Documentation for Terraform EKS Project Deployment

Table of Contents

1. Prerequisites

Ensure the following are available and properly configured:

- Access to an AWS account with necessary permissions to create EKS clusters, VPCs, and other resources.
- AWS CLI is installed and configured with credentials via aws configure.
- A code editor, such as VS Code, is installed.

2. Installing Necessary Tools

Install the following tools on your system:

1. **Terraform**
   - Download from Terraform Official Website.
   - Install and verify: terraform --version
2. **kubectl**
   - Follow the instructions on the Kubernetes Documentation.
   - Verify installation: kubectl version --client

3. **AWS CLI**
    - Verify installation: aws --version
4. **jq (optional)**
    - Install for JSON parsing: brew install jq
5. **Git**
    - Download from Git Official Website.
    - Verify installation: git --version

3. Configuring IAM Roles and Policies

IAM roles are essential for EKS clusters to operate properly. This project requires:

1. **EKS Node Group Role**

Attach the following policies to the role:

- AmazonEKSWorkerNodePolicy
- AmazonEKS_CNI_Policy
- AmazonEC2ContainerRegistryReadOnly

Terraform Implementation:

Define the role and policies in the iam.tf file:

```
resource "aws_iam_role" "eks_node_group_role" {
  name = "eks-node-group-role"
  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Effect = "Allow"
        Principal = { Service = "ec2.amazonaws.com" }
        Action = "sts:AssumeRole"
      }
    ]
  })
}
resource "aws_iam_role_policy_attachment" "eks_worker_node_policy" {
  role       = aws_iam_role.eks_node_group_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
}
```

Manual Configuration:

If creating manually via AWS Console:

1. Go to **IAM** > **Roles** > **Create Role**.

2. Select **EC2** as the trusted entity and attach the policies listed above.

3. Copy the **Role ARN** and update the variables.tf file:

```
variable "node_group_role_arn" {
  description = "IAM role ARN for the managed node group"
  default     = "arn:aws:iam::<account-id>:role/eks-node-group-role"
}
```

4. Project Directory Structure

```
terraform-eks-project/
├── main.tf
├── variables.tf
├── outputs.tf
├── iam.tf
├── modules/
│   ├── vpc/
│   │   ├── main.tf
│   │   ├── variables.tf
│   │   ├── outputs.tf
│   ├── eks/
│   │   ├── main.tf
│   │   ├── variables.tf
│   │   ├── outputs.tf
├── environments/
│   ├── dev.tfvars
│   ├── prod.tfvars
│   ├── staging.tfvars
```

5. Preparing Environment Variables

Edit the .tfvars files under environments/ for environment-specific configurations.

Example (dev.tfvars**):**

```
environment     = "dev"
cidr_block      = "10.0.0.0/16"
subnet_count    = 2
instance_type   = "t3.medium"
project_name    = "terraform-eks-project"
```

6. Manual AWS Resource Retrieval and Placement

Retrieve resource IDs and manually update the corresponding Terraform files:

1. **Security Group IDs:**

   - Retrieve using AWS CLI: `aws ec2 describe-security-groups --query 'SecurityGroups[*].[GroupId, GroupName]'`
   - Update in modules/eks/variables.tf:

```
variable "security_groups" {
  description = "List of security groups for the EKS node groups"
  default     = ["sg-XXXXXX", "sg-YYYYYY"]
```

```
}
```

2. **VPC ID:**

- Retrieve using AWS CLI: `aws ec2 describe-vpcs --query 'Vpcs[*].[VpcId, Tags]'`

- Update in modules/eks/variables.tf:

```
variable "vpc_id" {
  description = "ID of the VPC"
  default     = "vpc-XXXXXX"
}
```

3. **Subnet IDs:**

    - Retrieve using AWS CLI:

```
aws ec2 describe-subnets --query 'Subnets[*].[SubnetId, Tags]'
```

    - Update in modules/eks/variables.tf:

```
variable "subnet_ids" {
  description = "List of subnet IDs"
  default     = ["subnet-XXXXXX", "subnet-YYYYYY"]
}
```

7. Deployment Instructions

1. Clone the repository:

```
git clone https://github.com/ParkerPerry/terraform-eks-project.git
cd terraform-eks-project
```

2. Initialize Terraform:

```
terraform init
```

3. Plan and Apply:

```
terraform plan -var-file=environments/dev.tfvars
terraform apply -var-file=environments/dev.tfvars
```

8. Verifying the Deployment

Verify resources in the AWS Console:

- **EKS Cluster:** Check its status.
- **Node Groups:** Confirm instances are active.

Run:

```
aws eks describe-cluster --name <cluster-name> --query 'cluster.status'
kubectl get nodes
```

9. Load Balancer Access

Expose the application using a LoadBalancer:

```
kubectl expose deployment nginx --type=LoadBalancer --port=80
kubectl get service nginx --output=jsonpath='{.status.loadBalancer.ingress[0].hostname}'
```

10. Troubleshooting Common Issues

- **Ingress Rules:** Ensure port 80 is open for Security Groups.
- **IAM Role Permissions:** Verify attached policies.
- **EKS Cluster Connection:** Ensure kubeconfig is updated:

```
aws eks update-kubeconfig --region <region> --name <eks-cluster-name>
```

11. Cleanup and Deletion

Destroy resources:

```
terraform destroy -var-file=environments/dev.tfvars
```

12. Bash Commands Used Throughout the Project

Here's a consolidated list of the commands used for resource setup, validation, and troubleshooting:

AWS CLI Commands

1. **List Security Groups:**

```
aws ec2 describe-security-groups --query 'SecurityGroups[*].[GroupId, GroupName]' --output table
```

2. **Retrieve Subnet IDs:**

```
aws ec2 describe-subnets --query "Subnets[].SubnetId"
```

3. **Retrieve VPC IDs:**

```
aws ec2 describe-vpcs --query 'Vpcs[*].[VpcId, Tags]'
```

4. **Update** kubeconfig **for EKS Cluster Access:**

```
aws eks update-kubeconfig --region <region> --name <eks-cluster-name>
```

5. **Check EKS Cluster Status:**

```
aws eks describe-cluster --name <eks-cluster-name> --query "cluster.status"
```

Terraform Commands

1. **Initialize Terraform:** terraform init
2. **Plan Execution:** terraform plan -var-file=environments/<environment>.tfvars
3. **Apply Configuration:** terraform apply -var-file=environments/<environment>.tfvars
4. **Destroy Resources:** terraform destroy -var-file=environments/<environment>.tfvars

Kubectl Commands

1. **Check Node Status:** kubectl get nodes
2. **Deploy an Nginx Application:** kubectl create deployment nginx --image=nginx
3. **Expose the Nginx Deployment Using a LoadBalancer:**

```
kubectl expose deployment nginx --type=LoadBalancer --port=80
```

4. **Get Load Balancer DNS:**

```
kubectl get service nginx --output=jsonpath='{.status.loadBalancer.ingress[0].hostname}'
```

Git Commands

1. **Clone Repository:**

```
git clone https://github.com/ParkerPerry/terraform-eks-project.git
```

2. **Filter Out Large Files:**

```
git filter-repo --path .terraform/ --path .terraform.lock.hcl --invert-paths
```

3. **Commit and Push Changes:**

```
git add .
git commit -m "Initial commit for Terraform EKS project"
git push -u origin main
```

13. Additional Notes

Environment-Specific CIDR Blocks

Each environment (dev, staging, prod) has its own CIDR block defined in the .tfvars files. Ensure these are unique to avoid conflicts:

- **Dev Environment:** 10.0.0.0/16
- **Staging Environment:** 10.1.0.0/16
- **Prod Environment:** 10.2.0.0/16

Resource Naming Conventions

To differentiate resources across environments:

- Use the environment variable to append the environment name to resource names.

Example:

```
cluster_name = "${var.cluster_name}-${var.environment}"
name = "${var.name}-${var.environment}"
```

Scaling Adjustments

Modify the following parameters in the .tfvars files to adjust scaling configurations:

- **Instance Type (**instance_type**):** Change based on performance requirements.
- **Node Group Desired Size:** Update in modules/eks/main.tf under desired_size, max_size, and min_size.

Conclusion:

This documentation is designed to be a comprehensive guide for deploying the Terraform EKS project.

1. Set up the environment.
2. Retrieve and configure the necessary AWS resource identifiers.

3. Deploy and verify an EKS cluster and associated resources.

4. Expose and access applications deployed on the cluster.

If any issues arise during deployment or testing, refer to the **Troubleshooting** section and verify all steps were completed accurately.