Хочу поделиться информацией на тему работы с системой управления репозиториями git. Я расскажу, как установить и настроить gitlab, а так же на примерах покажу как ей пользоваться. В интернете информации много, но чаще всего она написана не для тех, кто никогда не работал с гитлаб и с гит. Я же постараюсь объяснить максимально просто для тех, кто знакомится с системой гит впервые.

Содержание:

Что такое Gitlab, его возможности

Gitlab — это сервис для работы с системой контроля версий **git**. Использовать gitlab можно непосредственно на сайте gitlab.com как SAAS сервис, зарегистрировав аккаунт. Так же вы можете установить gitlab на свой сервер и использовать по своему усмотрению. Об этом способе дальше пойдет речь в статье.

Gitalb представляет из себя web приложение, состоящее из нескольких компонентов:

- Ruby
- Go
- Nodejs
- База данных (PostgreSQL или MySQL)
- Redis
- Nginx

Так какие же возможности предоставляет gitlab и какие задачи решает? В первую очередь это инструмент для разработчиков. Он предоставляет следующие возможности:

- Управление приватными и публичными git репозиториями
- Управление пользователями, группами и правами доступа к репозиториям
- Анализ кода, отслеживание ошибок, деплой.
- Интеграция с различными системами CI (Jenkins и т.д.), а так же организация самостоятельного процесса CI с помощью встроенных средств

Это основное, что пришло в голову. Так же в gitlab реализован функционал api, wiki страниц, комментариев к проектам, отслеживание изменений, доски идей и задач и другое. Подробно все возможности перечислены в соответствующем разделе на официальном сайте проекта.

Я не очень подробно знаком с gitlab и с системой git в целом. Чаще всего она нужна разработчикам и тем, кто сопровождает их работу (devops). Я покажу на простых примерах, как ее может использовать в своих задачах системный администратор или обычный веб разработчик.

Сравнение gitlab с github и остальными

Коротко пройдусь по отличиям gitlab от остальных платформ управления git репозиториями, которые в последнее время стали стандартом, вытеснив все остальные системы контроля версий. Сразу предупреждаю, что судить буду по своим поверхностным знаниям указанных систем, так как плотно с ними не работал. Но так как системы известные, некоторой информацией в любом случае обладаю.

gitlab vs github

Основной конкурент gitlab это сайт и сервис **github**. Он появился гораздо раньше, поэтому вполне логично, что обладает гораздо большей популярностью. На сегодняшний день это сайт номер один для размещения open source проектов. Они, по-моему, все на нем размещаются. Вот его основные преимущества:

- Огромное комьюнити. Очень часто git напрямую ассоциируется с github.
- Богатая интеграция сторонних сервисов. По сути, все поддерживают работу с github.
- Все в одном месте. Имея аккаунт на github ты получаешь доступ практически ко всем опенсорс проектам. Можешь без проблем их форкать, контрибьютить и т.д.
- Аккаунт на гитхабе иногда просят показать при устройстве на работу. Актуально для программистов.

Минусов по сути только два, но существенные:

- На бесплатном аккаунте нет возможности создавать приватные репозитории. Они все будут с публичным доступом.
- Нет возможности установить локальную версию.

Еще из минусов для организаций я слышал о невозможности зарегистрировать аккаунт на организацию. Привязка идет к пользователю, что не очень удобно.

То есть в целом github более старый, зрелый, популярный сервис, который все знают и используют. Gitlab же появился только в 2011 году.

gitlab vs bitbucket

Сервис **BitBucket** не очень популярен у нас. Многие его вообще не знают, но насколько я слышал, он популярен на западе. Он позиционирует себя как сервис для небольших команд разработчиков. В качестве плюсов у него есть только парочка по сравнению с github:

- Возможность создавать неограниченное количество бесплатных репозиториев.
- Платная версия в целом дешевле стоит, чем у github.

Из недостатков по сравнению с gitlab только один:

- Бесплатная self-hosted версия позволяет работать не более, чем пяти пользователям.

В целом, у BitBucket вроде как интерфейс пошустрее, приятнее работать. Сам я никогда им не пользовался и даже аккаунта там нет.

Все три продукта зрелые, проверенные, известные. Принципиально gitlab отличается только возможностью установки на свои сервера. В остальном функционал схожий и чаще всего достаточный для большинства разработчиков. Более подробно нужно смотреть на нюансы, если они у вас есть, и на возможность интеграции с нужными вам продуктами. Непосредственно как системы контроля версий они все хороши и достаточны.

Установка Gitlab

Существует несколько способов установки Gitlab:

1. С помощью deb/rpm пактов из репозитория разработчика.
2. Ручная сборка из исходников.
3. Установка в docker контейнере.

Проще всего, к тому же рекомендуется разработчиком, установка готовых пакетов из репозитория. Второй способ подойдет для тех, кто хочет установить Gitlab на систему, для которой нет готовых пакетов. Например, на Freebsd. Процесс этот не сильно сложный, но утомительный, так как gitlab состоит из множества компонентов, которые нужно будет установить и настроить по отдельности, а потом связать между собой. Третий способ установки через docker контейнер мне вообще не понятен, так как идеология докера — один сервис, один контейнер. А тут будет целая куча сервисов в одном контейнере. В таком случае мне кажется более удобным использовать lxc контейнер, а не docker.

Я буду устанавливать Gitlab первым способом с помощью готовых пакетов из репозитория. Отдельно остановимся на системных требованиях.

Системные требования

Официально gitlab поддерживает установку на следующие операционные системы:

1. Debuan, Ubuntu
2. RHEL и производные (CentOS, Oracle Linux, Scientific Linux)
3. openSUSE

На все остальные системы нужно пробовать собирать из исходников, но успешный результат не гарантируется.

Системные требования по железу:

1. **CPU**. Рекомендуется 2 ядра. На одном тоже будет работать, но разработчики предупреждают, что может тормозить при работе каких-нибудь воркеров или задач в фоне. Для старта и знакомства достаточно будет одного ядра.
2. **Memory**. Установить gitlab на сервер можно только если у него 2 и более гб оперативной памяти. Если меньше, установщик сообщит об ошибке и прекратит работу. Работа с 2 Гб памяти возможна только в тестовом или ознакомительном режиме. Все будет сильно тормозить. Для полноценной работы надо от 4 гб памяти, лучше 8.
3. **Storage**. Здесь никаких рекомендаций и ограничений нет. Все будет зависеть от сценария использования.
4. **Database**. Gitlab поддерживает работу с двумя типами БД — PostgreSQL и MySQL. При этом настоятельно рекомендуется использовать PostgreSQL. Она же ставится по-умолчанию.

Установка на Centos

Устанавливаем зависимости, необходимые для работы gitlab:

```
# yum install curl policycoreutils-python postfix
```

Подключаем репозиторий gitlab. Для этого используется скрипт, который определяет версию системы и в зависимости от нее подключает нужный репозиторий:

```
# curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ee/script.rpm.sh | sudo bash
```

Запускаем непосредственно установку, указав адрес будущего сайта. Я для примера создал поддомен gl.serveradmin.ru, который буду использовать в

дальнейшем в статье.

```
# EXTERNAL_URL="http://gl.serveradmin.ru" yum install gitlab-ee
```

Установка на Ubuntu

Устанавливаем зависимости, необходимые для работы gitlab:

```
# apt-get install curl ca-certificates postfix
```

Подключаем репозиторий gitlab. Для этого используется скрипт, который определяет версию системы и в зависимости от нее подключает нужный репозиторий:

```
# curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ee/script.deb.sh | sudo bash
```

Запускаем непосредственно установку, указав адрес будущего сайта. Я для примера создал поддомен gl.serveradmin.ru, который буду использовать в дальнейшем в статье.

```
# EXTERNAL_URL="http://gl.serveradmin.ru" apt install gitlab-ee
```

После завершения установки, идем по указанному ранее адресу. Первым делом нужно установить пароль для администраторской учетки root.

После того, как это сделаете, можете зайти в систему.

Если у вас после успешной установки не загружается страница в браузере, проверьте firewall. Либо настройте его по моей инструкции, либо просто отключите, если не умеете или не хотите настраивать. Доступ к gitlab осуществляется по 80 или 443 порту, как к обычному сайту.

На этом установка закончена. Дальше рассмотрим основные настройки, прежде чем приступить к работе с системой.

Настройка Gitlab

nginx proxy_pass и ssl

Мы выполнили дефолтную установку. В настоящее время Gitlab работает по протоколу http. Если вы его только тестируете, то можете так и оставить. Если же планируете пользоваться в дальнейшем, то лучше настроить https, хотя бы для того, чтобы избавиться от назойливых предупреждений браузеров. Я изначально не стал выполнять установку на https адрес, так как может быть 2 варианта работы web сайта:

1. У вас одиночный сервер или виртуальная машина с белым ip адресом. На этом ip адресе кроме gitlab больше ничего нет.
2. У вас нет отдельного ip адреса для gitlab, он работает в локальной сети с серыми ip за шлюзом, который проксирует соединения в зависимости от домена на разные сервера.

Лично у меня чаще всего вторая ситуация. Я предпочитаю, когда возможно, держать внешний сервер с firewall и nginx, который проксирует запросы на целевые серверы. Это актуально, когда вы арендуете дедик и делите его на виртуальные машины.

В обоих случаях будем использовать бесплатные сертификаты от let’s encrypt. Если у вас первый случай с одиночным сервером, то выполните пару простых действий для настройки https в gitlab. В конфиге */etc/gitlab/gitlab.rb* измените следующие параметры:

```
letsencrypt['enable'] = true
external_url "https://gl.serveradmin.ru"
letsencrypt['contact_emails'] = ['zeroxzed@gmail.com']
```

И перезапустите gitlab:

```
# gitlab-ctl reconfigure
```

После этого сразу же заработает https с автоматическим редиректом с http. Больше ничего делать не надо.

Во втором случае настраивать https на самом сервере с гитлаб не обязательно. Сделаем это на сервере с nginx, который работает в режиме proxy_pass. Для начала получите сертификат, как описано у меня в инструкции по настройке web сервера. Затем используйте следующий конфиг для nginx:

```
server {
        listen 80;
        server_name gl.serveradmin.ru;
        return 301 https://$server_name$request_uri;
}
server {
        listen 443 ssl http2;
        server_name gl.serveradmin.ru;
        access_log /var/log/nginx/gl.serveradmin.ru-access.log;
        error_log /var/log/nginx/gl.serveradmin.ru-error.log;

        ssl on;
        ssl_certificate /etc/letsencrypt/live/gl.serveradmin.ru/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/gl.serveradmin.ru/privkey.pem;
        ssl_session_timeout 5m;
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        ssl_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';
        ssl_prefer_server_ciphers on;
        ssl_session_cache shared:SSL:10m;

location /.well-known {
        root /tmp;
        }

location / {
        proxy_pass http://10.10.10.10:80;
        proxy_read_timeout      300;
        proxy_connect_timeout   300;
```

```
        proxy_redirect          off;
        proxy_set_header        X-Forwarded-Proto $scheme;
        proxy_set_header        Host              $http_host;
        proxy_set_header        X-Real-IP         $remote_addr;
        proxy_set_header        X-Forwarded-For   $proxy_add_x_forwarded_for;
        proxy_set_header        X-Frame-Options   SAMEORIGIN;
    }
}
```

Создайте директорию для перевыпуска сертификатов:

```
# mkdir /tmp/.well-known && chown nginx. /tmp/.well-known
```

Не забудьте добавить задание в cron на перевыпуск сертификата. У меня это описано в статье про веб сервер, о которой я говорил выше.

ошибка 502 в gitlab

Частенько пользователи gitlab сталкиваются с ошибкой 502 (Whoops, GitLab is taking too much time to respond.). Вам ее показывает nginx. В общем случае это означает, что веб сервер не смог получить ответ от бэкенда. В случае с gitlab бэкендом выступает unix сокет — */var/opt/gitlab/gitlab-workhorse/socket*.

> Для справки. Конфигурация nginx для gitlab, как и многие другие, располагается по адресу /var/opt/gitlab, конкретно nginx вот тут — /var/opt/gitlab/nginx/conf.

Почему не отвечает бэкаенд и вылезает 502 ошибка наверняка сказать нельзя. Вот самые простые и очевидные причины:

1. У вас мало оперативной памяти на сервере. Если ее только 2 Гб, то ошибку 502 вы можете видеть время от времени, даже если работаете с gitlab один. Для работы nginx, redis, postgresql и прочих элементов гитлаба надо много памяти. Так что если у вас ее 2 гигабайта, просто увеличьте до 4-х и проверьте результат. Как вариант, можете увеличить или включить swap, если у вас его совсем не было.
2. У вас упала служба gitlab-workhorse, которая открывает сокет, который слушает nginx. Почему она упала — отдельный вопрос. Или почему она работает, а сокета нет. Попробуйте просто перезагрузить сервер. Если повторяться ошибка не будет, можете считать, что решили проблему. Одной

из причин падений сервиса может быть занятый порт какой-то службы, которая относится к gitlab. Это может случиться, если на сервере, помимо гитлаба работают другие службы. Возможно ошибки в конфиге или проблемы после обновления.
3. По какой-то причине могли измениться права доступа к сокету /var/opt/gitlab/gitlab-workhorse/socket, и nginx не может получить к нему доступ. Исправьте это. Посмотрите от какого пользователя работает nginx и убедитесь в том, что у него хватает прав для доступа к сокету.

Возможно, есть еще какие-то причины появления ошибки 502 в gitlab. Я рассмотрел самые основные, которые покрывают большинство случаев.

Как пользоваться Gitlab

Перейдем к практике. Мы установили и настроили gitlab. Теперь посмотрим, как с ним работать. Это очень функциональный инструмент и в двух словах тут все не расскажешь. Я рассмотрю самые азы использования гитлаб для начинающих, чтобы можно было просто познакомиться и понять, как и где его можно использовать.

Документация на русском

У gitlab есть неплохая официальная документация — https://docs.gitlab.com/ee/README.html. Там есть вообще все, о чем я тут рассказываю и буду рассказывать дальше. Если у вас есть время — изучайте ее. Более подробной информации я нигде не видел. К сожалению, документация только на английском языке. Хотя какое тут сожаление — это обычное дело. Без английского языка в IT никуда.

Я пытался найти переводы, возможно книги на тему гиталаба на русском языке, но не нашел ничего. Так что можно считать, что за исключением некоторых статьей в интернете, на русском языке о gitlab практически ничего нет. В том числе и поэтому я задумал написать эту подробную статью. Обычно я пишу о том, что не смог сам найти в нужно мне виде и стал собирать по частям из разных мест и разбираться самостоятельно.

Если у вас есть какая-то информация о книгах по gitlab на русском языке, поделитесь информацией. По git такие есть, конкретно по гитлабу не видел. Интересно было бы самостоятельно освоить CI на gitlab. Пока приходится искать информацию со всяких конференций и презентаций.

Настройка ssh доступа по ключам

Прежде чем приступить к работе, добавим свой ssh ключ в gitlab. Он нам нужен для того, чтобы потом с сервера без ручной авторизации подключаться к репозиториям гитлаба.

Подключаемся к серверу или рабочей машине, с которой мы будем работать с удаленными репозиториями гитлаба. Нам нужно сгенерировать новые ключи

для ssh. Если у вас уже есть пара приватного и публичного ключей, которые вы хотите использовать, то можете пропустить этот пункт. Я рассмотрю момент создания отдельного ключа именно для работы с gitlab. Генерируем пару ключей:

```
# ssh-keygen -t rsa -f ~/.ssh/gitlab
```

```
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/gitlab.
Your public key has been saved in /root/.ssh/gitlab.pub.
The key fingerprint is:
SHA256:X6Y67XZimkGR4321lqxoaXf7eGgB7LSKOZyHQKA root@serveradmin.ru
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|  .      .       |
| . .    o.       |
|E   .    o+o o   |
|   .    So.o@ .  |
|    .  o =o&..   |
|     o =BoB +o   |
|      Bo+==.+o.  |
|       o+*.+oo.  |
+----[SHA256]-----+
```

Указывать или нет пароль на сертификат, решайте сами. Все зависит от ситуации и вашей параноидальности. При указании пароля, его нужно будет вводить каждый раз, когда будете использовать ключ для авторизации.

Если у вас это единственный ключ в директории .ssh, то больше ничего делать не надо. В случае, если у вас их несколько, то надо создать конфигурационный файл и указать, для какого ресурса какой ключ использовать. Для этого создаем файл *~/.ssh/config* примерно следующего содержания:

```
# mcedit ~/.ssh/config
```

```
Host gl.serveradmin.ru
    IdentityFile /root/.ssh/gitlab
    port 22
```

Порт 22 указывать не обязательно. Я его указал для примера, чтобы в случае использования нестандартного порта, вы его там указали. В дальнейшем не придется каждый раз указывать порт.

Теперь копируем содержимое файла *~/.ssh/gitlab.pub* и идем в web интерфейс gitlab.

```
# cat ~/.ssh/gitlab.pub
```

Открываем раздел **Setting -> SSH Keys** и вставляем в поле key наш ключ.

С авторизацией по ssh закончили. Можно будет ее использовать в дальнейшем.

Создать и удалить проект

Давайте создадим наш первый проект в gitlab. Для этого достаточно на главной странице нажать на **Create a Project**, либо в любом месте сайта в верхнем меню нажать на **+**.

Я для примера создам проект wordpress, куда позже зальем исходники сайта, с которым будем работать. Для создания проекта достаточно просто указать его название и тип репозитория:

1. Private — доступ только у вас
2. Internal — доступ только у пользователей вашего сайта

3. Public — публичный доступ для всех без аутентификации.

Галочку **Initialize repository with a README** пока не нажимайте, так как дальше мы инициализируем репозиторий в другом месте и зальем в этот проект.

Проект создали. Теперь покажу, как его удалить в случае необходимости. Там так запрятали этот функционал, что совсем не очевидно, как удалить проект.

Для удаления проекта в gitlab, вам нужно зайти в сам проект. В левом меню выбрать раздел **Settings -> General**, в самом низу в блоке **Advanced** нажать на **Expand** и там уже выбрать **Remove Project**.

Двигаемся дальше в изучении работы в gitlab.

Добавление пользователя к проекту

Посмотрим, как добавить пользователя в проект в gitlab. Для этого открываем проект. В левом меню выбираем раздел **Settings -> Members**. Здесь вы можете проверить доступ других пользователей к проекту. При необходимости, добавить нового. Сделаем это. В поле **Select members to invite** выберите пользователя, укажите права доступа к проекту и срок, на который вы выдаете эти права и жмите **Add to Project**.

Все, пользователю добавлен доступ на просмотр к вашему проекту с ограничением доступа по времени.

Как загрузить или залить проект

Переходим к самому интересному. Посмотрим, как загрузить или выложить наш существующий проект на gitlab. Я для примера решил взять сайт на wordpress и выложить его исходники на gitlab. В дальнейшем изменение сайта будет фиксироваться через систему контроля версий git и заливаться на gitlab.

Удивительно, но в наше время далеко не все разработчики ведут свои проекты в git. Я постоянно вижу, как сайты на wordpress или bitrix правят прям на

живую, а потом не могут понять, чего понаделали и почему не работает. Спасает только регулярный инкрементный бэкап, если вовремя спохватиться и начать искать багнутые изменения за какую-то дату. Надеюсь мое руководство будет полезно для разработчиков, которые захотят жить по-новому и начнут работать с git :)

Все, что пойдет дальше, не имеет прямого отношения к gitlab. С ним мы фактически закончили. Дальше мы будем работать с локальным репозиторием git и заливать его на gitlab. Использоваться будут исключительно git команды. По самому git в интернете очень много информации, в том числе хороших книг.

Идем на сервер, где у нас располагается сайт. В моем случае это директория */web/sites/serveradmin.ru/www*. Возьму для примера копию своего сайта на резервном сервере. Если у вас никогда не было репозитория в этой директории, то выполним его инициализацию:

```
# git init
Initialized empty Git repository in /web/sites/serveradmin.ru/www/.git/
```

Если у вас на сервере не установлен git, сделайте это следующими командами, в зависимости от дистрибутива:

```
# yum install git
```

```
# apt install git
```

Теперь посмотрим статус нашего репозитория:

```
# git status
```

Вы увидите фразу **Untracked files** и дальше список файлов в директории. В настоящее время репозиторий пустой, нет отслеживаемых файлов. Нам надо добавить туда наш сайт, но для начала надо создать список исключений, так как не все файлы нужно добавлять в репозиторий. Там не нужны медиафайлы, временные файлы, кэш и т.д. То есть все то, что не является исходником сайта и не будет изменяться. То, за чем не нужен контроль версий.

Для задания списка исключений git создадим в корне сайта файл .gitignore примерно такого содержания.

```
# mcedit /web/sites/serveradmin.ru/www/.gitignore
```

```
wp-content/uploads/
wp-content/cache/
wp-content/gallery/
```

Список исключений для каждого проекта будет свой. Рекомендую внимательно отнестись к его составлению. Теперь можно добавить все остальные файлы сайта в репозиторий. Делается это следующей командой, выполненной из корня сайта.

```
# git add .
```

Теперь в выводе команды git status вы увидите перечисленными все файлы вашего сайта со статусом **new file**. Выполним первый commit и запишем изменения репозитория. А он изменился — был пустой, а стал наполнен файлами.

```
# git commit -m "Add full site"
```

Давайте зальем исходники сайта в gitlab репозиторий. Но перед этим добавим ссылку на удаленный репозиторй, чтобы с ним было проще работать. Делается это вот так:

```
# git remote add wordpress git@gl.serveradmin.ru:root/wordpress.git
```

Адрес git@gl.serveradmin.ru:root/wordpress.git взят из информации о проекте в gitlab.

Посмотрим информацию о добавленном репозитории:

```
# git remote show wordpress
* remote wordpress
  Fetch URL: git@gl.serveradmin.ru:root/wordpress.git
  Push  URL: git@gl.serveradmin.ru:root/wordpress.git
  HEAD branch: (unknown)
```

При первом подключении вас попросят подтвердить отпечаток ECDSA ключа. Теперь давайте загрузим локальный проект, который мы только что создали в удаленный репозиторий на gitlab.

```
# git push -u wordpress master
```

Исходники залились в репозиторий. Их можно посмотреть через браузер.

Как скачать проект

Помимо контроля изменений с использованием удаленных репозиториев git в gitlab вы получаете возможность очень быстро загрузить к себе исходники сайта и начать с ними работать. Для этого достаточно настроить подключение к репозиторию по ssh ключам, как я показал на примере заливки проекта и потом просто скачать проект одной командой:

```
# git clone git@gl.serveradmin.ru:root/wordpress.git
```

Вы можете открывать доступ к проекту другим людям, делать его публичным и т.д. В общем, с gitlab можно существенно упростить себе работу. Я, к примеру, использую его для хранения конфигов на все случаи жизни. Когда мне нужен какой-то конфиг, я просто копирую его из репозитория и вношу необходимые изменения на месте. В то же время я могу править свои конфиги и получать к ним доступ прямо из браузера.

Особо замороченные люди добавляют в репозитории сразу всю директорию /etc и имеют контроль изменений за всеми конфигами. Я так обычно не делаю. В каких-то случаях это может и оправданно, но лично мне достаточно вручную сохранить копию конфига и поставить в названии дату изменений. Все такие конфиги хранятся в отдельной директории на сервере.

Скачать проект с gitlab можно и напрямую через web интерфейс. Для этого откройте проект и выберите соответствующий пункт меню.

Backup и восстановление gitlab

Очень подробно процесс бэкапа и восстановления gitlab рассмотрен в документации. Я лишь сделаю краткую выжимку из информации оттуда.

У gitlab есть встроенный механизм создания резервной копии. Подготовить бэкап можно командой:

```
# gitlab-rake gitlab:backup:create
```

По-умолчанию бэкап будет создан в директории */var/opt/gitlab/backups*, откуда вы его можете переместить в другое место.

Существуют разные стратегии бэкапа. По-умолчанию, данные сразу упаковываются с помощью tar. Этот процесс не моментальный и занимает какое-то время. Если в этот момент с данными активно работать, то файлы будут изменены, а tar будет выдавать предупреждения. Чтобы этого избежать, можно указывать STRATEGY=copy, например вот так:

```
# gitlab-rake gitlab:backup:create STRATEGY=copy
```

В таком случае данные будут сначала копироваться во временную директорию, а потом упаковываться таром. По идее, так лучше, но требуется больше свободного места на сервере для успешного бэкапа. Это актуально для крупных серверов.

Директорию, куда бэкапить, можно указать в конфиге примерно вот так:

```
gitlab_rails['backup_upload_connection'] = {
  :provider => 'Local',
  :local_root => '/mnt/backups'
}

# The directory inside the mounted folder to copy backups to
# Use '.' to store them in the root directory
gitlab_rails['backup_upload_remote_directory'] = 'gitlab_backups'
```

В данном случае */mnt/backups* не обязательно локальная папка. Это может быть и сетевая папка nfs или smb, примонтированная к серверу. Можете использовать локальную директорию, а потом с помощью rsync куда-то передавать данные.

Помимо локальных директорий для бэкапа, поддерживается загрузка в некоторые облачные хранилища. Подробнее об этом написано в документации.

Можно указать параметр времени хранения бэкапа. Он работает только с локальными архивами. При достижении этого срока, gitlab сам будет удалять старые архивы.

```
## Limit backup lifetime to 7 days - 604800 seconds
gitlab_rails['backup_keep_time'] = 604800
```

Для регулярного выполнения backup данных gitlab можно создать задачу в cron. Вот пример, для регулярного резервного копирования в 2 часа ночи:

```
0 2 * * * /opt/gitlab/bin/gitlab-rake gitlab:backup:create CRON=1
```

Обязательно нужно учесть, что таким образом созданные бэкапы не включают в себя сам файл конфигурации gitlab и ключей ssh, которые хранятся в */etc/gitlab*. Эту директорию нужно бэкапить отдельно.

Для восстановления gitlab из бэкапа, вам необходимо установить чистую gitlab той же версии, что есть в архивной копии. Убедиться в том, что она работает. Затем восстановить вручную файл */etc/gitlab/gitlab-secrets.json*. Далее копируете файл с бэкапом в директорию */var/opt/gitlab/backups* и назначаете пользователя git владельцем файла. Дальше выполняете команды:

```
# gitlab-ctl stop unicorn
# gitlab-ctl stop sidekiq
# gitlab-ctl status
# gitlab-rake gitlab:backup:restore BACKUP=1493107454_2018_04_25_10.6.4-ce
```

После этого восстановите конфиг */etc/gitlab/gitlab.rb*. Перезапустите gitlab и выполните проверку.

```
# gitlab-ctl restart
# gitlab-rake gitlab:check SANITIZE=true
```

На этом восстановление закончено. Если все прошло штатно, то вы получите забэкапленную версию gitlab. Если будете пробовать восстановить на другую

версию, отличную от той, что была в бэкапе, гарантированно получите ошибку. Версия должна точно совпадать.

Обновление Gitlab

Обновление такой сложной структуры, как gitlab может оказаться не таким простым, как видится на первый взгляд. Есть много моментов, которые могут привести к ошибке. Так что процесс обновления продуктовых серверов я настоятельно рекомендую сначала отлаживать в тестовой среде, а потом пробовать обновить рабочую версию.

В общем случае все выглядит достаточно просто. Необходимо подключить репозиторий со свежей версией и выполнить обновление через менеджер пакетов, не забыв перед этим сделать как минимум бэкап.

Обновление gitlab:

```
# Debian/Ubuntu
sudo apt-get update
sudo apt-get install gitlab-ce

# Centos/RHEL
sudo yum install gitlab-ce
```

На деле, я подозреваю, на крупных и нагруженных серверах могут быть проблемы. Как минимум, нужно отключить сервер от пользователей и спокойно провести протестированное заранее обновление. В тестовой среде вы скорее всего все проверите без нагрузки. Поэтому на всякий случай, с продуктового сервера тоже снимите нагрузку. Нужно запланировать время простоя на процедуру обновления.

Заключение

> Не понравилась статья и хочешь научить меня администрировать? Пожалуйста, я люблю учиться. Комментарии в твоем распоряжении. Расскажи, как сделать правильно!

Статья получилась большая, объемная. Когда начинал писать, не планировал такой размах. Я не так уж хорошо знаю gitlab, работал с ним немного, поэтому где-то мог ошибиться или посоветовать то, что на самом деле неправильно. Если вы заметите такие моменты или захотите что-то посоветовать, сделайте это в комментариях.

Дальше я хочу рассмотреть вопрос CI с помощью gitlab, но на это надо время и тесты. Пока я только в общих чертах представляю себе этот процесс. Знаю о теоретических возможностях и методах, но не было времени проверить на практике.