Red Hat OpenShift Pipelines 1.15

CI/CD 파이프라인 생성

OpenShift Pipelines에서 작업 및 파이프라인 생성 및 실행 시작하기

Last Updated: 2024-09-02

Red Hat OpenShift Pipelines 1.15 CI/CD 파이프라인 생성

OpenShift Pipelines에서 작업 및 파이프라인 생성 및 실행 시작하기

법적 공지

초록

이 문서에서는 OpenShift Pipelines에서 작업 및 파이프라인 생성 및 실행에 대한 정보를 제공합니다.

차례

1장. OPENSHIFT PIPELINES를 사용하여 애플리케이션용 CI/CD 솔루션 작성 3
1.1. 사전 요구 사항 3
1.2. 프로젝트 생성 및 파이프라인 서비스 계정 검사 3
1.3. 파이프라인 작업 생성 4
1.4. 파이프라인 조립 5
1.5. 제한된 환경에서 파이프라인을 실행하도록 이미지 미러링 7
1.6. 파이프라인 실행 11
1.7. 파이프라인에 트리거 추가 12
1.8. 여러 네임스페이스를 제공하도록 이벤트 리스너 구성 16
1.9. WEBHOOK 생성 18
1.10. 파이프라인 실행 트리거 19
1.11. 사용자 정의 프로젝트의 트리거에 대한 이벤트 리스너 모니터링 활성화 20
1.12. GITHUB INTERCEPTOR에서 가져오기 요청 기능 구성 21
1.13. 추가 리소스 25

2장. 웹 콘솔에서 RED HAT OPENSHIFT PIPELINES 작업 26
2.1. 개발자 관점에서 RED HAT OPENSHIFT PIPELINES 작업 26
2.2. 추가 리소스 44
2.3. 관리자 화면에서 파이프라인 템플릿 생성 44
2.4. 웹 콘솔의 파이프라인 실행 통계 45

3장. 해결자를 사용하여 원격 파이프라인 및 작업 지정 50
3.1. TEKTON 카탈로그에서 원격 파이프라인 또는 작업 지정 50
3.2. TEKTON 번들에서 원격 파이프라인 또는 작업 지정 55
3.3. GIT 리포지토리에서 원격 파이프라인 또는 작업 지정 58
3.4. 동일한 클러스터에서 원격 파이프라인 또는 작업 지정 64
3.5. OPENSHIFT PIPELINES 네임스페이스에서 제공되는 작업 67
3.6. 추가 리소스 102

4장. OPENSHIFT PIPELINES에서 수동 승인 사용 103
4.1. 수동 승인 게이트 컨트롤러 활성화 103
4.2. 수동 승인 작업 지정 104
4.3. 수동 승인 작업 승인 105

5장. 파이프라인에서 RED HAT 인타이틀먼트 사용 110
5.1. 사전 요구 사항 110
5.2. ETC-PKI-ENTITLEMENT 시크릿을 수동으로 복사하여 RED HAT 인타이틀먼트 사용 111
5.3. SHARED RESOURCES CSI 드라이버 OPERATOR를 사용하여 시크릿을 공유하여 RED HAT 인타이틀먼트 사용 113
5.4. 추가 리소스 116

6장. 버전이 지정되지 않은 클러스터 작업 관리 118
6.1. 버전이 아닌 클러스터 작업과 버전이 지정되지 않은 클러스터 작업의 차이점 118
6.2. 버전이 아닌 클러스터 작업 및 버전이 지정된 클러스터 작업의 이점 및 단점 118
6.3. 버전이 지정되지 않은 클러스터 작업 비활성화 119

1장. OPENSHIFT 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 수행합니다.

- 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.

- **pipelines-vote-ui** Git 리포지토리에 소스 코드가 있는 프런트 엔드 인터페이스 **pipelines-vote-ui**.
- **pipelines-vote-api** Git 리포지토리에 소스 코드가 있는 백엔드 인터페이스 **pipelines-vote-api**.
- **pipelines-tutorial** Git 리포지토리의 **apply-manifests** 및 **update-deployment** 작업입니다.

1.1. 사전 요구 사항

- OpenShift Container Platform 클러스터에 액세스 권한을 보유합니다.
- OpenShift OperatorHub에 나열된 Red Hat OpenShift Pipelines Operator를 사용하여 OpenShift Pipelines를 설치했습니다. 설치를 마친 후 전체 클러스터에 적용할 수 있습니다.
- OpenShift Pipelines CLI 를 설치했습니다.
- GitHub ID를 사용하여 프런트 엔드 **pipelines-vote-ui** 및 백엔드 **pipelines-vote-api** Git 리포지토리를 분기했으며 이러한 리포지토리에 대한 관리자 액세스 권한이 있습니다.
- 선택 사항: **pipelines-tutorial** Git 리포지토리를 복제했습니다.

1.2. 프로젝트 생성 및 파이프라인 서비스 계정 검사

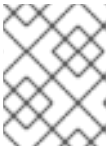
프로세스

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** 서비스 계정을 표시합니다.
Red Hat OpenShift Pipelines Operator는 이미지를 빌드하고 내보내기에 충분한 권한이 있는 **pipeline**이라는 서비스 계정을 추가하고 구성합니다. 이 서비스 계정은 **PipelineRun** 오브젝트에서 사용합니다.

```
$ oc get serviceaccount pipeline
```

1.3. 파이프라인 작업 생성

프로세스

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

```
$ oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/pipelines-1.15/01_pipeline/01_apply_manifest_task.yaml
$ oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/pipelines-1.15/01_pipeline/02_update_deployment_task.yaml
```

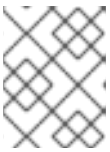
2. **tkn task list** 명령을 사용하여 생성한 작업 목록을 표시합니다.

```
$ tkn task list
```

apply-manifest 및 **update-deployment** 작업 리소스가 생성된 것이 출력에서 확인됩니다.

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

3. **tkn clustertasks list** 명령을 사용하여 Operator에서 설치한 추가 클러스터 작업 목록을 표시합니다(예: **buildah** 및 **s2i-python-3**).



참고

제한된 환경에서 **buildah** 클러스터 작업을 사용하려면 Dockerfile에서 내부 이미지 스트림을 기본 이미지로 사용해야 합니다.

```
$ tkn clustertasks list
```

Operator에서 설치한 **ClusterTask** 리소스가 출력에 나열됩니다.

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

중요

Red Hat OpenShift Pipelines 1.10에서 **ClusterTask** 기능은 더 이상 사용되지 않으며 향후 릴리스에서 제거될 예정입니다.

추가 리소스

- 버전이 지정되지 않은 클러스터 작업 관리

1.4. 파이프라인 조립

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

파이프라인은 **from** 및 **runAfter** 매개변수를 사용하여 작업들이 상호 작용하는 방법과 실행 순서를 지정합니다. 그리고 **workspaces** 필드를 사용하여 파이프라인의 각 작업 실행 중 필요한 하나 이상의 볼륨을 지정합니다.

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

파이프라인은 백엔드 애플리케이션 **pipelines-vote-api** 및 프런트 엔드 애플리케이션 **pipelines-vote-ui**에 대해 다음 작업을 수행합니다.

- **git-url** 및 **git-revision** 매개변수를 참조하여 Git 리포지토리에서 애플리케이션의 소스 코드를 복제합니다.
- **openshift-pipelines** 네임스페이스에 제공된 **buildah** 작업을 사용하여 컨테이너 이미지를 빌드합니다.
- image 매개변수를 참조하여 OpenShift 이미지 레지스트리에 **이미지를** 푸시합니다.
- **apply-manifests** 및 **update-deployment** 작업을 사용하여 OpenShift Container Platform에 새 이미지를 배포합니다.

프로세스

1. 다음 샘플 파이프라인 YAML 파일의 내용을 복사하여 저장합니다.

```
apiVersion: tekton.dev/v1
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: "pipelines-1.15"
  - name: IMAGE
    type: string
    description: image to be built from the code
  tasks:
  - name: fetch-repository
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: git-clone
      - name: namespace
        value: openshift-pipelines
    workspaces:
    - name: output
      workspace: shared-workspace
    params:
    - name: URL
      value: $(params.git-url)
    - name: SUBDIRECTORY
      value: ""
    - name: DELETE_EXISTING
      value: "true"
    - name: REVISION
      value: $(params.git-revision)
  - name: build-image
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: buildah
      - name: namespace
        value: openshift-pipelines
    workspaces:
    - name: source
      workspace: shared-workspace
    params:
    - name: IMAGE
      value: $(params.IMAGE)
    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
    params:
    - name: deployment
      value: $(params.deployment-name)
    - name: IMAGE
      value: $(params.IMAGE)
    runAfter:
    - apply-manifests
```

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

2. 파이프라인을 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/pipelines-1.15/01_pipeline/04_pipeline.yaml
```

3. **tkn pipeline list** 명령을 사용하여 파이프라인이 애플리케이션에 추가되었는지 확인합니다.

```
$ tkn pipeline list
```

출력에서 **build-and-deploy** 파이프라인이 생성되었는지 확인합니다.

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

1.5. 제한된 환경에서 파이프라인을 실행하도록 이미지 미러링

연결이 끊긴 클러스터 또는 제한된 환경에서 프로비저닝된 클러스터에서 OpenShift Pipelines를 실행하려면 Samples Operator가 제한된 네트워크용으로 구성되었는지 또는 클러스터 관리자가 미러링된 레지스트리가 있는 클러스터를 생성했는지 확인해야 합니다.

다음 절차에서는 **pipelines-tutorial** 예제를 사용하여 미러링된 레지스트리가 있는 클러스터를 사용하여 제한된 환경에서 애플리케이션에 대한 파이프라인을 생성합니다. **pipelines-tutorial** 예제가 제한된 환경에서 작동하도록 하려면 프런트 엔드 인터페이스 **pipelines-vote-ui**, 백엔드 인터페이스 **pipelines-vote-api**, **cli**의 미러 레지스트리에서 해당 빌더 이미지를 미러링해야 합니다.

프로세스

1. 프런트 엔드 인터페이스 **pipelines-vote-ui**의 미러 레지스트리에서 빌더 이미지를 미러링합니다.

   a. 필요한 이미지 태그를 가져오지 않았는지 확인합니다.

```
$ oc describe imagestream python -n openshift
```

-

출력 예

```
Name:   python
Namespace:  openshift
[...]

3.8-ubi9 (latest)
  tagged from registry.redhat.io/ubi9/python-38:latest
    prefer registry pullthrough when referencing this tag

  Build and run Python 3.8 applications on UBI 8. For more information about using this
builder image, including OpenShift considerations, see https://github.com/sclorg/s2i-
python-container/blob/master/3.8/README.md.
  Tags: builder, python
  Supports: python:3.8, python
  Example Repo: https://github.com/sclorg/django-ex.git

[...]
```

b. 지원되는 이미지 태그를 프라이빗 레지스트리로 미러링합니다.

```
$ oc image mirror registry.redhat.io/ubi9/python-39:latest <mirror-registry>:
<port>/ubi9/python-39
```

c. 이미지를 가져옵니다.

```
$ oc tag <mirror-registry>:<port>/ubi9/python-39 python:latest --scheduled -n openshift
```

이미지는 정기적으로 다시 가져와야 합니다. **--scheduled** 플래그를 사용하면 자동으로 이미지를 다시 가져올 수 있습니다.

d. 지정된 태그가 있는 이미지를 가져왔는지 확인합니다.

```
$ oc describe imagestream python -n openshift
```

출력 예

```
Name:   python
Namespace:  openshift
[...]

latest
  updates automatically from registry  <mirror-registry>:<port>/ubi9/python-39

  *  <mirror-registry>:<port>/ubi9/python-39@sha256:3ee...

[...]
```

2. 백엔드 인터페이스 **pipelines-vote-api**의 미러 레지스트리에서 빌더 이미지를 미러링합니다.

a. 필요한 이미지 태그를 가져오지 않았는지 확인합니다.

```
$ oc describe imagestream golang -n openshift
```

출력 예

```
Name:   golang
Namespace:  openshift
[...]

1.14.7-ubi8 (latest)
  tagged from registry.redhat.io/ubi8/go-toolset:1.14.7
    prefer registry pullthrough when referencing this tag

  Build and run Go applications on UBI 8. For more information about using this builder
image, including OpenShift considerations, see https://github.com/sclorg/golang-
container/blob/master/README.md.
  Tags: builder, golang, go
  Supports: golang
  Example Repo: https://github.com/sclorg/golang-ex.git

[...]
```

b. 지원되는 이미지 태그를 프라이빗 레지스트리로 미러링합니다.

```
$ oc image mirror registry.redhat.io/ubi9/go-toolset:latest <mirror-registry>:
<port>/ubi9/go-toolset
```

c. 이미지를 가져옵니다.

```
$ oc tag <mirror-registry>:<port>/ubi9/go-toolset golang:latest --scheduled -n openshift
```

이미지는 정기적으로 다시 가져와야 합니다. **--scheduled** 플래그를 사용하면 자동으로 이미지를 다시 가져올 수 있습니다.

d. 지정된 태그가 있는 이미지를 가져왔는지 확인합니다.

```
$ oc describe imagestream golang -n openshift
```

출력 예

```
Name:   golang
Namespace:  openshift
[...]

latest
  updates automatically from registry <mirror-registry>:<port>/ubi9/go-toolset

  * <mirror-registry>:<port>/ubi9/go-
toolset@sha256:59a74d581df3a2bd63ab55f7ac106677694bf612a1fe9e7e3e1487f55c421
b37

[...]
```

3. **cli**의 미러 레지스트리에서 빌더 이미지를 미러링합니다.

a. 필요한 이미지 태그를 가져오지 않았는지 확인합니다.

```
$ oc describe imagestream cli -n openshift
```

출력 예

```
Name:                   cli
Namespace:              openshift
[...]

latest
  updates automatically from registry quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:65c68e8c22487375c4c6ce6f18ed5485915f2bf612e41fef6d41cbfcdb143551


  * quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:65c68e8c22487375c4c6ce6f18ed5485915f2bf612e41fef6d41cbfcdb143551


[...]
```

b. 지원되는 이미지 태그를 프라이빗 레지스트리로 미러링합니다.

```
$ oc image mirror quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:65c68e8c22487375c4c6ce6f18ed5485915f2bf612e41fef6d41cbfcdb143551 <mirror-registry>:<port>/openshift-release-dev/ocp-v4.0-art-dev:latest
```

c. 이미지를 가져옵니다.

```
$ oc tag <mirror-registry>:<port>/openshift-release-dev/ocp-v4.0-art-dev cli:latest --scheduled -n openshift
```

이미지는 정기적으로 다시 가져와야 합니다. **--scheduled** 플래그를 사용하면 자동으로 이미지를 다시 가져올 수 있습니다.

d. 지정된 태그가 있는 이미지를 가져왔는지 확인합니다.

```
$ oc describe imagestream cli -n openshift
```

출력 예

```
Name:                   cli
Namespace:              openshift
[...]

latest
  updates automatically from registry <mirror-registry>:<port>/openshift-release-dev/ocp-v4.0-art-dev

  * <mirror-registry>:<port>/openshift-release-dev/ocp-v4.0-art-dev@sha256:65c68e8c22487375c4c6ce6f18ed5485915f2bf612e41fef6d41cbfcdb143551


[...]
```

추가 리소스

- 제한된 클러스터에 대한 Samples Operator 구성
- 미러링된 레지스트리로 클러스터 생성

1.6. 파이프라인 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/pipelines-1.15/01_pipeline/03_persistent_volume_claim.yaml \
    -p deployment-name=pipelines-vote-api \
    -p git-url=https://github.com/openshift/pipelines-vote-api.git \
    -p IMAGE='image-registry.openshift-image-registry.svc:5000/pipelines-tutorial/pipelines-
vote-api' \
    --use-param-defaults
```

   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. 프런트 엔드 애플리케이션의 파이프라인을 시작합니다.

```
$ tkn pipeline start build-and-deploy \
    -w name=shared-
workspace,volumeClaimTemplateFile=https://raw.githubusercontent.com/openshift/pipelines-
tutorial/pipelines-1.15/01_pipeline/03_persistent_volume_claim.yaml \
    -p deployment-name=pipelines-vote-ui \
    -p git-url=https://github.com/openshift/pipelines-vote-ui.git \
    -p IMAGE='image-registry.openshift-image-registry.svc:5000/pipelines-tutorial/pipelines-
vote-ui' \
    --use-param-defaults
```

4. 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.

5. 몇 분 후에 **tkn pipelinerun list** 명령을 사용하여 모든 파이프라인 실행을 나열하여 파이프라인이 성공적으로 실행되었는지 확인합니다.

```
$ tkn pipelinerun list
```

파이프라인 실행 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
```

6. 애플리케이션 경로를 가져옵니다.

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

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

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

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

추가 리소스

- 보안을 사용하여 리포지터리로 파이프라인 인증

1.7. 파이프라인에 트리거 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

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

```
apiVersion: triggers.tekton.dev/v1beta1
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)
```

2. **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/pipelines-1.15/03_triggers/01_binding.yaml
```

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

```
apiVersion: triggers.tekton.dev/v1beta1
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: pipelines-1.15
  - name: git-repo-name
    description: The name of the deployment to be created / patched

  resourcetemplates:
  - apiVersion: tekton.dev/v1
    kind: PipelineRun
    metadata:
      generateName: build-deploy-$(tt.params.git-repo-name)-
    spec:
      taskRunTemplate:
        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/pipelines-1.15/03_triggers/02_template.yaml
```

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

```
apiVersion: triggers.tekton.dev/v1beta1
kind: Trigger
metadata:
  name: vote-trigger
spec:
  taskRunTemplate:
    serviceAccountName: pipeline
  bindings:
    - ref: vote-app
  template:
    ref: 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://raw.githubusercontent.com/openshift/pipelines-tutorial/pipelines-1.15/03_triggers/03_trigger.yaml
```

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

```
apiVersion: triggers.tekton.dev/v1beta1
kind: EventListener
metadata:
  name: vote-app
spec:
  taskRunTemplate:
    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/v1beta1
kind: EventListener
metadata:
  name: vote-app
spec:
  taskRunTemplate:
    serviceAccountName: pipeline
```

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

8. 다음 단계를 수행하여 **EventListener** 리소스를 생성합니다.

- 보안 HTTPS 연결을 사용하여 **EventListener** 리소스를 생성하려면 다음을 수행합니다.

a. **Eventlistener** 리소스에 대한 보안 HTTPS 연결을 활성화하려면 레이블을 추가합니다.

```
$ oc label namespace <ns-name> operator.tekton.dev/enable-annotation=enabled
```

b. **EventListener** 리소스를 생성합니다.

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

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

```
$ oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/pipelines-1.15/03_triggers/04_event_listener.yaml
```

c. 재암호화 TLS 종료로 경로를 생성합니다.

```
$ oc create route reencrypt --service=<svc-name> --cert=tls.crt --key=tls.key --ca-cert=ca.crt --hostname=<hostname>
```

또는 재암호화 TLS 종료 YAML 파일을 만들어 보안 경로를 만들 수도 있습니다.

보안 경로의 TLS 종료 YAML에 대한 재암호화의 예

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: route-passthrough-secured (1)
spec:
  host: <hostname>
  to:
    kind: Service
    name: frontend (2)
  tls:
    termination: reencrypt (3)
    key: [as in edge termination]
    certificate: [as in edge termination]
    caCertificate: [as in edge termination]
    destinationCACertificate: |- (4)
      -----BEGIN CERTIFICATE-----
      [...]
      -----END CERTIFICATE-----
```

(1) (2) 63자로 제한되는 개체의 이름입니다.

3 **termination** 필드는 **reencrypt**로 설정됩니다. 이 필드는 유일한 필수 **tls** 필드입니다.

4 재암호화에 필요합니다. **destinationCACertificate**는 엔드포인트 인증서의 유효성을 검사하고 라우터에서 대상 pod로의 연결을 보호합니다. 서비스에서 서비스 서명 인증서를 사용 중이거나 관리자가 라우터의 기본 CA 인증서를 지정하고 서비스에 해당 CA에서 서명한 인증서가 있는 경우 이 필드를 생략할 수 있습니다.

자세한 옵션은 **oc create route reencrypt --help**를 참조하십시오.

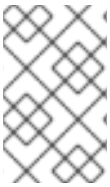
- 비보안 HTTP 연결을 사용하여 **EventListener** 리소스를 생성하려면 다음을 수행합니다.

  a. **EventListener** 리소스를 생성합니다.

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

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

1.8. 여러 네임스페이스를 제공하도록 이벤트 리스너 구성



참고

기본 CI/CD 파이프라인을 생성하려면 이 섹션을 건너뛸 수 있습니다. 그러나 배포 전략에 여러 네임스페이스가 포함된 경우 여러 네임스페이스를 제공하도록 이벤트 리스너를 구성할 수 있습니다.

클러스터 관리자는 **EvenListener** 오브젝트의 재사용 가능성을 높이기 위해 여러 네임스페이스를 제공하는 멀티 테넌트 이벤트 리스너로 구성하고 배포할 수 있습니다.

프로세스

1. 이벤트 리스너에 대한 클러스터 전체 가져오기 권한을 구성합니다.

   a. **ClusterRoleBinding** 및 **EventListener** 오브젝트에 사용할 서비스 계정 이름을 설정합니다. 예를 들면 **el-sa** 입니다.

   예시 ServiceAccount.yaml

   ```
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: el-sa
   ---
   ```

   b. **ClusterRole.yaml** 파일의 **규칙** 섹션에서 모든 이벤트 리스너 배포에 대한 적절한 권한을 클러스터 전체에서 작동하도록 설정합니다.

   ClusterRole.yaml예

   ```
   kind: ClusterRole
   apiVersion: rbac.authorization.k8s.io/v1
   metadata:
   ```

```
  name: el-sel-clusterrole
rules:
- apiGroups: ["triggers.tekton.dev"]
  resources: ["eventlisteners", "clustertriggerbindings", "clusterinterceptors",
"triggerbindings", "triggertemplates", "triggers"]
  verbs: ["get", "list", "watch"]
- apiGroups: [""]
  resources: ["configmaps", "secrets"]
  verbs: ["get", "list", "watch"]
- apiGroups: [""]
  resources: ["serviceaccounts"]
  verbs: ["impersonate"]
...
```

c. 적절한 서비스 계정 이름과 클러스터 역할 이름을 사용하여 클러스터 역할 바인딩을 구성합니다.

ClusterRoleBinding.yaml예

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: el-mul-clusterrolebinding
subjects:
- kind: ServiceAccount
  name: el-sa
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: el-sel-clusterrole
...
```

2. 이벤트 리스너의 **spec** 매개변수에 서비스 계정 이름(예: **el-sa**)을 추가합니다. 이벤트 리스너가 제공하려는 네임스페이스의 이름으로 **namespaceSelector** 매개변수를 작성합니다.

EventListener.yaml의 예

```
apiVersion: triggers.tekton.dev/v1beta1
kind: EventListener
metadata:
  name: namespace-selector-listener
spec:
  taskRunTemplate:
    serviceAccountName: el-sa
  namespaceSelector:
    matchNames:
    - default
    - foo
...
```

3. 필요한 권한(예: **foo-trigger-sa**)을 사용하여 서비스 계정을 생성합니다. 트리거를 역할 바인딩에 사용합니다.

예시 ServiceAccount.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: foo-trigger-sa
  namespace: foo
...
```

RoleBinding.yaml예

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: triggercr-rolebinding
  namespace: foo
subjects:
- kind: ServiceAccount
  name: foo-trigger-sa
  namespace: foo
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: tekton-triggers-eventlistener-roles
...
```

4. 적절한 트리거 템플릿, 트리거 바인딩 및 서비스 계정 이름을 사용하여 트리거를 생성합니다.

Trigger.yaml의 예

```
apiVersion: triggers.tekton.dev/v1beta1
kind: Trigger
metadata:
  name: trigger
  namespace: foo
spec:
  taskRunTemplate:
    serviceAccountName: foo-trigger-sa
  interceptors:
    - ref:
        name: "github"
      params:
        - name: "secretRef"
          value:
            secretName: github-secret
            secretKey: secretToken
        - name: "eventTypes"
          value: ["push"]
  bindings:
    - ref: vote-app
  template:
    ref: vote-app
...
```

1.9. WEBHOOK 생성

*Webhooks*는 리포지토리에 구성된 이벤트가 발생할 때마다 이벤트 리스너가 수신하는 HTTP POST 메시지입니다. 이어서 이벤트 페이로드가 트리거 바인딩에 매핑되고 트리거 템플릿에 의해 처리됩니다. 트리거 템플릿은 최종적으로 Kubernetes 리소스를 생성 및 배포를 수행할 하나 이상의 파이프라인 실행을 시작합니다.

여기서는 분기된 Git 리포지토리 **pipelines-vote-ui**와 **pipelines-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 Webhook을 추가하십시오.

프로세스

1. Webhook URL을 가져옵니다.
   - 보안 HTTPS 연결의 경우 다음을 수행합니다.

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

   - HTTP(비보안) 연결의 경우 다음을 수행합니다.

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

     출력에서 가져온 URL을 기록해 둡니다.
2. 프런트 엔드 리포지토리에서 수동으로 Webhook을 구성합니다.
   a. 브라우저에서 프런트 엔드 Git 리포지토리 **pipelines-vote-ui**를 엽니다.
   b. **Settings → Webhook → Webhook 추가**를 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 **pipelines-vote-api**에 대해 2단계를 반복합니다.

1.10. 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 **pipelines-vote-ui** 리포지토리로 내보냅니다. 그러면 파이프라인 실행이 트리거됩니다.

프로세스

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

```
$ git clone git@github.com:<your GitHub ID>/pipelines-vote-ui.git -b pipelines-1.15
```

2. 비어 있는 커밋을 푸시합니다.

```
$ git commit -m "empty-commit" --allow-empty && git push origin pipelines-1.15
```

3. 파이프라인 실행이 트리거되었는지 확인합니다.

```
$ tkn pipelinerun list
```

새로운 파이프라인 실행이 시작되었습니다.

1.11. 사용자 정의 프로젝트의 트리거에 대한 이벤트 리스너 모니터링 활성화

클러스터 관리자는 사용자 정의 프로젝트에서 **Triggers** 서비스에 대한 이벤트 리스너 메트릭을 수집하고 OpenShift Container Platform 웹 콘솔에 표시하려면 각 이벤트 리스너에 대한 서비스 모니터를 생성할 수 있습니다. HTTP 요청을 수신할 때 **Triggers** 서비스의 이벤트 리스너는 3개의 metrics Cryostat- **eventlistener_http_duration_seconds**,**eventlistener_event_count**, **eventlistener_triggered_resources** 를 반환합니다.

사전 요구 사항

- OpenShift Container Platform 웹 콘솔에 로그인했습니다.
- Red Hat OpenShift Pipelines Operator를 설치했습니다.
- 사용자 정의 프로젝트에 대한 모니터링을 활성화했습니다.

프로세스

1. 각 이벤트 리스너에 대해 서비스 모니터를 생성합니다. 예를 들어 **테스트** 네임스페이스에서 **github-listener** 이벤트 리스너에 대한 메트릭을 보려면 다음 서비스 모니터를 생성합니다.

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/managed-by: EventListener
    app.kubernetes.io/part-of: Triggers
    eventlistener: github-listener
  annotations:
    networkoperator.openshift.io/ignore-errors: ""
  name: el-monitor
  namespace: test
spec:
  endpoints:
    - interval: 10s
```

```
    port: http-metrics
  jobLabel: name
  namespaceSelector:
    matchNames:
      - test
  selector:
    matchLabels:
      app.kubernetes.io/managed-by: EventListener
      app.kubernetes.io/part-of: Triggers
      eventlistener: github-listener
...
```

2. 이벤트 리스너로 요청을 전송하여 서비스 모니터를 테스트합니다. 예를 들어 빈 커밋을 푸시합니다.

```
$ git commit -m "empty-commit" --allow-empty && git push origin main
```

3. OpenShift Container Platform 웹 콘솔에서 **관리자 → 모니터링 → 메트릭** 으로 이동합니다.

4. 지표를 보려면 이름으로 검색합니다. 예를 들어 **github-listener** 이벤트 리스너에 대한 **eventlistener_http_resources** 지표의 세부 정보를 보려면 **eventlistener_http_resources** 키워드를 사용하여 검색합니다.

추가 리소스

- 사용자 정의 프로젝트 모니터링 활성화

1.12. GITHUB INTERCEPTOR에서 가져오기 요청 기능 구성

GitHub Interceptor를 사용하면 GitHub Webhook의 유효성을 검사하고 필터링하는 논리를 생성할 수 있습니다. 예를 들어 Webhook의 출처를 확인하고 지정된 기준에 따라 들어오는 이벤트를 필터링할 수 있습니다. GitHub Interceptor를 사용하여 이벤트 데이터를 필터링하면 인터셉터가 필드에서 수락할 수 있는 이벤트 유형을 지정할 수 있습니다. Red Hat OpenShift Pipelines에서는 GitHub 인터셉터의 다음과 같은 기능을 사용할 수 있습니다.

- 변경된 파일을 기반으로 가져오기 요청 이벤트를 필터링
- 구성된 GitHub 소유자를 기반으로 가져오기 요청 검증

1.12.1. GitHub Interceptor를 사용하여 가져오기 요청 필터링

푸시 및 가져오기 이벤트를 위해 변경된 파일을 기반으로 GitHub 이벤트를 필터링할 수 있습니다. 이를 통해 Git 리포지토리의 관련 변경 사항만 파이프라인을 실행할 수 있습니다. GitHub Interceptor는 변경된 모든 파일의 쉼표로 구분된 목록을 추가하고 CEL 인터셉터를 사용하여 변경된 파일을 기반으로 들어오는 이벤트를 필터링합니다. 변경된 파일 목록은 최상위 **extensions** 필드에서 이벤트 페이로드의 **changed_files** 속성에 추가됩니다.

사전 요구 사항

- Red Hat OpenShift Pipelines Operator를 설치했습니다.

프로세스

1. 다음 중 하나를 실행합니다.

- 공개 GitHub 리포지토리의 경우 아래 YAML 구성 파일에서 **addChangedFiles** 매개변수 값을 **true** 로 설정합니다.

```
apiVersion: triggers.tekton.dev/v1beta1
kind: EventListener
metadata:
  name: github-add-changed-files-pr-listener
spec:
  triggers:
    - name: github-listener
      interceptors:
        - ref:
            name: "github"
            kind: ClusterInterceptor
            apiVersion: triggers.tekton.dev
          params:
          - name: "secretRef"
            value:
              secretName: github-secret
              secretKey: secretToken
          - name: "eventTypes"
            value: ["pull_request", "push"]
          - name: "addChangedFiles"
            value:
              enabled: true
        - ref:
            name: cel
          params:
          - name: filter
            value: extensions.changed_files.matches('controllers/')
...
```

- 프라이빗 GitHub 리포지토리의 경우 **addChangedFiles** 매개변수 값을 **true** 로 설정하고 아래에 표시된 YAML 구성 파일에 액세스 토큰 세부 정보 **secretName** 및 **secretKey** 를 제공합니다.

```
apiVersion: triggers.tekton.dev/v1beta1
kind: EventListener
metadata:
  name: github-add-changed-files-pr-listener
spec:
  triggers:
    - name: github-listener
      interceptors:
        - ref:
            name: "github"
            kind: ClusterInterceptor
            apiVersion: triggers.tekton.dev
          params:
          - name: "secretRef"
            value:
              secretName: github-secret
              secretKey: secretToken
          - name: "eventTypes"
            value: ["pull_request", "push"]
          - name: "addChangedFiles"
```

```
          value:
            enabled: true
            personalAccessToken:
              secretName: github-pat
              secretKey: token
        - ref:
            name: cel
          params:
          - name: filter
            value: extensions.changed_files.matches('controllers/')
...
```

2. 구성 파일을 저장합니다.

1.12.2. GitHub Interceptors를 사용하여 가져오기 요청 검증

GitHub Interceptor를 사용하여 리포지토리에 구성된 GitHub 소유자를 기반으로 가져오기 요청 처리를 검증할 수 있습니다. 이 검증은 **PipelineRun** 또는 **TaskRun** 오브젝트의 불필요한 실행을 방지하는 데 도움이 됩니다. GitHub 인터셉터는 사용자 이름이 소유자로 나열되거나 리포지토리 소유자가 구성 가능한 주석이 발행된 경우에만 가져오기 요청을 처리합니다. 예를 들어 가져오기 요청에서 소유자로 **/ok-to-test** 를 주석 처리하면 **PipelineRun** 또는 **TaskRun** 이 트리거됩니다.

참고

소유자는 리포지토리 루트의 **OWNERS** 파일에 구성됩니다.

사전 요구 사항

- Red Hat OpenShift Pipelines Operator를 설치했습니다.

프로세스

1. 시크릿 문자열 값을 생성합니다.
2. 해당 값을 사용하여 GitHub Webhook를 구성합니다.
3. 시크릿 값이 포함된 **secretRef** 라는 Kubernetes 시크릿을 생성합니다.
4. GitHub 인터셉터에 대한 참조로 Kubernetes 시크릿을 전달합니다.
5. **소유자** 파일을 만들고 승인자 섹션에 승인자 목록을 추가합니다.
6. **다음 중 하나를 실행합니다.**

- **공개 GitHub 리포지토리의 경우 아래 YAML 구성 파일에서 githubOwners 매개변수 값을 true 로 설정합니다.**

```
apiVersion: triggers.tekton.dev/v1beta1
kind: EventListener
metadata:
```

```
  name: github-owners-listener
spec:
  triggers:
    - name: github-listener
      interceptors:
        - ref:
            name: "github"
            kind: ClusterInterceptor
            apiVersion: triggers.tekton.dev
          params:
            - name: "secretRef"
              value:
                secretName: github-secret
                secretKey: secretToken
            - name: "eventTypes"
              value: ["pull_request", "issue_comment"]
            - name: "githubOwners"
              value:
                enabled: true
                checkType: none
...
```

- 프라이빗 GitHub 리포지토리의 경우 githubOwners 매개변수 값을 true 로 설정하고 아래에 표시된 YAML 구성 파일에 액세스 토큰 세부 정보 secretName 및 secretKey 를 제공합니다.

```
apiVersion: triggers.tekton.dev/v1beta1
kind: EventListener
metadata:
  name: github-owners-listener
spec:
  triggers:
    - name: github-listener
      interceptors:
        - ref:
            name: "github"
            kind: ClusterInterceptor
            apiVersion: triggers.tekton.dev
          params:
            - name: "secretRef"
              value:
                secretName: github-secret
                secretKey: secretToken
            - name: "eventTypes"
              value: ["pull_request", "issue_comment"]
            - name: "githubOwners"
              value:
                enabled: true
                personalAccessToken:
                  secretName: github-token
                  secretKey: secretToken
                checkType: all
...
```

참고

checkType 매개변수는 인증이 필요한 **GitHub** 소유자를 지정하는 데 사용됩니다. 해당 값을 **orgMembers,repoMembers** 또는 **all** 로 설정할 수 있습니다.

7. 구성 파일을 저장합니다.

1.13. 추가 리소스

- 동일한 리포지토리에 애플리케이션 소스 코드와 함께 **Pipeline**을 코드로 포함하려면 파이프라인 정보를 코드로 참조하십시오.
- 개발자 화면의 파이프라인에 대한 자세한 내용은 웹 콘솔의 **OpenShift Pipelines** 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.
- 재사용 가능한 작업 및 파이프라인을 위해 **Tekton Hub**의 사용자 정의 인스턴스를 설치하고 배포하려면 **Red Hat OpenShift Pipelines**에서 **Tekton Hub** 사용을 참조하십시오.
- 재암호화 **TLS** 종료에 대한 자세한 내용은 재암호화 종료를 참조하십시오.
- 보안 경로에 대한 자세한 내용은 보안 경로 섹션을 참조하십시오.

2장. 웹 콘솔에서 RED HAT OPENSHIFT PIPELINES 작업

관리자 또는 개발자 화면을 사용하여 OpenShift Container Platform 웹 콘솔의 Pipelines 페이지에서 Pipeline,PipelineRun, Repository 오브젝트를 생성하고 수정할 수 있습니다. 웹 콘솔의 개발자 화면에서 +추가 페이지를 사용하여 소프트웨어 제공 프로세스를 위한 CI/CD 파이프라인을 생성할 수도 있습니다.

2.1. 개발자 관점에서 RED HAT OPENSHIFT PIPELINES 작업

개발자 화면에서 +추가 페이지에서 파이프라인을 생성하기 위한 다음 옵션에 액세스할 수 있습니다.

- +추가 → 파이프라인 → 파이프라인 빌더 옵션을 사용하여 애플리케이션에 사용자 지정된 파이프라인을 생성합니다.
- 애플리케이션을 생성하는 동안 파이프라인 템플릿 및 리소스를 사용하여 파이프라인을 생성하려면 +추가 → Git에서 옵션을 사용합니다.

애플리케이션의 파이프라인을 생성한 후 Pipelines 보기에서 배포된 파이프라인을 보면서 시각적으로 상호 작용할 수 있습니다. Topology 보기에서도 From Git 옵션을 사용하여 생성된 파이프라인과 상호 작용할 수 있습니다. 토폴로지 보기에서 이를 확인하려면 파이프라인 빌더 를 사용하여 생성된 파이프라인에 사용자 정의 레이블을 적용해야 합니다.

사전 요구 사항

- OpenShift Container Platform 클러스터에 액세스하고 개발자 화면으로 전환했습니다.
- OpenShift Pipelines Operator가 클러스터에 설치되어 있어야 합니다.
- 클러스터 관리자 또는 작성 및 편집 권한이 있는 사용자입니다.
- 프로젝트를 생성했습니다.

2.1.1. 파이프라인 빌더를 사용하여 파이프라인 구성

콘솔의 개발자 화면에서 +추가 → 파이프라인 → 파이프라인 빌더 옵션을 사용하여 다음을 수행할 수 있습니다.

- 파이프라인 빌더 또는 **YAML** 보기를 사용하여 파이프라인을 구성합니다.

- 기존 작업 및 클러스터 작업을 사용하여 파이프라인 흐름을 구성합니다. **OpenShift Pipelines Operator**를 설치하면 재사용 가능한 파이프라인 클러스터 작업이 클러스터에 추가됩니다.

중요

Red Hat OpenShift Pipelines 1.10에서 **ClusterTask** 기능은 더 이상 사용되지 않으며 향후 릴리스에서 제거될 예정입니다.

- 파이프라인 실행에 필요한 리소스 유형을 지정하고, 필요한 경우 파이프라인에 매개변수를 추가합니다.

- 파이프라인의 각 작업에서 이러한 파이프라인 리소스를 입력 및 출력 리소스로 참조합니다.

- 필요한 경우 작업의 파이프라인에 추가된 매개변수를 참조합니다. 작업 매개변수는 작업 사양에 따라 미리 채워집니다.

- **Operator**에서 설치한 재사용 가능 조각과 샘플을 사용하여 세부 파이프라인을 생성합니다.

- 구성된 로컬 **Tekton Hub** 인스턴스에서 작업을 검색하고 추가합니다.

중요

개발자 화면에서 고유한 큐레이션 작업 세트를 사용하여 사용자 지정 파이프라인을 생성할 수 있습니다. 개발자 콘솔에서 직접 작업을 검색, 설치 및 업그레이드하려면 클러스터 관리자가 로컬 **Tekton Hub** 인스턴스를 설치 및 배포하고 해당 허브를 **OpenShift Container Platform** 클러스터에 연결해야 합니다. 자세한 내용은 *추가 리소스 섹션에서 **Tekton Hub with OpenShift Pipelines*** 사용을 참조하십시오. 기본적으로 로컬 **Tekton Hub** 인스턴스를 배포하지 않으면 클러스터 작업, 네임스페이스 작업 및 공용 **Tekton Hub** 작업에만 액세스할 수 있습니다.

프로세스

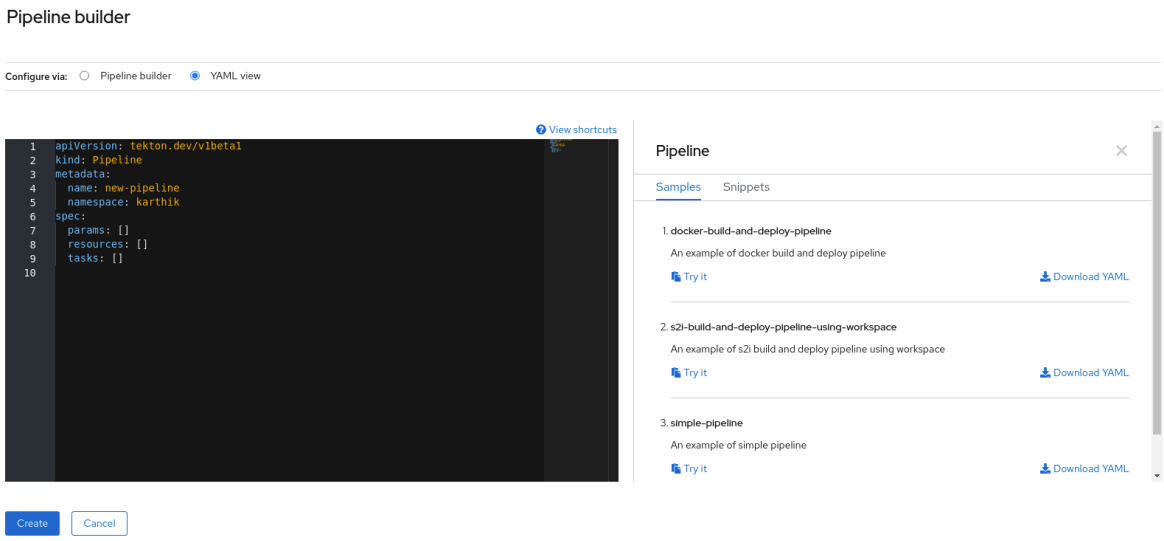
1. 개발자 화면의 +추가 보기에서 파이프라인 타일을 클릭하여 파이프라인 빌더 페이지를 표시합니다.

2. 파이프라인 빌더 보기 또는 **YAML** 보기를 사용하여 파이프라인을 구성합니다.

참고

파이프라인 빌더 보기에서는 제한된 수의 필드를 지원하는 반면 **YAML** 보기는 사용 가능한 모든 필드를 지원합니다. 필요한 경우 **Operator**가 설치한 재사용 가능한 스니펫과 샘플을 사용하여 자세한 파이프라인을 생성할 수도 있습니다.

그림 **2.1. YAML**보기



3. 파이프라인 빌더를 사용하여 파이프라인 을 구성합니다.

   a. 이름 필드에 파이프라인의 고유 이름을 입력합니다.

   b. 작업 섹션에서 다음을 수행합니다.

      i. 작업 추가를 클릭합니다.

      ii. 빠른 검색 필드를 사용하여 작업을 검색하고 표시된 목록에서 필요한 작업을 선택합니다.

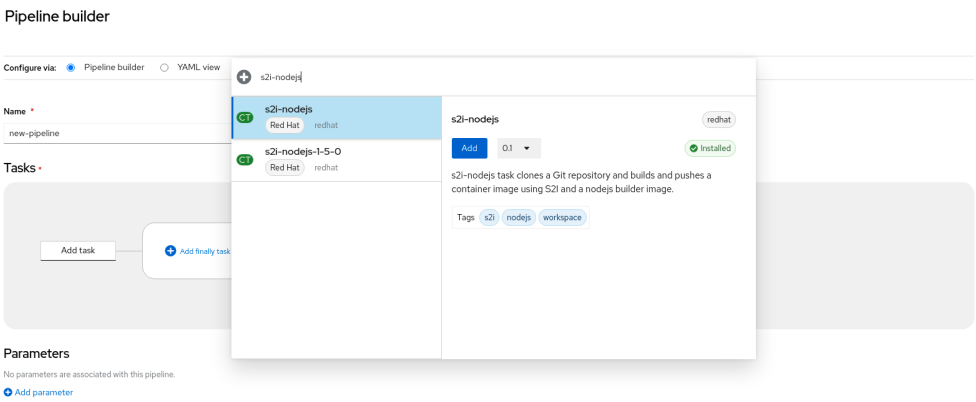
iii. 추가 또는 설치를 클릭하고 추가합니다. 이 예제에서는 **s2i-nodejs** 작업을 사용합니다.

참고

검색 목록에는 클러스터에서 사용할 수 있는 모든 **Tekton Hub** 작업과 작업이 포함되어 있습니다. 또한 작업이 이미 설치된 경우 **Add to add the task whereas it will show Install and add to install and add the task**가 표시됩니다. 업데이트된 버전으로 동일한 작업을 추가할 때 **Update** 및 **add**가 표시됩니다.

- 파이프라인에 순차적 작업을 추가하려면 다음을 수행합니다.
  - 작업 오른쪽 또는 왼쪽에 있는 더하기 아이콘을 클릭합니다. → 작업 추가 를 클릭합니다.
  - 빠른 검색 필드를 사용하여 작업을 검색하고 표시된 목록에서 필요한 작업을 선택합니다.
  - 추가 또는 설치를 클릭하고 추가합니다.

그림 2.2. 파이프라인 빌더



- 최종 작업을 추가하려면 다음을 수행합니다.
  - 마지막으로 추가 작업 → 작업 추가 를 클릭합니다.

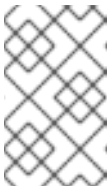
빠른 검색 필드를 사용하여 작업을 검색하고 표시된 목록에서 필요한 작업을 선택합니다.

- 추가 또는 설치를 클릭하고 추가합니다.

c. 리소스 섹션에서 리소스 추가를 클릭하여 파이프라인 실행에 사용할 리소스의 이름 및 유형을 지정합니다. 그러면 파이프라인의 작업에서 이러한 리소스를 입력 및 출력으로 사용합니다. 예시의 경우:

i. 입력 리소스를 추가합니다. 이름 필드에 **Source**를 입력하고 리소스 유형 드롭다운 목록에서 **Git**을 선택합니다.

ii. 출력 리소스를 추가합니다. 이름 필드에 **Img**를 입력하고 리소스 유형 드롭다운 목록에서 이미지를 선택합니다.



참고

리소스가 누락된 경우 작업 옆에 빨간색 아이콘이 표시됩니다.

d. 선택 사항: 작업의 매개변수는 작업 사양에 따라 미리 채워집니다. 필요에 따라 매개 변수 섹션에 있는 매개 변수 추가 링크를 사용하여 매개변수를 더 추가합니다.

e. 작업 공간 섹션에서 작업 공간 추가를 클릭하고 이름 필드에 고유한 작업 공간 이름을 입력합니다. 파이프라인에 여러 개의 작업 공간을 추가할 수 있습니다.

f. 작업 섹션에서 **s2i-nodejs** 작업을 클릭하여 작업 세부 정보가 있는 측면 패널을 확인합니다. 작업 측면 패널에서 **s2i-nodejs** 작업에 대한 리소스 및 매개변수를 지정합니다.

i. 필요에 따라 매개 변수 섹션에서 **$(params.<param-name>)** 구문을 사용하여 기본 매개변수에 매개변수를 더 추가합니다.

ii. 이미지 섹션에서 리소스 섹션에 지정된 대로 **Img**를 입력합니다.

:::

iii. 작업 공간 섹션의 소스 드롭다운에서 작업 공간을 선택합니다.

g. 리소스, 매개 변수 및 작업 공간을 **openshift-client** 작업에 추가합니다.

4. 생성을 클릭하여 파이프라인 세부 정보 페이지에서 파이프라인을 생성하고 봅니다.

5. 작업 드롭다운 메뉴를 클릭한 다음 시작을 클릭하여 파이프라인 시작 페이지를 확인합니다.

6. 작업 공간 섹션에는 이전에 생성한 작업 공간이 나열됩니다. 각 드롭다운을 사용하여 작업 공간의 볼륨 소스를 지정합니다. 빈 디렉토리,구성 맵,시크릿, 영구 볼륨 클레임, 볼륨 클레임 템플릿 옵션이 있습니다.

2.1.2. 애플리케이션과 함께 OpenShift Pipelines 생성

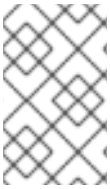
애플리케이션과 함께 파이프라인을 생성하려면 개발자 화면의 **Add+** 보기에서 **From Git** 옵션을 사용합니다. 사용 가능한 모든 파이프라인을 보고 코드를 가져오거나 이미지를 배포하는 동안 애플리케이션을 생성하는 데 사용할 파이프라인을 선택할 수 있습니다.

Tekton Hub 통합은 기본적으로 활성화되어 있으며 **Tekton Hub**에서 클러스터에서 지원하는 작업을 확인할 수 있습니다. 관리자는 **Tekton Hub** 통합을 비활성화할 수 있으며 **Tekton Hub** 작업은 더 이상 표시되지 않습니다. 생성된 파이프라인에 대한 **Webhook URL**이 있는지 확인할 수도 있습니다. **+**추가 흐름을 사용하여 생성된 파이프라인에 기본 **Webhook**가 추가되고 토폴로지 보기에서 선택한 리소스의 측면 패널에 **URL**이 표시됩니다.

자세한 내용은 개발자 화면을 사용하여 애플리케이션 생성을 참조하십시오.

2.1.3. 파이프라인을 포함하는 GitHub 리포지토리 추가

개발자 화면에서 파이프라인이 포함된 **GitHub** 리포지토리를 **OpenShift Container Platform** 클러스터에 추가할 수 있습니다. 이를 통해 내보내기 또는 가져오기 요청과 같은 관련 **Git** 이벤트가 트리거되면 클러스터의 **GitHub** 리포지토리에서 파이프라인 및 작업을 실행할 수 있습니다.



참고

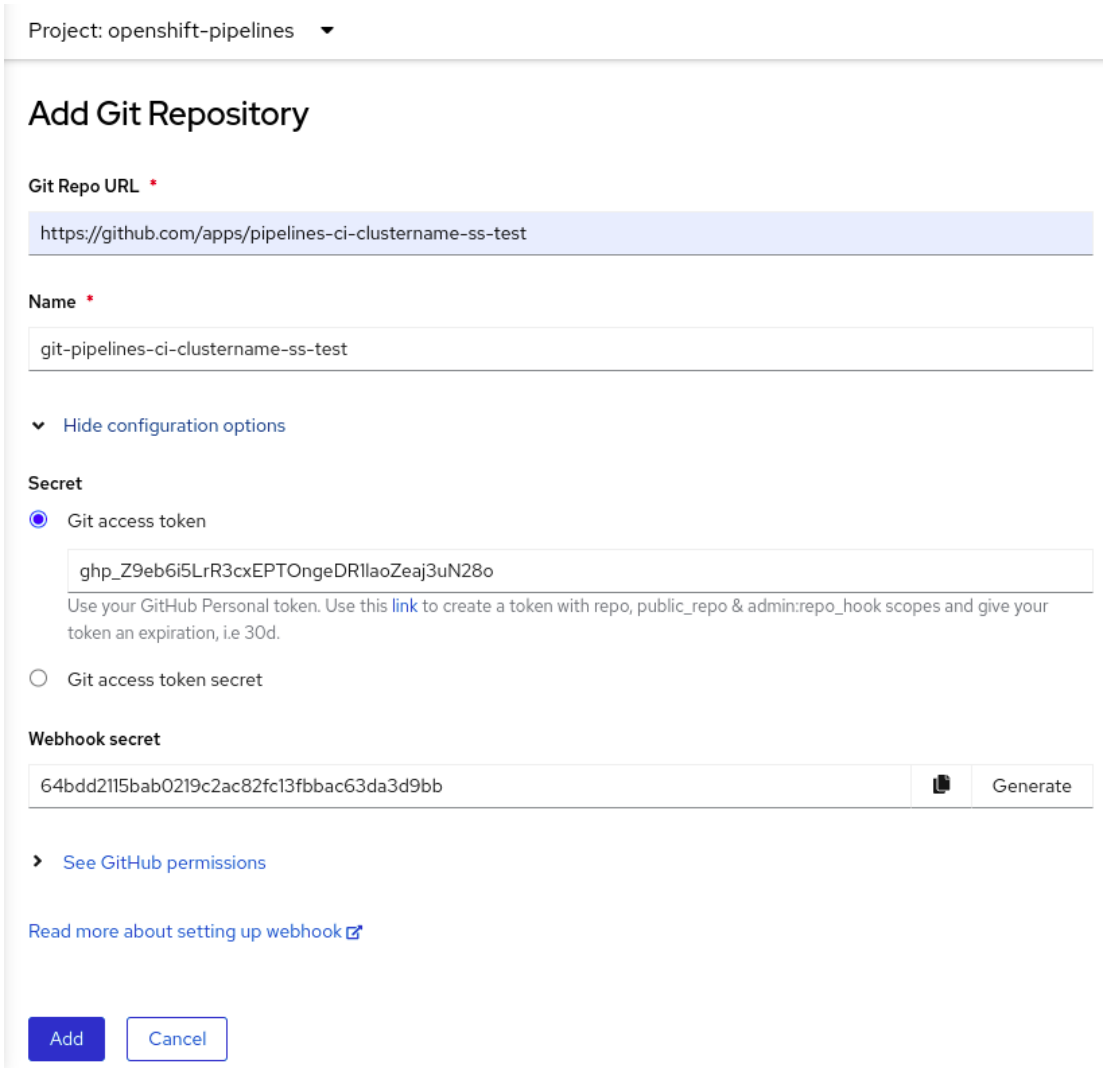
공용 및 개인 **GitHub** 리포지토리를 모두 추가할 수 있습니다.

사전 요구 사항

- 클러스터 관리자가 관리자 화면에서 필요한 **GitHub** 애플리케이션을 구성했는지 확인합니다.

프로세스

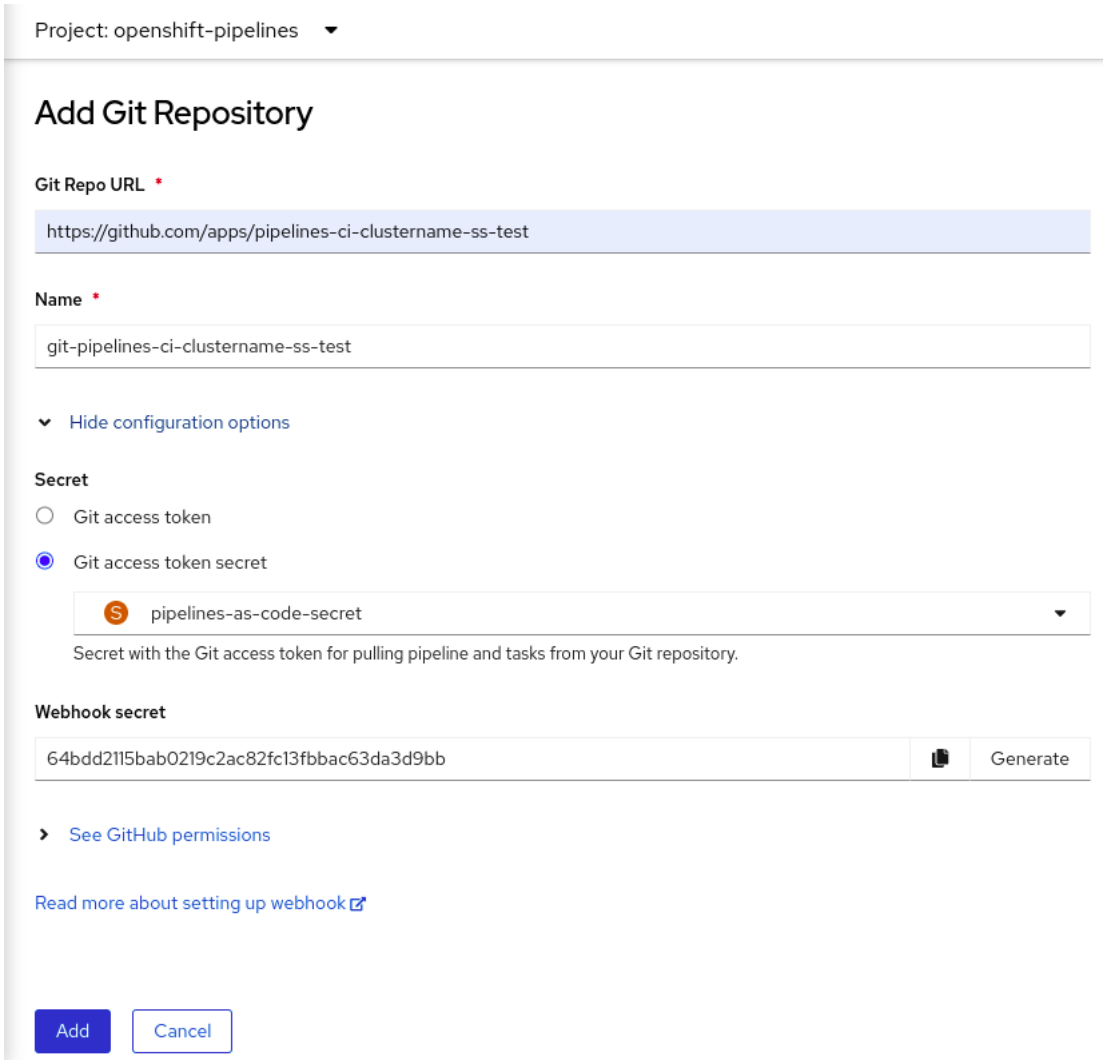
1. 개발자 화면에서 **GitHub** 리포지토리를 추가할 네임스페이스 또는 프로젝트를 선택합니다.
2. 왼쪽 탐색 창을 사용하여 **Pipeline** 으로 이동합니다.
3. **Pipelines** 페이지 오른쪽에 있는 생성 → 리포지토리 를 클릭합니다.
4. **Git Repo URL** 을 입력하면 콘솔에서 리포지터리 이름을 자동으로 가져옵니다.
5. 구성 옵션 표시를 클릭합니다. 기본적으로 **Webhook** 를 설정하는 방법은 한 가지뿐입니다. **GitHub** 애플리케이션이 구성된 경우 다음 두 가지 옵션이 표시됩니다.
   - **GitHub App:** 리포지토리에 **GitHub** 애플리케이션을 설치하려면 이 옵션을 선택합니다.
   - **Webhook** 설정**: GitHub** 애플리케이션에 **Webhook**를 추가하려면 이 옵션을 선택합니다.
6. 시크릿 섹션에서 다음 옵션 중 하나를 사용하여 **Webhook**를 설정합니다.
   - **Git** 액세스 토큰 을 사용하여 **Webhook**를 설정합니다.
     a. 개인 액세스 토큰을 입력합니다.
     b. **Webhook** 시크릿 필드에 해당하는 생성 을 클릭하여 새 **Webhook** 시크릿을 생성합니다.



참고

개인 액세스 토큰이 없고 새 액세스 토큰을 생성하려는 경우 **Git** 액세스 토큰 필드 아래에 링크를 클릭하면 됩니다.

- **Git** 액세스 토큰 시크릿 을 사용하여 **Webhook**를 설정합니다.
  - 드롭다운 목록에서 네임스페이스의 시크릿을 선택합니다. 선택한 보안에 따라 **Webhook** 보안이 자동으로 생성됩니다.



7. **GitHub 리포지토리에 Webhook 시크릿 세부 정보를 추가합니다.**

   a. **Webhook URL 을 복사하고 GitHub 리포지토리 설정으로 이동합니다.**

   b. **Webhook → Webhook 추가 를 클릭합니다.**

   c. **개발자 콘솔에서 Webhook URL 을 복사하여 GitHub 리포지토리 설정의 페이로드 URL 필드에 붙여넣습니다.**

   d. **콘텐츠 유형을 선택합니다.**

   e. **개발자 콘솔에서 Webhook 시크릿을 복사하여 GitHub 리포지토리 설정의 Secret 필드에 붙여넣습니다.**

f. **SSL 확인 옵션 중 하나를 선택합니다.**

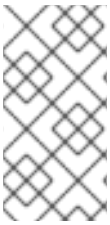
g. **이 Webhook를 트리거할 이벤트를 선택합니다.**

h. **Webhook 추가를 클릭합니다.**

8. **개발자 콘솔로 돌아가서 추가 를 클릭합니다.**

9. **수행해야 하는 단계의 세부 정보를 읽고 닫기 를 클릭합니다.**

10. **방금 생성한 리포지토리의 세부 정보를 확인합니다.**



참고

Git에서 가져오기를 사용하여 애플리케이션을 가져올 때 Git 리포지토리에 .tekton 디렉터리가 있는 경우 애플리케이션에 대한 pipelines-as-code 를 구성할 수 있습니다.

2.1.4. 개발자 화면을 사용하여 파이프라인과 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

프로세스

1. 개발자 화면의 파이프라인 보기에서 프로젝트 드롭다운 목록에 있는 프로젝트를 선택하여 해당 프로젝트의 파이프라인을 확인합니다.

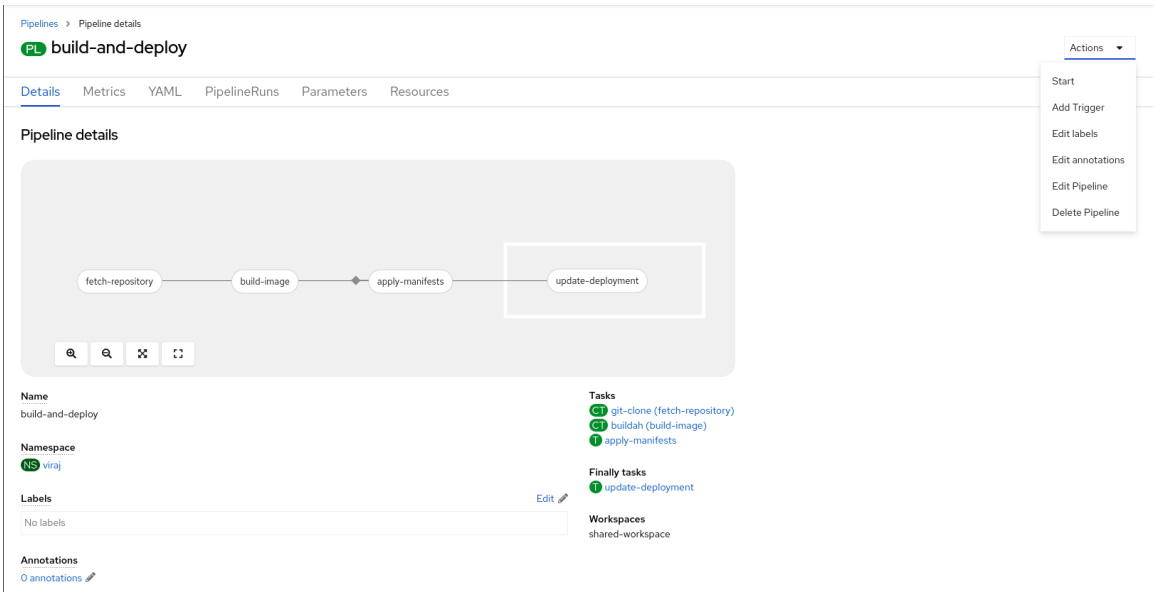
2. 필요한 파이프라인을 클릭하여 파이프라인 세부 정보 페이지를 확인합니다.

기본적으로 세부 정보 탭에는 모든 직렬 작업, 병렬 작업, **finally** 작업 및 파이프라인의 **when** 표현식이 시각적으로 표시됩니다. 작업 및 **finally** 작업은 페이지 오른쪽 하단 목록에 표시됩니다.

작업 세부 정보를 보려면 나열된 **Task** 및 최종 작업을 클릭합니다. 또한 다음을 수행할 수 있습니다.

- **Pipeline** 세부 정보 시각화의 왼쪽 아래에 표시된 표준 아이콘을 사용하여 확대, 축소, 화면에 맞게 보기 기능을 사용합니다.
- 마우스 휠을 사용하여 파이프라인 시각화의 확대/축소 요소를 변경합니다.
- 작업 위로 마우스를 가져가 작업 세부 정보를 확인합니다.

그림 **2.3.** 파이프 라인 세부 정보



3. 선택 사항: 파이프 라인 세부 정보 페이지에서 **Metrics** 탭을 클릭하여 파이프라인에 대한 다음 정보를 확인합니다.

- 파이프 라인 성공률
- 파이프 라인 실행 수
- 파이프 라인 실행 기간
- 작업 실행 기간

이 정보를 사용하여 파이프라인 라이프사이클 초기에 파이프라인 워크플로를 개선하고 문제를 제거할 수 있습니다.

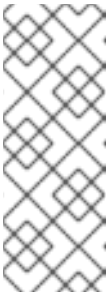
4. 선택 사항: **YAML** 탭을 클릭하여 파이프라인의 **YAML** 파일을 편집합니다.

5. 선택 사항: 파이프라인 실행 탭을 클릭하여 파이프라인 실행 상태가 완료, 실행 중 또는 실패인지 확인합니다.

파이프라인 실행 탭에는 파이프라인 실행, 작업 상태 및 실패한 파이프라인 실행 디버그 링크에 대한 세부 정보가 있습니다. 옵션 메뉴

를 사용하여 실행 중인 파이프라인을 중지하거나, 이전 파이프라인 실행과 동일한 매개변수 및 리소스를 사용하여 파이프라인 재실행 또는 파이프라인 실행을 삭제합니다.

- 필요한 파이프라인 실행을 클릭하여 파이프라인 실행 세부 정보 페이지를 확인합니다. 기본적으로 세부 정보 탭에는 모든 직렬 작업, 병렬 작업, **finally** 작업 및 파이프라인 실행의 **When** 표현식의 시각적 표현이 표시됩니다. 성공적인 실행 결과는 페이지 하단의 파이프라인 실행 결과 창에 표시됩니다. 또한 클러스터에서 지원하는 **Tekton Hub**의 작업만 볼 수 있습니다. 작업을 보는 동안 옆에 있는 링크를 클릭하여 작업 문서로 이동할 수 있습니다.



참고

파이프라인 실행 세부 정보 페이지의 세부 정보 섹션에는 실패한 파이프라인 실행에 대한 로그 조각이 표시됩니다. 로그 조각에는 일반적인 오류 메시지와 해당 로그의 조각이 있습니다. 로그 섹션 링크를 사용하면 실패한 실행에 대한 세부 정보에 빠르게 액세스할 수 있습니다.

- 파이프라인 실행 세부 정보 페이지에서 작업 실행 탭을 클릭하여 작업 상태가 완료, 실행 중 또는 실패인지 확인합니다.

작업 실행 탭은 해당 작업 및 **pod**에 대한 링크, 작업 실행 상태 및 기간과 함께 작업 실행에 대한 정보를 제공합니다. 옵션 메뉴

를 사용하여 작업 실행을 삭제합니다.

참고

TaskRuns 목록 페이지에는 열 관리 버튼이 있으며, 이 버튼을 사용하여 **Duration** 열을 추가할 수도 있습니다.

- 필요한 작업 실행을 클릭하여 작업 실행 세부 정보 페이지를 확인합니다. 성공적으로 실행된 결과는 페이지 하단에 있는 작업 실행 결과 창에 표시됩니다.

참고

작업 실행 세부 정보 페이지의 세부 정보 섹션에는 실패한 작업 실행에 대한 로그 조각이 표시됩니다. 로그 조각에는 일반적인 오류 메시지와 해당 로그의 조각이 있습니다. 로그 섹션 링크를 사용하면 실패한 작업 실행에 대한 세부 정보에 빠르게 액세스할 수 있습니다.

6. 매개변수 탭을 클릭하여 파이프라인에 정의된 매개변수를 확인합니다. 필요에 따라 매개변수를 추가하거나 편집할 수도 있습니다.

7. 리소스 탭을 클릭하여 파이프라인에 정의된 리소스를 확인합니다. 필요에 따라 리소스를 추가하거나 편집할 수도 있습니다.

2.1.5. Pipeline 보기에서 파이프라인 시작

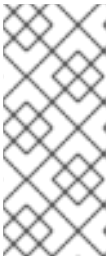
파이프라인을 생성한 후 포함된 작업을 정의된 순서로 실행하려면 파이프라인을 시작해야 합니다. 파이프라인 보기, 파이프라인 세부 정보 페이지 또는 토폴로지 보기에서 파이프라인을 시작할 수 있습니다.

프로세스

파이프라인 보기를 사용하여 파이프라인을 시작하려면 다음을 수행합니다.

1. 개발자 화면의 파이프라인 보기에서 파이프라인 옆에 있는 옵션 메뉴를 클릭하고 시작을 선택합니다.

2. 파이프라인 정의에 따라 파이프라인 시작 대화 상자에 **Git** 리소스 및 이미지 리소스가 표시됩니다.



참고

Git에서 옵션을 사용하여 생성한 파이프라인의 경우 파이프라인 시작 대화 상자의 매개변수 섹션에 **APP_NAME** 필드도 표시되며, 대화 상자의 모든 필드가 파이프라인 템플릿에 의해 미리 채워집니다.

   a. 네임스페이스에 리소스가 있는 경우 **Git Resources** 및 **Image Resources** 필드에 해당 리소스가 미리 채워집니다. 필요한 경우 드롭다운 목록을 사용하여 필요한 리소스를 선택하거나 생성한 다음 파이프라인 실행 인스턴스를 사용자 정의합니다.

3. 선택 사항: 고급 옵션을 수정하여 지정된 프라이빗 **Git** 서버 또는 이미지 레지스트리를 인증하는 자격 증명을 추가합니다.

   a. **Advanced Options**에서 **Show Credentials Options**를 클릭하고 **Add Secret**을 선택합니다.

   b. **Create Source Secre** 섹션에서 다음 사항을 지정합니다.

      i. 보안에 대한 고유한 보안 이름입니다.

      ii. **Designated provider to be authenticated** 섹션에서 **Access to** 필드에 인증할 공

급자를 지정하고 기본 **Server URL**을 지정합니다.

iii. **Authentication Type**을 선택하고 자격 증명을 제공합니다.

- 인증 유형 **Image Registry Crendentials**의 경우 인증할 레지스트리 서버 주소를 지정하고 사용자 이름, 암호, 이메일 필드에 자격 증명을 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.

- 인증 유형 **SSH Keys**의 경우 **SSH** 개인 키 필드 값을 지정합니다.

참고

기본 인증 및 **SSH** 인증의 경우 다음과 같은 주석을 사용할 수 있습니다.

- **tekton.dev/git-0: https://github.com**
- **tekton.dev/git-1: https://gitlab.com.**

iv. 확인 표시를 선택하여 보안을 추가합니다.

파이프라인의 리소스 수에 따라 여러 개의 보안을 추가할 수 있습니다.

4. 시작을 클릭하여 파이프라인을 시작합니다.

5. **PipelineRun** 세부 정보 페이지에는 실행 중인 파이프라인이 표시됩니다. 파이프라인이 시작된 후 작업과 각 작업 내 단계가 실행됩니다. 다음을 수행할 수 있습니다.

- **PipelineRun** 세부 정보 페이지 시각화의 왼쪽 아래에 있는 표준 아이콘을 사용하여 확대, 축소, 화면에 맞는 보기 기능을 사용합니다.

- 마우스 휠을 사용하여 **pipelinerun** 시각화의 확대/축소 요소를 변경합니다. 특정 확대/축소 요인에서 작업의 배경 색상이 변경되어 오류 또는 경고 상태를 나타냅니다.

- 작업 위로 마우스를 이동하여 각 단계, 작업 이름 및 작업 상태를 실행하는 데 걸린 시간과 같은 세부 정보를 확인합니다.

- 작업 배지 위로 마우스를 이동하여 완료된 총 작업 수 및 작업 수를 확인합니다.

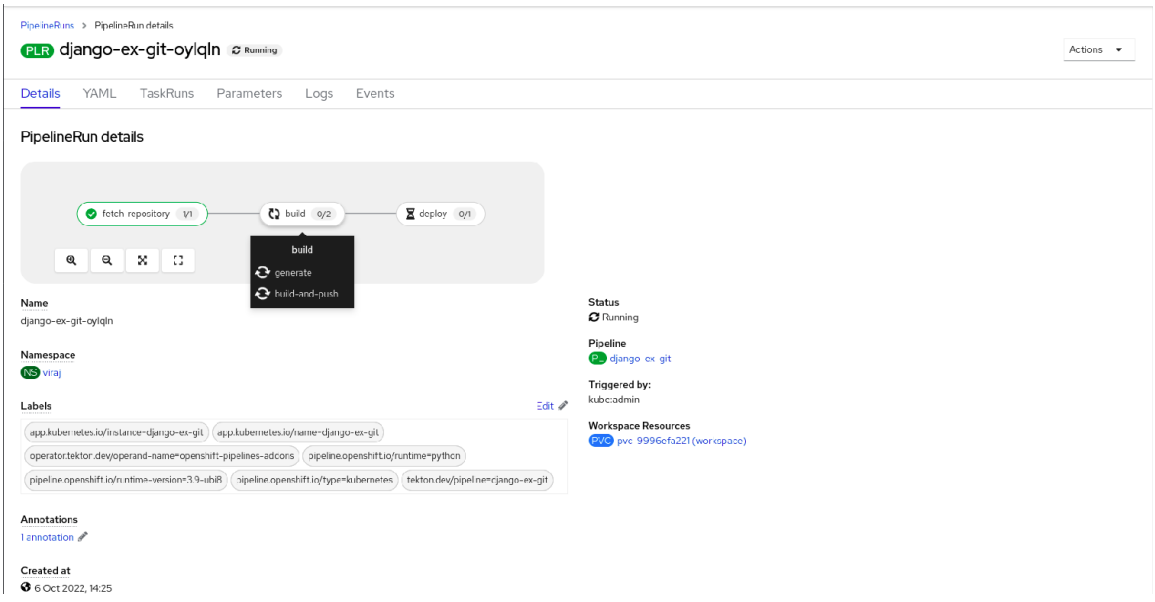
- 작업을 클릭하여 각 작업 단계에 대한 로그를 확인합니다.

- 로그 탭을 클릭하여 작업 실행 순서와 관련된 로그를 확인합니다. 관련 버튼을 사용하여 창을 확장하고 로그를 개별적 또는 일괄적으로 다운로드할 수도 있습니다.

- 이벤트 탭을 클릭하여 파이프라인 실행으로 생성된 이벤트 스트림을 확인합니다.

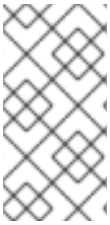
작업 실행, 로그, 이벤트 탭을 사용하면 실패한 파이프라인 실행 또는 실패한 작업 실행을 디버깅하는 데 도움이 될 수 있습니다.

그림 2.4. '파이프 라인 실행' 세부 정보



2.1.6. 토폴로지 보기에서 파이프라인 시작

Git에서 옵션을 사용하여 생성한 파이프라인의 경우 토폴로지 보기를 사용하여 파이프라인을 시작한 후 상호 작용할 수 있습니다.



참고

토폴로지 보기에서 파이프라인 빌더를 사용하여 생성한 파이프라인 을 보려면 파이프라인 레이블을 사용자 지정하여 파이프라인을 애플리케이션 워크로드에 연결합니다.

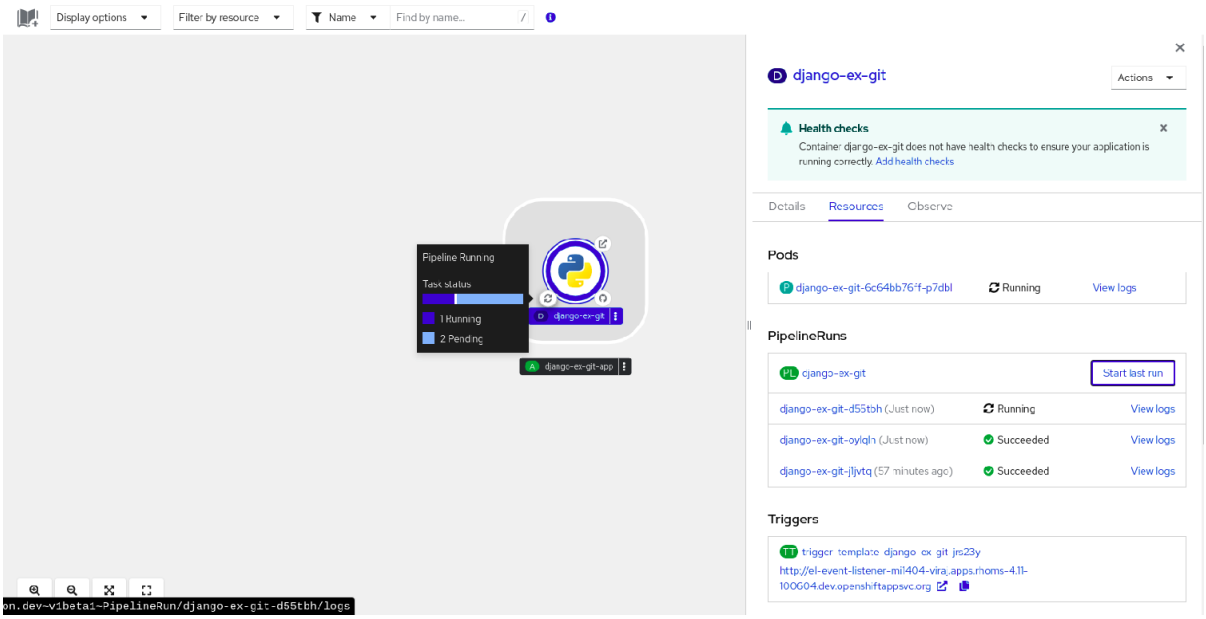
프로세스

1. 왼쪽 탐색 패널에서 **Topology** 를 클릭합니다.

2. 애플리케이션을 클릭하여 측면 패널에 파이프라인 실행을 표시합니다.

3. 파이프라인 실행 에서 마지막 실행 시작을 클릭하여 이전 파이프라인과 동일한 매개변수 및 리소스로 새 파이프라인 실행을 시작합니다. 파이프라인 실행이 시작되지 않은 경우 이 옵션이 비활성화되어 있습니다. 또한 파이프라인 실행을 생성할 때 시작할 수도 있습니다.

그림 2.5. 토폴로지 보기의 파이프라인



토폴로지 페이지에서 애플리케이션 왼쪽으로 마우스를 이동하여 파이프라인 실행 상태를 확인합니다. 파이프라인을 추가한 후 왼쪽 하단 아이콘은 연결된 파이프라인이 있음을 나타냅니다.

2.1.7. 토폴로지 보기에서 파이프라인과 상호 작용

토폴로지 페이지에 있는 애플리케이션 노드의 측면 패널에 파이프라인 실행 상태가 표시되고 상호 작용할 수 있습니다.

- 파이프라인 실행이 자동으로 시작되지 않으면 측면 패널에 파이프라인을 자동으로 시작할 수 없다는 메시지가 표시됩니다. 따라서 수동으로 시작해야 합니다.
- 파이프라인이 생성되었지만 사용자가 파이프라인을 시작하지 않은 경우 해당 상태가 시작되지 않습니다. 사용자가 **Not started** 상태 아이콘을 클릭하면 토폴로지 보기에 시작 대화 상자가 열립니다.
- 파이프라인에 빌드 또는 빌드 구성이 없으면 **Builds** 섹션이 표시되지 않습니다. 파이프라인 및 빌드 구성이 있는 경우 **Builds** 섹션 이 표시됩니다.
- 파이프라인 실행이 특정 작업 실행에서 실패하면 측면 패널에 로그 조각이 표시됩니다. 로그 조각은 리소스 탭의 파이프라인 실행 섹션에서 확인할 수 있습니다. 일반적인 오류 메시지와 로그 스니펫을 제공합니다. 로그 섹션 링크를 사용하면 실패한 실행에 대한 세부 정보에 빠르게 액세스할 수 있습니다.

2.1.8. 파이프라인 편집

웹 콘솔의 개발자 화면을 사용하여 클러스터의 파이프라인을 편집할 수 있습니다.

프로세스

1. 개발자 화면의 **Pipelines** 보기에서 편집할 파이프라인을 선택하여 파이프라인의 세부 정보를 확인합니다. **Pipeline Details** 페이지에서 **Actions**를 클릭하고 **Edit Pipelin**을 선택합니다.
2. **Pipeline** 빌더 페이지에서 다음 작업을 수행할 수 있습니다.
   - 파이프라인에 추가 작업, 매개변수 또는 리소스를 추가합니다.
   - 수정할 작업을 클릭하여 측면 패널에 작업 세부 정보를 표시하고 표시 이름, 매개변수 및 리소스와 같은 필요한 작업 세부 정보를 수정합니다.
   - 또는 작업을 삭제하려면 작업을 클릭하고 측면 패널에서 작업을 클릭하고 작업 제거를

선택합니다.

3. 저장을 클릭하여 수정된 파이프라인을 저장합니다.

2.1.9. 파이프라인 삭제

웹 콘솔의 개발자 화면을 사용하여 클러스터의 파이프라인을 삭제할 수 있습니다.

프로세스

1. 개발자 화면의 **Pipelines** 보기에서 **Pipeline** 옆에 있는 **Options**  메뉴를 클릭하고 **Delete Pipeline** 을 선택합니다.

2. **Delete Pipeline** 확인 프롬프트에서 **Delete**를 클릭하여 삭제를 확인합니다.

2.2. 추가 리소스

- OpenShift Pipelines와 함께 Tekton Hub 사용

2.3. 관리자 화면에서 파이프라인 템플릿 생성

클러스터 관리자는 개발자가 클러스터에서 파이프라인을 생성할 때 재사용할 수 있는 파이프라인 템플릿을 생성할 수 있습니다.

사전 요구 사항

- 클러스터 관리자 권한이 있는 **OpenShift Container Platform** 클러스터에 액세스하고 관리자 화면으로 전환했습니다.

- 클러스터에 **OpenShift Pipelines Operator**를 설치했습니다.

프로세스

1. **Pipelines** 페이지로 이동하여 기존 파이프라인 템플릿을 확인합니다.

2.  아이콘을 클릭하여 **YAML** 가져오기 페이지로 이동합니다.

3. 파이프라인 템플릿에 대한 **YAML**을 추가합니다. 템플릿에는 다음 정보가 포함되어야 합니다.

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
# ...
  namespace: openshift (1)
  labels:
    pipeline.openshift.io/runtime: <runtime> (2)
    pipeline.openshift.io/type: <pipeline-type> (3)
# ...
```

(1) 템플릿은 **openshift** 네임스페이스에 생성해야 합니다.

(2) 템플릿에는 **pipeline.openshift.io/runtime** 레이블이 포함되어야 합니다. 이 라벨에 허용되는 런타임 값은 **nodejs,golang,dotnet,java,php,ruby,perl,python,nginx, httpd** 입니다.

(3) 템플릿에는 **pipeline.openshift.io/type:** 레이블이 포함되어야 합니다. 이 라벨에 허용되는 유형 값은 **openshift,knative, kubernetes** 입니다.

4. 생성을 클릭합니다. 파이프라인이 생성되면 파이프라인 세부 정보 페이지로 이동하여 파이프라인에 대한 정보를 보거나 편집할 수 있습니다.

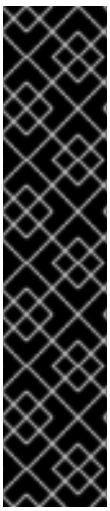
2.4. 웹 콘솔의 파이프라인 실행 통계

웹 콘솔에서 파이프라인 실행과 관련된 통계를 볼 수 있습니다.

통계 정보를 보려면 다음 단계를 완료해야 합니다.

- **Tekton** 결과를 설치합니다. **Tekton** 결과 설치에 대한 자세한 내용은 *추가 리소스* 섹션의 ***OpenShift Pipelines** 관찰 기능에 **Tekton Results** 사용을* 참조하십시오.
- **OpenShift Pipelines** 콘솔 플러그인을 활성화합니다.

통계 정보는 모든 파이프라인과 개별 파이프라인에 대해 사용할 수 있습니다.



중요

OpenShift Pipelines Pipelines 콘솔 플러그인은 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 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 기술 프리뷰 기능 지원 범위를 참조하십시오.

Additonal 리소스

- **OpenShift Pipelines** 관찰 기능에 **Tekton** 결과 사용

2.4.1. OpenShift Pipelines 콘솔 플러그인 활성화

통계 정보를 보려면 먼저 **OpenShift Pipelines** 콘솔 플러그인을 활성화해야 합니다.

사전 요구 사항

- 클러스터에 **Red Hat OpenShift Pipelines Operator**를 설치했습니다.
- 클러스터 관리자 권한이 있는 웹 콘솔에 로그인되어 있습니다.

중요

OpenShift Pipelines 콘솔 플러그인에는 **OpenShift Container Platform** 버전 **4.15** 또는 이후 버전이 필요합니다.

프로세스

1. 웹 콘솔의 관리자 화면에서 **Operator** → 설치된 **Operator** 를 선택합니다.
2. **Operator** 표에서 **Red Hat OpenShift Pipelines** 를 클릭합니다.
3. 화면 오른쪽 창에서 **Console** 플러그인 에서 **status** 레이블을 확인합니다. 레이블은 **Enabled** 또는 **Disabled** 중 하나입니다.
4. 라벨이 **Disabled** 이면 이 레이블을 클릭합니다. 표시되는 창에서 **Enable** 을 선택한 다음 **Save** 를 클릭합니다.

2.4.2. 모든 파이프라인의 통계 보기

시스템의 모든 파이프라인과 관련된 통합 통계 정보를 볼 수 있습니다.

사전 요구 사항

- 클러스터에 **Red Hat OpenShift Pipelines Operator**를 설치했습니다.
- **Tekton** 결과가 설치되어 있어야 합니다.
- **OpenShift Pipelines** 웹 콘솔 플러그인이 설치되어 있어야 합니다.

프로세스

1. 웹 콘솔의 관리자 화면에서 파이프라인 → 개요 를 선택합니다.

통계 개요가 표시됩니다. 이 개요에는 다음 정보가 포함됩니다. 일정 기간 동안 파이프라인 실행의 수 및 상태를 반영하는 그래프는 동일한 기간에 걸쳐 파이프라인 실행의 총 기간, 평균 및

최대 기간입니다. **동일 기간 동안의 총 파이프라인 실행 수입니다. **

파이프라인 테이블이도 표시됩니다. 이 표에는 기간 동안 실행된 모든 파이프라인이 나열되어 기간과 성공률이 표시됩니다.

2. 선택 사항: 필요에 따라 통계 표시 설정을 변경합니다.

- **Project:** 통계를 표시할 프로젝트 또는 네임스페이스입니다.
- **Time range:** 통계를 표시할 기간입니다.
- 새로 고침 간격: **Red Hat OpenShift Pipelines**가 표시되는 동안 창에서 데이터를 업데이트해야 하는 빈도입니다.

2.4.3. 특정 파이프라인의 통계 보기

특정 파이프라인과 관련된 통계 정보를 볼 수 있습니다.

사전 요구 사항

- 클러스터에 **Red Hat OpenShift Pipelines Operator**를 설치했습니다.
- **Tekton** 결과가 설치되어 있어야 합니다.
- **OpenShift Pipelines** 웹 콘솔 플러그인이 설치되어 있어야 합니다.

프로세스

1. 웹 콘솔의 관리자 화면에서 **Pipelines → Pipelines** 를 선택합니다.
2. 파이프라인 목록에서 파이프라인을 클릭합니다. **Pipeline** 세부 정보 보기가 표시됩니다.

3. 지표 탭을 클릭합니다.

통계 개요가 표시됩니다. 이 개요에는 다음 정보가 포함됩니다. 일정 기간 동안 파이프라인 실행의 수 및 상태를 반영하는 그래프는 동일한 기간에 걸쳐 파이프라인 실행의 총 기간, 평균 및 최대 기간입니다. **동일 기간 동안의 총 파이프라인 실행 수입니다. **

4. 선택 사항: 필요에 따라 통계 표시 설정을 변경합니다.

- **Project:** 통계를 표시할 프로젝트 또는 네임스페이스입니다.
- **Time range:** 통계를 표시할 기간입니다.
- 새로 고침 간격: **Red Hat OpenShift Pipelines**가 표시되는 동안 창에서 데이터를 업데이트해야 하는 빈도입니다.

3장. 해결자를 사용하여 원격 파이프라인 및 작업 지정

파이프라인 및 작업은 CI/CD 프로세스에 재사용 가능한 블록입니다. 이전에 개발했거나 다른 사용자가 개발한 파이프라인 또는 작업을 재사용하여 정의를 복사하여 붙여넣을 필요 없이 재사용할 수 있습니다. 이러한 파이프라인 또는 작업은 클러스터의 다른 네임스페이스에서 공용 카탈로그에 이르기까지 여러 유형의 소스에서 사용할 수 있습니다.

파이프라인 실행 리소스에서 기존 소스의 파이프라인을 지정할 수 있습니다. 파이프라인 리소스 또는 작업 실행 리소스에서 기존 소스의 작업을 지정할 수 있습니다.

이러한 경우 Red Hat OpenShift Pipelines *의 해결자*는 런타임 시 지정된 소스에서 파이프라인 또는 작업 정의를 검색합니다.

다음 해결자는 Red Hat OpenShift Pipelines의 기본 설치 프로그램에서 사용할 수 있습니다.

hub resolver

: Artifact Hub 또는 Tekton Hub에서 사용할 수 있는 Pipelines Catalog에서 작업 또는 파이프라인을 검색합니다.

bundles resolver

: OpenShift 컨테이너 리포지토리와 같은 OCI 리포지토리에서 사용할 수 있는 OCI 이미지인 Tekton 번들에서 작업 또는 파이프라인을 검색합니다.

클러스터 확인자

: 특정 네임스페이스의 동일한 OpenShift Container Platform 클러스터에서 이미 생성된 작업 또는 파이프라인을 검색합니다.

git resolver

: Git 리포지토리에서 작업 또는 파이프라인 바인딩을 검색합니다. 리포지토리, 분기 및 경로를 지정해야 합니다.

OpenShift Pipelines 설치에는 파이프라인에서 사용할 수 있는 표준 작업 세트가 포함되어 있습니다. 이러한 작업은 일반적으로 openshift-pipelines 네임스페이스인 OpenShift Pipelines 설치 네임스페이스에 있습니다. 클러스터 확인자를 사용하여 작업에 액세스할 수 있습니다.

3.1. TEKTON 카탈로그에서 원격 파이프라인 또는 작업 지정

hub resolver를 사용하여 Artifact Hub 의 공용 Tekton 카탈로그 또는 Tekton Hub 인스턴스에 정의된 원격 파이프라인 또는 작업을 지정할 수 있습니다.

중요

Artifact Hub 프로젝트는 Red Hat OpenShift Pipelines에서 지원되지 않습니다. Artifact Hub의 구성만 지원됩니다.

3.1.1. hub resolver 구성

hub resolver를 구성하여 리소스 가져오기에 대한 기본 허브와 기본 카탈로그 설정을 변경할 수 있습니다.

프로세스

1. TektonConfig 사용자 지정 리소스를 편집하려면 다음 명령을 입력합니다.

```
$ oc edit TektonConfig config
```

2. TektonConfig 사용자 정의 리소스에서 pipeline.hub-resolver-config 사양을 편집합니다.

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  pipeline:
    hub-resolver-config:
      default-tekton-hub-catalog: Tekton 1
      default-artifact-hub-task-catalog: tekton-catalog-tasks 2
      default-artifact-hub-pipeline-catalog: tekton-catalog-pipelines 3
      defailt-kind: pipeline 4
      default-type: tekton 5
      tekton-hub-api: "https://my-custom-tekton-hub.example.com" 6
      artifact-hub-api: "https://my-custom-artifact-hub.example.com" 7
```

1 리소스를 가져오는 기본 Tekton Hub 카탈로그입니다.

2

작업 리소스를 가져오는 기본 **Artifact Hub** 카탈로그입니다.

3

파이프라인 리소스를 가져오는 기본 **Artifact Hub** 카탈로그입니다.

4

참조의 기본 오브젝트 종류입니다.

5

Artifact Hub의 아티팩트 또는 **Tekton Hub**의 경우 **tekton** 중 하나로 리소스를 가져오는 기본 허브입니다.

6

default-type 옵션이 **tekton** 로 설정된 경우 사용되는 **Tekton Hub API**입니다.

7

선택 사항: **default-type** 옵션이 **artifact** 로 설정된 경우 사용되는 **Artifact Hub API**입니다.

중요

default-type 옵션을 **tekton** 로 설정하는 경우 **tekton -hub-api** 값을 설정하여 **Tekton Hub**의 고유 인스턴스를 구성해야 합니다.

default-type 옵션을 **artifact** 로 설정하면 확인자는 기본적으로 https://artifacthub.io/ 에서 공용 허브 **API**를 사용합니다. **artifact-hub-api** 값을 설정하여 고유한 **Artifact Hub API**를 구성할 수 있습니다.

3.1.2. hub resolver를 사용하여 원격 파이프라인 또는 작업 지정

파이프라인 실행을 생성할 때 **Artifact Hub** 또는 **Tekton Hub**에서 원격 파이프라인을 지정할 수 있습니다. 파이프라인 또는 작업 실행을 생성할 때 **Artifact Hub** 또는 **Tekton Hub**에서 원격 작업을 지정할 수 있습니다.

프로세스

-

Artifact Hub 또는 **Tekton Hub**에서 원격 파이프라인 또는 작업을 지정하려면 **pipelineRef** 또는 **taskRef** 사양에 다음 참조 형식을 사용합니다.

```
# ...
  resolver: hub
  params:
  - name: catalog
    value: <catalog>
  - name: type
    value: <catalog_type>
  - name: kind
    value: [pipeline|task]
  - name: name
    value: <resource_name>
  - name: version
    value: <resource_version>
# ...
```

표 3.1. hub resolver에 대해 지원되는 매개변수

매개변수	설명	예시 값
catalog	리소스를 가져올 카탈로그입니다.	기본값: **tekton-catalog-tasks** (**작업** 종류용), **tekton-catalog- pipeline s** (파이너 종류용).
type	리소스를 가져오는 카탈로그 유형입니다. Artifact Hub의 **아티팩트** 또는 Tekton Hub의 **tekton** 중 하나입니다.	기본값: **artifact**
kind	**작업** 또는 **파이프라인 중 하나입니다**.	기본값: **task**
name	허브에서 가져올 작업 또는 파이프라인의 이름입니다.	**golang-build**
version	허브에서 가져올 작업 또는 파이프라인 버전입니다. 숫자 주위에 따옴표(')를 사용해야 합니다.	**"0.5.0"**

파이프라인 또는 작업에 추가 매개변수가 필요한 경우 파이프라인, 파이프라인 실행 또는 작업 실행 사양의 **params** 섹션에 이러한 매개변수 값을 지정합니다. **pipelineRef** 또는 **taskRef** 사양의 **params** 섹션에는 해결자가 지원하는 매개변수만 포함되어야 합니다.

다음 예제 파이프라인 실행은 카탈로그의 원격 파이프라인을 참조합니다.

```
apiVersion: tekton.dev/v1
kind: PipelineRun
metadata:
  name: hub-pipeline-reference-demo
spec:
  pipelineRef:
    resolver: hub
    params:
    - name: catalog
      value: tekton-catalog-pipelines
    - name: type
      value: artifact
    - name: kind
      value: pipeline
    - name: name
      value: example-pipeline
    - name: version
      value: "0.1"
  params:
  - name: sample-pipeline-parameter
    value: test
```

카탈로그의 원격 작업을 참조하는 다음 예제 파이프라인입니다.

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: pipeline-with-cluster-task-reference-demo
spec:
  tasks:
  - name: "cluster-task-reference-demo"
    taskRef:
      resolver: hub
      params:
      - name: catalog
        value: tekton-catalog-tasks
      - name: type
        value: artifact
      - name: kind
        value: task
      - name: name
        value: example-task
      - name: version
        value: "0.6"
    params:
    - name: sample-task-parameter
      value: test
```

카탈로그의 원격 작업을 참조하는 다음 예제 작업 실행

```
apiVersion: tekton.dev/v1
```

```
kind: TaskRun
metadata:
  name: cluster-task-reference-demo
spec:
  taskRef:
    resolver: hub
    params:
    - name: catalog
      value: tekton-catalog-tasks
    - name: type
      value: artifact
    - name: kind
      value: task
    - name: name
      value: example-task
    - name: version
      value: "0.6"
  params:
  - name: sample-task-parameter
    value: test
```

3.2. TEKTON 번들에서 원격 파이프라인 또는 작업 지정

bundles resolver를 사용하여 **Tekton** 번들에서 원격 파이프라인 또는 작업을 지정할 수 있습니다. **Tekton** 번들은 **OpenShift** 컨테이너 리포지토리와 같은 **OCI** 리포지토리에서 사용할 수 있는 **OCI** 이미지입니다.

3.2.1. 번들 확인 프로그램 구성

bundles 확인자를 구성하여 **Tekton** 번들에서 리소스를 가져오는 기본 서비스 계정 이름 및 기본 유형을 변경할 수 있습니다.

프로세스

1. **TektonConfig** 사용자 지정 리소스를 편집하려면 다음 명령을 입력합니다.

```
$ oc edit TektonConfig config
```

2. **TektonConfig** 사용자 정의 리소스에서 **pipeline.bundles-resolver-config** 사양을 편집합니다.

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
```

```
pipeline:
  bundles-resolver-config:
    default-service-account: pipelines 1
    default-kind: task 2
```

1 번들 요청에 사용할 기본 서비스 계정 이름입니다.

2 번들 이미지의 기본 계층 종류입니다.

3.2.2. 번들 확인기를 사용하여 원격 파이프라인 또는 작업 지정

파이프라인 실행을 생성할 때 Tekton 번들에서 원격 파이프라인을 지정할 수 있습니다. 파이프라인 또는 작업 실행을 생성할 때 Tekton 번들에서 원격 작업을 지정할 수 있습니다.

프로세스

- Tekton 번들에서 원격 파이프라인 또는 작업을 지정하려면 pipelineRef 또는 taskRef 사양에 다음 참조 형식을 사용합니다.

```
# ...
  resolver: bundles
  params:
  - name: bundle
    value: <fully_qualified_image_name>
  - name: name
    value: <resource_name>
  - name: kind
    value: [pipeline|task]
# ...
```

표 3.2. bundles resolver에서 지원되는 매개변수

매개변수	설명	예시 값
serviceAccount	레지스트리 인증 정보를 구성할 때 사용할 서비스 계정의 이름입니다.	default
번들	가져올 이미지를 가리키는 번들 URL입니다.	gcr.io/tekton-releases/catalog/upstream/golang-build:0.1

매개변수	설명	예시 값
name	번들에서 가져올 리소스의 이름입니다.	**golang-build**
kind	번들에서 제거할 리소스의 종류입니다.	**task**

파이프라인 또는 작업에 추가 매개변수가 필요한 경우 파이프라인, 파이프라인 실행 또는 작업 실행 사양의 **params** 섹션에 이러한 매개변수 값을 지정합니다. **pipelineRef** 또는 **taskRef** 사양의 **params** 섹션에는 해결자가 지원하는 매개변수만 포함되어야 합니다.

다음 예제 파이프라인 실행은 Tekton 번들의 원격 파이프라인을 참조합니다.

```
apiVersion: tekton.dev/v1
kind: PipelineRun
metadata:
  name: bundle-pipeline-reference-demo
spec:
  pipelineRef:
    resolver: bundles
    params:
    - name: bundle
      value: registry.example.com:5000/simple/pipeline:latest
    - name: name
      value: hello-pipeline
    - name: kind
      value: pipeline
  params:
  - name: sample-pipeline-parameter
    value: test
  - name: username
    value: "pipelines"
```

다음 예제 파이프라인은 Tekton 번들의 원격 작업을 참조합니다.

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: pipeline-with-bundle-task-reference-demo
spec:
  tasks:
  - name: "bundle-task-demo"
    taskRef:
      resolver: bundles
      params:
      - name: bundle
```

```
      value: registry.example.com:5000/advanced/task:latest
    - name: name
      value: hello-world
    - name: kind
      value: task
  params:
  - name: sample-task-parameter
    value: test
```

다음 예제 작업 실행에서는 **Tekton** 번들의 원격 작업을 참조합니다.

```
apiVersion: tekton.dev/v1
kind: TaskRun
metadata:
  name: bundle-task-reference-demo
spec:
  taskRef:
    resolver: bundles
    params:
    - name: bundle
      value: registry.example.com:5000/simple/new_task:latest
    - name: name
      value: hello-world
    - name: kind
      value: task
  params:
  - name: sample-task-parameter
    value: test
```

3.3. GIT 리포지토리에서 원격 파이프라인 또는 작업 지정

Git 확인자를 사용하여 **Git repostory**에서 원격 파이프라인 또는 작업을 지정할 수 있습니다. 리포지토리에는 파이프라인 또는 작업을 정의하는 **YAML** 파일이 포함되어야 합니다. **Git** 확인자는 익명 복제 또는 인증된 **SCM API**를 사용하여 리포지토리에 액세스할 수 있습니다.

3.3.1. 익명 Git 복제에 대한 Git 확인자 구성

익명 **Git** 복제를 사용하려면 **Git** 리포지토리에서 원격 파이프라인 및 작업을 가져오기 위해 기본 **Git** 버전, 시간 초과 및 기본 리포지토리 **URL**을 구성할 수 있습니다.

프로세스

1. **TektonConfig** 사용자 지정 리소스를 편집하려면 다음 명령을 입력합니다.

```
$ oc edit TektonConfig config
```

2. TektonConfig 사용자 정의 리소스에서 pipeline.git-resolver-config 사양을 편집합니다.

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  pipeline:
    git-resolver-config:
      default-revision: main (1)
      fetch-timeout: 1m (2)
      default-url: https://github.com/tektoncd/catalog.git (3)
```

(1) 지정되지 않은 경우 사용할 기본 Git 버전입니다.

(2) 단일 Git 복제 확인에 최대 시간이 걸릴 수 있습니다(예: 1m,2s,700ms). Red Hat OpenShift Pipelines는 모든 해결 요청에 대해 글로벌 최대 시간 초과를 1분으로 적용합니다.

(3) 없음이 지정되지 않은 경우 익명 복제의 기본 Git 리포지토리 URL입니다.

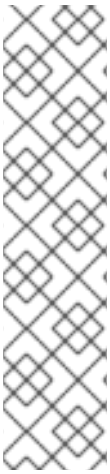
3.3.2. 인증된 SCM API에 대한 Git 확인 프로그램 구성

인증된 SCM API의 경우 인증된 Git 연결에 대한 구성을 설정해야 합니다.

go-scm 라이브러리에서 지원하는 Git 리포지토리 공급자를 사용할 수 있습니다. 모든 go-scm 구현이 Git 리졸버에서 테스트된 것은 아니지만 다음 공급자가 작동하는 것으로 알려져 있습니다.

- GitHub.com 및 GitHub Enterprise
- GitLab.com 및 자체 호스팅 Gitlab
- Gitea

- **BitBucket Server**
- **Bitbucket 클라우드**



참고

- 클러스터에 대해 인증된 **SCM API**를 사용하여 하나의 **Git** 연결만 구성할 수 있습니다. 이 연결은 클러스터의 모든 사용자가 사용할 수 있게 됩니다. 클러스터의 모든 사용자는 연결에 대해 구성하는 보안 토큰을 사용하여 리포지토리에 액세스할 수 있습니다.
- 인증된 **SCM API**를 사용하도록 **Git** 확인자를 구성하는 경우 익명 **Git** 복제 참조를 사용하여 파이프라인 및 작업을 검색할 수도 있습니다.

프로세스

1. **TektonConfig** 사용자 지정 리소스를 편집하려면 다음 명령을 입력합니다.

```
$ oc edit TektonConfig config
```

2. **TektonConfig** 사용자 정의 리소스에서 **pipeline.git-resolver-config** 사양을 편집합니다.

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  pipeline:
    git-resolver-config:
      default-revision: main (1)
      fetch-timeout: 1m (2)
      scm-type: github (3)
      server-url: api.internal-github.com (4)
      api-token-secret-name: github-auth-secret (5)
      api-token-secret-key: github-auth-key (6)
      api-token-secret-namespace: github-auth-namespace (7)
      default-org: tektoncd (8)
```

(1)

지정되지 않은 경우 사용할 기본 **Git** 버전입니다.

2

단일 Git 복제 확인에 최대 시간이 걸릴 수 있습니다(예: 1m,2s,700ms). Red Hat OpenShift Pipelines는 모든 해결 요청에 대해 글로벌 최대 시간 초과를 1분으로 적용합니다.

3

SCM 공급자 유형입니다.

4

인증된 SCM API와 함께 사용할 기본 URL입니다. github.com,gitlab.com 또는 BitBucket Cloud를 사용하는 경우에는 이 설정이 필요하지 않습니다.

5

SCM 공급자 API 토큰이 포함된 시크릿의 이름입니다.

6

토큰이 포함된 토큰 시크릿 내의 키입니다.

7

토큰 시크릿이 포함된 네임스페이스가 기본값이 아닌 경우입니다.

8

선택 사항: 인증된 API를 사용하는 경우 리포지토리의 기본 조직입니다. 이 조직은 확인자 매개변수에 조직을 지정하지 않는 경우 사용됩니다.

참고

인증된 SCM API를 사용하려면 scm-type,api-token-secret-name, api-token-secret-key 설정이 필요합니다.

3.3.3. Git 확인자를 사용하여 원격 파이프라인 또는 작업 지정

파이프라인 실행을 생성할 때 Git 리포지토리에서 원격 파이프라인을 지정할 수 있습니다. 파이프라인 또는 작업 실행을 생성할 때 Git 리포지토리에서 원격 작업을 지정할 수 있습니다.

사전 요구 사항

- 인증된 **SCM API**를 사용하려면 **Git** 확인자에 대해 인증된 **Git** 연결을 구성해야 합니다.

프로세스

1. **Git** 리포지토리에서 원격 파이프라인 또는 작업을 지정하려면 **pipelineRef** 또는 **taskRef** 사양에 다음 참조 형식을 사용합니다.

```
# ...
  resolver: git
  params:
  - name: url
    value: <git_repository_url>
  - name: revision
    value: <branch_name>
  - name: pathInRepo
    value: <path_in_repository>
# ...
```

표 3.3. Git 확인자에 지원되는 매개변수

매개변수	설명	예시 값
url	익명 복제를 사용하는 경우 리포지토리의 URL입니다.	**https://github.com/tektoncd/catalog.git**
repo	인증된 SCM API를 사용하는 경우 리포지토리 이름입니다.	**test-infra**
조직	인증된 SCM API를 사용하는 경우 리포지토리의 조직입니다.	**tektoncd**
버전	리포지토리의 Git 버전입니다. 분기 이름, 태그 이름 또는 커밋 SHA 해시를 지정할 수 있습니다.	**aeb957601cf41c012be462827053a21a420befca** **main** **v0.38.2**
pathInRepo	리포지토리에 있는 YAML 파일의 경로 이름입니다.	**task/golang-build/0.3/golang-build.yaml**

참고

리포지토리를 복제 및 가져오려면 **url** 매개변수를 사용합니다. 인증된 **SCM API**를 사용하려면 **repo** 매개변수를 사용합니다. **url** 매개변수와 **repo** 매개변수를 함께 지정하지 마십시오.

파이프라인 또는 작업에 추가 매개변수가 필요한 경우 파이프라인, 파이프라인 실행 또는 작업 실행 사양의 **params** 섹션에 이러한 매개변수 값을 지정합니다. **pipelineRef** 또는 **taskRef** 사양의 **params** 섹션에는 해결자가 지원하는 매개변수만 포함되어야 합니다.

다음 예제 파이프라인 실행은 Git 리포지토리의 원격 파이프라인을 참조합니다.

```
apiVersion: tekton.dev/v1
kind: PipelineRun
metadata:
  name: git-pipeline-reference-demo
spec:
  pipelineRef:
    resolver: git
    params:
    - name: url
      value: https://github.com/tektoncd/catalog.git
    - name: revision
      value: main
    - name: pathInRepo
      value: pipeline/simple/0.1/simple.yaml
  params:
  - name: name
    value: "testPipelineRun"
  - name: sample-pipeline-parameter
    value: test
```

다음 예제 파이프라인은 Git 리포지토리의 원격 작업을 참조합니다.

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: pipeline-with-git-task-reference-demo
spec:
  tasks:
  - name: "git-task-reference-demo"
    taskRef:
      resolver: git
      params:
      - name: url
        value: https://github.com/tektoncd/catalog.git
      - name: revision
        value: main
      - name: pathInRepo
        value: task/git-clone/0.6/git-clone.yaml
    params:
    - name: sample-task-parameter
      value: test
```

다음 예제 작업 실행에서는 Git 리포지토리의 원격 작업을 참조합니다.

```
apiVersion: tekton.dev/v1
kind: TaskRun
metadata:
  name: git-task-reference-demo
spec:
  taskRef:
    resolver: git
    params:
    - name: url
      value: https://github.com/tektoncd/catalog.git
    - name: revision
      value: main
    - name: pathInRepo
      value: task/git-clone/0.6/git-clone.yaml
  params:
  - name: sample-task-parameter
    value: test
```

3.4. 동일한 클러스터에서 원격 파이프라인 또는 작업 지정

클러스터 확인자를 사용하여 Red Hat OpenShift Pipelines가 실행 중인 OpenShift Container Platform 클러스터의 네임스페이스에 정의된 원격 파이프라인 또는 작업을 지정할 수 있습니다.

특히 클러스터 확인자를 사용하여 OpenShift Pipelines가 설치 네임스페이스(일반적으로 openshift-pipelines 네임스페이스)에서 제공하는 작업에 액세스할 수 있습니다.

3.4.1. 클러스터 확인 프로그램 구성

클러스터 확인자의 기본 종류 및 네임스페이스를 변경하거나 클러스터 확인자가 사용할 수 있는 네임스페이스를 제한할 수 있습니다.

프로세스

1. TektonConfig 사용자 지정 리소스를 편집하려면 다음 명령을 입력합니다.

```
$ oc edit TektonConfig config
```

2. TektonConfig 사용자 정의 리소스에서 pipeline.cluster-resolver-config 사양을 편집합니다.

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  pipeline:
    cluster-resolver-config:
      default-kind: pipeline (1)
      default-namespace: namespace1 (2)
      allowed-namespaces: namespace1, namespace2 (3)
      blocked-namespaces: namespace3, namespace4 (4)
```

(1) 매개변수에 지정되지 않은 경우 가져올 기본 리소스 유형입니다.

(2) 매개변수에 지정되지 않은 경우 리소스를 가져오는 기본 네임스페이스입니다.

(3) 확인자가 액세스할 수 있는 쉼표로 구분된 네임스페이스 목록입니다. 이 키가 정의되지 않은 경우 모든 네임스페이스가 허용됩니다.

(4) 확인자가 액세스할 수 없는 네임스페이스의 선택적 쉼표로 구분된 목록입니다. 이 키가 정의되지 않은 경우 모든 네임스페이스가 허용됩니다.

3.4.2. 클러스터 확인자를 사용하여 원격 파이프라인 또는 작업 지정

파이프라인 실행을 생성할 때 동일한 클러스터의 원격 파이프라인을 지정할 수 있습니다. 파이프라인 또는 작업 실행을 생성할 때 동일한 클러스터에서 원격 작업을 지정할 수 있습니다.

프로세스

- 동일한 클러스터의 원격 파이프라인 또는 작업을 지정하려면 **pipelineRef** 또는 **taskRef** 사양에 다음 참조 형식을 사용합니다.

```
# ...
  resolver: cluster
  params:
  - name: name
    value: <name>
  - name: namespace
```

```
    value: <namespace>
  - name: kind
    value: [pipeline|task]
# ...
```

표 3.4. 클러스터 확인기에서 지원되는 매개변수

매개변수	설명	예시 값
name	가져올 리소스의 이름입니다.	**some-pipeline**
네임스페이스	리소스가 포함된 클러스터의 네임스페이스입니다.	**other-namespace**
kind	가져올 리소스의 종류입니다.	**pipeline**

파이프라인 또는 작업에 추가 매개변수가 필요한 경우 이러한 매개변수를 매개변수에 제공합니다.

다음 예제 파이프라인 실행은 동일한 클러스터의 원격 파이프라인을 참조합니다.

```
apiVersion: tekton.dev/v1
kind: PipelineRun
metadata:
  name: cluster-pipeline-reference-demo
spec:
  pipelineRef:
    resolver: cluster
    params:
    - name: name
      value: some-pipeline
    - name: namespace
      value: test-namespace
    - name: kind
      value: pipeline
  params:
  - name: sample-pipeline-parameter
    value: test
```

다음 예제 파이프라인은 동일한 클러스터의 원격 작업을 참조합니다.

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: pipeline-with-cluster-task-reference-demo
spec:
  tasks:
```

```
- name: "cluster-task-reference-demo"
  taskRef:
    resolver: cluster
    params:
    - name: name
      value: some-task
    - name: namespace
      value: test-namespace
    - name: kind
      value: task
  params:
  - name: sample-task-parameter
    value: test
```

다음 예제 작업 실행에서는 동일한 클러스터의 원격 작업을 참조합니다.

```
apiVersion: tekton.dev/v1
kind: TaskRun
metadata:
  name: cluster-task-reference-demo
spec:
  taskRef:
    resolver: cluster
    params:
    - name: name
      value: some-task
    - name: namespace
      value: test-namespace
    - name: kind
      value: task
  params:
  - name: sample-task-parameter
    value: test
```

3.5. OPENSHIFT PIPELINES 네임스페이스에서 제공되는 작업

OpenShift Pipelines 설치에는 파이프라인에서 사용할 수 있는 표준 작업 세트가 포함되어 있습니다. 이러한 작업은 일반적으로 openshift-pipelines 네임스페이스인 OpenShift Pipelines 설치 네임스페이스에 있습니다. 클러스터 확인자를 사용하여 작업에 액세스할 수 있습니다.

ClusterTask 기능은 OpenShift Pipelines 1.10 이후 더 이상 사용되지 않으며 향후 릴리스에서 제거될 예정입니다. 파이프라인에서 ClusterTasks 를 사용하는 경우 클러스터 확인자를 사용하여 OpenShift Pipelines 설치 네임스페이스에서 사용할 수 있는 작업으로 다시 생성할 수 있습니다. 그러나 기존 ClusterTasks 와 비교하여 이러한 작업에서 특정 변경이 수행됩니다.

OpenShift Pipelines 설치 네임스페이스에서 사용할 수 있는 작업에는 사용자 정의 실행 이미지를 지정할 수 없습니다. 이러한 작업은 BUILDER_IMAGE,gitInitImage 또는 KN_IMAGE 와 같은 매개변수를

지원하지 않습니다. 사용자 정의 실행 이미지를 사용하려면 작업 사본을 생성하고 사본을 편집하여 이미지를 교체합니다.

buildah

buildah 작업에서는 소스 코드 트리를 컨테이너 이미지로 빌드한 다음 이미지를 컨테이너 레지스트리로 내보냅니다.

buildah 작업 사용 예

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: build-and-deploy
spec:
# ...
  tasks:
# ...
  - name: build-image
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: buildah
      - name: namespace
        value: openshift-pipelines
    params:
    - name: IMAGE
      value: $(params.IMAGE)
    workspaces:
    - name: source
      workspace: shared-workspace
# ...
```

표 3.5. buildah 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
IMAGE	Buildah에서 빌드할 정규화된 컨테이너 이미지 이름입니다.	**string**	
DOCKERFILE	소스 작업 공간을 기준으로 하는 **Dockerfile** (또는 **Containerfile**)의 경로입니다.	**string**	**./Dockerfile**

매개변수	설명	유형	기본값
컨텍스트	컨텍스트로 사용할 디렉터리의 경로입니다.	**string**	**.**
STORAGE_DRIVER	현재 클러스터 노드 설정의 설정을 반영하도록 Buildah 스토리지 드라이버를 설정합니다.	**string**	**vfs**
FORMAT	빌드할 컨테이너 형식(**oci** 또는 **docker**)입니다.	**string**	**OCI**
BUILD_EXTRA_ARGS	이미지를 빌드할 때 **빌드** 명령에 대한 추가 매개변수입니다.	**string**	
PUSH_EXTRA_ARGS	이미지를 푸시할 때 **push** 명령에 대한 추가 매개변수입니다.	**string**	
SKIP_PUSH	컨테이너 레지스트리로 이미지 푸시를 건너뜁니다.	**string**	**false**
TLS_VERIFY	TLS 확인 플래그, 일반적으로 **true**.	**string**	**true**
VERBOSE	자세한 로깅을 켜십시오. 실행된 모든 명령이 로그에 추가됩니다.	**string**	**false**

표 3.6. buildah 작업에 지원되는 작업 공간

Workspace	설명
소스	컨테이너 빌드 컨텍스트(일반적으로 **Dockerfile** 또는 **Containerfile** 파일이 포함된 애플리케이션 소스 코드).
dockerconfig	Buildah가 컨테이너 레지스트리에 액세스하는 데 사용하는 **.docker/config.json** 파일을 제공하기 위한 선택적 작업 공간입니다. **config.json** 또는 **.dockerconfigjson** 이라는 이름으로 작업 공간 루트에 파일을 배치합니다.
rhel-entitlement	Buildah가 RHEL(Red Hat Enterprise Linux) 서브스크립션에 액세스하는 데 사용하는 인타이틀먼트 키를 제공하는 선택적 작업 공간입니다. 마운트된 작업 공간에는 **Entitlement.pem** 및 **Entitlement-key.pem 파일이 포함되어야** 합니다.

표 3.7. buildah 작업이 반환되는 결과

결과	유형	설명
IMAGE_URL	**string**	빌드된 이미지의 정규화된 이름입니다.
IMAGE_DIGEST	**string**	빌드된 이미지의 요약입니다.

buildah ClusterTask의 변경 사항

- **VERBOSE 매개변수가 추가되었습니다.**
- **BUILDER_IMAGE 매개변수가 제거되었습니다.**

git-cli

git-cli 작업에서는 git 명령줄 유틸리티를 실행합니다. 전체 Git 명령 또는 여러 명령을 전달하여 GIT_SCRIPT 매개변수를 사용하여 실행할 수 있습니다. 예를 들어 Git 리포지토리에 대한 인증이 필요한 명령의 경우(예: 내보내기를 완료하려면 인증 자격 증명을 제공해야 합니다.

git-cli 작업 사용 예

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: update-repo
spec:
# ...
  tasks:
# ...
  - name: push-to-repo
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: git-cli
      - name: namespace
        value: openshift-pipelines
    params:
    - name: GIT_SCRIPT
      value: "git push"
    - name: GIT_USER_NAME
      value: "Example Developer"
    - name: GIT_USER_EMAIL
      value: "developer@example.com"
    workspaces:
```

```
    - name: ssh-directory
      workspace: ssh-workspace 1
    - name: source
      workspace: shared-workspace
  # ...
```

1

이 예에서 **ssh-workspace** 에는 **Git** 리포지토리에 대한 권한 부여에 유효한 키가 있는 **.ssh** 디렉터리 내용이 포함되어야 합니다.

표 3.8. git-cli 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
CRT_FILENAME	**ssl-ca-directory** 작업 공간에 있는 CA(인증 기관) 번들 파일 이름입니다.	**string**	**ca-bundle.crt**
HTTP_PROXY	HTTP 프록시 서버(TLS 이외의 요청).	**string**	
HTTPS_PROXY	HTTPS 프록시 서버(TLS 요청).	**string**	
NO_PROXY	HTTP/HTTPS 요청 프록시 비활성화.	**string**	
하위 디렉터리	git 리포지토리가 있는 **소스** 작업 공간에 대한 상대 경로입니다.	**string**	
USER_HOME	Pod의 Git 사용자 홈 디렉터리의 절대 경로입니다.	**string**	**/home/git**
DELETE_EXISTING	git 작업을 완료하기 전에 **소스** 작업 영역의 기존 콘텐츠를 지웁니다.	**string**	**true**
VERBOSE	실행된 모든 명령을 기록합니다.	**string**	**false**
SSL_VERIFY	글로벌 **http.sslVerify** 값입니다. 원격 리포지토리를 신뢰하지 않는 한 **false** 를 사용하지 마십시오.	**string**	**true**
GIT_USER_NAME	Git 작업을 수행하기 위한 Git 사용자 이름입니다.	**string**	
GIT_USER_EMAIL	Git 작업을 수행하기 위한 Git 사용자 이메일	**string**	

매개변수	설명	유형	기본값
GIT_SCRIPT	실행할 Git 스크립트입니다.	**string**	**Git 도움말**

표 3.9. git-cli 작업에 지원되는 작업 공간

Workspace	설명
ssh-directory	필요에 따라 개인 키, **known_hosts**,**config** 및 기타 파일이 있는**.ssh** 디렉터리입니다. 이 작업 영역을 제공하는 경우 작업은 Git 리포지토리에 대한 인증에 이 작업 공간을 사용합니다. 인증 정보의 보안 저장을 위해 이 작업 공간을 **Secret** 리소스에 바인딩합니다.
basic-auth	**.gitconfig** 및 **.git-credentials** 파일이 포함된 Workspace입니다. 이 작업 영역을 제공하는 경우 작업은 Git 리포지토리에 대한 인증에 이 작업 공간을 사용합니다. 가능한 경우 **basic-auth** 대신 **ssh-directory** 작업 공간을 사용하십시오. 인증 정보의 보안 저장을 위해 이 작업 공간을 **Secret** 리소스에 바인딩합니다.
ssl-ca-directory	CA 인증서가 포함된 작업 공간입니다. 이 작업 영역을 제공하는 경우 Git은 이러한 인증서를 사용하여 HTTPS를 사용하여 원격 리포지토리와 상호 작용할 때 피어를 확인합니다.
소스	가져온 Git 리포지토리가 포함된 작업 공간입니다.
입력	Git 리포지토리에 추가해야 하는 파일이 포함된 선택적 작업 공간입니다. **$(workspaces.input.path)** 를 사용하여 스크립트의 작업 공간에 액세스할 수 있습니다. 예를 들면 다음과 같습니다. **cp $(workspaces.input.path)/<file_that_i_want> .** **git add <file_that_i_want>**

표 3.10. git-cli 작업이 반환되는 결과

결과	유형	설명
커밋	**string**	복제된 Git 리포지토리에 있는 현재 분기의 HEAD에 있는 커밋의 SHA 다이제스트입니다.

git-cli ClusterTask의 변경 사항

- **몇 가지 새로운 매개변수가 추가되었습니다.**
- **BASE_IMAGE 매개변수가 제거되었습니다.**

- ssl-ca-directory 작업 공간이 추가되었습니다.
- USER_HOME 및 VERBOSE 매개변수의 기본값이 변경되었습니다.
- 결과 이름이 커밋 에서 COMMIT 로 변경되었습니다.

git-clone

git-clone 작업에서는 Git을 사용하여 작업 공간에 원격 리포지토리를 초기화하고 복제합니다. 이 작업은 빌드하거나 이 소스 코드를 처리하는 파이프라인 시작 시 사용할 수 있습니다.

git-clone 작업 사용 예

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: build-source
spec:
# ...
  tasks:
  - name: clone-repo
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: git-clone
      - name: namespace
        value: openshift-pipelines
    params:
    - name: URL
      value: "https://github.com/example/repo.git"
    workspaces:
    - name: output
      workspace: shared-workspace
```

표 3.11. git-clone 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
CRT_FILENAME	**ssl-ca-directory** 작업 공간에 있는 CA(인증 기관) 번들 파일 이름입니다.	**string**	**ca-bundle.crt**
HTTP_PROXY	HTTP 프록시 서버(TLS 이외의 요청).	**string**	
HTTPS_PROXY	HTTPS 프록시 서버(TLS 요청).	**string**	
NO_PROXY	HTTP/HTTPS 요청 프록시 비활성화.	**string**	
하위 디렉터리	작업이 Git 리포지토리를 배치하는 **출력** 작업 공간의 상대 경로입니다.	**string**	
USER_HOME	Pod의 Git 사용자 홈 디렉터리의 절대 경로입니다.	**string**	**/home/git**
DELETE_EXISTING	Git 작업을 실행하기 전에 기본 작업 공간의 콘텐츠(있는 경우)를 삭제합니다.	**string**	**true**
VERBOSE	실행된 명령을 기록합니다.	**string**	**false**
SSL_VERIFY	글로벌 **http.sslVerify** 값입니다. 원격 리포지토리를 신뢰하지 않는 한 이 매개변수를 **false** 로 설정하지 마십시오.	**string**	**true**
URL	Git 리포지토리 URL.	**string**	
버전	확인할 버전(예: 분기 또는 태그)입니다.	**string**	**main**
REFSPEC	버전을 확인하기 전에 작업에서 가져오는 리포지토리의 **refspec** 문자열입니다.	**string**	
하위 모듈	Git 하위 모듈을 초기화하고 가져옵니다.	**string**	**true**
DEPTH	가져올 커밋 수, "shallow clone"은 단일 커밋입니다.	**string**	**1**
SPARSE_CHECKOUT_DIRECTORIES	"스parse 체크아웃"을 수행하기 위해 쉼표로 구분된 디렉터리 패턴 목록입니다.	**string**	

표 3.12. git-clone 작업에 지원되는 작업 공간

Workspace	설명

Workspace	설명
ssh-directory	필요에 따라 개인 키, **known_hosts**,**config** 및 기타 파일이 있는 **.ssh** 디렉터리입니다. 이 작업 영역을 제공하는 경우 작업은 Git 리포지토리에 대한 인증에 이 작업 공간을 사용합니다. 인증 정보의 보안 저장을 위해 이 작업 공간을 **Secret** 리소스에 바인딩합니다.
basic-auth	**.gitconfig** 및 **.git-credentials** 파일이 포함된 Workspace입니다. 이 작업 영역을 제공하는 경우 작업은 Git 리포지토리에 대한 인증에 이 작업 공간을 사용합니다. 가능한 경우 **basic-auth** 대신 **ssh-directory** 작업 공간을 사용하십시오. 인증 정보의 보안 저장을 위해 이 작업 공간을 **Secret** 리소스에 바인딩합니다.
ssl-ca-directory	CA 인증서가 포함된 작업 공간입니다. 이 작업 영역을 제공하는 경우 Git은 이러한 인증서를 사용하여 HTTPS를 사용하여 원격 리포지토리와 상호 작용할 때 피어를 확인합니다.
출력	가져온 git 리포지토리가 포함된 작업 공간, 데이터는 작업 공간의 루트 또는 **SUBDIRECTORY** 매개변수에 정의된 상대 경로에 배치됩니다.

표 3.13. git-clone 작업이 반환되는 결과

결과	유형	설명
커밋	**string**	복제된 Git 리포지토리에 있는 현재 분기의 HEAD에 있는 커밋의 SHA 다이제스트입니다.
URL	**string**	복제된 리포지토리의 URL입니다.
COMMITTER_DATE	**string**	복제된 Git 리포지토리에 있는 현재 분기의 HEAD에 있는 커밋의 epoch 타임스탬프입니다.

git-clone ClusterTask의 변경 사항

- **모든 매개변수 이름이 대문자로 변경되었습니다.**
- **모든 결과 이름이 대문자로 변경되었습니다.**
- **gitInitImage 매개변수가 제거되었습니다.**

kn

kn 작업에서는 kn 명령줄 유틸리티를 사용하여 서비스, 버전 또는 경로와 같은 Knative 리소스에 대한 작업을 완료합니다.

kn 작업 사용 예

```
apiVersion: tekton.dev/v1
kind: PipelineRun
metadata:
 name: kn-run
spec:
 pipelineSpec:
  tasks:
  - name: kn-run
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: kn
      - name: namespace
        value: openshift-pipelines
    params:
    - name: ARGS
      value: [version]
```

표 3.14. kn 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
ARGS	**kn** 유틸리티의 인수입니다.	**array**	**-** 도움말

kn ClusterTask에서 변경 사항

- **KN_IMAGE** 매개변수가 제거되었습니다.

kn-apply

kn-apply 작업은 지정된 이미지를 **Knative** 서비스에 배포합니다. 이 작업에서는 **kn service apply** 명령을 사용하여 지정된 **Knative** 서비스를 생성하거나 업데이트합니다.

kn-apply 작업 사용 예

```
apiVersion: tekton.dev/v1
kind: PipelineRun
metadata:
 name: kn-apply-run
spec:
 pipelineSpec:
  tasks:
  - name: kn-apply-run
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: kn-apply
      - name: namespace
        value: openshift-pipelines
    params:
    - name: SERVICE
      value: "hello"
    - name: IMAGE
      value: "gcr.io/knative-samples/helloworld-go:latest"
```

표 3.15. kn-apply 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
SERVICE	Knative 서비스 이름입니다.	**string**	
IMAGE	배포할 이미지의 정규화된 이름입니다.	**string**	

kn-apply ClusterTask에서 변경 사항

- **KN_IMAGE 매개변수가 제거되었습니다.**

Maven

maven 작업은 Maven 빌드를 실행합니다.

maven 작업 사용 예

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
```

```
  name: build-and-deploy
spec:
# ...
  tasks:
# ...
  - name: build-from-source
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: maven
      - name: namespace
        value: openshift-pipelines
    workspaces:
    - name: source
      workspace: shared-workspace
# ...
```

표 3.16. maven 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
목표	실행할 Maven 목표입니다.	**array**	**- 패키지**
MAVEN_MIRROR_URL	Maven 리포지토리 미러 URL입니다.	**string**	
하위 디렉터리	작업에서 Maven 빌드를 실행하는 **소스** 작업 공간 내의 하위 디렉터리입니다.	**string**	**.**

표 3.17. maven 작업에 지원되는 작업 공간

Workspace	설명
소스	Maven 프로젝트가 포함된 작업 공간입니다.
server_secret	사용자 이름 및 암호와 같이 Maven 서버에 연결하기 위한 시크릿이 포함된 작업 공간입니다.
proxy_secret	사용자 이름 및 암호와 같이 프록시 서버에 연결하기 위한 자격 증명이 포함된 작업 공간입니다.
proxy_configmap	**proxy_port**,**proxy_host**,**proxy_protocol**,**proxy_non_proxy_hosts** 와 같은 프록시 구성 값이 포함된 Workspace입니다.

Workspace	설명
maven_settings	사용자 지정 Maven 설정이 포함된 Workspace입니다.

maven ClusterTask에서 변경 사항

- 매개 변수 이름 **CONTEXT_DIR** 이 **SUBDIRECTORY** 로 변경되었습니다.
- 작업 공간 이름 **maven-settings** 가 **maven_settings** 로 변경되었습니다.

openshift-client

openshift-client 작업에서는 **oc** 명령줄 인터페이스를 사용하여 명령을 실행합니다. 이 작업을 사용하여 **OpenShift Container Platform** 클러스터를 관리할 수 있습니다.

openshift-client 작업 사용 예

```
apiVersion: tekton.dev/v1
kind: PipelineRun
metadata:
 name: openshift-client-run
spec:
 pipelineSpec:
  tasks:
  - name: openshift-client-run
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: openshift-client
      - name: namespace
        value: openshift-pipelines
    params:
    - name: SCRIPT
      value: "oc version"
```

표 3.18. openshift-client 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
SCRIPT	실행할 **oc** CLI 인수입니다.	**string**	**oc help**
VERSION	사용할 OpenShift Container Platform 버전입니다.	**string**	**latest**

표 3.19. openshift-client 작업에 지원되는 작업 공간

Workspace	설명
manifest_dir	**oc** 유틸리티를 사용하여 적용할 매니페스트 파일이 포함된 작업 공간입니다.
kubeconfig_dir	클러스터에 액세스하기 위한 인증 정보가 포함된 **.kube/config** 파일을 제공할 수 있는 선택적 작업 공간입니다. 이 파일을 작업 공간의 루트에 배치하고 **kubeconfig** 로 이름을 지정합니다.

openshift-client ClusterTask에서 변경 사항

- 작업 공간 이름 **manifest-dir** 이 **manifest_dir** 로 변경되었습니다.
- 작업 공간 이름 **kubeconfig-dir** 이 **kubeconfig_dir** 로 변경되었습니다.

s2i-dotnet

s2i-dotnet 작업은 **OpenShift Container Platform** 레지스트리에서 **image-registry.openshift-image-registry.svc:5000/openshift/dotnet** 으로 사용할 수 있는 **S2I(Source to Image) dotnet** 빌더 이미지를 사용하여 소스 코드를 빌드합니다.

s2i-dotnet 작업의 사용 예

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: build-and-deploy
spec:
# ...
  tasks:
# ...
  - name: build-s2i
    taskRef:
      resolver: cluster
      params:
      - name: kind
```

```
      value: task
    - name: name
      value: s2i-dotnet
    - name: namespace
      value: openshift-pipelines
  workspaces:
  - name: source
    workspace: shared-workspace
# ...
```

표 3.20. s2i-dotnet 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
IMAGE	S2I 프로세스에서 빌드하는 컨테이너 이미지의 정규화된 이름입니다.	**string**	
IMAGE_SCRIPTS_URL	빌더 이미지에 대한 기본 assemble 및 run 스크립트가 포함된 URL입니다.	**string**	**image:///usr/libexec/s2i**
ENV_VARS	**KEY=VALUE** 형식으로 나열된 빌드 프로세스에 설정할 환경 변수의 값 배열입니다.	**array**	
컨텍스트	컨텍스트로 사용할 **소스** 작업 공간 내의 디렉터리 경로입니다.	**string**	**.**
STORAGE_DRIVER	현재 클러스터 노드 설정의 설정을 반영하도록 Buildah 스토리지 드라이버를 설정합니다.	**string**	**vfs**
FORMAT	빌드할 컨테이너 형식(**oci** 또는 **docker**)입니다.	**string**	**OCI**
BUILD_EXTRA_ARGS	이미지를 빌드할 때 **빌드** 명령에 대한 추가 매개변수입니다.	**string**	
PUSH_EXTRA_ARGS	이미지를 푸시할 때 **push** 명령에 대한 추가 매개변수입니다.	**string**	
SKIP_PUSH	컨테이너 레지스트리로 이미지 푸시를 건너뜁니다.	**string**	**false**
TLS_VERIFY	TLS 확인 플래그, 일반적으로 **true**.	**string**	**true**
VERBOSE	자세한 로깅을 켜십시오. 실행된 모든 명령이 로그에 추가됩니다.	**string**	**false**

매개변수	설명	유형	기본값
VERSION	언어 버전에 해당하는 이미지 스트림의 태그입니다.	**string**	**latest**

표 3.21. s2i-dotnet 작업에 지원되는 작업 공간

Workspace	설명
소스	S2I 워크플로의 빌드 컨텍스트인 애플리케이션 소스 코드입니다.
dockerconfig	Buildah가 컨테이너 레지스트리에 액세스하는 데 사용하는 **.docker/config.json** 파일을 제공하기 위한 선택적 작업 공간입니다. **config.json** 또는 **.dockerconfigjson** 이라는 이름으로 작업 공간 루트에 파일을 배치합니다.

표 3.22. s2i-dotnet 작업이 반환되는 결과

결과	유형	설명
IMAGE_URL	**string**	빌드된 이미지의 정규화된 이름입니다.
IMAGE_DIGEST	**string**	빌드된 이미지의 요약입니다.

s2i-dotnet ClusterTask에서 변경 사항

- **몇 가지 새로운 매개변수가 추가되었습니다.**
- **BASE_IMAGE 매개변수가 제거되었습니다.**
- **매개변수 이름 PATH_CONTEXT 가 CONTEXT 로 변경되었습니다.**
- **매개변수 이름 TLS_VERIFY 가 TLSVERIFY 로 변경되었습니다.**
- **IMAGE_URL 결과가 추가되었습니다.**

s2i-go

s2i-go 작업에서는 OpenShift Container Platform 레지스트리에서 image-registry.openshift-image-registry.svc:5000/openshift/golang 으로 사용할 수 있는 S2I Golang 빌더 이미지를 사용하여 소스 코

드를 빌드합니다.

s2i-go 작업의 사용 예

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: build-and-deploy
spec:
# ...
  tasks:
# ...
  - name: build-s2i
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: s2i-go
      - name: namespace
        value: openshift-pipelines
    workspaces:
    - name: source
      workspace: shared-workspace
# ...
```

표 3.23. s2i-go 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
IMAGE	S2I 프로세스에서 빌드하는 컨테이너 이미지의 정규화된 이름입니다.	**string**	
IMAGE_SCRIPTS_URL	빌더 이미지에 대한 기본 assemble 및 run 스크립트가 포함된 URL입니다.	**string**	**image:///usr/libexec/s2i**
ENV_VARS	**KEY=VALUE** 형식으로 나열된 빌드 프로세스에 설정할 환경 변수의 값 배열입니다.	**array**	
컨텍스트	컨텍스트로 사용할 **소스** 작업 공간 내의 디렉터리 경로입니다.	**string**	**.**
STORAGE_DRIVER	현재 클러스터 노드 설정의 설정을 반영하도록 Buildah 스토리지 드라이버를 설정합니다.	**string**	**vfs**

매개변수	설명	유형	기본값
FORMAT	빌드할 컨테이너 형식(**oci** 또는 **docker**) 입니다.	**string**	**OCI**
BUILD_EXTRA_ARGS	이미지를 빌드할 때 **빌드** 명령에 대한 추가 매개변수입니다.	**string**	
PUSH_EXTRA_ARGS	이미지를 푸시할 때 **push** 명령에 대한 추가 매개변수입니다.	**string**	
SKIP_PUSH	컨테이너 레지스트리로 이미지 푸시를 건너뜁니다.	**string**	**false**
TLS_VERIFY	TLS 확인 플래그, 일반적으로 **true**.	**string**	**true**
VERBOSE	자세한 로깅을 켜십시오. 실행된 모든 명령이 로그에 추가됩니다.	**string**	**false**
VERSION	언어 버전에 해당하는 이미지 스트림의 태그입니다.	**string**	**latest**

표 3.24. s2i-go 작업에 지원되는 작업 공간

Workspace	설명
소스	S2I 워크플로의 빌드 컨텍스트인 애플리케이션 소스 코드입니다.
dockerconfig	Buildah가 컨테이너 레지스트리에 액세스하는 데 사용하는 **.docker/config.json** 파일을 제공하기 위한 선택적 작업 공간입니다. **config.json** 또는 **.dockerconfigjson** 이라는 이름으로 작업 공간 루트에 파일을 배치합니다.

표 3.25. s2i-go 작업이 반환되는 결과

결과	유형	설명
IMAGE_URL	**string**	빌드된 이미지의 정규화된 이름입니다.
IMAGE_DIGEST	**string**	빌드된 이미지의 요약입니다.

s2i-go ClusterTask에서 변경 사항

- 몇 가지 새로운 매개변수가 추가되었습니다.
- BASE_IMAGE 매개변수가 제거되었습니다.
- 매개변수 이름 PATH_CONTEXT 가 CONTEXT 로 변경되었습니다.
- 매개변수 이름 TLS_VERIFY 가 TLSVERIFY 로 변경되었습니다.
- IMAGE_URL 결과가 추가되었습니다.

s2i-java

s2i-java 작업은 OpenShift Container Platform 레지스트리에서 image-registry.openshift-image-registry.svc:5000/openshift/java 로 사용할 수 있는 S2I Java 빌더 이미지를 사용하여 소스 코드를 빌드합니다.

표 3.26. s2i-java 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
IMAGE	S2I 프로세스에서 빌드하는 컨테이너 이미지의 정규화된 이름입니다.	string	
IMAGE_SCRIPTS_URL	빌더 이미지에 대한 기본 assemble 및 run 스크립트가 포함된 URL입니다.	string	image:///usr/libexec/s2i
ENV_VARS	KEY=VALUE 형식으로 나열된 빌드 프로세스에 설정할 환경 변수의 값 배열입니다.	array	
컨텍스트	컨텍스트로 사용할 소스 작업 공간 내의 디렉터리 경로입니다.	string	.
STORAGE_DRIVER	현재 클러스터 노드 설정의 설정을 반영하도록 Buildah 스토리지 드라이버를 설정합니다.	string	vfs
FORMAT	빌드할 컨테이너 형식(oci 또는 docker) 입니다.	string	OCI
BUILD_EXTRA_ARGS	이미지를 빌드할 때 빌드 명령에 대한 추가 매개변수입니다.	string	

매개변수	설명	유형	기본값
PUSH_EXTRA_ARGS	이미지를 푸시할 때 **push** 명령에 대한 추가 매개변수입니다.	**string**	
SKIP_PUSH	컨테이너 레지스트리로 이미지 푸시를 건너뜁니다.	**string**	**false**
TLS_VERIFY	TLS 확인 플래그, 일반적으로 **true**.	**string**	**true**
VERBOSE	자세한 로깅을 켜십시오. 실행된 모든 명령이 로그에 추가됩니다.	**string**	**false**
VERSION	언어 버전에 해당하는 이미지 스트림의 태그입니다.	**string**	**latest**

표 3.27. s2i-java 작업에 지원되는 작업 공간

Workspace	설명
소스	S2I 워크플로의 빌드 컨텍스트인 애플리케이션 소스 코드입니다.
dockerconfig	Buildah가 컨테이너 레지스트리에 액세스하는 데 사용하는 **.docker/config.json** 파일을 제공하기 위한 선택적 작업 공간입니다. **config.json** 또는 **.dockerconfigjson** 이라는 이름으로 작업 공간 루트에 파일을 배치합니다.

표 3.28. s2i-java 작업이 반환됨

결과	유형	설명
IMAGE_URL	**string**	빌드된 이미지의 정규화된 이름입니다.
IMAGE_DIGEST	**string**	빌드된 이미지의 요약입니다.

s2i-java ClusterTask에서 변경 사항

- **몇 가지 새로운 매개변수가 추가되었습니다.**
- **BUILDER_IMAGE,MAVEN_ARGS_APPEND,MAVEN_CLEAR_REPO, MAVEN_MIRROR_URL 매개변수가 제거되었습니다. MAVEN_ARGS_APPEND,MAVEN_CLEAR_REPO 및 MAVEN_MIRROR_URL 값을 환경 변수로 전달할 수 있습니다.**

- 매개변수 이름 **PATH_CONTEXT** 가 **CONTEXT** 로 변경되었습니다.
- 매개변수 이름 **TLS_VERIFY** 가 **TLSVERIFY** 로 변경되었습니다.
- **IMAGE_URL** 결과가 추가되었습니다.

s2i-nodejs

s2i-nodejs 작업에서는 **OpenShift Container Platform** 레지스트리에서 **image-registry.openshift-image-registry.svc:5000/openshift/nodejs** 로 사용할 수 있는 **S2I NodeJS** 빌더 이미지를 사용하여 소스 코드를 빌드합니다.

s2i-nodejs 작업 사용 예

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: build-and-deploy
spec:
# ...
  tasks:
# ...
  - name: build-s2i
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: s2i-nodejs
      - name: namespace
        value: openshift-pipelines
    workspaces:
    - name: source
      workspace: shared-workspace
# ...
```

표 **3.29. s2i-nodejs** 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
IMAGE	S2I 프로세스에서 빌드하는 컨테이너 이미지의 정규화된 이름입니다.	**string**	
IMAGE_SCRIPTS_URL	빌더 이미지에 대한 기본 assemble 및 run 스크립트가 포함된 URL입니다.	**string**	**image:///usr/libexec/s2i**
ENV_VARS	**KEY=VALUE** 형식으로 나열된 빌드 프로세스에 설정할 환경 변수의 값 배열입니다.	**array**	
컨텍스트	컨텍스트로 사용할 **소스** 작업 공간 내의 디렉터리 경로입니다.	**string**	**.**
STORAGE_DRIVER	현재 클러스터 노드 설정의 설정을 반영하도록 Buildah 스토리지 드라이버를 설정합니다.	**string**	**vfs**
FORMAT	빌드할 컨테이너 형식(**oci** 또는 **docker**)입니다.	**string**	**OCI**
BUILD_EXTRA_ARGS	이미지를 빌드할 때 **빌드** 명령에 대한 추가 매개변수입니다.	**string**	
PUSH_EXTRA_ARGS	이미지를 푸시할 때 **push** 명령에 대한 추가 매개변수입니다.	**string**	
SKIP_PUSH	컨테이너 레지스트리로 이미지 푸시를 건너뜁니다.	**string**	**false**
TLS_VERIFY	TLS 확인 플래그, 일반적으로 **true**.	**string**	**true**
VERBOSE	자세한 로깅을 켜십시오. 실행된 모든 명령이 로그에 추가됩니다.	**string**	**false**
VERSION	언어 버전에 해당하는 이미지 스트림의 태그입니다.	**string**	**latest**

표 3.30. s2i-nodejs 작업에 지원되는 작업 공간

Workspace	설명
소스	S2I 워크플로의 빌드 컨텍스트인 애플리케이션 소스 코드입니다.
dockerconfig	Buildah가 컨테이너 레지스트리에 액세스하는 데 사용하는 **.docker/config.json** 파일을 제공하기 위한 선택적 작업 공간입니다. **config.json** 또는 **.dockerconfigjson** 이라는 이름으로 작업 공간 루트에 파일을 배치합니다.

표 3.31. s2i-nodejs 작업이 반환됨

결과	유형	설명
IMAGE_URL	string	빌드된 이미지의 정규화된 이름입니다.
IMAGE_DIGEST	string	빌드된 이미지의 요약입니다.

s2i-nodejs ClusterTask에서 변경 사항

- 몇 가지 새로운 매개변수가 추가되었습니다.
- BASE_IMAGE 매개변수가 제거되었습니다.
- 매개변수 이름 PATH_CONTEXT 가 CONTEXT 로 변경되었습니다.
- 매개변수 이름 TLS_VERIFY 가 TLSVERIFY 로 변경되었습니다.
- IMAGE_URL 결과가 추가되었습니다.

s2i-perl

s2i-perl 작업은 OpenShift Container Platform 레지스트리에서 image-registry.openshift-image-registry.svc:5000/openshift/perl 로 사용할 수 있는 S2I Perl 빌더 이미지를 사용하여 소스 코드를 빌드합니다.

s2i-perl 작업의 사용 예

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: build-and-deploy
spec:
# ...
  tasks:
# ...
  - name: build-s2i
    taskRef:
      resolver: cluster
      params:
      - name: kind
```

```
      value: task
    - name: name
      value: s2i-perl
    - name: namespace
      value: openshift-pipelines
  workspaces:
  - name: source
    workspace: shared-workspace
# ...
```

표 3.32. s2i-perl 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
IMAGE	S2I 프로세스에서 빌드하는 컨테이너 이미지의 정규화된 이름입니다.	**string**	
IMAGE_SCRIPTS_URL	빌더 이미지에 대한 기본 assemble 및 run 스크립트가 포함된 URL입니다.	**string**	**image:///usr/libexec/s2i**
ENV_VARS	**KEY=VALUE** 형식으로 나열된 빌드 프로세스에 설정할 환경 변수의 값 배열입니다.	**array**	
컨텍스트	컨텍스트로 사용할 **소스** 작업 공간 내의 디렉터리 경로입니다.	**string**	**.**
STORAGE_DRIVER	현재 클러스터 노드 설정의 설정을 반영하도록 Buildah 스토리지 드라이버를 설정합니다.	**string**	**vfs**
FORMAT	빌드할 컨테이너 형식(**oci** 또는 **docker**)입니다.	**string**	**OCI**
BUILD_EXTRA_ARGS	이미지를 빌드할 때 **빌드** 명령에 대한 추가 매개변수입니다.	**string**	
PUSH_EXTRA_ARGS	이미지를 푸시할 때 **push** 명령에 대한 추가 매개변수입니다.	**string**	
SKIP_PUSH	컨테이너 레지스트리로 이미지 푸시를 건너뜁니다.	**string**	**false**
TLS_VERIFY	TLS 확인 플래그, 일반적으로 **true**.	**string**	**true**
VERBOSE	자세한 로깅을 켜십시오. 실행된 모든 명령이 로그에 추가됩니다.	**string**	**false**

매개변수	설명	유형	기본값
VERSION	언어 버전에 해당하는 이미지 스트림의 태그입니다.	**string**	**latest**

표 3.33. s2i-perl 작업에 지원되는 작업 공간

Workspace	설명
소스	S2I 워크플로의 빌드 컨텍스트인 애플리케이션 소스 코드입니다.
dockerconfig	Buildah가 컨테이너 레지스트리에 액세스하는 데 사용하는 **.docker/config.json** 파일을 제공하기 위한 선택적 작업 공간입니다. **config.json** 또는 **.dockerconfigjson** 이라는 이름으로 작업 공간 루트에 파일을 배치합니다.

표 3.34. s2i-perl 작업이 반환되는 결과

결과	유형	설명
IMAGE_URL	**string**	빌드된 이미지의 정규화된 이름입니다.
IMAGE_DIGEST	**string**	빌드된 이미지의 요약입니다.

s2i-perl ClusterTask에서 변경 사항

- 몇 가지 새로운 매개변수가 추가되었습니다.
- **BASE_IMAGE** 매개변수가 제거되었습니다.
- 매개변수 이름 **PATH_CONTEXT** 가 **CONTEXT** 로 변경되었습니다.
- 매개변수 이름 **TLS_VERIFY** 가 **TLSVERIFY** 로 변경되었습니다.
- **IMAGE_URL** 결과가 추가되었습니다.

s2i-php

s2i-php 작업은 **OpenShift Container Platform** 레지스트리에서 **image-registry.openshift-image-registry.svc:5000/openshift/php** 로 사용할 수 있는 **S2I PHP** 빌더 이미지를 사용하여 소스 코드를 빌드

합니다.

s2i-php 작업 사용 예

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: build-and-deploy
spec:
# ...
  tasks:
# ...
  - name: build-s2i
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: s2i-php
      - name: namespace
        value: openshift-pipelines
    workspaces:
    - name: source
      workspace: shared-workspace
# ...
```

표 3.35. s2i-php 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
IMAGE	S2I 프로세스에서 빌드하는 컨테이너 이미지의 정규화된 이름입니다.	**string**	
IMAGE_SCRIPTS_URL	빌더 이미지에 대한 기본 assemble 및 run 스크립트가 포함된 URL입니다.	**string**	**image:///usr/libexec/s2i**
ENV_VARS	**KEY=VALUE** 형식으로 나열된 빌드 프로세스에 설정할 환경 변수의 값 배열입니다.	**array**	
컨텍스트	컨텍스트로 사용할 **소스** 작업 공간 내의 디렉터리 경로입니다.	**string**	**.**
STORAGE_DRIVER	현재 클러스터 노드 설정의 설정을 반영하도록 Buildah 스토리지 드라이버를 설정합니다.	**string**	**vfs**

매개변수	설명	유형	기본값
FORMAT	빌드할 컨테이너 형식(**oci** 또는 **docker**) 입니다.	**string**	**OCI**
BUILD_EXTRA_ARGS	이미지를 빌드할 때 **빌드** 명령에 대한 추가 매개변수입니다.	**string**	
PUSH_EXTRA_ARGS	이미지를 푸시할 때 **push** 명령에 대한 추가 매개변수입니다.	**string**	
SKIP_PUSH	컨테이너 레지스트리로 이미지 푸시를 건너뜁니다.	**string**	**false**
TLS_VERIFY	TLS 확인 플래그, 일반적으로 **true**.	**string**	**true**
VERBOSE	자세한 로깅을 켜십시오. 실행된 모든 명령이 로그에 추가됩니다.	**string**	**false**
VERSION	언어 버전에 해당하는 이미지 스트림의 태그입니다.	**string**	**latest**

표 3.36. s2i-php 작업에 지원되는 작업 공간

Workspace	설명
소스	S2I 워크플로의 빌드 컨텍스트인 애플리케이션 소스 코드입니다.
dockerconfig	Buildah가 컨테이너 레지스트리에 액세스하는 데 사용하는 **.docker/config.json** 파일을 제공하기 위한 선택적 작업 공간입니다. **config.json** 또는 **.dockerconfigjson** 이라는 이름으로 작업 공간 루트에 파일을 배치합니다.

표 3.37. s2i-php 작업이 반환되는 결과

결과	유형	설명
IMAGE_URL	**string**	빌드된 이미지의 정규화된 이름입니다.
IMAGE_DIGEST	**string**	빌드된 이미지의 요약입니다.

s2i-php ClusterTask에서 변경 사항

- 몇 가지 새로운 매개변수가 추가되었습니다.
- **BASE_IMAGE** 매개변수가 제거되었습니다.
- 매개변수 이름 **PATH_CONTEXT** 가 **CONTEXT** 로 변경되었습니다.
- 매개변수 이름 **TLS_VERIFY** 가 **TLSVERIFY** 로 변경되었습니다.
- **IMAGE_URL** 결과가 추가되었습니다.

s2i-python

s2i-python 작업은 **OpenShift Container Platform** 레지스트리에서 **image-registry.openshift-image-registry.svc:5000/openshift/python** 으로 사용할 수 있는 **S2I Python** 빌더 이미지를 사용하여 소스 코드를 빌드합니다.

s2i-python 작업 사용 예

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: build-and-deploy
spec:
# ...
  tasks:
# ...
  - name: build-s2i
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: s2i-python
      - name: namespace
        value: openshift-pipelines
    workspaces:
    - name: source
      workspace: shared-workspace
# ...
```

표 3.38. s2i-python 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
IMAGE	S2I 프로세스에서 빌드하는 컨테이너 이미지의 정규화된 이름입니다.	**string**	
IMAGE_SCRIPTS_URL	빌더 이미지에 대한 기본 assemble 및 run 스크립트가 포함된 URL입니다.	**string**	**image:///usr/libexec/s2i**
ENV_VARS	**KEY=VALUE** 형식으로 나열된 빌드 프로세스에 설정할 환경 변수의 값 배열입니다.	**array**	
컨텍스트	컨텍스트로 사용할 **소스** 작업 공간 내의 디렉터리 경로입니다.	**string**	**.**
STORAGE_DRIVER	현재 클러스터 노드 설정의 설정을 반영하도록 Buildah 스토리지 드라이버를 설정합니다.	**string**	**vfs**
FORMAT	빌드할 컨테이너 형식(**oci** 또는 **docker**)입니다.	**string**	**OCI**
BUILD_EXTRA_ARGS	이미지를 빌드할 때 **빌드** 명령에 대한 추가 매개변수입니다.	**string**	
PUSH_EXTRA_ARGS	이미지를 푸시할 때 **push** 명령에 대한 추가 매개변수입니다.	**string**	
SKIP_PUSH	컨테이너 레지스트리로 이미지 푸시를 건너뜁니다.	**string**	**false**
TLS_VERIFY	TLS 확인 플래그, 일반적으로 **true**.	**string**	**true**
VERBOSE	자세한 로깅을 켜십시오. 실행된 모든 명령이 로그에 추가됩니다.	**string**	**false**
VERSION	언어 버전에 해당하는 이미지 스트림의 태그입니다.	**string**	**latest**

표 3.39. s2i-python 작업에 지원되는 작업 공간

Workspace	설명
소스	S2I 워크플로의 빌드 컨텍스트인 애플리케이션 소스 코드입니다.

Workspace	설명
dockerconfig	Buildah가 컨테이너 레지스트리에 액세스하는 데 사용하는 **.docker/config.json** 파일을 제공하기 위한 선택적 작업 공간입니다. **config.json** 또는 **.dockerconfigjson** 이라는 이름으로 작업 공간 루트에 파일을 배치합니다.

표 3.40. s2i-python 작업이 반환되는 결과

결과	유형	설명
IMAGE_URL	**string**	빌드된 이미지의 정규화된 이름입니다.
IMAGE_DIGEST	**string**	빌드된 이미지의 요약입니다.

s2i-python ClusterTask에서 변경 사항

- 몇 가지 새로운 매개변수가 추가되었습니다.
- **BASE_IMAGE** 매개변수가 제거되었습니다.
- 매개변수 이름 **PATH_CONTEXT** 가 **CONTEXT** 로 변경되었습니다.
- 매개변수 이름 **TLS_VERIFY** 가 **TLSVERIFY** 로 변경되었습니다.
- **IMAGE_URL** 결과가 추가되었습니다.

s2i-ruby

s2i-ruby 작업은 **OpenShift Container Platform** 레지스트리에서 **image-registry.openshift-image-registry.svc:5000/openshift/ruby** 로 사용할 수 있는 **S2I Ruby** 빌더 이미지를 사용하여 소스 코드를 빌드합니다.

s2i-ruby 작업 사용 예

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: build-and-deploy
```

```
spec:
# ...
  tasks:
# ...
  - name: build-s2i
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: s2i-ruby
      - name: namespace
        value: openshift-pipelines
    workspaces:
    - name: source
      workspace: shared-workspace
# ...
```

표 3.41. s2i-ruby 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
IMAGE	S2I 프로세스에서 빌드하는 컨테이너 이미지의 정규화된 이름입니다.	**string**	
IMAGE_SCRIPTS_URL	빌더 이미지에 대한 기본 assemble 및 run 스크립트가 포함된 URL입니다.	**string**	**image:///usr/libexec/s2i**
ENV_VARS	**KEY=VALUE** 형식으로 나열된 빌드 프로세스에 설정할 환경 변수의 값 배열입니다.	**array**	
컨텍스트	컨텍스트로 사용할 **소스** 작업 공간 내의 디렉터리 경로입니다.	**string**	**.**
STORAGE_DRIVER	현재 클러스터 노드 설정의 설정을 반영하도록 Buildah 스토리지 드라이버를 설정합니다.	**string**	**vfs**
FORMAT	빌드할 컨테이너 형식(**oci** 또는 **docker**)입니다.	**string**	**OCI**
BUILD_EXTRA_ARGS	이미지를 빌드할 때 **빌드** 명령에 대한 추가 매개변수입니다.	**string**	
PUSH_EXTRA_ARGS	이미지를 푸시할 때 **push** 명령에 대한 추가 매개변수입니다.	**string**	

매개변수	설명	유형	기본값
SKIP_PUSH	컨테이너 레지스트리로 이미지 푸시를 건너뜁니다.	**string**	**false**
TLS_VERIFY	TLS 확인 플래그, 일반적으로 **true**.	**string**	**true**
VERBOSE	자세한 로깅을 켜십시오. 실행된 모든 명령이 로그에 추가됩니다.	**string**	**false**
VERSION	언어 버전에 해당하는 이미지 스트림의 태그입니다.	**string**	**latest**

표 3.42. s2i-ruby 작업에 지원되는 작업 공간

Workspace	설명
소스	S2I 워크플로의 빌드 컨텍스트인 애플리케이션 소스 코드입니다.
dockerconfig	Buildah가 컨테이너 레지스트리에 액세스하는 데 사용하는 **.docker/config.json** 파일을 제공하기 위한 선택적 작업 공간입니다. **config.json** 또는 **.dockerconfigjson** 이라는 이름으로 작업 공간 루트에 파일을 배치합니다.

표 3.43. s2i-ruby 작업이 반환되는 결과

결과	유형	설명
IMAGE_URL	**string**	빌드된 이미지의 정규화된 이름입니다.
IMAGE_DIGEST	**string**	빌드된 이미지의 요약입니다.

s2i-ruby ClusterTask에서 변경 사항

- **몇 가지 새로운 매개변수가 추가되었습니다.**
- **BASE_IMAGE 매개변수가 제거되었습니다.**
- **매개변수 이름 PATH_CONTEXT 가 CONTEXT 로 변경되었습니다.**
- **매개변수 이름 TLS_VERIFY 가 TLSVERIFY 로 변경되었습니다.**

- IMAGE_URL 결과가 추가되었습니다.

Skopeo-copy

skopeo-copy 작업은 skopeo copy 명령을 실행합니다.

Skopeo는 원격 컨테이너 이미지 레지스트리를 사용하기 위한 명령줄 툴로, 이미지를 로드하고 실행하는 데몬 또는 기타 인프라가 필요하지 않습니다. skopeo copy 명령은 하나의 원격 레지스트리에서 다른 레지스트리로 이미지를 복사합니다(예: 내부 레지스트리에서 프로덕션 레지스트리로). Skopeo는 사용자가 제공하는 인증 정보를 사용하여 이미지 레지스트리에서 권한 부여를 지원합니다.

skopeo-copy 작업의 사용 예

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: build-deploy-image
spec:
# ...
  tasks:
  - name: copy-image
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: skopeo-copy
      - name: namespace
        value: openshift-pipelines
    params:
    - name: SOURCE_IMAGE_URL
      value: "docker://internal.registry/myimage:latest"
    - name: DESTINATION_IMAGE_URL
      value: "docker://production.registry/myimage:v1.0"
    workspaces:
    - name: output
      workspace: shared-workspace
```

표 3.44. skopeo-copy 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
SOURCE_IMAGE_URL	소스 컨테이너 이미지의 태그를 포함하여 정규화된 이름입니다.	**string**	
DESTINATION_IMAGE_URL	Skopeo가 소스 이미지를 복사하는 대상 이미지의 태그를 포함하여 정규화된 이름입니다.	**string**	
SRC_TLS_VERIFY	소스 레지스트리의 TLS 확인 플래그, 일반적으로 **true**.	**string**	**true**
DEST_TLS_VERIFY	대상 레지스트리의 TLS 확인 플래그, 일반적으로 **true**	**string**	**true**
VERBOSE	디버그 정보를 로그에 출력합니다.	**string**	**false**

표 3.45. skopeo-copy 작업에 지원되는 작업 공간

Workspace	설명
images_url	둘 이상의 이미지를 복사하려면 이 작업 공간을 사용하여 이미지 URL을 제공합니다.

표 3.46. skopeo-copy 작업이 반환됨

결과	유형	설명
SOURCE_DIGEST	**string**	소스 이미지의 SHA256 다이제스트입니다.
DESTINATION_DIGEST	**string**	대상 이미지의 SHA256 다이제스트입니다.

skopeo-copy ClusterTask의 변경 사항

- **모든 매개변수 이름이 대문자로 변경되었습니다.**
- **VERBOSE 매개변수가 추가되었습니다.**
- **작업 공간 이름이 images-url 에서 images_url 로 변경되었습니다.**
- **SOURCE_DIGEST 및 DESTINATION_DIGEST 결과가 추가되었습니다.**

tkn

tkn 작업은 tkn을 사용하여 Tekton 리소스에서 작업을 수행합니다.

tkn 작업 사용 예

```
apiVersion: tekton.dev/v1
kind: PipelineRun
metadata:
 name: tkn-run
spec:
 pipelineSpec:
  tasks:
  - name: tkn-run
    taskRef:
      resolver: cluster
      params:
      - name: kind
        value: task
      - name: name
        value: tkn
      - name: namespace
        value: openshift-pipelines
    params:
    - name: ARGS
```

표 3.47. tkn 작업에 지원되는 매개변수

매개변수	설명	유형	기본값
SCRIPT	실행할 **tkn** CLI 스크립트입니다.	**string**	**tkn $@**
ARGS	실행할 **tkn** CLI 인수입니다.	**array**	**- --help**

표 3.48. tkn 작업에서 지원되는 작업 공간

Workspace	설명
kubeconfig_dir	클러스터에 액세스하기 위한 인증 정보가 포함된 **.kube/config** 파일을 제공할 수 있는 선택적 작업 공간입니다. 이 파일을 작업 공간의 루트에 배치하고 **kubeconfig**로 이름을 지정합니다.

tkn ClusterTask에서 변경 사항

- **TKN_IMAGE** 매개변수가 제거되었습니다.

- 작업 공간 이름이 **kubeconfig** 에서 **kubeconfig_dir** 로 변경되었습니다.

3.6. 추가 리소스

- OpenShift Pipelines와 함께 Tekton Hub 사용

4장. OPENSHIFT PIPELINES에서 수동 승인 사용

파이프라인에 수동 승인 작업을 지정할 수 있습니다. 파이프라인이 이 작업에 도달하면 하나 또는 여러 **OpenShift Container Platform** 사용자의 승인을 일시 중지하고 기다립니다. 사용자가 작업을 승인하는 대신 거부하도록 선택하면 파이프라인이 실패합니다. 수동 승인 게이트 컨트롤러는 이 기능을 제공합니다.

중요

수동 승인 게이트는 기술 프리뷰 기능 전용입니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 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 기술 프리뷰 기능 지원 범위를 참조하십시오.

4.1. 수동 승인 게이트 컨트롤러 활성화

수동 승인 작업을 사용하려면 먼저 수동 승인 게이트 컨트롤러를 활성화해야 합니다.

사전 요구 사항

- 클러스터에 **Red Hat OpenShift Pipelines Operator**를 설치했습니다.
- **oc** 명령줄 유틸리티를 사용하여 클러스터에 로그인되어 있습니다.
- **openshift-pipelines** 네임스페이스에 대한 관리자 권한이 있습니다.

프로세스

1. **ManualApprovalGate CR**(사용자 정의 리소스)에 대해 다음 매니페스트를 사용하여 **manual-approval-gate-cr.yaml** 이라는 파일을 생성합니다.

```
apiVersion: operator.tekton.dev/v1alpha1
kind: ManualApprovalGate
metadata:
```

```
  name: manual-approval-gate
spec:
  targetNamespace: openshift-pipelines
```

2. 다음 명령을 입력하여 **ManualApprovalGate CR**을 적용합니다.

```
$ oc apply -f manual-approval-gate-cr.yaml
```

3. 다음 명령을 입력하여 수동 승인 게이트 컨트롤러가 실행 중인지 확인합니다.

```
$ oc get manualapprovalgates.operator.tekton.dev
```

출력 예

```
NAME                 VERSION    READY   REASON
manual-approval-gate   v0.1.0     True
```

READY 상태가 **True** 인지 확인합니다. **True** 가 아닌 경우 몇 분 기다렸다가 명령을 다시 입력합니다. 컨트롤러가 준비 상태에 도달하는 데 시간이 걸릴 수 있습니다.

4.2. 수동 승인 작업 지정

파이프라인에 수동 승인 작업을 지정할 수 있습니다. 파이프라인 실행 실행이 이 작업에 도달하면 파이프라인 실행이 중지되고 하나 이상의 사용자의 승인을 기다립니다.

사전 요구 사항

- 수동 승인자 게이트 컨트롤러를 활성화했습니다.
- 파이프라인의 **YAML** 사양을 생성하셨습니다.

프로세스

- 다음 예와 같이 파이프라인에 **ApprovalTask** 를 지정합니다.

```
apiVersion: tekton.dev/v1
kind: Pipeline
metadata:
  name: example-manual-approval-pipeline
spec:
  tasks:
# ...
  - name: example-manual-approval-task
    taskRef:
      apiVersion: openshift-pipelines.org/v1alpha1
      kind: ApprovalTask
    params:
    - name: approvers
      value:
      - user1
      - user2
      - user3
    - name: description
      value: Example manual approval task - please approve or reject
    - name: numberOfApprovalsRequired
      value: '2'
    - name: timeout
      value: '60m'
# ...
```

표 4.1. 수동 승인 작업의 매개변수

매개변수	유형	설명
승인자	array	작업을 승인할 수 있는 OpenShift Container Platform 사용자입니다.
description	string	선택 사항: 승인 작업에 대한 설명입니다. OpenShift Pipelines는 작업을 승인하거나 거부할 수 있는 사용자에게 설명을 표시합니다.
numberOfApprovalsRequired	string	작업에 필요한 다른 사용자의 승인 수입니다.
timeout	string	선택 사항: 승인의 제한 시간입니다. 이 기간 동안 작업에서 구성된 승인 수를 수신하지 못하면 파이프라인 실행이 실패합니다. 기본 제한 시간은 1시간입니다.

4.3. 수동 승인 작업 승인

승인 작업이 포함된 파이프라인을 실행하고 실행이 승인 작업에 도달하면 파이프라인 실행이 일시 중지되고 사용자 승인 또는 거부를 기다립니다.

사용자는 웹 콘솔 또는 **opc** 명령줄 유틸리티를 사용하여 작업을 승인하거나 거부할 수 있습니다.

작업에 구성된 승인자 중 하나가 작업을 거부하면 파이프라인 실행이 실패합니다.

한 사용자가 작업을 승인하지만 구성된 승인 수에 도달하지 못한 경우 동일한 사용자가 작업을 거부하도록 변경할 수 있으며 파이프라인 실행이 실패합니다.

4.3.1. 웹 콘솔을 사용하여 수동 승인 작업 승인

OpenShift Container Platform 웹 콘솔을 사용하여 수동 승인 작업을 승인하거나 거부할 수 있습니다.

수동 승인 작업에서 승인자로 나열되어 파이프라인 실행이 이 작업에 도달하면 웹 콘솔에 알림이 표시됩니다. 승인이 필요한 작업 목록을 보고 이러한 작업을 승인하거나 거부할 수 있습니다.

사전 요구 사항

- **OpenShift Pipelines** 콘솔 플러그인을 활성화했습니다.

프로세스

1. 다음 작업 중 하나를 완료하여 승인할 수 있는 작업 목록을 확인합니다.
   - 승인이 필요한 작업에 대한 알림이 표시되면 이 알림에서 승인으로 이동 탭을 클릭합니다.
   - 관리자 화면 메뉴에서 파이프라인 → 파이프라인을 선택한 다음 승인 탭을 클릭합니다.
   - 개발자 화면 메뉴에서 파이프라인을 선택한 다음 승인 탭을 클릭합니다.
   -

PipelineRun 세부 정보 창의 세부 정보 탭에서 수동 승인 작업을 나타내는 사각형을 클릭합니다. 목록에는 이 작업의 승인만 표시됩니다.

- **PipelineRun** 세부 정보 창에서 **ApprovalTasks** 탭을 클릭합니다. 목록에는 이 파이프라인 실행에 대한 승인만 표시됩니다.

2. 승인 작업 목록에서 승인할 작업을 나타내는 줄에서  아이콘을 클릭한 다음 다음 옵션 중 하나를 선택합니다.

- 작업을 승인하려면 **승인** 을 선택합니다.
- 작업을 거부하려면 **거부**를 선택합니다.

3. **Reason** 필드에 메시지를 입력합니다.

4. **Submit** 을 클릭합니다.

추가 리소스

- OpenShift Pipelines 콘솔 플러그인 활성화

4.3.2. 명령줄을 사용하여 수동 승인 작업 승인

opc 명령줄 유틸리티를 사용하여 수동 승인 작업을 승인하거나 거부할 수 있습니다. 승인자가 되는 작업 목록을 보고 승인 보류 중인 작업을 승인하거나 거부할 수 있습니다.

사전 요구 사항

- **opc** 명령줄 유틸리티를 다운로드하여 설치했습니다. 이 유틸리티는 **tkn** 명령줄 유틸리티와 동일한 패키지에서 사용할 수 있습니다.
- **oc** 명령줄 유틸리티를 사용하여 클러스터에 로그인되어 있습니다.

프로세스

1. 다음 명령을 입력하여 승인자로 나열된 수동 승인 작업 목록을 확인합니다.

```
$ opc approvaltask list
```

출력 예

```
NAME                                        NumberOfApprovalsRequired   PendingApprovals
Rejected   STATUS
manual-approval-pipeline-01w6e1-task-2   2                    0             0
Approved
manual-approval-pipeline-6ywv82-task-2   2                    2             0          Rejected
manual-approval-pipeline-90gyki-task-2   2                    2             0          Pending
manual-approval-pipeline-jyrkb3-task-2   2                    1             1          Rejected
```

2. 선택 사항: 이름, 네임스페이스, 파이프라인 실행 이름, 승인자 목록, 현재 상태를 포함한 수동 승인 작업에 대한 정보를 보려면 다음 명령을 입력합니다.

```
$ opc approvaltask describe <approval_task_name>
```

3. 필요에 따라 수동 승인 작업을 승인하거나 거부합니다.

- 수동 승인 작업을 승인하려면 다음 명령을 입력합니다.

```
$ opc approvaltask approve <approval_task_name>
```

필요한 경우 **-m** 매개변수를 사용하여 승인 메시지를 지정할 수 있습니다.

```
$ opc approvaltask approve <approval_task_name> -m <message>
```

- 수동 승인 작업을 거부하려면 다음 명령을 입력합니다.

```
$ opc approvaltask reject <approval_task_name>
```

선택적으로 **-m** 매개변수를 사용하여 거부에 대한 메시지를 지정할 수 있습니다.

```
$ opc approvaltask reject <approval_task_name> -m <message>
```

추가 리소스

- tkn 설치

5장. 파이프라인에서 RED HAT 인타이틀먼트 사용

RHEL(Red Hat Enterprise Linux) 인타이틀먼트가 있는 경우 이러한 인타이틀먼트를 사용하여 파이프라인에 컨테이너 이미지를 빌드할 수 있습니다.

Cryostat Operator는 SCA(Simple Common Access)에서 이 Operator로 가져온 후 인타이틀먼트를 자동으로 관리합니다. 이 Operator는 openshift-config-managed 네임스페이스에 etc-pki-entitlement 라는 시크릿을 제공합니다.

다음 두 가지 방법 중 하나로 파이프라인에서 Red Hat 인타이틀먼트를 사용할 수 있습니다.

- 보안을 파이프라인의 네임스페이스에 수동으로 복사합니다. 이 방법은 제한된 수의 파이프라인 네임스페이스가 있는 경우 덜 복잡합니다.
- CSI(Share Resources Container Storage Interface) Driver Operator를 사용하여 네임스페이스 간에 보안을 자동으로 공유합니다.

5.1. 사전 요구 사항

- oc 명령줄 툴을 사용하여 OpenShift Container Platform 클러스터에 로그인했습니다.
- OpenShift Container Platform 클러스터에서 Insights Operator 기능을 활성화했습니다. 공유 리소스 CSI 드라이버 Operator를 사용하여 네임스페이스 간에 보안을 공유하려면 공유 리소스 CSI 드라이버도 활성화해야 합니다. Insights Operator 및 공유 리소스 CSI 드라이버를 포함한 기능 활성화에 대한 자세한 내용은 기능 게이트를 사용하여 기능 활성화를 참조하십시오.

참고

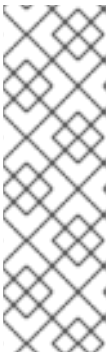
Insights Operator를 활성화한 후 클러스터가 이 Operator로 모든 노드를 업데이트하는지 확인하기 위해 잠시 기다려야 합니다. 다음 명령을 입력하여 모든 노드의 상태를 모니터링할 수 있습니다.

```
$ oc get nodes -w
```

Insights Operator가 활성 상태인지 확인하려면 다음 명령을 입력하여 insights-operator pod가 openshift-insights 네임스페이스에서 실행 중인지 확인합니다.

```
$ oc get pods -n openshift-insights
```

- Insights Operator로 Red Hat 인타이틀먼트를 가져올 수 있도록 구성했습니다. 자격을 가져오는 방법에 대한 자세한 내용은 Insights Operator를 사용하여 간단한 콘텐츠 액세스 인타이틀먼트 가져오기 를 참조하십시오.



참고

Insights Operator에서 인타이틀먼트를 활성 상태인지 확인하려면 다음 명령을 입력하여 etc-pki-entitlement 시크릿이 openshift-config-managed 네임스페이스에 있는지 확인합니다.

```
$ oc get secret etc-pki-entitlement -n openshift-config-managed
```

5.2. ETC-PKI-ENTITLEMENT 시크릿을 수동으로 복사하여 RED HAT 인타이틀먼트 사용

etc-pki-entitlement 보안을 openshift-config-managed 네임스페이스에서 파이프라인의 네임스페이스로 복사할 수 있습니다. 그런 다음 Buildah 작업에 이 시크릿을 사용하도록 파이프라인을 구성할 수 있습니다.

사전 요구 사항

- 시스템에 jq 패키지를 설치했습니다. 이 패키지는 RHEL(Red Hat Enterprise Linux)에서 사용할 수 있습니다.

프로세스

1.

다음 명령을 실행하여 **openshift-config-managed** 네임스페이스의 **etc-pki-entitlement** 보안을 파이프라인의 네임스페이스로 복사합니다.

```
$ oc get secret etc-pki-entitlement -n openshift-config-managed -o json | \
  jq 'del(.metadata.resourceVersion)' | jq 'del(.metadata.creationTimestamp)' | \
  jq 'del(.metadata.uid)' | jq 'del(.metadata.namespace)' | \
  oc -n <pipeline_namespace> create -f - (1)
```

(1)

& lt;pipeline_namespace& gt;를 파이프라인의 네임스페이스로 바꿉니다.

2.

Buildah 작업 정의에서 **openshift-pipelines** 네임스페이스에 제공된 **buildah** 작업 또는 이 작업의 사본을 사용하고 다음 예와 같이 **rhel-entitlement** 작업 공간을 정의합니다.

3.

Buildah 작업을 실행하는 작업 실행 또는 파이프라인 실행에서 다음 예와 같이 **etc-pki-entitlement** 시크릿을 **rhel-entitlement** 작업 공간에 할당합니다.

Red Hat 인타이틀먼트를 사용하는 파이프라인 및 작업 정의를 포함한 파이프라인 실행 정의의 예

```
apiVersion: tekton.dev/v1
kind: PipelineRun
metadata:
  name: buildah-pr-test
spec:
  workspaces:
    - name: shared-workspace
      volumeClaimTemplate:
        spec:
          accessModes:
            - ReadWriteOnce
          resources:
            requests:
              storage: 1Gi
    - name: dockerconfig
      secret:
        secretName: regred
    - name: rhel-entitlement (1)
      secret:
        secretName: etc-pki-entitlement
  pipelineSpec:
    workspaces:
      - name: shared-workspace
      - name: dockerconfig
      - name: rhel-entitlement (2)
    tasks:
```

```
# ...
    - name: buildah
      taskRef:
        resolver: cluster
        params:
        - name: kind
          value: task
        - name: name
          value: buildah
        - name: namespace
          value: openshift-pipelines
      workspaces:
      - name: source
        workspace: shared-workspace
      - name: dockerconfig
        workspace: dockerconfig
      - name: rhel-entitlement  3
        workspace: rhel-entitlement
      params:
      - name: IMAGE
        value: <image_where_you_want_to_push>
```

1

파이프라인 실행의 **rhel-entitlement** 작업 공간에 대한 정의로 **etc-pki-entitlement** 시크릿을 작업 공간에 할당합니다.

2

파이프라인 정의에서 **rhel-entitlement** 작업 공간 정의

3

작업 정의에서 **rhel-entitlement** 작업 공간 정의

5.3. SHARED RESOURCES CSI 드라이버 OPERATOR를 사용하여 시크릿을 공유하여 RED HAT 인타이틀먼트 사용

CSI(Share Resources Container Storage Interface) Driver Operator를 사용하여 **openshift-config-managed** 네임스페이스에서 다른 네임스페이스로 **etc-pki-entitlement** 시크릿 공유를 설정할 수 있습니다. 그런 다음 **Buildah** 작업에 이 시크릿을 사용하도록 파이프라인을 구성할 수 있습니다.

사전 요구 사항

-

oc 명령줄 유틸리티를 클러스터 관리자 권한이 있는 사용자로 사용하여 **OpenShift Container Platform** 클러스터에 로그인되어 있습니다.

- **OpenShift Container Platform** 클러스터에서 **Shared Resources CSI Driver Operator**를 활성화했습니다.

프로세스

1. 다음 명령을 실행하여 **etc-pki-entitlement** 시크릿을 공유할 **SharedSecret CR**(사용자 정의 리소스)을 생성합니다.

```
$ oc apply -f - <<EOF
apiVersion: sharedresource.openshift.io/v1alpha1
kind: SharedSecret
metadata:
  name: shared-rhel-entitlement
spec:
  secretRef:
    name: etc-pki-entitlement
    namespace: openshift-config-managed
EOF
```

2. 다음 명령을 실행하여 공유 보안에 대한 액세스를 허용하는 **RBAC** 역할을 생성합니다.

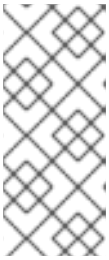
```
$ oc apply -f - <<EOF
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: shared-resource-rhel-entitlement
  namespace: <pipeline_namespace> 1
rules:
  - apiGroups:
      - sharedresource.openshift.io
    resources:
      - sharedsecrets
    resourceNames:
      - shared-rhel-entitlement
    verbs:
      - use
EOF
```

1 & lt;pipeline_namespace& gt;를 파이프라인의 네임스페이스로 바꿉니다.

3. 다음 명령을 실행하여 파이프라인 서비스 계정에 역할을 할당합니다.

```
$ oc create rolebinding shared-resource-rhel-entitlement --role=shared-shared-resource-rhel-entitlement \
  --serviceaccount=<pipeline-namespace>:pipeline 1
```

1 & lt;pipeline-namespace& gt;를 파이프라인의 네임스페이스로 바꿉니다.



참고

OpenShift Pipelines의 기본 서비스 계정을 변경했거나 파이프라인 실행 또는 작업 실행에 사용자 정의 서비스 계정을 정의하는 경우 파이프라인 계정 대신 이 계정에 역할을 할당합니다.

4. Buildah 작업 정의에서 openshift-pipelines 네임스페이스에 제공된 buildah 작업 또는 이 작업의 사본을 사용하고 다음 예와 같이 rhel-entitlement 작업 공간을 정의합니다.

5. Buildah 작업을 실행하는 작업 실행 또는 파이프라인 실행에서 다음 예와 같이 rhel-entitlement 작업 공간에 공유 시크릿을 할당합니다.

Red Hat 인타이틀먼트를 사용하는 파이프라인 및 작업 정의를 포함한 파이프라인 실행 정의의 예

```
apiVersion: tekton.dev/v1
kind: PipelineRun
metadata:
  name: buildah-pr-test-csi
spec:
  workspaces:
    - name: shared-workspace
      volumeClaimTemplate:
        spec:
          accessModes:
          - ReadWriteOnce
          resources:
            requests:
              storage: 1Gi
    - name: dockerconfig
      secret:
        secretName: regred
    - name: rhel-entitlement 1
```

```
      csi:
        readOnly: true
        driver: csi.sharedresource.openshift.io
        volumeAttributes:
          sharedSecret: shared-rhel-entitlement
    pipelineSpec:
      workspaces:
      - name: shared-workspace
      - name: dockerconfig
      - name: rhel-entitlement 2
      tasks:
# ...
      - name: buildah
        taskRef:
          resolver: cluster
          params:
          - name: kind
            value: task
          - name: name
            value: buildah
          - name: namespace
            value: openshift-pipelines
        workspaces:
        - name: source
          workspace: shared-workspace
        - name: dockerconfig
          workspace: dockerconfig
        - name: rhel-entitlement 3
          workspace: rhel-entitlement
        params:
        - name: IMAGE
          value: <image_where_you_want_to_push>
```

1

파이프라인 실행의 **rhel-entitlement** 작업 공간에 대한 정의로 **shared-rhel-entitlement CSI** 공유 시크릿을 작업 공간에 할당합니다.

2

파이프라인 정의에서 **rhel-entitlement** 작업 공간 정의

3

작업 정의에서 **rhel-entitlement** 작업 공간 정의

5.4. 추가 리소스

- 간단한 콘텐츠 액세스
- Insights Operator 사용
- Insights Operator를 사용하여 간단한 콘텐츠 액세스 인타이틀먼트 가져오기
- 공유 리소스 CSI Driver Operator
- OpenShift Pipelines의 기본 서비스 계정 변경

6장. 버전이 지정되지 않은 클러스터 작업 관리

클러스터 관리자는 **Red Hat OpenShift Pipelines Operator**를 설치하면 버전이 지정된 클러스터 작업 **(VCT)** 및 *버전이 아닌 클러스터 작업 (NVCT)이라는 각 기본 클러스터 작업*의 변형을 생성합니다. 예를 들어 **Red Hat OpenShift Pipelines Operator v1.7**을 설치하면 **buildah-1-7-0 CryostatT** 및 **buildah NVCT**가 생성됩니다.

NVCT 및 **CryostatT** 모두 **params, Workspaces** 및 단계를 포함한 동일한 메타데이터, 동작 및 사양을 갖습니다. 그러나 **Operator**를 비활성화하거나 업그레이드할 때 다르게 작동합니다.

중요

Red Hat OpenShift Pipelines 1.10에서 **ClusterTask** 기능은 더 이상 사용되지 않으며 향후 릴리스에서 제거될 예정입니다.

6.1. 버전이 아닌 클러스터 작업과 버전이 지정되지 않은 클러스터 작업의 차이점

버전이 아닌 클러스터 작업에는 다른 이름 지정 규칙이 있습니다. 또한 **Red Hat OpenShift Pipelines Operator**는 이를 다르게 업그레이드합니다.

표 6.1. 버전이 아닌 클러스터 작업과 버전이 지정되지 않은 클러스터 작업의 차이점

	지원되지 않는 클러스터 작업	버전이 지정된 클러스터 작업
nomenclature	NVCT에는 클러스터 작업의 이름만 포함됩니다. 예를 들어 Operator v1.7과 함께 설치된 Buildah의 NVCT 이름은 **buildah** 입니다.	CryostatT에는 클러스터 작업의 이름이 포함되고 그 뒤에 버전이 접미사로 표시됩니다. 예를 들어 Operator v1.7과 함께 설치된 Buildah의 이름은 **buildah-1-7-0** 입니다.
업그레이드	Operator를 업그레이드하면 최신 변경 사항으로 버전이 지정되지 않은 클러스터 작업을 업데이트합니다. NVCT의 이름은 변경되지 않습니다.	Operator를 업그레이드하면 최신 버전이 설치되고 이전 버전이 유지됩니다. 최신 버전은 업그레이드된 Operator에 해당합니다. 예를 들어 Operator 1.7을 설치하면 **buildah-1-7-0** 이 설치되고**buildah-1-6-0** 이 유지됩니다.

6.2. 버전이 아닌 클러스터 작업 및 버전이 지정된 클러스터 작업의 이점 및 단점

프로덕션 환경에서 버전 이외의 클러스터 작업을 표준으로 채택하기 전에 클러스터 관리자는 해당 이점과 단점을 고려할 수 있습니다.

표 6.2. 버전이 아닌 클러스터 작업 및 버전이 지정된 클러스터 작업의 이점 및 단점

클러스터 작업	이점	단점
버전이 없는 클러스터 작업(NVCT)	• 최신 업데이트 및 버그 수정으로 파이프라인을 배포하려면 NVCT를 사용합니다. • Operator를 업그레이드하면 버전이 지정되지 않은 클러스터 작업을 업그레이드하여 여러 버전의 클러스터 작업보다 적은 리소스를 사용합니다.	NVCT를 사용하는 파이프라인을 배포하는 경우 자동으로 업그레이드된 클러스터 작업이 이전 버전과 호환되지 않는 경우 Operator 업그레이드 후 중단될 수 있습니다.
버전이 지정된 클러스터 작업(VCT)	• 프로덕션에서 안정적인 파이프라인을 선호하는 경우 CryostatT를 사용합니다. • 이전 버전은 이후 버전의 클러스터 작업이 설치된 후에도 클러스터에 유지됩니다. 이전 클러스터 작업을 계속 사용할 수 있습니다.	• 이전 버전의 클러스터 작업을 계속 사용하는 경우 최신 기능 및 중요한 보안 업데이트가 누락될 수 있습니다. • 작동하지 않는 이전 버전의 클러스터 작업에서는 클러스터 리소스를 사용합니다. • * 업그레이드 후 Operator는 이전 CryostatT를 관리할 수 없습니다. **oc delete clustertask** 명령을 사용하여 이전 CryostatT를 수동으로 삭제할 수 있지만 복원할 수는 없습니다.

6.3. 버전이 지정되지 않은 클러스터 작업 비활성화

클러스터 관리자는 OpenShift Pipelines Operator가 설치한 클러스터 작업을 비활성화할 수 있습니다.

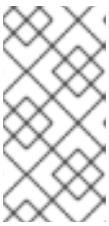
프로세스

1. **버전이 아닌 모든 클러스터 작업 및 최신 버전이 지정된 클러스터 작업을 삭제하려면 TektonConfig CRD(사용자 정의 리소스 정의)를 편집하고 spec.addon.params 에서 clusterTasks 매개변수를 false 로 설정합니다.**

TektonConfig CR의 예

```
apiVersion: operator.tekton.dev/v1alpha1
kind: TektonConfig
metadata:
  name: config
spec:
  params:
  - name: createRbacResource
    value: "false"
  profile: all
  targetNamespace: openshift-pipelines
  addon:
    params:
    - name: clusterTasks
      value: "false"
...
```

클러스터 작업을 비활성화할 때 **Operator**는 버전이 아닌 클러스터 작업을 모두 제거하고 클러스터에서 최신 버전의 클러스터 작업만 제거합니다.



참고

클러스터 작업을 다시 활성화하면 버전이 지정되지 않은 클러스터 작업이 설치됩니다.

2. 선택 사항: 이전 버전의 버전이 지정된 클러스터 작업을 삭제하려면 다음 방법 중 하나를 사용합니다.

   a. 이전 버전의 개별 클러스터 작업을 삭제하려면 **oc delete clustertask** 명령 다음에 버전이 지정된 클러스터 작업 이름을 사용합니다. 예를 들면 다음과 같습니다.

   ```
   $ oc delete clustertask buildah-1-6-0
   ```

   b. 이전 버전의 **Operator**에서 생성한 버전이 지정된 모든 클러스터 작업을 삭제하려면 해당 설치 프로그램 세트를 삭제할 수 있습니다. 예를 들면 다음과 같습니다.

   ```
   $ oc delete tektoninstallerset versioned-clustertask-1-6-k98as
   ```

경고

이전 버전의 클러스터 작업을 삭제하는 경우 복원할 수 없습니다. 현재 버전의 Operator 가 생성한 버전 및 버전이 아닌 클러스터 작업만 복원할 수 있습니다.