Best Practices with Terraform Scripts

Robert Rozas Navarro
Premier Field Engineer
Apps Domain

Microsoft

Agenda

1. Best Practices
2. Hands on Lab 2
3. Q&A

Terraform Best Practices

Microsoft

Best Practice 1

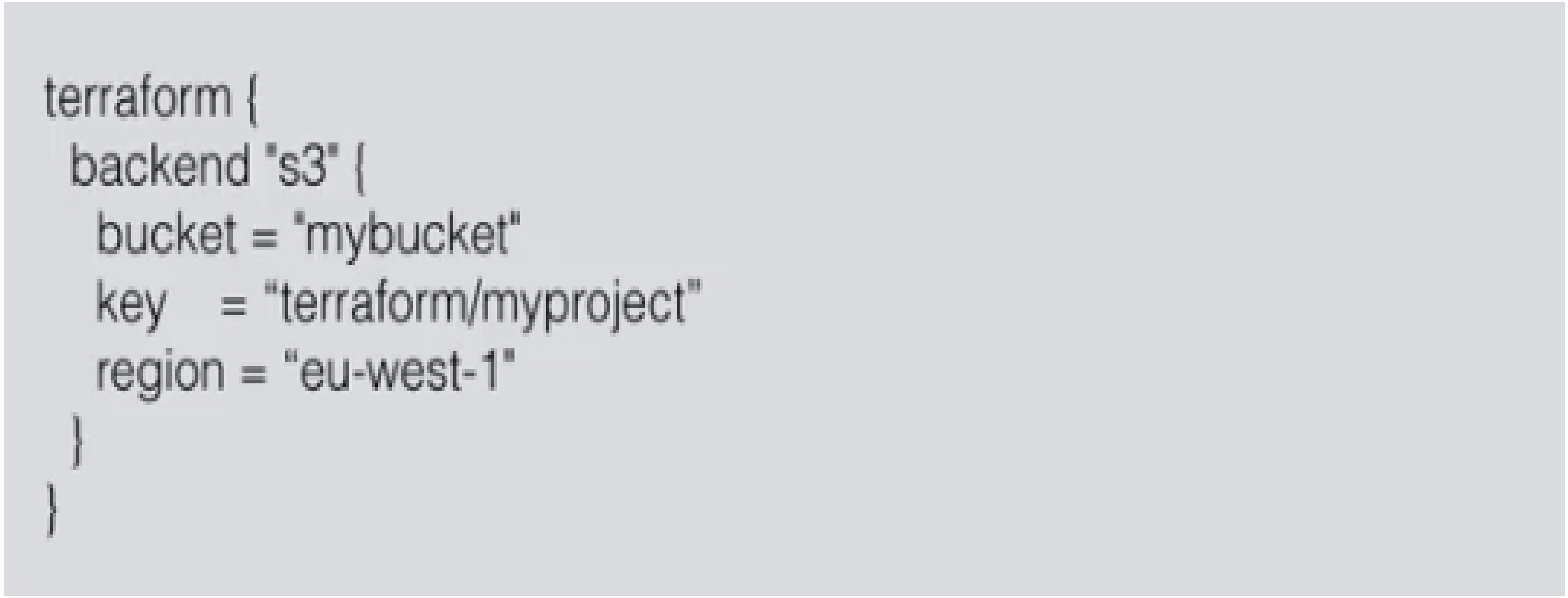
- Use remote backend
- Your laptop is no place for your infrastructure source of truth
- Use **data sources** and **terraform_remote_state** specifically as a glue between infrastructure modules within composition
- Managing a **tfstate** file in git is a nightmare
- Later when infrastructure layers starts to grow in any direction (number of dependencies or resources)

Best Practice 1 (..Continued..)

- Using the backend functionality has definitely benefits:
- Working in a team: it allows for collaboration, the remote state will always be available for the whole team
- The state file is note stored locally. Possible sensitive information is now only stored in the remote state
- Some backends will enable remote operations. The terraform apply will then run completely remote. These are called the enhanced backends

Best Practice 1 (..Continued..)

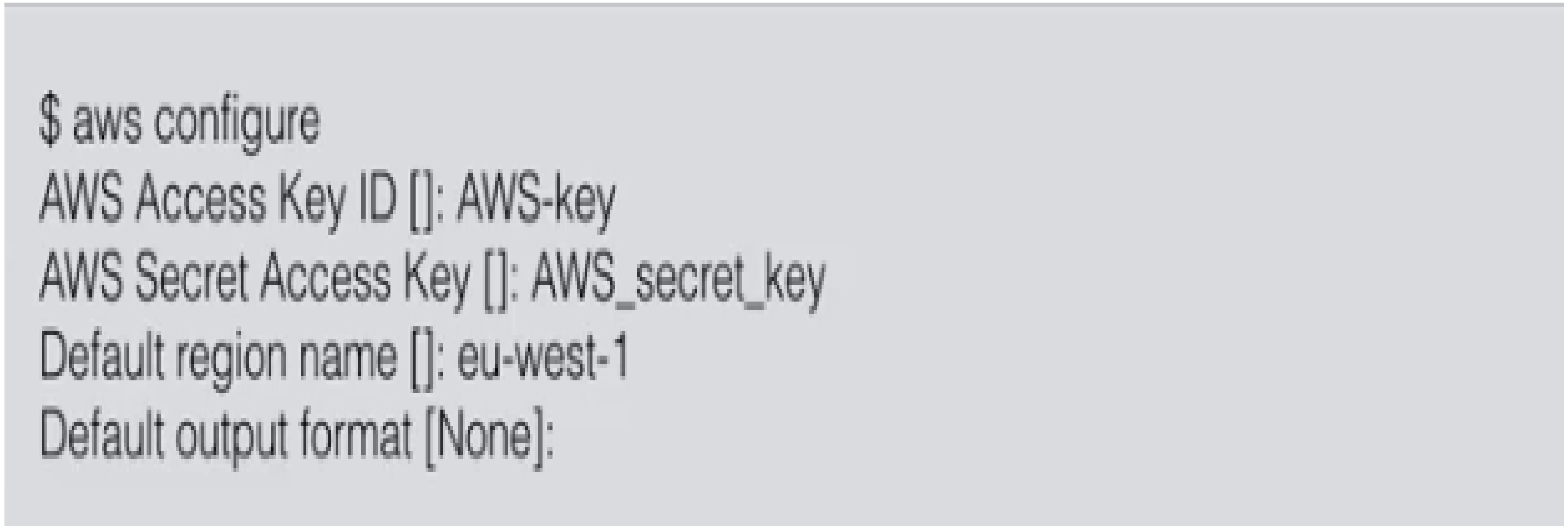
- You can also store your state in S3:



```
terraform {
  backend "s3" {
    bucket = "mybucket"
    key    = "terraform/myproject"
    region = "eu-west-1"
  }
}
```

Best Practice 1 (..Continued..)

- When using an S3 remote state, it's best to configure the AWS credentials:



```
$ aws configure
AWS Access Key ID []: AWS-key
AWS Secret Access Key []: AWS_secret_key
Default region name []: eu-west-1
Default output format [None]:
```

Best Practice 2

- Manage Terraform, AWS provider and modules version
- While individual resources are like atoms in the infrastructure, resource modules are molecules. Module is a smallest versioned and shareable unit.
- Examples and Terraform modules should contain documentation explaining features and how to use them.
- Infrastructure modules and compositions should persist their state in a remote location which can be reached by others in a controllable way

Best Practice 2 (..Continued..)

- You can use modules to make your terraform project more organized
- Infrastructure modules and compositions should persist their state in a remote location which can be reached by others in a controllable way
- Use third party modules (i.e from github)
- Allows you to reuse parts of your code

Best Practice 2 (..Continued..)

Use a module from git

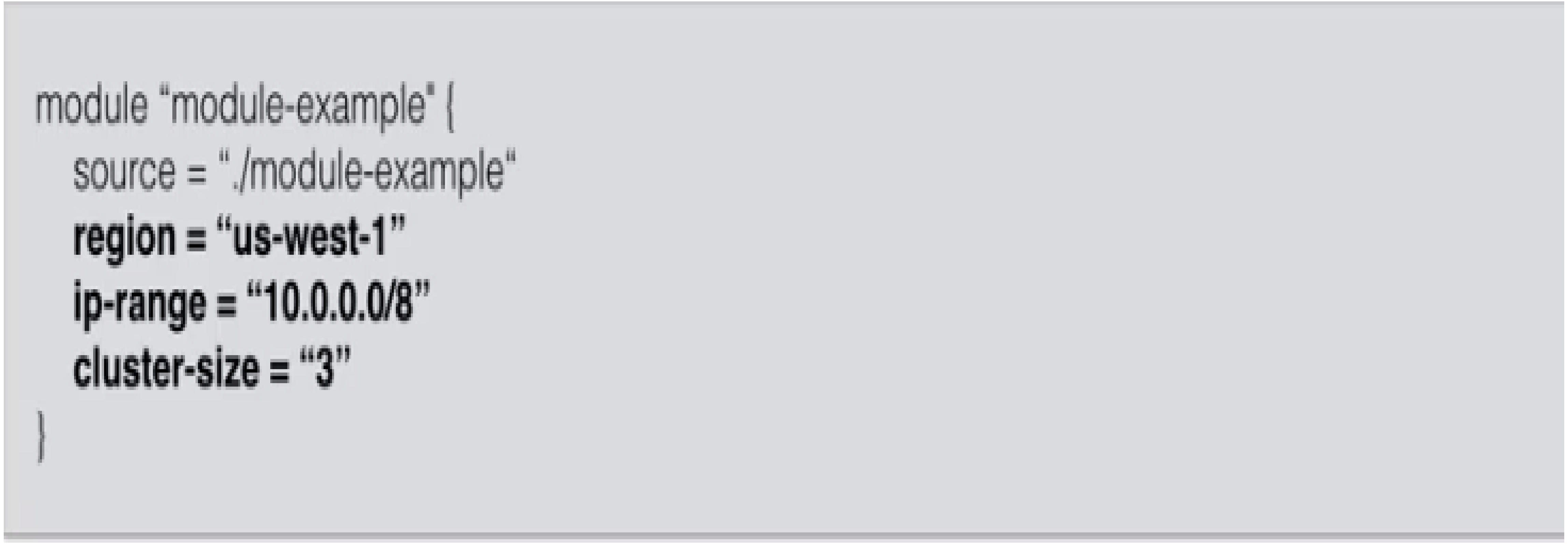
```
module “module-example" {
   source = “github.com/wardviaene/terraform-module-example“
}
```

Use a module from a local folder

```
module “module-example" {
   source = “./module-example"
}
```

Best Practice 2 (..Continued..)

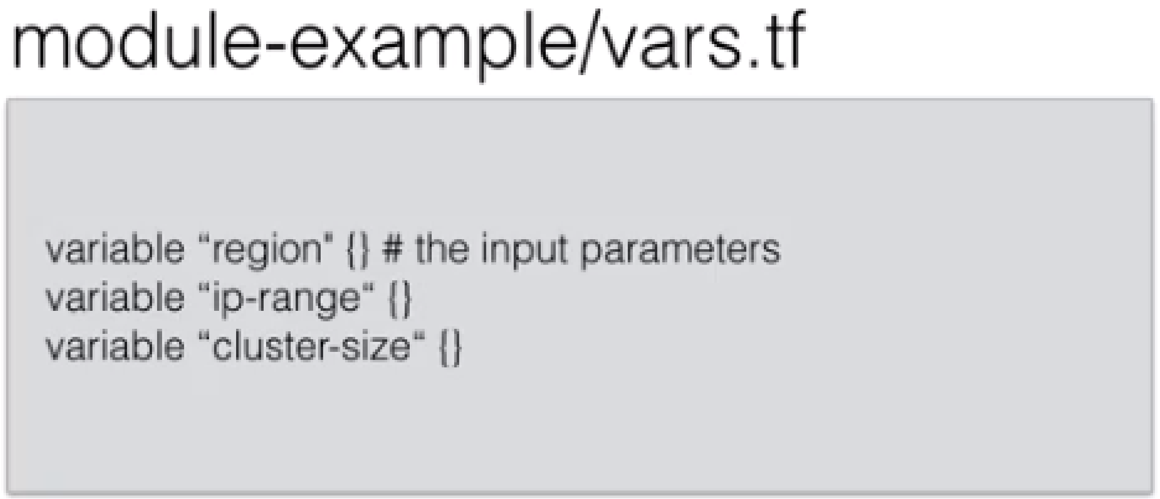
Pass arguments to the module



```
module “module-example" {
    source = “./module-example“
    region = “us-west-1”
    ip-range = “10.0.0.0/8”
    cluster-size = “3”
}
```

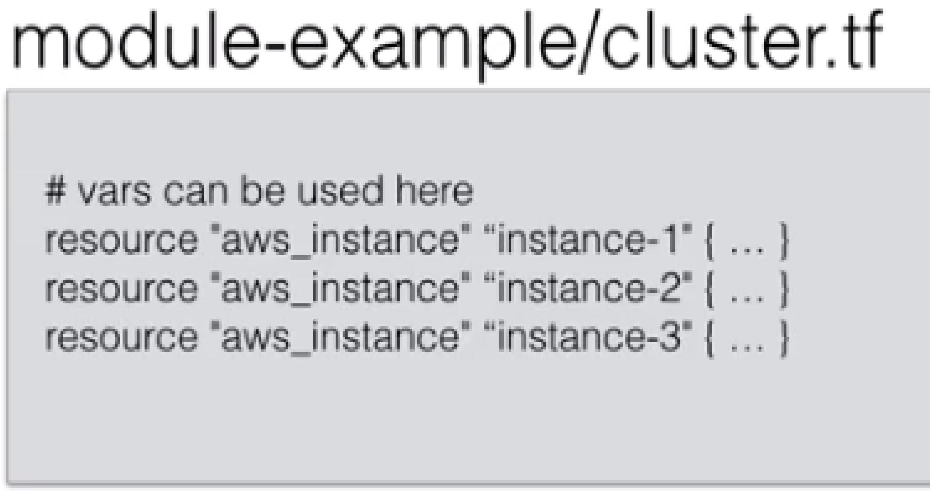
Best Practice 2 (..Continued..)

Inside the module folder, you just have again, terraform files:



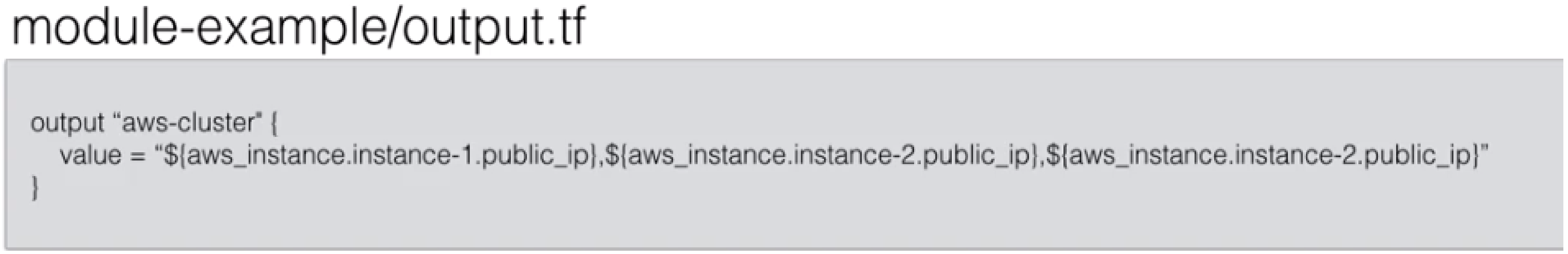
module-example/vars.tf

```
variable “region" {} # the input parameters
variable “ip-range“ {}
variable “cluster-size“ {}
```



module-example/cluster.tf

```
# vars can be used here
resource "aws_instance" “instance-1" { … }
resource "aws_instance" “instance-2" { … }
resource "aws_instance" “instance-3" { … }
```



module-example/output.tf

```
output “aws-cluster" {
   value = “${aws_instance.instance-1.public_ip},${aws_instance.instance-2.public_ip},${aws_instance.instance-2.public_ip}”
}
```

Best Practice 3

- Use implicit dependencies
- Implicit dependencies should be used whenever possible (see this article from terraform.io website for more information).
- With IaC the resources will be configured exactly as declared, and implicit dependencies can be used to ensure the creation order.

Best Practice 4

- Try to practice a consistent structure and **naming convention**
- Use _ (underscore) instead of - (dash) in all: resource names, data source names, variable names, outputs.
- Only use lowercase letters and numbers.

Generating Images

- The output of terraform graph is in the DOT format, which can easily be converted to an image by making use of dot provided by GraphViz:

```
$ terraform graph | dot -Tsvg > graph.svg
```

How should I structure my Terraform configurations?

- What is the complexity of your project?
- How often does your infrastructure change?
- How environments are grouped?
- Do not store all your code in a single file
- Provide a file structure that allows “separation of concerns”
- You can try more than one way to do it.

Terraform configurations (..Continued..)

provider.tf

```
provider "aws" {
   access_key = "${var.AWS_ACCESS_KEY}"
   secret_key = "${var.AWS_SECRET_KEY}"
   region = "${var.AWS_REGION}"
}
```

vars.tf

```
variable "AWS_ACCESS_KEY" {}
variable "AWS_SECRET_KEY" {}
variable "AWS_REGION" {
  default = "eu-west-1"
}
```

terraform.tfvars

```
AWS_ACCESS_KEY = “”
AWS_SECRET_KEY = “”
AWS_REGION = “”
```

instance.tf

```
resource "aws_instance" "example" {
  ami           = "ami-0d729a60"
  instance_type = "t2.micro"
}
```

Terraform configurations (..Continued..)

provider.tf

```
provider "aws" {
   access_key = "${var.AWS_ACCESS_KEY}"
   secret_key = "${var.AWS_SECRET_KEY}"
   region = "${var.AWS_REGION}"
}
```

instance.tf

```
resource "aws_instance" "example" {
  ami           = “${lookup(var.AMIS, var.AWS_REGION)}"
  instance_type = "t2.micro"
}
```

vars.tf

```
variable "AWS_ACCESS_KEY" {}
variable "AWS_SECRET_KEY" {}
variable "AWS_REGION" {
  default = "eu-west-1"
}
variable "AMIS" {
  type = "map"
  default = {
    us-east-1 = "ami-13be557e"
    us-west-2 = “ami-06b94666"
    eu-west-1 = "ami-0d729a60"
  }
}
```

https://cloud-images.ubuntu.com/locator/ec2/

Terraform configurations (..Continued..)

- Putting all code in main.tf is a good idea when you are getting started or writing an example code
- In all other cases you will be better having several files split logically like this:

  main.tf - call modules, locals and data-sources to create all resources

  variables.tf - contains declarations of variables used in main.tf

  outputs.tf - contains outputs from the resources created in main.tf
- terraform.tfvars should not be used anywhere except composition.

Hands On Lab 1

Cloning the Repository

https://github.com/AshWilliams/TerraformPoC.git

https://github.com/AshWilliams/terraform-best-practices-workshop

Q&A?

Microsoft