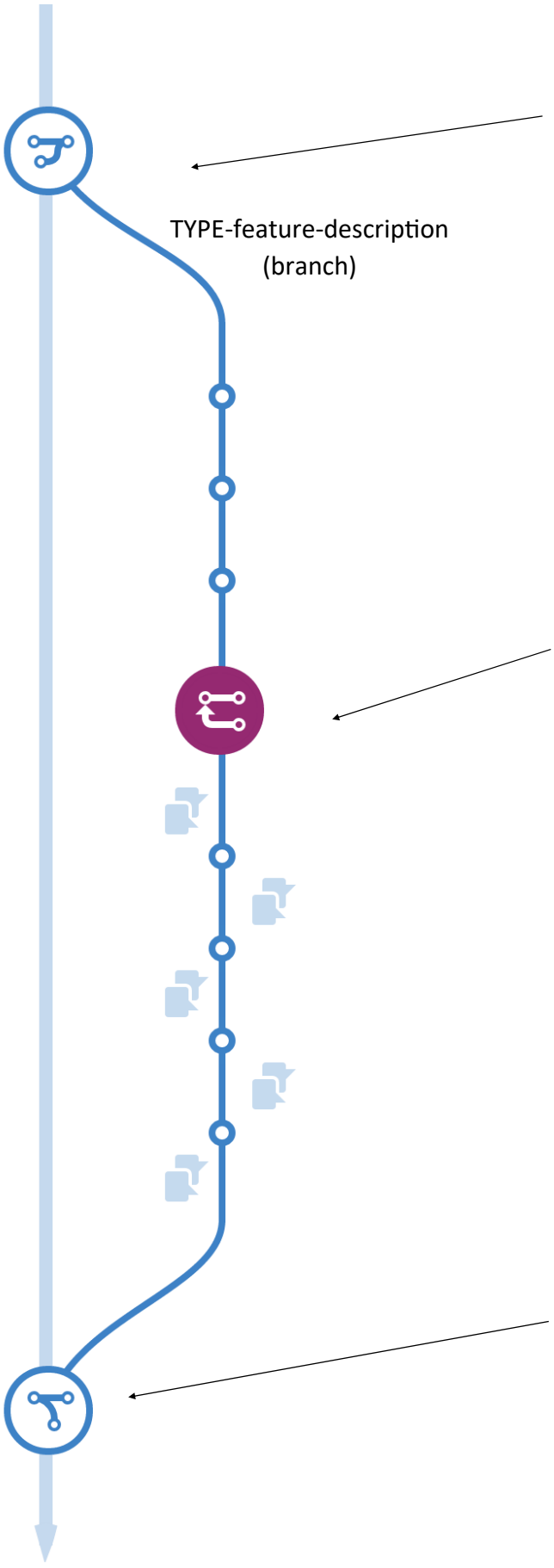
GitHub Flow for DPL

v1.0.1



1. Create new branch

1. Checkout master
2. Pull (to get latest changes)
3. Create a new branch (`TYPE-feature-description`)

2. Build the feature

1. `CONTRIBUTING.md` instructions
2. **Acceptance criteria (feature specifics)**
3. **Definition of done**
4. Update Trello card comments
5. Test in browsers and devices
6. (Update `CHANGELOG.md`)

3. Commit to Git

1. Commit little and often
2. Commit message style guide

4. Open a pull request

1. Commit message style guide
2. Push to GitHub
3. Update Trello card with link to GitHub pull request. Move card to QA.

5. QA code

1. Checkout branch
2. Build (`grunt`)
3. Review code and documentation against acceptance criteria and definition of done
4. Edit / commit / push to GitHub
5. Test in production environment.

6. Deploy

1. Bump version number (semver.org) in `package.json`.
2. Update `CHANGELOG.md`
3. Build (`grunt`)
4. Commit / push to GitHub
5. Deploy `/core` to `/cdn/x.y.z`
6. Deploy `/docs` to `/dpl/x.y.z`

7. Merge

1. Merge branch into master
   (See commit message style guide)
   `MERGE (Feature) Description`

8. Test master and delete branch

1. Checkout master
2. Build (`grunt`)
3. If it works as expected, delete branches (local and remote)
4. Tag as version number and push to GitHub

Commit message style guide cheat sheet

A well-formed commit message SHOULD look something like this:

```
TYPE (Component) Brief summary

Longer description with more details, such as a `code` being introduced

within the context of a `function()`.

Multiple paragraphs may be added as required.

Bulleted lists:

* are

* also

* okay

Closes #123
```

TYPE	Description
BREAK	A breaking change such as removing a feature.
DOCS	Changes to the documentation (readme, API docs, etc.).
FEAT	New feature in production code.
FIX	Bug fix in production code.
FORMAT	Code formatting, code comment change, etc; (compiled-code neutral).
MAINT	Updating dev-related maintenance ("non-production code") files.
MERGE	Pull request to merge branch into another branch or master.
TEST	Adding missing tests or editing tests.

Definition of done

The following definition of done has been agreed by the team. A card may not pass from QA to Done without fulfilling all of the following criteria:

1. Code must be **version controlled**, with each feature in its own branch, and a pull request created to merge into master.
2. Must meet **acceptance criteria** for the feature.
3. Must be **accessible**, including using WAI-ARIA landmark roles.
4. Must adhere to **code standards and style guide**.
5. Must include **print CSS rules**.
6. Code must be **well-commented** to explain why it has been done in a particular way not what it does -- that can be gleaned from the code itself: prioritise the why over the what.
7. Documentation needs to comply with **house style** and **writing for the web** guidelines.
8. **Drop the 'related patterns' section** of each pattern as the new interface for categorising patterns will make this redundant.
9. Need to convey **how a pattern relates to another** e.g. breadcrumb pattern must only be used after a navigation pattern.
10. **package.json** must be updated
11. **CHANGELOG.md** must be updated.
12. **Merge to master** may only take place after a peer code review, and deployment to live environment for testing.
13. **Deploy** feature to live.
14. Master has been **tagged** in Git, and pushed to GitHub.
15. Feature **branch may only be deleted** after code has been merged to master, checked out, pulled, and built successfully without error.