# Best Practices Workflows

## 1. Concepting

Concepting a development task contains three important steps

- *Draft* - Drafting  Issues, User Stories and Tasks
  The key objective of Draft, is to express the goal while affording team members the opportunity to interpret the task according to their expertise.

- *Estimate*- Estimating a workload is a collaborative effort to complete a task, helps you clarify the objective to ensure that estimation is well understood by everyone

- *Assign*- a development task should be assigned to a developer or QA to test, this helps to decide how to accomplish the task, and are, therefore, best suited to
  decide who should take care of it.

## 2. Branching

Git branching workflow is based on two main branches

- **master** — this branch contains production code. All development code is merged into master sometime.
- **develop** — this branch contains pre-production code. When the features are finished then they are merged into develop.

During the development cycle, a variety of supporting branches are used:

- **feature-*** — feature branches are used to develop new features for the upcoming releases. May branch off from develop and must merge into develop.
- **hotfix-*** — hotfix branches are necessary to act immediately upon an undesired status of master. May branch off from master and must merge into master and develop.
- **release-* —** release branches support preparation of a new production release. They allow many minor bugs to be fixed and preparation of meta-data for a release. May branch off from develop and must merge into master and develop.

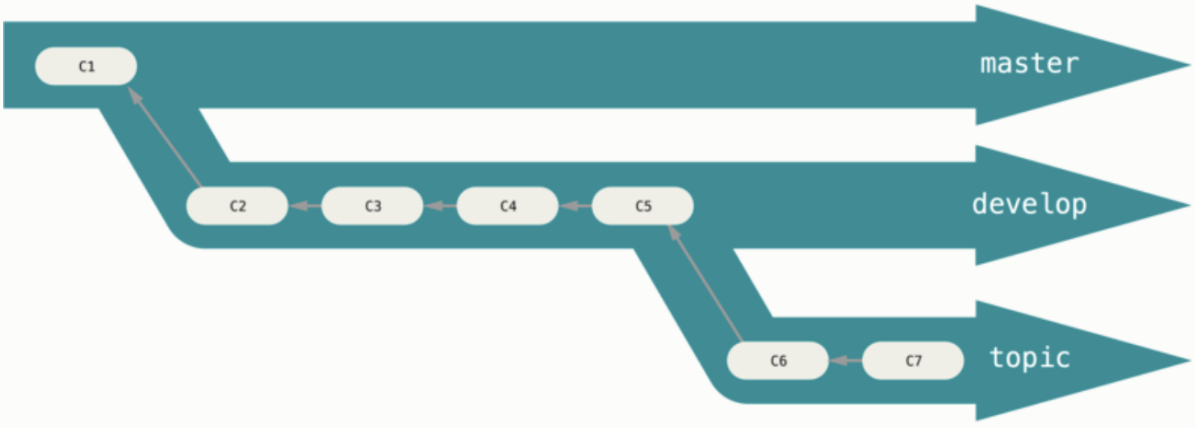
## 3. Git Workflows

Many Git developers have a workflow that embraces this approach, such as having only code that is entirely stable in their master branch — possibly only code that has been or will be released. They have another parallel branch named develop or next that they work from or use to test stability — it isn't necessarily always stable, but whenever it gets to a stable state, it can be merged into master. It's used to pull in topic branches, when they're ready, to make sure they pass all the tests and don't introduce bugs.

In reality, we're talking about pointers moving up the line of commits you're making. The stable branches are farther down the line in your commit history, and the bleeding-edge branches are farther up the history.


Figure 26. A linear view of progressive-stability branching

It's generally easier to think about them as work silos, where sets of commits graduate to a more stable silo when they're fully tested.



## 4. DTAP

DTAP is a common approach for testing software in phases. It stands for development, testing, acceptance, and production (DTAP).

This progression is sometimes referred to as a street, or a DTAP-street. With this approach, the cohort of testers is progressively increased until it includes the complete user base

## 5. Manual Testing

It is most common during the development process, where testing is executed according to a list of test cases. The output of which is a report on the state of the software.

## 6. Automation Testing

It is carried out by software tools that run the tests automatically. Automation is typically the result of iterations of manual testing that yield enough insights to begin automating the process and complete the QA pipeline in less time. This is especially important as the production environment grows in scope.