Browsing Clouds, Deployments and Servers

John Fitzpatrick

Table of Contents

Lab Overview and Requirements

Overview

This lab introduces creating and managing Servers and Deployments with the RightScale API. The tasks you will perform include

- Authenticating
- Creating Deployments and Servers
- Launching and Terminating Servers
- Listing Deployments and Servers
- Deleting Servers and Deployments

These tasks will be performed by issuing API Command from Bash scripts. This Server you'll create is similar to the Base Server created in the IT Vending Machine demonstration.

Lab Environment

You will launch an 'API Sender' server in AWS EC2 cloud, and SSH into to from your desktop. All API commands will then be issued from this server.

Requirements

- A currently supported Browser. See http://tinyurl.com/cl8p4mh
- Java must be enabled in the browser.
- Unrestricted Internet Access. In particular, you must be able to browse to rightscale.com on TCP port 80 (HTTP).
- You must have TCP port 22 (SSH) open so you can SSH into remote servers
- RightScale account with valid cloud credentials, and the following user role privileges - **'designer'**, **'actor'**, **'library'**, '**server_login**', '**security_manager**'
- You must also be familiar with `vi` editor
- This lab assumes you already have an **myname API Sender** server running, based on the ServerTemplate "**API Sender [RSED]**"

Section 1. Creating Deployments and Servers

See:

http://reference.rightscale.com/api1.5/resources/ResourceDeployments.html

http://reference.rightscale.com/api1.5/resources/ResourceServers.html

1.1 Overview

In this lab exercise you will complete a few tasks to introduce and familiarize you with the '`Deployments`' and '`Servers`' resources.

This lab walks you through launching a Linux server using the RightScale API. The tasks include:-

- Authenticating
- Create Sec Group
- Create SSH Key
- Import ServerTemplate
- Create Deployment
- Create Server
- Launch Server

1.2 Create a Session

See: http://reference.rightscale.com/api1.5/resources/ResourceSessions.html

create

Creates API session scoped to a given account. (API login)

This call requires a form of authentication (user and password), as well as the account for which the session needs to be created. Upon successfully authenticating the credentials, the system will return a 204 code and set of two cookies that will serve as the credentials for the session. Both of these cookies must be passed in any of the subsequent requests for this session. If an 302 redirect code is returned, the client is responsible of re-issuing the POST request against the content of the received Location header, passing the exact same parameters again.

Example Request using Curl:
curl -i -H X_API_VERSION:1.5 -c mycookies -X POST -d email='email@me.com' -d password='mypassword' -d account_href=/api/accounts/11 https://my.rightscal

URLs:	POST /api/session
	POST /api/session
HTTP response code:	204 No Content

Parameters

name	required	type	values	regexp	blank?	description
account_href	yes	String	*	^\/api\/accounts \/\d+$	no	The account href for which the session needs to be created.
email	yes	String	*	*	no	The email to login with.
password	yes	String	*	*	no	The corresponding password.

Before you can issue any API Requests you will need to create a session, but this time you'll do it using a bash script.

1. Create a bash script (in the directory `/opt/api/myscripts`) that you can run each time you need to authenticate and create a session.

```
[api]# cd /opt/api/myscripts
[api]# vi auth-bash.sh
```

2. Enter the following text, placing `EMAIL` with your Email address and `ACCOUNT` with the RightScale Account number used during the training.

```
#!/bin/bash -e

EMAIL="myemail@example.com"
ACCOUNT="33172"

unset password

prompt="Please enter the password associated with email '$EMAIL': "
while IFS= read -p "$prompt" -r -s -n 1 char
do
    if [[ $char == $'\0' ]]
    then
        break
    fi
    prompt='*'
    password+="$char"
done

echo ""
curl -i -H X-API-Version:1.5 -c ~/mycookie -X POST -d email=$EMAIL -d
password=$password -d account_href=/api/accounts/$ACCOUNT https://us-
3.rightscale.com/api/session
```

Note: This script will prompt you for your account password, and authenticate you to the account. The first part of the script prompts you for your password, and masks your inputs. The session cookie is stored in your home directory.

3. Make the script executable by changing the permissions.

```
[api]# chmod +x auth-bash.sh
```

4. Now run the script.

```
[api]# ./auth-bash.sh
Please enter the password associated with email 'myemail@example.com':
**********
HTTP/1.1 204 No Content
Server: nginx/1.0.14
Date: Wed, 06 Mar 2013 12:33:52 GMT
Connection: keep-alive
Status: 204 No Content
X-Runtime: 948
X-Request-Uuid: b86f746f2e744ec4acc522f8f412ede0
Set-Cookie:
rs_gbl=eNotkMtugkAARf9l1kwyDxgGki4UClJRqVAI3ZBxmBZbwSqoVMO_F5Iub3LOWdwHEMA
GlxZSoIGyBfZjHOoMbKLrOh000ElgY8qIYVJCsQb25chjQQTHREDODAExVgRaQpkQjwi1CDdKV
Y69Tv27HOmTO-ZBvlyjVGVo83Gr4nr-ftI5gyg6tGwhsXFoFpWz3CSNs8WdOSMlkl7Mvb5ros-
2KHrH5cEq47VT9aEfV21VtUV43OTc3criHJMoCKizC08_ae5f1HqP6P2oQjOfSR-
eYbLjZeguv2j0WiuzI0Xzq67zG3Kv92e22krPvOYvcebs08Rnb99P0yX9dImQ8nhpOmAzYnE-
DH99P1ul; domain=.rightscale.com; path=/; HttpOnly
Set-Cookie: _session_id=deadbeef0badf00dfeedfacec0ffee24; path=/; HttpOnly
Cache-Control: no-cache
```

You should have received a `Status: 204 No Content` in the reponse, indicating you now have a valid session.

1.2.1 Create an Alias

5. Create an alias to make it easier to invoke API commands from the command line.

   An alias is a shortcut so you don't have to type a long command.

```
[api]# alias mycurl='curl -H X-API-Version:1.5 -b ~/mycookie '
```

1.3 Determine Cloud ID

See: http://reference.rightscale.com/api1.5/resources/ResourceClouds.html

index

Lists the clouds available to this account.

URLs:	GET /api/clouds
HTTP response code:	200 OK
Content-type:	application/vnd.rightscale.cloud;type=collection

Required roles

- observer

Parameters

name	required	type	values	regexp	blank?	description
filter	no	Array	*	*	no	See below for valid filter parameters.

Filters

name	partial_match?	description
cloud_type	no	The type of the cloud to filter on.
description	yes	The cloud description field to filter on.
name	yes	The name of the cloud to filter on.

Once you have authenticated in an account, you can use the `clouds` resource to view details of cloud available in the account.

To view cloud resources you issue an `HTTP GET` to the href `/api/clouds`. You can optionally filter the output on `name`, `description` or `cloud_type`. You must have at least `observer` permissions to do this.

6. Invoke the following Command

```
[api]# curl -i -H X-API-Version:1.5 -b ~/mycookie -X GET https://us-
3.rightscale.com/api/clouds.xml
<?xml version="1.0" encoding="UTF-8"?>
<clouds>
  <cloud>
    <description>Amazon's US Cloud on the East Coast</description>
    <links>
      <link href="/api/clouds/1" rel="self"/>
      <link href="/api/clouds/1/datacenters" rel="datacenters"/>
      <link href="/api/clouds/1/instance_types" rel="instance_types"/>
      <link href="/api/clouds/1/security_groups" rel="security_groups"/>
      <link href="/api/clouds/1/instances" rel="instances"/>
      <link href="/api/clouds/1/ssh_keys" rel="ssh_keys"/>
      <link href="/api/clouds/1/images" rel="images"/>
      <link href="/api/clouds/1/ip_addresses" rel="ip_addresses"/>
      <link href="/api/clouds/1/ip_address_bindings"
            rel="ip_address_bindings"/>
      <link href="/api/clouds/1/volume_attachments"
            rel="volume_attachments"/>
      <link href="/api/clouds/1/recurring_volume_attachments"
            rel="recurring_volume_attachments"/>
      <link href="/api/clouds/1/volume_snapshots" rel="volume_snapshots"/>
      <link href="/api/clouds/1/volume_types" rel="volume_types"/>
      <link href="/api/clouds/1/volumes" rel="volumes"/>
    </links>
    <cloud_type>amazon</cloud_type>
    <name>EC2 us-east-1</name>
  </cloud>
</clouds>
```

You will need to select a cloud you wish to launch servers in, and make a note of that cloud's Cloud ID. The Cloud ID can be identified from the href URL, e.g. `1` in the above example for `EC2 us-east-1`. You will then use this number in later sections of this lab.

The output also includes a number of link hrefs that can be used to form URLs to access further information of the specific cloud.

7. Make a note of the Cloud ID for the cloud you're working in:-

   Cloud ID: _____

1.3.1 Drill Down Into Cloud Details

8. Now drill down into the cloud to show further details using a number of the hrefs given in the above example.

   For example, view '`datacenters`' or '`security_groups`', by appending the appropriate href to the URL used above, i.e.

```
[api]# mycurl -X GET https://us-3.rightscale.com/api/clouds/1/datacenters.xml
[api]# mycurl -X GET https://us-3.rightscale.com/api/clouds/1/security_groups.xml
```

1.4 Create Security Group

See:
http://reference.rightscale.com/api1.5/resources/ResourceSecurityGroups.html#create

create

URLs: POST /api/clouds/:cloud_id/security_groups
HTTP response code: 201 Created

Required roles

- security_manager

Parameters

name	required	type	values	regexp	blank?	description
security_group	yes	Hash	*	*	no	
security_group[description]	no	String	*	*	no	
security_group[name]	yes	String	*	*	no	

Before you can launch a server in ec2 you must have a Security Group. You will create one now using the API.

To create a Security Group you issue an `HTTP Post` to the href `/api/clouds/:cloud_id/security_groups`.

Note: Working with Security Groups requires "`security_manager`" permissions on the account. Not doing so results in an HTTP 403 Forbidden.

1.4.1 Create the Security Group

9. Create a bash script to invoke the API Request.

```
[api]# vi SecurityGroup-Create.sh
```

10. Enter the following text, replacing `CLOUD` with the Cloud ID for the cloud you're using for training, and `MYNAME` with your own name.

```
#!/bin/bash -e
CLOUD="1"
MYNAME="JDoe"

curl -i -H X_API_VERSION:1.5 -b ~/mycookie -X POST \
-d security_group[name]="$MYNAME Security Group" \
-d security_group[description]="$MYNAME API Training Security Group" \
https://us-3.rightscale.com/api/clouds/$CLOUD/security_groups
```

11. Make the script executable

```
[api]# chmod +x SecurityGroup-Create.sh
```

12. Now run the Script

Note: The '`tee SecurityGroup-Create.sh.out`' in the following command ensures the output is piped to the file `SecurityGroup-Create.sh.out`, as well as displayed on screen. You may need to refer to this output file later to retrieve certain information.

```
[api]# ./SecurityGroup-Create.sh | tee SecurityGroup-Create.sh.out
HTTP/1.1 201 Created
Server: nginx/1.0.14
Date: Mon, 11 Mar 2013 13:43:33 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Status: 201 Created
Location: /api/clouds/1/security_groups/26BERLJAK8BLC
X-Runtime: 1095
X-Request-Uuid: 2faf939f2f9e4c40b85b00eed04f314a
Set-Cookie:
Cache-Control: no-cache
```

Note the line of output similar to the following

```
Location: /api/clouds/1/security_groups/50K6AE2MB3LDL
```

You will need this Security Group ID number in future in order to manipulate and view the Security Group, and when you create your servers.

Note: This is different from the Security Group's '`Resource UID`' that you will see in the Dashboard.

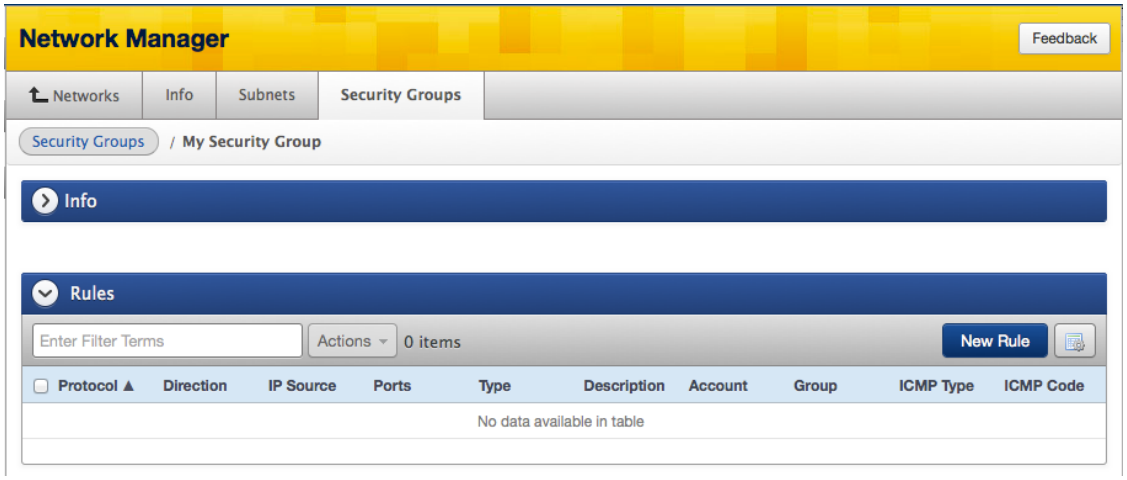
13. Make a note of the Security Group ID:-

    Security Group ID: _____

    You can always retrieve this value from the output file `SecurityGroup-Create.sh.out`.

1.4.2 Verify Security Group

14. Now browse to the Dashboard (Manage > Networks > [Select Cloud] > Security Groups) and you should see the newly created Security Group. It does not contain any rules yet.



1.4.3 Add Rules to the Security Group

See:
http://reference.rightscale.com/api1.5/resources/ResourceSecurityGroupRules.html

create

Create a security group rule for a security group.

URLs:	POST /api/security_group_rules
	POST /api/clouds/:cloud_id/security_groups/:security_group_id /security_group_rules
HTTP response code:	201 Created
Location:	Href of created SecurityGroup.

Required roles

- security_manager

Parameters

name	required	type	values	regexp	blank?	description
security_group_rule	yes	Hash	*	*	no	
security_group_rule[cidr_ips]	no	String	*	*	no	
security_group_rule[group_name]	no	String	*	*	no	
security_group_rule[group_owner]	no	String	*	*	no	
security_group_rule[protocol]	yes	String	tcp, udp, icmp	*	no	
security_group_rule[protocol_details]	yes	Hash	*	*	no	
security_group_rule[protocol_details][end_port]	no	String	*	*	no	
security_group_rule[protocol_details][icmp_code]	no	String	*	*	no	
security_group_rule[protocol_details][icmp_type]	no	String	*	*	no	
security_group_rule[protocol_details][start_port]	no	String	*	*	no	
security_group_rule[security_group_href]	no	String	*	*	no	
security_group_rule[source_type]	yes	String	cidr_ips, group	*	no	

Once you have created a Security Group, you then must create the actual rules. To create rules you can issue a `HTTP Post` to the href,

```
/api/clouds/:cloud_id/security_groups/:security_group_id/security_group_rules
```

where the `/api/clouds/:cloud_id/security_groups/:security_group_id` portion of the href is the '`Location`' header returned when in the previous step when the Security Group was created.

In this example, you will open the ports 22 and 80 in the Security Group for all IPs. In the script you will need to use the `Cloud ID` as well as the `Security Group ID` from the response of the previous command. You will parameterize these values to make future editing easier.

15. Create a bash script to invoke the API Request.

```
[api]# vi SecurityGroupRules-Create.sh
```

16. Enter the following text, replacing `CLOUD` with the Cloud ID for the cloud you're using for training, and `SG` with the Security Group ID

```
#!/bin/bash -ex

CLOUD="1"             #May need to change this
SG="50K6AE2MB3LDL"    #Will need to change this

#Open port 22 for SSH Access
curl -i -H X_API_VERSION:1.5 -b ~/mycookie -X POST \
-d security_group_rule[group_name]="SSH" \
-d security_group_rule[protocol]=tcp \
-d security_group_rule[cidr_ips]='0.0.0.0/0' \
-d security_group_rule[protocol_details][start_port]=22 \
-d security_group_rule[protocol_details][end_port]=22 \
-d security_group_rule[source_type]=cidr_ips \
https://us-
3.rightscale.com/api/clouds/$CLOUD/security_groups/$SG/security_group_rules

#Open port 80 for web access to Load Balancers
curl -i -H X_API_VERSION:1.5 -b ~/mycookie -X POST \
-d security_group_rule[group_name]="Web Access" \
-d security_group_rule[protocol]=tcp \
-d security_group_rule[cidr_ips]='0.0.0.0/0' \
-d security_group_rule[protocol_details][start_port]=80 \
-d security_group_rule[protocol_details][end_port]=80 \
-d security_group_rule[source_type]=cidr_ips \
https://us-
3.rightscale.com/api/clouds/$CLOUD/security_groups/$SG/security_group_rules

#Open port 8000 for LB to APP Server communications
curl -i -H X_API_VERSION:1.5 -b ~/mycookie -X POST \
-d security_group_rule[group_name]="LB APP Server Comms" \
-d security_group_rule[protocol]=tcp \
-d security_group_rule[cidr_ips]='0.0.0.0/0' \
-d security_group_rule[protocol_details][start_port]=8000 \
-d security_group_rule[protocol_details][end_port]=8000 \
-d security_group_rule[source_type]=cidr_ips \
https://us-
3.rightscale.com/api/clouds/$CLOUD/security_groups/$SG/security_group_rules
```

```
#Allow ICMP so we can ping the servers
curl -i -H X_API_VERSION:1.5 -b ~/mycookie -X POST \
-d security_group_rule[group_name]="LB APP Server Comms" \
-d security_group_rule[protocol]=icmp \
-d security_group_rule[cidr_ips]='0.0.0.0/0' \
-d security_group_rule[protocol_details][icmp_code]=-1 \
-d security_group_rule[protocol_details][icmp_type]=-1 \
-d security_group_rule[source_type]=cidr_ips \
https://us-
3.rightscale.com/api/clouds/$CLOUD/security_groups/$SG/security_group_rules
```

Note: You must specify the start and end port in the range – for a single port the start and the end port are the same. Also, the `0.0.0.0/0` means all IPs in CIDR notation. You cannot specify multiple start/end port combinations with a single API call.

17. Make the script executable

```
[api]# chmod +x SecurityGroupRules-Create.sh
```

18. Run the script

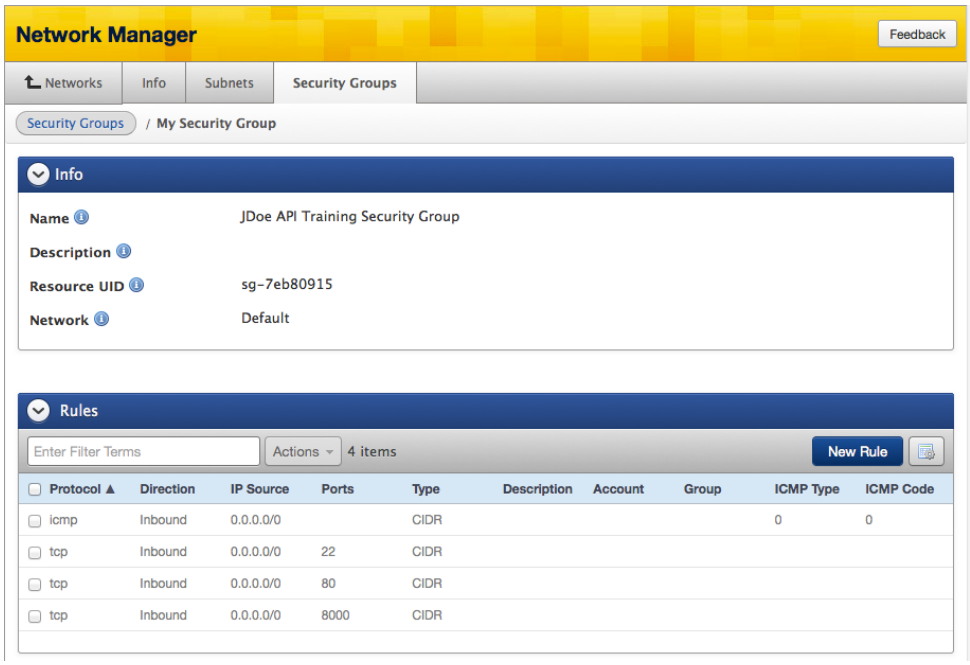
```
[api]# ./SecurityGroupRules-Create.sh | tee SecurityGroupRules-
Create.sh.out
```

Note the four hrefs in the response, e.g.

```
Location: /api/security_group_rules/351382003
Location: /api/security_group_rules/351383003
Location: /api/security_group_rules/351384003
Location: /api/security_group_rules/351385003
```

1.4.4 View the Security Group

19. Now browse to '**Clouds > *Cloud* > Security Groups**' in the Dashboard and view your Security Group.



20. Now view the Security Group by doing a `GET` on this `Location` href returned when it was created (adding the `.xml` extension to get XML returned)

```
[api]# mycurl -X GET https://us-
3.rightscale.com/api/clouds/1/security_groups/50K6AE2MB3LDL.xml
<?xml version="1.0" encoding="UTF-8"?>
<security_group>
  <actions></actions>
  <resource_uid>sg-700de81b</resource_uid>
  <links>
    <link href="/api/clouds/1/security_groups/50K6AE2MB3LDL" rel="self"/>
    <link href="/api/clouds/1" rel="cloud"/>
    <link
href="/api/clouds/1/security_groups/50K6AE2MB3LDL/security_group_rules"
rel="security_group_rules"/>
  </links>
  <name>jaf Security Group</name>
</security_group>
```

21. Now drill down to view the rules within the Security Group, by following the URLs within the response(s).

1.5 Create an SSH Key

See: http://reference.rightscale.com/api1.5/resources/ResourceSshKeys.html

create

Creates a new ssh key.

URLs:	POST /api/clouds/:cloud_id/ssh_keys
HTTP response code:	201 Created
Location:	Href of created SshKey.

Required roles

- actor

Parameters

name	required	type	values	regexp	blank?	description
ssh_key	yes	Hash	*	*	no	
ssh_key[name]	yes	String	*	*	no	The name for the SSH key to be created.

Before you can create a server in EC2, you must create an SSH Key. To crate an SSH Key you issue a `HTTP Post` to the href `/api/clouds/:cloud_id/ssh_keys`.

22. Create a bash script to invoke the API Request.

```
[api]# vi SSHKey-Create.sh
```

23. Enter the following text, replacing `CLOUD` with the Cloud ID for the cloud you're using for training, and `SG` with the Security Group ID

```
#!/bin/bash -e
CLOUD="1"
MYNAME="JDoe"

curl -i -H X_API_VERSION:1.5 -b ~/mycookie -X POST \
-d ssh_key[name]="$MYNAME SSH Key" \
https://us-3.rightscale.com/api/clouds/$CLOUD/ssh_keys
```

24. Make the script executable

```
[api]# chmod +x SSHKey-Create.sh
```

25. Run the script

```
[api]# ./SSHKey-Create.sh | tee SSHKey-Create.sh.out
HTTP/1.1 201 Created
Server: nginx/1.0.14
Date: Fri, 08 Mar 2013 16:15:26 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Status: 201 Created
Location: /api/clouds/1/ssh_keys/D8V0AVIR6F125
X-Runtime: 1240
X-Request-Uuid: 63eb377689f34ffe94563a6f3bd3b123
Set-Cookie:
Cache-Control: no-cache
```

26. Make a note of the Location of the new SSH Key, i.e.

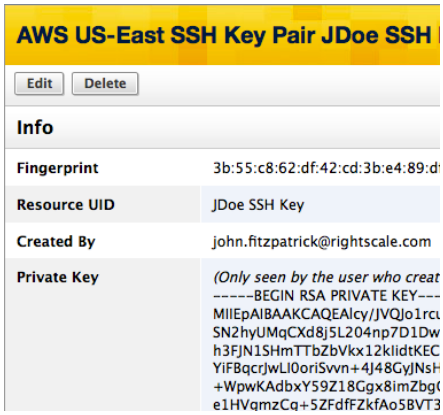
```
Location: /api/clouds/1/ssh_keys/81CP66J7228BF
```

SSH Key ID: _____

You will need this href later when you create your servers

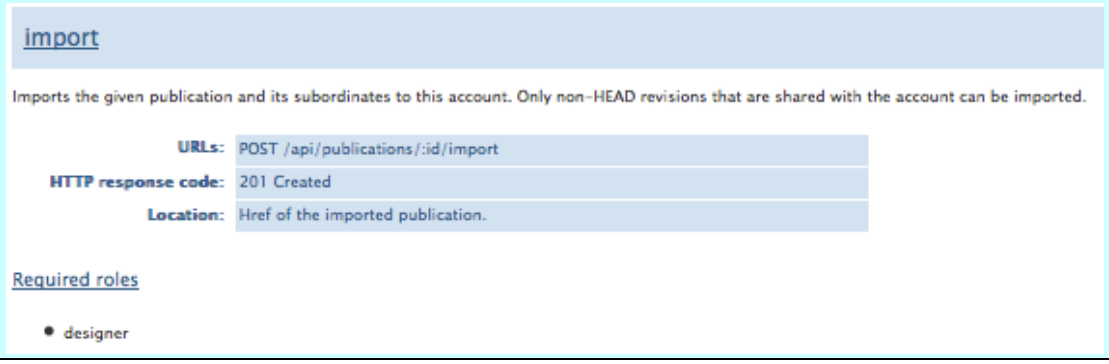
1.5.1 Verify SSH Key

27. Now browse to '**Clouds > *Cloud* > SSH Keys**' in the Dashboard and view your SSH Key.



1.6 Import the Required ServerTemplate

See:
http://reference.rightscale.com/api1.5/resources/ResourcePublications.html#import



import

Imports the given publication and its subordinates to this account. Only non-HEAD revisions that are shared with the account can be imported.

URLs:	POST /api/publications/:id/import
HTTP response code:	201 Created
Location:	Href of the imported publication.

Required roles

- designer

To import ServerTemplate you need to issue an `HTTP Post` to the href `/api/publications/:id/import`., where '`:id`" is the ServerTemplate ID Number.

To create your server you will need to import the ServerTemplate '**Base ServerTemplate for Linux**'.

28. In the Dashboard, browse to '**Design > MultiCLoud Marketplace > ServerTemplates**' and search for '**Base ServerTemplate for Linux (v13.3)**'
29. Click the 'Import' link

30. Make a note of the 'Object ID' in the URL displayed, e.g. 48499,

    https://us-3.rightscale.com/library/server_templates/Base-ServerTemplate-for-Linux-/48499

NOTE: Please note that this ID is likely to change as new revisions of the object are released, and the value shown above (48499) is correct at time of going to press.

'**Base ServerTemplate for Linux**' ServerTemplate ID#: _____

1.6.1 Import ServerTemplates

31. Create the bash script `ServerTemplates_Import.sh`

This script will invoke the API Request to import the ServerTemplates.

```
[api]# vi ServerTemplates_Import.sh
```

32. Enter the following text, replacing `ID` with the ID for the ServerTemplate determined in Step 28 above.

```
#!/bin/bash -e
ST="48499"

echo "Importing ServerTemplate"
curl -i -H X_API_VERSION:1.5 -b ~/mycookie -X POST \
https://us-3.rightscale.com/api/publications/$ST/import
```

33. Make the script executable

```
[api]# chmod +x ServerTemplates_Import.sh
```

34. Now run the Script, capturing the output in a file, as well as outputting to screen

```
[api]# ./ServerTemplates_Import.sh | tee ServerTemplates_Import.sh.out
```

Note: This script may take a few seconds to complete.

1.7 Create Deployment

See: http://reference.rightscale.com/api1.5/resources/ResourceDeployments.html

create

Creates a new deployment with the given parameters.

URLs:	POST /api/deployments
HTTP response code:	201 Created
Location:	Href of the created deployment

Required roles

- actor

Required settings

- premium_deployments

Parameters

name	required	type	values	regexp	blank?	description
deployment	yes	Hash	*	*	no	
deployment[description]	no	String	*	*	yes	The description of the deployment to be created.
deployment[name]	yes	String	*	*	no	The name of the deployment to be created.
deployment[server_tag_scope]	no	String	deployment, account	*	yes	The routing scope for tags for servers in the deployment.

You will now create a new Deployment as a container your Server. To create a deployment you issue a `HTTP Post` to the href `/api/deployments`. The parameter `deployment[name]` is mandatory.

35. Create the bash script `Deployment-Create.sh`

    You'll use this script to invoke the API Request to create the Deployment.

```
[api]# vi Deployment-Create.sh
```

36. Enter the following text, replacing `MYNAME` with the your own name.

    This name will for part of the Deployment Name and Description

```
#!/bin/bash -e
MYNAME="JDoe"

curl -i -H X_API_VERSION:1.5 -b ~/mycookie -X POST \
-d deployment[name]="$MYNAME Deployment" \
-d deployment[description]="$MYNAME Deployment created by API" \
https://us-3.rightscale.com/api/deployments
```

37. Make the script executable

```
[api]# chmod +x Deployment-Create.sh
```

38. Run the Script, and capture the output to file (as well as display on screen)

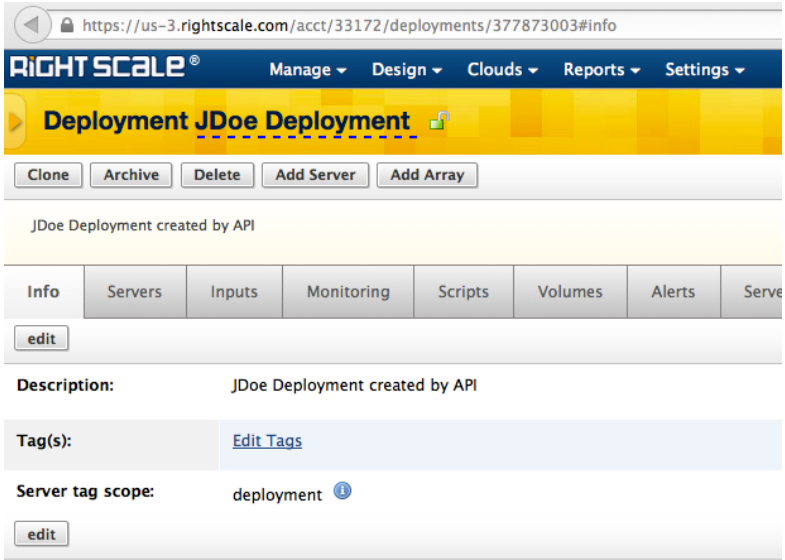
```
[api]# ./Deployment-Create.sh | tee Deployment-Create.sh.out
HTTP/1.1 201 Created
Server: nginx/1.0.14
Date: Mon, 11 Mar 2013 12:20:39 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Status: 201 Created
Location: /api/deployments/350944003
X-Runtime: 56
X-Request-Uuid: 62e9a4b694b943af9fcfb459b254e68f
Set-Cookie:
Cache-Control: no-cache
```

39. Make a note of the Deployment ID returned in the API Response (e.g. '`350944003`' in the example shown). You will use this number to interact with the Deployment in future API calls

    Deployment ID: _____

1.7.1 Verify Deployment

40. You can browse to the Dashboard and verify the Deployment exists



You will notice that the Deployment ID in the URL bar of the browser matches the `Location` href returned when the Deployment was created.

41. You can also invoke an API request to the `Location` returned, as follows

```
[api]# mycurl -X GET https://us-3.rightscale.com/api/deployments/350944003.xml
<?xml version="1.0" encoding="UTF-8"?>
<deployment>
  <server_tag_scope>deployment</server_tag_scope>
  <actions>
    <action rel="clone"/>
  </actions>
  <links>
    <link href="/api/deployments/350944003" rel="self"/>
    <link href="/api/deployments/350944003/servers" rel="servers"/>
    <link href="/api/deployments/350944003/server_arrays" rel="server_arrays"/>
    <link href="/api/deployments/350944003/inputs" rel="inputs"/>
  </links>
  <description>3 Tier Deployment created by API</description>
  <name>myname 3 Tier Deployment</name>
</deployment>
```

1.8 Create Server

See: http://reference.rightscale.com/api1.5/resources/ResourceServers.html

create

Creates a new server, and configures its corresponding "next" instance with the received parameters.

URLs:	POST /api/servers
	POST /api/deployments/:deployment_id/servers
HTTP response code:	201 Created
Location:	Href of the created Server

Required roles

- actor

Parameters

name	required	type	values	regexp	blank?	description
server	yes	Hash	*	*	no	
server[deployment_href]	no	String	*	*	no	The href of the deployment to which the Server will be added.
server[description]	no	String	*	*	no	The server description.
server[instance]	yes	Hash	*	*	no	
server[instance][cloud_href]	yes	String	*	*	no	The href of the cloud that the Server should be added to.
server[instance][datacenter_href]	no	String	*	*	no	The href of the Datacenter / Zone.
server[instance][image_href]	no	String	*	*	no	The href of the Image to use.
server[instance][inputs]	no	Enumerable	*	*	no	
server[instance][inputs][*]	no	String	*	*	no	The format used for passing 2.0-style Inputs. The key is the name of the input, and the value is the value to assign to

You will now create the Servers. To create a Server in a particular deployment you can issue an `HTTP Post` to the href `/api/deployments/:deployment_id/servers`.

Alternatively you can issue an `HTTP Post` to the href `/api/servers` and pass the Deployment ID as a parameter. In this example you will use the latter method.

If you refer to the Reference Documentation for the Servers resource, you will see there are quite a few parameters associated the '`create`' method. For clarity you will only populate the mandatory parameters, plus a few others that are relevant for the cloud you're using.

42. Create the script you'll use to create the Server

```
[api]# vi Server-Create.sh
```

43. Enter the following text replacing the correct values for

- Deployment ID (`$DEPLOYMENT`)
- Cloud ID (`$CLOUD`)
- ServerTemplate ID (`$ST`)
- Security Group (`$SG`)

- SSH Key (`$SSH`)

You may need to read through the output from previous scripts to get this information, .e.g.

```
[api]# grep Location *.out
Deployment-Create.sh.out:Location: /api/deployments/350938003
SecurityGroup-Create.sh.out:Location: /api/clouds/1/security_groups/50K6AE2MB3LDL
ServerTemplates_Import.sh.out:Location: /api/server_templates/282920003
SSHKey-Create.sh.out:Location: /api/clouds/1/ssh_keys/2OSIPDJU7Q55G
```

Your script should look like the following

```
#!/bin/bash -e
DEPLOYMENT="350938003"  # Deployment to add Server to
CLOUD="1"               # Specify Cloud to add Server to
ST="282920003"          # Set the Server ServerTemplate ID
SG="50K6AE2MB3LDL"      # Set the Security Group
SSH="2OSIPDJU7Q55G"     # Set the SSH Key
MYNAME="JDoe"           # Enter your name

echo "Creating Server"
curl -i -H X_API_VERSION:1.5 -b ~/mycookie -X POST \
-d server[name]="$MYNAME Sample Server" \
-d server[description]="$MYNAME Sample API Server" \
-d server[deployment_href]=/api/deployments/$DEPLOYMENT \
-d server[instance][cloud_href]=/api/clouds/$CLOUD \
-d server[instance][server_template_href]=/api/server_templates/$ST \
-d
server[instance][security_group_hrefs][]=/api/clouds/$CLOUD/security_group
s/$SG \
-d server[instance][ssh_key_href]=/api/clouds/$CLOUD/ssh_keys/$SSH \
https://us-3.rightscale.com/api/servers
```

44. Please take a few minutes to read though this script and the corresponding API Reference documentation to appreciate how the API comments in the script are put together.

    http://reference.rightscale.com/api1.5/resources/ResourceServers.html

45. Make the script executable

```
[api]# chmod +x Server-Create.sh
```

46. Run the script, saving the output a file as well as displaying it on screen

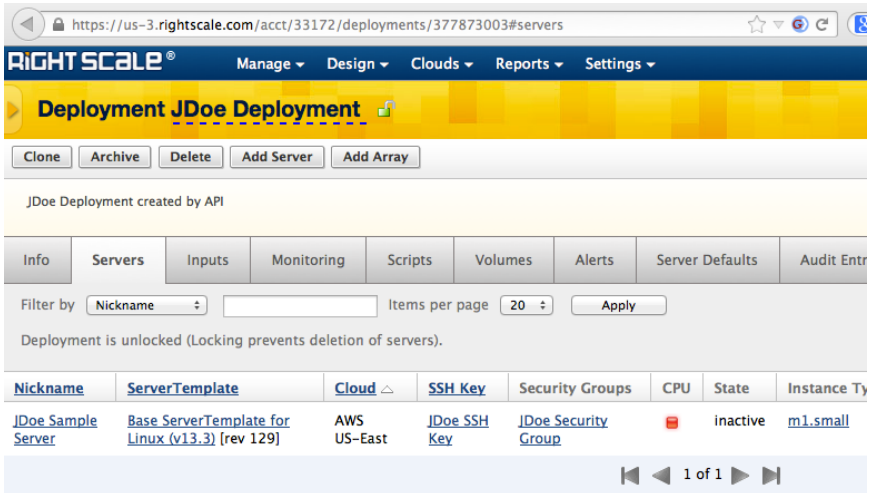
```
[api]# ./Server-Create.sh | tee Server-Create.sh.out
Creating Server
HTTP/1.1 201 Created
Server: nginx/1.0.14
Date: Mon, 15 Apr 2013 17:13:10 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Status: 201 Created
Location: /api/servers/747403003
X-Runtime: 770
X-Request-Uuid: 56acbbb4607441728b83e86cbe5e6668
Set-Cookie:
Cache-Control: no-cache
```

47. Make a note of the Location URL for each server from the API Response

    `Location: /api/servers/_____`

1.8.1 Verify Server

48. To verify the Server has been create correctly, browse to the Deployment in the Dashboard



Click the server and verify the Server ID as shown in the Dashboard is the same as that returned in the `Location` header.

Now invoke the following API command to list the contents of your Deployment (swapping `350944003` for your Deployment ID)

```
[api]# mycurl -X GET https://us-
3.rightscale.com/api/deployments/350944003/servers.xml
<?xml version="1.0" encoding="UTF-8"?>
<servers>
  <server>
    <actions>
      <action rel="launch"/>
      <action rel="clone"/>
    </actions>
    <created_at>2013/04/15 13:47:51 +0000</created_at>
    <updated_at>2013/04/15 13:47:52 +0000</updated_at>
    <description>Load Balancer server</description>
    <links>
      <link href="/api/servers/747399003" rel="self"/>
      <link href="/api/deployments/377873003" rel="deployment"/>
      <link href="/api/clouds/1/instances/6TKNMAHB7Q2Q"
rel="next_instance"/>
      <link href="/api/servers/747399003/alert_specs" rel="alert_specs"/>
    </links>
    <name>JDoe Sample Server</name>
    <state>inactive</state>
  </server>
</servers>
```

1.9 Launch Server

See: http://reference.rightscale.com/api1.5/resources/ResourceInstances.html#launch

launch

Launches an instance using the parameters that this instance has been configured with.

Note that this action can only be performed in "next" instances, and not on instances that are already running.

URLs: POST /api/clouds/:cloud_id/instances/:id/launch
POST /api/servers/:server_id/launch
POST /api/server_arrays/:server_array_id/launch

HTTP response code: 201 Created

Location: Href of the launched Instance

Required roles

- actor

Parameters

name	required	type	values	regexp	blank?	description
inputs	no	Enumerable	*	*	no	
inputs[*]	no	String	*	*	no	The format used for passing 2.0-style Inputs. The key is the name of the input, and the value is the value to assign to the input. For more details on 2.0-style inputs, please see Inputs#multi_update.
inputs[][name]	no	String	*	*	no	The input name. This format is used for passing legacy 1.0-style Inputs. Will eventually be deprecated.
inputs[][value]	no	String	*	*	no	The value of that input. Should be of the form 'text:my_value' or 'cred:MY_CRED' etc. This format is used for passing legacy 1.0-style Inputs. Will eventually be deprecated.

49. You will now launch your server. To launch the server you will invoke a `HTTP Post` to the href `/api/servers/:server_id/launch`.

    This particular ServerTemplate has default values for all inputs, so no changes are necessary. You will update inputs in a later example.

50. Create the script `Server-Launch.sh` with the following text

```
[api]# vi Server-Launch.sh
#!/bin/bash -e

SERVER="747403003"

curl -i -H X_API_VERSION:1.5 -b ~/mycookie -X POST \
https://us-3.rightscale.com/api/servers/$SERVER/launch.xml
```

51. Set the value for `SERVER` to the ID Number for your Server. This was returned in the Response for the API request that created it – see Step 47 .

    You can get the value from the output file `Servers-Create.sh.out`.

```
[api]# grep Location Servers-Create.sh.out
Location: /api/servers/747403003
```

or you can click the server in the Dashboard and make a note of the ID number from the browser URL

52. Make the script executable

```
[api]# chmod +x Server-Launch.sh
```

53. Run the script, sending output to file as well as to screen

```
[api]# ./Server-Launch.sh | tee Server-Launch.sh.out
HTTP/1.1 201 Created
Server: nginx/1.0.14
Date: Tue, 12 Mar 2013 14:28:39 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Status: 201 Created
Location: /api/clouds/1/instances/969GBPOIMG8QI
X-Runtime: 9472
X-Request-Uuid: 0c2713ca057d4d6a9c417765c499b0c3
Set-Cookie:
Cache-Control: no-cache
```

54. Note the `current_instance` ID, returned in the `Location` line of the API Response, e.g.

```
Location: /api/clouds/1/instances/969GBPOIMG8QI
```

This ID is used for all interactions with this server instance.

55. The Server to become operational in several minutes, depending on your cloud etc. To check progress, you can browse to the Dashboard.

Nickname	ServerTemplate	Cloud	SSH Key	Security Groups	CPU	State	Instance Type
JDoe Sample Server	Base ServerTemplate for Linux (v13.3) [rev 129]	AWS US-East	JDoe SSH Key	JDoe Security Group		pending	m1.small

56. Alternatively you can invoke a `HTTP GET` to the href to view more information on this instance.

```
[api]# mycurl -X GET https://us-
3.rightscale.com/api/clouds/1/instances/969GBPOIMG8QI.xml
<?xml version="1.0" encoding="UTF-8"?>
<instance>
  <public_ip_addresses></public_ip_addresses>
  <actions>
    <action rel="terminate"/>
    <action rel="run_executable"/>
  </actions>
  <created_at>2013/04/15 14:14:00 +0000</created_at>
  <resource_uid>i-6a7f0400</resource_uid>
  <updated_at>2013/04/15 14:14:30 +0000</updated_at>
  <pricing_type>fixed</pricing_type>
  <private_ip_addresses></private_ip_addresses>
  <links>
    <link href="/api/clouds/1/instances/FDU3R63K2C613" rel="self"/>
    <link href="/api/clouds/1" rel="cloud"/>
    <link href="/api/deployments/377873003" rel="deployment"/>
    <link href="/api/server_templates/290025003" rel="server_template"/>
    <link href="/api/multi_cloud_images/310816003" rel="multi_cloud_image"/>
    <link href="/api/servers/747399003" rel="parent"/>
    <link href="/api/clouds/1/instances/FDU3R63K2C613/volume_attachments"
rel="volume_attachments"/>
    <link href="/api/clouds/1/instances/FDU3R63K2C613/inputs" rel="inputs"/>
    <link href="/api/clouds/1/instances/FDU3R63K2C613/monitoring_metrics"
rel="monitoring_metrics"/>
  </links>
  <name>JDoe Sample Server</name>
  <state>pending</state>
</instance>
```

Section 2. Viewing Deployments and Servers

2.1 List All Deployments

See: http://reference.rightscale.com/api1.5/resources/ResourceDeployments.html#index

index

Lists deployments of the account.

Using the available filters, one can select or group which deployments to retrieve. The 'inputs_2_0' view is for retrieving inputs in 2.0 serialization (for more details please see Inputs#index.)

URLs:	GET /api/deployments
HTTP response code:	200 OK
Content-type:	application/vnd.rightscale.deployment;type=collection

Required roles

- observer

Parameters

name	required	type	values	regexp	blank?	description
filter	no	Array	*	*	no	See below for valid filter parameters.
view	no	String	default, inputs, inputs_2_0	*	no	Specifies how many attributes and/or expanded nested relationships to include.

Filters

name	partial_match?	description
description	yes	The deployment description field to filter on.
name	yes	The name of the deployment to filter on.
server_tag_scope	yes	The routing scope for tags for servers in the deployment.

To list all the Deployments in an account, i.e. index the `deployments` resource, you issue an `HTTP GET` to the href `/api/deployments`.

57. List all the deployments in the account using the following command:-

```
[api]# mycurl -X GET https://us-3.rightscale.com/api/deployments.xml
<?xml version="1.0" encoding="UTF-8"?>
<deployments>
  <deployment>
    <links>
      <link href="/api/deployments/265165003" rel="self"/>
      <link href="/api/deployments/265165003/servers" rel="servers"/>
      <link href="/api/deployments/265165003/server_arrays"
            rel="server_arrays"/>
      <link href="/api/deployments/265165003/inputs" rel="inputs"/>
    </links>
    <actions>
      <action rel="clone"/>
    </actions>
    <server_tag_scope>deployment</server_tag_scope>
    <description>General deployment for testing and debugging, and to run
                 individual servers.</description>
    <name>Default</name>
  </deployment>
  <deployment>
    <links>
      <link href="/api/deployments/339865003" rel="self"/>
      <link href="/api/deployments/339865003/servers" rel="servers"/>
      <link href="/api/deployments/339865003/server_arrays"
            rel="server_arrays"/>
      <link href="/api/deployments/339865003/inputs" rel="inputs"/>
    </links>
    <actions>
      <action rel="clone"/>
    </actions>
    <server_tag_scope>deployment</server_tag_scope>
    <description>John's Test Deployment created by the API</description>
    <name>John's Test API Deployment</name>
  </deployment>
</deployments>
```

2.2 Filter the Response

If you refer to the API Documentation (see link above), you will see there are three filters available for the 'Deployments' resource.

Filters

name	partial_match?	description
description	yes	The deployment description field to filter on.
name	yes	The name of the deployment to filter on.
server_tag_scope	yes	The routing scope for tags for servers in the deployment.

58. Now filter your results on 'name' to display a specific Deployment, as follows (you can use the `MYNAME` value as you used in the Deployment creation script, `Deployment-Create.sh` as search criteria)

```
[api]# mycurl -X GET -d filter[]="name==jdoe" \
https://us-3.rightscale.com/api/deployments.xml
<?xml version="1.0" encoding="UTF-8"?>
<deployments>
  <deployment>
    <links>
      <link href="/api/deployments/339864003" rel="self"/>
      <link href="/api/deployments/339864003/servers" rel="servers"/>
      <link href="/api/deployments/339864003/server_arrays"
            rel="server_arrays"/>
      <link href="/api/deployments/339864003/inputs" rel="inputs"/>
    </links>
    <actions>
      <action rel="clone"/>
    </actions>
    <server_tag_scope>deployment</server_tag_scope>
    <description>John's Test Deployment created by the API</description>
    <name>John's Test API Deployment</name>
  </deployment>
</deployments>
```

Note: The filter is not case sensitive, so `-d filter[]="name==TEST"` will give the same result as `-d filter[]="name==test"`

2.3 List Servers in the Deployment

See: http://reference.rightscale.com/api1.5/resources/ResourceServers.html#index

index

Lists servers.

By using the available filters, it is possible to retrieve servers that have common characteristics. For example, one can list:

- servers that have names that contain "app_server"
- all servers of a given deployment

For more filters, please see the 'index' action on 'Instances' resource as most of the attributes belong to a 'current_instance' than to a server.

URLs:	GET /api/servers
	GET /api/deployments/:deployment_id/servers
HTTP response code:	200 OK
Content-type:	application/vnd.rightscale.server;type=collection

Required roles

- observer

Parameters

name	required	type	values	regexp	blank?	description
filter	no	Array	*	*	no	See below for valid filter parameters.
view	no	String	default, instance_detail	*	no	Specifies how many attributes and/or expanded nested relationships to include.

Filters

name	partial_match?	description
cloud_href	no	The href of the Cloud to filter on.
deployment_href	no	The href of the Deployment to filter on.
name	yes	The name of the Server to filter on.

To list the servers in a deployment, you perform a `HTTP GET` request to the href `/api/deployments/<DeploymentID>/servers.xml`

This url is exposed in the output of the previous command used view deployment details, i.e.

```
<link href="/api/deployments/339864003/servers" rel="servers"/>
```

59. Make a note of the ID of the Deployment returned from the previous command, then execute the following command (substituting the appropriate Deployment ID)

```
[api]# mycurl -X GET \
https://us-3.rightscale.com/api/deployments/339864003/servers.xml
<?xml version="1.0" encoding="UTF-8"?>
<servers>
  <server>
    <actions>
      <action rel="terminate"/>
      <action rel="clone"/>
    </actions>
    <created_at>2013/04/15 13:47:51 +0000</created_at>
    <updated_at>2013/04/15 14:14:01 +0000</updated_at>
    <description>JDoe's Sample API Server </description>
    <links>
      <link href="/api/servers/747399003" rel="self"/>
      <link href="/api/deployments/377873003" rel="deployment"/>
      <link href="/api/clouds/1/instances/FDU3R63K2C613"
rel="current_instance"/>
      <link href="/api/clouds/1/instances/6TKNMAHB7Q2Q" rel="next_instance"/>
      <link href="/api/servers/747399003/alert_specs" rel="alert_specs"/>
    </links>
    <name>JDoe Sample Server</name>
    <state>operational</state>
  </server>
</servers>
```

2.4 Changing the View

The API Documentation states that there are also two `views` available for the `servers` resource. These are implemented as parameters.

Parameters

name	required	type	values	regexp	blank?	description
filter	no	Array	*	*	no	See below for valid filter parameters.
view	no	String	default, instance_detail	*	no	Specifies how many attributes and/or expanded nested relationships to include.

60. Now run the same command, but change the view to see more information on the inputs

```
[api]# mycurl -X GET -d view=instance_detail \
https://us-3.rightscale.com/api/deployments/373168003/servers.xml
<?xml version="1.0" encoding="UTF-8"?>
<servers>
  <server>
    <actions>
      <action rel="terminate"/>
      <action rel="clone"/>
    </actions>
    <created_at>2013/04/15 13:47:51 +0000</created_at>
    <current_instance>
      <public_ip_addresses>
        <public_ip_address>54.224.107.86</public_ip_address>
      </public_ip_addresses>
      <actions>
        <action rel="terminate"/>
        <action rel="reboot"/>
        <action rel="run_executable"/>
      </actions>
      <created_at>2013/04/15 14:14:00 +0000</created_at>
      <resource_uid>i-6a7f0400</resource_uid>
      <updated_at>2013/04/15 14:21:11 +0000</updated_at>
      <pricing_type>fixed</pricing_type>
      <private_ip_addresses>
        <private_ip_address>10.141.188.98</private_ip_address>
      </private_ip_addresses>
      <links>
        <link href="/api/clouds/1/instances/FDU3R63K2C613" rel="self"/>
        <link href="/api/clouds/1" rel="cloud"/>
        <link href="/api/deployments/377873003" rel="deployment"/>
        <link href="/api/server_templates/290025003" rel="server_template"/>
        <link href="/api/multi_cloud_images/310816003" rel="multi_cloud_image"/>
        <link href="/api/servers/747399003" rel="parent"/>
        <link href="/api/clouds/1/instances/FDU3R63K2C613/volume_attachments"
rel="volume_attachments"/>
        <link href="/api/clouds/1/instances/FDU3R63K2C613/inputs" rel="inputs"/>
```

```
        <link href="/api/clouds/1/instances/FDU3R63K2C613/monitoring_metrics"
rel="monitoring_metrics"/>
      </links>
      <name>JDoe Sample Server</name>
      <state>operational</state>
    </current_instance>
    <updated_at>2013/04/15 14:14:01 +0000</updated_at>
    <next_instance>
      <public_ip_addresses></public_ip_addresses>
      <actions></actions>
      <created_at>2013/04/15 13:47:52 +0000</created_at>
      <resource_uid>11336a3a-a5d3-11e2-956e-12313901ce13</resource_uid>
      <updated_at>2013/04/15 13:47:52 +0000</updated_at>
      <pricing_type>fixed</pricing_type>
      <private_ip_addresses></private_ip_addresses>
      <links>
        <link href="/api/clouds/1/instances/6TKNMAHB7Q2Q" rel="self"/>
        <link href="/api/clouds/1" rel="cloud"/>
        <link href="/api/deployments/377873003" rel="deployment"/>
        <link href="/api/server_templates/290025003" rel="server_template"/>
        <link inherited_source="server_template" href="/api/multi_cloud_images/310816003"
rel="multi_cloud_image"/>
        <link href="/api/servers/747399003" rel="parent"/>
        <link href="/api/clouds/1/instances/6TKNMAHB7Q2Q/volume_attachments"
rel="volume_attachments"/>
        <link href="/api/clouds/1/instances/6TKNMAHB7Q2Q/inputs" rel="inputs"/>
        <link href="/api/clouds/1/instances/6TKNMAHB7Q2Q/monitoring_metrics"
rel="monitoring_metrics"/>
      </links>
      <name>JDoe Sample Server</name>
      <state>inactive</state>
    </next_instance>
    <description>JDoe's Sample API Server</description>
    <links>
      <link href="/api/servers/747399003" rel="self"/>
      <link href="/api/deployments/377873003" rel="deployment"/>
      <link href="/api/clouds/1/instances/FDU3R63K2C613" rel="current_instance"/>
      <link href="/api/clouds/1/instances/6TKNMAHB7Q2Q" rel="next_instance"/>
```

```
      <link href="/api/servers/747399003/alert_specs" rel="alert_specs"/>
    </links>
    <name>JDoe Sample Server</name>
    <state>operational</state>
  </server>
</servers>
```

Section 3. Terminating and Deleting

3.1 Terminate the Server

See: http://reference.rightscale.com/api1.5/resources/ResourceInstances.html#terminate

terminate

Terminates a running instance.

Note that this action can succeed only if the instance is running. One cannot terminate instances of type "next".

URLs:	POST /api/clouds/:cloud_id/instances/:id/terminate
	POST /api/servers/:server_id/terminate
HTTP response code:	204 No Content

Required roles

- actor

You will now terminate your server. As the API Reference indicates, you can do this in one of two ways:-

- Issue an `HTTP POST` to the href `/api/servers/:server_id/terminate`.

  This uses the `Location` href returned in step 47 above when the **server was created** (or from `Server-Create.sh.out`)

- Issue an `HTTP POST` to the `/api/clouds/:cloud_id/instances/:id/terminate`

  In this case you use the `Location` href returned in step 53 above when this **instance was launched** (or from `Server-Launch.sh.out`)

In this example you'll use second method.

61. Create the script `Server-Terminate.sh` with the following text

```
[api]# vi Server-Terminate.sh
#!/bin/bash -ex

INSTANCE_HREF="/api/clouds/1/instances/969GBPOIMG8QI"

curl -i -H X-API-Version:1.5 -b ~/mycookie -X POST \
https://us-3.rightscale.com$INSTANCE_HREF/terminate
```

Replace `INSTANCE_HREF` with the appropriate values from the file `Server-Launch.sh.out` (i.e. `grep Location Server-Launch.sh.out`)

62. Make the script executable

```
[api]# chmod +x Server-Terminate.sh
```

63. Now run the script

```
[api]# ./Server-Terminate.sh | tee Server-Terminate.sh.out
HTTP/1.1 204 No Content
Server: nginx/1.0.14
Date: Tue, 16 Apr 2013 10:28:29 GMT
Connection: keep-alive
Status: 204 No Content
X-Runtime: 1185
X-Request-Uuid: 9eb512f75a8347c3a6b1abda7f0c8dc0
Set-Cookie: rs_gbl=eNotkN1ugjAARt-l1zTpH9CS7EJBxTiGoBK9MqUgY1OYWGbV-
O7DZJdf8p1zcR5AAg_0F0iBBYoL8B7DKDvgEcYYfVpAK-Bh6jgYcYyEBepi-
IsydwjiCErHtSHGJYECMQKpcqVAnFD3wAefLv9ZbNMXO-jBYcvL1UR8z7iDarnQ-
3Oyav1C1GGm1gnSt90mXuQVbdaynUJCwuCERskpjJaBSFCYdCbKRvb89pGeryzteNVIP4x27yZ
ii5jkavMDk0Cl5nNbra7TrMng0eeTkRlXnOq0MrNfwcaoTvs4Pn5JfcfLQLbEnDoHz9006Euf3
s9md7nu0dsriXklkUq1faOBRyl2yfP5BxePW20%3D; domain=.rightscale.com; path=/;
HttpOnly
Cache-Control: no-cache
```

64. Browse to the Dashboard and verify the Server has been terminated

Nickname	ServerTemplate	Cloud	SSH Key	Security Groups	CPU	State	Instance Type	Public IP
JDoe Sample Server	Base ServerTemplate for Linux (v13.3) [rev 129]	AWS US-East	JDoe SSH Key	JDoe Security Group		terminating	m1.small	54.224.170.101

Note: To terminate all instances running in a particular you could use the command `mycurl -X POST -d terminate_all=true https://us-3.rightscale.com/api/clouds/1/instances/multi_terminate`

3.2 Delete (Destroy) the Server

See: http://reference.rightscale.com/api1.5/resources/ResourceServers.html#destroy

destroy

Deletes a given server.

URLs:	DELETE /api/servers/:id
	DELETE /api/deployments/:deployment_id/servers/:id
HTTP response code:	204 No Content

Required roles

- actor

Now you will delete all servers in your deployment. As the API Reference indicates you can do this in one of two ways:-

- Issue an `HTTP DELETE` to the href `/api/servers/:id`
- Issue an `HTTP DELETE` to the `/api/deployments/:deployment_id/servers/:id`

In this example you'll use first method.

65. Create the script `Server-Destroy.sh` with the following text

```
[api]# vi Server-Destroy.sh
#!/bin/bash -ex

SERVER_HREF="/api/servers/747403003"

curl -i -H X-API-Version:1.5 -b ~/mycookie -X DELETE \
https://us-3.rightscale.com$SERVER_HREF
```

Replace `SERVER_HREF` with the appropriate values from the file `Server-Create.sh.out` (i.e. `grep Location Server-Create.sh.out`)

66. Make the script executable

```
[api]# chmod +x Server-Destroy.sh
```

67. Now run the script

```
[api]# ./Server-Destroy.sh | tee Server-Destroy.sh.out
```

3.3 Delete (Destroy) the Deployment

See: http://reference.rightscale.com/api1.5/resources/ResourceDeployments.html#destroy

destroy

Deletes a given deployment.

URLs:	DELETE /api/deployments/:id
HTTP response code:	204 No Content

Required roles

- actor

Required settings

- premium_deployments

To delete (or `Destroy`) a Deployment you issue a `HTTP Delete` to the href `/api/deployments/:id`.

Now delete the deployment. It is possible to do this from the bash script, but you can also do it from the command line as shown below.

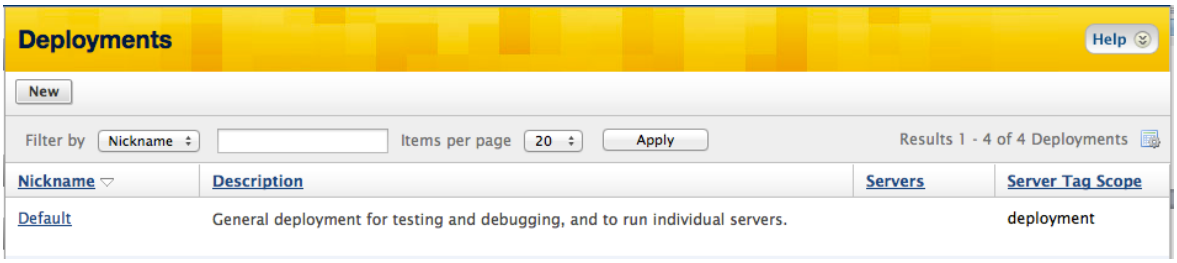
68. Determine your Deployment ID, either from the Dashboard or by running the following command

```
[api]# grep Location Deployment-Create.sh.out
Location: /api/deployments/377873003
```

69. Invoke the following command, replacing the Deployment ID as appropriate

```
[api]# mycurl -X DELETE https://us-3.rightscale.com/api/deployments/377873003
```

70. Browse to Dashboard again and verify the Deployment has been deleted



3.4 Delete (Destroy) Security Group

See: http://reference.rightscale.com/api1.5/resources/ResourceSecurityGroups.html

destroy

URLs:	DELETE /api/clouds/:cloud_id/security_groups/:id
HTTP response code:	204 No Content

Required roles

- security_manager

Now you will delete your Security Group. As the API Reference indicates you can do this by issuing an `HTTP DELETE` to the href `/api/clouds/:cloud_id/security_groups/:id`

71. You can find the correct HREF value in the file `SecurityGroup-Create.sh.out`

```
[api]# grep Location SecurityGroup-Create.sh.out
Location: /api/clouds/1/security_groups/7IHT5MDOJU1DJ
```

72. Create the script `SecurityGroup-Destroy.sh` with the following text

```
[api]# vi SecurityGroup-Destroy.sh
#!/bin/bash -ex
SECGROUP_HREF="/api/clouds/1/security_groups/7IHT5MDOJU1DJ"

curl -i -H X-API-Version:1.5 -b ~/mycookie -X DELETE \
https://us-3.rightscale.com$SECGROUP_HREF
```

Replace `SECGROUP_HREF` with the appropriate Security Group ID

73. Make the script executable

```
[api]# chmod +x SecurityGroup-Destroy.sh
```

74. Now run the script

```
[api]# ./SecurityGroup-Destroy.sh | tee SecurityGroup-Destroy.sh.out
```

3.5 Delete (Destroy) SSH Key

See: http://reference.rightscale.com/api1.5/resources/ResourceSshKeys.html#destroy

destroy

Deletes a given ssh key.

URLs:	DELETE /api/clouds/:cloud_id/ssh_keys/:id
HTTP response code:	204 No Content

Required roles

- actor

Now you will delete your SSH Key. As the API Reference indicates you can do this by issuing an `HTTP DELETE` to the href `/api/clouds/:cloud_id/ssh_keys/:id`

75. You can find the correct HREF value in the file `SSHKey-Create.sh.out`

```
[api]# grep Location SSHKey-Create.sh.out
Location: /api/clouds/1/ssh_keys/38OTC4OMVMCA5
```

76. Create the script `SecurityGroup-Destroy.sh` with the following text

```
[api]# vi SSHKey-Destroy.sh
#!/bin/bash -ex
SSHKey_HREF="/api/clouds/1/ssh_keys/38OTC4OMVMCA5"

curl -i -H X-API-Version:1.5 -b ~/mycookie -X DELETE \
https://us-3.rightscale.com$SSHKey_HREF
```

Replace `SSHKey_HREF` with the appropriate SSH Key ID

77. Make the script executable

```
[api]# chmod +x SSHKey-Destroy.sh
```

78. Now run the script

```
[api]# ./SSHKey-Destroy.sh | tee SSHKey-Destroy.sh.out
```