How to get the latest Z-Push code from GIT

This is intended as short how-to working with GIT to get access to the latest sources.

Install GIT

In order to execute any of the here described commands, you need the git binaries. On a linux host, the easiest is just intalling the git package, using:

```
apt-get install git
```

or

```
yum install git
```

You can also use GIT on Windows. Please have a look on the manual for windows or if the above method does not work: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git.

Get a single version - without cloning

Git works with a local copy of the repository, where you can easily switch (or update) between versions. Creating such a local copy is called *cloning.*

If you are only interested in a copy of the code (very similar to the one you get when extracting the tarball), you could simply do:

```
git clone -b 2.2.4beta https://stash.z-hub.io/scm/zp/z-push.git z-push/
```

This will get the code of the "2.2.4beta" to the `z-push` directory. The "2.2.4beta" is a *tag* which we create in git to mark a version. The tagname corresponds to the released versions.

Cloning the repository

To do more actions, you need to clone the repository to your local machine first. This creates a local copy and you can switch between version or also update much easier.

To clone, do:

```
git clone https://stash.z-hub.io/scm/zp/z-push.git /tmp/z-push-git
```

This will put the repository into `/tmp/z-push-git`. If you omit the folder at the end, it will create a `z-push` directory below your current location.

All following commands, you have to call from within this directory, so just `cd` into it.

Most common actions

Command	Description

`git status`	Shows information about the current status of your local repository. Most importantly, it shows which code version (branch) is currently available in the directory. Example:
	`On branch develop`
	`Your branch is up-to-date with 'origin/develop'.`

`git pull`	Gets the latest updates from the main online (also called `origin`) repository. This is similar to `svn update`.

`git tag`	Shows all available tags (versions). To update this list, you should do `git pull` first. Example:
	
	`2.2.3`
	`2.2.3beta`
	`2.2.4alpha`
	`2.2.4alpha1`
	`2.2.4beta1`

git checkout -b tag	Switches your workspace to the defined tag or branch and creates it localy. If you want to switch to it again later, omit the "-b" parameter.

Use cases

Below a list of a few use cases which could make it easier to use/update z-push on GIT basis.

Getting a certain version

You want to get the latest version and run on it. Just switch to your z-push git directory and do:

1. `git pull` to get the latest updates
2. `git tag` to get the latest tag name (version name) you are interested in
3. `git checkout -b` *`aTag`* to switch to that version.

The code in the local directory will not change until you do `git checkout` again.

Using this method you will also find alpha and beta versions which you can easily test (and do not forget to provide feedback).

Get updates for ...

Instead of fiddling around with tag names, you can just take one of the release branches and keep updating them with `git pull`.

Final versions

All versions considered "finals" are available in the "master" branch. There are no alphas or betas here, only final stuff.

You can just:

1. `git checkout master` to switch to the master branch.

That's it. To get updates, just do `git pull` and new versions are deployed automatically to your working directory. The code will change immediately after you do `git pull` (only if there are newer versions of course).

Alpha and Beta versions

These are available in the "release/X.X" branch.

Just:

1. `git checkout release/2.3` to switch to the release 2.3.x branch

This one contains 2.3.x alphas (for final QA'ing), betas and final releases.

Just do git pull to get the latest code.

You also could chose not to use e.g. alphas by **not pulling** until you see the beta announcement. The pull will immediately apply all changes. This is possible, but not very safe because you could accidentally pull and get an alpha version.

Latest development

There is also the "develop" branch, which contains everything available, independently from a version. This is basically the "nightly" build or the SVN trunk.

Code here should always work, but it also could not. I would not recommend using this on a productive system (unless for a few users that are aware of the risk that it may break).

Just:

1. `git checkout develop` to switch to the develop branch

You can use one or all of the described methods together. So, you can use `master`, but switch to `release/2.3` to test the current beta and then later switch back to master again. The only limitation to this might be local changes, e.g. to the config file which might lead to conflicts.