

Tutorial: Manually deploy and troubleshoot WebAuthnKit backend at AWS

- Introduction
- Prerequisites
- Clone or download the GitHub project
- Create an S3 bucket in AWS
 - Create an S3 bucket manually with the AWS console
 - Create an S3 bucket with a script using the AWS tools
- Build the SAM package
- Deploy the SAM package at AWS
- Troubleshooting
 - Update the environment variables for AWS Lambda
 - Verifying the RDS database tables
 - Re-build and re-deploy JavaWebAuthnLib.jar
 - Install Java JDK and Apache Maven
 - Re-build the JavaWebAuthnLib.jar
 - Re-deploy JavaWebAuthnLib.jar
 - Manual deployment of JavaWebAuthnLib.jar at the AWS console
 - AWS CLI for scripted deployment of JavaWebAuthnLib.jar
 - Attach the AWSLambdaRole policy
 - Enable the CORS policy at API Gateway
 - Delete stack, S3 bucket and Amplify apps after roll back
 - Delete the AWS CloudFormation stack
 - Delete the AWS S3 bucket
 - Delete the AWS Amplify apps

Introduction

The complete and automated deployment of the WebAuthn Starter Kit backend at AWS is described in the tutorial Automated WebAuthnKit deployment at AWS, which describes the pre-requisites, installation packages, source code and scripts needed for the deployment of the WebAuthn Starter Kit at AWS.

This guide, however, describes how to **manually** deploy the WebAuthn Starter Kit backend at AWS. Furthermore, it contains a troubleshooting section for identifying and fixing issues that may occur at the AWS backend.

Prerequisites

See the prerequisites section in the tutorial Automated WebAuthnKit deployment at AWS.

Clone or download the GitHub project

See the section on how to Clone or download the GitHub project in the tutorial Automated WebAuthnKit deployment at AWS.

Create an S3 bucket in AWS

Before the WebAuthn Starter Kit can be deployed a designated S3 bucket must be created in the AWS account. Also, the S3 bucket should be created in the same AWS Region in which the WebAuthn Starter Kit will be deployed. Furthermore, make sure that AWS Cognito is available for the S3 bucket's region. For example, Cognito is available for the region 'eu-west-2', but not for 'eu-north-1'. The regions where Cognito is currently available are displayed with white text in the drop-down menu under the region button.

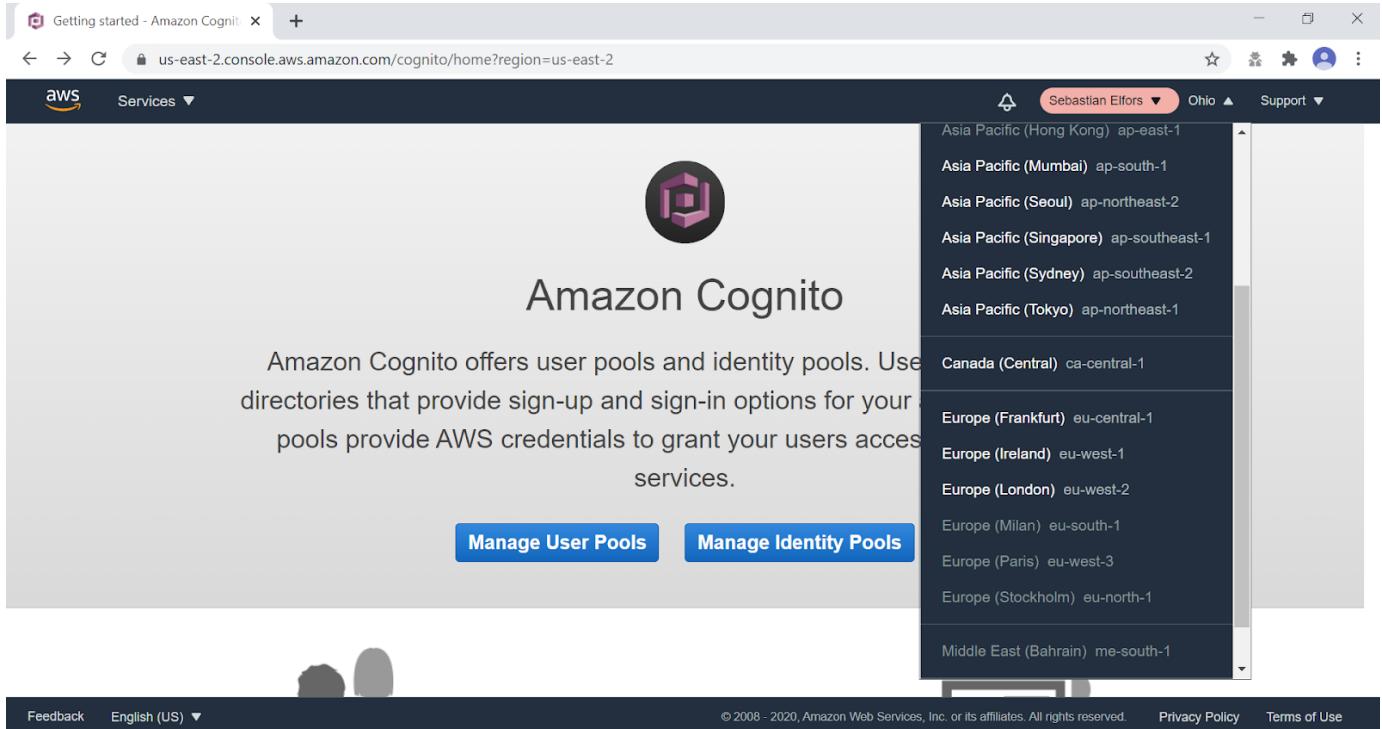


Figure 1 - Check the AWS Cognito regions

There are two ways to create an S3 bucket at AWS:

- Create an S3 bucket manually with the AWS console, or
- Create an S3 bucket with a script using the AWS tools at the workstation

Create an S3 bucket manually with the AWS console

The S3 buckets can be found in the AWS console by selecting Services followed by S3. Create an S3 bucket by using the AWS console according to the instructions in this AWS guide. Use the default settings when creating the S3 bucket.

A screenshot of the AWS S3 Management Console. The left sidebar shows navigation options like 'Amazon S3', 'Buckets', 'Batch operations', 'Access analyzer for S3', 'Block public access (account settings)', and 'Feature spotlight'. The main content area has a banner about AWS Storage Gateway. Below it, a search bar says 'Search for buckets' and a dropdown says 'All access types'. A button '+ Create bucket' is available. A table lists 46 buckets across 6 regions. The columns include 'Bucket name', 'Access', 'Region', and 'Date created'. Some bucket names are 'amplify-amplifyjsapp-dev-124928-deployment', 'amplify-awsamplifyfido5-dev-175550-deployment', etc. The table shows various locations like US West (Oregon), EU (Ireland), US East (N. Virginia), and EU (London). At the bottom, there are links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'.

Figure 2 - Create an S3 bucket in the AWS Console

Create an S3 bucket with a script using the AWS tools

Another option is to create the S3 bucket by using a script with the AWS tools at the workstation. A PowerShell prompt or Terminal could be used for this purpose.

Syntax of the AWS command for creating an S3 bucket in AWS:

```
aws s3api create-bucket --bucket <bucket-name> --region <region-value>
--create-bucket-configuration LocationConstraint=<region-value>
```

Build the SAM package

At your workstation, use a Terminal or Command Prompt to run the WebAuthnKit SAM build and package script from the same directory where the SAM template template.yaml is located. By default, the path is ~/WebAuthnKit/backend. This build is automated with the SAM template.

The SAM build and package command will process the SAM template, application code, and build dependencies. Built artifacts will be written to .aws-sam/build folder in the local project folder and uploaded to the designated S3 bucket.

Change the --profile to match what you have in ~<user-home>/.aws/credentials for running the AWS CLI and SAM commands. Then execute the command in a MacOS or Linux Terminal or Windows Command Prompt:

```
sam build --use-container --skip-pull-image && sam package --s3-bucket
<s3BucketName> --profile <wsCliProfile>
```

When using Windows PowerShell, the separator “**&&**” must be changed to “**,**”

```
sam build --use-container --skip-pull-image ; sam package --s3-bucket
<s3BucketName> --profile <awsCliProfile>
```

As part of this SAM command, the built artifacts will be uploaded to the S3 bucket. The upload to the S3 bucket takes a few minutes. It is worthwhile to check in the AWS S3 bucket that the packages have been properly uploaded.

The screenshot shows the AWS S3 Management Console interface. At the top, there's a navigation bar with the AWS logo, services dropdown, user info (Sebastian Elfors), and global support links. Below that is a breadcrumb trail: Amazon S3 > seb-webauthnstarterkit-eu-west-2. The main area is titled 'seb-webauthnstarterkit-eu-west-2' and has tabs for Overview, Properties, Permissions, Management, and Access points. The Overview tab is selected. A search bar at the top of the list table says 'Type a prefix and press Enter to search. Press ESC to clear.' Below the search bar are buttons for Upload, Create folder, Download, and Actions. The table lists nine items, each with a checkbox, name, last modified date, size, and storage class. The items include various file names like '25c776f164c1359f58e6a0a36ca0e04a' and 'a7e52d3b22ab8ad4b4b6e1faaa15201b.template'. The table header includes 'Name', 'Last modified', 'Size', and 'Storage class'. A note at the bottom right says 'Viewing 1 to 9'.

Name	Last modified	Size	Storage class
25c776f164c1359f58e6a0a36ca0e04a	Sep 17, 2020 2:48:30 PM GMT+0200	696.6 KB	Standard
31ac18e5ba01f6741749b010284ae6c73	Sep 17, 2020 2:48:28 PM GMT+0200	4.5 KB	Standard
992f8830273aecd3404659e2fe0f25e00	Sep 17, 2020 2:48:29 PM GMT+0200	27.9 KB	Standard
a7e52d3b22ab8ad4b4b6e1faaa15201b.template	Sep 17, 2020 3:01:36 PM GMT+0200	25.0 KB	Standard
c416d3db2cf4521ff47ab481c4e034c	Sep 17, 2020 2:48:28 PM GMT+0200	1.3 KB	Standard
d41d8cd98f00b204e9800998ecf8427e	Sep 17, 2020 2:48:28 PM GMT+0200	0 B	Standard
e79ac7980f67616318f7f81a6096abdd	Sep 17, 2020 2:48:31 PM GMT+0200	629.0 B	Standard

Figure 3 - Checking the S3 bucket

Next, the WebAuthnKit package will be deployed in the AWS environment using the SAM deploy command below.

Deploy the SAM package at AWS

The SAM deploy script below will take the SAM package and artifacts (already uploaded to S3 in the previous section) and deploy them to the AWS environment via CloudFormation. This deployment is automated with the SAM template.

Note: All variables marked with "\$" should be replaced with the values to match your deployment.

```
sam deploy --s3-bucket <s3BucketName> --stack-name <cfStackName> --
profile <awsCliProfile> --region <awsRegion> --capabilities
<capabilityIam> --parameter-overrides UserPoolName=<userPoolName>
DatabaseName=<databaseName> MasterUserName=<databaseMasterUsername>
MasterUserPassword=<databaseMasterPassword>
DefineAuthChallengeFuncName=<defineAuthChallengeFuncName>
CreateAuthChallengeFuncName=<createAuthChallengeFuncName>
VerifyAuthChallengeFuncName=<verifyAuthChallengeFuncName>
WebAuthnKitAPIFuncName=<webAuthnKitApiFuncName>
PreSignUpFuncName=<preSignUpFuncName>
JavaWebAuthnFuncName=<javaWebAuthnLibFuncName>
WebAuthnKitAPIName=<webAuthnKitApiName>
CreateDBSchemaFuncName=<createDatabaseSchemaFuncName>
CreateDBSchemaCallerFuncName=<createDatabaseSchemaCallerFuncName>
AmplifyHostingAppName=<amplifyHostingAppName>
```

The events of the deployment should be displayed through the command line interface.

The SAM deployment at AWS takes approximately 5 minutes to create all resources and 5 minutes to delete all resources.

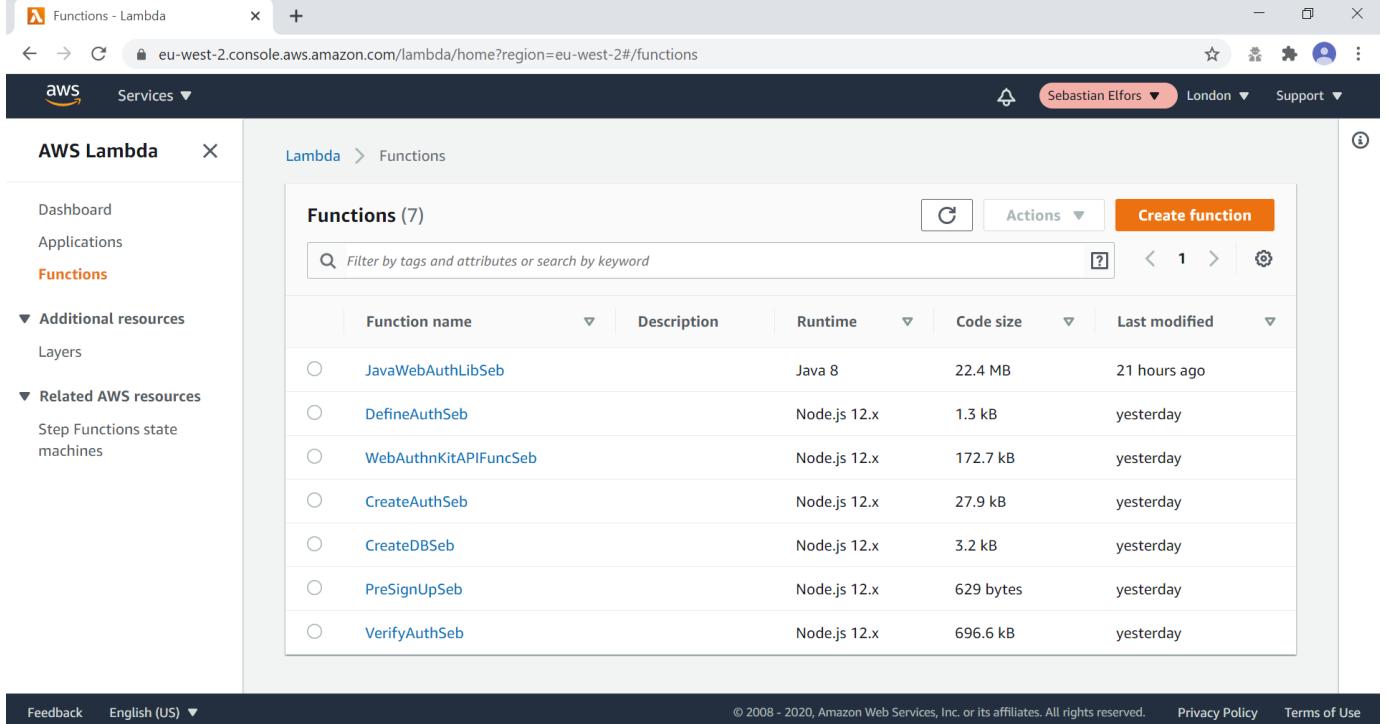
Troubleshooting

If the deployment at the AWS backend is not properly configured, it can be verified and debugged according the guidelines in this troubleshooting section.

Update the environment variables for AWS Lambda

Note: Before checking the environment environments according to this section, make sure to deploy the React component at AWS Amplify before.

Login to the AWS console, and select the Service called Lambda.



The screenshot shows the AWS Lambda service interface. On the left, there's a navigation sidebar with links for Dashboard, Applications, Functions (which is selected and highlighted in red), Additional resources (Layers), and Related AWS resources (Step Functions state machines). The main content area is titled "Functions (7)" and contains a table listing seven Lambda functions. The columns in the table are Function name, Description, Runtime, Code size, and Last modified. The functions listed are JavaWebAuthLibSeb, DefineAuthSeb, WebAuthnKitAPIFuncSeb, CreateAuthSeb, CreateDBSeb, PreSignUpSeb, and VerifyAuthSeb. Each function entry includes a small circular icon to its left and a link to its details page.

	Function name	Description	Runtime	Code size	Last modified
○	JavaWebAuthLibSeb		Java 8	22.4 MB	21 hours ago
○	DefineAuthSeb		Node.js 12.x	1.3 kB	yesterday
○	WebAuthnKitAPIFuncSeb		Node.js 12.x	172.7 kB	yesterday
○	CreateAuthSeb		Node.js 12.x	27.9 kB	yesterday
○	CreateDBSeb		Node.js 12.x	3.2 kB	yesterday
○	PreSignUpSeb		Node.js 12.x	629 bytes	yesterday
○	VerifyAuthSeb		Node.js 12.x	696.6 kB	yesterday

Figure 4 - Listing the Lambda Functions

In the list of Function names, select the Function called “JavaWebAuthLib<suffix>”. In this example, the Function is called “JavaWebAuthLibSeb”.

The deployment package of your Lambda function "JavaWebAuthLibS3" is too large to enable inline code editing. However, you can still invoke your function.

Key	Value
DBAuroraClusterArn	arn:aws:rds:eu-west-2:764504484168:cluster:fido2dbseb
DBSecretsStoreArn	arn:aws:secretsmanager:eu-west-2:764504484168:secret:fido2dbseb-5leyww
DatabaseName	fido2dbseb

Figure 5 - Checking the Lambda Function's environment variables

Scroll down to the environment variables and press the Edit button.

Key	Value	Action
DBAuroraClusterArn	arn:aws:rds:eu-west-2:764504484168:c	Remove
DBSecretsStoreArn	arn:aws:secretsmanager:eu-west-2:764!	Remove
DatabaseName	fido2dbseb	Remove
YUBICO_WEBAUTHN_ALLOWED_ORIGI	https://dev.d311a28dcPWM31.amplify	Remove
YUBICO_WEBAUTHN_RP_ID	dev.d311a28dcPWM31.amplifyapp.com	Remove
YUBICO_WEBAUTHN_RP_NAME	YubicoWebAuthnKit	Remove

Figure 6 - Editing the Lambda Function's environment variables

Press the button "Add environment variable" in order to add three environment variables that are declared in the JavaWebAuthnLib Config.java file:

- **YUBICO_WEBAUTHN_ALLOWED_ORIGINS:** This environment variable should be set to the complete URL that is configured for the AWS Amplify React front-end. For example: <https://dev.d311a28dcPWM31.amplifyapp.com>.
- **YUBICO_WEBAUTHN_RP_ID:** This environment variable should be set to the host name that is configured for the AWS Amplify React front-end. For example: dev.d311a28dcPWM31.amplifyapp.com.

- **YUBICO_WEBAUTHN_RP_NAME:** This environment variable could be set to any description. Example: YubicoWebAuthnKit.

The screenshot shows the AWS Lambda console interface for a function named "JavaWebAuthLibSeb". The "Environment variables" section is open, displaying six entries:

Key	Value
DBAuroraClusterArn	arn:aws:rds:eu-west-2:764504484168:cluster:fido2dbseb
DBSecretsStoreArn	arn:aws:secretsmanager:eu-west-2:764504484168:secret:fido2dbseb-5leyww
DatabaseName	fido2dbseb
YUBICO_WEBAUTHN_ALLOWED_ORIGINS	https://dev.d311a28dcPWM31.amplifyapp.com
YUBICO_WEBAUTHN_RP_ID	dev.d311a28dcPWM31.amplifyapp.com
YUBICO_WEBAUTHN_RP_NAME	YubicoWebAuthnKit

At the bottom of the page, there are "Tags (4)" and a "Save" button.

Figure 7 - Saving the Lambda Function's environment variables

Press the Save button to save the changes.

Verifying the RDS database tables

In order to verify that the Aurora RDS database tables have been properly created by the SAM deploy script, take the following actions.

Login to the AWS console, and select the Service called Amazon RDS.

Connect to the RDS database according to the parameters that were set in the SAM deploy script.

Hint: The Secrets manager ARN value can be found in the SAM deploy script output.

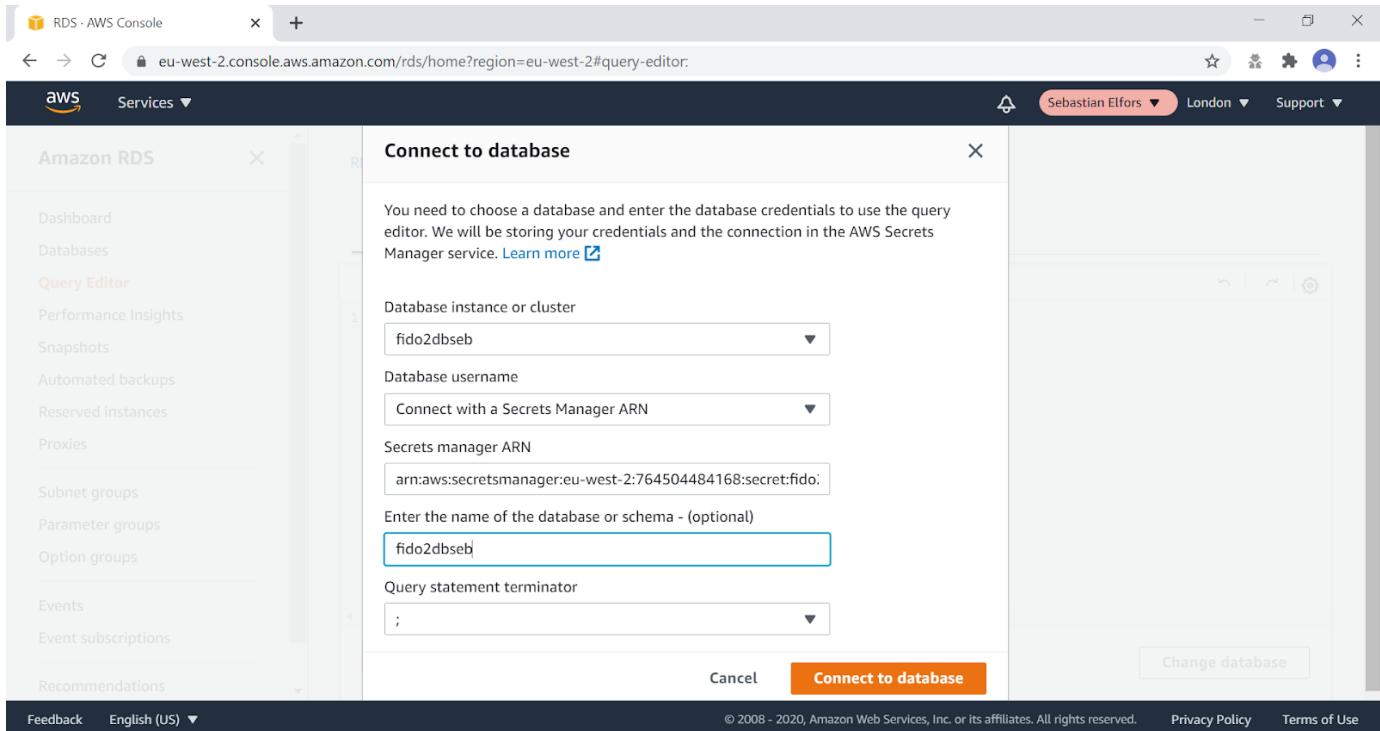


Figure 8 - Connecting to the RDS database

When you are logged in to the RDS database, select Query Editor in the left-hand pane.

In the Query Editor window, enter the SQL statement:

```
show tables;
```

Check the returned rows in the output:

The screenshot shows the 'Amazon RDS' service in the AWS console with 'Query Editor' selected. The main area displays the results of the 'show tables;' query. The 'Output' tab is active, and the results are shown under 'Result set 1 (5)'. The results are listed as follows:

Tables_in_fido2dbseb
assertionRequests
credential
credentialRegistrations
registrationRequests
user

At the top right of the results area, there is an 'Export to csv' button. The bottom of the screen shows the standard AWS footer with links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'.

Figure 9 - Checking the database tables

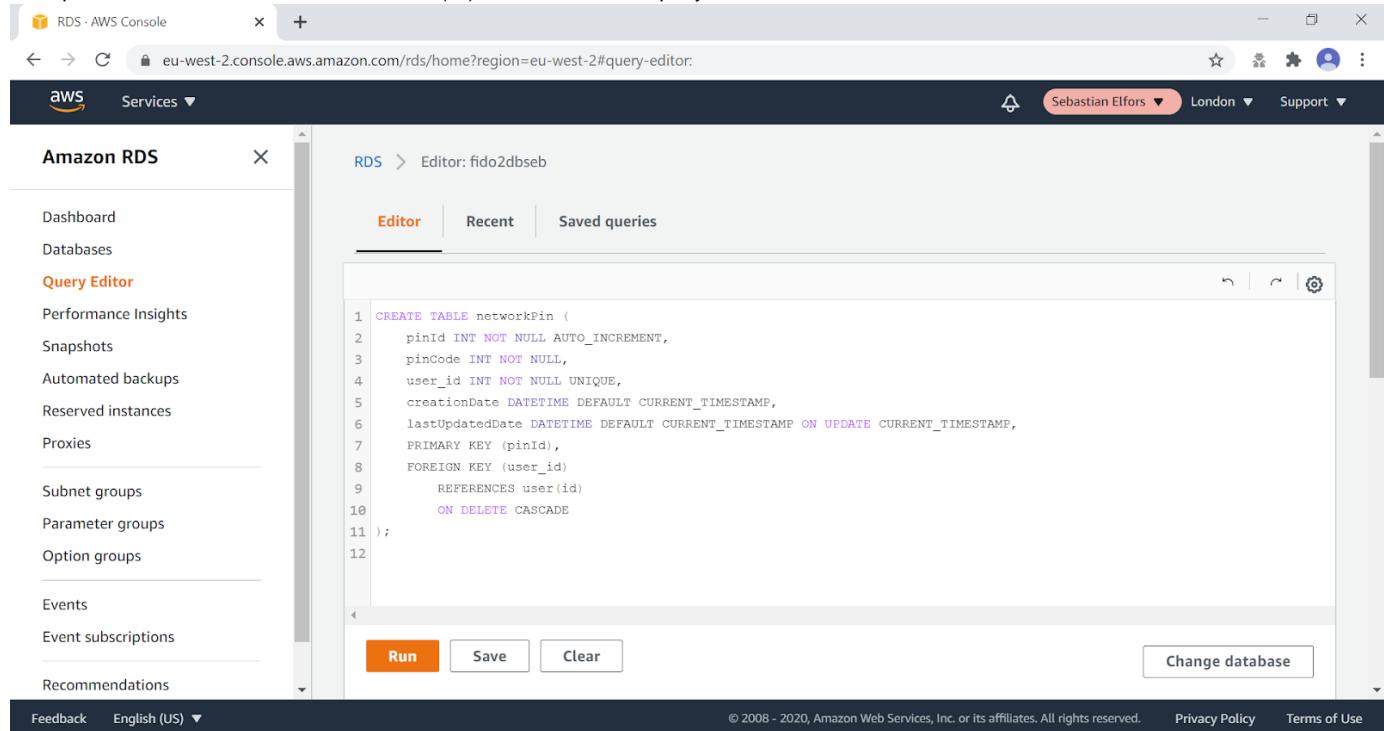
Then visit the WebAuthnKit GitHub repo and view the file database.md.

Compare the tables in the AWS RDS database with the tables in the GitHub file database.md. If there are any tables missing in the AWS RDS database, they need to be manually added.

In the screenshot example above, the database table “networkPin” is missing in the AWS RDS database.

In order to add the database table “networkPin”, copy the entire SQL transaction “CREATE TABLE networkPin (...)” from the GitHub file database.md.

Then paste the “CREATE TABLE networkPin (...)” in the AWS RDS query editor.



The screenshot shows the AWS RDS Query Editor interface. On the left, a sidebar menu lists various RDS features like Dashboard, Databases, and Query Editor. The main area is titled "RDS > Editor: fido2dbseb". It contains a code editor with the following SQL script:

```
1 CREATE TABLE networkPin (
2     pinId INT NOT NULL AUTO_INCREMENT,
3     pinCode INT NOT NULL,
4     user_id INT NOT NULL UNIQUE,
5     creationDate DATETIME DEFAULT CURRENT_TIMESTAMP,
6     lastUpdatedDate DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
7     PRIMARY KEY (pinId),
8     FOREIGN KEY (user_id)
9         REFERENCES userid
10        ON DELETE CASCADE
11 );
12
```

Below the code editor are three buttons: Run, Save, and Clear. To the right of the code editor is a "Change database" button. At the bottom of the screen, there are links for Feedback, English (US), Copyright notice (© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.), Privacy Policy, and Terms of Use.

Figure 10 - Creating a database table

Press the “Run” button to execute the SQL transaction.

Then execute the SQL statement “show tables;” again in the Query Editor and verify in the output section that all database tables have been created.

The screenshot shows the AWS RDS Query Editor interface. The left sidebar lists various RDS services: Dashboard, Databases, Query Editor (which is selected), Performance Insights, Snapshots, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Events, Event subscriptions, and Recommendations. The main area is titled "Result set 1 (6)" under the "Output" tab. It displays a list of table names from the "Tables_in_fido2dbseb" schema: assertionRequests, credential, credentialRegistrations, networkPin, registrationRequests, and user. There is also a "Rows returned (6)" section above the table list. A search bar and a "Export to csv" button are visible at the top of the results area. The bottom of the screen shows standard AWS footer links: Feedback, English (US) ▾, © 2008–2020, Amazon Web Services, Inc. or its affiliates. All rights reserved., Privacy Policy, and Terms of Use.

Figure 11 - Verifying the database tables

Re-build and re-deploy JavaWebAuthnLib.jar

If the JavaWebAuthnLib.jar file was not properly built and deployed by the SAM build command, it needs to be re-built by using Apache Maven and uploaded manually to AWS.

Install Java JDK and Apache Maven

If such build is required, install the Java SDK and Apache Maven as described below:

1. Install Java JDK (to be used for Apache Maven). Set JAVA_HOME as environment variable and in the path.
2. Install Apache Maven and set the environment variables MAVEN_HOME, M2_HOME and in the Path.

Example of Maven and Java environment variables on Windows 10:

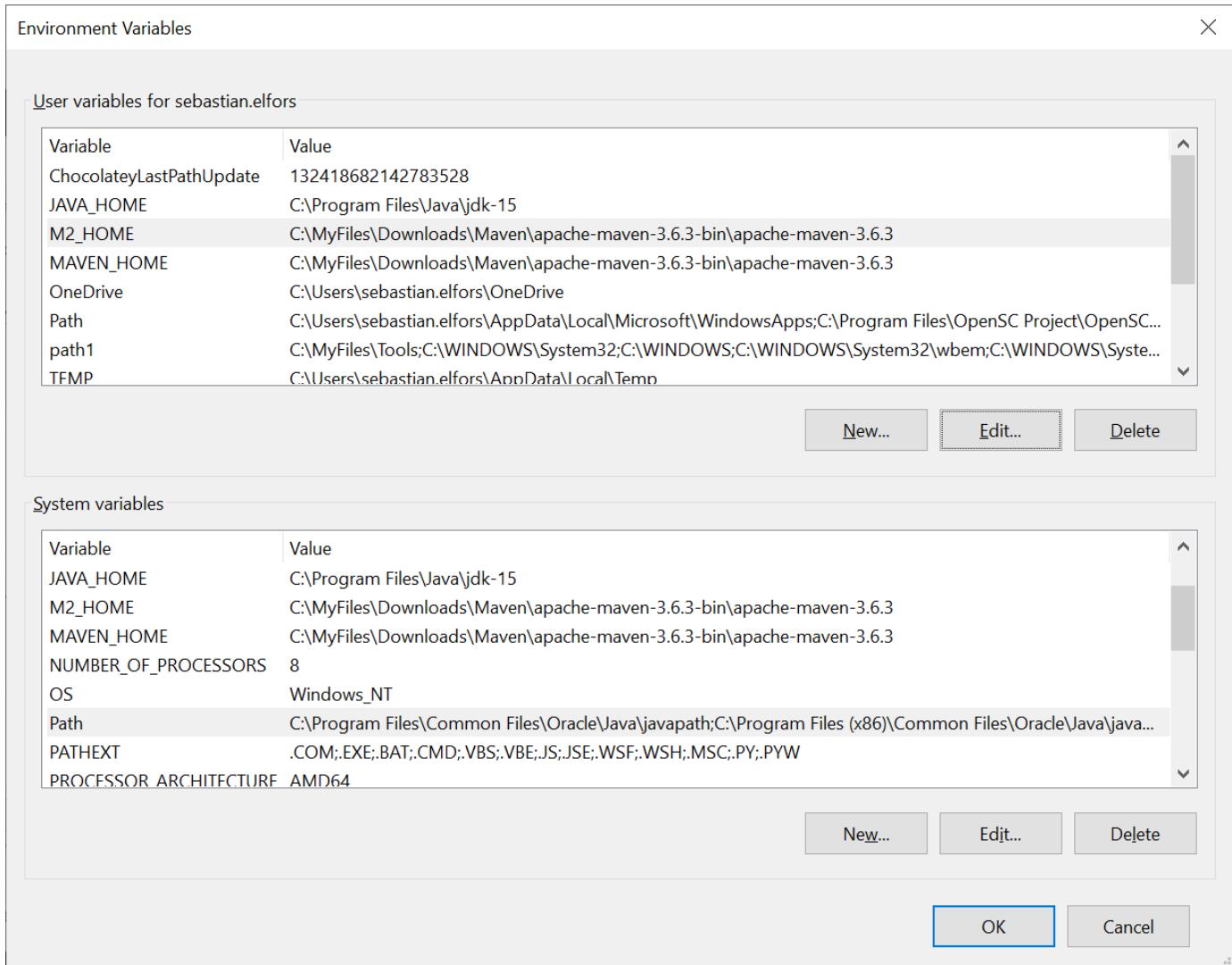
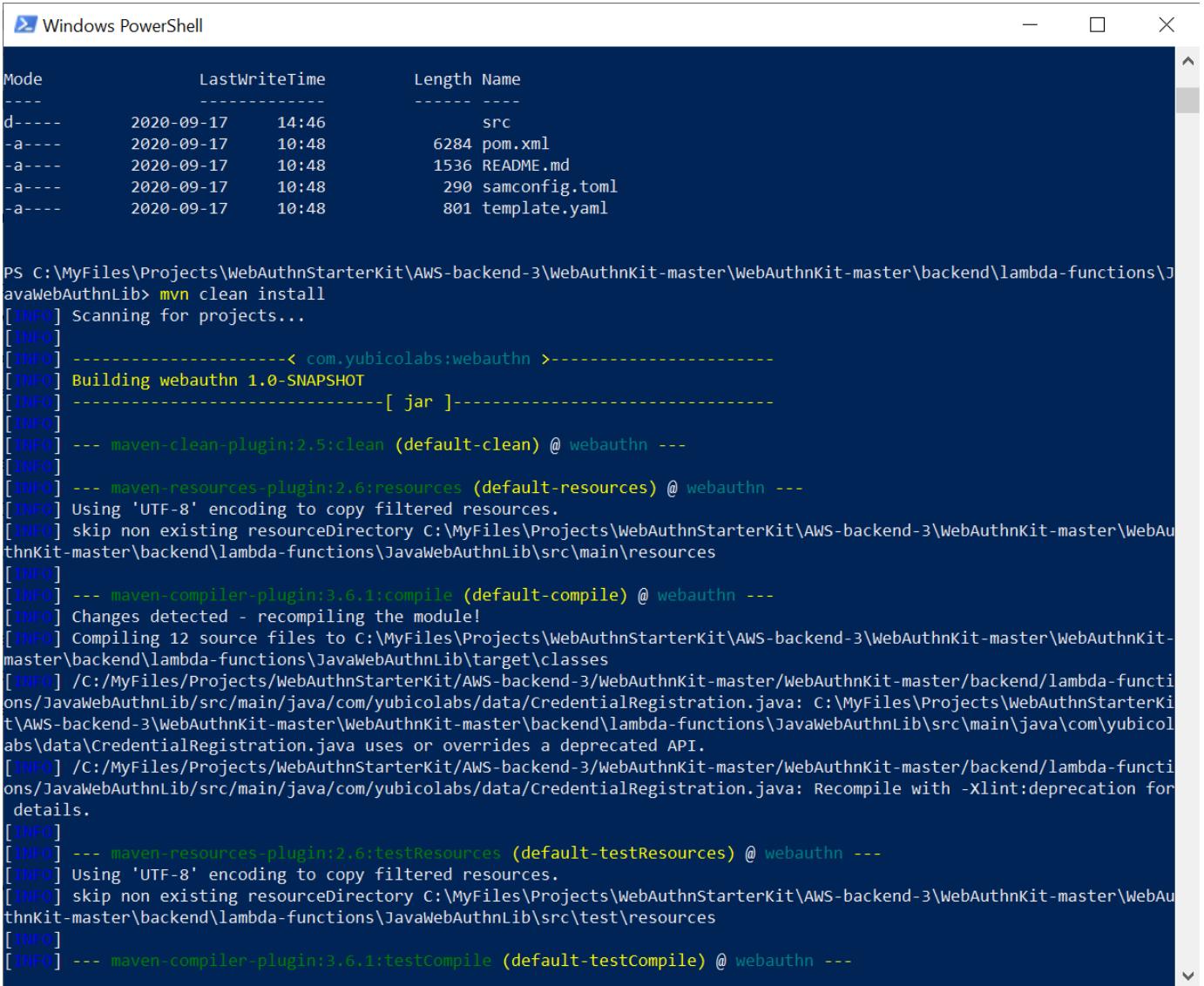


Figure 12 - Windows environment variables for Apache Maven and Java

Re-build the JavaWebAuthnLib.jar

At the workstation, start a Terminal, Command Prompt or PowerShell. Then navigate to the folder ~/WebAuthnKit/backend/lambda-functions/JavaWebAuthnLib/. Then execute the command:

```
mvn clean install
```



```

Windows PowerShell

Mode          LastWriteTime    Length Name
----          -----        ---- 
d----          2020-09-17    14:46      src
-a---          2020-09-17    10:48    6284 pom.xml
-a---          2020-09-17    10:48   1536 README.md
-a---          2020-09-17    10:48    290 samconfig.toml
-a---          2020-09-17    10:48   801 template.yaml

PS C:\MyFiles\Projects\WebAuthnStarterKit\AWS-backend-3\WebAuthnKit-master\WebAuthnKit-master\backend\lambda-functions\JavaWebAuthnLib> mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] [ com.yubicolabs:webauthn ] -----
[INFO] Building webauthn 1.0-SNAPSHOT
[INFO] [ jar ]
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ webauthn ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ webauthn ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\MyFiles\Projects\WebAuthnStarterKit\AWS-backend-3\WebAuthnKit-master\WebAuthnKit-master\backend\lambda-functions\JavaWebAuthnLib\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.6.1:compile (default-compile) @ webauthn ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 12 source files to C:\MyFiles\Projects\WebAuthnStarterKit\AWS-backend-3\WebAuthnKit-master\WebAuthnKit-master\backend\lambda-functions\JavaWebAuthnLib\target\classes
[INFO] /C:/MyFiles/Projects/WebAuthnStarterKit/AWS-backend-3/WebAuthnKit-master/WebAuthnKit-master/backend/Lambda-functions/JavaWebAuthnLib/src/main/java/com/yubicolabs/data/CredentialRegistration.java: C:\MyFiles\Projects\WebAuthnStarterKit\AWS-backend-3\WebAuthnKit-master\WebAuthnKit-master\backend\lambda-functions\JavaWebAuthnLib\src\main\java\com\yubicolabs\data\CredentialRegistration.java uses or overrides a deprecated API.
[INFO] /C:/MyFiles/Projects/WebAuthnStarterKit/AWS-backend-3/WebAuthnKit-master/WebAuthnKit-master/backend/Lambda-functions/JavaWebAuthnLib/src/main/java/com/yubicolabs/data/CredentialRegistration.java: Recompile with -Xlint:deprecation for details.
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ webauthn ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\MyFiles\Projects\WebAuthnStarterKit\AWS-backend-3\WebAuthnKit-master\WebAuthnKit-master\backend\lambda-functions\JavaWebAuthnLib\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.6.1:testCompile (default-testCompile) @ webauthn ---

```

Figure 13 - Re-building the webauthnlib.jar

This command will build the JAR-file ~/WebAuthnKit/backend/lambda-functions/JavaWebAuthnLib/target/webauthn.jar.

Re-deploy JavaWebAuthnLib.jar

There are two ways to re-deploy the JavaWebAuthnLib.jar file:

- Manual deployment at the AWS console, or
- Using the AWS CLI for scripted deployment

Manual deployment of JavaWebAuthnLib.jar at the AWS console

Login to the AWS console, and select the Service called Lambda.

The screenshot shows the AWS Lambda Functions list page. On the left, there's a navigation sidebar with links like Dashboard, Applications, Functions (which is selected), Additional resources (Layers), and Related AWS resources (Step Functions state machines). The main area displays a table titled "Functions (7)" with columns for Function name, Description, Runtime, Code size, and Last modified. The functions listed are:

Function name	Description	Runtime	Code size	Last modified
JavaWebAuthLibSeb		Java 8	22.4 MB	21 hours ago
DefineAuthSeb		Node.js 12.x	1.3 kB	yesterday
WebAuthnKitAPIFuncSeb		Node.js 12.x	172.7 kB	yesterday
CreateAuthSeb		Node.js 12.x	27.9 kB	yesterday
CreateDBSeb		Node.js 12.x	3.2 kB	yesterday
PreSignUpSeb		Node.js 12.x	629 bytes	yesterday
VerifyAuthSeb		Node.js 12.x	696.6 kB	yesterday

Figure 14 - Listing the Lambda Functions

In the list of Function names, select the Function called “JavaWebAuthLib<suffix>”. In this example, the Function is called “JavaWebAuthLibSeb”.

The screenshot shows the AWS Lambda function configuration page for "JavaWebAuthLibSeb". At the top, there are tabs for CloudWatch Management Console and IAM Management Console. The main area has tabs for Throttle, Qualifiers, Actions, Select a test event, Test, and Save. Below these are sections for Function code and Environment variables.

Function code (Info): A note says the deployment package is too large for inline code editing. It includes options to upload a .zip or .jar file or a file from Amazon S3.

Environment variables (3): A note says variables are encrypted at rest. An "Edit" button is shown. The table lists environment variables:

Key	Value
DBAuroraClusterArn	arn:aws:rds:eu-west-2:764504484168:cluster:fido2dbseb
DBSecretsStoreArn	arn:aws:secretsmanager:eu-west-2:764504484168:secret:fido2dbseb-5Ieyww

Figure 15 - Checking the Lambda Function's environment variables

Press the “Actions” button, and select the option “Upload a .zip or .jar file”.

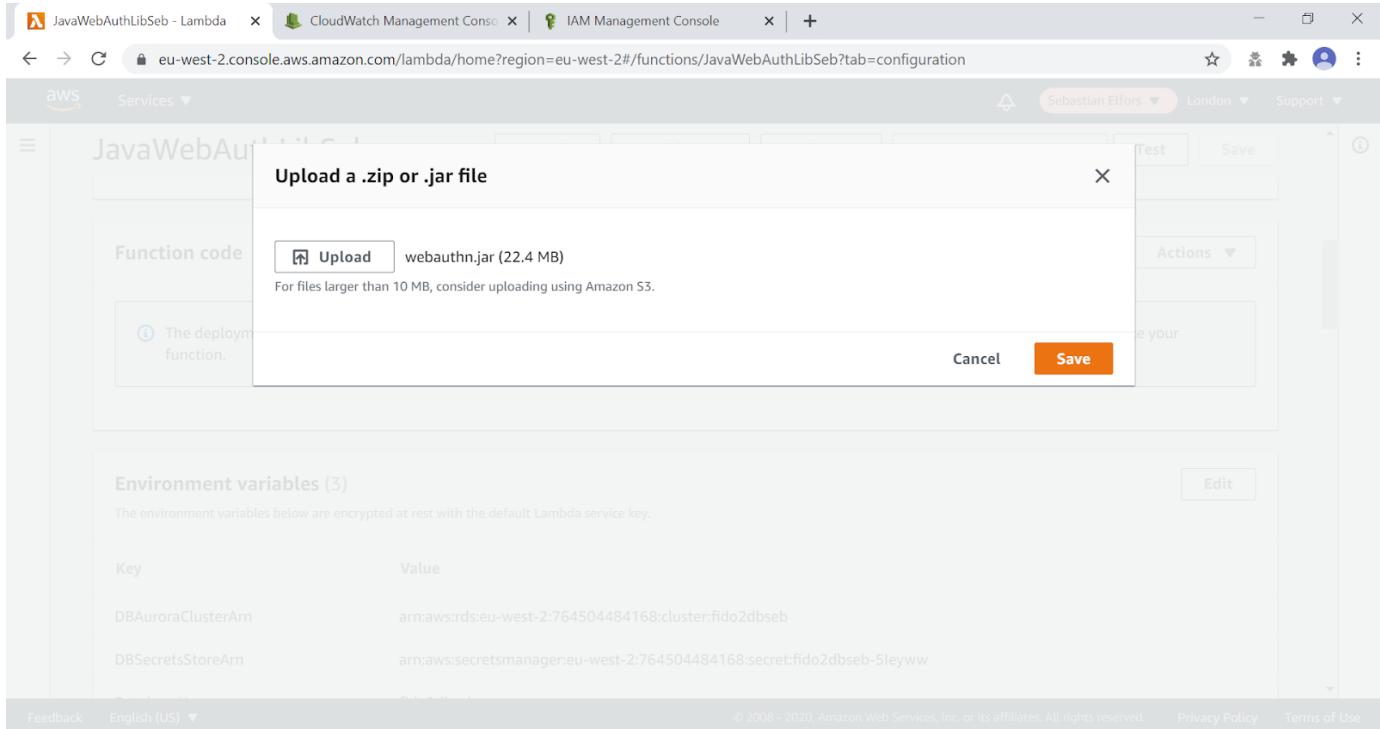


Figure 16 - Uploading the webauthn.jar file to AWS

Select the JAR-file `~/WebAuthnKit/backend/lambda-functions/JavaWebAuthnLib/target/webauthn.jar` at the workstation, and press the Save button. This completes the upload of the webauthn.jar file to AWS.

AWS CLI for scripted deployment of JavaWebAuthnLib.jar

It is also possible to use the AWS CLI for scripting the upload of the JavaWebAuthnLib.jar to the AWS S3 bucket and deploying it at the AWS Lambda function. See the AWS documentation for details on how to perform the upload and deployment using AWS CLI.

Below are examples of the AWS CLI commands needed for uploading and deploying JavaWebAuthnLib.jar to AWS S3 and AWS Lambda.

```
aws s3 cp ~\WebAuthnKit-master\WebAuthnKit-master\backend\lambdafunctions\JavaWebAuthnLib\target\webauthn.jar s3://<bucket-name>
aws lambda update-function-code --function-name JavaWebAuthnLib$suffix --s3-bucket $s3BucketName --s3-key webauthn.jar
```

Attach the AWSLambdaRole policy

If the AWS Lambda Role policy is not properly attached, take the following actions to attach the AWS Lambda Role policy to the RDSLambdaExecutionRole.

Login to the AWS console, and select the Service called Lambda.

The screenshot shows the AWS Lambda Functions list page. On the left, there's a sidebar with navigation links like Dashboard, Applications, Functions, Additional resources (Layers), and Related AWS resources (Step Functions state machines). The main area is titled "Functions (7)" and lists the following functions:

Function name	Description	Runtime	Code size	Last modified
JavaWebAuthLibSeb		Java 8	22.4 MB	21 hours ago
DefineAuthSeb		Node.js 12.x	1.3 kB	yesterday
WebAuthnKitAPIFuncSeb		Node.js 12.x	172.7 kB	yesterday
CreateAuthSeb		Node.js 12.x	27.9 kB	yesterday
CreateDBSeb		Node.js 12.x	3.2 kB	yesterday
PreSignUpSeb		Node.js 12.x	629 bytes	yesterday
VerifyAuthSeb		Node.js 12.x	696.6 kB	yesterday

Figure 17 - Listing the Lambda Functions

In the list of Function names, select the Function called “CreateAuth<suffix>”. In this example, the Function is called CreateAuthSeb.

In the next window, select the Permissions tab to display the Execution role. Press on the link under the Execution Role -> Role Name.

The screenshot shows the Lambda function details page for "CreateAuthSeb". The top navigation bar includes links for CreateAuthSeb - Lambda, IAM Management Console, and eu-west-2.console.aws.amazon.com. The main area shows the function configuration with tabs for Configuration, Permissions (which is selected), and Monitoring. Under the "Execution role" section, the Role name is listed as "webauthnkit-seb-RDLSLambdaExecutionRole-19M44EEDX9BN8". The "Resource summary" section shows "AWS Secrets Manager" with "1 action, 2 resources".

Figure 18 - Changing the execution role of a Lambda Function

Click the link under the Role name and the following window appears:

Identity and Access Management (IAM)

Dashboard

Access management

Groups

Users

Roles

Policies

Identity providers

Account settings

Access reports

Access analyzer

Archive rules

Analyzers

Settings

Credential report

Feedback English (US) ▾

Role ARN: arn:aws:iam::76450448168:role/webauthnkit-seb-RDLSLambdaExecutionRole-19M44EEDX9BN8

Role description: Edit

Instance Profile ARNs: [Edit](#)

Path: /

Creation time: 2020-09-17 15:05 UTC+0200

Last activity: 2020-09-17 15:35 UTC+0200 (Today)

Maximum session duration: 1 hour [Edit](#)

Permissions [Trust relationships](#) [Tags](#) [Access Advisor](#) [Revoke sessions](#)

Permissions policies (3 policies applied)

[Attach policies](#) [Add inline policy](#)

Policy name	Policy type
LambdaToAWSecrets	Inline policy

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Figure 19 - Changing the execution role of a Lambda Function

In the window above, click the button “Attach policies”, and the following window appears:

Add permissions to webauthnkit-seb-RDLSLambdaExecutionRole-19M44EEDX9BN8

Attach Permissions

[Create policy](#)

[Filter policies](#)

Showing 1 result

Policy name	Type	Used as
<input checked="" type="checkbox"/> AWSLambdaRole	AWS managed	Permissions policy (1)

[Cancel](#) [Attach policy](#)

Feedback English (US) ▾

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Figure 20 - Attaching the policy to a Lambda Function

Mark the checkbox next the AWSLambdaRole and press the button “Attach policy”. The following window appears:

Identity and Access Management (IAM)

Roles > webauthnkit-seb-RDLSLambdaExecutionRole-19M44EEDX9BN8

Summary

Policy AWSLambdaRole has been attached for the webauthnkit-seb-RDLSLambdaExecutionRole-19M44EEDX9BN8.

Role ARN	arn:aws:iam::764504484168:role/webauthnkit-seb-RDLSLambdaExecutionRole-19M44EEDX9BN8
Role description	Edit
Instance Profile ARNs	Edit
Path	/
Creation time	2020-09-17 15:05 UTC+0200
Last activity	2020-09-17 15:35 UTC+0200 (Today)
Maximum session duration	1 hour Edit

Permissions **Trust relationships** **Tags** **Access Advisor** **Revoke sessions**

▼ Permissions policies (4 policies applied)

[Attach policies](#) [Add inline policy](#)

Figure 21 - Attaching the policy to a Lambda Function

Assigning the policy to one Lambda execution role will update all Lambda roles, so it's sufficient to perform this step once.

Enable the CORS policy at API Gateway

If the CORS policy is not properly enabled at the API Gateway, take the following actions to enable the CORS policy.

Login to the AWS console, and select the Service called API Gateway.

APIs (1)

Name	Description	ID	Protocol	Endpoint type	Created
WebAuthnKitAPISeb	WebAuthn Starter API for User Credential Management	rgbvqyw1	REST	Edge	2020-09-17

Actions [Create API](#)

Figure 22 - Enabling the CORS policy at API Gateway (step 1)

In the list under APIs, select the API named “WebAuthnKitAPI<suffix>”. In this example, the API is called “WebAuthnKitAPISeb”. The following window appears:

The screenshot shows the AWS API Gateway interface. The left sidebar lists various API components: APIs, Custom Domain Names, VPC Links, API: WebAuthnKitAPI..., Resources, Stages, Authorizers, Gateway Responses, and Models. The 'Models' section is currently selected, indicated by an orange border. A 'Create' button is prominently displayed at the top right of the main content area. The main content area is titled 'Select a model'.

Figure 23 - Enabling the CORS policy at API Gateway (step 2)

In this window, press the “Create” button.

The screenshot shows the 'New Model' creation dialog. The left sidebar is identical to Figure 23. The main content area is titled 'New Model' with the sub-instruction 'Provide a name, content type, and a schema for your model. Models use JSON schema.' Below this, there are three input fields: 'Model name*' (set to 'Empty'), 'Content type*' (set to 'application/json'), and 'Model description'. At the bottom, there is a section titled 'Model schema*' containing a JSON schema editor. The schema is currently set to an empty object:

```

1  [
2    "$schema": "http://json-schema.org/draft-04/schema#",
3    "title" : "Empty Schema",
4    "type" : "object"
5  ]

```

Figure 24 - Enabling the CORS policy at API Gateway (step 3)

In the “New Model” pane, enter the following information:

- Model name: Can be set to anything, for example “Empty”.
- Content type: Must be set to “application/json”

- Model description: Can be left blank.
- In the Model schema, insert the following schema:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Empty Schema",
  "type": "object"
}
```

Then press the “Create” button to create the model.

In the AWS console, go back to Services called API Gateway, and select Resources in the left-hand pane.

The credentials methods will be listed, as shown in the screenshot below.

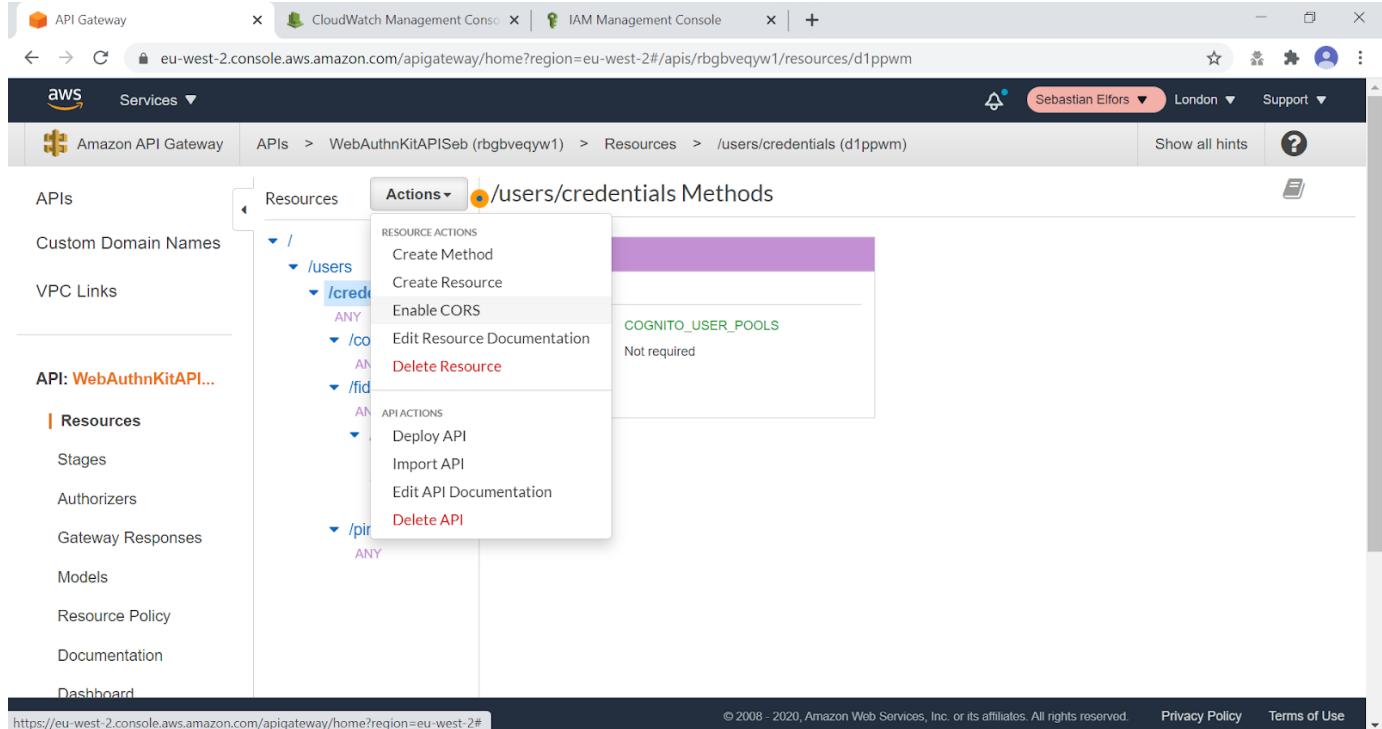
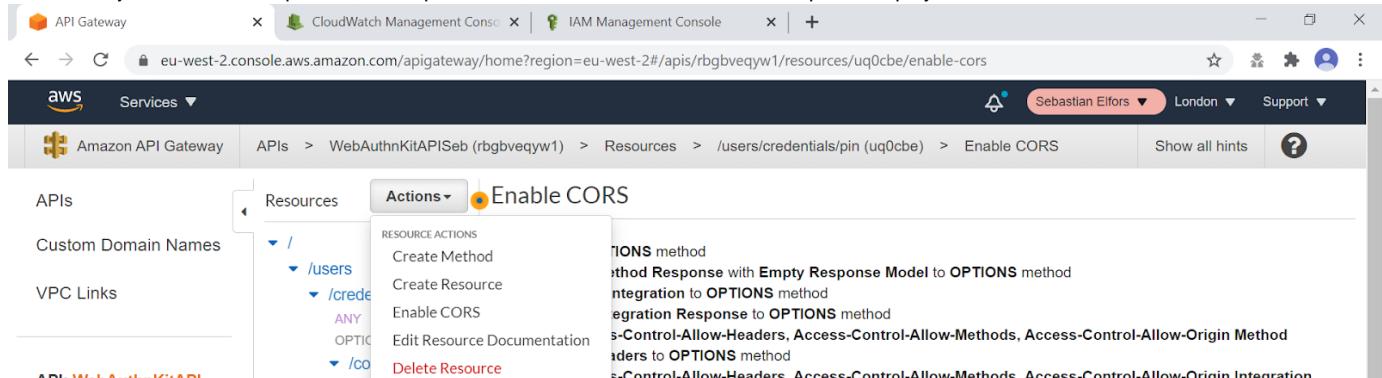


Figure 25 - Enabling the CORS policy at API Gateway (step 4)

For each object called “/credentials”, “/codes”, “/fido2”, “/register”, “/pin” do the following:

- Highlight the object.
- Press the Actions button, and select the option “Enable CORS” in the drop-down menu.

When all objects have been updated, select press the Actions button and select the option “Deploy API”:



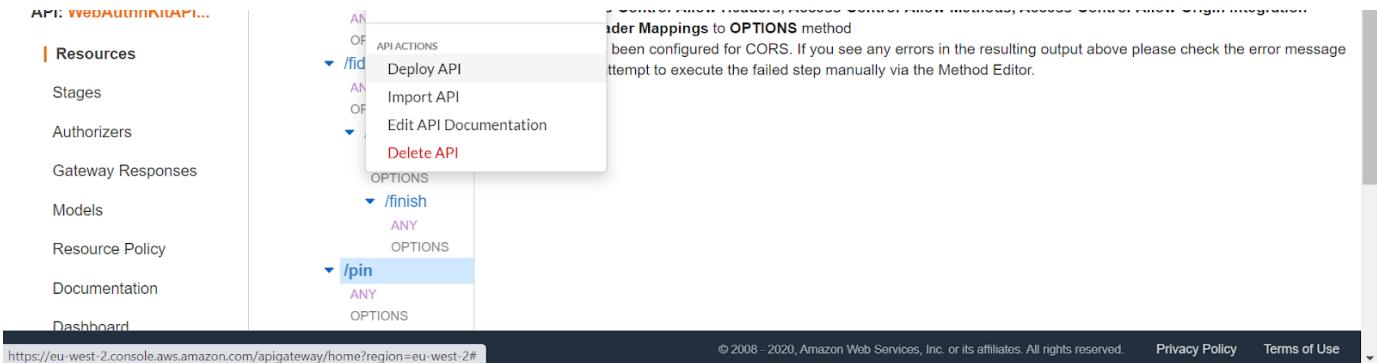


Figure 26 - Enabling the CORS policy at API Gateway (step 5)

In the Deploy API window, select Deployment stage “dev” and insert a deployment description. Press the Deploy button.

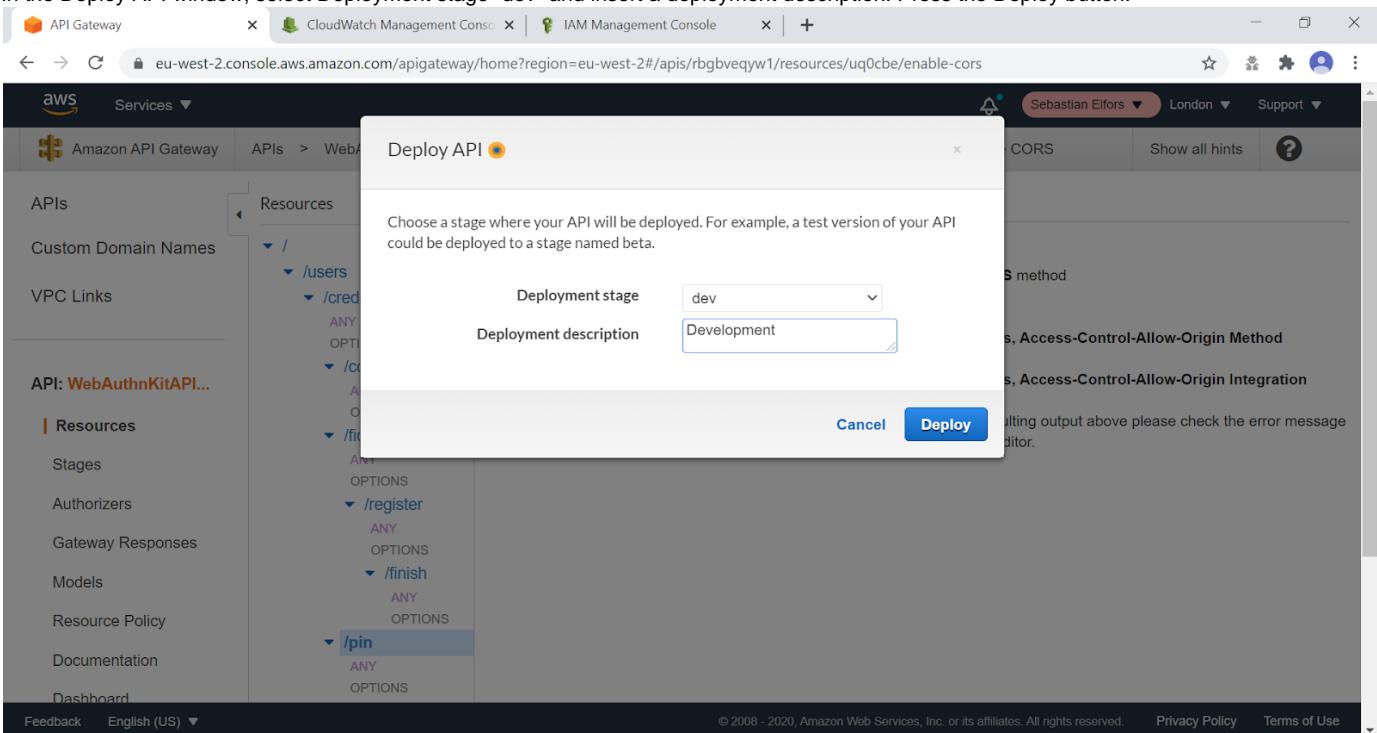
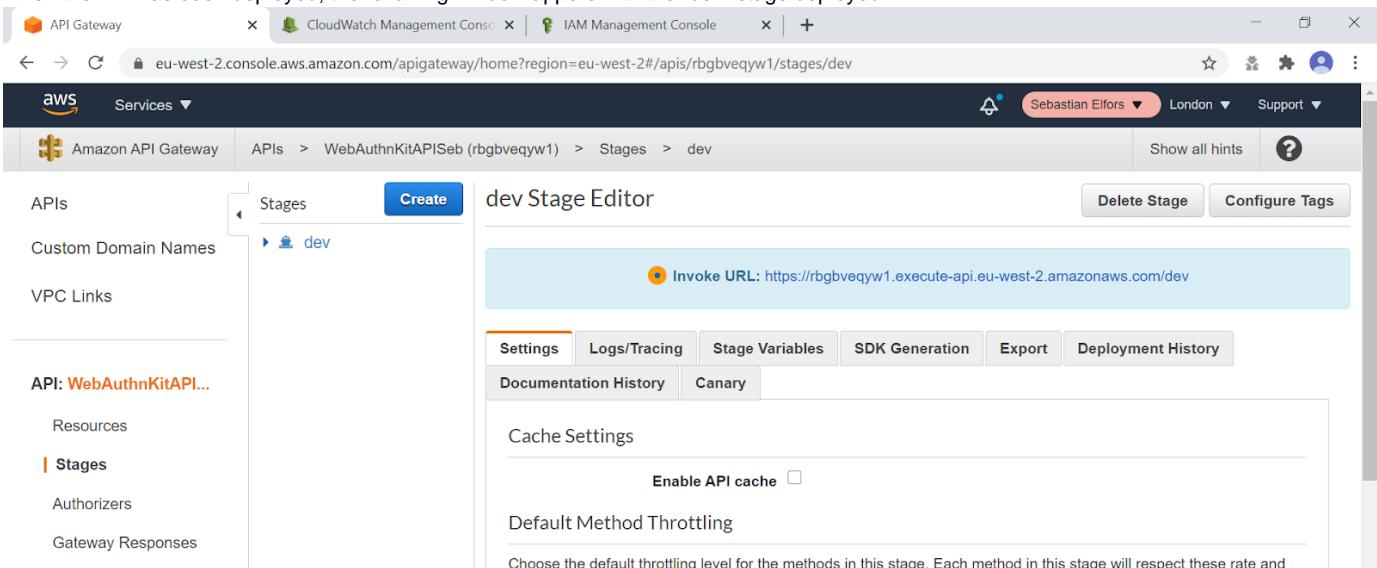


Figure 27 - Enabling the CORS policy at API Gateway (step 6)

When the API has been deployed, the following window appears with the “dev” stage deployed.



burst settings. Your current account level throttling rate is **10000** requests per second with a burst of **5000** requests.
[Read more about API Gateway throttling](#)

Enable throttling

Rate requests per second

Figure 28 - Enabling the CORS policy at API Gateway (step 7)

This completes the CORS deployment checks.

Delete stack, S3 bucket and Amplify apps after roll back

If the deployment fails and is rolled back by AWS CloudFormation, the AWS CloudFormation stack, AWS S3 bucket and potentially AWS Amplify app must be deleted before it is deployed again, particularly if the same suffix is used.

Delete the AWS CloudFormation stack

The CloudFormation stacks can be found in the AWS Console under Services CloudFormation Stacks.

Stack name	Status	Created time	Description
amplify-reactwebauthnseb35-dev-173454- NESTED	CREATE_COMPLETE	2020-10-12 17:36:05 UTC+0200	Branch stack creation for AWS Ampli...
amplify-reactwebauthnseb35-dev-173454	UPDATE_COMPLETE	2020-10-12 17:34:56 UTC+0200	Root stack for the Amplify AWS Clou...
webauthnkit-seb35	CREATE_COMPLETE	2020-10-12 17:27:48 UTC+0200	WebAuthn Starter Kit - Backend Serv...
amplify-reactwebauthnseb34-dev-142777- NESTED	CREATE_COMPLETE	2020-10-12 14:33:08 UTC+0200	Branch stack creation for AWS Ampli...
amplify-reactwebauthnseb34-dev-142777	UPDATE_COMPLETE	2020-10-12 14:32:02 UTC+0200	Root stack for the Amplify AWS Clou...
webauthnkit-seb34	CREATE_COMPLETE	2020-10-12 14:24:04 UTC+0200	WebAuthn Starter Kit - Backend Serv...
... (truncated)	CREATE_COMPLETE	2020-10-12 17:10:27 UTC+0200	WebAuthn Starter Kit - Backend Serv...

Figure 29 - Listing the CloudFormation Stacks

Select the stack to be deleted, and it will be displayed in a new window (as shown below).

amplify-reactwebauthnseb35-dev-173454

Stack info Events Resources Outputs Parameters Template Change sets

Overview

Stack ID	Description
amplify-reactwebauthnseb35-dev-173454-ho...	... (truncated)

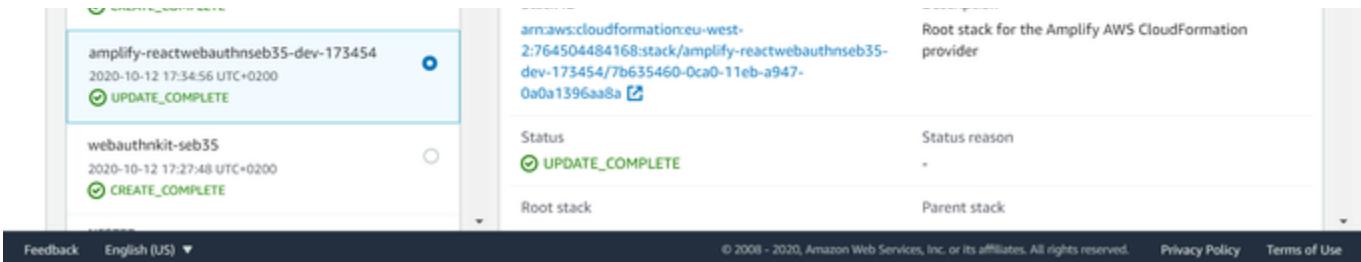


Figure 30 - Deleting a CloudFormation Stack

Press the Delete button to delete the CloudFormation stack.

Delete the AWS S3 bucket

The S3 bucket can be found in the AWS Console under Services Amazon S3.

			Buckets	Regions
<input type="checkbox"/>		seb-webauthnkit-seb29-2	Bucket and objects not public	EU (London)
<input type="checkbox"/>		seb-webauthnkit-seb30	Bucket and objects not public	EU (London)
<input type="checkbox"/>		seb-webauthnkit-seb31	Bucket and objects not public	EU (London)
<input type="checkbox"/>		seb-webauthnkit-seb32	Bucket and objects not public	EU (London)
<input type="checkbox"/>		seb-webauthnkit-seb33	Bucket and objects not public	EU (London)
<input checked="" type="checkbox"/>		seb-webauthnkit-seb34	Bucket and objects not public	EU (London)
<input type="checkbox"/>		seb-webauthnstarterkit-eu-north-1	Bucket and objects not public	EU (Stockholm)
<input type="checkbox"/>		seb-webauthnstarterkit-eu-west-2	Bucket and objects not public	EU (London)
<input type="checkbox"/>		seb-webauthnstarterkit-eu-west-2-1	Bucket and objects not public	EU (London)
<input type="checkbox"/>		seb-webauthnstarterkit-seb3	Bucket and objects not public	EU (London)
<input type="checkbox"/>		seb-webauthnstarterkit-seb4	Bucket and objects not public	EU (London)
<input type="checkbox"/>		sebastian-fidoreact	Bucket and objects not public	EU (Ireland)

Figure 31 - Deleting an S3 Bucket

Select the S3 bucket to be deleted and press the Delete button.

Delete the AWS Amplify apps

Also make sure that the AWS Amplify apps have been deleted. The AWS Amplify apps can be found in the AWS Console under Services AWS Amplify.



Figure 32 - Selecting an Amplify web app

Select the AWS Amplify to be deleted, which shows the app information below.

The screenshot shows the AWS Amplify Console interface. On the left, a sidebar lists 'All apps' with 'react' selected. The main area displays the 'react' app homepage, which includes a 'Frontend environments' tab showing a 'dev' branch with a successful deployment message. On the right, an 'Actions' dropdown menu is open, with 'Delete app' highlighted. The browser's address bar shows the URL `eu-west-2.console.aws.amazon.com/amplify/home?region=eu-west-2#/d3f1or1kne0rl5`.

Figure 33 - Deleting an Amplify web app

Select Actions and press the option “Delete app” in the dropdown list.