Rackspace Autoscale

Getting Started Guide

API v1.0 (Jul 2, 2013)

the open cloud company

docs.rackspace.com/api

Rackspace Autoscale Getting Started Guide

API v1.0 (2013-07-02)

Table of Contents

List of Examples

1. Overview

The Rackspace Autoscale service responds to events by changing capacity to meet current needs. This ability to expand the configuration in response to increased workload means that you can begin with a minimal cloud configuration and grow only when the cost of that growth is justified.

1.1. Document Change History

This version of the document replaces and obsoletes all previous versions. The most recent changes are described in the following table:

Revision Date	Summary of Changes
Jul 2, 2013	• Initial draft for preview release.

1.2. Additional Resources

You can see and experiment with all Rackspace Autoscale API calls at http://docs.autoscale.apiary.io/.

To use Rackspace Autoscale, you must be able to set cloud monitoring alarms and create cloud servers and cloud load balancers. You can read API documentation for those and other Rackspace services at http://docs.rackspace.com/.

2. Core Concepts

Autoscale is an API-based tool that enables you to scale your servers in response to variation in load. Autoscale functions by linking three services:

- a monitoring service, such as Rackspace Cloud Monitoring
- the Autoscale API
- load-balanced cloud servers, such as Rackspace Cloud Load Balancers working with Cloud Servers

2.1. Basic Workflow

A scaling group is monitored by Rackspace Cloud Monitoring. When Cloud Monitoring triggers an alarm for high utilization within the scaling group, a webhook is triggered. The webhook stimulates the Autoscale service, which consults a policy associated with the webhook. The policy determines how many cloud servers should be added (scaled up) or removed (scaled down) in response to the alarm.

Scale down events remove the oldest server in the group.

Note

For the Preview release, scaling down is not implemented. Autoscale can create servers, but you must manually remove those servers when you no longer want them to be active.

A cooldown is a period during which no other scaling activity occurs. Cooldowns allow some time for the initial scaling response to take effect, potentially resolving the situation that triggered the policy. When a scaling policy is triggered, cooldown periods begin for both the scaling policy and the group.

- While the group cooldown is active, any scaling requests to the group are discarded.
- While the policy cooldown is active, any scaling requests related to the policy are discarded.

After the cooldown period ends, if Cloud Monitoring still detects the situation that triggered the initial scaling activity, then another alarm triggers another round of scaling activity. By configuring cooldown periods between triggers, you can control how quickly Autoscale scales up toward the scaling group's maximum size or down toward the scaling group's minimum size.

Autoscale does not configure anything within a server. You must configure your services to function properly when each server is started. We recommend automating your servers' startup processes with Chef or a similar tool.

2.2. Sample Use Case

Five cloud servers are in a scaling group, with Rackspace Cloud Monitoring monitoring their CPU usage. Monitoring is configured to trigger an alarm when CPU utilization for any server within the group is at 90%. That alarm triggers a webhook that Autoscale created previously. When that webhook is activated, Autoscale receives the alert and carries out a policy specific to that webhook. This policy says "When this webhook is triggered, create five servers according to the launch configuration, and add them to the load balancer." Autoscale then initates creation of those servers within the scaling group.

Autoscaling can also work in the opposite direction. A policy can say "When this webhook is triggered, scale down by five servers."

Note

For the Preview release, scaling down is not implemented. Autoscale can create servers, but you must manually remove those servers when you no longer want them to be active.

3. Autoscale Components

Autoscale consists of the following components:

- scaling group configuration
- launch configuration
- scaling policies

3.1. Scaling Group Configuration

The scaling group configuration specifies the basic elements of scaling. It controls how many servers can participate in the scaling group.

The group configuration describes the following items:

- group name
- group cooldown
- minimum number of entities required in the scaling group
- maximum number of entities allowed in the scaling group

Within the group configuration, group cooldown describes, in seconds, how long the group must wait after a scaling policy is triggered before another request for scaling is accepted.

3.2. Launch Configuration

The launch configuration specifies how to create new servers. It controls what image a new server starts on, what flavor the new server is, and which load balancer the new server connects to.

The launch configuration describes the following items:

- launch configuration type
  - Launch Server *(For the Preview release, this is the only configuration type.)*
- server
  - name
  - flavorRef
  - imageRef *(This is the ID of the Cloud Server image to start.)*
- load balancer
  - loadBalancerId
  - port

Tip

To learn more about configuring cloud servers through an API, see the following information:

- Next Generation Cloud Servers Getting Started
- Next Generation Cloud Servers Developer Guide

To learn more about configuring cloud load balancers through an API, see the following information:

- Rackspace Cloud Load Balancers Getting Started
- Rackspace Cloud Load Balancers Developer Guide

The Rackspace Open Cloud Community is another place to learn about working with cloud resources through APIs. Within the community, the Developer Forum begins at https://community.rackspace.com/developers/default.

3.3. Scaling Policies

Scaling policies specify how to change the scaling group. A group can be managed by multiple scaling policies.

The scaling policy describes the following items:

- scaling policy name
- change value *(incremental or per cent)*
- policy cooldown *(in seconds)*
- execute webhook *(generated)*

Within the policy configuration, policy cooldown describes, in seconds, how long a group managed by this policy must wait after scaling before beginning to scale again.

4. Using the Autoscale API

You can create a oscaling group by following the steps described in this chapter.

Tip

If you want to try these steps in your own environment, you can use http://docs.autoscale.apiary.io/ to generate cURL commands.

4.1. Step 1: Authenticate

As with all Rackspace APIs, before you can use the Rackspace Autoscale API you must authenticate with your Rackspace credentials.

When you authenticate, you give your credentials to the Rackspace Cloud Identity Service. If your credentials are valid, the Identity Service responds with your Service Catalog, which includes your token, token ID, user ID, and token tenant ID. To make requests of Autoscale, you must use your token tenant id, also known as the tenantID.

If you subscribe to many Rackspace services, your Service Catalog response might be lengthy. For an example of an annotated Service Catalog response, showing how to recognize the token and other elements in both JSON and XML formats, see http://docs.rackspace.com/auth/api/v2.0/auth-client-devguide/content/Sample_Request_Response-d1e64.html.

After you have your token, you can use it to identify yourself to Rackspace Autoscale and other services.

Tip

For detailed information about the Rackspace Cloud Identity Service, visit http://docs.rackspace.com/auth/api/v2.0/auth-client-devguide/content/QuickStart-000.html.

4.2. Step 2: Create a Server and Save its Image

Create a cloud server. You can create a server through the control panel. You can also create a server through the Cloud Servers API by using a Create Server request.

With the Cloud Servers API, you can use a List Images request to retrieve a list of options available for configuring your server.

Example 4.1. Requesting a List of Cloud Server Images

```
curl -X GET -H "Content-Type: application/json" -H "X-Auth-token:{auth-token}"
 https://ord.servers.api.rackspacecloud.com/v2/{Tenant-id}/images?type=
SNAPSHOT | python -mjson.tool
```

Customize your cloud server so that it can process your requests. For example, if you are building a webhead scaling group, configure Apache to start on launch and serve the files that you need.

After you have created and customized your server, save its image and record the imageID.

4.3. Step 3: Create a Scaling Group

Create a scaling group by submitting a **POST** request customized with the tenant ID that you obtained during the Authentication step.

Example 4.2. Creating a Scaling Group

```
POST https://autoscale.api.rackspacecloud.com/v1.0/{tenantID}/groups/
```

```
{
    "groupConfiguration": {
        "name": "myFirstAutoscalingGroup",
        "cooldown": 60,
        "minEntities": 1,
        "maxEntities": 10
    },
    "launchConfiguration": {
        "type": "launch_server",
        "args": {
            "server": {
                "flavorRef": 3,
                "name": "webhead",
                "imageRef": "8de09731-2172-456f-9aab-13b8698fd530"
            },
            "loadBalancers": [
                {
                    "loadBalancerId": 141869,
                    "port": 80
                }
            ]
        }
    },
    "scalingPolicies": []
}
```

This action creates your scaling group, starts the minimum number of servers, and attaches those servers to the specified load balancer.

To change this scaling group, you must create policies.

4.4. Step 4: Create Scaling Policies

You can use multiple policies to manage your scaling group.

Create a policy by submitting a **POST** request customized with the tenant ID that you obtained during the Authentication step and the group ID that you obtained during the Create a Scaling Group step.

Example 4.3. Creating a Scaling Policy

```
POST https://autoscale.api.rackspacecloud.com/v1.0/{tenantID}/groups/
{groupID}/policies/
```

```
[
    {
        "name": "scale up by one server",
        "change": 1,
        "cooldown": 150,
        "type": "webhook"
    },
    {
        "name": "scale down by 5.5 percent",
        "changePercent": -5.5,
        "cooldown": 6,
        "type": "webhook"
    }
]
```

Note

For the Preview release, scaling down is not implemented. Autoscale can create servers, but you must manually remove those servers when you no longer want them to be active.

4.5. Step 5: Create Webhooks

You can use webhooks to tie policies to events. When a webhook is set, a configured event causes an HTTP **POST** callback request to be sent to a configured URI.

Example 4.4. Creating a Scaling Webhook: Request

```
POST https://autoscale.api.rackspacecloud.com/v1.0/{tenantID}/groups/
{groupID}/policies/{policyID}/webhooks
```

```
[
    {
        "name": "gus",
        "metadata": {
            "notes": "This is is note about a webhook. In the response, look
 above this line to the href to see the webhook's endpoint."
        }
    }
]
```

Example 4.5. Creating a Scaling Webhook: Response

```
{
    "webhooks": [
        {
            "id": "de94fb86-1cf5-41cc-913a-84a33a7d37f5",
            "links": [
                {
                    "href": "https://autoscale.api.rackspacecloud.
com/v1.0/384776/groups/1ce13f33-39c4-46ea-823e-c3726b294df9/
policies/33aeaf8e-8a0f-45ca-8275-b3727dea48cf/webhooks/
de94fb86-1cf5-41cc-913a-84a33a7d37f5/",
                    "rel": "self"
                },
                {
                    "href": "https://autoscale.api.rackspacecloud.com/v1.0/
execute/1/bd9ba9620b2564c49352e49ce12c81efb64522cdedead2830ce2aee649dca0b7/",
                    "rel": "capability"
                }
            ],
            "metadata": {
                "notes": "This is is note about a webhook. In the response,
 look above this line to the href to see the webhook's endpoint."
            },
            "name": "gus"
        }
    ]
}
```

4.6. Step 6: Execute a Scaling Policy

After you define a policy, you can activate that policy by issuing a **POST** request against its URL.

Example 4.6. Activating a Scaling Policy: cURL

```
curl -X POST https://autoscale.api.rackspacecloud.com/v1.0/execute/1/
{capability_hash}/ -v
```

```
[
    {
        "name": "scale up by one server",
        "change": 1,
        "cooldown": 150,
        "type": "webhook"
    },
    {
        "name": "scale down by 5.5 percent",
        "changePercent": -5.5,
        "cooldown": 6,
        "type": "webhook"
    }
]
```

The **POST** request causes the scaling policy to execute, which causes the scaling group to transform if it is able to do so.

Execution of a scaling policy always returns a `202 Accepted` status. This status indicates that the scaling policy was executed but does not indicate whether the execution successfully caused the scaling group to scale. If some element of the configuration is invalid, scaling can fail even if the scaling policy is executed.

4.7. Step 7: Deactivate and Delete a Scaling Group

You can stop a scaling group from scaling by configuring both its minimum and maximum sizes to `0`.

Example 4.7. Deactivating a Scaling Group

```
PUT /{tenantId}/groups/{groupID}/config
```

```
{
    "name": "workers",
    "cooldown": 60,
    "minEntities": 0,
    "maxEntities": 0,
    "metadata": {
        "firstkey": "this is a string",
        "secondkey": "1"
    }
}
```

Setting `maxEntities` to `0` prevents the scaling group from expanding.

Note

For the Preview release, scaling down is not implemented. Autoscale can create servers, but you must manually remove those servers when you no longer want them to be active.

You cannot delete a scaling group until all the servers within the group are removed.

When a group contains no servers, you can eliminate the group by sending a **DELETE** request to its group ID.

```
DELETE /[tenantId]/groups/[groupId[
```