Пример интеграции Platform SDK в CI/CD GitLab

SDK

Ниже описан процесс интеграции Platform SDK в CI/CD GitLab на примере сборки проекта, созданного в мастере Application SDK.

1. Создать новый пустой проект в GitLab (например, https://git.UrDomain.ru/username/myapp).
2. Создать в мастере Application SDK проект приложения (например, `MyApp`) и загрузить его в GitLab (https://git.UrDomain.ru/username/myapp/-/tree/master/MyApp).
3. В настройках проекта **Settings → General → Visibility, project features, permissions** активировать переключатель **CI/CD**, после чего в левом боковом меню появится пункт **CI/CD**.
4. В настройках проекта **Settings → CI/CD → Runners** в разделе **Shared runners** отключить опцию **Enable shared runners for this project**. Затем из раздела **Specific runners** скопировать токен `registration token` для нашего локального runner'a.
5. Далее необходимо будет создать пользователя для установки в его домашней директории Platform SDK и GitLab runner'a. Предварительно убедимся, что пользователя `gitlab-runner` нет, равно как и его службы:

```
sudo userdel -r -f gitlab-runner
sudo rm /etc/sudoers.d/gitlab-runner
sudo systemctl stop gitlab-runner.service
sudo rm /etc/systemd/system/multi-user.target.wants/gitlab-
  runner.service
sudo rm /etc/systemd/system/gitlab-runner.service
sudo rm -rf /etc/gitlab-runner
sudo rm /usr/local/bin/gitlab-runner
sudo systemctl daemon-reload
```

6. Создать пользователя `gitlab-runner`, установить и запустить в нем runner как службу:

```
# Создать и настроить пользователя GitLab Runner
CI_USER="gitlab-runner"
sudo useradd --comment 'GitLab Runner' --create-home $CI_USER --shell
  /bin/bash
sudo passwd --delete $CI_USER
sudo usermod -aG sudo $CI_USER
echo "$CI_USER ALL=(ALL:ALL) NOPASSWD: ALL" | sudo tee
  /etc/sudoers.d/$CI_USER
sudo rm -f /home/$CI_USER/.bash_logout

# Установить и запустить gitlab-runner как службу
sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-
  runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-
  linux-amd64
sudo chmod +x /usr/local/bin/gitlab-runner
sudo gitlab-runner install --user=$CI_USER --working-
  directory=/home/$CI_USER
sudo gitlab-runner start
```

7. Зарегистрировать runner с помощью токена из п.4:

```
sudo gitlab-runner register --url https://git.omprussia.ru/ --
  registration-token $REGISTRATION_TOKEN
```

После запуска команды нужно подтвердить несколько входных параметров:

- URL, на котором развернут GitLab
- Регистрационный токен

- Описание runner'a (по умолчанию - результат команды `hostname`)
- **Важно**: список тегов runner'a (например, `aurora_psdk_runner`). Этот список должен совпадать со списком тегов в файле `.gitlab-ci.yml`:

```
build-job:
  stage: build
  tags:
    - aurora_psdk_runner
```

- Опциональное примечание для runner'a (можно оставить пустым)
- **Важно**: исполняемый файл для runner'a - следует указать `shell`

После успешной регистрации можно обновить страницу **Settings** → **CI/CD** → **Runners**. В разделе **Available specific runners** появится зарегистрированный runner.

8. Переключиться на пользователя `gitlab-runner`, перейти в домашнюю директорию и установить Platform SDK
9. В проекте GitLab добавить скрипт `build.sh` для сборки:

```
#!/bin/bash

usage() {
    cat <<EOF

Build project in CI

Usage:
   $(basename $0) [OPTION]

Options:
   -n   | --name    <STRING>                project name
   -v   | --version <STRING>                Aurora Platform SDK
   version
   -r   | --release <STRING>                Aurora Platform SDK
   release
   -t   | --target  [i486|armv7hl|aarch64]  select target
   -h   | --help                            this help

EOF

    # exit if any argument is given
    [[ -n "$1" ]] && exit 1


fail() {
    echo ""
    echo "FAIL: $@"
    usage
    exit 1


# handle commandline options
while [[ ${1:-} ]]; do
    case "$1" in
    -n | --name ) shift
        OPT_PROJECT_NAME=$1; shift
        ;;
    -v | --version ) shift
        OPT_PSDK_VERSION=$1; shift
        ;;
    -r | --release ) shift
        OPT_PSDK_DEVEL_RELEASE=$1; shift
        ;;
    -t | --target ) shift
        OPT_TARGET=$1; shift
        ;;
    -h | --help ) shift
        usage quit
        ;;
    * )
        usage quit
        ;;
    esac
done

if [[ -z $OPT_PROJECT_NAME ]]; then
    fail "enter project name"
fi

if [[ -z $OPT_PSDK_VERSION ]] || [[ -z $OPT_PSDK_DEVEL_RELEASE ]];
   then
    fail "incorrect version or release"
fi
```

```
if [[ "$OPT_TARGET" != "i486" ]] && [[ "$OPT_TARGET" != "armv7hl" ]]
   && [[ "$OPT_TARGET" != "aarch64" ]]; then
    fail "incorrect target"
fi

PROJECT_NAME=$OPT_PROJECT_NAME
PSDK_VERSION=$OPT_PSDK_VERSION
PSDK_DEVEL_RELEASE=$OPT_PSDK_DEVEL_RELEASE

PROJECT_DIR="./$PROJECT_NAME"
CI_BUILD_DIR=$PWD

cd $PROJECT_DIR

PSDK_FULL_VERSION=$PSDK_VERSION.$PSDK_DEVEL_RELEASE
PSDK_TARGET=AuroraOS-$PSDK_FULL_VERSION

TARGET=$PSDK_TARGET-$OPT_TARGET

# Build project
echo "Build $PROJECT_NAME project for $TARGET"
mb2 --target $TARGET build

# Copy build artifacts
mkdir -p $CI_BUILD_DIR/artifacts
cp RPMS/* $CI_BUILD_DIR/artifacts/
```

10. В разделе **CI/CD** нажать кнопку **Use Template**, после чего вместо шаблона добавить следующий скрипт:

```
stages:
  - build

build-job:
  stage: build
  tags:
    - aurora_psdk_runner
  artifacts:
    paths:
      - ./artifacts/*.rpm
  script:
    - /home/gitlab-runner/AuroraPlatformSDK/sdks/aurora_psdk/sdk-
    chroot ./build.sh -n $PROJECT -v $VERSION -r $RELEASE -t
    $BUILD_TARGET
```

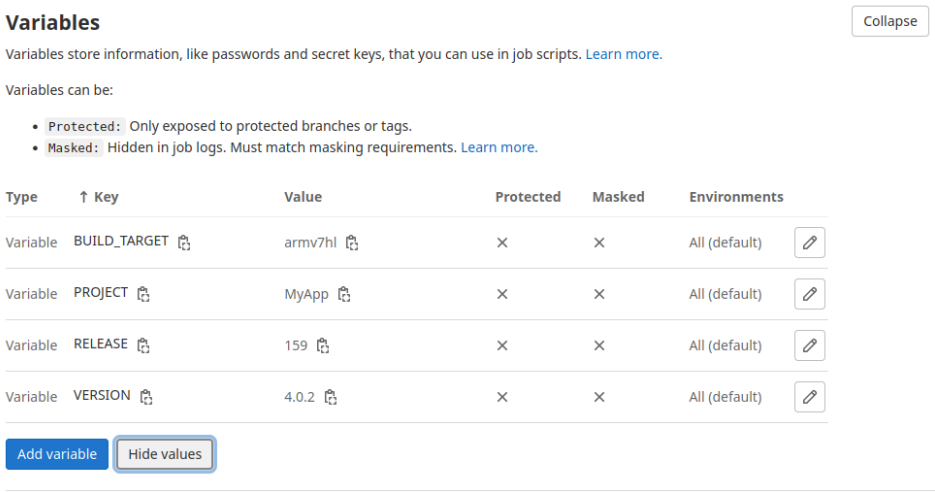
Здесь нужно еще раз убедиться, что в списке `tags` указаны теги, которые были прописаны в ходе регистрации runner'a в п.7.

Также следует убедиться, что корректно указан путь к скрипту запуска Platform SDK (см. п.8).

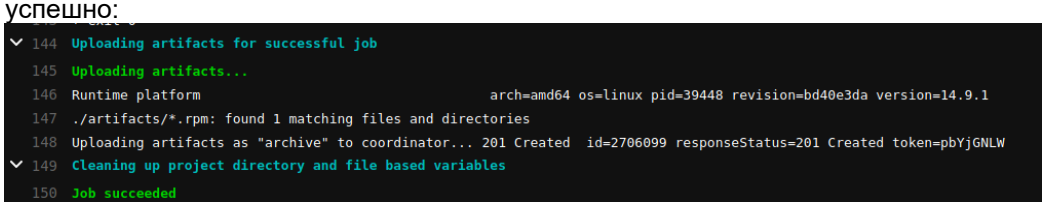
11. Проверить сборку проекта. Для этого в GitLab перейти в раздел **CI/CD**, нажать кнопку **Run pipeline**. Далее необходимо указать переменные, которые будут подаваться на вход скрипту `build.sh`:

    - `PROJECT` - название собираемого проекта (в нашем примере это `MyApp`, см. п.2)
    - `VERSION` - основная версия Platform SDK (например, `4.0.2`)
    - `RELEASE` - номер билда Platform SDK (например, `159`)
    - `BUILD_TARGET` - имя архитектуры таргета (например, `armv7hl`)

Чтобы не вводить каждый раз эти переменные при запуске сборки, их можно добавить в переменные GitLab:



12. Нажать кнопку **Run pipeline**. Если все настроено корректно, сборка проходит успешно:



Артефакт сборки (собранный в Platform SDK rpm-пакет приложения MyApp) будет доступен для скачивания по ссылке **Download** в разделе **Job artifacts**: