

Terraform CH1 Basics

Objectives

- AWS Infrastructure
- What Terraform is
- Terraform commands

Agenda

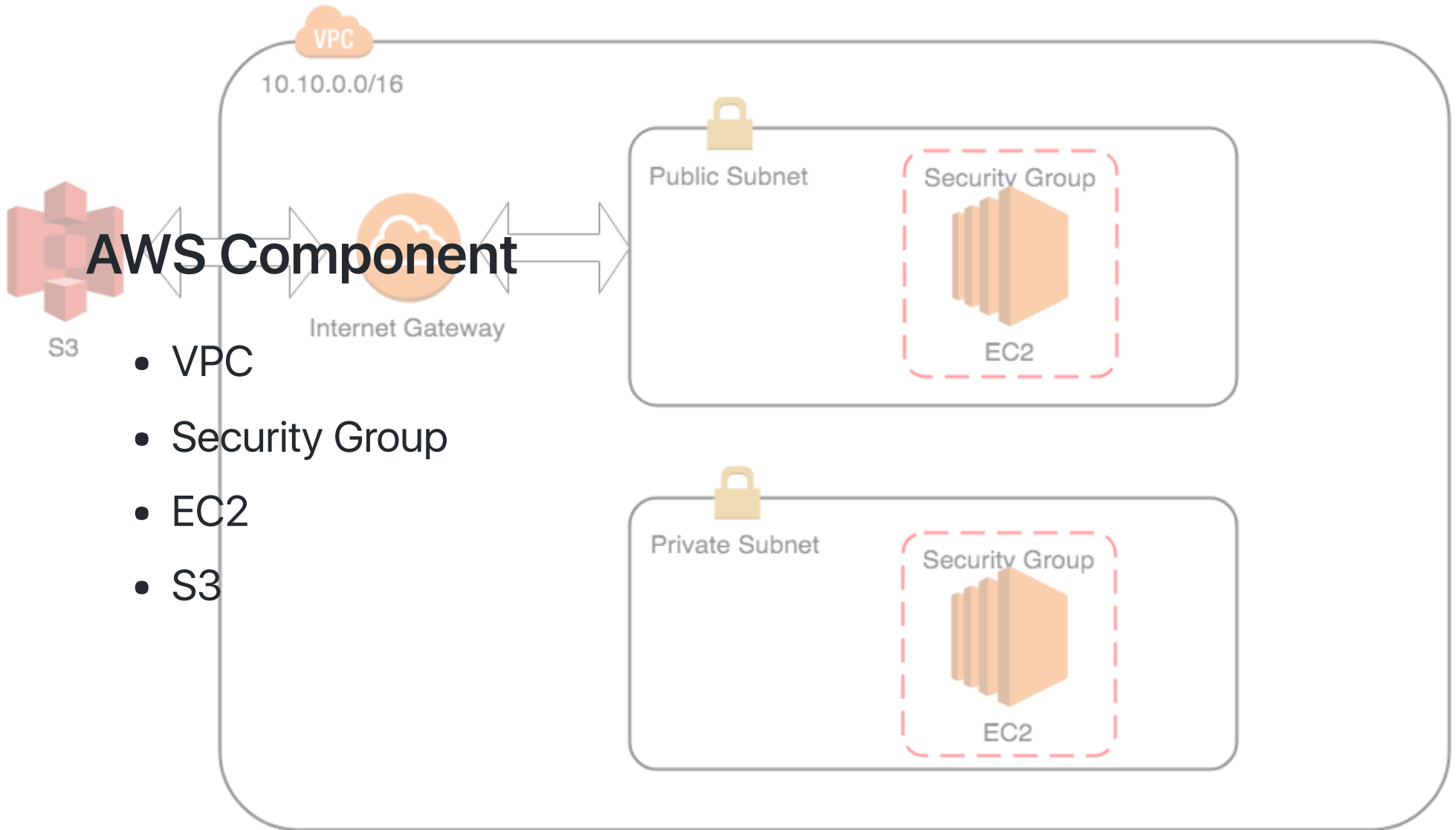
- Terraform Introduction
- AWS Introducea
 - VPC/EC2/Security Group/ELB/S3
- Setup Cloud9 Envieonment
- Spining up an instance with Terraform
 - Variables
 - Data
 - Output
 - Remote State (S3)
 - init/apply

Terraform Intro

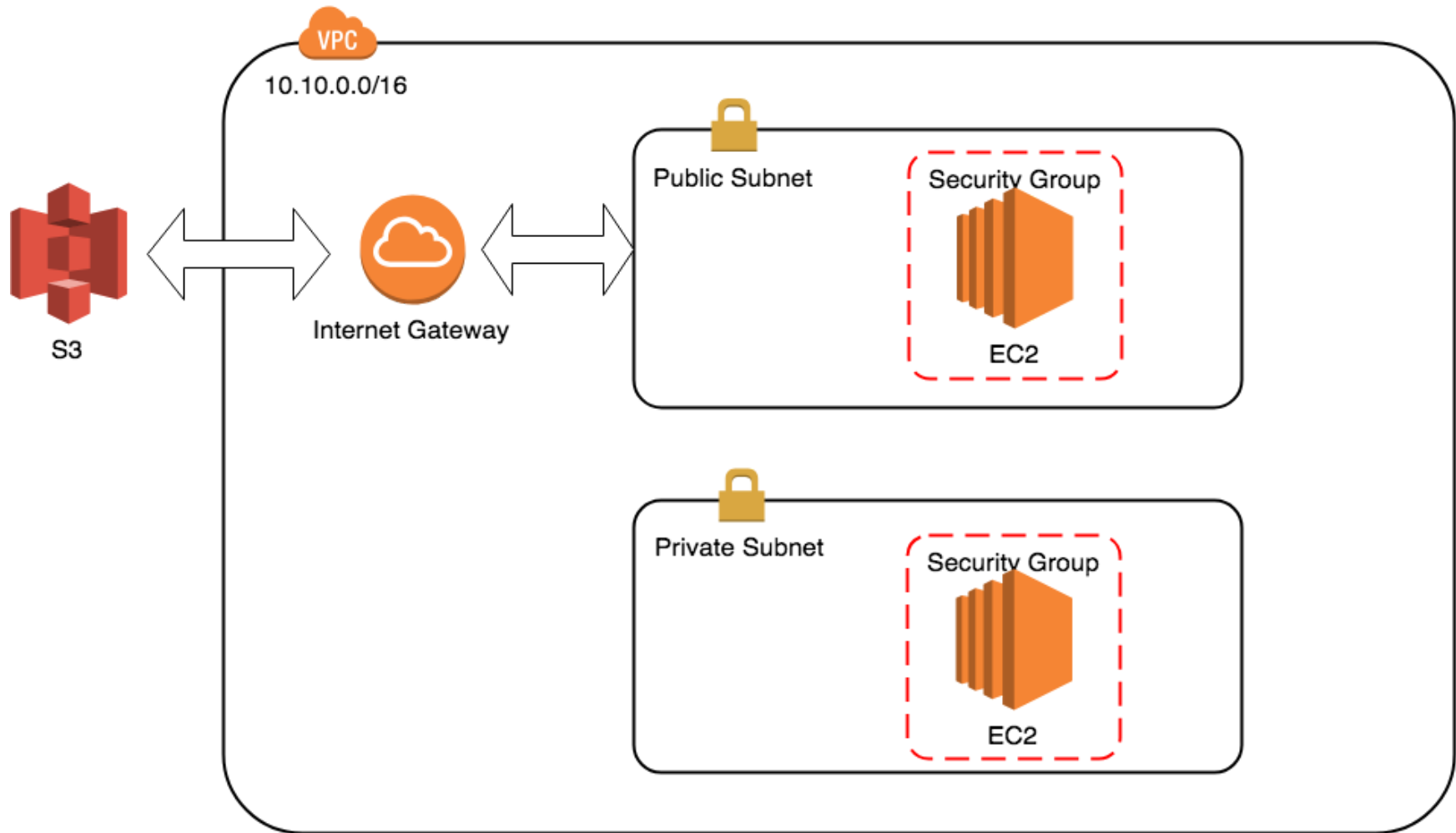
What is Terraform

- IaC (Infrastructure as Code) Tool
- Terraform is a tool for building, changing, and versioning infrastructure
- Support Major Cloud Provider (AWS, GCP, Azure ...etc)
- Bunch of Provider (DNS, Database, Monitor System ...etc)

Basic *AWS* Introduces



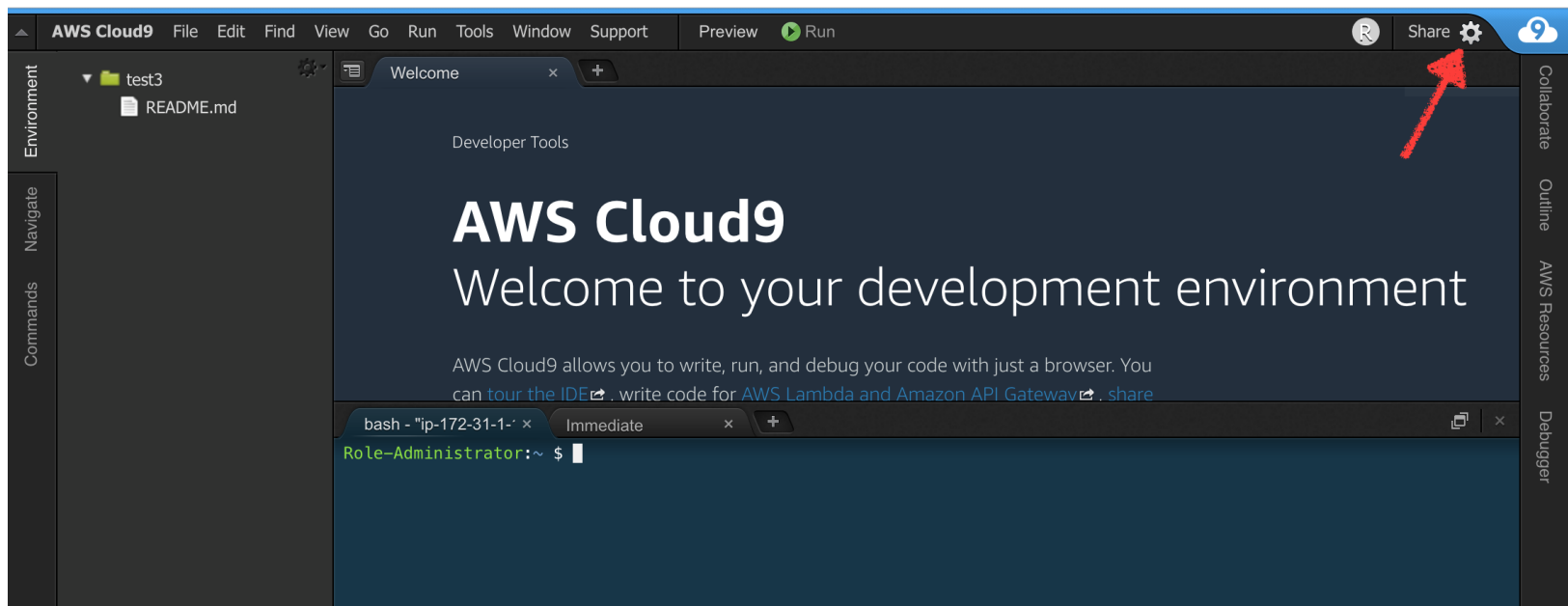
AWS Component



Setup Cloud9 Environment

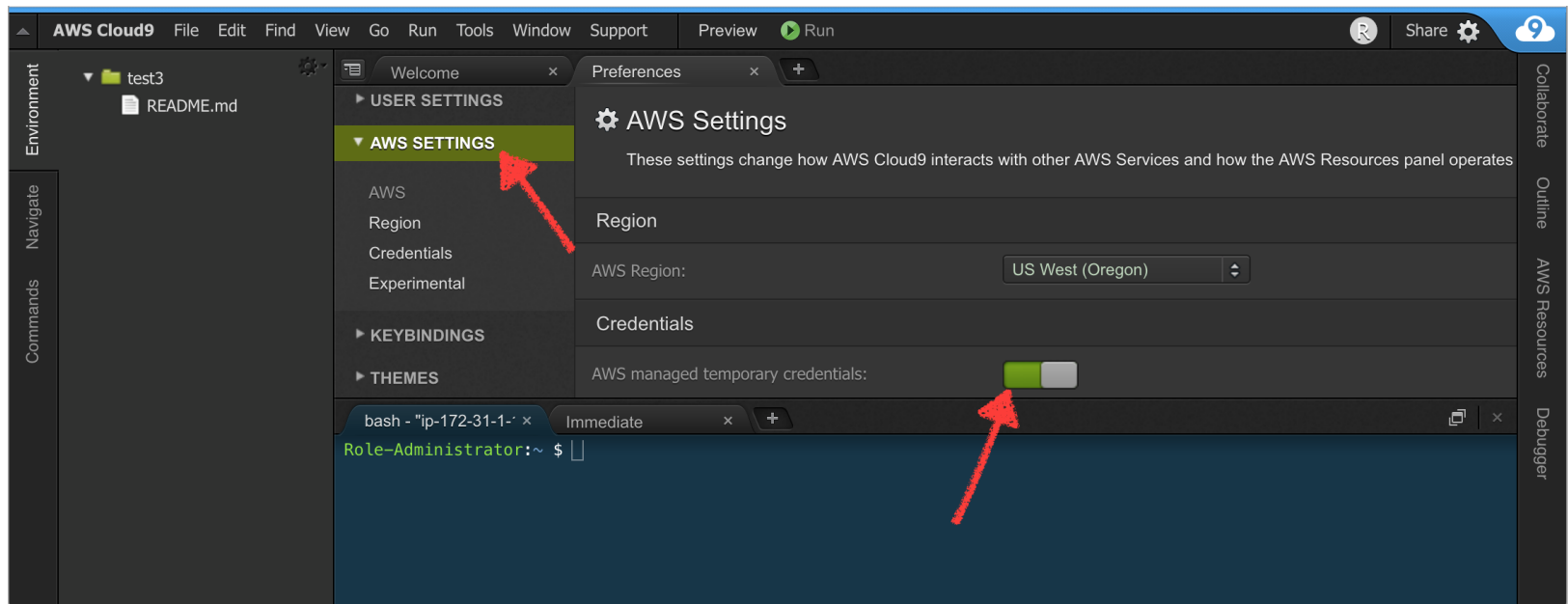
Cloud9 Environment Setting(1/3)

- Open Cloud9 (Service -> Cloud9 -> Your environment)
- Click Setting at top-right



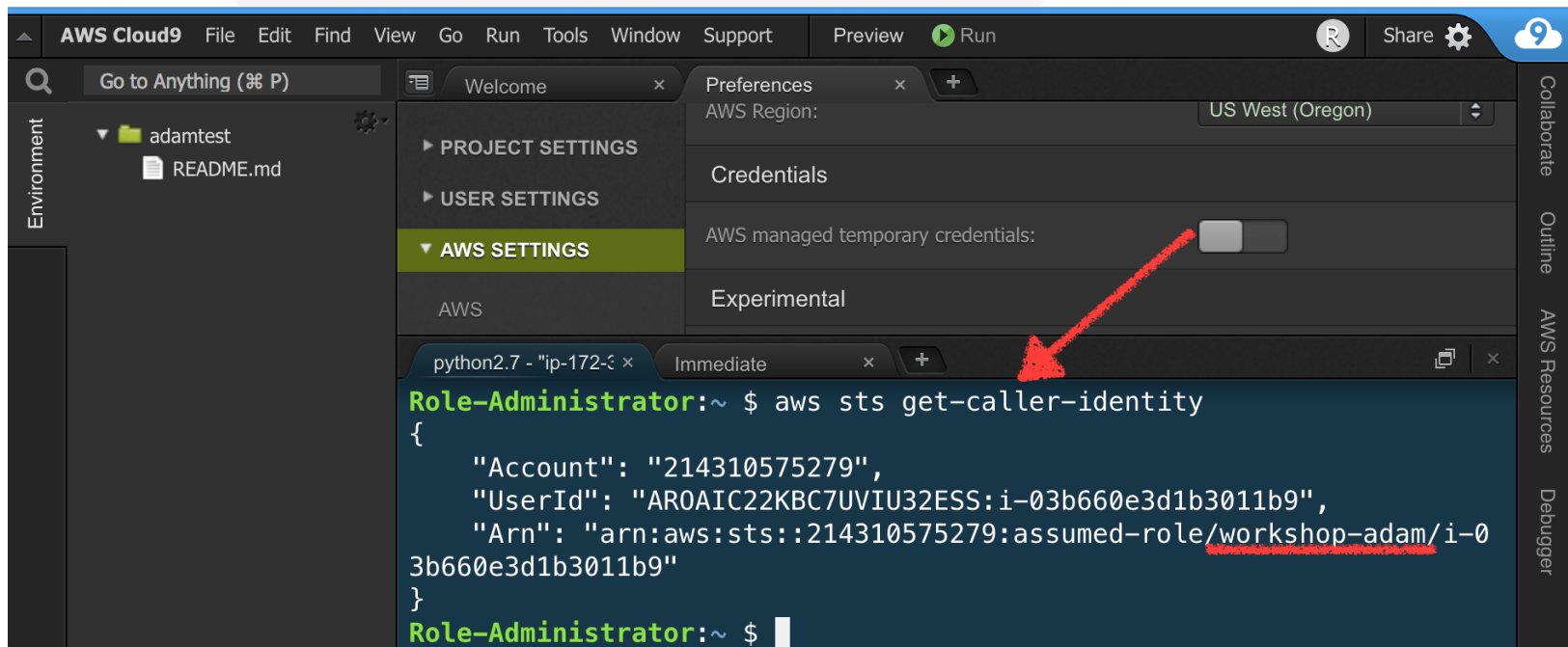
Cloud9 Environment Setting(2/3)

- Switch off "AWS Managed temporary credentials"



Cloud9 Environment Setting(3/3)

- Test with `aws sts get-caller-identity`



Terraform Commands

- init (初始化)
- plan (查看計畫)
- apply (執行計畫)
- destroy (移除資源)
- get (取得相關模組)
- graph (繪製元件關係圖)

First EC2 Instance

```
$ cd workshop/aws/ch01/100-create-instance  
main.tf
```

```
provider "aws" {  
    region = "us-east-1"  
}  
  
resource "aws_instance" "example" {  
    ami           = "ami-2757f631"  
    instance_type = "t2.micro"  
}
```

Terraform init

```
$ terraform init
```

Initializing provider plugins...

- Checking for available provider plugins on <https://releases.hashicorp.com>...
- Downloading plugin for provider "aws" (1.34.0)...

The following providers do not have any version constraints in configuration, so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking changes, it is recommended to add version = "..." constraints to the corresponding provider blocks in configuration, with the constraint strings suggested below.

```
* provider.aws: version = "~> 1.34"
```

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

Terraform plan / Terraform apply

```
$ terraform apply
```

Terraform will perform the following actions:

```
+ aws_instance.example
  id:                <computed>
  ami:               "ami-2757f631"
  arn:               <computed>
  associate_public_ip_address: <computed>
  availability_zone: <computed>
  cpu_core_count:    <computed>
  cpu_threads_per_core: <computed>
  ebs_block_device.#: <computed>
```

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: █

Terraform state file

```
$ cat terraform.tfstate
```

It's a JSON file, Terraform use it to map from real world resource to Terraform structures.

Terraform destroy

```
$ terraform destroy
```

Terraform will perform the following actions:

- `aws_instance.example`

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy?

Terraform will destroy all your managed infrastructure, as shown above.

There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

`aws_instance.example`: Destroying... (ID: `i-0c79ca926ea2094a5`)

`aws_instance.example`: Still destroying... (ID: `i-0c79ca926ea2094a5`, 10s elapsed)

`aws_instance.example`: Still destroying... (ID: `i-0c79ca926ea2094a5`, 20s elapsed)

`aws_instance.example`: Still destroying... (ID: `i-0c79ca926ea2094a5`, 30s elapsed)

`aws_instance.example`: Still destroying... (ID: `i-0c79ca926ea2094a5`, 40s elapsed)

`aws_instance.example`: Still destroying... (ID: `i-0c79ca926ea2094a5`, 50s elapsed)

`aws_instance.example`: Destruction complete after 58s

Destroy complete! Resources: 1 destroyed.

Terraform Remote State

```
$ cd workshop/aws/ch01/110-remote-state-variables  
backend.tf
```

```
terraform {  
  backend "s3" {  
    bucket = "a-long-name-to-s3-bucket-include-date"  
    key    = "prod/terraform.tfstate"  
    region = "us-east-1"  
  }  
}
```

Terraform Remote State

main.tf

```
provider "aws" {  
  region = "us-west-2"  
}  
  
resource "aws_instance" "example" {  
  ami           = "ami-0bbe6b35405ecebdb"  
  instance_type = "t2.micro"  
  tags {  
    Name = "HelloTerraform"  
  }  
}
```

Terraform Input Variables

variables.tf

```
variable "region" {  
    default = "us-west-1"  
}  
variable "instance_type" {}  
variable "ami" {}
```

main.tf

```
provider "aws" {  
    region = "${var.region}"  
}  
  
resource "aws_instance" "example" {  
    ami           = "${var.ami}"  
    instance_type = "${var.instance_type}"  
    tags {  
        Name = "HelloTerraform"  
    }  
}
```

Terraform Input Variables

prod.tfvar

```
region="us-west-2"  
ami="ami-0bbe6b35405ecebdb"  
vm_size="t2.micro"
```

Execute command:

```
$ terraform init  
$ terraform apply -var-file=./prod.tfvar
```

Terraform Output

Output public IP for user can be connect.
Important concept when write module.

Terraform Output

output.tf

```
output "public_ip" {  
  value = "${aws_instance.example.public_ip}"  
}
```

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

public_ip = 54.90.97.210

-

Genarate ssh key

Genarate ssh key

```
$ ssh-keygen -f /home/cloud9/.ssh/id_rsa -P ''
```

Create AWS Keypair

Create AWS Keypair

Execute

```
$ cd workshop/aws/ch01/110-remote-state-variables
```

```
$ ./genkey.sh
```

main.tf

```
resource "aws_key_pair" "devopsdays-workshop" {  
  key_name      = "devopsdays-workshop"  
  public_key    = "${file(pathexpand("/home/cloud9/.ssh/id_rsa.pub"))}"  
}
```

```
$ terraform apply
```

Key Takeaways

- Known AWS infrastructure
- Terraform commands
- Terraform Variable/State File/Output