Непрерывная интеграция

Непрерывная интеграция (CI, англ. Continuous Integration) — это практика разработки программного обеспечения, которая заключается в слиянии рабочих копий в общую основную ветвь разработки несколько раз в день и выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем.

Другими словами, это *набор автоматизированных действий* по проверке результата работы разработчиков с формированием сборки программного проекта в вид готовый к поставке в рабочее окружение, то есть создание **артефакта**.

Артефакт - результат работы системы автоматизированной сборки и проверки. Представляет собой готовую и протестированную систему, а так же все сопутствующие материалы (ресурсы, документация).

Автоматизация сборки проекта

Конвейер автоматической интеграции состоит из следующих частей:

- **Репозиторий**. Git, Mercurial, Subversion, TFS, и прочие
- **Управляющая программа**. GoCD, TeamCity, Jenkins, Buildbot, и пр. Краткий обзор управляющих программ можно прочитать по следующей ссылке.

  https://techrocks.ru/2019/03/11/10-continuous-integration-systems/
- **Система сборки проекта**. Maven, Gradle, Ant, и прочие, или рукописные сценарии на BASH (*nix) или Powershell (win)
- **Сторонние программы**. Для проверки кода на соответствие стилям и соглашениям, сборки ресурсов, создания документации, и прочего.

Схема работы конвейера

Система CI по некоторому событию (по расписанию, по сообщению, или по ручному запуску) запускает конвейер который последовательно:

1. Загружает свежую копию кода из репозитория
2. (опционально). Запускает системы статической проверки кода.
3. (опционально). Выполняет сценарии подготовки входных данных. Иногда такой шаг необходим для отдельных тестовых случаев.
4. Выполняет сборку проекта.
5. Выполняет автоматизированные модульные тесты.
6. Выполняет автоматизированные интеграционные тесты.
7. (опционально) Выполняет прочие проверки работоспособности системы в целом.
8. (опционально) Выполняет сценарии создания документации к системе.
9. Собирает все ресурсы проекта в один каталог/архив. Присваивает номер версии сборки. Результат работы конвейера: артефакт который в дальнейшем можно использовать для поставки в тестовую среду для проведения приемочных, интерфейсных, и прочих тестов.
10. Уведомляет разработчиков каким либо способом (почта, чатбот, мессенджер) о успешности или не успешности сборки.
11. Результат шага 9 теперь может использоваться для поставки на staging сервер, или в случае отсутствия его необходимости, сразу на сервер заказчика, или в production.

По результатам работы конвейера создается отчет, который попадает разработчикам, в котором видно была ли сборка успешной или нет, и если нет, то какие именно шаги не были успешными и почему.

Вообще, процесс непрерывной сборки помогает вам чаще выпускать более надёжный продукт, быстрее находить ошибки в вашем проекте. Это достигается при условии *надлежащего покрытия вашего кода модульными и интеграционными тестами, а так же использования систем контроля качества кода и статического анализа*.

Так как проект, обычно, собирается в контролируемой среде (на отдельном сервере), то только результаты сборки этого проекта считаются достоверно рабочими. И только эти результаты сборки должны будут использоваться для дальнейшего тестирования, и финальной поставки на сервер заказчика.

Рекомендации при создании процесса непрерывной сборки.

1. **Быстрая обратная связь**. Ваш процесс должен прекратить своё выполнение как только будут обнаружены какие либо проблемы со сборкой или выполнением тестов. Сообщение об ошибках должно быть отправлено разработчикам любым удобным способом. Это позволит исправлять ошибки в максимально короткий срок.
2. **В первую очередь выполняйте быстрые и важные тесты**. Если они успешно выполняются, переходите к следующим более медленным тестам.

3. **Разрабатывайте ваш конвейер так, что бы сборка и тестирование выполнялись параллельно**. Это позволит вам получить более быстрые циклы разработки, тестирования, и обратной связи.
4. **Сценарии сборки и тестирования должны быть независимы от платформы выполнения**. В случае смены системы CI/CD вы сможете запустить конвейер практически сразу.

Практика: GoCD

Сайт проекта: https://www.gocd.org/

Проект GoCD является инструментом для внедрения CI/CD (непрерывной сборки и поставки) в ваш процесс создания вашего ПО.

Проект состоит из двух модулей:

- Сервер. Управляющий модуль, который определяет задачи которые будут выполняться агентами.
- Агент. Модуль, обычно, устанавливающийся на отдельный компьютер, который выполняет шаги конвейера. Такое деление позволяет достичь более высокой скорости выполнения конвейера, за счет параллельного выполнения шагов конвейера и при условии использования нескольких агентов.

Установка и настройка

Для установки системы зайдите на https://www.gocd.org/download/ и скачайте версию для вашей операционной системы. На момент написания этой документации версия системы GoCD была 20.2.0.11344.

```
Следующие шаги описывают установку для системы
Ubuntu/Debian.
```

Для более легкого обновления модулей рекомендуется добавить GoCD репозиторий в систему на каждый компьютер, где установлен сервер и/или агент. Но в этом руководстве для более простой демонстрации мы скачиваем deb пакеты и устанавливаем их.

После окончания загрузки, в каталоге где находятся deb-пакеты, вам, необходимо открыть терминал и выполнить следующие команды:

```
# для установки сервера
sudo dpkg -i
./go-server_20.2.0-11344_all.deb # для
установки агента
sudo dpkg -i ./go-agent_20.2.0-11344_all.deb
# на случай если установка завершится с ошибкой
```

```
sudo apt-get -f install
```

GoCD хранит свои основные настройки в файле по следующему пути.

```
/usr/share/go-server/wrapper-config/
                                    wrapper-properties.conf
```

Для наших целей стандартных настроек по умолчанию достаточно, и на данный момент мы их менять не будем.

Запуск

Команда для запуска сервера:

```
sudo service start go-server
```

Команда для запуска агента:

```
sudo service start go-agent
```

Если агент и сервер у вас установлены на одном компьютере, то вам необходимо выполнить обе команды.

Панель управления

После запуска сервера и агента, можно открыть браузер и перейти в панель управления, где мы создадим первый конвейер.

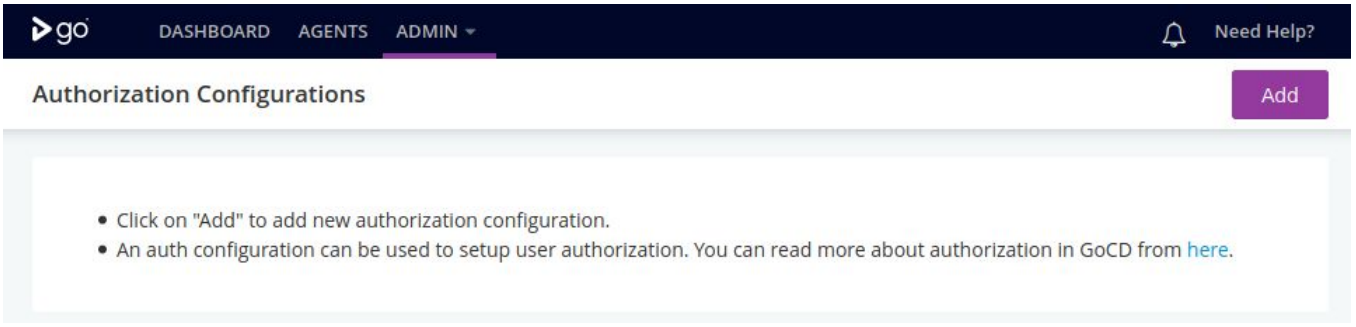
http://localhost:8153/go

Настройка пользователей

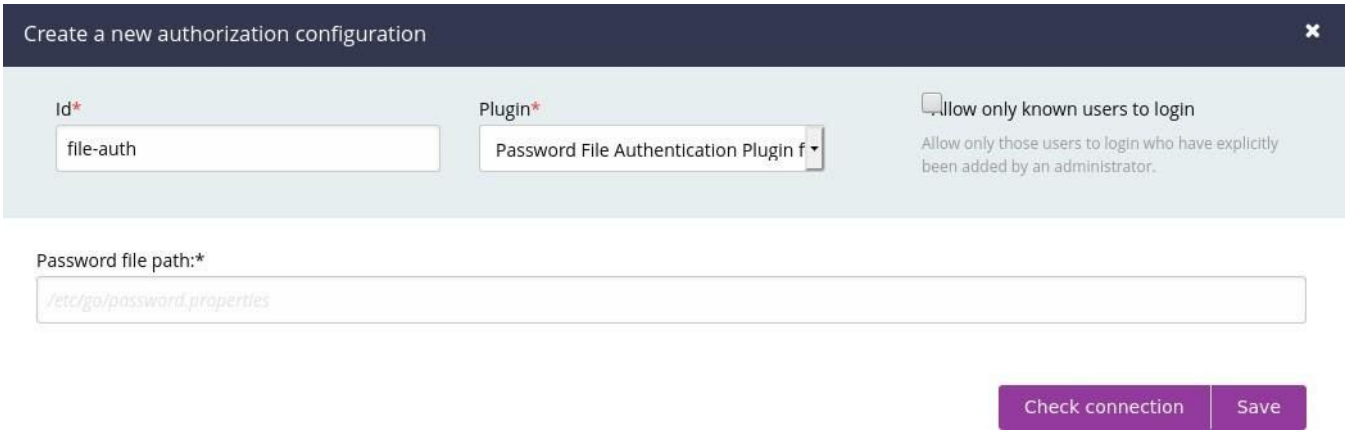
По умолчанию GoCD работает с выключенной авторизацией пользователей. Это позволяет нам протестировать определенные функции системы, но не все. Некоторые модули GoCD требуют авторизованного пользователя. В комплекте с системой поставляется два таких модуля: авторизация через файл паролей, и авторизация через LDAP.

Для более быстрой настройки воспользуемся авторизацией через файл паролей. Перейдём в блок настроек:

http://localhost:8153/go/admin/security/auth_configs



И добавим авторизацию через файл паролей



Укажите путь к вашему файлу с пользователями и паролями.

Создание файла пользователей/паролей

Полные инструкции по созданию файла пользователей/паролей:

https://github.com/gocd/gocd-filebased-authentication-plugin#usage-instructions

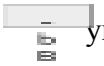
Откройте терминал в том каталоге, где вы хотите разместить файл с пользователями.

Для создания паролей нам понадобится дополнительный инструмент htpasswd из пакета apache2-utils. Установить его можно командой:

```
sudo apt-get install apache2-utils
```

Для создания пароля нам надо воспользоваться командой:

```
htpasswd -bB имя-файла имя-пользователя    пароль
```

ключ  указывает, что мы используем алгоритм bcrypt, и что пароль будет передаваться через командную строку. Передача пароля в открытом виде (через командную строку), является не очень безопасным методом, но сейчас для наших целей этого будет достаточно. Записи о пользователях будут дописываться в указанный файл.

```
# создадим файл
touch
password.properties #
добавим пользователей
htpasswd -bB password.properties admin admin
htpasswd -bB password.properties user user
```

В результате мы получим что-то вроде следующих строк. Это цифровой отпечаток пароля в специальном формате.

```
admin:$2y$05$micA0Q3dDkHUubn/EmzE0O1uvJkLZ40ipfBNPZruNXuDM7nh19CGq
user:$2y$05$.xVXJQ/0GPSQGvdiEoWZUeaerOuM0rKwXoNNgTibq1Qkk85Tdtve2
```

После перезагрузки страницы в браузере и входа в систему мы можем настроить модуль уведомлений.

Обратная связь. Модуль почтовых уведомлений.

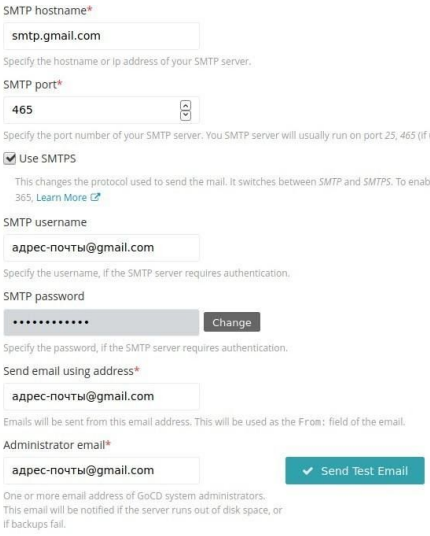
Важно, что бы в вашем конвейере были циклы обратной связи. Они позволят вам быстрее находить и устранять проблемы, которые могут появиться при внесении изменений в ваш продукт.

Обычно для этого используют модули уведомлений, которые отправляют сообщения в удобный для разработчиков канал связи. Стандартным механизмом является отправка письма на определенный адрес при условии если сборка была успешной или не успешной. Этим занимается специальный модуль уведомлений, который встроен в GoCD.

Настройка для Gmail

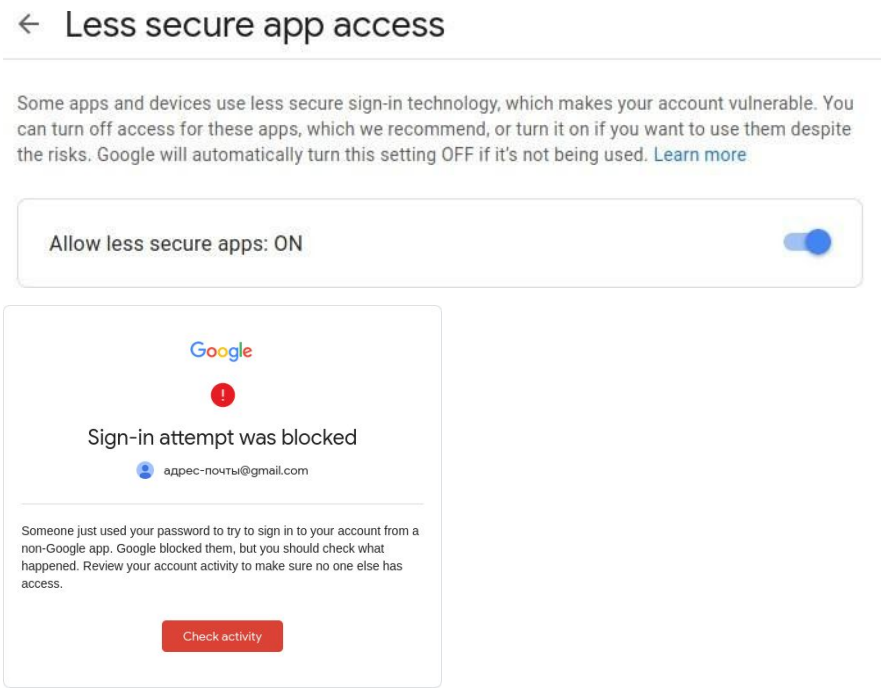
Внимание! Настройка для других почтовых провайдеров может отличаться.

Для того, что бы модуль уведомлений смог отправить почту по какому либо событию конвейера (успешному или не успешному завершению сборки), необходимо настроить его. Для этого необходимо зайти вот сюда: http://localhost:8153/go/admin/config/server#! email-server



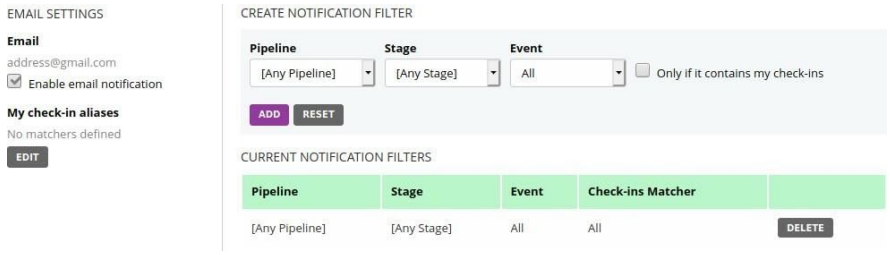
Указать параметры вашего почтового сервера, через который вы будете отправлять почту. Если вы будете использовать Gmail, то его настройки можно посмотреть вот тут https://support.google.com/mail/answer/7126229

Кроме обычной настройки, вам необходимо включить в настройках аккаунта доступ для "менее безопасных приложений". И разрешить доступ к вашему аккаунту при первой попытке отправки почты.



После внесения изменений нажмите "Send Test Email", убедитесь, что вы получили тестовое письмо, и сохраните настройки.

Следующим этапом вам необходимо настроить рассылку для каждого пользователя который должен получать уведомления.



Ссылка на профиль настроек: http://localhost:8153/go/preferences/notifications

Добавьте ваш адрес почты в настройки пользователя, и укажите имя, которым вы подписываетесь в коммитах в репозиторий, либо снимите галочку, иначе не будете получать уведомления, и добавьте фильтр на все конвейеры. В дальнейшем, если у вас будет более одного конвейера, вы сможете поменять эти настройки. Но на сейчас этого будет достаточно.

Вы можете сделать настройку фильтров таким образом, что вы будете получать уведомления только о тех сборках которые поломались с момента последней успешной сборки. Для этого в событии (event), выберите "Breaks".

Подробнее о уведомлениях: https://docs.gocd.org/current/configuration/dev_notifications.html

Что такое конвейер.

Основной идеей в системе GoCD является **конвейер**. Это логическое понятие, означающее процесс преобразования некоторого исходного "материала" в некоторый готовый "продукт".

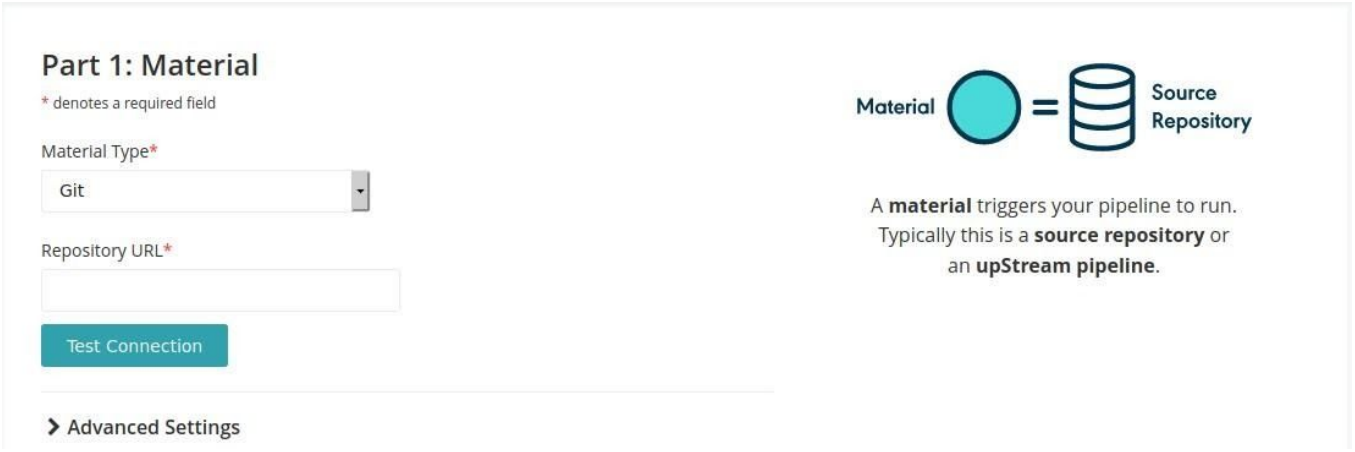
Материалы поступают на вход конвейера, который состоит из этапов обработки, которые состоят из заданий, которые состоят из шагов. И если провести аналогию с заводом, то:

- **материалы** (material) - это детали поступающие на вход конвейера
- **конвейер** (pipeline) - это линия сборки в целом, которая состоит из сборочных станций
- **этап** (stage) - это сборочная станция которая выполняет одну или несколько логически связных задач. Например: "обработать делать на шлифовальном станке", "проверить допуски" и так далее
- **задача** (job) - один или несколько шагов необходимых для выполнения некоторого действия.
- **шаг** (task) - это атомарная единица работы. Например: "взять деталь", "поставить деталь в нужное место", "использовать точечную сварку", "проверить места сварки".
- **артефакт** (artifact) - это результат выполнения задачи. Это могут быть метрики, или готовая деталь.

Первый конвейер CI/CD

Все конвейеры в GoCD начинаются с подключения "материала". **Материалом** обычно является репозиторий с исходным кодом, но может быть и результат работы другого конвейера.

На первом шаге создания конвейера, укажем откуда будут поступать наши материалы.

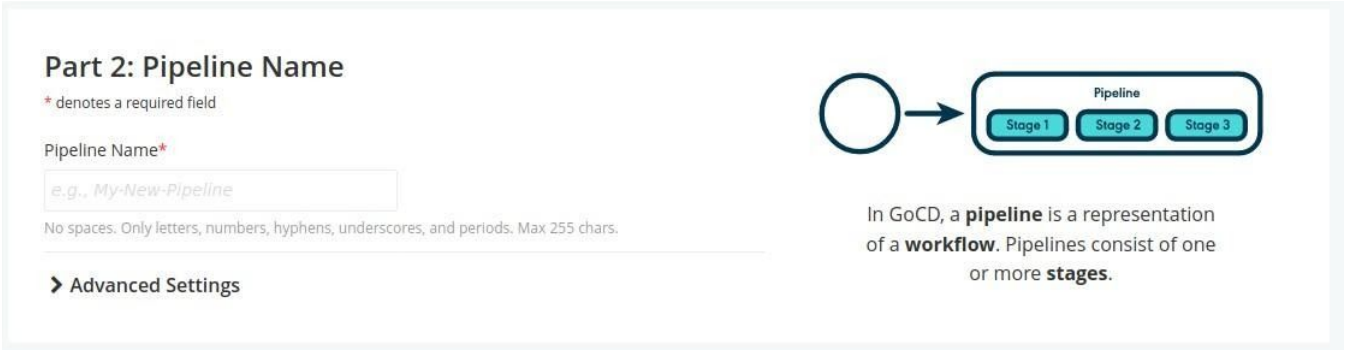


Подключим наш репозиторий, выбрав в типе материала Git, и указав путь к нашему репозиторию в Repository URL.

Открыв дополнительные настройки, мы можем указать какую ветку мы хотим использовать в поле Repository Branch, и при необходимости имя пользователя и пароль для доступа к репозиторию в соответствующие поля.

По умолчанию *материалы* проверяются на наличие изменений 1 раз в минуту. Эту настройку можно изменить после создания конвейера.

На втором шаге укажите имя конвейера. Обычно тут указывается имя проекта и дополнительные данные, полезные для идентификации конвейера.



На третьем шаге создания конвейера мы указываем имя этапа. Например "выполнить-сборку", или "выполнить-модульные-тесты".

Типичный конвейер состоит из минимум одной стадии или этапа. **Этап (stage)** - это набор заданий, который необходимо выполнить. Если ваш конвейер будет состоять более чем из одного этапа, вы сможете добавить их позже. Задачи должны быть **независимыми** друг от друга, так как они могут выполняться на разных агентах.

На четвертом шаге создания конвейера вы указываем шаги, которые необходимо выполнить для первого задания. Например `mvn clean test`

Типичный этап конвейера состоит из минимум одного задания (job). **Задание (job)** - это набор шагов, каждый из которых, в идеале, выполняет только одно действие. Команды внутри одного задания выполняются последовательно.

Инструменты конвейера.

Так как реальная работа выполняется на стороне "агента", а он может быть установлен на другом компьютере, то необходимо, что бы все инструменты которые используются в конвейере, были доступны на том компьютере.

Стандартный набор путей на операционной системе ubuntu|debian в которых могут размещаться инструменты которые требуются вашему конвейеру.

```
/usr/local/sbin
/usr/local/bin
/usr/sbin
/usr/bin
/sbin
/bin
/snap/bin
```

Если вы используете нестандартные инструменты, то вы можете разместить их или в любой удобный для вас каталог и указать путь к нему в настройках агента, или используя команду `update-alternatives`, создать ссылку на каждый из ваших инструментов и поместить её в один из стандартных каталогов (обычно для этого используется каталог `/usr/bin`).

Если какой-то инструмент не будет доступен, ваш конвейер остановится, и вы сможете посмотреть ошибку в результатах работы.

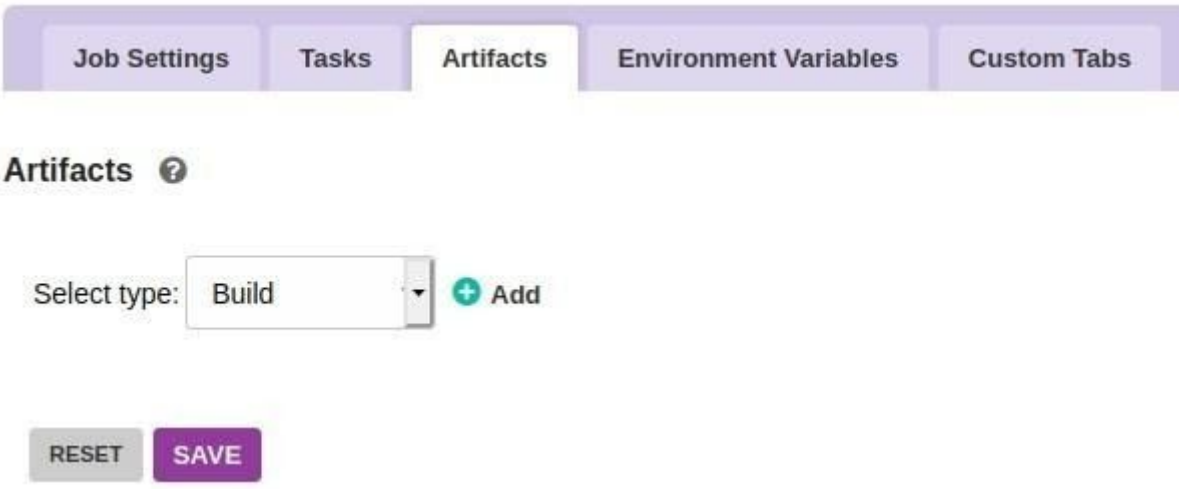
Артефакт - результат работы задач конвейера

Артефакт в GoCD это файл или каталог, который производится в процессе выполнения задач конвейера. Обычно это то, что мы хотим получить на выходе процесса, или результаты тестов. Каждая из задач конвейера может быть настроена на публикацию одного или нескольких артефактов.

Примерами артефактов являются: отчеты выполнения тестов, отчеты покрытия кода тестами, исполняемые файлы, установщики/дистрибутивы, документация, образы docker контейнеров, архивы, и прочие файлы, которые требуется сохранить для других конвейеров или прочих бизнес процессов.

То есть, артефактом можно назвать некоторый "физический" результат успешного завершения одного задания одного этапа конвейера.

Так как артефакты создаются на стороне go-агентов, то мы должны явно указать, что именно мы должны скопировать на сервер по завершению задания.

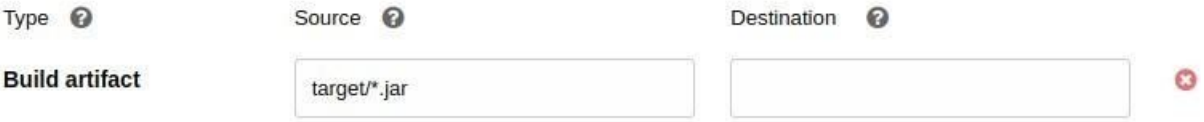


Артефактов существует три типа:

- **Test** - это какой либо отчет создаваемый системой тестирования. Например для JUnit таким артефактом является XML-файл который содержит в себе результаты прогона всех тестов. Для maven проектов результаты JUnit тестов находятся в каталоге target/surefire-reports/.

  И мы можем скопировать его добавив шаг копирования тестовых артефактов.

- **Build** - это некоторый результат работы, который не является отчетом тестирования, который важен для нас, или какого-то бизнес процесса. Например, после сборки maven проекта , это JAR-файл который появляется в каталоге target.



- **External** - это артефакт, который загружается из внешнего хранилища. Обычно используется для передачи артефактов между go-серверами. В нашем случае этот тип артефактов не используется, так как требует настройки дополнения для подключения внешнего хранилища артефактов.

Дополнительные модули.

Систему GoCD можно расширить через использование дополнительных модулей. Например мы можем подключить модуль уведомлений в чат-систему Slack, и получать уведомления о сборке напрямую в этот чат. Или разработать дополнение для отправки уведомлений в чат телеграма.

Список доступных дополнений можно посмотреть по этой ссылке: https://www.gocd.org/plugins/

Дополнения устанавливаются в специальный каталог, и становятся доступными после перезагрузки go-сервера.

Каталог для дополнений в системе ubuntu: `/var/lib/go-server/plugins/external/`

Дополнительные материалы

- https://www.gocd.org/getting-started/
- http://blog.agilix.ru/2019/09/27/непрерывная-интеграция/
- https://ru.wikipedia.org/wiki/DevOps
- https://youtu.be/HnWuIjUw_Q8
- https://youtu.be/a9r-lXLDLvk
- https://www.martinfowler.com/articles/continuousIntegration.html