4.29 Оценка
280.79 Рейтинг

Southbridge

Обеспечиваем стабильную работу highload-проектов

zubarek 2 часа назад

Быстрое начало работы с Gitlab CI/CD: пайплайн для веб-сайта на AWS S3 Bucket

5 мин 362

Блог компании Southbridge, Тестирование IT-систем*, Системное администрирование

Туториал Перевод

Автор оригинала: Seifeldin Mahjoub

Перевели статью о создании пайплайна для развертывания статического веб-сайта на AWS S3 Bucket на примере Gitlab CI/CD, чтобы быстро вникнуть в основы технологии и начать применять ее в работе. В статье рассматриваются базовые концепции CI и CD, а также этапы CI/CD-пайплайна.

От автора

Мне повезло быть частью некоторых профессиональных команд, каждая из которых применяла несколько DevOps практик. И меня поразило то, как качество кода, скорость разработки и позитивный настрой команды коррелируют с CI/CD-пайплайном.

По моему мнению, зрелость пайплайна может служить прекрасным показателем опытности разработчика, качества кода и эффективности всей команды.

Во многих случаях, которые я наблюдал, пайплайны были выстроены либо DevOps-инженером, либо отдельной DevOps-командой. Да и последний отчет State of CD 2022 продемонстрировал, что только 22% разработчиков создают пайплайны.

Моя цель — увеличить это число: помочь разработчикам взять на себя ответственность за пайплайны, выстраивать непрерывный процесс доставки и создавать качественный код.

В статье рассматриваются фундаментальные концепции CI и CD.

Что такое CI/CD?

Многие бизнесы применяют фреймворки Agile, так как они позволяют менять приоритеты и повышать скорость доставки. Кроме всего прочего, такой подход улучшает атмосферу в команде и помогает увеличить прибыль.

Если ваша компания следует по пути Agile, то принятие культуры, философии и практик DevOps станет ее большим преимуществом.

Модное словечко последних десятилетий, DevOps сегодня считается настоящим стандартом индустрии. CI/CD — это практика DevOps, которая помогает разработчикам ПО доставлять изменения в коде с высокой частотой и надежностью.

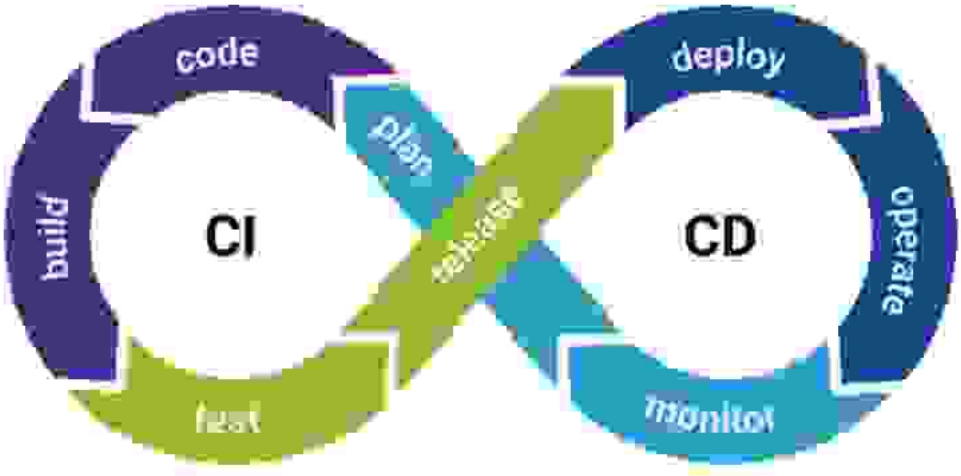
«Быстрый билд, быстрый тест, быстрый фейл»

При наличии автоматизированных тестов команды тяготеют к общей автоматизации задач и частым, надежным поставкам кода. Создание CI/CD-пайплайна в этом случае может привести к нескольким преимуществам.

Бизнес выигрывает от снижения затрат и повышения производительности, ускорения Time to Market и адаптации к изменяющимся требованиям рынка.

Команда выигрывает от быстрой обратной связи, улучшения эффективности разработки, уменьшения количества бутылочных горлышек и повышения уровня вовлеченности и удовлетворенности сотрудников.

Фазы CI и CD



CI — непрерывная интеграция. Непрерывная интеграция позволяет по много раз в день коммитить изменения в основную ветку вашей кодовой базы.

Учитывая ограниченные когнитивные способности человека, CI стимулирует разработчиков *вносить в код небольшие изменения*, которые легче рассмотреть, покрыть автоматическими тестами и часто релизить.

Это позволяет избежать напряженных и переполненных merge conflict-ами дней подготовки к релизу с тоннами ручного тестирования.

CD — непрерывная доставка. Следующий шаг после CI позволяет гарантировать, что кодовая база постоянно готова к деплою, а задеплоить ее можно одним нажатием кнопки.

При этом неважно, с чем вы работаете: с масштабной распределенной системой, сложной производственной средой и т. д. Ключевой момент — автоматизация.

CD — непрерывное развертывание. Последний этап зрелого CI/CD-пайплайна, где все изменения в коде автоматически развертываются в продакшн без ручного вмешательства.

Само собой, для этого требуется большое количество хорошо продуманных автоматических тестов. State of CD 2022 утверждает, что «*47% разработчиков применяют CI или CD, но только один из пяти использует оба подхода для автоматизации сборки, тестирования и развертывания кода*».

Книга Accelerate подводит итоги многолетнего исследования с использованием отчетов State of DevOps, основанных на 23 000 наборов данных компаний по всему миру. Как видите, высокопроизводительные команды могут деплоить по требованию (или несколько раз в день).

2017	High Performers	Medium Performers	Low Performers
Deployment Frequency	On demand (multiple deploys per day)	Between once per week and once per month	Between once per week and once per month*
Lead Time for Changes	Less than one hour	Between one week and one month	Between one week and one month*
MTTR	Less than one hour	Less than one day	Between one day and one week
Change Failure Rate	0–15%	0–15%	31–45%

* Low performers were lower on average (at a statistically significant level) but had the same median as the medium performers.

Этапы CI/CD-пайплайна

Стадия исходного кода — здесь запускается пайплайн. Обычно это происходит после изменений в Git-репозитории, которые проявляются в открытии нового Pull Request-а или в пуше в ветку. Другой способ заключается в настройке инструментария CI/CD для запуска пайплайна через автоматическое расписание или после запуска другого пайплайна.

Стадия сборки — этап, в процессе которого происходит проверка и сборка кода. Здесь особенно полезны такие инструменты, как Docker: они обеспечивают однородную среду.

Стадия тестирования — CI/CD невозможно представить без автоматизированных тестов. В конце концов, все хотят быть уверены, что изменения в коде не сломают продакшн.

Стадия развертывания — на последнем этапе (после успешного прохождения всех предыдущих стадий) код можно развернуть в выбранной среде.

Пример с Gitlab

В этом примере будет использован Gitlab CI/CD, однако концепции аналогичны и для остальных инструментов, поэтому их можно применить к другим сервисам хостинга репозиториев.

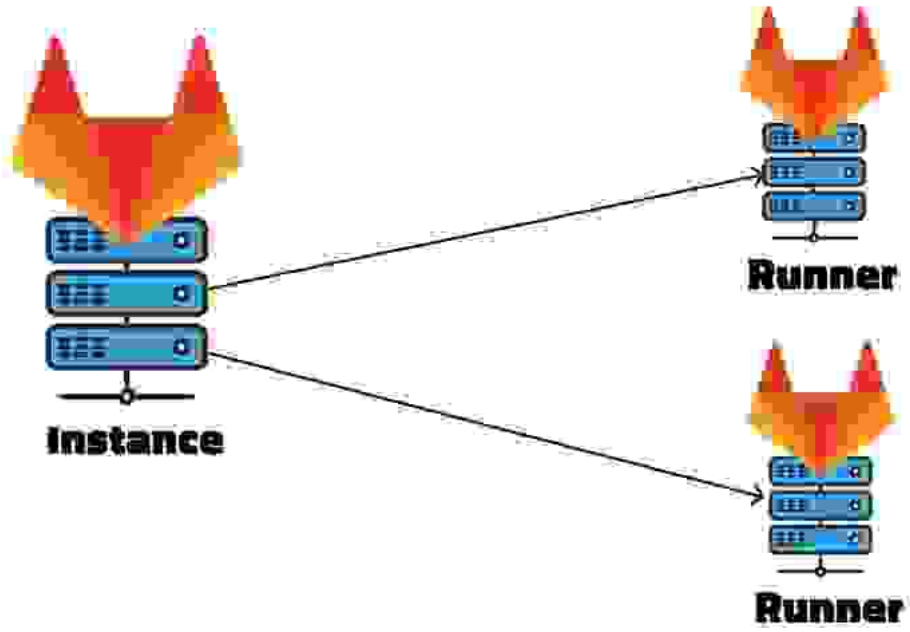
Существует несколько инструментов CI/CD, например всемирно известный Jenkins. Этот инструмент требует некоторой настройки и конфигурации, в то время как другие поставляются сервисами хостинга репозиториев (такими как GitHub Actions и Bitbucket Pipelines) с предварительной настройкой.

Поэтому если ваш код размещен на Gitlab, то легче всего использовать Gitlab CI/CD, поскольку код и управление CI/CD находятся на одной платформе.

Как все это может работать без настроек?

Для ответа на этот вопрос стоит немного погрузиться в архитектуру Gitlab, а именно — в инстансы и раннеры. Инстансы хранят код приложения и конфигурации пайплайна. Раннеры выступают в качестве агентов, выполняющих операции в пайплайнах.

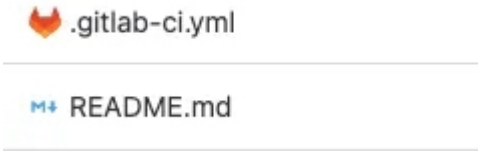
В Gitlab каждый инстанс может быть подключен к одному или нескольким раннерам.



Gitlab.com — это управляемый инстанс с несколькими раннерами, которые сам Gitlab и поддерживает. Следовательно, если вы используете этот инстанс, то получаете все необходимое из коробки.

Приступим к работе

Gitlab предлагает несколько шаблонов при создании нового проекта. Конфигурация пайплайна Gitlab CI/CD по умолчанию находится в файле .gitlab-ci.yml в корневом каталоге.



Предположим, мы хотим создать простой пайплайн, который проверяет: написан, протестирован и развернут ли код. Вот несколько концепций и терминов для ознакомления перед началом работы.

Пайплайн (Pipeline)

Пайплайн — это набор заданий, разделенных на этапы. Gitlab предлагает различные типы пайплайнов, например parent-child или multi-project. Полный список см. здесь.

Этап (Stage)

Этап — это шаг в пайплайне, предоставляющий информацию о том, какие задания запускать (сборка, тестирование и т. д.). Один этап может включать одно или несколько заданий.

Задание (Job)

Задание — основной блок пайплайна (компиляция, линтинг и т. д.). Для каждого задания должны быть определены name и script. После выполнения всех заданий на этапе пайплайн переходит к следующему.

Теперь — к коду

Выстраиваем пайплайн Gitlab CI/CD, который собирает, тестирует и разворачивает статический веб-сайт в AWS S3 Bucket.

Для начала создадим новый .gitlab-ci.yml

1. Определим переменные

```
variables: # variabiles definitions for easier reuse of valu
 CI_NODE_IMAGE: "node:16.13.2"
```

2. Определим этапы

```
# Pipeline stages
stages:
 - install
 - build
 - test
 - deploy
```

3. Определим задания на каждом этапе

```
#install job definition
install:
 stage: install
 image: "$CI_NODE_IMAGE" # variable reference
 script: # Shell script that is executed by the runner.
   - npm ci
 cache: # List of files that should be cached between subsec
   key:
     files:
       - package.json
       - package-lock.json
   paths: # directories to cache
     - node_modules

# Build Job definition
```

```
build:
 stage: build
 image: $CI_NODE_IMAGE
 script:
   - npm run build
 artifacts: # list of files and directories that are attache
   paths:
     - dist/
 cache:
   key:
     files:
       - package.json
       - package-lock.json
   paths:
     - node_modules
   policy: pull

# Test Job definition
test:
 stage: test
 image: $CI_NODE_IMAGE
 script:
   - npm run test


# Deploy Job definition
deploy:
 stage: deploy
 image: registry.gitlab.com/gitlab-org/cloud-deploy/aws-base
 script:
   - aws s3 cp --recursive dist s3://bucket-name # copies th
```

На этом все, спасибо за внимание.

Научиться работать с пайплайнами, билдами и артефактами можно на курсе Gitlab CI/CD в Слёрм. Вы узнаете, из чего состоит Gitlab и какие у него возможности и настройки, а также разберете лучшие практики построения пайплайна, особенности шаблонизации и работы с переменными.

Видеокурс доступен всегда. Посмотреть программу:
https://slurm.club/3JUKdzT

Теги: ci/cd, gitlab-ci, aws, gitlab, pipeline

Хабы: Блог компании Southbridge, Тестирование IT-систем, Системное администрирование, Программирование, DevOps

+7 | 14 | 0

Southbridge

Обеспечиваем стабильную работу highload-проектов

Сайт Сайт

14 16.9
КармаРейтинг

Лиза Зубарькова @zubarek

Пользователь

Комментировать

Публикации

ЛУЧШИЕ ЗА СУТКИ | ПОХОЖИЕ

ИНФОРМАЦИЯ

Сайт	southbridge.io
Дата регистрации	15 ноября 2012
Дата основания	22 февраля 2008
Численность	51–100 человек
Местоположение	Россия
Представитель	Антон Скобин