Lambda Library Compatibility Issue & Solution

Issue Summary

The Lambda function deployment is successful, but execution fails with the following error:

```
Runtime.ImportModuleError: Unable to import module 'lambda_handler':
/lib64/libc.so.6: version `GLIBC_2.28' not found
(required by /var/task/cryptography/hazmat/bindings/_rust.abi3.so)
```

Root Cause

The deployment package was built on **Ubuntu** which uses **GLIBC 2.28+**, but AWS Lambda Python 3.11 runtime uses **Amazon Linux 2** which has **GLIBC 2.26**. The `cryptography` library (a dependency of `simple-salesforce` and other packages) contains compiled binary extensions that are incompatible.

Solution Options

Option 1: Build with Docker (Recommended)

Use Docker with the official AWS Lambda Python base image to build the package:

```
cd /home/ubuntu/collaboratemd-salesforce-middleware

# Build using Docker
docker run --rm \
  -v "$PWD":/var/task \
  -w /var/task \
  public.ecr.aws/lambda/python:3.11 \
  bash -c "pip install -r requirements.txt -t lambda_package_linux/ && \
           cp lambda_handler.py lambda_package_linux/ && \
           cp -r src lambda_package_linux/ && \
           cd lambda_package_linux && \
           zip -r ../lambda_deployment.zip . -q"

# Deploy the new package
./scripts/deploy_lambda.sh
```

Option 2: Use EC2 with Amazon Linux 2

1. Launch an EC2 instance with Amazon Linux 2
2. Install Python 3.11
3. Clone the repository
4. Build the package:

```
python3.11 -m pip install -r requirements.txt -t lambda_package/
cp lambda_handler.py lambda_package/
cp -r src lambda_package/
cd lambda_package && zip -r ../lambda_deployment.zip . -q
```

1. Download and deploy the package

Option 3: Use AWS Lambda Layers

Split the dependencies into layers:
1. Create a Lambda Layer with the dependencies
2. Deploy the application code separately
3. Attach the layer to the function

Option 4: Use AWS SAM or CDK

Use AWS SAM or CDK which automatically handle cross-platform builds.

Quick Fix Script

A script has been created at `scripts/build_lambda_package_docker.sh` that uses Docker to build the package correctly.

Prerequisites:
- Docker must be installed and running
- Docker daemon must have access to pull images

Usage:

```
cd /home/ubuntu/collaboratemd-salesforce-middleware
./scripts/build_lambda_package_docker.sh
./scripts/deploy_lambda.sh
```

Verification

After rebuilding and redeploying:

```
# Test the function
aws lambda invoke \
  --function-name collaboratemd-salesforce-sync \
  --region us-east-1 \
  --cli-binary-format raw-in-base64-out \
  --payload '{"full_sync":false}' \
  /tmp/lambda-response.json

# Check the response
cat /tmp/lambda-response.json | jq .

# Check CloudWatch logs
aws logs tail /aws/lambda/collaboratemd-salesforce-sync --follow --region us-east-1
```

Alternative: Use Pre-built Wheels

Modify `requirements.txt` to use pre-built wheels compatible with manylinux2014:

```
--platform manylinux2014_x86_64
--only-binary=:all:
requests
simple-salesforce
boto3
python-dotenv
```

Current Status

✅ Lambda function created successfully
✅ IAM roles and permissions configured
✅ DynamoDB table created
✅ Environment variables set correctly
❌ Function execution fails due to library compatibility

Next Steps

1. Install Docker on the build machine OR use EC2 with Amazon Linux 2
2. Rebuild the deployment package using Option 1 or 2
3. Redeploy using `./scripts/deploy_lambda.sh`
4. Test the function invocation
5. Monitor CloudWatch logs for successful execution

References

- AWS Lambda Runtimes (https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtimes.html)
- Building Lambda deployment packages (https://docs.aws.amazon.com/lambda/latest/dg/python-package.html)
- Using container images with Lambda (https://docs.aws.amazon.com/lambda/latest/dg/python-image.html)