**Part 1 Create a local demo folder.**

**Step 1**

Download base folder “AWS\_lambda” from FaaSr/ base\_docker\_images. Build the demo folder in local.

Link: <https://github.com/renatof/FaaSr/tree/main/base_docker_images/AWS_lambda>

**Step 2**

The structure of AWS lambda demo folder in local should look like:

aws\_lambda\_folder\_name/

--Dockerfile

--entry.sh

--lambda\_function.py

--app/

-- faasr.R

-- exec.R

-- simple\_sequence1.R

-- FaaSr.schema.json

“FaaSr.schema.json” can download from <https://github.com/renatof/FaaSr/blob/main/schema/FaaSr.schema.json>

“faasr.R” can download from <https://github.com/renatof/FaaSr/blob/main/R/faasr.R>

“simple\_sequence1.R” can download from <https://github.com/renatof/FaaSr/blob/main/examples/simple_sequence1.R>

**Part 2 Build docker image and upload to AWS ECR.**

For AWS lambda function example, we need to upload image to Amazon Elastic Container Registry, we can use AWS CLI.

**Step 1** Create Access Key, Install AWS CLI and Configuring the AWS CLI

1. Create AWS Access Key ID and AWS Secret Access Key

Use AWS console to create Key. By referencing this link: <https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#Using_CreateAccessKey>

In “Access key best practices & alternatives” part, we can choose “Command Line Interface (CLI)”

Note: When "Access key ID" and "Secret access key" are created, download the key or copy the keys directly from the console. This is the only time we see the Secret Access Key.

1. Install AWS CLI and Configuring the AWS CLI

By referencing the install/update part of this link: <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

After installing AWS CLI, we will configure it with AWS access key ID, secret access key, and default region. In terminal, run “**aws configure**” to quickly set.

By Referencing “Set and view configuration settings using commands” part of this link: <https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html>

**Step 2** Create an ECR Repository

Reference link: <https://docs.aws.amazon.com/AmazonECR/latest/userguide/getting-started-cli.html#cli-create-image>

1. Create ECR repository: **aws ecr create-repository --repository-name your\_repo\_name**

(it will return the **repositoryUri**)

2. Connect ECR and Docker: **aws ecr get-login-password --region region | docker login --username AWS --password-stdin aws\_account\_id.dkr.ecr.region.amazonaws.com**

3. Build docker image: build image with the demo folder. (When building image, we can build this image with a name: **your\_aws\_account\_id.dkr.ecr.your\_ecr\_region.amazonaws.com/ecr\_repo\_name:latest**

In next step, we can push image with this name to ECR.)

4. Push an image to Amazon ECR:

**docker push your\_aws\_account\_id.dkr.ecr.your\_ecr\_region.amazonaws.com/ecr\_repo\_name:latest**

**Part 3 Create AWS Lambda function and setting configurations.**

Below are two methods to create lambda function: AWS Lambda console or AWS CLI command line.

**Method 1 Using AWS Lambda console.**

**Step1** create Lambda functions.

Reference link: <https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-images.html>

In AWS Lambda console, select a region(same region as the ECR repo, such like: us-east-1) to create Lambda function.

Choose **Create function,**

Choose the **Container image** option,

Under **Basic information**, enter function name, and select image For Container image URI, we can enter the Amazon ECR image URI or browse for the Amazon ECR image.

for our demo(using simple\_sequence1.json), we need to create four functions, and function name will be **F1, F2, F3, F4**. These four Lambda functions will use the same one ECR image URI.

Select **Create Function**

**Step 2** setting configurations

For these four lambda functions F1, F2, F3, F4. We will set the Memory size and Timeout.

For each lambda function, under **Configuration**, select **General configuration**, select **edit**, set **Memory** with 256MB, **Timeout** with 1min.

**Method 2 Using AWS CLI command line.**

Reference link:

<https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-awscli.html#with-userapp-walkthrough-custom-events-delete-function>

<https://awscli.amazonaws.com/v2/documentation/api/latest/reference/lambda/create-function.html>

Open a terminal window in local.

**Step 1** create a role for lambda function.

Enter Command:

aws iam create-role --role-name faasr-lambda-demo-role --assume-role-policy-document '{"Version": "2012-10-17","Statement": [{ "Effect": "Allow", "Principal": {"Service": "lambda.amazonaws.com"}, "Action": "sts:AssumeRole"}]}'

**step 2** attach policy to role

Enter Command:

aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole --role-name faasr-lambda-demo-role

**step 3** creates lambda function (F1, F2, F3,and F4)

Enter Command:

aws lambda create-function --region your\_function\_region --function-name F1 \

--package-type Image \

--code ImageUri=your\_aws\_account\_id.dkr.ecr.ecr\_region.amazonaws.com/repo\_name:tag \

--role arn:aws:iam::your\_aws\_account\_id:role/faasr-lambda-demo-role \

--timeout 60 \

--memory-size 256

**Note**: when create F2, F3, F4, just repeat step 3 and replace “--function-name” with F2, F3 or F4 to create function.

**Part 4 Test AWS Lambda Functions.**

After creating lambda functions. We will use AWS CLI to invoke lambda function F1.

**Note**, we should first upload the "input1.csv" and "input2.csv" to the S3 bucket folder.

**Step 1** save the payload.json file in local folder (for example lambda\_demo), we can use simple\_sequence1.json as our payload.json file, and modify relevant values.

Link: <https://github.com/renatof/FaaSr/blob/main/examples/simple_sequence1.json>

**Step 2** In lambda\_demo folder, open a terminal window, enter command line:

aws lambda invoke --function-name F1 --cli-binary-format raw-in-base64-out --invocation-type RequestResponse --payload file://payload\_valid.json outputfile.txt

In lambda function console, under **Monitor,** select **View CloudWatch Logs** to see lambda execution logs.