

USE CASE – 1

Question: -

In a multi-tier architecture as shown below the EC2 in the Public Subnet hosts front end applications like web applications and RDS/EC2 instances are hosted in the Private Subnet. This way only the web application can be accessed by the public and the RDS/EC2 in the private subnet cannot be accessed by the public.

How should we start: -

So, how do we manage the EC2/RDS in the private subnet. For ex., creating tables and populating with data, giving permissions to tables, updating the OS with the latest patches.

One way is to use the bastion host, which is not that reliable the bastion host has a public IP and hackers can try to exploit this.

A much-preferred approach is to use VPN. In this use case we will explore how to setup a Client VPN in AWS and try to connect to the resources in the private subnet for the sake of maintenance.

Important note: -

Note that all the entire setup is done in us-east-1 (North Virginia) and the commands reflect the same. The lab can be done in some other region also, but the commands need to be modified.

How to start: -

1. we would be setting up a VPN Endpoint in the Default VPC. And create a MyAppVPC along with an EC2 instance in the private subnet.
2. Finally, we would be establishing a VPN connection to the VPN Endpoint and connect to the EC2 instance using the private IP address from our Laptop.
3. Along the way we also need to setup a Peering connection across the two VPCs.

Follow Along: -

USE CASE – 1

- **Step: - 1**

Go to the IAM Screen and create an IAM Role as shown below.

Review

Provide the required information below and review this role before you create it.


Role name* RoleEC2Adminaccess

Use alphanumeric and '+=, @-_' characters. Maximum 64 characters.

Role description Allows EC2 instances to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=, @-_' characters.

Trusted entities AWS service: ec2.amazonaws.com

Policies  AdministratorAccess [↗](#)

Permissions boundary Permissions boundary is not set

- **Step: - 2**

Create an Ubuntu EC2 in the Default VPC and assign it a Role with Administrator Access Policy. "t2.micro" instance type should be good enough and make sure that the port 22 is allowed in the Security Group inbound rules.

Name the EC2 as For Certificate Generation as this EC2 will be used for the VPN Certificate generation purpose only.


<input checked="" type="checkbox"/>	Name ▾	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability Zon
<input checked="" type="checkbox"/>	ForCertifi... ↗	i-0bdc2eae8e02b8539	Running 🔍	t2.micro	–	No alarms +	us-east-1a

USE CASE – 1

- **Step: - 3**

Login to the EC2 via the Putty or some other SSH client using the Public IP Address of the EC2 instance.

 ubuntu@ip-172-31-29-12: ~

ubuntu@ip-172-31-29-12:~\$ 

USE CASE – 1

- **Step: -4**

Install the AWS CLI, by executing the below commands in the Putty session. Finally, run the aws command as shown below to make sure that the aws command is there in the PATH.

```
sudo apt-get update
sudo apt-get install python2.7 python-pip
pip install awscli --upgrade
mkdir .aws
echo -e "[default]\nregion=us-east-1" > .aws/config
export PATH="$PATH:/home/ubuntu/.local/bin/"
```

- **Step: - 5**

Clone the Easy RSA Git Repo using the below command-

```
git clone https://github.com/SanketVyas0809/aws-handson
cd easy-rsa/easyrsa3
```

Initialize the PKI using the below command.

```
./easyrsa init-pki
```

```
ubuntu@ip-172-31-29-12:~$
ubuntu@ip-172-31-29-12:~$
ubuntu@ip-172-31-29-12:~$
ubuntu@ip-172-31-29-12:~$
ubuntu@ip-172-31-29-12:~$ git clone https://github.com/OpenVPN/easy-rsa.git
Cloning into 'easy-rsa'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 2082 (delta 0), reused 2 (delta 0), pack-reused 2077
Receiving objects: 100% (2082/2082), 11.72 MiB | 21.85 MiB/s, done.
Resolving deltas: 100% (913/913), done.
ubuntu@ip-172-31-29-12:~$ cd easy-rsa/easyrsa3
ubuntu@ip-172-31-29-12:~/easy-rsa/easyrsa3$ ./easyrsa init-pki

init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /home/ubuntu/easy-rsa/easyrsa3/pki
```

USE CASE – 1

- **Step: - 6**

Build Certificate Authority using the below command. When prompted enter the common name as thecloudavenue.com. Anything can be used, but the same has to be reflected in the upcoming commands also.

```
./easyrsa build-ca nopass
```

- **Step: - 7**

Build the Server Certificate using the below command.

```
./easyrsa build-server-full thecloudavenue.com nopass
```

```
ubuntu@ip-172-31-29-12:~/easy-rsa/easyrsa3$ ./easyrsa build-ca nopass
Using SSL: openssl OpenSSL 1.1.1 11 Sep 2018
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:thecloudavenue.com

CA creation complete and you may now import and sign cert requests.
Your new CA certificate file for publishing is at:
/home/ubuntu/easy-rsa/easyrsa3/pki/ca.crt

ubuntu@ip-172-31-29-12:~/easy-rsa/easyrsa3$ ./easyrsa build-server-full thecloudavenue.com nopass
Using SSL: openssl OpenSSL 1.1.1 11 Sep 2018
Generating a RSA private key
.....+++++
...+++++
writing new private key to '/home/ubuntu/easy-rsa/easyrsa3/pki/easy-rsa-8435.iBsHTf/tmp.zAwFkG'
-----
Using configuration from /home/ubuntu/easy-rsa/easyrsa3/pki/easy-rsa-8435.iBsHTf/tmp.Zx7uv9
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName      :ASN.1 12:'thecloudavenue.com'
Certificate is to be certified until May  6 14:55:20 2023 GMT (825 days)

Write out database with 1 new entries
Data Base Updated

ubuntu@ip-172-31-29-12:~/easy-rsa/easyrsa3$ █
```

- **Step: - 8**

Build the Client Certificate using the below command.

```
./easyrsa build-client-full sanket.thecloudavenue.com nopass
```

USE CASE – 1

- **Step: - 9**

Move the certificates and the keys to a folder called acm using the below commands.

```
mkdir acm
cp pki/ca.crt acm
cp pki/issued/thecloudavenue.com.crt acm
cp pki/issued/sanket.thecloudavenue.com.crt acm
cp pki/private/thecloudavenue.com.key acm
cp pki/private/sanket.thecloudavenue.com.key acm
cd acm
```

```
rm: cannot remove 'dir': No such file or directory
rm: cannot remove 'acm': Is a directory
ubuntu@ip-172-31-29-12:~/easy-rsa/easyrsa3$ cp pki/ca.crt acm
ubuntu@ip-172-31-29-12:~/easy-rsa/easyrsa3$ cp pki/issued/thecloudavenue.com.crt acm
ubuntu@ip-172-31-29-12:~/easy-rsa/easyrsa3$ cp pki/issued/sanket.thecloudavenue.com.crt acm
ubuntu@ip-172-31-29-12:~/easy-rsa/easyrsa3$ cp pki/private/thecloudavenue.com.key acm
ubuntu@ip-172-31-29-12:~/easy-rsa/easyrsa3$ cp pki/private/sanket.thecloudavenue.com.key acm
ubuntu@ip-172-31-29-12:~/easy-rsa/easyrsa3$ cd acm
ubuntu@ip-172-31-29-12:~/easy-rsa/easyrsa3/acm$ ls -l
total 40
-rw-r----- 1 ubuntu ubuntu 1233 Jan 31 15:05 ca.crt
-rw-r----- 1 ubuntu ubuntu 4570 Jan 31 15:06 sanket.thecloudavenue.com.crt
-rw-r----- 1 ubuntu ubuntu 1704 Jan 31 15:06 sanket.thecloudavenue.com.key
-rw-r----- 1 ubuntu ubuntu 4571 Jan 31 14:33 sripati.thecloudavenue.com.crt
-rw-r----- 1 ubuntu ubuntu 1704 Jan 31 14:35 sripati.thecloudavenue.com.key
-rw-r----- 1 ubuntu ubuntu 4699 Jan 31 15:05 thecloudavenue.com.crt
-rw-r----- 1 ubuntu ubuntu 1708 Jan 31 15:06 thecloudavenue.com.key
```

- **Step:- 10**

Import the Certificates into the AWS Certificate Manager using the below commands.

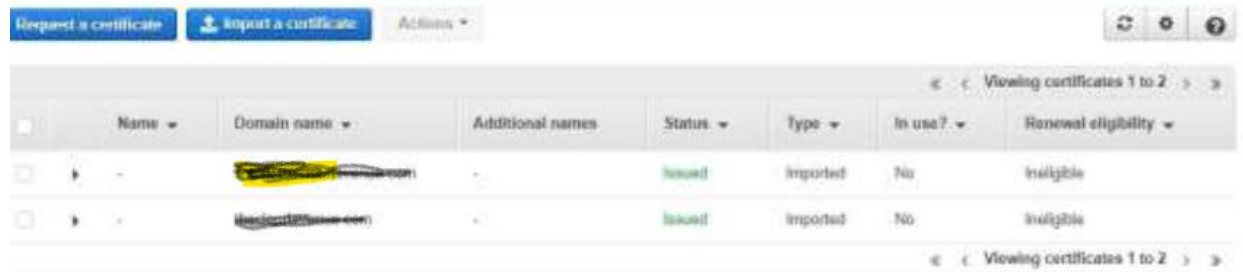
```
aws acm import-certificate --certificate fileb://thecloudavenue.com.crt --
private-key
fileb://thecloudavenue.com.key --certificate-chain fileb://ca.crt --region
us-east-1
```

```
aws acm import-certificate --certificate
fileb://sripati.thecloudavenue.com.crt --
private-key fileb://sanket.thecloudavenue.com.key --certificate-chain
fileb://ca.crt --
region us-east-1
```

USE CASE – 1

- **Step: - 11**

Go to the AWS Certificate Manager Console and the imported Certificates should appear as shown below.



The screenshot shows the AWS Certificate Manager console. At the top, there are buttons for 'Request a certificate', 'Import a certificate', and an 'Actions' dropdown. Below these is a table of certificates. The table has columns for Name, Domain name, Additional names, Status, Type, In use?, and Renewal eligibility. Two certificates are listed, both with a status of 'Issued' and type of 'Imported'.




	Name	Domain name	Additional names	Status	Type	In use?	Renewal eligibility
<input type="checkbox"/>	mydemoendpoint	mydemoendpoint.com		Issued	Imported	No	Ineligible
<input type="checkbox"/>	mydemoendpoint	mydemoendpoint.com		Issued	Imported	No	Ineligible

- **Step: - 12**

In the VPC Management Console go to “Client VPN Endpoints” and click on “Create Client VPN Endpoints”.

Create Client VPN Endpoint

Create a new Client VPN endpoint to enable clients to access networks over a TLS VPN session

Name Tag	<input type="text" value="mydemoendpoint"/>	
Description	<input type="text"/>	
Client IPv4 CIDR*	<input type="text" value="20.0.0.0/18"/>	

- **Step: - 13**

Enter the name, the “Client IPv4 CIDR”, select the “Server certificate ARN”

(thecloudavenue.com), “Use mutual authentication” and finally select the “Client certificate ARN” (sanket.thecloudavenue.com).

USE CASE – 1

- **Step: - 14**

For the “Connection Logging” select No. Enable the “split-tunnel” and finally select the VPC.

The screenshot displays the 'Other optional parameters' section of the AWS Client VPN configuration console. It includes the following fields and options:

- Connection Logging:** A section header.
- Do you want to log the details on client connections?:** Radio buttons for 'Yes' and 'No'. The 'No' option is selected.
- DNS Server 1 IP address:** An empty text input field.
- DNS Server 2 IP address:** An empty text input field.
- Transport Protocol:** Radio buttons for 'TCP' and 'UDP'. The 'UDP' option is selected.
- Enable split tunnel:** A checkbox that is checked.
- VPC ID:** A dropdown menu showing 'vpc-c128a3b6'.
- Security Group IDs:** A text input field containing 'sg-e61fcb09'.
- Select security groups:** A button with a downward arrow.

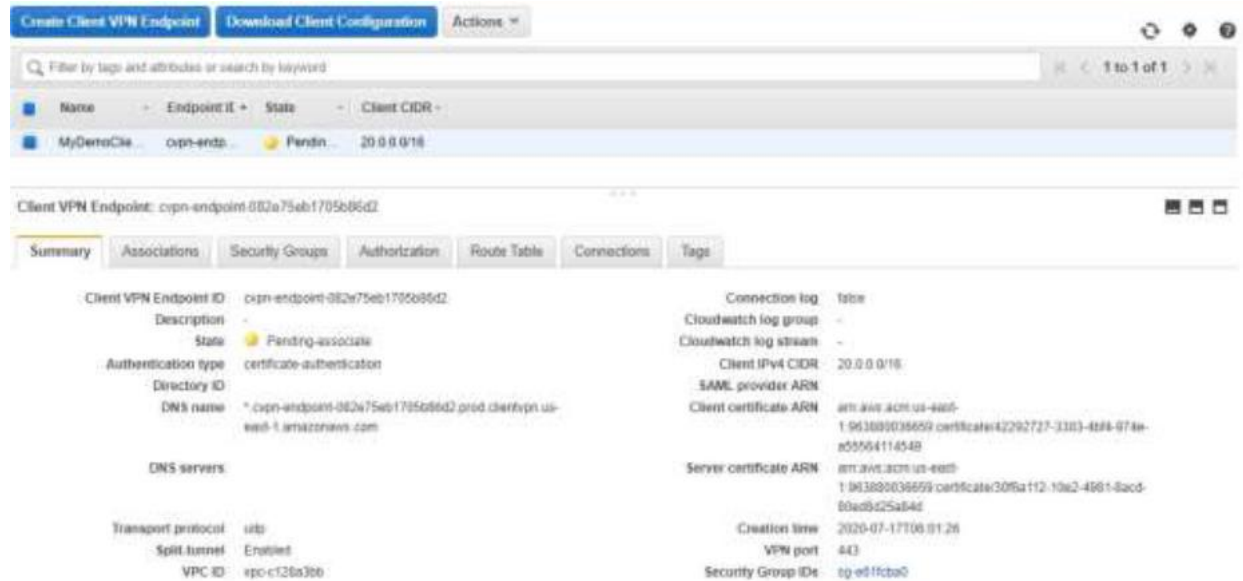
- **Step: - 15**

Click on “Create Client VPN Endpoint”.

USE CASE – 1

- **Step: - 16**

The “Client VPN Endpoint” will be created as shown below. Initially it would be in a “Pending-associate” state for it to be associated with a Subnet.



- **Step: -17**

Click on the Associations tab and the Associate button.

- **Step: - 18**

Select the default VPC and choose any of the Subnet. It doesn't matter which Subnet is selected. Click on Associate.

- **Step: -19**

It will take about 5 minutes for the Client VPN Endpoint to become in the Available State and the Association to the Subnet in an Associated state.

- **Step: -20**

In the same VPC Console, go to the “Elastic IPs” screen and click on “Allocate Elastic IP Address”. The same is required for the sake of VPC Peering which we would be creating later.

USE CASE – 1

- **Step: -21**

Click on Allocate to assign an Elastic IP address as shown below.

- **Step: - 22**

Now that the Elastic IP has been created, it's time to create a new VPC. Go to the "VPC Dashboard" and click on "Launch VPC Wizard".

- **Step: - 23**

Select the "VPC with Public and Private Subnets" option.



- **Step: -24**

Give the VPC a name and select the Elastic IP created earlier. Rest of the default options are good enough. Click on "Create VPC". It will take a few minutes for the VPC to be created.

- **Step: - 25**

We will notice that there are two VPCs. One is the default and the other one is MyAppVPC which we created a few minutes back.

Make sure to note the "IPv4 CIDR" for both the VPCs.

USE CASE – 1

- **Step: - 26**

Go to the EC2 Console and create and Ubuntu EC2 in the MyAppVPC and the Private Subnet. In the Configure Instance screen make sure to select the right VPC and the Subnet as shown below.

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1	Launch into Auto Scaling Group
Purchasing option	<input type="checkbox"/> Request Spot instances	
Network	vpc-011129096c3c562961 MyAppVPC Create new VPC	
Subnet	subnet-0f7e51fe081c9b2d1 PrivateSubnet-us-east-1 Create new subnet 251 IP Addresses available	
Auto-assign Public IP	Use subnet setting (Disable)	
Placement group	<input type="checkbox"/> Add instance to placement group	
Capacity Reservation	Open Create new Capacity Reservation	
IAM role	None Create new IAM role	
Shutdown behavior	Stop	
Stop - Hibernate behavior	<input type="checkbox"/> Enable hibernation as an additional stop behavior	
Enable termination protection	<input type="checkbox"/> Protect against accidental termination	
Monitoring	<input type="checkbox"/> Enable CloudWatch detailed monitoring	

- **Step: -27**

Also make sure to create a new Security Group to allow the SSH traffic as shown below.

- **Step: -28**

Name the EC2 as "MyAppVPC-PrivateSubnet". This EC2 has only the Private IP and not the Public IP as seen below. And this EC2 is primarily used for backend applications like Databases, for which the end users/customers don't have access to because of the lack of Public IP.

The whole exercise is about connecting to this EC2 from outside the Cloud, like from our Laptop for the sake of maintenance.

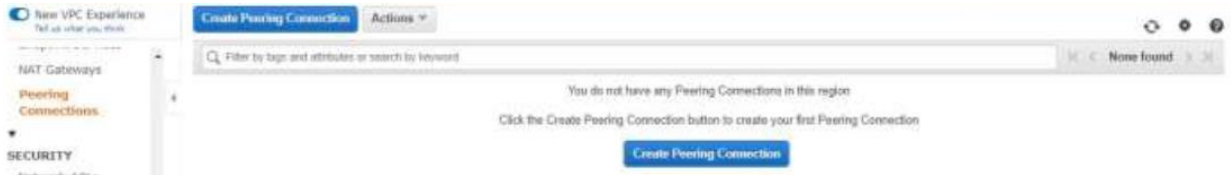
- **Step: -29**

We need to create a Peering connection between the Default and the MyAppVPC. In the VPC Management Console, go to the "Peering Connections" and click on "Create Peering Connection".

USE CASE – 1

• **Step: -30**

Give the Peering connection a name and select the Default VPC as the VPC(Requester) and the MyAppVPC as the VPC (Acceptor). Rest of the options should be fine. Click on “Create Peering Connection”.



• **Step: -31**

Initially the Peering Connection would be in a Pending Acceptance State. Go to the Actions option and select the Accept option. In a few seconds, the state of the Peering connection will change to Active.

• **Step: -32**

Now that the VPC Peering has been setup, the routing table has to be updated for the two VPCs to be able to communicate to each other. For updating the routing table, the below steps have to be repeated for both the EC2s. The order of the EC2 doesn't matter. Make sure to select one of the EC2 and click on the Subnet ID in the description to go to the Subnet screen.



USE CASE – 1

- **Step: -34**

In the Subnet Screen, click on the "Route Table" link to go the "Route Table" properties.

- **Step: -35**

Click on the Routes tab and click on "Edit routes".

- **Step: - 36**

Notice that there is a route for one of the VPC CIDR. Click on "Add route" and add the CIDR for the other VPC. For the Target select the Peering Connection and finally the VPC peering created earlier. Click on "Save routes".

Note that all the above steps from selecting the EC2 has to be done for other EC2 also. This completes the Peering process across the VPCs.

- **Step: -37**

In the VPC Management Console, go to the "Client VPN Endpoints" and select the Authorization tab. Click on "Authorize Ingress".

- **Step: -38**

Enter the CIDR of MyAppVPC and some description. Click on "Add authorization rule".



The screenshot shows the 'Add authorization rule' form in the AWS VPC Management Console. The form is titled 'Add authorization rule' and has a subtitle 'Add authorization rules to grant clients access to the networks'. The 'Client VPN Endpoint ID' field is populated with 'cvpn-endpoint-082e75eb1705b86d2'. The 'Destination network to enable' field is a text input containing '10.0.0.0/16'. The 'Grant access to:' section has two radio buttons: 'Allow access to all users' (which is selected) and 'Allow access to users in a specific access group'. The 'Description' field is a text input containing 'Allow to MyAppVPC'. There are information icons (i) next to the 'Destination network to enable' and 'Description' fields.

- **Step: - 39**

USE CASE – 1

In a few seconds the state of the Authorization will be Active.

- **Step: -40**

Click on the "Route Table tab". Click on "Create Route".

- **Step: -41**

Give the CIDR for the MyAppVPC, Select the Target VPC Subnet ID and give some description. Click on "Create Route".

- **Step: -42**

The "Route Table" would be updated as shown below.

- **Step: -43**

Click on "Download Client Configuration" to download the downloaded-clientconfig.ovpn. It's a simple text file and need to be edited later.

- **Step: -44**

Copy the sanket.thecloudavenue.com.key and sanket.thecloudavenue.com.crt files from the first EC2 instance where the certificates and keys were generated. The same can be done by doing a cat on the file in the Putty session and copying the content or using a tool like WinSCP. Make sure to copy the content carefully without missing anything.

Open the downloaded-client-config.ovpn in a notepad. Add a random string before the VPN endpoint. Like the way "abcd." has been below. Also, provide the cert and key paths as shown below towards the end, make sure to specify the proper path. This file has all the details for us to establish a VPN connection.

- **Step: - 45**

Download and install the AWS Client VPN
(<https://aws.amazon.com/vpn/clientvpn-download/>).

USE CASE – 1

- **Step: - 46**

Start the AWS VPN Client.

On the File Menu, select "Manage Profiles" and click on "Add Profile".

- **Step: - 47**

Give the Profile a Display Name and Point to the VPN Configuration file and click on "Add Profile".

Click on Connect and it takes a few seconds for the VPN connection to be established.

Finally, the connection to the VPN should be established and the Connect button should be changed to Disconnect as the connection has been established.

- **Step: - 48**

In the "Client VPN Endpoints", go to the Connections tab and an active connection should be displayed.

Click on Connect and it takes a few seconds for the VPN connection to be established.

- **Step: -49**

Go back to the EC2 Console. Select the EC2 with the name "MyAppVPCPrivateSubnet" and grab the Private IP.

- **Step: -50**

Use the Putty Session to connect to this EC2 via Private IP. We should be able to connect as we have established a VPN connection to the AWS Cloud.

- **Step: -51**

Below is the Putty Session for the same.

USE CASE – 1

- **Clearing the session:** -

The following AWS resources have to be cleaned up in the same order.

- Disconnect from the AWS VPN Client
- Terminate the EC2 instances
- Delete the VPC Peering connections
- Delete the NAT Gateway from the VPC Console
- Wait for a few Minutes
- Release the Elastic IP
- Finally delete the VPC (MyAppVPC)
- Disassociate the Subnet association to the Client VPN Endpoint
- Delete the Client VPN Endpoint
- Delete the Certificates in Certification Manager
- Delete the Role in the IAM which has been created for the EC2