Cert Solution Guide - ACME Configuration Guide

Version: 2022.1.0

Copyright AppViewX, Inc.

External Reference Links

This product includes software developed by the CentOS Project (www.centos.org).

This product includes software developed by Red Hat, Inc. (www.redhat.com).

This product includes software developed by VMware, Inc. (www.vmware.com).

Contact Information

AppViewX, Inc.

222 Broadway, FL 19

New York, NY 10038

Email: info@appviewx.com

Web: www.appviewx.com

Contents

Preface

Revision History

Revision	Description	Date
1.0	Initial release of document for Release 2022.1.0	June 2022

About this Guide

This guide describes the step-by-step process to configure and validate the ACME service in AppViewX

Text Conventions

The following text conventions are used in this document:

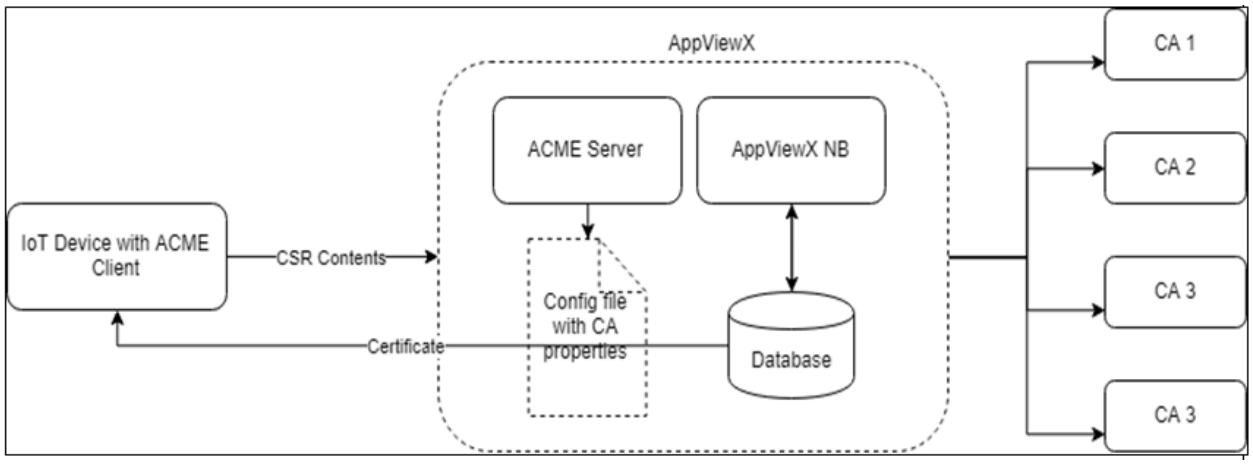
Convention	Description
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in the text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`codeblock`	Indicates commands with a paragraph, URLs, codes in examples, text that appears on the screen, or text that you enter.

Chapter 1: ACME Service Flow

- Objective
- Challenge Verification
- Certificate Enrollment
- Certificate Revoke

Objective

To support ACME protocol which helps IoT devices to request for certificates through AppViewX's ACME server



Challenge Verification

AppViewX ACME service supports both HTTP and DNS challenge verification.

- HTTP Challenge Verification
- DNS Challenge Verification

HTTP Challenge Verification

During this process, ACME service will collect the request from the client and send back the HTTP challenge value as the response to the client. The client will have to collect the challenge value manually, convert it into a file and place it in the predefined directory for ACME service in the webserver.

Once the challenge value is pasted, the client will have to trigger the verification process request to the ACME service provided with the method of verification detail in it. Once the verification becomes successful, ACME service will proceed with the client requests of certificate enrolment or revoke.

DNS Challenge Verification

During this process, ACME service will collect the request from the client and send back the DNS challenge value as the response to the client. The client will have to collect the challenge value manually, convert it into a txt record and paste it in the config file of the web server.

Once the challenge value is pasted in the webserver config file, the client will have to trigger the verification process request to the ACME service provided with the method of verification detail in it. Once the verification becomes successful, ACME service will proceed with the client requests of certificate enrolment or revoke.

Note: In the case of enrollment from standard ACME clients (Cert-Manager and Certbot) the challenge value is automatically used by the ACME client and the user does not need to manually copy and paste the challenge.

Certificate Enrollment

The above mentioned any one of the verification processes must have been completed successfully so that the client's request for the certificate enrolment will take place by ACME service. During the request, the client will share the CSR (Certificate Signing Request), account details and order confirmation to AppViewX through ACME service.

AppViewX will collect it and forward it to the user defined CA, during its respective ACME GUI configuration in AppViewX. Based on the Retry count and Retry Frequency values, the number of calls and the interval between those calls for fetching the CA certificates will be performed. Once the certificate has been received by AppViewX, in the holistic view of that certificate the chain of trust, CA connector and the application connectors will be formed and shown. Using the application connector the certificate would have been shared as the response with the client through ACME service.

Note: User can ignore the level of approvals during certificate enrolment process in AppViewX by disabling Approval Required option in its respective group-policy, which is selected during its ACME service configuration in AppViewX GUI. Else, the respective approver has to login to the AppViewX GUI for approving the certificate.

Certificate Revoke

The above mentioned verification processes must be completed successfully for this revoke request from the client to be processed by the ACME service. During this request, the client sends the account details and the revocation information to the ACME service.

Once the ACME service receives the request, it will proceed with invoking the revocation call with the respective CA, which has been selected during its ACME service configuration in AppViewX GUI.

Chapter 2: Limitations

1. IoT device must have a ACME client installed.
   - AppViewX must be configured with details such as Certificate Authority to issue the certificate, Group to which the certificate must fall under.
2. Private key will be generated within end device and only the CSR content is passed to AppViewX for certificate creation..
3. One ACME Server one CA: In SCEP the end client passes the CSR information along with the CA which must issue the certificate, but in case of ACME, the client cannot let the ACME server know from which CA the certificate must be issued. Due to this limitation, if the customer has 4 different CAs, then 4 ACME server has to be spun up with individual CA settings for each ACME servers.
4. Unable to spin up server through UI interaction: Triggering spin up of ACME server and assigning CA settings to it is not feasible through User interface.

Chapter 3: ACME in Containers

For storing certificates and keys in containers ACME is leveraged, with ACME users can have the certificates and keys stored in the secrets of the k8s and openshift containers and have it deployed to the service during the DevOps process itself which does not consume more time

Chapter 4: Appviewx CLM for Auto Enrollment

ACME Server Implementation

AppViewX CLM has the ACME server implementation which can be used for issuing certificates to the ACME client based on the enrollment request from the client.The certificates are issued based on the challenge verification done at the ACME server end, the challenges can be of any type DNS challenge or HTTP challenge.

Advantages of AppViewX CLM as an ACME Server

1. AppViewX provides CA signed certificates based on policy configured.
2. Validation of the challenge HTTP/DNS is performed by AppViewX.
3. The DNS challenge validation is automated in certmanager, so AppViewX ACME server can validate the challenge and provide the cert once the challenge is validated.

ACME Client Implementation

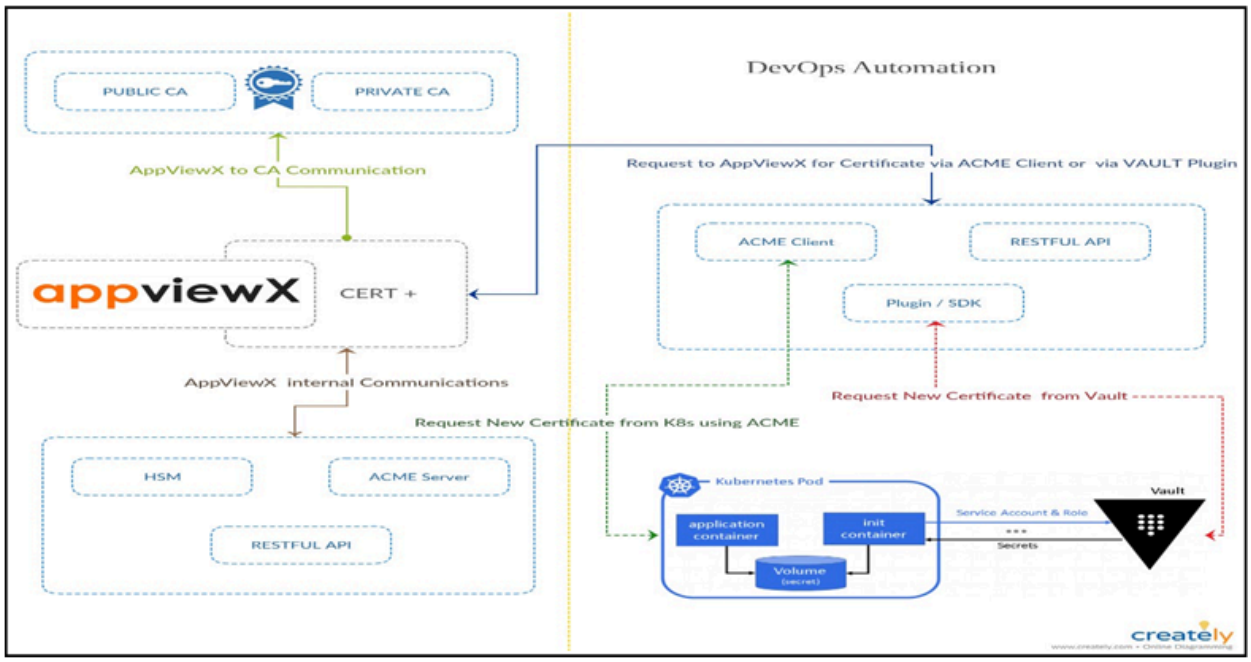
CertManager which is available in Openshift and Kuberenetes containers has the ACME client implementation which can be used for certificate enrollment and management operations.

CertManager helps in deploying certificates into the openshift and kubernetes containers which can be used for certificate operations within containers or for SSL inspection for traffic passing from the external world to the container.

These certificates are stored in a generic place called secrets

Process Flow

Process flow on how Certificate enrollment happens via a ACME client from containers to AppViewX CLM’s ACME Server.



Chapter 5: Run Book

This sample run book uses the below environment:

- One master node and 2 slave nodes of Kubernetes
- Kubernetes running version 18
- Cert Manager running v.15.1
- Base OS Centos 7.5

Chapter 6: Kubernetes Installation

- Initial Setup
- Cluster Initialization
- Adding Nodes to Cluster

Initial Setup

Execute the below in all 3 nodes.

1. Disable SELinux for running docker.

```
setenforce 0
sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/sysconfig/selinux
```

2. Enable br_netfilter for kubernetes installation.

```
modprobe br_netfilter
echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

3. Disable Swap.

```
swapoff -a
```

4. Edit /etc/fstab and comment the line below UUID tag.
5. Install docker.

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

6. Add docker repository and install docker.

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/dockerce.repo
yum install -y docker-ce
```

7. Install Kubernetes by adding the below lines as a kube repo.

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
```

```
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

8. Install kubernetes services kubeadm,kubelet and kubectl.

```
yum install -y kubelet kubeadm kubectl
```

9. Reboot the node and ensure firewalld is not running. Now start the docker and kubernetes and enable them to be started automatically during reboots.

```
systemctl start docker && systemctl enable docker
systemctl start kubelet && systemctl enable kubelet
```

10. Run kubernetes and docker in the same cgroup driver.

```
docker info | grep -i cgroup
```

11. If the result of the command is cgroupfs. Execute the below command.

```
sed -i 's/cgroup-driver=systemd/cgroup-driver=cgroupfs/g'
/usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf
```

12. Restart system and kubelet.

```
systemctl daemon-reload
systemctl restart kubelet
```

Cluster Initialization

Execute the below only in the master node.

1. Init the master node.

```
kubeadm init
```

2. Copy the 'kubeadm join ... ...' command to your text editor. The command will be used to register new nodes to the kubernetes cluster.
3. Create a new '.kube' configuration directory and copy the configuration 'admin.conf'.

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

4. Install the Calico Networking Plugin in the Cluster.

```
kubectl apply –f https://adminguide.appviewx.com/1619000091
```

5. Check the output of the below commands.

```
kubectl get nodes

kubectl get pods --all-namespaces
```

6. The first one should display the master to be in a ready state and the second to show the output of the pods in running state.

Adding Nodes to Cluster

To add nodes to the cluster execute the below in the secondary nodes.

Copy the output of the Join command printed in the earlier steps and execute the same in the secondary nodes.

Sample `kubeadm join 192.168.145.7:6443 --token p40x6p.yuovu27vaokom60i \ --discoverytoken-ca-cert-hash sha256:28c6d2b87f41918682cf701a1ddd3fc959be58f0c53ade447d1baf440d86de9c`

The above will join the secondary nodes to the master node.

Execute the kubectl get nodes in the master node and the output should show all the nodes in Ready state.

Chapter 7: Cert-Manager Installation

- Installation
- Validation

Installation

Execute the below steps in the master node for the cert-manager installation.

1. Create a cert-manager namespace.

```
kubectl create namespace cert-manager
```

2. Install helm for cert-manager installation.

```
cd /tmp

curl

https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get > installhelm.sh

chmod u+x install-helm.sh

./install-helm.sh

helm init
```

3. Create a tiller service account.

```
kubectl create serviceaccount tiller --namespace=kube-system

kubectl create clusterrolebinding tiller-admin --serviceaccount=kube-system:tiller --

clusterrole=cluster-admin

helm init --service-account=tiller

helm init --service-account tiller --override

spec.selector.matchLabels.'name'='tiller',spec.selector.matchLabels.'app'='helm' --output

yaml | sed 's@apiVersion: extensions/v1beta1@apiVersion: apps/v1@' | kubectl apply -f

-

kubectl get pods --namespace kube-system
```

4. Install Cert-manager using helm.

```
helm install --name cert-manager --namespace cert-manager jetstack/cert-manager --

version v0.15.1 --set installCRDs=true
```

5. Verify installation and Api service.

```
kubectl get pods -n cert-manager

kubectl get apiservice | grep cert
```

6. The result for the above will be

```
cert-manager cert-manager-7944d6bcfbbcqzc 1/1 Running 0 34s
cert-manager cert-manager-cainjector-86576c9999-
vkh52 1/1 Running 0 34s
cert-manager cert-manager-webhook-5fd7c64d48-
zltmr 1/1 Running 1 34s
kubectl get apiservice | grep cert
v1alpha2.acme.cert-manager.io Local True 29s
v1alpha2.cert-manager.io Local True 29s
v1alpha3.acme.cert-manager.io Local True 29s
v1alpha3.cert-manager.io Local True 29s
v1beta1.certificates.k8s.io Local True 20m
```

Validation

To verify cert-manager installation, we can create a sample issuer and see if a self signed cert is created.

Steps

1. Create a test resource file with the config.

```
cat <<EOF > test-resources.yaml
apiVersion: v1
kind: Namespace
metadata:
 name: cert-manager-test
---
apiVersion: cert-manager.io/v1alpha2
kind: Issuer
metadata:
 name: test-selfsigned
 namespace: cert-manager-test
spec:
 selfSigned: {}
---
apiVersion: cert-manager.io/v1alpha2
kind: Certificate
metadata:
```

```
 name: selfsigned-cert
 namespace: cert-manager-test
spec:
 commonName: example.com
 secretName: selfsigned-cert-tls
 issuerRef:
 name: test-selfsigned
EOF
```

2. Create the test certificate with the command.

```
kubectl apply -f test-resources.yaml
```

3. Verify the certificate created with the command.

```
kubectl describe certificate -n cert-manager-test
```

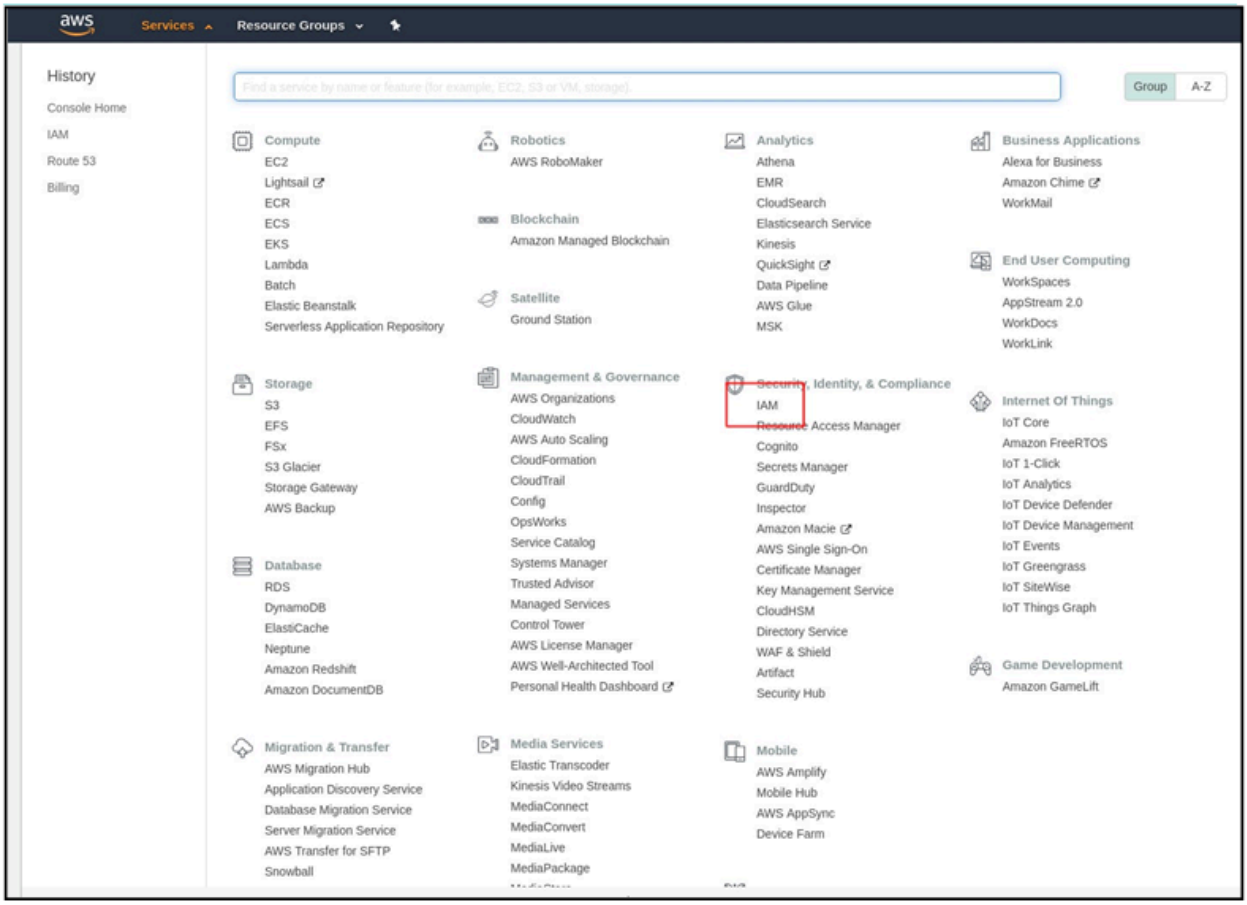
Chapter 8: AWS-Route53 IAM Account Setup

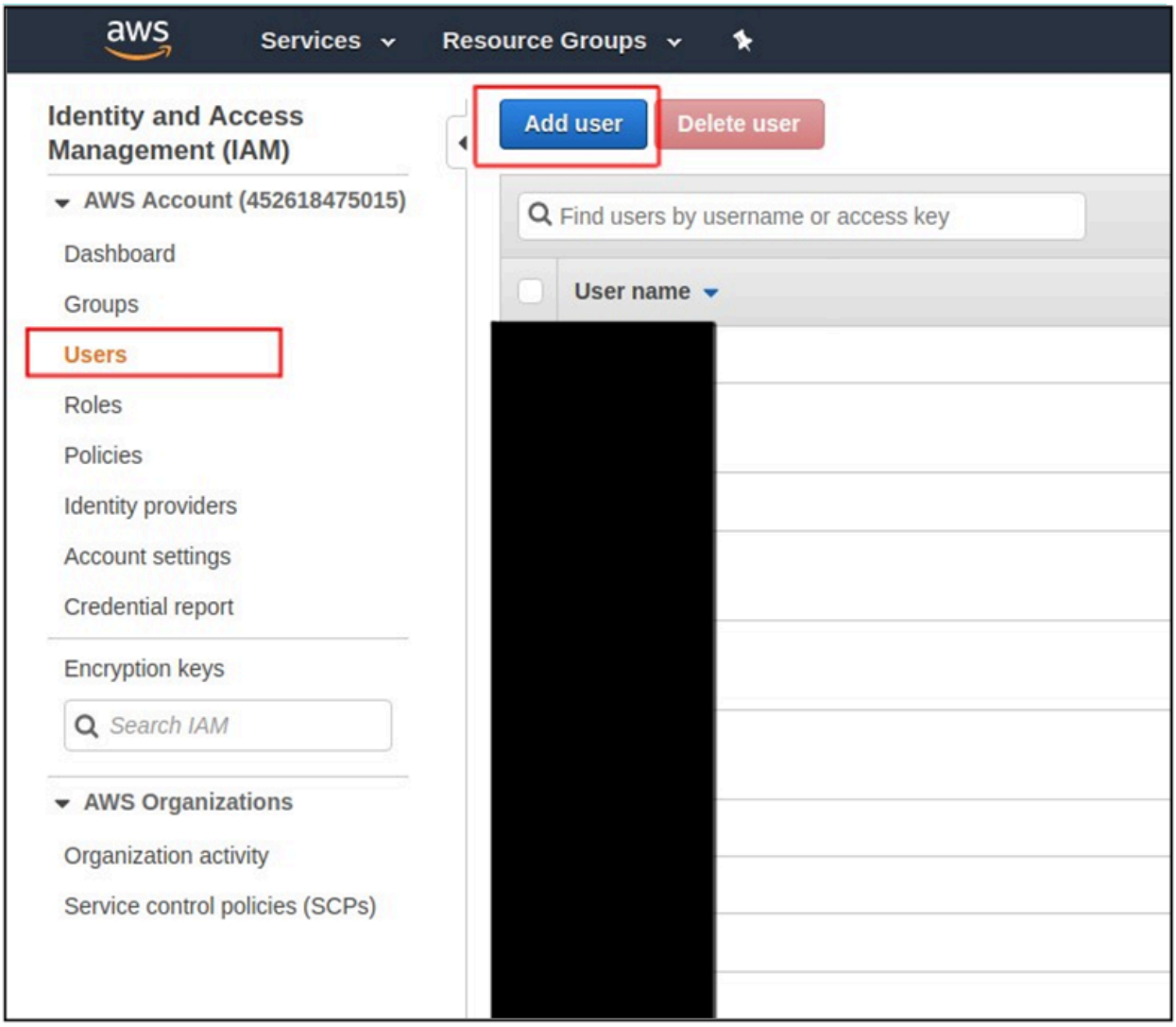
To enroll certificates via ACME the protocol supports various challenge mechanisms which are used to prove ownership of a domain so that a valid certificate can be issued for that domain.

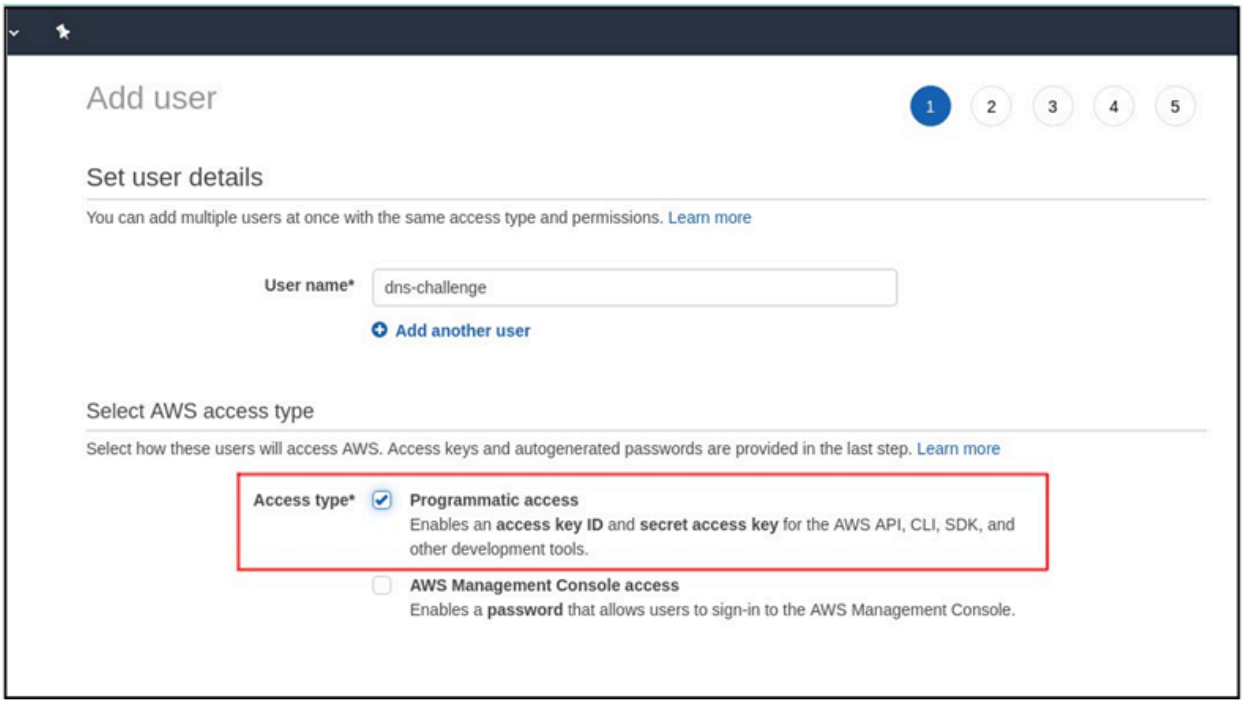
One such challenge mechanism is DNS-01. With a DNS-01 challenge, you prove ownership of a domain by proving you control its DNS records. This is done by creating a TXT record with specific content that proves you have control of the domain DNS records.

We will use Amazon Route53 to solve DNS01 ACME challenges.

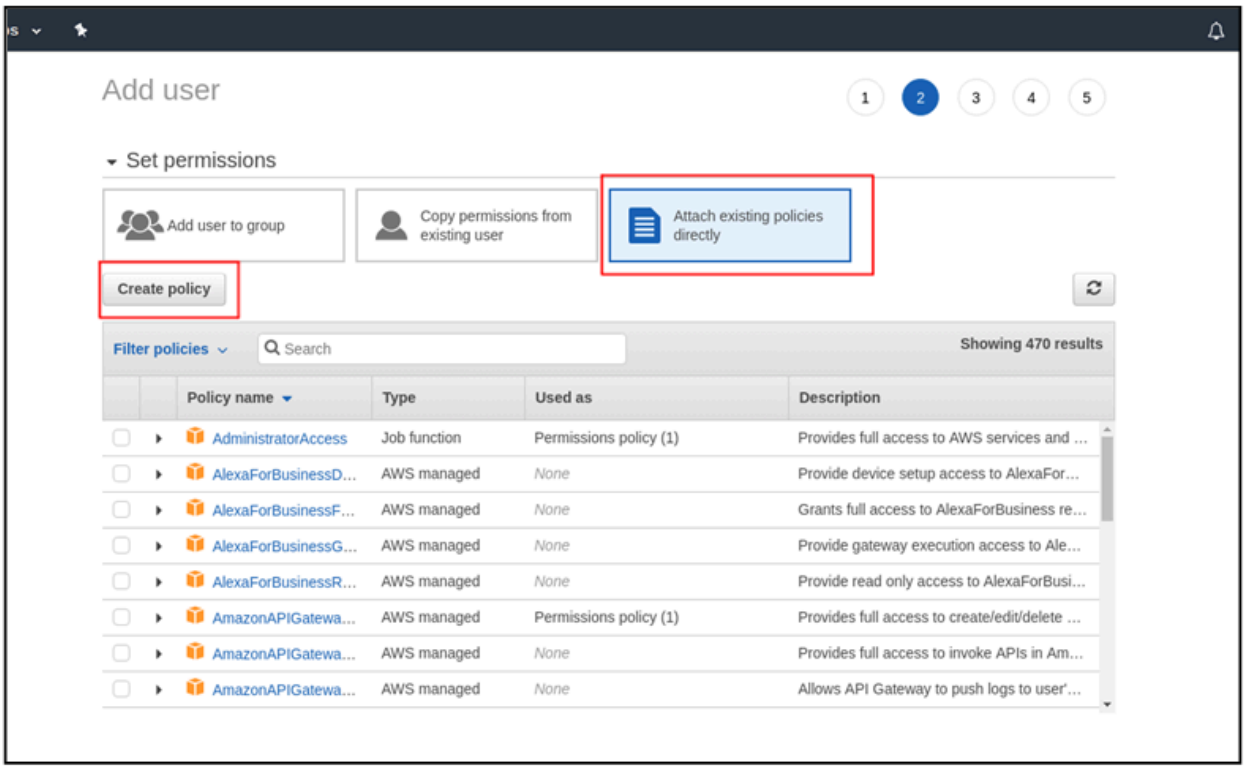
Go to IAM page and create a user.







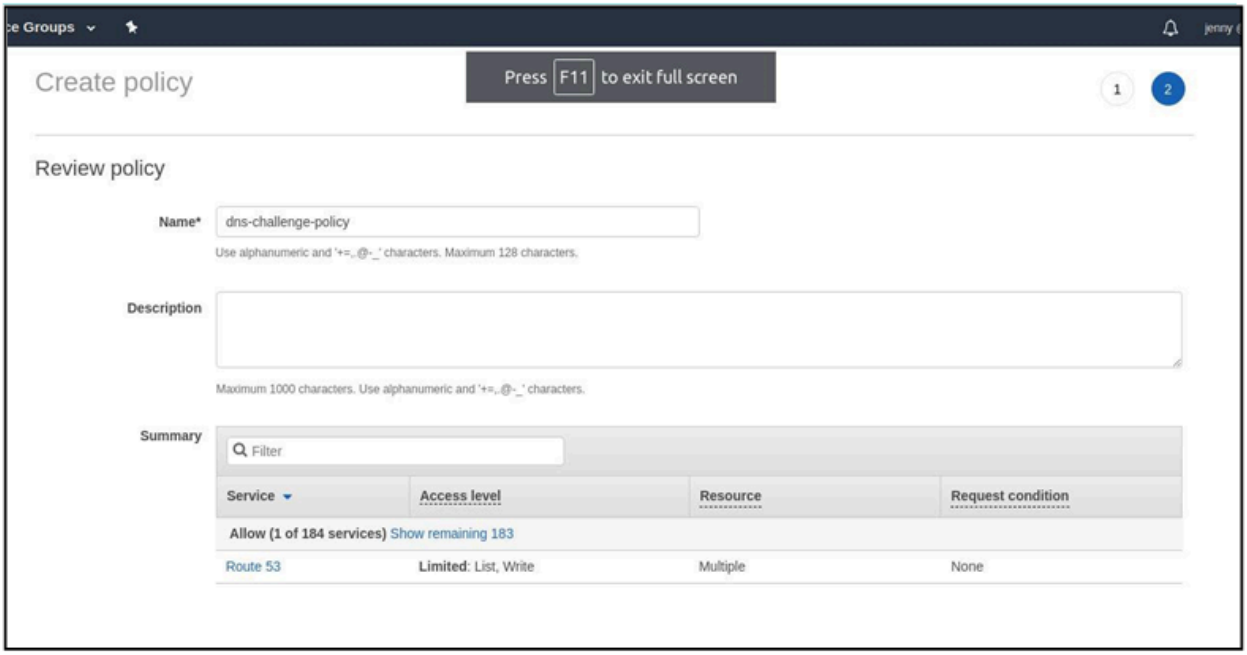
Click on next and select Attach existing policies directly and click on Create Policy. This will take you to a new page.



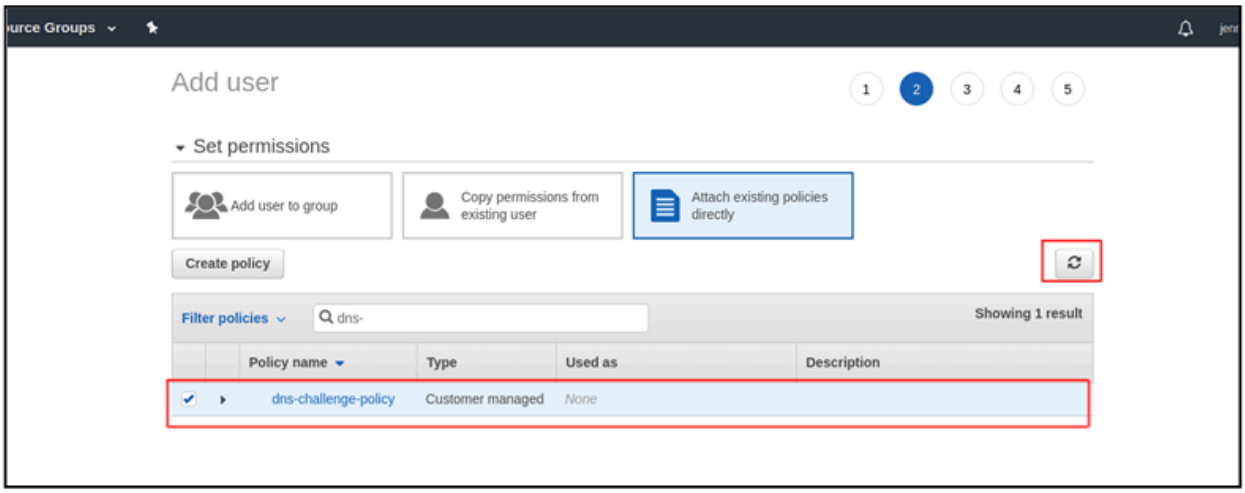
Now click on json and paste this and click Review Policy.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "route53:GetChange",
 "Resource": "arn:aws:route53:::change/*"
 },
 {
 "Effect": "Allow",
 "Action": "route53:ChangeResourceRecordSets",
 "Resource": "arn:aws:route53:::hostedzone/*"
 },
 {
 "Effect": "Allow",
 "Action": "route53:ListHostedZonesByName",
 "Resource": "*"
 }
 ]
}
```

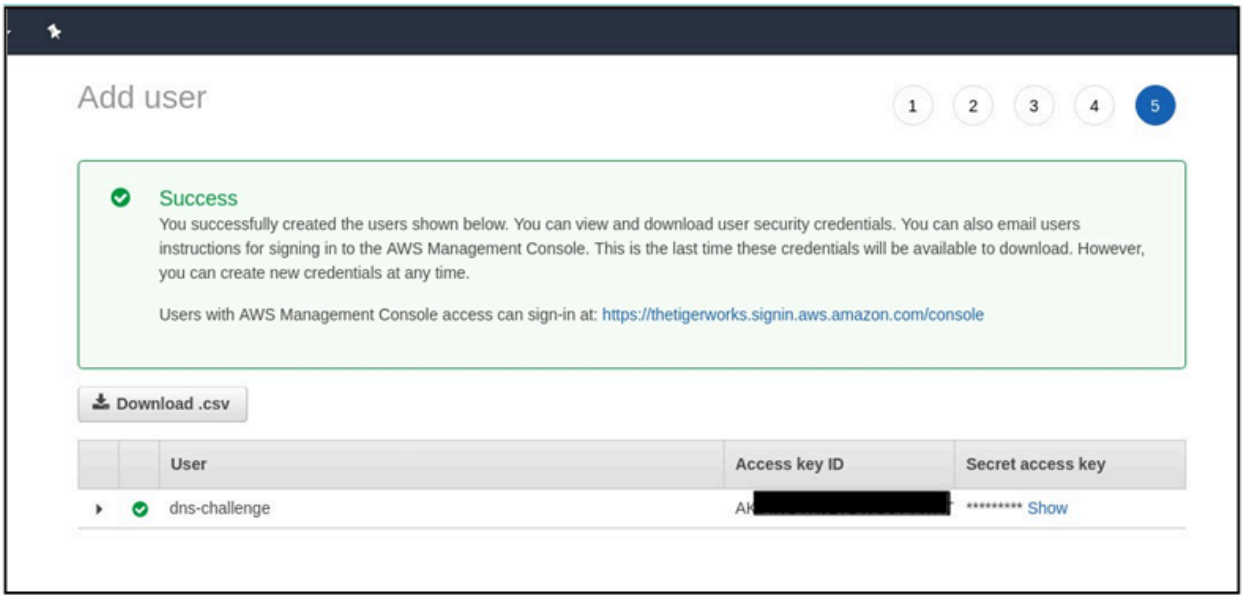
Name the policy and click Create policy.



Now go back to previous add user page, hit the refresh button and attach this policy to this user:



Click on next (tags are optional - you can ignore this) and finish the process. Download the .csv file.



Chapter 9: ACME Server Side

ACME Server Side

AppViewX CERT+ is a Certificate Lifecycle Management platform that can be configured as an ACME service end-point for DevOps and other ACME clients. The following document enumerates the steps to be followed to configure the ACME server on AppViewX, leveraging OpenTrust as a certificate authority signing the certificate requests.

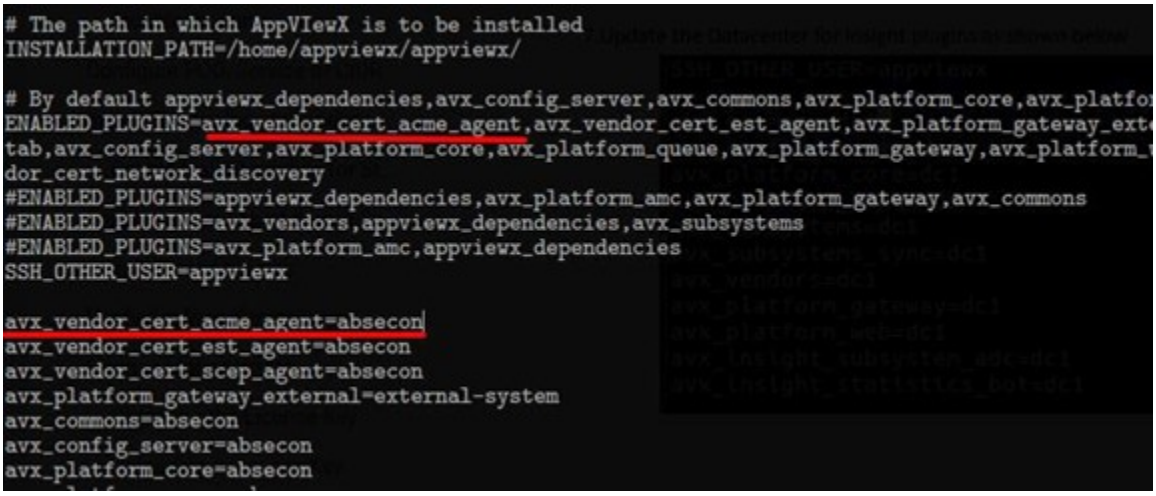
Prerequisites

1. Enable the AppViewX ACME plug-in.
2. Configure Open Trust CA account in AppViewX.
3. Create a Certificate Group.
4. Create a Certificate Policy.
5. Configure ACME settings.
6. The ACME client should have connectivity to AppViewX.
7. Update the ACME URL on the issuer YAML file on Kubernetes.
8. The CSR submitted for signing should contain one of the subject parameters (organization, state, or country) apart from SAN.

Enabling AppViewX ACME plug-in

To enable the AppViewX ACME plug-in, follow the steps given below,

1. Navigate to **<INSTALLER_PATH>/appviewx_kubernetes/scripts** folder.
2. Open **appviewx.conf** in editor mode using command **vi appviewx.conf**
3. Include **avx_vendor_cert_acme_agent** in the **ENABLED_PLUGINS** field and update the data center for the ACME plugin.



4. **Save** and **Exit** the **appviewx.conf** file.
5. Run **./plugins_install.sh** from scripts folder.
6. After successful installation, the ACME plug-in will be up and running.

Configure OpenTrust CA in AppViewX

1. Click the **Menu** button.
2. Navigate to **CERT+** > **Administration** > **Certificate Authority**.
3. Select **OpenTrust** from the left panel.
4. Click the **+ (Add)** button or the **Configuration Now** button.
5. Under the **General Information** section fill in the following details:
   a. Enter the **CA Account Name** which is a unique name for the CA device.
   b. From the **Proper/Usage** drop-down list, select **Server**.
   c. Select the **Data center** (AppViewX node running vendor plug-in) which has network reachability to **OpenTrust CA**.

> **Note:** The Proxy Required checkbox can be left unchecked. Proxy in CA setting allows the user to select if the proxy is needed for communicating with the Opentrust CAs web APIs. Sometimes the CA is hosted in the internet or in a secure network which needs to go through proxy

6. Under the **CA Configuration** section fill in the below details:
   a. Upload the **Client Authentication** certificate for the user account configured for AppViewX login to OpenTrust by clicking the **Upload** button.
   b. Provide the OpenTrust RA **URL** in the text field.

> **Note:** Only after the **Client Authentication** and the **URL** fields are provided, the **Validate and Fetch** button is enabled to fetch the **Certificate Management Profile**.

   c. Click the **Validate and Fetch** button to fetch the Certificate Management Profile
   d. From the **Certificate Management Profile** drop-down list, select the profiles which can be leveraged by an AppViewX user to enroll a certificate.
7. Click the **Save** button to save the CA configuration in AppViewX.
8. Click the **Check** button to test the connectivity between AppViewX and the certificate authority.
9. The **Connection Status Logs** pop-up window opens displaying the connection test result.
10. CA configuration with a successful connection status can be used for certificate enrollment.

Create a Certificate Group

The certificates generated/discovered in the AppViewX can be logically grouped for ease of management. All the certificate actions on a specific group can be restricted via RBAC. By default, all the certificates will be added under the Default group.

To create a Certificate Group, perform the following steps:

1. Click the **Menu** button.
2. Navigate to **CERT+** > **Groups & Policies** > **Groups**.
3. Click the **+ Create** button to configure certificate groups.
4. On the **Certificate Group** creation page, enter the details in the **Group Details** and **Other Details** sections.
5. Under the **Group Details** section fill in the below details:
   a. Select **Hierarchy** as **Default** from the drop-down list.
   b. Enter the **Group Name** for the new group.
   c. Enter the **Application ID** that is associated with the business units.
   d. Provide the **Description** of the certificate group.
6. Under the **Other details** section fill in the below details:

a. Certificate(s) associated with this group can be pushed automatically to the end machines only when the **Push Certificate Automatically** option is enabled.
b. Certificate(s) associated with this group can be renewed automatically to the end machines only when the **Renew Automatically** option is enabled.
c. When the **Renew Automatically** option is enabled, AppViewX can auto-renew the certificate based on the specified days before expiry.
d. Disable **Approval Required**, to proceed auto-renewal without approvals or enable **Approval Required** if approval required for auto-renewal as well.

> **Note:** Other fields like Contact Name, Line of Business Name, Email, Employment Name, Phone Number, Inventory Number, and Cost Center/Hierarchy are not mandatory.

7. Click **Create** to add the certificate group to the system.

> **Note:** In AppViewX plugin, a default group is present if not values are provided.

8. Click the Group **Name** to view/modify the certificate details.

Create CA Policy

A defined set of certificate parameters can be created as policies. This helps in enforcing security compliance over certificate creation across the organization. Also, these policies will be used to generate compliance reports against existing certificates.

1. Click the **Menu** button.
2. Navigate to **CERT+** > **Groups & Policies** > **CA Policy**.
3. On the CA Policy inventory, click on **+ Create** to configure certificate policy.
4. On the **policy creation** page, enter the details in the **Policy Details**, **CA Details**, and **Group Selection** sections.
5. Under the **Policy Details** section, enter the required details in the respective fields:
   a. Enter the **Policy Name**.
   b. In the **Description** field, enter the policy information.
   c. Choose the **Policy Type** as **Strict** or **Suggestive**:
      i. **Strict** - Enforce the standards defined in the policy where a user cannot modify any parameters while certificate enrollment in AppViewX
      ii. **Suggestive** - Suggests users with policy parameters. A user can modify suggested values if required while certificate enrollment in AppViewX

6. When the **Certificate Requests Need Approval** is enabled, it will enforce the peer approval process for any requests made to enroll, renew, regenerate, reissue, or revoke certificates. By default, Approval Required is disabled.
7. To export (or) download the private key of a server certificate, **Enable Access to Private Key** on the policy page. By default, Private Key Access is disabled to secure private key export/download by the users
8. **Enable Access to Private Key** for read-only users.
9. The private key of the certificate in the read-only group can be exported only when **Enable certificate push-bind access for the read-only user** is enabled. By default, both Private Key Access and private key access for read-only users are disabled.
10. By enabling **Enable certificate push-bind access for the read-only user** option, users in the read-only user group can perform certificate push, bind, and rollback operations from the holistic view of the certificate associated with this policy. By default, **Enable certificate push-bind access for the read-only user** is disabled.
11. Enable **Validate issuer and root certificate for compliance** to perform a compliance check. This option validates if the issuer and root of the certificate are also compliant with the standard that is defined in the Policy.
12. In the **CA Details** section, enter the required details in the respective fields:
    a. On the left pane, a window with a list of AppViewX supported Certificate Authorities are listed.
    b. Select the **OpenTrust** in the left pane.
        i. **CA Accounts** - Select the CA account name configured in the Certificate Authority page and click the **Add** button to proceed.
        ii. **Bit Length** - **Key Type** - Select the certificate bit length and key type as per organization standards and policies.
        iii. **Hash Function** - Select the certificate hash algorithm as per organization standards and policies. Recommending to use SHA256 and above.

Note: Only when the EC DSA curve is selected in **Bit Length** – **Key Type** field, the ECDSA curve parameter field is displayed.

13. Under the **Certificate Parameters** section. Certificate parameters filled in the policy page will automatically pre-populate in the certificate enrollment page.
    a. **Common Name** - Enter the fully qualified domain name (FQDN) of the server for which the certificate is requested.
    b. **Organization** - The name of the organization requesting the certificate.
    c. **Organizational Unit** - The division of the organization requesting the certificate.
    d. **Locality** - The location of the organization requesting the certificate.

e. **State** - The state in which the organization is located.
f. **Country Code** - The country in which the organization is located.
g. **Email** - The email contact details of the person responsible for maintaining the certificate.
h. **Subject Alternative Name** - Any additional hostnames, such as alternative websites, IP addresses can be added.

14. Click the **Save CA Details** to save the added information before group selection.
15. Under the **Group Selection** section, select the group(s) on which the policy has to be enforced.
16. Under the **Compliance Check** section, enable the **Perform Compliance Check**, to trigger a compliance check for the certificates in the inventory against the policy details.
17. Click the **Create Policy** button.
18. The policy is created and enforced based on the organizational standards.

Configure ACME Settings

To configure the ACME service, follow the steps given below,

1. Click the Menu button.
2. Navigate to **CERT+** > **Administration** > **Auto Enrollment** > **ACME**.
3. Select the **+ Add** or the **Configure Now** button.
4. Under the **Agent Details** section fill in the below details,
   a. Provide a unique **Name** to identify the ACME service.
   b. Provide **Gateway IP** and **Gateway Port** in ***.appviewx_configuration*** file.
      • Navigate to **<INSTALLATION PATH>** and open the ***.appviewx_configuration*** file. Search for AppViewX WEB URL to copy the IP address and port. Enter the copied IP address and port in the **Gateway IP** and **Gateway Port**.
   c. Enter the **Gateway Port** corresponding to the AppViewX ACME plug-in.
   d. Select **HTTP** from the **Challenge Type** drop-down list.
5. Under the CA Account section fill in the below details:
   a. Select the **Certificate Group** under which the certificate needs to be enrolled.
   b. Select **Certificate Type** as **Server**.
   c. **Select CA** as **OpenTrust** to enroll certificates from OpenTrust CA.
   d. Select the **CA Account** from the drop-down list.
   e. Select the **Certificate Management Profile** and **Zone** from the corresponding drop-down lists.
   f. The **CA Connector Name** is auto-filled.
   g. Provide the **Certificate Validity** in days.
6. Click the **Save** button to proceed.

Validate ACME Configuration

After the ACME settings are saved, one can check for the validity of the setting. ACME enrollment can be processed only if the settings are valid. Follow the below steps to validate the ACME configuration

1. Click the **Menu** button.
2. Navigate to **CERT+** > **Administration** > **Auto Enrollment** > **ACME**.
3. In the Auto Enrollment inventory, click **Check** to test the ACME service status.
   Once the validation is completed, the connection **Status** log is displayed.
4. Only if the validation is successful, the ACME setting can be used for certificate enrollment.

> **Note:** Once ACME setting is configured in AppViewX, copy the URL present on the ACME page and use it on the respective ACME client.

Chapter 10: ACME Client Side (Kubernetes and Cert-Manager)

To create the issuer in the Kubernetes environment for issuing certificates we need to createthe ACME client with the following settings in cert-manager.

1. Create the Kubernetes secret with AWS secret key for the cert-manager to access AWSroute 53 automation.

```
kubectl create secret generic route53-secret --from-literal=secret-access-
key="Pbg8GHW+vjgg6KdSdMmbvT4DMqTblz4sWtX6iao5"
```

Create an appviewx_issuer.yaml file for issuer config with below yaml config.

```
apiVersion: cert-manager.io/v1alpha2
kind: Issuer
metadata:
 name: acme-appviewx
spec:
 acme:
 email: sme@appviewx.com
 server: https://192.168.150.42:5300/avxapi/appviewx/acme/directory
 skipTLSVerify: true
 privateKeySecretRef:
 name: acme-appviewx-secret
 solvers:
 - selector:
 dnsZones:
 - "try.appviewx.com"
 dns01:
 route53:
 region: us-east-1
 accessKeyID: AKIAVXKISL45YCBKCLQL
 secretAccessKeySecretRef:
 name: route53-secret
 key: secret-access-key
 hostedZoneID: Z231A0MAAUOOZI
```

Notations Used above:

Below are the user defined values which vary depending on the environment.

acme-appviewx → ACME Issuer name which will issue certs.

https://192.168.150.42:5300/avxapi/appviewx/acme/directory → ACME Server URL

acme-appviewx-secret → Secret name to store the metadata for the issuer.

"Try.appviewx.com" → Zone for creating records

route53 → AWS Route53 DNS challenge issuer

us-east-1 → AWS region for the zone and dns automation

AKIAVXKISL45YCBKCLQL → Access Key ID create during the AWS-Route53 IAM Account Setup process.

route53-secret → Secret name created in the previous step.

Z231A0MAAUOOZI → Zone ID of the hosted in Route53.

2. Now create the issuer config on the Kubernetes environment with the below command.

```
kubectl apply -f appviewx_issuer.yaml
```

3. Create the certificate request appviewx_ca.yaml file for the cert to be created in AppViewX CLM. Sample yaml config.

```
apiVersion: cert-manager.io/v1alpha2
kind: Certificate
metadata:
 name: appviewx-demo
 namespace: default
spec:
 secretName: appviewx-demo
 duration: 2160h # 90d
 renewBefore: 360h # 15d
 issuerRef:
 kind: Issuer
 name: acme-appviewx
 dnsNames:
 - demo.try.appviewx.com
```

Notations Used above :

Below are the user defined values which vary depending on the environment.

appviewx-demo → name of the certificate

appviewx-demo → name of the secret in which the retrieved cert will be stored

demo.try.appviewx.com → common name of the certificate.

4. Now create the certificate with the command with the below command.

```
kubectl apply -f appviewx_ca.yaml
```

5. The above command will create the CSR and store the private key in the respective secret. The csr will be sent to AppViewX.
6. AppViewX will act as an RA, before sending the csr to the CA which is configured via the policy, AppViewX ACME server will create a DNS TXT record and send it to the ACME client which is to be configured on the DNS server by the ACME client.
7. ACME Client (Cert-Manager) will create records on the respective DNS server on the selected Hosted Zone.
8. AppViewX ACME server now will validate the create TXT challenge by performing an nslookup from the AppViewX server and if the DNS record is propagated AppViewX ACME server considers the requested certificate domain is an authorized domain.
9. AppViewX will now pass the CSR to the CA, get the certificate approved and send it to the ACME Client.
10. The generated certificate will now be stored in the specified secret.

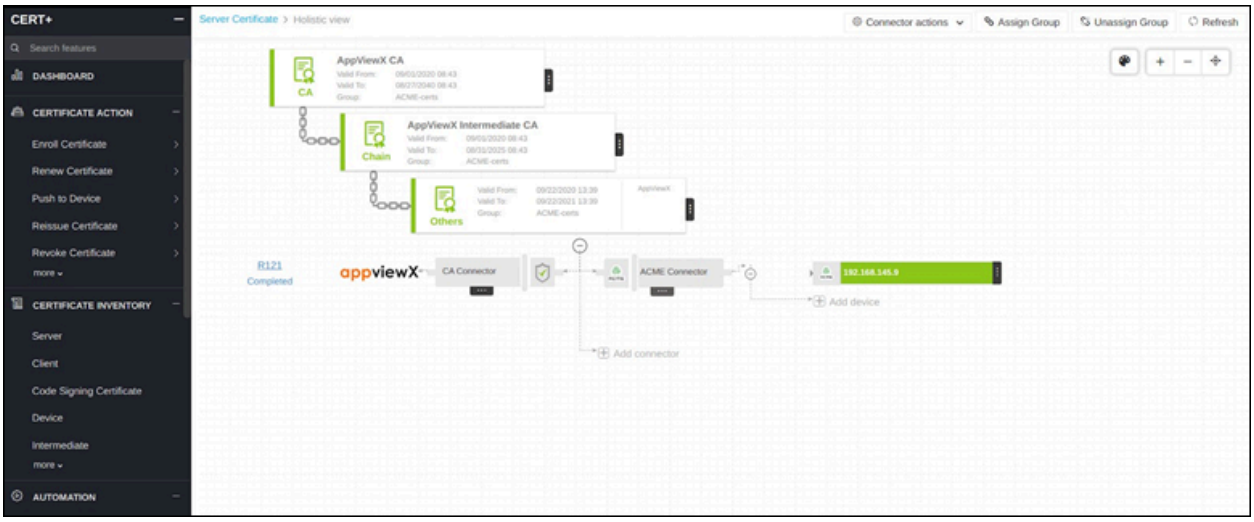
Chapter 11: Certificate Validation on AppViewX and K8s

Certificate Validation on AppViewX and K8s

The certificate created from ACME Client can be validated in AppViewX and K8s with the below steps.

AppViewX Validation

Login to AppViewX and navigate to the certificate inventory. The certificate created can be searched with the DNS name.



K8s Validation

Login to K8s master server and execute the command to get validate the certificate from the secret.

```
kubectl get secret freddi-acme-secret -o yaml
```

The above command will provide a yaml output of the secret as below.

```
apiVersion: v1 data:

ca.crt: "" tls.crt:

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUUrakNDQStLZ0F3SUJBZ0lSQVBVcG9rR2JJcFU4

eUpFRER2WDFPMEF3RFFZSktvWklodmNOQVFFTEJRQXcKYmpFaE1COEdBMVVFQXd3WVFYQnd

WbWxsZDFnZ1NXNTBaWEp0WldScFlYUmxJRU5CTVJVd0V3WURWUVFLREF4QgpjSEJXYVdWM1

dDQkpibU14RURBT0JnTlZCQWNNQjFObFlYUjBiR1V4RXpBUkJnTlZCQWdNQ2xkaGMyaHBibWQw

CmIyNHhDekFKQmdOVkJBWVRBbFZUTUI0WERUSXdNRGt5TWpFek16a3pNMW9YRFRJeE1Ea3lN

akV6TXprek0xb3cKQURDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnR

UJBTFR5bVRIdDZ5dUdwYThPZklaawpyRTRRSG5lT28zbzJDTEV0RTlWendvY2g1RjRJbEN0RVR2WX

hvVUpja1dZMmR6Wm1GUkV4UFFJcEhyTG1YdUlaClJhdWRqa3RyN1VZVHlmOUhuYmJ5YXF4bEV

TZFFBVFNrWkJzenNFazJ5bFBXWC9MMmFNQm80aFJzVzZHd0sycE8KNzBKYVdvU3pNSFFtbVYvW

W9saGQ1d0h3WHdCb3ZPQ0lvaUY5OG9Qa1NPRDdsZ3B2WU4xMUNhUXdVc1NwWkx2bgpWU0

RrU1dST0NNVUc3RWxld25yRWNXd0hxdEVSNjRjSkxNc2UwQlVuNmpPQmJaTnA0R1pGdWgzL3F

MaXh4dE9ZCld0MWFIOWhBbjl5MFA4aTJLM2FFUDdjRk96V2xzMWgzUnkvQ3lYNzJKNU9TQi9QK

zBSM1A0ekRKdlpuR3BRTmsKdjI4Q0F3RUFBYU9DQWY4d2dnSDdNQjBHQTFVZERnUVdCQlNyUV

htYUIwUnRubDFYaEVzbW1WM3RMQ1dJUGpBZApCZ05WSFNVRUZqQVVCZ2dyQmdFRkJRY0RB

UVlJS3dZQkJRVUhBd0l3REFZRFZSMFRBUUgvQkFJd0FEQXJCZ05WCkhSRUJBZjhFSVRBZmdoMW1

jbVZrWkdsa1pXMXZMblJ5YVdGc0xtRndjSFpwWlhkNExtTnZiVENCbWdZRFZSMGoKQklHU01JR1B

nQlJRWTNoUStPdU43ZnRmUjY3MGFFNFJNKzk0YmFGbHBHTXdZVEVVTUJJR0ExVUVBd3dMUVh

CdwpWbWxsZDFnZ1EwRXhGVEFUQmdOVkJBb01ERUZ3Y0ZacFpYZFlJRWx1WXpFUU1BNEdBMV

VFQnd3SFUyVmhkSFJzClpURVRNQkVHQTFVRUNBd0tWMkZ6YUdsdVozUnZiakVMTUFrR0ExVUV

CaE1DVlZPQ0VHRmswc0MzelZzTTJpdWMKVS81ekJWd3dhUVlEVlIwZkJHSXdZREJlb0Z5Z1dvWllN

VGt5TGpFMk9DNDVOQzR4TmpndlkyOXVkSEp2Ykd4bApjaTloZG5oamNtdy9ZM0pzUm1sc1pVN

WhiV1U5TVRJNU5EVTROakU1T0RVeE5qazRNekE0TlRnMk9ESXpNREk0Ck56VXhPRE0wTnpRME

1UVTJMbU55YkRCNEJnZ3JCZ0VGQlFjQkFRUnNNR293YUFZSUt3WUJCUVVITUFHR1hERTUKTWk0

eE5qZ3VPVFF1TVRZNEwyTnZiblJ5YjJ4c1pYSXZZWFo0YjJOemNEOXBjM04xWlhKelpYSnBZV3h1ZF

cxaQpaWEk5TVRJNU5EVTROakU1T0RVeE5qazRNekE0TlRnMk9ESXpNREk0TnpVeE9ETTBOelEwT

VRVMk1BMEdDU3FHClNJYjNEUUVCQ3dVQUE0SUJBUUJzazR3TEpob1VvY3NuVFFrbzJlSkErWVRt

cFl1eWt5TklUaGNuWFpHREhFZjlKNm0vSWVLczI0MVBFQ3Z0SjY3dTVGNHJUV1VtbThleTVYaGpu

OUoxSEU5ejN2Yk81NklIYWFhUjFTM0Fyd0hHWQpwTExnRFVSRk5HL1BVRnU3cEpkS2tKbExGSWJ

xMXc2bW5BTExySm9LM0ZuR0ZqRnRIVnBFZzMya0hzc0VXYmQ3CmhTUHFQQUR4TWYwV2Nsb

WErY1FSSE4rWUlSN0dtR244NEJ5RE9KUjczVVZBaDM1dXVOaXJIOTNKNEl6UEhYT3AKbURBMWp

LZzlLVGtkbDR2TWVpQUpyTE0vVG9XZ1pNY3NoTFNBcFg0UVBBYUcxendhOW9jUjc5WEFjbDNhd

HA3LwpGQ0J2VCtHazB6cmRsdmNjTDRscEV4Qm1DRVVlOVY3WUJ1d0JsM25ECi0tLS0tRU5EIENF

UlRJRklDQVRFLS0tLS0KLS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUQ0ekNDQXN1Z0F3SUJ

BZ0lRQWtQM2dtMXpPd3JMOFFwaXllbGtrakFOQmdrcWhraUc5dzBCQVFzRkFEQmgKTVJRd0Vn

WURWUVFEREF0QmNIQldhV1YzV0NCRFFURVZNQk1HQTFVRUNnd01RWEJ3Vm1sbGQxZ2dTVzV
```

qTVJBdwpEZ1lEVlFRSERBZFRaV0YwZEd4bE1STXdFUVlEVlFRSURBcFhZWE5vYVc1bmRHOXVNUX

N3Q1FZRFZRUUdFd0pWClV6QWVGdzB5TURBNU1ERXdPRFF6TXpOYUZ3MDBNREE0TWpjd09EU

XpNek5hTUdFeEZEQVNCZ05WQkFNTUMwRncKY0ZacFpYZFlJRU5CTVJVd0V3WURWUVFLREF4Q

mNIQldhV1YzV0NCSmJtTXhFREFPQmdOVkJBY01CMU5sWVhSMApiR1V4RXpBUkJnTlZCQWdNQ2

xkaGMyaHBibWQwYjI0eEN6QUpCZ05WQkFZVEFsVlRNSUlCSWpBTkJna3Foa2lHCjI3MEJBUUVG

QUFPQ0FROEFNSUlCQ2dLQ0FRRUE0QU40NTBMRkdoMitFVEFBQ2hBc3d3V2ttTnA4QWZ0WExD

dDMKOHdqRGs4a2VWV1FUakhHTzNpWkkrVDJjaEY1TnoweGdPK3JCdHAzWDM2WkRjMWRQWj

VYV09sSXJYeC9VcWhVcQpiMTdjRk9weS9maWZ3MEp2M25qL1NPV0I2OFp1ek9yYk5HNzFzcnM

wb1Z0Z2crV0tUazhWUjh2d0N4ZEdicVIyCnRGcmkwcWVQRmRVMWFyRUdIVlRJVllZRXZnSllxb2M

xZml2dXJjdFMwRjI4Y3JLUXZ3U0kwU0lKL1Fha1U1elMKbS91TENVZlREYWc4YlE4VEhhUFNCVUg4

QlN5b013RG5LNmNrNXU2RTRrVXgyZk5nUFd0ZW1aU0UyNEQ1NU8vYgpzZUdDeS95ZnJWbUpa

RDhHandOQUZHbDhoLzdqdmdRNmEva2tYT0xBWFB1YjhzVlJud0lEQVFBQm80R1dNSUdUCk1CM

EdBMVVkRGdRV0JCUjQxTWtmcmJaM2pqc01scWtacmU2R285VmQrREFQQmdOVkhSTUJBZjhFQ

lRBREFRSC8KTUE0R0ExVWREd0VCL3dRRUF3SUJCakJSQmdnckJnRUZCUWNCQVFSRk1FTXdRUVlJ

S3dZQkJRVUhNQUdHTlRFNQpNaTR4TmpndU9UUXVNVFk0TDJOdmJuUnliMnhzWlhJdllYWjRiMk

56Y0Q5cGMzTjFaWEp6WlhKcFlXeHVkVzFpClpYSTlNQTBHQ1NxR1NJYjNEUUVCQ3dVQUE0SUJBU

UEveVVWdlNRaU00ZUJLOXV5VEcvRXJLZHB0WktFdWFlOHEKbk1xdFUwM3B1RTREWlZxMnZZeU

9zOFdWK005ZWhMNU4wNmk0b2dnai9DYXFYY0xHQWN1ZUxvL2swdkpRVVR0cAp4U0tUKy8yO

UVnWHhqUXk2M3dCd1NnZ2ZSbFRpb3RHbzVTVmlQNGVET05LdjNZOExFRmVQbzlCb1paWklTjg

3CmJHVy9MdlJmbUxjUEs0emE2bWJiTWM4NnpYSEg4OEVRN1dJUzAxRU5LYjBSdnUzWTNsUlBv

N3ZWM2hkUXFud2EKb28vZzArazE1ejlvTkRBajhSdGR3SHRWbE5BSjVlRjhSTUo0VUJoZWlCUVlRR

0VhQk9JeU5wazFrNHFSMlpFMQpOTWFpUkNHZzRTenNkVHdZcU1BR0g1WG9IV01Kc2xIdXNKQT

AvSXF5OHB3Vk40YUJ2Q0dpCi0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0KLS0tLS1CRUdJTiBDRVJUSU

ZJQ0FURS0tLS0tCk1JSUZIVENDQkFXZ0F3SUJBZ0lRWVdUU3dMZk5Xd3phSzV4VC9uTUZYREFOQ

mdrcWhraUc5dzBCQVFzRkFEQmgKTVJRd0VnWURWUVFEREF0QmNIQldhV1YzV0NCRFFURVZNQ

k1HQTFVRUNnd01RWEJ3Vm1sbGQxZ2dTVzVqTVJBdwpEZ1lEVlFRSERBZFRaV0YwZEd4bE1STXdF

UVlEVlFRSURBcFhZWE5vYVc1bmRHOXVNUXN3Q1FZRFZRUUdFd0pWClV6QWVGdzB5TURBNU1

ERXdPRFF6TXpOYUZ3MHlOVEE0TXpFd09EUXpNek5hTUc0eElUQWZCZ05WQkFNTUdFRncKY0Za

cFpYZFlJRWx1ZEdWeWJXVmthV0YwWlNCRFFURVZNQk1HQTFVRUNnd01RWEJ3Vm1sbGQxZ2dT

VzVqTVJBdwpEZ1lEVlFRSERBZFRaV0YwZEd4bE1STXdFUVlEVlFRSURBcFhZWE5vYVc1bmRHOXVN

UXN3Q1FZRFZRUUdFd0pWClV6Q0NBU0l3RFFZSktvWklodmNOQVFFQkJRQURnZ0VQQURDQ0FR

b0NnZ0VCQUpYU2tlM0F6TzlXYUR3bXk2VnQKYmpRbXo1dDI0UnNNZUY5dVQ4S3llVzd1R0g5cnhI

VWRPMnZnU3BCQlBUdjI0VE1jdFpBaHZSdm9EM3NTUVlFLwpYMkpFK2c3SVZDUVR0VDlGOFZ3N

GMxVnByWktQNlZleE4zVENlYU5YQ1VmSGI5aW9xTFkrc3lUdy9xVldQR0h0CnJjMWZXdE0yTW11

K1NRQVhuUlRxUFJndjIcGtJUDFSbWFKWWV2T2svaGp4VU1uSHdlbHEyTTZQQXpTRzhKVksKc3dv

c1c1a3ZHSkpKdWdNR0dub0pkY0hZeVl3TlJzQnlXVWxzREhxVUk1a3lGeE50YUFiclIzT0FBRUtsb0d

UcQpCUGQxaW9kVHpuVWdKbHhZYzVvUUlJVzR6dXlmQTlXd2FMeExUYUZ6WkxTOFBlWUJmVFB

zc01yM0dzWTZXTE5NCjFva0NBd0VBQWFPQ0FjSXdnZ0crTUIwR0ExVWREZ1FXQkJSUVkzaFErT3
VON2Z0ZlI2NzBhRTRSTSs5NGJUQVAKQmdOVkhSTUJBZjhFQlRBREFRSC9NQTRHQTFVZER3RUIvd
1FFQXdJQkJqQ0JtZ1lEVllwakJJR1NNSUdQZ0JSNAoxTWtmcmJaM2pqc01scWtacmU2R285VmQrS
0ZscEdNd1lURVVNQklHQTFVRUF3d0xRWEJ3Vm1sbGQxZ2dRMEV4CkZUQVRCZ05WQkFvTURFR
ndjRlpwWlhkWUlFbHVZekVRTUE0R0ExVUVCd3dIVTJWaGRIUnNaVEVUTUJFR0ExVUUKQ0F3S1Y
yRnphR2x1WjNSdmJqRUxNQWtHQTFVRUJoTUNWVk9DRUFKRDk0SnRjenNLeS9FS1lzbnBaSkl3W
ndZRApWUjBmQkdBd1hqQmNvRnFnV0laV01Ua3lMakUyT0M0NU5DNHhOamd2WTI5dWRISnZi
R3hsY2k5aGRuaGpjbXcvClkzSnNSbWxzWlU1aGJXVTlNekF4TVRNMU9UazJPVGM1T0RBMk5UTT
NNalEwTlRVeU1qZzNOekUxTWpNeU1qY3cKTmk1amNtd3dkZ1lJS3dZQkJRVUhBUUVFYWpCb01
HWUdDQ3NHQVFVRkJ6QUJobG94T1RJdU1UWTRMamswTGpFMgpPQzlqYjI1MGNtOXNiR1Z5TD
JGMmVHOWpjM0EvYVhOemRXVnljMlZ5YVdGc2JuVnRZbVZ5UFRNd01URXpOVGs1Ck5qazNPVG
d3TmpVek56STBORFUxTWpJNE56Y3hOVEl6TWpJM01EWXdEUVlKS29aSWh2Y05BUUVMQlFBRG
dnRUIKQUpaeUoyeENMY2RkUkRlL1ZjZDJvcU56QTdsU2k0Z0tGSFdmdG41MHcwblRtejhQVHAxYj
kyNGZPYVppUGpXVApnT1YwYUwyMFNXa1ZQaG54OVM3eUU1Yy9SVkt5OVI1Smp4K0FPeWxtTE
dOb2FCOEo3WEhJYmw5OUhoWVBLVis1CmtaQ2x3ekF6ejBFU0wyUTZEOFNPRk9Na1c4cXlOa0x6
QjBCT0hpRVhrbzRkdUNlK1YyckZkd29GWXkvckZDME0KMzYzMkluMUF5UVZVM1gxQi85SzMrbG
FvS0tNK1NnbjN4Y0owM3lmcWIyTXA3UXBHTmNBdjhUNzJDMHZqV2NZMwpZTXlqMHhyUC9rZlh
mMnVsWW54ci92RTJHRERTWjRxaCtGdmRQc0lSeXdva3ltN3Q5K2E2blAzTTd1K2NXM2t1CmVidz
dNUElVK1h0TG5UVnBLN3B2VG1rPQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
tls.key:
LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFb3dJQkFBS0NBUUVBdFBLWk1lM3JLNG
Fscnc1OGhtU3NUaEFlZDQ2amVqWUlzUzBUMVhQQ2h5SGtYZ2lVCkswUk85akdoUWx5UlpqWjN
ObVlWRVRFOUFpa2VzdVplNGhsRnE1Mk9TMnZ0UmhQSi8wZWR0dkpxckdVUkoxQUIKTktSa0d6
T3dTVGJLVTlaZjh2Wm93R2ppRkd4Ym9iQXJhazd2UWxwYWhMTXdkQ2FaWDlpaVdGM25BZkJm
QUdpOAo0SWlpSVgzeWcrUkk0UHVXQ205ZzNYVUpwREJTeEtsa3UrZFZJT1JKWkU0SXhRYnNTVjd
DZXNSeGJBZXEwUkhyCmh3a3N5eDdRRlNmcU00RnRrMm5nWmtXNkhmK291TEhHMDVoYTNW
b2YyRUNmM0xRL3lMWXJkb1EvdHdVN05hV3oKV0hkSEw4TEpmdlluazVJSDgvN1JIYy9qTU1tOW
1jYWxBMlMvYndJREFRQUJBb0lCQUc1djdwNHhwcW0vR1ZwWgpqaGo4VXNtbVB1SWJkWjArWT
FtM3VIWTdtOHVjSE9SamVnTmFRcE15UVVqZWRJT2MxRytvS0UyQTdzcGx6RzNGCityMG1UWVp
ydUJCQ1ErUFVhaUhiVk4vMjNySnZVT1BjaVkvV2taNmtrb0JzNTFuNTZqL0FzcmhVaXorSnp2ZEYKT
FlwSnVPK1hjODJ3Qysrb0swMzdTMEtCdFY5UmpkUjd2R0htSEJlQ1Mwc29TK0ZscWRYcUhJcWRM
TVN1T2NTUwozQ1d6cHFxZEpNTDVUdjhTSWcveHhNRGFRSHRjSWhIMm5wYis0dDFyWGVzNTdE
eWFLQnpUZGhkeFNERXdHODJECIR3UXZsQ3lWTzJWbjk2MWxpZTZZVFUxck1xcWRPb1VJSHdpa1
BNNHV3bWVVOFp3c3VuQmRpL2Zia0RHR3FIbnEKY2ovUVdwa0NnWUVBM3p3TFZmS1JwT0VVd
GhIR0QrVDg1SkdOaEUzZEpPL3NNRy9NNVhiN2l4VE5yVUpycnltNAovOHhqZGRYUmZzVmJZSGlLY
1NVZHJoNkxIcjkzR1diREhLbDFQVllKdEhNOEVLVUJ3V0orQm91NmpBdGh2RWZwCmJ2UWwrZFd
3RWt4aUZNNmxqenZYRS80UW1tUW5yVVkxN0VmZVFoWGViMmE0K2EwOTNLZFExRE1DZ1lFQ

```
Xo0R2oKdWxhNTBxL0Qxb29mREVmOHBtMmY3anZIVmRjNVR2NDdLNHVOOUVMUGwyQlVnT0
51OFB5dHdDYURpY0xtY2lkNQpWaWo1Z1hNbzIreUxJT2FjUC9YQSsxZG9neEdiTlllWUFTVG1FMzg
2Y1RER1cwemZUUEQ5dytmYXhvSno5ZlUyCjR6cC8vM2NndEdFa1g0RURhU2J3b0RNM3VDWWJi
WkFjTHdYaUM5VUNnWUFRQy9ub29MbzFkdlNOTGZQbDJwa1AKT1NOS0dtckdKL1Q3ZWwzcE5
NZGFOUkdmM2NGdDR2cm9jR0hKNjZmV2pBdElnUUc2cG1kZXg5VktUWXNiS1ZibgpMWXMrOEh
0Y0hGaDUvZEtGZXpZQjlITmhjOFAveTgrTTZVSDlsbzNFa3ZjTjlkalMxZ05jN1l0MS9iTVljbzZYCndJYk
tpMVhrL2laWDFFczVIRkhrSHdLQmdBcXROT3RsZ3ZJZ01RZGMyL1FaMEJvUUVlYUVGT0dKelYwUE
g2UkIKN2UvZmNDYnR3ZXRQeUdtRjFycklRSVJhRHRsWXhwbDBTa0lxMXJOYUhGNTQveGtiWCs2T
3RNWHBwZjYvRjBjWAp5dTY5NFJmcDFvcmtEYWs3elJMRFh6SlBpLy9mdWc4KzkwOEJwaUVoem
1sTGpnMWUvNHArbUNOVmNwdlVpbDZ4CkFzeHhBb0dCQUk2b1lkY29qMWlGU1hwUWc0WXB
LTWlZa3ZVbitrM0VYVmh6ODFNSW5mYTRVcnRIVFkyRUNTMjgKYmQ1UnNrTXZIRXNWWFgrN3F
MTUt3U0o0bHFOVEJDYi9menc1SVpod1VsUUptZW9jMnkyOUJYUitUZ2ovYmNwMApXTE1qRzNK
cXkrdFJuaHBWVERRbXBNTU0yWUpqZk5PWjdBZS8xVld2UElMZG1Xd24vTC9lCi0tLS0tRU5EIFJTQ
SBQUklWQVRFIEtFWS0tLS0tCg==
kind: Secret
metadata:
 annotations:
 cert-manager.io/alt-names: freddidemo.trial.appviewx.com
 cert-manager.io/certificate-name: freddi-acme-cert
 cert-manager.io/common-name: ""
 cert-manager.io/ip-sans: ""
 cert-manager.io/issuer-kind: Issuer
 cert-manager.io/issuer-name: freddi-acme
 cert-manager.io/uri-sans: ""
 creationTimestamp: "2020-09-22T13:34:57Z"
 managedFields:
 - apiVersion: v1
 fieldsType: FieldsV1
 fieldsV1:
 f:data:
 .: {}
 f:ca.crt: {}
 f:tls.crt: {}
 f:tls.key: {}
 f:metadata:
 f:annotations:
 .: {}
```

```
f:cert-manager.io/alt-names: {}
f:cert-manager.io/certificate-name: {}
f:cert-manager.io/common-name: {}
f:cert-manager.io/ip-sans: {}
f:cert-manager.io/issuer-kind: {}
f:cert-manager.io/issuer-name: {}
f:cert-manager.io/uri-sans: {}
f:type: {}
manager: controller
operation: Update
time: "2020-09-22T13:39:37Z"
name: freddi-acme-secret
namespace: default
resourceVersion: "3886295"
selfLink: /api/v1/namespaces/default/secrets/freddi-acme-secret
uid: f4f4190c-55d8-448d-8426-8db2303b338d
type: kubernetes.io/tls
```

The certificate and the private key will be base64 encoded.

Chapter 12: Certificate Auto Enrollment outside containers using Certbot

Certificate Auto Enrollment outside containers using Certbot

Certbot is another standard ACME client for auto enrollment of certificates from ACME server which can be run as a command from the CLI of the Web Servers directly.

Certbot is installed as a package on the OS of the Web Servers and supports major Linux and BSD variant operating systems.

Installation

The below steps need to be executed on the base machine / OS for installation of the Certbot ACME client.

Note: The current OS used is CentOS and the steps may differ depending on the OS. Steps for installing the ACME server remain the same.

Package Installation.

- yum -y update
- yum -y install mod_ssl
- yum install bind-utils
- yum install python 3
- yum install certbot
- yum install python3-pip
- pip3 install certbot-dns-route53

If python2 install the packages like below.

- yum install certbot
- yum install --setopt=obsoletes=0 python2-certbot-dns-route53

In the case of NTP issues execute commands as below.

- yum install ntp ntpdate
- systemctl start ntpd
- systemctl enable ntpd
- ntpdate -u -s 0.centos.pool.ntp.org 1.centos.pool.ntp.org 2.centos.pool.ntp.org
- systemctl restart ntpd

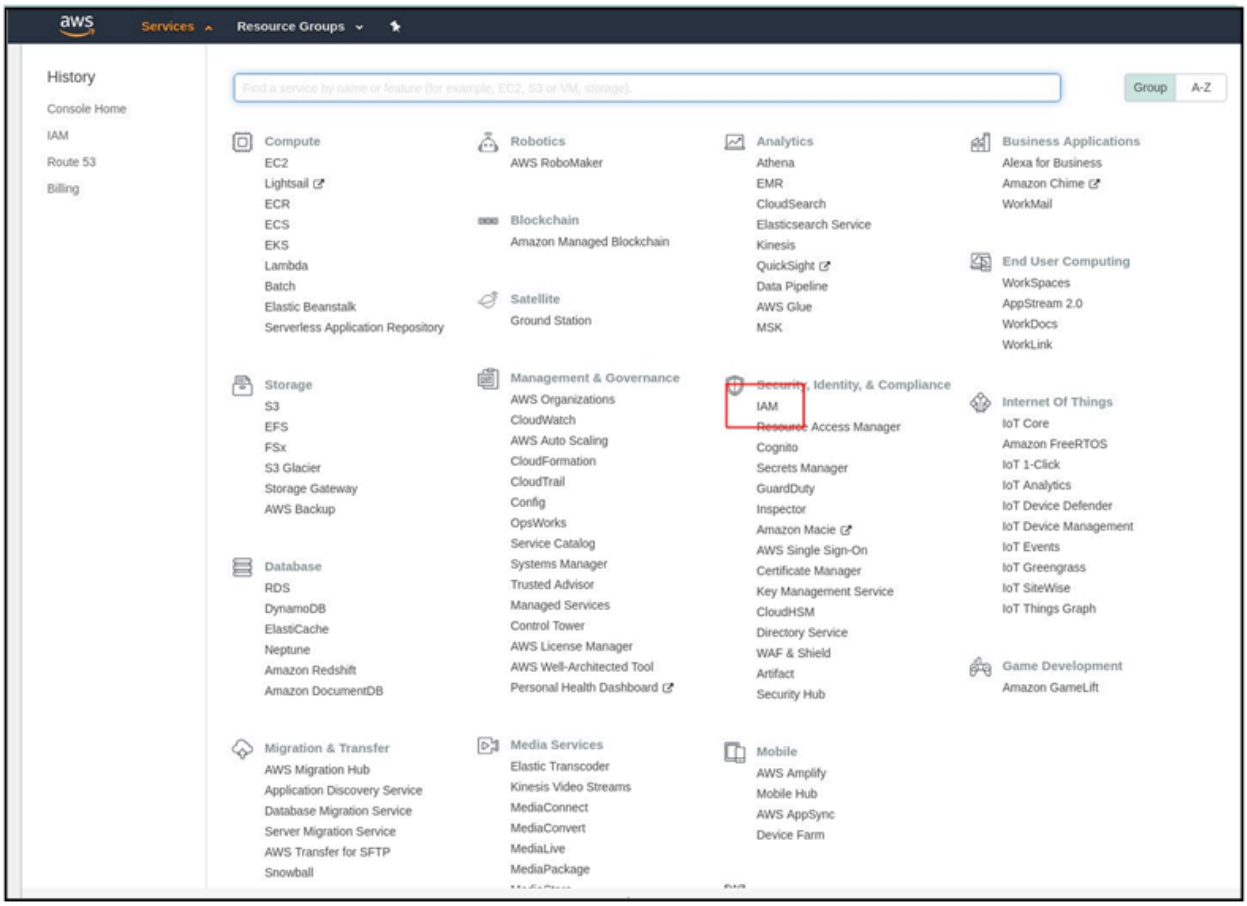
AWS-Route53 IAM Account Setup

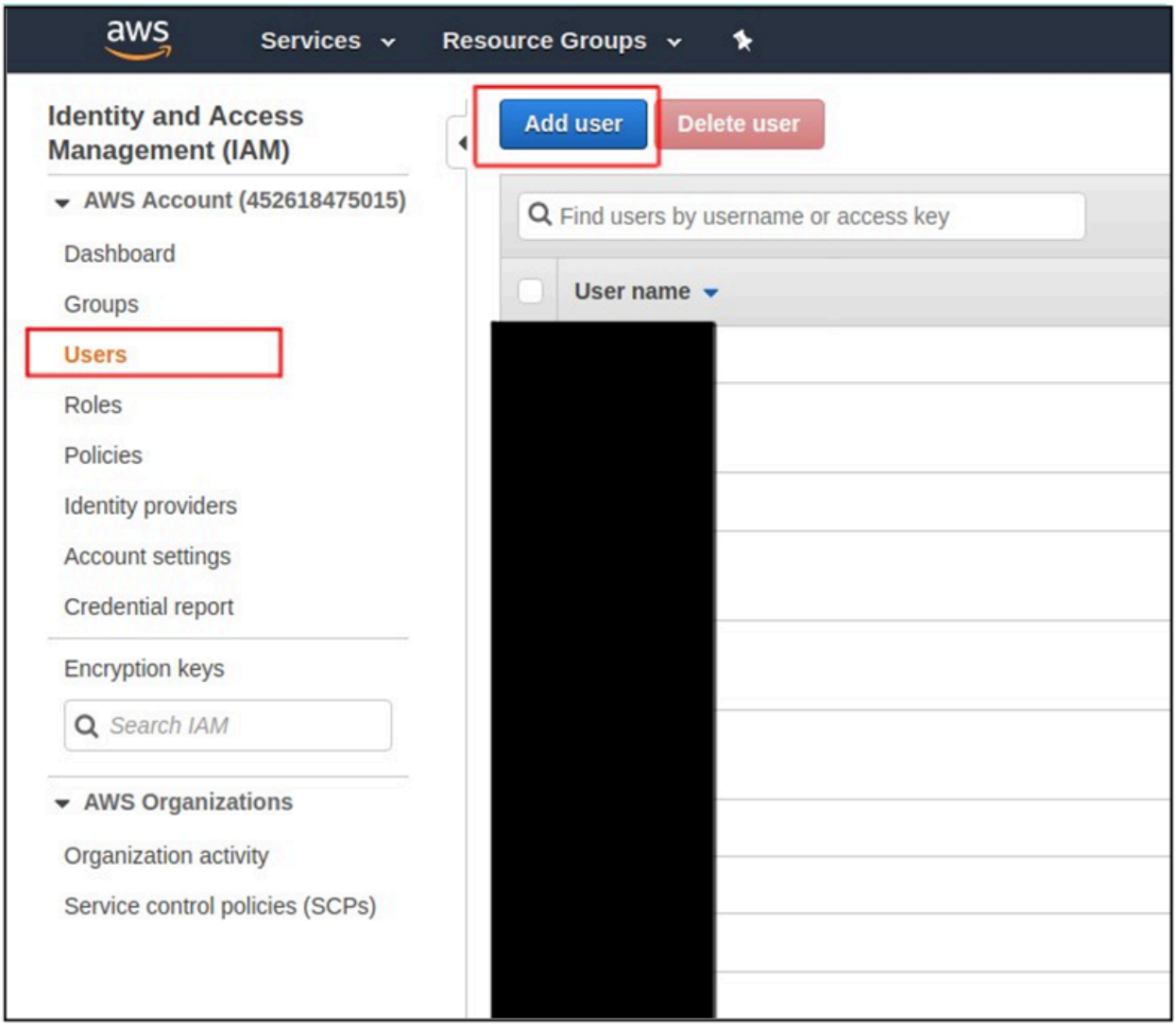
To enroll certificates via ACME the protocol supports various challenge mechanisms which are used to prove ownership of a domain so that a valid certificate can be issued for that domain.

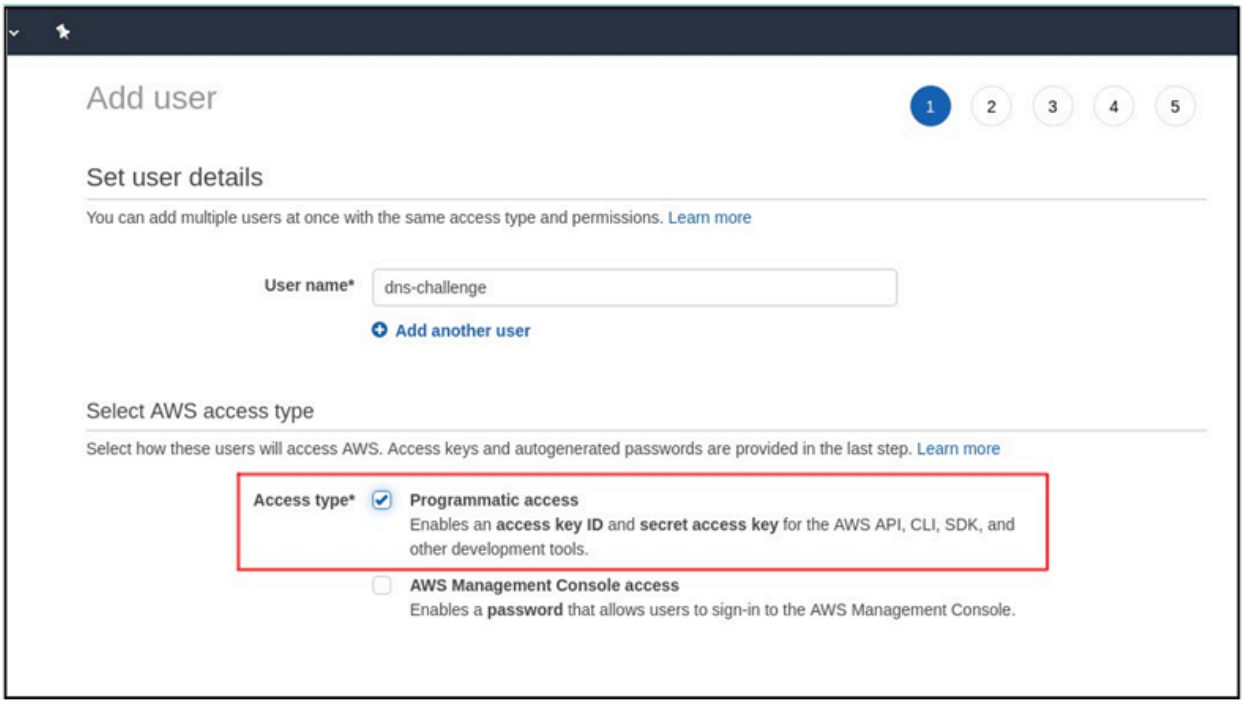
One such challenge mechanism is DNS-01. With a DNS-01 challenge, you prove ownership of a domain by proving you control its DNS records. This is done by creating a TXT record with specific content that proves you have control of the domain DNS records.

We will use Amazon Route53 to solve DNS01 ACME challenges.

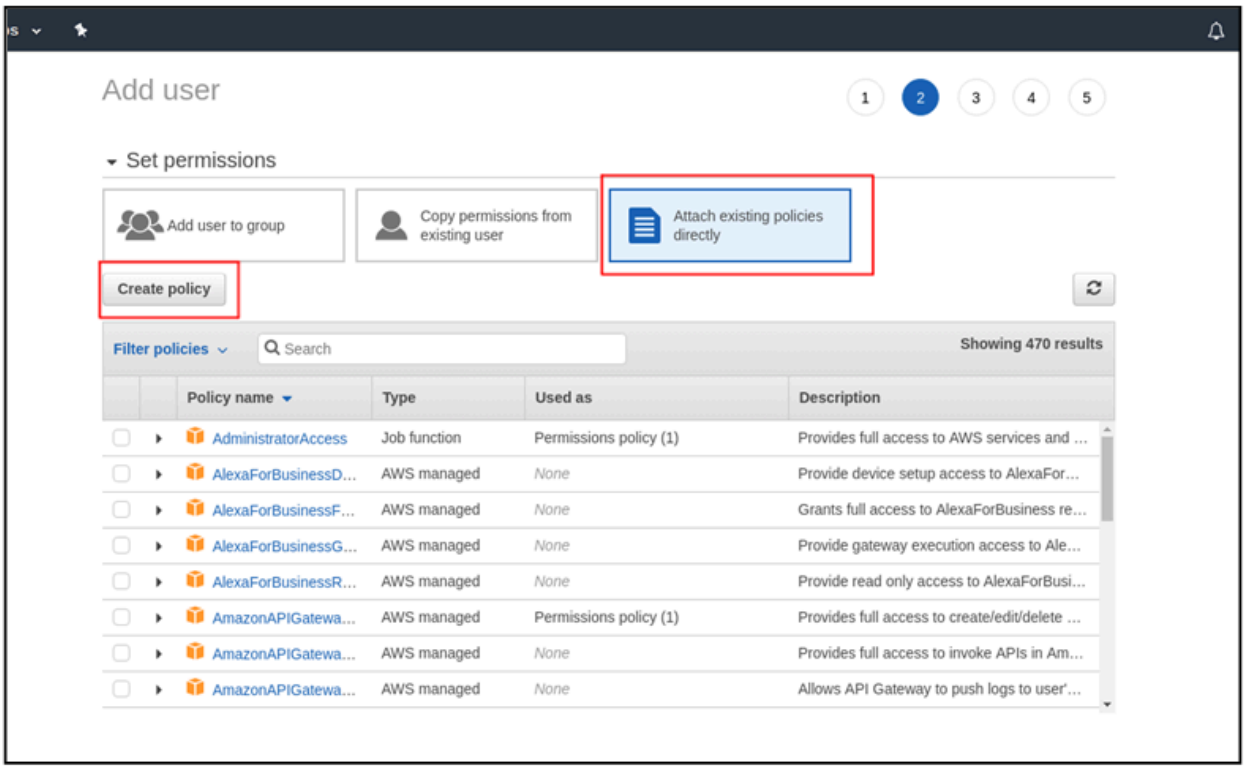
Go to IAM page and create a user.







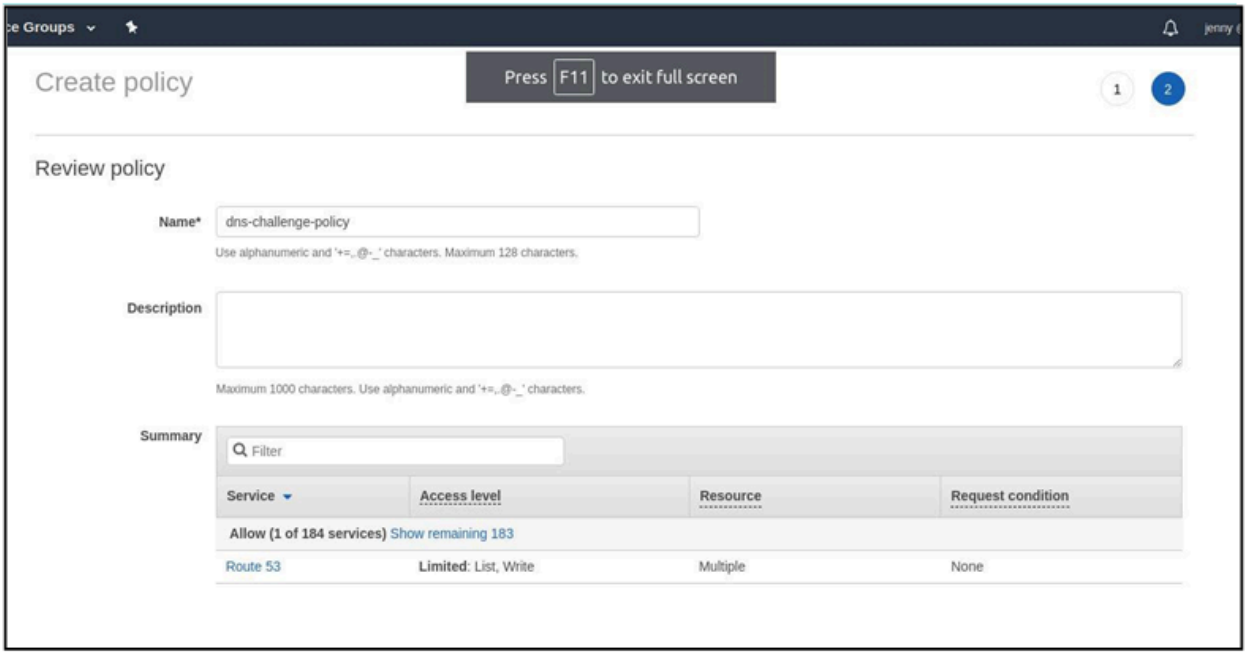
Click on next and select Attach existing policies directly and click on Create Policy. This will take you to a new page.



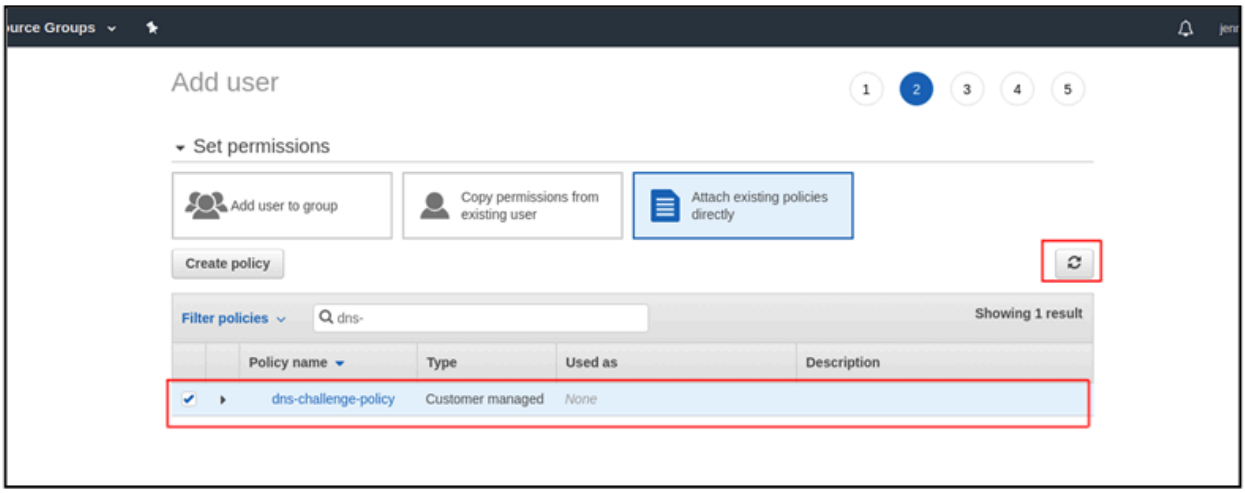
Now click on json and paste this and click Review Policy.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "route53:GetChange",
 "Resource": "arn:aws:route53:::change/*"
 },
 {
 "Effect": "Allow",
 "Action": "route53:ChangeResourceRecordSets",
 "Resource": "arn:aws:route53:::hostedzone/*"
 },
 {
 "Effect": "Allow",
 "Action": "route53:ListHostedZonesByName",
 "Resource": "*"
 }
 ]
}
```

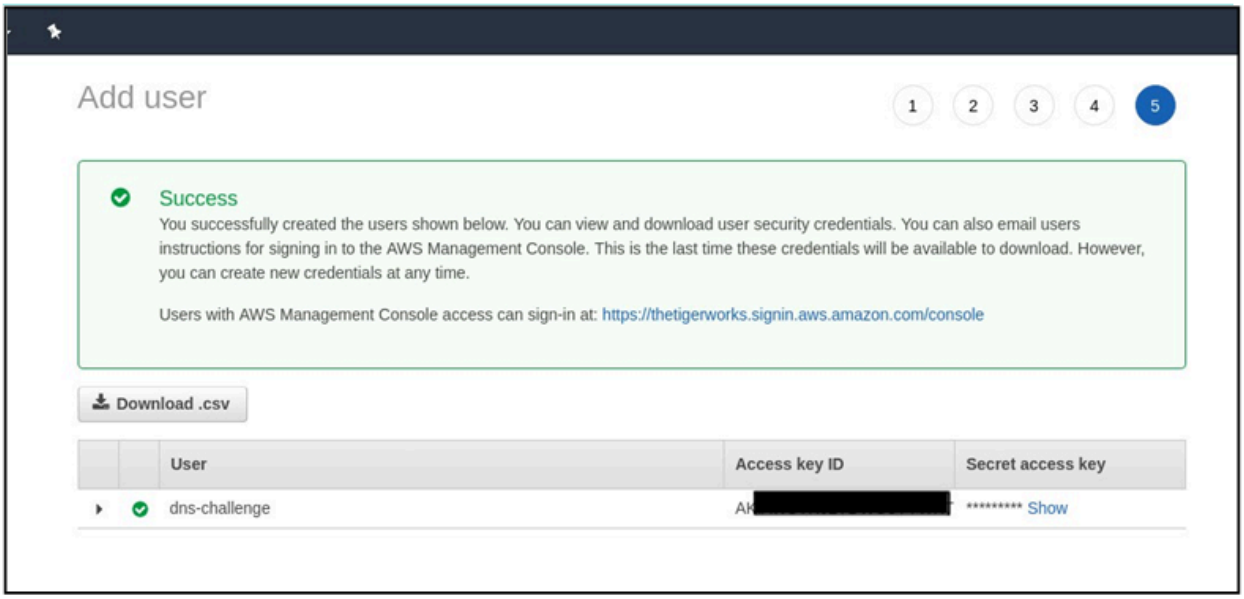
Name the policy and click Create policy.



Now go back to previous add user page, hit the refresh button and attach this policy to this user:



Click on next (tags are optional - you can ignore this) and finish the process. Download the .csv file.



Have the access_key_id and the secret_access_key copied from the policy and have it created in the linux environment where the certbot is running.

The path for storing would be from the home folder create a directory .aws

Command

```
cd ~ ; mkdir .aws ; cd .aws ;
```

Create a file called credentials and paste the content as

[default]

```
aws_access_key_id=XXXXXXXXXXXXXXXXXXXXXXXXX
aws_secret_access_key=XXXXXXXXXXXXXXXXXXX
```

Client Execution

ACME Client command

```
certbot certonly -d freddidemo.trial.appviewx.com --dns-route53 --logs-dir /etc/letsencrypt/log/
--config-dir /etc/letsencrypt/config/ --work-dir /etc/letsencrypt/work/ -m sme@appviewx.com --
no-verify-ssl --server
```

https://192.168.94.168:5300/avxapi/appviewx/acme/directory

Legends

- -d → is the domain name of the certificate
- -m → mail address
- --no-verify-ssl → for ignoring the ssl error
- --server → ACME server

1. Executing the command will create the csr and the key in the Linux environment where the certbot runs.

   The csr and the key are in the path /etc/letsencrypt/csr and /etc/letsencrypt/key

2. Now once the csr and key is created, a call to AppViewX ACME server would be done to create the nonce and further steps of acme will be sequentially executed.
3. In the process with the given aws creds certbot will make a call to make the entry of the TXT record for trial.appviewx.com.
4. AppViewX will act as an RA, before sending the csr to the CA which is configured via the policy, AppViewX ACME server will create a DNS TXT record and send it to the ACME client which is to be configured on the DNS server by the ACME client.
5. ACME Client (Certbot) will create records on the respective DNS server on the selected Hosted Zone.
6. AppViewX ACME server now will validate the create TXT challenge by performing an nslookup from the AppViewX server and if the DNS record is propagated AppViewX ACME server considers the requested certificate domain is an authorized domain.
7. AppViewX will now pass the CSR to the CA, get the certificate approved and send it to the ACME Client.
8. Certificates can be found in the directory

   /etc/letsencrypt/config/live/