



Quickstart Installation

Canton Network Quickstart Guide | 2025

Version: 1.0.0-2025-03-14

[Contents](#)

[Canton Network Quickstart Installation](#)

[Introduction](#)

[Overview](#)

[Prerequisites](#)

[Nix Download support](#)

[Step-by-step Instructions](#)

[Clone From Github](#)

[Artifactory](#)

[Docker](#)

[Install Daml SDK](#)

[Deploy a Validator on LocalNet](#)

[Closing the Application](#)

[Close Canton Console](#)

[Close Daml Shell](#)

[Close the CN-QS](#)

[Next Steps](#)

[Resources](#)

Canton Network Quickstart Installation

Introduction

The QS is designed to help teams become familiar with CN application development by providing scaffolding to kickstart development. The QS application is intended to be incrementally extended by you to meet your specific business needs. Once you are familiar with the QS, please review the technology choices and the application design to determine what changes are needed - technology and design decisions are ultimately up to you. Please be aware that the Canton Network Quickstart (CN-QS) is a rapidly evolving work in progress.

Overview

The CN-QS and its guides are a work-in-progress (WIP). As a result, the CN-QS guides may not accurately reflect the state of the application. If you find errors or other inconsistencies, please contact your representative at Digital Asset.

This guide walks through the installation and LocalNet deployment of the CN-QS.

Prerequisites

Access to the [CN-Quickstart Github repository](#) and [CN Docker repository](#) is needed to successfully pull the Digital Asset artifacts from JFrog Artifactory.

Access to the *Daml-VPN* connection or [a SV Node](#) that is whitelisted on the CN is required to connect to DevNet. The GSF publishes a [list of SV nodes](#) who have the ability to sponsor a Validator node. To access DevNet, contact your sponsoring SV agent for VPN connection information.

If you need access or additional support, email support@digitalasset.com.

The CN-QS is a Dockerized application and requires [Docker Desktop](#). Running CN-QS is resource intensive. We recommend allocating 24 GB of memory to Docker Desktop. If your machine does not have that much memory consider declining Observability when prompted.

Other requirements include:

- [Curl](#)
- [Direnv](#)
- [Nix](#)
- Windows users must install and use [WSL 2](#) with administrator privileges

Nix Download support

Check for Nix on your machine.

```
nix --version
```

If the command returns something like:

```
Nix (Nix) 2.25.2
```

Congratulations, you're done.

Recommended installation for MacOS.

```
sh <(curl -L https://nixos.org/nix/install)
```

Recommended installation for Linux.

(Windows users should run this and all following commands in WSL 2).

```
sh <(curl -L https://nixos.org/nix/install) --daemon
```

Step-by-step Instructions

Clone From Github

Clone and cd into the `cn-quickstart` repository into your local machine.

```
git clone https://github.com/digital-asset/cn-quickstart.git
cd cn-quickstart
direnv allow
```

```
(base) cn-quickstart ~ % direnv allow
direnv: loading ~/Projects/daml/cn-quickstart/.envrc
direnv: using nix
direnv: export +AR +AS +CC +CONFIG_SHELL +CXX +HOST_PATH +IN_NIX_SHELL +LD +LD_DYLD_PATH +MACOSX_DEPLOYMENT_TARGET +NIX_BINTOOLS +NIX_BINTOOLS_WRAPPER_TARGET_HOST_aarch64_apple_darwin +NIX_BUILD_CORES +NIX_BUILD_TOP +NIX_CC [+NIX_CC_WRAPPER_TARGET_HOST_aarch64_apple_darwin +NIX_CFLAGS_COMPILE +NIX_DONT_SET_RPATH +NIX_DONT_SET_RPATH_FOR_BUILD +NIX_ENFORCE_NO_NATIVE +NIX_HARDENING_ENABLE +NIX_IGNORE_LD_THROUGH_GCC +NIX_LDFLAGS +NIX_LDFLAGS_FOR_BUILD +NIX_NO_SELF_RPATH +NIX_STORE +NM +NODE_PATH +PATH_LOCALE +RANLIB +SIZE +SOURCE_DATE_EPOCH +STRINGS +STRIP +TEMP +TMPDIR +TMP +ZERO_AR_DATE +__CF_USER_TEXT_ENCODING +__darwinAllowLocalNetworking +__impureHostDeps +__propagatedImpureHostDeps +__propagatedSandboxProfile +__sandboxProfile +__structuredAttrs +buildInputs +buildPhase +builder +cmakeFlags +configureFlags +depsBuildBuild +depsBuildBuildPropagated +depsBuildTarget +depsBuildTargetPropagated +depsHostHost +depsHostHostPropagated +depsTargetTarget +depsTargetTargetPropagated +doCheck +doInstall +lCheck +mesonFlags +name +nativeBuildInputs +out +outputs +patches +phases +preferLocalBuild +propagatedBuildInputs +propagatedNativeBuildInputs +shell +shellHook +stdenv +strictDeps +system ~JAVA_HOME ~PATH ~TMPDIR ~XDG_DATA_DIRS
```

Artifactory

Necessary artifacts are located in Digital Artifact's JFrog Artifactory. These files are accessed through the repository's build system using a `~/.netrc` configuration file.

Check if a `~/.netrc` file already exists.

```
cat ~/.netrc
```

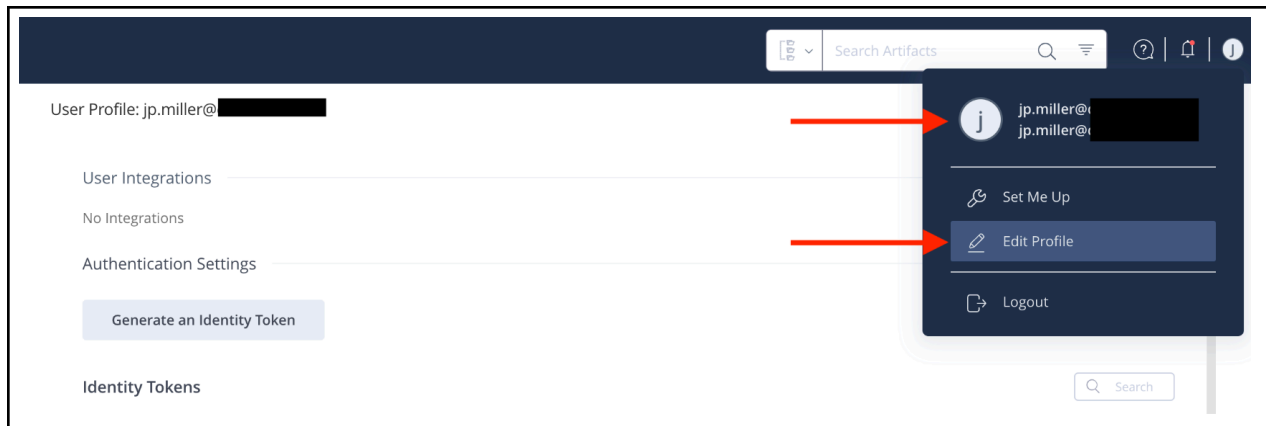
Create or edit the `~/.netrc` file at root.

```
vim ~/.netrc
```

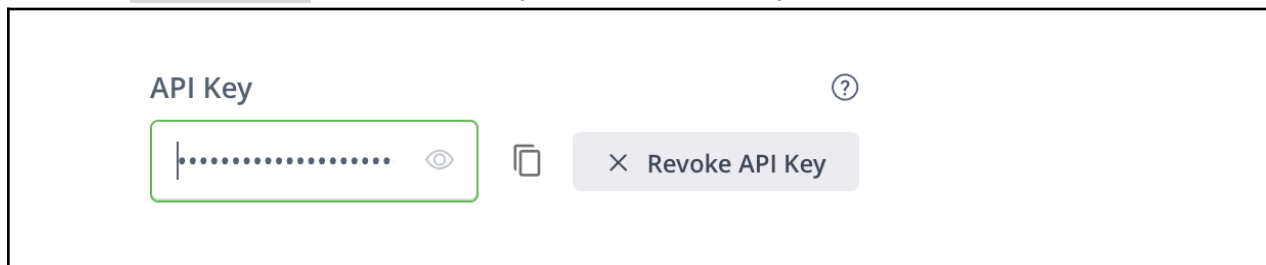
Add the Artifactory's login and password.

```
machine digitalasset.jfrog.io
login <username>
password <password>
```

Replace `<username>` with the JFrog Artifactory user profile email.



Replace `<password>` with the API Key. Create an API Key if none exists.



The `~/.netrc` configuration file should look something like:

```
machine digitalasset.jfrog.io
login email@domain.com
password plain_text_api_key_or_password
```

Manually set `.netrc`'s correct permissions.

```
chmod 600 ~/.netrc
```

Check for Artifactory connectivity using `.netrc` credentials after populating the username and password.

```
curl -v --netrc
"https://digitalasset.jfrog.io/artifactory/api/system/ping"
```

```
* Request completely sent off
< HTTP/1.1 200
< Date: Wed, 12 Feb 2025 21:33:55 GMT
< Content-Type: text/plain
< Transfer-Encoding: chunked
< Connection: keep-alive
< X-JFrog-Version: Artifactory/7.106.3 80603900
< X-Artifactory-Id: 7ab03617d6964dd54ed8546c4ec24a4023554fd6
< X-Artifactory-Node-Id: digitalasset-artifactory-primary-2
< Strict-Transport-Security: max-age=31536000; includeSubDomains
< X-Request-ID: e2c3f32a82407565f6bec2ef8c74b367:e2c3f32a82407565f
<
* Connection #0 to host digitalasset.jfrog.io left intact
OK
```

A response of "OK" indicates a successful connection.

Authentication problems often result in a 401 or 403 error. If an error response occurs, double check `~/.netrc` and be sure that `.netrc` is a source file (in root) and not a local file.

Docker

Be sure that Docker Desktop is running.

Login to Docker repositories via the terminal.

```
docker login digitalasset-docker.jfrog.io
```

```
docker login digitalasset-canton-network-docker.jfrog.io
```

```
docker login
```

The last command requires a [Docker Hub](#) username and password or *Personal Access Token* (PAT). Commands should return 'Login Succeeded'.

Install Daml SDK

cd into the `quickstart` subdirectory and install the Daml SDK from the quickstart subdirectory.

```
cd quickstart
make install-daml-sdk
```

The makefile providing project choreography is in the `quickstart/` directory. make only operates within `quickstart/`. If you see errors related to make, double check your present working directory.

The Daml SDK is large and can take several minutes to complete.

```
(base) quickstart ~ % make install-daml-sdk

> Task :daml:fetchDamlSdk
Download https://digitalasset.jfrog.io/artifactory/assembly/daml/3.2.0-snapshot.20241031.13398.0.vf95d2607/daml-sdk-3.2.0-snapshot.20241031.13398.0.vf95d2607-macos.tar.gz
<-----> 0% EXECUTING [1m 30s]
> :daml:fetchDamlSdk > daml-sdk-3.2.0-snapshot.20241031.13398.0.vf95d2607-macos.tar.gz > 99.68 MB/520.58 MB downloaded

> Task :daml:unpackDamlSdk
Unpacked SDK archive to /Users/jpmiller/Projects/daml/cn-quickstart/quickstart/daml/.sdk/extracted

> Task :daml:installDamlSdk
Installing SDK release from directory.
Installed Daml SDK runtime 3.2.0-snapshot.20241031.13398.0.vf95d2607 as 3.2.0-snapshot.20241106.0
Cleaning up downloaded files

BUILD SUCCESSFUL in 7m 32s
13 actionable tasks: 4 executed, 9 up-to-date
(base) quickstart ~ %
```

Deploy a Validator on LocalNet

From the quickstart subdirectory, build the application.

```
make build
```

```
BUILD SUCCESSFUL in 5s
22 actionable tasks: 2 executed, 20 up-to-date
docker compose -f compose.yaml --env-file .env -f docker/o11y/cadvisor-darwin.yaml -f
[+] Building 1.9s (15/15) FINISHED
=> [await-onboarding-done internal] load build definition from Dockerfile
=> => transferring dockerfile: 248B
```

```
=> CACHED [backend-service 2/2] RUN apt-get update && apt-get install -y jq curl bash
=> [backend-service] exporting to image
=> => exporting layers
=> => writing image sha256:dad288f32e529b879f2d2f07190f0e8660459ee59510ac75e13d2f13f6f8acff
=> => naming to docker.io/library/quickstart-backend-service
=> [backend-service] resolving provenance for metadata file
[+] Building 2/2
✓ await-onboarding-done Built
✓ backend-service Built
```

Once complete, start the application, Canton services and Observability.

```
make start
```

The first time running `make start`, a helper assistant prompts to set up a local deployment. It offers the choice of running `DevNet` or `LocalNet`, enabling `Observability`, and specifying a party hint. In the future, this helper can be accessed by running `make setup`.

Begin the first application in `LocalNet` with `Observability` enabled. Leave the party hint blank to use the default.

The party hint is used as a party node's alias of their identification hash. The Party Hint is not part of the user's identity. It is a convenience feature. It is possible to have multiple party nodes with the same hint.

```
Enable LocalNet? (Y/n): Y
```

```
LOCALNET_ENABLED set to 'true'.
```

```
Enable Observability? (Y/n): Y
```

```
OBSERVABILITY_ENABLED set to 'true'.
```

```
Specify a party hint (this will identify the participant in the network)
```

```
[quickstart-USERNAME-1]:
```

```
PARTY_HINT set to 'quickstart-USERNAME-1'.
```

```
.env.local updated successfully.
```

Consider declining `Observability` if your machine has less than 24 GB of memory to allocate to Docker Desktop.


```

Enable LocalNet? (Y/n): Y
  LOCALNET_ENABLED set to 'true'.

Enable Observability? (Y/n): Y
  OBSERVABILITY_ENABLED set to 'true'.

Specify a party hint (this will identify the participant in the network) [quickstart-USERNAME-1]:
  PARTY_HINT set to 'quickstart-USERNAME-1'.

.env.local updated successfully.
Environment updated. Please re-run make start.
(base) quickstart ~ % make start

```

If prompted to re-run `make start`, do so.

`make start`

```

BUILD SUCCESSFUL in 2s
21 actionable tasks: 2 executed, 19 up-to-date
docker compose -f compose.yaml --env-file .env --profile localnet --env-file
[+] Building 0.0s (0/0)
[+] Running 32/32
  ✓ Network quickstart_splice-sv-public      Created
  ✓ Network quickstart_splice-sv-private     Created
  ✓ Network quickstart                       Created
  ✓ Container scan-web-ui                   Started
  ✓ Container postgres-splice-metrics        Started
  ✓ Container nginx-metrics                 Started
  ✓ Container nginx-sv-metrics               Started
  ✓ Container cadvisor                      Started
  ✓ Container loki                          Started
  ✓ Container wallet-web-ui                 Started
  ✓ Container postgres-splice                Healthy
  ✓ Container postgres-splice-sv-metrics     Started
  ✓ Container postgres-splice-sv            Healthy
  ✓ Container otel-collector                 Started
  ✓ Container prometheus                    Started
  ✓ Container sv-web-ui                     Started
  ✓ Container oauth                         Started
  ✓ Container tempo                         Started
  ✓ Container grafana                       Started
  ✓ Container domain                        Healthy
  ✓ Container sv-app                        Healthy
  ✓ Container participant-sv                 Healthy
  ✓ Container scan                          Started
  ✓ Container domain-global                  Healthy
  ✓ Container validator-sv                   Started
  ✓ Container participant                    Healthy
  ✓ Container validator                      Started
  ✓ Container nginx-sv                      Started
  ✓ Container await-parties-allocated        Healthy
  ✓ Container pqs                           Started
  ✓ Container backend-service                Started
  ✓ Container nginx                         Started
(base) quickstart ~ %

```

`make start` initiates the LocalNet containers, which can be computationally demanding. If you see unhealthy containers or error containers on `make start` try to increase RAM access to Docker to at least 24 GB.

In the future, run the following series of commands from `cn-quickstart/` to update and initiate Quickstart:

```
git pull; cd quickstart; make install-daml-sdk; make setup; make build; make start
```

In a separate shell, from the quickstart subdirectory, run the Canton Consoles.

```
make console-app-provider
```

```
make console-app-user
```

```
17:50:48,153 |-INFO in ch.qos.logback.classic.joran.JoranConfigurator@3b8f0a79 -
17:50:48,153 |-INFO in ch.qos.logback.classic.util.ContextInitializer@14f9390f -
NEXT_IF_ANY

Compiling (synthetic)/ammonite/predef/ArgsPredef.sc
Compiling /app/(console)

Canton

Welcome to Canton!
Type 'help' to get started. 'exit' to leave.

@
```

In a third shell, from the quickstart subdirectory, begin the Daml Shell.

```
make shell
```

```
(base) quickstart ~ % make shell
[+] Building 0.0s (0/0)
docker:desktop-linux
[+] Building 0.0s (0/0)
docker:desktop-linux
Connecting to jdbc:postgresql://postgres-splice:5432/scribe...
Connected to jdbc:postgresql://postgres-splice:5432/scribe
WARNING: The connected database has a newer schema version (021) than the supported range.
011-013, Scribe version range: 0.4.0-0.4.7.
postgres-splice:5432/scribe 5b → 5b> 7
connected Session range: 5b → 5b Datastore range: 5b → 5b
```

Closing the Application

 (If you plan on immediately using the CN-QS then delay execution of this section)

Close Canton Console

When complete, open the Canton console terminal. Run `exit` to stop and remove the console container.

Close Daml Shell

In the Daml Shell terminal, execute `quit` to stop the Shell container.

Close the CN-QS

Finally, close the application and observability services with:

```
make stop && make clean-all
```

It is wise to run `make clean-all` during development and at the end of each session to avoid conflict errors on subsequent application builds.

Next Steps

You have successfully installed the CN-QS. The next section, “Exploring The Demo,” provides a demonstration of the application in `LocalNet` and `DevNet` environments.

Resources

[Curl](#)

[Direnv](#)

[Docker Desktop](#)

[Docker Hub](#)

[GSF List of SV Nodes](#)

[JFrog CN Artifactory](#)

[Nix](#)

[Quickstart GitHub Repository](#)

[Validator Onboarding Documentation](#)

[WSL 2](#)