National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Chidiebube Okeke
Student ID: x24109487

School of Computing
National College of Ireland

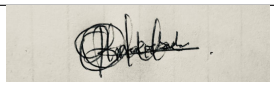
Supervisor: Yasantha Samarawickrama

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Chidiebube Okeke
Student ID:	x24109487
Programme:	Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Yasantha Samarawickrama
Submission Due Date:	11/12/2025
Project Title:	Configuration Manual
Word Count:	858
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:	
Date:	11th December 2025

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

Chidiebube Okeke
x24109487

1 Overview of the Experimental Environment

This document contains brief configuration manual describes all required tools, cloud resources, and environment parameters needed to fully reproduce the research project experiment "Benchmarking of Multi-Cloud Disaster Recovery (DR) using CIS-Benchmark".

This document outlines the neccessary setup to enable consistent reproduction of the following;

- Terraform (2025) Infrastructure-as-Code provisioning on AWS and Azure.
- Execution of three experimental variants (Basic, Hardened security).
- Integration of the Center for Internet Security (2025) Assessor for pre- and post-failover compliance scoring.
- Automated failover testing and data collection.

2 Experiment Requirements

The experiment requires the following software in the table are installed.

Tool	**Version**
Source Code	-
Terraform	$\geq$ 1.0
AWS CLI (2025)	v2.x
Azure CLI (2025)	Latest Stable Release
Bash Shell	-
Python	3.8+ (for parsing scripts)
Java(JRE) (2025)	Latest
CIS-CAT Lite Assessor	v4.56.0
VS Code (Optional) (2025)	-

Table 1: System Requirements and Prerequisites

2.1 Credentials

Authentication is required to perform cloud provisioning.

2.1.1 AWS setup using AWS Academy Lab

From CLI AWS Academy dashboard.

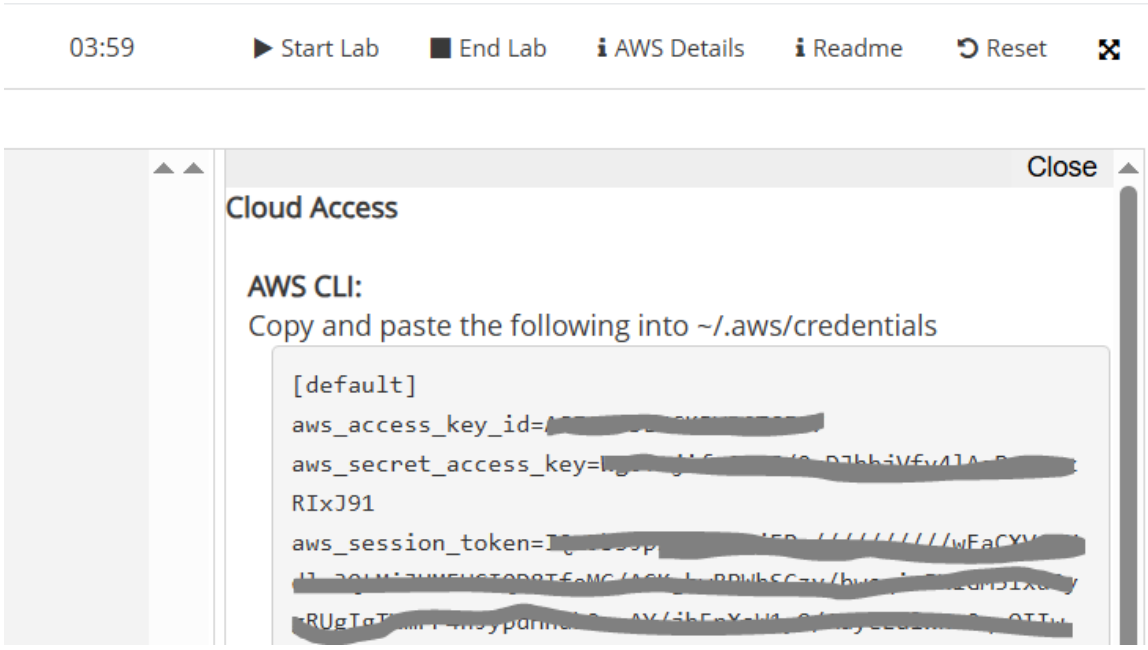


Figure 1: Example AWS Credential from AWS Academy

Follow the "Copy and paste the credentials to /.aws/credentials".

2.1.2 Azure for Students Subscription using Cloud NCI

Use a terminal and enter the command to authenticate.

```
$ az login
```

Use the command below to confirm current subscription.

```
$ az account show
```

Run each of the following commands below to get permission on the following Azure resource provider (2025) used in this experiment:

```
$ az provider register --namespace Microsoft.Network
$ az provider register --namespace Microsoft.Compute
$ az provider register --namespace Microsoft.Storage
$ az provider register --namespace Microsoft.DBforPostgreSQL
$ az provider register --namespace Microsoft.Insights
$ az provider register --namespace Microsoft.Monitor
$ az provider register --namespace Microsoft.ManagedIdentity
$ az provider register --namespace Microsoft.Authorization
$ az provider register --namespace Microsoft.Resources
```

2.2 SSH Key pair

Each cloud uses its own key pair. In your terminal
For AWS:

```
$ aws ec2 create-key-pair --key-name dr-test-key > dr-test-key.pem
```

For Azure:
Run the command below that creates to keys(public, and private)

```
$ ssh-keygen -t rsa -b 4096 -f  ~/.ssh/azure-dr-key
```

Verfiy key generation

2.3 CIS-CAT Storage Configuration

CIS-CAT Lite zip artifact must be uploaded to cloud storage so that instances can download it during initialization. Open terminal, and change directory to where CIS-CAT-Lite.zip is located.
For AWS
Make S3 Bucket using;

```
$ aws s3 mb s3://cis-cat-bucket
```

Go to the created bucket on AWS console, change public access, and add the bucket policy(see image below)

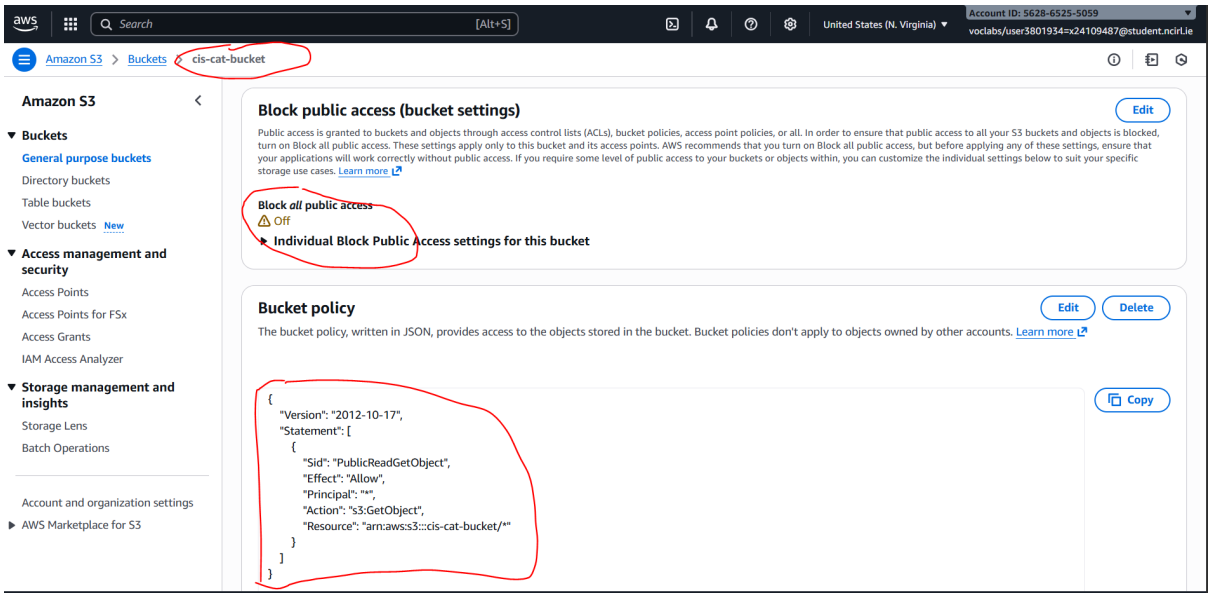


Figure 2: Bucket policy for bucket

Upload a zipped CIS CAT artifact

```
$ aws s3 cp CIS-CAT-Lite.zip s3://cis-cat-bucket/ --acl public-read
```

Verify Upload

```
$ aws s3 ls s3://cis-cat-bucket/
```

For Azure
In your terminal, run the commands:
To create a resource group for storage

```
$ az group create --name dr-storage-rg --location centralus
```

Create storage account

```
$ az storage account create \
    --name ciscatstorage \
    --resource-group dr-storage-rg \
    --location eastus \
    --sku Standard_LRS
```

Create container or blob

```
$ az storage blob upload \
    --account-name ciscatstorage \
    --container-name ciscat \
    --name CIS-CAT-Lite.zip \
    --file CIS-CAT-Lite.zip
```

Verify upload

```
$ az storage blob list \
    --account-name ciscatstorage \
    --container-name ciscat --output table
```

Note: These two storage buckets are used throughout the experimentation.

3 Codebase Configuration (AWS, and Azure)

The research includes three variants, each reproduced identically on AWS and Azure
Open Terminal, and change directory to the root folder of the codebase.
The codebase root directory consists of 3 folders namely, aws, azure, and scripts.

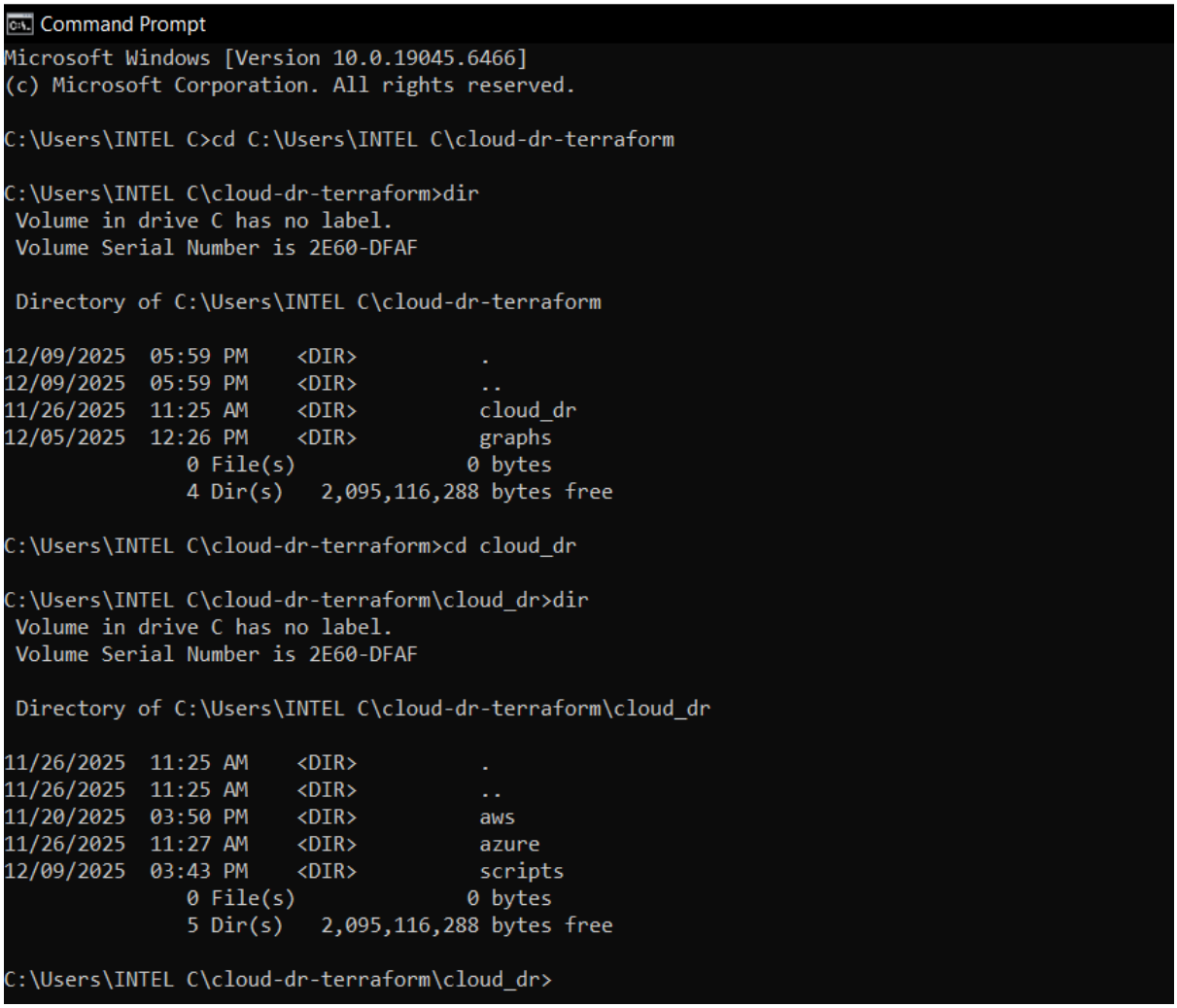


Figure 3: Code to change to root project directory

The codebase containing the deployment Terraform files and script should be downloaded, extracted, and opened in your code editor(suggest VS Code). See image below.

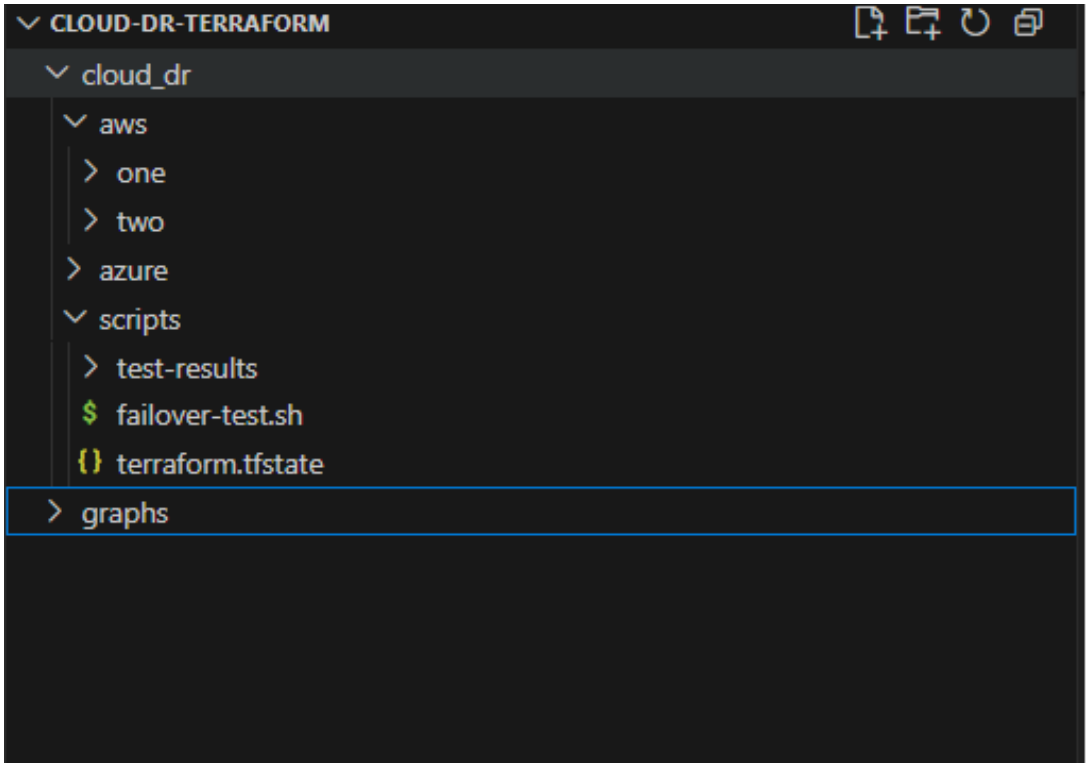


Figure 4: Codebase Folder Structure

(aws or azure)/one - Variant 1 (Basic Security)
(aws or azure)/two - Variant 2 (Hardened Security)

The following workflow was used for every experimental run. Reproduction must follow this sequence exactly.

3.1 Terraform Deployment

Run the following command sequentially, and wait for each command to run successfully.

```
$ cd <cloud\_type | aws or azure>/<variant | one or two>

$ terraform init

$ terraform plan

$ time terraform apply -auto-approve

$ terraform output > outputs.txt
```

For example - AWS Variant 1, see image below:

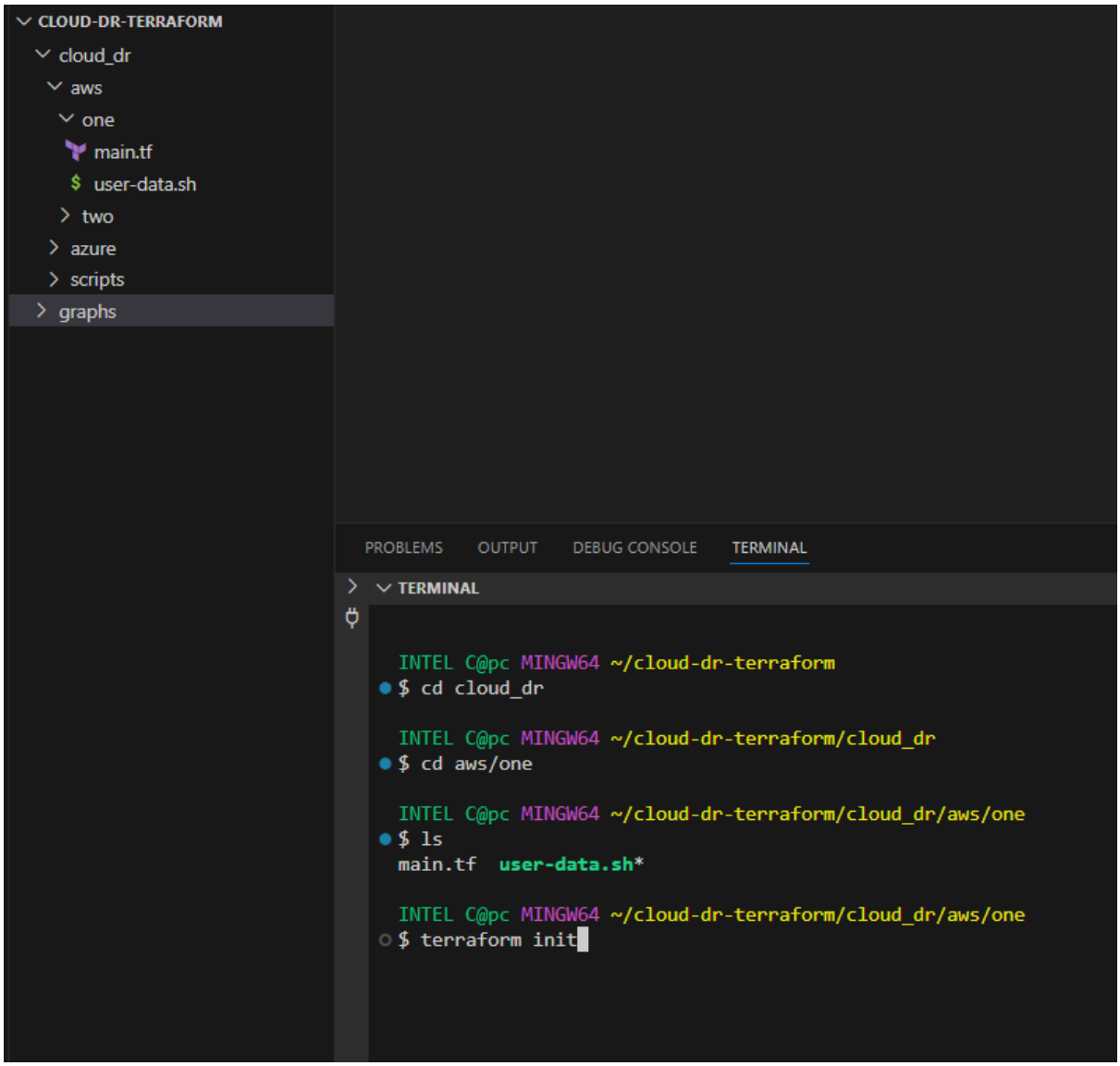


Figure 5: Command to Initialize Terraform Deployment

Go To AWS Console to verify cloud resources e.g EC2, RDS resources
Verify service health via terminal:

```
$ curl http://<load-balancer-dns>/health
```

It should return OK
Check outputs.txt for value of "load-balancer-dns"

3.2 Automated Failover Test Script

Copy AWS SSH dr-test-key.pem to scripts folder (see step 1, 2, and in the image)
All failover experiments uses "scripts/failover-test.sh"
For failover script execution(see step 4 in the image below)
After the script runs successfully, the result(RTO's, CIS Compliance score e.t.c) are logged in the "test-result" folder(see step 5 in the image)

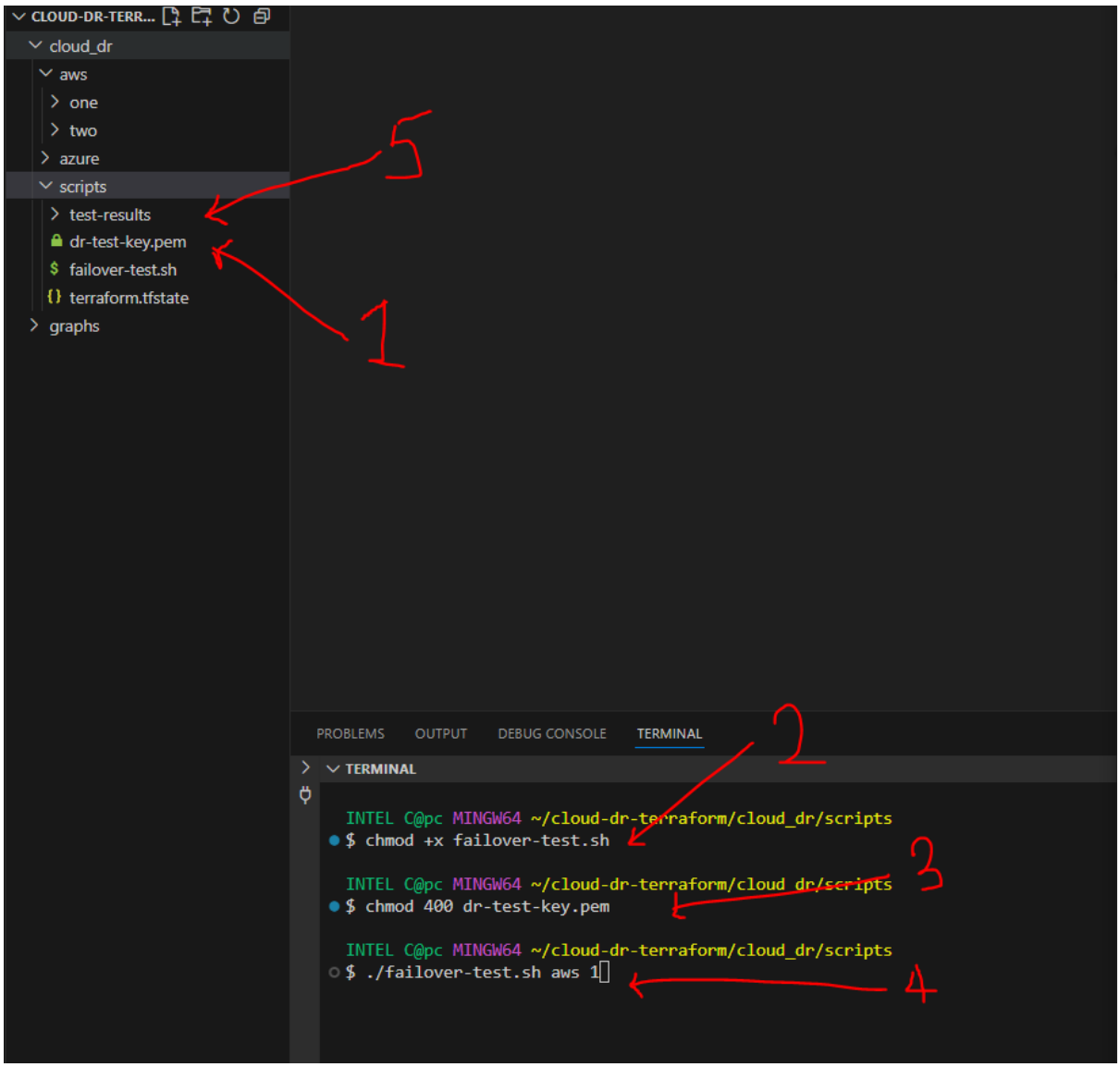


Figure 6: Code to change to root project directory

The results are collected manually into a file in the format

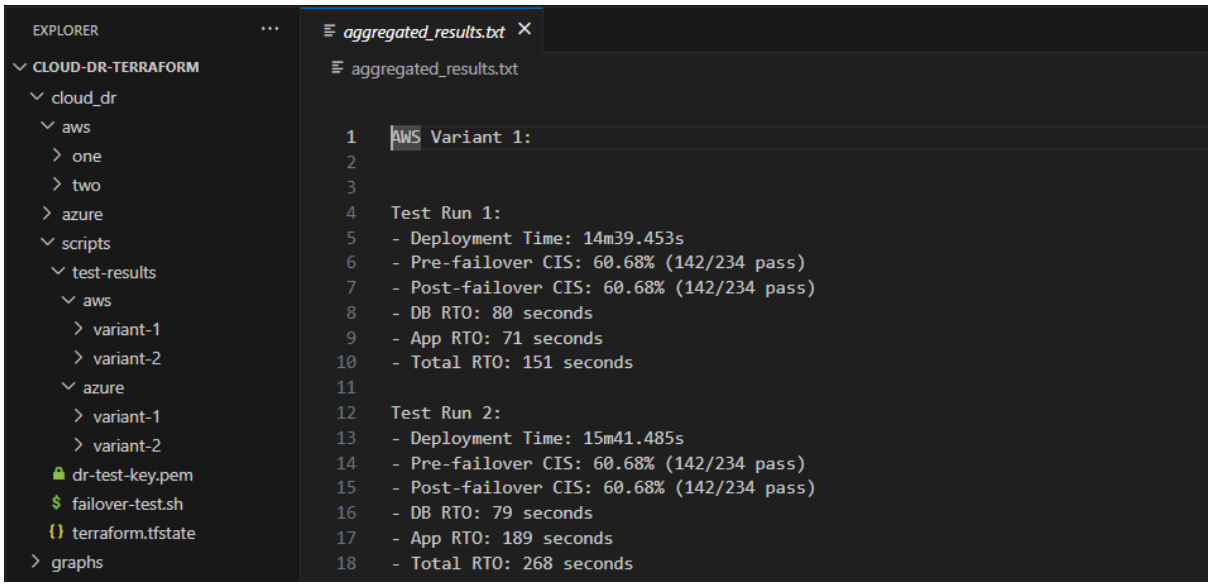


Figure 7: Code to change to root project directory

4 Cleanup Requirements

After each test run, and acquiring results:
In your terminal, run the command below

```
$ time terraform destroy -auto-approve
```

Wait for a few period of time to perform the next experiments.

References

VS Code (Optional) (2025). Visual studio code. Optional source code editor used during development.
URL: *https://code.visualstudio.com/*

AWS CLI (2025). Aws command line interface. Used for AWS resource management and automation.
URL: *https://aws.amazon.com/cli/*

Azure CLI (2025). Azure command-line interface (azure cli). Used for managing Azure resources and services.
URL: *https://learn.microsoft.com/cli/azure/*

Azure resource provider (2025). Azure resource manager. Used for managing Azure resources and services.
URL: *https://learn.microsoft.com/en-us/cli/azure/provider*

Center for Internet Security (2025). Cis-cat lite assessor. Used to perform CIS Benchmark security compliance assessments.
URL: *https://www.cisecurity.org/cis-cat-lite*

Java(JRE) (2025). Java runtime environment. Required to execute CIS-CAT Lite Assessor.
URL: *https://www.java.com/en/download/*

Terraform (2025). Terraform. Infrastructure as Code tool used for multi-cloud provisioning.
URL: *https://www.terraform.io/*