Bootstrap requests

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/v1/bootstrap</td>
<td>Get the local node's bootstrap status</td>
</tr>
<tr>
<td>POST</td>
<td>/v1/bootstrap/{action}</td>
<td>Initiate bootstrapping</td>
</tr>
</tbody>
</table>

Get bootstrap status

GET /v1/bootstrap

Get the local node's bootstrap status.

This request is accepted as soon the cluster software is installed and before the node is part of an active cluster.

Once the node is part of an active cluster, authentication is required.

Request

Example HTTP request

GET /bootstrap

Request headers

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>application/json</td>
<td>Accepted media type</td>
</tr>
</tbody>
</table>

Response

The JSON response object contains two other objects:

- `bootstrap_status` which is described below
- `local_node_info` which is a subset of a `node object` that provides information about the node configuration

bootstrap_status object:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>state</td>
<td>Current bootstrap state.</td>
</tr>
<tr>
<td>idle: No bootstrapping started.</td>
<td></td>
</tr>
<tr>
<td>initiated: Bootstrap request received.</td>
<td></td>
</tr>
<tr>
<td>creating_cluster: In the process of creating a new cluster.</td>
<td></td>
</tr>
<tr>
<td>joining_cluster: In the process of joining an existing cluster.</td>
<td></td>
</tr>
<tr>
<td>error: The last bootstrap action failed.</td>
<td></td>
</tr>
<tr>
<td>completed: The last bootstrap action completed successfully.</td>
<td></td>
</tr>
<tr>
<td>start_time</td>
<td>Bootstrap process start time</td>
</tr>
<tr>
<td>end_time</td>
<td>Bootstrap process end time</td>
</tr>
<tr>
<td>error_code</td>
<td>If state is error, this error code describes the type of error encountered.</td>
</tr>
<tr>
<td>error_details</td>
<td>An error-specific object that may contain additional information about the error. A common field in use is message which provides a more verbose error message.</td>
</tr>
</tbody>
</table>

**Example JSON body**

```json
{
  "state": "error",
  "start_time": "2023-04-01T12:00:00Z",
  "end_time": "2023-04-01T13:00:00Z",
  "error_code": "connection_error",
  "error_details": {
    "message": "Failed to connect to server"}
}
```
```json
{
    "bootstrap_status": {
        "start_time": "2014-08-29T11:19:49Z",
        "end_time": "2014-08-29T11:19:49Z",
        "state": "completed"
    },
    "local_node_info": {
        "uid": 3,
        "software_version": "0.90.0-1",
        "cores": 2,
        "ephemeral_storage_path": "/var/opt/redislabs/tmp",
        "ephemeral_storage_size": 1018889.8304,
        "os_version": "Ubuntu 14.04 LTS",
        "persistent_storage_path": "/var/opt/redislabs/persist/redis",
        "persistent_storage_size": 1018889.8304,
        "total_memory": 24137,
        "uptime": 50278,
        "available_addrs": [{
            "address": "172.16.50.122",
            "format": "ipv4",
            "if_name": "eth0",
            "private": true
        },
        {
            "address": "10.0.3.1",
            "format": "ipv4",
            "if_name": "lxcbr0",
            "private": true
        },
        {
            "address": "172.17.0.1",
            "format": "ipv4",
            "if_name": "docker0",
            "private": true
        },
        {
            "address": "2001:db8:0:f101::1",
            "format": "ipv6",
            "if_name": "eth0",
            "private": false
        }]
    }
}
```

### Error codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config_error</td>
<td>An error related to the bootstrap configuration provided (e.g. bad JSON).</td>
</tr>
<tr>
<td>connect_error</td>
<td>Failed to connect to cluster (e.g. FQDN DNS could not resolve, no/wrong node IP provided, etc.)</td>
</tr>
<tr>
<td>access_denied</td>
<td>Invalid credentials supplied.</td>
</tr>
</tbody>
</table>
invalid_license
The license string provided is invalid. Additional info can be fetched from the `error_details` object, which includes the violation code in case the license is valid but its terms are violated.

repair_required
Cluster is in degraded mode and can only accept replacement nodes. When this happens, `error_details` contains two fields: `failed_nodes` and `replace_candidate`. The `failed_nodes` field is an array of objects, each describing a failed node with at least a `uid` field and an optional `rack_id`. `replace_candidate` is the UID of the node most suitable for replacement.

insufficient_node_memory
An attempt to replace a dead node fails because the replaced node does not have enough memory. When this happens, `error_details` contains a `required_memory` field which indicates the node memory requirement.

insufficient_node_flash
An attempt to replace a dead node fails because the replaced node does not have enough flash. When this happens, `error_details` contains a `required_flash` field which indicates the node flash requirement.

time_not_sync
An attempt to join a node with system time not synchronized with the rest of the cluster.

rack_id_required
An attempt to join a node with no `rack_id` in a rack-aware cluster. In addition, a `current_rack_ids` field will include an array of currently used rack ids.

socket_directory_mismatch
An attempt to join a node with a socket directory setting that differs from the cluster.

node_config_mismatch
An attempt to join a node with a configuration setting (e.g. `confdir`, `osuser`, `installdir`) that differs from the cluster.

path_error
A needed path does not exist or is not accessible.

internal_error
A different, unspecified internal error was encountered.

### Status codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 OK</td>
<td>No error</td>
</tr>
</tbody>
</table>

### Start bootstrapping

**POST /v1/bootstrap/{action}**

Initiate bootstrapping.

The request must contain a bootstrap configuration JSON object, as described in Object attributes or a minimal subset.

Bootstrapping is permitted only when the current bootstrap state is `idle` or `error` (in which case the process will restart with the new configuration).

This request is asynchronous - once the request has been accepted, the caller is expected to poll bootstrap status while waiting for it to complete.
Request

Example HTTP request

POST /bootstrap/create_cluster

Example JSON body

Join cluster

```json
{
  "action": "join_cluster",
  "cluster": {
    "nodes": ["1.1.1.1", "2.2.2.2"]
  },
  "node": {
    "paths": {
      "persistent_path": "/path/to/persistent/storage",
      "ephemeral_path": "/path/to/ephemeral/storage",
      "bigstore_path": "/path/to/bigstore/storage"
    },
    "bigstore_driver": "rocksdb",
    "identity": {
      "addr": "1.2.3.4",
      "external_addr": ["2001:0db8:85a3:0000:0000:8a2e:0370:7334", "3.4.5.6"]
    }
  },
  "credentials": {
    "username": "my_username",
    "password": "my_password"
  }
}
```

Create cluster
{  
  "action": "create_cluster",
  "cluster": {
    "nodes": [],
    "name": "my.cluster"
  },
  "node": {
    "paths": {
      "persistent_path": "/path/to/persistent/storage",
      "ephemeral_path": "/path/to/ephemeral/storage",
      "bigstore_path": "/path/to/bigredis/storage"
    },
    "identity": {
      "addr": "1.2.3.4",
      "external_addr": ["2001:0db8:85a3:0000:0000:8a2e:0370:7334", "3.4.5.6"]
    },
    "bigstore_driver": "rocksdb"
  },
  "license": "",
  "credentials": {
    "username": "my_username",
    "password": "my_password"
  }
}

Request headers

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>cnm.cluster.fqdn</td>
<td>Domain name</td>
</tr>
<tr>
<td>Accept</td>
<td>application/json</td>
<td>Accepted media type</td>
</tr>
<tr>
<td>Host</td>
<td>cnm.cluster.fqdn</td>
<td>Domain name</td>
</tr>
<tr>
<td>Accept</td>
<td>application/json</td>
<td>Accepted media type</td>
</tr>
</tbody>
</table>

Request body

Include a **bootstrap object** in the request body.

Response

Status codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 OK</td>
<td>Request received and processing begins.</td>
</tr>
<tr>
<td>409 Conflict</td>
<td>Bootstrap already in progress (check state)</td>
</tr>
</tbody>
</table>

Updated: April 6, 2022