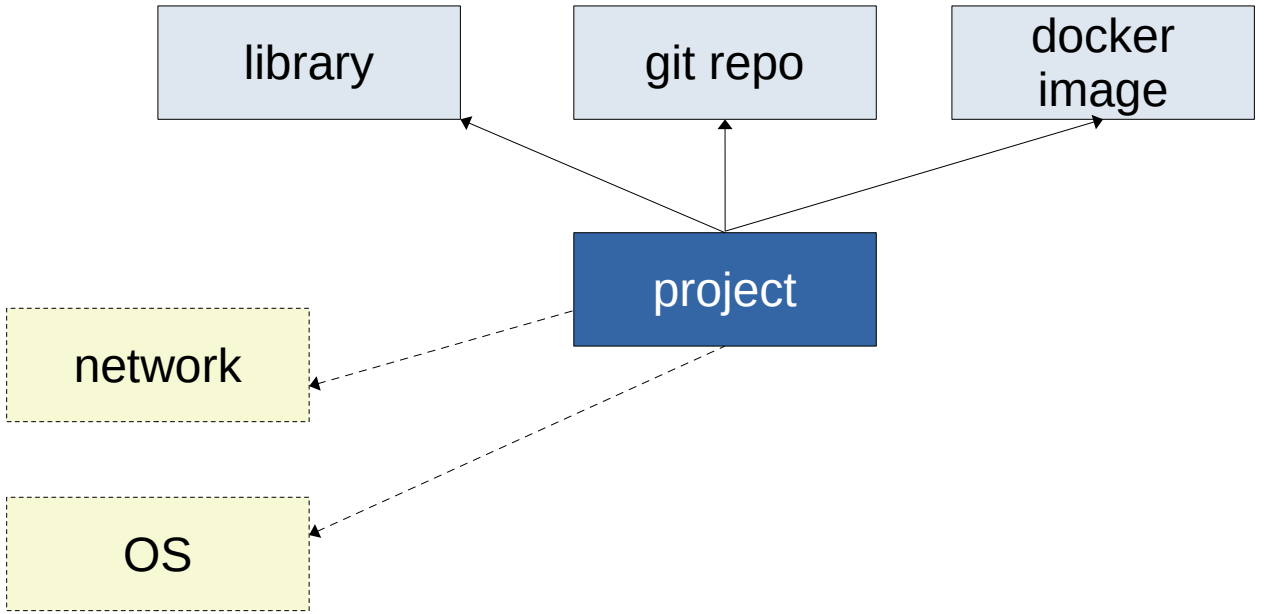
Разработка инфраструктуры программного обеспечения

Пакетный менеджмент

Лаборатория RISC-V технологий,
2025 г.

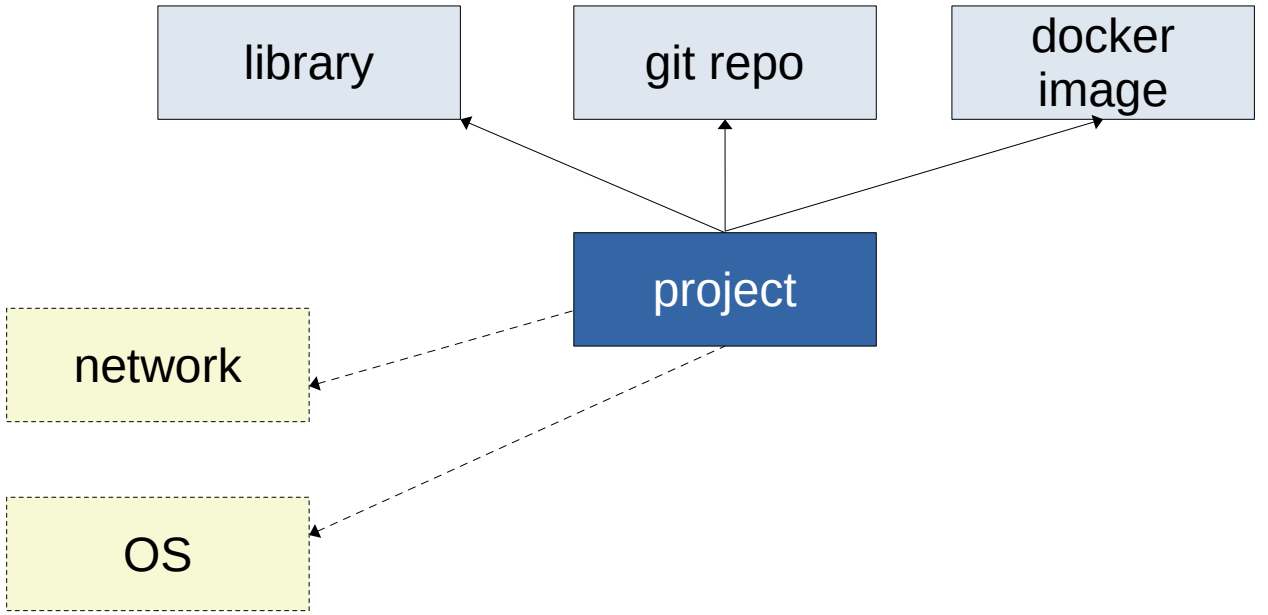
Внешние зависимости

- **Проблема:** внешние зависимости



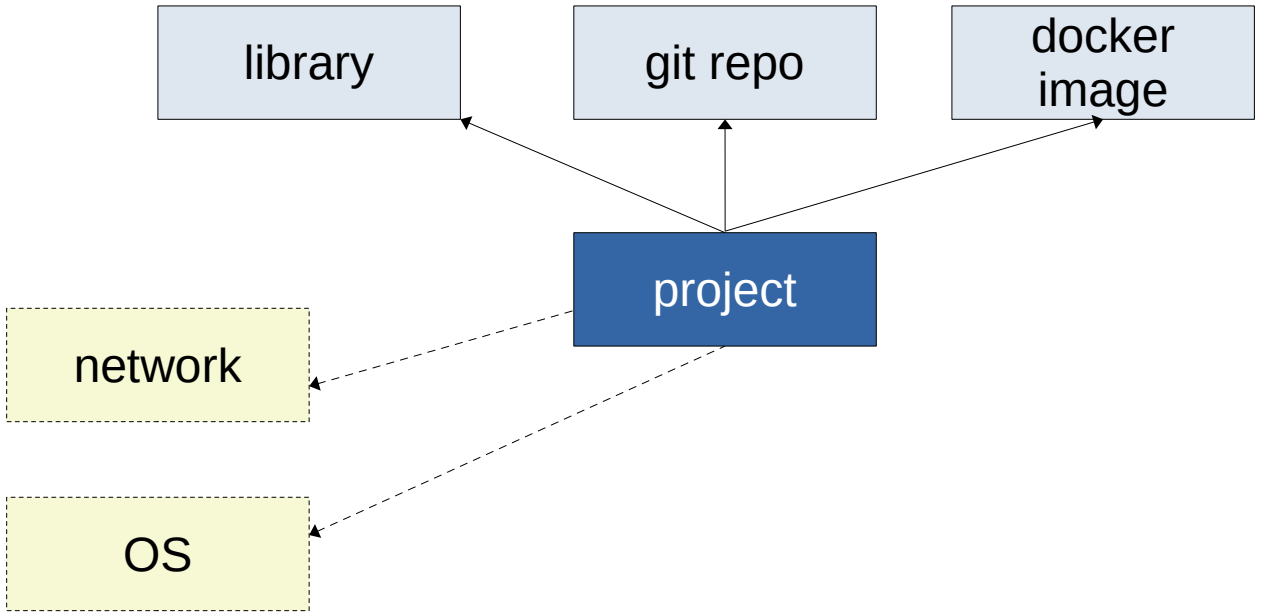
Внешние зависимости

- **Проблема:** внешние зависимости
- **Внешние зависимости** — это такие зависимости вашего проекта, которые не являются частью вашего репозитория



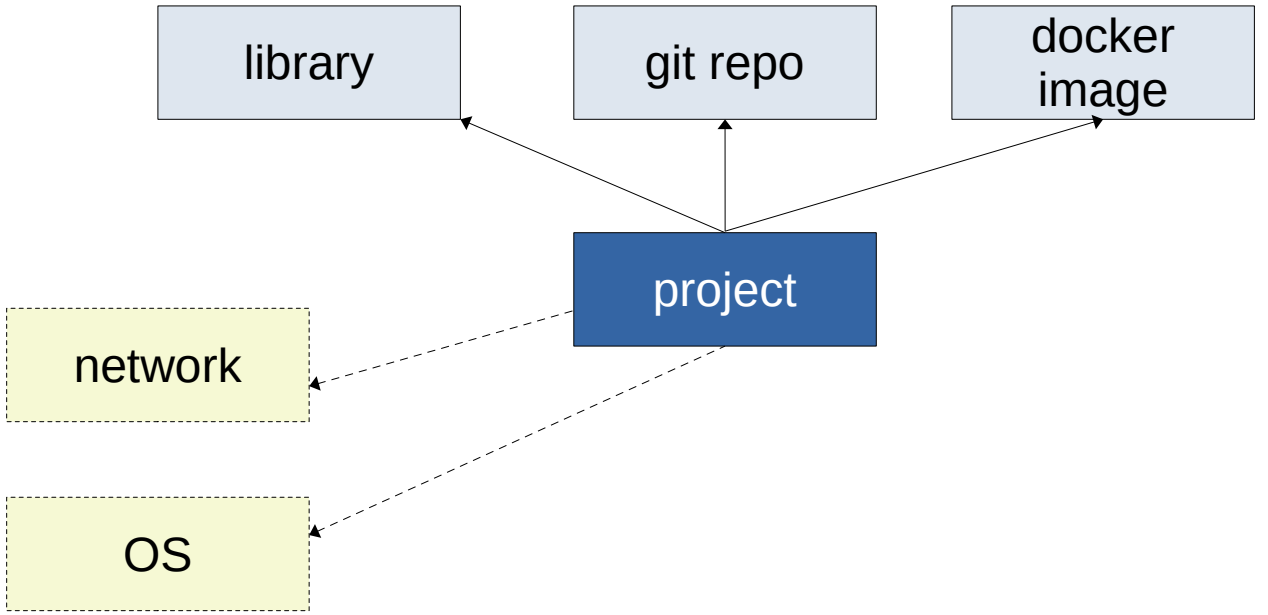
Внешние зависимости

- **Проблема:** внешние зависимости
- **Внешние зависимости** — это такие зависимости вашего проекта, которые не являются частью вашего репозитория
- Внешние зависимости имеют такое же влияние на успешность вашей сборки, тестирования, как и внутренние



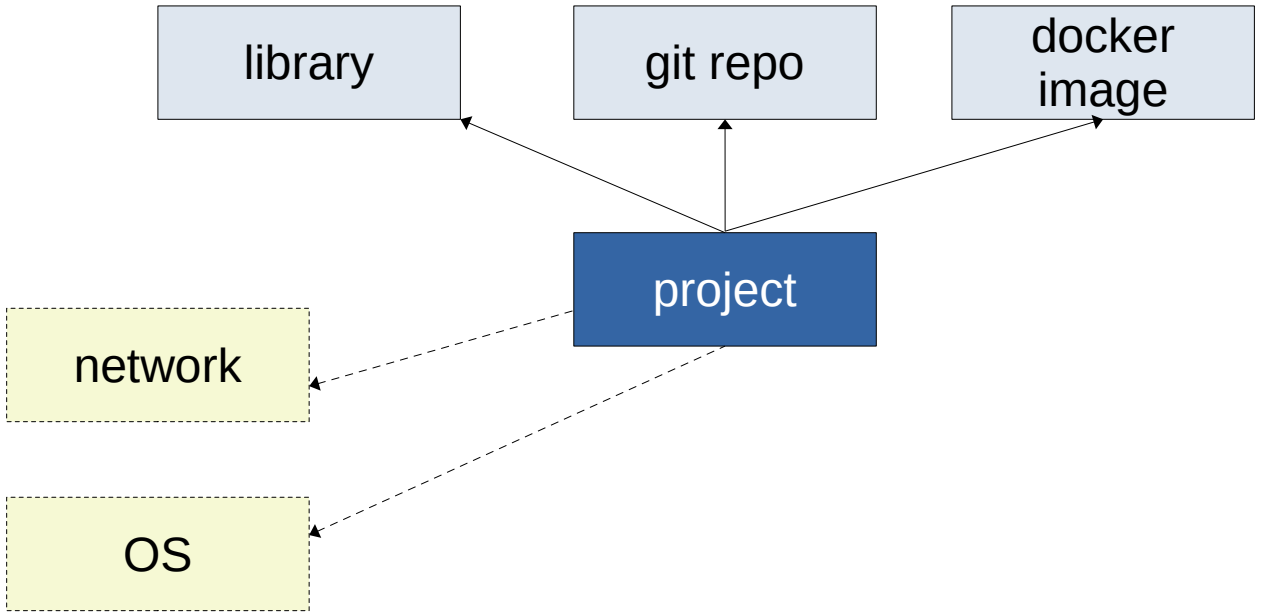
Внешние зависимости

- **Проблема:** внешние зависимости
- **Внешние зависимости** — это такие зависимости вашего проекта, которые не являются частью вашего репозитория
- Внешние зависимости имеют такое же влияние на успешность вашей сборки, тестирования, как и внутренние
- Данные зависимости являются такими же важными, как и любая часть вашего проекта



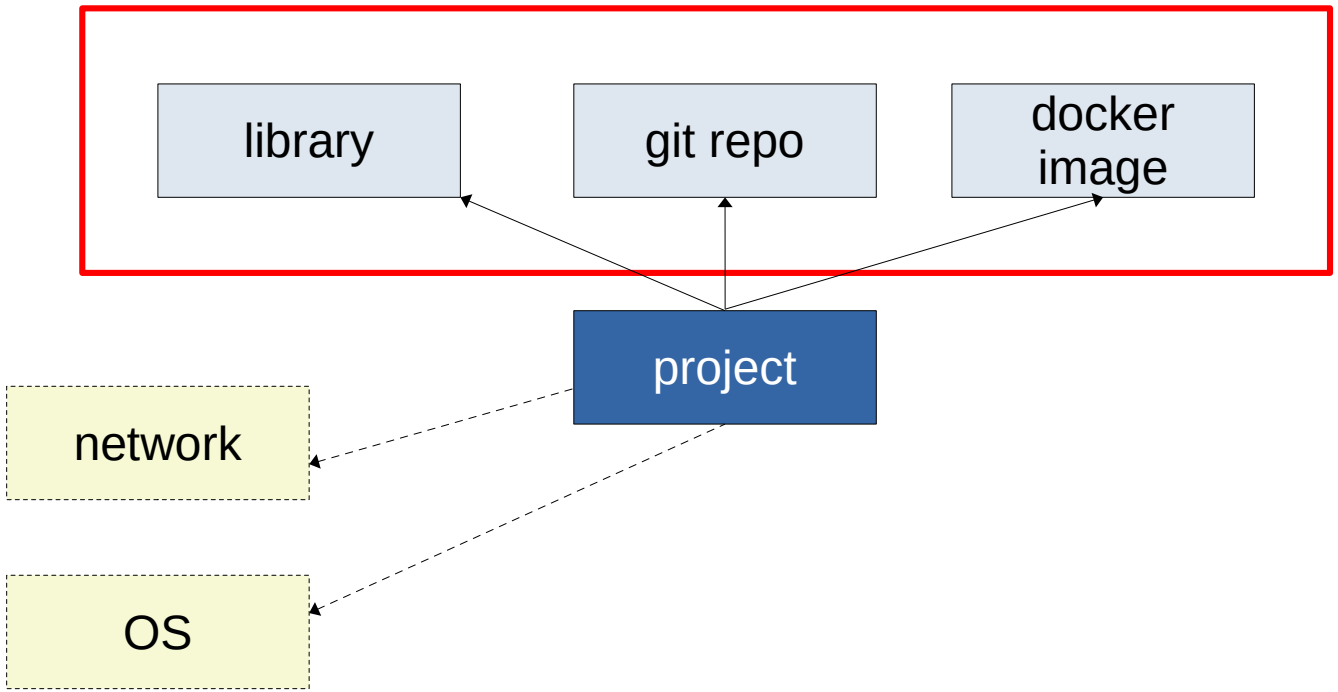
Внешние зависимости

- Внешние зависимости можно разделить на **контролируемые** и **неконтролируемые**



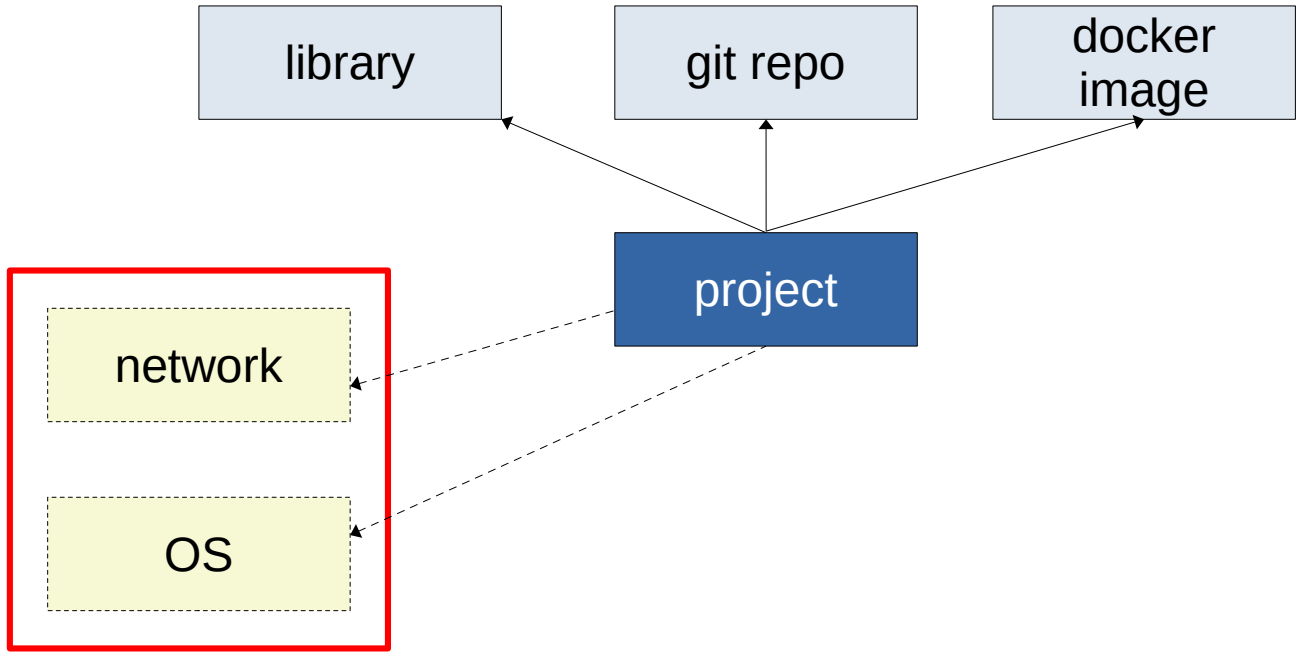
Внешние зависимости

- Внешние зависимости можно разделить на **контролируемые** и **неконтролируемые**
- Библиотека, гит репозиторий, докер образ — контролируемые



Внешние зависимости

- Внешние зависимости можно разделить на **контролируемые** и **неконтролируемые**
- Библиотека, гит репозиторий, докер образ — контролируемые
- Сеть, некоторые свойства операционной системы — неконтролируемые (**инфраструктура**)



Воспроизводимость во времени

- **Проблема:** воспроизводимость состояния проекта во времени

GitHub Workflow (github-workflow.json)

```yaml
name: first workflow

on: [push]

jobs:
  first-job:
    runs-on: ubuntu-latest
    steps:
      - name: Say Hello world
        run: echo Hello world
```

```
# Fetch latest file (Assuming you know its name or use listing)
echo "Fetching latest file from LATEST folder..."
LATEST_FILE=$(curl -u "$USERNAME:$PASSWORD" -s "$ARTIFACTORY_URL/api/storage/$REPO_NAME/$LATEST_FOLDER" |
```

Воспроизводимость во времени

- **Проблема:** воспроизводимость состояния проекта во времени

GitHub Workflow (github-workflow.json)

```yaml
name: first workflow

on: [push]

jobs:
  first-job:
    runs-on: ubuntu-latest
    steps:
      - name: Say Hello world
        run: echo Hello world
```

```
# Fetch latest file (Assuming you know its name or use listing)
echo "Fetching latest file from LATEST folder..."
LATEST_FILE=$(curl -u "$USERNAME:$PASSWORD" -s "$ARTIFACTORY_URL/api/storage/$REPO_NAME/$LATEST_FOLDER" |
```

Воспроизводимость во времени

- **Проблема:** воспроизводимость состояния проекта во времени
- Зачастую гораздо проще указывать latest везде где это можно

GitHub Workflow (github-workflow.json)

```yaml
name: first workflow

on: [push]

jobs:
  first-job:
    runs-on: ubuntu-latest
    steps:
      - name: Say Hello world
        run: echo Hello world
```

```
# Fetch latest file (Assuming you know its name or use listing)
echo "Fetching latest file from LATEST folder..."
LATEST_FILE=$(curl -u "$USERNAME:$PASSWORD" -s "$ARTIFACTORY_URL/api/storage/$REPO_NAME/$LATEST_FOLDER" |
```

Воспроизводимость во времени

- **Проблема:** воспроизводимость состояния проекта во времени
- Зачастую гораздо проще указывать latest везде где это можно
- К чему это приводит?

```
error: 'filesystem' is not a member of 'boost'
```

```
error: 'TEST' was not declared in this scope
```

```
// Example 3: Binary Incompatibility (OpenSSL)
#include <openssl/evp.h>
void useOpenSSL() {
    EVP_PKEY* pkey = EVP_PKEY_new();  // May cause linker errors if OpenSSL updates
    if (pkey) {
        EVP_PKEY_free(pkey);
    }
}
/*
undefined reference to `EVP_PKEY_new'
collect2: error: ld returned 1 exit status
*/
```

Воспроизводимость во времени

- Пока мы еще не говорили про пакетный менеджмент, но поговорили про внешние зависимости

Воспроизводимость во времени

- Пока мы еще не говорили про пакетный менеджмент, но поговорили про внешние зависимости
- Все **версии** внешние зависимости вашего проекта должны быть **точно зафиксированы**

starlark Copy Edit

```
http_archive(
    name = "googletest",
    urls = ["https://github.com/google/googletest/archive/refs/tags/v1.13.0.zip"],
    strip_prefix = "googletest-1.13.0",
)
```

```
# Pin Boost to exact version 1.75.0
find_package(Boost 1.75.0 REQUIRED)

# Use exact version of GoogleTest
FetchContent_Declare(
    googletest
    URL https://github.com/google/googletest/archive/refs/tags/v1.13.0.zip
)
FetchContent_MakeAvailable(googletest)
```

Воспроизводимость во времени

- Пока мы еще не говорили про пакетный менеджмент, но поговорили про внешние зависимости
- Все **версии** внешние зависимости вашего проекта должны быть **точно зафиксированы**
- Способ фиксации может быть разные — фиксированный хеш коммит, фиксированная ссылка на релиз, фиксированный тег и т.д

starlark Copy Edit

```
http_archive(
    name = "googletest",
    urls = ["https://github.com/google/googletest/archive/refs/tags/v1.13.0.zip"],
    strip_prefix = "googletest-1.13.0",
)
```

```
# Pin Boost to exact version 1.75.0
find_package(Boost 1.75.0 REQUIRED)

# Use exact version of GoogleTest
FetchContent_Declare(
    googletest
    URL https://github.com/google/googletest/archive/refs/tags/v1.13.0.zip
)
FetchContent_MakeAvailable(googletest)
```

Воспроизводимость во времени

- Пока мы еще не говорили про пакетный менеджмент, но поговорили про внешние зависимости
- Все **версии** внешние зависимости вашего проекта должны быть **точно зафиксированы**
- Способ фиксации может быть разные — фиксированный хеш коммит, фиксированная ссылка на релиз, фиксированный тег и т.д
- Обновление таких зависимостей должно быть **только через коммит с тестированием**

```
starlark                                                    Copy    Edit

http_archive(
    name = "googletest",
    urls = ["https://github.com/google/googletest/archive/refs/tags/v1.13.0.zip"],
    strip_prefix = "googletest-1.13.0",
)
```

```
# Pin Boost to exact version 1.75.0
find_package(Boost 1.75.0 REQUIRED)

# Use exact version of GoogleTest
FetchContent_Declare(
    googletest
    URL https://github.com/google/googletest/archive/refs/tags/v1.13.0.zip
)
FetchContent_MakeAvailable(googletest)
```

Способы менеджмента внешних зависимостей

- C/C++ библиотеки (**zlib, boost)**

Способы менеджмента внешних зависимостей

- C/C++ библиотеки (**zlib, boost**)
- C/C++ executables (**gcc, clang, qemu**)

Способы менеджмента внешних зависимостей

- C/C++ библиотеки (**zlib, boost**)
- C/C++ executables (**gcc, clang, qemu**)
- Python / Go / JavaScript modules (**black, pylint, pathlib ,conan)**

Способы менеджмента внешних зависимостей

- C/C++ библиотеки (**zlib, boost**)
- C/C++ executables (**gcc, clang, qemu**)
- Python / Go / JavaScript modules (**black, pylint, pathlib ,conan)**
- Docker images

Способы менеджмента внешних зависимостей

- C/C++ библиотеки (**zlib, boost**)
- C/C++ executables (**gcc, clang, qemu**)
- Python / Go / JavaScript modules (**black, pylint, pathlib ,conan)**
- Docker images
- Системы сборки и генераторы систем сборок (**cmake, make, ninja)**

Способы менеджмента внешних зависимостей

- C/C++ библиотеки (**zlib, boost**)
- C/C++ executables (**gcc, clang, qemu**)
- Python / Go / JavaScript modules (**black, pylint, pathlib ,conan)**
- Docker images
- Системы сборки и генераторы систем сборок (**cmake, make, ninja)**
- Гит репозитории

Способы менеджмента внешних зависимостей

- C/C++ библиотеки (**zlib, boost**)
- C/C++ executables (**gcc, clang, qemu**)
- Python / Go / JavaScript modules (**black, pylint, pathlib ,conan)**
- Docker images
- Системы сборки и генераторы систем сборок (**cmake, make, ninja)**
- Гит репозитории
- Всякие скрипты и т.д...

Способы менеджмента внешних зависимостей

Как?

Способы менеджмента внешних зависимостей

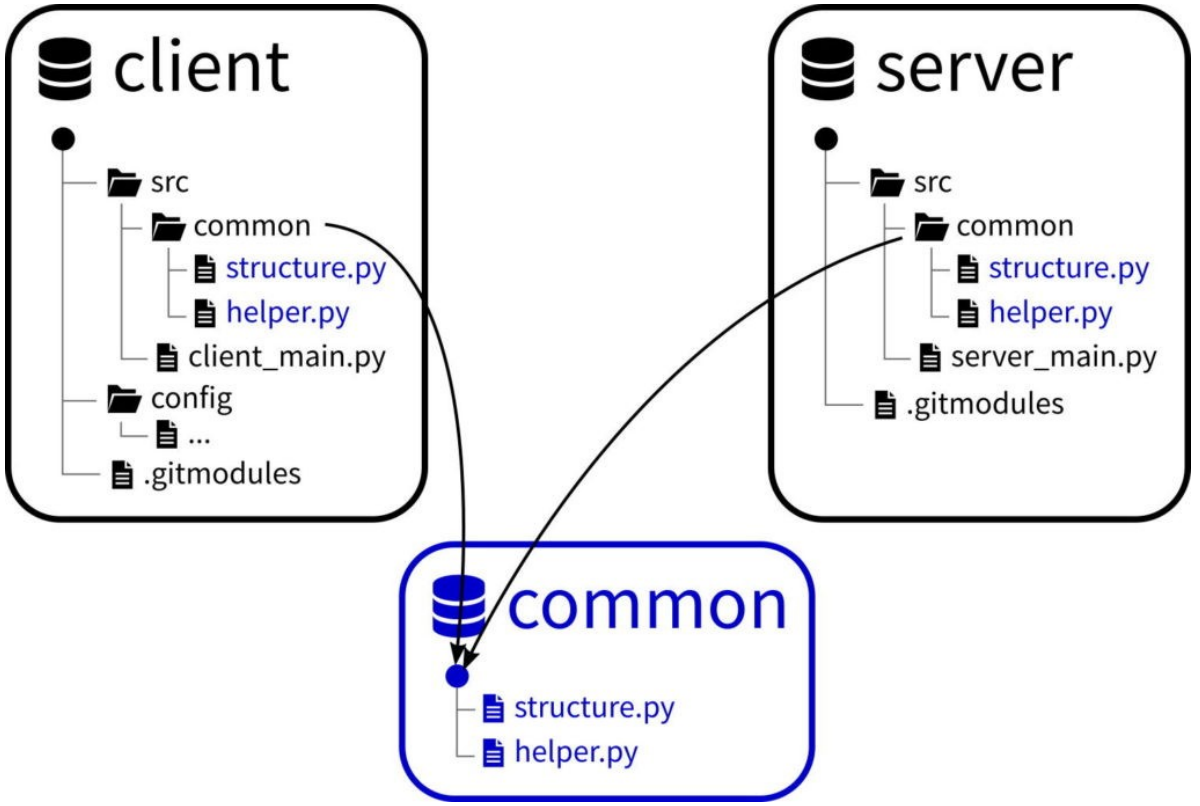
Git submodules

Плюсы:

- Очень легко настроить
- Легко обновлять
- Можно пиннить конкретные версии

Минусы:

- Сабмодуль копируется столько раз, сколько раз встречается в древе зависимостей
- Крайне плохо справляется, когда зависимости представляют собой DAG
- Не масштабируются
- Нет поддержки бинарных артефактов



Способы менеджмента внешних зависимостей

Cmake FetchContent

Плюсы:

- Легко интегрируется в, well, cmake
- Умеет собирать зависимость на лету (только если зависимость сама на cmake)

Минусы:

- Завязывается только на cmake, не масштабируется на другие генераторы
- Не поддерживает бинарные артефакты
- Не может разрешать граф зависимостей, довольно примитивен

```
FetchContent_Declare(
  googletest
  GIT_REPOSITORY https://github.com/google/googletest.git
  GIT_TAG        703bd9caab50b139428cea1aaff9974ebee5742e # release-1.10.0
)
FetchContent_Declare(
  myCompanyIcons
  URL      https://intranet.mycompany.com/assets/iconset_1.12.tar.gz
  URL_HASH MD5=5588a7b18261c20068beabfb4f530b87
)

FetchContent_MakeAvailable(googletest myCompanyIcons)
```

Ромбовидные зависимости

- **Проблема**: зависимость встречается два или более раз в графе зависимостей **с разными версиями**