

IaC – DevOps with Terraform

Objective: Get practical experience in DevOps (Development and Operation) and Infrastructure-as-Code through the automated cloud infrastructure creation and deployment on AWS using Terraform framework.

Tasks:

1. Describing the virtual infrastructure (VM, network adapter, security group, firewall rules)
2. Deployment of the virtual infrastructure on AWS EC2

Lab environment:

- Visual Studio Code with the following plugins:
 - Hashicorp Terraform
- Terraform
- AWS CLI

Contents

Step 1: Opening sample Terraform project	3
Step 2: Understanding Terraform Code	5
Step 3: Updating key VM settings	8
Step 4: Configuring AWS credentials for AWS CLI	11
Step 5: Deploying AWS Virtual Infrastructure	13
Step 6: Checking Deployed Resources	14
Step 7: Removing AWS EC2 Resources	17
Step 8: Advanced Self Task	17
References	17

Terraform by HashiCorp is an open-source **infrastructure-as-code (IaC)** software framework that allows DevOps engineers to programmatically provision and manage the virtual computing resources and cloud infrastructure.

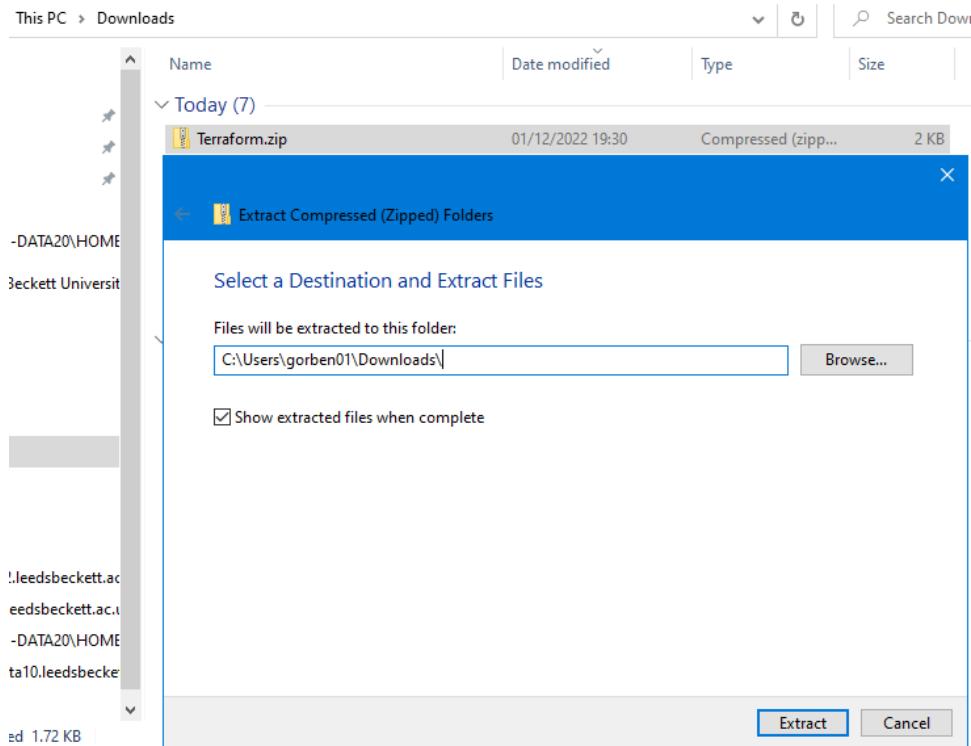
Infrastructure-as-code is an IT practice to manage computing infrastructure through programming. This approach to resource allocation allows developers to logically manage, monitor and provision virtual computing resources – as opposed to configuring each required resource manually.

Terraform allows to describe computing infrastructure by using a JSON-like configuration language called HCL (HashiCorp Configuration Language) and automatically deploy it in the private or public virtualised environment, e.g. AWS, Azure, etc.

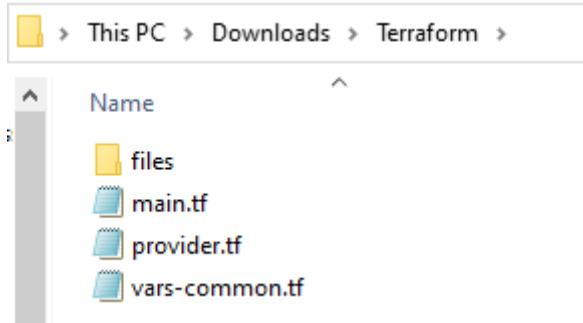
Step 1: Opening sample Terraform project

In this lab we are using Visual Studio Code IDE with the Terraform plugin installed. Do not forget to delete AWS resources after use.

1. Download and unzip a sample Terraform project (Terraform.zip) from the VLE to the Download folder (e.g. C:\Users\c1234567\Downloads\, where c1234567 is your student ID).

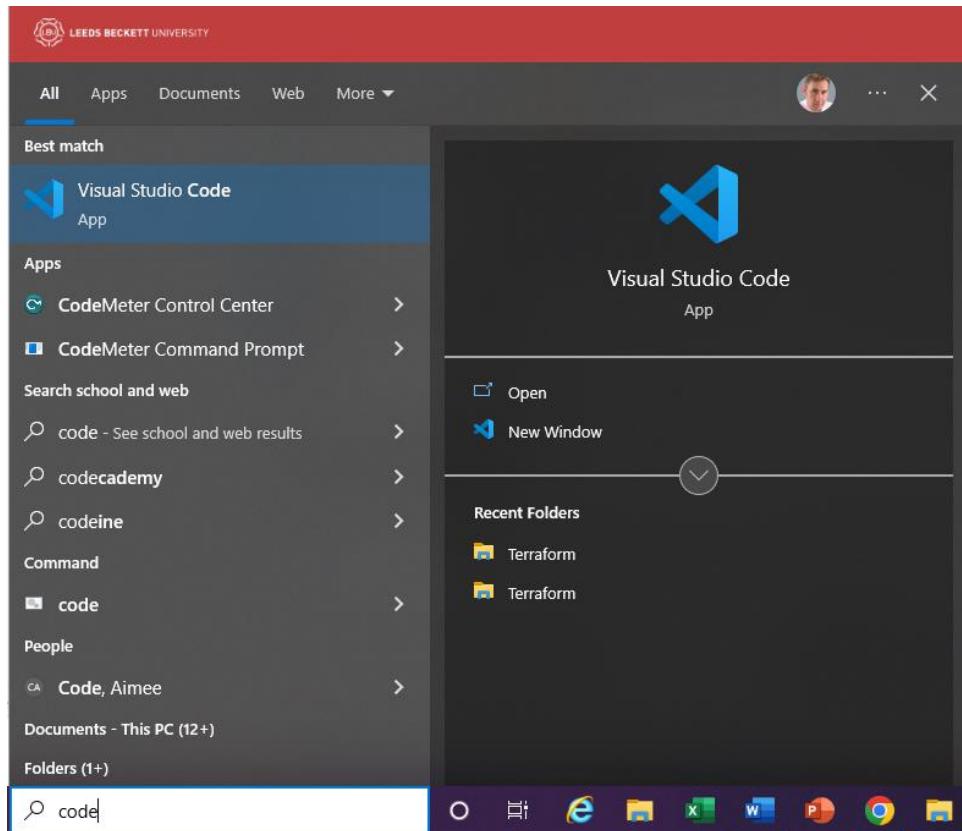


Make sure you have the following file structure after you unzip the project (avoid having 'Terraform' folder inside of another 'Terraform' folder):

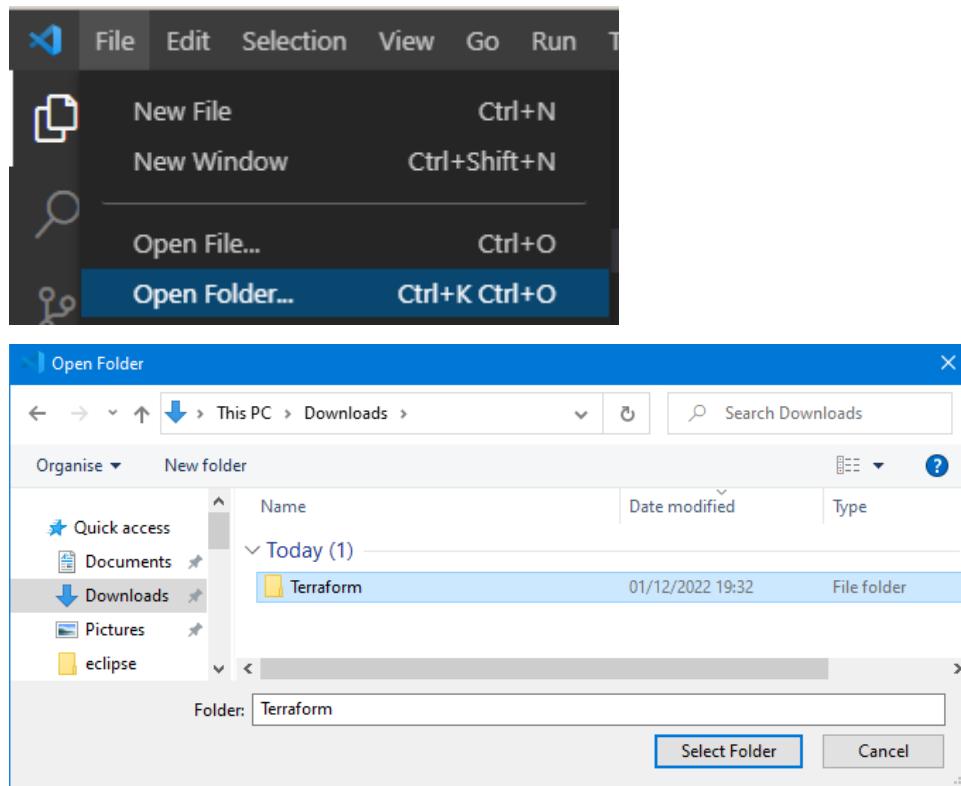


Note: Check, that your Download folder is on C: drive (e.g. C:\Users\c1234567\Downloads\), not on P: drive (!!!the project will not work from P:!!). If you do not have 'Downloads' on C:, then unzip the file to C:\Users\c1234567\Documents\.

2. Run Visual Studio Code IDE (not Visual Studio Community/Enterprise):



3. In the Visual Studio Code select File->Open Folder and browse to the unzipped 'Terraform' folder in Downloads:



Step 2: Understanding Terraform Code

1. The sample project includes three Terraform configuration files, describing the virtual instance we are going to deploy (`main.tf`, `provider.tf` and `vars-common.tf`) and the startup shell script (`files/template.tpl`) installing and running Apache web server and creating a simple web page. Explore these files.

```

main.tf
1 #Create Security Group and rules
2 resource "aws_security_group" "vm" {
3   name      = "${var.project_name}-vm-sg"
4   description = "Controls in/out traffic for vm networking"
5   vpc_id    = "vpc-075b0a07621fed9d6" #default VPC for us-east-1
6 }
7
8 resource "aws_security_group_rule" "inbound_http_to_vm" {
9   security_group_id = aws_security_group.vm.id
10  description      = "Allow http protocol for any"
11  type            = "ingress"
12  from_port       = "80"
13  to_port         = "80"
14  protocol        = "tcp"
15  cidr_blocks     = ["0.0.0.0/0"]
16 }
17
18 resource "aws_security_group_rule" "inbound_ssh_to_vm" {
19   security_group_id = aws_security_group.vm.id
20   description      = "Allow ssh protocol for any"
21   type            = "ingress"

```

2. File 'provider.tf'

This file simply defines a cloud provider we are going to deploy our VM with (AWS in our example):

```
provider "aws" {
  profile = "default"
  region = var.aws_region
}
```

The AWS region is described in `aws_region` variable which is defined in `vars-common.tf` file.

3. File 'vars-common.tf'

This file defines common variables used in the project, e.g. name of the project and AWS region we would like to deploy our VM in:

```
variable "project_name" {
  default = "CCD_DevOps_Example_c1234567"
}

#Set your aws_region
variable "aws_region" {
  default = "us-east-1"
}
```

Replace `c1234567` with your student id.

4. File 'main.tf'

This is the main Terraform config file describing the computing infrastructure we are going to create and deploy.

```
#Create Security Group and rules for inbound and outbound network traffic
resource "aws_security_group" "vm" {
  name          = "${var.project_name}-vm-sg"
  description   = "Controls in/out traffic for vm networking"
  vpc_id        = "vpc-075b0a07621fed9d6" #id of the Virtual Private Network
- you will need to update it
}

resource "aws_security_group_rule" "inbound_http_to_vm" {
  security_group_id = aws_security_group.vm.id
  description       = "Allow http protocol for any"
  type              = "ingress"
  from_port         = "80"
  to_port           = "80"
  protocol          = "tcp"
  cidr_blocks       = ["0.0.0.0/0"]
}

resource "aws_security_group_rule" "inbound_ssh_to_vm" {
  security_group_id = aws_security_group.vm.id
  description       = "Allow ssh protocol for any"
  type              = "ingress"
```

```

        from_port      = "22"
        to_port       = "22"
        protocol      = "tcp"
        cidr_blocks   = ["0.0.0.0/0"]
    }

resource "aws_security_group_rule" "vm_outbound_any" {
    security_group_id = aws_security_group.vm.id
    type             = "egress"
    from_port        = 0
    to_port          = 0
    protocol         = "all"
    cidr_blocks     = ["0.0.0.0/0"]
}

#Create network interface
resource "aws_network_interface" "main" {
    subnet_id      = "subnet-0e498a34c58db50e9" #subnet id for the specific
availability zone, e.g. us-east-1b - you will need to update it
    security_groups = [aws_security_group.vm.id]
    tags = {
        Name = "main-nic"
    }
}

#Create VM
resource "aws_instance" "CCD_demo" {
    ami           = "ami-0b0dcb5067f052a63" #Amazon Linux 2 AMI (for
us-east-1 N.Virginia) - you might need to update it if you deploy your
VM in a region other than us-east-1
    instance_type = "t2.micro"
    user_data     = file("./files/template.tpl") #User's startup shell
script - get content from files/template.tpl file
    key_name      = "your_keypair_name" #Use the name of YOUR key pair
    tags = {
        Name = "ccd-test-linux-vm"
    }
    network_interface {
        network_interface_id = aws_network_interface.main.id
        device_index         = 0
    }
}

```

Our computing infrastructure we are going to create and deploy is a single virtual web server using Amazon Linux OS to be deployed in us-east-1 (us-east-1b availability zone).

Step-by-step we perform the following:

- a) Create a new security group in us-east-1 Virtual Private Cloud (VPC);
- b) Add firewall rules to the security group for inbound and outbound traffic enabling: (i) inbound HTTP traffic (local port 80) to provide access to a web server on the VM, (ii) inbound SSH traffic (local port 22) to allow external management via SSH, and (iii) all outbound network connections.
- c) Create a new network interface which applies the above security group;
- d) Create a new Linux VM by: (i) defining VM image (ami), which correspond to Amazon Linux 2 OS; (ii) specifying instance type/size, i.e. hardware resources as t2.micro; (iii) providing path to user's startup shell script which will download

and install Apache web server and create a simple web page; (iv) attaching the networking interface created above.

Step 3: Updating key VM settings

In the above code you MUST replace the following settings with yours, which are different for different users and/or AWS regions:

```
key_name      = "your_keypair_name"  
vpc_id        = "vpc-075b0a07621fed9d6"  
subnet_id     = "subnet-0e498a34c58db50e9"  
ami           = "ami-0b0dc5067f052a63"
```

key_name:

You need to specify the name of YOUR key-pair (instead of "your_keypair_name") which you previously created and will need to use to connect to the VM via SSH.

Check your security key-pair name (or create a new one if you lost the .pem file) in the EC2 Dashboard:

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with options like 'New EC2 Experience', 'EC2 Dashboard', 'Events', 'Tags', 'Limits', and 'Instances' (with 'Instances' currently selected). The main area is titled 'Resources' and displays the following information:

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:					
Instances (running)	0	Dedicated Hosts	0	Elastic IPs	0
Instances	0	Key pairs	1	Load balancers	0
Placement groups	0	Security groups	8	Snapshots	0
Volumes	0				

ami, vpc_id, subnet_id:

You can notice desired ami, vpc_id and subnet_id in AWS EC2 console by going through 'Launch Instances Wizard'.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like 'New EC2 Experience', 'EC2 Dashboard', 'EC2 Global View', 'Events', 'Tags', 'Limits', and 'Instances' (which is currently selected). The main area has a search bar at the top and a table header for 'Instances Info' with columns for Name, Instance ID, Instance state, Instance type, and Status check. Below the table, it says 'No instances' and 'You do not have any instances in this region'. A red-bordered button labeled 'Launch instances' is visible.

AMI (OS image):

The screenshot shows the AWS AMI selection page. It features a 'Quick Start' section with icons for Amazon Linux, macOS, Ubuntu, Windows, and Red Hat. A magnifying glass icon leads to a 'Browse more AMIs' section. Below this, a specific AMI is highlighted: 'Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type'. The details show: ami-0b0dcb5067f052a63 (64-bit (x86)) / ami-01b5ec3ed8678d8b7 (64-bit (Arm)), Virtualization: hvm, ENA enabled: true, Root device type: ebs. A 'Free tier eligible' badge is present. The 'Description' section below lists 'Amazon Linux 2 Kernel 5.10 AMI 2.0.20221103.3 x86_64 HVM gp2'. The 'Architecture' dropdown is set to '64-bit (x86)' and the 'AMI ID' is 'ami-0b0dcb5067f052a63', which is marked as a 'Verified provider'.

Vpc_id:

Go to network settings and notice id for VPC (Virtual Private Cloud):

The screenshot shows the AWS Network settings page for a VPC. It displays the 'Network settings' section with a 'VPC - required' dropdown menu. The dropdown shows 'vpc-075b0a07621fed9d6' and '172.31.0.0/16' as the selected options, with '(default)' and a dropdown arrow next to it.

Subnet_id:

Click on 'Edit' -> 'Subnet' of 'Network settings' and notice the subnet_id:

The screenshot shows two instances of the 'Network settings' configuration page. In the top instance, the 'Edit' button is visible in the top right corner. Below it, the 'Subnet Info' section shows 'No preference (Default subnet in any availability zone)'. In the bottom instance, the 'VPC - required' dropdown is set to 'vpc-075b0a07621fed9d6 (default) 172.31.0.0/16'. The 'Subnet Info' dropdown is set to 'No preference'. A search bar at the top of the list shows 'No preference'. Below the search bar is a list of subnets:

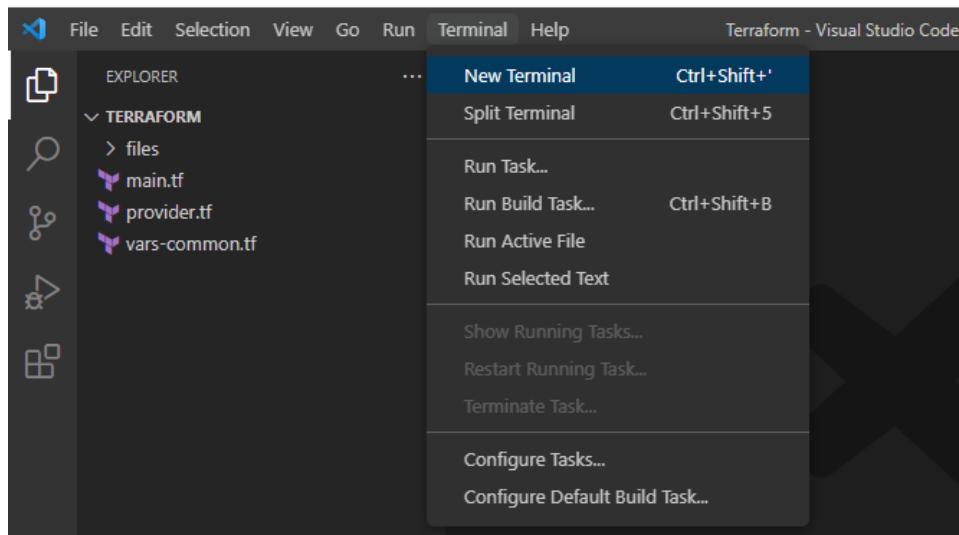
Subnet ID	VPC	Owner	Availability Zone	IP addresses available	CIDR
subnet-0e498a34c58db50e9	vpc-075b0a07621fed9d6	604421567424	us-east-1b	4090	172.31.80.0/20
subnet-0cde408e4499e13e1	vpc-075b0a07621fed9d6	604421567424	us-east-1f	4091	172.31.64.0/20
subnet-0bb1f0767a3eff1e4	vpc-075b0a07621fed9d6	604421567424	us-east-1a	4091	172.31.0.0/20
subnet-0654d4edb433da32e	vpc-075b0a07621fed9d6	604421567424	us-east-1d	4091	172.31.32.0/20
subnet-004693733c3f4af92	vpc-075b0a07621fed9d6	604421567424	us-east-1e	4091	172.31.48.0/20
subnet-03be9c5331dc01107	vpc-075b0a07621fed9d6	604421567424	us-east-1c	4091	172.31.16.0/20

If later you would like to deploy a Windows web server, you will need to change ami and use another startup script, e.g. the power shell script we used in one of previous labs and replace it in the user_data settings.

```
user_data    = file("./files/template.tpl")
```

Step 4: Configuring AWS credentials for AWS CLI

Open a New Terminal Window in Visual Studio Code and run the following commands.



1. First, check if the Terraform framework is installed (you can copy/paste the command; use right click to paste):

```
| _terraform -v
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under `TERRAFORM` containing `.terraform`, `files`, `template.tpl` (marked with a yellow exclamation), `main.tf` (selected), `provider.tf`, `vars-common.tf`, and `terraform.tfstate.backup`.
- Code Editor:** Displays the `main.tf` file with Terraform configuration code. The code creates a security group and two rules for inbound traffic to a VM.
- Terminal:** Shows the command `terraform -v` being run in a PowerShell window. The output indicates Terraform v1.2.6 is installed on windows_amd64, with provider `registry.terraform.io/hashicorp/aws` v4.44.0. It also notes that the version is out of date.
- Bottom Status Bar:** Shows the current file is `main.tf - Terraform - Visual Studio Code`, with status indicators for problems (1), output, debug console, terminal, and file tabs.

- Second, configure YOUR aws credentials (copy/paste your AWS access and secret keys when prompted from the .csv file you downloaded earlier):

```
aws configure
```

The screenshot shows the terminal window with the following output:

```
PS P:\Terraform> aws configure
AWS Access Key ID [None]: A[REDACTED]
AWS Secret Access Key [None]: ojT[REDACTED]W5
Default region name [None]: us-east-1
Default output format [None]:
PS P:\Terraform>
```

Step 5: Deploying AWS Virtual Infrastructure

Run the following commands:

```
terraform init
```

```
PS C:\Users\gorben01\Downloads\Terraform> terraform init
```

```
terraform validate
```

```
PS C:\Users\gorben01\Downloads\Terraform> terraform validate
```

```
terraform plan
```

You will see a list of actions to be performed to deploy your infrastructure:

```
PS C:\Users\gorben01\Downloads\Terraform> terraform plan
```

```
Terraform used the selected providers to generate the following execution plan.  
+ create
```

Terraform will perform the following actions:

```
# aws_instance.CCD_demo will be created  
+ resource "aws_instance" "CCD_demo" {  
    + ami                                = "ami-0b0dcb5067f052a63"  
    + arn                                = (known after apply)  
    + associate_public_ip_address        = (known after apply)  
    + availability_zone                  = (known after apply)  
    + cpu_core_count                    = (known after apply)  
    + cpu_threads_per_core              = (known after apply)  
    + disable_api_stop                 = (known after apply)  
    + disable_api_termination          = (known after apply)  
    + ebs_optimized                     = (known after apply)  
    + get_password_data                = false  
    + host_id                            = (known after apply)  
    + host_resource_group_arn           = (known after apply)  
    + iam_instance_profile              = (known after apply)  
    + id                                 = (known after apply)  
    + instance_initiated_shutdown_behavior = (known after apply)  
    + instance_state                   = (known after apply)  
    + instance_type                     = "t2.micro"  
    + ipv6_address_count               = (known after apply)  
    + ipv6_addresses                   = (known after apply)  
    + key_name                           = "CCD2022"
```

```
terraform apply
```

```
PS C:\Users\gorben01\Downloads\Terraform> terraform apply
```

Answer 'yes' when you are prompted to perform these actions.

Wait until you receive a message that the instance was successfully created:

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG

aws_security_group_rule.inbound_ssh_to_vm: Creation complete after 0s [id=sgrule-3241596016]aws
063306f5027c]
aws_instance.CCD_demo: Creating...
aws_security_group_rule.vm_outbound_any: Creation complete after 1s [id=sgrule-2661320798]
aws_security_group_rule.inbound_http_to_vm: Creation complete after 2s [id=sgrule-2008844741]
aws_instance.CCD_demo: Still creating... [10s elapsed]
aws_instance.CCD_demo: Still creating... [20s elapsed]
aws_instance.CCD_demo: Still creating... [30s elapsed]
aws_instance.CCD_demo: Creation complete after 33s [id=i-07cf576b2843c591f]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
PS C:\Users\gorben01\Downloads\Terraform>
```

Step 6: Checking Deployed Resources

1. Log in to AWS <https://aws.amazon.com/> and go to AWS console -> EC2 Dashboard and check your instances:

New EC Experience Tell us what you think

EC2 Dashboard

- EC2 Global View
- Events
- Tags
- Limits
- Instances
- Instances New

Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	1	Dedicated Hosts	0
Instances	1	Key pairs	3
Placement groups	0	Security groups	8
Volumes	1		

2. Check, that the web server has successfully started and the simple web page we created is available for browsing (make sure you use [http](http://) instead of [https](https://)):

New EC Experience Tell us what you think

EC2 Dashboard

EC2 Global View

Events

Tags

Limits

Instances

Instances New

Instance summary for i-00e9589b66236a1fa (ccd-test-linux-vm) [Info](#)

Updated less than a minute ago

Actions	Instance ID	Public IPv4 address	Private IPv4 addresses
Actions	i-00e9589b66236a1fa (ccd-test-linux-vm)	44.212.9.30 open address	172.31.92.19
	IPv6 address	Instance state	Public IPv4 DNS
	-	Running	ec2-44-212-9-30.compute-1.amazonaws.com open address
	Hostname type	Private IP DNS name (IPv4 only)	
	IP name: ip-172-31-92-19.ec2.internal	ip-172-31-92-19.ec2.internal	

http://ec2-44-212-9-30.compute-1.amazonaws.com

Welcome to the Cloud Computing Development Module!

This is your first AWS EC2 Linux Web Server

The screenshot shows a web page with the following content:

- Header: "LOAD TEST" and "RDS".
- Table: "Meta-Data" vs "Value".

Meta-Data	Value
InstanceId	i-00e9589b66236a1fa
Availability Zone	us-east-1b
- Text: "Current CPU Load: 0%".
- Text: "Prof. Anatoliy Gorbenko, KhAI/LBU".

3. Connect to your instance and run some commands in the terminal window:

The screenshot shows the "Connect to instance" dialog for the instance i-00e9589b66236a1fa. The dialog includes the following fields and options:

- Header: "EC2 > Instances > i-00e9589b66236a1fa > Connect to instance".
- Section: "Connect to instance" with a "Info" link.

Connect to your instance i-00e9589b66236a1fa (ccd-test-linux-vm) using any of these options
- Buttons: "EC2 Instance Connect" (selected), "Session Manager", "SSH client", and "EC2 serial console".
- Fields:
 - Instance ID: "i-00e9589b66236a1fa (ccd-test-linux-vm)".
 - Public IP address: "44.212.9.30".
 - User name: "ec2-user".
- Note: "Connect using a custom user name, or use the default user name ec2-user for the AMI used to launch the instance."
- Note box: "Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name."
- Buttons at the bottom: "Cancel" and "Connect" (orange).

The screenshot shows the AWS CloudWatch Terminal interface. At the top, there's a navigation bar with the AWS logo, 'Services' (selected), and a search bar. Below that is a header bar with a circular icon and the text 'Resource Groups & Tag Editor'. The main area is a terminal window displaying a shell session. The session starts with a prompt indicating the instance type: 'Amazon Linux 2 AMI'. It then shows standard Linux commands like 'ls', 'whoami' (outputting 'ec2-user'), and 'pwd' (outputting '/home/ec2-user'). The session ends with a closing bracket ']'.

```
Amazon Linux 2 AMI
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-92-19 ~]$ ls
[ec2-user@ip-172-31-92-19 ~]$ whoami
ec2-user
[ec2-user@ip-172-31-92-19 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-92-19 ~]$ ]
```

4. You can also connect to the instance using an SSH client similar to how you did in one of the first CCD labs (do not forget to provide a path to the .pem keypair file which you need to have on your desktop PC).

This screenshot shows the 'Connect to instance' page for an EC2 instance with ID i-00e9589b66236a1fa. The instance is associated with a group named 'ccd-test-linux-vm'. The page includes tabs for 'EC2 Instance Connect', 'Session Manager', 'SSH client' (which is selected and highlighted in orange), and 'EC2 serial console'. Below the tabs, there's a section for the instance ID with a copy icon. A numbered list provides instructions for connecting via SSH:

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is CCD.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 CCD.pem
4. Connect to your instance using its Public DNS:
ec2-44-212-9-30.compute-1.amazonaws.com

Below the instructions is an 'Example:' section with a copy icon and the command:

```
ssh -i "CCD.pem" ec2-user@ec2-44-212-9-30.compute-1.amazonaws.com
```

A note in a callout box states: 'Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.'

Step 7: Removing AWS EC2 Resources

You can terminate the instance from the AWS Console or directly from Visual Code by running (type ‘yes’ when prompted):

```
terraform destroy
```

```
PS C:\Users\gorben01\Downloads\Terraform> terraform destroy
```

Type ‘yes’ to confirm your choice.

Wait until you receive a message that your AWS resources have been successfully deleted.

```
Destroy complete! Resources: 6 destroyed.
```

```
PS C:\Users\gorben01\Downloads\Terraform>
```

Go to AWS EC2 Dashboard and check your resources.

Step 8: Advanced Self Task

Update the Terraform specification to create a Windows Web Server similar to one we created manually through the AWS GUI (see Week 2). Make sure you replace the startup bash script with the PowerShell script which installs the Internet Information Server on Windows and deploy a simple web page.

References

Watch this video if you want to install and use Terraform on your home PC/laptop:

<https://www.youtube.com/watch?v=hmKC6YagHqY>