DIY Rackspace Cloud Monitoring and Auto Scale Deployment with cURL for Apache Site

Introduction 1

Audience 1

Prerequisites 2

Process 2

Before you begin 2

Tips 2

Typographic conventions 3

Part 1: Authenticate, export tokens, generate SSH keys 3

Authenticate 3

Export tokens 4

Generate SSH keys 4

Part 2: Create a cloud server 4

Part 3: Configure the server and create the server image 5

Install Apache 2 5

Add the Apache boot up script 5

Install and set up the monitoring agent 6

Configure the Cloud Monitoring check and alarm 7

Part 4: Create the Cloud Server image 8

Part 5: Create a load balancer 10

Part 6: Configure Auto Scale 11

Part 7: Create webhook notifications and update the notification plan 13

Part 8: Install the testing tool and test 14

Install the website traffic tool 14

Test the deployment 14

Part 9: Tear down the deployment 15

## Introduction

Rackspace Auto Scale and Cloud Monitoring provide a secure way to dynamically manage your cloud resources in response to load. This use case walks you through the steps of creating an Auto Scale and Cloud Monitoring deployment with a Rackspace cloud server hosting an Apache website that is load-balanced by a Rackspace cloud load balancer. The exercise describes all aspects of the deployment, including the required setup and traffic load tool to use for testing, and the proper teardown of the deployment.

To configure this deployment you will use cURL to invoke the API calls directly. Follow the commands exactly to achieve the end result of Cloud Monitoring triggering Auto Scale policies.

## Audience

This document is intended for:

* Experienced Rackspace API users to see all of the steps involved in a complete deployment utilizing Cloud Monitoring and Auto Scale
* Less experienced Rackspace API users for testing purposes.

## Prerequisites

On your workstation install the following tools:

|  |  |
| --- | --- |
| **Tool** | **Download URL / Notes** |
| cURL | (<http://curl.haxx.se/>) |
| Apache HTTP server benchmarking tool | <http://httpd.apache.org/docs/2.2/programs/ab.html>  This tool allows you to test your deployment. It triggers the Monitoring alarm that triggers the Auto Scale scaling. |
| Pip | **Sudo easy\_install pip** |

## Process

1. Install prerequisites.
2. Authenticate, export IDs, and set up SSH keys. If you already have SSH keys, you can use them but only if they are in your Home directory; otherwise, it’s best to generate new ones and delete them later.
3. Create a Cloud Server.
4. Configure the server by installing the Apache Server 2, and the Apache agent boot-at-startup script. Also, create a notification plan (with placeholder notifications), install the Cloud Monitoring agent and configure a monitoring check, and alarm using the notification plan.
5. Create an image of the configured server. Auto Scale will use this server image to scale up or down.
6. Create a load balancer, modify it to have the Least-Connections algorithm, and attach the Cloud Server image you created as a node.
7. Configure the Auto Scale scaling group, as well as the scaling policies for scaling up and down; in the response you will get the webhooks that Cloud Monitoring will use in notifications (you create the notifications next and then attach them to your notification plan). Note: a “webhook” is a URL that triggers an action, such as sending an email. The webhooks the scaling group creates allow monitoring notifications that trigger the scaling policies.
8. Create notifications for scaling up and scaling down using the scaling group webhooks, and update the notification plan that you specified in the monitoring check.
9. Run a load test using the Apache Benchmarking tool and observe notifications and Auto Scale: Critical notification and scale-up; without load: OK notification and scale down
10. Tear down the deployment.

## Before you begin

These instructions use cURL commands. **Be sure to** set the necessary environment variables, as described before running any commands.

### Tips

* Use a text editor, such as Sublime, to form the commands. That is, copy the command from this document into the editor and make the required changes there. Then, paste the results into a shell window and execute the command (press Enter). Note: Sublime isn’t free, but there are several editors out there like [Notepad++](http://alternativeto.net/software/notepad-plus-plus/), [Vim](http://alternativeto.net/software/vim/), and [Brackets](http://brackets.io/), that are free. Trying to use the commands, even modified, from the Word file directly may not work.
* Use the accompanying worksheet to note values. **NOTE** indicates a worksheet item
* Watch out for quotes and dashes
* Commands must use straight quotes (not curly) and dashes must be single hyphens (not em or en dashes).
* Be careful to not allow a space between an equals sign (=) and the following value.

## Typographic conventions

The commands you use are shown in **Courier bold** font.

Variables in the command that you must replace with the appropriate text are shown in ***Courier bold italic*** font.

## Part 1: Authenticate, export tokens, generate SSH keys

Before you can run any calls, you need to get an authentication token. Also, you’ll want to generate a set of SSH keys to use.

### Authenticate

**Step 1**: Before you begin, you need an authentication token. To get a 24-hour authentication token, you use your Cloud account username and API key (you can find this through the Control Panel for your account); you’ll also need your 6-digit account number referred to as the **"*tenantID*"**

**NOTE** your username, API key, and account number in your worksheet

Open a shell window and run this command, replacing the italicized text:  
  
**curl -s https://identity.api.rackspacecloud.com/v2.0/tokens -X 'POST' \  
-d '{"auth":{"RAX-KSKEY:apiKeyCredentials": {"username":"*YOUR\_ACCOUNT\_USER\_NAME*", "apiKey":"*YOUR\_API\_KEY*"}}}' \  
-H "Content-Type: application/json" | python -m json.tool**  
  
Result: A print out of all service access API endpoints allowed for your account. Also a "tenant" section similar to this, the authToken is shown highlighted:

**{"access": {**

**"user": {...},**

**"serviceCatalog": [...],**

**"token": {**

**"RAX-AUTH:authenticatedBy": ["APIKEY"],**

**"expires": "2014-12-30T22:22:50.457Z",**

**"id": "e4aa72f87d174d7abefb290dabc90",**

**"tenant": {**

**"id": "90000",**

**"name": "90000"**

**}**

**}**

**NOTE** the token “id” in your worksheet; this is known as “the token” or the “authToken.”

### Export tokens

**Step 2:** Export your tenant ID and Auth Token, this allows you to use the commands in this procedure with the placeholders, **$account** (for tenantID) and **$token** (for authToken) left as placeholders. The system will fill them in with your information:

**export account="*tenantId*"**

**export token="*authToken*"**

### Generate SSH keys

**Step 3**: Generate a pair of SSH keys for use with the servers. The following example uses the ssh-keygen tool. Replace the italicized text with a name for the key file. This is a one-time operation. In your shell window, run this command:

**ssh-keygen -q -t rsa -f *ssh-keys* -N ""**

Result: Two key files are generated in your home directory, one with the “.pub” extension that is the public SSH key. Note that you can use SSH keys that you have previously generated, only you will have to modify the path to them in some of the following commands, if they are not in your home directory. You will also have to base 64 encode them at some point for use in the create scaling group command.

**NOTE** the SSH keys filename in your worksheet; you will use it in later operations.

You can check for the keys in your home directory with these commands:

**cd ~**

**ls**

**Step 4**: Base64 encode the public key file by typing the following, replacing italicized text:

|  |
| --- |
| **base64 *ssh-keys*.pub** |

**NOTE** the result as PUBKEY in your worksheet (it is a long number)

## Part 2: Create a cloud server

The first part in creating the Auto Scale deployment is to create a master cloud server from an existing server image. Your Rackspace Cloud Server account already provides a set of preconfigured server images that you can use to create your image from.

In this deployment, you will create an Ubuntu 14.10 server with the flavor “performance1-1”.

**Step 1**: Create a Cloud Server.

1. Create the **json** server file locally, this example uses nano:  
   **nano server\_build.json**
2. Add these details to the **server\_build.json** file to create the Ubuntu server:

**{**

**"server": {**

**"name": "myDIYServer",**

**"imageRef": "8f569a31-ee74-409b-9dcb-fb7576e307e9",**

**"flavorRef": "performance1-1",**

**"metadata": {**

**"My Server Name": "DIY Ubuntu 14.10 server"**

**}**

**}**

**}**

1. Create the server by issuing the following cURL command:

**curl -s "https://dfw.servers.api.rackspacecloud.com/v2/$account /servers -X POST" -H "Content-Type: application/json" -H "X-Auth-Token:$token" -T server\_build.json | python -m json.tool**

**NOTE** the server ID and adminPassword that you receive in the response.

**Step 2:** Get server details so you have the IP address:

**curl -s "https://dfw.servers.api.rackspacecloud.com/v2/$account/servers/*SERVER\_ID*"  
-H "X-Auth-Token:$token" | python -m json.tool**

**NOTE** the IP “public: v4” address of the newly created Cloud Server. The server status must be Active in order to proceed to **Step 3**, next.

**Step 3**: Log into the new Cloud Server by typing the following in a shell window, replacing *SSH-KEYS* with the name of your SSH key file and *SERVER\_IP\_ADDRESS* accordingly:

**ssh -i *SSH-KEYS* root@*SERVER\_IP\_ADDRESS***(without ssh keys, you can login with **ssh root@*SERVER\_IP\_ADDRESS***)

## Part 3: Configure the server and create the server image

While logged into your cloud server, follow these steps to install the Cloud Monitoring agent and the agent boot script. After the server is configured, you’ll log out and create the server image.

### Install Apache 2

While logged into the Cloud server, install apache2 by typing the following at the root prompt:

**apt-get update**

**apt-get --yes install apache2**

**apt-get install apache2-utils**

You can test this by opening a browser page to the IP address of the server.

### Add the Apache boot up script

While logged into the server:

1. Create the agent boot script, by typing the following at the command line (ignore warnings):

**echo "service apache2 start && service rackspace-monitoring-agent start" > /etc/init.d/apache\_on\_start**

Result: Prompt returned

**cd /etc/init.d && sudo chmod +x /etc/init.d/apache\_on\_start && sudo update-rc.d apache\_on\_start defaults**

Result: Warning you can ignore, and prompt returned

1. Exit the server:

**sudo reboot**Result: System reboots

1. Wait for a short bit and SSH back in again, replacing italicized text:

**ssh -i *ssh-keys* root@*SERVER\_IP\_ADDRESS***

1. Check if Apache2 is running once you log back in by typing:

**sudo netstat –tnlp**Result: Apache should be listening on port 80.

### Install and set up the monitoring agent

**Step 1**: While logged into your cloud server, install the monitoring agent.

1. For the Ubuntu 14.10 server you created, install the monitoring package with this command:

**sudo sh -c 'echo "deb http://stable.packages.cloudmonitoring.rackspace.com/ubuntu-14.10-x86\_64 cloudmonitoring main" > /etc/apt/sources.list.d/rackspace-monitoring-agent.list'**

Result: Prompt only returned

1. Download the signing key for the agent repository and add it by issuing the following cURL command:

**curl https://monitoring.api.rackspacecloud.com/pki/agent/linux.asc | sudo apt-key add -**

Result: Stats returned

1. Run an APT update to get package information for the new repository:

**sudo apt-get update**

Result: Packages installed

1. Install the agent:

**sudo apt-get install rackspace-monitoring-agent**

Result: Agent unpacked and installed

**Step 2:** Setup the Agent. The agent setup program performs the following tasks:

* Creates an agent token, which performs authentication with Cloud Monitoring.
* Creates an agent configuration file.
* Runs a test connection to verify connectivity with the Rackspace data centers.
* Associates the agent ID with the server's entity ID in Cloud Monitoring.

1. Run the agent setup command for Ubuntu 14.10, which will start the agent setup interface:

**sudo rackspace-monitoring-agent --setup**

Enter your Rackspace Cloud user name when prompted.  
Enter your API key when prompted.

1. Select the "Create New Token" option by pressing the associated number.   
   Result: The message **Agent successfully connected!  
   NOTE** the monitoring Entity ID, near the bottom of the output. That is the monitoring entity that represents your cloud server and with which the agent is associated. The entity name starts with the letters *en*. You use this entity ID in the next section to configure monitoring.
2. You can now start the agent:  
   **service rackspace-monitoring-agent start**

Result: Agent unpacked and installed

For more details, see [Install the Cloud Monitoring Agent](http://www.rackspace.com/knowledge_center/article/install-the-cloud-monitoring-agent).

### Configure the Cloud Monitoring check and alarm

**Step 1**: Create a notification plan (with placeholder notifications, you’ll update it later).

1. On the cloud server, make sure pip is installed (respond “y” to prompts as needed):  
   **sudo apt-get install python-pip**
2. On your local machine, create a notification plan with placeholder notifications. Later, after creating the scaling group and webhooks, you will create notifications for that plan. This command follows the convention of naming the notification plan after the server name with the extension “\_NP.” Use “$Notification” as-is (a placeholder)  
     
   **curl -i -X POST --data-binary '{"label": "myDIYServer\_NP", "critical\_state": ["$Notification"], "ok\_state": ["$Notification "] }' -H "X-Auth-Token:$token" -H 'Content-Type: application/json' -H 'Accept: application/json' "https://monitoring.api.rackspacecloud.com/v1.0/$account/notification\_plans"**

**NOTE** the notification plan ID that is returned.

**Step 2**: Create a monitoring check and alarm.

1. Create a monitoring check by typing the following and correctly replacing the value for ENTITY\_ID. This check sets the timeout at 60 and the period at 150.

**curl -i --data-binary '{"details":{ }, "label":"apache\_check", "period":"60", "timeout":30, "type":"agent.apache"}' -H "X-Auth-Token:$token" -H "Content-Type: application/json" -H "Accept: application/json" "https://monitoring.api.rackspacecloud.com/v1.0/$account/entities/*ENTITY\_ID*/checks"**

**NOTE** the CHECK\_ID that is returned; it begins with “ch”

1. Create an alarm for the check, replacing italicized text:  
     
   **curl -i -X POST --data-binary '{"check\_id": "*CHECK\_ID*", "notification\_plan\_id": "*NOTIFICATION-PLAN\_ID*", "criteria: ":set consecutiveCount=1 if (metric[\"busy\_workers\"] > 1) {return new AlarmStatus(CRITICAL);} return new AlarmStatus(OK)" -H "X-Auth-Token:$token" -H 'Content-Type: application/json' -H 'Accept: application/json' "https://monitoring.api.rackspacecloud.com/v1.0/$account/entities/*ENTITY\_ID*/alarms"**

**Step 3**: Test that the check works by typing the following and replacing ENTITY\_ID, and CHECK\_ID with the appropriate values:

**curl -i --data-binary '{"details" : { }, "label" : "*CHECK\_ID*", "monitoring\_zones\_poll": [ "mzdfw" ], "period" : "60", "target\_alias": "default", "timeout": 30, "type": "agent.apache"}' -H "X-Auth-Token: $token" -H 'Content-Type: application/json' -H 'Accept: application/json' "https://monitoring.api.rackspacecloud.com/v1.0/$account/entities/*ENTITY\_ID*/test-check/"**

## Part 4: Create the Cloud Server image

Now that you have configured your cloud server, create an image for it using a shell window on your local workstation. The image will be used by Auto Scale to add new servers as needed. This command follows the convention of naming the server image with the extension “snap.”

1. Create the server image, replacing italicized text:

**curl -s -d '{"createImage": {"name": "myDIYServer\_IMAGE"} }' -X POST -H "X-Auth-Token:$token" -H "Content-Type: application/json" "https://dfw.servers.api.rackspacecloud.com/v2/$account/servers/*SERVER\_ID*/action"**Result: Prompt only returned.

1. Verify the image creation and get the server image ID. This command prints out to your home directory (example follows):

**curl -s "https://dfw.servers.api.rackspacecloud.com/v2/$account/images/detail" -H "X-Auth-Token:$token" | python -m json.tool > output.txt**

**NOTE** the server image ID

The contents of output.txt will include several entries similar to the following example; the image ID is highlighted (your image ID, and other data, will be different).

{

"image": {

"status": "ACTIVE",

"updated": "2015-01-29T00:13:09Z",

"links": [

{

"href": "https://dfw.servers.api.rackspacecloud.com/v2/820712/images/126a6674-6308-421f-801e-fc302ab4f53f",

"rel": "self"

},

{

"href": "https://dfw.servers.api.rackspacecloud.com/820712/images/126a6674-6308-421f-801e-fc302ab4f53f",

"rel": "bookmark"

},

],

"OS-DCF:diskConfig": "AUTO",

"id": "126a6674-6308-421f-801e-fc302ab4f53f",

"OS-EXT-IMG-SIZE:size": 758777106,

"name": "myDIYServer\_IMAGE"

"created": "2015-01-28T19:31:37Z",

"minDisk": 20,

"minRam": 512,

"visibility":"public",

"tags":[

"ping",

"pong"

],

"metadata": {

"flavor\_classes": "\*,!onmetal",

"auto\_disk\_config": "disabled"

},

"schema":"/v2/schemas/image"

}

}

## Part 5: Create a load balancer

The load balancer will allow Auto Scale to add (and remove) servers without affecting your server’s IP address.

1. Create a load balancer and add the Cloud Server you created to the load balancer by typing the following at a shell window, replacing the italicized text. This command follows the convention of naming the load balancer after the server name with the extension “LB.”

**curl -s -d '{"loadBalancer": {"name": " myDIYServer\_LB", "port": 80, "protocol": "HTTP", "virtualIps": [ {"type": "PUBLIC"} ],"nodes": [ {"address": "*SERVER\_IP\_ADDRESS*", "port": 80, "condition": "ENABLED" } ] }}' -H "X-Auth-Token:$token" -H "Content-Type: application/json" "https://dfw.loadbalancers.api.rackspacecloud.com/v1.0/$account/loadbalancers" | python -m json.tool**

Result: The load balancer ID (*LOAD\_BALANCER\_ID*) and VIP\_IP (*LOAD\_BALANCER\_IP\_ADDRESS*) are returned

**NOTE** the load balancer ID and load balancer VIP\_IP that you receive in the response.

1. Wait until the load balancer is active. You can check the status of the load balancer by typing the following, replacing *italic text*:

**curl -s -H "X-Auth-Token:$token" -H "Accept: application/json" "https://dfw.loadbalancers.api.rackspacecloud.com/v1.0/$account/loadbalancers/*LOAD\_BALANCER\_ID*" | python -m json.tool**

**Verify that the load balancer is active by looking at the status: section in the response.**

1. Change the algorithm to “Least Connections” by typing the following, replacing *LOAD\_BALANCER\_ID*:

**curl -s -d '{"loadBalancer":{"algorithm": "LEAST\_CONNECTIONS"}}' -H "X-Auth-Token:$token " -H "Content-Type: application/json" -X PUT "https://dfw.loadbalancers.api.rackspacecloud.com/v1.0/$account/loadbalancers/*LOAD\_BALANCER\_ID* "**Result: Only the prompt is returned. You can verify the change by repeating the status command in Step 2.

1. Test the connectivity by typing the following, replacing *SERVER\_IP\_ADDRESS*:

**curl http://*SERVER\_IP\_ADDRESS*/**

Result: This call should return the initial Apache web page, at the command line. You can also use a Web browser and type the *SERVER\_IP\_ADDRESS* in the address bar.

## Part 6: Configure Auto Scale

This section describes the Auto Scale configurations needed for a deployment with an Apache website. You create a scaling group with a launch configuration that specifies the server image to use when scaling up new servers. You also create scaling policies.

**Step 1**: Create an Auto Scale group through [create scaling group API](http://docs.rackspace.com/cas/api/v1.0/autoscale-devguide/content/POST_createGroup_v1.0__tenantId__groups_autoscale-groups.html#POST_createGroup_v1.0__tenantId__groups_autoscale-groups-Request) with the following 2 policies:

* scale up by 2
* scale down by 2

Run the following at a shell window (*replace endpoint region if needed*), replacing *LOAD\_BALANCER\_ID*, *SERVER\_IMAGE\_ID*, and *PUBKEY* with appropriate values. This command follows the convention of naming the Scaling Group after the server name with the extension “\_SG.”

**curl -i "https://dfw.autoscale.api.rackspacecloud.com/v1.0/$account/groups" -X POST -H 'Content-Type: application/json' -H "Accept: application/json" -H "X-Auth-Token:$token" --data-binary '{"launchConfiguration": {"args": {"loadBalancers": [{"port": 80, "loadBalancerId": *LOAD\_BALANACER\_ID*}], "server": {"name": "myDIYServer", "imageRef": "*SERVER\_IMAGE\_ID*", "flavorRef": "performance1-1", "OS-DCF:diskConfig": "AUTO", "networks": [{"uuid": "11111111-1111-1111-1111-111111111111"},{"uuid":"00000000-0000-0000-0000-000000000000"}], "personality": [{ "path": "/root/.ssh/authorized\_keys", "contents": "*PUBKEY*" } ]}}, "type": "launch\_server"}, "groupConfiguration": {"maxEntities": 5, "cooldown": 300, "name": " myDIYServer\_SG", "minEntities": 1}, "scalingPolicies": [ {"cooldown": 600, "type": "webhook", "name": "scaleUpBy2", "change": 2}, { "cooldown": 600, "type": "webhook", "name": "scaleDownBy2", "change": -2 } ]}'**

A successful response will look something like this; important information is highlighted and explained:

{"group": {"links": [{"href": "https://dfw.autoscale.api.rackspacecloud.com/v1.0/905240/groups/b7c6a3b1-c7c5-4c0c-b6d2-b3b7d4e123fc/", "rel": "self"}], "scalingPolicies": [{"name": "scale up by 2", "links": [{"href": "https://dfw.autoscale.api.rackspacecloud.com/v1.0/905240/groups/b7c6a3b1-c7c5-4c0c-b6d2-b3b7d4e123fc/policies/4cb356cb-f90c-4a4d-907f-c5ae64dc7075/", "rel": "self"}], "cooldown": 600, "type": "webhook", "id": "4cb356cb-f90c-4a4d-907f-c5ae64dc7075", "change": 2}, {"name": "scale down by 2", "links": [{"href": "https://dfw.autoscale.api.rackspacecloud.com/v1.0/905240/groups/b7c6a3b1-c7c5-4c0c-b6d2-b3b7d4e123fc/policies/123da0fa-d6ea-4625-807a-4563eab549d5/", "rel": "self"}], "cooldown": 600, "type": "webhook", "id": "123da0fa-d6ea-4625-807a-4563eab549d5", "change": -2}], "scalingPolicies\_links": [{"href": "https://dfw.autoscale.api.rackspacecloud.com/v1.0/905240/groups/b7c6a3b1-c7c5-4c0c-b6d2-b3b7d4e123fc/policies/", "rel": "policies"}], "state": {"desiredCapacity": 0, "paused": false, "active": [], "pendingCapacity": 0, "activeCapacity": 0, "name": "DIY-CS-3-3-15-SG-2"}, "launchConfiguration": {"args": {"loadBalancers": [{"port": 80, "loadBalancerId": 440551}], "server": {"name": "DIY-CS-3-3-15", "imageRef": "2eb7be80-37c0-46bf-ae68-2019484ddfe2", "flavorRef": "performance1-1", "OS-DCF:diskConfig": "AUTO", "networks": [{"uuid": "11111111-1111-1111-1111-111111111111"}, {"uuid": "00000000-0000-0000-0000-000000000000"}], "personality": [{"path": "/root/.ssh/authorized\_keys", "contents": ""}]}}, "type": "launch\_server"}, "groupConfiguration": {"maxEntities": 5, "cooldown": 300, "name": "DIY-CS-3-3-15-SG-2", "minEntities": 1, "metadata": {}}, "id": "b7c6a3b1-c7c5-4c0c-b6d2-b3b7d4e123fc"}}

**NOTE** the Scale-Up policy ID and link, and the Scale-Down policy ID and link, and the group ID

**Step 2**: Add a webhook for each of the policies by typing the following and passing the appropriate values for italic text. BE CAREFUL of inserting an extra slash (/) before “webhooks.”

1. Run the following command to add a webhook for the scale-up policy, example follows:

**curl -i *SCALE\_UP\_POLICY\_LINK*/webhooks -X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "X-Auth-Token:$token" --data-binary '[{"name": "scaleUpBy2", "metadata": {"owner": "*YOUR\_NAME*"}}]'**

**NOTE** the ID, and the “execute” URL that is returned in the response as the SCALE\_UP\_WEBHOOK

1. Run the following to add a webhook for the scale-down policy, replace italic text:

**curl -i *SCALE\_DOWN\_POLICY\_LINK*/webhooks -X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "X-Auth-Token:$token" --data-binary '[{"name": "scaleDownBy2", "metadata": {"owner": "*YOUR\_NAME*"}}]'**

**NOTE** the ID, and the “execute” URL that is returned in the response as the SCALE\_DOWN\_WEBHOOK

Here is an example response; the “execute” webhook ID, URL and name are highlighted:

HTTP/1.1 201 Created

Content-Type: application/json

Via: 1.1 Repose (Repose/2.12)

x-response-id: 68059065-80ab-41e7-8a74-db91070408cb

Location: https://dfw.autoscale.api.rackspacecloud.com/v1.0/905240/groups/aa6a20cc-4267-403c-a6d6-c076868533fa/policies/b6fb711e-af22-4cf7-988f-78587a5921b3/webhooks/

Date: Tue, 30 Dec 2014 02:01:12 GMT

Transfer-Encoding: chunked

Server: Jetty(8.0.y.z-SNAPSHOT)

{"webhooks": [{"metadata": {"owner": "mabrahms"}, "id": "2f36bd3b-8167-4850-bf78-80eddbaa3ba2", "links": [{"href": "https://dfw.autoscale.api.rackspacecloud.com/v1.0/905240/groups/aa6a20cc-4267-403c-a6d6-c076868533fa/policies/b6fb711e-af22-4cf7-988f-78587a5921b3/webhooks/2f36bd3b-8167-4850-bf78-80eddbaa3ba2/", "rel": "self"}, {"href": "https://dfw.autoscale.api.rackspacecloud.com/v1.0/execute/1/eae662ef28fa1253d839231425eabc7a4d1b9106b84906d028cd05a32d227c8e/", "rel": "capability"}], "name": "scaleUpBy2"}]}

## Part 7: Create webhook notifications and update the notification plan

Now that you have the webhooks created in Auto Scale, you can use them to create webhook notifications in Cloud Monitoring. Then you update your existing notification plan with the webhook notifications.

**Step 1**: Create webhook and email-based notifications using the values that you recorded in the previous step.

Use the appropriate values for SCALE\_UP\_WEBHOOK\_URL, and SCALE\_DOWN\_WEBHOOK\_URL.

1. Create a webhook-based notification for the SCALE-UP webhook by typing the following:

**curl -i -X POST --data-binary '{ "details": {"address": "*SCALE-UP\_WEBHOOK\_URL*" }, "label": "DIY-scaleup-notification", "type" : "webhook"}' -H "X-Auth-Token:$token" -H 'Content-Type: application/json' "https://monitoring.api.rackspacecloud.com/v1.0/$account/notifications"**

**NOTE** the Notification Resource ID for **DIY-scaleup-notification**

1. Create a webhook-based notification for the SCALE-DOWN webhook by typing the following:

**curl -i -X POST --data-binary '{"details" : {"address": "*SCALE-DOWN\_WEBHOOK\_URL*"}, "label": "DIY-scaledown-notification", "type": "webhook"}' -H "X-Auth-Token:$token" -H 'Content-Type: application/json' https://monitoring.api.rackspacecloud.com/v1.0/$account/notifications**

**NOTE** the Notification Resource ID for **DIY-scaledown-notification**

1. Create an email notification by typing the following and using your email address:

**curl -i -X POST --data-binary '{"details": { "address" : "*YOUR\_EMAIL\_ADDRESS*"}, "label": "DIY-email-notification", "type": "email"}' -H "X-Auth-Token:$token" -H 'Content-Type: application/json' https://monitoring.api.rackspacecloud.com/v1.0/$account/notifications**

**NOTE** the Notification Resource ID for **DIY-email-notification**

**Step 2**: Update the notification plan you used in the monitoring check by typing the following, replacing SCALE\_UP\_NOTIF\_ID, SCALE\_DOWN\_NOTIF\_ID, EMAIL\_NOTIF\_ID with the appropriate values

**curl -i -X PUT --data-binary '{"label": "myDIYServer\_NP", "critical\_state": [ "*SCALE\_UP\_NOTIF\_ID*,*EMAIL\_NOTIF\_ID*" ], "ok\_state": [ "*SCALE\_DOWN\_NOTIF\_ID*,*EMAIL\_NOTIF\_ID*" ] }' -H "X-Auth-Token:$token" -H 'Content-Type: application/json' -H 'Accept: application/json' "https://monitoring.api.rackspacecloud.com/v1.0/$account/notification\_plans"**

## Part 8: Install the testing tool and test

In this section you install the load-generating testing tool and see Auto Scale respond to Cloud Monitoring alerts. You should also receive Cloud Monitoring notifications to the configured email address.

### Install the website traffic tool

1. **Log into your cloud server:  
   ssh -i *ssh-keys* root@*SERVER\_IP\_ADDRESS***
2. **Modify the Apache configuration file, this example uses nano:  
   nano /etc/apache2/apache2.conf**

**Add the server name to the bottom of the file, like this:  
ServerName MyDIYServer**

1. **Stop and then sart the tool:  
   sudo /usr/sbin/apachectl stop  
   sudo /usr/sbin/apachectl start**
2. **Verify that apache is installed.  
   cd /usr/sbin**  
   **ls -l apache\***

### Test the deployment

**Step 1**: This example uses the [apache benchmarking tool](http://httpd.apache.org/docs/2.2/programs/ab.html) to load the load balancer, but you can use your favorite URL load test framework, for example, **ab** or **jmeter** to pump load.

Run the Apache Benchmarking command given, as often as needed, where **LB\_VIP\_IP** is the VIP IPv4 of the load balancer created earlier. Issue this command as needed (not from your server). Sometimes the first couple of times fail, just keep trying the command until the **Be Patient** message starts ticking off load numbers.

Once you get the CRITICAL notifications, cease loading the site. Wait 10 minutes or so. You should next receive an OK notification and Auto Scale should shortly afterwards delete the two servers that it added in the scale-up operation.

**ab -n 1000000 -c 30 http://*LB\_VIP\_IP*/**

Notice how the servers are added using this command:

**curl -s "https://dfw.servers.api.rackspacecloud.com/v2/$account/servers/detail" -H "X-Auth-Token:$token" | python -m json.tool**

**Step 2**: After the load subsides, the new servers may not be deleted automatically based on the timing as to when the alarm fires notification. This is due to alarm suppression in MaaS. It is possible that while the scaling group policy is in the cool down phase, that MaaS might send a scale down notification that will be ignored. MaaS is going to suppress additional notifications due to already being in OK state. You can manually scale down by executing the command directly.

**curl -i -X POST *SCALE-DOWN\_WEBHOOK\_URL***

## Part 9: Tear down the deployment

**Step 1**: Delete the scaling group:

1. Find the group ID, highlighted in the following example (or check worksheet):  
   **curl -i "https://dfw.autoscale.api.rackspacecloud.com/v1.0/$account/groups" -H "X-Auth-Token:$token"**  
   The response looks something like this, the group ID is highlighted:  
     
   {"groups\_links": [], "groups": [{"state": {"desiredCapacity": 1, "paused": false, "active": [{"id": "a0f619c4-645c-4f97-95b9-3d7e3382e71d", "links": [{"href": "https://dfw.servers.api.rackspacecloud.com/v2/905240/servers/a0f619c4-645c-4f97-95b9-3d7e3382e71d", "rel": "self"}, {"href": "https://dfw.servers.api.rackspacecloud.com/905240/servers/a0f619c4-645c-4f97-95b9-3d7e3382e71d", "rel": "bookmark"}]}], "pendingCapacity": 0, "activeCapacity": 1, "name": "DIY-CS-1-6-15-sg"}, "id": "1d1468e1-229f-49c1-96c2-b97cd216b2b8", "links": [{"hrefSFOSFOSFOSFOSFOs-MacBookSFOSFOSFOSFOSFOSFOSFOSFOs-MacBook-Pro-7SFOs-M
2. Optional. Find group details (or check worksheet for the group name and ID):  
   **curl -i "https://dfw.autoscale.api.rackspacecloud.com/v1.0/$account/groups/*GROUP\_ID*" -H "X-Auth-Token:$token"**
3. Set min and max entities to zero:  
   **curl -i -d '{"name": "*GROUP\_NAME*","cooldown": 600,"minEntities": 0,"maxEntities": 0,"metadata": {}}' -H "X-Auth-Token:$token" -H "Content-Type: application/json" -X PUT "https://dfw.autoscale.api.rackspacecloud.com/v1.0/$account/groups/*GROUP\_ID*/config"**
4. Delete the group:  
   **curl -i "https://dfw.autoscale.api.rackspacecloud.com/v1.0/$account/groups/*GROUP\_ID*" -H "X-Auth-Token:$token" -X DELETE**

**Step 2:** Delete the cloud server:

Delete server (get server ID from worksheet):  
**curl -i "https://dfw.servers.api.rackspacecloud.com/v2/$account/servers/*SERVER\_ID*" -H "X-Auth-Token:$token" -X DELETE**

**Step 3:** Delete the cloud server image:

Delete the image (get image ID from worksheet):  
**curl -i "https://dfw.servers.api.rackspacecloud.com/v2/$account/images/*IMAGE\_ID*" -H "X-Auth-Token:$token" -X DELETE**

**Step 4:** Delete all entities:

Delete entities (get ID from worksheet; some entities cannot be deleted):  
 **curl -i "https://monitoring.api.rackspacecloud.com/v1.0/$account/entities/*ENTITY\_ID*" -H "X-Auth-Token:$token" -X DELETE**

**Step 5:** Delete the load balancer:

Delete load balancers (get load balancer ID from worksheet):  
**curl -i "https://dfw.loadbalancers.api.rackspacecloud.com/v1.0/$account/loadbalancers/*LOAD\_BALANCER\_ID*" -H "X-Auth-Token:$token" -X DELETE**