Configuration Manual

MSc Research Project
Cybersecurity

Wei Tong
Student ID: X21202648

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Wei Tong

Student ID: X21202648

Programme: MSc. in Cybersecurity **Year:** 2022

Module: Final Thesis

Supervisor: Ross Spelman

Submission Due Date: 14th December 2023

Project Title: Cloud Native Application Disaster Recovery in a Multi-Cloud Environment – A DevOps Approach using Terraform

Word Count: 1016 **Page Count:** 10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Wei Tong

Date: 14th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Wei Tong
X21202648

1 Prerequisites

To orchestrate cloud resources in Amazon Web Service (AWS) and Microsoft Azure utilising the Infrastructure as Code (IaC) tool, the following environment and software will need to be installed:

1. Running environment – Windows Subsystem for Linux (WSL) or OS X Terminal or Linux Terminal.
2. Terraform – The Terraform binary package can be installed from one of these methods.
3. AWS Command Line Interface (CLI) - The AWS CLI is available for all the running environments in step 1, it can be installed from one of these methods here.
4. Azure CLI – The Azure CLI is also available for all running environments; it can be installed from one of these methods here.

Hardware requirements for the cloud resource orchestration environment:

- 4 CPU cores or more
- 8GB of RAM or more
- 10GB Hard Disk or more

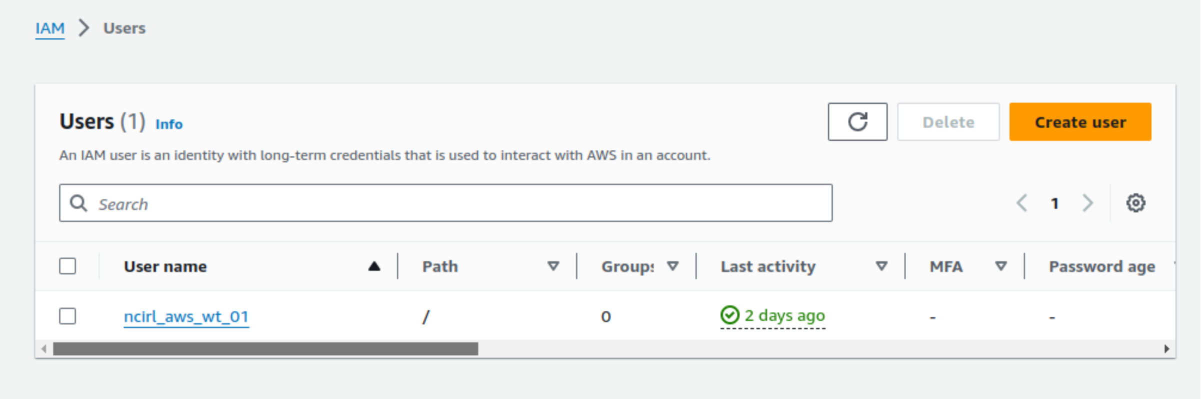
User accounts are created within AWS and Azure in order to provision services in the cloud environment.

2 Running Environment Setup

A few environment variables need to be set up correctly before provisioning services in AWS and Azure cloud. Terraform leverages AWS CLI and Azure CLI local settings to communicate and authenticate with AWS and Azure Application Programming Interface (API).

2.1 Setup AWS CLI

In the AWS console, create a Identify and Access Management (IAM) user, this user will be used by Terraform to perform infrastructure orchestration tasks, see Figure 1.



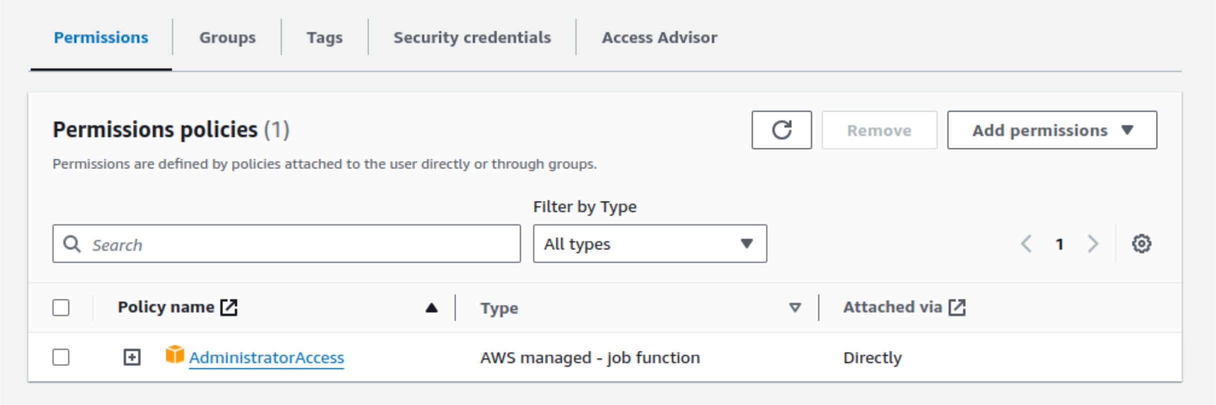
Users (1)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

☐	User name ▲	Path	Groups	Last activity	MFA	Password age
☐	ncirl_aws_wt_01	/	0	2 days ago	-	-

Figure 1. AWS IAM user

This IAM user should have the `AdministratorAccess` policy attached, so that it can perform cloud resource provisioning tasks, see Figure 2.



Permissions policies (1)

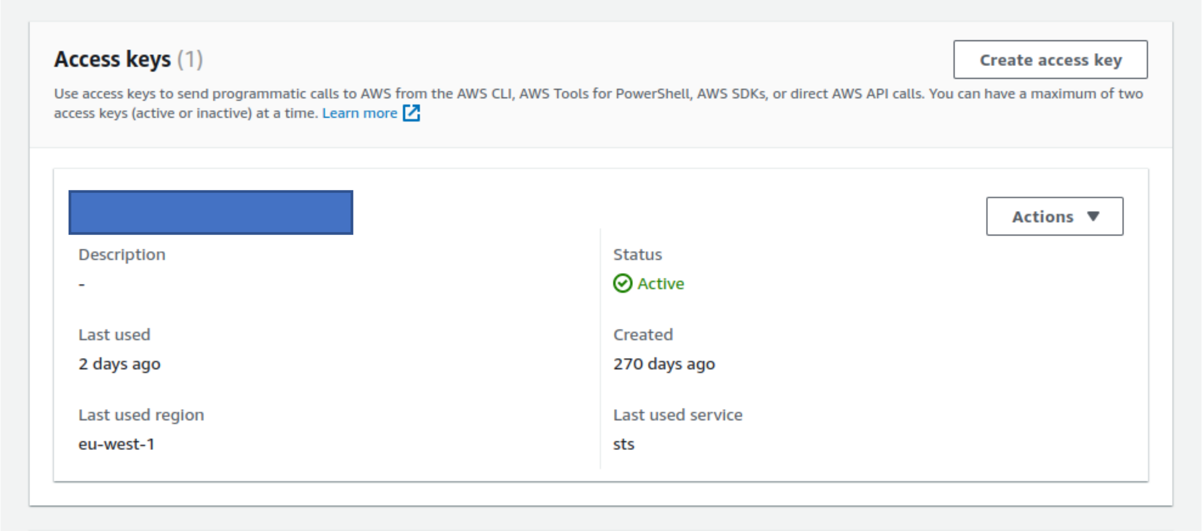
Permissions are defined by policies attached to the user directly or through groups.

Filter by Type

□	Policy name ▲	Type	Attached via
□	AdministratorAccess	AWS managed - job function	Directly

Figure 2. IAM user attached policy

Create a `Access key` and `Secret access key`, see Figure 3.



Access keys (1)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

[redacted]

Description: -

Status: Active

Last used: 2 days ago

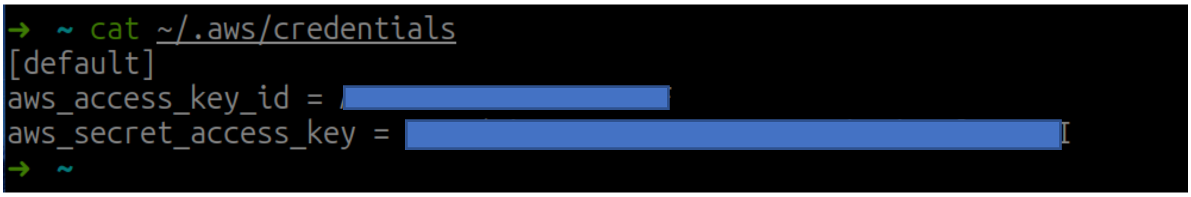
Created: 270 days ago

Last used region: eu-west-1

Last used service: sts

Figure 3. Access keys

Create a `credentials` file in your Terminal, copy and paste the access keys to the file, see Figure 4.



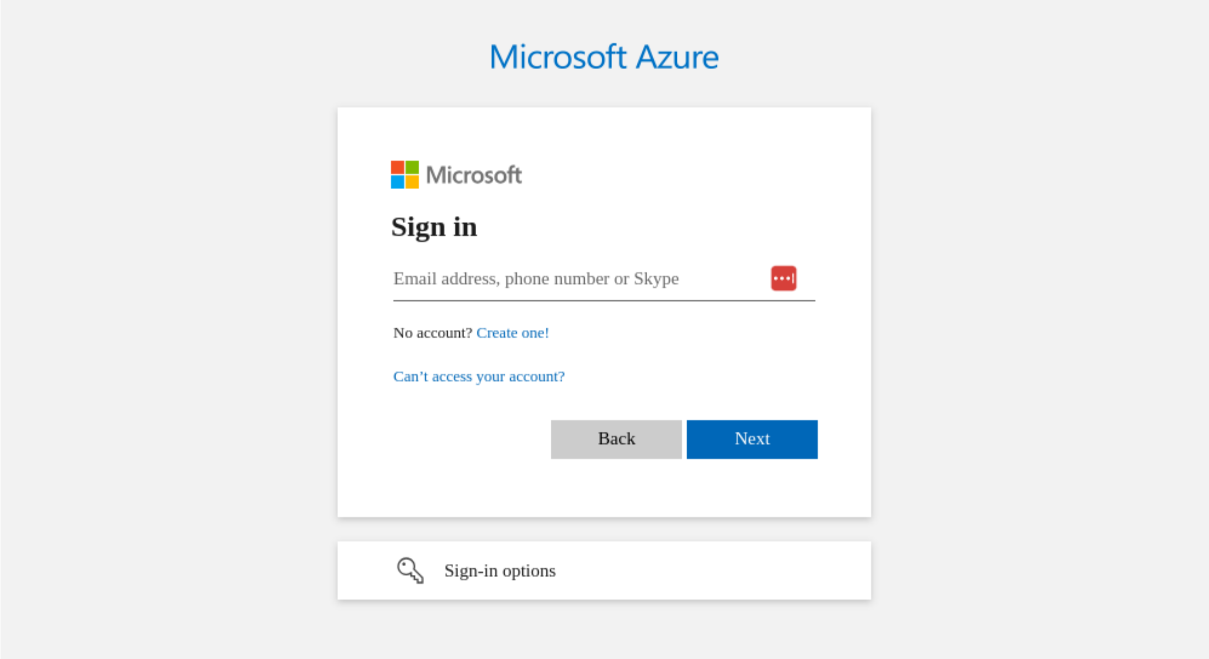
```
→ ~ cat ~/.aws/credentials
[default]
aws_access_key_id = [redacted]
aws_secret_access_key = [redacted]
→ ~
```

Figure 4. AWS CLI credentials file

Now, the AWS command should work in the terminal.

2.2 Setup Azure CLI

Run `az login` command in the terminal. It will prompt you to login to your Microsoft Azure account through the default browser, see Figure 5.



Sign in

Email address, phone number or Skype

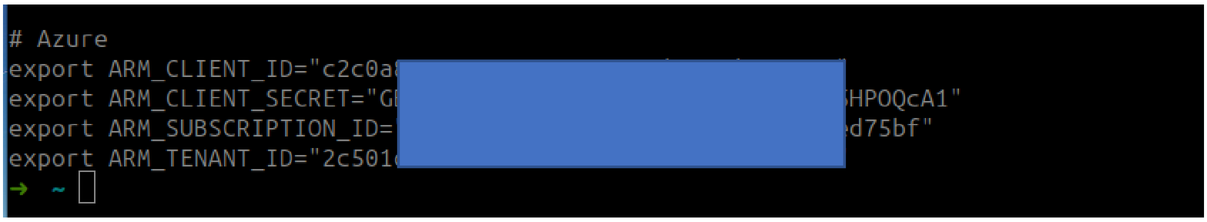
No account? Create one!

Can't access your account?

Figure 5. Microsoft Azure Login

Once logged in, the following information will be displayed in the terminal, `client id`, `client secret`, `subscription id` and `tenant id`.

Append the following information to the `.bashrc` file, see Figure 6.



```
# Azure
export ARM_CLIENT_ID="c2c0a[redacted]
export ARM_CLIENT_SECRET="G                                        HPOQcA1"
export ARM_SUBSCRIPTION_ID=                                        d75bf"
export ARM_TENANT_ID="2c501
→ ~ 
```

Figure 6. Azure CLI local settings

Now the Azure command should work in the terminal.

2.3 Terraform commands

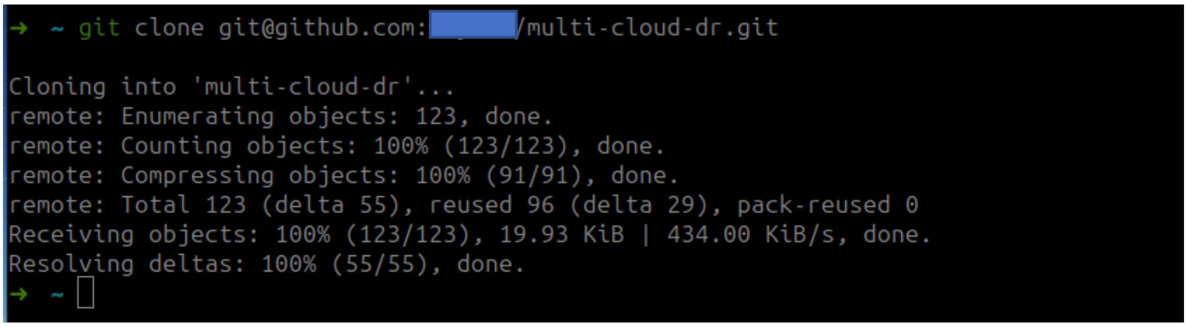
Once the AWS CLI and Azure CLI are configured, the following Terraform commands can run in both AWS and Azure Terraform modules.

- `terraform version` - Displays the current installed Terraform and Terraform provider versions.
- `terraform validate` - Validate all Terraform files in the current folder, and display any errors found.
- `terraform init` - Initialise the Terraform environment and install all the dependencies.
- `terraform plan` - Generate a Terraform execution plan for provisioning cloud resources.
- `terraform apply` - Generate a Terraform execution plan and provision the cloud resources based on the plan.
- `terraform output` - Print all the values from the output file.
- `terraform destroy` - Destroy all cloud resources according to the Terraform files.

- `terraform graph` - Generate a Terraform dependency graph based on the Terraform file.
- `terraform workspace new/delete/show/list` - Create and delete Terraform workspaces.
- `Terraform state list` - List all values form the Terraform state file.

3 AWS Infrastructure

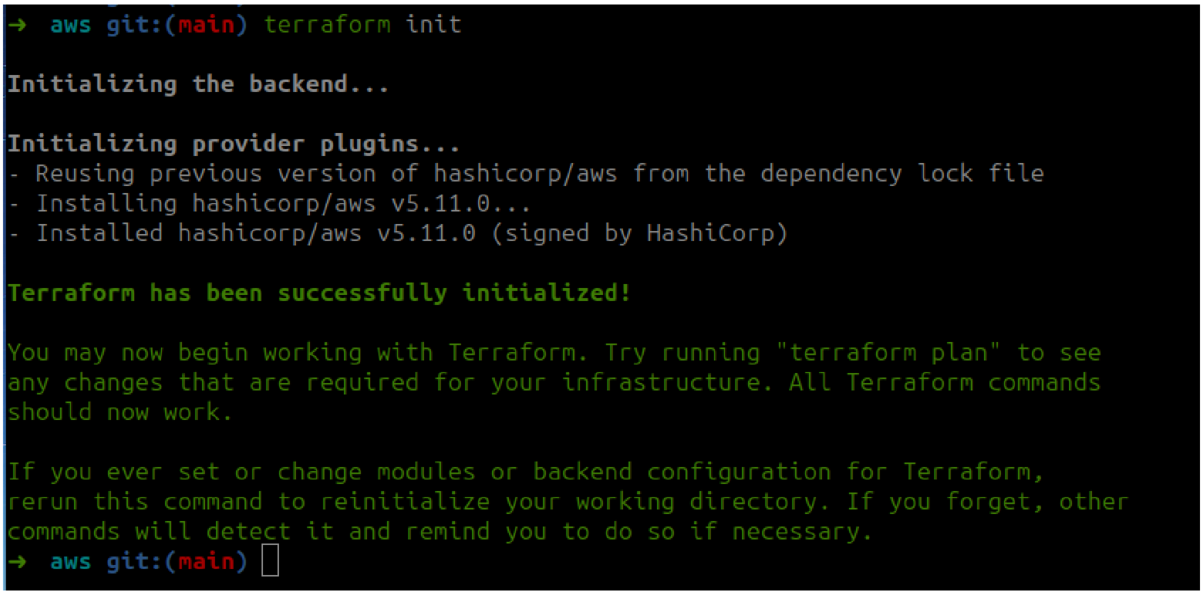
To provision AWS cloud resources, simply download the code form github.com, see Figure 7.



```
➜ ~ git clone git@github.com:[redacted]/multi-cloud-dr.git

Cloning into 'multi-cloud-dr'...
remote: Enumerating objects: 123, done.
remote: Counting objects: 100% (123/123), done.
remote: Compressing objects: 100% (91/91), done.
remote: Total 123 (delta 55), reused 96 (delta 29), pack-reused 0
Receiving objects: 100% (123/123), 19.93 KiB | 434.00 KiB/s, done.
Resolving deltas: 100% (55/55), done.
➜ ~ 
```

Figure 7. Clone Git Repository

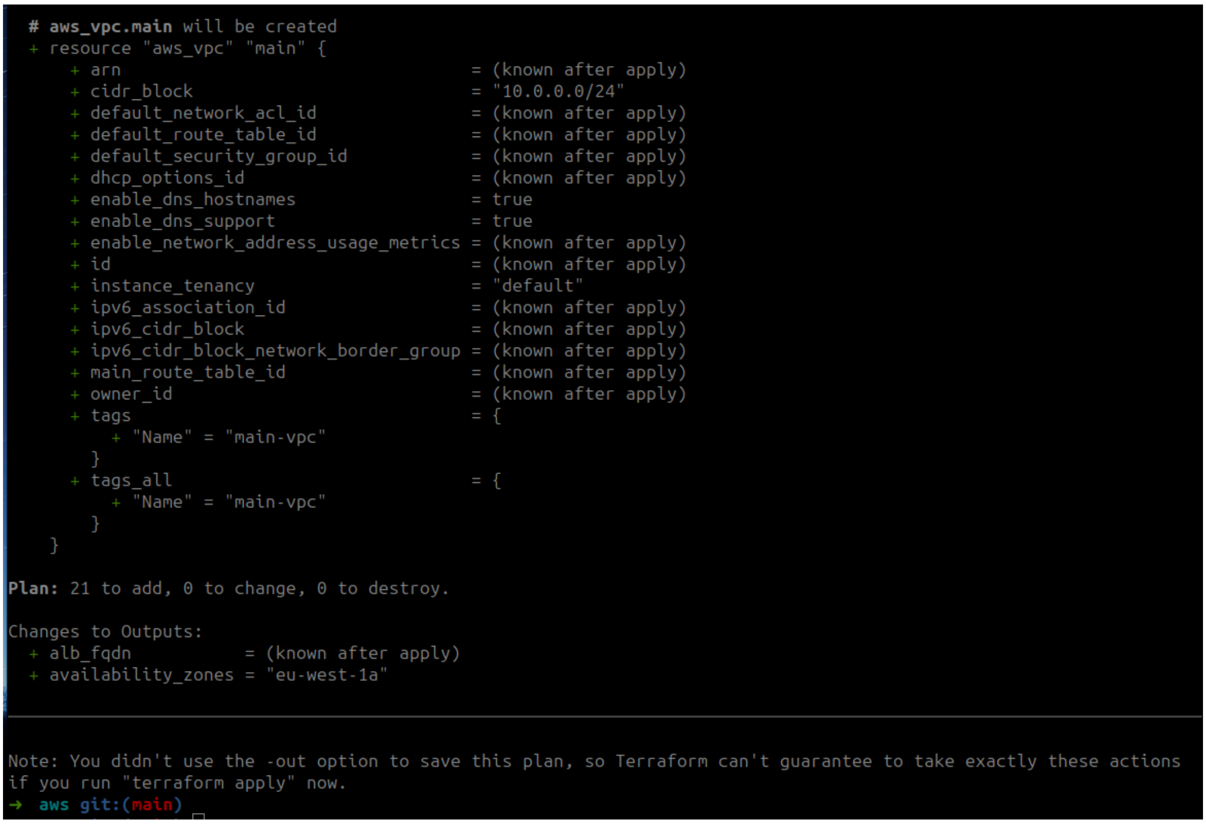
Initialise the Terraform environment by running `terraform init` command, see Figure 8.



```
➜ aws git:(main) terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Installing hashicorp/aws v5.11.0...
- Installed hashicorp/aws v5.11.0 (signed by HashiCorp)

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.
➜ aws git:(main) 
```

Figure 8. Terraform init

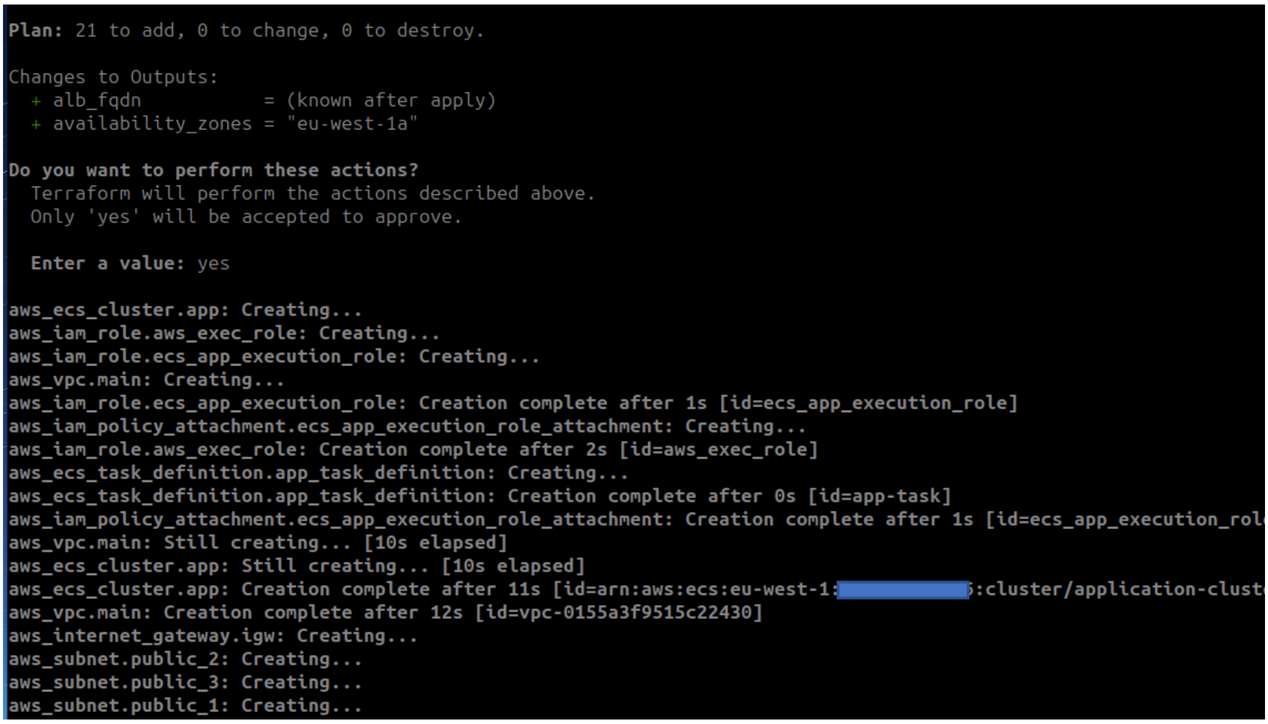
Generate a Terraform execution plan by running `terraform plan` command, see Figure 9.



```
  # aws_vpc.main will be created
  + resource "aws_vpc" "main" {
      + arn                                  = (known after apply)
      + cidr_block                           = "10.0.0.0/24"
      + default_network_acl_id               = (known after apply)
      + default_route_table_id               = (known after apply)
      + default_security_group_id            = (known after apply)
      + dhcp_options_id                      = (known after apply)
      + enable_dns_hostnames                 = true
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                   = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = "main-vpc"
        }
      + tags_all                             = {
          + "Name" = "main-vpc"
        }
    }

Plan: 21 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + alb_fqdn           = (known after apply)
  + availability_zones = "eu-west-1a"

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions
if you run "terraform apply" now.
➜  aws git:(main)
```

Figure 9. Terraform plan

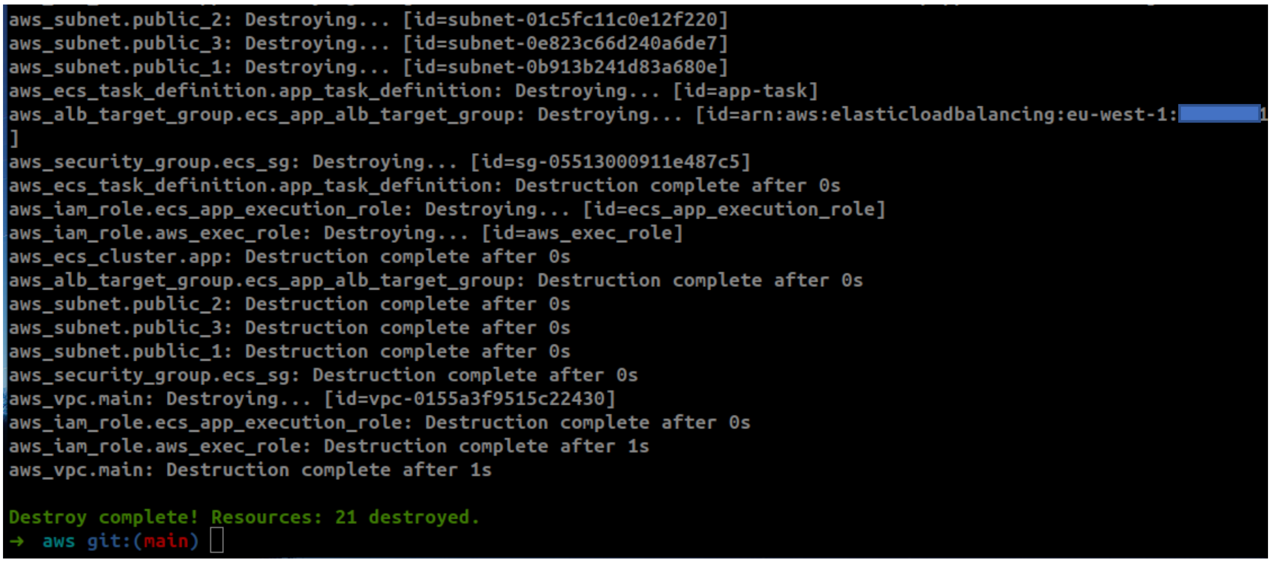
Orchestrate the infrastructure by running `terraform apply` command, see Figure 10.



```
Plan: 21 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + alb_fqdn           = (known after apply)
  + availability_zones = "eu-west-1a"

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_ecs_cluster.app: Creating...
aws_iam_role.aws_exec_role: Creating...
aws_iam_role.ecs_app_execution_role: Creating...
aws_vpc.main: Creating...
aws_iam_role.ecs_app_execution_role: Creation complete after 1s [id=ecs_app_execution_role]
aws_iam_policy_attachment.ecs_app_execution_role_attachment: Creating...
aws_iam_role.aws_exec_role: Creation complete after 2s [id=aws_exec_role]
aws_ecs_task_definition.app_task_definition: Creating...
aws_ecs_task_definition.app_task_definition: Creation complete after 0s [id=app-task]
aws_iam_policy_attachment.ecs_app_execution_role_attachment: Creation complete after 1s [id=ecs_app_execution_rol
aws_vpc.main: Still creating... [10s elapsed]
aws_ecs_cluster.app: Still creating... [10s elapsed]
aws_ecs_cluster.app: Creation complete after 11s [id=arn:aws:ecs:eu-west-1:[redacted]:cluster/application-clust
aws_vpc.main: Creation complete after 12s [id=vpc-0155a3f9515c22430]
aws_internet_gateway.igw: Creating...
aws_subnet.public_2: Creating...
aws_subnet.public_3: Creating...
aws_subnet.public_1: Creating...
```

Figure 10. Terraform apply.

Delete all resources by running `terraform destroy` command, see Figure 11.

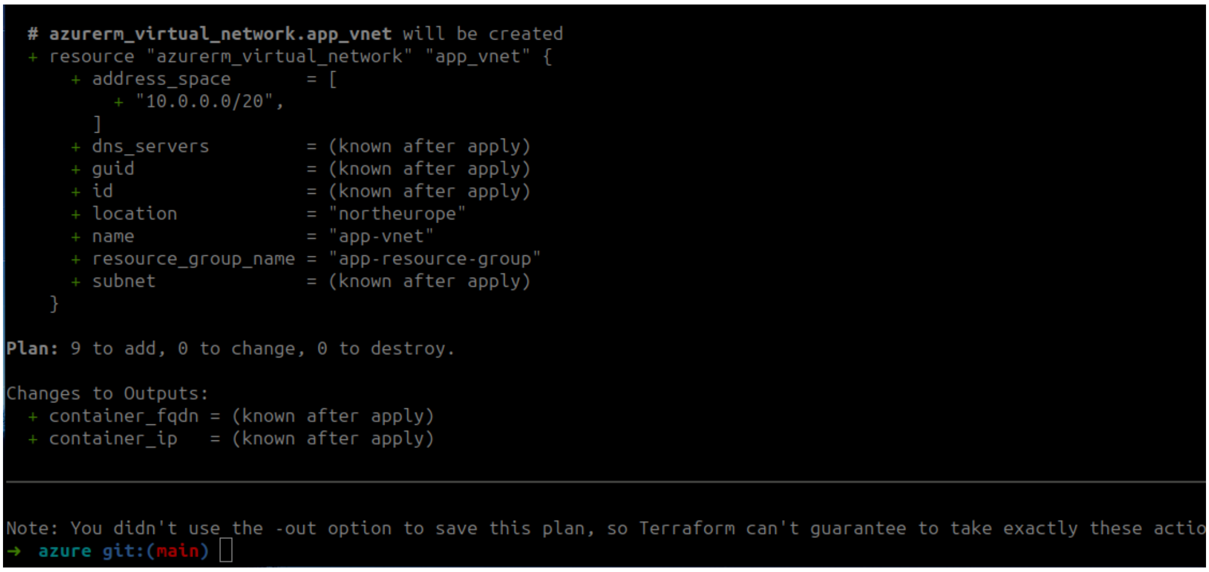


```
aws_subnet.public_2: Destroying... [id=subnet-01c5fc11c0e12f220]
aws_subnet.public_3: Destroying... [id=subnet-0e823c66d240a6de7]
aws_subnet.public_1: Destroying... [id=subnet-0b913b241d83a680e]
aws_ecs_task_definition.app_task_definition: Destroying... [id=app-task]
aws_alb_target_group.ecs_app_alb_target_group: Destroying... [id=arn:aws:elasticloadbalancing:eu-west-1:[redacted]
]
aws_security_group.ecs_sg: Destroying... [id=sg-05513000911e487c5]
aws_ecs_task_definition.app_task_definition: Destruction complete after 0s
aws_iam_role.ecs_app_execution_role: Destroying... [id=ecs_app_execution_role]
aws_iam_role.aws_exec_role: Destroying... [id=aws_exec_role]
aws_ecs_cluster.app: Destruction complete after 0s
aws_alb_target_group.ecs_app_alb_target_group: Destruction complete after 0s
aws_subnet.public_2: Destruction complete after 0s
aws_subnet.public_3: Destruction complete after 0s
aws_subnet.public_1: Destruction complete after 0s
aws_security_group.ecs_sg: Destruction complete after 0s
aws_vpc.main: Destroying... [id=vpc-0155a3f9515c22430]
aws_iam_role.ecs_app_execution_role: Destruction complete after 0s
aws_iam_role.aws_exec_role: Destruction complete after 1s
aws_vpc.main: Destruction complete after 1s

Destroy complete! Resources: 21 destroyed.
➜ aws git:(main)
```

Figure 11. Terraform destroy

4 Azure Infrastructure

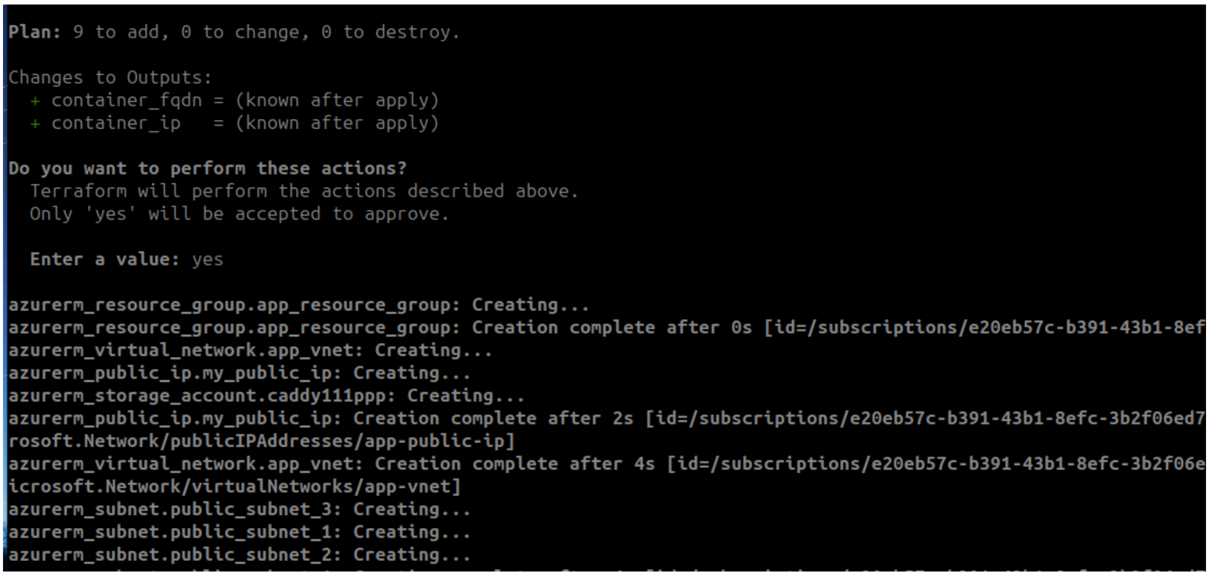
Similar to the AWS environment, run `terraform init` to initialise the Azure environment, and then run `terraform plan` to generate the execution plan for the Azure infrastructure, see Figure 12.



```
  # azurerm_virtual_network.app_vnet will be created
  + resource "azurerm_virtual_network" "app_vnet" {
      + address_space       = [
          + "10.0.0.0/20",
        ]
      + dns_servers         = (known after apply)
      + guid                = (known after apply)
      + id                  = (known after apply)
      + location            = "northeurope"
      + name                = "app-vnet"
      + resource_group_name = "app-resource-group"
      + subnet              = (known after apply)
    }

Plan: 9 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + container_fqdn = (known after apply)
  + container_ip   = (known after apply)

─────────────────────────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actio
➜ azure git:(main)
```

Figure 12. Terraform plan

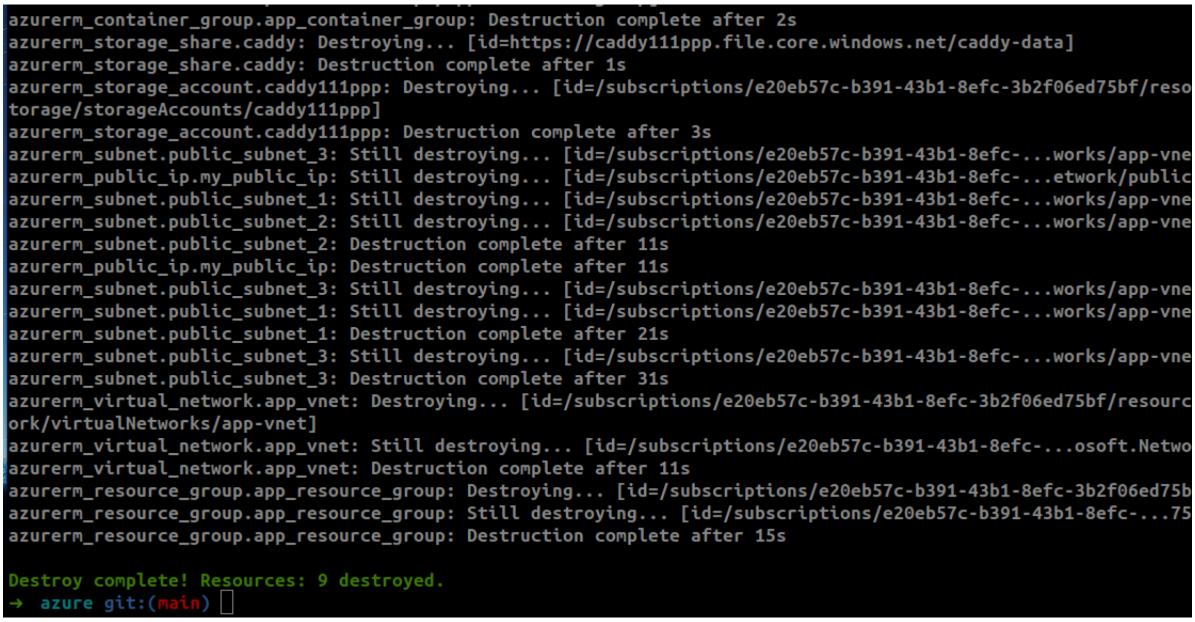
Run `terraform apply` to provision the resource in Azure, see Figure 13.



```
Plan: 9 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + container_fqdn = (known after apply)
  + container_ip   = (known after apply)

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

azurerm_resource_group.app_resource_group: Creating...
azurerm_resource_group.app_resource_group: Creation complete after 0s [id=/subscriptions/e20eb57c-b391-43b1-8ef
azurerm_virtual_network.app_vnet: Creating...
azurerm_public_ip.my_public_ip: Creating...
azurerm_storage_account.caddy111ppp: Creating...
azurerm_public_ip.my_public_ip: Creation complete after 2s [id=/subscriptions/e20eb57c-b391-43b1-8efc-3b2f06ed7
rosoft.Network/publicIPAddresses/app-public-ip]
azurerm_virtual_network.app_vnet: Creation complete after 4s [id=/subscriptions/e20eb57c-b391-43b1-8efc-3b2f06e
icrosoft.Network/virtualNetworks/app-vnet]
azurerm_subnet.public_subnet_3: Creating...
azurerm_subnet.public_subnet_1: Creating...
azurerm_subnet.public_subnet_2: Creating...
```

Figure 13. Terraform apply

Run `terraform destroy` to delete all the resource in Azure, see Figure 14.



```
azurerm_container_group.app_container_group: Destruction complete after 2s
azurerm_storage_share.caddy: Destroying... [id=https://caddy111ppp.file.core.windows.net/caddy-data]
azurerm_storage_share.caddy: Destruction complete after 1s
azurerm_storage_account.caddy111ppp: Destroying... [id=/subscriptions/e20eb57c-b391-43b1-8efc-3b2f06ed75bf/reso
torage/storageAccounts/caddy111ppp]
azurerm_storage_account.caddy111ppp: Destruction complete after 3s
azurerm_subnet.public_subnet_3: Still destroying... [id=/subscriptions/e20eb57c-b391-43b1-8efc-...works/app-vne
azurerm_public_ip.my_public_ip: Still destroying... [id=/subscriptions/e20eb57c-b391-43b1-8efc-...etwork/public
azurerm_subnet.public_subnet_1: Still destroying... [id=/subscriptions/e20eb57c-b391-43b1-8efc-...works/app-vne
azurerm_subnet.public_subnet_2: Still destroying... [id=/subscriptions/e20eb57c-b391-43b1-8efc-...works/app-vne
azurerm_subnet.public_subnet_2: Destruction complete after 11s
azurerm_public_ip.my_public_ip: Destruction complete after 11s
azurerm_subnet.public_subnet_3: Still destroying... [id=/subscriptions/e20eb57c-b391-43b1-8efc-...works/app-vne
azurerm_subnet.public_subnet_1: Still destroying... [id=/subscriptions/e20eb57c-b391-43b1-8efc-...works/app-vne
azurerm_subnet.public_subnet_1: Destruction complete after 21s
azurerm_subnet.public_subnet_3: Still destroying... [id=/subscriptions/e20eb57c-b391-43b1-8efc-...works/app-vne
azurerm_subnet.public_subnet_3: Destruction complete after 31s
azurerm_virtual_network.app_vnet: Destroying... [id=/subscriptions/e20eb57c-b391-43b1-8efc-3b2f06ed75bf/resourc
ork/virtualNetworks/app-vnet]
azurerm_virtual_network.app_vnet: Still destroying... [id=/subscriptions/e20eb57c-b391-43b1-8efc-...osoft.Netwo
azurerm_virtual_network.app_vnet: Destruction complete after 11s
azurerm_resource_group.app_resource_group: Destroying... [id=/subscriptions/e20eb57c-b391-43b1-8efc-3b2f06ed75b
azurerm_resource_group.app_resource_group: Still destroying... [id=/subscriptions/e20eb57c-b391-43b1-8efc-...75
azurerm_resource_group.app_resource_group: Destruction complete after 15s

Destroy complete! Resources: 9 destroyed.
➜  azure git:(main) 
```

Figure 14. Terraform destroy