Liquibase

Connect to Azure Blob Storage

Last updated: July 24, 2025

To connect Liquibase to Azure Blob Storage, you authenticate using Azure service principal credentials and reference your files with `az://` URLs. This setup enables Liquibase to securely read from and write to Azure Blob Storage when running commands, such as `update`, `snapshot`, `diff`, `defaults-file`, `flow-file`, and `changelog-file`. The connection is established automatically at runtime when Liquibase detects an `az://` path and valid Azure credentials.

This guide will walk you through setting up your Azure credentials as environment variables and testing your ability to establish a connection to Azure.

Procedure

Filter By Operating System

1 Set up environment variables

Environment variables are key-value pairs used to securely pass configuration data such as credentials and connection details to applications.

To connect Liquibase to Azure Blob Storage, you must set specific environment variables that allow Liquibase to authenticate with your Azure storage account. These variables are used at runtime to establish a secure connection and authorize access to blob storage resources.

You can use the following example code in the command-line interface (CLI) to set these environment variables before running Liquibase commands.

Before you run the example code, be sure to:

- Update *your_storage_account* with your storage account name.

- Update *your_tenant_id* with your tenant ID.
- Update *your_client_id* with your client ID.
- Update *your_client_secret* with your client secret.

Windows

```
set LIQUIBASE_AZURE_STORAGE_ACCOUNT="your_storag
set AZURE_TENANT_ID="your_tenant_id"
set AZURE_CLIENT_ID="your_client_id"
set AZURE_CLIENT_SECRET="your_client_secret"
```

Linux/macOS

```
export LIQUIBASE_AZURE_STORAGE_ACCOUNT="your_storage
export AZURE_TENANT_ID="your_tenant_id"
export AZURE_CLIENT_ID="your_client_id"
export AZURE_CLIENT_SECRET="your_client_secret"
```

Note: This sets the environment variables temporarily for the duration of the current terminal session. Once you close the terminal, this setting will be lost, and you will need to reset the environment variables for your new terminal session. These variables include sensitive information like client secrets, so we do not recommend persisting them to avoid long-term storage in configuration files.

2 Verify the connection

To confirm that Liquibase is successfully connected to Azure Blob Storage, you can run the snapshot command to output a file directly to your Blob Storage container using the following example code.

Before you run the example code, be sure to:

- Replace *your_blob_container* with the name of your blob storage container in Azure where you would like to upload the file.

```
liquibase snapshot \
--snapshot-format=json \
--output-file="az://your_blob_container/test-snapsho
```

After running the command, go to the Azure portal, and navigate to your Blob Storage account and container to verify that the `test-snapshot.json` file was uploaded successfully.

What's next

- Store Snapshots, Diffs, and SQL Output from Liquibase in Azure Blob Storage — Now that you've connected to azure and stored a file, you can write additional Liquibase files to Azure Blob Storage, allowing you to store them centrally and access them remotely.
- Load and Execute Changelogs, Flows, and Properties from Azure Blob Storage — You can load and execute Liquibase files directly from Azure Blob Storage, enabling you to run database changes without storing the files on your local system.