EPOCH Quick Start

第0.1.0版

Exastro developer

目次

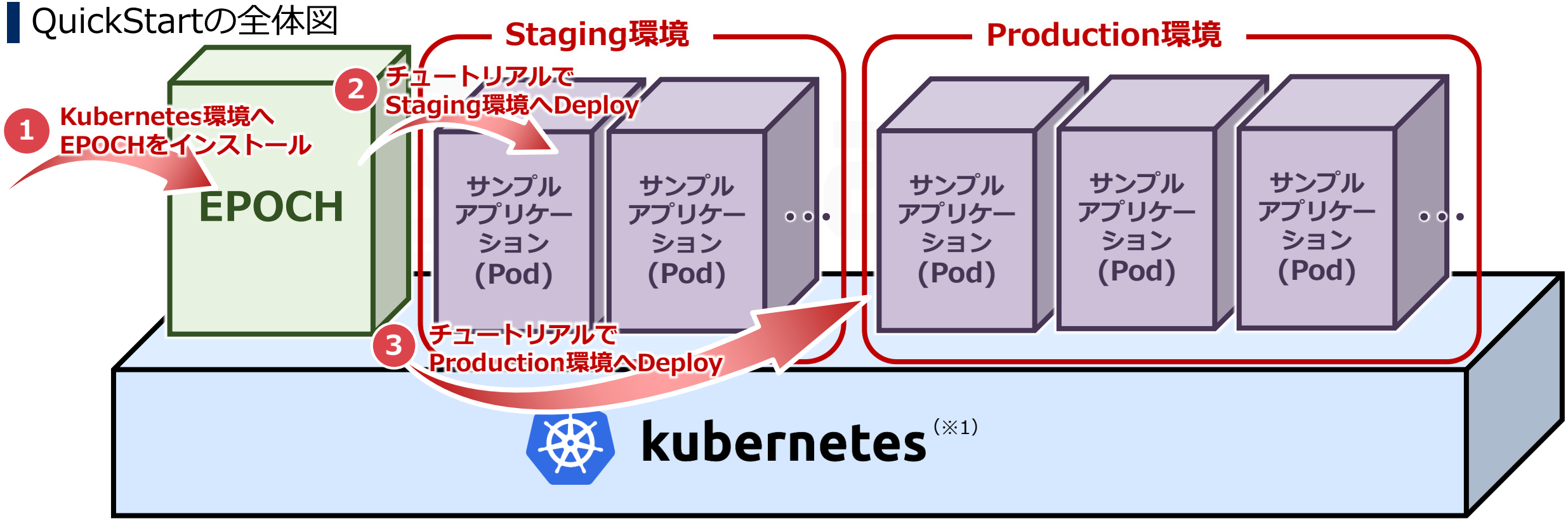
1. はじめに

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。

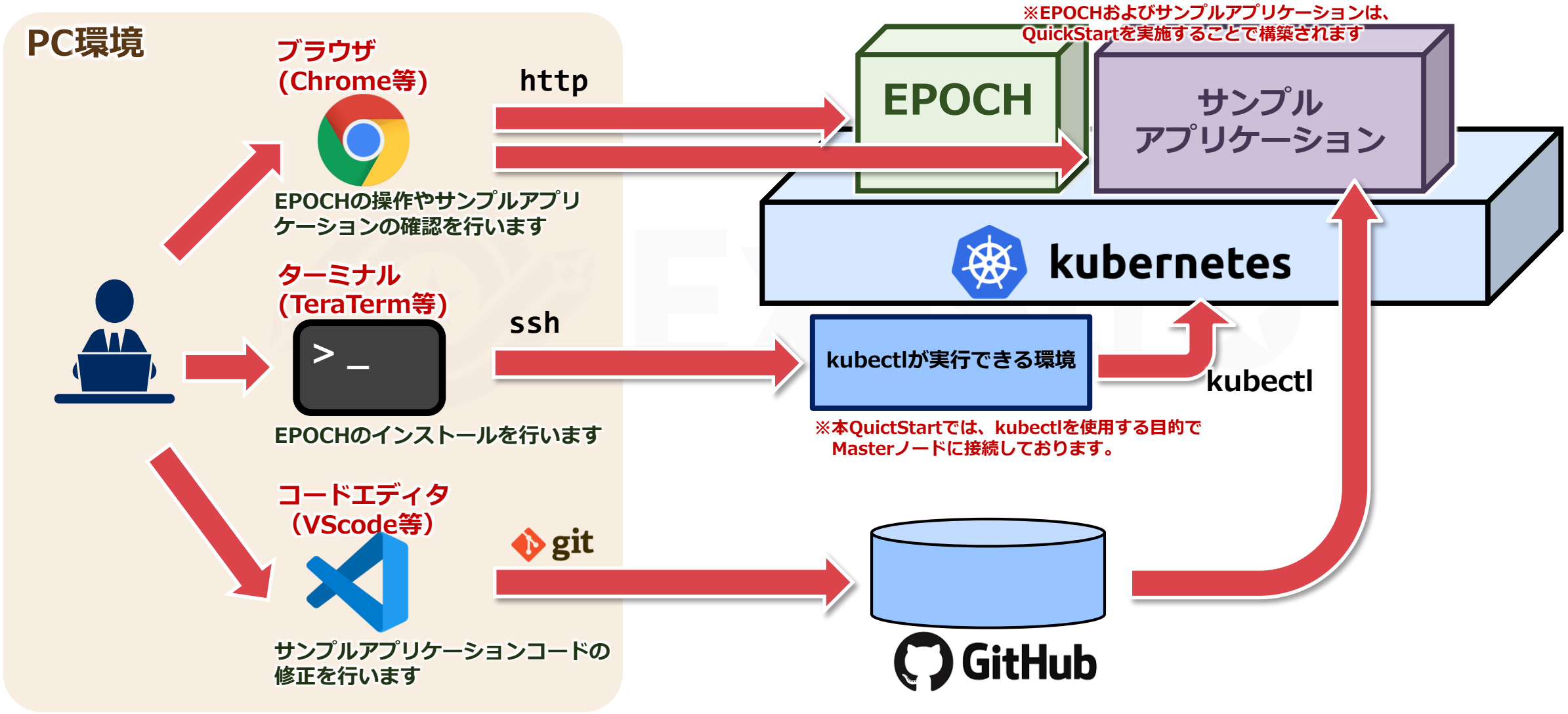
QuickStartの全体図



(※1)本クイックスタートでは手順を簡素化するため1つのKubernetesクラスタ上で構成します

1.2 QuickStartを実施するPC環境ついて

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。



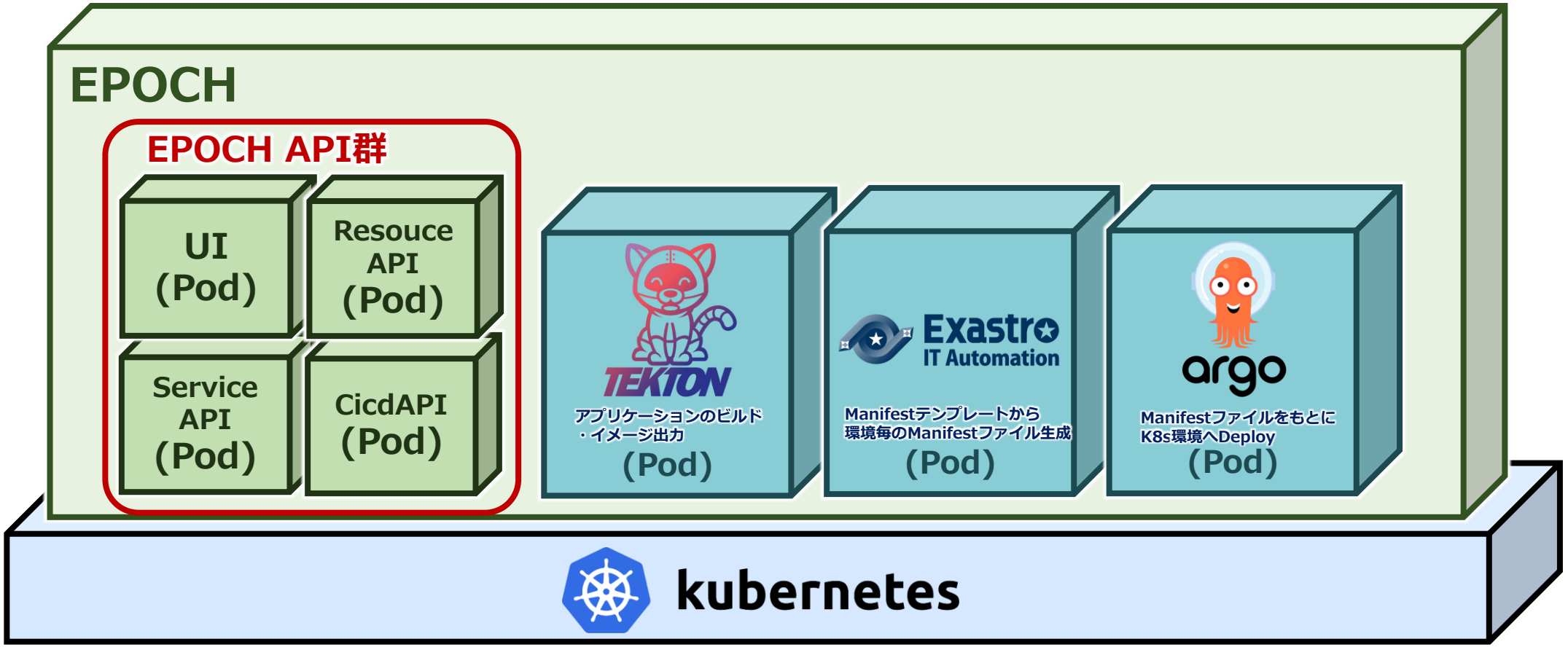
2. インストール

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
```

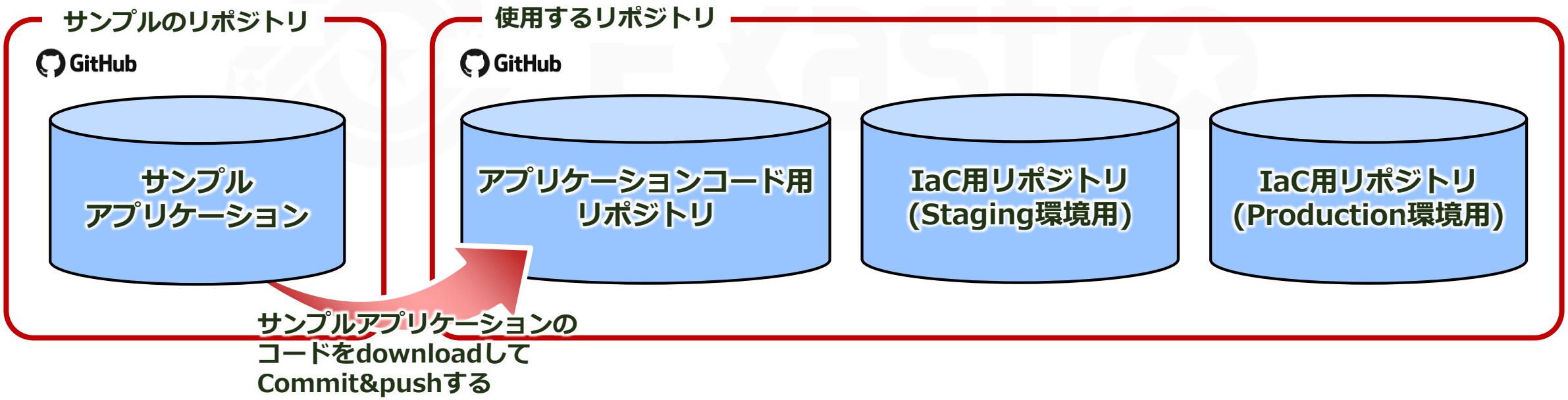
以上でEPOCHのインストールは完了しました。
次にチュートリアルを実施するための事前準備を実施しましょう！

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

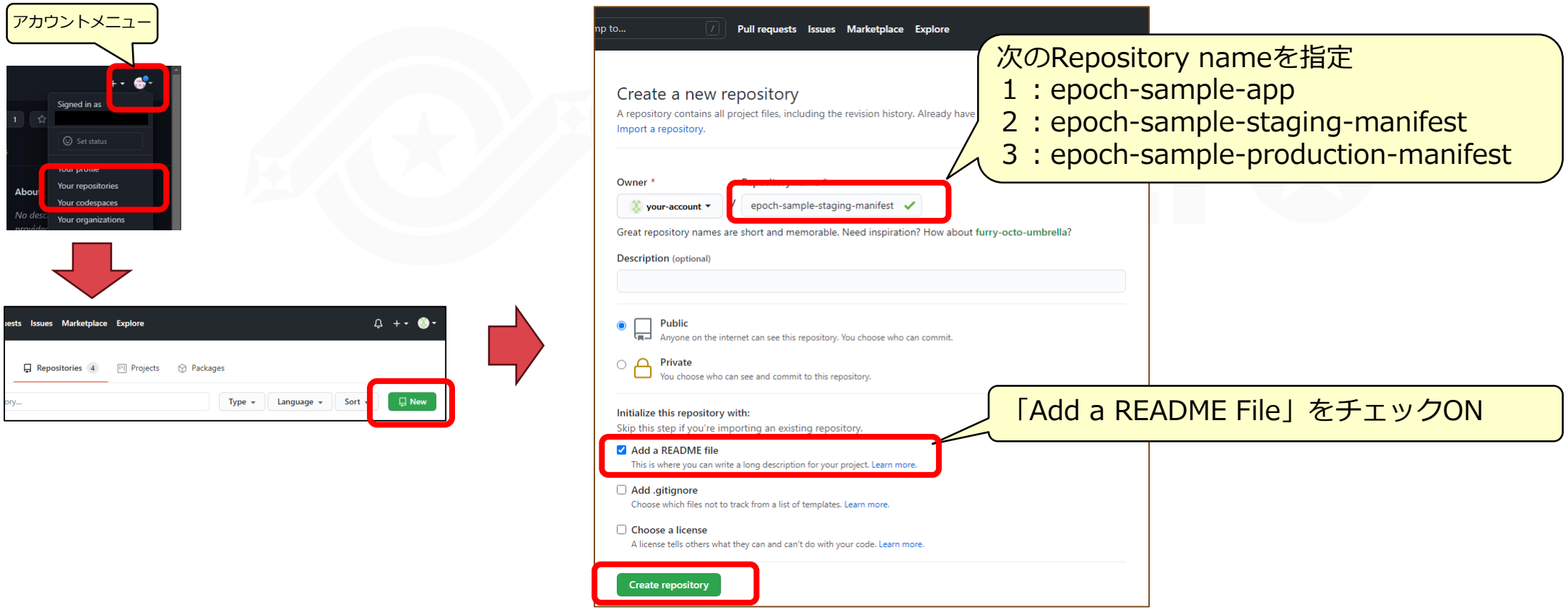


2.2 リポジトリ準備(2/4)

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 Repositriesを選択します。
- Newを選択し、図で示した値を入力し、Create repositryを選択します。



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。

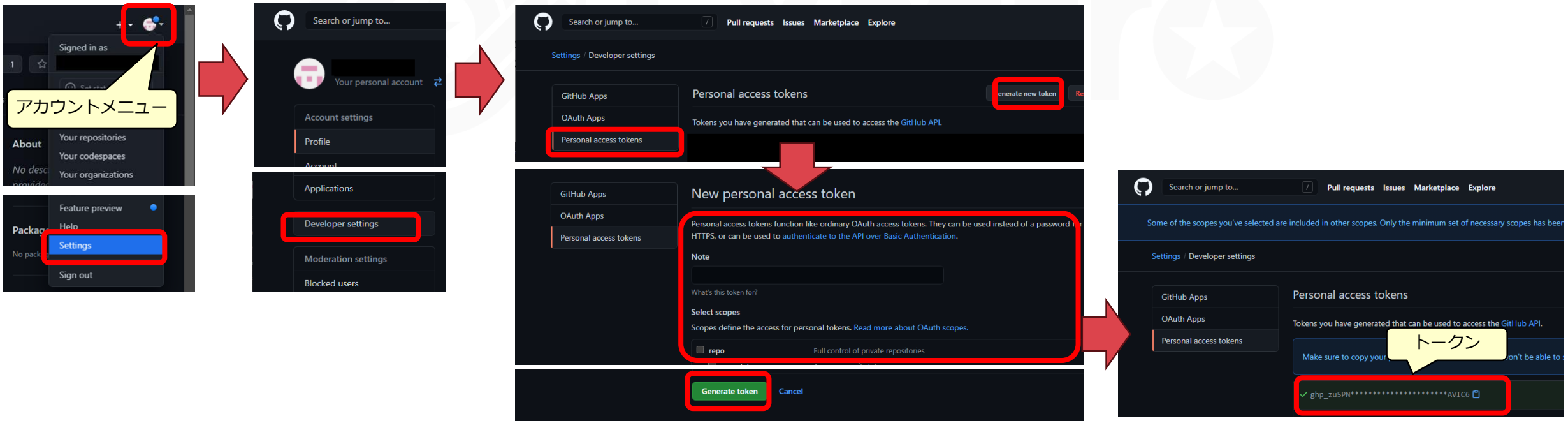
```
>   cd "[clone先のフォルダ]"
>   git clone https://github.com/[Githubのアカウント名]/epoch-sample-app.git
>   cd epoch-sample-app
>   git config user.name "[GitHubのユーザ名]"
>   git config user.email "[GitHubのemailアドレス]"
```

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

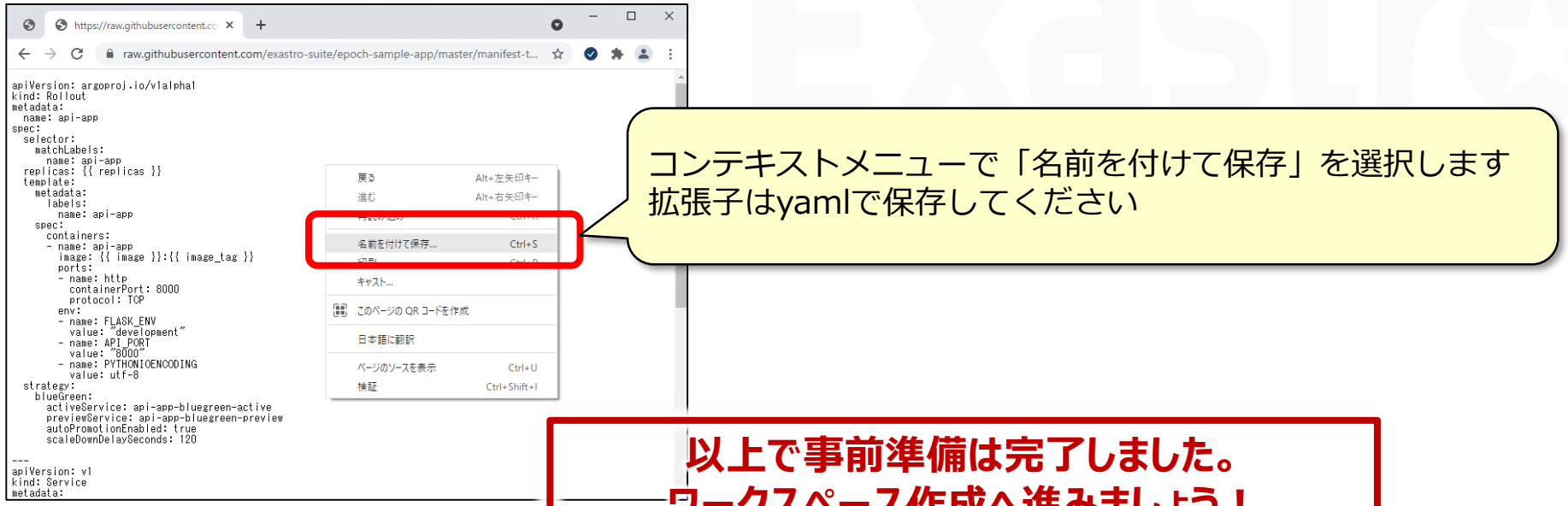
EPOCHにアップロードするManifestテンプレートファイル（２ファイル）をダウンロードします。

- 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

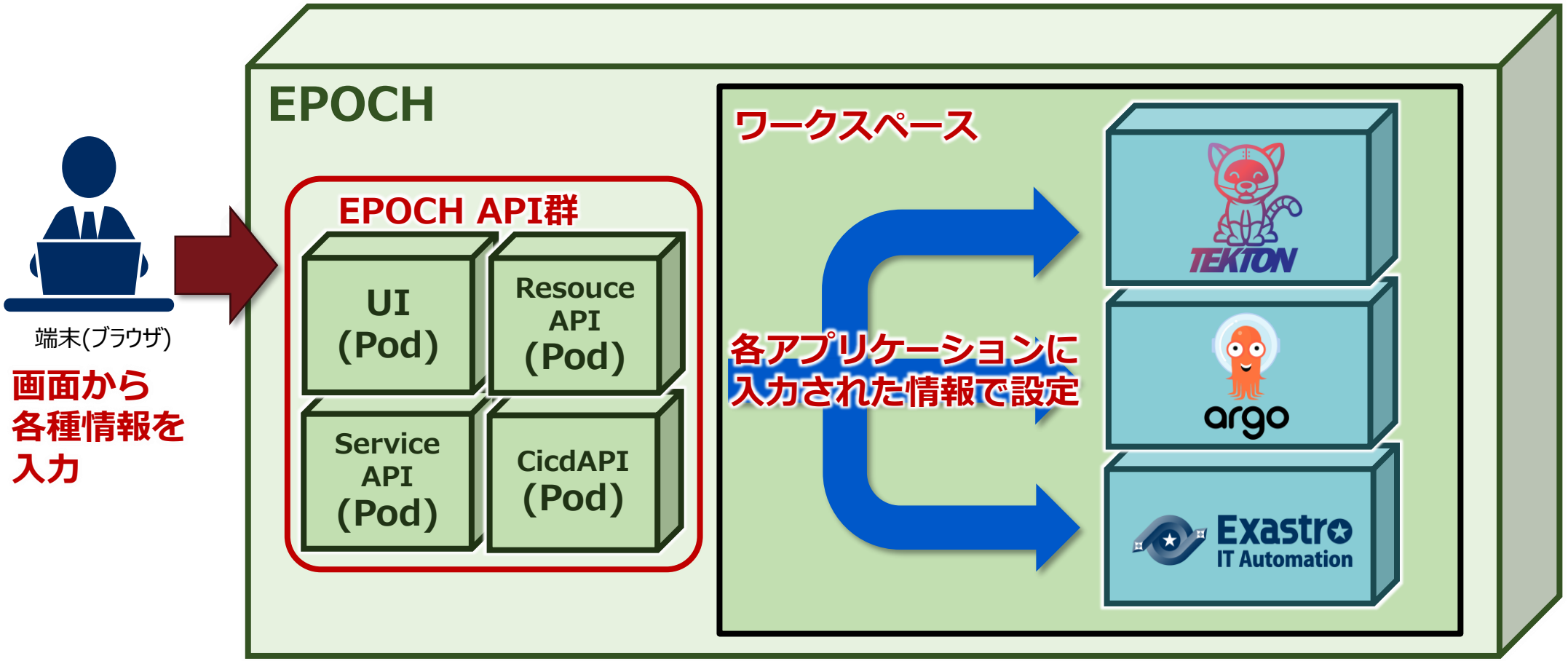
ワークスペースを作成し、CI/CDの準備をしましょう。

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。

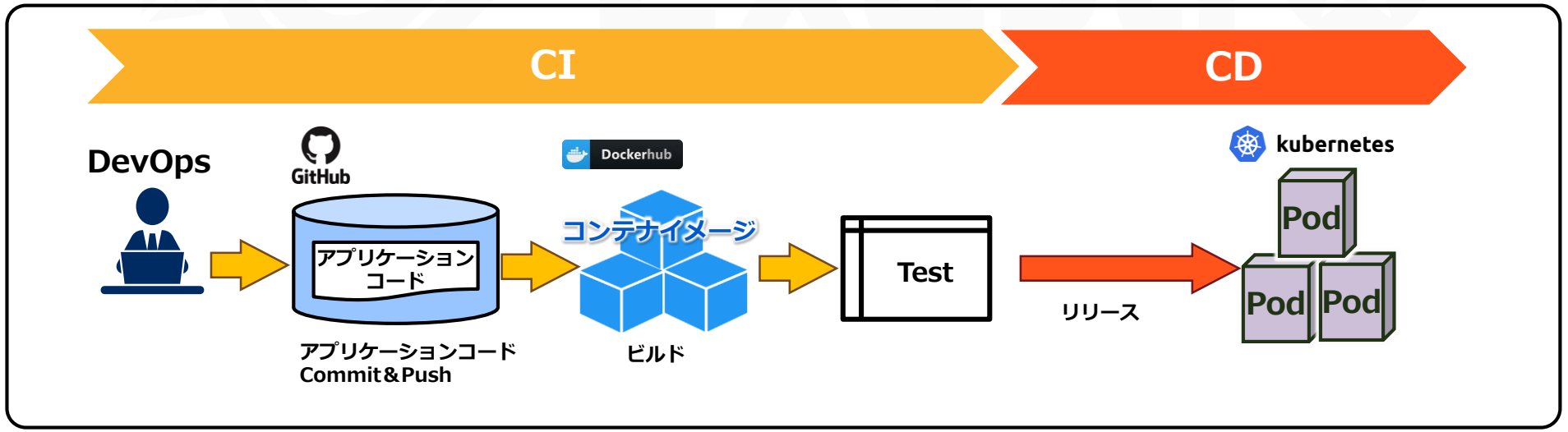
- CI（継続的インテグレーション）

  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。

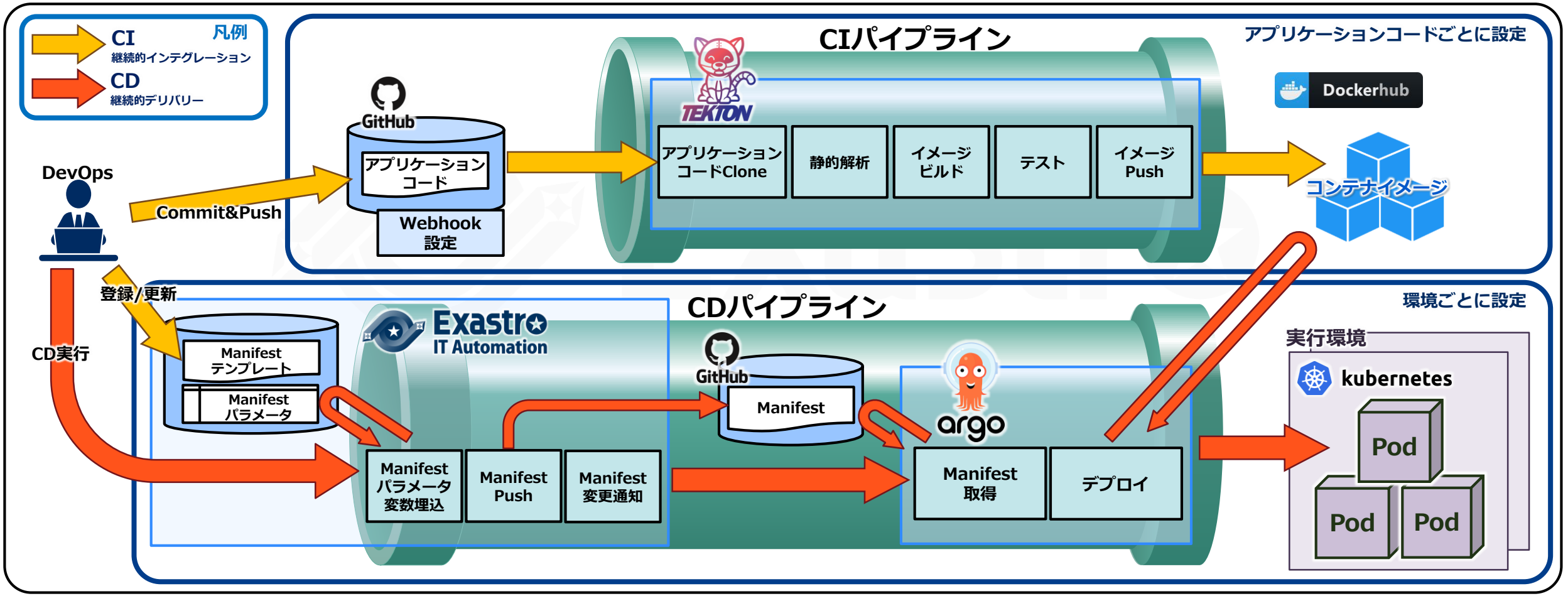
- CI/CDのイメージ



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

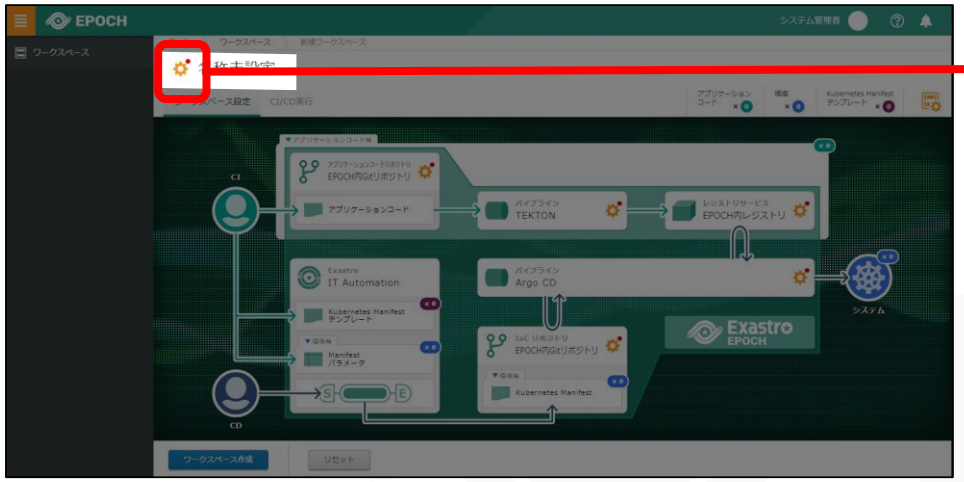
ブラウザより以下のURLで接続します。

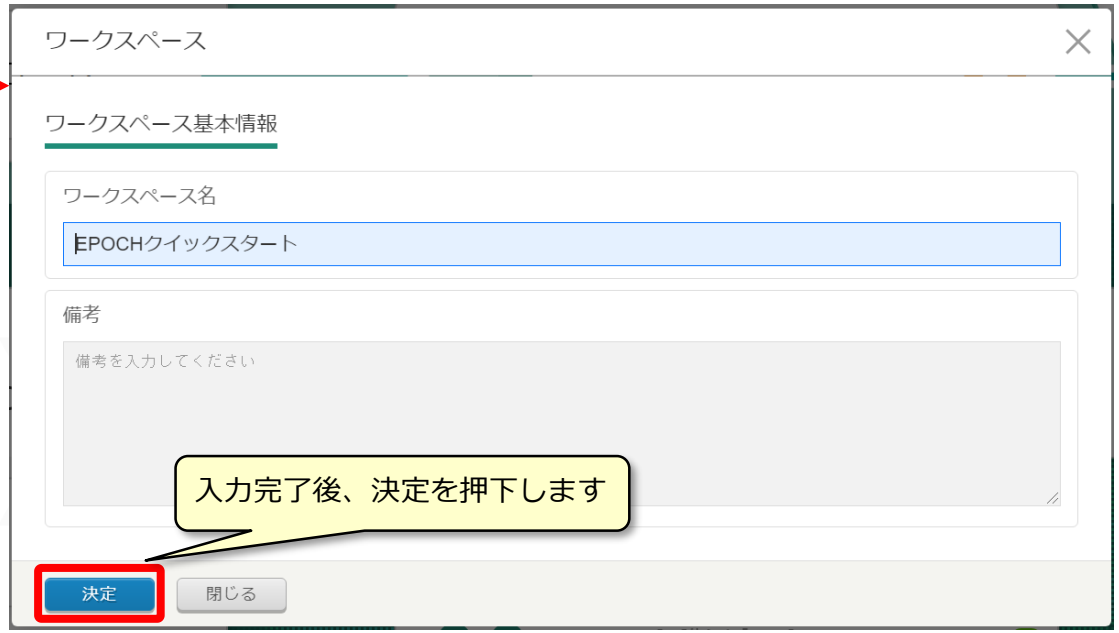
https://[インストール先のIPアドレスまたはホスト名]:30443/workspace.html

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。



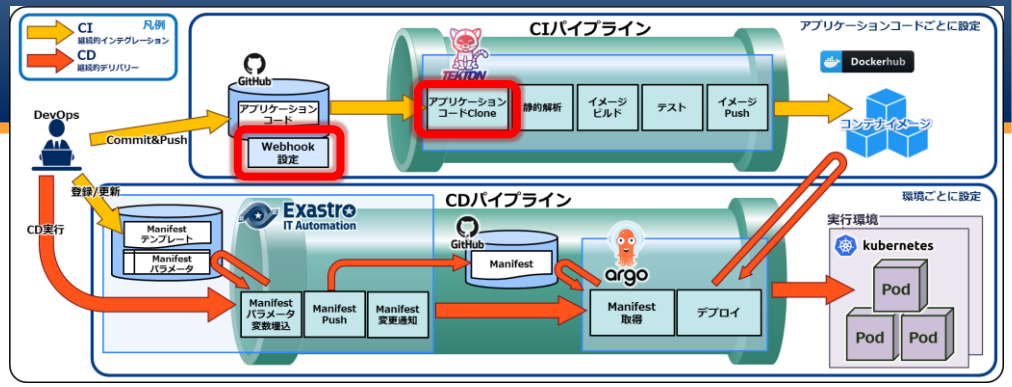


項目	入力・選択内容	説明
ワークスペース名	EPOCHクイックスタート	作成するワークスペース名
備考	なし	作成するワークスペースの説明や備考

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。



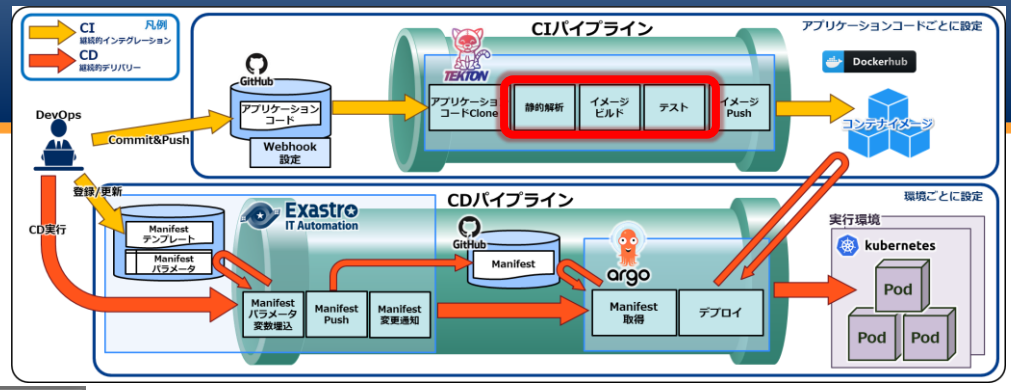
※赤枠部分の設定値を指定しています

項目	入力・選択内容	説明
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。



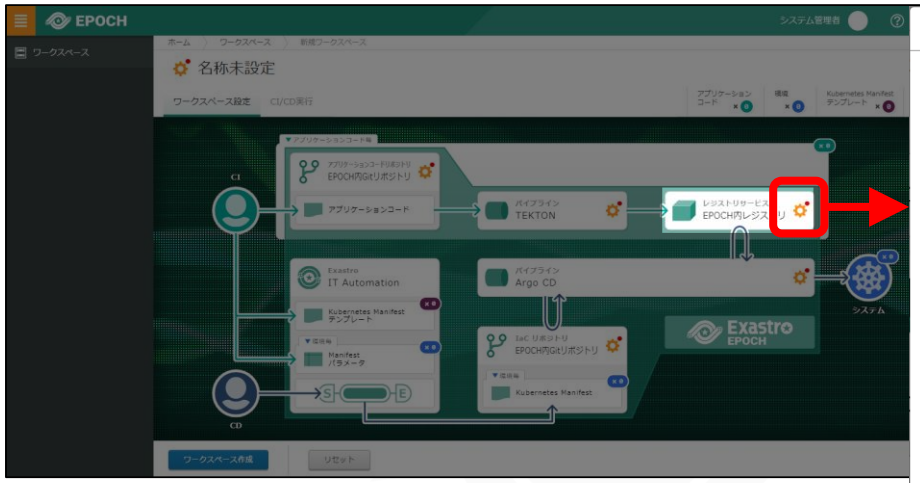
※赤枠部分の設定値を指定しています
※静的解析、テスト等については対応する予定です

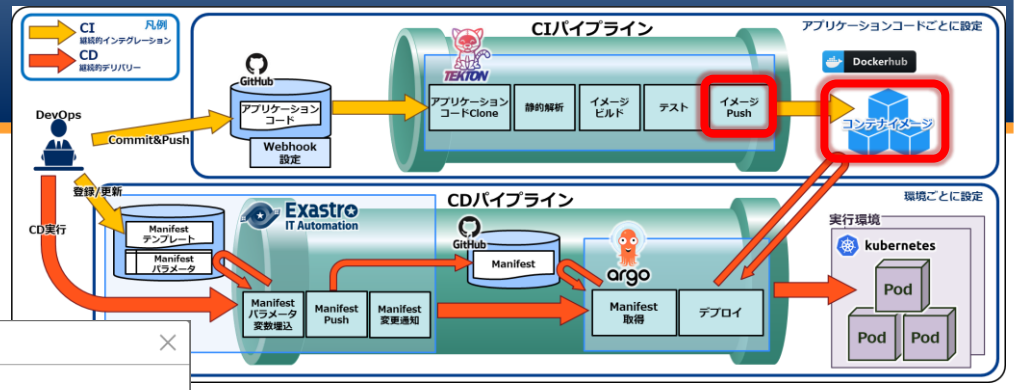
項目	入力・選択内容	説明
ビルドブランチ	main,master	ビルド対象のアプリケーションのGitHubのブランチ
ビルドDockerファイルパス	./api-app/Dockerfile	アプリケーションのDockerfileのパス

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。





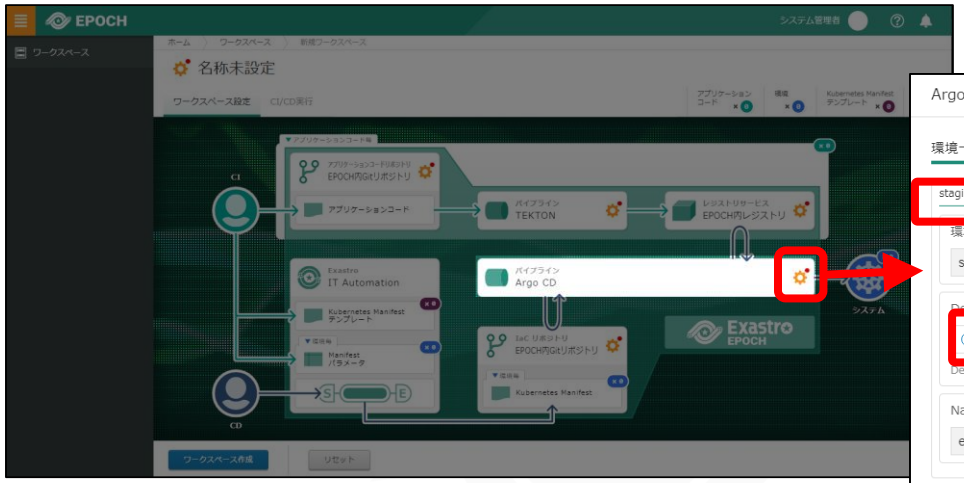
※赤枠部分の設定値を指定しています

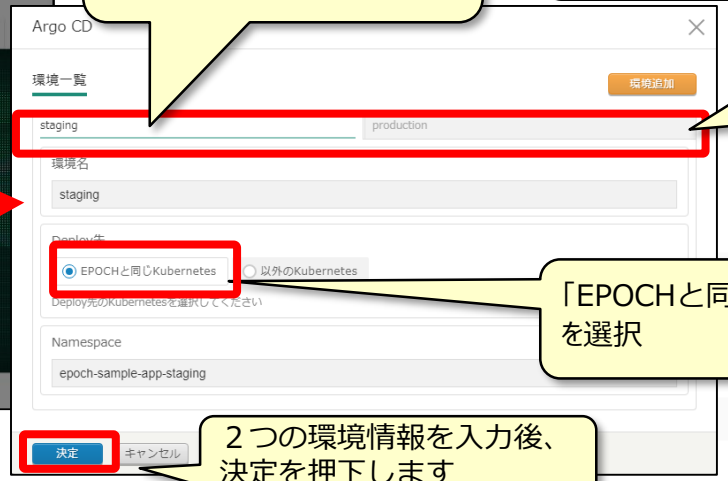
項目	入力・選択内容	説明
ユーザ名	（自身のDockerHubのアカウント名）	DockerHubのアカウント名
パスワード	（自身のDockerHubのパスワード）	DockerHubのパスワード
イメージ出力先	[DockerHubのアカウント名]/epoch-sample-**api** ※ユーザ名入力後に表示される内容を修正してください。	DockerHubのイメージ出力先のパス

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。



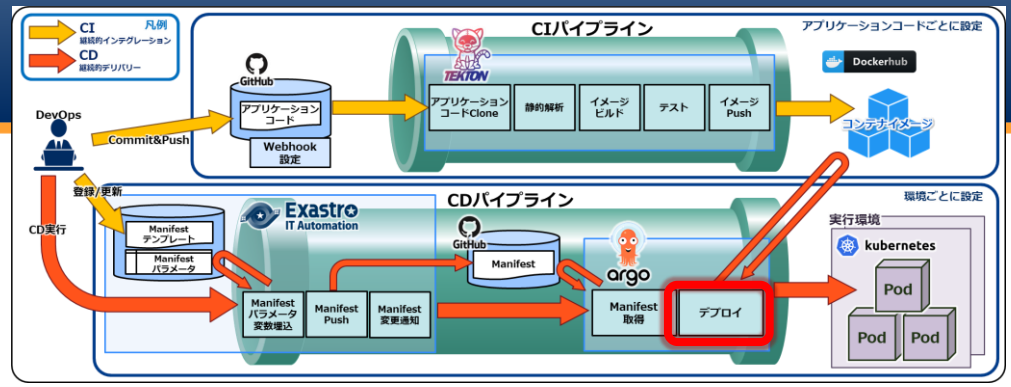


環境の１つめと２つめはタブを選択して切り替えます

本QuickStartでは2つの環境に対してDeployしますので、【環境追加】ボタンを押下して、環境を追加します

「EPOCHと同じKubernetes」を選択

２つの環境情報を入力後、決定を押下します



※赤枠部分の設定値を指定しています

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

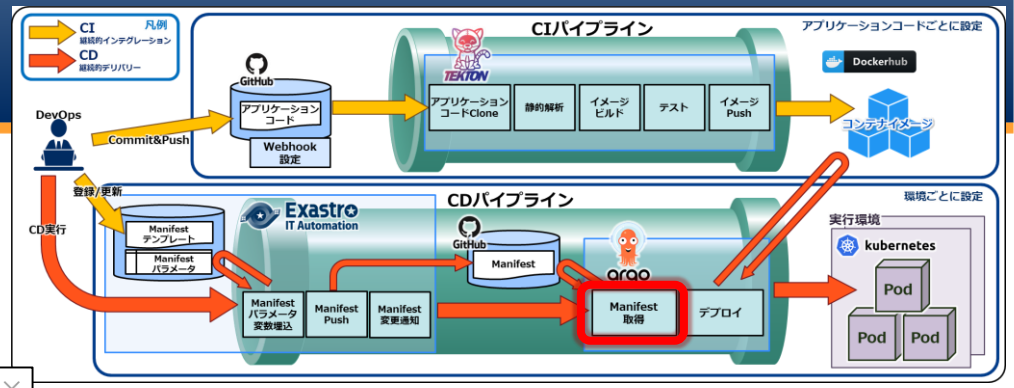
環境2：Production環境

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。



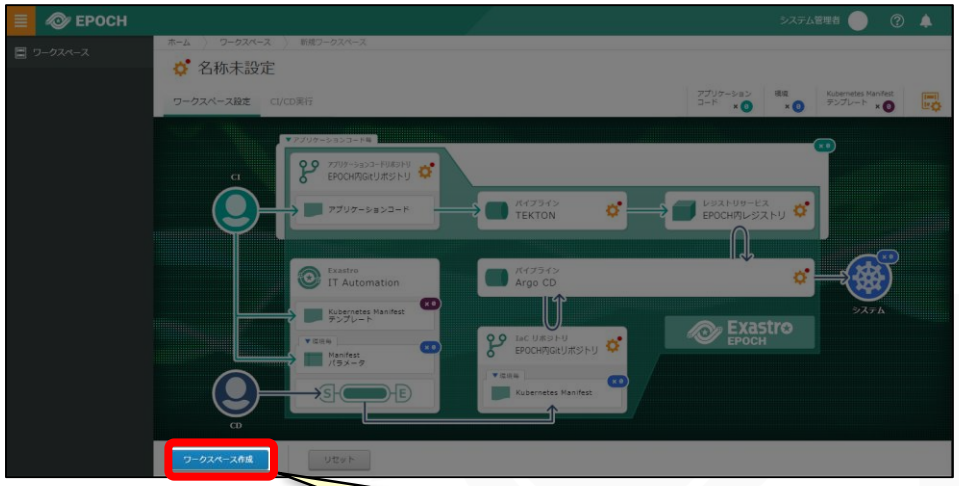
※赤枠部分の設定値を指定しています

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。

依頼主からの要望で
ECサイトのアプリを開発

依頼主

国内外向けのTシャツ販売の
ECサイトを作ってくれないか

依頼主

クラウドネイティブでCI/CDを
回せるようなシステムでお願いね

かしこまりました！

DevOps

やってみます…。

DevOps

kubernetesでのコンテナオーケストレーションで
構成して、CI/CDを回すためにパイプラインを準備し
て、トリガー設定が必要でそのためには通信がこうし
てああして、CDはシステム不停止の方式にしたいけど
どのパターンか検討しないと、、、

画面(UI)のPodとバックエンド(API)
のPodを建てて、staging環境と
production環境の2系統用意かな

チュートリアルの34p～58pの内容になります。

1回目のCI/CDワークフロー手順を実施

DevOps

CICD環境を準備したり大変だなぁ…
何かないかなぁ…
EPOCHってCICD回せるんだ

EPOCHを使ってみよう！

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

できました！

DevOps

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

依頼主

この前のECサイトはとても好評だったよ

ありがとうございます

DevOps

依頼主

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

かしこまりました！

DevOps

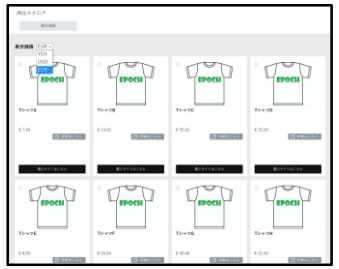
CI/CDの仕組みはできているから
コード修正してからデプロイまでは
単純作業♪

チュートリアルの59p～80pの内容になります。

2回目のCI/CDワークフロー手順を実施

DevOps

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

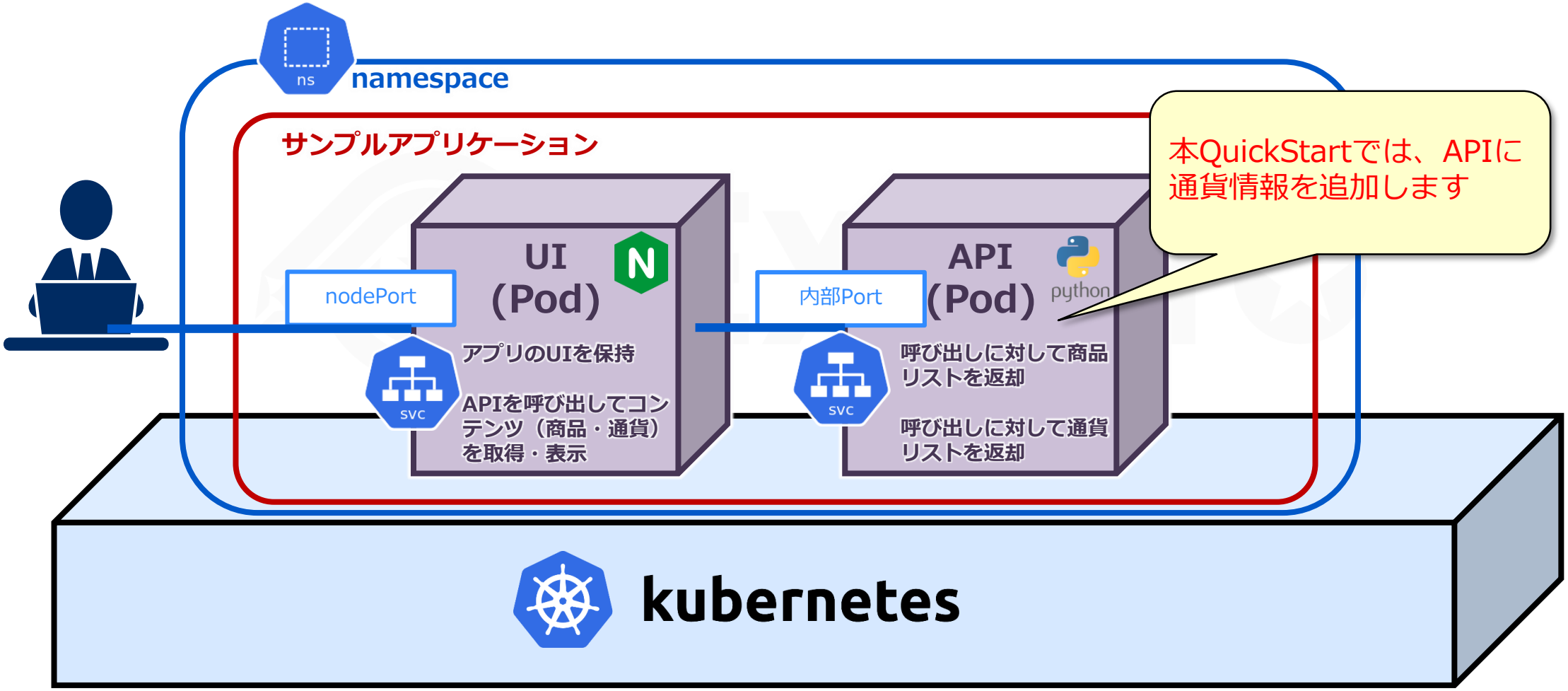


できました！

DevOps

サンプルアプリの構成

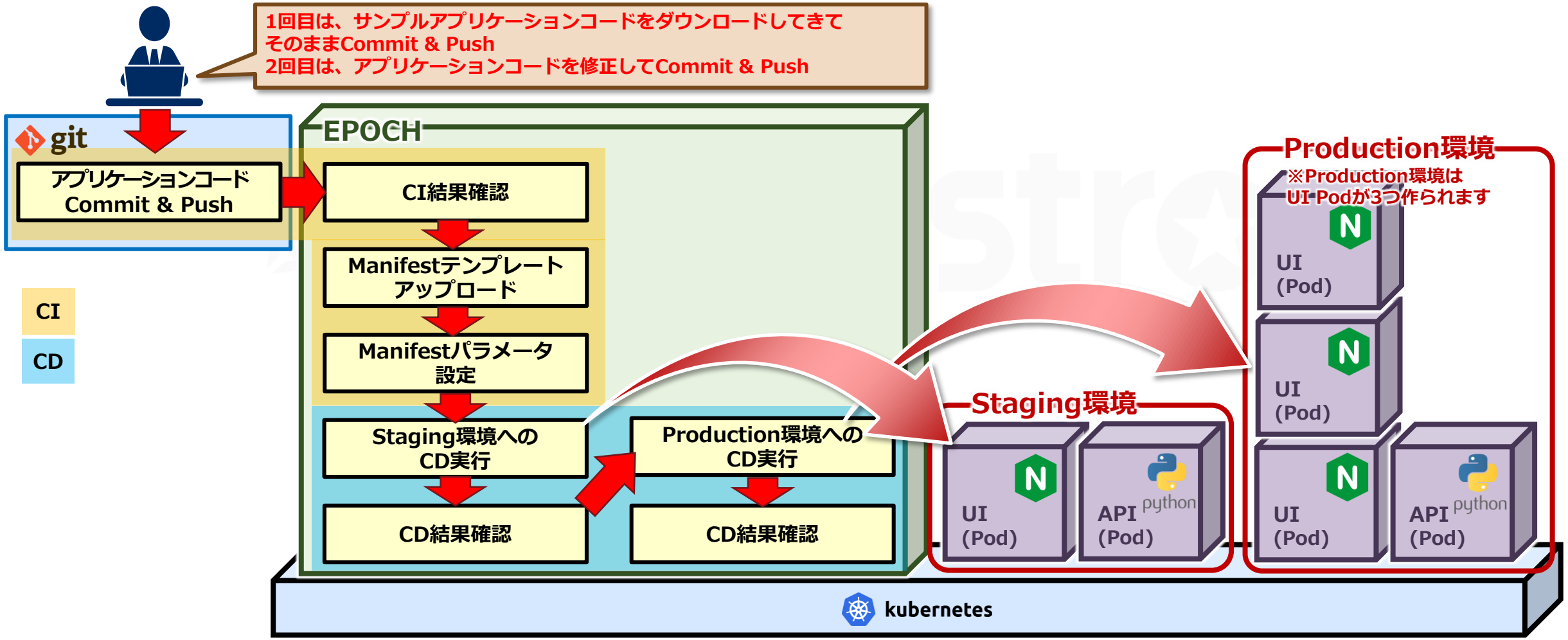
サンプルアプリケーションは、UIとAPIの2つで構成されております。



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の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。

※変数名は、image、image_tag、param01~param20で設定できます。

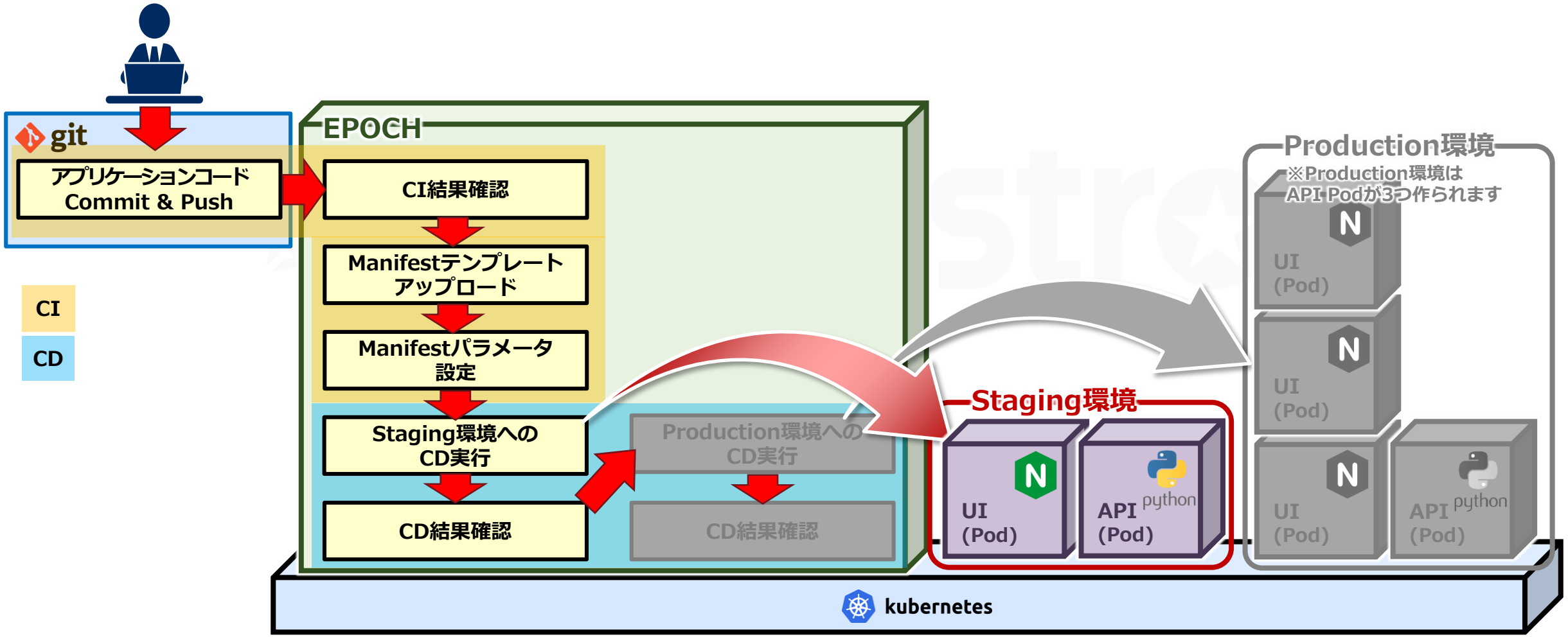
**それでは1回目のCI/CDワークフロー手順を
実行してみましょう！**

1回目のCI/CDワークフロー手順

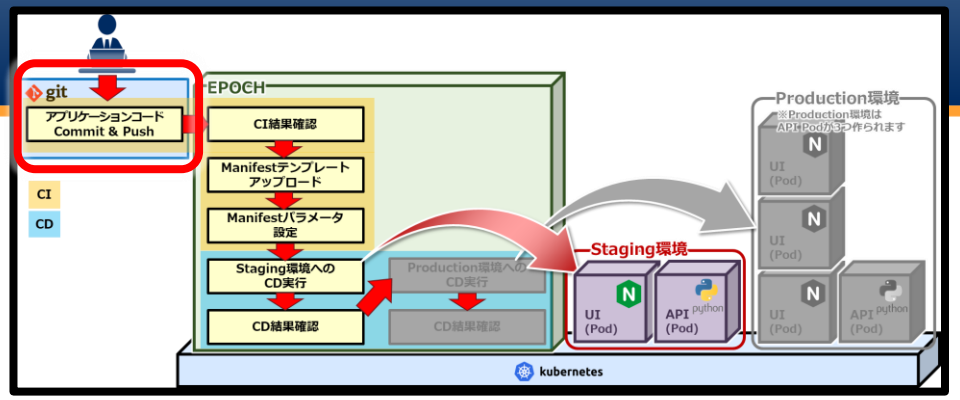
4.5 1回目のCI/CDワークフロー手順(1/25)

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)

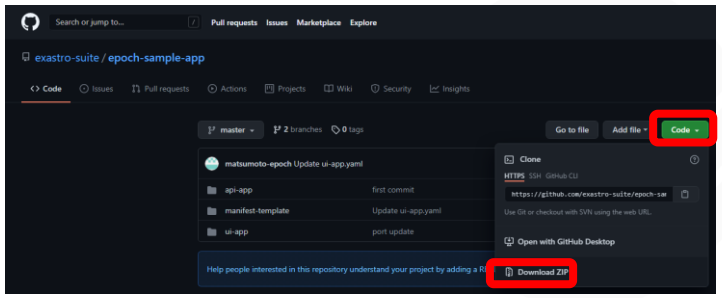


アプリケーションコード Commit & Push

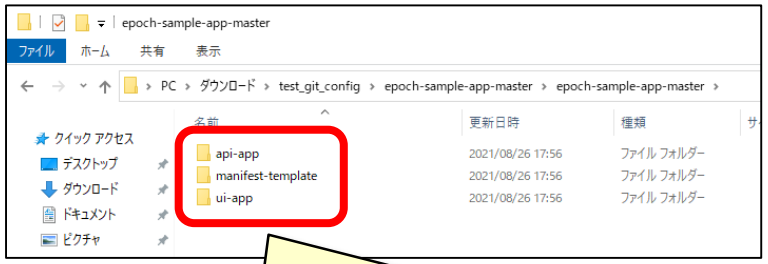
初回デプロイするコンテナイメージを作るためCIパイプラインを実行します。

- 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

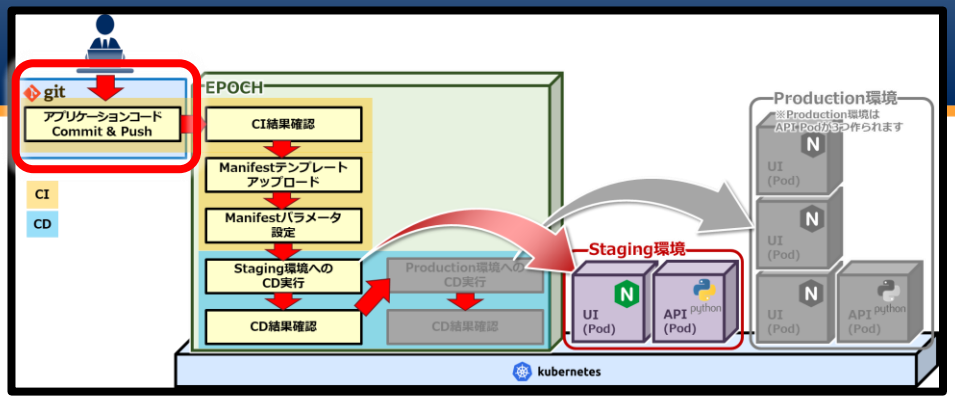
- サンプルアプリケーションの画面から「Code」を押下して、「Download ZIP」からコードを取得します。



- 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
```

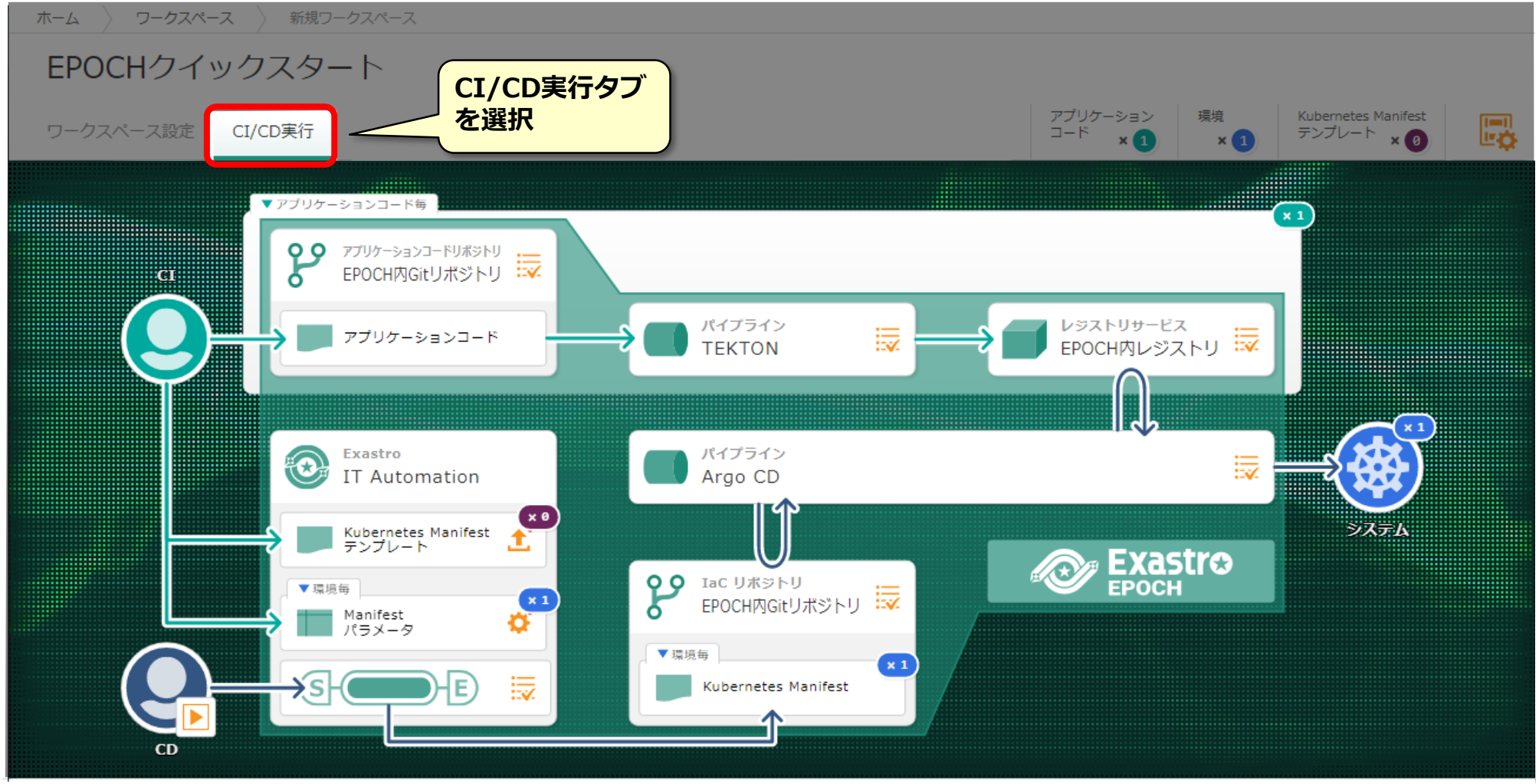
※ git push時に認証情報を求められた場合は、自身のGitHubアカウント情報を入力してください。

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。

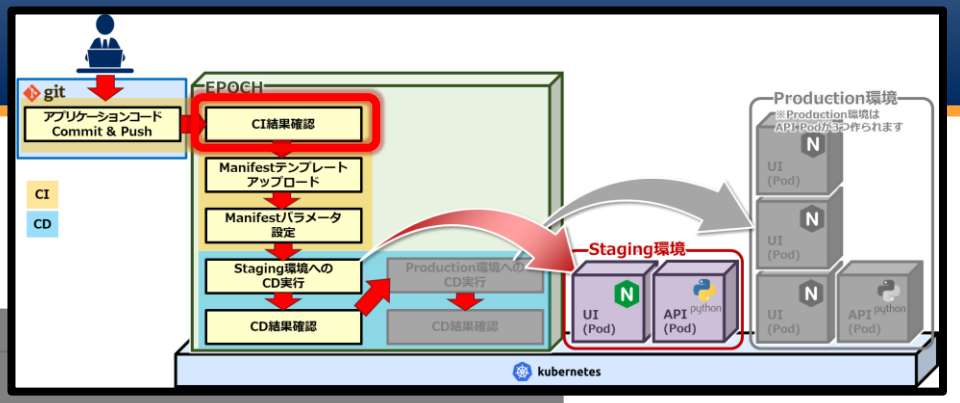
4.5 1回目のCI/CDワークフロー手順(4/25)

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)



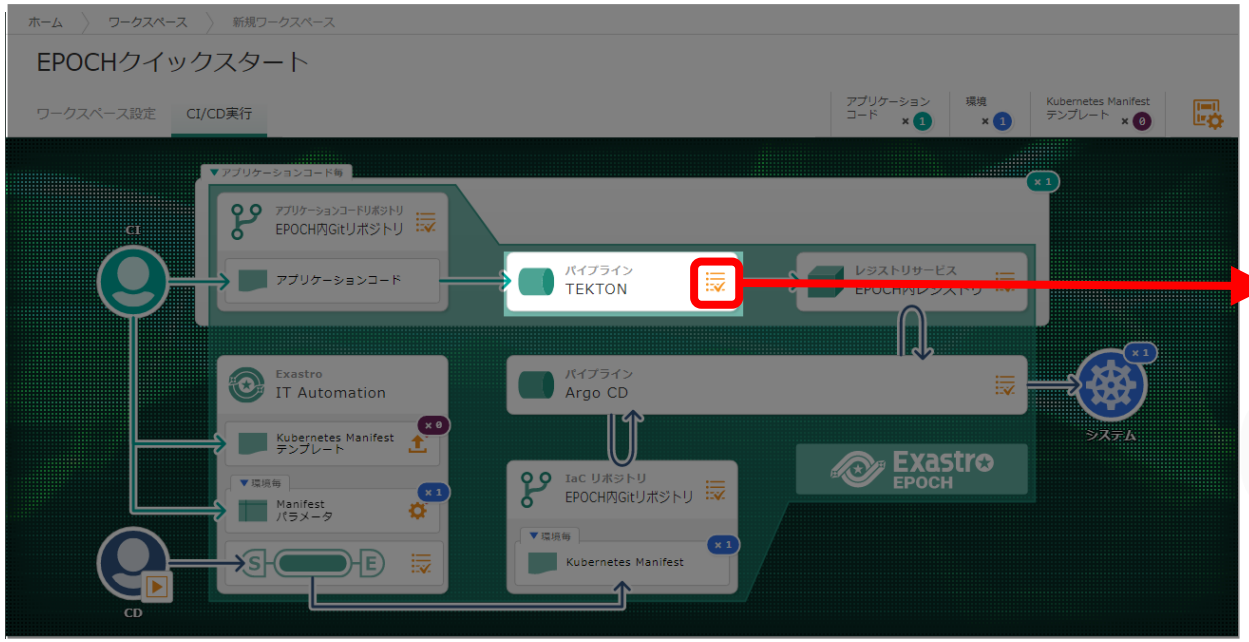
CI結果確認

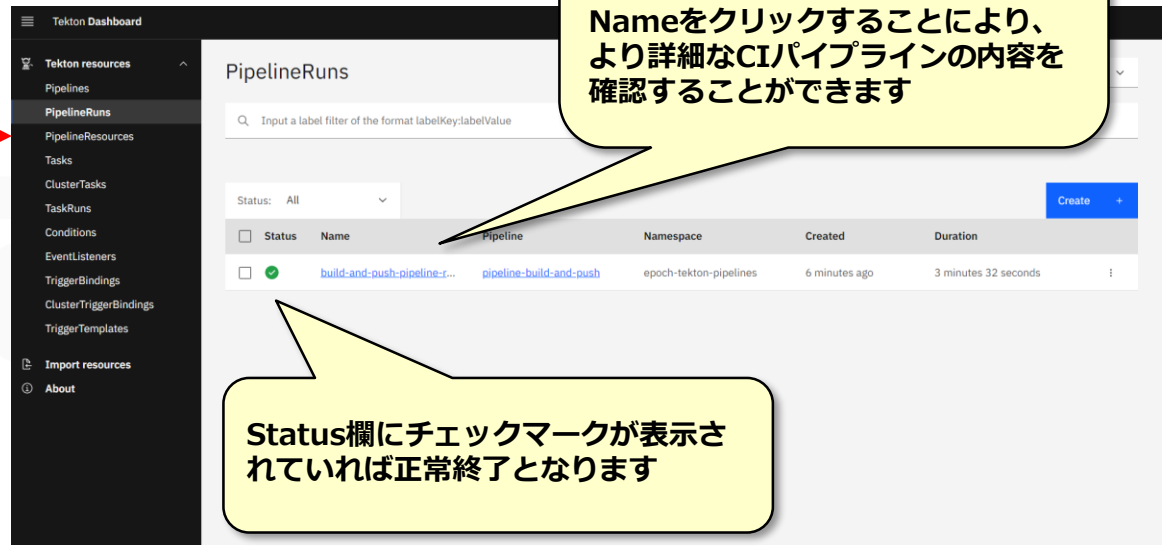
- アプリケーションコードのビルド結果を確認します。

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。





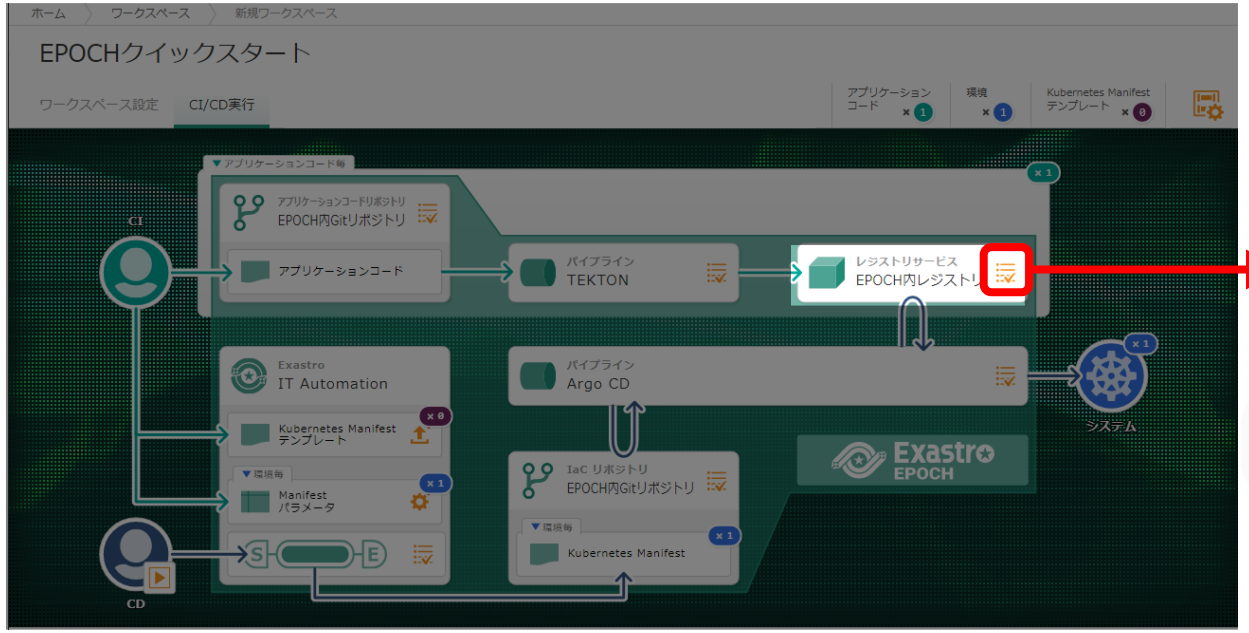
CIパイプラインの動作は、TEKTONダッシュボードにて確認することができます。Nameをクリックすることにより、より詳細なCIパイプラインの内容を確認することができます

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

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

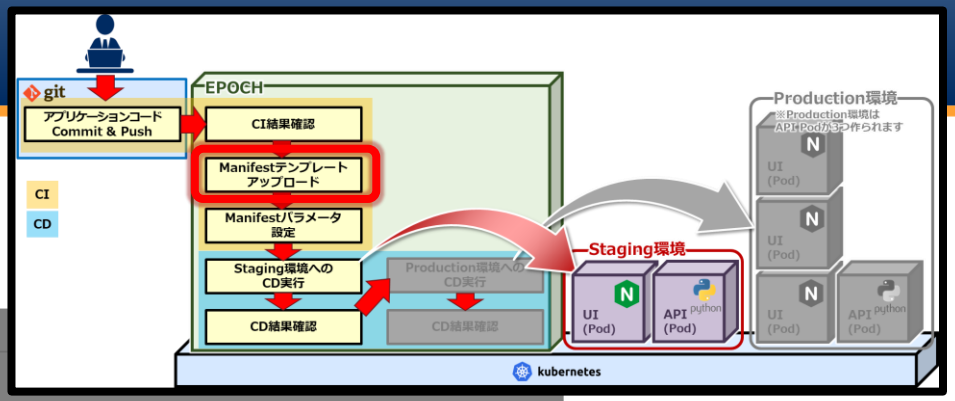
- レジストリサービスの画面を開き、ビルドしたコンテナイメージのTagを確認します。



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)



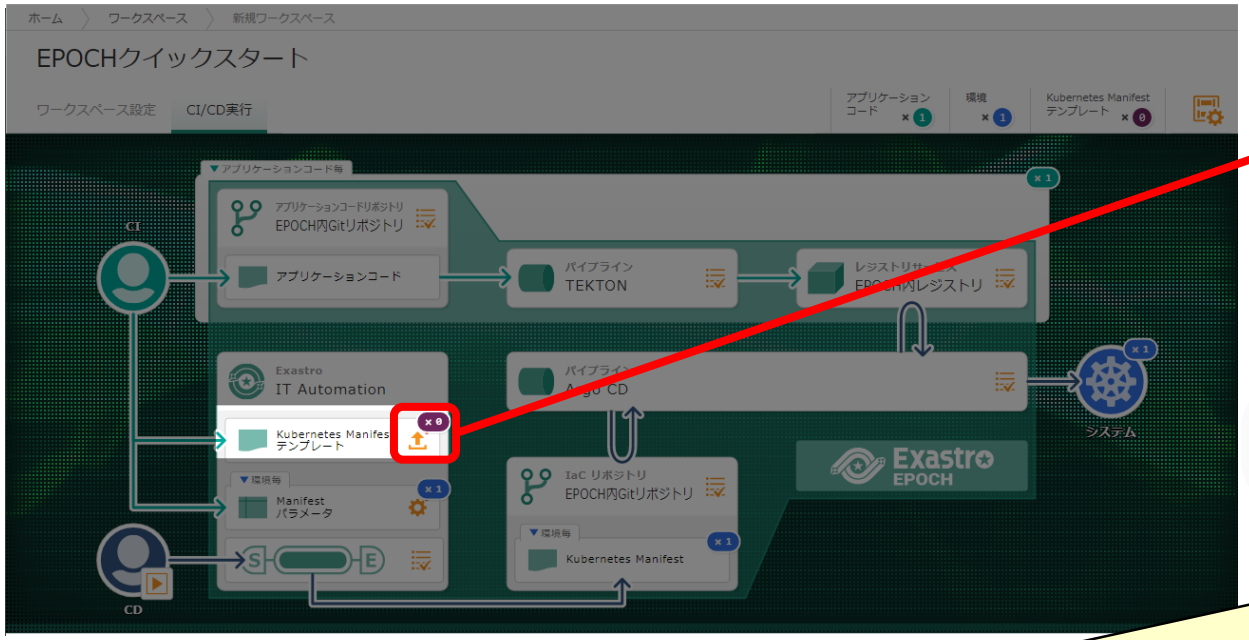
Manifestテンプレートアップロード

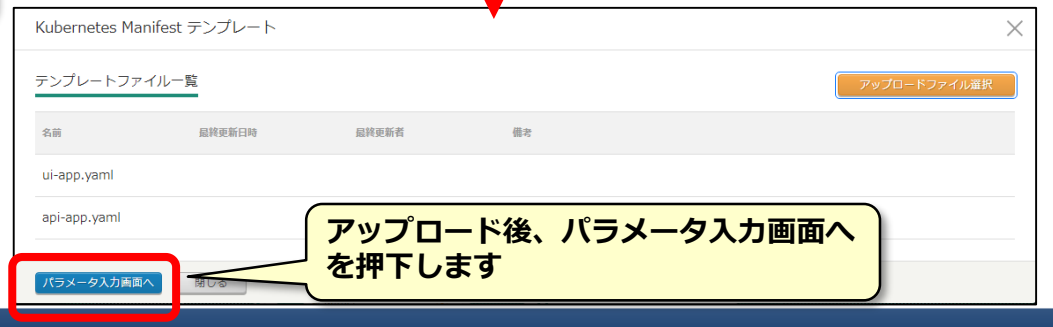
- ダウンロードしたManifestテンプレートをアップロードします。

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。

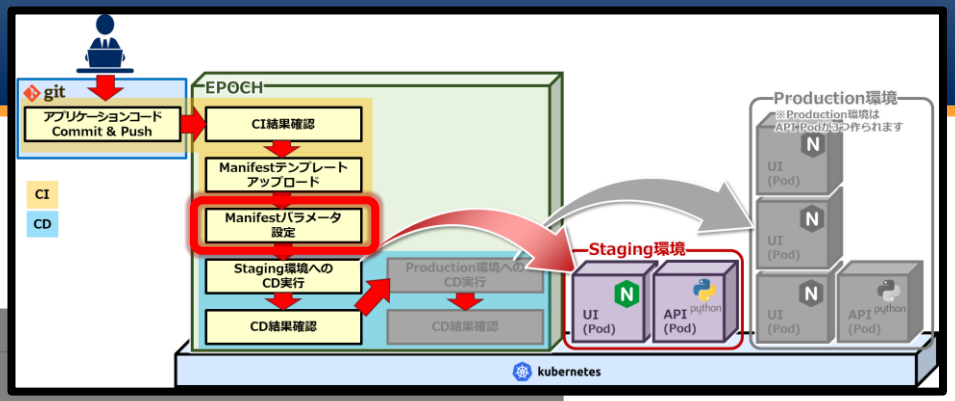




ファイル：api-app.yaml, ui-app.yamlを右の画面領域にドロップします

アップロード後、パラメータ入力画面へを押下します

4.5 1回目のCI/CDワークフロー手順(10/25)



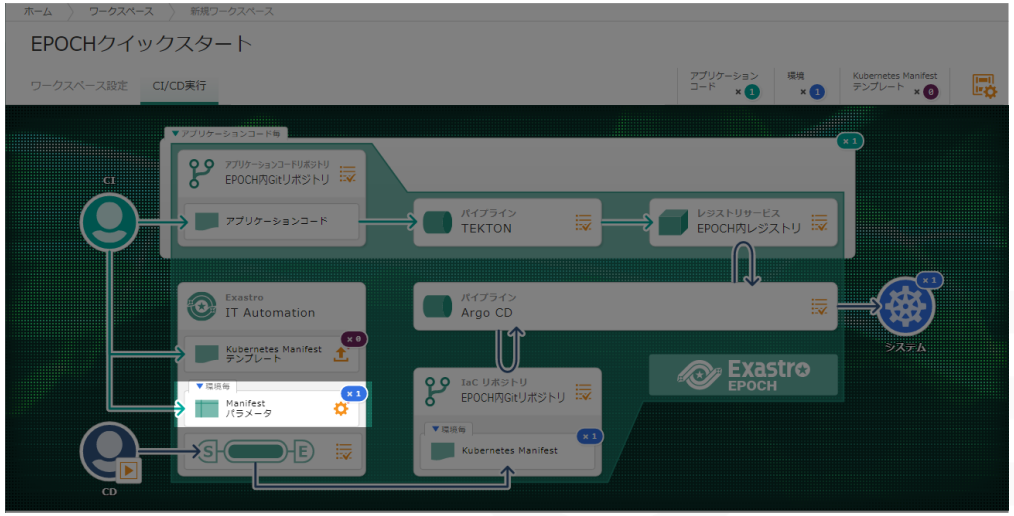
Manifestパラメータ

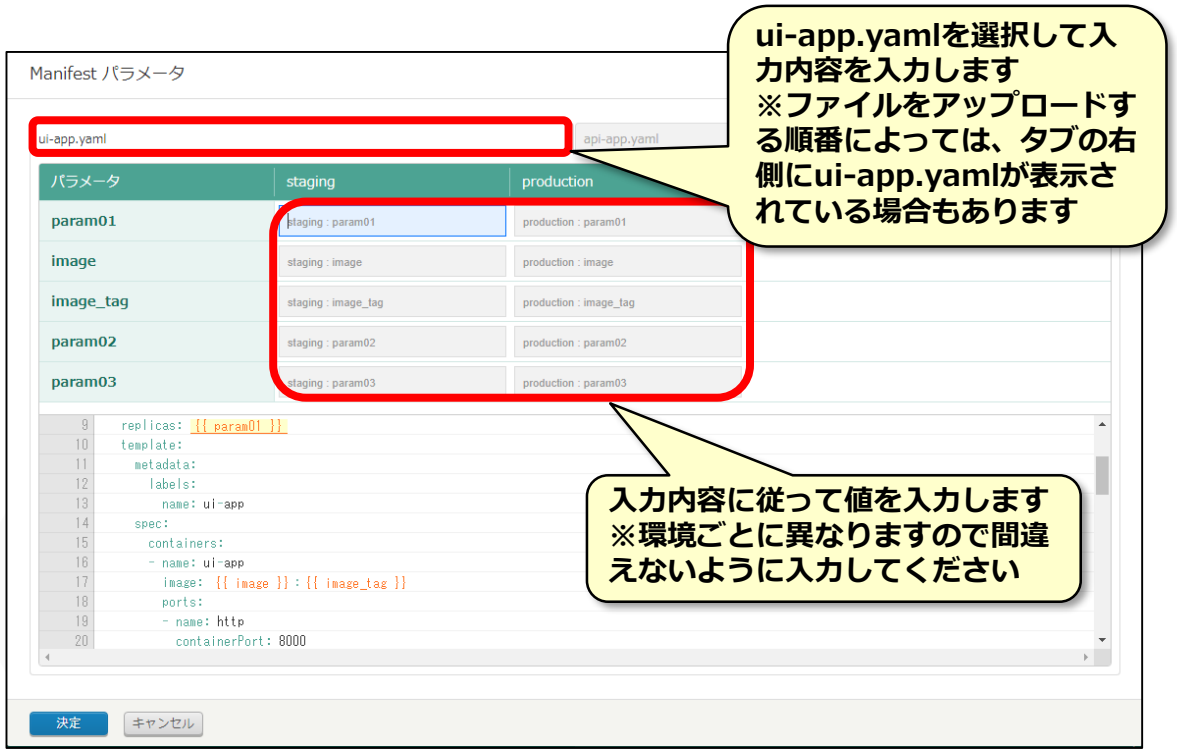
- Deployに必要なManifestパラメータを入力します。

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
{{ 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

※ui-app.yamlのimage_tagは事前にビルドしたコンテナイメ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。

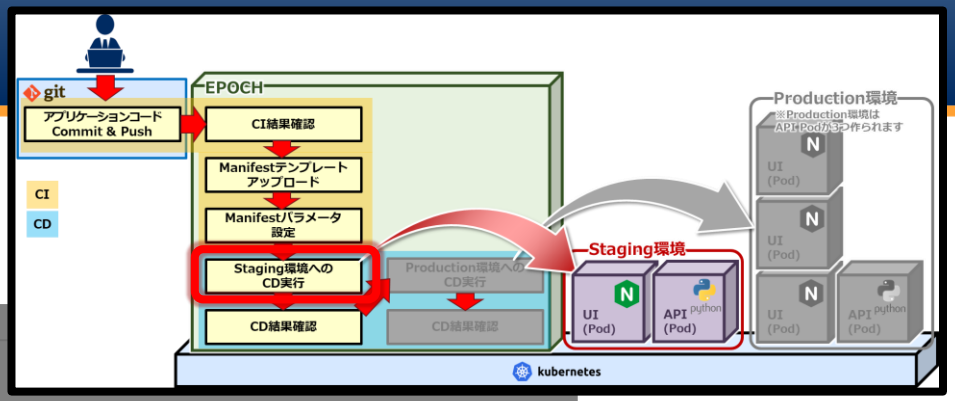
4.5 1回目のCI/CDワークフロー手順(12/25)

- 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)



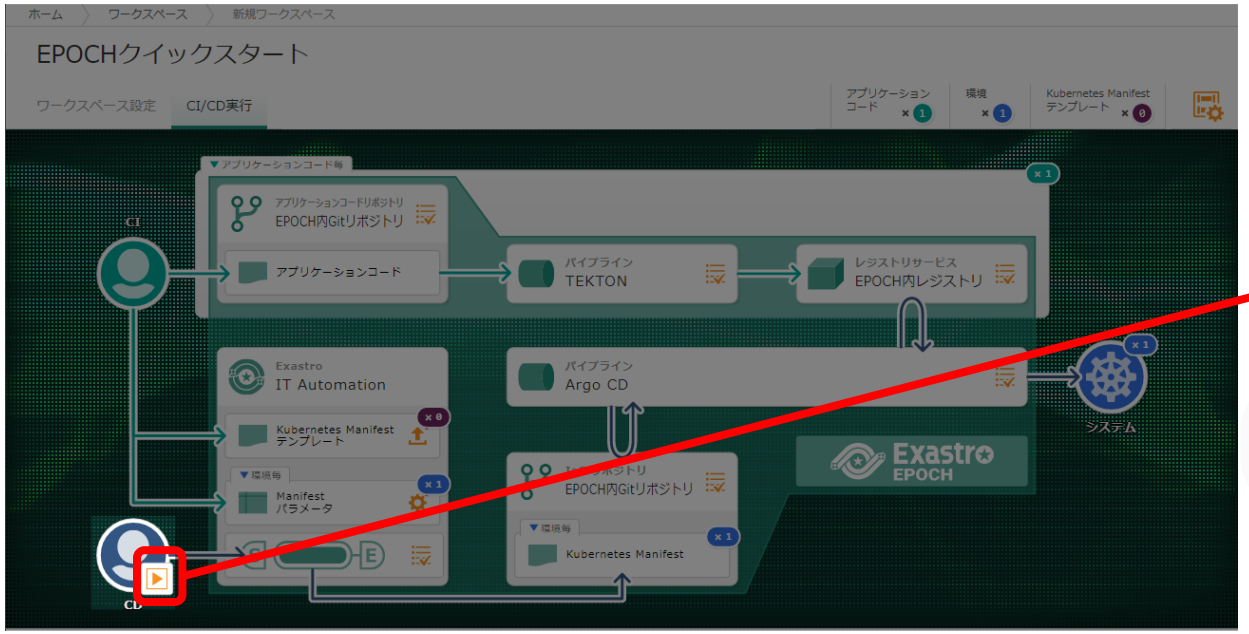
Staging環境へのDeploy実行

- 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)

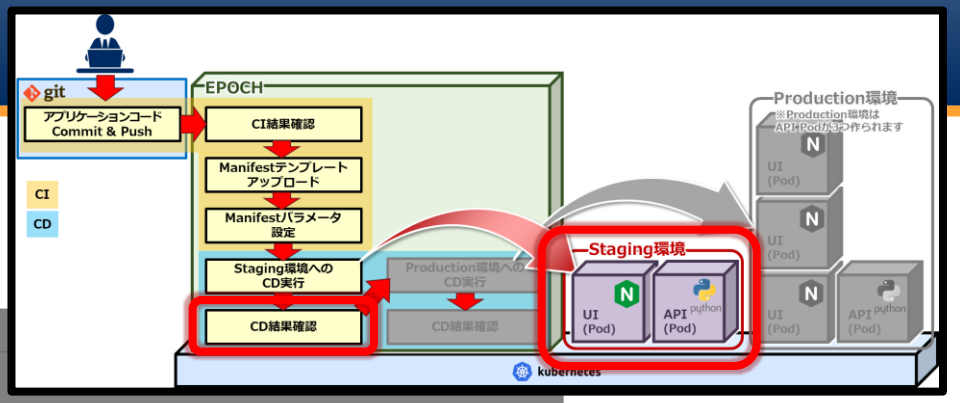
Staging環境のCD実行

- 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)

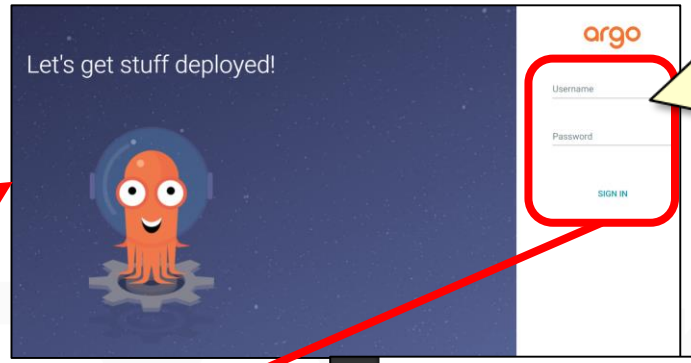
Manifestファイルの生成確認(Staging環境)

- 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。

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)

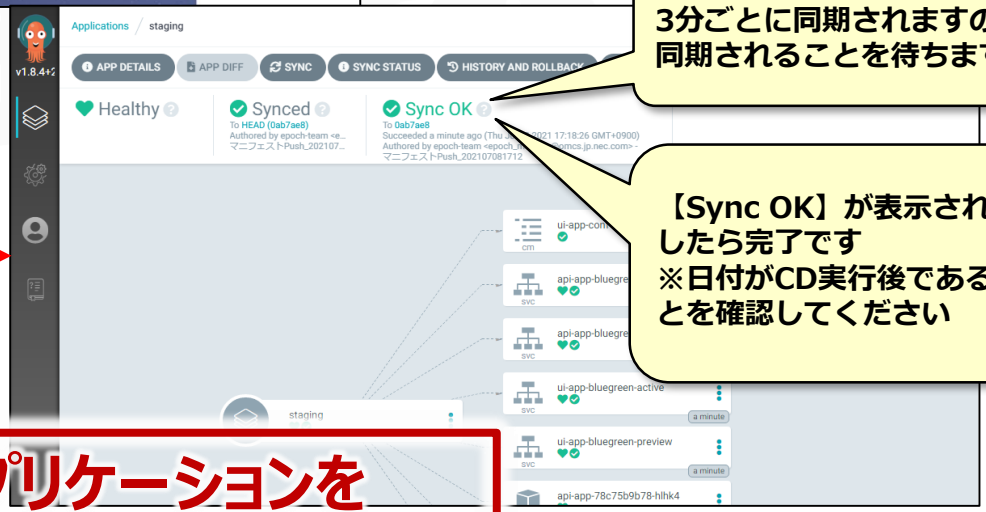
- ManifestがkubernetesにEPOCHに反映されるまでの状況を確認します。



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

stagingを選択



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

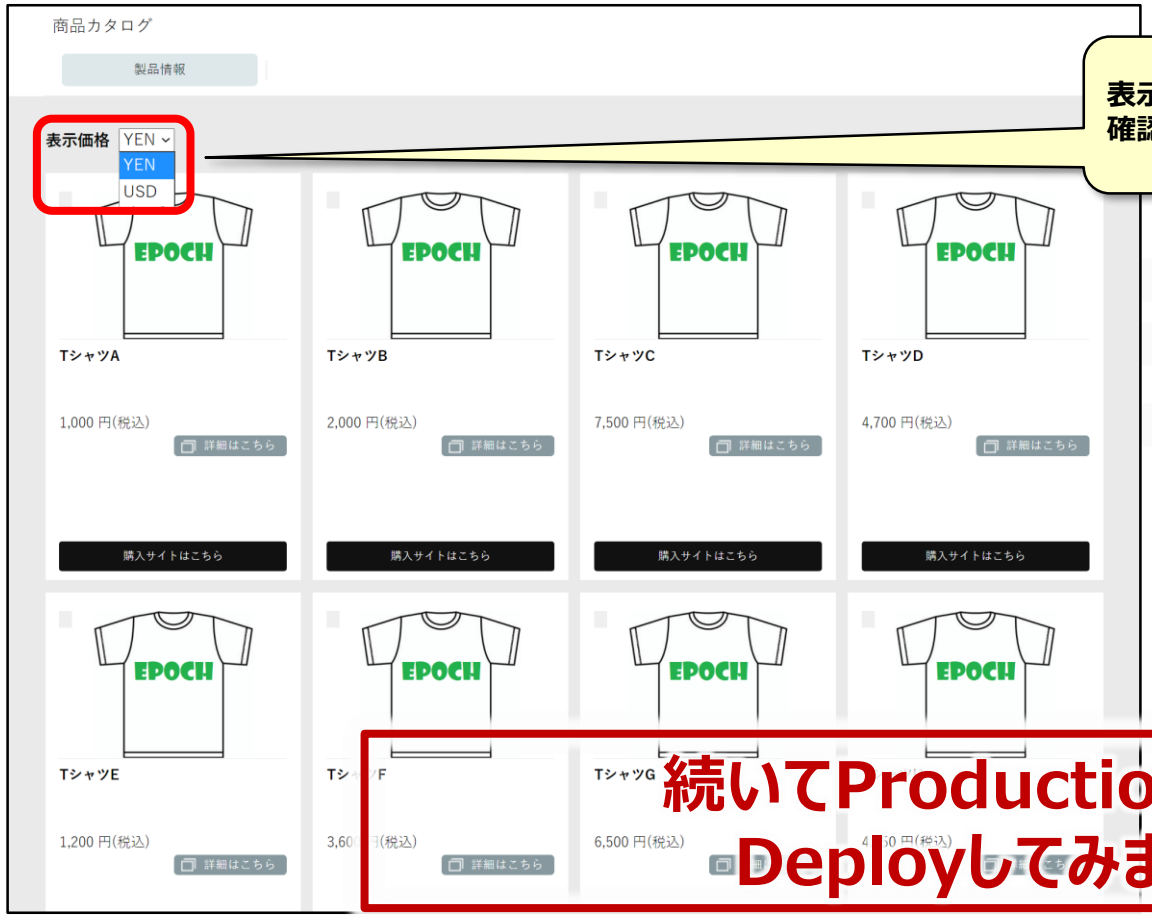
Deployされたサンプルアプリケーションを確認してみましょう

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



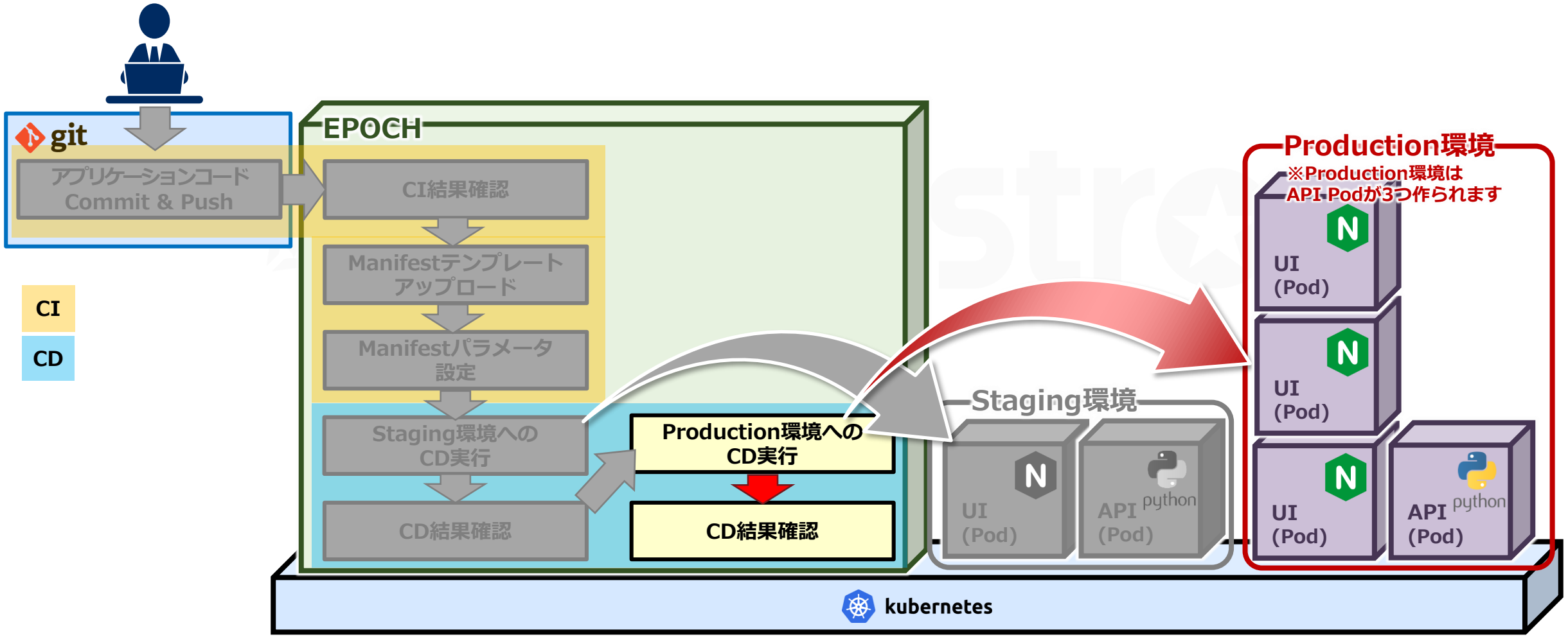
表示価格の選択欄に「YEN」「USD」が表示されていることを確認します。
確認後、「USD」を選択して、表示が切り替わることを確認します

続いてProduction環境へ
Deployしてみましょう

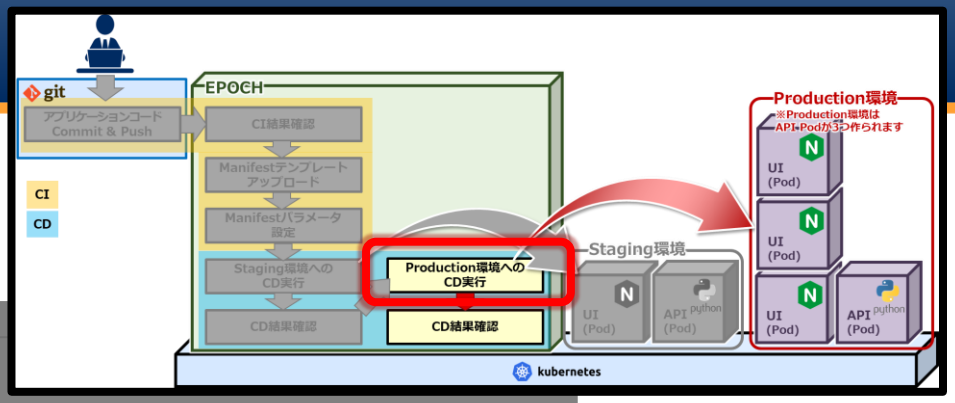
4.5 1回目のCI/CDワークフロー手順(19/25)

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。

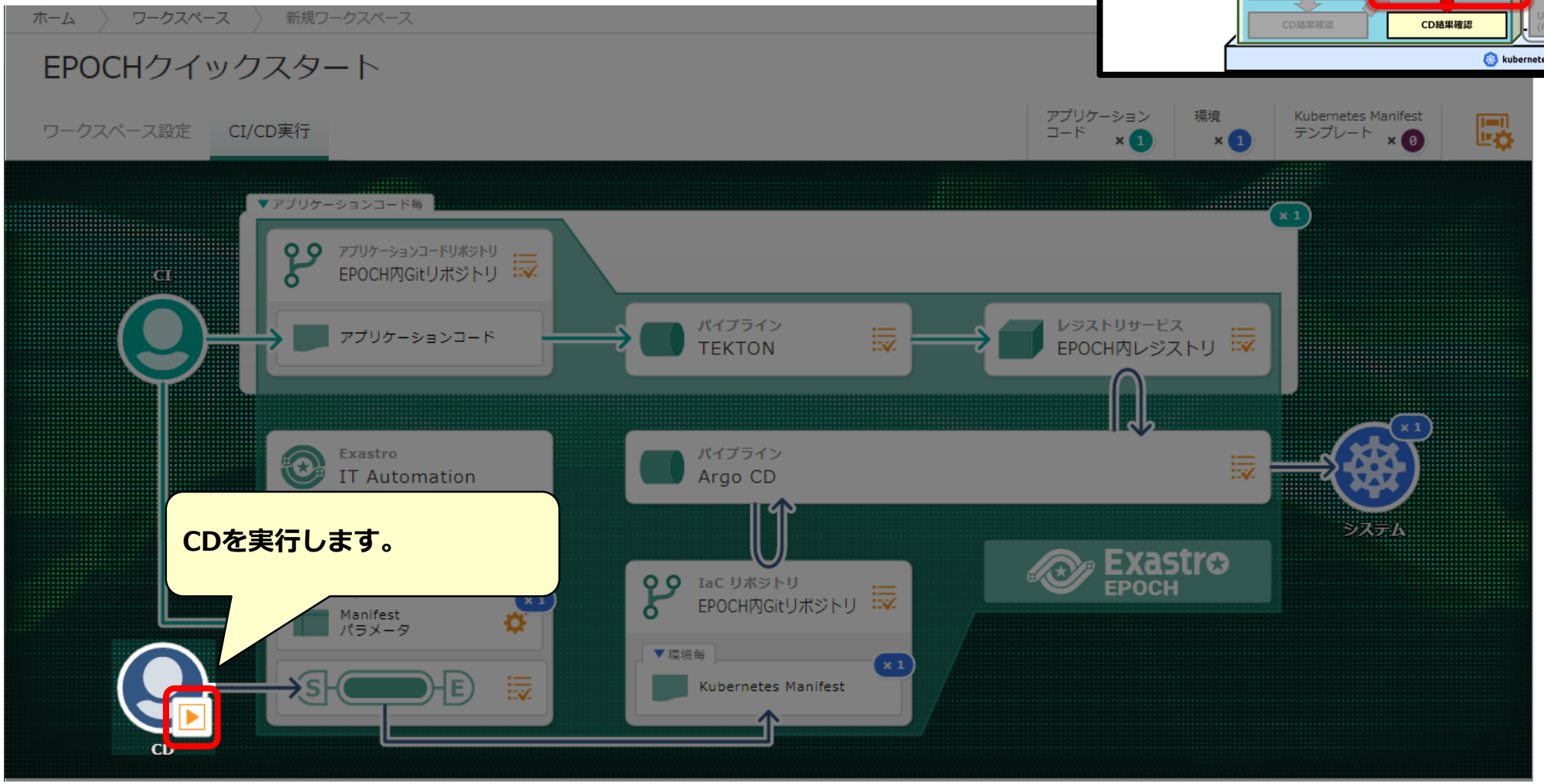


4.5 1回目のCI/CDワークフロー手順(20/25)



Production環境へのDeploy実行

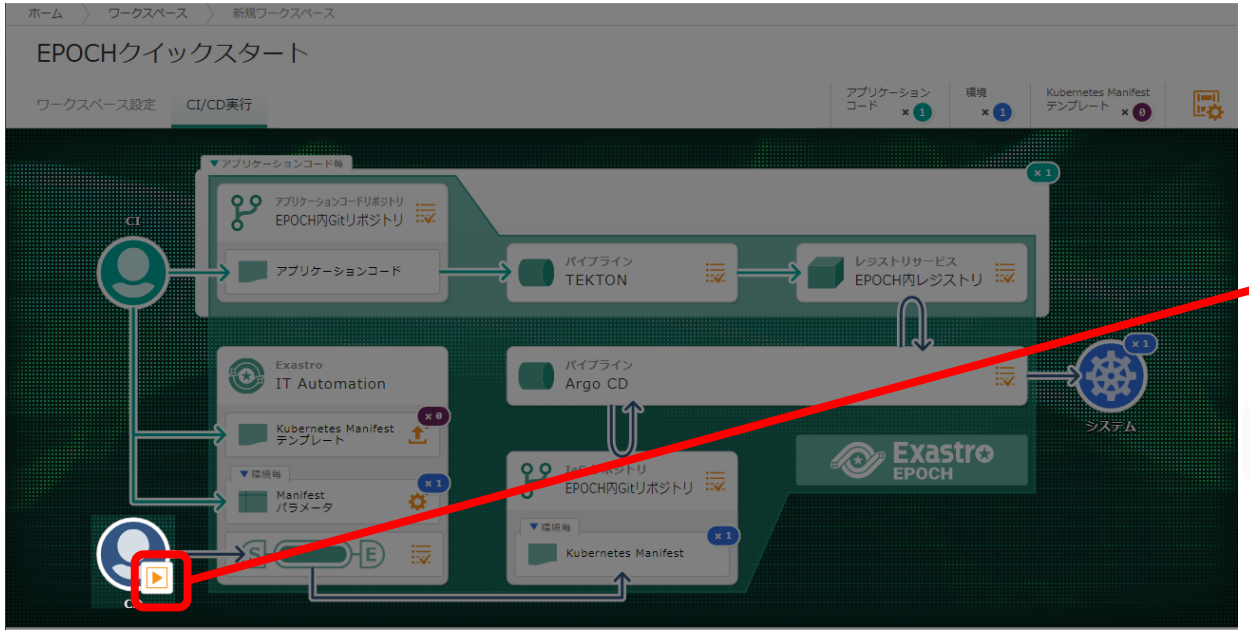
- CD実行で、Production環境へDeployを実行します。



4.5 1回目のCI/CDワークフロー手順(21/25)

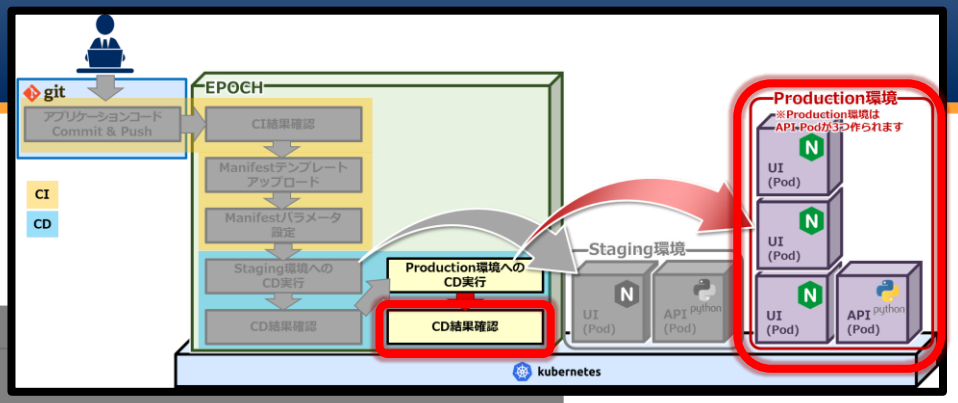
Production環境のCD実行

- Deploy先の環境を選択して実行します。



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)

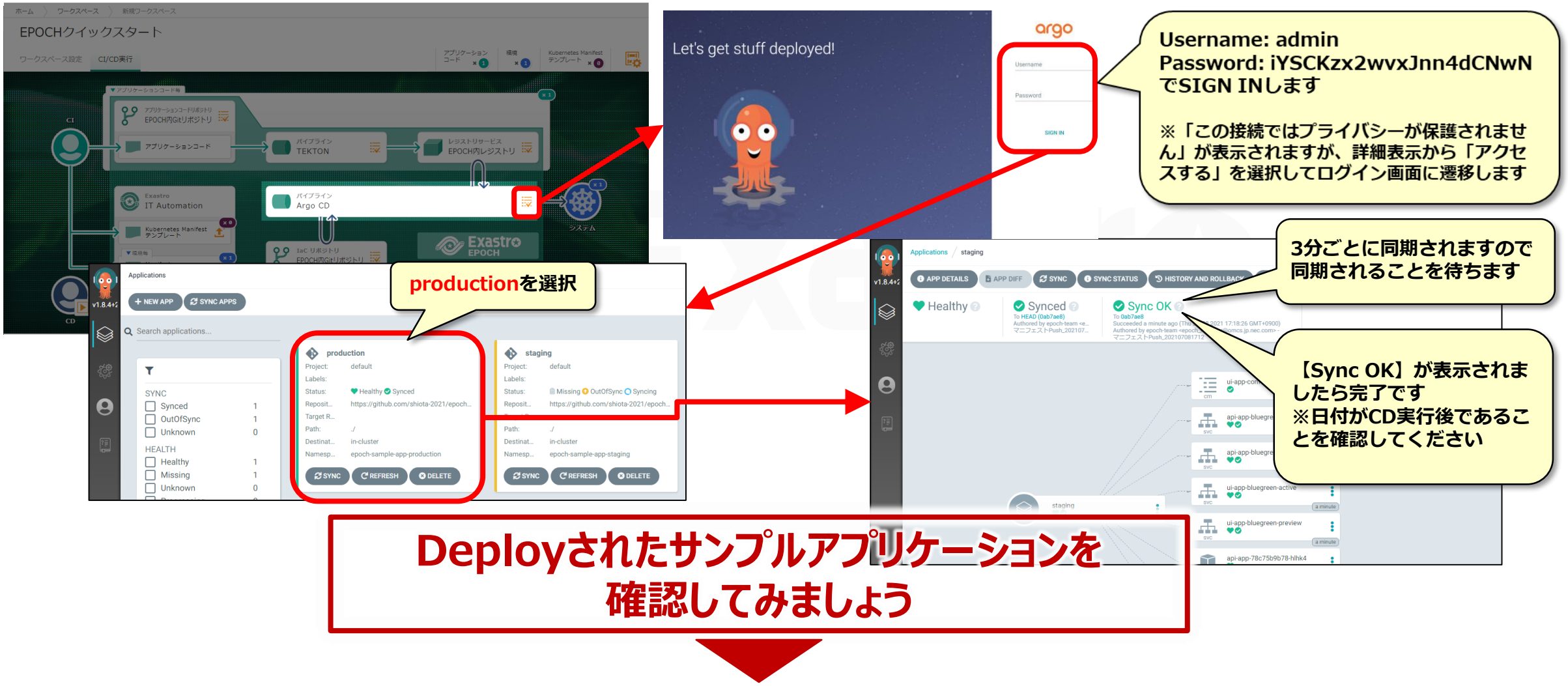
Manifestファイルの生成確認(Production環境)

- 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。

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EPOCHに反映されるまでの状況を確認します。



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

Staging環境と同様に画面が表示されることを確認します

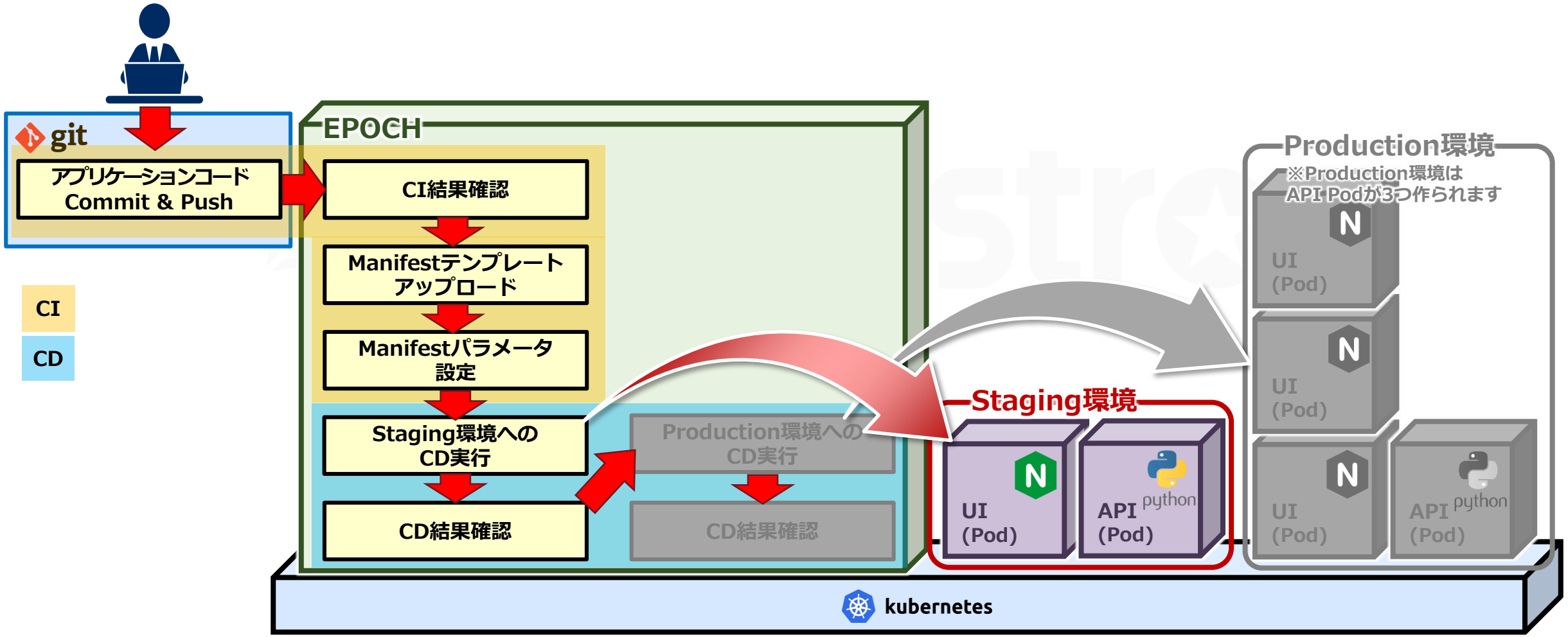
1回目のCI/CDワークフローはここで完了です
続いて実際にアプリケーションコードを修正し
Deployされるまでの2回目のCI/CDワークフロー手順
を実行してみましょう！

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  }
```

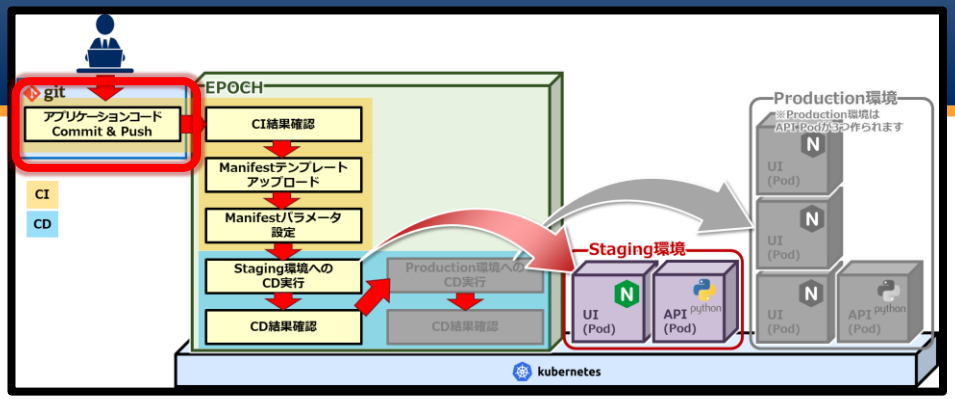
9~12行目を追加

- 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

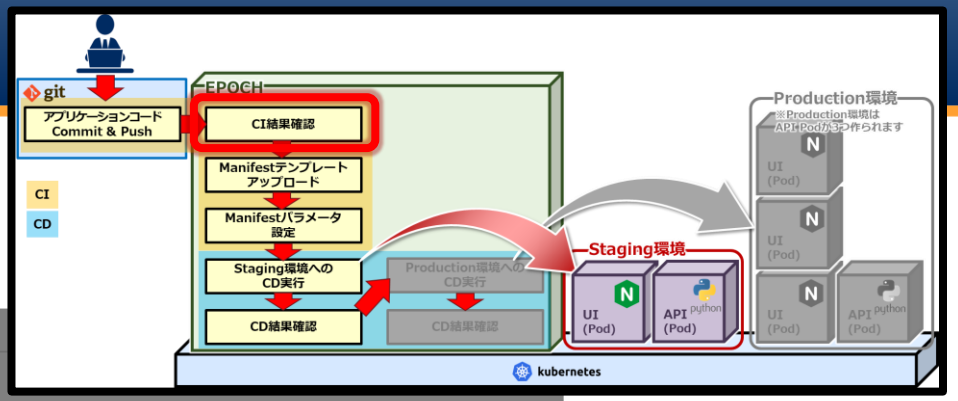
修正した内容をCommit&Pushします。

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)



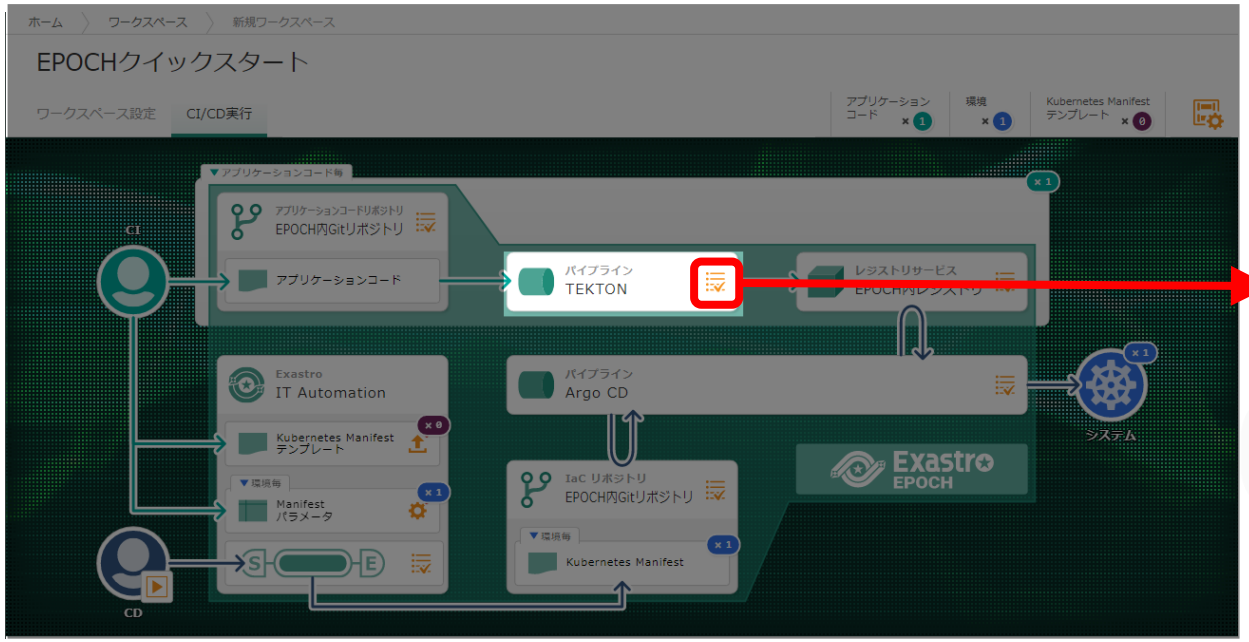
CI結果確認

- 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。



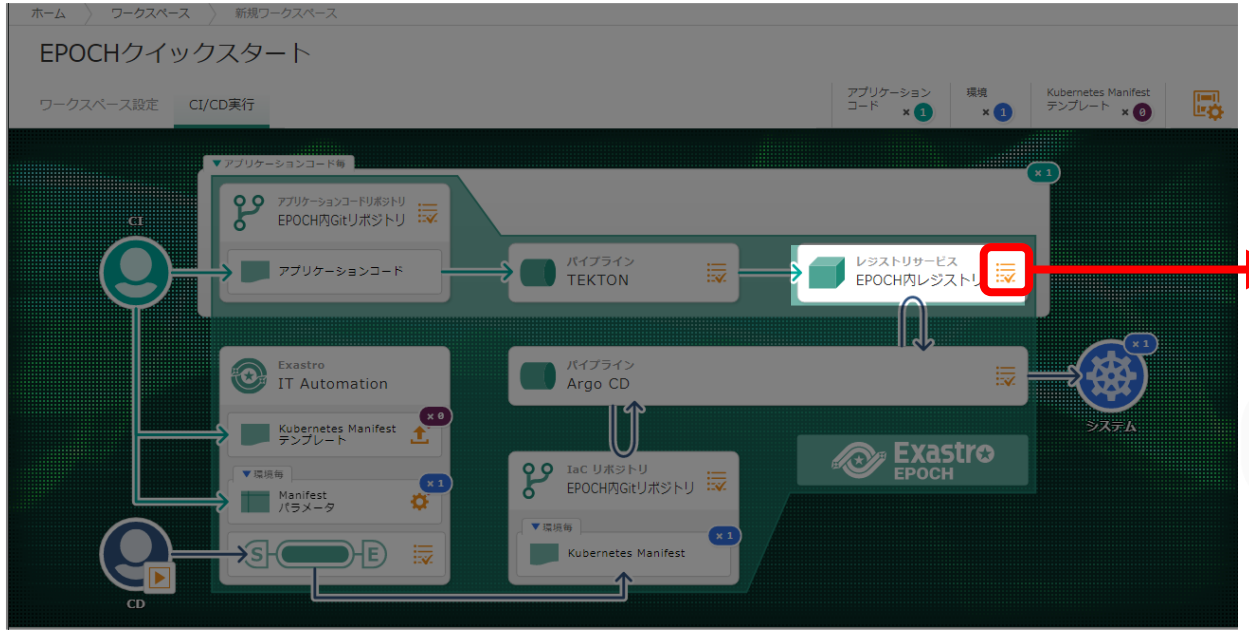
CIパイプラインの動作は、TEKTONダッシュボードにて確認することができます。Nameをクリックすることにより、より詳細なCIパイプラインの内容を確認することができます

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

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、ビルドしたコンテナイメージのTagを確認します。



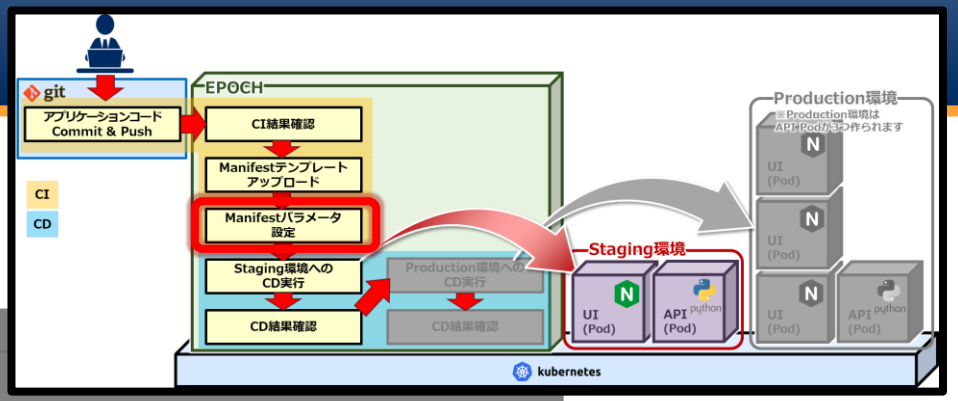
Dockerhubの画面でepoch-sample-apiのTagを確認します

※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

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

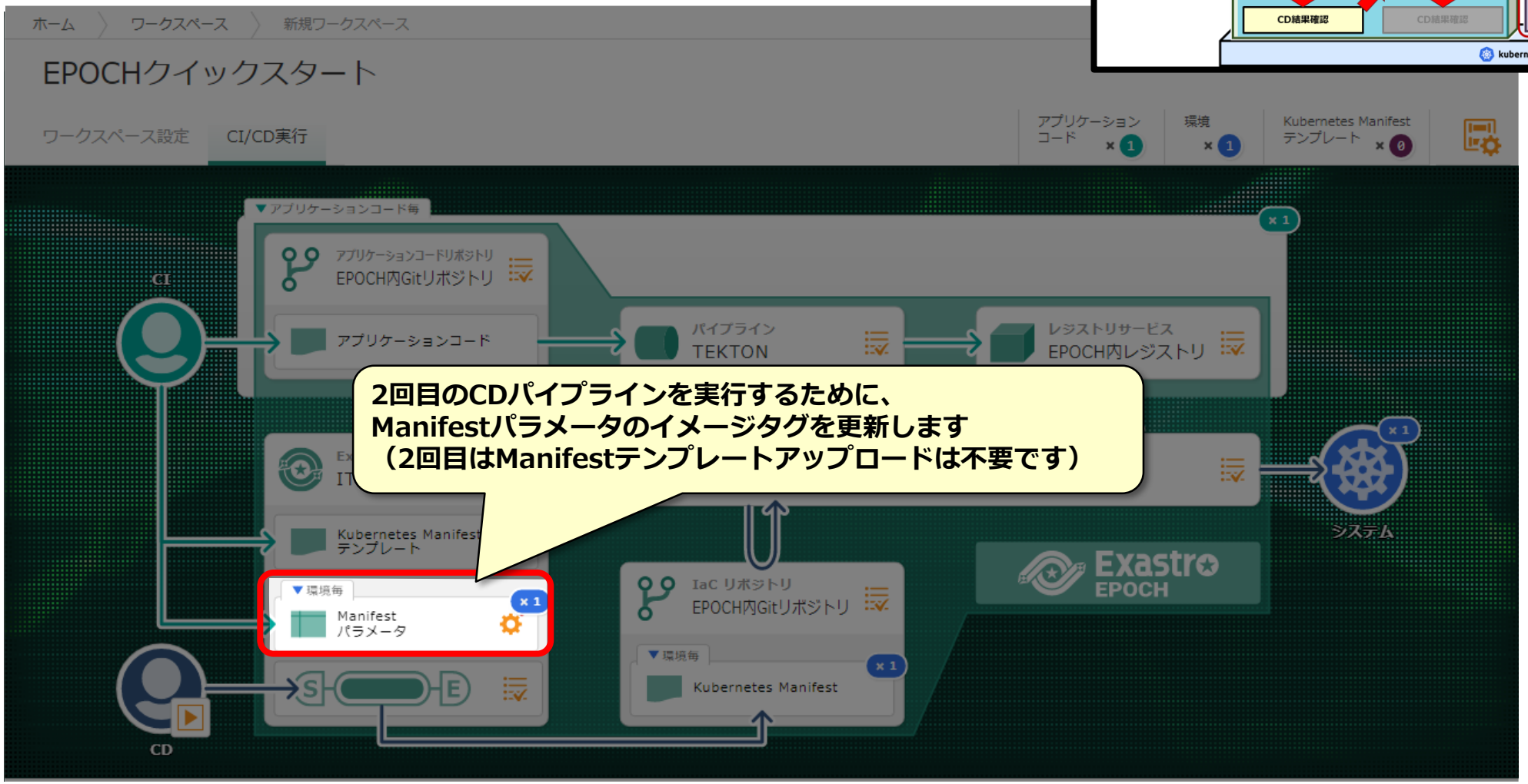
※今後、パイプラインで生成されたimage_tagは選択できるように変更する予定です

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

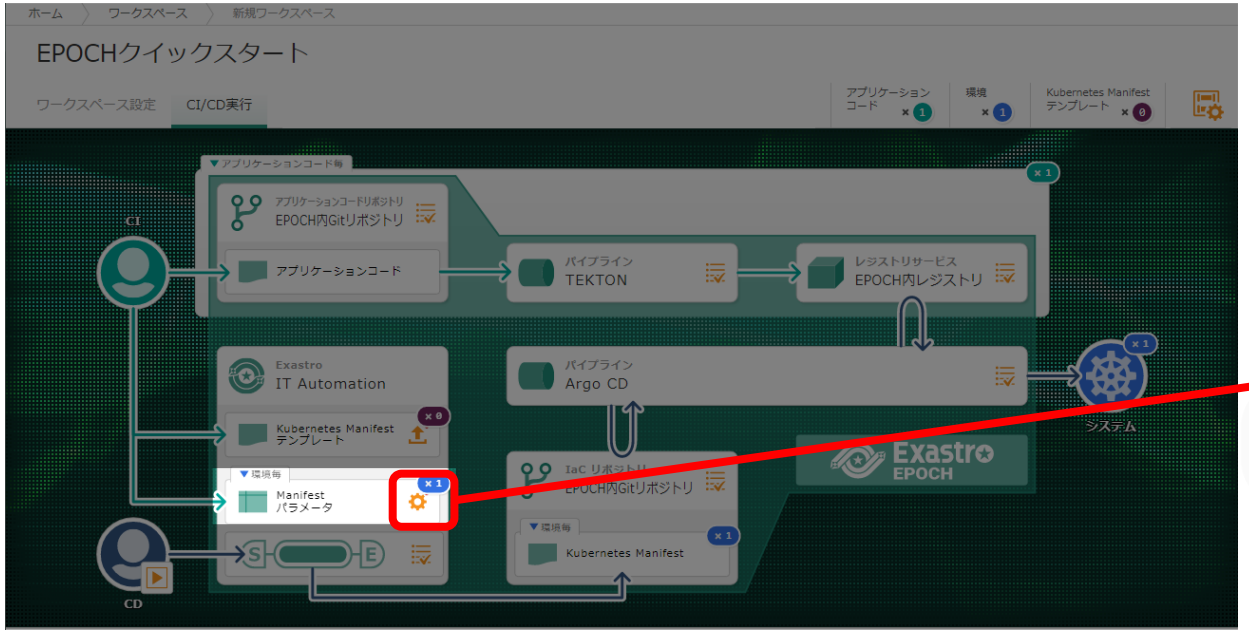
- Manifestパラメータの設定のイメージタグを更新します。

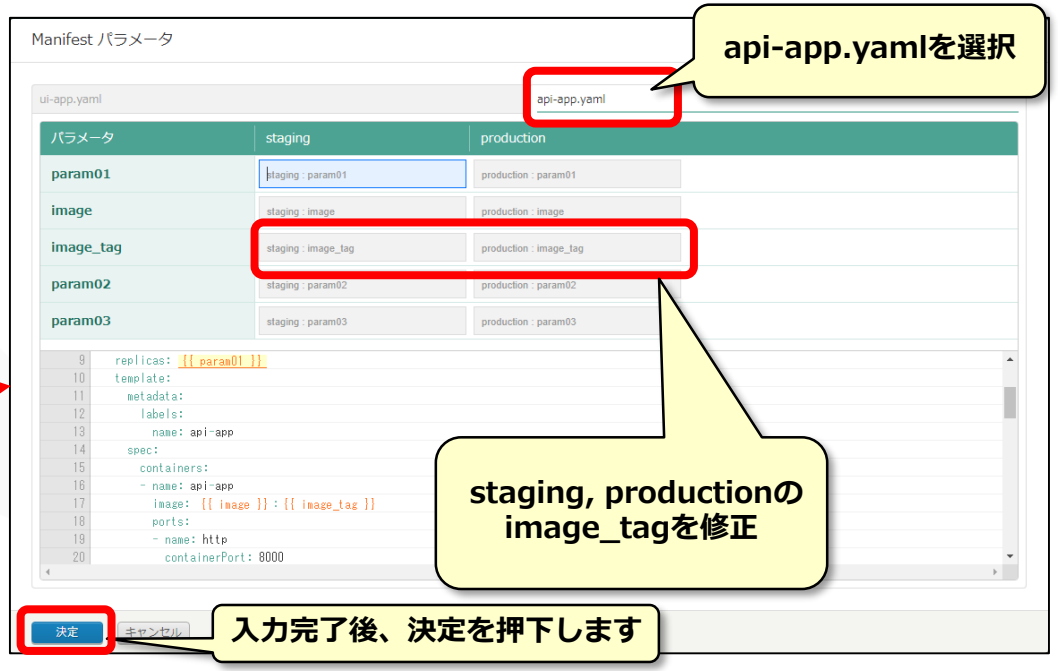


4.6 2回目のCI/CDワークフロー手順(8/21)

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。





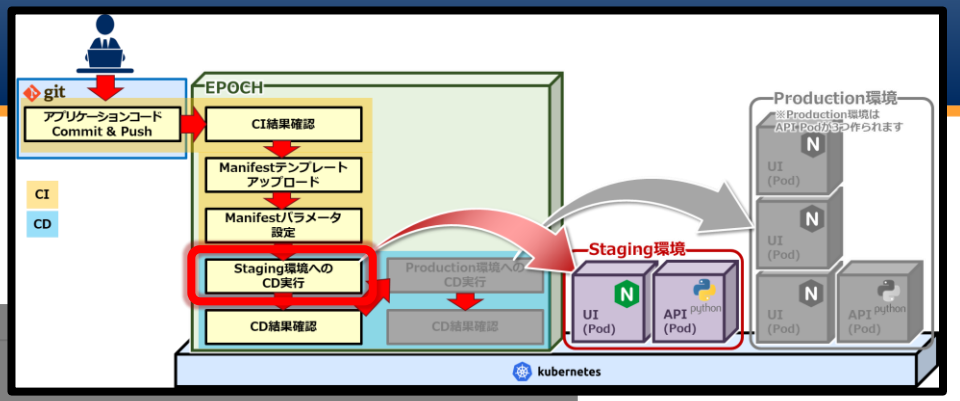
api-app.yaml

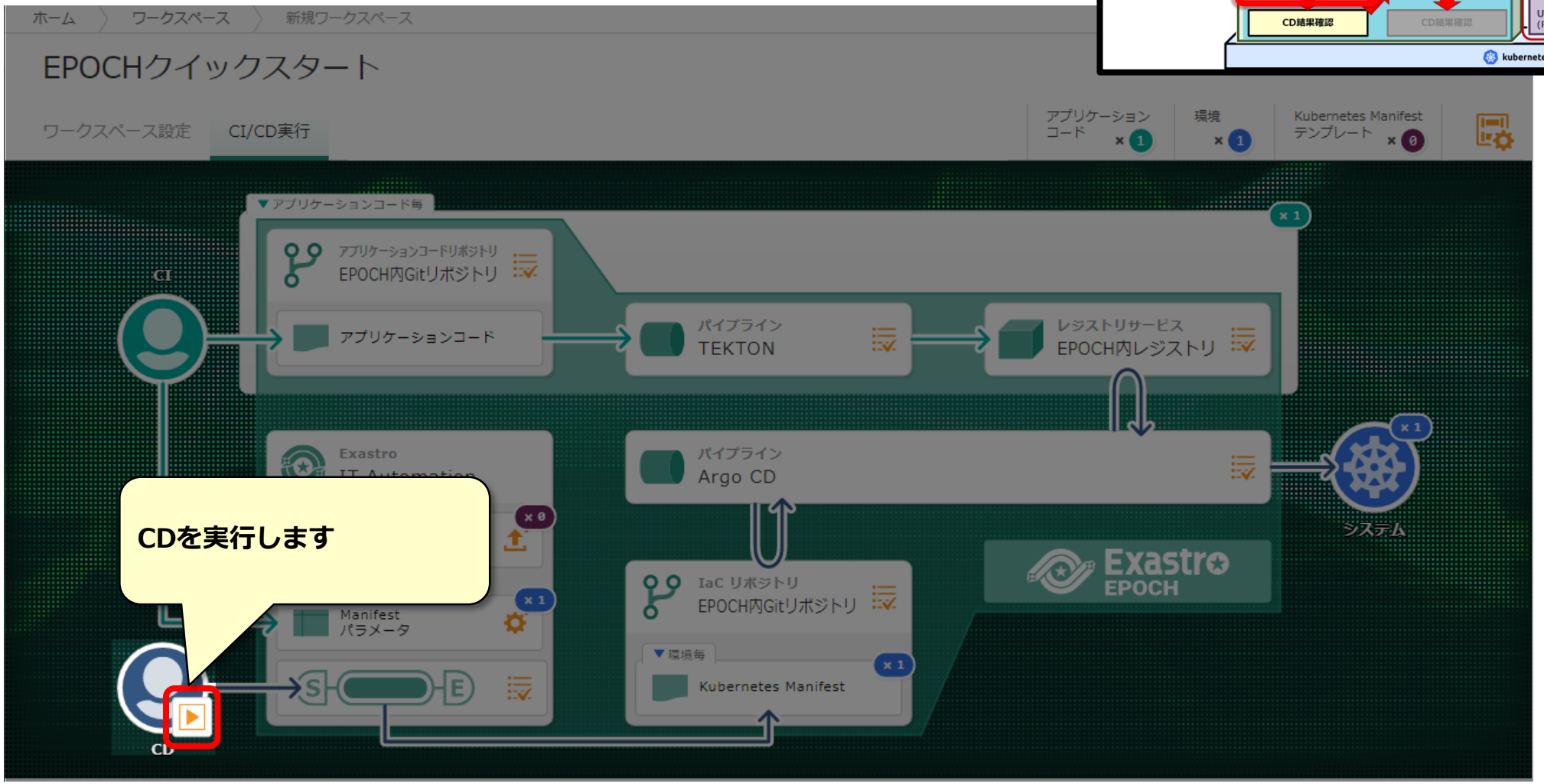
項目	入力内容(staging)	入力内容(production)	説明
{{ image_tag }}	[レジストリサービスで確認した最新のイメージのタグ名]	[レジストリサービスで確認した最新のイメージのタグ名]	コンテナイメージのタグ

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。

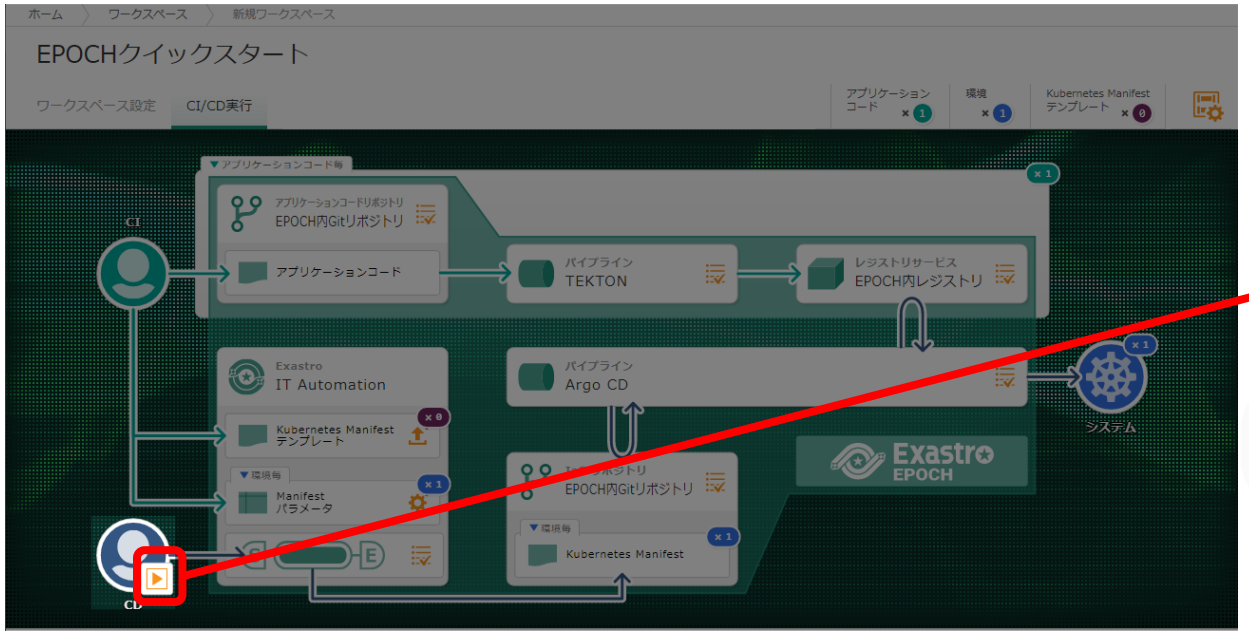




4.6 2回目のCI/CDワークフロー手順(10/21)

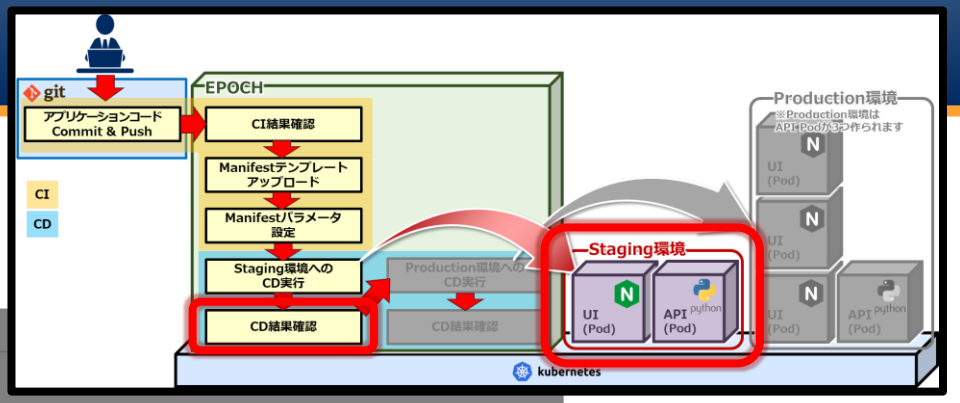
CD実行指定

- Deploy先の環境を選択してDeployを実行します。



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)

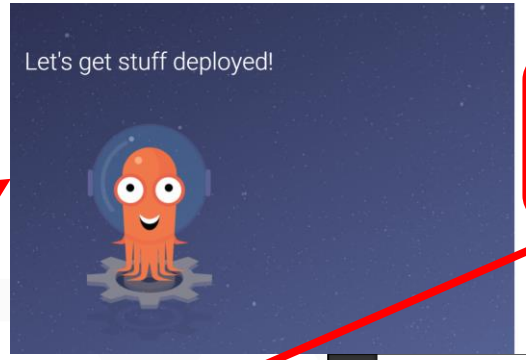
Manifestファイルの生成確認(Staging環境)

- 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。

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)

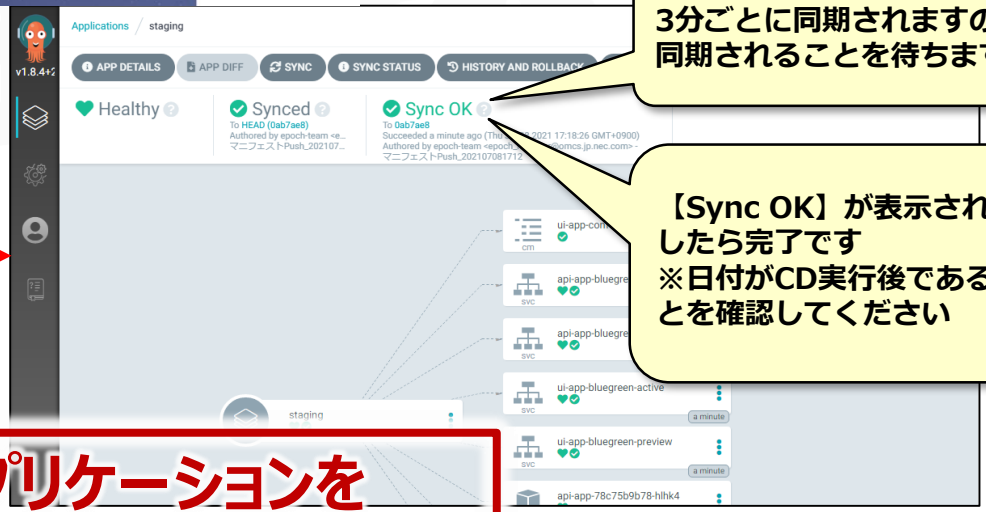
- ManifestがkubernetesにEPOCHに反映されるまでの状況を確認します。



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

stagingを選択



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

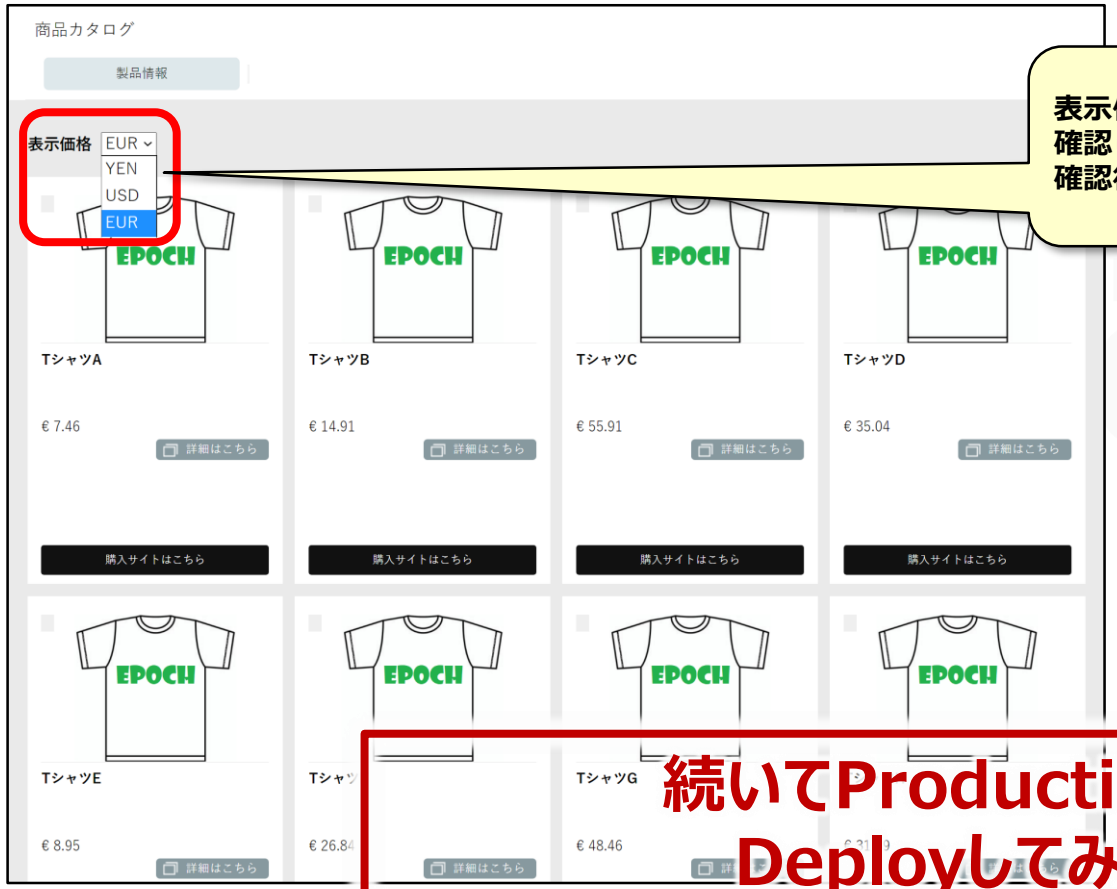
Deployされたサンプルアプリケーションを確認してみましょう

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



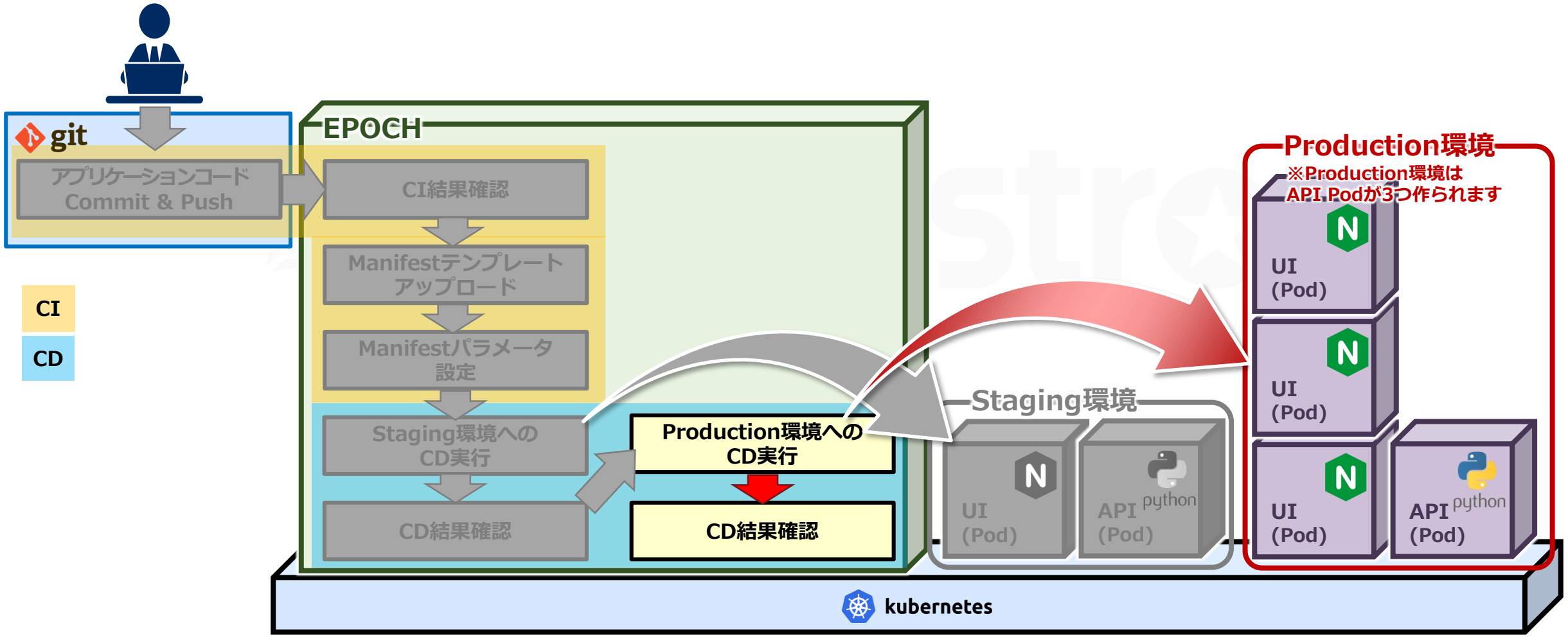
表示価格の選択欄に「YEN」「USD」「EUR」が表示されていることを確認します。
確認後、「EUR」を選択して、表示が切り替わることを確認します

続いてProduction環境へ
Deployしてみましょう

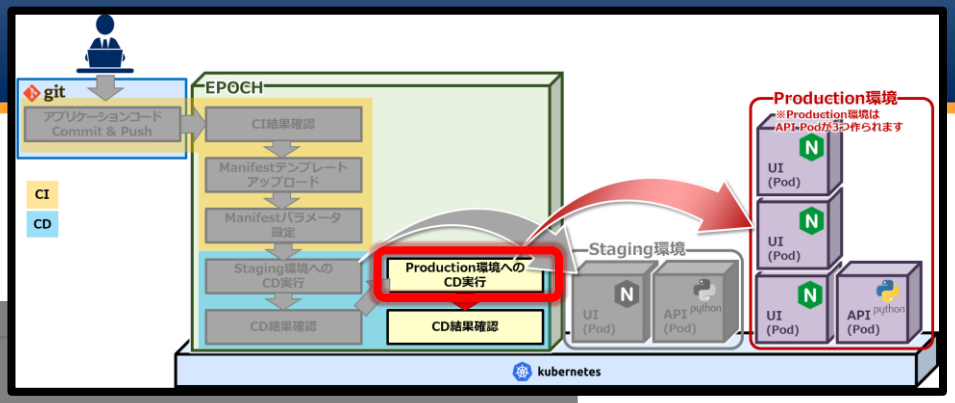
4.6 2回目のCI/CDワークフロー手順(15/21)

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。

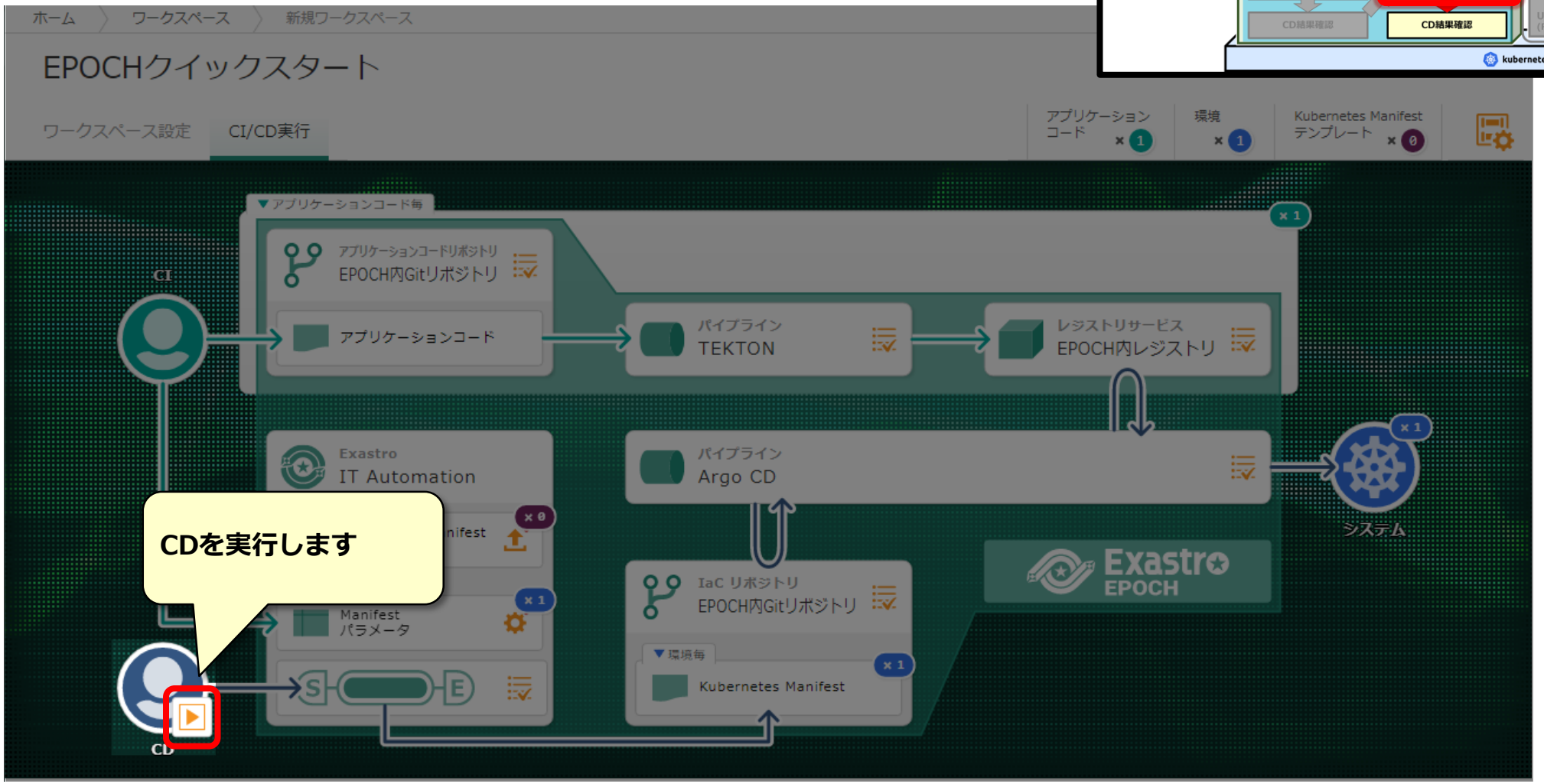


4.6 2回目のCI/CDワークフロー手順(16/21)



Production環境へのCD実行

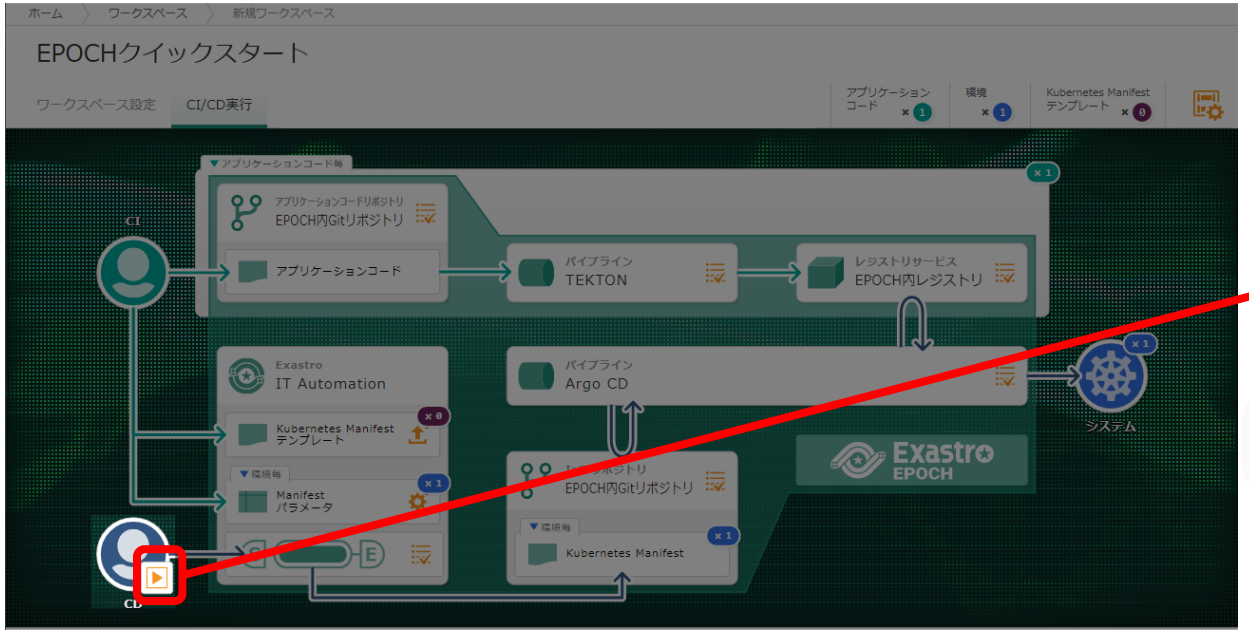
- CD実行で、実際にProduction環境へDeployを実行します。



4.6 2回目のCI/CDワークフロー手順(17/21)

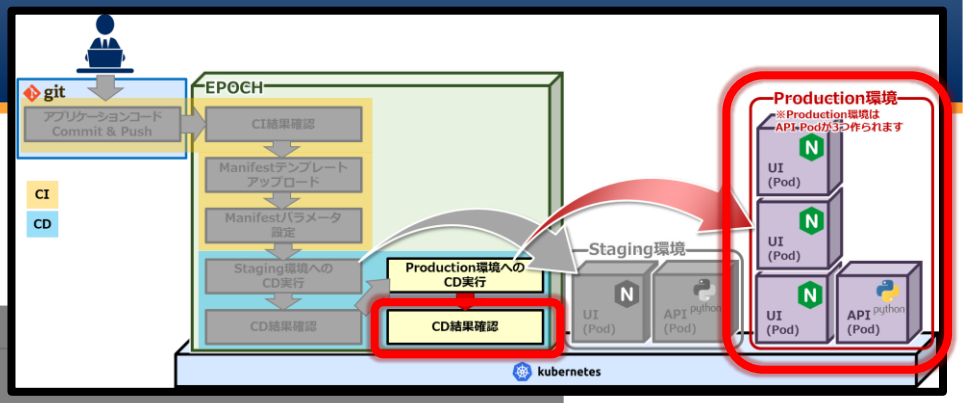
CD実行指定

- 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)

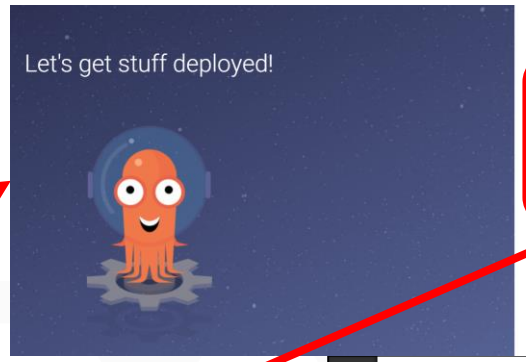
Manifestファイルの生成確認(Production環境)

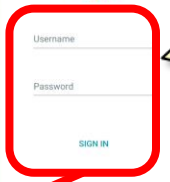
- Exastro IT-Automationから、IaCリポジトリへManifestファイルを登録するまでの状況を確認します。

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)

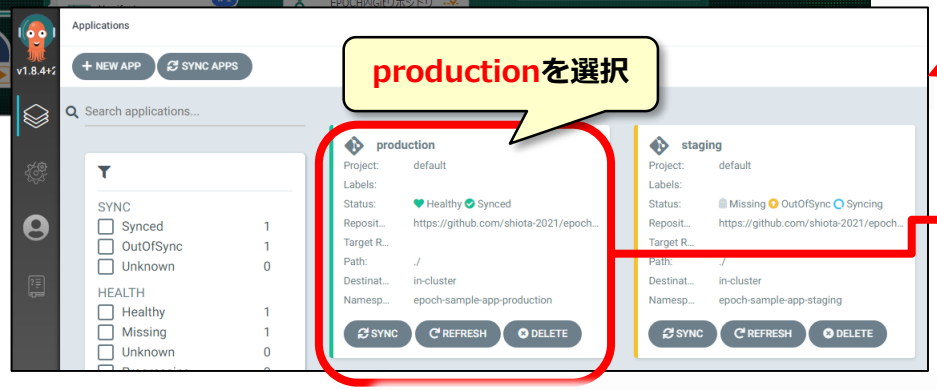
- ManifestがkubernetesにEPOCHに反映されるまでの状況を確認します。



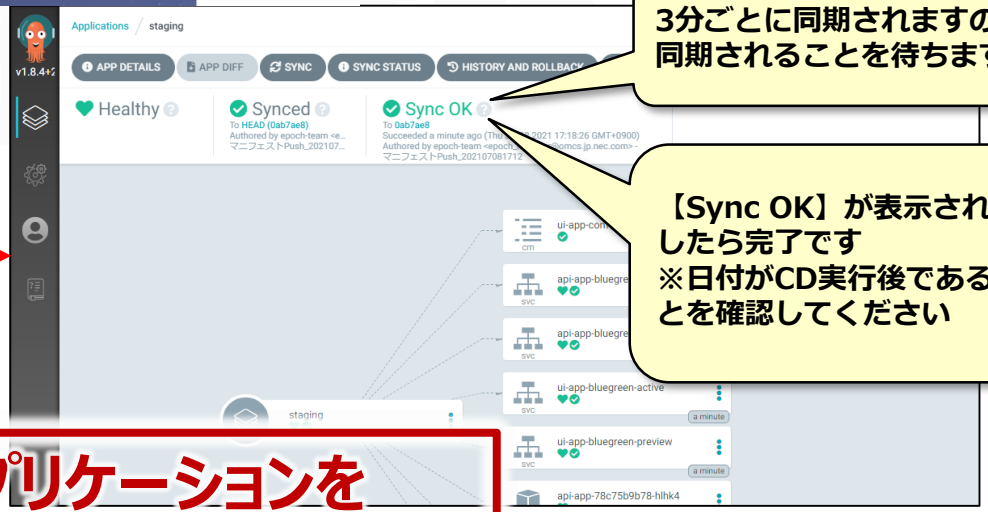


Username: admin
Password: iYSCKzx2wvxJnn4dCNwN
でSIGN INします

※「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



productionを選択



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

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(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

Staging環境と同様に画面が表示されることを確認します

以上でチュートリアルは終了になります。
お疲れ様でした。

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
{{ image }}	コンテナイメージ
{{ image_tag }}	コンテナイメージのタグ
{{ param01 }}	ユーザが自由に使用できる固定の変数名01
〜	
{{ param20 }}	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