

# GITHUB ACTIONS クイックリファレンス

ワークフロー、トリガー、ジョブ、シークレット、キャッシュ、アーティファクト

## ワークフローの基本

### 最小ワークフロー

```
# .github/workflows/ci.yml
name: CI
on: push
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: echo "Hello from CI"
```

### 主要なコンセプト

- Workflow** 自動化を定義する`.github/workflows/`のYAMLファイル
- Event** ワークフローを起動するトリガー (push、PR、scheduleなど)
- Job** 同じランナーで実行されるステップのセット
- Step** 個別のタスク。コマンドを実行するかアクションを使用
- Runner** ジョブを実行するVM (ubuntu-latest、macos-latest、windows-latest)
- Action** uses: で参照する再利用可能なコードの単位

## トリガー

### よく使うイベント

```
on:
  push:
    branches: [main]
  pull_request:
    branches: [main]
  schedule:
    - cron: "0 6 * * 1" # 毎週月曜 UTC 6 時
  workflow_dispatch: # 手動トリガー
```

### イベントフィルター

- branches:** 特定のブランチのみトリガー
- paths:** 一致するファイルが変更された場合のみトリガー
- tags:** タグのプッシュでトリガー (v\*)
- types: [opened, synchronize]** PRのアクティビティタイプをフィルタ
- branches-ignore:** 特定のブランチを除外
- paths-ignore:** 特定のファイルパスを除外

## ジョブとステップ

### ジョブの設定

```
jobs:
  test:
    runs-on: ubuntu-latest
    needs: build # build ジョブに依存
    if: github.ref == 'refs/heads/main'
    timeout-minutes: 10
    steps:
      - uses: actions/checkout@v4
      - run: npm test
```

### ステップの種類

- run:** シェルコマンドを実行
- uses:** 公開されたアクションを使用
- with:** アクションに入力を渡す
- name:** UIでの表示名
- id:** `steps.<id>.outputs`でステップ出力を参照
- if:** 条件付き実行
- continue-on-error: true** ステップが失敗してもジョブを失敗させない

## アクション

### アクションの使用

```
steps:
  - uses: actions/checkout@v4
  - uses: actions/setup-node@v4
    with:
      node-version: 20
  - uses: ./github/actions/my-action # ローカルアクション
```

### 人気のアクション

- actions/checkout@v4** リポジトリのコードをチェックアウト
- actions/setup-node@v4** Node.jsをインストール
- actions/setup-python@v5** Pythonをインストール
- actions/upload-artifact@v4** ビルドアーティファクトをアップロード
- actions/download-artifact@v4** 別ジョブのアーティファクトをダウンロード
- actions/cache@v4** 実行間で依存関係をキャッシュ
- actions/github-script@v7** GitHub API クライアントでJSを実行

## 環境変数

### 変数の設定

```
env:
  NODE_ENV: production # ワークフローレベル
jobs:
  build:
    env:
      CI: true # ジョブレベル
    steps:
      - run: echo "$MY_VAR"
        env:
          MY_VAR: hello # ステップレベル
```

### デフォルト変数

- github.sha** ワークフローをトリガーしたコミット SHA
- github.ref** ブランチまたはタグのref (refs/heads/main)
- github.repository** オwner/リポジトリ名
- github.actor** ワークフローをトリガーしたユーザー
- github.event\_name** ワークフローをトリガーしたイベント
- runner.os** ランナーのOS (Linux、macOS、Windows)

## シークレット

### シークレットの使用

```
steps:
  - run: deploy --token "$TOKEN"
    env:
      TOKEN: ${ secrets.DEPLOY_TOKEN }
  - uses: some/action@v1
    with:
      api-key: ${ secrets.API_KEY }
```

### シークレットのルール

- secrets.GITHUB\_TOKEN** リポジトリにスコープされた自動生成トークン
- Settings → Secrets** リポジトリまたは組織設定でシークレットを追加
- マスキング** シークレット値はログで自動的にマスキングされる
- 環境シークレット** デプロイ環境にスコープされる
- 組織シークレット** 組織内のリポジトリ間で共有

## マトリクス戦略

### マトリクスビルド

```
jobs:
  test:
    strategy:
      matrix:
        os: [ubuntu-latest, macos-latest]
        node: [18, 20]
    runs-on: ${ matrix.os }
    steps:
      - uses: actions/setup-node@v4
        with:
          node-version: ${ matrix.node }
```

### マトリクスオプション

- matrix:** 展開する変数の組み合わせを定義
- include:** マトリクスに追加の組み合わせを加える
- exclude:** 特定の組み合わせを削除
- fail-fast: false** 1つが失敗しても他のジョブを継続
- max-parallel: 2** 同時実行するマトリクスジョブを制限

## キャッシュ

### 依存関係をキャッシュ

```
- uses: actions/cache@v4
  with:
    path: ~/.npm
    key: npm-${ hashFiles('package-lock.json') }
    restore-keys: npm-
```

### 組み込みキャッシュ

```
- uses: actions/setup-node@v4
  with:
    node-version: 20
    cache: npm # npm/yarn/pnpmを自動キャッシュ
- uses: actions/setup-python@v5
  with:
    python-version: "3.12"
    cache: pip # pipを自動キャッシュ
```

## アーティファクト

### アップロードとダウンロード

```
- uses: actions/upload-artifact@v4
  with:
    name: build-output
    path: dist/
    retention-days: 7
- uses: actions/download-artifact@v4
  with:
    name: build-output
```

### アーティファクトの注意点

- retention-days** N日後に自動削除 (デフォルト 90)
- path** アップロードするファイルまたはディレクトリ (ジョブ対話)
- ジョブ間** 1つのジョブでアップロード、`needs:`で別ジョブからダウンロード
- compression-level** 0 (なし) ~ 9 (最大)、デフォルト 6

## よくあるパターン

### 条件付きデプロイ

```
- name: Deploy to production
  if: github.ref == 'refs/heads/main'
  run: ./deploy.sh
- name: Post PR comment
  if: github.event_name == 'pull_request'
  run: gh pr comment $PR --body "Build passed"
```

### 便利な式

- success()** 前のすべてのステップが成功した場合にtrue
- failure()** 前のいずれかのステップが失敗した場合にtrue
- always()** ステータスに関わらず実行 (クリーンアップ)

### cancelled()

ワークフローがキャンセルされた場合にtrue

### contains(github.ref, 'release')

文字列の含有チェック

### startsWith(github.ref, 'refs/tags')

文字列のプレフィックスチェック

### hashFiles(\*\*/lock\*)

ファイルのSHA-256 (キャッシュキー用)