Using Variables in Deployments

Working with Variables

- Name, type, default
- Multiple sources
  - File, environment variable, var option
- Overriding variables and precedence
  - Environment, file, command line
- Select values based on environment
- Split Terraform configuration file

Variables Examples

```
#Specify default variable and type

variable "environment_name" {

type = string
default = "development"

}

#Specify variable in file

environment_name = "uat"

#Specify variable in-line

terraform plan –var
'environment_name=production'
```

Variables Examples

```hcl
#Create variable map
variable "cidr" { type = map(string) default = {

development = "10.0.0.0/16"

uat = "10.1.0.0/16"

production = "10.2.0.0/16"
} }
#Use map based on environment
cidr_block = lookup(var.cidr,var.environment_name)
```
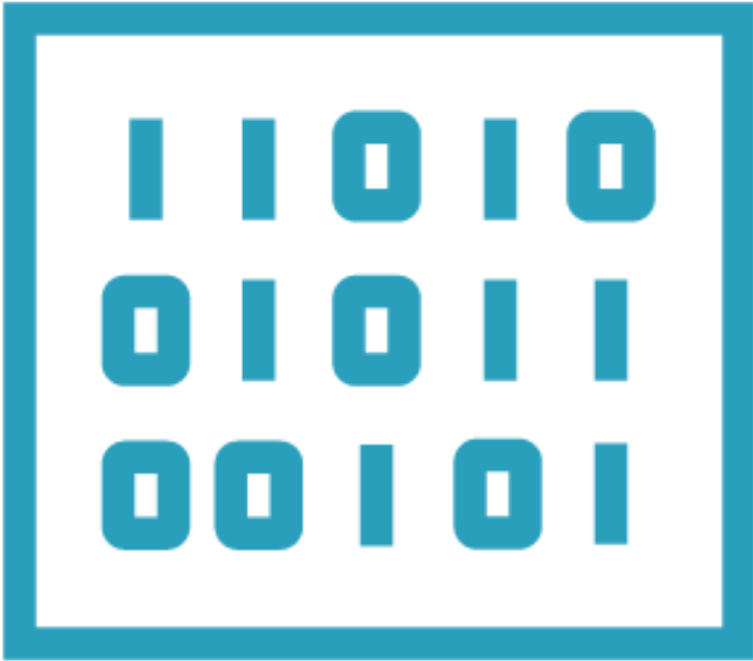
Multiple Environments

Commonality and differences

Abstractions and reuse

Production access

Using workspaces

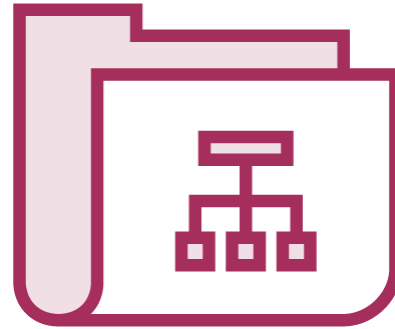# Multiple Environment Decisions

State management

Variables data

Credentials management

Complexity and overhead

State File Example

main_config.tf
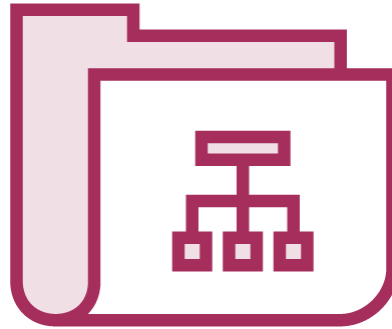**common.tfvars**

dev.state
dev.tfvars

uat.state
uat.tfvars

prod.state
prod.tfvars

```
terraform plan –state=".\dev\dev.state"
  -var-file="common.tfvars"
  -var-file=".\dev\dev.tfvars"
```

Workspace Example

**main_config.tf
terraform.tfvars**

**terraform.
tfstate.d**
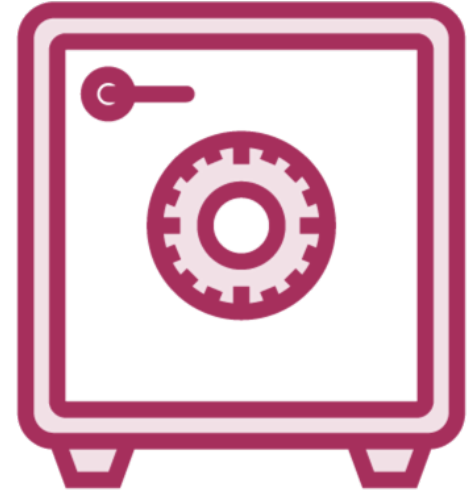
terraform workspace new dev
terraform plan

Secrets Options

Variables file

Environment variables

Secrets management

Environment Variables

#AWS Environment Variables

AWS_ACCESS_KEY_ID

AWS_SECRET_ACCESS_KEY

AWS_SHARED_CREDENTIALS_FILE
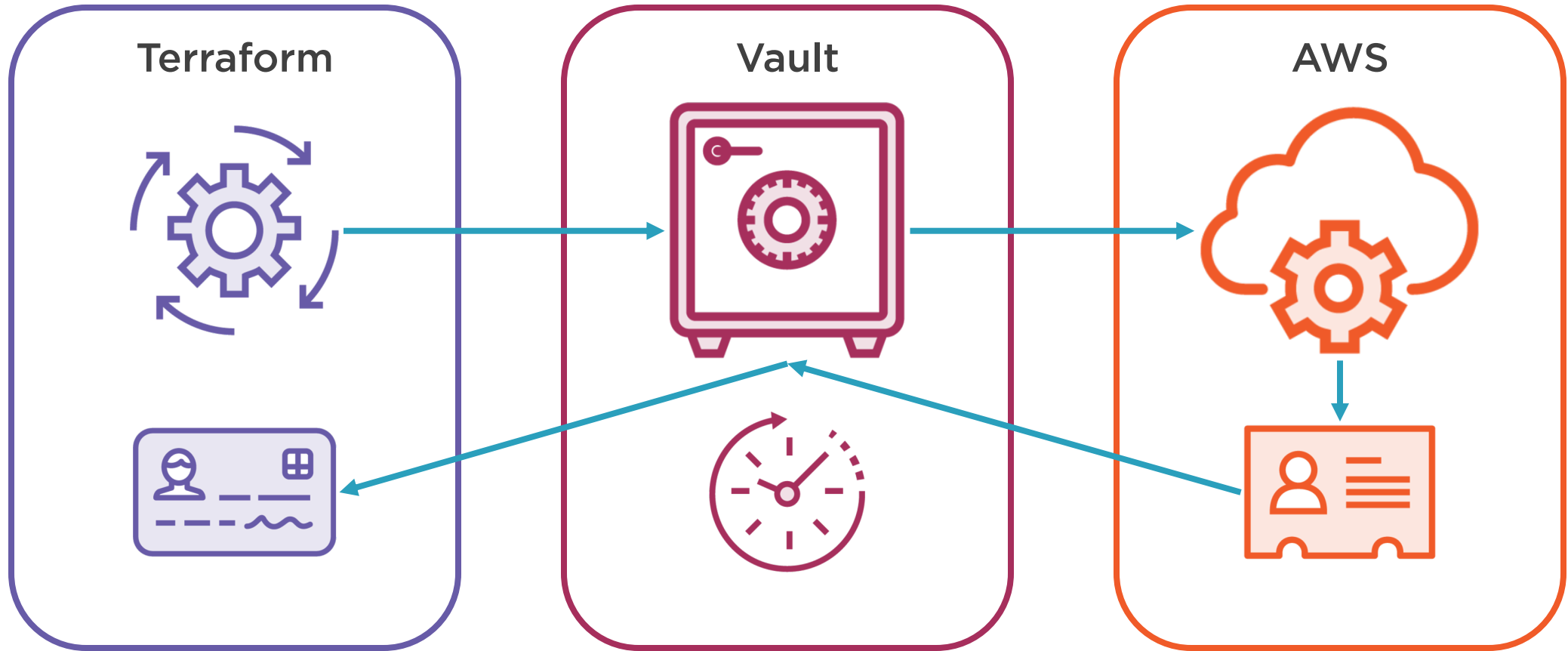
AWS_PROFILE

#PowerShell

$env:AWS_ACCESS_KEY_ID = "AASAS9283708FDKJ"

#Linux

export AWS_ACCESS_KEY_ID="AASAS9283708FDKJ"

Vault Example

Summary

Variables, more than meets the eye

Multiple states, one configuration

Storing secrets securely

Coming up

- Investigating modules

- Abstracting common components