Proudit

静的HTML公開フローをサーバレスでDevOps！

[本編]

はじめに

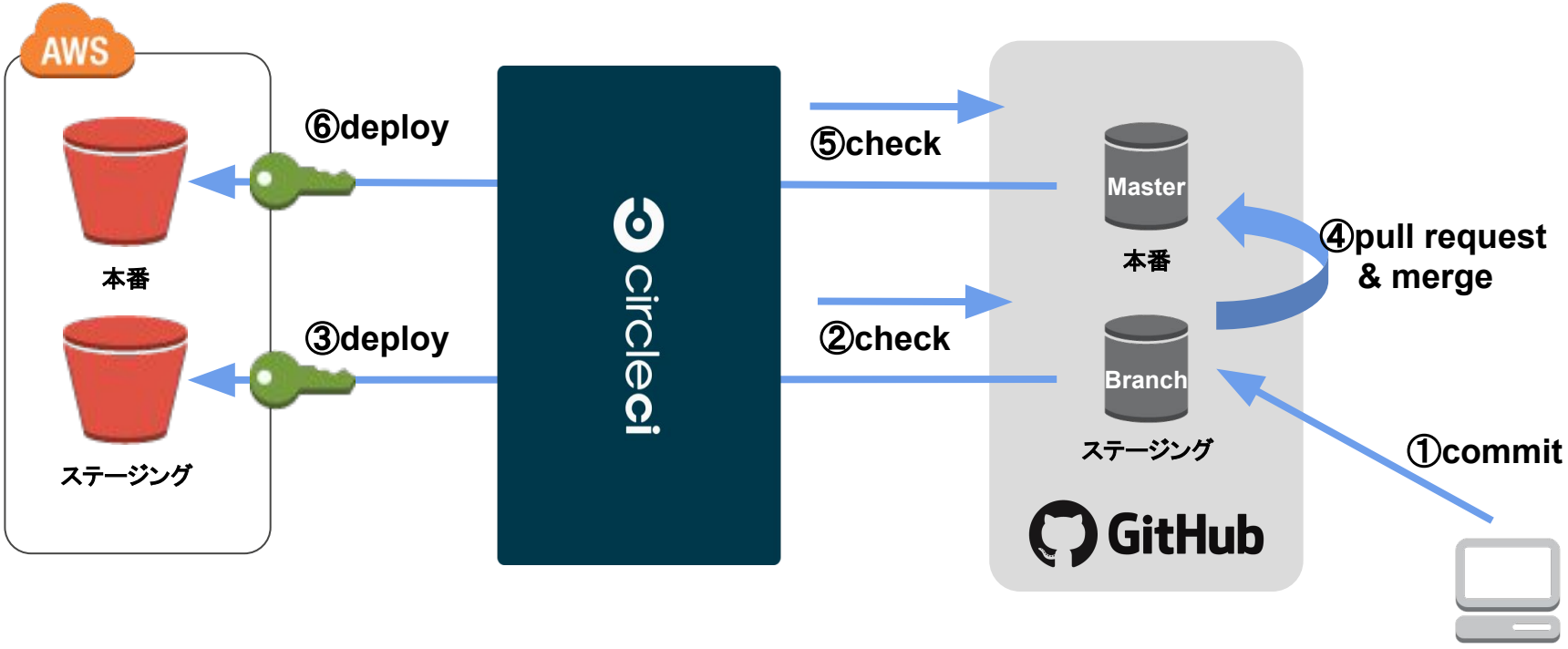
本資料では以下の作業が完了したことを前提に進めさせていただきます。

- AWSアカウントの作成
- GitHubアカウントの作成

もし、作成がまだの方がいましたら実施をお願いいたします。
また、本資料で利用しているAWSマネージメントコンソールの言語設定は英語となります。
日本語設定で進める場合は別途読み替えをお願い致します。

他にも、Workshop中にご不明点等がございましたらお気軽にスタッフにお声がけください。
全力でサポートさせていただきます。

完成イメージ（DevOpsフロー）



パラメータ、サンプルコード確認

ハンズオンで利用するサンプルコードは下記に用意してありますので、利用時にコピー＆ペーストしてご利用ください。
https://github.com/proudit/jawsdays2016_handson_s3-devops/blob/master/README.md

下記赤字の部分は各ユーザ固有の値となりますので、適宜編集、置き換えをお願いします。

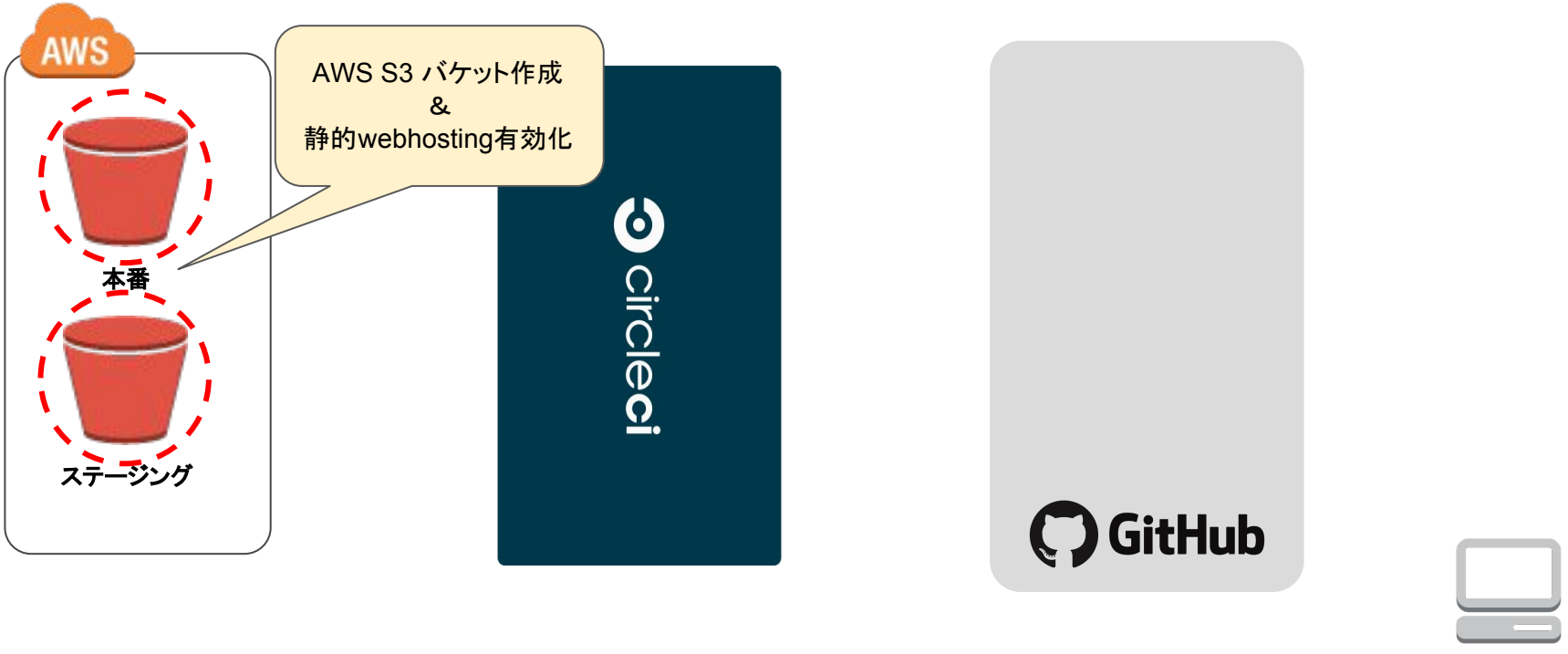
- IAM
  - User Name : jawsdays2016-s3-deploy-user
  - Policy Name : jawsdays2016-s3-deploy-policy
  - Access Key Id : ＜取得したAccess Key Id＞
  - Secret Access Key : ＜取得した Secret Access Key＞
- S3
  - 本番バケット名 : prod.＜GitHubアカウント＞-jawsdays2016.jp
  - ステージングバケット名 : stg.＜GitHubアカウント＞-jawsdays2016.jp
  - リージョン : Tokyo

※S3バケットは全AWSユーザ間で重複出来ませんので必ず固有の値にしてください。
（例ではgithubアカウントIDを利用して固有化してます）

S3バケットの作成

フェーズ1.

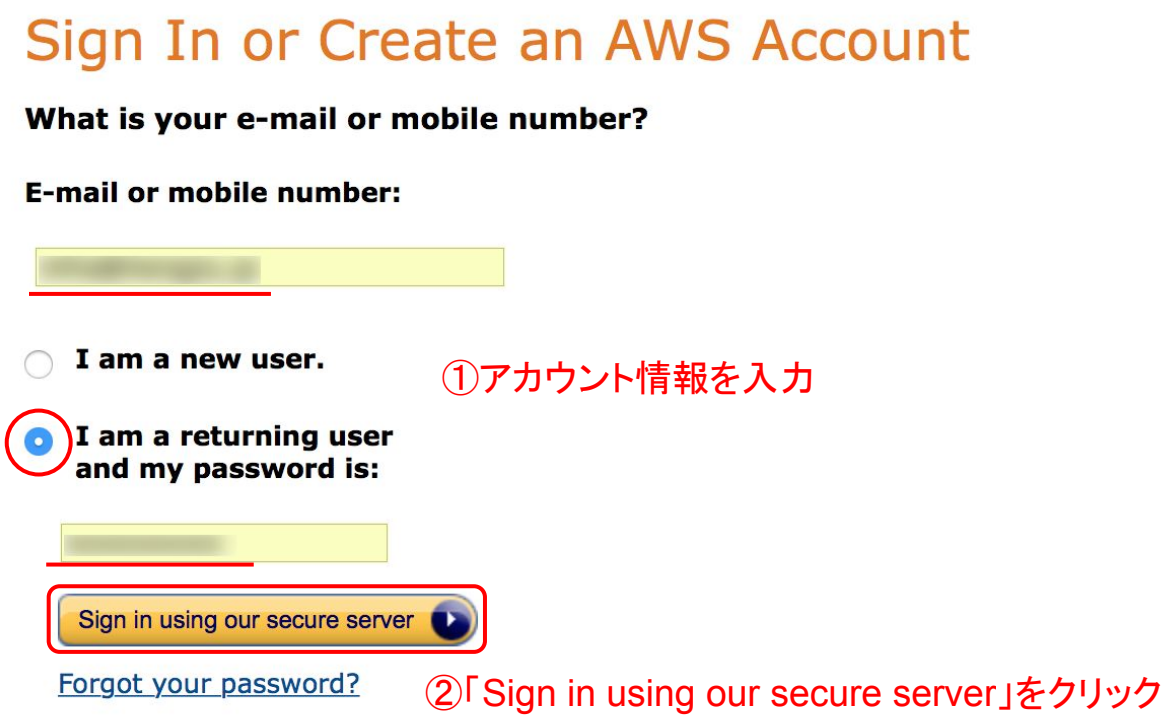
S3バケット作成イメージ



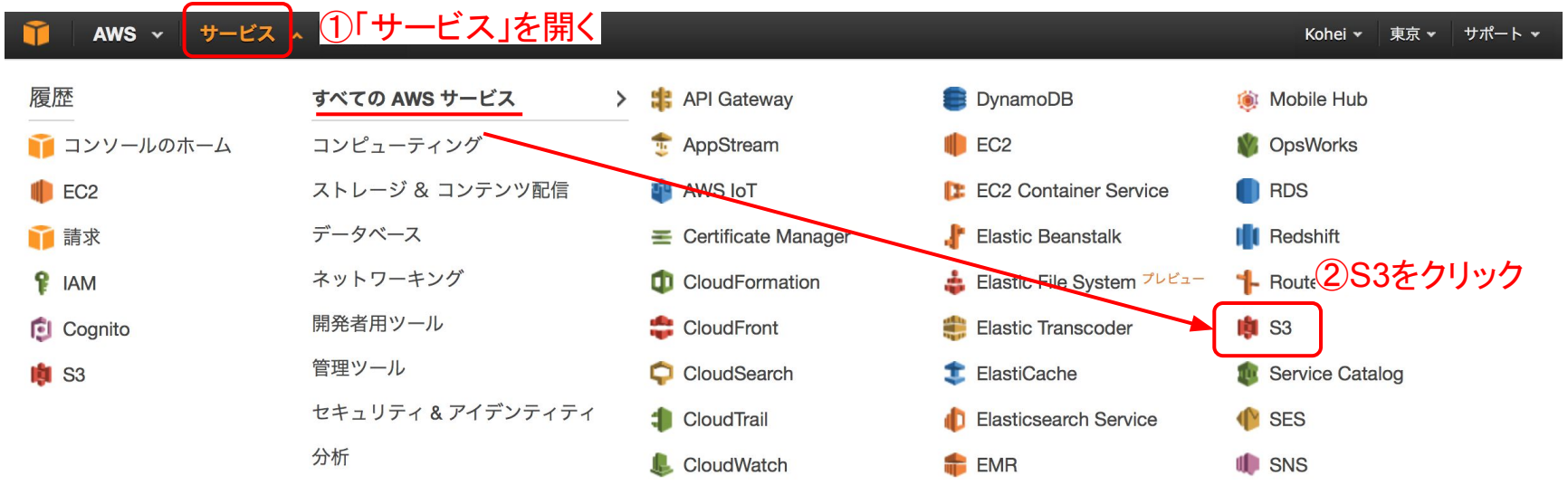
1-1. AWS マネジメントコンソールへログイン①

http://aws.amazon.com/jp/　①URLへアクセス

1-2. AWS マネジメントコンソールへログイン②



1-3. S3サービス画面へ移動

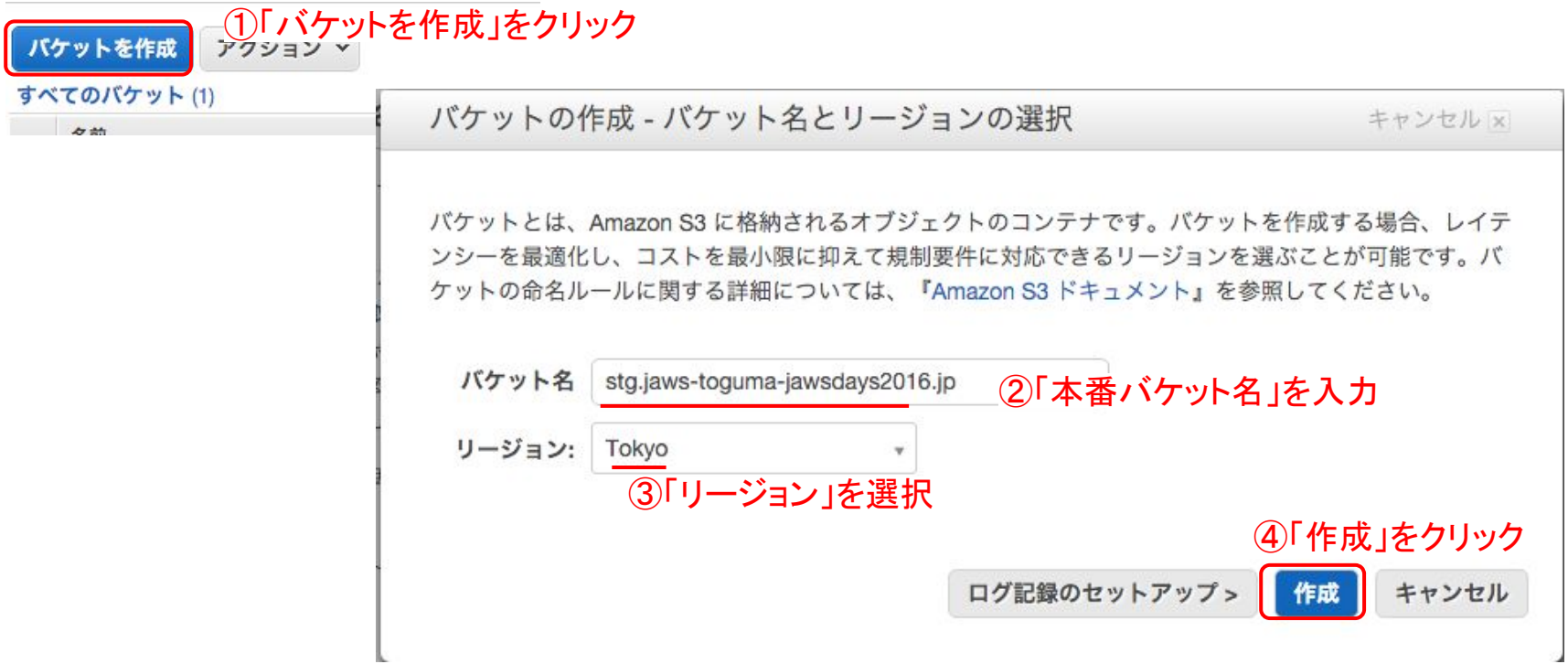


1-4. 本番用バケットの作成①

本番とステージングの両方のバケット作成します。

1-5. ステージング用バケットの作成①

本番とステージングの両方のバケット作成します。



1-6. 本番用バケットポリシーの設定①

1-7. 本番用バケットポリシーの設定②

1-8. ステージング用バケットポリシーの設定①

②「プロパティ」をクリック

アップロード　フォルダの作成　アクション

プレフィックスでの検索　なし　プロパティ　転送

すべてのバケット / stg.jaws-toguma-jawsdays2016.jp

名前　ストレ

①ステージングバケットを選択

バケット「stg.jaws-toguma-jawsdays2016.jp」は空です

バケット: stg.jaws-toguma-jawsdays2016.jp

バケット: stg.jaws-toguma-jawsdays2016.jp
リージョン: Tokyo
作成日: Fri Mar 11 15:06:47 GMT+900 2016
所有者: tech

③「アクセス許可」をクリックして開く

アクセス許可

アクセスポリシーを使い、バケットとそのコンテンツへのアクセスを制御できます。詳細はこちら 。

被付与者: tech　リスト　アップロード/削除　アクセス許可の表示

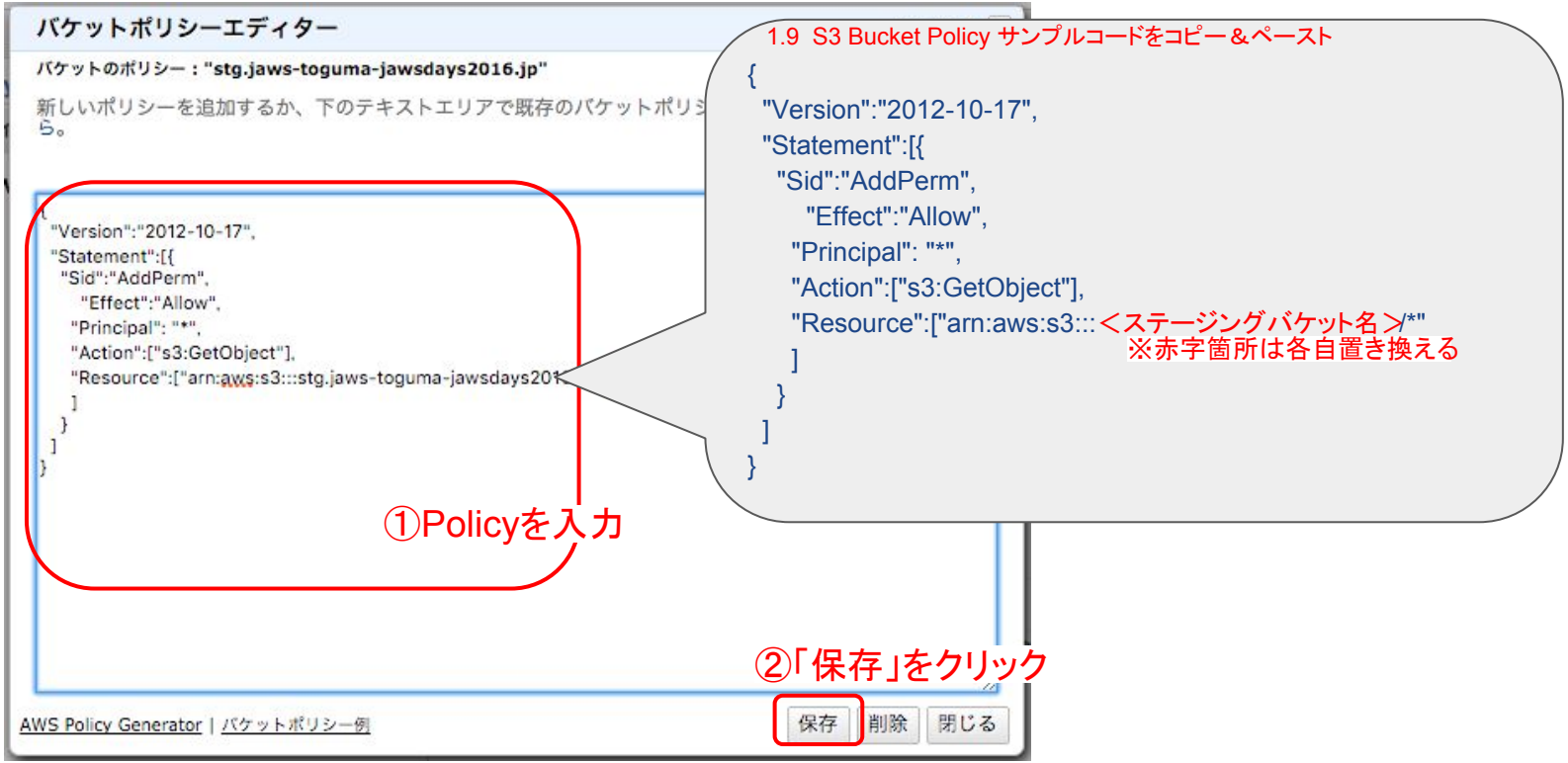
アクセス許可の編集

④Bucket Policyの設定

さらにアクセス許可を追加する　バケットポリシーの追加　CORS 設定の追加

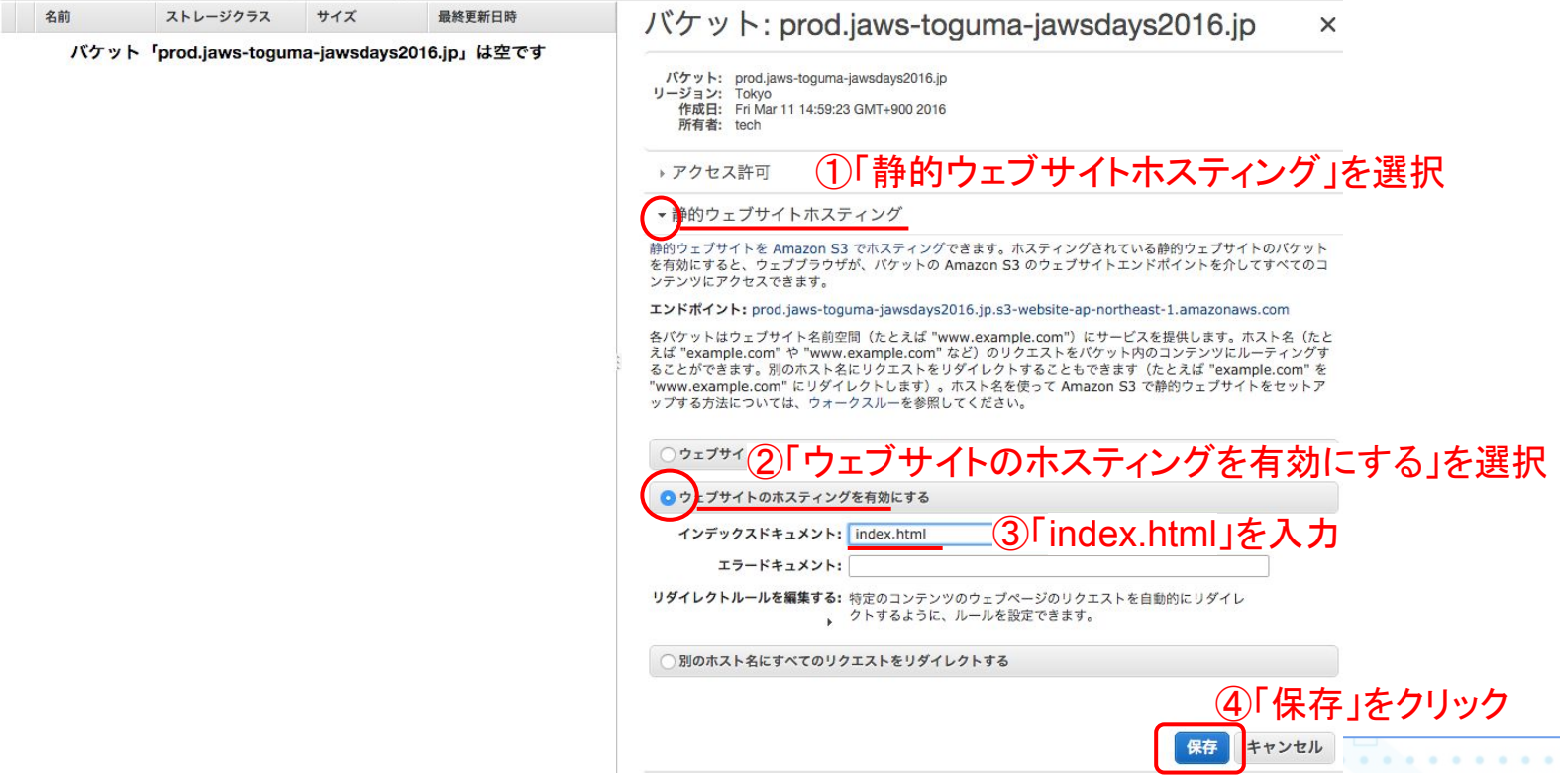
保存　キャンセル

1-9. ステージング用バケットポリシーの設定②



1-10. 本番用ウェブサイトの有効化

バケットのウェブサイトホスティングを有効化します。



1-11. ステージング用ウェブサイトの有効化

バケットのウェブサイトホスティングを有効化します。

名前	ストレージクラス	サイズ	最終更新日時

バケット「stg.jaws-toguma-jawsdays2016.jp」は空です

バケット: stg.jaws-toguma-jawsdays2016.jp ×

バケット: stg.jaws-toguma-jawsdays2016.jp
リージョン: Tokyo
作成日: Fri Mar 11 15:06:47 GMT+900 2016
所有者: tech

アクセス許可

①「静的ウェブサイトホスティング」を選択

静的ウェブサイトホスティング

静的ウェブサイトを Amazon S3 でホスティングできます。ホスティングされている静的ウェブサイトのバケットを有効にすると、ウェブブラウザが、バケットの Amazon S3 のウェブサイトエンドポイントを介してすべてのコンテンツにアクセスできます。

エンドポイント: stg.jaws-toguma-jawsdays2016.jp.s3-website-ap-northeast-1.amazonaws.com

各バケットはウェブサイト名前空間（たとえば "www.example.com"）にサービスを提供します。ホスト名（たとえば "example.com" や "www.example.com" など）のリクエストをバケット内のコンテンツにルーティングすることができます。別のホスト名にリクエストをリダイレクトすることもできます（たとえば "example.com" を "www.example.com" にリダイレクトします）。ホスト名を使って Amazon S3 で静的ウェブサイトをセットアップする方法については、ウォークスルーを参照してください。

ウェブサイト

②「ウェブサイトのホスティングを有効にする」を選択

ウェブサイトのホスティングを有効にする

インデックスドキュメント: index.html

③「index.html」を入力

エラードキュメント:

リダイレクトルールを編集する: 特定のコンテンツのウェブページのリクエストを自動的にリダイレクトするように、ルールを設定できます。

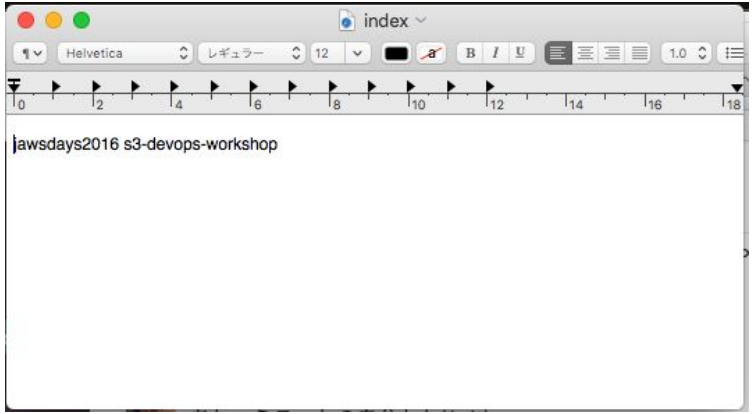
別のホスト名にすべてのリクエストをリダイレクトする

④「保存」をクリック

保存 キャンセル

1-12. webサイト確認①

任意のテキストエディタでサイト確認用のindex用HTMLを作成。



「index.html」というファイル名で保存。

1-13. webサイト確認②

用意した「index.html」を、各バケットへアップロード。

※本番とステージング両方のバケットで行います。

1-14. 静的webホスティングサイト確認③

①「プロパティ」をクリック

アップロード　フォルダの作成　アクション

プレフィックスでの検索　なし　プロパティ　転送

すべてのバケット / prod.jaws-toguma-jawsdays2016.jp

名前	ストレージクラス	サイズ	最終更新日時
index.html	スタンダード	694 bytes	Fri Mar 11 15:4

バケット: prod.jaws-toguma-jawsdays2016.jp
リージョン: Tokyo
作成日: Fri Mar 11 14:59:23 GMT+900 2016
所有者: tech

アクセス許可

静的ウェブサイトホスティング

静的ウェブサイトを Amazon S3 でホスティングできます。ホスティングされている静的ウェブサイトのバケットを有効にすると、ウェブブラウザが、バケットの Amazon S3 のウェブサイトエンドポイントを介してすべてのコンテンツにアクセスできます。

エンドポイント: prod.jaws-toguma-jawsdays2016.jp.s3-website-ap-northeast-1.amazonaws.com

各バケットはウェブサイト名前空間（たとえば "www.example.com"）にサービスを提供します。ホスト名（たとえば "example.com" や "www.example.com"

②「エンドポイント」をクリック

※ブラウザからindex.htmlの内容が表示されているかを確認

1-15. ブラウザで確認

Endpointをクリックするとブラウザにて閲覧確認できます。

Proudit

IAM作成

フェーズ2.

2. IAMとは

AWS Identity and Access Management (IAM) の略であり、

ユーザー（またはサービス）に対して特定のAWSリソース への

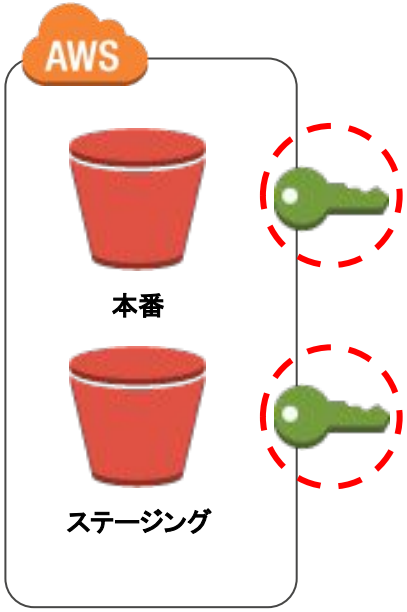
アクセスを安全に制御するための仕組み

このハンズオンの例だと特定のS3バケットへ

ファイルアップするために必要な権限のみ、

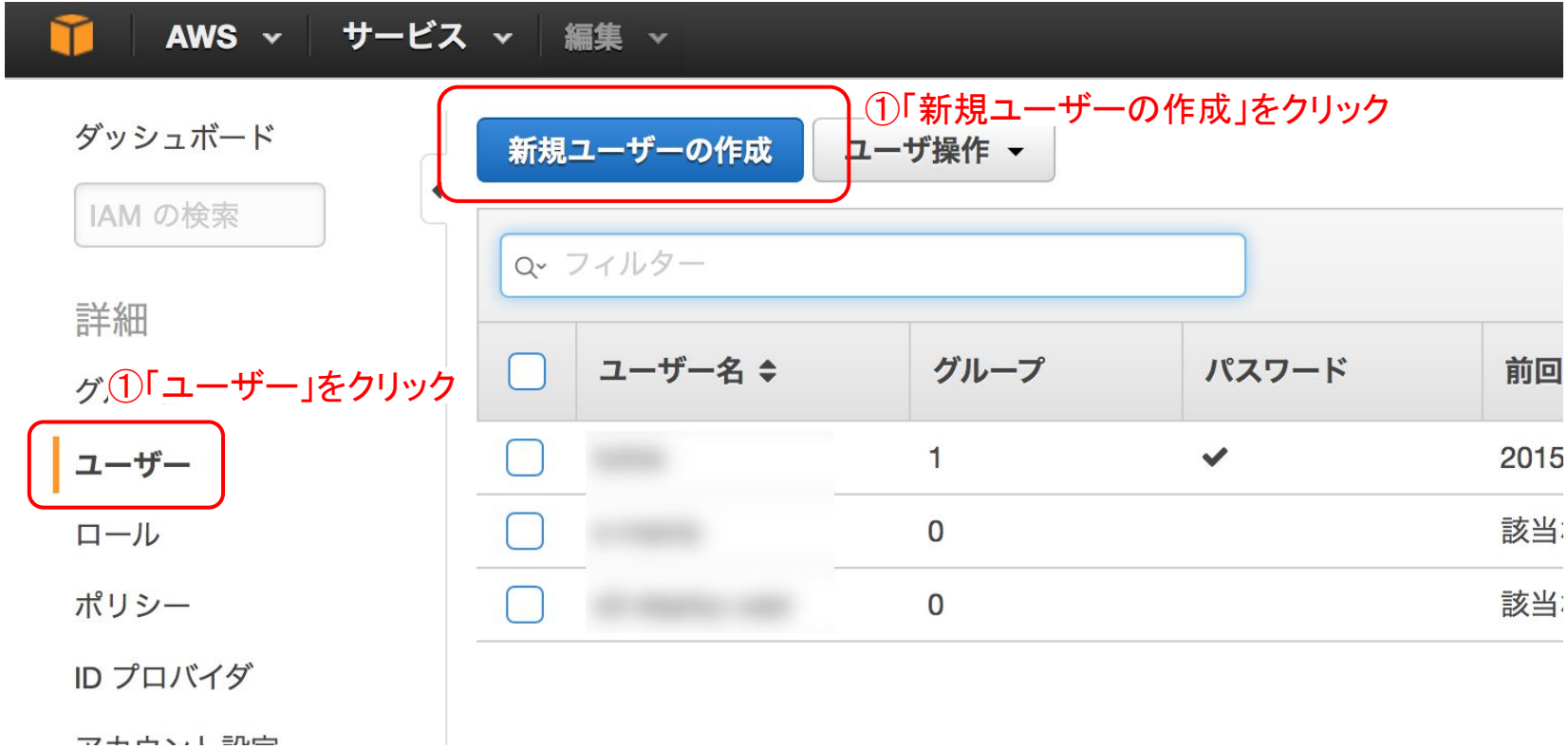
CircleCIへ付与するために利用。

IAM設置 イメージ



2-1. IAMサービス画面へ移動

2-2. IAMユーザ作成①



2-3. IAMユーザ作成②

ユーザー名の入力:

1. jawsdays2016-s3-deploy-user
2.
3.
4.
5.

最大でそれぞれ 64 文字

①ユーザー名を入力

②チェックが入っていることを確認

☑ ユーザーごとにアクセスキーを生成

ユーザーが AWS サービス API に対して安全な REST またはクエリプロトコルリクエストを行うには、アクセスキーが必要です。

AWS マネジメントコンソールにアクセスする必要があるユーザーについては、このウィザードの完了後に [ユーザー] パネルでパスワードを作成します。

③「作成」をクリック

キャンセル　作成

2-4. IAMユーザ作成③

1 ユーザーが正常に作成されました。

これは、これらのユーザーセキュリティ認証情報をダウンロード

これらの認証情報はいつでも管理および再作成できます。

▼ ユーザーのセキュリティ認証情報を非表示

①「ユーザーのセキュリティ認証情報を表示」

jawsdays2016-s3-deploy-user

②「アクセスキーID」と「シークレットアクセキー」をメモ

アクセスキー ID: [redacted]NS7NA

シークレットアクセスキー: [redacted]RG/8Jx9RXPCiw0soYbRrPv

※ここに表示されるID情報は後で必要になります。

③「認証情報のダウンロード」をクリック

④「閉じる」をクリック

閉じる　認証情報のダウンロード

2-5. IAMポリシー作成①

2-6. IAMポリシー作成②

AWS サービス 編集 Kohei グローバル サポート

ポリシーの作成

手順 1: ポリシーの作成

手順 2: 許可を設定

手順 3: ポリシーの確認

ポリシーの作成

ポリシーとは所定の書式に従って、1 つ以上のアクセス権限を記述したドキュメントです。AWS 管理ポリシーをコピー、Policy Generator を使用、または独自のカスタムポリシーを入力して、ポリシーを作成します。

AWS 管理ポリシーをコピー

AWS 管理ポリシーをニーズに合わせてカスタマイズします。

選択

Policy Generator

Policy Generator を使用して、リストからサービスおよびアクションを選択します。選択に応じて Policy Generator でポリシーを作成します。

選択

独自のポリシーを作成

ポリシーエディターを使用して、独自のポリシーを入力または貼り付けます。

選択

①「独自のポリシーを作成」を選択

2-7. IAMポリシー作成③

AWS サービス 編集

ポリシーの作成

手順 1: ポリシーの作成

手順 2: 許可を設定

手順 3: ポリシーの確認

ポリシーの確認

以下のポリシードキュメントを編集してアクセス権限をカスタマイズします
の「ポリシーの概要」を参照してください。 変更を適用する前に、このポ
します。

ポリシー名

jawsdays2016-s3-deploy-policy

説明

①Policy Nameを入力

ポリシードキュメント

```
9            "s3:DeleteObject",
10           "s3:ListBucket"
11       ],
12       "Resource": [
13           "arn:aws:s3:::prod.jaws-toguma-jawsday
14           "arn:aws:s3:::prod.jaws-toguma-jawsdays
15           "arn:aws:s3:::stg.jaws-toguma-jawsdays201
16           "arn:aws:s3:::stg.jaws-toguma-jawsdays2016.jp
17       ]
18   }
19 ]
```

ポリシーの編集に自動フォーマットを使用する

キャンセル ポリシーの検証 戻る ポリシーの作成

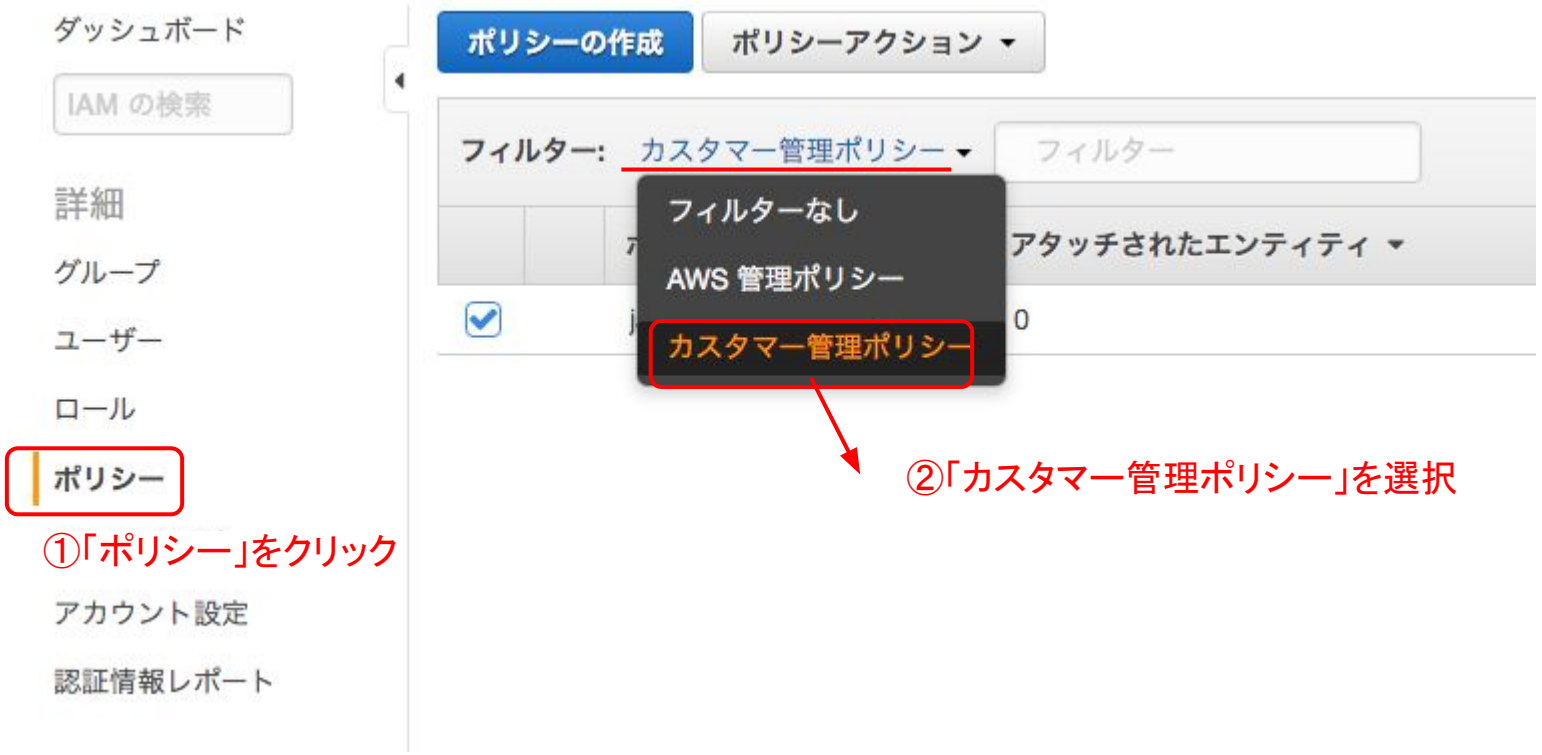
②IAMサンプルコードをコピー&ペースト
下記赤字の部分を自環境のものへ置き換える

```
{
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
      "s3:DeleteObject",
      "s3:ListBucket"
    ],
    "Resource": [
      "arn:aws:s3:::<本番Bucket Name>",
      "arn:aws:s3:::<本番Bucket Name>/*",
      "arn:aws:s3:::<ステージングBucket Name>",
      "arn:aws:s3:::<ステージングBucket Name>/*"
    ]
  }
]
}
```

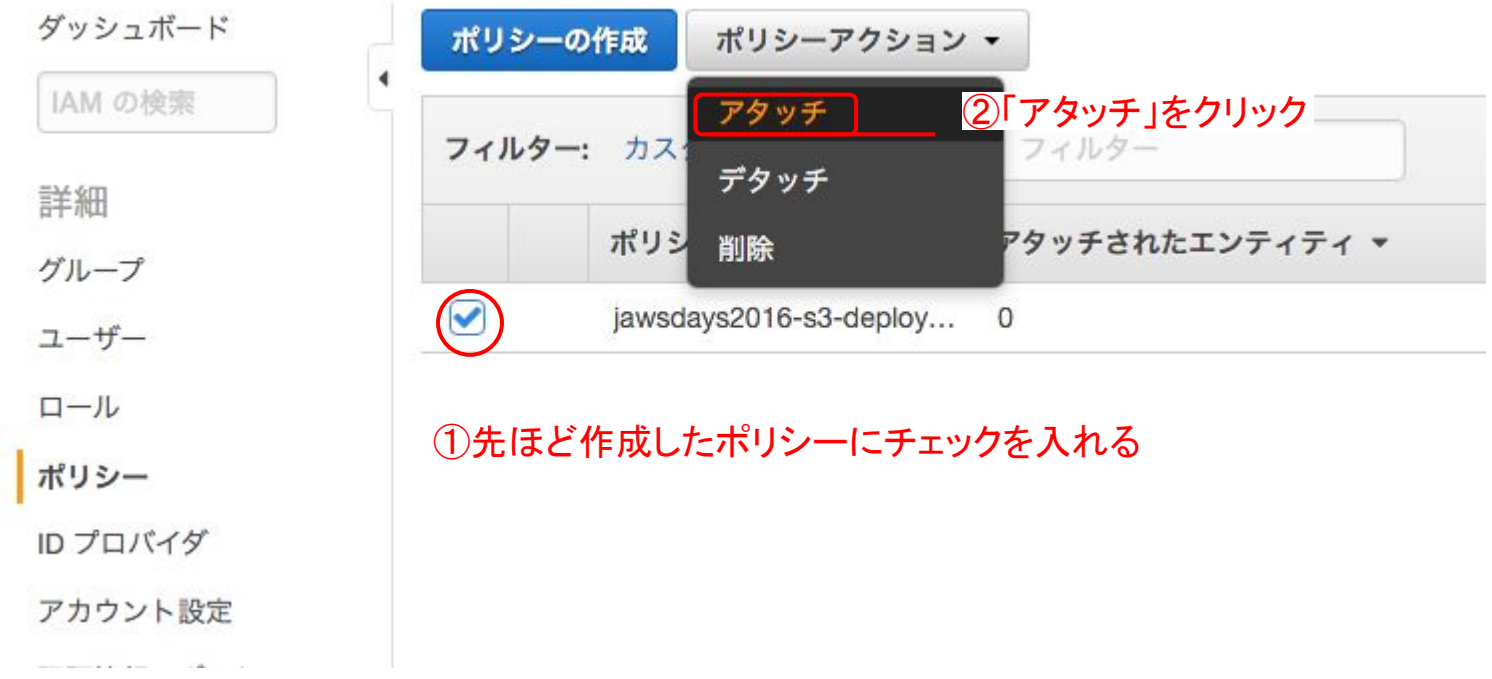
※赤字箇所は各自置き換える

③修正後「ポリシーの作成」をクリック

2-8. ユーザへーポリシーをアタッチ①



2-9. ユーザへポリシーをアタッチ②

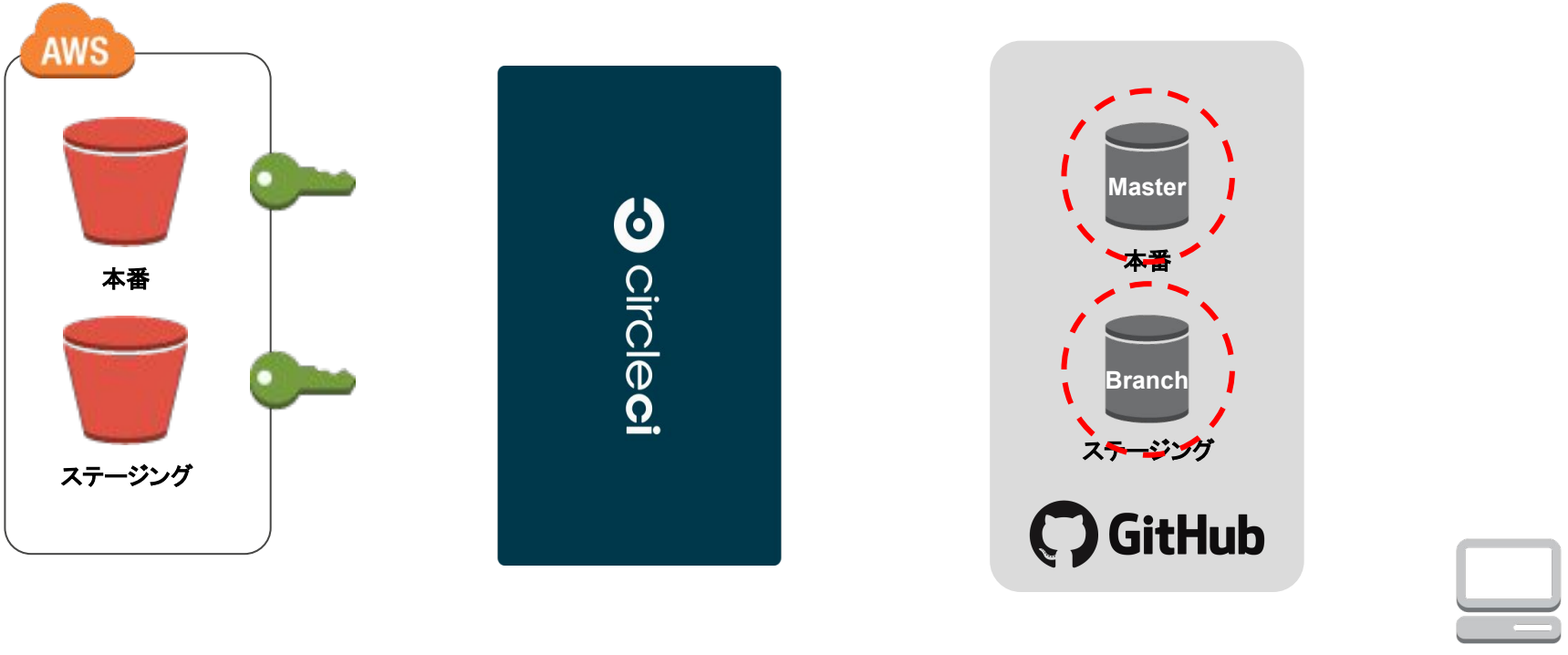


2-10. ユーザへポリシーをアタッチ③

GitHubリポジトリの準備

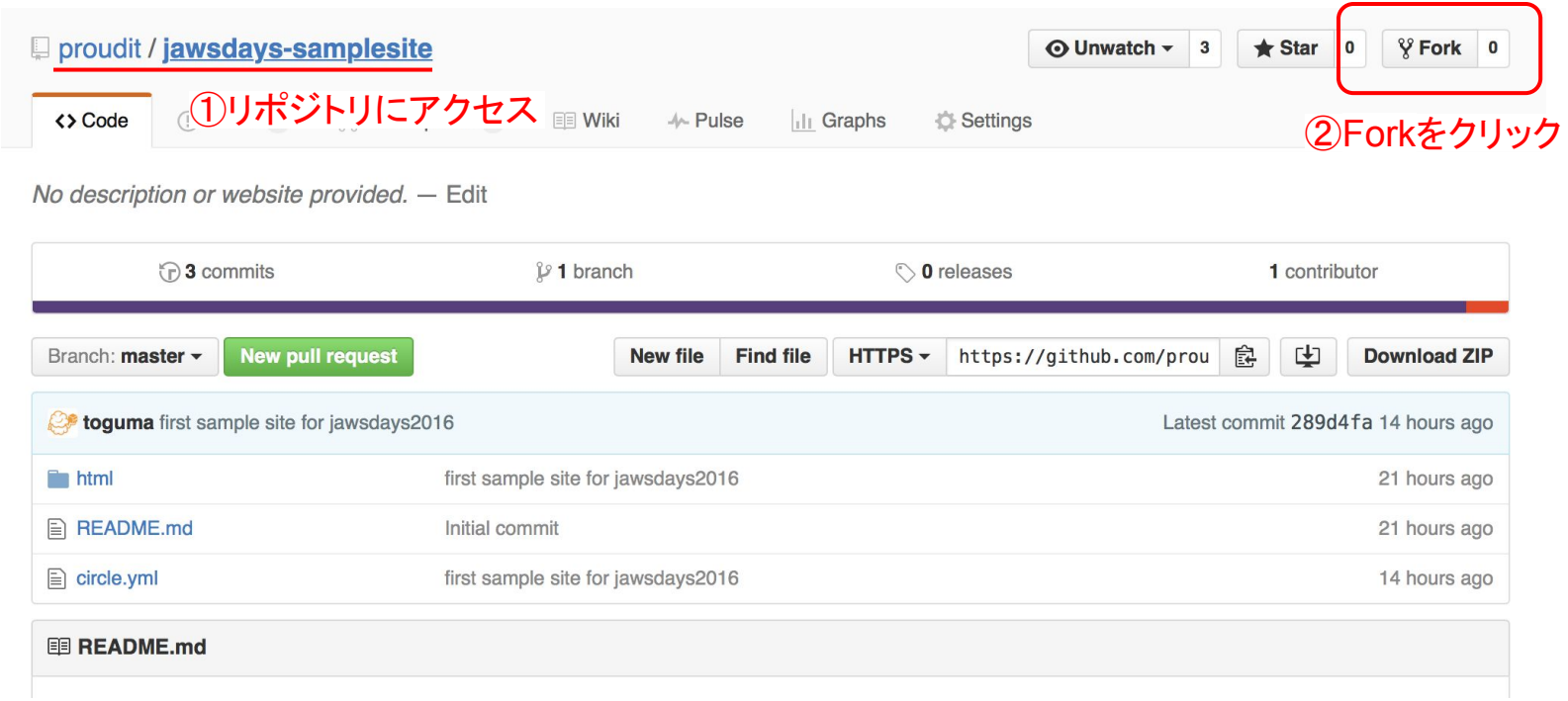
フェーズ3.

イメージ



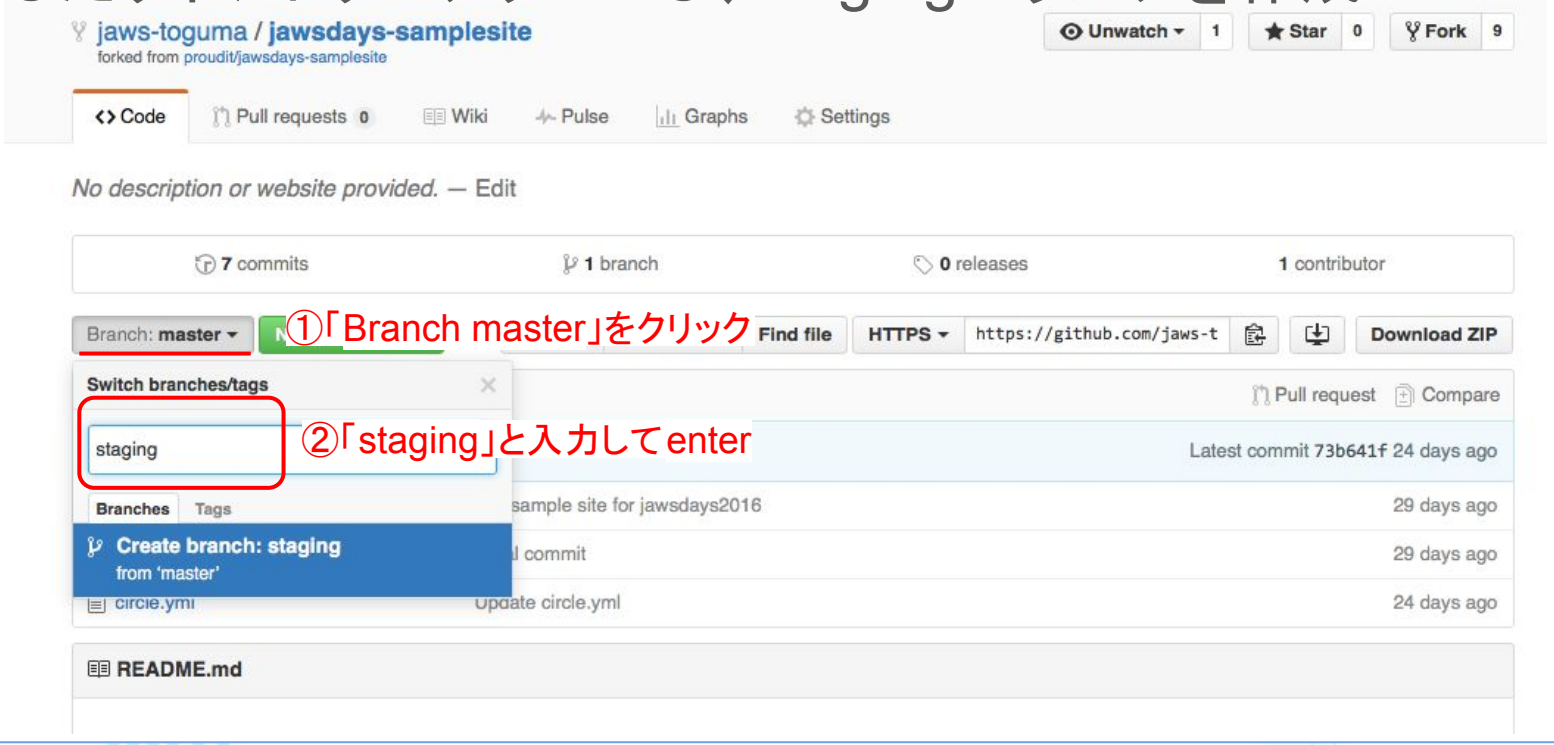
3-1. リポジトリの作成(Fork)

https://github.com/proudit/jawsdays-samplesiteをForkしてリポジトリを作成します。

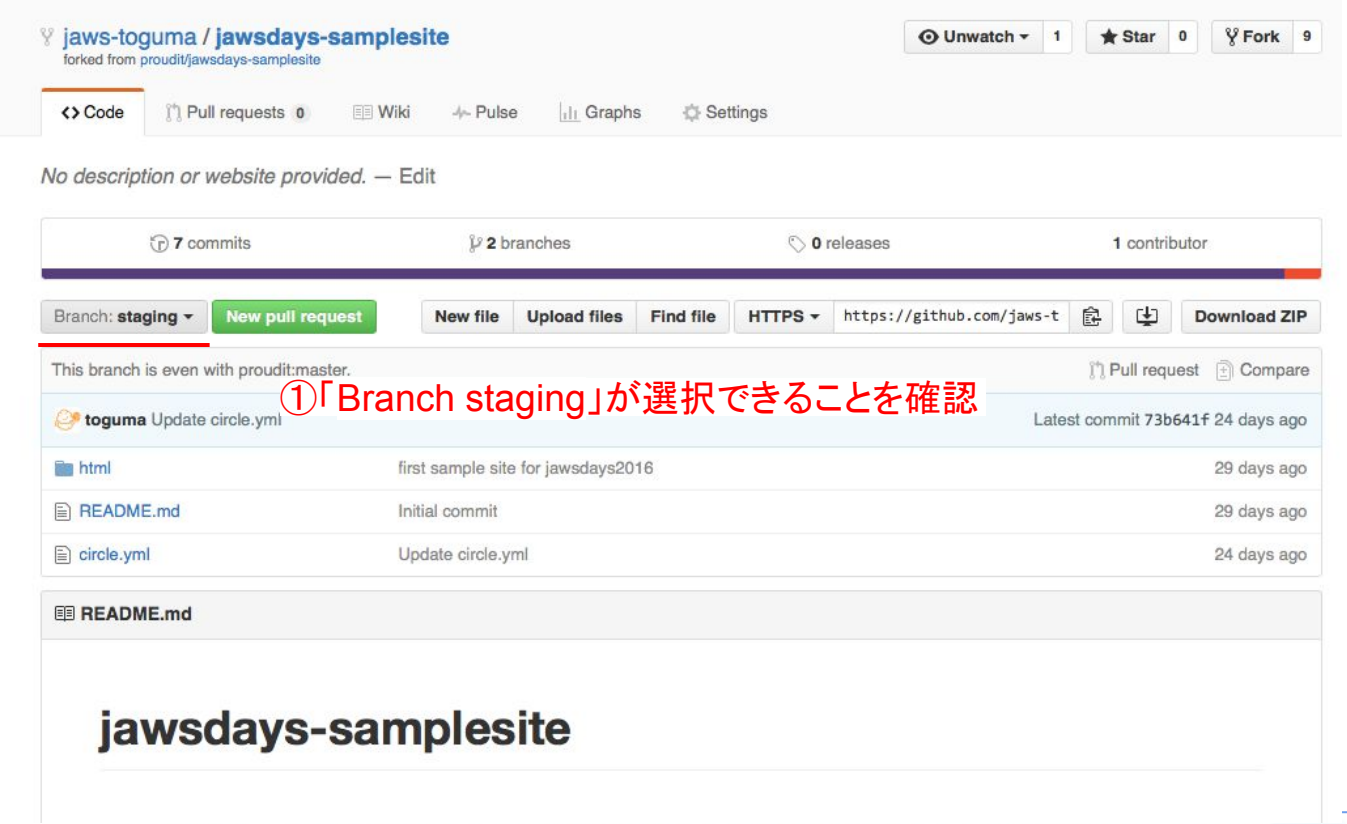


3-2. Branchの作成

Forkしたリポジトリへアクセスし、stagingブランチを作成



3-3. Branchの作成確認



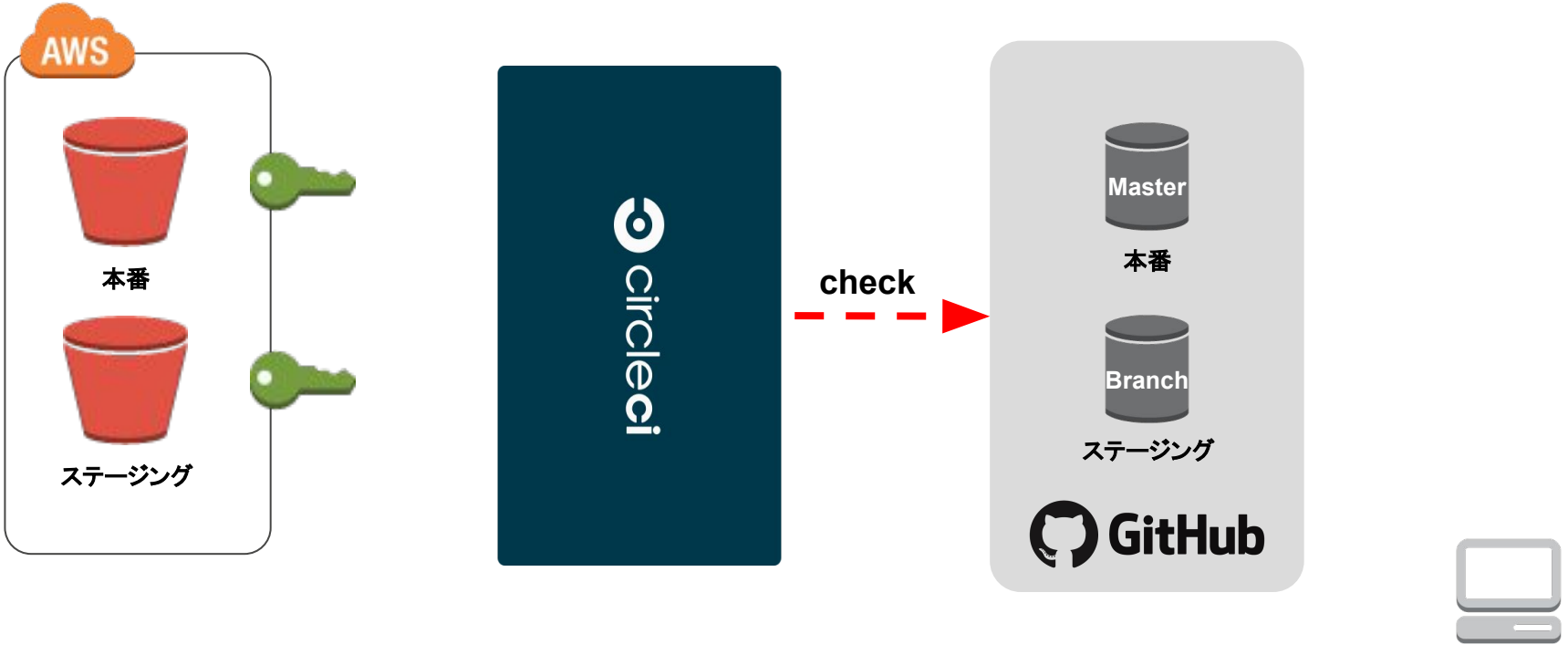
Proudit

AWS

GitHubとCircleCIの連携

フェーズ4.

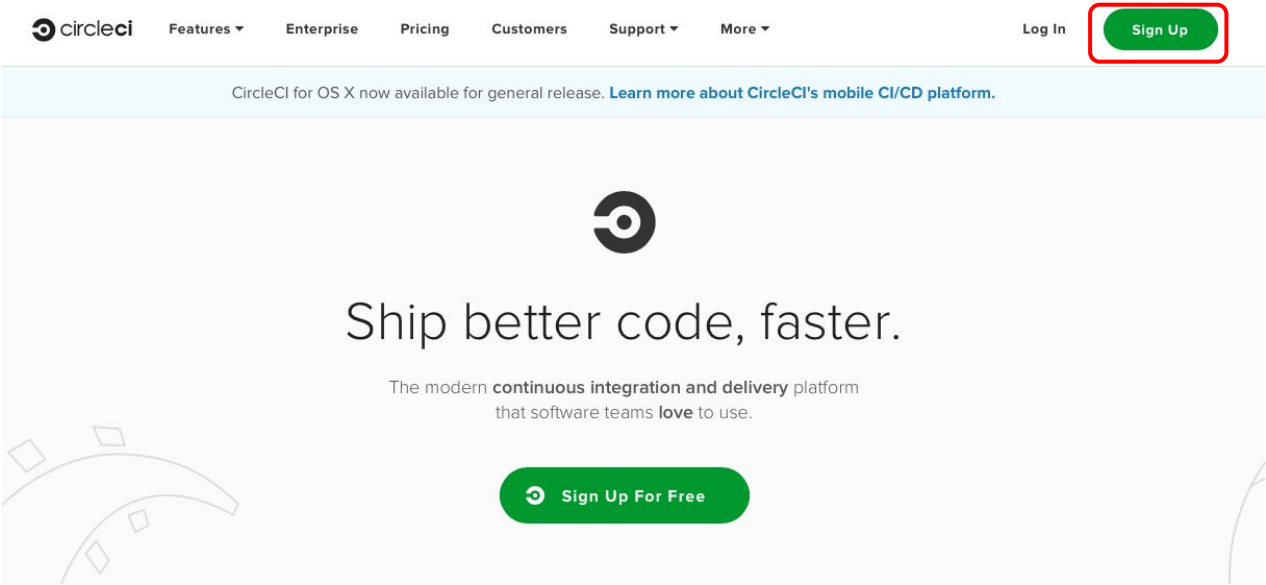
イメージ


AWS
本番
ステージング
circleci
check
Master
本番
Branch
ステージング
GitHub

4-1.ログイン

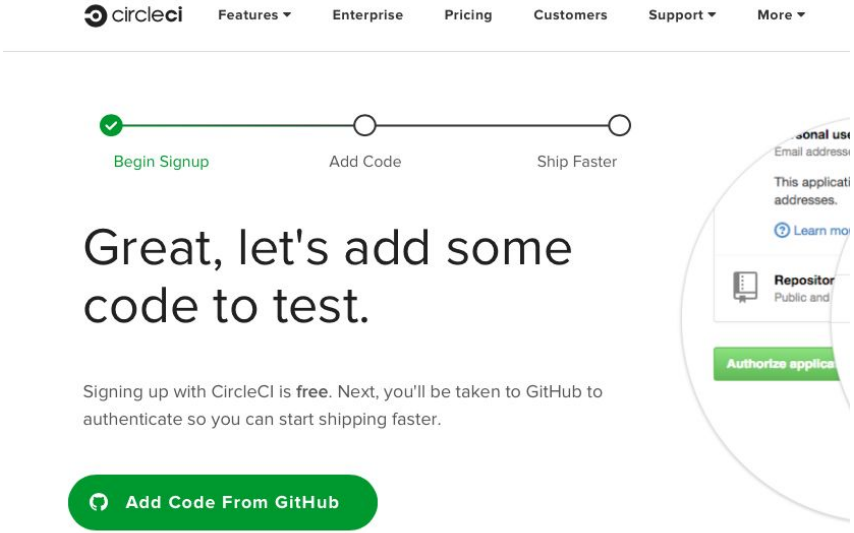
Circleciへアクセスし「Sign in」をクリックします。

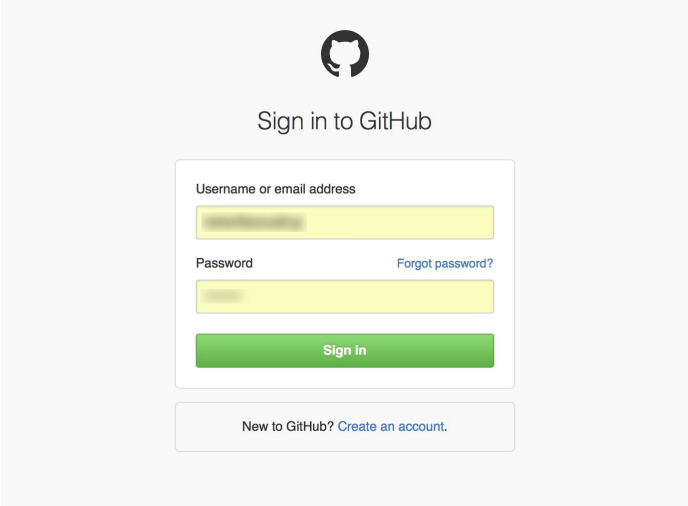
https://circleci.com/



4-2.GitHubアカウントと連携

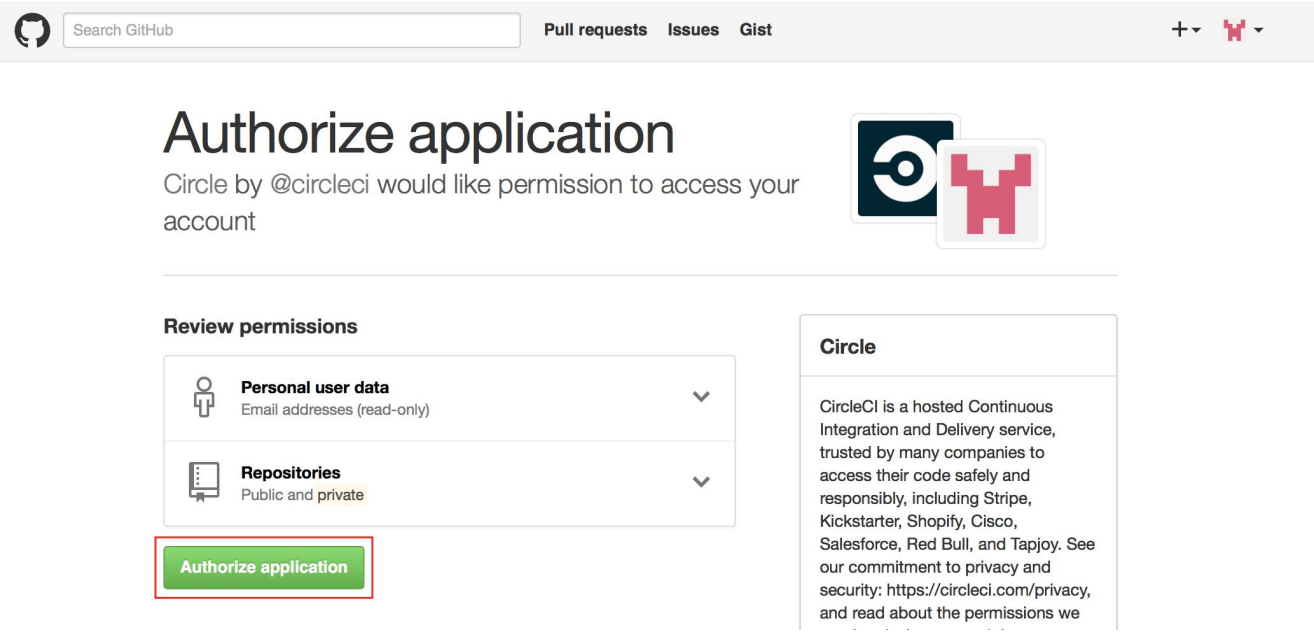
ログイン画面が表示されるので事前に準備した GitHubアカウントへログインします。
※すでにGitHubへログインが完了している場合はこの画面は表示されません。





4-3.認証

認証画面が表示されるため、「 Authorize application」をクリックします。



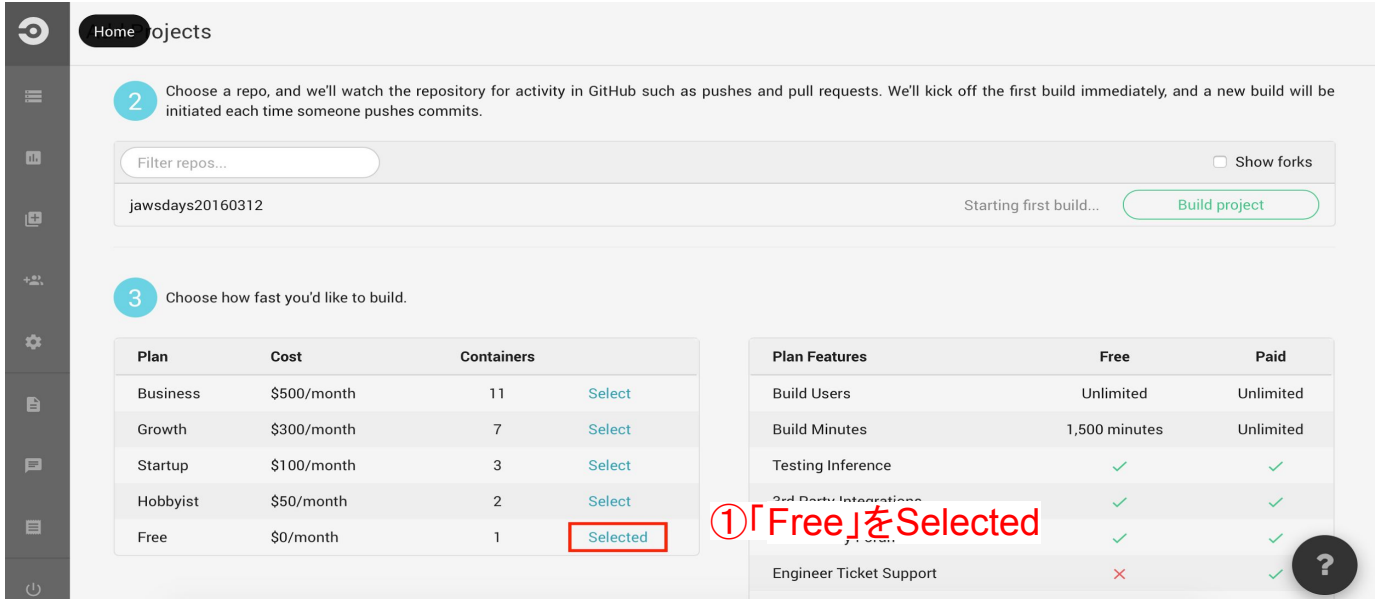
4-4.リポジトリの選択

アカウントを指定すると、リポジトリがリストされるので連携したいリポジトリの「 Build project」をクリックします。

4-5.プランの選択

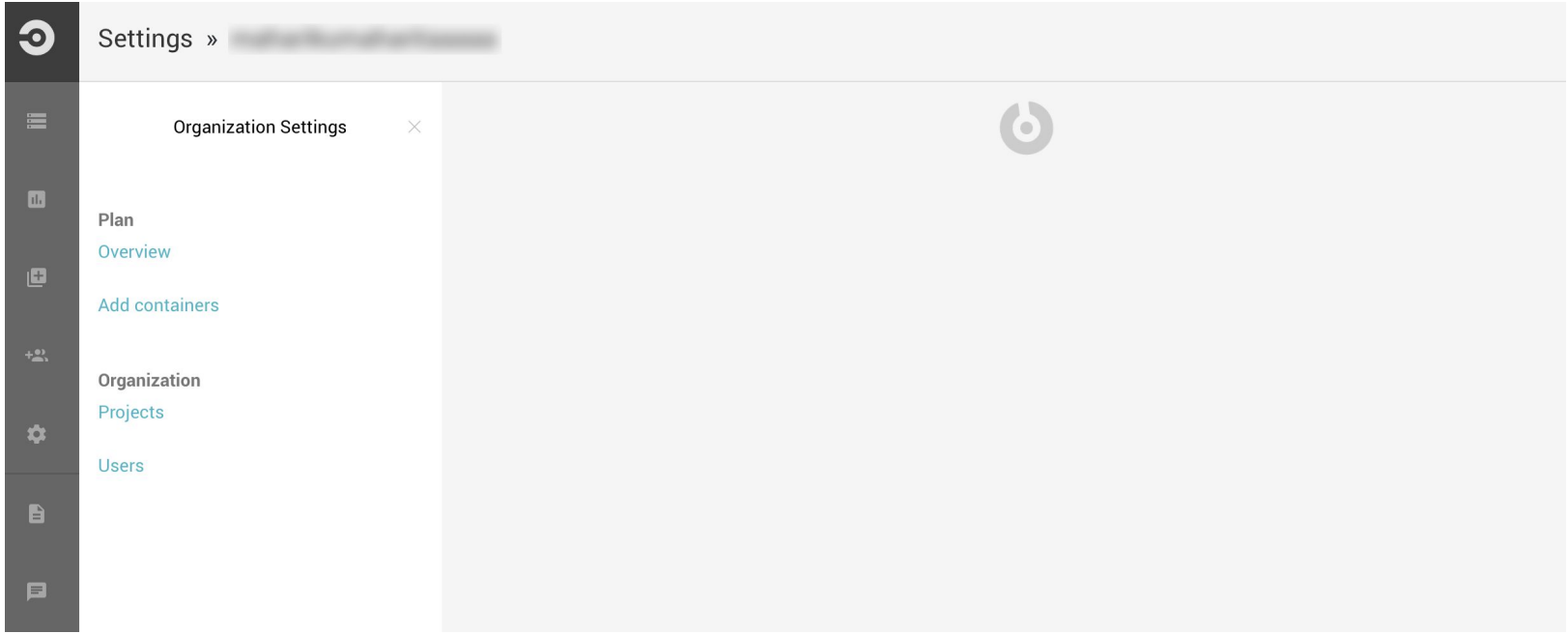
利用するプランを指定します。

※このステップは表示されない場合があるのでその際は気にせず次へ進んでください。



4-7.完了

以上で連携が完了です。

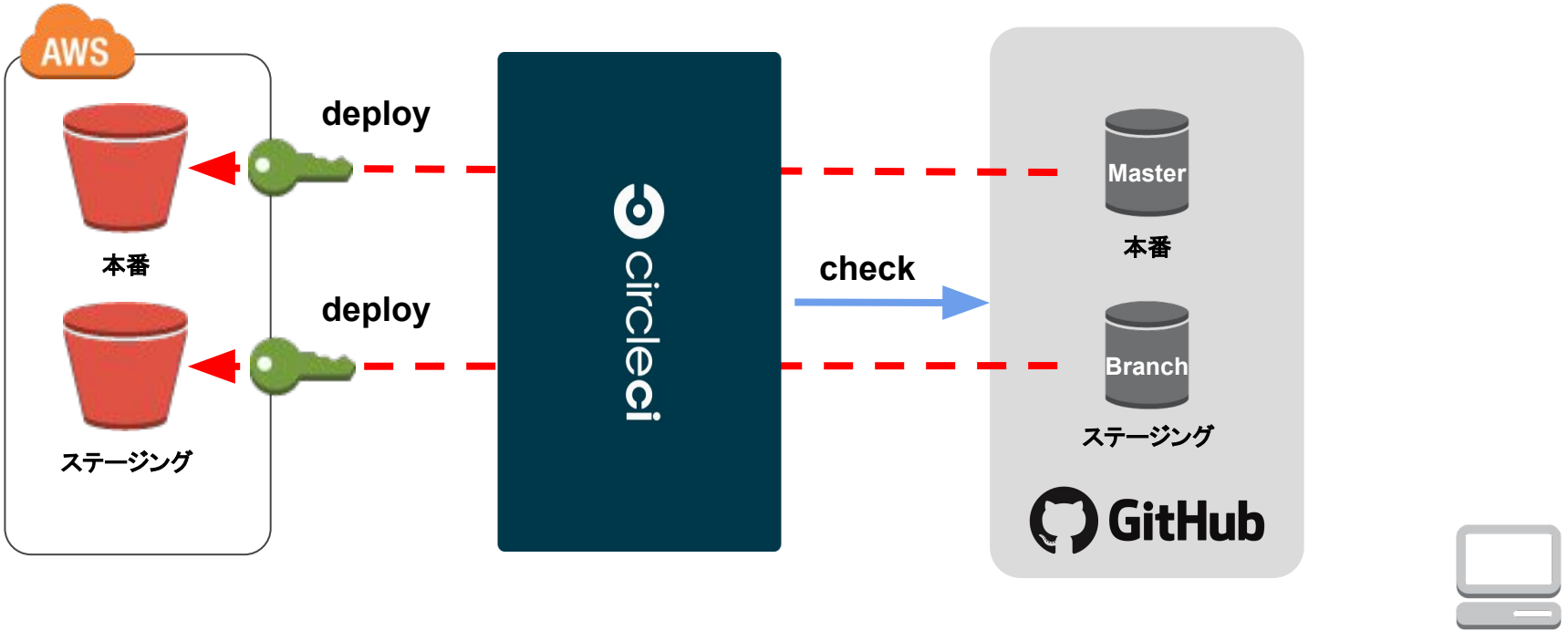


Proudit

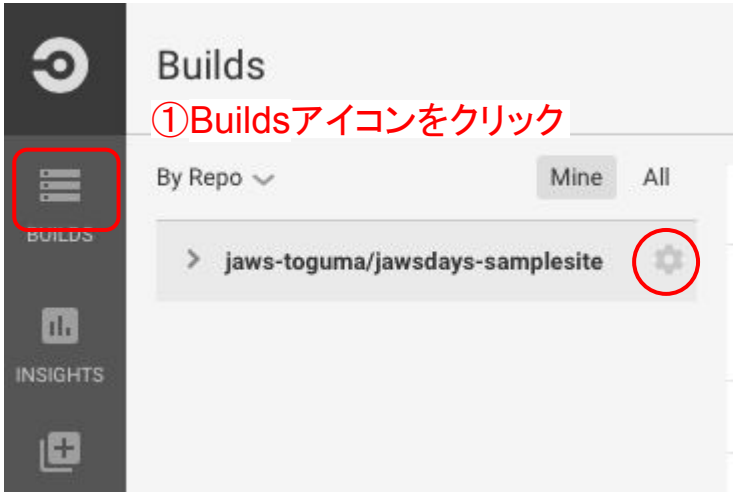
CircleCIとAWSの連携

フェーズ5.

イメージ

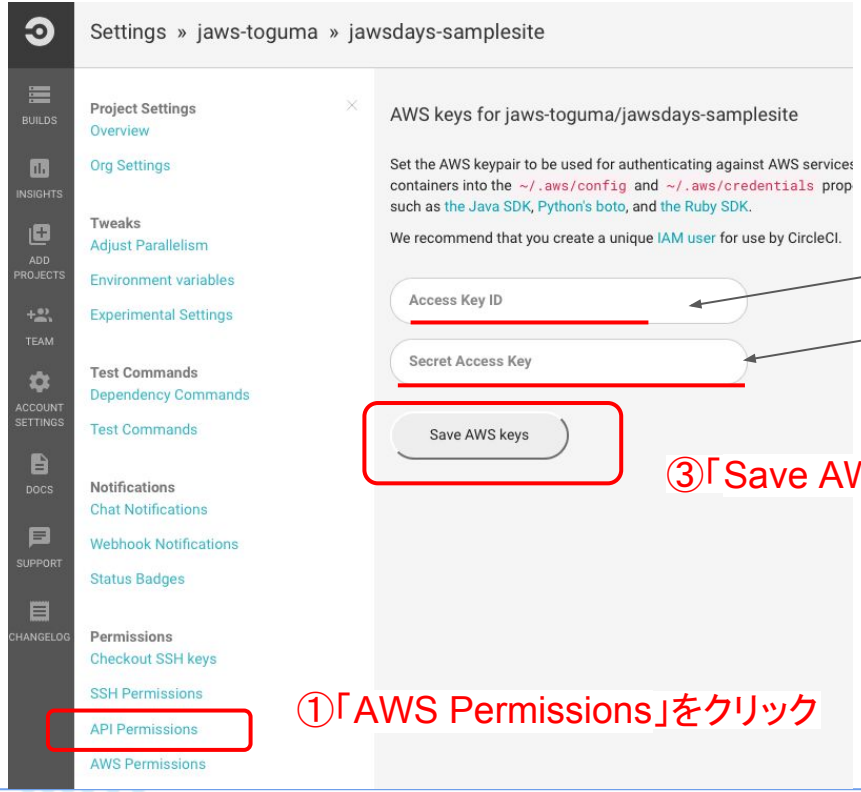

AWS
本番
ステージング
deploy
deploy
circleci
check
Master
本番
Branch
ステージング
GitHub

5-1. CircleCIにアクセス



②⚙をクリックして設定画面を表示

5-2. CircleCIにアクセス



① 「AWS Permissions」をクリック

② 2-4でメモした
「Access Key ID」と
「Secret Access Key」を入力

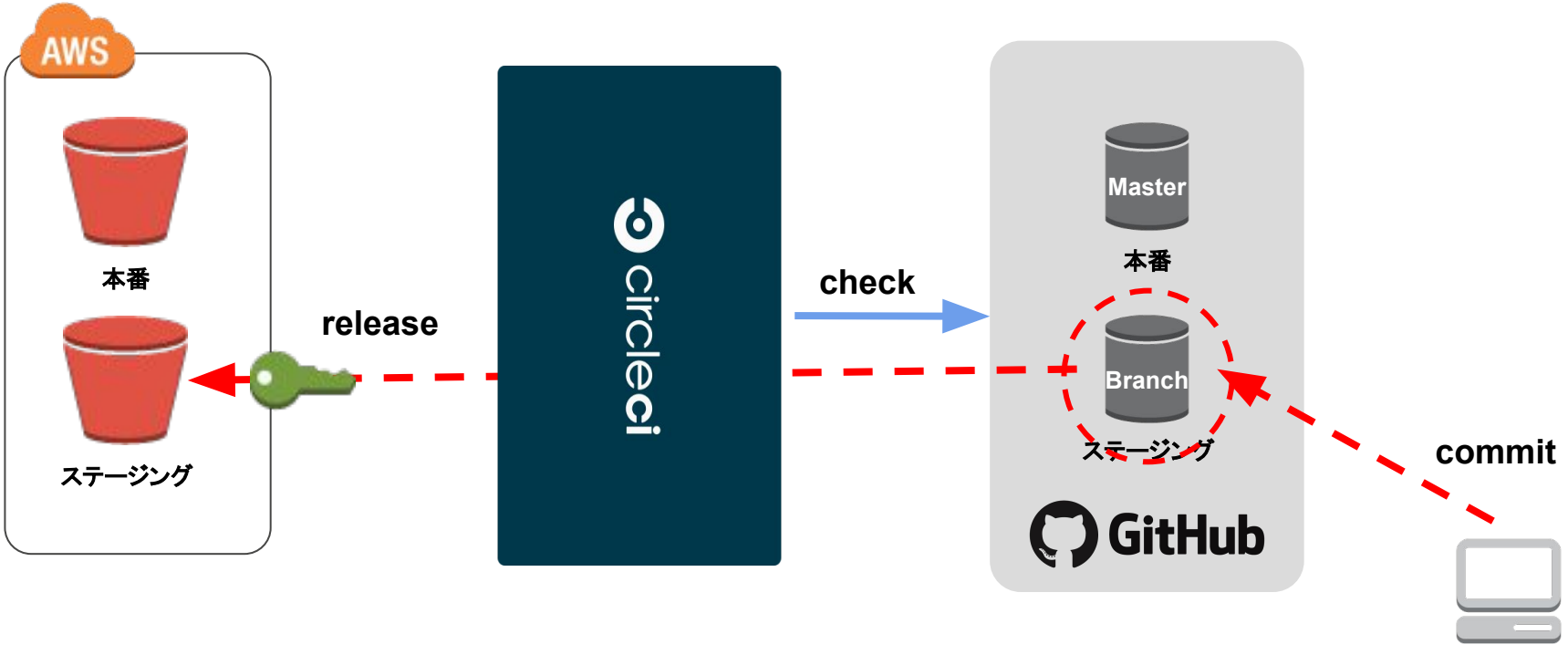
③「Save AWS keys」で保存

Proudit

CircleCIの自動デプロイ設定

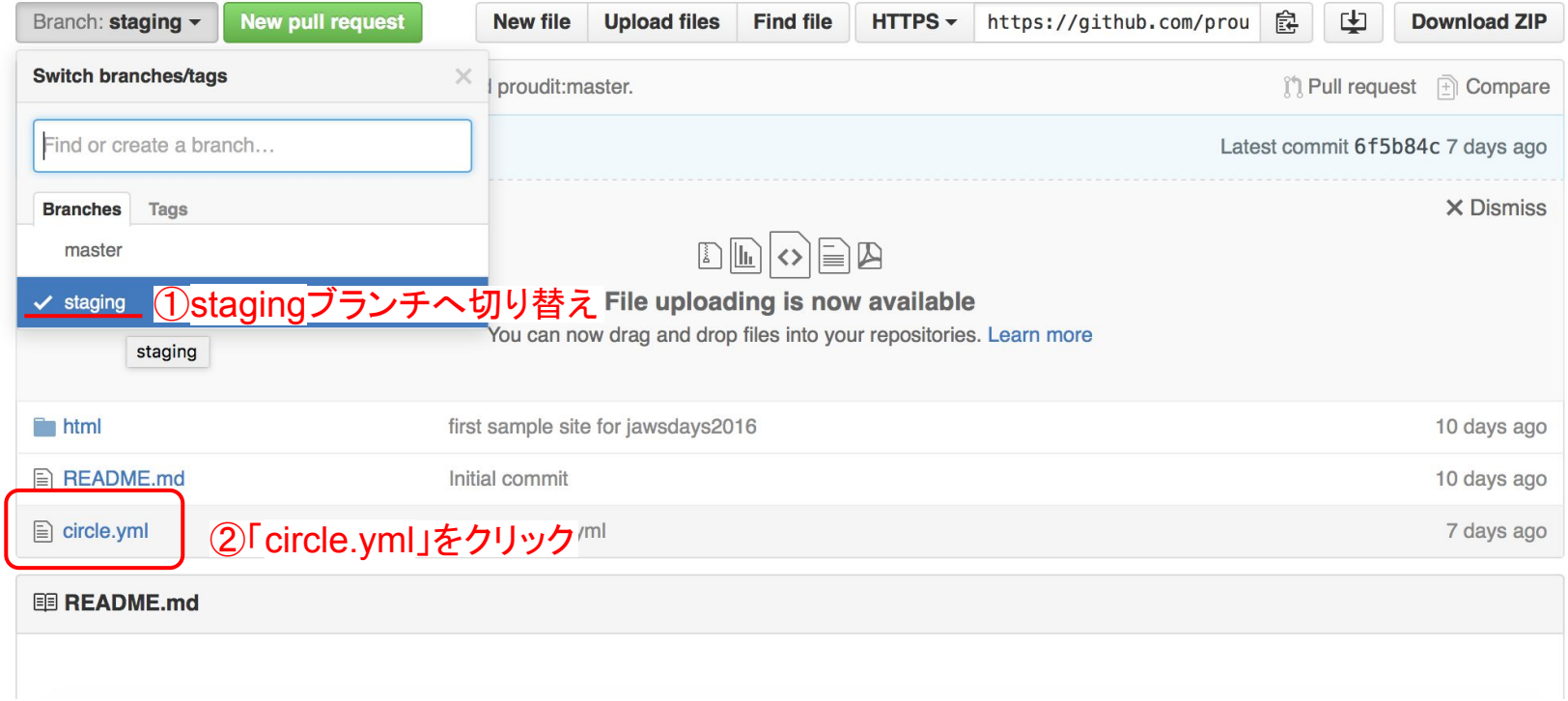
フェーズ6.

ステージングサイトへのデプロイ イメージ



6-1. circle.ymlの編集①

GitHubへアクセスし、stagingブランチにある「circle.yml」を編集します。



6-2. circle.ymlの編集②

25 lines (20 sloc) | 440 Bytes

Raw Blame History

①クリックして編集

```
machine:
  timezone:
    Asia/Tokyo

dependencies:
    override:
        - sudo pip install awscli
    post:
        - aws configure set region ap-northeast-1

test:
  override:
    - echo "All the tests are passed!!"

deployment:
  production:
    branch: master
    commands:
      - aws s3 sync html/ s3://[your s3 bucket]/ --delete
  staging:
    branch: staging
    commands:
      - aws s3 sync html/ s3://[your s3 bucket]/ --delete
```

6-3. circle.ymlの編集③

```
15 deployment:
16   production:
17     branch: master
18     commands:
19       - aws s3 sync html/ s3://[your s3 bucket]/ --delete
20   staging:
21     branch: staging
22     commands:
23       - aws s3 sync html/ s3://[your s3 bucket]/ --delete
24
25
```

～ 省略 ～
deployment:
 production:
  branch: master
  commands:
   - aws s3 sync html/ s3://＜本番バケット名＞ --delete
 staging:
  branch: staging
  commands:
   - aws s3 sync html/ s3://＜ステージングバケット名＞/ --delete

①バケット名を修正

※手元のテキスト資料から貼り付ける

Commit changes

Update circle.yml

Add an optional extended description…

Commit directly to the `master` branch

Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Commit changes　Cancel

②「Commit changes」をクリック

6-4. CircleCIにアクセス

CircleCIの管理画面からステータスが確認できます。

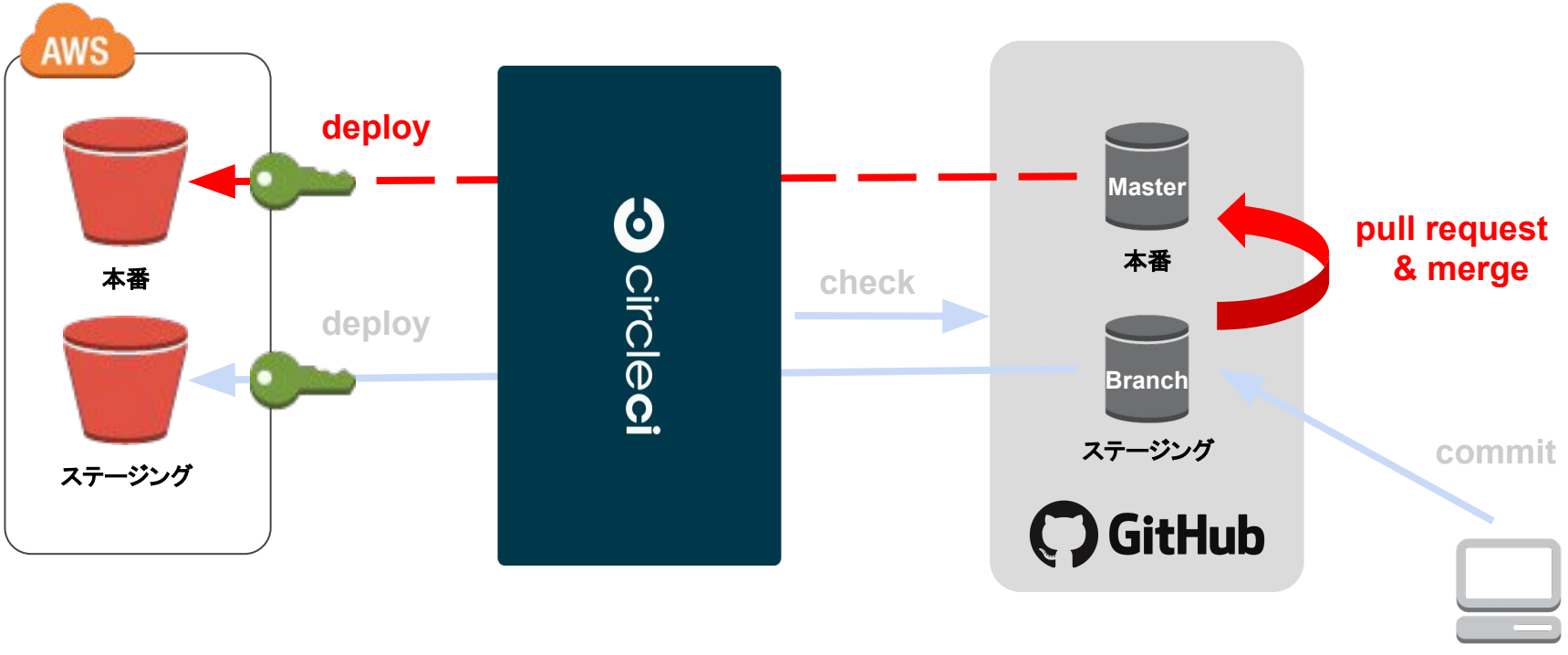
6-5. 確認

stagingのEndpointにアクセスして以下が表示されればOKです。

本番環境へデプロイ

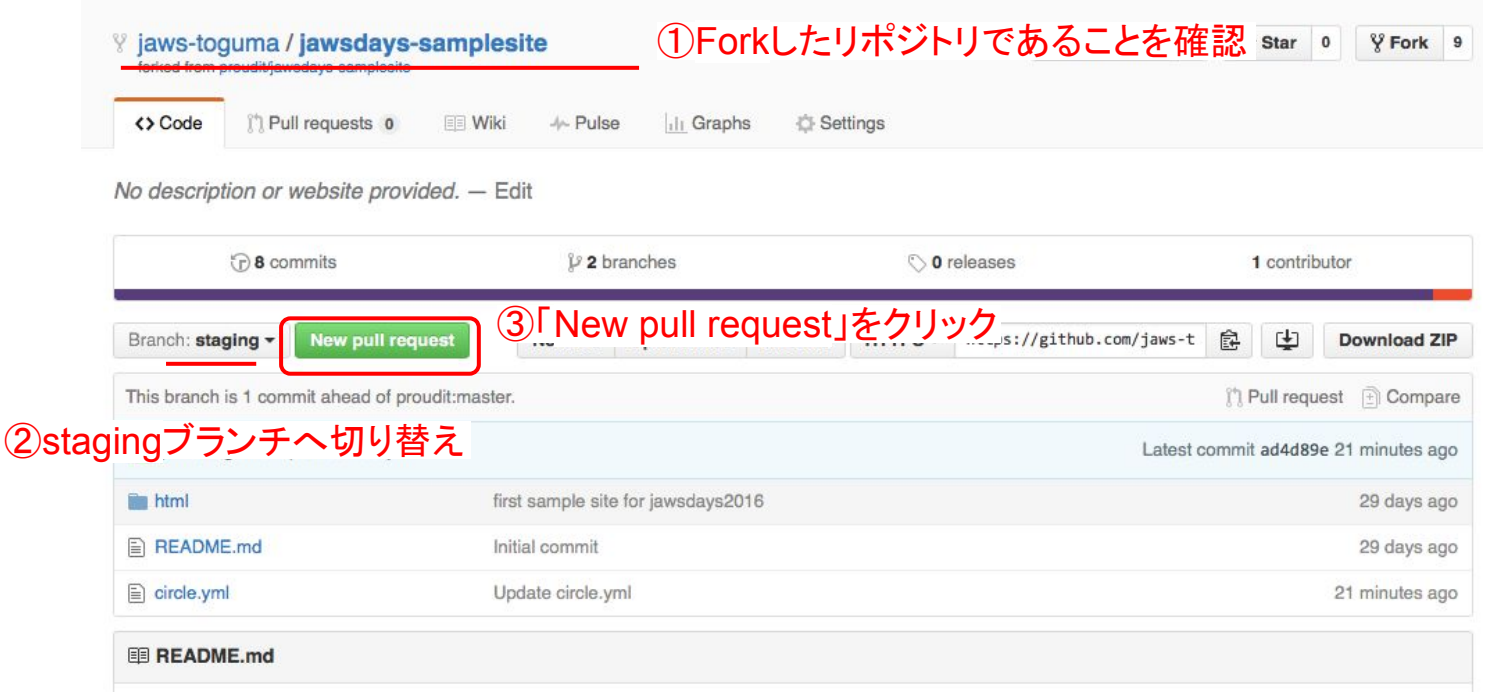
フェーズ7.

本番サイトへのデプロイ イメージ

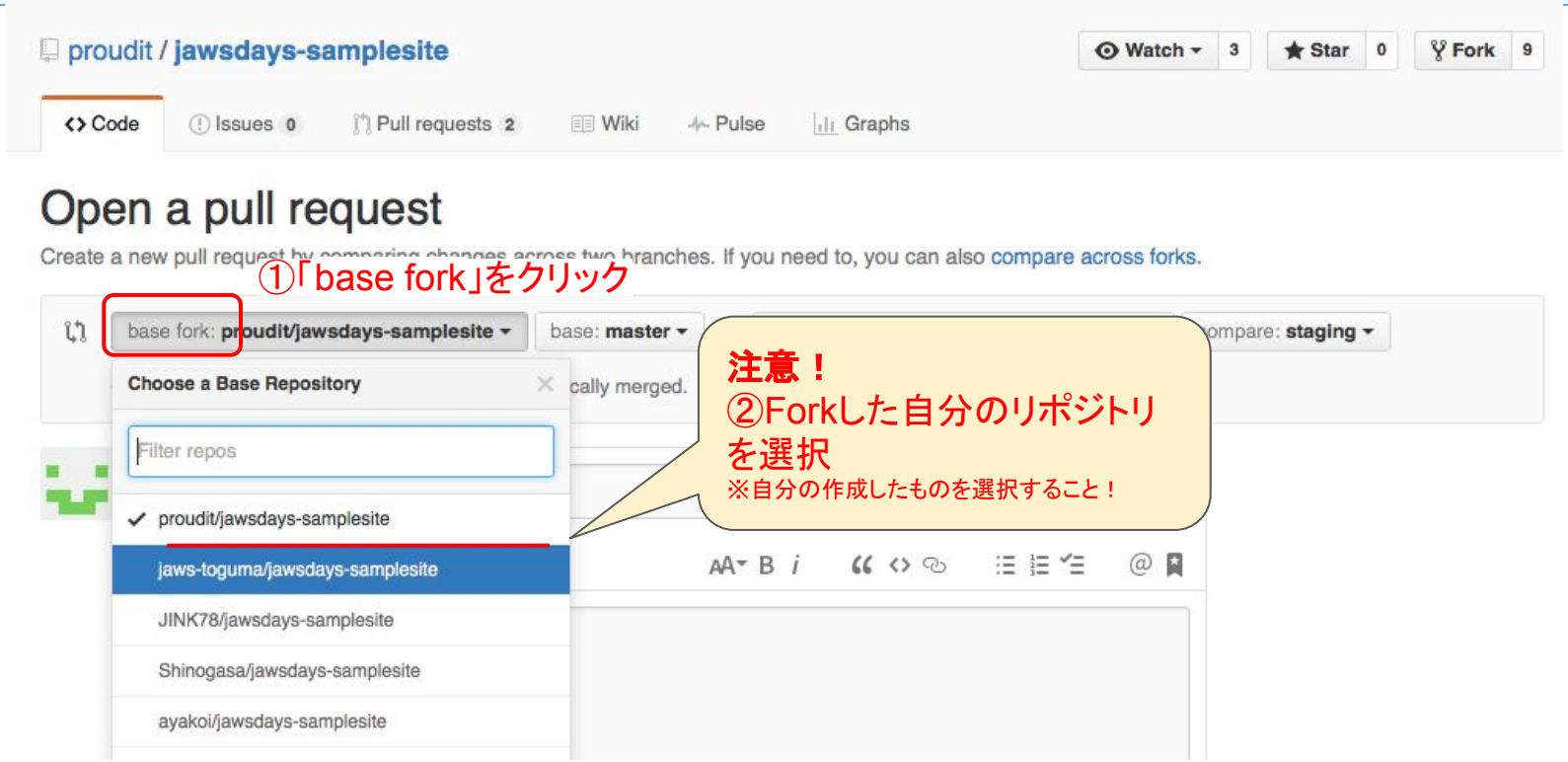


7-1. pull requestの作成①

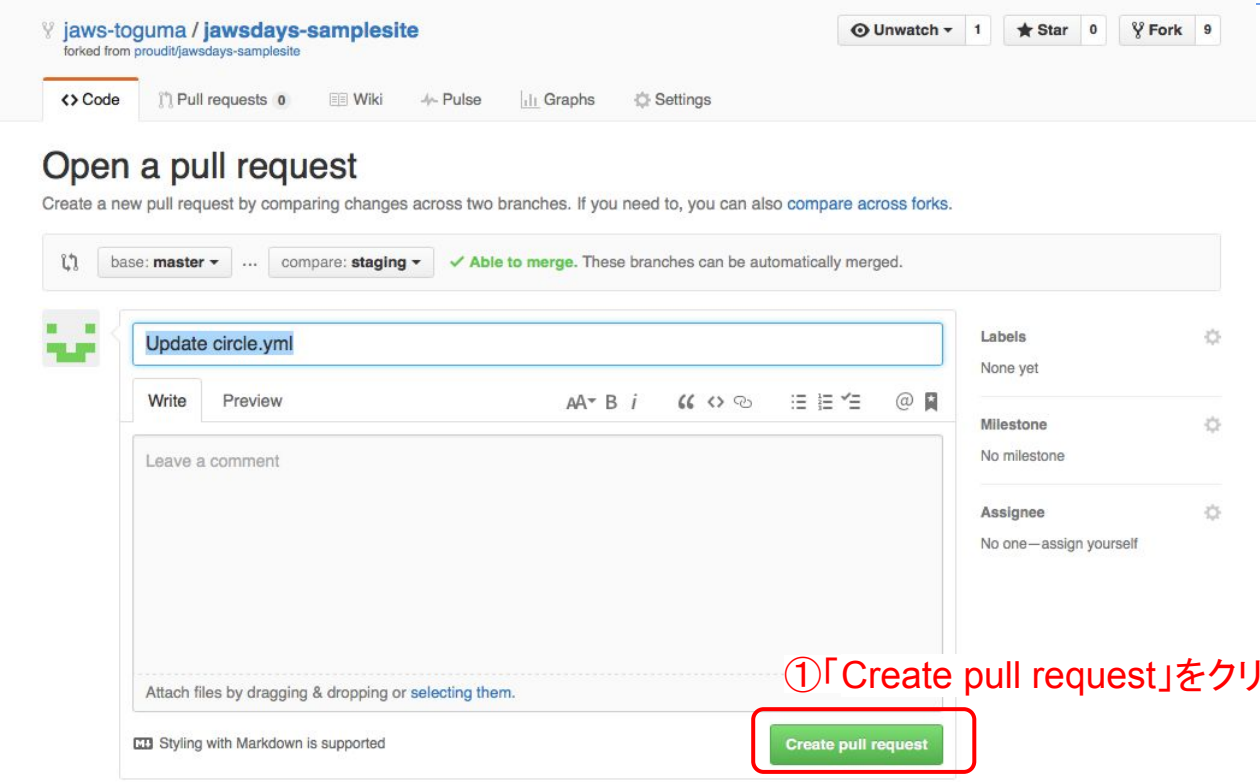
GitHubへアクセスし、Forkしたリポジトリへ移動しプルリクエストを作成します。



7-2. pull requestの作成②



7-3. pull requestの作成③



①「Create pull request」をクリック

7-4. masterブランチへmerge①

7-5. masterブランチへmerge②

7-6. 確認①

Githubにて「merge」されると自動でCircleCIが走ります。

CircleCIの管理画面からステータスが確認できます。

7-7. 確認②

本番のEndpointにブラウザアクセスしてサイトが表示されればOKです。

Welcome JawsDays Workshop!!

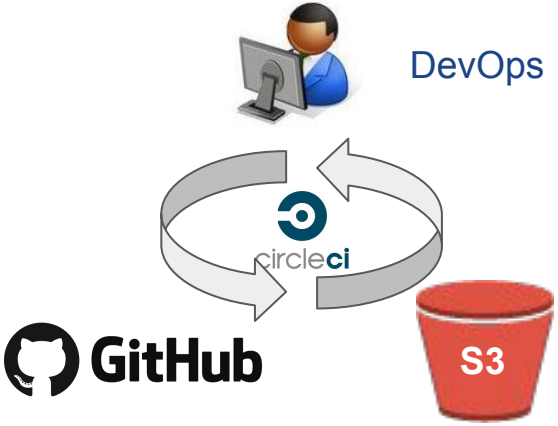
静的HTML公開フローをサーバレスでDevOps！

8-1　静的HTML公開フローをDevOps!

AWS S3静的ホスティングというスケーラブルな Webサーバ上のDevOps環境が完成です！

早速DevOpsを実践しましょう！

1. コーダー（制作者）が Githubの[staging]ブランチを修正し「push」
   →自動でステージングサイトに Deploy
2. Githubにて[master]ブランチへ「pull request」＝プルリク
3. プルリクを受けビューワー（責任者）がステージングサイトを確認。
4. OKならGithubにて「merge」をポチる。
   →自動で本番サイトに Deploy
5. 1へ戻る。



終了後

以下の設定を参考に必要に応じてリソースを削除してください。

1. 【CircleCI】プロジェクトの削除
   - 作成済のプロジェクトを以下の手順を参考に削除
     - Buildsより対象プロジェクトのedit【歯車アイコン】を選択後、Stop Building on Circleを選択
2. 【GitHub】リポジトリの削除
   - Forkにて作成済のリポジトリを以下のサイトを参考に削除
     - http://qiita.com/PlanetMeron/items/4d164eff7bff2243cf06
3. 【AWS】S3バケットおよびIAMユーザー、IAMポリシー削除
   - 作成済のS3バケットを以下のサイトを参考に削除
     - https://docs.aws.amazon.com/ja_jp/gettingstarted/latest/swh/getting-started-cleanup-s3.html
   - 作成済のIAMユーザーおよびIAMポリシーを以下のサイトを参考に削除
     - http://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/id_users_manage.html#id_users_deleting
     - http://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/access_policies_managed-using.html#delete-managed-policy