NOvA Jenkins Documentation

Jonathan Davies

March 23, 2017

Abstract

This document outlines details of the Jenkins build server and its use by NOvA to build NOvASoft releases and the nightly build. Details of what Jenkins is, how it is configured, how builds are carried out, how builds are downloaded and distributed as well as a troubleshooting section follow.

Contents

1 Quick Introduction

We use the Jenkins build system for providing builds of the trunk of NOvASoft nightly as well as tagged releases. The system is accessible via the url: `https://buildmaster.fnal.gov/`. We have 4 build projects there: a pair for each of the nightly build and a pair for the building of releases. In each of these pairs one project does the building whilst the second packages up and makes available the output of those builds (known as artifacts).

The second part to using the Jenkins build system is the scripting that is run currently run via cronjobs on `novabuild01.fnal.gov` which detects new builds and downloads the output (artifacts).

2 Why Jenkins

A long time ago in a galaxy far away we did our building of software manually using a dedicated machine `novabuild01.fnal.gov`. Fermilab then introduced Jenkins to allow experiments to manage the building of software on a pool of machines running a variety of operating systems.

NOvA were early adopters of Jenkins, but there was little access to information / common tools back then. As a result NOvA's Jenkins setup (configurations, scripting etc..) was all written by Jonathan Davies via much trial and error. It may not be the best implementation, and Fermilab may even provide tools now, but it works.

3 Accessing Jenkins

Jenkins has a user interface that can be accessed in a web browser here: `https://buildmaster.fnal.gov/`. For users who are not physically on the lab's network they need to use a VPN client in to the network in order to access the webpage. Details regarding installing and using a VPN can be found here: `http://networking.fnal.gov/vpn/`.

Users need to request access rights to NOvA's Jenkins projects in order to kick off builds and to alter the build projects. They should create a service desk ticket clearly stating this. They should also join the Jenkins users' email list: `build-service-users@fnal.gov`.

4 NOvA's Jenkins Projects

NOvA currently has 4 build projects that provide builds of NOvASoft:

- `nova_SRT_slf6_nightly_build` - Builds development nightly
- `nova_SRT_slf6_nightly_build_output` - If above if successful this packages up the output
- `nova_SRT_slf6_release_build` - Builds releases
- `nova_SRT_slf6_release_build_output` - If above if successful this packages up the output

4.1 Build projects

4.1.1 What they do

The build projects do two things:

- Set up the environment for a build
- Build the software

This is acheived by downloading and running a pair of scripts:

- `~/trunk/SRT_NOVA/scripts/jenkins_builds/jenkins_setup_script.sh`
- `~/trunk/SRT_NOVA/scripts/jenkins_builds/jenkins_build_script.sh`

These two scripts rely upon the existance (or lack of) these environment variables: `OS`, `RELEASE`, `BUILDTYPE`, `MTHREADED`, and `BRANCH`. These are inserted into the shell that is run for a build via configurations listed below.

4.1.2 Configuration

The build projects use a 'configuration matrix' of settings:

- `OS` - Slaves Axis

  The `OS` Axis lets specific operating systems be selected. Currently this is set to just `SLF6`. To cause builds in more than one operating system using you could select additional slaves labelled by operating system here.

- `BUILDTYPE` - User-defined Axis

  The `BUILDTYPE` Axis has two values `debug` and `maxopt`. This causes one build for each of these, so there are actually a pair of builds in this project.

In addition to the 'configuration matrix' (which is a fixed set of configurations) there are 'parameters' which are selected by the user when the build is triggered:

- `MTHREADED`

  This defaults to `TRUE`. The user can, if they wish, set this to `FALSE` to cause a serial build, which is much slower but easier to identify compile time errors.

- `RELEASE`

  The release that you will build. This defaults to `development` for the nightly build.

- `BRANCH`

  This only exists for the release build project. It defaults to `FALSE`. If it is set to `TRUE` it will attempt to build a branch not a release.

4.2 Output projects

The output projects are set as "downstream projects" for the build configuraitons. This means that if a build project completes successfully (i.e. there wasn't an error in the build) then it triggers the relevant output project.

This means that the output projects are assumed to ALWAYS SUCCEED by the download scripting.

These jobs just package up the built libraries etc. in the relevant "upstream project".

Each instance of this build running is given a build number, which increments by one. When we refer to a build number in any scripting it is the build number of the output project, not the build project.

5 Jenkins Download scripts

The second part to how NOvA uses and interacts with Jenkins is the downloading of releases. This is managed by one script:

```
^/trunk/SRT_NOVA/scripts/jenkins_builds/get_release_from_jenkins.sh
```

which is run on `novabuild01.fnal.gov` as a cronjob every 10 minutes in two different configurations (staggered by 5 minutes). One configuration is for downloading nightly builds, the other for downloading releases.

The job of this script is to detect new builds of either development or releases and download them. The two different configurations mentioned above are selected using the `-r` option.

If the option `-r development` is passed to this script it attempts to download the nightly build. If the option is not supplied, or a name other than `development` is passed the script will attempt to download a release build (if the `-r` option is not supplied it will try to detect the name of the release itself).

5.1 Details

5.1.1 Output

The nightly and release cronjobs that run these scripts send their output here:

```
/nova/app/home/novasoft/nightly_build/logs/get_release_from_jenkins.slf6.release.log
/nova/app/home/novasoft/nightly_build/logs/get_release_from_jenkins.slf6.nightly.log
```

Note that these logs (as with all the other cronjob logs in that directory) are "rotated" using `logrotate`. Each night the log file `*.log` is moved to `*.log.1`, and `*.log.{N}` is moved to `*.log.{N+1}`. Only the last 5 days of logs are kept.

5.1.2 Lock Files

In order to prevent multiple instances of the nightly and release download scripts running they first of all check for a lock file. If it exists they exit, else they create it, run and then remove it. The lock files are:

```
/nova/app/home/novasoft/.get_nightly_from_jenkins_lock_file_slf6
/nova/app/home/novasoft/.get_release_from_jenkins_lock_file_slf6
```

It may be necessary to remove an orphaned lock file. The need to do this should become clear from inspecting the output logs.

5.1.3 Download location

The script downloads releases to:

```
/nova/app/home/novasoft/slf6/build/releases/Jenkins_Downloads
```

Once they are downloaded they are unzipped and moved to:

```
/nova/app/home/novasoft/slf6/build/releases
```

This location is not the default installation location for the GPVM nodes. It was a choice to not download straight there and instead to a "staging area" to make sure that if something goes crazy we aren't overwriting our main software location. As a result the user should use the scripting detailed elsewhere `~/trunk/SRT_NOVA/scripts/` to move the release into place. For CVMFS there is a bash function defined in the the login that can do the download from Jenkins (as we can run the usual script above as a cronjob) to download a release. Again this should be documented elsewhere.

5.1.4 Build Number

When run in the release / nightly build cronjobs the script reads the build number to download from these files:

```
/nova/app/home/novasoft/slf6/build/releases/Jenkins_Downloads/next_build_number_release
/nova/app/home/novasoft/slf6/build/releases/Jenkins_Downloads/next_build_number_nightly
```

If the release is not found on Jenkins or the build is not complete / successful the script ends. If the build is found on Jenkins, successful and complete it is downloaded then the build number stored in the relevant file is updated to be `build number + 1`.

It is possible to pass a build number as an option to the script (this should match the build number of the relevant output project for the release / nightly build you are attempting to download). Beware that this will overwrite the number stored in the files above, so the next time the relevant cronjob runs it will be looking for `build number + 1`.

5.1.5 Record of Download

This isn't really used for much, but it seemed a useful feature to keep a file with details of what releases were successfully downloaded and when. Such files are populated by the download script and are located here:

```
/nova/app/home/novasoft/slf6/build/releases//Jenkins_Downloads/tars/build_record_{release,nightly}
```

6 Jenkins Project Configuration Backup

It is pretty fiddly to configure everything necessary for a project from scratch using the web interface. It is also good to be defensive about losing data. With this in mind the current NOvA Jenkins project configurations are backed up in our repository as xml files here:

```
^/trunk/SRT_NOVA/scripts/jenkins_builds/configurations/
```

These are backed up using this script

```
^/trunk/SRT_NOVA/scripts/jenkins_builds/backup_jenkins_config.sh
```

The options are pretty straight forward. Pass `-l` option and it will print out a list of the projects configurations that NOvA has on Jenkins. Pick one of those named projects to pass as an argument of the `-j` option along with a directory name to output the configurations into via `-o` and it should download the config for you.

7 Troubleshooting / FAQ

- Did the Jenkins build output project fail?

  `get_release_from_jenkins.sh` can get stuck trying to download a build number corresponding to a failed build. If this is the case increment the build number in the relevant `next_build_number_{release,nightly}` file to skip to the next build (of the build output project).

- Has my build finished downloading?

  Look at the log files and look for the completion of the download. It takes a while to download and also a while to unpack the tarballs, so be patient. Also note that the logs get a bit confused with output from the process that is downloading and any subsequent instance that opens, detects a lock file and end.

- The nightly / release build project failed

  Check the output from `{debug,maxopt}` for that particular build and try to find the error. If this proves difficult you can rebuild with `MTHREADED` set to `FALSE` to help track down the error. Otherwise off Jenkins build debugging is the way forward.

- The nightly / release build project succeeded but the corresponding output project failed

  This does happen, but very infrequently. There could be a problem on the node that the build / project was being carried out on that is independent of our code. We did have an issue in March 2017 where the build output projects needed to have the nodes on which they run restricted via configuration (we only figured this out because we reached out to the build users email list).

8 Improvements

- `get_release_from_jenkins.sh` stderr / stdout improvement

  A simple fix to include the process ID of the instance of this script being inserted in ALL output statements to help identify which cron process is reporting what to the log files

- Consolidating build projects

  It is quite possible that we don't need to have separate nightly and release build projects. If we just had one pair of build / output projects it would simplify things a little. My only concern would be figuring out how to correctly allow two concurrent builds of the same project without problems. Maybe that is easy / already works. Maybe it does't

9 Appendix

This is a bit ugly, but here are the functions that were defined in the CVMFS `.bashrc` to make downloading of Jenkins builds of releases easy to do.

```
function nova_cvmfs_get_jenkins_build_number(){

    if [ "$#" -lt 2 ];then
echo "Usage: nova_cvmfs_get_jenkins_build_number <release> <os>" >&2
return 1
    fi



    local release=$1
    local this_os=$2
    if [ "$this_os" == "slf5" ];then
local remote_dir=/build/nova/novasoft/releases
    elif [ "$this_os" == "slf6" ];then
local remote_dir=/nova/app/home/novasoft/slf6/build/releases
    else
        echo "ERROR: os \"${this_os}\" is not slf5 or slf6" >&2
        return 1
    fi

    local local_build_record=${HOME}/.jenkins_build_record_${this_os}
    if [ -e $local_build_record ];then
rm $local_build_record
    fi
    scp novasoft@novabuild01.fnal.gov:${remote_dir}/Jenkins_Downloads/tars/build_record_release\
     $local_build_record
    echo "INFO : Checking build record" >&2

    if [ -e $local_build_record ];then
cat $local_build_record | grep $release 1>&2
rm $local_build_record
    else
echo "ERROR: Couldn't get the build record from GPVM node" >&2
    fi
```

```
}

function nova_cvmfs_copy_release_from_jenkins(){
    if [ "$#" -lt 3 ];then
echo "Usage: nova_cvmfs_copy_release_from_jenkins <release> <os> <buildnum>" >&2
return 1
    fi

    local this_origin=$PWD
    local release=$1
    local this_os=$2
    local THIS_OS=`echo $this_os | tr '[:lower:]' '[:upper:]'`
    local buildnum=$3

    #Check that a transaction is open
    local TRANSACTION_FILE=~/.nova_server_transaction
    if [ ! -e $TRANSACTION_FILE ];then
echo "ERROR: Transaction not open" >&2
return 1
    fi
    if [ "$this_os" == "slf5" ];then
local local_dir=/cvmfs/nova.opensciencegrid.org/novasoft/slf5/novasoft/releases
    elif [ "$this_os" == "slf6" ];then
local local_dir=/cvmfs/nova.opensciencegrid.org/novasoft/slf6/novasoft/releases
    else
        echo "ERROR: os \"${this_os}\" is not slf5 or slf6" >&2
        return 1
    fi

    if [ -e ${local_dir}/${release} ];then
        echo "ERROR: Release already exists \"${local_dir}/${release}\"" >&2
        return 1
    fi

    local URL_BASE=https://buildmaster.fnal.gov/view/Nova/job/nova_SRT_${this_os}_\
    release_build_output/${buildnum}/artifact/buildout
    for build in debug maxopt;
      do
      local tarball=novabuild.${release}.${THIS_OS}.${build}.tar.bz2
      echo "INFO : Getting tarball \"${tarball}\"" >&2
      echo "       from URL_BASE \"${URL_BASE}\"" >&2
      echo "INFO : Destination \"${local_dir}\"" >&2
      if [ -e ${local_dir}/${tarball} ];then
          echo "ERROR: Tarball already exists \"${local_dir}/${tarball}\"" >&2
          return 1
      fi

      cd $local_dir
      wget --no-check-certificate ${URL_BASE}/${tarball}
      local RETVAL=$?
```

```
        if [ "$RETVAL" != 0 ];then
            echo "ERROR: Problem downloading tarball" >&2
            cd $this_origin
            return 1
        fi
        echo "INFO : Unzipping the tarball" >&2
        tar -xjf $tarball
        echo "INFO : Removing tarball" >&2
        rm  $tarball
        echo "" >&2
        cd $this_origin
      done;

}
```