Setting Up a GitHub Self-Hosted Runner on an Azure VM

This document provides a detailed, step-by-step guide to setting up a GitHub self-hosted runner on an Azure VM

Prerequisites

- Azure subscription and the ability to create and manage resources.
- GitHub repository with administrative access.
- SSH key pair for accessing the VM.

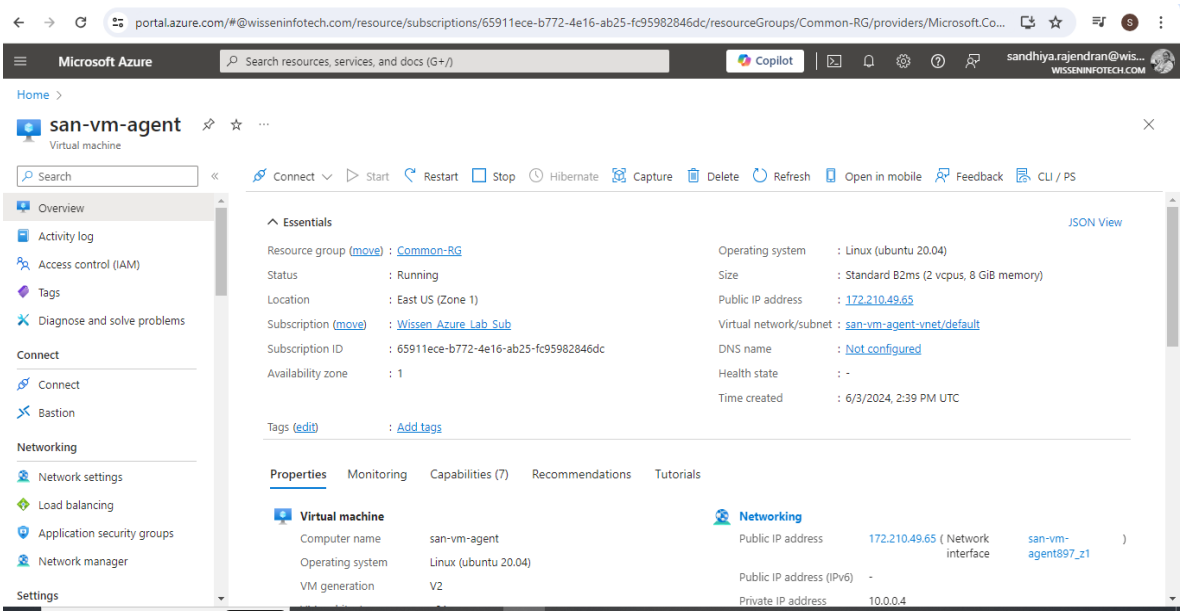
Step 1: Create and Configure the Azure VM

1.1 Log in to Azure Portal

1. Navigate to the Azure Portal.

1.2 Create a New Virtual Machine

2. Click **Create a resource**.
3. Select **Virtual Machine**.
4. Fill in the required details:
    - **Resource Group**: Choose an existing one or create a new one.
    - **Virtual Machine Name**: Provide a name for your VM.
    - **Region**: Select a region.
    - **Image**: Choose a Linux distribution (e.g., Ubuntu 20.04 LTS).
    - **Size**: Select a VM size that meets your requirements.
5. Configure the **Administrator account**:
    - **Authentication type**: SSH public key.
    - **Username**: Provide a username.
    - **SSH public key**: Paste your SSH public key
6. Configure the **Inbound port rules**:
    - Allow SSH (port 22).
7. Click **Review + create** and then **Create**.



1.3 Connect to the VM

8. Once the VM is created, go to the **Virtual Machines** section.
9. Select your VM and note its public IP address.
10. Connect to the VM using SSH:

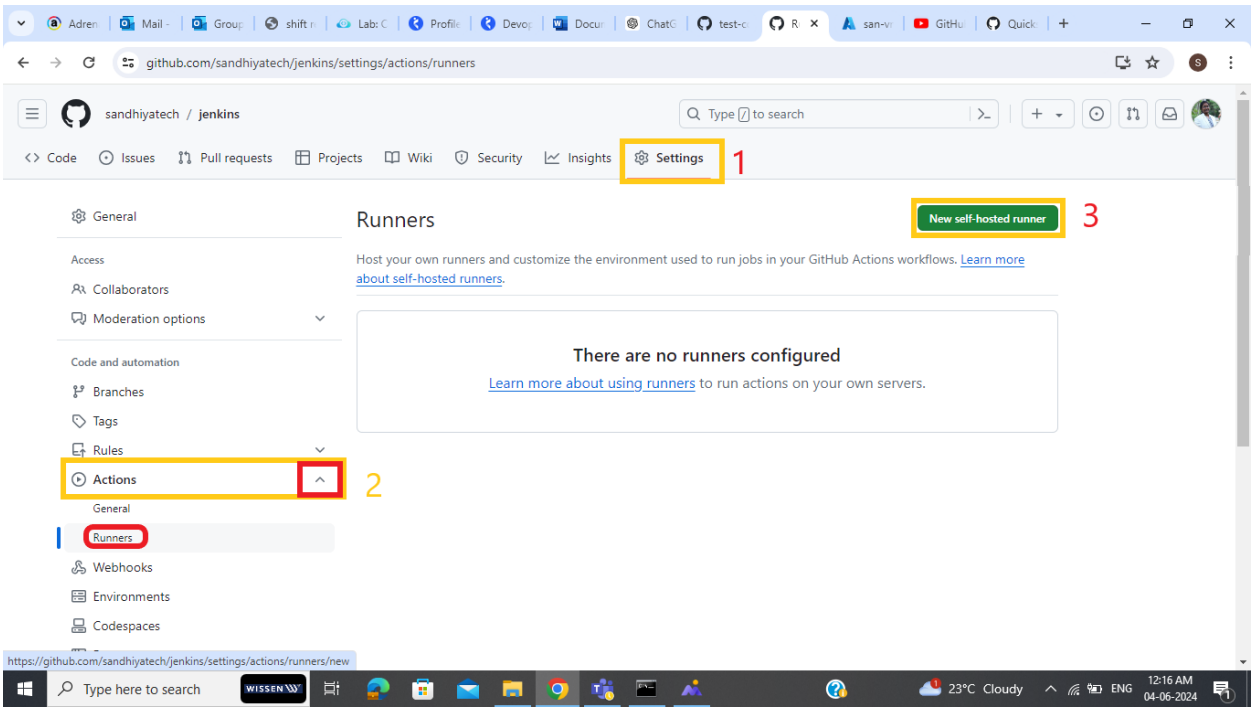
Command: ssh path/to/privatekey <your-username>@<public-ip-address>

```
C:\Users\005377\Downloads>ssh -i C:\Users\005377\Downloads\san-vm-agent_key.pem azureuser@172.210.49.65
The authenticity of host '172.210.49.65 (172.210.49.65)' can't be established.
ECDSA key fingerprint is SHA256:zqrQcIGM8F1xCOZpVJoCa3dJB8bKjkPGqcTtNwv0N5s.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.210.49.65' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1064-azure x86_64)

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

 System information as of Mon Jun  3 14:41:14 UTC 2024

  System load:  0.48              Processes:             123
  Usage of /:   5.1% of 28.89GB   Users logged in:       0
  Memory usage: 3%                IPv4 address for eth0: 10.0.0.4
  Swap usage:   0%
```

Step 2: Install and Configure GitHub Runner

2.1 Configure the Runner

11. Go to your GitHub repository.
12. Navigate to **Settings** > **Actions** > **Runners**.
13. Click **New self-hosted runner** and follow the instructions to generate a runner token.



2.2 Create a Directory for the Runner

```
azureuser@san-vm-agent:~$ mkdir actions-runner && cd actions-runner
azureuser@san-vm-agent:~/actions-runner$ curl -o actions-runner-linux-x64-2.316.1.tar.gz -L https://github.com/actions/runner/releases/download/v2.316.1/actions-runner-
linux-x64-2.316.1.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  179M  100  179M    0     0   255M      0 --:--:-- --:--:-- --:--:--  255M
azureuser@san-vm-agent:~/actions-runner$ echo "d62de2400eeeacd195db91e2ff011bfb646cd5d85545e81d8f78c436183e09a8  actions-runner-linux-x64-2.316.1.tar.gz" | shasum -a 25
6 -c
actions-runner-linux-x64-2.316.1.tar.gz: OK
azureuser@san-vm-agent:~/actions-runner$ tar xzf ./actions-runner-linux-x64-2.316.1.tar.gz
azureuser@san-vm-agent:~/actions-runner$ ./config.sh --url https://github.com/sandhiyadevopsorg --token AS33PNDHX6WBBMDMSA6O35LGLXJ3E
```

14. Create a directory for the runner and navigate into it:

```
mkdir actions-runner && cd actions-runner
```

2.3 Download the GitHub Runner

15. Download the latest runner package

```
curl -o actions-runner-linux-x64-2.303.0.tar.gz -L
https://github.com/actions/runner/releases/download/v2.303.0/actions-runner-linux-x64-2.303.0.tar.gz
```

13. Extract the downloaded package:

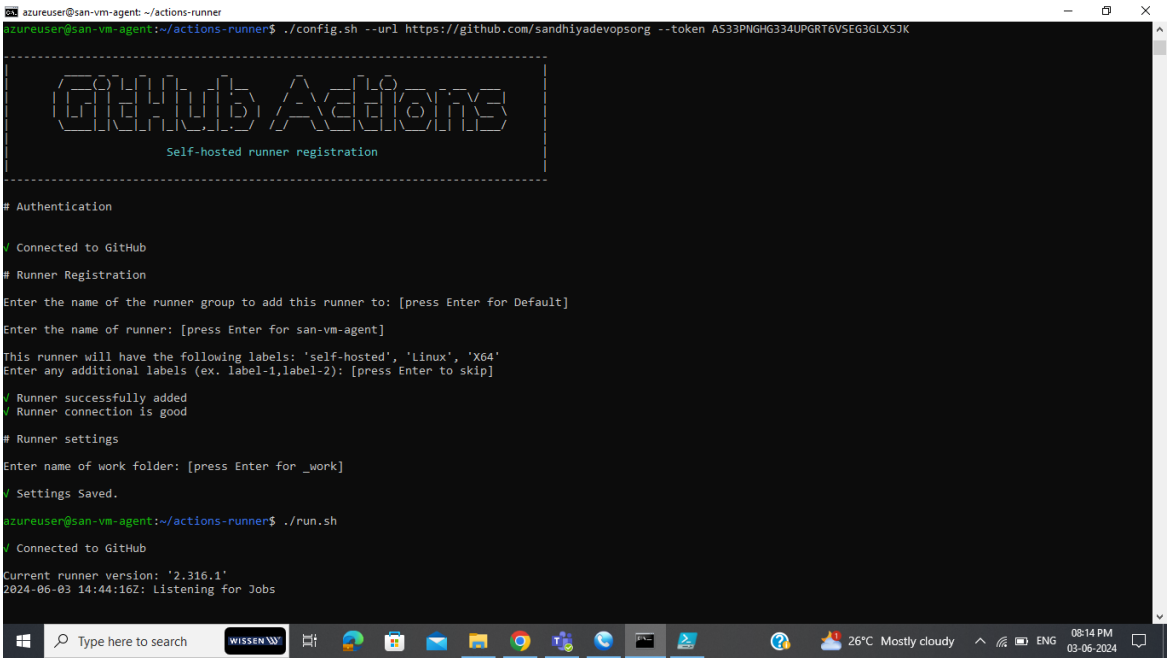
```
tar xzf actions-runner-linux-x64-2.303.0.tar.gz
```

16. Run the configuration script with your repository URL and token:

`./config.sh --url https://github.com/<your-username>/<your-repo> --token <your-generated-token>`

2.4 Install and Start the Runner Service

17. Start the runner:
    ./run.cmd



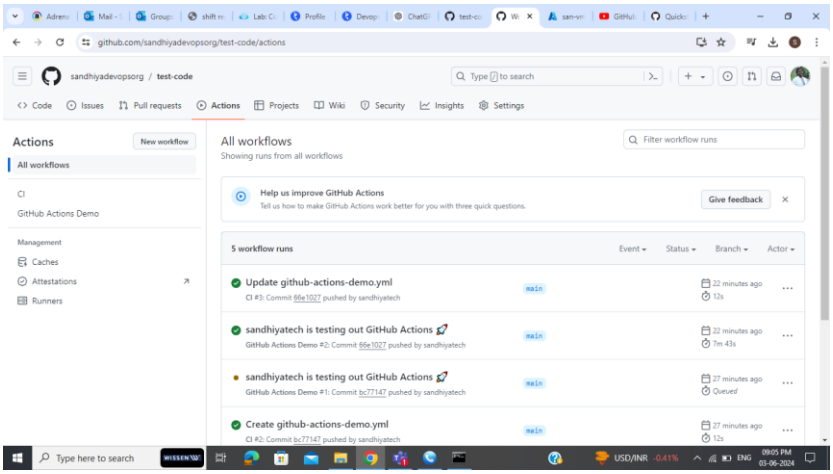
Step 3: Verify the Runner

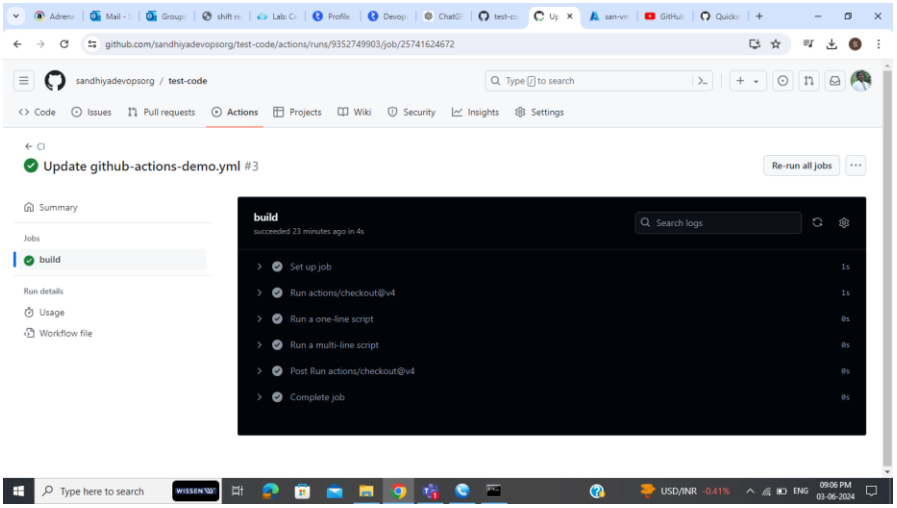
3.1 Check Runner Status on GitHub

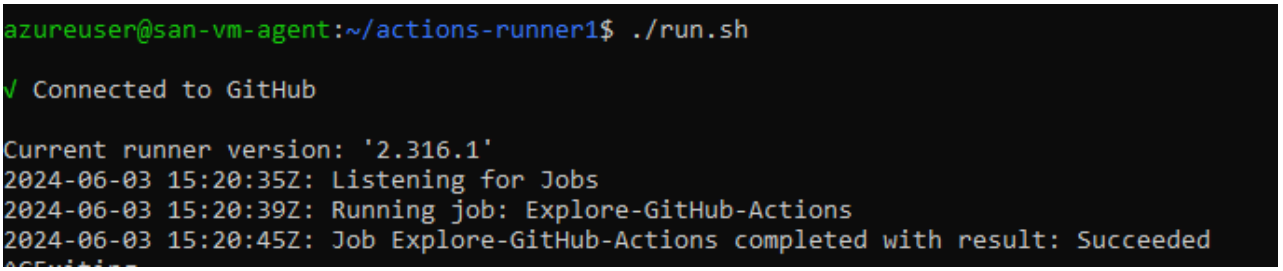
18. Go to **Settings** > **Actions** > **Runners** in your GitHub repository.
19. Ensure the runner is listed and shows as **`Online`**.

3.2 Run a Workflow

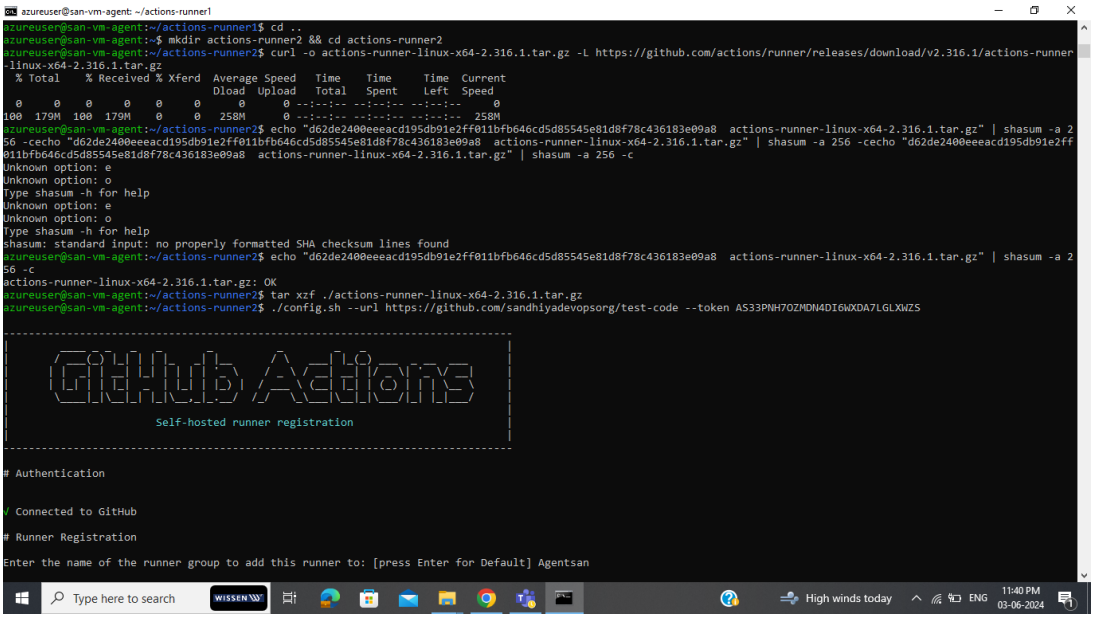
20. Trigger a GitHub Actions workflow to verify that the runner picks up jobs correctly.
21. Github url: https://github.com/sandhiyadevopsorg/test-code/blob/main/.github/workflows/github-actions-demo.yml







Step 4: (Optional) Running Multiple Runners on the Same VM



4.1 Create Another Directory for the Second Runner

22. Create a new directory for the second runner:

```
mkdir actions-runner-2 && cd actions-runner-2
```

4.2 Download, Extract, and Configure the Second Runner

23. Repeat the download and extract steps for the second runner.
24. Configure the second runner with a new token:

```
./config.sh --url https://github.com/<your-username>/<your-repo> --token <your-generated-token-2>
```

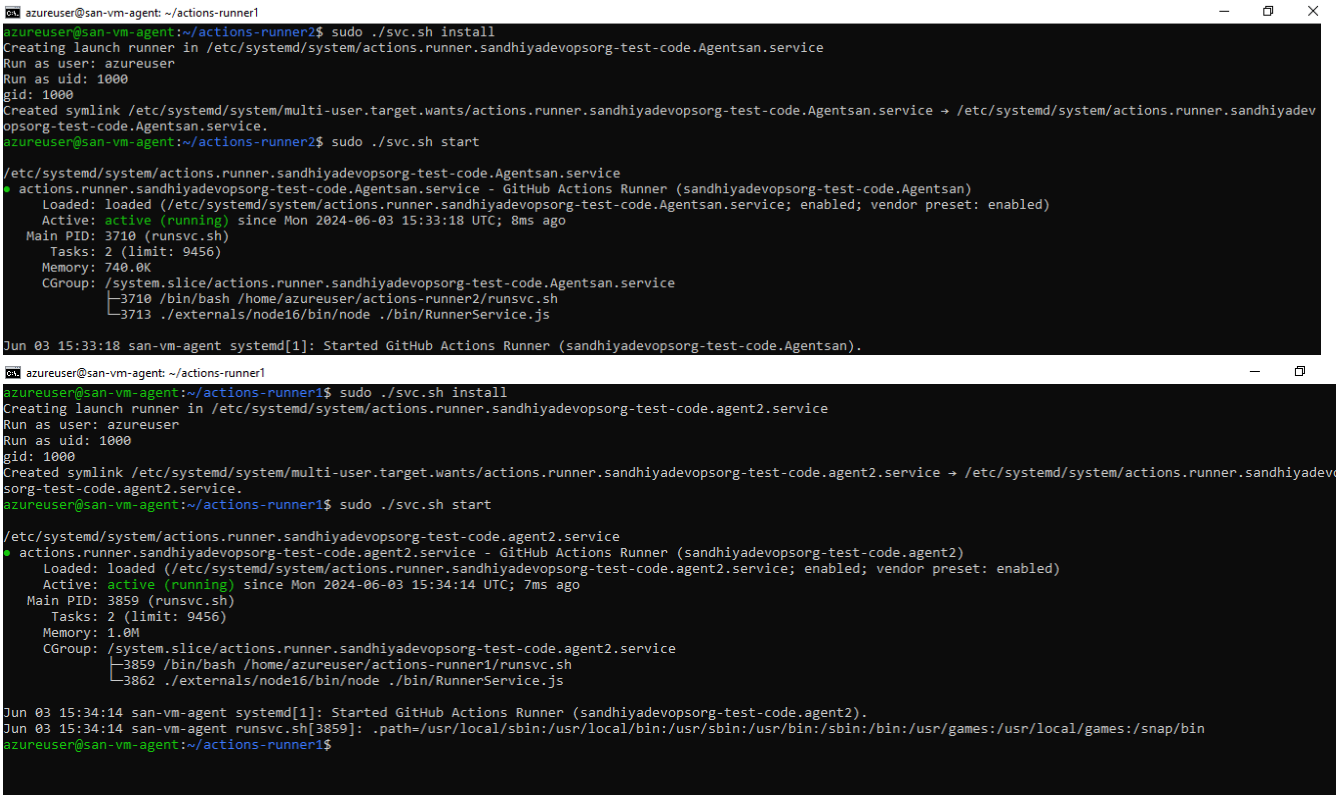
4.3 Install and Start the Second Runner Service

25. Install the second runner service:

```
sudo ./svc.sh install
```

26. Start the second runner service:

```
sudo ./svc.sh start
```



Instead of using ./run.cmd use this runner service to run multiple runners simultaneously.

Summary

By following these steps, you can set up a GitHub self-hosted runner on an Azure Linux VM, allowing you to execute GitHub Actions workflows on your infrastructure. This setup provides greater control over the execution environment and can be scaled by adding multiple runners. Ensure your VM has adequate resources to handle the workload of multiple runners if needed.