Ruby on Rails 6,7,8

最近の機能を知って開発者体験を考える

2024-11-19

https://github.com/nishimotz/hello-rails

Rails 6の新機能とテスト戦略

- 機能
  - Action Text: リッチテキスト編集が簡単に
    - テキストエリアのテスト追加
  - Action Mailbox: メールの受信処理機能
  - パラレルテスト（Parallel Testing）の導入
- テスト戦略
  - Action Text：ビューのテストでWYSIWYGエディタの動作を確認
  - Action Mailbox：受信メールのテストシナリオを追加
  - パラレルテスト：複数のテストを同時に実行
    - CI/CDのスピードを最適化

Rails 7の新機能とテスト戦略

- 機能
  - Hotwire（Turbo & Stimulus）
    - ページリロード不要のインタラクティブなUI
  - Async Query：非同期クエリのサポート
  - 複合プライマリキーのサポート
- テスト戦略
  - Hotwire：エンドツーエンド（E2E）テストでのUI反応確認を強化
  - Async Query：非同期処理が関わるテストの信頼性を高める
  - 複合プライマリキー：複雑なデータ構造へのテストケースを追加

Rails 8の新機能とテスト戦略

- 機能
  - より強力なTypeScriptのサポート：TypeScriptを導入したテスト
  - RailsファーストのAPIモードの進化：API向けのテストを強化
- テスト戦略
  - TypeScriptサポート
    - 型安全なテストが可能に、フロントエンドとバックエンドの統合テストを増やす
  - APIモードの進化
    - APIエンドポイントのバリデーションやレスポンステストを強化

Dev Container

- Rails 7.2 から: rails new myapp --devcontainer

rails-new

- https://github.com/rails/rails-new
- Docker をインストールする
- バイナリをダウンロードする
- tar xvfz する
- rails-new バイナリにパスを通す
  - 例えば ~/bin/rails-new に置く

Usage: **rails-new** [OPTIONS] <ARGS>...
rails-new [OPTIONS] [ARGS]... <COMMAND>

Commands:
rails-help Prints `rails new --help`
help Print this message or the help of the given subcommand(s)

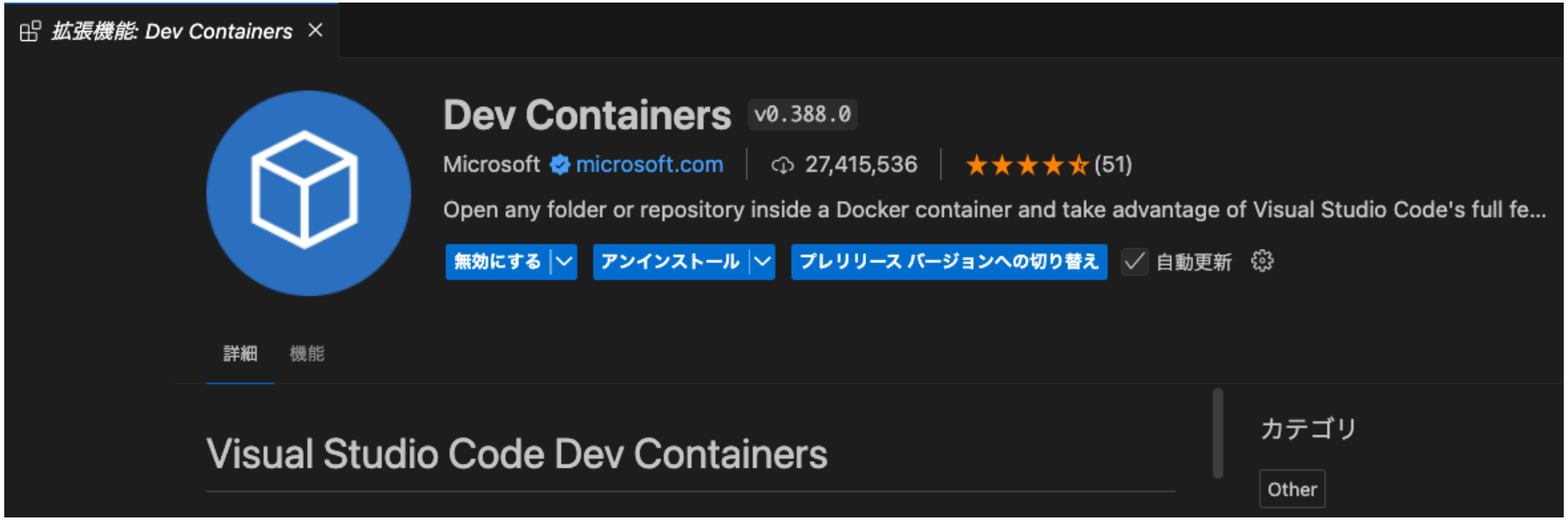
Arguments:
<ARGS>... arguments passed to `rails new`

Options:
-u, **--ruby-version** <RUBY_VERSION> [default: 3.3.4]
-r, **--rails-version** <RAILS_VERSION> [default: 7.2.0]
-h, **--help** Print help
-V, **--version** Print version

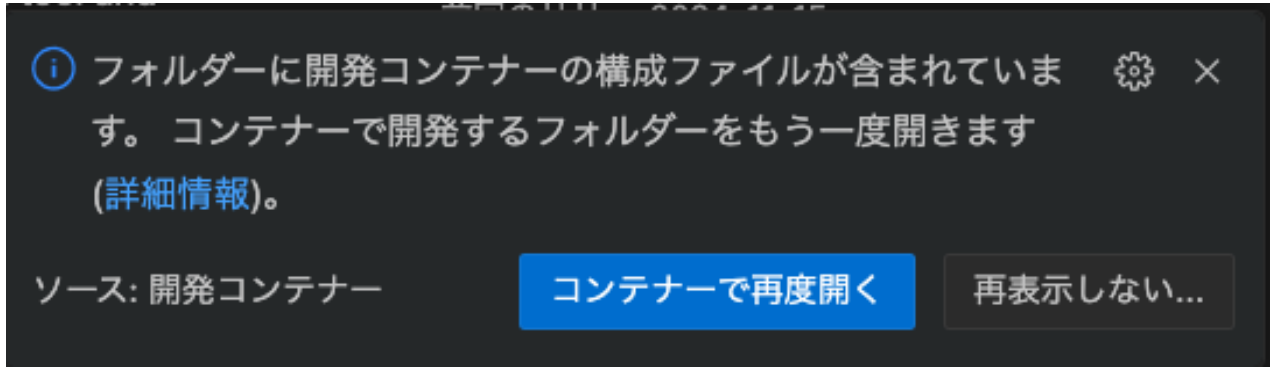
% rails-new -u 3.3.6 -r 8.0.0 hello-rails --devcontainer
% cd hello-rails
% code .

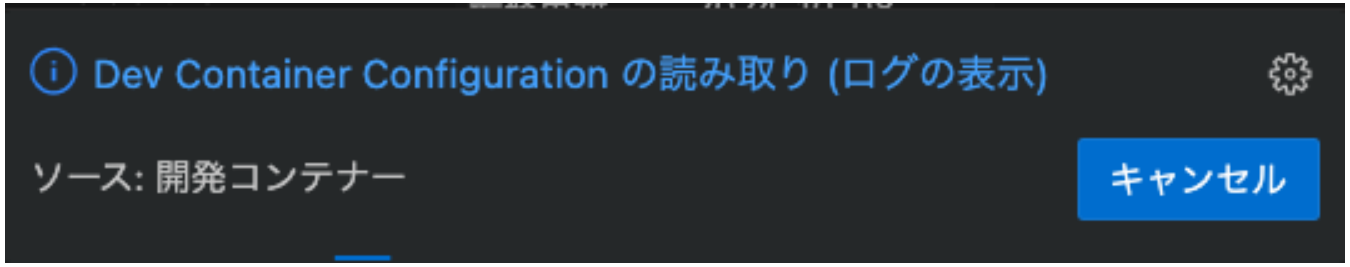
DevContainerを使うには

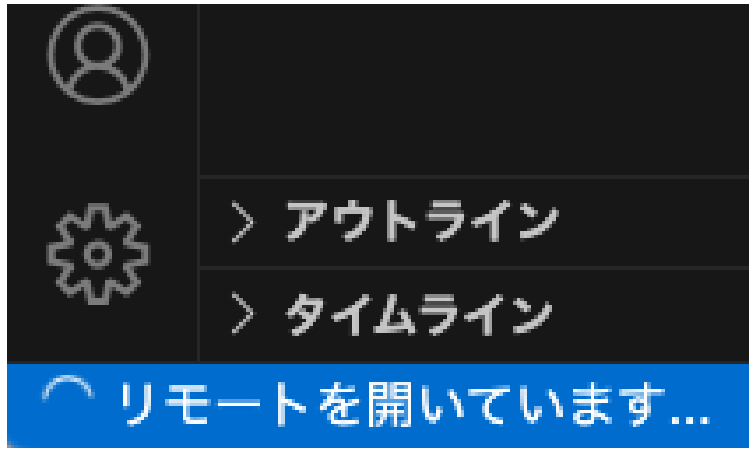
- GitHub Codespaces
  - 無料プランでは限られた時間のみ使用可能
- Visual Studio Code とその拡張機能 "Dev Containers”
  - Rails ガイドの説明はこちら

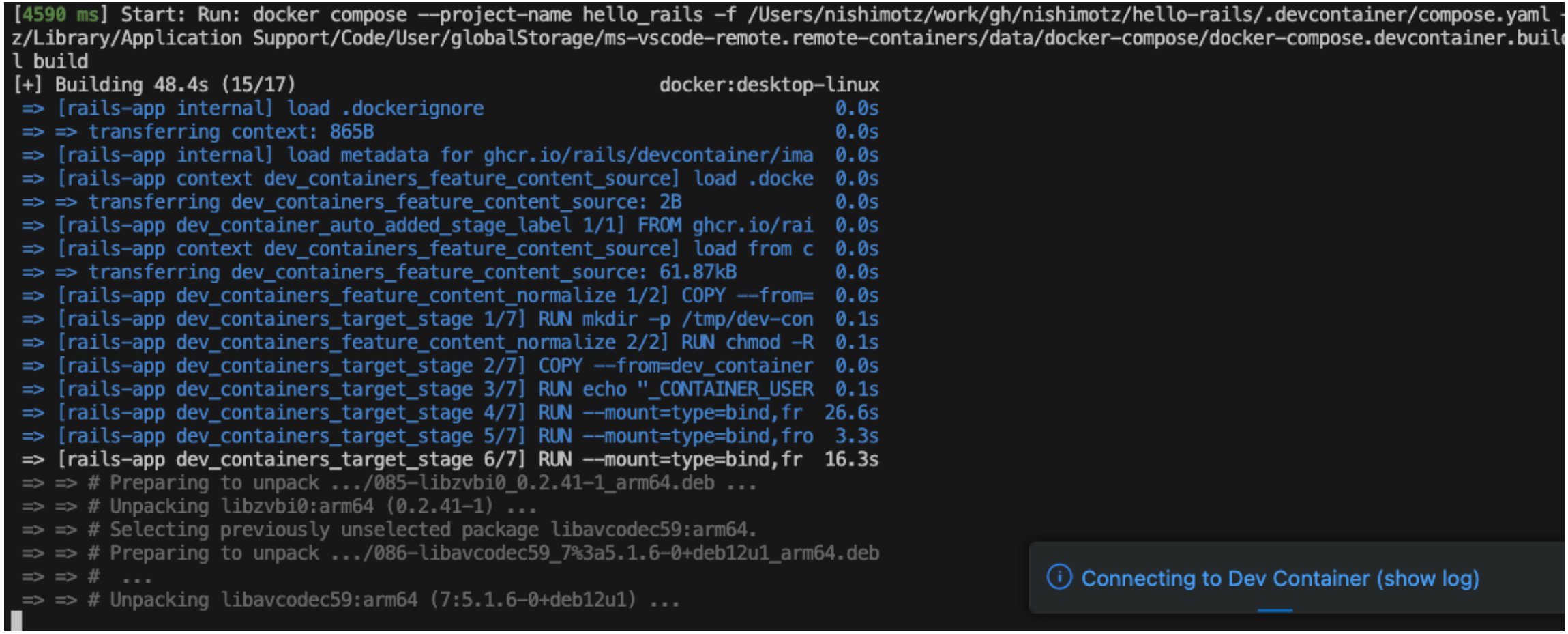


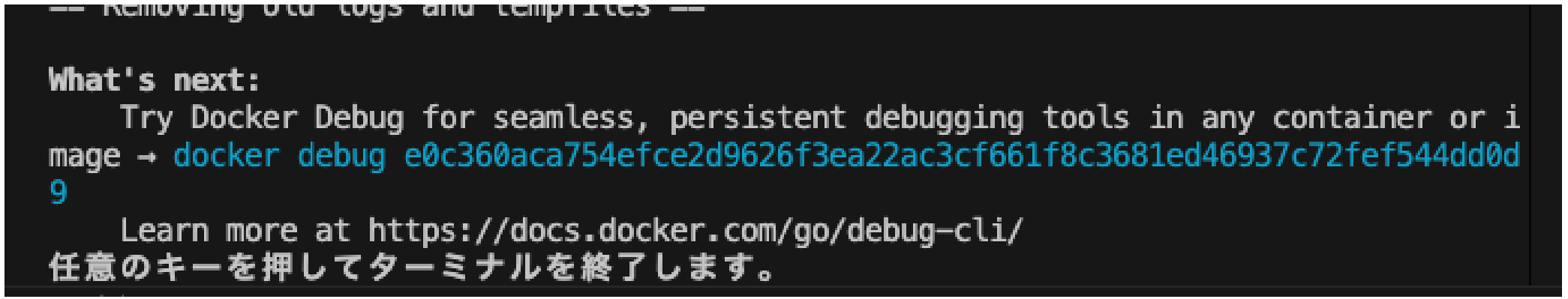
コンテナーで再度開く


フォルダーに開発コンテナーの構成ファイルが含まれていま
す。 コンテナーで開発するフォルダーをもう一度開きます
(詳細情報)。
ソース: 開発コンテナー
コンテナーで再度開く
再表示しない...

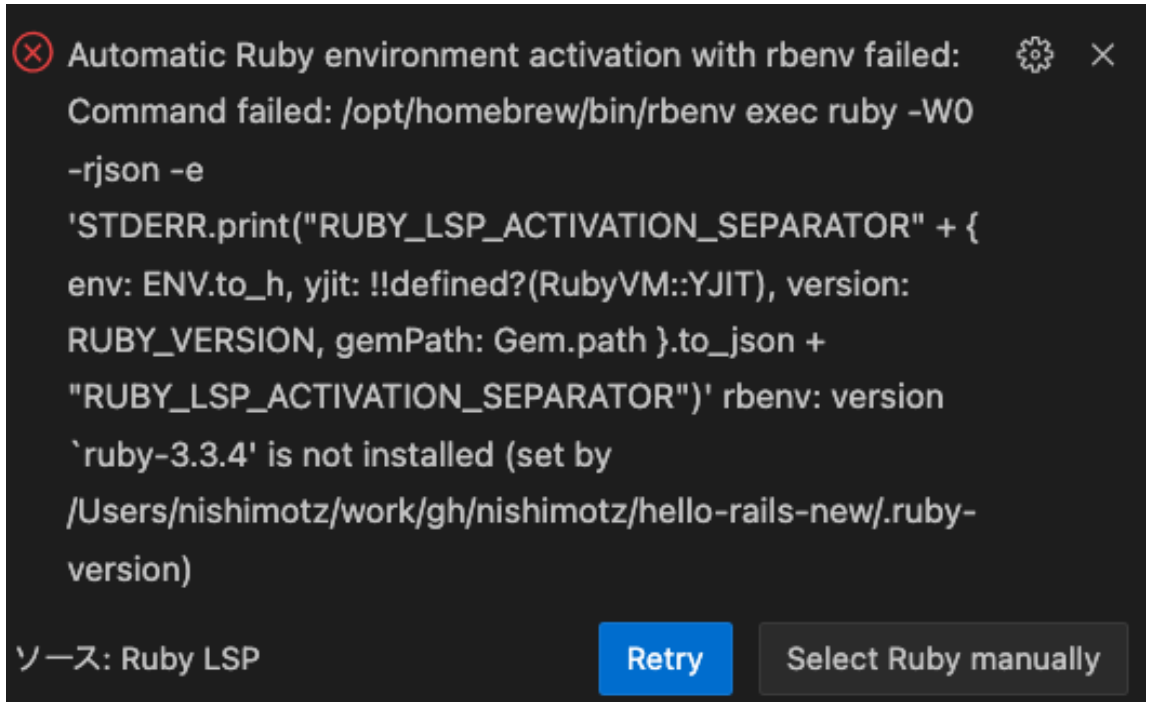

Dev Container Configuration の読み取り (ログの表示)
ソース: 開発コンテナー
キャンセル


アウトライン
タイムライン
リモートを開いています...

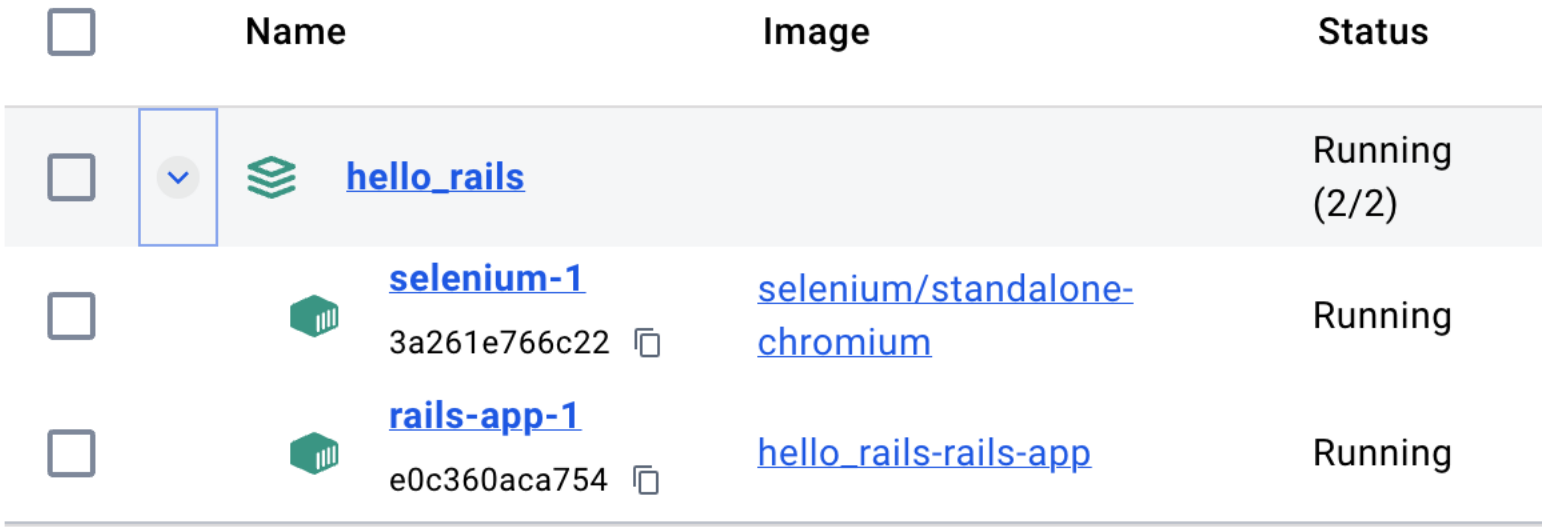

[4590 ms] Start: Run: docker compose --project-name hello_rails -f /Users/nishimotz/work/gh/nishimotz/hello-rails/.devcontainer/compose.yaml -
z/Library/Application Support/Code/User/globalStorage/ms-vscode-remote.remote-containers/data/docker-compose/docker-compose.devcontainer.buil
l build
[+] Building 48.4s (15/17) docker:desktop-linux
=> [rails-app internal] load .dockerignore 0.0s
=> => transferring context: 865B 0.0s
=> [rails-app internal] load metadata for ghcr.io/rails/devcontainer/ima 0.0s
=> [rails-app context dev_containers_feature_content_source] load .docke 0.0s
=> => transferring dev_containers_feature_content_source: 2B 0.0s
=> [rails-app dev_container_auto_added_stage_label 1/1] FROM ghcr.io/rai 0.0s
=> [rails-app context dev_containers_feature_content_source] load from c 0.0s
=> => transferring dev_containers_feature_content_source: 61.87kB 0.0s
=> [rails-app dev_containers_feature_content_normalize 1/2] COPY --from= 0.0s
=> [rails-app dev_containers_target_stage 1/7] RUN mkdir -p /tmp/dev-con 0.1s
=> [rails-app dev_containers_feature_content_normalize 2/2] RUN chmod -R 0.1s
=> [rails-app dev_containers_target_stage 2/7] COPY --from=dev_container 0.0s
=> [rails-app dev_containers_target_stage 3/7] RUN echo "_CONTAINER_USER 0.1s
=> [rails-app dev_containers_target_stage 4/7] RUN --mount=type=bind,fr 26.6s
=> [rails-app dev_containers_target_stage 5/7] RUN --mount=type=bind,fro 3.3s
=> [rails-app dev_containers_target_stage 6/7] RUN --mount=type=bind,fr 16.3s
=> => # Preparing to unpack .../085-libzvbi0_0.2.41-1_arm64.deb ...
=> => # Unpacking libzvbi0:arm64 (0.2.41-1) ...
=> => # Selecting previously unselected package libavcodec59:arm64.
=> => # Preparing to unpack .../086-libavcodec59_7%3a5.1.6-0+deb12u1_arm64.deb
=> => # ...
=> => # Unpacking libavcodec59:arm64 (7:5.1.6-0+deb12u1) ...
Connecting to Dev Container (show log)


What's next:
Try Docker Debug for seamless, persistent debugging tools in any container or i
mage → docker debug e0c360aca754efce2d9626f3ea22ac3cf661f8c3681ed46937c72fef544dd0d
9
Learn more at https://docs.docker.com/go/debug-cli/
任意のキーを押してターミナルを終了します。

Ruby LSP の警告を無視する

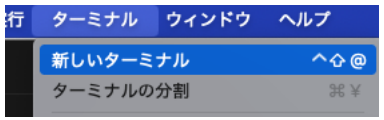


Docker Desktop

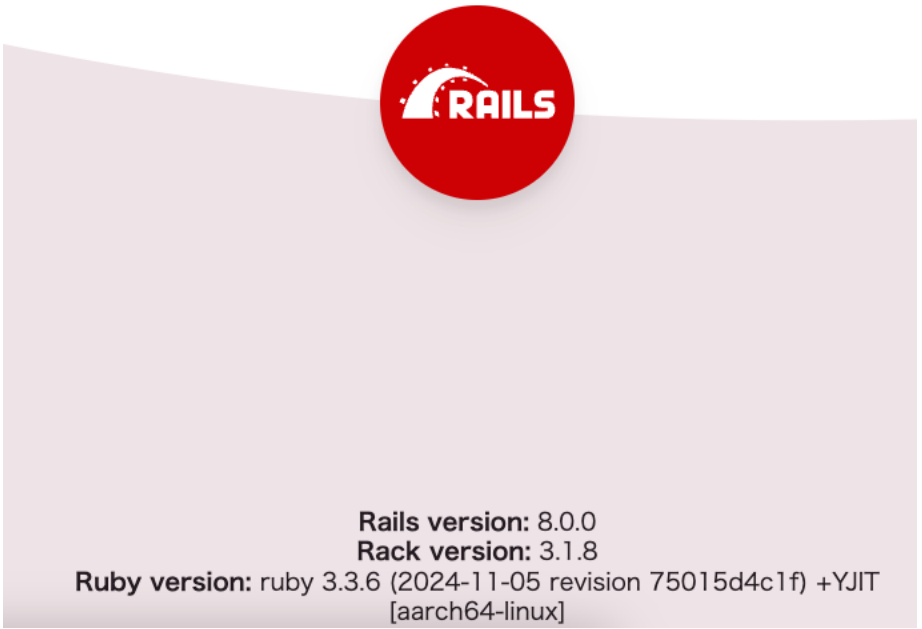


新しいターミナル

- rails –version
- bin/rails server
- Chrome で http://127.0.0.1:3000/

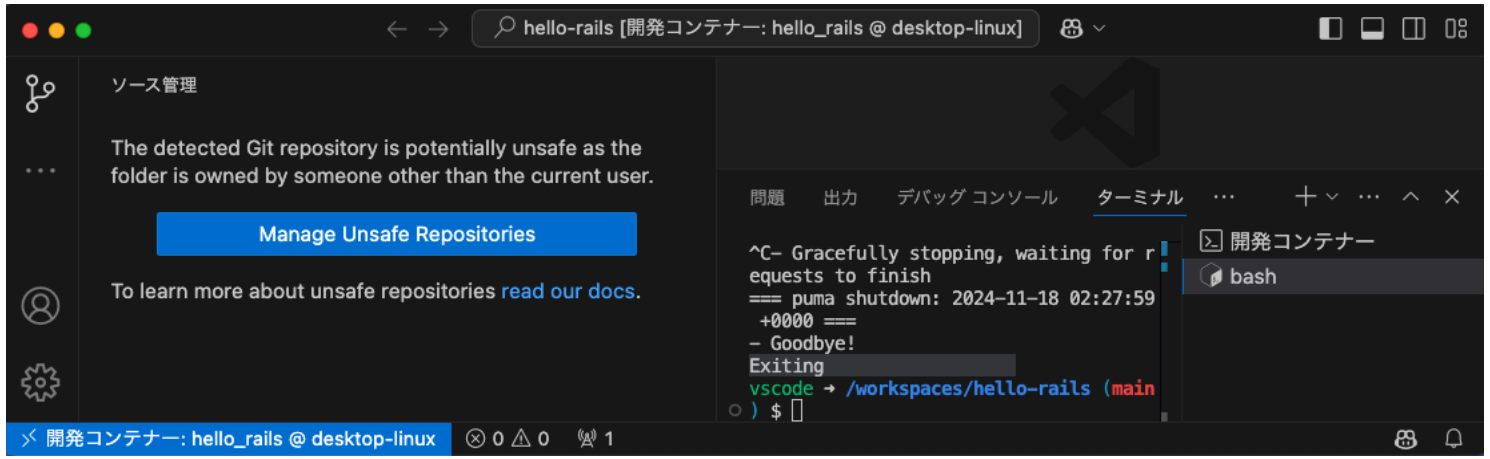



```
vscode ➜ /workspaces/hello-rails (main) $ rails --version
Rails 8.0.0
vscode ➜ /workspaces/hello-rails (main) $ sqlite3 --version
3.40.1 2022-12-28 14:03:47 df5c253c0b3dd24916e4ec7cf77d3db5294cc9fd45ae7b9
c5e82ad8197f3alt1
vscode ➜ /workspaces/hello-rails (main) $ bin/rails server
=> Booting Puma
=> Rails 8.0.0 application starting in development
=> Run `bin/rails server --help` for more startup options
Puma starting in single mode...
* Puma version: 6.4.3 (ruby 3.3.6-p108) ("The Eagle of Durango")
*  Min threads: 3
*  Max threads: 3
*  Environment: development
*          PID: 3529
* Listening on http://127.0.0.1:3000
* Listening on http://[::1]:3000
Use Ctrl-C to stop
```



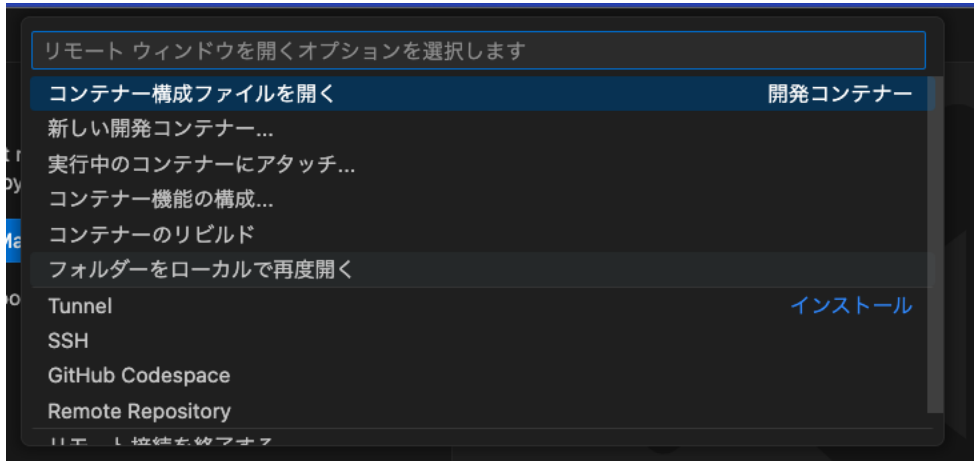
git はどこでどう使うのか

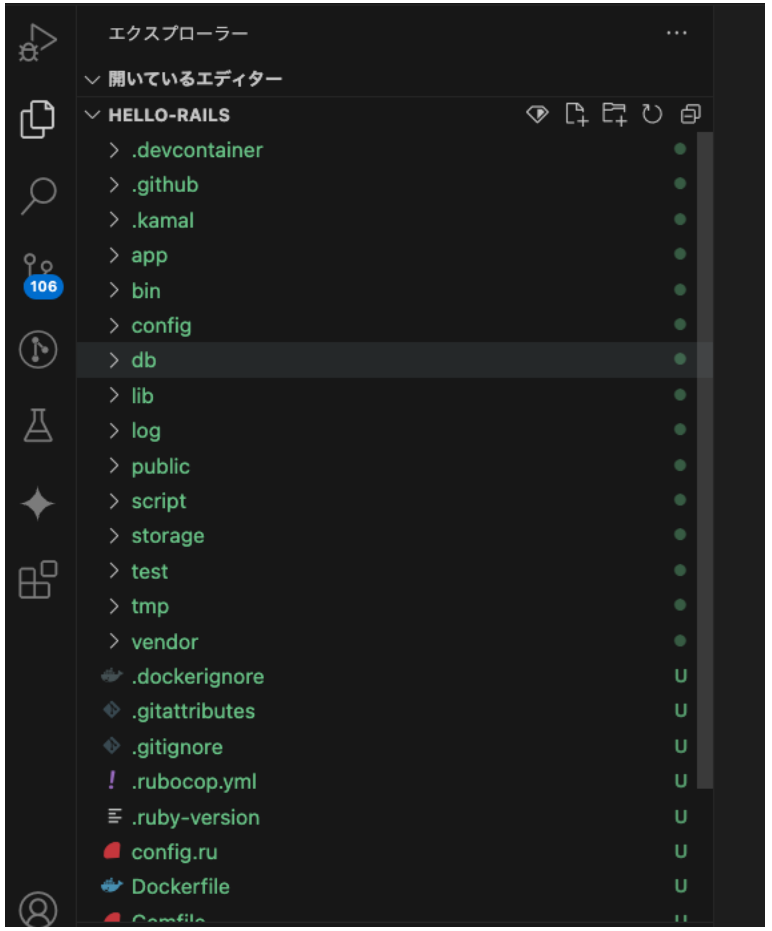
```
nishimotz@mac-8094 hello-rails % git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .devcontainer/
        .dockerignore
        .gitattributes
        .github/
        .gitignore
        .kamal/
        .rubocop.yml
        .ruby-version
        Dockerfile
        Gemfile
        Gemfile.lock
        README.md
        Rakefile
```



フォルダーをローカルで再度開く

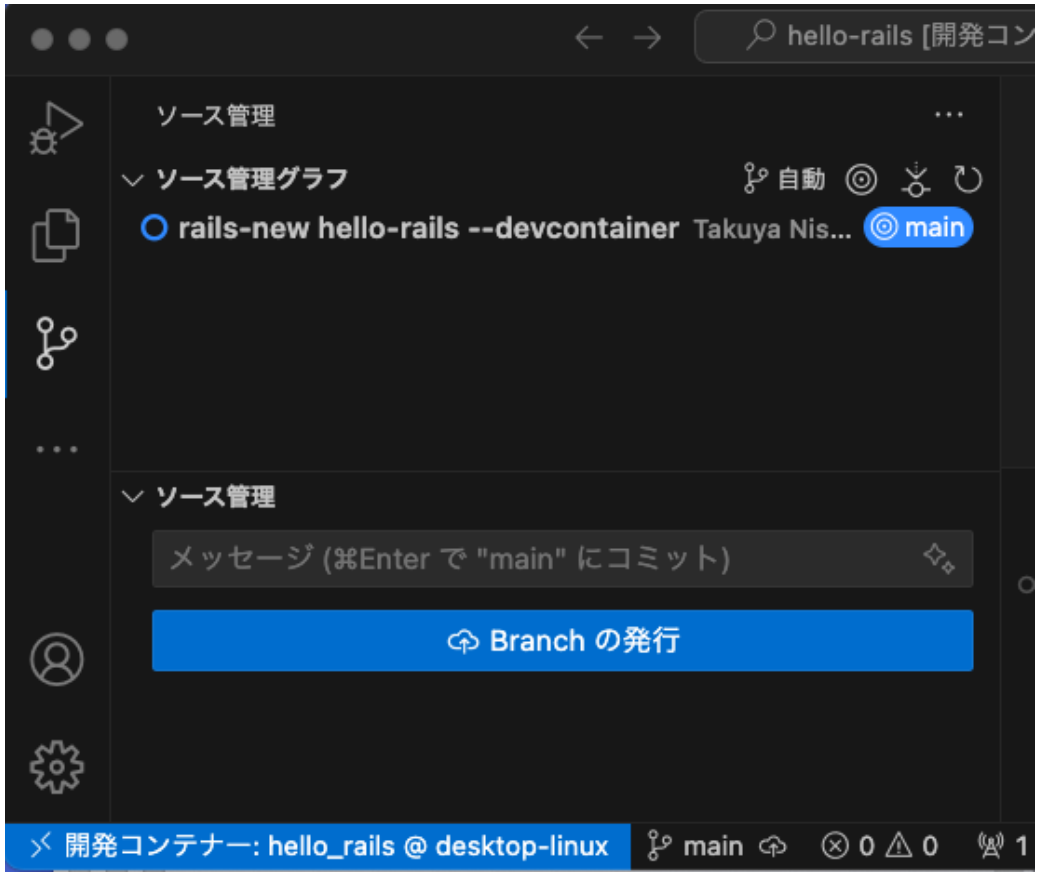
- ターミナルで見ている状態になる
- こちらでコミットする





開発コンテナーに戻る

- こちらからソース管理してもいい？

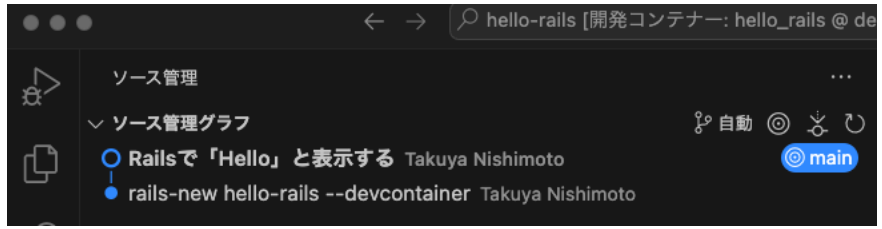


Hello と表示する

- config/routes.rb
- bin/rails generate controller Articles index --skip-routes
- app/views/articles/index.html.erb
- http://127.0.0.1:3000/articles

開発コンテナーからgit

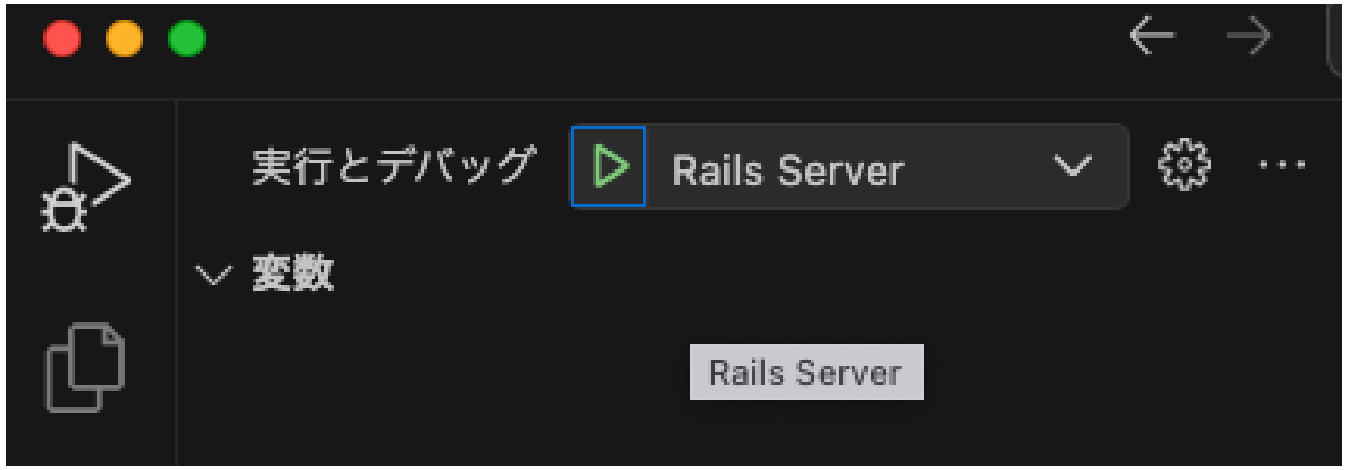
- ローカルに戻す必要がない

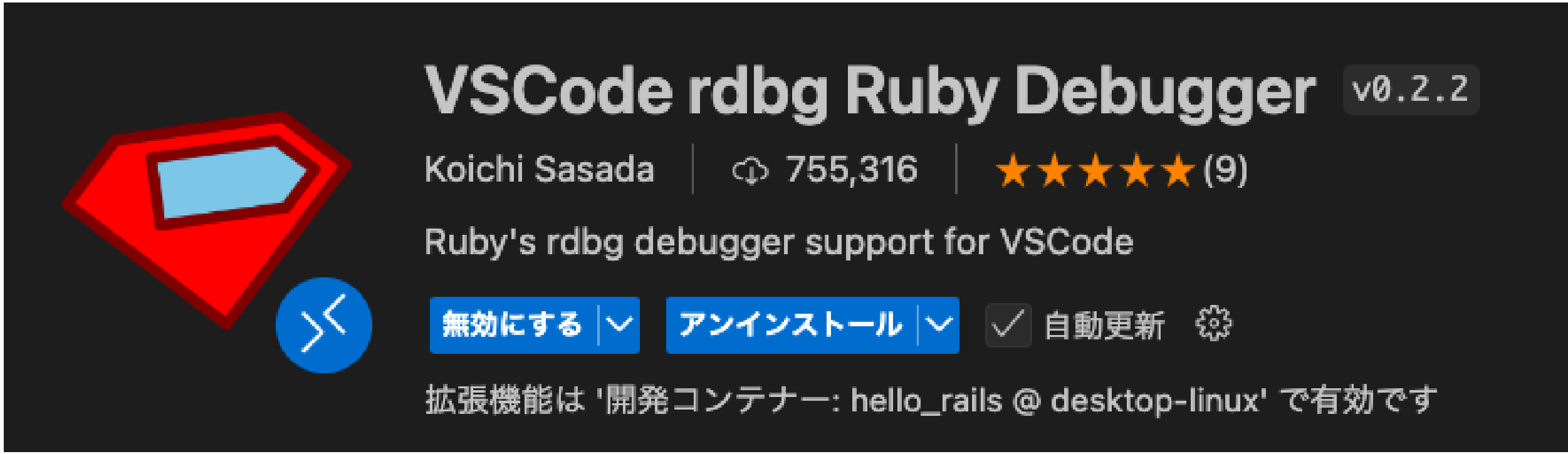



```
nishimotz@mac-8094 hello-rails % git show
commit b61db67a8766f12ddbddeffdda365eb2c67741d8 (HEAD -> main)
Author: Takuya Nishimoto <nishimotz@gmail.com>
Date:   Mon Nov 18 02:46:26 2024 +0000

    Railsで「Hello」と表示する

diff --git a/app/controllers/articles_controller.rb b/app/contro
new file mode 100644
index 0000000..e5904fd
--- /dev/null
+++ b/app/controllers/articles_controller.rb
@@ -0,0 +1,4 @@
+class ArticlesController < ApplicationController
+  def index
+  end
+end
diff --git a/app/helpers/articles_helper.rb b/app/helpers/articl
new file mode 100644
index 0000000..2968277
--- /dev/null
+++ b/app/helpers/articles_helper.rb
```

デバッグ環境を試す

- bundle add debug
- 実行とデバッグ
- launch.json ファイルを作成

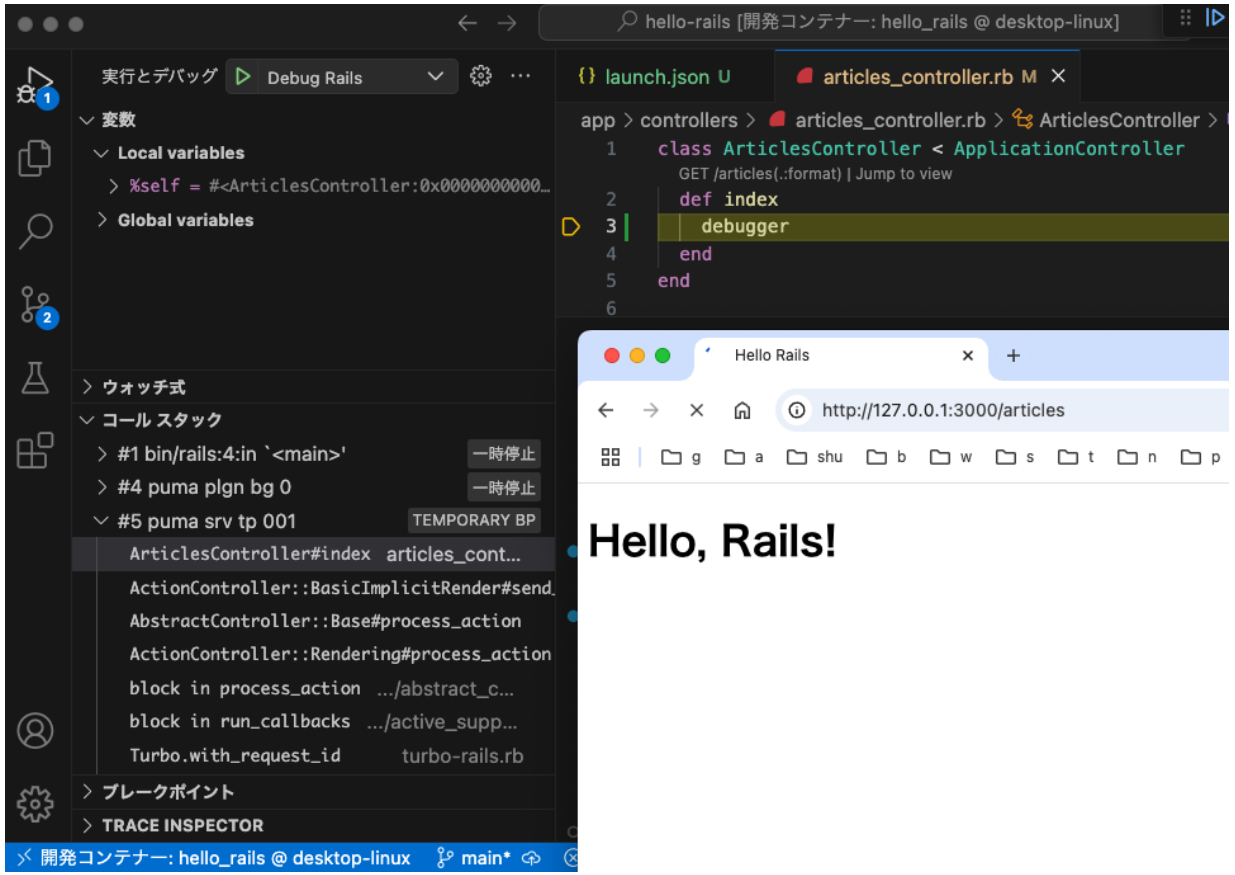

実行とデバッグ
Rails Server
変数
Rails Server


VSCode rdbg Ruby Debugger
v0.2.2
Koichi Sasada
755,316
(9)
Ruby's rdbg debugger support for VSCode
無効にする
アンインストール
自動更新
拡張機能は '開発コンテナー: hello_rails @ desktop-linux' で有効です

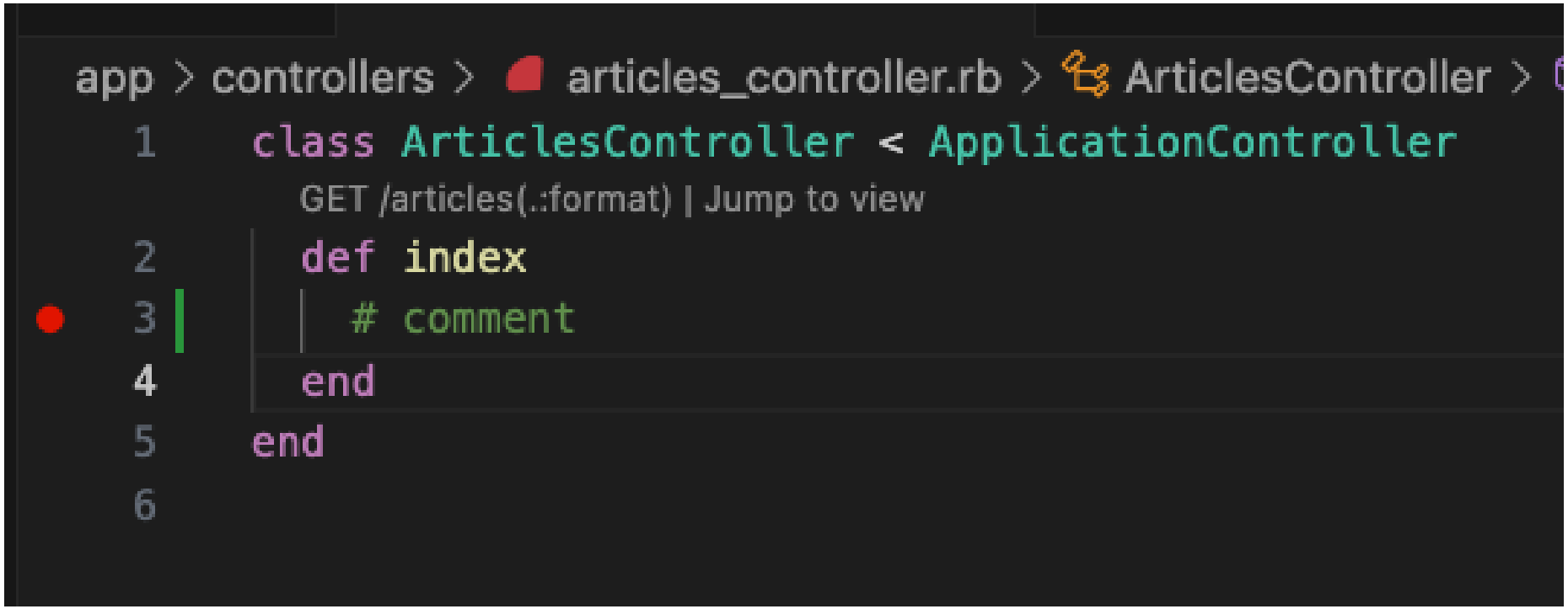
```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Debug Rails",
      "type": "rdbg",
      "request": "launch",
      "command": "bin/rails",
      "script": "s",
      "args": [
        "-b",
        "0.0.0.0"
      ]
    },
    {
      "name": "Debug Minitest with current file",
      "type": "rdbg",
      "request": "launch",
      "command": "ruby",
      "script": "${file}"
    }
  ]
}
```

debuggerコマンドで止まる

- macOS から仮想マシンの ruby をデバッグできている

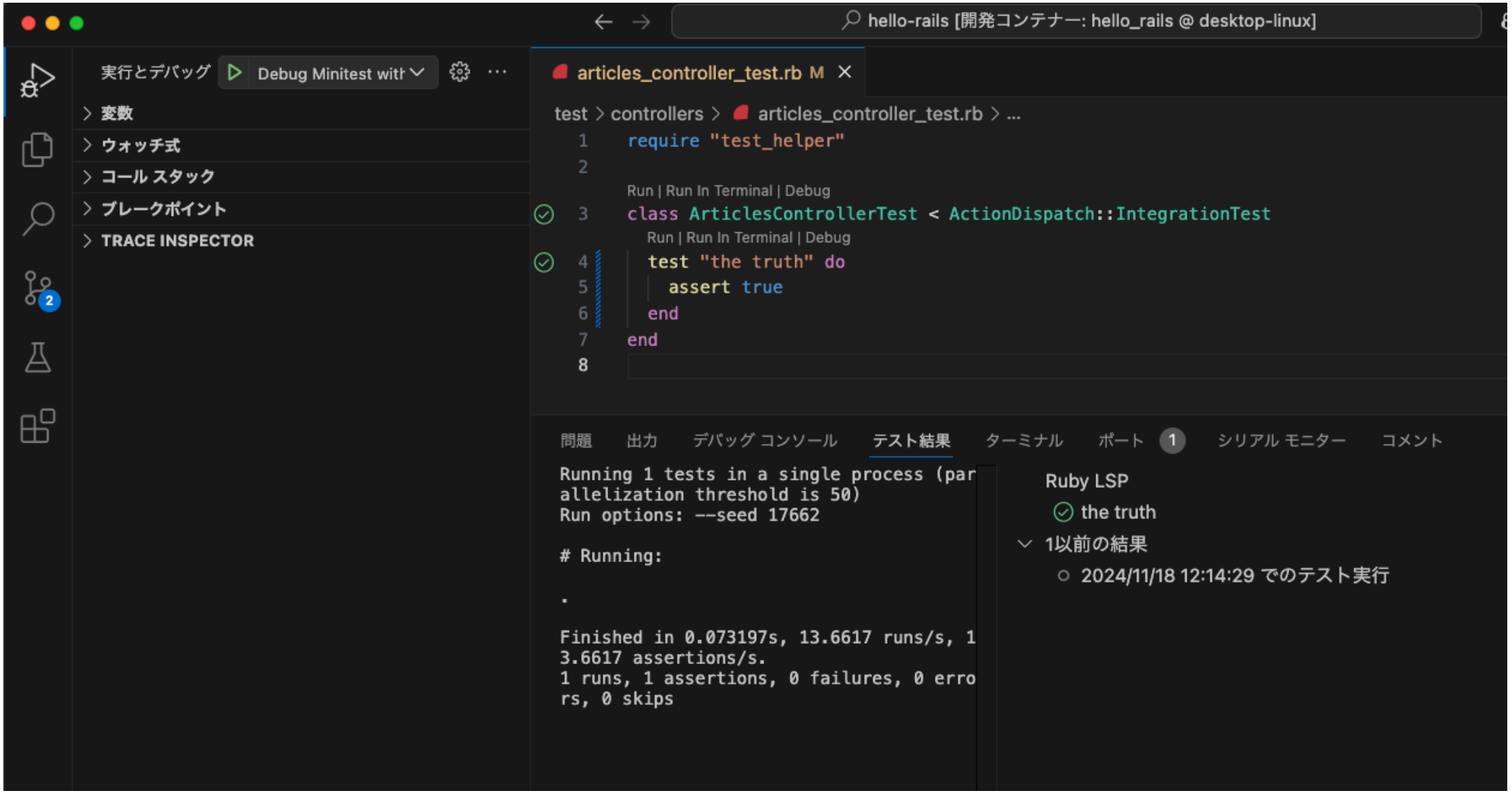


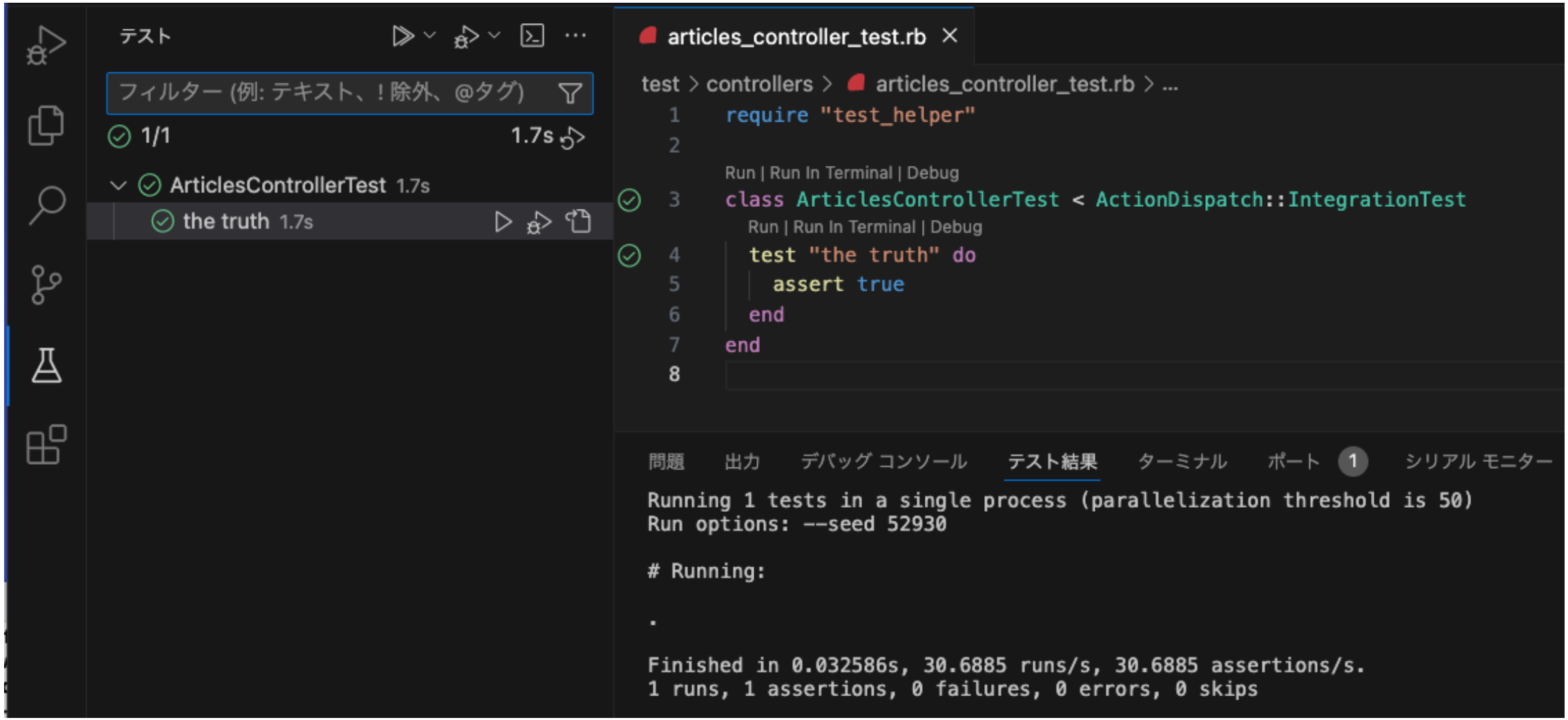
ブレークポイントでも止まる



Debug Minitest with current file

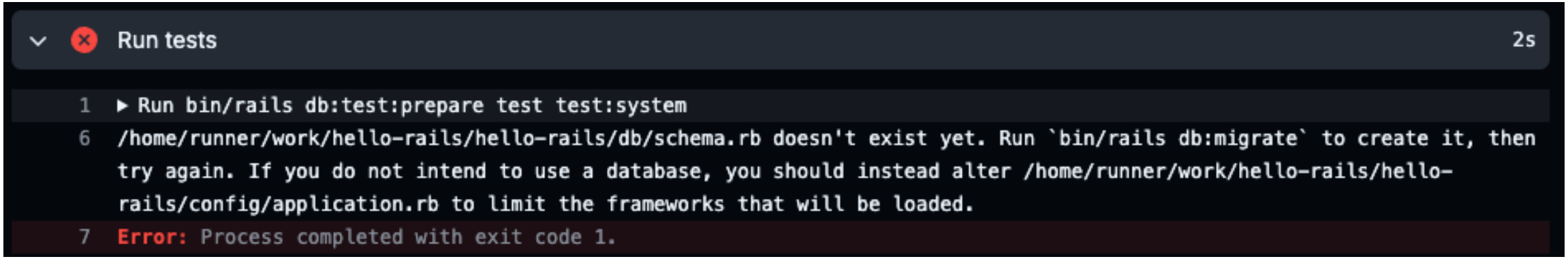
- articles_controller_test.rb
- the truth 有効化
- ファイルを開いた状態で実行


hello-rails [開発コンテナー: hello_rails @ desktop-linux]
実行とデバッグ
Debug Minitest with
変数
ウォッチ式
コール スタック
ブレークポイント
TRACE INSPECTOR
articles_controller_test.rb M
test > controllers > articles_controller_test.rb > ...
1 require "test_helper"
2
Run | Run In Terminal | Debug
3 class ArticlesControllerTest < ActionDispatch::IntegrationTest
Run | Run In Terminal | Debug
4 test "the truth" do
5 assert true
6 end
7 end
8
問題
出力
デバッグ コンソール
テスト結果
ターミナル
ポート 1
シリアル モニター
コメント
Running 1 tests in a single process (parallelization threshold is 50)
Run options: --seed 17662
Running:
.
Finished in 0.073197s, 13.6617 runs/s, 13.6617 assertions/s.
1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
Ruby LSP
the truth
1以前の結果
2024/11/18 12:14:29 でのテスト実行

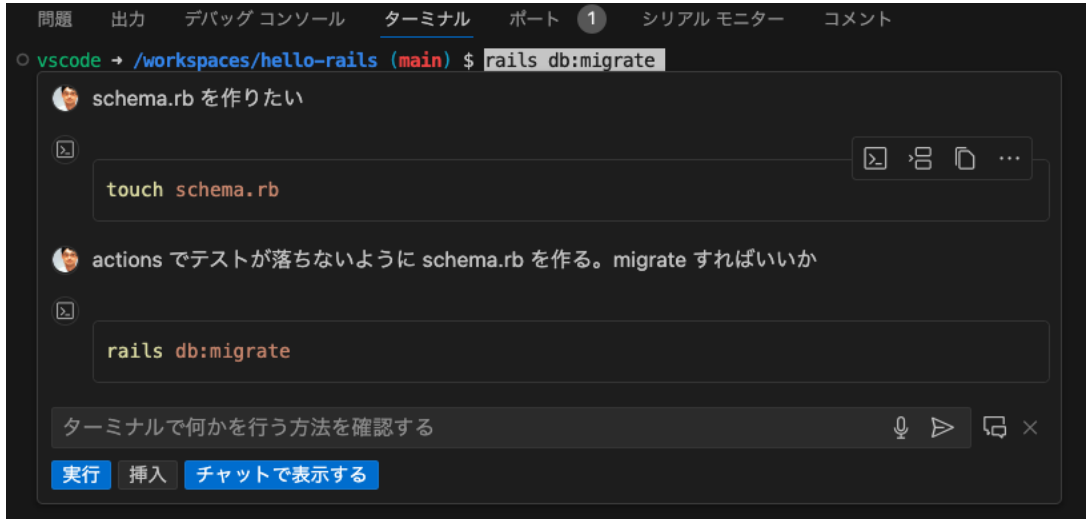

テスト
フィルター (例: テキスト、! 除外、@タグ)
1/1
1.7s
ArticlesControllerTest 1.7s
the truth 1.7s
articles_controller_test.rb
test > controllers > articles_controller_test.rb > ...
1 require "test_helper"
2
Run | Run In Terminal | Debug
3 class ArticlesControllerTest < ActionDispatch::IntegrationTest
Run | Run In Terminal | Debug
4 test "the truth" do
5 assert true
6 end
7 end
8
問題
出力
デバッグ コンソール
テスト結果
ターミナル
ポート 1
シリアル モニター
Running 1 tests in a single process (parallelization threshold is 50)
Run options: --seed 52930
Running:
.
Finished in 0.032586s, 30.6885 runs/s, 30.6885 assertions/s.
1 runs, 1 assertions, 0 failures, 0 errors, 0 skips

gh repo create すると actions が落ちる

- db/schema.rb doesn't exist yet

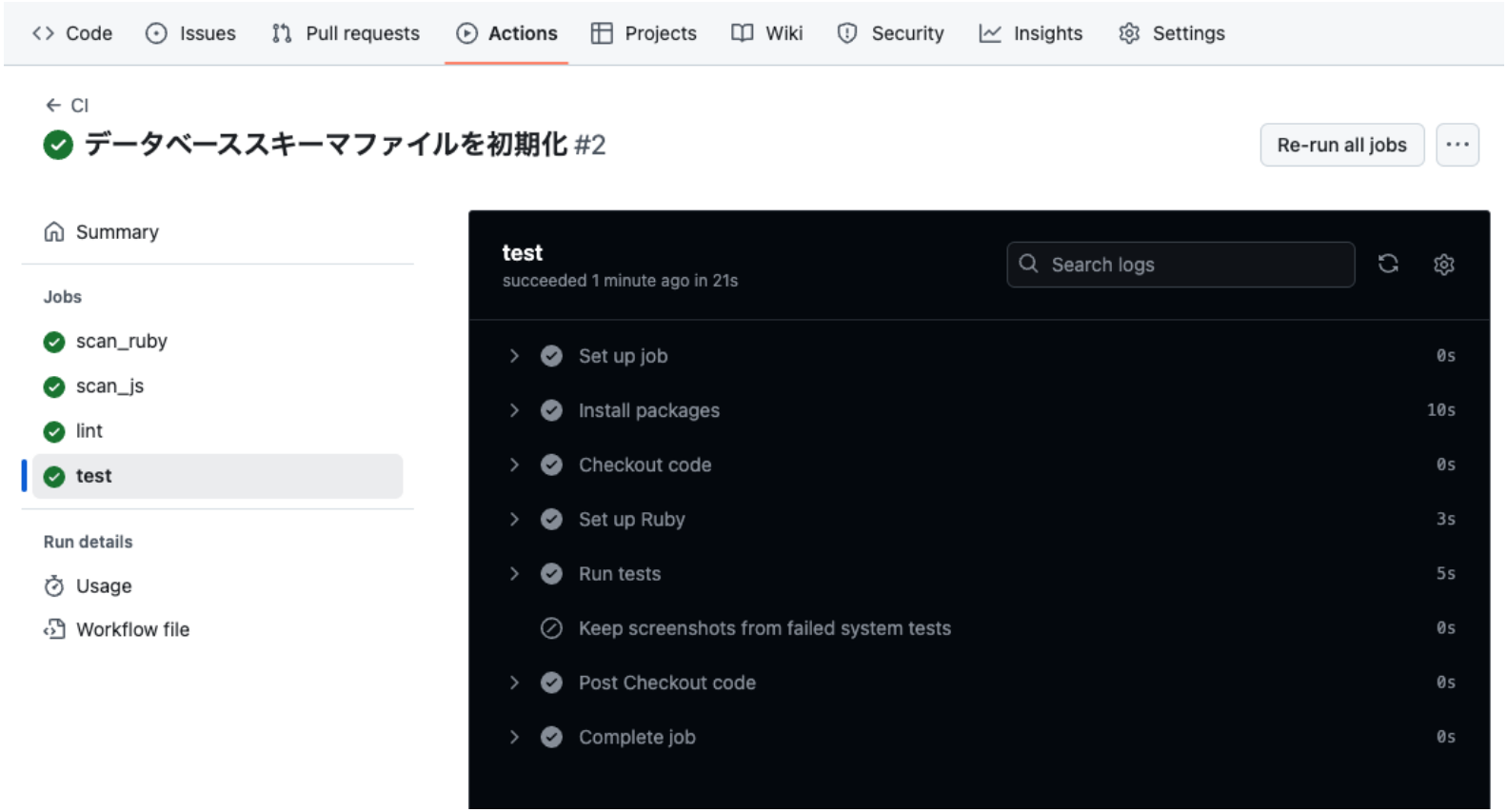


ターミナルで rails db:migrate




```
ActiveRecord::Schema[8.0].define(version: 0) do
end
```

Actions が緑になる



ci.yml の内容

- scan_ruby: brakeman でRuby コードの静的解析を行う
  - SQLインジェクション
  - マスアサインメント
  - セッション固定
- scan_js: JavaScript 依存関係のセキュリティ脆弱性
  - XSS, CSRF, インジェクション攻撃, 認証認可
  - 依存関係
- lint: RuboCop を実行
  - コードスタイル、未使用変数、メソッド複雑度、不要な空行

Kamalでデプロイにも活用

- DevContainer 内に Kamal をインストール
- Docker がインストールされているサーバを用意
- コンテナ内のターミナルで kamal deploy を実行
  - docker-in-docker (DinD)
    - GitHub Actions など CI/CD で推奨
    - セキュリティ的に有利
  - docker-outside-of-docker (DoD)
    - macOS, Windows, Linux など一般的な開発環境で推奨
    - 性能的に有利
- SSH 経由で Kamal がサーバに接続してデプロイ