# *kubenetes_setup*

Below is a Terraform script to set up a basic Kubernetes cluster with one master node and two worker nodes on AWS. This setup will include a VPC, subnets, an internet gateway, security groups, and the EC2 instances for the master and worker nodes.

The error "No valid credential sources found" indicates that Terraform is unable to find AWS credentials to authenticate and interact with your AWS account. Here are a few steps to resolve this issue:

## Step 1: Set Up AWS Credentials

Terraform requires AWS credentials to manage AWS resources. You can provide these credentials in several ways:

### 1. **AWS CLI Configuration**

If you have the AWS CLI installed and configured, Terraform can use the credentials stored in the AWS CLI configuration.

• To install the AWS CLI, follow the instructions here.

• Configure the AWS CLI with your credentials:shCopy code

```
aws configure
```

This command will prompt you to enter your AWS Access Key ID, Secret Access Key, region, and output format.

### 2. **Environment Variables**

You can set environment variables that Terraform will use for AWS credentials.

shCopy code

```
export AWS_ACCESS_KEY_ID="your-access-key-id"
export AWS_SECRET_ACCESS_KEY="your-secret-access-key"
export AWS_DEFAULT_REGION="us-west-2"
```

## Directory Structure

Create a directory for your Terraform project and navigate into it. Inside this directory, create the following files:

• `main.tf`
• `variables.tf`
• `outputs.tf`
• `provider.tf`

## Commands to Deploy

1. Initialize the Terraform working directory.
shCopy code

```
terraform init
```

2. Review the plan.
shCopy code

```
terraform plan
```

3. Apply the plan to create the resources.
shCopy code

```
terraform apply
```

This script sets up a basic Kubernetes cluster with one master node and two worker nodes. You can modify this script to suit your specific requirements by adjusting instance types, adding more worker nodes, or configuring additional settings. After deploying the instances, you will need to manually install and configure Kubernetes on each node to complete the cluster setup.

## *provider.tf*

```
provider "aws" {
  region = "us-west-2"
}
```

## *variables.tf*

```
variable "region" {
  default = "us-west-2"
}

variable "instance_type" {
  default = "t2.medium"
}

variable "ami_id" {
  description = "AMI ID for the EC2 instances"
  default     = "ami-0c55b159cbfafe1f0"
}

variable "vpc_cidr" {
  default = "10.0.0.0/16"
}

variable "subnet_cidr" {
  default = "10.0.1.0/24"
}

variable "key_name" {
  description = "Name of the key pair"
  default     = "my-key-pair"
}
```

## *main.tf*

```
resource "aws_vpc" "main" {
  cidr_block = var.vpc_cidr

  tags = {
    Name = "k8s-vpc"
  }
}

resource "aws_subnet" "main" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = var.subnet_cidr
  availability_zone = "us-west-2a"

  tags = {
    Name = "k8s-subnet"
  }
}

resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "k8s-igw"
  }
}

resource "aws_route_table" "main" {
  vpc_id = aws_vpc.main.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.main.id
  }

  tags = {
    Name = "k8s-route-table"
  }
}

resource "aws_route_table_association" "main" {
  subnet_id      = aws_subnet.main.id
  route_table_id = aws_route_table.main.id
}

resource "aws_security_group" "allow_ssh" {
  vpc_id = aws_vpc.main.id
```

```
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "allow_ssh"
  }
}

resource "aws_instance" "master" {
  ami           = var.ami_id
  instance_type = var.instance_type
  subnet_id     = aws_subnet.main.id
  key_name      = var.key_name

  vpc_security_group_ids = [aws_security_group.allow_ssh.id]

  tags = {
    Name = "k8s-master"
  }
}

resource "aws_instance" "worker1" {
  ami           = var.ami_id
  instance_type = var.instance_type
  subnet_id     = aws_subnet.main.id
  key_name      = var.key_name

  vpc_security_group_ids = [aws_security_group.allow_ssh.id]

  tags = {
    Name = "k8s-worker1"
  }
}

resource "aws_instance" "worker2" {
  ami           = var.ami_id
  instance_type = var.instance_type
  subnet_id     = aws_subnet.main.id
  key_name      = var.key_name
```

```
  vpc_security_group_ids = [aws_security_group.allow_ssh.id]

  tags = {
    Name = "k8s-worker2"
  }
}
```

## *outputs.tf*

```
output "master_instance_id" {
  value = aws_instance.master.id
}

output "master_instance_public_ip" {
  value = aws_instance.master.public_ip
}

output "worker1_instance_id" {
  value = aws_instance.worker1.id
}

output "worker1_instance_public_ip" {
  value = aws_instance.worker1.public_ip
}

output "worker2_instance_id" {
  value = aws_instance.worker2.id
}

output "worker2_instance_public_ip" {
  value = aws_instance.worker2.public_ip
}

output "vpc_id" {
  value = aws_vpc.main.id
}
```