Введение

Системы непрерывной интеграции - это программное обеспечение, созданное для автоматизированного тестирования програмных продуктов. Такие системы отслеживают атомарные изменения (коммиты) в системе контроля версий создаваемого продукта, и производят сборку и тестирование проекта в данном состоянии. В конце автоматизированного процесса сборки и тестирования разработчик получает либо положительный (ПО собралось и успешно прошло все этапы тестирования), либо отрицательный результат (возникла ошибка на каком-то этапе). Также разработчик может проследить весь процесс по генерируемым логам (журналам событий сборки и тестирования) и найти примерное место, в котором произошла ошибка. Необходимо заметить, что для таких систем наиболее важным параметром является время выполнения цикла сборки и тестирования. Удовлетворительным является время до 10 минут [3], но это также зависит и от масштабов проекта. Важно, чтобы разработчик получал как можно более быстрый отклик по изменениям, которые он внес в проект.

Системы непрерывной интеграции на сегодняшний день являются неотъемлемой частью крупных проектов по разработке ПО, а также добавляют удобства разработки для небольших проектов.

По сравнению с ручной сборкой и тестированием автоматические системы имеют ряд неоспоримых преимуществ:

* Весь процесс автоматизирован. Таким образом исключено влияние человека на процесс, а, следовательно, и связанные с этим фактором ошибки и неточности выполнения инструкций. Также у разработчика освобождается дополнительное время, которое он может потратить на разработку продукта.
* Однотипное, изолированное окружение – для каждой новой сборки создается выделенный контейнер, в котором создается окружение, где существует только то, что необходимо данному ПО и исключены влияния посторонних процессов.
* Сборка и тестирование на множестве платформ одновременно – для каждой целевой операционной системы можно создать виртуальную машину и/или контейнер и параллельно собирать и запускать тесты на разных ОС.
* Быстрота – программе не нужно тратить время на набор комманд и сверку с инструкциями.
* Автоматически генерируемые отчеты о сборках.

Данная работа посвящена аспектам настройки системы непрерывной интеграции в рамках существующей инфраструктуры проекта MPD Root.

Проект MPD Root — это набор программ и библиотек, позволяющих проводить симуляцию и анализ данных, полученных на Многоцелевом Детекторе (Multi Purpose Detector, MPD), созданном для изучения ядерной материи при экстремальных значениях плотности и температуры. Эксперимент MPD будет проводиться на создаваемом в ОИЯИ на основе Нуклотрона коллайдерном комплексе для ускорения тяжелых ионов (Nuclotron-based Ion Collider fAcility, NICA) [2].

Определения

**Непрерывная интеграция (Continous Integration, CI)** — практика частой сборки и тестирования программного проекта с целью выявления ошибок на ранней стадии.

**Система контроля версий** — ПО для облегчения работы с изменяющейся информацией, позволяет хранить несколько версий одного документа и, при необходимости, возвращаться к более ранним, определять, кто и когда сделал то или иное изменение, а также многое другое.

**Репозиторий** — хранилище в системе контроля версий, отведенное под конкретный проект.

**Коммит** — объект, характеризующий изменения в репозитории.

**Docker-контейнер** — изолированная среда выполнения приложения, запущенная на базе операционной системы Linux.

**Образ (image) docker-контейнера** — объект, хранящий состояние docker-контейнера. При запуске контейнера указывается образ, на базе которого его нужно запустить.

**Программное окружение** — совокупность доступных для запуска программ в данной ОС.

**Сборка** — процесс компиляции ПО.

**Развертывание ПО** — процесс настройки и запуска ПО.

**Виртуальная машина** -

Постановка задачи

Пусть имеется 2 виртуальные машины Windows Azure с установленной на ней ОС с ядром Linux: одна – для развертывания сервера системы контроля версий, вторая – для системы непрерывной интеграции.

Характеристики первого сервера (VCS):

* **CPU**: Octa core Intel Xeon CPU E5-2673 v3 (-HT-MCP-) 2.40GHz
* **Количество ядер:** 8
* **Потоков на ядро:** 1
* **RAM:** 28GB
* **SSD:** 91.6GB
* **ОС:** Ubuntu 14.04

Характеристики второго сервера (CI):

* **CPU**: 20 cores Intel Xeon CPU E5-2673 v3 (-HT-MCP-) 2.40GHz
* **Количество ядер:** 20
* **Потоков на ядро:** 1
* **RAM:** 140GB
* **SSD:** 1TB
* **ОС:** Ubuntu 14.04

Мы рассмотрим процесс развертывания и настройки обоих серверов.

Конечная система должна удовлетворять следующим требованиям:

* Сборка и тестирование должны производиться при каждом коммите
* Весь процесс сборки и тестирования должен описываться в текстовом файле(-ах), находящемся в репозиторий проекта и включенном в систему контроля версий (чтобы каждую сборку, при желании, было можно произвести по отдельному правилу)
* Процесс сборки и тестирования не должен в среднем занимать времени больше 10 минут [3]
* Каждая сборка должна производиться в выделенном контейнере
* По каждой сборке должен создаваться и храниться отчет
* Система должна легко развертываться на описанных выше серверах

Обзор использованных технологий

GitLab

GitLab CI Runner

Docker

Развертывание и настройка сервера контроля версий

1. root@mpdci:/home/panda# df -H
2. Filesystem      Size  Used Avail Use% Mounted on
3. udev             74G  8.2k   74G   1% /dev
4. tmpfs            15G  418k   15G   1% /run
5. /dev/sda1        31G  2.5G   28G   9% /
6. none            4.1k     0  4.1k   0% /sys/fs/cgroup
7. none            5.3M     0  5.3M   0% /run/**lock**
8. none             74G     0   74G   0% /run/shm
9. none            105M     0  105M   0% /run/user
10. none             66k     0   66k   0% /etc/network/interfaces.dynamic.d
11. /dev/sdb1       296G   66M  281G   1% /mnt

Развертывание и настройка сервера сборки и тестирования

Создание скрипта сборки и тестирование

Эксперименты

Заключение

Список литературы