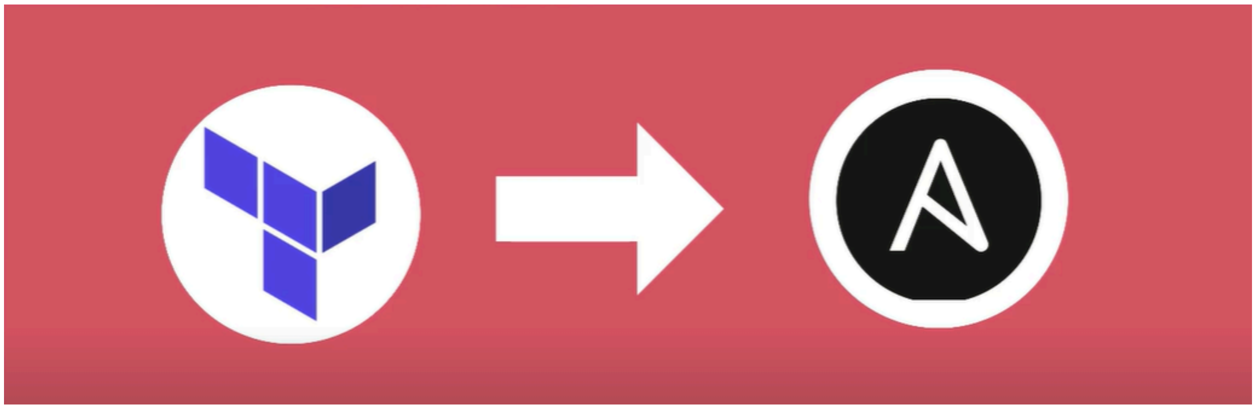
Demo Project:
Ansible Integration in Terraform

Project Description

This project integrates **Ansible** with **Terraform** to automate the provisioning and configuration of an AWS **EC2** instance. Once Terraform creates the server, Ansible is executed automatically to configure the instance with **Docker and other dependencies**.



Step 1: Create AWS EC2 Instance with Terraform

1. Clone the Terraform repository from this repository: `git clone` `https://gitlab.com/twn-devops-projects/ansible/terraform-learn.git`

2. Switch to the appropriate branch: `git checkout feature/deploy-to-ec2-default-components`
3. Update main.tf with the Ansible Provisioner:

```
resource "null_resource" "configure-server" {
   triggers = {
      trigger = aws_instance.myapp-server.public_ip
   }

   provisioner "local-exec" {
      working_dir = "/mnt/c/Users/eduar/devops_projects2/08-ansible/ansible-pr
      command = "ansible-playbook --inventory ${aws_instance.myapp-server.p
   }
}
```

4. Add the private **SSH key** `ssh_key_private` **variable** to `terraform.tfvars`:

```
vpc_cidr_blocks = "10.0.0.0/16"
subnet_cidr_block = "10.0.10.0/24"
avail_zone = "us-east-1a"
env_prefix = "dev"
my_ip = "167.57.247.97/32"
instance_type = "t2.micro"
public_key_location = "/home/eb/.ssh/id_rsa.pub"
ssh_key_private = "/home/eb/.ssh/id_rsa"
image_name = "al2023-ami-2023.*-x86_64"
```

Step 2: Modify the Ansible Playbook

- Add the **"Wait for SSH connection"** block in `deploy-docker-ec2-new-user.yaml`:

```
- name: Wait for SSH connection
  hosts: all
  gather_facts: False
  tasks:
    - name: Ensure SSH port open
      wait_for:
        port: 22
        delay: 10
        timeout: 100
        search_regex: OpenSSH
        host: '{{ (ansible_ssh_host|default(ansible_host))|default(inventory_hostnam
    ansible_connection: local
    ansible_python_interpreter: /usr/bin/python3
```

Why is this necessary?

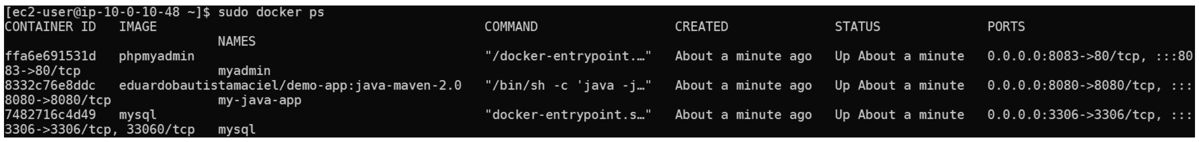
- When Terraform creates the EC2 instance, it may take time for **SSH to be ready**.
- This prevents Ansible from failing by ensuring SSH is **available before continuing**.

Step 3: Apply Terraform Configuration

1. Initialize Terraform: `terraform init`
2. Apply the configuration: `terraform apply -auto-approve`
3. Copy the **EC2 Public IP** from the Terraform output.
4. SSH into the instance: `ssh ec2-user@<EC2_PUBLIC_IP>`

5. Verify that Docker is running: `sudo docker ps`

   Note: Since the containers are started by the user `eduardo` (not `ec2-user`), `sudo` is required.



```
[ec2-user@ip-10-0-10-48 ~]$ sudo docker ps
CONTAINER ID   IMAGE                                         COMMAND                  CREATED              STATUS              PORTS
                          NAMES
ffa6e691531d   phpmyadmin                                    "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:8083->80/tcp, :::80
83->80/tcp                myadmin
8332c76e8ddc   eduardobautistamaciel/demo-app:java-maven-2.0   "/bin/sh -c 'java -j…"   About a minute ago   Up About a minute   0.0.0.0:8080->8080/tcp, :::
8080->8080/tcp            my-java-app
7482716c4d49   mysql                                         "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3306->3306/tcp, :::
3306->3306/tcp, 33060/tcp   mysql
```

Step 4: Clean Up Resources

To destroy all resources:

`terraform destroy --auto-approve`

Troubleshooting

Issue: Ansible is being run in a world-writable directory

Error Message:

> Error running command 'ansible-playbook --inventory 3.92.222.157, --private-
> exit status 4. Output: [WARNING]: Ansible is being run in a world writable direc
> (/mnt/c/Users/eduar/devops_projects2/08-ansible/ansible-projects), ignoring i
> as an ansible.cfg source

Solution:

1. **Check current permissions:**

   `ls -ld /mnt/c/Users/eduar/devops_projects2/08-ansible/ansible-projects`

   If it shows permissions like `drwxrwxrwx`, it means the directory is **world-writable.**

2. Remove World-Writable Permissions:

`chmod o-w /mnt/c/Users/eduar/devops_projects2/08-ansible/ansible-projects`

This removes the **write permission (`w`) for "others" (`o`)** while keeping access for you.

3. Open the **WSL mount configuration file**: `sudo nano /etc/wsl.conf`
4. Add the following lines to **force restrictive permissions**:

   ```
   [automount]
   options = "metadata,umask=022"
   ```

   - `metadata` enables Linux-style file permissions.
   - `umask=022` sets the correct permissions (`755` instead of `777`).
5. **Restart WSL** for the changes to take effect

   Then, reopen your WSL terminal.
6. **Verify that permissions are fixed**: `ls -ld /mnt/c/Users/eduar/devops_projects2/08-ansible/ansible-projects`

   It should now show something like:

   ```
   drwxr-xr-x
   ```

   If it still shows `777`, you may need to manually change the permissions again: `chmod 755 /mnt/c/Users/eduar/devops_projects2/08-ansible/ansible-projects`
7. **Retry Ansible**: `ansible-inventory -i inventory_aws_ec2.yaml --list`
