This is documentation for the developer preview release of the AWS SDK for Rust. Do not use it in production as it is subject to breaking changes.

Getting started with the AWS SDK for Rust

PDF (/pdfs/sdk-for-rust/latest/dg/aws-sdk-rust-developer-guide.pdf#getting-started) | **RSS (aws-sdk-rust-developer-guide.rss)**

This chapter describes how to get started with the AWS SDK for Rust (the SDK). It includes information about getting an AWS account, installing the SDK, and creating a "Hello world" code example that lists all of your Amazon DynamoDB (DynamoDB) tables in a region.

Topics

Prerequisites

Before you can use the SDK, you will need an AWS account and get your AWS access keys.

Get an AWS account

To get an account, navigate to the Amazon Web Services signup page (http://portal.aws.amazon.com/billing/signup) and follow the on-screen prompts to create and activate a new account. For detailed instructions,

see How do I create and activate a new AWS account? (http://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/) .

After you activate your new AWS account, follow the instructions in Creating your first IAM admin user and group (https://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_create-admin-group.html#getting-started_create-admin-group-console) in the IAM User Guide (https://docs.aws.amazon.com/IAM/latest/UserGuide/) . Use this account instead of the root account when signing in to the AWS Management Console or making requests to AWS services programmatically. For more information, see Security best practices in IAM (https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#lock-away-credentials) in the IAM User Guide (https://docs.aws.amazon.com/IAM/latest/UserGuide/) .

Get and store your access keys

To access AWS resources, you need to use credentials for an IAM user that has read and write access to AWS services. To make requests to AWS services using the AWS SDK for Rust, you must have an access key and a secret access key to use as credentials.

1. Sign in to the IAM console (https://console.aws.amazon.com/iam/)
2. In the navigation pane on the left, select **Users**.
3. Then select **Add user**.
4. Enter the value **DynamoReader** for **User name** and select **Programmatic access**.
5. Select **Next: Permissions**.
6. Under **Set permissions**, select **Attach existing policies directly**.
7. In the list of policies, select **AmazonDynamodbReadOnlyAccess**.
8. Select **Next: Tags**.
9. Select **Next: Review**.
10. Select **Create user**.
11. On the *Success* screen, select **Download .csv**.

    The downloaded file contains you Access Key ID and the Secret Access Key for this user. Treat these with care, as anyone with these values

can create resources, which might cost you dearly; save their values in a trusted location and do not share them.

> **Note**
>
> You will not have another opportunity to download or copy these values, although you can always create new ones.

Now that you have your access key and secret access key, store them in a location where the SDK can access them.

On Linux and MacOS, this file is found at **~/.aws/credentials**. On Microsoft Windows it is found at **%USERPROFILE%\.aws\credentials**.

1. If the **credentials** file does not exist, create it.
2. Add the following to the file, where *YOUR-ACCESS-KEY* is the value of your access key and *YOUR-SECRET-KEY* is the value of your secret key:

```
[default]
aws_access_key_id=YOUR-ACCESS-KEY
aws_secret_access_key=YOUR-SECRET-KEY
```

You can also specify the default AWS Region for your API requests in the **credentials** file, such as in the following example, which makes **us-west-2** your default Region.

```
region=us-west-2
```

Create your first SDK app

Let's create a simple Rust app that lists your DynamoDB tables.

1. Navigate to a location on your computer where you want to create the app.
2. Execute the following commands to create a **hello_world** directory and populate it with a skeleton Rust project:

```
cargo new hello_world --bin
```

3. Install the aws-config (https://crates.io/crates/aws-config) and aws-sdk-dynamodb (https://crates.io/crates/aws-sdk-dynamodb) crates from **crates.io**, as described in the previous section. Note the version number of each crate.
4. Navigate into the **hello_world** directory and edit **Cargo.toml** to include these crates in **[dependencies]**, where *VERSION* is the version number you found on **crates.io**:

```
[dependencies]
aws-config = "VERSION"
aws-sdk-dynamodb = "VERSION"
tokio = { version = "1", features = ["full"] }
```

5. Update **main.rs** in the **src** directory to include the following code.

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_dynamodb::{Client, Error};

/// Lists your DynamoDB tables in the default
Region or us-east-1 if a default Region isn't set.
#[tokio::main]
async fn main() -> Result<(), Error> {
    let region_provider =
RegionProviderChain::default_provider().or_else("u
s-east-1");
    let config =
aws_config::from_env().region(region_provider).loa
d().await;
    let client = Client::new(&config);

    let resp = client.list_tables().send().await?;

    println!("Tables:");

    let names =
```

```
resp.table_names().unwrap_or_default();

    for name in names {
        println!("  {}", name);
    }

    println!();
    println!("Found {} tables", names.len());

    Ok(())
}
```

6. Run the program:

```
cargo run
```

   You should see a list of your table names.