**Lab Steps**

Task 1: Sign in to AWS Management Console

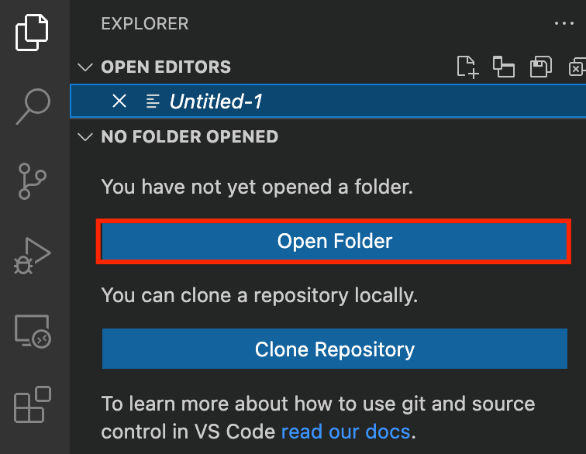
1. Click on the button, and you will get redirected to AWS Console in a new browser tab.
2. On the AWS sign-in page,

* 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 **US East (N. 12-digitVirginia) us-east-1**.

Task 2: Setup Visual Studio Code

1. Open the visual studio code.
2. If you have already installed and using 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.



1. 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\_10118

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

cd task\_10118

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\_10118**
4. (Optional) Click on Authorize button for allowing Visual Studio Code to use the task\_10118 folder. This will only be asked when you have been using Visual Studio code for a while as you are allowing a new folder to be accessed by VSC.
5. Visual Studio Code is now ready to use.

Task 3: Create a variable file

In this task, you will create variable files where you will declare all the global variables with a short description and a default value.

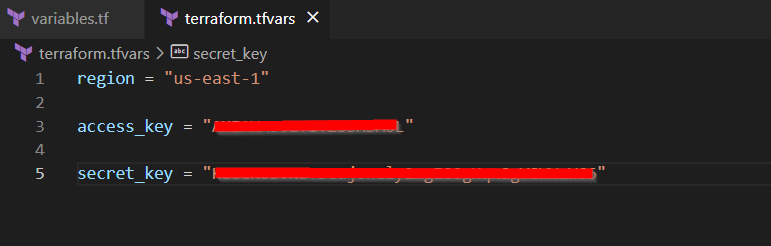
1. To create a variable file, expand the folder **task\_10118** and click on the **New File** icon to add the file.
2. Name the file as **variables.tf** and press **Enter** to save it.
3. **Note: Don't change the location of the new file, keep it default, i.e. inside the task\_10118 folder.**
4. Paste 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**.
3. Now expand the folder **task\_10118** and click on the **New File** icon to add the file.
4. Name the file as **terraform.tfvars** and press Enter to save it.
5. 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 a Launch template in the main.tf file

 In this task, you will create a main.tf file where you will add details of the provider and resources.

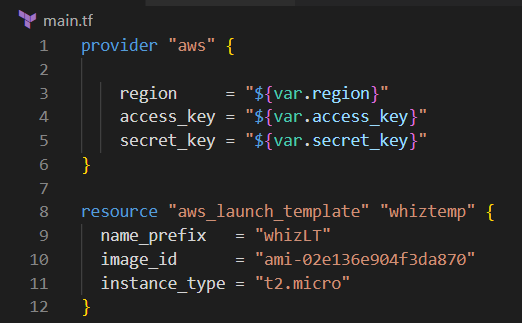
1. To create a **main.tf** file, expand the folder **task\_10118** and click on the **New File** icon to add the file.
2. Name the file as **main.tf** and press **Enter** to save it.
3. Paste the below content into the **main.tf** file.

|  |
| --- |
| 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 launch template for the autoscaling group.
3. Paste the following code into the **main.tf** file.

|  |
| --- |
| resource "aws\_launch\_template" "whiztemp" {    name\_prefix   = "whizLT"    image\_id      = "ami-02e136e904f3da870"    instance\_type = "t2.micro"  } |

1. In the above code, name of the launch template, image id and instance type is mentioned in the **main.tf** file.



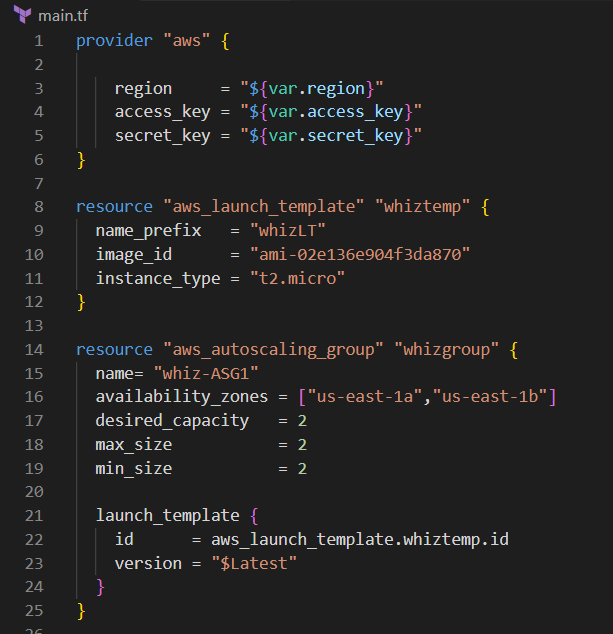
1. Save the file by pressing **Ctrl + S**.

Task 5: Create an autoscaling group in main.tf file

In this task we are going to create an autoscaling group.

1. To create an autoscaling group , paste the following contents in the **main.tf** file:

|  |
| --- |
| resource "aws\_autoscaling\_group" "whizgroup" {    name= "whiz-ASG1"    availability\_zones = ["us-east-1a","us-east-1b"]    desired\_capacity   = 2    max\_size           = 2    min\_size           = 2    launch\_template {      id      = aws\_launch\_template.whiztemp.id      version = "$Latest"    }  } |



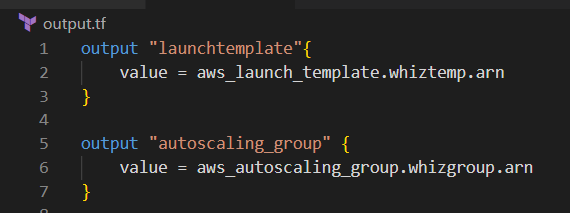
1. In the above code, the name , availability zones , desired capacity, maximum size , minimum size are defined. Also for the configuration of EC2 instances to be launched, the launch template created earlier is also associated with it.
2. Save the file by pressing **Ctrl + S**.

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** file, expand the folder **task\_10118** 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. In the above code, we will extract the **arn** of both the template and auto scaling group.

|  |
| --- |
| output "launchtemplate"{      value = aws\_launch\_template.whiztemp.arn  }  output "autoscaling\_group" {      value = aws\_autoscaling\_group.whizgroup.arn  } |



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\_10118

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

terraform version

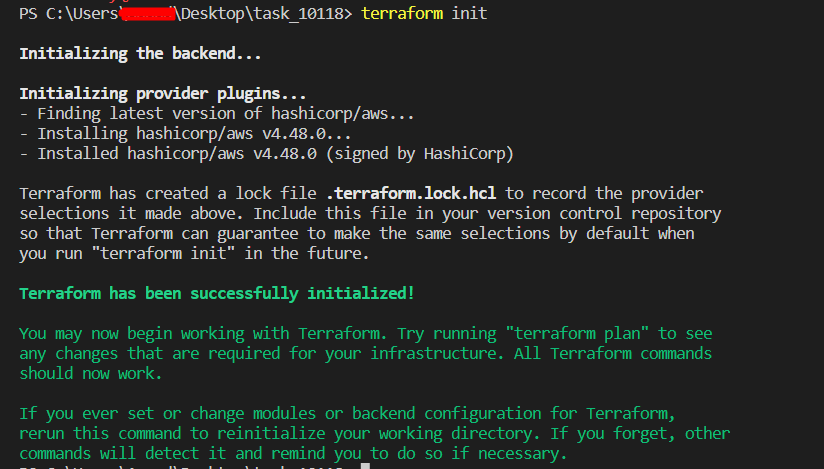
1. If you are getting output as command not found: terraform, this means that terraform is not installed on your system, To install terraform follow the official guide link provided in the Prerequisite section above.

Task 8: Apply terraform configurations

1. Initialize Terraform by running the below command,

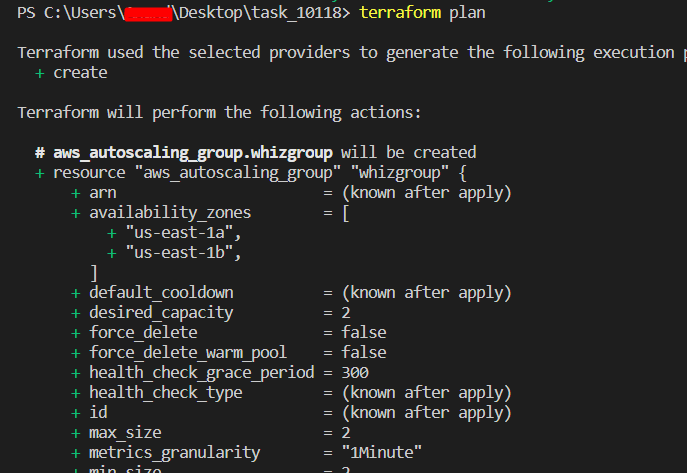
terraform init

**Note: terraform init will check for all the plugin dependencies and download them if required, this will be used for creating a deployment plan**



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

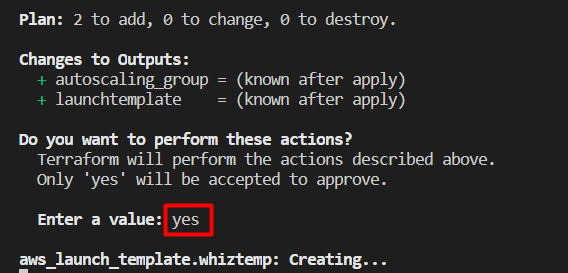
terraform plan



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

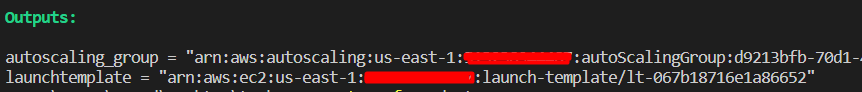
terraform apply

1. Approve the creation of all the resources by entering yes.



* **Note: This process will take around 5-10 minutes.**

1. Id’s of all the resources created by terraform will be visible there.

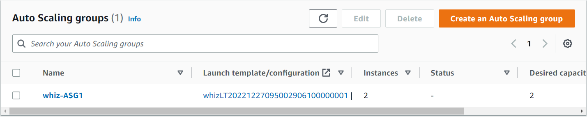


Task 9: Check the resources in AWS Console

1. Make sure you are in the **US East (N. Virginia) us-east-1** Region.
2. Navigate to **EC2** by clicking on **Services** on the top, then click on **EC2** in the **Compute** section.
3. Click on **Launch templates** in left navigation menu.



1. Click on **Auto Scaling Groups** under **Auto Scaling** in the left navigation panel. You can view the auto scaling group created successfully.

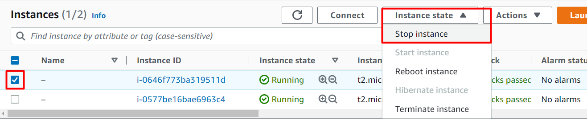


1. Go to **Instances** in the left navigation panel, you can see that two instances have been launched successfully.

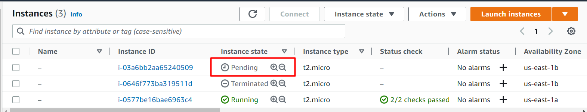


Task 10: Testing Auto Scaling group

1. For testing the auto-scaling policy, go to the EC2 instance list and select one of your instances.
2. Next, **select** any instance and click on **Instance state** and then **Stop instance**.



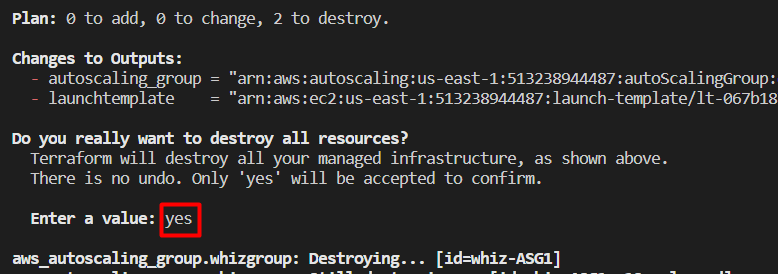
1. Click the **Stop** button on the pop-up window to stop your instance.
2. Once your instance is stopped (after 1-2 minutes) you can see that your stopped instance will be terminated automatically, and a new instance will be launched to fulfill the policy condition. This can take 4-5 minutes. A sample screenshot is provided below:



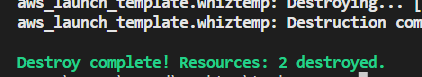
Task 11: Delete AWS Resources

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

terraform destroy



1. You can see the **Destroy complete!** Message.



**Completion and Conclusion**

* You have succesfully set up the Visual Studio Code editor.
* You have succesfully created variables.tf and terraform.tfvars files.
* You have successfully created a launch template in the main.tf file.
* You have successfully created an autoscaling group in the main.tf file
* You have successfully created an output file.
* You have successfully checked all the resources created by opening the Console.
* You have successfully tested the autoscaling group.
* You have deleted all the resources.

**End Lab**

1. Sign out of AWS Account.
2. You have successfully completed the lab.
3. Once you have completed the steps, click on from your whizlabs dashboard.