**Lab Steps**

Task 1: Sign in to AWS Management Console

1. Click on the button, and you will get redirected to AWS Console in a new browser tab.
2. On the AWS sign-in page,

* Leave the Account ID as default. Never edit/remove the 12 digit Account ID present in the AWS Console. otherwise, you cannot proceed with the lab.
* Now copy your **User Name** and **Password** in the Lab Console to the **IAM Username** and **Password** in AWS Console and click on the **Sign in** button.

1. Once Signed In to the AWS Management Console, Make the default AWS Region as **US East (N. Virginia) us-east-1.**

Task 2: Setup Visual Studio Code

1. Open the Visual Studio Code.
2. If you have already installed and using the Visual Studio code, open a new window.
3. A new window will open a new file and release notes page (only if you have installed or updated Visual Studio Code recently). Close the Release notes tab.
4. Open terminal by selecting View from the Menu bar and choose Terminal.
5. It may take up to 2 minutes to open the terminal window.
6. Once the terminal is ready, let us navigate to the Desktop.

cd Desktop

1. Create a new folder by running the below command:

mkdir task\_10093\_ebs

1. Change your present working directory to use the newly created folder by running the command:

cd task\_10093\_ebs

1. Get the location of the present working directory by running the below command:

pwd

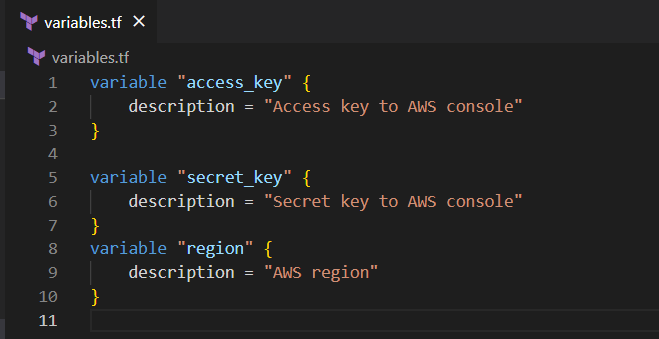
1. Note down the location, as you will open the same in the next steps.
2. Now click on the first icon Explorer present on the left sidebar.
3. Click on the button called Open folder and navigate to the location of folder **task\_10093\_ebs**
4. Visual Studio code is now ready to use.

Task 3: Create a variables file

1. To create a variables file, expand the folder **task\_10093\_ebs** and click on the **New File** icon to add the file.
2. Name the file as **variables.tf** and press **Enter** to save it.
3. **Note: Don't change the location of the new file, keep it default, i.e. inside the task\_10093\_ebs folder.**
4. Paste the below contents in **variables.tf** file.

|  |
| --- |
| variable "access\_key" {      description = "Access key to AWS console"  }  variable "secret\_key" {      description = "Secret key to AWS console"  }  variable "region" {      description = "AWS region"  } |

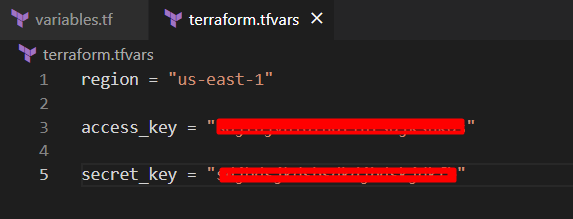
1. In the above content, you are declaring a variable called, access\_key, secret\_key, and region with a short description of all 3.
2. After pasting the above contents, save the file by pressing **ctrl + S**.



1. Now expand the folder **task\_10093\_ebs** and click on the New File icon to add the file.
2. Name the file as **terraform.tfvars** and press **Enter** to save it.
3. Paste the below content into the **terraform.tfvars** file.

|  |
| --- |
| region = "us-east-1"  access\_key = "<YOUR\_ACCESS\_KEY>"  secret\_key = "<YOUR\_SECRET\_KEY>" |

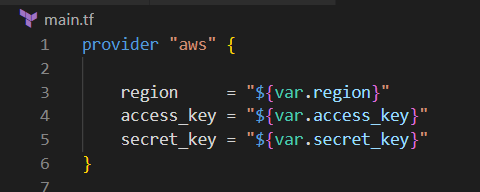
1. In the above code, you are defining the dynamic values of variables declared earlier.
2. Replace the values of access\_key and secret\_key by copying from the lab page.
3. After replacing the values of access\_key and secret\_key, save the file by pressing **Ctrl + S**.



Task 4: Create an Elastic BeanStalk Application in the main.tf file

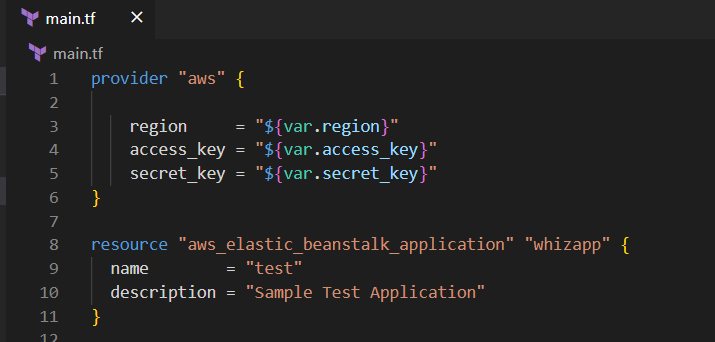
1. To create a main.tf file, expand the folder **task\_10093\_ebs** and click on the **New File** icon to add the file.
2. Name the file as **main.tf** and press Enter to save it.
3. Paste the below content into the **main.tf** file.

|  |
| --- |
| provider "aws" {      region     = "${var.region}"      access\_key = "${var.access\_key}"      secret\_key = "${var.secret\_key}"  } |



1. In the above code, you are defining the provider as **aws**.
2. Next, we want to tell Terraform to create a Elastic Beanstalk application
3. Paste the below content into the **main.tf** file after the provider.

|  |
| --- |
| resource "aws\_elastic\_beanstalk\_application" "whizapp" {    name        = "test"    description = "Sample Test Application"  } |



1. Let’s add another set of code after elastic beanstalk application creation. Paste the following code in the main.tf file. This code will create an instance profile for the EC2 Instance. The code includes creation of an IAM role, and attaching the requires IAM policies to the role.

|  |
| --- |
| resource "aws\_iam\_instance\_profile" "subject\_profile" {    name = "test\_role\_new"    role = aws\_iam\_role.role.name  }  resource "aws\_iam\_role" "role" {    name = "test\_role\_new"    path = "/"    assume\_role\_policy = <<EOF  {      "Version": "2012-10-17",      "Statement": [          {              "Action": "sts:AssumeRole",              "Principal": {                 "Service": "ec2.amazonaws.com"              },              "Effect": "Allow",              "Sid": ""          }      ]  }  EOF  }  resource "aws\_iam\_role\_policy\_attachment" "role-policy-attachment" {    for\_each = toset([      "arn:aws:iam::aws:policy/AWSElasticBeanstalkWebTier",      "arn:aws:iam::aws:policy/AWSElasticBeanstalkMulticontainerDocker",      "arn:aws:iam::aws:policy/AWSElasticBeanstalkWorkerTier",    ])    role = "${aws\_iam\_role.role.name}"    policy\_arn = each.value  } |



Task 5: Creating an Elastic environment in the main.tf file

1. To create an elastic environment for the application, paste the below code in the **main.tf**.

|  |
| --- |
| resource "aws\_elastic\_beanstalk\_environment" "whizenv" {    name                = "whizenvironment"    application         = aws\_elastic\_beanstalk\_application.whizapp.name    solution\_stack\_name = "64bit Amazon Linux 2 v3.4.1 running Corretto 17"  setting {      namespace = "aws:autoscaling:launchconfiguration"      name = "IamInstanceProfile"      value = "${aws\_iam\_instance\_profile.subject\_profile.name}"    }   setting {      namespace = "aws:elasticbeanstalk:environment:process:default"      name      = "MatcherHTTPCode"      value     = "200"    }   setting {      namespace = "aws:elasticbeanstalk:environment"      name      = "LoadBalancerType"      value     = "application"    }     setting {      namespace = "aws:autoscaling:launchconfiguration"      name      = "InstanceType"      value     = "t2.micro"    }     setting {      namespace = "aws:autoscaling:asg"      name      = "MinSize"      value     = 1    }    setting {      namespace = "aws:autoscaling:asg"      name      = "MaxSize"      value     = 2    }    setting {      namespace = "aws:elasticbeanstalk:healthreporting:system"      name      = "SystemType"      value     = "enhanced"    }  } |



1. In the above code, we have declared resource type as **aws\_elastic\_beanstalk\_environment** for the environment of the Elastic Beanstalk application.

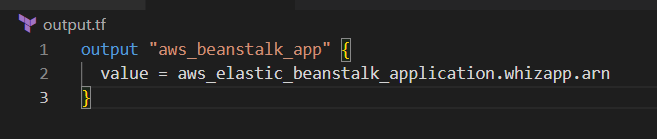
* **solution\_stack\_name** is the name of an environment that Elastic Beanstalk will set up on servers it manages. The solution stack here is **64bit Amazon Linux 2 v3.4.1 running Corretto 17** for **Java**.
* There are other additional settings added for the environment that are required for the environment to launch like Autoscaling Launch configuration , Load balancer , and the minimum and maximum size for the autoscaling group.

Task 6: Create an output file

In this task, you will create an **output.tf** file where you will add details of the provider and resources.

1. To create an **output.tf f**ile, expand the folder **task\_10093\_ebs** and click on the New File icon to add the file.
2. Name the file as **output.tf** and press Enter to save it.
3. Paste the below content into the **output.tf** file.

|  |
| --- |
| output "aws\_beanstalk\_app" {    value = aws\_elastic\_beanstalk\_application.whizapp.arn  } |



1. In the above code, we will extract the **arn** of the elastic beanstalk application.

Task 7: Confirm the installation of Terraform by checking the version

1. In the Visual Studio Code, open Terminal by selecting View from the Menu bar and choose Terminal.
2. If you are not in the newly created folder change your present working directory by running the below command.

cd task\_10093\_ebs

1. To confirm the installation of Terraform, run the below command to check the version:

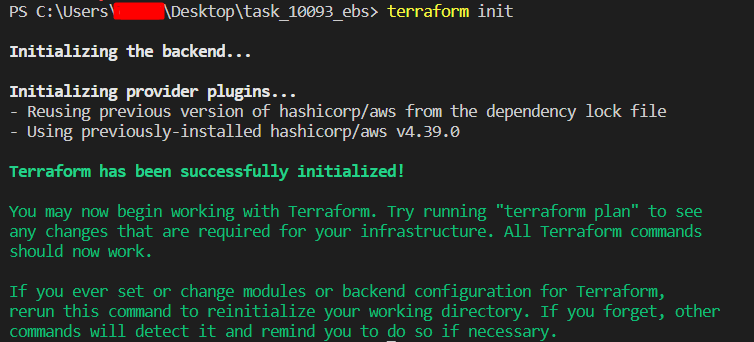
terraform version

1. If you are getting output as command not found: terraform, this means that terraform is not installed on your system, To install terraform follow the official guide link provided in the Prerequisite section above.

Task 8: Applying terraform configurations

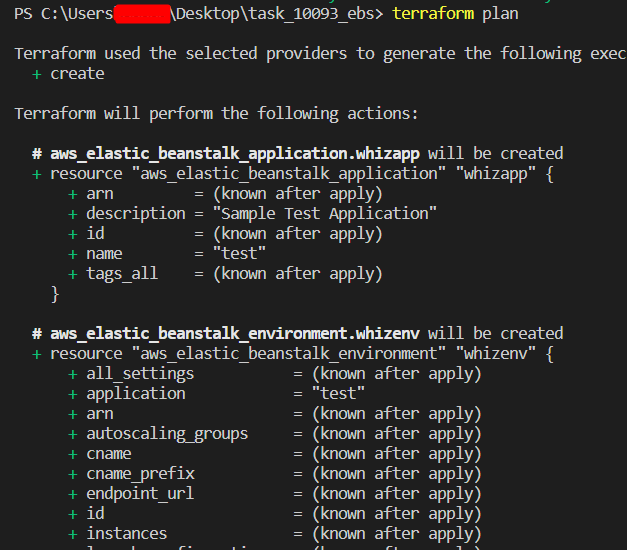
1. Initialize Terraform by running the below command,

terraform init



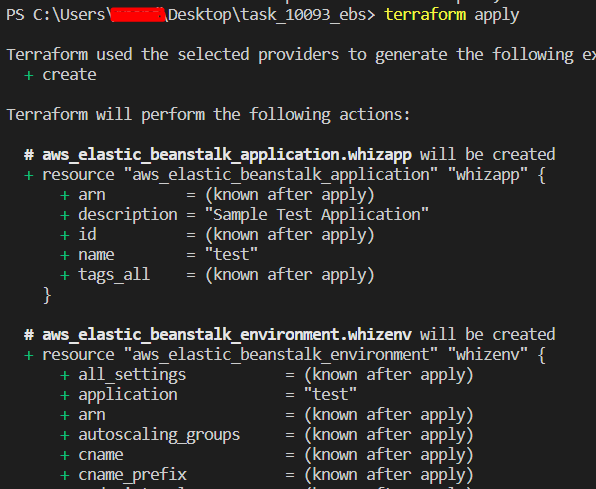
1. To generate the action plans run the below command,

terraform plan

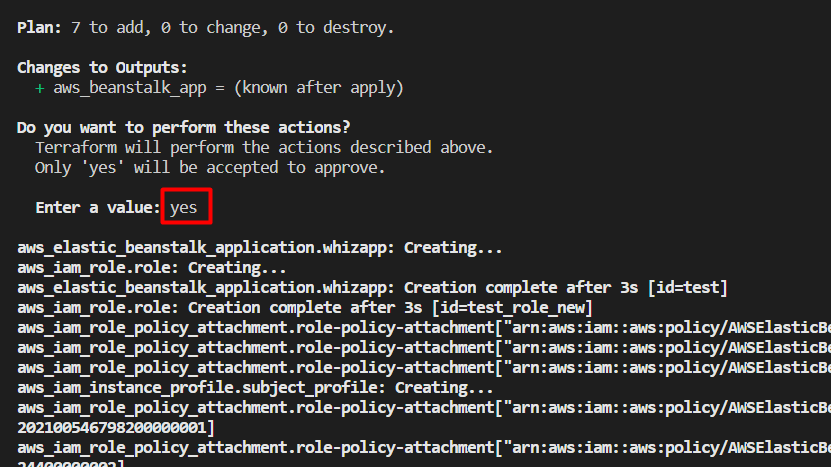


1. To create all the resources declared in **main.tf** configuration file, run the following command,

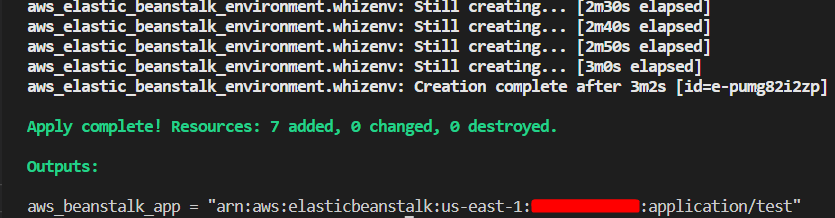
terraform apply



1. Enter **yes** and the resources will be created in **10-15** minutes.

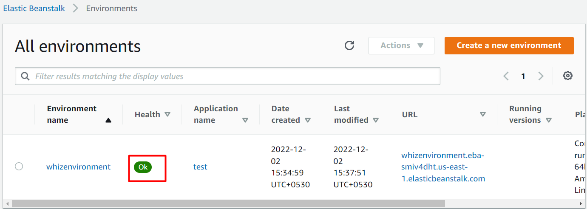


1. You can see the output printed.

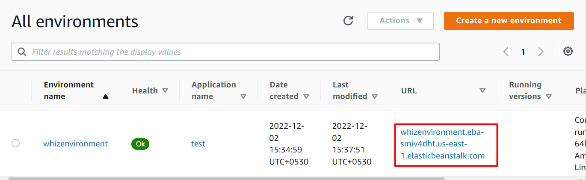


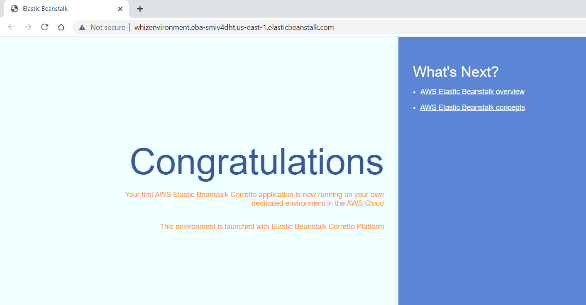
Task 9: Check the resources in the AWS Console

1. Navigate to the **Elastic Beanstalk** page by clicking on the **Services** menu at the top. **Elastic Beanstalk** is available under the **Compute** section.
2. Click on **Environments** on the left navigation panel.
3. You can see that the environment is created successfully.



1. **Copy the URL** and **paste** it into the browser. You can see that the application is created successfully.



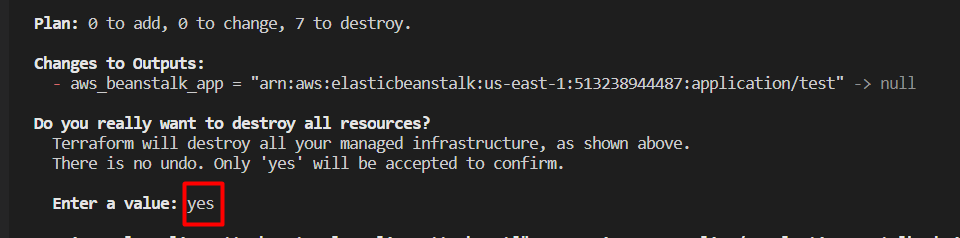
    

Task 10: Delete AWS Resources

1. To delete the resources, open **Terminal** again.
2. Run the below command to delete all the resources.

terraform destroy

1. Enter **yes** to confirm the deletion.



**Completion and Conclusion**

1. You have set up the Visual Studio Code
2. You have successfully created variables.tf and terraform.tfvars files.
3. You have successfully created a main.tf file.
4. You have executed the terraform configurations commands to create the resources.
5. You have checked the resources created by opening the AWS Console.
6. You have deleted all the resources created.

**End Lab**