Развёртывание компонент ИС метаданных SPD

Основная цель эксперимента SPD - проведение экспериментов по изучению спиновой структуры нуклонов и других процессов, зависящих от спина, путем измерений асимметрий в процессе Дрелла-Яна, процессы рождения J/Ψ частиц и прямых фотонов в столкновениях поляризованных протонов и дейтронов.

В условиях проведения сложнейших экспериментов на установках ключевая роль отводится системе сбора данных. Единовременным фиксированием различных по своей сути сигналов занимаются датчики в количестве нескольких сотен тысяч штук. Сбор данных — это процесс выборки сигналов, измеряющих реальные физические условия, и преобразования полученных выборок в цифровые числовые значения, которыми можно манипулировать с помощью компьютера.

Так как была выбрана концепция микросервисной архитектуры, то для эффективной разработки, поддержки системы и ее развертывания требуется автоматизация некоторых процессов (сборка, развертывание, тестирование).

Было принято решение использовать подход CI/CD и инструмент для его реализации – GitLab.

Ключевыми особенностями GitLab являются непрерывность процесса разработки и взаимная интеграция различных элементов.

- Локальное развертывание (jinr.gitlab);
- Фичи по CI/CD (Gitlab CI/CD);
- Автоматическое обнаружение секретов и тестирование безопасности;
- Явные разрешения для ограничения слияния и отправки кода определенным пользователям;
- Бесплатные статические веб-сайты в репозиториях Git (Gitlab Pages);

GitFlow — это определенная надстройка над моделью ветвления Git, которая включает в себя использование фича веток и несколько основных веток. По сравнению с разработкой на основе «магистрали», GitFlow имеет многочисленные, более долгоживущие ветки и более крупные фиксации. В рамках этой модели разработчики создают фича ветку и откладывают ее слияние с основной веткой до тех пор, пока функционал не будет завершен. Эти долгоживущие фича ветки требуют большего сотрудничества для слияния и имеют более высокий риск отклонения от ветви магистрали. Они также могут вводить противоречивые обновления.

Но, по сути, если отбросить все условности, то модель разработки GitFlow заключается между 2х постоянных веток:

- master (стабильно работающая или продакшен версия кода)
- develop (последняя или «nightly»-версия кода)

main — это основная ветка кода, продакшен версия кода, то есть все сборки для реальных клиентов собирают как раз из этой ветки. Ветка может быть помечена тегами, в которых есть информация о версии и часто к тегу еще привязываются релизы. Любая новая задача, взятая в работу, начинается с мастер ветки.

Суть этой ветки хранить самую последнюю работоспособную версию кода.

develop branch - Исходя из практического опыта программисты не часто работают с этой веткой, а большую часть времени с ней проводят, как раз тестировщики анализирую функционал на работоспособность и наличие багов, а программисты просто сливают в нее новый код.

Суть этой ветки просто хранить в себе самую последнюю кодовую базу проекта.

Также немаловажной особенностью мы считаем использование соглашения о коммитах.

Мотивация – в процессе разработки ПО возникает потребность в его реактивной поддержке. У пользователей появляется желание получать новые версии с новым функционалом и исправлением багов чаще. Перед каждым релизом разработчики вынуждены проводить тестирование ПО, поэтому каждый выпуск релиза после добавления функциональности вызывает чрезмерные трудозатраты. Следовательно – разработчикам не выгодно делать релизы после новой фичи или исправления, а пользователям этого хочется.

Суть – автоматизация процессов, которые занимают много лишнего времени. Можно сравнить с конвейером, куда попадает код после коммита.

Мотивация:

• разработчик отпраляет коммит в репозиторий, создаёт merge request через сайт, или ещё каким-либо образом явно или неявно запускает пайплайн;

• из конфигурации выбираются все задачи, условия которых позволяют их запустить в данном контексте;

• задачи организуются в соответствии со своими этапами;

• этапы по очереди выполняются — т.е. параллельно выполняются все задачи этого этапа,

• если этап завершается неудачей (т.е. завершается неудачей хотя бы одна из задач этапа) — пайплайн останавливается (почти всегда);

• если все этапы завершены успешно, пайплайн считается успешно прошедшим.

При развертывании на сервере, старые версии падают, удаляются и запускаются новые, обновленные, взятые из реестра.

Как я уже отметил ранее – Чтобы пайплайн работал нужна среда для выполнения - Docker контейнер. Мы могли бы юзать DockerHub, но зачем использовать сервис извне?

Каждый запуск пайплайна сопровождается сборкой образов наших микросервисов, автоматическим присваиванием тега на основе ID совершённого коммита, отправкой этих образов в Container Registry. При коммите в main – ветку, образам присваивается тег latest, с заменой предыдущей версии в реестре.

Для эффективной поддержки продукта, требуется документация, но, как правило, она устарвает быстрее всего в проектах. Поэтому необходимо средство для автоматической генерации документации и её актуализации для всех команды проекта.

С помощью Gitlab Pages можно публиковать статические веб-сайты непосредственно из хранилища Gitlab:

- Можно использовать любой генератор статических сайтов (SSG) или обычный HTML;
- Создание веб-сайта для своих проектов, групп или учетной записи пользователя;
- Размещение на собственном экземпляре Gitlab или Gitlab.com бесплатно;

- Подключение своих пользовательских доменов и сертификатов TLS.

Для того чтобы по исходному коду и комментарицям в определенном формате сгенерировать документацию – мы используем Сфинкс, а для того чтобы поделиться документацией со всеми участниками проекта – используется GitlabPages, который размещает и обновляет сайт с документацией во в рамках выполнения пайплайна

Для развертывания требуются данные для авторизации к БД, например. Для автоматизации развертывания необходимо хранить все пароли используя Гитлаб, однако это не безопасно. Соответственно, необходимо хранить данные в зашифрованном виде

Для каждой задачи (job) генерируется свой уникальный токен JSON Web Token (JWT), доступный только как значение переменной окружения CI_JOB_JWT конкретной задачи.

Данный JWT может быть использован для аутентификации в Vault при помощи метода JWT Auth.

Токен кодируется по стандарту RS256 и подписывается приватным ключом OpenID Connect вашего GitLab инстанса, причем этот ключ периодически изменяется без вашего ведома. И, если приватный ключ был изменен, при следующем запуске задания новый JWT будет подписан с новым ключом. Срок валидности токена будет устанавливаться в соответствии с таймаутом вашего задания, а если он не задан, то срок валидности будет 5 минут.

Изучены проектируемые информационные системы эксперимента SPD, модели данных, а также реализации сходных систем в других экспериментах;

- Внедрен GitFlow с использованием стандартизированных принципов коммита («Соглашение о коммитах»);
- Настроен реестр Container Registry, в котором уже хранятся Docker образы микросервисов БД ИС SPD;
- Подключены и настроены собственные Gitlab Runners;
- Реализован работающий пайплайн на базе Gitlab CI/CD, включающий автоматическую:
    - сборку Docker-образов микросервисов;
    - выгрузку образов Docker в реестр Container Registry проекта;
    - развертывание собранных образов из реестра на деплоймент-сервер;
    - публикацию документации на проектном домене экземпляра git.jinr;
- Поднят и актуализируется сайт с документацией на основе Sphinx на тестовом репозитории (из-за ограничений прав доступа, об этом в дальнейших планах);
- Сконфигурирован и поднят Vault-сервер для хранения зашифрованных переменных окружения и данных аутентификации.

Дальнейшие планы:

- Реализовать автоматическое юнит и нагрузочное тестирование в рамках выполнения пайплайна на отдельном тестовом сервере (виртуальной машине);
- Связать имеющихся раннеров с хранилищем Vault для передачи чувствительной информации и переменных окружения в зашифрованном виде;
- Настройка прав доступа для сайта Gitlab Pages;
- Оптимизация пайплайна;
- Оркестрация;
- Переход с Docker на Apptainer;
- Внедрение подхода CI/CD в другие группы и подразделения работающие над ПО для SPD.