Firsthand Platform

Platform Configuration and Deployment Procedures

2017-05-01

Version **1.0**

Revision History

Date	Version	Description	Author
2017-05-01	1.0	Initial revision.	Craig Leinoff

Table of Contents

1 Definitions

The Application, see **Platform Software**.

Application Data is data that is stored in the Application's database, containing mostly business data about Users, their availability within the Platform, the consultations they have or will hold on the Platform, and some non-technical customization data for clients, such as branding, terminology, and other discrete, non-technical data.

A **Branch** is a duplication of the Codebase under Version Control so that changes to code can exist in parallel to the changes committed to other Branches.

The Code, or **The Codebase** specifically refers only to the Platform Software written or introduced by Firsthand Staff as part of the primary server-side application logic, front-end logic transmitted from our platform to be executed client-side, associated libraries referenced by the Application.

A **Commit** is an atomic set of changes to files under Version Control in the Codebase to be stored and distributed to other Engineers.

Continuous Integration is the cycle espoused by Firsthand for the Application, wherein code is frequently committed and merged into the Codebase, and checked against a battery of predefined tests and expectations as specified by Firsthand Engineering Staff.

Engineering, or **Firsthand Engineering Staff** is any employee or authorized contractor working in development, deployment, maintenance, or any related capacity in service of the Firsthand Platform.

The **Firsthand Platform**, or **The Platform** is the software-as-a-service product made available to Firsthand clients under the designation "Alumni Mentorship Platform", "Career Advisor Platform", "Webinar-Only Platform", and any future products utilizing the same codebase.

The Platform Software, is the resultant combination of the software side of the Firsthand Platform, composed of the primary server-side application logic, front-end logic transmitted from our platform to be executed client-side, associated libraries referenced by the Application, and any unrelated software executed in service of the application (search servers; regularly scheduled event scripts; deployment scripts) created or maintained by Firsthand Engineering Staff.

Procedures are the specific policies outlined in Section 5 of this document, outlined and followed in accordance with the goals stated in Section 2.

A **Pull Request** is a method of submitting contributions to the Codebase. It packages up a series of Commits on a single Branch, and allows them to be easily reviewed and merged into another branch by an Engineer.

Quality Assurance Team, or **QA Team** refers to, when possible, a set of authorized, third-party contractors specializing in providing professional testing of software applications as a service, specializing in functional testing, regression testing, and light security testing. In the event where a specializing third-party QA team is unavailable, other Firsthand staff may be deputized to provide light QA testing in a similar capacity. This term may refer to either the external or internal team participating in QA work.

Search Index refers to a normalized data store, separate from the Application's database, containing data that has been processed (via tokenization, stemming, filtering, etc.). Its primary purpose is to provide a search application with (a) a means of quickly finding relevant documents by term and (b) storing data to be returned when a matching document is found.

Staff, refers to **Firsthand Engineering Staff** specifically.

The **Version Control System** or **VCS** refers to a software-based system for tracking and coordinating changes in computer files in service of The Codebase by use of Engineering.

2 Introduction

This is the Configuration and Deployment Plan for the Firsthand Platform. It defines the procedures by which Firsthand's Engineering Staff prepares and releases software changes to the platform, and outlines standards for modifying and maintaining the server/networking hardware required for proper functioning of the Platform.

The Procedures outlined herein are defined as such to facilitate the consistent, efficient, effective, safe, and rapid development and deployment of the Firsthand Platform. Specifically, these Procedures are intended to:

- Provide ease-of-understanding of code and hardware implementations for Staff
- Provide a simple and repeatable methodology for making regular changes to the Platform.
- Proactively address security concerns by way of peer review.
- Encourage best-practices in all engineering endeavors (software and hardware) by way of automated testability, optimization, and separation of concerns.

3 Scope and Applicability

These Procedures are applicable to all of Firsthand software and server/networking hardware that might impact the network performance, operations, and security of the Platform. Hardware and software used for specialty or business purposes that are disconnected from the Firsthand Platform do not fall under the scope of this Procedure.

4 Audience

The primary audience for the Configuration Management Procedure includes all Firsthand in roles that are directly responsible for the configuration, management, oversight, and successful day-to-day operations of Firsthand Platform hardware, software and applicable documentation.

5 Policy

5.1 Software Development and Deployment

Firsthand Engineering Staff should follow a standardized workflow for generating and deploying new code and logic for the Codebase. The following specifications and assumptions should be in place and further adhered-to throughout the lifecycle of logic in the Codebase:

- The codebase should be stored permanently in a version control system (VCS) such as Git.
- The codebase should maintain separate branches for production-ready (“master”) vs staging code that requires further testing (“staging”).
- Deployed and staged code should be placed into separate, easily-revertible tags.
- Non-trivial database changes should be atomic, auditable, and revertible.
- There will exist a separate *staging* environment separate from the Application’s live-data *production* environment, but otherwise identical in environment, kernel, and relevant packages.
- Firsthand engineers will maintain individual *development* environments, replicating (but distinct from) the *production* and *staging* environments.

5.1.1 Software Development and Team Coordination

The following protocol outlines the lifecycle of a software change from initial development to such time that it is merged into the greater Codebase in preparation of further testing and/or deployment:

1. Firsthand Engineering Staff will produce bugfixes, new features, specification changes, database updates, or any combination thereof as commits in the Codebase VCS.
2. A completed set of commits should be bundled up in a discrete Branch, based off the main branch of the Codebase, and shared with the team. New Branches should be designated either as “features”, “bugfixes”, or “hotfixes”. Hotfix branches will be merged into the “master” branch of the Codebase, and may only be used in the event that changes are limited in scope, do not require database migrations, and do not introduce new functionality.
3. Wherever possible, any newly-deployed code should contain Unit Tests.
4. Upon “completion” of a Branch, the primary engineer should generate a Pull Request to merge it into the main Codebase.
5. All Pull Requests must undergo an automated Continuous Integration process that runs all Unit Tests in the Codebase to ensure against regressions.
6. All Pull Requests must undergo a manual, line-by-line code review by a developer unaffiliated with the Branch.
7. All Pull Requests must be additionally tested in a private *development* environment by a developer unaffiliated with the Branch.
8. In the event that there are gaps in security, best practices, unexpected functionality, or any other issue with the Pull Request, it should be fixed and re-reviewed by another developer, or sent back to the original developer for redress.
9. If the Pull Requests meets all the aforementioned standards, it should be merged in. *Hotfix* branches may be merged directly into the *master* branch. All other branches must be merged into a *staging* branch for further testing.

5.1.2 Software Testing

After a branch has met the standards of Engineering’s internal code review process and merged into either the *staging* or *master* branch, the following protocol should apply to further test the “staging” branch before deployment.

1. The *staging* branch should be kept in synchronization with the *master* branch, but may also contain additional features or bugfixes awaiting testing.
2. The *staging* branch should be regularly deployed to the *staging* environment.
3. Backups of the database from the *production* environment should be regularly copied to, sanitized, and replicated into the *staging* environment.
4. The *staging* branch and any new code merged therein should then exist in the *staging* environment for as long as reasonably required for adequate quality assurance (QA) testing. Ideally, this period should occur for at least three days of QA review.
5. An external QA team should review all code in the *staging* environment nightly to ensure consistency of functionality, lack of regression, and a reasonable hardening to erroneous or malicious end-user input whenever relevant.
6. Issues identified by the QA team should be sent back to Engineering for review and redress.

7. If required, new changes should go through the “Software Development and Team Coordination” phase, followed by the “Software Testing” phase to ensure no further issue.

5.1.3 Deployment

When such time occurs that either the *staging* or *master* branches have been sufficiently vetted and tested as to be ready for deployment, the following protocol should be used for deployment into the production environment:

1. Engineering should generate a new Tag from either the *master* or *staging* branch (depending on what is being deployed) named in the format of “tags/release/YYYY-MM-DD-HHMM”, where “YYYY” refers to the numerical year that the Tag is generated; “MM” and “DD” refer to the month and day, respectively, in two-digit format; and HHMM refers to the 2-digit hours and minutes, in 24-hour format, that the Tag was generated.
2. Newly-generated Tags should include a detailed message identifying exactly which Pull Requests (representing features, hotfixes, and bugfixes) were merged in to newly compose this release on top of the previous chronological one.
3. Engineering should execute a script to trigger a final backup of the production database in the event of an unforeseen emergency.
4. Engineering should execute a script, with the newly-created Tag name as a parameter, to trigger the codebase in a single web server to do the following:
    a. Pull the code for the new Tag.
    b. Switch its webroot to this new code.
    c. Run any outstanding database migrations.
    d. Update any newly-created or security-updated libraries specified in this branch.
    e. Clear any caches.
5. Once database migrations on the initial web server deployment have been completed, all other web servers will run the same process.
6. Engineering should execute a final battery of manual “sanity tests” at their discretion to ensure everything went as planned. These tests should establish that:
    a. An active site representing each of the company’s products is running and responding as expected.
    b. Both advisor and advisee users are able to log into the site as expected.
    c. Search functionality, specifically, remains uninterrupted.
7. In the event of any serious issue resulting from the deployment that would compromise the experience of users or impinge on Firsthand’s uptime guarantee, Firsthand Engineers should attempt the following:
    a. If the problem is limited to one web server, that web server should be removed from the load balancer until such time that the issue has been resolved.
    b. If the problem cannot be quickly resolved, the deployment script should be re-run, with the previous Tag specified.