

Using **AWS Cloud Formation** to create a simple client-side VPN.

So what is cloud Formation actually?

AWS cloud Formation is one of the many services available in aws to streamline user interaction with various other services within the AWS ecosystem. It helps the client to provision, set-up, configure the resources(Ex: EC2 instance, RDB instance etc..) and policies attached automatically with ease from a pre-defined template that defined all the said specifications, so that the client can focus on actually focus on maintaining and improving the resources than actually spend resources and time on deploying them individually.

Understanding the use-case:

For a scalable web application with a backend database, you typically use an Auto Scaling group, Elastic Load Balancing, and Amazon RDS. Manually provisioning and configuring these resources can be complex and time-consuming.

Instead, you can use a CloudFormation template to streamline this process. A template defines all your resources and their properties. When you create a CloudFormation stack from the template, CloudFormation provisions and configures the Auto Scaling group, load balancer, and database for you. This approach allows you to manage your resources as a single unit, simplifying deployment and maintenance. You can also easily delete the stack, which removes all associated resources. CloudFormation enables quick replication and efficient management of your infrastructure.

How cloud formation works?

Cloud formation consists of two main key concepts:

- . Templates
- . Stacks

When creating a stack, CloudFormation makes service calls to AWS to provision and configure resources, acting within your permission limits. For instance, to create or terminate EC2 instances, you need corresponding permissions, managed via AWS IAM.

The template defines all resource actions. For example, a template specifying a t2.micro EC2 instance prompts CloudFormation to call the EC2 create instance API with that instance type.



Example of how a JSON template looks like :

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "A simple EC2 instance",
    "Resources": {
        "MyEC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-0ff8a91507f77f867",
                "InstanceType": "t2.micro"
            }
        }
    }
}
```

Explantion:

{

"AWSTemplateFormatVersion": "2010-09-09",

. This line specifies the version of the AWS CloudFormation template format you are using.
"2010-09-09" is the latest version as of this writing.

"Description": "A simple EC2 instance",

. This line provides a brief description of what the CloudFormation stack will do. In this case, it describes the creation of a simple EC2 instance.

"Resources": {

This line begins the section where you define the AWS resources that CloudFormation will create

"MyEC2Instance": {

. This line specifies a logical name for the resource within the template. "MyEC2Instance" is a unique identifier for this EC2 instance in the template.

"Type": "AWS::EC2::Instance",

. This line defines the type of AWS resource to create. Here, "AWS::EC2::Instance" specifies that the resource is an EC2 instance.

"Properties": {

. This line starts the section where you define properties for the EC2 instance.

"ImageId": "ami-0ff8a91507f77f867",

. This line specifies the Amazon Machine Image (AMI) ID that the EC2 instance will use. The AMI ID "ami-0ff8a91507f77f867" refers to a specific Amazon Linux AMI.

"InstanceType": "t2. micro"

. This line specifies the instance type for the EC2 instance. "t2.micro" is a small instance type suitable for low-traffic applications and testing.

Creating a client side vpn from pre-defined template:

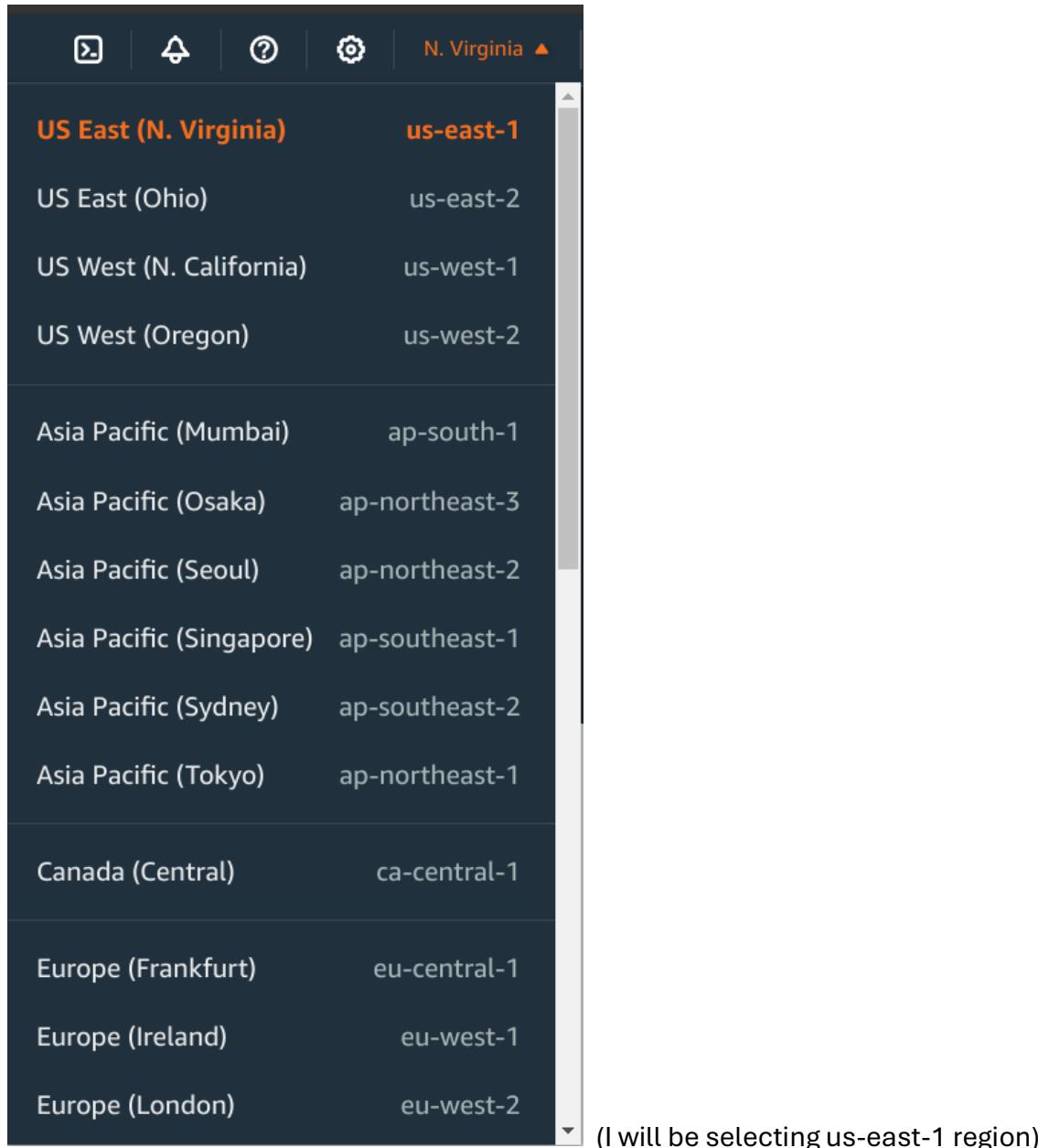
(Step-1):

Home page of AWS cloud formation:

The screenshot shows the AWS CloudFormation home page. The left sidebar includes links for Stacks, StackSets, Exports, Application Composer (New), Registry (Public extensions, Activated extensions, Publisher), Spotlight, and Feedback. The main content area features the title "AWS CloudFormation" and the subtitle "Model and provision all your cloud infrastructure". It explains that CloudFormation provides a common language to describe and provision infrastructure resources. Below this is a "How it works" section with a callout about simplifying infrastructure management using AWS. To the right, there's a "Create a CloudFormation stack" section with a "Create stack" button, and a "Getting started" sidebar with links for "What is AWS CloudFormation", "Getting started with CloudFormation", "Learn template basics", and "Quick starts". The URL at the bottom is us-east-1.console.aws.amazon.com/console/home?... and the page footer includes links for Privacy, Terms, and Cookie preferences.

(Step-2)

Region selection:



(Step-3):

Stacks are created using *Templates*, which can be done in three ways:

1. Using a pre-defined template or
2. Create a scratch template using *application composer*

We will be using a pre-defined template, loaded from an amazon S3 URL. (*s3*, stands for simple storage service).

The template is loaded from <https://learn-cantrill-labs.s3.amazonaws.com/aws-client-vpn/A4LVPc.yaml>.

(Step-4):

Click next.

Stack name: sreeram

VPC CIDR: 10.16.0.0/16

AZA: RESERVEDACIDR

The screenshot shows the AWS CloudFormation template configuration interface. It displays several parameters:

- IP Address range for A4L-SN-DB-C Subnet**: 10.16.144.0/20
- PRIVCCIDR**: IP Address range for A4L-SN-PRIV-C Subnet (10.16.160.0/20)
- PUBCCIDR**: IP Address range for A4L-SN-PUB-C Subnet (10.16.176.0/20)
- AddSSMEndpoints**: Create SSM Endpoints for public and private instance management (true)
- EnableIPv6**: Enable IPv6 on the VPC and subnets (true)
- LatestAmiId**: AMI for any EC2 Instances (if applicable) (default is latest AlmaLinux2) (/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2)
- PublicIPv4Subnets**: Create Public IPv4 Subnets? (true)

At the bottom, there are links for CloudShell, Feedback, and a footer with copyright information.

As seen, all of the required resources and policies have been defined by the sample template.

Authorizing IAM role for access and permissions:

The screenshot shows the 'Create Stack' step in the AWS CloudFormation console. On the left, a sidebar lists options like Stacks, StackSets, Exports, Application Composer, Registry, Spotlight, and Feedback. The main area contains the following fields:

- Create Public IPv4 Subnets?**: true
- Permissions - optional**: Specify an existing AWS Identity and Access Management (IAM) service role that CloudFormation can assume.
- IAM role - optional**: Choose the IAM role for CloudFormation to use for all operations performed on the stack. A dropdown shows "Sample-role-name".
- Capabilities**: A box containing a note about IAM resources and a checked checkbox: **I acknowledge that AWS CloudFormation might create IAM resources.**

At the bottom right are buttons for **Cancel**, **Create change set**, and **Create stack**.

Click **Create Stack**.

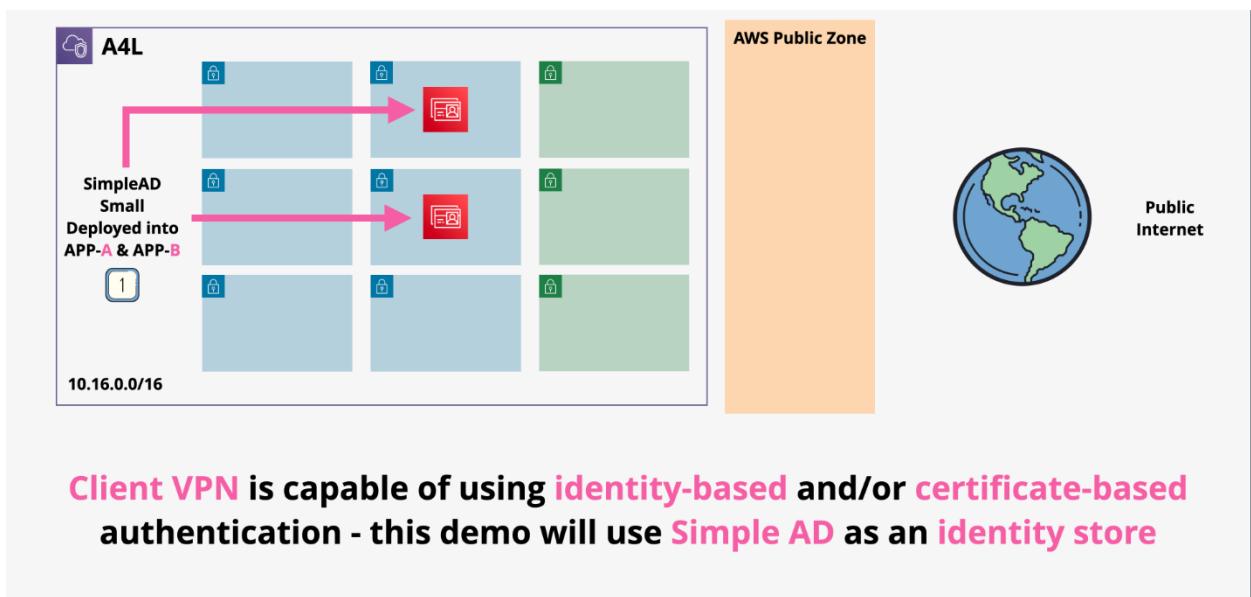
(Step-5):

Wait till the status changed from **CREATE_IN_PROGRESS** to **CREATE_COMPLETE**.

The screenshot shows the AWS CloudFormation console. On the left, the navigation pane includes 'CloudFormation', 'Stacks', 'Drifts', 'StackSets', 'Exports', 'Application Composer', 'IaC generator', 'Registry', 'Public extensions', 'Activated extensions', 'Publisher', and 'Spotlight'. The main area shows a stack named 'A4L' with a creation timestamp of '2024-07-09 21:57:52 UTC+0530'. The 'Events' tab is selected, displaying a table of events:

Timestamp	Logical ID	Status	Detailed status
2024-07-09 22:00:28 UTC+0530	A4L	CREATE_COMPLETE	-
2024-07-09 22:00:27 UTC+0530	SessionManagerInstanceProfile	CREATE_COMPLETE	-
2024-07-09 21:59:25 UTC+0530	ssmec2messagesinterfaceendpoint	CREATE_COMPLETE	-
2024-07-09 21:59:25 UTC+0530	ssminterfaceendpoint	CREATE_COMPLETE	-

Stage:1



We will be using AWS directory service in simple AD mode. AWS Directory Service in Simple AD mode refers to a specific configuration of AWS Directory Service that provides basic Active Directory (AD) features without requiring a full Microsoft AD setup. It is designed to be simple to set up and manage, making it suitable for smaller organizations or less complex use cases.

(Step-6)

Choose AWS Directory service.

The screenshot shows the AWS Directory Service landing page. On the left, there's a sidebar with 'Active Directory' and 'Cloud Directory' sections. The main area features a large heading 'Directory Service' with the subtext 'helps you store information and manage access to resources.' Below this, there's a paragraph about migrating Active Directory-dependent applications to the AWS Cloud. A callout box titled 'Set up directory' says 'Choose which directory best fits your business needs and we'll walk you through how to set it up.' It has a dropdown menu set to 'AWS Managed Microsoft AD' and a 'Set up directory' button. At the bottom right of the main area, there's a 'Getting started' button.

Select simple AD in the drop down menu:

This screenshot shows the 'Select directory type' step of the AWS Directory Service setup wizard. The left sidebar lists steps 1 through 4. Step 1 is 'Select directory type', which is currently active. Step 2 is 'Enter directory information', Step 3 is 'Choose VPC and subnets', and Step 4 is 'Review & create'. The main content area is titled 'Select directory type' with a 'Info' link. It shows a 'Directory types' section with four options: 'AWS Managed Microsoft AD' (radio button), 'Simple AD' (radio button, selected), 'AD Connector', and 'Amazon Cognito User Pools'. To the right of this is a 'Simple AD' section with the text: 'Simple AD is a standalone managed directory that is powered by a Linux-Samba Active Directory-compatible server.' and a 'Learn more' link. At the bottom right are 'Cancel' and 'Next' buttons.

Directory information:

Screenshot of the AWS Directory Service 'Set up a directory' wizard Step 1: Select directory type.

The page shows the following steps:

- Step 1: Select directory type
- Step 2: Enter directory information
- Step 3: Choose VPC and subnets
- Step 4: Review & create

Enter directory information

Directory information Simple AD is managed Samba 4 Active Directory Compatible Server hosted on the AWS cloud and provides a subset of Microsoft Active Directory capabilities.

Directory type Simple AD

Directory size Simple AD is available in the following two sizes:

- Small**
~USD 36.000/mo (USD 0.0500/hr)
Small directories can have up to 2000 objects, including ~500 users, groups and computers each.
- Large**
~USD 108.000/mo (USD 0.1500/hr)
Large directories can have up to 20,000 objects, including ~5000 users, groups and computers each.

Directory DNS name A fully qualified domain name. This name will resolve inside your VPC only. It does not need to be publicly resolvable.
directory.animals4life.org

Directory NetBIOS name - optional

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot of the AWS Directory Service 'Set up a directory' wizard Step 2: Enter directory information.

The page shows the following fields:

- Directory NetBIOS name - optional**: CORP
A short identifier for your domain. If you do not specify a NetBIOS name, it will default to the first part of your Directory DNS name.
- Default administrative user**: Administrator
- Administrator password**: The password for the default administrative user named Administrator.
Passwords must be between 8 and 64 characters and include three of these four categories: lowercase, uppercase, numeric, and special characters.
- Confirm password**: This password must match the Administrator password above.
- Directory description - optional**: Directory service for client side vpn demo
Descriptive text that appears on the details page after the directory has been created.

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Choosing vpc and subnets:

The screenshot shows the AWS Directory Service 'Set up a directory' wizard. The current step is 'Step 3: Choose VPC and subnets'. The 'Networking' section displays the VPC and subnets assigned to the directory. A VPC dropdown shows 'A4L-VPC | vpc-05909905da0b44e27 (10.16.0.0/16)'. Subnets listed are 'A4L-SN-PRIV-A | subnet-087501a309b11e933 (10.16.32.0/20, us-east-1a)' and 'A4L-SN-PRIV-B | subnet-00945e47b073e9362 (10.16.96.0/20, us-east-1b)'. Below this, there's an 'Initial AD site name' field set to 'Default-First-Site-Name'. Navigation buttons at the bottom include 'Cancel', 'Previous', 'Next', and 'Create'.

Directories (1) [Info](#)

Directory ID	Directory name	Type	Size	Multi-Region	Status	Launch date
d-9067eb0127	directory.animals4life.org	Simple AD	Small	Not applicable	Active	Jul 10, 2024

Creating a Server Certificate for Client VPN

Prerequisites:

- Ensure the directory is in the active state.

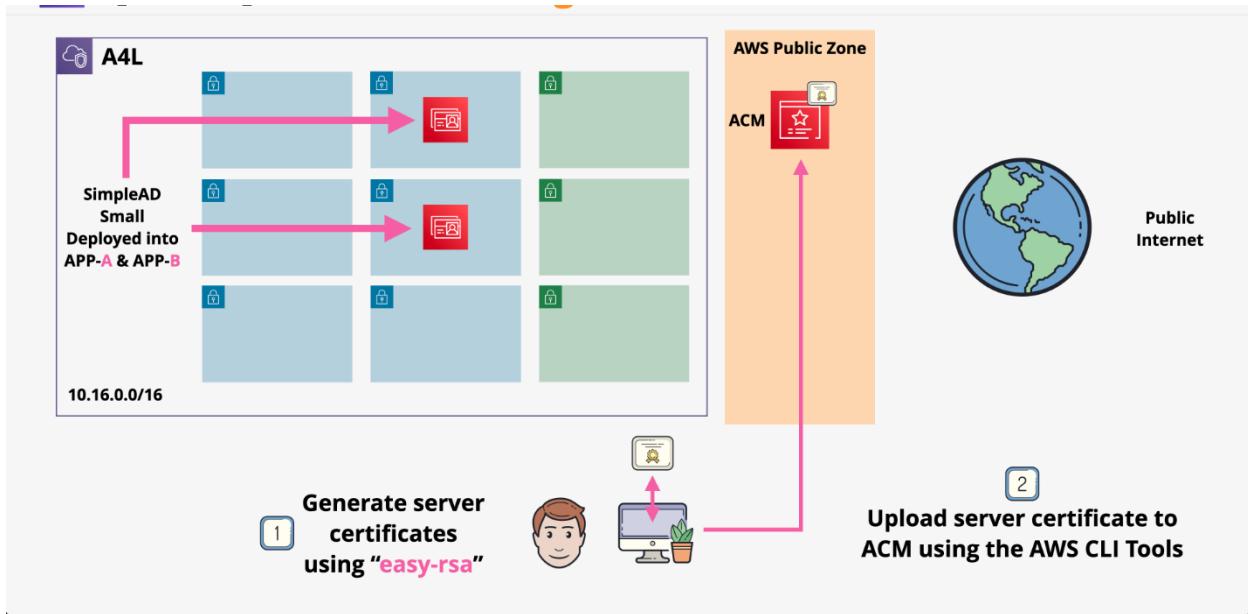
Authentication Method:

There are two ways we can implement authentication.

- Using Certificate based authentication.
- Using Identity-based Authentication.

Our choice method in the implementation is *Identity-based Authentication*.

We are at this level:



(Step-7):

“A service certificate is generated in Access Management”

Moving to local machine.....

OS used: Linux(debian)>

Steps:

```
frenzy@pop-os:~$ cd /tmp
frenzy@pop-os:/tmp$ git clone https://github.com/OpenVPN/easy-rsa.git
Cloning into 'easy-rsa'...
remote: Enumerating objects: 7095, done.
remote: Counting objects: 100% (1871/1871), done.
remote: Compressing objects: 100% (829/829), done.
remote: Total 7095 (delta 1088), reused 1439 (delta 1033), pack-reused 5224
Receiving objects: 100% (7095/7095), 52.45 MiB | 878.00 KiB/s, done.
Resolving deltas: 100% (3337/3337), done.
```

```
frenzy@pop-os:/tmp$ cd easy-rsa/easyrsa3
bash: cd: easy-rsa/easyrsa3: No such file or directory
frenzy@pop-os:/tmp$ cd easy-rsa/easyrsa3
frenzy@pop-os:/tmp/easy-rsa/easyrsa3$ ./easyrsa init-pki
Notice
-----
'init-pki' complete; you may now create a CA or requests.

Your newly created PKI dir is:
* /tmp/easy-rsa/easyrsa3/pki

Using Easy-RSA configuration:
* undefined
```

```
frenzy@pop-os:/tmp/easy-rsa/easyrsa3$ ./easyrsa build-ca nopass
=====
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]:frenzy
Notice
-----
CA creation complete. Your new CA certificate is at:
* /tmp/easy-rsa/easyrsa3/pki/ca.crt
frenzy@pop-os:/tmp/easy-rsa/easyrsa3$ y
```

Server buiding:

```
frenzy@pop-os:/tmp/easy-rsa/easyrsa3$ ./easyrsa build-server-full server nopass
=====
Notice
-----
Private-Key and Public-Certificate-Request files created.
Your files are:
* req: /tmp/easy-rsa/easyrsa3/pki/reqs/server.req
* key: /tmp/easy-rsa/easyrsa3/pki/private/server.key

You are about to sign the following certificate:
Requested CN:      'server'
Requested type:    'server'
Valid for:        '825' days

subject=
  commonName          = server
Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes
Using configuration from /tmp/easy-rsa/easyrsa3/pki/4ff5053c/temp.2.1
Check that the request matches the signature
-----
```

Name: "server"

Server certificate successfully completed....

```
You are about to sign the following certificate:

Requested CN:      'server'
Requested type:    'server'
Valid for:        '825' days

subject=
commonName          = server

Type the word 'yes' to continue, or any other input to abort.
Confirm request details: yes

Using configuration from /tmp/easy-rsa/easyrsa3/pki/4ff5053c/temp.2.1
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName           :ASN.1 12:'server'
Certificate is to be certified until Oct 13 13:09:16 2026 GMT (825 days)

Write out database with 1 new entries
Data Base Updated

Notice
-----
Certificate created at:
* /tmp/easy-rsa/easyrsa3/pki/issued/server.crt

Notice
-----
Inline file created:
* /tmp/easy-rsa/easyrsa3/pki/inline/server.inline

frenzy@pop-os:/tmp/easy-rsa/easyrsa3$
```

"Pki folder contains all the required certificates. All the contents in easyrsa folder are to uploaded to ACM, so that client vpn could authenticate and access."

#Attaching a user profile to ACM

The screenshot shows the AWS IAM console. On the left, the navigation pane is open with the 'Identity and Access Management (IAM)' tab selected. Under 'Access management', the 'Users' option is highlighted. The main content area displays the 'Users (1) Info' section. A table lists one user:

User name	Path	Groups	Last activity	MFA	Password age	Console
sreeram-user1	/	0	-	-	-	-

Created a user-profile using aws console, with IAM.

. Accessing user-profile 'sreeram-user1' using aws acm profile.

frenzy@pop-os: /tmp/easy-rsa/easyrsa3

```
frenzy@pop-os:~$ aws configure --profile sreeram-user1
AWS Access Key ID [None]: AKIAVVLQWDCUN5HIBLXA
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
frenzy@pop-os:~$ cd /path/to/easy-rsa/easyrsa3/pki
bash: cd: /path/to/easy-rsa/easyrsa3/pki: No such file or directory
frenzy@pop-os:~$ cd /tmp/easy-rsa/easyrsa3
frenzy@pop-os:/tmp/easy-rsa/easyrsa3$ 
```

Adding custom policy to allow write ACM server certificates upload:

The screenshot shows the AWS Identity and Access Management (IAM) service interface. On the left, a navigation pane lists 'Identity and Access Management (IAM)' under 'Dashboard', followed by sections for 'Access management' (User groups, Roles, Policies, Identity providers, Account settings), 'Access reports' (Access Analyzer, External access, Unused access, Analyzer settings), and 'CloudWatch Metrics' (CloudWatch Metrics, CloudWatch Metrics Insights). Below these are links for 'CloudShell' and 'Feedback'. The main content area is titled 'Users (1) Info' and contains a table with one row for 'sreeram-user1'. The table columns include 'User name' (with a checkbox), 'Path' (showing '/'), 'Group' (0), 'Last activity' (empty), 'MFA' (empty), 'Password age' (empty), and 'Console' (empty). At the top of the main area are buttons for 'Create user' (orange), 'Delete' (grey), and a search bar.

#Creating a new inline policy:

The screenshot shows the AWS Identity and Access Management (IAM) console. On the left, the navigation menu includes 'Dashboard', 'Access management' (with 'Users' selected), 'Roles', 'Policies', 'Identity providers', and 'Account settings'. Under 'Access reports', it lists 'Access Analyzer', 'External access', 'Unused access', and 'Analyzer settings'. The main content area is titled 'Permissions' and shows 'Permissions policies (1)'. A table lists one policy: 'IAMUserChangePassword' (AWS managed, Attached via Directly). Below the table are sections for 'Permissions boundary (not set)' and 'Generate policy based on CloudTrail events' (with a 'Generate policy' button).

Adding policy in form of json, which would only allow AWS CLI to access and write into the simple AD directory service using ACM.

“Sreeram-user1” is only allowed to accessed through AWS CLI, for which the access keys and secrey access keys have been generated....

The screenshot shows two sequential steps in the AWS IAM console for creating a new policy:

Step 1: Specify permissions

The "Policy editor" interface is displayed in JSON mode. It contains a single statement allowing the "acm:ImportCertificate" action on all resources:

```
1▼ {
2  "Version": "2012-10-17",
3  "Statement": [
4    {
5      "Effect": "Allow",
6      "Action": "acm:ImportCertificate",
7      "Resource": "*"
8    }
9  ]
10 }
11 }
```

On the right, there's a sidebar titled "Edit statement" with a "Select a statement" section and a "Add new statement" button.

Step 2: Review and create

The "Review and create" interface shows the policy details and the defined permissions:

Policy details

Policy name: acmwriteaccesspermission

Permissions defined in this policy

Allow (1 of 420 services)

Service	Access level	Resource	Request condition
Certificate Manager	Limited: Write	All resources	None

Buttons at the bottom include "Cancel", "Previous", and a prominent orange "Create policy" button.

Policy has been successfully created.....

Identity and Access Management (IAM)

Policy acmwriteaccesspermission created.

Created: July 10, 2024, 19:19 (UTC+05:30) | Last console sign-in: Never | Access key 2: Create access key

Permissions | Groups | Tags (1) | Security credentials | Access Advisor

Permissions policies (2)

Permissions are defined by policies attached to the user directly or through groups.

Policy name	Type	Attached via
acmwriteaccesspermission	Customer inline	Inline
IAMUserChangePassword	AWS managed	Directly

Permissions boundary (not set)

Uploaded the server certificates to “sreeram-user1” profile using AWS ACM successfully...

```
frenzy@pop-os:/tmp/easy-rsa/easyrsa3$ aws acm import-certificate --certificate fileb://pki/issued/server.crt --private-key fileb://pki/private/server.key --certificate-chain fileb://pki/ca.crt --profile sreeram-user1
{
  "CertificateArn": "arn:aws:acm:us-east-1:389467740328:certificate/0c041bf4-e3cd-41dc-8bd3-55bc3b2c0921"
}
frenzy@pop-os:/tmp/easy-rsa/easyrsa3$
```

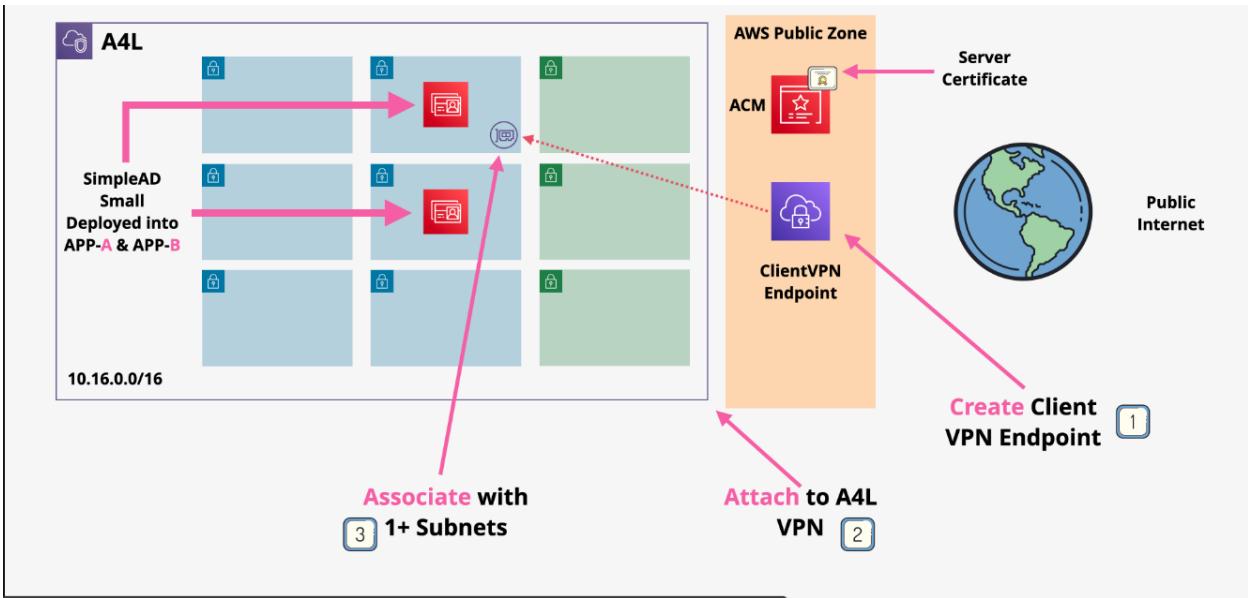
AWS Certificate Manager (ACM)

Certificates (1)

Certificate ID	Domain name	Type	Status
0c041bf4-e3cd-41dc-8bd3-55bc3b2c0921	-	Imported	Issued

(Step-8):

We are here.....



This stage involves creating a vpn end point and association using the VPC service within the aws console:

1. Choose vpc service
2. Select client VPN endpoint

Challenges faced:

As specified through the inline policy , acmwriteaccess is alone not enough for locating the server ARN certificates.

Unable to update or add inline policy using aws cli:

```
frenzy@pop-os:/tmp/easy-rsa/easyrsa3$ aws iam attach-user-policy --policy-arn arn:aws:iam::aws:policy/AmazonACMCertificateFullAccess --user-name sreeram-user1
Unable to locate credentials. You can configure credentials by running "aws configure".
```

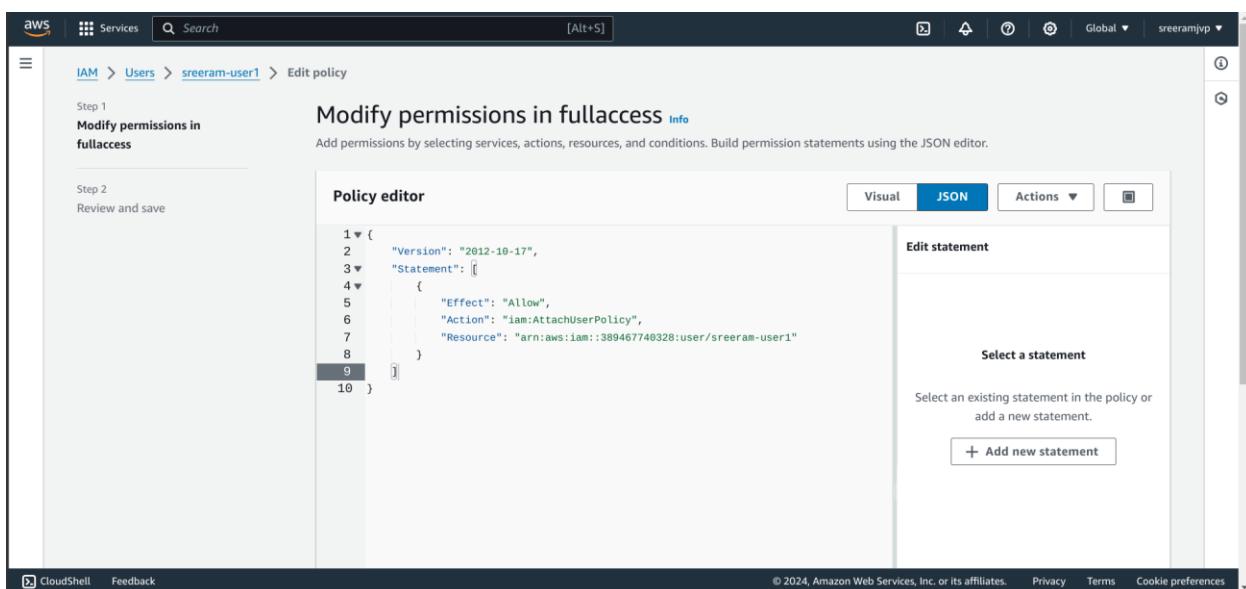
Solution;

Used ‘aws configure’ to authenticate request from my local machine.

Lack of policy permissions for sreeram-user1 to manifest policy in Identity access management:

```
frenzy@pop-os:/tmp/easy-rsa/easyrsa3$ aws iam attach-user-policy --policy-arm arn:aws:iam::aws:policy/IAMFullAccess --user-name sreeram-user1
An error occurred (AccessDenied) when calling the AttachUserPolicy operation: User: arn:aws:iam::389467740328:user/sreeram-user1 is not authorized to perform: iam:AttachUserPolicy on resource: user sreeram-user1 because no identity-based policy allows the iam:AttachUserPolicy action
```

To solve this, I have added an inline policy that allows attaching policy to user-1



On adding this policy, we are now able to attach AWS managed policy to allow fullaccess to the acm service for “sreeram-user1”:

Success....

```
frenzy@pop-os:/tmp/easy-rsa/easyrsa3$ aws iam attach-user-policy --policy-arm arn:aws:iam::aws:policy/IAMFullAccess --user-name sreeram-user1
Unknown output type: JSON
frenzy@pop-os:/tmp/easy-rsa/easyrsa3$
```

To solve this, we need to add a new inline policy that allows acm discoverability(i.e. ALLOWFULLACCESS), to ensure its visible to the VPC service.

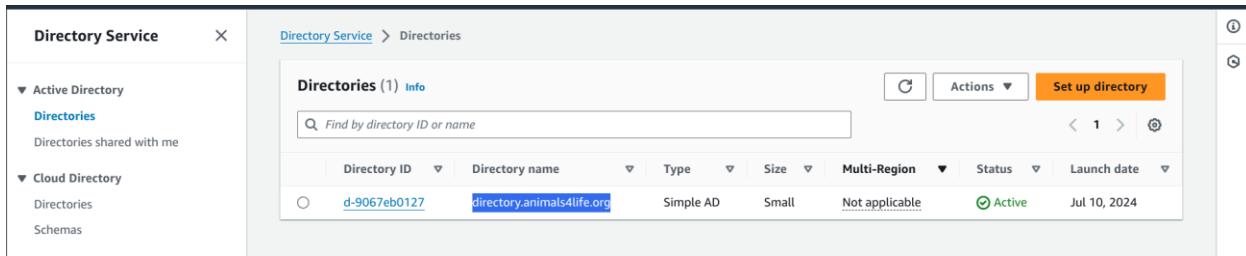
Now lets configure the client VPN end points...

But no so fast...

We still can't get our VPC to recognise the ACM certificate:

On further investigation, with reference to <https://repost.aws/questions/QUBHwO-HGfTcWCrSc5fBZmKw/server-certificate-not-showing-for-vpn-endpoint>, the main error stems from not including domain name in the server build command in easyrsa.

Our domain name, that we have created in simple AD, using Directory service, is "directory.animals4life.org"

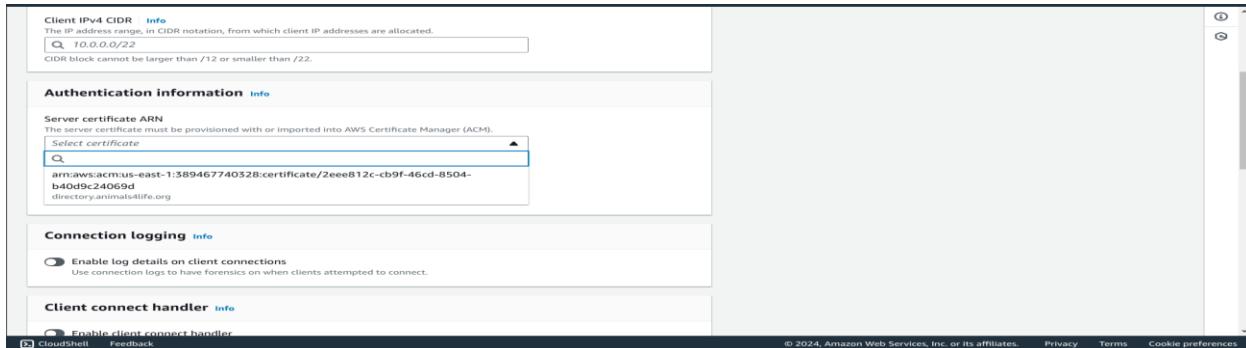


The screenshot shows the AWS Directory Service console. On the left, there's a sidebar with 'Directory Service' at the top, followed by 'Active Directory' and 'Cloud Directory'. Under 'Active Directory', there's a 'Directories' section with 'Directories shared with me'. Under 'Cloud Directory', there's a 'Directories' section with 'Schemas'. The main area is titled 'Directories (1) Info' and shows a table with one row. The row contains: Directory ID (d-9067eb0127), Directory name (directory.animals4life.org), Type (Simple AD), Size (Small), Multi-Region (Not applicable), Status (Active), and Launch date (Jul 10, 2024). There are buttons for 'Actions' and 'Set up directory' at the top right of the table.

On repeating the steps from creating sever certificate on our local machine, we have modified the command to :

```
"frenzy@pop-os:/tmp/easy-rsa/easyrsa3$ ./easyrsa build-server-full  
directory.animals4life.org nopass"
```

SUCCESS!



The screenshot shows the 'Create Client VPN endpoint' configuration page. It has several sections: 'Client IPv4 CIDR' (set to 10.0.0.0/22), 'Authentication information' (with a dropdown for 'Select certificate' containing 'arn:aws:acm:us-east-1:389467740328:certificate/2eee812c-cb9f-46cd-8504-b40d9c24069d directory.animals4life.org'), 'Connection logging' (checkbox for 'Enable log details on client connections'), and 'Client connect handler' (checkbox for 'Enable client connect handler'). At the bottom, there are links for 'CloudShell', 'Feedback', and copyright information: '© 2024, Amazon Web Services, Inc. or its affiliates.' and 'Privacy Terms Cookie preferences'.

Lets move to client vpn endpoint creation!

Client IPv4 CIDR [Info](#)
The IP address range, in CIDR notation, from which client IP addresses are allocated.
192.168.12.0/22 [X](#)
CIDR block cannot be larger than /12 or smaller than /22.

Authentication information [Info](#)

Server certificate ARN
The server certificate must be provisioned with or imported into AWS Certificate Manager (ACM).
arn:aws:acm:us-east-1:389467740328:certificate/2eee812c-cb9f-46cd-8504-b4... [▼](#)

Authentication options
Choose one or a combination of authentication methods to use.
 Use mutual authentication
 Use user-based authentication

User-based authentication options
 Active directory authentication
 Federated authentication

Directory ID
The ID of the AWS directory services directory to use.
d-9067eb0127 [▼](#)

Connection logging [Info](#)

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

For DNS server , it is essential when a user needs to resolve any connectivity issues.

We will use simple AD service for dns addressing:

Directory details

Directory type Simple AD	Directory DNS name directory.animals4life.org	Directory ID d-9067eb0127
Directory size Small	Directory NetBIOS name CORP	Description - Edit Directory service for aws client vpn demo

Networking & security [Edit](#) Application management Maintenance

Networking details

VPC vpc-093ee0a8565f3a520 Edit	Subnets subnet-0babbae923efdee68 Edit subnet-035e1d39baf062ffd Edit	Status Active
Availability zones us-east-1a us-east-1b	DNS address 10.16.103.176 10.16.41.126	Last updated Wednesday, July 10, 2024
		Launch time Wednesday, July 10, 2024

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The dns service is provided to our client end vpn within the vpc by the ad service itself!

We would also want our user to access the internet resources without the vpn . The vpn is configured in such a way that, it allows the connection to attach itself only to the resources that lie inside within the VPC.

To achieve that functionality, *split tunnel* is enabled.

The screenshot shows the AWS Client VPN configuration wizard. The configuration includes:

- Transport protocol used by the TLS sessions:** UDP (selected)
- VPC ID:** vpc-093ee0a8565f3a520 (A4L-VPC)
- Security group IDs:** sg-0fb4690310fd4c834 (default) (selected), sg-02f6d137d130eb508 (A4L-GeneralSGIPv6-QCgterOyNkk0) (selected)
- VPN port:** 443
- Session timeout hours:** 24
- Client banner:** Enabled

At the bottom, there are links for CloudShell, Feedback, and a footer with copyright information.

SUCCESS!!

The screenshot shows the AWS VPC dashboard with the following details:

- EC2 Global View:** A4L Client VPN
- Virtual private cloud:** Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways.
- Client VPN endpoints (1):** A4L Client VPN (Client VPN endpoint ID: cvpn-endpoint-09bb87caf6e325b2a, State: Pending-associate, Client CIDR: 192.168.12.0/22)

A success message at the top indicates: "You successfully created cvpn-endpoint-09bb87caf6e325b2a / A4L Client VPN."

The state remains pending association as it still needs to be attached with a subnet within a VPC.

Client VPN Endpoint State

- Pending-associate:** This state indicates that the Client VPN endpoint is waiting to be associated with a subnet in your VPC (Virtual Private Cloud).

Association with a Subnet

- **Subnet Association:** You need to link the Client VPN endpoint with a subnet in your VPC. This subnet will host the network interfaces for the Client VPN.

Network Interfaces

- **Client VPN Interfaces:** These are virtual network interfaces created in the associated subnet. They facilitate communication between the VPN clients and the VPC.

Traffic Flow

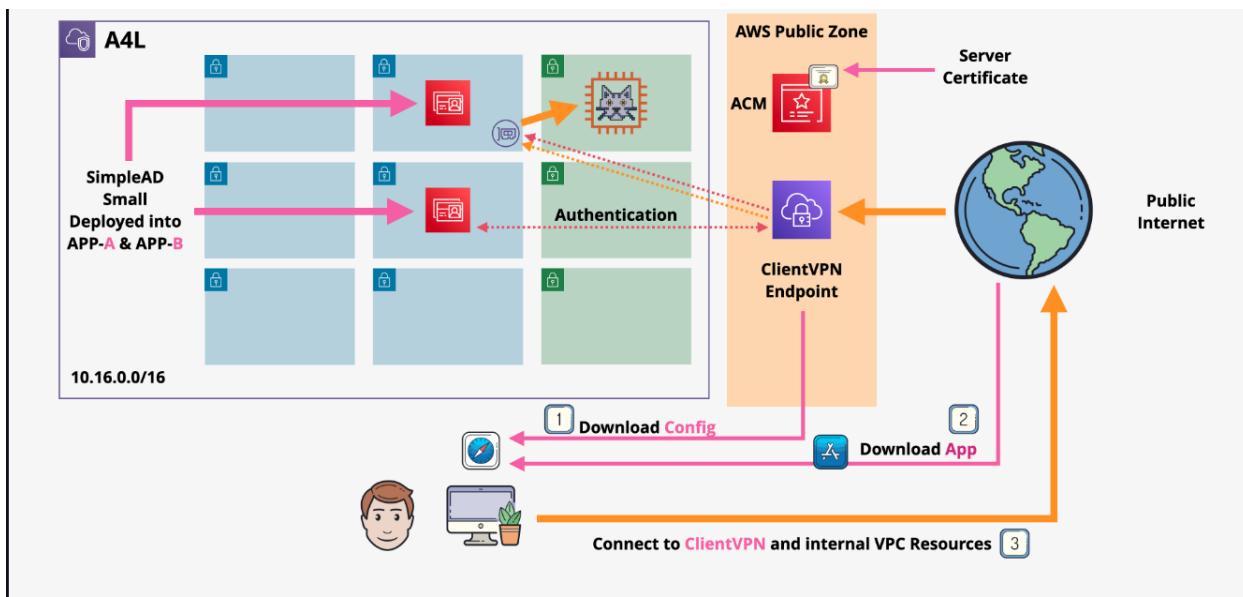
- **Traffic from Clients to VPC:** Traffic from the VPN clients will exit through these interfaces into the VPC.
- **Traffic from VPC to Clients:** Traffic destined for the VPN clients from within the VPC will enter through these interfaces.

(Step-9)

Creating association...

The screenshot shows the 'Associate target network' step of the AWS CloudFormation Create Stack wizard. The URL in the browser is [VPC > Client VPN endpoints > cvpn-endpoint-09bb87caf6e325b2a > Associate target network](#). The page title is 'Associate target network' with an 'Info' link. A note states: 'A target network is a subnet in a VPC. You associate a subnet in an Availability Zone to the client VPN endpoint. You can associate one subnet per Availability Zone. You can associate subnets in one VPC to a client VPN endpoint.' Below this is a 'Details' section with fields: 'Client VPN endpoint ID' set to 'cvpn-endpoint-09bb87caf6e325b2a', 'VPC' dropdown set to 'vpc-093ee0a8565f3a520 (A4L-VPC)', and 'Choose a subnet to associate' dropdown set to 'subnet-035e1d39baf062ffd (A4L-SN-PRIV-A)'. At the bottom are 'Cancel' and 'Associate target network' buttons, with the latter being orange.

We are here....



Wait for the target association status to change to “Associated”...

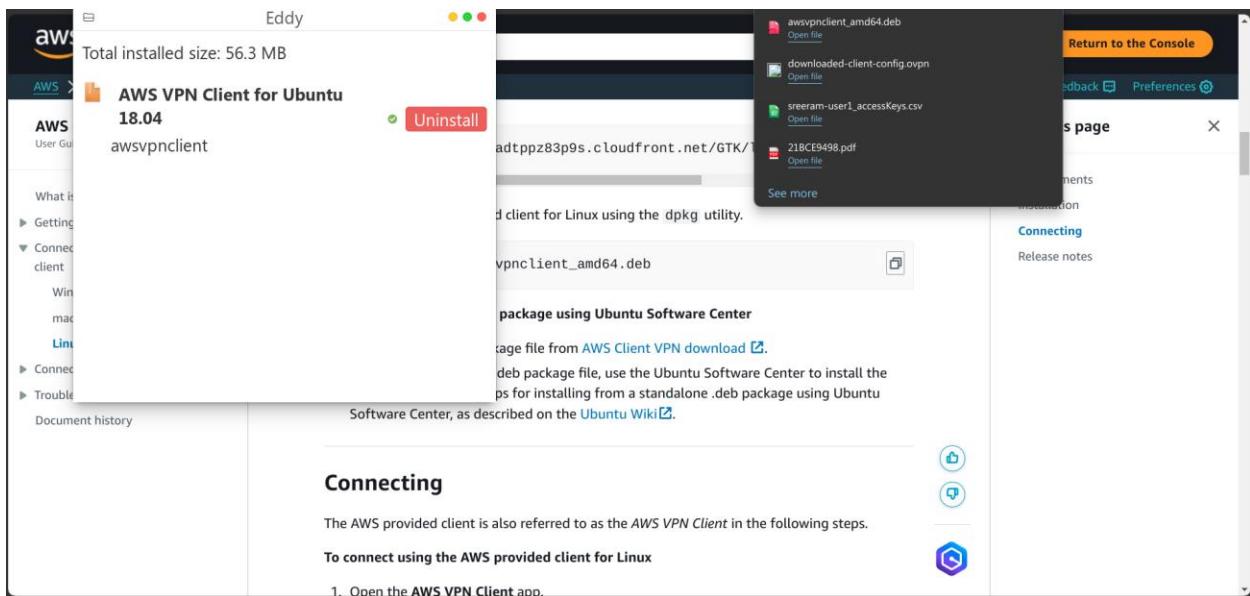
After that, download client configuration settings...

The screenshot shows the AWS CloudFront console. At the top, there's a header with 'VPN' on the left, followed by 'cvpn-endpoint-09bb87caf6e325b2a' in the center, 'Available' with a checkmark icon, and '192.168.12.0/22' on the right. Below this, a modal dialog box is open with the title 'Download client configuration'. It contains the message 'Download VPN client configuration file for client VPN endpoint **cvpn-endpoint-09bb87caf6e325b2a**?'. Underneath, it shows 'Client VPN endpoint ID' and a checkbox labeled 'cvpn-endpoint-09bb87caf6e325b2a'. At the bottom of the modal are 'Cancel' and 'Download client configuration' buttons, with the latter being orange. Behind the modal, there's a table titled 'Target network associations (1) Info' with one row. The table has columns for 'Association ID', 'State', 'Network ID', 'Security groups', and 'Endpoint ID'. The first row shows 'ssoc-03d04c056cf59d379' with 'Associated' checked, 'subnet-035e1d39ba0f062ffd' under Network ID, '2 security groups' under Security groups, and 'cvpn-endpoint-09bb87caf6e3...' under Endpoint ID.

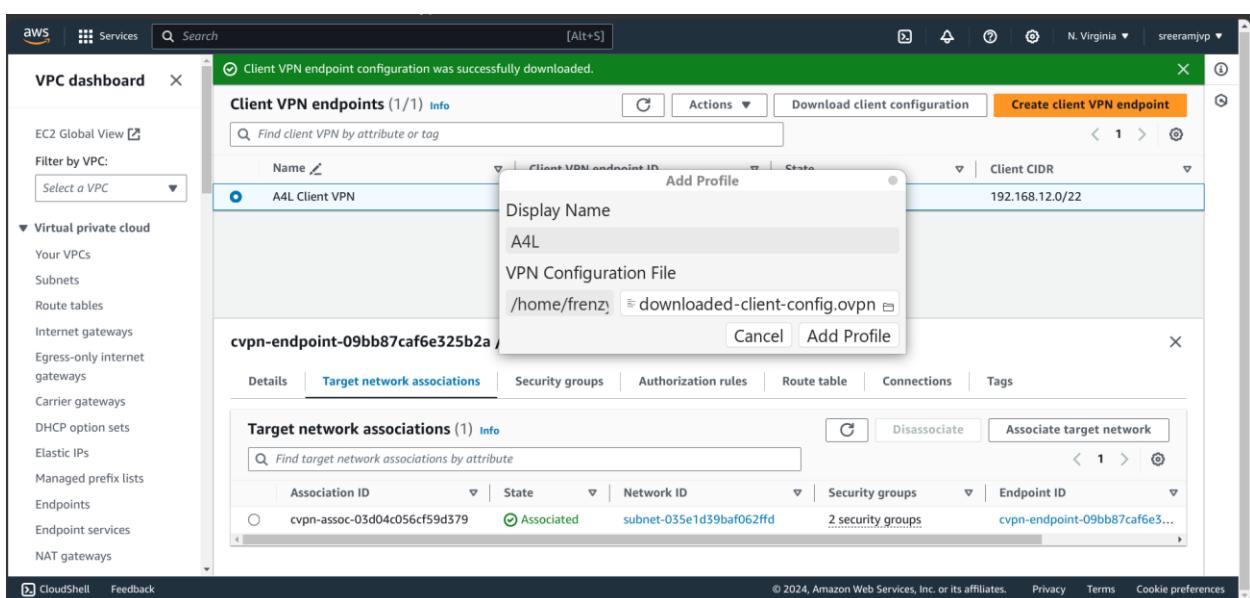
....

We are required to download the client vpn ...

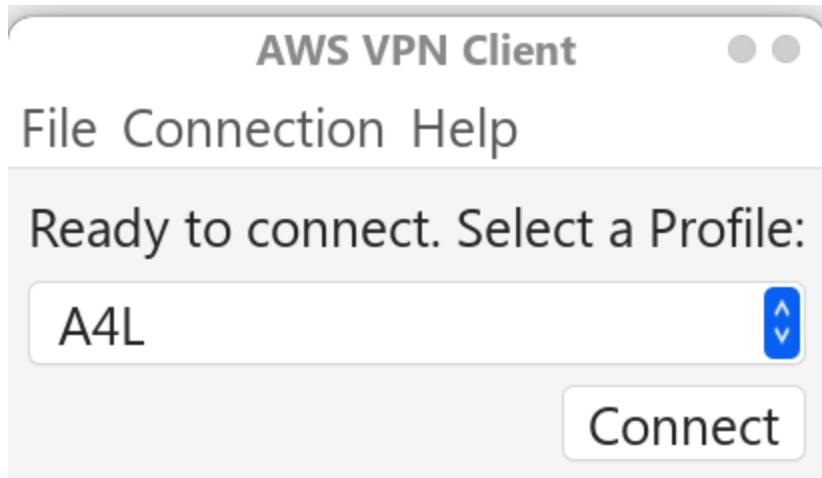
The screenshot shows the AWS Client VPN User Guide for Linux. The left sidebar has a 'User Guide' section with links for 'What is AWS Client VPN?', 'Getting started', 'Connect using an AWS provided client' (with sub-links for Windows, macOS, and Linux), 'Connect using an OpenVPN client', 'Troubleshooting', and 'Document history'. The main content area is titled 'Installation' and explains that there are multiple methods to install the AWS provided client for Linux. It provides 'Option 1 -- Install via package repository' with two steps: 1. Add the AWS VPN Client public key to your Ubuntu OS, and 2. Use the applicable command to add the repository to your Ubuntu OS, depending on your Ubuntu version. Below these steps are two code snippets: one for Ubuntu 18.04 and one for Ubuntu 20.04. The right sidebar is titled 'On this page' and lists 'Requirements', 'Installation' (which is bolded), 'Connecting', and 'Release notes'.



Associate openVPN profile to the AWS VPN CLIENT...



Click on connect...and enter the simple ad credentials that was created at the beginning..



Well....there seems to be another problem with the dependencies associated with the .deb file. On further digging, its been found that, AWS is no longer supporting AWS client vpn for Ubuntu version, and the client would only run on older versions of lib, which would be a no no due to security concerns.

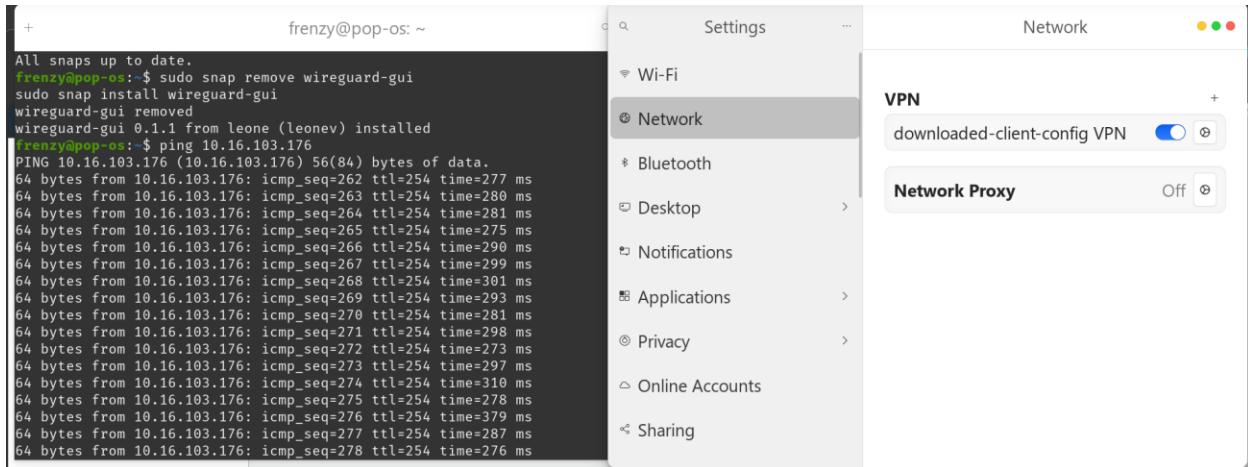
The alternative would be to use, wireguard with gui installed using snapd, which is a proprietary software packaging by *canonical softwares*.

Wireguard is similar to AWS client vpn, and is supported and updated. The repository is updated .

```
Processing triggers for desktop-file-utils (0.28-1ubuntu1) ...
frenzy@pop-os:~$ sudo snap install wireguard-gui
2024-07-10T23:00:06+05:30 INFO Waiting for automatic snapd restart...
wireguard-gui 0.1.1 from leone (leonev) installed
frenzy@pop-os:~$ █
```

After adding in user-name and password , go to the vpn client settings and select authorization roles, and approve,

SUCCESS!!!



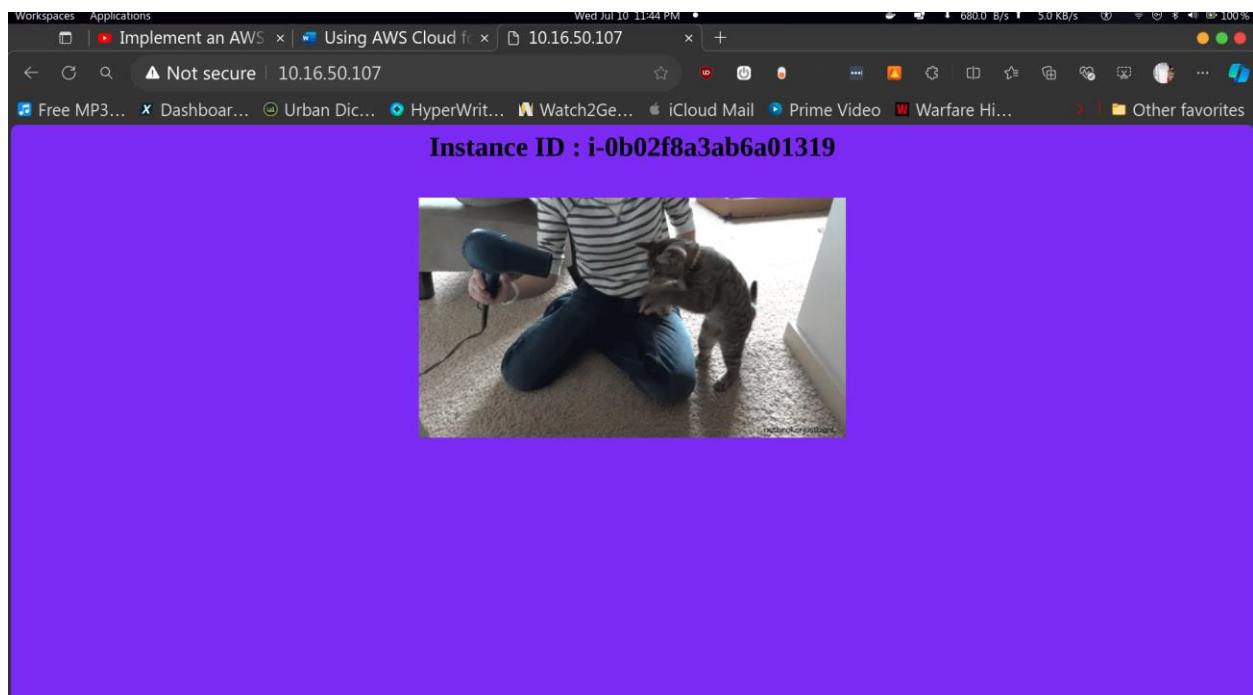
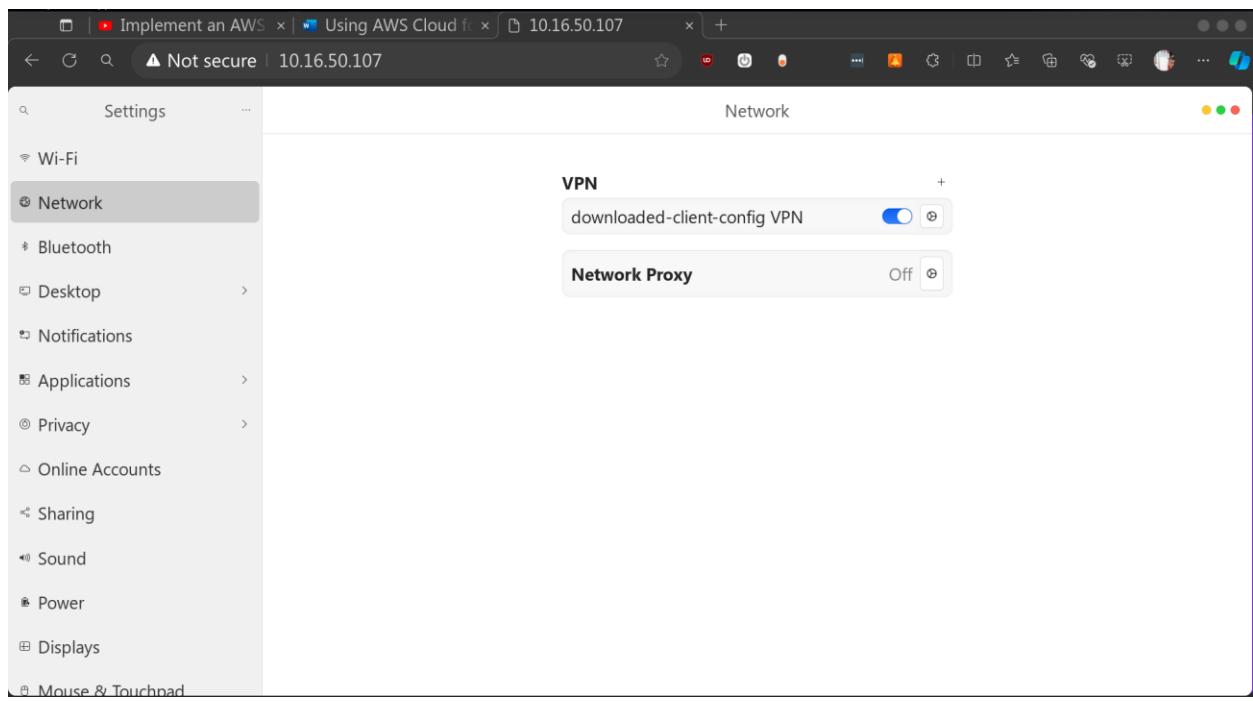
Another way to verify if our VPN is working is to select Ec2 instance:

The screenshot shows the AWS CloudWatch Metrics interface. A metric named "AWS/EC2/Instances" is displayed with a value of 1. The metric has a timestamp of 2023-09-11T10:00:00Z and a unit of "Count". The metric is associated with the EC2 service and the us-east-1 region.

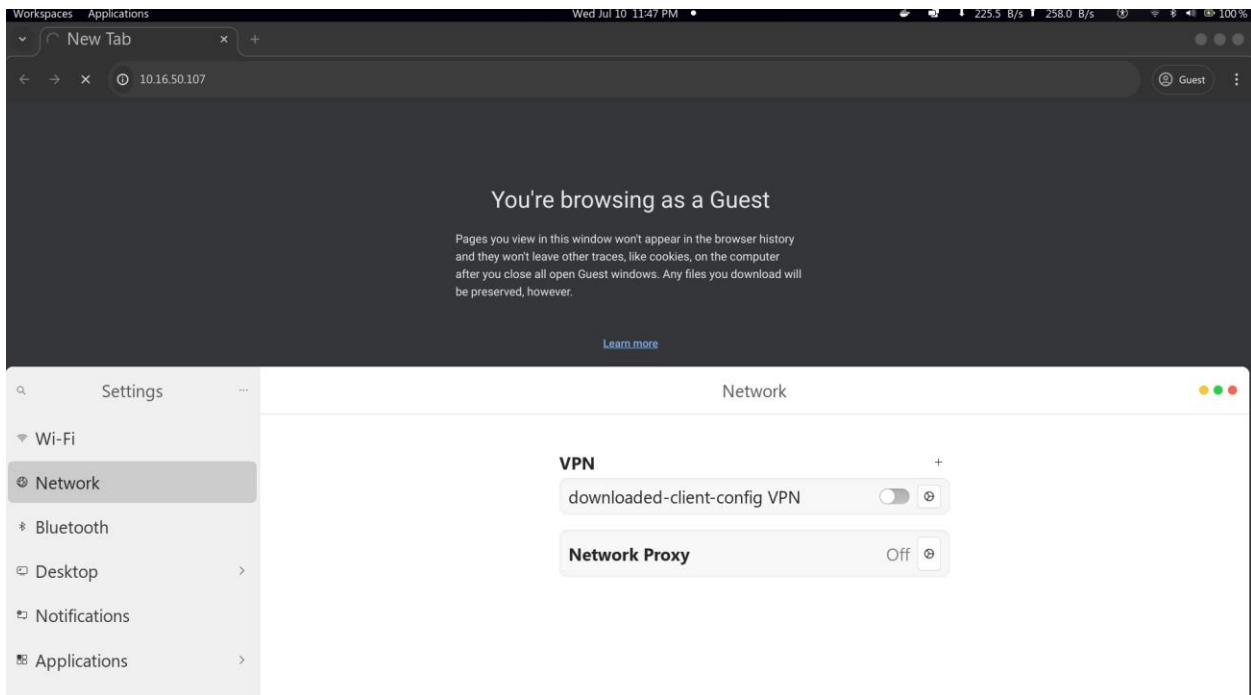
We can see there is a an ec2 instance running hosting a static website. Choose the ec2 instance and copy the private IPV4 instance ip address:

The screenshot shows the AWS EC2 Instances interface. An EC2 instance named "CATWEB" (with ID i-0b02f8a3ab6a01319) is selected and shown in the details pane. The instance is running in the t3.micro instance type, located in the us-east-1a availability zone, and has a public IPv4 address of 44.203.102.149. A tooltip indicates that the private IPv4 address is 10.16.50.107.

VPN ON:



With vpn off:



.....

“Successfully set up a client side vpn using simple AD service, AWS cli, directory service, vpc, vpn, and verified it using an ec2 instance within the vpc network.”