Git Workflow and Versioning Plan

The process described in this document is a modified version of the Gitflow Workflow.

With a standard Gitflow Workflow, the features that complete Tracker stories are pushed to a Develop branch. When enough features have accumulated in the Develop branch, or a predetermined release date approaches, a Release branch is forked off of Develop, and no more features are added to the Release. Release is then merged into Master.

This standard Gitflow Workflow works well for developers, but our needs in documentation are slightly different. We’ve modified the process to meet two primary needs:

1. We want to integrate updated documentation to Master daily, or even more frequently.

With our workflow, documents are merged into the Master branch as soon as they are delivered. This lets us publish new and updated documents as fast as we can write them, instead of waiting for a Release.

1. We want to keep information about unreleased features isolated from the rest of our documentation.

With our workflow described here, any documents that complete Tracker stories for unreleased features are not merged into the Master branch. Instead, they are merged into a Pre-Release branch, and are not merged to Master until we are ready for a Release.

### Terminology

In the following descriptions, “Master” consists of the master branch of a book repo and the master branch of every repo in the book.

“Version” refers to any one of a number of separate branches of a book and the repos in the book. For example, the v1.6 Version branch consists of the v1.6 branch of a book repo and the v1.6 branch of every repo in the book.

## GitHub Branches

### Main Branches

1. **Master**: The Master branch stores the official release history and serves as an integration branch for stories. When a writer delivers a Standard story, they merge the documents that satisfy the story into the Master branch.

1. **Pre-release:** The isolated branch, Pre-release, keeps documents containing information about unreleased features isolated from the rest of our documents.

The pre-release branch contains all Pre-release story documents. The Pre-release branch stays completely separate from the Master branch until a Release. During a Release, the Pre-release branch merges with the Master branch.

The PM forks the Master branch to create the Pre-release branch when the first Pre-release story is prioritized. A Pre-release story contains material about a feature that does not exist in current versions of our software, and that we therefore cannot yet publish.

1. **Version:** Contain older documentation. There will be a version branch for each older version of our software.

### Story Branches

When we move a Tracker story into our backlog, we must put it into one of three categories. Work on each of these types of stories is done in a branch of the corresponding type. For example, a writer works on a Standard story in a Standard Story branch.

The types of Stories and Story Branches are as follows:

* **Standard stories**: Standard stories contain material about features that exist in the current, released version of our software. Most of the stories worked on by the CF Docs team are Standard stories.

**Standard Story branches**: Writers create a new branch for each new Standard story, and push this branch to the central repository for backup and collaboration. These Standard story branches use Master as their parent branch. When a Standard story is delivered, the writer merges the branch back into Master.

* **Pre-release stories**: Pre-release stories contain material about features that do not yet exist in released versions of our software. Until the new version of our software is released, we cannot publish the documents resulting from these stories to production. To guard against accidents, this new process keeps documents from Pre-release stories from ever being merged into the Master.

**Pre-release Story branches**: Writers create a new branch for each new Pre-release story, and push this branch to the central repository for backup and collaboration. These Pre-release story branches use the Pre-release branch as their parent branch. When a Pre-release story is delivered, the writer merges the branch into the Pre-release branch. Pre-release Story branches never interact directly with the Master branch.

* **Version Bug Fix stories**: Version Bug Fix stories contain information that corrects something in an older version of our software. Version Bug Fix stories are made on Version branches only.

**Version Bug Fix Story branches**: Writers create a new branch for each Version Bug Fix story, and push this branch to the central repository for backup and collaboration. These Version Bug Fix story branches use a Version branch as their parent branch. When a Version Bug Fix story is delivered, the writer merges the branch into a Version branch. Version Bug Fix story branches never interact directly with Master.

## Web Sites and Concourse Pipelines

**Staging Site**: Builds from the Master branch, just like the existing Staging Site, but also includes subdirectories, each built from a Version branch. Editor acceptance for Standard stories occurs on the Staging Site, and acceptance for Version Bug Fix stories occurs on the appropriate version subdirectory on the Staging Site.

**Production Site**: Builds from the last green Staging Site build. Manually triggered.

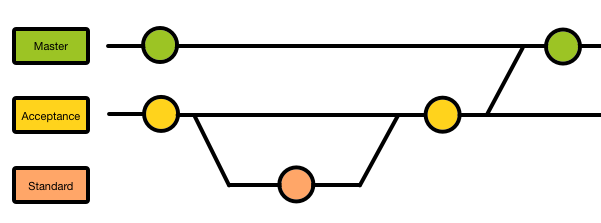
****

**Pre-release Site**: Builds from the Pre-release branch. Triggered by any changes to the Pre-release branch. Editor acceptance for Pre-release stories occurs on the Pre-release Site.



## Workflows

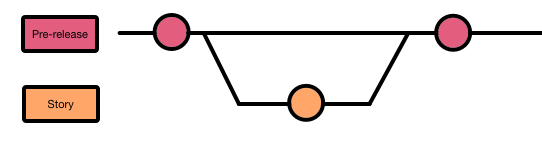
### Standard Story

The diagram shows the branch workflow when a writer works on a Standard story.

#### Standard Story Activity Sequence

1. Writer picks up story from Tracker.
2. Writer creates a Standard Story branch off of the Master branch.
3. Writer creates or modifies documents to meet the requirements of the story.
4. Writer merges the Standard Story branch back into the Master branch.
5. Writer add links to the new or modified documents in the story in Tracker.
6. Writer delivers the story in Tracker.
7. Editor reviews the delivered documents.
   1. If editor rejects the story, writer goes back to step 3.
8. Editor accepts the story.

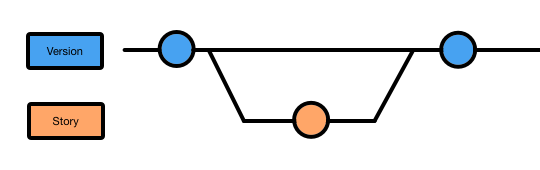
### Pre-release Story



#### Pre-release Story Activity Sequence

1. Writer picks up story from Tracker.
2. Writer creates a Pre-release Story branch off of the Pre-release branch.
3. Writer creates or modifies documents to meet the requirements of the story.
4. Writer merges the Pre-release Story branch back into the Pre-release branch.
5. Writer add links to the new or modified documents in the story in Tracker.
6. Writer delivers the story in Tracker.
7. Editor reviews the delivered documents.
   1. If editor rejects the story, writer goes back to step 3.
8. Editor accepts the story.

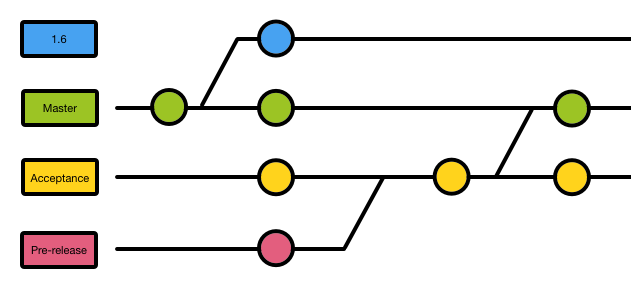
### Version Bug Fix Story



#### Version Bug Fix Story Activity Sequence

1. Writer picks up story from Tracker.
2. Writer creates a Version Bug Fix Story branch off of a Version branch. The specific Version branch depends on which document version needs to be modified.
3. Writer creates or modifies documents to meet the requirements of the story.
4. Writer merges the Version Bug Fix Story branch back into the parent Version branch.
5. Writer add links to the new or modified documents in the story in Tracker.
6. Writer delivers the story in Tracker.
7. Editor reviews the delivered documents.
   1. If editor rejects the story, writer goes back to step 3.
8. Editor accepts the story.

### The Release



#### Release Activity Sequence

When we are ready to release a new version of our software, the followings steps take place:

1. PM forks the Master branch to create a Version branch. The name of the Version branch is one earlier than the new software release version. For example, if we release PCF 1.7, the Version branch should be named **v1.6**.
2. PM makes changes to Concourse, config.yml, and credentials to publish the Version branch to a subdirectory of the Staging Site.
3. PM merges the Pre-release branch into the Master branch. The Master branch now contains documentation for the new release and features.