Apache Beam Containers

Overview

The idea is to build a set of public SDKHarness pre-built images, that users can utilize to run their portable pipelines without having to manually build them, or use these images as base images for customization.

Background information

- SDKHarness architecture / design docs ?
- Image structure ?
- Source ?

Location/Naming

This section describes the naming scheme and location for publication of the images.

Proposed repository

We are reviewing following three docker registries.

1. Gcr
   - **Quotas:**
     - It seems like gcr only has hit limit from each IP, I didn't find any documentation about image limit. ~~But it would have size limit~~. Container Registry uses Cloud Storage for each registry's underlying storage. Cloud Storage Quotas & Limits apply to each registry.
   - **Permissions:**
     - **Pulling** - public
     - **Pushing** - limited to authorized accounts that have correct permissions under apache-beam-testing:
       - publishing the snapshots nightly might be feasible similar to how we currently publish nightly maven snapshots, by creating a Jenkins job;
       - publishing at release time can be another job triggered manually by the release owner;
   - **GCP Project:**
     - Apache-beam-testing
   - **Advantage:**
     - Gcloud command-line tool provides an easy way to manage remote images, including remove tagging, remove images etc.
2. Bintray
   - **Quotas for OSS**:
     - 10GB storage
     - 1TB Downloads
   - **Permissions:**
     - **Pushing**: Can limit to certain users/groups. The maximum file size for uploads is 250 MB for OSS users.
     - **Pulling**: On the OSS plan, any content we upload to Bintray is publicly available. Anyone can download it, even if they don't have a Bintray account.
3. Docker Hub (free)
   - **Quotas**: Wasn't specified.

- **Permissions:** Manages at three levels, read/read-write/admin.
    - **Pushing**: Need read-write or above permission
    - **Pulling**: Any user can pull public repositories.

After comparing the above three docker registries, Docker Hub is a winner for release images. It is free, no quota limits, well adopted at industry. However, GCR is the easiest to manage remote images because it provides gcloud toolkit.

Remain tasks:

1. Where to host daily snapshot images?
2. How Dataflow uses release images? Mirror images from docker hub to gcr or other options?

Proposed naming and tagging scheme

We will push release candidate images first and make it official ones after validation. We are using tagging to differentiate candidate images and final images. The candidate images will be tagged with '_rc' suffix. The final release images will not have the suffix. Meanwhile, we will have a *latest* tag which always points to the most recent verified release image so users can pull it by default.

Release image name scheme: apachebeam/{repository}:tag

	Repository	Tagging	Example
Snapshot images	language + language_version	yyyymmdd_{suffix} in UTC	{location}/snapshot/java:20190820
Release images	{language}_ {language_version}_ sdk	{sdk_version}_{suffix}	apachebeam/python2.7_sdk:2.10.1 apachebeam/java_sdk:2.10.1_rc

**Java and Go will not have language version until we support multi versions.*

Publication Schedule

Snapshot images

- **how do we publish the snapshots, what's the frequency?**

        - automatic, when HEAD is built, nightly similar to mavens.
- **when and how do we cleanup the snapshot images:**
    - when you publish an image you have a new version/hash and still can use any previous versions. They take space and will count towards a quota. We need to clean them up periodically:
        - make it part of the publish job to look and delete the versions that are more than X days (or versions) old?

Release Images

- **How should we build and publish the images for release versions of Beam?**
    - Release manager will be provided a script to build and publish images.
- **Should it be a blocker for the release?**
    - Should we make it part of the release and not mark release as complete until the images are published?
        - For the first release(v2.16), we can make it optional, and if the releases go well, we can make it mandatory of the release from v2.17.
    - Should it be done by the same release owner?
        - Yes, a script and a release instruction will be provided..
    - Should the validation be part of the release validation?
        - Yes, when we validate other release artifacts.

Docker images

Language	Supported versions	Docker image name	Image size before compression	Image size after compression (on gcr)
Python	2.7	python2.7_sdk	1.82GB	563MB
	3.5	python3.5_sdk	1.83GB	568MB
	3.6	python3.6_sdk	1.84GB	569MB
	3.7	python3.7_sdk	1.85GB	575MB
Java	8	java_sdk	550MB	255MB
	11	not available	-	-
Go	1.12	go_sdk	125MB	54MB

Commands

This section describes the commands that are used to build, publish, run tests and examples for the images.

Prerequisites

Docker should be installed.

```
$ docker -v
Docker version 18.09.3, build 774a1f4
```

Staging release images

Shell script

Publishing release images

Shell script

Release Images Validation

This section describes how to validate a built and/or published image.

Validation testing

Test on Dataflow with uploaded images.

```
# for Python
# this should run against py2.7, py3.5, py3.6 and py3.7.
$ pwd
[...]beam/sdks/python

$ python -m apache_beam.examples.wordcount \
--input gs://apache-beam-samples/shakespeare/hamlet.txt \
--output gs://temp-storage-for-end-to-end-tests/staging-$USER/output \
--runner DataflowRunner \
--project apache-beam-testing \
--temp_location gs://temp-storage-for-end-to-end-tests/staging-$USER/\
--worker_harness_container_image $REPOSITORY/python2.7_sdk:$TAG \
--experiment beam_fn_api \
```

```
--sdk_location sdks/python/container/py2/build/target/apache-beam.tar.gz
```

```
# for Java
$ pwd
[...]beam/sdks/java

# get WordCount example code as a maven project
$ mvn archetype:generate \
      -DarchetypeGroupId=org.apache.beam \
      -DarchetypeArtifactId=beam-sdks-java-maven-archetypes-examples \
      -DarchetypeVersion=2.6.0 \
      -DgroupId=org.example \
      -DartifactId=word-count-beam \
      -Dversion="0.1" \
      -Dpackage=org.apache.beam.examples \
      -DinteractiveMode=false

# run Java project
$ pwd
[...]/word-count-beam

$ mvn compile exec:java
-Dexec.mainClass=org.apache.beam.examples.WordCount -Dexec.args="\
--runner=DataflowRunner \
--project=apache-beam-testing \
--stagingLocation=gs://temp-storage-for-end-to-end-tests/staging-$USER/\
--workerHarnessContainerImage=$REPOSITORY/java_sdk:$TAG \
--experiments=beam_fn_api \
--output=gs://temp-storage-for-end-to-end-tests/staging-$USER/output" \
-Pdataflow-runner
```

```
# for Go
$ pwd
[...]beam/sdks/go

$ go run examples/wordcount/wordcount.go \
--runner=dataflow \
--project=apache-beam-testing \
--staging_location=gs://temp-storage-for-end-to-end-tests/staging-$USER/\
--worker_harness_container_image=$REPOSITORY/go_sdk:$TAG \
--output=gs://temp-storage-for-end-to-end-tests/staging-$USER/output
```

Push to final release location

```
export NEW_REPOSITORY=gcr.io/apache-beam-testing/beam/sdks/release
# for Python
docker tag $REPOSITORY/python2.7:$TAG $NEW_REPOSITORY/python2.7:$TAG
docker push $NEW_REPOSITORY/python2.7:$TAG

docker tag $REPOSITORY/python3.5:$TAG $REPOSITORY/python3.5:$NEW_TAG
docker tag $REPOSITORY/python3.6:$TAG $REPOSITORY/python3.6:$NEW_TAG
docker tag $REPOSITORY/python3.7:$TAG $REPOSITORY/python3.7:$NEW_TAG

# for Java
docker tag $REPOSITORY/java:$TAG $REPOSITORY/java:$NEW_TAG

# for Go
docker tag $REPOSITORY/go:$TAG $REPOSITORY/go:$NEW_TAG
```

Backwards compatibility

- Do we want new images to be able to run old pipelines?
    - This is decided by SDK, not container specific.
- How long do we support backwards compatibility for?
    - This is decided by SDK, not container specific.

Other verification

- Do we sign the artifacts, images?
    - ??
- Do we check hashes, signatures?
    - No
- How to ensure that Python container image contains all dependencies of Apache Beam SDK, all versions are compatible and there are no dependency conflicts? Some ideas discussed on dev mailing list:
    - List all dependencies of Beam Python SDK, including transitive dependencies in base_image_requirements.txt. This could be done in a two step process:
        - use a human-generated requirements file like base_image_requirements today which has a set of curated requirements. The human-generated file should be periodically updated (as a separate process, independent of SDK release)
        - Changes to the first file would result in a generated file with all transitive dependencies. Second file could be used as the source of truth for all dependencies at a particular commit. Generated file could be used for the container builds.

- During container build, check that there are no dependency conflicts (they typically look like: Package X requires version A of dependency Y, but you will have B, which is incompatible).
- During container build, we verify that at the moment of Apache Beam installation, no new dependencies are pulled from PyPi.
    - One possibility is to run apache beam installation command without access to internet, so that a successful installation is a signal that all dependencies are present.
        - unshare -n -r pip install apache-beam-2.15.0.dev0.tar.gz

Use cases

How do we allow the images to be used?

1. Snapshot images can be used for daily testing.
2. Snapshot images are always created from head, so users can use Beam at head if they want to.
3. Customize containers on top of published images.
4. ...

Do we support users using them in production as is?

Release images can be used in production. Snapshot images can be used in production, but we don't guarantee they are as stable as release images.

Is there some quota for downloading the prebuilt images?

From gCloud instruction, gcr has following quota limitations.

Any request sent to Container Registry has a 2 hour timeout limit.
The fixed rate limits per client IP address are:

- 30,000 HTTP requests every 10 minutes
- 500,000 HTTP requests per day

Container Registry uses Cloud Storage for each registry's underlying storage. Cloud Storage Quotas & Limits apply to each registry.

Any special/extra license needs to be attached to the images?

Yes, need to include licenses/notices for third party dependencies for each image.