Quickstart Installation

Canton Network Quickstart Guide | 2025

Version: 1.0.0-2025-05-16

Contents

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 8 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_TARG
ET +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_BUI
LD +NIX_NO_SELF_RPATH +NIX_STORE +NM +NODE_PATH +PATH_LOCALE +RANLIB +SIZE +SOURCE_DATE_EPOCH +STRINGS +STRIP +T
EMP +TEMPDIR +TMP +ZERO_AR_DATE +__CF_USER_TEXT_ENCODING +__darwinAllowLocalNetworking +__impureHostDeps +__prop
agatedImpureHostDeps +__propagatedSandboxProfile +__sandboxProfile +__structuredAttrs +buildInputs +buildPhase +
builder +cmakeFlags +configureFlags +depsBuildBuild +depsBuildBuildPropagated +depsBuildTarget +depsBuildTargetP
ropagated +depsHostHost +depsHostHostPropagated +depsTargetTarget +depsTargetTargetPropagated +doCheck +doInstal
lCheck +mesonFlags +name +nativeBuildInputs +out +outputs +patches +phases +preferLocalBuild +propagatedBuildInp
uts +propagatedNativeBuildInputs +shell +shellHook +stdenv +strictDeps +system ~JAVA_HOME ~PATH ~TMPDIR ~XDG_DAT
A_DIRS
```

Artifactory

Check the ~/.netrc file

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
```

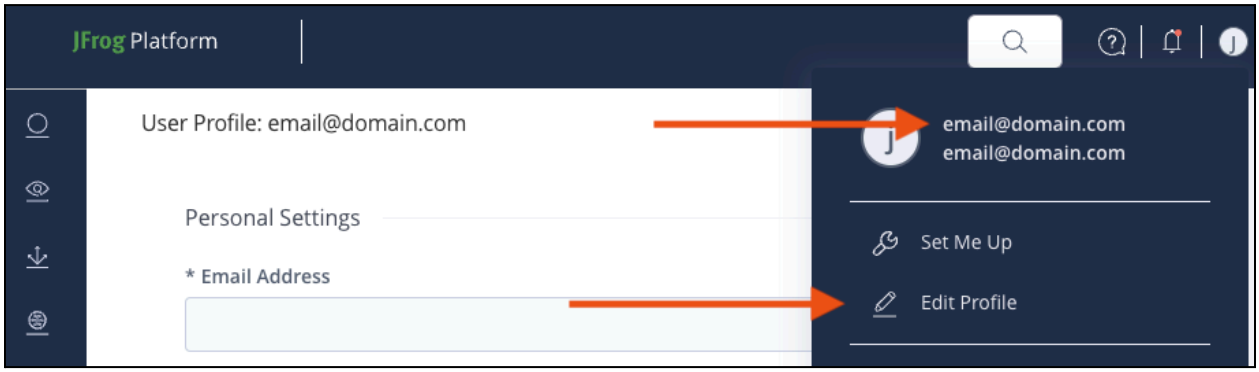
Paste the boiler plate content into ~/.netrc.

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

Locate login for ~/.netrc

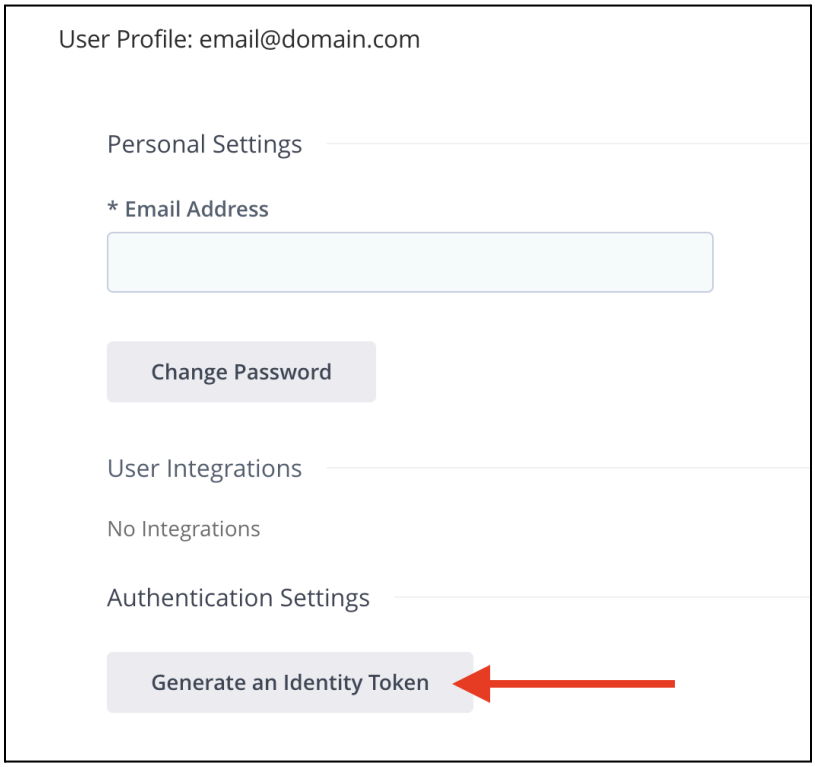
Log into the JFrog and click the profile icon in the top right.

Your email address is the login value. Replace `<username>` with the JFrog Artifactory user profile email.

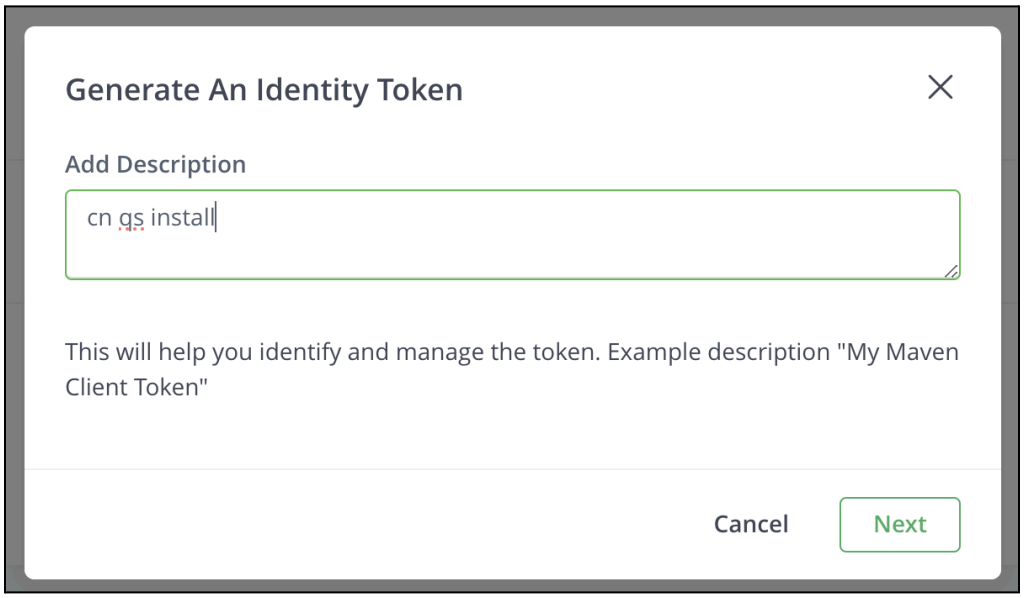


Create an API key

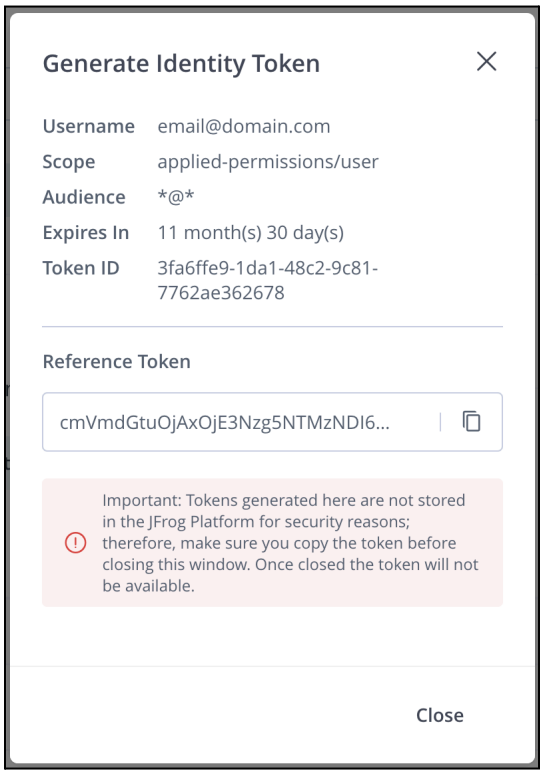
Toward the bottom of the same profile page, click "Generate an Identity Token".



Add an identity token description.



Copy the Reference token. This token is also referred to as “the API key” and is the “`<password>`” in `~/.netrc`.



Complete ~/.netrc

The API key is stored as "`password`" in `~/.netrc`. Replace <password> with the API Key from the JFrog profile. When complete, the `~/.netrc` file is complete it will look like:

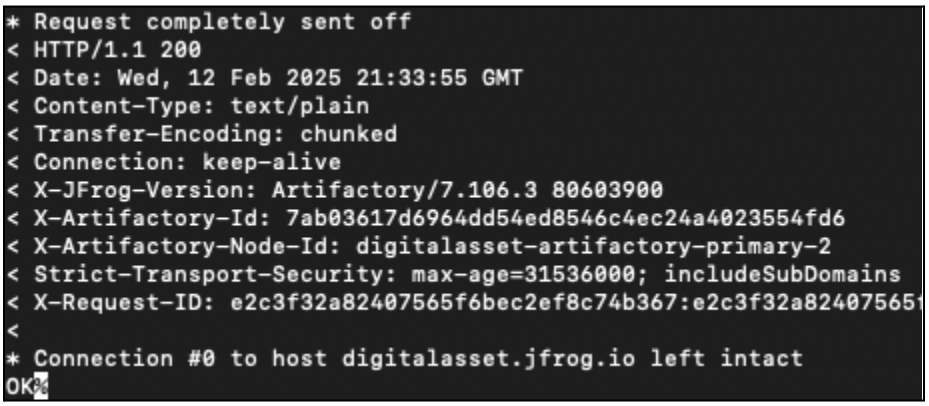
```
machine digitalasset.jfrog.io
login email@domain.com
password cmVmdGtuOjAxOjE3Nzg5NTQzNjc6UmhYaFNaZWpUNGtFMzJyYXRyWEQya...
```

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"
```



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` to confirm that `.netrc` is a source file (in root) and not a local file.

Docker

Verify 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% 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
```

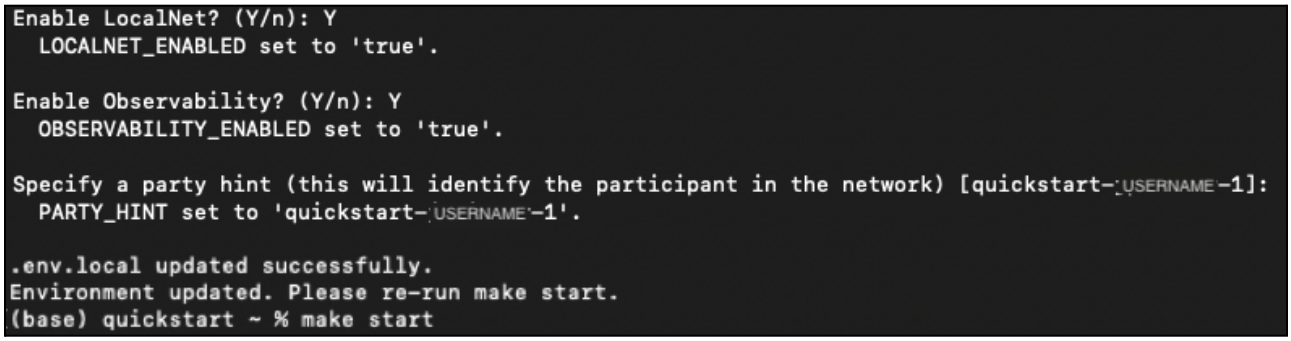
The first time running `make start`, a helper assistant prompts to set up a local deployment. It offers the choice of running `DevNet` or `LocalNet`, enabling `Observability`, and specifying a party hint. In the future, this helper can be accessed by running `make setup`.

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.

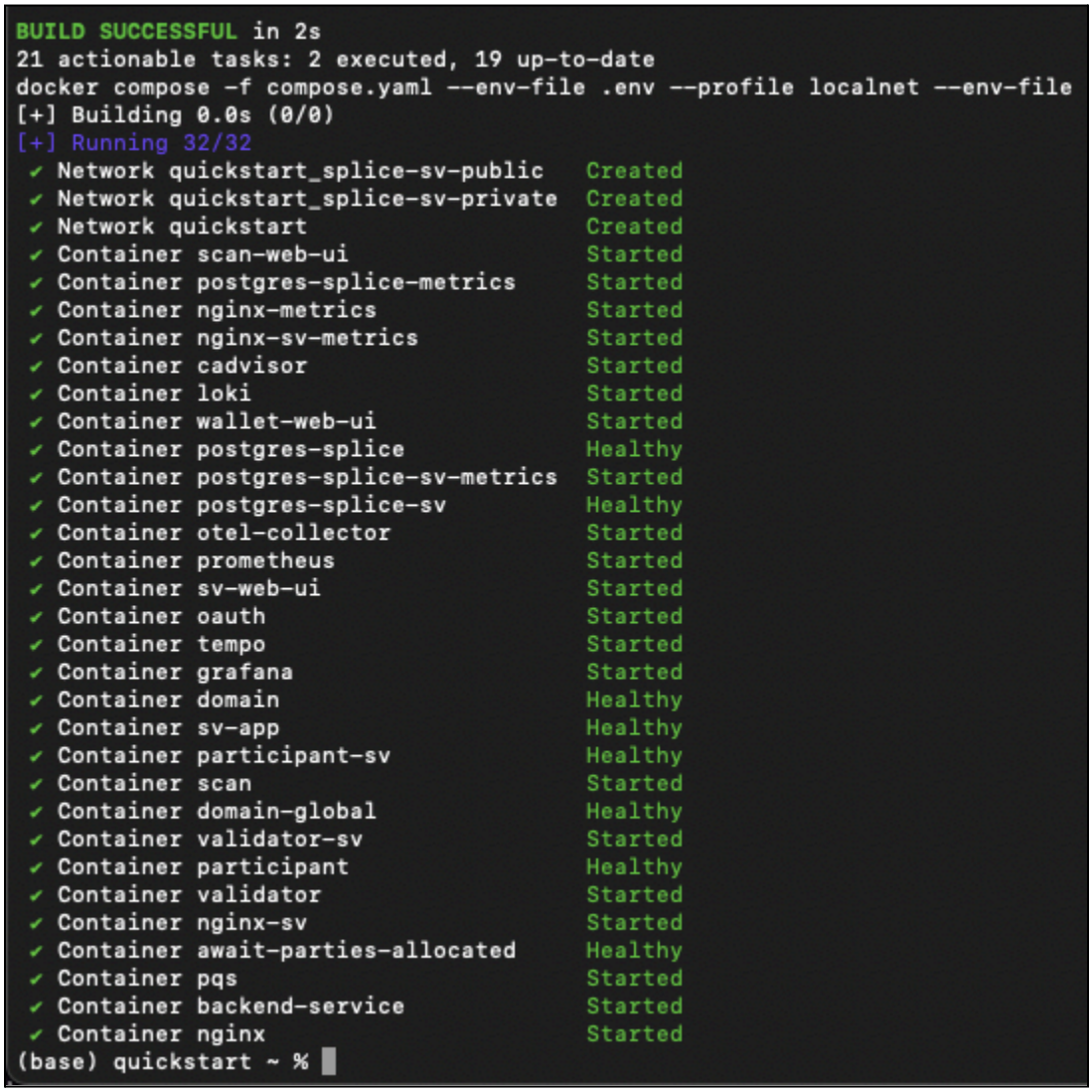
```
Enable LocalNet? (Y/n): Y
  LOCALNET_ENABLED set to 'true'.

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.



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

```
make start
```



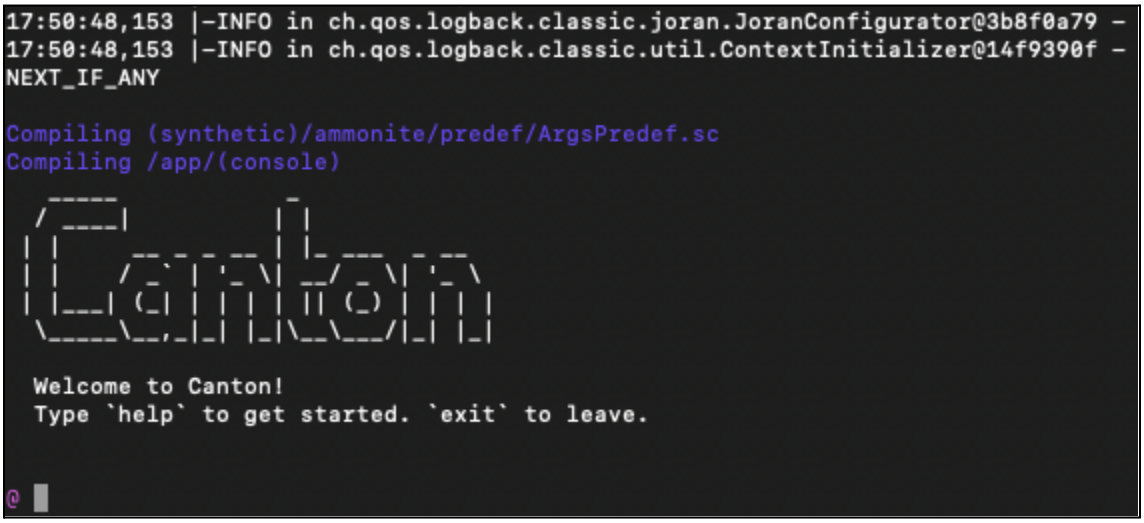
In the future, you may run the following series of commands from `cn-quickstart/` to clone and initiate Quickstart:

```
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.

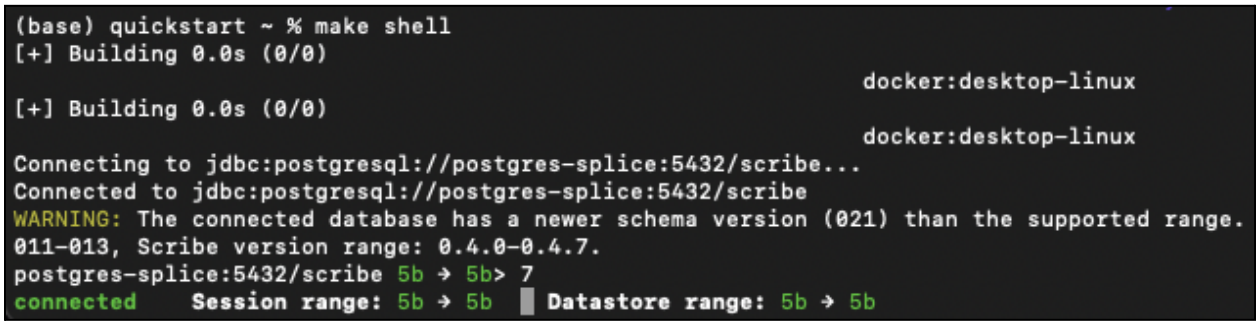
```
make console-app-provider
```

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



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

```
make shell
```



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