LPF Training Handbook

M Hewitson 2014-04-25

1. Introduction

This note should serve as a handbook for the LISA Pathfinder training sessions. It will discuss the software you should install and the tools you should be familiar with before starting.

The steps are:

1. Setup a GIT account on the AEI server (see section 2)
2. Request access to the LPF team (see section 2)
3. Clone the ltpda_training GIT repository (see section 2)
4. Install LTPDA (see section 2)
5. Clone the ltpda_extensions repository (see section 2)
6. Install the extension module (see section 2)
7. Work through the 1st & 2nd LTPDA training sessions (see section 3)

More details on the usage of GIT are given in Section 4.

2. Software setup

The training session will use LTPDA Version 2.8 (which will be released around May 28th). As well as the core toolbox, an additional extension module is required (`LPF_DA_Module`) which is in the `ltpda_extensions` GIT repository. For the training, you should use the 'operations' branch, where you should **never** push. Step-by-step instructions are below.

The following table summarises the minimum requirements for the LPF operations:

Software	Version
MATLAB	2011a
Signal Processing Toolbox	6.13
Symbolic Math Toolbox	5.4
LTPDA Toolbox	2.8
LPF_DA_Module	GIT branch `operations` of `ltpda_extensions`

Accessing the relevant software repositories

As well as the GIT repository for the LTPDA extension module, another GIT repository for the management of files created/used during the training has been setup on the AEI GIT server. This 'training' repository also contains the version of LTPDA that should be used. A quick tutorial on the use of GIT is included later in this document, so if you are unfamiliar with GIT, it may be sensible to read that first to get your GIT client and environment properly set up.

You will need an account on this server. To get one do:

1

In a web browser, navigate to:

http://gitmaster.atlas.aei.uni-hannover.de

2

Register for an account by clicking the 'Register' link in the upper right.

3

Inform Martin Hewitson about your username (by email).

mailto: hewitson<at>aei.mpg.de

You will then be added to the LPF Ops team, which will give you access to the necessary git modules.

Once you are added to the team, you will be able to

4

Clone the LPF training module and the LTPDA Extensions module.

```
git@gitmaster.atlas.aei.uni-hannover.de:ltpda/lpf_training.git
git@gitmaster.atlas.aei.uni-hannover.de:ltpda/ltpda_extensions.git
```

From the command line, you can simply do:

```
$ cd /the/place/where/you/have/ltpda_extensions
$ git fetch
$ git checkout -b origin/operations
```

Further (brief) instructions on using GIT and graphical clients can be found in section GIT Usage.

The `lpf_training` GIT module will contain the basic structure needed to get us started.

Getting the software

The version of LTPDA will be available as part of the training GIT module; it will also be available on the LTPDA website.

Installing LTPDA

Follow the installation guide in the LTPDA user manual which is on-line at:

http://www.lisa.uni-hannover.de/ltpda/

In brief:

1

Unzip the `ltpda_toolbox_28.zip` file in the `lpf_training/201406/software` folder

2

Launch MATLAB and ensure no other LTPDA versions are in the MATLAB path.

Go to MATLAB's preferences, look under the path section and check for any paths which contain old LTPDA versions. If you find any, delete them. If you have no other folders in your path, you can also click on the 'Default' button to return MATLAB to a clean path, ready for installing LTPDA.

3

Add the new LTPDA folder (the unzipped folder) to the MATLAB path (select 'with subdirectories').

4

Run `ltpda_startup` at the MATLAB command prompt.

5

Run `run_tests` at the MATLAB command prompt to ensure correct installation.

6

Install the 3rd party Graphviz software. This is used for visualisation of object history and pipeline flow diagrams. Instructions are here:

http://www.lisa.aei-hannover.de/ltpda/usermanual/ug/additional_progs.html

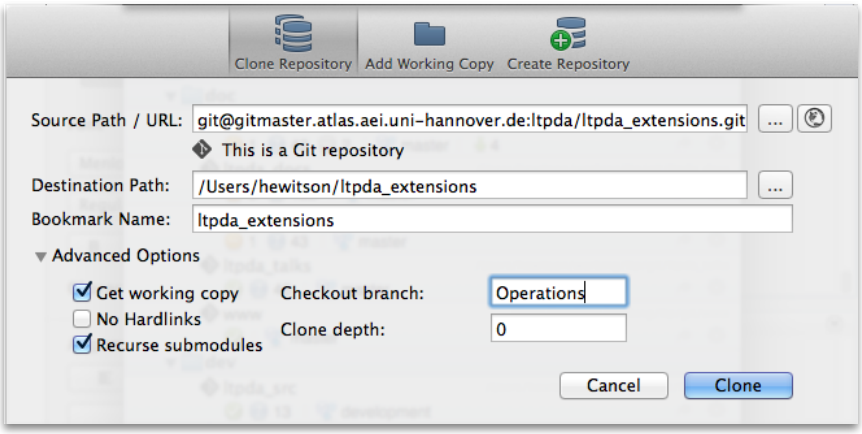
Installation of Extension modules

The relevant version of the `ltpda_extensions` module exists on a GIT branch called `operations`. You can 'checkout' this branch during the cloning process, or after cloning.

To do this during the cloning process, if you're using a command line GIT client, you would do

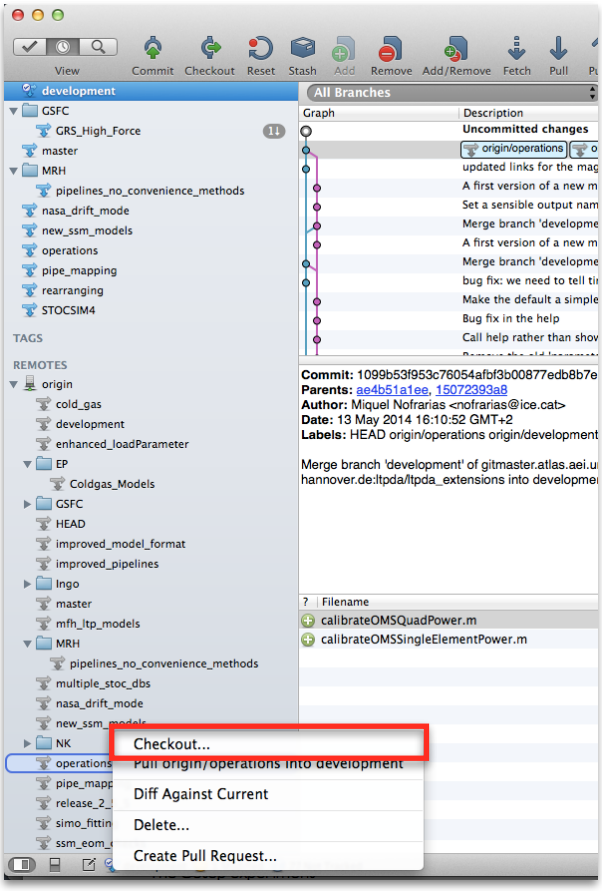
```
git clone -b operations git@gitmaster.atlas.aei.uni-hannover.de:ltpda/ltpda_extensions.git
```

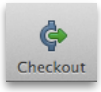
If you are using a graphical client, for example, SourceTree, you would clone the repository using File->New/Clone then in the dialog specify the repository URL and in the advanced options, the branch name, like this:



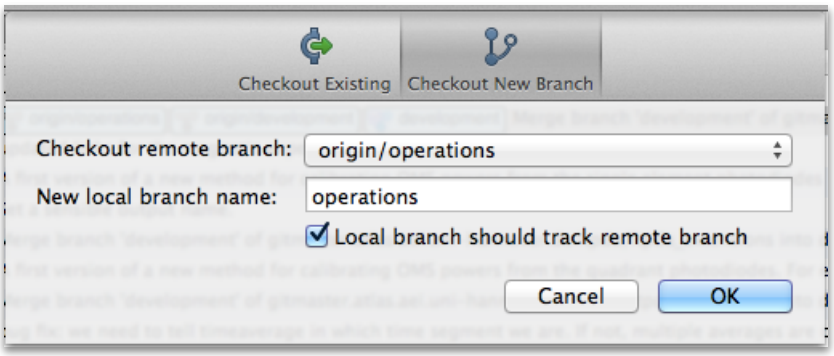
If you already have a clone of a repository, and you have pulled to update your local version, then you must switch to the 'operations' branch. The action here is to 'checkout' the operations branch. In the screenshot that follows, you see you need to:

1. expand the list of 'remotes' on the left of source tree
2. expand the branches that are under 'origin'
3. select the 'operations' branch
4. right click to get the context menu and choose 'checkout'



Alternatively, you can click the 'Checkout' button 

which will show this:



Choose the 'Checkout New Branch' tab and select 'origin/operations' from the branch drop down list. Then click 'ok'.

This can also be done from the command line:

```
$ cd /the/place/where/you/have/ltpda_extensions
$ git fetch
$ git checkout -b origin/operations
```

Once you have a clone, and you are on the correct branch, you need to add this module to LTPDA:

1. Launch MATLAB

2. Start LTPDA:

```
>> ltpda_startup
```

3. Open LTPDA's preferences:

```
>> LTPDAprefs
```

4

Remove any reference to older versions of LTPDA extension modules, like LTPDA_STOC_Module or LPF_DA_Module.

5

Click 'Browse' and navigate to the folder called `LPF_DA_Module` inside the `ltpda_extensions` directory you cloned with GIT.

6

Click the '+' button to add the module to LTPDA. Close the preferences.

7

Restart MATLAB and run `ltpda_startup` again.

This last step (7) requires a git command line client to be available to MATLAB. On a Mac, this should be an automatic process, though there have been reports that this is not always the case. On other systems, ensure that MATLAB has access to a git client. To check this, in MATLAB you can do

```
>> !git
```

and you should see something like

```
>> !git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-
path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]
```

Known Issue

On Mac OS X versions which use java 1.7 (on newer versions of MATLAB) there is an issue with the LTPDAprefs GUI when selecting paths of extension modules. There is a bug in the underlying java file-chooser widget which means there is an 'Open' button instead of a 'Choose' button. The result is that you can't select a directory but must, instead, select a file (for example the .xml file in the extension) then delete the filename from the path before pressing the + button to add the extension.

3. Getting Started with LTPDA

If you are not already familiar with LTPDA, the best thing to do is to walk through the two on-line LTPDA training sessions. They should take a couple of hours for each session. In the June Operations Training event, we are assuming that you have done this!

Access to the training sessions is via the MATLAB documentation. In MATLAB do:

>> doc

and navigate to the LTPDA user manual. *(On new versions of MATLAB, this is somewhat hidden under the "Supplemental Software" link at the bottom left of the documentation browser.)*

or via the on-line version under:

http://www.lisa.aei-hannover.de/ltpda/usermanual/ug/ltpda_training_intro.html

http://www.lisa.aei-hannover.de/ltpda/usermanual/ug/ltpda_training_2_intro.html

4. Basic GIT Usage

For the LPF Operations, you will need to familiarise yourselves with the usage of GIT for version control of all manner of files. I describe the basic usage below.

You will need to get a GIT client installed on your machine. The following are available:

1) GIT command line client
2) SmartGIT
3) SourceTree (Mac OS X and Windows)

You will need to create an ssh key (public and private pair). On a Mac, for example, in a terminal you do:

$ ssh-keygen -t dsa

this will create two files:
~/.ssh/id_dsa
~/.ssh/id_dsa.pub

Once you have your account on the gitorious web interface:

https://gitmaster.atlas.aei.uni-hannover.de

1) log into the web interface and click on the Dashboard link at the top of the first page
2) From there, click on “Manage SSH keys”

3) Click “Add SSH Key”
4) Then in a terminal do:

```
$ cat ~/.ssh/id_dsa.pub
```

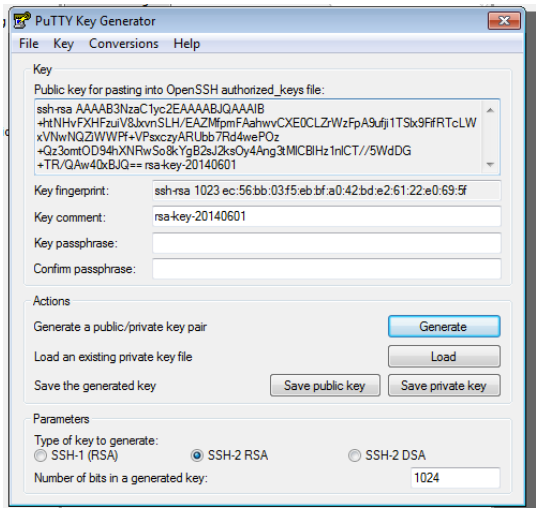
5) Copy the output text and paste it into the field in the web browser and then click “Save”

Following these steps allows your machine to ‘talk’ to the GIT server without using a password for logging in. (You still need your username and password for the web interface, however.)

Now you can setup your GIT client to ‘clone’ the desired repositories. That’s covered in the section “Cloning” below.

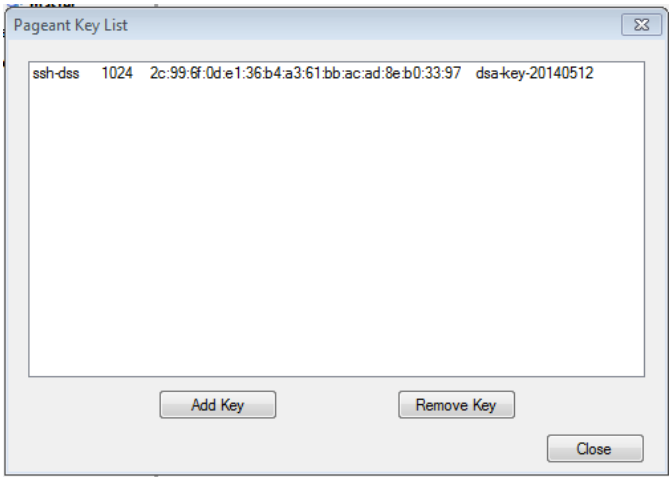
SourceTree and SSH on Windows

1) Install SourceTree. It contains handy tools for managing SSH keys.
2) After completing the installation, click on ‘Tools’ -> ‘Create or Import SSH Keys’
3) Click ‘Generate’. You will be asked to move randomly the mouse over a certain area
4) Click 'Save private key' and select a folder
5) Copy _all_ the content of the window where the numbers appeared (see figure)



6. Go to Gitorious webpage (https://gitmaster.atlas.aei.uni-hannover.de), select 'Your dashboard' and click on 'Manage SSH keys'
7. Click on 'Add SSH key' and paste the content you copied in step 5)
8. In the system tray, ensure the SSH Agent is active (if not, you can launch it from SourceTree by clicking 'Tools' -> 'Launch SSH Agent')
9. In the SSH Agent, ensure the SSH key is loaded (right-click on the SSH Agent icon in the system tray
10. If he SSH key is not loaded , load it by clicking on 'Add key' and selecting the private key you saved in Step 4

[This last step might be necessary upon SourceTree restart or system reboot]



Cloning

The first step in your GIT experience is to make a clone of the LPF Training repository. GIT is a distributed version control system, which means that you are always operating on your local copy of a repository. To make your changes available to others, you will 'push' your changes to the central server (or to any other server, in principle).

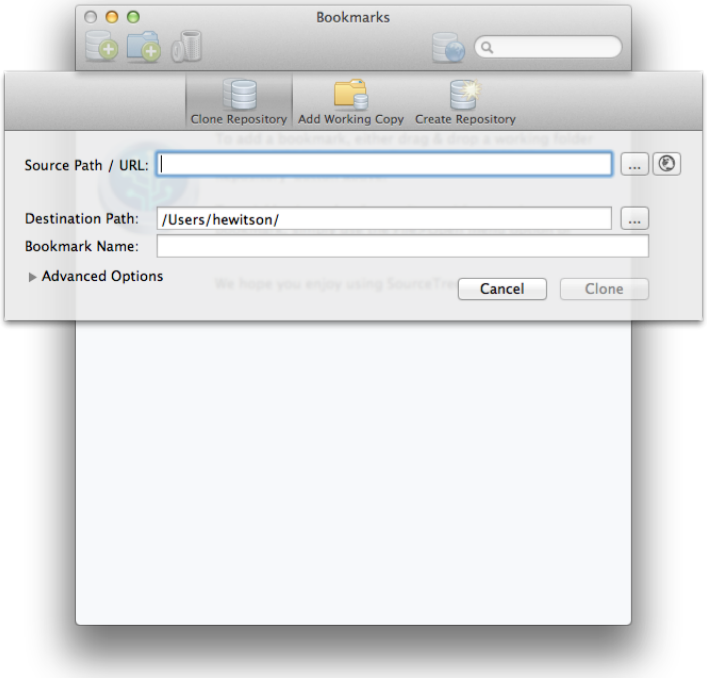
In the training session, you will be pushing your logbooks and scripts to the LPF Training repository, but you should never need to push anything to the LTPDA Extensions repository. Care should be taken!

So let's clone the LPF Training repository. I assume you are using SourceTree on a Mac here, but any other client will be similar.

When you start SourceTree for the first time it will look like this:

You now click on the first button to create a new clone (or use File->New or cmd-N).
The source URL can be found on the gitorious web interface. Log in to the web interface and click on “Dashboard”. Then click on `ltpda/lpf_training` under the “Your Repositories” list on the right-hand side. From there you can see the URL you need to copy into the URL field in SourceTree. It will be something like:

```
git@gitorious.atlas.aei.uni-hannover.de:ltpda/lpf_training.git
```



So you enter that in the SourceTree interface, select a local path for the cloned repository, and click “Clone”.

Once you have cloned the `lpf_training` repository, you now have a working repository on your machine. That means you can make commits against your local repository. This is a different model to CVS and suggests a different work-flow. Now you can commit often, such that the changes per commit are small, and these commits are only present on your local repository. Only once you push them back to the server, do these commits become available to others. More on that in the next section.

Staging Commits and Committing

When you make changes to files managed by the repository, you have to commit those changes into the branch you are currently working on, typically the main branch. I won’t talk about branches here - the interested reader/developer can google for “git branching”, or similar, and read all about it.

Before committing, you can decide what you want to commit. For example, you may have made 2 changes at different places in a single file. You can decide to commit one of those changes, but not the other. To do that, you ‘stage’ that ‘hunk’ of text that has changed which you want to commit, then go ahead and commit it.

Again, committing is only against your local repository - to make your changes available to others you need to ‘push’ them to the server. More on that in the next section.

Pushing and Pulling

As stated a couple of times above, with git you work with a local repository. To propagate your changes to others, or to get the changes others have made, you need to either push or pull. Typically, before you push your changes back to the server, you need to pull any

changes that others have made, resolve any merge conflicts, commit those conflict resolutions, and then push. This all sounds complicated, but it is essentially the same as we are used to with CVS other than you have these two additional steps - pull and push.

One of the great advantages of GIT and other distributed systems, however, is that you can commit changes without the need of a network connection. Thus you can commit changes as you work, rather than collecting a huge number of changes while you are off-line, and then have to go through them once you have network again.

For the LPF Ops, frequent commits, pulls and pushes is the recommended workflow. This will allow everyone to see what everyone else is doing. Don't wait until a script or document is final before committing and pushing. A sensible approach maybe to pause every ~30 minutes to commit and push any changes, especially to logbooks.