builds for Red Hat OpenShift 1.0

About Builds

Introduction to Builds

Last Updated: 2024-02-13

builds for Red Hat OpenShift 1.0 About Builds

Introduction to Builds

Abstract

This document provides an overview of Builds features. It also includes release notes and details on how to get support.

Table of Contents

CHAPTER 1. RELEASE NOTES

Release notes contain information about new and deprecated features, breaking changes, and known issues. The following release notes apply for the most recent Builds releases on OpenShift Container Platform.

Builds is an extensible build framework based on the Shipwright project, which you can use to build container images on an OpenShift Container Platform cluster. You can build container images from source code and Dockerfiles by using image build tools, such as Source-to-Image (S2I) and Buildah. You can create and apply build resources, view logs of build runs, and manage builds in your OpenShift Container Platform namespaces.

Builds includes the following capabilities:

- Standard Kubernetes-native API for building container images from source code and Dockerfiles
- Support for Source-to-Image (S2I) and Buildah build strategies
- Extensibility with your own custom build strategies
- Execution of builds from source code in a local directory
- Shipwright CLI for creating and viewing logs, and managing builds on the cluster
- Integrated user experience with the **Developer** perspective of the OpenShift Container Platform web console

For more information about Builds, see Overview of Builds.

1.1. COMPATIBILITY AND SUPPORT MATRIX

In the table, components are marked with the following statuses:

TP	Technology Preview
GA	General Availability

The Technology Preview features are experimental features and are not intended for production use.

Table 1.1. Compatibility and support matrix

Builds Version	Component Version		Compatible Openshift Pipelines Version	OpenShift Version
Operator	Builds (Shipwright)	CLI		
1.0	0.12.0 (GA)	0.12.0 (GA)	1.12 and later	4.14

1.2. MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

1.3. RELEASE NOTES FOR BUILDS GENERAL AVAILABILITY 1.0

Builds General Availability (GA) 1.0 is now available on OpenShift Container Platform 4.14.

1.3.1. New features

The following sections highlight what is new in Builds 1.0.

1.3.1.1. Builds

- This release supports the **buildah** and **source-to-image** build strategies. The Builds for Red Hat OpenShift Operator automatically installs these strategies for use.

  NOTE

  The **buildpacks** build strategy is currently in Developer Preview. Red Hat plans to make this strategy generally available for use in a future release.

- With this release, you can define volume mounts in your **BuildStartegy** resource. When defining a strategy, you can declare volumes which can be shared across build steps and build runs. You can also declare the mount point for the volume which is fixed across all build steps. Volumes can either have a fixed volume source, or an overridable volume source that you can set in a **Build** or **BuildRun** resource.

- With this release, you can embed a complete build specification into your **BuildRun** resource by using the **spec.build.spec** field. By embedding specifications, you can build an image without creating and maintaining a dedicated **Build** resource.

- This release supports automated cleaning of completed build runs after a specific time or when a certain number of build runs is reached. By using retention parameters, you can specify the duration for which a completed build run can exist and the number of succeeded or failed build runs that can exist.

- With this release, you can configure a build by defining the source, build strategy, parameter values, builder or docker file, output, retention parameters, and volumes in a **Build** resource.

- With this release, you can configure a build strategy by defining strategy parameters, system parameters, step resources definitions, annotations, and volumes in a **BuildStrategy** or **ClusterBuildStrategy** resource.

- With this release, you can configure a build run by defining the build reference, build specification, parameter values, service account, output, retention parameters, and volumes.

- With this release, you can monitor your build resources by using build controller metrics.

- With this release, you can add an annotation **build.shipwright.io/referenced.secret: "true"** to a build secret. Based on this annotation, the build controller takes a reconcile action when an event, such as create, update, or delete triggers for the build secret.

1.3.1.2. CLI

- With this release, you can create, delete, list, or run a **Build** resource.
- With this release, you can create or list a **BuildRun** resource, or view its logs.

CHAPTER 2. OVERVIEW OF BUILDS

Builds is an extensible build framework based on the Shipwright project, which you can use to build container images on an OpenShift Container Platform cluster. You can build container images from source code and Dockerfiles by using image build tools, such as Source-to-Image (S2I) and Buildah. You can create and apply build resources, view logs of build runs, and manage builds in your OpenShift Container Platform namespaces.

Builds includes the following capabilities:

- Standard Kubernetes-native API for building container images from source code and Dockerfiles
- Support for Source-to-Image (S2I) and Buildah build strategies
- Extensibility with your own custom build strategies
- Execution of builds from source code in a local directory
- Shipwright CLI for creating and viewing logs, and managing builds on the cluster
- Integrated user experience with the **Developer** perspective of the OpenShift Container Platform web console

Builds consists of the following custom resources (CRs):

- **Build**
- **BuildStrategy** and **ClusterBuildStrategy**
- **BuildRun**

2.1. BUILD RESOURCE

The **Build** resource defines the source code of your application and the location where your application images will be pushed. The following example shows a simple build that consists of a Git source, a build strategy, and an output image:

```
apiVersion: shipwright.io/v1beta1
kind: Build
metadata:
  name: buildah-golang-build
spec:
  source:
    git:
      url: https://github.com/username/taxi
  strategy:
    name: buildah
    kind: ClusterBuildStrategy
  output:
    image: registry.mycompany.com/my-org/taxi-app:latest
```

You can also extend a **Build** resource to push your images to a private registry or use a Dockerfile.

2.2. BUILDSTRATEGY AND CLUSTERBUILDSTRATEGY RESOURCES

The **BuildStrategy** and **ClusterBuildStrategy** resources define a series of steps to assemble an application. You can use the **BuildStrategy** resources within a namespace and the **ClusterBuildStrategy** resources within a cluster.

The specification of a **BuildStrategy** or **ClusterBuildStrategy** resource consists of a **steps** object. The following example shows the specification of the **buildah** cluster build strategy:

```
apiVersion: shipwright.io/v1beta1
kind: ClusterBuildStrategy
metadata:
  name: buildah
spec:
  steps:
    - name: build-and-push
      image: quay.io/containers/buildah:v1.31.0
      workingDir: $(params.shp-source-root)
      command:
        - /bin/bash
       # ...
# ...
```

2.3. BUILDRUN RESOURCE

A **BuildRun** resource invokes a build on your cluster, similar to any cluster job or Tekton task run. The **BuildRun** resource represents a workload on your cluster, which results in a running pod. A **BuildRun** is the running instance of a build. It instantiates a build for execution with specific parameters on a cluster.

A **BuildRun** resource helps you to define the following elements:

- A unique **BuildRun** name to monitor the status of the build
- A referenced **Build** instance to use during the build
- A service account to host all secrets for the build

Each **BuildRun** resource is available within a namespace.

2.4. BUILD CONTROLLER

The build controller monitors any updates in the **Build** resource and performs the following tasks:

- Validates if the referenced **Strategy** object exists in the **Build** resource.
- Validates if the specified parameters in the **Build** CR exist in the referenced build strategy. It also validates if the parameter names collide with any reserved names.
- Validates if the container registry output secret exists in the **Build** resource.
- Validates if the referenced **spec.source.git.url** endpoint URL exists in the **Build** resource.

The build run controller monitors any updates in the **Build** or **TaskRun** resource and performs the following tasks:

- Searches for any existing **TaskRun** resource and updates its parent **BuildRun** resource status.

- Retrieves the specified service account and sets it along with the output secret in the **Build** resource.
- If a **TaskRun** resource does not exist, the controller generates a new Tekton **TaskRun** resource and sets a reference to the **TaskRun** resource.
- For any subsequent updates in the **TaskRun** resource, the controller updates the parent **BuildRun** resource.

2.4.1. Build validations

To avoid triggering **BuildRun** resources that will fail because of incorrect or missing dependencies or configuration settings, the build controller validates them in advance. If all validations are successful, you view a **status.reason** field named **Succeeded**. However, if any validations fail, you must check the **status.reason** and **status.message** fields to understand the root cause.

Table 2.1. Validation of builds by the build controller

status.reason field	Description
BuildStrategyNotFound	The referenced strategy at namespace level does not exist.
ClusterBuildStrategyNotFound	The referenced strategy at cluster level does not exist.
SetOwnerReferenceFailed	Setting owner references between a **Build** and a **BuildRun** resources failed. This status is triggered when you set the **spec.retention.atBuildDeletion** field to **true** in a build.
SpecSourceSecretRefNotFound	The secret used to authenticate to Git does not exist.
SpecOutputSecretRefNotFound	The secret used to authenticate to the container registry does not exist.
SpecBuilderSecretRefNotFound	The secret used to authenticate to the container registry does not exist.
MultipleSecretRefNotFound	Multiple secrets used for authentication are missing.
RestrictedParametersInUse	One or many defined **params** are colliding with any reserved parameters.
UndefinedParameter	The parameters are not defined in the referenced strategy. You must define those parameters in the **spec.parameters** specification in your strategy.
RemoteRepositoryUnreachable	The defined **spec.source.git.url** specification was not found. This validation only takes place for HTTP and HTTPS protocols.

status.reason field	Description
BuildNameInvalid	The build name in the **metadata.name** field is invalid. You must use a valid label value for the build name.
SpecEnvNameCanNotBeBlank	Indicates that the name for a user-provided environment variable is blank.
SpecEnvValueCanNotBeBlank	Indicates that the value for a user-provided environment variable is blank.

2.5. ADDITIONAL RESOURCES

- Installing Builds by using the console
- Installing Builds by using the CLI

CHAPTER 3. BUILD STRATEGIES

You can use a curated set of build strategies or cluster build strategies on the OpenShift Container Platform cluster. The Builds for Red Hat OpenShift Operator automatically installs these strategies for use. This automated installation of strategies helps you to quickly get started with Builds.

Builds supports the following cluster build strategies:

- **buildah**: Supported on all platforms
- **source-to-image**: Supported on the linux/amd64 platform

NOTE

The **buildpacks** build strategy is currently in Developer Preview. For more information, see the buildpacks example.

3.1. BUILDAH

The **buildah** cluster build strategy uses a Dockerfile to build a container image and pushes it to the target registry. You must specify the Dockerfile in the **spec.paramValues** field of the **Build** CR.

You can share the **buildah** strategy across different namespaces within your cluster because the Builds for Red Hat OpenShift Operator installs the **buildah** strategy at cluster level.

You can configure the following parameters for the **buildah** strategy:

Table 3.1. Configuration parameters for**buildah**

Name	Type	Description	Default
build-args	array	Key-value pair of the arguments required by the Dockerfile that is used during the build	[]
registries-block	array	List of registries that must be blocked	[]
registries-insecure	array	List of insecure registries with their fully qualified domain name (FQDN)	[]
registries-search	array	List of registries to search short name images	["registry.redhat.io", "quay.io"]
dockerfile	string	Path of the Dockerfile that is used during the build	"Dockerfile"

Name	Type	Description	Default
storage-driver	string	Storage drivers that are used by **buildah**, such as overlay or vfs	"vfs"

NOTE

For more information, see *Configuring build strategies* in the *Additional resources* section.

3.2. SOURCE-TO-IMAGE

This build strategy is composed of **source-to-image** and **buildah**. You can use this strategy to generate a container file and prepare the application to build with a builder image. You must specify the builder image in the **spec.paramValues** field of the **Build** CR.

You can share the **source-to-image** strategy across different namespaces within your cluster because the Builds for Red Hat OpenShift Operator installs the **source-to-image** strategy at cluster level.

You can configure the following parameters for the **source-to-image** strategy:

Table 3.2. Configuration parameters for**source-to-image**

Name	Type	Description	Default
registries-block	array	List of registries that must be blocked	[]
registries-insecure	array	List of insecure registries with their FQDN	[]
registries-search	array	List of registries to search short name images	["registry.redhat.io", "quay.io"]
builder-image	string	Location of the builder image that is used during the build	NA
storage-driver	string	Storage drivers that are used by **source-to-image**, such as overlay or vfs	"vfs"

3.3. ADDITIONAL RESOURCES

- Installing Builds by using the web console

- Installing Builds by using the CLI
- Configuring build strategies
- Creating a ShipwrightBuild resource by using the web console