**Website permission JSON in AWS Bucket**:

|  |
| --- |
| {  "Version": "2012-10-17",  "Statement": [  {  "Sid": "PublicReadGetObject",  "Effect": "Allow",  "Principal": {  "AWS": "\*"  },  "Action": "s3:GetObject",  "Resource": "arn:aws:s3:::YOUR-BUCKET-NAME-HERE/\*"  }  ]  } |

--------------------------------------------------------------------------------------------------------------------------------

**Use AWS S3 bucket as FTP or mount S3 as directory**

S3FS service with AWS S3 and CentOs Linux <https://cloudacademy.com/blog/s3-ftp-server/>

(Or use FileZilla FTP server on WinServer EC2 ($0.05/h): <https://cloudinfrastructureservices.co.uk/how-to-setup-filezilla-ftp-server-windows-2016-on-amazon-aws/> )

-=-=-=-

MySQL DB in AWS <https://aws.amazon.com/getting-started/tutorials/create-mysql-db/?trk=gs_card>

and MS SQL <https://aws.amazon.com/getting-started/tutorials/create-microsoft-sql-db/?trk=gs_card>

--------------------------------------------------------------------------------------------------------------------------------

**AWS RDS MySQL + Debian MySQL client**

AWS RDS -> create database -> MySQL ->

You can use “**Free tier**”

DB instance identifier **database**

Master username **admin**

Master password **Cisco123!**

DB instance class **Burstable classes (includes t classes) -> db.t2.micro**

VPC: choose same VPS you use for SQL client instance

VPC security groups: You need 3306 opened for your SQL client instance

Additional configuration -> Initial database name -> **LabDB**

(If you do not specify a database name, Amazon RDS does not create a database)

**Create**

After creation go to **Connectivity & security ->**

for example, endpoint name is **database-1.cizgiewcuj2y.us-east-1.rds.amazonaws.com**

Go to your SQL client Linux instance console (ssh), install MySQL client (i use debian/ubuntu):

|  |
| --- |
| ubuntu@ip-172-31-90-169:~$ sudo apt-get install mysql-client |

|  |
| --- |
| ubuntu@ip-172-31-90-169:~$ sudo mysql --user admin --password --host database-1.cizgiewcuj2y.us-east-1.rds.amazonaws.com  Enter password:\*\*\*\*\*\*\*  Welcome to the MySQL monitor. Commands end with ; or \g.  ....  mysql> CREATE TABLE LabDB.staff (firstname text, lastname text, phone text);  Query OK, 0 rows affected (0.02 sec)  mysql> INSERT INTO LabDB.staff VALUES ("John", "Smith", "555-1234");  Query OK, 1 row affected (0.00 sec)  mysql> INSERT INTO LabDB.staff VALUES ("Sarah", "Jones", "555-8866");  Query OK, 1 row affected (0.00 sec)  mysql> SELECT \* FROM LabDB.staff WHERE firstname = "Sarah";  +-----------+----------+----------+  | firstname | lastname | phone |  +-----------+----------+----------+  | Sarah | Jones | 555-8866 |  +-----------+----------+----------+  1 row in set (0.01 sec)  mysql> |

**Host migration FROM VMWARE TO AWS**

You can use the AWS Command Line Interface (AWS CLI) to run a VM Import/Export job. Then, a copy of your server is created as an Amazon Machine Image (AMI) and uploaded to an Amazon S3 bucket. You can launch the AMI as an EC2 instance.

For large-scale migrations, AWS Server Migration Service (SMS) automates the migration process

-=-=-=-=-=-=-=-=-=- VM import/export requirements -=-=-=-=-=-=-=-=-=-

<https://docs.aws.amazon.com/vm-import/latest/userguide/vmie_prereqs.html>

--> we will use OVA image format for migration. So, first of all, we prepare OS to convertion to OVA.

for Windows need to be done:

- Install the AWS CLI

- Disable any antivirus and enything that can prevent RDP session to future AWS instance

- Uninstall the VMware Tools from your VMware VM

- Disconnect any CD-ROM drives

- Your source VM must have a functional DHCP client service

- Shut down your VM

- Enable and check rdp client

- Install .NET Framework 4.5 or later on the VM.

>>>>>>>>>>>>>>> check Licensing options, link above <<<<<<<<<<<<<

... If you cannot connect with mstsc to your VM because of security updates and limitation... You know... move limitation =D

REG ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\CredSSP\Parameters /v AllowEncryptionOracle /t REG\_DWORD /d 2

-=-=-=-=-=- Creating OVA fom VMDK and other VMWare stuff -=-=-=-=-

If you use VMWare workstation, use "File -> Export to OVF..." tpo make OVF image from your installation

After it search for app OVFTool (it must be somewhere around VMWare directory), use this app to convert OVF to OVA:

> ovftool "C:\Users\\*\*\*\Documents\Virtual Machines\EXPORT\Windows\_Server\_2012.ovf" "C:\Users\\*\*\*\Documents\Virtual Machines\EXPORT\Windows\_Server\_2012.ova"

Now it is ready for upload to AWS S3.

-=-=-=-=-=-=-=-=- AWS CLI -=-=-=-=-=-=-=-

...I've just created user with next permissions (A LOT OF POWER!!!!! =D):

- IAMFullAccess

- AmazonS3FullAccess

- PowerUserAccess

Install and configure AWS CLI with this user:

|  |
| --- |
| C:\Users\\*\*\*>aws configure  AWS Access Key ID [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ZHKO]: \*\*\*\*\*\*\*\*\*\*6QAL  AWS Secret Access Key [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*G2wJ]: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*oPPs  Default region name [us-east-1]: us-east-1  Default output format [ENTER]: json |

VM Import/Export requires a role to perform certain operations on your behalf. So... just create local file "trust-policy.json" with following content (change S3 bucket name "larx-os-images" to.. whatever you create):

trust-policy.json file:

|  |
| --- |
| {  "Version":"2012-10-17",  "Statement":[  {  "Effect": "Allow",  "Action": [  "s3:GetBucketLocation",  "s3:GetObject",  "s3:ListBucket"  ],  "Resource": [  "arn:aws:s3:::larx-os-images",  "arn:aws:s3:::larx-os-images/\*"  ]  },  {  "Effect": "Allow",  "Action": [  "s3:GetBucketLocation",  "s3:GetObject",  "s3:ListBucket",  "s3:PutObject",  "s3:GetBucketAcl"  ],  "Resource": [  "arn:aws:s3:::larx-os-images",  "arn:aws:s3:::larx-os-images/\*"  ]  },  {  "Effect": "Allow",  "Action": [  "ec2:ModifySnapshotAttribute",  "ec2:CopySnapshot",  "ec2:RegisterImage",  "ec2:Describe\*"  ],  "Resource": "\*"  }  ]  } |

and then execute aws cli (change path for the file tho):

|  |
| --- |
| >aws iam create-role --role-name vmimport --assume-role-policy-document "file://C:\VM\_to\_AWS\trust-policy.json" |

Use the create-role command to create a role named vmimport and grant VM Import/Export access to it. Ensure that you specify the full path to the location of the "trust-policy.json" file that you created in the previous step, and that you include the file:// prefix as shown the following example:

role-policy.json file:

|  |
| --- |
| {  "Version":"2012-10-17",  "Statement":[  {  "Effect": "Allow",  "Action": [  "s3:GetBucketLocation",  "s3:GetObject",  "s3:ListBucket"  ],  "Resource": [  "arn:aws:s3:::larx-os-images",  "arn:aws:s3:::larx-os-images/\*"  ]  },  {  "Effect": "Allow",  "Action": [  "s3:GetBucketLocation",  "s3:GetObject",  "s3:ListBucket",  "s3:PutObject",  "s3:GetBucketAcl"  ],  "Resource": [  "arn:aws:s3:::larx-os-images",  "arn:aws:s3:::larx-os-images/\*"  ]  },  {  "Effect": "Allow",  "Action": [  "ec2:ModifySnapshotAttribute",  "ec2:CopySnapshot",  "ec2:RegisterImage",  "ec2:Describe\*"  ],  "Resource": "\*"  }  ]  } |

create role "vmimport":

|  |
| --- |
| >aws iam put-role-policy --role-name vmimport **--policy-name** **vmimport** --policy-document "file://C:\VM\_to\_AWS\role-policy.json" |

Copy your OVA VM to S3:

|  |
| --- |
| >aws s3 **cp** "C:\Users\\*\*\*\Documents\Virtual Machines\EXPORT\Windows\_Server\_2012.ova" s3://larx-os-images    upload: Documents\Virtual Machines\EXPORT\Windows\_Server\_2012.ova to s3://larx-os-images/Windows\_Server\_2012.ova |

-=-=-=-=-=-=-=-=- Import the VM from S3 to AMI =-=-=-=-=-=-=-=-

https://docs.aws.amazon.com/vm-import/latest/userguide/vmimport-image-import.html

After you upload your VM image file to Amazon S3, you can use the AWS CLI to import the image. These tools accept either the Amazon S3 bucket and path to the file or a URL for a public Amazon S3 file. Private Amazon S3 files require a presigned URL.

Import an OVA

Use the following command to import an image with a single disk.

|  |
| --- |
| >aws ec2 import-image --description "My server VM" --disk-containers "file://C:\VM\_to\_AWS\containers.json" |

The following is an example containers.json file.

|  |
| --- |
| [  {  "Description": "My 2012R2 Server OVA",  "Format": "ova",  "UserBucket": {  "S3Bucket": "larx-os-images",  "S3Key": "Windows\_Server\_2012.ova"  }  }]      Checking:  {  "Description": "My server VM",  "ImportTaskId": "import-ami-08ee9af6ff79363fd",  "Progress": "1",  "SnapshotDetails": [  {  "Description": "My 2012R2 Server OVA",  "DiskImageSize": 0.0,  "Format": "OVA",  "UserBucket": {  "S3Bucket": "larx-os-images",  "S3Key": "Windows\_Server\_2012.ova"  }  }  ],  "Status": "active",  "StatusMessage": "pending"  } |

SEE the "ImportTaskId": "import-ami-08ee9af6ff79363fd"? You can use it for import checking status:

|  |
| --- |
| >aws ec2 describe-import-image-tasks --import-task-ids import-ami-013fd4f65fba717fb  {  "ImportImageTasks": [  {  "Description": "My server VM",  "ImportTaskId": "import-ami-013fd4f65fba717fb",  "Platform": "Windows",  "Progress": "27",  "SnapshotDetails": [  {  "DeviceName": "/dev/sdf",  "DiskImageSize": 6002596864.0,  "Format": "VMDK",  "Status": "completed",  "UserBucket": {  "S3Bucket": "larx-os-images",  "S3Key": "Windows\_Server\_2012.ova"  }  }  ],  "Status": "active",  "StatusMessage": "updating",  "Tags": []  }  ]  } |

On the AWS:

launch new AMI in EC2:

- create VPC (like 192.168.0.0/16)

- create IGW (Internet gateway) and attach it to VPC

- create subnet insede the VPC(like 192.168.1.0/24)

- fix Routing table if needed (check fop 0.0.0.0/0 to IGW)

Start the instance!!!!!

------------------------------------------

just in case

<https://aws.amazon.com/premiumsupport/knowledge-center/import-server-ec2-instance/>

---------------------------------------------------------------------------------------------------------------------------------------

**AWS EC2 Bootstrapping**

Linux : /var/log/cloud-init-output.log

Windows 2016: C:\ProgramData\Amazon\EC2-Windows\Launch\UserdataExecution.log

Windows 2012: C:\Program Files\Amazon\Ec2ConfigService\Logs\Ec2ConfigLog.txt

Windows 2008: C:\Program Files\Amazon\Ec2ConfigService\Logs\Ec2ConfigLog.txt

#!/bin/bash

sudo apt update && sudo apt install -y nginx

sudo systemctl start nginx

---------------------------------------------------------------------------------------------------------------------------------------

**To access your instance:**

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (My-AWS-KeyPair.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

**sysadmin@localhost:~$** chmod 400 My-AWS-KeyPair.pem

1. Connect to your instance using its Public DNS:

*ec2-34-207-196-208.compute-1.amazonaws.com* **(example!)**

For instance:

**sysadmin@localhost:~**$ ssh -i /home/alec/Desktop/My-AWS-KeyPair.pem ec2-user@34.207.196.208

**BOOTSTRAPING example**

[**ec2-user@ip-172-31-32-14** ~]$ sudo su

[**root@ip-172-31-32-14 ec2-user**]# yum update -y

yum install -y httpd.x86\_64

echo “Install httpd (Apache)”

systemctl start httpd.service

systemctl enable httpd.service

Echo “checking Apache service localy”

curl localhost:80

echo “Hello world from $(hostname -f) > /var/www/html/index.html”

In Windows, when You use **Putty**, you have to convert **.pem** key to **.ppk** (Putty format).

Then you can use Putty to connect to yours AWS instance:



**EBS**

# **Making an Amazon EBS Volume Available for Use on Linux**

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-using-volumes.html>

**How-to: format and mount an EBS volume on Linux**

1. Connect to your instance using SSH.
2. Use the **lsblk** command to view your available disk devices and their mount points

[ec2-user @IP-172-31-2-111 ~]$ *lsblk*

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

xvda 202:0 0 8G 0 disk

└─xvda1 202:1 0 8G 0 part /

xvdb 202:16 0 4G 0 disk

The root device is /dev/xvda. The attached volume is /dev/xvdf, which is not yet mounted.

Use the **file -s** command to get information about a device, such as its file system type. If the output shows simply data, as in the following example output, there is no file system on the device and you must create one.

[ec2-user ~]$ sudo file -s */dev/xvdb*

/dev/xvdb: data

If you have an empty volume, use the mkfs -t command to create a file system on the volume.

[ec2-user ~]$ sudo mkfs -t xfs */dev/xvdb*

Use the mkdir command to create a mount point directory for the volume AND mount the volume at the directory you created:

[ec2-user ~]$ sudo mkdir */data*

[ec2-user ~]$ sudo mount */dev/xvdb* */data*

[ec2-user @IP-172-31-2-111 ~]$ *lsblk*

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

xvda 202:0 0 8G 0 disk

└─xvda1 202:1 0 8G 0 part /

xvdb 202:16 0 4G 0 disk /data

To mount an attached volume automatically after reboot:

[ec2-user ~]$ sudo cp /etc/fstab /etc/fstab.orig

[ec2-user ~]$ sudo nano /etc/fstab

UUID=*aebf131c-6957-451e-8d34-ec978d9581ae* */data* *xfs* defaults, nofa$

# ADD LINE:

/dev/xvdb /data ext4 defaults, nofail 0 2