sf-login-bridge
Onboarding guide

Version 1.0.0 · November 2026 · Self-hosted deployment for one Salesforce org

This guide takes you from nothing to a legacy integration signing in to Salesforce through the bridge. Plan on about an hour, most of it in Salesforce Setup.

1. What you are setting up
2. Before you start
3. Get the bridge
4. Create a key pair
5. Set up Salesforce
6. Configure the bridge
7. Provide https
8. Start, add users, verify
9. Connect your clients
10. Go live
11. Day to day
12. Troubleshooting

---

1. What you are setting up

Salesforce is retiring the SOAP `login()` call (first production breakages 31 January 2027) and the OAuth username password flow (20 February 2027). The bridge answers those two requests so existing software keeps signing in, unchanged apart from its login URL and password.

1. The legacy client calls `login()` on the bridge with its username and a **bridge password**.
2. The bridge checks that password locally. A wrong password stops here and never reaches Salesforce.
3. The bridge asks Salesforce for an access token through your External Client App, using the JWT bearer flow.
4. It returns a standard `LoginResult`: the token as `sessionId`, your org as `serverUrl`.
5. The client sends every data call straight to Salesforce. Record data never passes through the bridge.

No Salesforce password is stored outside Salesforce, and your admin can revoke the bridge's access at any time from Setup.

---

2. Before you start

- ☐ A server or VM with **Docker** that your legacy clients can reach, and outbound https to `*.my.salesforce.com`.
- ☐ **Salesforce admin access** to create an External Client App. Do it in a sandbox first.
- ☐ A **hostname and TLS certificate** for the bridge, or a reverse proxy that already has one.
- ☐ A **licence key** (starts with `SFLB2.`). Start with the **free sandbox licence** from the landing page; buy for production once the sandbox works. Each key covers the orgs it was bought for, and their sandboxes.
- ☐ Access to the **repository**: `<repository URL>`.

- [ ] A list of the legacy integrations to move, with the Salesforce user each one runs as.

> **One Salesforce user per concurrent job.** Salesforce gives every login of a user through the same app one shared session, and a `logout()` ends it for all of them. Jobs that run at the same time need a Salesforce user each. API only Integration User licences suit this.

---

3. Get the bridge

```
git clone <repository URL> sf-login-bridge
cd sf-login-bridge
docker build -t sf-login-bridge:1.0.0 .
```

The rest of the guide works in `deploy/self-hosted/`. Everything the container reads and writes lives in its `data/` folder.

Path	What it is
`deploy/self-hosted/docker-compose.yml`	Runs the container, read only, as an unprivileged user
`bridge.config.example.json`	Starting point for your config
`docs/`	Detailed guides for each client type, operations and security
`src/`	Source code (Rust)

---

4. Create a key pair

The bridge signs its token requests with a private key. Salesforce checks them against the matching certificate.

```
cd deploy/self-hosted
mkdir -p data/keys
openssl req -x509 -newkey rsa:2048 -sha256 -days 730 -noenc \
  -keyout data/keys/server.key -out data/keys/server.crt -subj "/CN=sf-login-bridge"
```

- `server.key` never leaves this server. Treat it like a password.
- `server.crt` is uploaded to Salesforce in the next step.
- The key must be RSA: Salesforce's JWT bearer flow only accepts RS256 signatures.

---

5. Set up Salesforce

5.1 Create the External Client App

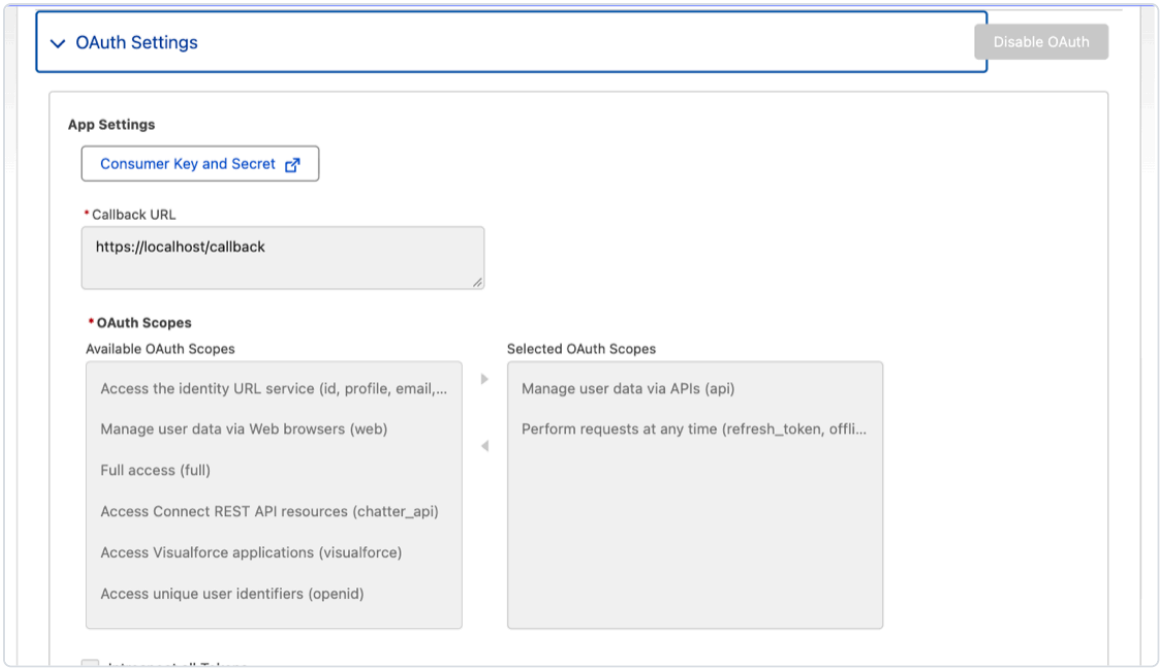
Setup, Quick Find **External Client App Manager**, **New External Client App**. Enter a name (for example *Login Bridge*), a contact email, and Distribution State **Local**.

5.2 OAuth settings

On the app's **Settings** tab, open **OAuth Settings** and enable OAuth:

- **Callback URL:** any https URL, for example `https://localhost/callback`. The bridge does not use it.

- **Selected OAuth Scopes:** *Manage user data via APIs (api)* and *Perform requests at any time (refresh_token, offline_access)*.

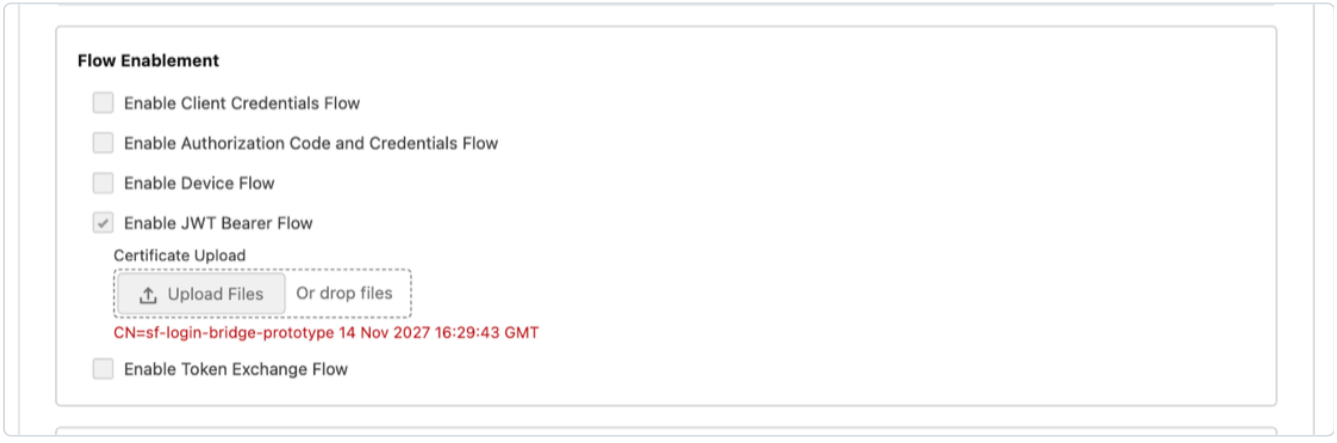


OAuth settings with the callback URL and both scopes selected.

Click **Consumer Key and Secret** and copy the **Consumer Key**. You need it in step 6. The bridge does not need the secret.

5.3 JWT bearer flow

Under **Flow Enablement**, tick **Enable JWT Bearer Flow** and upload `server.crt`. The certificate's name and expiry appear below the upload box.

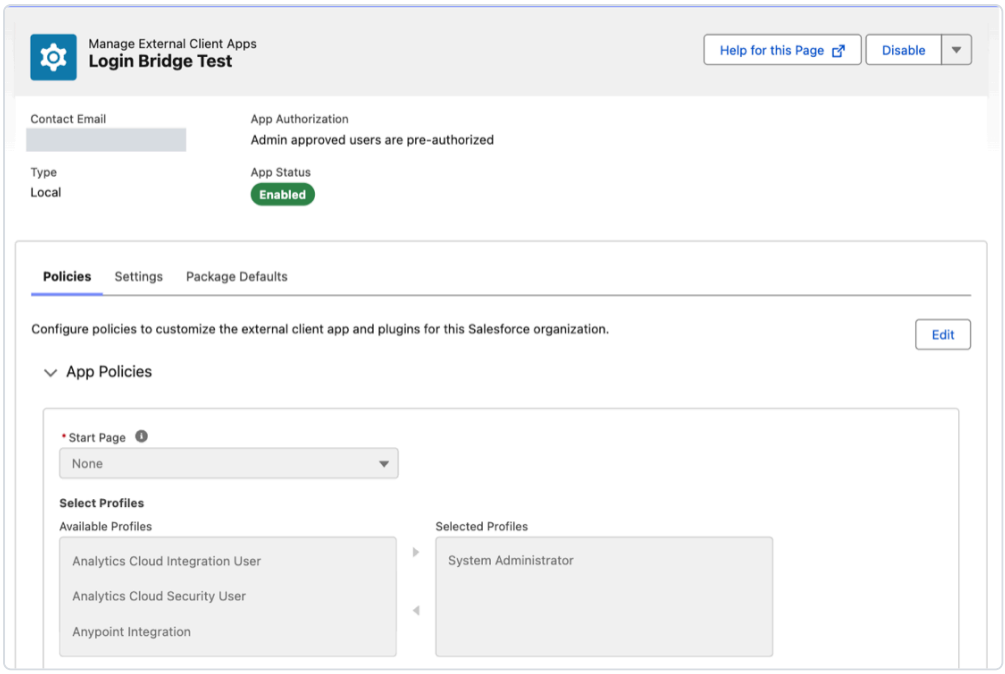


JWT bearer flow enabled, certificate uploaded.

5.4 Policies

On the **Policies** tab, click **Edit**:

- **Permitted Users:** *Admin approved users are pre-authorized*.
- **Select Profiles** or **Select Permission Sets:** add only the integration users. The bridge can get a token for anyone listed here.



Admin approved users pre-authorized, with one profile selected.

> A new app can take up to 10 minutes to become active. Until then, step 8's check reports `invalid_client_id`.

---

6. Configure the bridge

Create `deploy/self-hosted/data/bridge.config.json`:

```
{
  "host": "0.0.0.0",
  "port": 8080,
  "licenseKey": "SFLB2....",
  "salesforce": {
    "loginUrl": "https://yourcompany.my.salesforce.com",
    "audience": "https://login.salesforce.com",
    "clientId": "<consumer key from step 5.2>",
    "privateKeyPath": "./keys/server.key"
  },
  "users": {}
}
```

Field	Value
`loginUrl`	Your My Domain URL: Setup, Quick Find **My Domain**
`audience`	`https://login.salesforce.com`, or `https://test.salesforce.com` for a sandbox
`clientId`	The consumer key
`licenseKey`	Your licence key. Or install it later with `sf-login-bridge set-license '<key>'`, which checks it covers your org
`users`	Leave empty. Step 8 fills it

Unknown fields are rejected, so a typo fails loudly instead of being ignored. The full list of settings is in `docs/operations.md`.

---

7. Provide https

Legacy clients need https; Data Loader refuses anything else. Pick one:

Option	How
Your reverse proxy	Terminate TLS at the proxy, forward to port 8080, and add `"trustProxy": true` so the audit log and lockout see real client addresses. The bridge must then be reachable only through the proxy.
The bridge itself	Put the certificate and key in `data/keys/` and add `"tls": { "certPath": "./keys/tls.crt", "keyPath": "./keys/tls.key" }`

> The bridge only needs to be reachable by the legacy clients. Wrong passwords never reach Salesforce and lock only the address sending them, so exposing it is safe, but there is no reason to.

---

8. Start, add users, verify

```
chmod -R a+rwX data            # the container runs as uid 65532
docker compose up -d
```

Add a bridge user for each legacy login. The bridge password is printed once: put it in the legacy client in place of the Salesforce password.

```
docker compose exec bridge sf-login-bridge add-user integration.user@yourcompany.com
```

If the legacy login name differs from the Salesforce username, map it:

```
docker compose exec bridge sf-login-bridge add-user legacy.job@yourcompany.com \
  --sf-username dataload.integration@yourcompany.com
```

Then check the whole chain: config, licence, and a Salesforce token for every user.

```
docker compose exec bridge sf-login-bridge check
```

```
PASS  config  1 org(s)
PASS  licence  Your Company | expires 2027-12-31 | 1 org(s) | valid, 410 day(s) left
PASS  org default: Salesforce token for integration.user@yourcompany.com  instance https://you
rcompany.my.salesforce.com
```

Finally, sign in the way a legacy client would, then query through the session it gets back:

```
docker compose exec -e CLIENT_PASSWORD='<bridge password>' bridge \
  sf-login-bridge login-client --url http://localhost:8080 --user integration.user@yourcompan
y.com
```

```
PASS  login() at http://localhost:8080/services/Soap/u/40.0  serverUrl https://yourcompany.my.
salesforce.com/...
PASS  query() at https://yourcompany.my.salesforce.com/services/Soap/u/40.0/...  1 row(s)
```

Users added or removed later take effect within a few seconds, without a restart.

---

9. Connect your clients

For every client the change is the same: replace the Salesforce login host with the bridge, and the Salesforce password (and security token) with the bridge password.

Data Loader (batch mode)

In `process-conf.xml`. The first two entries are the ones people miss: without them Data Loader 64 silently logs in to Salesforce directly.

```
<entry key="sfdc.endpoint.Production" value="https://bridge.yourcompany.internal"/>
<entry key="sfdc.oauth.environment" value="Production"/>
<entry key="sfdc.endpoint" value="https://bridge.yourcompany.internal"/>
<entry key="sfdc.username" value="integration.user@yourcompany.com"/>
<entry key="sfdc.password" value="<bridge password, encrypted with Data Loader's encrypt tool
>"/>
```

Clients built from the Partner or Enterprise WSDL

Change the auth endpoint, keeping the path and API version:

```
https://login.salesforce.com/services/Soap/u/40.0   becomes
https://bridge.yourcompany.internal/services/Soap/u/40.0
```

Password grant clients

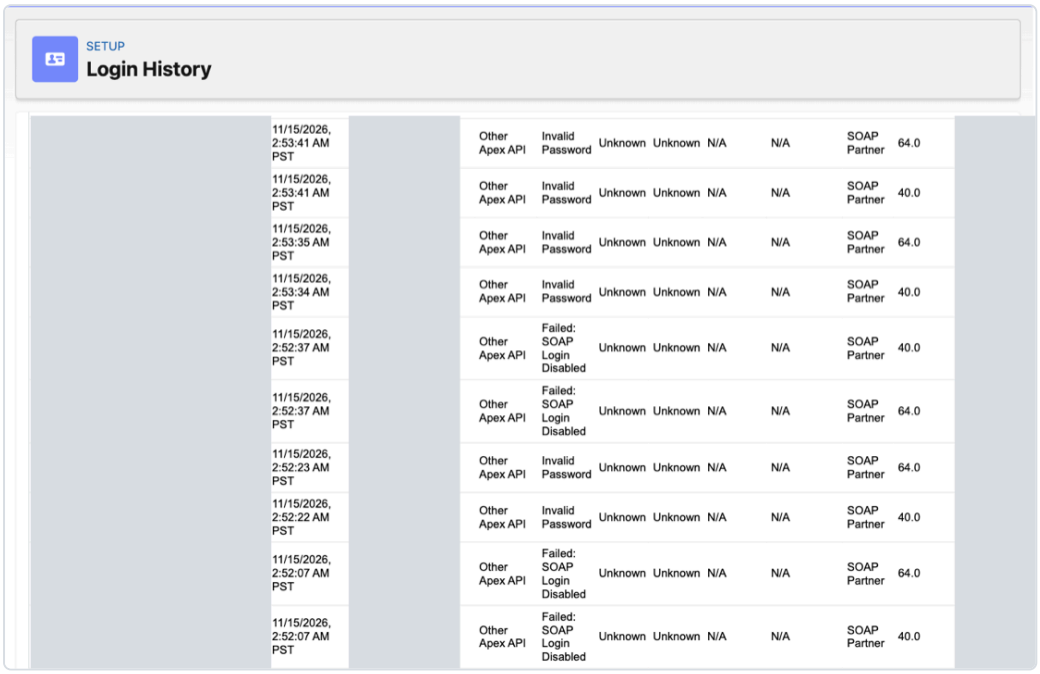
Change the token URL to `https://bridge.yourcompany.internal/services/oauth2/token`.

Client	Status
Data Loader 64.1 batch mode	Verified
Java (Salesforce WSC) on the Partner WSDL	Verified at API 31, 40 and 64
jsforce, restforce, Apache Camel	Login URL accepts any host (from their source code)
.NET WSDL clients, MuleSoft 11, Boomi, Informatica	Expected to work; not yet run against the bridge
simple-salesforce (Python)	Not supported: it only builds `*.salesforce.com` URLs
Microsoft Entra provisioning	Not supported: it only accepts `*.my.salesforce.com`

A tool with `login.salesforce.com` hard coded cannot use the bridge. Per client detail is in `docs/data-loader.md`, `docs/wsdl-client.md` and `docs/password-grant.md`.

10. Go live

1. **Sandbox first.** Run steps 4 to 9 against a sandbox and run each integration's normal job.
2. **Production before 31 January.** Switch while direct `login()` still works, so you can fall back by restoring the old URL and password.
3. **Watch the log** during the first runs: `docker compose logs -f bridge`. Each login shows as one line.
4. **Confirm the retirement** once Salesforce enforces it. A refused direct `login()` returns a generic `INVALID_LOGIN`; Login History records the real reason as **Failed: SOAP Login Disabled**.



Setup, Login History: direct SOAP logins refused with "Failed: SOAP Login Disabled". Usernames and addresses blanked out.

11. Day to day

Run inside `deploy/self-hosted/` as `docker compose exec bridge <command>`:

Task	Command
Add a user or rotate its password	`sf-login-bridge add-user <login name>`
Remove a user	`sf-login-bridge remove-user <login name>`
List users	`sf-login-bridge list-users`
Check config, licence, Salesforce access	`sf-login-bridge check`
Export login attempts as CSV	`sf-login-bridge audit-export --since 2027-02-01 --out /data/logins.csv`
Show the licence	`sf-login-bridge license-show`

- **Lockout.** Ten wrong passwords from one address in 15 minutes lock that address for 15 minutes. Users are never locked, so nobody can lock your integration out from elsewhere.
- **Audit log.** `data/logs/audit-YYYY-MM-DD.jsonl`, one line per request. No passwords, tokens or record data. Archive old files to suit your retention policy.

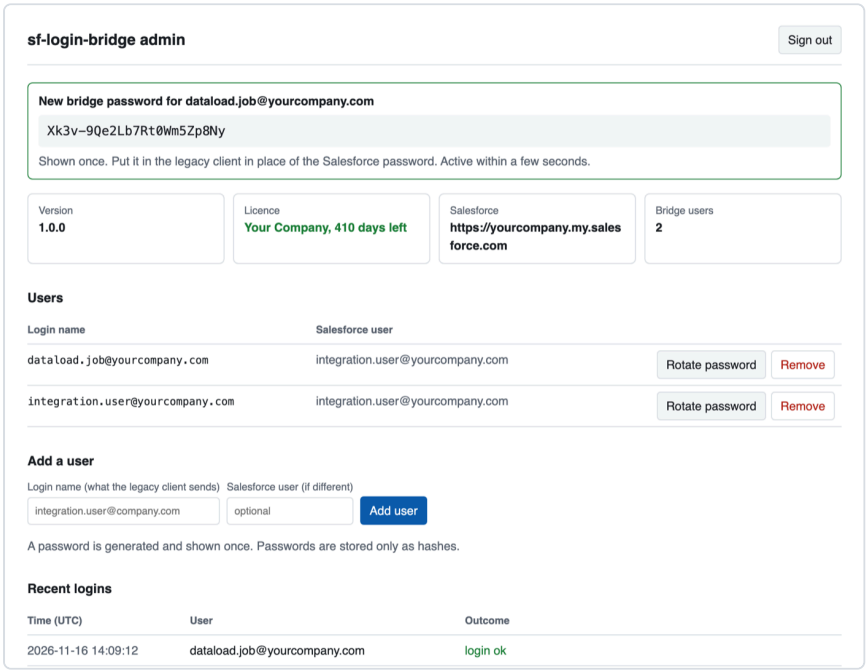
- **Licence.** Renewals are fetched automatically twice a day. Warnings start 30 days before expiry; logins stop 14 days after it. Moving from the sandbox licence to production: `sf-login-bridge set-license '<key>'`, no restart needed.
- **Old API versions.** Logins on API 31 to 40 are flagged in the log and the admin page: Salesforce retires those versions in Summer '28.
- **Revoking access.** Remove the certificate or disable the app in Salesforce. The bridge stops getting tokens immediately.
- **Upgrades.** Pull the new version, rebuild the image, `docker compose up -d`. Config and logs in `data/` are kept.

Admin page

The same user tasks are available on a one page admin UI, with licence status and recent logins. It is off until you set an admin password:

```
docker compose exec bridge sf-login-bridge set-admin-password
```

Add `"admin": { "host": "0.0.0.0" }` next to the generated password hash in the config, restart with `docker compose restart`, and open `http://127.0.0.1:8081` on the server. The compose file publishes it on the server's loopback only; reach it remotely through an SSH tunnel.



The admin page right after adding a user: the new bridge password is shown once.

12. Troubleshooting

Symptom	Cause and fix
`check` : `invalid_client_id`	The app is not active yet, or `clientId` is wrong. Wait 10 minutes, then compare with the consumer key.
`check` : `user hasn't approved this consumer`	The user's profile or permission set is not in the app's policies (step 5.4).
`check` : `invalid_grant` about audience or signature	Wrong `audience` for the org type, or the uploaded certificate is not this server's `server.crt` .
Client gets `INVALID_LOGIN`	Wrong bridge password, unknown user, or a locked address. The log line says which.
Data Loader: *only supports ... https protocol*	The endpoint is `http://` . Use https (step 7).
Data Loader: `LoginFault` , nothing in the bridge log	`sfdc.endpoint.Production` is missing, so Data Loader went to Salesforce directly.
Client: `PKIX path building failed`	Java does not trust the bridge's certificate. Use a certificate from a CA your clients trust.
Two jobs fail together after one finishes	One called `logout()` on the shared session. Give each concurrent job its own Salesforce user.
Log: `non-login call ... sent to the bridge`	The client ignores `serverUrl` and sends data calls to the login host. It cannot use a login bridge.

More detail: `README.md` and `docs/` in the repository.