**Общее**

Git - распределенная система контроля версий, разработанная Линусом Торвальдсом. Изначально Git предназначалась для использования в процессе разработки ядра Linux, но позже стала использоваться и во многих других проектах — таких, как, например, X.org и Ruby on Rails. На данный момент Git является самой быстрой распределенной системой, использующей самое компактное хранилище ревизий.

Bazaar  - система разработка которой поддерживается компанией Canonical — известной своими дистрибутивом Ubuntu и сайтом <https://launchpad.net/>. Система в основном написана на языке Python и используется такими проектами, как, например, MySQL и Drupal.

**Основные концепции**

**Git**

Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов и хранилище, содержащее собственно файлы. Структура хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов.

По умолчанию репозиторий хранится в подкаталоге с названием «.git» в корневом каталоге рабочей копии дерева файлов, хранящегося в репозитории. Любое файловое дерево в системе можно превратить в репозиторий git, отдав команду создания репозитория из корневого каталога этого дерева (или указав корневой каталог в параметрах программы). Репозиторий может быть импортирован с другого узла, доступного по сети. При импорте нового репозитория автоматически создаётся рабочая копия, соответствующая последнему зафиксированному состоянию импортируемого репозитория (то есть не копируются изменения в рабочей копии исходного узла, для которых на том узле не была выполнена команда commit).

В классическом обычном сценарии в репозитории git есть три типа объектов — файл, дерево и коммит. Файл есть какая-то версия какого-то пользовательского файла, дерево — совокупность файлов из разных поддиректорий, коммит — дерево + некая дополнительная информация (например, родительский(е) коммит(ы), а также комментарий).

Очень мощной возможностью git являются ветви, реализованные куда более полно, чем в SVN. Создать новую ветвь так же просто, как и совершить коммит. По сути, ветвь git есть не более чем читаемое человеком имя, «навешенное» на некий коммит в репозитории (используется поддиректория refs). Коммит без создания новой ветви всего лишь передвигает эту ссылку на себя, а коммит с созданием ветви — оставляет старую ссылку на месте, но создает новую на новый коммит, и объявляет её текущей. Заменить локальные девелоперские файлы на набор файлов из иной ветви, тем самым перейдя к работе с ней — также тривиально.

**Bazaar**

Bazaar - наиболее гибкая из распределенных систем контроля версий. Одним из примеров гибкости Bazaar может служить возможность использования как централизованной модели, так и распределенной и даже смешивание этих моделей контроля версий.

Во многих UNIX-совместимых операционных системах Bazaar может быть установлен штатными средствами системы. Для Windows и других ОС, в которых Bazaar не доступен для установки штатными средствами, его можно скачать с официального сайта.

Конфигурационные файлы Bazaar хранятся в директории *$HOME/.bazaar* для UNIX-подобных операционных систем и в директории *C:Documents and Settings<username>Application DataBazaar2.0* для Windows. В этом директории хранятся три основных файла конфигурации:

* bazaar.conf — описывает конфигурационные параметры по умолчанию;
* locations.conf — описывает конфигурационные параметры для отдельных веток;
* authentication.conf — описывает идентификационную информацию для доступа к удаленным серверам;

**Ревизия** — это сохраненное состояние файлов и директорий, включая их содержимое и иерархию в заданный момент времени. С ревизией также связана мета-информация, например: - Автор изменения; - Дата изменения; - Комментарий связанный с изменением; - Родительские ревизии от которых произведена данная ревизия;

После создания, ревизии не меняются и могут быть идентифицированы глобально уникальным номером ревизии (revision-id), например таким:

pqm@pqm.ubuntu.com-20071129184101-u9506rihe4zbzyyz

Идентификаторы ревизий создаются в момент фиксации (commit) изменений, или в момент импорта ревизий из других систем. Идентификаторы ревизий необходимы для функционирования системы, но вряд ли кто-то сможет использовать идентификатор такого типа в разговоре. Специально для упрощения идентификации ревизий разработаны специфичные для ветки номера ревизий.

Номера ревизий генерируются «на лету» в момент исполнения команд и представляют из себя последовательно увеличивающийся номер. Таким образом первая ревизия имеет номер 1, следующая за ней, более поздняя, ревизия - номер 2 и так далее. При объединении изменений с других веток, номер ревизии представляется тремя числами, разделенными точками:

1. Номер ревизии для данной ветки, от которой были произведены объединенные изменения;
2. Порядковый номер (счетчик) объединенной ветки. Так как объединение может затрагивать сразу несколько веток, этот номер служит их счетчиком;
3. Порядковый номер ревизии с момента создания объединяемой ветки;

* **Рабочее дерево** — это директория под контролем версий содержащий файлы, которые может редактировать пользователь. Рабочее дерево ассоциировано с веткой. Многие из команд Bazaar используют рабочее дерево для своей работы. Например, команда commit (зафиксировать изменения) создает новую ревизию на основе текущего состояния рабочего дерева.
* **Ветка** - в простейшем случае это, упорядоченная серия ревизий, где последняя ревизия называется верхушкой. Ветки могут разделяться на несколько веток и затем, позже объединяться обратно формируя граф ревизий. При этом в рамках ветки может быть основная линия разработки, ревизии в которой помечаются обычными номерами ревизий. Также ветка может иметь другие линии разработки, помечаемые номерами ревизий через точку, как описано выше.
* **Репозиторий** — это хранилище ревизий. Обычно, ветка имеет свой собственный репозиторий, но также несколько веток могут разделять один репозиторий, что бы уменьшить используемое ветками место на диске.