# OpenStack client and API

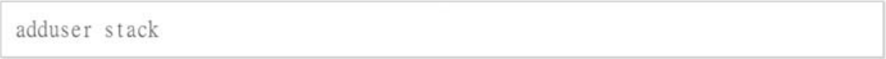
**Tien-Fu Chen**

Dept. of Computer Science and Information Engineering

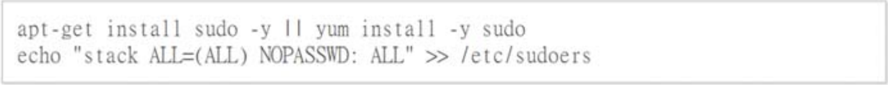
**National Chiao Tung Univ.**

## DevStack: All-In-One Single Machine

1. **Add your user**

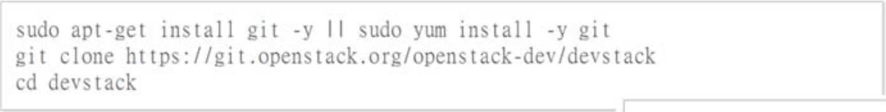


```
adduser stack
```



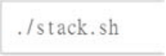
```
apt-get install sudo -y || yum install -y sudo
echo "stack ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
```

2. **Download DevStack**



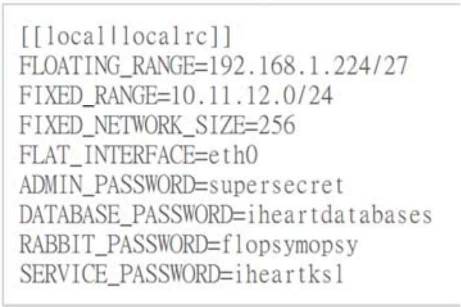
```
sudo apt-get install git -y || sudo yum install -y git
git clone https://git.openstack.org/openstack-dev/devstack
cd devstack
```

3. **Run DevStack**



```
./stack.sh
```

4. local.conf



```
[[local|localrc]]
FLOATING_RANGE=192.168.1.224/27
FIXED_RANGE=10.11.12.0/24
FIXED_NETWORK_SIZE=256
FLAT_INTERFACE=eth0
ADMIN_PASSWORD=supersecret
DATABASE_PASSWORD=iheartdatabases
RABBIT_PASSWORD=flopsymopsy
SERVICE_PASSWORD=iheartksl
```

5. browse

http://192.168.1.201/ for the dashboard (aka Horizon)

# OpenStack API

- Use OpenStack APIs to
  - launch server instances,
  - create images,
  - assign metadata to instances and images,
  - create storage containers and objects, and
  - complete other actions in your OpenStack cloud
- You can launch instances from images and assign metadata to instances through the Compute API or the openstack command-line client.

# Sending API requests

- OpenStack command-line client
  - The OpenStack project provides a command-line client that enables you to access APIs through easy-to-use commands.
- cURL
  - A command-line tool that lets you send HTTP requests and receive responses. See the section called OpenStack APIs.
- REST clients
  - Both Mozilla and Google provide browser-based graphical interfaces for REST.
- OpenStack Python Software Development Kit (SDK)
  - Use this SDK to write Python automation scripts that create and manage resources in your OpenStack cloud.

# OpenStack clients

- OpenStack provides **command line clients** which allow you to manage resources.
- Command line client called ***openstack*** can be used for all resource management tasks.
- You may install the clients on any computer, e.g. on your VM or your local computer (Win/Mac/Linux).

- There are also **application programming interfaces (APIs)** available for Python, C++, Java and more.
  - a list of known software development kits refer to https://wiki.openstack.org/wiki/SDKs

# OpenStack clients

**Exercise 1**: Install the ***openstack*** command line client on your computer.

Follow instructions in the **On-Line Documentation!** Summary:

<table>
<tr>
<td rowspan="2"><b><i>Windows</i></b><br>Install Python incl. <i>pip</i> from www.python.org<br>Install <i>setuptools</i> (see docs).<br>Open windows command line:<br><code>$ pip install pyOpenSSL</code><br><code>$ set PATH=%PATH%;</code><br><br><code>C:\Python27\Scripts</code><br><code>$ pip install python-openstackclient</code></td>
<td><b><i>Ubuntu Linux:</i></b><br><code>$ sudo apt-get install python-openstackclient</code></td>
</tr>
<tr>
<td><b><i>Mac OS X:</i></b><br><code>$ brew install python</code> <i>or</i> install  from www.python.org<br>Install <i>setuptools</i> (see documentatiaon).<br>Upgrade <i>setuptools</i> and install clients:<br><code>$ sudo pip install --upgrade setuptools</code><br><code>$ sudo pip install python-openstackclient</code></td>
</tr>
</table>

# OpenStack credentials

- The ***openstack*** command line client is now installed on your computer.
- Before you can use it, you need to ***load your credentials***, so the client can connect to your account.
- Where to get your credentials?

# OpenStack clients

**Exercise 2:** Get your OpenStack credentials.

- Go to *Dashboard → Compute → Access & Security → API Access.*
- Download your **OpenStack RC file** (button top right).
- You will also need your **OpenStack password**.
  - This is **not** the same password you use to log onto the Dashboard!
  - You need to *reset* your password to activate it.
    - Click next to your user name (your e-mail) on the top right and select *Settings*.
    - Click “Reset password” and copy&paste the password, save it as text file somewhere safe.

# OpenStack clients

**Exercise 3:** ***Load*** your OpenStack credentials.

***Windows:***

Change your OpenStack RC file to openrc.ps1:

```
$env:OS_AUTH_URL=
  "https://keystone.rc.nectar.org.au:5000/v2.0/"
$env:OS_TENANT_ID="f12d34....c"
$env:OS_TENANT_NAME=
   "<your-tentant-name>"
$env:OS_USERNAME="<your-email>"
$env:OS_PASSWORD=”<OpenStack-Passwd>";
$env:OS_REGION_NAME=”<Region-Name>”
```

Open PowerShell from Windows Command line:

```
$ powershell.exe
```

Load the credentials:

```
$ C:\<Path-to-OpenRC>\openrc.ps1
```

***Linux / Mac OSX:***

Load your credentials:

```
$ source
     <path-to-openrc.sh>
```

# OpenStack clients

- You can now use the *openstack* command line client.
- Every time you open a new terminal to use *openstack*, you have to load your credentials again (*“source”* your OpenStack RC script file)!
- The client is structured into several “**tools**” for various tasks.

# OpenStack command help

- To get help on the client, type:

```
$ openstack help
```

  - This will print a list of all the “tools”.
  - To print help on a tool:

```
$ openstack help <tool-name>
```

  - For example for the server tool:

```
$ openstack help server
```

# Accessing the Object Store

**Exercise 4**: List objects and create container

Read the help:

```
$ openstack help object
$ openstack help container
```

List your containers:

```
$ openstack container list
```

Create a container called *MyTestContainer*:

```
$ openstack container create MyTestContainer
```

List files in the container (still empty):

```
$ openstack object list MyTestContainer
```

# Accessing the Object Store

**Exercise 5**: Upload / Download files

Create a new text file *MyTestFile.txt* on your computer and upload it:

```
$ cd <folder-containing-MyNewTextFile.txt>
$ openstack object create
    MyTestContainer MyNewTextFile.txt
```

List the file in the container:

```
$ openstack object list MyTestContainer
```

Download file again and save as *MyDownloadedFile.txt*:

```
$ openstack object save --file MyDownloadedFile.txt
    MyTestContainer MyNewTextFile.txt
```

# Controlling an instance

**Exercise 6**: Launching an instance.

Read the help:

```
$ openstack help server
$ openstack help server create
```

Get the ID of the NeCTAR Ubuntu image you would like to launch:

```
$ openstack image list | grep NeCTAR
```

Launch an instance called *ClientLaunchedInstance*:

```
$ openstack server create --flavor m1.small
    --image <image-id> --key-name Nectar_Key
    --security-group icmp --security-group ssh
     ClientLaunchedInstance
```

List your instances:

```
$ openstack server list
```

# Controlling an instance

**Exercise 7**: Create a snapshot of the instance.

Create a snapshot called *ClientLaunchedSnapshot*:

```
$ openstack server image create
    --name ClientLaunchedSnapshot
    ClientLaunchedInstance
```

Show details of the snapshot:

```
$ openstack image show ClientLaunchedSnapshot
```

# Controlling an instance

**Exercise 8**: Launch a new instance from the snapshot.

List your private images (incl. snapshots):

```
$ openstack image list --private
```

Launch a new instance:

```
$ openstack server create --flavor m1.small
    --image ClientLaunchedSnapshot --key-name
Nectar_Key
    --security-group icmp --security-group ssh
    CopyOfClientLaunchedInstance
```

Show details of the new instance:

```
$ openstack server show
CopyOfClientLaunchedInstance
```

# Managing Volumes

- Creating and deleting volumes
- Attaching / detaching volumes to an instance.
- Make a *“backup”* of a volume
  - *Backup* vs. *Snapshot* was discussed in Module 9.
- Restore a volume from a backup.
- Create a snapshot of a volume
- Create a new volume of a snapshot

# Managing volumes

**Exercise 9:** Create a new volume (only users with allocation)

Read the help:

```
$ openstack help volume
$ openstack help volume create
```

List availability zones:

```
$ openstack availability zone list
```

Create a new volume called *MyNewStorage*:

```
$ openstack volume create
   --description "Description of the volume"
    --availability-zone <your zone name>
    --size 1 MyNewStorage
```

List all volumes:

```
$ openstack volume list
```

# Managing volumes

**Exercise 10:** Attach a volume (only users with allocation)

Read the help:

```
$ openstack server help | grep volume
```

Attach to your instance *ClientLaunchedInstance*:

```
$ openstack server add volume
    ClientLaunchedInstance MyNewStorage
```

List the volumes:

```
$ openstack volume list
```

Detach the volume:

```
$ openstack server remove volume
    ClientLaunchedInstance MyNewStorage
```

# Managing Volumes

**Exercise 11:** Backup a volume

Read the help:

```
$ openstack help backup
$ openstack help backup create
```

Create a backup of your volume *MyNewStorage*:

```
$ openstack backup create --container Backups
    --name Backup1 --description "Backup
MyNewStorage"
    MyNewStorage
```

List your backup files:

```
$ openstack backup list
```

Display your backup file in the object store:

```
$ openstack container list
$ openstack object list Backups
```

# Managing Volumes

**Exercise 12:** Restore from a *backup* and delete the *backup*.

Read the help:

```
$ openstack help backup restore
```

Get the ID of your backup:

```
$ openstack backup list
```

Restore the backup onto your volume *MyNewStorage*:

```
$ openstack backup restore <Backup-ID>
MyNewStorage
```

Delete the backup file from the Object Store:

```
$ openstack backup delete <Backup-ID>
```

# Managing Volumes

**Exercise 13:** Create a snapshot of a volume.

Make sure the volume is detached (status “available”):

```
$ openstack volume list
```

Create a snapshot of the new Volume *MyNewStorage*:

```
$ openstack snapshot create

    --name MyNewStorageSnapshot1
    --description "First snapshot" MyNewStorage
```

## Managing Volumes

**Exercise 14:** Create a new volume of the snapshot.

List the snapshots and copy the snapshot's ID:

```
$ openstack snapshot list
```

Create a new volume called *MyRestoredVolume* of the snapshot:

```
$ openstack volume create
     --snapshot <ID of MyNewStorageSnapshot1>
     --description "My restored Volume"
     --size 2 MyRestoredVolume
```

List your volumes to see the new one:

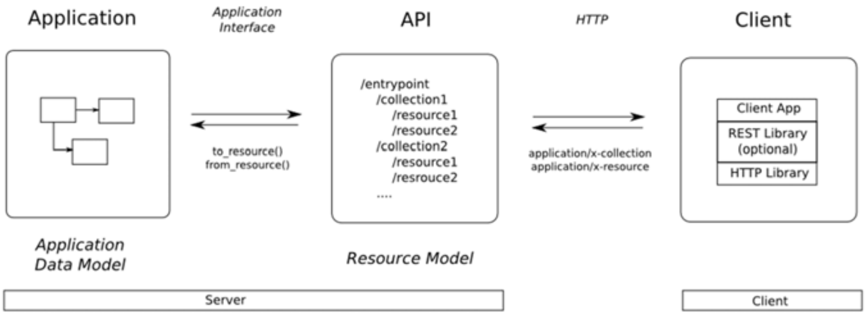
```
$ openstack volume list
```

To delete your snapshot:

```
$ openstack snapshot delete MyNewStorageSnapshot1
```

# REST

- ❑ REST (REpresentational State Transfer) is an architectural style, and an approach to communications that is often used in the development of web services
- ❑ REST is a lightweight alternative to Web Services and RPC.
  - – REST is often preferred over the more heavyweight SOAP (Simple Object Access Protocol) style
- ❑ REST does not leverage as much bandwidth, which makes it a better fit for use over the Internet



# Resources

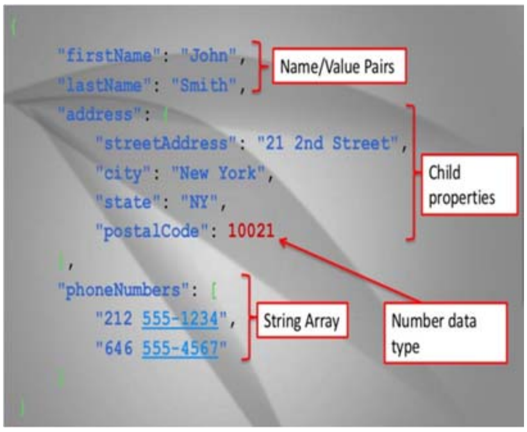
- The fundamental concept in any RESTful API is the resource.
- Resources model objects from the application data model.
- These resources can be pictures, video files, Web pages, business information, etc.
- A resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it.
- Each resource has a unique URL

# Addressing Resources

- A RESTful service uses a directory hierarchy like human readable URIs to address its resources.
- The job of a URI is to identify a resource or a collection of resources.
- The actual operation is determined by an HTTP verb.
- The URI should not say anything about the operation or action. This enables us to call the same URI with different HTTP verbs to perform different operations.
  - Bad: http://api.company.com/DeletePerson?id=1
- Example resource: http://jsonplaceholder.typicode.com/

# Resource Data

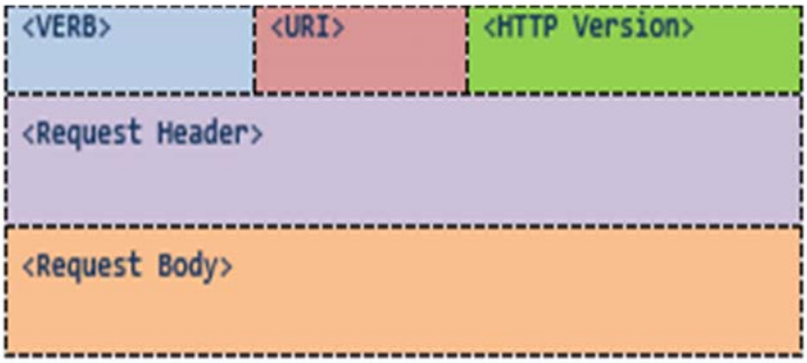
- ❑ Resources have data associated with them.
- ❑ In JSON, just three types of data exist:
  - – scalar (number, string, boolean, null).
  - – array
  - – object
- ❑ Data associated with a resource is modeled as key:value pairs on the JSON object.



```
{
    "firstName": "John",
    "lastName": "Smith",
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        "212 555-1234",
        "646 555-4567"
    ]
}
```

# HTTP Messages

- ❑ The client and service talk to each other via messages.
- ❑ Clients send a request to the server, and the server replies with a response.
- ❑ Apart from the actual data, these messages also contain some metadata about the message.
- ❑ HTTP Request:



<VERB> <URI> <HTTP Version>

<Request Header>

<Request Body>

# Request Message Example

```
POST http://MyService.com/Person/
Host: MyService.com
Content-Type: text/xml; charset=utf-8
Content-Length: 123
<?xml version="1.0" encoding="utf-8"?>
<Person>
  <Name>Larry</Name>
  <Email>larry@gmail.com</Email>
  <Country>US</Country>
</Person>
```

# Response Message Example

```
HTTP/1.1 200 OK
Date: Sat, 23 Aug 2014 18:31:04 GMT
Server: Apache/2
Last-Modified: Wed, 01 Sep 2004 13:24:52 GMT
Accept-Ranges: bytes
Content-Length: 32859
Cache-Control: max-age=21600, must-revalidate
Expires: Sun, 24 Aug 2014 00:31:04 GMT
Content-Type: text/html; charset=iso-8859-1
<html>
<head><title>CS449 Calendar</title></head>
<body>
...
```

# HTTP Methods

- Methods are verbs or actions that can be performed on resources
- Methods can be executed on resources via their URL.
- Standard methods that have a well-defined meaning for all resources and collections:

| Method | Scope | Semantics | Quality |
|---|---|---|---|
| GET | Collection | Retrieve all resources in a collection | Safe |
| GET | Resource | Retrieve a single resource | Safe |
| POST | Collection | Create a new resource in a collection | N/A |
| PUT | Resource | Update a resource | Idempotent |
| DELETE | Resource | Delete a resource | Idempotent |
| HEAD | Resource | Retrieve only the response headers | Safe |
| OPTIONS | Resource | List the allowed operations on a resource. | Safe |

# Methods [cont.]

- GET is Safe. A **Safe** operation is an operation that does not have any effect on the original value of the resource.
- PUT and DELETE are Idempotent. An **Idempotent** operation is an operation that gives the same result no matter how many times you perform it. Note, if you are adding a resource with PUT you have to specify the unique ID of the resource.

# Difference between PUT and POST

- PUT is idempotent while POST is not.
- No matter how many times you send a PUT request, the results will be same.
- POST is not an idempotent method. Making a POST multiple times may result in multiple resources getting created on the server.
- With PUT, it is the client's job to choose a unique name or ID for the resource. With POST, the server decides. This is why POST is not idempotent.
- There is no difference between PUT and POST if the resource already exists

# Options

- The method OPTIONS is used to get a list of allowed operations on the resource. For example

Request:

```
  OPTIONS
http://api.business.com/Persons/1
HTTP/1.1
  HOST: api.business.com
```

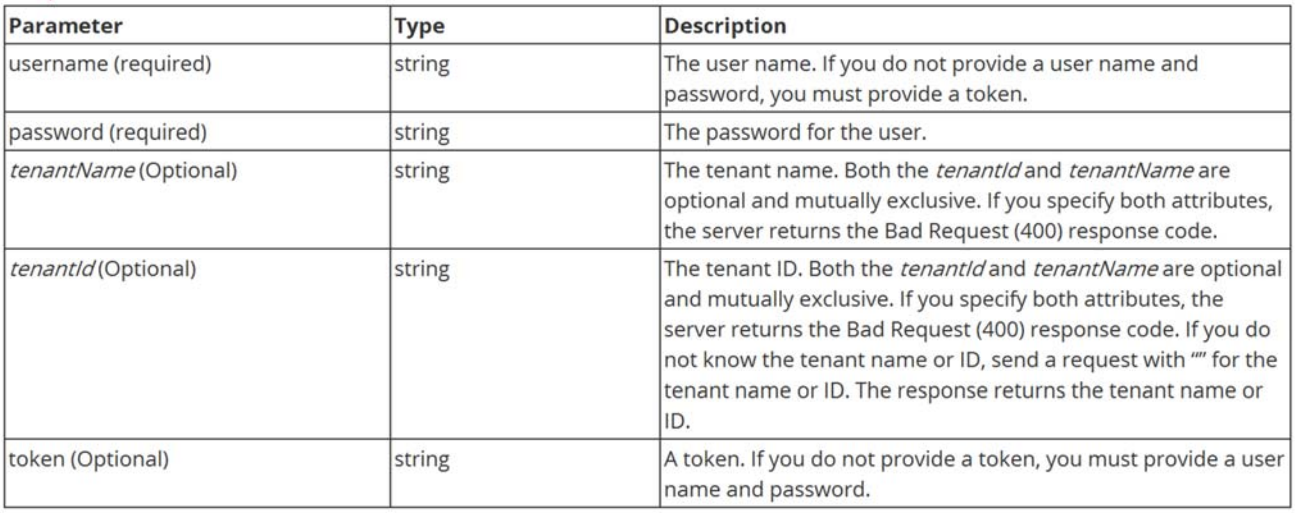
Response:

```
  200 OK
  Allow: HEAD, GET, PUT
```

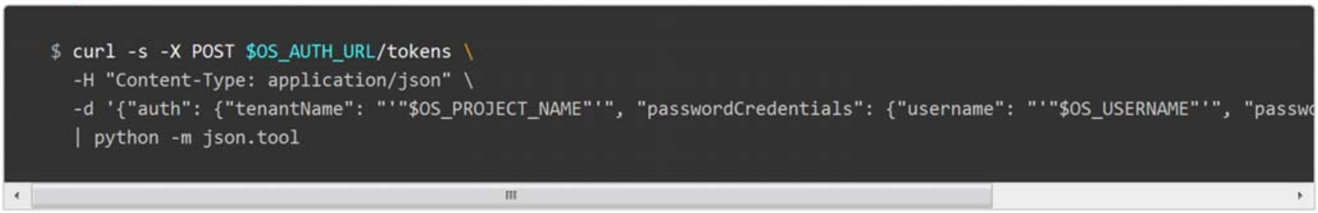
# Use cURL

- Open a terminal window:
  - Click on the terminal icon on the left menu or open a new tab on existing terminal window.
- cURL format:
  - curl --user <user>:<password> -H <header 1> -H <header-2> -X <requesttype> <url> -d '<request-body>'
- Get the topology with cURL:
  - curl --user "admin":"admin" -H "Accept: application/xml" –H "Content-type: application/xml" -X GET http://localhost:8181/restconf/operational/network-topology:network-topology/

# Authentication and API request



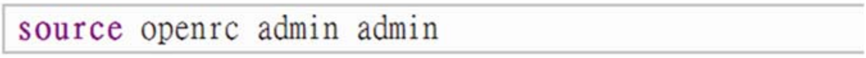
| Parameter | Type | Description |
|---|---|---|
| username (required) | string | The user name. If you do not provide a user name and password, you must provide a token. |
| password (required) | string | The password for the user. |
| *tenantName* (Optional) | string | The tenant name. Both the *tenantId* and *tenantName* are optional and mutually exclusive. If you specify both attributes, the server returns the Bad Request (400) response code. |
| *tenantId* (Optional) | string | The tenant ID. Both the *tenantId* and *tenantName* are optional and mutually exclusive. If you specify both attributes, the server returns the Bad Request (400) response code. If you do not know the tenant name or ID, send a request with “” for the tenant name or ID. The response returns the tenant name or ID. |
| token (Optional) | string | A token. If you do not provide a token, you must provide a user name and password. |



```
$ curl -s -X POST $OS_AUTH_URL/tokens \
  -H "Content-Type: application/json" \
  -d '{"auth": {"tenantName": "'"$OS_PROJECT_NAME"'", "passwordCredentials": {"username": "'"$OS_USERNAME"'", "passwo
  | python -m json.tool
```

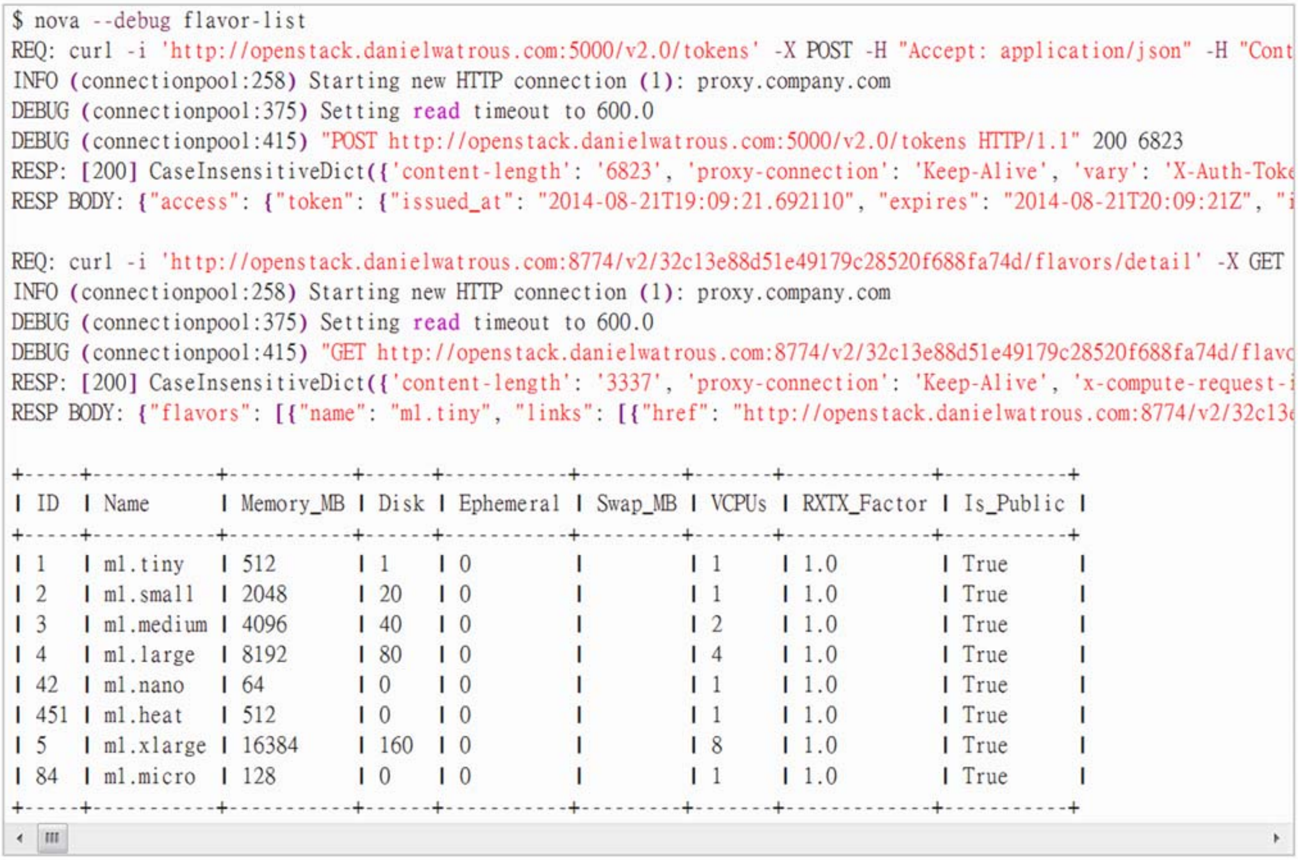
# Command Line Interfaces (CLI)

- Manage OpenStack components make use of the REST APIs behind the scenes
- Brings consistency to OpenStack management efforts and discourages disparity between standard tooling (CLI) and custom tooling (direct API access).
- Credentials are required to access the REST APIs.
- In your devstack

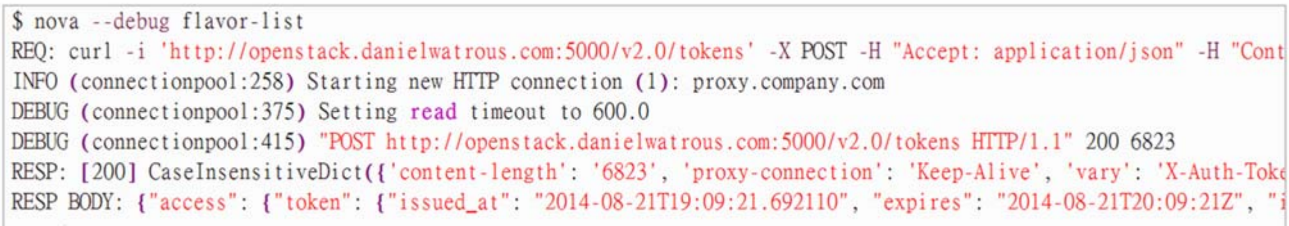


```
source openrc admin admin
```

- Some clients support a debug option
  - output full details about the request and response cycle.
  - Raw request and response details can be helpful when learning the APIs or
  - creating programmatic access libraries that wrap the APIs.

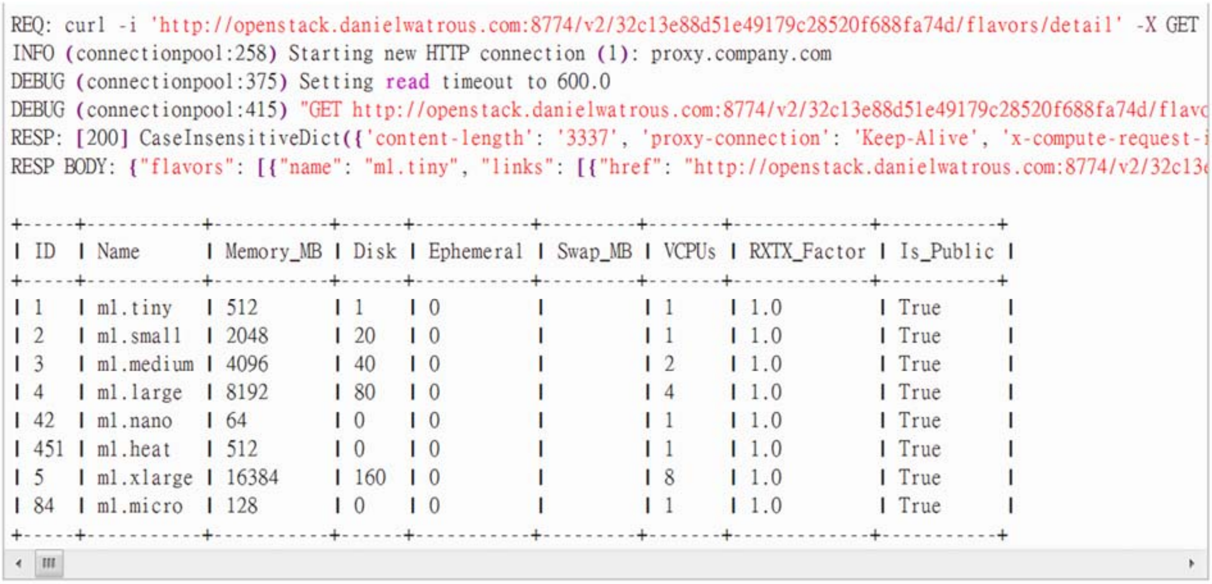


```
$ nova --debug flavor-list
REQ: curl -i 'http://openstack.danielwatrous.com:5000/v2.0/tokens' -X POST -H "Accept: application/json" -H "Cont
INFO (connectionpool:258) Starting new HTTP connection (1): proxy.company.com
DEBUG (connectionpool:375) Setting read timeout to 600.0
DEBUG (connectionpool:415) "POST http://openstack.danielwatrous.com:5000/v2.0/tokens HTTP/1.1" 200 6823
RESP: [200] CaseInsensitiveDict({'content-length': '6823', 'proxy-connection': 'Keep-Alive', 'vary': 'X-Auth-Toke
RESP BODY: {"access": {"token": {"issued_at": "2014-08-21T19:09:21.692110", "expires": "2014-08-21T20:09:21Z", "i

REQ: curl -i 'http://openstack.danielwatrous.com:8774/v2/32c13e88d51e49179c28520f688fa74d/flavors/detail' -X GET
INFO (connectionpool:258) Starting new HTTP connection (1): proxy.company.com
DEBUG (connectionpool:375) Setting read timeout to 600.0
DEBUG (connectionpool:415) "GET http://openstack.danielwatrous.com:8774/v2/32c13e88d51e49179c28520f688fa74d/flavo
RESP: [200] CaseInsensitiveDict({'content-length': '3337', 'proxy-connection': 'Keep-Alive', 'x-compute-request-i
RESP BODY: {"flavors": [{"name": "m1.tiny", "links": [{"href": "http://openstack.danielwatrous.com:8774/v2/32c13e

+-----+-----------+-----------+------+-----------+---------+-------+-------------+-----------+
| ID  | Name      | Memory_MB | Disk | Ephemeral | Swap_MB | VCPUs | RXTX_Factor | Is_Public |
+-----+-----------+-----------+------+-----------+---------+-------+-------------+-----------+
| 1   | m1.tiny   | 512       | 1    | 0         |         | 1     | 1.0         | True      |
| 2   | m1.small  | 2048      | 20   | 0         |         | 1     | 1.0         | True      |
| 3   | m1.medium | 4096      | 40   | 0         |         | 2     | 1.0         | True      |
| 4   | m1.large  | 8192      | 80   | 0         |         | 4     | 1.0         | True      |
| 42  | m1.nano   | 64        | 0    | 0         |         | 1     | 1.0         | True      |
| 451 | m1.heat   | 512       | 0    | 0         |         | 1     | 1.0         | True      |
| 5   | m1.xlarge | 16384     | 160  | 0         |         | 8     | 1.0         | True      |
| 84  | m1.micro  | 128       | 0    | 0         |         | 1     | 1.0         | True      |
+-----+-----------+-----------+------+-----------+---------+-------+-------------+-----------+
```



```
$ nova --debug flavor-list
REQ: curl -i 'http://openstack.danielwatrous.com:5000/v2.0/tokens' -X POST -H "Accept: application/json" -H "Cont
INFO (connectionpool:258) Starting new HTTP connection (1): proxy.company.com
DEBUG (connectionpool:375) Setting read timeout to 600.0
DEBUG (connectionpool:415) "POST http://openstack.danielwatrous.com:5000/v2.0/tokens HTTP/1.1" 200 6823
RESP: [200] CaseInsensitiveDict({'content-length': '6823', 'proxy-connection': 'Keep-Alive', 'vary': 'X-Auth-Toke
RESP BODY: {"access": {"token": {"issued_at": "2014-08-21T19:09:21.692110", "expires": "2014-08-21T20:09:21Z", "i
```

- ❑ The first two sections are calls the REST APIs,
  - first for the keystone service to Authenticate and receive a token.
  - Responses come as JSON due to the **Accept** header of **application/json**.
- ❑ The response actually included an access token and entry point URLs for each of the services that are integrated with keystone.

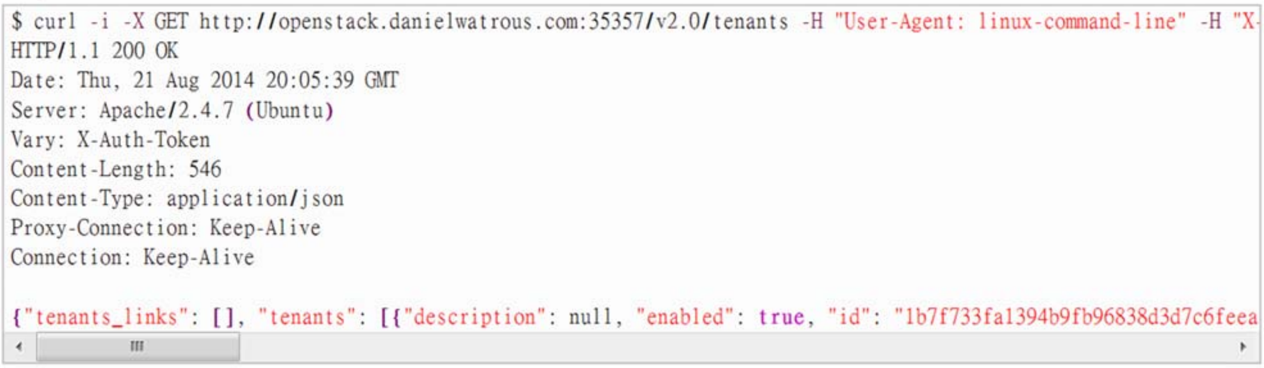


```
REQ: curl -i 'http://openstack.danielwatrous.com:8774/v2/32c13e88d51e49179c28520f688fa74d/flavors/detail' -X GET
INFO (connectionpool:258) Starting new HTTP connection (1): proxy.company.com
DEBUG (connectionpool:375) Setting read timeout to 600.0
DEBUG (connectionpool:415) "GET http://openstack.danielwatrous.com:8774/v2/32c13e88d51e49179c28520f688fa74d/flavo
RESP: [200] CaseInsensitiveDict({'content-length': '3337', 'proxy-connection': 'Keep-Alive', 'x-compute-request-i
RESP BODY: {"flavors": [{"name": "m1.tiny", "links": [{"href": "http://openstack.danielwatrous.com:8774/v2/32c13e

+-----+-----------+-----------+------+-----------+---------+-------+-------------+-----------+
| ID  | Name      | Memory_MB | Disk | Ephemeral | Swap_MB | VCPUs | RXTX_Factor | Is_Public |
+-----+-----------+-----------+------+-----------+---------+-------+-------------+-----------+
| 1   | m1.tiny   | 512       | 1    | 0         |         | 1     | 1.0         | True      |
| 2   | m1.small  | 2048      | 20   | 0         |         | 1     | 1.0         | True      |
| 3   | m1.medium | 4096      | 40   | 0         |         | 2     | 1.0         | True      |
| 4   | m1.large  | 8192      | 80   | 0         |         | 4     | 1.0         | True      |
| 42  | m1.nano   | 64        | 0    | 0         |         | 1     | 1.0         | True      |
| 451 | m1.heat   | 512       | 0    | 0         |         | 1     | 1.0         | True      |
| 5   | m1.xlarge | 16384     | 160  | 0         |         | 8     | 1.0         | True      |
| 84  | m1.micro  | 128       | 0    | 0         |         | 1     | 1.0         | True      |
+-----+-----------+-----------+------+-----------+---------+-------+-------------+-----------+
```

- ❑ The second section is the actual call to the nova API.
- ❑ It returns a list of eight flavors.
- ❑ The final section is a tabular view of the JSON response created by the nova command line client.

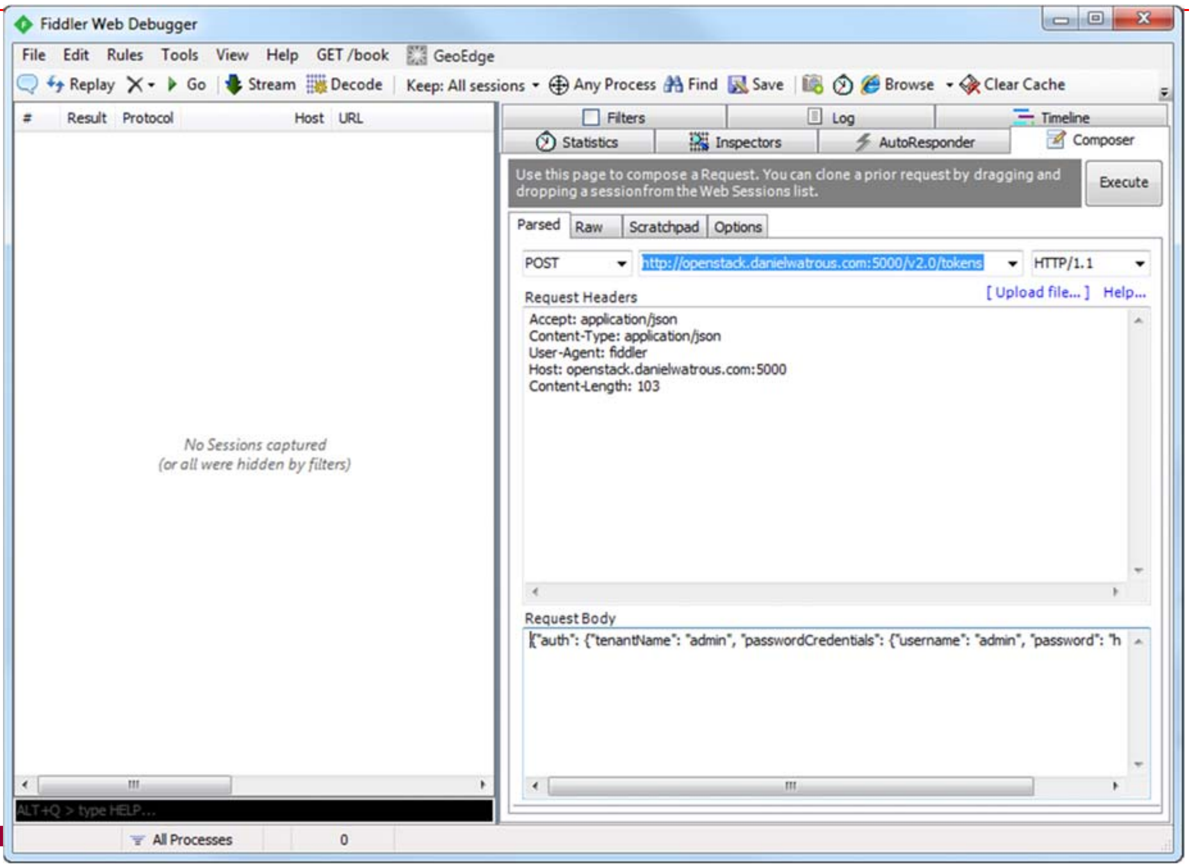
# Call keystone to get a list of tenants

```
$ curl -i -X GET http://openstack.danielwatrous.com:35357/v2.0/tenants
-H "User-Agent: linux-command-line" -H "X-Auth-Token: TOKEN"
HTTP/1.1 200 OK
```



```
$ curl -i -X GET http://openstack.danielwatrous.com:35357/v2.0/tenants -H "User-Agent: linux-command-line" -H "X-
HTTP/1.1 200 OK
Date: Thu, 21 Aug 2014 20:05:39 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: X-Auth-Token
Content-Length: 546
Content-Type: application/json
Proxy-Connection: Keep-Alive
Connection: Keep-Alive

{"tenants_links": [], "tenants": [{"description": null, "enabled": true, "id": "1b7f733fa1394b9fb96838d3d7c6feea
```

# Use Fiddler to create REST calls



Fiddler Web Debugger
File Edit Rules Tools View Help GET /book GeoEdge
Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache
# Result Protocol Host URL
Filters Log Timeline
Statistics Inspectors AutoResponder Composer
Execute
Parsed Raw Scratchpad Options
[ Upload file... ] Help...
ALT+Q > type HELP...
All Processes 0

No Sessions captured
(or all were hidden by filters)

Use this page to compose a Request. You can clone a prior request by dragging and dropping a session from the Web Sessions list.

POST http://openstack.danielwatrous.com:5000/v2.0/tokens HTTP/1.1

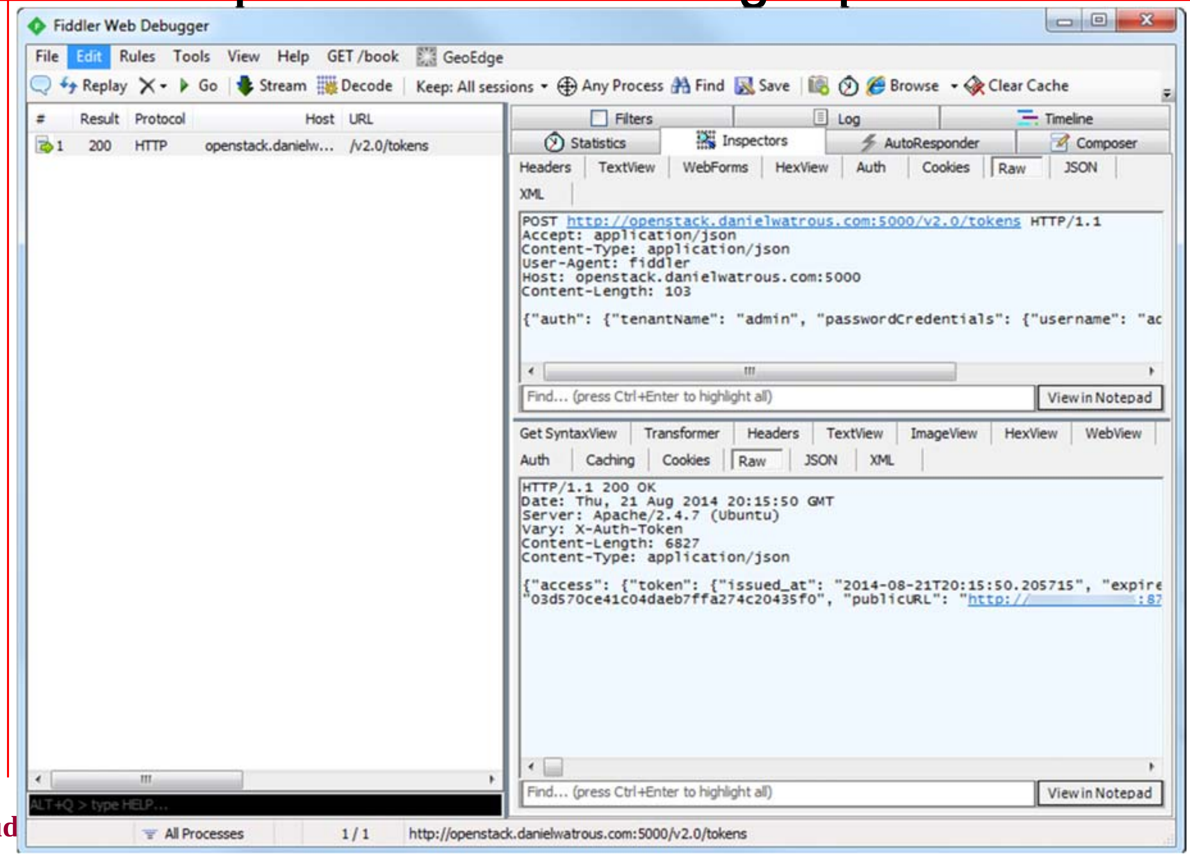
Request Headers

```
Accept: application/json
Content-Type: application/json
User-Agent: fiddler
Host: openstack.danielwatrous.com:5000
Content-Length: 103
```

Request Body

```
{"auth": {"tenantName": "admin", "passwordCredentials": {"username": "admin", "password": "h
```

# Response can be viewed in the left pane and Inspectors tab in the right pane.



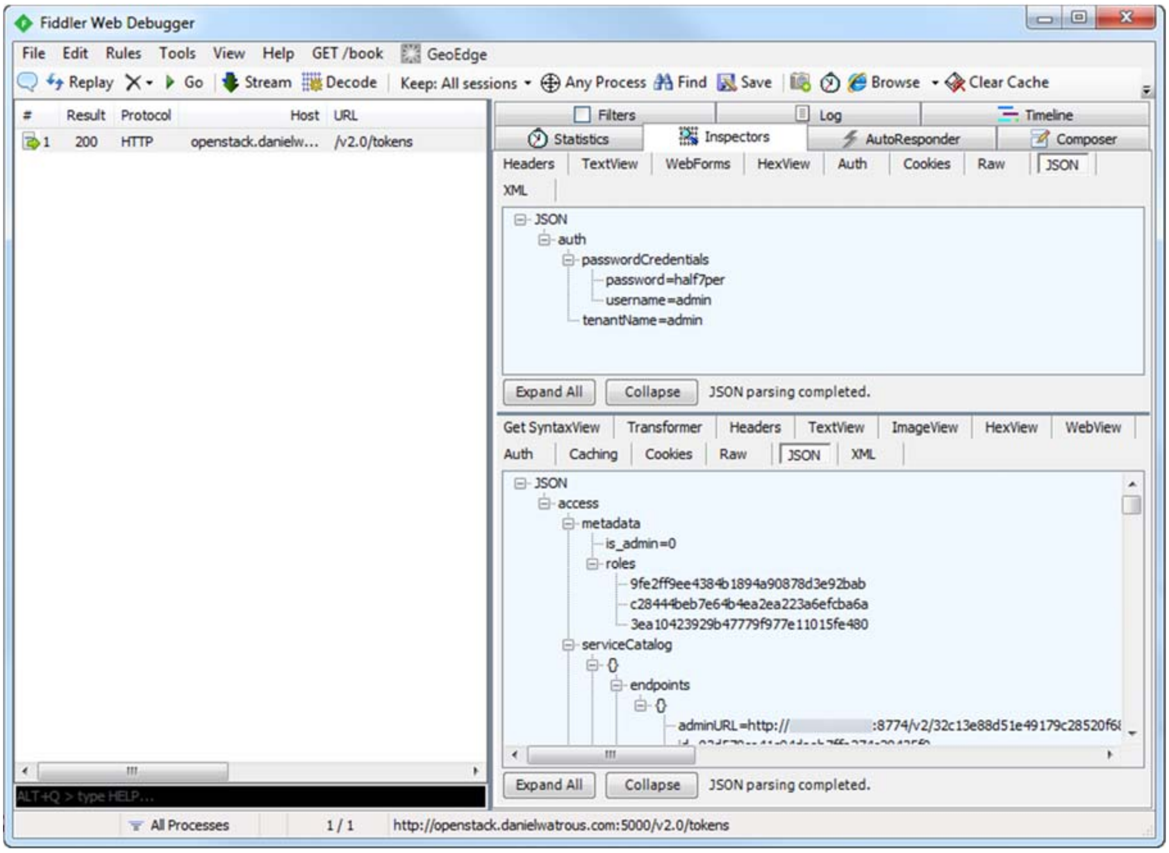
| # | Result | Protocol | Host | URL |
|---|---|---|---|---|
| 1 | 200 | HTTP | openstack.danielw... | /v2.0/tokens |

```
POST http://openstack.danielwatrous.com:5000/v2.0/tokens HTTP/1.1
Accept: application/json
Content-Type: application/json
User-Agent: fiddler
Host: openstack.danielwatrous.com:5000
Content-Length: 103

{"auth": {"tenantName": "admin", "passwordCredentials": {"username": "ac
```

```
HTTP/1.1 200 OK
Date: Thu, 21 Aug 2014 20:15:50 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: X-Auth-Token
Content-Length: 6827
Content-Type: application/json

{"access": {"token": {"issued_at": "2014-08-21T20:15:50.205715", "expire
"03d570ce41c04daeb7ffa274c20435f0", "publicURL": "http://[redacted]:87
```

http://openstack.danielwatrous.com:5000/v2.0/tokens

# Fiddler provides various JSON parsers



| # | Result | Protocol | Host | URL |
|---|---|---|---|---|
| 1 | 200 | HTTP | openstack.danielw... | /v2.0/tokens |

```
JSON
  auth
    passwordCredentials
      password=half7per
      username=admin
    tenantName=admin
```

JSON parsing completed.

```
JSON
  access
    metadata
      is_admin=0
      roles
        9fe2ff9ee4384b1894a90878d3e92bab
        c28444beb7e64b4ea2ea223a6efcba6a
        3ea10423929b47779f977e11015fe480
    serviceCatalog
      {}
        endpoints
          {}
            adminURL=http://[redacted]:8774/v2/32c13e88d51e49179c28520f6
```

JSON parsing completed.

http://openstack.danielwatrous.com:5000/v2.0/tokens