Git Flows

We use in Online Essentials
v0.2

Agenda

- Conventions
- Generic flow
- Default config
- Refresh your master
- Create new branch
- Work on the branch
- Share your stuff
- Test builds
- Return to master
- Clean up

Conventions

Branch names has to comply with the following conventions:

- lower case;
- we use dash instead of underscore;
- no naming intersection (e.g. feature-a and feature-aB);
- we use alphanumerics.

Default branches:

- master;
- dev;
- feature-name x n.

Generic flow

[USER STORY]

1. Pull from master.
2. Create dev from master.
* Create your branch from dev.
3. Commit your code.
4. Pull from remote branch.
5. Push your branch remote.
6. Make pull request.
7. Code review.
9. Pull accept / merge with master.
10. Clean up branches / recreate dev.

[BUG]

1. Commit your code on master.
2. Push your code on master.
3. Rebase dev.
4. Rebase feature branches.

Default repo config

```
[core]
     repositoryformatversion = 0
     filemode = true
     bare = false
     logallrefupdates = true
[remote "origin"]
     url = git@github.dtc.avira.com:<username>/<repo>.git
     fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
     remote = origin
     merge = refs/heads/master
```

Preparations

Mandatory:

```
# git config --global user.name "[name]"
# git config --global user.email "[email address]"
```

It is safe to:

```
# git config --global push.default current
# git config --global pull.default current
# git config --global color.ui auto
```

Refresh your master

Make sure you are on master:

```
# git checkout master
# git pull origin master
```

without the merge:

```
# git checkout master
# git fetch origin master
```

Create new branch

Create the branch:

```
# git branch feature-a
```

Then switch it on:

```
# git checkout feature-a
```

Or do everything in one step:

```
# git checkout -b feature-a
```

Work on the local branch

Commit your code:

```
# git checkout feature-a
# git commit -a -m ‘My comment’
```

Sync with dev if anything changed in dev branch:

```
# git rebase dev
```

Share your stuff

Make sure you are on the branch you want:

```
# git checkout feature-a
# git commit -a -m ‘Comments’
```

If you have your `push.default` setup to `current`:

```
# git push
```

Or else you should do it like that:

```
# git push origin feature-a
```

Use the amend feature to update your last commit:

```
# git commit -a --amend
```

Share _some_of_ your stuff

Use patch to commit only some chunks of your work, and not everything you changed:

```
# git commit --patch
```

1. Press `‘s’` to split into chunks.
2. Press `‘y’` to agree to commit a specific chunk.
3. Press `‘d’` when you are done.

Make sure you put a comment into your git commit.

Get new stuff

To get a new branch that has been pushed remotely:

```
# git fetch --prune origin
# git branch -r
# git checkout -b feature-a origin/feature-a
```

Or:

```
# git checkout feature-a
# git branch --track feature-a origin/feature-a
```

Refresh the branch:

```
# git checkout feature-a
# git pull origin feature-a
```

Tests multiple features

Make sure you are on dev branch:

```
# git checkout dev
```

Merge your branch:

```
# git merge feature-a
```

Push the the local dev branch on the remote dev branch:

```
# git push origin dev
```

* If specific multiple features need to be tested, then a dedicated branch can be created to collect them and make the build from that branch.

Return to master

Switch to master:

```
# git checkout master
# git merge dev
```

Or better initiate a GitHub pull request. Code review included.

Clean up

Delete the old branch:

```
# git branch -d feature-a
# git push origin --delete  feature-a
```

Recreate the dev:

```
# git checkout dev
# git reset --hard origin/master
# git push origin dev --force
```

Rollback remote commits

Make sure you are on the desired branch

```
# git checkout feature-a
```

Find out what is the wanted commit ID:

```
# git log
```

And reset to it (you cannot undo a hard reset):

```
# git reset --hard <commit_id>
```

Delete the remote branch and basically recreate it:

```
# git push origin --delete feature-a
# git push origin feature-a
```

Bug fixing

Make changes and push them:

```
# git checkout master
# git commit -a -m ‘My bug fix’
# git push origin master
```

Resync dev with master:

```
# git checkout dev
# git rebase master
```

References

https://na1.salesforce.com/help/pdfs/en/salesforce_git_developer_cheatsheet.pdf

http://byte.kde.org/~zrusin/git/git-cheat-sheet-medium.png

http://nvie.com/posts/a-successful-git-branching-model/