Git workflow for infrastructure-puppet repo

Who this workflow is for

This page describes the workflow for those infra peeps with git repo commit access.

If you do not have commit access, please Fork our repo on Github and issue a Pull Request - we will be notified and respond to PRs in a timely manner. Should PRs not be resolved within a reasonable timeframe, please open a Jira ticket and link it to the PR. In general PRs are triaged the same as Jira, in order of importance.

For Infra folks with commit bit - the current workflow to use goes something like this.

**DO NOTE: Deployment branch is currently locked, only root can push to it!**

**Working on the infrastructure-puppet repo**

```bash
  git clone https://gitbox.apache.org/repos/asf/infrastructure-puppet.git if you haven't already.
```

Since this repo is on Gitbox, you can push to Gitbox or Github, but it's best to choose one and stick to it (split brain is a hell of a thing)

Ensure you have deployment branch up-to date.

```bash
  git checkout deployment
  git pull
```

**DO NOT** use master. It is not used any more and changes pushed to master will be ignored.

**work on your branch**

```bash
  git checkout -b $yourbranch
```

*(do some amazing coding)*

Do some lint checking:

```bash
  bundle exec puppet-lint modules/ [1]
```

```bash
  git add .
  git commit [-S] -m "changes to my branch" (Optionally consider using -S to have your commit(s) verified [2])
  git push origin $yourbranch # ALWAYS specify what branch you're pushing to!
```

*(ask an active infra member to approve (+1) your branch and either they will merge it or you can self-merge quoting the +1 and its author in the log message.)*

**Pull requests**

once you've made a branch, feel free to just make a pull request on github https://github.com/apache/infrastructure-puppet/pull/new/deployment

Github is pretty smart and will recognize branches you've made to select against

This is can be a better method for merging large changes that require discussion or are more asynchronous across timezones/ etc

**Merge your own branch in case of emergency**

*Important: This means having to merge without a +1 from anyone else - i.e. nobody has reviewed your branch and nobody is around to do so and that the branch is in need of immediate merge, i.e. some service is down and needs it. LEAVE A NOTE IN THE COMMIT MESSAGE*

```bash
  git pull
  git checkout deployment
```

Do some lint checking:

```bash
  bundle exec puppet-lint modules/ [1]
  git merge origin/$yourbranch # will provide you an opportunity for a commit message
  git push origin deployment # ALWAYS specify what branch you're pushing to!
```

**Merging others branches (approve + merge) (default workflow)**
git pull

Do some lint checking:

bundle exec puppet-lint modules/ [1]

git checkout deployment

git merge origin/$theirbranch # may provide you an opportunity for a commit message

git push origin deployment # ALWAYS specify what branch you're pushing to!

Keeping your branch in sync with deployment (a.k.a rebase)

git pull

git checkout $mybranch

git rebase deployment

( fix any merge conflicts by using git status for what needs to be done and asking those in #asfinfra)

git push origin $mybranch # ALWAYS specify what branch you're pushing to!

Merging Conflicts

something like (whilst on your branch) :

git rebase deployment

(edit and resolve conflicts, the stuff between <<<<<<<< HEAD and >>>>>>> $branch)

git add $conflicted_file(s)

git rebase --continue

Cleaning up after yourself (deleting your merged branch)

(delete branch locally after merge)

git branch -d $branch (it may complain, and if so, use -D)

(delete branch on remote)

git push origin :$branch

Links to external resources

Resolving merge conflicts in many more situations

More tips on cleaning up branches etc :-


Footnotes

Sign your commits with your GPG key

Consider using `git commit -S -m "blat"` when committing. That is use -S in conjunction with your GPG key to have your commits verified.

- Add `signingkey = YOURKEY` to your .gitconfig file (example Daniel Takamori would use `signingkey = E2BA6F3F`
- Add your GPG key to your Github Settings
- Make your commits with the -S flag

To make the setting permanant per repo run this command in your repo:

`git config commit.gpgsign true`

no need to remember the -S switch then.

For more info see: [this GitHub article on GPG signing your commits](http://railsware.com/blog/2014/08/11/git-housekeeping-tutorial-clean-up-outdated-branches-in-local-and-remote-repositories/)

Prepare for lint checking

From root of puppet repo on deployment branch, run

gem install bundler

From root of puppet repo on deployment branch, run

bundle install

To check again puppet-lint, run

bundle exec puppet-lint modules/

For finer grained lint checking ignoring stuff we can’t do much about currently use (and tweak as needed)

Lint checks with some ignore args

bundle exec puppet-lint --no-nested_classes_or_defines-check --no-autoloader_layout-check modules/

lint for the yaml files

gem install yaml-lint
yaml-lint data/nodes/name-of.yaml
yaml-lint data/nodes (leave out filename to check them all)

Other tips:-

In Vim to remove all trailing whitespaces from all lines in a file:-

:%s/\s\+$//

Recently upgraded your Mac?

Having just upgraded from Sierra to High Sierra - to be able to edit yaml again and other stuff I had to:-

Re-install some stuff

gem install bundler
# cd to your infrastructure-puppet git checkout
bundle install
xcode-select --install
# optional:
gem install ruby_gpg

ruby_gpg is possibly optional, try without it first if you like.

Now, your should have a back to fully working setup. (If not, add here what else you had to do!)