SMART
CONSULTING

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«Digit МЭВ»

Сведения о хранении и компиляции

Листов 7

2024

СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ

Сокращение	Наименование
GIT	Распределенная система управления версиями, которая отслеживает и фиксирует изменения в файлах (чаще всего речь идет об исходном коде программ, однако систему можно использовать для файлов любого назначения).
GitLab	Инструмент хранения и версионирования исходного кода с использованием системы контроля версий Git. Кроме того, инструмент под названием Gitlab CI, интегрированный в Gitlab, используется для автоматизации выполнения непрерывного релизного цикла продуктов, исходный код которых разрабатывается в Gitlab.
GitLab Runner	Корпоративная сеть передачи данных – телекоммуникационная сеть, объединяющая в единое информационное пространство все краевые структурные подразделения органов власти
GitLab CI	Инструмент автоматизации выполнения непрерывного релизного цикла продуктов, исходный код которых разрабатывается в Gitlab
Pipeline	Цепочка задач, организованных в последовательные этапы, ветвление на параллельно запускаемые этапы также возможно. Цепочка конфигурируется из входных параметров, подготовленных CI/CD системой на основании определяемых начальных условий, например, ветка/тег git-репозитория
YAML	Формат данных, близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования. Формат удобен для редактирования, поэтому активно используется как один из основных конфигурационных форматов.
Репозиторий	Централизованное хранилище исходного кода в Git, проект в Gitlab

1 Хранение исходного кода

Для работы над проектом используется локально развернутая версия системы контроля версий GitLab.

GitLab — веб-инструмент жизненного цикла DevOps с открытым исходным кодом, представляющий систему управления репозиториями кода для Git с собственной вики, системой отслеживания ошибок, CI/CD пайплайном и другими функциями.

Команда разработчиков взаимодействует с консольным или браузерным инструментом для выгрузки кода на сервер и изменения структуры.

Gitlab CI/CD — платформа, предоставляющая набор инструментов для обеспечения процессов непрерывной интеграции и доставки. Gitlab позволяет автоматизировать процесс разработки и внедрить в него такие действия как:

- Проверка исходного кода;
- Сборка артефакта по исходному коду;
- Публикация артефакта в хранилище артефактов.

Все эти действия могут выполняться в зависимости от каких-либо событий или триггеров, совершенных с репозиторием, например:

- Выполнена фиксация изменений (коммит) в ветку feature/*;
- Выполнено слияние изменений в ветку master;
- Установлен тег на ветку rc;
- и т.д.

Соответствие таких событий и автоматических действий, а также их последовательность в совокупности являются конвейером CI/CD.

Конвейер CI/CD:

- Регламентируется процессом разработки (flow);
- Конфигурируется файлом .gitlab-ci.yml в корне репозитория;
- Обеспечивается утилитой Gitlab Runner.

2 Особенности единого репозитория GitLab

— управление публичными и приватными git-репозиториями;

— управление пользователями и группами, правами доступа к git-репозиториям;

— отслеживание ошибок, деплой, анализ кода;

— интеграция с разными CI-системами CI (Jenkins и т. п.), организация самостоятельного процесса CI посредством встроенных средств.

Ролевая модель, используемая в Gitlab, состоит из следующих ролей:

- Owner — Владелец группы/проекта;
- Maintainer — Ответственный за группу/проект;
- Developer — Разработчик в группе/проекте;
- Reporter — Пользователь группы/проекта;
- Guest — Гость в группе/проекте.

Группы и проекты

Проект — ключевая сущность Gitlab, агрегирующая в себе следующие функциональные возможности:

- работа с git-репозиторием исходного кода;
- управление конвейером CI/CD;
- формирование релизного цикла;
- выполнение проверки кода;
- управление проектами и задачами;
- контроль доступа;
- просмотр активности и прочей аналитики.

Группа — сущность Gitlab, с помощью которой можно объединять и структурировать проекты.

3 Компиляция исходного кода

Процесс компиляции и сборки файлов JAR-файла из исходных файлов в GitLab обычно включает в себя несколько шагов. Настраивается пайплайн (pipeline) с использованием файла конфигурации (например, .gitlab-ci.yml), который будет описывать, какие шаги выполнять для сборки и компиляции проекта.

Пример простого процесса компиляции и сборки Java-проекта в GitLab:

1. **Создание файла .gitlab-ci.yml:** В корневом каталоге репозитория создаем файл .gitlab-ci.yml, если его ещё нет. Этот файл будет содержать описание пайплайна.
2. **Определение структуры пайплайна:** Определение этапов, которые будут включены в пайплайн. Пример структуры:

```
stages:
  - build
build:
  stage: build
  script:
    - echo "Compiling and building JAR..."
    - javac -d build/ src/*.java
    - jar cvf myapp.jar -C build/ .
  artifacts:
    paths:
      - myapp.jar
```

Описание этапа сборки: В примере используется единственный этап *build*. В этом этапе мы указываем команды для компиляции Java-файлов и создания JAR-файла.

Сохранение артефактов: В блоке *artifacts* указываются пути к файлам и каталогам, которые мы хотим сохранить как артефакты пайплайна. В данном случае, мы сохраняем созданный JAR-файл.

Коммит и пуш: Делаем коммит файла .gitlab-ci.yml и любых других необходимых изменений, затем отправляем их в наш GitLab репозиторий.

Запуск пайплайна: После пуша, GitLab автоматически обнаружит файл .gitlab-ci.yml и начнет запуск пайплайна. Необходимо перейти на страницу репозитория в GitLab и открыть раздел "CI/CD" или "Pipelines", чтобы следить за ходом выполнения.

Этот пример демонстрирует базовый процесс сборки JAR-файла из Java-исходных файлов в GitLab с использованием GitLab CI/CD. В более сложных проектах мы можем добавить этапы тестирования, статического анализа и др.