Running XNAT in a Vagrant Virtual Machine

In a matter of minutes, you can deploy a running copy of XNAT in a virtual machine, using Vagrant. For advanced users, you can also choose to setup one of multiple configurations for your XNAT VM.

Prerequisites:

- Install **Vagrant** (**1.8.7** or later is recommended)
- Install **Git**
- Install **VirtualBox** (**5.1** or later is recommended)

One-line XNAT VM Setup

If you have Git, Vagrant, and VirtualBox installed, the fastest way to get a fresh XNAT VM running is to enter the following command from an empty folder at a BASH terminal prompt:

```
git clone --branch master https://bitbucket.org/xnatdev/xnat-vagrant.git && cd xnat-vagrant && ./run xnat setup
```

This will clone the xnat-vagrant repo and run the script to install the latest official release. Just wait a few minutes, and when the text is done flying through your terminal, you should have a new VM running with instructions displayed on how to access the XNAT instance running in it.

Additional Prerequisites for Windows Users

- Bash terminal program - Git Bash is adequate and is included with the Git installer - this is what will be used for these instructions. You can also use **Cygwin**, but it isn't as straightforward to use.
- When installing Git, make sure to select to check-out as-is and commit with Unix line endings in your install options.
- Recommended SSH key management: **PuTTY**, **PuTTYgen**, and **Pageant**

As of this writing, Vagrant 1.8.7 is the recommended version for building an XNAT VM on Windows.

Download XNAT Vagrant

There are a couple different ways to get the XNAT Vagrant project.

Option 1: Install the xnat-vagrant Download Bundle

You can download the XNAT Vagrant project directly from our download repository:

Once you've downloaded the archive, unzip it using a standard zip utility. It will expand into a folder named **xnat-vagrant**. Proceed to run-setup-script.

Option 2: Clone the xnat-vagrant Source Repository

Clone the xnat-vagrant repository to your local computer by logging into the Terminal, navigating to your desired folder location, then running the following command:

clone

```
git clone --branch master https://bitbucket.org/xnatdev/xnat-vagrant.git
```

Follow the instructions at **https://bitbucket.org/xnatdev/xnat-vagrant** to clone your repo.

The 'develop' branch of **xnat-vagrant** is updated more often, and therefore possibly less stable, but can offer enhancements and improvements sooner. Just change the value for the `--branch` flag to 'develop' when cloning:

clone

```
git clone --branch develop https://bitbucket.org/xnatdev/xnat-vagrant.git
```

Configure and Customize XNAT Vagrant

The default properties for a particular configuration depend on the settings for that configuration. The available configurations can be found by looking in the **configs** folder under your **xnat-vagrant** project. As of this writing, the available configurations are:

Configuration	Description
xnat-release	Downloads a pre-built war file that can be deployed directly into Tomcat without going through the build process. The download is saved into a folder on the host computer so subsequent deployments don't require downloading the files again. However, you should download a new version if a new one has been released!
xnat-dev	Maps a source code folder on the host computer to a folder inside the VM at **/data/*project*/src/xnat**.to build XNAT, where *project* is the name of the configured project (this is **xnat** by default; see below for information on changing the default settings).
xnat-latest	Downloads and builds the XNAT source code inside the VM itself. This is the slowest of the standard XNAT 1.7 configurations (although still not too slow). Using this configuration is an easy way to download and evaluate the latest XNAT source code. This can be useful for troubleshooting build errors: if you are experiencing errors building locally, comparing these error with the **xnat-latest** build can help narrow down potential issues.
xnat165-box	Builds a VM running a pre-built, pre-configured XNAT 1.6.5 system. This is useful for working with existing code that runs on 1.6.5 and hasn't been migrated to work with 1.7.x. XNAT does not need to be built in this VM - the box file that's used already has XNAT 1.6.5 built and running.

The settings for each configuration can be found in the file **config.yaml** in the configuration folder. For example, the settings for the **xnat** configuration are in the file **configs/xnat/config.yaml**.

For many purposes, you can simply use the default values for a configuration. If you do need to override values in a configuration's default settings, create a file named **local.yaml** in the configuration's folder. You can then define your own values for the available properties. There's a file named **sample.local.yaml** you can refer to as an example. You can also reference the **config.yaml** file to see all of the properties that are set and available to override.

For example, a common requirement is to specify an IP address compatible with the local network configuration, as well as increase the amount of RAM available to the VM. Looking in the **config.yaml** for the **xnat** configuration, these have the following default values:

Default configuration settings

```
vm_ip:          10.1.1.17
ram:            2048
```

Your **local.yaml** file might look like this:

Custom configuration settings

```
vm_ip:          192.168.1.150
ram:            4096
```

When you run the script that actually creates your XNAT VM, the values in **local.yaml** override the values in **config.yaml**.

Create the XNAT Vagrant VM

Once you've downloaded the XNAT Vagrant project and set any custom properties required, you can create the VM one of three ways:

- The master **run** script
- The **setup.sh** script for the specific configuration you want to create
- Running the various **vagrant** commands manually

Master Run Script

The master **run** script is located in the top-level folder of the **xnat-vagrant** project. This script requires two arguments:

- The name of the configuration you want to use (see configure-and-customize for information on the various configurations)
- The command you want to execute

The available commands are:

- setup
- ssh
- start
- stop
- destroy

start and **stop** are aliases for Vagrant's own reload and halt commands respectively and can be used to start and stop VMs that have completed setup.

From the **xnat-vagrant** folder:

Running master run script

```
./run xnat setup
```

This launches the setup script in the **xnat** configuration folder. The setup script automates downloading the XNAT war and pipeline installer archive, then executes a series of Vagrant commands to get the VM fully provisioned. You'll see a bunch of text fly up the console, and the script should ultimately exit with a success message that displays the URL where your new XNAT instance can be accessed with your web browser.

Configuration Script

Each configuration folder has its own **setup.sh** script. For the most part, the master **run** script described in the previous section just invokes the **setup.sh** script for the specified configuration. You can do this directly as well:

Running configuration setup.sh

```
$ cd configs/xnat
$ ./setup.sh
```

The advantage to using a configuration-specific **setup.sh** script rather than the master **run** script is that being in the configuration folder makes it easier to work with your **local.yaml** to test settings and configurations for your VM.

Vagrant Commands

Finally you can run Vagrant commands directly within the configuration folders, with the main difference being you'll run **'vagrant reload'** *instead of* **'vagrant up'** after the initial setup. This is a more advanced option that can give you a great deal of control over how your VM is configured and created, but requires more knowledge over the specifics of configuring and running Vagrant. To find the commands used by the XNAT Vagrant project to build VMs, you should look through the **setup.sh** script that most closely resembles the configuration you want to work with.

Updating the XNAT Stack Base Box

Each Vagrant VM that builds XNAT 1.7.x uses an "xnatstack" base box for its machine environment. This environment is downloaded once and rarely updated. But on those occasions where it does need updating, you should know how to do it.

First, check to see if your installed base box is up to date.

```
$ vagrant box outdated --global
Loading existing configuration from /repos/xnat-vagrant/configs/xnat-release/vars.yaml...
* 'nrgxnat/xnatstack-ubuntu1604' is outdated! Current: 1.1. Latest: 1.1.1
```

If an outdated box is found, run an update command to update it.

```
$ vagrant box update --box nrgxnat/xnatstack-ubuntu1604
```

Work with the XNAT Vagrant VM

Once you've created an XNAT Vagrant VM, you can start to use the XNAT server there. There are a couple of steps you'll need to take to make this work:

- Configure the VM's host and server name
- Configure XNAT
- Log into the VM to manage services

Configure Host and Server Name

You can access your server directly through the configured IP address. By default, this is set to **`10.1.1.17`**. This can be easily changed in **local.yaml** before creating the VM. If you're comfortable using the bare IP address, you don't need to worry about the host and server name.

If you'd like something more readable and memorable, you'll probably want to configure the host and server name. The server name, also known as the *fully qualified domain name* or FQDN, is the standard web address, something like **images.xnat.org**. The hostname is the first part of the FQDN, in this case **images**. To work with a custom host and server name, you need to make a couple of changes:

- When customizing your VM configuration, add the **name** and **server** settings to your **local.yaml** configuration. The default value of **name** is **xnat-dev** so if you're OK with that as your host name, you don't need to override it. Note that **name** is also used to name the VM in the VirtualBox inventory, so it must be unique to the VMs running on your host server. You can also specify **name** and **host** separately: by default, **host** is set to the same value as **name**, but they can be different! The value for **server** should be set to the value you want for the FQDN:

  Configuring host and server name

  ```
  name:           xnatdev
  server:         xnatdev.xnat.org
  ```

- Creating the VM with the host and server name won't help your host machine find the server. For this, your machine needs to have a way to associate the FQDN with the VM's IP address. The easiest way to do this is to configure the **hosts** file on your host machine. For most operating systems, this is located in the file **/etc/hosts** and requires root access to modify:

  Editing /etc/hosts

  ```
  $ sudo vim /etc/hosts
  ```

  Windows also has a **hosts** file, but editing it can be trickier.

  Once you've opened the **hosts** file, you just need to add the line to associate the VM's IP address with the FQDN:

  Adding VM FQDN

  ```
  10.1.1.17 images.xnat.org
  ```

  Save the **hosts** file and you're done. No services need to be restarted or reloaded for this change to take effect.

It's possible to configure a DNS server to point to the VM or even modify the VM's network configuration so that it gets its IP from a DHCP server which can7 then manage the host name resolution to the IP address, but that depends on your local network architecture and configuration and is outside of the scope of this documentation.

Initialize and Configure XNAT

Once you have your VM running and can reach the server through its IP address or FQDN, you can configure your XNAT system. This task is the same regardless of whether you're running a VM, installing on a server, or running on a local development instance. This is described in XNAT Setup - First Time Configuration.

Manage VM Services

If you use a particular VM for more than trivial or trial operations, you'll eventually want to be able to manage the VM itself, not just the XNAT it hosts. To access the VM through the command line, you can use the Vagrant **ssh** command from the configuration-specific folder. Once on the VM, you can run commands as the default user, which has access to the Tomcat server and XNAT data folders. For more secure operations, you can use the **sudo** command to get root access. The code below shows a sample set of operations.

```
$ vagrant ssh
Loading /home/rherrick/Development/XNAT/1.7/vagrant/configs/xnat/config.yaml for Vagrant configuration...
Loading local overrides from /home/rherrick/Development/XNAT/1.7/vagrant/configs/xnat/local.yaml...
Setting up share from ../../logs to /data/xnat/home/logs
Setting up share from ../../plugins to /data/xnat/home/plugins
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)
* Documentation: https://help.ubuntu.com/
------------------------------------------------------------------
Ubuntu 14.04.4 LTS built 2016-04-25
------------------------------------------------------------------
xnat@xnatdev:~$ pwd
/data/xnat/home
xnat@xnatdev:~$ ls
config        logs  plugins  work
xnat@xnatdev:~$ sudo -i
root@xnatdev:~# service tomcat7 status
 * Tomcat servlet engine is running with pid 2554
root@xnatdev:~# service postgresql status
9.5/main (port 5432): online
root@xnatdev:~# logout
xnat@xnatdev:~$ sudo service tomcat7 restart
 * Stopping Tomcat servlet engine tomcat7                                  [ OK ]
 * Starting Tomcat servlet engine tomcat7                                  [ OK ]
xnat@xnatdev:~$ cd /data/xnat/
xnat@xnatdev:/data/xnat$ ls
archive  build        cache  ftp  home  pipeline  prearchive        src
xnat@xnatdev:/data/xnat$ cd
xnat@xnatdev:~$
```

The purpose of this is just to show how you can perform operations on your XNAT VM. The full range of operations you can perform as a Linux and XNAT system administrator is outside the scope of this documentation. For more information, you can find many tutorials and reference guides to working with Linux on the internet.