

Lab 4. Использование OpenShift Pipelines

Предварительные требования.

Установленный кластер OpenShift

Можно использовать кластер, развернутый на предыдущем курсе, либо следовать инструкции по установке.

Возможно использовать CRC

Так же потребуются NFS для предоставления хранилища для постоянных томов. Можно настроить службу на хосту Linux с ролью DNS развернутой для нужд кластера. Для этого следуйте инструкции:

Установите VM с ОС Linux (Debian/Ubuntu или RHEL/Centos/Fedora)

В случае Debian/Ubuntu выполните команды:

```
sudo apt update -y && sudo apt install -y nfs-kernel-server
```

Добавьте к VM второй виртуальный диск: тип – динамический, размер 512G

Создайте на нем 2 раздела по 256 G

Отформатируйте:

```
mkfs.ext4 -L vol1 /dev/sdb1
```

```
mkfs.ext4 -L vol2 /dev/sdb2
```

Создайте директории:

```
mkdir /var/vol1
```

```
mkdir /var/vol2
```

Предоставьте права доступа к ним:

```
chmod -R 777 /var/vol1
```

```
chmod -R 777 /var/vol2
```

Добавьте записи монтирования в /etc/fstab

```
sudo nano /etc/fstab
```

В конец файла добавьте строки:

```
LABEL=vol1    /var/vol1    ext4 defaults    0    0
```

```
LABEL=vol2    /var/vol2    ext4 defaults    0    0
```

Выполните команду для монтирования созданных файловых систем

```
sudo mount -a
```

Настройте nfs:

```
sudo nano /etc/exports
```

Building Kubernetes Applications

/var/vol1 -rw,async,root_squash,no_wdelay,mp адрес_сети/маска

/var/vol2 -rw,async,root_squash,no_wdelay,mp адрес_сети/маска

Building Kubernetes Applications

На хостовой машине установите следующее ПО (в случае Windows следующим образом):

Установка клиента Tekton в Powershell:

```
choco install tektoncd-cli --confirm
```

Установка kustomize в Powershell:

```
choco install kustomize
```

Установка helm в Powershell:

```
irm get.scoop.sh | iex
```

```
scoop install helm
```

С выпуском OpenShift Pipelines based on Tekton, стратегия построения конвейеров Jenkins устарела. Пользователям следует либо использовать файлы Jenkins непосредственно в Jenkins, либо использовать облачную CI/CD с Openshift Pipelines. Попробуйте учебник OpenShift Pipelines

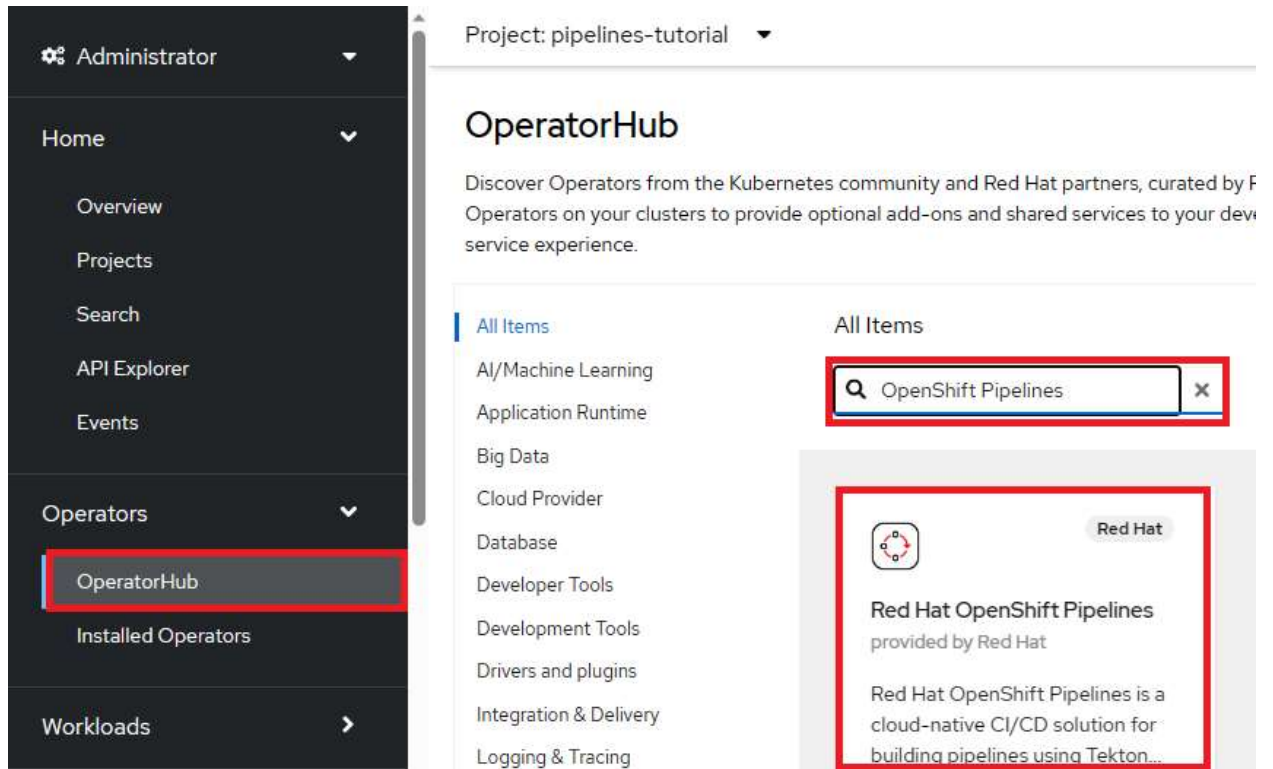
OpenShift Pipelines

OpenShift Pipelines — это облачное решение для непрерывной интеграции и доставки (CI/CD) для построения конвейеров с использованием Tekton.

Building Kubernetes Applications

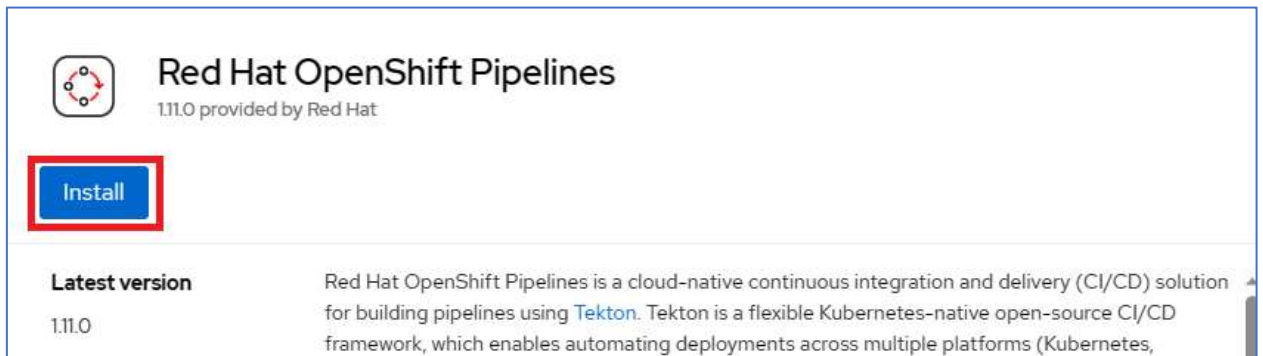
Войдите в веб-консоль в перспективу администратора. Перейдите в Operators -> OperatorHub

Введите в поиск OpenShift Pipelines

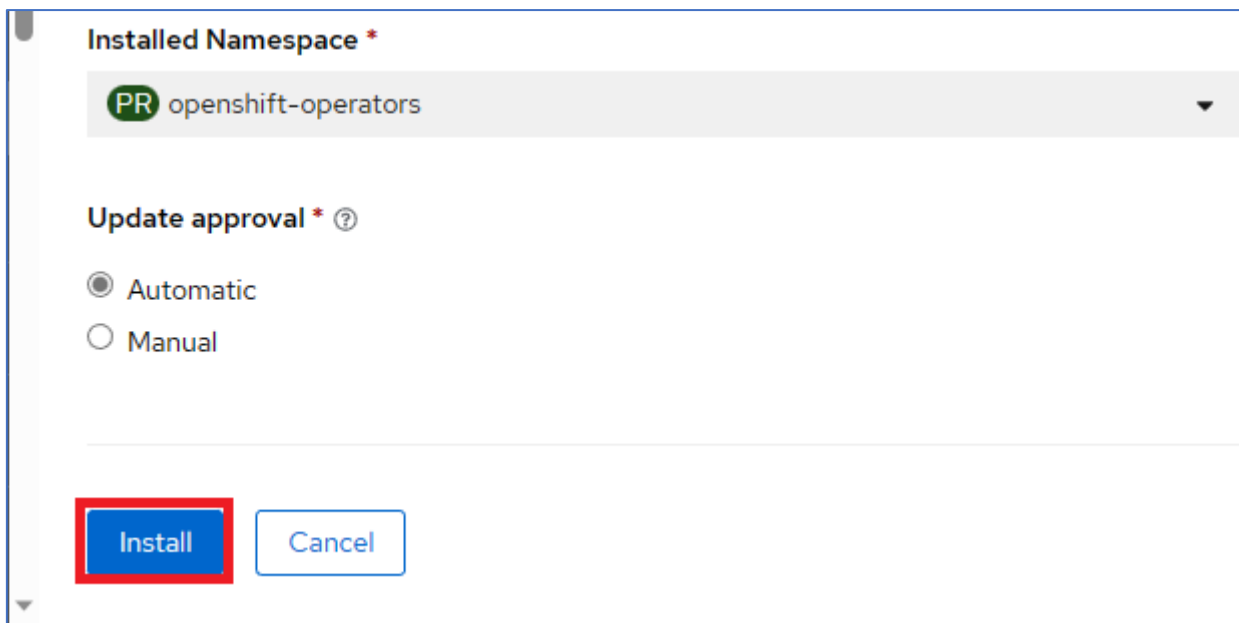


Нажмите на плитке Red Hat OpenShift Pipelines

В появившемся окне нажмите кнопку **Install**



И в открывшемся окне оставьте настройки по умолчанию и нажмите внизу кнопку **Install**



Installed Namespace *

PR openshift-operators

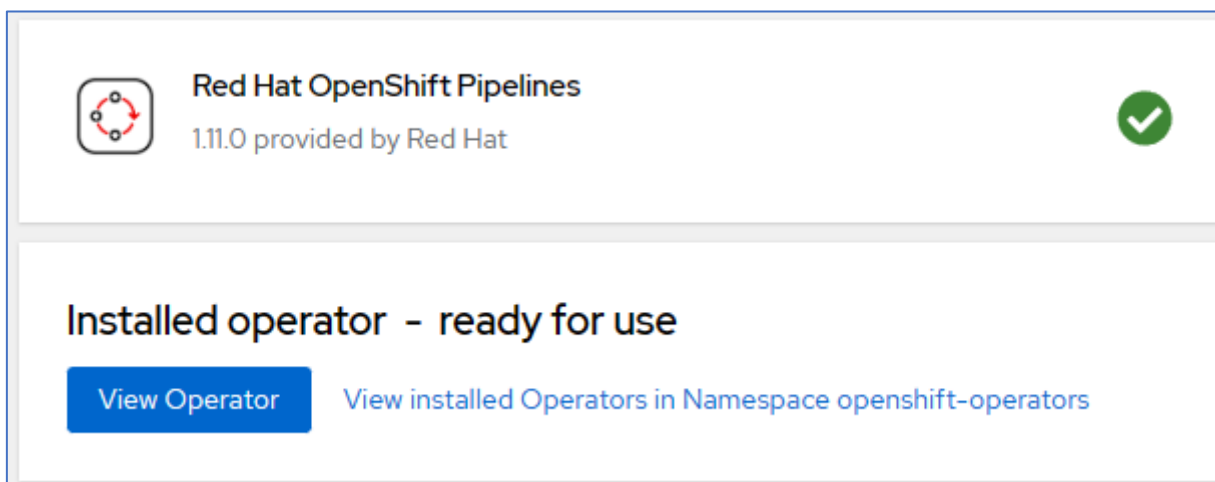
Update approval * ?



☒ Automatic

☐ Manual

Install **Cancel**

Дождитесь установки оператора.

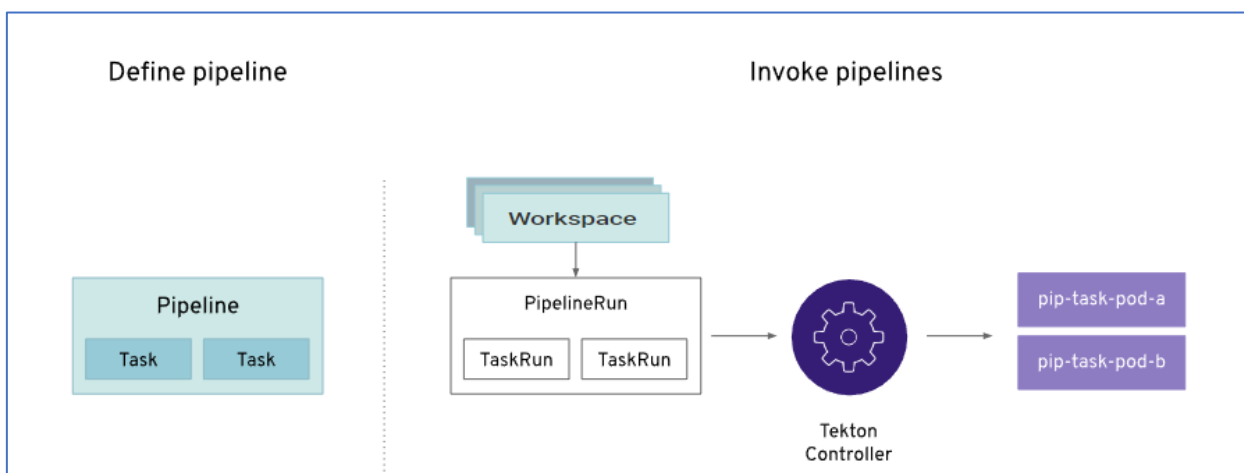


 **Red Hat OpenShift Pipelines** 1.11.0 provided by Red Hat 

Installed operator - ready for use

[View Operator](#) [View installed Operators in Namespace openshift-operators](#)

Tekton — это гибкая, нативная для Kubernetes платформа CI/CD с открытым исходным кодом, которая позволяет автоматизировать развертывание.



OpenShift Pipelines предоставляется как надстройка поверх OpenShift, которую можно установить с помощью оператора, доступного в OpenShift OperatorHub.

Building Kubernetes Applications

Workspace может быть secret, pvc, config или emptyDir.

Создадим два постоянных тома.

Можно в веб-консоли, можно через oc

Создайте файлы pv1.yml и pv2.yml

Файл pv1.yml следующего содержания:

apiVersion: v1

kind: PersistentVolume

metadata:

name: nfs-vol1

spec:

capacity:

storage: **10Gi**

accessModes:

- **ReadWriteOnce**

persistentVolumeReclaimPolicy: Retain

nfs:

server: 192.168.0.2

path: /var/vol1

И второй файл – pv2.yml:

apiVersion: v1

kind: PersistentVolume

metadata:

name: nfs-vol2

spec:

capacity:

storage: **10Gi**

accessModes:

- **ReadWriteOnce**

persistentVolumeReclaimPolicy: Retain

nfs:

server: 192.168.0.2

path: /var/vol2

Building Kubernetes Applications

Создайте 2 постоянных тома:

```
oc apply -f pv1.yml
```

```
oc apply -f pv2.yml
```

Развертывание примера приложения

Создайте проект для примера приложения, которое вы будете использовать в этом руководстве:

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

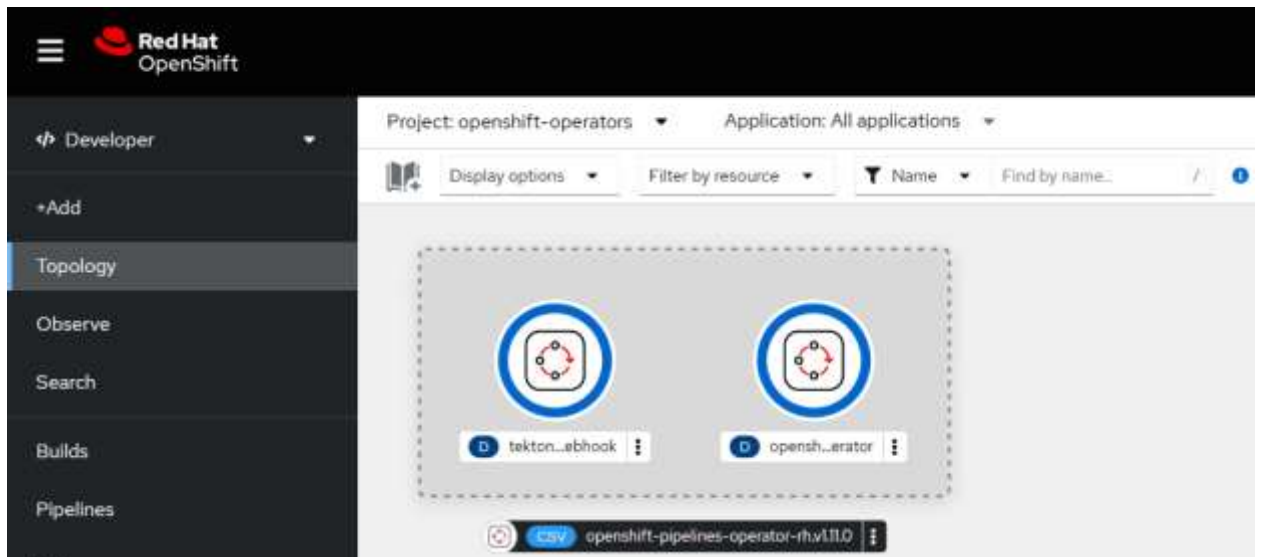
OpenShift Pipelines автоматически добавляет и настраивает ServiceAccount для pipeline, имеющее достаточные разрешения для создания и отправки образа.

Выполните следующую команду, чтобы увидеть учетную запись службы pipeline:

```
oc get serviceaccount pipeline
```

В этом примере будем использовать простое приложение, которое имеет внешний и внутренний интерфейсы.

Перейдите в веб-консоль в перспективу разработчика и далее в Topology:



Install Tasks

Задачи состоят из ряда шагов, которые выполняются последовательно. Задачи выполняются/запускаются путем создания TaskRuns. TaskRun планирует Pod. Каждый шаг выполняется в отдельном контейнере внутри одного пода. Они также могут иметь входы и выходы для взаимодействия с другими задачами в конвейере.

Building Kubernetes Applications

Установите из репозитория с помощью `oc` или `kubectl` задачи `apply-manifests` и `update-deployment`, которые вам понадобятся для создания конвейера в следующем разделе:

```
oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/master/01_pipeline/01_apply_manifest_task.yaml
```

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



```
Command Prompt
C:\Users\Michael>oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/master/01_pipeline/01_apply_manifest_task.yaml
task.tekton.dev/apply-manifests created

C:\Users\Michael>oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/master/01_pipeline/02_update_deployment_task.yaml
task.tekton.dev/update-deployment created
```

Можно увидеть созданные задачи с помощью Tekton CLI :

```
tkn task ls
```

Мы будем использовать `buildahclusterTasks`, который устанавливается вместе с Operator. Оператор устанавливает несколько `ClusterTask`, которые вы можете видеть.

```
tkn clustertasks ls
```

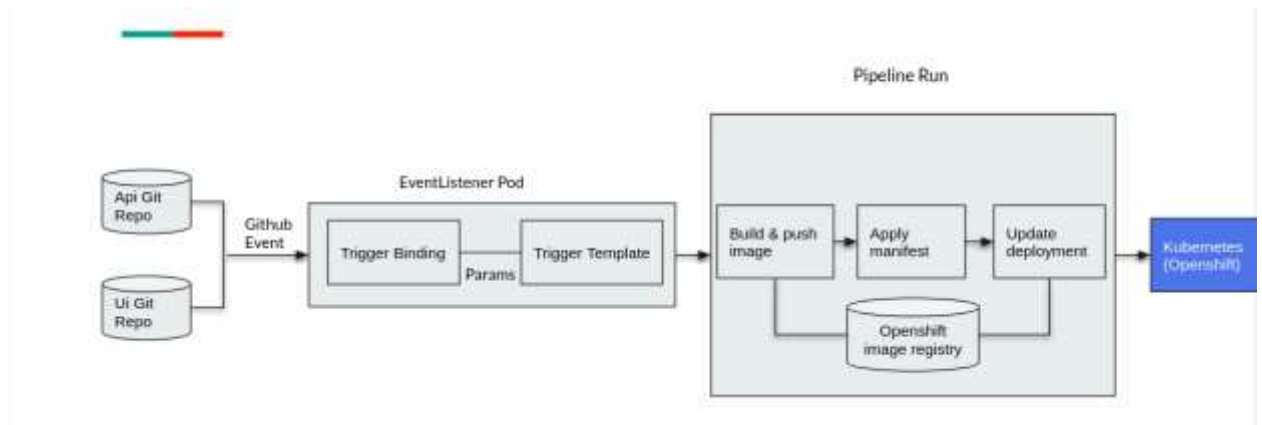
NAME	DESCRIPTION	AGE
buildah		1 day ago
buildah-v0-14-3		1 day ago
git-clone		1 day ago
s2i-php		1 day ago
tkn		1 day ago

Создадим конвейер

Конвейер определяет ряд задач, которые должны быть выполнены, и то, как они взаимодействуют друг с другом через свои входы и выходы.

В этом примере создадим конвейер, который берет исходный код приложения из GitHub, а затем создает и разворачивает его в OpenShift.

Building Kubernetes Applications



Select Command Prompt

```
C:\Users\Michael\samples>oc create -f sample-pipeline.yml
pipeline.tekton.dev/build-and-deploy created

C:\Users\Michael\samples>
```

Создайте файл:

apiVersion: tekton.dev/v1beta1

kind: Pipeline

metadata:

name: build-and-deploy

spec:

workspaces:

- name: shared-workspace

params:

- name: deployment-name

type: string

description: name of the deployment to be patched

- name: git-url

type: string

description: url of the git repo for the code of deployment

- name: git-revision

type: string

description: revision to be used from repo of the code for deployment

default: master

Building Kubernetes Applications

```
- name: IMAGE
  type: string
  description: image to be build from the code
tasks:
- name: fetch-repository
  taskRef:
    name: git-clone
    kind: ClusterTask
  workspaces:
    - name: output
      workspace: shared-workspace
  params:
    - name: url
      value: ${params.git-url}
    - name: subdirectory
      value: ""
    - name: deleteExisting
      value: "true"
    - name: revision
      value: ${params.git-revision}
- name: build-image
  taskRef:
    name: buildah
    kind: ClusterTask
  params:
    - name: IMAGE
      value: ${params.IMAGE}
  workspaces:
    - name: source
      workspace: shared-workspace
  runAfter:
    - fetch-repository
```

Building Kubernetes Applications

```
- 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
```

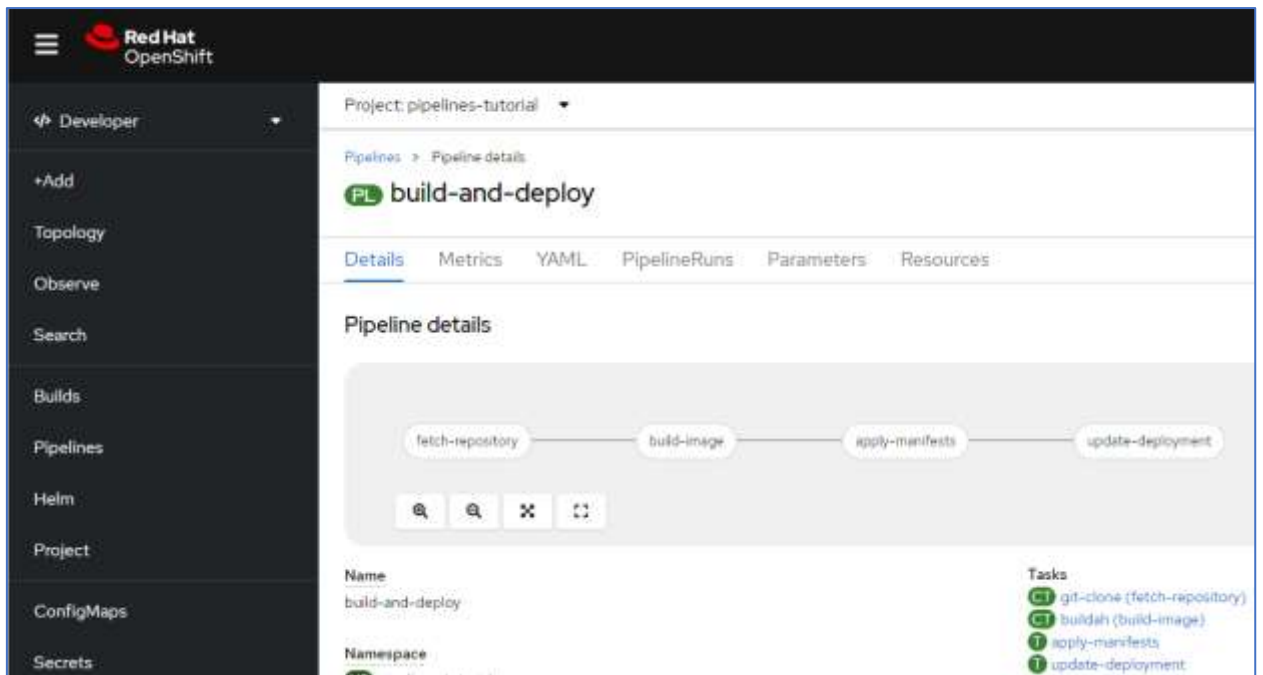
Создайте pipeline:

```
oc create -f sample-pipeline.yml
```

Или можно так:

```
oc create -f https://raw.githubusercontent.com/openshift/pipelines-tutorial/master/01_pipeline/
04_pipeline.yaml
```

После развертывания конвейеров вы сможете визуализировать поток конвейера в веб-консоли OpenShift, переключившись на проекцию «Разработчик» веб-консоли OpenShift. выберите вкладку конвейера, выберите проект как pipelines-tutorial и нажмите на конвейер build-and-deploy



Этот конвейер помогает вам создавать и разворачивать серверную/внешнюю часть, настраивая правильные ресурсы для конвейера.

Шаги конвейера:

1. Клонировать исходный код приложения из репозитория git, ссылаясь на (git-url и git-revisionparam)
2. Создает образ контейнера приложения, используя buildah кластерную задачу, которая использует Buildah для создания образа.
3. Образ приложения помещается в реестр образов по ссылке (imageparam)
4. Новый образ приложения разворачивается в OpenShift с помощью задач apply-manifests и update-deployment.

Вы могли заметить, что нет никаких ссылок на репозиторий git или реестр образов, куда он будет помещен в конвейере. Это связано с тем, что конвейеры в Tekton спроектированы так, чтобы быть универсальными и повторно использоваться в различных средах и на разных этапах жизненного цикла приложения.

При запуске конвейера вы можете предоставить различные репозитории git и реестры образов, которые будут использоваться во время выполнения конвейера.

Проверьте, что пайплайн появился:

```
tkn pipeline ls
```

Теперь, когда конвейер создан, вы можете активировать его для выполнения задач, указанных в конвейере.

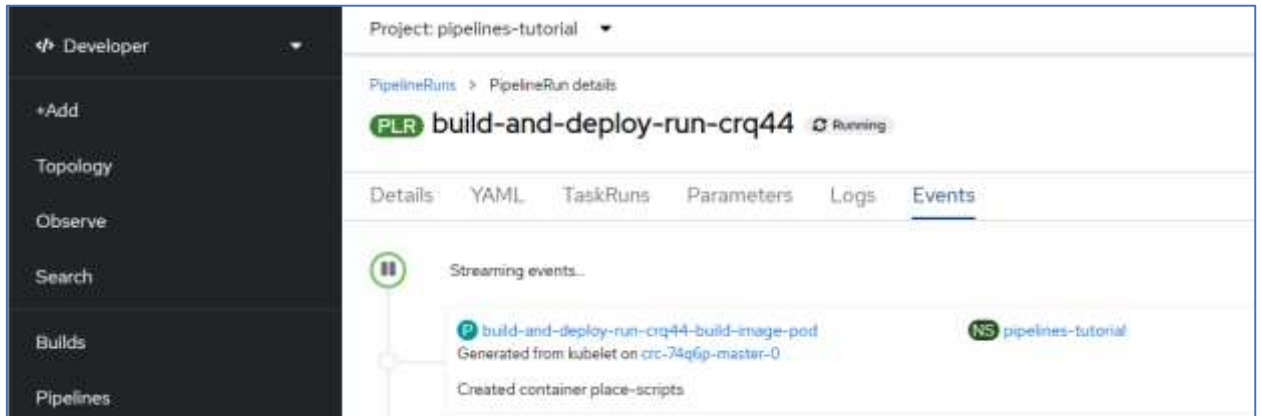
Можем запустить конвейер и привязать его к persistVolumeClaim и параметрам, которые должны использоваться для этого конкретного вызова.

Building Kubernetes Applications

Запустим конвейер для сборки и развертывания серверного приложения, используя tkn:

```
tkn pipeline start build-and-deploy -w name=shared-  
workspace,volumeClaimTemplateFile=https://raw.githubusercontent.com/openshift/pipelines-  
tutorial/master/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
```

Дожидаемся выполнения пайплайна.



Аналогичным образом запустите конвейер для сборки и развертывания внешнего интерфейса:

```
tkn pipeline start build-and-deploy -w name=shared-  
workspace,volumeClaimTemplateFile=https://raw.githubusercontent.com/openshift/pipelines-  
tutorial/master/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
```

Для просмотра хода выполнения пайплайна выполните:

```
tkn pipelinerun logs build-and-deploy-run-xxxxx -f -n pipelines-tutorial
```

Как только вы запустите build-and-deploy конвейер, будет создан экземпляр конвейера, и будут созданы модули для выполнения задач, определенных в конвейере.

```
tkn pipeline list
```

```
tkn pipelinerun ls
```

Посмотрите журналы выполнения конвейера по мере его запуска с помощью команды `tkn pipeline logs`, которая в интерактивном режиме позволяет вам выбрать интересующий вас конвейер и просмотреть журналы:

```
tkn pipeline logs -f
```

Building Kubernetes Applications

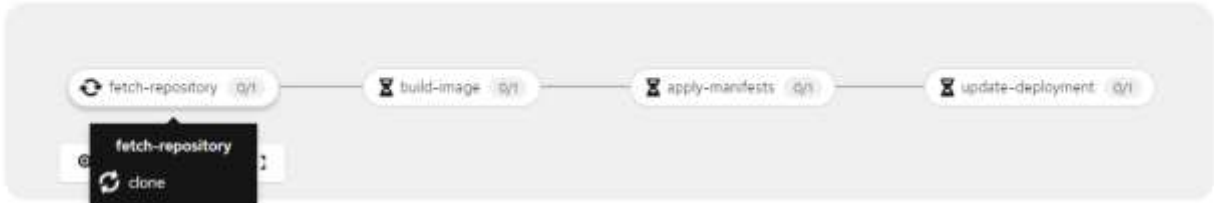
Project: pipelines-tutorial

PipelineRuns > PipelineRun details

PLR build-and-deploy-run-lnh28 Running

[Details](#) [YAML](#) [TaskRuns](#) [Parameters](#) [Logs](#) [Events](#)

PipelineRun details



The diagram illustrates a linear sequence of four steps in a PipelineRun:

- fetch-repository** (0/1) - Status: **done**
- build-image** (0/1)
- apply-manifests** (0/1)
- update-deployment** (0/1)

Name
build-and-deploy-run-lnh28

Status
Running

Namespace
NS pipelines-tutorial

Pipeline
PL build-and-deploy

VolumeClaimTemplate Resources
PVC source-pvc-739480533a

[Edit](#)

В финале вы должны увидеть, что образы успешно собраны и развернуты.