

XNAT Documentation / XNAT Administration / Configuring the Pipeline Engine


## Installing Pipelines in XNAT

 In XNAT version 1.9 onwards, the Pipeline Engine Framework now requires two components:

- the [Pipeline Engine](#)
- the [XNAT Pipeline Engine Plugin](#).

Specific pipelines are used in XNAT to take data from your project, run a process on that data, and potentially return new data to your project. In order to install a pipeline, you must have Terminal access to your XNAT web application. Once you have installed a pipeline on your site, any project owner can add that pipeline to their project and configure it for that project's use.

### Installing a Pipeline into your XNAT Web App


 This process requires you to rebuild the pipeline engine. We recommend that you do not do this while any pipelines are running.

To install a pipeline in XNAT, you need to get file system access and know where your **PIPELINE\_HOME** directory is located. By default, this is located at `/data/xnat/pipeline`, but another location can be specified in the [XNAT File System Settings](#) in the Admin UI.

Pipelines will have three major components: the **pipeline descriptor** component, which contains XML definitions and scripts; the **webapp** component, which will contain any template files, event handlers, or scripts to be used by the XNAT UI, as well as any data type definitions; and the **executable**, which is the processing package that the pipeline will apply to your data. Each of these components needs to be installed separately.

### Installing the Pipeline Descriptor Component



 This process assumes that you have a separate repo for the pipeline component of your code.

1. SSH into your Tomcat instance using your command line tool of choice, sudo as the user account that owns your XNAT webapp, and go to the `/tmp` folder.

2. Create a new directory in `/tmp` called `modules`, then enter that directory.

CODE

Copy

```
1 $ mkdir modules
2 $ cd modules
```

3. Clone your pipeline repository into this folder. This demo uses the DicomToNifti pipeline as an example.

CODE

Copy

```
1 $ hg clone https://bitbucket.org/nrg_customizations/nrg_pipeline_dicomtonifti
```

4. Navigate back to the `/tmp` folder, then clone the `xnat-pipeline` repo into this folder, then



XNAT Documentation

```
1 $ cd ..
2 $ git clone https://github.com/NrgXnat/xnat-pipeline-engine.git
3 $ cd xnat-pipeline-engine
```

5. Copy the provided sample gradle properties file into a working `gradle.properties` file, then open that file in an editor.

CODE

Copy

```
1 $ cp sample.gradle.properties gradle.properties
2 $ vi gradle.properties
```

6. In this file, find the "destination" setting, uncomment that line of code, and enter the path to your `PIPELINE_HOME` directory.

CODE

Copy

```
1 destination=/data/xnat/pipeline
```

**7.** In the same file, find the "modulePaths" setting, uncomment that line of code, and enter the temporary path that you just created.

CODE

Copy

```
1 modulePaths=/tmp/modules
```

**8.** Save and close your changes to this file. (hit CTRL-C , then type :wq to do this in VI.)

**9. Rebuild the Pipeline Engine.** In the xnat-pipeline folder, run gradlew to rebuild the pipeline engine.

CODE

Copy

```
1 $ ./gradlew
```

This Gradle command will now look in the modules directory that you created for any XML pipeline modules, and build them into the XNAT pipeline engine. When this is complete, this will place your pipeline XML files into your PIPELINE\_HOME catalog directory, and you will be able to enable them in your XNAT.

## Installing Webapp Components of your Pipeline as a Plugin

If you already have a JAR file, you can upload it to your plugins directory using the method of your choice, and skip to Step 4.

**1. Clone the pipeline webapp repository into your /tmp folder, and navigate into that folder.**

This demonstration uses the Dicom To Nifti plugin.

CODE

Copy

```
1 $ git clone https://bitbucket.org/nrg_customizations/nrg_plugin_dicomtonifti.git
2 $ cd nrg_plugin_dicomtonifti
```

## 2. Create a JAR file



A typical plugin uses the command `./gradlew jar`, but if your plugin requires dependencies, you may need to use the command `./gradlew fatJar` instead.

See the readme file of your downloaded plugin for details.

CODE

Copy

```
1 $ ./gradlew jar
```

This will create a new plugin jar file in the build/libs directory inside your pipeline repo. Depending on the code setup, a beans jar may also be created. This can be ignored.

### 3. Copy the JAR file into your plugins directory

In XNAT, your plugins directory is located in **XNAT\_HOME/plugins**. In a default XNAT installation, this would be **/data/xnat/home/plugins**.

CODE

Copy

```
1 $ cp build/libs/nrg_plugin_dicomtonifti-plugin-1.0.2-SNAPSHOT.jar /data/xnat/home/pluginir
```

### 4. Restart Tomcat



**Any active XNAT sessions will be immediately ended.** This should be a pre-scheduled operation on a running production XNAT.

## Installing the Pipeline Executable

### Installing into an XNAT Vagrant VM

Many pipelines require one or more external processing tools in order to execute correctly. For example, the DICOM to NIFTI pipeline requires on the **dcm2nii** executable, which is the program that actually does the heavy lifting to convert the data. The pipeline executable needs to be available in the PATH (environment variable) for the XNAT pipeline engine to be able to reference it and execute it. In a Vagrant VM, the easy way to make this happen is to install the executable in **/usr/local/bin**, since this is always on the PATH.

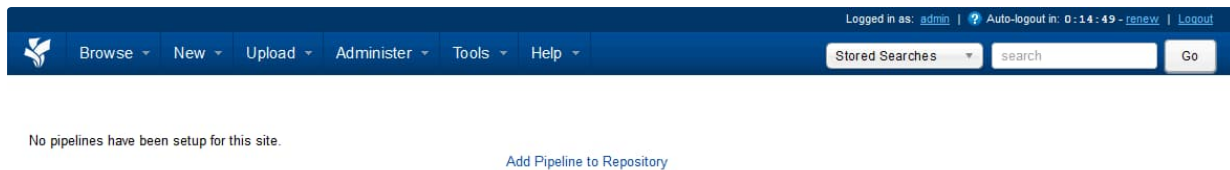
Once your executable is in place, you need to make sure that the XNAT pipeline engine can execute it. To so do, run the following command to set its permissions, where

{EXECUTABLE\_NAME} is the name of your executable file. (You should be sudoed to the root user if you aren't already.)

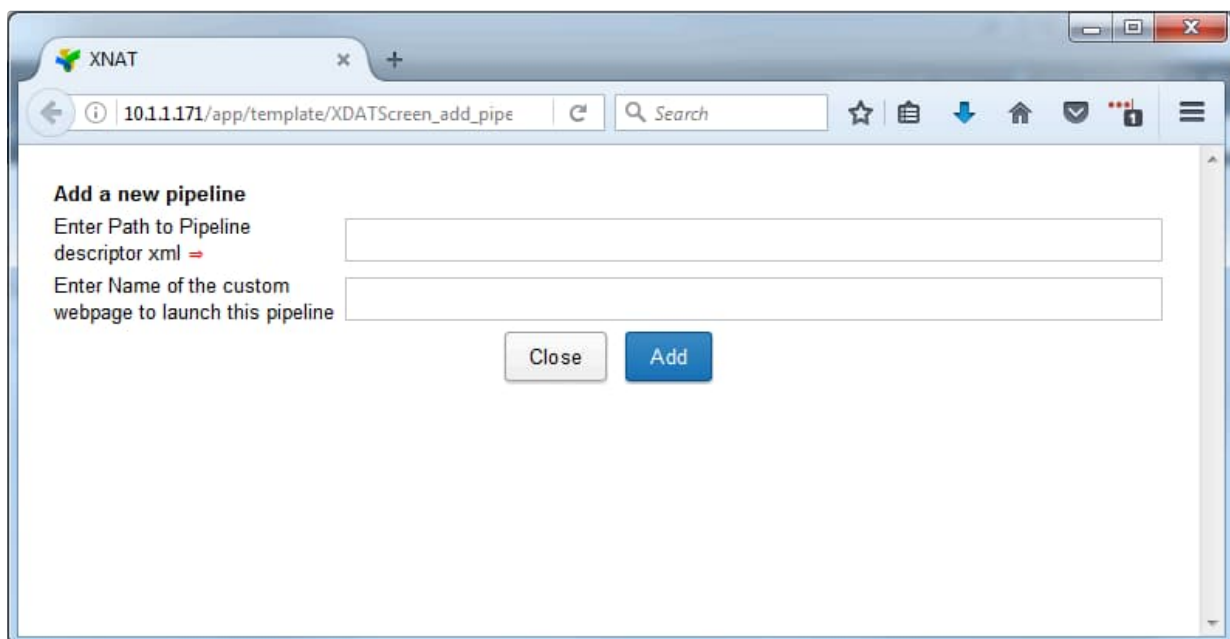
**CODE**[Copy](#)

```
1 $ chmod +x {EXECUTABLE_NAME}
```

## Registering your Pipeline in the Pipeline Administration UI



To register pipelines in your XNAT pipeline repository, go to **Administer > Pipelines**. If you are working on a brand-new XNAT with no pipelines installed, you will see a screen like this. Click the "Add Pipeline to Repository" link to get started. A popup window will open.



**1. REQUIRED:** Specify the location of your **pipeline descriptor's** XML file. You will need to type the entire system path, starting with the location of your PIPELINE\_HOME directory. For example: /data/xnat/pipeline/catalog/mricron/DicomToNifti.xml

**2. OPTIONAL:** Specify the name of your pipeline's launch screen. This file will be located in your **pipeline webapp** repository. The launch screen will typically be located in a directory like `/src/main/resources/META-INF/resources/templates/screens/{templatefile.vm}`. However, for this field, you only need to specify the **name** of the launch screen .vm file: e.g. `{templatefile.vm}`.

- ✓ XNAT provides a default pipeline launch screen, which will step through every parameter that the pipeline XML descriptor asks for and presents an input field to the user for it. These default launch screens are not especially attractive; typically a custom launch screen will be more desirable.

**3.** Click "Add" to add the pipeline to the XNAT pipeline repository. The popup will display a success notification, and the Administer Pipelines page will reload and display your newly added pipeline.

## Enabling the Pipeline Build Action on your XNAT Data Type

- i This step only needs to be performed once for each data type, and does not need to be redone for any new pipelines.

In order for your users to run pipelines on an image session through the XNAT UI, an XNAT Administrator must explicitly add the **PipelineScreen\_launch\_pipeline** action to the actions menu of the report page for that image session's data type.

- 1. Go to Administer > Data Types** to get started.
- Click on the data type element ID that you wish to modify. (For an MR Session, click on **xnat:mrSessionData**)
- A modal screen will open displaying all of the XNAT configuration, report, and security settings for that data type. Click **Edit** in the top right corner of the modal.
- The modal will now display editable fields for each of these components. Navigate to **Available Report Actions** and add the following values (other values can be left blank):

- **Name:** PipelineScreen\_launch\_pipeline
- **Display Name:** Build

- **Popup:** always
- **Secure Access:** edit

Notes:

- The **Display Name** field will be the label that appears in the actions menu for your report page. "Build" is the traditional label, but you can use something else like "Run Pipeline" or "Execute Process" as you see fit.
- The Grouping field will allow you to place your action in a submenu of the Actions menu.
- The **Sequence** field will specify where in the Actions box your new label appears. By default, the new action will float up near the top of the menu, directly beneath "Edit". This field accepts integer values, where higher numbers cause the menu to move down in the list of actions.

5. Click **Save** to save your configuration changes to the data type.

## Adding and Configuring a Pipeline in A Project

See XNAT User Documentation: [Adding and Configuring Pipelines for your Project](#).

## Monitoring, Stopping or Restarting a Running Pipeline

- ✓ You can view a log of all active or completed workflow actions at **Administer > More > View All Workflows**

...

Copyright © 2025 • Powered by Scroll Viewport & Atlassian Confluence