Demo Project: Automate Kubernetes Deployment

Project Description

In this project, we will:

- **Create an EKS cluster using Terraform**
- **Deploy an application to a new namespace using Ansible**
- **Verify the deployment and expose the application**

Step 1: Create EKS cluster with Terraform

1. **Clone the Terraform repository**:

```
git clone https://gitlab.com/twn-devops-projects/ansible/terraform-learn.git
cd terraform-learn
git checkout feature/eks
```

2. **Ensure `terraform.tfvars` contains the following:**

```
vpc_cidr_block = "10.0.0.0/16"
private_subnet_cidr_blocks = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
public_subnet_cidr_blocks = ["10.0.4.0/24", "10.0.5.0/24", "10.0.6.0/24"]
```

2. **Initialize and apply Terraform**:

```
terraform init
terraform apply --auto-approve
```

3. **Verify the EKS cluster in AWS Console**.

Step 2: Create a Namespace in EKS cluster

Configure Environment

1. **Update the `kubeconfig` file to allow `kubectl` to connect to the EKS cluster:**

   ```
   aws eks update-kubeconfig --region us-east-1 --name myapp-eks-cluster --kub
   econfig /mnt/c/Users/eduar/devops_projects2/08-ansible/terraform-learn/kubec
   onfig_myapp-eks-cluster
   Added new context arn:aws:eks:us-east-1:038462748802:cluster/myapp-eks-clu
   ster to /mnt/c/Users/eduar/devops_projects2/08-ansible/terraform-learn/kubeco
   nfig_myapp-eks-cluster
   ```

   This command generates or updates the kubeconfig file with the necessary authentication details for AWS EKS.

Install Required Dependencies

The `kubernetes.core.k8s` module requires the following dependencies:

- Python >= 3.9
- Kubernetes >= 24.2.0
- PyYAML >= 3.11
- jsonpatch

Reference:
https://docs.ansible.com/ansible/latest/collections/kubernetes/core/k8s_module.html#id2

To check if these dependencies are installed:

- `python3 -c "import kubernetes"`

- `python3 -c "import yaml"`
- `python3 -c "import jsonpatch"`

If you encounter errors such as:

```
Traceback (most recent call last):
  File "<string>", line 1, in <module>
    import kubernetes
ModuleNotFoundError: No module named 'kubernetes'
/mnt/c/Users/eduar/devops_projects2/08-ansible
Traceback (most recent call last):
  File "<string>", line 1, in <module>
    import YAML
ModuleNotFoundError: No module named 'YAML'
/mnt/c/Users/eduar/devops_projects2/08-ansible/terraform-learn
Traceback (most recent call last):
  File "<string>", line 1, in <module>
    import jsonpatch
ModuleNotFoundError: No module named 'jsonpatch'
/mnt/c/Users/eduar/devops_projects2/08-ansible
```

It means the packages need to be installed.

To install them:

`pip3 install kubernetes --user`

`pip3 install pyyaml --user`

`pip3 install jsonpatch --user`

Step 3: Create and Deploy Namespace after deploy the nginx application within the new K8s namespace using Ansible.

1. **Modify Ansible configuration (** `ansible.cfg` **)** `inventory = inventory_aws_ec2.yaml` to `inventory = hosts` so its doesnt run plugins

```
[defaults]
host_key_checking = False
```

```
inventory = hosts
# inventory = inventory_aws_ec2.yaml

interpreter_python = /usr/bin/python3.9

enable_plugins = aws_ec2

remote_user = ec2-user
private_key_file = /home/eb/.ssh/id_rsa
```

2. **Create an Ansible playbook** `deploy-to-k8s.yaml` **:**

```
---
- name: Deploy app in new namespace
  hosts: localhost
  tasks:
  - name: Create a k8s namespace
    kubernetes.core.k8s:
      name: my-app
      api_version: v1
      kind: namespace
      state: present
  - name: Deploy nginx app
    kubernetes.core.k8s:
      src: /mnt/c/Users/eduar/devops_projects2/08-ansible/nginx-config.yaml
      state: present
      namespace: my-app
```

3. **Execute the Ansible playbook:**

   - `export K8S_AUTH_KUBECONFIG=/mnt/c/Users/eduar/devops_projects2/08-ansible/terraform-learn/kubeconfig_myapp-eks-cluster`
   - `ansible-playbook deploy-to-k8s.yaml`

4. **Verify the namespace was created:**

   - In the terminal: `export KUBECONFIG=/mnt/c/Users/eduar/devops_projects2/08-ansible/terraform-learn/kubeconfig_myapp-eks-cluster`

- `kubectl get ns`

  Ensure `my-app` appears in the namespace list.

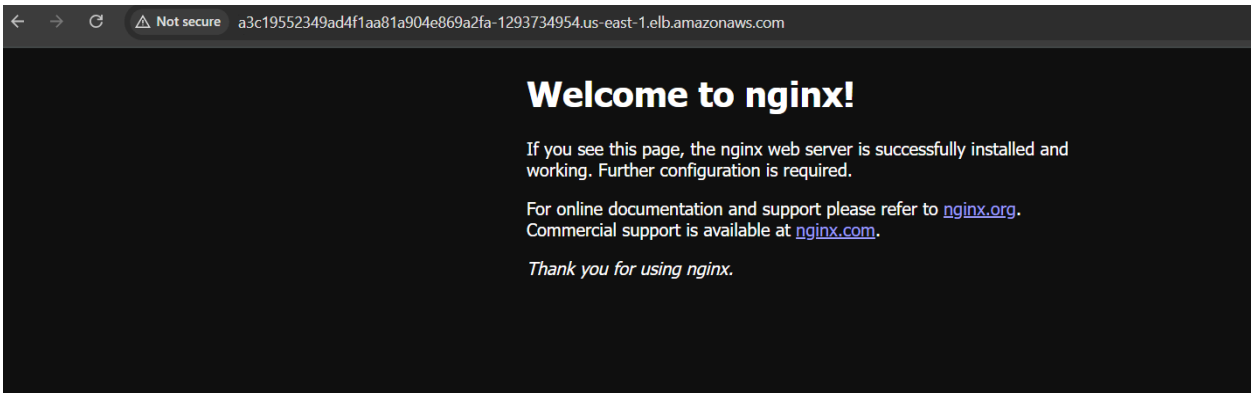
5. **Verify the application deployment:**

- `kubectl get pod -n my-app`
- `kubectl get svc -n my-app`

Example output:

```
 eb@DESKTOP-C8NETJI  ~  kubectl get pod -n my-app
NAME                    READY   STATUS    RESTARTS   AGE
nginx-55f598f8d-vjhvd   1/1     Running   0          15s
 eb@DESKTOP-C8NETJI  ~  kubectl get svc -n my-app
NAME    TYPE           CLUSTER-IP      EXTERNAL-IP                                                               PORT(S)        AGE
nginx   LoadBalancer   172.20.16.208   a3c19552349ad4f1aa81a904e869a2fa-1293734954.us-east-1.elb.amazonaws.com   80:31590/TCP   21s
```

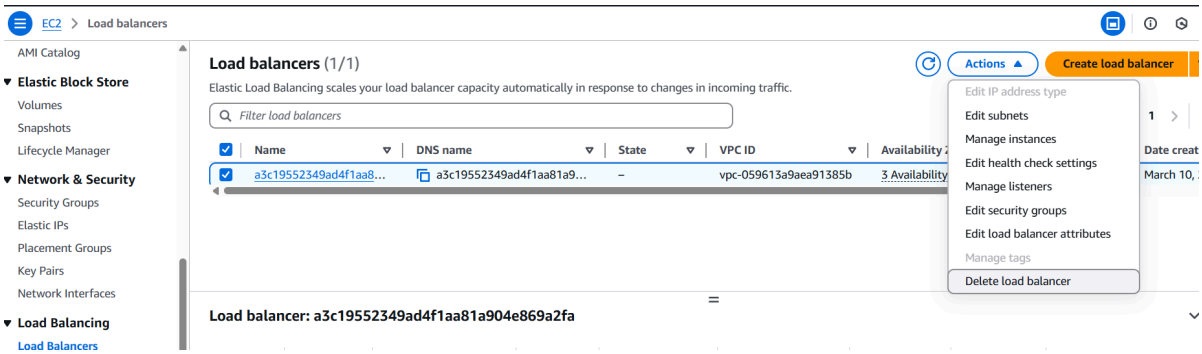
6. **Copy the LoadBalancer's public DNS and paste it into a browser to access the application.**



Step 4: Cleanup

- `terraform destroy --auto-approve`

  Note: If you encounter that subnets can’t be deleted by terraform, manually delete the Load Balancers and try again.



Understanding the Difference Between `K8S_AUTH_KUBECONFIG` and `KUBECONFIG`

1. `K8S_AUTH_KUBECONFIG` (Ansible-Specific Variable)

- `K8S_AUTH_KUBECONFIG` is used **only by Ansible's** `kubernetes.core.k8s` **module** to specify the kubeconfig path.
- When you run: `export K8S_AUTH_KUBECONFIG=/mnt/c/Users/eduar/devops_projects2/08-ansible/terraform-learn/kubeconfig_myapp-eks-cluster` it tells **Ansible** where to find the Kubernetes config file, but **it does not affect** `kubectl` **or other Kubernetes tools.**
- If you don't set `K8S_AUTH_KUBECONFIG`, you must provide the `kubeconfig` path explicitly in your playbook:

  `kubeconfig: /mnt/c/Users/eduar/devops_projects2/08-ansible/terraform-learn/kubeconfig_myapp-eks-cluster`

2. `KUBECONFIG` (Standard Kubernetes Variable)

- `KUBECONFIG` is the **default environment variable** that `kubectl` and most Kubernetes tools use to locate the configuration file.
- When you run `export KUBECONFIG=/mnt/c/Users/eduar/devops_projects2/08-ansible/terraform-learn/kubeconfig_myapp-eks-cluster` it tells `kubectl`, Helm, and other Kubernetes CLI tools where to find the config file.
- Example usage:
  - `kubectl get nodes`