Design Patterns for Git

Lorna Mitchell, IBM
Velocity Amsterdam 2016
Well-Organised Repositories
Git Branching
A Branch is a Label

- A branch is a pointer to a commit.
- The pointer moves along when you commit on a branch.

```
[master]$ git rev-parse master
ba8969833794d14abb218f8532edb5fc4ce9ab

[master]$ cat .git/refs/heads/master
ba8969833794d14abb218f8532edb5fc4ce9ab
```
Commits Aren’t 'On' Branches
Commits Aren’t 'On' Branches

@lornajane
Pro-tip: The reflog Command

If you move a branch pointer and then regret it:

```bash
git reflog
```

Will show every revision your HEAD pointer has visited

To rescue a "lost" revision check it out and then create a new branch
Branching Strategies
Branching Strategies

A living, changing process document.
Git Flow

Comprehensive branching strategy, supported by scripts.

- Article: http://lrnja.net/1ebawKU
- Scripts: https://github.com/nvie/gitflow
Alternatively: Github Flow

Use with feature toggles
Topic Branches Pattern

Starting something new? Use a branch.
Topic Branches Pattern

Rebase branch onto master, review and merge

@lornajane
Branch Naming

- master  default branch name
- develop  often used for bleeding edge branch

Configure your repo accordingly
Environment Branches

Maintain one branch per platform

• branch always reflects current state of platform
• plays well with automated deployment
• non-live branches can be disposed of at any time
Environment Branches

features

develop

release

master
Hotfix Pattern
Hotfix Pattern

Branch off, commit once, merge with --no-ff

• makes it very clear what happened
• easily merge the branch back upstream if needed
• makes it possible to cleanly undo without losing anything
Tagging for Releases

Use annotated tags and attach a message

git tag -a v2.11 -m "v2.11 with caching"
One Repo Per Person?

Normal for open source, rare for commercial teams
One Repo Per Deployable

One codebase tracked in revision control, many deploys

https://12factor.net/
One Repo Per Deployable

- Keep projects independent
- Separate ownership allows separate evolution
- Different components can use different workflows
Working With Many Repos

Think about how to manage upgrades

- It's hard to deploy two components at the same time reliably
- Can safely add features
- Breaking changes may require a new version number
Build Every Platform The Same Way

Keep development, staging, and production as similar as possible https://12factor.net/
Build Every Platform The Same Way

Keeping platforms the same:

- Use the same config mechanisms
- Check for required services the same way
- Use the same dependency management
Managing Internal Dependencies

Explicitly declare and isolate dependencies

https://12factor.net/
Managing Internal Dependencies

Your options are:

• Keep everything in one repo
• Use git's submodules or subtrees to organise dependencies
• Bring in dependencies at build time
Both approaches allow keeping code in a separate repo

- submodules put a nested repo into a subdirectory
- subtree adds part of another repo as a first-class citizen of your current repo
Surviving Submodules

What do you want to do?

- Clone repo
- Receive upstream changes
- Push changes
Surviving Submodules

Clone repo
- git clone --recursive git://...

Receive upstream changes
- // first check that you have no local changes
- git pull
- git submodule update
Surviving Submodules

1. Push changes
   - Are there changes in the submodule? (Yes or No)
   - If No, commit and push as normal
     // DO NOT `git add`
   - If Yes, Did you put them there? (Yes or No)
     - If No, git submodule update

Surviving Submodules

Did you put them there?

// cd into (each) submodule
git add -p
git commit
git push
// return to root of project
// add and commit changes in this repo
// now add and commit submodule changes

git push
Manage Private Dependencies

With npm:

```
"package-name": "git+ssh://git@github.com/<user>/<repo>.git"
```

With composer:

```
"repositories": [  
  {  "type": "vcs",  
    "url": "git@bitbucket.org:vendor/my-private-repo.git" } 
]`
Well-Organised Repositories
Resources

- These slides: http://lornajane.net/resources
- Twitter: @lornajane
- Email: lorna.mitchell@uk.ibm.com