**Continuous Integration**

Непрерывная интеграция — это практика разработки программного обеспечения, которая заключается в выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем. В обычном проекте, где над разными частями системы разработчики трудятся независимо, стадия интеграции является заключительной. Она может непредсказуемо задержать окончание работ. Переход к непрерывной интеграции позволяет снизить трудоёмкость интеграции и сделать её более предсказуемой за счет наиболее раннего обнаружения и устранения ошибок и противоречий.

**Требования к проекту**

* Исходный код и всё, что необходимо для сборки и тестирования проекта, хранится в репозитории системы управления версиями;
* Операции копирования из репозитория, сборки и тестирования всего проекта автоматизированы и легко вызываются из внешней программы.

**Организация**

На выделенном сервере организуется служба, в задачи которой входят:

* получение исходного кода из репозитория;
* сборка проекта;
* выполнение тестов;
* развёртывание готового проекта;
* отправка отчетов.

Локальная сборка может осуществляться:

* по внешнему запросу,
* по расписанию,
* по факту обновления репозитория и по другим критериям.

В случае сборки по расписанию (англ. daily build — рус. ежедневная сборка), они, как правило, проводятся каждой ночью в автоматическом режиме — ночные сборки (чтобы к началу рабочего дня были готовы результаты тестирования). Для различия дополнительно вводится система нумерации сборок — обычно, каждая сборка нумеруется натуральным числом, которое увеличивается с каждой новой сборкой. Исходные тексты и другие исходные данные при взятии их из репозитория (хранилища) системы контроля версий помечаются номером сборки. Благодаря этому, точно такая же сборка может быть точно воспроизведена в будущем — достаточно взять исходные данные по нужной метке и запустить процесс снова. Это даёт возможность повторно выпускать даже очень старые версии программы с небольшими исправлениями.

**Преимущества**

* проблемы интеграции выявляются и исправляются быстро, что оказывается дешевле;
* немедленный прогон модульных тестов для свежих изменений;
* постоянное наличие текущей стабильной версии вместе с продуктами сборок — для тестирования, демонстрации, и т. п.
* немедленный эффект от неполного или неработающего кода приучает разработчиков к работе в итеративном режиме с более коротким циклом.

**Недостатки**

* затраты на поддержку работы непрерывной интеграции;
* потенциальная необходимость в выделенном сервере под нужды непрерывной интеграции;
* немедленный эффект от неполного или неработающего кода отучает разработчиков от выполнения периодических резервных включений кода в репозиторий.
  + в случае использования системы управления версиями исходного кода с поддержкой ветвления, эта проблема может решаться созданием отдельной «ветки» (англ. branch) проекта для внесения крупных изменений (код, разработка которого до работоспособного варианта займет несколько дней, но желательно более частое резервное копирование в репозиторий). По окончании разработки и индивидуального тестирования такой ветки, она может быть объединена (англ. merge) с основным кодом или «стволом» (англ. trunk) проекта.

**TeamCity**

TeamCity — серверное программное обеспечение от компании JetBrains, написанное на языке Java, билд-сервер для обеспечения непрерывной интеграции. Первый релиз состоялся 2 октября 2006 года.

Возможности:

* Предварительное тестирование кода перед коммитом. Предотвращает возможность коммита программного кода содержащего ошибки, нарушающие нормальную сборку проекта, путём удалённой сборки изменений перед коммитом.
* Грид-сборка проекта. Предоставляет возможность производить несколько сборок проекта одновременно, производя тестирование на разных платформах и в различном программном окружении.
* Интеграция с системами оценки покрытия кода, инспекции кода и поиска дублирования кода.
* Интеграция с различными средами разработки: Eclipse, IntelliJ IDEA, Visual Studio.
* Поддержка различных платформ: Java, PHP, .NET и Ruby.

**Jenkins**

Некоторые плюсы использования Jenkins:

* Когда кто-то ломает билд, вы узнаете об этом сразу, что позволяет быстро устранить проблему;
* Вы можете автоматизировать прогон тестов, деплой приложения на тестовый сервер, проверку code style и тому подобные вещи;
* Также в Jenkins можно хранить собранные deb-пакеты, отчеты о прогоне тестов или Javadoc/Doxygen/EDoc-документацию;

**Links**

<http://habrahabr.ru/post/132521/>

<https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D1%8F>

<http://habrahabr.ru/post/82724/>