BeeCR

Руководство администратора

Команда BeeCR

Содержание

1. Развертывание в вашей инфраструктуре

1.1 В Docker контейнерах

1.1.1 Обзор конфигурации Docker Compose для развертывания BeeCR

Данный раздел документации посвящен особенностям конфигурационного файла для запуска BeeCR на собственном сервере с помощью Docker.

Примечание: Если вы ищете пошаговые инструкции по развертыванию на собственных серверах с помощью Docker, пожалуйста, обратитесь к данному руководству.

Требования

Для развертывания решения на сервере требуется:

1. Сервер на основе Linux (желательно Ubuntu или Debian).
2. Графический ускоритель NVidia (минимум 24 ГБ памяти, рекомендуется 40 ГБ).
3. Драйвер NVidia.
   Вы можете убедиться, что драйвер присутствует и работает, выполнив команду типа `nvidia-smi` (команда должна перечислить доступные GPU).
4. Docker, а также Docker Compose.
5. NVIDIA Container Toolkit

Краткая инструкция

1. Скачайте файл on-premises/compose.yml в любое место на ваш выбор.
2. Поместите ваш файл лицензии (предоставляется нашей командой) в ту же папку, где находится `compose.yml`.
3. Запустите следующую команду в каталоге с `compose.yml`, чтобы запустить контейнеры AI и API в режиме демона:

```
docker-compose up -d
```

Расширенная настройка

Рекомендуется изучить документацию "Getting started" по Docker Compose, если вы не знакомы с этим инструментом.

ВНЕШНИЙ ПОРТ API

По умолчанию предоставленная конфигурация Docker Compose устанавливает порт `8000` для API:

```
...
ports:
  - "8000:80"
...
```

Замените `8000` на предпочтительный для вас порт.

ФАЙЛ ЛИЦЕНЗИИ

В предоставленной конфигурации Docker Compose предполагается, что файл лицензии размещается в той же папке, где находится и сам `compose.yml` :

```
...
volumes:
  - "./beecr.lic:/app/beecr.lic:ro"
...
```

Если вы хотите сохранить его в другом месте, просто замените `./beecr.lic` на соответствующий путь в вашей файловой системе или в Docker-томе.

Актуальные разделы документации к Docker: Volumes, Bind Mounts.

АДРЕС GITLAB

По умолчанию предоставленная конфигурация Docker Compose устанавливает адрес сервера GitLab как `https://gitlab.com` с помощью переменной окружения `GITLAB_HOST` :

```
- "GITLAB_HOST=https://gitlab.com"
```

Если вы используете экземпляр GitLab, развернутый в вашей собственной инфраструктуре, то рекомендуется переопределить, задав актуальный адрес GitLab. Однако это не всегда обязательно, так опция применяется только в том случае, если актуальный адрес GitLab не был передан в параметрах HTTP-запроса к API серверу. Если вы собираетесь использовать предоставленные компоненты/шаблоны CI/CD, то нет необходимости переопределять `GITLAB_HOST` , потому что компоненты/шаблоны CI/CD всегда передают адрес GitLab в параметрах запроса.

ТОКЕН ДОСТУПА К GITLAB

В большинстве случаев API сервер BeeCR будет получать токен доступа GitLab через параметры запроса HTTP в каждом запросе на проверку. Однако в некоторых случаях вы можете захотеть установить значение токена по умолчанию на стороне сервера. Для этого выпустите токен доступа к GitLab проекта или группы и установите переменную среды `GITLAB_TOKEN` :

```
- "GITLAB_TOKEN=токен-доступа-gitlab"
```

Если вы собираетесь использовать предоставленные компоненты/шаблоны CI/CD, то нет необходимости переопределять `GITLAB_TOKEN`, потому что компоненты/шаблоны CI/CD всегда передают токен доступа к GitLab в заголовках HTTP-запроса.

OLLAMA

Предоставленная конфигурация Docker Compose предполагает, что доступ к ИИ модели выполняется через ИИ сервер с Ollama-совместимым API, который запущен на смежном Docker-контейнером:

```
- OLLAMA_HOST=http://beecr-ai:11435
```

Обновите переменную среды `OLLAMA_HOST`, если вы хотите развернуть ИИ сервер отдельно.

Если же вы собираетесь использовать модели через OpenAI-совместимый API (например модель `gpt-4o`), то установка опции `OLLAMA_HOST` не имеет смысла.

OPENAI

Если вы собираетесь использовать модели ИИ через OpenAI-совместимый API, то, возможно, вам может потребоваться задать `OPENAI_HOST` и/или `OPENAI_API_KEY`:

- `OPENAI_HOST`: Адрес OpenAI-совместимого API сервера. Измените его, если вы используете непосредственно OpenAI,но через прокси-сервер или же используете какой-то сторонний API, совместимый с OpenAI.
- `OPENAI_API_KEY`: Ваш ключ доступа к OpenAI API.

Примечание: Если вы собираетесь использовать модели ИИ через OpenAI-совместимый API, то, вероятно, вам не нужно запускать Docker контейнер с ИИ локально. В такому случае просто удалите (или закомментируйте) всю секцию `beecr-ai` в конфигурации Docker Compose.

МОДЕЛЬ

Предоставленная конфигурация Docker Compose задаёт ИИ модель через переменную среды `MODEL`.

```
- "MODEL=ollama/codestral:22b"
```

Эта опция применяется только в том случае, если входящие запросы HTTP API не указывают модель явно. Если вы собираетесь использовать предоставленные компоненты/шаблоны CI/CD, то нет необходимости переопределять `MODEL`, потому что компоненты/шаблоны CI/CD всегда передают модель через запросы.

Примечание:

- Для OpenAI-совместимых API используйте имена моделей "как есть" например, `gpt-4o`.
- Для Ollama-совместимых API используйте имена моделей с префиксом `ollama/`, например, `ollama/codestral:22b`.

1.1.2 Пошаговое руководство по развертыванию BeeCR в Docker

В данном руководстве вы познакомитесь с процессом развертывания BeeCR в вашей собственной инфраструктуре с помощью Docker.

Требования

Для развертывания решения на сервере требуется:

1. Сервер на основе Linux (желательно Ubuntu или Debian).
2. Графический ускоритель NVidia (минимум 24 ГБ памяти, рекомендуется 40 ГБ).
3. Драйвер NVidia.
   Вы можете убедиться, что драйвер присутствует и работает, выполнив команду типа `nvidia-smi` (команда должна перечислить доступные GPU).
4. Docker, а также Docker Compose.
5. NVIDIA Container Toolkit

1 Подготовьте конфигурацию Docker Compose

Скачайте конфигурационный файл Docker Compose в удобное место:

```
curl -sLO 'https://docs.beecr.ru/on-premises/compose.yml'
```

2 Подготовьте файл лицензии

Предполагается, что у вас уже есть лицензионный файл, выданный нашей командой. Пожалуйста, свяжитесь с нами, чтобы получить/купить лицензию для BeeCR, если у вас ее еще нет.

После получения файла лицензии поместите его в ту же директорию, где вы сохранили `compose.yml` на предыдущем шаге.

3 Загрузите Docker-образы *(опционально)*

Если целевой сервер имеет подключение к Интернету, то вы можете пропустить этот шаг, потому что Docker автоматически загрузит образы. В противном случае вам может потребоваться предварительно загрузить и сохранить образы на компьютере с подключением к Интернету, передать их на целевой сервер.

Скачайте последние доступные Docker-образы из Интернета и экспортируйте их в `tar` архивы:

```
docker pull "cr.yandex/crpnppdjigsa1876k1eq/beecr/ai"
docker pull "cr.yandex/crpnppdjigsa1876k1eq/beecr/api"
docker save "cr.yandex/crpnppdjigsa1876k1eq/beecr/ai" -o "beecr-ai.tar"
docker save "cr.yandex/crpnppdjigsa1876k1eq/beecr/api" -o "beecr-api.tar"
```

Передайте `tar` архивы образов на целевой сервер с помощью удобного метода, например, команды `scp`:

```
scp beecr*.tar USER@SERVER
```

На целевом сервере загрузите Docker-образы из `tar` архивов:

```
docker load -i "beecr-ai.tar"
docker load -i "beecr-api.tar"
```

4 Измените конфигурацию Docker Compose *(опционально)*

Установленные по умолчанию параметры в предоставленном `compose.yml` удовлетворяют потребности большинства пользователей. Однако вы можете захотеть настроить некоторые параметры.

Например, по умолчанию предоставленная конфигурация Docker Compose открывает API на порте `8000`. Если вы хотите использовать другой порт, внесите изменения в соответствующий раздел `compose.yml`.

Еще один пример параметра, который можете захотеть изменить, это путь к файлу лицензии. Как упоминалось выше, ожидаемое местоположение этого файла - это директория, содержащая `compose.yml`. Однако, если вы хотите сохранить его в другом месте, просто измените `./beecr.lic` на соответствующий путь в вашей файловой системе или в Docker-томе.

Ссылки:

- Если вам нужна дополнительная информация о предоставленной конфигурации Docker Compose, обратитесь к соответсующей документации.
- Если вы не знакомы с Docker Compose, рекомендуется прочитать документацию "Getting Started" к данному инструменту.
- Вам также может быть актуальный следующие разделы документации Docker: Volumes, Bind Mounts.

5 Запустите контейнеры

Выполните следующую команду в директории, содержащей `compose.yml`, чтобы запустить API и AI контейнеры в фоновом режиме:

```
docker-compose up -d
```

6 Проверьте логи

После запуска контейнеров рекомендуется проверить логи, чтобы убедиться, что все работает корректно.

Проверьте логи контейнера с API сервером BeeCR:

```
docker logs "beecr-api"
```

Вывод должен быть похож на следующий:

```
...
{"loglevel": "info", "workers": 4, "bind": "0.0.0.0:80", "graceful_timeout": 120, "timeout": 3600, "keepalive":
5, "errorlog": "-", "accesslog": "-", "workers_per_core": 1.0, "use_max_workers": 4, "host": "0.0.0.0", "port":
"80"}
  License check passed. License info: GPU based license.
INFO  [BeeCR] Starting gunicorn 20.1.0
INFO  [BeeCR] Listening at: http://0.0.0.0:80 (1)
INFO  [BeeCR] Using worker: app.gunicorn_conf.MyUvicornWorker
INFO  [BeeCR] Booting worker with pid: 10
INFO  [BeeCR] Booting worker with pid: 12
...
```

Обратите внимание на логи, связанные с проверкой лицензии. Если все работает правильно, вы должны увидеть сообщение `License check passed` с дополнительными деталями о лицензии.

Также проверьте логи контейнера, обслуживающего ИИ модель:

```
docker logs "beecr-ai"
```

Присутствие модели GPU в логах - хороший знак, указывающий на то, что ИИ сервер использует видеокарту.

```
...
time=2024-06-06T15:49:27.338Z level=INFO source=types.go:98 msg="inference compute" id=GPU-96b9ff66-b2f6-
e4c5-123a-eac0416a2789 library=cuda compute=8.9 driver=12.2 name="NVIDIA GeForce RTX 4090" total="23.6 GiB"
available="23.3 GiB"
...
```

7 Проверьте работу API сервера

Наконец, давайте убедимся, что сервер API отвечает, отправив GET HTTP-запрос к конечной точке `/version`.

Примечание: Рекомендуется выполнить этот запрос извне сервера, чтобы подтвердить, что ваш брандмауэр (если настроен) разрешает входящие запросы.

Используйте следующую команду для отправки запроса:

```
curl 'http://SERVER:8000/version'
```

Замените `SERVER` на адрес вашего целевого сервера.

Если все работает правильно, вы должны получить текстовый ответ с версией API сервера, например:

```
v1.27.1
```

Заключение

Из данного материла вы узнали, как развернуть BeeCR на собственных серверах с помощью Docker. Следуя шагам изложенным в данном руководстве, вы настроили необходимую среду, получили необходимую лицензию, загрузили Docker-образы, настроили параметры Docker Compose и запустили контейнеры. Проверка логов позволяет убедиться минимальным образом, что серверы работают как ожидается. Если у вас возникнут проблемы или возникнут дополнительные вопросы, обращайтесь в нашу поддержку.

1.2 Напрямую в системе

1.2.1 Пошаговое руководство по развертыванию BeeCR через установочный скрипт

Примечание: В данному руководстве описана установка BeeCR на собственный сервер с помощью установочного скрипта. Если вы являетесь опытным пользователем и хотите контролировать многие аспекты процесса установки и настройки, обратитесь к расширенному руководству "Пошаговое руководство по развертыванию BeeCR вручную".

Данный документ представляет собой пошаговое руководство по установке и настройке решения BeeCR на собственных серверах без использования Docker. Процесс установки продемонстрирован на сервере с графическим процессором "NVIDIA Tesla V100", работающим под управлением операционной системы "Linux Ubuntu 20.04 LTS". Это руководство призвано помочь пользователям пройти установочный процесс, убедившись, что все предварительные требования выполнены и необходимые компоненты установлены и настроены правильно.

Примечание: По состоянию на лето 2024 года дистрибутив Ubuntu 20.04 LTS по-прежнему поддерживается компанией Canonical, но является несколько устаревшим. Мы выбрали эту версию ОС для данного руководства, потому что для настройки окружения требуются дополнительные шаги, и мы хотим их продемонстрировать. Если вы используете более новую версию Ubuntu, просто пропустите шаги, которые вам не требуются.

Требования

Для развертывания решения на сервере вам понадобится следующее:

1. Сервер с архитектурой "x86_64" под управлением Linux (желательно Ubuntu или Debian).
2. Графический ускоритель NVidia (минимум 24 ГБ памяти, рекомендуется 40 ГБ).
3. Утилита командной строки `curl`.
4. Python (поддерживаются версии 3.9 – 3.12) и утилита командной строки `virtualenv`.
5. Пользователь, запускающий процесс установки, должен иметь привилегии "root" или "sudoer".
6. Архив с установочными файлами BeeCR.

1 Запустите скрипт установки

Запустите скрипт установки `install.sh` с опцией `-i` (означает "install").

```
sh ./install.sh -i
```

Примечание: Укажите опцию командной строки `-6`, если вы хотите привязать API сервер к сокету IPv6.

Пример вывода:

```
WARNING [BeeCR Installer]    Current user is not root. This script will ask for superuser permissions.
INFO    [BeeCR Installer]    Step 1 - Check environment...
INFO    [BeeCR Installer]    OS is Linux.
INFO    [BeeCR Installer]    Architecture is "x86_64".
INFO    [BeeCR Installer]    Kernel is non-WSL.
INFO    [BeeCR Installer]    "cURL" is available.
ERROR   [BeeCR Installer]    The "virtualenv" (Python virtual environment creator) is not available. You can
install it, for example, using your system package manager (search for the "python3-virtualenv" package or
similar).
```

Установщик проверил окружение и сообщил об ошибке, связанной с Python. Для работы требуется утилита `virtualenv` (для создания виртуальной среды Python), которая отсутствует.

2 Установите Python virtualenv

Пропустите этот шаг, если `virtualenv` уже установлен на вашей системе.

Чтобы установить `virtualenv` на Ubuntu, используйте менеджер пакетов `apt`:

```
sudo apt-get update
sudo apt-get install -y python3-virtualenv
```

3 Повторно запустите скрипт установки

После установки `virtualenv` повторно запустите скрипт установки.

Пример вывода:

```
WARNING [BeeCR Installer]    Current user is not root. This script will ask for superuser permissions.
INFO    [BeeCR Installer]    Step 1 - Check environment...
INFO    [BeeCR Installer]    OS is Linux.
INFO    [BeeCR Installer]    Architecture is "x86_64".
INFO    [BeeCR Installer]    Kernel is non-WSL.
INFO    [BeeCR Installer]    "cURL" is available.
ERROR   [BeeCR Installer]    Python 3.9 (or higher) is required.
INFO    [BeeCR Installer]  Note:
    Installing Python 3.9 alongside older versions of Python may be tricky and
    highly dependent on your Linux distribution.

    Once you have installed Python 3.9 (e.g., using your system package manager),
    consider updating symbolic links to your Python interpreter and shared libraries.

    For example, on Ubuntu 20.04 with Python 3.8 preinstalled, you may need to
    run the following commands as a superuser:
        sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.8 1
        sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.9 2
        sudo update-alternatives --install /usr/bin/python python /usr/bin/python3 1
        sudo ln -s /usr/lib/python3/dist-packages/apt_pkg{.cpython-38-x86_64-linux-gnu,}.so
```

Установщику по-прежнему не удается выполнить установку, потому что Ubuntu 20.04 поставляется с предустановленным Python 3.8. Скрипт предлагает подсказки по установке и настройке более новой версии Python (например, 3.9).

4 Установите более новую версию Python

Пропустите этот шаг, если ваша установка Python уже соответствует требованиям (версия 3.9 – 3.12).

Для установки Python 3.9 на Ubuntu используйте менеджер пакетов `apt`:

```
sudo apt-get update
sudo apt-get install -y python3.9
```

После установки более новой версии Python вам может потребоваться сделать ее основным интерпретатором Python. Используйте утилиту командной строки `update-alternatives`:

```
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.8 1
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.9 2
sudo update-alternatives --install /usr/bin/python python /usr/bin/python3 1
sudo ln -s /usr/lib/python3/dist-packages/apt_pkg{.cpython-38-x86_64-linux-gnu,}.so
```

Команды выше делают следующее:

1. Устанавливают приоритет `python3.8` (старая версия Python) на уровень `1` для псевдонима `python3`.
2. Устанавливают более высокий приоритет для `python3.9` (новая версия Python) на уровень `2` для псевдонима `python3`.
3. Устанавливают приоритет для псевдонима `python` на `python3`.
4. Исправляют потенциальную проблему с `apt_pkg`, создавая символическую ссылку на общую библиотеку, принадлежащую более старой версии Python.

5 Повторно запустите скрипт установки

После того как установленная версия Python удовлетворяет требованиям (3.9 – 3.12), повторно запустите скрипт установки.

Примечание: Если вы запускаете скрипт от имени обычного пользователя, он запросит привилегии суперпользователя. Убедитесь, что у вас есть привилегии sudo и можете вводить свой пароль во время процесса установки.

Если все верно, вывод будет похож на следующий (для наглядности логи сторонних команд/скриптов скрыты за многоточием):

```
WARNING [BeeCR Installer]    Current user is not root. This script will ask for superuser permissions.
INFO    [BeeCR Installer]    Step 1 - Check environment...
INFO    [BeeCR Installer]    OS is Linux.
INFO    [BeeCR Installer]    Architecture is "x86_64".
```

```
INFO     [BeeCR Installer]    Kernel is non-WSL.
INFO     [BeeCR Installer]    "cURL" is available.
INFO     [BeeCR Installer]    "Python 3" and "virtualenv" are available.
INFO     [BeeCR Installer]    Step 2 - Install AI server...
...
INFO     [BeeCR Installer]    AI server installed.
INFO     [BeeCR Installer]    Step 3 - Configure AI server...
INFO     [BeeCR Installer]    AI server configured.
INFO     [BeeCR Installer]    Step 4 - Pull AI model...
...
INFO     [BeeCR Installer]    AI model pulled.
...
INFO     [BeeCR Installer]    Step 5 - Install ASGI server to serve API...
...
INFO     [BeeCR Installer]    "ASGI server" installed.
INFO     [BeeCR Installer]    Step 6 - Install API server...
...
INFO     [BeeCR Installer]    "API server" installed.
INFO     [BeeCR Installer]    Step 7 - Configure API server...
...
WARNING [BeeCR Installer]    API server license is missing. Put the license file "beecr.lic" to the "/opt/
beecr" directory and restart the service using command "systemctl restart beecr".
INFO     [BeeCR Installer]  Note: To request a license file, please provide this information to
beecr@cvisionlab.com: "GPU 0: Tesla V100-PCIE-32GB (UUID: GPU-d541606d-4910-6857-7cc0-4821bec3f358)".
```

6 Получите и установите лицензию

Как видно из предыдущего шага, все проверки предварительных требований должны быть пройдены, и все компоненты должны быть установлены. Однако скрипт установки предупреждает, что отсутствует лицензия на сервер API.

Если у вас нет файла лицензии, выполните следующие шаги:

1. Свяжитесь с нашей командой (например, через электронную почту по адресу beecr@cvisionlab.com).
2. Предоставьте нашей команде информацию о вашем графическом процессоре (например, `GPU 0: Tesla V100-PCIE-32GB (UUID: GPU-d541606d-4910-6857-7cc0-4821bec3f358)`).

После получения файла лицензии (он должен называться `beecr.lic`), поместите его в директорию `/opt/beecr` и перезапустите службу `beecr`, выполнив следующую команду:

```
sudo systemctl restart beecr
```

7 Проверка логов и донастройка *(опционально)*

Скрипт установки создает "systemd" службу с именем `beecr`. Используйте стандартные команды "systemd" для управления службой.

Например, чтобы просмотреть логи, используйте утилиту командной строки `journalctl`:

```
journalctl -fu beecr
```

Пример вывода:

```
Jul 16 10:53:42 code-review-2004-v100 python3[41610]: INFO        [BeeCR--]        Uvicorn running on http://
0.0.0.0:8000 (Press CTRL+C to quit)
Jul 16 10:53:42 code-review-2004-v100 systemd[1]: beecr.service: Succeeded.
Jul 16 10:53:45 code-review-2004-v100 systemd[1]: Stopped BeeCR Service.
Jul 16 10:53:45 code-review-2004-v100 systemd[1]: Started BeeCR Service.
Jul 16 10:53:46 code-review-2004-v100 python3[41624]:   License check passed. License info: GPU based license.
Jul 16 10:53:46 code-review-2004-v100 python3[41624]: INFO        [BeeCR--]        Started server process
[41624]
Jul 16 10:53:46 code-review-2004-v100 python3[41624]: INFO        [BeeCR--]        Waiting for application
startup.
Jul 16 10:53:46 code-review-2004-v100 python3[41624]: INFO        [BeeCR--]        Settings: pkg_name='beecr-
api' version='v1.26.2' domain='beecr.dev' protocol='http' port=8000 model='ollama/codestral:22b' temperature=0.
8 use_original_code=True language='English' openai_host='https://api.openai.com' openai_api_key='*****'
ollama_host='http://localhost:11435' gitlab_host='https://gitlab.com' target=None target_extra=None
target_exclude=None.
Jul 16 10:53:46 code-review-2004-v100 python3[41624]: INFO        [BeeCR--]        Application startup complete.
Jul 16 10:53:46 code-review-2004-v100 python3[41624]: INFO        [BeeCR--]        Uvicorn running on http://
0.0.0.0:8000 (Press CTRL+C to quit)
```

Сообщение " License check passed" указывает на то, что API сервер проверки кода видит и принимает лицензию.

Для управления настройками API сервера отредактируйте соответствующие файлы конфигурации службы:

- `/etc/systemd/system/beecr.service` – основные настройки (например, порт);
- `/etc/systemd/system/beecr.service.d/env.conf` – дополнительные настройки.

Заключение

Следуя данному руководству, вы успешно установили и настроили решение BeeCR для проверки кода на своем собственном сервере. Вы узнали, как решать распространенные проблемы при установке, такие как отсутствие требуемых утилит, устаревшая версия Python. Также изучили приёмы управления службой с помощью "systemd". Теперь вы можете использовать возможности BeeCR для улучшения процессов проверки вашего кода. Если у вас возникнут дополнительные проблемы или вам понадобится лицензия, пожалуйста, свяжитесь с нашей командой.

1.2.2 Пошаговое руководство по развертыванию BeeCR в вашей инфраструктуре вручную

Примечание: Это руководство предназначено для опытных пользователей, которые хотят контролировать различные аспекты процесса установки. Если вы ищете более простое руководство, обратитесь к документу "Пошаговое руководство по развертыванию BeeCR через установочный скрипт".

Данный документ представляет собой пошаговое руководство по установке и настройке решения BeeCR на собственных серверах без использования Docker. Процесс установки продемонстрирован на сервере с графическим процессором "NVIDIA Tesla V100", работающим под управлением операционной системы "Linux Ubuntu 20.04 LTS". Это руководство призвано помочь пользователям пройти установочный процесс, убедившись, что все предварительные требования выполнены и необходимые компоненты установлены и настроены правильно.

Примечание: По состоянию на лето 2024 года дистрибутив Ubuntu 20.04 LTS по-прежнему поддерживается компанией Canonical, но является несколько устаревшим. Мы выбрали эту версию ОС для данного руководства, потому что для настройки окружения требуются дополнительные шаги, и мы хотим их продемонстрировать. Если вы используете более новую версию Ubuntu, просто пропустите шаги, которые не требуются.

Требования

Для развертывания решения на сервере вам понадобится следующее:

1. Сервер с архитектурой "x86_64" под управлением Linux (желательно Ubuntu или Debian).
2. Графический ускоритель NVidia (минимум 24 ГБ памяти, рекомендуется 40 ГБ).
3. Утилита командной строки `curl`.
4. Python (поддерживаются версии 3.9 – 3.12) и утилита командной строки `virtualenv`.
5. Пользователь, запускающий процесс установки, должен иметь привилегии "root" или "sudoer".
6. Вы загрузили архив с установочными файлами BeeCR.

1 Подготовьте окружение

ОПЕРАЦИОННАЯ СИСТЕМА

Убедитесь, что ваша ОС - Linux, архитектура - "x86_64", а используемое ядро не "Microsoft WSL":

```
uname -smr
```

Пример вывода:

```
Linux 5.15.0-71-generic x86_64
```

CURL

Убедитесь, что установлен `curl`. Если нет, установите `curl` через менеджер пакетов (на Ubuntu используйте `apt`):

```
sudo apt-get update
sudo apt-get install -y curl
```

PYTHON И VIRTUALENV

Убедитесь, что установлены Python (3.9 – 3.12) и `virtualenv`.

Для установки Python 3.9 и `virtualenv` на Ubuntu 20.04 LTS используйте менеджер пакетов `apt`:

```
sudo apt-get update
sudo apt-get install -y python3.9 python3-virtualenv
```

Если установлено несколько версий Python, вам может потребоваться настроить версию по умолчанию:

```
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.8 1
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.9 2
sudo update-alternatives --install /usr/bin/python python /usr/bin/python3 1
sudo ln -s /usr/lib/python3/dist-packages/apt_pkg{.cpython-38-x86_64-linux-gnu,}.so
```

Команды выше делают следующее:

1. Устанавливают приоритет `python3.8` (старая версия Python) на уровень `1` для псевдонима `python3`.
2. Устанавливают более высокий приоритет для `python3.9` (новая версия Python) на уровень `2` для псевдонима `python3`.
3. Устанавливают приоритет для псевдонима `python` на `python3`.
4. Исправляют потенциальную проблему с `apt_pkg`, создавая символическую ссылку на общую библиотеку, принадлежащую более старой версии Python.

2 Установите и настройте ИИ сервер

УСТАНОВИТЕ ИИ СЕРВЕР

```
curl -fsSL https://ollama.com/install.sh | OLLAMA_VERSION="0.3.6" sh
```

Создайте дополнительную директорию конфигурации `/etc/systemd/system/ollama.service.d`. Затем поместите туда файл `env.conf` с следующим содержимым:

```
[Service]
Environment="OLLAMA_HOST=0.0.0.0:11435"
Environment="OLLAMA_KEEP_ALIVE=-1"
Environment="OLLAMA_NUM_PARALLEL=2"
```

Перезагрузите "systemd" и включите службу `ollama`:

```
sudo systemctl daemon-reload
sudo systemctl enable ollama
sudo systemctl start ollama
```

Убедитесь, что ИИ сервер запущен:

```
curl -s "http://localhost:11435"
```

СКАЧАЙТЕ ИИ МОДЕЛЬ

```
OLLAMA_HOST="0.0.0.0:11435" ollama pull "codestral:22b"
```

3 Установите и настройте API сервер

СОЗДАЙТЕ ВИРТУАЛЬНОЕ ОКРУЖЕНИЕ ДЛЯ PYTHON

```
sudo virtualenv "/opt/beecr/venv"
```

УСТАНОВИТЕ UVICORN (ASGI-СЕРВЕР)

```
sudo /opt/beecr/venv/bin/pip3 install "uvicorn~=0.30"
```

УСТАНОВИТЕ API СЕРВЕР

Выберите whl-пакет в соответствии с вашей версией Python. Для Python 3.9 установите пакет, содержащий `py39` в имени. Для Python 3.10 установите пакет, содержащий `py310` в имени. И так далее.

Если вы не знаете вашу версию Python, просто выполните команду `/opt/beecr/venv/bin/python3 --version`.

```
sudo /opt/beecr/venv/bin/pip3 install "./path/to/whl/package"
```

НАСТРОЙТЕ API СЕРВЕР

Создайте файл конфигурации "systemd" `/etc/systemd/system/beecr.service` со следующим содержимым:

```
[Unit]
Description=BeeCR Service
After=network-online.target
```

```
[Service]
WorkingDirectory=/opt/beecr
ExecStart=/opt/beecr/venv/bin/python3 -m uvicorn --host 0.0.0.0 --port $PORT --log-config /opt/beecr/
logging.json beecr.api.main:app
Restart=always
RestartSec=3
Environment="PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"
Environment="PORT=8000"

[Install]
WantedBy=default.target
```

Отредактируйте переменную окружения `PORT`, если хотите, чтобы сервер API работал на другом порте.

Примечание: Если вы хотите привязать сервер API к сокету IPv6, замените `0.0.0.0` на `::` в конфигурации службы.

Создайте дополнительную директорию конфигурации `/etc/systemd/system/beecr.service.d`. Затем поместите туда файл `env.conf` с следующим содержимым:

```
[Service]

# Адрес и токен GitLab по умолчанию.
#
# Будут использоваться только в случае, если запросы не передают эти параметры явно.
# Environment="GITLAB_HOST=https://gitlab.com"
# Environment="GITLAB_TOKEN="


# Настройки для использования ИИ через Ollama-совместимое API.
Environment="OLLAMA_HOST=http://localhost:11435"


# Настройки для использования ИИ через OpenAI-совместимое API.
#
# Измените, если хотите использовать ИИ модели через API,
# совместимое с OpenAI, например "gpt4-o".
# Environment="OPENAI_HOST=https://api.openai.com"
# Environment="OPENAI_API_KEY="


# Имя ИИ модели по умолчанию.
#
# Эта модель будет использоваться для обработки API-запросов
# без явно указанного имени модели.
#
# Для API, совместимых с OpenAI, используйте имена моделей,
# как они есть, например, "gpt-4o".
#
# Для API, совместимых с Ollama, используйте имена моделей
# с префиксом "ollama/", например, "ollama/codestral:22b".
Environment="MODEL=ollama/codestral:22b"
```

Создайте рабочую директорию для сервера API `/opt/beecr`. Затем поместите туда файл `logging.json` со следующим содержимым:

```
{
    "version": 1,
    "filters": {
        "correlation_id": {
            "()": "asgi_correlation_id.CorrelationIdFilter",
            "uuid_length": 32,
```

```
            "default_value": "-"
        }
    },
    "formatters": {
        "default": {
            "format": "%(levelname)s\t[BeeCR-%(correlation_id)s]\t%(message)s",
            "datefmt": "%Y-%m-%d %H:%M:%S"
        }
    },
    "handlers": {
        "console": {
            "formatter": "default",
            "filters": ["correlation_id"],
            "class": "logging.StreamHandler",
            "stream": "ext://sys.stdout",
            "level": "DEBUG"
        }
    },
    "root": {
        "handlers": ["console"],
        "level": "INFO"
    },
    "loggers": {
        "uvicorn": {
            "propagate": true
        },
        "uvicorn.access": {
            "propagate": true
        }
    }
}
```

Перезагрузите "systemd" и включите службу `beecr`:

```
sudo systemctl daemon-reload
sudo systemctl enable beecr
```

4 Получите и установите лицензию

Если у вас нет файла лицензии, выполните следующие шаги:

1. Свяжитесь с нашей командой (например, через электронную почту по адресу beecr@cvisionlab.com).
2. Выполните команду `nvidia-smi -L` и предоставьте команде BeeCR информацию о вашем графическом процессоре (например, `GPU 0: Tesla V100-PCIE-32GB (UUID: GPU-d541606d-4910-6857-7cc0-4821bec3f358)`).

После получения файла лицензии (он должен иметь имя `beecr.lic`), поместите его в директорию `/opt/beecr` и перезапустите службу `beecr`, выполнив следующую команду:

```
sudo systemctl restart beecr
```

5 Проверка логов *(опционально)*

Используйте стандартные команды "systemd" для управления службой.

Например, чтобы просмотреть логи, используйте утилиту командной строки `journalctl`:

```
journalctl -fu beecr
```

Пример ожидаемого результата выполнения команды:

```
Jul 16 10:53:42 code-review-2004-v100 python3[41610]: INFO        [BeeCR--]        Uvicorn running on http://
0.0.0.0:8000 (Press CTRL+C to quit)
Jul 16 10:53:42 code-review-2004-v100 systemd[1]: beecr.service: Succeeded.
Jul 16 10:53:45 code-review-2004-v100 systemd[1]: Stopped BeecR Service.
Jul 16 10:53:45 code-review-2004-v100 systemd[1]: Started BeecR Service.
Jul 16 10:53:46 code-review-2004-v100 python3[41624]:   License check passed. License info: GPU based license.
Jul 16 10:53:46 code-review-2004-v100 python3[41624]: INFO        [BeeCR--]        Started server process
[41624]
Jul 16 10:53:46 code-review-2004-v100 python3[41624]: INFO        [BeeCR--]        Waiting for application
startup.
Jul 16 10:53:46 code-review-2004-v100 python3[41624]: INFO        [BeeCR--]        Settings: pkg_name='beecr-
api' version='v1.27.1' domain='beecr.io' protocol='http' port=8000 model='ollama/codestral:22b' temperature=0.8
use_original_code=True language='English' openai_host='https://api.openai.com' openai_api_key='******'
ollama_host='http://localhost:11435' gitlab_host='https://gitlab.com' target=None target_extra=None
target_exclude=None.
Jul 16 10:53:46 code-review-2004-v100 python3[41624]: INFO        [BeeCR--]        Application startup complete.
Jul 16 10:53:46 code-review-2004-v100 python3[41624]: INFO        [BeeCR--]        Uvicorn running on http://
0.0.0.0:8000 (Press CTRL+C to quit)
```

Заключение

Следуя данному руководству, вы успешно установили и настроили решение BeeCR для проверки кода на своем собственном сервере. Вы узнали, как решать распространенные проблемы при установке, такие как отсутствие требуемых утилит, устаревшая версия Python. Также изучили приёмы управления службой с помощью "systemd". Теперь вы можете использовать возможности BeeCR для улучшения процессов проверки вашего кода. Если у вас возникнут дополнительные проблемы или вам понадобится лицензия, пожалуйста, свяжитесь с нашей командой.

1.3 Настройки API-сервера BeeCR

В случае развертывания BeeCR на вашем локальном сервере или виртуальной машине может возникнуть необходимость осуществить дополнительную настройку API сервера. API-сервер BeeCR можно настроить с помощью переменных окружения. В данном разделе документации вы найдете обзор всех переменных окружения, которые вы можете использовать для управления настройками API-сервера.

Примечание:

Клиент API может переопределить любую из настроек в запросе (через HTTP-заголовки и/или параметры запроса и/или тело HTTP-запроса и/или даже через файл конфигурации в репозитории, анализ которого выполняется в данный момент). Если клиент API каким-либо образом передает параметр в запросе, то значение от клиента будет иметь больший приоритет, чем настройки сервера.

1.3.1 Адрес GitLab и токен доступа

Если вы собираетесь использовать предоставленные компоненты/шаблоны CI/CD, то обычно нет необходимости переопределять `GITLAB_HOST` и `GITLAB_TOKEN`, потому что компоненты/шаблоны CI/CD передают адрес GitLab в параметрах запроса.

Пример:

```
GITLAB_HOST='https://gitlab.com'
GITLAB_TOKEN='your-gitlab-token'
```

1.3.2 Модель ИИ

BeeCR поддерживает два альтернативных API для взаимодействия с моделями ИИ: OpenAI и Ollama.

- Для OpenAI-совместимых API используйте имена моделей "как есть" например, `gpt-4o`.
- Для Ollama-совместимых API используйте имена моделей с префиксом `ollama/`, например, `ollama/codestral:22b`.

Пример:

```
MODEL='ollama/codestral:22b'
```

1.3.3 Ollama

Задайте переменную среды `OLLAMA_HOST`, если вы хотите использовать ИИ сервер на базе Ollama. Если же вы собираетесь использовать модели через OpenAI-совместимый API (например, модель `gpt-4o`), то задавать значение опции `OLLAMA_HOST` не имеет смысла.

Пример:

```
OLLAMA_HOST='http://your-ollama-address:port'
```

1.3.4 OpenAI

Задайте переменный среды `OPENAI_HOST` и `OPENAI_API_KEY`, если вы хотите использовать модели семейства ChatGPT (или любую другую модель через API, совместимый с OpenAI). Если же вы собираетесь использовать модель через Ollama-совместимый API, то задавать значения опций `OPENAI_HOST` и `OPENAI_API_KEY` не имеет смысла.

- `OPENAI_HOST`: Адрес OpenAI-совместимого API сервера. По умолчанию `https://api.openai.com`.
- `OPENAI_API_KEY`: Ваш ключ доступа к OpenAI API.

Пример:

```
OPENAI_HOST='https://api.openai.com'
OPENAI_API_KEY='your-gitlab-token'
```

1.3.5 Определение файлов, подлежащих анализу

Основная настройка (target)

`TARGET` задаёт регулярное выражение, которое определяет файлы, подлежащие проверке.

Значение по умолчанию:

```
\.(py|c|h|cpp|hpp|cs|java|kt|swift|php|go|sh|(j|t)sx?)$
```

Использование данного регулярного выражения приведет к проверке файлов для следующих языков программирования:

- Python: `.py`
- C/C++: `.h`, `.hpp`, `.c`, `.cpp`
- C#: `.cs`
- Java: `.java`
- Kotlin: `.kr`
- Swift: `.swift`
- PHP: `.php`
- Go: `.go`
- Bash/Shell: `.sh`
- JavaScript/Typescript: `.js`, `.jsx`, `.mjs`, `.mjsx`, `.ts`, `.tsx`

Однако вы можете настроить ее так, как вам удобно, чтобы поддерживать больше языков или, наоборот, исключить некоторые из стандартных. Пример регулярного выражения:

```
\.(java|hs)$
```

Дополнительная настройка (target extra)

`TARGET_EXTRA` позволяет задать дополнительное регулярное выражение. Полезно, например, в тех случаях, когда вы не хотите переопределять `TARGET`, а лишь "дополнить" его.

По умолчанию параметр `TARGET_EXTRA` не задан.

Пример:

```
TARGET_EXTRA='\.(json|yml)$'
```

Настройка исключений (target exclude)

`TARGET_EXCLUDE` позволяет задать регулярное выражение для исключения файлов из проверки (даже если они соответствуют `TARGET` или `TARGET_EXTRA`). Полезно, например, если вы не хотите переопределять `TARGET`.

По умолчанию параметр `TARGET_EXCLUDE` не задан.

Пример:

```
TARGET_EXCLUDE='\.(sh)$'
```

1.3.6 Язык

Мы рекомендуем использовать английский язык, так как в этому случае комментарии от модели ИИ наиболее полные и качественные для всех протестированных моделей.

Если вы собираетесь использовать предоставленные компоненты/шаблоны CI/CD, то обычно нет необходимости задавать язык, так как компоненты/шаблоны CI/CD передают значение данного параметра вместе с запросом на анализ.

Значение по умолчанию: `English`.

Пример:

```
LANGUAGE='English'
```

1.3.7 Конфигурационный файл в репозитории

По умолчанию, API сервер будет пытаться в т.ч. получить настройки из файла `.beecr.yml` в корне репозитория проекта. При желании путь к файлу можно переопределить через параметр `REPO_CONFIG_PATH`.

Пример:

```
REPO_CONFIG_PATH='.beecr.yml'
```

1.3.8 Ключевое выражение-триггер в комментариях

В случае интеграции GitLab и BeeCR с помощью вебхуков для событий "Comment" изменения в запросах на слияния (Merge Requests) будут анализироваться только в том случае, если появился комментарий, который содержит выражение (триггер) `/beecr`. При желании триггер можно переопределить через параметр `NOTE_TRIGGER`.

Пример:

```
NOTE_TRIGGER='/beecr'
```

2. Интеграция с GitLab

2.1 Вебхук BeeCR для GitLab

Примечание: Этот раздел описывает настройку проверки кода в GitLab через Вебхук. Если вы предпочитаете использовать CI/CD, обратитесь к альтернативным методам интеграции:

- Через компонент задачи CI/CD (для GitLab 17+): обзор компонента, пошаговое руководство;
- Через шаблон задачи CI/CD (подходит и для более старых версий GitLab): обзор шаблона, пошаговое руководство.

2.1.1 Краткая инструкция

1. Создайте новый вебхук в вашем проекте на GitLab.
2. Установите полный URL вебхука, ссылающийся на конечную точку `/gitlab-review?asynchronous=True`. Например, для версии SaaS полный URL выглядит следующим образом:

```
https://beecr.io/api/v1/gitlab-review?asynchronous=True
```

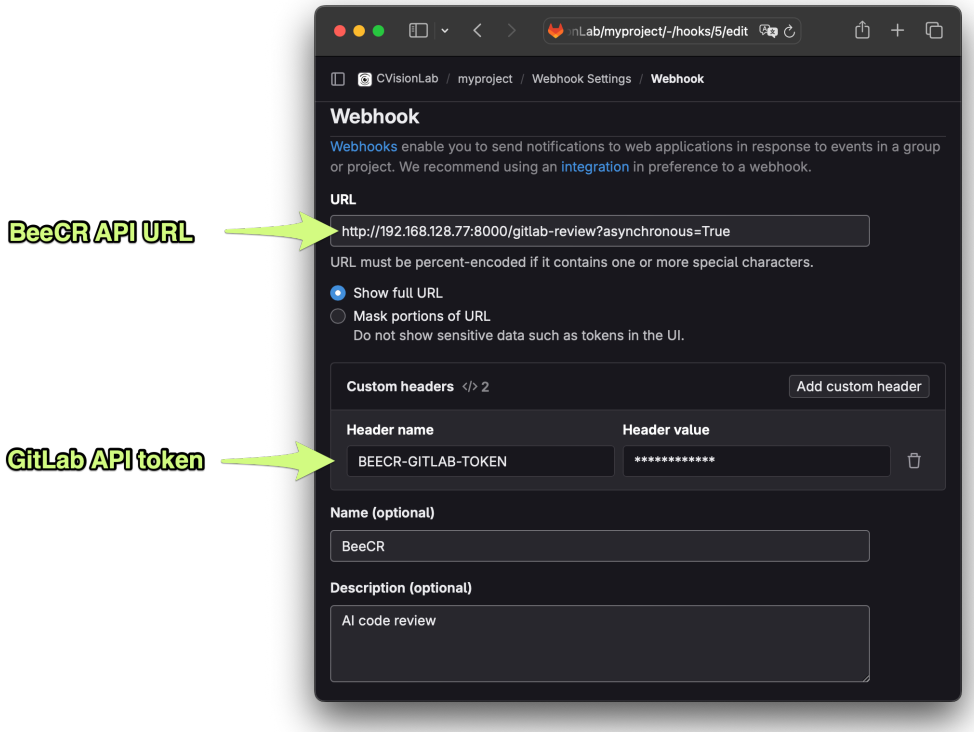
3. Определите следующие HTTP-заголовки:

- `BEECR-API-KEY` – ваш API-ключ, предоставленный вам командой BeeCR (необходим только при использовании BeeCR SaaS).
- `BEECR-GITLAB-TOKEN` – токен доступа к GitLab, который вы должны выпустить для проекта или группы проектов). Токен должен иметь права как минимум уровня "Developer" и предоставлять доступ к "api".

  Совет: Мы рекомендуем присвоить токену осмысленное имя, например, BeeCR Code Reviewer, поскольку комментарии к коду будут написаны ботом, представляющимся этим именем.

- `BEECR-OPENAI-API-KEY` – ваш API-ключ доступа к OpenAI (необходим только при использовании моделей от OpenAI).

4. Установите триггер на события `Comments`.



Теперь вы можете запустить проверку изменений в файле в вашем открытом запросе на слияние (Merge Request), оставив комментарий с со специальным выражением `/beecr` к конкретному файлу.

2.1.2 URL

Полный URL вебхука, ссылающийся на конечную точку `/gitlab-review?asynchronous=True`.

Полный URL должен содержать адрес API-сервера BeeCR. Конкретное значение зависит от того, где размещен сервер API:

- Для версии SaaS используйте следующий URL:
  `https://beecr.io/api/v1/gitlab-review?asynchronous=True`
- Если сервер API BeeCR размещен локально, вам нужно самостоятельно определить адрес. Например:
  `http://192.168.128.77:8000/gitlab-review?asynchronous=True`

Параметр `asynchronous=True` даёт указание API-серверу BeeCR выполнять запрос в асинхронном режиме. Это означает, что сервер ответит быстро, но запланирует фактическую проверку во внутренней очереди обработки. Это является важным аспектом, так как GitLab требует, чтобы вебхуки отвечали примерно за 10 секунд или быстрее.

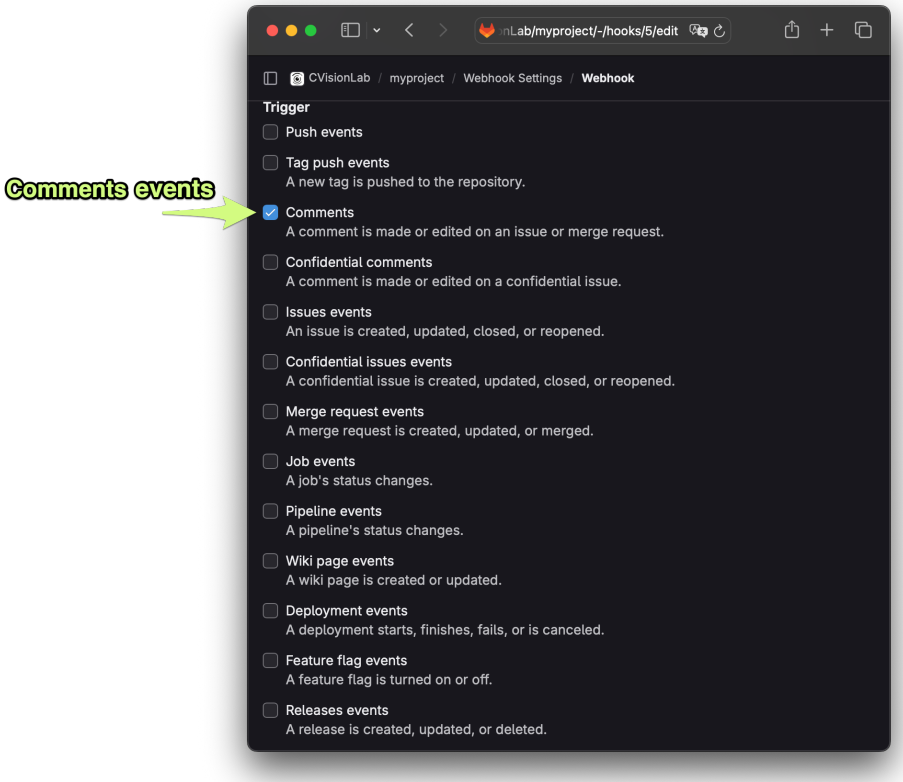
2.1.3 Выбор событий для реакции

GitLab позволяет вам активировать срабатывание вебхука через различные типы событий. Вебхук BeeCR поддерживает следующие два типа:

- Comments
- Merge Request

События "Comments"

В случае выбора событий типа `Comments`, BeeCR будет проверять изменения в запросах на слияние если кто-то оставляет комментарий, содержащий специальное выражение `/beecr` (можно изменить при необходимости).



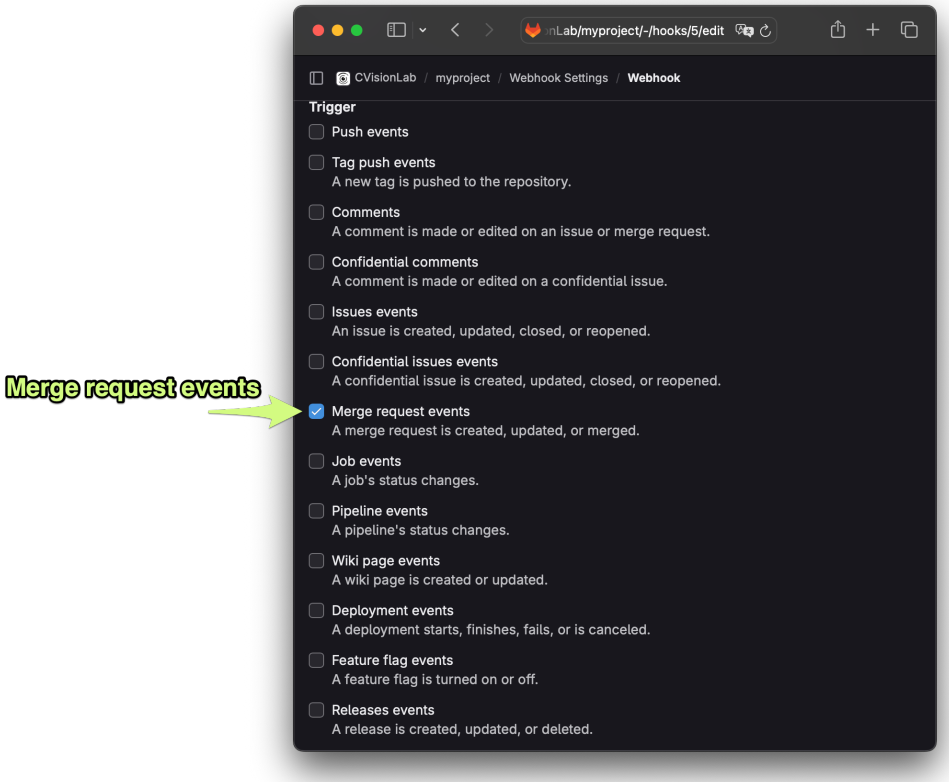
В зависимости от того, где именно оставлен комментарий, BeeCR будет проверять изменения в одном конкретном файле или изменения во всех целевых файлах:

- Если комментарий оставлен где-либо в обсуждении, связанном с изменениями в определенном файле, то будет проверен только этот файл. В этом случае любые настройки, определяющие набор целевых файлов, будут проигнорированы.
- Если же комментарий оставлен ко всему запросу на слияние, а не к конкретному файлу в рамках запроса, то будут проверены изменения во всех "целевых" файлах.

  Примечание: Список "целевых" файлов для проверки по умолчанию пуст. Убедитесь, что вы настроили цели либо в параметрах вебхука, либо в настройках API-сервера (если вы используете BeeCR, развернутый на вашем сервере).

События "Merge Request"

В случае выбора событий типа `Merge Request`, BeeCR автоматически будет проверять изменения в "целевых" файлах каждый раз, когда создается запрос на слияние или код в запросе на слияние обновляется.



Примечание: Список "целевых" файлов для проверки по умолчанию пуст. Убедитесь, что вы настроили цели либо в параметрах вебхука, либо в настройках API-сервера (если вы используете BeeCR, развернутый на вашем сервере).

2.1.4 Конфигурирование

Большинство параметров BeeCR можно настроить четырьмя способами:

1. Через HTTP-заголовки.
2. Через параметры запроса (т.е. как часть URL).

   Примечание: Согласно техническим стандартам web при передачи параметров в URL требуется проведение процедуры кодирования (т.н. **URL encoding**).
   При передаче параметров через URL убедитесь, что вы применили URL encoding, используя любой удобный для вас инструмент (например, через online ресурс https://urlencoder.org).
3. Через файл конфигурации `.beecr.yml` в корне вашего репозитория. Пример:

```
target: '\.(py|c|h|cpp|hpp|cs|sh|(j|t)sx?|md|tf|vue)$'
language: English
```

4. В настройках API-сервера (если вы используете BeeCR, развернутый на вашем сервере).

Мы рекомендуем:

- В общем случае:
  - устанавливайте все секреты (API-ключи и токены) через HTTP-заголовки, т.к. это наиболее безопасно;
  - устанавливайте все остальные параметры через конфигурационный файл `.beecr.yml` (например, "целевые" файлы для проверки, модель, язык и т.д.), т.к. это наиболее удобно.
- Для пользователей, устанавливающих API-сервер локально:
  - рассмотрите возможность задания некоторых параметров через настройки API-сервера, если посчитаете это более удобным.

Примечание: Ниже вы найдете описание поддерживаемых параметров и способы их определения. Пожалуйста, уделите особое внимание параметрам, отмеченным .

API-ключ (SaaS)

API-ключ требуется только при использовании облачной версии BeeCR SaaS.

Способы задания ключа API:

- HTTP-заголовок `BEECR-API-KEY` в настройках вебхука (**рекомендованный способ**).
- Параметр запроса `api-key` в URL вебхука.

Токен доступа к GitLab

Для корректной работы BeeCR требуется выпустить токен доступа к GitLab (проектный или групповой). Токен должен иметь права как минимум уровня "Developer" и предоставлять доступ к "api".

Совет: Дайте токену осмысленное имя, например, "BeeCR Code Reviewer", поскольку комментарии к коду будут написаны ботом, представляющимся этим именем.

Способы задания токена доступа к GitLab:

- HTTP-заголовок `BEECR-GITLAB-TOKEN` в настройках вебхука (**рекомендованный способ**).
- Параметр запроса `gitlab-token` в URL вебхука.
- Настройка API-сервера `GITLAB_TOKEN` (применимо только если вы используете BeeCR, развернутый на вашем сервере).

Адрес GitLab

В большинстве случаев при использовании интеграции BeeCR через вебхуки вам не нужно заботится о задании адреса GitLab, так как BeeCR получает адрес автоматически из данных, которые GitLab передаёт в

теле HTTP-запроса. Однако в некоторых случаях (например, нетривиальная настройка вашей локальной сети) автоматическое определение адреса может не сработать и тогда вам потребуется задать адрес явно.

Способы задания адреса GitLab:

- HTTP-заголовок `BEECR-GITLAB-HOST` в настройках вебхука.
- Параметр запроса `gitlab-host` в URL вебхука.
- Настройка API-сервера `GITLAB_HOST` (применимо только если вы используете BeeCR, развернутый на вашем сервере).

Значение адреса GitLab умолчанию: `https://gitlab.com`.

Модель ИИ

BeeCR может использовать различные модели искусственного интеллекта через два альтернативных API: OpenAI и Ollama. При желании пользователи могут сами выбрать, какую модель через какое API использовать:

- В случае использование OpenAI API, используйте оригинальные названия моделей (т.е. "как есть"). Пример: `gpt-4o`.
- В случае использование Ollama API, используйте названия моделей с префиксом `ollama/`. Пример: `ollama/codestral:22b`.

Значение по умолчанию:

- `gpt-4o` в BeeCR SaaS (облачная версия).
- `ollama/codestral:22b` в BeeCR on-premises (версия для установки на вашем сервере).

Способы задания модели:

- HTTP-заголовок `BEECR-MODEL` в настройках webhook.
- Параметр запроса `model` в URL webhook.
- Ключ `model` в конфигурационном файле `.beecr.yml` в корне вашего репозитория.
- Настройка API-сервера `MODEL` (применимо только если вы используете BeeCR, развернутый на вашем сервере).

Ollama

Примечание: Информация в этом разделе актуальна только в том случае, если вы собираетесь использовать модели через Ollama API.

Для использования моделей через Ollama, вам может потребоваться определить адрес сервера Ollama.

Способы определения адреса сервера Ollama:

- HTTP-заголовок `BEECR-OLLAMA-HOST` в настройках webhook.
- Параметр запроса `ollama-host` в URL webhook.
- Ключ `ollama-host` в конфигурационном файле `.beecr.yml` в корне вашего репозитория.
- Настройка API-сервера `OLLAMA_HOST` (**рекомендованный способ**; применимо только если вы используете BeeCR, развернутый на вашем сервере).

OpenAI

Примечание: Информация в этом разделе актуальна только в том случае, если вы собираетесь использовать модели через OpenAI API.

Для использования моделей через OpenAI API, вам может потребоваться определить адрес сервера OpenAI и API-ключ. По умолчанию адрес OpenAI установлен как `https://api.openai.com`, что подходит большинству пользователей.

Способы задания адреса OpenAI:

- HTTP-заголовок `BEECR-OPENAI-HOST` в настройках вебхука.
- Параметр запроса `openai-host` в URL вебхука.
- Ключ `openai-host` в конфигурационном файле `.beecr.yml` в корне вашего репозитория.
- Настройка API-сервера `OPENAI_HOST` (**рекомендованный способ**; применимо только если вы используете BeeCR, развернутый на вашем сервере).

Способы задания API-ключа:

- HTTP-заголовок `BEECR-OPENAI-API-KEY` в настройках вебхука (**рекомендованный способ**).
- Параметр запроса `openai-api-key` в URL вебхука.
- Ключ `openai-api-key` в конфигурационном файле `.beecr.yml` в корне вашего репозитория.
- Настройка API-сервера `OPENAI_API_KEY` (применимо только если вы используете BeeCR, развернутый на вашем сервере).

Настройки цели для проверки

ОСНОВНАЯ НАСТРОЙКА (TARGET)

Параметр "target" позволяет задать регулярное выражение для определения файлов, подлежащих проверке.

Значение по умолчанию:

```
\.(py|c|h|cpp|hpp|cs|java|kt|swift|php|go|sh|(j|t)sx?)$
```

Использование данного регулярного выражения приведет к проверке файлов для следующих языков программирования:

- Python: `.py`
- C/C++: `.h`, `.hpp`, `.c`, `.cpp`
- C#: `.cs`
- Java: `.java`
- Kotlin: `.kr`
- Swift: `.swift`
- PHP: `.php`
- Go: `.go`
- Bash/Shell: `.sh`
- JavaScript/Typescript: `.js`, `.jsx`, `.mjs`, `.mjsx`, `.ts`, `.tsx`

Однако вы можете настроить ее так, как вам удобно, чтобы поддерживать больше языков или, наоборот, исключить некоторые из стандартных. Пример регулярного выражения:

```
\.(java|hs)$
```

Способы задания параметра "target":

- HTTP-заголовок `BEECR-TARGET` в настройках вебхука.
- Параметр запроса `target` в URL вебхука.
- Ключ `target` в конфигурационном файле `.beecr.yml` в корне вашего репозитория (**рекомендованный способ**).
- Настройка API-сервера `BEECR_TARGET` (применимо только если вы используете BeeCR, развернутый на вашем сервере).

ДОПОЛНИТЕЛЬНАЯ НАСТРОЙКА (TARGET EXTRA)

Параметр "target extra" аналогичен параметру "target". Он позволяет задать дополнительное регулярное выражение для определения файлов, подлежащих проверке. Наличие такой настройки полезно в тех случаях, когда вы не хотите полностью переопределять "target", а лишь немного расширить его.

По умолчанию, параметр "target extra" пуст.

Способы задания параметра "target extra":

- HTTP-заголовок `BEECR-TARGET-EXTRA` в настройках вебхука.
- Параметр запроса `target-extra` в URL вебхука.
- Ключ `target-extra` в конфигурационном файле `.beecr.yml` в корне вашего репозитория.
- Настройка API-сервера `BEECR_TARGET_EXTRA` (применимо только если вы используете BeeCR, развернутый на вашем сервере).

НАСТРОЙКА ИСКЛЮЧЕНИЙ (TARGET EXCLUDE)

Параметр "target exclude" позволяет задать регулярное выражение для исключения файлов из проверки (даже если они подпадают под действие "target" или "target extra"). Наличие такой настройки полезно в тех случаях, когда вы не хотите полностью переопределять "target" или "target extra".

По умолчанию, параметр "target exclude" пуст.

Способы задания параметра "target exclude":

- HTTP-заголовок `BEECR-TARGET-EXCLUDE` в настройках вебхука.
- Параметр запроса `target-exclude` в URL вебхука.
- Ключ `target-exclude` в конфигурационном файле `.beecr.yml` в корне вашего репозитория.
- Настройка API-сервера `BEECR_TARGET_EXCLUDE` (применимо только если вы используете BeeCR, развернутый на вашем сервере).

Язык проверки

По умолчанию, комментарии к проверке будут создаваться на английском языке (`English`). Мы рекомендуем использовать английский язык, так как в этому случае комментарии от модели ИИ наиболее полные и качественные для всех протестированных моделей.

Способы задания "языка":

- HTTP-заголовок `BEECR-LANGUAGE` в настройках вебхука.
- Параметр запроса `language` в URL вебхука.
- Ключ `language` в конфигурационном файле `.beecr.yml` в корне вашего репозитория.
- Настройка API-сервера `BEECR_LANGUAGE` (применимо только если вы используете BeeCR, развернутый на вашем сервере).

Путь к конфигурационному файлу в репозитории

Как отмечалось ранее, по умолчанию API-сервер попытается получить настройки в т.ч. из файла `.beecr.yml` в корне репозитория проекта. Такой вариант подойдет большинству пользователей. Однако вы можете задать другой путь к файлу (относительно корня репозиторя) если пожелаете.

Способы задания пути к конфигурационному файлу в репозитории:

- HTTP-заголовок `BEECR-REPO-CONFIG-PATH` в настройках вебхука.
- Параметр запроса `repo-config-path` в URL вебхука.
- Настройка API-сервера `REPO_CONFIG_PATH` (применимо только если вы используете BeeCR, развернутый на вашем сервере).

Специальное выражение-триггер для комментариев

В случае интеграции BeeCR через вебхуки для событий "Comments" проверка изменений в запросе на слияние будет выполнятся в том случае, если кто-то оставил комментарий в соответствующем запросе на слияние с упоминанием специального выражения `/beecr`:

- Если комментарий оставлен где-либо в обсуждении, связанном с изменениями в определенном файле, то будет проверен только этот файл.
- Если же комментарий оставлен ко всему запросу на слияние, а не к конкретному файлу в рамках запроса, то будут проверены изменения во всех "целевых" файлах.

Вы можете задать другое выражение-триггер вместо `/beecr`, если пожелаете. Способы задания выражения-триггера:

- HTTP-заголовок `BEECR-NOTE-TRIGGER` в настройках вебхука.
- Параметр запроса `note-trigger` в URL вебхука.
- Ключ `note-trigger` в конфигурационном файле `.beecr.yml` в корне вашего репозитория.
- Настройка API-сервера `BEECR_NOTE_TRIGGER` (применимо только если вы используете BeeCR, развернутый на вашем сервере).

2.2 Через компонент CI/CD (для GitLab 17+)

2.2.1 CI/CD компонент BeeCR (для GitLab 17+)

> **Примечание:** Данный раздел описывает настройку проверки кода в проекте на GitLab через использование компонента GitLab CI/CD (поддерживается GitLab-ом начиная с версии 17). Если вы хотите настроить проверку кода с помощью шаблонов (поддерживается и в более старых версиях GitLab), обратитесь к документу "CI/CD шаблон BeeCR". Если же вы предпочитаете вовсе не использовать CI/CD, то обратитесь к альтернативному методу интеграции через "Webhook".

Краткая инструкция

1. Определите следующие переменные CI/CD в вашем проекте (или группе проектов):

- `BEECR_API_KEY` – API-ключ, предоставленный вам командой BeeCR (необходим только при использовании BeeCR SaaS).
- `BEECR_GITLAB_TOKEN` – токен доступа к GitLab, который вы должны выпустить для проекта или группы проектов). Токен должен иметь права как минимум уровня "Developer" и предоставлять доступ к "api".

> **Совет:** Мы рекомендуем присвоить токену осмысленное имя, например, BeeCR Code Reviewer, поскольку комментарии к коду будут написаны ботом, представляющимся этим именем.

2. В настройках вашего проекта убедитесь, что CI/CD включен и доступен хотя бы один сервер запуска задач (GitLab runner).
3. Создайте (или измените, если уже существует) конфигурационный файл `.gitlab-ci.yml` в корне вашего репозитория. Добавьте этап проверки кода в список CI/CD этапов (stages). Подключите CI/CD компонент BeeCR с помощью ключевого слова `include`.

```yaml
stages:
  - review

include:
  - component: gitlab.com/cvisionlab/beecr/beecr@1.4.2
```

Особенности

1. Задача запускается только для запросов на слияние кода (Merge Request).
2. Результаты проверки добавляются в виде комментариев к файлам.
3. Задача проверки кода не запускается, если сообщение в коммите содержит одну из следующих конструкций: `[no_review]`, `[skip_review]`, `[no review]`, `[skip review]`, `[no-review]`, `[skip-review]`,

`[mute]`, `:no_review:`, `:skip_review:`, `:no review:`, `:skip review:`, `:no-review:`, `:skip-review:`, `:mute:`

Настройка

ПАРАМЕТРЫ ДОСТУПА

Параметры доступа (к BeeCR из GitLab и к GitLab из BeeCR) могут быть настроены с помощью переменных среды. Следующие переменные должны быть определены:

- `BEECR_API_KEY` – API-ключ, предоставленный вам командой BeeCR (необходим только при использовании BeeCR SaaS).
- `BEECR_GITLAB_TOKEN` – токен доступа к GitLab, который вы должны выпустить для проекта или группы проектов). Токен должен иметь права как минимум уровня "Developer" и предоставлять доступ к "api".

Совет: GitLab предоставляет два альтернативных способа установки переменных:

- Доопределение CI/CD задачи (т.н. deep merging). Конфигурацию включенной задачи можно осуществить, объявив локальную задачу с тем же именем и переопределив параметры.
- Задание переменных CI/CD в настройках GitLab для проекта, группы проектов или даже экземпляра GitLab (если вы используете GitLab, развернутый на ваших серверах).

Мы рекомендуем устанавливать переменные, содержащие конфиденциальную информацию (API-ключи, токены и т. д.) через настройки GitLab и включать для них режим "маскирования", чтобы предотвратить утечку.

ПРОЧИЕ НАСТРОЙКИ

Список параметров CI/CD компонента:

- `url` – определяет URL API сервера BeeCR.
  Значение по умолчанию: `https://beecr.io/api/v1`.
- `stage` – определяет этап CI/CD пайплайна, на котором будет выполнена задача.
  Значение по умолчанию: `review`.
- `when` – определяет, будет ли задача выполнена вручную или автоматически.
  Используйте `manual`, чтобы запуск задачи проверки кода выполнялся вручную.
  Значение по умолчанию: `always`.
- `target` – задаёт регулярное выражение, которое определяет файлы, подлежащие проверке.
  Значение по умолчанию: `\.(py|c|h|cpp|hpp|cs|java|kt|swift|php|go|sh|(j|t)sx?)$`. Данное регулярное выражение позволяет проводить проверку для файлов на следующих языках:
  - Python
  - C/C++
  - C#
  - Java
  - Kotlin
  - Swift
  - PHP
  - Go
  - Bash/Shell
  - JavaScript/Typescript
- `target-extra` – позволяет задать дополнительное регулярное выражение. Полезно, например, в тех случаях, когда вы не хотите переопределять `target`, а лишь "дополнить" его.
  Значение по умолчанию: пустая строка.
- `target-exclude` – позволяет задать регулярное выражение для исключения файлов из проверки (даже если они соответствуют `target` или `target-extra`). Полезно, например, если вы не хотите переопределять `target`.
  Значение по умолчанию: пустая строка.
- `language` – определяет язык проверки.
  Значение по умолчанию: `English`.
  Мы рекомендуем использовать английский язык, так как в этому случае комментарии от ИИ модели наиболее полные и качественные для всех протестированных моделей.

Пример настройки:

```
stages:
  - review

include:
  - component: gitlab.com/cvisionlab/beecr/beecr@1.4.2
    inputs:
      target-extra: '\.json$'
```

Ручное выполнение

По тем или иным причинам вы можете захотеть контролировать выполнение задачи проверки кода вручную. Для этих целей просто установите значение `manual` для параметра `when`:

```
stages:
  - review

include:
  - component: gitlab.com/cvisionlab/beecr/beecr@1.4.2
    inputs:
      when: manual
```

Закрепление версии компонента

Во всех приведенных выше примерах ссылка на CI/CD компонент содержит явное указание его версии. Если вы хотите всегда ссылаться на последнюю актуальную версию, просто используйте `main` в URL-адресе компонента вместо конкретного тега версии. Пример:

```
stages:
  - review

include:
  - component: gitlab.com/cvisionlab/beecr/beecr@main
```

2.2.2 Пошаговое руководство по интеграции BeeCR через CI/CD компонент (для GitLab 17+)

Данное руководство поможет вам пройти процесс настройки BeeCR в проекте в современном GitLab 17+ через CI/CD компонент.

Примечание: В данном руководстве описано, как настроить проект в современном GitLab 17+ на конкретном примере. Если вам нужна обзорная документация по CI/CD компоненту (GitLab 17+), обратитесь к этому документу. Если же вам нужна обзорная документация по шаблону CI/CD задач (работает как в старых версиях GitLab, так и в более современных GitLab 17+), то обратитесь к этому документу.

Требования

1. Вы получили доступ к BeeCR SaaS или развернули BeeCR на собственном сервере.
2. Вы обладаете достаточным уровнем прав для настройки проекта в GitLab и создание токенов доступа.

1 Создайте токен доступа к GitLab

Сервер BeeCR должен иметь доступ к вашему репозиторию для проведения проверки кода. GitLab предоставляет несколько типов токенов:

- проектный
- групповой
- личный

Мы рекомендуем вам создать токен проекта или группы. Групповой токен более удобен, если вы собираетесь использовать BeeCR в нескольких проектах в одной группе.

Выполните следующие действия:

1. Чтобы создать токен, перейдите в "Настройки" проекта или группы, затем в "Токены доступа" (Access Tokens) и создайте новый токен.
2. Токен должен иметь права как минимум уровня "Developer" и предоставлять доступ к "api".
3. Присвойте токену осмысленное имя, например, "BeeCR Code Reviewer", поскольку комментарии к коду будут написаны ботом, представляющимся этим именем.

2 Убедитесь, что на проекте доступен CI/CD

Пропустите этот шаг, если вы знакомы с GitLab CI/CD и уверены, что он доступен на вашем проекте. В противном случае, рекомендуем убедиться, что CI/CD включен, и что хотя бы один CI/CD сервер (GitLab Runner) доступен в настройках вашего проекта:

1. Перейдите в "Settings" (настройки проекта), затем в "General", и прокрутите до "Visibility, project features, permissions". Опция "CI/CD" должна быть включена.
2. Перейдите в "Настройки" проекта, затем в "CI/CD", и разверните раздел "Runners". Убедитесь, что хотя бы один CI/CD сервер (проектный, групповой или общесистемный) доступен.

Если на вашем проекте не доступен CI/CD, обратитесь к вашему администратору GitLab, системному администратору или команде DevOps. Также вы можете рассмотреть альтернативный метод интеграции через "Webhook", не требующий CI/CD.

3 Настройте основные параметры

Определите следующие переменные CI/CD в вашем проекте или группе (или даже общесистемно для всего GitLab сервера, если используемый вами GitLab развернут на ваших серверах и вы имеете соответствующие права доступа):

- `BEECR_API_KEY` – API-ключ, предоставленный вам командой BeeCR (необходим только при использовании BeeCR SaaS).
- `BEECR_GITLAB_TOKEN` – токен доступа к GitLab, созданный на одном из предыдущих этапов.

4 Подключите CI/CD компонент

1. Создайте (или измените, если уже существует) конфигурационный файл `.gitlab-ci.yml` в корне вашего репозитория. Добавьте этап проверки кода `review` в список CI/CD этапов (`stages`). Подключите CI/CD компонент BeeCR с помощью ключевого слова `include`.

```yaml
stages:
  - review

include:
  - component: gitlab.com/cvisionlab/beecr/beecr@main
```

Совет: Если вы используете GitLab, развернутый на ваших собственных серверах и подключение CI/CD компонента с внешнего репозитория не разрешено политикой вашей организации, вы можете склонировать наш репозиторий в ваш GitLab или же просто скопировать и вставить в вашу конфигурацию содержимое YML файла.

5 Сконфигурируйте CI/CD компонент

Настройки по умолчанию, установленные в предоставленном компоненте CI/CD, удовлетворяют потребностям большинства пользователей. Однако вы можете дополнительно захотеть настроить некоторые параметры.

Ниже приведены несколько примеров. Для получения дополнительной информации о доступных опциях обратитесь к соответствующей документации для CI/CD компонента.

АДРЕС API СЕРВЕРА

Если вы используете API сервер BeeCR, развернутый на ваших собственных серверах, то вам необходимо задать его адрес. Для этого просто установите опцию `url` в актуальное значение (например, `http://192.168.1.2:8000`).

Пример конфигурации:

```yaml
stages:
  - review

include:
  - component: gitlab.com/cvisionlab/beecr/beecr@main
    inputs:
      url: 'http://192.168.1.2:8000'
```

СПИСОК ФАЙЛОВ ДЛЯ ПРОВЕРКИ

Еще один пример - настройка списка расширений файлов для проверки. По умолчанию проверка включена для файлов со следующими расширениями:

- Python: `.py`
- C/C++: `.h`, `.hpp`, `.c`, `.cpp`
- C#: `.cs`
- Java: `.java`
- Kotlin: `.kr`
- Swift: `.swift`
- PHP: `.php`
- Go: `.go`
- Bash/Shell: `.sh`
- JavaScript/Typescript: `.js`, `.jsx`, `.mjs`, `.mjsx`, `.ts`, `.tsx`

У вас есть несколько вариантов для модификации этого списка. Вот некоторые сценарии и предложенные методы:

- Если вы хотите значительно сократить список поддерживаемых расширений (например, оставить только `.java`), то определите опцию `target` как `\.java$`.
- Если вы удовлетворены списком по умолчанию, но хотите добавить еще одно или несколько расширений, например, `.md`, то наиболее удобным способом будет определить опцию `target-extra` как `\.md$`.
- Если вы хотите исключить некоторые расширения из списка по умолчанию, например, `.sh`, то мы рекомендуем определить опцию `target-exclude` как `\.sh$`

Более подробно о том, как управлять списком файлов, подлежащих проверке, читайте в разделе документации с обзором CI/CD компонента

Пример:

```
stages:
  - review

include:
  - component: gitlab.com/cvisionlab/beecr/beecr@main
    inputs:
      target-exclude: '\.sh$'
```

Заключение

Из данного руководства вы узнали, как настроить проект в современном GitLab 17+ для интеграции с BeeCR, создав токены доступа, подключив CI/CD компонент и настроив основные параметры. Следуя этим шагам, вы подготовили ваш GitLab проект к процессам проверки кода с помощью BeeCR. Если вам нужна дополнительная помощь или дополнительная информация, не стесняйтесь обращаться в поддержку BeeCR.

2.3 Через шаблон CI/CD (для более старых версий GitLab)

2.3.1 CI/CD шаблон BeeCR (для более старых версий GitLab)

> **Примечание:** Данный раздел описывает настройку проверки кода в проекте на GitLab через использование шаблона GitLab CI/CD. Если вы хотите настроить проверку кода с помощью компонента GitLab CI/CD (поддерживается в GitLab 17), обратитесь к документу "CI/CD компонент BeeCR". Если же вы предпочитаете вовсе не использовать CI/CD, то обратитесь к альтернативному методу интеграции через "Webhook".

Краткая инструкция

1. Определите следующие переменные CI/CD в вашем проекте (или группе проектов):

- `BEECR_API_KEY` – API-ключ, предоставленным вам командой BeeCR (необходим только при использовании BeeCR SaaS).
- `BEECR_GITLAB_TOKEN` – токен доступа к GitLab, который вы должны выпустить для проекта или группы проектов). Токен должен иметь права как минимум уровня "Developer" и предоставлять доступ к "api".

> **Совет:** Мы рекомендуем присвоить токену осмысленное имя, например, BeeCR Code Reviewer, поскольку комментарии к коду будут написаны ботом, представляющимся этим именем.

2. В настройках вашего проекта убедитесь, что CI/CD включен и доступен хотя бы один сервер запуска задач (GitLab runner).
3. Создайте (или измените, если уже существует) конфигурационный файл `.gitlab-ci.yml` в корне вашего репозитория. Добавьте этап проверки кода в список CI/CD этапов (stages). Подключите CI/CD шаблон BeeCR с помощью ключевого слова `include`.

```yaml
stages:
  - review

include:
  - remote: 'https://gitlab.com/cvisionlab/beecr/-/raw/1.4.2/templates/beecr/template-always.yml'
```

Особенности

1. Задача запускается только для запросов на слияние кода (Merge Request).
2. Результаты проверки добавляются в виде комментариев к файлам.
3. Задача проверки кода не запускается, если сообщение в коммите содержит одну из следующих конструкций: `[no_review]`, `[skip_review]`, `[no review]`, `[skip review]`, `[no-review]`, `[skip-review]`, `[mute]`, `:no_review:`, `:skip_review:`, `:no review:`, `:skip review:`, `:no-review:`, `:skip-review:`, `:mute:`

Настройка

ПАРАМЕТРЫ ДОСТУПА

Параметры доступа (к BeeCR из GitLab и к GitLab из BeeCR) могут быть настроены с помощью переменных среды. Следующие переменные должны быть определены:

- `BEECR_API_KEY` – API-ключ, предоставленным вам командой BeeCR (необходим только при использовании BeeCR SaaS).
- `BEECR_GITLAB_TOKEN` – токен доступа к GitLab, который вы должны выпустить для проекта или группы проектов). Токен должен иметь права как минимум уровня "Developer" и предоставлять доступ к "api".

Совет: GitLab предоставляет два альтернативных способа установки переменных:

- Доопределение CI/CD задачи (т.н. deep merging). Конфигурацию включенной задачи можно осуществить, объявив локальную задачу с тем же именем и переопределив параметры.
- Задание переменных CI/CD в настройках GitLab для проекта, группы проектов или даже экземпляра GitLab (если вы используете GitLab, развернутый на ваших серверах).

Мы рекомендуем устанавливать переменные, содержащие конфиденциальную информацию (API-ключи, токены и т. д.) через настройки GitLab и включать для них режим "маскирования", чтобы предотвратить утечку.

ПРОЧИЕ НАСТРОЙКИ

Список прочих параметров CI/CD шаблона, также определяемых через переменные:

- `BEECR_URL` – определяет URL API сервера BeeCR.
  Значение по умолчанию: `https://beecr.io/api/v1`.
- `BEECR_TARGET` – задаёт регулярное выражение, которое определяет файлы, подлежащие проверке.
  Значение по умолчанию: `\.(py|c|h|cpp|hpp|cs|java|kt|swift|php|go|sh|(j|t)sx?)$`. Данное регулярное выражение позволяет проводить проверку для файлов на следующих языках:
  - Python
  - C/C++
  - C#
  - Java
  - Kotlin
  - Swift
  - PHP
  - Go
  - Bash/Shell
  - JavaScript/Typescript
- `BEECR_TARGET_EXTRA` – позволяет задать дополнительное регулярное выражение. Полезно, например, в тех случаях, когда вы не хотите переопределять `BEECR_TARGET`, а лишь "дополнить" его.
  Значение по умолчанию: пустая строка.
- `BEECR_TARGET_EXCLUDE` – позволяет задать регулярное выражение для исключения файлов из проверки (даже если они соответствуют `BEECR_TARGET` или `BEECR_TARGET_EXTRA`). Полезно, например, если вы не хотите переопределять `BEECR_TARGET`.
  Значение по умолчанию: пустая строка.
- `BEECR_LANGUAGE` – определяет язык проверки.
  Значение по умолчанию: `English`.
  Мы рекомендуем использовать английский язык, так как в этому случае комментарии от ИИ модели наиболее полные и качественные для всех протестированных моделей.

Пример настройки:

Пример настройки через доопределение CI/CD задачи

```yaml
stages:
  - review

include:
  - remote: 'https://gitlab.com/cvisionlab/beecr/-/raw/1.4.2/templates/beecr/template-always.yml'
```

```
beecr:
  variables:
    BEECR_TARGET_EXTRA: '\.json$'
```

Ручное выполнение

По тем или иным причинам вы можете захотеть контролировать выполнение задачи проверки кода вручную. Для этих целей просто используйте значение шаблон `template-manual.yml` вместо `template-always.yml`:

```
stages:
  - review

include:
  - remote: 'https://gitlab.com/cvisionlab/beecr/-/raw/1.4.2/templates/beecr/template-manual.yml'
```

Закрепление версии шаблона

Во всех приведенных выше примерах ссылка на шаблон CI/CD содержит явное указание его версии. Если вы хотите всегда ссылаться на последнюю актуальную версию, просто используйте `main` в URL-адресе шаблона вместо конкретного тега версии. Пример:

```
stages:
  - review

include:
  - remote: 'https://gitlab.com/cvisionlab/beecr/-/raw/main/templates/beecr/template-always.yml'
```

2.3.2 Пошаговое руководство по интеграции BeeCR через CI/CD шаблон (для более старых версий GitLab)

Данное руководство поможет вам пройти процесс настройки BeeCR в проекте в GitLab через CI/CD шаблон. Руководство актуально как для современных версий GitLab, так и для более старых.

Примечание: В данном руководстве описано, как настроить проект в GitLab через CI/CD шаблон на конкретном примере. Если вам нужна обзорная документация по CI/CD шаблону, обратитесь к этому документу. Если же вам нужна общая документация по CI/CD компоненту (работает только современных версиях GitLab 17+), то обратитесь к этому документу.

Требования

1. Вы получили доступ к BeeCR SaaS или развернули BeeCR на собственном сервере.
2. Вы обладаете достаточным уровнем прав для настройки проекта в GitLab и создание токенов доступа.

1 Создайте токен доступа к GitLab

Сервер BeeCR должен доступ к вашему репозиторию для проведения проверки кода. GitLab предоставляет несколько типов токенов:

- проектный
- групповой
- личный

Мы рекомендуем вам создать токен проекта или группы. Групповой токен более удобен, если вы собираетесь использовать BeeCR в нескольких проектах в одной группе.

Выполните следующие действия:

1. Чтобы создать токен, перейдите в "Настройки" проекта или группы, затем в "Токены доступа" (Access Tokens) и создайте новый токен.
2. Токен должен иметь права как минимум уровня "Developer" и предоставлять доступ к "api".
3. Дайте токену осмысленное имя, например, "BeeCR Code Reviewer", поскольку комментарии к коду будут написаны ботом, представляющимся этим именем.

2 Убедитесь, что на проекте доступен CI/CD

Пропустите этот шаг, если вы знакомы с GitLab CI/CD и уверены, что он доступен на вашем проекте. В противном случае, рекомендуем убедиться, что CI/CD включен, и что хотя бы один CI/CD сервер (GitLab Runner) доступен в настройках вашего проекта:

1. Перейдите в "Settings" (настройки проекта), затем в "General", и прокрутите до "Visibility, project features, permissions". Опция "CI/CD" должна быть включена.
2. Перейдите в "Настройки" проекта, затем в "CI/CD", и разверните раздел "Runners". Убедитесь, что хотя бы один CI/CD сервер (проектный, групповой или общесистемный) доступен.

   Если на вашем проекте не доступен CI/CD, обратитесь к вашему администратору GitLab, системному администратору или команде DevOps. Также вы можете рассмотреть альтернативный метод интеграции через "Webhook", не требующий CI/CD.

3 Настройте основные параметры

Определите следующие переменные CI/CD в вашем проекте или группе (или даже общесистемно для всего GitLab сервера, если используемый вами GitLab развернут на ваших серверах и вы имеете соответствующие права доступа):

- `BEECR_API_KEY` – API-ключ, предоставленным вам командой BeeCR (необходим только при использовании BeeCR SaaS).
- `BEECR_GITLAB_TOKEN` – токеном доступа к GitLab, созданным на одном из предыдущих этапов.

4 Подключите CI/CD шаблон

1. Создайте (или измените, если уже существует) конфигурационный файл `.gitlab-ci.yml` в корне вашего репозитория. Добавьте этап проверки кода `review` в список CI/CD этапов (`stages`). Подключите CI/CD шаблон BeeCR с помощью ключевого слова `include`.

```yaml
stages:
  - review
```

```
include:
  - remote: 'https://gitlab.com/cvisionlab/beecr/-/raw/main/templates/beecr/template-always.yml'
```

Совет: Если вы используете GitLab, развернутый на ваших собственных серверах и подключение CI/CD шаблонов с внешнего репозитория не разрешено политикой вашей организации, вы можете склонировать наш репозиторий в ваш GitLab или же просто скопировать и вставить в вашу конфигурацию содержимое YML файла.

Примечание: По тем или иным причинам вы можете захотеть контролировать выполнение задачи проверки кода вручную. Для этих целей просто подключите шаблон `template-manual.yml` вместо `template-always.yml`.

5 Сконфигурируйте CI/CD шаблон

Настройки по умолчанию, установленные в предоставленном CI/CD шаблоне, удовлетворяют потребностям большинства пользователей. Однако вы можете дополнительно захотеть настроить некоторые параметры (через переменные среды).

GitLab предоставляет два альтернативных способа установки переменных:

- Доопределение CI/CD задачи (т.н. deep merging). Конфигурацию включенной задачи можно осуществить, объявив локальную задачу с тем же именем и переопределив параметры.
- Задание переменных CI/CD в настройках GitLab для проекта, группы проектов или даже экземпляра GitLab (если вы используете GitLab, развернутый на ваших серверах).

Ниже приведены несколько примеров. Для получения дополнительной информации о доступных опциях обратитесь к соответствующей документации для CI/CD шаблона.

АДРЕС API СЕРВЕРА

Если вы используете API сервер BeeCR, развернутый на ваших собственных серверах, то вам необходимо задать его адрес. Для этого просто установите переменную `BEECR_URL` в актуальное значение (например, `http://192.168.1.2:8000`).

Пример:

```
stages:
  - review

include:
  - remote: 'https://gitlab.com/cvisionlab/beecr/-/raw/main/templates/beecr/template-always.yml'

beecr:
  variables:
    BEECR_URL: 'http://192.168.1.2:8000'
```

СПИСОК ФАЙЛОВ ДЛЯ ПРОВЕРКИ

Еще один пример - настройка списка расширений файлов для проверки. По умолчанию проверка включена для файлов со следующими расширениями:

- Python: `.py`
- C/C++: `.h`, `.hpp`, `.c`, `.cpp`
- C#: `.cs`
- Java: `.java`
- Kotlin: `.kr`
- Swift: `.swift`
- PHP: `.php`
- Go: `.go`
- Bash/Shell: `.sh`
- JavaScript/Typescript: `.js`, `.jsx`, `.mjs`, `.mjsx`, `.ts`, `.tsx`
- Если вы хотите значительно сократить список поддерживаемых расширений (например, оставить только `.java`), то определите опцию `BEECR_TARGET` как `\.java$`.
- Если вы удовлетворены списком по умолчанию, но хотите добавить еще одно или несколько расширений, например, `.md`, то наиболее удобным способом будет определить опцию `BEECR_TARGET_EXTRA` как `\.md$`.
- Если вы хотите исключить некоторые расширения из списка по умолчанию, например, `.sh`, то мы рекомендуем определить опцию `BEECR_TARGET_EXCLUDE` как `\.sh$`

Более подробно о том, как управлять списком файлов, подлежащих проверке, читайте в разделе документации с обзором CI/CD компонента

Пример:

```yaml
stages:
  - review

include:
  - remote: 'https://gitlab.com/cvisionlab/beecr/-/raw/main/templates/beecr/template-always.yml'

beecr:
  variables:
    BEECR_TARGET_EXCLUDE: '\.sh$'
```

Заключение

Из данного руководства вы узнали, как настроить проект в GitLab для интеграции с BeeCR, создав токены доступа, подключив CI/CD шаблон и настроив основные параметры. Следуя этим шагам, вы подготовили ваш

GitLab проект к процессам проверки кода с помощью BeeCR. Если вам нужна дополнительная помощь или дополнительная информация, не стесняйтесь обращаться к этому учебному пособию или обращаться в поддержку BeeCR.