

1. Gitea.

Для того, чтобы получить доступ к Gitea в поисковый адрес нужно ввести `http://localhost:3000/`. Так как я работаю из-под виртуальной машины, нужно использовать IP адрес машины. В моем случае это `192.168.0.182:3000`. После перехода так будет выглядеть успешный запуск Gitea.

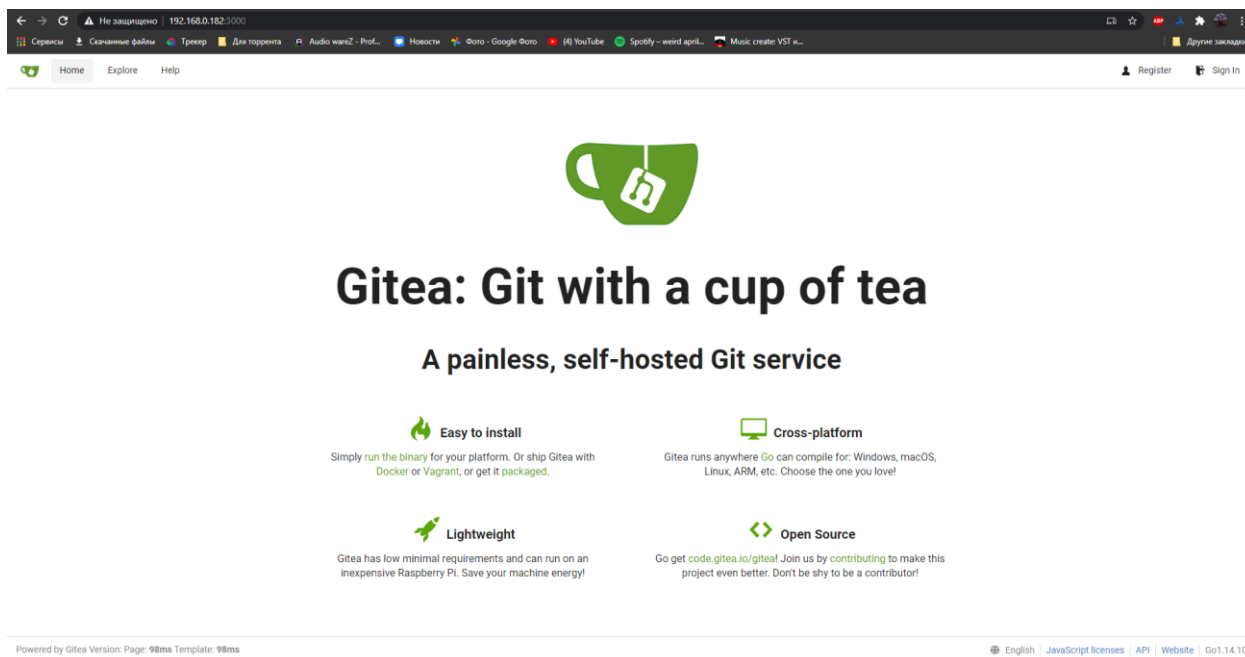


Рисунок 1. Успешный первый вход на страницу Gitea.

Далее нужно нажать на регистрацию, но вначале установить базу данных.


The screenshot displays the 'Initial Configuration' page of Gitea. It is divided into two main sections: 'Database Settings' and 'General Settings'. In the 'Database Settings' section, the 'Database Type' is set to 'SQLite3' and the 'Path' is '/data/gitea/gitea.db'. The 'General Settings' section includes fields for 'Site Title' (Gitea: Git with a cup of tea), 'Repository Root Path' (/data/git/repositories), 'Git LFS Root Path' (/data/git/lfs), 'Run As Username' (git), 'SSH Server Domain' (localhost), and 'SSH Server Port' (22). Each field has a description and instructions on how to use it.

Рисунок 2. Установка базы данных.

Здесь нужно лишь изменить все названия, содержащие `localhost` на адрес вашего хоста и установка пройдет успешно. Далее нужно зарегистрироваться и создать репозиторий.

New Repository

Owner *



alex

▼

Some organizations may not show up in the dropdown due to a maximum repository count limit

Repository Name *

test

Good repository names use short, memorable and unique keywords.

Visibility

☐ Make Repository Private

Only the owner or the organization members if they have rights, will be able to see it.

Description

Template

Select a template.

Issue Labels

Select an issue label set.

.gitignore

Select .gitignore templates.

License

Select a license file.

README

Default

☐ Initialize Repository (Adds .gitignore, License and README)

Default Branch

master

Рисунок 3. Создание репозитория.

Все, что от нас требуется – ввести название.

После успешного создания репозитория, туда нужно загрузить что-то, чтобы мы могли работать с другими сервисами.

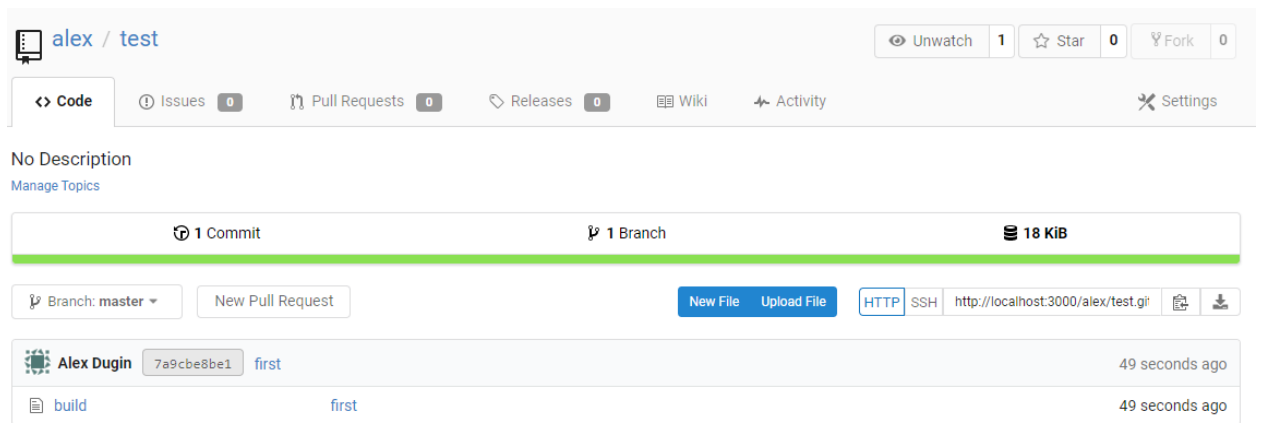


Рисунок 4. Залитое на Gitea приложение.

Для проверки работы я добавил обычное bash приложение, которое выводит некоторые строки на консоль. Далее переходим к следующему сервису.

2. GoCD.

Для того, чтобы попасть на сайт сервиса, нужно ввести в адресной строке браузера <http://192.168.0.182:8153/> или же <http://localhost:8153/>, если вы не работаете из-под виртуальной машины. Если сайт не открывается, то проверьте сервис gocd в консоли на ошибку Permission Denied. Это значит, что папка gocd/data доступна только для чтения. Измените права папки и перезапустите docker-compose.



Рисунок 5. Удачный экран загрузки.

На моей машине GoCD загружается немного дольше остальных сервисов. Пожалуйста, будьте терпеливы и ждите пока не появится кнопка “GoCD Server Loaded - Click here”

Нас перекинет на страницу первоначальной настройки.

Здесь следует обратить внимание на такие вещи как:

- Repository URL – URL нашего репозитория, как бы логично это не звучало. В эту строку нужно вставить адрес нашего репозитория. Её можно взять просто из адресной строки, находясь в репозитории. В моем случае это <http://192.168.0.182:3000/alex/test>

Part 1: Material

* denotes a required field

Material Type*

Git


Repository URL*

http://192.168.0.182:3000/alex/test

Test Connection

Connection OK

Advanced Settings



A **material** triggers your pipeline to run. Typically this is a **source repository** or an **upStream pipeline**.

Part 2: Pipeline Name


* denotes a required field

Pipeline Name*

def

No spaces. Only letters, numbers, hyphens, underscores, and periods. Max 255 chars.

Advanced Settings



In GoCD, a **pipeline** is a representation of a **workflow**. Pipelines consist of one or more **stages**.

Рисунок 6. Первоначальная настройка.

В Part 2 Pipeline Name можно задать любое имя для pipeline. Также и в Part 3 Stage Name. Также и в Part 4 Job Name. Далее, где показан текст в виде терминала нужно ввести команду для выполнения нашей задачи. Так как я использую bash-скрипт в приложении и называется он build, то для запуска скрипта нужно ввести лишь ./build. Так и вводим.

Part 3: Stage Details

* denotes a required field

Stage Name*

No spaces. Only letters, numbers, hyphens, underscores, and periods. Max 255 chars.

➤ Advanced Settings

Part 4: Job and Tasks

* denotes a required field

Job Name*

No spaces. Only letters, numbers, hyphens, underscores, and periods. Max 255 chars.

Type your tasks below at the prompt*

+ Caveats

+ Help

Press <enter> to save, <shift-enter> for newline

\$./build

\$

➤ Advanced Settings

Рисунок 7. описание для запуска скрипта.

Нажимаем ниже на кнопку “Save and Run This Pipeline” и переходим на наш Dashboard.

Далее нужно удостовериться, что агент смог подключиться к серверу. Переходим по верхней кнопке в секцию “Agents”. Подключение также занимает некоторое время. Когда мы увидим, что агент подключился, нам нужно активировать его.

STATIC ELASTIC						
DELETE	ENABLE	DISABLE	ENVIRONMENTS ▾	RESOURCES ▾	Total : 1 Pending : 1 Enabled : 0 Disabled : 0	
✓	AGENT NAME ▾	SANDBOX ▾	OS ▾	IP ADDRESS ▾	STATUS ▾	FREE SPACE ▾
✓	dd8ef07d6af6	/go	debian 9.13	172.24.0.8	Pending	10.17 GB

Рисунок 8. Успешное подключение агента.

Для его активации, нужно нажать на чекбокс слева от него и нажать на кнопку "Enable". Все, агент включен.

Далее перейдем обратно, на страницу Dashboard. Наведясь на прямоугольник состояния Pipeline, можем увидеть, что проект успешно собран.

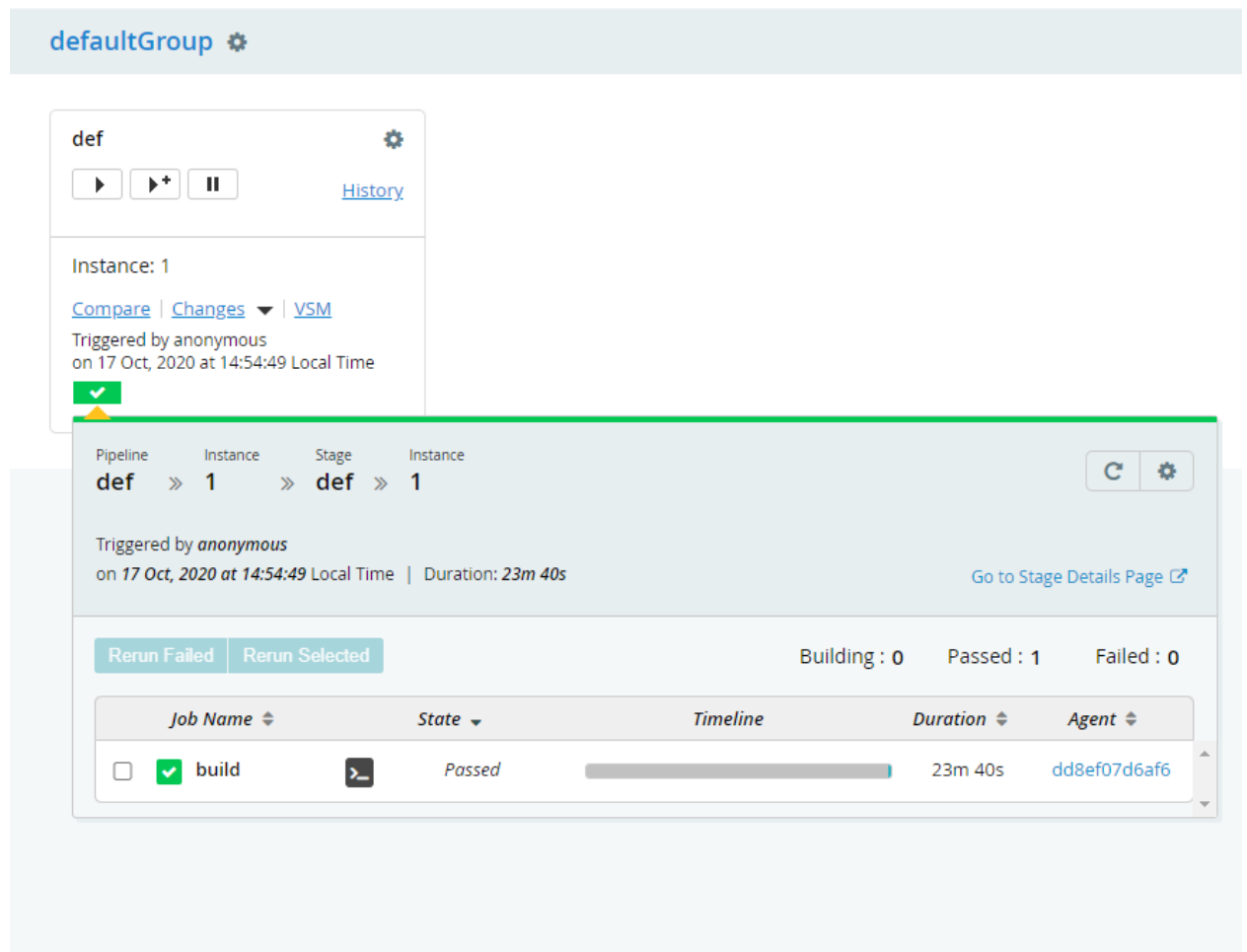


Рисунок 9. Успешная сборка проекта.

Нажав на шестеренку, мы перейдем в настройки. В настройках во вкладке GENERAL есть настройка триггеров на сборку проекта. По умолчанию стоит автоматическая сборка, то есть триггер срабатывает, когда в Gitea появляется новый коммит. Можно отключить эту функцию сняв флажок с "Automatic pipeline scheduling". Ниже находится настройка триггера по времени. В ней используется синтаксис Cron. По желанию, это можно настроить, но я рекомендую автоматическую сборку.

Также, нелишним будет демонстрация сборки с ошибкой. Внесем изменения в bash-скрипт таким образом, чтобы при вызове его он выводил ошибку.

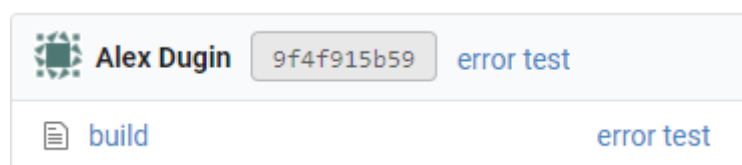


Рисунок 10. Коммит с заведомо ошибочным скриптом.

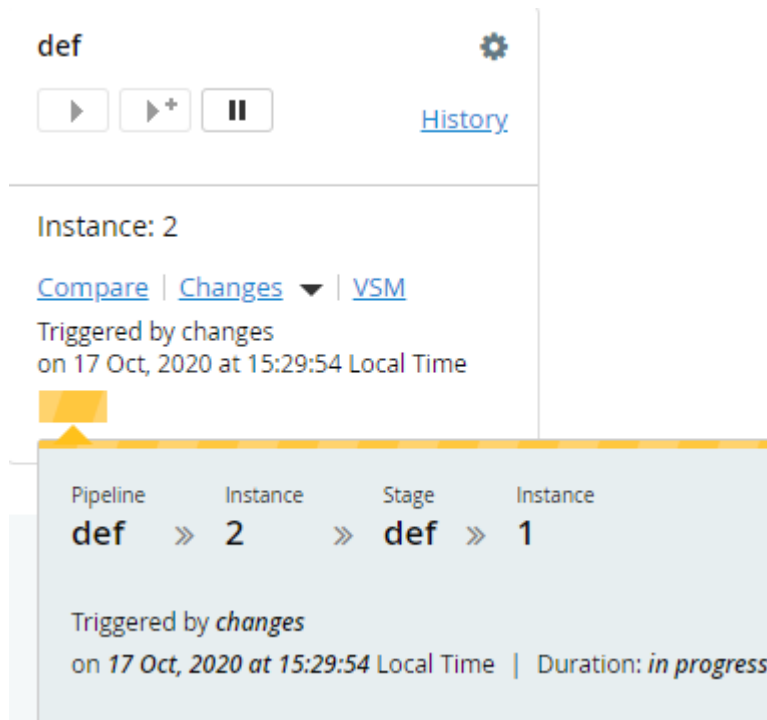


Рисунок 11. Скрипт сработал из-за изменений (Triggered by changes)

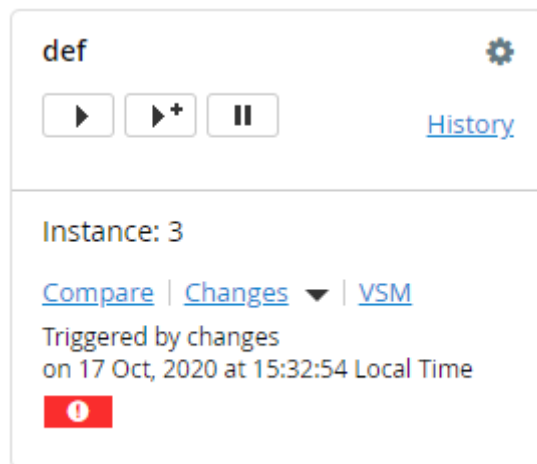


Рисунок 12. Пример теста заведомо ошибочного скрипта.

Как видно, тест сработал верно. Нажав на красный прямоугольник и на кнопку в виде `cmd` строки, можно увидеть вывод в консоли.

Job Details

Pipeline **def** » Instance **3 VSM** » Stage **def / 1** » Job **Build** ⚙

SCHEDULED ON: 17 Oct 2020 at 15:32:54 Local Time

COMPLETED ON: 17 Oct 2020 at 15:33:15 Local Time [more...](#)

DURATION: 0.0s

Console

Tests

Failures

Artifacts

Materials

▶ [go] Job Started: 2020-10-17 12:33:13 UTC

▶ [go] Start to prepare def/3/def/1/build on dd8ef07d6af6 [/go]

[go] Start to build def/3/def/1/build on dd8ef07d6af6 [/go]

❶ ▶ [go] Task: **./build** took: 0.16s exited: 2

./build: line 15: unexpected EOF while looking for matching `''

./build: line 16: syntax error: unexpected end of file

[go] Task status: failed, took: 0.16s, exited: 2

❶ [go] Current job status: failed

[go] Start to create properties def/3/def/1/build on dd8ef07d6af6 [/go]

[go] Start to upload def/3/def/1/build on dd8ef07d6af6 [/go]

[go] Job completed def/3/def/1/build on dd8ef07d6af6 [/go]

Рисунок 13. Вывод консоли

На этом интеграция GoCD и Gitea закончена.

3. Taiga.

Чтобы перейти на сайт сервиса нужно ввести в адресную строку <http://192.168.0.182:80> или же <http://localhost:80>, если вы не работаете из-под виртуальной машины. Нас встретит страница входа. Для входа нужно ввести логин – admin и пароль – 123123.

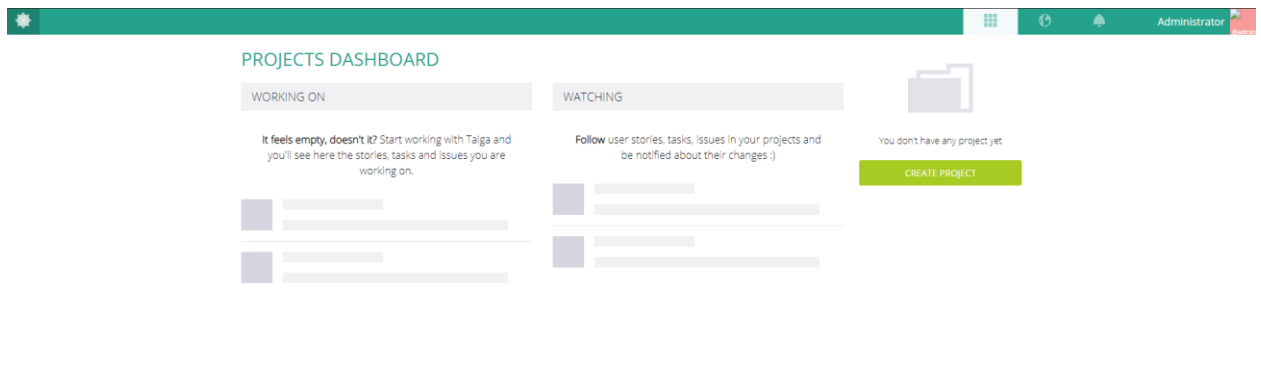


Рисунок 14. Страница с успешным входом

Далее нужно создать проект, нажав на кнопку “Create project”. Далее выбрать “Kanban”, далее дать название и описание и нажать “Create project”. Может показаться, что страница зависла, но, видимо, у нее просто нет редиректа на панель Kanban. Просто нажмите вверх “Dashboard”.

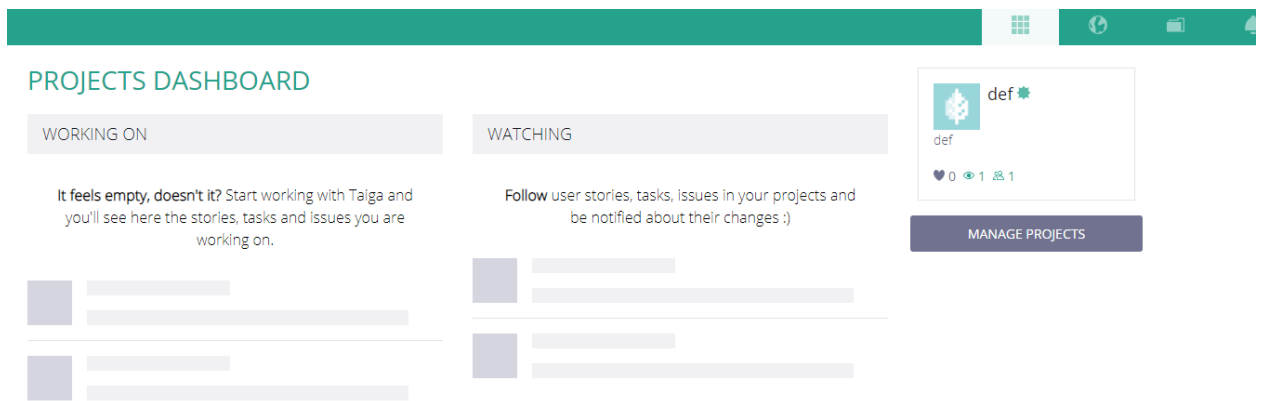


Рисунок 15. попадаем сюда после создания.

Выберем наш проект и слева в панели выберем Kanban. Чтобы связать этот сервис с Gitea, первое что нужно сделать - создать задачу. Для этого на одной из колонок требуется нажать на +, ввести название и выбрать “Assigned to me” и нажать “Create”. Чтобы задача была видимой, нужно обновить страницу.

KANBAN DEF

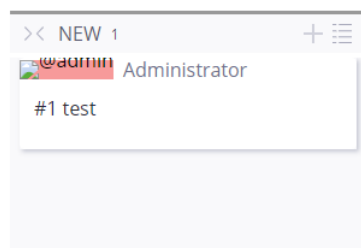


Рисунок 16. добавлена задача.

Теперь в панели слева нужно нажать на “Admin” -> “Integrations” и выбрать “GOGS”, так как Taiga не имеет официальной связи с Gitea, но они взаимозаменяемые, поэтому интеграция проходит успешно.

Далее нужно зайти на созданный gitea-репозиторий, “Settings” -> “Webhooks” -> “Add Webhooks” -> “Gitea”. Теперь нужно скопировать Secret Key и Payload URL из Taiga и вставить их в соответствующие графы в Gitea. Замечание: в Payload URL нужно заменить “taiga.lan” на адрес вашего хоста.

The screenshot shows the 'Update Webhook' configuration page in Gitea. At the top, there's a notification: 'A fake event has been added to the delivery queue. It may take few seconds before it shows up in the delivery history.' The page title is 'Update Webhook'. Below it, a note states: 'Gitea will send post requests with a specified content type to the target URL. Read more in the [webhooks guide](#).' The configuration fields are as follows:

- Target URL ***: `http://192.168.0.182/api/v1/gogs-hook?project=1`
- HTTP Method**: `POST`
- POST Content Type**: `application/json`
- Secret**: A field with a masked secret key (dots).
- Trigger On:** Radio buttons for `Push Events`, `All Events` (selected), and `Custom Events...`
- Branch filter**: A text field containing an asterisk `*`.

Below the 'Branch filter' field, there is a note: 'Branch whitelist for push, branch creation and branch deletion events, specified as glob pattern. If empty or *, events for all branches are reported. See [github.com/gobwas/glob](#) documentation for syntax. Examples: master, {master, release*}.'

Рисунок 17. Правильная конфигурация Webhook

Теперь нужно проверить работу интеграции. Для этого нужно сделать коммит в Gitea с содержанием специальных ключей для Taiga. Это должно выглядеть так – текст_коммита TG-номер_задачи #стадия_разработки

Например, Тестируем файл TG-1 #ready-for-test.

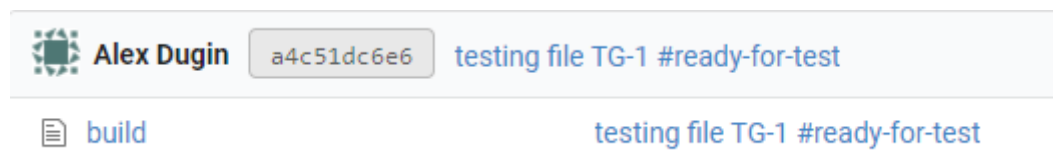


Рисунок 18. Коммит на Gitea

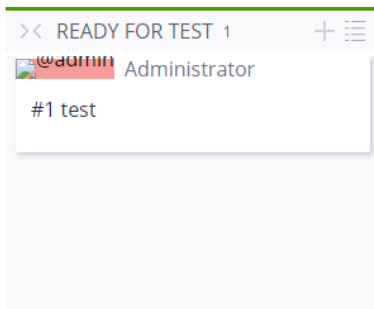


Рисунок 19. Задача успешно поменяла статус.

На этом интеграция Gitea и Taiga да и, в принципе, систем окончена.