LOCKSS System Manual

LOCKSS Program

Oct 30, 2023

LOCKSS 2.0-ALPHA3 SYSTEM MANUAL

Latest release: 2.0.34-alpha3 (2021-06-04)
First release: 2.0.31-alpha3 (2020-10-29)
System manual last built: Oct 30, 2023

Welcome to the LOCKSS 2.0-alpha3 System Manual.

CHAPTER

ONE

INTRODUCTION

The LOCKSS system is a distributed digital preservation software system developed by the LOCKSS Program at Stanford University Libraries.

The 2.x series of the LOCKSS system stems from the LAAWS (LOCKSS Architected As Web Services) initiative, an ambitious modernization project that includes rewriting the classic LOCKSS daemon as a suite of containerized components. This version, LOCKSS 2.0-alpha3, is the third preview release on the road to LOCKSS 2.0.

1.1 System Prerequisites

1.1.1 Machine

The LOCKSS system runs on a **64-bit Linux host** (physical or virtual), with at least **4 cores** (8 or more preferable), at least **8 GB of memory** (16 GB or more preferable) and at least **50 GB of disk space** (100 GB or more preferable).

1.1.2 Operating System

The LOCKSS system requires a **64-bit Linux** host compatible with Systemd, Snap and MicroK8s.

Flavors of Linux we have tested include:

- CentOS 8.2, 8.1, 8.0, 7.8, 7.6.

  Tip: CentOS 7 is our recommended OS.

  Caution: Snap is not available on CentOS 7.5 or earlier; version 7.6 or later is required.

- Debian 10.6, 10.5, 10.4, 10.3, 10.2, 10.1, 10.0, 9.13, 9.12, 9.9, 9.6, 9.5, 9.4, 9.3, 9.2, 9.1, 9.0.
- Linux Mint 20.0, 19.3, 19.2, 19.1, 19.0, 18.3, 18.2.

  Caution: Snap is not available on Linux Mint 18.1 or earlier; version 18.2 or later is required.

- OpenSUSE Leap 15.2, 15.1, 15.0.
- RHEL 8.2, 7.8.
- Ubuntu 20.04 LTS, 19.10, 19.04, 18.10, 18.04 LTS, 17.10, 17.04, 16.10, 16.04 LTS.

Tip: LOCKSS 2.0-alpha3 can probably be installed successfully on slightly different versions of the operating systems above, for instance CentOS 7.7 or Debian 9.11. Additionally, savvy users will likely succeed at installing LOCKSS 2.0-alpha3 on other Linux flavors.

Caution: We do not recommend Arch Linux or Fedora Linux 32, because MicroK8s 1.18.9, the currently available version in the required 1.18 series, does not seem to work on these platforms as documented. This highlights an inconvenience of the default Snap-only distribution of MicroK8s that we hope to address in 2.0-alpha4.

1.2 Security Considerations

Important: LOCKSS 2.0-alpha3 is a technology preview, not yet suitable for production environments.

Although the LOCKSS software itself and especially the LOCKSS peer-to-peer protocol remain as secure as ever, the operating environment for alpha versions of LOCKSS 2.0 is still being hardened. Please read about the security considerations below that are relevant as of LOCKSS 2.0-alpha3.

1.2.1 Networking

LOCKSS 2.0-alpha3 is the first version of the LAAWS (LOCKSS Architected As Web Services) initiative deployed in a Kubernetes environment. The Kubernetes networking model is sophisticated and requires complex interactions with the host operating system's network and firewall stacks. LOCKSS 2.0-alpha3, for the purposes of demonstrating basic functionality, requires disabling any of the user-friendly wrappers around `iptables`, such as `firewalld` or `ufw`, which can interfere with Kubernetes' `iptables` manipulations. Better integration with these firewall wrappers will arrive in LOCKSS 2.0-alpha4.

1.2.2 System Privileges

Likewise, to demonstrate basic functionality, LOCKSS 2.0-alpha3 runs as a dedicated `lockss` system user with `sudo` privileges. This requirement will be relaxed in future versions as we integrate better with the underlying operating system.

1.3 Upgrading From LOCKSS 2.0-alpha2

1.3.1 Recommended Approach

If you have been using LOCKSS 2.0-alpha2 (or LOCKSS 2.0-alpha1, or the LOCKSS 2.0-alpha technology preview), we thank you for helping us bring LOCKSS 2.0 closer to fruition through your testing and feedback.

Although there is an upgrade path from LOCKSS 2.0-alpha2, LOCKSS 2.0-alpha3 is organized significantly differently than prior alpha releases, and we recommend *installing LOCKSS 2.0-alpha3 from scratch* when possible.

1.3.2 Upgrade Path

If you intend to upgrade a LOCKSS 2.0-alpha2 system, please read this section.

Updating the LOCKSS Installer

On the command line, in the `lockss-installer` directory, type:

```
git checkout master
git pull
```

to update to the latest version of `lockss-installer` from GitHub.

Running the Upgrade Command

On the command line in the updated `lockss-installer` directory, type:

```
sudo scripts/upgrade-alpha2-to-alpha3
```

The script will purge your Docker environment of components, configuration files and images used by the LOCKSS system.

Installing Snap and MicroK8s

The LOCKSS system's containers are no longer orchestrated by Docker Swarm and no longer require Docker to run. The system now uses **MicroK8s**, a lightweight Kubernetes environment. To install the MicroK8s application package, you will need to install and use **Snap**. See *Installing Snap* and *Installing MicroK8s*.

Modifying the Environment

In order for LOCKSS 2.0-alpha3 to work properly, you will need to disable frontends to `iptables` like `firewalld` or `ufw`, and configure MicroK8s to use DNS in a way that avoids loopback addresses. See *Disabling Packet Filters* and *Configuring DNS* for details.

Reconfiguring the System

Upon successful completion, you will prompted to run *scripts/configure-lockss*. **Be advised that the configuration process will prompt you for the PostgreSQL database password.**

Starting LOCKSS

Once configuration is complete you can run lockss as usual with *scripts/start-lockss*.

CHAPTER

TWO

INSTALLING THE LOCKSS SYSTEM

This section describes how to install the LOCKSS system.

2.1 Creating the `lockss` User

The LOCKSS system runs under a system user named `lockss`, which is in a group named `lockss`, and which is capable of using `sudo`. The `lockss` user's password will be needed at various points during installation, both by explicit invocations of `sudo`, and in some cases by `microk8s` commands.

Important: See the *Security Considerations* section for more about this short-term requirement.

2.1.1 Creating the User on CentOS, OpenSUSE and RHEL

Type these commands:

```
sudo useradd --system --user-group --groups=wheel --create-home --shell=/bin/bash lockss

sudo passwd lockss
```

By default on **CentOS**, **OpenSUSE** and **RHEL**, `sudo` privileges and membership in the `wheel` group are equated. Adjust the above commands accordingly if your system has `sudo` configured differently.

2.1.2 Creating the User on Debian, Linux Mint and Ubuntu

Type these commands:

```
sudo useradd --system --user-group --groups=sudo --create-home --shell=/bin/bash lockss

sudo passwd lockss
```

By default on **Debian**, **Linux Mint** and **Ubuntu**, `sudo` privileges and membership in the `sudo` group are equated. Adjust the above commands accordingly if your system has `sudo` configured differently.

2.1.3 Obtaining a shell running as `lockss`

All commands shown in this document except those that explicitly invoke `sudo` should be issued from a shell running as the `lockss` user. Depending on your preference, you may login as `lockss`, or switch to the `lockss` user with this command:

```
sudo -i -u lockss
```

2.2 Disabling Packet Filters

Version 2.0-alpha3 of the LOCKSS system requires, in the short term, disabling any of the user-friendly wrappers around `iptables`, such as `firewalld` or `ufw`, which can interfere with Kubernetes' `iptables` manipulations.

Important: See the *Security Considerations* section for more about this short-term requirement.

2.2.1 Disabling `firewalld`

By default, **CentOS**, **OpenSUSE** and **RHEL** come with `firewalld`. You can check whether `firewalld` is running with:

```
sudo firewall-cmd --state
```

If it is running, stop and disable it with this command:

```
sudo systemctl disable --now firewalld
```

2.2.2 Disabling `ufw`

By default, **Ubuntu** comes with `ufw`. You can chech whether that `ufw` is running with:

```
sudo ufw status
```

If it is running, stop and disable it with this command:

```
sudo systemctl disable --now ufw
```

2.3 Installing Git

Git is a version control system, used to interact with code repositories.

The LOCKSS Installer is available from GitHub, and you will need a Git client to download it.

2.3.1 Checking for Git

Your operating system may already be equipped with a Git client. Type:

```
git --version
```

If the output is a version number, for example:

```
git version 2.28.0
```

then Git is already installed and you do not need to take further action.

If you see an error message similar to the following:

```
bash: git: command not found
```

then you need to install Git.

2.3.2 Installing Git

On many flavors of Linux, you can install Git with the built-in package manager:

- CentOS 7: see *Installing Git with Yum*
- CentOS 8: see *Installing Git with Dnf*
- Debian: see *Installing Git with Apt*
- Linux Mint: see *Installing Git with Apt*
- OpenSUSE: see *Installing Git with Zypper*
- RHEL 7: see *Installing Git with Yum*
- RHEL 8: see *Installing Git with Dnf*
- Ubuntu: see *Installing Git with Apt*

Installing Git with Apt

Apt is the package manager on **Debian**, **Linux Mint** and **Ubuntu**.

Use these Apt commands to install Git:

```
sudo apt update

sudo apt install git
```

Installing Git with Dnf

Dnf is the package manager on **CentOS 8** and **RHEL 8**.

Use this Dnf command to install Git:

```
sudo dnf install git
```

Installing Git with Yum

Yum is the package manager on **CentOS 7** and **RHEL 7**.

Use this Yum commands to install Git:

```
sudo yum install git
```

Installing Git with Zypper

Zypper is the package manager on **OpenSUSE**.

Use these Zypper commands to install Git:

```
sudo zypper refresh

sudo zypper install git
```

2.4 Downloading the LOCKSS Installer

You can download the LOCKSS Installer from GitHub using a Git command, as the `lockss` user:

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

All the remaining instructions assume that the current working directory is `lockss-installer`; `cd` to it now:

```
cd lockss-installer
```

2.5 Installing Snap

Snap is a Linux application package manager maintained by Canonical, makers of Ubuntu.

Snap is needed to install MicroK8s (a lightweight Kubernetes environment used by the LOCKSS system), which is also maintained by Canonical (and therefore only installed via Snap).

More complete instructions can be found at “Installing snapd” on Snapcraft, Snap’s home Web site, but we also provide some high level installation instructions below.

2.5.1 Checking for Snap

Some Linux flavors come with Snap pre-installed, for instance **Ubuntu**. To determine if your operating system is already be equipped with Snap, type:

```
snap version
```

If you see something similar to the following:

```
snap    2.46.1-1
snapd   2.46.1-1
series  16
kernel  5.8.13
```

then Snap is already installed and you do not need to take further action.

If you see an error message similar to the following:

```
bash: snap: command not found
```

then you need to install Snap.

2.5.2 Installing Snap

On many flavors of Linux, you can install Snap with the built-in package manager:

- CentOS 7: see *Installing Snap with Yum*
- CentOS 8: see *Installing Snap with Dnf*
- Debian: see *Installing Snap with Apt*
- Linux Mint: see *Installing Snap with Apt*
- OpenSUSE: see *Installing Snap with Zypper*
- RHEL 7: see *Installing Snap with Yum*
- RHEL 8: see *Installing Snap with Dnf*
- Ubuntu: see *Installing Snap with Apt*

Installing Snap with Apt

Apt is the package manager on **Debian**, **Linux Mint** and **Ubuntu**.

Preliminary Steps for Linux Mint 20

Before you can install Snap on **Linux Mint 20**, you first need to type this command:

```
sudo rm /etc/apt/preferences.d/nosnap.pref
```

This step is not needed for Linux Mint 19.

Use these Apt commands to install Snap:

```
sudo apt update

sudo apt install snapd
```

You can then proceed to the next step, *Enabling Classic Confinement*.

Installing Snap with Dnf

Dnf is the package manager on **CentOS 8** and **RHEL 8**.

Preliminary Steps for CentOS 8

Before you can install Snap on **CentOS 8**, you first need to type this Dnf command:

```
sudo dnf install epel-release
```

Preliminary Steps for RHEL 8

Before you can install Snap on **RHEL 8**, you first need to type this Dnf command:

```
sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

Use this Dnf command to install Snap:

```
sudo dnf install snapd
```

You can then proceed to the next step, *Enabling Classic Confinement*.

Installing Snap with Yum

Yum is the package manager on **CentOS 7** and **RHEL 7**.

Preliminary Steps for CentOS 7

Before you can install Snap on **CentOS 7**, you first need to type this Yum command:

```
sudo yum install epel-release
```

Preliminary Steps for RHEL 7

Before you can install Snap on **RHEL 7**, you first need to type these commands:

```
sudo rpm -ivh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

sudo subscription-manager repos --enable "rhel-*-optional-rpms" --enable "rhel-*-extras-
↪rpms"
```

Use this Yum command to install Snap:

```
sudo yum install snapd
```

You can then proceed to the next step, *Enabling Classic Confinement*.

References

- Installing snap on CentOS
- Installing snap on Red Hat Enterprise Linux (RHEL)

Installing Snap with Zypper

Zypper is the package manager on **OpenSUSE**.

First, use one of these Zypper commands (note the slight variation based on the exact version of your system):

```
# For OpenSUSE Leap 15.2:
sudo zypper addrepo --refresh https://download.opensuse.org/repositories/system:/snappy/
↪openSUSE_Leap_15.2 snappy

# For OpenSUSE Leap 15.1:
sudo zypper addrepo --refresh https://download.opensuse.org/repositories/system:/snappy/
↪openSUSE_Leap_15.1 snappy

# For OpenSUSE Leap 15.0:
sudo zypper addrepo --refresh https://download.opensuse.org/repositories/system:/snappy/
↪openSUSE_Leap_15.0 snappy
```

Then use these Zypper commands to install Snap:

```
sudo zypper --gpg-auto-import-keys refresh

sudo zypper dup --from snappy

sudo zypper install snapd
```

You can then proceed to the next step, *Enabling Classic Confinement*.

References

- Installing snap on openSUSE

2.5.3 Enabling Classic Confinement

MicroK8s uses Snap's so-called classic confinement model, which expects a top-level directory named `/snap` on your system. Nowadays this directory is located at `/var/lib/snapd/snap`. In order for Snap to install MicroK8s correctly, you need to create a symbolic link from `/snap` to `/var/lib/snapd/snap` with this command:

```
sudo ln -s /var/lib/snapd/snap /snap
```

(On some systems like Debian, `/snap` may already exist.)

2.5.4 Enabling Snap

You can then enable Snap on your system with the following command:

```
sudo systemctl enable --now snapd.socket
```

2.5.5 Logging Out and Back In

Log out and back in again (or restart your system) to ensure Snap's paths are updated correctly.

2.5.6 Verifying Snap

Snap offers a way to verify that things work correctly, by installing and running the *hello-world* Snap package. Type this Snap command:

```
sudo snap install hello-world
```

and then verify that this command:

```
hello-world
```

outputs the greeting `Hello World!`.

2.5.7 Configuring Snap Updates

The snap daemon will automatically update any installed Snap packages and by default it will check every four hours for updates.

For stability, you should adjust the frequency at which Snap checks and updates your Snap packages.

To adjust your update schedule to a year (the maximum allowed), use a refresh hold:

```
sudo snap set system refresh.hold="$(date --date='364 days' +%Y-%m-%dT%H:%M:%S%:z)"
```

2.6 Installing MicroK8s

MicroK8s is a lightweight Kubernetes environment. (Kubernetes is a system for managing and deploying containerized applications like the LOCKSS system.) This page will walk you through the initial installation of MicroK8s.

The LOCKSS system requires **MicroK8s 1.18**.

All the commands on this page should be run as the *lockss* user.

2.6.1 Installing MicroK8s

To install the MicroK8s Snap package, run this Snap command:

```
sudo snap install microk8s --classic --channel=1.18/stable
```

Troubleshooting

In some flavors of Linux (including Debian 9), sometimes the above command fails. Try running the following command first:

```
sudo snap install core
```

then retry installing the MicroK8s Snap package.

2.6.2 Joining the `microk8s` Group

MicroK8s creates a group to enable usage of commands which require admin privilege. To add your current user to the group and gain access to the `.kube` caching directory, run the following two commands:

```
sudo usermod -G microk8s -a lockss
```

2.6.3 Logging Out and Back In

Log out and back in again (or restart your system) for the group update to take place.

After you log back in as `lockss`, try:

```
microk8s --help
```

to check that MicroK8s is on your `PATH`. You should see a help message similar to the following:

```
Available subcommands are:
     add-node
     cilium
     config
...
```

If you see an error message instead (such as `bash:  microk8s:  command not found`), you need to ensure `/snap/bin` is on the `PATH`.

2.6.4 Generating the Kubernetes Configuration

Generate the Kubernetes configuration file from MicroK8s using these commands:

```
mkdir -p ~/.kube

sudo chown -f -R lockss ~/.kube

microk8s config --use-loopback > ~/.kube/config
```

2.6.5 Starting MicroK8s

Type the following command which will start MicroK8s and wait until it is fully ready.

```
microk8s status --wait-ready
```

It will then display the status of various MicroK8s subsystems:

```
microk8s is running
addons:
dashboard: disabled
dns: disabled
...
```

REFERENCES

Additional Documentation

- *Using MicroK8s*

MicroK8s References

- Complete MicroK8s Documentation
- MicroK8s Commands
- Troubleshooting Guide

Kubectl References

- Kubectl commands
- Kubectl Cheatsheet

2.7 Configuring DNS

After MicroK8s is up and running, adjustments need to be made to DNS processing in MicroK8s, which is handled by a MicroK8s component named **CoreDNS**. By default, CoreDNS is configured to use Google's nameservers; this is often undesirable in an institutional network, and unworkable for LOCKSS hosts with no public DNS records.

This section will reconfigure CoreDNS to use the same name servers configured for normal use on the host, i.e. those specified in `/etc/resolv.conf`. This can be done automatically as long as `/etc/resolv.conf` does not contain any loopback adresses; if it does, you will need to enter IP addresses of upstream name servers.

2.7.1 Configuring CoreDNS

From the `lockss-installer` directory and as the `lockss` user, run the following script:

```
scripts/configure-dns
```

You may be prompted for the `lockss` password for `sudo`, and if the script detects that `/etc/resolv.conf` contains loopback addresses, you will be prompted for a **semicolon-separated list of IP addresses of upstream DNS servers** that MicroK8s should use. Enter up to 3 non-loopback addresses from `/etc/resolv.conf`.

Note: Please note that IPv6 addresses do not currently work if entered at this prompt.

Example 1

Successful output from a run not requiring IP addresses of upstream DNS servers will look something like the following:

```
Enabling DNS
Applying manifest
serviceaccount/coredns created
configmap/coredns created
deployment.apps/coredns created
service/kube-dns created
clusterrole.rbac.authorization.k8s.io/coredns created
clusterrolebinding.rbac.authorization.k8s.io/coredns created
Restarting kubelet
DNS is enabled
Updating CoreDNS ConfigMap to use /etc/resolv.conf...
configmap/coredns configured
--------------------------------------------------------------------
Successfully changed CoreDNS ConfigMap
    forward . /etc/resolv.conf
--------------------------------------------------------------------
```

Example 2

Successful output from a run requiring IP addresses of upstream DNS servers will look something like the following:

```
Enabling DNS
Applying manifest
serviceaccount/coredns created
configmap/coredns created
deployment.apps/coredns created
service/kube-dns created
clusterrole.rbac.authorization.k8s.io/coredns created
clusterrolebinding.rbac.authorization.k8s.io/coredns created
Restarting kubelet
DNS is enabled
The /etc/resolv.conf file in your system contains a loopback address.
CoreDNS does not allow a loopback address to be assigned to pods.
Please enter a list of ip addresses of upstream dns resolvers.
```

(continues on next page)

(continued from previous page)

```
IP address(es) for dns lookup, separated by ';': [8.8.8.8;8.8.4.4] 208.67.222.222;8.8.8.8
Updating CoreDNS ConfigMap to use  208.67.222.222 8.8.8.8...
configmap/coredns configured
--------------------------------------------------------------------
Successfully changed CoreDNS ConfigMap
    forward .  208.67.222.222 8.8.8.8
--------------------------------------------------------------------
```

2.7.2 Verifying CoreDNS

If you type:

```
microk8s kubectl get all --all-namespaces
```

you should see output similar to the following:

```
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
kube-system   pod/coredns-588fd544bf-xq8ck     1/1     Running   0          5h51m

NAMESPACE     NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    ␣
↪             AGE
default       service/kubernetes     ClusterIP   10.152.183.1    <none>        443/TCP    ␣
↪             23h
kube-system   service/kube-dns       ClusterIP   10.152.183.10   <none>        53/UDP,53/
↪TCP,9153/TCP   5h51m

NAMESPACE     NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
kube-system   deployment.apps/coredns   1/1     1            1           5h51m

NAMESPACE     NAME                                 DESIRED   CURRENT   READY   AGE
kube-system   replicaset.apps/coredns-588fd544bf   1         1         1       5h51m
```

consisting of sections for different kinds of resources: pods, services, deployments, replica sets, etc. The pod containing `coredns` in the name (here `pod/coredns-588fd544bf-xq8ck`) should be in `Running` status and display `1/1` (one of one) ready.

2.8 Checking the System

After installing the LOCKSS system, you can confirm the status of installed components by running:

```
sudo scripts/check-sys
```

in the `lockss-installer` directory.

The script will do its best to check for any missing elements and permissions needed to run the LOCKSS cluster on the host machine:

- Check for Snap
- Check for MicroK8s
- Check for a user `lockss`.

- Check user `lockss` has appropriate group memberships and permissions.

CHAPTER

THREE

CONFIGURING THE LOCKSS SYSTEM

After installing the LOCKSS system, configure the system with the configure script:

```
scripts/configure-lockss
```

(If you have experience with classic LOCKSS daemon version 1.x, this is the equivalent of `hostconfig`.)

When run the first time, some of the questions asked by the script will have a suggested or default value, displayed in square brackets; hit `Enter` to accept the suggested value, or type the correct value and hit `Enter`. Any subsequent runs will use the previous values as the default value; review and hit `Enter` to leave unchanged. Password prompts will not display the previous value but can still be left unchanged with `Enter`.

The questions are:

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 NAT router.
4. `Initial subnet for admin UI access:` Enter a semicolon-separated list of subnets in CIDR or mask notation that should initially have access to the Web user interfaces of the system. The access list can be modified later via the UI.
5. `LOCKSS subnet for container access:` This is calculated from the MicroK8s node and should not need to be modified in a standard installation.
6. `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`.
7. `PROXY port:` Port for the LOCKSS content proxy. Accept the default – it can be changed later if necessary.
8. `Mail relay for this machine:` Hostname of this machine's outgoing mail server.
9. `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. `Password for <mailuser>@<mailhost> (again):` Re-enter the mail server password (if the two passwords do not match, the password will be asked again).

10. `E-mail address for administrator:` Enter the e-mail address of the person or team who will administer the LOCKSS system on this machine.
11. `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.
12. `Configuration proxy (host:port):` If a proxy server is required to reach the configuration server, enter its host:port here, otherwise leave this blank.
13. `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.
14. `Content data storage directory:` Enter the full path of a directory to use as the root of the main storage area of the LOCKSS system. This is where preserved content will be stored, along with several databases; it is the analog of `/cache0` in the classic LOCKSS system.
15. `Use additional directories for content storage?:` If you want to use more than one filesystem to store preserved content answer `Y`.

1. `Enter path to additional content storage directory <n> (q to quit):` If you entered `Y` to `Use additional directories` you will be prompted repeatedly for those paths; enter them one at a time, then enter `q` when done.

16. `Service logs directory:` Defaults to the content data storage directory; enter a different path if you want to put the logs elsewhere. In the classic LOCKSS system this was `/var/log/lockss`, but now there will be a set of subdirectories, one for each component service.
17. `Temporary storage directory:` Defaults to the content data storage directory. If that directory is remote (e.g. NFS), performance can be improved by supplying a local disk directory here. Do not use a RAM-based `tmpfs`; in some circumstances a substantial amount of temporary space (tens of GB) may be needed.
18. `User name for web UI administration:` Enter a username for the primary administrative user in the LOCKSS system's Web user interfaces.
19. `Password for web UI administration user <uiuser>:` Enter a password for the primary administrative user.
20. `Password for web UI administration user <uiuser> (again):` Re-enter the password for the primary administrative user (if the two passwords do not match, the password will be asked again).
21. `Use LOCKSS Metadata Query Service?:` Enter `Y` if you want the metadata query service to be run, otherwise `N`.
22. `Use LOCKSS Metadata Extractor Service?:` Enter `Y` if you want the metadata extraction service to be run, otherwise `N`.
23. `Use LOCKSS PostgreSQL DB Service?:`

- Enter `Y` to use the embedded PostgreSQL database. This is recommended in most cases.
  1. `Password for PostgreSQL database:` Enter a password for the embedded PostgreSQL database.
  2. `Password for PostgreSQL database (again):` Re-enter the password for the PostgreSQL database (if the two passwords do not match, the password will be asked again).
- Enter `N` if you wish to use your own PostgreSQL database. You will be queried for the details of your PostgreSQL service.
  1. `Fully qualified hostname (FQDN) of PostgreSQL host:` Enter the hostname of your PostgreSQL database (e.g. `mypgsql.myuniversity.edu`).

    2. `Port used by PostgreSQL host:` Enter the port where your running PostgreSQL database can be reached.
    3. `Login name for PostgreSQL service:` Enter the user name for your PostgreSQL database. The default is `LOCKSS`.
    4. `Schema for PostgreSQL service:` Enter the schema name to be used by the LOCKSS system. The default is `LOCKSS`.
    5. `Database name prefix for PostgreSQL service:` Prefix to use for any LOCKSS databases. The default is `Lockss` (note the uppercase/lowercase).
    6. `Password for PostgreSQL database:` Enter the password for your PostgreSQL database.
    7. `Password for PostgreSQL database (again):` Re-enter the password for your PostgreSQL database (if the two passwords do not match, the password will be asked again).

24. `Use LOCKSS Solr Service?:`

- Enter `Y` to use the embedded Solr server. This is recommended in most cases.
- Enter `N` to use your own Solr server.
    1. `Fully qualified hostname (FQDN) of Solr host:` Enter the hostname of your Solr database server (e.g. `mysolr.myuniversity.edu`).
    2. `Port used by Solr host:` Enter the port where your running Solr database server can be reached.
    3. `Solr core repo name:` Enter name of the Solr core for the LOCKSS repository. The default is `lockss-repo`.

25. `Use LOCKSS PyWb Service?:` Enter `Y` to use PyWb for content replay; enter `N` and you will be offered the option to use OpenWayback instead.
26. `Use LOCKSS OpenWayback Service?:` Enter `Y` to use OpenWayback for content replay (only if you did not opt for PyWb).
    1. `Okay to turn off authentication for read-only requests for LOCKSS Repository Service?:` OpenWayback currently does not supply user credentials when reading content from the LOCKSS repository, so the repository must be configured to respond to unauthenticated read requests. Enter `Y` to accept this, otherwise OpenWayback will not be enabled.
27. `OK to store this configuration:` Enter `Y` if the configuration values are to your liking, otherwise `N` to make edits.

If you enter `Y`, some checks will be run, necessary directories will be created, and you will be prompted to run `scripts/start-lockss` to start the configured system.

CHAPTER

FOUR

RUNNING THE LOCKSS SYSTEM

4.1 Starting the LOCKSS System

Run `scripts/start-lockss`. This script will call in turn:

- `scripts/generate-lockss`: This script takes your configuration data and turns it into a set of configuration files containing the right values.
- `scripts/assemble-lockss`: This script puts the configuration files and puts them in the right places, and ensures that all storage volumes are ready for use (creating them if necessary).
- `scripts/deploy-lockss`: This script deploys your LOCKSS stack by invoking Kubernetes.

4.2 Shutting down the LOCKSS System

Run `scripts/shutdown-lockss`.

4.3 Restarting a Running LOCKSS System

Run `scripts/restart-lockss`.

4.4 Removing a Configured LOCKSS System

To remove all configurations, volumes and networks installed by the LOCKSS system, run `scripts/uninstall-lockss`. This will **not** remove files from the persistent store.

CHAPTER

FIVE

USING THE LOCKSS SYSTEM

This section describes how to use the LOCKSS system.

5.1 Using the LOCKSS Configuration Service

Note: This page is under construction.

5.1.1 Accessing the Web User Interface

If you are already connected to the Web user interface (UI) of another component of the LOCKSS System, click *Config Service* in the top-left menu.

Alternatively, if your primary hostname is *<HOST>*, you can use your browser to connect to the LOCKSS Configuration Service Web user interface (UI) at `http://<HOST>:24621`.

Enter your Web UI username and password to login if prompted.

5.1.2 Adding Archival Units

To add AUs to the system for preservation:

1. In the top-right menu, click *Journal Configuration*.
2. In the center menu, click *Add AUs*.
3. Select one or more collections of AUs by selecting the checkbox next to the appropriate collection.
4. Click the *Select AUs* button. It may take a bit of time (60+ seconds) for the next screen to appear, while the list of AUs is built.
5. Select one or more AUs from the AU list. You may click *Select All* if you would like to select all AUs. If you choose to use select all AUs, please note that the next step may take some time to load.
6. Click the *Add Selected AUs* button. The time it takes for the page to refresh depends on the number of AUs added. Give the LOCKSS system some time to load the AUs and reload the page before moving on.
7. A screen will show a list of added AUs. Crawling of these new AUs will start automatically – no further action is necessary unless prompted by a footnote next to an AU's name.

5.1.3 Configuring a Crawl Proxy

If Web crawls must be routed through a Web proxy:

1. In the top-right menu, click *Content Access Options*.
2. In the center menu, click *Proxy Client Options*.
3. Select the *Proxy crawls* checkbox.
4. Enter the hostname and port of the Web proxy in the *HTTP Proxy host* and *Port* text areas, respectively.
5. Click the *Update Proxy Client* button.

5.1.4 Managing Access to the Web User Interfaces

This section is under construction.

5.2 Using the LOCKSS Crawler Service

Note: This page is under construction.

5.2.1 Accessing the Web User Interface

Note: Currently the crawler service is run as part of the poller service.

If you are already connected to the Web user interface (UI) of another component of the LOCKSS System, click *Crawler Service* in the top-left menu.

Alternatively, if your primary hostname is <*HOST*>, you can use your browser to connect to the LOCKSS Configuration Service Web user interface (UI) at `http://<HOST>:24631`.

Enter your Web UI username and password to login if prompted.

5.2.2 Monitoring Crawl Status in the System

The Crawl status of all configured AUs is available in the Archival Unit table

1. In the top-right menu, click *Daemon Status*.
2. Open the control in the middle of the screen that says *Overview* and select *Archival Units:guilabel:* from the drop down menu.
   - If prompted, enter your Username and Password again.
   - It will take a bit of time for the next screen to appear while the AU list is being built.
3. The Archival Units screen lists statistics for each configured AU

- the *Last Successful Crawl* column provides a timestamp of the most recent sucessful crawl.
- the *Last Crawl Start* column provides a timestamp of the last attempted crawl.
- the *Last Crawl Result* column provides the exit status of the last attempted crawl.

5.2.3 Causing an Archival Unit to Crawl

Archival units (AUs) that have been added to the system for preservation crawl periodically, but you can cause an AU to crawl on demand:

1. In the top-right menu, click *Debug Panel*.
2. Select an AU in the *AU Actions: select AU* drop-down list.
3. Click the *Start Crawl* button.
4. If the AU has crawled recently, you will be prompted to confirm that you wish to override the usual recrawl interval by clicking on the *Force Start Crawl* button.

5.2.4 Crawl Status Screen

To inspect the state of crawls, access the *Crawl Status* screen:

1. In the top-right menu, click *Daemon Status*.
2. In the center drop-down list, select *Crawl Status*. Alternatively, in the center overview, click on the second line, which says "*N* active crawls".

Top-Level Crawl Information

The top left of the Crawl Status table contains the number of active, successful or failed crawls, and a countdown until the next time the system will look at the AUs being preserved and pick some that are ready to crawl or recrawl.

Crawl Status Entry

Each line in the Crawl Status table contains:

- The name of the AU
- The type of crawl
- The start time of the crawl
- The duration of a finished or in-progress crawl
- The status of the crawl
- The number of bytes fetched over the network as part of the crawl
- The number of URLs fetched as part of the crawl
- The number of URLs parsed for more links
- The number of URLs remaining to be fetched as part of this crawl
- The number of URLs encountered as part of this crawl but excluded from being fetched
- The number of URLs fetched as part of the crawl, that received an HTTP Not Modified response
- The number of URLs that caused errors as part of this crawl
- The number of different content types encountered as part of the crawl

Most of these values can be clicked to see a list of the corresponding objects.

5.3 Using the LOCKSS Poller Service

Note: This page is under construction.

5.3.1 Accessing the Web User Interface

If you are already connected to the Web user interface (UI) of another component of the LOCKSS System, click *Poller Service* in the top-left menu.

Alternatively, if your primary hostname is <*HOST*>, you can use your browser to connect to the LOCKSS Configuration Service Web user interface (UI) at `http://<HOST>:24631`.

Enter your Web UI username and password to login if prompted.

5.3.2 Requesting Polls

This section is under construction.

5.3.3 Monitoring Polling and Voting

This section is under construction.

5.4 Using the LOCKSS Metadata Extraction Service

Note: This page is under construction.

5.4.1 Accessing the Web User Interface

If you are already connected to the Web user interface (UI) of another component of the LOCKSS System, click *Metadata Extraction Service* in the top-left menu.

Alternatively, if your primary hostname is <*HOST*>, you can use your browser to connect to the LOCKSS Configuration Service Web user interface (UI) at `http://<HOST>:24641`.

Enter your Web UI username and password to login if prompted.

5.4.2 Requesting Metadata Extraction

This section is under construction.

5.5 Using the LOCKSS Metadata Service

Note: This page is under construction.

5.5.1 Accessing the Web User Interface

If you are already connected to the Web user interface (UI) of another component of the LOCKSS System, click *Metadata Service* in the top-left menu.

Alternatively, if your primary hostname is <*HOST*>, you can use your browser to connect to the LOCKSS Configuration Service Web user interface (UI) at `http://<HOST>:24651`.

Enter your Web UI username and password to login if prompted.

5.5.2 Requesting Metadata Information

This section is under construction.

5.6 Replaying Web Content with Pywb

5.6.1 Accessing the Pywb User Interface

Given that your primary hostname is samp:*{<HOST>}*, you can use your browser to connect to the Pywb user interface (UI) at `http://<HOST>:8080`.

5.6.2 Replaying a URL

To view a URL from Pywb:

1. The Pywb screen provides a list of links to available collections. Click on the top-most collection which should be `/lockss`.
2. Enter the URL you want to replay in the URL search box.
3. Click the *Search* button.
4. Replay the most recent URL by clicking on the topmost entry of the third column.

5.6.3 Finding a URL From an AU to Replay

There are multiple ways to discover URLs belonging to an AU in the Configuration Service UI:

1. Obtaining a URL by clicking on “pages fetched” inside of crawl status
    - In the top-right menu, click *Daemon Status*.
    - Open the control in the middle of the screen that says *Overview* and select *Crawl Status* from the drop down menu.
    - Picking an AU from the active crawls, click on the number associated with *Pages Fetched* to bring up a list of URLs that have been crawled.

    - Copy one of the URLs and paste it in the Pywb interface as described previously.

2. Obtaining a Substance URL
    - In the top-right menu, click *Daemon Status*.
    - Open the control in the middle of the screen that says *Overview* and select *Archival Units* from the drop down menu. If prompted, enter your Username and Password again. It will take a bit of time for the next screen to appear while the AU list is being built.
    - Select an AU by clicking on the AU title in the first column.
    - Open the *Substance URLs* link
    - Copy one of the URLs and paste it in the Pywb interface as described previously.

5.7 Replaying Web Content with OpenWayback

5.7.1 Accessing the OpenWayback User Interface

Given that your primary hostname is samp:*{<HOST>}*, you can use your browser to connect to the Pywb user interface (UI) at `http://`*<HOST>*`:8080/wayback`.

5.7.2 Replaying a URL

To view a URL from OpenWayback:

1. Enter the URL you want to replay in the URL search box.
2. Click the *Search* button.
3. Select the *Year* or leave as :guilabel:`All*
4. Click *Take Me Back*.

5.7.3 Finding a URL From an AU to Replay

There are multiple ways to discover URLs belonging to an AU in the Configuration Service UI:

1. Obtaining a URL by clicking on "pages fetched" inside of crawl status
    - In the top-right menu, click *Daemon Status*.
    - Open the control in the middle of the screen that says *Overview* and select *Crawl Status* from the drop down menu.
    - Picking an AU from the active crawls, click on the number associated with *Pages Fetched* to bring up a list of URLs that have been crawled.
    - Copy one of the URLs and paste it in the OpenWayback interface as described previously.
2. Obtaining a Substance URL
    - In the top-right menu, click *Daemon Status*.
    - Open the control in the middle of the screen that says *Overview* and select *Archival Units* from the drop down menu. If prompted, enter your Username and Password again. It will take a bit of time for the next screen to appear while the AU list is being built.
    - Select an AU by clicking on the AU title in the first column.

- Open the *Substance URLs* link
- Copy one of the URLs and paste it in the OpenWayback interface as described previously.

CHAPTER SIX

APPENDIX

This appendix contains additional pages of information about the LOCKSS system.

6.1 Release Notes

6.1.1 LOCKSS 2.0.34-alpha3

Released: 2021-06-04
Also known as: LOCKSS 2.0-alpha3d

LOCKSS 2.0.34-alpha3 (also known as LOCKSS 2.0-alpha3d) is a bug fix release and the altest version of the LOCKSS 2.0-alpha3 system. It addresses a bug in the LOCKSS Installer.

Release Notes

- Fix previously deleted or renamed files.

Component Versions

LOCKSS 2.0.34-alpha3 consists of a configurable set of the following components:

- LOCKSS Installer version 2.0.34-alpha3
- LOCKSS Repository Service version 2.0.10.1
- LOCKSS Configuration Service version 2.0.4.1
- LOCKSS Poller Service version 2.0.2.1
- LOCKSS Metadata Extraction Service version 2.0.3.1
- LOCKSS Metadata Service version 2.0.2.1
- PostgreSQL version 9.6.12
- Apache Solr version 7.2.1
- Pywb version 2.4.2 (custom version 2.4.2-1)
- OpenWayback version 2.4.0 (custom version 2.4.0-1)

6.1.2 LOCKSS 2.0.33-alpha3

Released: 2021-01-29
Also known as: LOCKSS 2.0-alpha3c

LOCKSS 2.0.33-alpha3 (also known as LOCKSS 2.0-alpha3c) is a security release of the LOCKSS 2.0-alpha3 system. It addresses a vulnerability in a dependent code library.

Release Notes

- Use components patched to use Jackson-Databind 2.9.10.8 (CVE-2021-20190).

Component Versions

LOCKSS 2.0.33-alpha3 consists of a configurable set of the following components:

- LOCKSS Installer version 2.0.33-alpha3
- LOCKSS Repository Service version 2.0.10.1
- LOCKSS Configuration Service version 2.0.4.1
- LOCKSS Poller Service version 2.0.2.1
- LOCKSS Metadata Extraction Service version 2.0.3.1
- LOCKSS Metadata Service version 2.0.2.1
- PostgreSQL version 9.6.12
- Apache Solr version 7.2.1
- Pywb version 2.4.2 (custom version 2.4.2-1)
- OpenWayback version 2.4.0 (custom version 2.4.0-1)

6.1.3 LOCKSS 2.0.32-alpha3

Released: 2020-11-09
Also known as: LOCKSS 2.0-alpha3b

LOCKSS 2.0.32-alpha3 (also known as LOCKSS 2.0-alpha3b) is a bug fix release of the LOCKSS 2.0-alpha3 system. It addresses a bug in the LOCKSS Installer.

Release Notes

- Fix for broken multi-volume substitutions.

Component Versions

LOCKSS 2.0.32-alpha3 consists of a configurable set of the following components:

- LOCKSS Installer version 2.0.32-alpha3
- LOCKSS Repository Service version 2.0.10.0
- LOCKSS Configuration Service version 2.0.4.0
- LOCKSS Poller Service version 2.0.2.0
- LOCKSS Metadata Extraction Service version 2.0.3.0
- LOCKSS Metadata Service version 2.0.2.0
- PostgreSQL version 9.6.12
- Apache Solr version 7.2.1
- Pywb version 2.4.2 (custom version 2.4.2-1)
- OpenWayback version 2.4.0 (custom version 2.4.0-1)

6.1.4 LOCKSS 2.0.31-alpha3

Released: 2020-10-29
Also known as: LOCKSS 2.0-alpha3a

LOCKSS 2.0.31-alpha3 (also known as LOCKSS 2.0-alpha3a) is the first release of the LOCKSS 2.0-alpha3 system.

Release Notes

- The system's Docker containers are now managed by MicroK8s, a lightweight Kubernetes environment by Ubuntu makers Canonical, rather than Docker Swarm.
- Design and performance improvements to the repository layer, including support for multiple disk storage volumes (in preparation for migrating existing LOCKSS boxes, many of which have multiple disk storage volumes).
- The runcluster development environment can be used to run a lightweight LOCKSS system from JAR artifacts built locally from the Git codebase or retrieved from Maven Central or Sonatype OSSRH.
- Infrastructure for building LOCKSS plugins in the LAAWS environment.
- IP filtering for REST endpoints (similar to IP filtering for the LOCKSS Web user interface).
- Pywb 2.4.2.
- Bugfixes and performance improvements throughout the system.

Component Versions

LOCKSS 2.0.31-alpha3 consists of a configurable set of the following components:

- LOCKSS Installer version 2.0.31-alpha3
- LOCKSS Repository Service version 2.0.10.0
- LOCKSS Configuration Service version 2.0.4.0
- LOCKSS Poller Service version 2.0.2.0
- LOCKSS Metadata Extraction Service version 2.0.3.0
- LOCKSS Metadata Service version 2.0.2.0
- PostgreSQL version 9.6.12
- Apache Solr version 7.2.1
- Pywb version 2.4.2 (custom version 2.4.2-1)
- OpenWayback version 2.4.0 (custom version 2.4.0-1)

6.2 Network Ports

This page describes the default network ports used by the LOCKSS system.

Unless otherwise noted, all ports are **TCP**.

All ports in the 24600-24699 range should be considered reserved. The LCAP (LOCKSS polling and repair) port retains its historical value of 9729.

- 8080: Pywb or OpenWayback replay engine
- 9729: LCAP (LOCKSS polling and repair)
- 24600: *reserved* (currently LOCKSS Configuration Service UI)
- 24602: PostgreSQL
- 24603: Solr
- 24606: ActiveMQ
- 24610: LOCKSS Repository Service - REST port
- 24619: *reserved* (HDFS FS port)
- 24620: LOCKSS Configuration Service - Rest port
- 24621: LOCKSS Configuration Service - UI port
- 24630: LOCKSS Poller Service - REST port
- 24631: LOCKSS Poller Service - UI port
- 24640: LOCKSS Metadata Extraction Service - REST port
- 24641: LOCKSS Metadata Extraction Service - UI port
- 24650: LOCKSS Metadata Service - REST port
- 24651: LOCKSS Metadata Service - UI port
- 24670: LOCKSS Proxy

- 24671: *reserved*
- 24672: LOCKSS Audit Proxy
- 24673: *reserved*
- 24674: ICP server **(UDP)**
- 24680: LOCKSS Content Server (ServeContent)
- 24681: *reserved* Pywb replay engine
- 24682: *reserved* OpenWayback replay engine

6.3 Using MicroK8s

This document will provide instructions to using MicroK8s and Kubernetes commandline to access your cluster.

6.3.1 Using MicroK8s

Typing:

```
microk8s --help
```

will give a list of all commands:

```
Available subcommands are:
     add-node
     cilium
     config
     ctr
     dashboard-proxy
     disable
     enable
     helm
     helm3
     istioctl
     join
     juju
     kubectl
     leave
     linkerd
     refresh-certs
     remove-node
     reset
     start
     status
     stop
     inspect
```

To get more details about a command, type:

```
microk8s <command> --help
```

Getting the Status

To check the status of your cluster and which addons are enabled:

```
microk8s status
```

Example:

```
microk8s is running
addons:
dashboard: enabled
dns: enabled
metrics-server: enabled
ambassador: disabled
cilium: disabled
fluentd: disabled
gpu: disabled
helm: disabled
helm3: disabled
host-access: disabled
ingress: disabled
istio: disabled
jaeger: disabled
knative: disabled
kubeflow: disabled
linkerd: disabled
metallb: disabled
multus: disabled
prometheus: disabled
rbac: disabled
registry: disabled
storage: disabled
```

Starting and Stopping

MicroK8s will continue running until you decide to stop it. You can stop MicroK8s and its services by typing the command:

```
microk8s stop
```

You can restart by typing:

```
microk8s start
```

Accessing the Dashboard Locally

MicroK8s provides access to the standard Kubernetes dashboard. You can enable the dashboard and proxy to it on the local system.

```
microk8s enable dashboard

microk8s kubectl proxy &
```

The dashboard is available at the following URL: `http://127.0.0.1:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/`

6.3.2 Using Kubernetes

MicroK8s bundles its own version of `kubectl` for accessing Kubernetes. Use it to run commands to monitor and control your Kubernetes. Kubectl commands are prefixed by `microk8s`.

Getting a List of Commands

To get a list of commands, run:

```
microk8s kubectl
```

Example:

```
kubectl controls the Kubernetes cluster manager.

Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/

     Basic Commands (Beginner):
       create        Create a resource from a file or from stdin.
       expose        Take a replication controller, service, deployment or pod and␣
↪expose it as a new Kubernetes Service
       run           Run a particular image on the cluster
       set           Set specific features on objects

     Basic Commands (Intermediate):
       explain       Documentation of resources
       get           Display one or many resources
       edit          Edit a resource on the server
       delete        Delete resources by filenames, stdin, resources and names, or by␣
↪resources and label selector

     Deploy Commands:
       rollout       Manage the rollout of a resource
       scale         Set a new size for a Deployment, ReplicaSet or Replication␣
↪Controller
       autoscale     Auto-scale a Deployment, ReplicaSet, or ReplicationController

     Cluster Management Commands:
       certificate   Modify certificate resources.
       cluster-info  Display cluster info
```

(continues on next page)

(continued from previous page)

```
    top           Display Resource (CPU/Memory/Storage) usage.
    cordon        Mark node as unschedulable
    uncordon      Mark node as schedulable
    drain         Drain node in preparation for maintenance
    taint         Update the taints on one or more nodes

  Troubleshooting and Debugging Commands:
    describe      Show details of a specific resource or group of resources
    logs          Print the logs for a container in a pod
    attach        Attach to a running container
    exec          Execute a command in a container
    port-forward  Forward one or more local ports to a pod
    proxy         Run a proxy to the Kubernetes API server
    cp            Copy files and directories to and from    containers.
    auth          Inspect authorization

  Advanced Commands:
    diff          Diff live version against would-be applied version
    apply         Apply a configuration to a resource by filename or stdin
    patch         Update field(s) of a resource using strategic merge patch
    replace       Replace a resource by filename or stdin
    wait          Experimental: Wait for a specific condition on one or many␣
↪resources.
    convert       Convert config files between different API versions
    kustomize     Build a kustomization target from a directory or a remote url.

  Settings Commands:
    label         Update the labels on a resource
    annotate      Update the annotations on a resource
    completion    Output shell completion code for the specified shell (bash or zsh)

  Other Commands:
    alpha         Commands for features in alpha
    api-resources Print the supported API resources on the server
    api-versions  Print the supported API versions on the server, in the form of
↪"group/version"
    config        Modify kubeconfig files
    plugin        Provides utilities for interacting with plugins.
    version       Print the client and server version information

  Usage:
    kubectl [flags] [options]

  Use "kubectl <command> --help" for more information about a given command.
  Use "kubectl options" for a list of global command-line options (applies to all␣
↪commands).
```

Viewing Nodes

To view your nodes:

```
microk8s kubectl get nodes
```

View Cluster Information

To view cluster information:

```
microk8s kubectl cluster-info
```

To view everything currently running in the cluster:

```
microk8s kubectl get all --all-namespaces
```

To view running services in the default namespace:

```
microk8s kubectl get services
```

- Use `--all-namespaces` for all services in all namespaces
- Use `-n kube-system` for the Kubernetes system
- Use `-n lockss` for LOCKSS-specific services

Viewing Pod Logs

```
microk8s kubectl get pods <options>

microk8s kubectl logs <podname>
```

Describing a Running Pod

To describe a running pod:

```
microk8s kubectl get pods <options>

microk8s kkubectl describe <podname>
```