# Using your account from the AWS CLI

Sandbox Studio gives you access to your sandbox account through an IAM role that it creates for you inside the account. Because there is no IAM Identity Center assignment, `aws sso login` **does not work** for sandbox accounts. It fails with:

```
An error occurred (ForbiddenException) when calling the GetRoleCredentials operation: No access
```

Instead, you can set up a normal AWS CLI profile that fetches short-lived credentials from Sandbox Studio automatically. Once it is configured, you use it like any other profile (`--profile my-lease`) and the CLI refreshes credentials for you.

> **Just need credentials quickly?** Open your account in Sandbox Studio and use the credentials option to copy `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` and `AWS_SESSION_TOKEN`. They are valid for 1 hour. The setup below avoids having to copy them again each time.

## How it works

1. A small helper script authenticates to the Sandbox Studio API with a personal **API client** (client ID + secret).
2. It asks Sandbox Studio for temporary credentials for your lease (`POST /api/leases/{leaseId}/access`).
3. It prints them in the format the AWS CLI expects for `credential_process`.

The CLI calls the script whenever it needs credentials and again when they expire.

## Prerequisites

- An **Active** account (lease) in Sandbox Studio, and its **lease ID**. The lease ID appears in the lease URL in Sandbox Studio, for example `.../leases/11111111-2222-3333-4444-555555555555`.
- **AWS CLI v2**
- `curl` and `jq`
- A secure secret store. The examples use the **macOS Keychain** (`security` command). Linux alternatives are described below.
- Your Sandbox Studio address, for example `https://sandbox.example.com`

# Step 1 – Create an API client

1. Sign in to Sandbox Studio.
2. Open your **API clients** page (or **Integrations → API**) and create a new client.
3. Copy the **Client ID** and **Client Secret**. The secret is shown only once.

> **Important:** The API client has **the same permissions as you**. If you are a Manager or Admin, the secret is equally powerful. Never commit it, paste it into chat or tickets, or put it in `.zshrc` / `.env`. If it leaks, delete the client and create a new one.

# Step 2 – Store the client ID and secret securely

## macOS (Keychain)

Run each command. Each one **prompts** for the value, so the secret never appears in your shell history:

```
security add-generic-password -U -a "$USER" -s sandbox-studio-client-id -w
security add-generic-password -U -a "$USER" -s sandbox-studio-client-secret -w
```

## Linux (libsecret)

```
secret-tool store --label="Sandbox Studio client id" service sandbox-studio-client-id
secret-tool store --label="Sandbox Studio client secret" service sandbox-studio-client-secret
```

On Linux, replace the `kc()` function in the script with:

```
kc() { secret-tool lookup service "$1"; }
```

A password manager CLI (for example `op read "op://Private/sandbox-studio/client_secret"`) works the same way.

# Step 3 – Install the helper script

Save the following as `~/bin/sandbox-creds`, and set `BASE_URL` to your Sandbox Studio address:

```
#!/usr/bin/env bash
# credential_process helper for Sandbox Studio. Needs: curl, jq, a secret store.
# Usage: sandbox-creds [leaseId]   (defaults to your first Active lease)
set -euo pipefail
```

```
BASE_URL="${SANDBOX_STUDIO_URL:-https://sandbox.example.com}"
BASE_URL="${BASE_URL%/}/api"
LEASE_ID="${1:-}"

kc() { security find-generic-password -a "$USER" -s "$1" -w; }
CLIENT_ID=$(kc sandbox-studio-client-id)
CLIENT_SECRET=$(kc sandbox-studio-client-secret)

# The secret is passed through stdin so it never appears in the process list.
TOKEN=$(printf 'grant_type=client_credentials&client_id=%s&client_secret=%s' \
  "$CLIENT_ID" "$CLIENT_SECRET" |
  curl -sSf -X POST "$BASE_URL/oauth/token" \
    -H 'Content-Type: application/x-www-form-urlencoded' --data @- |
  jq -er .access_token)

if [ -z "$LEASE_ID" ]; then
  LEASE_ID=$(curl -sSf "$BASE_URL/leases?status=Active" \
    -H "Authorization: Bearer $TOKEN" |
    jq -er '(.data // .).result[0] | (.leaseId // .id)')
fi

curl -sSf -X POST "$BASE_URL/leases/$LEASE_ID/access" \
  -H "Authorization: Bearer $TOKEN" \
  -H 'Content-Type: application/json' -d '{"role":"USER"}' |
  jq -e '(.data // .).credentials | {Version:1, AccessKeyId:.accessKeyId,
SecretAccessKey:.secretAccessKey, SessionToken:.sessionToken, Expiration:.expiration}'
```

Make the script executable and readable only by you:

```
mkdir -p ~/bin
chmod 700 ~/bin/sandbox-creds
```

Test it. It should print a JSON object containing `"Version": 1`:

```
~/bin/sandbox-creds <lease-id> | jq '{Version, Expiration}'
```

# Step 4 – Add an AWS CLI profile

Add this to `~/.aws/config`. Use the full path to the script; `~` is not expanded here.

```
[profile my-lease]
credential_process = /Users/<you>/bin/sandbox-creds <lease-id>
region = eu-west-2
output = json
```

Set `region` to the region you work in. The profile must not contain `sso_*` settings.

Check that the profile works:

```
aws sts get-caller-identity --profile my-lease
```

You should see an assumed role named `SandboxStudio-…-SandboxRole-…` in your sandbox account, with a session named `sbs-<your user id>`.

> **macOS:** The first time the script runs, Keychain may ask for permission. Click **Always Allow** so the CLI can run the script without prompting.

# Using it

Use the profile like any other profile. There is no login step.

```
aws s3 ls --profile my-lease

# or for the whole terminal session
export AWS_PROFILE=my-lease
aws ec2 describe-instances --region ap-southeast-2
```

The AWS SDKs and tools that read AWS profiles work with `AWS_PROFILE=my-lease`, including boto3, the JavaScript SDK, Terraform and the CDK.

For tools that only read environment variables:

```
eval "$(aws configure export-credentials --profile my-lease --format env)"
```

Exported credentials **do not refresh automatically**. Run the command again after an hour.

## When you get a new lease

Update the lease ID on the `credential_process` line. Alternatively, remove the lease ID from that line and the script uses your first **Active** lease.

# Things to know

- **Credentials last up to 1 hour.** This limit applies regardless of the template's session duration. The CLI fetches new credentials automatically when they expire.
- **Your actions are attributed to you.** Sessions are named after your user and tagged with `UserId` and `LeaseId`, which appear in CloudTrail.
- **Permissions are those of your account template.** The CLI gives you exactly the same access as the console login.
- **Access ends with the lease.** When the lease is suspended, terminated or expired, the script fails and the CLI reports an error.

# Troubleshooting

| Symptom | Cause / fix |
|---|---|
| `ForbiddenException ... GetRoleCredentials` | You are still using an `sso_*` profile. Replace it with `credential_process` (Step 4). |
| `curl: (22) ... 403` on the token call | The URL is wrong (it must include `/api`), or the client ID/secret is wrong. |
| `401` / `403` after it worked before | The API client was deleted or has expired. Create a new client and update the stored secret (Step 2). |
| `403` / `404` on the `/access` call | The lease isn't Active, or it isn't your lease. Check the lease in Sandbox Studio. |
| `jq: error ... null` | No Active lease was found. Pass the lease ID explicitly. |
| A Keychain prompt appears every time | Click **Always Allow**, not **Allow**. |
| `credential_process` errors with no detail | Run `~/bin/sandbox-creds <lease-id>` directly to see the error. |

---

Revision #4
Created 2026-11-06 11:00:31 UTC by Doco
Updated 2026-11-06 11:05:14 UTC by Paul