



Singularity Global Client: Google Build

These sections will detail use of the [Google Build](#) client for `sregistry`, which means using the Google Build API to build a container, and then sending it to Google Storage. If you are interested in just pushing a container to Google Storage, see the [google-storage](#) client.

Getting Started

If you are using the `sregistry` image, the client is likely already installed. If you want to install this natively (or build a custom container) the command to install the module extras is:

```
pip install sregistry[google-build]

# or locally
git clone https://www.github.com/singularityhub/sregistry-cli.git
cd sregistry-cli
pip install -e .[google-build]
```

The next steps we will take are to first set up authentication, and then define your Storage Bucket (and other settings) via environment variables. The main difference between Google Build and other clients is that since we are building containers remotely, the “push” command is out of scope (and you would use push with the `google-storage` client to handle instead).



Singularity Registry Global Client works by way of obtaining information from the environment, which are cached when appropriate for future use. For Google Build, you will first need to [set up authentication](#) by following those steps. It comes down to creating a file and saving it on your system with the variable name `GOOGLE_APPLICATION_CREDENTIALS`. This variable will be found and used every time you use the storage Client, without needing to save anything to the secrets.

Thus, only required variable is the following:

- `GOOGLE_APPLICATION_CREDENTIALS` should point to the file provided.
- `SREGISTRY_GOOGLE_PROJECT` should be the name of your Google Project.

Optional variables include:

- `SREGISTRY_GOOGLE_BUILD_CACHE`: after build, do *not* delete intermediate dependencies in cloudbuild bucket (keep them as cache for rebuild if needed). Defaults to being unset, meaning that files are cleaned up. If you export this as anything, the build files will be cached.
- `SREGISTRY_GOOGLE_BUILD_SINGULARITY_VERSION`: if you want to specify a version of Singularity. The version must coincide with a container tag hosted under [singularityware/singularity](#). The version will default to the latest release, `3.0.2-slim`. If you want to use a different version, update this variable.
- `SREGISTRY_GOOGLE_STORAGE_BUCKET`: is the name for the bucket you want to create. If not provided, we use your username prefixed with “sregistry-“. Additionally, a temporary bucket is created with the same name ending in `_cloudbuild`. This bucket is for build time dependencies, and is cleaned up after the fact.
- `SREGISTRY_GOOGLE_STORAGE_PRIVATE`: by default, images that you upload will be made public, meaning that a user that stumbles on the URL (or has permission to read your bucket otherwise) will be able to see and download them. If you want to make an image private (one time or globally with an export in your bash profile) you should export this variable as some derivative

configuration for all subsequent images.

For a detailed list of other (default) environment variables and settings that you can configure, see the [getting started](#) pages. For the globally shared commands (e.g., “add”, “get”, “inspect,” “images,” and any others that are defined for all clients) see the [commands](#) documentation. Here we will review the set of commands that are specific to the Google Storage client:

- [build](#): [remote] build a container remotely, and save to Google Storage.
- [pull](#): [remote->local] pull an image from Google Storage to the local database and storage.
- [search](#): [remote] list all image collections in Google Storage

For all of the examples below, we will export our client preference to be `google-build`

```
SREGISTRY_CLIENT=google-build
export SREGISTRY_CLIENT
```

but note that you could just as easily define the variable for one command:

```
SREGISTRY_CLIENT=google-build sregistry shell
```

Build

Let’s get off to a running start and build! Note that we have a build recipe in the present working directory, and we also want to provide the context (all the recursive files that are alongside and below it). Let’s ask for help first:

```
$ sregistry build --help
usage: sregistry build [-h] [--preview] [--name NAME]
                    [commands [commands ...]]
```

Commands Google Build + Storage

```
-----  
build [recipe] [context] -----  
build [recipe] . -----  
build [recipe] file1 file2 -----
```

optional arguments:

```
-h, --help      show this help message and exit  
--preview, -p   preview the parsed configuration file only.  
--name NAME     name of image, in format "library/image"
```

Don't forget to export these variables:

```
export SREGISTRY_GOOGLE_PROJECT=my-project  
export GOOGLE_APPLICATION_CREDENTIALS=/path/to/application-credentials.json
```

Now let's launch a build, and provide the entire present working directory as context. Notice that we haven't exported `SREGISTRY_GOOGLE_BUILD_CACHE=yes` so we won't save intermediate build files.

Example Recipe

Let's say we've created a folder called "test" and added some Singularity recipe in it. If you have local filesystem dependencies (files to add to the container), put them in this folder. Here is the recipe - this is just about the simplest and smallest you can get:

```
Bootstrap: docker  
From: busybox:latest
```

Next, cd into the folder and run the build. Note that we don't end with "." so we only need the Singularity recipe as a build context.

```
gistry build --name <name> <recipe> <context>  
gistry build --name vanessa/llama Singularity
```

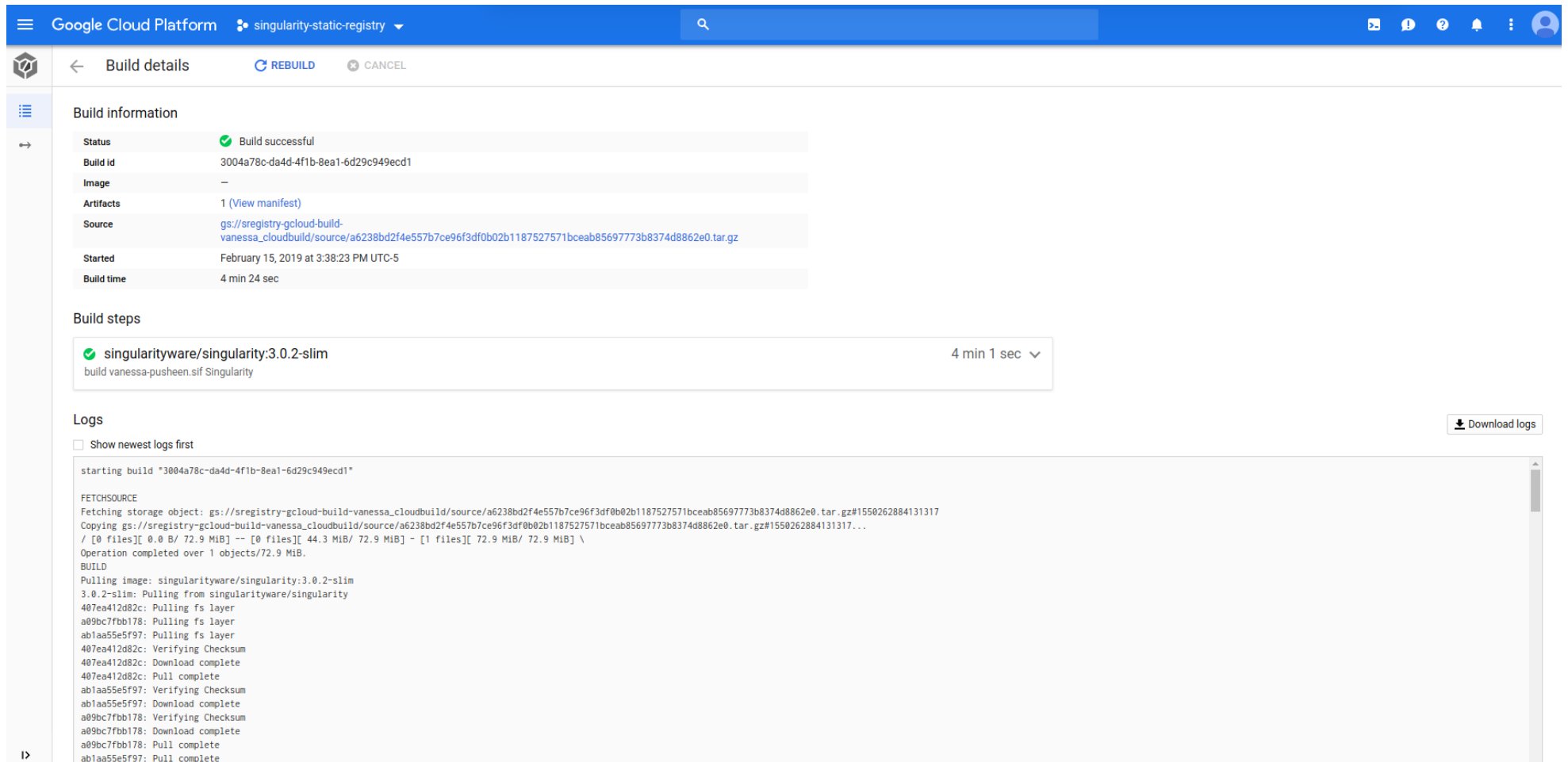
The above says “Build a container with name vanessa/llama using the Singularity recipe.” The tags defaults to latest. If we had other dependencies to upload in the folder, you could do either:

```
$ sregistry build --name vanessa/llama Singularity .  
$ sregistry build --name vanessa/llama Singularity file1 file2
```

Then you’ll see the build package generation (the dependency files from the present working directory), the upload (in the case that the files were not cached from a previous build) and then the build progressing from having status QUEUED, to WORKING, to (hopefully) SUCCESS. The message is updated every 15 seconds.

```
$ sregistry build --name vanessa/llama Singularity  
[client|google-build] [database|sqlite:///home/vanessa/.singularity/sregistry.db]  
[bucket][sregistry-gcloud-build-vanessa]  
LOG Generating build package for 1 files...  
LOG Uploading build package!  
PROJECT singularity-static-registry0/0 MB - 00:00:00  
BUILD singularityware/singularity:3.0.2-slim  
LOG build b73d08bb-2599-4f06-9d01-023d1894638f: QUEUED  
LOG build b73d08bb-2599-4f06-9d01-023d1894638f: WORKING  
LOG build b73d08bb-2599-4f06-9d01-023d1894638f: WORKING  
LOG build b73d08bb-2599-4f06-9d01-023d1894638f: SUCCESS  
LOG Total build time: 45.74 seconds  
SUCCESS gs://sregistry-gcloud-build-vanessa/vanessa-llama-latest.sif  
LOG https://storage.googleapis.com/sregistry-gcloud-build-vanessa/vanessa-llama-latest.sif  
LOG https://console.cloud.google.com/gcr/builds/b73d08bb-2599-4f06-9d01-023d1894638f?project=287055059824
```

public link to the container (a direct link), and the Google Cloud Console link to view output logs. You are encouraged to look at the logs because there is a lot of meaningful information here, especially if you need to debug your build!



The screenshot shows the Google Cloud Platform interface for a build. The top navigation bar includes the Google Cloud Platform logo, the project name 'singularity-static-registry', and a search bar. The main content area is titled 'Build details' and includes buttons for 'REBUILD' and 'CANCEL'. The 'Build information' section shows a successful build with the following details:

Status	Build successful
Build id	3004a78c-da4d-4f1b-8ea1-6d29c949ecd1
Image	-
Artifacts	1 (View manifest)
Source	gs://sregistry-gcloud-build-vanessa_cloudbuild/source/a6238bd2f4e557b7ce96f3df0b02b1187527571bceab85697773b8374d8862e0.tar.gz
Started	February 15, 2019 at 3:38:23 PM UTC-5
Build time	4 min 24 sec

The 'Build steps' section shows a single step: 'singularityware/singularity:3.0.2-slim' with a duration of 4 min 1 sec. The 'Logs' section is expanded, showing the following output:

```
starting build *3004a78c-da4d-4f1b-8ea1-6d29c949ecd1*
FETCHSOURCE
Fetching storage object: gs://sregistry-gcloud-build-vanessa_cloudbuild/source/a6238bd2f4e557b7ce96f3df0b02b1187527571bceab85697773b8374d8862e0.tar.gz#1550262884131317
Copying gs://sregistry-gcloud-build-vanessa_cloudbuild/source/a6238bd2f4e557b7ce96f3df0b02b1187527571bceab85697773b8374d8862e0.tar.gz#1550262884131317...
/ [0 files][ 0.0 B/ 72.9 MiB] -- [0 files][ 44.3 MiB/ 72.9 MiB] - [1 files][ 72.9 MiB/ 72.9 MiB] \
Operation completed over 1 objects/72.9 MiB.
BUILD
Pulling image: singularityware/singularity:3.0.2-slim
3.0.2-slim: Pulling from singularityware/singularity
407ea412d82c: Pulling fs layer
a09bc77bb178: Pulling fs layer
ab1aa55e5f97: Pulling fs layer
407ea412d82c: Verifying Checksum
407ea412d82c: Download complete
407ea412d82c: Pull complete
ab1aa55e5f97: Verifying Checksum
ab1aa55e5f97: Download complete
a09bc77bb178: Verifying Checksum
a09bc77bb178: Download complete
a09bc77bb178: Pull complete
ab1aa55e5f97: Pull complete
```

Given the https link, you can directly pull and run the container using Singularity:

```
$ singularity pull https://storage.googleapis.com/sregistry-gcloud-build-vanessa/vanessa-llama-latest.sif
WARNING: Authentication token file not found : Only pulls of public images will succeed
756.00 KiB / 756.00 KiB [=====]
```

If you use the interactive (from within Python) method, you are also returned this direct link to the container (`response['public_url']`) and you are free to put that in whatever database you are using to keep track of your containers. This is shown in the next section [Shell](#). If you want to search your storage later, see [Pull](#).

Shell

Next, let's do this from the interactive shell. Note that we have exported `SREGISTRY_CLIENT` above, as we are looking to interact with a shell for the google-build `sregistry` client.

```
sregistry shell

[client|google-build] [database|sqlite:///home/vanessa/.singularity/sregistry.db]
[bucket][sregistry-gcloud-build-vanessa]
Python 3.6.4 |Anaconda custom (64-bit)| (default, Jan 16 2018, 18:10:19)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

Here we see straight away that we are using the default bucket name (`sregistry-gcloud-build-vanessa`) and the google-build client. The printing of the bucket on the first line indicates we successfully connected to it, and we've also connected to the bucket of the same name ending with `_cloudbuild` (`sregistry-gcloud-build-vanessa_cloudbuild`). Next, we just need to provide the same arguments to the function to run the build.

```
> recipe = "Singularity"
```

```
ponse = client.build(name=name,  
                    recipe=recipe,  
                    context=context)
```

The output will be the same as shown previously. You are again encouraged to look at the logs link if the build isn't a success. With the interactive shell mode, you can also provide the `preview` argument to just inspect the configuration for the build:

```
> config = client.build(name=name,  
                       recipe=recipe,  
                       context=context,  
                       preview=True)
```

```
{  
  "steps": [  
    {  
      "name": "singularityware/singularity:3.0.2-slim",  
      "args": [  
        "build",  
        "vanessa-pusheen-latest.sif",  
        "Singularity"  
      ]  
    }  
  ],  
  "source": {  
    "storageSource": {  
      "bucket": "sregistry-gcloud-build-vanessa_cloudbuild",  
      "object": "source/8937958aa81fa7b81d8b6fc6eb89daca6c176cd99895c903517d74fc575a9dc9.tar.gz"  
    }  
  },  
}
```



```
    "location": "gs://sregistry-gcloud-build-vanessa",  
    "paths": [  
      "vanessa-pusheen-latest.sif"  
    ]  
  }  
}
```

Pull and Search

Now let's say that we built an image (some long time ago!) and want to find it in Google Storage. We would want to pull the image to our local `sregistry` database.

Search

A search without any parameters will essentially list all containers in the configured storage bucket. But how do we know what is a container?

a container is defined by having the metadata key "type" with value "container" and this is set by the upload (push) client.

Thus, if you do some fancy operation outside of using the client to upload containers to storage, make sure that you add this metadata value, otherwise they will not be found. Let's do a quick search to get our list in Google Storage. This action has no dependency on a local storage or database. Let's say we just built the container `vanessa/omgtacos:latest`. Can we find it?

```
$ sregistry search  
[client|google-build] [database|sqlite:///home/vanessa/.singularity/sregistry.db]  
[bucket][sregistry-gcloud-build-vanessa]
```

is! Then to look at details for a more specific search, let's try searching for "vanessa/avocados"

```
$ sregistry search vanessa/avocados
[client|google-build] [database|sqlite:///home/vanessa/.singularity/sregistry.db]
[bucket][sregistry-gcloud-build-vanessa]
[gs://sregistry-gcloud-build-vanessa] Found 1 containers
vanessa-avocados-latest.sif
id:      sregistry-gcloud-build-vanessa/vanessa-avocados-latest.sif/1550267799739163
uri:     vanessa/avocados-latest
updated: 2019-02-15 21:56:52.139000+00:00
size:    1 MB
md5:     GJ4jl2mFfaa+ckZhJu0wJg==
```

Pull

Finally, let's say we've found the container that we like, and we want to pull it.

```
$ sregistry pull vanessa/avocados:latest
[client|google-build] [database|sqlite:///home/vanessa/.singularity/sregistry.db]
[bucket][sregistry-gcloud-build-vanessa]
Searching for vanessa/avocados:latest in gs://sregistry-gcloud-build-vanessa
Progress |=====| 100.0%
[container][update] vanessa/avocados-latest@1e38ac7437da5afb8d89937115f052e0
Success! /home/vanessa/.singularity/shub/vanessa-avocados-latest@1e38ac7437da5afb8d89937115f052e0.sif
```

and if we list images, we see our container:

```
Containers:  [date]  [client] [uri]
             bruary 15, 2019      [google-build]      vanessa/avocados:latest@1e38ac7437da5afb89937115f052e0
```

The different versions (of the same name and tag) are listed. To get a path to any of the files:

```
$ sregistry get vanessa/avocados:latest@1e38ac7437da5afb89937115f052e0
/home/vanessa/.singularity/shub/vanessa-avocados-latest@1e38ac7437da5afb89937115f052e0.sif
```

You can also see in the pull output that on the backend of pull is the same search as you did before. This means that if you want to be precise, you should ask for the complete uri (version included). If you aren't precise, it will do a search across name fields and give you the first match. Be careful, my linux penguins.

For debugging scripts, see [Google Cloud Debugging](#).



Singularity Global Client is maintained by [Vanessa Sochat](#).

Contribute on [GitHub](#).