Part 2

AWS Workshop Series

Day 8: Terraform in AWS

Taking Enterprise Beyond the Cloud by TrueIDC

Mr. Athiwat Itthiwatana

Cloud & Solution Consultant

Presented by

- Athiwat Itthiwatana (HAM)
- Cloud & Solution Consultant, TrueIDC
- AWS Specialist
- SAP Basis Specialist
- athiwat.itt@ascendcorp.com

aws certified
Solutions Architect
PROFESSIONAL

aws certified
SAP on AWS
SPECIALTY

aws certified
Security
SPECIALTY

aws partner network
ambassador

trueIDC

Agenda

- Infrastructure as Code
- What is Terraform ?
- How to setup on AWS ?

Why Infrastructure as Code ?

- More automation involves fewer human errors
- Support collaboration in DevOPS
- Traceability
- Integrity
- Repeatability
- Agility
- Code is the best document

What is Terraform ?

Terraform is an infrastructure as code tool that lets you build, change, and version infrastructure safely and efficiently. This includes low-level components like compute instances, storage, and networking; and high-level components like DNS entries and SaaS features.

Benefit of using Terraform

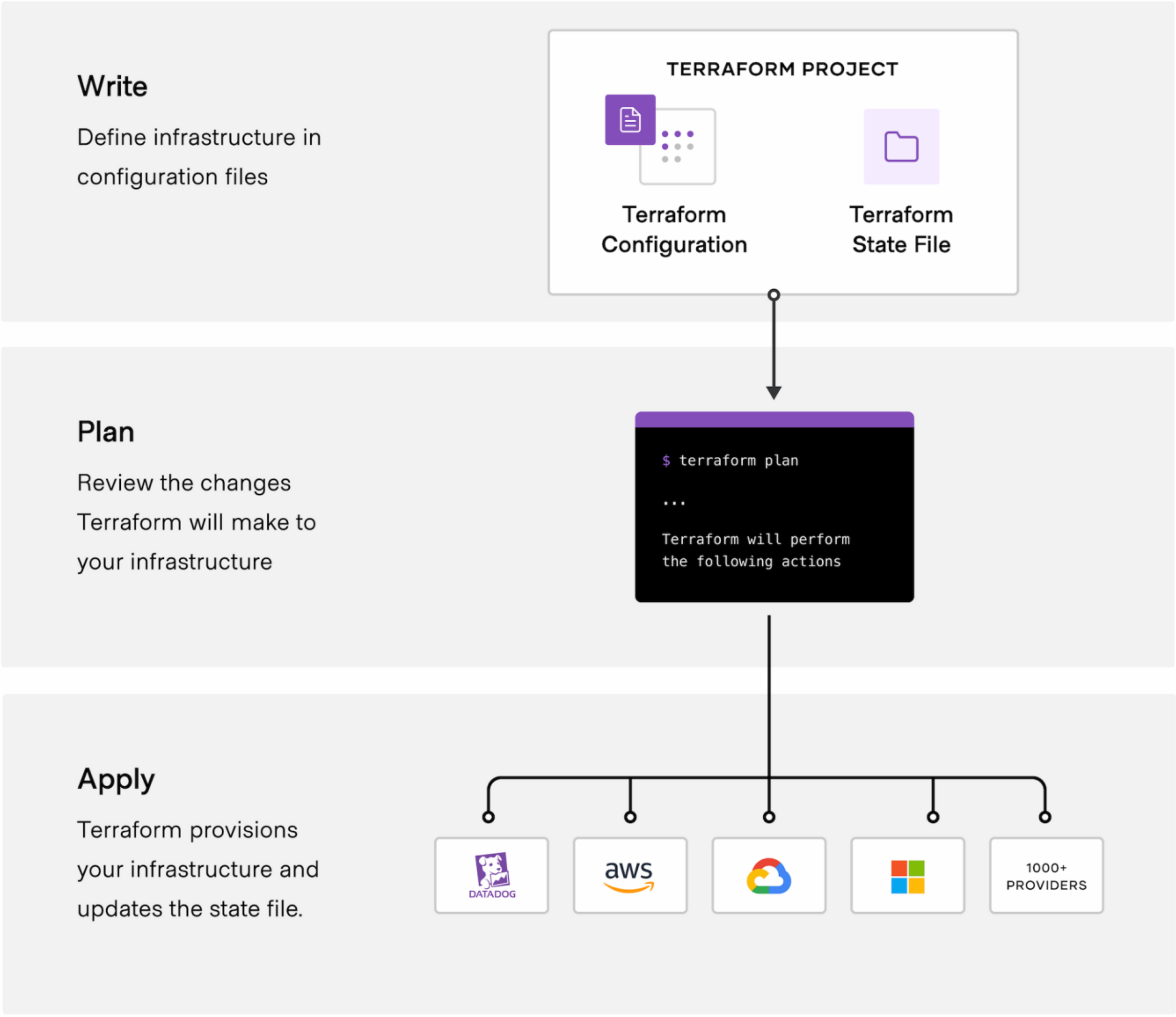
Multiple Cloud Platforms

Configuration for Humans

Tracking Resources with State

The core Terraform workflow

- **Scope**: Identify the infrastructure for your project
- **Author**: Write configuration to define your infrastructure
- **Initialize**: Install the required Terraform providers
- **Plan**: Preview the changes Terraform will make
- **Apply**: Make the changes to your infrastructure



Write

Define infrastructure in configuration files

Plan

Review the changes Terraform will make to your infrastructure

Apply

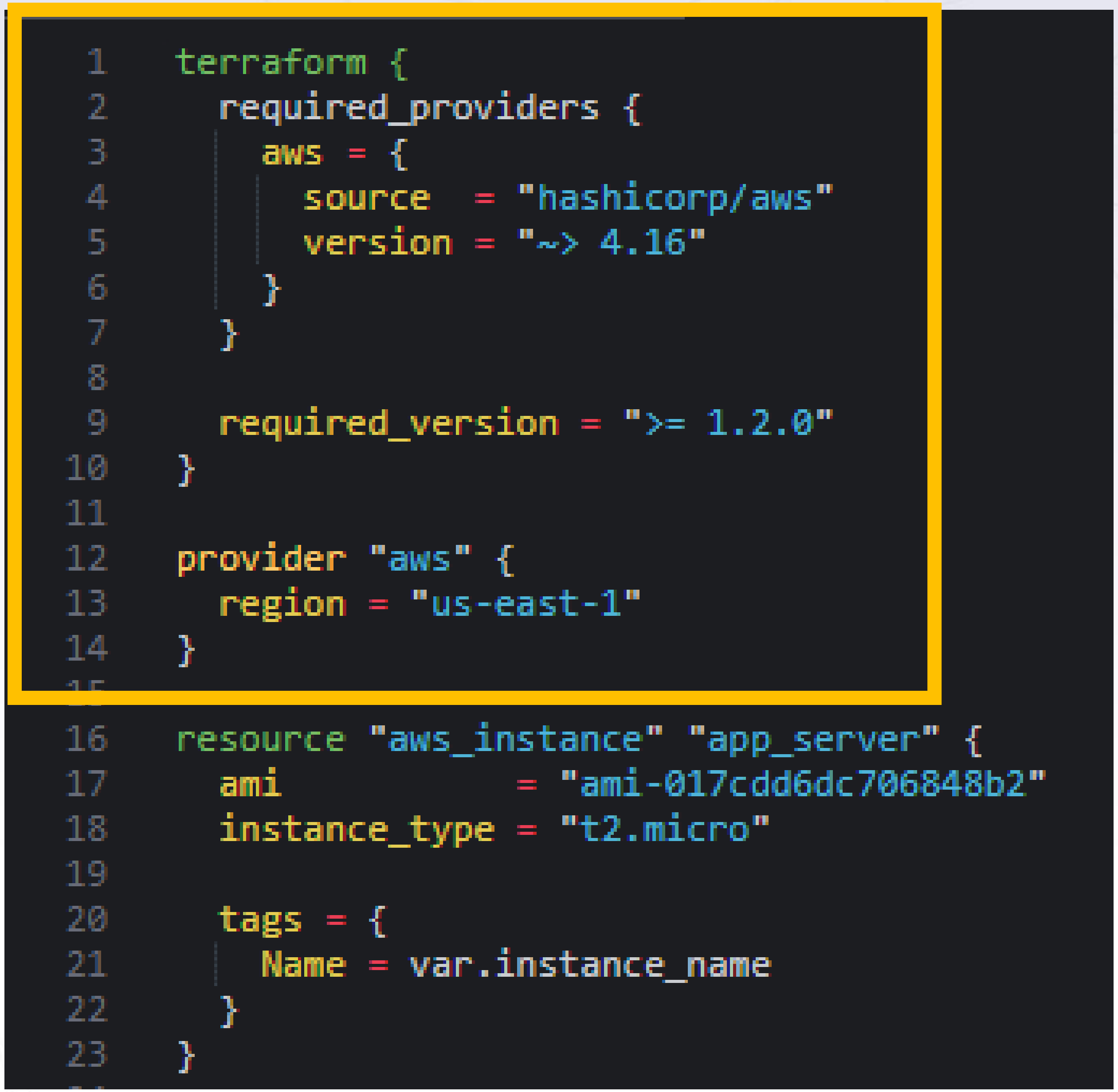
Terraform provisions your infrastructure and updates the state file.

```
$ terraform plan

...

Terraform will perform
the following actions
```

Terraform Syntax
Providers

Responsible for understanding API interactions and exposing resources.

Example: AWS, GCP, Azure

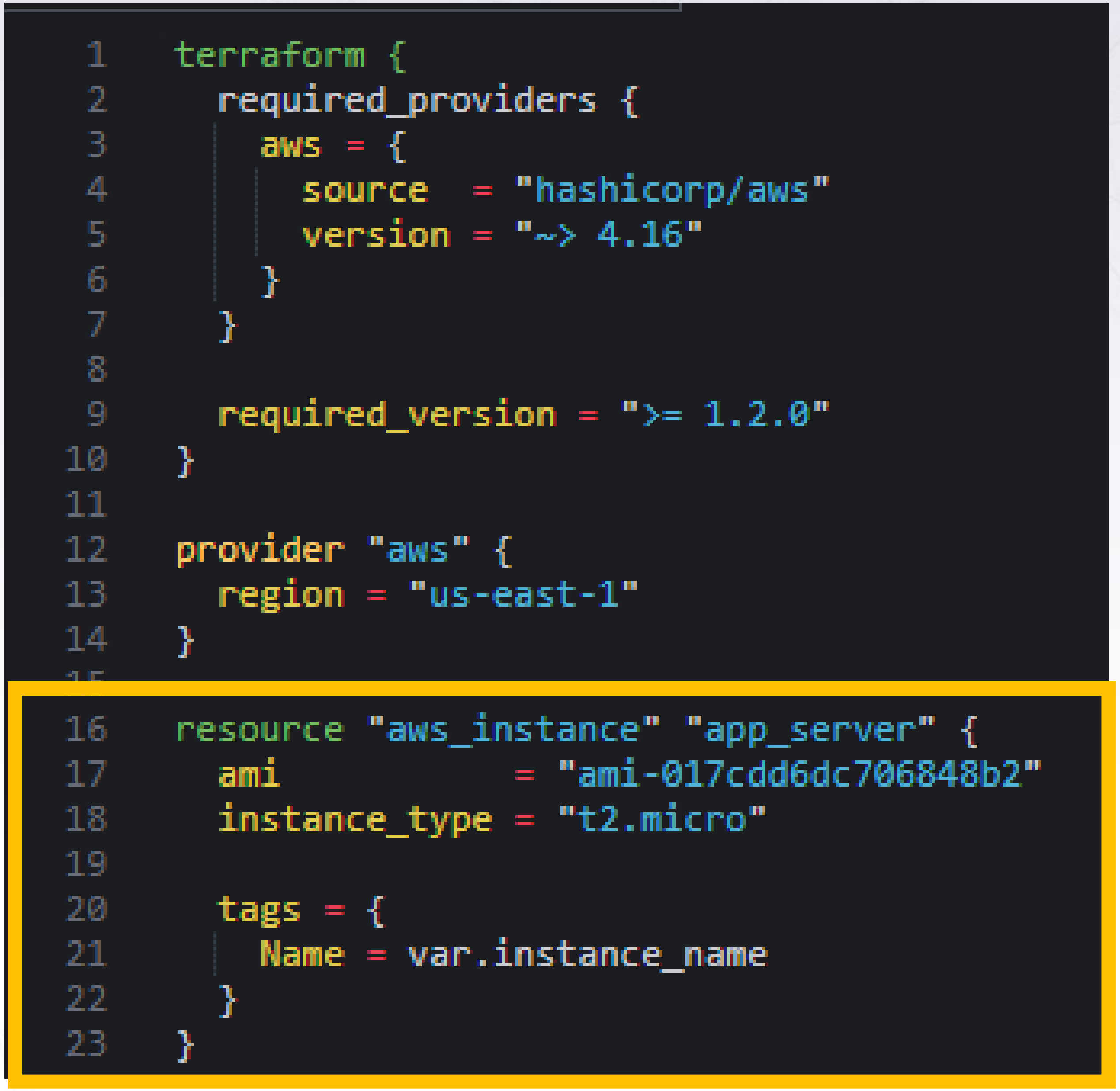


```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }

  required_version = ">= 1.2.0"
}

provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "app_server" {
  ami           = "ami-017cdd6dc706848b2"
  instance_type = "t2.micro"

  tags = {
    Name = var.instance_name
  }
}
```

Terraform Syntax
Resources

What you want to build

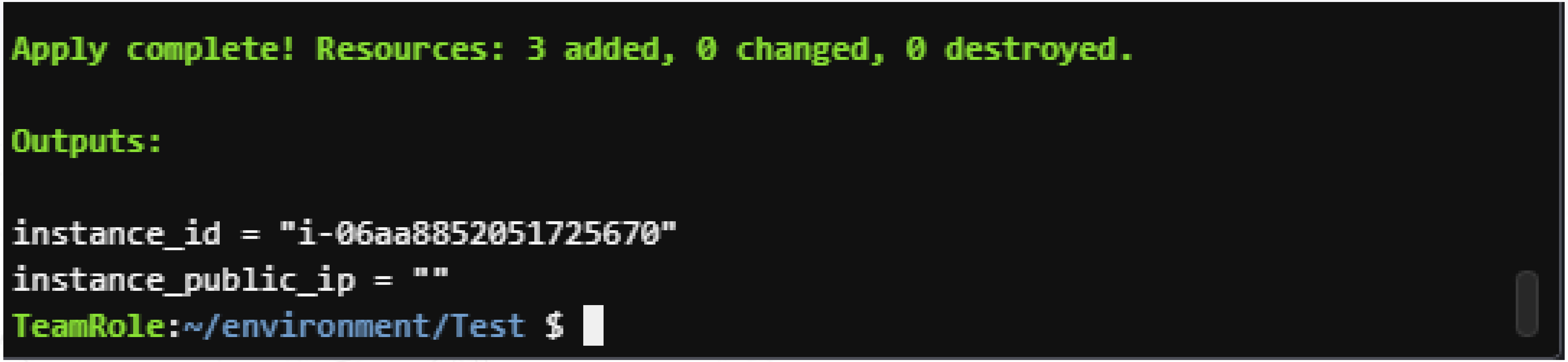
Example: EC2, VPC or any other services.



```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }

  required_version = ">= 1.2.0"
}

provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "app_server" {
  ami           = "ami-017cdd6dc706848b2"
  instance_type = "t2.micro"

  tags = {
    Name = var.instance_name
  }
}
```

Terraform Syntax
Output

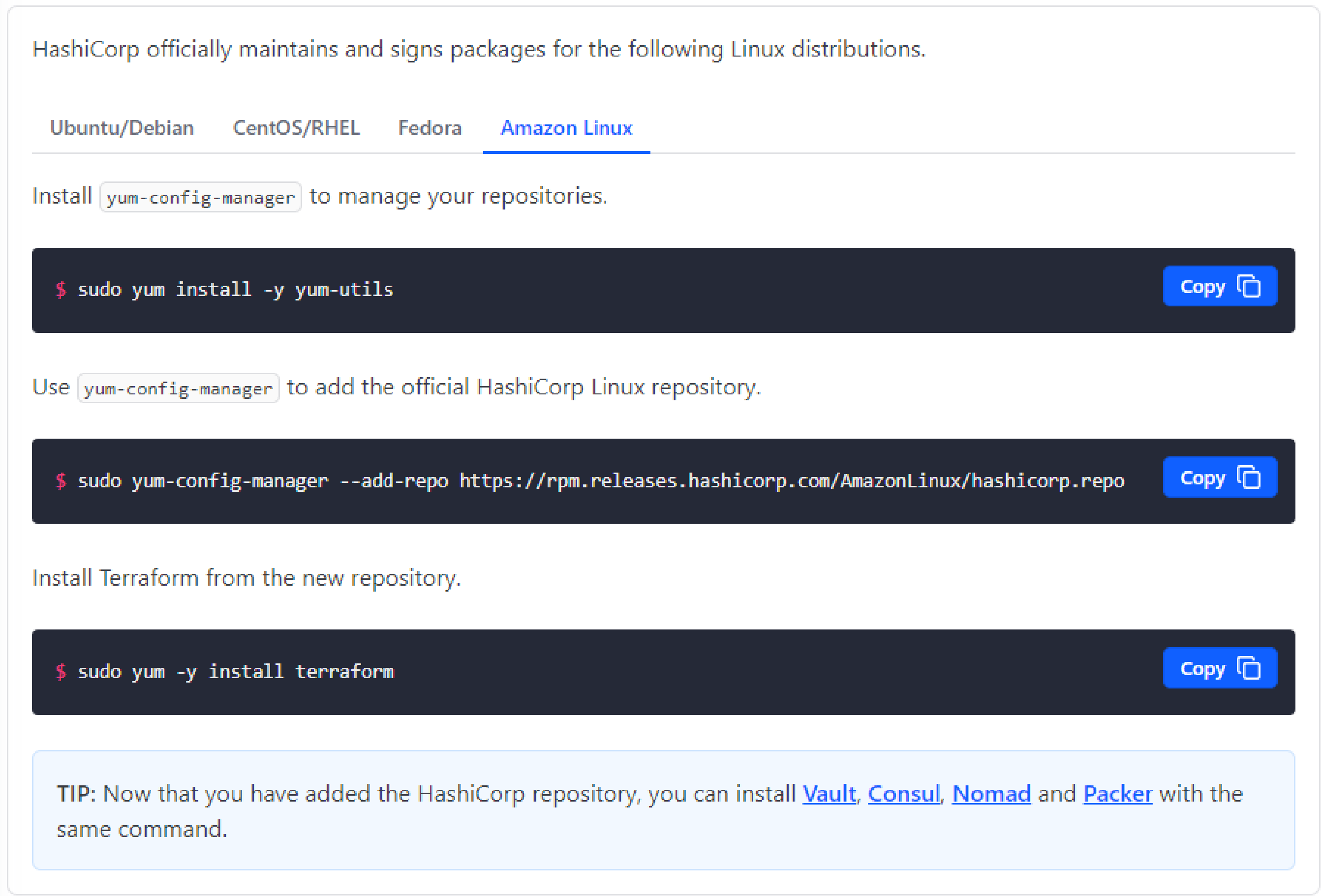
Output queried and shown back on screen after apply complete



```
Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Outputs:

instance_id = "i-06aa8852051725670"
instance_public_ip = ""
TeamRole:~/environment/Test $ 
```

How to setup Terraform on AWS

- Create Cloud9 environment
- Install Terraform with Amazon Linux instruction



HashiCorp officially maintains and signs packages for the following Linux distributions.

Install `yum-config-manager` to manage your repositories.

```
$ sudo yum install -y yum-utils
```

Use `yum-config-manager` to add the official HashiCorp Linux repository.

```
$ sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
```

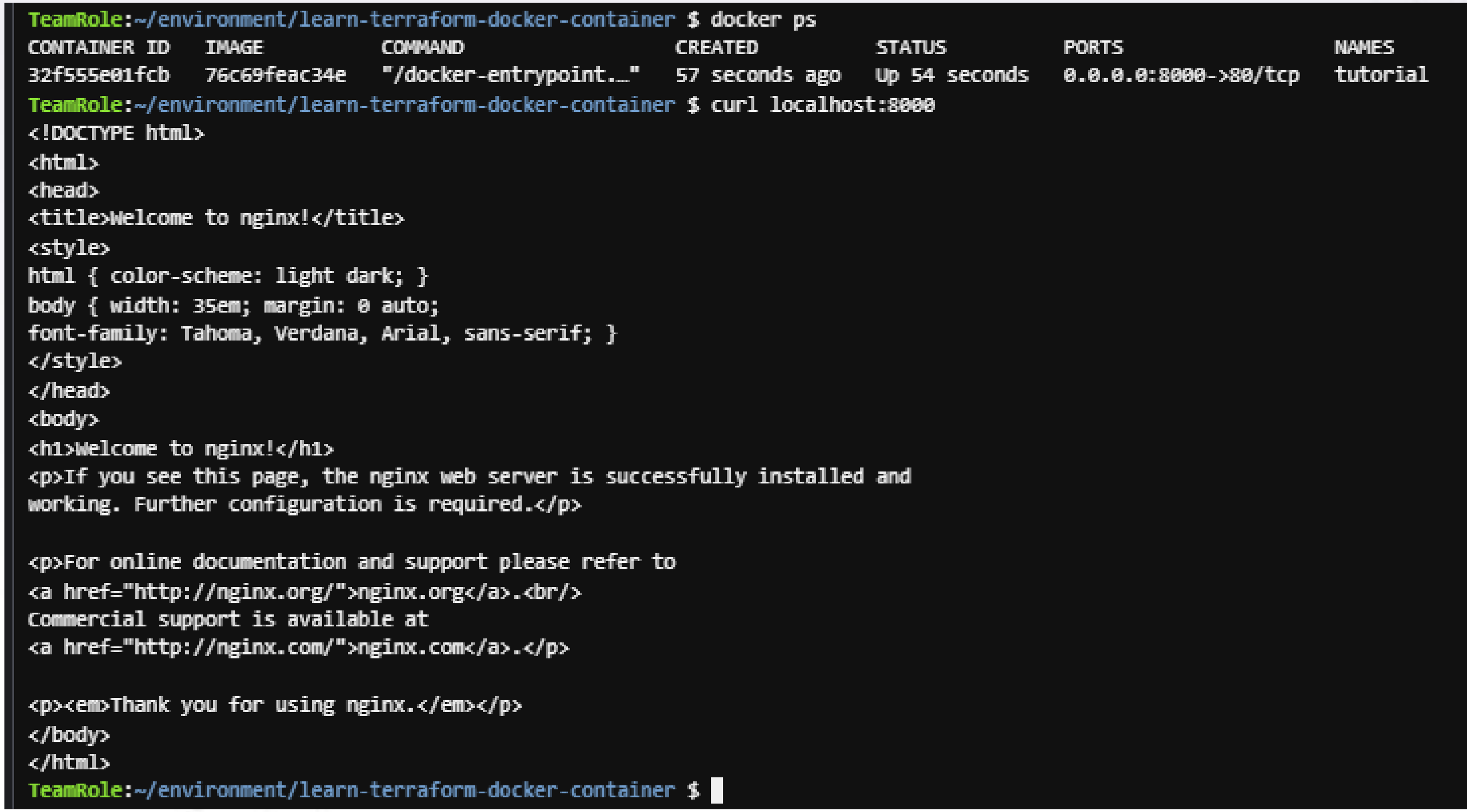
Install Terraform from the new repository.

```
$ sudo yum -y install terraform
```

TIP: Now that you have added the HashiCorp repository, you can install Vault, Consul, Nomad and Packer with the same command.

Quick Start Tutorial

- Skip install Docker Engine
- Create Folder and Terraform file
- Test run with curl localhost:8000



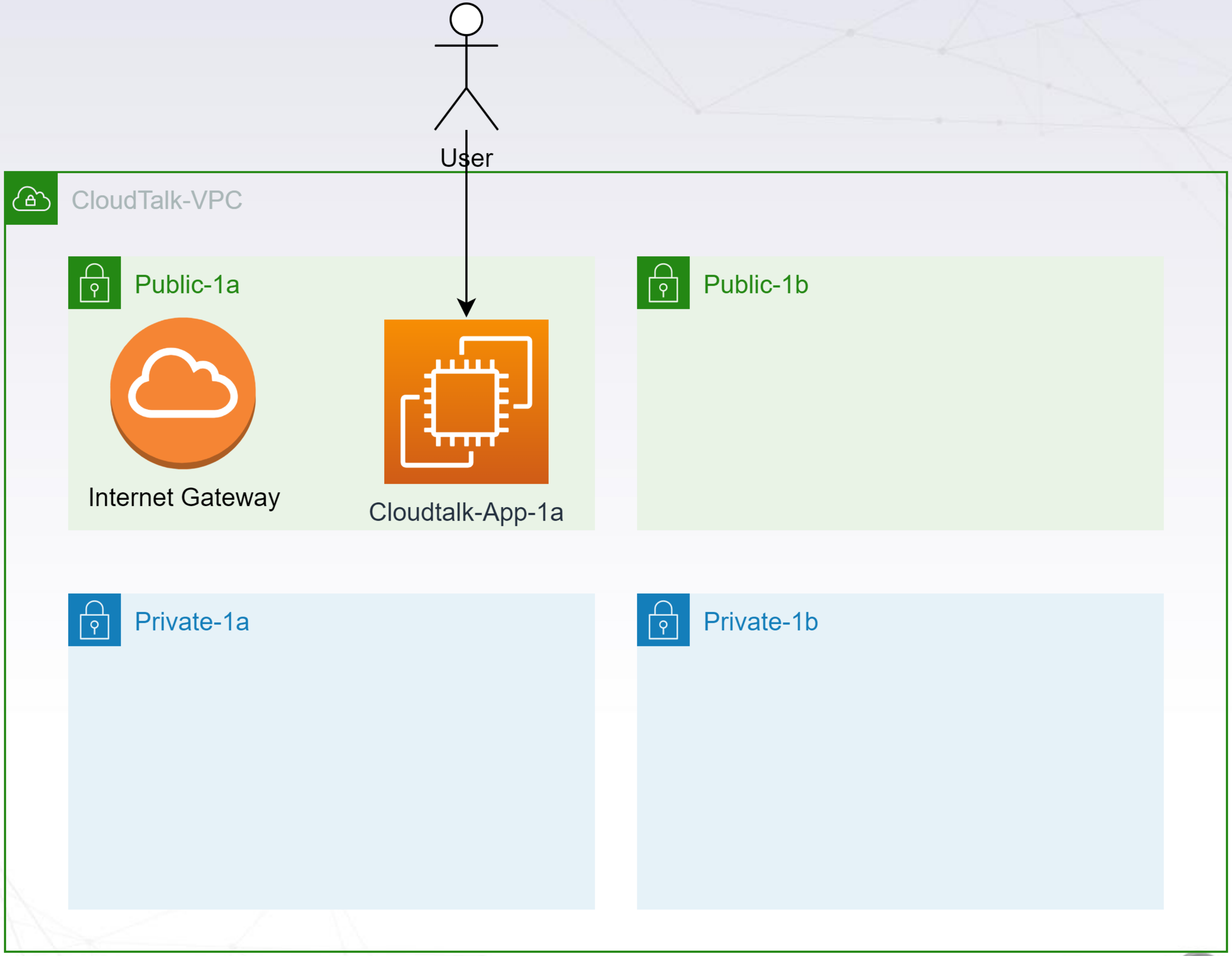
```
TeamRole:~/environment/learn-terraform-docker-container $ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                  NAMES
32f555e01fcb   76c69feac34e   "/docker-entrypoint.…"   57 seconds ago   Up 54 seconds   0.0.0.0:8000->80/tcp   tutorial
TeamRole:~/environment/learn-terraform-docker-container $ curl localhost:8000
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
TeamRole:~/environment/learn-terraform-docker-container $ 
```

Lab: Terraform on AWS

Source: AWS Immersion day

Challenge

- VPC 192.168.0.0/16
- Subnet /24
- No key allow in EC2
- Start website using user data



Source: AWS Immersion day

THANK YOU FOR JOINING US!

trueIDC

REGIONAL
DATA CENTER &
CLOUD SERVICE
PROVIDER