Using Variables in Deployments

Ned Bellavance
MICROSOFT MVP, CLOUD AND DATACENTER MANAGEMENT
@ned1313 www.nedinthecloud.com

Overview

Dev, QA, production and more

Variables and multiple states

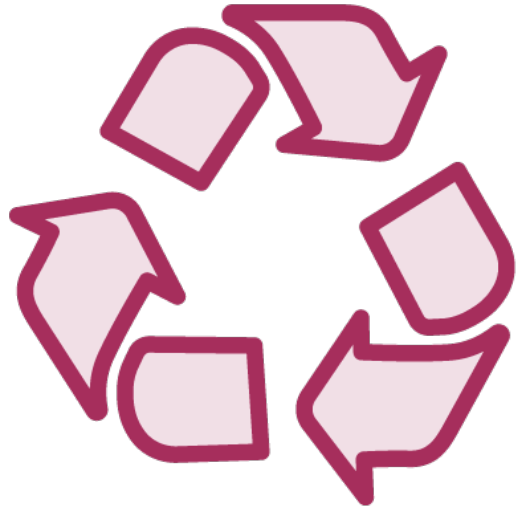
Store secrets in a safe place

Automating Infrastructure Deployment

Provisioning resources

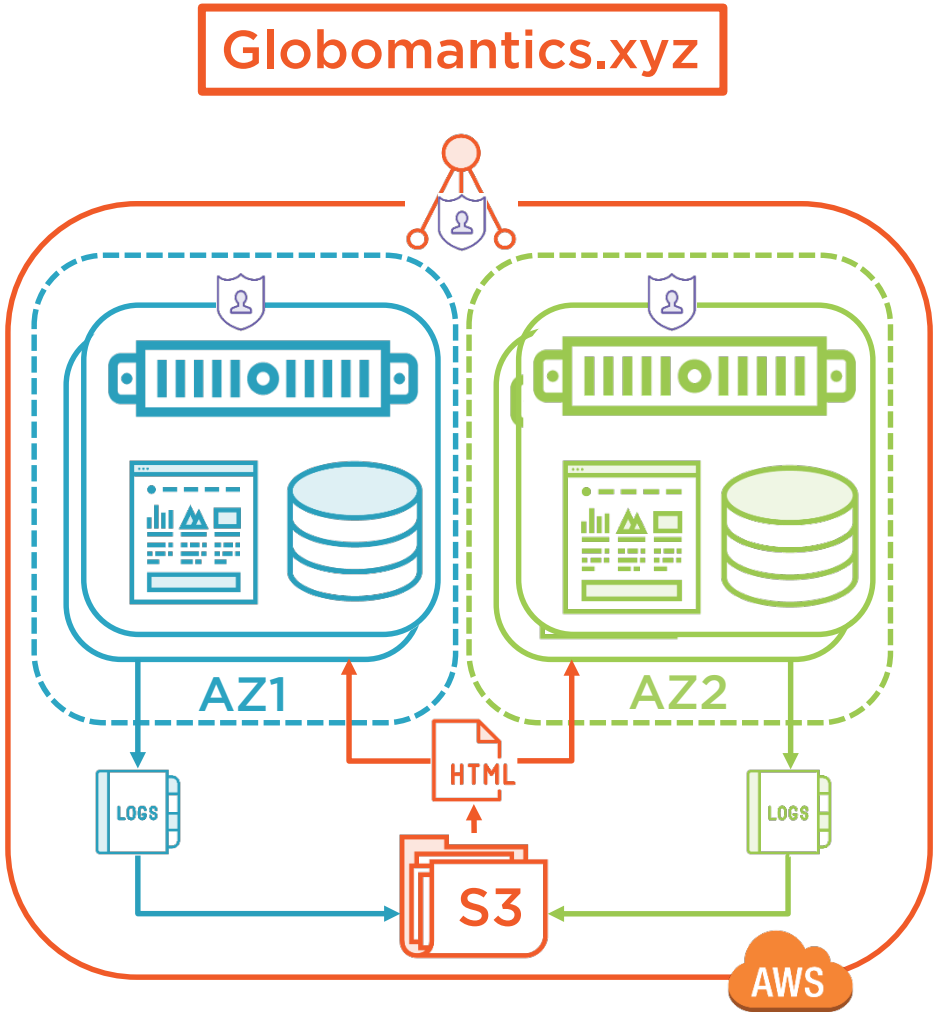
Planning updates

Using source control

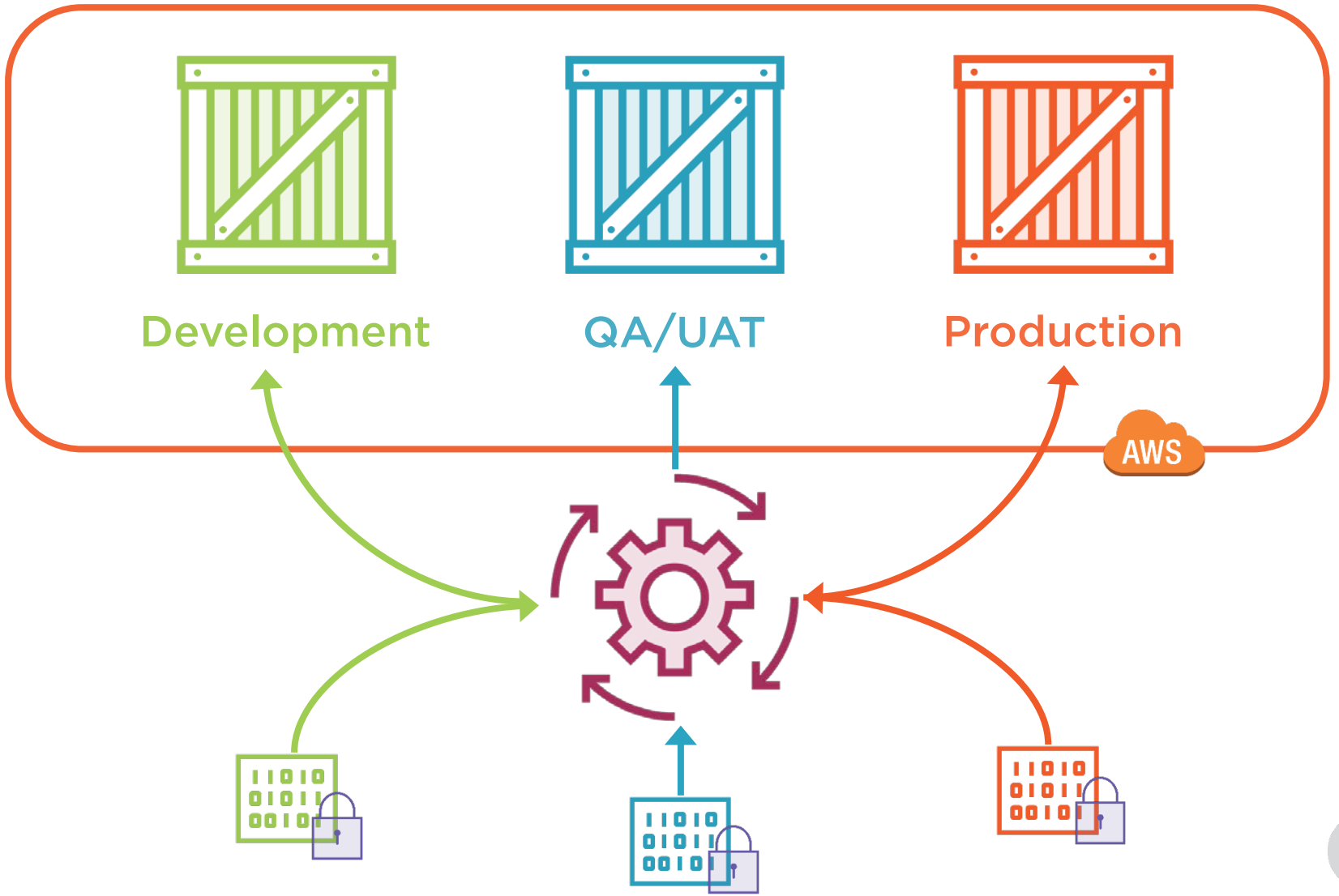


Reusing templates

The Scenario



The Scenario



Working with Variables

Separating variables

Overriding variables and precedence

Select values based on environment

Using conditionals

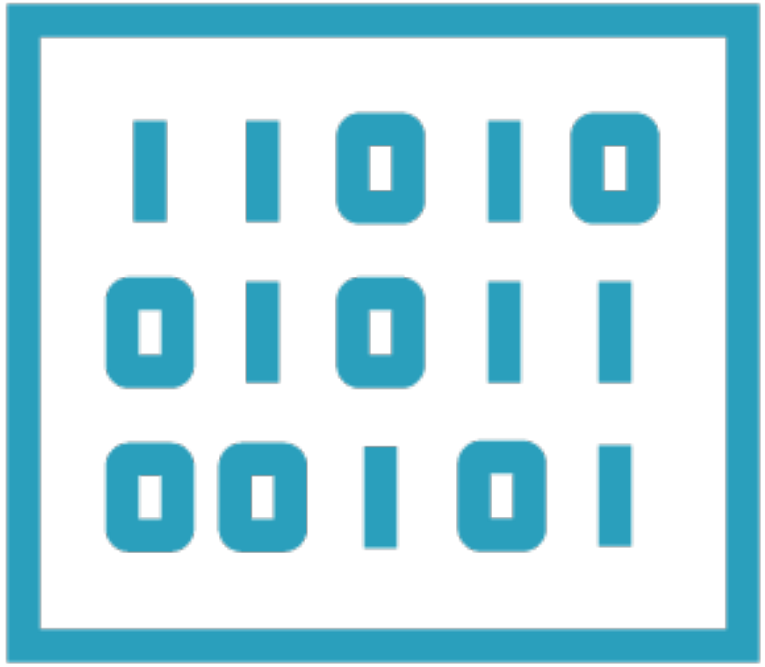
Variables Examples

```
#Specify default variable
variable environment_name {
 default = “development”
}

#Specify variable in file
environment_name = “uat”

#Specify variable in-line
terraform plan –var ‘environment_name=production’
```

Variables Examples

```
#Create variable map
variable cidr {
 type="map"
 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



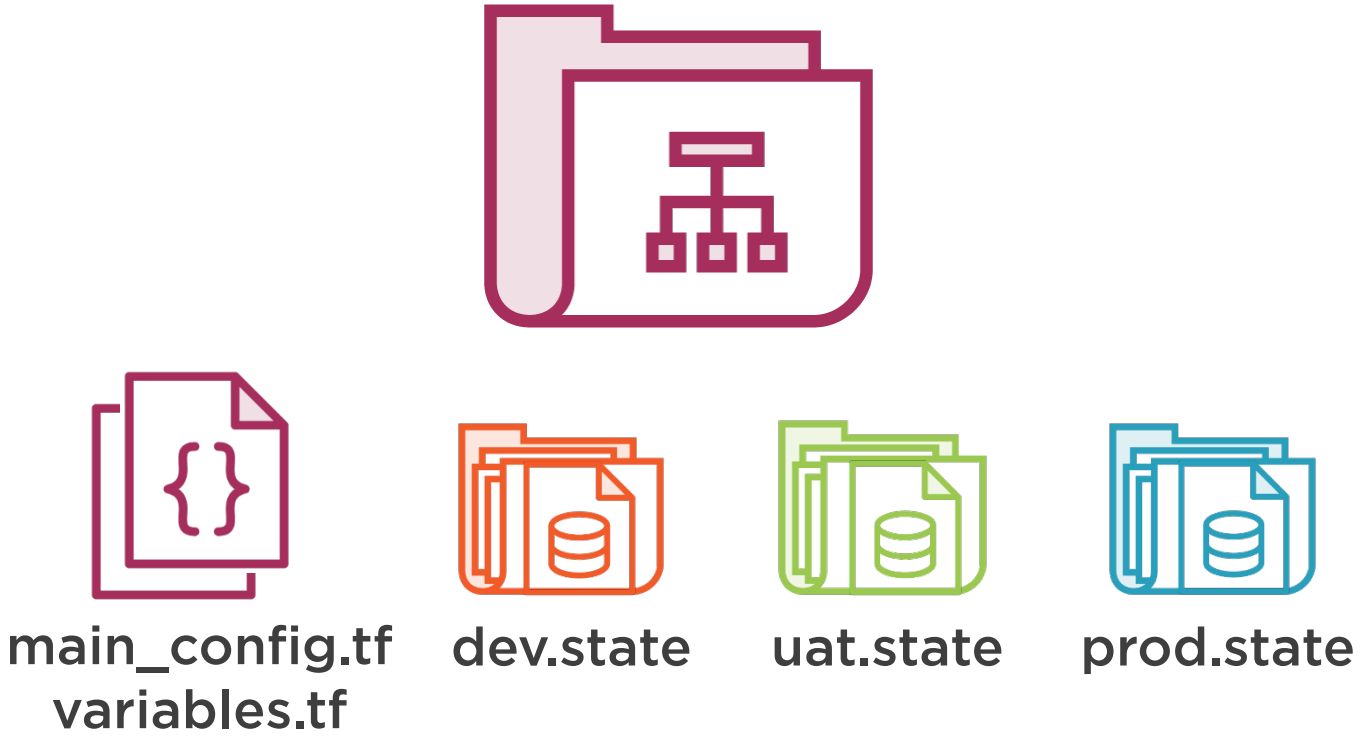
State file commands

State file storage

Folder structure

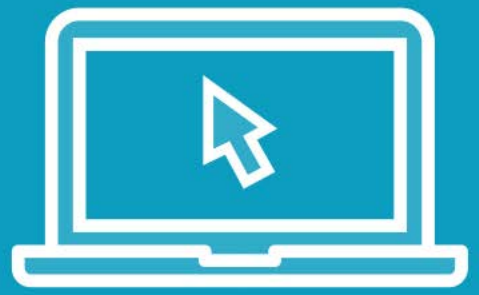
Common patterns

State File Example




```
C:\>terraform apply –state=“.\development\dev.state” `
  –var=“environment_name=development”
```

Demo



Examine the Terraform file

Deploy the configuration

Review the results

Play along!

- AWS account
- Azure subscription
- DNS domain
- Terraform software (terraform.io)
- Demo files

Some of the resources deployed in AWS and Azure may cost **money**. You've been warned.

Summary

Variables, more than meets the eye

Multiple states, one configuration

Storing secrets

Coming up

- Investigating modules
- Abstracting common components
- Code reuse through modules