Daniel Allison

danielallison77@gmail.com

n01465744

Assignment 1

CCGC 5502 Automation – Humber College

Table of Contents

[**Phase 1: Development** 3](#_Toc2)

[Setup 3](#_Toc3)

[GitHub Repo 3](#_Toc4)

[Cocogitto 4](#_Toc5)

[backend.tf and providers.tf 5](#_Toc6)

[terraform init 6](#_Toc7)

[Project Metadata (root variables.tf) 7](#_Toc8)

[Resource Group Child Module 8](#_Toc9)

[Definition 8](#_Toc10)

[Dev Test: Validation, Provisioning, Verifying in Azure Portal 9](#_Toc11)

[Release 10](#_Toc12)

[Network Child Module 11](#_Toc13)

[Definition 11](#_Toc14)

[Dev Test: Validation, Provisioning, Verifying in Azure Portal 14](#_Toc15)

[Release 15](#_Toc16)

Table of Figures

[SCREENSHOT 1: The mostly empty initialized code repository in GitHub 3](#_Toc2)

[SCREENSHOT 2: cog.toml, the configuration file for cocogitto, a conventional commits tool 4](#_Toc3)

[SCREENSHOT 3: backend.tf and providers.tf 5](#_Toc4)

[SCREENSHOT 4: terraform init and then again with migrate state to the new backend tfstate key 6](#_Toc5)

[SCREENSHOT 5: variables.tf in the root module showing the project\_metadata variable, used for tagging every resource 7](#_Toc6)

[SCREENSHOT 6: the resource group child module’s main.tf, variables.tf, and outputs.tf 8](#_Toc7)

[SCREENSHOT 7: terraform validate shows valid configuration for newly added resource group 9](#_Toc8)

[SCREENSHOT 8: terraform state list | nl shows the deployed resource group 9](#_Toc9)

[SCREENSHOT 9: Azure Portal shows the assignment1.tfstate after resource group deployed 9](#_Toc10)

[SCREENSHOT 10: Azure Portal shows deployed resource group with tags 10](#_Toc11)

[SCREENSHOT 11: the network child module's main.tf, with a vnet, subnet, nsg, and net sec rules 11](#_Toc12)

[SCREENSHOT 12: network child module’s variables.tf with variables for vnet, subnet, nsg, local module and project metadata 12](#_Toc13)

[SCREENSHOT 13: network child module's outputs.tf with vnet and subnet names 13](#_Toc14)

[SCREENSHOT 14: terraform validate shows valid configuration for newly added network module 14](#_Toc15)

[SCREENSHOT 15: terraform state list shows 9 resources deployed after network module added 14](#_Toc16)

[SCREENSHOT 16: Azure Portal shows assignment1.tfstate after network module deployed 14](#_Toc17)

[SCREENSHOT 17: Azure Portal shows deployed network security group with open ports 22, 3389, 5985, and 80 15](#_Toc18)

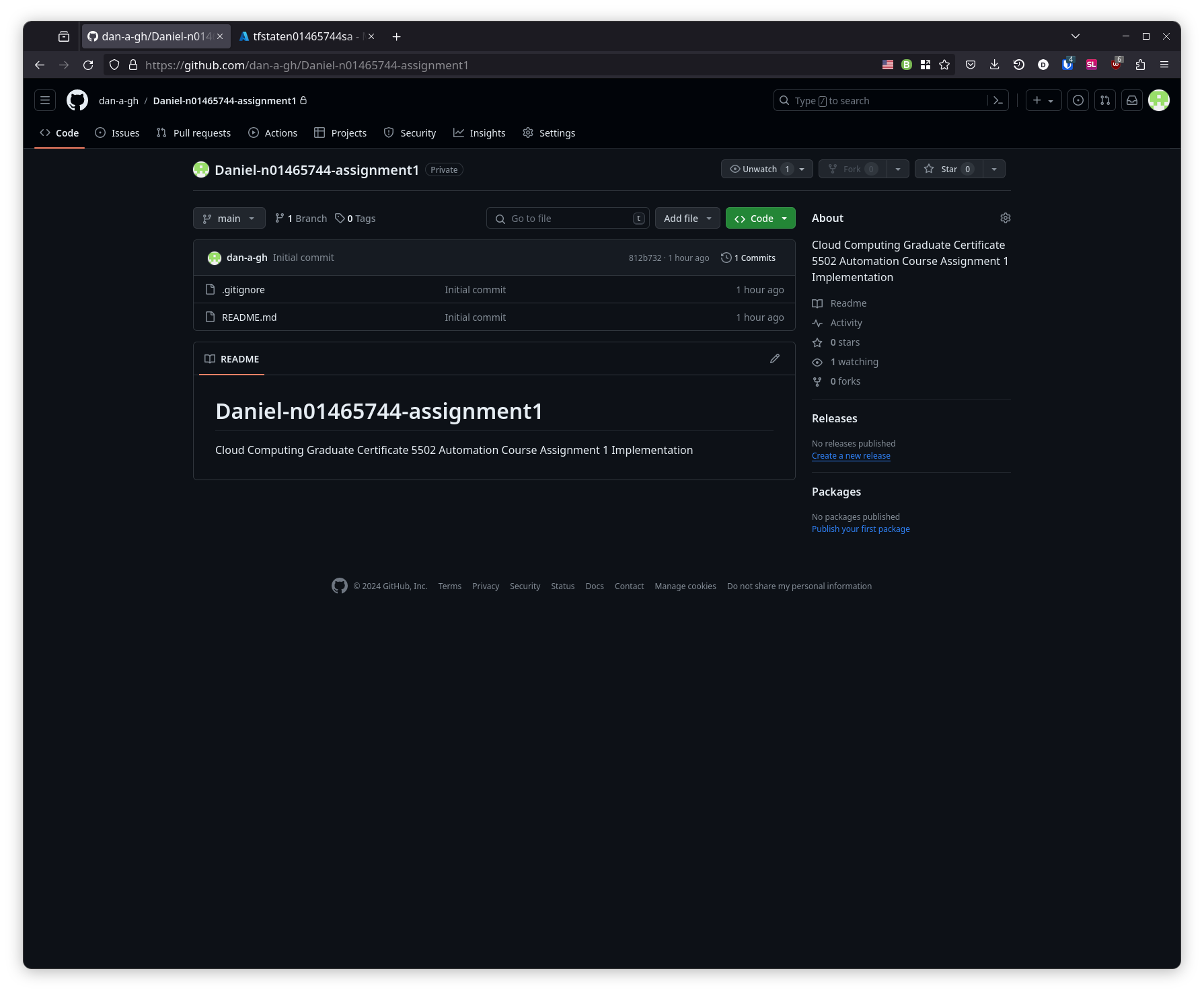
# **Phase 1: Development**

## Setup

### GitHub Repo

The first step I took to start this assignment was to create a GitHub repository in which the code could be stored and shared. It was initialized with a Terraform specific .gitignore and a README.md file.

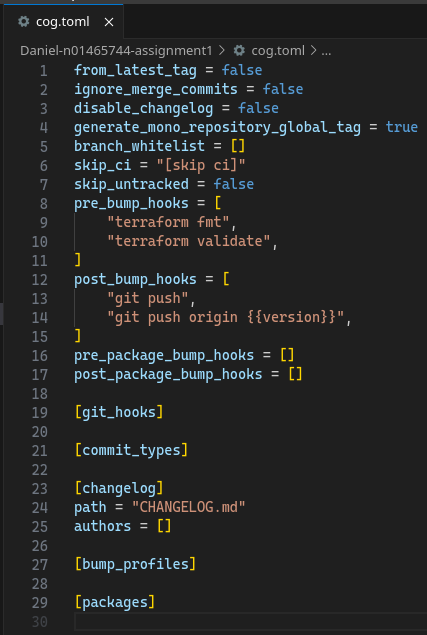
SCREENSHOT 1: The mostly empty initialized code repository in GitHub



### Cocogitto

Cocogitto is a tool to create conventional commits, which is a standard to help organize git commit messages. It can also manage versioning based on what type of work was identified in the commit messages since the last version was tagged. Since I already had the tool installed, setup involved running the command cog init, and filling out the cog.toml configuration file with commands that use git hooks to run before/after bumping the version number.

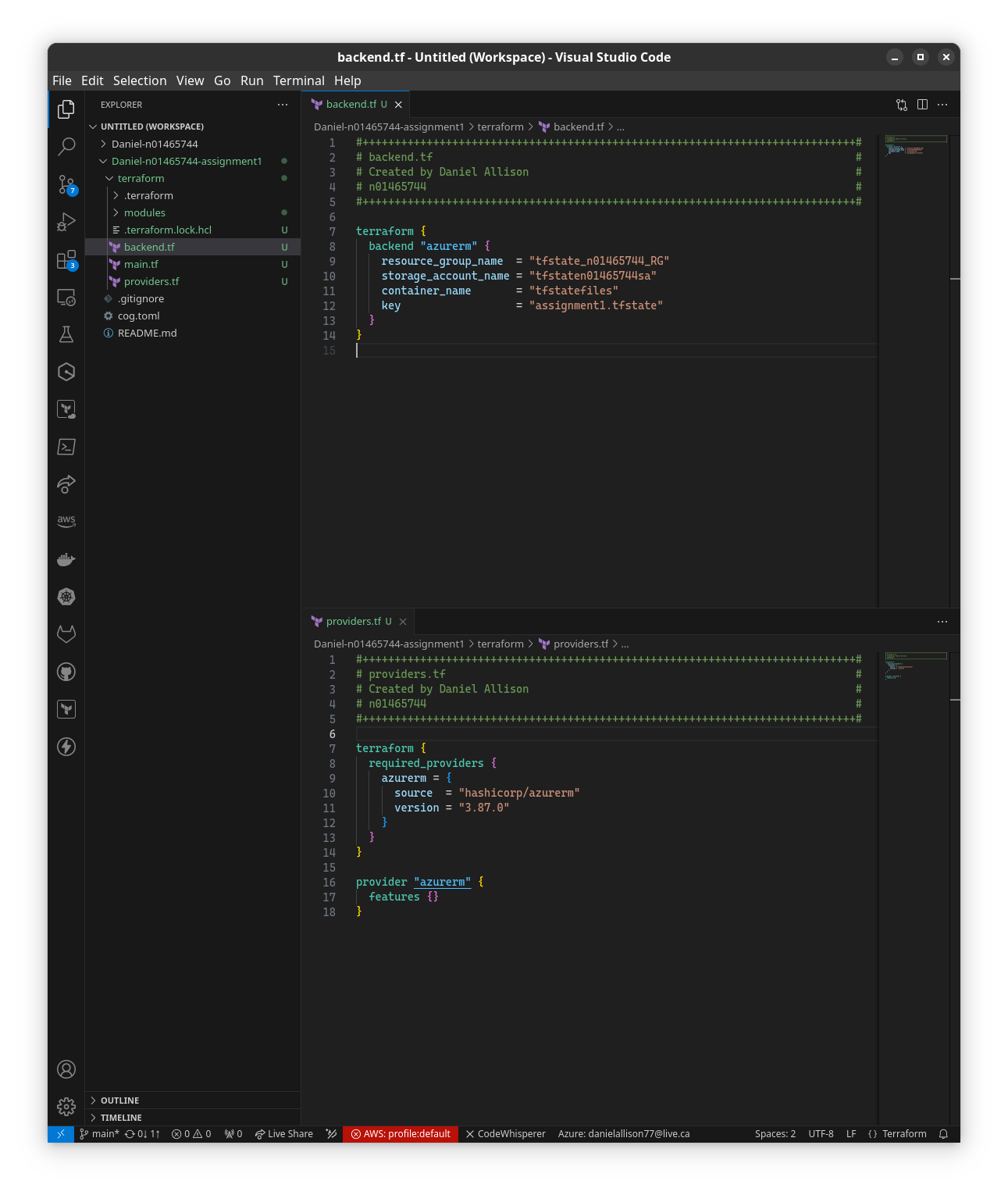
SCREENSHOT 2: cog.toml, the configuration file for cocogitto, a conventional commits tool



### backend.tf and providers.tf

The backend.tf and providers.tf files were next to be created. I drew heavy inspiration from work performed in lab 05. I changed the name of the key once I realized that it was going to overwrite tfstate from the labs if no changes were made.

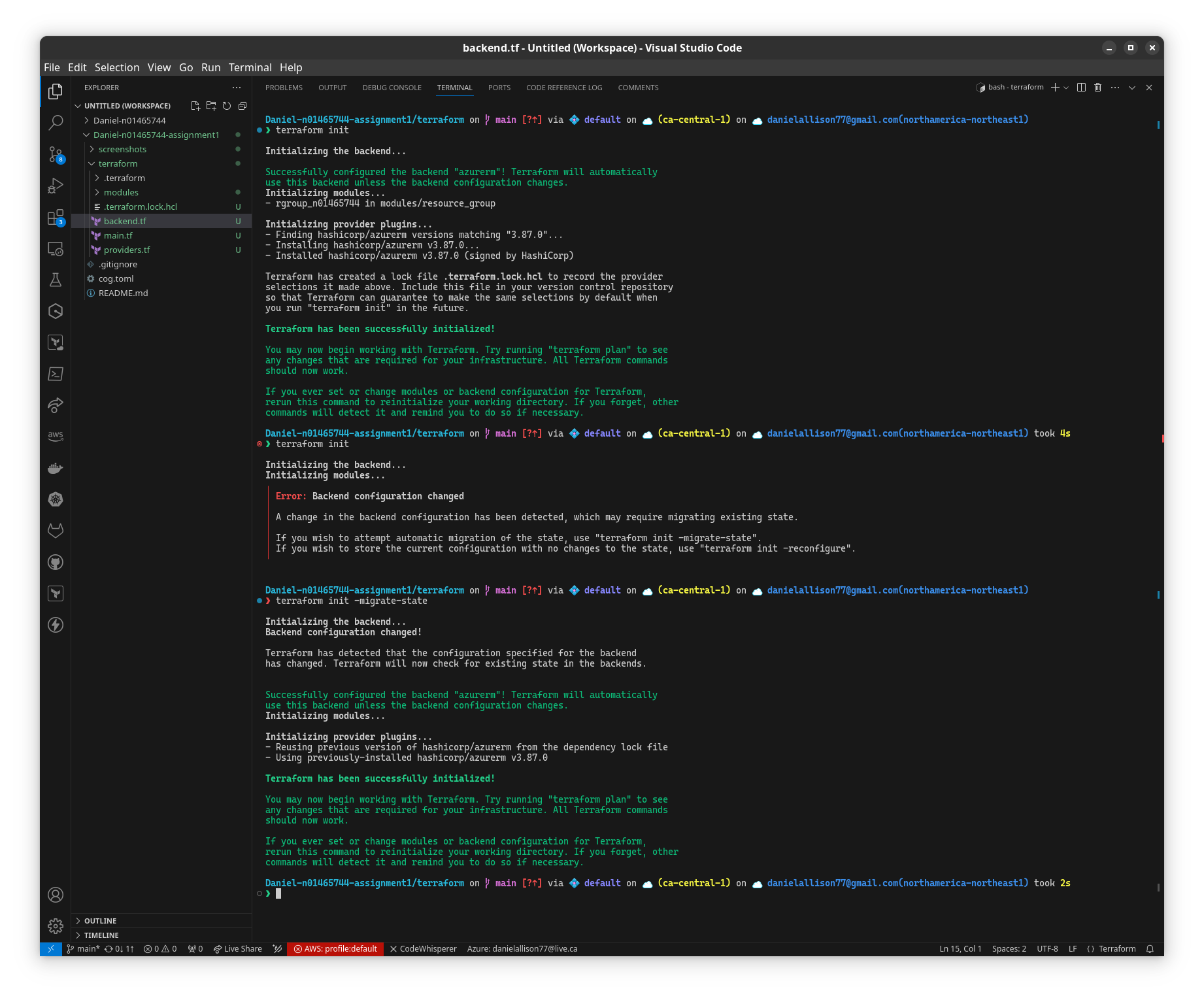
SCREENSHOT 3: backend.tf and providers.tf



### terraform init

With the bare minimum files in place to initialize, I went ahead and tested my code with terraform init. The first time was with the old key tfstate that would have overwritten the labs work, the second successful time with the migrate state option uses the new backend tfstate key.

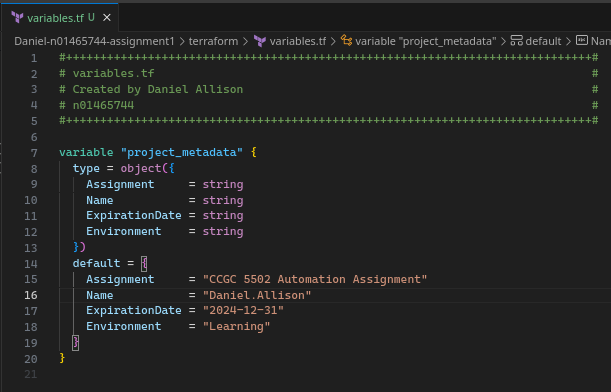
SCREENSHOT 4: terraform init and then again with migrate state to the new backend tfstate key



### Project Metadata (root variables.tf)

For the data used in tags common to all resources, I have created a variables.tf at the root level with one object variable called “project\_metadata”. I chose this approach over using locals because locals are meant to be used within a module, like a function’s temporary local variables in traditional programming languages.[[1]](#footnote-3)

SCREENSHOT 5: variables.tf in the root module showing the project\_metadata variable, used for tagging every resource

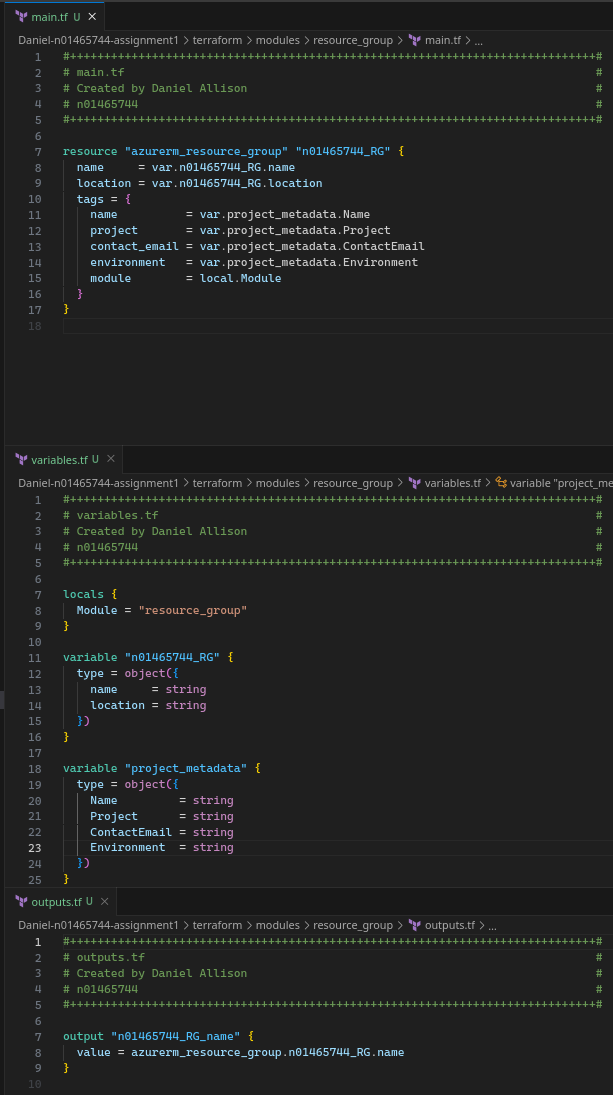


## Resource Group Child Module

### Definition

The resource group resource is defined in main.tf with parametrized fields, tagged with project metadata, corresponding variables are declared in variables.tf, and outputs.tf returns the resource group name.

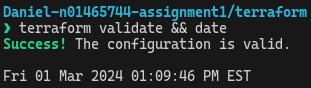
SCREENSHOT 6: the resource group child module’s main.tf, variables.tf, and outputs.tf



### *Dev Test: Validation, Provisioning, Verifying in Azure Portal*

This module was tested with terraform validate, plan, and deploy.

SCREENSHOT 7: terraform validate shows valid configuration for newly added resource group

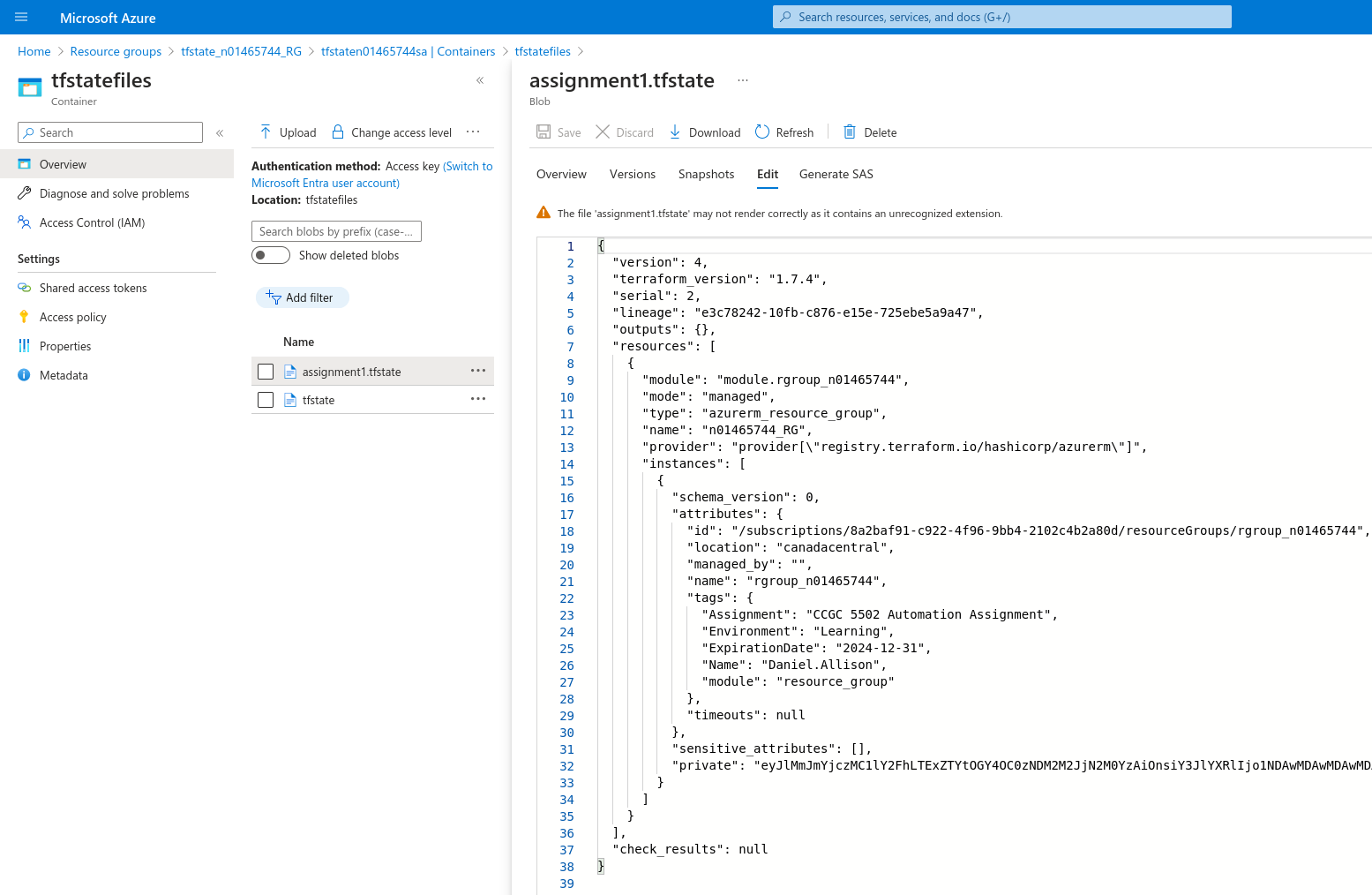


SCREENSHOT 8: terraform state list | nl shows the deployed resource group

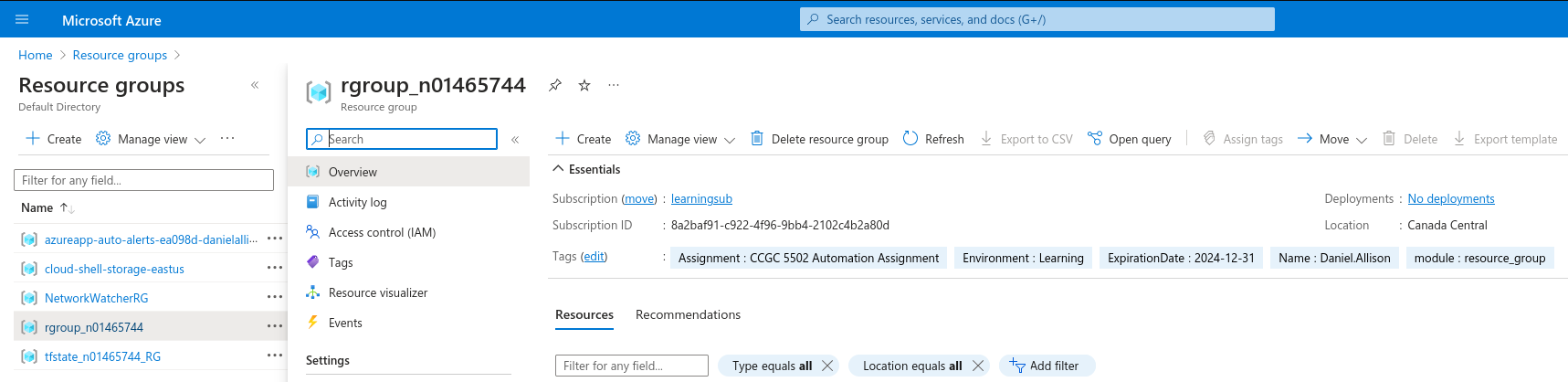


The resource group was verified as deployed in Azure Portal.

SCREENSHOT 9: Azure Portal shows the assignment1.tfstate after resource group deployed



SCREENSHOT 10: Azure Portal shows deployed resource group with tags



### Release

This module was released in version 0.1.0.

## Network Child Module

### Definition

The virtual networking, subnet, network security group, network security rule, and subnet network security group association resources are defined in main.tf with parametrized fields, and tagged with project metadata.[[2]](#footnote-4)

SCREENSHOT 11: the network child module's main.tf, with a vnet, subnet, nsg, and net sec rules



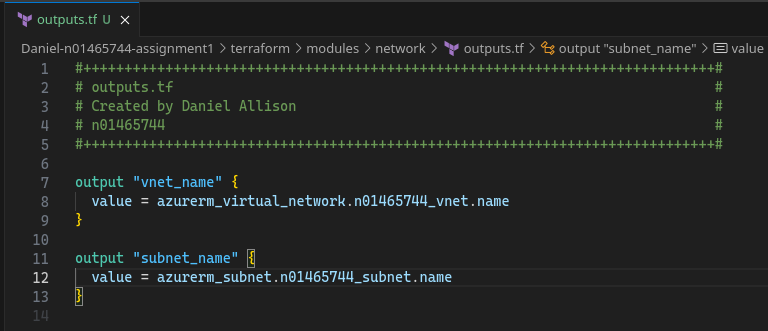
The corresponding variables are declared in variables.tf.

SCREENSHOT 12: network child module’s variables.tf with variables for vnet, subnet, nsg, local module and project metadata



outputs.tf returns the names of the virtual network and subnet to the root module.

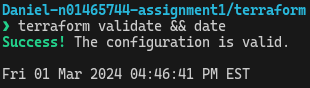
SCREENSHOT 13: network child module's outputs.tf with vnet and subnet names



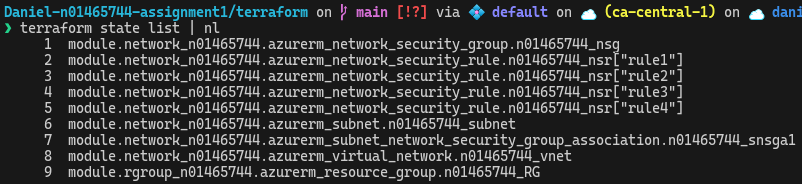
### *Dev Test: Validation, Provisioning, Verifying in Azure Portal*

This module was tested with terraform validate, plan, and deploy.

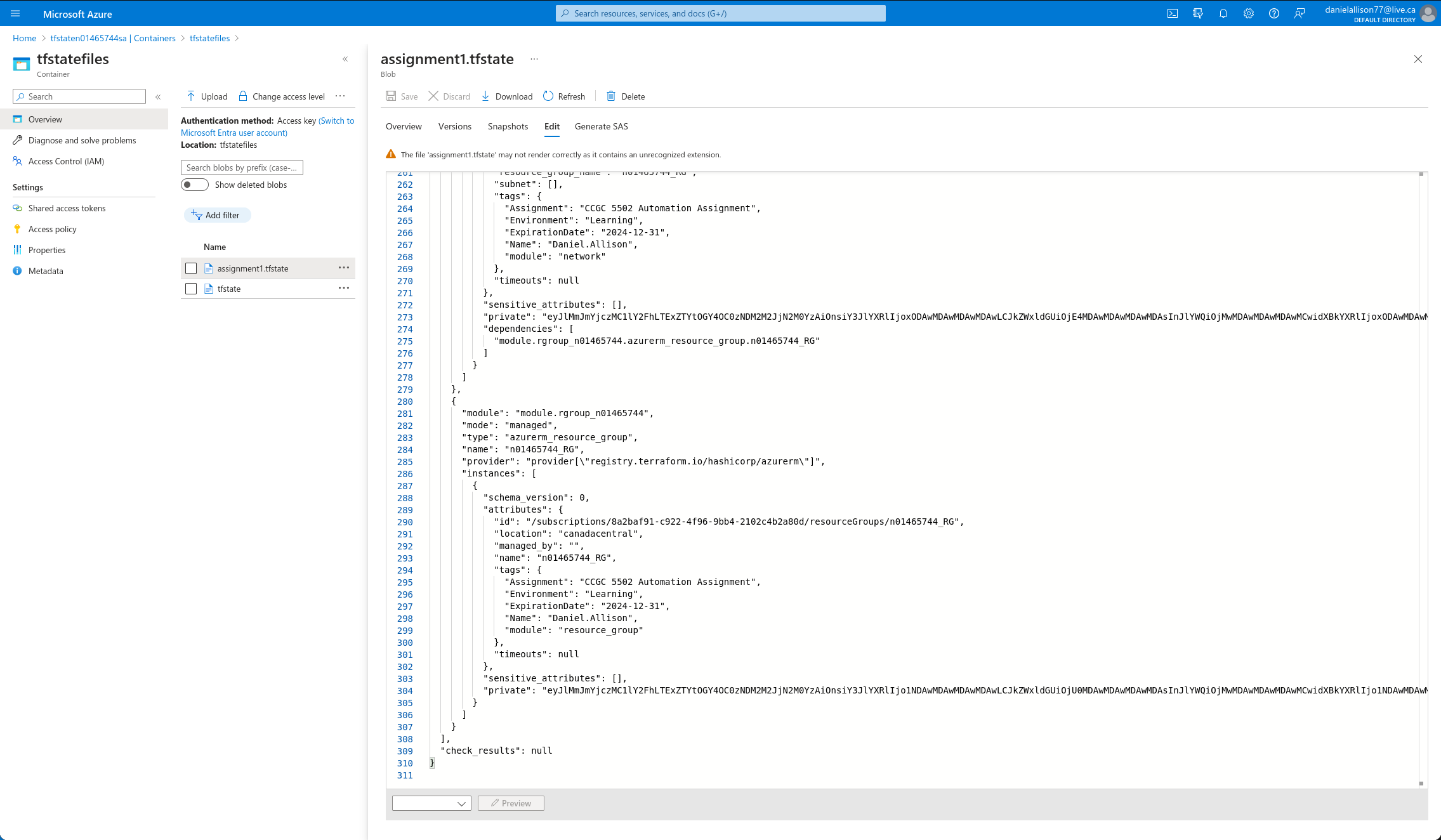
SCREENSHOT 14: terraform validate shows valid configuration for newly added network module



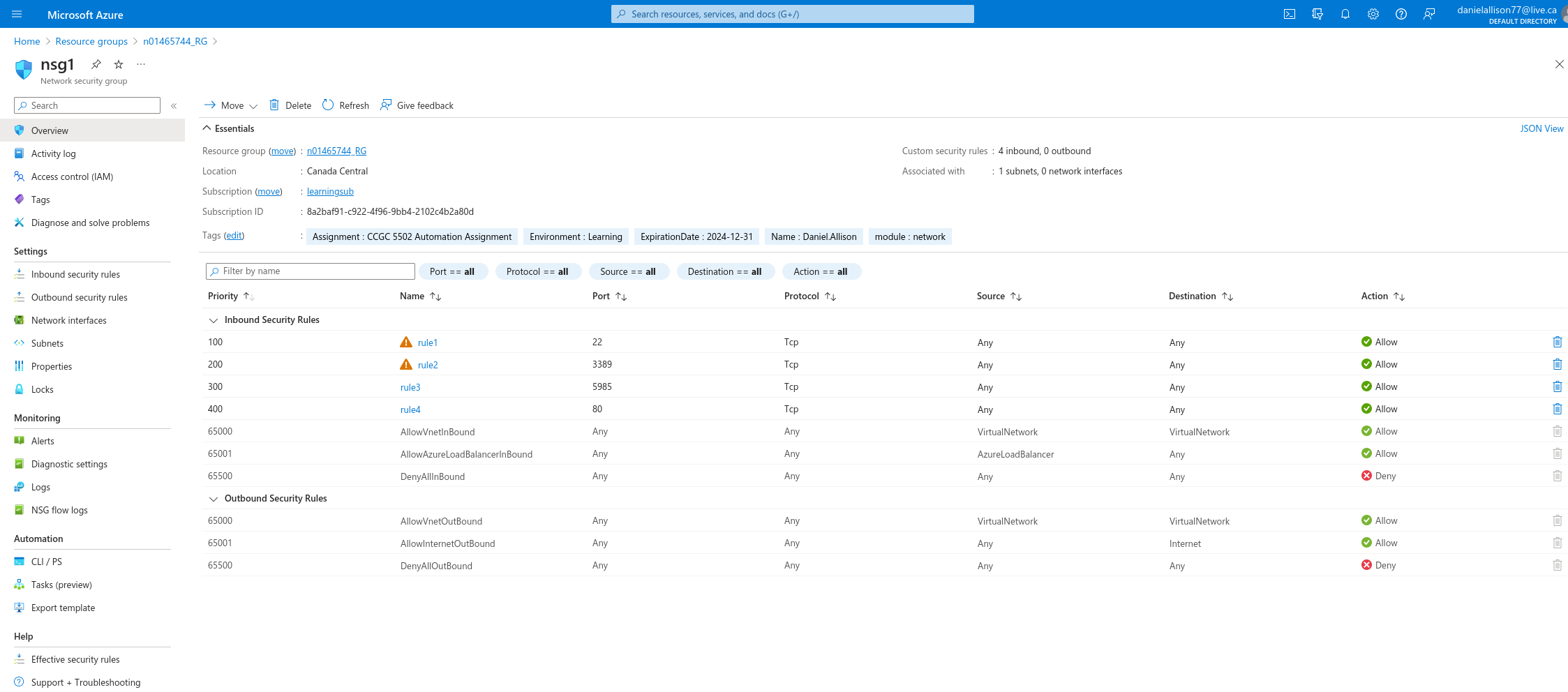
SCREENSHOT 15: terraform state list shows 9 resources deployed after network module added



SCREENSHOT 16: Azure Portal shows assignment1.tfstate after network module deployed



SCREENSHOT 17: Azure Portal shows deployed network security group with open ports 22, 3389, 5985, and 80



### Release

This module was released in version 0.2.0.

1. https://developer.hashicorp.com/terraform/language/values/locals [↑](#footnote-ref-3)
2. Inspiration for for\_each iterating over objects taken from this StackOverflow answer: https://stackoverflow.com/a/67871267 [↑](#footnote-ref-4)