Кейс 1: Настройка простого пайплайна CI с использованием GitHub Actions

Задание:

1. Создать репозиторий на GitHub.
2. Настроить простой пайплайн CI с использованием GitHub Actions для автоматической сборки приложения при каждом коммите.

Ожидаемый результат:

- Репозиторий на GitHub создан.
- Пайплайн CI настроен и успешно выполняется при каждом коммите.

Кейс 2: Автоматическое тестирование с использованием GitLab CI/CD

Задание:

1. Создать проект на GitLab.
2. Настроить файл `.gitlab-ci.yml` для автоматического запуска юнит-тестов при каждом коммите.

Ожидаемый результат:

- Проект на GitLab создан.
- Файл `.gitlab-ci.yml` настроен и юнит-тесты запускаются автоматически при каждом коммите.

Кейс 3: Автоматическое развертывание на Heroku с использованием CircleCI

Задание:

1. Создать проект на GitHub.
2. Настроить CircleCI для автоматической сборки и развертывания приложения на Heroku при каждом пуше в ветку `main`.

Ожидаемый результат:

- Проект на GitHub создан.
- CircleCI настроен и приложение автоматически развертывается на Heroku при каждом пуше в `main`.

Кейс 4: Использование Docker в пайплайне CI/CD

Задание:

1. Создать Dockerfile для приложения.
2. Настроить Jenkins для сборки Docker-образа и его отправки в Docker Hub при каждом коммите.

Ожидаемый результат:

- Dockerfile создан.
- Jenkins настроен для автоматической сборки и отправки Docker-образа в Docker Hub при каждом коммите.

Кейс 5: Многоэтапный пайплайн с проверкой кода и развертыванием

Задание:

1. Настроить Jenkins для выполнения многоэтапного пайплайна, включающего проверку кода (lint), юнит-тесты, сборку и развертывание.
2. Использовать различные стадии (stages) для каждой части пайплайна.

Ожидаемый результат:

- Jenkins настроен для выполнения многоэтапного пайплайна.
- Все этапы пайплайна выполняются последовательно и успешно.

Кейс 6: Настройка уведомлений о статусе пайплайна

Задание:

1. Настроить уведомления о статусе пайплайна (успешно/неудачно) в Slack или по электронной почте.
2. Использовать GitHub Actions или Jenkins для отправки уведомлений.

Ожидаемый результат:

- Уведомления настроены и отправляются при каждом выполнении пайплайна.

Кейс 7: Интеграция с системой управления версиями (VCS)

Задание:

1. Настроить автоматическое создание релизов на GitHub/GitLab при пуше тегов.
2. Включить генерацию changelog и прикрепление артефактов сборки.

Ожидаемый результат:

- Релизы автоматически создаются при пуше тегов.
- Changelog генерируется и артефакты сборки прикрепляются к релизам.

Кейс 8: Развертывание в Kubernetes с использованием CI/CD

Задание:

1. Настроить GitLab CI/CD для автоматической сборки Docker-образов и развертывания их в кластер Kubernetes.
2. Использовать Helm для управления развертываниями.

Ожидаемый результат:

- GitLab CI/CD настроен для сборки и развертывания в Kubernetes.
- Приложение развернуто в кластере Kubernetes с использованием Helm.

Кейс 9: Проверка безопасности и качества кода

Задание:

1. Интегрировать в пайплайн инструменты для анализа безопасности (например, SonarQube).
2. Настроить автоматический запуск анализа при каждом коммите.

Ожидаемый результат:

- Инструменты для анализа безопасности интегрированы в пайплайн.
- Анализ безопасности выполняется автоматически при каждом коммите.

Кейс 10: Автоматизация базы данных

Задание:

1. Настроить пайплайн для автоматического применения миграций базы данных при каждом развертывании.
2. Использовать инструменты для управления миграциями (например, Flyway или Liquibase).

Ожидаемый результат:

- Пайплайн настроен для автоматического применения миграций.
- Миграции базы данных применяются автоматически при каждом развертывании.