



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の準備

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 チュートリアルの概要
- 4.2 サンプルアプリの構成
- 4.3 チュートリアルの流れ(CI/CDワークフロー)
- 4.4 Manifestテンプレートファイルについて
- 4.5 1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

5. 付録

- 5.1 注意事項・制限事項

1. はじめに

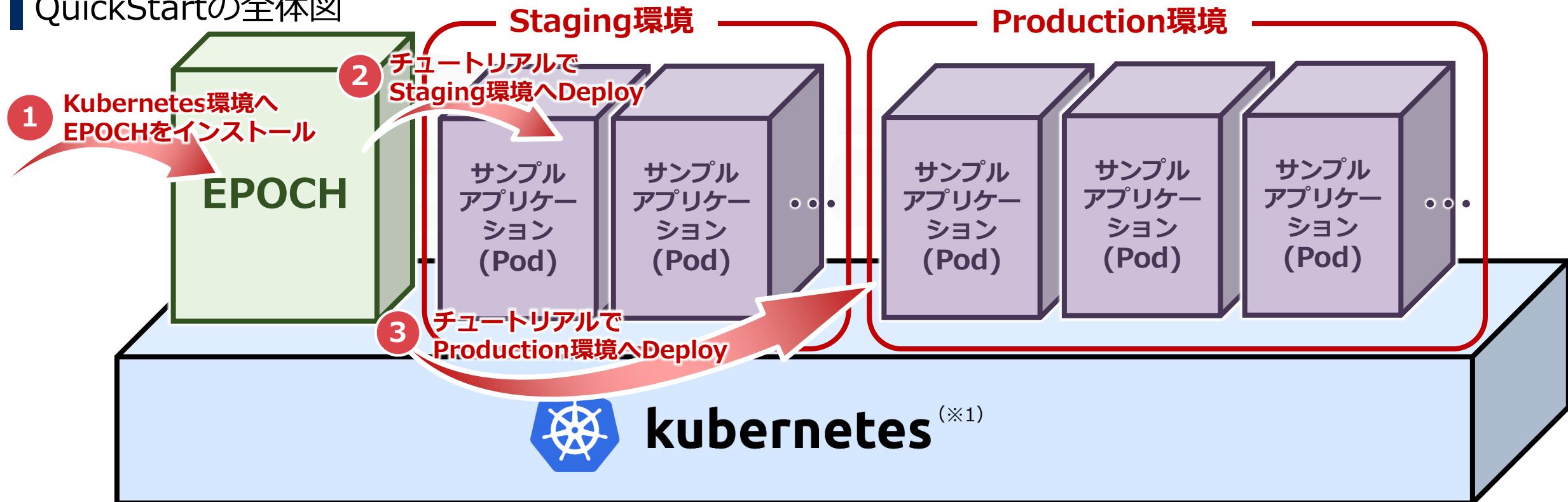


1.1 QuickStartについて

はじめに

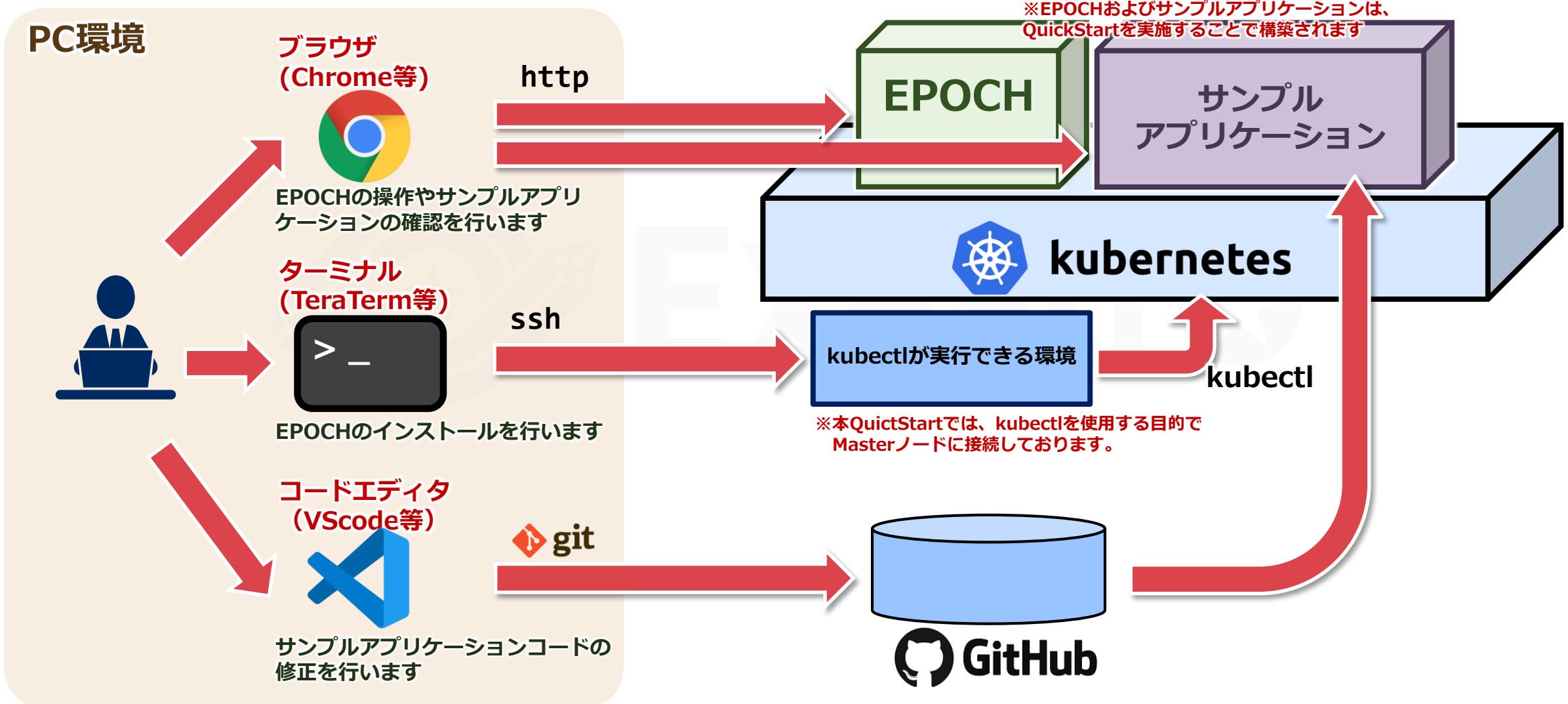
本書は、Exastro EPOCH(以降、EPOCHと表記する)の導入方法ならびに簡単な使い方をチュートリアルを用いて説明します。

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

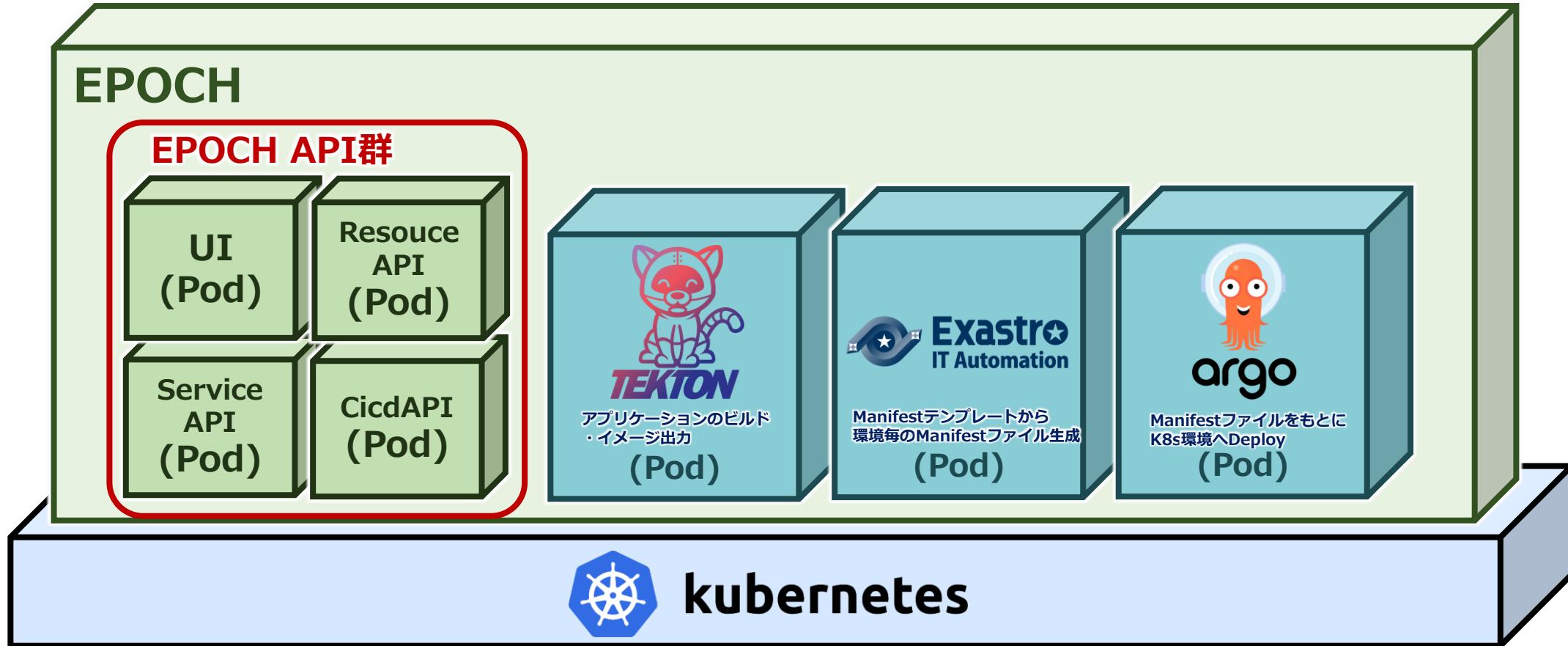
EPOCHをインストールして、CI/CDの環境を準備をしましょう。



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
- PC環境にGit for Windowsがインストールされていること
- 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！

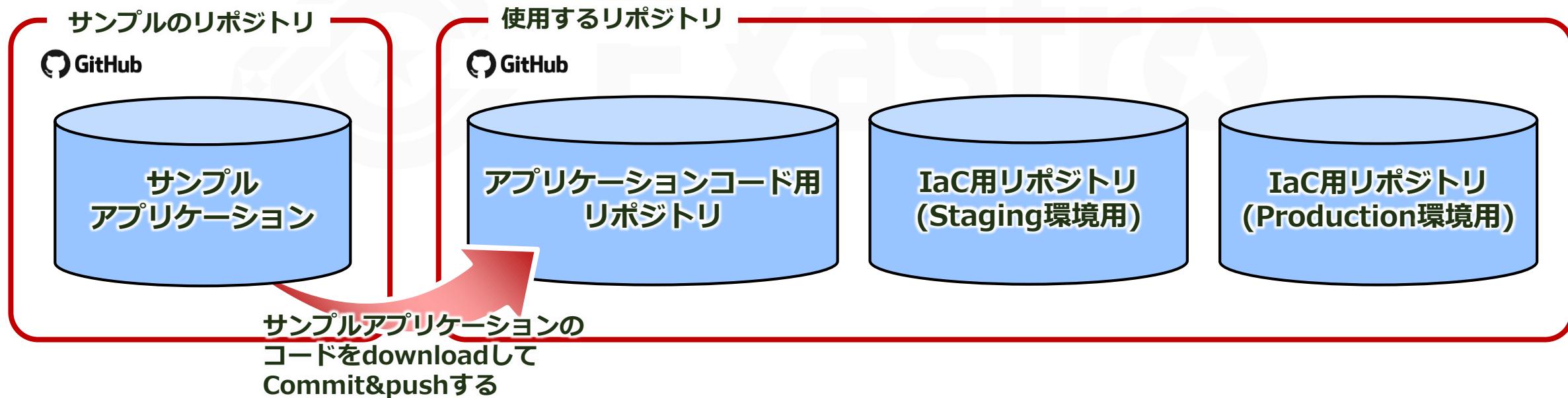
2.2 リポジトリ準備(1/4)

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

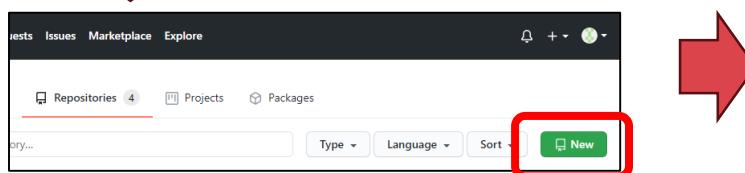
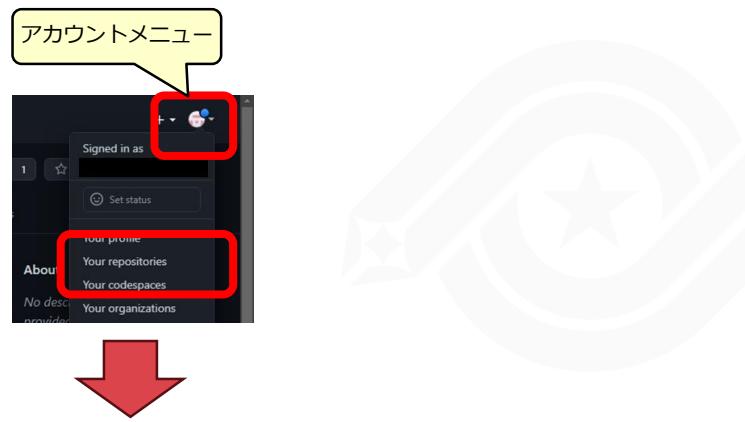


2.2 リポジトリ準備(2/4)

リポジトリの準備

Gitリポジトリを3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。



A screenshot of the GitHub 'Create a new repository' form. A yellow box highlights the 'Repository name' field containing 'epoch-sample-staging-manifest'. A callout bubble says: '次のRepository nameを指定
1 : epoch-sample-app
2 : epoch-sample-staging-manifest
3 : epoch-sample-production-manifest'. A red box highlights the 'Add a README file' checkbox, which is checked. A callout bubble says: 「Add a README File」をチェックON. A red box highlights the 'Create repository' button at the bottom.

Create a new repository

A repository contains all project files, including the revision history. Already have Import a repository.

Owner *

your-account / epoch-sample-staging-manifest ✓

Great repository names are short and memorable. Need inspiration? How about [furry-octo-umbrella](#)?

Description (optional)

Public Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file This is where you can write a long description for your project. [Learn more](#).

Add .gitignore Choose which files not to track from a list of templates. [Learn more](#).

Choose a license A license tells others what they can and can't do with your code. [Learn more](#).

Create repository

2.2 リポジトリ準備(3/4)

■ アプリケーションコード用リポジトリをPC環境へ準備

- 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。

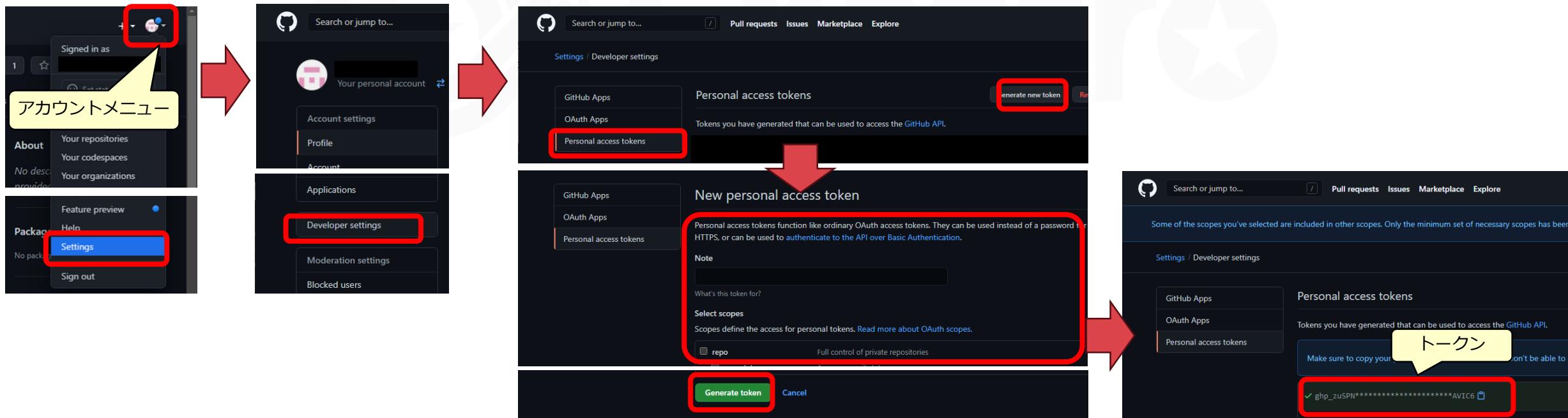
```
> cd "[clone先のフォルダ]"
> git clone https://github.com/[Githubのアカウント名]/epoch-sample-app.git
> cd epoch-sample-app
> git config user.name "[GitHubのユーザ名]"
> git config user.email "[GitHubのemailアドレス]"
```

ここでcloneしたローカルリポジトリを使って、チュートリアルを行います。

2.2 リポジトリ準備(4/4)

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を全て選択し、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。



2.3 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

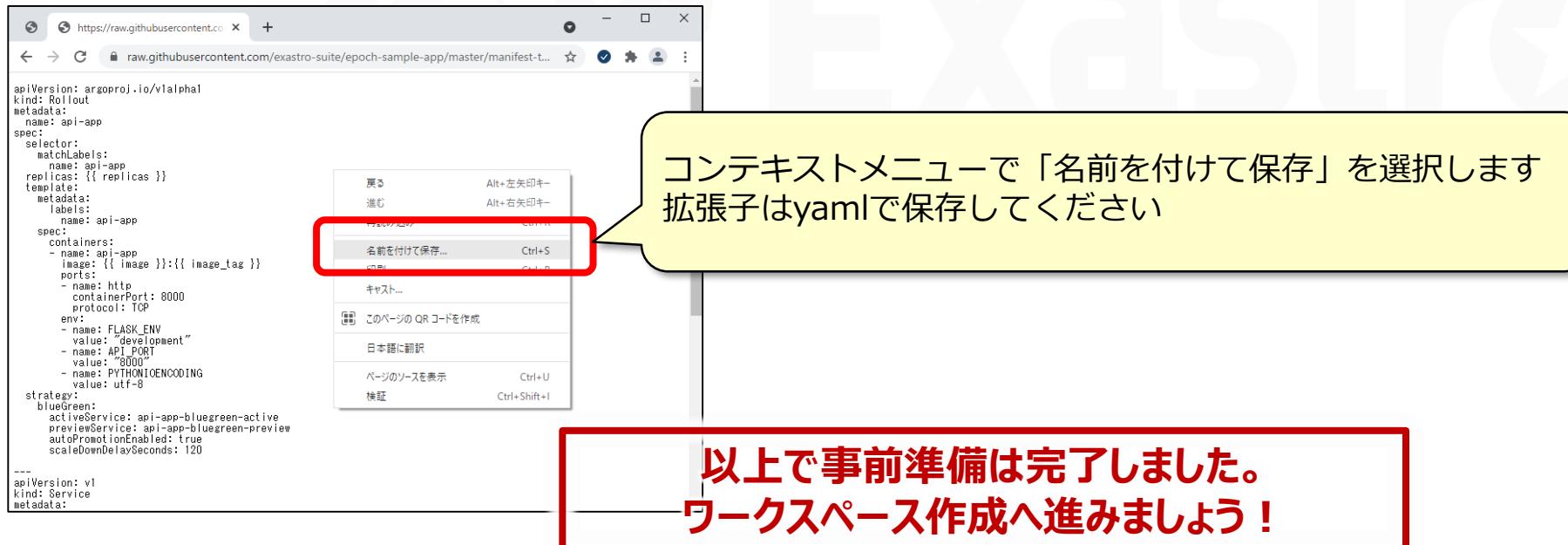
EPOCHにアップロードするManifestテンプレートファイル（2ファイル）をダウンロードします。

- 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

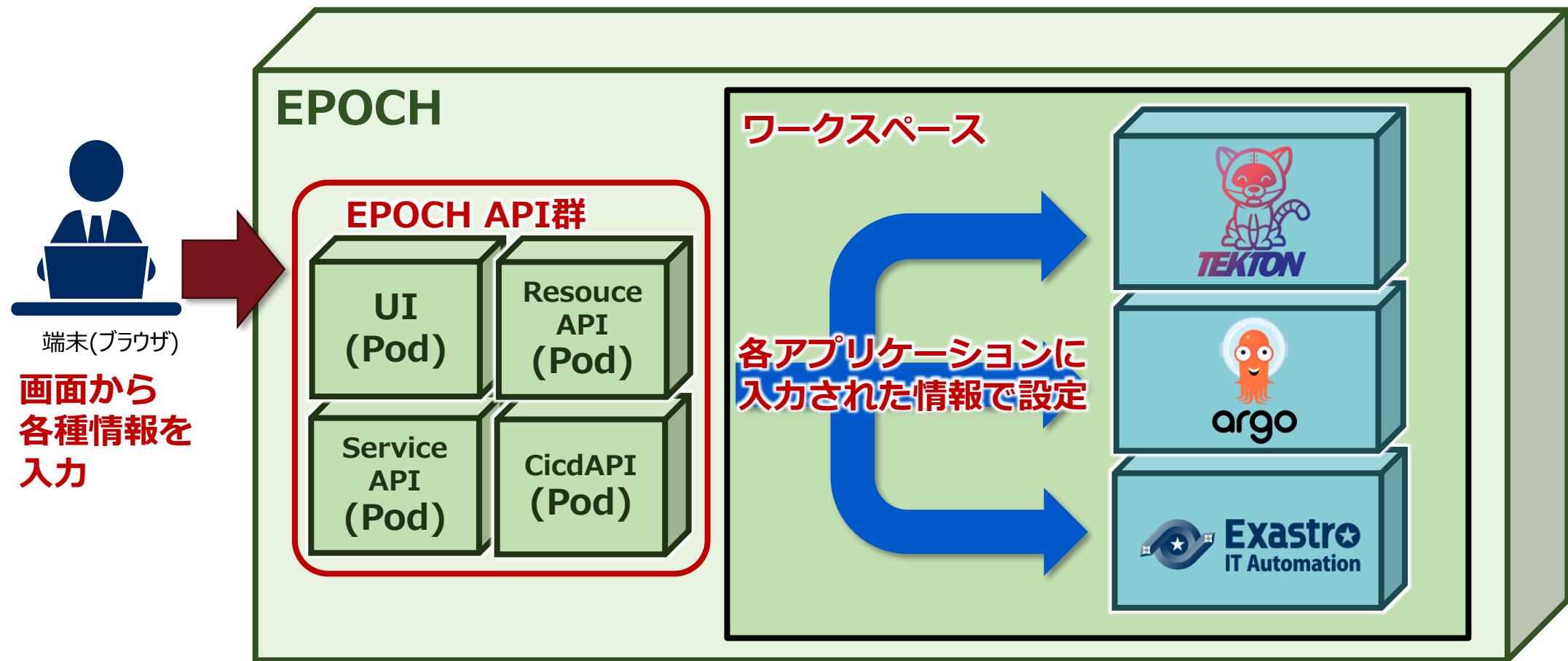
ワークスペースを作成し、CI/CDの準備をしましょう。

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。

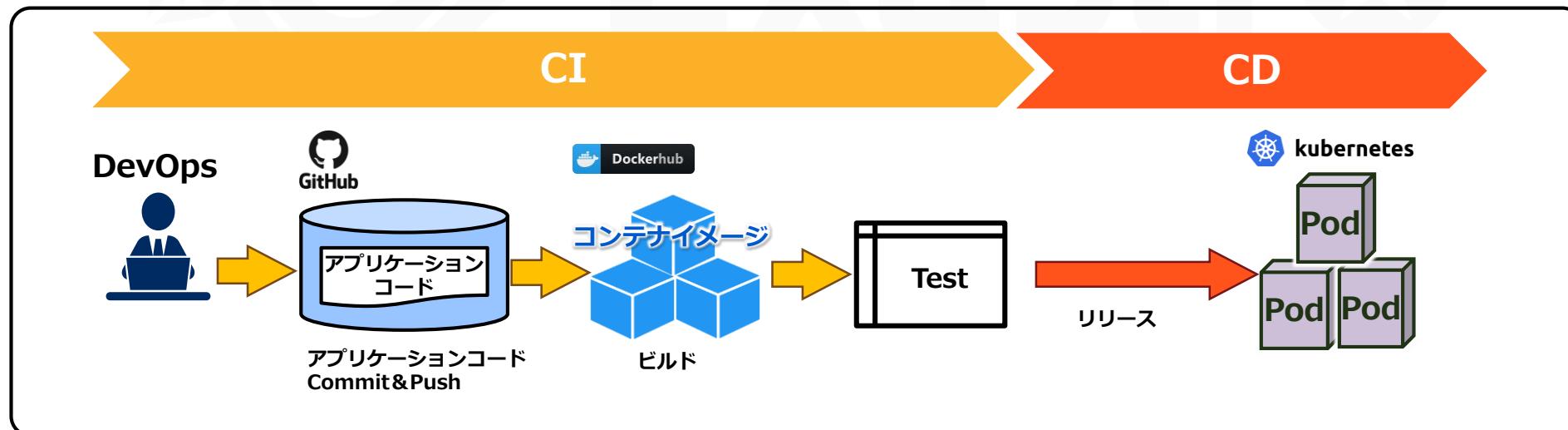
- CI（継続的インテグレーション）

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。

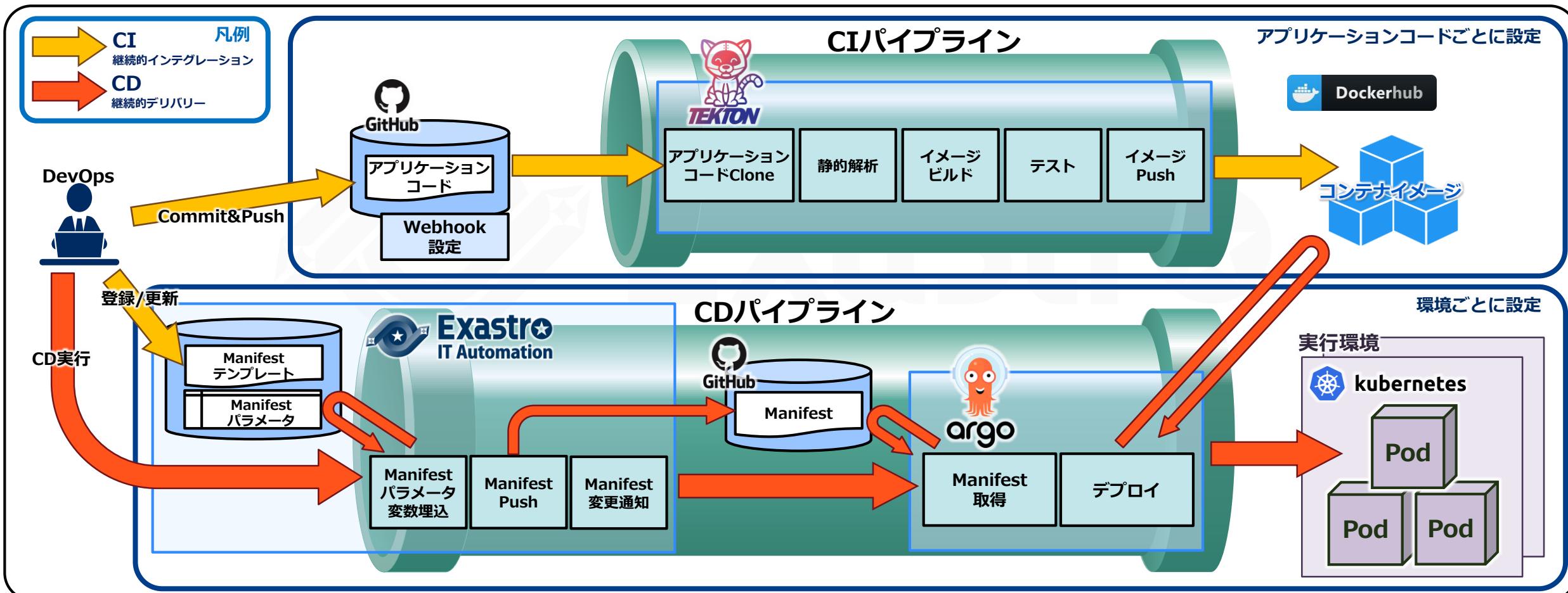
- CI/CDのイメージ



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。

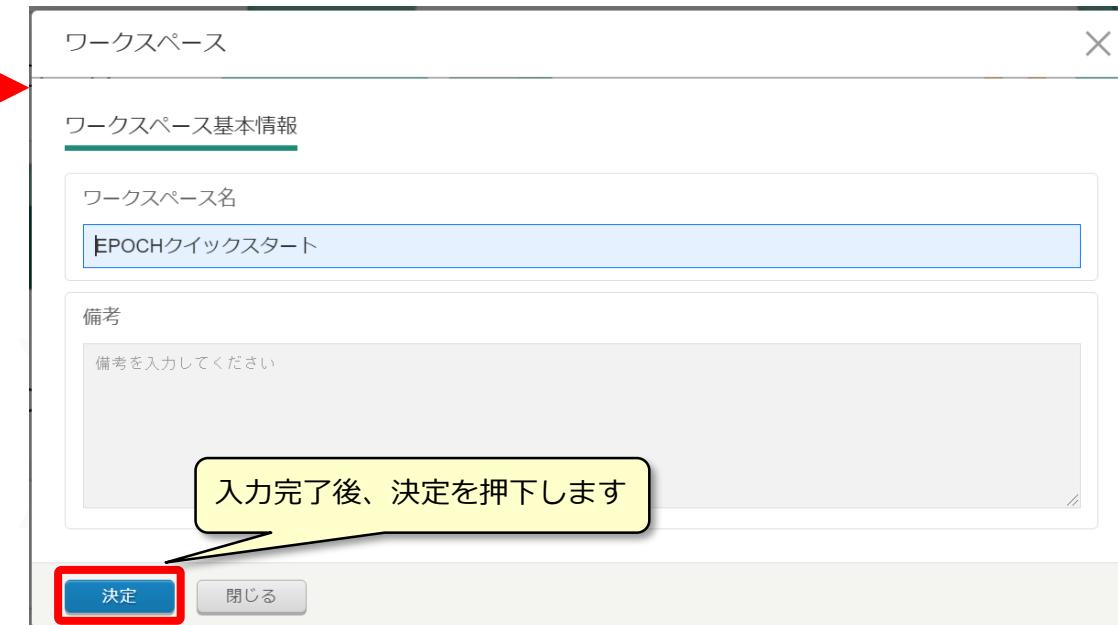
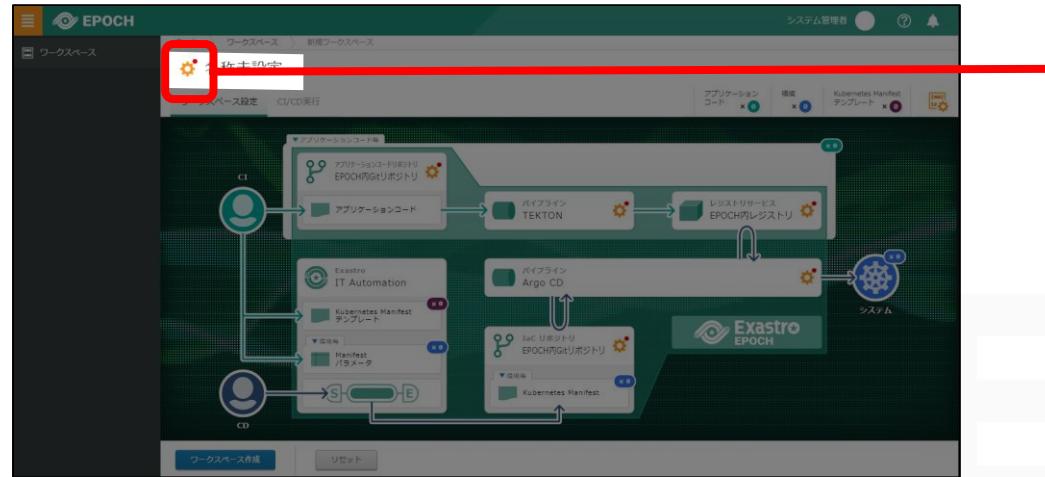
[https://\[インストール先のIPアドレスまたはホスト名\]:30443/workspace.html](https://[インストール先のIPアドレスまたはホスト名]:30443/workspace.html)



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

EPOCH内Gitリポジトリ GitHub

Gitアカウント指定

ユーザ名: your-github-account

トークン:

Gitリポジトリ一覧

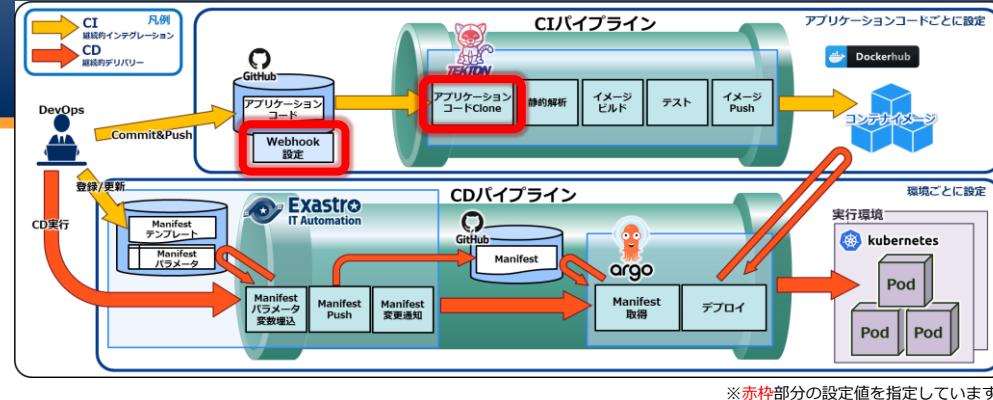
epoch-sample-app

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

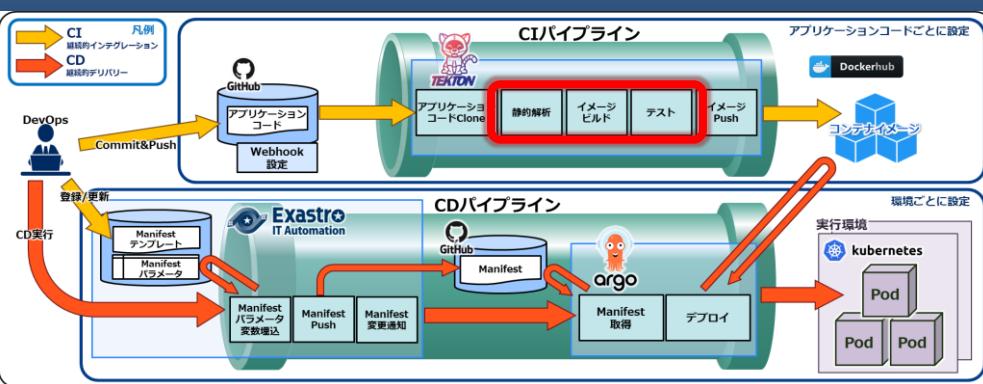
決定 キャンセル

入力完了後、決定を押下します

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	準備で作成したアプリケーションコード用リポジトリの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。

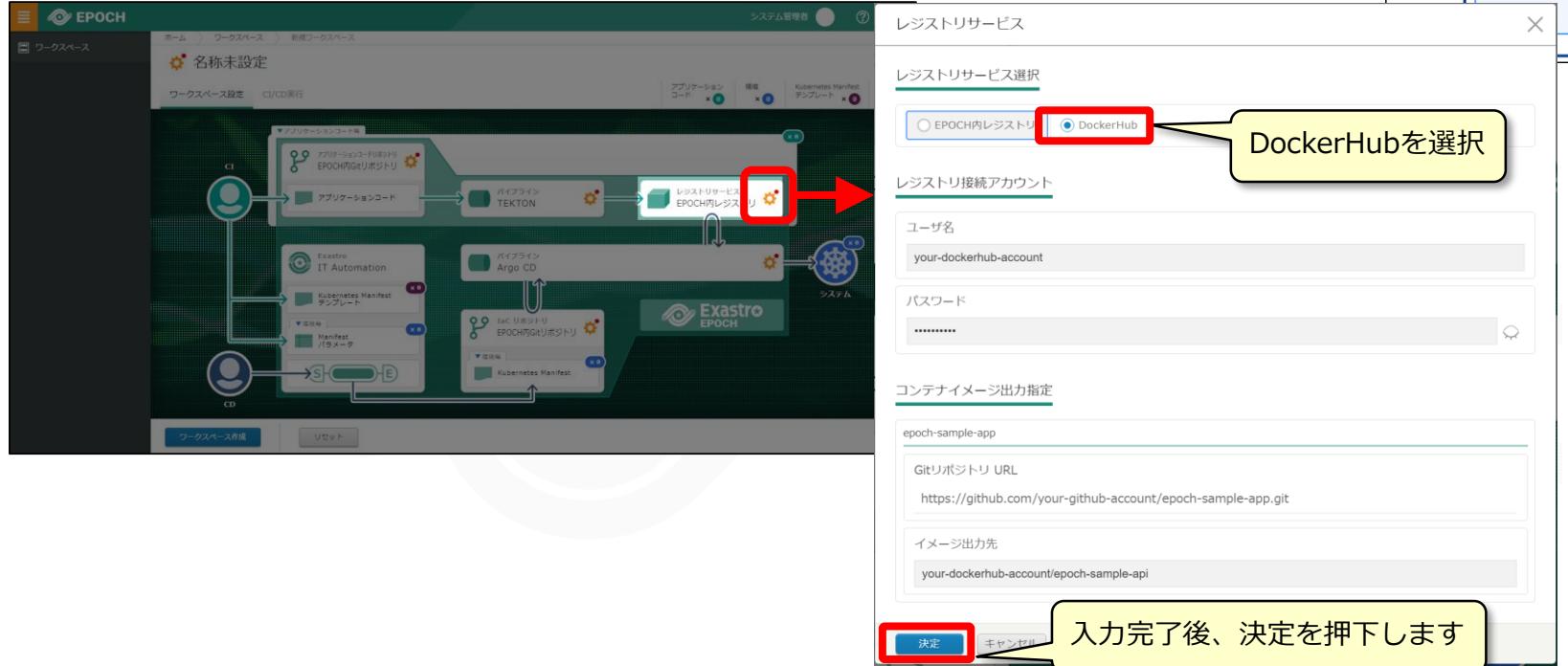


項目	入力・選択内容	説明
ビルドブランチ	main,master	ビルド対象のアプリケーションのGitHubのブランチ
ビルドDockerファイルパス	./api-app/Dockerfile	アプリケーションのDockerfileのパス

3.5 ワークスペース作成(4/7)

レジストリサービス

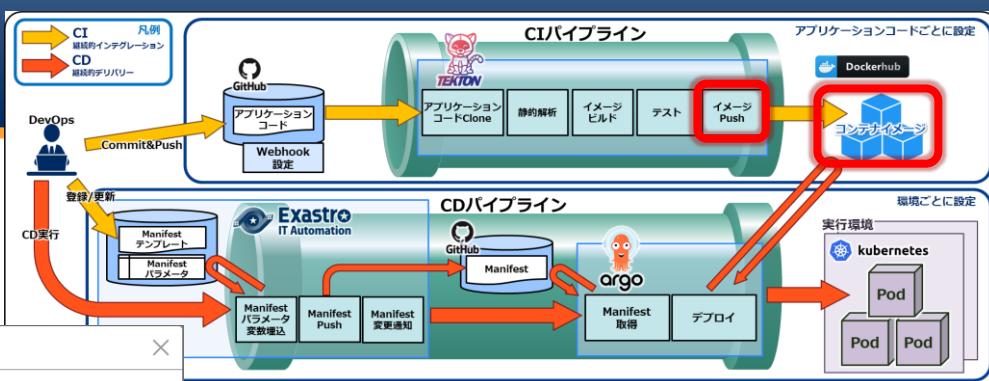
ビルド後のイメージ登録先（レジストリ）情報を入力します。



DockerHubを選択

入力完了後、決定を押下します

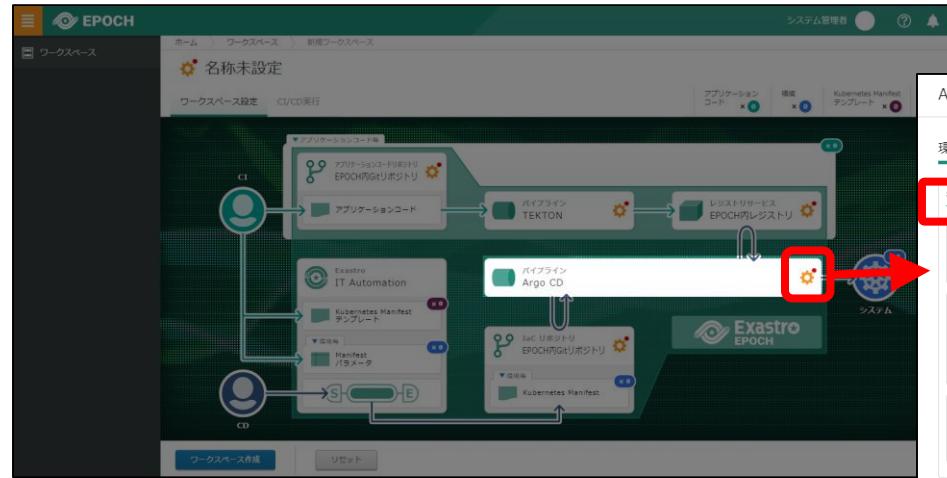
項目	入力・選択内容	説明
ユーザ名	(自身のDockerHubのアカウント名)	DockerHubのアカウント名
パスワード	(自身のDockerHubのパスワード)	DockerHubのパスワード
イメージ出力先	[DockerHubのアカウント名]/epoch-sample- api ※ユーザ名入力後に表示される内容を修正してください。	DockerHubのイメージ出力先のパス



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。



環境の1つめと2つめ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

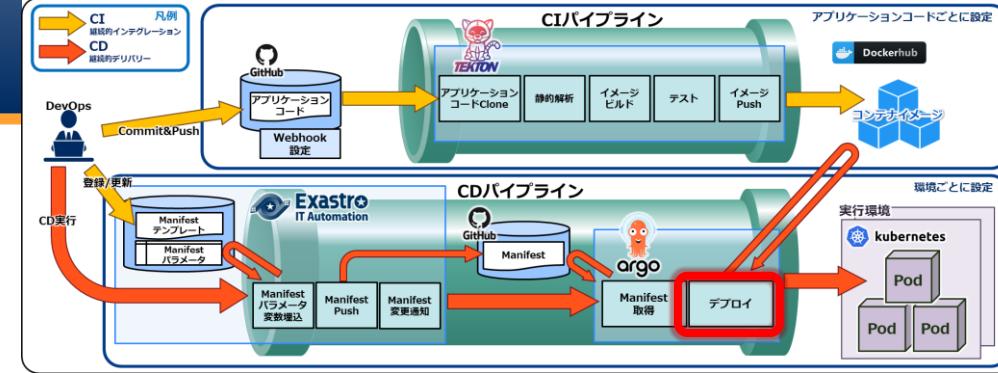
決定 キャンセル

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

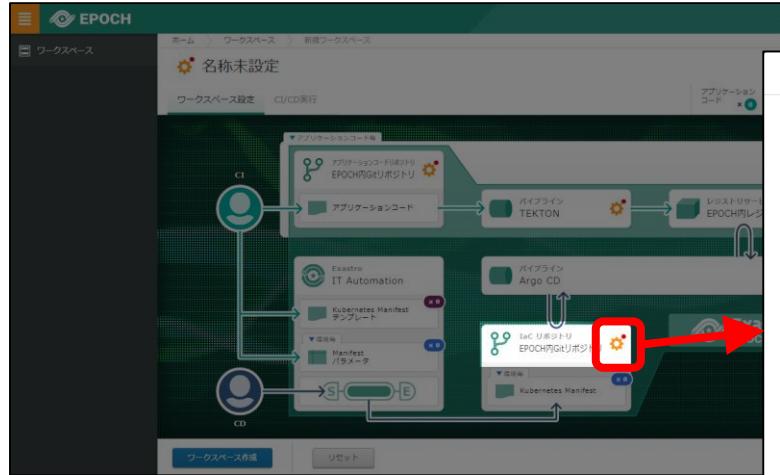
項目	入力・選択内容	説明
環境名	production	デプロイ環境の名前
Namespace	epoch-sample-app-production	デプロイ先のNamespace



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
 GitHub EPOCH内Gitリポジトリ

Gitアカウント指定
 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。

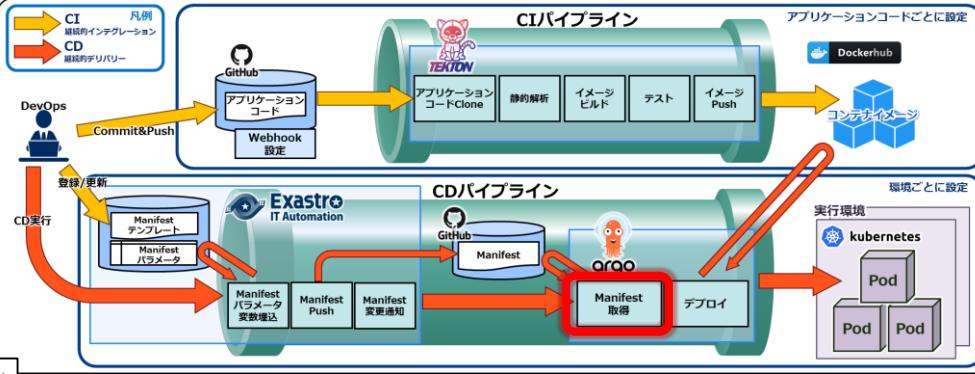
環境ごとに入力

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

入力完了後、決定を押下します



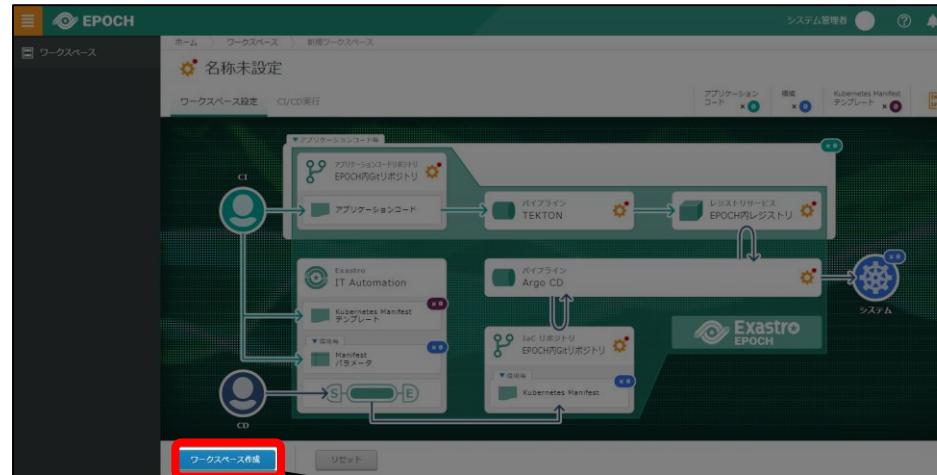
※赤枠部分の設定値を指定しています

項目	入力・選択内容	説明
GitリポジトリURL	https://github.com/[GitHubのアカウント名]/[各環境のリポジトリ].git	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

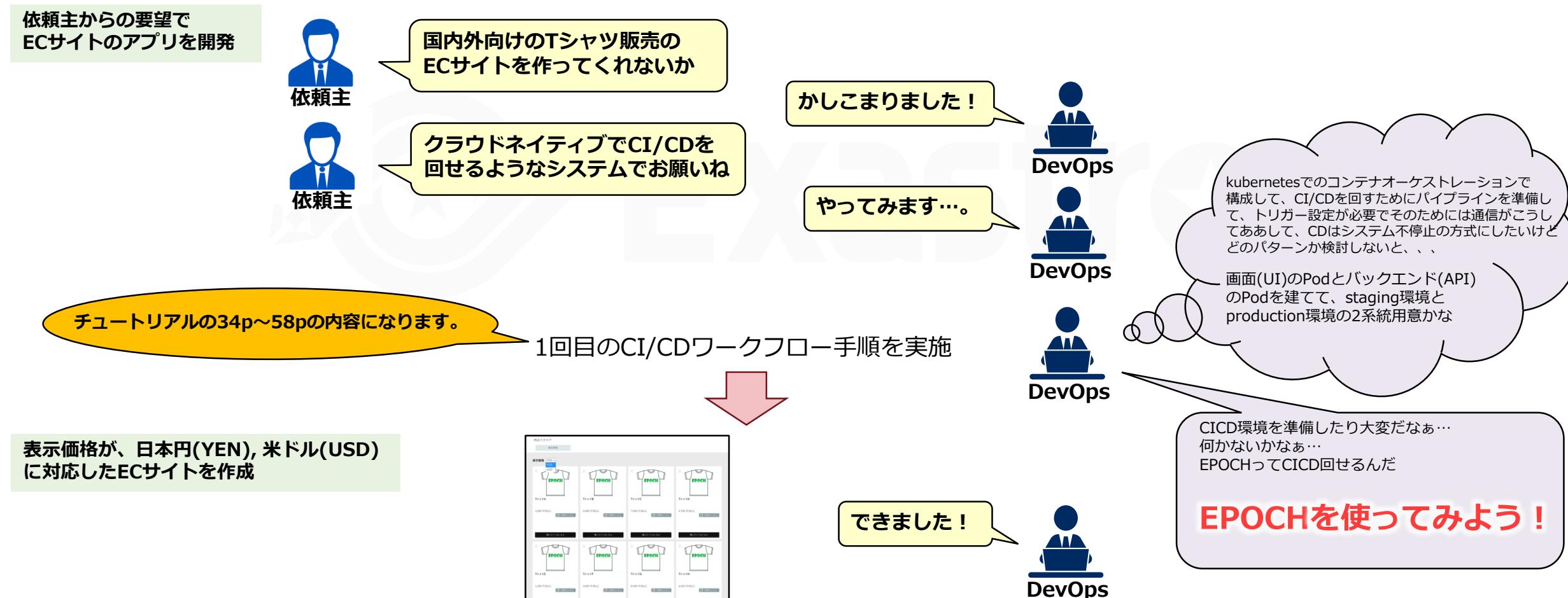
CI/CDワークフローを体験してみましょう。



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



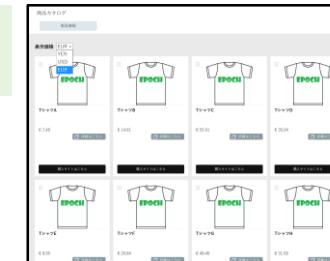
この前のECサイトはとても好評
だったよ



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♪



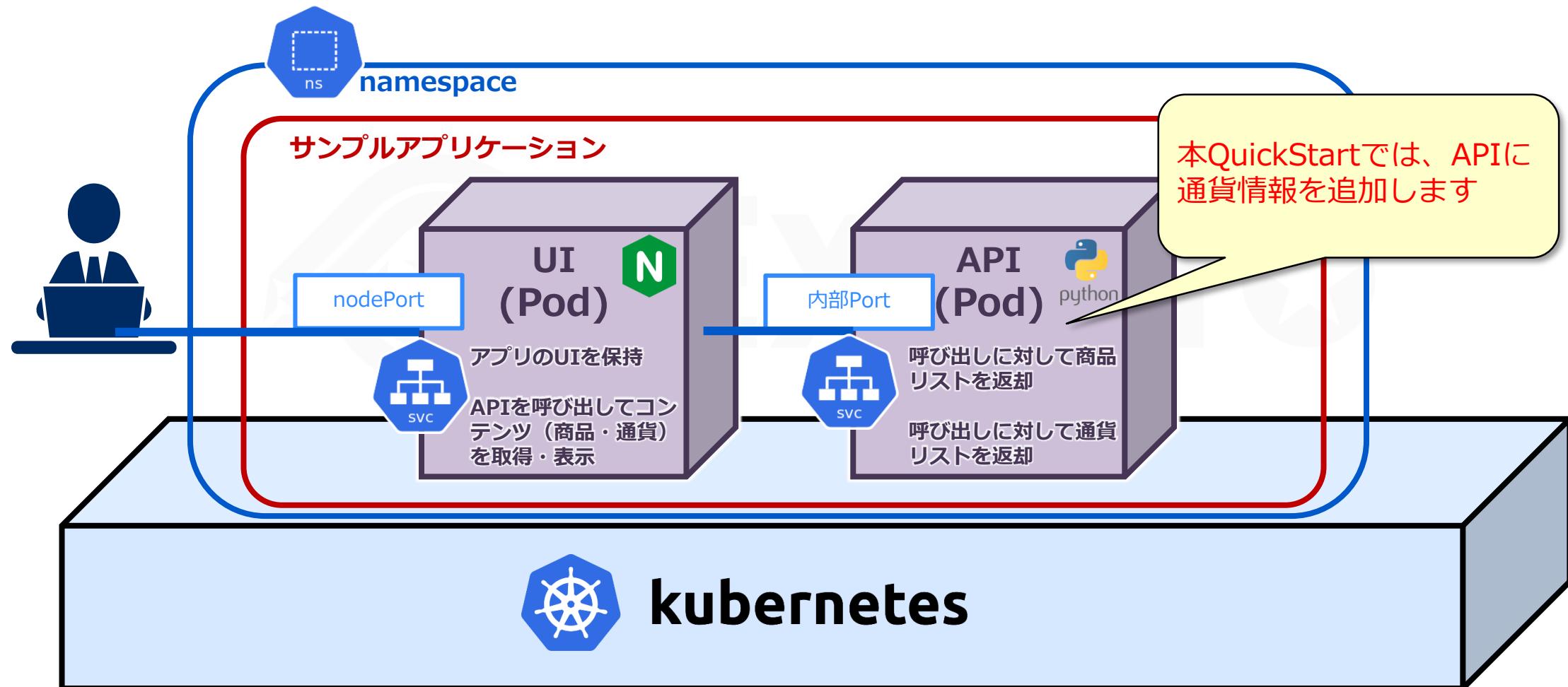
できました！



4.2 サンプルアプリの構成

サンプルアプリの構成

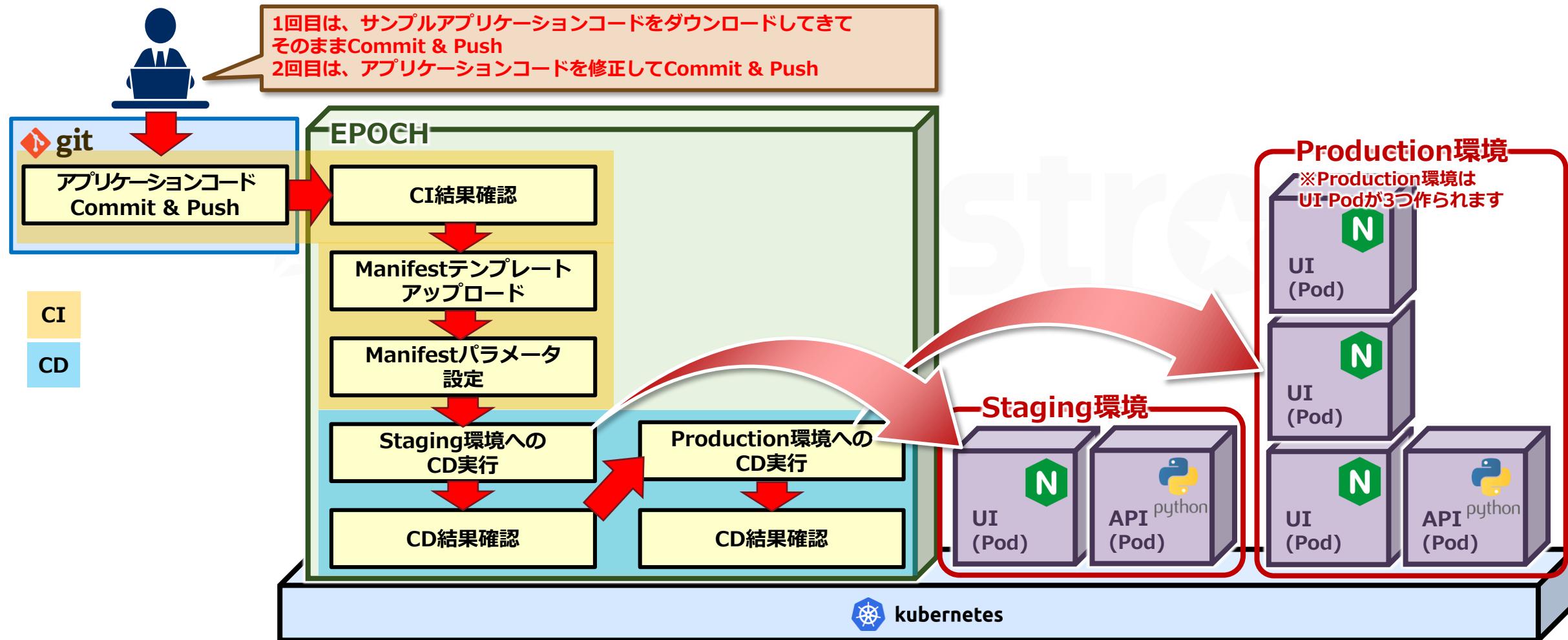
サンプルアプリケーションは、UIとAPIの2つで構成されております。



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

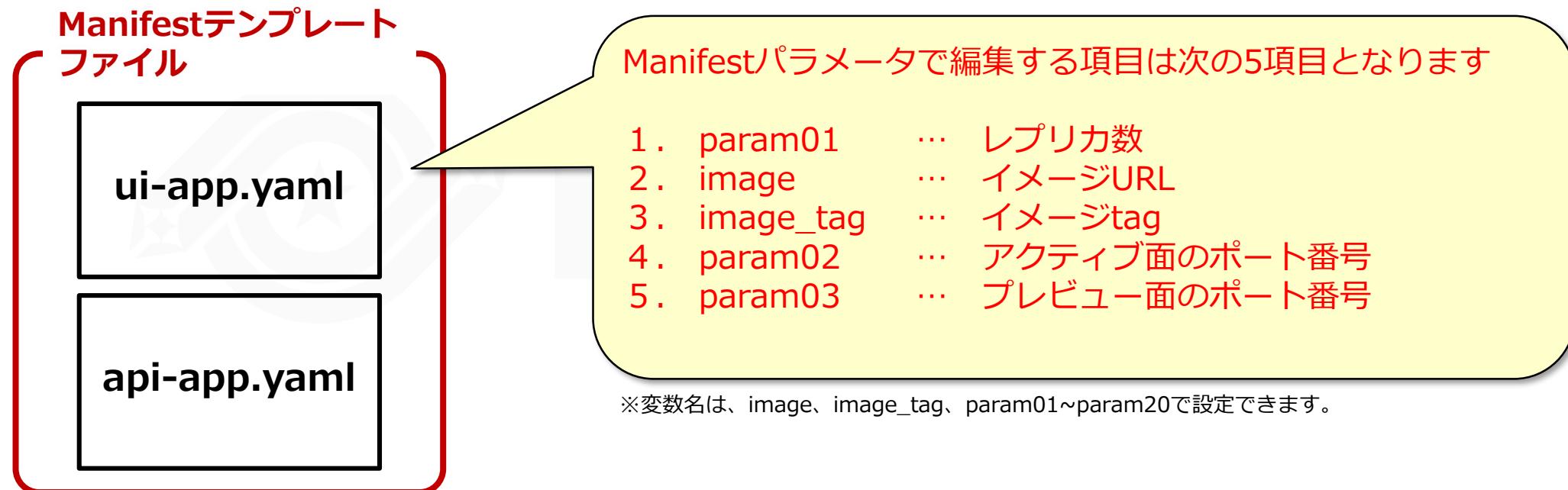
本説明では、サンプルアプリケーションをStaging環境、Production環境へDeploy、その後アプリケーションコードの修正を行い、Staging環境、Production環境へのDeployする手順を説明していきます。



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！

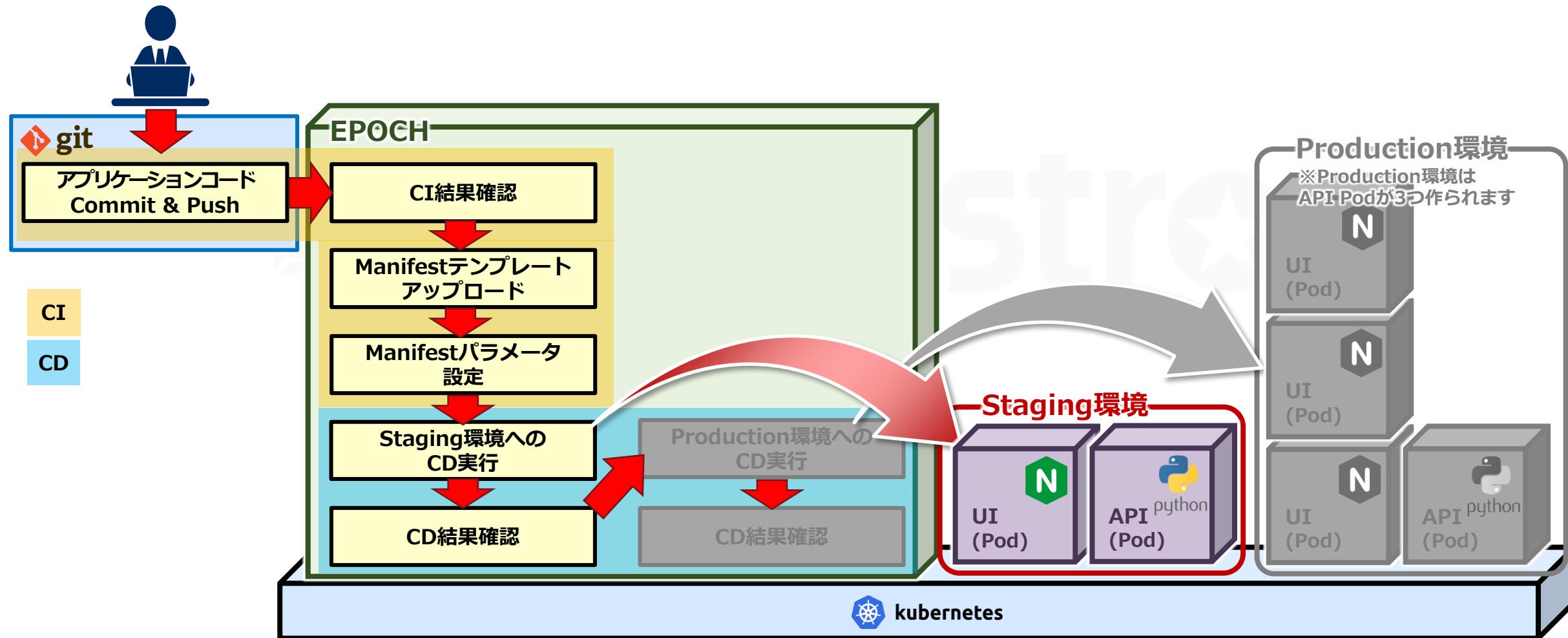
1回目のCI/CDワークフロー手順



4.5 1回目のCI/CDワークフロー手順(1/25)

Staging環境へのDeploy

1回目のCI/CDワークフローとして、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(2/25)

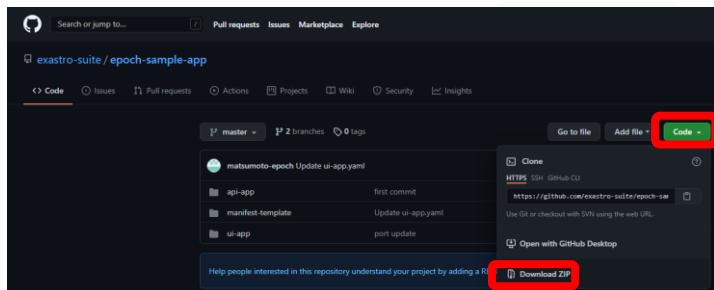
アプリケーションコード Commit & Push

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。

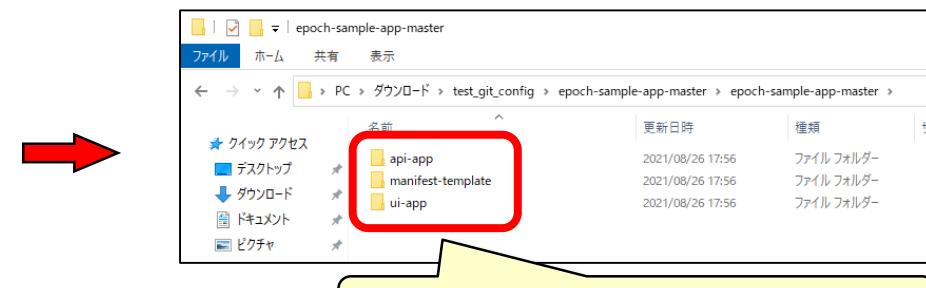
- ブラウザにてサンプルアプリケーションのURLを表示します。

<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「Code」を押下して、「Download ZIP」からコードを取得します。



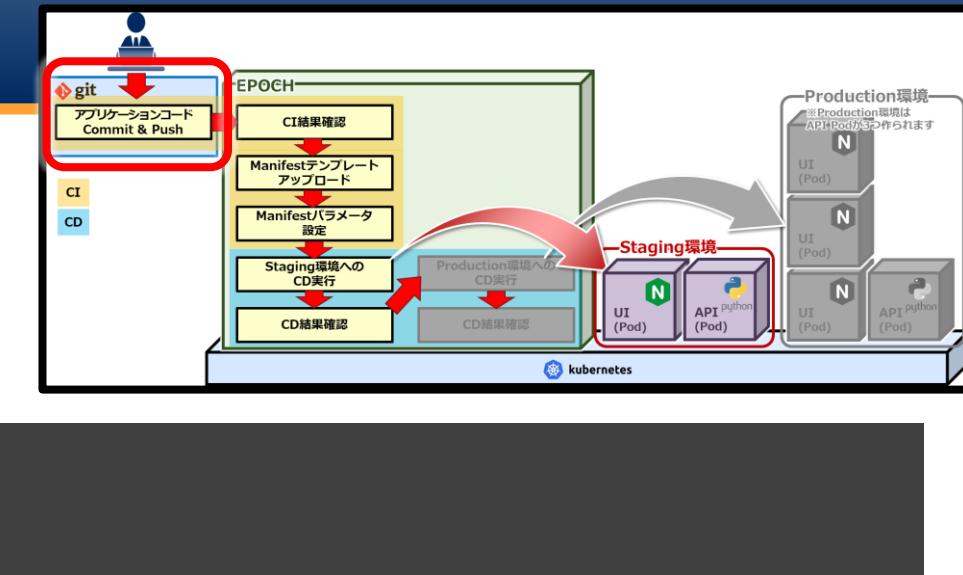
- ダウンロードしたZIPファイルを展開し、cloneしたアプリケーションコード用リポジトリにコピーします。



4.5 1回目のCI/CDワークフロー手順(3/25)

- 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。

```
> git add .  
> git commit -m "first build"  
> git push origin main
```



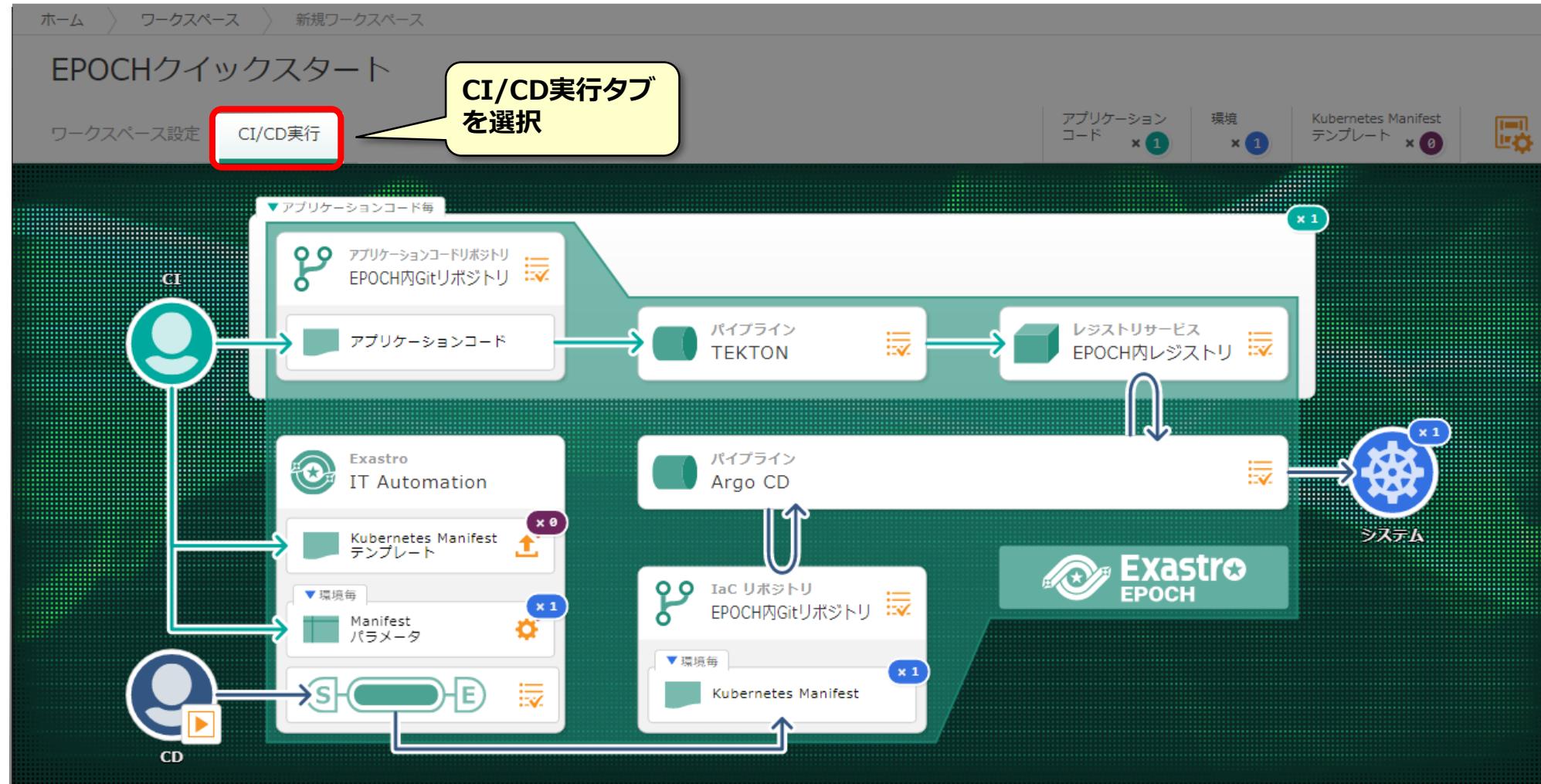
※ git push時に認証情報を求められた場合は、自身のGitHubアカウント情報を入力してください。

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(4/25)

CI/CD実行画面の表示

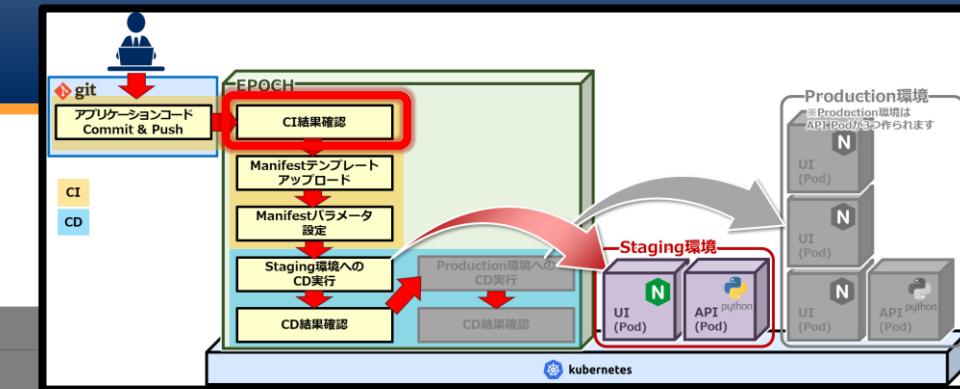
ワークスペース画面のCI/CD実行タブを選択し、CI/CD実行画面を表示します。



4.5 1回目のCI/CDワークフロー手順(5/25)

CI結果確認

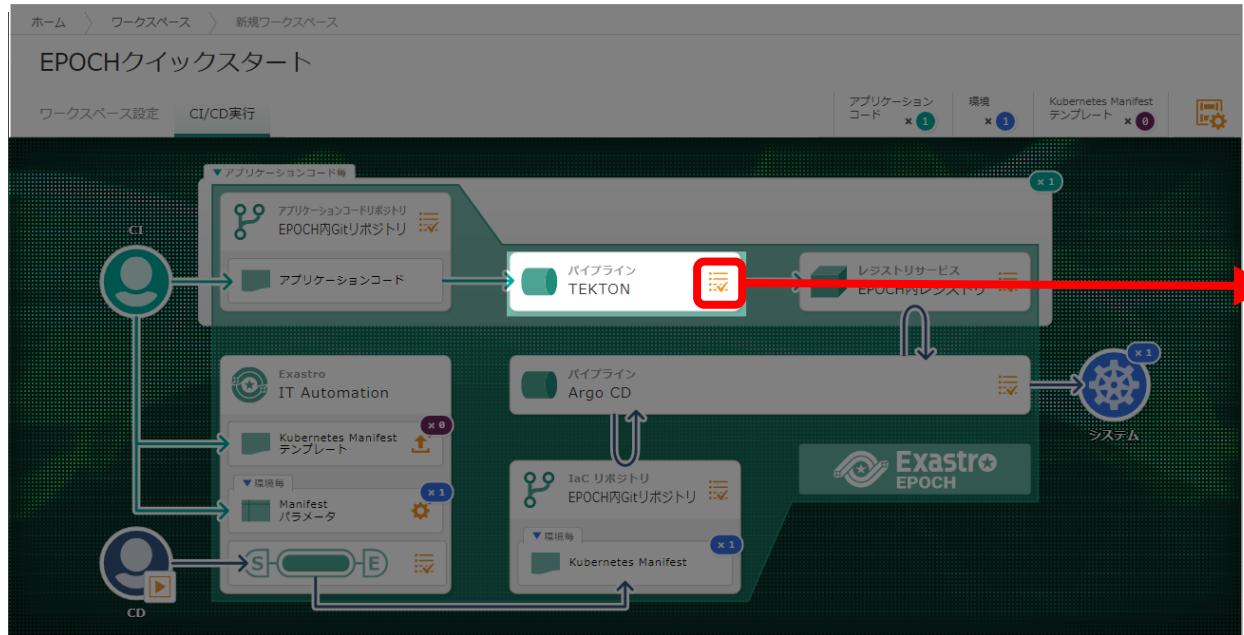
- アプリケーションコードのビルド結果を確認します。



4.5 1回目のCI/CDワークフロー手順(6/25)

■ 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。

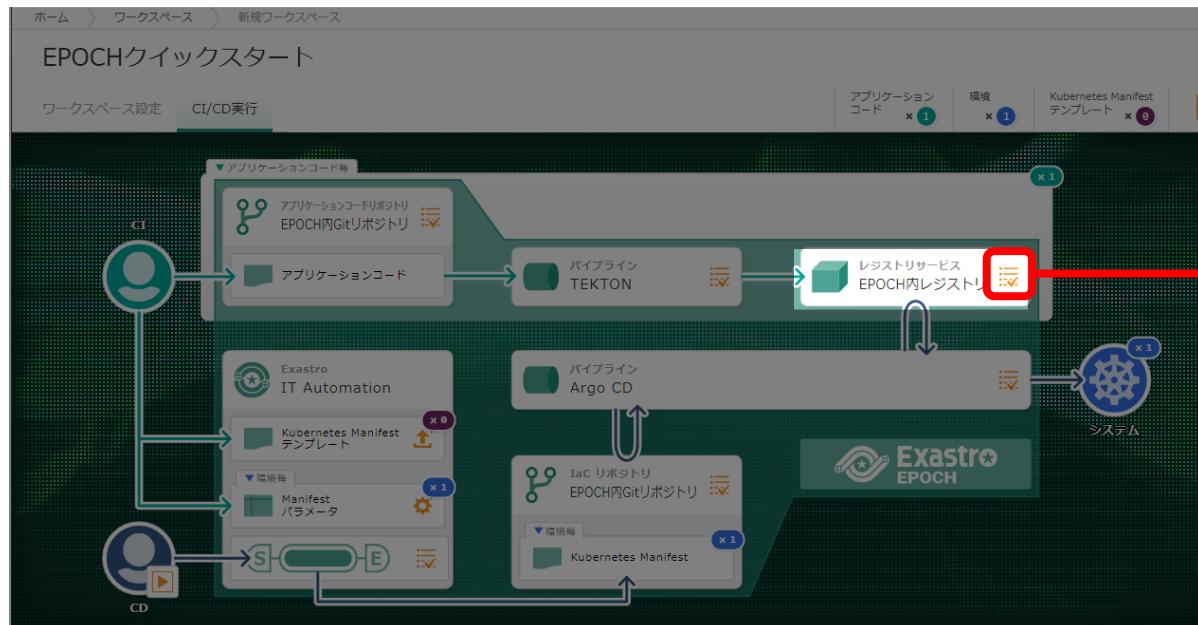


The screenshot shows the 'Tekton Dashboard' with the 'PipelineRuns' tab selected. The left sidebar includes 'Tekton resources', 'Pipelines', 'PipelineRuns' (selected), 'PipelineResources', 'Tasks', 'ClusterTasks', 'TaskRuns', 'Conditions', 'EventListeners', 'TriggerBindings', 'ClusterTriggerBindings', 'TriggerTemplates', 'Import resources', and 'About'. The main area is titled 'PipelineRuns' with a search bar 'Input a label filter of the format labelKey:labelValue'. Below it is a table with columns: Status, Name, Pipeline, Namespace, Created, and Duration. One entry is visible: 'build-and-push-pipeline-r...' under 'Name', 'pipeline-build-and-push' under 'Pipeline', 'epoch-tekton-pipelines' under 'Namespace', '6 minutes ago' under 'Created', and '3 minutes 32 seconds' under 'Duration'. A yellow callout bubble points to this entry 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 yellow callout bubble points to the 'PipelineRuns' tab with the text: '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(7/25)

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 Docker Hub page for the repository `/epoch-sample-api` shows two tags:

- TAG** master.20210701-171058 (highlighted with a red box)
- TAG** master.20210701-134114 (highlighted with a red box)

A yellow callout box contains the text: "Dockerhubの画面でepoch-sample-apiの Tagを確認します" and "※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".

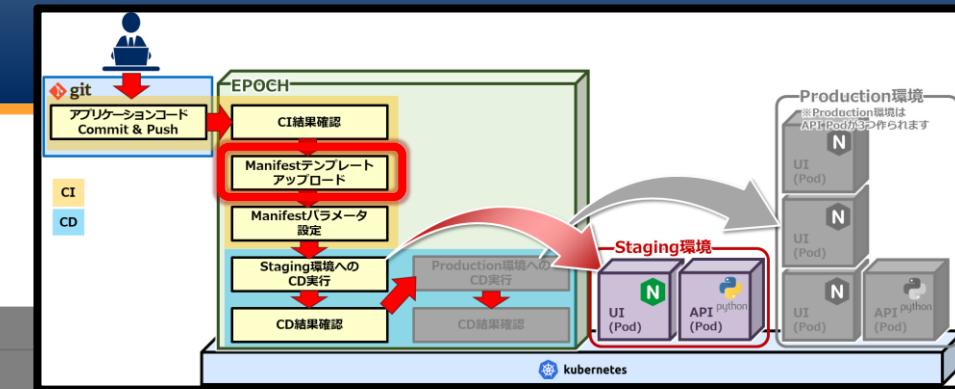
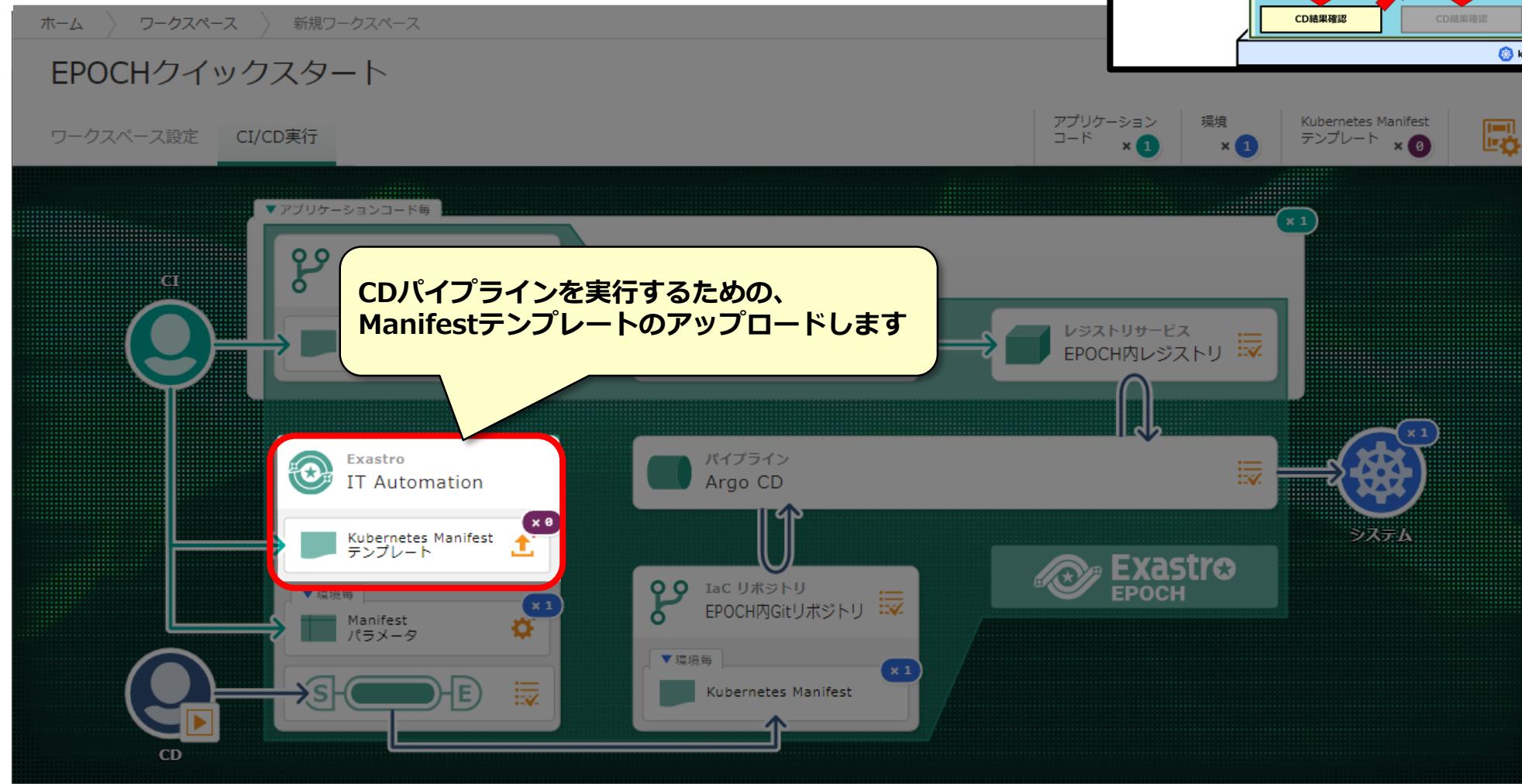
ここで確認したイメージのTagは、
次の手順で、Manifestパラメータ(image_tag)に
手入力が必要になるため控えておいてください

※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(8/25)

Manifestテンプレートアップロード

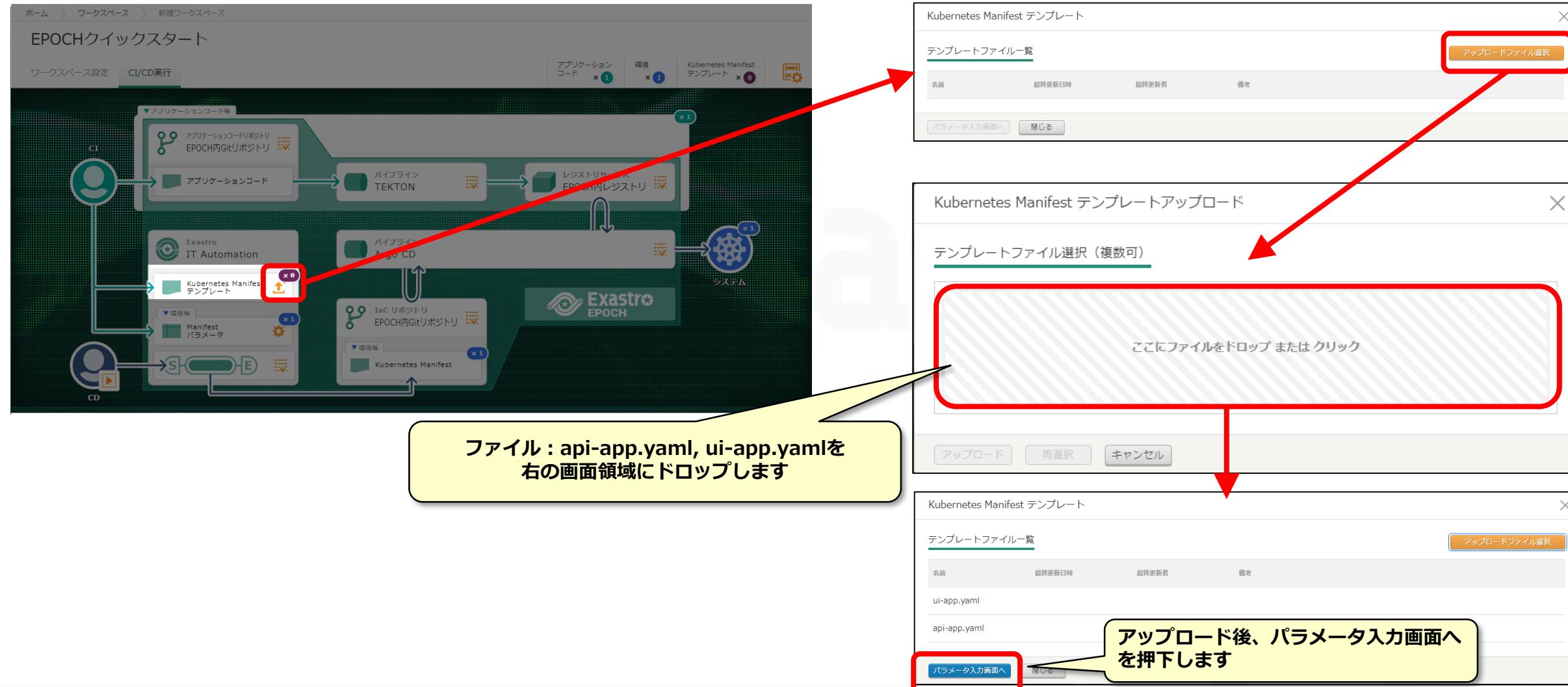
- ダウンロードしたManifestテンプレートをアップロードします。



4.5 1回目のCI/CDワークフロー手順(9/25)

Manifestテンプレートアップロード

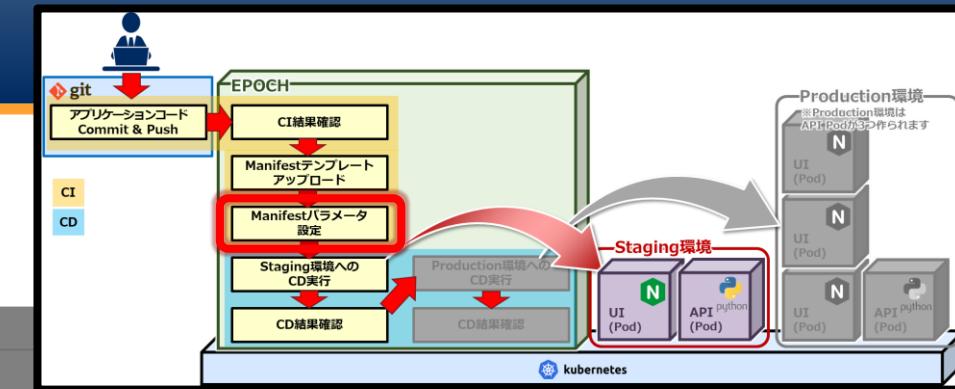
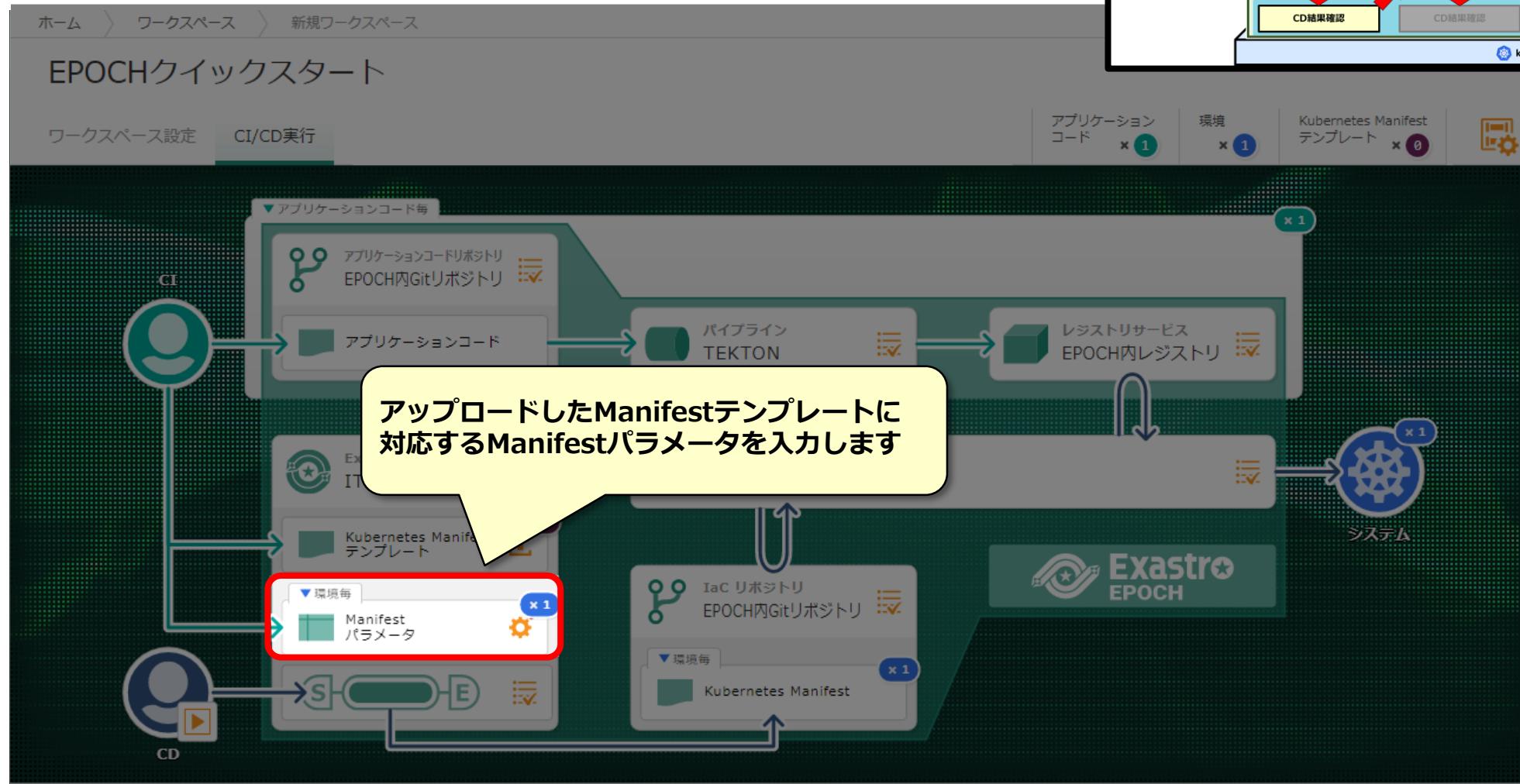
- Manifestテンプレートファイルの準備でダウンロードしたManifestテンプレートファイルをアップロードします。



4.5 1回目のCI/CDワークフロー手順(10/25)

Manifestパラメータ

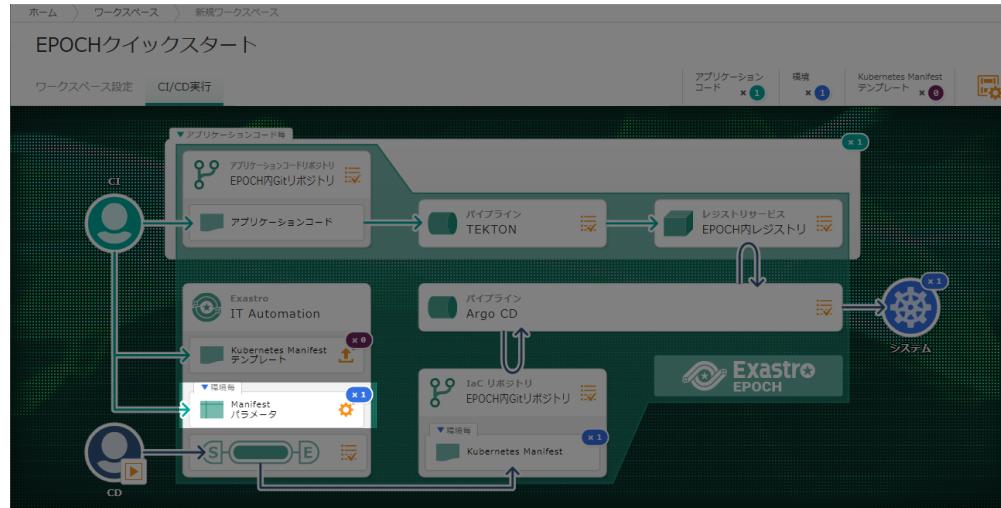
- Deployに必要なManifestパラメータを入力します。



4.5 1回目のCI/CDワークフロー手順(11/25)

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

項目	入力内容(staging)	入力内容(production)	説明
<code>{{ param01 }}</code>	1	3	レプリカ数
<code>{{ image }}</code>	exastro/epochsampleappui	exastro/epochsampleappui	コンテナイマージ
<code>{{ image_tag }}</code>	master.20210708183910	master.20210708183910	コンテナイマージのタグ
<code>{{ param02 }}</code>	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
<code>{{ param03 }}</code>	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

Manifest パラメータ

ui-app.yaml

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

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

決定 キャンセル

ui-app.yamlを選択して入力内容を入力します
※ファイルをアップロードする順番によっては、タブの右側にui-app.yamlが表示されている場合もあります

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

※ui-app.yamlのimage_tagは事前にビルドしたコンテナイマージのものを使用しています。
※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(12/25)

- タブを切り替えてapi-app.yamlにも入力します。



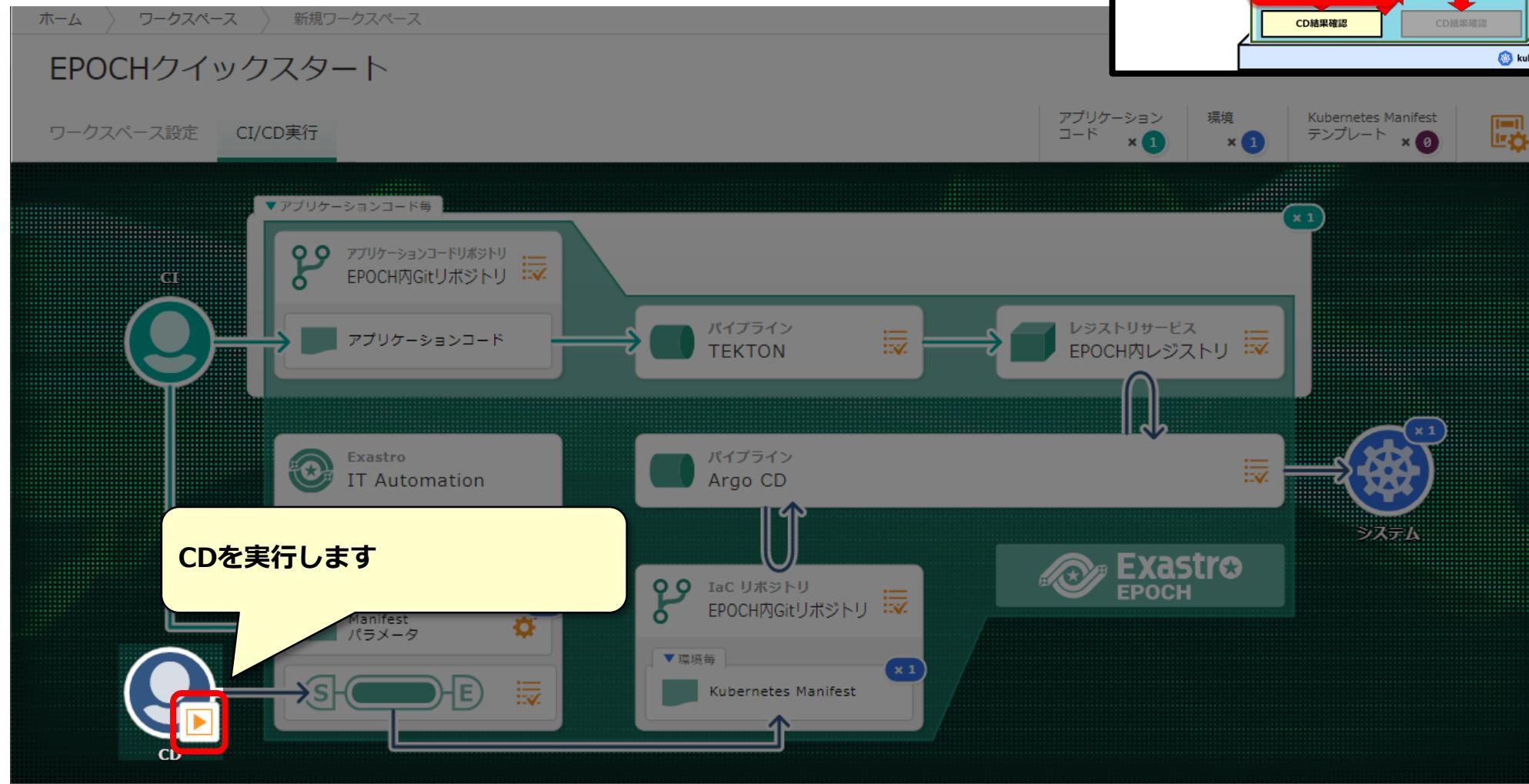
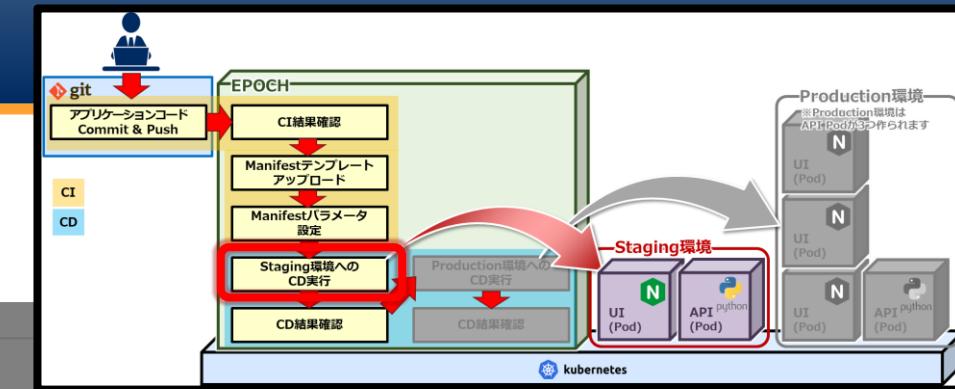
api-app.yaml

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(13/25)

Staging環境へのDeploy実行

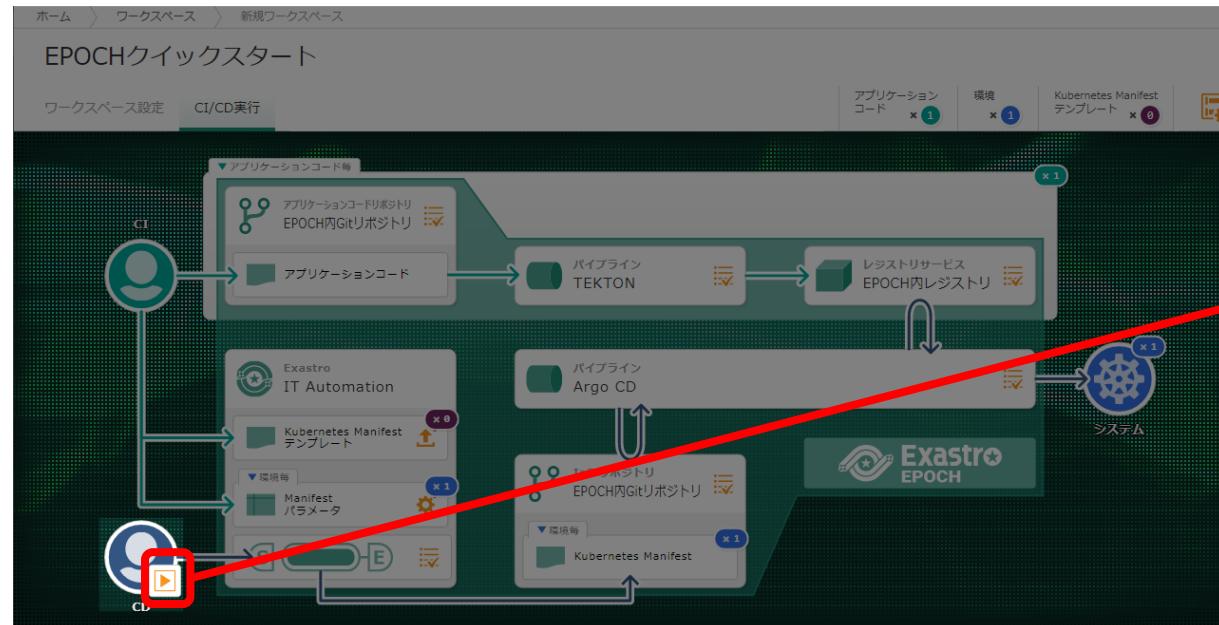
- CD実行で、Staging環境へDeployを実行します。



4.5 1回目のCI/CDワークフロー手順(14/25)

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(15/25)

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(16/25)

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. It shows the Exastro IT Automation interface with various components like EPOCH Quick Start, Application Code, Environment, Pipelines (TEKTON, Argo CD), and Kubernetes Manifest. A red box highlights the 'Conductor' icon in the bottom navigation bar.

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

A yellow callout points to the 'Conductor' menu item in the left sidebar, which is then selected to view the 'Conductor Job List'.

[2] フィルタをクリック

A yellow callout points to the 'Filter' button in the 'Conductor Job List' interface, which is used to apply filters.

[3] 【CD実行】の詳細をクリック

A yellow callout points to the 'CD execution' details section in the 'Conductor Job List' interface, specifically highlighting the 'Details' button.

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

A yellow callout provides login credentials for the Exastro IT Automation interface.

すべて[DONE]と表示されていれば完了です

A yellow callout indicates that the process is completed when all tasks are marked as [DONE].

4.5 1回目のCI/CDワークフロー手順(17/25)

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, there's a 'CI/CD実行' (CI/CD Execution) section with a flowchart showing 'Application Code' going through 'TEKTON' and 'Argo CD' to 'Kubernetes Manifest'. Below it is an 'Exastro IT Automation' section. In the center, there's a 'Let's get stuff deployed!' screen with a cartoon character. To the right, there's a 'argo' sign-in form with fields for 'Username' and 'Password'. A yellow callout box says: 'Username: admin' and 'Password: iYSCKzx2wvxJnn4dCNwN でSIGN INします'. Another yellow callout box 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'. Further down, there's a 'staging' application details screen with a red box around the 'staging' project. A yellow callout box says: 'stagingを選択'. To the right of this is another 'staging' application details screen showing 'Sync OK'. A yellow callout box 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'. 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'.

stagingを選択

Username: admin
Password: iYSCKzx2wvxJnn4dCNwN でSIGN INします

※「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

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 OK

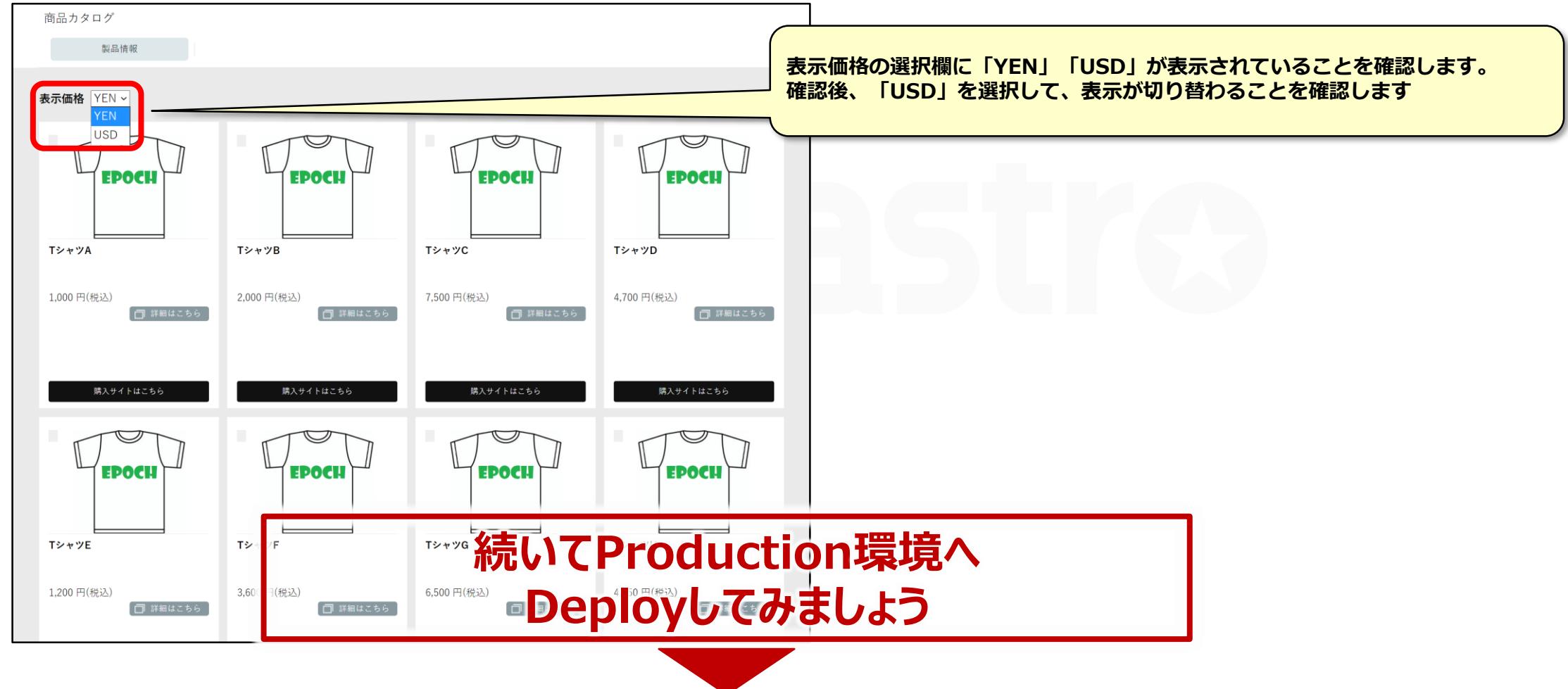
Deployされたサンプルアプリケーションを確認してみましょう

4.5 1回目のCI/CDワークフロー手順(18/25)

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

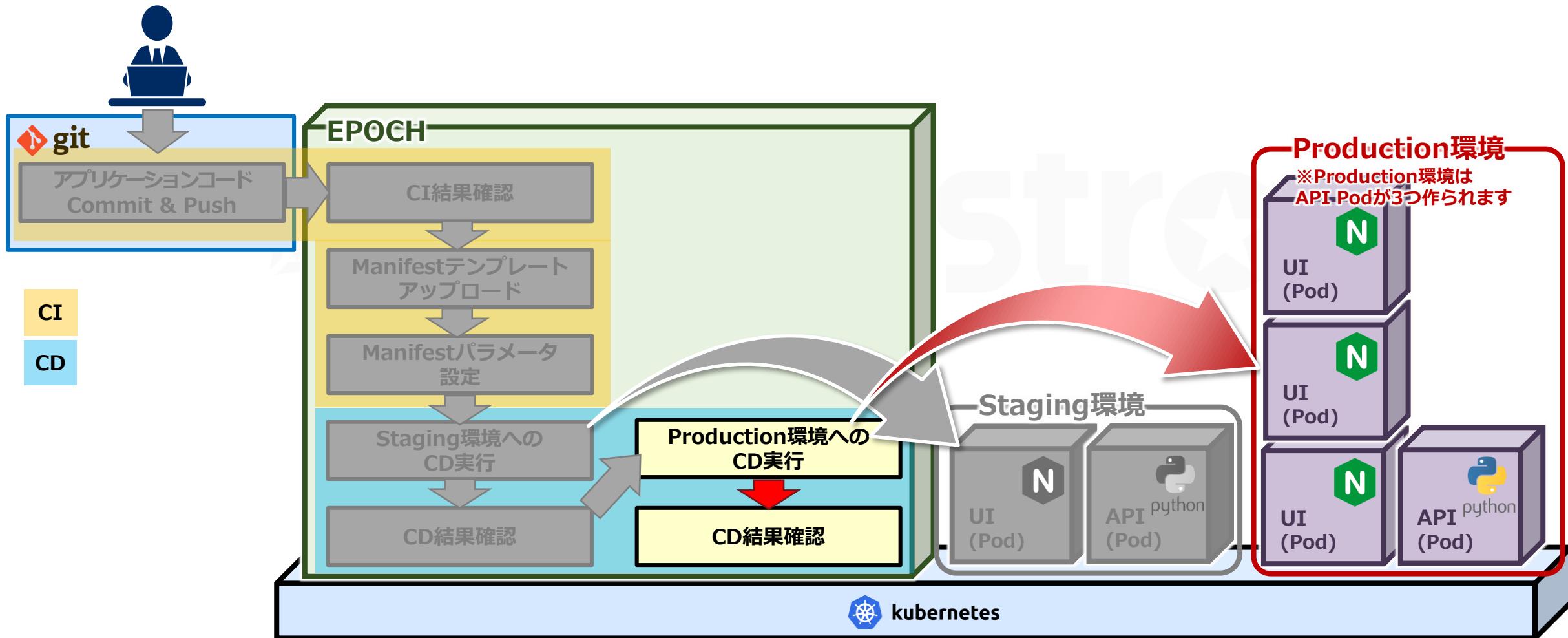
[http://\[Kubernetes masterノードのIPアドレスまたはホスト名\]:31001/front-end.html](http://[Kubernetes masterノードのIPアドレスまたはホスト名]:31001/front-end.html)



4.5 1回目のCI/CDワークフロー手順(19/25)

Production環境へのDeploy

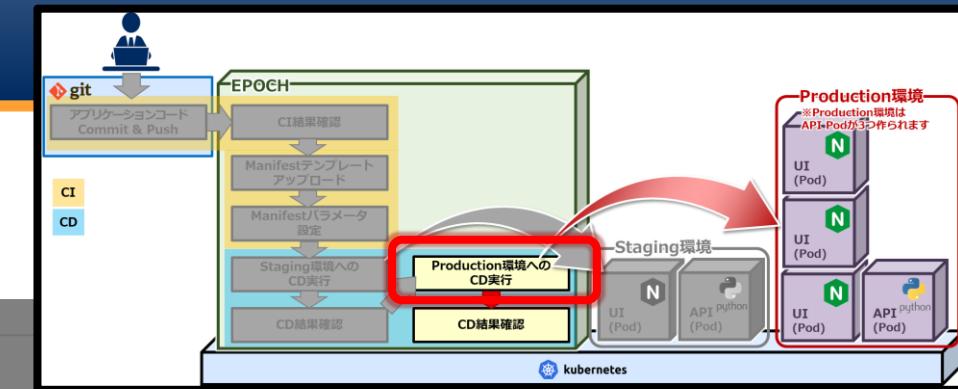
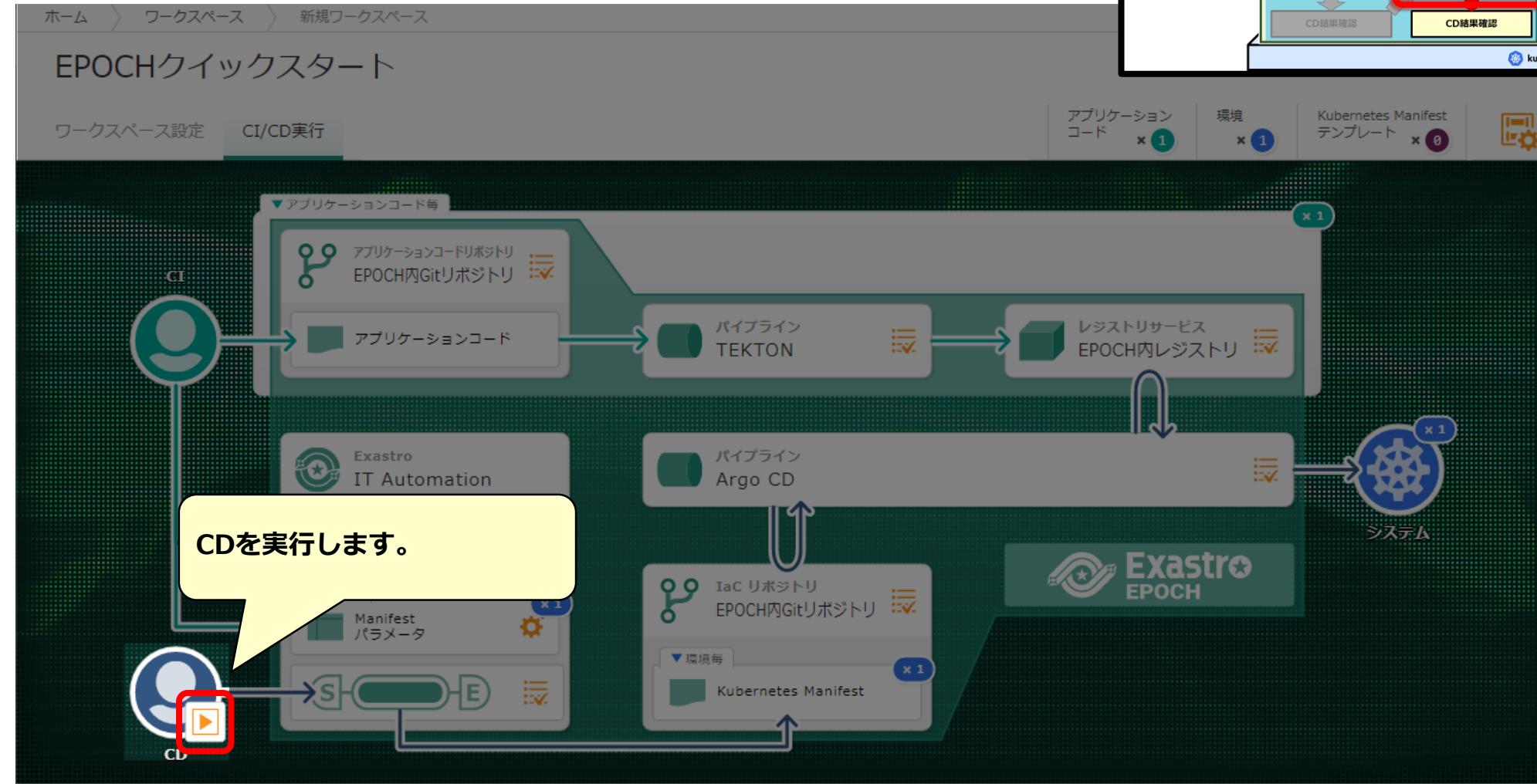
Staging環境へDeploy後、以下の内容でProduction環境へDeployします。



4.5 1回目のCI/CDワークフロー手順(20/25)

Production環境へのDeploy実行

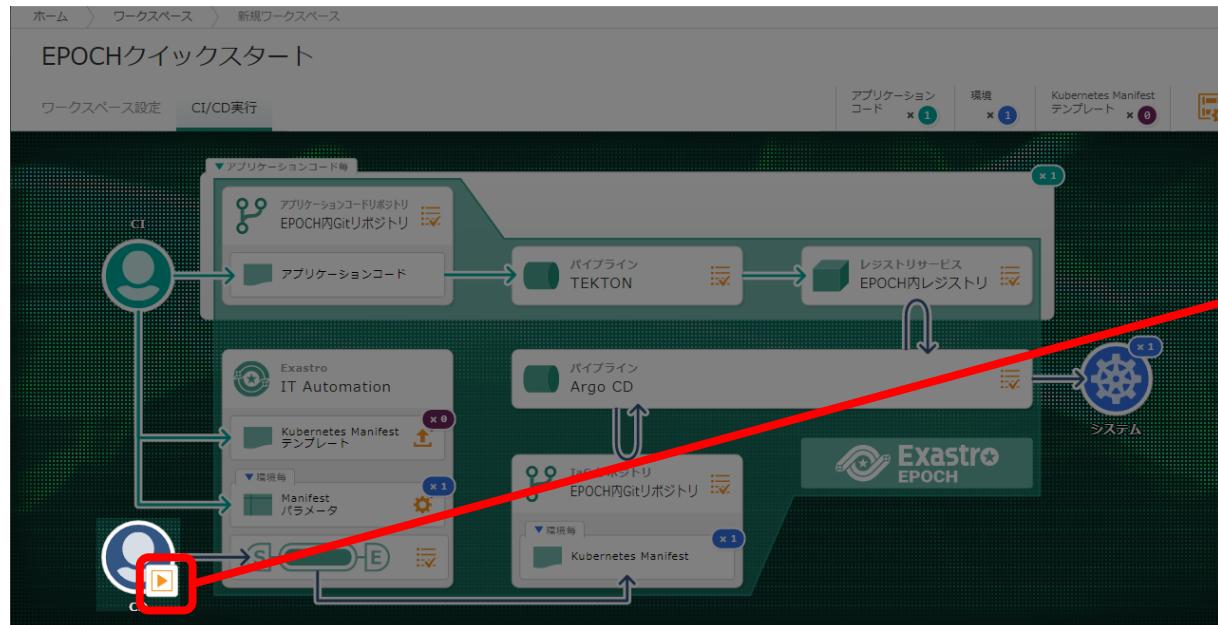
- CD実行で、Production環境へDeployを実行します。



4.5 1回目のCI/CDワークフロー手順(21/25)

Production環境のCD実行

- Deploy先の環境を選択して実行します。



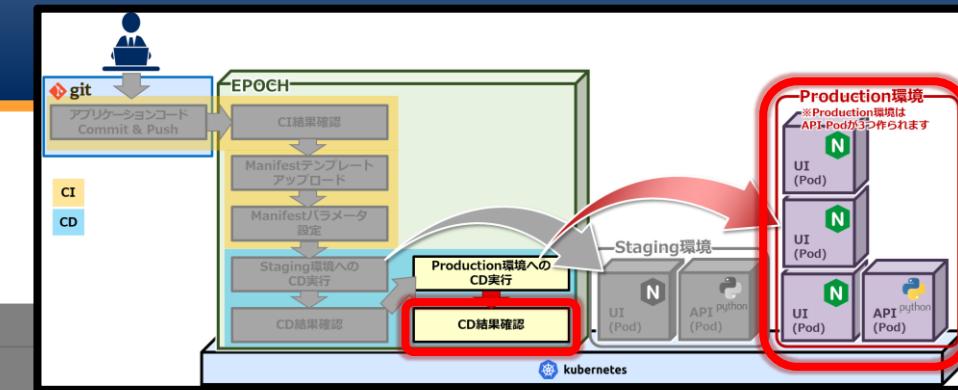
The screenshot shows the 'CD Execution Specification' dialog box. It includes fields for 'Execution Conditions' (即実行 selected), 'Environment' (production selected), 'Manifest Parameters', and 'ArgoCD Pipeline'. A callout box points to the 'production' environment selection with the text 'productionを選択します'. Another callout box points to the 'Execute' button with the text '環境選択後、実行ボタンを押下します'.

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(22/25)

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(23/25)

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.

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

右側のConductor操作画面では、Conductor作業一覧が表示され、各作業のステータスが「DONE」として確認できます。メッセージに「すべて[DONE]と表示されていれば完了です」とあります。

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

4.5 1回目のCI/CDワークフロー手順(24/25)

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 pipeline flow. It starts with Application Code being pushed to a Git Repository, which triggers a Pipeline TEKTON. This pipeline then interacts with the EPOCH Registry and the Argo CD component. The Argo CD interface shows the deployment status for the 'production' environment, indicating it is healthy and Synced. A red box highlights the 'production' entry in the list. Red arrows point from the Argo CD interface to the 'Sync OK' status in the Applications screen and finally to the deployed application components in the Kubernetes cluster view.

productionを選択

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

Username: admin
Password: iYSCKzx2wvxJnn4dCNwN
でSIGN INします

※「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

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 OK】が表示されましたら完了です
※日付がCD実行後であることを確認してください

4.5 1回目のCI/CDワークフロー手順(25/25)

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)	詳細はこちら
TシャツB	2,000 円(税込)	詳細はこちら
TシャツC	7,500 円(税込)	詳細はこちら
TシャツD	4,700 円(税込)	詳細はこちら
TシャツE	1,200 円(税込)	詳細はこちら

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！

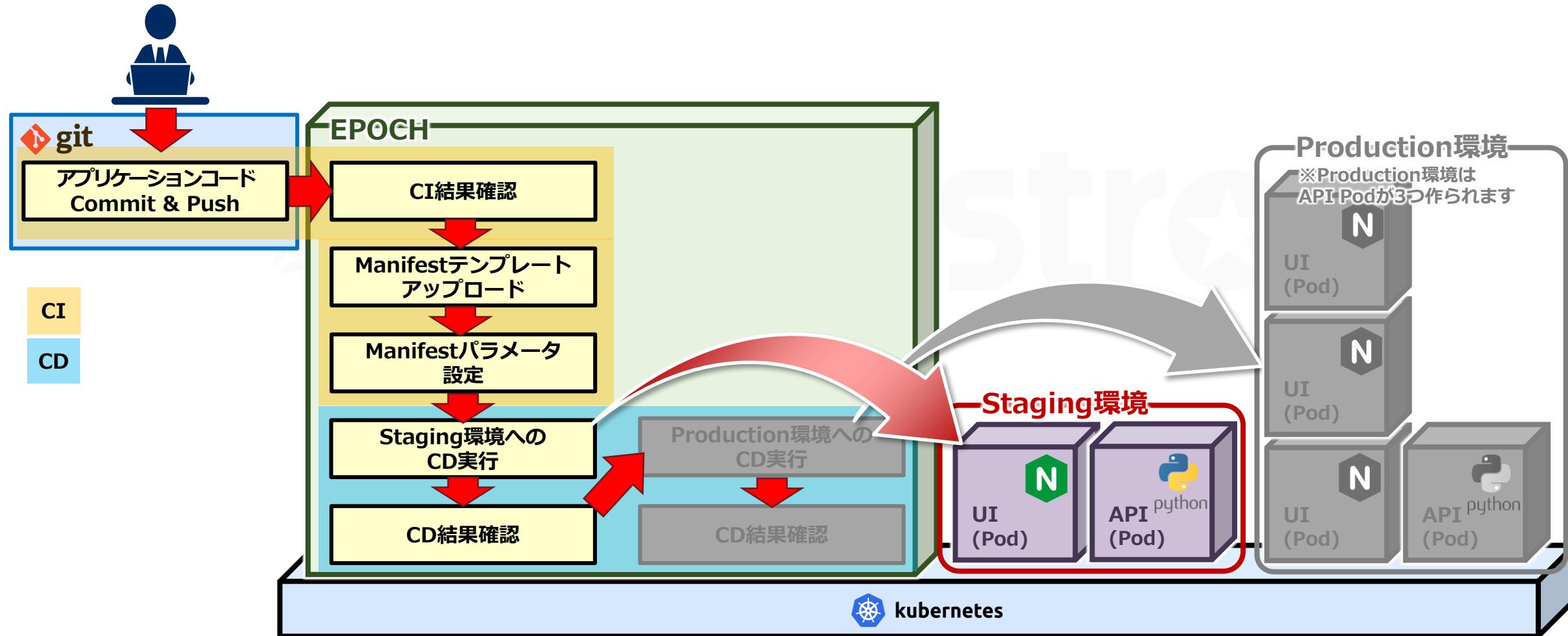
2回目のCI/CDワークフロー手順



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

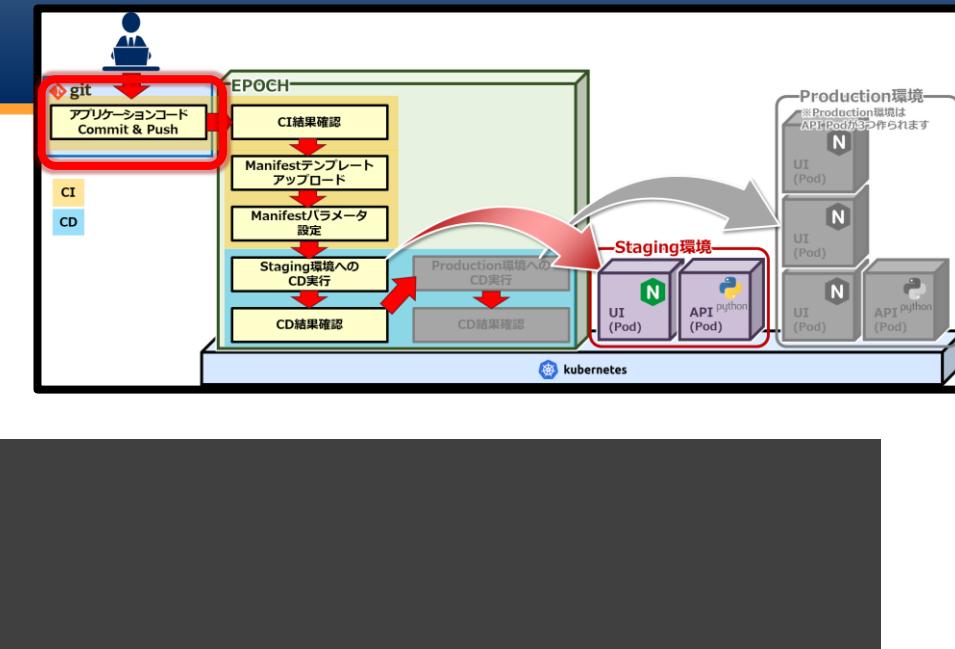
4.6 2回目のCI/CDワークフロー手順(3/21)

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 main
```

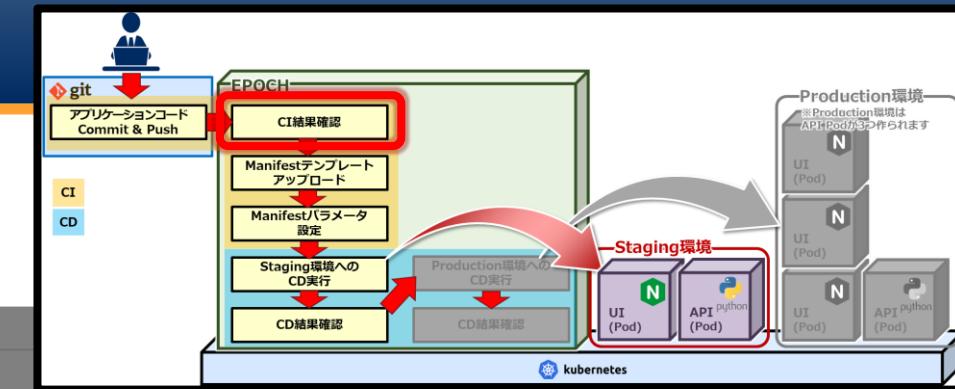


※ git push時に認証情報を求められた場合は、自身のGitHubアカウント情報を入力してください。

4.6 2回目のCI/CDワークフロー手順(4/21)

CI結果確認

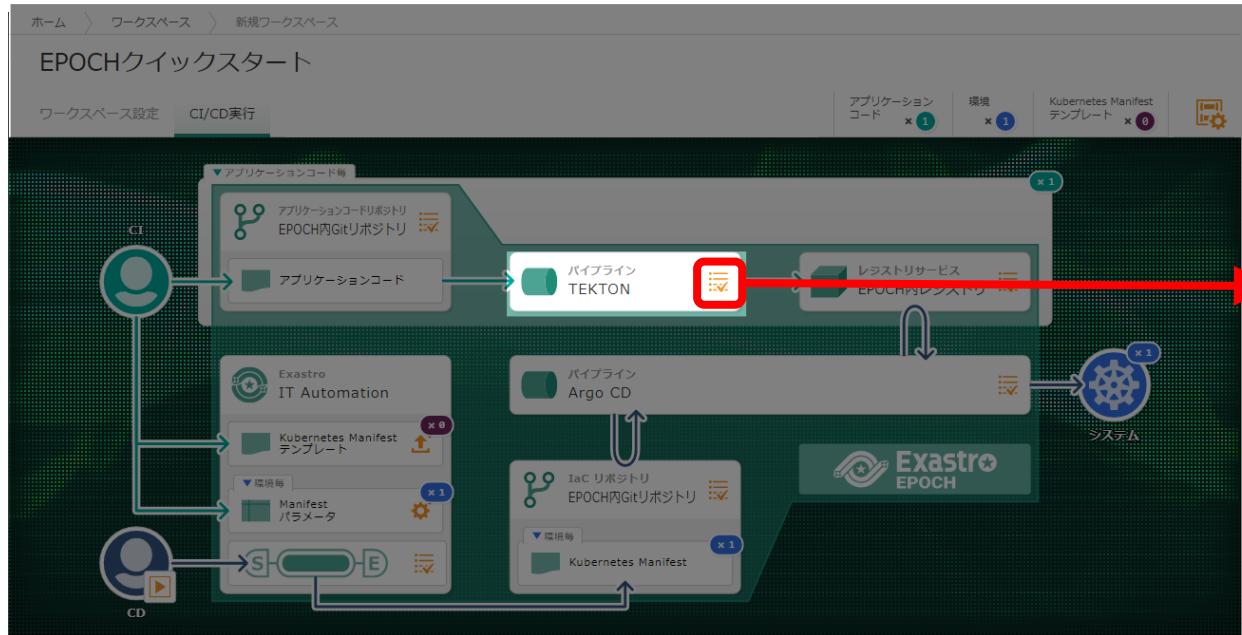
- アプリケーションコードのビルド結果を確認します。



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' list. A single entry is visible:

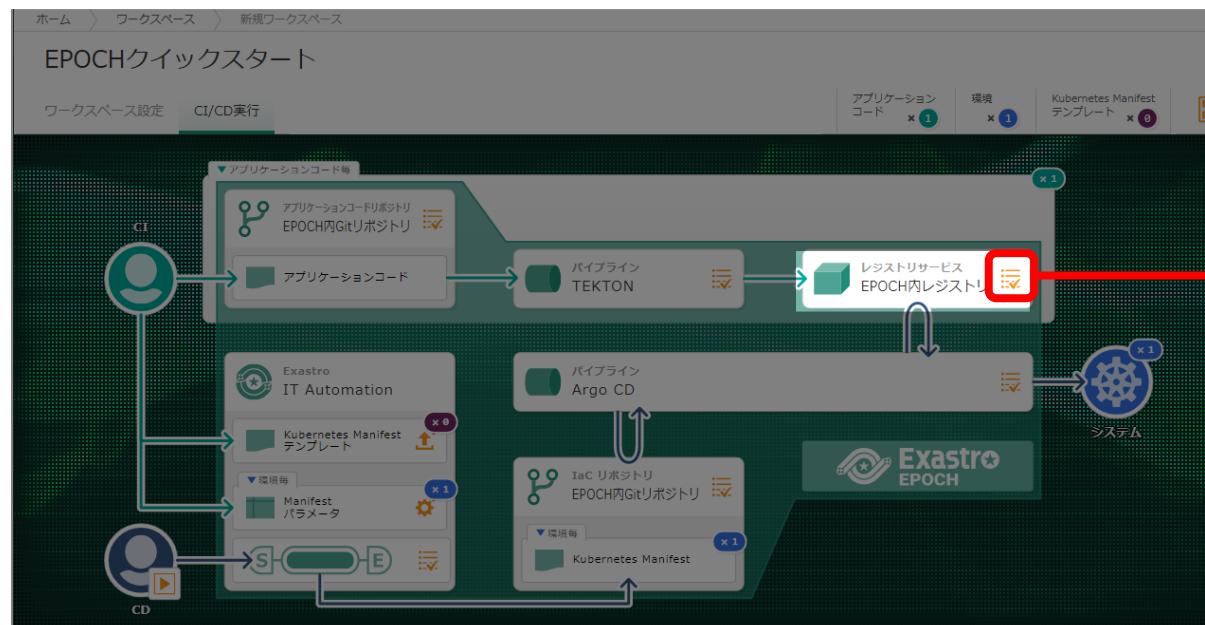
Status	Name	Pipeline	Namespace	Created	Duration
Success	build-and-push-pipeline-r...	pipeline-build-and-push	epoch-tekon-pipelines	6 minutes ago	3 minutes 32 seconds

A yellow callout 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' (If a checkmark is displayed in the Status column, it indicates a successful completion) points to the 'Status' column of the table. Another callout with the text '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 link will allow you to view more detailed CI pipeline information) points to the 'Name' column of the table.

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。



The screenshot shows the Docker Hub page for the repository 'epoch-sample-api'. It displays two tags:

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

Both tags were last pushed 7 days ago. A yellow callout box contains the following 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

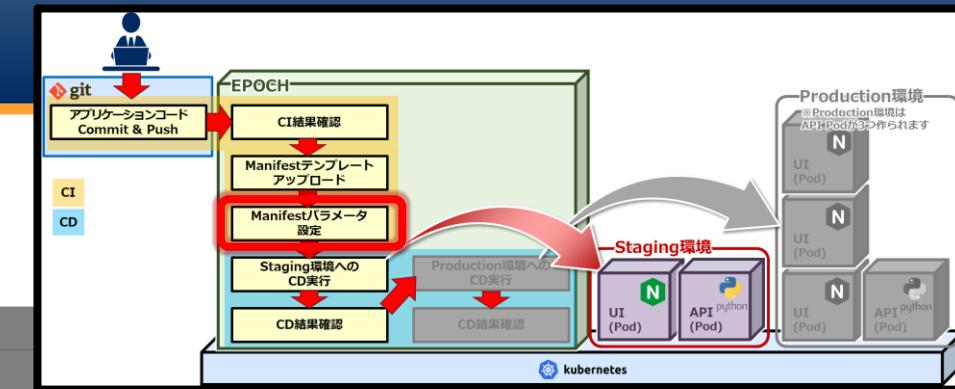
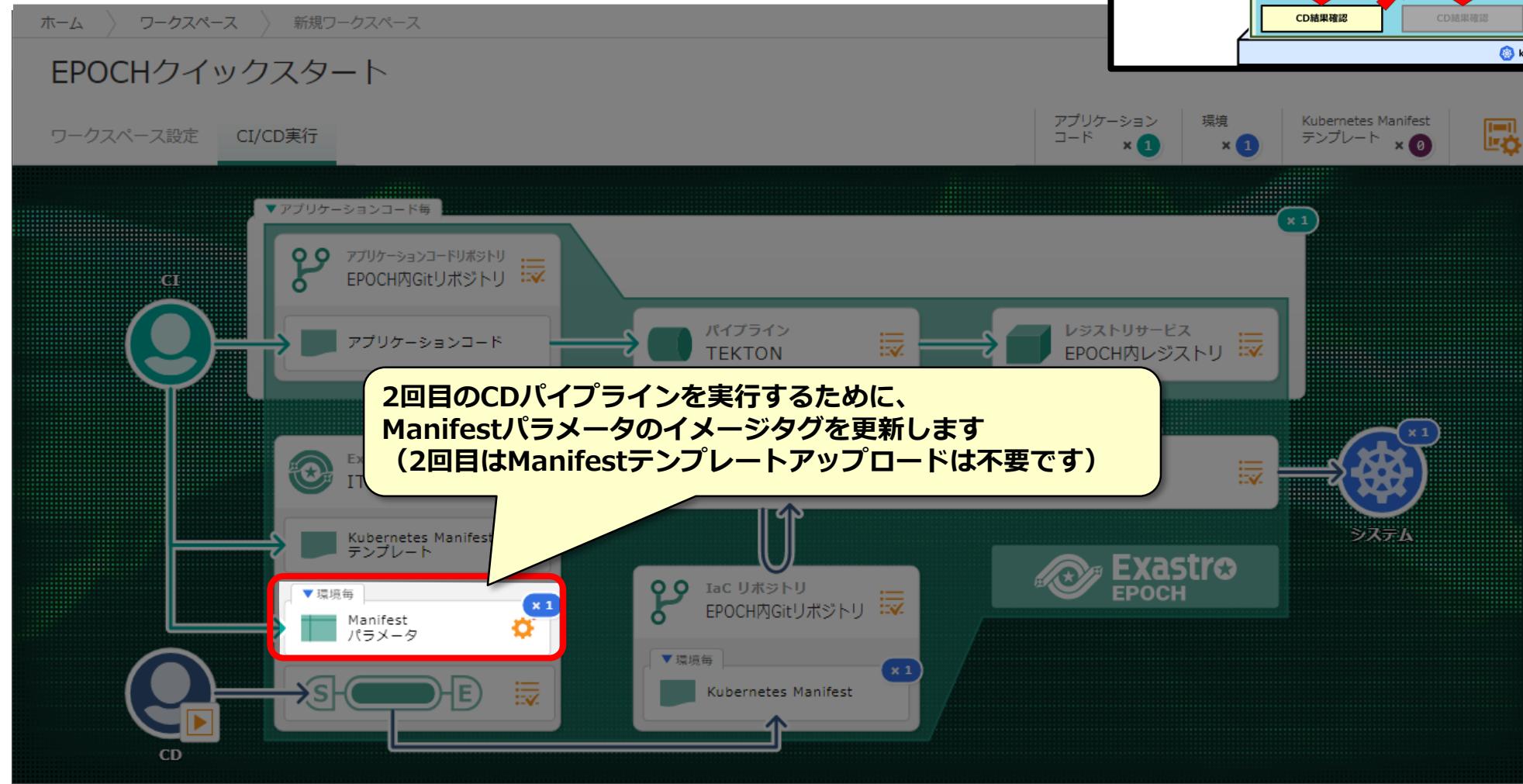
ここで確認したイメージのTagは、
次の手順で、Manifestパラメータ(image_tag)に
手入力が必要になるため控えておいてください

※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

4.6 2回目のCI/CDワークフロー手順(7/21)

Manifestパラメータ設定

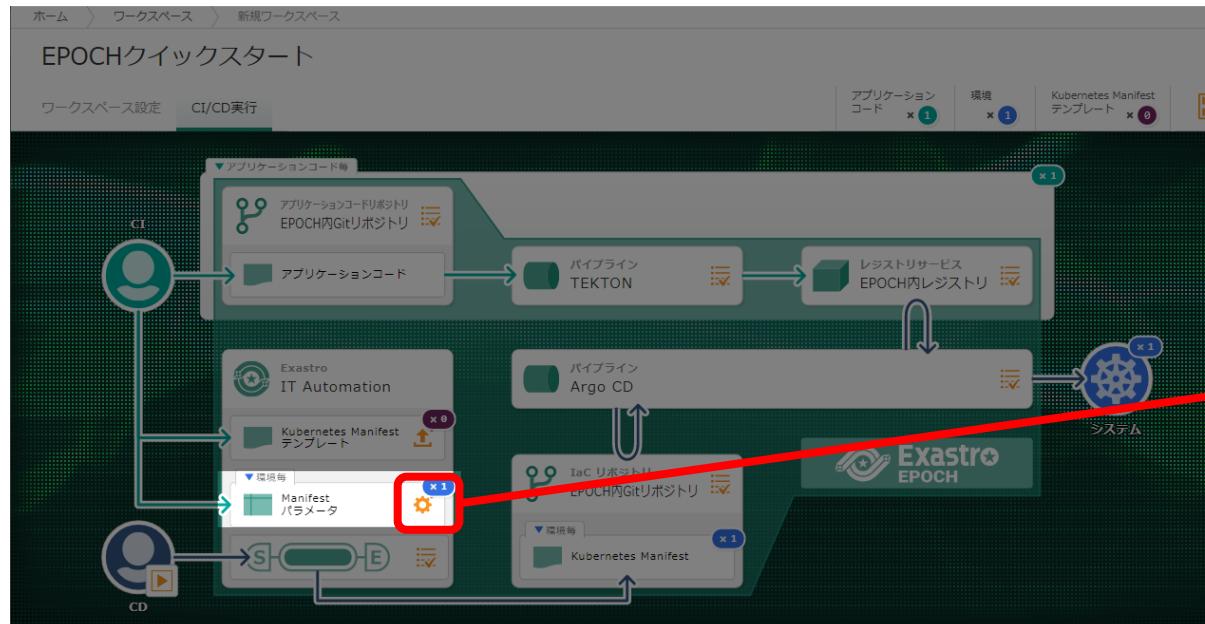
- Manifestパラメータの設定のイメージタグを更新します。



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。



Manifest パラメータ

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

api-app.yamlを選択

```
ui-app.yaml          api-app.yaml  
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: api-app  
spec:  
  containers:  
    - name: api-app  
      image: {{ image }} : {{ image_tag }}  
      ports:  
        - name: http  
          containerPort: 8000
```

staging, productionの image_tagを修正

決定 キャンセル

入力完了後、決定を押下します

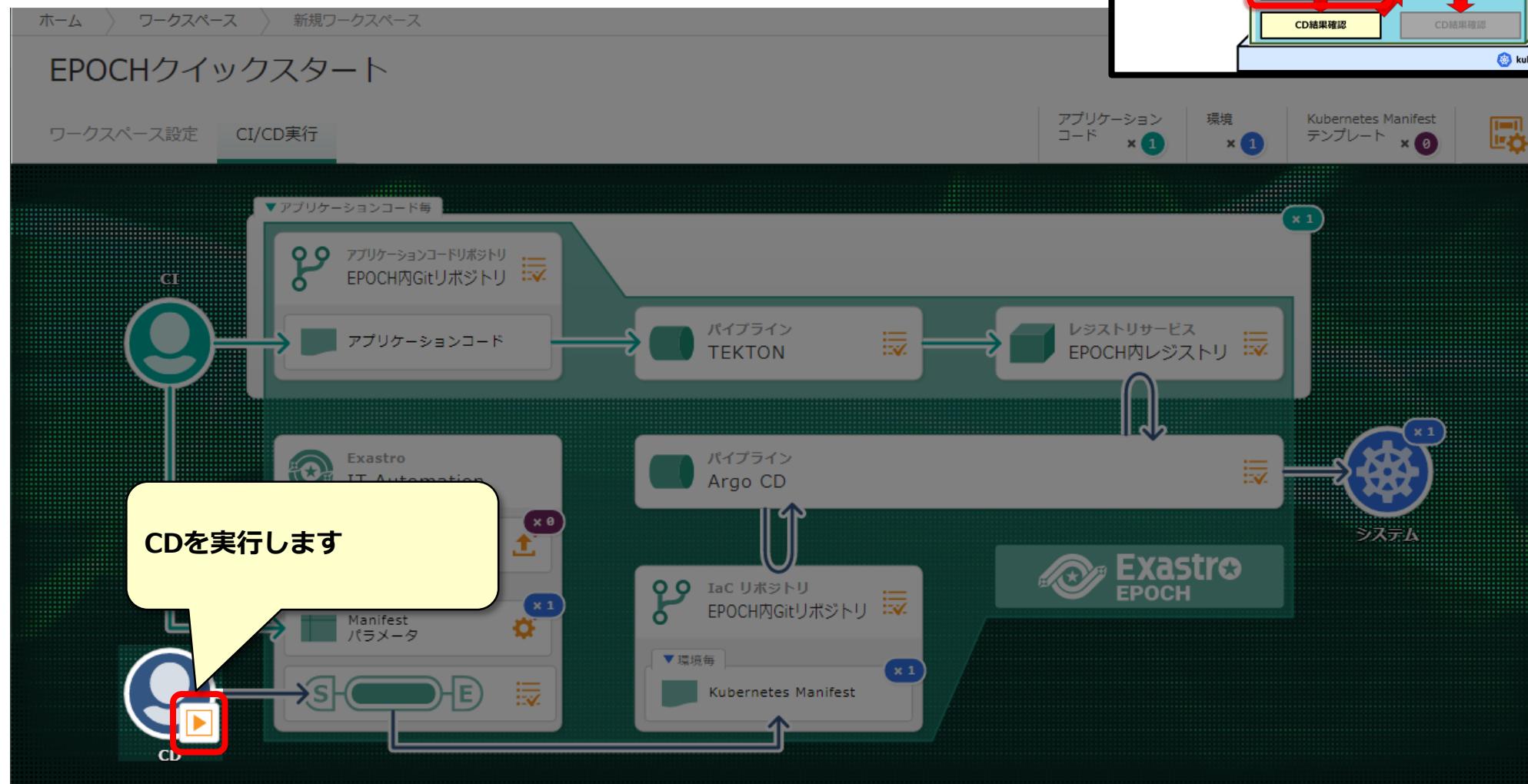
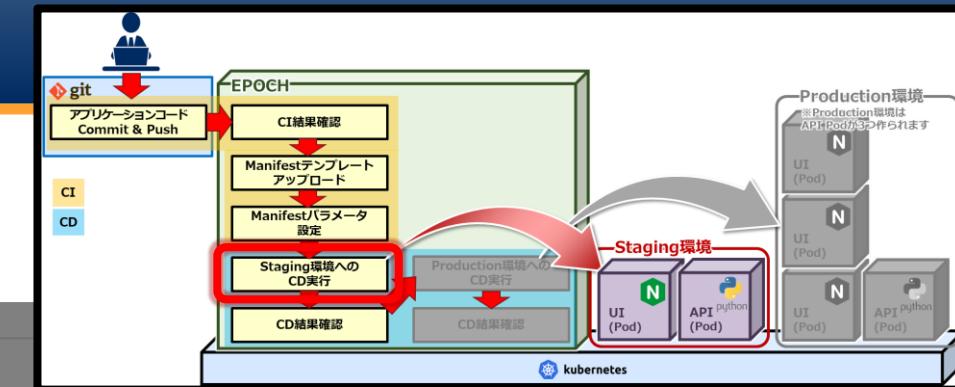
api-app.yaml

項目	入力内容(staging)	入力内容(production)	説明
{{ image_tag }}	[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

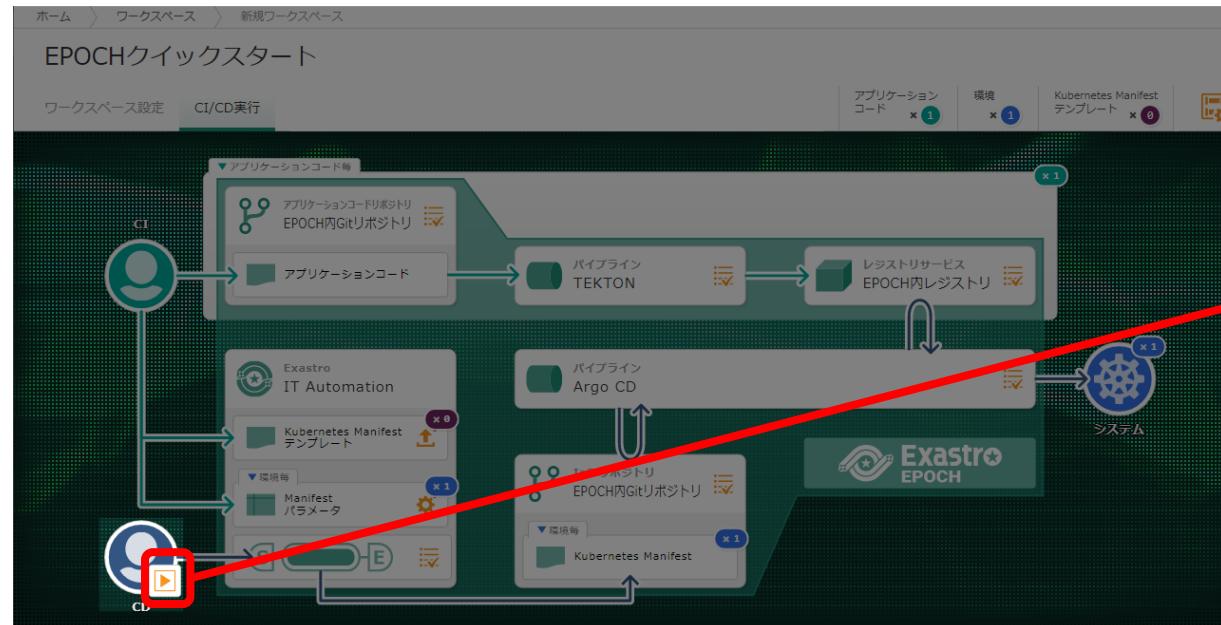
- CD実行で、実際にStaging環境へDeployします。



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。



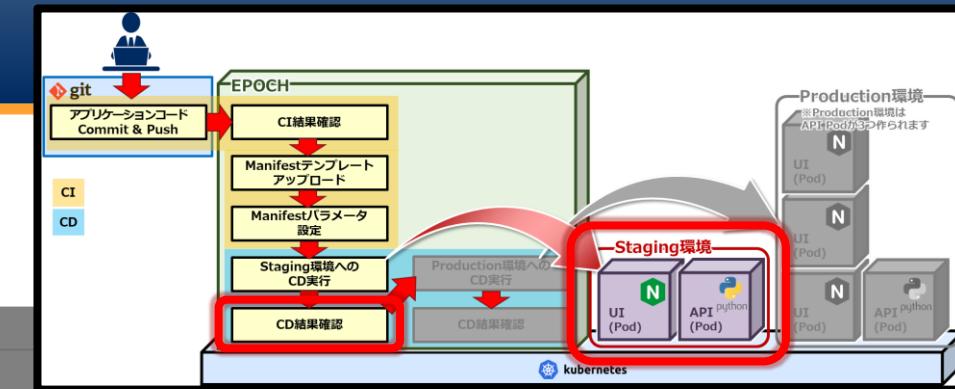
The dialog box shows the configuration for the CD execution. Under '実行条件' (Execution Conditions), '即実行' (Run Now) is selected. Under '環境' (Environment), 'staging' is chosen. A callout box highlights this selection with the text 'stagingを選択します' (Select staging). Below this, under 'Manifest/パラメータ' (Manifest/Parameters), the environment name is set to 'staging'. The 'Manifestリポジトリ' (Manifest Repository) is set to 'https://github.com/epoch-team/argocd_manifest.git'. The 'Kubernetes API Server URL' is 'https://kubernetes.default.svc' and the 'Namespace' is 'staging-app'. At the bottom, a red button labeled '実行' (Execute) is highlighted with a callout box containing the text '環境選択後、実行ボタンを押下します' (Press the execute button after selecting the environment).

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. It shows the Exastro IT Automation interface with various components like EPOCH Quick Start, Application Code, Environment, Pipelines (TEKTON, Argo CD), and Kubernetes Manifest. A red box highlights the 'Conductor' icon in the bottom navigation bar.

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

A yellow callout points to the 'Conductor' menu item in the left sidebar, which is then selected to view the 'Conductor作业一覧' (Conductor Job List) screen. A red box highlights the 'Conductor作业一覧' button in the sidebar.

[2] フィルタをクリック

A yellow callout points to the 'フィルタ' (Filter) button on the 'Conductor作业一覧' screen. A red box highlights the 'フィルタ' button.

[3] 【CD実行】の詳細をクリック

A yellow callout points to the '詳細' (Details) button for a specific CD execution entry in the list. A red box highlights the '詳細' button.

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

A yellow callout contains the login credentials: 'epoch-user' and 'c2jthascR93ijdcyzwJY'. A red box highlights the 'ログイン' (Login) button on the 'Exastro IT Automation' login screen.

すべて[DONE]と表示されていれば完了です

A yellow callout points to the 'CHECKING' tab of the 'Conductor' interface, showing a pipeline with several green 'DONE' status boxes. A red box highlights the 'DONE' status in the pipeline.

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, there's a 'CI/CD実行' (Execution) section with a flowchart showing 'CI' leading to 'アプリケーションコード' (Application Code), which then goes through 'TEKTON' and 'Argo CD' to 'レジストリサービス' (Registry Service). Below this, the 'Applications' section shows two projects: 'production' and 'staging'. The 'staging' project is highlighted with a red box and a callout 'stagingを選択' (Select staging). A red arrow points from the 'Argo CD' step in the flowchart to the 'staging' project in the applications list. Another red arrow points from the 'staging' project to a detailed view of the 'staging' application. This detailed view shows the application is 'Healthy' and has a 'Sync OK' status with a timestamp of '2021-07-08T17:18:26 GMT+0900'. The UI also displays various Kubernetes components like 'ui-app-com' and 'api-app-bluegreen'. 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).

Username: admin
Password: iYSCKzx2wvxJnn4dCNwN
でSIGN INします

※「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

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 OK】が表示されましたたら完了です
※日付がCD実行後であることを確認してください

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

4.6 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 € 14.91 € 55.91 € 35.04

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 TシャツH

€ 8.95 € 26.84 € 48.46 € 21.99

詳細はこちら 詳細はこちら 詳細はこちら 詳細はこちら

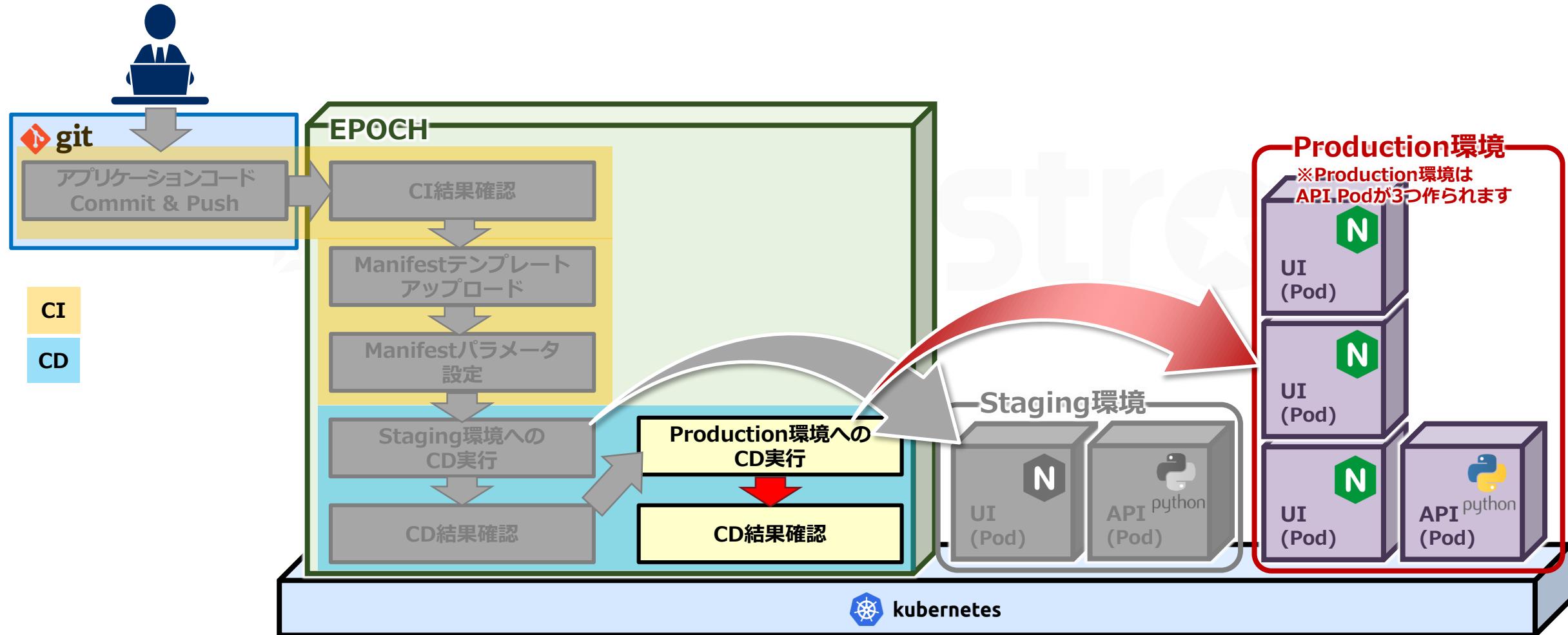
購入サイトはこちら 購入サイトはこちら 購入サイトはこちら 購入サイトはこちら

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

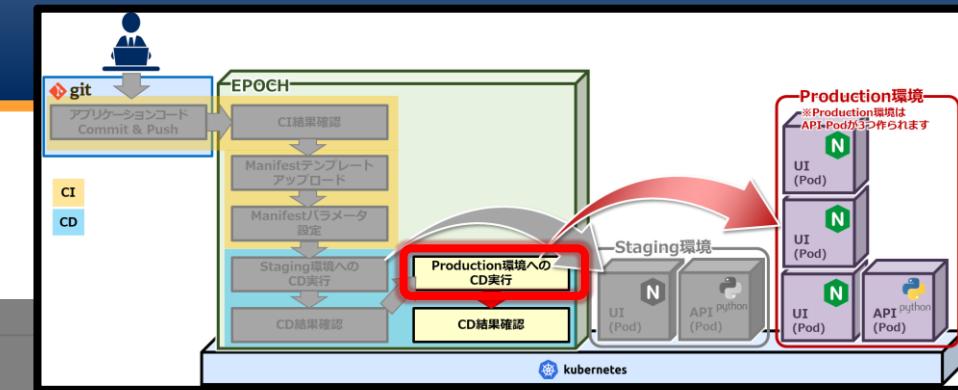
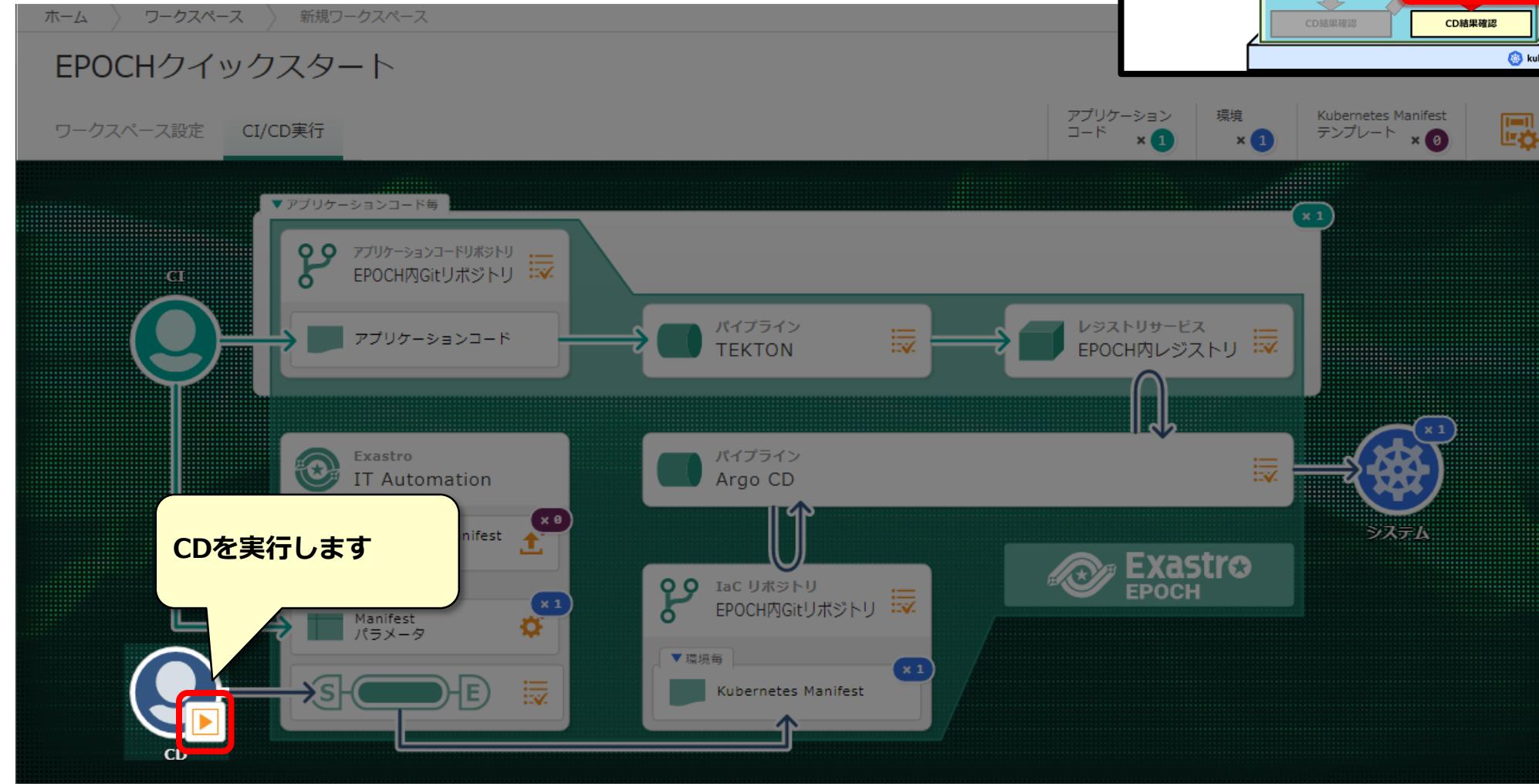
Staging環境へDeploy後、Production環境へDeployする流れを説明します。



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

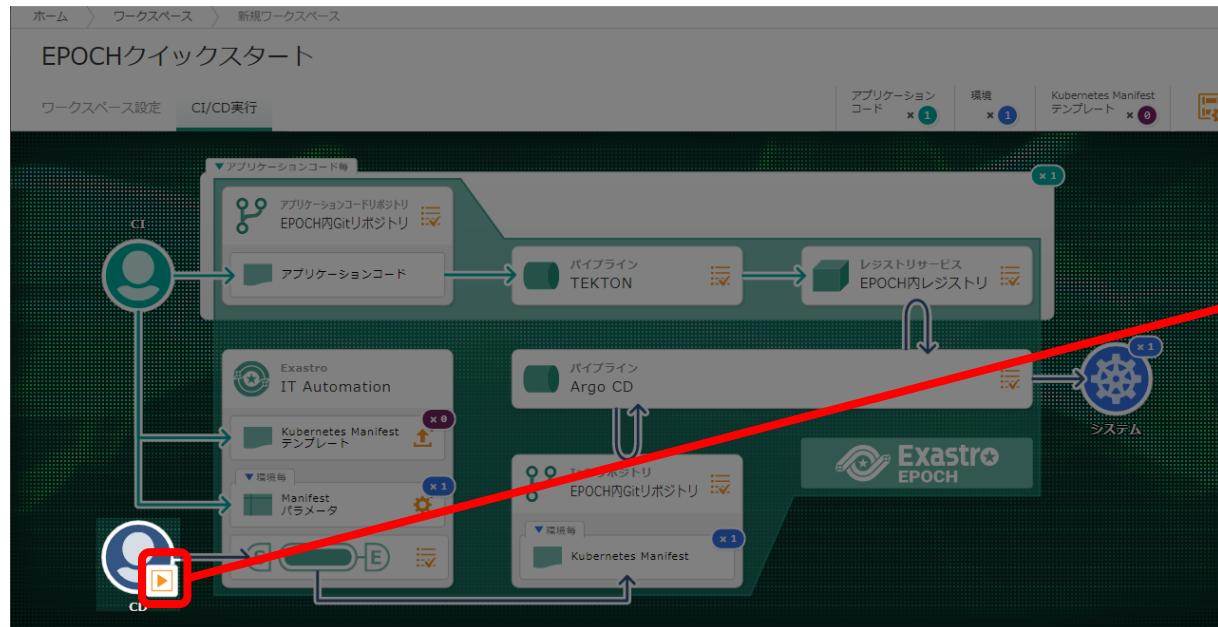
- CD実行で、実際にProduction環境へDeployを実行します。



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。

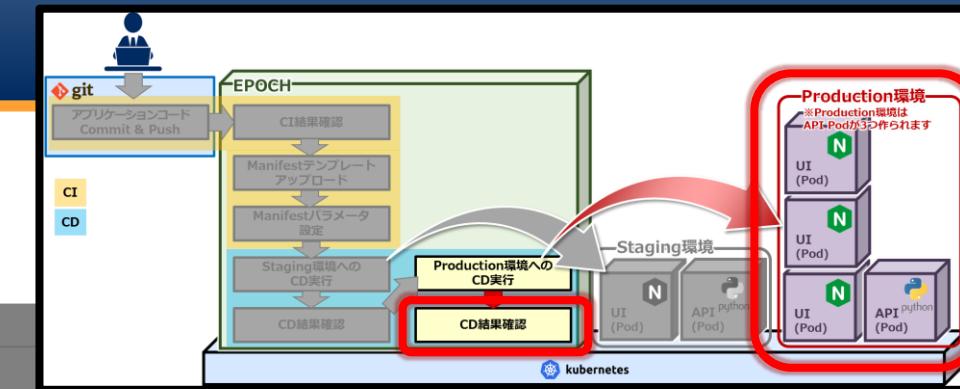


Production環境へのCD実行が完了しました
CD実行結果を確認してみましょう

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.

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

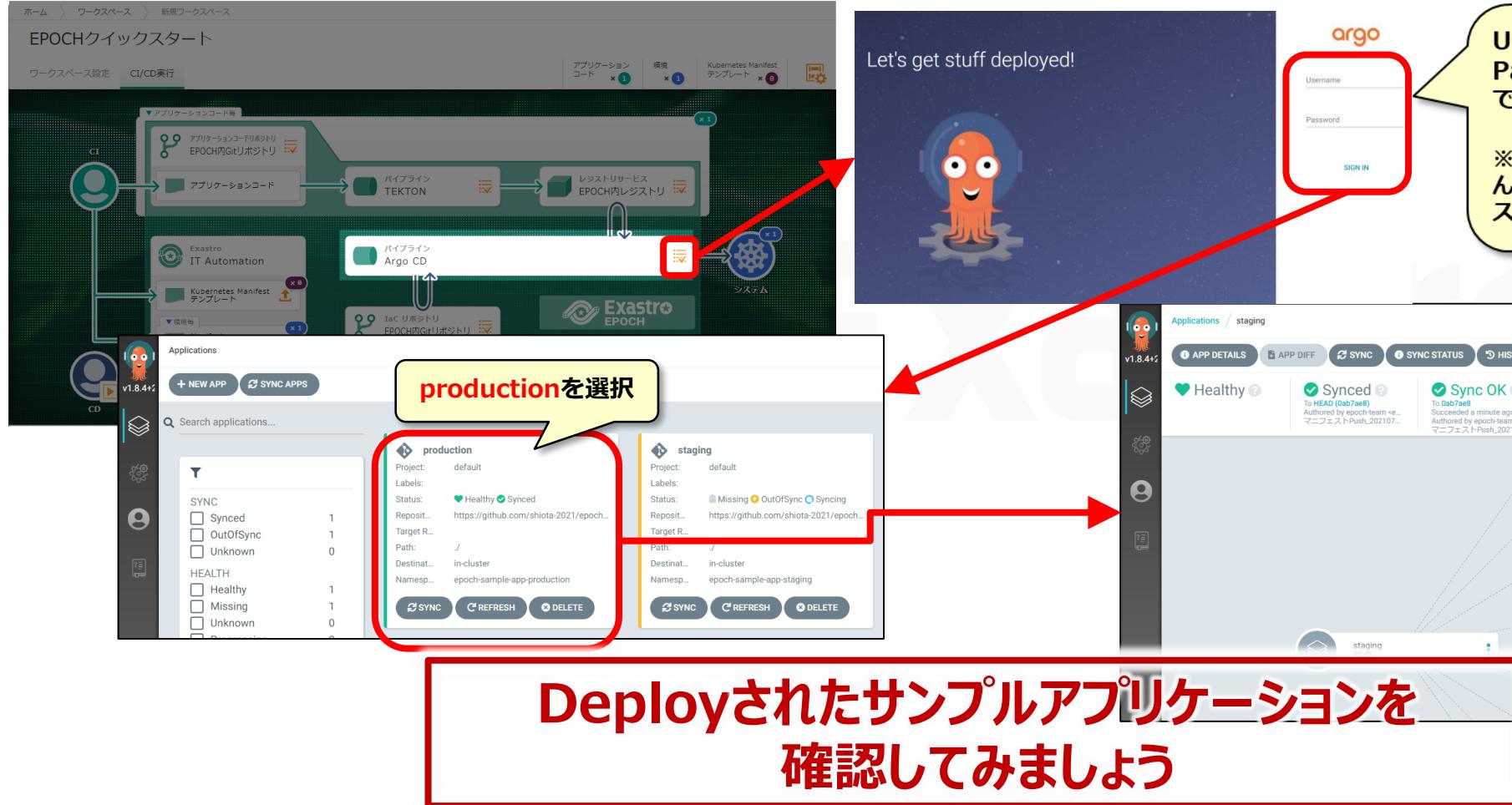
**ログインID: epoch-user
パスワード: c2jthascR93ijdcyzwJY
でログインします**

すべて[DONE]と表示されていれば完了です

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。



Username: admin
Password: iYSCKzx2wvxJnn4dCNwN
でSIGN INします

※「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

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 OK】が表示されました
※日付がCD実行後であることを確認してください

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 