



**Exastro**

# EPOCH Quick Start

第0.1.0版

Exastro developer

# 目次

## 1. はじめに

1.1 QuickStartについて

1.2 QuickStartを実施するPC環境について

## 2. インストール

2.1 EPOCHのインストール

2.2 リポジトリ準備

2.3 Manifestテンプレートファイル準備

## 3. ワークスペース作成

3.1 ワークスペース

3.2 CI/CDについて

3.3 EPOCHのCI/CD

3.4 EPOCH起動

3.5 ワークスペース作成

## 4. チュートリアル

4.1 サンプルアプリの構成

4.2 チュートリアルの流れ(CI/CDワークフロー)

4.3 Manifestテンプレートファイルについて

4.4 1回目のCI/CDワークフロー手順

4.5 2回目のCI/CDワークフロー手順

## 5. 付録

5.1 注意事項・制限事項

# 1. はじめに

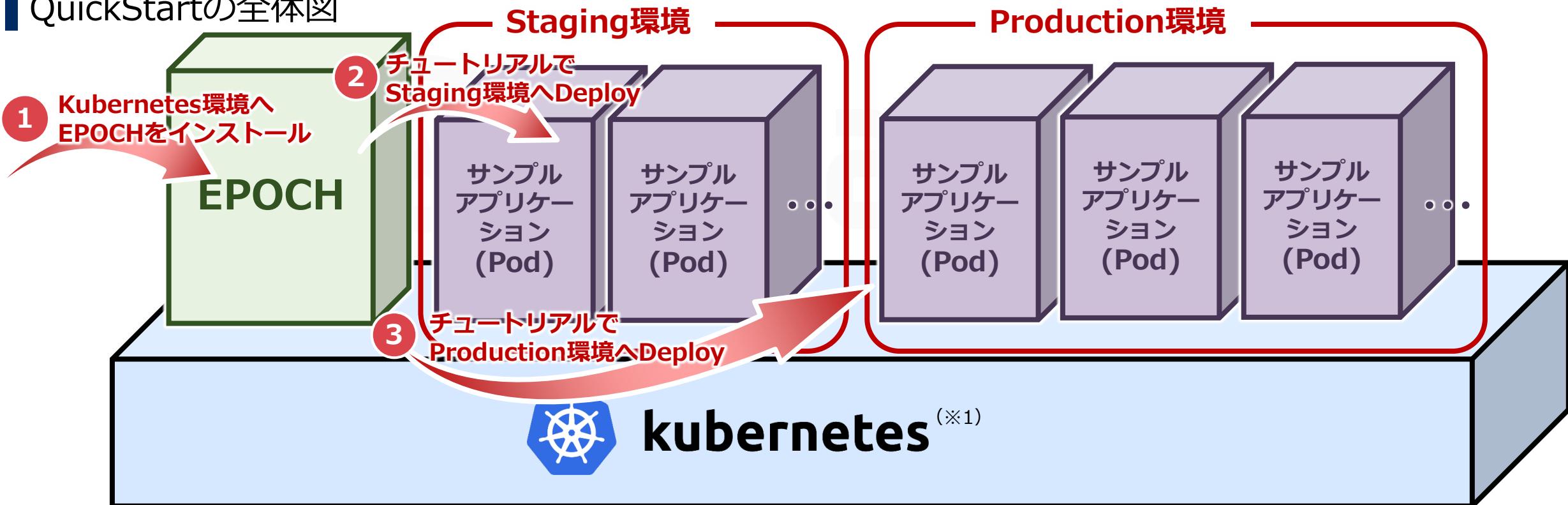


# 1.1 QuickStartについて

## はじめに

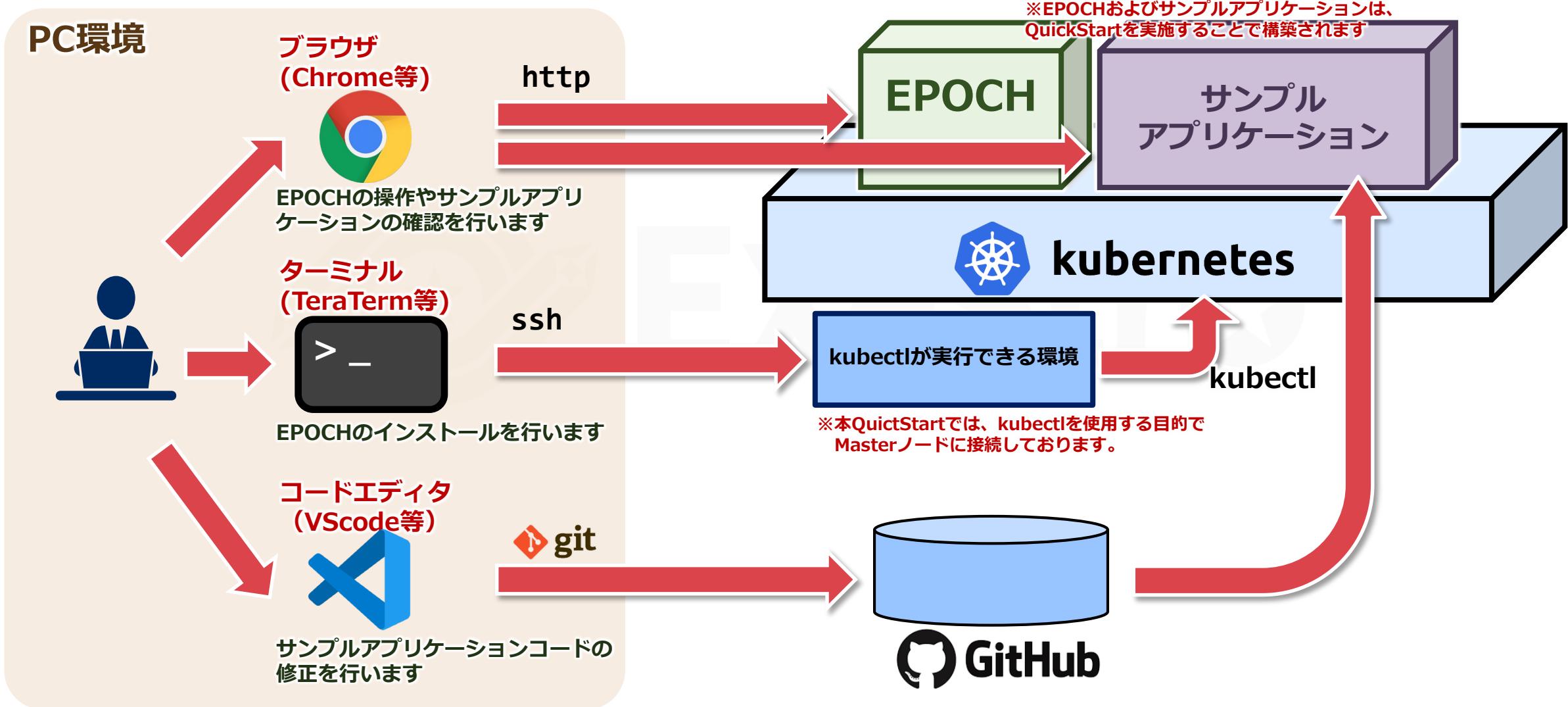
本書は、Exastro EPOCH(以降、EPOCHと表記する)の導入方法ならびに簡単な使い方をチュートリアルを用いて説明します。

## QuickStartの全体図



## 1.2 QuickStartを実施するPC環境について

QuickStartの手順を実施するにあたってのPCのソフトウェアは以下の通りです。





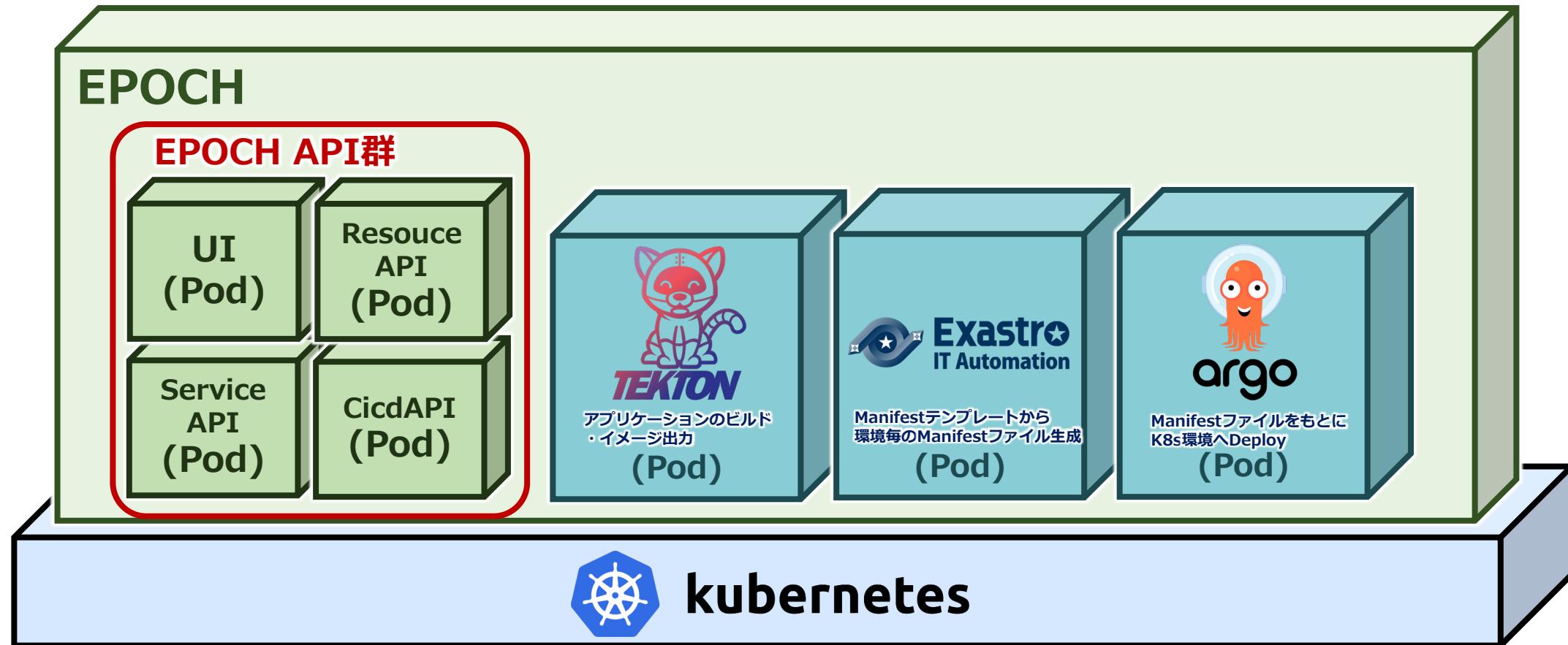
## 2. インストール

EPOCHをインストールして、CI/CDの環境を準備をしましょう。

## 2.1 EPOCHのインストール(1/5)

### EPOCH全体図

EPOCHをインストールおよびワークスペースを作成した後の構成は、以下の図のようになります。



## 2.1 EPOCHのインストール(2/5)

### ■ 前提条件

- 環境
  - Kubernetes環境が構築されていること
  - 使用するServiceAccountにcluster-adminロールが付与されていること
  - Kubernetes環境から外部インターネットに接続できること
  - PC環境から外部インターネットに接続できること
  - ポート番号(30080, 30081, 30443, 30801 , 30804, 30805, 30901～30907)が使用できること  
(ポート番号はepoch-install.yamlに記述されており、変更する際は編集後インストールを実行する必要があります)

- アカウント
  - アプリケーションコードを登録するGitHubのアカウントが準備されていること
  - Kubernetes Manifestを登録するGitHubのアカウントが準備されていること
  - コンテナイメージを登録するDockerHubのアカウントが準備されていること

## 2.1 EPOCHのインストール(3/5)

### EPOCHインストール

- ターミナルでkubectlが実行できる環境にSSHログインし、以下のコマンドを実行してEPOCHをインストールします。

```
$ kubectl apply -f https://github.com/exastro-suite/epoch/releases/download/v0.1.0/epoch-install.yaml
```

以下のコマンドでインストールの進行状況を確認できます。

```
$ kubectl get pod -n epoch-system
```

コマンド結果に表示されているすべてのコンポーネントのSTATUSが“Running”であることを確認します。

#### 【コマンド結果 イメージ】

NAME	READY	STATUS	RESTARTS	AGE
epoch-cicd-api-*****-****	1/1	Running	0	**s
epoch-rs-organization-api-*****-****	1/1	Running	0	**s
epoch-rs-workspace-api-*****-****	1/1	Running	0	**s
~				

## 2.1 EPOCHのインストール(4/5)

### 永続ボリューム設定

パイプライン設定用の永続ボリュームを設定します。

- 以下のコマンドを実行し、マニフェストをGitHubから取得します。

```
$ curl -OL https://github.com/exastro-suite/epoch/releases/download/v0.1.0/epoch-pv.yaml
```

- 以下のコマンドを実行し、Workerノードのホスト名を確認します。

```
$ kubectl get node
```

#### 【コマンド結果 イメージ】

NAME	STATUS	ROLES	AGE	VERSION
epoch-kubernetes-master1	Ready	control-plane,master	**d	v1.*.*.*
epoch-kubernetes-worker1	Ready	worker	**d	v1.*.*.*

- epoch-pv.yamlを修正します。（修正箇所はepoch-pv.yamlの最終行）

「# Please specify the host name of the worker node #」の部分を、先ほど確認したWorkerノードのホスト名に置き換え保存します。

#### 【変更イメージ】

```
values:  
- # Please specify the host name of the worker node #
```

```
values:  
- epoch-kubernetes-worker1
```

- 以下のコマンドでkubernetes環境へ反映します。

```
$ kubectl apply -f epoch-pv.yaml
```

## 2.1 EPOCHのインストール(5/5)

### ArgoRolloutインストール

- 以下のコマンドを実行し、ArgoRolloutのインストールします。

```
$ kubectl create namespace argo-rollouts  
$ kubectl apply -n argo-rollouts -f https://github.com/argoproj/argo-rollouts/releases/latest/download/install.yaml
```

以上でEPOCHのインストールは完了しました。  
次にチュートリアルを実施するための事前準備を実施しましょう！

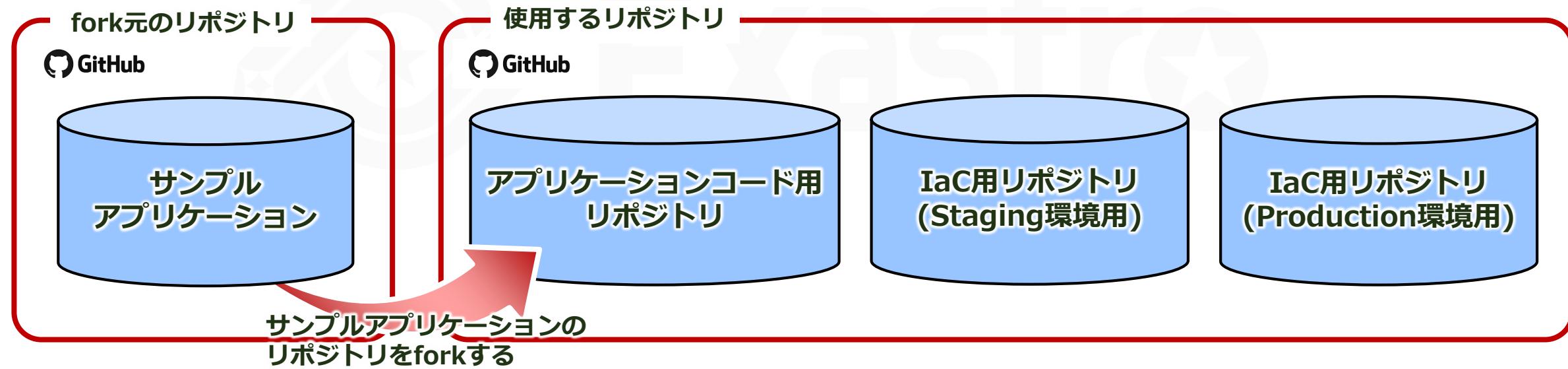
## 2.2 リポジトリ準備(1/4)

### 使用するリポジトリについて

- 本クイックスタートで使用するリポジトリは以下の通りです。

- アプリケーションコード用リポジトリ
- IaC用リポジトリ(Staging環境用)
- IaC用リポジトリ(Production環境用)

- イメージ図



## 2.2 リポジトリ準備(2/4)

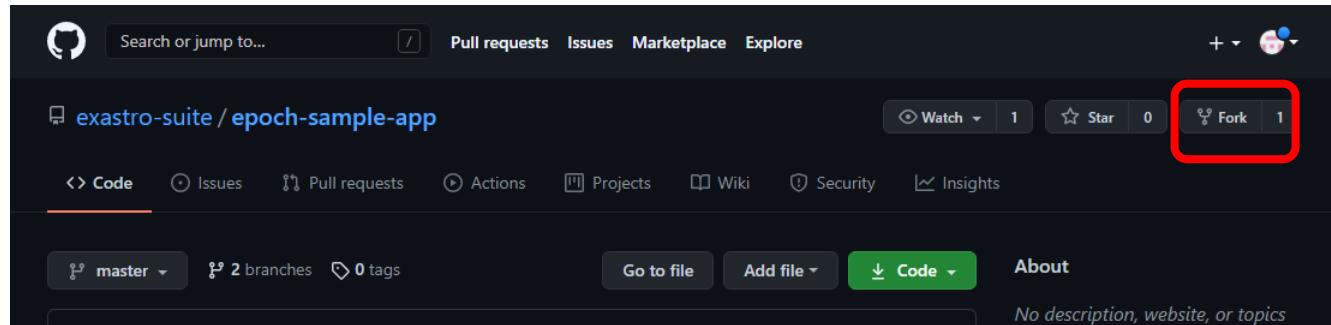
### ■ アプリケーションコード用リポジトリの準備

- ブラウザにてサンプルアプリケーションのURLを表示します。

<https://github.com/exastro-suite/epoch-sample-app>

- サンプルアプリケーションの画面から「Fork」を押下して、アプリケーションコード用リポジトリを作成します。

(「Fork」押下後にGitHubのサインインを求められたときはご自身のGitHubアカウントでサインインしてください)



- アプリケーションコード用リポジトリのclone

アプリケーションコード用リポジトリをPC環境にcloneします。

例としてコマンドプロンプトでは、以下の通りとなります。

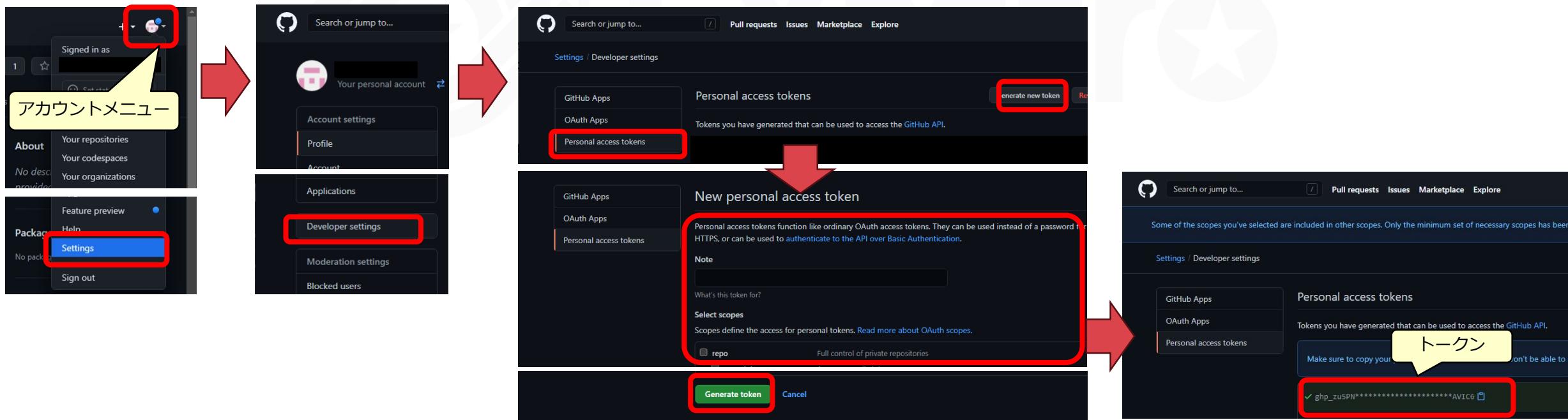
```
> cd "[clone先のフォルダ]"
> git clone https://github.com/[Githubのアカウント名]/epoch-sample-app.git
> cd epoch-sample-app
> git config user.name "[GitHubのユーザ名]"
> git config user.email "[GitHubのemailアドレス]"
```

ここでcloneしたソースコードを使って、チュートリアルを行います。

## 2.2 リポジトリ準備(3/4)

### Git トークンの払い出し

- ブラウザにて自身のGitHubのアカウントでGitHubにサインインします。
- アカウントメニューからSettingsを選択します。
- Account settings画面からDeveloper settingsメニューを選択します。
- Developer settings画面からPersonal access tokensメニューを選択し、Generate new tokenボタンを選択します。
- New personal access token画面でNote(任意の名称)、Select scopesを選択し、Generate tokenボタンを選択します。
- 表示されたトークン(ghp\_\*\*\*\*\*)を後に使用しますので控えてください。

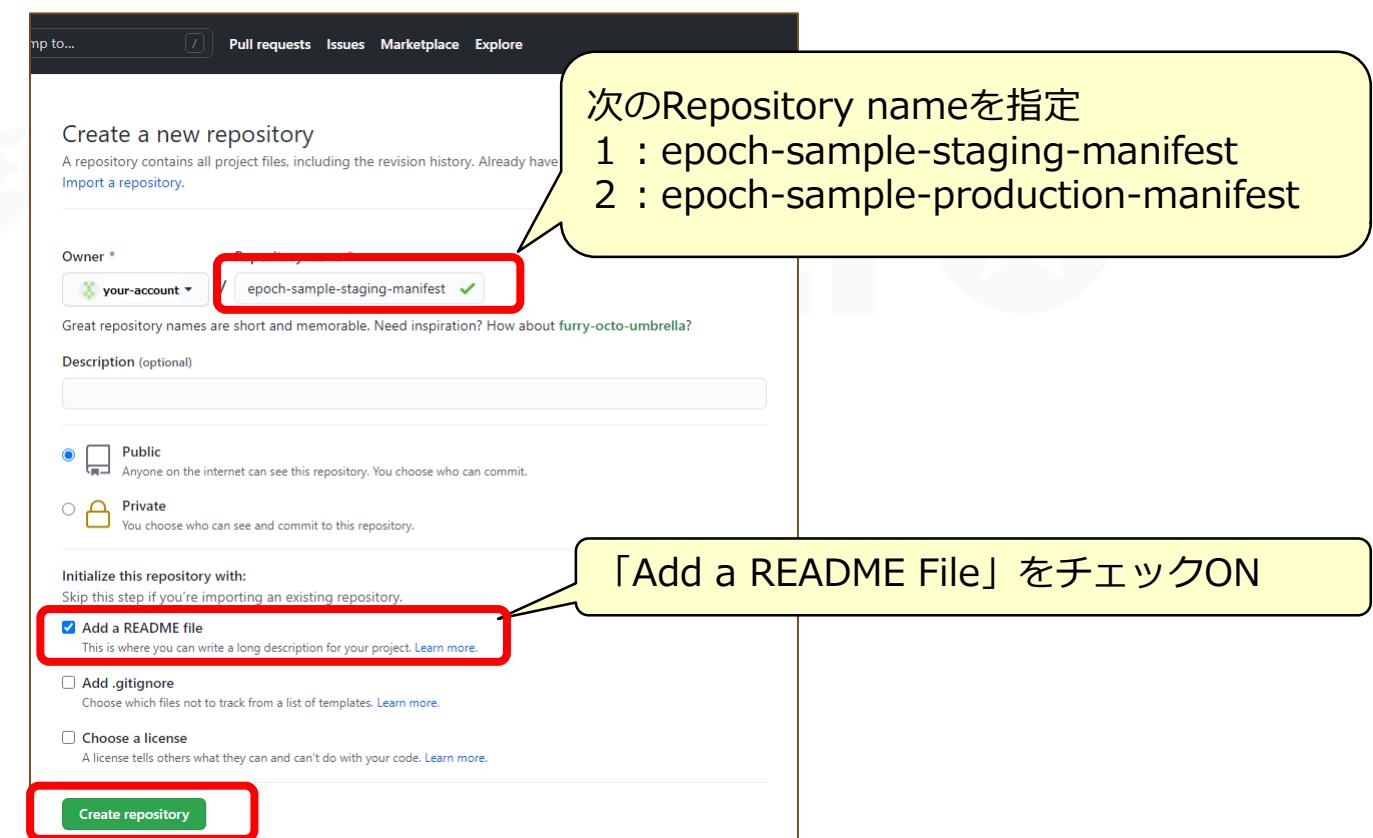
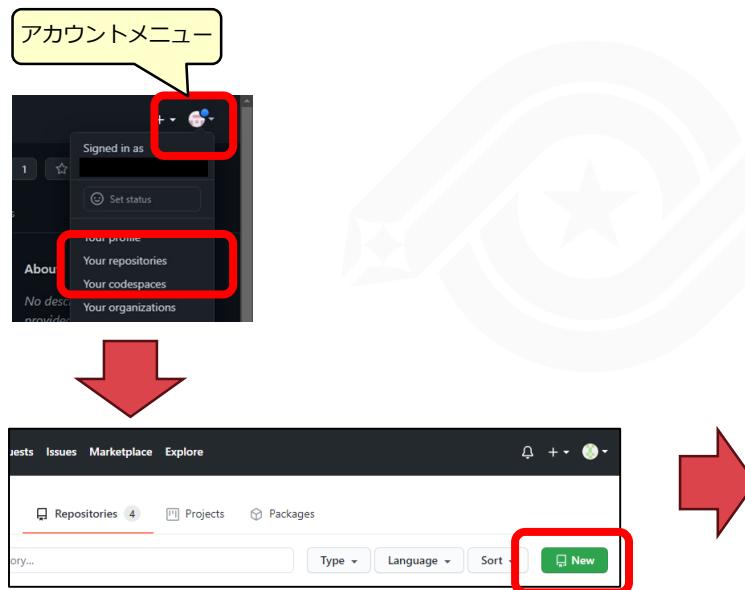


## 2.2 リポジトリ準備(4/4)

### IaC用リポジトリの準備

Manifestを格納するGitリポジトリを2つ用意します。

- ブラウザにて自身のGitHubのアカウントでGitHubにサインインします。
- アカウントメニューからYour Repositoriesを選択します。
- Newを選択し、図で示した値を入力し、Create repositoryを選択します。



## 2.3 Manifestテンプレートファイルの準備

### Manifestテンプレートファイルのダウンロード

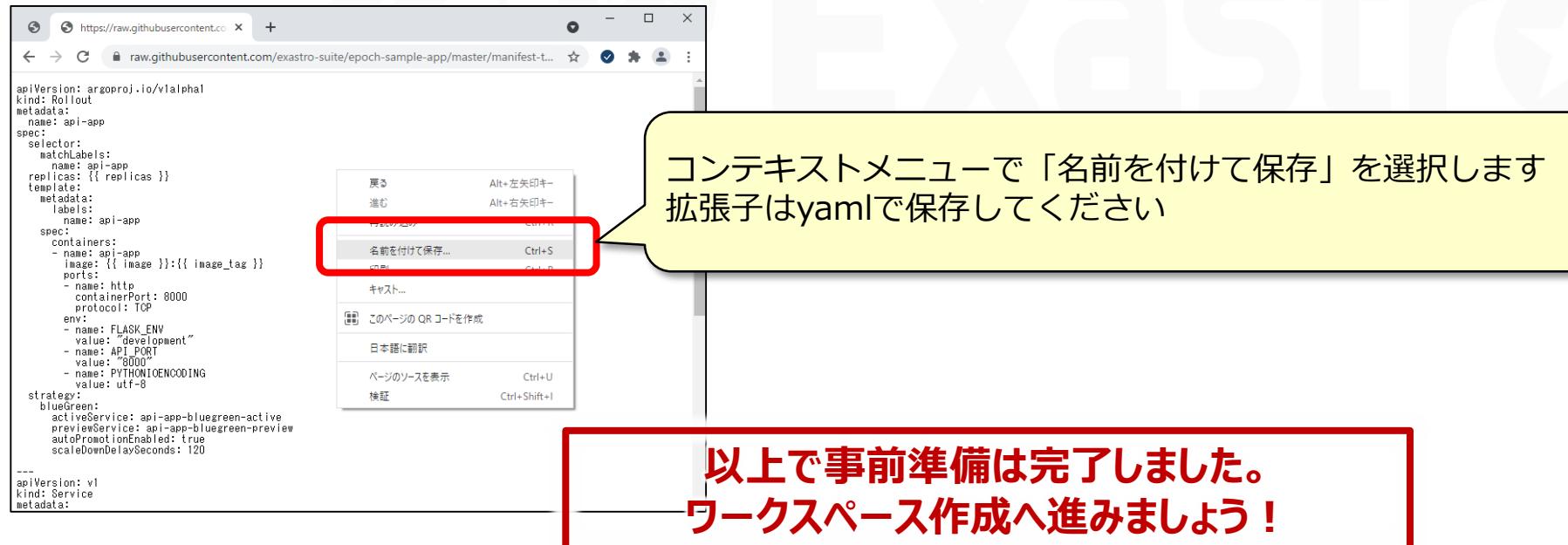
EPOCHにアップロードするManifestテンプレートファイル（2ファイル）をダウンロードします。

- ブラウザで以下のURLを表示します。

ファイル1 : <https://raw.githubusercontent.com/exastro-suite/epoch-sample-app/master/manifest-template/api-app.yaml>

ファイル2 : <https://raw.githubusercontent.com/exastro-suite/epoch-sample-app/master/manifest-template/ui-app.yaml>

- ブラウザにManifestテンプレートが表示されますので、操作しているPCに保存します。





### 3. ワークスペース作成

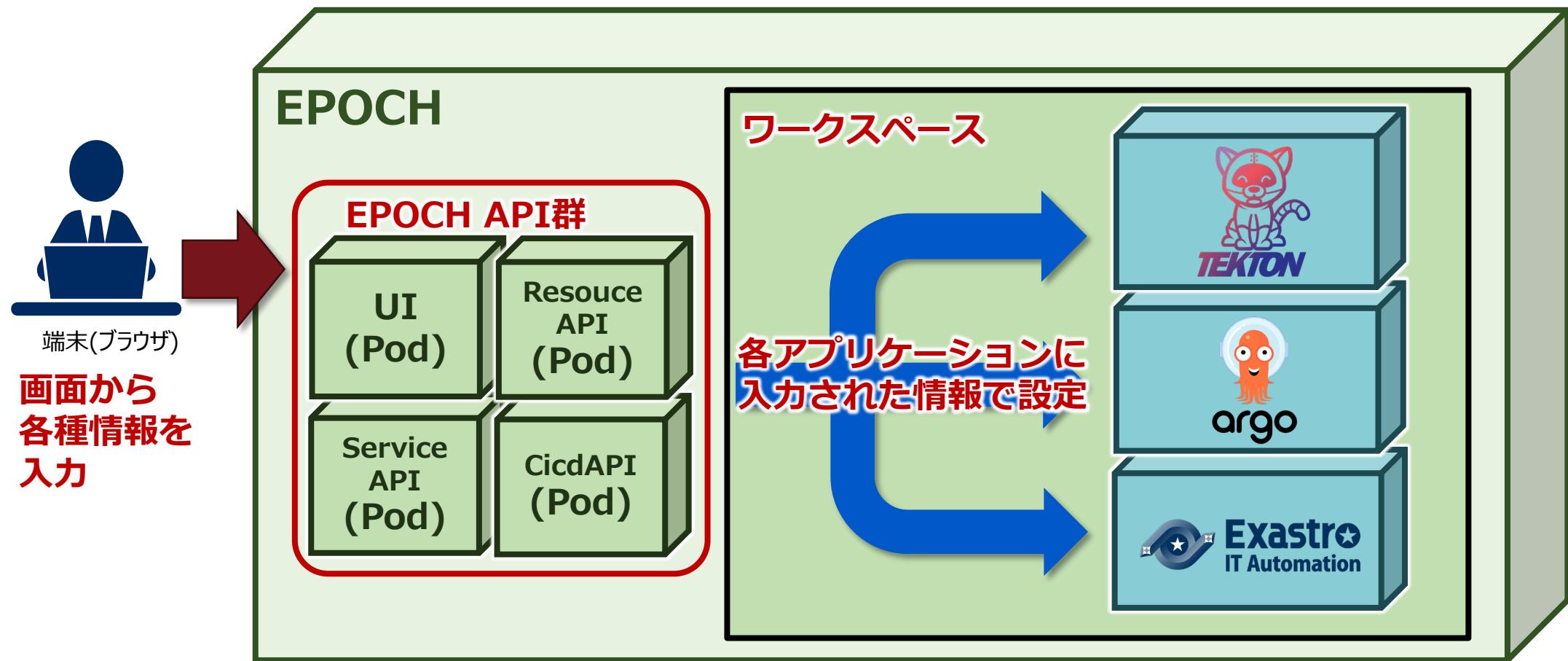
ワークスペースを作成し、CI/CDの準備をしましょう。

### 3.1 ワークスペース

#### ワークスペース

EPOCHでは、1つの開発環境をワークスペースという単位で管理します。

ワークスペース作成は、画面から入力された情報をもとに、各アプリケーションへ必要な情報を登録し、CI/CDの準備を行ないます。



## 3.2 CI/CDについて

### CI/CDとは

アプリケーションの開発～リリースまでの一連の作業を自動化し、アプリケーション提供の頻度を高める手法です。

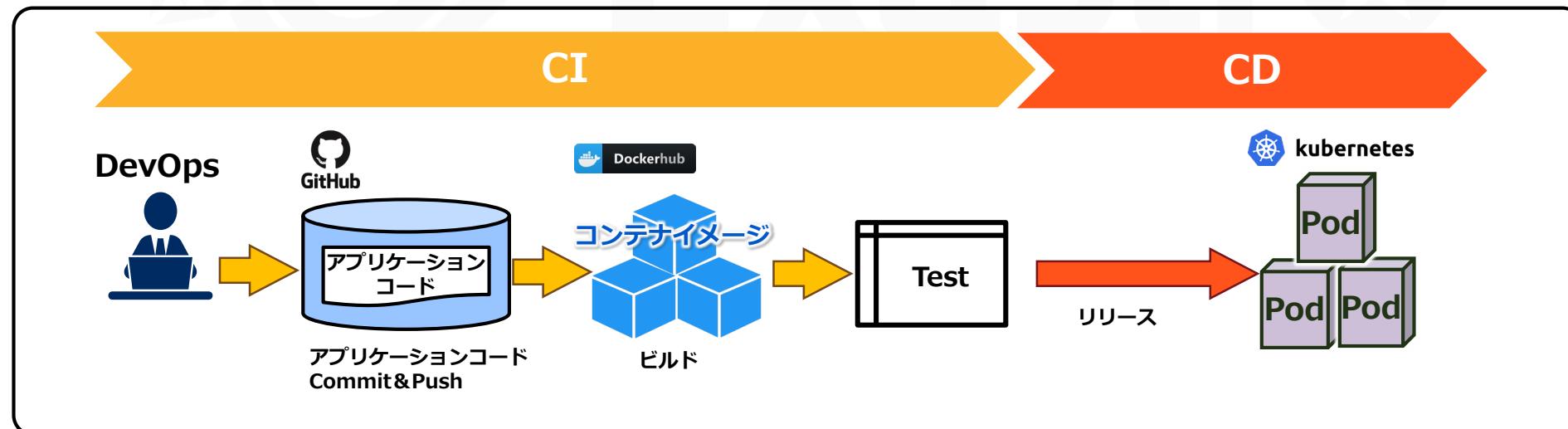
- CI（継続的インテグレーション）

アプリケーションコードの変更を起点に、ビルドやテストの実行といった開発者の作業を自動化する手法を指します。

- CD（継続的デリバリー）

実行環境へのリリースまでを自動化する手法を指します。

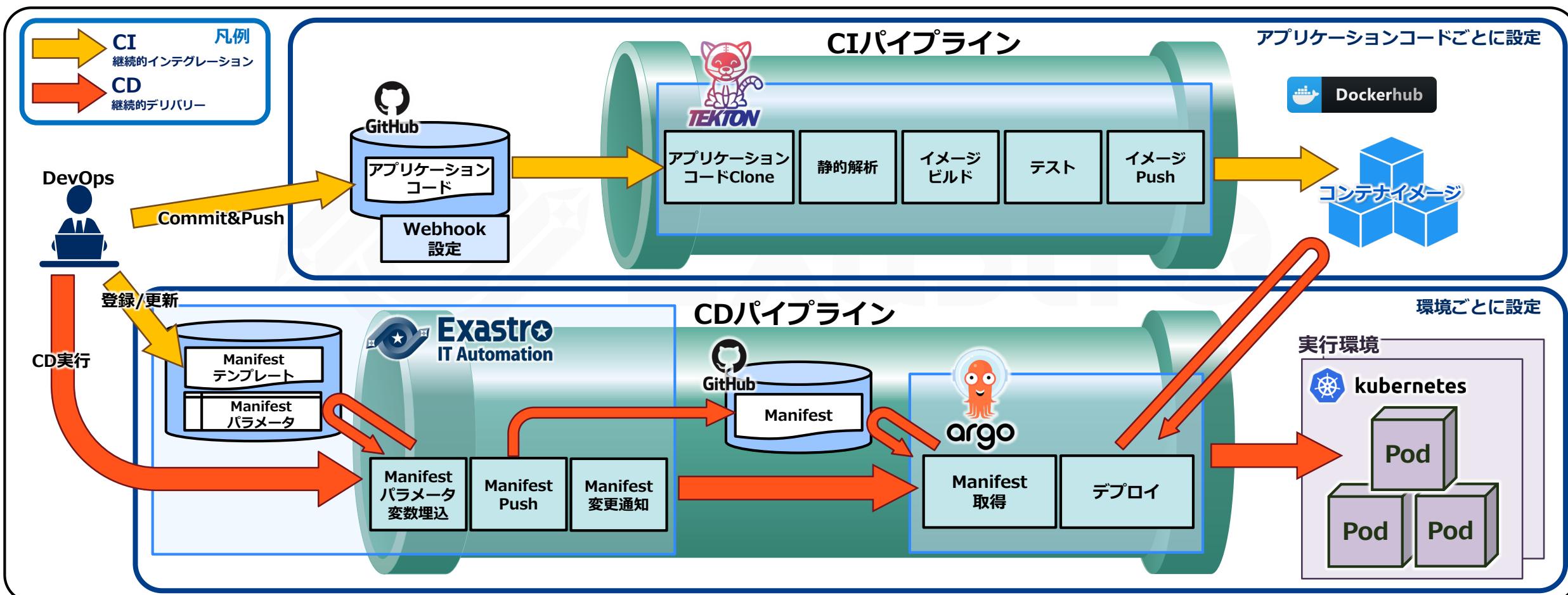
- CI/CDのイメージ



### 3.3 EPOCHのCI/CD

#### EPOCHのCI/CD

EPOCHのCI/CDの流れを、下図に示します。



### 3.4 EPOCH起動

ブラウザより以下のURLで接続します。

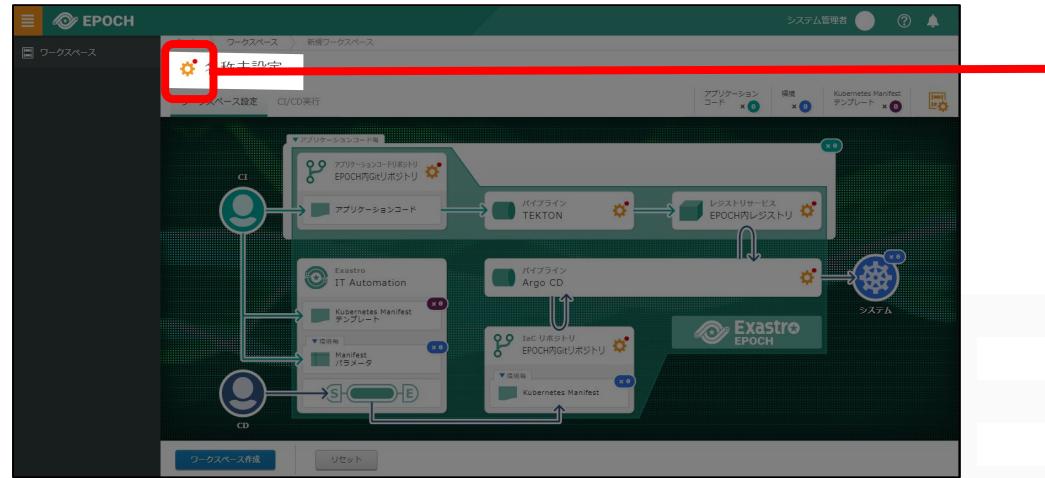
**https://[インストール先のIPアドレスまたはホスト名]:30443/workspace.html**



### 3.5 ワークスペース作成(1/7)

#### ワークスペース基本情報

ワークスペース名を入力します。



項目	入力・選択内容	説明
ワークスペース名	EPOCHクイックスタート	作成するワークスペース名
備考	なし	作成するワークスペースの説明や備考

## 3.5 ワークスペース作成(2/7)

### アプリケーションコードリポジトリ

アプリケーションコードリポジトリの情報を入力します。

GitHubを選択

Gitサービス選択

GitHub

Gitアカウント指定

ユーザ名: your-github-account

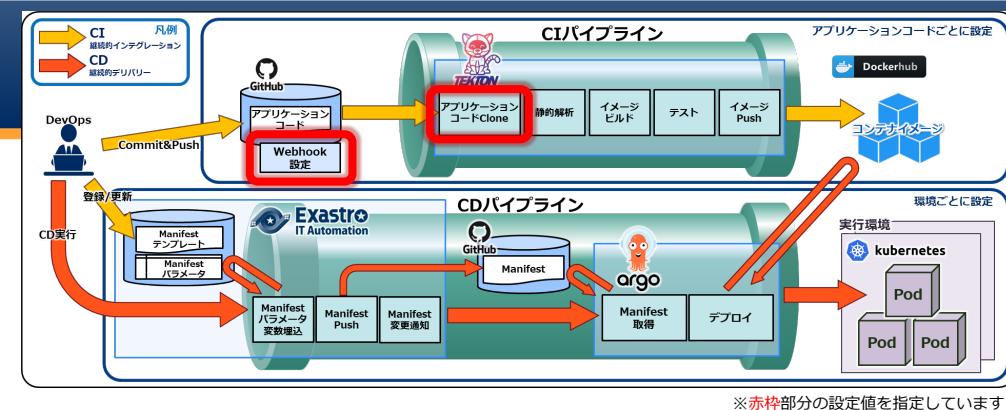
トークン: ..... (redacted)

Gitリポジトリ一覧

epoch-sample-app

Gitリポジトリ URL: https://github.com/your-github-account/epoch-sample-app.git

決定 キャンセル

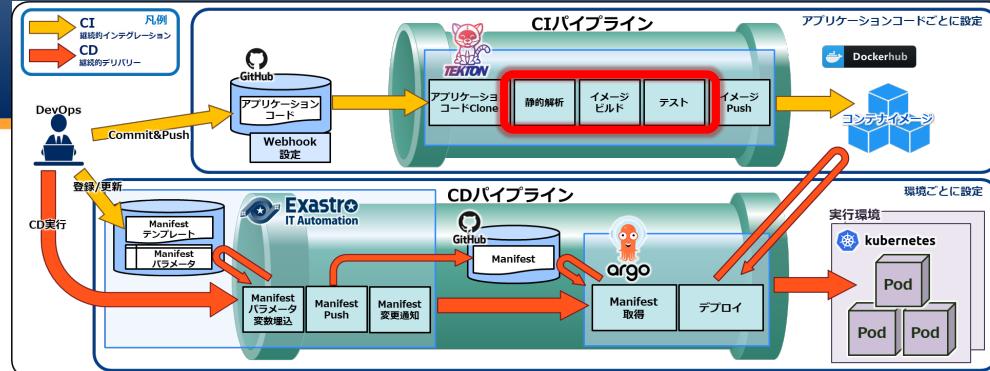
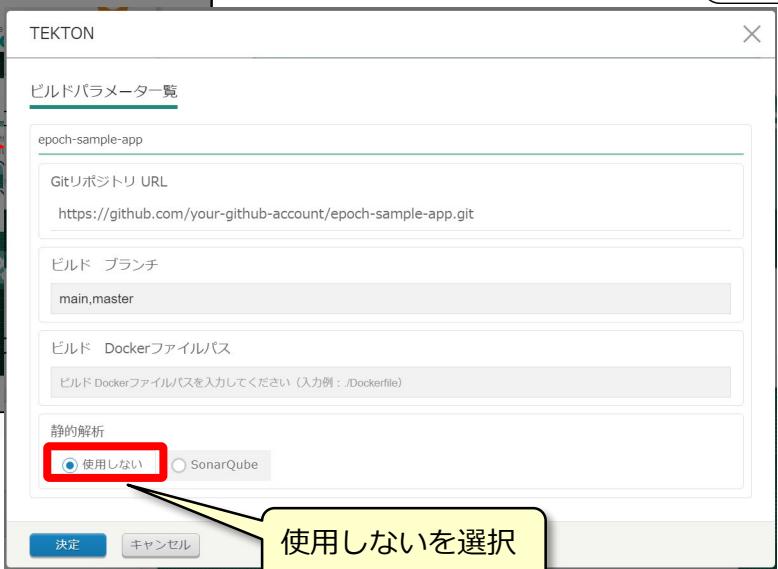
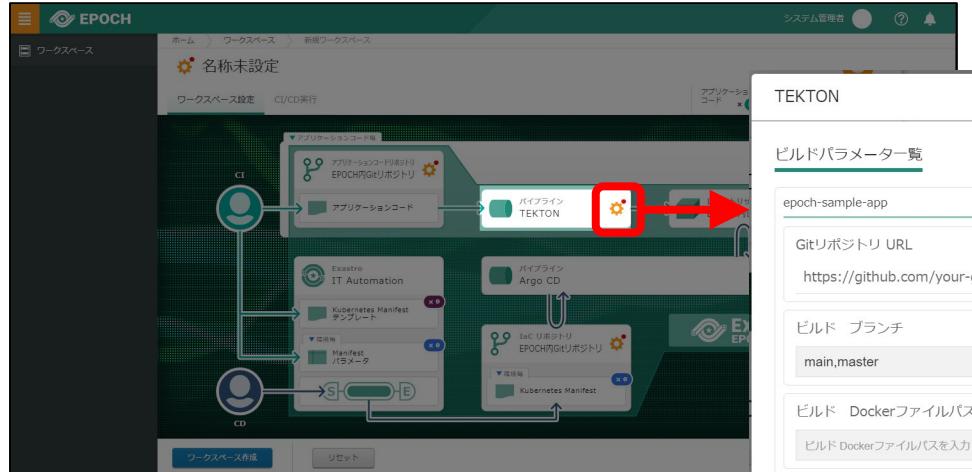


項目	入力・選択内容	説明
ユーザ名	(自身のGitHubのアカウント名)	GitHubのアカウント名
トークン	(自身のGitHubのトークン)	GitHubのトークン (事前準備 Gitトークンの払い出しを参照)
GitリポジトリURL	https://github.com/[GitHubのアカウント名]/epoch-sample-app.git	アプリケーションコードをforkしたリポジトリのURL

# 3.5 ワークスペース作成(3/7)

## パイプラインTEKTON

TEKTONに設定するパイプライン情報を入力します。



※赤枠部分の設定値を指定しています  
※静的解析、テスト等については対応する予定です

項目	入力・選択内容	説明
ビルドブランチ	main, master	ビルド対象のアプリケーションのGitHubのブランチ
ビルドDockerファイルパス	./api-app/Dockerfile	アプリケーションのDockerfileのパス

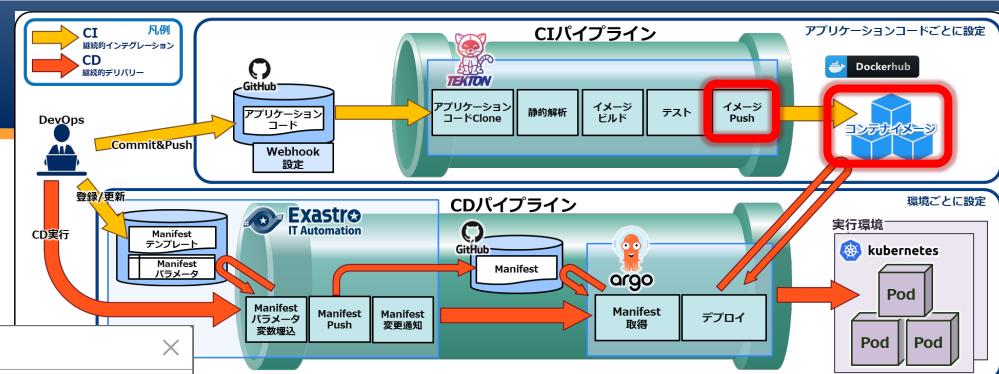
### 3.5 ワークスペース作成(4/7)

#### レジストリサービス

ビルド後のイメージ登録先（レジストリ）情報を入力します。

The screenshot shows the EPOCH UI for workspace creation. On the left, a large diagram illustrates the CI/CD pipeline flow from code repository to deployment. On the right, a detailed configuration dialog for 'Registry Service' is open. In the dialog, the 'DockerHub' option is selected under 'Registry Service Selection'. Below it, fields for 'User Name' (your-dockerhub-account) and 'Password' (redacted) are filled. Under 'Image Output Destination', the 'Git Repository URL' is set to <https://github.com/your-github-account/epoch-sample-app.git> and the 'Image Output Destination' is set to `your-dockerhub-account/epoch-sample-api`. At the bottom are '决定' (Confirm) and 'キャンセル' (Cancel) buttons.

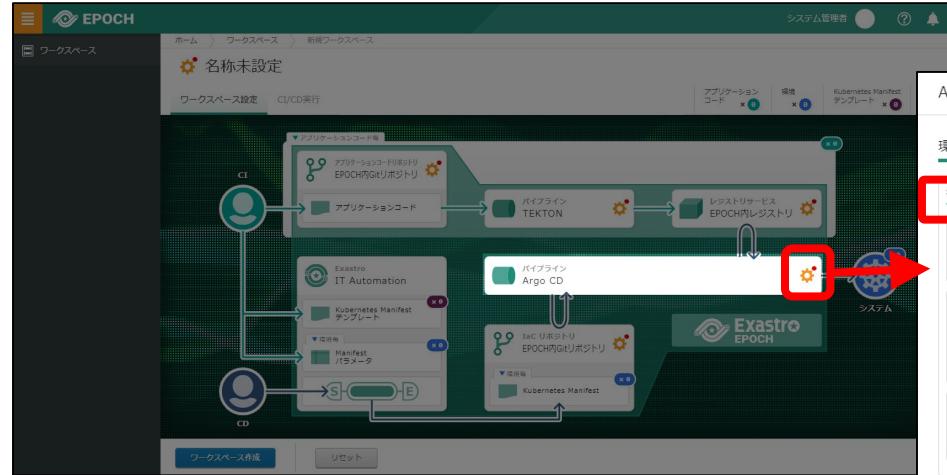
項目	入力・選択内容	説明
ユーザ名	(自身のDockerHubのアカウント名)	DockerHubのアカウント名
パスワード	(自身のDockerHubのパスワード)	DockerHubのパスワード
イメージ出力先	[DockerHubのアカウント名]/epoch-sample-api ※ユーザ名入力後に表示される内容を修正してください。	DockerHubのイメージ出力先のパス



# 3.5 ワークスペース作成(5/7)

## パイプラインArgo CD

ArgoCDに設定するDeploy先の情報を入力します。



環境1つめ2つめの切り替えはタブを選択して切り替えます

Argo CD

環境一覧

staging production

環境名: staging

Deploy先:

- EPOCHと同じKubernetes
- 以外のKubernetes

Namespace: epoch-sample-app-staging

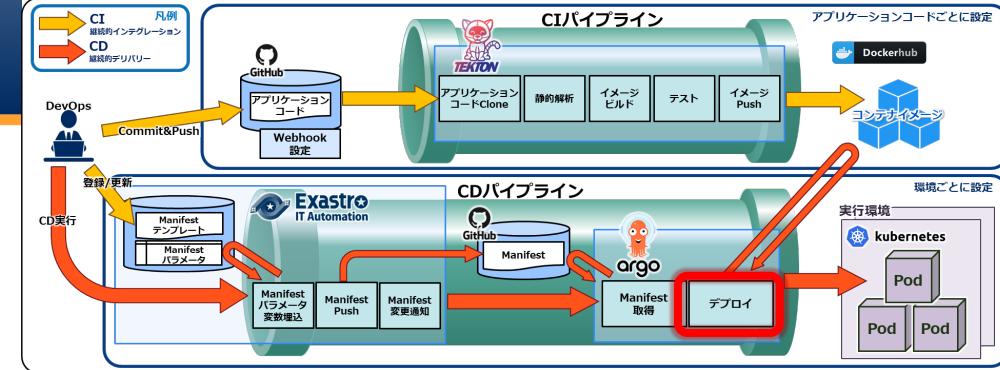
決定 キャンセル

環境1 : Staging環境

項目	入力・選択内容	説明
環境名	staging	デプロイ環境の名前
Namespace	epoch-sample-app-staging	デプロイ先のNamespace

環境2 : Production環境

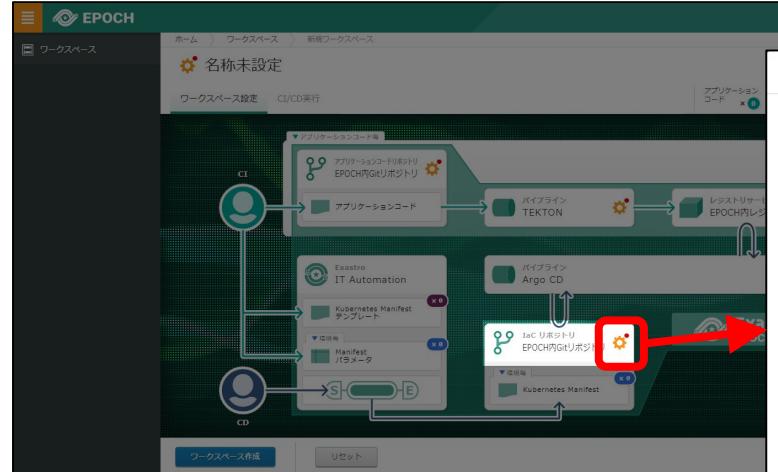
項目	入力・選択内容	説明
環境名	production	デプロイ環境の名前
Namespace	epoch-sample-app-production	デプロイ先のNamespace



# 3.5 ワークスペース作成(6/7)

## IaCリポジトリ

マニフェストの登録先となるリポジトリ情報を入力します。



**IaCリポジトリ**

GitHubを選択

Gitサービス選択

EPOCH内Gitリポジトリ  GitHub

Gitアカウント指定

Gitアカウント選択

アプリケーションコードリポジトリと同一  入力する

ユーザー名（アプリケーションコードリポジトリと同一）  
epoch-user

アプリケーションコードリポジトリ設定から参照しています。

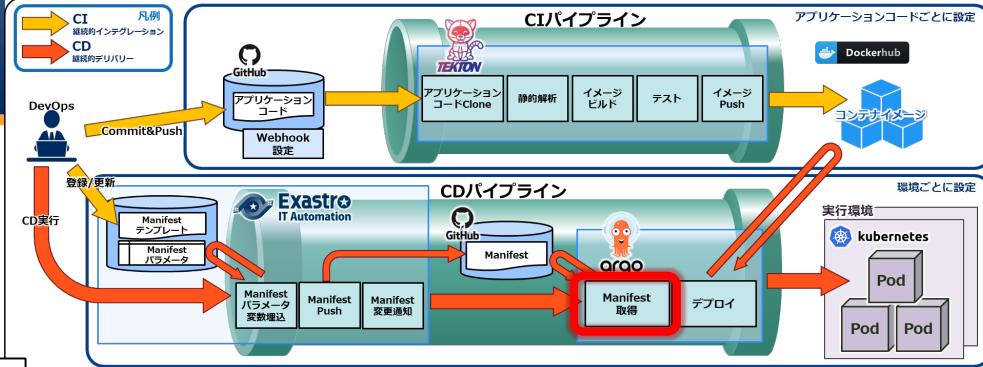
環境ごとに入力

Gitリポジトリ一覧

staging production

Gitリポジトリ URL  
Gitリポジトリ URLを入力してください

決定 キャンセル

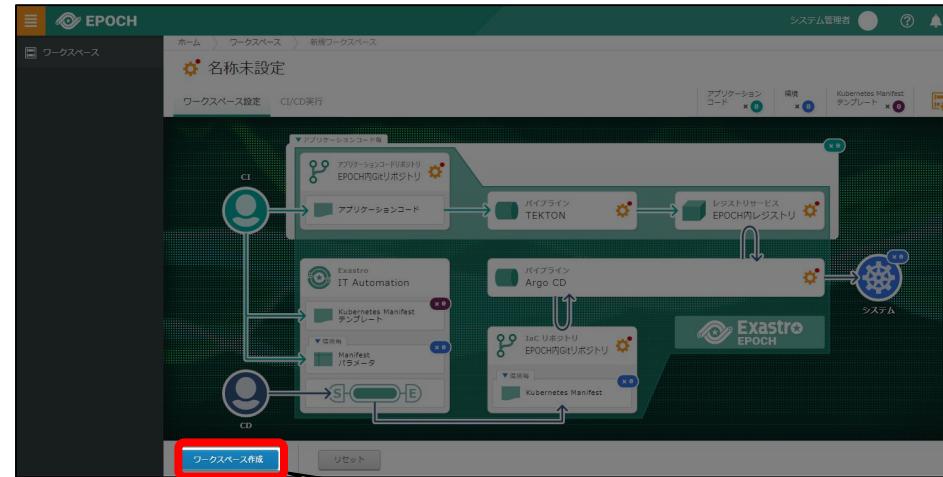


項目	入力・選択内容	説明
ユーザ名	(自身のGitHubのアカウント名)	GitHubのアカウント名
トークン	(自身のGitHubのトークン)	GitHubのトークン (事前準備 Gitトークンの払い出しを参照)
GitリポジトリURL	<a href="https://github.com/[GitHubのアカウント名]/[各環境のリポジトリ].git">https://github.com/[GitHubのアカウント名]/[各環境のリポジトリ].git</a>	各環境のmanifestリポジトリのURL (事前準備 IaC用リポジトリの準備を参照)

### 3.5 ワークスペース作成(7/7)

#### ワークスペース作成

すべての入力が完了しましたら【ワークスペース作成】ボタンを押下します。



【ワークスペース作成】ボタンを押下します

これでCI/CDパイプラインが構築されました。  
チュートリアルを実践してCI/CDパイプラインを体験してみましょう！



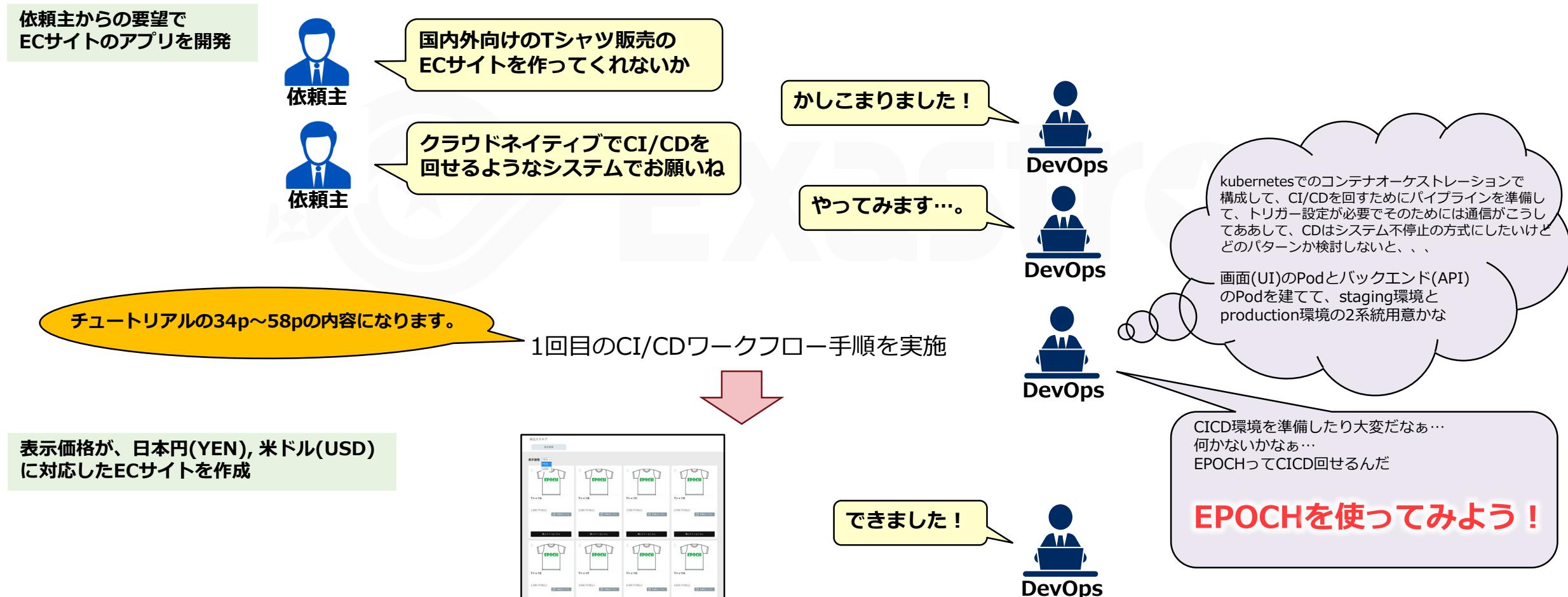
## 4. チュートリアル

CI/CDワークフローを体験してみましょう。

## 4.1 チュートリアルの概要(1/2)

## CI/CD開発シナリオ

チュートリアルでは以下のシナリオに沿って、CI/CDワークフローの手順を実施していきます。本QuickStartで作成するECサイトは、サンプルアプリケーションを用いて実施していきます。



## 4.1 チュートリアルの概要(2/2)

少し経ってから  
追加案件が発生



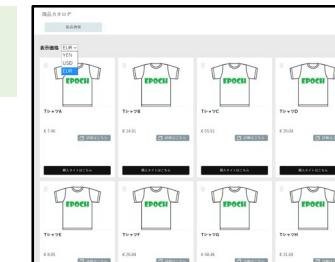
この前のECサイトはとても好評  
だったよ



ヨーロッパ向けにも展開したい  
通貨にEURも追加してほしい

チュートリアルの59p~80pの内容になります。

表示価格が、日本円(YEN), 米ドル(USD), **ユーロ(EUR)**  
に対応したECサイトを作成



2回目のCI/CDワークフロー手順を実施



ありがとうございます



かしこまりました！



CI/CDの仕組みはできているから  
コード修正してからデプロイまでは  
単純作業♪

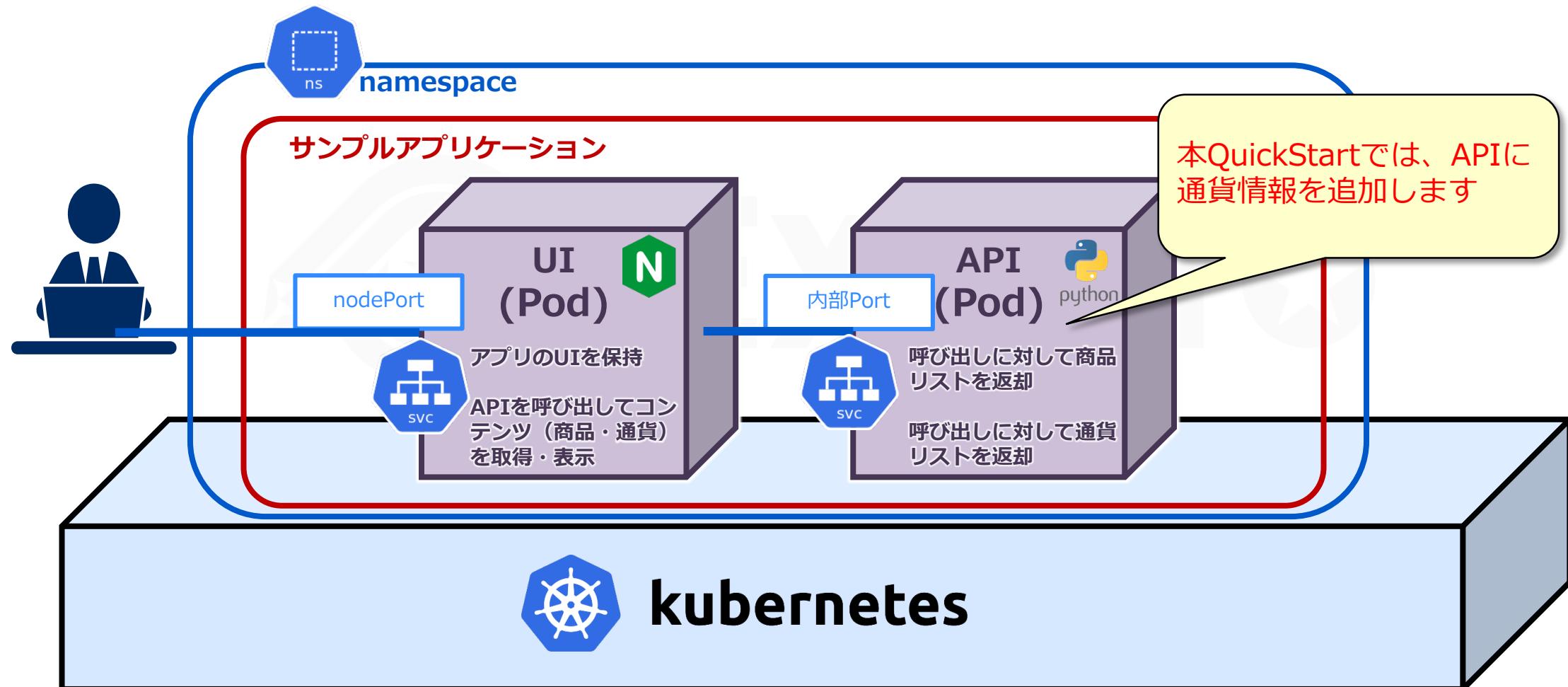
できました！



## 4.2 サンプルアプリの構成

### サンプルアプリの構成

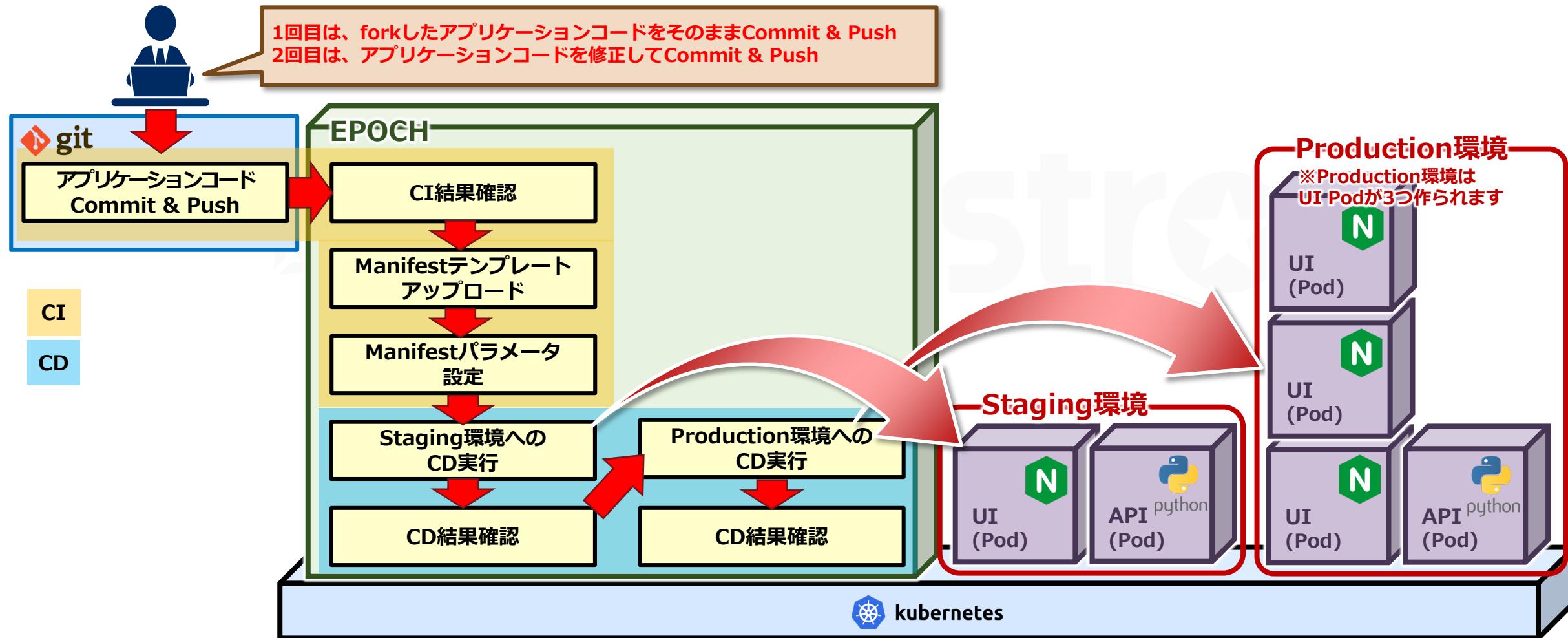
サンプルアプリケーションは、UIとAPIの2つで構成されております。



## 4.3 チュートリアルの流れ(CI/CDワークフロー)

### CI/CDワークフロー

本説明では、サンプルアプリケーションをStaging環境、Production環境へDeploy、その後アプリケーションコードの修正を行い、Staging環境、Production環境へのDeployする手順を説明していきます。



## 4.4 Manifestテンプレートファイルについて

### Manifestテンプレートファイル

サンプルアプリケーションのManifestテンプレートファイルは、UIとAPI用の2つが用意されています。環境一致を考慮した上で可変部分を変数化したテンプレート形式となっています。



それでは1回目のCI/CDワークフロー手順を  
実行してみましょう！

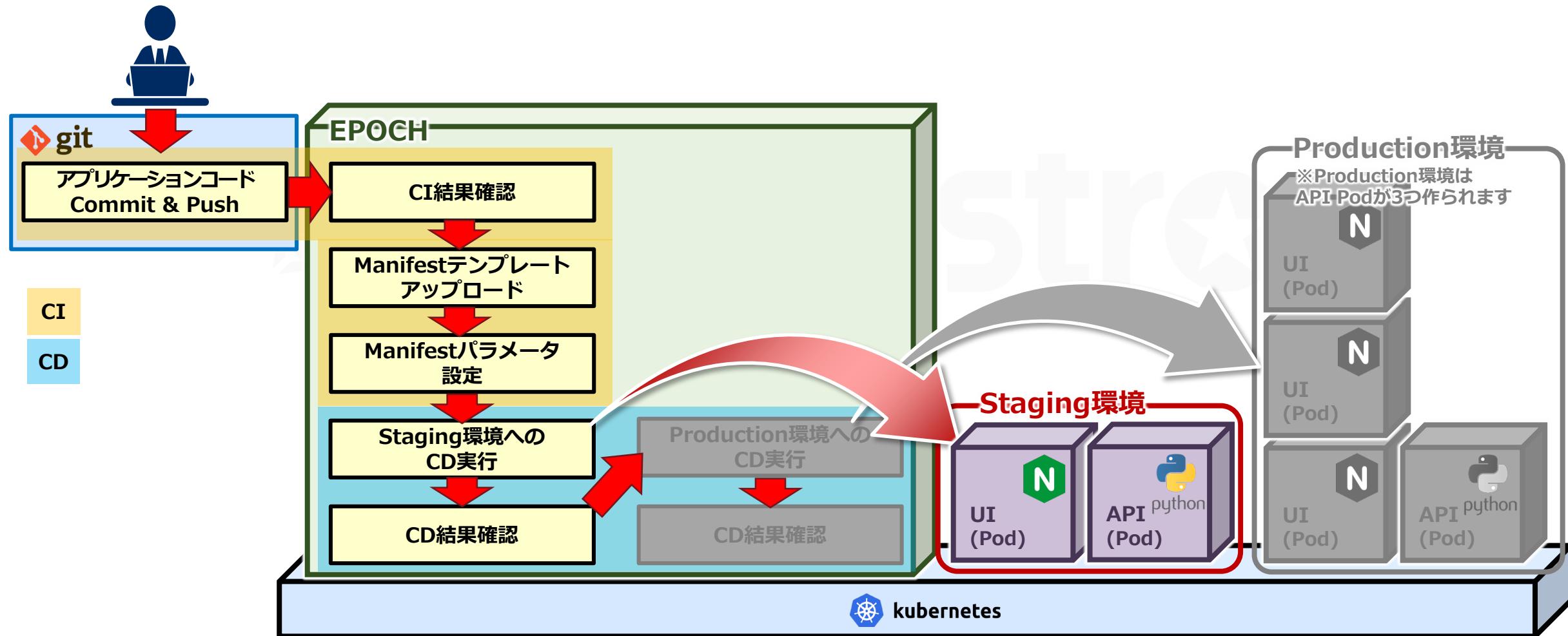
# 1回目のCI/CDワークフロー手順



## 4.5 1回目のCI/CDワークフロー手順(1/24)

### Staging環境へのDeploy

1回目のCI/CDワークフロー手順として、アプリケーションコードのCommit & PushからStaging環境へのDeploy、CD結果確認までの手順は以下の通りとなります。



## 4.5 1回目のCI/CDワークフロー手順(2/24)

### ■ アプリケーションコード Commit & Push

初回デプロイするコンテナイメージを作るためCIパイプラインを実行します。

- PC環境にcloneしたアプリケーションコード用リポジトリでCommit & Pushします

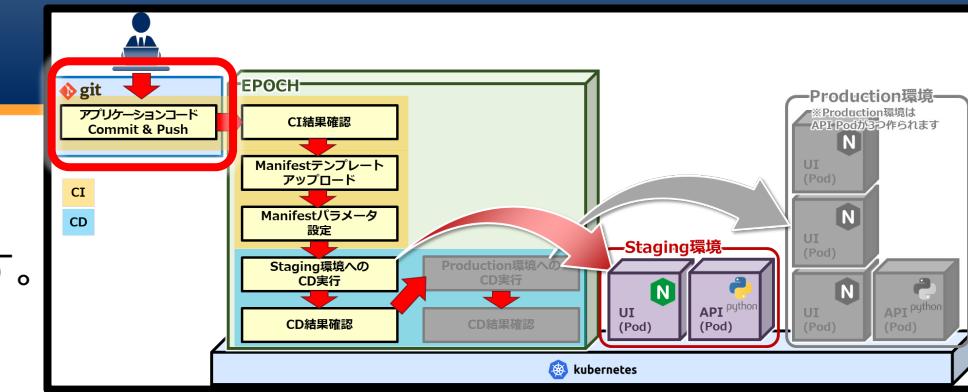
例としてコマンドプロンプトでは、以下の通りとなります。

```
> cd "[clone先のフォルダ]"
> cd epoch-sample-app
> echo "first build" > dummy.txt
> git add .
> git commit -m "first build"
> git push origin master
```

```
Username for 'https://github.com': [自身のGitHubユーザ名を入力]
Password for 'https://xxxxxx@github.com': [自身のGitHubのトークンを入力]
```

Pushが完了すると、パイプラインTEKTONで設定されたCIパイプラインが自動的に動き出します。

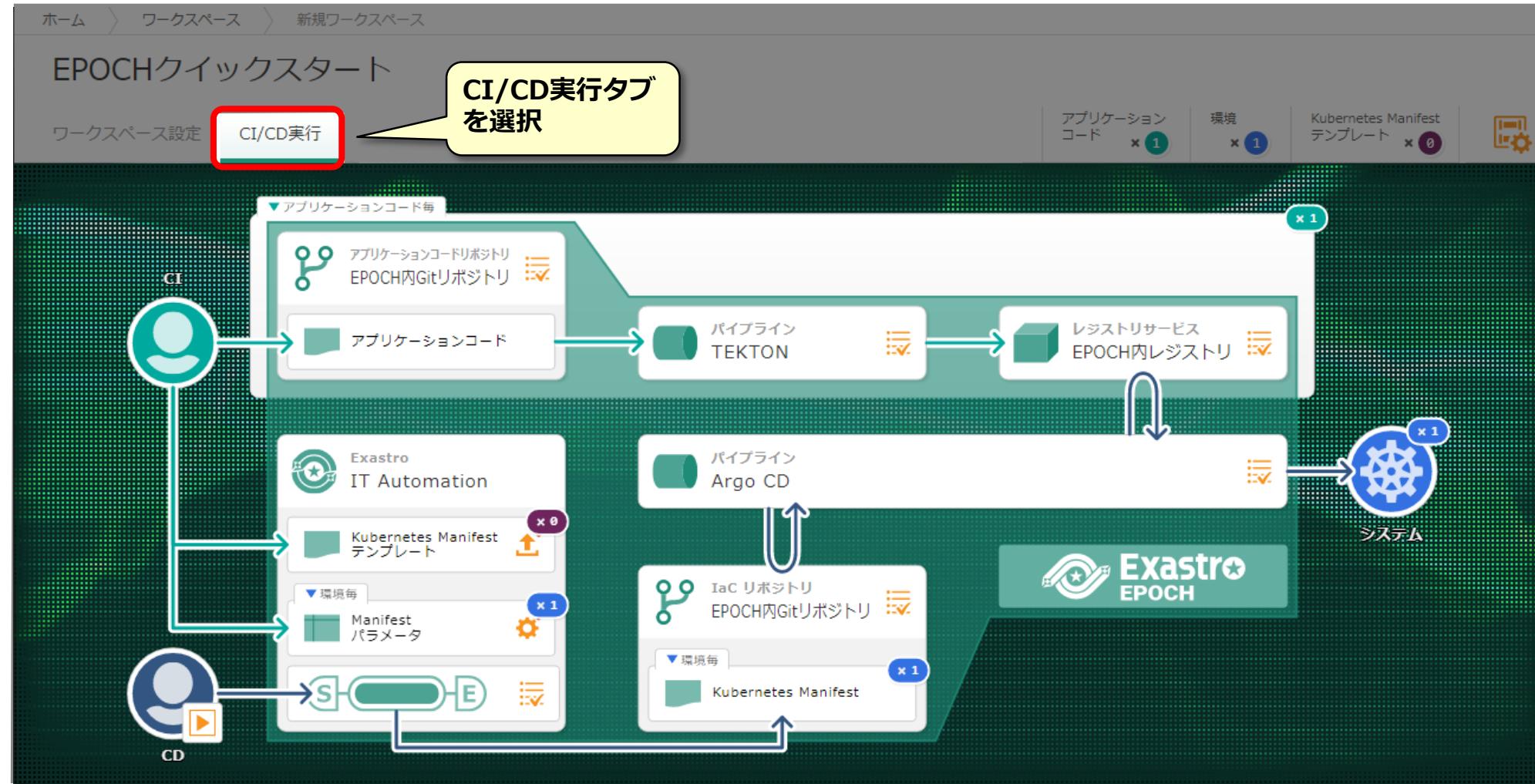
CIパイプラインの結果を確認していきましょう。



## 4.5 1回目のCI/CDワークフロー手順(3/24)

### CI/CD実行画面の表示

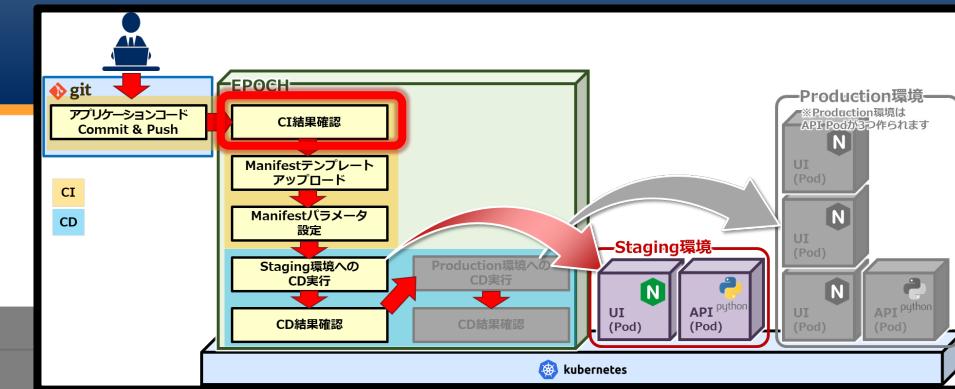
ワークスペース画面のCI/CD実行タブを選択し、CI/CD実行画面を表示します。



## 4.5 1回目のCI/CDワークフロー手順(4/24)

### CI結果確認

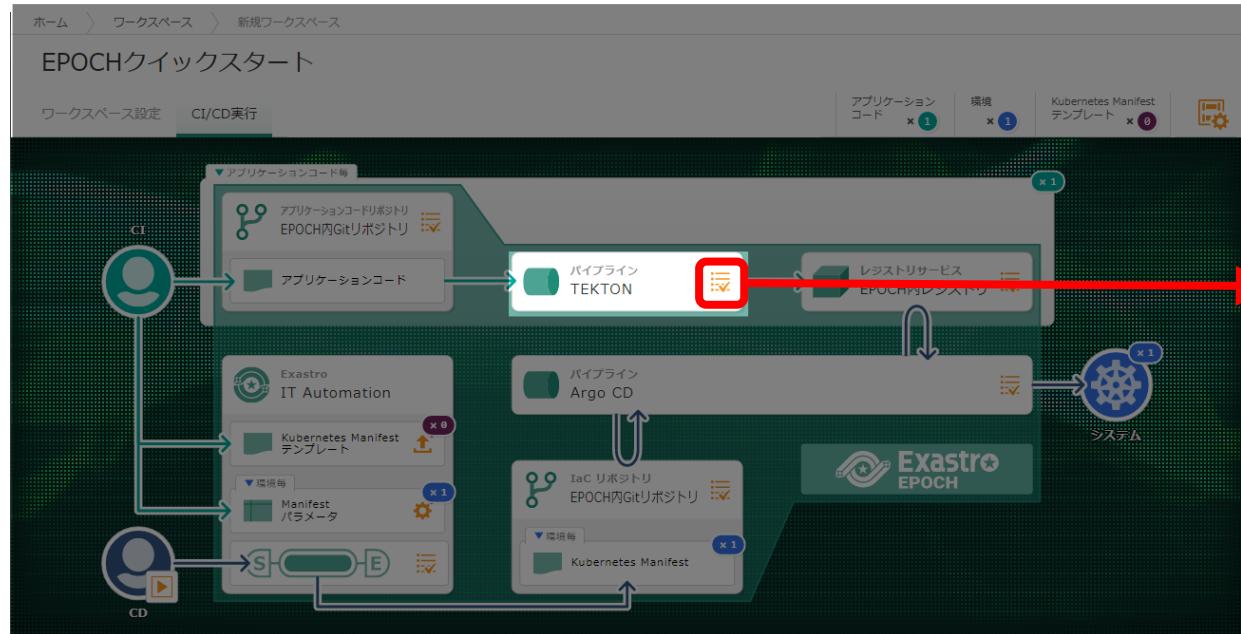
- アプリケーションコードのビルド結果を確認します。



# 4.5 1回目のCI/CDワークフロー手順(5/24)

## ■ パイプラインTEKTONの結果確認

- TEKTONのパイプラインを実際に確認し、ビルドが正常に終了したか確認します。



The Tekton Dashboard shows the 'PipelineRuns' table. A single entry is listed:

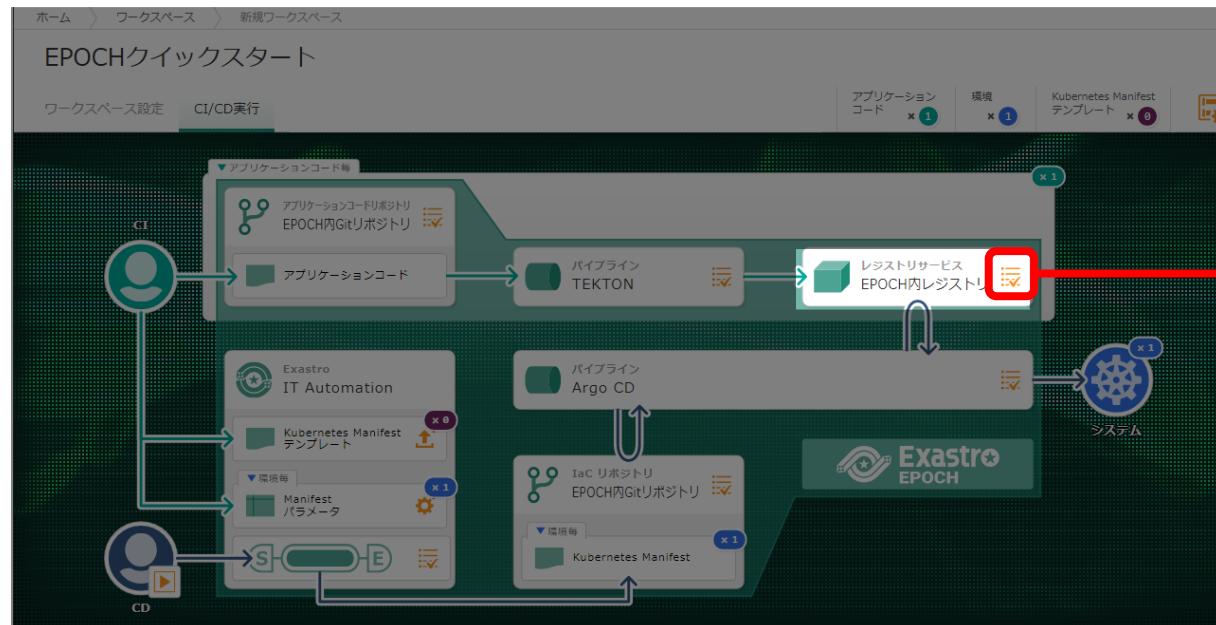
Status	Name	Pipeline	Namespace	Created	Duration
	build-and-push-pipeline-r...	pipeline-build-and-push	epoch-tekon-pipelines	6 minutes ago	3 minutes 32 seconds

A callout box points to the 'Status' column with the text: "Status欄にチェックマークが表示されていれば正常終了となります。". Another callout box on the right side of the dashboard states: "CIパイプラインの動作は、TEKTONダッシュボードにて確認することができます。Nameをクリックすることにより、より詳細なCIパイプラインの内容を確認することができます。".

# 4.5 1回目のCI/CDワークフロー手順(6/24)

## コンテナイメージのタグ名の確認

- レジストリサービスの画面を開き、ビルトしたコンテナイメージのTagを確認します。



The screenshot shows the Dockerhub page for the repository `/epoch-sample-api`. It displays two tags:

- TAG** master.20210701-171058
- TAG** master.20210701-134114

A yellow callout box contains the text:

Dockerhubの画面でepoch-sample-apiのTagを確認します  
※イメージが登録されていないときは、パイプラインTEKTONから結果を確認してください

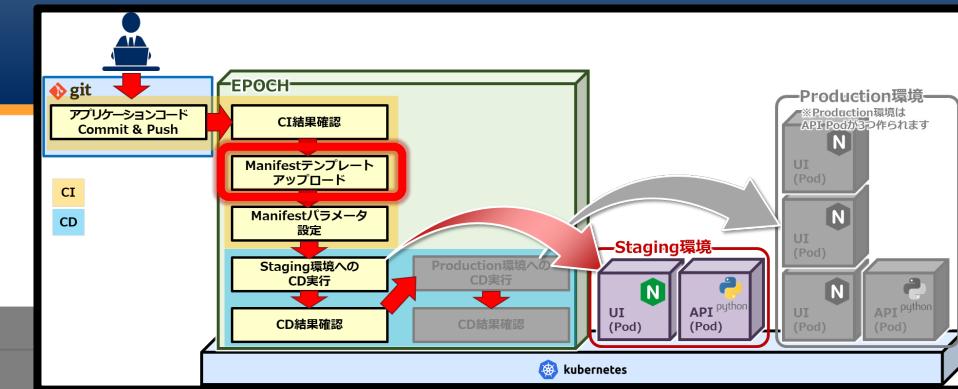
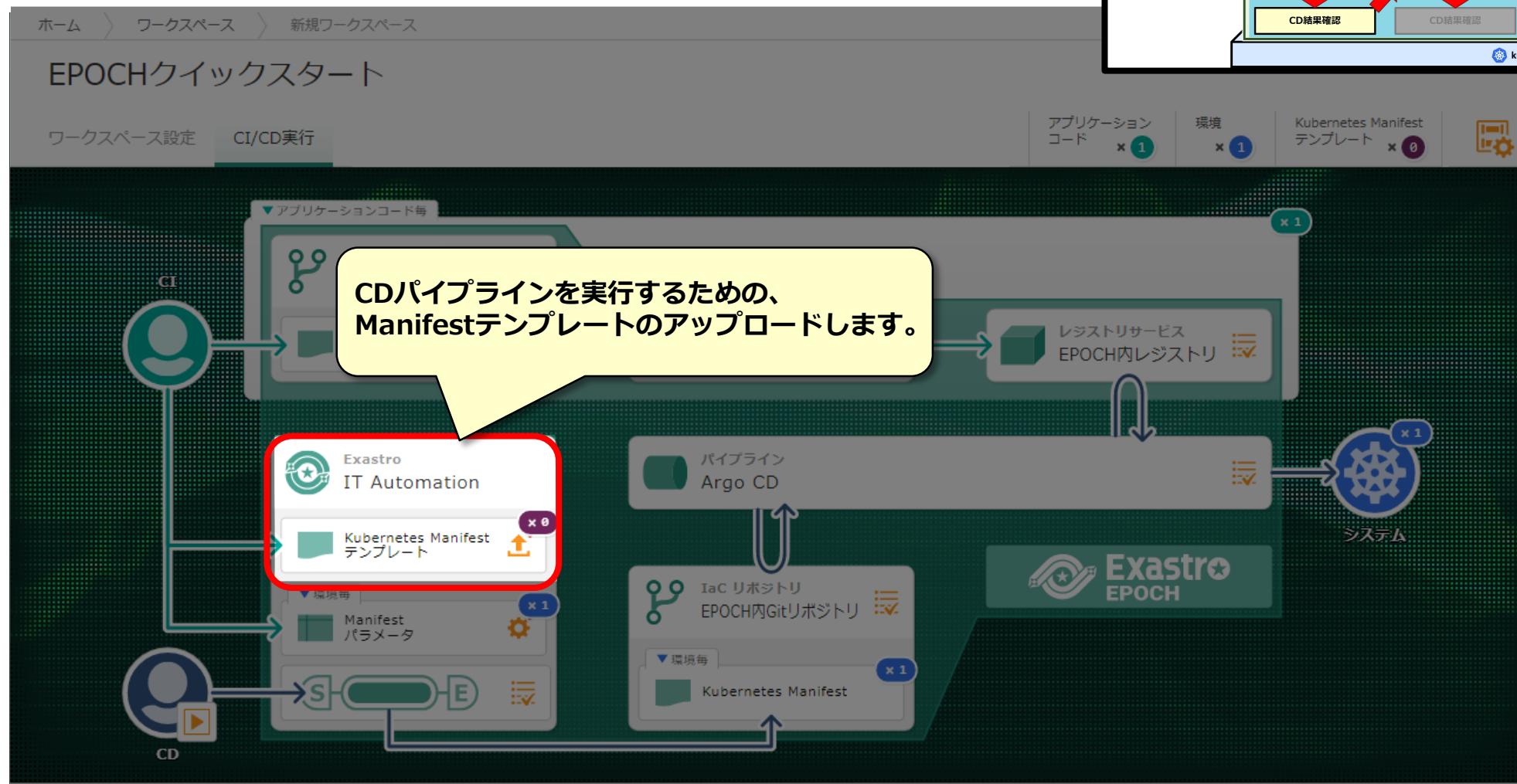
ここで確認したイメージのTagは、  
次の手順で、Manifestパラメータ(image\_tag)に  
手入力が必要になるため控えておいてください

※今後、パイプラインで生成されたimage\_tagは選択できるように変更する予定です

## 4.5 1回目のCI/CDワークフロー手順(7/24)

### Manifestテンプレートアップロード

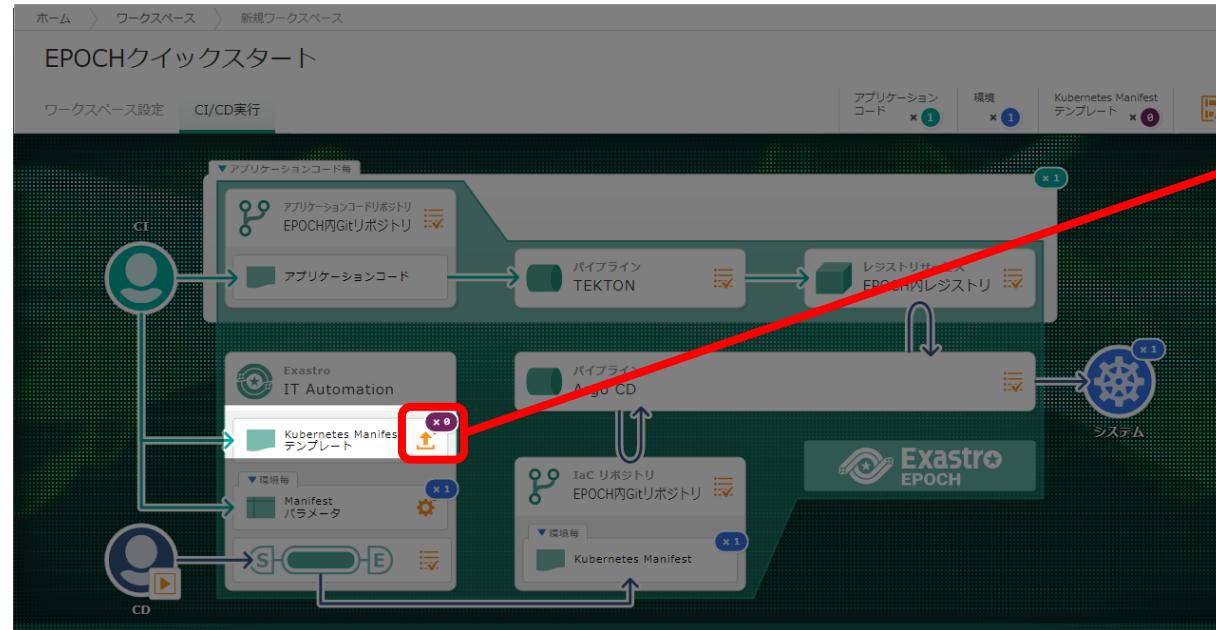
- ダウンロードしたManifestテンプレートをアップロードします。



# 4.5 1回目のCI/CDワークフロー手順(8/24)

## Manifestテンプレートアップロード

- Manifestテンプレートファイルの準備でダウンロードしたManifestテンプレートファイルをアップロードします。



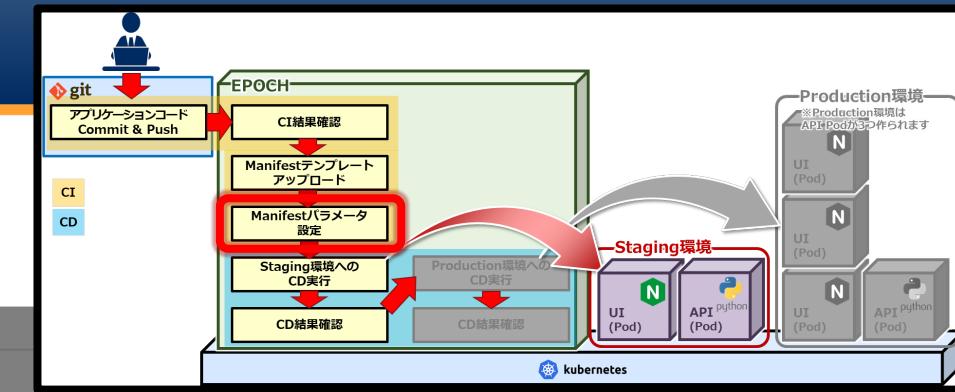
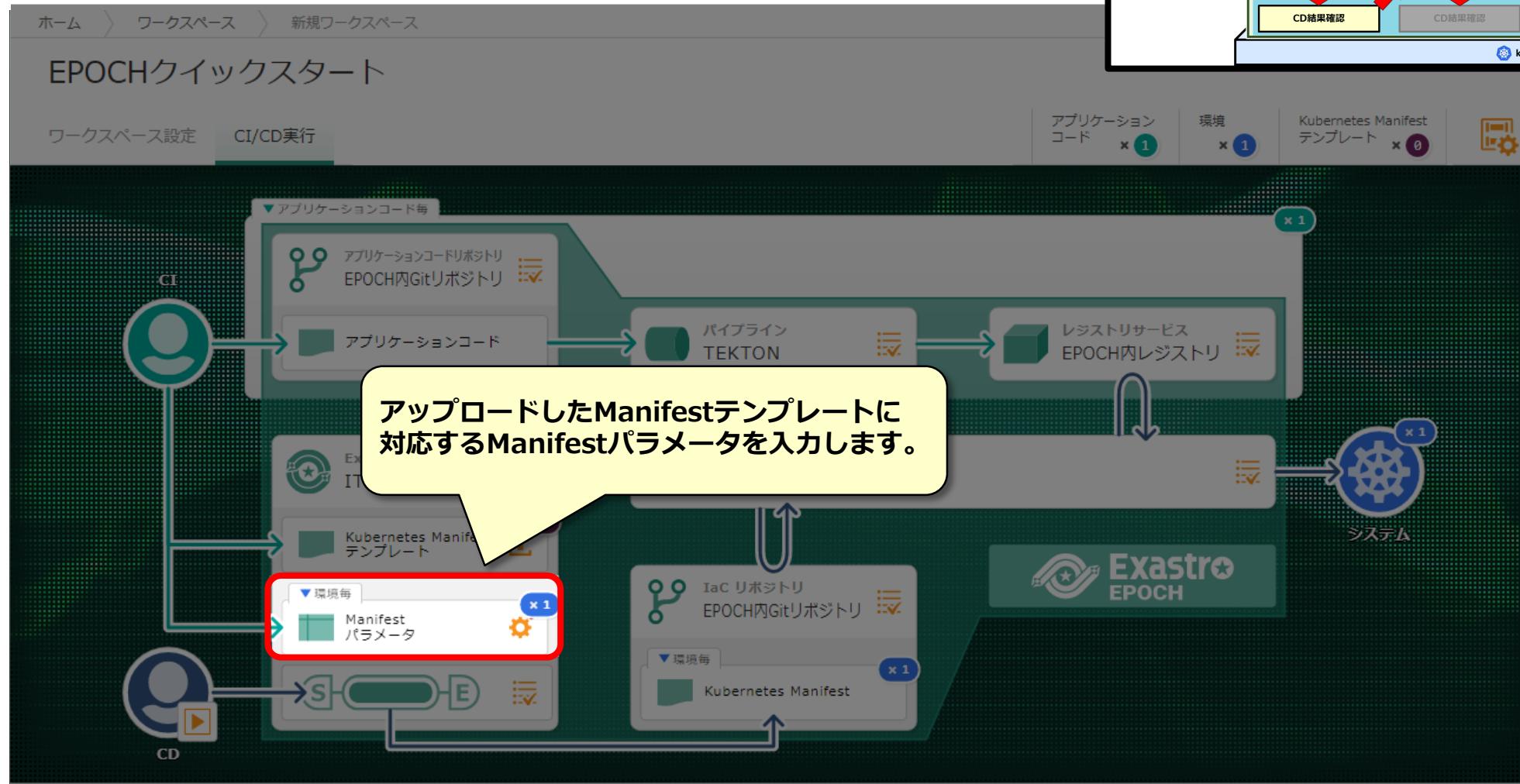
ファイル : api-app.yaml, ui-app.yamlを  
右の画面領域にドロップします

The screenshot shows two overlapping dialog boxes. The top one is titled 'Kubernetes Manifest テンプレート' and has a red box around its 'アップロードファイル選択' button. The bottom one is titled 'Kubernetes Manifest テンプレートアップロード' and has a large red box around its file upload area labeled 'ここにファイルをドロップ または クリック'. Both dialogs include 'アップロード', '再選択', and 'キャンセル' buttons at the bottom.

# 4.5 1回目のCI/CDワークフロー手順(9/24)

## Manifestパラメータ

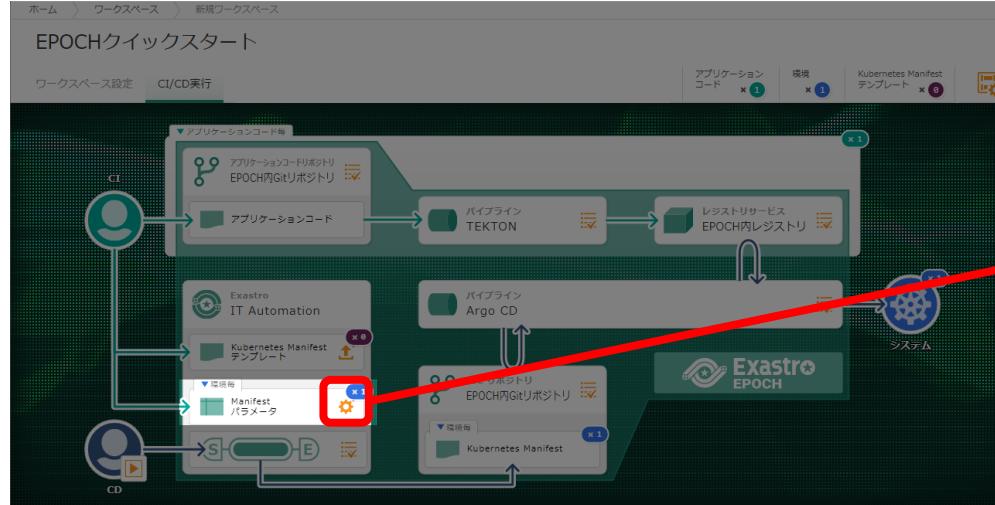
- Deployに必要なManifestパラメータを入力します。



# 4.5 1回目のCI/CDワークフロー手順(10/24)

## Manifestパラメータ入力

- 入力内容に従って、Manifestパラメータを入力します。



ui-app.yamlを選択して入力内容を入力します

入力内容に従って値を入力します  
※環境ごとに異なりますので間違えないように入力してください

Manifest パラメータ

ui-app.yaml

パラメータ	staging	production
param01	staging : param01	production : param01
image	staging : image	production : image
image_tag	staging : image_tag	production : image_tag
param02	staging : param02	production : param02
param03	staging : param03	production : param03

```
replicas: {{ param01 }}  
template:  
  metadata:  
    labels:  
      name: ui-app  
  spec:  
    containers:  
      - name: ui-app  
        image: {{ [image] }}:{{ [image_tag] }}  
        ports:  
          - name: http  
            containerPort: 8000
```

決定 キャンセル

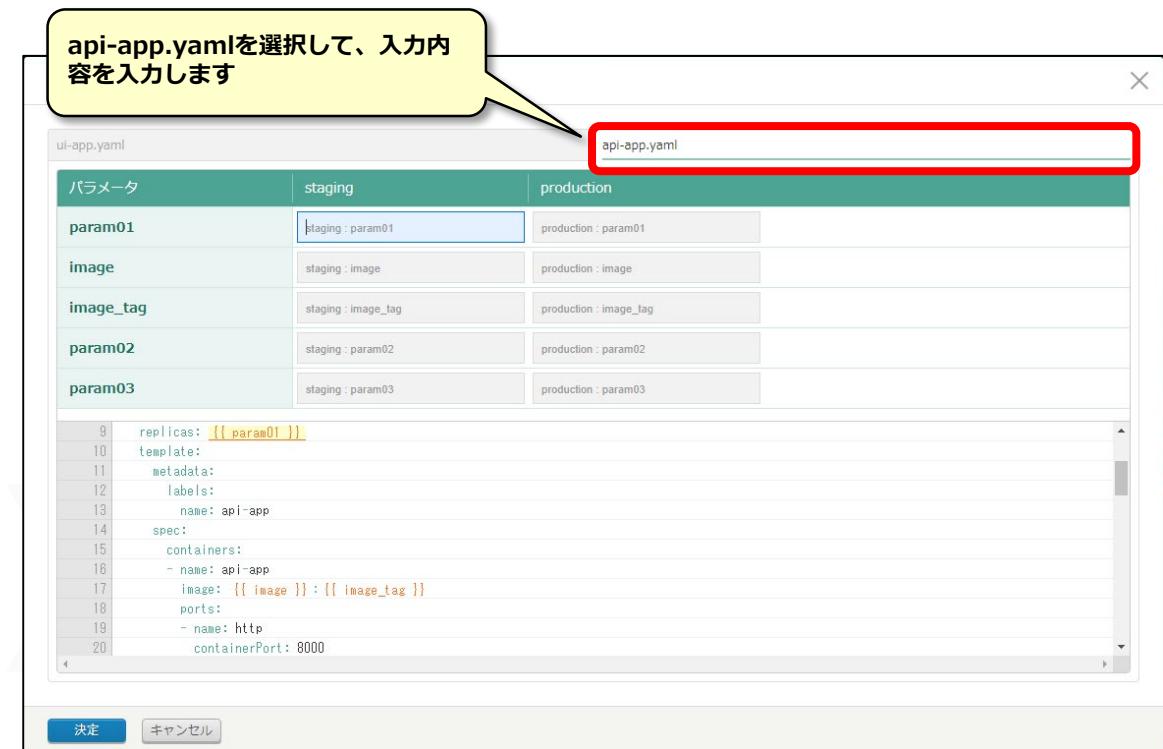
ui-app.yaml

項目	入力内容(staging)	入力内容(production)	説明
{{ param01 }}	1	3	レプリカ数
{{ image }}	exastro/epochsampleappui	exastro/epochsampleappui	コンテナイメージ
{{ image_tag }}	master.20210708183910	master.20210708183910	コンテナイメージのタグ
{{ param02 }}	31001	31003	ブルーグリーンデプロイ用のブルー面のポート番号
{{ param03 }}	32001	32003	ブルーグリーンデプロイ用のグリーン面のポート番号

※今後、image、image\_tagの入力については選択項目に変更する予定です

## 4.5 1回目のCI/CDワークフロー手順(11/24)

- タブを切り替えてapi-app.yamlにも入力します。



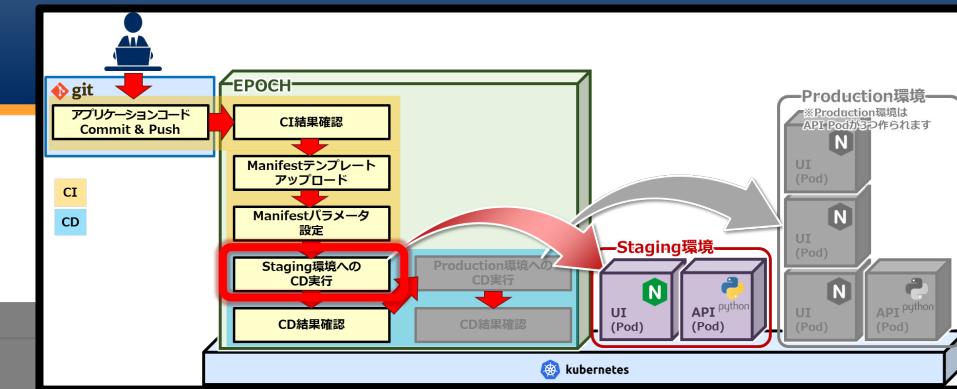
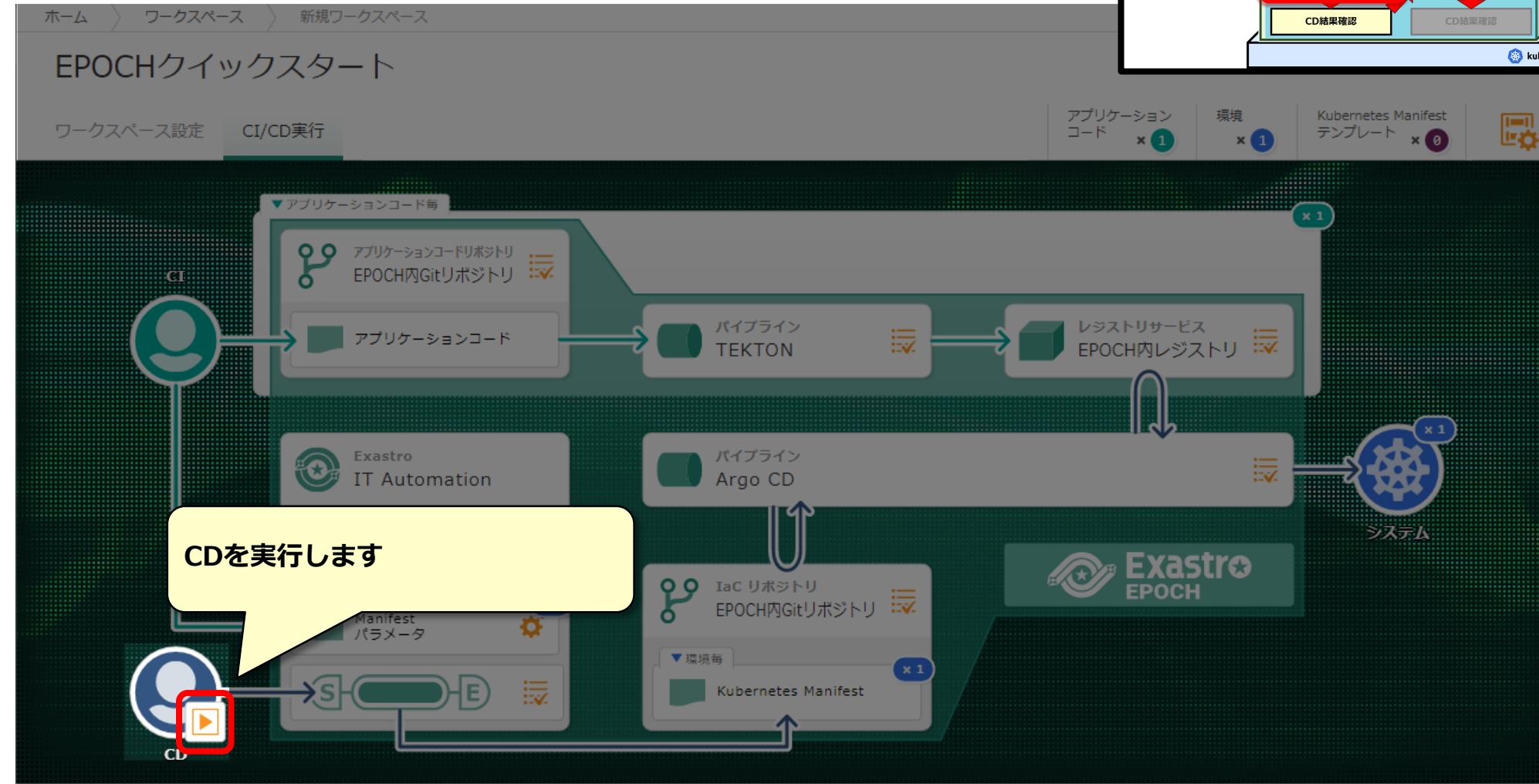
api-app.yaml

項目	入力内容(staging)	入力内容(production)	説明
{{ param01 }}	1	1	レプリカ数
{{ image }}	[Dockerhubのアカウント名]/epoch-sample-api	[Dockerhubのアカウント名]/epoch-sample-api	コンテナイメージ
{{ image_tag }}	[レジストリサービスで確認したimageのタグ名]	[レジストリサービスで確認したimageのタグ名]	コンテナイメージのタグ
{{ param02 }}	31002	31004	ブルーグリーンデプロイ用のブルー面のポート番号
{{ param03 }}	32002	32004	ブルーグリーンデプロイ用のグリーン面のポート番号

## 4.5 1回目のCI/CDワークフロー手順(12/24)

### Staging環境へのDeploy実行

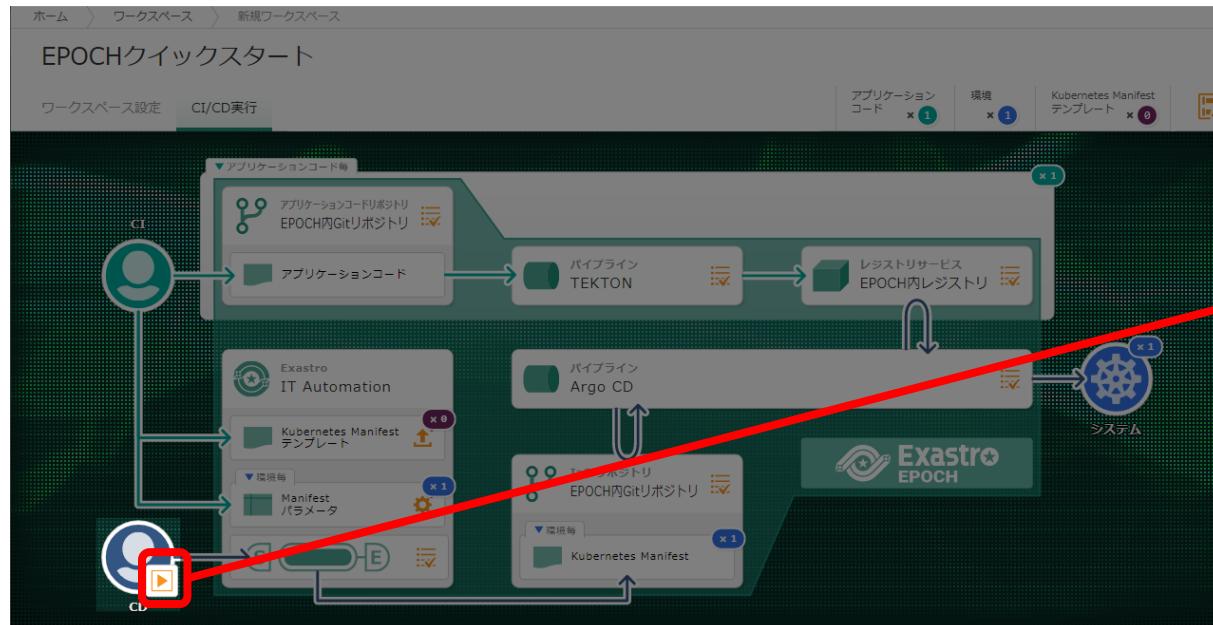
- CD実行で、Staging環境へDeployを実行します。



# 4.5 1回目のCI/CDワークフロー手順(13/24)

## Staging環境のCD実行

- Deploy先の環境を選択して実行します。

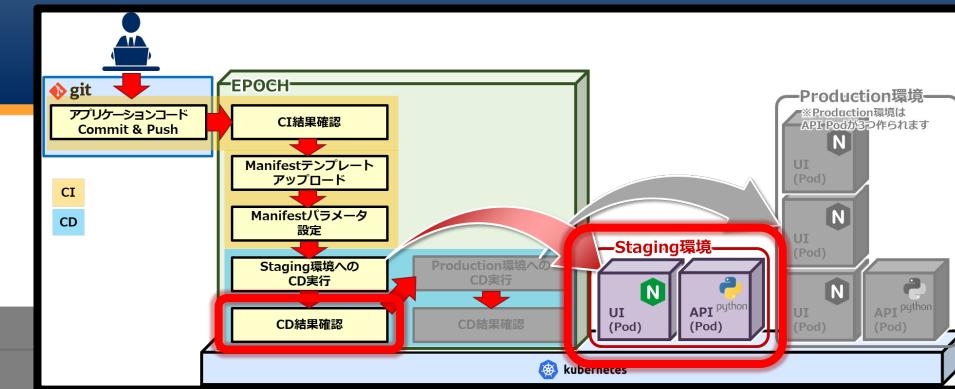


Staging環境へのCD実行が完了しました  
CD実行結果を確認してみましょう

## 4.5 1回目のCI/CDワークフロー手順(14/24)

### Staging環境のCD結果確認

- CDの実行結果は、Exastro IT-Automation、ArgoCDより確認できます。



## 4.5 1回目のCI/CDワークフロー手順(15/24)

### Manifestファイルの生成確認(Staging環境)

- Exastro IT-Automationから、IaCリポジトリへManifestファイルを登録するまでの状況を確認します。

The screenshot shows the Exastro IT Automation interface. On the left, the 'EPOCHクイックスタート' dashboard displays various CI/CD components like Application Code, Pipeline TEKTON, Pipeline Argo CD, and Kubernetes Manifest. A red box highlights the 'Conductor' icon in the bottom right corner of the dashboard area. A yellow callout box [1] points to this icon with the text '[1] Conductorメニュー > Conductor作業一覧を選択します'. In the center, a 'Conductor' task list window is open. It shows a table with columns: 延止 (Stop), Conductor-インスタンスID (Conductor instance ID), Conductor名称 (Conductor name), オペレーター (Operator), and 一覧 (List). A red box highlights the 'フィルタ' (Filter) button. A yellow callout box [2] points to it with the text '[2] フィルタをクリック'. At the bottom of the table, a red box highlights the '詳細' (Details) button for a specific task. A yellow callout box [3] points to it with the text '[3] 【CD実行】の詳細をクリック'. On the right, a 'CHECKING' status window shows a workflow with several green 'DONE' status boxes, indicating successful execution. A yellow callout box [4] points to this window with the text 'すべて[DONE]と表示されていれば完了です'.

ログインID: epoch-user  
パスワード: c2jthascR93ijdcyzwJY  
でログインします

[1] Conductorメニュー > Conductor作業一覧を選択します

[2] フィルタをクリック

[3] 【CD実行】の詳細をクリック

すべて[DONE]と表示されていれば完了です

## 4.5 1回目のCI/CDワークフロー手順(16/24)

### パイプラインArgoCDの結果確認(Staging環境)

- Manifestがkubernetesに反映されるまでの状況を確認します。

The diagram illustrates the CI/CD pipeline and the Argo CD sync status for the staging environment.

**EPOCH CI/CD Pipeline:** Shows the flow from Application Code to Application Manifest, through TKETON and Argo CD, to the Kubernetes Registry and System.

**Argo CD Sync Status:** Shows the sync status for the **staging** environment. The **staging** project is highlighted with a red box and a callout "stagingを選択". The sync status indicates it is **Missing** and **Syncing**. A red arrow points from the Argo CD interface to the sync status in the EPOCH interface.

**Login Information:** A callout shows the login credentials: **Username: admin** and **Password: iYSCKzx2wvxJnn4dCNwN**. It also notes that "SIGN INします" (Sign In) is selected.

**Sync Status Monitoring:** A callout states "3分ごとに同期されますので同期されることを待ちます" (Sync occurs every 3 minutes, so wait for synchronization). A red arrow points from the sync status in the EPOCH interface to this callout.

**Deployment Confirmation:** A large red box at the bottom contains the text "Deployされたサンプルアプリケーションを確認してみましょう" (Let's check the deployed sample application).

## 4.5 1回目のCI/CDワークフロー手順(17/24)

### Staging環境のサンプルアプリケーション確認

- 次のURLでデプロイしたサンプルアプリケーションを表示します

[http://\[Kubernetes masterノードのIPアドレスまたはホスト名\]:31001/front-end.html](http://[Kubernetes masterノードのIPアドレスまたはホスト名]:31001/front-end.html)

商品カタログ

表示価格 YEN YEN USD

TシャツA TシャツB TシャツC TシャツD

1,000 円(税込) 2,000 円(税込) 7,500 円(税込) 4,700 円(税込)

詳細はこちら 詳細はこちら 詳細はこちら 詳細はこちら

購入サイトはこちら 購入サイトはこちら 購入サイトはこちら 購入サイトはこちら

TシャツE TシャツF TシャツG

1,200 円(税込) 3,600 円(税込) 6,500 円(税込) 4,500 円(税込)

詳細はこちら 詳細はこちら 詳細はこちら 詳細はこちら

表示価格の選択欄に「YEN」「USD」が表示されていることを確認します。  
確認後、「USD」を選択して、表示が切り替わることを確認します。

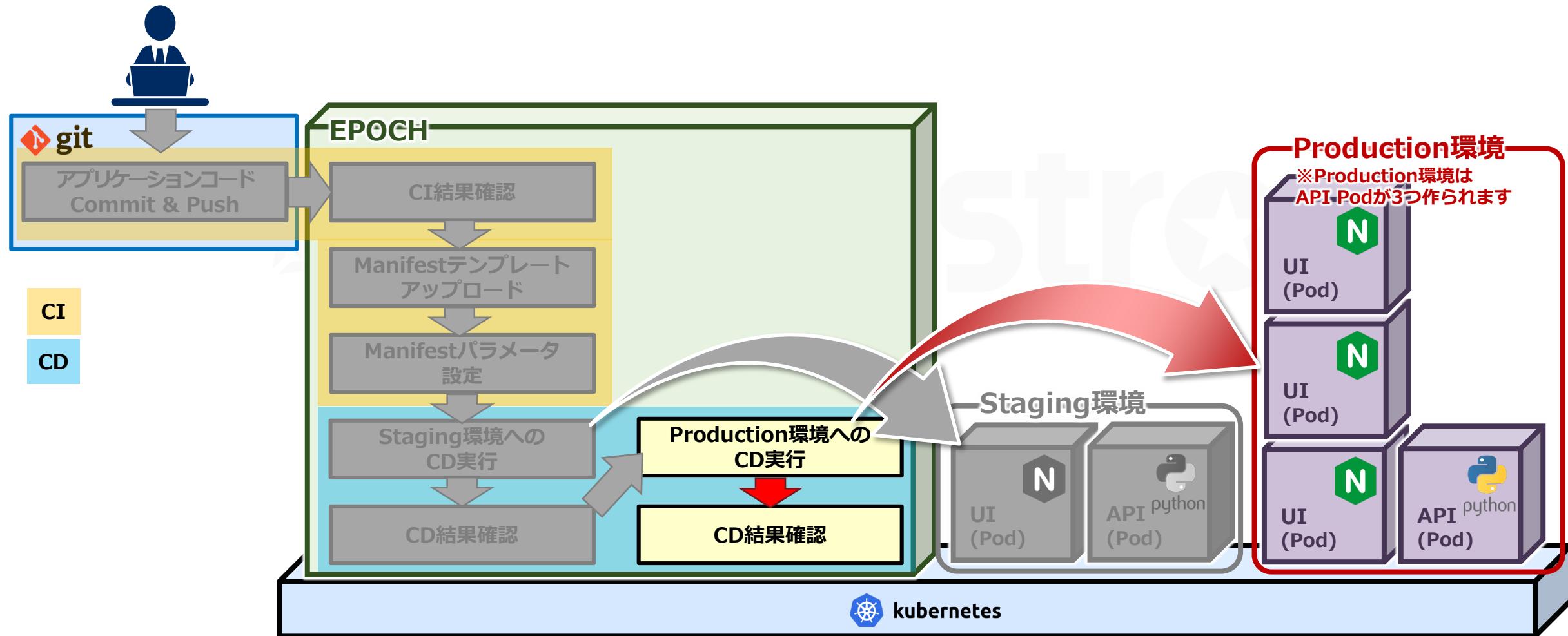
Exastro

続いてProduction環境へ  
Deployしてみましょう

## 4.5 1回目のCI/CDワークフロー手順(18/24)

### Production環境へのDeploy

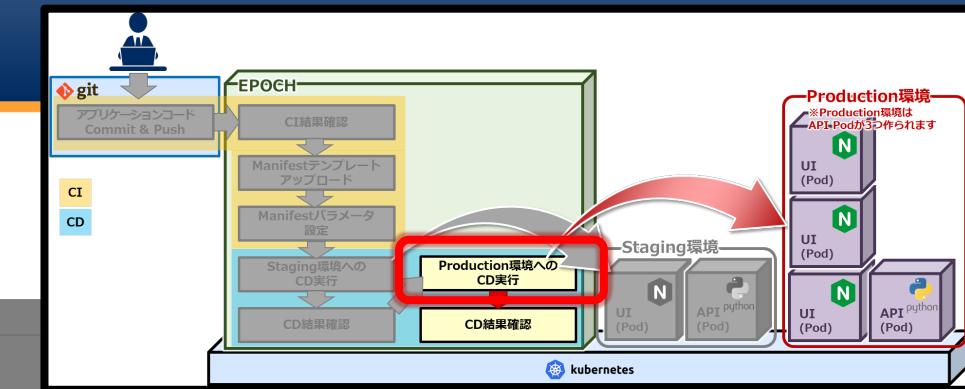
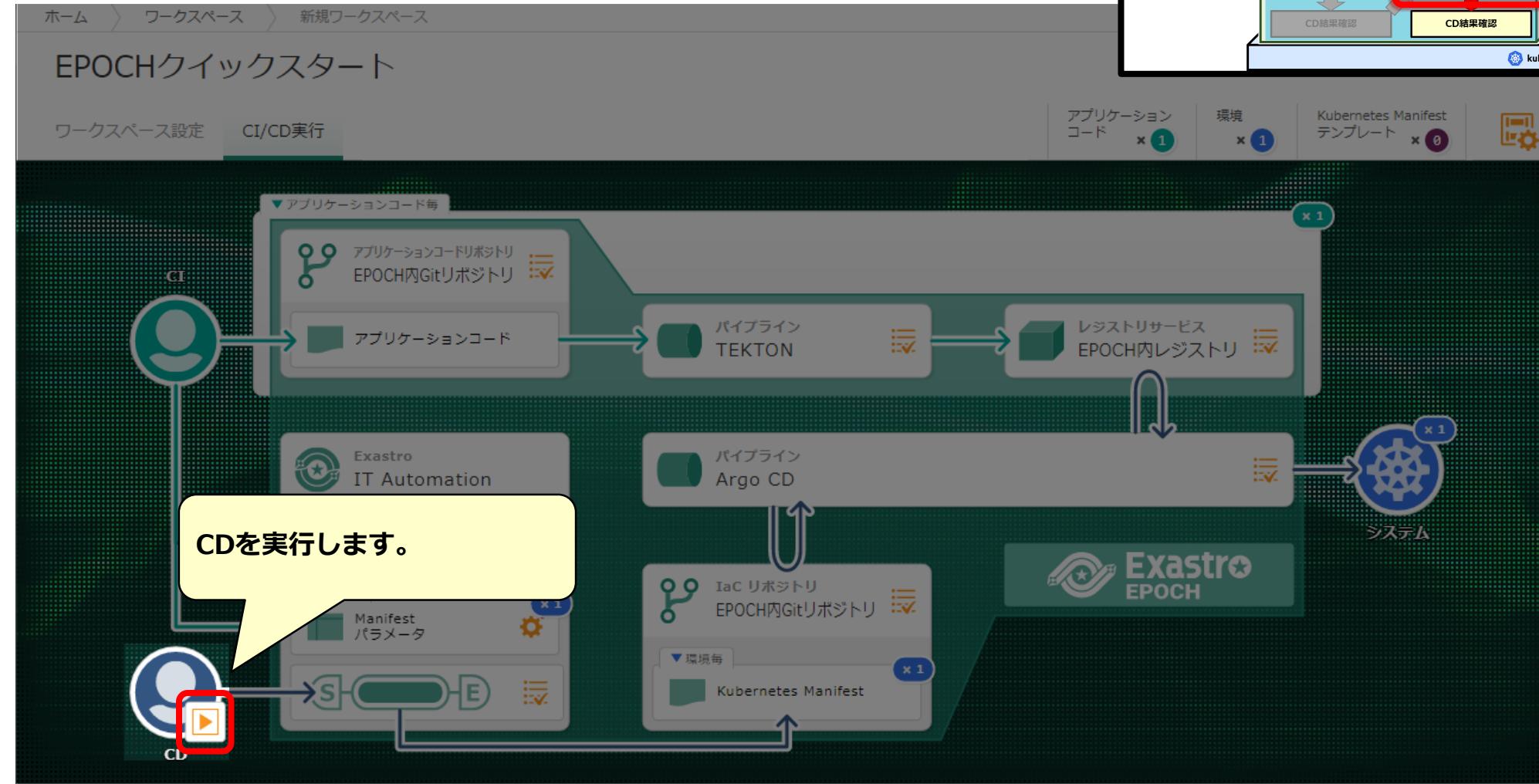
Staging環境へDeploy後、以下の内容でProduction環境へDeployします。



## 4.5 1回目のCI/CDワークフロー手順(19/24)

### Production環境へのDeploy実行

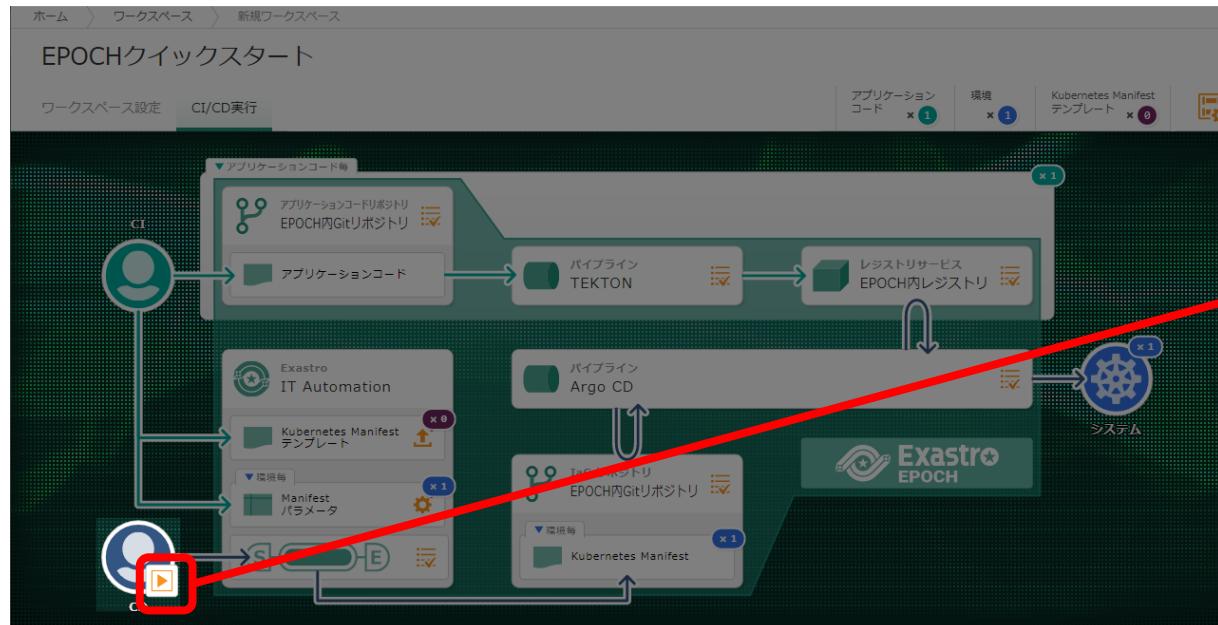
- CD実行で、Production環境へDeployを実行します。



## 4.5 1回目のCI/CDワークフロー手順(20/24)

### Production環境のCD実行

- Deploy先の環境を選択して実行します。



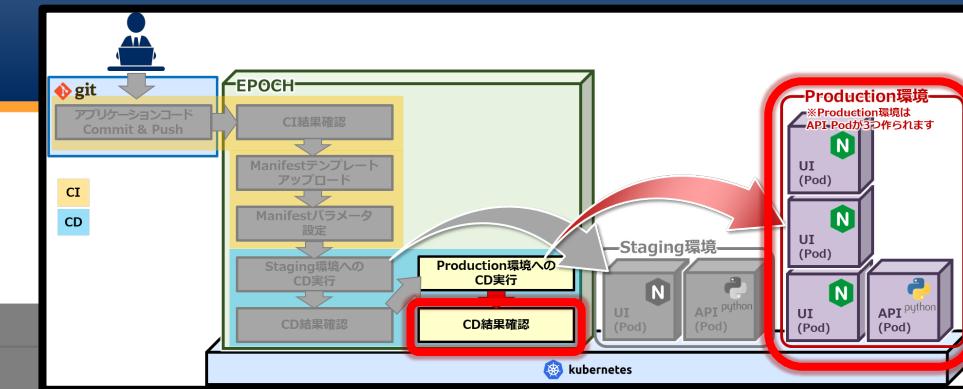
The screenshot shows the 'CD Execution Specification' dialog box. It has sections for 'Execution Conditions' (选择条件), 'Manifest Parameters' (Manifestパラメータ), and 'ArgoCD Pipeline' (ArgoCDパイプライン). A red box highlights the 'Environment' dropdown where 'production' is selected. A callout bubble says 'productionを選択します' (Select production). Another callout bubble at the bottom right says '環境選択後、実行ボタンを押下します' (After selecting the environment, press the execute button).

Production環境へのCD実行が完了しました  
CD実行結果を確認してみましょう

## 4.5 1回目のCI/CDワークフロー手順(21/24)

### Production環境のCD結果確認

- CDの実行結果は、Exastro IT-Automation、ArgoCDより確認できます。



# 4.5 1回目のCI/CDワークフロー手順(22/24)

## Manifestファイルの生成確認(Production環境)

- Exastro IT-Automationから、IaCリポジトリへManifestファイルを登録するまでの状況を確認します。

The screenshot illustrates the workflow for generating and confirming Manifest files in the Production environment. It shows the Exastro IT Automation interface with various components like EPOCH, TEKTON, Argo CD, and Conductor.

**Step 1:** [1] Conductorメニュー > Conductor作業一覧を選択します

**Step 2:** [2] フィルタをクリック

**Step 3:** [3] 【CD実行】の詳細をクリック

**Step 4:** ログインID: epoch-user  
パスワード: c2jthascR93ijdcyzwJY  
でログインします

**Step 5:** すべて[DONE]と表示されていれば完了です

# 4.5 1回目のCI/CDワークフロー手順(23/24)

## パイプラインArgoCDの結果確認(Production環境)

- Manifestがkubernetesに反映されるまでの状況を確認します。

The diagram illustrates the CI/CD workflow and deployment status across three main stages:

- CI (Left):** Shows the flow from Application Code to Pipeline TEKTON, which then interacts with the Registry Service EPOCH内レジストリ and the Argo CD Pipeline.
- CD (Bottom Left):** Shows the Applications screen where the "production" environment is selected. It displays two entries:
  - production:** Project: default, Status: Healthy Synced, Repository: https://github.com/shiota-2021/epoch..., Target R...: /, Destination: in-cluster, Namespace: epoch-sample-app-production. Buttons: SYNC, REFRESH, DELETE.
  - staging:** Project: default, Status: Missing OutOfSync Syncing, Repository: https://github.com/shiota-2021/epoch..., Path: /, Destination: in-cluster, Namespace: epoch-sample-app-staging. Buttons: SYNC, REFRESH, DELETE.
- Production Environment (Top Right):** Shows the Argo CD login screen with a yellow callout: "Username: admin, Password: iYSCKzx2wvxJnn4dCNwN でSIGN INします". A note below says: "※「この接続ではプライバシーが保護されません」が表示されますが、詳細表示から「アクセスする」を選択してログイン画面に遷移します".
- Deployment Status (Bottom Right):** Shows the Applications / staging screen with a yellow callout: "3分ごとに同期されますので同期されることを待ちます". It shows the application status as "Sync OK" with a timestamp: "Sync OK To b6b7ae Succeeded a minute ago (17:18:26 GMT+0900) Authored by epoch-team <epoch-team@epoch-nec.com> Manifest-Push\_202107081712". Below it, a detailed view of the deployed services shows "ui-app-com" and "api-app-bluegreen" services with their respective pods and endpoints.
- Final Callout (Bottom Center):** A large red box contains the text: "Deployされたサンプルアプリケーションを確認してみましょう".

## 4.5 1回目のCI/CDワークフロー手順(24/24)

### Production環境のサンプルアプリケーション確認

- 次のURLでデプロイしたサンプルアプリケーションを表示します

[http://\[Kubernetes masterノードのIPアドレスまたはホスト名\]:31003/front-end.html](http://[Kubernetes masterノードのIPアドレスまたはホスト名]:31003/front-end.html)

商品カタログ

表示価格 YEN  
YEN  
USD

Tシャツ	価格	操作
TシャツA	1,000 円(税込)	<a href="#">詳細はこちら</a>
TシャツB	2,000 円(税込)	<a href="#">詳細はこちら</a>
TシャツC	7,500 円(税込)	<a href="#">詳細はこちら</a>
TシャツD	4,700 円(税込)	<a href="#">詳細はこちら</a>
TシャツE	1,200 円(税込)	<a href="#">詳細はこちら</a>
	3,600 円(税込)	<a href="#">詳細はこちら</a>
	6,500 円(税込)	<a href="#">詳細はこちら</a>

購入サイトはこちら

Staging環境と同様に画面が表示されることを確認します

**1回目のCI/CDワークフローはここで完了です  
続いて実際にアプリケーションコードを修正し  
Deployされるまでの2回目のCI/CDワークフロー手順  
を実行してみましょう！**

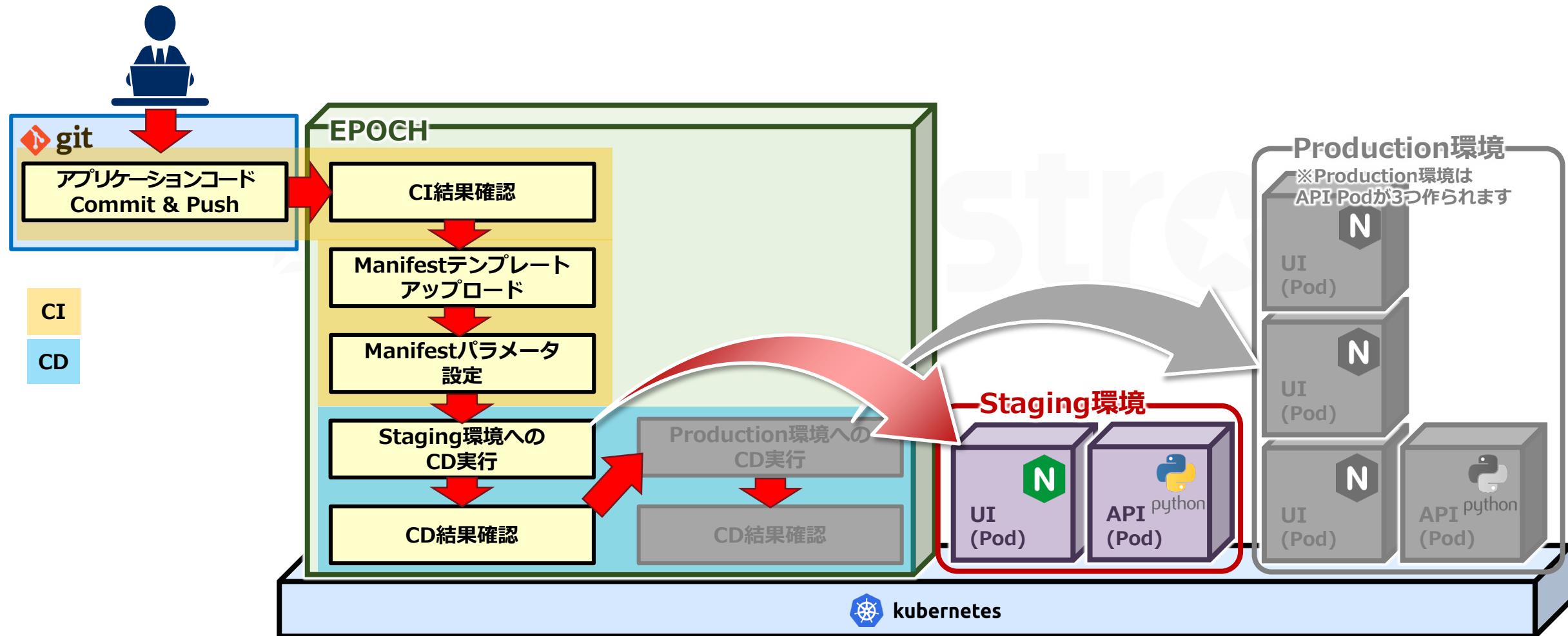
## 2回目のCI/CDワークフロー手順



## 4.6 2回目のCI/CDワークフロー手順(1/21)

### Staging環境へのDeploy

2回目のCI/CDワークフロー手順として、アプリケーションコードを修正してCommit & PushからStaging環境へのDeploy、CD結果確認までの手順を説明します。



## 4.6 2回目のCI/CDワークフロー手順(2/21)

### ■ アプリケーションコードの修正

アプリケーションコードを修正して、2回目のCI/CDを実行していきます。  
チュートリアルでは、画面に通貨（ユーロ）の表示追加を行います。

PC環境にcloneしたアプリケーションコード用リポジトリの、以下のファイルをコードエディタで修正します。

- 修正対象① : api-app/data/currency.json

```
6   "USD": {  
7       "symbol"      :    "$",  
8       "formatter"   :    "{symbol} {price:,.2f} (Tax Included)"  
9   },  
10  "EUR": {  
11      "symbol"      :    "€",  
12      "formatter"   :    "{symbol} {price:,.2f}"  
13  }  
14 }
```

} 9~12行目を追加

- 修正対象② : api-app/data/rate.json

```
1  {  
2      "USD": 110.56,  
3      "EUR": 134.15  
4  }
```

} 2行目の末尾にカンマを追加  
} 3行目を追加

## 4.6 2回目のCI/CDワークフロー手順(3/21)

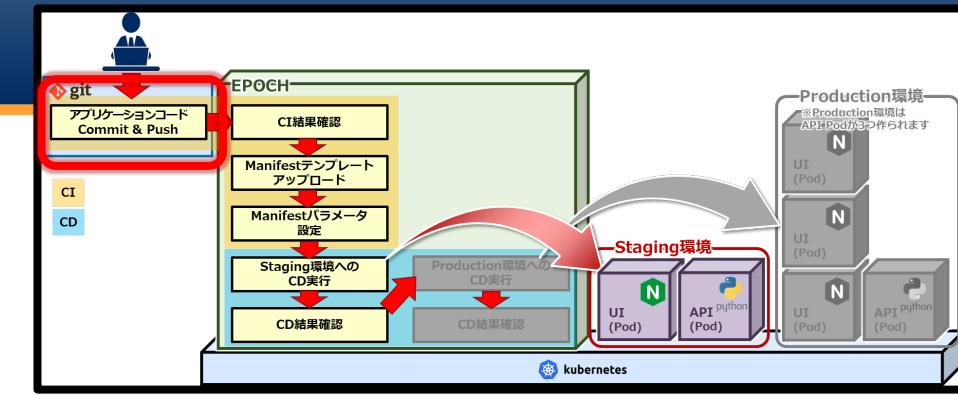
### アプリケーションコード Commit & Push

修正した内容をCommit&Pushします。

コマンドプロンプトで、以下の様に実行します。

```
> cd "[clone先のフォルダ]"
> cd epoch-sample-app
> git add .
> git commit -m "通貨追加(EUR)"
> git push origin master
```

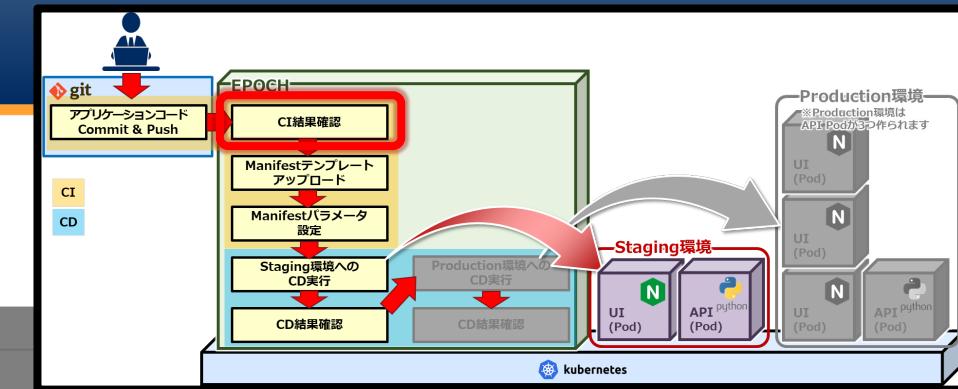
```
Username for 'https://github.com': [自身のGitHubユーザ名を入力]
Password for 'https://xxxxxx@github.com': [自身のGitHubパスワードを入力]
```



# 4.6 2回目のCI/CDワークフロー手順(4/21)

## CI結果確認

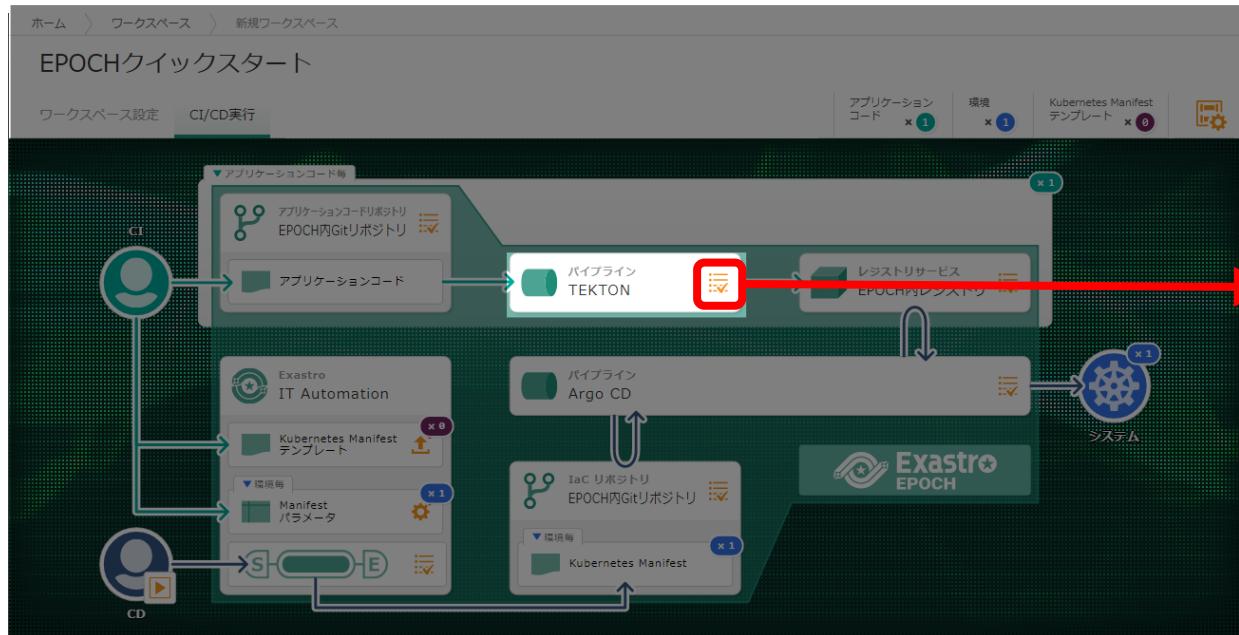
- アプリケーションコードのビルド結果を確認します。



# 4.6 2回目のCI/CDワークフロー手順(5/21)

## ■ パイプラインTEKTONの結果確認

- TEKTONのパイプラインを実際に確認し、ビルドが正常に終了したか確認します。



The Tekton Dashboard shows the 'PipelineRuns' table. A single entry is listed:

Status	Name	Pipeline	Namespace	Created	Duration
	build-and-push-pipeline-r...	pipeline-build-and-push	epoch-tekon-pipelines	6 minutes ago	3 minutes 32 seconds

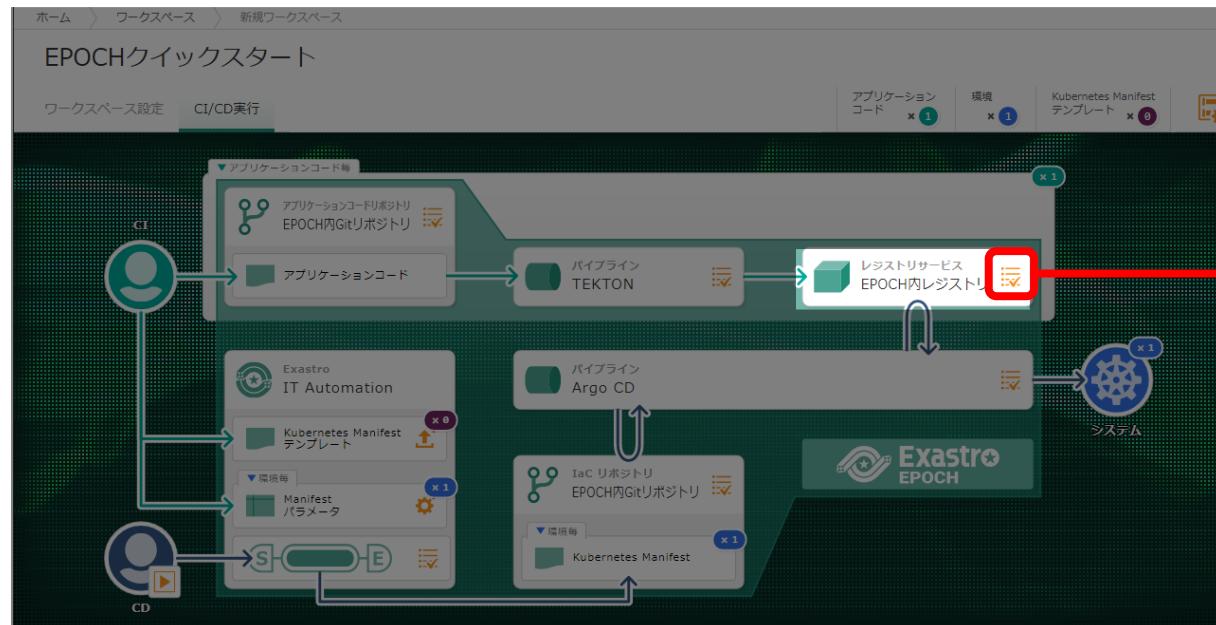
A callout bubble indicates: "CIパイプラインの動作は、TEKTONダッシュボードにて確認することができます。Nameをクリックすることにより、より詳細なCIパイプラインの内容を確認することができます。" (The CI pipeline's operation can be confirmed via the Tekton Dashboard. Clicking on the Name will allow you to view more detailed CI pipeline information.)

Another callout bubble indicates: "Status欄にチェックマークが表示されていれば正常終了となります。" (If a checkmark is displayed in the Status column, it indicates a successful completion.)

# 4.6 2回目のCI/CDワークフロー手順(6/21)

## コンテナイメージのタグ名の確認

- レジストリサービスの画面を開き、ビルトしたコンテナイメージのTagを確認します。



Dockerhubの画面でepoch-sample-apiのTagを確認します  
※イメージが登録されていないときは、パイプラインTEKTONから結果を確認してください

TAG	DIGEST	OS/ARCH	COMPRESSED SIZE
master.20210701-171058		linux/amd64	87.84 MB
master.20210701-134114		linux/amd64	87.84 MB

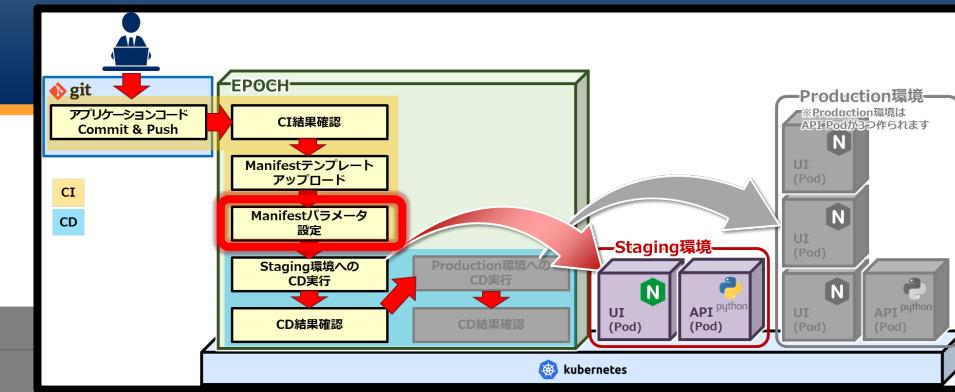
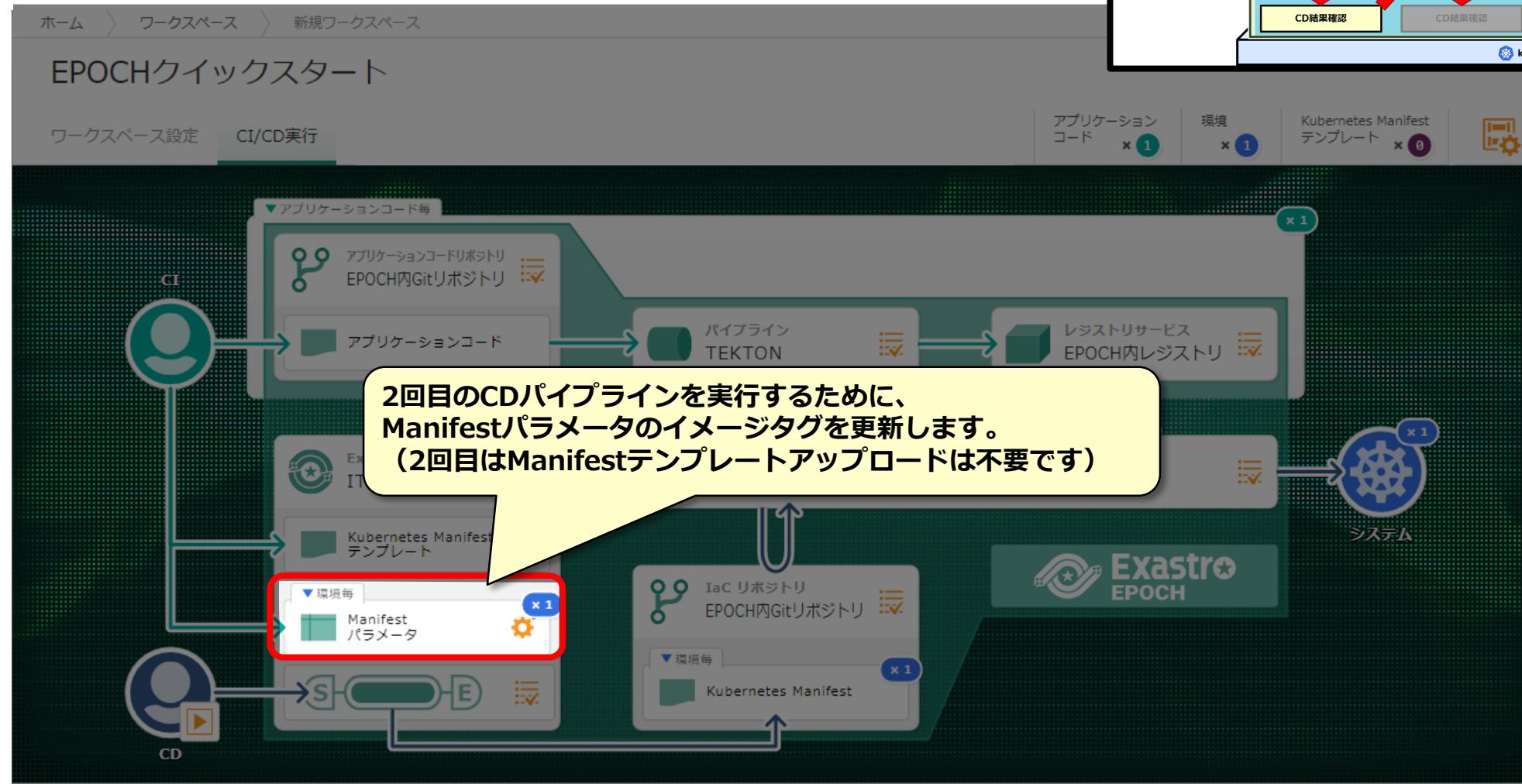
ここで確認したイメージのTagは、  
次の手順で、Manifestパラメータ(image\_tag)に  
手入力が必要になるため控えておいてください

※今後、パイプラインで生成されたimage\_tagは選択できるように変更する予定です

# 4.6 2回目のCI/CDワークフロー手順(7/21)

## Manifestパラメータ設定

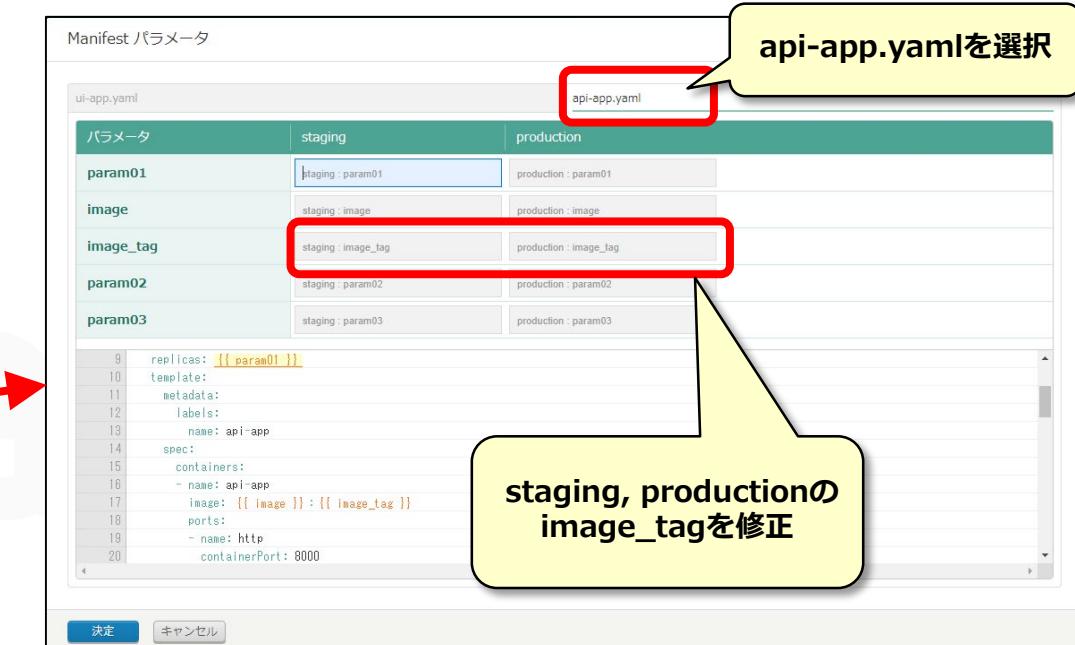
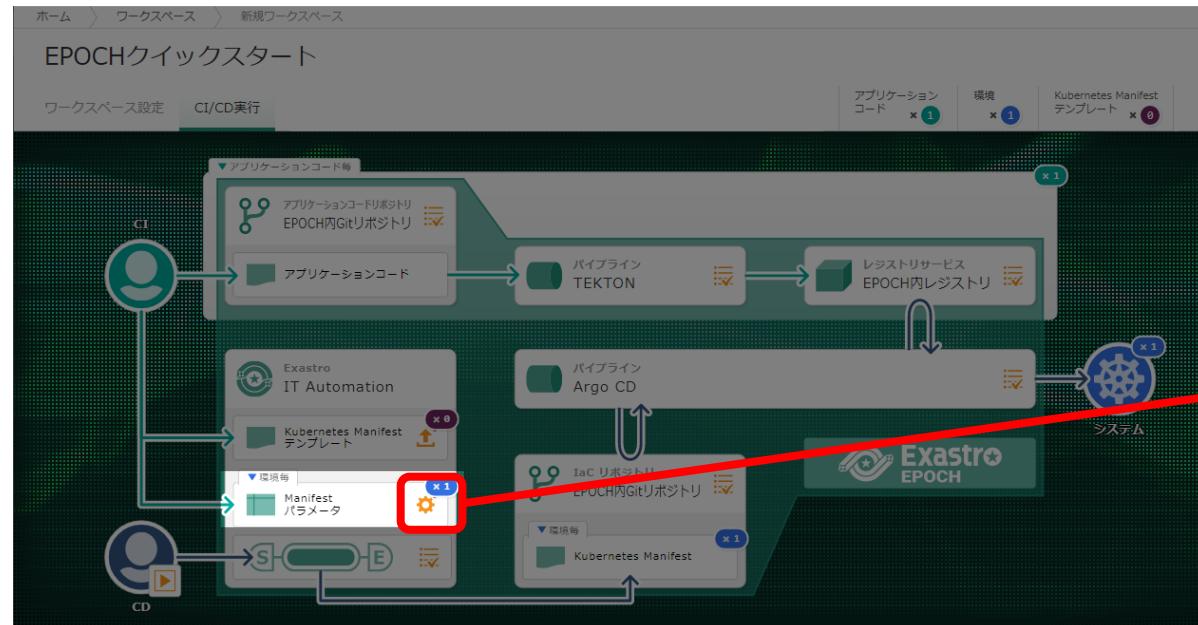
- Manifestパラメータの設定のイメージタグを更新します。



# 4.6 2回目のCI/CDワークフロー手順(8/21)

## Manifestパラメータ(image\_tag)の修正

- Manifestパラメータで、staging, production環境のイメージのタグ名を修正します。



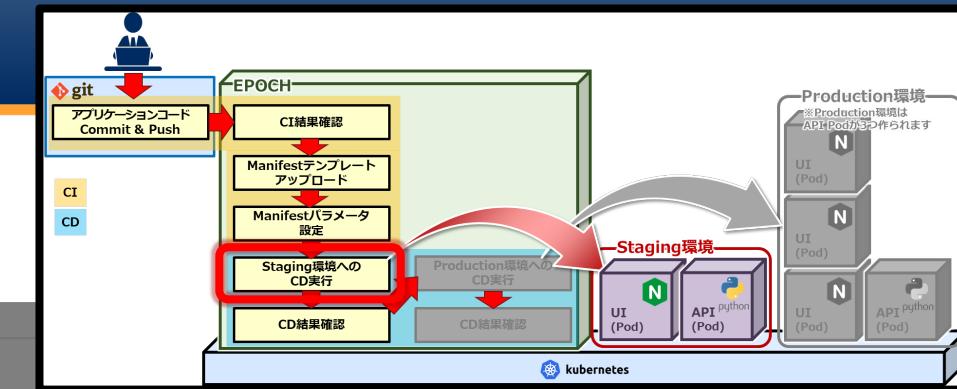
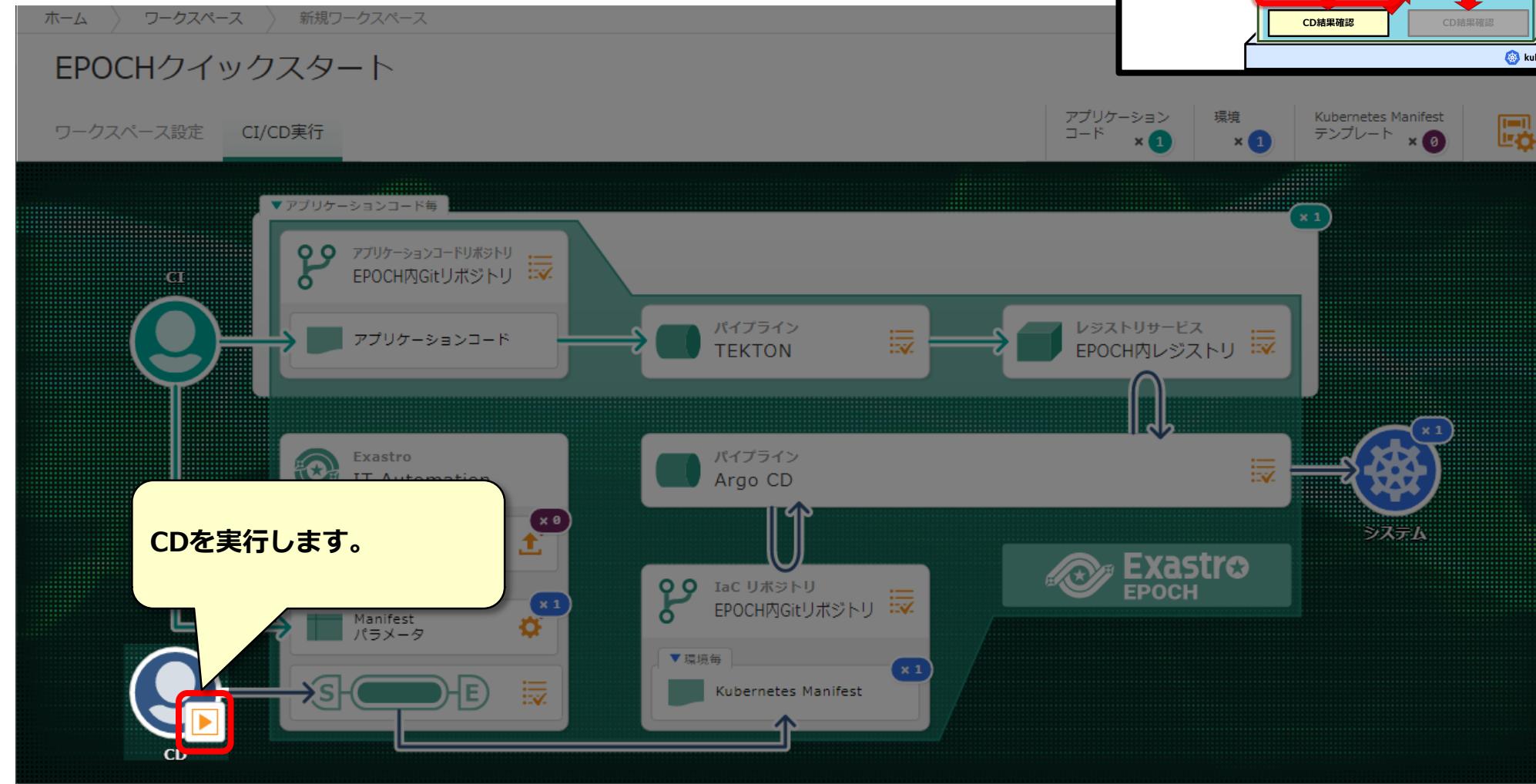
api-app.yaml

項目	入力内容(staging)	入力内容(production)	説明
<code>{{ image_tag }}</code>	[レジストリサービスで確認した最新のイメージのタグ名]	[レジストリサービスで確認した最新のイメージのタグ名]	コンテナイメージのタグ

## 4.6 2回目のCI/CDワークフロー手順(9/21)

### Staging環境へのCD実行

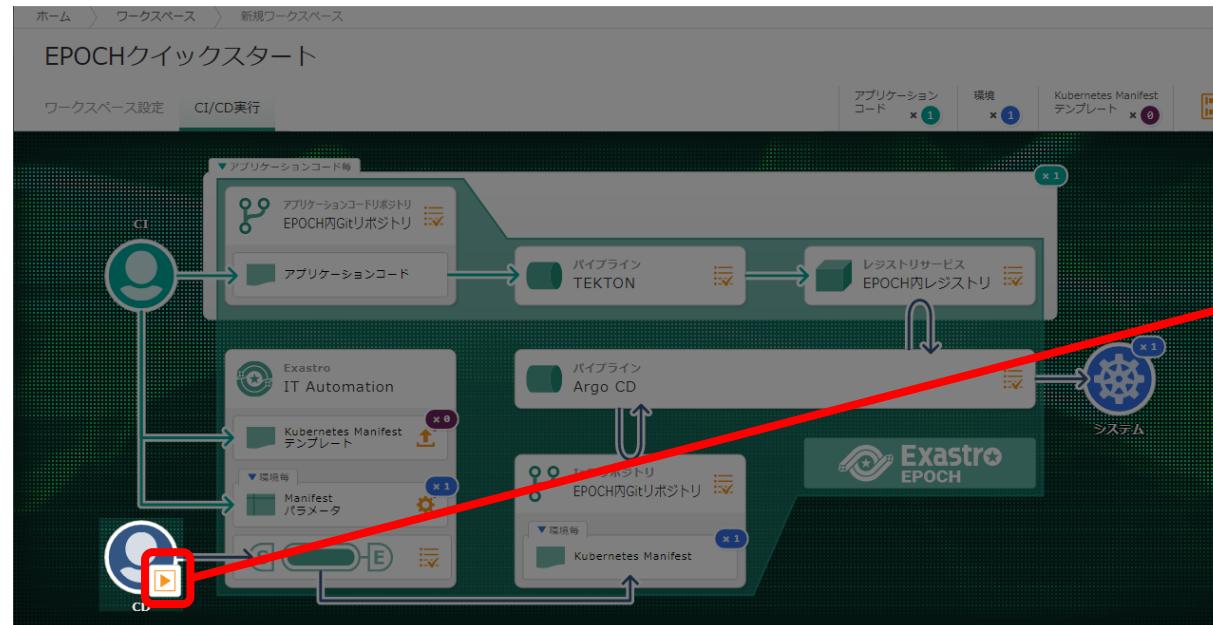
- CD実行で、実際にStaging環境へDeployします。



# 4.6 2回目のCI/CDワークフロー手順(10/21)

## CD実行指定

- Deploy先の環境を選択してDeployを実行します。

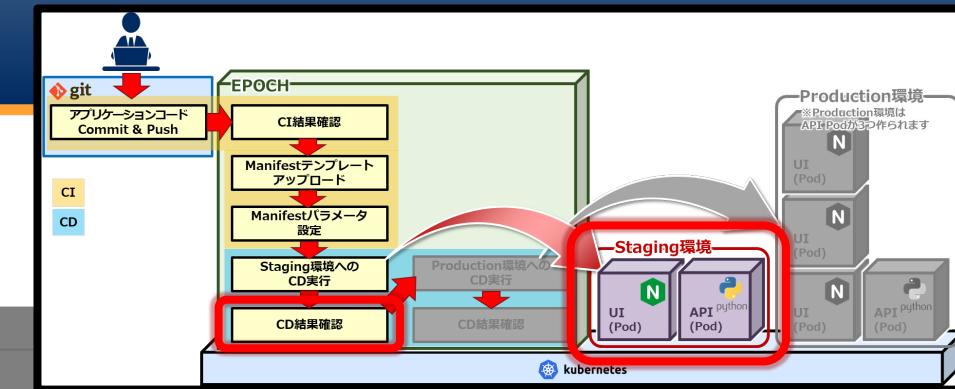


Staging環境へのCD実行が完了しました  
CD実行結果を確認してみましょう

## 4.6 2回目のCI/CDワークフロー手順(11/21)

### Staging環境のCD結果確認

- CDの実行結果は、Exastro IT-Automation、ArgoCDより確認できます。



# 4.6 2回目のCI/CDワークフロー手順(12/21)

## Manifestファイルの生成確認(Staging環境)

- Exastro IT-Automationから、IaCリポジトリへManifestファイルを登録するまでの状況を確認します。

The screenshot illustrates the workflow for generating and confirming Manifest files in the Staging environment using the Exastro IT Automation platform.

**Top Left:** A screenshot of the Exastro IT Automation home page showing the CI/CD pipeline. A red box highlights the 'Conductor' icon in the bottom navigation bar.

**Top Right:** A 'Login' dialog box from the Exastro IT Automation Conductor interface. A yellow callout box contains the text: 'ログインID: epoch-user パスワード: c2jthascR93ijdcyzwJY でログインします' (Login ID: epoch-user Password: c2jthascR93ijdcyzwJY Login).

**Middle Left:** A screenshot of the Conductor interface showing the 'Conductor' menu and a list of tasks. A red box highlights the 'Conductor作業一覧' (Conductor Task List) option. A yellow callout box contains the text: '[1]Conductorメニュー > Conductor作業一覧を選択します' (Select Conductor Task List from the Conductor menu).

**Middle Center:** A screenshot of the Conductor Task List screen. A red box highlights the 'フィルタ' (Filter) button. A yellow callout box contains the text: '[2]フィルタをクリック' (Click Filter).

**Middle Right:** A screenshot of the Conductor interface showing the task details. A red box highlights the '詳細' (Details) button for a specific task. A yellow callout box contains the text: '[3] [CD実行] の詳細をクリック' (Click the [CD execution] details).

**Bottom Right:** A screenshot of the Conductor interface showing the task status. A yellow callout box contains the text: 'すべて[DONE]と表示されていれば完了です' (If all are displayed as [DONE], it is completed).

## 4.6 2回目のCI/CDワークフロー手順(13/21)

### パイプラインArgoCDの結果確認(Staging環境)

- Manifestがkubernetesに反映されるまでの状況を確認します。

The screenshot shows the EPOCH UI interface. On the left, the 'EPOCHクイックスタート' (Quick Start) panel displays the CI/CD pipeline flow: Application Code → Pipeline TEKTON → Registry Service EPOCH内レジストリ. A red arrow points from this panel to the central 'Let's get stuff deployed!' screen, which features a cartoon octopus character.

In the center, there is a 'staging' application detail view. A yellow callout box says 'stagingを選択' (Select staging). Another red box highlights the 'staging' section in the list. A red arrow points from this section to the right-hand 'Sync OK' status message.

On the right, a 'Sign In' dialog box is shown with a yellow callout: 'Username: admin' and 'Password: iYSCKzx2wvxJnn4dCNwN'. Below it, a note says: '※「この接続ではプライバシーが保護されません」が表示されますが、詳細表示から「アクセスする」を選択してログイン画面に遷移します' (Note: 'This connection does not protect privacy' is displayed, but select 'Access' in the detailed view to navigate to the login screen).

A large yellow callout at the bottom right says: '3分ごとに同期されますので同期されることを待ちます。' (Sync occurs every 3 minutes, so wait for synchronization).

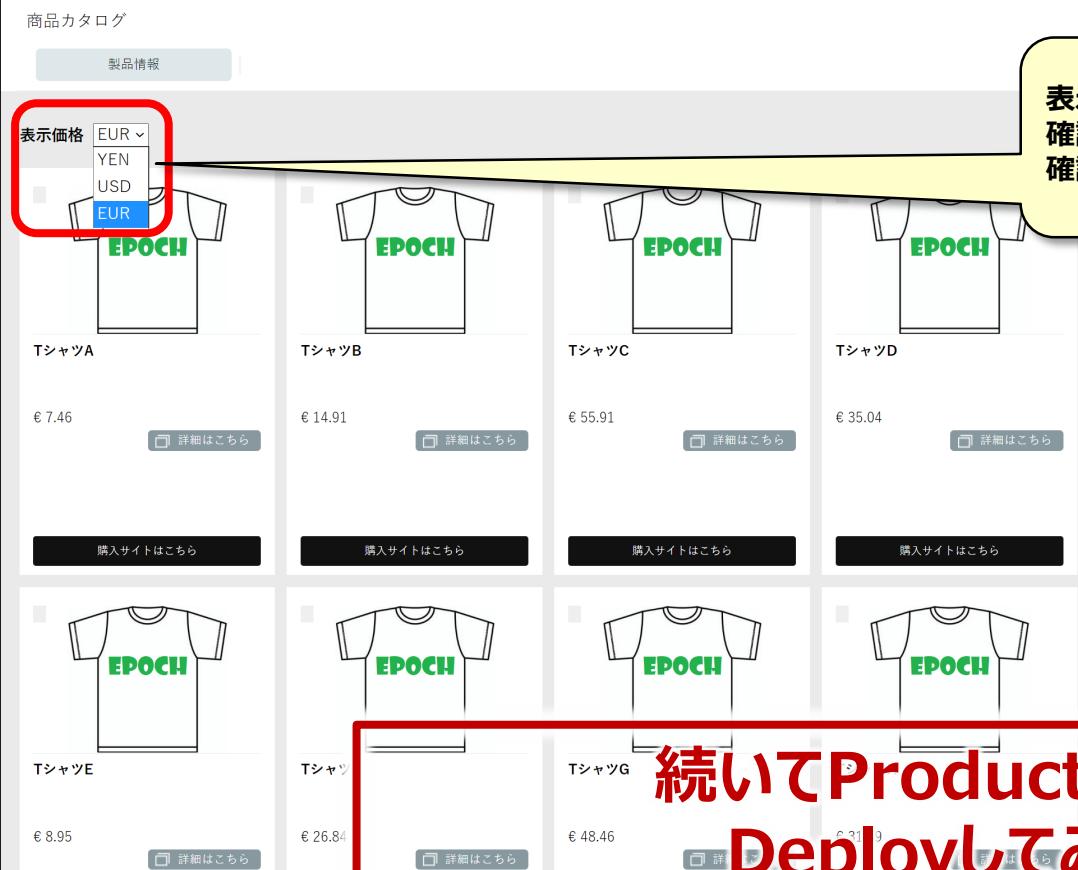
At the bottom, a large red box contains the text: 'Deployされたサンプルアプリケーションを確認してみましょう' (Let's check the deployed sample application).

## 4.6 2回目のCI/CDワークフロー手順(14/21)

### Staging環境のアプリケーションの確認

- ブラウザで以下のURLに接続し、デプロイしたサンプルアプリケーションを表示します

[http://\[Kubernetes masterノードのIPアドレスまたはホスト名\]:31001/front-end.html](http://[Kubernetes masterノードのIPアドレスまたはホスト名]:31001/front-end.html)



表示価格の選択欄に「YEN」「USD」「EUR」が表示されていることを確認します。  
確認後、「EUR」を選択して、表示が切り替わることを確認します。

商品カタログ

表示価格 EUR  
YEN  
USD  
EUR

TシャツA TシャツB TシャツC TシャツD

€ 7.46 詳細はこちら 購入サイトはこちら

TシャツE TシャツF TシャツG TシャツH

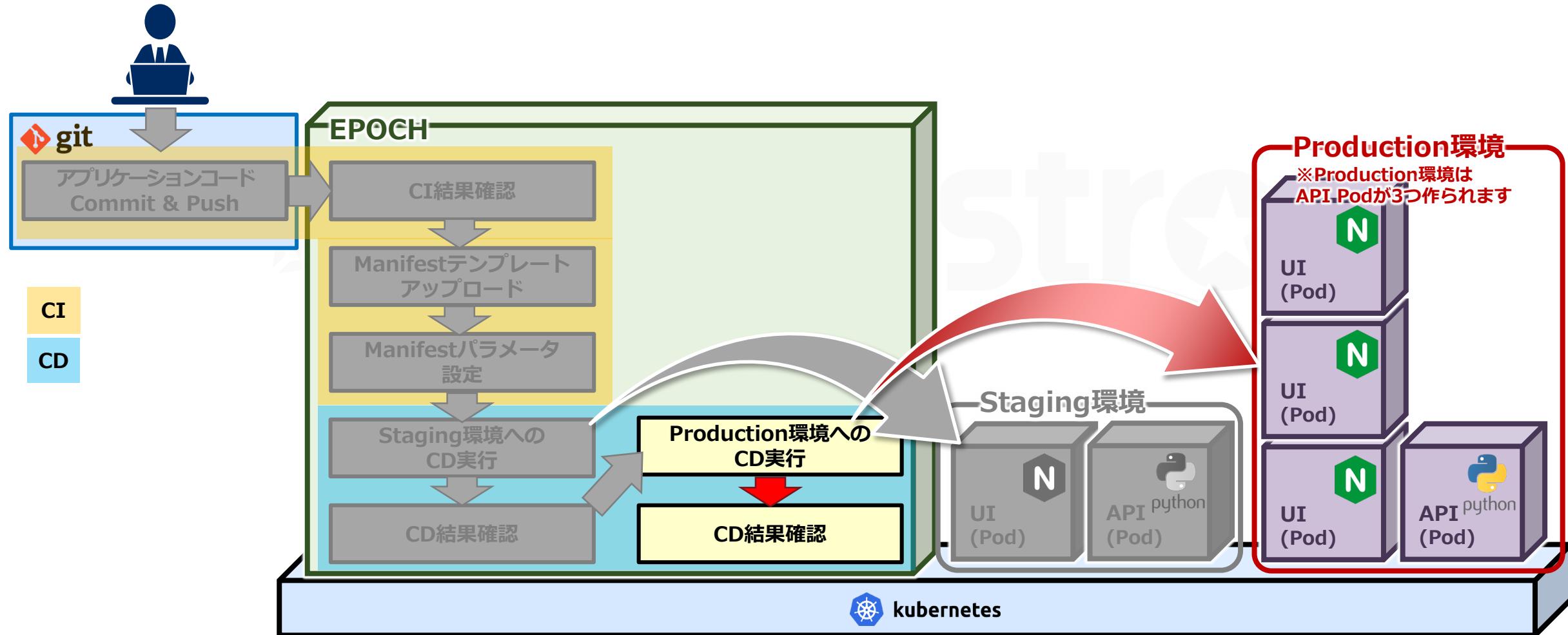
€ 8.95 € 26.84 € 48.46 € 31.99 詳細はこちら 購入サイトはこちら

続いてProduction環境へ Deployしてみましょう

## 4.6 2回目のCI/CDワークフロー手順(15/21)

### Production環境へのDeploy

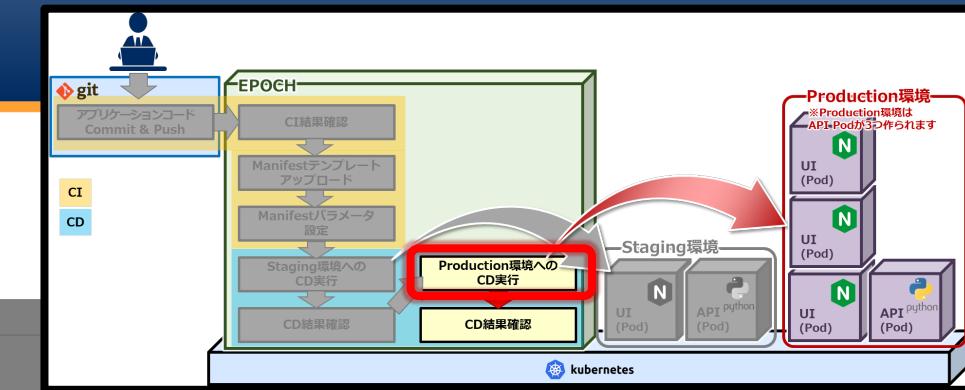
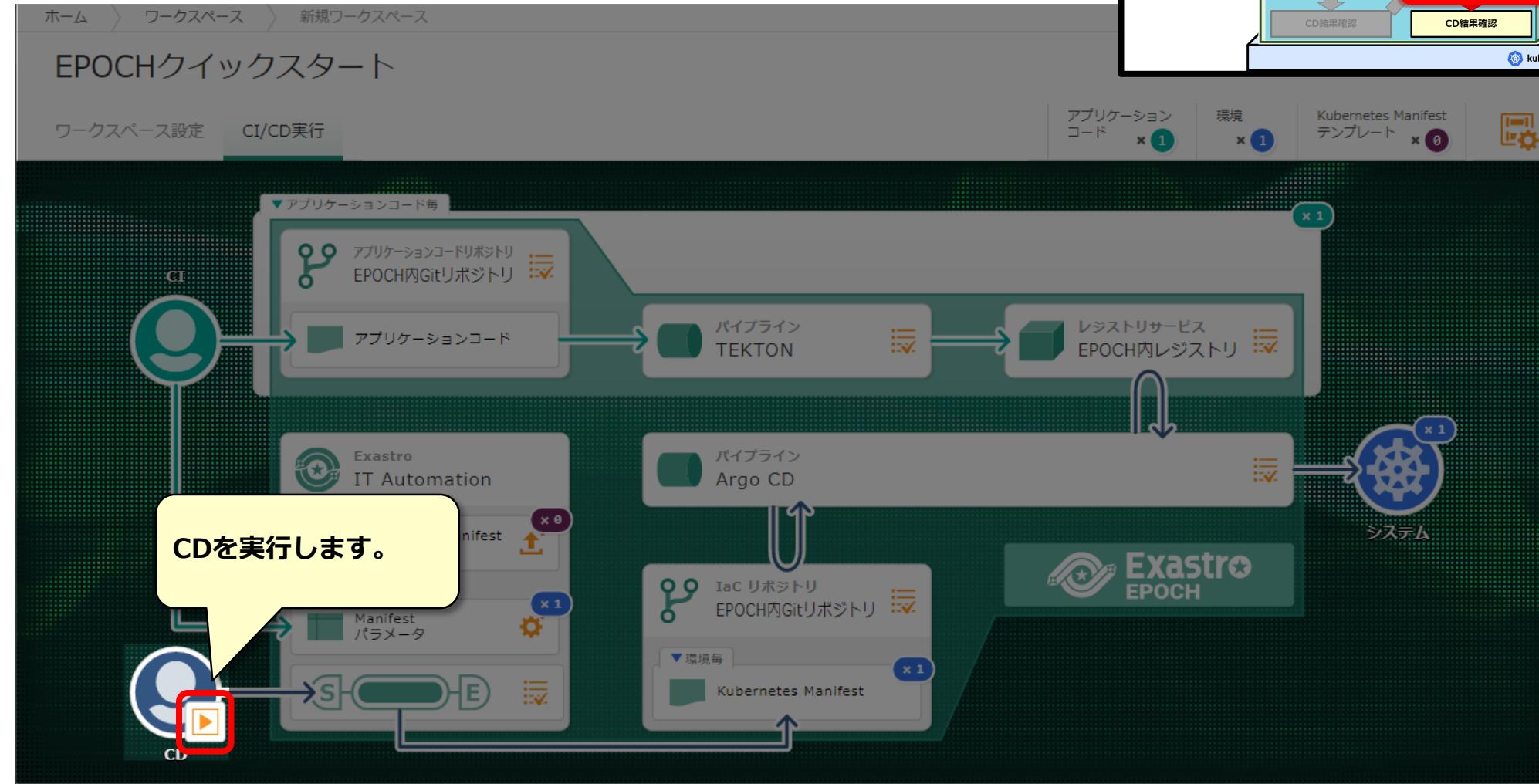
Staging環境へDeploy後、Production環境へDeployする流れを説明します。



## 4.6 2回目のCI/CDワークフロー手順(16/21)

### Production環境へのCD実行

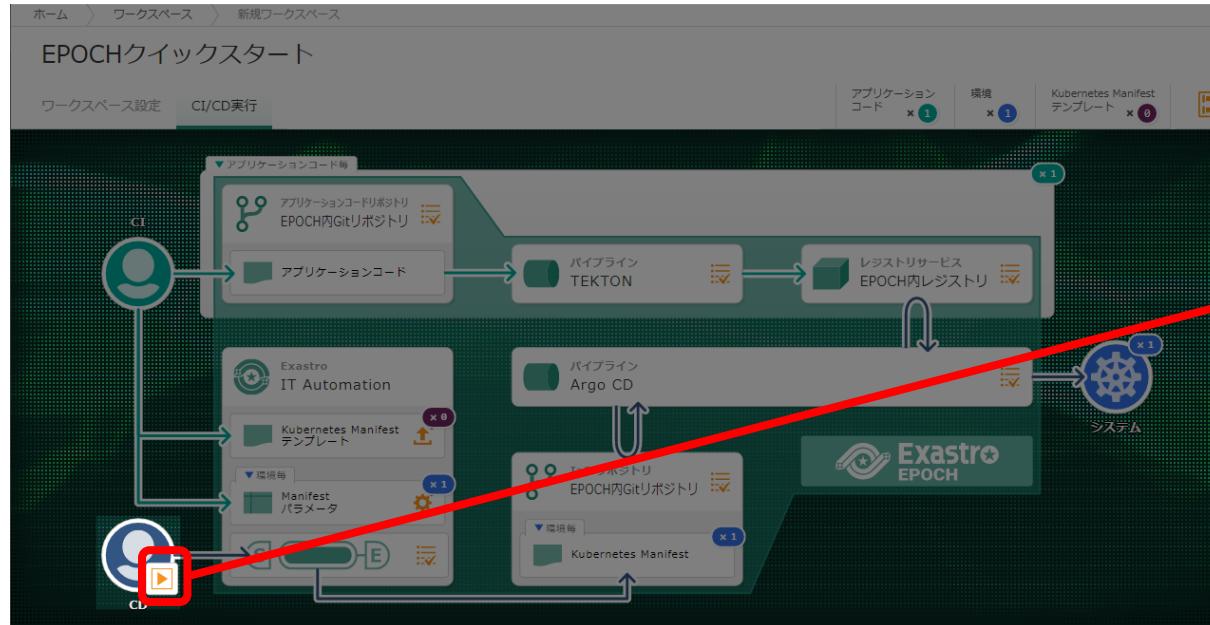
- CD実行で、実際にProduction環境へDeployを実行します。



## 4.6 2回目のCI/CDワークフロー手順(17/21)

### CD実行指定

- Deploy先の環境を選択してDeployを実行します。



The dialog box contains the following fields:

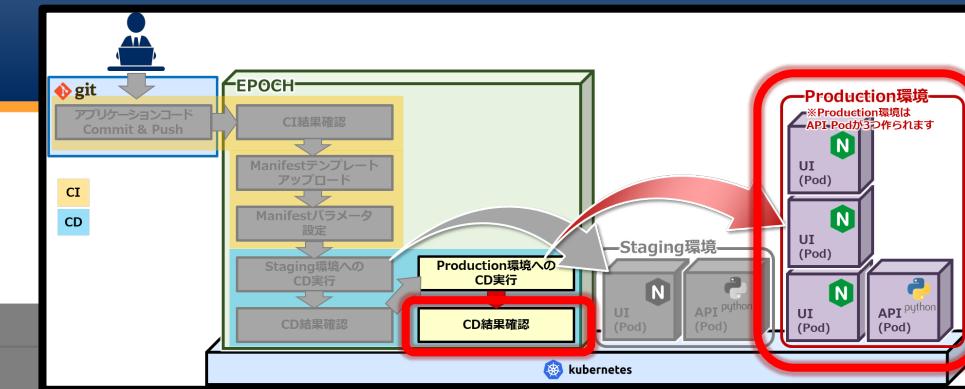
- 実行条件 (Execution Conditions): CD実行日時 (Execution Date/Time) set to 即実行 (Now), 予約日時指定 (Scheduled Date/Time) set to 2021/07/08 14:17.
- 環境 (Environment): A dropdown menu currently set to staging, which is highlighted with a red border. A callout bubble says "productionを選択します" (Select production).
- Manifest/パラメータ (Manifest/Parameters): Environment name: staging, Manifest repository: https://github.com/epoch-team/argocd\_manifest.git, Kubernetes API Server URL: https://kubernetes.default.svc, Namespace: staging-app.
- ArgoCDパイプライン (ArgoCD Pipeline): Below this section, there is a note: "以下の内容でDeployします。よろしいですか？" (Are you sure to Deploy with the following content?).
- 操作 (Operations): A '実行' (Execute) button at the bottom, which is also highlighted with a red border. A callout bubble says "環境選択後、実行ボタンを押下します" (Press the execute button after selecting the environment).

Production環境へのCD実行が完了しました  
CD実行結果を確認してみましょう

## 4.6 2回目のCI/CDワークフロー手順(18/21)

### Production環境のCD結果確認

- CDの実行結果は、Exastro IT-Automation、ArgoCDより確認できます。



# 4.6 2回目のCI/CDワークフロー手順(19/21)

## Manifestファイルの生成確認(Production環境)

- Exastro IT-Automationから、IaCリポジトリへManifestファイルを登録するまでの状況を確認します。

The screenshot illustrates the workflow for generating and confirming Manifest files in the Production environment. It shows the Exastro IT Automation interface with various components like EPOCH, TEKTON, Argo CD, and Conductor.

**Top Left:** Shows the CI/CD pipeline. A red box highlights the 'Conductor' icon in the bottom right corner of the pipeline diagram.

**Top Right:** Shows the 'Login' screen for Exastro IT Automation. A yellow callout box contains the login information: **ログインID: epoch-user** and **パスワード: c2jthascR93ijdcyzwJY**.

**Middle Left:** Shows the 'Conductor' menu. A yellow callout box [1] indicates selecting the 'Conductor作業一覧' (Conductor Job List) option. A red box highlights the 'Conductor作業一覧' button in the menu.

**Middle Center:** Shows the 'Conductor Job List' screen. A yellow callout box [2] indicates clicking the 'フィルタ' (Filter) button. A red box highlights the 'フィルタ' button.

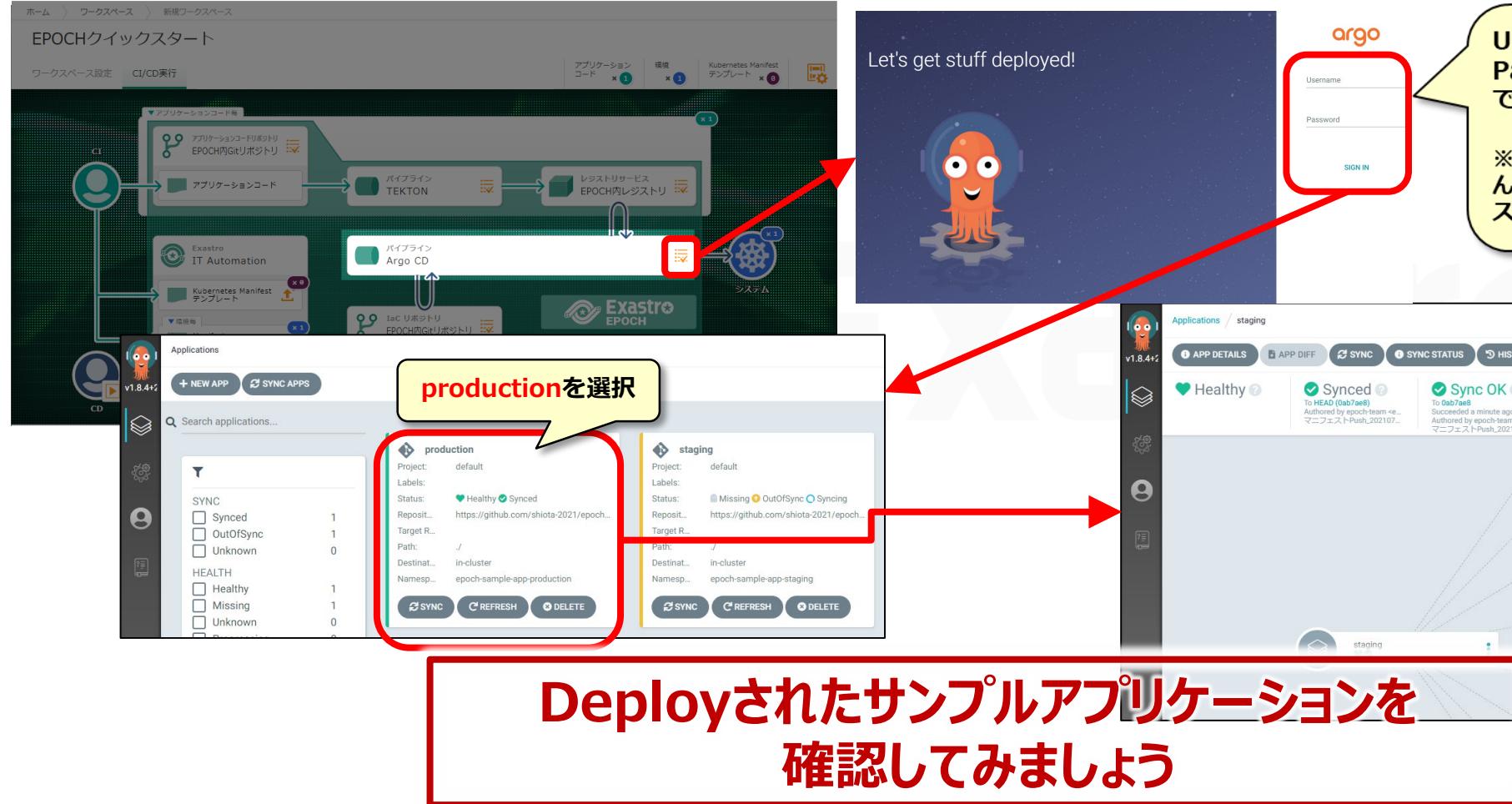
**Bottom Left:** Shows the details of a specific job. A yellow callout box [3] indicates clicking the '詳細' (Details) button. A red box highlights the '詳細' button.

**Bottom Right:** Shows the 'Checking' screen of the Conductor interface. A yellow callout box indicates that all tasks are completed, with the message: **すべて[DONE]と表示されていれば完了です** (If all are displayed as [DONE], it is complete).

# 4.6 2回目のCI/CDワークフロー手順(20/21)

## パイプラインArgoCDの結果確認(Production環境)

- Manifestがkubernetesに反映されるまでの状況を確認します。



## 4.6 2回目のCI/CDワークフロー手順(21/21)

### Production環境のアプリケーションの確認

- ブラウザで以下のURLに接続し、デプロイしたサンプルアプリケーションを表示します

[http://\[Kubernetes masterノードのIPアドレスまたはホスト名\]:31003/front-end.html](http://[Kubernetes masterノードのIPアドレスまたはホスト名]:31003/front-end.html)





## 5. 付録

本資料中で行った内容の補足をします。

## 5.1 注意事項・制限事項

以下、現在のExastro EPOCHのバージョンでの制限事項となります。今後のバージョンで変更される可能性があります。

### ● 制限事項（今後対応する予定）

- アプリケーションコードのリポジトリは、現在1つのみ対応となっております。
- 現在は、アプリケーションコード毎のGitアカウントには対応しておりません。
- Gitサービス選択は次バージョン以降で対応予定です。現在は指定されたURLのGitリポジトリの動作となります。
- ビルドブランチは次バージョン以降で対応予定です。現在はPushされた内容でビルドされます。
- 静的解析は次バージョン以降で対応予定です。現在はSonarQubeを選択した場合に動作しません。
- レジストリサービスは現在内部のレジストリサービスのみとなっております。
- イメージ出力先以外の項目については次バージョン以降で対応予定です。
- Authentication token, Base64 encoded certificateは次バージョン以降で対応予定です。
- テンプレートで指定できる変数は、現在固定です。詳細は「コラム」を参照してください。

### ● 注意事項

- EPOCHをインストールすると、TEKTONもインストールされます。
- 変数は "{{ 変数名 }}" で指定した内容となります。

# コラム：Manifestテンプレートと変数名

Manifestテンプレートをアップロードするとファイル内の定義文字が解析され、パラメータ入力できる状態になります。

```
9  replicas: {{ param01 }}  
10 template:  
11   metadata:  
12     labels:  
13       name: api-app  
14   spec:  
15     containers:  
16       - name: api-app  
17         image: {{ image }}:{{ image_tag }}  
18         ports:  
19           - name: http  
20             containerPort: 8000
```

`{{ 変数名 }}` の形式で記述された文字が変数として認識され、ユーザが入力できるようになります。  
現在、EPOCHで使用できる変数名は以下の通りとなります。

変数名	説明
<code>{{ image }}</code>	コンテナイメージ
<code>{{ image_tag }}</code>	コンテナイメージのタグ
<code>{{ param01 }}</code>	ユーザが自由に使用できる固定の変数名01
<code>{</code>	
<code>{{ param20 }}</code>	ユーザが自由に使用できる固定の変数名20（20が上限）

※ユーザ任意の変数名につきましては、今後対応する予定です



**Exastro** 