HandsOn Demo

1) Clone the github repo https://github.com/16sa/3-tier-app-deployment-HandsOn-with-terraform-modules-on-AWS.git

2) Create the AWS account - https://aws.amazon.com/console/

3) Install Docker and terraform on windows

https://docs.docker.com/desktop/install/windows-install/

https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

For Docker Desktop, make sure to install WSL 2 Linux subsystem on your machine and Enable Hyper-V and Containers in Windows Features before installation

Make sure to start Docker Desktop before proceeding with next steps

4) Install AWS CLI on you Machine

5) Create an IAM user from AWS console and add access key ID to that use, Make sure to save the Access key ID and Access secret somewhere.

6) From powershell or cmd tape "aws configure" Command with the following input:

AWS Access Key ID [None]: USER\_ACCESS\_KEY\_ID

AWS Secret Access Key [None]: USER\_SECRET\_ACCESS\_KEY

Default region name [None]: YOUR\_AWS\_REGION

Default output format [None]: json

You can verify your AWS CLI configuration by running this command "aws sts get-caller-identity"

7) Install jq Executabled on you windows machine

8) Grant the IAM user the necessary ECR permissions: From AWS console management, Navigate to the IAM console Users

and create an IAM Policy for your created user with the following content, Update the Region according to your Location

you can name the policy ECR-ha-app-Permissions-eu-west-3 for example

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "AllowECRBasic",

"Effect": "Allow",

"Action": [

"ecr:GetAuthorizationToken",

"ecr:DescribeRepositories",

"ecr:CreateRepository",

"ecr:BatchCheckLayerAvailability",

"ecr:GetDownloadUrlForLayer",

"ecr:GetRepositoryPolicy",

"ecr:ListImages",

"ecr:BatchGetImage",

"ecr:DeleteRepository"

],

"Resource": "arn:aws:ecr:eu-west-3:USER\_ACCOUNT\_ID:repository/ha-app-\*"

},

{

"Sid": "AllowPushPull",

"Effect": "Allow",

"Action": [

"ecr:InitiateLayerUpload",

"ecr:UploadLayerPart",

"ecr:CompleteLayerUpload",

"ecr:PutImage"

],

"Resource": "arn:aws:ecr:eu-west-3:USER\_ACCOUNT\_ID:repository/ha-app-\*"

}

]

}

9) Execute the linux command to give permission

chmod +x setup-ecrs.sh

10) Run this on terminal to create the ECR repo and to create images in local

and send those to ECR; Make sure to update the AWS region on the script

./setup-ecrs.sh

In summary, this script automates the process of:

Logging into your AWS ECR registry.

Creating two ECR repositories (ha-app-application-tier and ha-app-presentation-tier) if they don't already exist.

Navigating to the application-tier directory.

Building a Docker image for the application tier.

Tagging the application tier image with the correct ECR repository URI.

Pushing the application tier image to ECR.

Navigating to the presentation-tier directory.

Building a Docker image for the presentation tier.

Tagging the presentation tier image with the correct ECR repository URI.

Pushing the presentation tier image to ECR.

11) Before proceding with terraform session you need to Create a TerraformRole from AWS console Management and Attach the necessary policies:

==> TerraformAppInfraPolicy:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"elasticloadbalancing:CreateLoadBalancer",

"elasticloadbalancing:DescribeLoadBalancers",

"elasticloadbalancing:DeleteLoadBalancer",

"elasticloadbalancing:CreateTargetGroup",

"elasticloadbalancing:DescribeTargetGroups",

"elasticloadbalancing:DeleteTargetGroup",

"elasticloadbalancing:RegisterTargets",

"elasticloadbalancing:ModifyLoadBalancerAttributes",

"elasticloadbalancing:ModifyTargetGroupAttributes",

"elasticloadbalancing:AddTags",

"elasticloadbalancing:DescribeTargetGroupAttributes",

"elasticloadbalancing:SetSecurityGroups",

"elasticloadbalancing:DescribeLoadBalancerAttributes",

"elasticloadbalancing:CreateListener",

"elasticloadbalancing:DescribeListeners",

"elasticloadbalancing:DeleteListener",

"autoscaling:CreateAutoScalingGroup",

"autoscaling:DescribeAutoScalingGroups",

"autoscaling:DeleteAutoScalingGroup",

"autoscaling:CreateLaunchConfiguration",

"autoscaling:DeleteLaunchConfiguration",

"autoscaling:UpdateAutoScalingGroup",

"autoscaling:AttachLoadBalancerTargetGroups",

"autoscaling:DetachLoadBalancerTargetGroups"

],

"Resource": "\*"

}

]

}

==> TerraformEC2Describe:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"ec2:DescribeImages",

"ec2:DescribeAvailabilityZones"

],

"Resource": "\*"

}

]

}

==> TerraformEC2Policy:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"ec2:CreateVpc",

"ec2:DescribeVpcs",

"ec2:DeleteVpc",

"ec2:CreateSubnet",

"ec2:DescribeSubnets",

"ec2:DeleteSubnet",

"ec2:CreateSecurityGroup",

"ec2:DescribeSecurityGroups",

"ec2:DeleteSecurityGroup",

"ec2:AuthorizeSecurityGroupIngress",

"ec2:AuthorizeSecurityGroupEgress",

"ec2:CreateTags",

"ec2:DescribeImages",

"ec2:DescribeAvailabilityZones",

"ec2:ModifySubnetAttribute",

"ec2:RevokeSecurityGroupEgress",

"ec2:DescribeVpcAttribute",

"ec2:DescribeInternetGateways",

"ec2:DescribeRouteTables",

"ec2:DescribeNetworkInterfaces",

"ec2:AllocateAddress",

"ec2:AssociateRouteTable",

"ec2:DescribeAddresses",

"ec2:DescribeAccountAttributes",

"ec2:ReleaseAddress",

"ec2:CreateLaunchTemplate",

"ec2:CreateNatGateway",

"ec2:DescribeLaunchTemplates",

"ec2:DescribeNatGateways",

"ec2:DescribeLaunchTemplateVersions",

"ec2:DeleteLaunchTemplate",

"ec2:DeleteNatGateway",

"ec2:CreateRouteTable",

"ec2:RunInstances",

"ec2:CreateRoute",

"ec2:DeleteRouteTable",

"ec2:DisassociateRouteTable",

"ec2:DetachInternetGateway",

"ec2:DeleteInternetGateway"

],

"Resource": "\*"

}

]

}

==> TerraformIAMPolicy:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"iam:CreateRole",

"iam:GetRole",

"iam:DeleteRole",

"iam:CreateInstanceProfile",

"iam:DeleteInstanceProfile",

"iam:AddRoleToInstanceProfile",

"iam:RemoveRoleFromInstanceProfile",

"iam:AttachRolePolicy",

"iam:DetachRolePolicy",

"iam:PassRole",

"iam:ListAttachedRolePolicies",

"iam:PutRolePolicy",

"iam:ListRolePolicies",

"iam:GetInstanceProfile",

"iam:GetRolePolicy",

"iam:DeleteRolePolicy",

"iam:CreateServiceLinkedRole",

"iam:ListInstanceProfilesForRole"

],

"Resource": "\*"

}

]

}

==> TerraformRDSPolicy:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"rds:CreateDBSubnetGroup",

"rds:DescribeDBSubnetGroups",

"rds:DeleteDBSubnetGroup",

"rds:CreateDBInstance",

"rds:DescribeDBInstances",

"rds:DeleteDBInstance",

"rds:ModifyDBInstance",

"rds:AddTagsToResource",

"rds:ListTagsForResource"

],

"Resource": "\*"

}

]

}

12) Ensure the User Has sts:AssumeRole Permission: Attach this AWSSTSServiceRoleAssume policy to user

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "sts:AssumeRole",

"Resource": "arn:aws:iam::USER\_ACCOUNT\_ID:role/TerraformRole"

}

]

}

13) We need to set trust relationship for the TerraformRole: This policy defines which entities are allowed to assume the role.

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"AWS": "USER\_ARN"

},

"Action": "sts:AssumeRole"

}

]

}

14) From powershell terminal run this command:

aws sts assume-role --role-arn "arn:aws:iam::USER\_ACCOUNT\_ID:role/TerraformRole"  --role-session-name "TerraformSession"

15) Once you have these temporary credentials, you need to set them as environment variables in your terminal session so that Terraform can use them

From Powershell terminal run this command:

Write-Host $env:AWS\_ACCESS\_KEY\_ID # Replace with your actual Access Key ID

Write-Host $env:AWS\_SECRET\_ACCESS\_KEY # Replace with your actual Secret Access Key

Write-Host $env:AWS\_SESSION\_TOKEN # Replace with your actual Session Token

16) Make sure to put the suitable AWS Region in your terraform.tfvars file before proceeding

17) Go to terraform folder -

terraform init

terraform plan

terraform apply

18) Hit the Front end load balancerfront-

end-lb-\*\*\*\*\*\*\*\*\*\*.AWS\_REGION.elb.amazonaws.com/

19) Delete the Architecture

terraform delete

20) Run this on terminal to delete the ECR repo

chmod +x destroy-ecrs.sh

./destroy-ecrs.sh