

Final group work - GBA 6270

Angel Santoyo, Anna Betova, Thong Nguyen

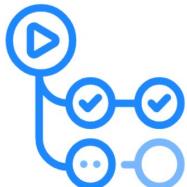
What is our project?

- ❖ **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
```

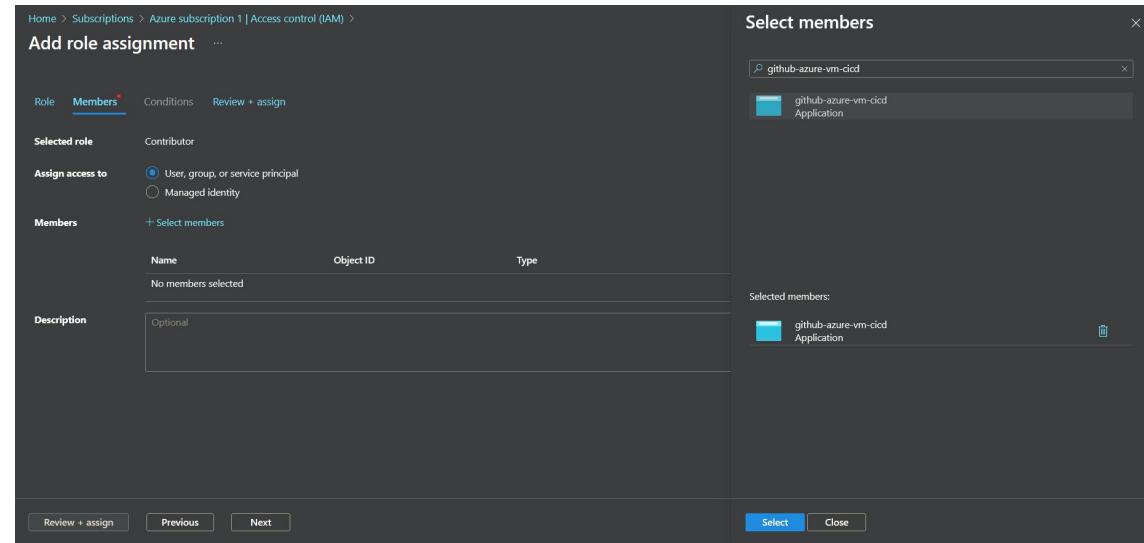
Setup Process

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 | certificates & 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.

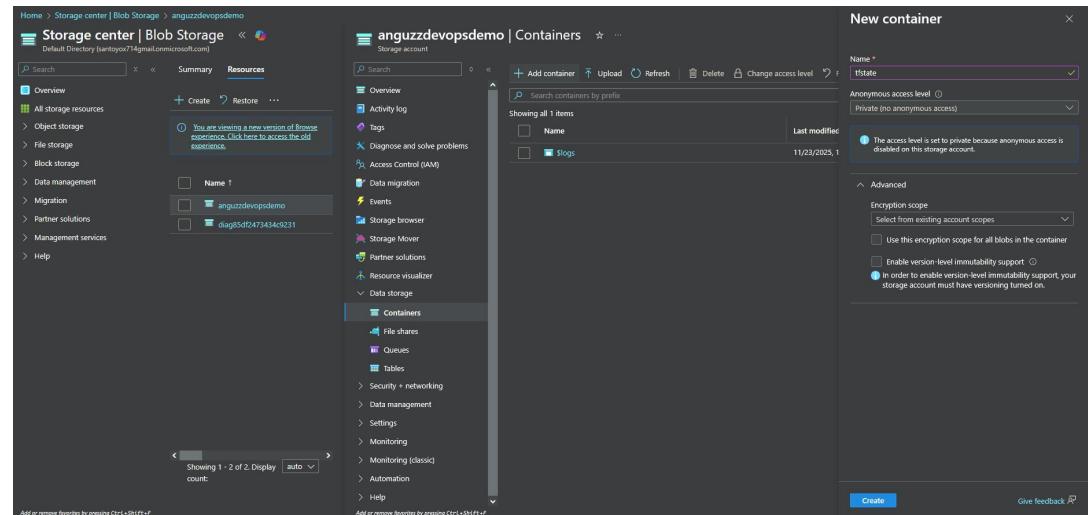
Create Azure Storage Account for Remote State

- ❖ 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)

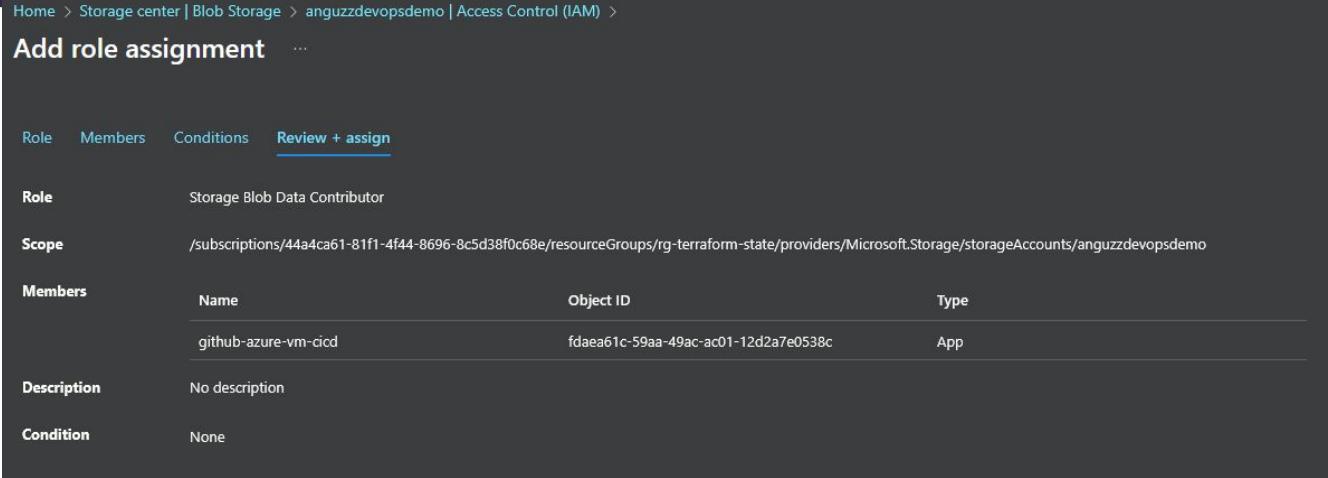
The screenshot shows the 'Create a storage account' wizard in the Azure portal. The 'Basics' tab is selected. The 'Subscription' dropdown is set to 'Azure subscription 1'. The 'Resource group' dropdown is set to 'rg-terraform-state' with a 'Create new' option available. The 'Storage account name' field contains 'anguzzdevopsdemo'. The 'Region' dropdown is set to '(US) West US 2' with a 'Deploy to an Azure Extended Zone' option. The 'Preferred storage type' dropdown is set to 'Choose preferred storage type' with a note: 'This helps us provide relevant guidance. It doesn't restrict your storage to this resource type.' The 'Performance' section has two options: 'Standard' (selected) and 'Premium'. The 'Redundancy' dropdown is set to 'Locally-redundant storage (LRS)'. At the bottom, there are 'Previous' and 'Next' buttons, and a highlighted 'Review + create' button.

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



The screenshot shows the 'Add role assignment' page in the Azure portal. The top navigation bar includes 'Home', 'Storage center | Blob Storage', 'anguzzdevopsdemo', 'Access Control (IAM)', and a 'Review + assign' button which is underlined, indicating it's the active step. Below this, the 'Role' is set to 'Storage Blob Data Contributor' and the 'Scope' is the specific storage account resource. The 'Members' section lists a single member: 'github-azure-vm-cicd' with object ID 'fdaea61c-59aa-49ac-ac01-12d2a7e0538c' and type 'App'. There are also sections for 'Description' (No description) and 'Condition' (None).

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**

The screenshot shows the GitHub repository settings page for managing secrets and variables. The left sidebar lists various repository settings like General, Access, and Code and automation. The 'Actions' section is currently selected, indicated by a highlighted bar at the bottom of the sidebar.

The main content area is titled 'Actions secrets and variables'. It contains two tabs: 'Secrets' (selected) and 'Variables'. The 'Secrets' tab displays the 'Environment secrets' section, which shows a message: 'This environment has no secrets.' with a 'Manage environment secrets' button. The 'Repository secrets' section lists three secrets:

| Name | Last updated | Actions | |
|-----------------------|----------------|---------|--|
| AZURE_CLIENT_ID | 52 minutes ago | | |
| AZURE_SUBSCRIPTION_ID | 51 minutes ago | | |
| AZURE_TENANT_ID | 51 minutes ago | | |

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:

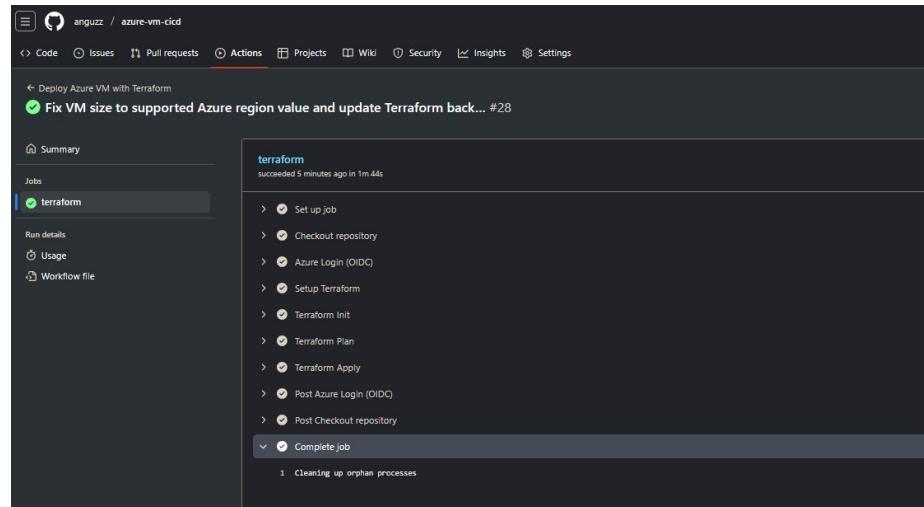
- ❖ Force a full redeploy
- ❖ Apply any new Terraform changes
- ❖ Recreate the VM if it was deleted in Azure
- ❖ Ensure the VM always matches the state defined in code

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:

Home > rg-devops-demo Resource group

Export resource groups using Bicep or Terraform | How to manage changes with deployment tools? | How to manage these changes more efficiently with deployment tools?

+ Create Manage view Delete resource group Refresh Export to CSV Open query Assign tags Move Delete Export template Open in mobile ...

Overview

Activity log Access control (IAM) Tags Resource visualizer Events Settings Cost Management Monitoring Automation Help

Subscription (move) : Azure subscription_1 Deployments : No deployments

Subscription ID : 44a4ca61-81f1-4f44-8696-8c5d38f0c68e Location : West US 2

Tags (edit) : Add tags

Resources Recommendations

Filter for any field... Type equals all × Location equals all × + Add filter

| | Type | Location |
|--------------------------|------------------------|-----------|
| <input type="checkbox"/> | Storage account | West US 2 |
| <input type="checkbox"/> | Network security group | West US 2 |
| <input type="checkbox"/> | Network Interface | West US 2 |
| <input type="checkbox"/> | Disk | West US 2 |
| <input type="checkbox"/> | Public IP address | West US 2 |
| <input type="checkbox"/> | Virtual machine | West US 2 |
| <input type="checkbox"/> | Virtual network | West US 2 |
| <input type="checkbox"/> | SSH key | West US 2 |
| <input type="checkbox"/> | diag31e79febb7147054 | |
| <input type="checkbox"/> | myNetworkSecurityGroup | |
| <input type="checkbox"/> | myNIC | |
| <input type="checkbox"/> | myOsDisk | |
| <input type="checkbox"/> | myPublicIP | |
| <input type="checkbox"/> | myVM | |
| <input type="checkbox"/> | myVNet | |
| <input type="checkbox"/> | sshchampionmolly | |

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`).

The screenshot shows the Azure portal interface for connecting to a virtual machine named 'myVM'. On the left, the 'Connect' section of the navigation menu is selected. In the center, the 'Native SSH' connection method is chosen. The 'SSH command' field contains the command: `ssh -i <private-key-file-path> azureadmin@20.112.112.61`. A warning message indicates that the private key file path is missing. Below the command, there are 'Edit settings' and 'Check access' buttons. To the right, the 'SSH using Azure CLI' blade is open, showing connection details: Source machine IP 45.19.160.154, Destination VM IP 20.112.112.61, and Port 22. It also lists connection prerequisites: VM access (Configured), Microsoft Entra ID SSH extension (Configured), System assigned managed identity (Configured), and Port 22 is accessible from source IP (Configured). At the bottom, there are 'Connect using Cloud Shell' and 'Connect' buttons.

Connect via Azure CLI (recommended for Initial Access)

```
Warning: Permanently added '20.112.112.61' (ED25519) to the list of known hosts.
Learned new hostkey: RSA SHA256:MLxyoIBEGrFvTPhavie+mwN9fcJx3n5BdNGAA1A7k
                                                               Learned new hostkey: ECDSA SHA256:sw8gyAt9rwSqUMHtEAQVFpK5zsVB1d4gj3gn72LyZ1A
                                                               Adding new key for 20.112.112.61
h5BdNGAA1A7k
                                                               Adding new key for 20.112.112.61 to /home/angel/.ssh/known_hosts: ecdsa-sha2-nistp256 SHA256:sw8gyAt9rwSqUMHtEAQVFpK5zsVB1d4gj3gn72LyZ1A
                                                               Welcome to Ubuntu 22.04.5 LTS

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro

System information as of Sun Nov 23 22:20:33 UTC 2025

System load:  0.11      Processes:          187
Usage of /:   6.3% of 28.89GB  Users logged in:     0
Memory usage: 17%
Swap usage:   0%
IPv4 address for eth0: 10.0.1.4

Expanded Security Maintenance for Applications is not enabled.

11 updates can be applied immediately.
10 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

santoyox714@gmail.com@hostname:~$
```

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

```
▶ test SSH Connection
Run Ansible Playbook

 1 ► Run ansible-playbook -i ansible/inventory.ini ansible/configure-python.yml
10
11 PLAY [Install Python3, pip3, and numpy on VM] ****
12
13 TASK [Gathering Facts] ****
14 Warning: : Host '20.112.112.61' is using the discovered Python interpreter at '/usr/bin/python3.10', but future installation of another Python interpreter could cause a different interpreter to be discovered. See https://docs.ansible.com/ansible-core/2.19/reference\_appendices/interpreter\_discovery.html for more information.
15 ok: [20.112.112.61]
16
17 TASK [Ensure python3, pip3, and build dependencies are installed] ****
18 ok: [20.112.112.61]
19
20 TASK [Install system numpy via apt] ****
21 changed: [20.112.112.61]
22
23 PLAY RECAP ****
24 20.112.112.61      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
25
```

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

```
requested by:
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                2.20.11-0ubuntu8.10       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>

Github URL QR



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>

Thank You for Listening

Any Questions?

