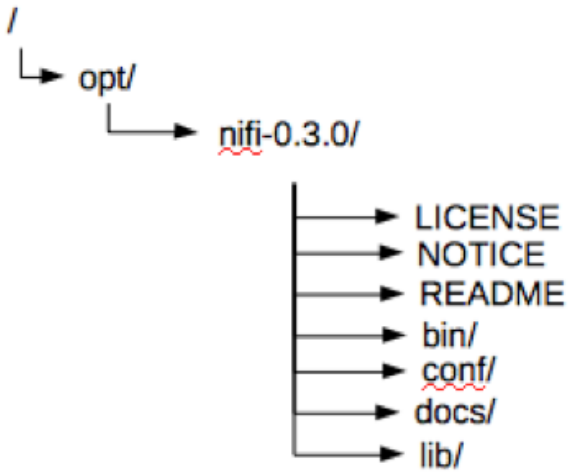
0.x.0 to 0.x.0 Upgrade

Upgrading NiFi

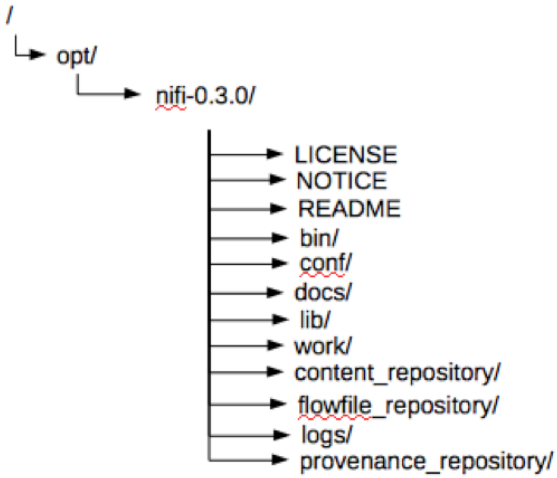
The purpose of this document is to explain how best to proceed with upgrading NiFi from an older version to a newer version. If done correctly, an upgrade can be performed with very minimal interruption to production dataflows. This document will initially focus on upgrading a single instance of NiFi and then cover changes to that procedure when upgrading a NiFi cluster. Before diving directly in to the procedure, it helps to understand the various pieces of a NiFi installation and how they all interconnect. By doing so we can paint a better picture of the dependencies that exist.

The NiFi software immediately following being installed has very few directories and files at its root installation level. Lets assume we are working with NiFi version 0.3.0 and that we have installed the software in a directory named /opt. This by default would create a single new directory inside /opt named nifi-0.3.0. Within the nifi-0.3.0 directory you will have the following three files (LICENSE, NOTICE, README) and four directories (bin, conf, docs, lib):



- **Bin** directory: This contains the executable for starting, stopping, obtaining NiFI status, and creating a NiFi dump output.
- **Conf** directory: This contains all the files that can/should be configured to control how your NiFi installation is bounded.
- **Docs** directory: This contains the various guides that are also available via the NiFi UI after it is running.
- **Lib** directory: This contains all the jar and nar files that are included with the currently installed version of NiFi.

So as we can see all the files that we as a NiFi installer/user care about are located in a single sub-directory. Lets continue to assume we installed our NiFi 0.3.0 by only doing the bare minimum required configuration (edit the nifi.properties file by adding a sensitive props key value) to get it running. After we were to start NiFi we would see the number of directories under /opt/nifi-0.3.0/ increase:



We can see by using the default NiFi configuration, NiFi has now created six additional new directories. These directories are used by NiFi to house its various repositories and logs the software uses to operate. A work directory is also created where NiFi dependencies (jars, nars, etc.) are exploded. In our example, that base installation path would be /opt/nifi-0.3.0/. Here is a quick break down of these six directories:

- **Content repository:** This repository contains the actual content for every FlowFile that is currently active in the NiFi instance. If archiving has been enabled, the archived copies are also kept here. * NiFi can be configured to have more then one content repository.

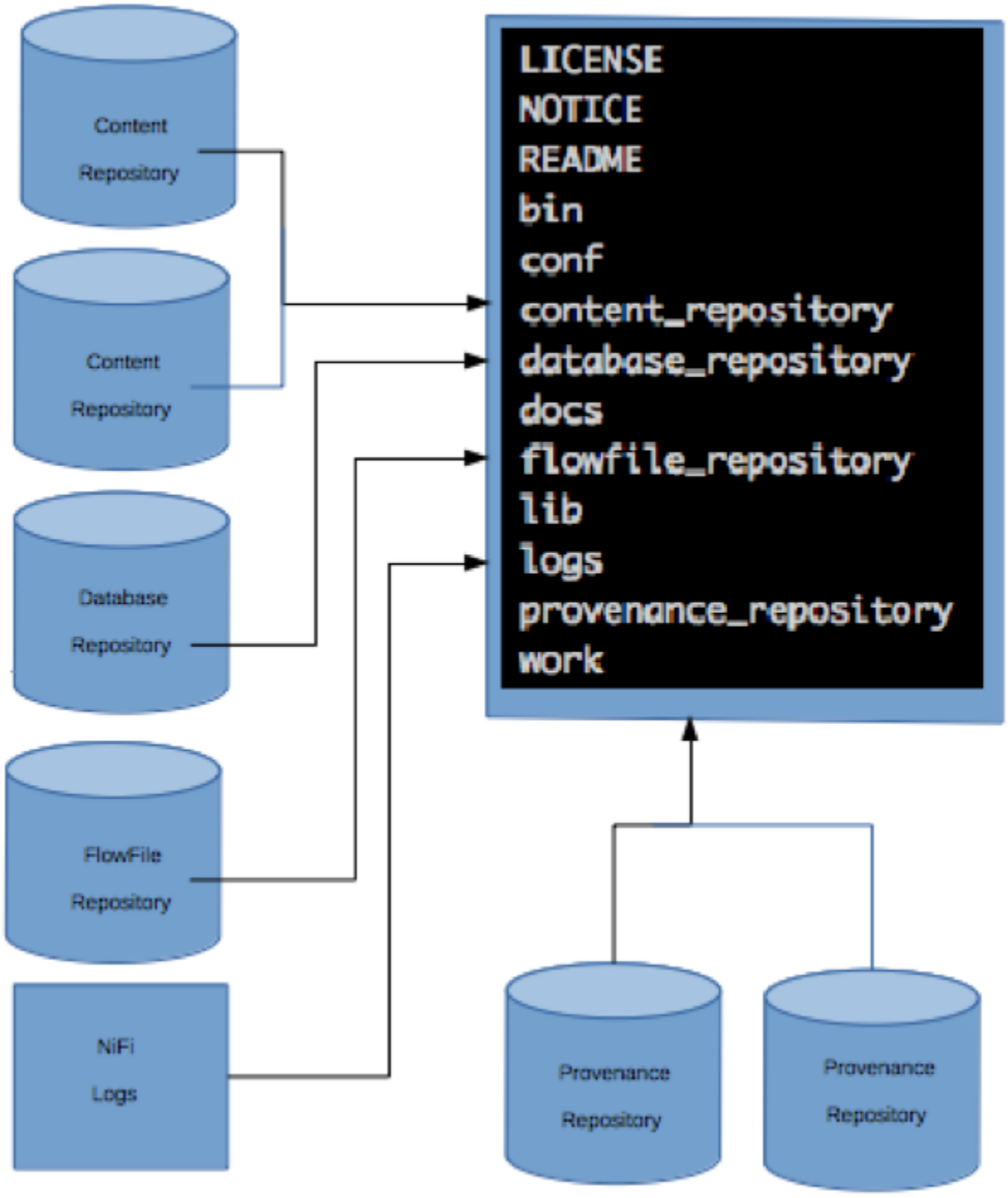
- **Database repository:** This directory contains two H2 databases. One database keeps track of all changes made within the NiFi graph/canvas. The other database tracks all users who have accessed the UI only if the NiFi instance has been configured to run securely.

- **Flowfile repository:** This repository keeps track of all the FlowFiles currently active in NiFi. It contains information such as where in eth dataflow is a FlowFile currently and what FlowFile attributes exist on those FlowFiles.

- **Logs:** This directory contains the various logs that NiFi outputs.

- **Provenance repository:** This repository contains events reported at various stages of a FlowFiles life through a dataflow(s). This information is used to generate a FlowFiles lineage. * NiFi can be configured to have more than one provenance repository.

- **Work:** This directory is where NiFi explodes the various jar and nar packages used by NiFi.



In addition to creating these six new directories at the base level of the install, NiFi also creates an additional directory and file inside the conf sub-directory. The flow.xml.gz file and the templates directory are created. The flow.xml.gz file contains everything you build on your graph/canvas, any reporting tasks or controller services created, and any NiFi configuration changes made via the UIs various interfaces. The templates directory will contain any NiFi templates that are created.

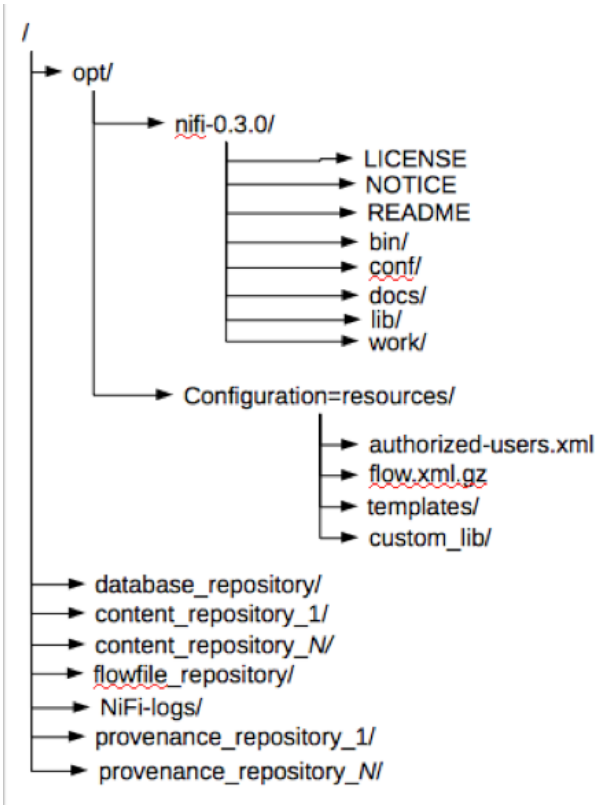
To greatly simplify the upgrade process we want to change the default configuration of NiFi, so that five of the six directories are created somewhere outside of the base installation path. There are additional considerations that should be taken in to account with regards to the various repositories and where they should be created for performance and software stability reasons. Refer to the Admin Guide for more details. We do not want to change the location of the work directory. We also want to move the static items out of the conf directory that will not change from version to version. That would include the flow.xml.gz file, authorized-users.xml file, and templates directory.

From an upgrade standpoint, how does re-locating these directories and files make the job of upgrading NiFi easier? These various repositories will not change between minor releases of the NiFi software. **In most cases, they do not change between major releases as well; *however*, one should consult the release notes to make sure before upgrading between major release versions.** The logging from version to version also remains unchanged.

What about my custom processors? Many users of NiFi write custom processors that are loaded in to NiFi's lib directory before starting NiFi. NiFi has also taken that in to consideration with regards to simplifying upgrading. A new line can be added in the nifi.properties file to add a second lib directory. If we name our second lib directory custom_lib, the new line might look something like this:

```
nifi.nar.library.directory=./lib
nifi.nar.library.directory.custom=/opt/configuration_resources/custom_lib
```

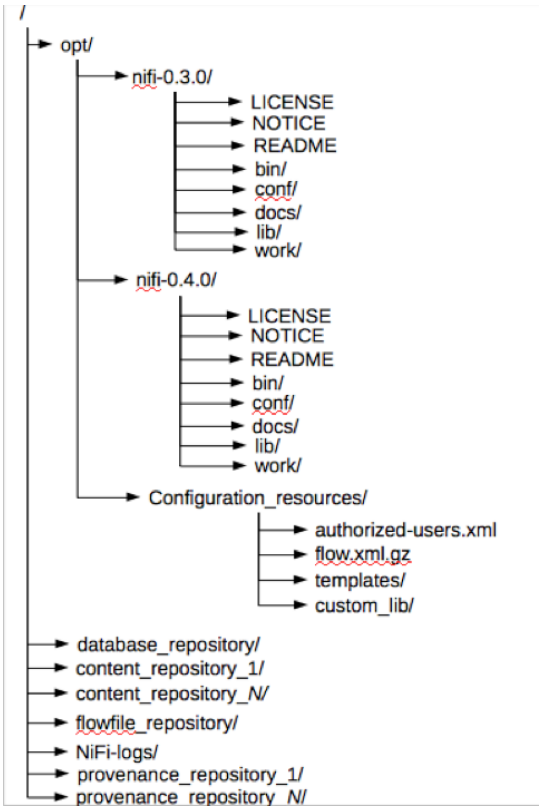
The default NiFi lib directory should remain unchanged. Just add the new line below it to specify where NiFi should find any additional custom jar/nar files. A good directory structure that supports easy upgrading may look similar to below:



Notice how a configuration_resources directory was created just under /opt where the static files from the conf directory will now be located, as well as, out custom_lib directory if needed. We have also moved all our repositories and the nifi logs to the root level. In most cases the logs directory (nifi-logs) and the repositories should be separate high speed RAIDs, high-speed external storage, or logical volumes when the other two options are not available.

Now that we have the best configuration to support easy upgrading, we can talk about the actual procedure to use. In our example thus far, we have an ideal installation and NiFi 0.3.0 is running in a production capacity. Now we want to upgrade to NiFi 0.4.0. Using the example directory structure above, here are the steps to follow:

1. Download the latest NiFi release
2. Install the new release in /opt. You should now have a directory structure similar to this:



3. Using the values already configured in the following files inside the conf directory of nifi-0.3.0, update the corresponding lines in the same files under nifi-04.0:

authority-providers.xml
bootstrap.conf
logback.xml
nifi.properties

*** It is **very** important that you make sure that no typos were made when configuring the various paths to your existing repos and the path to the flow.xml.gz file in the nifi.properties file in the newer release.
*** It is **very** important to make sure you have the same run as user configured in the bootstrap.conf file.
*** It may be possible to copy these files directly from an older version to a newer, but verify that no new lines were added or old lines removed first. If you do copy the nifi.properties file, update the nifi.version number.

4. Make sure all file and directory ownerships for nifi-0.4.0 match what was set on nifi-0.3.0
5. Stop NiFi version 0.3.0.

*** It is ok to stop your NiFi while files still exist in your dataflow(s) on the graph/canvas.

6. Start NiFi version 0.4.0 and immediately tail the app log.

    a. Verify via the app log that the new NiFi has fully started and begins processing data.
    b. If it fails to start for some reason, you can always restart the previous NiFi version while you investigate the cause.
    c. NiFi is commonly configured to run as a service. Make sure that any path or links for that service are updated to point at the newly installed version's executables now.
    d. Once new version has been verified as having started, is processing data, and can be accessed via the UI, the older version can be deleted.

What about upgrading a NiFI cluster?

The Nifi software configuration described in this procedure applies to both standalone and cluster Node instances of NiFi software. While it is acceptable have your NiFI cluster NCM configured the same way, it is not necessary to specify separate locations for content, flowfile, database, and provenance repos since the NCM does not ever process any FlowFiles itself.

The same procedure above can be followed up to and including step number 4.

5. Stop the NCM and all Nodes running 0.3.0.

*** It is important to stop everything. You cannot upgrade and restart one Node or NCM at a time. You could end up with mismatched versions of NiFi connected to your cluster NCM.
*** It is ok to stop your NiFi cluster while files still exist in your dataflow(s) on the graph/canvas.

6. Start the NiFi 0.4.0 NCM followed by starting all the NiFI 0.4.0 Nodes

    a. You should monitor your NCM via the NCM nifi-app.log or through the UI to make sure every Node joins the cluster.
    b. Any Nodes that fail to join should have their Node nifi-app.log checked for errors.

*** If you must restart the older NiFi version on a Node, you should bring that Node up as a standalone instance out of the cluster. It will work on the data it has in its repos, while you troubleshoot the problems with the newer version.

7. NiFi is commonly configured to run as a service. Make sure that any path or links for that service are updated to point at the newly installed version's executables now.
8. Once new version has been verified as having started, is processing data, and can be accessed via the UI, the older version can be deleted.

Easy upgrading configuration file example:

Below is an example of the relevant portions of a nifi.properties file and authority-providers.xml file configuration that make upgrading easy. Those portions have been configured to match the example setup presented in this material:

nifi.properties:

```
# Core Properties #

nifi.version=0.3.0

nifi.flow.configuration.file=/opt/configuration_resources/flow.xml.gz

nifi.flow.configuration.archive.dir=/opt/configuration_resourcesflow./flow_archive/

nifi.templates.directory=/opt/configuration_resources/nifi_templates

nifi.nar.library.directory=./lib

nifi.nar.library.directory.custom=/opt/configuration_resources/custom_lib


# H2 Settings

nifi.database.directory=/database_repository


# FlowFile Repository

nifi.flowfile.repository.directory=/flowfile_repository


# Content Repository

nifi.content.repository.directory.cont1=/content_repository_1

nifi.content.repository.directory.cont2=/content_repository_2

nifi.content.repository.directory.cont3=/content_repository_3


# Persistent Provenance Repository Properties

nifi.provenance.repository.directory.prov1=/provenance_repository_1

nifi.provenance.repository.directory.prov2=/provenance_repository_2
```

authority-providers.xml:

```
<authorityProviders>
    <provider>
        <identifier>file-provider</identifier>
        <class>org.apache.nifi.authorization.FileAuthorizationProvider</class>
        <property name="Authorized Users File">/opt/configuration_resources/authorized-users.xml</property>
        <property name="Default User Roles"></property>
    </provider>

    <!--<provider>
        <identifier>cluster-ncm-provider</identifier>
        <class>org.apache.nifi.cluster.authorization.ClusterManagerAuthorizationProvider</class>
        <property name="Authority Provider Port"></property>
        <property name="Authority Provider Threads">10</property>
        <property name="Authorized Users File">/opt/configuration_resources/authorized-users.xml </property>
        <property name="Default User Roles"></property>
    </provider>-->
```

Related articles