aws_vpc.main.id
  cidr_block        = cidrsubnet(var.vpc_cidr, 8, count.index)
  availability_zone = var.availability_zone
}
```

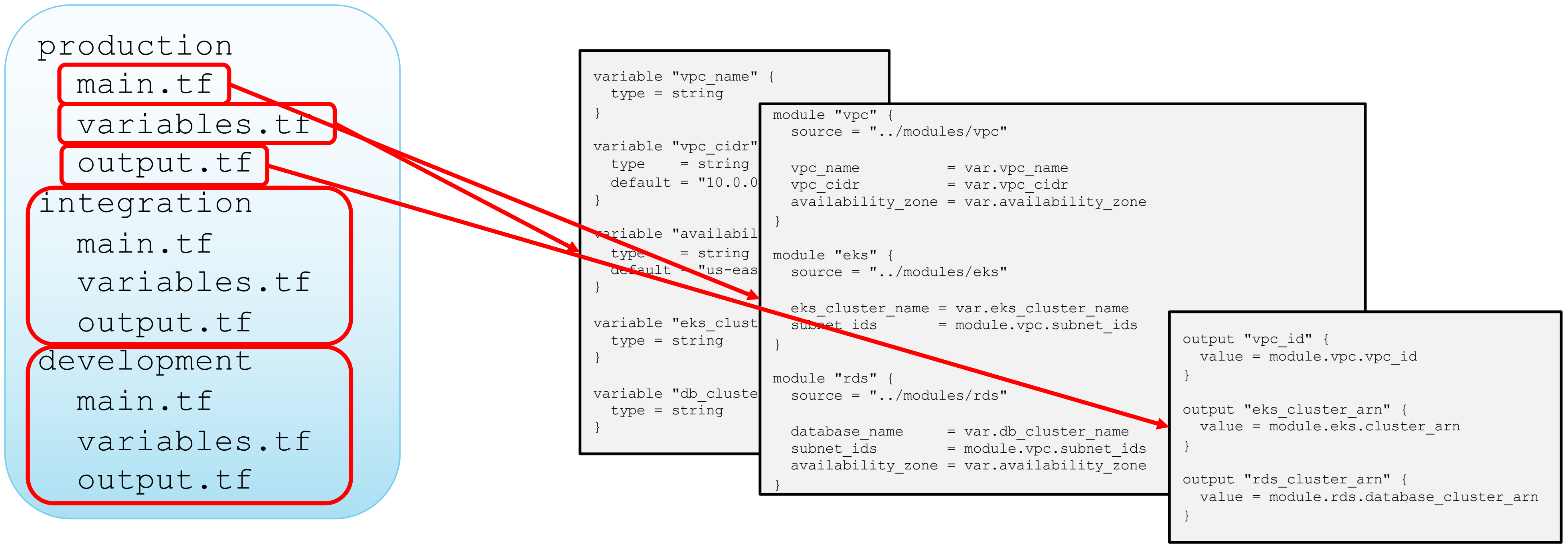
./modules/eks

```
resource "aws_eks_cluster" "this" {
  name     = var.eks_cluster_name
  role_arn = aws_iam_role.eks_cluster_role.arn

  vpc_config {
    subnet_ids = var.subnets_ids
  }
}
```

./modules/rds

```
resource "aws_rds_cluster" "main" {
  engine             = "aurora-postgresql"
  database_name      = var.db_cluster_name
  availability_zones = [var.availability_zone]
}

resource "aws_rds_cluster_instance" "cluster_instances" {
  count              = 2
  identifier         = "${var.db_cluster_name}_${count.index}"
  cluster_identifier = aws_rds_cluster.main.id
}
```

Split Terraform Code into Modules



```
production
   main.tf
   variables.tf
   output.tf
integration
   main.tf
   variables.tf
   output.tf
development
   main.tf
   variables.tf
   output.tf
```

```
variable "vpc_name" {
  type = string
}

variable "vpc_cidr"
  type    = string
  default = "10.0.0
}

variable "availabil
  type    = string
  default = "us-eas
}

variable "eks_clust
  type = string
}

variable "db_cluste
  type = string
}
```

```
module "vpc" {
  source = "../modules/vpc"

  vpc_name          = var.vpc_name
  vpc_cidr          = var.vpc_cidr
  availability_zone = var.availability_zone
}

module "eks" {
  source = "../modules/eks"

  eks_cluster_name = var.eks_cluster_name
  subnet_ids       = module.vpc.subnet_ids
}

module "rds" {
  source = "../modules/rds"

  database_name     = var.db_cluster_name
  subnet_ids        = module.vpc.subnet_ids
  availability_zone = var.availability_zone
}
```

```
output "vpc_id" {
  value = module.vpc.vpc_id
}

output "eks_cluster_arn" {
  value = module.eks.cluster_arn
}

output "rds_cluster_arn" {
  value = module.rds.database_cluster_arn
}
```

Terragrunt Way

```
modules
  vpc
    main.tf
  eks
    main.tf
  rds
    main.tf
environments
  production
    vpc
      terragrunt.hcl
```

```
terraform {
  source = "../../../modules//vpc"
}

inputs = {
  vpc_name          = "production"
  vpc_cidr          = "10.10.0.0/16"
  availability_zone = "us-east-1a"
}
```

Terragrunt Way

```
modules
  vpc
    main.tf
  eks
    main.tf
  rds
    main.tf
environments
  production
    vpc
      terragrunt.hcl
    eks
      terragrunt.hcl
```

```
terraform {
  source = "../../../modules//vpc"
}

inputs = {
  vpc_name          = "
  vpc_cidr          = "
  availability_zone = "
}
```

```
terraform {
  source = "../../../modules//eks"
}

dependency "vpc" {
  config_path = "${get_terragrunt_dir()}/../vpc"
}

inputs = {
  eks_cluster_name = "k8s_production"
  subnet_ids       = dependency.vpc.outputs.subnet_ids
}
```

Terragrunt Way

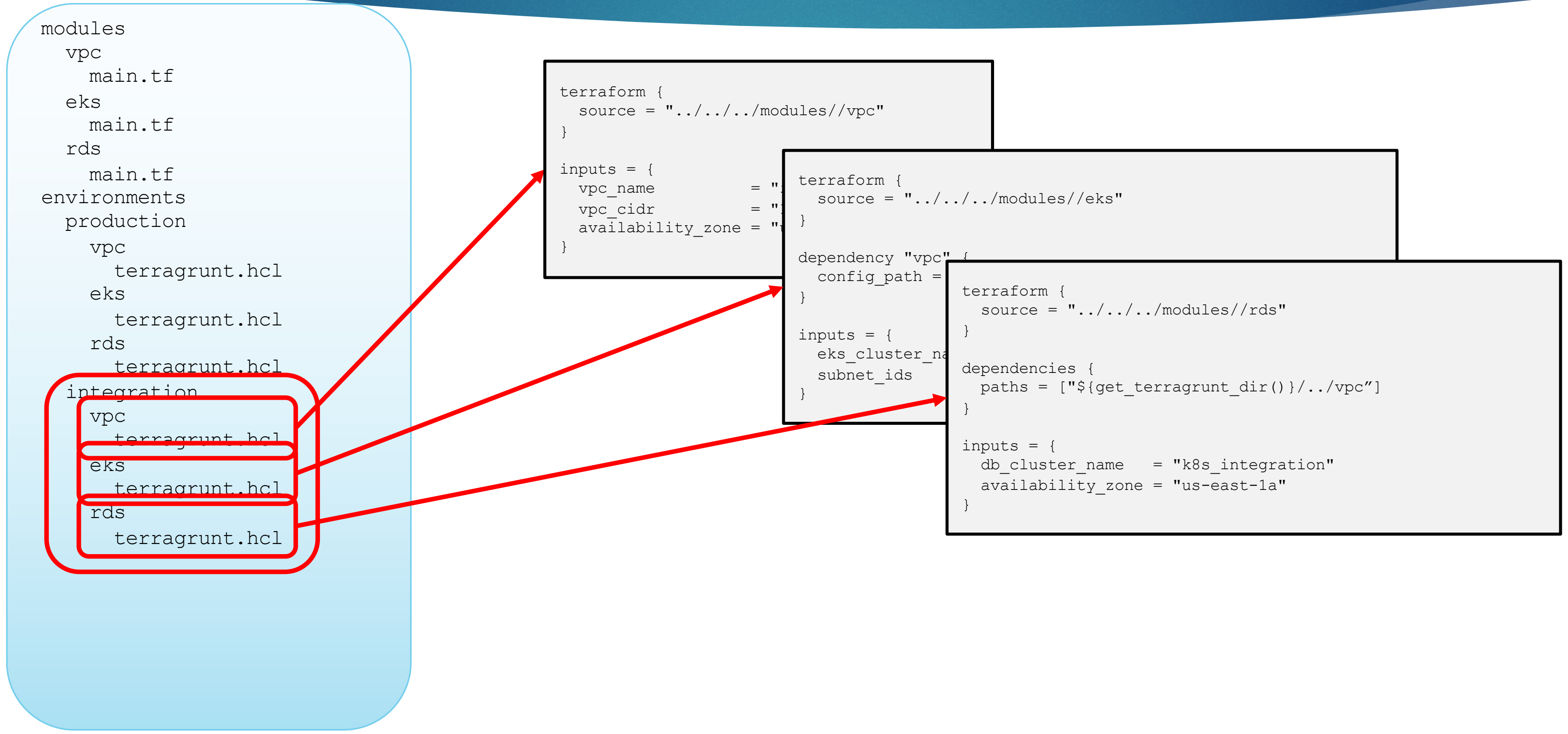
```
modules
  vpc
    main.tf
  eks
    main.tf
  rds
    main.tf
environments
  production
    vpc
      terragrunt.hcl
    eks
      terragrunt.hcl
    rds
      terragrunt.hcl
```

```
terraform {
  source = "../../../modules//vpc"
}

inputs = {
  vpc_name          = "
  vpc_cidr          = "
  availability_zone = "
}
```

```
terraform {
  source = "../../../modules//eks"
}

dependency "vpc" {
  config_path =
}

inputs = {
  eks_cluster_na
  subnet_ids
}
```

```
terraform {
  source = "../../../modules//rds"
}

dependencies {
  paths = ["${get_terragrunt_dir()}/../vpc"]
}

inputs = {
  db_cluster_name   = "k8s_production"
  availability_zone = "us-east-1a"
}
```

Terragrunt Way



```
modules
  vpc
    main.tf
  eks
    main.tf
  rds
    main.tf
environments
  production
    vpc
      terragrunt.hcl
    eks
      terragrunt.hcl
    rds
      terragrunt.hcl
  integration
    vpc
      terragrunt.hcl
    eks
      terragrunt.hcl
    rds
      terragrunt.hcl
```

```
terraform {
  source = "../../../modules//vpc"
}

inputs = {
  vpc_name          = "
  vpc_cidr          = "
  availability_zone = "
}
```

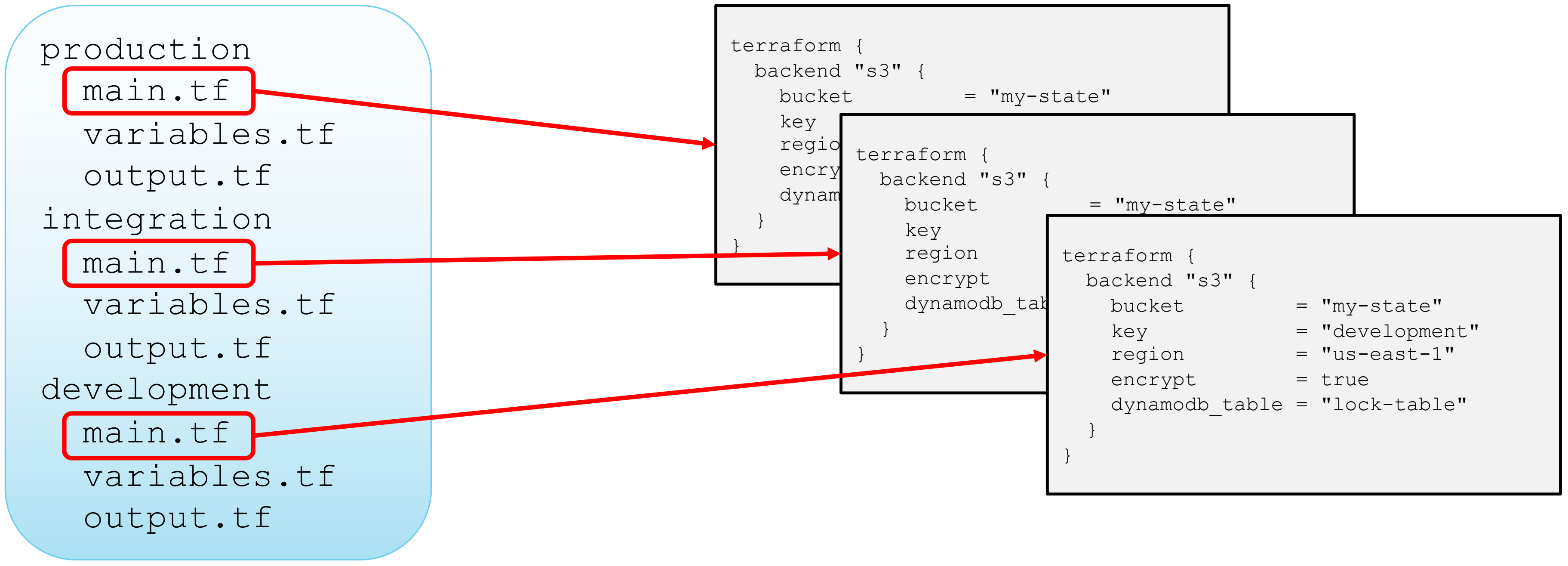
```
terraform {
  source = "../../../modules//eks"
}

dependency "vpc" {
  config_path =
}

inputs = {
  eks_cluster_na
  subnet_ids
}
```

```
terraform {
  source = "../../../modules//rds"
}

dependencies {
  paths = ["${get_terragrunt_dir()}/../vpc”]
}

inputs = {
  db_cluster_name   = "k8s_integration"
  availability_zone = "us-east-1a"
}
```

Terragrunt Way

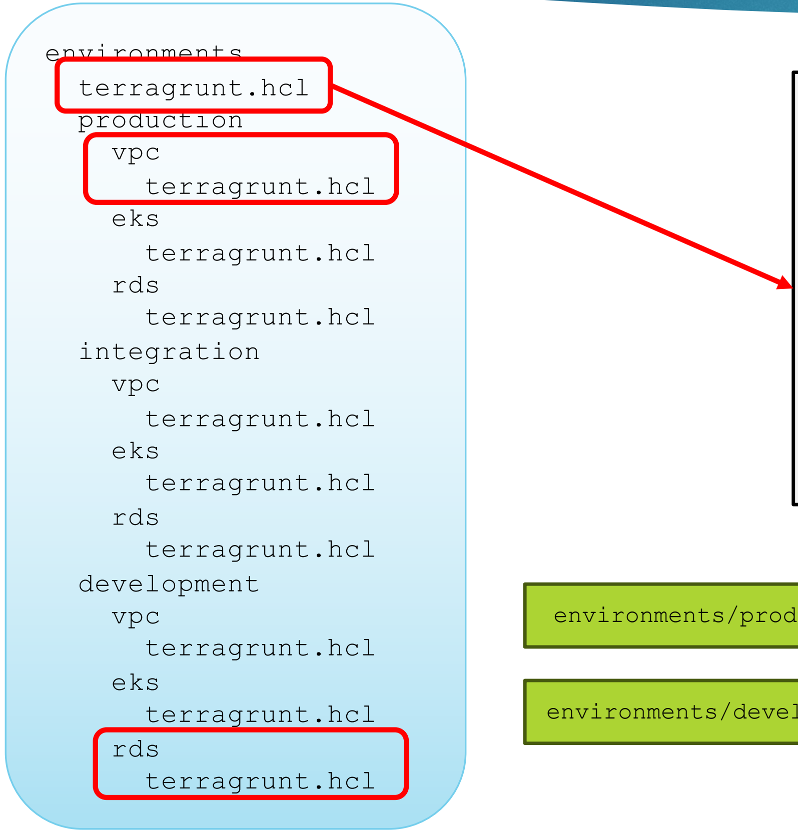
```
modules
  vpc
    main.tf
  eks
    main.tf
  rds
    main.tf
environments
  production
    vpc
      terragrunt.hcl
    eks
      terragrunt.hcl
    rds
      terragrunt.hcl
  integration
    vpc
      terragrunt.hcl
    eks
      terragrunt.hcl
    rds
      terragrunt.hcl
  development
    vpc
      terragrunt.hcl
    eks
      terragrunt.hcl
    rds
      terragrunt.hcl
```

Remote State



```
terraform {
  backend "s3" {
    bucket         = "my-state"
    key            = "development"
    region         = "us-east-1"
    encrypt        = true
    dynamodb_table = "lock-table"
  }
}
```

Terragrunt Remote State



```
environments
  terragrunt.hcl
  production
    vpc
      terragrunt.hcl
    eks
      terragrunt.hcl
    rds
      terragrunt.hcl
  integration
    vpc
      terragrunt.hcl
    eks
      terragrunt.hcl
    rds
      terragrunt.hcl
  development
    vpc
      terragrunt.hcl
    eks
      terragrunt.hcl
    rds
      terragrunt.hcl
```

```
remote_state {
  backend = "s3"
  generate = {
    path      = "backend.tf"
    if_exists = "overwrite"
  }
  config = {
    bucket         = "my-state"
    key            = "${path_relative_to_include()}/terraform.tfstate"
    region         = "us-east-1"
    encrypt        = true
    dynamodb_table = "my-lock-table"
  }
}
```

environments/production/vpc/terragrunt.hcl → environments/production/vpc/terraform.tfstate

environments/development/rds/terragrunt.hcl → environments/development/rds/terraform.tfstate

IT'S
DEMO TIME
makeameme.org

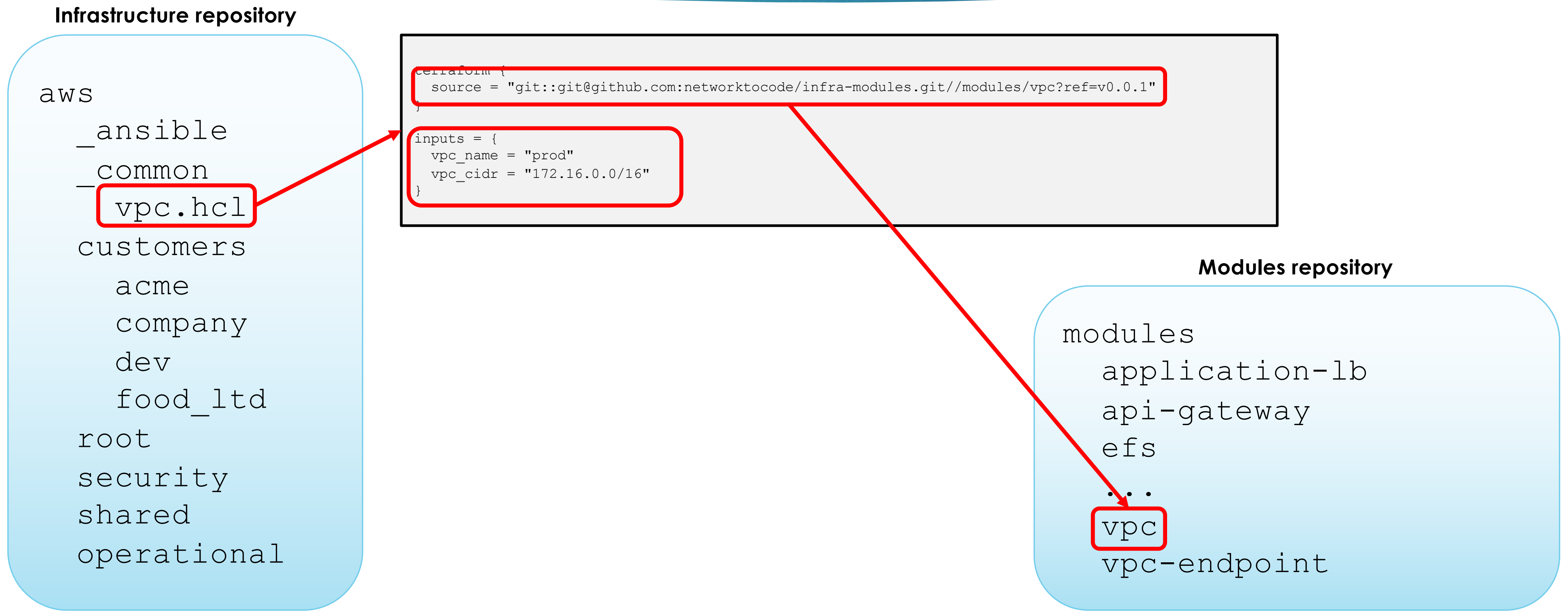
Modules Git Repository

```
modules
   application-lb
   api-gateway
   efs
   eks-control-plane
   eks-nodes
   k8s-subnets
   lambda-function
   rds
   route53-public
   route53-private
   s3-bucket
   vpc
   vpc-endpoint
```

Infrastructure Git Repository

```
aws
  _ansible
  _common
  customers
    acme
    company
    dev
    food_ltd
  root
  security
  shared
  operational
```

Defining Default Values



Infrastructure repository

```
aws
  _ansible
  _common
    vpc.hcl
  customers
    acme
    company
    dev
    food_ltd
  root
  security
  shared
  operational
```

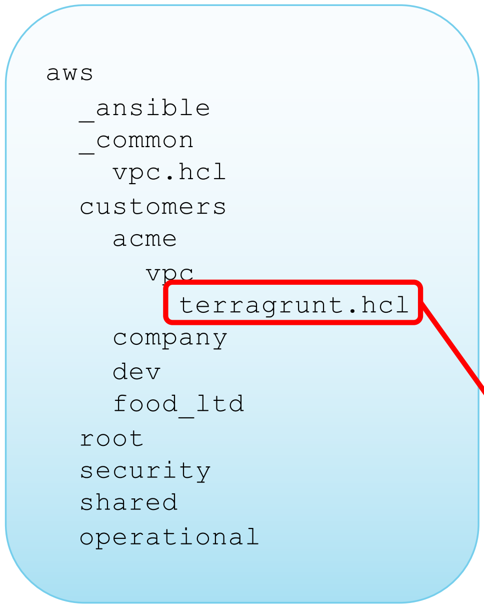
```
terraform {
  source = "git::git@github.com:networktocode/infra-modules.git//modules/vpc?ref=v0.0.1"
}

inputs = {
  vpc_name = "prod"
  vpc_cidr = "172.16.0.0/16"
}
```

Modules repository

```
modules
  application-lb
  api-gateway
  efs
  ...
  vpc
  vpc-endpoint
```

Initializing Module

Infrastructure repository



```
aws
  _ansible
  _common
    vpc.hcl
  customers
    acme
      vpc
        terragrunt.hcl
    company
    dev
    food_ltd
  root
  security
  shared
  operational
```

```
terraform {
  source = "git::git@github.com:networktocode/infra-modules.git//modules/vpc?ref=v0.0.1"
}

inputs = {
  vpc_name = "prod"
  vpc_cidr = "172.16.0.0/16"
}
```

```
include root {
  path = find_in_parent_folders()
}

include "common" {
  path = ”${dirname(find_in_parent_folders())/../../_common/vpc.hcl"
}
```

Override Default Values

Infrastructure repository

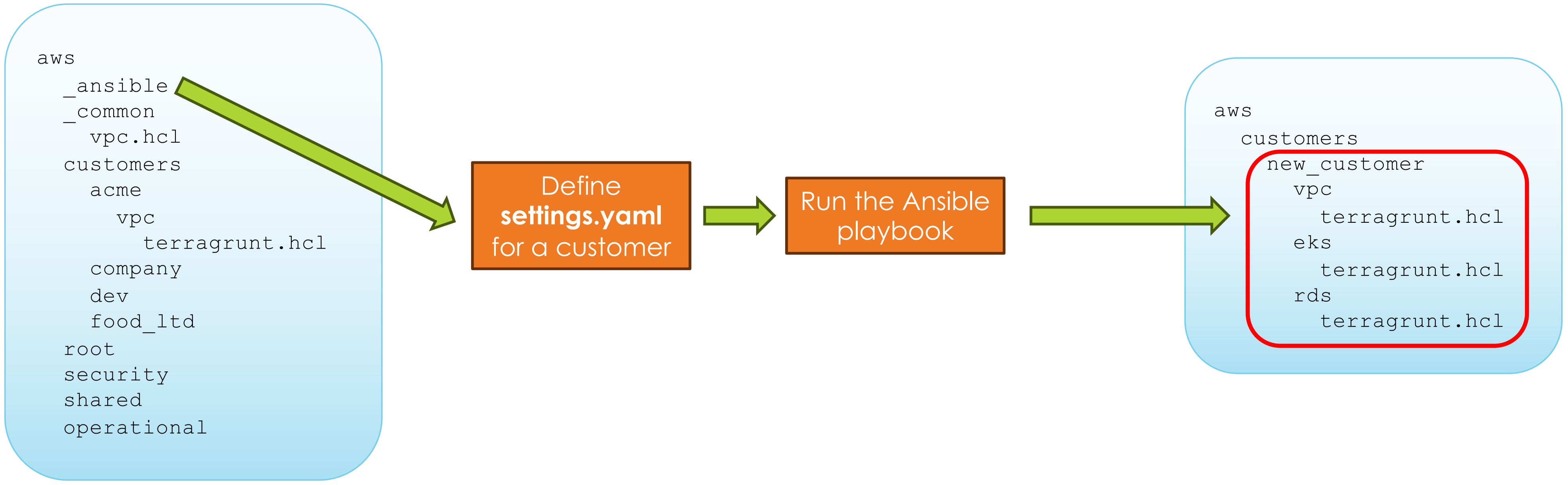
```
aws
  _ansible
  _common
    vpc.hcl
  customers
    acme
      vpc
        terragrunt.hcl
    company
    dev
    food_ltd
  root
  security
  shared
  operational
```

```
terraform {
  source = "git::git@github.com:networktocode/infra-modules.git//modules/vpc?ref=v0.0.1"
}

inputs = {
  vpc_name = "prod"
  vpc_cidr = "172.16.0.0/16"
}
```

```
include root {
  path = find_in_parent_folders()
}

include "common" {
  path = ”${dirname(find_in_parent_folders())/../../_common/vpc.hcl"
}

inputs = {
  vpc_cidr = "192.168.0.0/16"
}
```

Generate Customer Configuration

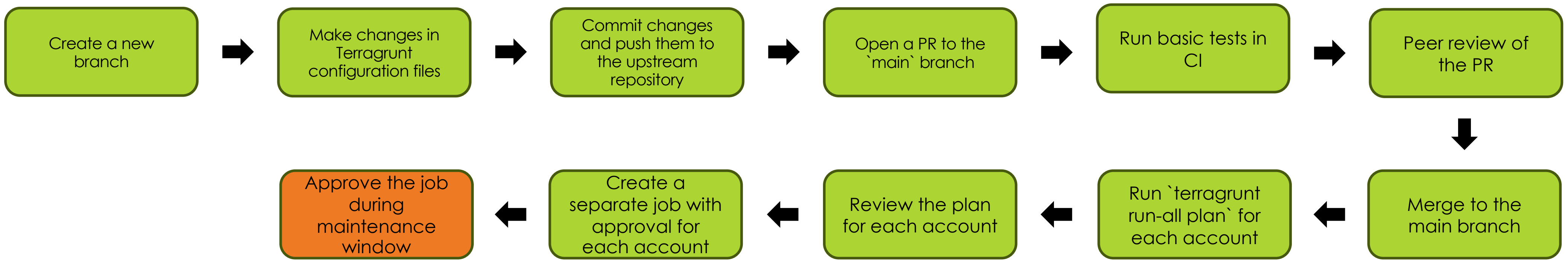
Infrastructure repository



```
aws
  _ansible
  _common
    vpc.hcl
  customers
    acme
      vpc
        terragrunt.hcl
    company
    dev
    food_ltd
  root
  security
  shared
  operational
```

```
aws
  customers
    new_customer
      vpc
        terragrunt.hcl
      eks
        terragrunt.hcl
      rds
        terragrunt.hcl
```

Deploy New Configuration



Create a new branch
Make changes in Terragrunt configuration files
Commit changes and push them to the upstream repository
Open a PR to the `main` branch
Run basic tests in CI
Peer review of the PR
Merge to the main branch
Run `terragrunt run-all plan` for each account
Review the plan for each account
Create a separate job with approval for each account
Approve the job during maintenance window

Thank you