



# OpenShift Container Platform 4.6

## Pipeline

OpenShift Container Platform에서 Pipeline 구성 및 사용



# OpenShift Container Platform 4.6 Pipeline

---

OpenShift Container Platform에서 Pipeline 구성 및 사용

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

## 법적 공지

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Pipelines.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## 초록

이 문서에서는 OpenShift Container Platform에서 Pipeline을 구성 및 사용하는 방법에 대한 지침을 설명합니다.

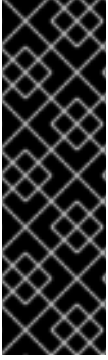
## 차례

<b>1장. OPENSIFT PIPELINES 이해</b> .....	<b>4</b>
1.1. 주요 기능	4
1.2. RED HAT OPENSIFT PIPELINES 개념	4
1.3. 자세한 OPENSIFT PIPELINE 개념	5
1.3.1. Task	5
1.3.2. TaskRun	6
1.3.3. 파이프라인	7
1.3.4. PipelineRun	9
1.3.5. Workspace	10
1.3.6. Trigger	12
1.4. 추가 리소스	15
<b>2장. OPENSIFT PIPELINES 설치</b> .....	<b>16</b>
사전 요구 사항	16
2.1. 웹 콘솔에서 RED HAT OPENSIFT PIPELINES OPERATOR 설치	16
2.2. CLI를 사용하여 OPENSIFT PIPELINES OPERATOR 설치	17
<b>3장. OPENSIFT PIPELINES 설치 제거</b> .....	<b>19</b>
3.1. RED HAT OPENSIFT PIPELINES 구성 요소 및 사용자 정의 리소스 삭제	19
3.2. RED HAT OPENSIFT PIPELINES OPERATOR 설치 제거	19
<b>4장. OPENSIFT PIPELINES를 사용하여 애플리케이션용 CI/CD 솔루션 작성</b> .....	<b>20</b>
4.1. 사전 요구 사항	20
4.2. 프로젝트 생성 및 PIPELINE SERVICEACCOUNT 검사	20
4.3. PIPELINE TASK 생성	21
4.4. PIPELINE 조립	22
4.5. PIPELINE 실행	24
4.6. PIPELINE에 TRIGGER 추가	25
4.7. WEBHOOK 생성	28
4.8. 파이프라인 실행 트리거	29
4.9. 추가 리소스	29
<b>5장. 개발자 관점을 사용하여 RED HAT OPENSIFT PIPELINES 작업</b> .....	<b>30</b>
사전 요구 사항	30
5.1. PIPELINE 빌더를 사용하여 PIPELINE 구성	30
5.2. OPENSIFT PIPELINES를 사용하여 애플리케이션 작성	32
5.3. 개발자 관점을 사용하여 PIPELINE과 상호 작용	32
5.4. PIPELINE 시작	33
5.5. PIPELINE 편집	36
5.6. PIPELINE 삭제	36
<b>6장. RED HAT OPENSIFT PIPELINES 릴리스 정보</b> .....	<b>38</b>
6.1. 지원 요청	38
6.2. RED HAT OPENSIFT PIPELINES TECHNOLOGY PREVIEW 1.2 릴리스 정보	38
6.2.1. 새로운 기능	38
6.2.1.1. 파이프라인	38
6.2.1.2. Pipeline CLI	39
6.2.1.3. Trigger	40
6.2.2. 더 이상 사용되지 않는 기능	40
6.2.3. 확인된 문제	40
6.2.4. 해결된 문제	41
6.3. RED HAT OPENSIFT PIPELINES TECHNOLOGY PREVIEW 1.1 릴리스 정보	42

6.3.1. 새로운 기능	42
6.3.1.1. Pipeline	42
6.3.1.2. Pipeline CLI	43
6.3.1.3. Trigger	44
6.3.2. 사용되지 않는 기능	44
6.3.3. 확인된 문제	45
6.3.4. 해결된 문제	45
6.4. RED HAT OPENSIFT PIPELINES TECHNOLOGY PREVIEW 1.0 릴리스 정보	46
6.4.1. 새로운 기능	46
6.4.1.1. Pipeline	46
6.4.1.2. Pipeline CLI	46
6.4.1.3. Trigger	47
6.4.2. 더 이상 사용되지 않는 기능	47
6.4.3. 확인된 문제	48
6.4.4. 해결된 문제	49



# 1장. OPENSIFT PIPELINES 이해



## 중요

OpenShift Pipelines는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수도 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 <https://access.redhat.com/support/offerings/techpreview/>를 참조하십시오.

Red Hat OpenShift Pipelines는 Kubernetes 리소스 기반의 클라우드 네이티브 CI/CD(연속 통합 및 연속 제공) 솔루션입니다. Tekton 빌딩 블록을 사용하여 기본 구현 세부 사항을 요약함으로써 여러 플랫폼에서 배포를 자동화합니다. Tekton은 Kubernetes 배포 전반에서 이식 가능한 CI/CD Pipeline을 정의하는 데 사용되는 여러 가지 표준 CRD(Custom Resource Definitions)를 도입합니다.

## 1.1. 주요 기능

- Red Hat OpenShift Pipelines는 격리된 컨테이너에서 필요한 모든 종속 항목이 포함된 Pipeline을 실행하는 서버리스 CI/CD 시스템입니다.
- Red Hat OpenShift Pipelines는 마이크로 서비스 기반 아키텍처에서 작업하는 분산된 팀을 위해 설계되었습니다.
- Red Hat OpenShift Pipelines는 쉽게 확장하고 기존 Kubernetes 툴과 통합할 수 있는 표준 CI/CD 파이프라인 정의를 사용하므로 필요에 따라 스케일링할 수 있습니다.
- Red Hat OpenShift Pipelines를 사용하면 모든 Kubernetes 플랫폼에서 이식 가능한 S2I(Source-to-Image), Buildah, Buildpacks, Kaniko 등의 Kubernetes 툴로 이미지를 빌드할 수 있습니다.
- OpenShift Container Platform 개발자 콘솔을 사용하여 Tekton 리소스를 생성하고, Pipeline Run 로그를 검토하고, OpenShift Container Platform 네임스페이스에서 Pipeline을 관리할 수 있습니다.

## 1.2. RED HAT OPENSIFT PIPELINES 개념

Red Hat OpenShift Pipelines는 애플리케이션에 대한 CI/CD 파이프라인을 어셈블할 수 있는 빌드 블록 역할을 하는 표준 CRD(Custom Resource Definitions) 집합을 제공합니다.

### Task

Task는 Pipeline에서 구성 가능한 최소 단위입니다. 근본적으로 Pipeline 빌드를 형성하는 입력 및 출력의 기능입니다. 개별적으로 또는 Pipeline의 일부로 Task를 실행할 수 있습니다. Pipeline에 하나 이상의 Task가 포함되며, 각 Task는 하나 이상의 단계(Step)로 구성됩니다. 단계(Step)는 Task에서 순차적으로 실행되는 일련의 명령 집합입니다.

### Pipeline

Pipeline은 애플리케이션 빌드, 배포 및 제공 작업을 자동화하는 복잡한 워크플로를 구성하기 위해 실행되는 일련의 Task로 구성됩니다. PipelineResources와 매개변수, 그리고 하나 이상의 Task로 구성된 집합체가 Pipeline입니다. Pipeline은 입력 및 출력으로서 Task에 추가되는 PipelineResources를 사용하여 외부 세계와 상호 작용합니다.

### PipelineRun



PipelineRun은 Pipeline의 실행 중 인스턴스입니다. PipelineRun에 의해 Pipeline이 시작되고, Pipeline에서 실행되는 각 Task에 대한 TaskRun 생성이 관리됩니다.

### TaskRun

TaskRun은 Pipeline의 각 Task에 대한 PipelineRun에 의해 자동으로 생성되며, Pipeline에서 Task 인스턴스를 실행한 결과입니다. Pipeline 밖에서 Task를 실행하는 경우, 수동으로 생성해야 할 수도 있습니다.

### Workspace

Workspace는 런타임 시 입력을 수신하거나 출력을 제공하기 위해 Task에 필요한 스토리지 볼륨입니다. Task 또는 Pipeline은 Workspace를 선언하고, TaskRun 또는 PipelineRun은 선언된 Workspace에 마운트되는 스토리지 볼륨의 실제 위치를 제공합니다. 그 결과 유연하고 재사용 가능한 Task가 되며, 여러 Task에서 Workspace를 공유할 수 있게 됩니다.

### Trigger

Trigger는 Git pull 요청과 같은 외부 이벤트를 캡처하고, 이벤트 페이로드를 처리하여 주요 정보를 추출합니다. 추출된 정보는 사전 정의된 매개변수 세트에 맵핑되고, 이 매개변수 세트는 Kubernetes 리소스 생성 및 배포와 관련된 일련의 Task를 트리거합니다. Pipeline과 함께 Trigger를 사용하여 Kubernetes 리소스를 통해 실행이 전체적으로 정의된 완전한 CI/CD 시스템을 생성할 수 있습니다.

### Condition

Condition은 Pipeline에서 Task를 실행하기 전에 수행되는 검증 또는 검사를 나타냅니다. Condition은 반환 값이 **True** 또는 **False**인 논리 테스트를 수행하는 **if** 문과 같습니다. 모든 Condition에서 **True**가 반환되면 Task가 실행되지만 Condition 중 하나라도 실패하면 해당 Task와 모든 후속 Task를 건너 뛩니다. Pipeline에서 Condition을 사용하여 여러 가지 시나리오를 다루는 복잡한 워크플로를 생성할 수 있습니다.

## 1.3. 자세한 OPENSIFT PIPELINE 개념

이 안내서에서는 다양한 Pipeline 개념을 소개합니다.

### 1.3.1. Task

Tasks는 Pipeline 구성 블록이며, 순차적으로 실행되는 단계들로 구성됩니다. Task는 재사용이 가능하며 여러 Pipeline에서 사용할 수 있습니다.

Steps는 이미지 빌드와 같은 특정 목표를 달성하는 일련의 명령입니다. 모든 작업은 Pod로 실행되며 각 단계는 동일한 Pod 내의 자체 컨테이너에서 실행됩니다. 동일한 Pod 내에서 단계가 실행되므로 파일, ConfigMap 및 보안을 캐싱하기 위해 동일한 볼륨에 액세스할 수 있습니다.

다음 예는 **apply-manifests** Task를 보여줍니다.

```
apiVersion: tekton.dev/v1beta1 ①
kind: Task ②
metadata:
  name: apply-manifests ③
spec: ④
  params:
    - default: k8s
      description: The directory in source that contains yaml manifests
      name: manifest_dir
      type: string
  steps:
    - args:
      - |
        echo Applying manifests in $(inputs.params.manifest_dir) directory
```

```

oc apply -f $(inputs.params.manifest_dir)
echo -----
command:
- /bin/bash
- -c
image: quay.io/openshift/origin-cli:latest
name: apply
workingDir: /workspace/source
workspaces:
- name: source
    
```

- 1 Task API 버전은 **v1beta1**입니다.
- 2 Kubernetes 개체의 유형을 지정합니다. 예에서는 **Task**입니다.
- 3 이 Task의 고유한 이름입니다.
- 4 Task의 매개변수 및 단계(Step), Task에서 사용하는 Workspace를 나열합니다.

이 작업은 Pod를 시작하고 **maven:3.6.0-jdk-8-slim** 이미지를 사용하여 해당 Pod 내에서 컨테이너를 실행하여 지정된 명령을 실행합니다. 그리고 애플리케이션의 소스 코드가 포함된 입력 디렉토리인 **workspace-git**을 수신합니다.

이 Task는 Git 리포지토리의 자리 표시자만 선언하고 사용할 Git 리포지토리는 지정하지 않습니다. 따라서 여러 Pipeline과 목적에 Task를 재사용할 수 있습니다.



**주의**

TP(기술 프리뷰)의 Red Hat OpenShift Pipelines 1.3 및 이전 버전을 사용하면 사용자가 SCC( [보안 컨텍스트 제약 조건](#) )를 확인하지 않고도 작업을 생성할 수 있었습니다. 따라서 인증된 사용자는 권한 있는 SCC로 실행되는 컨테이너를 사용하여 작업을 생성할 수 있습니다.

프로덕션 시나리오에서 이러한 보안 문제를 방지하려면 TP에 있는 Pipeline 버전을 사용하지 마십시오. 대신 Pipelines 1.4 이상과 같이 일반적으로 사용 가능한 버전으로 Operator를 업그레이드하는 것이 좋습니다.

### 1.3.2. TaskRun

TaskRun은 클러스터에서 특정 입력, 출력 및 실행 매개변수를 사용하여 실행할 Task를 인스턴스화합니다. 자체로 또는 PipelineRun의 일부로 호출될 수 있습니다.

Task는 컨테이너 이미지를 실행하는 하나 이상의 단계(Step)로 구성되며, 각 컨테이너 이미지의 특정 빌드 작업을 수행합니다. TaskRun은 Task의 모든 단계(Step)를 지정된 순서로 실행하며, 모든 단계(Step)가 성공적으로 실행되거나 실패하는 단계가 발생하면 실행을 멈춥니다.

다음 예는 관련 입력 매개변수를 사용하여 **apply-manifests** Task를 실행하는 TaskRun을 보여줍니다.

```

apiVersion: tekton.dev/v1beta1 1
kind: TaskRun 2
    
```

```

metadata:
  name: apply-manifests-taskrun ③
spec: ④
  serviceAccountName: pipeline
  taskRef: ⑤
    kind: Task
    name: apply-manifests
  workspaces: ⑥
    - name: source
  persistentVolumeClaim:
    claimName: source-pvc

```

- ① TaskRun API 버전은 **v1beta1**입니다.
- ② Kubernetes 개체의 유형을 지정합니다. 예에서는 **TaskRun**입니다.
- ③ 이 TaskRun을 식별하는 고유한 이름입니다.
- ④ TaskRun의 정의입니다. 이 TaskRun에 대한 Task 및 필수 작업 Workspace가 지정됩니다.
- ⑤ 이 TaskRun에 사용되는 Task 참조의 이름입니다. 이 TaskRun은 **apply-manifests** Task를 실행합니다.
- ⑥ TaskRun에서 사용하는 Workspace입니다.

### 1.3.3. 파이프라인

*파이프라인*은 특정 실행 순서대로 정렬된 **Task** 리소스 컬렉션입니다. 하나 이상의 작업이 포함된 파이프라인을 사용하여 애플리케이션에 대한 CI/CD 워크플로를 정의할 수 있습니다.

**Pipeline** 리소스 정의는 함께 사용하면 파이프라인에서 특정 목표를 달성할 수 있는 여러 필드 또는 특성으로 구성됩니다. 각 **Pipeline** 리소스 정의에는 특정 입력을 수집하고 특정 출력을 생성하는 **Task**가 하나 이상 포함되어야 합니다. 또한 파이프라인 정의에는 애플리케이션 요구 사항에 따라 *Conditions*, *Workspaces*, *Parameters* 또는 *Resources*가 선택적으로 포함될 수 있습니다.

다음 예제에서는 **buildah ClusterTask** 리소스를 사용하여 Git 리포지토리에서 애플리케이션 이미지를 빌드하는 **build-and-deploy** 파이프라인을 보여줍니다.

```

apiVersion: tekton.dev/v1beta1 ①
kind: Pipeline ②
metadata:
  name: build-and-deploy ③
spec: ④
  workspaces: ⑤
    - name: shared-workspace
  params: ⑥
    - name: deployment-name
      type: string
      description: name of the deployment to be patched
    - name: git-url
      type: string
      description: url of the git repo for the code of deployment
    - name: git-revision
      type: string

```

```
description: revision to be used from repo of the code for deployment
default: "release-tech-preview-3"
- name: IMAGE
  type: string
  description: image to be built from the code
tasks: 7
- name: fetch-repository
  taskRef:
    name: git-clone
    kind: ClusterTask
  workspaces:
  - name: output
    workspace: shared-workspace
  params:
  - name: url
    value: $(params.git-url)
  - name: subdirectory
    value: ""
  - name: deleteExisting
    value: "true"
  - name: revision
    value: $(params.git-revision)
- name: build-image 8
  taskRef:
    name: buildah
    kind: ClusterTask
  params:
  - name: TLSVERIFY
    value: "false"
  - name: IMAGE
    value: $(params.IMAGE)
  workspaces:
  - name: source
    workspace: shared-workspace
  runAfter:
  - fetch-repository
- name: apply-manifests 9
  taskRef:
    name: apply-manifests
  workspaces:
  - name: source
    workspace: shared-workspace
  runAfter: 10
  - build-image
- name: update-deployment
  taskRef:
    name: update-deployment
  workspaces:
  - name: source
    workspace: shared-workspace
  params:
  - name: deployment
    value: $(params.deployment-name)
  - name: IMAGE
```

```

value: $(params.IMAGE)
runAfter:
- apply-manifests

```

- 1 Pipeline API 버전은 **v1beta1**입니다.
- 2 Kubernetes 개체의 유형을 지정합니다. 예에서는 **Pipeline**입니다.
- 3 이 Pipeline의 고유한 이름입니다.
- 4 Pipeline의 정의와 구조를 지정합니다.
- 5 Pipeline의 모든 Task에 사용되는 Workspace입니다.
- 6 Pipeline의 모든 Task에 사용되는 매개변수입니다.
- 7 Pipeline에서 사용되는 Task 목록을 지정합니다.
- 8 **buildah** ClusterTask를 사용하여 주어진 Git 리포지토리에서 애플리케이션 이미지를 빌드하는 Task **build-image**입니다.
- 9 동일한 이름의 사용자 지정 Task를 사용하는 Task **apply-manifests**입니다.
- 10 Pipeline에서 Task가 실행되는 순서를 지정합니다. 예에서는 **apply-manifests** Task는 **build-image** Task가 완료된 후에만 실행됩니다.

### 1.3.4. PipelineRun

*PipelineRun*은 클러스터에서 특정 입력, 출력 및 실행 매개변수를 사용하여 실행할 Pipeline을 인스턴스화합니다. *PipelineRun*의 각 Task에 해당하는 *TaskRun*이 자동으로 생성됩니다.

Pipeline의 모든 Task는 정의된 순서로 실행되며, 모든 Task가 성공적으로 실행되거나 실패하는 Task가 발생되면 실행을 멈춥니다. **status** 필드는 모니터링 및 감시 목적으로 *PipelineRun*에서 각 *TaskRun*의 진행 상황을 추적하고 저장합니다.

다음 예는 관련 리소스 및 매개변수를 사용하여 **build-and-deploy** Pipeline을 실행하는 *PipelineRun*을 보여줍니다.

```

apiVersion: tekton.dev/v1beta1 1
kind: PipelineRun 2
metadata:
  name: build-deploy-api-pipelinerun 3
spec:
  pipelineRef:
    name: build-and-deploy 4
  params: 5
  - name: deployment-name
    value: vote-api
  - name: git-url
    value: http://github.com/openshift-pipelines/vote-api.git
  - name: IMAGE
    value: image-registry.openshift-image-registry.svc:5000/pipelines-tutorial/vote-api
  workspaces: 6
  - name: shared-workspace

```

```

volumeClaimTemplate:
  spec:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 500Mi

```

- 1 PipelineRun API 버전은 **v1beta1**입니다.
- 2 Kubernetes 개체의 유형을 지정합니다. 예에서는 **PipelineRun**입니다.
- 3 이 PipelineRun을 식별하는 고유한 이름입니다.
- 4 실행할 Pipeline의 이름입니다. 예에서는 **build-and-deploy**입니다.
- 5 Pipeline을 실행하는 데 필요한 매개변수 목록을 지정합니다.
- 6 PipelineRun에서 사용하는 Workspace입니다.

### 1.3.5. Workspace



#### 참고

PipelineResources는 디버그하기 어렵고 범위가 제한되며 Task의 재사용 가능성을 낮추기 때문에 OpenShift Pipelines에서는 PipelineResources 대신 Workspace를 사용할 것을 권장합니다.

Workspace는 런타임 시 Pipeline의 Task에 필요한 공유 스토리지 볼륨을 선언합니다. 볼륨의 실제 위치를 지정하는 대신 Workspace를 사용하여 런타임 시 필요한 파일 시스템 전체 또는 파일 시스템의 일부를 선언할 수 있습니다. TaskRun 또는 PipelineRun에서 해당 Workspace에 마운트된 볼륨의 특정 위치 세부 사항을 제공해야 합니다. 이러한 방식으로 런타임 스토리지 볼륨에서 볼륨 선언을 분리하면 사용자 환경에 종속되지 않으며 유연성 높고 재사용 가능한 Task로 만들 수 있습니다.

다음과 같은 용도로 Workspace를 활용할 수 있습니다.

- Task 입력 및 출력 저장
- Task 간 데이터 공유
- 시크릿에 보관된 자격 증명의 마운트 지점으로 작업 공간 활용
- ConfigMaps에 보관된 구성의 마운트 지점으로 작업 공간 활용
- 조직에서 공유하는 공통 도구의 마운트 지점으로 작업 공간 활용
- 작업 속도를 높이는 빌드 아티팩트 캐시 생성

다음을 사용하여 TaskRun 또는 PipelineRun에서 Workspace를 지정할 수 있습니다.

- 읽기 전용 ConfigMaps 또는 Secret
- 다른 Task와 공유되는 기존 PersistentVolumeClaim
- 제공된 VolumeClaimTemplate의 PersistentVolumeClaim

- TaskRun이 완료되면 삭제되는 emptyDir

다음은 Pipeline에 정의된 대로 **build-image** 및 **apply-manifests** Task에 대한 **shared-workspace** Workspace를 선언하는 **build-and-deploy** Pipeline의 코드 스니펫 예입니다.

```
apiVersion: tekton.dev/v1beta1
kind: Pipeline
metadata:
  name: build-and-deploy
spec:
  workspaces: ❶
  - name: shared-workspace
  params:
  ...
  tasks: ❷
  - name: build-image
    taskRef:
      name: buildah
      kind: ClusterTask
    params:
      - name: TLSVERIFY
        value: "false"
      - name: IMAGE
        value: $(params.IMAGE)
    workspaces: ❸
      - name: source ❹
        workspace: shared-workspace ❺
    runAfter:
      - fetch-repository
  - name: apply-manifests
    taskRef:
      name: apply-manifests
    workspaces: ❻
      - name: source
        workspace: shared-workspace
    runAfter:
      - build-image
  ...
```

- ❶ Pipeline에 정의된 Task 사이에 공유되는 Workspace 목록입니다. Pipeline은 필요한 만큼 Workspace를 정의할 수 있습니다. 예에서는 **shared-workspace**라는 Workspace 한 개만 선언됩니다.
- ❷ Pipeline에서 사용되는 Task의 정의입니다. 이 스니펫은 공통 Workspace를 공유하는 두 개의 Task, **build-image**와 **apply-manifest**를 정의합니다.
- ❸ **build-image** Task에 사용되는 Workspace 목록입니다. Task 정의에 필요한 만큼의 Workspace를 포함할 수 있습니다. 하지만 Task에 사용되는 쓰기 가능한 Workspace를 한 개로 제한하는 것이 좋습니다.
- ❹ Task에서 사용되는 Workspace를 고유하게 식별하는 이름입니다. 이 Task는 **source**라는 Workspace 한 개를 사용합니다.
- ❺ Task에서 사용하는 Pipeline Workspace의 이름입니다. 이어서 **source** Workspace는 **shared-workspace**라는 Pipeline Workspace를 사용한다는 점에 주목하십시오.

- 6 **apply-manifests** Task에 사용되는 Workspace 목록입니다. 이 Task는 **build-image** Task와 **source** Workspace를 공유한다는 점에 주목하십시오.

작업 공간을 사용하면 여러 작업에서 데이터를 공유하고 파이프라인의 각 작업을 실행하는 동안 필요한 하나 이상의 볼륨을 지정할 수 있습니다. 영구 볼륨 클레임을 생성하거나 사용자를 대신하여 영구 볼륨 클레임을 생성하는 볼륨 클레임 템플릿을 제공할 수 있습니다.

다음의 **build-deploy-api-pipelinerun** PipelineRun 코드 조각에서는 볼륨 클레임 템플릿을 사용하여 **build-and-deploy** 파이프라인에 사용된 **shared-workspace** 작업 공간의 스토리지 볼륨을 정의하는 영구 볼륨 클레임을 생성합니다.

```
apiVersion: tekton.dev/v1beta1
kind: PipelineRun
metadata:
  name: build-deploy-api-pipelinerun
spec:
  pipelineRef:
    name: build-and-deploy
  params:
  ...

  workspaces: 1
  - name: shared-workspace 2
    volumeClaimTemplate: 3
      spec:
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 500Mi
```

- 1 PipelineRun에서 볼륨 바인딩이 제공될 Pipeline Workspace 목록을 지정합니다.
- 2 볼륨이 제공될 Pipeline의 Workspace 이름입니다.
- 3 작업 공간의 스토리지 볼륨을 정의하기 위해 영구 볼륨 클레임을 생성하는 볼륨 클레임 템플릿을 지정합니다.

### 1.3.6. Trigger

Kubernetes 리소스에서 전체 CI/CD 실행을 정의하는 완전한 CI/CD 시스템을 생성하려면 파이프라인과 함께 *트리거*를 사용합니다. 트리거는 외부 이벤트를 캡처하고 처리하여 주요 정보를 추출합니다. 이 이벤트 데이터를 미리 정의된 매개변수 집합에 매핑하면 Kubernetes 리소스를 생성 및 배포하고 파이프라인을 인스턴스화할 수 있는 일련의 작업이 트리거됩니다.

애플리케이션에 Red Hat OpenShift Pipeline을 사용하여 CI/CD 워크플로를 정의하는 경우를 예로 들 수 있습니다. 새로운 변경 사항을 애플리케이션 리포지토리에 적용하려면 파이프라인을 시작해야 합니다. 트리거는 모든 변경 이벤트를 캡처하여 처리하고 최신 변경 사항이 적용된 새 이미지를 배포하는 파이프라인 실행을 트리거하는 방식으로 이 프로세스를 자동화합니다.

트리거는 함께 작동하여 재사용 가능하고 분리되고 자체 유지되는 CI/CD 시스템을 형성하는 다음과 같은 주요 리소스로 구성됩니다.



- **TriggerBinding** 리소스는 이벤트를 검증하고 이벤트 페이로드에서 필드를 추출한 다음 해당 필드를 매개변수로 저장합니다.
- **TriggerTemplate** 리소스는 리소스를 생성해야 하는 방법에 대해 표준 역할을 합니다. **TriggerBinding** 리소스에서 매개변수화된 데이터를 사용하는 방식을 지정합니다. 트리거 템플릿은 트리거 바인딩을 통해 입력을 수신한 다음 새 파이프라인 리소스를 생성하고 새 파이프라인 실행을 시작하는 일련의 작업을 수행합니다.
- **EventListener** 리소스는 JSON 페이로드와 함께 들어오는 HTTP 기반 이벤트를 수신 대기하는 끝점 또는 이벤트 싱크를 제공합니다. 각 **TriggerBinding** 리소스에서 이벤트 매개변수를 추출한 다음 이 데이터를 처리하여 해당 **TriggerTemplate** 리소스에서 지정하는 Kubernetes 리소스를 생성합니다. 또한 **EventListener** 리소스는 페이로드 유형을 확인하고 선택적으로 수정하는 이벤트 **interceptors**를 사용하여 페이로드에 대한 간단한 이벤트 처리 또는 기본 필터링 작업을 수행합니다. 현재 파이프라인 트리거는 다음 네 가지 유형의 인터셉터를 지원합니다. *Webhook 인터셉터*, *GitHub 인터셉터*, *GitLab 인터셉터* 및 *CEL (Common Expression Language) 인터셉터*.
- **Trigger** 리소스는 **TriggerBinding** 및 **TriggerTemplate** 리소스를 연결하고, **EventListener** 사양에서 이 **Trigger** 리소스를 참조합니다.

다음은 수신된 이벤트 페이로드에서 Git 리포지토리 정보를 추출하는 **TriggerBinding** 리소스의 코드 조각 예입니다.

```
apiVersion: triggers.tekton.dev/v1alpha1 ❶
kind: TriggerBinding ❷
metadata:
  name: vote-app ❸
spec:
  params: ❹
  - name: git-repo-url
    value: $(body.repository.url)
  - name: git-repo-name
    value: $(body.repository.name)
  - name: git-revision
    value: $(body.head_commit.id)
```

❶ **TriggerBinding** 리소스의 API 버전입니다. 이 예제에서는 **v1alpha1**입니다.

❷ Kubernetes 개체의 유형을 지정합니다. 예에서는 **TriggerBinding**입니다.

❸ **TriggerBinding** 리소스를 확인하는 고유한 이름입니다.

❹ 수신된 이벤트 페이로드에서 추출하여 **TriggerTemplate** 리소스로 전달할 매개변수 목록입니다. 예에서는 Git 리포지토리 URL, 이름 및 개정 정보가 이벤트 페이로드의 본문에서 추출됩니다.

다음은 방금 생성한 **TriggerBinding** 리소스에서 수신된 Git 리포지토리 정보를 사용하여 애플리케이션을 생성하는 **TriggerTemplate** 리소스의 코드 조각입니다.

```
apiVersion: triggers.tekton.dev/v1alpha1 ❶
kind: TriggerTemplate ❷
metadata:
  name: vote-app ❸
spec:
  params: ❹
  - name: git-repo-url
```

```

description: The git repository url
- name: git-revision
description: The git revision
default: master
- name: git-repo-name
description: The name of the deployment to be created / patched

resourcetemplates: 5
- apiVersion: tekton.dev/v1beta1
  kind: PipelineRun
  metadata:
    name: build-deploy-$(tt.params.git-repo-name)-$(uid)
  spec:
    serviceAccountName: pipeline
    pipelineRef:
      name: build-and-deploy
    params:
      - name: deployment-name
        value: $(tt.params.git-repo-name)
      - name: git-url
        value: $(tt.params.git-repo-url)
      - name: git-revision
        value: $(tt.params.git-revision)
      - name: IMAGE
        value: image-registry.openshift-image-registry.svc:5000/pipelines-tutorial/$(tt.params.git-repo-
name)
    workspaces:
      - name: shared-workspace
    volumeClaimTemplate:
      spec:
        accessModes:
          - ReadWriteOnce
      resources:
        requests:
          storage: 500Mi

```

- 1 **TriggerTemplate** 리소스의 API 버전입니다. 이 예제에서는 **v1alpha1**입니다.
- 2 Kubernetes 개체의 유형을 지정 합니다. 예에서는 **TriggerTemplate**입니다.
- 3 **TriggerTemplate** 리소스를 식별하는 고유 이름입니다.
- 4 **TriggerBinding** 또는 **EventListener** 리소스에서 제공하는 매개변수입니다.
- 5 **TriggerBinding** 또는 **EventListener** 리소스를 통해 수신한 매개변수를 사용하여 리소스를 생성해 야 하는 방법을 지정하는 템플릿 목록입니다.

다음 예제는 **TriggerBinding** 및 **TriggerTemplate** 리소스를 연결하는 **vote-trigger**라는 **Trigger** 리소스의 코드 조각을 보여줍니다.

```

apiVersion: triggers.tekton.dev/v1alpha1 1
kind: Trigger 2
metadata:
  name: vote-trigger 3
spec:

```

```

serviceAccountName: pipeline ④
bindings:
  - ref: vote-app ⑤
template: ⑥
  name: vote-app

```

- ① **Trigger** 리소스의 API 버전입니다. 이 예제에서는 **v1alpha1**입니다.
- ② Kubernetes 개체의 유형을 지정합니다. 이 예제에서는 **Trigger**입니다.
- ③ **Trigger** 리소스를 식별하는 고유 이름입니다.
- ④ 사용할 서비스 계정 이름입니다.
- ⑤ **TriggerTemplate** 리소스에 연결할 **TriggerBinding** 리소스의 이름입니다.
- ⑥ **TriggerBinding** 리소스에 연결할 **TriggerTemplate** 리소스의 이름입니다.

다음 예제에서는 **vote-trigger**라는 **Trigger** 리소스를 참조하는 **EventListener** 리소스를 보여줍니다.

```

apiVersion: triggers.tekton.dev/v1alpha1 ①
kind: EventListener ②
metadata:
  name: vote-app ③
spec:
  serviceAccountName: pipeline ④
  triggers:
    - triggerRef: vote-trigger ⑤

```

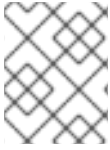
- ① **EventListener** 리소스의 API 버전입니다. 이 예제에서는 **v1alpha1**입니다.
- ② Kubernetes 개체의 유형을 지정합니다. 예에서는 **EventListener**입니다.
- ③ **EventListener** 리소스를 식별하는 고유 이름입니다.
- ④ 사용할 서비스 계정 이름입니다.
- ⑤ **EventListener** 리소스에서 참조하는 **Trigger** 리소스의 이름입니다.

## 1.4. 추가 리소스

- 파이프라인 설치에 대한 내용은 [OpenShift Pipelines 설치](#)를 참조하십시오.
- 사용자 정의 CI/CD 솔루션 생성에 대한 자세한 내용은 [CI/CD 파이프라인을 사용하여 애플리케이션 생성](#)을 참조하십시오.

## 2장. OPENSIFT PIPELINES 설치

이 가이드에서는 클러스터 관리자에게 Red Hat OpenShift Pipelines Operator를 OpenShift Container Platform 클러스터에 설치하는 프로세스를 안내합니다.



### 참고

Red Hat OpenShift Pipelines Operator는 제한된 네트워크 환경에서 설치에 지원됩니다. 자세한 내용은 [제한된 네트워크에서 Operator Lifecycle Manager 사용](#) 을 참조하십시오.

### 사전 요구 사항

- **cluster-admin** 권한이 있는 계정을 사용하여 OpenShift Container Platform 클러스터에 액세스할 수 있습니다.
- **oc** CLI를 설치했습니다.
- 로컬 시스템에 [OpenShift Pipelines \(tkn\) CLI](#)를 설치했습니다.

## 2.1. 웹 콘솔에서 RED HAT OPENSIFT PIPELINES OPERATOR 설치

OpenShift Container Platform OperatorHub에 나열된 Operator를 사용하여 Red Hat OpenShift Pipelines를 설치할 수 있습니다. Red Hat OpenShift Pipelines Operator를 설치하면 Pipeline 구성에 필요한 CR(사용자 정의 리소스)이 Operator와 함께 자동으로 설치됩니다.

### 절차

1. 웹 콘솔의 관리자 화면에서 **Operator** → **OperatorHub**로 이동합니다.
2. 키워드로 필터링 상자를 사용하여 카탈로그에서 **Red Hat OpenShift Pipelines Operator** 를 검색합니다. **OpenShift Pipelines Operator** 타일을 클릭합니다.



### 참고

**OpenShift Pipelines Operator**의 커뮤니티 버전이 선택되지 않았는지 확인하십시오.

3. **Red Hat OpenShift Pipelines Operator** 페이지에서 **Operator**에 대한 간략한 설명을 읽어 보십시오. 설치를 클릭합니다.
4. **Operator** 설치 페이지에서 다음을 수행합니다.
  - a. **Installation Mode**로 **All namespaces on the cluste(default)**를 선택합니다. 이 모드에서는 기본 **openshift-operators** 네임스페이스에 Operator가 설치되므로 Operator가 클러스터의 모든 네임스페이스를 감시하고 사용 가능하게 만들 수 있습니다.
  - b. **Approval Strategy**으로 **Automatic**을 선택합니다. 그러면 Operator에 향후 지원되는 업그레이드가 OLM(Operator Lifecycle Manager)에 의해 자동으로 처리됩니다. **Manual** 승인 전략을 선택하면 OLM에서 업데이트 요청을 생성합니다. 클러스터 관리자는 Operator를 새 버전으로 업데이트하려면 OLM 업데이트 요청을 수동으로 승인해야 합니다.
  - c. **Update Channel**을 선택합니다.

- **ocp-<4.x>** 채널을 사용하면 Red Hat OpenShift Pipelines Operator의 안정적인 최신 릴리스를 설치할 수 있습니다.
- **preview** 채널을 사용하면 Red Hat OpenShift Pipelines Operator의 최신 프리뷰 버전을 설치할 수 있습니다. 이 버전에는 4.x 업데이트 채널에서 아직 사용할 수 없는 기능이 포함될 수 있습니다.

5. 설치를 클릭합니다. **Installed Operators** 페이지의 목록에 해당 Operator가 나타납니다.



#### 참고

Operator는 **openshift-operators** 네임스페이스에 자동으로 설치됩니다.

6. Red Hat OpenShift Pipelines Operator가 성공적으로 설치되었는지 확인하려면 **상태가 최신 업데이트 완료**로 설정되어 있는지 확인합니다.

## 2.2. CLI를 사용하여 OPENSIFT PIPELINES OPERATOR 설치

CLI를 사용하여 OperatorHub에서 Red Hat OpenShift Pipelines Operator를 설치할 수 있습니다.

### 프로세스

1. 서브스크립션 오브젝트 YAML 파일을 생성하여 Red Hat OpenShift Pipelines Operator에 네임스페이스를 서브스크립션합니다(예: **sub.yaml**).

#### Subscription의 예

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-pipelines-operator
  namespace: openshift-operators
spec:
  channel: <channel name> 1
  name: openshift-pipelines-operator-rh 2
  source: redhat-operators 3
  sourceNamespace: openshift-marketplace 4
```

- 1 Operator를 서브스크립션할 채널 이름을 지정합니다.
- 2 서브스크립션할 Operator의 이름입니다.
- 3 Operator를 제공하는 CatalogSource의 이름입니다.
- 4 CatalogSource의 네임스페이스입니다. 기본 OperatorHub CatalogSources에는 **openshift-marketplace**를 사용합니다.

2. 서브스크립션 오브젝트를 생성합니다.

```
$ oc apply -f sub.yaml
```

이제 Red Hat OpenShift Pipelines Operator가 기본 타겟 네임스페이스인 **openshift-operators**에 설치되었습니다.

## 추가 리소스

- [adding Operators to a cluster](#) 섹션에서 OpenShift Container Platform에 Operator를 설치하는 방법을 확인할 수 있습니다.

## 3장. OPENSIFT PIPELINES 설치 제거

Red Hat OpenShift Pipelines Operator 설치 제거는 2단계로 구성된 프로세스입니다.

1. Red Hat OpenShift Pipelines Operator를 설치할 때 기본적으로 추가된 CR(Custom Resource)을 삭제합니다.
2. Red Hat OpenShift Pipelines Operator를 설치 제거합니다.

Operator를 설치 제거하는 것만으로 설치 과정에서 기본적으로 생성된 Red Hat OpenShift Pipelines 구성 요소가 제거되지는 않습니다.

### 3.1. RED HAT OPENSIFT PIPELINES 구성 요소 및 사용자 정의 리소스 삭제

Red Hat OpenShift Pipelines Operator 설치 과정에서 기본적으로 생성된 CR(사용자 정의 리소스)을 삭제합니다.

#### 프로세스

1. 웹 콘솔의 관리자 화면에서 **Administration** → **Custom Resource Definition**로 이동합니다.
2. 이름으로 필터링 박스에 **config.operator.tekton.dev**를 입력하여 Red Hat OpenShift Pipelines Operator CR을 검색합니다.
3. **CRD Config**를 클릭하여 **Custom Resource Definition Details** 페이지를 엽니다.
4. **Actions** 드롭다운 메뉴를 클릭하고 **Delete Custom Resource Definition**를 선택합니다.



#### 참고

CR을 삭제하면 Red Hat OpenShift Pipelines 구성 요소가 삭제되고 클러스터의 모든 작업과 파이프라인이 사라집니다.

5. **Delete**를 클릭하여 CR 삭제를 확인합니다.

### 3.2. RED HAT OPENSIFT PIPELINES OPERATOR 설치 제거

#### 프로세스

1. **Operators** → **OperatorHub** 페이지에서 키워드로 필터링 박스를 사용하여 **Red Hat OpenShift Pipelines Operator**를 검색합니다.
2. **OpenShift Pipelines Operator** 타일을 클릭합니다. Operator 타일은 Operator가 설치되었음을 나타냅니다.
3. **OpenShift Pipelines Operator** 설명자 페이지에서 **Uninstall**를 클릭합니다.

#### 추가 리소스

- OpenShift Container Platform에서 Operator를 설치 제거하는 방법에 대한 자세한 내용은 [클러스터에서 Operator 삭제](#) 섹션에서 확인할 수 있습니다.

## 4장. OPENSIFT PIPELINES를 사용하여 애플리케이션용 CI/CD 솔루션 작성

Red Hat OpenShift Pipelines를 사용하면 애플리케이션을 빌드, 테스트, 배포하는 사용자 정의 CI/CD 솔루션을 생성할 수 있습니다.

애플리케이션에 사용할 완전한 셀프 서비스 CI/CD 파이프라인을 생성하려면 다음 작업을 수행해야 합니다.

- 사용자 정의 작업을 생성하거나 재사용 가능한 기존 작업을 설치합니다.
- 애플리케이션용 제공 파이프라인을 생성하고 정의합니다.
- 다음 접근 방법 중 하나를 사용하여 파이프라인 실행을 위해 작업 공간에 연결된 스토리지 볼륨 또는 파일 시스템을 제공합니다.
  - 영구 볼륨 클레임을 생성하는 볼륨 클레임 템플릿 지정
  - 영구 볼륨 클레임 지정
- 파이프라인을 인스턴스화하고 호출할 **PipelineRun** 오브젝트를 생성합니다.
- 소스 리포지토리의 이벤트를 캡처하는 트리거를 추가합니다.

여기서는 **pipelines-tutorial** 예제를 사용하여 선행 Task들을 보여줍니다. 예에서는 다음으로 구성된 간단한 애플리케이션을 사용합니다.

- **ui-repo** Git 리포지토리에 소스 코드가 있는 프론트 엔드 인터페이스 **vote-ui**
- **api-repo** Git 리포지토리에 소스 코드가 있는 백엔드 인터페이스 **vote-api**
- **pipelines-tutorial** Git 리포지토리의 **apply-manifests** 및 **update-deployment** 작업

### 4.1. 사전 요구 사항

- OpenShift Container Platform 클러스터에 액세스 권한을 보유하고 있습니다.
- OpenShift OperatorHub에 나열된 Red Hat OpenShift Pipelines Operator를 사용하여 [OpenShift Pipelines](#)를 설치했습니다. 설치를 마쳤으면 전체 클러스터에 적용할 수 있습니다.
- [OpenShift Pipelines CLI](#)를 설치했습니다.
- GitHub ID를 사용하여 프론트 엔드 **ui-repo** 및 백엔드 **api-repo** Git 리포지토리를 분기했으며, 이러한 리포지토리에 관리자 액세스 권한이 있습니다.
- 선택 사항: **pipelines-tutorial** Git 리포지토리를 복제했습니다.

### 4.2. 프로젝트 생성 및 PIPELINE SERVICEACCOUNT 검사

#### 프로세스

1. OpenShift Container Platform 클러스터에 로그인합니다.

```
$ oc login -u <login> -p <password> https://openshift.example.com:6443
```



2. 샘플 애플리케이션용 프로젝트를 생성합니다. 예시 워크플로에서는 **pipelines-tutorial** 프로젝트를 생성합니다.

```
$ oc new-project pipelines-tutorial
```



#### 참고

다른 이름으로 프로젝트를 생성하는 경우, 예시에 사용된 리소스 URL을 사용자의 프로젝트 이름으로 업데이트하십시오.

3. **Pipeline** ServiceAccount를 표시합니다.  
Red Hat OpenShift Pipelines Operator는 이미지를 빌드하고 내보내기에 충분한 권한이 있는 **pipeline**이라는 ServiceAccount를 추가하고 구성합니다. 이 ServiceAccount는 PipelineRun에서 사용됩니다.

```
$ oc get serviceaccount pipeline
```

## 4.3. PIPELINE TASK 생성

### 절차

1. 파이프라인의 재사용 가능한 작업 목록이 포함된 **pipelines-tutorial** 리포지토리에서 **apply-manifests** 및 **update-deployment Task** 리소스를 설치합니다.

```
$ oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/release-tech-preview-3/01_pipeline/01_apply_manifest_task.yaml
$ oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/release-tech-preview-3/01_pipeline/02_update_deployment_task.yaml
```

2. **tkn task list** 명령을 사용하여 생성한 작업 목록을 표시합니다.

```
$ tkn task list
```

**apply-manifest** 및 **update-deployment Task** 리소스가 생성되었는지 출력에서 확인합니다.

NAME	DESCRIPTION	AGE
apply-manifests		1 minute ago
update-deployment		48 seconds ago

3. **tkn clustertasks list** 명령을 사용하여 Operator에서 설치한 추가 **ClusterTask** 리소스를 나열합니다(예:**buildah** 및 **s2i-python-3**).



#### 참고

**buildah ClusterTask** 리소스를 실행하려면 권한 있는 보안 컨텍스트가 필요하므로 권한이 있는 Pod 컨테이너를 사용해야 합니다. Pod의 보안 컨텍스트 제약 조건(SCC)에 대한 자세한 내용은 추가 리소스 섹션을 참조하십시오.

```
$ tkn clustertasks list
```

Operator에서 설치한 **ClusterTask** 리소스가 출력에 나열됩니다.

NAME	DESCRIPTION	AGE
buildah		1 day ago
git-clone		1 day ago
s2i-php		1 day ago
tkn		1 day ago

## 4.4. PIPELINE 조립

Pipeline은 CI/CD 흐름을 나타내며 실행할 Task들로 정의됩니다. 여러 애플리케이션 및 환경에서 포괄적으로 적용되고 재사용 가능하도록 설계되었습니다.

Pipeline은 **from** 및 **runAfter** 매개변수를 사용하여 Task들이 상호 작용하는 방법과 실행 순서를 지정합니다. 그리고 **workspaces** 필드를 사용하여 Pipeline의 각 Task 실행 중 필요한 하나 이상의 볼륨을 지정합니다.

이 섹션에서는 GitHub에서 애플리케이션의 소스 코드를 가져와 OpenShift Container Platform에서 빌드 및 배포하는 Pipeline을 생성합니다.

Pipeline은 백엔드 애플리케이션 **vote-api** 및 프론트 엔드 애플리케이션 **vote-ui**에 대해 다음 작업을 수행합니다.

- **git-url** 및 **git-revision** 매개변수를 참조하여 Git 리포지토리에서 애플리케이션의 소스 코드를 복제합니다.
- **buildah** ClusterTask를 사용하여 컨테이너 이미지를 빌드합니다.
- **image** 매개변수를 참조하여 내부 이미지 레지스트리로 이미지를 푸시합니다.
- **apply-manifests** 및 **update-deployment** Task를 사용하여 OpenShift Container Platform에 새 이미지를 배포합니다.

### 프로세스

1. 다음 샘플 Pipeline YAML 파일의 내용을 복사하여 저장합니다.

```

apiVersion: tekton.dev/v1beta1
kind: Pipeline
metadata:
  name: build-and-deploy
spec:
  workspaces:
    - name: shared-workspace
  params:
    - name: deployment-name
      type: string
      description: name of the deployment to be patched
    - name: git-url
      type: string
      description: url of the git repo for the code of deployment
    - name: git-revision
      type: string
      description: revision to be used from repo of the code for deployment
      default: "release-tech-preview-3"
    - name: IMAGE
      type: string

```

```

description: image to be built from the code
tasks:
- name: fetch-repository
  taskRef:
    name: git-clone
    kind: ClusterTask
  workspaces:
    - name: output
      workspace: shared-workspace
  params:
    - name: url
      value: $(params.git-url)
    - name: subdirectory
      value: ""
    - name: deleteExisting
      value: "true"
    - name: revision
      value: $(params.git-revision)
- name: build-image
  taskRef:
    name: buildah
    kind: ClusterTask
  params:
    - name: TLSVERIFY
      value: "false"
    - name: IMAGE
      value: $(params.IMAGE)
  workspaces:
    - name: source
      workspace: shared-workspace
  runAfter:
    - fetch-repository
- name: apply-manifests
  taskRef:
    name: apply-manifests
  workspaces:
    - name: source
      workspace: shared-workspace
  runAfter:
    - build-image
- name: update-deployment
  taskRef:
    name: update-deployment
  workspaces:
    - name: source
      workspace: shared-workspace
  params:
    - name: deployment
      value: $(params.deployment-name)
    - name: IMAGE
      value: $(params.IMAGE)
  runAfter:
    - apply-manifests

```

Pipeline 정의는 Git 소스 리포지토리 및 이미지 레지스트리의 세부 사항을 요약합니다. 이러한 세부 사항은 Pipeline이 트리거되고 실행될 때 **params**로 추가됩니다.

2. Pipeline을 생성합니다.

```
$ oc create -f <pipeline-yaml-file-name.yaml>
```

또는 Git 리포지토리에서 직접 YAML 파일을 실행할 수도 있습니다.

```
$ oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/release-tech-preview-3/01_pipeline/04_pipeline.yaml
```

3. `tkn pipeline list` 명령을 사용하여 Pipeline이 애플리케이션에 추가되었는지 확인합니다.

```
$ tkn pipeline list
```

**build-and-deploy** Pipeline이 생성된 것이 출력에서 확인됩니다.

```
NAME          AGE          LAST RUN   STARTED  DURATION  STATUS
build-and-deploy  1 minute ago  ---      ---      ---      ---
```

## 4.5. PIPELINE 실행

**PipelineRun** 리소스는 파이프라인을 시작하고 특정 호출에 사용해야 하는 Git 및 이미지 리소스에 연결합니다. 그리고 파이프라인의 각 작업에 대해 **TaskRun** 리소스를 자동으로 생성하고 시작합니다.

### 프로세스

1. 백엔드 애플리케이션의 파이프라인을 시작합니다.

```
$ tkn pipeline start build-and-deploy \
-w name=shared-
workspace,volumeClaimTemplateFile=https://raw.githubusercontent.com/openshift/pipelines-tutorial/release-tech-preview-3/01_pipeline/03_persistent_volume_claim.yaml \
-p deployment-name=vote-api \
-p git-url=http://github.com/openshift-pipelines/vote-api.git \
-p IMAGE=image-registry.openshift-image-registry.svc:5000/pipelines-tutorial/vote-api \
```

위 명령은 파이프라인 실행을 위한 영구 볼륨 클레임을 생성하는 볼륨 클레임 템플릿을 사용합니다.

2. 파이프라인 실행의 진행 상황을 추적하려면 다음 명령을 입력합니다.

```
$ tkn pipelinerun logs <pipelinerun_id> -f
```

위 명령의 `<pipelinerun_id>`는 이전 명령의 출력에서 반환된 **PipelineRun**의 ID입니다.

3. 프론트 엔드 애플리케이션의 Pipeline을 시작합니다.

```
$ tkn pipeline start build-and-deploy \
-w name=shared-
workspace,volumeClaimTemplateFile=https://raw.githubusercontent.com/openshift/pipelines-tutorial/release-tech-preview-3/01_pipeline/03_persistent_volume_claim.yaml \
-p deployment-name=vote-ui \
-p git-url=http://github.com/openshift-pipelines/vote-ui.git \
-p IMAGE=image-registry.openshift-image-registry.svc:5000/pipelines-tutorial/vote-ui \
```

- 파이프라인 실행의 진행 상황을 추적하려면 다음 명령을 입력합니다.

```
$ tkn pipelinerun logs <pipelinerun_id> -f
```

위 명령의 <pipelinerun\_id>는 이전 명령의 출력에서 반환된 **PipelineRun**의 ID입니다.

- 몇 분 후에 **tkn pipelinerun list** 명령을 사용하여 모든 PipelineRun을 나열하여 Pipeline이 성공적으로 실행되었는지 확인합니다.

```
$ tkn pipelinerun list
```

PipelineRuns 목록이 출력됩니다.

```
NAME                STARTED    DURATION    STATUS
build-and-deploy-run-xy7rw  1 hour ago  2 minutes   Succeeded
build-and-deploy-run-z2rz8  1 hour ago  19 minutes  Succeeded
```

- 애플리케이션 경로를 가져옵니다.

```
$ oc get route vote-ui --template='http://{{.spec.host}}'
```

이전 명령의 출력에 주목하십시오. 이 경로를 사용하여 애플리케이션에 액세스할 수 있습니다.

- 이전 파이프라인의 파이프라인 리소스 및 서비스 계정을 사용하여 마지막 파이프라인 실행을 다시 실행하려면 다음을 실행합니다.

```
$ tkn pipeline start build-and-deploy --last
```

## 4.6. PIPELINE에 TRIGGER 추가

트리거를 사용하면 파이프라인에서 내보내기 이벤트 및 가져오기 요청 등의 외부 GitHub 이벤트에 응답할 수 있습니다. 애플리케이션에 대한 파이프라인을 어셈블하고 시작한 후 **TriggerBinding**, **TriggerTemplate**, **Trigger**, **EventListener** 리소스를 추가하여 GitHub 이벤트를 캡처합니다.

### 프로세스

- 다음 샘플 **TriggerBinding** YAML 파일의 내용을 복사하여 저장합니다.

```
apiVersion: triggers.tekton.dev/v1alpha1
kind: TriggerBinding
metadata:
  name: vote-app
spec:
  params:
    - name: git-repo-url
      value: $(body.repository.url)
    - name: git-repo-name
      value: $(body.repository.name)
    - name: git-revision
      value: $(body.head_commit.id)
```

- TriggerBinding** 리소스를 생성합니다.

```
$ oc create -f <triggerbinding-yaml-file-name.yaml>
```

또는 **pipelines-tutorial** Git 리포지토리에서 직접 **TriggerBinding** 리소스를 생성할 수 있습니다.

```
$ oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/release-tech-preview-3/03_triggers/01_binding.yaml
```

- 다음 샘플 **TriggerTemplate** YAML 파일의 내용을 복사하여 저장합니다.

```
apiVersion: triggers.tekton.dev/v1alpha1
kind: TriggerTemplate
metadata:
  name: vote-app
spec:
  params:
    - name: git-repo-url
      description: The git repository url
    - name: git-revision
      description: The git revision
      default: release-tech-preview-3
    - name: git-repo-name
      description: The name of the deployment to be created / patched

  resourcetemplates:
    - apiVersion: tekton.dev/v1beta1
      kind: PipelineRun
      metadata:
        name: build-deploy-${tt.params.git-repo-name}-${uid}
      spec:
        serviceAccountName: pipeline
        pipelineRef:
          name: build-and-deploy
        params:
          - name: deployment-name
            value: ${tt.params.git-repo-name}
          - name: git-url
            value: ${tt.params.git-repo-url}
          - name: git-revision
            value: ${tt.params.git-revision}
          - name: IMAGE
            value: image-registry.openshift-image-registry.svc:5000/pipelines-tutorial/${tt.params.git-repo-name}
        workspaces:
          - name: shared-workspace
            volumeClaimTemplate:
              spec:
                accessModes:
                  - ReadWriteOnce
                resources:
                  requests:
                    storage: 500Mi
```

템플릿은 작업 영역의 스토리지 볼륨을 정의하기 위해 영구 볼륨 클레임을 생성하는 볼륨 클레임 템플릿을 지정합니다. 따라서 데이터 스토리지를 제공하기 위해 영구 볼륨 클레임을 생성할 필요가 없습니다.

4. **TriggerTemplate** 리소스를 생성합니다.

```
$ oc create -f <triggertemplate-yaml-file-name.yaml>
```

또는 **pipelines-tutorial** Git 리포지토리에서 직접 **TriggerTemplate** 리소스를 생성할 수도 있습니다.

```
$ oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/release-tech-preview-3/03_triggers/02_template.yaml
```

5. 다음 샘플 **Trigger** YAML 파일의 콘텐츠를 복사하여 저장합니다.

```
apiVersion: triggers.tekton.dev/v1alpha1
kind: Trigger
metadata:
  name: vote-trigger
spec:
  serviceAccountName: pipeline
  bindings:
    - ref: vote-app
  template:
    name: vote-app
```

6. **Trigger** 리소스를 생성합니다.

```
$ oc create -f <trigger-yaml-file-name.yaml>
```

또는 **pipelines-tutorial** Git 리포지토리에서 직접 **Trigger** 리소스를 생성할 수도 있습니다.

```
$ oc create -f https://github.com/openshift/pipelines-tutorial/blob/release-tech-preview-3/03_triggers/03_trigger.yaml
```

7. 다음 샘플 **EventListener** YAML 파일의 내용을 복사하여 저장합니다.

```
apiVersion: triggers.tekton.dev/v1alpha1
kind: EventListener
metadata:
  name: vote-app
spec:
  serviceAccountName: pipeline
  triggers:
    - triggerRef: vote-trigger
```

또는 트리거 사용자 정의 리소스를 정의하지 않은 경우 트리거 이름을 참조하는 대신 바인딩 및 템플릿 사양을 **EventListener** YAML 파일에 추가합니다.

```
apiVersion: triggers.tekton.dev/v1alpha1
kind: EventListener
metadata:
  name: vote-app
spec:
  serviceAccountName: pipeline
  triggers:
```

```
- bindings:
- ref: vote-app
template:
name: vote-app
```

8. **EventListener** 리소스를 생성합니다.

```
$ oc create -f <eventlistener-yaml-file-name.yaml>
```

또는 **pipelines-tutorial** Git 리포지토리에서 직접 **EventListener** 리소스를 생성할 수도 있습니다.

```
$ oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/release-tech-preview-3/03_triggers/04_event_listener.yaml
```

9. **EventListener** 서비스에 공개 액세스가 가능하도록 이 서비스를 OpenShift Container Platform 경로로 노출합니다.

```
$ oc expose svc el-vote-app
```

## 4.7. WEBHOOK 생성

*Webhooks*는 리포지토리에 구성된 이벤트가 발생할 때마다 **EventListener**가 수신하는 HTTP POST 메시지입니다. 이어서 이벤트 페이로드가 **TriggerBindings**에 매핑되고 **TriggerTemplates**에 의해 처리됩니다. **TriggerTemplates**는 최종적으로 Kubernetes 리소스를 생성 및 배포를 수행할 하나 이상의 **PipelineRuns**을 시작합니다.

여기서는 분기된 Git 리포지토리 **vote-ui**와 **vote-api**에 대한 Webhook URL을 구성합니다. 이 URL은 공개 액세스 가능한 **EventListener** 서비스 경로를 가리킵니다.



### 참고

Webhook를 추가하려면 리포지토리에 대한 관리자 권한이 필요합니다. 리포지토리에 대한 관리자 액세스 권한이 없으면 시스템 관리자에게 요청하여 Webhook를 추가하십시오.

### 프로세스

1. Webhook URL을 가져옵니다.

```
$ echo "URL: $(oc get route el-vote-app --template='http://{{.spec.host}}')"
```

출력에서 가져온 URL을 기록해 둡니다.

2. 프론트 엔드 리포지토리에서 수동으로 Webhook를 구성합니다.
  - a. 브라우저에서 프론트 엔드 Git 리포지토리 **vote-ui**를 엽니다.
  - b. **Settings** → **Webhook** → **Webhook** 추가를 클릭합니다.
  - c. **Webhooks/Add Webhook** 페이지에서:
    - i. **Payload URL** 필드에 1단계의 Webhook URL을 입력합니다.
    - ii. **Content type**으로 **application/json**을 선택합니다.



- iii. **Secret** 필드에 시크릿을 지정합니다.
  - iv. **Just the push event**이 선택되어 있는지 확인합니다.
  - v. **Active**를 선택하십시오.
  - vi. **Add Webhook**를 클릭합니다.
3. 백엔드 리포지토리 **vote-api**에 대해 2단계를 반복합니다.

## 4.8. 파이프라인 실행 트리거

Git 리포지토리에서 **push** 이벤트가 발생할 때마다 구성된 Webhook에서 공개 노출된 **EventListener** 서비스 경로로 이벤트 페이로드를 보냅니다. 애플리케이션의 **EventListener** 서비스는 페이로드를 처리하여 관련 **TriggerBinding** 및 **TriggerTemplate** 쌍으로 전달합니다. **TriggerBinding** 리소스는 매개변수를 추출하고 **TriggerTemplate** 리소스는 이러한 매개변수를 사용하여 리소스 생성 방식을 지정합니다. 그리고 애플리케이션을 다시 빌드 및 배포할 수도 있습니다.

이 섹션에서는 비어 있는 커밋을 프런트 엔드 **vote-ui** 리포지토리로 내보냅니다. 그러면 파이프라인 실행이 트리거됩니다.

### 프로세스

1. 터미널에서 분기된 Git 리포지토리 **vote-ui**를 복제합니다.

```
$ git clone git@github.com:<your GitHub ID>/vote-ui.git -b release-tech-preview-3
```

2. 비어 있는 커밋을 푸시합니다.

```
$ git commit -m "empty-commit" --allow-empty && git push origin release-tech-preview-3
```

3. 파이프라인 실행이 트리거되었는지 확인합니다.

```
$ tkn pipelinerun list
```

새로운 파이프라인 실행이 시작되었습니다.

## 4.9. 추가 리소스

- **개발자** 관점의 파이프라인에 대한 자세한 내용은 [개발자 관점에서 파이프라인 작업](#) 섹션을 참조하십시오.
- SCC(보안 컨텍스트 제약 조건)에 대한 자세한 내용은 [보안 컨텍스트 제약 조건 관리](#) 섹션을 참조하십시오.
- 재사용 가능 작업의 예를 더 보려면 [OpenShift 카탈로그](#) 리포지토리를 참조하십시오. Tekton 프로젝트의 Tekton 카탈로그도 참조할 수 있습니다.

## 5장. 개발자 관점을 사용하여 RED HAT OPENSIFT PIPELINES 작업

OpenShift Container Platform 웹 콘솔의 **Developer** 관점을 사용하여 소프트웨어 제공 프로세스를 위한 CI/CD Pipeline을 생성할 수 있습니다.

**Developer** 관점에서:

- **Add → Pipeline → Pipeline Builder** 옵션을 사용하여 애플리케이션에 사용자 지정된 Pipeline을 생성합니다.
- **Add → From Git** 옵션을 사용하여 OpenShift Container Platform에서 애플리케이션을 생성하는 동안 운영자 설치 Pipeline 템플릿과 리소스를 이용해 Pipeline을 생성합니다.

애플리케이션의 Pipeline을 생성한 후 **Pipelines** 보기에서 배포된 Pipeline을 보면서 시각적으로 상호 작용할 수 있습니다. **Topology** 보기에서도 **From Git** 옵션을 사용하여 생성된 Pipeline과 상호 작용할 수 있습니다. **Topology** 보기에서 볼 수 있으려면 **Pipeline Builder**를 사용하여 생성된 Pipeline에 사용자 지정 레이블을 적용해야 합니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 액세스하고 **개발자 화면으로 전환했습니다**.
- 클러스터에 **OpenShift Pipelines Operator**를 설치했습니다.
- 클러스터 관리자 또는 작성 및 편집 권한이 있는 사용자입니다.
- 프로젝트를 생성했습니다.

### 5.1. PIPELINE 빌더를 사용하여 PIPELINE 구성

콘솔의 **Developer** 관점에서 **Add → Pipeline → Pipeline Builder** 옵션을 사용하여 다음을 수행할 수 있습니다.

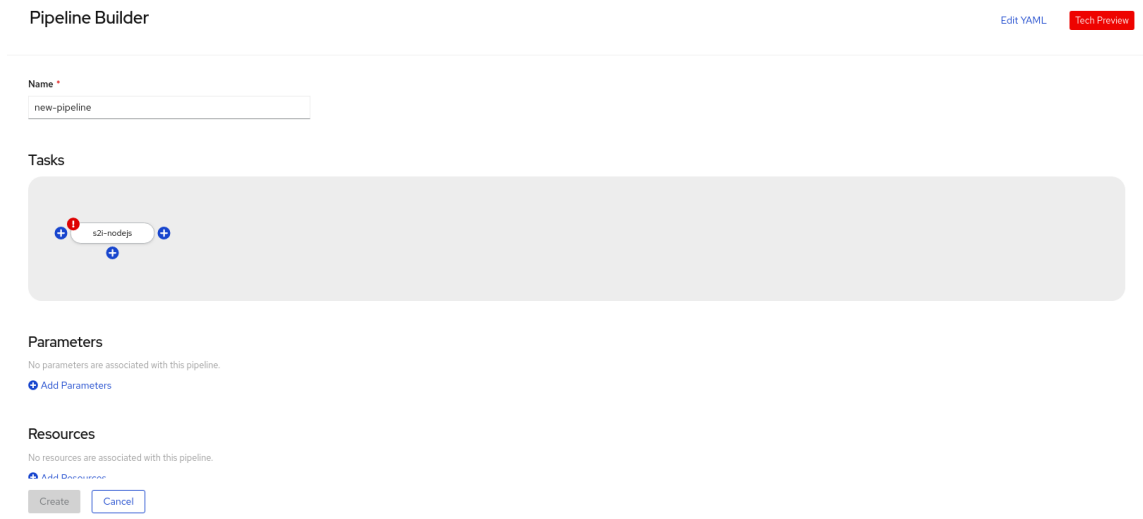
- 기존 Task 및 ClusterTasks를 사용하여 Pipeline 흐름을 구성합니다. OpenShift Pipelines Operator를 설치하면 재사용 가능한 Pipeline ClusterTasks가 클러스터에 추가됩니다.
- Pipeline Run에 필요한 리소스 유형을 지정하고, 필요하면 Pipeline에 매개변수를 더 추가합니다.
- Pipeline의 Task 각각에서 이러한 Pipeline 리소스를 입력 및 출력 리소스로 참조합니다.
- Task 매개변수는 Task의 사양에 따라 미리 채워집니다. 필요한 경우, Task의 Pipeline에 추가된 다른 매개변수도 참조합니다.

프로시저

1. **Developer** 관점의 **Add** 보기에서 **Pipeline** 타일을 클릭하여 **Pipeline Builder** 페이지를 표시합니다.
2. Pipeline의 고유한 이름을 입력합니다.
3. **Select task** 목록에서 Pipeline에 추가할 Task를 선택합니다. 예에서는 **s2i-nodejs** Task를 사용합니다.

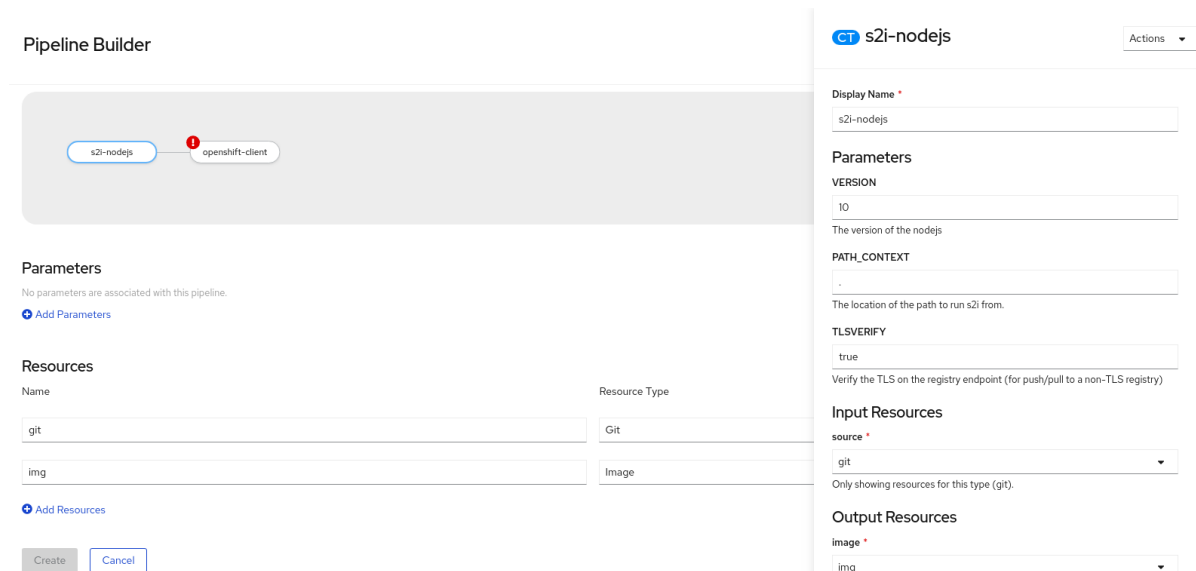
- Pipeline에 순차 Task를 추가하려면 Task 오른쪽 또는 왼쪽에 있는 더하기 아이콘을 클릭하고 **Select task** 목록에서 Pipeline에 추가할 Task를 선택합니다. 예를 들어, **openshift-client** Task를 추가하려면 **S2I-nodejs** Task 오른쪽에 있는 더하기 아이콘을 사용합니다.
- 기존 Task에 병렬 Task를 추가하려면 Task 아래에 표시된 더하기 아이콘을 클릭하고 **Select task** 목록에서 Pipeline에 추가할 병렬 Task를 선택합니다.

그림 5.1. Pipeline 빌더



- Add Resources**를 클릭하여 Pipeline Run에 사용할 리소스의 이름 및 유형을 지정합니다. 추가되는 리소스는 Pipeline의 Task에 입력 및 출력으로 사용됩니다. 예시의 경우:
  - 입력 리소스를 추가합니다. **Name** 필드에 **Source**를 입력하고 **Resource Type** 드롭다운 목록에서 **Git**을 선택합니다.
  - 출력 리소스를 추가합니다. **Name** 필드에 **Img**를 입력하고 **Resource Type** 드롭다운 목록에서 **Image**를 선택합니다.
- Task **Parameters**는 Task의 사양에 따라 미리 채워집니다. 필요하다면 **Add Parameters** 링크를 사용하여 매개변수를 더 추가합니다.
- Task의 리소스가 지정되지 않은 경우 Task에 **Missing Resources** 경고가 표시됩니다. **s2i-nodejs** Task를 클릭하여 측면 패널을 열어 Task에 대한 세부 사항을 확인합니다.

그림 5.2. Pipeline 빌더의 Task 세부 사항



7. Task 측면 패널에서 리소스 및 매개변수를 지정합니다.
  - a. **Input Resources** → **Source** 섹션의 **Select Resources** 드롭다운 목록에 사용자가 Pipeline에 추가한 리소스가 표시됩니다. 예에서는 **Source** 선택합니다.
  - b. **Output Resources** → **Image** 섹션에서 **Select Resources** 목록을 클릭하고 **Img**를 선택합니다.
  - c. 필요하다면 **Parameters** 섹션에서 **\$(params.<param-name>)** 구문을 사용하여 기본 매개변수에 매개변수를 더 추가합니다.
8. 마찬가지로 **openshift-client** Task에 대한 입력 리소스를 추가합니다.
9. **Create**를 클릭하여 Pipeline을 생성합니다. 생성된 Pipeline의 세부 사항을 표시하는 **Pipeline Details** 페이지로 이동됩니다.
10. **작업** 드롭다운 메뉴를 클릭한 다음 **시작**을 클릭하여 파이프라인을 시작합니다.

선택적으로 **Pipeline Builder** 페이지의 오른쪽 상단에 있는 **Edit YAML** 링크를 사용하여 콘솔에서 Pipeline YAML 파일을 직접 수정할 수도 있습니다. Operator가 설치한 재사용 가능한 스니펫과 샘플을 사용하여 세부 사항이 제공된 Pipeline을 생성할 수 있습니다.

## 5.2. OPENSIFT PIPELINES를 사용하여 애플리케이션 작성

애플리케이션과 함께 Pipeline을 생성하려면 **Developer** 관점의 **Add** 보기에서 **From Git** 옵션을 사용합니다. 자세한 정보는 [Creating applications using the Developer perspective](#) 을 참조하십시오.

## 5.3. 개발자 관점을 사용하여 PIPELINE과 상호 작용

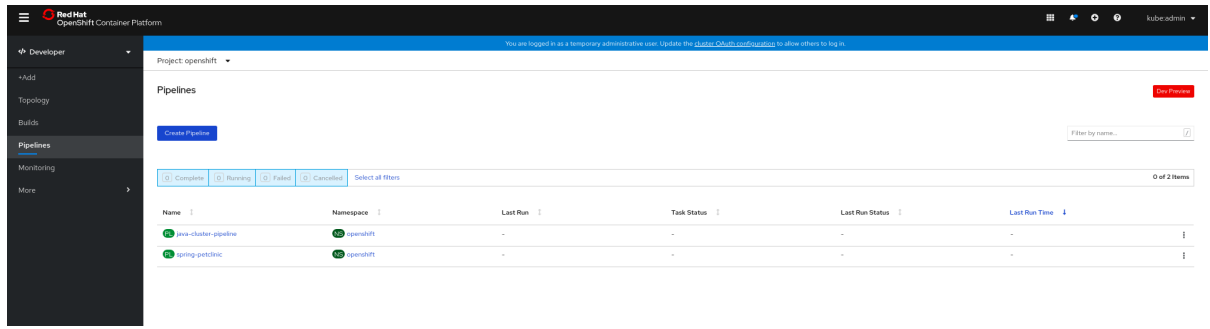
개발자 화면의 **파이프라인** 보기에는 다음 세부 정보와 함께 프로젝트의 모든 파이프라인이 나열됩니다.

- 파이프라인이 생성된 네임스페이스
- 마지막 파이프라인 실행
- 파이프라인 실행 시 작업 상태
- 파이프라인 실행의 상태
- 마지막 파이프라인 실행 생성 시간

### 절차

1. **Developer** 관점의 **Pipelines** 보기에서 **Project** 드롭다운 목록에서 프로젝트를 선택하여 해당 프로젝트의 Pipeline을 확인하십시오.

### 그림 5.3. Developer 관점에서 Pipeline 보기




- 필요한 Pipeline을 클릭하여 **Pipeline Details** 페이지를 확인하십시오. 이 페이지에는 Pipeline의 모든 직렬 및 병렬 Task가 시각적으로 표시됩니다. 페이지 오른쪽 아래에 Task 목록도 표시됩니다. 목록의 **Tasks**를 클릭하면 Task 세부 사항을 확인할 수 있습니다.
- 선택적으로 **Pipeline Details** 페이지에서 다음을 수행합니다.
  - YAML** 탭을 클릭하여 Pipeline의 YAML 파일을 편집합니다.
  - 파이프라인 실행** 탭을 클릭하여 파이프라인 실행 상태가 완료, 실행 중 또는 실패인지 확인합니다.



#### 참고

**파이프라인 실행 세부 정보** 페이지의 **세부 정보** 섹션에는 실패한 파이프라인 실행에 대한 **로그 조각**이 표시됩니다. **로그 조각**에는 일반적인 오류 메시지와 해당 로그의 조각이 있습니다. **로그** 섹션 링크를 사용하면 실패한 실행에 대한 세부 정보에 빠르게 액세스할 수 있습니다. **로그 조각**은 **작업 실행 세부 정보** 페이지의 **세부 정보** 섹션에도 표시됩니다.

옵션 메뉴 에서는 실행 중인 파이프라인 중지, 이전 파이프라인 실행과 동일한 매개변수 및 리소스를 사용하여 파이프라인 재실행 또는 파이프라인 실행 삭제 작업을 수행할 수 있습니다.


- 매개변수** 탭을 클릭하여 파이프라인에 정의된 매개변수를 확인합니다. 필요에 따라 매개변수를 더 추가하거나 편집할 수도 있습니다.
- 리소스** 탭을 클릭하여 파이프라인에 정의된 리소스를 확인합니다. 필요에 따라 리소스를 더 추가하거나 편집할 수도 있습니다.

## 5.4. PIPELINE 시작

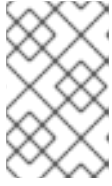
Pipeline을 생성한 후 포함된 Task를 정의된 순서로 실행하려면 Pipeline을 시작해야 합니다. **Pipelines** 보기, **Pipeline Details** 페이지 또는 **Topology** 보기에서 Pipeline Run을 시작할 수 있습니다.

### 프로시저

**Pipelines** 보기를 사용하여 Pipeline을 시작하려면 다음을 수행합니다.

- Developer** 관점의 **Pipelines** 보기에서 Pipeline 옆에 있는 **Options**  메뉴를 클릭하고 **Start**를 선택합니다.

2. Pipeline 정의에 따라 **Git Resource** 및 **Image Resources**가 **Start Pipeline** 대화 상자에 표시됩니다.



### 참고

**From Git** 옵션을 사용하여 생성한 Pipeline의 경우, **Start Pipeline** 대화 상자에는 **Parameters** 섹션의 **APP\_NAME** 필드도 표시되며, 대화 상자의 모든 필드는 Pipeline 템플릿으로 미리 채워집니다.

- a. 네임스페이스에 리소스가 있는 경우 **Git Resources** 및 **Image Resources** 필드에 해당 리소스가 미리 채워집니다. 필요하면 드롭다운 목록을 사용하여 필요한 리소스를 선택하거나 생성한 다음 Pipeline Run 인스턴스를 사용자 지정하십시오.
3. 선택 사항: **Advanced Options(고급 옵션)**를 수정하여 지정된 프라이빗 Git 서버 또는 Docker 레지스트리를 인증하기 위해 자격 증명을 추가합니다.
    - a. **Advanced Options**에서 **Show Credentials Options**를 클릭하고 **Add Secret**을 선택합니다.
    - b. **Create Source Secret** 섹션에서 다음 사항을 지정합니다.
      - i. 보안에 대한 고유한 **보안 이름**입니다.
      - ii. **Designated provider to be authenticated** 섹션에서 **Access to** 필드에 인증할 공급자를 지정하고 기본 **Server URL**을 지정합니다.
      - iii. **Authentication Type**을 선택하고 자격 증명을 제공합니다.
        - **Authentication Type Image Registry Credentials**의 경우 인증할 **Registry Server Address**를 지정하고 **Username, Password** 및 **Email** 필드에 자격 증명 정보를 제공합니다.  
추가 **Registry Server Address**를 지정하려면 **Add Credentials**를 선택하십시오.
        - **Authentication Type Basic Authentication**의 경우 **UserName** 및 **Password or Token** 필드 값을 지정합니다.
        - **Authentication Type SSH 키**의 경우 **SSH 개인 키** 필드 값을 지정합니다.
      - iv. 확인 표시를 선택하여 시크릿을 추가합니다.

Pipeline의 리소스 수에 따라 여러 시크릿을 추가할 수 있습니다.

4. **Start**를 클릭하여 PipelineRun을 시작합니다.
5. **Pipeline Run Details** 페이지에 실행 중인 Pipeline이 표시됩니다. Pipeline이 시작된 후 Task와 Task 내 단계(Step)가 실행됩니다. 다음을 수행할 수 있습니다.
  - Task 위로 마우스를 이동하여 단계(Step)별 실행 시간을 확인할 수 있습니다.
  - Task를 클릭하면 Task 내 단계(Step)별 로그를 볼 수 있습니다.
  - Task의 실행 순서에 따라 로그를 보려면 **Logs** 탭을 클릭하고 **Download** 버튼을 사용하여 로그를 텍스트 파일로 다운로드합니다.

## 그림 5.4. Pipeline Run

Pipeline Runs > Pipeline Run Details

**PLR** nodejs-ex-48nede Running

Details **YAML** Logs

### Pipeline Run Details

**Pipeline**  
**PL** nodejs-ex

**Triggered by:**  
 kubeadmin

**Pipeline Resources**  
**PR** git-eeaglz  
**PR** image-fx5obs

**Labels**  
 app.kubernetes.io/instance=nodejs-ex pipeline.openshift.io/runtime=nodejs  
 pipeline.openshift.io/started-by=kubeadmin pipeline.openshift.io/type=kubernetes tekton.dev/pipeline=nodejs-ex

**Annotations**  
 0 Annotations

**Created At**  
 3 minutes ago

6. **From Git** 옵션을 사용하여 생성한 Pipeline의 경우, **Topology** 보기를 사용하여 Pipeline을 시작한 후 상호 작용할 수 있습니다.

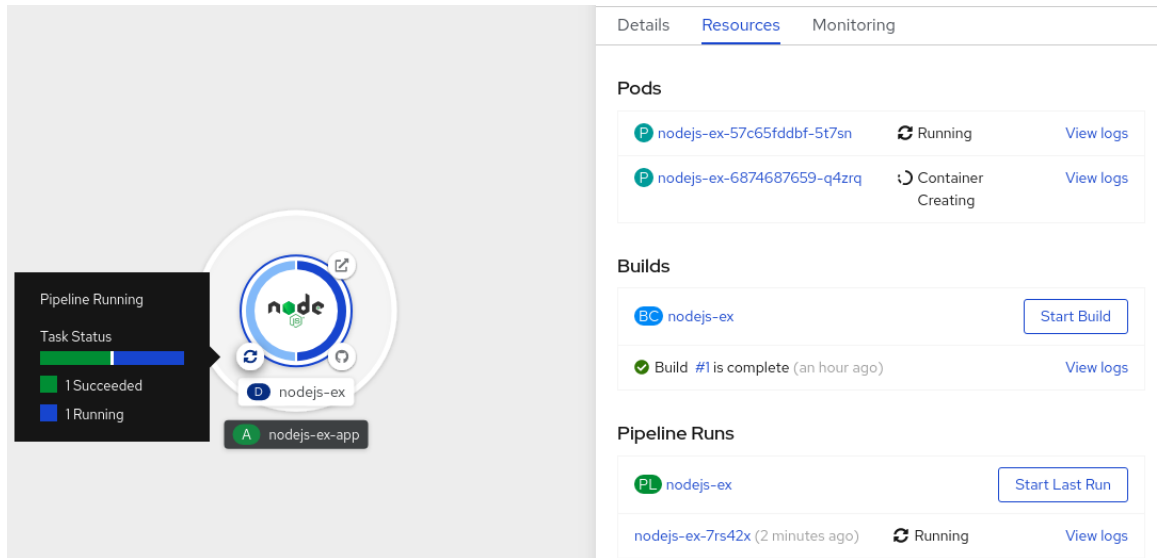


### 참고

**Topology** 보기에서 **Pipeline Builder**를 사용하여 생성한 Pipeline을 보려면 Pipeline 레이블을 사용자 지정하여 Pipeline을 애플리케이션 워크로드에 연결합니다.

- 왼쪽 탐색 패널에서 **Topology**를 클릭하고 애플리케이션을 클릭하여 사이드 패널에서 Pipeline Run 목록을 확인하십시오.
- Pipeline Run** 섹션에서 **Start Last Run**을 클릭하여 이전 Pipeline과 동일한 매개변수 및 리소스로 새 Pipeline 런을 시작합니다. Pipeline Run이 시작되지 않았으면 이 옵션을 사용할 수 없습니다.

그림 5.5. 토폴로지 보기의 Pipeline



- c. 토폴로지 페이지에서 애플리케이션 왼쪽으로 커서를 이동하여 애플리케이션의 파이프라인 실행 상태를 확인합니다.



참고

토폴로지 페이지의 애플리케이션 노드 사이드 패널에는 파이프라인 실행이 특정 작업 실행에서 실패할 때 로그 조각이 표시됩니다. 로그 조각은 리소스 탭의 파이프라인 실행 섹션에서 확인할 수 있습니다. 로그 조각에는 일반적인 오류 메시지와 해당 로그의 조각이 있습니다. 로그 섹션 링크를 사용하면 실패한 실행에 대한 세부 정보에 빠르게 액세스할 수 있습니다.

### 5.5. PIPELINE 편집

웹 콘솔의 개발자 화면을 사용하여 클러스터의 Pipeline을 편집할 수 있습니다.

프로세스


1. 개발자 화면의 Pipelines 보기에서 편집할 Pipeline을 선택하여 Pipeline의 세부 사항을 표시합니다. Pipeline Details 페이지에서 Actions를 클릭하고 Edit Pipelin을 선택합니다.
2. Pipeline Builder 페이지에서:
  - 추가 Task, 매개변수 또는 리소스를 Pipeline에 추가할 수 있습니다.
  - 수정하려는 Task를 클릭하면 측면 패널에 Task 세부 사항이 표시됩니다. 여기서 표시 이름, 매개변수 및 리소스와 같이 필요한 Task 세부 사항을 수정할 수 있습니다.
  - 또는 Task를 클릭하고 측면 패널에서 Actions를 클릭하고 Remove Task를 선택하여 Task를 삭제할 수도 있습니다.
3. Save를 클릭하여 수정된 Pipeline을 저장합니다.

### 5.6. PIPELINE 삭제

웹 콘솔의 개발자 화면을 사용하여 클러스터의 Pipeline을 삭제할 수 있습니다.



## 프로세스

1. 개발자 화면의 **Pipelines** 보기에서 Pipeline 옆에 있는 **Options**  메뉴를 클릭하고 **Delete Pipeline**를 선택합니다.
2. **Delete Pipeline** 확인 프롬프트에서 **Delete**를 클릭하여 삭제를 확인합니다.

## 6장. RED HAT OPENSIFT PIPELINES 릴리스 정보

Red Hat OpenShift Pipelines는 다음을 제공하는 Tekton 프로젝트를 기반으로 하는 클라우드 네이티브 CI/CD 환경입니다.

- 표준 CRD(Kubernetes 네이티브 Pipeline 정의)
- CI 서버 관리 오버헤드가 없는 서버리스 Pipeline
- S2I, Buildah, JIB 및 Kaniko와 같은 Kubernetes 도구를 사용하여 이미지를 빌드할 수 있는 확장성
- 모든 Kubernetes 배포판에서 이식성
- Pipeline과 상호 작용하기 위한 강력한 CLI
- OpenShift Container Platform 웹 콘솔의 개발자 관점과 통합된 사용자 경험

Red Hat OpenShift Pipelines 개요는 [OpenShift Pipelines 이해](#)를 참조하십시오.

### 6.1. 지원 요청

이 문서에 설명된 프로시저를 따르는 데 어려움이 있으면 Red Hat 고객 포털을 방문하여 [Red Hat Technology Preview features support scope](#)에 대해 알아보십시오.

질문이나 의견이 있으시면 제품팀에 이메일([pipelines-interest@redhat.com](mailto:pipelines-interest@redhat.com))로 보내주시기 바랍니다.

## 6.2. RED HAT OPENSIFT PIPELINES TECHNOLOGY PREVIEW 1.2 릴리스 정보

### 6.2.1. 새로운 기능

이제 OpenShift Container Platform 4.6에서 Red Hat OpenShift Pipelines TP(Technology Preview) 1.2를 사용할 수 있습니다. 다음을 지원하도록 Red Hat OpenShift Pipelines TP 1.2가 업데이트되었습니다.

- Tekton Pipelines 0.16.3
- Tekton **tkn** CLI 0.13.1
- Tekton Triggers 0.8.1
- Tekton Catalog 0.16 기반 ClusterTasks
- OpenShift Container Platform 4.6의 IBM Power Systems
- OpenShift Container Platform 4.6의 IBM Z 및 LinuxONE

수정 및 안정성 개선 사항 외에 OpenShift Pipelines 1.2의 새로운 기능도 소개합니다.

#### 6.2.1.1. 파이프라인

- 이 Red Hat OpenShift Pipelines 릴리스에서는 오프라인 설치를 지원합니다.



## 참고

제한된 환경에서의 설치에 현재 IBM Power Systems, IBM Z, LinuxONE에서 지원되지 않습니다.

- 이제 **conditions** 대신 **when** 필드를 사용하여 특정 기준이 충족될 때만 작업을 실행할 수 있습니다. **WhenExpressions**의 주요 구성 요소는 **Input, Operator, Values**입니다. 모든 **WhenExpressions**가 **True**로 평가되면 작업이 실행됩니다. **WhenExpressions**가 **False**로 평가되면 작업을 건너뛵니다.
- 작업 실행이 취소되거나 타임아웃되는 경우 단계 상태가 업데이트됩니다.
- **git-init**에서 사용하는 기본 이미지를 빌드하는 데 Git LFS(Large File Storage) 지원이 제공됩니다.
- **taskSpec** 필드를 사용하여 작업이 파이프라인에 포함된 경우 라벨 및 주석과 같은 메타데이터를 지정할 수 있습니다.
- 파이프라인 실행에서 클라우드 이벤트를 지원합니다. 클라우드 이벤트 파이프라인 리소스에서 보내는 클라우드 이벤트에 **backoff**를 통해 재시도할 수 있습니다.
- **Task** 리소스에서 선언해도 **TaskRun** 리소스에서 명시적으로 제공하지 않는 모든 작업 공간에 기본 **Workspace** 구성을 설정할 수 있습니다.
- **PipelineRun** 네임스페이스 및 **TaskRun** 네임스페이스에 대한 네임스페이스 변수 보간을 지원합니다.
- **TaskRun** 리소스가 유사성 도우미와 연결되어 있을 때 여러 개의 영구 볼륨 클레임 작업 공간이 사용되지 않는지 확인하기 위해 **TaskRun** 오브젝트에 대한 검증 작업이 추가되었습니다. 영구 볼륨 클레임 작업 공간이 두 개 이상 사용되면 **TaskRunValidationFailed** 조건이 포함된 작업 실행이 실패합니다. 기본적으로 유사성 도우미는 Red Hat OpenShift Pipelines에서 비활성화되어 있으므로 이 도우미를 사용하려면 활성화해야 합니다.

### 6.2.1.2. Pipeline CLI

- **tkn task describe, tkn taskrun describe, tkn clustertask describe, tkn pipeline describe, tkn pipelinerun describe** 명령에서 다음을 수행합니다.
  - **Task, TaskRun, ClusterTask, Pipeline, PipelineRun** 리소스 중 하나만 있는 경우 각 리소스를 자동으로 선택합니다.
  - **Task, TaskRun, ClusterTask, Pipeline, PipelineRun** 리소스의 결과를 출력에 각각 표시합니다.
  - **Task, TaskRun, ClusterTask, Pipeline, PipelineRun** 리소스에 선언된 작업 공간을 각각 표시합니다.
- **tkn clustertask start** 명령에 **--prefix-name** 옵션을 사용하여 작업 실행 이름에 접두사를 지정할 수 있습니다.
- **tkn clustertask start** 명령에 대화형 모드가 지원됩니다.
- **TaskRun** 및 **PipelineRun** 오브젝트에 대한 로컬 또는 원격 파일 정의를 사용하여 파이프라인에서 지원하는 **PodTemplate** 속성을 지정할 수 있습니다.
- **tkn clustertask start** 명령에 **--use-params-defaults** 옵션을 사용하여 **ClusterTask** 구성에 설정된 기본값을 사용하고 작업 실행을 생성할 수 있습니다.

- 일부 매개변수에 기본값이 지정되지 않은 경우 **tkn pipeline start** 명령의 **--use-param-defaults** 플래그는 대화형 모드를 표시합니다.

### 6.2.1.3. Trigger

- **parseYAML**이라는 CEL(Common Expression Language) 함수가 추가되어 YAML 문자열을 문자열 맵으로 구문 분석합니다.
- CEL 표현식 구문 분석에 대한 오류 메시지가 개선되어 표현식을 평가하는 동안 그리고 평가 환경을 생성하기 위해 후크 본문을 구문 분석할 때 더 세부적인 내용이 표시됩니다.
- 부울 값 및 맵이 CEL 오버레이 메커니즘에서 표현식 값으로 사용되는 경우 부울 값 및 맵 마살링이 지원됩니다.
- 다음 필드가 **EventListener** 오브젝트에 추가되었습니다.
  - **replicas** 필드를 사용하면 YAML 파일의 복제본 수를 지정하여 이벤트 리스너에서 여러 개의 Pod를 실행할 수 있습니다.
  - **NodeSelector** 필드를 사용하면 **EventListener** 오브젝트에서 이벤트 리스너 Pod를 특정 노드에 예약할 수 있습니다.
- Webhook 인터셉터에서 **EventListener-Request-URL** 헤더를 구문 분석하여 이벤트 리스너로 처리 중인 원래 요청 URL에서 매개변수를 추출할 수 있습니다.
- 이벤트 리스너에 있는 주석을 배포 Pod, 서비스 Pod 및 기타 Pod로 전파할 수 있습니다. 서비스 또는 배포에 대한 사용자 정의 주석을 덮어쓰므로 해당 주석을 전파하려면 이벤트 리스너 주석에 추가해야 합니다.
- 사용자가 **spec.replicas** 값을 **negative** 또는 **zero**로 지정하는 경우에도 **EventListener** 사양의 복제본을 올바르게 검증할 수 있습니다.
- **TriggerRef** 필드를 통해 **EventListener** 사양 내의 **TriggerCRD** 오브젝트를 참조로 지정하여 **TriggerCRD** 오브젝트를 별도로 생성한 다음 **EventListener** 사양 내에서 바인딩할 수 있습니다.
- **TriggerCRD** 오브젝트에 검증을 수행하고 기본값을 사용할 수 있습니다.

### 6.2.2. 더 이상 사용되지 않는 기능

- **resourcetemplate** 매개변수와 **triggertemplate** 매개변수를 혼동하지 않도록 **\$(params)**를 제거하고 **\$(tt.params)**로 교체했습니다.
- 선택적 **EventListenerTrigger** 기반 인증 수준의 **ServiceAccount** 참조가 오브젝트 참조에서 **ServiceAccountName** 문자열로 변경되었습니다. 이로 인해 **ServiceAccount** 참조가 **EventListenerTrigger** 오브젝트와 동일한 네임스페이스에 있습니다.
- **Conditions** CRD(사용자 정의 리소스 정의)가 더 이상 사용되지 않습니다. 대신 **WhenExpressions** CRD를 사용합니다.
- **PipelineRun.Spec.ServiceAccountNames** 오브젝트가 더 이상 사용되지 않고 **PipelineRun.Spec.TaskRunSpec[].ServiceAccountName** 오브젝트로 교체됩니다.

### 6.2.3. 확인된 문제

- 이 Red Hat OpenShift Pipelines 릴리스에서는 오프라인 설치를 지원합니다. 그러나 클러스터 작업에서 사용하는 일부 이미지는 연결이 끊긴 클러스터에서 작업하려면 미러링해야 합니다.

- **openshift** 네임스페이스의 파이프라인은 Red Hat OpenShift Pipelines Operator를 설치 제거한 후에도 삭제되지 않습니다. 이 파이프라인을 삭제하려면 **oc delete pipelines -n openshift --all** 명령을 사용합니다.
- Red Hat OpenShift Pipelines Operator를 설치 제거해도 이벤트 리스너는 제거되지 않습니다. 해결 방법은 **EventListener** 및 **Pod** CRD를 제거하는 것입니다.

1. **foregroundDeletion** 종료자를 사용하여 **EventListener** 오브젝트를 다음과 같이 편집합니다.

```
$ oc patch el/<eventlistener_name> -p '{"metadata":{"finalizers":["foregroundDeletion"]}}' --type=merge
```

예를 들면 다음과 같습니다.

```
$ oc patch el/github-listener-interceptor -p '{"metadata":{"finalizers":["foregroundDeletion"]}}' --type=merge
```

2. **EventListener** CRD를 삭제합니다.

```
$ oc patch crd/eventlisteners.triggers.tekton.dev -p '{"metadata":{"finalizers":[]}}' --type=merge
```

- IBM Power Systems(ppc64le) 또는 IBM Z(s390x) 클러스터에서 명령 사양 없이 다중 아키텍처 컨테이너 이미지 작업을 실행하면 **TaskRun** 리소스가 실패하고 다음 오류 메시지가 표시됩니다.

```
Error executing command: fork/exec /bin/bash: exec format error
```

해결 방법은 아키텍처별 컨테이너 이미지를 사용하거나 sha256 다이제스트를 지정하여 올바른 아키텍처를 가리키는 것입니다. sha256 다이제스트를 가져오려면 다음을 입력합니다.

```
$ skopeo inspect --raw <image_name> | jq '.manifests[] | select(.platform.architecture == "<architecture>") | .digest'
```

#### 6.2.4. 해결된 문제

- CEL 필터, Webhook 유효성 검증기의 오버레이, 인터셉터의 표현식을 확인하는 간단한 구문 검증 기능이 추가되었습니다.
- Trigger가 더 이상 기본 배포 및 서비스 오브젝트에 설정된 주석을 덮어쓰지 않습니다.
- 이전에는 이벤트 리스너에서 이벤트 수락을 중지했습니다. 이 수정에서는 이 문제를 해결하기 위해 **EventListener** 싱크에 120초의 유힬 상태 타임아웃이 추가되었습니다.
- 이전에는 **Failed(Canceled)** 상태로 파이프라인 실행을 취소하면 성공 메시지가 표시되었습니다. 이제 오류 메시지를 표시하도록 수정되었습니다.
- **tkn eventlistener list** 명령에서 나열된 이벤트 리스너의 상태를 제공하므로 사용 가능한 리스너를 쉽게 확인할 수 있습니다.
- 트리거가 설치되지 않았거나 리소스가 없는 경우 **triggers list** 및 **triggers describe** 명령에 대한 오류 메시지가 일관되게 표시됩니다.

- 이전에는 클라우드 이벤트를 제공하는 동안 다수의 유틸리티 연결이 빌드되었습니다. 이 문제를 해결하기 위해 **cloudeventclient** 구성에 **DisableKeepAlives: true** 매개변수가 추가되었습니다. 따라서 모든 클라우드 이벤트에 대해 새로운 연결이 설정됩니다.
- 이전에는 지정된 유형의 자격 증명이 제공되지 않은 경우에도 **creds-init** 코드에서 디스크에 빈 파일을 작성했습니다. 이번 수정에서는 올바른 주석이 있는 보안에서 실제로 마운트된 자격 증명 이 있는 경우에만 파일을 작성하도록 **creds-init** 코드를 수정했습니다.

## 6.3. RED HAT OPENSIFT PIPELINES TECHNOLOGY PREVIEW 1.1 릴리스 정보

### 6.3.1. 새로운 기능

이제 OpenShift Container Platform 4.6에서 Red Hat OpenShift Pipelines TP(Technology Preview) 1.1을 사용할 수 있습니다. 다음을 지원하도록 Red Hat OpenShift Pipelines TP 1.1이 업데이트되었습니다.

- Tekton Pipelines 0.14.3
- Tekton **tkn** CLI 0.11.0
- Tekton Triggers 0.6.1
- Tekton Catalog 0.14 기반 ClusterTasks

수정 및 안정성 개선 사항 외에 OpenShift Pipelines 1.1의 새로운 기능도 소개합니다.

#### 6.3.1.1. Pipeline

- 이제 PipelineResources 대신 Workspace를 사용할 수 있습니다. PipelineResources는 디버깅하기 어렵고 범위가 제한되며 Task의 재사용 가능성을 낮추기 때문에 OpenShift Pipelines에서 Workspace를 사용할 것을 권장합니다. Workspace에 대한 자세한 내용은 OpenShift Pipelines 이 해를 참조하십시오.
- VolumeClaimTemplates에 대한 Workspace 지원이 추가되었습니다.
  - 이제 PipelineRun 및 TaskRun에 대한 VolumeClaimTemplate을 Workspace의 볼륨 소스로서 추가할 수 있습니다. 그런 다음 tekton-controller가 Pipeline의 모든 TaskRun에 대해 PVC로 표시되는 템플릿을 사용하여 PVC(PersistentVolumeClaim)를 생성합니다. 따라서 여러 Task에 걸쳐 있는 Workspace를 바인드할 때마다 PVC 구성을 정의해야 합니다.
  - VolumeClaimTemplate이 볼륨 소스로 사용될 때 PersistentVolumeClaim의 이름 검색 지원에서 이제 변수 대체를 사용하여 사용할 수 있습니다.
- 감사 개선 지원:
  - 이제 **PipelineRun.Status** 필드에 Pipeline의 모든 TaskRun 상태와 PipelineRun의 진행 상황을 모니터링하기 위해 PipelineRun을 인스턴스화하는 데 사용되는 Pipeline 사양이 포함됩니다.
  - Pipeline 결과가 Pipeline 사양 및 **PipelineRun** 상태에 추가되었습니다.
  - 이제 **TaskRun.Status** 필드에 **TaskRun**을 인스턴스화하는 데 사용되는 정확한 Task 사양이 포함됩니다.
- Condition에 기본 매개변수 적용을 지원합니다.

- ClusterTask를 참조하여 생성된 TaskRun이 이제 **tekton.dev/task** 레이블 대신 **tekton.dev/clusterTask** 레이블을 추가합니다.
- 이제 **kubeconfigwriter**가 리소스 구조에 **ClientKeyData** 및 **ClientCertificateData** 구성을 추가하여 Pipeline 리소스 유형 클러스터를 kubeconfig-creator Task로 교체할 수 있습니다.
- 이제 **feature-flags** 및 **config-defaults** ConfigMap의 이름을 이제 사용자 지정할 수 있습니다.
- 이제 TaskRun에 사용되는 PodTemplate에서 HostNetwork를 지원할 수 있습니다.
- 이제 Affinity Assistant를 사용하여 Workspace 볼륨을 공유하는 TaskRuns에서 노드 선호도를 지원할 수 있습니다. OpenShift Pipelines에서는 기본적으로 노드 선호도가 비활성화됩니다.
- PodTemplate이 **imagePullSecrets**를 지정하도록 업데이트되어 Pod를 시작할 때 컨테이너 런타임에서 컨테이너 이미지 가져오기를 승인하는 데 사용할 보안을 확인합니다.
- 컨트롤러가 TaskRun을 업데이트하지 못하는 경우 TaskRun 컨트롤러에서 경고 이벤트를 발송하도록 지원합니다.
- 애플리케이션 또는 구성 요소에 속하는 리소스를 식별하도록 표준 또는 권장 k8s 레이블이 모든 리소스에 추가되었습니다.
- 이제 Entrypoint 프로세스에 신호 알림이 전송되며, 이러한 신호는 Entrypoint 프로세스의 전용 PID Group을 사용하여 전파됩니다.
- 이제 **TaskRunSpecs**를 사용하여 런타임 시 Task 수준에서 PodTemplate을 설정할 수 있습니다.
- Kubernetes 이벤트 발송 지원:
  - 이제 컨트롤러가 추가 TaskRun 수명 주기 이벤트(**taskrun started** 및 **taskrun running**)에 대한 이벤트를 발송합니다.
  - 이제 PipelineRun 컨트롤러가 Pipeline이 시작될 때마다 이벤트를 발송합니다.
- 이제 기본 Kubernetes 이벤트 외에 TaskRuns에 대한 CloudEvents 지원도 제공됩니다. 생성, 시작 및 실패와 같은 TaskRun 이벤트를 클라우드 이벤트로서 발송하도록 컨트롤러를 구성할 수 있습니다.
- PipelineRuns 및 TaskRuns에서 적절한 이름을 참조하도록 **\$context.  
<task|taskRun|pipeline|pipelineRun>.name** 변수 사용을 지원합니다.
- 이제 PipelineRun 매개변수에 대한 유효성 검사를 사용하여 PipelineRun에서 Pipeline에 필요한 모든 매개변수가 제공되는지 확인할 수 있습니다. 이를 통해 PipelineRuns에서 필수 매개변수 외에 추가 매개변수도 제공할 수 있습니다.
- 이제 Pipeline YAML 파일의 **finally** 필드를 사용하여 모든 Task를 성공적으로 완료한 후 또는 Pipeline의 Task 중 하나가 실패한 후 Pipeline이 종료되기 전에 항상 실행될 Pipeline 내 Task를 지정할 수 있습니다.
- 이제 **git-clone** ClusterTask를 사용할 수 있습니다.

### 6.3.1.2. Pipeline CLI

- 이제 포함된 Trigger 바인딩 지원을 **tkn evenlistener describe** 명령에 사용할 수 있습니다.
- 하위 명령을 권장하고 잘못된 하위 명령을 사용할 때 의견을 제시하는 기능을 지원합니다.

- 이제 Pipeline에 Task가 한 개뿐인 경우 **tkn task describe** 명령에 의해 Task가 자동으로 선택됩니다.
- 이제 **tkn task start** 명령에 **--use-param-defaults** 플래그를 지정하여 기본 매개변수 값으로 Task를 시작할 수 있습니다.
- 이제 **tkn pipeline start** 또는 **tkn task start** 명령과 함께 **--workspace** 옵션을 사용하여 PipelineRuns 또는 TaskRuns에 대한 volumeClaimTemplate을 이제 지정할 수 있습니다.
- 이제 **tkn pipelinerun logs** 명령으로 **finally** 섹션에 나열된 최종 Task에 대한 로그를 표시할 수 있습니다.
- 이제 **tkn task start** 명령과 함께 **pipeline, pipelinerun, task, taskrun, clustertask** 및 **pipelineresource**와 같은 tkn 리소스에 대한 **describe** 하위 명령에 대화형 모드가 지원됩니다.
- 이제 **tkn version** 명령으로 클러스터에 설치된 Trigger의 버전을 표시할 수 있습니다.
- 이제 **tkn pipeline describe** 명령으로 Pipeline에 사용된 Task에 대해 지정된 매개변수 값과 시간 초과 사항을 표시할 수 있습니다.
- **tkn pipelinerun describe** 및 **tkn taskrun describe** 명령에 가장 최근 PipelineRun 또는 TaskRun을 각각 설명하는 **--last** 옵션에 대한 지원이 추가되었습니다.
- 이제 **tkn pipeline describe** 명령으로 Pipeline의 Task에 적용 가능한 조건을 표시할 수 있습니다.
- 이제 **tkn resource list** 명령과 함께 **--no-headers** 및 **--all-namespaces** 플래그를 사용할 수 있습니다.

### 6.3.1.3. Trigger

- 이제 다음과 같은 CEL(Common Expression Language) 기능을 사용할 수 있습니다.
  - URL의 일부를 구문 분석하고 추출하기 위한 **parseURL**
  - **deployment** WebHook의 **payload** 필드에 있는 문자열에 포함된 JSON 값 유형을 구문 분석하는 **parseJSON**
- Bitbucket의 WebHook에 대한 새로운 인터셉터가 추가되었습니다.
- 이제 EventListeners가 **kubectl get** 명령으로 나열될 때 추가 필드로 **Address URL** 및 **Available status**를 표시합니다.
- TriggerTemplate과 ResourceTemplates 매개변수 간의 혼동을 줄이기 위해 이제 TriggerTemplate 매개변수들에 **\$(params.<paramName>)** 대신 **\$(tt.params.<paramName>)** 구문을 사용합니다.
- 보안 또는 관리 문제로 인해 모든 노드가 오염된 경우에도 EventListeners가 동일한 구성으로 배포되도록 EventListener CRD에 **허용 오차**를 추가할 수 있습니다.
- 이제 **URL/live**에서 EventListener 배포에 대한 준비 프로브를 추가할 수 있습니다.
- EventListener Trigger에 TriggerBinding 사양 포함 지원
- 이제 권장 **app.kubernetes.io** 레이블을 사용하여 Trigger 리소스에 주석을 삽입할 수 있습니다.

### 6.3.2. 사용되지 않는 기능



이 릴리스에서는 더 이상 사용되지 않은 기능은 다음과 같습니다.

- **clustertask** 및 **clustertriggerbinding** 명령을 포함하여 모든 클러스터 단위 명령에 **--namespace** 또는 **-n** 플래그는 더 이상 사용되지 않습니다. 향후 릴리스에서 제거됩니다.
- EventListener 내 **triggers.bindings**의 **name** 필드가 더 이상 사용되지 않고 향후 릴리스에서 제거될 것이며 **ref** 필드 사용을 권장합니다.
- Pipeline 변수 보간 구문과 혼동을 줄이기 위해 **\$(params)**를 사용한 TriggerTemplates의 변수 보간은 더 이상 사용되지 않고, **\$(tt.params)** 사용을 권장합니다. **\$(params.<paramName>)** 구문은 향후 릴리스에서 제거됩니다.
- ClusterTasks에서 **tekton.dev/task** 레이블이 더 이상 사용되지 않습니다.
- **TaskRun.Status.ResourceResults.ResourceRef** 필드가 더 이상 사용되지 않으며 제거됩니다.
- **tkn pipeline create**, **tkn task create** 및 **tkn resource create -f** 하위 명령이 제거되었습니다.
- **tkn** 명령에서 네임스페이스 유효성 검사가 제거되었습니다.
- 기본 시간 초과 **1h**와 **tkn ct start** 명령에 대한 **-t** 플래그가 제거되었습니다.
- **s2i** ClusterTask가 더 이상 사용되지 않습니다.

### 6.3.3. 확인된 문제

- Condition이 Workspace를 지원하지 않습니다.
- **tkn clustertask start** 명령에 **--workspace** 옵션과 대화형 모드가 지원되지 않습니다.
- **\$(params.<paramName>)**의 역호환성 지원에 따라 Pipeline 특정 매개변수와 함께 TriggerTemplates를 사용하도록 수정되었습니다. 이는 Triggers WebHook에서 Trigger 매개변수와 Pipeline 매개변수와 구별할 수 없기 때문입니다.
- **tekton\_taskrun\_count** 및 **tekton\_taskrun\_duration\_seconds\_count**에 대한 promQL 쿼리를 실행할 때 Pipeline 메트릭이 잘못된 값을 보고합니다.
- Workspace에 제공된 기존 PVC 이름이 없는 경우에도 PipelineRuns 및 TaskRuns이 **Running** 및 **Running(Pending)** 상태를 각각 유지합니다.

### 6.3.4. 해결된 문제

- 전에는 Task와 ClusterTask 이름이 동일할 때 **tkn task delete<name>--trs** 명령으로 Task와 ClusterTask가 모두 삭제되었습니다. 수정판에서는 이 명령으로 Task <name>에 의해 생성된 TaskRuns만 삭제됩니다.
- 전에는 **tkn pr delete -p<name>--keep 2** 명령을 **--keep** 플래그와 함께 사용할 때 **-p** 플래그가 무시되고 최근 두 개를 제외한 모든 PipelineRun이 삭제되었습니다. 수정판에서는 이 명령으로 최근 두 개를 제외하고 Pipeline <name>에 의해 생성된 PipelineRuns만 삭제됩니다.
- 이제 **tkn triggertemplate describe** 출력에 YAML 형식 대신 테이블 형식으로 ResourceTemplates가 표시됩니다.
- 전에는 컨테이너에 새 사용자를 추가할 때 **buildah** ClusterTask가 실패했습니다. 수정판에서는 이러한 문제가 해결되었습니다.

## 6.4. RED HAT OPENSIFT PIPELINES TECHNOLOGY PREVIEW 1.0 릴리스 정보

### 6.4.1. 새로운 기능

이제 OpenShift Container Platform 4.6에서 Red Hat OpenShift Pipelines TP(Technology Preview) 1.0을 사용할 수 있습니다. 다음을 지원하도록 Red Hat OpenShift Pipelines TP 1.0이 업데이트되었습니다.

- Tekton Pipelines 0.11.3
- Tekton **tkn** CLI 0.9.0
- Tekton Triggers 0.4.0
- Tekton Catalog 0.11 기반 ClusterTasks

수정 및 안정성 개선 사항 외에 OpenShift Pipelines 1.0의 새로운 기능을 소개합니다.

#### 6.4.1.1. Pipeline

- v1beta1 API 버전을 지원합니다.
- 개선된 LimitRange를 지원합니다. 전에는 TaskRun 및 PipelineRun에 대해 LimitRange가 단독으로 지정되었습니다. 이제 LimitRange를 명시적으로 지정할 필요가 없습니다. 네임스페이스에서 최소 LimitRange가 사용됩니다.
- TaskResults 및 TaskParams를 사용하여 Task 간 데이터 공유를 지원합니다.
- 이제 **HOME** 환경 변수와 단계(Step)의 **workingDir**을 겹쳐쓰지 않도록 Pipeline을 구성할 수 있습니다.
- Task 단계(Step)와 유사하게 **sidecars**가 이제 스크립트 모드를 지원합니다.
- 이제 TaskRun **podTemplate**에서 다른 스케줄러 이름을 지정할 수 있습니다.
- Star Array Notation을 사용한 변수 대체를 지원합니다.
- 이제 개별 네임스페이스를 모니터링하도록 Tekton Controller를 구성할 수 있습니다.
- 이제 새로운 설명 필드가 Pipeline, Task, ClusterTask, Resource 및 Condition의 사양에 추가되었습니다.
- Git PipelineResources에 프록시 매개변수 추가

#### 6.4.1.2. Pipeline CLI

- 이제 **eventlistener**, **condition**, **triggertemplate**, **clustertask**, 및 **triggerbinding**와 같은 **tkn** 리소스에 **describe** 하위 명령이 추가됩니다.
- **v1alpha1**에 대한 역호환성 지원과 함께 **clustertask**, **task**, **pipeline**, **pipelinerun**, 및 **taskrun** 명령에 **v1beta1** 지원이 추가되었습니다.
- 이제 다음 명령에 **--all-namespaces** 플래그 옵션을 사용하여 모든 네임스페이스 목록을 출력할 수 있습니다.
  - **tkn task list**

- **tkn pipeline list**
- **tkn taskrun list**
- **tkn pipelinerun list**  
**--no-headers** 플래그 옵션을 사용하면 명령의 출력에 헤더 없이 정보가 표시되도록 향상되었습니다.
- 이제 **tkn pipelines start** 명령에서 **--use-param-defaults** 플래그를 지정하여 기본 매개변수 값을 사용하여 Pipeline을 시작할 수 있습니다.
- 이제 **tkn pipeline start** 및 **tkn task start** 명령에 Workspace에 대한 지원이 추가되었습니다.
- **describe, delete, list** 하위 명령과 함께 이제 새로운 **clustertriggerbinding** 명령이 추가되었습니다.
- 이제 로컬 또는 원격 **yaml** 파일을 사용하여 Pipeline Run을 직접 시작할 수 있습니다.
- 이제 **describe** 하위 명령이 이제 보강되고 상세한 출력을 표시합니다. **description, timeout, param description** 및 **sidecar status**와 같은 새로운 필드가 추가되면서 특정 **tkn** 리소스에 대한 자세한 정보가 명령 출력에 제공됩니다.
- 네임스페이스에 있는 작업이 한 개뿐인 경우 **tkn task log** 명령으로 바로 로그를 표시할 수 있습니다.

#### 6.4.1.3. Trigger

- Trigger가 이제 **v1alpha1** 및 **v1beta1** Pipeline 리소스를 모두 생성할 수 있습니다.
- 새로운 CEL(Common Expression Language) 인터셉터 기능 **-compareSecret** 지원 이 기능은 보안을 유지하면서 문자열을 CEL 표현식의 보안과 비교합니다.
- EventListener Trigger 수준에서 인증 및 승인을 지원합니다.

#### 6.4.2. 더 이상 사용되지 않는 기능

이 릴리스에서는 더 이상 사용되지 않는 기능은 다음과 같습니다.

- 단계(Step) 사양의 환경 변수 **\$HOME** 및 변수 **workingDir**은 더 이상 사용되지 않으며 향후 릴리스에서 변경될 수 있습니다. 현재 단계(Step) 컨테이너의 **HOME**과 **workingDir**이 **/tekton/home**과 **/workspace**을 각각 덮어씁니다.  
 향후 릴리스에서 이 두 필드는 수정되지 않으며, 컨테이너 이미지 및 Task YAML에 정의된 값으로 설정될 것입니다. 이번 릴리스에서는 **disable-home-env-override** 및 **disable-working-directory-override** 플래그를 사용하여 **HOME** 및 **workingDir** 변수의 덮어쓰기 기능을 비활성화하십시오.
- 다음은 더 이상 사용되지 않는 명령들이며 향후 릴리스에서 제거될 수 있습니다.
  - **tkn pipeline create**
  - **tkn task create**
- **tkn resource create** 명령과 함께 **-f** 플래그가 더 이상 사용되지 않습니다. 향후 릴리스에서 제거될 수 있습니다.

- **tkn clustertask create** 명령에서 **-t** 플래그와 **--timeout** 플래그(초 형식)가 더 이상 사용되지 않습니다. 이제 지속 시간 초과 형식만 지원됩니다(예: **1h30s**). 더 이상 사용되지 않는 이러한 플래그는 향후 릴리스에서 제거될 수 있습니다.

### 6.4.3. 확인된 문제

- 이전 버전의 Red Hat OpenShift Pipelines에서 업그레이드하는 경우 Red Hat OpenShift Pipelines 버전 1.0으로 업그레이드하기 전에 기존 배포를 삭제해야 합니다. 기존 배포를 삭제하려면 먼저 사용자 정의 리소스를 삭제한 다음 Red Hat OpenShift Pipelines Operator를 설치 제거해야 합니다. 자세한 내용은 Red Hat OpenShift Pipeline 설치 제거 섹션을 참조하십시오.
- 동일한 **v1alpha1** Task를 두 번 이상 제출하면 오류가 발생합니다. **v1alpha1** Task를 다시 제출할 때 **oc apply** 대신 **oc replace**를 사용하십시오.
- 컨테이너에 사용자가 새로 추가되면 **buildah** ClusterTask가 작동하지 않습니다. Operator가 설치되면 **buildah** ClusterTask에 대한 **--storage-driver** 플래그가 지정되지 않으므로 플래그가 기본값으로 설정됩니다. 스토리지 드라이버가 잘못 설정되는 경우도 발생할 수 있습니다. 사용자가 새로 추가되면 잘못된 스토리지 드라이버로 인해 다음 오류가 발생되면서 **buildah** ClusterTask가 실패합니다.

```
useradd: /etc/passwd.8: lock file already used
useradd: cannot lock /etc/passwd; try again later.
```

이 문제를 해결하려면 **buildah-task.yaml** 파일에서 **--storage-driver** 플래그 값을 **overlay**로 직접 설정하십시오.

1. **cluster-admin** 권한으로 클러스터에 로그인합니다.

```
$ oc login -u <login> -p <password> https://openshift.example.com:6443
```

2. **oc edit** 명령을 사용하여 **buildah** ClusterTask를 편집합니다.

```
$ oc edit clustertask buildah
```

**buildah** clustertask YAML 파일의 현재 버전이 **EDITOR** 환경 변수에 의해 설정된 편집기에서 열립니다.

3. **steps** 필드에서 다음 **command** 필드를 찾습니다.

```
command: ['buildah', 'bud', '--format=${(params.FORMAT)}', '--tls-verify=${(params.TLSVERIFY)}', '--layers', '-f', '${(params.DOCKERFILE)}', '-t', '${(resources.outputs.image.url)}', '${(params.CONTEXT)}']
```

4. **command** 필드를 다음으로 변경합니다.

```
command: ['buildah', '--storage-driver=overlay', 'bud', '--format=${(params.FORMAT)}', '--tls-verify=${(params.TLSVERIFY)}', '--no-cache', '-f', '${(params.DOCKERFILE)}', '-t', '${(params.IMAGE)}', '${(params.CONTEXT)}']
```

5. 파일을 저장하고 종료합니다.

또는 **Pipelines** → **Cluster Tasks** → **buildah**로 이동하여 웹 콘솔에서 직접 **buildah** ClusterTask YAML 파일을 수정할 수도 있습니다. **Actions** 메뉴에서 **Edit Cluster Task**를 선택하고 이전 프롬시저에서 안내한 대로 **command** 필드를 변경합니다.

#### 6.4.4. 해결된 문제

- 이전에는 이미지 빌드가 이미 진행 중인 경우에도 **DeploymentConfig** Task가 새 배포 빌드를 트리거했습니다. 이로 인해 Pipeline 배포가 실패로 끝납니다. 수정판에서는 진행 중인 배포를 마칠 때까지 대기하는 **oc rollout status** 명령으로 이제 **deploy task** 명령을 대체합니다.
- **APP\_NAME** 매개변수에 대한 지원이 이제 Pipeline 템플릿에 추가됩니다.
- 전에는 Java S2I용 Pipeline 템플릿이 레지스트리에서 이미지를 찾지 못했습니다. 수정판에서는 사용자가 제공한 **IMAGE\_NAME** 매개변수 대신 기존 이미지 PipelineResources를 사용하여 이미지를 검색합니다.
- 이제 모든 OpenShift Pipelines 이미지가 Red Hat UBI(Universal Base Images, 범용 기본 이미지)를 기반으로 합니다.
- 전에는 **tekton-pipelines** 이외 네임스페이스에 Pipeline을 설치했을 때 **tkn version** 명령에서 Pipeline 버전을 **unknown**으로 표시했습니다. 수정판에서는 **tkn version** 명령으로 이제 모든 네임스페이스에 올바른 Pipeline 버전을 표시할 수 있습니다.
- **tkn version** 명령에 더 이상 **-c** 플래그가 지원되지 않습니다.
- 관리자 권한이 없는 사용자도 이제 ClusterTriggerBindings 목록을 볼 수 있습니다.
- CEL 인터셉터에 대한 EventListener CompareSecret 기능이 수정되었습니다.
- Task와 ClusterTask의 이름이 같은 경우 **task** 및 **clustertask**에 대한 **list**, **describe** 및 **start** 하위 명령의 출력이 이제 올바르게 표시됩니다.
- 이전에는 OpenShift Pipelines Operator에서 권한이 필요한 SCC(보안 컨텍스트 제약 조건)를 수정하여 클러스터 업그레이드 도중 오류가 발생했습니다. 이 오류는 이제 수정되었습니다.
- **tekton-pipelines** 네임스페이스에서 모든 TaskRuns 및 PipelineRuns의 시간 초과 값이 이제 ConfigMap을 사용하여 **default-timeout-minutes** 필드 값으로 설정됩니다.
- 전에는 관리자 권한이 없는 사용자에게는 웹 콘솔의 Pipeline 섹션이 표시되지 않았습니다. 이 문제는 이제 해결되었습니다.