elastisys:scale cloud adapter REST API Documentation

Release 2.0.0

Elastisys AB

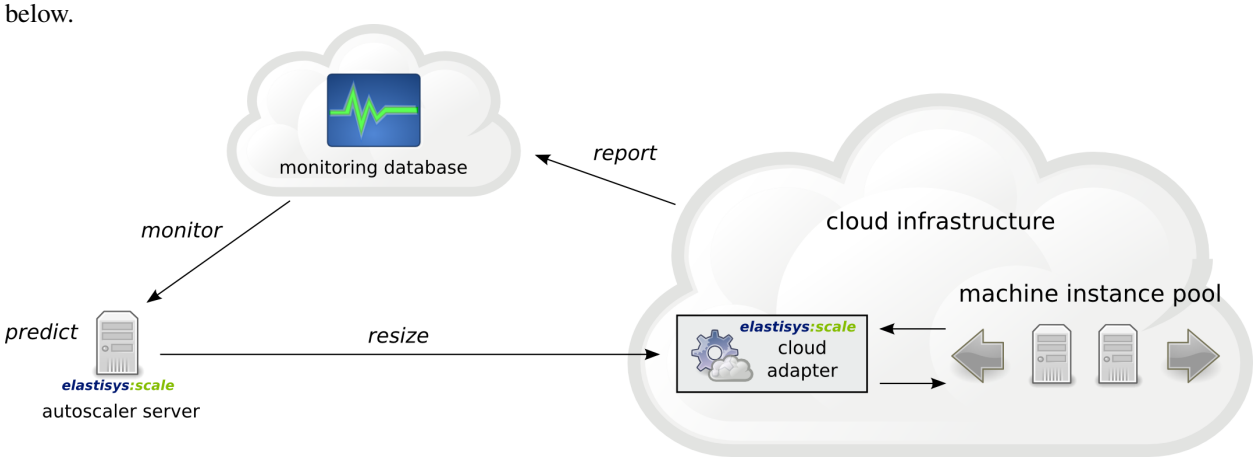
March 05, 2015

Contents

This documentation covers version 2.0 of the REST API that all elastisys:scale cloud adapters are required to publish.

A *cloud adapter* is a key component in an elastisys:scale autoscaling setup.

An elastisys:scale autoscaling system consists of two main parts: (1) an *autoscaler server* and (2) a *cloud adapter*. The autoscaler server collects monitoring data reported by the application from a monitoring database and applies scaling algorithms to, ultimately, emit a number of required VM instances to keep the auto-scaling-enabled service running smoothly. This number is communicated over a secured (SSL) channel to the cloud adapter, which instructs the cloud infrastructure to add or remove VMs, as appropriate. A schematical overview of the system is shown in the image below.



So a cloud adapter manages an elastic pool of machines for a particular cloud provider, handling communication with the cloud provider according to its API. The cloud adapter provides a cloud-neutral API to clients, such as the autoscaler, with a number of management primitives for the machine pool. In general terms, these primitives allow clients to:

- track the machine pool members and their states
- modify the size of the machine pool (the cloud adapter continuously starts/stops machine instances so that the number of machines in the pool matches the desired size set for the pool).

The interface between the autoscaler and the cloud adapter is a REST API. All cloud adapters are required to implement this REST API and make it available over secure HTTP (HTTPS). The REST API is described in greater detail in the *elastisys:scale cloud adapter REST API*.

Documentation:

CHAPTER 1

elastisys:scale cloud adapter REST API

All elastisys:scale cloud adapters are required to publish the REST API described below.

The primary purpose of the cloud adapter API is to serve as a bridge between an autoscaler and a certain cloud provider, allowing the autoscaler to operate in a cloud-neutral manner. As such, it focuses on primitives for managing a dynamic collection of machines.

All API methods assume a `Content-Type` of `application/json`.

The REST API should be made available over secure HTTP (HTTPS).

1.1 Terminology and machine state model

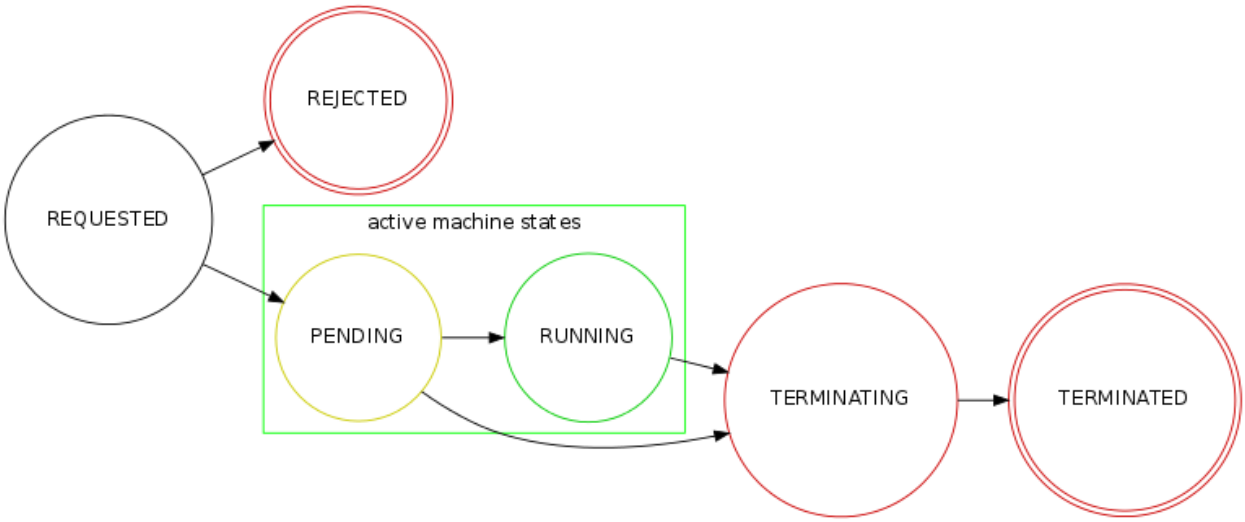
Cloud providers differ in how they refer to the computational resources they provide. Some common terms are *instances*, *servers* and *VMs*/*virtual machines*.

The cloud adapter API strives to be as cloud-neutral as possible and simply refers to the computational resources being managed as **machines**. The logical group of machines that a cloud adapter manages is referred to as its **machine pool**.

A cloud adapter needs to be able to report the execution state of its machine pool members in a cloud-neutral manner (see *Get machine pool*). Since cloud providers differ quite a lot in the state models they use, the cloud adapter needs to map the cloud-native state of the machine to one of the **machine states** supported by the cloud adapter API. These states are described in the *machine state table* below.

Machine state	Description
`REQUESTED`	The machine has been requested from the underlying infrastructure and the request is pending fulfillment.
`REJECTED`	The machine request was rejected by the underlying infrastructure.
`PENDING`	The machine is in the process of being launched.
`RUNNING`	The machine is launched. However, the boot process may not yet have completed and the machine may not be operational (the machine's *service state* may provide more detailed state information).
`TERMINATING`	The machine is in the process of being stopped/shut down.
`TERMINATED`	The machine has been stopped/shut down.

The diagram below illustrates the state transitions that describe the lifecycle of a machine.



The PENDING and RUNNING states are said to be the *active machine states*. Machines in an active state are executing. However, just because a machine is active doesn't necessarily mean that it is doing useful work. For example, it may have failed to properly boot, it may have crashed or encountered a fatal bug.

There are cases where we need to be able to reason about the operational state of the service running on the machine. For example, we may not want to register a running machine to a load balancer until it is fully initialized and ready to accept requests. To this end, a cloud adapter may include a *service state* for a machine. Whereas the *machine state* should be viewed as the execution state of the machine as reported by the cloud API, the **service state** of a machine is the operational health of the service running on the machine.

A cloud adapter does not need to monitor the service state (health) of its machines, although it could. However, a cloud adapter is required to be ready to receive service state updates (see *Set service state*) for the machines in its pool and to include those states on subsequent queries about the pool members (*Get machine pool*). A human operator or an external monitoring service should be able to set the service state for a certain machine. In case no service state has been reported for a machine, the cloud adapter should report the machine as being in a service state of UNKNOWN.

The range of permissible service states are as follows:

Service state	Description
BOOTING	The service is being bootstrapped and may not (yet) be operation
IN_SERVICE	The service is operational and ready to accept work (health chec
UNHEALTHY	The service is not functioning properly (health checks fail).
OUT_OF_SERVICE	The service is unhealthy and is in need of repair. It should be co able to accept work until it has recovered. When this service stat replacement instance will be launched, as the machine is no long machine pool (even if it is in machine state RUNNING).
UNKNOWN	The service state of the machine cannot be (or has not yet been)

The OUT_OF_SERVICE state marks a machine pool member as being taken out of service and awaiting troubleshooting. The cloud adapter should take measures to ensure that a replacement machine is launched for any out-of-service machine.

Except for the OUT_OF_SERVICE state, service states are only informational "marker states" that, for example, can be used to monitor service health and register machines with a load balancer as they become IN_SERVICE and unregister them when in state UNHEALTHY or OUT_OF_SERVICE.

At any time, the *effective size* of the machine pool should be interpreted as the number of allocated machines that have not been marked out-of-service (awaiting troubleshooting/repair). That is, all machines in one of the machine states REQUESTED, PENDING, or RUNNING and *not* being in service state OUT_OF_SERVICE.

1.2 Operations

1.2.1 Get machine pool

- **Method**: `GET /pool`
- **Description**: Retrieves the current machine pool members.

  Note that the returned machines may be in any *machine state* (`REQUESTED`, `RUNNING`, `TERMINATED`, etc). The machines that are considered part of the active pool are machines in machine states `REQUESTED`, `PENDING` or `RUNNING` and that are *not* in service state `OUT_OF_SERVICE`. The *service state* should be reported as `UNKNOWN` for machine instances that have not had any service state reported (see *Set service state*).
- **Input**: None
- **Output:**
  - On success: HTTP response code 200 with a *Machine pool message*
  - On error: HTTP response code 500 with an *Error response message*

1.2.2 Get pool size

- **Method**: `GET /pool/size`
- **Description**: Returns the current size of the machine pool – both in terms of the desired size and the actual size (as these may differ at any time).
- **Input**: None
- **Output:**
  - On success: HTTP response code 200 with a *Pool size message*
  - On error: HTTP response code 500 with an *Error response message*

1.2.3 Set desired size

- **Method**: `POST /pool/size`
- **Description**: Sets the desired number of machines in the machine pool. This method is asynchronous and returns immediately after updating the desired size. There may be a delay before the changes take effect and are reflected in the machine pool.

  Note: the cloud adapter should take measures to ensure that requested machines are recognized as pool members. The specific mechanism to mark group members, which may depend on the features offered by the particular cloud API, is left to the implementation but could, for example, make use of tags.
- **Input**: The desired number of machine instances in the pool as a *Set desired size message*.
- **Output:**
  - On success: HTTP response code 200 without message content.
  - **On error:**
    - on illegal input: code 400 with an *Error response message*
    - otherwise: HTTP response code 500 with an *Error response message*

1.2.4 Terminate machine

- **Method**: `POST /pool/<machineId>/terminate`
- **Description**: Terminates a particular machine pool member with id `<machineId>`. The caller can control if a replacement machine is to be provisioned via the `decrementDesiredSize` parameter.
- **Input**: A *Terminate machine message*.
- **Output:**
  - On success: HTTP response code 200 without message content.
  - **On error:**
    - on illegal input: code 400 with an *Error response message*
    - if the machine is not a pool member: code 404 with an *Error response message*
    - otherwise: HTTP response code 500 with an *Error response message*

1.2.5 Set service state

- **Method**: `POST /pool/<machineId>/serviceState`
- **Description**: Sets the service state of a machine pool member with id `<machineId>`. Setting the service state has no side-effects, unless the service state is set to `OUT_OF_SERVICE`, in which case a replacement machine will (eventually) be launched since out-of-service machines are not considered effective members of the pool. An out-of-service machine can later be taken back into service by another call to this method to re-set its service state.

  The specific mechanism to mark group members, which may depend on the features offered by the particular cloud API, is left to the implementation but could, for example, make use of tags.
- **Input**: A *Set service state message*.
- **Output:**
  - On success: HTTP response code 200 without message content.
  - **On error:**
    - on illegal input: code 400 with an *Error response message*
    - if the machine is not a pool member: code 404 with an *Error response message*
    - otherwise: HTTP response code 500 with an *Error response message*

1.2.6 Detach machine

- **Method**: `POST /pool/<machineId>/detach`
- **Description**: Removes a particular machine pool member with id `<machineId>` from the pool without terminating it. The machine keeps running but is no longer considered a pool member and, therefore, needs to be managed independently. The caller can control if a replacement machine is to be provisioned via the `decrementDesiredSize` parameter.
- **Input**: A *Detach machine message*.
- **Output:**
  - On success: HTTP response code 200 without message content.

  - **On error:**
    - on illegal input: code 400 with an *Error response message*
    - if the machine is not a pool member: code 404 with an *Error response message*
    - otherwise: HTTP response code 500 with an *Error response message*

1.2.7 Attach machine

- **Method**: `POST /pool/<machineId>/attach`
- **Description**: Attaches an already running machine with a given `<machineId>` to the machine pool, growing the pool with a new member. This operation implies that the desired size of the group is incremented by one.
- **Input**: None
- **Output:**
  - On success: HTTP response code 200 without message content.
  - **On error:**
    - on illegal input: code 400 with an *Error response message*
    - if the machine does not exist: code 404 with an *Error response message*
    - otherwise: HTTP response code 500 with an *Error response message*

1.3 Messages

1.3.1 Set desired size message

Description	A message used to request that the machine pool be resized to a desired number of machine instances.
Schema	`{ "desiredSize": <number> }`

Sample document:

```
{ "desiredSize": 3 }
```

States that we want three machine instances in the pool.

1.3.2 Error response message

Description	Contains further details (in addition to the HTTP response code) on server-side errors.
Schema	`{ "message": <string>, "detail": <string> }`

The `message` is a human-readable error message intended for presentation, whereas the `detail` attribute holds error details (such as a stack trace).

This is a sample error message:

```
{
  "message": "failure to process pool get request",
  "detail": "... long stacktrace ..."
}
```

1.3.3 Machine pool message

Description	Describes the current status of the monitored machine pool.

The machine pool schema has the following structure:

```
{
  "timestamp": <iso-8601 datetime>,
  "machines": [ <machine> ... ]
}
```

Here, every `<machine>` is also a json document with the following structure:

```
{
  "id": <string>,
  "machineState": <machine state>,
  "serviceState": <service state>,
  "launchtime": <iso-8601 datetime>,
  "publicIps": [<ip-address>, ...],
  "privateIps": [<ip-address>, ...],
  "metadata": <jsonobject>
}
```

The attributes are to be interpreted as follows:

- `id`: The identifier of the machine.
- `machineState`: The execution state of the machine. See the *machine state table* for the range of possible values.
- `serviceState`: The operational state of the service running on the machine. See the *service state table* for the range of possible values.
- `launchtime`: The launch time of the machine if it has been launched. If the machine is in a state where it hasn't been launched yet (`REQUESTED` state) this attribute may be left out or set to `null`.
- `publicIps`: The list of public IP addresses associated with this machine. Depending on the state of the machine, this list may be empty.
- `privateIps`: The list of private IP addresses associated with this machine. Depending on the state of the machine, this list may be empty.
- `metadata`: Additional cloud provider-specific meta data about the machine. This field is optional (may be `null`).

Below is a sample machine pool document:

```
{
  "timestamp": "2013-11-07T13:50:00.000Z",
  "machines": [
    {
      "id": "i-123456",
      "machineState": "RUNNING",
      "serviceState": "IN_SERVICE",
      "launchtime": "2013-11-07T14:50:00.000Z",
      "publicIps": ["54.211.230.169"],
      "privateIps": ["10.122.122.69"],
      "metadata": {
        "scaling-group": "mygroup"
      }
    },
    {
```

```
      "id": "i-123457",
      "machineState": "PENDING",
      "serviceState": "BOOTING",
      "launchtime": "2013-11-07T13:49:50.000Z",
      "publicIps": [],
      "privateIps": [],
      "metadata": {
        "scaling-group": "mygroup",
      }
    }
  ]
}
```

1.3.4 Pool size message

Description	Carries information about the pool size, both desired and actual size.
Schema	`{ "desiredSize": <number>, "allocated": <number>, "outOfService": <number> }`

The attributes are to be interpreted as follows:

- `desiredSize`: The last desired size set for the machine pool (see *Set desired size*).
- `allocated`: The number of allocated machines in the pool (in one of machine states `REQUESTED`, `PENDING`, `RUNNING`)
- `outOfService`: The number of machines in the pool that are marked with a `OUT_OF_SERVICE` service state.

Example:

```
{ "desiredSize": 3, "allocated": 4, "outOfService": 1 }
```

1.3.5 Terminate machine message

Description	Specifies if the desired size of the machine pool should be decremented after terminating the machine (that is, it controls if a replacement machine should be launched)
Schema	`{ "decrementDesiredSize": <boolean> }`

The attributes are to be interpreted as follows:

- `decrementDesiredSize`: `true` if the desired pool size should be decremented, `false` otherwise.

Example where a replacement machine is desired:

```
{ "decrementDesiredSize": false }
```

1.3.6 Detach machine message

Description	Specifies if the desired size of the machine pool should be decremented after detaching the machine (that is, it controls if a replacement machine should be launched)
Schema	`{ "decrementDesiredSize": <boolean> }`

The attributes are to be interpreted as follows:

- `decrementDesiredSize`: `true` if the desired pool size should be decremented, `false` otherwise.

Example where a replacement machine is desired:

```
{ "decrementDesiredSize": false }
```

1.3.7 Set service state message

Description	Specifies the service state to set for the machine.
Schema	`{ "serviceState": "<service state>" }`

The attributes are to be interpreted as follows:

- `serviceState`: The *service state* to set.

Example where a replacement machine is desired:

```
{ "serviceState": "IN_SERVICE" }
```