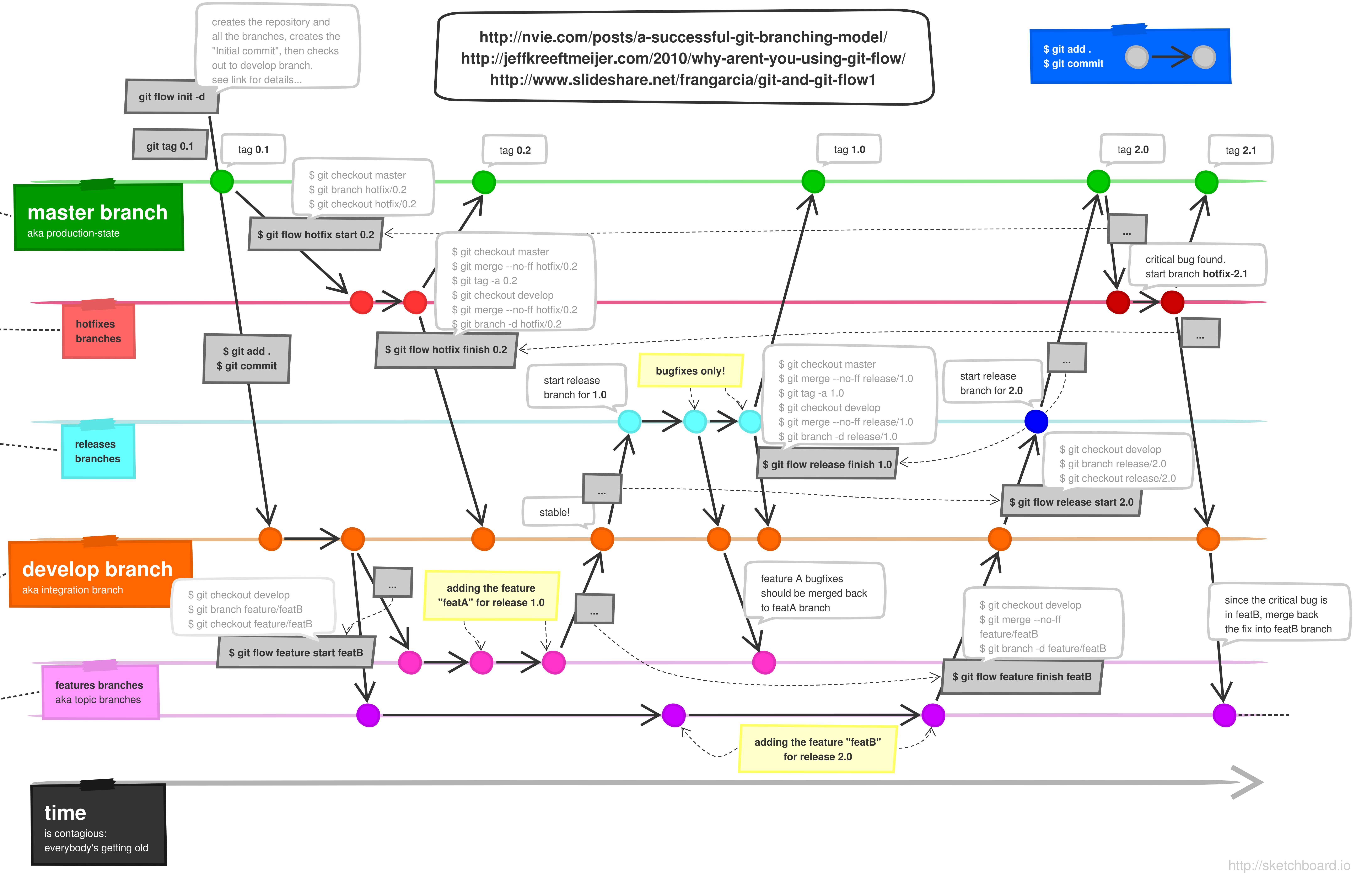
master branch
aka production-state
hotfixes branches
releases branches
develop branch
aka integration branch
features branches
aka topic branches
time
is contagious: everybody's getting old
tag 0.1
tag 0.2
tag 1.0
tag 2.0
tag 2.1
...
stable!
bugfixes only!
start release branch for 1.0
start release branch for 2.0
critical bug found. start branch hotfix-2.1
adding the feature "featA" for release 1.0
adding the feature "featB" for release 2.0
feature A bugfixes should be merged back to featA branch
since the critical bug is in featB, merge back the fix into featB branch
creates the repository and all the branches, creates the "Initial commit", then checks out to develop branch. see link for details...
http://sketchboard.io

http://nvie.com/posts/a-successful-git-branching-model/
http://jeffkreeftmeijer.com/2010/why-arent-you-using-git-flow/
http://www.slideshare.net/frangarcia/git-and-git-flow1

```
$ git add .
$ git commit
```

```
git flow init -d
```

```
git tag 0.1
```

```
$ git checkout master
$ git branch hotfix/0.2
$ git checkout hotfix/0.2
```

```
$ git flow hotfix start 0.2
```

```
$ git checkout master
$ git merge --no-ff hotfix/0.2
$ git tag -a 0.2
$ git checkout develop
$ git merge --no-ff hotfix/0.2
$ git branch -d hotfix/0.2
```

```
$ git flow hotfix finish 0.2
```

```
$ git add .
$ git commit
```

```
$ git checkout master
$ git merge --no-ff release/1.0
$ git tag -a 1.0
$ git checkout develop
$ git merge --no-ff release/1.0
$ git branch -d release/1.0
```

```
$ git flow release finish 1.0
```

```
$ git checkout develop
$ git branch release/2.0
$ git checkout release/2.0
```

```
$ git flow release start 2.0
```

```
$ git checkout develop
$ git branch feature/featB
$ git checkout feature/featB
```

```
$ git flow feature start featB
```

```
$ git checkout develop
$ git merge --no-ff feature/featB
$ git branch -d feature/featB
```

```
$ git flow feature finish featB
```

We consider origin/master to be the main branch where the source code of HEAD always reflects a production-ready state.

Therefore, each time when changes are merged back into master, this is a new production release by definition. We tend to be very strict at this, so that theoretically, we could use a Git hook script to automatically build and roll-out our software to our production servers everytime there was a commit on master.

May branch off from **master**
Must merge back into **develop and master**
Branch naming convention: hotfix-*

Hotfix branches arise from the necessity to act immediately upon an undesired state of a live production version. When a critical bug in a production version must be resolved immediately, a hotfix branch may be branched off from the corresponding tag on the master branch that marks the production version.

May branch off from **develop**
Must merge back into **develop** and **master**
Branch naming convention: release-*

Release branches support preparation of a new production release. They allow for last-minute dotting of i's and crossing t's. Furthermore, they allow for minor bug fixes.

The key moment to make a new release branch is when develop (almost) reflects the desired state of the new release. It is exactly at the start of a release branch that the upcoming release gets assigned a version number.

We consider origin/develop to be the main branch where the source code of HEAD always reflects a state with the latest delivered development changes for the next release.

When the source code in the develop branch reaches a stable point and is ready to be released, all of the changes should be merged back into master somehow and then tagged with a release number.

May branch off from **develop**
Must merge back into **develop**
Branch naming convention: anything except master, develop, release-*, or hotfix-*

Feature branches are used to develop new features for the upcoming or a distant future release.

The essence of a feature branch is that it exists as long as the feature is in development, but will eventually be merged back into develop (to definitely add the new feature to the upcoming release) or discarded (in case of a disappointing experiment).

Feature branches typically exist in developer repos only, not in origin.