builds for Red Hat OpenShift 1.1

Work with Builds

Using Builds

Last Updated: 2025-10-01

builds for Red Hat OpenShift 1.1 Work with Builds

Using Builds

Legal Notice

Abstract

This document provides procedural examples for using Builds.

Table of Contents

CHAPTER 1. MANAGING BUILDS

After installing Builds, you can create builds using **buildah** or **source-to-image** build strategies. You can also delete custom resources that are not required for a build.

1.1. CREATING A BUILDAH BUILD

You can create a **buildah** build and push the created image to the target registry.

Prerequisites

- You have installed the Builds for Red Hat OpenShift Operator on the OpenShift Container Platform cluster.
- You have created a **ShipwrightBuild** resource.
- You have installed the **oc** CLI.
- Optional: You have installed the **shp** CLI.

Procedure

1. Create a **Build** resource and apply it to the OpenShift Container Platform cluster by using one of the CLIs:

   #### Example: Using oc CLI

```
$ oc apply -f - <<EOF
apiVersion: shipwright.io/v1beta1
kind: Build
metadata:
  name: buildah-golang-build
spec:
  source: 1
    type: Git
    git:
      url: https://github.com/shipwright-io/sample-go
    contextDir: docker-build
  strategy: 2
    name: buildah
    kind: ClusterBuildStrategy
  paramValues: 3
  - name: dockerfile
    value: Dockerfile
  output: 4
    image: image-registry.openshift-image-registry.svc:5000/buildah-example/sample-go-app
EOF
```

1. The location where the source code is placed.
2. The build strategy that you use to build the container.
3. The parameter defined in the build strategy. To set the value of the **dockerfile** strategy parameter, specify the Dockerfile location required to build the output image.

4 The location where the built image is pushed. In this procedural example, the built image is pushed to the OpenShift Container Platform cluster internal registry. **buildah-example** is the name of the current project. Ensure that the specified project exists to allow the image push.

Example: Using shp CLI

```
$ shp build create buildah-golang-build \
--source-url="https://github.com/redhat-openshift-builds/samples" --source-context-
dir="buildah-build" \1
--strategy-name="buildah" \2
--dockerfile="Dockerfile" \3
--output-image="image-registry.openshift-image-registry.svc:5000/buildah-example/go-app"
4
```

1 The location where the source code is placed.

2 The build strategy that you use to build the container.

3 The parameter defined in the build strategy. To set the value of the **dockerfile** strategy parameter, specify the Dockerfile location required to build the output image.

4 The location where the built image is pushed. In this procedural example, the built image is pushed to the OpenShift Container Platform cluster internal registry. **buildah-example** is the name of the current project. Ensure that the specified project exists to allow the image push.

2. Check if the **Build** resource is created by using one of the CLIs:

Example: Using oc CLI

```
$ oc get builds.shipwright.io buildah-golang-build
```

Example: Using shp CLI

```
$ shp build list
```

3. Create a **BuildRun** resource and apply it to the OpenShift Container Platform cluster by using one of the CLIs:

Example: Using oc CLI

```
$ oc apply -f - <<EOF
apiVersion: shipwright.io/v1beta1
kind: BuildRun
metadata:
  name: buildah-golang-buildrun
spec:
```

```
  build:
    name: buildah-golang-build 1
EOF
```

1 The **spec.build.name** field denotes the respective build to run, which is expected to be available in the same namespace.

Example: Using shp CLI

```
$ shp build run buildah-golang-build --follow 1
```

1 Optional: By using the **--follow** flag, you can view the build logs in the output result.

4. Check if the **BuildRun** resource is created by running one of the following commands:

Example: Using oc CLI

```
$ oc get buildrun buildah-golang-buildrun
```

Example: Using shp CLI

```
$ shp buildrun list
```

The **BuildRun** resource creates a **TaskRun** resource, which then creates the pods to execute build strategy steps.

Verification

1. After all the containers complete their tasks, verify the following:

- Check whether the pod shows the **STATUS** field as **Completed**:

```
$ oc get pods -w
```

Example output

```
NAME                                READY   STATUS    RESTARTS   AGE
buildah-golang-buildrun-dtrg2-pod   2/2     Running   0          4s
buildah-golang-buildrun-dtrg2-pod   1/2     NotReady  0          7s
buildah-golang-buildrun-dtrg2-pod   0/2     Completed 0          55s
```

- Check whether the respective **TaskRun** resource shows the **SUCCEEDED** field as **True**:

```
$ oc get tr
```

Example output

```
NAME                          SUCCEEDED  REASON     STARTTIME   COMPLETIONTIME
buildah-golang-buildrun-dtrg2  True       Succeeded  11m         8m51s
```

- Check whether the respective **BuildRun** resource shows the **SUCCEEDED** field as **True**:

```
$ oc get br
```

Example output

```
NAME                    SUCCEEDED   REASON      STARTTIME   COMPLETIONTIME
buildah-golang-buildrun  True       Succeeded   13m         11m
```

During verification, if a build run fails, you can check the **status.failureDetails** field in your **BuildRun** resource to identify the exact point where the failure happened in the pod or container.

NOTE

The pod might switch to a **NotReady** state because one of the containers has completed its task. This is an expected behavior.

2. Validate whether the image has been pushed to the registry that is specified in the **build.spec.output.image** field. You can try to pull the image by running the following command from a node that can access the internal registry:

```
$ podman pull image-registry.openshift-image-registry.svc:5000/<project>/<image> 1
```

1 The project name and image name used when creating the **Build** resource. For example, you can use **buildah-example** as the project name and **sample-go-app** as the image name.

1.2. CREATING A SOURCE-TO-IMAGE BUILD

You can create a **source-to-image** build and push the created image to a custom Quay repository.

Prerequisites

- You have installed the Builds for Red Hat OpenShift Operator on the OpenShift Container Platform cluster.
- You have created a **ShipwrightBuild** resource.
- You have installed the **oc** CLI.
- Optional: You have installed the **shp** CLI.

Procedure

1. Create a **Build** resource and apply it to the OpenShift Container Platform cluster by using one of the CLIs:

Example: Using oc CLI

```
$ oc apply -f - <<EOF
```

```
apiVersion: shipwright.io/v1beta1
kind: Build
metadata:
  name: s2i-nodejs-build
spec:
  source: 1
    type: Git
    type: Git
      git:
        url: https://github.com/redhat-openshift-builds/samples
      contextDir: s2i-build/nodejs
  strategy: 2
    name: source-to-image
    kind: ClusterBuildStrategy
  paramValues: 3
  - name: builder-image
    value: quay.io/centos7/nodejs-12-centos7:master
  output:
    image: quay.io/<repo>/s2i-nodejs-example 4
    pushSecret: registry-credential 5
EOF
```

1. The location where the source code is placed.
2. The build strategy that you use to build the container.
3. The parameter defined in the build strategy. To set the value of the **builder-image** strategy parameter, specify the builder image location required to build the output image.
4. The location where the built image is pushed. You can push the built image to a custom Quay.io repository. Replace **repo** with a valid Quay.io organization or your Quay user name.
5. The secret name that stores the credentials for pushing container images. To generate a secret of the type **docker-registry** for authentication, see "Authentication to container registries".

Example: Using shp CLI

```
$ shp build create s2i-nodejs-build \
--source-url="https://github.com/redhat-openshift-builds/samples" --source-context-dir="s2i-build/nodejs" \1
--strategy-name="source-to-image" \2
--builder-image="quay.io/centos7/nodejs-12-centos7" \3
--output-image="quay.io/<repo>/s2i-nodejs-example" \4
--output-credentials-secret="registry-credential" 5
```

1. The location where the source code is placed.
2. The build strategy that you use to build the container.
3. The parameter defined in the build strategy. To set the value of the **builder-image** strategy parameter, specify the builder image location required to build the output image.

4. The location where the built image is pushed. You can push the built image to a custom Quay.io repository. Replace **repo** with a valid Quay.io organization or your Quay user name.

5. The secret name that stores the credentials for pushing container images. To generate a secret of the type **docker-registry** for authentication, see "Authentication to container registries".

2. Check if the **Build** resource is created by using one of the CLIs:

Example: Using oc CLI

```
$ oc get builds.shipwright.io s2i-nodejs-build
```

Example: Using shp CLI

```
$ shp build list
```

3. Create a **BuildRun** resource and apply it to the OpenShift Container Platform cluster by using one of the CLIs:

Example: Using oc CLI

```
$ oc apply -f - <<EOF
apiVersion: shipwright.io/v1beta1
kind: BuildRun
metadata:
  name: s2i-nodejs-buildrun
spec:
  build:
    name: s2i-nodejs-build (1)
EOF
```

1. The **spec.build.name** field denotes the respective build to run, which is expected to be available in the same namespace.

Example: Using shp CLI

```
$ shp build run s2i-nodejs-build --follow (1)
```

1. Optional: By using the **--follow** flag, you can view the build logs in the output result.

4. Check if the **BuildRun** resource is created by running one of the following commands:

Example: Using oc CLI

```
$ oc get buildrun s2i-nodejs-buildrun
```

Example: Using shp CLI

```
$ shp buildrun list
```

The **BuildRun** resource creates a **TaskRun** resource, which then creates the pods to execute build strategy steps.

Verification

1. After all the containers complete their tasks, verify the following:
   - Check whether the pod shows the **STATUS** field as **Completed**:

     ```
     $ oc get pods -w
     ```

     Example output

     ```
     NAME                                READY   STATUS     RESTARTS   AGE
     s2i-nodejs-buildrun-phxxm-pod       2/2     Running    0          10s
     s2i-nodejs-buildrun-phxxm-pod       1/2     NotReady   0          14s
     s2i-nodejs-buildrun-phxxm-pod       0/2     Completed  0          2m
     ```

   - Check whether the respective **TaskRun** resource shows the **SUCCEEDED** field as **True**:

     ```
     $ oc get tr
     ```

     Example output

     ```
     NAME                        SUCCEEDED  REASON     STARTTIME   COMPLETIONTIME
     s2i-nodejs-buildrun-phxxm   True       Succeeded  2m39s       13s
     ```

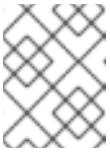
   - Check whether the respective **BuildRun** resource shows the **SUCCEEDED** field as **True**:

     ```
     $ oc get br
     ```

     Example output

     ```
     NAME                SUCCEEDED  REASON     STARTTIME   COMPLETIONTIME
     s2i-nodejs-buildrun True       Succeeded  2m41s       15s
     ```

     During verification, if a build run fails, you can check the **status.failureDetails** field in your **BuildRun** resource to identify the exact point where the failure happened in the pod or container.

     NOTE

     The pod might switch to a **NotReady** state because one of the containers has completed its task. This is an expected behavior.

2. Validate whether the image has been pushed to the registry that is specified in the **build.spec.output.image** field. You can try to pull the image by running the following command after logging in to the registry:

```
$ podman pull quay.io/<repo>/<image> 1
```

1 The repository name and image name used when creating the **Build** resource. For example, you can use **s2i-nodejs-example** as the image name.

Additional resources

- Authentication to container registries

1.3. EDITING THE RESOURCES

You can edit the resources that are created by **buildah** and **source-to-image** build processes using the **oc** CLI. You can modify the resources as needed in your project.

Prerequisites

- You have installed the **oc** CLI.

Procedure

1. Run the following command to open the YAML definition in the default editor:

```
$ oc edit <resource_name> <build_resource_name> 1
```

1 Replace **<resource_name>** with the name of the resource (**build**, **buildrun** or **buildstrategy**) and **<build_resource_name>** with the name of the build resource that you want to edit.

2. Edit the YAML definition and save the file.

1.4. DELETING THE RESOURCES

You can delete resources created by the **Buildah** and **Source-to-Image** (S2I) build processes using the oc CLI or the shp CLI. Deleting these resources helps you to clean up build configurations that are no longer required in your project.

1.4.1. Deleting a build resource

You can delete a **build** resource if it is not required in your project.

Prerequisites

- You have installed the **oc** CLI.
- Optional: You have installed the shp CLI.

Procedure

- Delete a **build** resource by using one of the following CLIs:

- Using **oc** CLI

```
$ oc delete builds <build_resource_name> 1
```

1 Replace <build_resource_name> with the name of the **build** resource.

- Using **shp** CLI

```
$ shp build delete <build_resource_name> 1
```

1 Replace <build_resource_name> with the name of the **build** resource.

1.4.2. Deleting a buildrun resource

You can delete a **buildrun** resource if it is not required in your project.

Prerequisites

- You have installed the **oc** CLI.
- Optional: You have installed the shp CLI.

Procedure

- Delete a **build** resource by using one of the following CLIs:
  - Using **oc** CLI

```
$ oc delete buildrun <buildrun_resource_name> 1
```

1 Replace <buildrun_resource_name> with the name of the **buildrun** resource.

  - Using **shp** CLI

```
$ oc delete buildrun <buildrun_resource_name> 1
```

1 Replace <buildrun_resource_name> with the name of the **buildrun** resource.

1.4.3. Deleting a buildstrategy resource

You can delete a **buildstrategy** resource if it is not required in your project.

Prerequisites

- You have installed the **oc** CLI.

Procedure

- Delete a **buildstrategy** resource by using the **oc** CLI:

```
$ oc delete buildstrategy <buildstartegy_resource_name> 1
```

1 Replace <buildstartegy_resource_name> with the name of the **buildstrategy** resource.

1.5. ADDITIONAL RESOURCES

- Authentication to container registries
- Creating a ShipwrightBuild resource by using the web console