Daniel Allison

danielallison77@gmail.com

n01465744

Assignment 1

CCGC 5502 Automation – Humber College

Table of Contents

[**Phase 1: Development** 4](#_Toc22)

[Setup 4](#_Toc23)

[GitHub Repo 4](#_Toc24)

[Cocogitto 5](#_Toc25)

[backend.tf and providers.tf 6](#_Toc26)

[terraform init 7](#_Toc27)

[Project Metadata (root variables.tf) 8](#_Toc28)

[Resource Group Child Module 9](#_Toc29)

[Definition 9](#_Toc30)

[Dev Test: Validation, Provisioning, Verifying in Azure Portal 10](#_Toc31)

[Release 11](#_Toc32)

[Network Child Module 12](#_Toc33)

[Definition 12](#_Toc34)

[Dev Test: Validation, Provisioning, Verifying in Azure Portal 15](#_Toc35)

[Release 16](#_Toc36)

[Common Child Module 17](#_Toc37)

[Definition 17](#_Toc38)

[Dev Test: Validation, Provisioning, Verifying in Azure Portal 20](#_Toc39)

[Release 21](#_Toc40)

[Linux Child Module 21](#_Toc41)

[Research 21](#_Toc42)

[Definition 22](#_Toc43)

[Dev Test: Validation, Provisioning, Verifying in Azure Portal 28](#_Toc44)

[Release 29](#_Toc45)

[Windows Child Module 30](#_Toc46)

[Research 30](#_Toc47)

[Securing Credentials 30](#_Toc48)

[Definition 32](#_Toc49)

[Dev Test: Validation, Provisioning, Verifying in Azure Portal 37](#_Toc50)

Table of Figures

[SCREENSHOT 1: The mostly empty initialized code repository in GitHub 4](#_Toc22)

[SCREENSHOT 2: cog.toml, the configuration file for cocogitto, a conventional commits tool 5](#_Toc23)

[SCREENSHOT 3: backend.tf and providers.tf 6](#_Toc24)

[SCREENSHOT 4: terraform init and then again with migrate state to the new backend tfstate key 7](#_Toc25)

[SCREENSHOT 5: variables.tf in the root module showing the project\_metadata variable, used for tagging every resource 8](#_Toc26)

[SCREENSHOT 6: Resource group child module’s main.tf, variables.tf, and outputs.tf 9](#_Toc27)

[SCREENSHOT 7: terraform validate shows valid configuration for newly added resource group 10](#_Toc28)

[SCREENSHOT 8: terraform state list | nl shows the provisioned resource group 10](#_Toc29)

[SCREENSHOT 9: Azure Portal shows the assignment1.tfstate after resource group provisioned 10](#_Toc30)

[SCREENSHOT 10: Azure Portal shows provisioned resource group with tags 11](#_Toc31)

[SCREENSHOT 11: Network child module's main.tf, with a vnet, subnet, nsg, and net sec rules 12](#_Toc32)

[SCREENSHOT 12: Network child module’s variables.tf with variables for vnet, subnet, nsg, local module and project metadata 13](#_Toc33)

[SCREENSHOT 13: Network child module's outputs.tf with vnet and subnet names 14](#_Toc34)

[SCREENSHOT 14: terraform validate shows valid configuration for newly added network module 15](#_Toc35)

[SCREENSHOT 15: terraform state list shows 9 resources provisioned after network module added 15](#_Toc36)

[SCREENSHOT 16: Azure Portal shows assignment1.tfstate after network module provisioned 15](#_Toc37)

[SCREENSHOT 17: Azure Portal shows provisioned network security group with open ports 22, 3389, 5985, and 80 16](#_Toc38)

[SCREENSHOT 18: Common child module's main.tf including log analytics workspace, recovery services vault, and storage account 17](#_Toc39)

[SCREENSHOT 19: Local module variable, variables for resource group, log analytics workspace, recovery services vault, storage account, and project metadata 18](#_Toc40)

[SCREENSHOT 20: Common child module’s outputs.tf returns log analytics workspace name, recovery services vault name, and storage account name 19](#_Toc41)

[SCREENSHOT 21: terraform validate shows valid configuration for newly added common module 20](#_Toc42)

[SCREENSHOT 22: terraform state list | nl shows 12 provisioned resources including newly added log analytics workspace, recovery services vault, and storage account 20](#_Toc43)

[SCREENSHOT 23: Azure Portal shows assignment1.tfstate showing the last lines of provisioned resources 20](#_Toc44)

[SCREENSHOT 24: Azure Cloud Shell shows a PowerShell command that identifies CentOS 8.2 offerings from OpenLogic 21](#_Toc45)

[SCREENSHOT 25: Azure Cloud Shell shows a PowerShell command that identifies the version number for CentOs 8\_2-gen2 21](#_Toc46)

[SCREENSHOT 26: Azure Cloud Shell shows PowerShell commands being run to find out the virtual machine extensions' types, versions, and publisher 21](#_Toc47)

[SCREENSHOT 27: Linux child module's main.tf, lines 1-64 22](#_Toc48)

[SCREENSHOT 28: Linux child module's main.tf, lines 65-118 23](#_Toc49)

[SCREENSHOT 29: Linux child module's main.tf, lines 118-134 24](#_Toc50)

[SCREENSHOT 30: Linux child module's variables.tf, lines 1-66 25](#_Toc51)

[SCREENSHOT 31: Linux child module's variables.tf lines 67-108 26](#_Toc52)

[SCREENSHOT 32: Linux child module's outputs.tf 27](#_Toc53)

[SCREENSHOT 33: Linux child module's provisioner.tf 27](#_Toc54)

[SCREENSHOT 34: terraform validate shows configuration is valid after Linux child module added 28](#_Toc55)

[SCREENSHOT 35: terraform state list | nl shows 25 provisioned resources after Linux module added 28](#_Toc56)

[SCREENSHOT 36: Azure Portal shows provisioned resources after Linux module added 29](#_Toc57)

[SCREENSHOT 37: Azure Cloud Shell shows a PowerShell commands used to find the details of the antimalware vm extension 30](#_Toc58)

[SCREENSHOT 38: Az cli is used to create a new resource group for a new key vault 30](#_Toc59)

[SCREENSHOT 39: Az cli is used to create a new key vault 31](#_Toc60)

[SCREENSHOT 40: Az cli is used to store a secret, the Windows admin password 31](#_Toc61)

[SCREENSHOT 41: Windows child module's main.tf, lines 1-62 32](#_Toc62)

[SCREENSHOT 42: Windows child module's main.tf, lines 62-112 33](#_Toc63)

[SCREENSHOT 43: Windows child module's variables.tf, lines 1-60 34](#_Toc64)

[SCREENSHOT 44: Windows child module's variables.tf, lines 60-84 35](#_Toc65)

[SCREENSHOT 45: Windows child module's outputs.tf 36](#_Toc66)

[SCREENSHOT 46: terraform validate shows configuration is valid after Windows module added 37](#_Toc67)

[SCREENSHOT 47: terraform state list shows provisioned resources 37](#_Toc68)

[SCREENSHOT 48: Azure Portal shows all provisioned resources, including the newly added ones from the Windows child module 38](#_Toc69)

[SCREENSHOT 49: Azure Portal shows provisioned Windows VM with antimalware extension 39](#_Toc70)

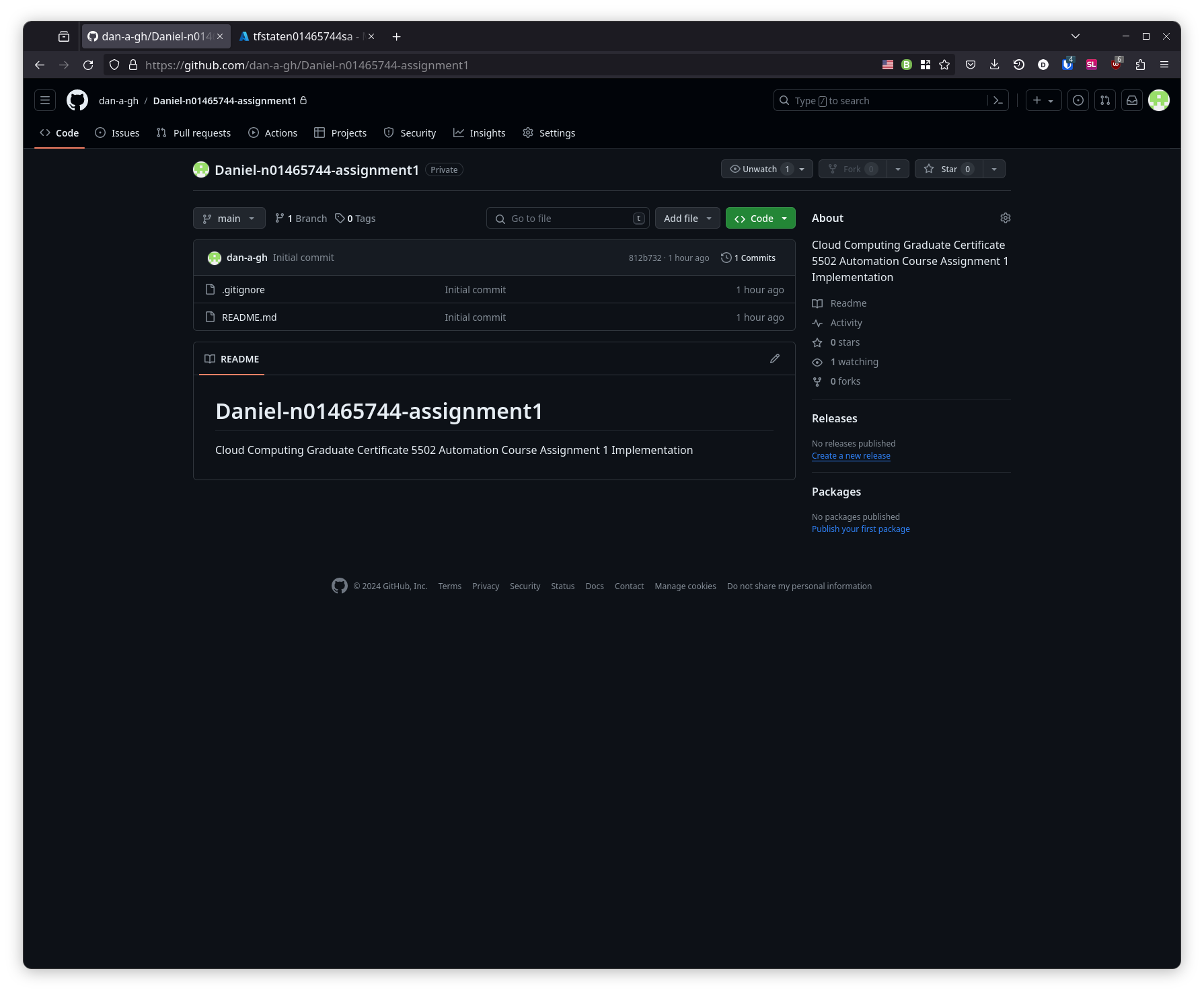
# **Phase 1: Development**

## 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.

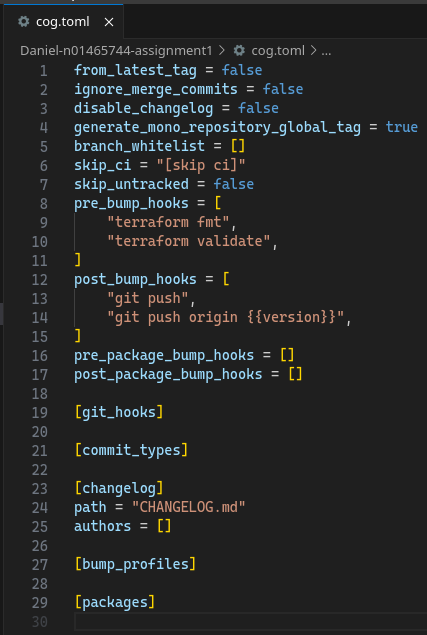
SCREENSHOT 1: The mostly empty initialized code repository in GitHub



### 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.

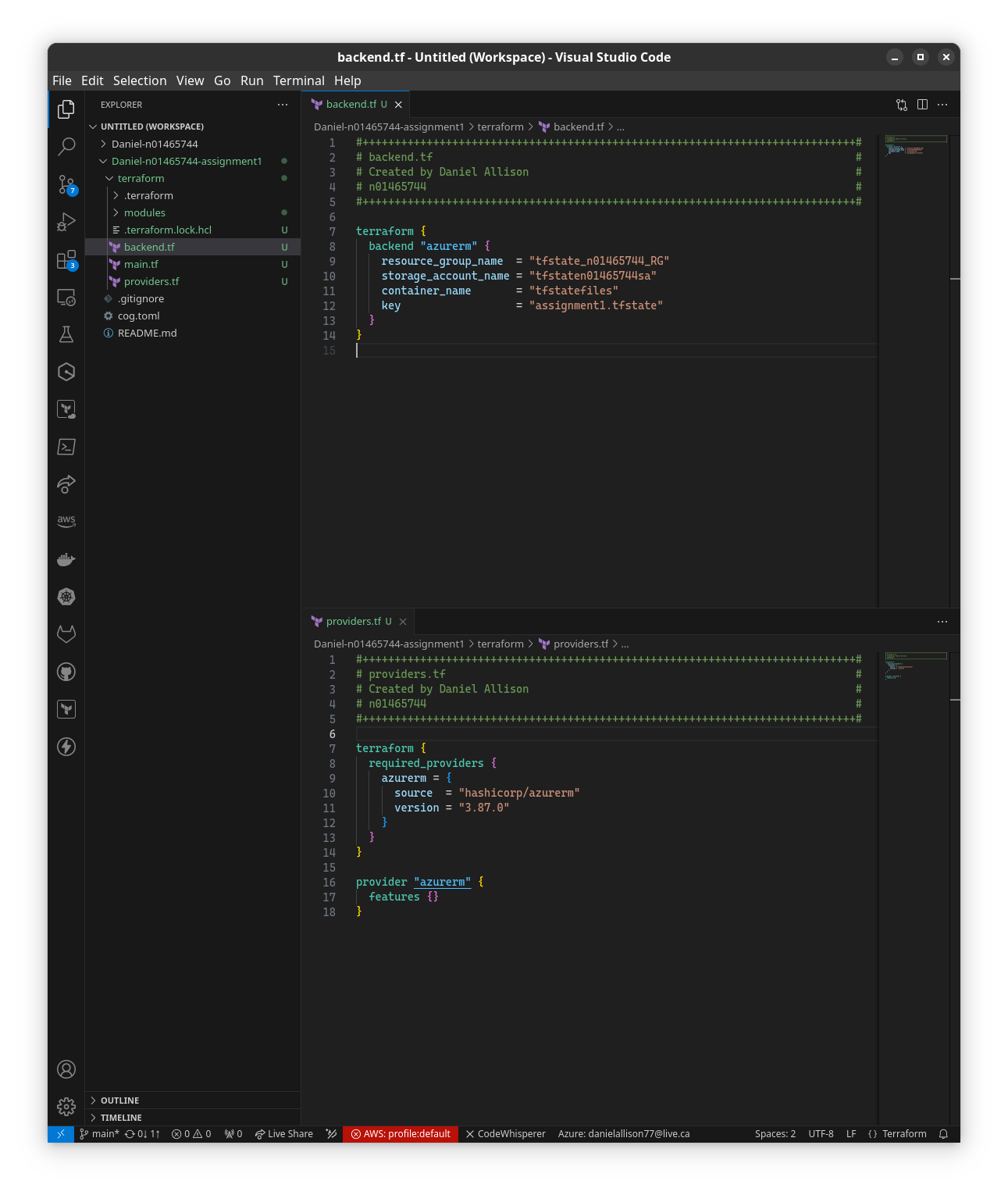
SCREENSHOT 2: cog.toml, the configuration file for cocogitto, a conventional commits tool



### 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.

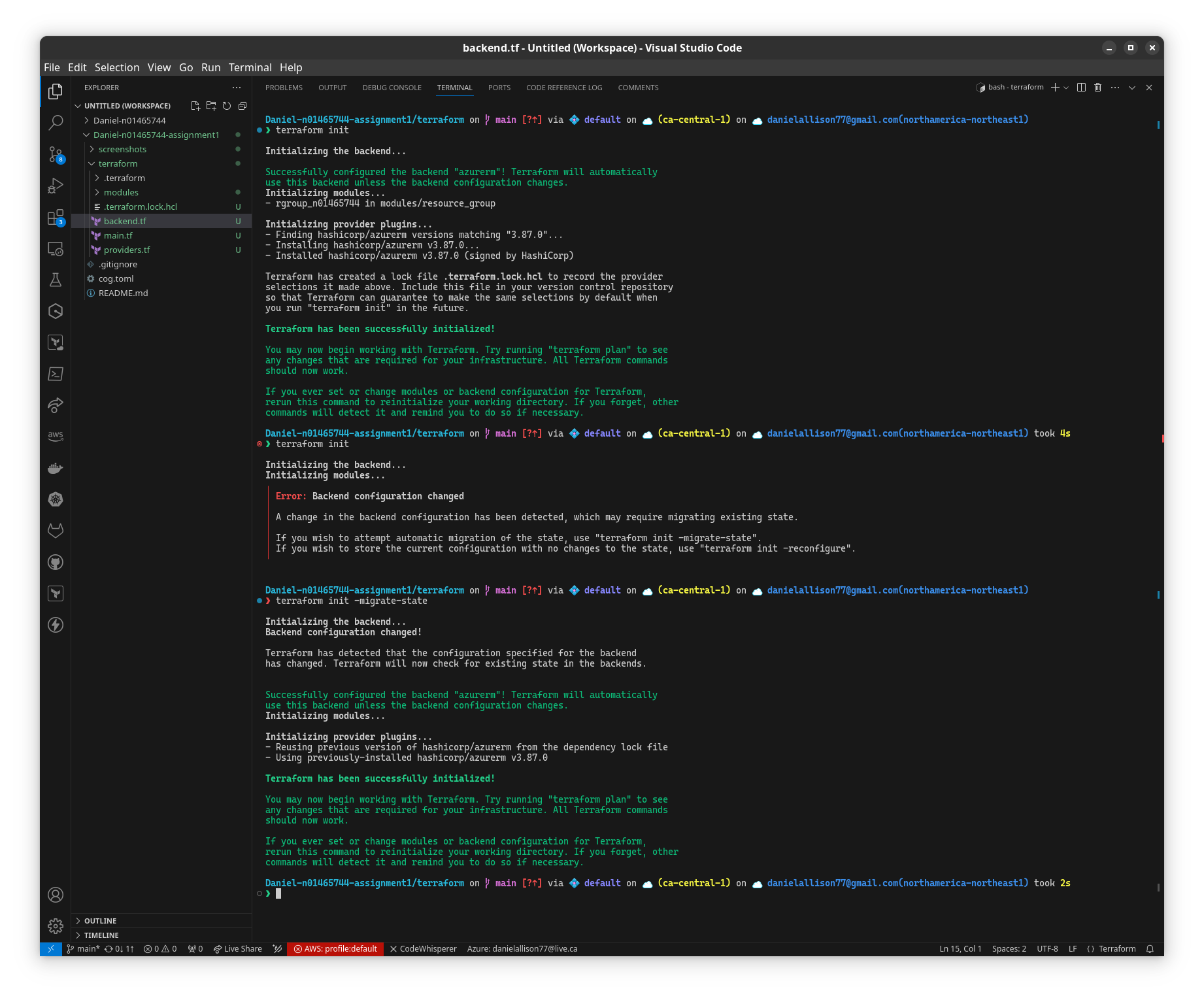
SCREENSHOT 3: backend.tf and providers.tf



### 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.

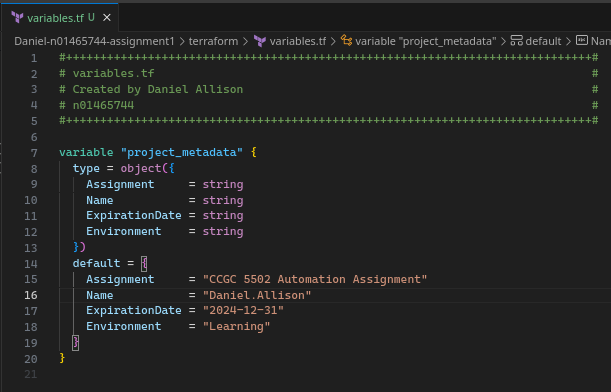
SCREENSHOT 4: terraform init and then again with migrate state to the new backend tfstate key



### 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

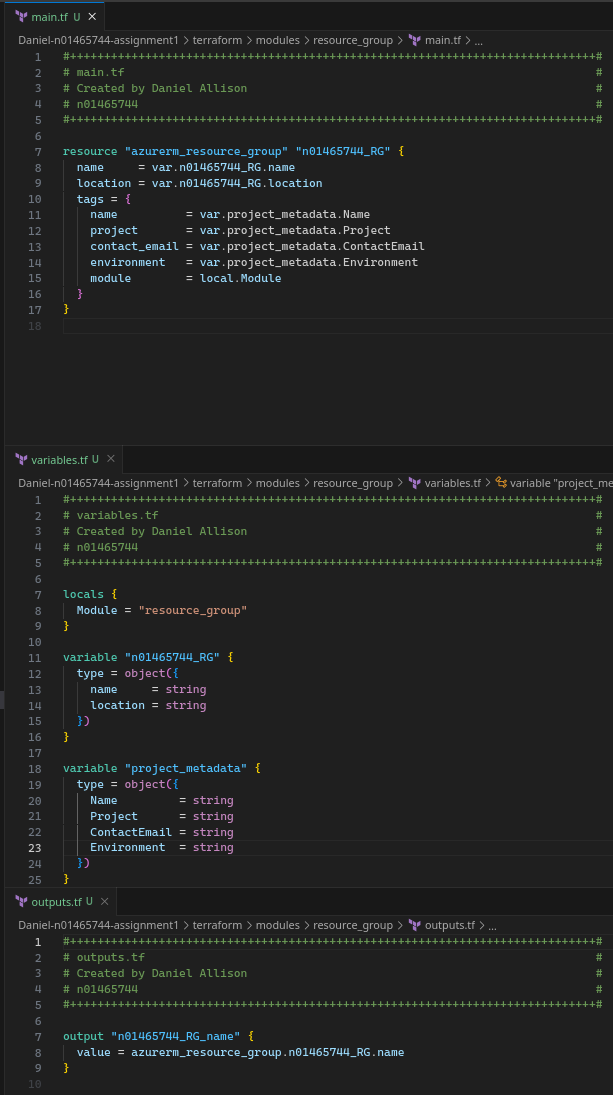


## Resource Group Child Module

### 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.

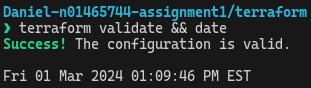
SCREENSHOT 6: Resource group child module’s main.tf, variables.tf, and outputs.tf



### *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

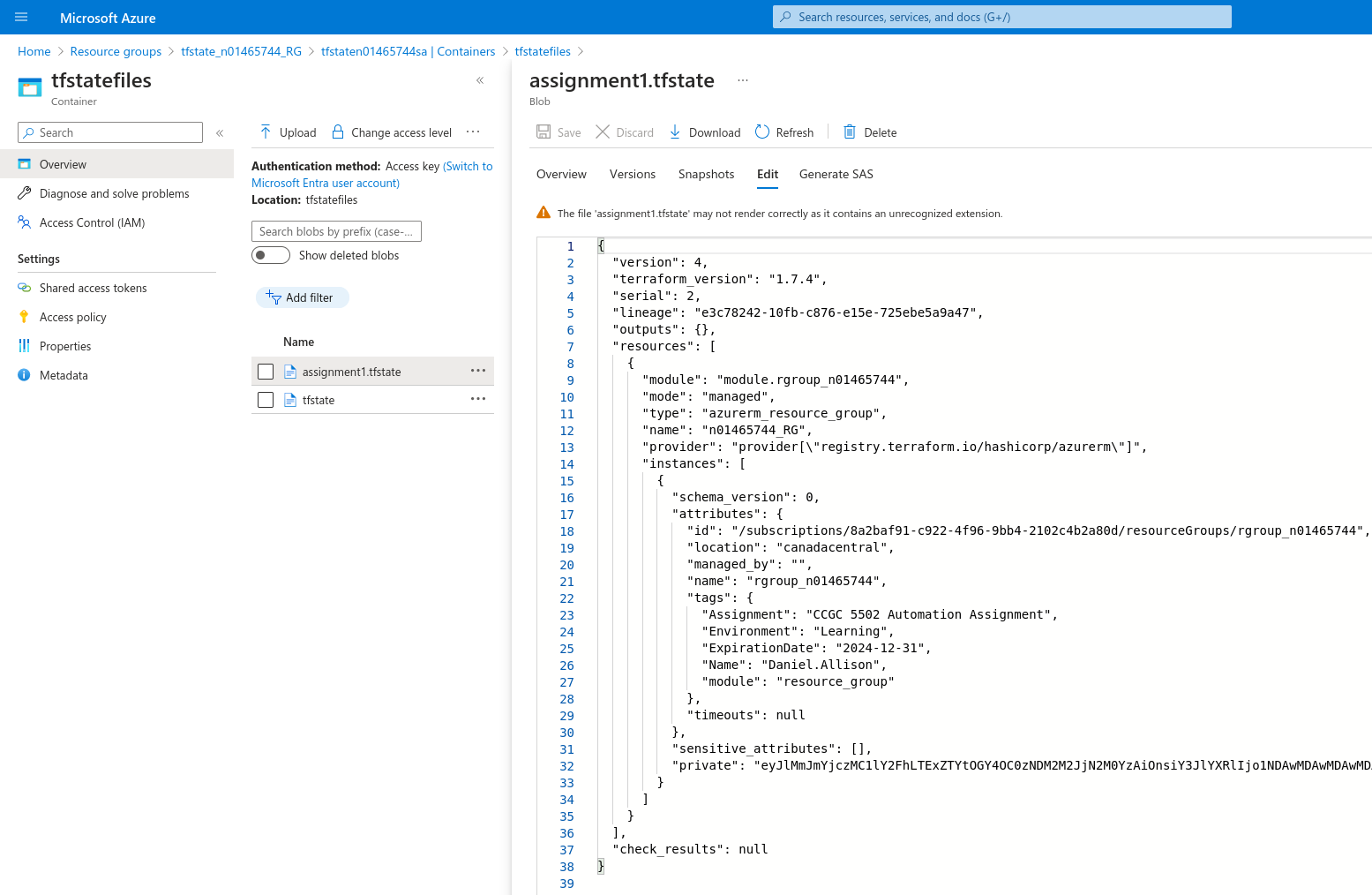


SCREENSHOT 8: terraform state list | nl shows the provisioned resource group

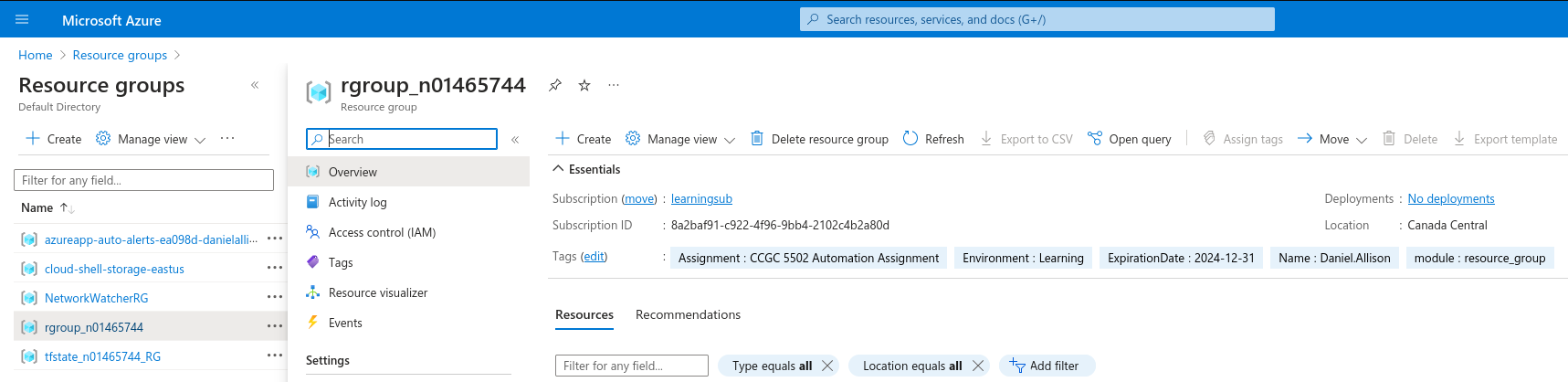


The resource group was verified as provisioned in Azure Portal.

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



SCREENSHOT 10: Azure Portal shows provisioned resource group with tags



### Release

The resource group child module was released in version 0.1.0.

## Network Child Module

### Definition

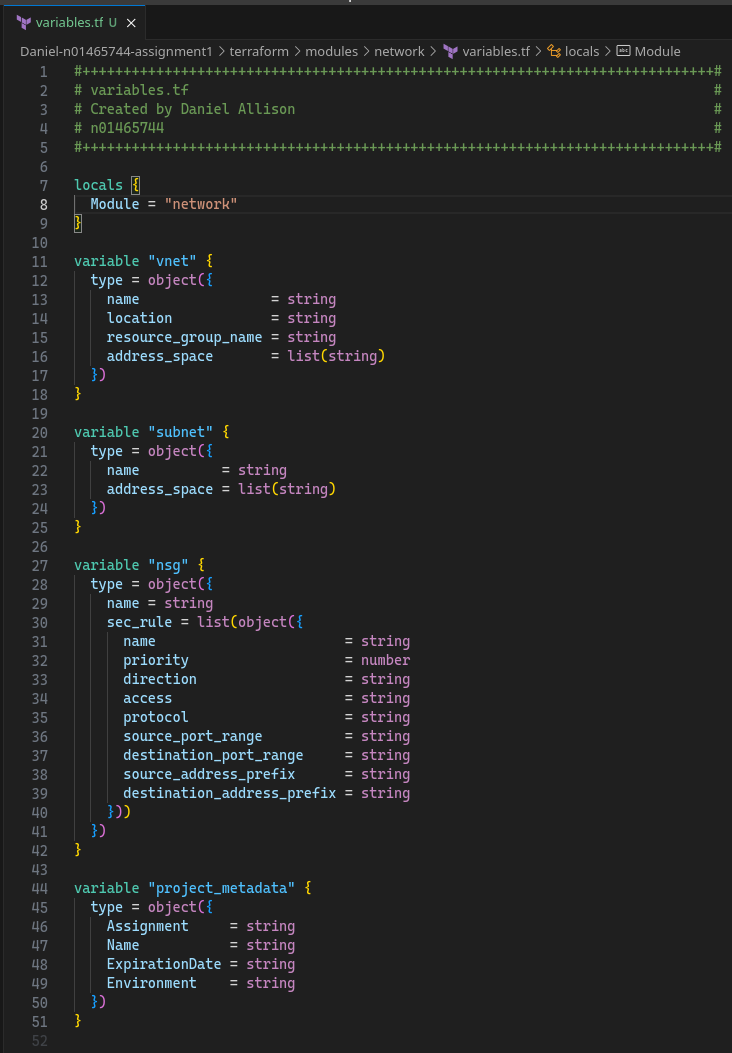
The virtual networking, subnet, network security group, network security rule, and subnet network security group association resources are defined in main.tf with parametrized fields, and tagged with project metadata.[[2]](#footnote-4)

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



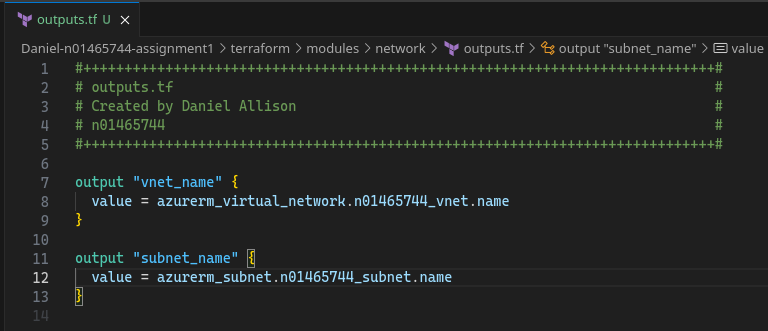
The corresponding variables are declared in variables.tf.

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.

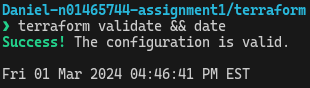
SCREENSHOT 13: Network child module's outputs.tf with vnet and subnet names



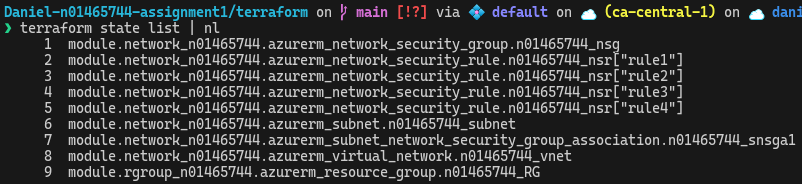
### *Dev Test: Validation, Provisioning, Verifying in Azure Portal*

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

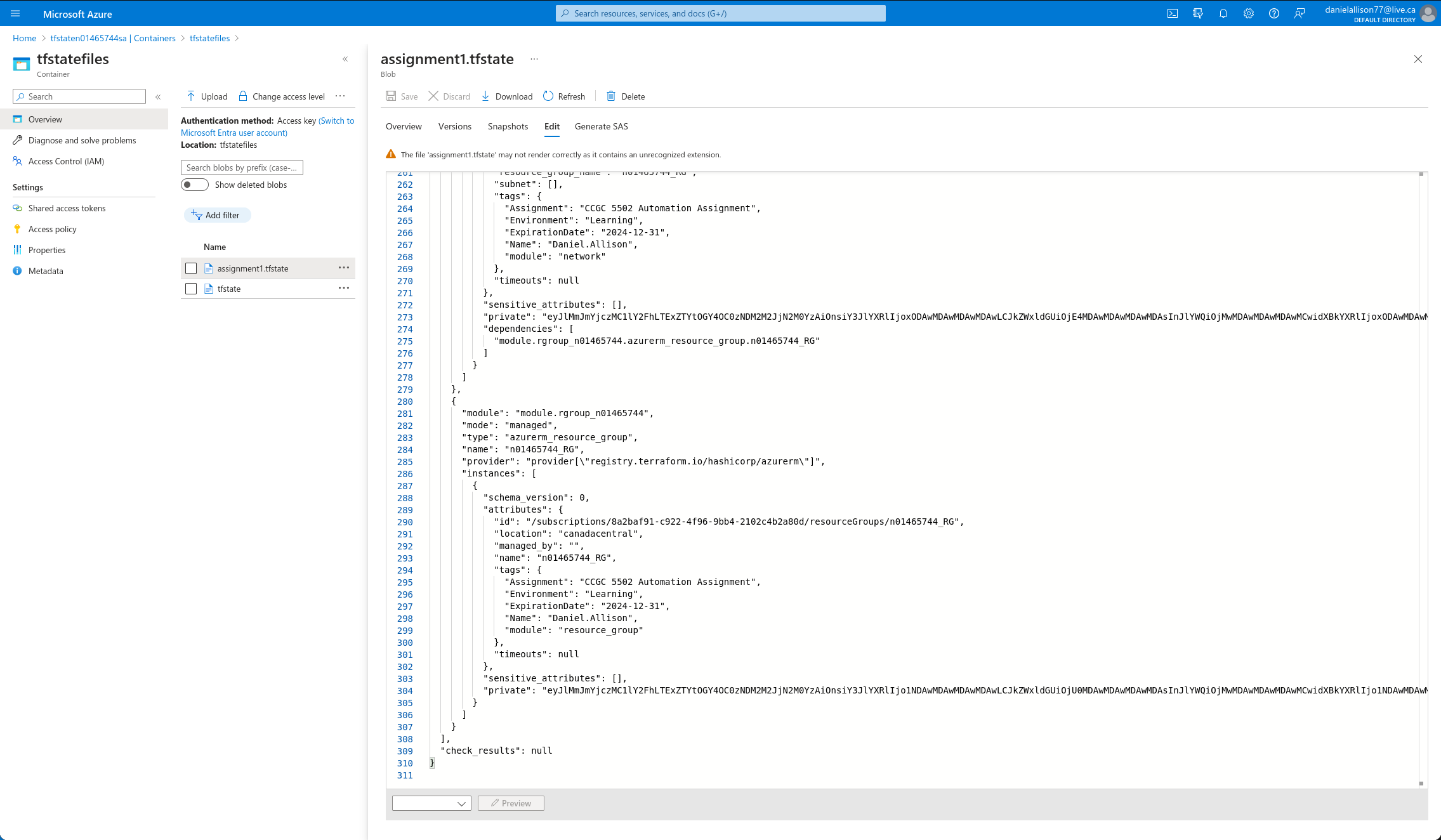
SCREENSHOT 14: terraform validate shows valid configuration for newly added network module



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



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



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



### Release

The network child module was released in version 0.2.0.

## Common Child Module

### Definition

The log analytics workspace, recovery services vault, and storage account are defined in main.tf with parametrized fields, and tagged with project metadata.

SCREENSHOT 18: Common child module's main.tf including log analytics workspace, recovery services vault, and storage account

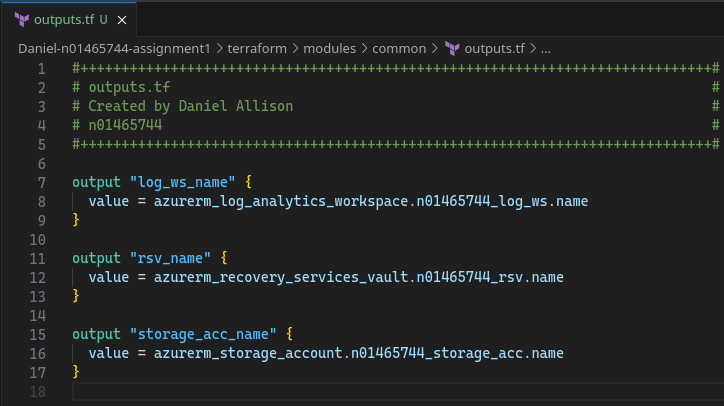


The corresponding variables are declared in variables.tf.

SCREENSHOT 19: Local module variable, variables for resource group, log analytics workspace, recovery services vault, storage account, and project metadata



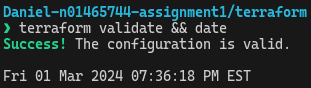
SCREENSHOT 20: Common child module’s outputs.tf returns log analytics workspace name, recovery services vault name, and storage account name



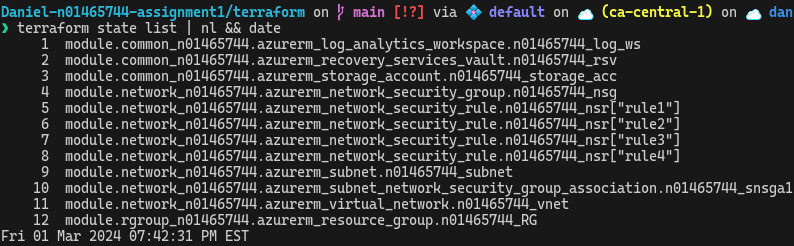
### Dev Test: *Validation, Provisioning, Verifying in Azure Portal*

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

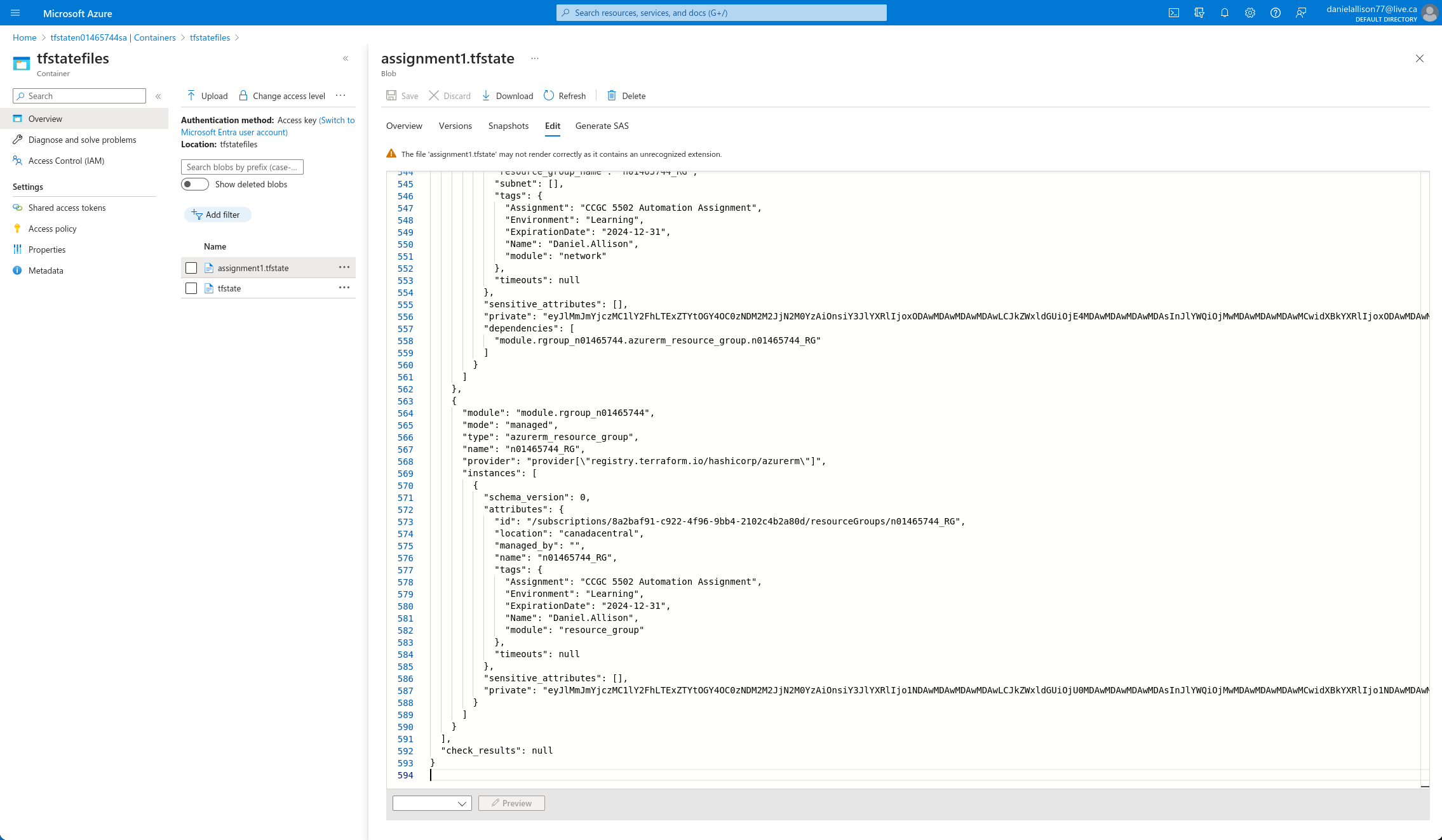
SCREENSHOT 21: terraform validate shows valid configuration for newly added common module



SCREENSHOT 22: terraform state list | nl shows 12 provisioned resources including newly added log analytics workspace, recovery services vault, and storage account



SCREENSHOT 23: Azure Portal shows assignment1.tfstate showing the last lines of provisioned resources



### Release

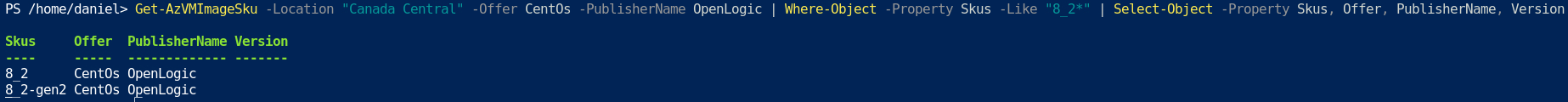
The common child module was released in version 0.3.0.

## Linux Child Module

### Research

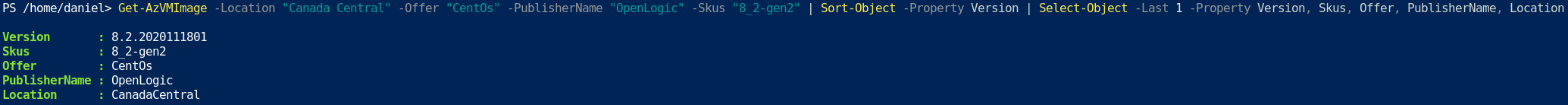
In order to deploy a CentOS virtual machine, a suitable offering needed to be found from the Azure Marketplace. Microsoft maintains a list of endorsed Linux distributions[[3]](#footnote-5), which is a good place to start to look for CentOS offering. OpenLogic by Perforce, formerly RogueWave, maintains several images. Using the following PowerShell command in Azure Cloud Shell, two candidates were identified:

SCREENSHOT 24: Azure Cloud Shell shows a PowerShell command that identifies CentOS 8.2 offerings from OpenLogic



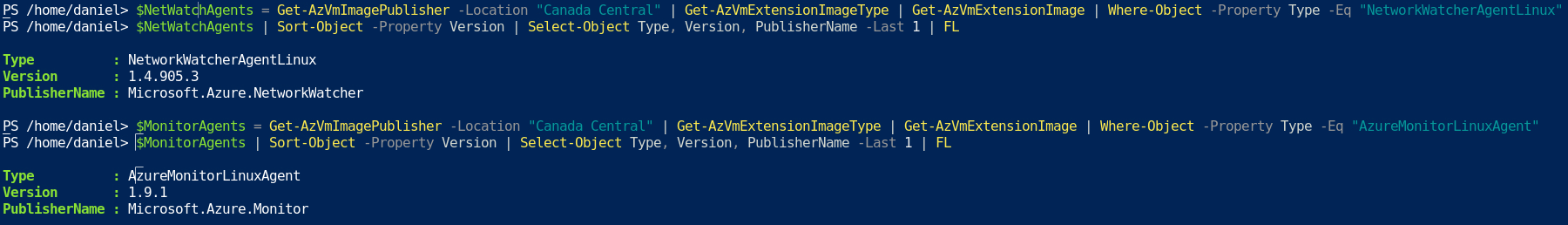
8\_2-gen2 was picked. This next command identified the version number of the latest version:

SCREENSHOT 25: Azure Cloud Shell shows a PowerShell command that identifies the version number for CentOs 8\_2-gen2



The virtual machine extensions required some extra care to figure out an acceptable value for the type\_handler\_version. The following PowerShell commands were run[[4]](#footnote-6) in Azure Cloud Shell to find information about the latest versions of the Monitor and Network Watch extensions.

SCREENSHOT 26: Azure Cloud Shell shows PowerShell commands being run to find out the virtual machine extensions' types, versions, and publisher



Terraform won’t accept those version numbers unmodified though. They must be truncated to major.minor only.[[5]](#footnote-7) This was not clear in the terraform documentation.[[6]](#footnote-8)

### Definition

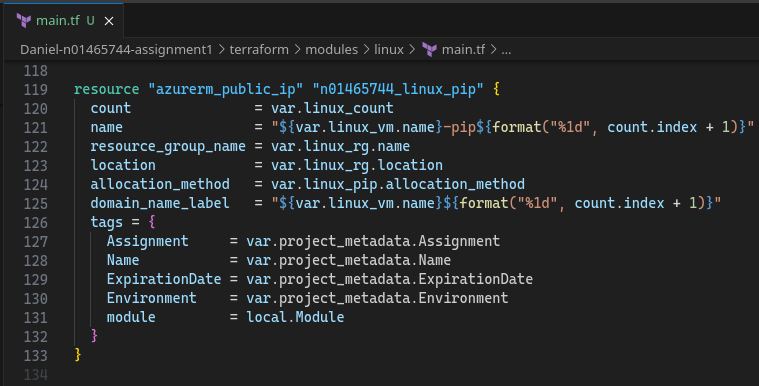
SCREENSHOT 27: Linux child module's main.tf, lines 1-64



SCREENSHOT 28: Linux child module's main.tf, lines 65-118



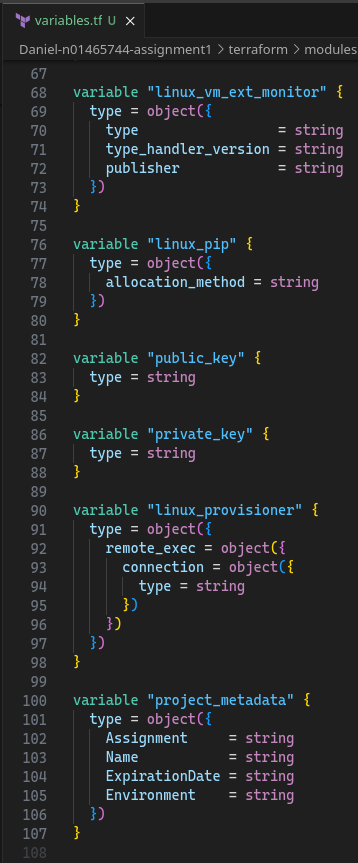
SCREENSHOT 29: Linux child module's main.tf, lines 118-134



SCREENSHOT 30: Linux child module's variables.tf, lines 1-66



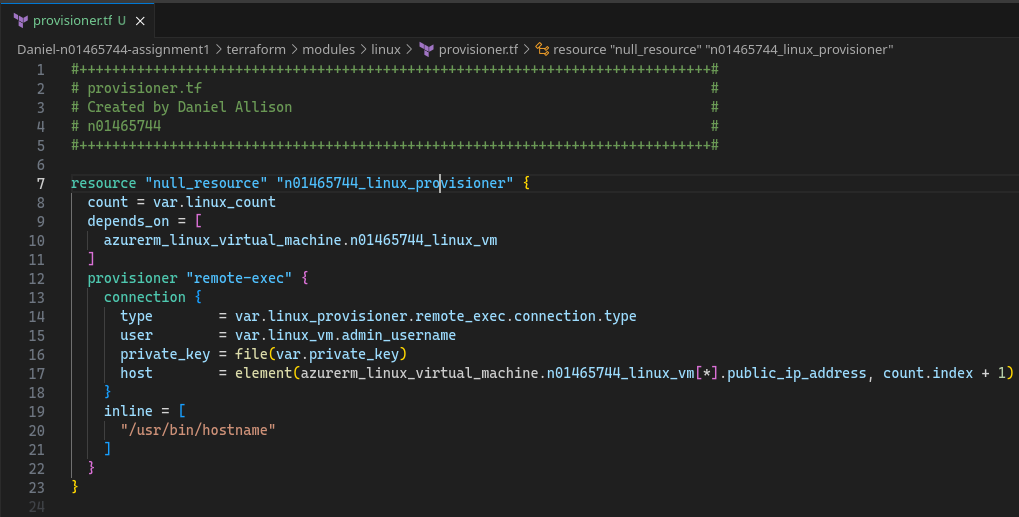
SCREENSHOT 31: Linux child module's variables.tf lines 67-108



SCREENSHOT 32: Linux child module's outputs.tf



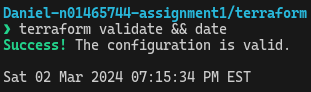
SCREENSHOT 33: Linux child module's provisioner.tf



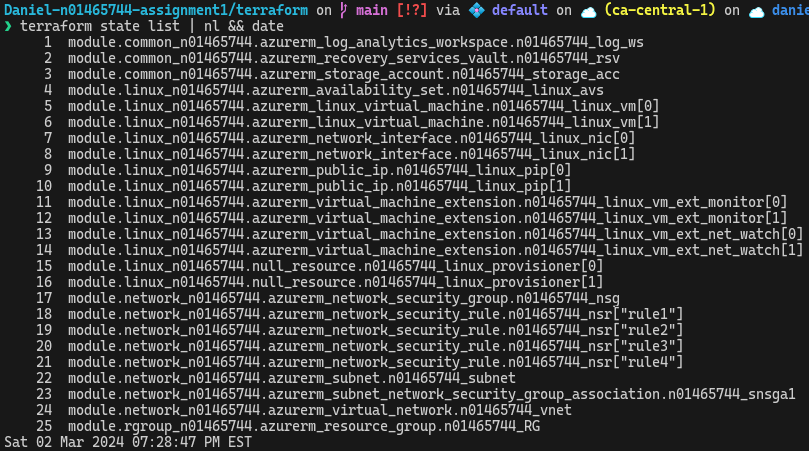
### Dev Test: *Validation, Provisioning, Verifying in Azure Portal*

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

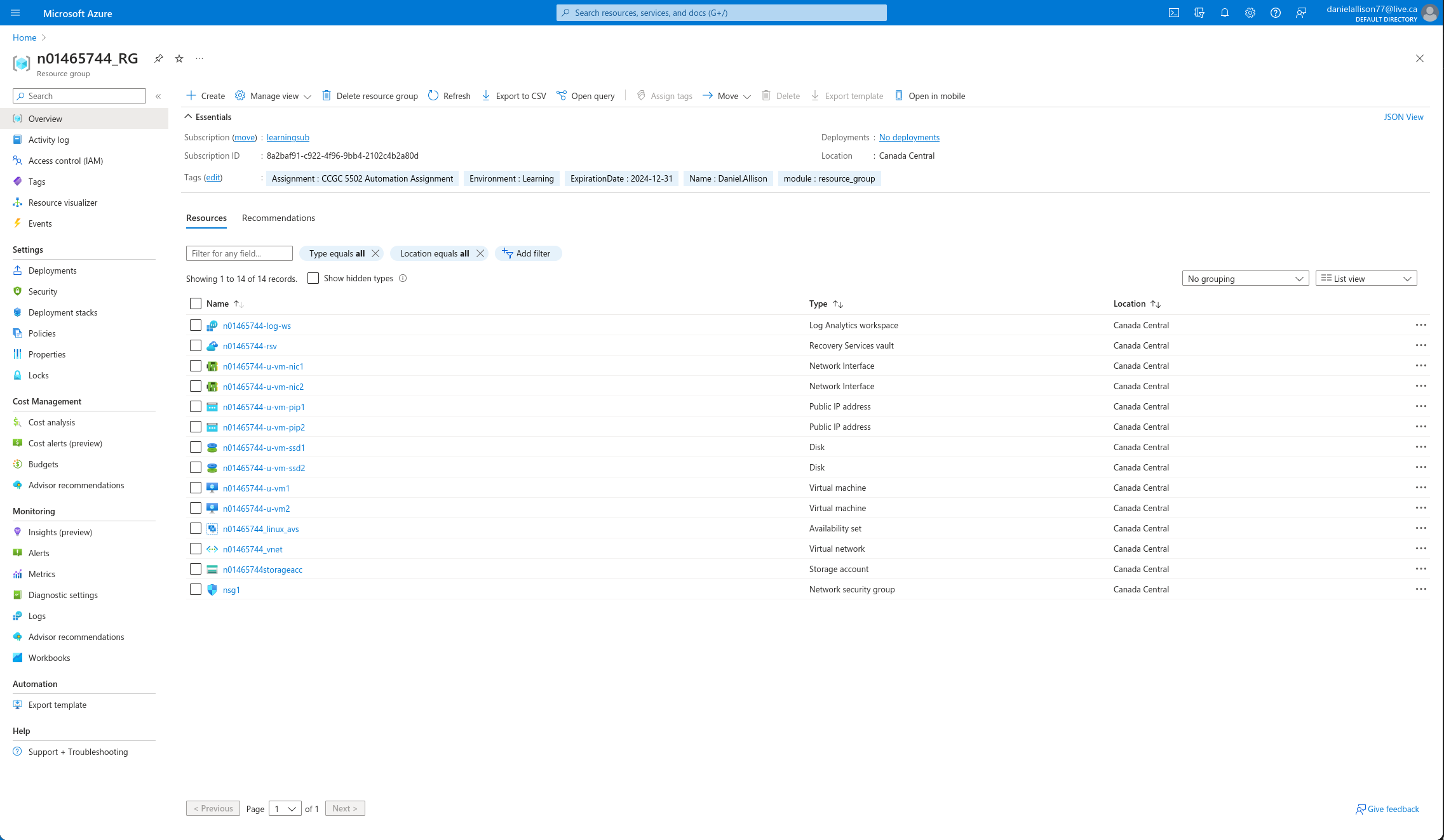
SCREENSHOT 34: terraform validate shows configuration is valid after Linux child module added



SCREENSHOT 35: terraform state list | nl shows 25 provisioned resources after Linux module added



SCREENSHOT 36: Azure Portal shows provisioned resources after Linux module added



### Release

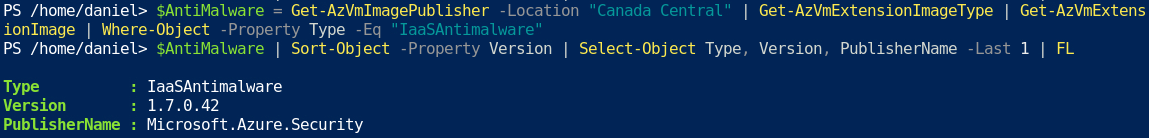
The Linux child module was released in version 0.4.0.

## Windows Child Module

### Research

The following PowerShell commands were run[[7]](#footnote-9) in Azure Cloud Shell to find information about the latest versions of the antimalware extension.

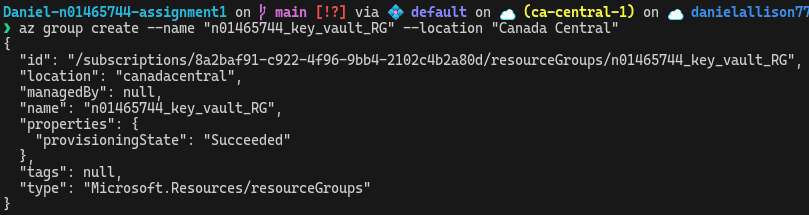
SCREENSHOT 37: Azure Cloud Shell shows a PowerShell commands used to find the details of the antimalware vm extension



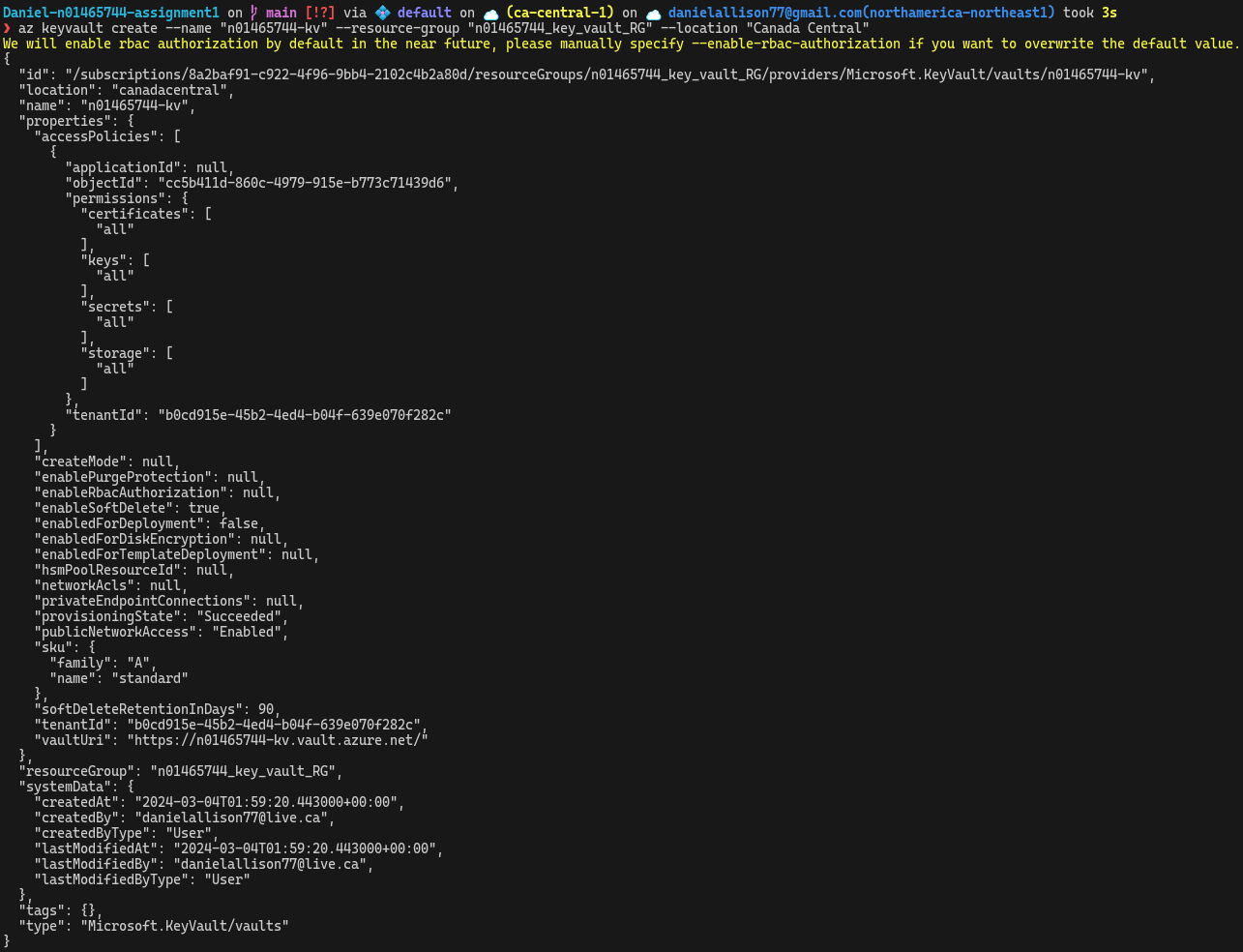
### Securing Credentials

User credentials should not be directly stored in Terraform code. To secure them, a new Azure Key Vault was created in it’s own separate resource group. The az cli tool was used for this purpose[[8]](#footnote-10):

SCREENSHOT 38: Az cli is used to create a new resource group for a new key vault



SCREENSHOT 39: Az cli is used to create a new key vault



SCREENSHOT 40: Az cli is used to store a secret, the Windows admin password

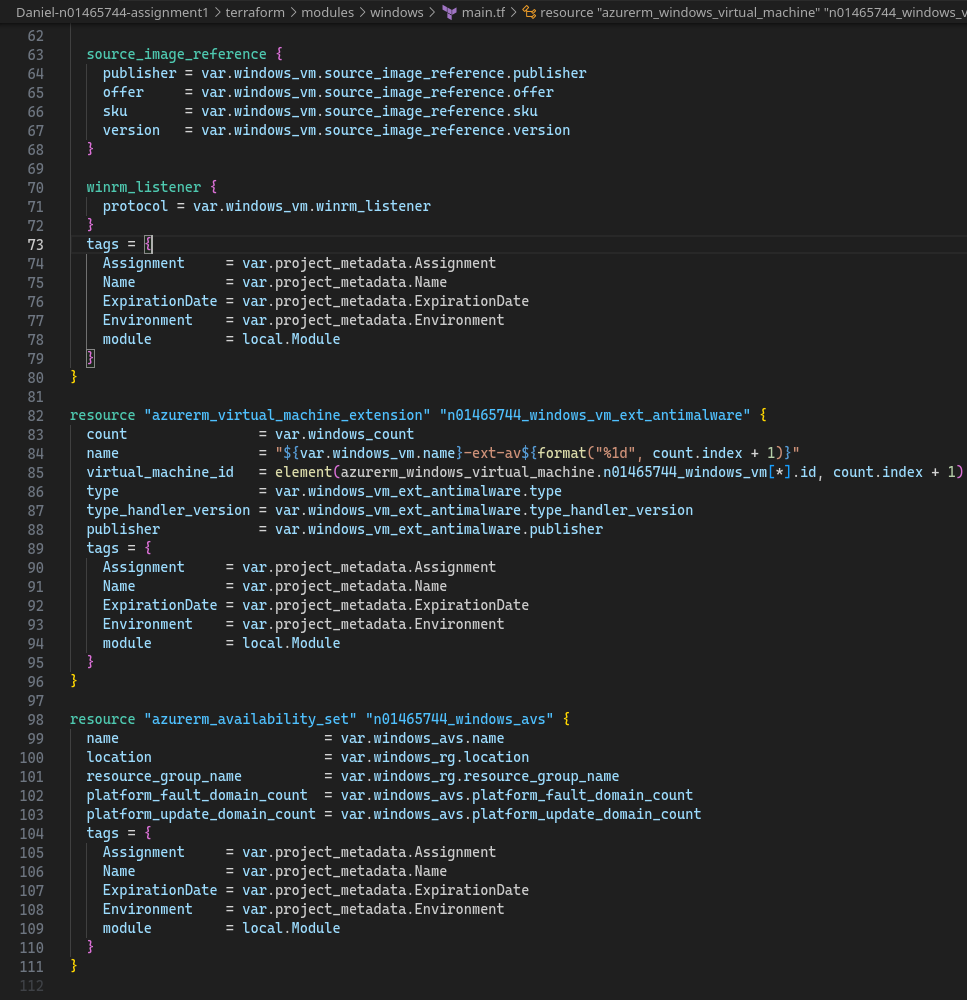


### *Definition*

SCREENSHOT 41: Windows child module's main.tf, lines 1-62



SCREENSHOT 42: Windows child module's main.tf, lines 62-112



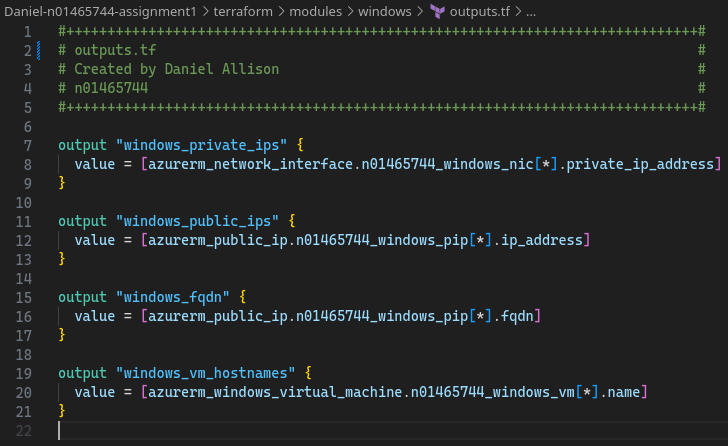
SCREENSHOT 43: Windows child module's variables.tf, lines 1-60



SCREENSHOT 44: Windows child module's variables.tf, lines 60-84



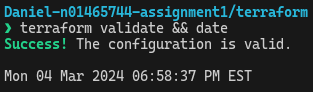
SCREENSHOT 45: Windows child module's outputs.tf



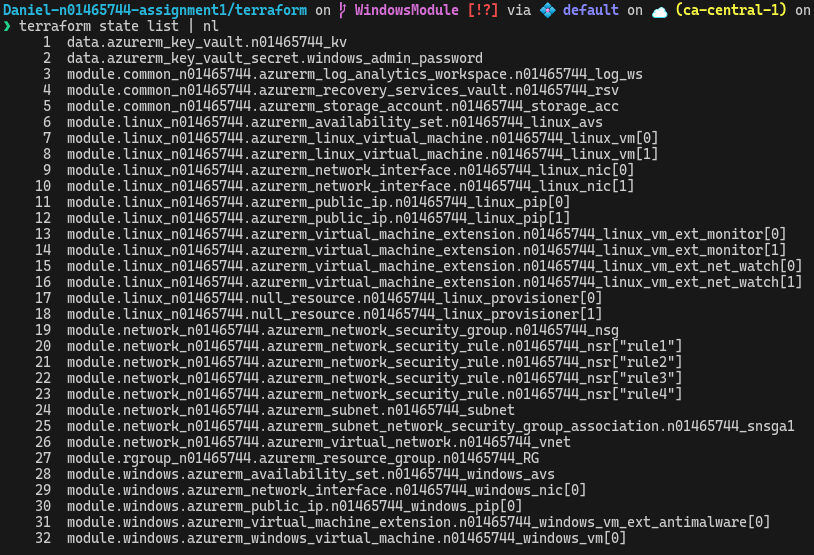
### *Dev Test: Validation, Provisioning, Verifying in Azure Portal*

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

SCREENSHOT 46: terraform validate shows configuration is valid after Windows module added



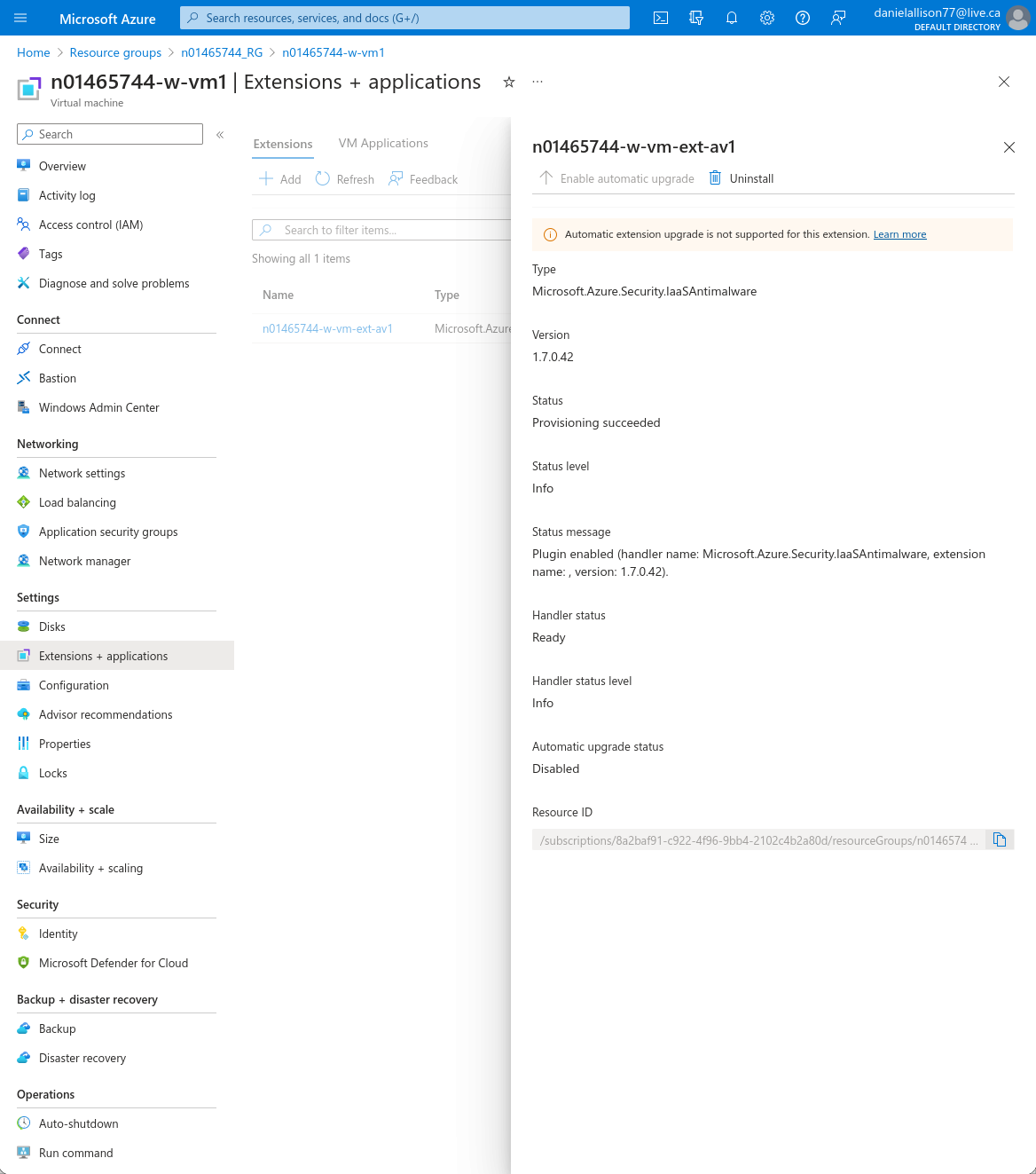
SCREENSHOT 47: terraform state list shows provisioned resources



SCREENSHOT 48: Azure Portal shows all provisioned resources, including the newly added ones from the Windows child module



SCREENSHOT 49: Azure Portal shows provisioned Windows VM with antimalware extension



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)
3. https://learn.microsoft.com/en-us/azure/virtual-machines/Linux/endorsed-distros [↑](#footnote-ref-5)
4. https://learn.microsoft.com/en-us/azure/virtual-machines/extensions/features-Windows [↑](#footnote-ref-6)
5. https://github.com/hashicorp/terraform-provider-azurestack/issues/125 [↑](#footnote-ref-7)
6. https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/virtual\_machine\_extension#type\_handler\_version [↑](#footnote-ref-8)
7. https://learn.microsoft.com/en-us/azure/virtual-machines/extensions/features-Windows [↑](#footnote-ref-9)
8. Commands sourced from https://learn.microsoft.com/en-us/azure/key-vault/secrets/quick-create-cli?source=recommendations [↑](#footnote-ref-10)