LOCKSS System Manual

LOCKSS Program

Sep 28, 2023

CONTENTS

Released: 2019-04-05. Last modified: 2021-04-12.

At the 2019 LOCKSS Alliance Meeting, the LOCKSS engineering team presented a Technology Preview of the upcoming LOCKSS 2.0-alpha1 release. **The LOCKSS 2.0-alpha Technology Preview and the upcoming LOCKSS 2.0-alpha1 release are for evaluation and testing purposes.**

The LOCKSS 2.0-alpha1 release will be the first public release of the next generation of LOCKSS software. It represents a major evolution of the classic LOCKSS daemon (1.x) into a suite of specialized software components. These components will interoperate via REST APIs and run as Docker components as part of a Docker stack.

Unlike the production LOCKSS 2.0 release, the LOCKSS 2.0-alpha Technology Preview consists of a fixed set of LOCKSS components, and the supporting components are provided and managed by the LOCKSS system:

- Provided Postgres database container
- Provided Solr database container
- Provided HDFS (Hadoop File System) container
- LOCKSS Repository Service
- LOCKSS Configuration Service
- LOCKSS Metadata Extraction Service
- LOCKSS Metadata Service
- LOCKSS Poller Service
- Provided Pywb container

CHAPTER

ONE

OVERVIEW AND QUICK START

To run the LOCKSS 2.0-alpha Technology Preview, you will need a Linux host with Docker running in Swarm mode with the Local-Persist Docker volume plugin installed.

You will also need to install a couple of Python modules and download a small project from GitHub.

Quick Start:

- Install Docker (18.09 or better, in Swarm mode), the *Local-Persist* Docker volume plugin, the setuptools and *Pystache* Python modules, and *Git*.
- `git clone --branch develop https://github.com/lockss/lockss-installer`
- `cd lockss-installer`
- `scripts/configure-lockss`
- `scripts/assemble-lockss`
- `scripts/deploy-lockss`
- Use the system
- Shut down the system with `script/shutdown-lockss`

CHAPTER

TWO

SYSTEM PRE-REQUISITES

Machine and operating system:

- 64-bit Linux flavor running `systemd`
- 4 cores (8 or more recommended)
- 8GB of memory (16GB or more recommended)

Disk space:

- A few gigabytes of storage space to process a dozen archival units as part of the provided demo network (a few LOCKSS plugins for open access journals)
- Proportionately more storage space if you process more archival units

CHAPTER

THREE

SOFTWARE PRE-REQUISITES

The software pre-requisites are:

- Docker (18.09 or better)
- *Local-Persist* (see below)
- *Pystache* (see below)
- *Git* (see below)

3.1 Local-Persist

Local-Persist is a plugin for Docker.

- *(recommended)* Direct installation with Curl: `curl -fsSL https://raw.githubusercontent.com/CWSpear/local-persist/master/scripts/install.sh | sudo bash`
- Direct installation with Wget: `wget -qO - https://raw.githubusercontent.com/CWSpear/local-persist/master/scripts/install.sh | sudo bash`
- Manual installation: https://github.com/CWSpear/local-persist#manual-way

To verify that the Local-Persist Docker plugin is working, run `docker info` and check that the `Volume` category under `Plugins` contains `local-persist`.

3.2 Pystache

Pystache is a Python implementation of the Mustache templating language.

- Via pip: `sudo pip install pystache`
- Via a system package:
  - Arch Linux: `python-pystache` from the AUR (https://aur.archlinux.org/packages/python-pystache/)
  - CentOS 7: `sudo yum install pystache`
  - Debian 9 (Stretch) or better: `sudo apt-get install python-pystache`
  - Fedora 28 or better: `sudo yum install python2-pystache`
  - Ubuntu 16.04 LTS (Xenial) or better: `sudo apt-get install python-pystache`

To verify that Pystache is installed, run `pystache --help` and check that you get a help message rather than an error message. Alternatively, run `pystache-test` to run Pystache's internal test suite.

3.3 Git

Git is a version control system.

If Git is not installed already (run `git --help` and see if you get a help message or an error message), it can be installed via a system package:

- Arch Linux: `sudo pacman -S git`
- CentOS 7: `sudo apt-get install git`
- Debian 9 (Stretch) or better: `sudo yum install git`
- Fedora 28 or better: `sudo yum install git`
- Ubuntu 16.04 LTS (Xenial) or better: `sudo apt-get install git`

CHAPTER

FOUR

INSTALLING THE LOCKSS INSTALLER

Clone the `develop` branch of the `lockss-installer` GitHub project:

```
git clone --branch develop https://github.com/lockss/lockss-installer
```

and change into the newly created directory:

```
cd lockss-installer
```

In this document, this directory is referred to as `${INSTALLER_HOME}`.

CHAPTER FIVE

CONFIGURING THE SYSTEM

To configure the system in `${INSTALLER_HOME}`, run:

```
scripts/configure-lockss
```

The gist of this script will be familiar to users of the classic LOCKSS daemon (1.x) installation script `hostconfig`.

The questions asked by the script often come with a suggested value, displayed in square brackets; hit Enter to accept the suggested value, or type the correct value and hit Enter. Questions include:

1. `Fully qualified hostname (FQDN) of this machine:` Enter the machine's hostname (e.g. `locksstest.myuniversity.edu`).
2. `IP address of this machine:` The publicly routable IP address of the machine, or if it is not publicly routable but will be accessible via network address translation (NAT), its IP address on the internal network.
3. `Is this machine behind NAT?:` Enter `Y` if the machine is not publicly routable but will be accessible via network address translation (NAT), or `N` otherwise.
    1. `External IP address for NAT:` If you answered `Y` to the previous question, enter the publicly routable IP address of the machine.
4. `Initial subnet for admin UI access:` Enter a semicolon-separated list of subnets in CIDR notation that should have access to the Web user interfaces of the system.
5. `LCAP V3 protocol port:` Enter the port on the publicly routable IP address that will be used to receive LCAP (LOCKSS polling and repair) traffic. Historically, most LOCKSS nodes use 9729.
6. `PROXY port:` Not yet re-enabled in 2.0-alpha; ignore.
7. `Mail relay for this machine:` Hostname for this machine's outgoing mail server.
8. `Does mail relay <mailhost> need user & password`: Enter `Y` if the outgoing mail server requires password authentication, `N` otherwise.
    1. `User for <mailhost>:` If you answered `Y` to the outgoing mail server password authentication question, enter the username for the mail server.
    2. `Password for <mailuser>@<mailhost>:` Enter the password for the given username.
    3. `Again:` Re-enter the mail server password (if the two passwords do not match, the password will be asked again).
9. `E-mail address for administrator:` Enter the e-mail address of the person or team who will administer the LOCKSS system on this machine.
10. `Configuration URL:` Enter the URL of the LOCKSS network configuration file. If you are not running your own LOCKSS network, use `http://props.lockss.org:8001/demo/lockss.xml`, the configuration file for a demo network set up for LOCKSS 2.0 pre-release testing.

11. `Configuration proxy (host:port):` enter a host:port combination for the proxy server needed to reach the network configuration file (or simply hit Enter if none is needed).
12. `Preservation group(s):` Enter a semicolon-separated list of preservation network identifiers. If you are not joining an existing network or running your own, enter demo, the network identifier for the demo network set up for LOCKSS 2.0 pre-release testing.
13. `Content data storage directory:` Enter the path of a directory that is the root of the main storage area of the LOCKSS system. If you are used to the classic LOCKSS daemon (1.x), this would be the equivalent of `/cache0`.
14. `Service logs directory:` Enter the path of a directory that is the root of the storage area for LOCKSS-related log files (historically `/var/log/lockss`).
15. `Temporary storage directory:` not actively used in LOCKSS 2.0-alpha; ignore.
16. `User name for web UI administration:` Enter the username for an administrative user in the LOCKSS system's Web user interfaces.
17. `Password for web UI administration user <uiuser>:` Enter the password for the given administrative user in the LOCKSS system's Web user interfaces.
18. `Password for web UI administration user <uiuser> (again):` Re-enter the password for the given administrative user in the LOCKSS system's Web user interfaces (if the two passwords do not match, the password will be asked again).
19. `Password for database:` Enter the password for the embedded Postgres database included in LOCKSS 2.0-alpha. *Future versions will allow you to use an existing Postgres database and enter credentials accordingly.*
20. `Password for database (again):` Re-enter the password for the embedded Postgres database (if the two passwords do not match, the password will be asked again).
21. `OK to store this configuration:` confirm with `Y` that the summarized configuration data is correct and that you are ready to write it to a file.

If prompted to generate files, accept (or run `scripts/generate-lockss` immediately after).

CHAPTER

SIX

PREPARING TO RUN THE SYSTEM

Run the following script from `${INSTALLER_HOME}`:

```
scripts/assemble-lockss
```

This script sets up Docker-related infrastructure, configuration files and configuration data.

CHAPTER

SEVEN

RUNNING THE SYSTEM

Run the following script from `${INSTALLER_HOME}`:

```
script/deploy-lockss
```

This has the effect of creating a Docker stack (an orchestrated group of Docker services) by calling `docker stack create ... lockss-stack1` with a generated Docker Compose file parameterized appropriately and flanked by necessary infrastructure (Docker configs, Docker secrets, Docker volumes, etc.)

CHAPTER

EIGHT

SHUTTING DOWN THE SYSTEM

Run the following script from `${INSTALLER_HOME}`:

```
script/shutdown-lockss
```

This has the effect of calling `docker stack rm lockss-stack1`. Note that it takes a moment for all service containers to properly shut down.