Final group work - GBA 6270

Angel Santoyo, Anna Betova, Thong Nguyen

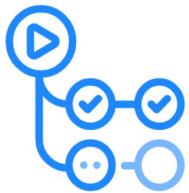
What is our project?

- **Project motivation:** Cloud resources created by hand are inconsistent and difficult to scale. We built an automated IaC pipeline so the same workflow can deploy **5, 10, or 20+ VMs** reliably with minimal configuration changes. *(For demo purposes + compute cost we are deploying one VM+Resource group)*

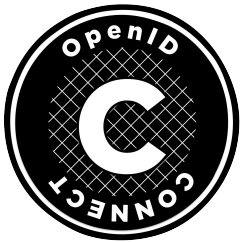
  - We automated Azure virtual machine and resource group deployment using **GitHub Actions** and **Infrastructure as Code**.
  - The VM infrastructure is defined in **Terraform** and deployed through GitHub Actions.
  - **Terraform remote state** ensures the pipeline can update, recreate, or reconcile resources over time.
  - **Ansible** installs required dependencies (Python, pip, numpy, etc.) through a separate configuration workflow

Technologies used

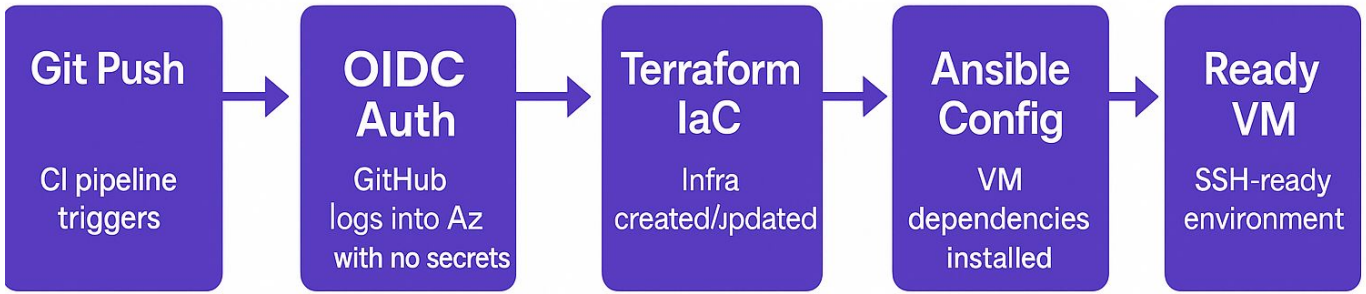
- **GitHub Actions** – CI/CD automation
- **Terraform** – Infrastructure as Code for Azure resources
- **Ansible** – Configuration management for installed packages
- **Azure** – Cloud provider hosting the VM
- **OIDC –** Secure identity federation (no secrets, aside from Azure IDs and an SSH key).



GitHub Actions



End-to-end VM deployment pipeline



How it Works:

- Infrastructure is defined as code (IaC) using Terraform.
- A single VM is created based on values in variables.tf.
- Terraform state stored in Azure Storage ensures consistent updates.
- Each workflow run validates, plans, and applies changes to **that same VM**.

Current Assumptions:

- One VM
- Some hard-coded configurations
- Terraform Managed lifecycle
- **Long-Term stateful management*

Directory Structure

```
root/
├── .github/
│   └── workflows/
│       └── deploy.yml
│       └── configure-ansible.yml
├── terraform/
│   ├── main.tf
│   ├── variables.tf
│   └── outputs.tf
│   └── providers.tf
│   └── ssh.tf
│   └── ssh_key.pub
├── ansible/
│   ├── configure-python.yml
│
├── README.md
```

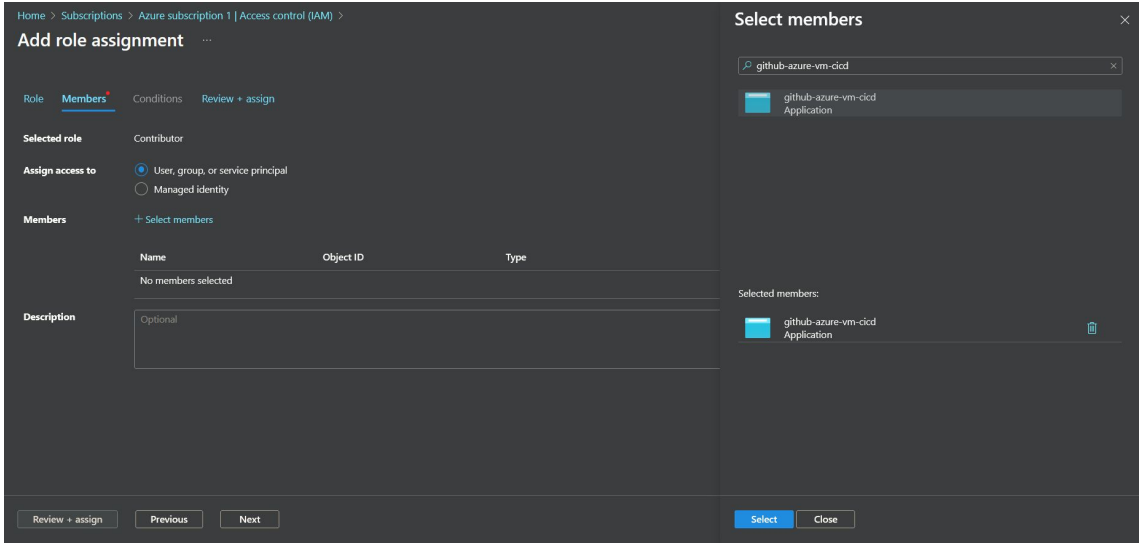
Setup Process

Set up an Entra app registration

- Go to https://portal.azure.com/#home
- Microsoft Entra ID
- App Registrations
- New Registration
  - Github-azure-vm-cicd
  - Default Directory only - Single Tenant
- *App registrations define application identity, authentication, permissions, and API access in Azure.*

Link the App Registration to the Azure Subscription (Assign RBAC Role)

- Navigate to **Home > Subscriptions > Azure Subscription 1**
- Open **Access Control (IAM)**
- Click **Add > Add Role Assignment**
- Select the **Contributor** Role
- Click **Next**, then choose: User, Group, or Service Principal
- Click Select Members and search for github-azure-vm-cicd
- Select it > **Review + Assign**



This grants GitHub's identity permission to create/update Azure resources using Terraform.

Enable OIDC federated identity

OIDC (OpenID Connect) federated identity is a way for GitHub Actions to log into Azure without using secrets, passwords, or client secrets. It sets up a direct trust between Azure & Github.

Go to App registration > github-azure-vm-cicd | certifcates & secrets > Federated Credentials > Add credential

- Federated credential scenario : GitHub Actions deploying Azure resources
- Organization: anguzz
- Repository: azure-vm-cicd
- Entity type: Branch
- GitHub branch name: main
- Credential details: github-oidc-main
- Description: OIDC federated credential for GitHub Actions (main branch) deploying Azure resources.

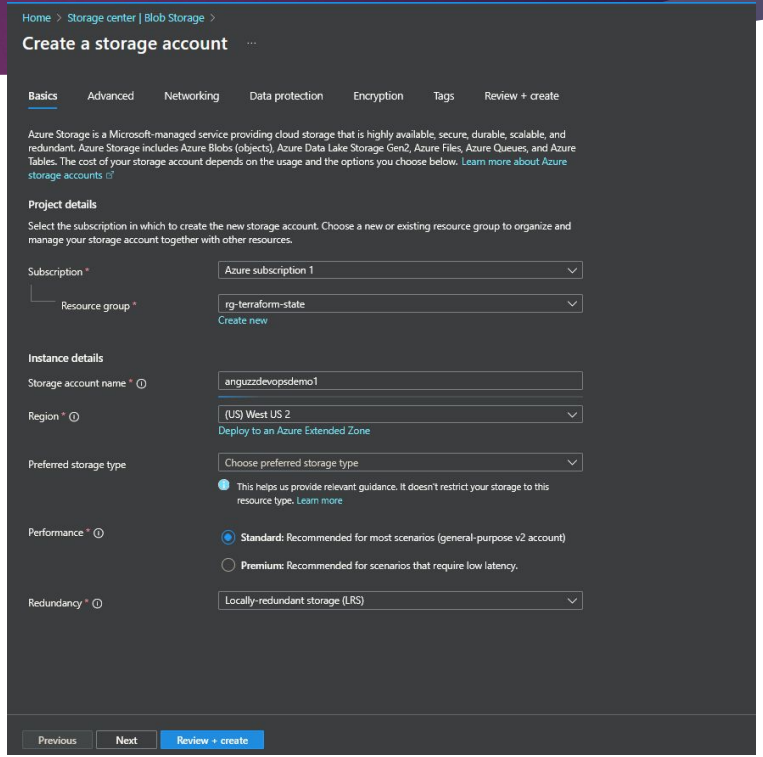
Create Azure Storage Account for Remote State

To ensure deployment state is maintained across GitHub Actions runs and to prevent the "resource already exists" error, Terraform state must be stored remotely in a dedicated Azure Storage Account.

Note: This Storage Account must be created **manually** or using a separate, one-time Terraform run, as it is a dependency for all subsequent infrastructure deployments.

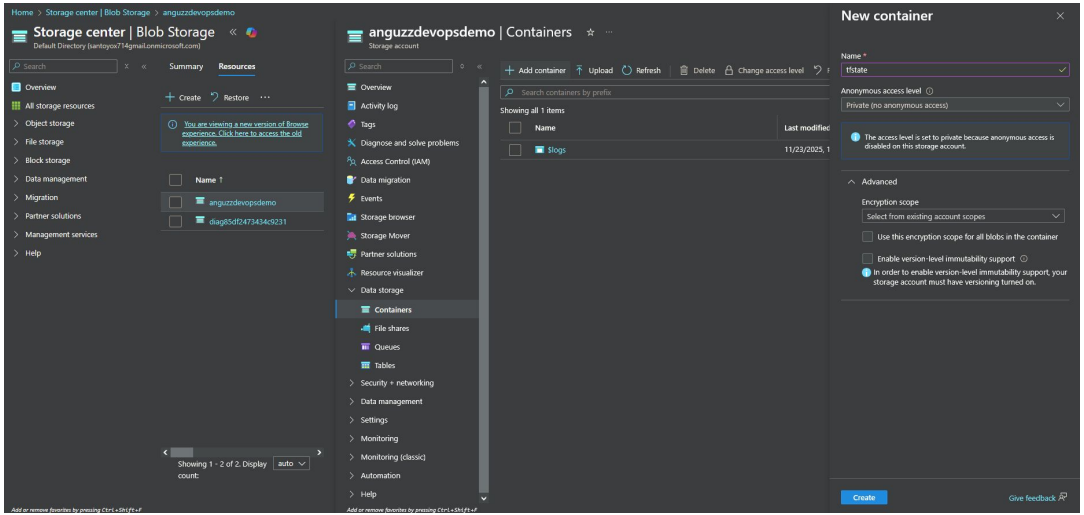
Create Azure Storage Account for Remote State

- Navigate to the Azure Portal and create a new **Resource Group** dedicated to state (e.g., rg-terraform-state). (Optional, but recommended for clean separation.)
- Create a **Storage Account** with a globally unique name (e.g., anguzzdevopsdemo) inside this Resource Group.
  - **Account Kind:** General-purpose v2
  - **Performance:** Standard
  - **Redundancy:** LRS (*Locally-redundant storage) (Not production critical, other redundancies are more expensive but used for critical apps, since this is a demo we went with LRS*

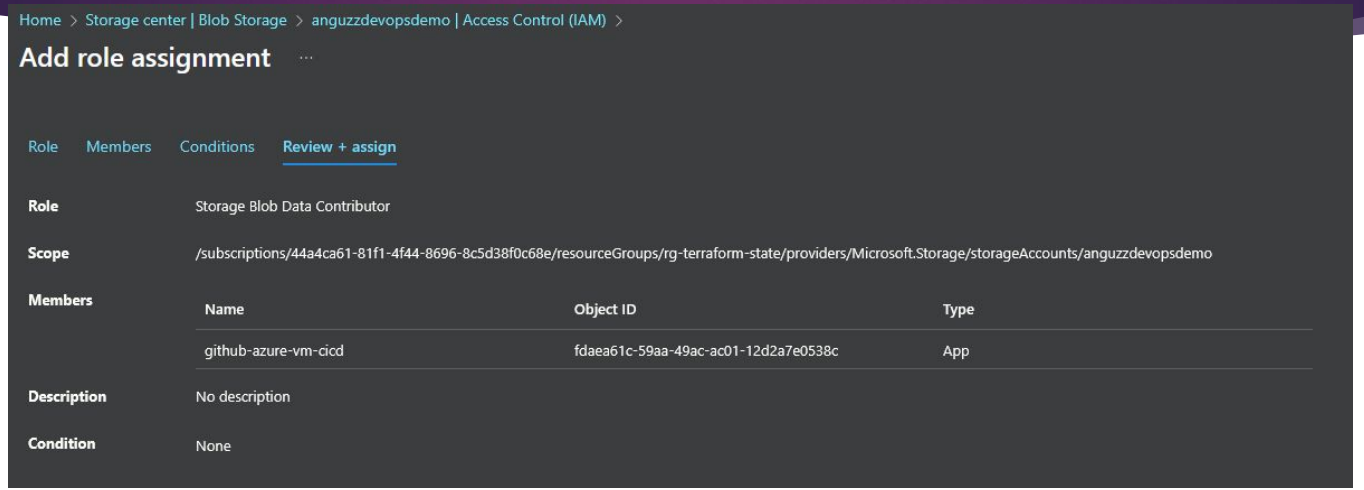


Create Azure Storage Account for Remote State

- Once the Storage Account is created, navigate to the **Containers** blade and create a new container named **tfstate**.
- This dedicated account is referenced in terraform/providers.tf to define the **backend** location.



Grant Storage Data Access

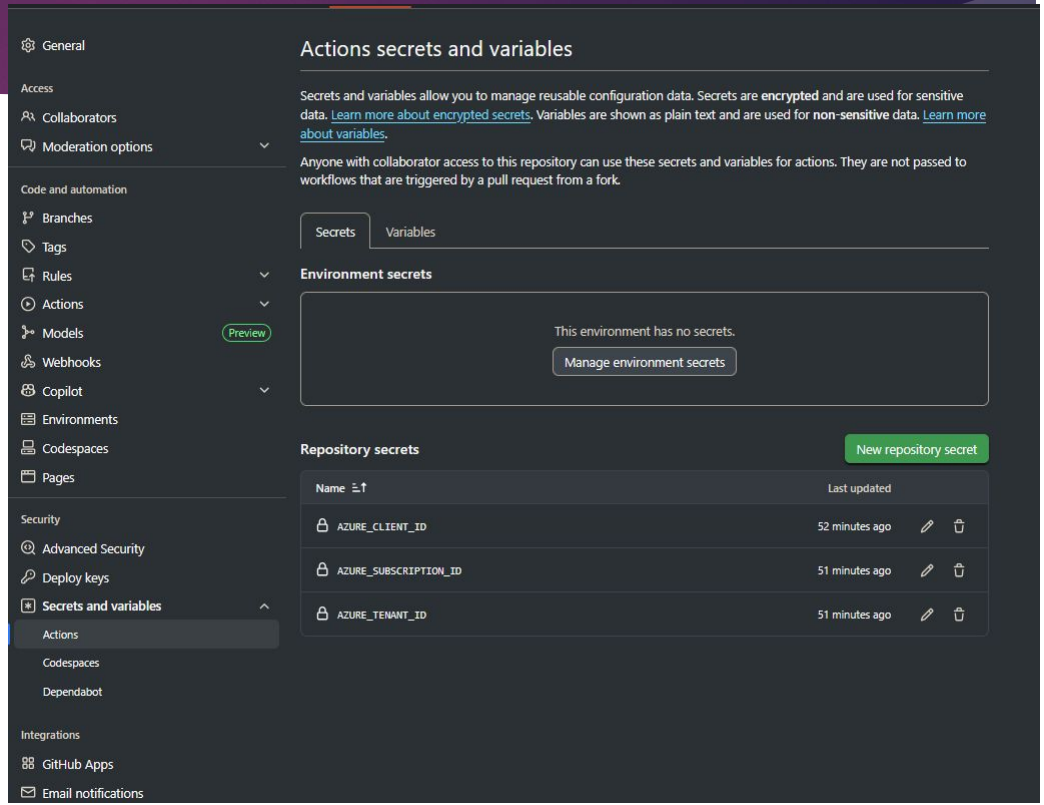


While the general **Contributor** role allows managing resources, it does **not** grant permission to read or write data *inside* a Storage Account. To allow Terraform to save the state file (.tfstate), you must explicitly assign a data-plane role.

- Navigate to the Storage Account you created (e.g., anguzzdevopsdemo).
- Open **Access control (IAM)** > **Add role assignment**.
- Select the **Storage Blob Data Contributor** role.
- Assign it to your App Registration: github-azure-vm-cicd.

Create github actions and add secrets

- Under the repo, go to settings
- Go to secrets & variables > actions
- New Secret
- Add the secrets and the corresponding values in .env-example



Create github action workflow

Github actions run Terraform against the files in the terraform/ folder:

Terraform Overview

- **main.tf** – defines all Azure resources (resource group, network, public IP, NIC, and the Ubuntu VM).
- **variables.tf** – holds input values such as VM name, region, size, admin username, and SSH key path. You can customize these.
- **outputs.tf** – prints useful information after deployment, such as the VM's public IP.

Terraform will automatically create any resources that do not already exist (including the Resource Group). GitHub Actions handles the deployment by running terraform init, plan, and apply on each push to main.

Redeployment

Redeploying the VM

To redeploy the VM at any time, go to:

- GitHub > Actions > Deploy Azure VM > Run workflow

This will:

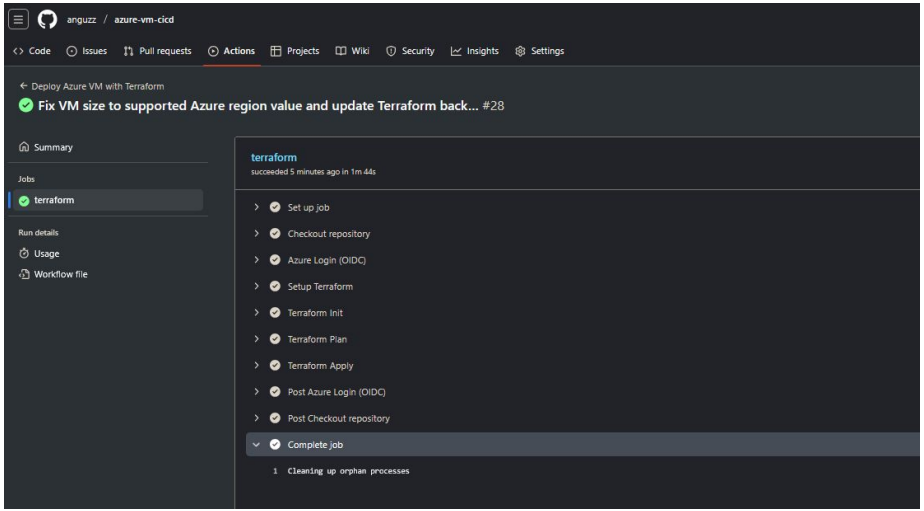
- Force a full redeploy
- Apply any new Terraform changes
- Recreate the VM if it was deleted in Azure
- Ensure the VM always matches the state defined in code

Deployment (Working CI/CD Pipeline)

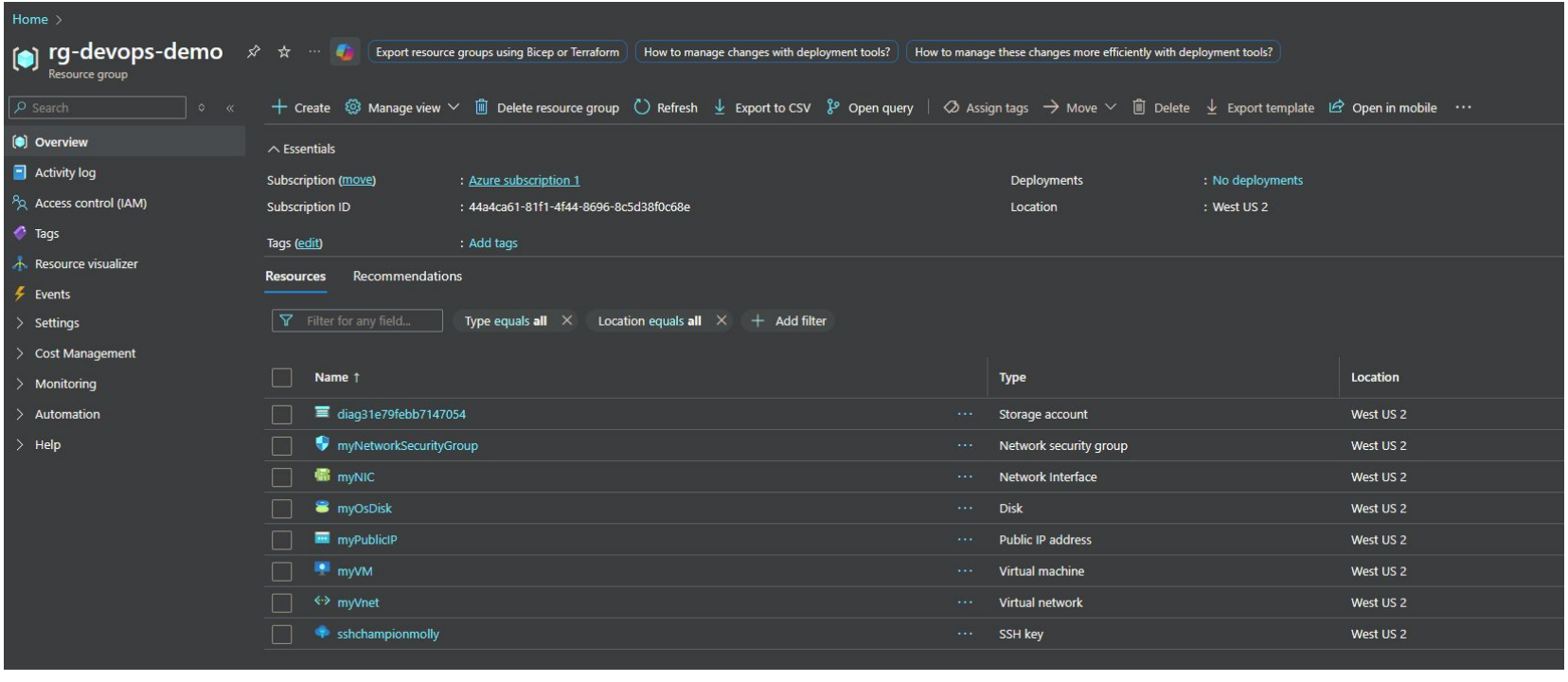
Deploying Azure VM with Terraform & Github actions

A full end-to-end deployment was successfully executed through GitHub Actions:

- Code pushed to **main**
- GitHub Actions authenticated to Azure using **OIDC**
- Terraform ran **init → plan → apply**
- All Azure resources were created automatically
- The VM is now reachable via SSH
- Remote state stored in Azure Storage



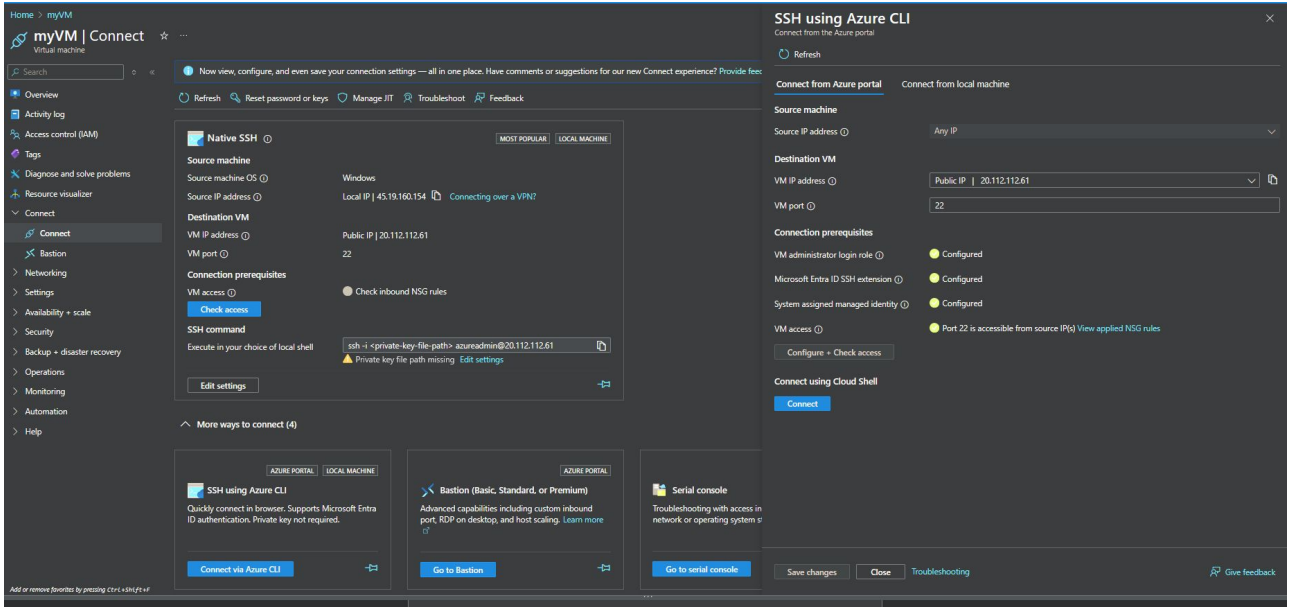
Deployed Resources:



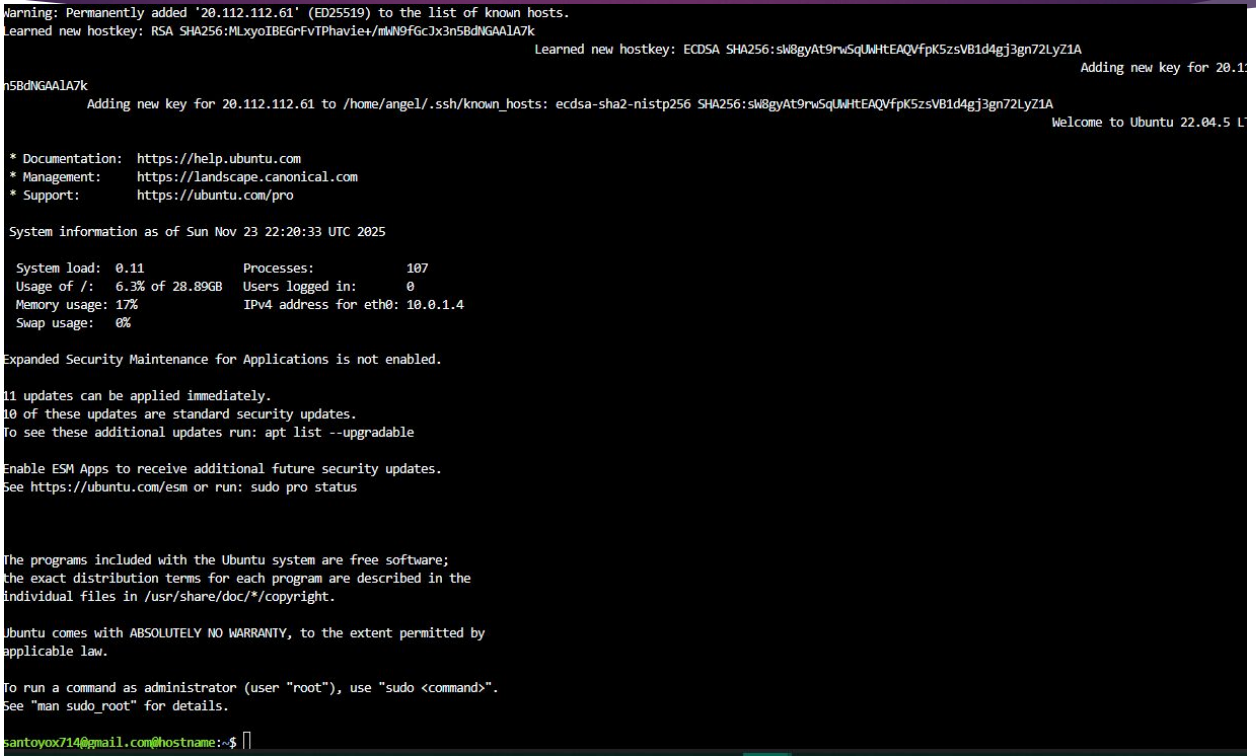
Connecting to the Deployed VM

Connect via Azure CLI
(recommended for Initial Access)

The Azure CLI offers a quick way to establish an SSH connection without manually handling the private key on your local machine, using built-in identity management (assuming you are authenticated locally via az login).



Connect via Azure CLI
(recommended for Initial Access)



Connect via SSH
(Requires Local Private Key)

Alternatively, you can connect directly using the private key that Terraform dynamically generated and stored in your remote state file.

- This key is also used for SSH connectivity through ansible, since ansible needs to go through the same pipeline using OIDC

- Trying to generate separate private SSH key and use it will not auth with OIDC and fail our checks with our tf state

Retrieve the Key: Run the following commands in your local terminal (ensure you have run `terraform init` and `az login` first):

```
terraform output -raw private_key_pem > mykey.pem
# Linux/Mac only:
chmod 600 mykey.pem
```

SSH Command: Use the key file and the public IP address:

```
ssh -i mykey.pem azureadmin@<VM_PUBLIC_IP_ADDRESS>
```

Ansible Setup

Ansible Setup

1. **Terraform generates the SSH keypair:** Terraform uses azapi_resource_action to create an SSH key. The public key is injected into the VM at build time.
2. **Terraform remote state stores the private key securely:** The private key is available as a sensitive output:

```
output "private_key_pem" {
  value     = azapi_resource_action.ssh_public_key_gen.output.privateKey
  sensitive = true
}
```

3. **GitHub Actions retrieves the private key at runtime:** The workflow loads the private key directly from the Terraform output:

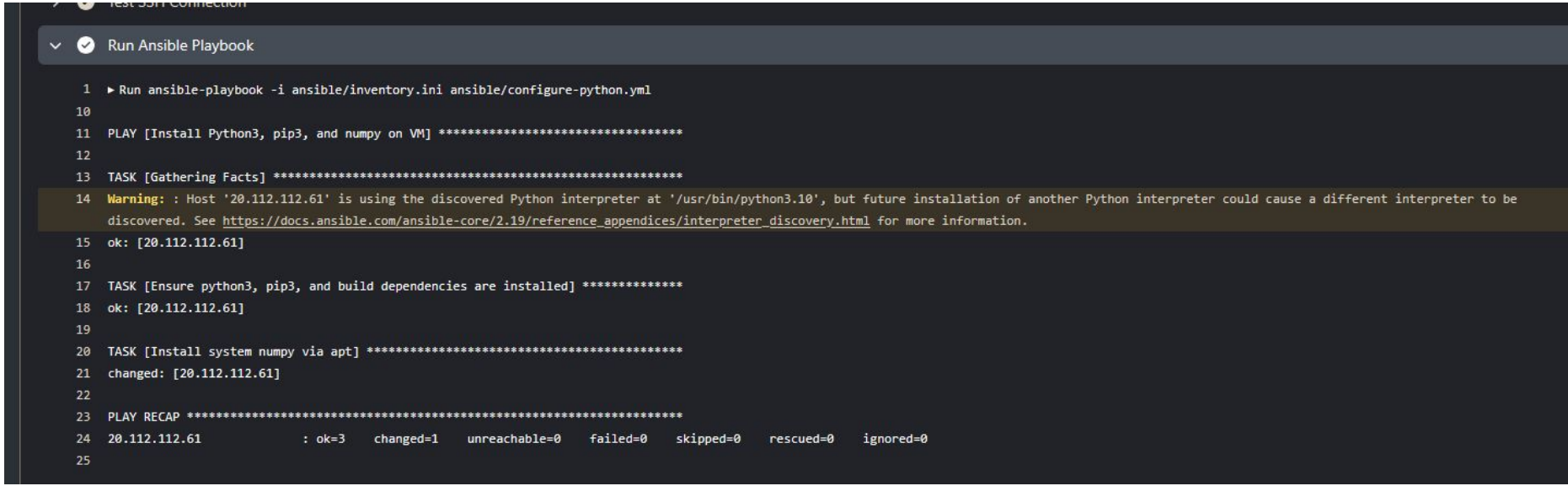
```
PRIVATE_KEY=$(terraform output -raw private_key_pem)
echo "$PRIVATE_KEY" > ~/.ssh/id_rsa
chmod 600 ~/.ssh/id_rsa
```

4. **A dynamic inventory is generated for Ansible:** The VM IP and SSH key are inserted into `ansible/inventory.ini` automatically:

```
echo "[vm]" > ansible/inventory.ini
echo "$VM_IP ansible_user=azureadmin ansible_ssh_private_key_file=$HOME/.ssh/id_rsa ansible_ssh_common_args='-
```

5. **GitHub Actions runs the Ansible playbook:** No secrets, no manual keys, no local setup.

Build Validation



Client Side Validation

All dependencies in our configure-python.yml playbook were installed

```
santoyox714@gmail.com@hostname:~$ python3 --version
Python 3.10.12
santoyox714@gmail.com@hostname:~$ pip3 --version
pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)
santoyox714@gmail.com@hostname:~$ pip3 show numpy
Name: numpy
Version: 2.2.6
Summary: Fundamental package for array computing in Python
Home-page:
Author: Travis E. Oliphant et al.
Author-email:
License: Copyright (c) 2005-2024, NumPy Developers.
         All rights reserved.

         Redistribution and use in source and binary forms, with or without
         modification, are permitted provided that the following conditions are
         met:
```

Client Side Validation II

```
santoyox714@gmail.com@hostname:~$ dpkg -l | grep python3
ii  libpython3-dev:amd64                3.10.6-1~22.04.1          amd64        header files and a static library for Python (default)
ii  libpython3-stdlib:amd64             3.10.6-1~22.04.1          amd64        interactive high-level object-oriented language (default python3 version)
ii  libpython3.10:amd64                 3.10.12-1~22.04.12        amd64        Shared Python runtime library (version 3.10)
ii  libpython3.10-dev:amd64             3.10.12-1~22.04.12        amd64        Header files and a static library for Python (v3.10)
ii  libpython3.10-minimal:amd64         3.10.12-1~22.04.12        amd64        Minimal subset of the Python language (version 3.10)
ii  libpython3.10-stdlib:amd64          3.10.12-1~22.04.12        amd64        Interactive high-level object-oriented language (standard library, version 3.10)
ii  python3                             3.10.6-1~22.04.1          amd64        interactive high-level object-oriented language (default python3 version)
ii  python3-apport                      2.20.11-0ubuntu82.10      all          Python 3 library for Apport crash report handling
ii  python3-apt                         2.4.0ubuntu4              amd64        Python 3 interface to libapt-pkg
ii  python3-attr                        21.2.0-1                  all          Attributes without boilerplate (Python 3)
ii  python3-automat                     20.2.0-1                  all          Self-service finite-state machines for the programmer on the go
ii  python3-babel                       2.8.0+dfsg.1-7            all          tools for internationalizing Python applications - Python 3.x
ii  python3-bcrypt                      3.2.0-1build1             amd64        password hashing library for Python 3
ii  python3-blinker                     1.4+dfsg1-0.4             all          fast, simple object-to-object and broadcast signaling library
ii  python3-certifi                     2020.6.20-1               all          root certificates for validating SSL certs and verifying TLS hosts (python3)
ii  python3-cffi-backend:amd64          1.15.0-1build2            amd64        Foreign Function Interface for Python 3 calling C code - runtime
ii  python3-chardet                     4.0.0-1                   all          universal character encoding detector for Python3
ii  python3-click                       8.0.3-1                   all          Wrapper around optparse for command line utilities - Python 3.x
ii  python3-colorama                    0.4.4-1                   all          Cross-platform colored terminal text in Python - Python 3.x
ii  python3-commandnotfound             22.04.0                   all          Python 3 bindings for command-not-found.
ii  python3-configobj                   5.0.6-5ubuntu0.1          all          simple but powerful config file reader and writer for Python 3
ii  python3-constantly                  15.1.0-2                  all          Symbolic constants in Python
ii  python3-cryptography                3.4.8-1ubuntu2.2          amd64        Python library exposing cryptographic recipes and primitives (Python 3)
ii  python3-dbus                        1.2.18-3build1            amd64        simple interprocess messaging system (Python 3 interface)
ii  python3-debconf                     1.5.79ubuntu1             all          interact with debconf from Python 3
ii  python3-debian                      0.1.43ubuntu1.1           all          Python 3 modules to work with Debian-related data formats
ii  python3-dev                         3.10.6-1~22.04.1          amd64        header files and a static library for Python (default)
ii  python3-distro                      1.7.0-1                   all          Linux OS platform information API
ii  python3-distro-info                 1.1ubuntu0.2              all          information about distributions' releases (Python 3 module)
ii  python3-distupgrade                 1:22.04.20                all          manage release upgrades
```

Resources

Github repo:
https://github.com/anguzz/azure-vm-cicd

Azure documentation:

https://learn.microsoft.com/en-us/azure/virtual-machines/windows/quick-create-terraform

https://learn.microsoft.com/en-us/azure/virtual-network/ip-services/public-ip-addresses#limitations

https://learn.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-terraform?tabs=azure-cli

https://learn.microsoft.com/en-us/azure/virtual-machines/windows/quick-create-terraform?tabs=azure-cli

Github URL QR

Thank You for Listening

Any Questions?