報道各位

2020年6月17日
株式会社インプレスR&D
https://nextpublishing.jp/

もっと手軽にCI/CD活用！

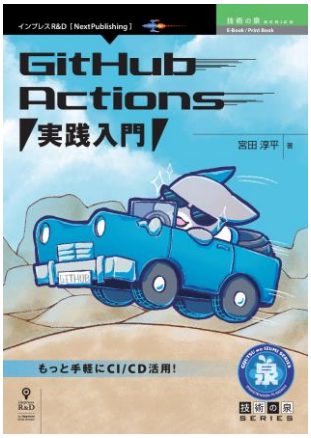
『GitHub Actions 実践入門』発行

技術の泉シリーズ、6月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレスR&Dは、『GitHub Actions 実践入門』(著者:宮田 淳平)を発行いたします。

最新の知見を発信する『技術の泉シリーズ』は、「技術書典」や「技術書同人誌博覧会」をはじめとした各種即売会や、勉強会・LT 会などで頒布された技術同人誌を底本とした商業書籍を刊行し、技術同人誌の普及と発展に貢献することを目指します。

『GitHub Actions 実践入門』
https://nextpublishing.jp/isbn/9784844378716

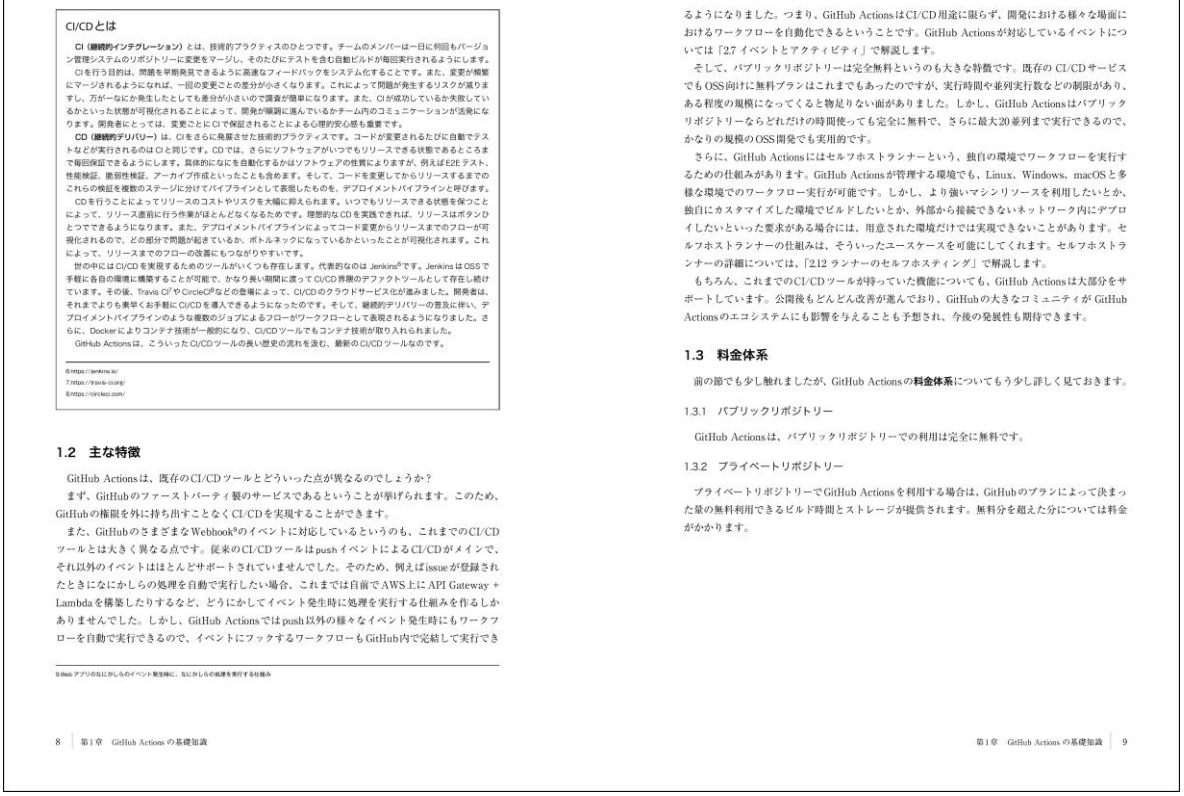


著者:宮田 淳平
小売希望価格:電子書籍版 1600円(税別)／印刷書籍版 2000円(税別)
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5判／カラー／本文130ページ
ISBN:978-4-8443-7871-6
発行:インプレスR&D

＜＜発行主旨・内容紹介＞＞

本書はGitHubが提供するCI/CDサービスの「GitHub Actions」の解説書です。
基礎的な知識をはじめ、実際の開発現場で活用するところまでを丁寧に解説しています。
(本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。)

GitHub Actions の基本を紹介



CI/CDとは

CI（継続的インテグレーション）とは、技術的プラクティスのひとつです。チームのメンバーは一日に何回もバージョン管理システムのリポジトリーに変更をマージし、そのたびにテストを含む自動ビルドが毎回実行されるようにします。

CIを行う目的は、問題を早期発見できるように高速なフィードバックをシステム化することです。また、変更が頻繁にマージされるようになれば、一回の変更ごとの差分が小さくなります。これによって問題が発生するリスクが減りますし、万が一なにか発生したとしても差分が小さいので調査が簡単になります。また、CIが成功しているか失敗しているかといった状態が可視化されることによって、開発が順調に進んでいるかチーム内のコミュニケーションが活発になります。開発者にとっては、変更ごとにCIで保証されることによる心理的安心感も重要です。

CD（継続的デリバリー）は、CIをさらに発展させた技術的プラクティスです。コードが変更されるたびに自動でテストなどが実行されるのはCIと同じです。CDでは、さらにソフトウェアがいつでもリリースできる状態であるところまで毎回保証できるようにします。具体的になにを自動化するかはソフトウェアの性質によりますが、例えばE2Eテスト、性能検証、脆弱性検証、アーカイブ作成といったことも含めます。そして、コードを変更してからリリースするまでのこれらの検証を複数のステージに分けてパイプラインとして表現したものを、デプロイメントパイプラインと呼びます。

CDを行うことによってリリースのコストやリスクを大幅に抑えられます。いつでもリリースできる状態を保つことによって、リリース直前に行う作業がほとんどなくなるためです。理想的なCDを実践できれば、リリースはボタンひとつでできるようになります。また、デプロイメントパイプラインによってコード変更からリリースまでのフローが可視化されるので、どの部分で問題が起きているか、ボトルネックになっているかといったことが可視化されます。これによって、リリースまでのフローの改善にもつながりやすいです。

世の中にはCI/CDを実現するためのツールがいくつも存在します。代表的なのは Jenkins[6]です。Jenkinsは OSSで手軽に各自の環境に構築することが可能で、かなり長い期間に渡ってCI/CD界隈のデファクトツールとして存在し続けています。その後、Travis CI[7]やCircleCI[8]などの登場によって、CI/CDのクラウドサービス化が進みました。開発者は、それまでよりも素早くお手軽にCI/CDを導入できるようになったのです。そして、継続的デリバリーの普及に伴い、デプロイメントパイプラインのような複数のジョブによるフローがワークフローとして表現されるようになりました。さらに、Dockerによりコンテナ技術が一般的になり、CI/CDツールでもコンテナ技術が取り入れられました。

GitHub Actionsは、こういったCI/CDツールの長い歴史の流れを汲む、最新のCI/CDツールなのです。

6:https://jenkins.io/
7:https://travis-ci.org/
8:https://circleci.com/

1.2 主な特徴

GitHub Actionsは、既存のCI/CDツールとどういった点が異なるのでしょうか？

まず、GitHubのファーストパーティ製のサービスであるということが挙げられます。このため、GitHubの権限を外に持ち出すことなくCI/CDを実現することができます。

また、GitHubのさまざまなWebhook[9]のイベントに対応しているというのも、これまでのCI/CDツールとは大きく異なる点です。従来のCI/CDツールはpushイベントによるCI/CDがメインで、それ以外のイベントはほとんどサポートされていませんでした。そのため、例えばissueが登録されたときになにかしらの処理を自動で実行したい場合、これまでは自前でAWS上にAPI Gateway + Lambdaを構築したりするなど、どうにかしてイベント発生時に処理を実行する仕組みを作るしかありませんでした。しかし、GitHub Actionsではpush以外の様々なイベント発生時にもワークフローを自動で実行できるので、イベントにフックするワークフローもGitHub内で完結して実行できるようになりました。つまり、GitHub ActionsはCI/CD用途に限らず、開発における様々な場面におけるワークフローを自動化できるということです。GitHub Actionsが対応しているイベントについては「2.7 イベントとアクティビティ」で解説します。

そして、パブリックリポジトリーは完全無料というのも大きな特徴です。既存のCI/CDサービスでもOSS向けに無料プランはこれまでもあったのですが、実行時間や並列実行数などの制限があり、ある程度の規模になってくると物足りない面がありました。しかし、GitHub Actionsはパブリックリポジトリーならどれだけの時間使っても完全に無料で、さらに最大20並列まで実行できるので、かなりの規模のOSS開発でも実用的です。

さらに、GitHub Actionsにはセルフホストランナーという、独自の環境でワークフローを実行するための仕組みがあります。GitHub Actionsが管理する環境でも、Linux、Windows、macOSと多様な環境でのワークフロー実行が可能です。しかし、より強いマシンリソースを利用したいとか、独自にカスタマイズした環境でビルドしたいとか、外部から接続できないネットワーク内にデプロイしたいといった要求がある場合には、用意された環境だけでは実現できないことがあります。セルフホストランナーの仕組みは、そういったユースケースを可能にしてくれます。セルフホストランナーの詳細については、「2.12 ランナーのセルフホスティング」で解説します。

もちろん、これまでのCI/CDツールが持っていた機能についても、GitHub Actionsは大部分をサポートしています。公開後もどんどん改善が進んでおり、GitHubの大きなコミュニティが GitHub Actionsのエコシステムにも影響を与えることも予想され、今後の発展性も期待できます。

1.3 料金体系

前の節でも少し触れましたが、GitHub Actionsの**料金体系**についてもう少し詳しく見ておきます。

1.3.1 パブリックリポジトリー

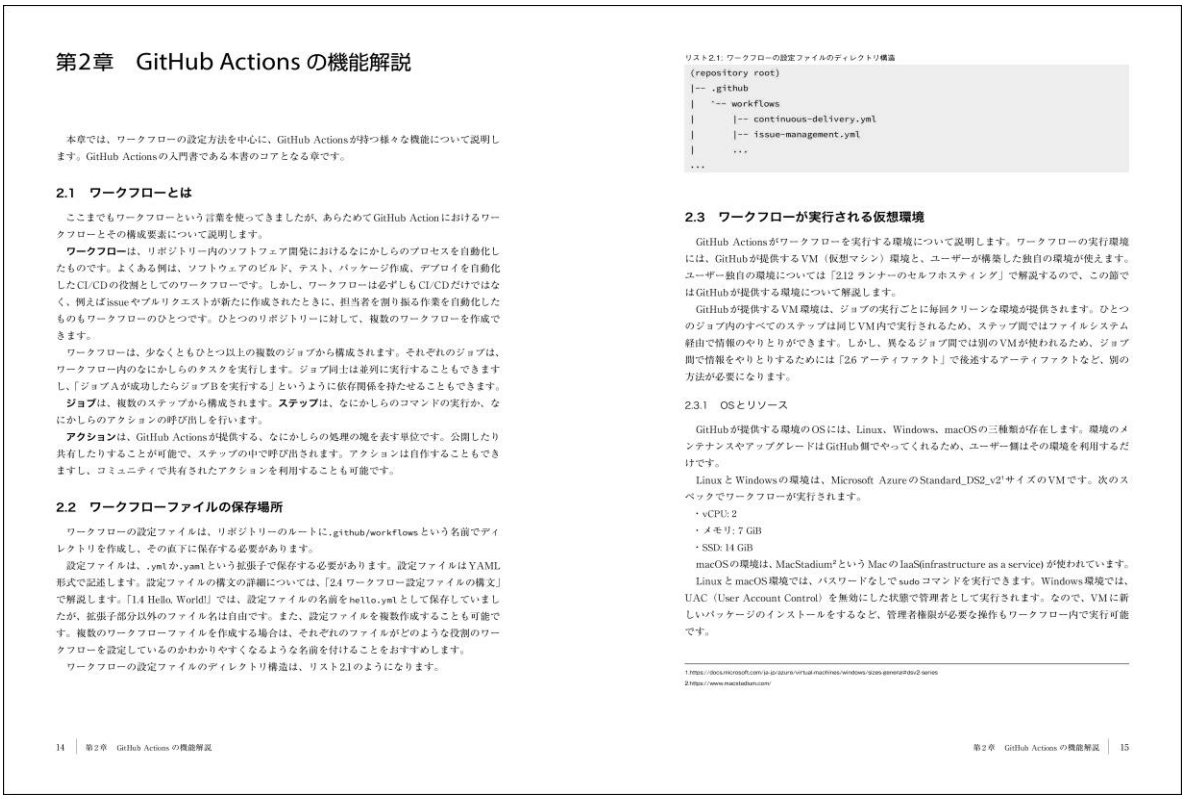
GitHub Actionsは、パブリックリポジトリーでの利用は完全に無料です。

1.3.2 プライベートリポジトリー

プライベートリポジトリーでGitHub Actionsを利用する場合は、GitHubのプランによって決まった量の無料利用できるビルド時間とストレージが提供されます。無料分を超えた分については料金がかかります。

9:Hub アプリのなにかしらのイベント発生時に、なにかしらの処理を実行する仕組み

GitHub Actions の各機能を解説



第2章 GitHub Actions の機能解説

本章では、ワークフローの設定方法を中心に、GitHub Actionsが持つ様々な機能について説明します。GitHub Actionsの入門書である本書のコアとなる章です。

2.1 ワークフローとは

ここまでもワークフローという言葉を使ってきましたが、あらためてGitHub Actionにおけるワークフローとその構成要素について説明します。

ワークフローは、リポジトリー内のソフトウェア開発におけるなにかしらのプロセスを自動化したものです。よくある例は、ソフトウェアのビルド、テスト、パッケージ作成、デプロイを自動化したCI/CDの役割としてのワークフローです。しかし、ワークフローは必ずしもCI/CDだけではなく、例えばissueやプルリクエストが新たに作成されたときに、担当者を割り振る作業を自動化したものもワークフローのひとつです。ひとつのリポジトリーに対して、複数のワークフローを作成できます。

ワークフローは、少なくともひとつ以上の複数のジョブから構成されます。それぞれのジョブは、ワークフロー内のなにかしらのタスクを実行します。ジョブ同士は並列に実行することもできますし、「ジョブAが成功したらジョブBを実行する」というように依存関係を持たせることもできます。

ジョブは、複数のステップから構成されます。**ステップ**は、なにかしらのコマンドの実行か、なにかしらのアクションの呼び出しを行います。

アクションは、GitHub Actionsが提供する、なにかしらの処理の塊を表す単位です。公開したり共有したりすることが可能で、ステップの中で呼び出されます。アクションは自作することもできますし、コミュニティで共有されたアクションを利用することも可能です。

2.2 ワークフローファイルの保存場所

ワークフローの設定ファイルは、リポジトリーのルートに`.github/workflows`という名前でディレクトリを作成し、その直下に保存する必要があります。

設定ファイルは、`.yml`か`.yaml`という拡張子で保存する必要があります。設定ファイルはYAML形式で記述します。設定ファイルの構文の詳細については、「2.4 ワークフロー設定ファイルの構文」で解説します。「1.4 Hello, World!」では、設定ファイルの名前を`hello.yml`として保存していましたが、拡張子部分以外のファイル名は自由です。また、設定ファイルを複数作成することも可能です。複数のワークフローファイルを作成する場合は、それぞれのファイルがどのような役割のワークフローを設定しているのかわかりやすくなるような名前を付けることをおすすめします。

ワークフローの設定ファイルのディレクトリ構造は、リスト2.1のようになります。

リスト2.1: ワークフローの設定ファイルのディレクトリ構造

```
(repository root)
|-- .github
|   `-- workflows
|       |-- continuous-delivery.yml
|       |-- issue-management.yml
|       ...
...
```

2.3 ワークフローが実行される仮想環境

GitHub Actionsがワークフローを実行する環境について説明します。ワークフローの実行環境には、GitHubが提供するVM（仮想マシン）環境と、ユーザーが構築した独自の環境が使えます。ユーザー独自の環境については「2.12 ランナーのセルフホスティング」で解説するので、この節ではGitHubが提供する環境について解説します。

GitHubが提供するVM環境は、ジョブの実行ごとに毎回クリーンな環境が提供されます。ひとつのジョブ内のすべてのステップは同じVM内で実行されるため、ステップ間ではファイルシステム経由で情報のやりとりができます。しかし、異なるジョブ間では別のVMが使われるため、ジョブ間で情報をやりとりするためには「2.6 アーティファクト」で後述するアーティファクトなど、別の方法が必要になります。

2.3.1 OSとリソース

GitHubが提供する環境のOSには、Linux、Windows、macOSの三種類が存在します。環境のメンテナンスやアップグレードはGitHub側でやってくれるため、ユーザー側はその環境を利用するだけです。

LinuxとWindowsの環境は、Microsoft AzureのStandard_DS2_v2[1]サイズのVMです。次のスペックでワークフローが実行されます。

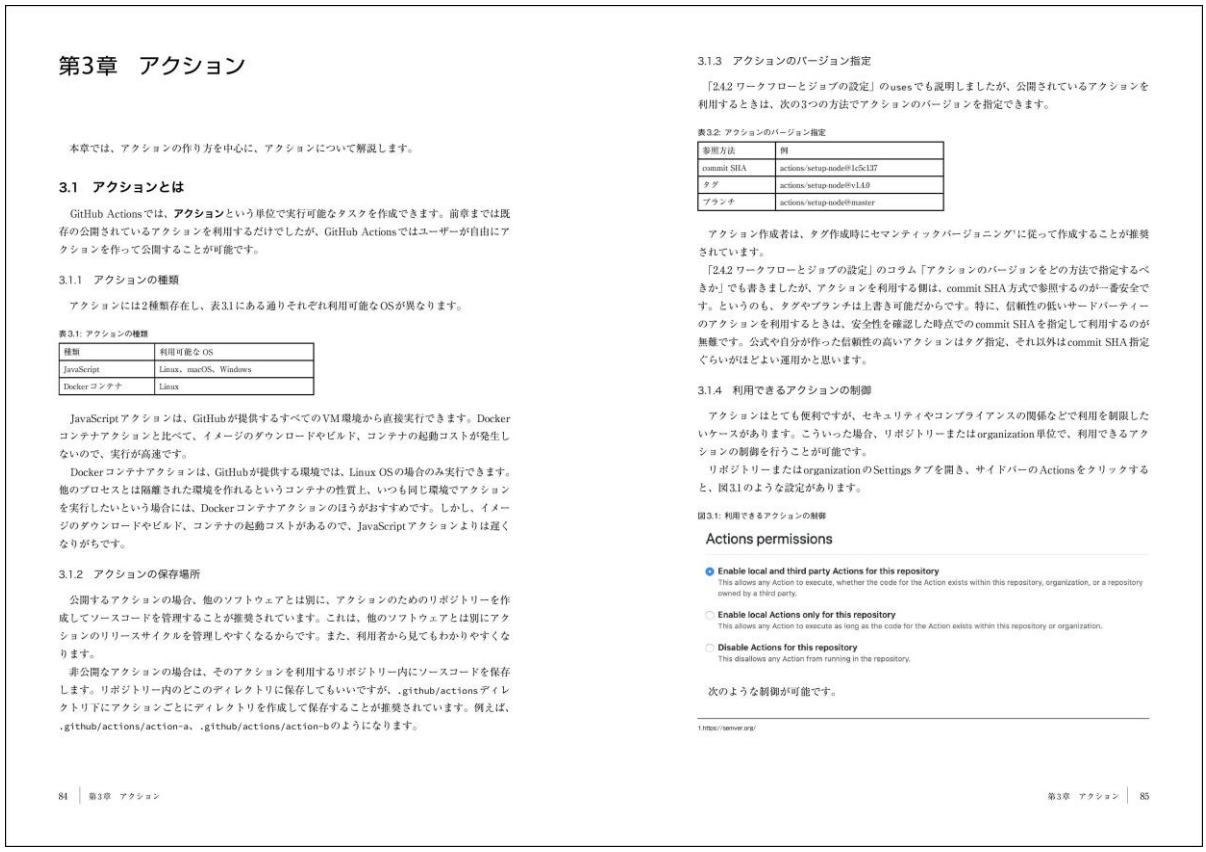
- vCPU: 2
- メモリ: 7 GiB
- SSD: 14 GiB

macOSの環境は、MacStadium[2]というMacのIaaS(infrastructure as a service)が使われています。

LinuxとmacOS環境では、パスワードなしで`sudo`コマンドを実行できます。Windows環境では、UAC（User Account Control）を無効にした状態で管理者として実行されます。なので、VMに新しいパッケージのインストールをするなど、管理者権限が必要な操作もワークフロー内で実行可能です。

1:https://docs.microsoft.com/ja-jp/azure/virtual-machines/windows/sizes-general#dsv2-series
2:https://www.macstadium.com/

アクションの作り方を中心に解説



第3章　アクション

本章では、アクションの作り方を中心に、アクションについて解説します。

3.1　アクションとは

GitHub Actionsでは、**アクション**という単位で実行可能なタスクを作成できます。前章までは既存の公開されているアクションを利用するだけでしたが、GitHub Actionsではユーザーが自由にアクションを作って公開することが可能です。

3.1.1　アクションの種類

アクションには2種類存在し、表3.1にある通りそれぞれ利用可能なOSが異なります。

表3.1: アクションの種類

種類	利用可能なOS
JavaScript	Linux、macOS、Windows
Dockerコンテナ	Linux

JavaScriptアクションは、GitHubが提供するすべてのVM環境から直接実行できます。Dockerコンテナアクションと比べて、イメージのダウンロードやビルド、コンテナの起動コストが発生しないので、実行が高速です。

Dockerコンテナアクションは、GitHubが提供する環境では、Linux OSの場合のみ実行できます。他のプロセスとは隔離された環境を作れるというコンテナの性質上、いつも同じ環境でアクションを実行したいという場合には、Dockerコンテナアクションのほうがおすすめです。しかし、イメージのダウンロードやビルド、コンテナの起動コストがあるので、JavaScriptアクションよりは遅くなりがちです。

3.1.2　アクションの保存場所

公開するアクションの場合、他のソフトウェアとは別に、アクションのためのリポジトリーを作成してソースコードを管理することが推奨されています。これは、他のソフトウェアとは別にアクションのリリースサイクルを管理しやすくなるからです。また、利用者から見てもわかりやすくなります。

非公開なアクションの場合は、そのアクションを利用するリポジトリー内にソースコードを保存します。リポジトリー内のどこのディレクトリに保存してもいいですが、`.github/actions`ディレクトリ下にアクションごとにディレクトリを作成して保存することが推奨されています。例えば、`.github/actions/action-a`、`.github/actions/action-b`のようになります。

3.1.3　アクションのバージョン指定

「2.4.2 ワークフローとジョブの設定」のusesでも説明しましたが、公開されているアクションを利用するときは、次の3つの方法でアクションのバージョンを指定できます。

表3.2: アクションのバージョン指定

参照方法	例
commit SHA	actions/setup-node@1e5e137
タグ	actions/setup-node@v1.4.0
ブランチ	actions/setup-node@master

アクション作成者は、タグ作成時にセマンティックバージョニング[1]に従って作成することが推奨されています。

「2.4.2 ワークフローとジョブの設定」のコラム「アクションのバージョンをどの方法で指定するべきか」でも書きましたが、アクションを利用する側は、commit SHA方式で参照するのが一番安全です。というのも、タグやブランチは上書き可能だからです。特に、信頼性の低いサードパーティーのアクションを利用するときは、安全性を確認した時点でのcommit SHAを指定して利用するのが無難です。公式や自分が作った信頼性の高いアクションはタグ指定、それ以外はcommit SHA指定ぐらいがほどよい運用かと思います。

3.1.4　利用できるアクションの制御

アクションはとても便利ですが、セキュリティやコンプライアンスの関係などで利用を制限したいケースがあります。こういった場合、リポジトリーまたはorganization単位で、利用できるアクションの制御を行うことが可能です。

リポジトリーまたはorganizationのSettingsタブを開き、サイドバーのActionsをクリックすると、図3.1のような設定があります。

図3.1: 利用できるアクションの制御

Actions permissions

Enable local and third party Actions for this repository
This allows any Action to execute, whether the code for the Action exists within this repository, organization, or a repository owned by a third party.

Enable local Actions only for this repository
This allows any Action to execute as long as the code for the Action exists within this repository or organization.

Disable Actions for this repository
This disallows any Action from running in the repository.

次のような制御が可能です。

1 https://semver.org/

＜＜目次＞＞

第 1 章　GitHub Actions　の基礎知識
第 2 章　GitHub Actions　の機能解説
第 3 章　アクション
第 4 章　サンプルレシピ
付録 A　コミュニティ

＜＜著者紹介＞＞

宮田 淳平

2009 年にサイボウズ株式会社へ入社。大規模向けグループウェア『ガルーン』やビジネスアプリ作成プラットフォーム『kintone』の開発を経験した後、生産性向上チームを立ち上げ、組織横断で開発基盤の整備と自動化の推進を行う。GitHub Actions Meetup Tokyo コミュニティを運営。

＜＜販売ストア＞＞

電子書籍:

Amazon Kindle ストア、楽天 kobo イーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER

印刷書籍:

Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス

※　各ストアでの販売は準備が整いしだい開始されます。

※　全国の一般書店からもご注文いただけます。

【インプレス R&D】 https://nextpublishing.jp/

株式会社インプレス R&D（本社：東京都千代田区、代表取締役社長：井芹昌信）は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishingを使った「インターネット白書」の出版など IT 関連メディア事業を展開しています。

※NextPublishing は、インプレス R&D が開発した電子出版プラットフォーム（またはメソッド）の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド（POD）による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 https://www.impressholdings.com/

株式会社インプレスホールディングス（本社：東京都千代田区、代表取締役：唐島夏生、証券コード：東証 1 部 9479）を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「旅・鉄道」「学術・理工学」を主要テーマに専門性の高いメデイア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】

株式会社インプレス R&D NextPublishing センター
TEL 03-6837-4820
電子メール: np-info@impress.co.jp