Docker Workload Onboarding

How to use docker containers with Bacalhau

Docker Workloads

Bacalhau executes jobs by running them within containers. Bacalhau employs a syntax closely resembling Docker, allowing you to utilize the same containers. The key distinction lies in how input and output data are transmitted to the container via IPFS, enabling scalability on a global level.

This section describes how to migrate a workload based on a Docker container into a format that will work with the Bacalhau client.

> You can check out this example tutorial on how to work with custom containers in Bacalhau to see how we used all these steps together.

Requirements

Here are few things to note before getting started:

1. **Container Registry**: Ensure that the container is published to a public container registry that is accessible from the Bacalhau network.
2. **Architecture Compatibility**: Bacalhau supports only images that match the host node's architecture. Typically, most nodes run on `linux/amd64`, so containers in `arm64` format are not able to run.
3. **Input Flags**: The `--input ipfs://...` flag supports only **directories** and does not support CID subpaths. The `--input https://...` flag supports only **single files** and does not support URL directories. The `--input s3://...` flag supports S3 keys and prefixes. For example, `s3://bucket/logs-2023-04*` includes all logs for April 2023.

> You can check to see a list of example public containers used by the Bacalhau team

> **Note**: Only about a third of examples have their containers here. The rest are under random docker hub registries.

Runtime Restrictions

To help provide a safe, secure network for all users, we add the following runtime restrictions:

1. **Limited Ingress/Egress Networking**:

All ingress/egress networking is limited as described in the networking documentation. You won't be able to pull `data/code/weights/` etc. from an external source.

2. **Data Passing with Docker Volumes**:

A job includes the concept of input and output volumes, and the Docker executor implements support for these. This means you can specify your CIDs, URLs, and/or S3 objects as `input` paths and also write results to an `output` volume. This can be seen in the following example:

```
bacalhau docker run \
  -i s3://mybucket/logs-2023-04*:/input \
  -o apples:/output_folder \
  ubuntu \
  bash -c 'ls /input > /output_folder/file.txt'
```

The above example demonstrates an input volume flag `-i s3://mybucket/logs-2023-04*`, which mounts all S3 objects in bucket `mybucket` with `logs-2023-04` prefix within the docker container at location `/input` (root).

Output volumes are mounted to the Docker container at the location specified. In the example above, any content written to `/output_folder` will be made available within the `apples` folder in the job results CID.

Once the job has run on the executor, the contents of `stdout` and `stderr` will be added to any named output volumes the job has used (in this case `apples`), and all those entities will be packaged into the results folder which is then published to a remote location by the publisher.

Onboarding Your Workload

Step 1 - Read Data From Your Directory

If you need to pass data into your container you will do this through a Docker volume. You'll need to modify your code to read from a local directory.

We make the assumption that you are reading from a directory called `/inputs`, which is set as the default.

> You can specify which directory the data is written to with the `--input` CLI flag.

Step 2 - Write Data to the Your Directory

If you need to return data from your container you will do this through a Docker volume. You'll need to modify your code to write to a local directory.

We make the assumption that you are writing to a directory called `/outputs`, which is set as the default.

> You can specify which directory the data is written to with the `--output-volumes` CLI flag.

Step 3 - Build and Push Your Image To a Registry

At this step, you create (or update) a Docker image that Bacalhau will use to perform your task. You build your image from your code and dependencies, then push it to a public registry so that Bacalhau can access it. This is necessary for other Bacalhau nodes to run your container and execute the given task.

> Most Bacalhau nodes are of an `x86_64` architecture, therefore containers should be built for `x86_64` systems.

For example:

```
$ export IMAGE=myuser/myimage:latest
$ docker build -t ${IMAGE} .
$ docker image push ${IMAGE}
```

Step 4 - Test Your Container

To test your docker image locally, you'll need to execute the following command, changing the environment variables as necessary:

```
$ export LOCAL_INPUT_DIR=$PWD
$ export LOCAL_OUTPUT_DIR=$PWD
$ export CMD=(sh -c 'ls /inputs; echo do something useful > /outputs/stdout')
$ docker run --rm \
  -v ${LOCAL_INPUT_DIR}:/inputs  \
  -v ${LOCAL_OUTPUT_DIR}:/outputs \
  ${IMAGE} \
  ${CMD}
```

Let's see what each command will be used for:

```
$ export LOCAL_INPUT_DIR=$PWD
Exports the current working directory of the host system to the LOCAL_INPUT_DI

$ export LOCAL_OUTPUT_DIR=$PWD
Exports the current working directory of the host system to the LOCAL_OUTPUT_D

$ export CMD=(sh -c 'ls /inputs; echo do something useful > /outputs/stdout')
Creates an array of commands CMD that will be executed inside the container. I

$ docker run ... ${IMAGE} ${CMD}
Launches a Docker container using the specified variables and commands. It bir
```

Bacalhau will use the default ENTRYPOINT if your image contains one. If you need to specify another entrypoint, use the `--entrypoint` flag to `bacalhau docker run`.

For example:

```
$ export LOCAL_INPUT_DIR=$PWD
$ export LOCAL_OUTPUT_DIR=$PWD
$ export CMD=(sh -c 'ls /inputs; echo "do something useful" > /outputs/stdout'
$ export IMAGE=ubuntu
$ docker run --rm \
  -v ${LOCAL_INPUT_DIR}:/inputs  \
  -v ${LOCAL_OUTPUT_DIR}:/outputs \
  ${IMAGE} \
  ${CMD}
$ cat stdout
```

The result of the commands' execution is shown below:

```
do something useful
```

Step 5 - Upload the Input Data

Data is identified by its content identifier (CID) and can be accessed by anyone who knows the CID. You can use either of these methods to upload your data:

Copy data from a URL to public storage

Pin Data to public storage

Copy Data from S3 Bucket to public storage

> You can mount your data anywhere on your machine, and Bacalhau will be able to run against that data

Step 6 - Run the Workload on Bacalhau

To launch your workload in a Docker container, using the specified image and working with `input` data specified via IPFS CID, run the following command:

```
$ bacalhau docker run --input ipfs://${CID} ${IMAGE} ${CMD}
```

To check the status of your job, run the following command:

```
$ bacalhau job list --id-filter JOB_ID
```

To get more information on your job,run:

```
$ bacalhau job describe JOB_ID
```

To download your job, run:

```
$ bacalhau job get JOB_ID
```

For example, running:

```
JOB_ID=$(bacalhau docker run ubuntu echo hello | grep 'Job ID:' | sed 's/.*Job
echo "The job ID is: $JOB_ID"
bacalhau job list --id-filter $JOB_ID
sleep 5

bacalhau job list --id-filter $JOB_ID
bacalhau get $JOB_ID

ls shards
```

outputs:

```
CREATED   ID        JOB                      STATE      VERIFIED  PUBLISHED
 10:26:00  24440f0d  Docker ubuntu echo h...  Verifying
 CREATED   ID        JOB                       STATE      VERIFIED  PUBLISHED
 10:26:00  24440f0d  Docker ubuntu echo h...  Published            /ipfs/bafyb
11:26:09.107 | INF bacalhau/get.go:67 > Fetching results of job '24440f0d-3c06
11:26:10.528 | INF ipfs/downloader.go:115 > Found 1 result shards, downloading
11:26:13.144 | INF ipfs/downloader.go:195 > Combining shard from output volume
job-24440f0d-3c06-46af-9adf-cb524aa43961-shard-0-host-QmYgxZiySj3MRkwLSL4X2MF5
```

> The `--input` flag does not support CID subpaths for `ipfs://` content.

Alternatively, you can run your workload with a publicly accessible http(s) URL, which will download the data temporarily into your public storage:

```
$ export URL=https://download.geofabrik.de/antarctica-latest.osm.pbf
$ bacalhau docker run --input ${URL} ${IMAGE} ${CMD}

$ bacalhau job list

$ bacalhau job get JOB_ID
```

> The `--input` flag does not support URL directories.

Troubleshooting

If you run into this compute error while running your docker image

```
Creating job for submission ... done ✅
Finding node(s) for the job ... done ✅
Node accepted the job ... done ✅
Error while executing the job.
```

This can often be resolved by re-tagging your docker image

Support

If you have questions or need support or guidance, please reach out to the Bacalhau team via Slack (**#general** channel)

Previous
Container

Next
WebAssembly (WASM) Workloads

Last updated 7 days ago