Running a Job over S3 data

Here is a quick tutorial on how to copy Data from S3 to a public storage. In this tutorial, we will scrape all the links from a public AWS S3 buckets and then copy the data to IPFS using Bacalhau.

Prerequisite

To get started, you need to install the Bacalhau client, see more information here

Running a Bacalhau Job

```
%%bash --out job_id
bacalhau docker run \
-i "s3://noaa-goes16/ABI-L1b-RadC/2000/001/12/OR_ABI-L1b-RadC-
M3C01*:/inputs,opt=region=us-east-1" \
--id-only \
--wait \
alpine \
-- sh -c "cp -r /inputs/* /outputs/"
```

Structure of the Command

Let's look closely at the command above:

- `bacalhau docker run`: call to bacalhau
- `-i "s3://noaa-goes16/ABI-L1b-RadC/2000/001/12/OR_ABI-L1b-RadC-M3C01*:/inputs,opt=region=us-east-1`: defines S3 objects as inputs to the job. In this case, it will download all objects that match the prefix `ABI-L1b-RadC/2000/001/12/OR_ABI-L1b-RadC-M3C01` from the bucket `noaa-goes16` in `us-east-1` region, and mount the objects under `/inputs` path inside the docker job.
- `-- sh -c "cp -r /inputs/* /outputs/"`: copies all files under `/inputs` to `/outputs`, which is by default the result output directory which all of its content will be published to the specified destination, which is IPFS by default

When a job is submitted, Bacalhau prints out the related `job_id`. We store that in an environment variable so that we can reuse it later on.

> **TIP**
>
> This only works with datasets that are publicly accessible and don't require an AWS account or pay to use buckets.

Checking the State of your Jobs

- **Job status**: You can check the status of the job using `bacalhau list`.

```
%%bash
bacalhau list --id-filter ${JOB_ID} --wide
```

When it says `Published` or `Completed`, that means the job is done, and we can get the results.

- **Job information**: You can find out more information about your job by using `bacalhau describe`.

```
%%bash
bacalhau describe ${JOB_ID}
```

- **Job download**: You can download your job results directly by using `bacalhau get`. Alternatively, you can choose to create a directory to store your results. In the command below, we created a directory and downloaded our job output to be stored in that directory.

```
%%bash
rm -rf results && mkdir -p results # Temporary directory to store the results
bacalhau get $JOB_ID --output-dir results # Download the results
```

After the download has finished you should see the following contents in results directory.

Viewing your Job Output

To view your file, run the following command:

```
%%bash
ls -1 results/outputs
```

OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20170671748180.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20170691603180.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20170751219598.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20170752149454.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20170752204183.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20170752234173.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20170901216521.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20170951807462.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20171000619157.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20171061215161.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20171071918365.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20171091517487.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20171152112459.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20171221432456.nc

OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20171232313205.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20171301618116.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20171572234151.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20171592127442.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20171801512461.nc
OR_ABI-L1b-RadC-M3C01_G16_s20000011200000_e20000011200000_c20171941452463.nc

Extract Result CID

Installing jq to extract CID from the result description

```
%%bash
sudo apt update
sudo apt install jq
Extracting the CIDs from output json
```

```
%%bash
bacalhau describe ${JOB_ID} --json \
| jq -r '.State.Executions[].PublishedResults.CID | select (. != null)'
```

QmYFhG668yJZmtk84SMMdbrz5Uvuh78Q8nLxTgLDWShkhR

Publishing Results to S3-Compatible Destinations

You can publish your results to Amazon s3 or other S3-compatible destinations like MinIO, Ceph, or SeaweedFS to conveniently store and share your outputs.

Publisher Spec

To facilitate publishing results, define publishers and their configurations using the PublisherSpec structure.

For S3-compatible destinations, the configuration is as follows:

```
type PublisherSpec struct {
    Type   Publisher              `json:"Type,omitempty"`
    Params map[string]interface{} `json:"Params,omitempty"`
}
```

For Amazon S3, you can specify the PublisherSpec configuration as shown below:

```
PublisherSpec:
  Type: S3
  Params:
    Bucket: <bucket>              # Specify the bucket where results will be
```

```
stored
    Key: <object-key>             # Define the object key (supports dynamic
naming using placeholders)
    Compress: <true/false>        # Specify whether to publish results as a
single gzip file (default: false)
    Endpoint: <optional>          # Optionally specify the S3 endpoint
    Region: <optional>            # Optionally specify the S3 region
```

Example Usage

Let's explore some examples to illustrate how you can use this:

- Publishing results to S3 using default settings

```
bacalhau docker run -p s3://<bucket>/<object-key> ubuntu ...
```

- Publishing results to S3 with a custom endpoint and region:

```
bacalhau docker run -p s3://<bucket>/<object-
key>,opt=endpoint=http://s3.example.com,opt=region=us-east-1 ubuntu ...
```

- Publishing results to S3 as a single compressed file

```
bacalhau docker run -p s3://<bucket>/<object-key>,opt=compress=true ubuntu ...
```

- Utilizing naming placeholders in the object key

```
bacalhau docker run -p s3://<bucket>/result-{date}-{jobID} ubuntu ...
```

Content Identification

Tracking content identification and maintaining lineage across different jobs' inputs and outputs can be challenging. To address this, the publisher encodes the SHA-256 checksum of the published results, specifically when publishing a single compressed file.

Here's an example of a sample result:

```
{
    "NodeID": "QmYJ9QN9Pbi6gBKNrXVk5J36KSDGL5eUT6LMLF5t7zyaA7",
    "Data": {
        "StorageSource": "S3",
        "Name": "s3://<bucket>/run3.tar.gz",
        "S3": {
            "Bucket": "<bucket>",
            "Key": "run3.tar.gz",
            "Checksum": "e0uDqmflfT9b+rMfoCnO5G+cy+8WVTOPUtAqDMnXWbw=",
            "VersionID": "hZoNdqJsZxE_bFm3UGJuJ0RqkITe9dQ1"
        }
```

```
    }
}
```

Support for the S3-compatible storage provider

To enable support for the S3-compatible storage provider, no additional dependencies are required. However, valid AWS credentials are necessary to sign the requests. The storage provider uses the default credentials chain, which checks the following sources for credentials:

- Environment variables, such as `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`
- Credentials file `~/.aws/credentials`
- IAM Roles for Amazon EC2 Instances

Need Support?

For questions, feedback, please reach out in our forum