How Builds Work

What Is a Build?

A *build* in OpenShift Container Platform is the process of transforming input parameters into a resulting object. Most often, builds are used to transform source code into a runnable container image.

A *build configuration*, or `BuildConfig`, is characterized by a *build strategy* and one or more sources. The strategy determines the aforementioned process, while the sources provide its input.

The build strategies are:

- Source-to-Image (S2I) (description, options)
- Pipeline (description, options)
- Docker (description, options)
- Custom (description, options)

And there are six types of sources that can be given as *build input*:

- Git
- Dockerfile
- Binary
- Image
- Input secrets

- External artifacts

It is up to each build strategy to consider or ignore a certain type of source, as well as to determine how it is to be used. Binary and Git are mutually exclusive source types. Dockerfile and Image can be used by themselves, with each other, or together with either Git or Binary. The Binary source type is unique from the other options in how it is specified to the system.

What Is a BuildConfig?

A build configuration describes a single build definition and a set of triggers for when a new build should be created. Build configurations are defined by a `BuildConfig`, which is a REST object that can be used in a POST to the API server to create a new instance.

Depending on how you choose to create your application using OpenShift Container Platform, a `BuildConfig` is typically generated automatically for you if you use the web console or CLI, and it can be edited at any time. Understanding the parts that make up a `BuildConfig` and their available options can help if you choose to manually tweak your configuration later.

The following example `BuildConfig` results in a new build every time a container image tag or the source code changes:

BuildConfig Object Definition

```
kind: "BuildConfig"
apiVersion: "v1"
metadata:
  name: "ruby-sample-build" (1)
spec:
  runPolicy: "Serial" (2)
  triggers: (3)
    -
      type: "GitHub"
      github:
        secret: "secret101"
    - type: "Generic"
      generic:
        secret: "secret101"
    -
      type: "ImageChange"
  source: (4)
    git:
      uri: "https://github.com/openshift/ruby-hello-world"
  strategy: (5)
    sourceStrategy:
      from:
        kind: "ImageStreamTag"
        name: "ruby-20-centos7:latest"
  output: (6)
    to:
      kind: "ImageStreamTag"
      name: "origin-ruby-sample:latest"
  postCommit: (7)
      script: "bundle exec rake test"
```

1 This specification will create a new `BuildConfig` named **ruby-sample-build**.

2 The `runPolicy` field controls whether builds created from this build configuration can be run simultaneously. The default value is `Serial`, which means new builds will run sequentially, not simultaneously.

3 You can specify a list of triggers, which cause a new build to be created.

4 The `source` section defines the source of the build. The source type determines the primary source of input, and can be either `Git`, to point to a code repository location, `Dockerfile`, to build from an inline Dockerfile, or `Binary`, to accept binary payloads. It is possible to have multiple sources at once, refer to the documentation for each source type for details.

5 The `strategy` section describes the build strategy used to execute the build. You can specify a `Source` , `Docker` , or `Custom` strategy here. This above example uses the `ruby-20-centos7` container image that Source-To-Image will use for the application build.

6 After the container image is successfully built, it will be pushed into the repository described in the `output` section.

7 The `postCommit` section defines an optional build hook.