Home / AWS / Guided Lab / Access S3 bucket from EC2 Instance using Terraform

Access S3 bucket from EC2 Instance using Terraform

Level: **Intermediate**

Amazon EC2 Amazon S3 Identity And Access Management Amazon Web Services Terraform

0h 43m 54s left

End Lab

Open Console

Validation

Lab Credentials

User Name

Whiz_User_80425.90218978

WHIZLABS 0 K

AKIATKKMFIONSJNW4CWK

Secret Key

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

Lab Resources

No Lab Resources Found

Support Documents

1. FAQs and Troubleshooting
2. Labs - Instructions and Guidelines

Need help?

How to use Hands on Lab

Troubleshooting Lab

FAQs

Submit Feedback | Share

Lab Overview | Lab Steps | Lab Validation

Cloud Architect

Storage, Security, Compute, Infrastructure

Lab Steps

Task 1: Sign in to AWS Management Console

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

Note: If you face any issues, please go through **FAQs and Troubleshooting for Labs**.

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.
6. Once the terminal is ready, let us navigate to the Desktop.

```
cd desktop
```

7. Create a new folder by running the below command.

```
mkdir task_10003_ec2
```

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

```
cd task_10003_ec2
```

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

```
pwd
```

10. Note down the location, as you will open the same in the next steps.
11. Now click on the first icon Explorer present on the left sidebar.
12. Click on the button called Open folder and navigate to the location of folder **task_10003_ec2**.
13. (Optional) Click on Authorize button for allowing Visual Studio Code to use the **task_10003_ec2** 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.
14. Visual Studio Code is now ready to use.

Task 3: Create a variables file

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

1. To create variables files, click on the **File** from the menu bar and choose **New file**
2. Press Ctrl + S to save the new file as **variables.tf** and click on the **Save** button after entering the file name.

3. **Note:** Don't change the location of the new file, keep it default, i.e. inside the **task_10003_ec2** folder**.**

4. Paste the below contents in **variables.tf** file:

```
# required for AWS
variable "access_key" {}
variable "secret_key" {}
variable "region" {
    default = "us-east-1"
}
variable "bucket_name" {}
```

   In the above content, you are declaring a variable called, access_key, secret_key, and region having default value as us-east-1.

5. After pasting the above contents, save the file by pressing Ctrl + S.

6. Now create the **terraform.tfvars** file by selecting **New file** present under **File** in the menu bar.

7. Name the file by pressing Ctrl + S and enter **terraform.tfvars**

8. Paste the below content into **terraform.tfvars** file

```
access_key = "<YOUR AWS CONSOLE ACCESS ID>"
secret_key = "<YOUR AWS CONSOLE SECRET KEY>"
bucket_name = "<Your S3 Bucket name>"
```

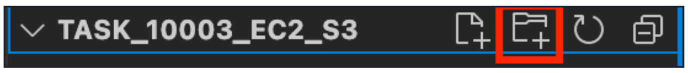
   In the above code, you are defining the dynamic values of variables declared earlier.

9. Replace the values of **access_key** and **secret_key** by copying from the lab page. And put a unique name for **bucket_name.**

10. After replacing the values of access_key and secret_key, **save the file by pressing Ctrl + S.**

Task 4: Create main.tf file and outputs.tf file

1. Create a new folder to place the index.html file. To create a folder, click on the **Create folder icon** present on the left side panel under your folder called **task_10003_ec2**.



2. Name this folder as **html.** And, press enter.

3. Next, download the sample HTML file below and place them inside the newly created folder called html.

- Download index.html

4. Or you can manually create an index.html inside the **html** folder and paste the contents of downloaded files.
5. To create **main.tf** file, click on the **File** from the menu bar and choose **New file**
6. Press Ctrl + S to save the new file as **main.tf** and click on the **Save** button after entering the file name.
7. Paste the below content into **main.tf** file. In below code you are defining aws provider and you will create an S3 Bucket and upload the index.html file into the newly created S3 Bucket.

```
provider "aws" {
     region     = var.region
     access_key = var.access_key
     secret_key = var.secret_key
}

resource "aws_s3_bucket" "blog" {
    bucket = var.bucket_name
}

resource "aws_s3_object" "object1" {
    for_each = fileset("html/", "*")
    bucket = aws_s3_bucket.blog.id
    key = each.value
    source = "html/${each.value}"
    etag = filemd5("html/${each.value}")
    content_type = "text/html"
}
```

8. **Before moving to the next step, save the file using Ctrl + S.**
9. Paste the below content into **main.tf** file.

```
resource "aws_instance" "web" {
    ami = "ami-02e136e904f3da870"
    instance_type = "t2.micro"
    vpc_security_group_ids = [aws_security_group.web-sg.id]
    iam_instance_profile = aws_iam_instance_profile.SSMRoleForEC2.name
    user_data = <<EOF

    #!/bin/bash
    sudo su
    yum update -y
    yum install httpd -y
    aws s3 cp s3://${aws_s3_bucket.blog.id}/index.html  /var/www/html/index.html
    systemctl start httpd
    systemctl enable httpd
    EOF
  tags = {
    Name = "Whiz-EC2-Instance"
  }
}
```

In the above code, you are defining resource block using **aws_instance** for creating an **Amazon EC2 Instance** having AMI ID as mentioned, EC2 Instance type as t2.micro, and IAM role or IAM Instance profile as mentioned in the next step. And, in user_data section, you are installing the HTTPD and copying index.html file from the s3 bucket.

10. Add the code for the Security group, authorizing inbound traffic from port 80 and port 443, and outbound traffic from all ports.

```
resource "aws_security_group" "web-sg" {
  name = "Web-SG"
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 443
    to_port     = 443
    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"]
  }
}
```

11. Add another block of just below the EC2 Instance creation code, this **block of code will create an IAM role having trust permission having use case of EC2 Instance.**

```
resource "aws_iam_role" "SSMRoleForEC2" {
    name = "SSMRoleForEC2"
    assume_role_policy = <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
}

resource "aws_iam_instance_profile" "SSMRoleForEC2" {
    name = "SSMRoleForEC2"
    role = aws_iam_role.SSMRoleForEC2.name
}
```

12. Attach the AWS Managed IAM Policy **AmazonSSMManagedInstanceCore** and **AmazonS3ReadOnlyAccess** to the IAM Role created above. This IAM Policy will allow EC2 Instance to use Session Manager for SSH without key pair and to access S3 buckets and its bucket.

```
resource "aws_iam_role_policy_attachment" "role-policy-attachment" {
  for_each = toset([
    "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
  ])

  role       = aws_iam_role.SSMRoleForEC2.name
  policy_arn = each.value
}
```

13. Create an **outputs.tf** file required for displaying the output as Public IP of EC2 Instance.

14. To create **outputs.tf** file, click on the **File** from the menu bar and choose **New file**

15. Press Ctrl + S to save the new file as **outputs.tf** and click on the **Save** button after entering the file name.

16. Paste the below content into **outputs.tf** file.

```
output "s3_bucket_id" {
  value = aws_instance.web.public_ip
}
```

17. Save the file by pressing Ctrl + S.

Task 5: Confirm the installation of Terraform by checking the version

1. Open the Terminal by clicking on the **View** from the menu bar and choosing **Terminal**

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

```
terraform version
```

3. 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 6: Apply terraform configurations

1. Initialize Terraform by running the below command:

```
terraform init
```

2. **Note:** terraform init will check for all the plugin dependencies and download them if required, this will be used for creating a deployment plan.
3. To generate the action plans run the below command:

```
terraform plan
```

4. Review the whole generated plan.
5. To create all the resources declared in **main.tf** configuration file, run the below command:

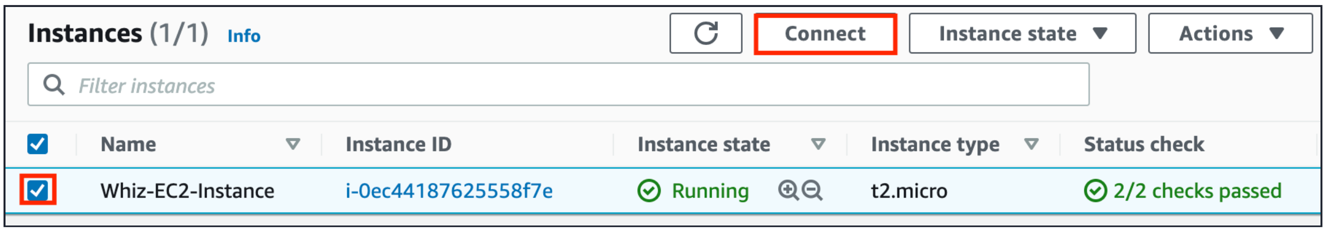
```
terraform apply
```

6. You will be able to see the resources which will be created, approve the creation of all the resources by entering **yes.**
7. It may take up to 2 minutes for the **terraform apply** command to create the resources.
8. Ids of all the resources created by terraform will be visible there.
9. Copy the public IP and paste into the browser to test the feature for the apache web server.
10. Optionally, you can note down the IDs of all the resources.

Task 7: SSH into EC2 Instance using Session manager through AWS Console

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

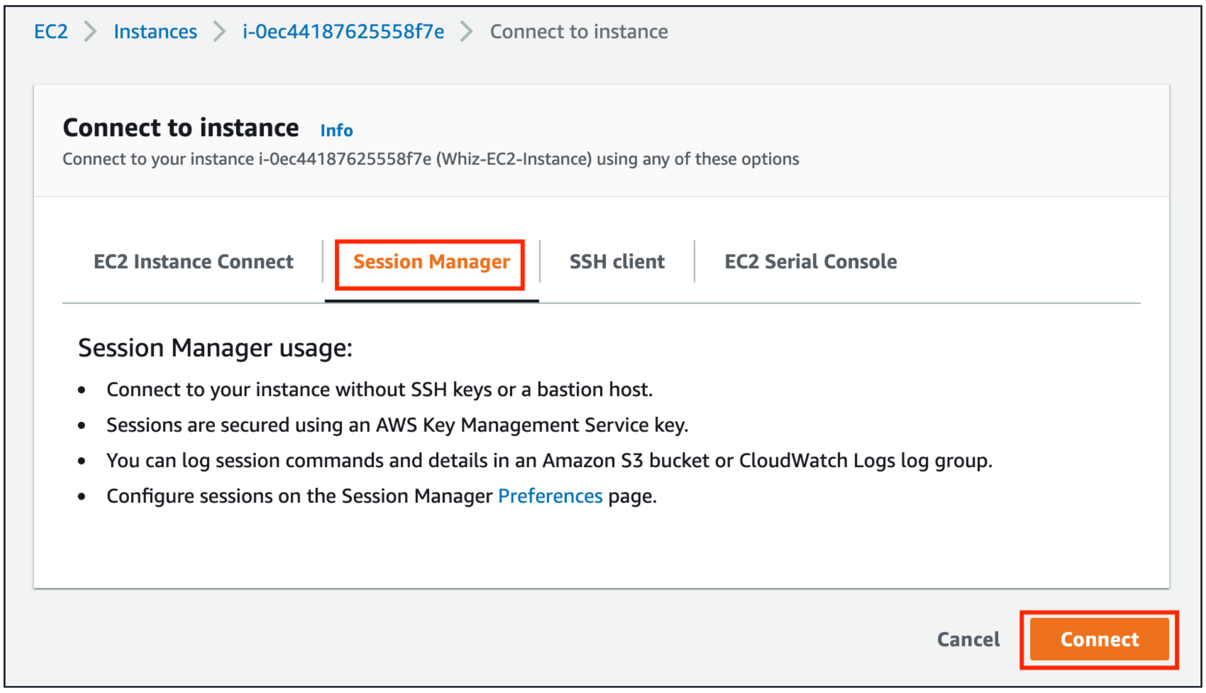
4. EC2 Instance may be Pending state, wait for in to get into **Running** state. And, complete the **Status check,** it will be updated to **2/2 checks passed** from Initializing
5. To use the Session manager feature for SSH, select the Instance and click on the **Connect** button.



Instances (1/1) Info

	Name	Instance ID	Instance state	Instance type	Status check
✓	Whiz-EC2-Instance	i-0ec44187625558f7e	Running	t2.micro	2/2 checks passed

6. Switch to the **Session Manager** tab and click on the **Connect** button.



EC2 > Instances > i-0ec44187625558f7e > Connect to instance

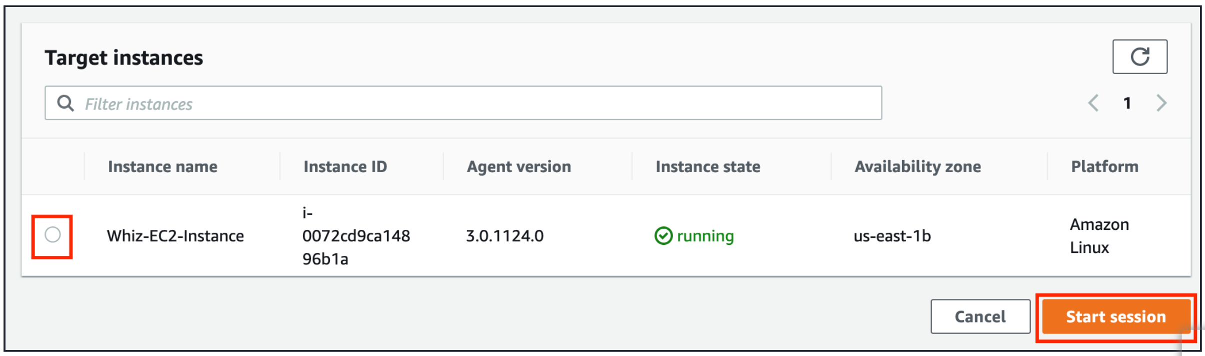
Connect to instance Info

Connect to your instance i-0ec44187625558f7e (Whiz-EC2-Instance) using any of these options

Session Manager usage:

- Connect to your instance without SSH keys or a bastion host.
- Sessions are secured using an AWS Key Management Service key.
- You can log session commands and details in an Amazon S3 bucket or CloudWatch Logs log group.
- Configure sessions on the Session Manager Preferences page.

7. Upon **Connect**, the Session Manager will create a new session in a new tab.
8. **Note 1:** Make sure you have enabled pop-up in your web browser.
9. **Note 2**: If the session manager throws an error, click on this link, **select the instance** and choose the **Start session** option.



Target instances

	Instance name	Instance ID	Agent version	Instance state	Availability zone	Platform
○	Whiz-EC2-Instance	i-0072cd9ca14896b1a	3.0.1124.0	running	us-east-1b	Amazon Linux

10. Run the first command to list all the S3 Buckets. And, copy the bucket name.

```
aws s3 ls
```

11. Copy the bucket name and list its objects.

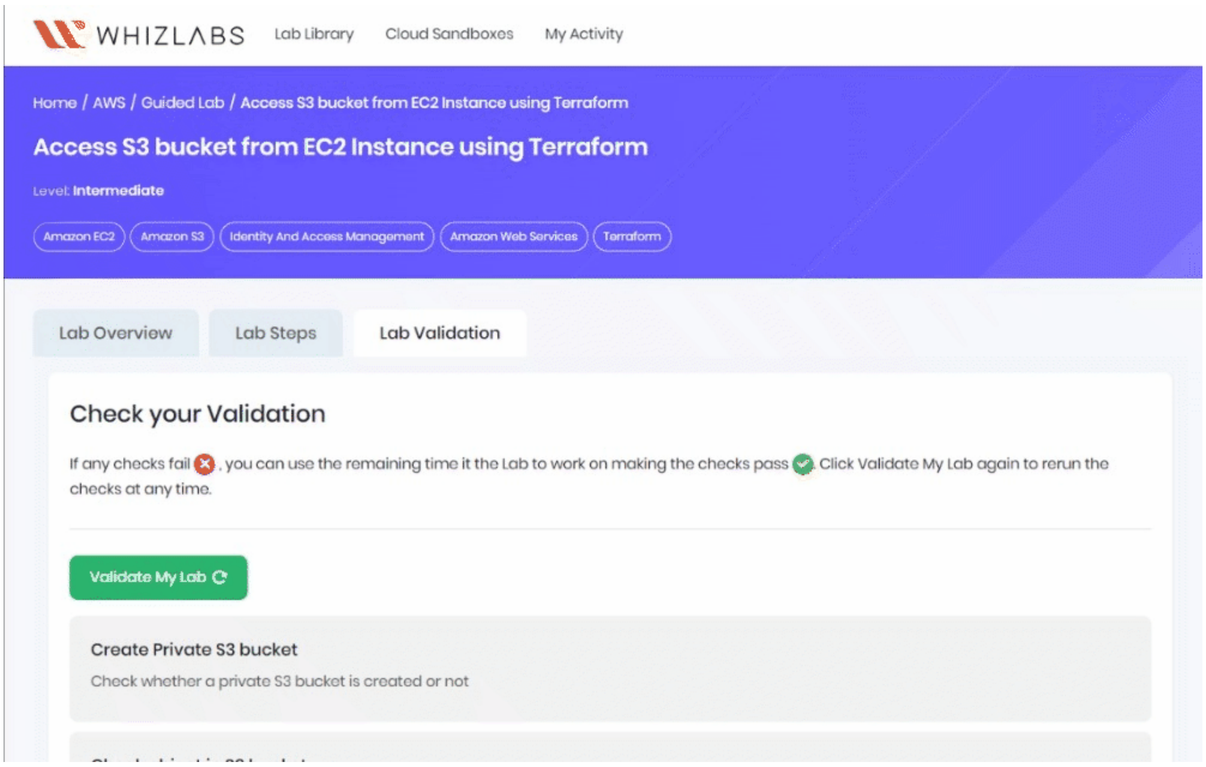
```
aws s3 ls s3://<Bucket-Name>
```

Do You know?

By using IAM roles and instance profiles, you can securely and conveniently grant EC2 instances the required permissions to access Amazon S3 and other AWS services, while also ensuring good security practices and avoiding the exposure of long-term access keys on the instances.

Task 8: Validation of the lab

1. Once the lab steps are completed, please click on the **Validation** button on the left side panel.
2. This will validate the resources in the AWS account and displays whether you have completed this lab successfully or not.
3. Sample output :



Access S3 bucket from EC2 Instance using Terraform

Level: Intermediate

Check your Validation

If any checks fail, you can use the remaining time it the Lab to work on making the checks pass. Click Validate My Lab again to rerun the checks at any time.

Create Private S3 bucket

Check whether a private S3 bucket is created or not

Task 9: Delete AWS Resources

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

```
terraform destroy
```

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

Completion and Conclusion

- You have successfully done the setup of the Visual Studio Code editor.
- You have successfully created **variables.tf** and **terraform.tfvars** file
- You have successfully created a **main.tf** file.
- You have successfully executed the terraform configuration commands to create the resources.
- You have successfully accessed EC2 Instance using Session Manager, And listed the S3 Bucket and its objects.
- You have successfully 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 **End lab** from your whizlabs dashboard.