Bacalhau API overview

> Note that in version 1.4.0 API logic and endpoints have changed. Check out the release notes and updated API description in the API documentation section.

Welcome to the official API documentation for Bacalhau. This guide provides a detailed insight into Bacalhau's RESTful HTTP APIs and demonstrates how to make the most out of them.

Overview

Bacalhau prioritizes an "API-first" design, enabling users to interact with their deployed systems programmatically. In the `v1.4.0` the API model was changed to include only two endpoints, focused on orchestrating, querying and managing your network nodes and jobs. Each endpoint has a clear, separate environment and goal, allowing to manage coordination between nodes, jobs, and executions more effectively.

- **Endpoint Prefix**: All APIs are versioned and prefixed with `/api/v1`.
- **Default Port**: By default, Bacalhau listens on port `1234`.

API endpoints

Orchestrator

The Majority of Bacalhau's functionality is channeled through the `Orchestrator` endpoint and its operations. It handles user requests and schedules and it is critical for creating, managing, monitoring, and analyzing jobs within Bacalhau. It also provides mechanisms to query information about the nodes in the cluster.

```
api/v1/orchestrator/
```

Here's the job submission format, where you can tag a YAML file with the job specifications or input the commands with your CLI

```
# Submit a job
curl -X PUT \
     -H "Content-Type: application/json" \
     -d '{
          "Job": {
            "Name": "test-job",
            "Namespace": "default",
            "Type": "batch",
            "Count": 1,
            "Labels": {
              "foo": "bar",
              "env": "dev"
            },
            "Tasks": [
              {
                "Name": "task1",
                "Engine": {
                  "Type": "docker",
                  "Params": {
                    "Image": "ubuntu:latest",
                    "Entrypoint": [
                      "echo",
                      "hello"
                    ]
                  }
                },
                "Publisher": {
                  "Type": "noop",
                  "Params": {}
                },
                "ResourcesConfig": {
                  "CPU": "0.1",
                  "Memory": "10mb"
                },
                "Network": {
                  "Type": "None"
```

```
                    "Type": "None"
                  },
                  "Timeouts": {
                    "ExecutionTimeout": 30
                  }
                }
              ]
            }
          }' \
      http://0.0.0.0:20000/api/v1/orchestrator/jobs

{"JobID":"28c08f7f-6fb0-48ed-912d-a2cb6c3a4f3a","Eva
```

Agent

This endpoint offers a convenient route to collate detailed information about the Bacalhau node you're interacting with, whether it's acting as the orchestrator or a compute node. It provides you with insights into the node's health, capabilities, and the deployed Bacalhau version.

```
api/v1/agent/node
```

Here's the command structure for querying your current node. You can check on its status and collate information on its health and capabilities:

```
# Is alive
curl 0.0.0.0:20000/api/v1/agent/alive
```

Features

Pagination

To handle large datasets, Bacalhau supports pagination. Users can define the `limit` in their request and then utilize the `next_token` from the response to fetch subsequent data chunks.

Ordering

To sort the results of list-based queries, use the `order_by` parameter. By default, the list will be sorted in ascending order. If you want to reverse it, use the `reverse` parameter. Note that the fields available for sorting might vary depending on the specific API endpoint.

Pretty JSON Output

By default, Bacalhau's APIs provide a minimized JSON response. If you want to view the output in a more readable format, append `pretty` to the query string.

HTTP Methods

Being RESTful in nature, Bacalhau's API endpoints rely on standard HTTP methods to perform various actions:

1. **GET**: Fetch data.
2. **PUT**: Update or create data.
3. **DELETE**: Remove data.

The behavior of an API depends on its HTTP method. For example, `/api/v1/orchestrator/jobs`:

1. **GET**: Lists all jobs.
2. **PUT**: Submits a new job.
3. **DELETE**: Stops a job.

HTTP Response Codes

Understanding HTTP response codes is crucial. A `2xx` series indicates a successful operation, `4xx` indicates client-side errors, and `5xx` points to server-side issues. Always refer to the message accompanying the code for more information.

Since `/api/v1/requester/*` was changed to `/api/v1/orchestrator/` in `v1.4.0`, all `/api/v1/requester/*` requests will result in 410 error.

Previous
API Guide

Next
Best Practices

Last updated 23 days ago