

# CircleCI 実践入門

CI/CDがもたらす開発速度と品質の両立

Urai Masato Orake Tomoya Kim Hirokuni  
浦井誠人 / 大竹智也 / 金 洋国

[著]

技術評論社

本書は、小社刊の以下の刊行物をもとに、大幅に加筆と修正を行い書籍化したものです。

- 『WEB+DB PRESS』Vol.107特集1  
「実践 CircleCI——ワークフローで複雑なCI/CDを自動化」

本書の内容に基づく運用結果について、著者、ソフトウェアの開発元および提供元、株式会社技術評論社は一切の責任を負いかねますので、あらかじめご了承ください。

本書に記載されている情報は、特に断りがない限り、執筆時点(2020年)の情報に基づいています。ご使用時には変更されている可能性がありますのでご注意ください。

本書で利用しているクラウドサービスなど、一部有償のものが含まれます。詳細はそれぞれのサービスのWebサイトなどご確認ください。

本書に記載されている会社名・製品名は、一般に各社の登録商標または商標です。本書中では、™、©、®マークなどは表示していません。

## はじめに

---

本書の目的は、CircleCIを通してCI/CDに入門することです。CI/CDを使いこなすことができれば、開発／運用効率だけではなく品質と開発速度もアップし、あなたのチームは一段上のソフトウェア開発ができるようになるでしょう。本書がその足掛かりになれば幸いです。

今後さらなるスピードが求められる世界では、すばやい開発をしながら失敗を少なくするためにCI/CDの導入は欠かすことができません。今はCI/CDがなくても問題ないかもしれませんが、数年後には最低限のCI/CDの導入は必須となるでしょう。それはちょうど、Gitが今の開発現場においてなくてはならないような存在になったことと似ているかもしれません。

CircleCIはCI/CDの分野をリードするサービスです。今この文章の筆をとっている筆者の一人は、CircleCIで開発者として5年以上働いており、CircleCIの成長を間近で見してきました。たくさんのCI/CDサービスが出ては消える中、CircleCIは堅調に成長を続けてきました。それは、CircleCI自体がCI/CDの考え方を尊重し、日々の開発に取り入れてきたからだだと筆者は考えています。システムの複雑性を考慮し、自分たちの限界を受け入れ、安全に失敗し続けるCI/CDのエッセンスを身をもって実践してきた結果だと思っています。

しかし、本書はCircleCI社によって書かれた本ではありません。CircleCIを毎日使い、CI/CDの力を信じるユーザーによって書かれた本です。CircleCIに限らず、できるだけほかのサービスやツールでも応用が効くようにCI/CDに関する重要な考え方やTipsを盛り込んだつもりです。本書をきっかけとして、CI/CDの重要性と可能性について考えてもらえることを願っています。

執筆者を代表して 2020年8月 金 洋国

## 謝辞

---

コロナ禍で自身の身の回りの環境が大きく変化し、一時は執筆が手につかなくなることもありました。そんな中でも、共著者である大竹さんと金さんのおかげで、とても素晴らしい書籍ができました。ありがとうございます。

読者の皆さまへ。手にとっていただき、ありがとうございます。本書はCircleCIの基礎の基礎からOrbsの作り方といった応用までじっくり解説した書籍です。本書とCircleCIを通して、CI/CDに触れることで、CI/CDで効率化することの楽しさとCI/CDのできるものの可能性を感じていただけると嬉しいです。

浦井 誠人

今回は自身初めての共著となりましたが、快く誘いを引き受けてくれた金さん、浦井さんの両名に深く感謝します。おかげでひとりでは書けない素晴らしい内容の書籍になりました。また、コロナ禍で保育園が休園になる中、惜しみなく協力してくれた愛する映子とまりあと、内容はわからないけれども出版を楽しみにしてくれている両親、そして本書を手にとってくださったあなたに感謝します。

大竹 智也

業務で忙しいにもかかわらず本書のレビューを快諾してくれたCircleCI合同会社の車井さんと伊藤さん、本当にありがとうございました。お二人のおかげで、筆者たちがあまり詳しくない内容も丁寧に解説できました。

感謝の気持ちをいつもうまく伝えられない妻にも、この場を借りて感謝の言葉を伝えさせてください。毎日の夕食後の執筆時間があつたおかげで本書を完成させることができました。いつも仕事ばかりの自分を支えてくれてありがとう。

一緒に執筆した大竹さんと浦井さん、お疲れさまでした。新型コロナウイルスの影響により、執筆時間が大きく削られ一時は未完成になってしまうことを危惧しました。そんな困難な時期を乗り越え、誰一人当初のやる気を失わず本書を完成できたことは僕の誇りです。

金 洋国

## 本書の読み方

---

- 文章やコードを利用させていただいているサイト

解説を行ううえで、以下に挙げるサイトの文章や図、コードを許諾を受けたうえで一部利用させていただいております。

- **Contents and diagrams in CircleCI Docs** (<https://circleci.com/docs/>)
- **Code samples in CircleCI Public** (<https://github.com/CircleCI-Public>)

- 各章の執筆担当者と内容

各章の執筆担当者は以下のとおりです。

章	タイトル	著者名
第1章	なぜCI/CDが必要か	金 洋国
第2章	CircleCIの基本	浦井 誠人
第3章	環境構築	大竹 智也
第4章	ワークフローでジョブを組み合わせる	浦井 誠人
第5章	実践的な活用方法	大竹 智也
第6章	テストの基本と最適化	大竹 智也
第7章	継続的デプロイの実践	金 洋国
第8章	Webアプリケーション開発、インフラでの活用	金 洋国、浦井 誠人
第9章	モバイルアプリ開発での活用	金 洋国
第10章	デスクトップ/ネイティブアプリ開発での活用	浦井 誠人
第11章	さまざまなタスクの自動化	浦井 誠人
第12章	Orbsの作成	浦井 誠人
Appendix	config.ymlの基本構造	浦井 誠人

第1章ではCI/CDとは切り離すことができないアジャイルとDevOpsが発展してきた経緯を解説しながら、なぜCI/CDが今後のソフトウェア開発に欠かせないかについて解説しています。

第2章から第5章まではCircleCIの基礎と応用について詳しく解説しています。ここまで読めばCircleCIの基本的な機能を使いこなすことができるでしょう。

第6章はキャッシュや並列処理を用いてテストの実行を最適化する方法について解説しています。CircleCIを使いこなせるようになってくると、もっと効率化したくなってくるはずですよ。第6章はそんなときに役立つ方法をたくさん

解説しています。

第7章は継続的デプロイメント、いわゆるCDについてです。前半はCDの必要性や可能性についての概論を解説し、後半ではAWSを使って実際に手を動かしながらCDの導入をしていきます。

第8章から10章までは実践編です。各言語やフレームワークでCircleCIを導入するときに参考になる情報やTipsを掲載しています。実際の業務でCircleCIを導入する際にきつと役立つでしょう。すべてを読む必要はなく、自分に関係性が低い章はスキップしても大丈夫です。

第11章ではCI/CD以外にもCircleCIを使ってさまざまなタスクを自動化する方法を解説しています。

第12章ではCircleCIをより便利に使うことができるOrbsの作り方を解説しています。Orbsを使えば複数のプロジェクトで共通する処理をまとめるCircleCI専用のライブラリのようなものを作成して公開できるようになります。

巻末にあるAppendixにはCircleCIの設定ファイルに登場する要素の詳細な説明と使い方を掲載しています。

#### ●—— diff表記の見方

本書では、コードの変更内容をdiff表記(ユニファイド形式)により表現しています。

```
--- a/.circleci/config.yml ❶
+++ b/.circleci/config.yml ❷
@@ -2,7 +2,10 @@ version: 2 ❸
 jobs:
   build:
     docker:
       - image: <Docker Hubアカウント名>/circleci-node-aws-cli
       + image: <AWSアカウントID>.dkr.ecr.us-east-1.amazonaws.com/circleci-node-aws-cli:latest
       + aws_auth:
       +   aws_access_key_id: <AWSアクセスキーID>
       +   aws_secret_access_key: <AWSシークレットアクセスキー>
     steps:
       - run:
         name: AWS CLIのバージョン確認
```

❶と❷は比較しているファイルです。この例では同じファイルを比較していて、変更前と変更後の差分を表示しています。❸の@@ -2,7 +2,10 @@は4行目以下に表示している内容の開始行番号と行数です。-2,7は変更前ファイルの2行目から7行、+2,10は変更後ファイルの2行目から10行を表示しているという意味になります。version: 2は表示開始行の直前にある共通内容を表していま

す。

4行目からは実際の差分表示になっていて、行頭に-が書かれている行は削除、+が書かれている行は追加を表しています。

●—— **正誤情報とサンプルコード**

正誤情報は、以下の本書サポートページに掲載します。

<https://gihyo.jp/book/2020/978-4-297-11411-4/support>

本書で利用しているサンプルコードは以下のサイトで公開しています。

<https://github.com/circleci-book>

各章の利用箇所では、リポジトリのURLを示しています。

はじめに.....	iii
謝辞.....	iv
本書の読み方.....	v
目次.....	viii

## 第1章

### なぜCI/CDが必要か..... 1

#### 1.1 アジャイルとDevOps..... 2

アジャイルとは?.....	2
DevOpsとは?.....	3
DevOpsの導入効果.....	3
CI/CDとの関係.....	4
自動化の必要性.....	4
誰がCI/CD環境を用意するか.....	4
CI/CD導入後の運用チームの役割.....	5

#### 1.2 自動化で品質と開発速度をアップ..... 5

CI/CDで自動化できること.....	5
ビルド.....	6
テスト.....	6
デプロイ.....	6
その他さまざまなタスク.....	7
自動化することで得られる効果.....	7
テストし忘れの防止.....	7
テストに対する信頼性の向上.....	7
積極的な機能リリース.....	8
品質と開発速度の向上.....	8
DevOpsの効果を測る4つの指標.....	8
a リードタイム.....	9
b デプロイ頻度.....	9
c 平均修復時間(MTTR).....	9
d 失敗の頻度.....	10
品質と速度の両立性.....	10

#### 1.3 CI/CDツールの選び方..... 10

オンプレミス vs. SaaS.....	10
----------------------	----



オンプレミス型のメリット/デメリット .....	11
SaaS型のメリット/デメリット .....	11
どちらを選べばよいか? .....	11
<b>代表的なツール/サービスの紹介 .....</b>	<b>12</b>
CircleCI .....	12
Travis CI .....	12
Jenkins .....	12
AWS CodeBuild .....	13
GitHub Actions .....	13
<b>CircleCIの特徴 .....</b>	<b>13</b>
Dockerのサポート .....	13
多様なプログラミング言語に対応 .....	13
ワークフロー(パイプライン) .....	14
リソースクラス .....	14
従量課金プラン .....	14
Column Dockerについて .....	15

## 第2章

### CircleCIの基本 .....

<b>2.1 CircleCIの動作フロー .....</b>	<b>18</b>
ジョブの開始から完了まで .....	18
ジョブが失敗した場合 .....	19
<b>2.2 CircleCIの基本 .....</b>	<b>19</b>
ビルド——アプリケーションの構築 .....	19
設定ファイル——コード化された設定 .....	19
プロジェクト——コードホスティングサービスにおけるリポジトリ .....	20
ステップ——ジョブの設定の最小単位 .....	21
Column コンビニエンスイメージ .....	21
runステップ .....	22
ビルトインステップ .....	22
ジョブ——ステップのグループ化 .....	22
Executor——ジョブの実行環境 .....	22
Docker Executor .....	22
Machine Executor .....	23
macOS Executor .....	23
Windows Executor .....	23
Column Docker in Docker問題 .....	24
ワークフロー——ジョブ実行順序の制御 .....	25

ワークスペース/キャッシュ/アーティファクト——データの永続化.....	25
ワークスペース.....	25
キャッシュ.....	26
アーティファクト.....	27
パイプライン——ワークフローのグループ化.....	27
パイプラインとは何か.....	27
パイプラインの活用.....	28
Orbs——ジョブ設定の再利用.....	28
Orbsのしくみ.....	28
Orb Registry.....	30
料金体系——従量課金とOSSプラン.....	30
従量課金プラン.....	30
クレジット.....	31
有料オプション.....	31
シート料金.....	31
アクティブユーザー.....	31
注意点.....	32
OSSプラン.....	33

## 2.3 **YAMLの基本** 33

Column サポート体制.....	33
リスト.....	34
マップ.....	34
スカラ.....	35
複数行の記述.....	36
アンカーとエイリアス.....	36

# 第3章

## 環境構築..... 39

### 3.1 **GitHubとの連携** 40

GitHubアカウントの連携.....	40
プロジェクトの追加.....	41
個人リポジトリの場合.....	41
GitHub Organizationのリポジトリの場合.....	42
プライベートリポジトリを追加した場合のアクセス権.....	42
個人リポジトリの場合.....	43
GitHub Organizationのリポジトリの場合.....	43

### 3.2 **CircleCIの実行環境** 43

	クラウド環境での実行 .....	43
	ローカル環境での実行 .....	44
<b>3.3</b>	<b>ローカル環境での初めてのジョブ実行</b> .....	<b>44</b>
	CircleCI CLIのインストール .....	44
	config.ymlの作成 .....	45
	config.ymlのバリデーション .....	46
	ジョブの実行 .....	47
<b>3.4</b>	<b>クラウド環境での初めてのジョブ実行</b> .....	<b>48</b>
	config.ymlの自動追加 .....	49
	config.ymlの手動追加 .....	49
<b>3.5</b>	<b>プロジェクト追加後の通常のジョブ実行</b> .....	<b>51</b>
	失敗したジョブの修正 .....	51
	実行のスキップ .....	52
	再実行 .....	52
	キャンセル .....	53
<b>3.6</b>	<b>SSHによる失敗したテストのデバッグ</b> .....	<b>54</b>
	SSHデバッグでできること .....	54
	SSHデバッグの実行 .....	54
	開始 .....	55
	終了 .....	57
<b>3.7</b>	<b>サンプルコードでCI環境構築を実践</b> .....	<b>57</b>
	実装とテストコード .....	57
	config.ymlの作成とローカル環境実行 .....	59
	CircleCIでジョブ実行 .....	60
	SSHデバッグによるテスト失敗の原因調査 .....	61
	CircleCIで失敗するコードの追加 .....	61
	SSHデバッグでテストの成功確認 .....	62
	config.ymlを修正してテストの成功確認 .....	63

## 第4章

### ワークフローでジョブを組み合わせる .....

<b>4.1</b>	<b>ワークフローとは</b> .....	<b>66</b>
	ワークフローでできること .....	66
	ジョブのオーケストレーションの種類 .....	67

	シーケンシャルジョブ .....	67
	ファンアウト/ファンイン .....	67
<b>4.2</b>	<b>ワークフローの基本的な使い方</b> .....	<b>68</b>
	ワークフローに対応するconfig.yml .....	68
	ワークフローの実行 .....	68
	ワークフローのステータス .....	69
	失敗したワークフローの再実行 .....	69
<b>4.3</b>	<b>ジョブの分割</b> .....	<b>70</b>
	ジョブを分割するメリット .....	70
	再利用可能なコンフィグを使ってジョブを分割 .....	71
	executorsキー .....	71
	commandsキー .....	73
	再利用可能なコンフィグのパラメータ .....	73
<b>4.4</b>	<b>複数ジョブの同時実行</b> .....	<b>74</b>
	同時実行するメリット .....	74
	requiresキーでジョブ間の依存関係を制御 .....	74
<b>4.5</b>	<b>ワークスペースによるジョブ間のファイル共有</b> .....	<b>75</b>
	ワークスペースの利用方法 .....	75
	persist_to_workspace .....	75
	attach_workspace .....	76
	利用できないジョブ .....	76
	ワークスペースのライフサイクル .....	78
	キャッシュとの違い .....	78
	証明書エラーへの対応 .....	79
<b>4.6</b>	<b>そのほかのワークフロー</b> .....	<b>79</b>
	フィルタリング .....	79
	フィルタリングのしくみ .....	79
	タグによるフィルタリング .....	79
	ブランチによるフィルタリング .....	80
	スケジュール .....	81
	スケジュール実行のしくみ .....	81
	cronキー利用時の注意点 .....	82
	承認ジョブ .....	82
	ワークフローをコントロール .....	83
	具体的な設定例 .....	83

## 第5章

## 実践的な活用方法..... 85

5.1	<b>プロジェクト設定によるジョブの実行タイミングの調整</b>	86
	フォークされたリポジトリのビルド.....	86
	プルリクエストのみのビルド.....	87
	自動キャンセルによる最新のコミットのみのビルド.....	87
5.2	<b>GitHubのブランチプロテクションによるマージのブロック</b>	88
	ブランチプロテクションでできること.....	88
	CIステータスによるマージのブロック.....	89
5.3	<b>CircleCI Checksによる詳細なCIステータスの取得</b>	89
	CircleCI Checksでできること.....	89
	CircleCI Checksの導入.....	91
	有効化.....	91
	無効化.....	93
	ブランチプロテクションの詳細設定.....	93
5.4	<b>環境変数を利用する理由</b>	94
	パスワードやAPIキーなどの秘匿情報の保護.....	94
	アプリケーション設定とコードの分離.....	95
5.5	<b>ビルトイン環境変数</b>	96
	ビルトイン環境変数を利用する理由.....	96
	主なビルトイン環境変数一覧.....	96
	ビルトイン環境変数の利用.....	97
5.6	<b>ユーザー定義の環境変数</b>	98
	ユーザー定義の環境変数を利用する理由.....	98
	インライン環境変数の利用.....	99
	ステップ.....	99
	ジョブ.....	100
	イメージ.....	101
	プロジェクト設定の利用.....	102
	コンテキストの利用.....	103
	コンテキストの設定.....	104
	コンテキスト環境変数の利用.....	105
	複数行の環境変数を利用する方法.....	106
	セキュリティ.....	107

環境変数の出力 .....	107
SSH接続による環境変数の出力 .....	109
Column コマンドのパス(PATH)を通すには? .....	109

## 5.7 通知の活用 110

通知の設定 .....	110
Slackへの通知(Webhook URLの取得) .....	111
Slackへの通知(Slack Orb) .....	113
Slack通知の調整 .....	114
GitHubへのコメント .....	114
ステータスバッジ .....	116
テンプレートコード .....	117
プライベートリポジトリの場合 .....	117

## 5.8 SSHキーの活用 119

SSHキー登録のしくみ .....	119
ユーザーキーとデプロイキー .....	120
ベストプラクティス .....	122
デプロイキーの使い方 .....	123
追加 .....	123
利用 .....	125
同一ホストの複数デプロイキー .....	126
add_ssh_keysの複数回実行 .....	127
環境変数の利用 .....	127
ユーザーキーの使い方 .....	128
追加 .....	129
利用 .....	131

# 第6章

## テストの基本と最適化 133

### 6.1 基本的なテストの実行方法 134

最小構成のテスト .....	134
設定 .....	134
解説 .....	136
データベースを使ったテスト .....	136
設定 .....	137
解説 .....	138
ブラウザを使ったテスト .....	139
設定 .....	139

解説.....	140
<b>CircleCIでテストを実行する際の注意点</b> .....	141
並列実行数.....	141
メモリ量.....	141
<b>6.2 CI実行速度の改善</b> .....	142
<b>CIを改善するタイミング</b> .....	142
<b>実行時間の改善方法</b> .....	143
複数ジョブの同時実行.....	143
キャッシュ.....	144
ジョブ内の並列実行.....	144
リソースクラスの変更.....	145
<b>改善方法の決定方針</b> .....	145
<b>6.3 キャッシュの活用</b> .....	146
<b>キャッシュの種類と特徴</b> .....	147
ファイルキャッシュ.....	147
Dockerイメージキャッシュ.....	147
<b>ファイルキャッシュの活用方法</b> .....	148
依存パッケージのキャッシュ.....	148
キャッシュのクリア.....	150
Column キャッシュを削除できない理由——不変性とべき等性.....	151
部分キャッシュリストア.....	152
適切なキャッシュキーの設計.....	153
<b>Dockerイメージキャッシュの活用方法</b> .....	155
Dockerイメージとレイヤ構造.....	155
DLCのしくみ.....	157
DLCの利用.....	157
<b>6.4 最適化済みDockerイメージの活用</b> .....	158
<b>CI用のDockerイメージを用意するメリット</b> .....	158
<b>イメージの取得</b> .....	158
Docker Hubからの取得.....	158
ECRからの取得.....	160
<b>6.5 テストサマリーでテスト結果をわかりやすく表示する</b> .....	161
<b>テストサマリーを利用する目的</b> .....	161
<b>エラーレポートの確認</b> .....	161
<b>テストサマリーの活用方法</b> .....	162
サポートされているレポートフォーマット.....	162
さまざまなツールによるレポートファイル出力.....	163
レポートファイルの保存.....	164

<b>6.6</b>	<b>ジョブ内並列実行の活用</b>	165
	並列実行のしくみ.....	165
	テスト分割コマンドの使い方.....	166
	タイミングデータを利用したテストの分割.....	168
	並列実行の利用方法.....	168
<b>6.7</b>	<b>リソースクラスを活用し、ジョブ実行環境の性能を変更</b>	170
	リソースクラスとは?.....	170
	リソースクラスの利用方法.....	170
	種類と選択方針.....	170
	resource_classキーの利用.....	171

## 第7章

### 継続的デプロイの実践..... 173

<b>7.1</b>	<b>継続的デプロイ</b>	174
	継続的デリバリとの違い.....	174
	広義の継続的デリバリ.....	175
<b>7.2</b>	<b>なぜ継続的デプロイを行うのか</b>	176
	本番環境によるテスト.....	176
	テスト環境でのQAの限界.....	176
	実際の失敗事例.....	176
	何が問題なのか?.....	177
	本番環境でのテスト.....	177
	フィードバックループの構築.....	177
	フィードバックループとは何か?.....	178
	フィードバックループの重要性.....	178
	継続的デプロイとフィードバックループの関係.....	179
<b>7.3</b>	<b>継続的デプロイの難しさ</b>	179
	組織的な理由.....	179
	ビジネス的な理由.....	180
<b>7.4</b>	<b>継続的デプロイの導入を助ける手法</b>	180
	承認ジョブによる承認フローへの対応.....	180
	承認ジョブの設定例.....	180
	承認ジョブの注意点.....	181
	フィーチャーフラグによる段階的リリース.....	181
	フィーチャーフラグの使用例.....	181



Column	デプロイとリリースは同じ?	182
	フィーチャーフラグの導入方法	183
	<b>新規プロジェクトからの導入</b>	183
	リリースの前倒し	183
	アジャイル的な思考の普及	184
<b>7.5</b>	<b>Orbsを使った継続的デプロイの実践例</b>	184
	<b>Orbsの探し方</b>	184
	認定済み、パートナー、サードパーティーOrbs	184
	Orbsのバージョン	185
	<b>必要なもの</b>	186
	アカウント	186
	Docker	186
	サンプルコード	186
	CircleCIプロジェクト	186
	<b>全体の流れ</b>	187
	<b>ECRへのデプロイ</b>	188
	IAMユーザーの権限	188
	CloudFormationでECRの作成	190
	CircleCIで環境変数の設定	194
	.circleci/config.yml	195
	ECR Orbのインポート	195
	aws-ecr/build-and-push-imageジョブ	195
	<b>ECSへのデプロイ</b>	196
	CloudFormationでECSの作成	196
	.circleci/config.yml	199
	ECS Orbのインポート	200
	aws-ecs/deploy-service-updateジョブ	200
<b>7.6</b>	<b>継続的デプロイを使った開発の流れ</b>	201
	<b>デプロイ</b>	202
	<b>本番環境でテスト</b>	202
	本番環境の監視	203
	本番環境に対するE2Eテスト	203
	<b>ロールバック</b>	205
	<b>その他の継続的デプロイ手法</b>	205
	カナリアリリース	206
	ブルー/グリーンデプロイ	206
	ローリングデプロイ	206

## 第8章

# Webアプリケーション開発、インフラでの活用 ..... 207

### 8.1 TypeScript ..... 208

.circleci/config.yml .....	209
ビルド .....	211
テスト .....	212
マトリックスビルド .....	212
JavaScriptへのコンパイル .....	213
テスト結果とカバレッジレポートの作成 .....	213
テスト結果とカバレッジレポートの表示 .....	214

### 8.2 Ruby (Ruby on Rails) ..... 215

.circleci/config.yml .....	216
ビルド .....	218
テスト .....	219
データベースを用いたテスト .....	219
テストを分割して複数コンテナで実行 .....	220
カバレッジのマージ——アプリケーション側の設定 .....	221
カバレッジのマージ——config.yml側の設定 .....	223

### 8.3 PHP (Laravel) ..... 224

.circleci/config.yml .....	225
ソースコードのチェックアウト .....	226
ビルド .....	227
テスト .....	228
Base64を使ってファイルを環境変数として挿入 .....	228
テストの実行 .....	229

### 8.4 Java (Spring Boot) ..... 229

.circleci/config.yml .....	230
OOM問題対策 .....	231
Exit Code 137に注意 .....	232
ヒープサイズに関する環境変数 .....	232
ヒープサイズに関する環境変数の優先度 .....	232
環境変数 .....	234
ビルド .....	234
Gradleバイナリのキャッシュ .....	235
依存関係のキャッシュ .....	235
テスト .....	235

テスト分割の概要 .....	236
ファイル名からテストクラス名を動的に作成 .....	236
テストレポート .....	237
アーティファクト .....	237
<b>8.5 Docker</b> .....	<b>239</b>
<b>Dockerコマンドを使うために</b> .....	239
setup_remote_dockerでリモートホストの立ち上げ .....	239
リモートDocker環境のスペック .....	239
<b>.circleci/config.yml</b> .....	240
Machine Executorを使う場合 .....	241
Docker Layer Caching .....	241
<b>Dockerイメージのキャッシュ戦略</b> .....	242
レジストリからプルする方法 .....	242
save_cacheを使う方法 .....	242
ジョブ間でのイメージの受け渡し .....	243
<b>リモートDocker環境との通信</b> .....	244
実行中のサービスへのアクセス .....	244
ファイルの受け渡し .....	245
<b>8.6 Terraform</b> .....	<b>246</b>
<b>.circleci/config.yml</b> .....	246
<b>tfnotify</b> .....	248
tfnotifyのインストール .....	249
tfnotifyを使うための準備 (GitHub) .....	249
tfnotifyを使うための準備 (Slack) .....	250
tfnotifyの実行 .....	252
<b>Terraformの実行</b> .....	253
IAMユーザーの権限 .....	254
コンテキストの作成 .....	254
run_terraform_planジョブ .....	255
run_terraform_applyジョブ .....	256
<b>State Lockingの導入</b> .....	257
Stateについて .....	257
State Lockingについて .....	258
DynamoDBの作成 .....	258
State Lockingの利用 .....	259
State Lockingの解除 .....	260

## 第9章

# モバイルアプリ開発での活用..... 261

### 9.1 Android 262

静的解析、ユニットテスト、APKの作成.....	262
.circleci/config.yml(build_and_setupジョブ).....	262
Dockerイメージ.....	263
環境変数.....	264
テスト.....	264
アーティファクトとテストレポート.....	265
Test Labと連携.....	267
Test Labの準備.....	267
環境変数.....	267
.circleci/config.yml(run_ftlジョブ).....	268
ワークスペースからAPKのダウンロード.....	269
gcloudによる認証.....	269
Test Labを使ったテストの実行.....	269
アーティファクトとテストレポート.....	270

### 9.2 iOS(macOS) 271

テスト.....	272
.circleci/config.yml(build-and-testジョブ).....	272
macOS Executor.....	273
テスト.....	274
matchによる証明書の作成.....	275
matchについて.....	276
matchの初期化.....	276
証明書とプロビジョニングファイルの作成.....	277
AdHoc IPAの作成.....	277
.circleci/config.yml(generate-ipaジョブ).....	278
開発環境でのIPAの作成.....	278
SSHキーの追加.....	278
追加したSSHキーの使用.....	279
IPAの作成とアップロード.....	279

## 第10章

# デスクトップ/ネイティブアプリ開発での活用..... 281

### 10.1 Windows 282

	.circleci/config.yml.....	282
	Windows Executor.....	283
	さまざまなシェルの使用.....	283
<b>10.2</b>	<b>クロスプラットフォーム</b>	<b>284</b>
	.circleci/config.yml.....	284
	マトリックスビルド.....	286
	キャッシュ.....	287
	クロスプラットフォームのディストリビューション作成.....	287
<b>10.3</b>	<b>Unity</b>	<b>289</b>
	.circleci/config.yml.....	289
	ライセンスのアクティベーション.....	292
	テスト.....	294
	ビルド.....	295
第	<b>11</b>	章
	<b>さまざまなタスクの自動化</b> .....	<b>299</b>
<b>11.1</b>	<b>なぜ自動化するのか</b>	<b>300</b>
	自動化するメリット.....	300
	自動化に適したタスク.....	300
<b>11.2</b>	<b>Webサイトのリリース</b>	<b>300</b>
	ワークフローが担うタスク.....	301
	ワークフローの設定.....	301
	事前準備.....	301
	設定ファイル.....	302
<b>11.3</b>	<b>バージョンごとのリリース作業</b>	<b>305</b>
	ワークフローが担うタスク.....	305
	ワークフロー.....	305
	事前準備.....	305
	設定ファイル.....	306
<b>11.4</b>	<b>セキュリティアラート</b>	<b>307</b>
	ワークフローが担うタスク.....	308
	ワークフロー.....	308
	事前準備.....	308
	設定ファイル.....	309

**11.5 依存ライブラリのアップデート** 311

ワークフローが担うタスク..... 311  
ワークフロー ..... 311  
    事前準備 ..... 311  
    設定ファイル ..... 312

**11.6 ドキュメントの校正** 314

ワークフローが担うタスク..... 314  
ワークフロー ..... 315  
    事前準備 ..... 315  
    textlintの設定 ..... 315  
    reviewdogの設定..... 316

## 第 12 章

### Orbsの作成 ..... 319

**12.1 Orbs作成の基礎** 320

バージョンング ..... 320  
開発用Orbsと本番用Orbs ..... 320  
    それぞれの違い ..... 320  
    公開時の注意点 ..... 321

**12.2 初めてのOrbs作成とデプロイ** 321

Orbsクイックスタート ..... 321  
config.ymlに記述するインラインOrbs ..... 322  
    インラインOrbsを作成..... 322  
    ローカル環境での実行 ..... 322  
Orbsの公開 ..... 323  
    Orbsを公開／利用するためのセキュリティ設定の有効化 ..... 323  
    CircleCIトークンを取得 ..... 324  
    Orbsの名前空間を作成 ..... 325  
    Orbsの公開 ..... 326  
Orbsの設計 ..... 327  
    descriptionキーの設定 ..... 327  
    コマンドとExecutorを同梱する ..... 328  
    Orb内のコマンドやジョブ名には簡潔な名前を付ける ..... 329  
    パラメータにはなるべくデフォルト値を付ける ..... 329  
    ジョブだけのOrbを作らないようにする ..... 330  
    stepsパラメータを使う ..... 331  
    Examplesを用意する ..... 332

<b>12.3</b>	<b>Orbs開発でもCI/CDを実現</b>	<b>333</b>
	Orbsのテスト .....	333
	バリデーション .....	335
	展開テスト—circleci config process.....	335
	ランタイムテスト—circleci local execute.....	337
	インテグレーションテスト .....	338
	Orbsのテストからデプロイまでの流れ.....	340
	orb-toolsを使ったOrbs開発.....	341
	orb-tools/publish-devジョブ .....	341
	dev-promote-prod-from-commit-subjectジョブ.....	342
	orb-tools/packジョブ.....	344
	orb-tools/trigger-integration-tests-workflowジョブ.....	345
<b>12.4</b>	<b>orb-toolsを使ったテスト/デプロイの自動化</b>	<b>346</b>

## Appendix config.ymlの基本構造 .....

<b>A.1</b>	<b>versionキー</b>	<b>350</b>
	version.....	350
<b>A.2</b>	<b>ジョブ</b>	<b>351</b>
	jobs.....	351
	Docker Executor .....	352
	Machine Executor .....	353
	macOS Executor .....	354
	Windows Executor .....	354
	steps .....	355
<b>A.3</b>	<b>workflows</b>	<b>355</b>
	version.....	355
	ワークフロー .....	355
	triggers.....	356
	schedule .....	356
	jobs.....	357
<b>A.4</b>	<b>runステップ</b>	<b>360</b>
	run .....	360
<b>A.5</b>	<b>ビルトインステップ</b>	<b>361</b>
	when/unless.....	361

checkout.....	363
setup_remote_docker.....	363
save_cache/restore_cache.....	364
deploy(非推奨).....	366
store_artifacts.....	368
store_test_results.....	368
persist_to_workspace/attach_workspace.....	369
add_ssh_keys.....	370

**A.6**

**その他 370**

parameters.....	370
string(文字列).....	371
boolean(真偽値).....	371
integer(整数).....	372
enum(列挙型).....	372
executor.....	373
steps.....	374
env_var_name(環境変数名).....	374
orbs.....	375
commands.....	376
executors.....	376
パイプライン変数/パイプラインパラメータ.....	378
パイプライン変数の設定例.....	379
パイプラインパラメータの設定例.....	380
Column シェルオプションの初期値.....	382
索引.....	383
執筆者プロフィール.....	391