Using community-contributed builders and custom builders

This page explains how to use community-contributed builders (https://github.com/GoogleCloudPlatform/cloud-builders-community) and custom builders in Cloud Build. The Cloud Build developer community provides open-source builders (https://github.com/GoogleCloudPlatform/cloud-builders-community) that you can use to execute your tasks. If the task you want to perform requires capabilities that are not provided by an existing image, you can build your own custom image and use it in a build step. To learn about the different types of builders, see Cloud Builders (/cloud-build/docs/cloud-builders).

If you're new to Cloud Build, read the quickstarts (/cloud-build/docs/quickstarts) and the Build configuration overview (/cloud-build/docs/build-config) first.

Using community-contributed builders

Pre-built images are not available for community-contributed builders; to use these builders in a Cloud Build config file, you must first build the image and push it to Container Registry (/container-registry/docs) in your project.

To use a community-contributed builder:

1. Build and push the builder:
    a. Navigate to your project root directory.
    b. Clone the cloud-builders-community (https://github.com/GoogleCloudPlatform/cloud-builders-community) repository:

```
git clone https://github.com/GoogleCloudPlatform/cloud-builders-community.
```

    c. Navigate to the builder image you want to use, where ***builder-name*** is the directory that contains the builder:

```
cd cloud-builders-community/builder-name
```

d. Submit the builder to your project:

```
gcloud builds submit .
```

e. Navigate back to your project root directory:

```
cd ../..
```

f. Remove the repository from your root directory:

```
rm -rf cloud-builders-community/
```

2. In your Cloud Build config file, use the builder in a build step:

YAML JSON (#json)

```
steps:
- name: 'gcr.io/project-id/builder-name'
  args: ['arg1', 'arg2', ...]
...
```

3. Use the build config file to start the build manually (/cloud-build/docs/running-builds/start-build-manually) or build using triggers (/cloud-build/docs/automating-builds/create-manage-triggers).

For examples on using community-contributed builders, see Deploy to Firebase (/cloud-build/docs/deploying-builds/deploy-firebase) and Build VM images using Packer (/cloud-build/docs/building/build-vm-images-with-packer).

Creating a custom builder

If the task you want to perform requires capabilities that are not provided by a public image, a supported builder, or a community-contributed buider (/cloud-build/docs/cloud-builders), you can build your own image and use it in a build step.

Some examples of when you might want to use a custom builder image are:

- Downloading source code or packages from external locations.
- Using an external tool chain.
- Caching any necessary libraries.
- Pre-building source (with Cloud Build responsible only for potentially packaging the build into an image).

Like any other builder, a custom builder runs with the source mounted under `/workspace`, and is run with a working directory in `/workspace`. Any files left in `/workspace` by a given build step are available to other build steps.

Your custom builder can push to or pull from a repository in Container Registry (/container-registry/docs) (hosted at `gcr.io/$PROJECT-NAME/`) to which your Cloud Build service account (/cloud-build/docs/securing-builds/set-service-account-permissions) has access.

The following steps show how to create and use a custom builder with an example `Dockerfile`:

1. Create a Docker image:
    a. Create the `Dockerfile` for the custom builder. The following code shows an example `Dockerfile`:

```
FROM alpine
RUN apk add curl
CMD curl https://httpbin.org/ip -s > myip.txt; echo "*** My IP is: $(cat
```

    b. Build and push the custom builder to the Container Registry in your project, replacing values for ***project-id*** and ***image-name***:

```
gcloud builds submit --tag gcr.io/project-id/image-name
```

2. Use the image in a build step in your build config file:

YAMLJSON (#json)

```
steps:
- name: 'gcr.io/project-id/image-name'
  id: Determine IP of this build worker
```

3. Use the build config file to start the build manually (/cloud-build/docs/running-builds/start-build-manually) or build using triggers (/cloud-build/docs/automating-builds/create-manage-triggers).

Users can specify a working directory using the **`dir`** field in a build config file. Because your custom builder's us
pecify any **`dir`** value, the builder should avoid hard-coding **`/workspace`** if possible. Instead, use the current wo
ry and relative paths.

What's next

- Learn how to run bash scripts in build steps (/cloud-build/docs/configuring-builds/run-bash-scripts).
- Learn how to configure build step order (/cloud-build/docs/configuring-builds/configure-build-step-order).
- Learn how to write a basic build config file (/cloud-build/docs/configuring-builds/create-basic-configuration).

Last updated 2020-06-22 UTC.