SCRIPTING

Purpose

This script automates the creation and configuration of AWS infrastructure, including a Virtual Private Cloud (VPC), subnets, an internet gateway, a NAT gateway, security groups, and an EC2 instance. The goal is to streamline the setup of a networked environment in AWS.

Why Use Scripting?

- **Automation**: Scripts automate repetitive tasks, reducing the chance of errors and saving time.
- **Consistency**: Running a script ensures that tasks are performed in the exact same way every time.
- **Efficiency**: Large-scale configurations, like deploying infrastructure, become much faster with scripts.
- **Reproducibility**: Scripts make it easier to reproduce a setup on different machines or environments.

Advantages of Using Scripts

- **Scalability**: Easily manage resources, such as setting up multiple servers or configuring networks.
- **Simplicity**: Simplifies complex tasks by breaking them into a series of automated commands.
- **Version Control**: Scripts can be tracked in version control systems like Git, making it easy to review changes.
- **Documentation**: Well-written scripts with comments and descriptive commands act as documentation for your setup.

Disadvantages of Using Scripts

- **Debugging Complexity**: Errors in scripts can be harder to debug, especially in long or complex scripts.

- **Maintenance**: Scripts need to be updated if the underlying system or commands change.
- **Learning Curve**: It takes time to learn scripting languages like Bash or PowerShell.
- **Risk of Mistakes**: If a script is not written carefully, it could misconfigure resources or cause data loss.

Best Practices for Writing Scripts

1. **Add Comments**: Use comments (# Comment text) to explain what each section of the script does.
2. **Use Variables**: Store values in variables to make the script easier to update and read.
3. **Error Handling**: Check the outcome of commands and handle errors gracefully.
4. **Keep It Modular**: Break down large scripts into functions or smaller scripts for easier maintenance.
5. **Security**: Avoid hardcoding sensitive information, like passwords or keys, directly in the script.

Common Uses of Scripts

- **Infrastructure Setup**: Automating the creation and configuration of servers, databases, and networks.
- **Data Processing**: Automating data transformation or analysis tasks.
- **Deployment**: Automating the deployment of applications or updates.
- **Monitoring and Alerts**: Scripts can be used to check system status and send alerts if something goes wrong.

END-END SCRIPTING

CREATE a ENV VARIABLE.SH file (env_variables.sh)

```
#!/bin/bash


export VPC_CIDR="10.0.0.0/16"

export  PUBLIC_SUBNET_CIDR_1="10.0.1.0/24"

export PUBLIC_SUBNET_CIDR_2="10.0.2.0/24"

export PRIVATE_SUBNET_CIDR="10.0.3.0/24"

export AVAILABILITY_ZONE_1="ap-southeast-1a"

export AVAILABILITY_ZONE_2="ap-southeast-1b"

export INSTANCE_TYPE="t2.micro"

export AMI_ID="ami-04a43fe766897dd2e"  # Replace with your AMI ID

export KEY_NAME="script"  # Replace with your key pair name

export LOAD_BALANCER_NAME="script-load-balancer"

export TARGET_GROUP_NAME="script-target-group"

export LAUNCH_TEMPLATE_NAME="script-launch-template"

export AUTO_SCALING_GROUP_NAME="script-auto-scaling-group"

export SECURITY_GROUP_NAME="script-security-group"

export USER_DATA_FILE="user_data.sh"  # Specify your User Data script
```

CREATE a scripts..sh file and add the destination of the variables file

```
#!/bin/bash


source ./env_variables.sh


# Script starts
echo "Creating VPC..."
VPC_ID=$(aws ec2 create-vpc --cidr-block $VPC_CIDR --query 'Vpc.VpcId' --output text)
echo "VPC Created: $VPC_ID"


echo "Creating Public Subnet..."
PUBLIC_SUBNET_ID=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-block
$PUBLIC_SUBNET_CIDR_1 --availability-zone $AVAILABILITY_ZONE_1 --query
'Subnet.SubnetId' --output text)
echo "Public Subnet Created: $PUBLIC_SUBNET_ID"


echo "Creating Additional Public Subnet..."
PUBLIC_SUBNET_ID_2=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-block
$PUBLIC_SUBNET_CIDR_2 --availability-zone $AVAILABILITY_ZONE_2 --query
'Subnet.SubnetId' --output text)
echo "Additional Public Subnet Created: $PUBLIC_SUBNET_ID_2"


echo "Creating Private Subnet..."
PRIVATE_SUBNET_ID=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-block
$PRIVATE_SUBNET_CIDR --availability-zone $AVAILABILITY_ZONE_1 --query
'Subnet.SubnetId' --output text)
echo "Private Subnet Created: $PRIVATE_SUBNET_ID"


echo "Creating Internet Gateway..."
```

```
IGW_ID=$(aws ec2 create-internet-gateway --query 'InternetGateway.InternetGatewayId' --
output text)

aws ec2 attach-internet-gateway --vpc-id $VPC_ID --internet-gateway-id $IGW_ID

echo "Internet Gateway Created and Attached: $IGW_ID"


echo "Creating Public Route Table..."

ROUTE_TABLE_ID=$(aws ec2 create-route-table --vpc-id $VPC_ID --query
'RouteTable.RouteTableId' --output text)

aws ec2 create-route --route-table-id $ROUTE_TABLE_ID --destination-cidr-block 0.0.0.0/0 --
gateway-id $IGW_ID

aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
$PUBLIC_SUBNET_ID

aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
$PUBLIC_SUBNET_ID_2

echo "Public Route Table Created and Associated: $ROUTE_TABLE_ID"


echo "Allocating Elastic IP for NAT Gateway..."

EIP_ALLOC_ID=$(aws ec2 allocate-address --domain vpc --query 'AllocationId' --output text)


echo "Creating NAT Gateway..."

NAT_GW_ID=$(aws ec2 create-nat-gateway --subnet-id $PUBLIC_SUBNET_ID --allocation-id
$EIP_ALLOC_ID --query 'NatGateway.NatGatewayId' --output text)

echo "NAT Gateway Created: $NAT_GW_ID"


echo "Waiting for NAT Gateway to become available..."

aws ec2 wait nat-gateway-available --nat-gateway-ids $NAT_GW_ID

echo "NAT Gateway is now available."


echo "Creating Private Route Table..."
```

```
PRIVATE_ROUTE_TABLE_ID=$(aws ec2 create-route-table --vpc-id $VPC_ID --query
'RouteTable.RouteTableId' --output text)

aws ec2 create-route --route-table-id $PRIVATE_ROUTE_TABLE_ID --destination-cidr-block
0.0.0.0/0 --nat-gateway-id $NAT_GW_ID

aws ec2 associate-route-table --route-table-id $PRIVATE_ROUTE_TABLE_ID --subnet-id
$PRIVATE_SUBNET_ID

echo "Private Route Table Created and Associated: $PRIVATE_ROUTE_TABLE_ID"


echo "Creating Security Group..."

SECURITY_GROUP_ID=$(aws ec2 create-security-group --group-name
$SECURITY_GROUP_NAME --description "Security group for script" --vpc-id $VPC_ID --
query 'GroupId' --output text)

aws ec2 authorize-security-group-ingress --group-id $SECURITY_GROUP_ID --protocol tcp --
port 22 --cidr 0.0.0.0/0

aws ec2 authorize-security-group-ingress --group-id $SECURITY_GROUP_ID --protocol tcp --
port 80 --cidr 0.0.0.0/0

echo "Security Group Created: $SECURITY_GROUP_ID"


echo "Creating Load Balancer..."

LOAD_BALANCER_ARN=$(aws elbv2 create-load-balancer --name
$LOAD_BALANCER_NAME --subnets $PUBLIC_SUBNET_ID $PUBLIC_SUBNET_ID_2 --
security-groups $SECURITY_GROUP_ID --query 'LoadBalancers[0].LoadBalancerArn' --
output text)

echo "Load Balancer Created: $LOAD_BALANCER_ARN"


echo "Creating Target Group..."

TARGET_GROUP_ARN=$(aws elbv2 create-target-group --name $TARGET_GROUP_NAME
--protocol HTTP --port 80 --vpc-id $VPC_ID --query 'TargetGroups[0].TargetGroupArn' --output
text)

echo "Target Group Created: $TARGET_GROUP_ARN"
echo "Creating Launch Template..."

LAUNCH_TEMPLATE_ID=$(aws ec2 create-launch-template --launch-template-name
$LAUNCH_TEMPLATE_NAME --version-description "v1" --launch-template-data "{
  \"ImageId\": \"$AMI_ID\",
  \"InstanceType\": \"$INSTANCE_TYPE\",
  \"KeyName\": \"$KEY_NAME\",
  \"SecurityGroupIds\": [\"$SECURITY_GROUP_ID\"],
  \"UserData\": \"$(base64 -w 0 $USER_DATA_FILE)\"
}" --query 'LaunchTemplate.LaunchTemplateId' --output text)
echo "Launch Template Created: $LAUNCH_TEMPLATE_ID"


echo "Creating Auto Scaling Group..."
aws autoscaling create-auto-scaling-group --auto-scaling-group-name
$AUTO_SCALING_GROUP_NAME --launch-template
"LaunchTemplateId=$LAUNCH_TEMPLATE_ID,Version=1" --min-size 1 --max-size 3 --
desired-capacity 2 --vpc-zone-identifier "$PUBLIC_SUBNET_ID,$PUBLIC_SUBNET_ID_2" -
-target-group-arns $TARGET_GROUP_ARN
echo "Auto Scaling Group Created and Attached to Target Group:
$AUTO_SCALING_GROUP_NAME"


echo "Attaching Target Group to Load Balancer..."
aws elbv2 create-listener --load-balancer-arn $LOAD_BALANCER_ARN --protocol HTTP --
port 80 --default-actions Type=forward,TargetGroupArn=$TARGET_GROUP_ARN
echo "Target Group Attached to Load Balancer"


echo "Infrastructure Creation Complete!"
echo "VPC ID: $VPC_ID"
echo "Public Subnet ID 1: $PUBLIC_SUBNET_ID"
echo "Public Subnet ID 2: $PUBLIC_SUBNET_ID_2"
echo "Private Subnet ID: $PRIVATE_SUBNET_ID"

echo "Internet Gateway ID: $IGW_ID"

echo "NAT Gateway ID: $NAT_GW_ID"

echo "Security Group ID: $SECURITY_GROUP_ID"

echo "Load Balancer ARN: $LOAD_BALANCER_ARN"

echo "Target Group ARN: $TARGET_GROUP_ARN"

echo "Launch Template ID: $LAUNCH_TEMPLATE_ID"

echo "Auto Scaling Group Name: $AUTO_SCALING_GROUP_NAME"
```