



Softmart
Компания

 nem 20 сентября 2016 в 14:22

GitLab CI: Учимся деплоить

Автор оригинала: Ivan Nemytchenko

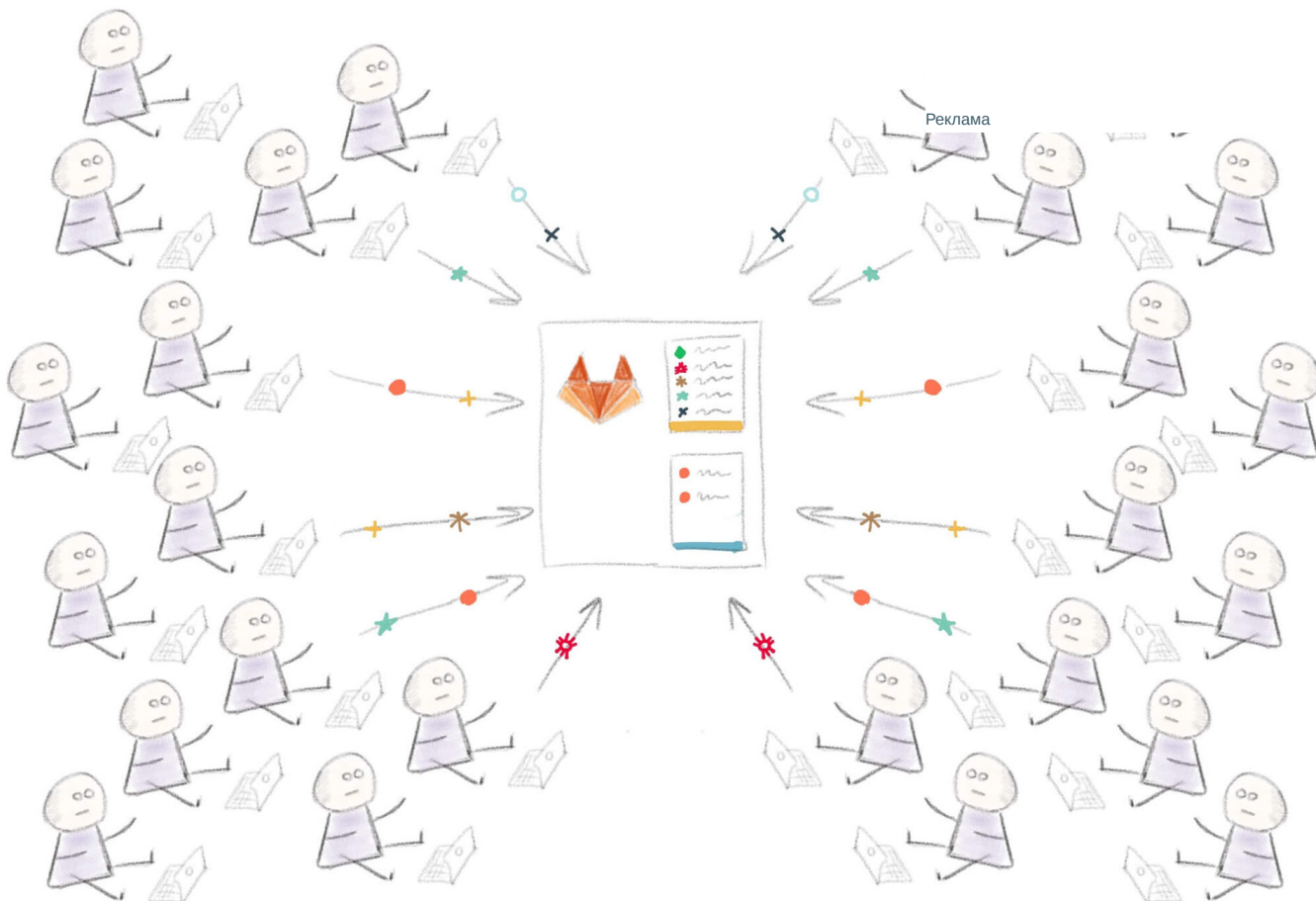
Блог компании Softmart, Open source, Git, Системы управления версиями, Системы сборки

Перевод

В данной статье речь пойдет об истории успеха воображаемого новостного портала, счастливы. К счастью, вы уже храните код проекта на GitLab.com и знаете, что для тестирования мс. Теперь вам интересно, можно ли пойти дальше и использовать CI еще и для развертывания возможности при этом открываются.

Чтобы не привязываться к какой-либо конкретной технологии, предположим, что ваше прил HTML-файлов, никакого выполнения кода на сервере, никакой компиляции JS assets. Деплс

У автора нет цели дать рецепты для конкретной технологии в этой статье. Наоборот, пример чтобы слишком на них не заикливаться. Смысл в том чтобы вы посмотрели на фишки и прир потом применили их для вашей технологии.



Начнем с начала: в вашем вымышленном приложении пока еще нет поддержки CI.

Начало

Развертывание: в данном примере результатом развертывания должно быть появление набора HTML-файлов в вашем бакете (bucket) S3, который уже настроен для хостинга статических вебсайтов).

Добиться этого можно множеством различных способов. Мы будем использовать библиотеку `awscli` от Amazon.

Полностью команда развертывания выглядит следующим образом:

```
aws s3 cp ./ s3://yourbucket/ --recursive --exclude "*" --include "*.html"
```



Пуш кода в репозиторий и развертывание — это два разных процесса

Важно: эта команда ожидает, что вы передадите ей переменные окружения `AWS_ACCESS_KEY_ID` и `AWS_SECRET_ACCESS_KEY`. Также от вас может потребоваться указать `AWS_DEFAULT_REGION`.

Попробуем автоматизировать этот процесс с использованием GitLab CI.

Первое автоматическое развертывание

При работе с GitLab используются все те же команды, что и при работе на локальном терминале. Вы можете настроить GitLab CI в соответствии с вашими требованиями так же, как вы настраиваете ваш локальный терминал. Все команды, которые можно выполнить на вашем компьютере, можно выполнить и в GitLab, передав их в CI. Просто добавьте ваш скрипт в файл `.gitlab-ci.yml` и сделайте пуш — вот и все, CI запускает задачу (*job*) и выполняет ваши команды.

Добавим деталей в нашу историю: ваш веб-сайт небольшой — 20-30 посетителей в день, а в репозитории есть только одна ветка — `master`.

Начнем с того, что добавим задачу с командой `aws` в `.gitlab-ci.yml`:

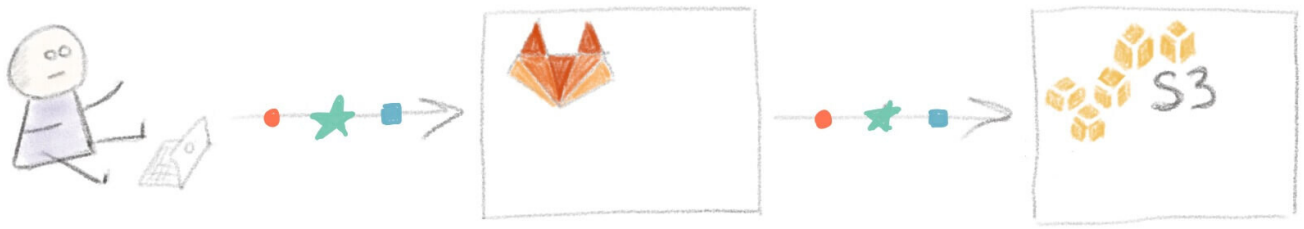
```
deploy:
  script: aws s3 cp ./ s3://yourbucket/ --recursive --exclude "*" --include "*.html"
```

Неудача:

```
$ aws s3 cp ./ s3://yourbucket/ --recursive --exclude "*" --include "*.html"
/bin/bash: line 47: aws: command not found
ERROR: Build failed: exit code 1
```

Нам нужно удостовериться в наличии исполняемого файла `aws`. Для установки `awscli` требуется `pip` — инструмент для установки пакетов Python. Укажем образ Docker с предустановленным Python, в котором также должен содержаться `pip`:

```
deploy:
  image: python:latest
  script:
    - pip install awscli
    - aws s3 cp ./ s3://yourbucket/ --recursive --exclude "*" --include "*.html"
```



При пуше на GitLab код автоматически разворачивается с помощью CI

Установка `awscli` увеличивает время выполнения задачи, но сейчас нас это не особенно волнует. Если вы все же хотите ускорить процесс, можете поискать образ Docker с предустановленным `awscli`, или же создать такой своими силами.

Также, не стоит забывать о переменных окружения, полученных из AWS Console:

```
variables:
  AWS_ACCESS_KEY_ID: "AKIAIOSFODNN7EXAMPLE"
  AWS_SECRET_ACCESS_KEY: "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
deploy:
  image: python:latest
  script:
    - pip install awscli
    - aws s3 cp ./ s3://yourbucket/ --recursive --exclude "*" --include "*.html"
```

Должно сработать, однако держать секретные ключи в открытом виде (даже в приватном репозитории) — не самая лучшая идея. Посмотрим, что с этим можно сделать.

Сохранение секретов

В GitLab есть специальный раздел для секретных переменных: **Settings > Variables**

Secret Variables

These variables will be set to environment by the runner.

So you can use them for passwords, secret keys or whatever you want.

The value of the variable can be visible in build log if explicitly asked to do so.

Add a variable

Key

Value

Add new variable

Все данные, помещенные в этот раздел станут **переменными окружения**. Доступ к этому разделу есть только у администратора проекта.

Раздел `variables` можно удалить из настроек CI, но лучше давайте воспользуемся им для другой цели.

Создание и использование публичных переменных

С увеличением размера файла конфигурации становится удобнее хранить некоторые параметры в его начале в качестве переменных. Такой подход особенно актуален в случае, когда эти параметры используются в нескольких местах. Хоть в нашем случае до такого пока что не дошло, в демонстрационных целях создадим **переменную** для названия бакета S3:

```
variables:
  S3_BUCKET_NAME: "yourbucket"
deploy:
  image: python:latest
  script:
    - pip install awscli
    - aws s3 cp ./ s3://$S3_BUCKET_NAME/ --recursive --exclude "*" --include "*.html"
```

Сборка успешна:

```
Running with gitlab-ci-multi-runner 1.4.2 (bcc1794)
Using Docker executor with image python:latest ...
Pulling docker image python:latest ...
Running on runner-8a2f473d-project-1481102-concurrent-0 via runner-8a2f473d-machine-1471200526-cead9c82-digital-ocean-4gb...
Cloning repository...
Cloning into '/builds/inem/deploy'...
Checking out f9cb2128 as master...
$ pip install awscli
Collecting awscli
  Downloading awscli-1.10.56-py2.py3-none-any.whl (986kB)
Collecting rsa<=3.5.0,>=3.1.2 (from awscli)
  Downloading rsa-3.4.2-py2.py3-none-any.whl (46kB)
Collecting s3transfer<0.2.0,>=0.1.0 (from awscli)
  Downloading s3transfer-0.1.1-py2.py3-none-any.whl (49kB)
Collecting colorama<=0.3.7,>=0.2.5 (from awscli)
  Downloading colorama-0.3.7-py2.py3-none-any.whl
Collecting docutils>=0.10 (from awscli)
  Downloading docutils-0.12-py3-none-any.whl (508kB)
Collecting botocore==1.4.46 (from awscli)
  Downloading botocore-1.4.46-py2.py3-none-any.whl (2.5MB)
Collecting pyasn1>=0.1.3 (from rsa<=3.5.0,>=3.1.2->awscli)
  Downloading pyasn1-0.1.9-py2.py3-none-any.whl
Collecting python-dateutil<3.0.0,>=2.1 (from botocore==1.4.46->awscli)
  Downloading python_dateutil-2.5.3-py2.py3-none-any.whl (201kB)
Collecting jmespath<1.0.0,>=0.7.1 (from botocore==1.4.46->awscli)
  Downloading jmespath-0.9.0-py2.py3-none-any.whl
Collecting six>=1.5 (from python-dateutil<3.0.0,>=2.1->botocore==1.4.46->awscli)
  Downloading six-1.10.0-py2.py3-none-any.whl
Installing collected packages: pyasn1, rsa, docutils, six, python-dateutil, jmespath, botocore, s3transfer, colorama, awscli
Successfully installed awscli-1.10.56 botocore-1.4.46 colorama-0.3.7 docutils-0.12 jmespath-0.9.0 pyasn1-0.1.9 python-dateutil-2.5.3 rsa-3.4.2 s3transfer-0.1.1 six-1.10.0
$ aws s3 cp ./ s3://gitlab-deploy/ --recursive --exclude "*" --include "*.html"
upload: ./index.html to s3://gitlab-deploy/index.html
Build succeeded
```

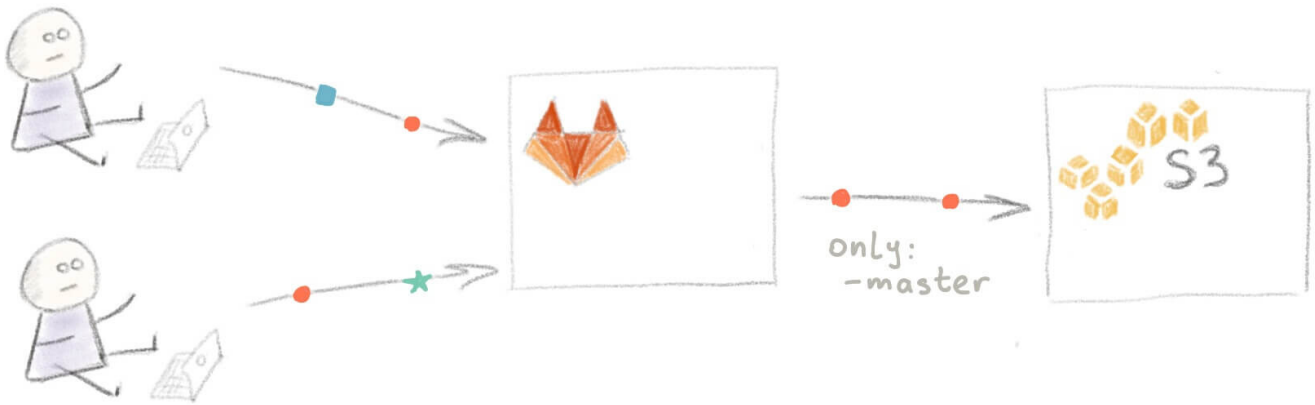
Тем временем посещаемость вашего сайта повысилась, и вы наняли разработчика, чтобы он вам помогал. Теперь у вас есть команда. Посмотрим как работа в команде влияет на рабочий процесс.

Работа в команде

Теперь в одном и том же репозитории работают два человека, и использовать ветку `master` для разработки более не целесообразно. Поэтому вы принимаете решение использовать различные ветки для разработки новых фич и написания статей и мержить их в `master` по мере готовности.

Проблема в том, что при текущей настройке CI не поддерживает работу с ветками — при пуше в любую ветку на GitLab происходит развертывание текущего состояния `master` на S3.

К счастью, это легко исправить — просто добавьте `only: master` в задачу `deploy`.



Мы хотим избежать развертывания каждой ветки на сайте

С другой стороны, было бы неплохо иметь возможность предпросмотра изменений, внесенных из веток для выделенной функциональности (feature-веток).

Создание отдельного пространства для тестирования

Патрик (разработчик, которого вы недавно наняли) напоминает вам, что существует такая функциональность, как GitLab Pages. Как раз то, что нужно — место для предпросмотра новых изменений.

Для размещения вебсайта на GitLab Pages ваша конфигурация CI должна удовлетворять трем простым требованиям:

- Задача должна иметь имя `pages`
- Должен присутствовать раздел `artifacts`, а в нем содержаться папка `public`
- В этой самой папке `public` должно находиться все, что вы хотите разместить на сайте

Содержимое папки `public` будет размещено по адресу `http://<username>.gitlab.io/<projectname>/`

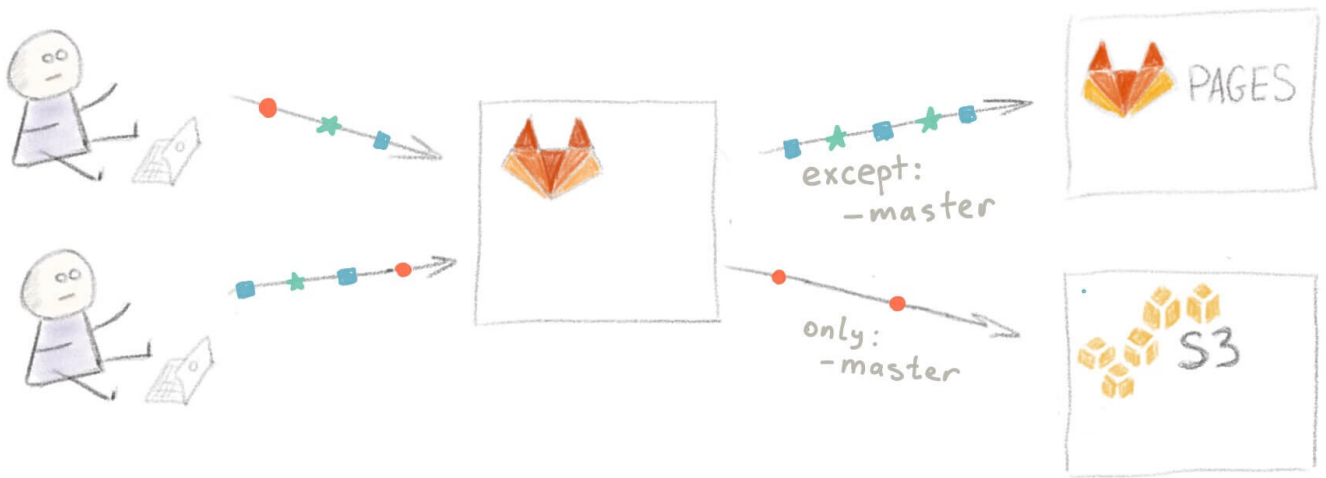
После добавления кода из примера для сайтов на чистом HTML файл настройки CI выглядит следующим образом:

```
variables:
  S3_BUCKET_NAME: "yourbucket"

deploy:
  image: python:latest
  script:
    - pip install awscli
    - aws s3 cp ./ s3://$S3_BUCKET_NAME/ --recursive --exclude "*" --include "*.html"
  only:
    - master

pages:
  image: alpine:latest
  script:
    - mkdir -p ./public
    - cp ./*.html ./public/
  artifacts:
    paths:
      - public
  except:
    - master
```

Всего в нем содержатся две задачи: одна (`deploy`) проводит развертывание сайта на S3 для ваших читателей, а другая (`pages`) на GitLab Pages. Назовем их соответственно "**Production** environment" и "**Staging** environment".



Развертывание всех веток, кроме master, будет проводиться на GitLab Pages

Среды развертывания

GitLab поддерживает работу со средами развертывания. Все, что вам нужно сделать, это назначить соответствующую среду для каждой задачи развертывания:

```
variables:
  S3_BUCKET_NAME: "yourbucket"

deploy to production:
  environment: production
  image: python:latest
  script:
    - pip install awscli
    - aws s3 cp ./ s3://$S3_BUCKET_NAME/ --recursive --exclude "*" --include "*.html"
  only:
    - master

pages:
  image: alpine:latest
  environment: staging
  script:
    - mkdir -p ./public
    - cp ./*.html ./public/
  artifacts:
    paths:
      - public
  except:
    - master
```

GitLab отслеживает все ваши процессы развертывания, так что вы всегда можете увидеть, что в текущий момент развертывается на ваших серверах:

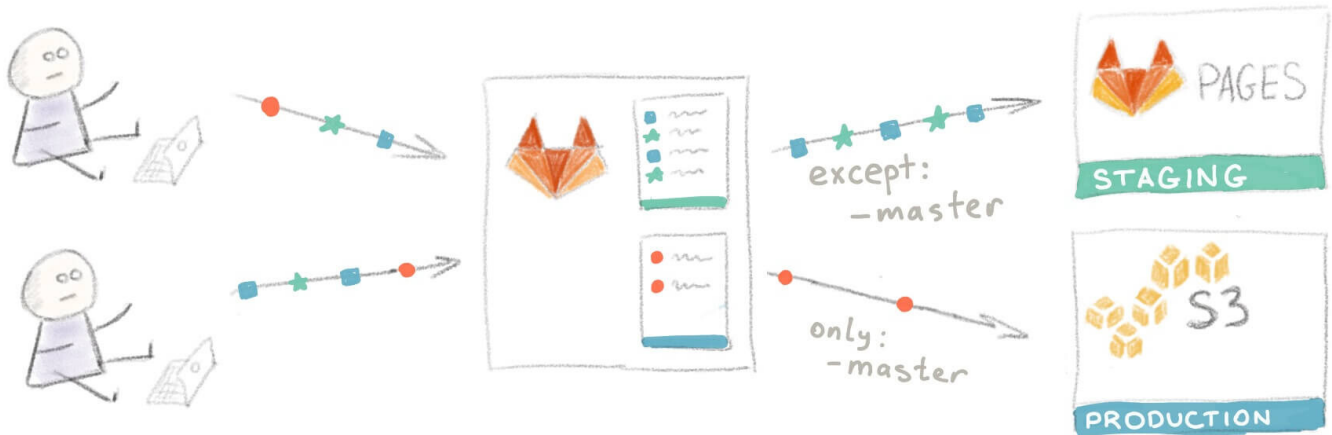
Pipelines Builds Environments		
New environment		
Environment	Last deployment	Date
production	master · 9e09ec35 added environments	2 minutes ago
staging	experiment · 9e09ec35 added environments	5 minutes ago

Также, GitLab хранит полную историю всех развертываний для каждой среды:

Staging

ID	Commit	Build	Date
#3	experiment · 10a96bbb one more experiment	pages (#2944789)	less than a minute ago

#1	experiment · 9e09ec35 added environments	pages (#2944561)	9 minutes ago
----	---	----------------------------------	---------------



Итак, мы оптимизировали и настроили все, что только могли, и готовы к новым вызовам, которые не заставят себя долго ждать.

Работа в команде, часть 2

Опять. Это опять произошло. Стоило вам только запустить свою feature-ветку для превью, как спустя минуту Патрик сделал то же самое и тем самым перезаписал содержимое staging. #\$\$! Третий раз за сегодня!

Идея! Используем Slack для оповещений о развертываниях, чтобы никто не пушил новые изменения, пока старые не закончили развертываться.

Оповещения Slack

Настроить оповещения Slack несложно. Надо лишь взять из Slack URL входящего вебхука...

[Browse Apps](#) > [Custom Integrations](#) > [Incoming WebHooks](#) > [Edit configuration](#)



Incoming WebHooks

Added by inem on Aug 9th, 2016

[Disable](#) • [Remove](#)

Incoming Webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details described later.

[Message Attachments](#) can also be used in Incoming Webhooks to display richly-formatted messages that stand out from regular chat messages.

Setup Instructions

We'll guide you through the steps necessary to configure an Incoming Webhook so you can start sending data to Slack.

[close](#)

Webhook URL

<https://hooks.slack.com/services/XXXXXXXXXXXX/XXXXXXXXXXXX/XXXXXXXXXXXX>

... и передать его в **Settings > Services > Slack** вместе с вашим логином Slack:

→ ☒ **Build**

#general

Event will be triggered when a build status changes

☐ **Wiki page**

#general

Event will be triggered when a wiki page is created/updated

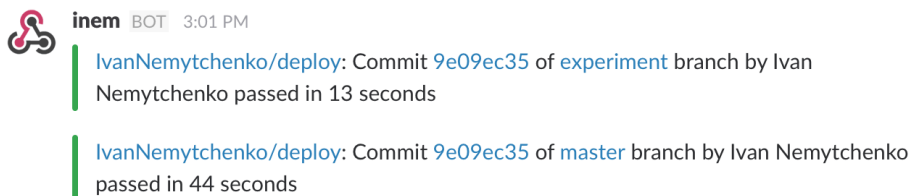
Webhook ←

Username ←

Notify only broken builds ☐

Save changes **Test settings** **Cancel**

Поскольку вы хотите получать уведомления только о развертываниях, в показанных выше настройках можно убрать галочки на всех пунктах, кроме "Build". Вот и все, теперь вы будете получать оповещения о каждом развертывании:



Работа в большой команде

Со временем ваш сайт стал очень популярным, а ваша команда выросла с двух до восьми человек. Разработка происходит параллельно, и людям все чаще приходится ждать в очереди для превью на Staging. Подход "Проводите развертывание каждой ветки на Staging" больше не работает.



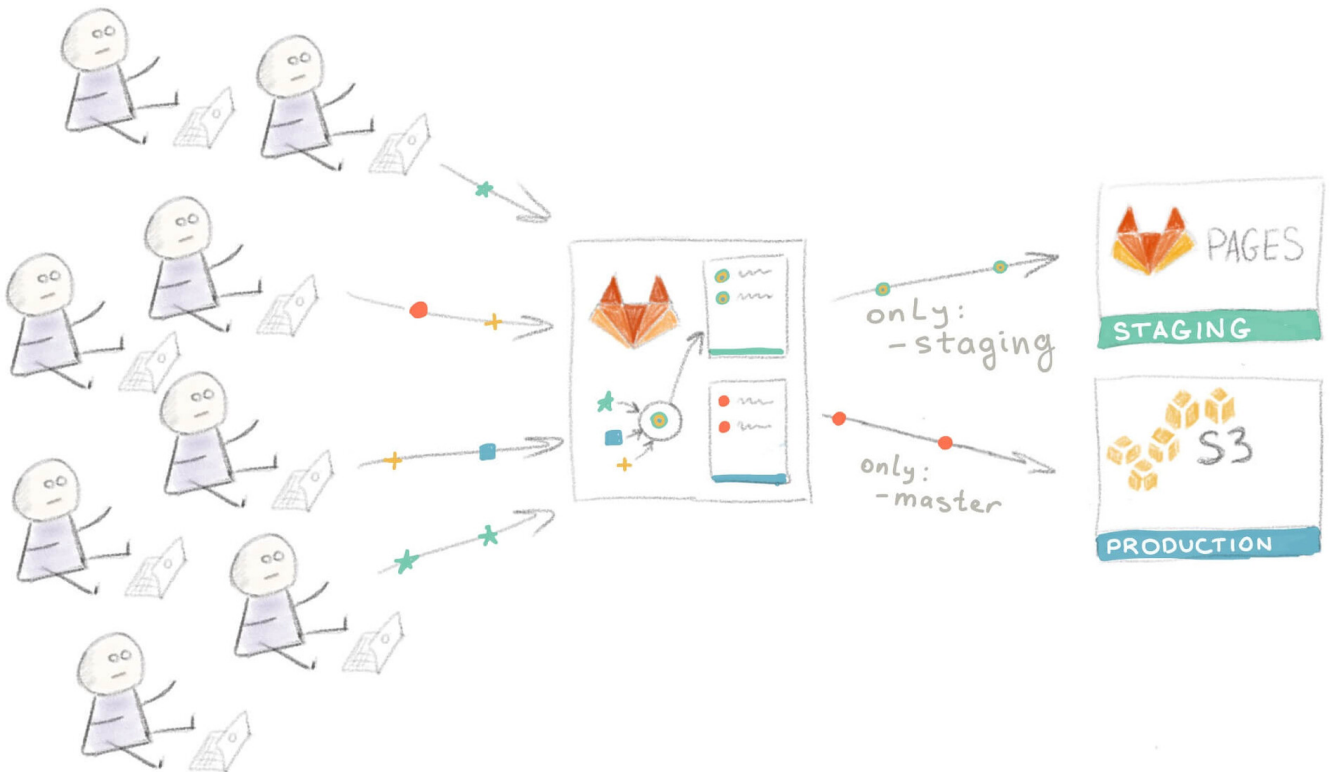
Пришло время вновь модифицировать рабочий процесс. Вы и ваша команда пришли к соглашению, что для выкатывания изменений на staging-сервер нужно сначала сделать мерж этих изменений в ветку "staging".

Для добавления этой функциональности нужно внести лишь небольшие изменения в файл `.gitlab-ci.yml`:

```
except:  
  - master
```


становится

```
only:  
- staging
```



Разработчики проводят мерж своих feature-веток перед превью на Staging

Само собой, при таком подходе на мерж тратятся дополнительное время и силы, но все в команде согласны, что это лучше, чем ждать в очереди.

Непредвиденные обстоятельства

Невозможно все контролировать, и неприятности имеют свойство случаться. К примеру, кто-то неправильно смержил ветки и запустил результат прямо в production как раз когда ваш сайт находился в топе HackerNews. В результате тысячи человек увидели кривую версию сайта вместо вашей шикарной главной страницы.




К счастью, нашелся человек, который знал про кнопку **Rollback**, так что уже через минуту после обнаружения проблемы сайт принял прежний вид.

Production				Destroy
ID	Commit	Build	Date	
#10	master · e9964eb3 production	deploy to production (#3061414)	5 days ago	Re-deploy
#8	master · 9e09ec35 added environments	deploy to production (#3011226)	6 days ago	Rollback

Rollback перезапускает более раннюю задачу, порожденную в прошлом каким-то другим коммитом

Чтобы избежать подобного в дальнейшем, вы решили отключить автоматическое развертывание в production и перейти на развертывание вручную. Для этого в задачу нужно добавить `when: manual`.

Для того, чтобы запустить развертывание вручную, перейдите на вкладку **Pipelines > Builds** и нажмите на вот эту кнопку:

Status	Commit	Stage	Name	
<div> <div>⌚ skipped</div> <div>#3170577  master  7e3f7a6e</div> </div> <div>manual</div>	test	deploy to production		

И вот ваша компания превратилась в корпорацию. Над сайтом работают сотни человек, и некоторые из предыдущих рабочих практик уже не очень подходят к новым обстоятельствам.

Ревью приложений

Следующим логическим шагом является добавление возможности развертывания временного инстанса приложения каждой feature-ветки для ревью.

В нашем случае для этого надо настроить еще один бакет S3, с той лишь разницей, что в этом случае содержимое сайта копируется в “папку” с названием ветки. Поэтому URL выглядит следующим образом:

http://<REVIEW_S3_BUCKET_NAME>.s3-website-us-east-1.amazonaws.com/<branchname>/

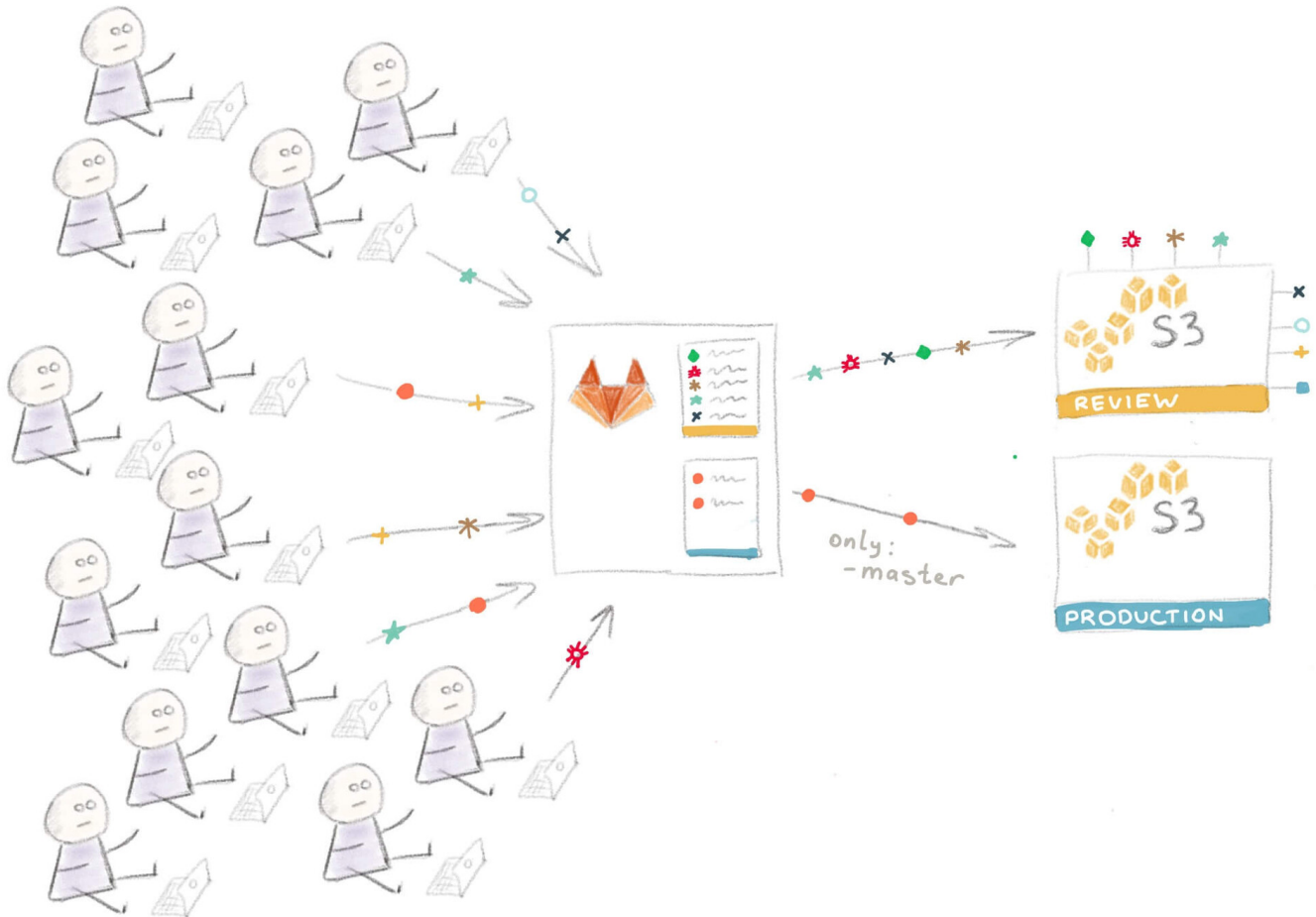
А так будет выглядеть код, замещающий задачу pages:

```
review apps:
  variables:
    S3_BUCKET_NAME: "reviewbucket"
  image: python:latest
  environment: review
  script:
    - pip install awscli
    - mkdir -p ./${CI_BUILD_REF_NAME}
    - cp ./*.html ./${CI_BUILD_REF_NAME}/
    - aws s3 cp ./ s3://${S3_BUCKET_NAME}/ --recursive --exclude "*" --include "*.html"
```

Стоит объяснить откуда у нас появилась переменная `CI_BUILD_REF_NAME` — из списка [предопределенных переменных окружения GitLab](#), которые вы можете использовать для любой своей задачи.

Обратите внимание на то, что переменная `S3_BUCKET_NAME` определена внутри задачи — таким образом можно переписывать определения более высокого уровня.

Визуальная интерпретация такой конфигурации:



Технические детали реализации такого подхода сильно разнятся в зависимости от используемых в вашем стеке технологий и от того, как устроен ваш процесс развертывания, что выходит за рамки этой статьи.

Реальные проекты, как правило, значительно сложнее, чем наш пример с сайтом на статическом HTML. К примеру, поскольку инстансы временные, это сильно усложняет их автоматическую загрузку со всеми требуемыми сервисами и софтом “на лету”. Однако это выполнимо, особенно, если вы используете Docker или хотя бы Chef или Ansible.

Про развертывание при помощи Docker будет рассказано в другой статье. Честно говоря, я чувствую себя немного виноватым за то, что упростил процесс развартывания до простого копирования HTML-файлов, совершенно упуская более хардкорные сценарии. Если вам это интересно, рекомендую почитать статью “Building an Elixir Release into a Docker image using GitLab CI”.

А пока что давайте обсудим еще одну, последнюю проблему.

Развертывание на различные платформы

В реальности мы не ограничены S3 и GitLab Pages; приложения разворачиваются на различные сервисы.

Более того, в какой-то момент вы можете решить переехать на другую платформу, а для этого вам нужно будет переписать все скрипты развертывания. В такой ситуации использование gem’a `dr1` сильно упрощает жизнь.

В приведенных в этой статье примерах мы использовали `awscli` в качестве инструмента для доставки кода на сервис Amazon S3. На самом деле, неважно, какой инструмент вы используете и куда вы доставляете код — принцип остается тот же: запускается команда с определенными параметрами и в нее каким-то образом передается секретный ключ для идентификации.

Инструмент для развертывания `dr1` придерживается этого принципа и предоставляет унифицированный интерфейс для определенного списка плагинов (providers), предназначенных для развертывания вашего кода на разных хостинговых площадках.

Задача для развертывания в production с использованием `dr1` будет выглядеть вот так:

```
variables:
  S3_BUCKET_NAME: "yourbucket"
```

```
deploy to production:
  environment: production
  image: ruby:latest
  script:
  - gem install dpl
  - dpl --provider=s3 --bucket=$S3_BUCKET_NAME
  only:
  - master
```

Так что если вы проводите развертывание на различные хостинговые площадки или часто меняете целевые платформы, подумайте над использованием dpl в скриптах развертывания — это способствует их единообразию.

Подводя итоги

1. Развертывание происходит при помощи регулярного выполнения команды (или набора команд), так что оно может запускаться в рамках GitLab CI
2. В большинстве случаев этим командам нужно передавать различные секретные ключи. Храните эти ключи в разделе **Settings** > **Variables**
3. С помощью GitLab CI вы можете выбирать ветки для развертывания
4. GitLab сохраняет историю развертываний, что позволяет делать откат на любую предыдущую версию
5. Существует возможность включения развертывания вручную (а не автоматически) для наиболее важных частей инфраструктуры проекта

Перевод с английского выполнен переводческой артелью «Надмозг и партнеры», <http://nadmosq.ru>. Над переводом работал @sgnl_05.

Теги: git, gitlab, gitlab-ci, gitlab ci, continuous integration, ci, deploy, deployment, deployment tools

↑ +26 ↓ 245 👁 71,4k 💬 3



Softmart 0,00
Компания



49,0 **75,2** **54** **10**
Карма Рейтинг Подписчики Подписки

Иван Немытченко @nem
Smartprogrammer.ru

Поделиться публикацией

ПОХОЖИЕ ПУБЛИКАЦИИ

13 апреля 2017 в 17:18

Вышел GitLab 9.0: Подгруппы и Deploy Boards

↑ +30 👁 11,4k 📖 36 💬 9

3 января 2017 в 10:37

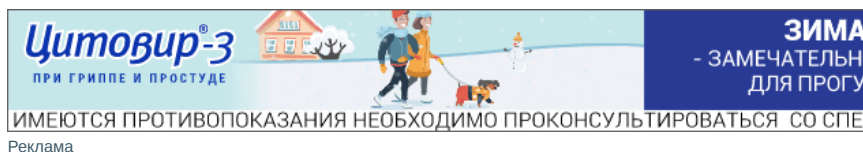
Вышел GitLab 8.15

↑ +27 👁 14k 📖 66 💬 13

7 сентября 2016 в 10:35

Введение в GitLab CI





Комментарии 3

levkin 20 сентября 2016 в 21:05 0

Подскажите, можно ли разрешить нажимать кнопку Deploy to production, например, только мастеру? И чтобы Reporter, Developer ее вообще не видели.

evgen 21 сентября 2016 в 20:04 0

Да, и как защитится от случайного изменения файла .gitlab-ci.yml на произвольных ветках? Например от удаления only: master в фичер ветке для деплоя и получения развертывания этой фичи в продакшене.

Да, есть роллбэк, но все равно неприятно.

nem 21 сентября 2016 в 21:39 0

Пока нет, это в разработке — должно быть выкачено в одном из ближайших релизов.

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

САМОЕ ЧИТАЕМОЕ

Сутки

Неделя

Месяц

Как американцы живого хорька в коллайдер засунули

+63 33,4k 36 37

Что айтишнику не стоит делать в 2020?

+33 29,2k 93 58

В Windows 10 версии 2004 можно отслеживать температуру видеокарты, а новые драйверы будут помечать как обновления

+15 21,7k 2 37

Еще один способ высокотехнологичного мошенничества

+73 35,8k 79 91

Прионы — страх и ужас будущего

+78 27,2k 85 119

Войти	Публикации	Правила	Реклама
Регистрация	Новости	Помощь	Тарифы
	Хабы	Документация	Контент
	Компании	Соглашение	Семинары
	Пользователи	Конфиденциальность	Мегaproекты
	Песочница		

Если нашли опечатку в посте, выделите ее и нажмите Ctrl+Enter, чтобы сообщить автору.

© 2006 – 2019 «ТМ»



Настройка языка

О сайте

Служба поддержки

Мобильная версия

