# Simple steps for the *Cytominer VM*:

Things to be needed in the beginning:

1. S3 bucket (For mounting the S3 bucket, we need *S3 secret key* and *S3 access key*)
2. EFS-ID
3. Securitygroup_ID
4. Subnet_ID
5. VPC_ID

## How to get the S3 *Secret key and Access Key*?

To get your S3 access key and secret key, you need to create or retrieve AWS IAM user credentials. Follow these steps to generate these keys:

Using the AWS Management Console

1. Sign in to the AWS Management Console:

Go to the [AWS Management Console](https://aws.amazon.com/console/).

2. **Navigate to the IAM Service:**

In the search bar at the top, type ` IAM` and select **IAM (Identity and Access Management)** from the drop-down menu.

3. **Create a New IAM User (if needed):**

- In the left-hand menu, click on **Users**.
- Click on **Add user**.
- Enter a username.
- Select **Programmatic access** to create an access key and secret access key.
- Click **Next: Permissions**.

4. **Attach Policies to the User:**

  - Choose **Attach policies directly**.

  - Select a policy that grants the necessary permissions to S3, such as `AmazonS3FullAccess`. If you need more granular control, you can create a custom policy.

  - Click **Next: Tags** (optional) and then **Next: Review**.

  - Click **Create user**.

5. **Download Credentials:**

  - After the user is created, you will see the **Access key ID** and **Secret access key**.

  - Download the **.csv** file or copy the keys and store them securely. **This is the only time you will be able to view the secret access key.**

### Using an Existing IAM User

If you want to generate new access keys for an existing user:

1. **Navigate to the IAM Service:**

  In the search bar at the top, type `IAM` and select **IAM (Identity and Access Management)** from the drop-down menu.

2. **Select the User:**

  - In the left-hand menu, click on **Users**.

  - Click on the username of the user for whom you want to create access keys.

3. **Create Access Keys:**

  - Click on the **Security credentials** tab.

  - Scroll down to the **Access keys** section.

  - Click on **Create access key**.

  - A new **Access key ID** and **Secret access key** will be generated.

  - Download the **.csv** file or copy the keys and store them securely. **This is the only time you will be able to view the secret access key.**

*Security Best Practices*

*1. **Do not share your access keys.***

*2. **Rotate your access keys regularly.***

*3. **Use IAM roles instead of access keys when running on AWS services (e.g., EC2).***

*4. **Grant the least privilege required for your IAM user or roles.***

*By following these steps, you can generate or retrieve your S3 access key and secret key securely.*

## How to get the EFS-ID?

To obtain the EFS ID (Elastic File System ID) from AWS, follow these steps:

### Using the AWS Management Console

1. **Sign in to the AWS Management Console:**
  Go to the [AWS Management Console](https://aws.amazon.com/console/).

2. **Navigate to the EFS Service:**
  - In the search bar at the top, type `EFS` and select **Amazon Elastic File System** from the drop-down menu.

3. **Locate Your EFS File System:**
  - In the EFS dashboard, you will see a list of file systems if you have any created.
  - If you don't have an EFS file system, you will need to create one. Click on **Create file system** and follow the prompts to set up your EFS.

4. **Find the EFS ID:**
  - The EFS ID is listed in the **File system ID** column next to the name of your file system.
  - It typically looks like `fs-12345678`.

### Using the AWS CLI

If you prefer using the AWS CLI, you can obtain the EFS ID with the following command:

1. **List EFS File Systems:**

  Run the command to list your EFS file systems:
  ```sh
  aws efs describe-file-systems --query "FileSystems[*].FileSystemId" --output text
  ```
  This command will return the EFS IDs for all your file systems.

2. **Get Detailed Information:**
  If you need more details, you can describe a specific file system:
  ```sh
  aws efs describe-file-systems --file-system-id fs-12345678
  ```
  Replace `fs-12345678` with your actual EFS ID.

### Example CLI Commands

```sh
# List all EFS file systems and their IDs
aws efs describe-file-systems --query "FileSystems[*].FileSystemId" --output text

# Detailed description of a specific file system (replace with your EFS ID)
aws efs describe-file-systems --file-system-id fs-12345678
```

### Summary

By following these steps, you can easily find or obtain your EFS ID using either the AWS Management Console or the AWS CLI. Use this ID in your `efs.sh` script as described in your Packer configuration instructions.

## Configuring the Security group IDs (they should allow SSH access on port 22 NFS access on part 2049 (for EFS))

To configure your security groups to allow SSH access on port 22 and NFS access on port 2049 for EFS, follow these steps:

### Step-by-Step Guide

#### 1. **Create or Modify Security Groups**

1. **Sign in to the AWS Management Console:**
  Go to the [AWS Management Console](https://aws.amazon.com/console/).

2. **Navigate to the EC2 Dashboard:**
  In the Services menu, select **EC2** under the Compute section.

3. **View Security Groups:**
  In the left-hand menu, under **Network & Security**, click on **Security Groups**.

#### 2. **Create a New Security Group (Optional)**

If you don't already have a suitable security group, you can create a new one:

1. **Create Security Group:**
  - Click on the **Create Security Group** button.

2. **Configure Security Group:**
  - **Name:** Enter a name for your security group (e.g., `ssh-nfs-access`).
  - **Description:** Enter a description (e.g., `Allows SSH and NFS access`).
  - **VPC:** Select the VPC where you want this security group to be used.

3. **Add Inbound Rules:**
  - Click on the **Add Rule** button.

#### 3. **Add Inbound Rules for SSH and NFS**

**Inbound Rules:**

1. **SSH Access:**
  - **Type:** SSH
  - **Protocol:** TCP
  - **Port Range:** 22
  - **Source:** Custom (or your IP, or a security group for internal access)
  - **CIDR/IP:** `0.0.0.0/0` (for global access, restrict to specific IP or CIDR for security)

2. **NFS Access:**
  - **Type:** Custom TCP Rule
  - **Protocol:** TCP
  - **Port Range:** 2049
  - **Source:** Custom (or your VPC CIDR block)
  - **CIDR/IP:** `0.0.0.0/0` (for global access, restrict to specific IP or CIDR for security)

3. **Save Rules:**
  - After adding the rules, click on **Create** to save the security group.

#### 4. **Modify an Existing Security Group (If Applicable)**

If you already have a security group that you want to use, you can add the necessary rules:

1. **Select Security Group:**
  - Find and select the security group you want to modify from the list.

2. **Edit Inbound Rules:**
  - Click on the **Inbound rules** tab and then click on **Edit inbound rules**.

3. **Add Rules:**
  - Follow the same steps as described above to add rules for SSH and NFS.

4. **Save Rules:**
  - After adding the rules, click on **Save rules**.

### Summary of Security Group Rules

- **SSH Access (Port 22):**
  - Type: SSH
  - Protocol: TCP
  - Port Range: 22
  - Source: Custom (specify your IP, CIDR, or security group)
  - CIDR/IP: `0.0.0.0/0` (or a more restricted range)

- **NFS Access (Port 2049):**
  - Type: Custom TCP Rule
  - Protocol: TCP
  - Port Range: 2049
  - Source: Custom (specify your VPC CIDR block or security group)
  - CIDR/IP: `0.0.0.0/0` (or a more restricted range)

By following these steps, you can ensure that your security groups are configured to allow the necessary access for your Packer configuration.

# How to create an IAM-S3-Role

An IAM (Identity and Access Management) role in the configuration instance details allows your EC2 instance to interact with AWS services without needing to manually provide credentials. The IAM role grants specific permissions to the instance.

### How to Create and Assign an IAM Role

#### 1. **Create an IAM Role**

1. **Sign in to the AWS Management Console:**
  Go to the [AWS Management Console](https://aws.amazon.com/console/).

2. **Navigate to the IAM Dashboard:**

In the Services menu, select **IAM** under the Security, Identity, & Compliance section.

3. **Create a New Role:**

- In the left-hand menu, click on **Roles**.
- Click the **Create role** button.

4. **Select Trusted Entity:**

- Choose **AWS service**.
- Select **EC2** as the service that will use this role.
- Click **Next: Permissions**.

5. **Attach Policies:**

- Attach the policies that your instance will need. For instance, to allow access to S3 and EFS, you might choose:
  - `AmazonS3FullAccess`
  - `AmazonElasticFileSystemFullAccess`
- Click **Next: Tags**.

6. **Add Tags (Optional):**

- Add any tags you need (optional).
- Click **Next: Review**.

7. **Review and Create Role:**

- Give the role a name, such as `s3-imaging-platform-role`.
- Review the settings and click **Create role**.

#### 2. **Assign the IAM Role to Your EC2 Instance**

1. **Navigate to the EC2 Dashboard:**

In the Services menu, select **EC2** under the Compute section.

2. **View Instances:**

   In the left-hand menu, click on **Instances**.

3. **Select Your Instance:**

   - Select the instance to which you want to assign the IAM role.

4. **Actions Menu:**

   - Click on the **Actions** button.

   - Under **Security**, select **Modify IAM Role**.

5. **Attach IAM Role:**

   - In the **IAM role** drop-down, select the IAM role you created earlier (`s3-imaging-platform-role`).

   - Click **Update IAM role**.

### Summary

By following these steps, you create an IAM role with the necessary permissions and assign it to your EC2 instance. This setup allows your instance to interact with S3 and EFS without manually handling AWS credentials, enhancing security and simplifying management.

### Example Configuration in Instance Details

When launching a new EC2 instance, you can specify the IAM role during the configuration:

- **Instance Type:** m4.xlarge
- **Network:** vpc-35149752
- **Subnet:** Default (imaging platform terraform)
- **IAM Role:** s3-imaging-platform-role
- **Security Group:** SSH_HTTP

This IAM role ensures your EC2 instance can access the necessary AWS resources, such as S3 and EFS, based on the permissions you assigned to the role.

## Where to get the subnet-ids?

To obtain the subnet ID from AWS, follow these steps:

### Using the AWS Management Console

1. **Sign in to the AWS Management Console:**
  Go to the [AWS Management Console](https://aws.amazon.com/console/).

2. **Navigate to the EC2 Dashboard:**
  In the Services menu, select **EC2** under the Compute section.

3. **View Subnets:**
  In the left-hand menu, under **Network & Security**, click on **Subnets**.

4. **Find Subnet ID:**
  - The list of subnets will be displayed.
  - The **Subnet ID** column contains the subnet IDs.
  - Note the ID of the subnet you want to use. It looks like `subnet-xxxxxxxx`.

### Steps Illustrated with Screenshots

1. **EC2 Dashboard:**
  ![EC2 Dashboard](https://d1.awsstatic.com/console-screenshots/ec2_dashboard.f019a295bfcd7efbfe429caedc7d9e3b3c97c31a.png)

2. **Network & Security -> Subnets:**
  ![Subnets](https://docs.aws.amazon.com/vpc/latest/userguide/images/subnet-list.png)

### Using the AWS CLI

Alternatively, you can use the AWS CLI to list and describe subnets.

1. **List Subnets:**
  Run the following command to list all subnets:
  ```sh
  aws ec2 describe-subnets --query "Subnets[*].{ID:SubnetId,VPC:VpcId,CIDR:CidrBlock}" --output table
  ```

2. **Detailed Information:**

To get more detailed information about a specific subnet, use:
```sh
aws ec2 describe-subnets --subnet-ids subnet-xxxxxxxx
```

### Example CLI Commands

```sh
# List all subnets with their IDs, VPCs, and CIDR blocks
aws ec2 describe-subnets --query "Subnets[*].{ID:SubnetId,VPC:VpcId,CIDR:CidrBlock}" --output table

# Get detailed information about a specific subnet
aws ec2 describe-subnets --subnet-ids subnet-xxxxxxxx
```

### Summary

By following these steps, you can easily find the subnet ID from the AWS Management Console or using the AWS CLI. Use this subnet ID in your Packer configuration as required.

## Where to get the VPC-id?

The VPC-ID can also found on the EFS-ID page and the subnet-ID page. If there is an issue its always a good thing to go and search for the vpc id in the managenment console.

## How to delete the custom AMI completely when something goes wrong?

To delete a custom Amazon Machine Image (AMI) from the EC2 instance menu in the AWS Management Console, follow these steps:

1. **Log in to the AWS Management Console**:
 Open your web browser and log in to the [AWS Management Console](https://aws.amazon.com/console/).

2. **Navigate to the EC2 Dashboard**:
 In the AWS Management Console, navigate to the EC2 service. You can find EC2 under the "Compute" section or by using the search bar at the top of the console.

3. **Open the AMI List**:
 On the EC2 Dashboard, look for the "Images" section in the left-hand navigation pane and click on "AMIs."

4. **Locate the Custom AMI**:
In the AMIs list, you will see all the AMIs available to you, including both public and private AMIs. You can filter by "Owned by me" to find the AMIs you have created.

5. **Select the Custom AMI**:
Find the custom AMI you want to delete. Select it by clicking the checkbox next to the AMI ID.

6. **Deregister the AMI**:
With the custom AMI selected, click on the "Actions" button at the top of the page, then select "Deregister AMI" from the dropdown menu.

7. **Confirm Deregistration**:
A confirmation dialog will appear asking if you are sure you want to deregister the AMI. Confirm by clicking the "Deregister" button.

> **Note:** Deregistering an AMI does not delete the associated snapshots. You will need to delete those separately if you no longer need them.

8. **Delete Associated Snapshots**:
To delete the snapshots associated with the AMI, follow these steps:
- In the left-hand navigation pane, click on "Snapshots" under the "Elastic Block Store" section.
- Locate the snapshots that are associated with the deregistered AMI. You can use the snapshot IDs listed in the "Description" field of the AMI details.
- Select the snapshots you want to delete by clicking the checkboxes next to them.
- Click on the "Actions" button, then select "Delete Snapshot" from the dropdown menu.
- Confirm the deletion by clicking the "Delete" button in the confirmation dialog.

By following these steps, you will successfully deregister a custom AMI and delete the associated snapshots from your AWS account.

# Building the VM:

Now we Build the VM following the “***VM for profiling / Cytominer-vm”*** documentation.

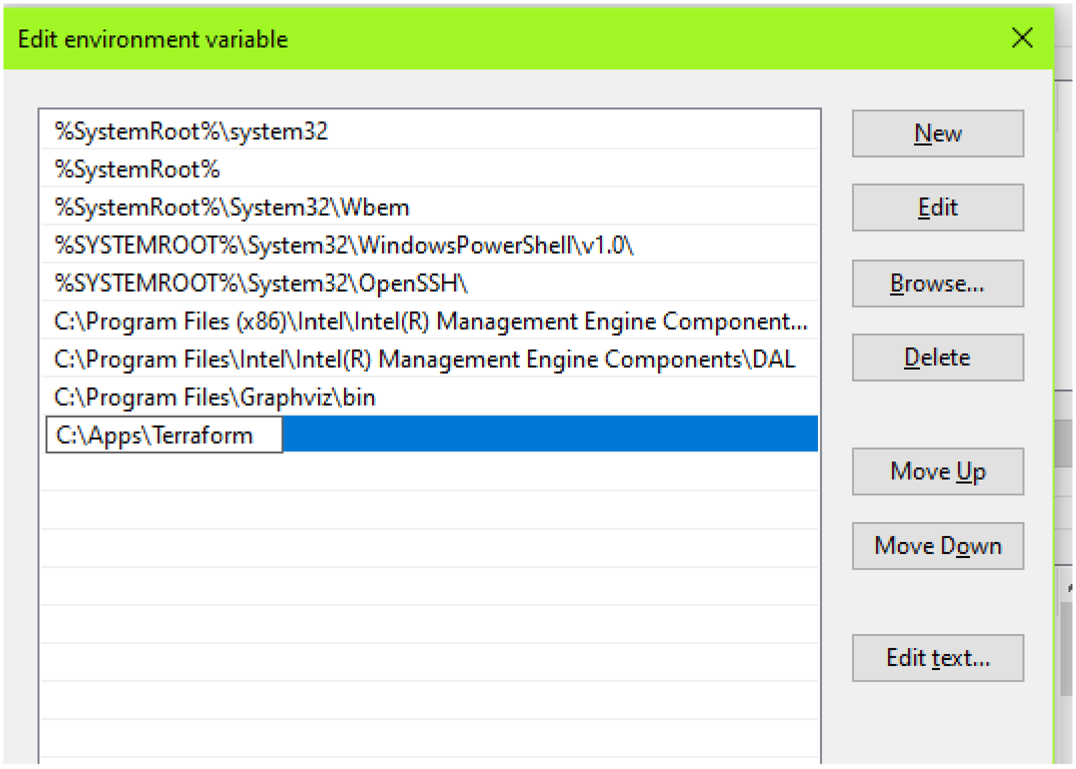
1. First make a directory anywhere. I made a directory on desktop using *“mkdir”* command.
2. Then using the *“gitclone”* command clone the cytominer-vm in the directory. Use command: *“git clone https://github.com/cytomining/cytominer-vm.git”*
3. Now we need to edit the efs.sh and s3.sh files using own credential. Editing these files can be done with VS code.
4. In efs.sh, we need to mention the EFS-ID and the save it.
5. In s3.sh, we need to mention the S3-Role and the Bucket ID . Eg. If you have a bucket named *‘dltrain’* the corresponding S3 role will be s3-cytominerVM-role and bucket ID will be dl-train.

6. Now open the cmd to initiate the AWS-CLI (You need to install AWS-Cli initially.)
7. Insert your secret key and access key and mount your s3 bucket.
8. Next we install packer. Its important to follow the instruction from Hashi-corp page.
9. IMPORTANT: After installing packer its important to set a path for the packer, which will help us to run it from the command line. Follow the stackoverflow article for setting it. Below is an example for making the path for 'Terraform'. We can follow the same steps for packer.

Here I'm providing solution to setup Terraform environment variable in windows for beginners.

1. Download the terraform ZIP file from Terraform site.
2. Extract the .exe from the ZIP file to a folder eg C:\Apps\Terraform copy this path location like C:\Apps\terraform\
3. Add the folder location to your PATH variable, eg: `Control Panel -> System -> System settings -> Environment Variables`

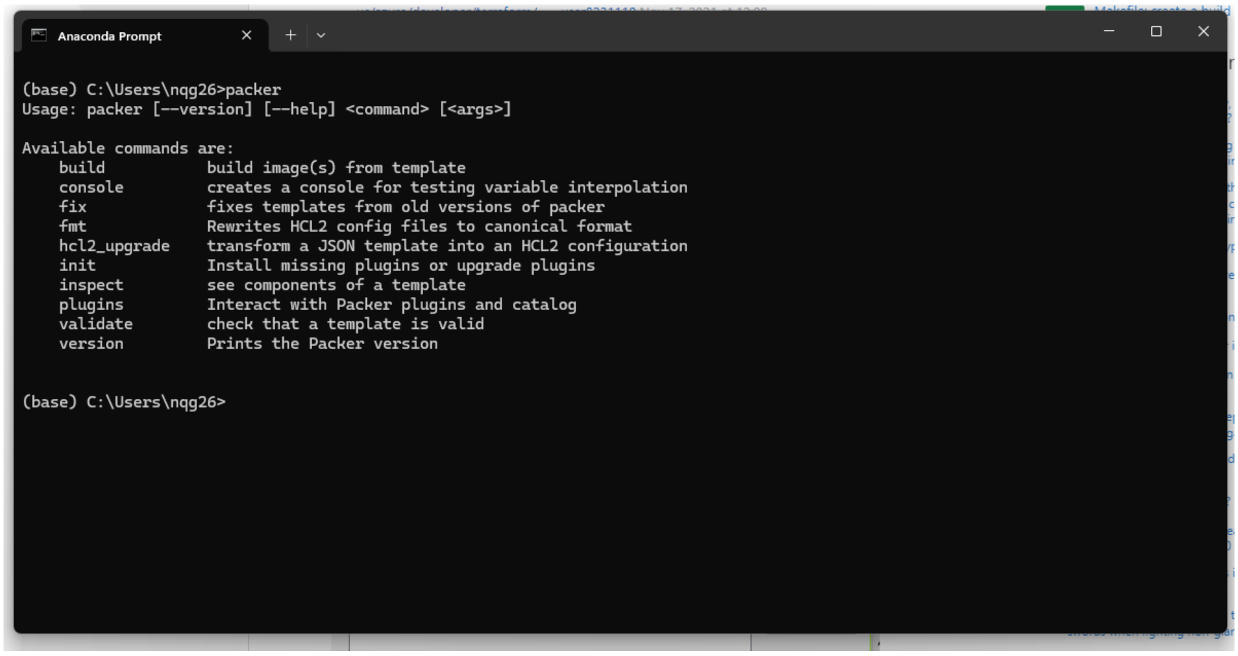
In `System Variables`, select `Path` > `edit` > `new` > Enter the location of the Terraform .exe, eg `C:\Apps\Terraform` then click `OK`


Edit environment variable

%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files (x86)\Intel\Intel(R) Management Engine Component...
C:\Program Files\Intel\Intel(R) Management Engine Components\DAL
C:\Program Files\Graphviz\bin
C:\Apps\Terraform

5. Open a new CMD/PowerShell and the Terraform command should work

10. After this, packer installation can be verified from command prompt by typing $packer command. The window will look like this below.

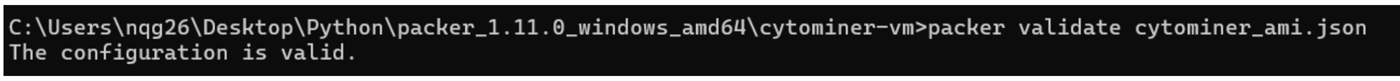


```
(base) C:\Users\nqg26>packer
Usage: packer [--version] [--help] <command> [<args>]

Available commands are:
    build           build image(s) from template
    console         creates a console for testing variable interpolation
    fix             fixes templates from old versions of packer
    fmt             Rewrites HCL2 config files to canonical format
    hcl2_upgrade    transform a JSON template into an HCL2 configuration
    init            Install missing plugins or upgrade plugins
    inspect         see components of a template
    plugins         Interact with Packer plugins and catalog
    validate        check that a template is valid
    version         Prints the Packer version


(base) C:\Users\nqg26>
```

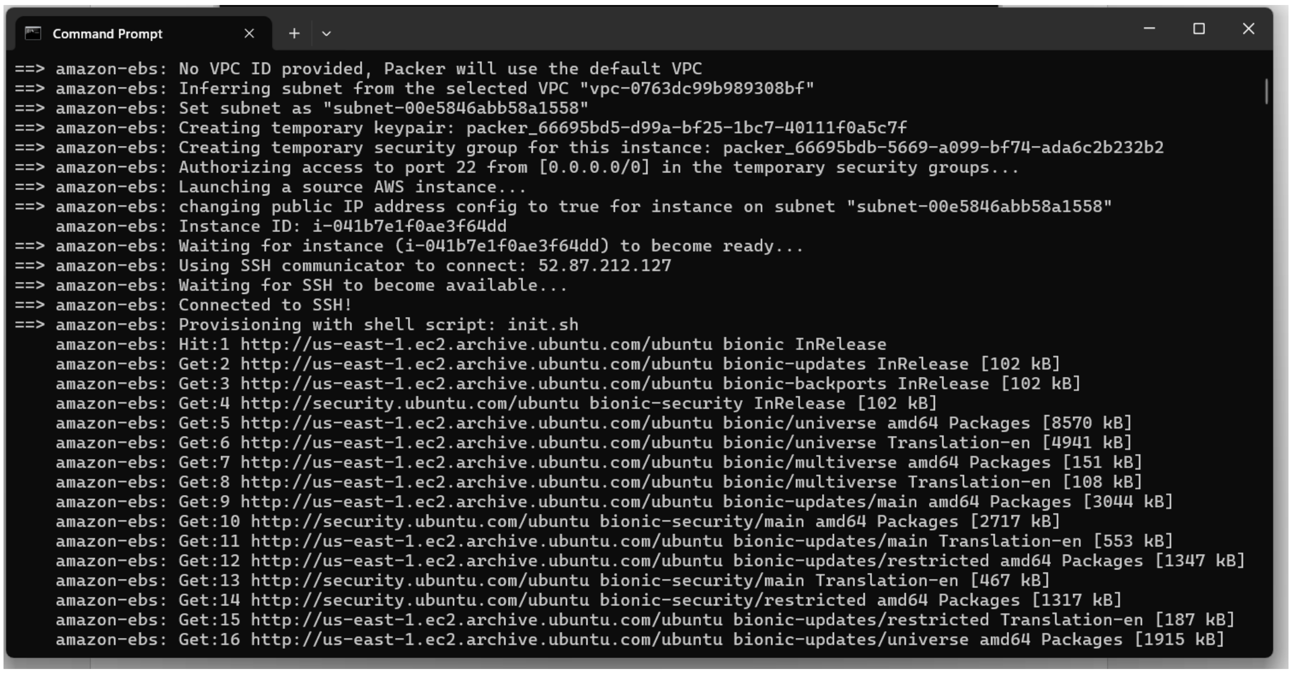
11. Next we validate the preliminary settings before creating the machine image. In cmd we write ‘ *packer validate cytominer_ami.json* ’



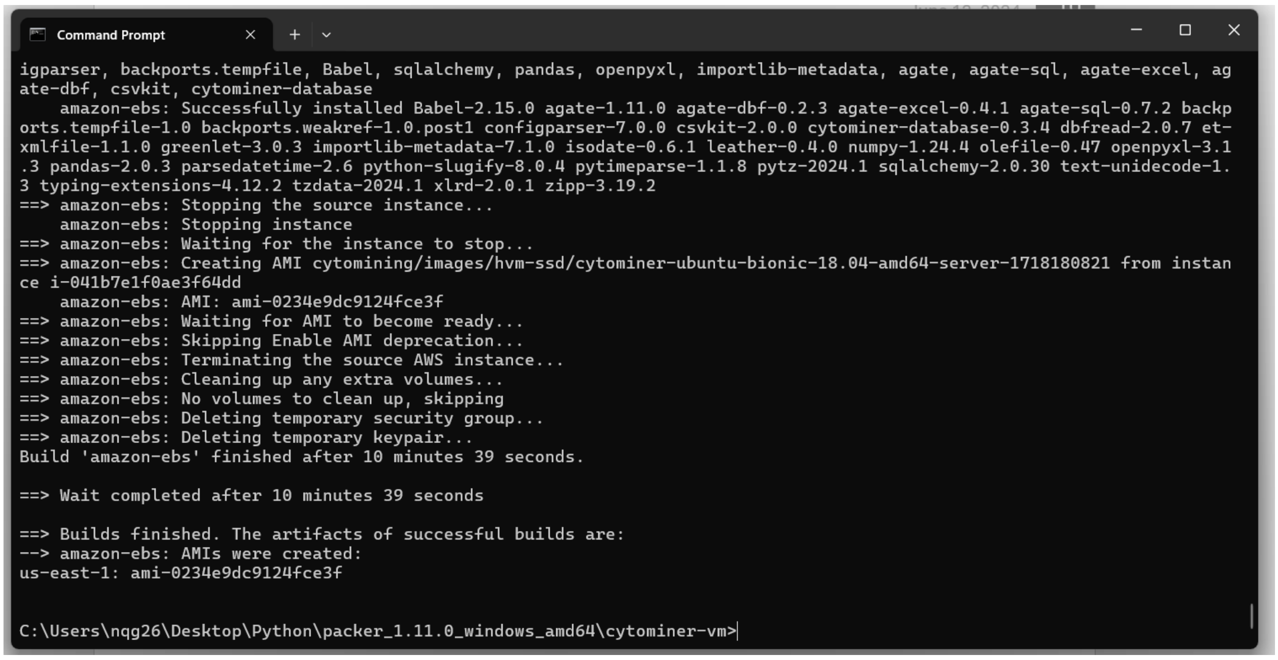
```
C:\Users\nqg26\Desktop\Python\packer_1.11.0_windows_amd64\cytominer-vm>packer validate cytominer_ami.json
The configuration is valid.
```

12. Next we edit cytominer_ami.json with the security_group_id, subnet_id, and vpc_id (from AWS console) and run the following command to build the AMI (Amazon Machine image / Cytominer VM):

    packer build -var 'security_group_ids=sg-01234567,sg-89abcdef' -var 'subnet_id=subnet-01234567' -var 'vpc_id=vpc-01234567' cytominer_ami.json



```
==> amazon-ebs: No VPC ID provided, Packer will use the default VPC
==> amazon-ebs: Inferring subnet from the selected VPC "vpc-0763dc99b989308bf"
==> amazon-ebs: Set subnet as "subnet-00e5846abb58a1558"
==> amazon-ebs: Creating temporary keypair: packer_66695bd5-d99a-bf25-1bc7-40111f0a5c7f
==> amazon-ebs: Creating temporary security group for this instance: packer_66695bdb-5669-a099-bf74-ada6c2b232b2
==> amazon-ebs: Authorizing access to port 22 from [0.0.0.0/0] in the temporary security groups...
==> amazon-ebs: Launching a source AWS instance...
==> amazon-ebs: changing public IP address config to true for instance on subnet "subnet-00e5846abb58a1558"
    amazon-ebs: Instance ID: i-041b7e1f0ae3f64dd
==> amazon-ebs: Waiting for instance (i-041b7e1f0ae3f64dd) to become ready...
==> amazon-ebs: Using SSH communicator to connect: 52.87.212.127
==> amazon-ebs: Waiting for SSH to become available...
==> amazon-ebs: Connected to SSH!
==> amazon-ebs: Provisioning with shell script: init.sh
    amazon-ebs: Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic InRelease
    amazon-ebs: Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease [102 kB]
    amazon-ebs: Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelease [102 kB]
    amazon-ebs: Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [102 kB]
    amazon-ebs: Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [8570 kB]
    amazon-ebs: Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic/universe Translation-en [4941 kB]
    amazon-ebs: Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [151 kB]
    amazon-ebs: Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic/multiverse Translation-en [108 kB]
    amazon-ebs: Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [3044 kB]
    amazon-ebs: Get:10 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [2717 kB]
    amazon-ebs: Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/main Translation-en [553 kB]
    amazon-ebs: Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [1347 kB]
    amazon-ebs: Get:13 http://security.ubuntu.com/ubuntu bionic-security/main Translation-en [467 kB]
    amazon-ebs: Get:14 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [1317 kB]
    amazon-ebs: Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/restricted Translation-en [187 kB]
    amazon-ebs: Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1915 kB]
```



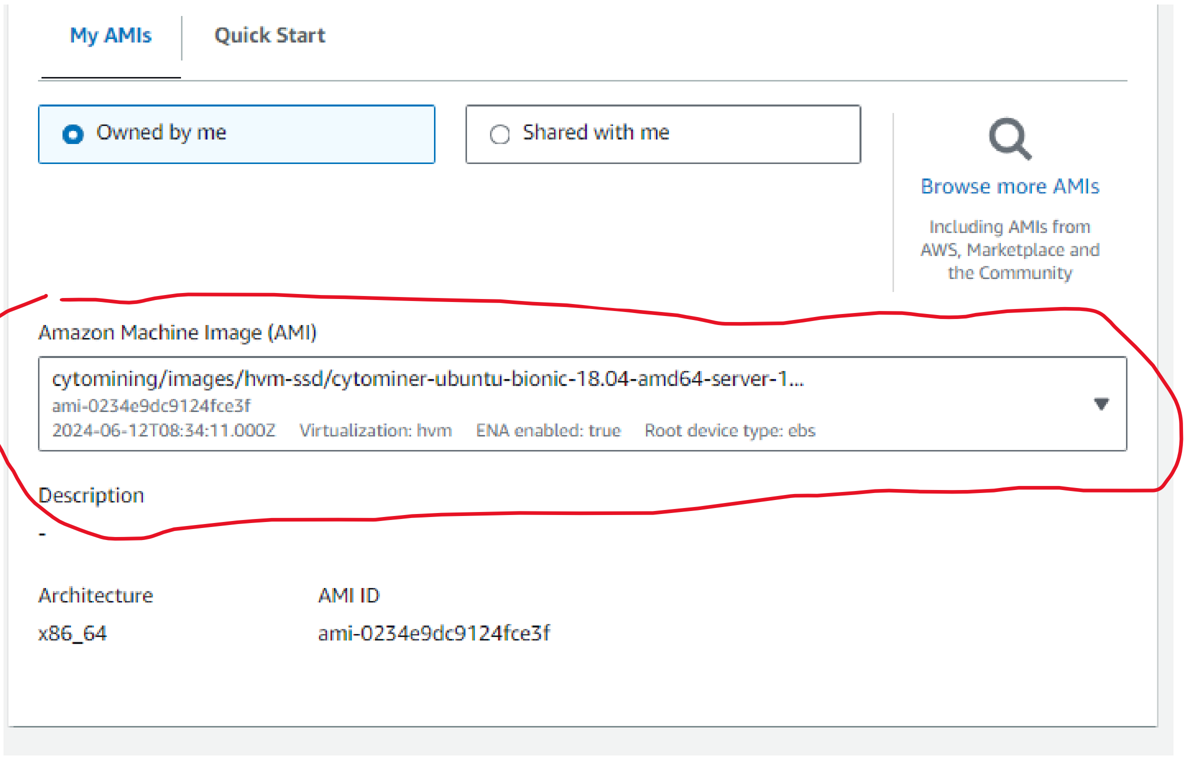
```
igparser, backports.tempfile, Babel, sqlalchemy, pandas, openpyxl, importlib-metadata, agate, agate-sql, agate-excel, ag
ate-dbf, csvkit, cytominer-database
    amazon-ebs: Successfully installed Babel-2.15.0 agate-1.11.0 agate-dbf-0.2.3 agate-excel-0.4.1 agate-sql-0.7.2 backp
orts.tempfile-1.0 backports.weakref-1.0.post1 configparser-7.0.0 csvkit-2.0.0 cytominer-database-0.3.4 dbfread-2.0.7 et-
xmlfile-1.1.0 greenlet-3.0.3 importlib-metadata-7.1.0 isodate-0.6.1 leather-0.4.0 numpy-1.24.4 olefile-0.47 openpyxl-3.1
.3 pandas-2.0.3 parsedatetime-2.6 python-slugify-8.0.4 pytimeparse-1.1.8 pytz-2024.1 sqlalchemy-2.0.30 text-unidecode-1.
3 typing-extensions-4.12.2 tzdata-2024.1 xlrd-2.0.1 zipp-3.19.2
==> amazon-ebs: Stopping the source instance...
    amazon-ebs: Stopping instance
==> amazon-ebs: Waiting for the instance to stop...
==> amazon-ebs: Creating AMI cytomining/images/hvm-ssd/cytominer-ubuntu-bionic-18.04-amd64-server-1718180821 from instan
ce i-041b7e1f0ae3f64dd
    amazon-ebs: AMI: ami-0234e9dc9124fce3f
==> amazon-ebs: Waiting for AMI to become ready...
==> amazon-ebs: Skipping Enable AMI deprecation...
==> amazon-ebs: Terminating the source AWS instance...
==> amazon-ebs: Cleaning up any extra volumes...
==> amazon-ebs: No volumes to clean up, skipping
==> amazon-ebs: Deleting temporary security group...
==> amazon-ebs: Deleting temporary keypair...
Build 'amazon-ebs' finished after 10 minutes 39 seconds.

==> Wait completed after 10 minutes 39 seconds

==> Builds finished. The artifacts of successful builds are:
--> amazon-ebs: AMIs were created:
us-east-1: ami-0234e9dc9124fce3f


C:\Users\nqg26\Desktop\Python\packer_1.11.0_windows_amd64\cytominer-vm>
```

13. Now the VM is created, it can be found on the EC2 Instances dashboard on aws.



My AMIs | Quick Start

Owned by me [✓] | Shared with me

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

cytomining/images/hvm-ssd/cytominer-ubuntu-bionic-18.04-amd64-server-1...
ami-0234e9dc9124fce3f
2024-06-12T08:34:11.000Z Virtualization: hvm ENA enabled: true Root device type: ebs

Description

-

| Architecture | AMI ID |
|---|---|
| x86_64 | ami-0234e9dc9124fce3f |

# Launching the CytominerVM from EC2 instances

## Possible errors:

1. to me: C:\Users\nqg26\Desktop\Python\packer_1.11.0_windows_amd64\cytominer-vm>packer validate cytominer_ami.json

   **Error: Failed to initialize build "amazon-ebs"**

   The builder amazon-ebs is unknown by Packer, and is likely part of a plugin that

   is not installed. You may find the needed plugin along with installation instructions documentedon the Packer integrations page.

   https://developer.hashicorp.com/packer/integrations?filter=amazon

   **Fix:** The error message you're seeing indicates that Packer doesn't recognize the `amazon-ebs` builder, which means the necessary plugin for Amazon Elastic Block Store (EBS) is not installed or not properly configured. Here are the steps to resolve this issue:

   Real Fix followed: use code " packer plugins install github.com/hashicorp/amazon "

   Step 1: Install the Amazon EBS Plugin

   1. **Download the Plugin:**

   - Visit the [Packer integrations page](https://developer.hashicorp.com/packer/integrations?filter=amazon).
   - Find the `amazon` plugin and download the appropriate version for your system.

   2. **Install the Plugin:**

   - Unzip the downloaded plugin.
   - Move the plugin to the Packer plugin directory. By default, this directory is located at `~/.packer.d/plugins`.

   ### Step 2: Verify Plugin Installation

Ensure the plugin is installed correctly by running the following command:

```bash
packer plugins
```

This command should list the `amazon` plugin among others.

### Step 3: Validate the Packer Template

After ensuring the plugin is installed, run the `packer validate` command again:

```bash
packer validate cytominer_ami.json
```

### Additional Steps (if needed):

If the above steps don't resolve the issue, follow these additional steps:

1. **Check Packer Version:**

 Ensure you are using a version of Packer that supports the `amazon-ebs` builder. You can check your Packer version with:

 ```bash
 packer --version
 ```

2. **Update Packer:**

 If your Packer version is outdated, download and install the latest version from the [Packer download page](https://developer.hashicorp.com/packer/downloads).

3. **Review Configuration:**

 Double-check your `cytominer_ami.json` file for any configuration issues. Make sure the builder section is correctly specified as:

 ```json
 {
   "builders": [
 ```

```
    {
     "type": "amazon-ebs",
     // other configuration options
    }
   ]
  }
```

Following these steps should help you resolve the error and successfully validate your Packer template. If you encounter any further issues, please provide additional details, and I can offer more specific assistance.

## Error: My S3 is not mounted properly

The error message indicates that the `s3fs` command is failing to get the IAM credentials for the specified role. This typically happens when there is an issue with the IAM role setup or the instance's access to the IAM role. Here are a few steps to troubleshoot and resolve the issue:

### 1. Verify IAM Role and Policies

Ensure that the IAM role `cellprofilerAWSArka` has the necessary policies attached to it. The role should have permissions to access the S3 bucket. You can check this in the AWS Management Console:

1. **Navigate to IAM**:
   - Open the IAM Dashboard in the AWS Management Console.
   - Click on "Roles" in the left-hand menu.
   - Find and click on the role `cellprofilerAWSArka`.

2. **Check Permissions**:
   - Ensure the role has a policy that allows `s3:ListBucket`, `s3:GetObject`, `s3:PutObject`, etc. Here is an example policy:

```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": [
           "s3:ListBucket",
           "s3:GetObject",
           "s3:PutObject"
```

```
        ],
        "Resource": [
          "arn:aws:s3:::my-example-bucket",
          "arn:aws:s3:::my-example-bucket/*"
        ]
      }
    ]
  }
```

### 2. Attach IAM Role to EC2 Instance

Ensure that the EC2 instance is associated with the correct IAM role:

1. **Navigate to EC2 Dashboard**:
  - Open the EC2 Dashboard in the AWS Management Console.
  - Select the instance you want to check.

2. **Check IAM Role Association**:
  - Click on the "Actions" dropdown menu.
  - Select "Security" and then "Modify IAM role".
  - Ensure that the IAM role `cellprofilerAWSArka` is attached to the instance.

### 3. Instance Metadata Service (IMDS) Configuration

Verify that the instance can access the IMDS (Instance Metadata Service) and retrieve IAM credentials.

1. **Check IMDSv2 Configuration**:
  - Ensure that your EC2 instance can access the metadata service. You can check this by running the following command on your instance:
```bash
curl -H "X-aws-ec2-metadata-token: $(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")" "http://169.254.169.254/latest/meta-data/iam/security-credentials/"
```
  - This should return the name of the IAM role attached to the instance.

### 4. Correct Command Usage

Ensure that you are using the correct syntax and options for the `s3fs` command:

1. **Mount Command**:
```bash
s3fs my-example-bucket /home/ubuntu/bucket -o iam_role=cellprofilerAWSArka -o dbglevel=info -f
```
  Replace `my-example-bucket` with the actual bucket name.

### 5. Additional Debugging

Enable detailed debugging to get more information about what might be going wrong:

1. **Run with Debug Options**:
  ```bash
  s3fs my-example-bucket /home/ubuntu/bucket -o iam_role=cellprofilerAWSArka -o dbglevel=debug -f
  ```
  This will provide more detailed logs which can help in pinpointing the issue.

### Summary

1. **Verify IAM role permissions and policies.**
2. **Ensure the IAM role is correctly attached to the EC2 instance.**
3. **Check that the instance can access the Instance Metadata Service (IMDS).**
4. **Use the correct syntax for the `s3fs` command.**
5. **Enable detailed debugging for more information.**

By following these steps, you should be able to resolve the issue and successfully mount your S3 bucket on the EC2 instance.

### Steps to Resolve:

2. **Update Bucket Policy**:
  Ensure the S3 bucket policy allows access from your IAM role. It might look something like this:

  ```json
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "AWS": "arn:aws:iam::YOUR_AWS_ACCOUNT_ID:role/cellprofilerAWSArka"
        },
        "Action": "s3:*",
        "Resource": [
          "arn:aws:s3:::my-example-bucket",
          "arn:aws:s3:::my-example-bucket/*"
        ]
      }
    ]
  }
  ```

` ` `

3. **Specify the Correct Region in s3fs Command**:
Update your `s3fs` command to include the region option:

```bash
s3fs my-example-bucket /home/ubuntu/bucket -o iam_role=cellprofilerAWSArka -o dbglevel=info -o endpoint=ap-northeast-1 -f
```

4. **Verify Bucket Access**:
Ensure that your bucket is not accidentally blocked. Check the bucket's settings in the AWS Management Console under the Permissions tab.

By following these steps, you should be able to mount your S3 bucket correctly on your EC2 instance. If you continue to experience issues, double-check the role permissions and bucket policies to ensure they are correctly configured.