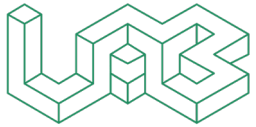
컨트리뷰션 가이드라인

Backend.AI

프로젝트 개발 규칙

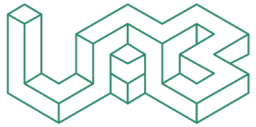
BACKEND.AI

개발



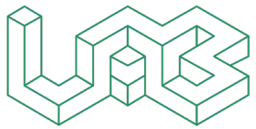
- GitHub 저장소의 README는 아래 내용을 포함합니다:
  - 프로젝트 제목
  - 프로젝트 한 줄 요약
  - 설치 / 실행 / 개발 방법
    - 별도의 문서 사이트를 가리키는 링크도 됩니다.
  - 브렌치 관리 규칙 (없다면 아래 설명할 저희 규칙을 따라주세요)

개발: 브랜치 관리 규칙



- **`master`** 또는 **`main`**: 가장 최신의 개발 버전
- **`X.Y`**: 안정 릴리즈 버전들에 해당
    - CI/CD가 연결된 클라우드 서비스의 경우, **`production`** 브렌치를 사용합니다.
    - 버전 스키마는 프로젝트마다 다르지만, 일반적으로 연.월 방식을 사용합니다. 예) 21.03
- **`X.Y.Z`**: 각 릴리즈에 해당되는 태그
- 다른 규칙을 사용하고 싶은 경우, 저장소의 README에 명기하고 공동작업자들에게 알립니다.
    - e.g., ***`main`*** 을 안정 버전 브렌치로, **`develop`** 을 개발 브렌치로
- v1.0 이전
    - 1-3명정도의 소수의 사람들이 코드를 만듭니다.
    - 바로 main에 커밋해도 됩니다.
- v1.0 이후
    - 아무리 작은 변경 사항도 PR을 만듭니다.

개발: 워크플로우 설명



```
$ git checkout main  # the target base branch
$ git checkout -b feature/improve-something
(Do the work)
$ git add ...
$ git commit -v
$ git push -u origin feature/improve-something
(Create a pull request)
(Take the review feedback and apply it)
$ git commit -av
(Merge the updated base branch if required)
$ git fetch && git merge origin/main
(Add the news fragment for this PR: "changes/XXX.feature")
(Done and merged)      # the merged remote branch is auto-deleted by GitHub
$ git checkout master
$ git pull -p          # update & sync deletion of the remote branches
$ git branch -D feature/improve-something  # remove the local branch
```

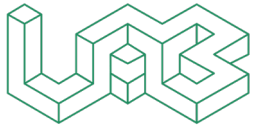
➡ At this point, you will get the PR number XXX.

※ See also: https://cli.github.com

```
$ gh pr create
$ gh pr status
```

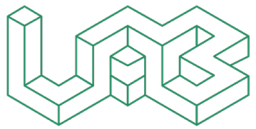
➡ See the next slide.

개발: 작업 기록 관리



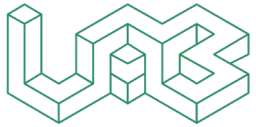
- 체인지로그 자동 작성을 위해 `towncrier` 를 사용하고 있습니다.
  - https://pypi.org/project/towncrier/
  - 프로젝트에 "changes" 디렉토리가 있는경우, 변경사항의 간단 요약이 담긴 파일을 "changes/XXX.suffix" 형태로 함께 추가해주셔야 합니다.
    - XXX : GitHub 이슈 번호 또는 PR 번호
    - the suffix : "pyproject.toml " 에 정의된 값들 중 하나.
  - 작업 기록은 Markdown을 사용할 수 있습니다.
  - PR은 한 줄로 요약하는 것을 추천합니다.

개발: 브랜치 이름 규칙



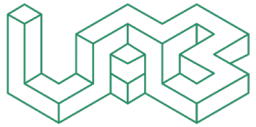
- `feature/short-desc`
  - 다른 방법: `feature/XXX-short-desc` (XXX 는 GitHub 이슈 번호.)
- `fix/short-desc`
- `tests/short-desc & docs/short-desc`
  - 이상적으로는, 얘네들은 feature 랑 bugfix에 같이 들어가야 합니다.
- `hotfix/short-desc`
  - 릴리즈 브랜치에서 긴급한 수정이 필요한 경우
- `backport/XXX` (XXX : main의 PR 번호)
- `maintenance/short-desc`
  - 저장소 관련 변경 사항, CI/CD 관련 변경 사항

개발: 커밋 메세지 규칙



- 브랜치 이름처럼, 커뮤니티에서 널리 사용하는 "conventional commit" 스타일을 사용합니다.
  - https://www.conventionalcommits.org/en/v1.0.0/
- 약간의 변경이 있습니다.
  - 브랜치 A의 특정 커밋을 체리피킹 (cherry-picking) 해서 브랜치 B에 반영할 때, 메세지 뒤에 ”Ported-From: A” 를 붙입니다.
  - 반드시 지켜야 하는 규칙은 아니지만 권장됩니다.
    - PR을 리뷰하는 사람들이 PR을 squash 해서 커밋 메세지를 새로 작성할 때 도움이 됩니다.

개발: PR 요구사항



- 절대적으로 지킬 것: lint 통과, 타입체크, Travis CI, CLA (Contributor License Agreement)
- 꼭 지킬것: codecov 체크 통과.
    - 이상적으로는 모든 코드 변경사항이 있는 PR들은 그에 해당하는 테스트 코드가 있어야 합니다. (변경사항 이전에는 실패하고, 변경 이후에는 통과)
    - 현재는 테스트 슈이트의 폭을 천천히 넓혀 가는 중입니다.

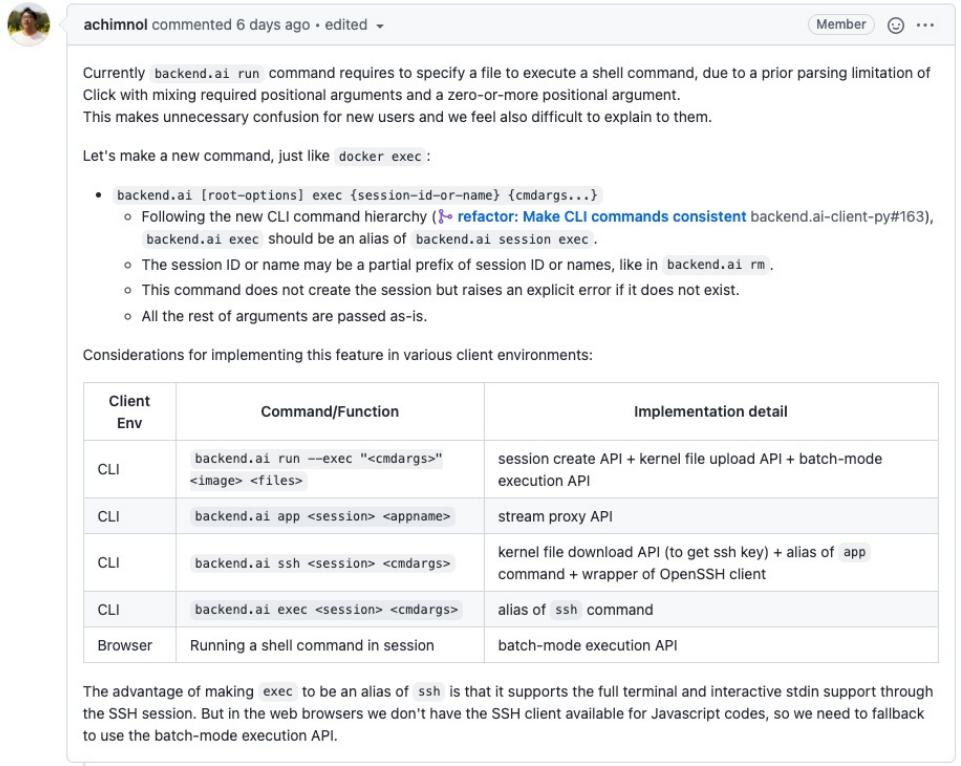
예시로 보는 개발 규칙

좋은 예와 나쁜 예

BACKEND.

이슈 생성시 Description 좋은 예와 나쁜 예

- 좋은 예 👍
  - backend.ai/issues/283 참고



achimnol commented 6 days ago • edited — Member

Currently `backend.ai run` command requires to specify a file to execute a shell command, due to a prior parsing limitation of Click with mixing required positional arguments and a zero-or-more positional argument.
This makes unnecessary confusion for new users and we feel also difficult to explain to them.

Let's make a new command, just like `docker exec`:

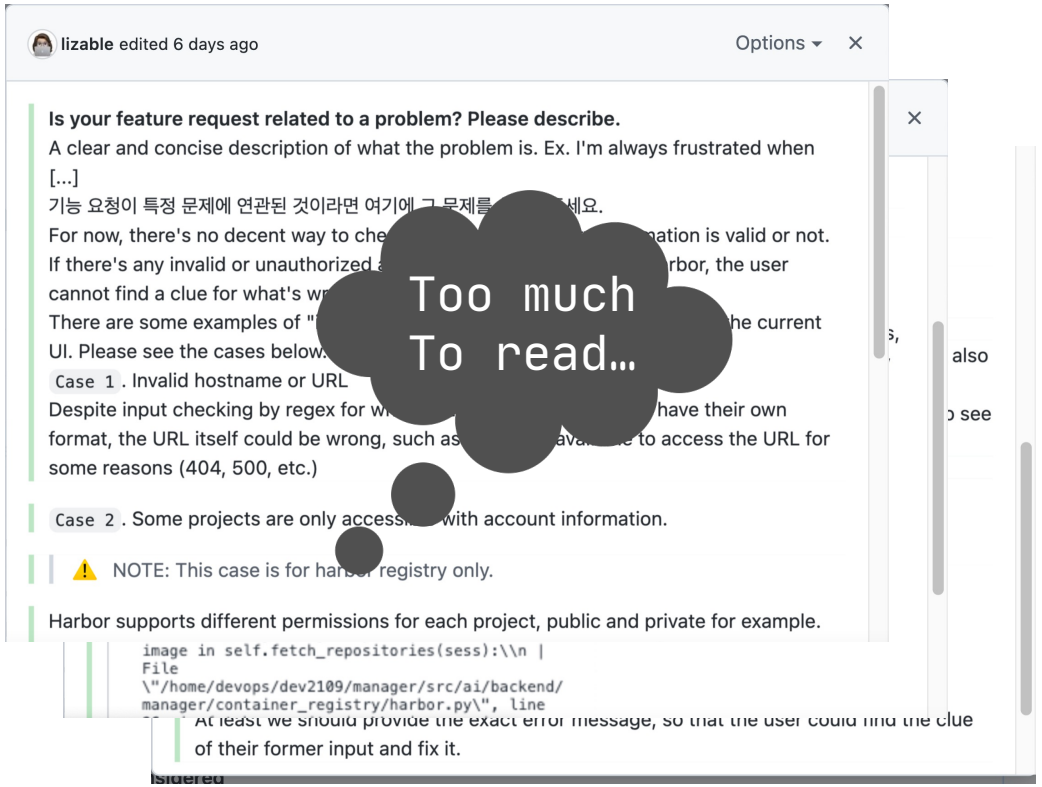
- `backend.ai [root-options] exec {session-id-or-name} {cmdargs...}`
  - Following the new CLI command hierarchy (refactor: Make CLI commands consistent backend.ai-client-py#163), `backend.ai exec` should be an alias of `backend.ai session exec`.
  - The session ID or name may be a partial prefix of session ID or names, like in `backend.ai rm`.
  - This command does not create the session but raises an explicit error if it does not exist.
  - All the rest of arguments are passed as-is.

Considerations for implementing this feature in various client environments:

Client Env	Command/Function	Implementation detail
CLI	`backend.ai run --exec "<cmdargs>" <image> <files>`	session create API + kernel file upload API + batch-mode execution API
CLI	`backend.ai app <session> <appname>`	stream proxy API
CLI	`backend.ai ssh <session> <cmdargs>`	kernel file download API (to get ssh key) + alias of `app` command + wrapper of OpenSSH client
CLI	`backend.ai exec <session> <cmdargs>`	alias of `ssh` command
Browser	Running a shell command in session	batch-mode execution API

The advantage of making `exec` to be an alias of `ssh` is that it supports the full terminal and interactive stdin support through the SSH session. But in the web browsers we don't have the SSH client available for Javascript codes, so we need to fallback to use the batch-mode execution API.

- 나쁜 예 👎
  - backend.ai/issues/284 의 첫번째 revision 참고



상황 설명

- 시나리오:
  - Backend.AI WebUI 독립 이슈
  - 기능/버그? 버그
  - 사용자가 회원가입할 때 마주칠 수 있는 UI 관련 버그임
  - 상세 내용:
    - 서비스 이용정책과 개인정보 이용방침에 대한 안내 다이얼로그가 X(닫기) 버튼을 누른 뒤에 아무리 다시 열기 버튼을 클릭해도 열리지 않는 버그가 발생함.
    - 이슈는 이미 생성되어 있음 (webui#1123)
    - 발생 빈도: 100%
    - 해야할 일:
      1. Fork한 내 저장소에 이슈 브랜치 생성
      2. 이슈 브랜치에 버그를 해결할 코드가 반영된 커밋 올리기
      3. Fork한 내 저장소/이슈 브랜치 → **원래 저장소/main 브랜치**에 PR 올리기

브랜치 이름 짓기의 좋은 예와 나쁜 예

- 좋은 예 👍
  - fix/ToS-PrivacyPolicy-dialog-display-properly
  - fix/ToS-PrivacyPolicy-dialog-display
- 나쁜 예 👎
  - fix-dialog
  - fix-dialog-properly
  - fix/Dialog-view
  - fix/Login-Dialog

커밋 메시지 작성의 좋은 예와 나쁜 예

- 좋은 예 👍
  - 하나의 커밋에 들어갈 코드의 양?
    - 다음과 같은 단위로 커밋 쓰는 것을 권장합니다.
      - 주요 변경 내역(기능이나 버그를 바로 고칠 수 있는 부분)
      - 기타 코스메틱한 부분
        - 오타와 같은 에러

- 나쁜 예 👎
  - 하나의 커밋에 들어갈 코드의 양?
    - 일단 작동이 되는 것을 확인만 하고 git diff로 변경 내역 확인 할 때 스크롤을 풀로 5번 이상 땡겨야 하는 양의 변경 내역을 때려넣음
      ⚠️ 리뷰어가 확인도 어렵고, 나중에 커밋을 작성한 본인 역시 리뷰어의 리뷰 내용을 확인할 때 꽤나 애를 먹습니다.

PR Description 작성의 좋은 예와 나쁜 예

- 좋은 예 👍
  - 잘 설명된 이슈 넘버가 있을 경우:

    This PR resolves <issue #>.
  - 이슈 넘버가 없을 경우:

    Before / After
- 나쁜 예 👎
  - 잘 설명된 이슈 넘버가 있을 경우,
    이슈 넘버가 없을 경우:

    No description to display.

    (a.k.a. Title Says All (제곧내))

PR Description을 잘 쓰는 방법은 아래 링크에서도 자세하게 보실 수 있어요!

👉👉 How to make a proper description for a pull request 👈👈

결론:

내가 작업한 것을 상대방이 **이해를 할 수 있도록** 작성합시다.

너무 성의 없게 쓰는 것보다는 자세하게 쓰는 것이 낫지만,
너무 길게 쓸 경우 상대방은 이슈를 직접 해결하는 시간 보다
그 내용 자체를 이해하는 데에 시간을 더 쓰게 될 수 있습니다.