



**Exastro**

# EPOCH Quick Start

Version 0.1.0

Exastro developer

# Table of contents

1. Introduction  
1.1 QuickStart guide  
1.2 Quickstart PC Environment

2. Install  
2.1 Install EPOCH  
2.2 Prepare repositories  
2.3 Manifest template files

3. Create Workspace  
3.1 Workspace  
3.2 CI/CD  
3.3 CI/CD in EPOCH  
3.4 Start EPOCH  
3.5 Create Workspace

4. Tutorial  
4.1 Tutorial overview  
4.2 Sample application configuration  
4.3 Tutorial (CI/CD Workflow)  
4.4 Manifest template files  
4.5 CI/CD Workflow No.1  
4.6 CI/CD Workflow No.2

5. Appendix  
5.1 Restrictions



# 1. Introduction

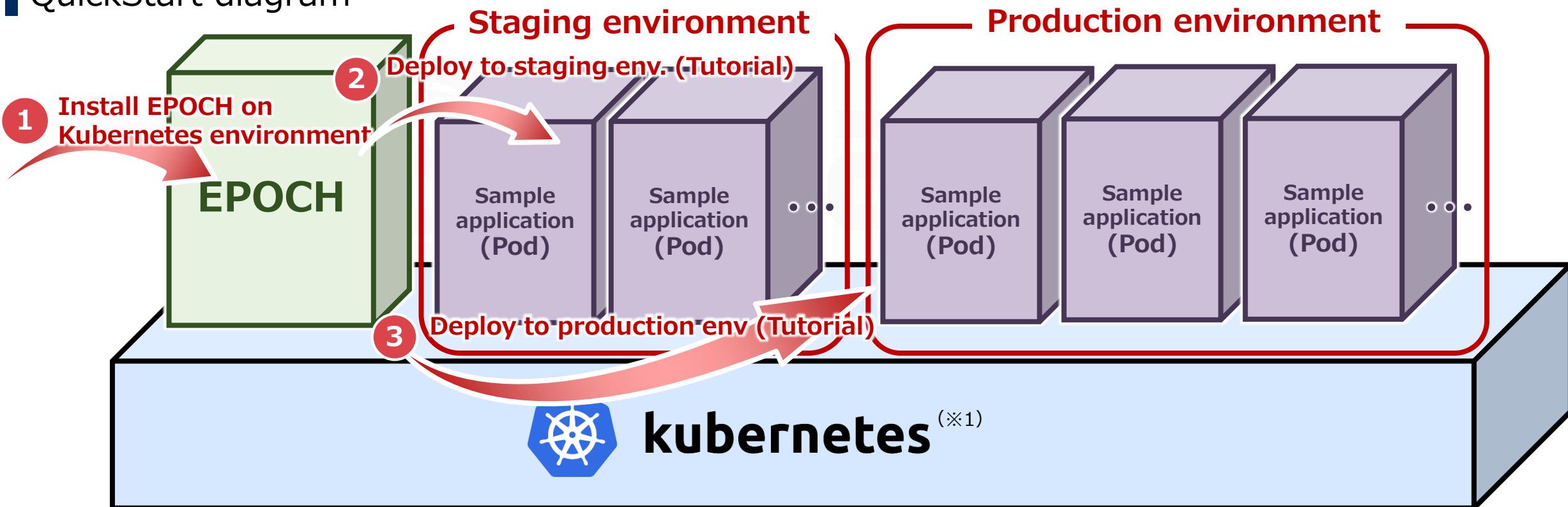


# 1.1 Quickstart guide

## About this document

This document aims to introduce the reader to Exastro EPOCH(Hereafter written as “EPOCH”), how to install it while also guiding the reader through a simple tutorial.

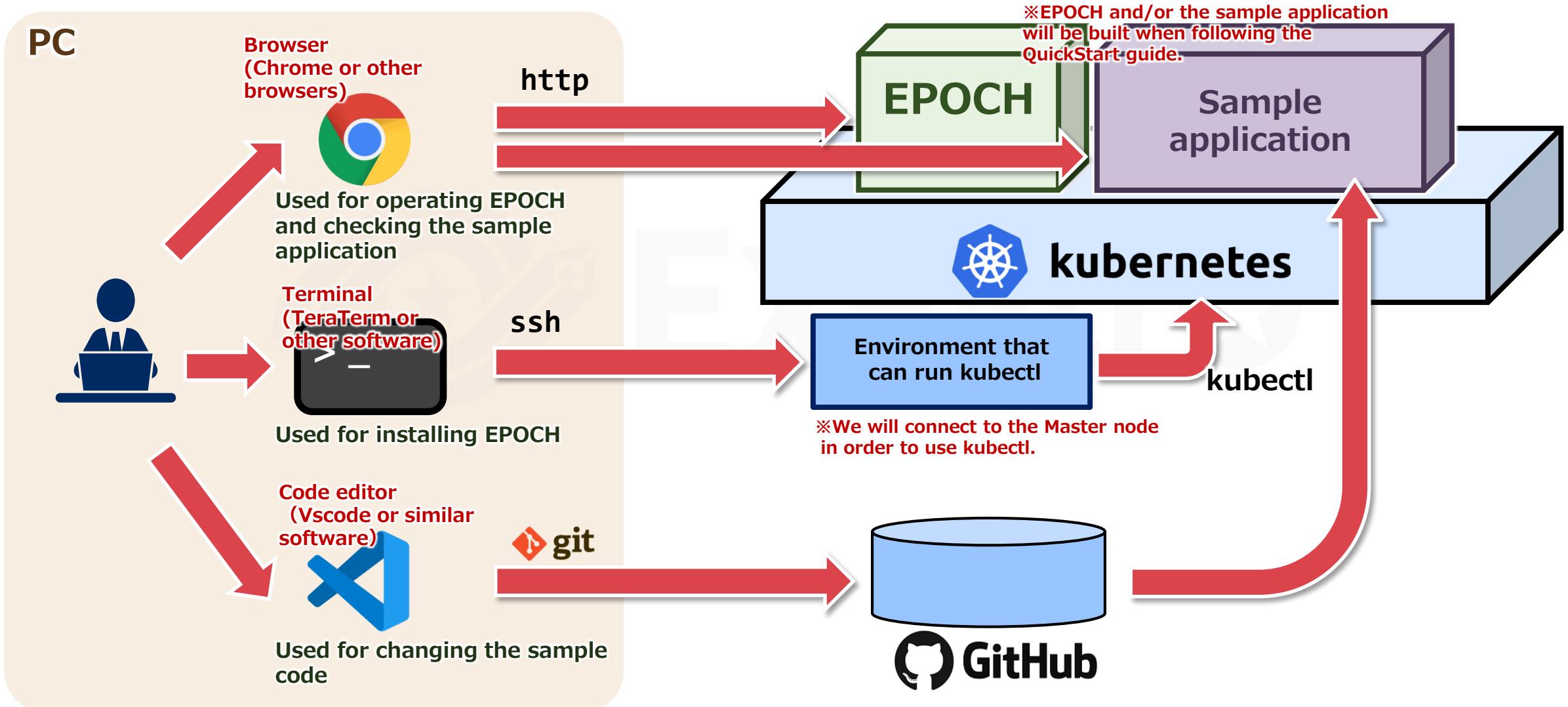
## QuickStart diagram



(※1)In order to keep the tutorial simple, everything will be constructed on a single Kubernetes cluster.

## 1.2 Quickstart PC environment

The following figure illustrates the PC software necessary for following this guide.



## 2. Install

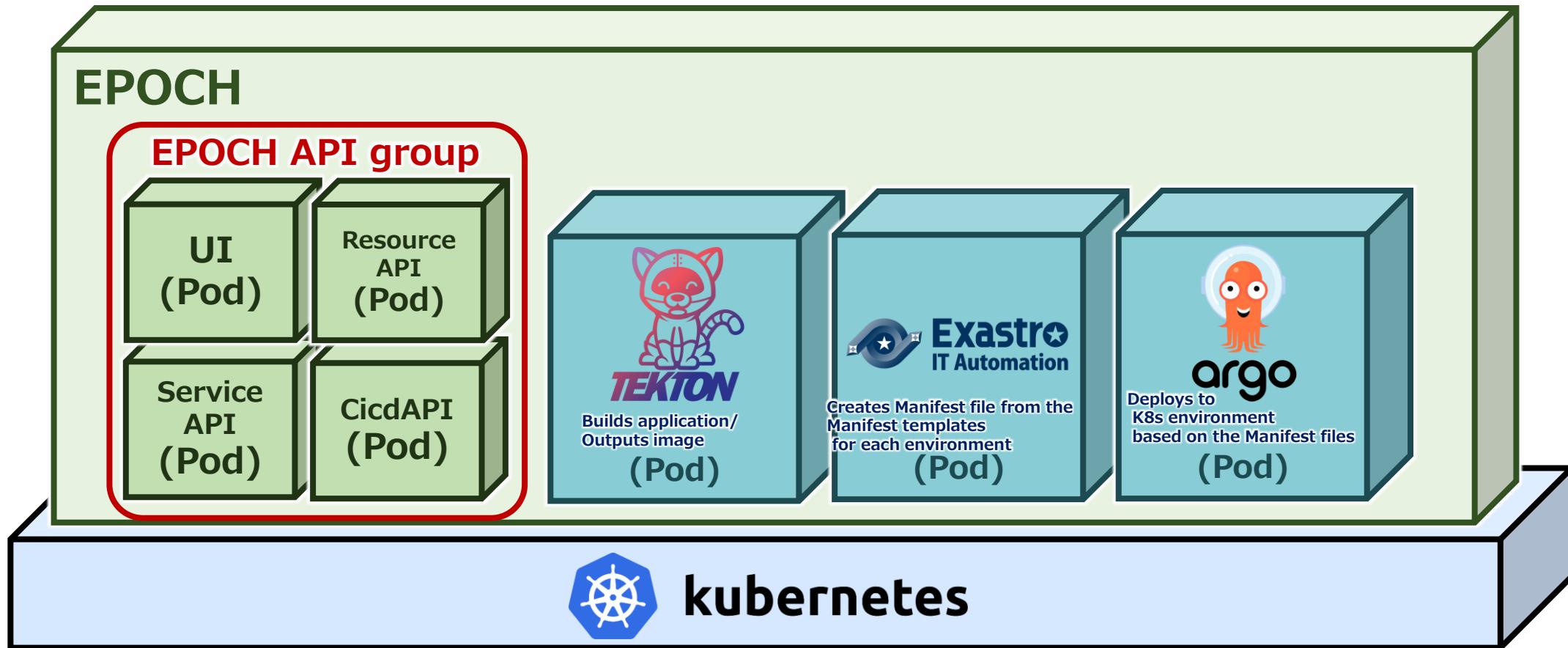
Install EPOCH and prepare the CI/CD environment



## 2.1 Install EPOCH (1/5)

### EPOCH diagram

The following diagram illustrates EPOCH and the Workspace configuration after it has been created



## 2.1 Install EPOCH (2/5)

### Prerequisites

- Environment

- A Kubernetes environment must be created
- The ServiceAccount must have the cluster-admin role
- The user must be able to connect to external internet from the Kubernetes environment
- “Git for Windows” must be installed in the PC environment
- The following port numbers must be usable(30080, 30081, 30443, 30801 , 30804, 30805, 30901~30907)  
(The port numbers are written in the epoch-install.yaml file. Run the installer in order to have the changes take effect)

- Account

- Must have a GitHub account that can register application code
- Must have a GitHub account that can register Kubernetes Manifests
- Must have a GitHub account that can register Container images

## 2.1 Install EPOCH (3/5)

### Install EPOCH

- Use a terminal to SSH login to an environment that can run kubectl. Run the following command and install

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

The following command checks the installation progress.

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

Check that the command result "STATUS" column displays "Running".

#### 【Command result】

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 Install EPOCH (4/5)

### Configure persistent volume

Configure persistent volume. This will be used for configuring the pipelines.

- Run the following command and acquire the manifest from GitHub.

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

- The following command checks the Worker nodes and host names.

```
$ kubectl get node
```

#### [Command result]

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

- Modify the epoch-pv.yaml file. (The last line of the epoch-pv.yaml file)

Change the "# Please specify the host name of the worker node #" text with the worker node and the host name acquired from the last command and save the file.

#### [Modified values]

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

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

- Use the following command to reflect it to the kubernetes environment.

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

## 2.1 Install EPOCH (5/5)

### Install ArgoRollout

- Run the following command to install ArgoRollout.

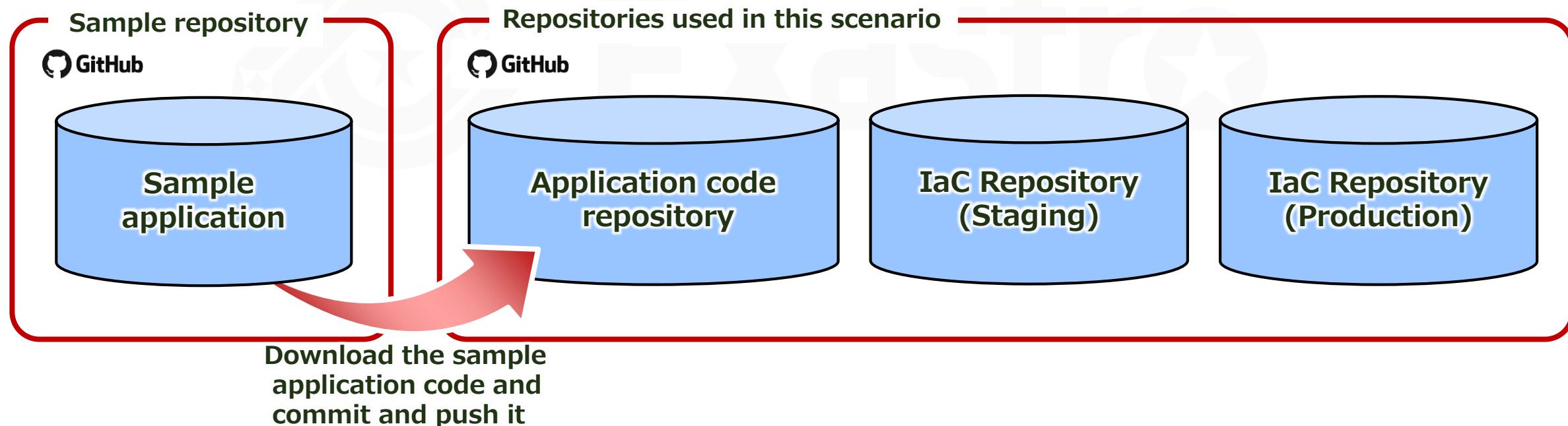
```
$ kubectl create namespace argo-rollouts  
$ kubectl apply -n argo-rollouts -f https://github.com/argoproj/argo-rollouts/releases/latest/download/install.yaml
```

This concludes the steps to installing EPOCH.  
Let us prepare for the tutorial!

## 2.2 Prepare repositories (1/4)

### About the repositories

- The repositories used in this document are as following
  - Repository for application codes
  - Repository for IaC (Staging environment)
  - Repository for IaC (Production environment)
- Repository diagram

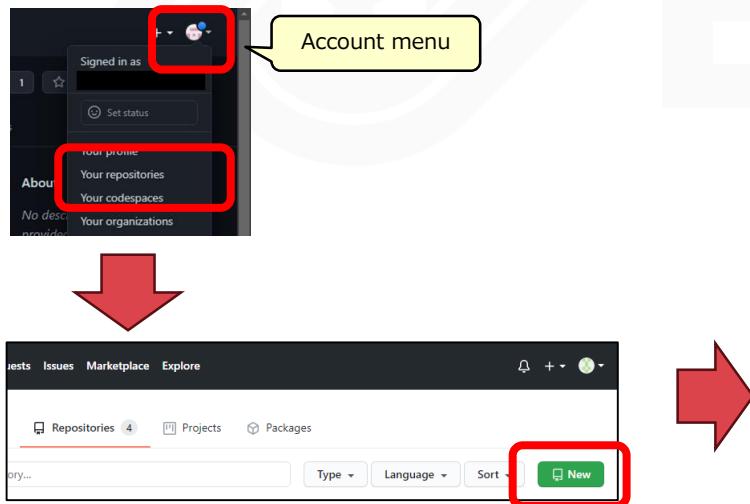


## 2.2 Prepare repositories (2/4)

### Prepare the repositories

Prepare 3 Git repositories.

- Use your browser to sign into GitHub with your own GitHub account.
- Go to the account menu and press “Your repositories”.
- Press the “New” button in the upper right corner and follow the diagram below to fill in the values and create the repository.
- Lastly, press the “Create repository” button.



Specify the following Repository names

- 1 : epoch-sample-app
- 2 : epoch-sample-staging-manifest
- 3 : epoch-sample-production-manifest

Check the “Add a README File”

Create a new repository

A repository contains all project files, including the revision history. Already have [Import a repository](#).

Owner \*  /  ✓

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 Prepare repositories (3/4)

### Prepare repository for application code.

- Clone the Application code repository

Clone the Application code repository to the PC environment.

The commando prompt should look something like this:

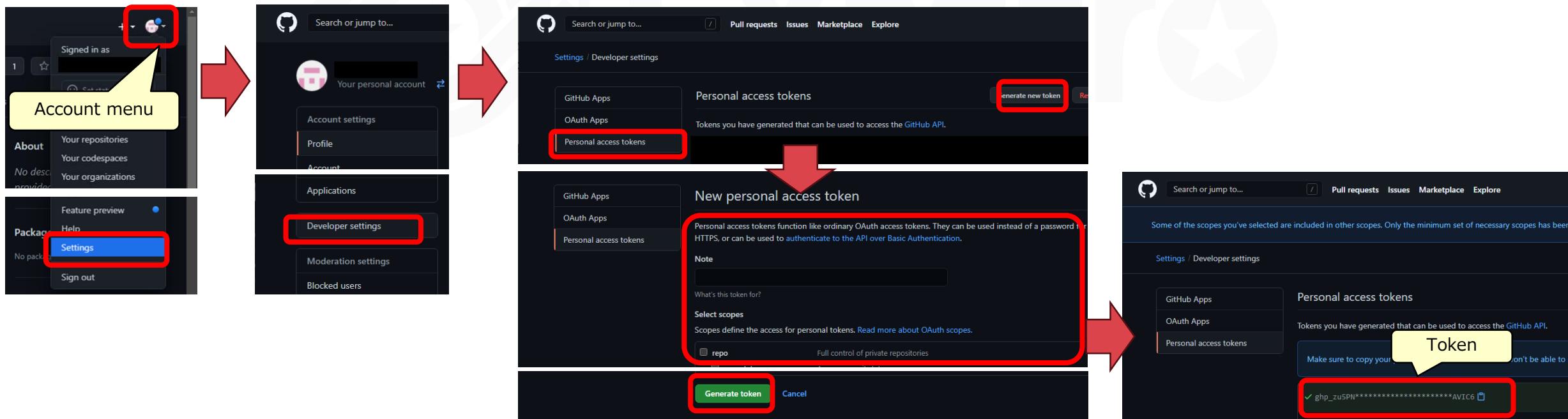
```
> cd "[clone folder]"
> git clone https://github.com/[Github account name]/epoch-sample-app.git
> cd epoch-sample-app
> git config user.name "[GitHub user name]"
> git config user.email "[GitHub email address]"
```

The next part of the tutorial will use the cloned local repository.

## 2.2 Prepare repositories (4/4)

### Pay out Git token

- Use your browser to sign into GitHub with your own GitHub account.
- Go to the account menu and press “Settings”.
- Go to the Account settings and then to the “Developer settings” menu.
- Go to the “Personal access tokens” menu and press “Generate new token”.
- Input a name om the “Note” field and select all the scoped before pressing the “Generate token”
- The token (ghp\_\*) will be used later , so make sure to save it somewhere.



## 2.3 Manifest template files

### Download Manifest template files

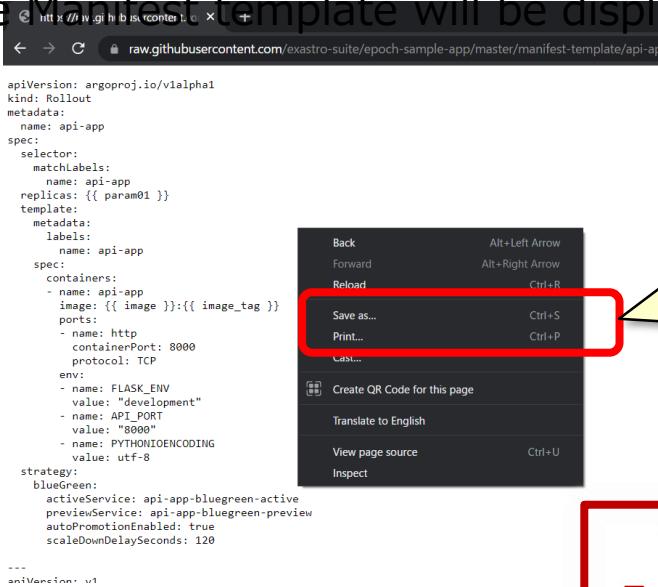
Download the 2 Manifest template files we will upload to EPOCH.

- Access the files from the following URLs.

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

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

- The Manifest template will be displayed in the browser window. Save the template to the PC.



```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: api-app
spec:
  selector:
    matchLabels:
      name: api-app
  replicas: {{ param01 }}
  template:
    metadata:
      labels:
        name: api-app
    spec:
      containers:
        - name: api-app
          image: {{ image }}:{{ image_tag }}
      ports:
        - name: http
          containerPort: 8000
          protocol: TCP
      environment:
        - name: FLASK_ENV
          value: "development"
        - name: API_PORT
          value: "8000"
        - name: PYTHONIOENCODING
          value: utf-8
  strategy:
    blueGreen:
      activeService: api-app-bluegreen-active
      previewService: api-app-bluegreen-preview
      autoPromotionEnabled: true
      scaleDownDelaySeconds: 120
---
```

Make sure to save it as an YAML file  
by right clicking and pressing “Save as …”.

**That concludes the preparation.  
In the next part, we will create our workspace.**

### 3. Create Workspace

Create Workspace and prepare CI/CD.

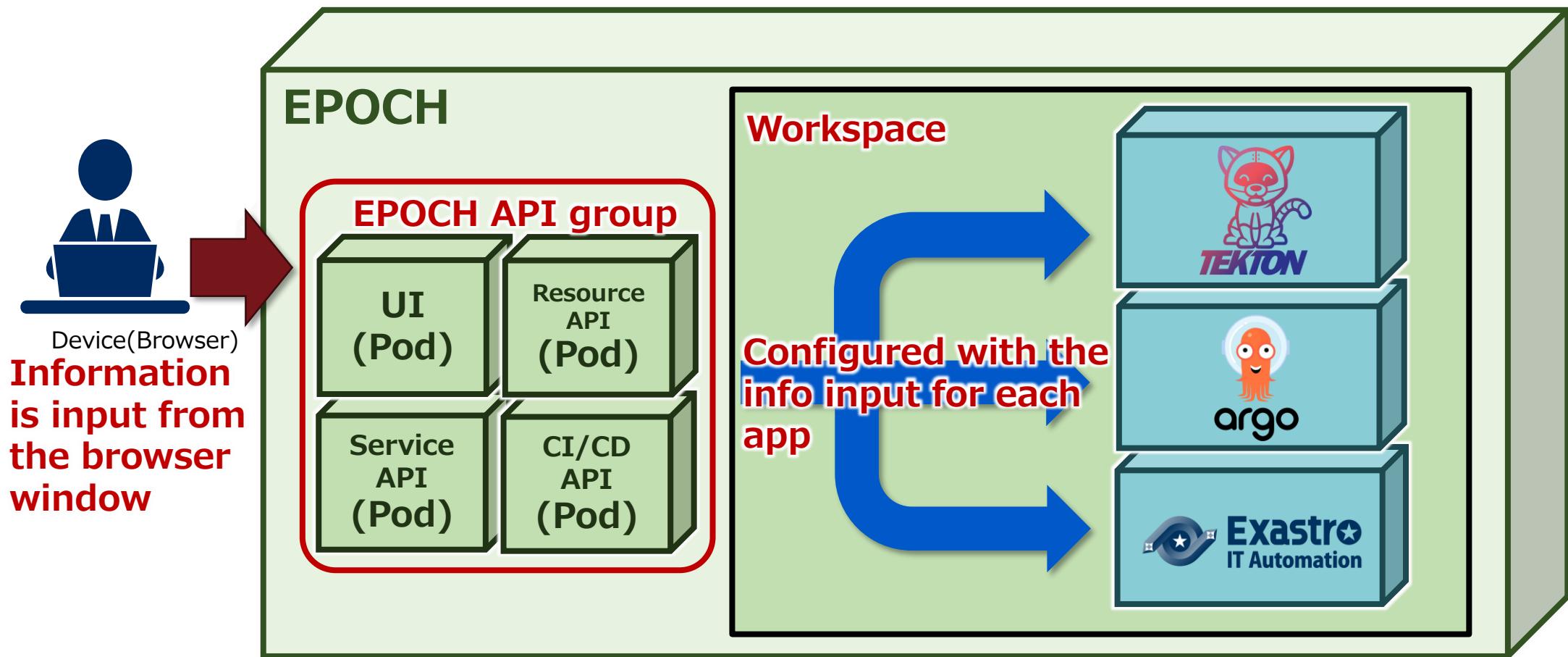


# 3.1 Workspace

## Workspace

In EPOCH, the word “Workspace” specifies a workspace for 1 development environment.

A workspace can be created by inputting the necessary information on the EPOCH screen.



## 3.2 CI/CD

### What is CI/CD?

CI/CD is a method for developers to increase the frequency of app delivery by automating the tasks from app development to release.

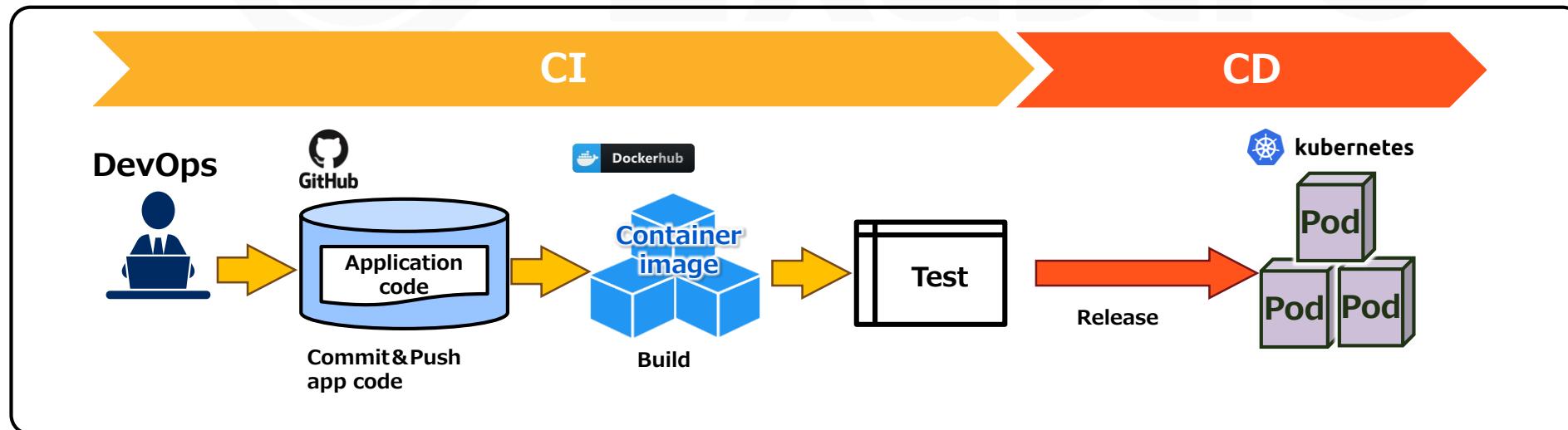
- CI (Continuous Integration)

CI is the name of the process of automated building and testing application code whenever there has been a change to it.

- CD (Continuous Delivery)

CD is the name of the process of automating the tasks of releasing the code to the runtime environment.

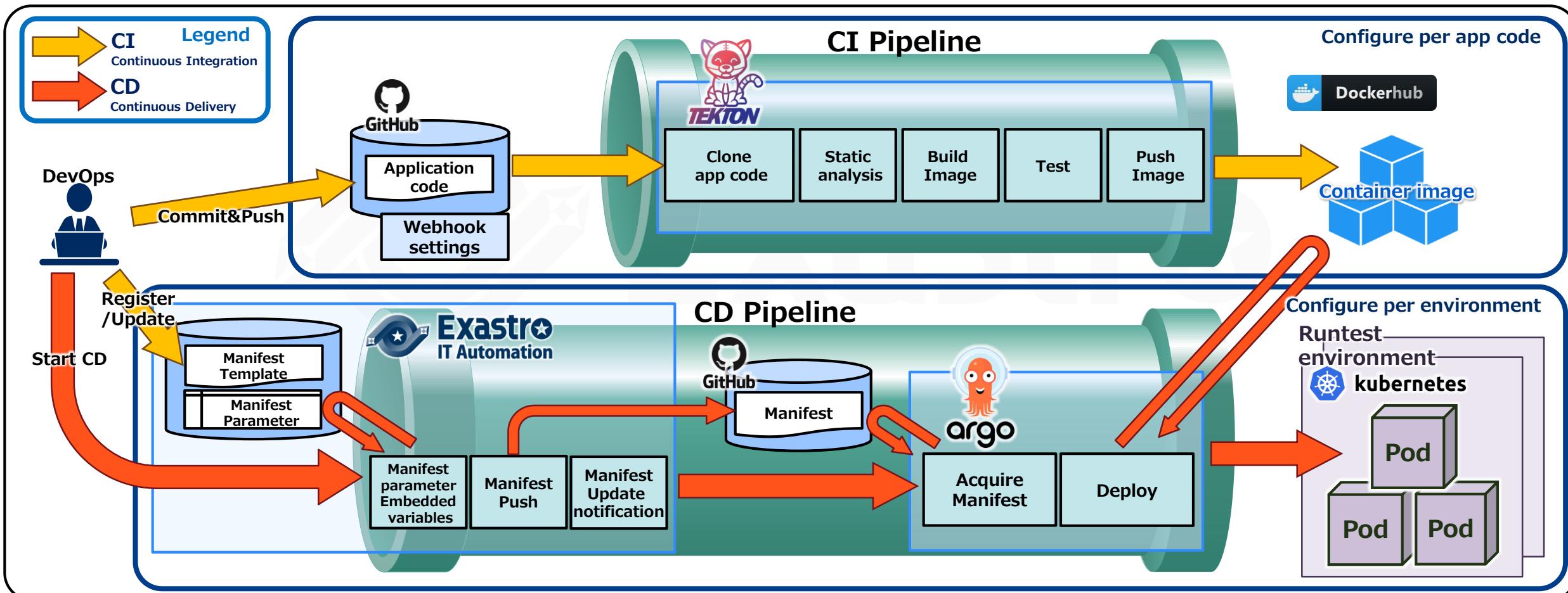
- CI/CD image



# 3.3 CI/CD in EPOCH

## CI/CD in EPOCH

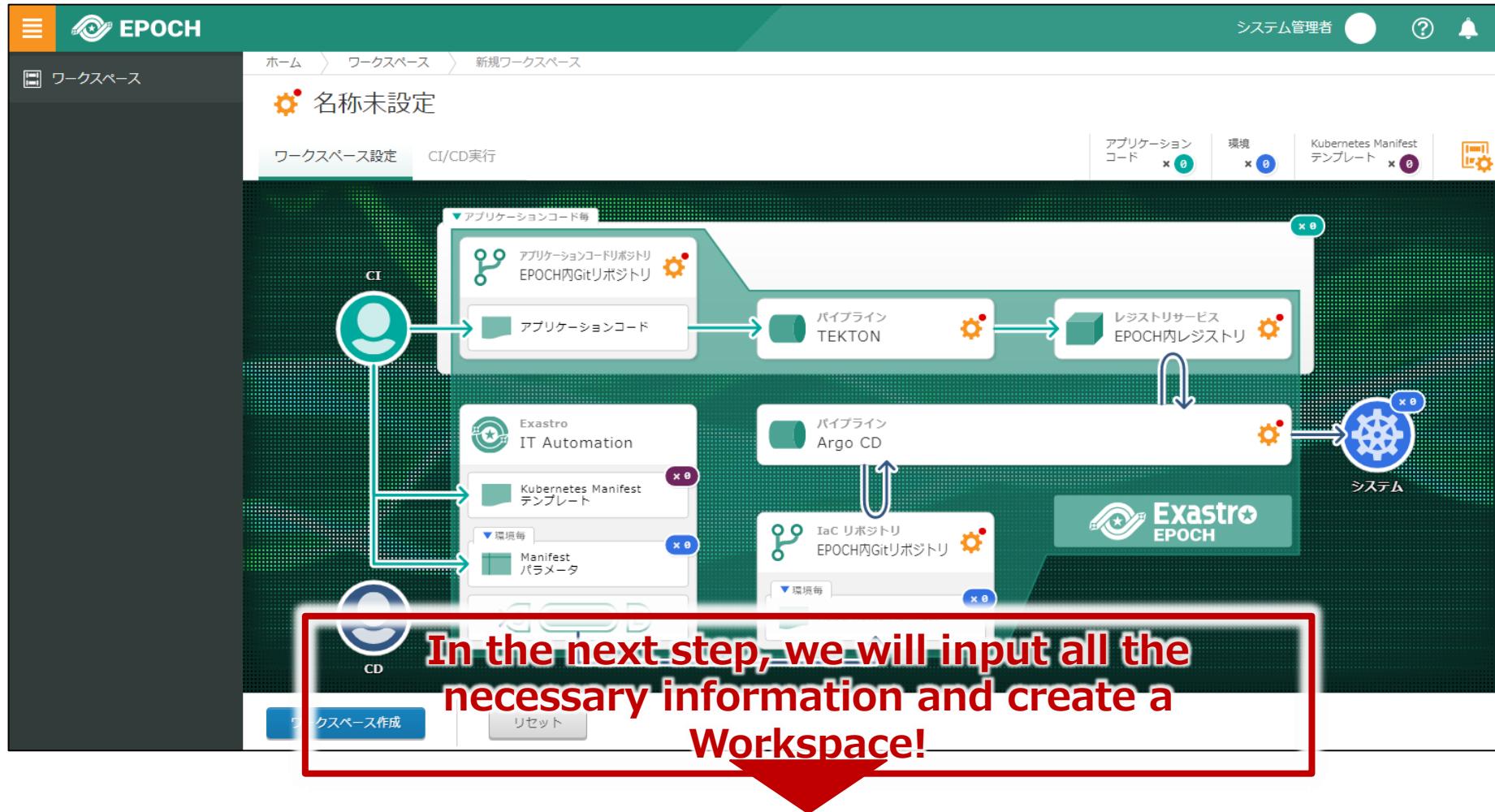
The following diagram illustrates the CI/CD used in EPOCH.



### 3.4 Start EPOCH

Use the following URL to access the EPOCH environment.

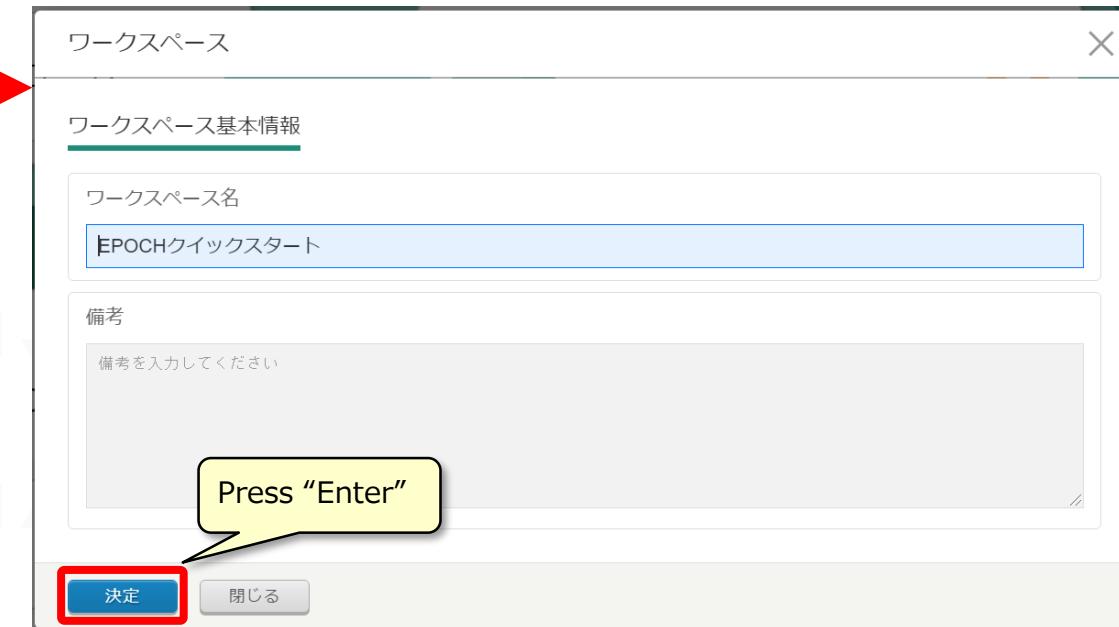
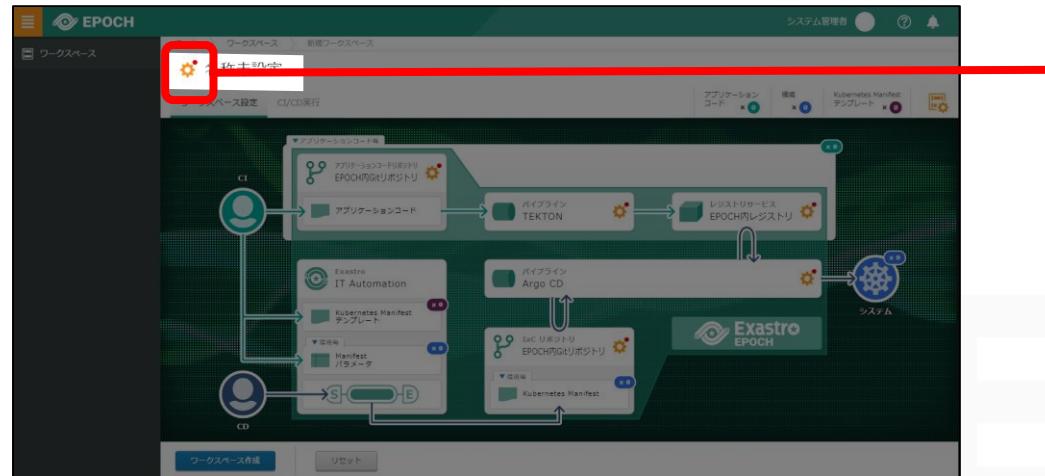
**https://[IP Address or Host name]:30443/workspace.html**



# 3.5 Create Workspace (1/7)

## Workspace information

Input the Workspace name.



Item name	Input · Select values	Description
Workspace name	EPOCH Quickstart	Name of the workspace
Remarks	BLANK	Free space for describing the workspace

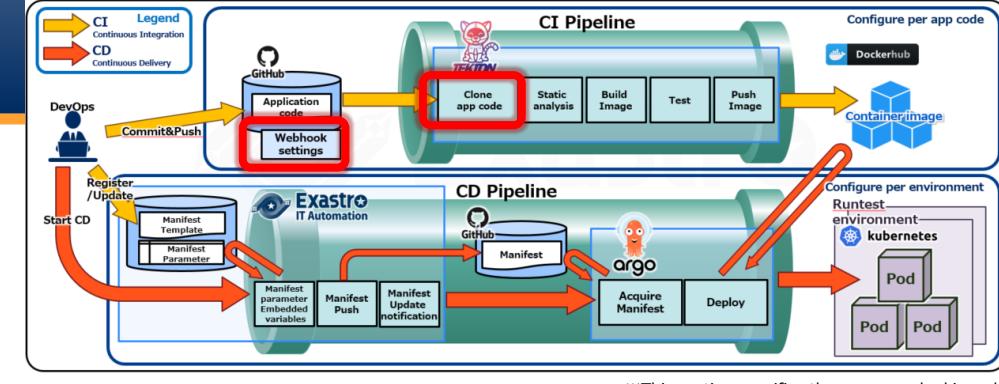
# 3.5 Create Workspace (2/7)

## App code repository

Input information for the application code repository.

Select "GitHub"

Press "Enter"

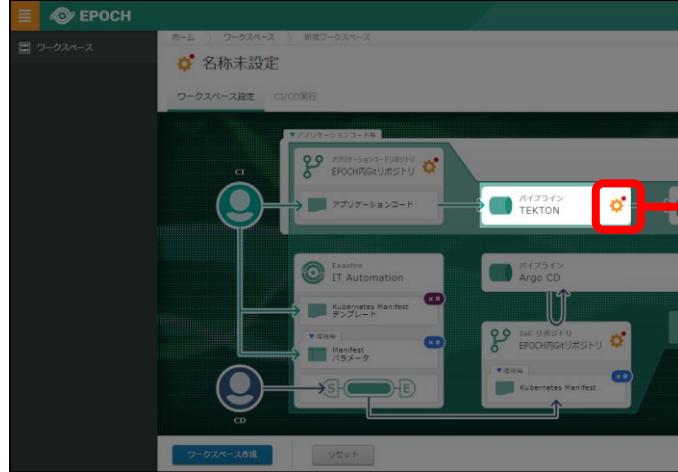


Item	Input/Select	Description
User name	(GitHub account name)	Your personal GitHub account name
Token	(GitHub token)	GitHub token (See the Git token you created earlier)
Git Repository URL	<a href="https://github.com/[GitHub account name]/epoch-sample-app.git">https://github.com/[GitHub account name]/epoch-sample-app.git</a>	Input the URL for the application code repository you created earlier.

# 3.5 Create Workspace (3/7)

## TEKTON Pipeline

Input information for the pipeline that will be set to TEKTON.



**TEKTON**

ビルドパラメータ一覧

epoch-sample-app

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

ビルド ブランチ  
main, master

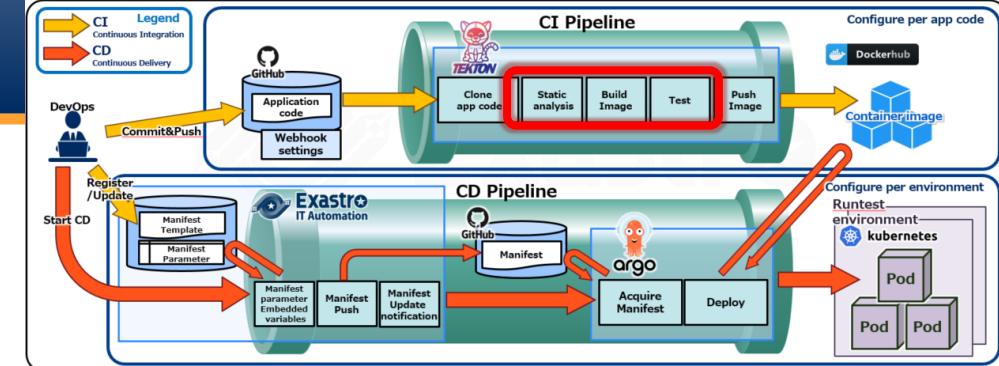
ビルド Dockerファイルパス  
Dockerfile

静的解析  
 使用しない  SonarQube

Select "None"

決定 キャンセル

Press "ENTER"



※This section specifies the areas marked in red.  
※Static response tests and other tests are planned to be added in the future.

Item	Input/Select	説明
Build branch	main, master	The build's target GitHub application branch
Docker build file path	./api-app/Dockerfile	Application docker file path

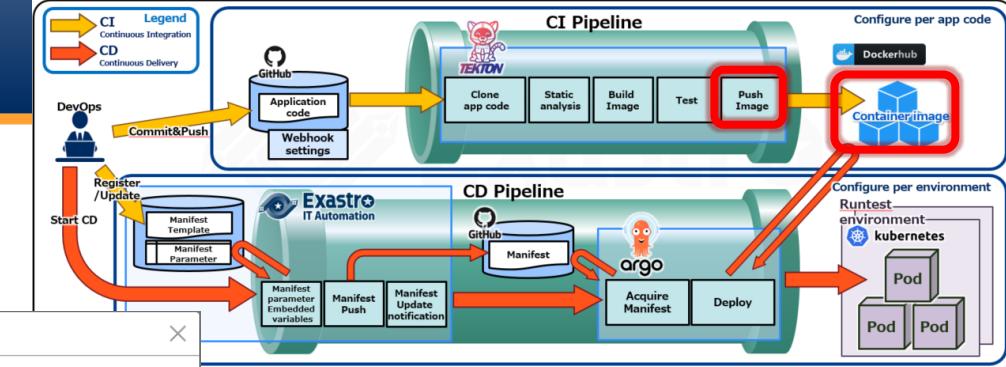
# 3.5 Create Workspace (4/7)

## Registry service

Input the image registration destination (Registry).

The screenshot shows the EPOCH workspace creation interface. On the left, there's a diagram of the CI/CD pipeline. On the right, a detailed configuration window titled 'レジストリサービス' (Registry Service) is open. In the configuration window, a radio button for 'DockerHub' is selected, indicated by a red box and a callout 'Select DockerHub'. Below it, there's a section for 'Registry connection account' with fields for 'User name' (set to 'your-dockerhub-account') and 'Password' (redacted). At the bottom, there's a '决定' (Confirm) button highlighted with a red box and a callout 'Press "Enter"'.

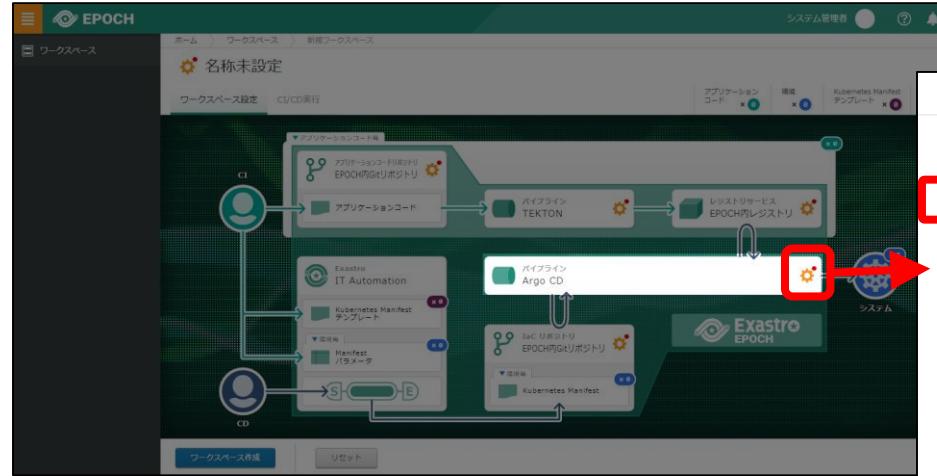
Item	Input/Select	Description
User name	(Your personal DockerHub account name)	DockerHub account name
Password	(Your personal DockerHub password)	DockerHub password
Image output destination	[DockerHub account name]/epoch-sample- <b>api</b> ※Modify the text that displays after inputting the user name.	DockerHub image output destination



# 3.5 Create Workspace (5/7)

## Argo CD Pipeline

Input the Deploy information that will be set to ArgoCD.

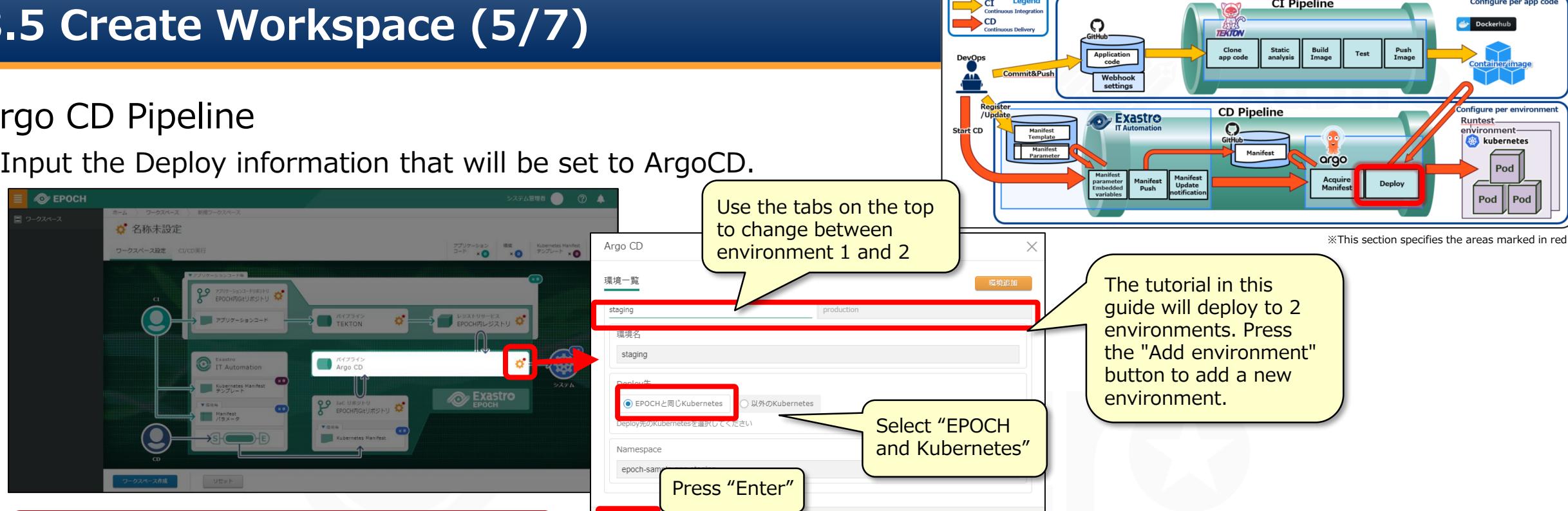


### Environment 1 : Staging environment

Item	Input/Select	Description
Environment name	staging	Name of the environment
Namespace	epoch-sample-app-staging	Deploy destination Namespace

### Environment2 : Production environment

Item	Input/Select	Description
Environment name	production	Name of the environment
Namespace	epoch-sample-app-production	Deploy destination Namespace

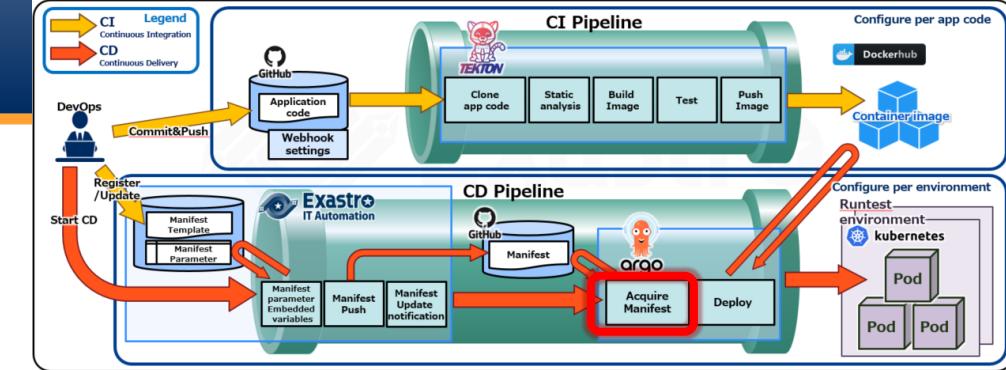


# 3.5 Create Workspace (6/7)

## IaC repository

Input the repository information.

This will determine where the manifest will be registered to.

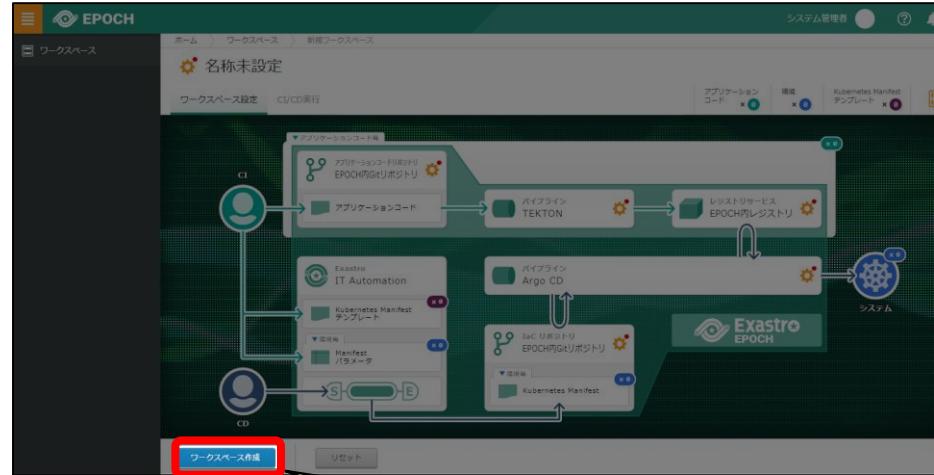


Item	Input/Select	Description
Git repository URL	<a href="https://github.com/">https://github.com/</a> [GitHub account name]/[Environment repository].git	The Environment's Manifest repository. (See the IaC repository we prepared earlier)

# 3.5 Create Workspace (7/7)

## Create workspace

After you have input all the necessary information, press “Create Workspace”.



Press “Create Workspace”

You have now created the CI/CD pipeline!  
The next tutorial will guide you through a scenario  
of using the CI/CD pipeline.!



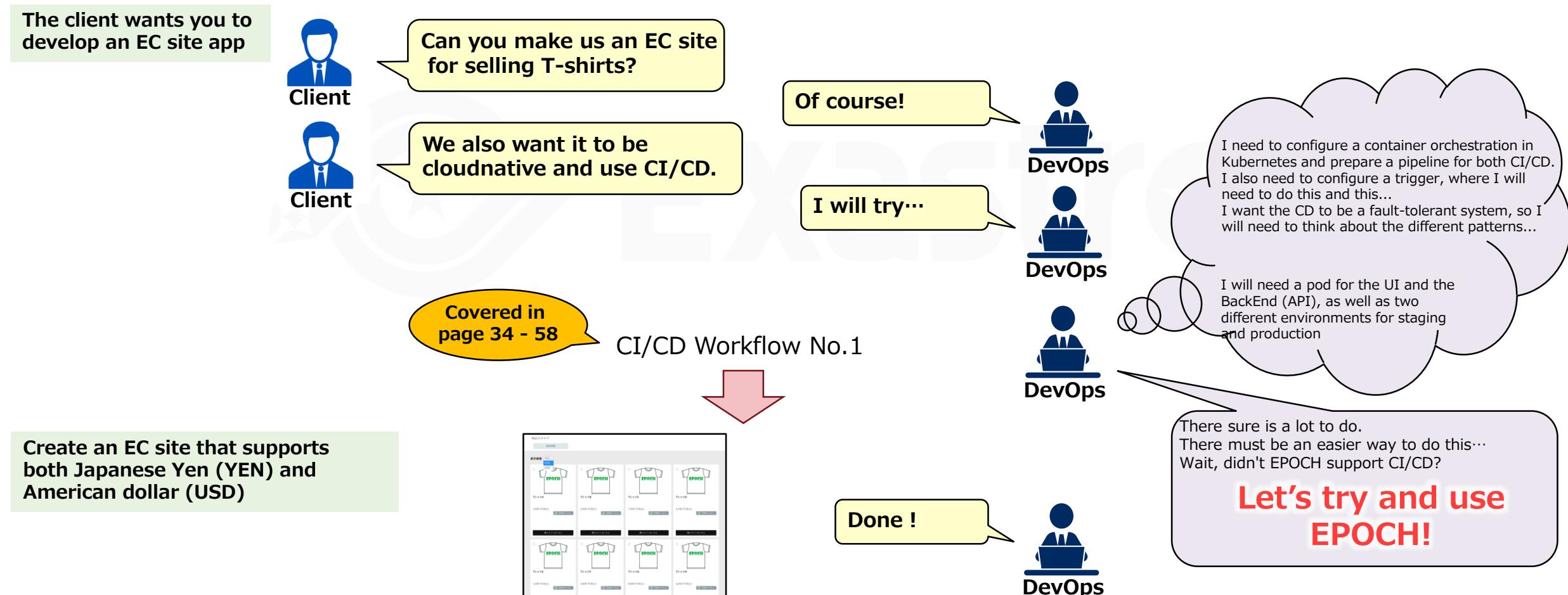
## 4. Tutorial

Let's use the CD/CD workflow

# 4.1 Tutorial overview (1/2)

## CI/CD development scenario

This tutorial uses the following scenario as a base to guide the reader through a CI/CD workflow.  
In this tutorial, the reader will create an EC site by using a prepared sample application.



## 4.1 Tutorial overview (2/2)

The client asks for a new



Client

The EC site is doing great!



Client

We want to appeal to the European market as well. Can we add Euro as well?

Covered in page 59 - 80.

That is good to hear



DevOps

Of course!



DevOps

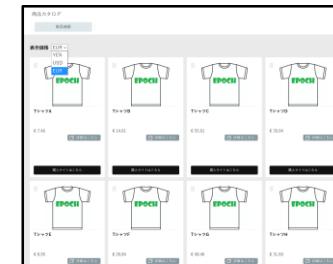
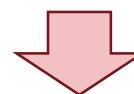


DevOps

The CI/CD is already completed, so it's easy to deploy once I've fixed the code!

Update the EC site so that it supports Euro(EUR) in addition to JPY and USD

CI/CD Workflow No.1



Done !

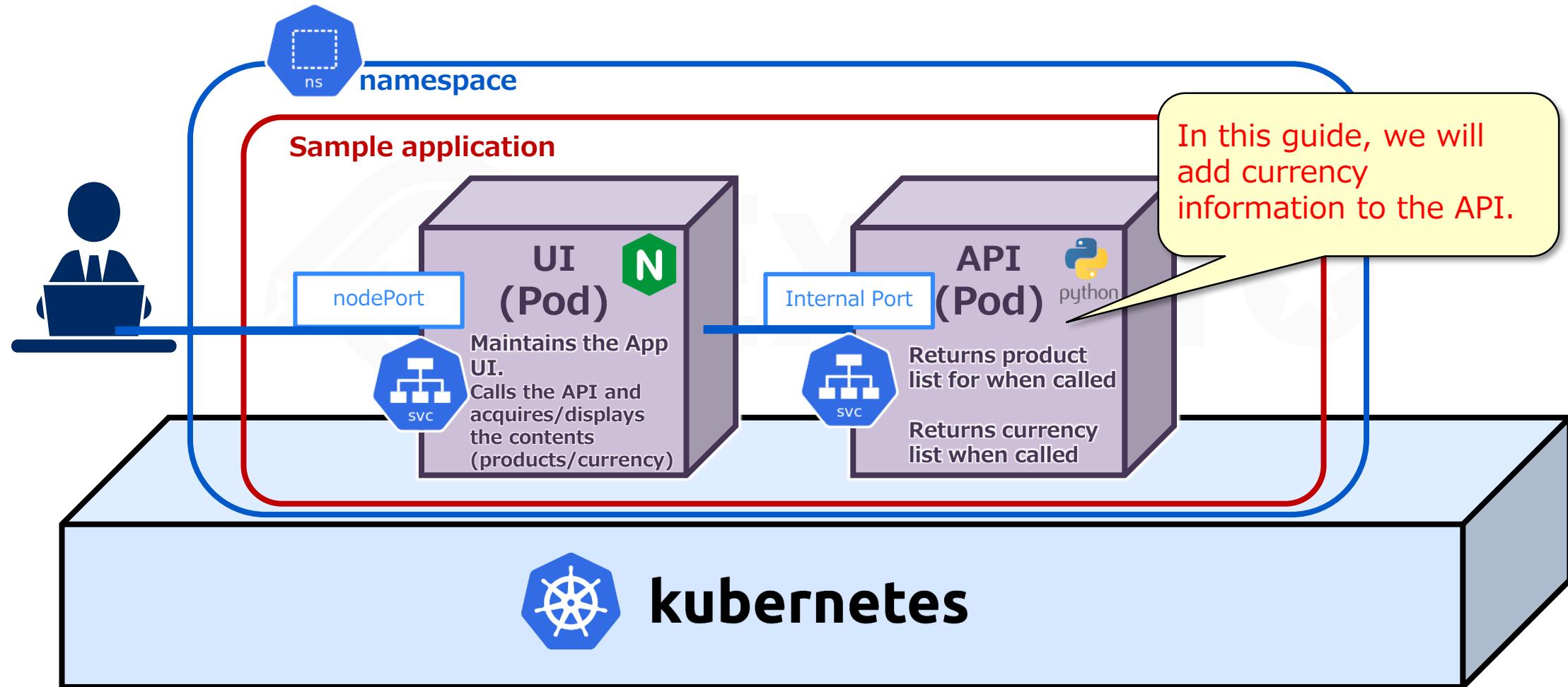


DevOps

## 4.2 Sample application configuration

### Sample application configuration

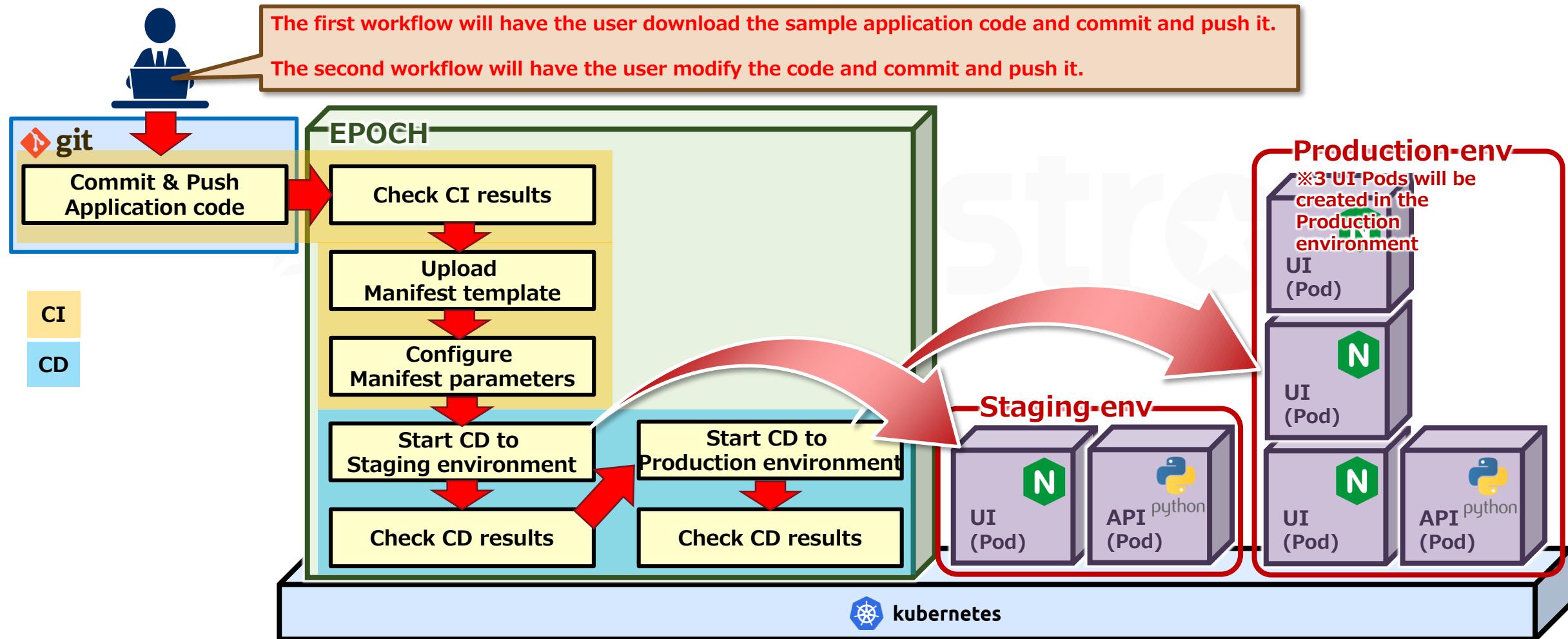
The sample application is created by an UI and an API.



## 4.3 Tutorial (CI/CD workflow)

### CI/CD workflow

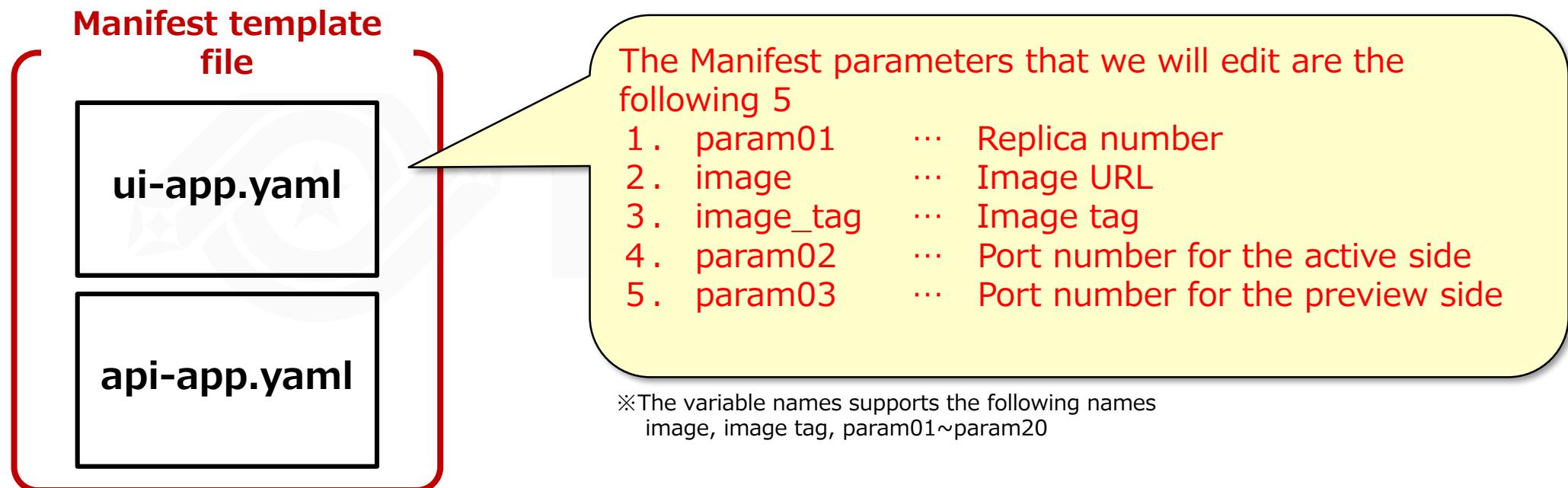
In this scenario, the user will deploy the sample code to both the Staging and the Production environment. After that, they will modify the sample code and proceed to deploy it to both the Staging and the Production environment.



## 4.4 Manifest template file

### Manifest template file

There are two Manifest template files for the sample application. One for UI and one for API. They are template files that have modifiable variables for matching environments.



**In the next section, we will start the first CI/CD workflow!**

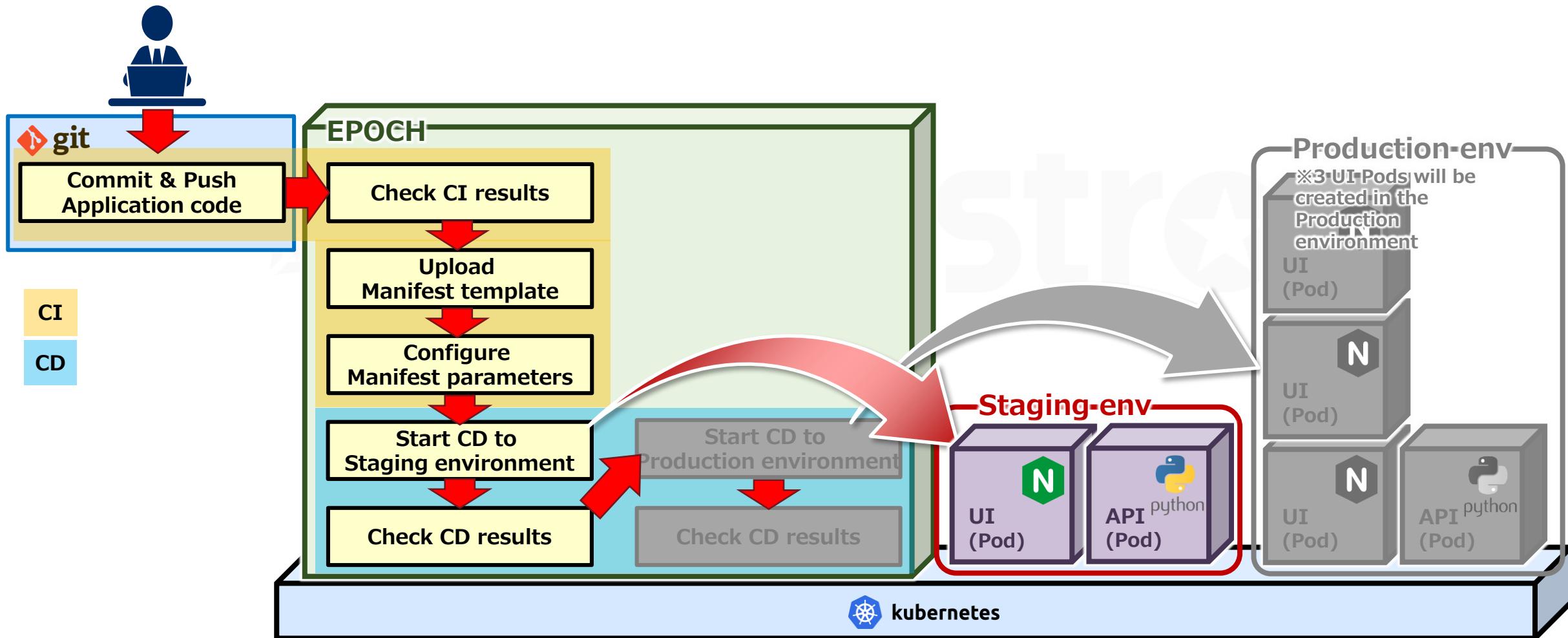
# CI/CD Workflow No.1



## 4.5 CI/CD Workflow No.1 (1/25)

### Deploy to Staging environment

The first CI/CD workflow will have the user commit and push the application code, deploy it to the Staging environment, as well as checking the results of the CD.



# 4.5 CI/CD Workflow No.1 (2/25)

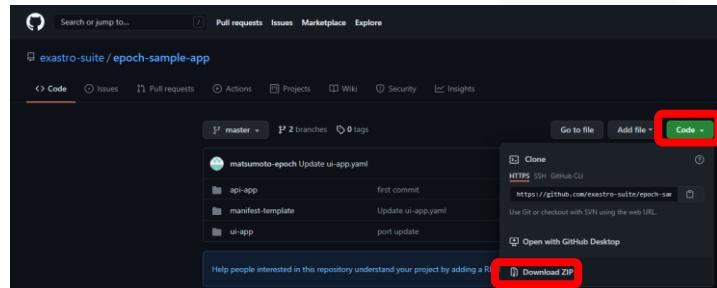
## Commit & Push Application code

Run the CI Pipeline in order to create the container image we will deploy.

- Access the following URL to download the sample app.

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

- From the GitHub page, press the "Code" button and "Download ZIP" in order to download the code.



- Extract the zip file and copy it to the cloned application code repositories.



## 4.5 CI/CD Workflow No.1 (3/25)

- Commit and Push to the Cloned application code repository.

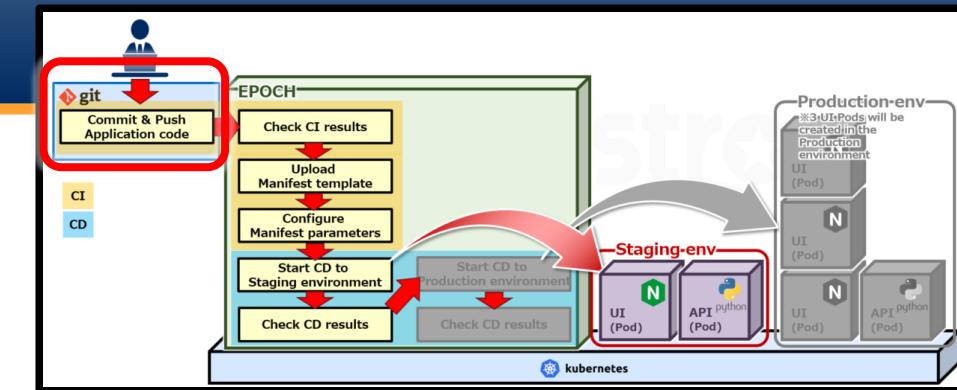
The command prompt should look something like this:

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

※ If you are asked to authenticate yourself when you are running the git push command, input the necessary GitHub account information.

When you have successfully run the Push command, The CI Pipeline set by the TEKTON Pipeline should automatically start.

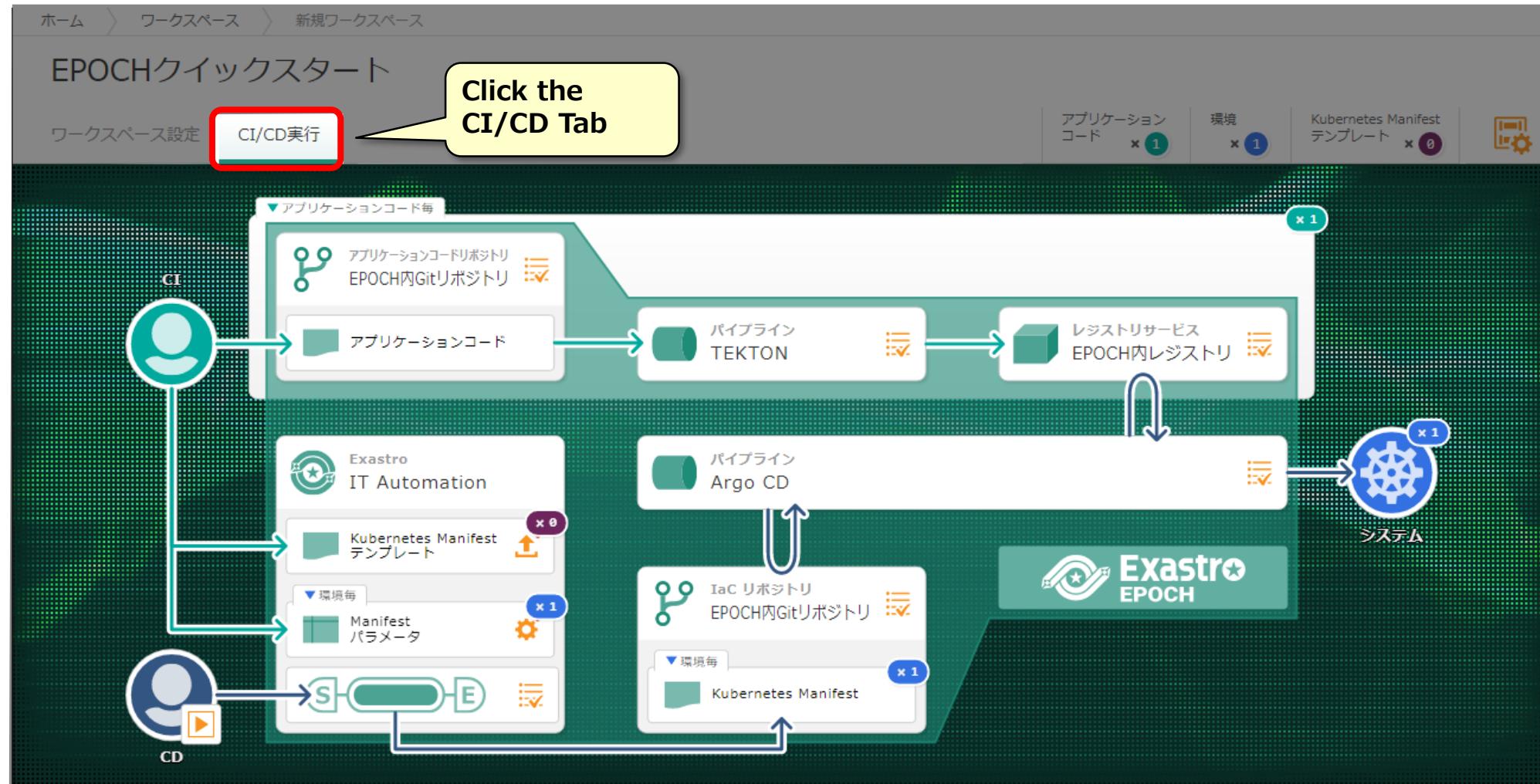
Now we need to check the CI Pipeline results.



# 4.5 CI/CD Workflow No.1 (4/25)

## CI/CD Execution screen

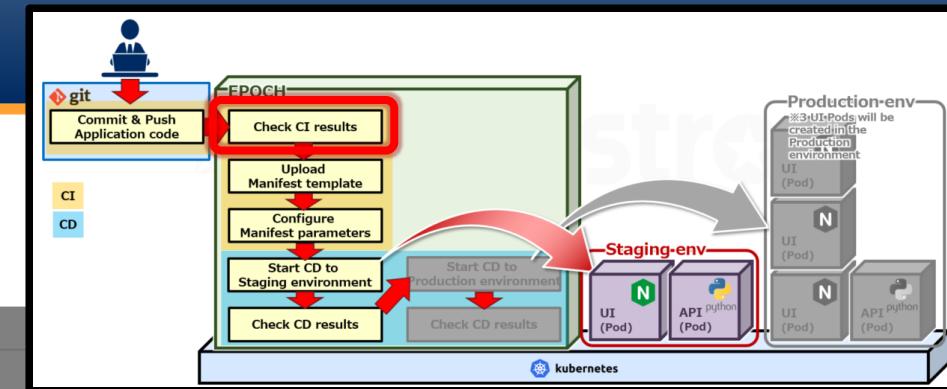
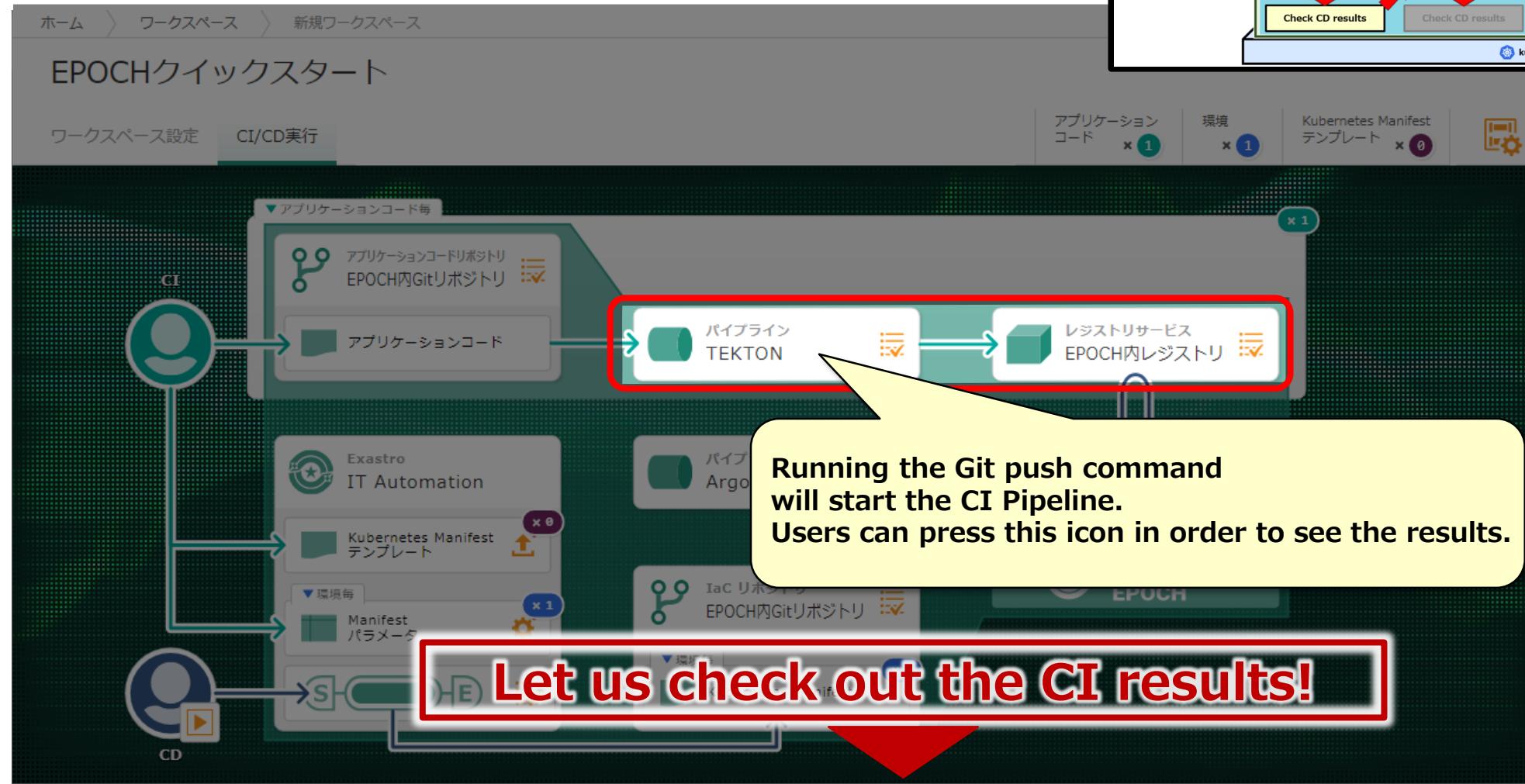
Press the CI/CD Tab in the Workspace screen to go to the CI/CD Execution screen.



# 4.5 CI/CD Workflow No.1 (5/25)

## Check CI results

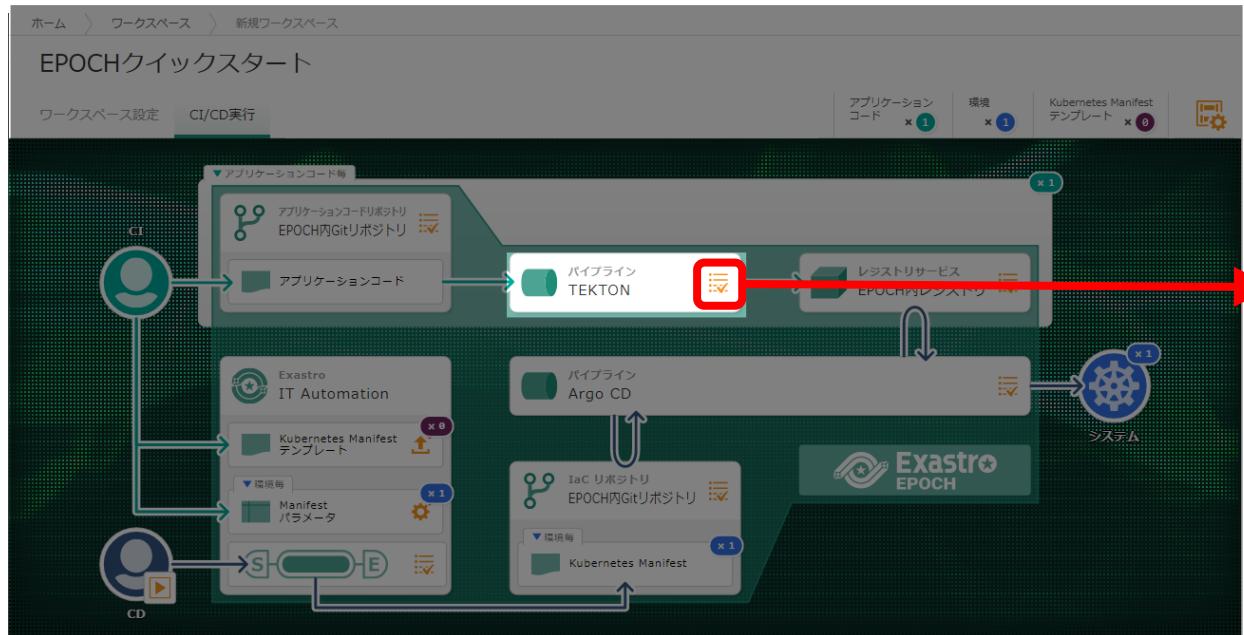
- Check the build results of the Application code.



# 4.5 CI/CD Workflow No.1 (6/25)

## Check the TEKTON Pipeline

- Check the TEKTON pipeline that the build has ended successfully.



The screenshot shows the Tekton Dashboard. On the left, a sidebar lists various resources: Tekton resources, Pipelines, PipelineRuns (which is selected), PipelineResources, Tasks, ClusterTasks, TaskRuns, Conditions, EventListeners, TriggerBindings, ClusterTriggerBindings, TriggerTemplates, Import resources, and About. The main area is titled "PipelineRuns" and shows a table of operations:

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

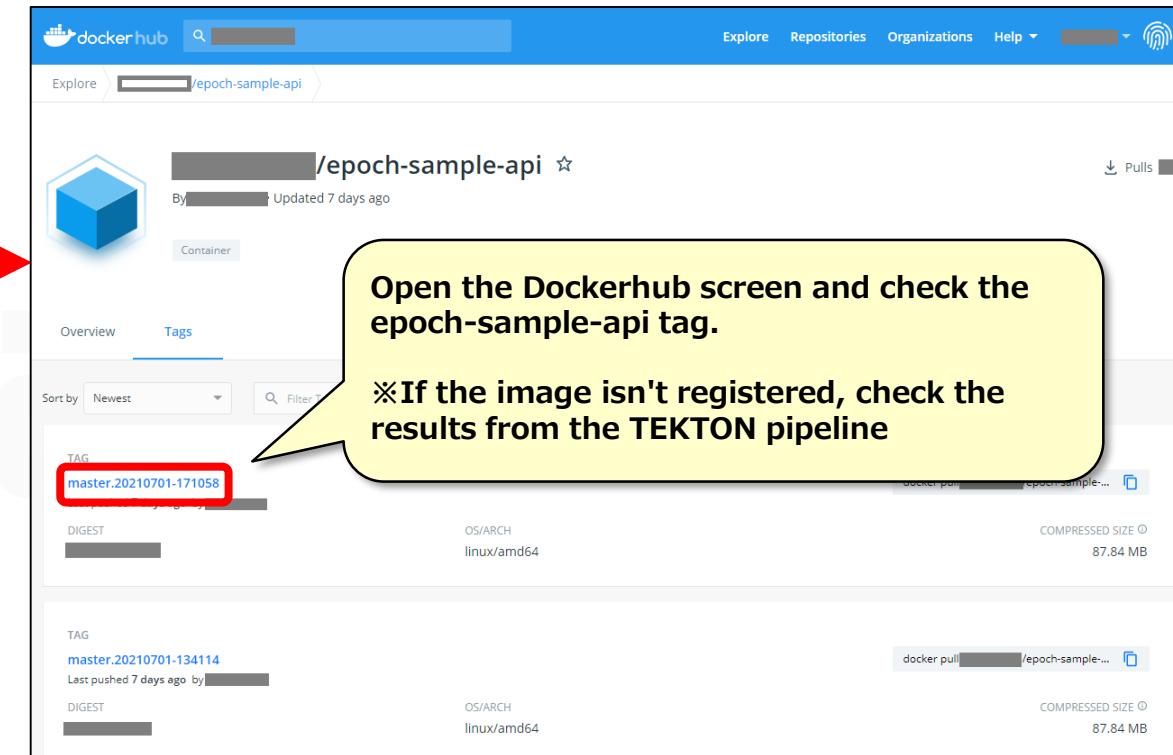
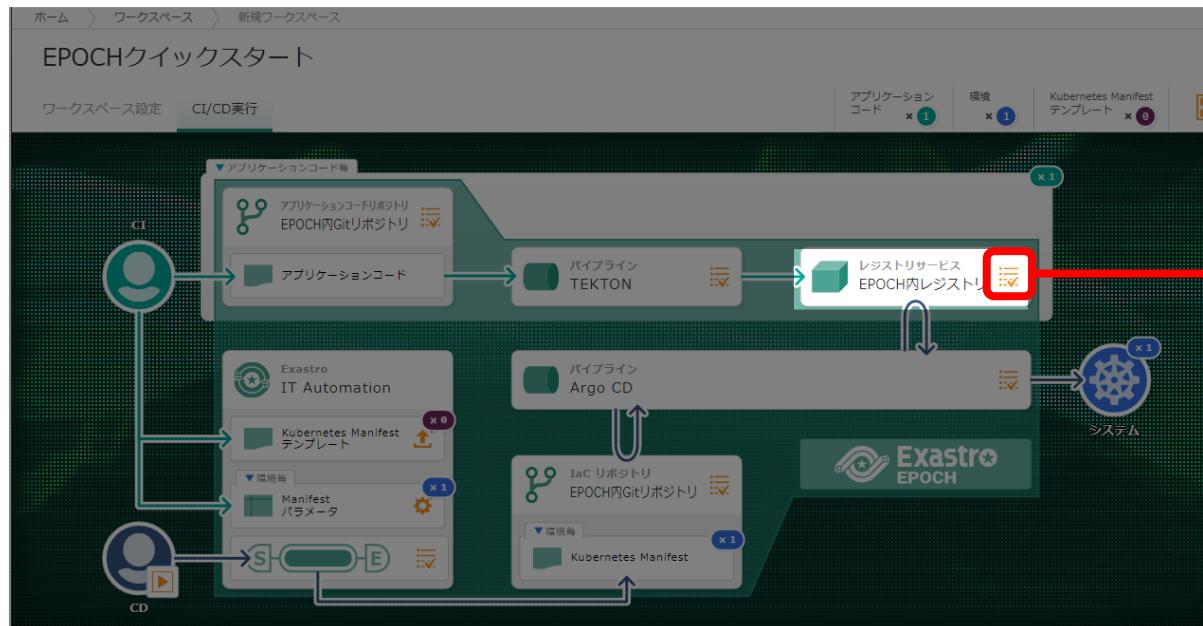
A callout bubble points to the "Status" column of the first row, stating: "If the ‘Status’ displays a green checkmark, the operation has ended successfully."

Another callout bubble points to the "Pipeline" column of the same row, stating: "Users can check the CI Pipeline operations from the TEKTON dashboard. By pressing the name of the operation, users can see more information regarding the CI Pipeline contents."

# 4.5 CI/CD Workflow No.1 (7/25)

## Check the Container image tag name

- Open the registry service screen and check the tag of the built container image.



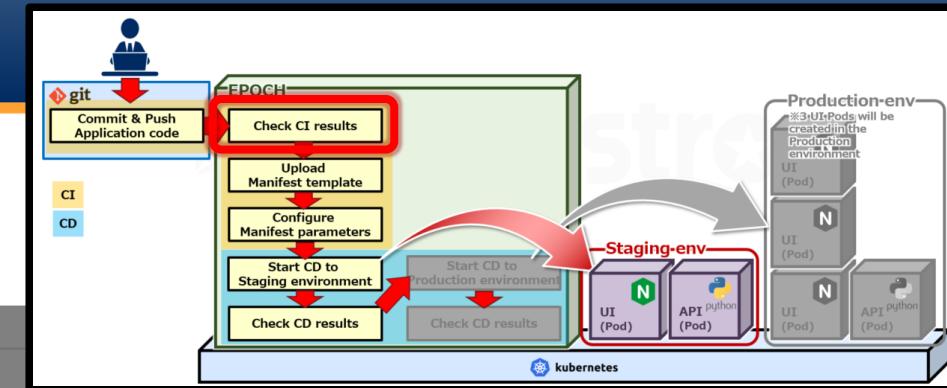
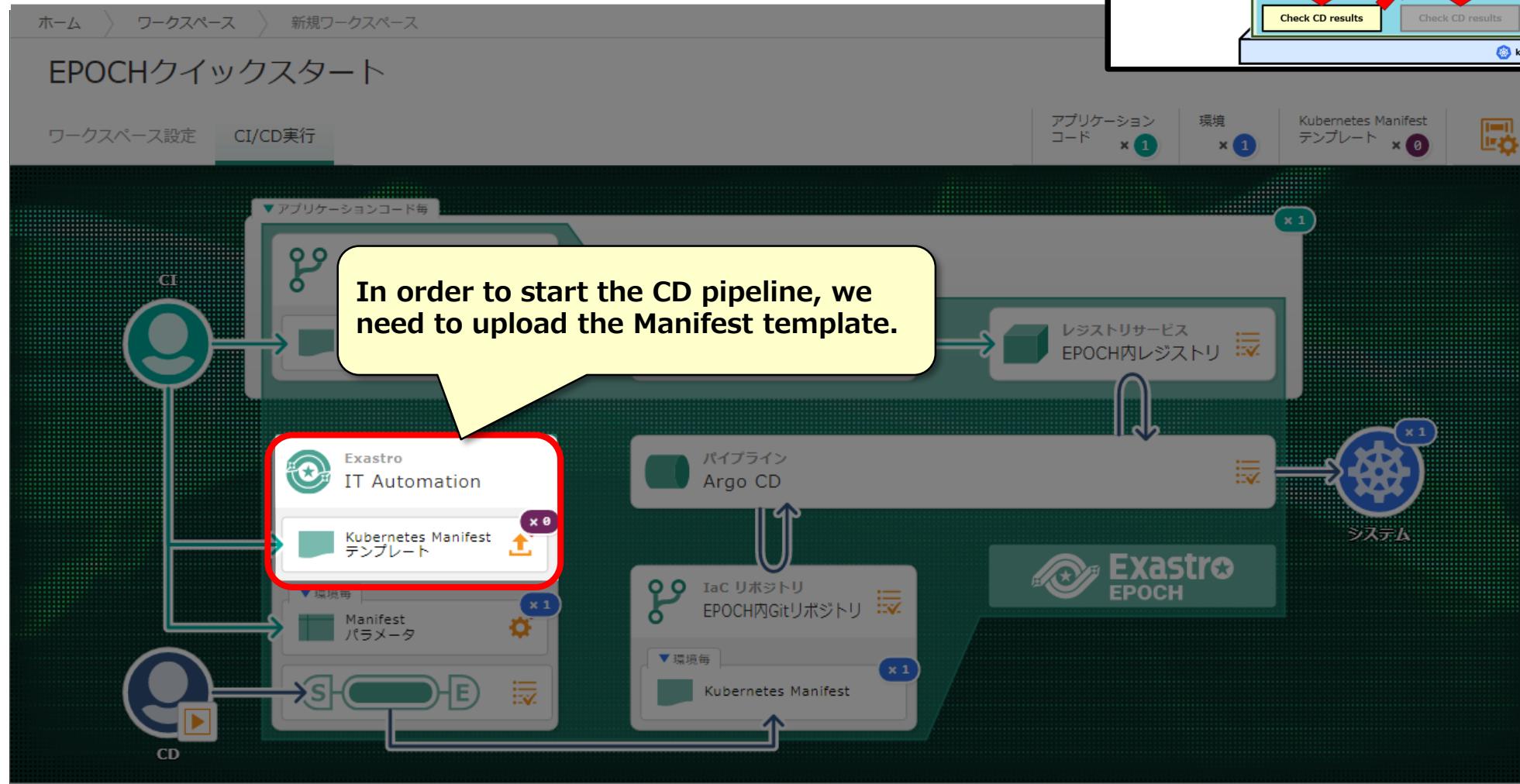
**The Image tag will be used in the next step where we input it into the Manifest parameter (image\_tag), so make sure to save it.**

※We plan to make it possible for users to select the image\_tag created in the Pipeline.

# 4.5 CI/CD Workflow No.1 (8/25)

## Upload Manifest template

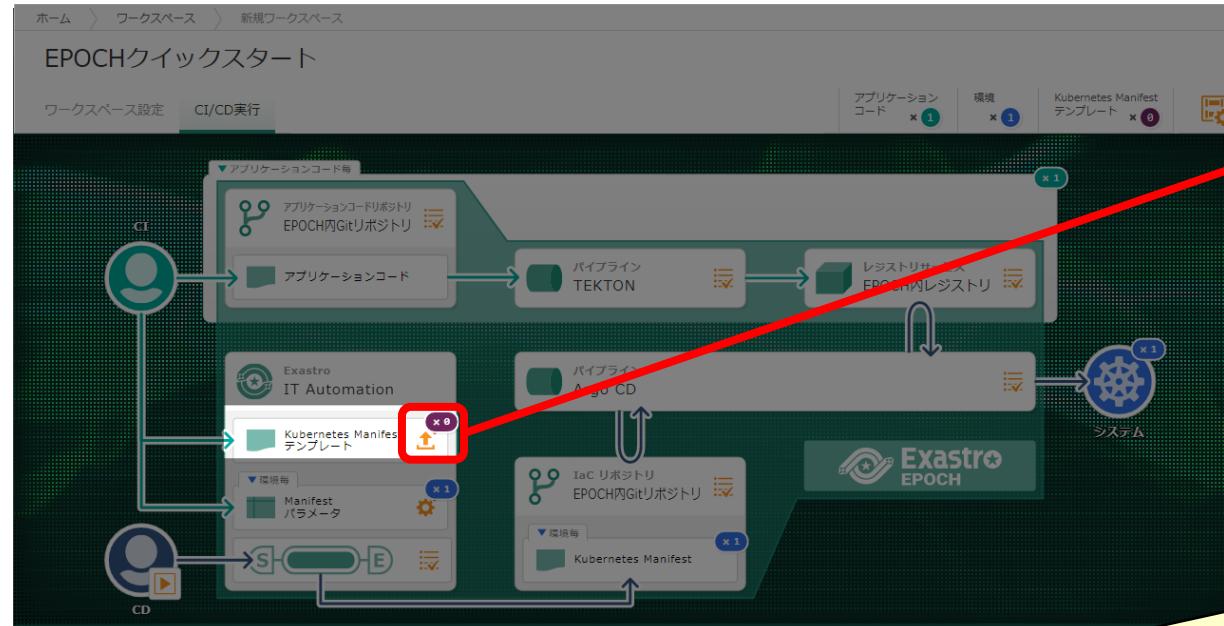
- Upload the Manifest template we downloaded earlier



# 4.5 CI/CD Workflow No.1 (9/25)

## Upload Manifest template

- Upload the Manifest template file we prepared earlier.



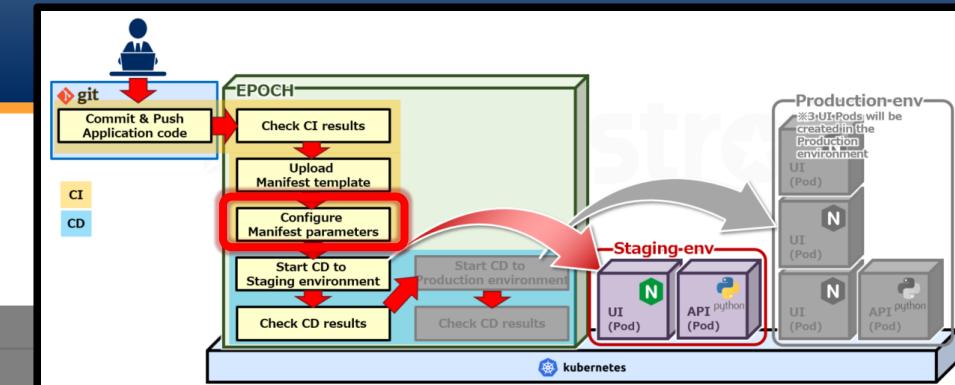
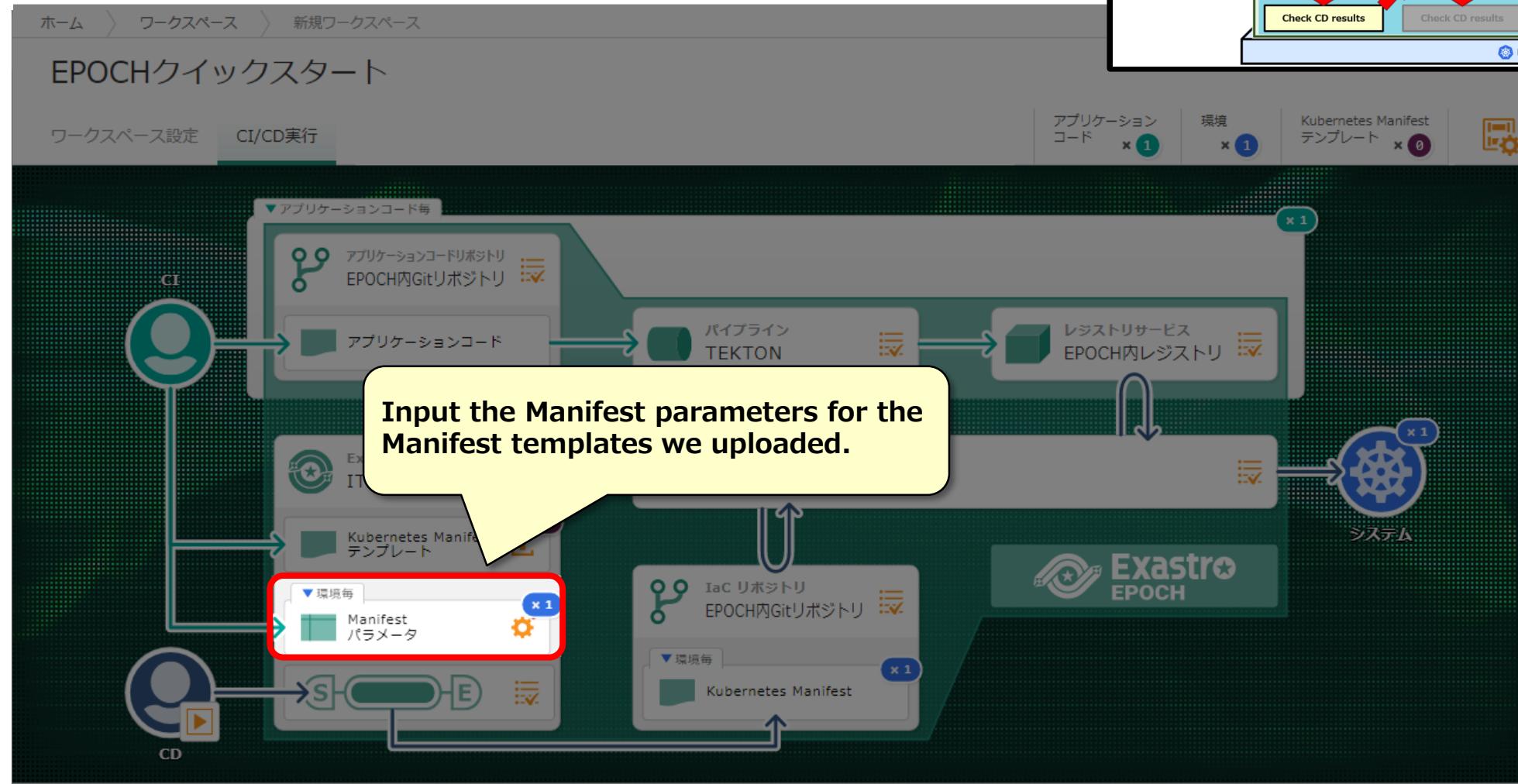
Drop the **api-app.yaml** and **ui-app.yaml** files to the area on the right



# 4.5 CI/CD Workflow No.1 (10/25)

## Manifest parameters

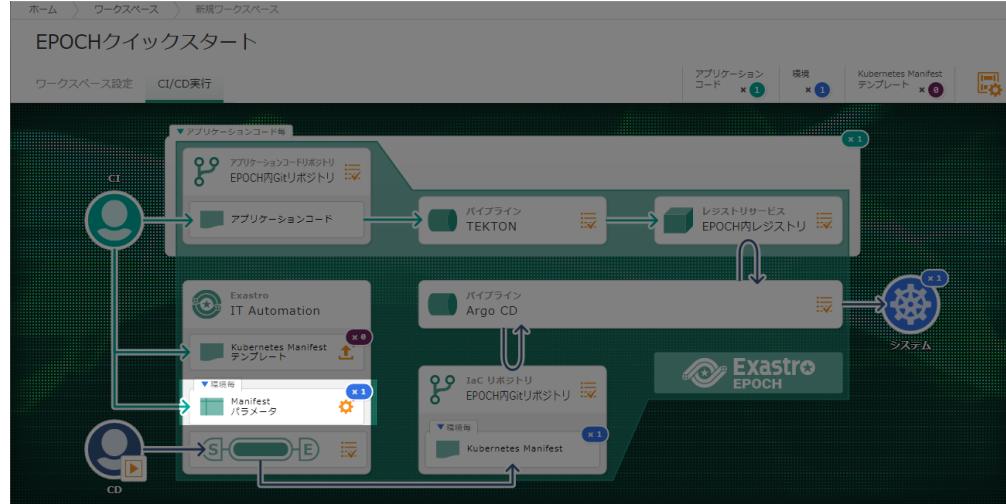
- Input the Manifest parameters needed for deploying.



# 4.5 CI/CD Workflow No.1 (11/25)

## Input Manifest parameters

- Follow the table below and input the Manifest parameters



ui-app.yaml

Item	Input contents(staging)	Input contents(production)	Description
<code>{{ param01 }}</code>	1	3	Replica numbers
<code>{{ image }}</code>	exastro/epochsampleappui	exastro/epochsampleappui	Container image
<code>{{ image_tag }}</code>	master.20210708183910	master.20210708183910	Container image tag
<code>{{ param02 }}</code>	31001	31003	Port number for the blue side (Blue-Green deployment)
<code>{{ param03 }}</code>	32001	32003	Port number for the green side (Blue-Green deployment)

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

```
8 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

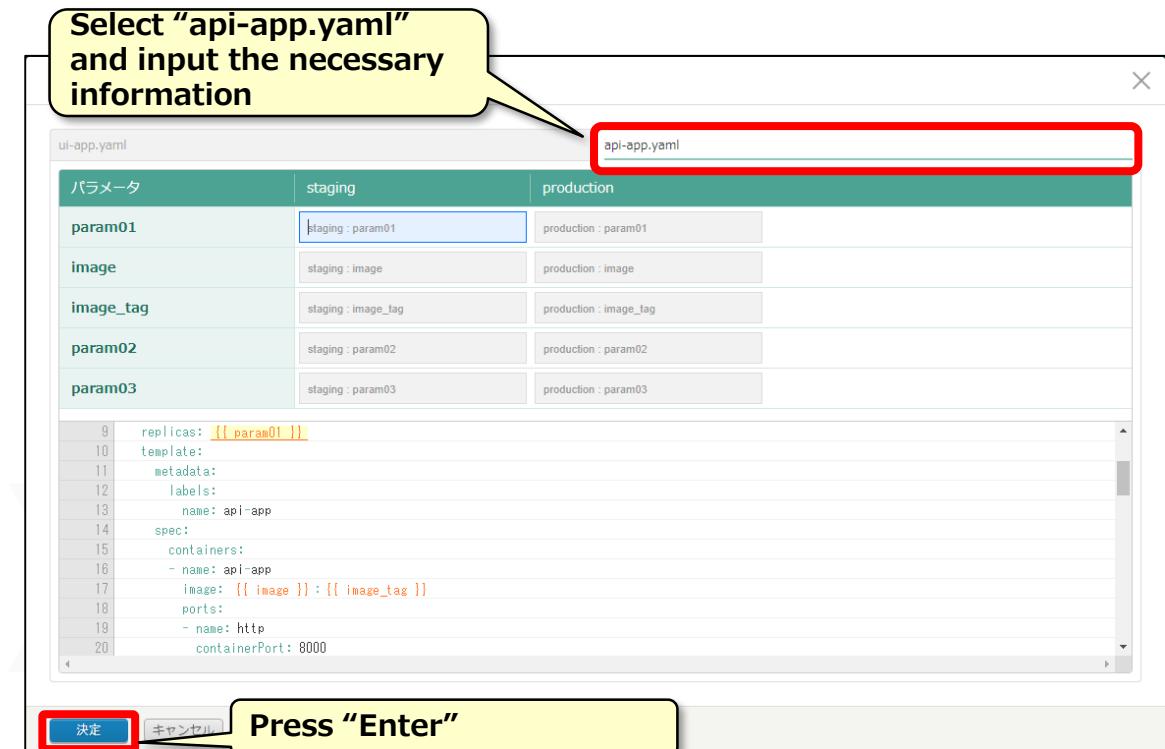
Select ui-app.yaml and input the Input contents from the table below.  
※The ui-app.yaml might be on the right tab, depending on the order you uploaded the files

Follow the Input contents from the table below.  
※Note that the values differ depending on the environment.

※The image\_tag for ui-app.yaml is the same one from the container image we built earlier.  
※We plan to make "image" and "image\_tag" selection items in the future.

# 4.5 CI/CD Workflow No.1 (12/25)

- Change to the api-app.yaml tab and repeat.



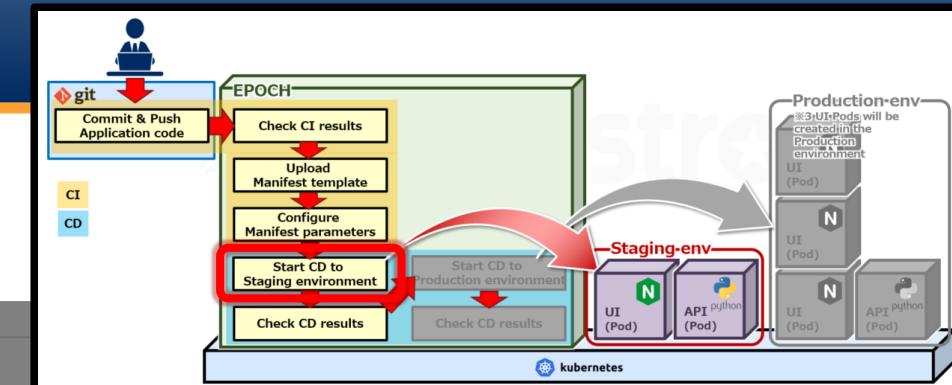
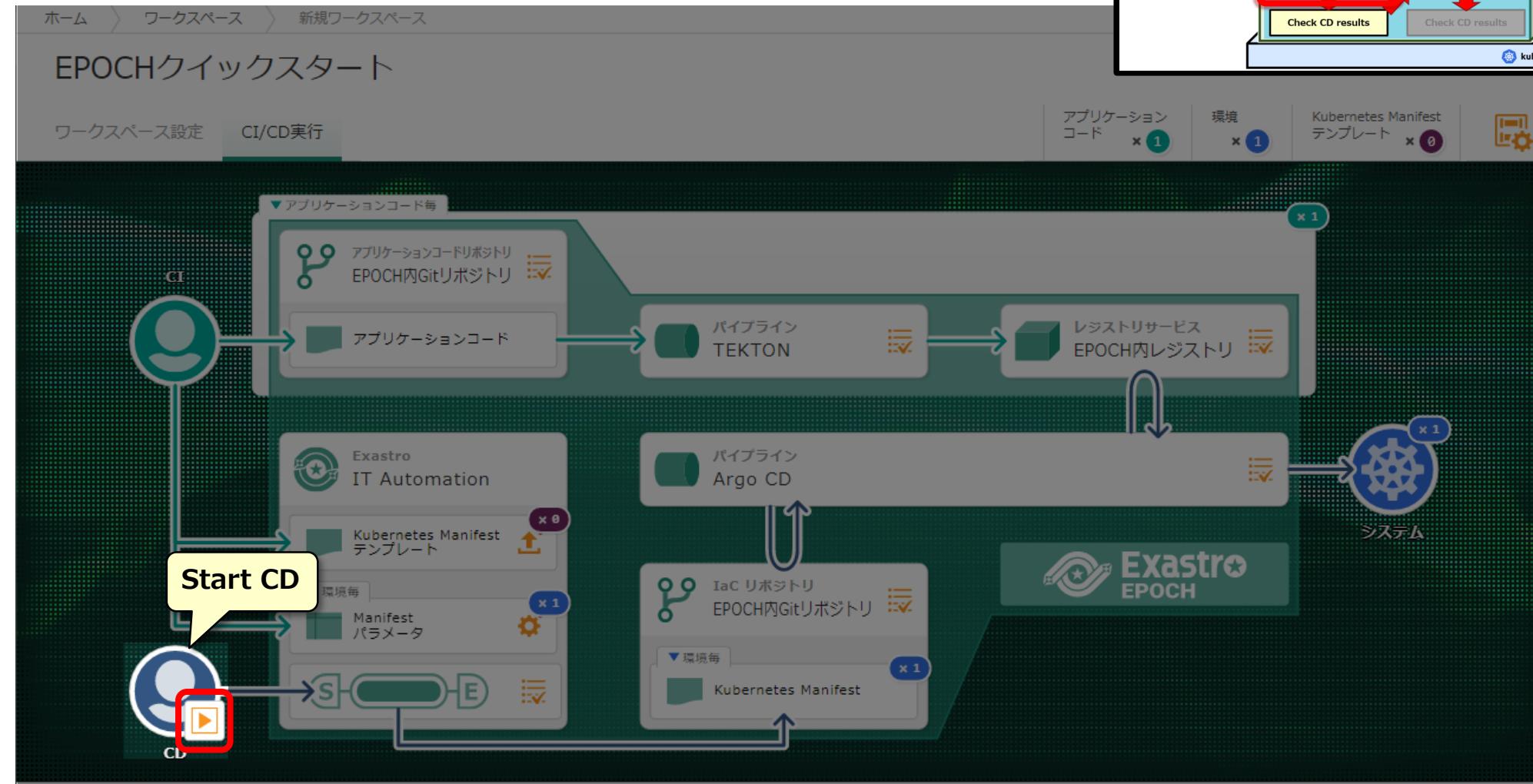
api-app.yaml

Item	Input contents(staging)	Input contents(production)	Description
{{ param01 }}	1	1	Replica number
{{ image }}	[Dockerhub account name]/epoch-sample-api	[Dockerhub account name]/epoch-sample-api	Container image
{{ image_tag }}	[Image tag name]	[Image tag name]	Container image tag acquired from the Registry service.
{{ param02 }}	31002	31004	Port number for the blue side (Blue-Green deployment)
{{ param03 }}	32002	32004	Port number for the green side (Blue-Green deployment)

# 4.5 CI/CD Workflow No.1 (13/25)

## Deploy to Staging environment

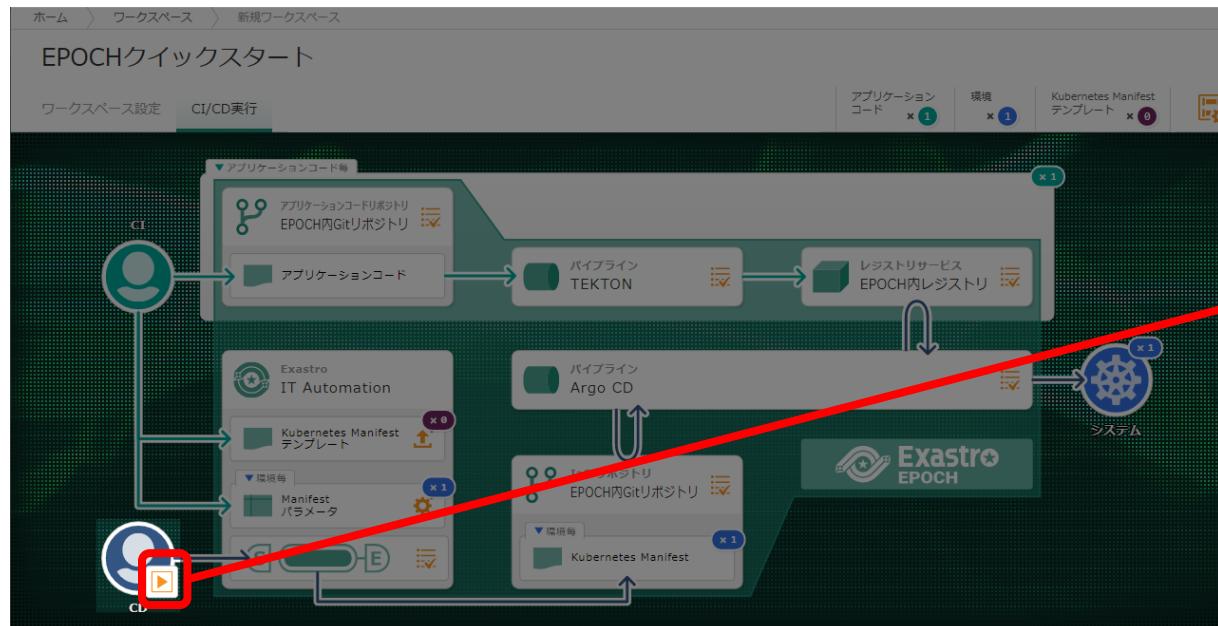
- Start CD and deploy to the Staging environment.



# 4.5 CI/CD Workflow No.1 (14/25)

## Start CD for the Staging environment

- Select the environment you want to deploy to.



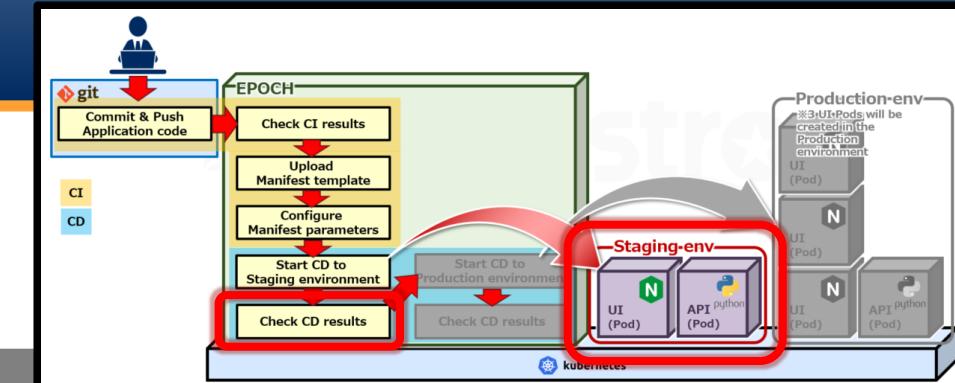
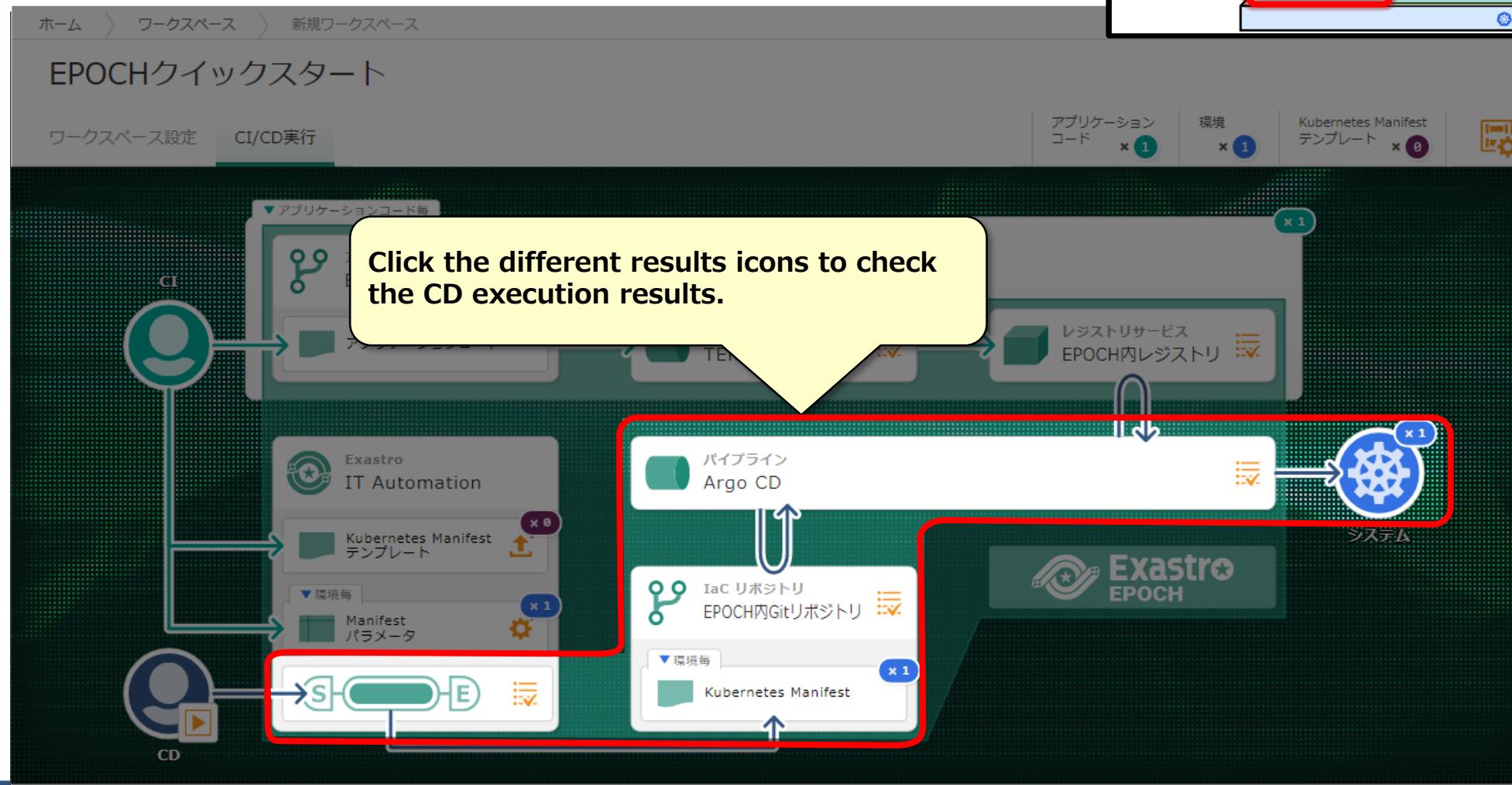
This is a modal dialog titled 'CD実行指定'. It contains fields for '実行条件' (Execution Conditions) and 'Manifest/パラメータ' (Manifest/Parameters). In the '実行条件' section, 'CD実行日時' is set to '即実行' (Run Now) and the date is '2021/07/08 14:17'. The '環境' (Environment) dropdown is set to 'staging', highlighted with a red border and a yellow callout 'Select "staging"'. In the 'Manifest/パラメータ' section, the environment name is 'staging', manifest repository is 'https://github.com/epoch-team/argocd\_manifest.git', Kubernetes API Server URL is 'https://kubernetes.default.svc', and Namespace is 'staging-app'. At the bottom are '実行' (Run) and 'キャンセル' (Cancel) buttons, with a yellow callout 'Press "Enter"' pointing to the '実行' button.

**The CD process for the Staging environment has started.  
Let us check that it is working as it should.**

# 4.5 CI/CD Workflow No.1 (15/25)

## Check the CD results

- We can check the CD execution results from Exastro IT-Automation and ArgoCD.



## 4.5 CI/CD Workflow No.1 (16/25)

Check that the Manifest file has been created (Staging environment)

- Open Exastro IT-Automation and check that the Manifest file has been registered to the IaC repository.

The screenshot shows the Exastro IT-Automation interface with several panels:

- Top Left Panel:** Shows the CI/CD pipeline. A red arrow points from the "Conductor" button in the pipeline to the "Filter" button in the Conductor list.
- Login Screen:** Displays the login credentials:  
LoginID: epoch-user  
Password: c2jthascR93ijdcyzwJY
- Conductor List:** Shows the Conductor list with a red box around the "Filter" button. A yellow box [1] indicates to go to this screen. A red arrow points from the "Filter" button here to the "Filter" button in the Conductor list.
- Conductor List (Details):** Shows the details of a "CD" record. A yellow box [3] indicates to press the "Details" button. A red box highlights the "詳細" (Details) button.
- Conductor Details:** Shows the Conductor details screen with a green "DONE" status for all nodes. A yellow box [4] indicates to check if all nodes say "DONE".

[1] Go to Conductor -> Conductor list

[2] Click "Filter"

[3] Press "Details" on the "CD" record.

[4] If all the nodes says "DONE", then the operation has ended successfully.

LoginID: epoch-user  
Password: c2jthascR93ijdcyzwJY

# 4.5 CI/CD Workflow No.1 (17/25)

Check the results of the ArgoCD Pipeline (Staging environment)

- Check that the Manifest shows up on Kubernetes

Let's get stuff deployed!

Username: admin  
Password: iYSCKzx2wvxJnn4dCNwN

Your browser might say that the connection is not secure. If that is the case, open up the “more information” tab and press “Access”.

Select staging

ArgoCD synchronizes every 3 minutes.

If the Sync status says “Sync OK”, the operation is complete.  
※ Make sure that the displayed date/time is after the CD was run.

Let us check the deployed sample application

## 4.5 CI/CD Workflow No.1 (18/25)

Check the Staging environment sample code.

- Access the sample application from the URL below

[http://\[Kubernetes master node IP address/Host name\]:31001/front-end.html](http://[Kubernetes master node IP address/Host name]:31001/front-end.html)

The screenshot shows a product catalog page with a grid of six T-shirts. Each T-shirt has the word 'EPOCH' printed on it. The T-shirts are arranged in two rows of three. The first row contains T-shirts A, B, and C. The second row contains T-shirts D, E, and F. Each T-shirt has a price label below it and a 'View details' button. The top left corner of the page has a 'Product Catalog' header and a 'Product Information' tab. On the right side of the page, there is a yellow callout box containing the instruction: 'Check that both "YEN" and "USD" is displayed and selectable from the Currency pulldown selection.' A red box highlights the currency dropdown menu, which currently shows 'YEN' as the selected option. At the bottom of the page, a large red box contains the text: 'Now let's deploy to the Production environment'.

商品カタログ

製品情報

表示価格 YEN  
YEN  
USD

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

TシャツF  
3,600 円(税込)  
詳細はこちら

TシャツG  
6,500 円(税込)  
詳細はこちら

TシャツH  
4,250 円(税込)  
詳細はこちら

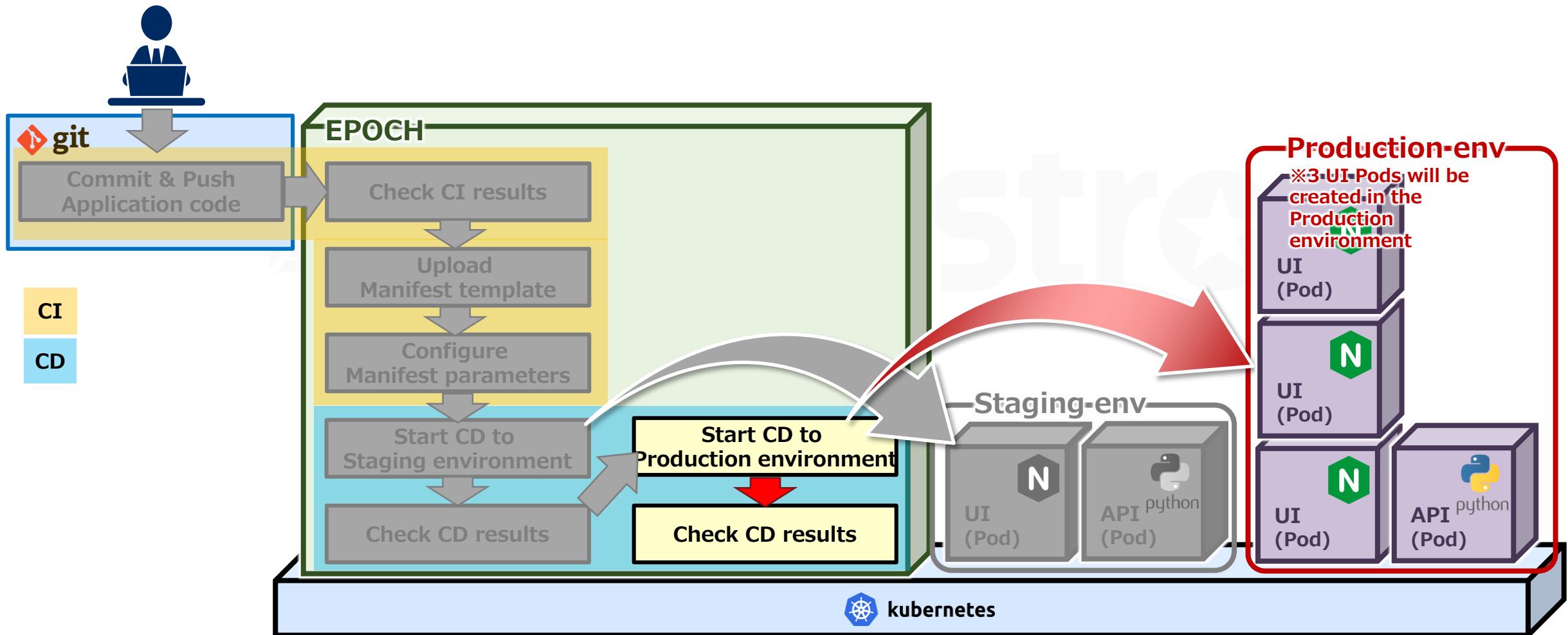
Check that both "YEN" and "USD" is displayed and selectable from the Currency pulldown selection.

Now let's deploy to the Production environment

## 4.5 CI/CD Workflow No.1 (19/25)

### Deploy to Production environment

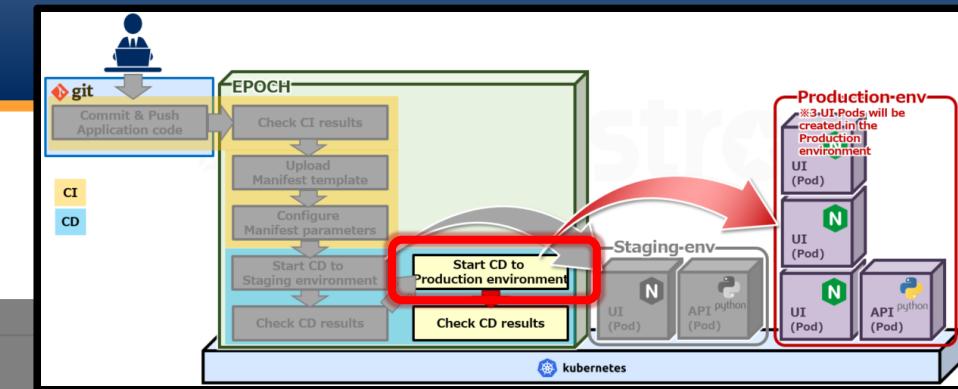
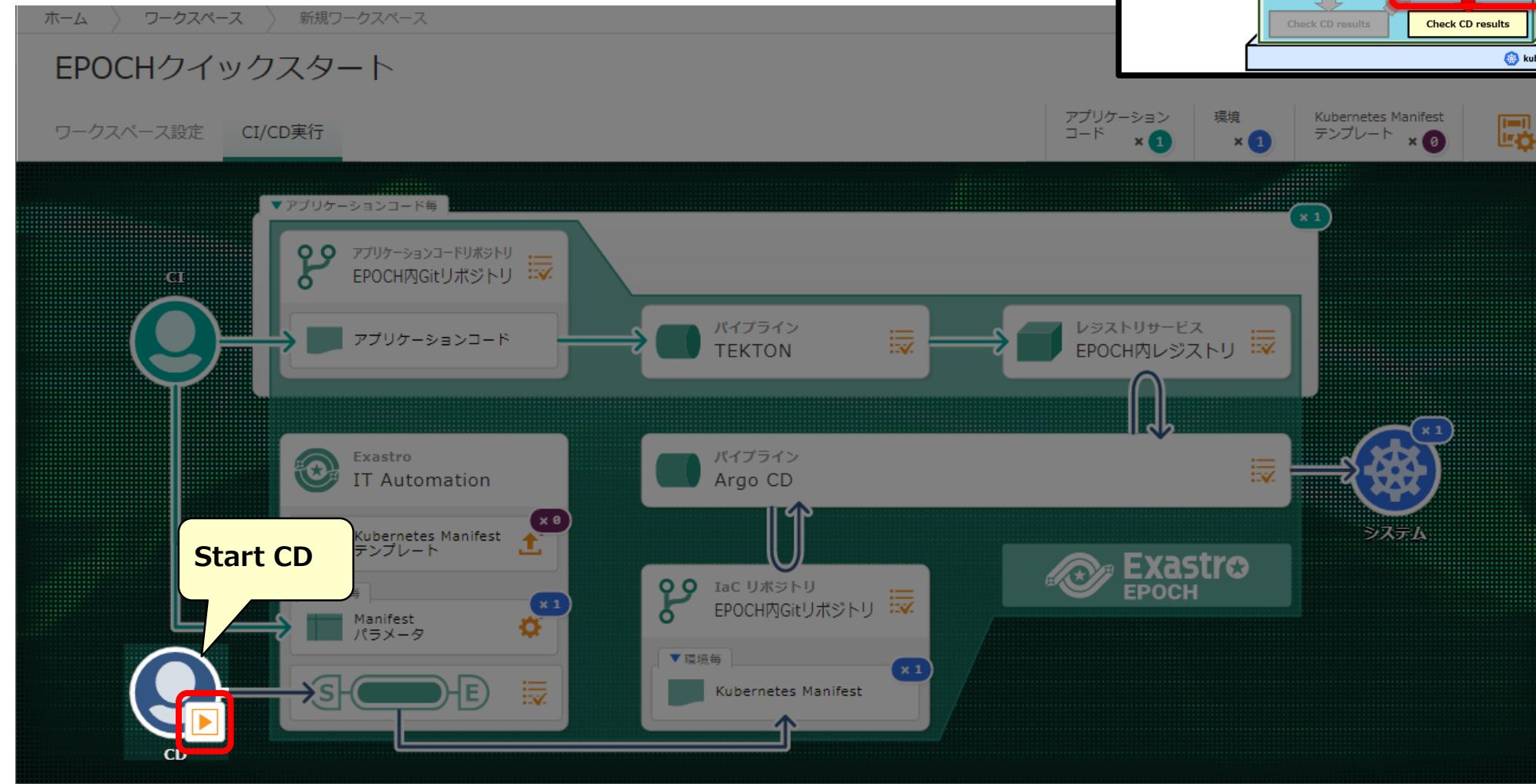
Now that we have deployed to the Staging environment and checked that everything functions as it should, we can now deploy to the Production environment.



# 4.5 CI/CD Workflow No.1 (20/25)

## Deploy to the Production environment

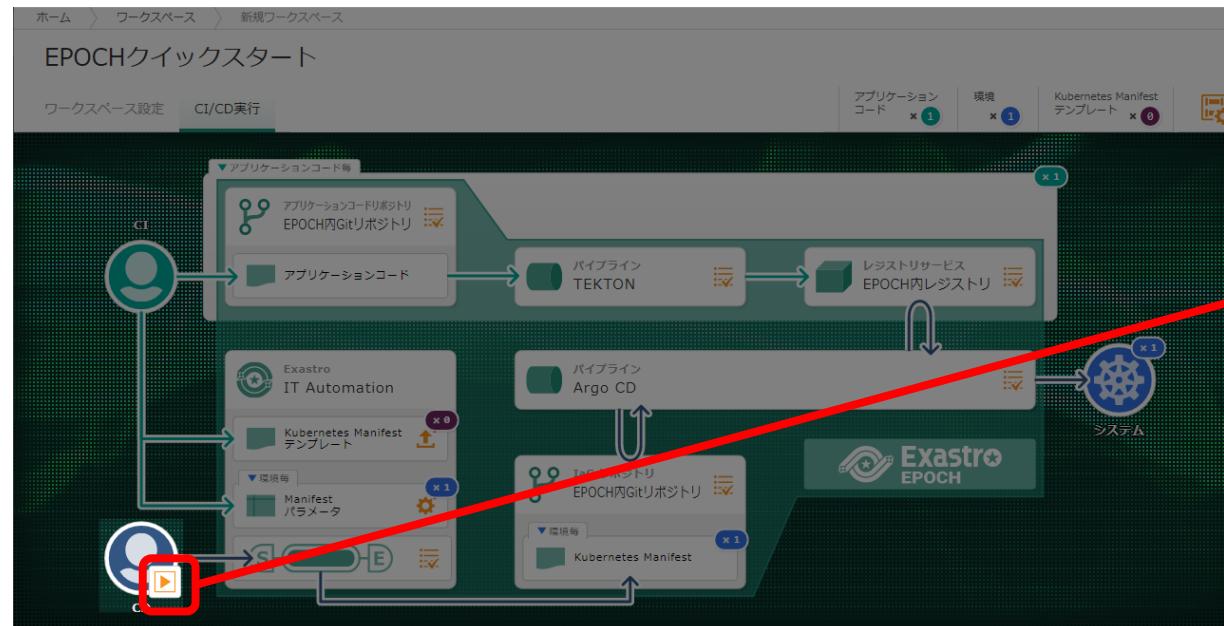
- Start the CD and deploy to the Production environment.



# 4.5 CI/CD Workflow No.1 (21/25)

## Start CD for the Production environment

- Select the environment you want to deploy to.



The dialog box is titled 'CD実行指定'. It contains the following fields:

- 実行条件**: Includes 'CD実行日時' (即実行 selected, 2021/07/08 14:17) and '環境' (production selected).
- Manifestパラメータ**: Shows 'Manifestリポジトリ' as [https://github.com/epoch-team/argocd\\_manifest-184.git](https://github.com/epoch-team/argocd_manifest-184.git).
- ArgoCDパイプライン**: Shows '環境名' as 'production', 'Manifestリポジトリ' as [https://github.com/epoch-team/argocd\\_manifest-184.git](https://github.com/epoch-team/argocd_manifest-184.git), 'Kubernetes API Server URL' as <https://kubernetes.default.svc>, and 'Namespace' as 'production-app'.

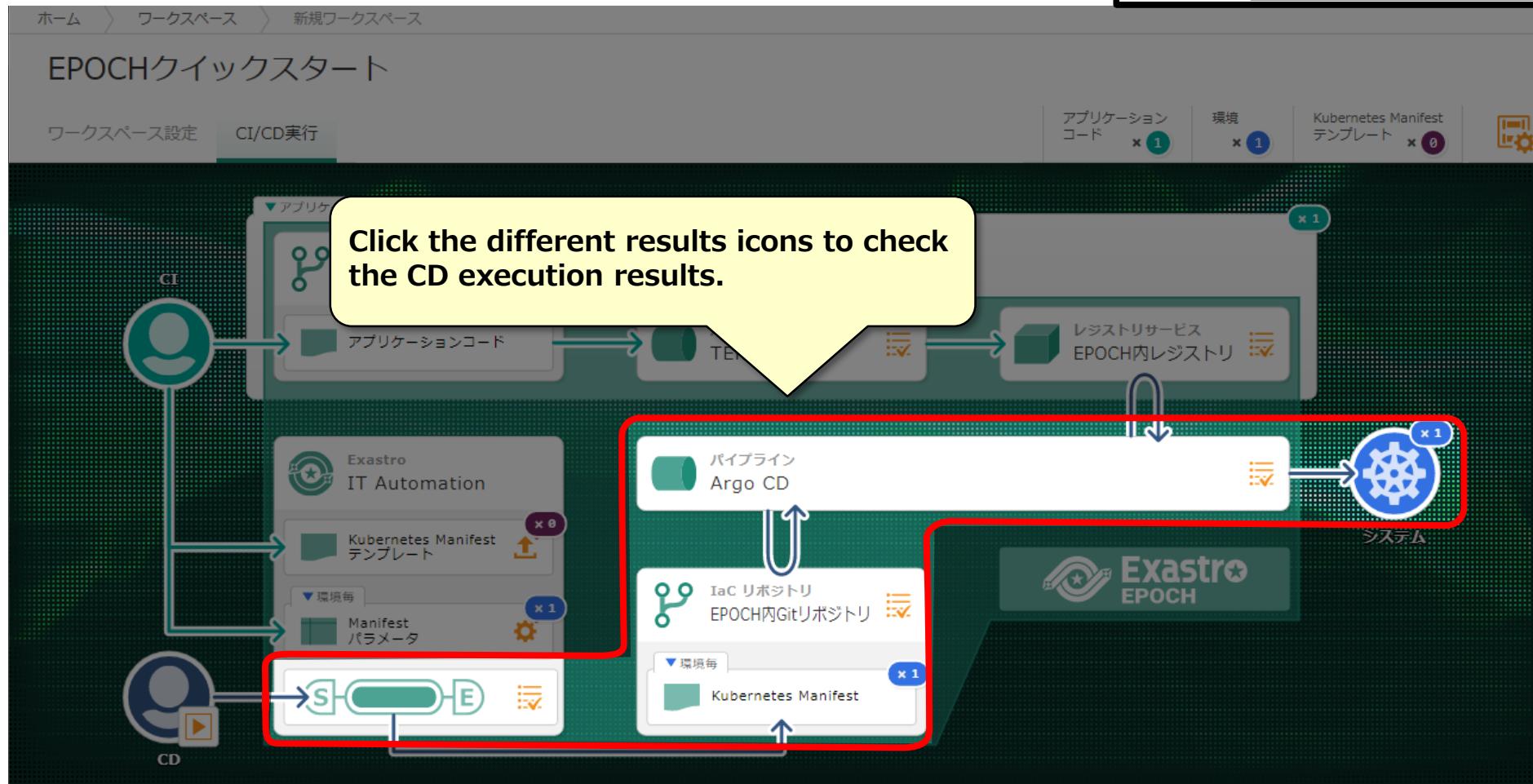
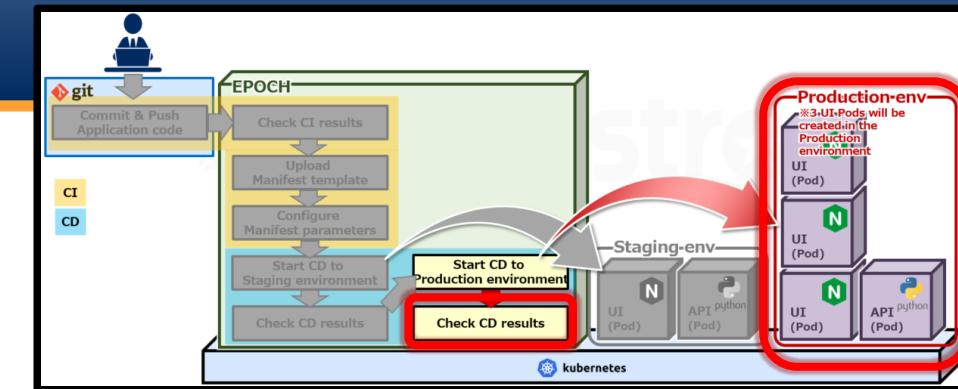
A yellow callout box with a red border and a red arrow points to the 'Select "production"' field, with the text 'Select "production"' inside. Another yellow callout box with a red border and a red arrow points to the '実行' (Execute) button, with the text 'Press "Enter"' inside.

**The CD process for the Production environment has started.  
Let us check that it is working as it should.**

# 4.5 CI/CD Workflow No.1 (22/25)

## Check the CD results (Production)

- We can check the CD execution results from Exastro IT-Automation and ArgoCD.



## 4.5 CI/CD Workflow No.1 (23/25)

Check that the Manifest file has been created (Production environment)

- Open Exastro IT-Automation and check that the Manifest file has been registered to the IaC repository.

The image shows the Exastro IT-Automation interface with several windows open:

- Top Left:** A dashboard showing the CI/CD pipeline. A red arrow points from the "CD" section of the pipeline to the "Conductor" list window.
- Top Right:** An "Exastro IT Automation" login screen with fields for "LoginID" and "Password". A yellow callout box contains the credentials:

LoginID: epoch-user  
Password: c2jthascR93ijdcyzwJY
- Middle Left:** The "Conductor" list window. A yellow callout box [1] says "[1] Go to Conductor -> Conductor list". A red box highlights the "Conductor一覧" button. Another yellow callout box [2] says "[2] Click "Filter"" and points to the "フィルタ" button. A third yellow callout box [3] says "[3] Press "Details" on the "CD" record." and points to the "詳細" button next to a "CD" entry in the list.
- Middle Right:** The "Conductor" list window showing the status of a task. A yellow callout box says "If all the nodes says \"DONE\", then the operation has ended successfully." and points to a node labeled "DONE".

# 4.5 CI/CD Workflow No.1 (24/25)

Check the results of the ArgoCD Pipeline (Production environment)

- Check that the Manifest shows up on Kubernetes

The screenshot illustrates the EPOCH CI/CD pipeline and its integration with ArgoCD and Kubernetes.

**EPOCH CI/CD Pipeline:** The top left shows the pipeline stages: Application Code → Pipeline TEKTON → Registry Service EPOCH Internal Registry. A red arrow points from the Argo CD stage to the ArgoCD synchronization interface.

**ArgoCD Synchronization:** The middle section shows the ArgoCD interface with the message "Let's get stuff deployed!" and a cartoon character. A red box highlights the "SIGN IN" button. A yellow callout box provides login credentials: **Username: admin** and **Password: iYSCKzx2wvxJnn4dCNwN**. Another yellow callout box states: "Your browser might say that the connection is not secure. If that is the case, open up the “more information” tab and press “Access”."

**Application Status:** The bottom left shows the "Applications" screen with two entries: **production** (Synced, Healthy) and **staging** (OutOfSync, Missing). A yellow callout box says "Select “production”". Red arrows point from the production entry to the ArgoCD interface and the Kubernetes cluster view.

**Kubernetes Cluster View:** The bottom right shows the Kubernetes dashboard with the "staging" namespace selected. It displays various services and pods, with a yellow callout box stating: "ArgoCD synchronizes every 3 minutes." Another yellow callout box says: "If the Sync status says “Sync OK”, the operation is complete. ※Make sure that the displayed date/time is after the CD was run."

**Final Callout:** A large red box at the bottom center contains the text: **Let us check the deployed sample application**.

## 4.5 CI/CD Workflow No.1 (25/25)

Check the Production environment sample code.

- Access the sample application from the URL below

[http://\[Kubernetes master node IP address/Host name\]:31003/front-end.html](http://[Kubernetes master node IP address/Host name]:31003/front-end.html)

商品カタログ

表示価格

Tシャツ	価格	操作
TシャツA	1,000 円(税込)	<input type="button" value="詳細はこちら"/>
TシャツB	2,000 円(税込)	<input type="button" value="詳細はこちら"/>
TシャツC	7,500 円(税込)	<input type="button" value="詳細はこちら"/>
TシャツD	4,700 円(税込)	<input type="button" value="詳細はこちら"/>
TシャツE	1,200 円(税込)	<input type="button" value="詳細はこちら"/>
TシャツF	3,600 円(税込)	<input type="button" value="詳細はこちら"/>
TシャツG	6,500 円(税込)	<input type="button" value="詳細はこちら"/>
TシャツH	4,250 円(税込)	<input type="button" value="詳細はこちら"/>

**Check that the screen displayed is the same as the one in the Staging environment.**

This concludes the first CI/CD Workflow. In the next section, CI/CD Workflow No.2, we will modify the application code and deploy it once more.

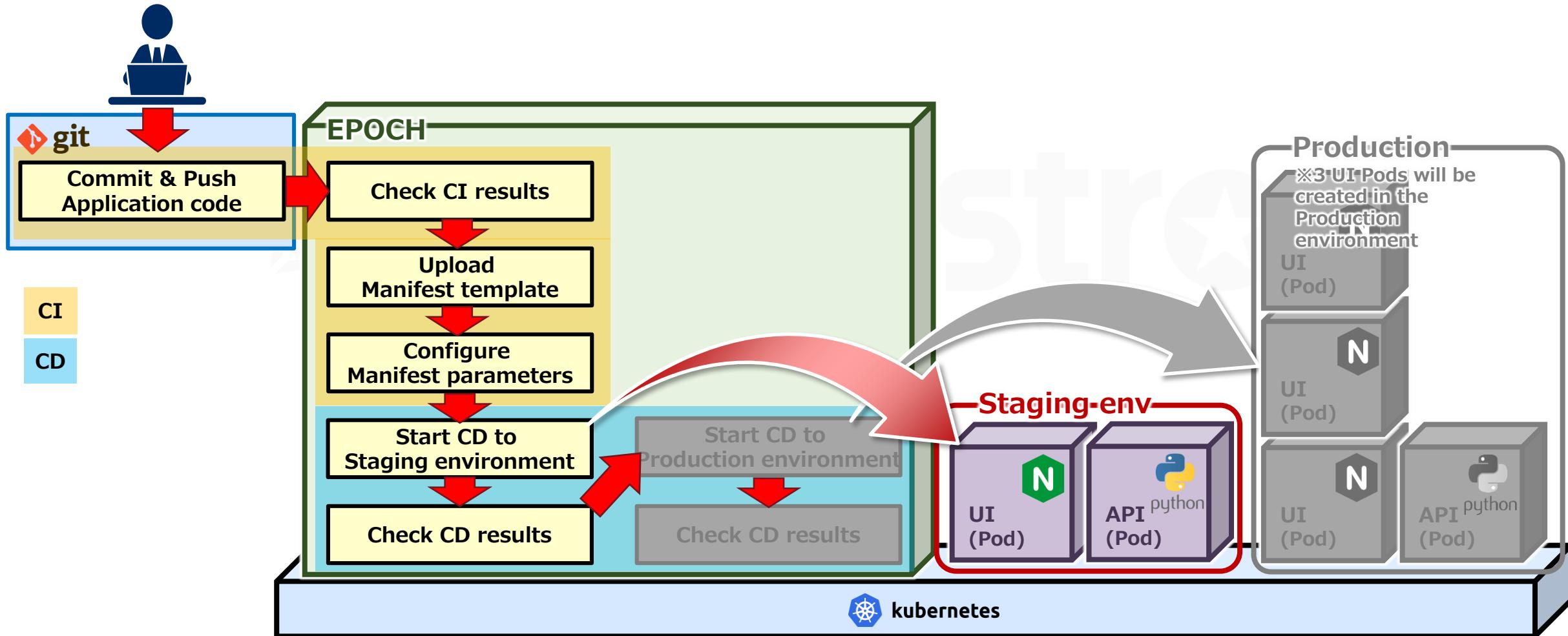
# CI/CD Workflow No.2



## 4.6 CI/CD Workflow No.2 (1/21)

### Deploy to the Staging environment

In the second CI/CD Workflow, the user will modify the application code, commit and push it to the Staging environment and finally deploy it.



## 4.6 CI/CD Workflow No.2 (2/21)

### Modify application code

Modify the application code and start the second CI/CD process.

In this part of the tutorial, we will make it so Euro can be selected as a currency.

Use your code editor and open the following files from the cloned application code repository and modify them.

- File 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 }
```

} Add line 9~12

- File 2② : api-app/data/rate.json

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

, Add a comma to the end of line 2  
} Add line 3

## 4.6 CI/CD Workflow No.2 (3/21)

### Commit and push the application code

Take the modified application code and commit and push it.

Run the following Git commands

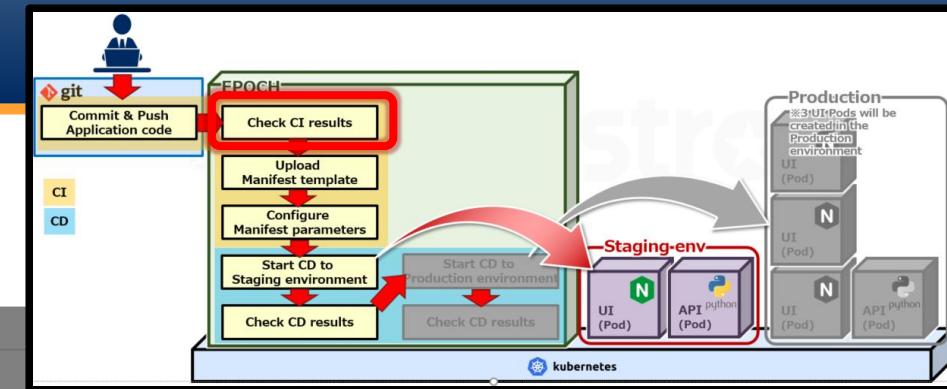
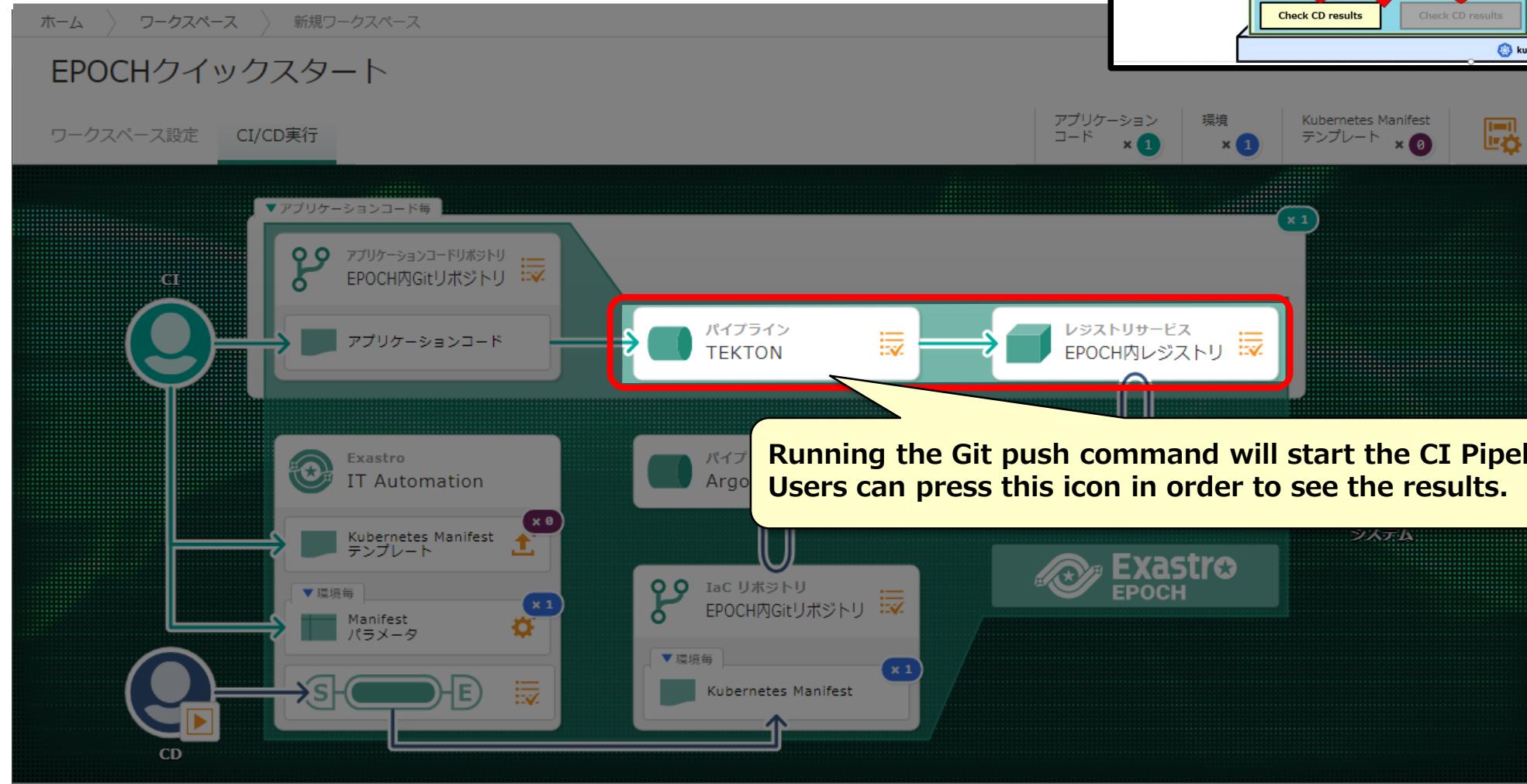
```
> cd "[clone destination folder]"
> cd epoch-sample-app
> git add .
> git commit -m "addcurrency(EUR)"
> git push origin main
```

※ If you are asked to authenticate yourself when you are running the git push command, input the necessary GitHub account information.

## 4.6 CI/CD Workflow No.2 (4/21)

### Check CI results

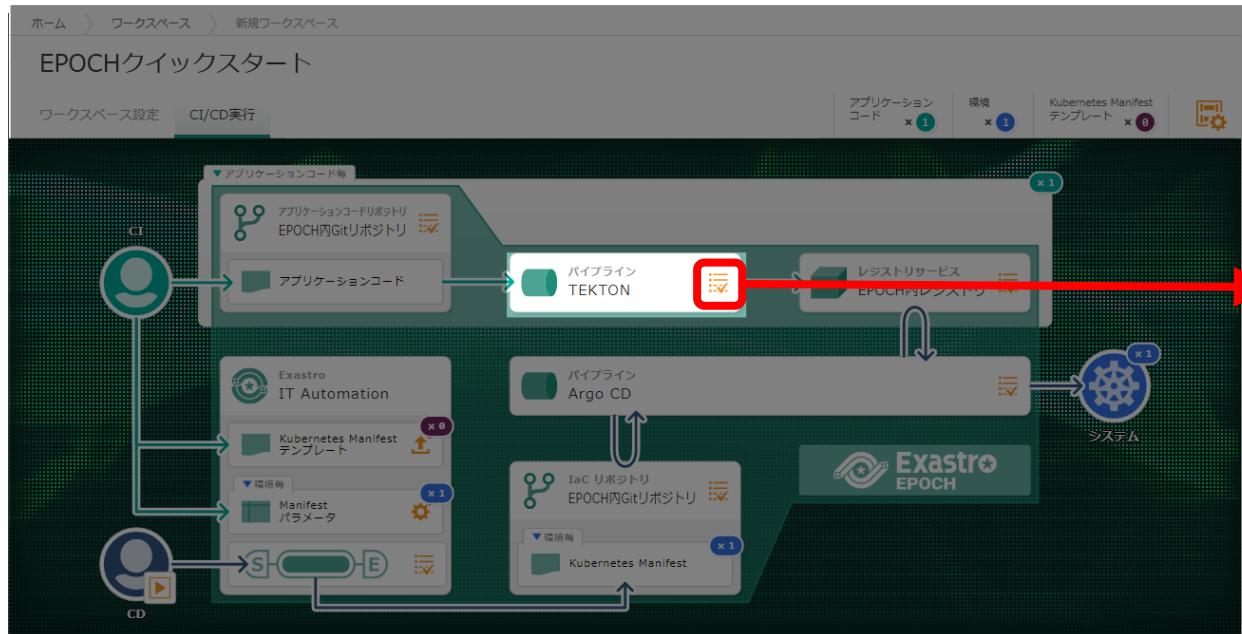
- Check the build results of the Application code.



## 4.6 CI/CD Workflow No.2 (5/21)

### Check the TEKTON Pipeline

- Check the TEKTON pipeline that the build has ended successfully.



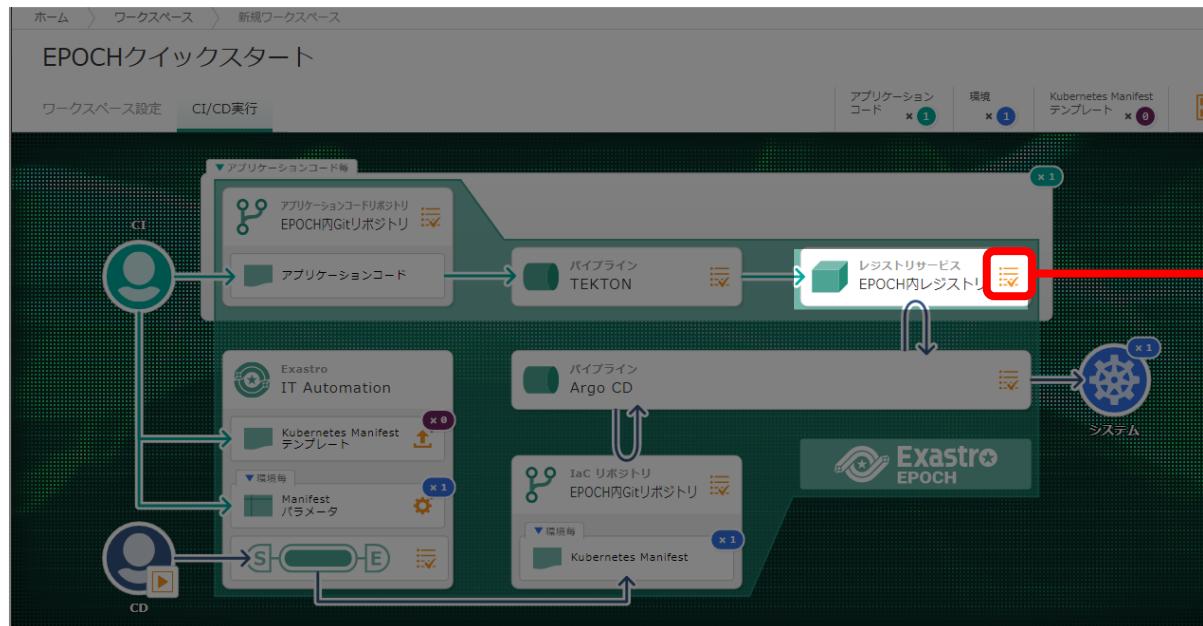
The screenshot shows the Tekton Dashboard's PipelineRuns page. It lists a single pipeline run named "build-and-push-pipeline-r..." with a status of "Success". A callout box on the right provides instructions: "Users can check the CI Pipeline operations from the TEKTON dashboard. By pressing the name of the operation, users can see more information regarding the CI Pipeline contents." Another callout below states: "If the 'Status' displays a green checkmark, the operation has ended successfully."

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

## 4.6 CI/CD Workflow No.2 (6/21)

### Check the Container image tag name

- Open the registry service screen and check the tag of the built container image.



The Dockerhub screenshot shows the repository '/epoch-sample-api'. It lists two tags:

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

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

**Open the Dockerhub screen and check the epoch-sample-api tag.**  
※If the image isn't registered, check the results from the TEKTON pipeline

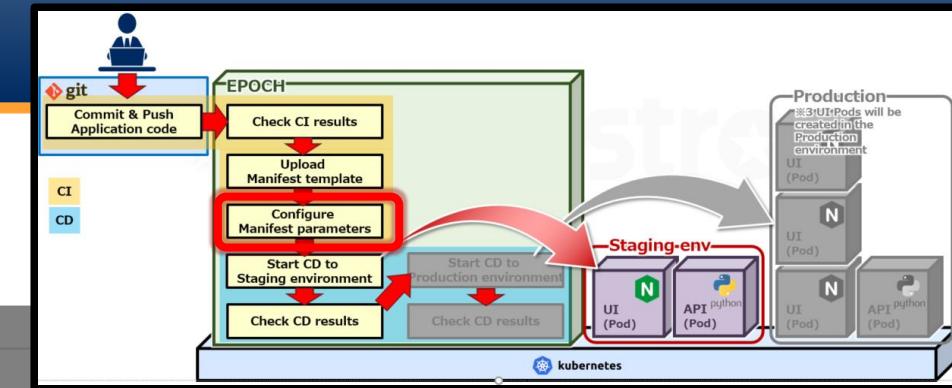
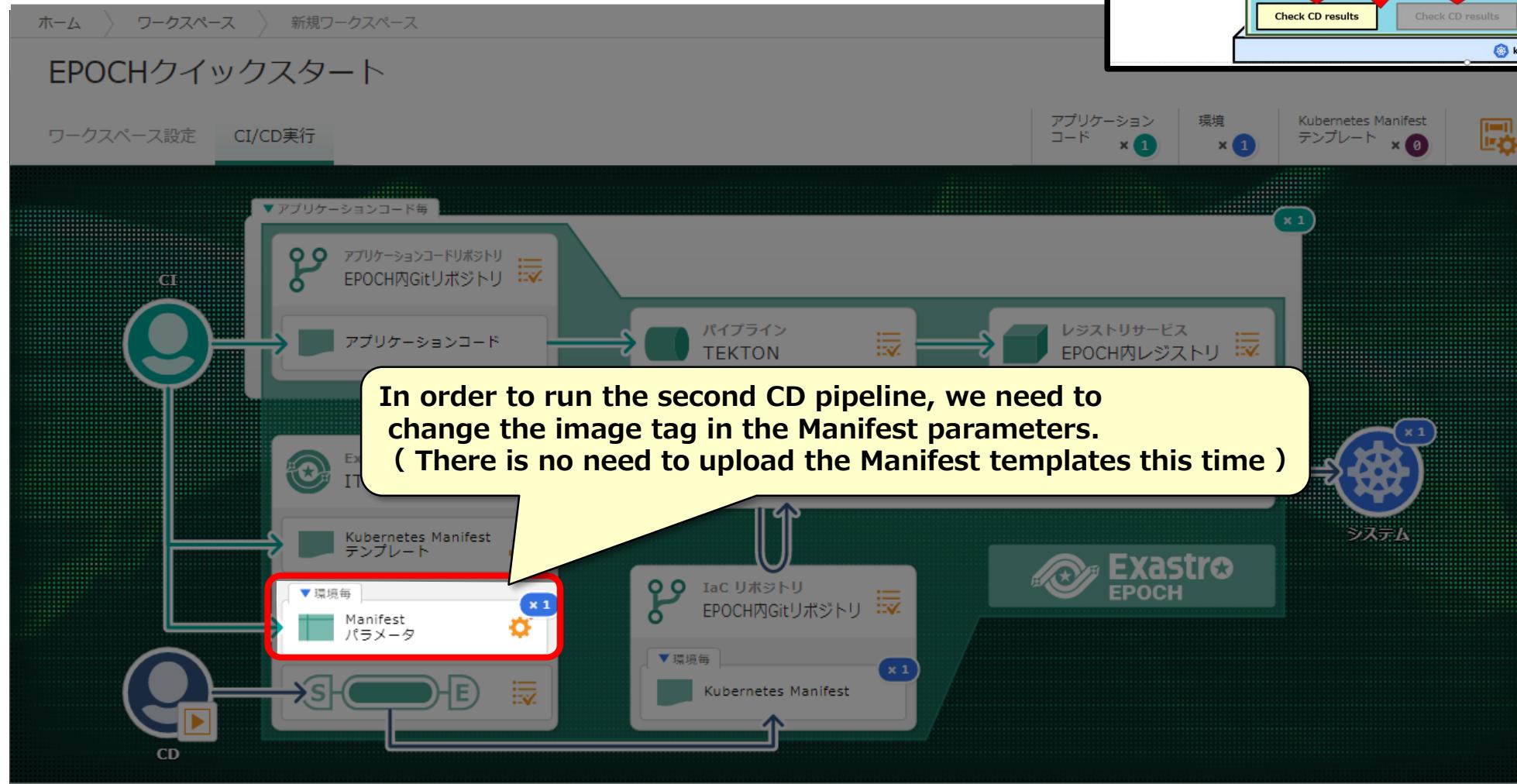
**The Image tag will be used in the next step where we input it into the Manifest parameter (image\_tag), so make sure to save it.**

※We plan to make it possible for users to select the image\_tag created in the Pipeline.

## 4.6 CI/CD Workflow No.2 (7/21)

### Upload Manifest template

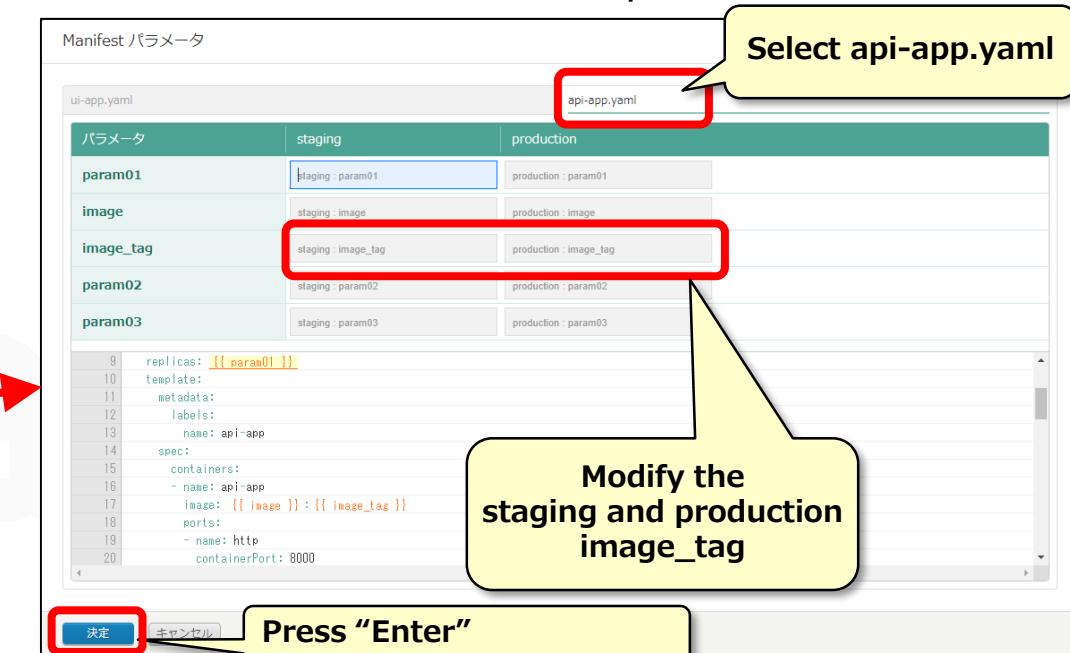
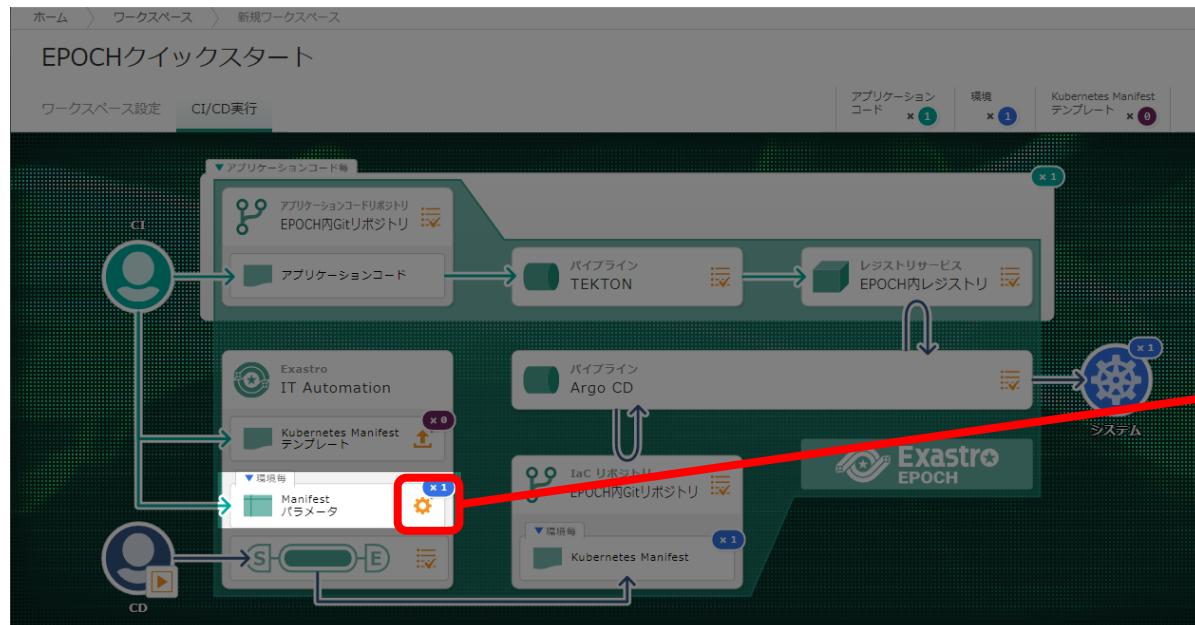
- Upload the Manifest template we downloaded earlier



## 4.6 CI/CD Workflow No.2 (8/21)

### Modify Manifest parameter(image\_tag)

- Change the image tag name for both the staging and the production environment Manifest parameters.



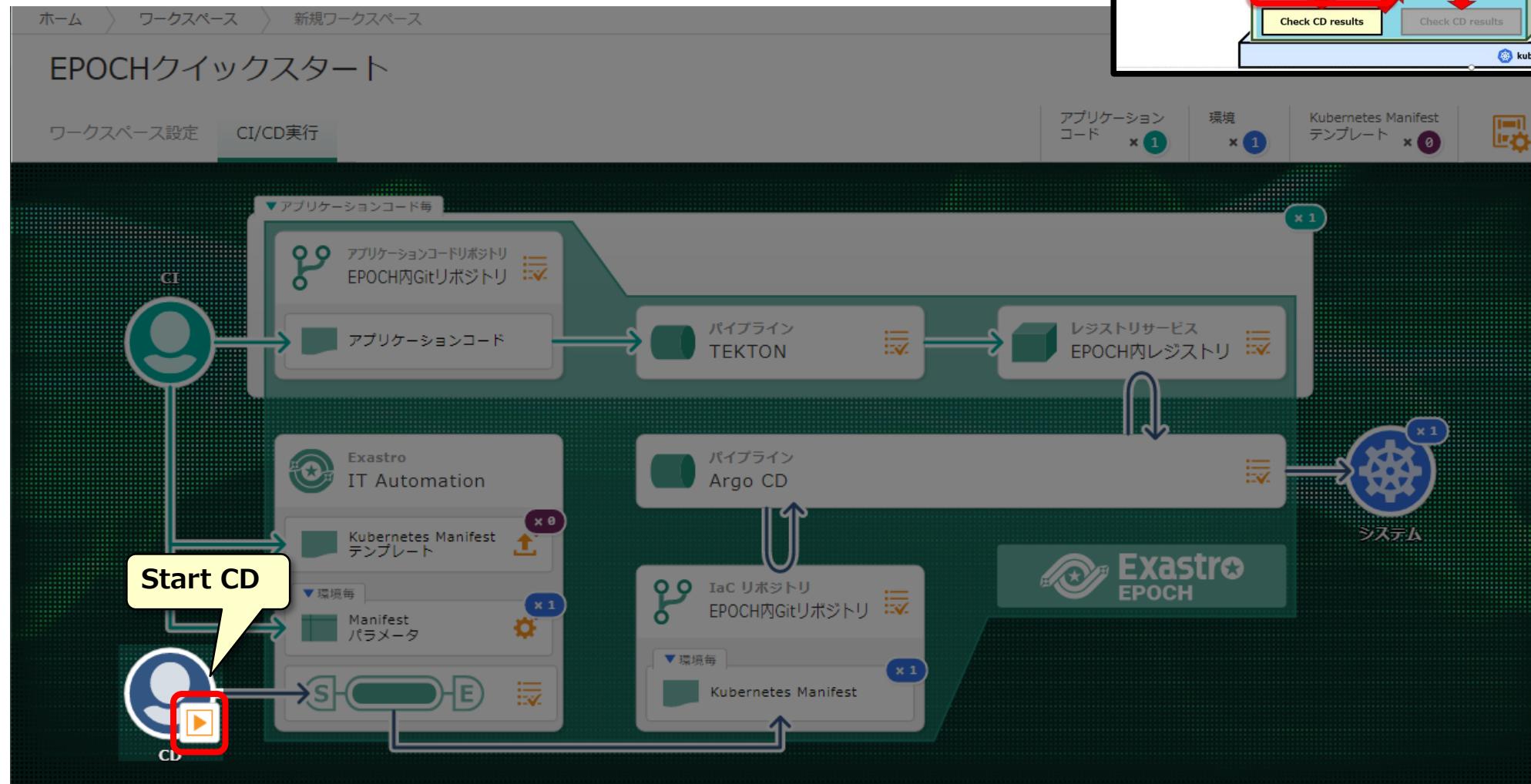
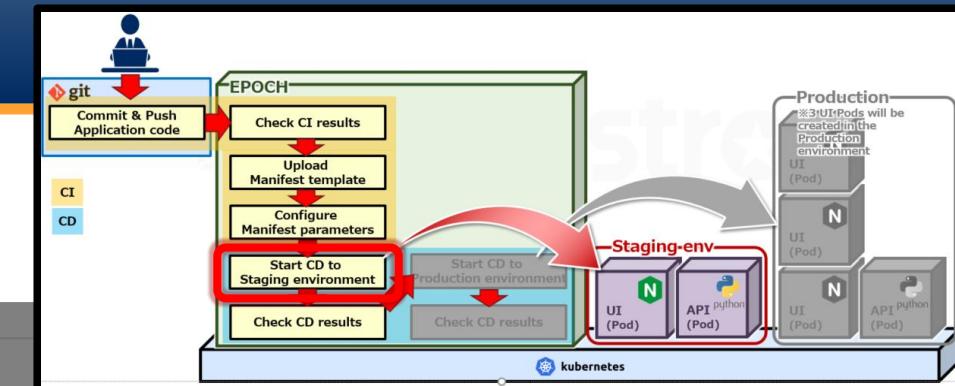
api-app.yaml

Item	Input contents(staging)	Input contents(production)	Description
<code>{{ image_tag }}</code>	[The most recent image tag checked in the registry service]	[The most recent image tag checked in the registry service]	Container image tag

## 4.6 CI/CD Workflow No.2 (9/21)

### Deploy to Staging environment

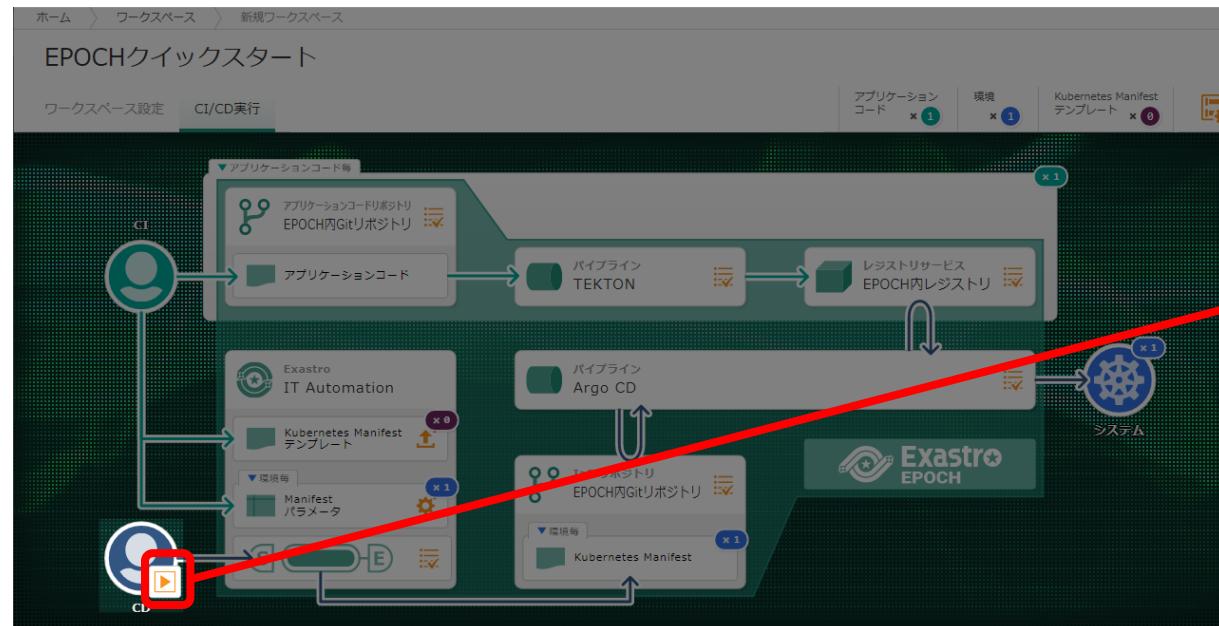
- Start CD and deploy to the Staging environment.



## 4.6 CI/CD Workflow No.2 (10/21)

### Start CD for the Staging environment

- Select your desired environment and start deploying.



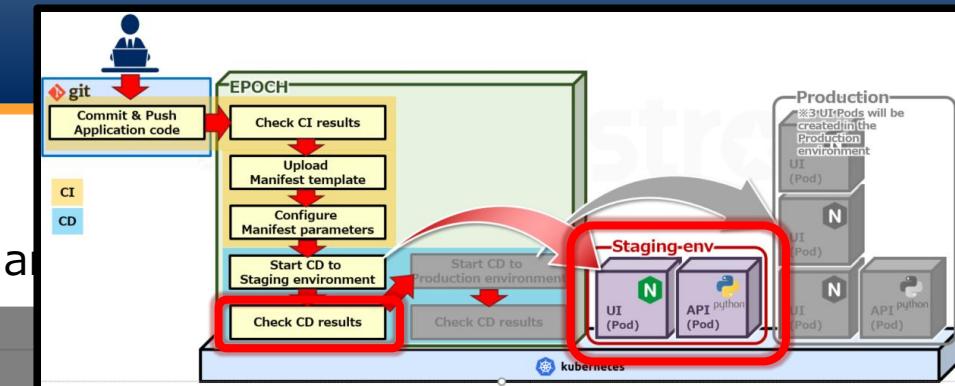
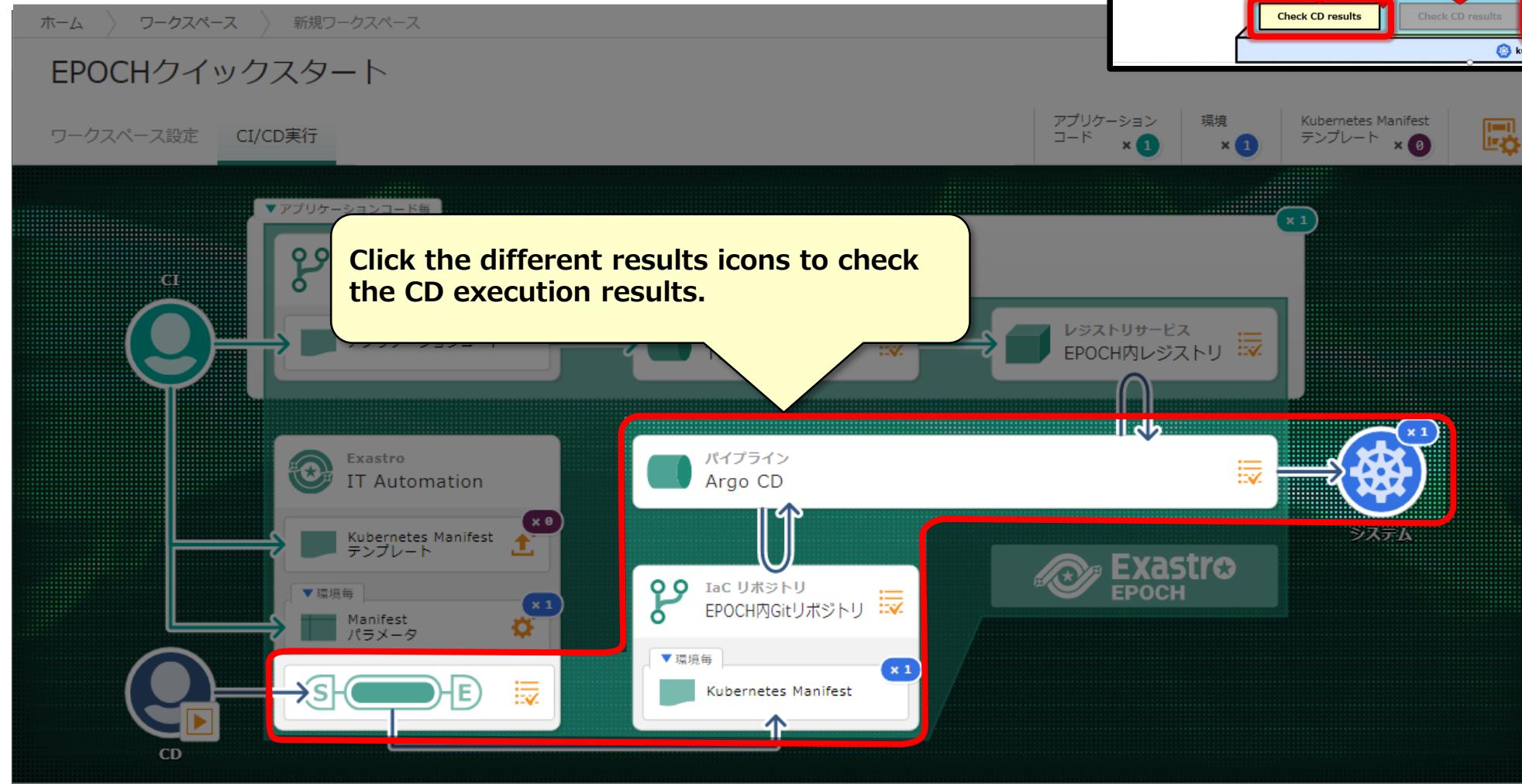
The dialog box shows the configuration for the deployment. Under '実行条件' (Execution Conditions), '即実行' (Run Now) is selected. In the '環境' (Environment) dropdown, 'staging' is chosen. A yellow callout box points to this dropdown with the text 'Select "staging"'. Below it, under 'Manifest/パラメータ' (Manifest/Parameters), the environment name is set to 'staging'. A yellow callout box points to the '実行' (Execute) button with the text 'Press "Enter"'. The dialog also lists the Manifest repository URL and Kubernetes API server URL.

**The CD process for the Staging environment has started.  
Let us check that it is working as it should.**

## 4.6 CI/CD Workflow No.2 (11/21)

### Check the CD results

- We can check the CD execution results from Exastro IT-Automation and EPOCH.



## 4.6 CI/CD Workflow No.2 (12/21)

Check that the Manifest file has been created (Staging environment)

- Open Exastro IT-Automation and check that the Manifest file has been registered to the IaC repository.

The image shows the Exastro IT-Automation interface with several windows open:

- Top Left:** A dashboard showing the CI/CD pipeline. A red arrow points from the "Manifest" node in the pipeline to the "Conductor" list screen below.
- Top Right:** An "Exastro IT Automation" login window. A yellow callout box contains the login credentials:

LoginID: epoch-user  
Password: c2jthascR93ijdcyzwJY
- Middle Left:** The "Conductor" list screen. A yellow callout box [1] says "[1] Go to Conductor -> Conductor list". A red box highlights the "Conductor一覧" button in the sidebar. Another yellow callout box [2] says "[2] Click "Filter"" with a red box around the "フィルタ" button. A third yellow callout box [3] says "[3] Press "Details" on the "CD" record." with a red box around the "詳細" button in the table.
- Middle Right:** The "Conductor" list screen showing the status of a task. A yellow callout box says "If all the nodes says \"DONE\", then the operation has ended successfully." with a red box around the "DONE" status in the table.
- Bottom Right:** A screenshot of the "Checking" step in the Conductor workflow, showing a green "DONE" status.

## 4.6 CI/CD Workflow No.2 (13/21)

Check the results of the ArgoCD Pipeline (Staging environment)

- Check that the Manifest shows up on Kubernetes

The diagram illustrates the CI/CD workflow and deployment status across various platforms:

- EPOCH UI (Top Left):** Shows the CI/CD pipeline stages: Application Code → Pipeline TEKTON → Registry Service EPOCH Internal Registry. A red arrow points from the Argo CD stage to the Argo CD sign-in screen.
- Argo CD Sign-in Screen (Top Right):** Displays the Argo CD login interface with the following credentials:
  - Username: admin
  - Password: iYSCKzx2wvxJnn4dCNwNA callout notes: "Your browser might say that the connection is not secure. If that is the case, open up the “more information” tab and press “Access”."
- Argo CD Applications (Bottom Left):** Shows two environments: production and staging. A callout says: "Select “staging”". Red boxes highlight the staging environment details:
  - Project: default
  - Status: Missing (red)
  - Repository: https://github.com/shiota-2021/epoch...
  - Path: /
  - Destination: in-cluster
  - Namespace: epoch-sample-app-stagingA red arrow points from the Argo CD Applications screen to the Argo CD Sync Status screen.
- Argo CD Sync Status (Bottom Middle):** Shows the sync status for the staging environment:
  - Synced to HEAD (0d07ae)
  - Authored by epoch-team <...>
  - Succeeded a minute ago (17:18:26 GMT+0900)
  - Authored by epoch-team <epoch-cms-jp.nec.com>
  - Manifest Push, 202107081712A callout says: "ArgoCD synchronizes every 3 minutes."
- Kubernetes Cluster (Bottom Right):** Shows the deployed sample application resources:
  - ui-app-com (cm)
  - api-app-bluegreen (svc)
  - api-app-bluegreen (svc)
  - ui-app-bluegreen-active (svc)
  - ui-app-bluegreen-preview (svc)
  - api-app-78c75b9b78-hlhk4 (pod)A callout says: "If the Sync status says “Sync OK”, the operation is complete. ※Make sure that the displayed date/time is after the CD was run."

**Let us check the deployed sample application**

## 4.6 CI/CD Workflow No.2 (14/21)

Check the Staging environment sample code.

- Access the sample application from the URL below

[http://\[Kubernetes master node IP address/Host name\]:31001/front-end.html](http://[Kubernetes master node IP address/Host name]:31001/front-end.html)

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 € 48.45 € 32.10

詳細はこちら 詳細はこちら 詳細はこちら 詳細はこちら

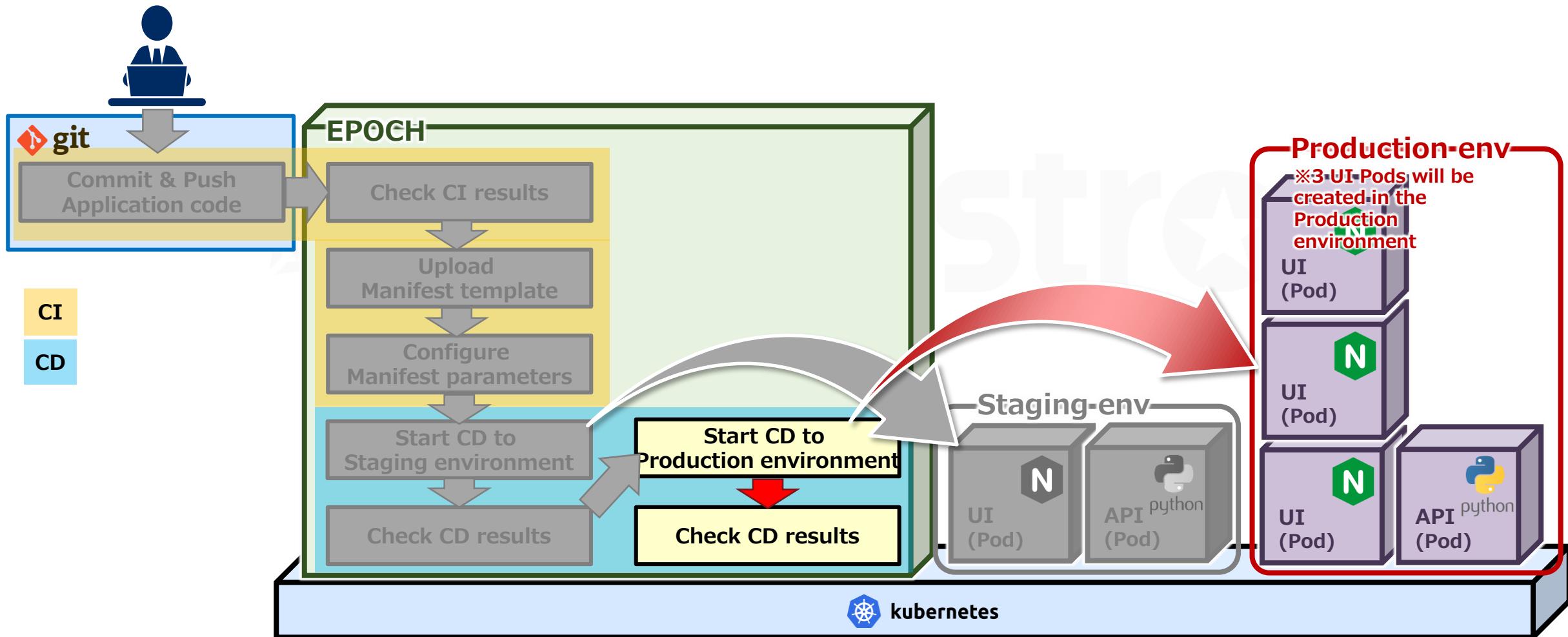
Check that "YEN", "USD", and the newly added "EUR" is displayed and selectable from the Currency pulldown selection.

Now let's deploy to the Production environment

## 4.6 CI/CD Workflow No.2 (15/21)

### Deploy to Production environment

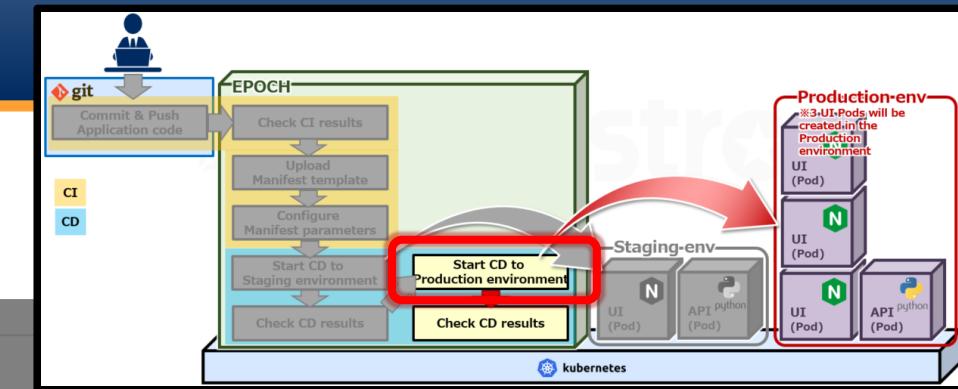
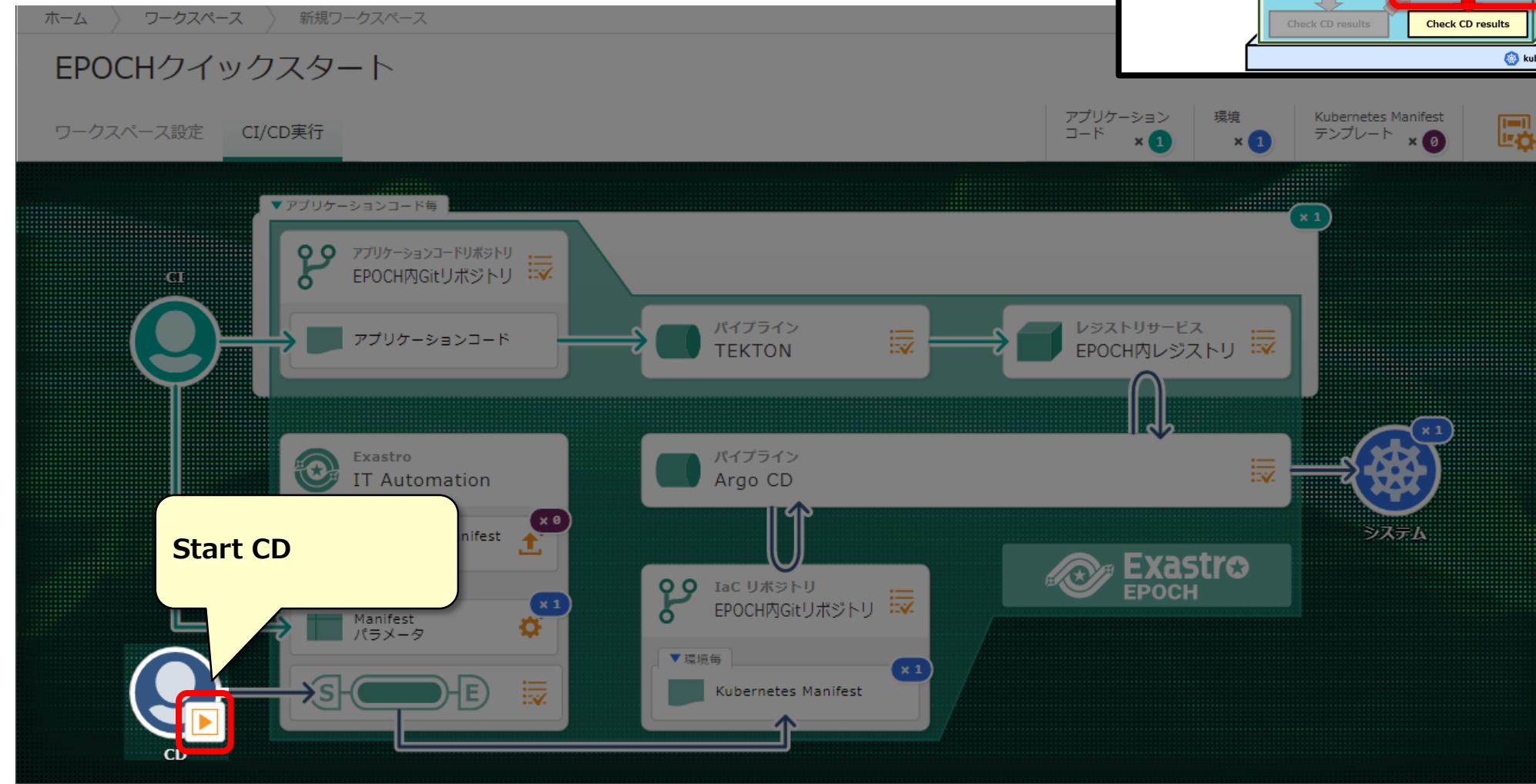
Now that we have deployed to the Staging environment and checked that everything functions as it should, we can now deploy to the Production environment.



# 4.6 CI/CD Workflow No.2 (16/21)

## Deploy to the Production environment

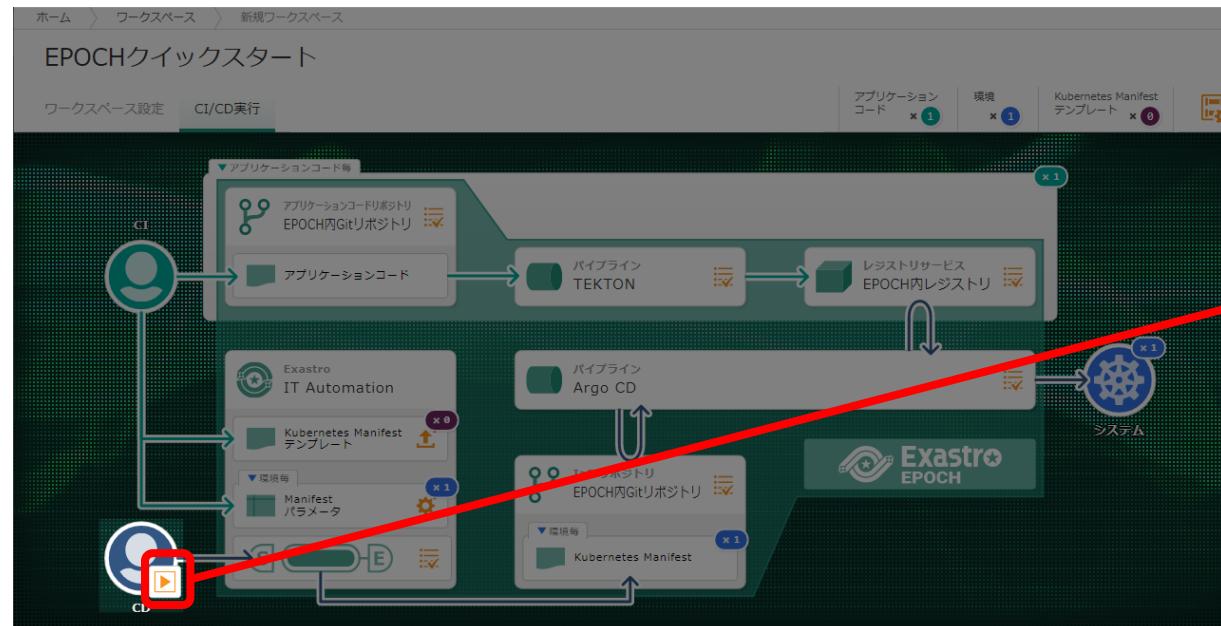
- Start the CD and deploy to the Production environment.



## 4.6 CI/CD Workflow No.2 (17/21)

### Start CD for the Production environment

- Select the environment you want to deploy to.



The dialog box is titled "CD実行指定". It contains the following fields:

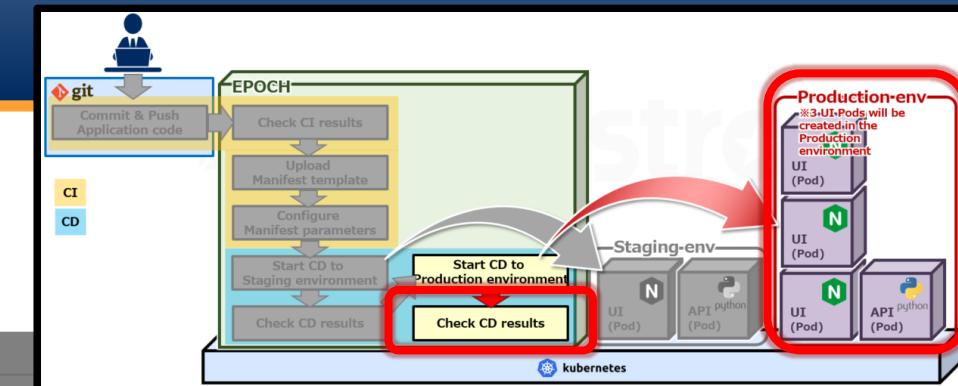
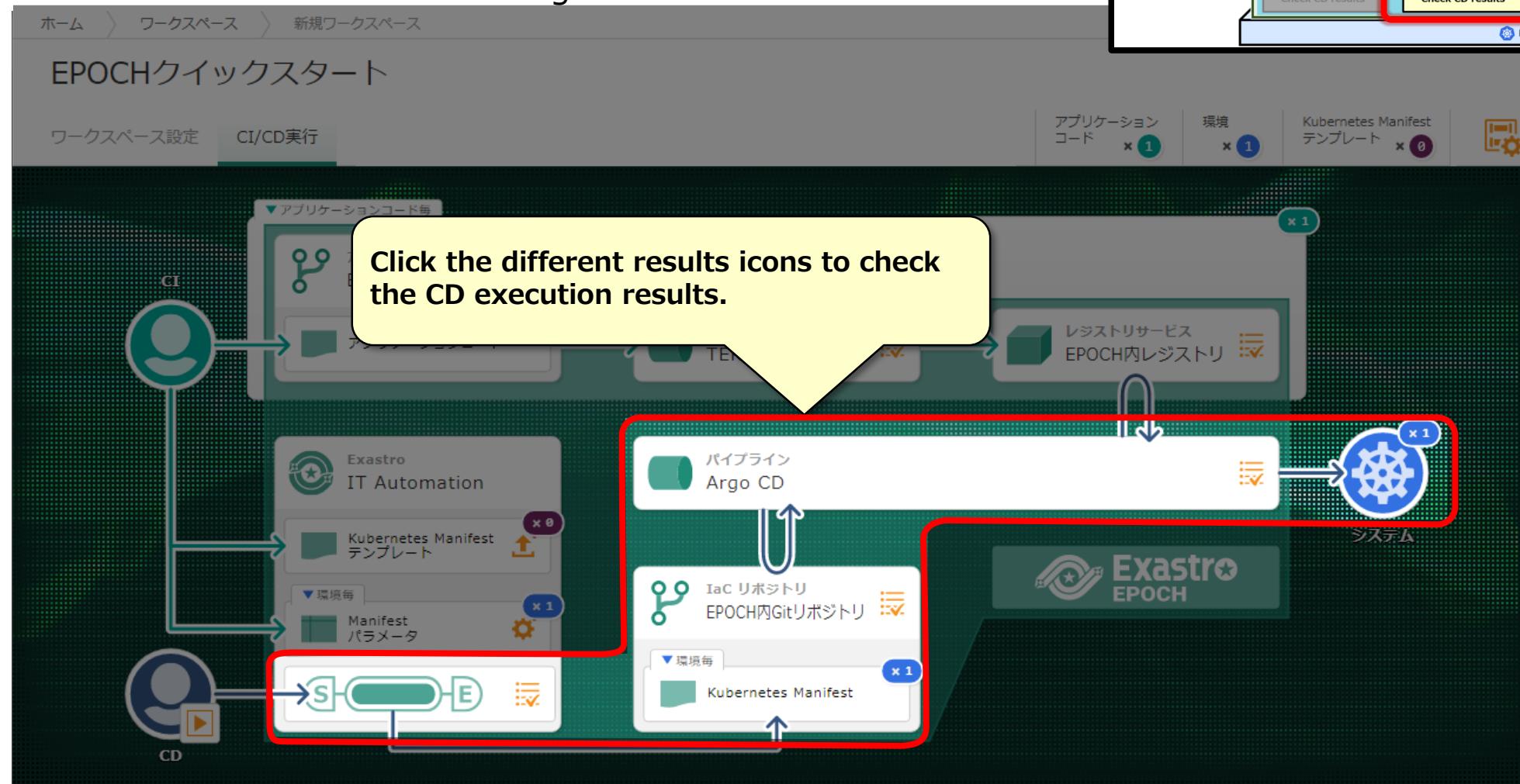
- 実行条件 (Execution Conditions):
  - CD実行日時 (CD Execution Date): 即実行 (Immediate Execution) is selected, with the date 2021/07/08 14:17.
  - 環境 (Environment): A dropdown menu is set to "production".
- Manifest/パラメータ (Manifest/Parameters):
  - 以下的内容でDeployします。よろしいですか? (Will you deploy with the following content?)
  - 環境名 (Environment Name): production
  - Manifestリポジトリ (Manifest Repository): https://github.com/epoch-team/argocd\_manifest-184.git
  - Kubernetes API Server URL (Kubernetes API Server URL): https://kubernetes.default.svc
  - Namespace (Namespace): production-app
- ArgoCDパイプライン (ArgoCD Pipeline):
- Buttons at the bottom:
  - 実行 (Execute) button, which is highlighted with a red border.
  - Press "Enter" callout pointing to the Execute button.

**The CD process for the Production environment has started.  
Let us check that it is working as it should.**

## 4.6 CI/CD Workflow No.2 (18/21)

### Check the CD results (Production)

- We can check the CD execution results from Exastro IT-Automation and ArgoCD.



## 4.6 CI/CD Workflow No.2 (19/21)

Check that the Manifest file has been created (Production environment)

- Open Exastro IT-Automation and check that the Manifest file has been registered to the IaC repository.

The screenshot illustrates the Exastro IT-Automation interface for a CI/CD workflow. On the left, the main dashboard shows various 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. In the center, a 'Login' dialog box is shown with fields for 'LoginID' and 'Password'. A yellow callout box provides the credentials: **LoginID: epoch-user** and **Password: c2jthascR93ijdcyzwJY**. Below the login is the 'Conductor' interface, which includes a 'Filter' button and a 'Details' button on a 'CD' record. A yellow callout [1] points to the 'Conductor' list, [2] points to the 'Filter' button, and [3] points to the 'Details' button. To the right, a 'Checking' screen shows a pipeline with nodes labeled 'DONE' and a yellow callout stating: **If all the nodes says "DONE", then the operation has ended successfully.**

[1] Go to Conductor -> Conductor list

[2] Click "Filter"

[3] Press "Details" on the "CD" record.

LoginID: epoch-user  
Password: c2jthascR93ijdcyzwJY

If all the nodes says "DONE", then the operation has ended successfully.

# 4.6 CI/CD Workflow No.2 (20/21)

Check the results of the ArgoCD Pipeline (Production environment)

- Check that the Manifest shows up on Kubernetes

The screenshot illustrates the Exastro EPOCH CI/CD pipeline. On the left, the 'CI/CD実行' (Execution) tab of the 'EPOCHクイックスタート' (Quick Start) workspace is shown. It displays a flowchart of the pipeline: Application Code → Pipeline TEKTON → Registry Service EPOCH内レジストリ. A red arrow points from this flowchart to the 'Let's get stuff deployed!' screen in the center. The 'Let's get stuff deployed!' screen features a cartoon character and the text 'Let's get stuff deployed!'. Below it is the Argo CD login interface, which is highlighted with a red box. To the right of the login screen is a yellow callout box containing the Argo CD login credentials:

**Username: admin  
Password: iYSCKzx2wvxJnn4dCNwN**

Below the login box is another yellow callout box with the following text:

Your browser might say that the connection is not secure. If that is the case, open up the “more information” tab and press “Access”.

On the far left, the 'Applications' section shows two environments: 'production' and 'staging'. A red box highlights the 'production' environment, and a speech bubble above it says 'Select "production"'. Red arrows point from the 'production' environment to the Argo CD login screen and to the 'Sync OK' status in the 'staging' environment details. The 'staging' environment details are shown in a separate window on the right, which includes a 'Sync OK' status message and a list of deployed components:

- ui-app-com (cm)
- api-app-bluegreen (svc)
- api-app-bluegreen (svc)
- ui-app-bluegreen-active (svc)
- ui-app-bluegreen-preview (svc)
- api-app-78c75b9b78-hlhk4 (pod)

A yellow callout box next to the 'staging' window says:

ArgoCD synchronizes every 3 minutes.

At the bottom, a large red box contains the text:

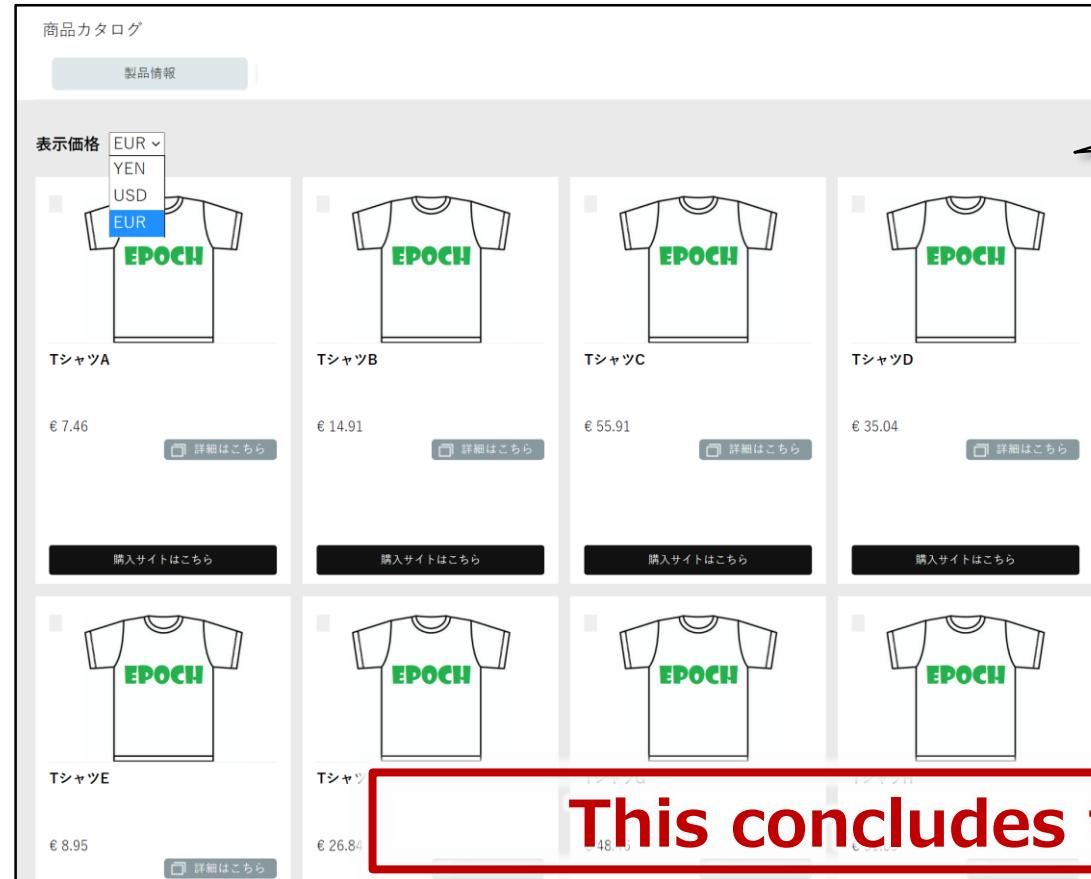
**Let us check the deployed sample application**

## 4.6 CI/CD Workflow No.2 (21/21)

Check the Production environment sample code.

- Access the sample application from the URL below

[http://\[Kubernetes master node IP address/Host name\]:31003/front-end.html](http://[Kubernetes master node IP address/Host name]:31003/front-end.html)



Check that the screen displayed is the same as the one in the Staging environment.

This concludes the tutorial.



## 5. Appendix

The next section contains supplementary information for the tutorial.

# 5.1 Restrictions

The following are the restrictions that are present in the current Exastro EPOCH version and may be subject to change.

- Restrictions (Planned fix in the future)

- Only 1 application code repository is supported at the moment.
- You can only have one collective Git account for the application codes.
- An option for selecting Git service is planned to be added in future releases. Only Git repository URLs are supported in this version.
- Support for build branches is planned to be added in future releases. Only pushed contents is supported in this version.
- Support for Static response analysis is planned to be added in future releases. 現在はSonarQubeを選択した場合に動作しません。
- The only registry service supported is the internal registry service.
- Items other than the image output destination is planned to be added in future releases.
- Support for Authentication tokens and Base64 encoded certificates is planned to be added in future releases.
- The variables that can be specified in the template are fixed. Please see "Column" for more information.

- Points to note

- Installing EPOCH will also install TEKTON.
- The "{{ Variable name }}" variables are specified contents.

# Column : Manifest templates and their variable names

When a Manifest template is uploaded to EPOCH, the system will analyze pre-defined text and make them into modifiable parameters.

```
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
```

Strings that are described in this format :  
**{{ Variable name }}**

will be recognized as user-inputable variables.

The variables that can be used in the current version of EPOCH are the following:

Variable name	Description
{{ image }}	Container image
{{ image_tag }}	Container image tag
{{ param01 }}	Fixed variables that users can use freely.
?	
{{ param20 }}	Fixed variables that users can use freely. (Can use max 20)

※Support for user-specified variable names is planned to be added in future releases.



**Exastro** 