appviewX

CLMaaS Deployment Guide

Version: 2021.1.0

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 2021.1.0	September 2021

About this Guide

This guide outlines the deployment of the AppViewX CLMaaS using the CI/CD pipeline and configuring the backup and restore processes.

Audience

This guide is intended for AppViewX's Site Reliability Engineers (SRE) team.

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 text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`codeblock`	Indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Chapter 1: Onboarding

Onboarding

To onboard a customer:

Gather the following information from the customer:

- Total no. of nodes required
- Details of the latest AMI with the latest installation package
- Volume size required
- Instance type
- Customer email address(es) for the alerts
- Data retention period

Once this information has been gathered, it can be used to configure the CloudFormation template and trigger the CI/CD pipeline for the CLMaaS deployment.

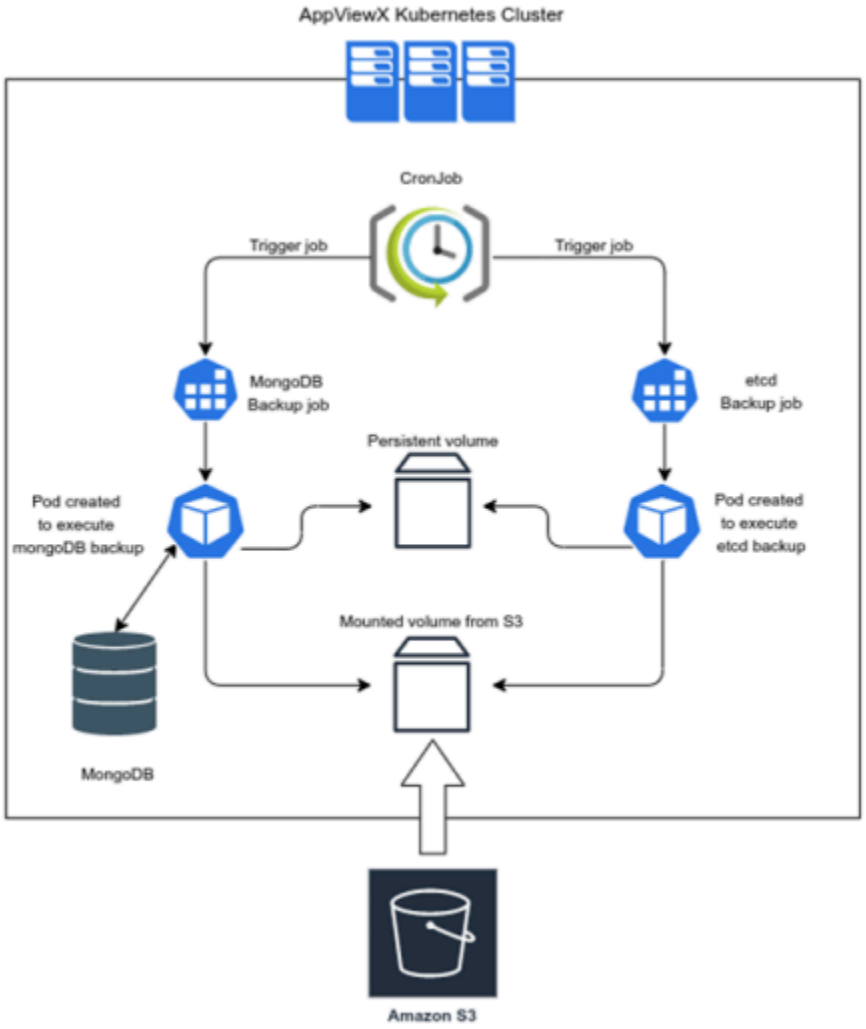
Chapter 2: CLMaaS Deployment

Overview: Backup and Restore for the CERT+ CLMaaS

For the CERT+ CLMaaS deployment, AppViewX enables built-in processes to backup MongoDB and etcd data periodically for fast recovery in the event of an outage. For the backed up data, a user-specified data retention period is specified. The implementation comes with mechanisms for encrypting the backup and restoring it seamlessly.

Backup and Restore: Design and Implementation

The image below is a graphical representation of how the backup and restore processes are implemented in the CERT+ CLMaaS deployed.



Backup Frequency

For Active Customers

For active customers, three backups are taken in a day, in an 8-hour interval, and stored in an AWS S3 bucket as follows:

- Day 1 backup:
  - Backup 1
  - Backup 2
  - Backup 3 (latest backup from Day 1)
- Day 2 backup:
  - Backup 3 (latest backup from Day 1)
  - Backup 4
  - Backup 5
  - Backup 6 (latest backup from Day 2)

The backed-up data is:

- Stored in the AWS S3 bucket
- Encrypted using the encryption method supported by the designated S3 bucket

NOTE: AWS S3 lifecycle policy, encryption method, and bucket policy must be configured via the CI/CD process.

For Customer Churns

In the event of customer churns, AppViewX will take the latest backup copy from the S3 bucket and archive it in the AWS Glacier. The churned-up data will be retained in the AWS Glacier for 30 days, as per the default data retention policy.

The churned-up data is:

- Stored in the AWS Glacier
- Encrypted using the encryption method supported by the AWS glacier by default

Backup and Recovery Policy for the AWS S3 Bucket and the AWS Glacier

AppViewX's backup and recovery policy provisions to:

- Define the backup interval for applications, vaults, logs, and so on.
- Configure and enable custom backup intervals as policy (example: 1 backup per day or 3 backups per day)
- Configure the lifecycle rotation policy for the AWS S3 bucket.
- Configure the recovery process to recover data from the AWS S3 bucket and the AWS Glacier.

CI/CD Pipeline for the CLMaaS Deployment

For AppViewX, the CI/CD Pipeline is a combination of processes that enable Continuous Integration and Continuous Deployment of the CERT+ CLMaaS for its tenants. Implementation of the CI/CD pipeline automates the building, testing, monitoring, and the deployment of applications, thus making it the backbone of DevOps operations.

Continuous Integration, or CI, integrates the application code with the code repository, which then deploys it as part of the Continuous Deployment process. While CI is responsible for integrating all code-related updates, CD is a more complex process that is in-charge of testing, staging, and deploying these code updates.

Deployment Prerequisites

1. Create a S3 bucket in the newly created AWS account.
2. Download all the files and folders from *https://saas-infra-cf-appviewx.s3.amazonaws.com/templates/* to the newly created S3 bucket and follow the same folder hierarchy.

3. Ensure both master node (ami-0f0a7816afa53b8df), worker node (ami-081868575fbc496af) and bastion host (ami-04db49c0fb2215364) AMI's are accessible from the 767865944424.

   Note: Master node and worker node AMI id will change once repackage is performed.

4. Create a pem file and upload it under pem-files/

5. Create an ACM certificate from the Certificate Manager.

6. Ensure there is a hosted zone and add it to Route53.

7. Ensure the two policies are added to the IAM user.
   - Policies are listed in JSON format, so copy and create them as per the below naming conventions:
     - saas_infra_resources_policy_1
     - saas_infra_resources_policy_2

> **Note:** There will be some limitations to the length of the policy file depending upon the AWS account, so split multiple policies into one by following the same naming conventions.

saas_infra_resources_policy_1

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "kms:*"
         ],
         "Resource": "*"
      },
      {
         "Effect": "Allow",
         "Action": [
            "iam:AttachRolePolicy",
            "iam:CreateRole",
            "iam:PutRolePolicy",
            "iam:GetRole",
            "iam:DeleteRole",
            "iam:GetRolePolicy",
            "iam:TagRole",
            "iam:PassRole",
            "iam:ListRoles",
```

```
            "iam:ListRoleTags",
            "iam:UntagRole",
            "iam:GetInstanceProfile",
            "iam:CreateInstanceProfile",
            "iam:AddRoleToInstanceProfile",
            "iam:RemoveRoleFromInstanceProfile",
            "iam:DeleteRolePolicy",
            "iam:DeleteInstanceProfile"
        ],
        "Resource": "*"
    },
    {
        "Action": "ec2:*",
        "Effect": "Allow",
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": "elasticloadbalancing:*",
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": "cloudwatch:*",
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": "autoscaling:*",
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": "iam:CreateServiceLinkedRole",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
```

```
            "iam:AWSServiceName": [
                "autoscaling.amazonaws.com",
                "ec2scheduled.amazonaws.com",
                "elasticloadbalancing.amazonaws.com",
                "spot.amazonaws.com",
                "spotfleet.amazonaws.com",
                "transitgateway.amazonaws.com"
            ]
        }
    }
},
{
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "iam:ChangePassword"
    ],
    "Resource": [
        "arn:aws:iam::*:user/${aws:username}"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "iam:GetAccountPasswordPolicy"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:AcceptVpcPeeringConnection",
        "ec2:AcceptVpcEndpointConnections",
```

```
"ec2:AllocateAddress",
"ec2:AssignIpv6Addresses",
"ec2:AssignPrivateIpAddresses",
"ec2:AssociateAddress",
"ec2:AssociateDhcpOptions",
"ec2:AssociateRouteTable",
"ec2:AssociateSubnetCidrBlock",
"ec2:AssociateVpcCidrBlock",
"ec2:AttachClassicLinkVpc",
"ec2:AttachInternetGateway",
"ec2:AttachNetworkInterface",
"ec2:AttachVpnGateway",
"ec2:AuthorizeSecurityGroupEgress",
"ec2:AuthorizeSecurityGroupIngress",
"ec2:CreateCarrierGateway",
"ec2:CreateCustomerGateway",
"ec2:CreateDefaultSubnet",
"ec2:CreateDefaultVpc",
"ec2:CreateDhcpOptions",
"ec2:CreateEgressOnlyInternetGateway",
"ec2:CreateFlowLogs",
"ec2:CreateInternetGateway",
"ec2:CreateLocalGatewayRouteTableVpcAssociation",
"ec2:CreateNatGateway",
"ec2:CreateNetworkAcl",
"ec2:CreateNetworkAclEntry",
"ec2:CreateNetworkInterface",
"ec2:CreateNetworkInterfacePermission",
"ec2:CreateRoute",
"ec2:CreateRouteTable",
"ec2:CreateSecurityGroup",
"ec2:CreateSubnet",
"ec2:CreateTags",
"ec2:CreateVpc",
"ec2:CreateVpcEndpoint",
"ec2:CreateVpcEndpointConnectionNotification",
"ec2:CreateVpcEndpointServiceConfiguration",
```

```
"ec2:CreateVpcPeeringConnection",
"ec2:CreateVpnConnection",
"ec2:CreateVpnConnectionRoute",
"ec2:CreateVpnGateway",
"ec2:DeleteCarrierGateway",
"ec2:DeleteCustomerGateway",
"ec2:DeleteDhcpOptions",
"ec2:DeleteEgressOnlyInternetGateway",
"ec2:DeleteFlowLogs",
"ec2:DeleteInternetGateway",
"ec2:DeleteLocalGatewayRouteTableVpcAssociation",
"ec2:DeleteNatGateway",
"ec2:DeleteNetworkAcl",
"ec2:DeleteNetworkAclEntry",
"ec2:DeleteNetworkInterface",
"ec2:DeleteNetworkInterfacePermission",
"ec2:DeleteRoute",
"ec2:DeleteRouteTable",
"ec2:DeleteSecurityGroup",
"ec2:DeleteSubnet",
"ec2:DeleteTags",
"ec2:DeleteVpc",
"ec2:DeleteVpcEndpoints",
"ec2:DeleteVpcEndpointConnectionNotifications",
"ec2:DeleteVpcEndpointServiceConfigurations",
"ec2:DeleteVpcPeeringConnection",
"ec2:DeleteVpnConnection",
"ec2:DeleteVpnConnectionRoute",
"ec2:DeleteVpnGateway",
"ec2:DescribeAccountAttributes",
"ec2:DescribeAddresses",
"ec2:DescribeAvailabilityZones",
"ec2:DescribeCarrierGateways",
"ec2:DescribeClassicLinkInstances",
"ec2:DescribeCustomerGateways",
"ec2:DescribeDhcpOptions",
"ec2:DescribeEgressOnlyInternetGateways",
```

```
"ec2:DescribeFlowLogs",
"ec2:DescribeInstances",
"ec2:DescribeInternetGateways",
"ec2:DescribeIpv6Pools",
"ec2:DescribeLocalGatewayRouteTables",
"ec2:DescribeLocalGatewayRouteTableVpcAssociations",
"ec2:DescribeKeyPairs",
"ec2:DescribeMovingAddresses",
"ec2:DescribeNatGateways",
"ec2:DescribeNetworkAcls",
"ec2:DescribeNetworkInterfaceAttribute",
"ec2:DescribeNetworkInterfacePermissions",
"ec2:DescribeNetworkInterfaces",
"ec2:DescribePrefixLists",
"ec2:DescribeRouteTables",
"ec2:DescribeSecurityGroupReferences",
"ec2:DescribeSecurityGroupRules",
"ec2:DescribeSecurityGroups",
"ec2:DescribeStaleSecurityGroups",
"ec2:DescribeSubnets",
"ec2:DescribeTags",
"ec2:DescribeVpcAttribute",
"ec2:DescribeVpcClassicLink",
"ec2:DescribeVpcClassicLinkDnsSupport",
"ec2:DescribeVpcEndpointConnectionNotifications",
"ec2:DescribeVpcEndpointConnections",
"ec2:DescribeVpcEndpoints",
"ec2:DescribeVpcEndpointServiceConfigurations",
"ec2:DescribeVpcEndpointServicePermissions",
"ec2:DescribeVpcEndpointServices",
"ec2:DescribeVpcPeeringConnections",
"ec2:DescribeVpcs",
"ec2:DescribeVpnConnections",
"ec2:DescribeVpnGateways",
"ec2:DetachClassicLinkVpc",
"ec2:DetachInternetGateway",
"ec2:DetachNetworkInterface",
```

```
"ec2:DetachVpnGateway",
"ec2:DisableVgwRoutePropagation",
"ec2:DisableVpcClassicLink",
"ec2:DisableVpcClassicLinkDnsSupport",
"ec2:DisassociateAddress",
"ec2:DisassociateRouteTable",
"ec2:DisassociateSubnetCidrBlock",
"ec2:DisassociateVpcCidrBlock",
"ec2:EnableVgwRoutePropagation",
"ec2:EnableVpcClassicLink",
"ec2:EnableVpcClassicLinkDnsSupport",
"ec2:ModifyNetworkInterfaceAttribute",
"ec2:ModifySecurityGroupRules",
"ec2:ModifySubnetAttribute",
"ec2:ModifyVpcAttribute",
"ec2:ModifyVpcEndpoint",
"ec2:ModifyVpcEndpointConnectionNotification",
"ec2:ModifyVpcEndpointServiceConfiguration",
"ec2:ModifyVpcEndpointServicePermissions",
"ec2:ModifyVpcPeeringConnectionOptions",
"ec2:ModifyVpcTenancy",
"ec2:MoveAddressToVpc",
"ec2:RejectVpcEndpointConnections",
"ec2:RejectVpcPeeringConnection",
"ec2:ReleaseAddress",
"ec2:ReplaceNetworkAclAssociation",
"ec2:ReplaceNetworkAclEntry",
"ec2:ReplaceRoute",
"ec2:ReplaceRouteTableAssociation",
"ec2:ResetNetworkInterfaceAttribute",
"ec2:RestoreAddressToClassic",
"ec2:RevokeSecurityGroupEgress",
"ec2:RevokeSecurityGroupIngress",
"ec2:UnassignIpv6Addresses",
"ec2:UnassignPrivateIpAddresses",
"ec2:UpdateSecurityGroupRuleDescriptionsEgress",
"ec2:UpdateSecurityGroupRuleDescriptionsIngress"
```

```
        ],
        "Resource": "*"
      },
      {
        "Action": [
          "sns:*"
        ],
        "Effect": "Allow",
        "Resource": "*"
      }
    ]
}
```

saas_infra_resources_policy_2

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "route53:*",
        "route53domains:*",
        "cloudfront:ListDistributions",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticbeanstalk:DescribeEnvironments",
        "s3:ListBucket",
        "s3:GetBucketLocation",
        "s3:GetBucketWebsite",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeRegions",
        "sns:ListTopics",
        "sns:ListSubscriptionsByTopic",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:GetMetricStatistics"
      ],
      "Resource": "*"
    },
```

```
{
    "Effect": "Allow",
    "Action": "apigateway:GET",
    "Resource": "arn:aws:apigateway:*::/domainnames"
},
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:*"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:DescribeStacks",
        "cloudformation:ListStackResources",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricData",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "kms:ListAliases",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListRolePolicies",
        "iam:ListRoles",
        "lambda:*",
        "logs:DescribeLogGroups",
        "states:DescribeStateMachine",
        "states:ListStateMachines",
        "tag:GetResources",
        "xray:GetTraceSummaries",
        "xray:BatchGetTraces"
```

```
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": "iam:PassRole",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "iam:PassedToService": "lambda.amazonaws.com"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "logs:DescribeLogStreams",
            "logs:GetLogEvents",
            "logs:FilterLogEvents"
        ],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/lambda/*"
    },
    {
        "Action": [
            "autoscaling:Describe*",
            "cloudwatch:*",
            "logs:*",
            "sns:*",
            "iam:GetPolicy",
            "iam:GetPolicyVersion",
            "iam:GetRole"
        ],
        "Effect": "Allow",
        "Resource": "*"
    },
    {
        "Effect": "Allow",
```

```
        "Action": "iam:CreateServiceLinkedRole",
        "Resource": "arn:aws:iam::*:role/aws-service-role/events.amazonaws.com/AWSServiceRoleForCloudWatchEvents*",
        "Condition": {
          "StringLike": {
            "iam:AWSServiceName": "events.amazonaws.com"
          }
        }
      }
  ]
}
```

How to Launch Stack

How to Launch a Stack

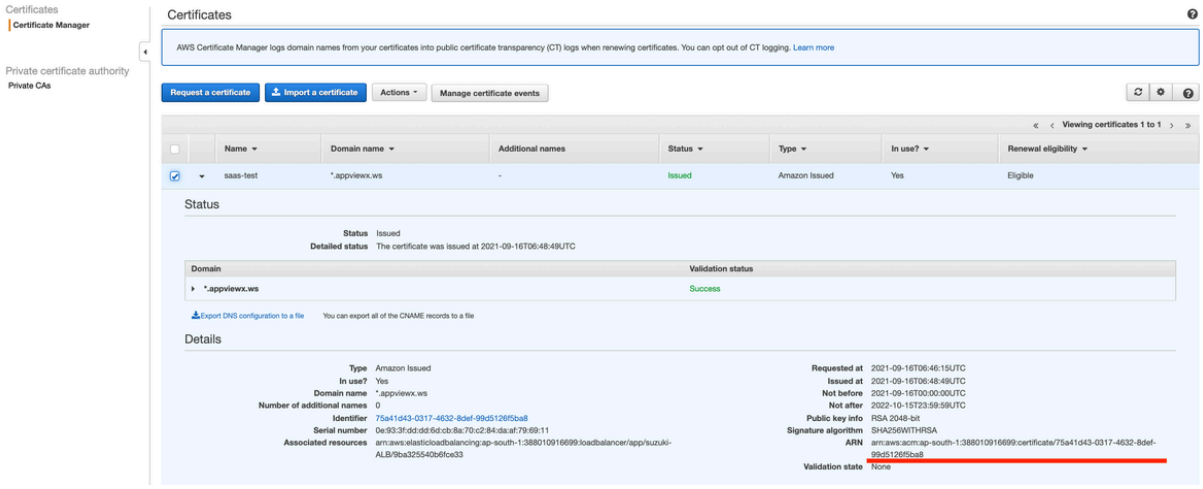
1. Login to the AWS console as a root or IAM user and launch the CloudFormation service and select **With new resources (standard)**.

2. Enter the master CF template URL which is stored in the S3 bucket and click **Next**.

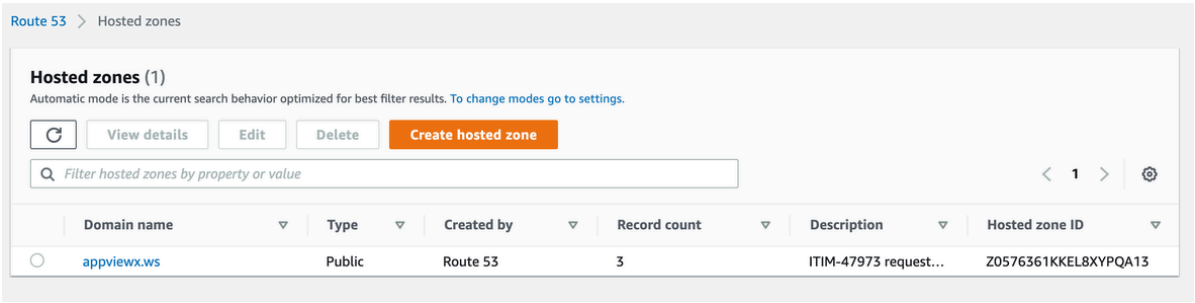
Parameters Section

1. Enter the Stack Name (use a short form of the client's name)
2. Template S3 Bucket URI
   - Enter the S3 bucket template URI, which contains the CF templates.
3. Template S3 Bucket Name
   - Enter the S3 bucket name, which contains the CF templates
4. ACM Certificate ARN

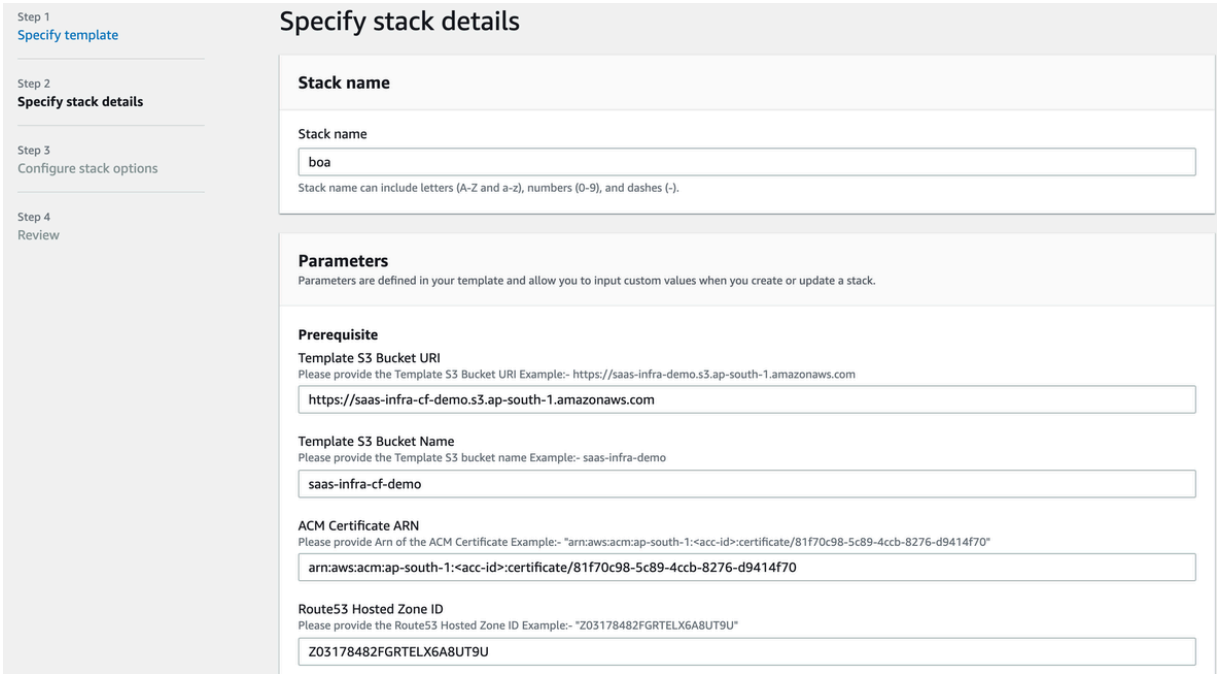
- Enter the certificate manager ARN



5. Route53 Hosted Zone ID
   - Enter the hosted Zone ID from the Route53 service

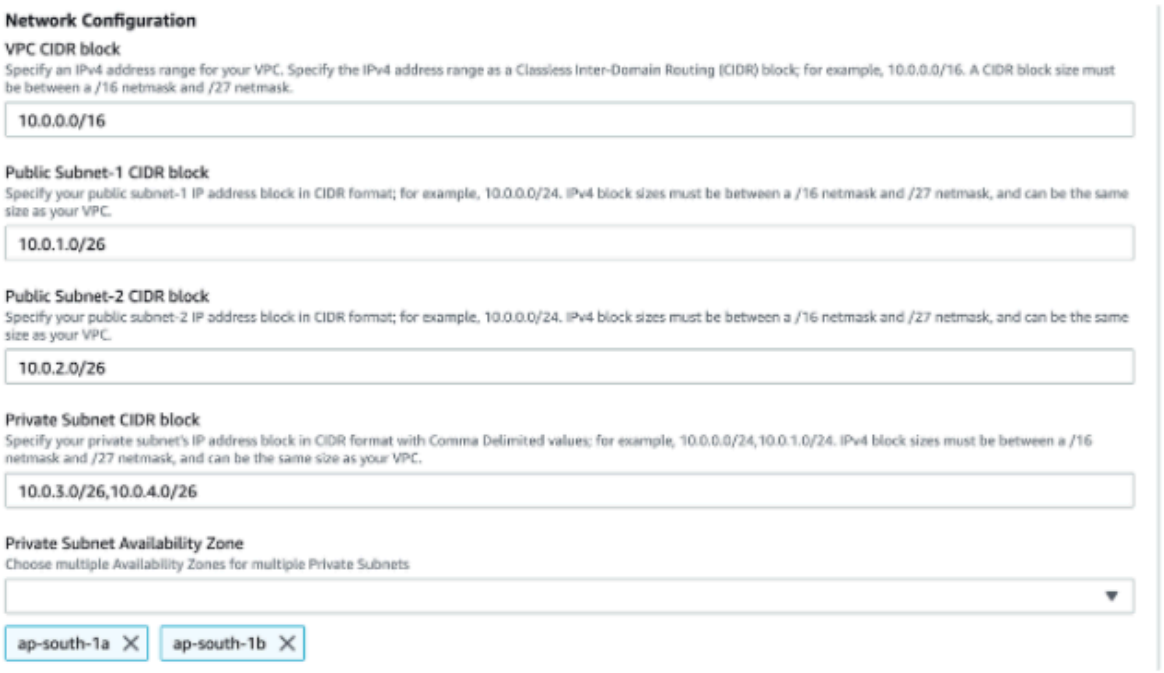


6. The prerequisites reference screenshot is shown below:



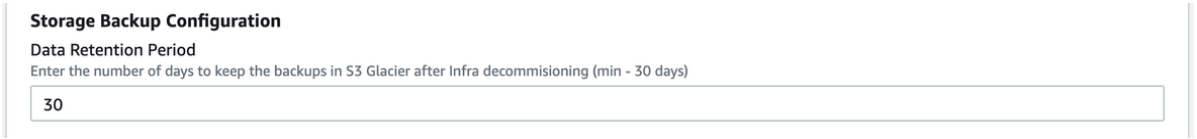
7. Customer Details:
   - Enter the Customer Name.

8. Network Configuration:
   a. VPC CIDR block
   b. Public subnet-1 CIDR block
   c. Public subnet-2 CIDR block
   d. Private subnets CIDR block
   e. Private subnets Availability Zone

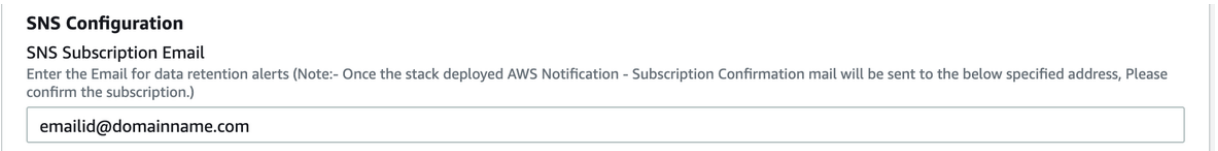


9. EC2 Configuration:
   a. **Single or Multinode configuration**
      i. Enter the Number of Master Nodes required value as 1 and
      ii. Number of Worker Nodes required as 0
      iii. The value in the worker node sections can remain the same.
   b. Master Node Amazon Machine Image(AMI)
   c. Number of Master Nodes required
   d. Master node Instance Type
   e. Master node Volume size
   f. Worker Node Amazon Machine Image(AMI)
   g. Number of Worker Nodes required
   h. Worker node Instance Type
   i. Worker node volume size
   j. Volume size of Worker Nodes
   k. Key pair Name
      i. Ensure the pem file is uploaded to the S3 bucket under <bucketname>/pem-files
      ii. If a new key pair needs to be used, then create and add it to the template S3 bucket and it will display on the dropdown list.

10. Storage Backup Configuration
    - Data Retention period(in days) after decommission of product

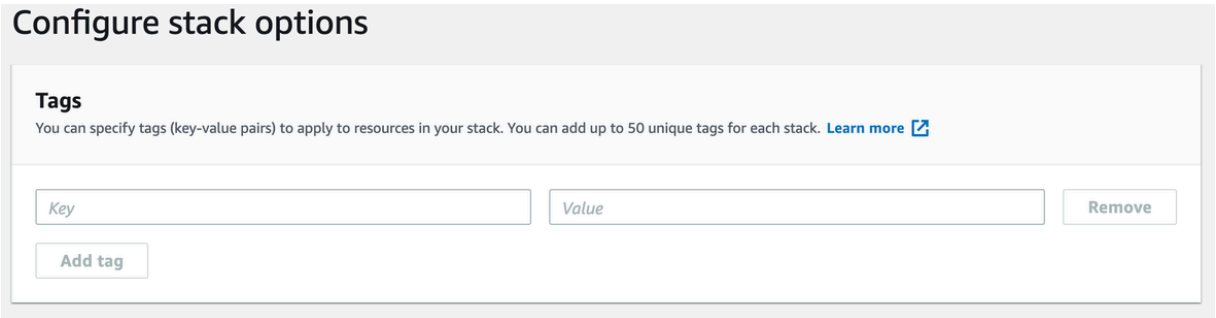


11. SNS Configuration
    - Email alert notification for all product monitoring



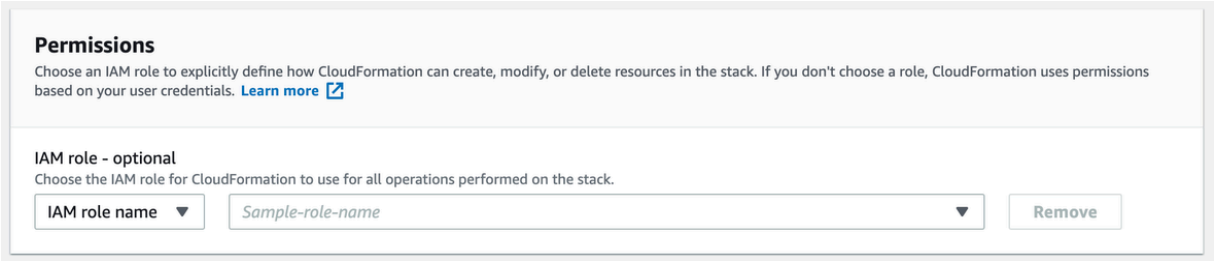
12. Parameters for AppViewX Installation
    a. Appviewx User Password
    b. Version: Appviewx product installation version
    c. Installation Path: Absolute path for product installation
    d. ELK: Enable or Disable the monitoring
    e. INSIGHT: Enable or Disable insight
    f. SYSLOG: Enable or Disable the syslog

13. After entering all the above values, click next to “Configure stack Options”.
14. Add tags for feature groupings, else they can be left blank.



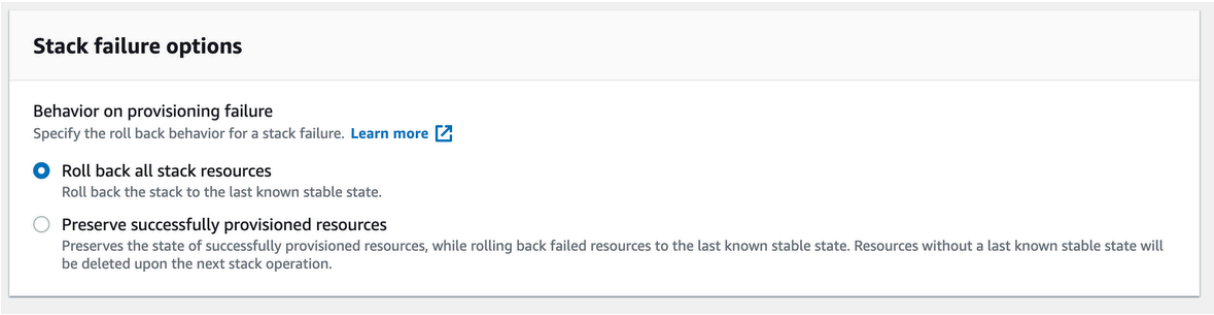
Permissions

It can be left blank as it is already handled inside the template.



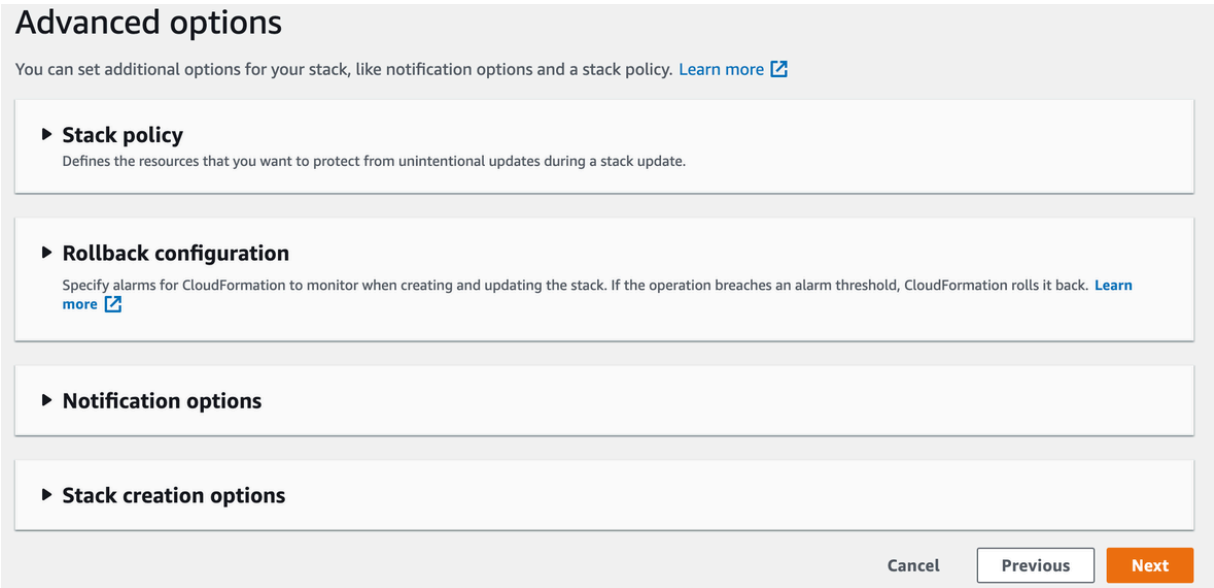
Stack Failure Options

Select “Preserve successfully provisioned resources“ only if you want to debug the stack failures, else leave it to default “Roll back all stack resources“.

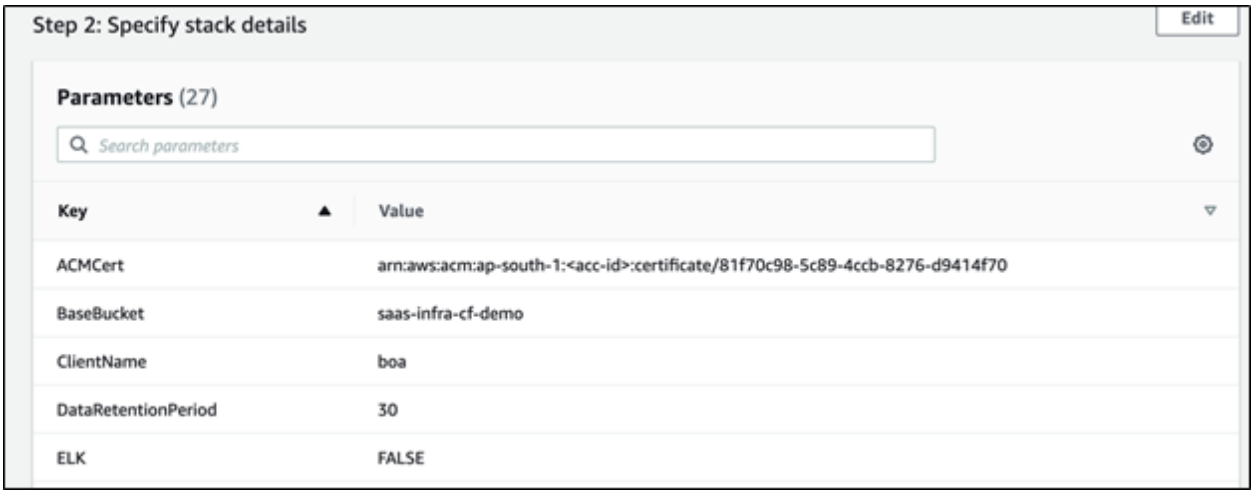


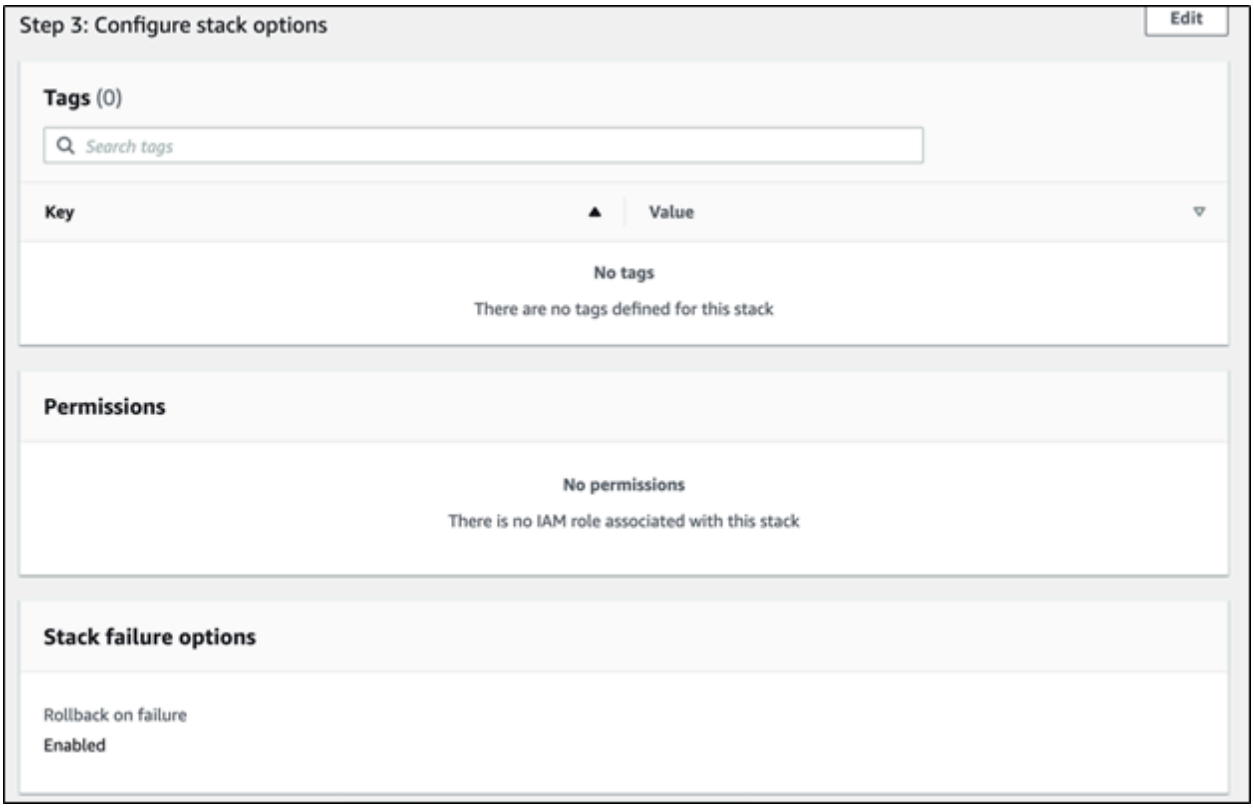
Advanced Options

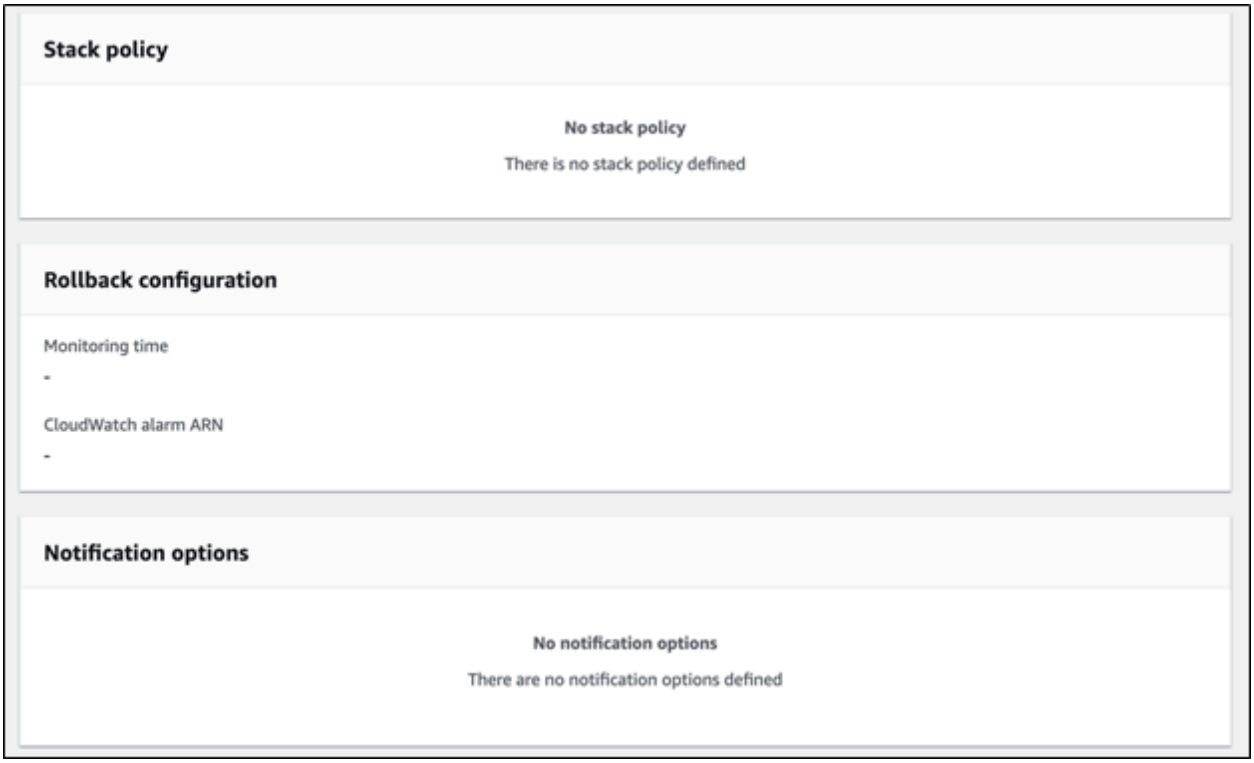
Proceed to the next page to review the values.



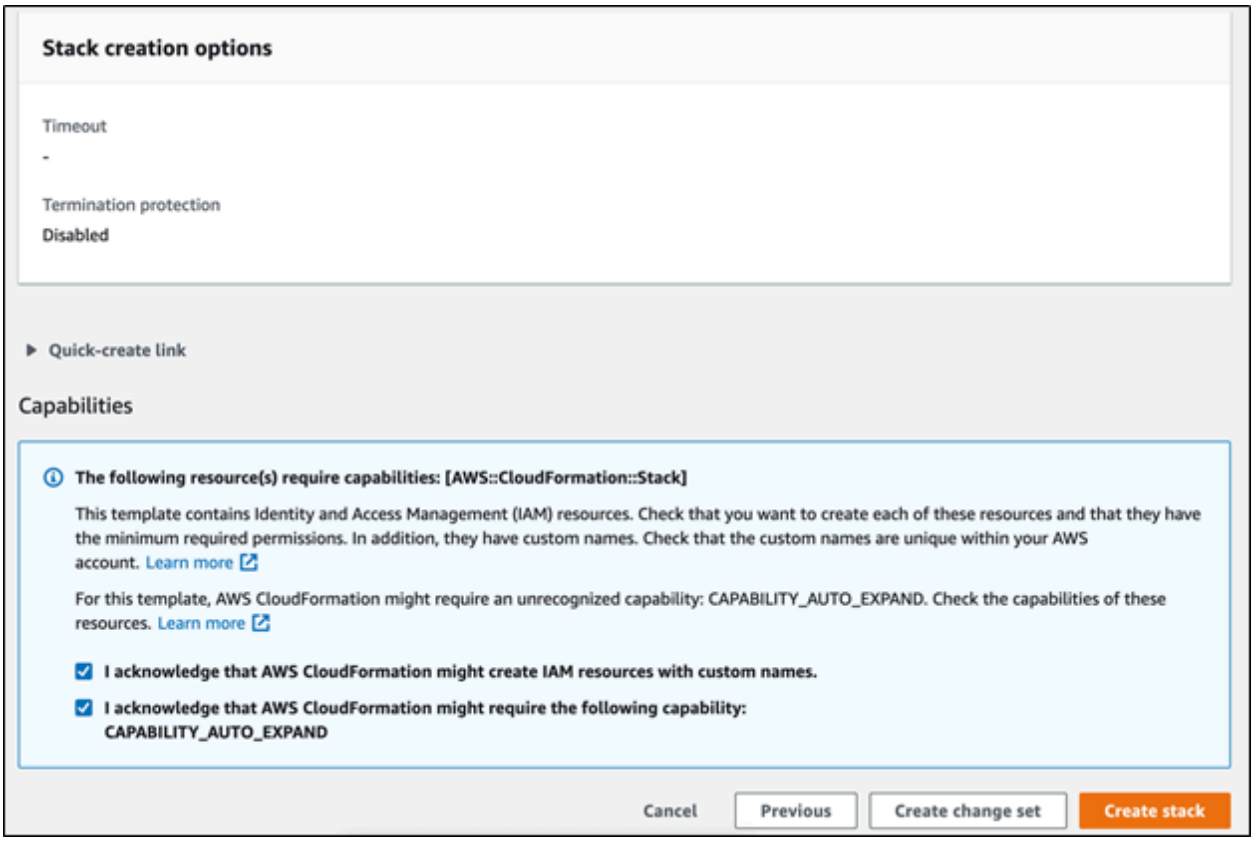
Review Stack



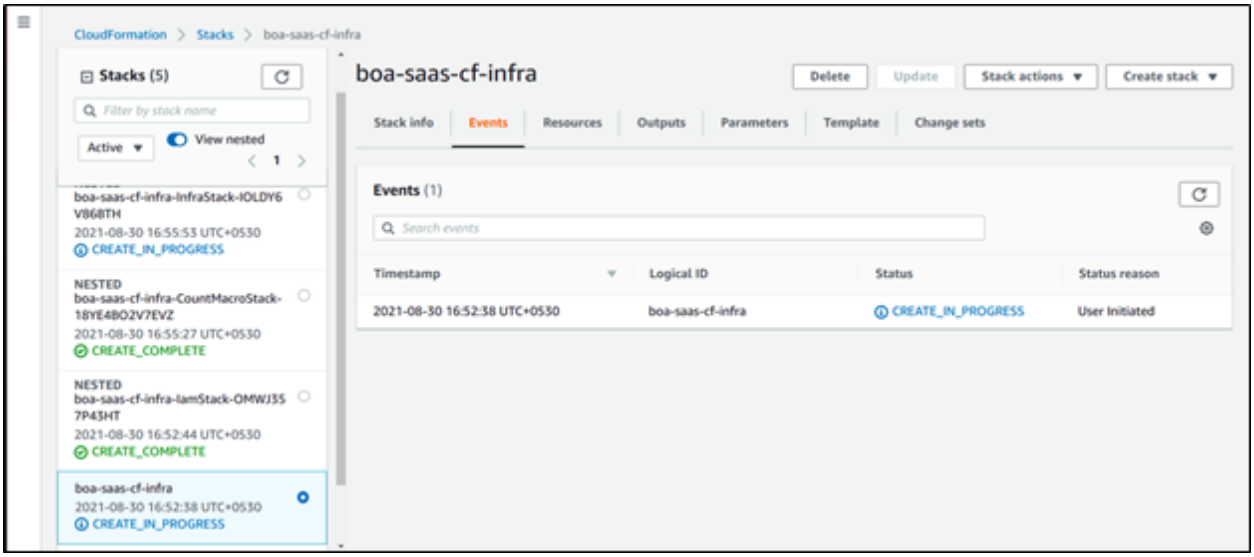




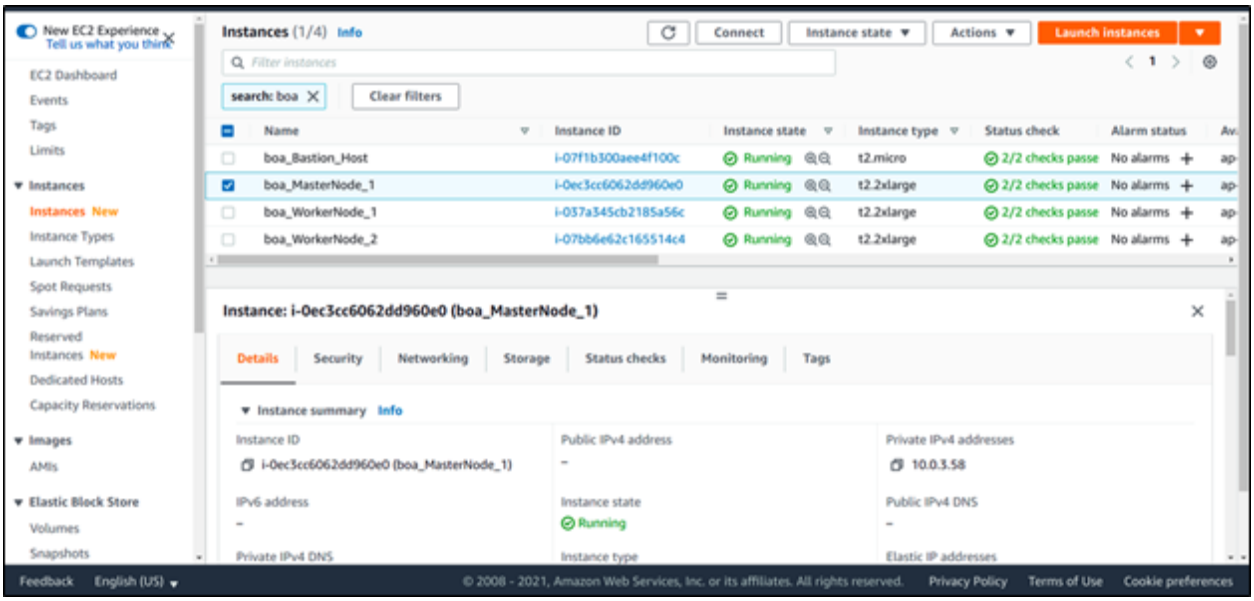
Select the two checkboxes below and click **Create stack**.



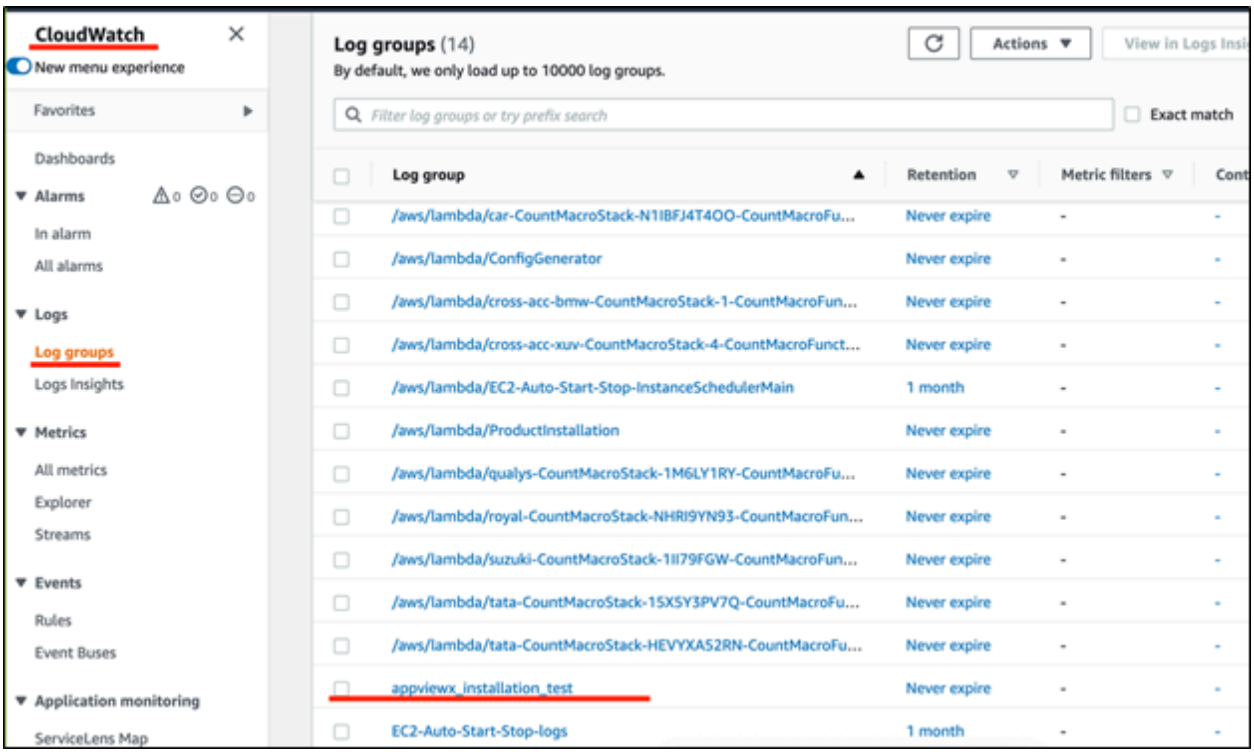
Once the stack creation gets started, you can see progress as shown below.



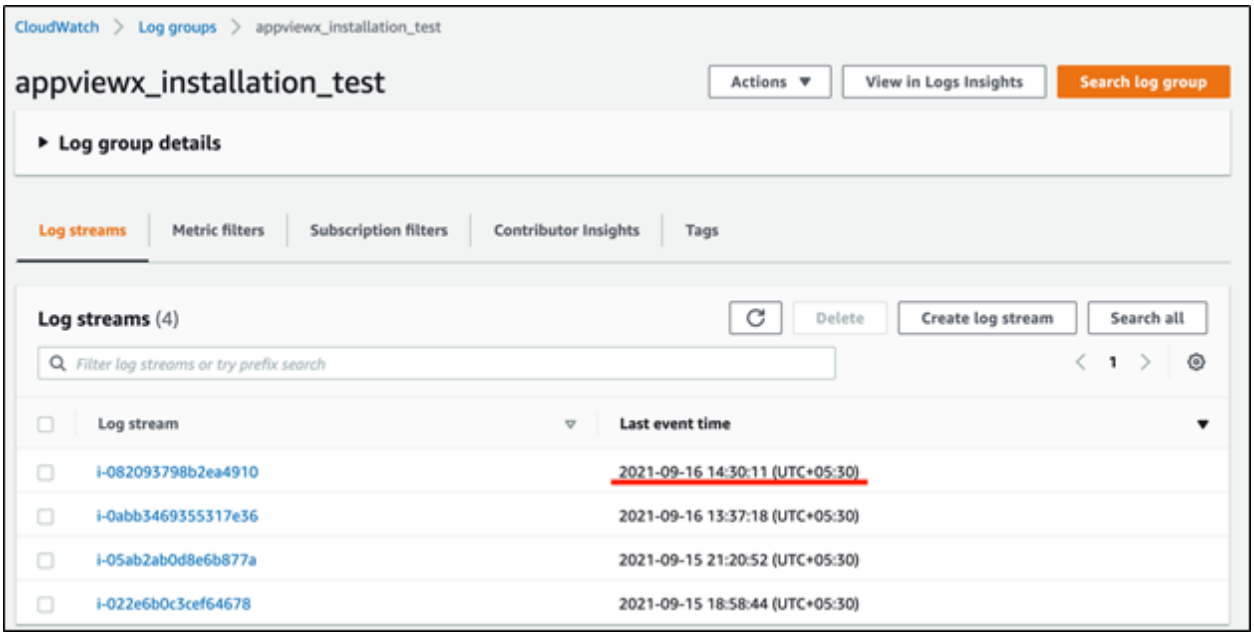
And you can see all the infra elements are created with a name starting with “Customer name“.



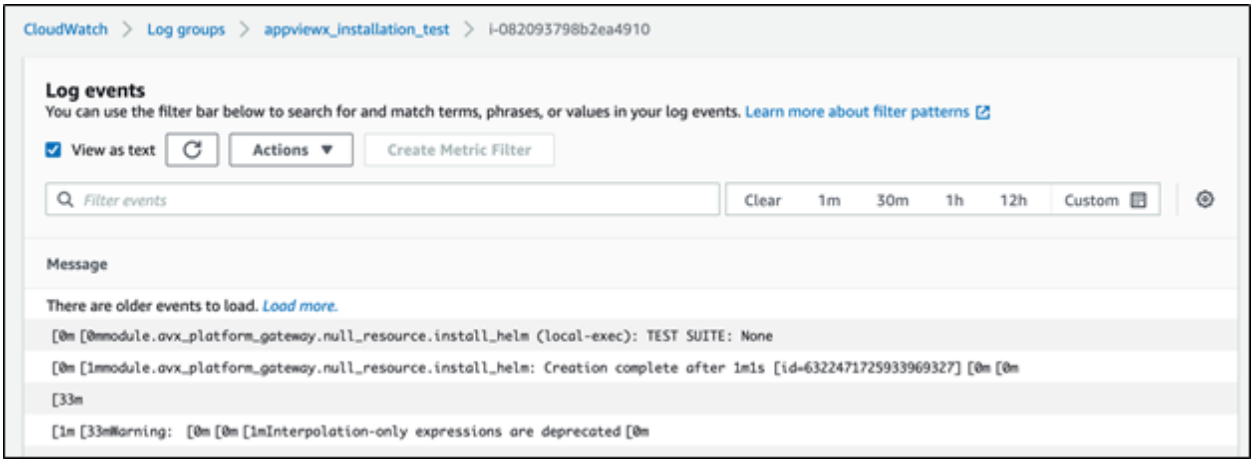
The product deployment logs in CloudWatch are seen as shown below:



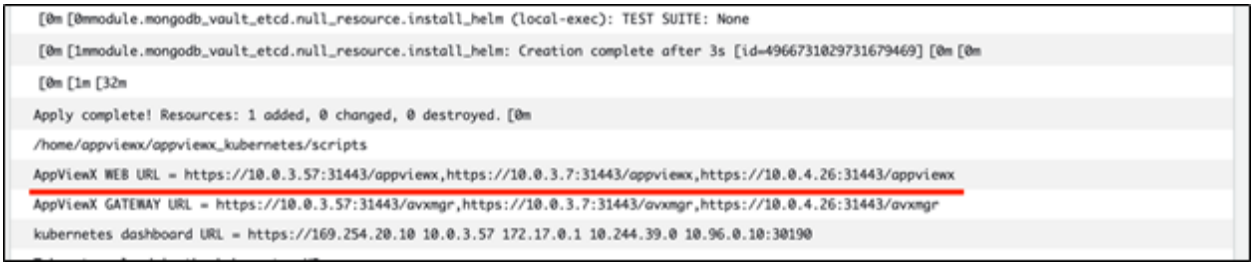
Select the latest timestamp log for watching the live installation progress.



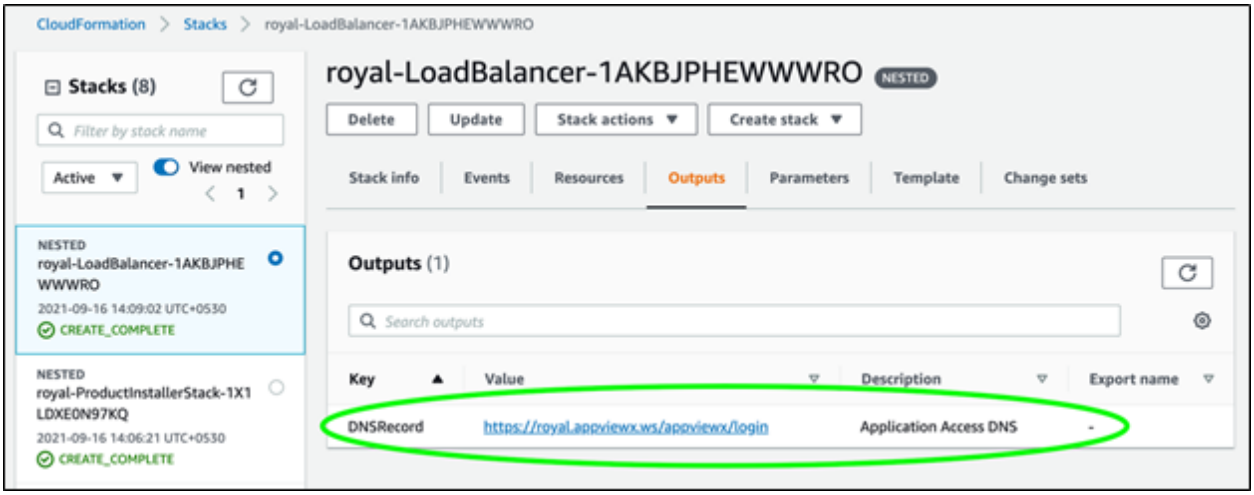
Select the **View as text** checkbox.



Once the installation is completed, you can see the web access URL which is used for internal routing instead of using the DNSRecord URL.



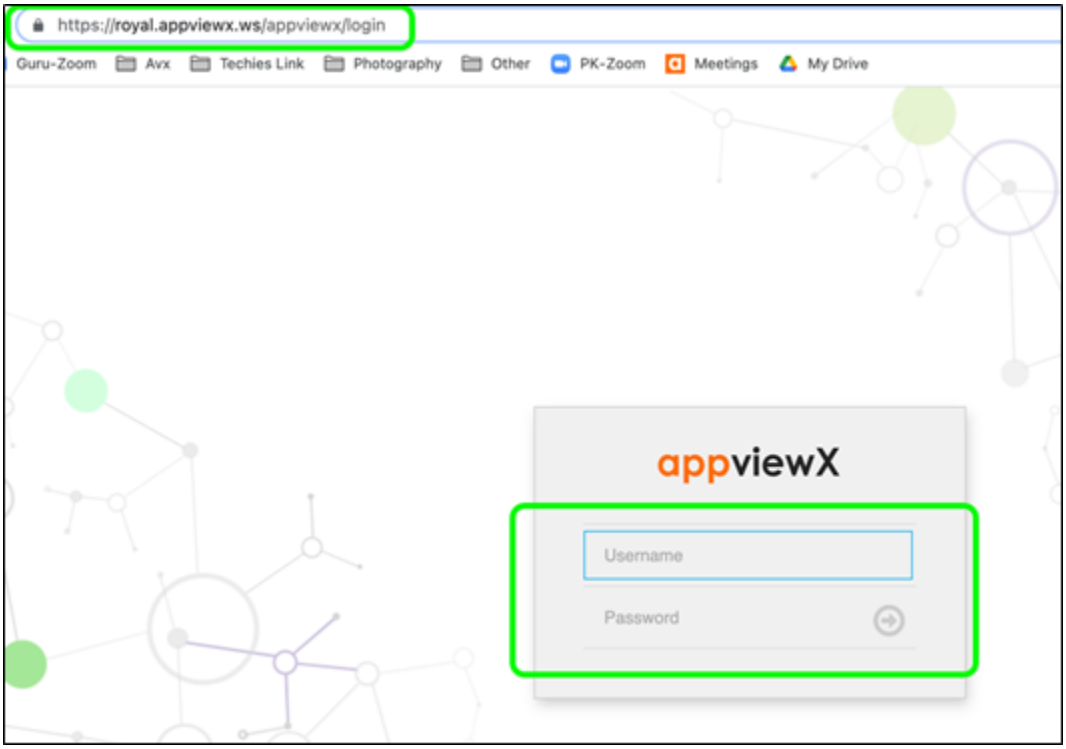
Now open the stack and click on the “Loadbalancer“ stack to get the final publicly accessible URL



Login with default credentials

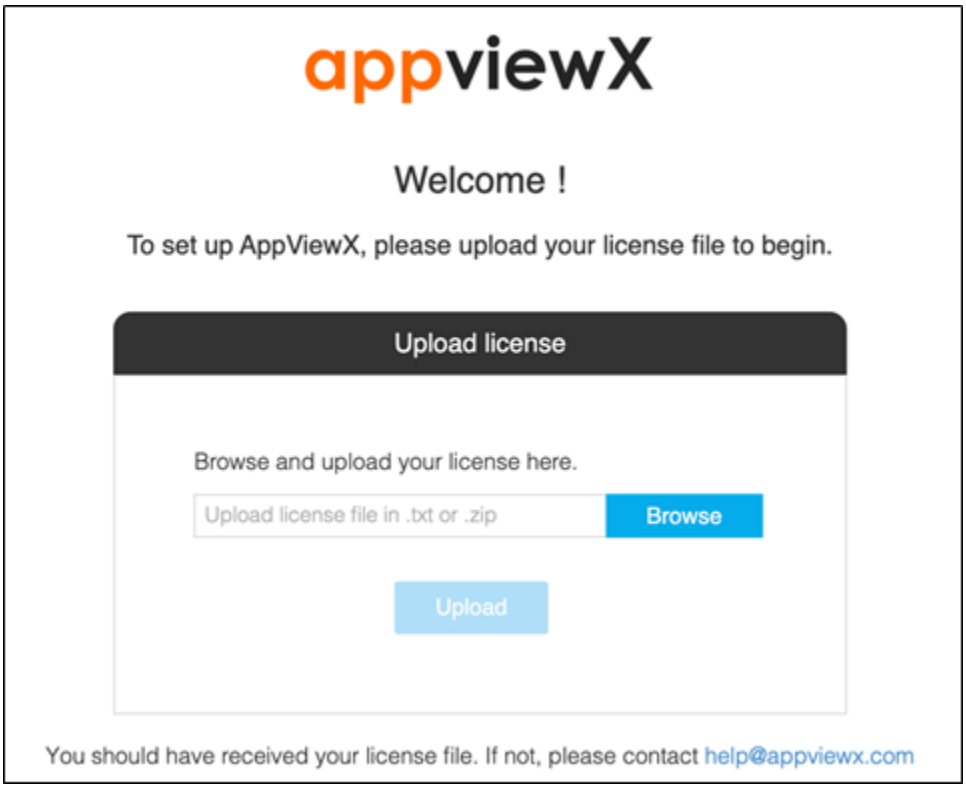
User name: admin

Password: AppViewX@123



Please create a license uploaded here to proceed with application access.

WIP: Auto upload license



Configuring MongoDB Backup

To back up the MongoDB data, you need to execute the cron job located in the **~/appviewx_kubernetes/yaml/appviewx_backup** folder (made available with the installation package).

To execute this cron job:

1. From the terminal, navigate to the MongoDB folder to be backed up.
2. Trigger the cron job using the following syntax:

```
kubectl create job -- from=cronjob/<name of cronjob> <name of job>
```

Example:

```
kubectl create job -- from=cronjob/mongo-backup -n avx-jobs mongo-backup manual-001
```

3. To check if the data has backed up successfully in the AWS CloudFormation interface, navigate to the S3 bucket location in which the folder will be backed up. You should be able to see the backed up data.

> **Note:** The duration for backing up the selected data depends on its volume. The time taken to restore 1GB data is approximately 3 minutes.

Setting up Backup and Restore

Restoring MongoDB Backup

To restore the MongoDB backup using a script:

1. Navigate to the node where AppViewX is installed.
2. Navigate to the **~/appviewx_kubernetes/yaml/mongo_restore** folder.
3. Run the following command:

```
sh run.sh '<appviewx_installed_path>' '<single/multinode_boolean_value>' '<file_in_s3_to_restore_mongodb_without_extention>'
```

Time taken to restore 1GB data is approximately 3 minutes.

Example:

```
sh run.sh /home/appviewx/appviewx true mongo_backup_Thu_Jul_1_04_01_03_UTC_2021
```

Restoring MongoDB

To restore MongoDB from one AppViewX to another AppViewX instance:

1. Install AppViewX on a different node.
2. Move the backup file you want to restore in the newly installed AppViewX node in the **{installation_path}/mongo-backup** folder.
3. Restore the backup taken using the script.
4. Navigate to the AppViewX installed node, and navigate to the following folder : **~/appviewx_kubernetes/yaml/mongo_restore**.

5. Run the `run.sh` script using the following command:

```
sh run.sh '<appviewx_installed_path>' '<single/multinode_boolean_value>' '<file_in_s3_to_restore_mongodb_without_extention>'
```

Example:

```
sh run.sh /home/appviewx/appviewx true mongo_backup_Thu_Jul_1_04_01_03_UTC_2021
```

6. Do gateway refresh using the API /avxmgr/refresh.
7. Go to **Certificate > Certificate Authority** and delete **Certificate Authority AppViewX Kafka CA**.
8. Install Kafka.
9. Delete the old cloud connectors.
10. Register the new cloud connectors and install the cloud connector and check the communication.

Modifying Scheduled Backup

To modify the cronexpression of backup job, use the following command in the AppView installed node:

```
kubectl patch cronjob mongo-backup -n avx-jobs -p '{"spec":{"schedule": "<cron_expression>"}}'
```

Example:

```
kubectl patch cronjob mongo-backup -n avx-jobs -p '{"spec":{"schedule": "0 */5 * * *"}}'
```

Chapter 3: Enabling KMS and DB

Enabling KMS and DB

By default, the SaaS deployments will be using AWS Key Management Service (KMS), instead of the Hashicorp Vault. The KMS configuration to be created per installation before proceeding with the installation. Each deployment should have a unique key created per tenant.

To enable the AWS KMS:

- Access the KMS using the credentials authorized for AppViewX.

The credentials are stored in the database in the **aws_access_info** collection, as shown below:

```
{
"usage" : "KMS",
 "accessKeyID" : "<AWS_ACCESS_KEY_ID>",
"encryptedSecretAccessKey" : "<AWS_ENCRYPTED_SECRET_KEY>", // secret key encrypted with the
random key method
"regionName" : "<AWS_REGION_NAME>",
"encryptionKey" : "<ACCESS_KEY_ENCRYPTION_KEY>" // the key output from the random key
method"usage" : "KMS",
"accessKeyID" : "<AWS_ACCESS_KEY_ID>",
"encryptedSecretAccessKey" : "<AWS_ENCRYPTED_SECRET_KEY>", // secret key encrypted with the
random key method
"regionName" : "<AWS_REGION_NAME>",
 "encryptionKey" : "<ACCESS_KEY_ENCRYPTION_KEY>" // the key output from the random key
method
}
```

- For data encryption and decryption, the AWS KMS uses a data key. For a second level of security, this data key is encrypted and decrypted using a master key. This master key is created first in the KMS at the time of the license upload. The platform then calls the framework API that creates the master key and generates the data key. The master key and the data key are stored in the database in the **kms_key_info** collection.

```
{
"encryptedDataKey" : "<ENCRYPTED_DATA_KEY>",
"keyId" : "<REFERENCE_TO_MASTER_KEY>",
"timestamp" : "<CREATED_TIMESTAMP>"
```

```
}
```

- When the framework receives an encryption request:
  1. The master key is used to decrypt the data key.
  2. The data key is used to encrypt the text using AppViewX’s in-house encryption method.
- When the framework receives a decryption request:
  1. The master key is used to decrypt the data key.
  2. The data key is used to decrypt the text using AppViewX’s in-house decryption method.

Chapter 4: Upgrading an Infrastructure Instance

Upgrading an Infrastructure Instance

Based on an updated requirement, to upgrade an infrastructure instance:

1. For the updated requirements, gather details for the upgraded parameters of the infrastructure.
2. In the deployed AWS CloudFormation template, select **Update.**
3. Update the template parameters as required.
4. Click **Save.**

   The AWS CloudFormation template triggers the upgrade for the infrastructure.

Chapter 5: Offboarding

Offboarding

To offboard a tenant:

1. Decommission the infrastructure instance set up for the tenant.
2. Archive the backup data stored in the standard AWS S3 bucket in the AWS Glacier.
3. Notify the tenant that this backed up data will be archived for the data retention period specified at the time of onboarding, after which the data will be deleted.

Chapter 6: Troubleshooting

Troubleshooting

To navigate and view the JAR files for the starter plugin:

1. Login to the k3d node.

   To login to the **k3d**, use the following command:

   ```
   docker exec -it k3d-cc-server-0 /bin/sh
   ```

2. Navigate to **/plugin-deps/mid-server-starter/lib/<version>/**.

   The cloud_connector_platform_dependencies.tar.gz is not downloaded (if the cloud connector platform plugin does not start up) even after the recommended threshold of 20 minutes has lapsed

   Possible causes:

   Kafka has not read the offset properly.

   Manual fix:

   a. Download the **cloud_connector_platform_dependencies.tar.gz** from Nexu s http://repo.appviewx.in and copy it inside the **avx_cloud_connector_starter** pod using following command:

      ```
      kubectl cp -n avx mid_server_platform_dependencies.tar.gz {mid-server-starter pod name}:/tmp/
      ```

   b. Execute the cloud-connector-starter pod and move it to the **/appviewx/dependencies/utils/** folder.

   c. Navigate to the location where the **cloud_connector_platform_dependencies.tar.gz** is located.

      To download the platform and install it in the Cloud Connector, execute the following command:

      ```
      ./platform_upgrade.sh /tmp/
      ```