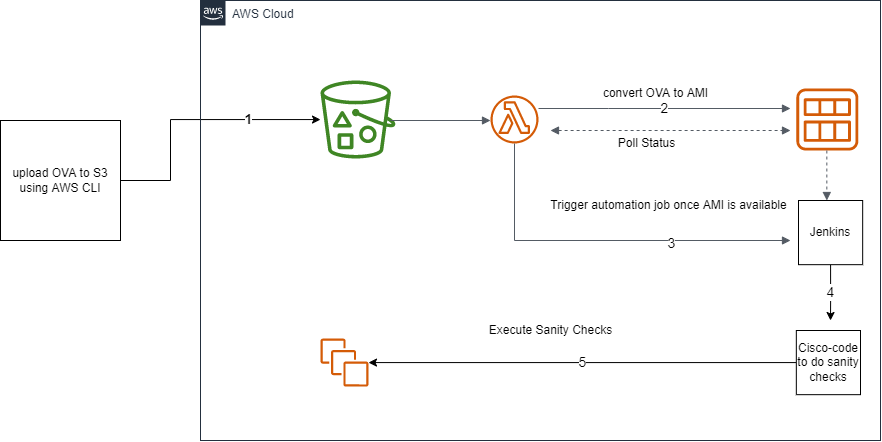
**Automate AWS Image Generation:**

**Overview:**

The task we had in hand was to automate the image generation process where when we upload an OVA file provided by client to the S3 bucket.

**Architecture Diagram:**



The task had 3 phases

**Phase:1**

Once an OVA is uploaded to S3 bucket automatic **OVA** to **AMI** conversion should happen.

**Phase:2**

Since the conversion takes time, we should be able to track the conversion status i.e., when the conversion is complete.

**Phase:3**

Once the conversion is complete, we should be able to get the AMI-ID of the converted image to call a Jenkins job to start sanity testing.

**AWS Services used for the conversion process:**

1. **S3** - For storage
2. **Lambda** – For conversion automation
3. **Step Functions** – Used as a workflow to check the status of conversion and bind the lambdas together
4. **IAM** - To give adequate role to the lambdas to execute the transformation

**Steps to achieve the functionality:**

**IAM POLICY:**

1. Created a role in IAM so that the lambda function which doing the conversion of OVA to AMI can execute the import\_image API call.

Graphical user interface, application

Description automatically generated

1. The import\_image\_role has the following permission list attached to the role in which vmimport-for-lambda is a custom policy which was written out of the box.

Graphical user interface, text, application

Description automatically generated

1. The policy details for vmimport-for-lambda are as follows

Text, letter

Description automatically generated

1. Next, we created a **lambda function** that should get triggered when an OVA is uploaded to the s3 bucket and assigned **import\_image\_role** to the same lambda-function.

**LAMBDAS USED:**

We have used three lambdas in order to execute the automation job –**OVAUpload-GenerateAMI** , **GenerateAMI-CheckStatus** and **AMIGenerated-StartSanityChecks**.

1. The **OVAUpload-GenerateAMI** is responsible for starting the OVA to AMI conversion and after starting the conversion it calls the step function named [**test-ami-generation**](javascript:void(0))
2. The **GenerateAMI-CheckStatus** lambda is responsible for getting the status response for OVA to AMI conversion
3. The **AMIGenerated-StartSanityChecks** lambda is responsible to run the Jenkins job from the lambda function

In **AMIGenerated-StartSanityChecks** lambda we have installed a 3rd party library for python to start the jenkins task. Following are the steps to install third party libraries into the lambda environment

**Install layer in lambda**

* + - * In a ubuntu VM in AWS:
        + mkdir -p layer/python/lib/python3.9/site-packages
        + pip3 install python-jenkins -t layer/python/lib/python3.9/site-packages/
        + cd layer/
        + zip -r mypackage.zip \*
        + sftp and extract the mypackage.zip file

**Deploy the Jenkins dependency**

* + - * In AWS lambda console click on layers

Graphical user interface, text, application

Description automatically generated

* + - * Click on create layer

Graphical user interface, application

Description automatically generated with medium confidence

* + - * Name your layer as "jenkins"(you can give any name as you want). Notice that you have an option to upload a zip file or upload a file from amazon s3. We will use upload a zip directly.
      * Choose compatible architecture as: x86\_64. Since we selected the same while creating our lambda function. And choose compatible runtime as python3.9, upload zip file, and click on create.
      * A lambda layer will be successfully created. Now we just need to attach this with our lambda function.
      * Navigate back to the lambda function< **AMIGenerated-StartSanityChecks** >. Scroll down to the bottom and click on add a layer (Under Layers section).
      * Click on the custom layer and select ‘jenkins’, select the version and click on add.

**STEP FUNCTIONS WORKFLOW USED**

Below is the screenshot of the step-function execution flow

Graphical user interface, text, application, chat or text message

Description automatically generated

**ROLE USED BY THE STEP FUNCTION: StepFunctions-test-ami-generation-role-4eb904ec**

Graphical user interface, text, application, email

Description automatically generated

1. The step function [**test-ami-generation**](javascript:void(0)) acts as a workflow and does three things –
   1. The step **ami-ova-conversion** checks the status of ova to ami conversion through a lambda function OVAUpload-GenerateAMI

Graphical user interface

Description automatically generated with medium confidence

The input payload for the step **ami-ova-conversion** we have taken as “**use state input as payload**”

Graphical user interface, text, application

Description automatically generated

For output we have kept the following configurationGraphical user interface, text, application, email

Description automatically generated

* 1. The step **check-conversion-status** has a rule configured that if the output of the previous step (step ami-ova-conversion) has status as “completed” then it goes to the next step (Invoke Jenkins Job) where the Jenkins job is called else it goes to the wait state(wait-for-5-minutes) where it waits for 5 minutes and then calls the ami-ova-conversion step again.

A picture containing diagram

Description automatically generated

Following are the rules written for the same check-conversion-status choice stateGraphical user interface, text, application, email

Description automatically generated

* 1. In Wait state we have configured the timer to wait for 300 seconds/5 minutes and the next state/step is configured as to call ami-ova-conversion stepGraphical user interface, text, application

     Description automatically generated
  2. In **Invoke Jenkins Job** step we call the lambda function [Run\_Jenkins\_Job](https://ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions/Run_Jenkins_Job) which calls the Jenkins job that validates the instance sanity created out of the generated AMI

Graphical user interface

Description automatically generated with low confidence