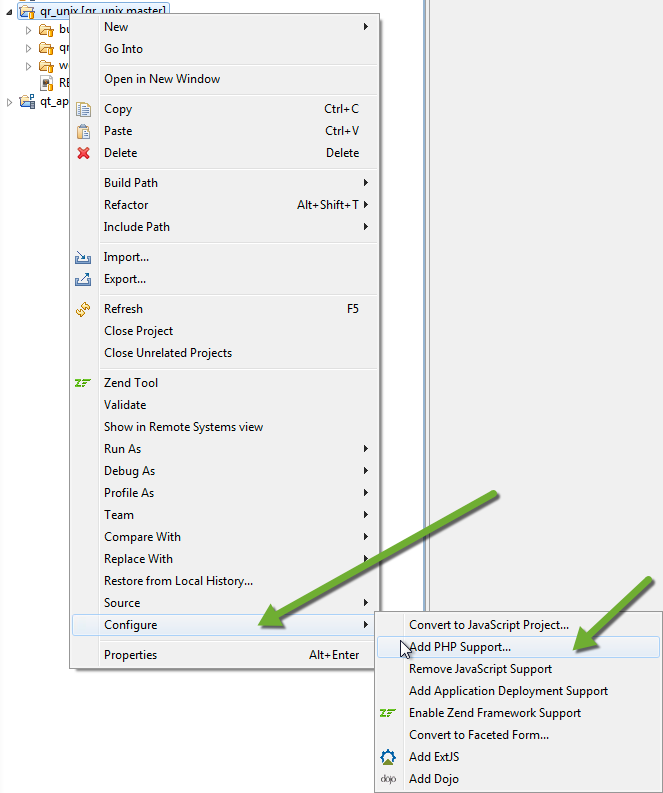
**Step Four:**

Setting up the imported project to auto-update changes to your DMS development port

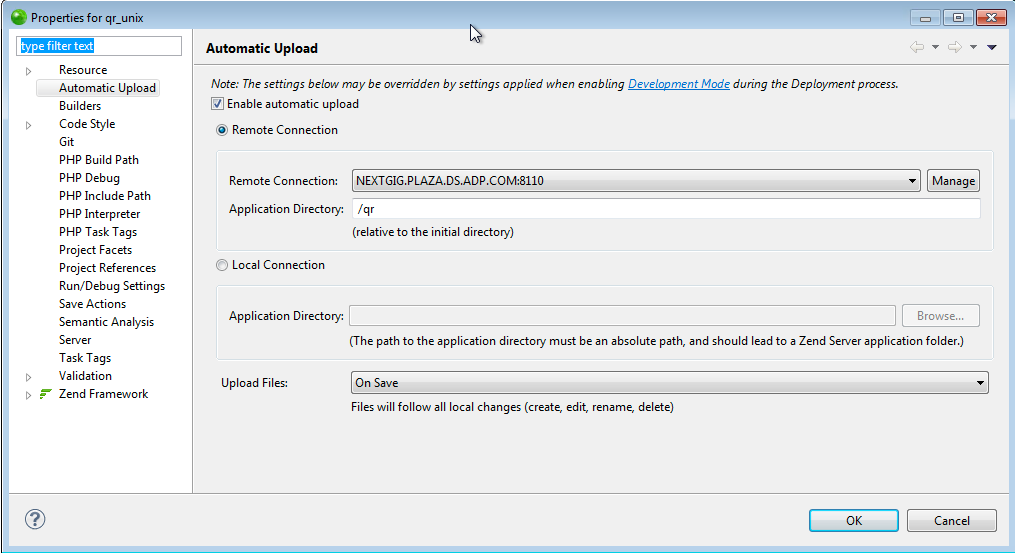
One last missing piece – updating your DMS development port get auto-updated with changes you make in Zend.

**Right Click** on the new project and ‘Configure’ and ‘Add PHP Support’ menus:



Now set the properties for this repo. Right click the repo and select properties. See the screen shot above for the right click menu options.

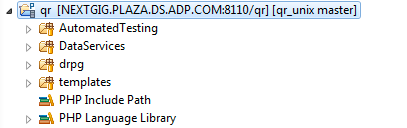
When you add PHP support, you also get ‘Automatic Upload’:



Check the ‘Enable automatic upload’ and then select the same ‘Remote Connection’ you were using before GIT came along. Take a look at what you are using today with your current setup.

In this example, the development server is NEXTGIG.PLAZA.DS.ADP.COM. The development port in the example is 8110 (your development port is a different number – 8110 is mine).

Once this is done, your Zend PHP explorer will look (something) like this:

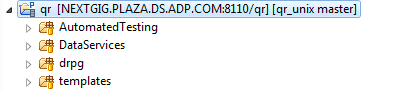


Congratulations – you can return to your normal development now!

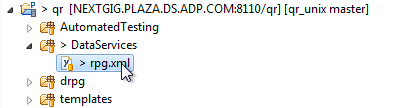
**Using Zend Studio and Source Tree:**

This section shows some examples of basic operations using the new tools

Before you make any changes to your source code, the PHP explorer looks something like this:



Once you begin to make changes to the source code, Zend Studio give you **‘>’** cues to show changes:



In the example above, the rpg.xml was updated, so we see the **‘>’** at that file and it’s parent directories. These visual cues will stay until we commit the change or discard the change.

A rough guide for folks coming from **Clear Case**:

* When you start to edit a file; that is about equal to **Clear Case** “Check Out”
* GIT “Discard Changes” is about equal to **Clear Case** “Uncheck Out”
* GIT “Stage & Commit” steps are about equal to **Clear Case** “Check In”

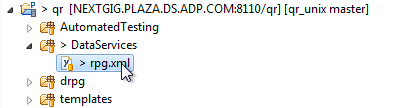
Some examples are in order.

**Example One:**

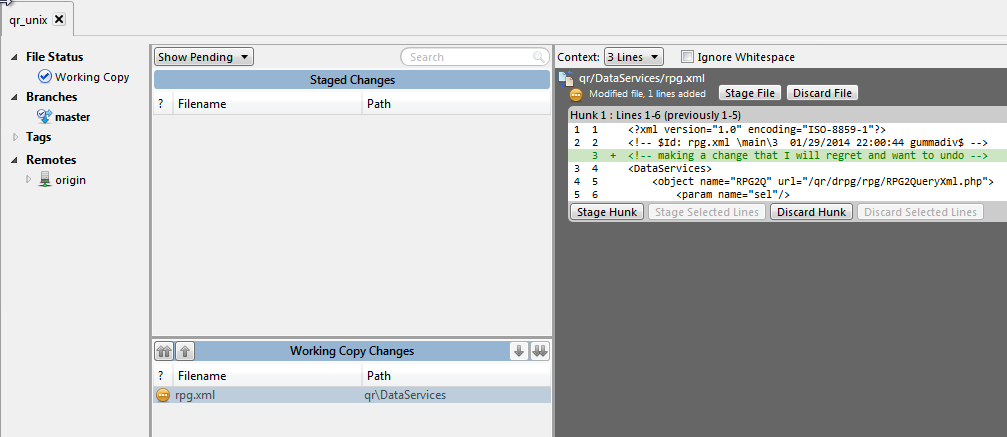
When you start to edit a file; that is about equal to **Clear Case** “Check Out”

I have edited the rpg.xml file using **Zend Studio** and saved the changes to disk.

In **Zend Studio** I see:



In Source Tree, the same change looks like this:



This operation is (about) the same as if I had done an ‘Unofficial Checkout’ in **Clear Case**. The biggest difference we didn’t go the Stash sever to do this. We just started editing our local copy.

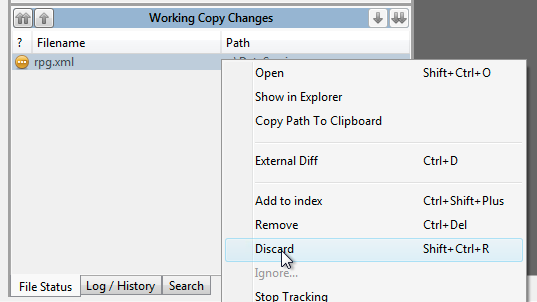
It is not a bad idea to do a ‘Pull’ using **Source Tree** to get the most recent source code before you start a large editing session.

**Example Two:**

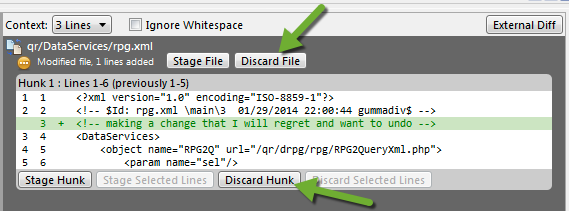
GIT “Discard Changes” are about equal to **Clear Case** “Uncheck Out”

I have edited the rpg.xml file and just realized this is the wrong file. I want to put it back the way it was before I messed with it. In **Clear Case**, we would do an “Uncheck Out”. In **Source Tree**, we do a ‘Discard Changes”.

You can right click on the ‘Working Copy’ and select ‘Discard’ to discard all changes in the file:



Or you can use the buttons in the diff window:



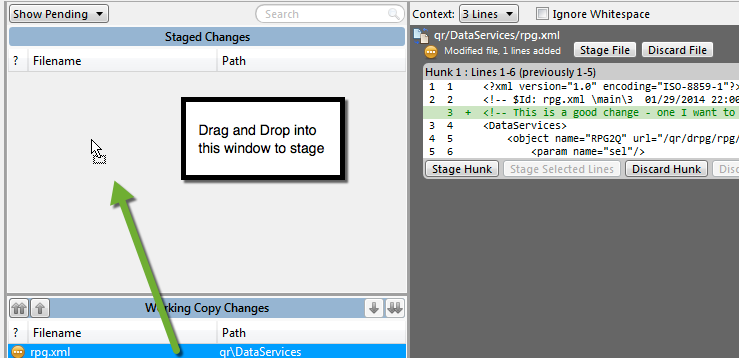
Note that you have two options in the diff window: To discard the changes to the ‘whole file’ here or just discard the ‘hunk of changes’ you are viewing. If you made two changes to a file and only want to keep one, the ‘Discard Hunk’ allows that.

**Example Three:**

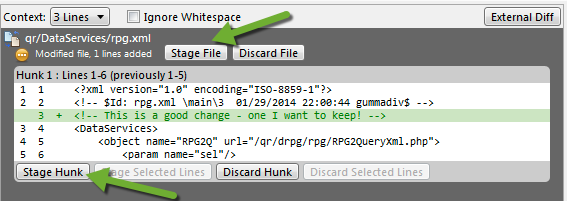
GIT “Stage & Commit” steps are about equal to **Clear Case** “Check In”

I have edited the rpg.xml file and tested it, so now I want to keep it. In **Clear Case** we would ‘Check In’ the file. Is **Source Tree**, we ‘Stage & Commit” the set of changes we want to keep.

To stage the working copy, you can drag the whole file into the ‘Staged Changes’ window:



The other option is to use the buttons in the diff window to stage the changes:



Notice we have two choices here – We can stage the ‘whole file’ or just stage ‘hunks of changes’. For most of us, we will always ‘Stage File’.

**General notes:**

* Any staged change can be ‘unstaged’ if you have not committed it yet.
* Once your changes are ‘committed’, they are still only on your local PC/VDI.
* To get the committed changes to Stash, you will need to do a ‘Pull’, and then a ‘Push’.
* Once you have done a ‘Pull’ & ‘Push’, the changes are in Stash and your pseudo ‘check in’ is really complete (others can now see and use your new code).