OpenClos – ReST API

Deepti Chandra

Table of Contents

Overview

OpenClos is an open-source (available on GitHub) set of Python scripts and libraries that facilitate setting up IP Fabric networks built on BGP. It addresses the problem of automating the creation of IP fabrics at a large scale. Many datacenter use cases require the IP Fabric architecture. OpenClos eases the provisioning process by taking in a set of input parameters and generating network switch configuration files and optionally a cabling plan.

OpenClos Use Cases

OpenClos can generate configuration files for devices used in a spine and leaf architecture using input parameters interface assignments, control plane and high availability values. It also supports Zero Touch Provisioning (ZTP) which allows for loading the precompiled configuration files on network devices, bringing the IP fabric up and ready to use. Adding new functionality is also relatively easier due to the use of templates that can be easily modified based on the added constraints. In addition to the initial provisioning of the IP fabric architectures, it can also help ease the process of making incremental configuration changes at a large scale in datacenter environments.

OpenClos Code Hierarchy

The user can use the ReST API or the Command Line Interface (CLI) to interact with OpenClos.

```
root@ubuntu:/home/deeptic/OpenClos-devR2.5/jnpr/openclos#
drwxrwxrwx  6 root  4096  Sep 18 12:58 conf    < --- config files and templates drwxrwxrwx
2 root  4096  Sep 18 14:23 data    < --- database file (if SQLLite used) drwxrwxrwx 12
root  4096  Sep 18 13:11 out     < --- contains generated config/other files drwxrwxrwx  2
root  4096  Sep  9 12:20 script  < --- used by Network Director
drwxrwxrwx  4 root  4096  Sep  9 12:20 tests   < --- unit test files
-rwxrwxrwx  1 root  31808 Sep  9 12:20 rest.py < --- source code for the ReST API
```

This document explores the usage of the ReST API and demonstrates the generation of configuration files and cabling plan used to bring up an IP fabric.

OpenClos Installation

The management server needs to have the appropriate versions of Python and PyEZ running prior to setting up OpenClos. The environment below was setup on a virtual machine running Ubuntu 14.04 LTS, to act like the management server.

Prerequisites:
****

<u>Python 2.7</u>

```
root@ubuntu:/home/deeptic#  python --version
Python 2.7.6
```

<u>PyEZ</u>

```
sudo apt-get install libxml2-dev libxslt-dev
sudo apt-get install python-setuptools
sudo easy_install-2.7 -U pip

sudo apt-get install git
root@ubuntu:/home/deeptic#  git --version
git version 1.9.1

sudo apt-get install python-dev
sudo pip install junos-eznc
```

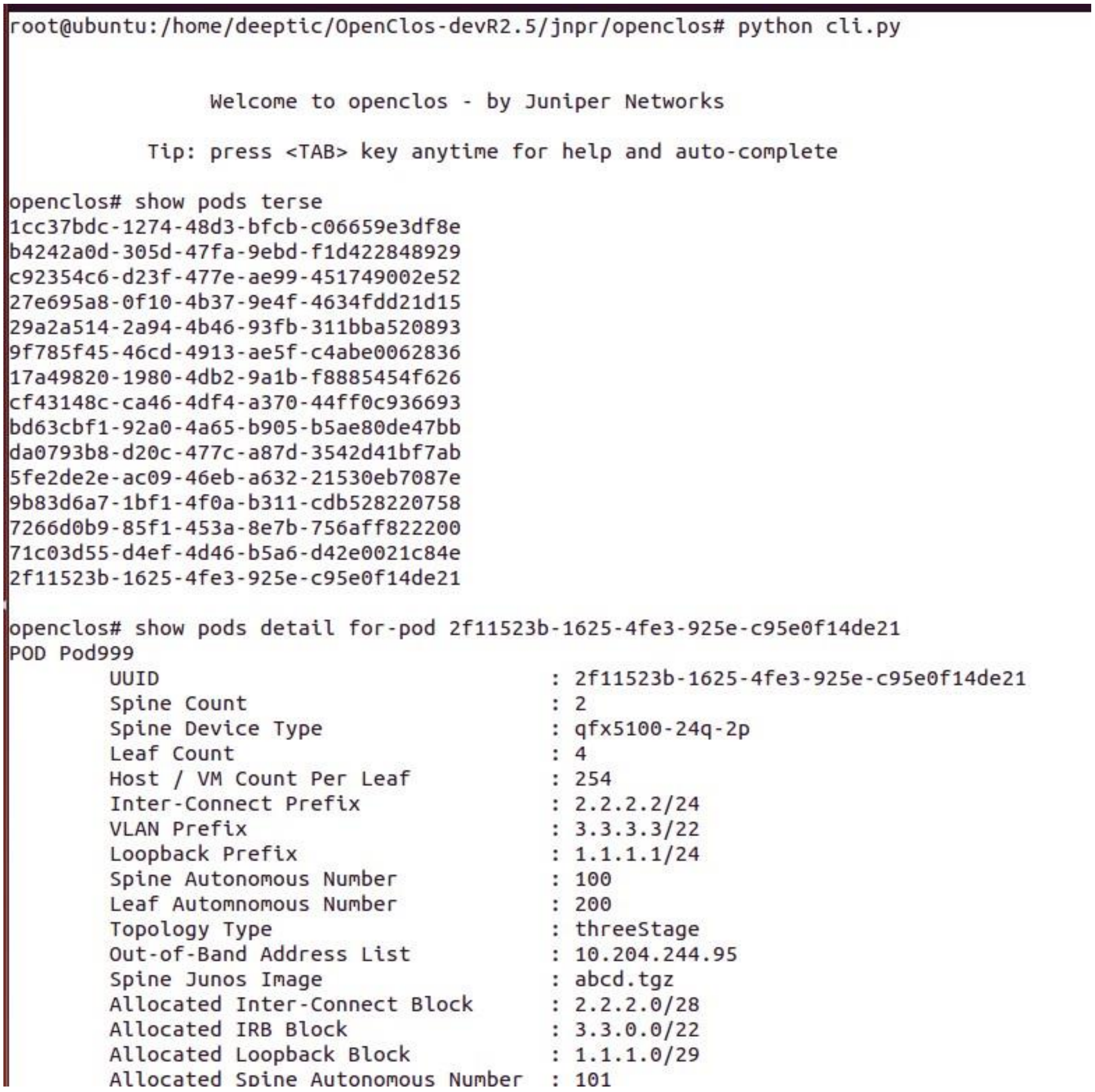
<u>OpenClos</u>

```
root@ubuntu:/home/deeptic#  curl -L -u user:password -o OpenClos.zip
https://github.com/Juniper/OpenClos/archive/devR2.5.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   122    0   122    0     0    277      0 --:--:-- --:--:-- --:--:--   277
  0     0    0  154k    0     0   131k      0 --:--:--  0:00:01 --:--:--  550k



root@ubuntu:/home/deeptic#  sudo pip install --egg OpenClos.zip

root@ubuntu:/home/deeptic#  unzip OpenClos.zip

root@ubuntu:/home/deeptic#  cd OpenClos-devR2.5/
root@ubuntu:/home/deeptic/OpenClos-devR2.5#  ls -lrt
total 76
-rw-r--r-- 1 root root  3748 Sep  9 12:20 setup.py
-rw-r--r-- 1 root root   219 Sep  9 12:20 requirements.txt
-rw-r--r-- 1 root root  7003 Sep  9 12:20 README.md
-rw-r--r-- 1 root root 33149 Sep  9 12:20 openClosLogo.jpeg
-rw-r--r-- 1 root root   453 Sep  9 12:20 MANIFEST.in
-rw-r--r-- 1 root root 10273 Sep  9 12:20 LICENSE
drwxr-xr-x 3 root root  4096 Sep  9 12:20 jnpr
-rw-r--r-- 1 root root  1483 Sep  9 12:20 COPYRIGHT
-rw-r--r-- 1 root root     0 Sep  9 12:20 AUTHORS
```

OpenClos User Interface

Users can interact with OpenClos using either the Command Line Interface (CLI) or ReSTful Application Programming Interface (ReST API). Both user interfaces provide the same functionality for provisioning and management of network devices.

Commands can be run from OpenClos CLI (cli.py) to create device configuration files or retrieve current POD information. Analogous operations are mapped to the ReST API as well.



```
root@ubuntu:/home/deeptic/OpenClos-devR2.5/jnpr/openclos# python cli.py


                Welcome to openclos - by Juniper Networks

           Tip: press <TAB> key anytime for help and auto-complete

openclos# show pods terse
1cc37bdc-1274-48d3-bfcb-c06659e3df8e
b4242a0d-305d-47fa-9ebd-f1d422848929
c92354c6-d23f-477e-ae99-451749002e52
27e695a8-0f10-4b37-9e4f-4634fdd21d15
29a2a514-2a94-4b46-93fb-311bba520893
9f785f45-46cd-4913-ae5f-c4abe0062836
17a49820-1980-4db2-9a1b-f8885454f626
cf43148c-ca46-4df4-a370-44ff0c936693
bd63cbf1-92a0-4a65-b905-b5ae80de47bb
da0793b8-d20c-477c-a87d-3542d41bf7ab
5fe2de2e-ac09-46eb-a632-21530eb7087e
9b83d6a7-1bf1-4f0a-b311-cdb528220758
7266d0b9-85f1-453a-8e7b-756aff822200
71c03d55-d4ef-4d46-b5a6-d42e0021c84e
2f11523b-1625-4fe3-925e-c95e0f14de21

openclos# show pods detail for-pod 2f11523b-1625-4fe3-925e-c95e0f14de21
POD Pod999
        UUID                                : 2f11523b-1625-4fe3-925e-c95e0f14de21
        Spine Count                         : 2
        Spine Device Type                   : qfx5100-24q-2p
        Leaf Count                          : 4
        Host / VM Count Per Leaf            : 254
        Inter-Connect Prefix                : 2.2.2.2/24
        VLAN Prefix                         : 3.3.3.3/22
        Loopback Prefix                     : 1.1.1.1/24
        Spine Autonomous Number             : 100
        Leaf Automnomous Number             : 200
        Topology Type                       : threeStage
        Out-of-Band Address List            : 10.204.244.95
        Spine Junos Image                   : abcd.tgz
        Allocated Inter-Connect Block       : 2.2.2.0/28
        Allocated IRB Block                 : 3.3.0.0/22
        Allocated Loopback Block            : 1.1.1.0/29
        Allocated Spine Autonomous Number   : 101
```

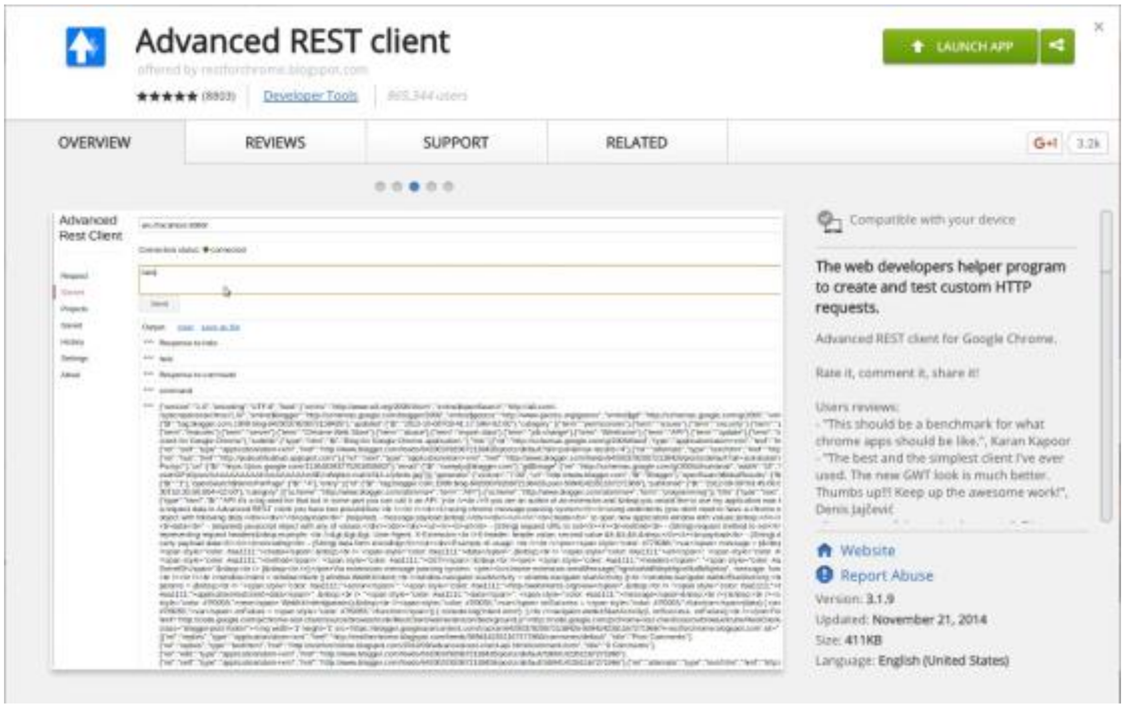
OpenClos ReST API Workflow

The ReST API allows interaction with OpenClos using HTTP requests. It creates a uniform interface by mapping HTTP methods like POST, GET, PUT, DELETE to CRUD – Create, Retrieve, Update and Delete. OpenClos (rest.py) uses bottle to create a ReSTful Python API.

```
# GET APIs
bottle.route('/', 'GET', self.getIndex)
bottle.route('/openclos', 'GET', self.getIndex)
bottle.route('/openclos/conf', 'GET', self.getOpenClosConfigParams)
bottle.route('/openclos/pods', 'GET', self.getPods)
bottle.route('/openclos/images/<junosImageName>', 'GET', self.getJunosImage)
bottle.route('/openclos/pods/<podId>', 'GET', self.getPod)
bottle.route('/openclos/pods/<podId>/cabling-plan', 'GET', self.getCablingPlan)
bottle.route('/openclos/pods/<podId>/ztp-configuration','GET', self.getZtpConfig)
bottle.route('/openclos/pods/<podId>/device-configuration', 'GET', self.getDeviceConfigsInZip)
bottle.route('/openclos/pods/<podId>/leaf-generic-configurations/<deviceModel>', 'GET', self.getLeafGenericConfiguration)
bottle.route('/openclos/pods/<podId>/l2-report', 'GET', self.getL2Report)
bottle.route('/openclos/pods/<podId>/l3-report', 'GET', self.getL3Report)
bottle.route('/openclos/pods/<podId>/devices', 'GET', self.getDevices)
bottle.route('/openclos/pods/<podId>/devices/<deviceId>', 'GET', self.getDevice)
bottle.route('/openclos/pods/<podId>/devices/<deviceId>/config', 'GET', self.getDeviceConfig)

# POST/PUT APIs
bottle.route('/openclos/pods', 'POST', self.createPod)
bottle.route('/openclos/pods/<podId>/cabling-plan', 'PUT', self.createCablingPlan)
bottle.route('/openclos/pods/<podId>/device-configuration', 'PUT', self.createDeviceConfiguration)
bottle.route('/openclos/pods/<podId>/ztp-configuration', 'PUT', self.createZtpConfiguration)
bottle.route('/openclos/pods/<podId>', 'PUT', self.reconfigPod)
bottle.route('/openclos/conf/', 'PUT', self.setOpenClosConfigParams)

# DELETE APIs
bottle.route('/openclos/pods/<podId>', 'DELETE', self.deletePod)
```

For the ReST environment, a server needs to run on the management server and the user needs a client to send requests interacting with the server.

The 'Advanced ReST client' available in the web store was used as a part of this demonstration. Alternately, any other ReST client can be used.



The web developers helper program to create and test custom HTTP requests.

Advanced REST client for Google Chrome.

Rate it, comment it, share it!

Users reviews:
- "This should be a benchmark for what chrome apps should be like.", Karan Kapoor
- "The best and the simplest client I've ever used. The new GWT look is much better. Thumbs up!!! Keep up the awesome work!", Denis Jajčević

Version: 3.1.9
Updated: November 21, 2014
Size: 411KB
Language: English (United States)

To set up the server, a few changes are required to be made to the OpenClos source code.

The openclos.yaml file located at /jnpr/openclos/conf defines global parameters that are used by some files. The server IP and port need to be specifed to have a ReST server up and listen to the client. The port that is used for the server should be available for use.

```
httpServer :

    ipAddr : 172.16.193.134

    port : 8080
```

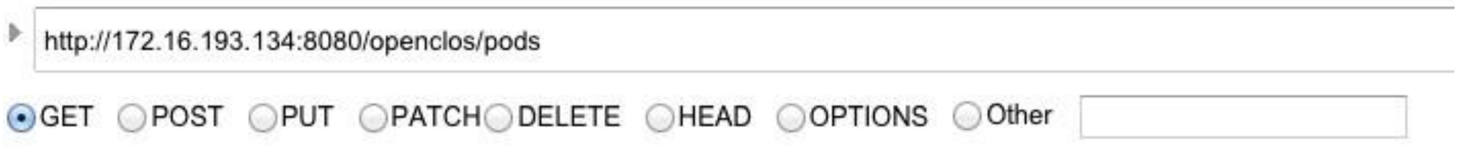
Based on the desired requirements, a few other changes can be made to the same yaml file.

Successful initialization will bring the ReST server up and also displays details of the message exchange.

```
2015-09-23 10:54:04,183 [rest] [INFO] [7fdb0a457740] RestServer initRest() done

2015-09-23 10:54:04,185 [rest] [INFO] [7fdb0a457740] REST server starting at
172.16.193.134:8080

Bottle v0.12.8 server starting up (using WSGIRefServer())...

Listening on http://172.16.193.134:8080/

Hit Ctrl-C to quit.
```

Proof of Concept – Configuration Generation

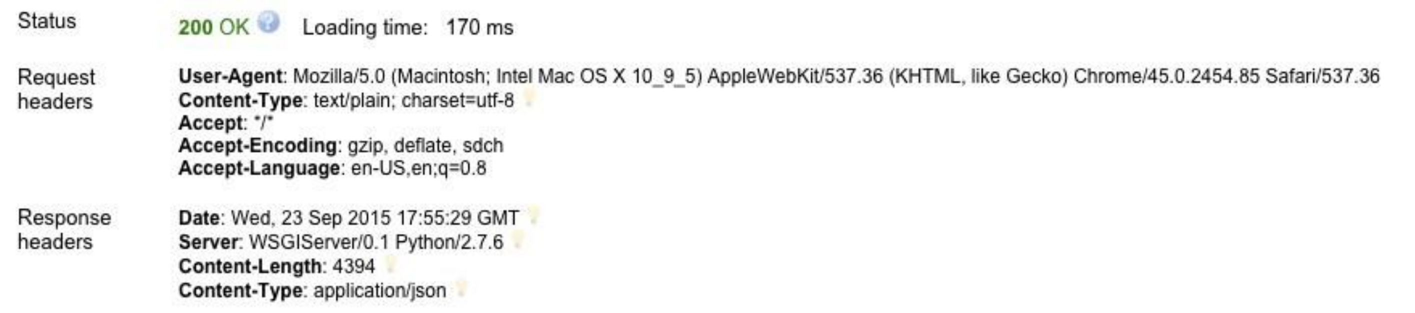
To retrieve the state of the current management server, the ReST client sends a GET request to see detailed information on the PODs and related device configuration files.



http://172.16.193.134:8080/openclos/pods

[✓ GET] POST PUT PATCH DELETE HEAD OPTIONS Other

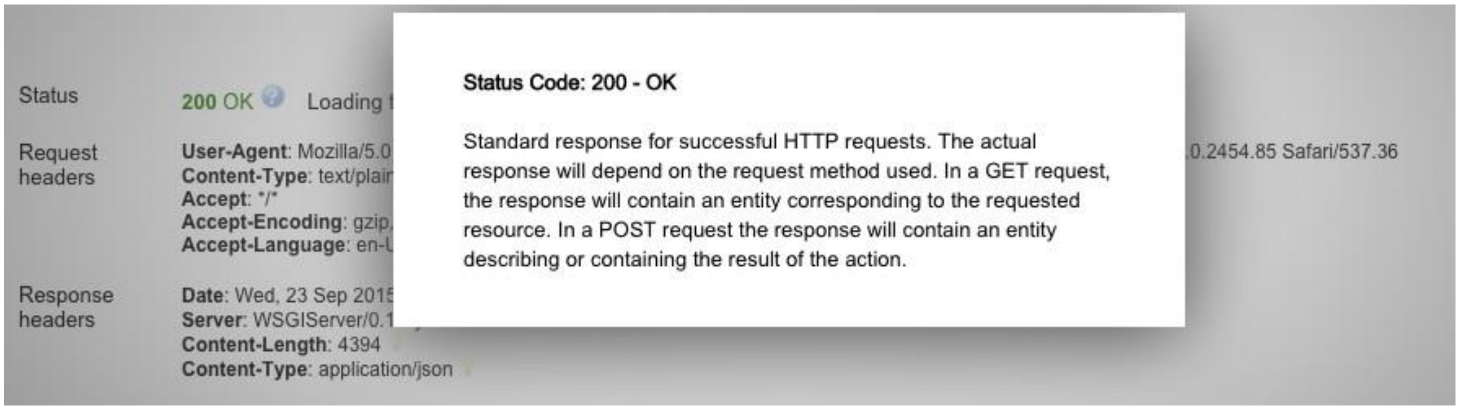
The ReST client displays detailed information about the exchange and the execution status of the request is tied to an HTTP response code.



Status 200 OK Loading time: 170 ms

Request headers
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.85 Safari/537.36
Content-Type: text/plain; charset=utf-8
Accept: */*
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8

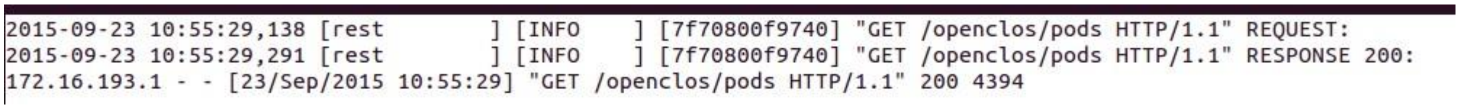
Response headers
Date: Wed, 23 Sep 2015 17:55:29 GMT
Server: WSGIServer/0.1 Python/2.7.6
Content-Length: 4394
Content-Type: application/json



Status Code: 200 - OK

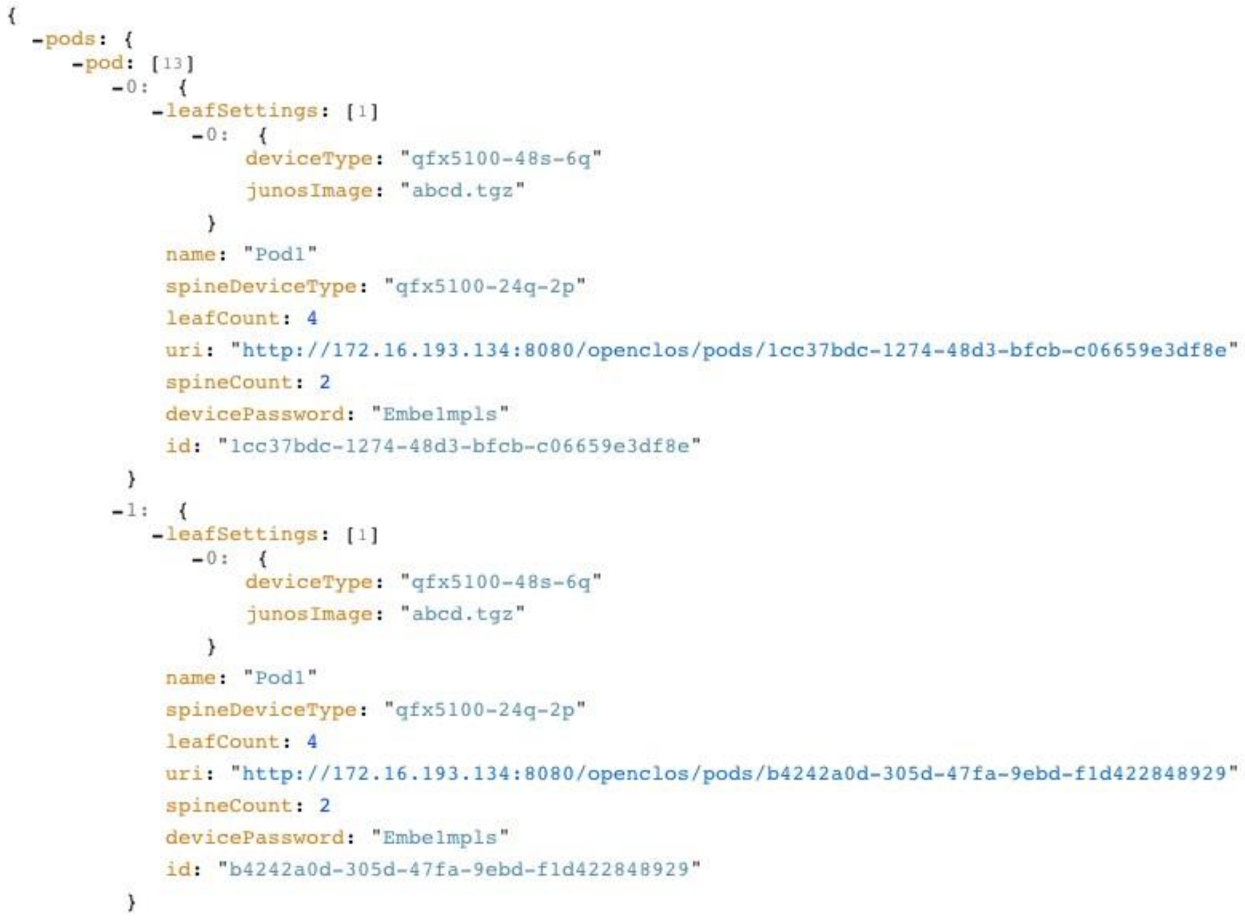
Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request the response will contain an entity describing or containing the result of the action.

Similar information can also be viewed at the ReST server on the management POD.



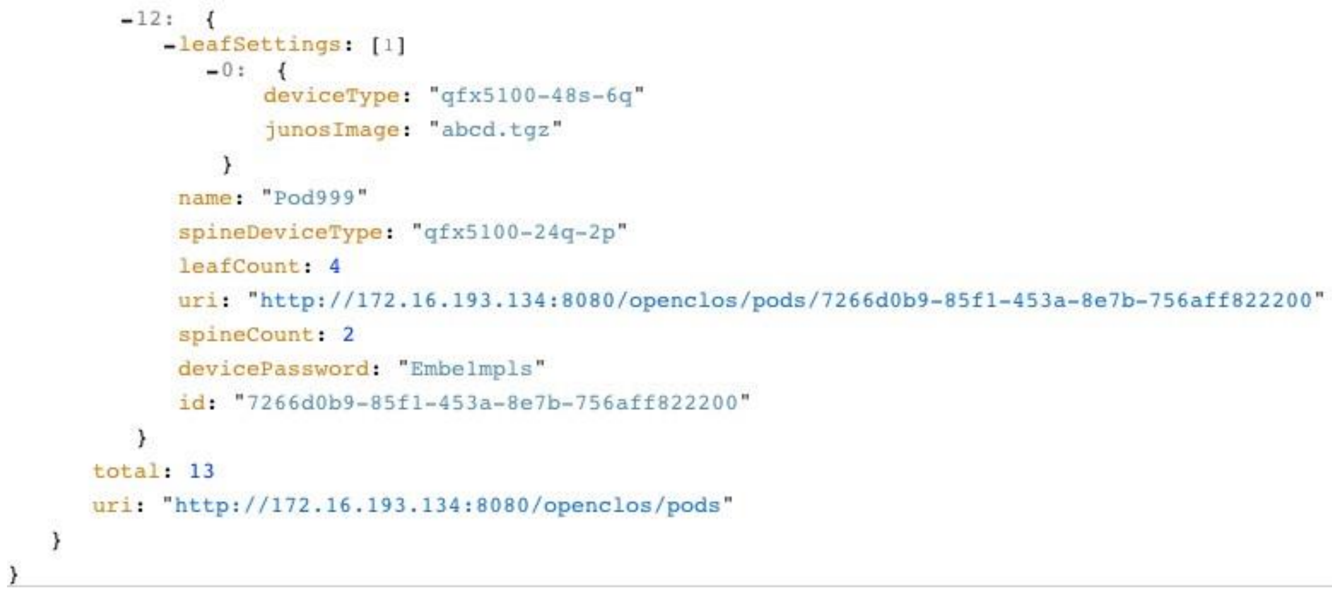
```
2015-09-23 10:55:29,138 [rest        ] [INFO    ] [7f70800f9740] "GET /openclos/pods HTTP/1.1" REQUEST:
2015-09-23 10:55:29,291 [rest        ] [INFO    ] [7f70800f9740] "GET /openclos/pods HTTP/1.1" RESPONSE 200:
172.16.193.1 - - [23/Sep/2015 10:55:29] "GET /openclos/pods HTTP/1.1" 200 4394
```

The output of the GET request displays the state of available PODs and their attributes. Each POD is assigned a unique ID that is used to identify a POD in every transaction.



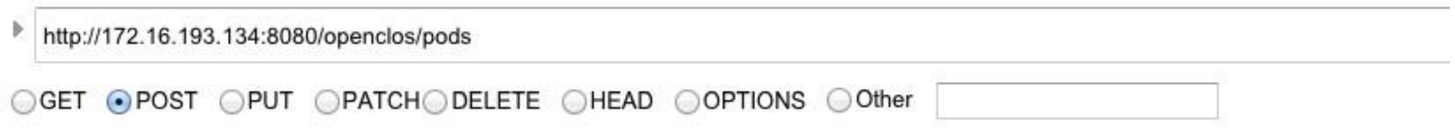
```
{
  -pods: {
     -pod: [13]
        -0:  {
           -leafSettings: [1]
              -0:  {
                   deviceType: "qfx5100-48s-6q"
                   junosImage: "abcd.tgz"
                }
            name: "Pod1"
            spineDeviceType: "qfx5100-24q-2p"
            leafCount: 4
            uri: "http://172.16.193.134:8080/openclos/pods/1cc37bdc-1274-48d3-bfcb-c06659e3df8e"
            spineCount: 2
            devicePassword: "Embe1mpls"
            id: "1cc37bdc-1274-48d3-bfcb-c06659e3df8e"
         }
        -1:  {
           -leafSettings: [1]
              -0:  {
                   deviceType: "qfx5100-48s-6q"
                   junosImage: "abcd.tgz"
                }
            name: "Pod1"
            spineDeviceType: "qfx5100-24q-2p"
            leafCount: 4
            uri: "http://172.16.193.134:8080/openclos/pods/b4242a0d-305d-47fa-9ebd-f1d422848929"
            spineCount: 2
            devicePassword: "Embe1mpls"
            id: "b4242a0d-305d-47fa-9ebd-f1d422848929"
         }
```

```
<snip>
```



```
        -12:  {
           -leafSettings: [1]
              -0:  {
                   deviceType: "qfx5100-48s-6q"
                   junosImage: "abcd.tgz"
                }
            name: "Pod999"
            spineDeviceType: "qfx5100-24q-2p"
            leafCount: 4
            uri: "http://172.16.193.134:8080/openclos/pods/7266d0b9-85f1-453a-8e7b-756aff822200"
            spineCount: 2
            devicePassword: "Embe1mpls"
            id: "7266d0b9-85f1-453a-8e7b-756aff822200"
         }
      total: 13
      uri: "http://172.16.193.134:8080/openclos/pods"
   }
}
```

To create a new POD, a POST request is used with valid parameters set in the input JSON file.

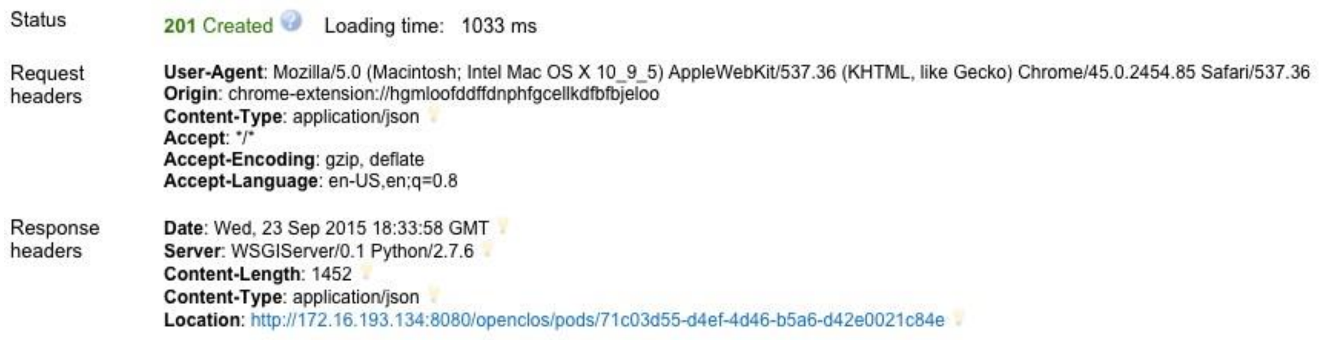

http://172.16.193.134:8080/openclos/pods

GET [✓ POST] PUT PATCH DELETE HEAD OPTIONS Other

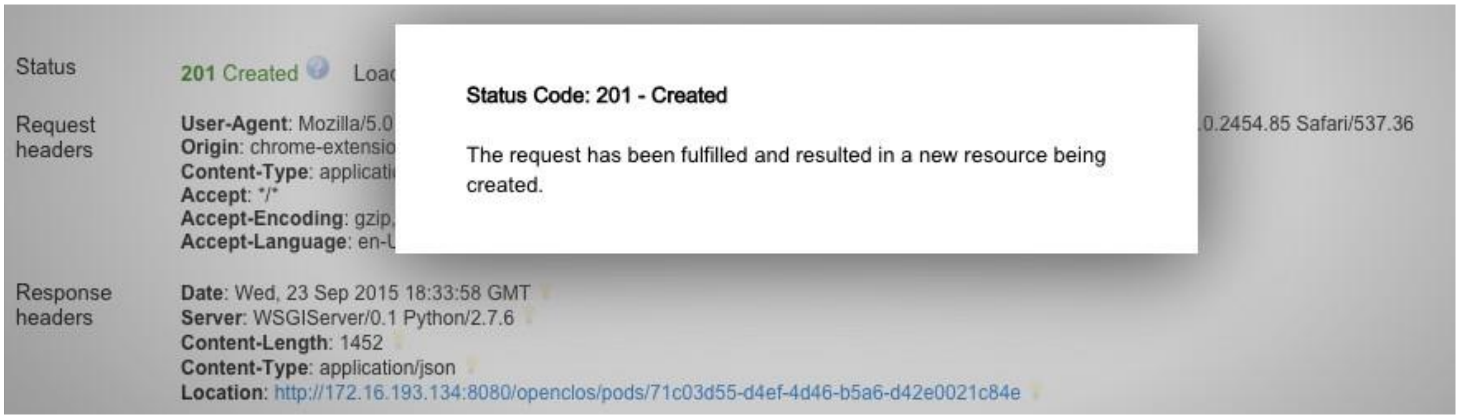
The attributes set in the JSON file are used as initialization for configuration generation.
The Pod999 below contains 2 spine devices with 4 leaf devices. The Junos version, IP addresses, various control plane variables and access related information is initialized through this JSON file, which is used as payload for the POST request.

```
{
  "pod": {
    "name": "Pod999",
    "spineDeviceType": "qfx5100-24q-2p",
    "spineCount": 2,
    "spineAS": 100,
    "spineJunosImage": "abcd.tgz",
    "leafSettings": [{"deviceType":"qfx5100-48s-6q", "junosImage": "abcd.tgz"}],
    "leafCount": 4,
    "leafAS": 200,
    "topologyType": "threeStage",
    "loopbackPrefix": "1.1.1.1/24",
    "vlanPrefix": "3.3.3.3/22",
    "interConnectPrefix": "2.2.2.2/24",
    "outOfBandAddressList": "10.204.244.95",
    "outOfBandGateway": "10.204.244.254",
    "managementPrefix": "4.4.4.0/24",
    "description": "test POD",
    "vcCreateLag": false,
    "hostOrVmCountPerLeaf": 254,
    "devicePassword": "Embe1mpls",
    "devices": [
       {"name": "test12321-spine-0", "family": "qfx5100-24q-2p", "username": "root", "password":
       "Embe1mpls", "serialNumber": "1234567", "role": "spine", "deployStatus": "deploy"},
       {"name": "test12321-spine-1", "family": "qfx5100-24q-2p", "serialNumber": "JNPR-1234", "role":
       "spine", "deployStatus": "deploy"},
       {"name": "test12321-leaf-0", "family": "qfx5100-48s-6q", "serialNumber": "JNPR-3456", "role":
       "leaf", "deployStatus": "deploy"},
       {"name": "test12321-leaf-1", "family": "qfx5100-48t-6q", "role": "leaf", "deployStatus":
       "deploy"},
       {"name": "test12321-leaf-2", "family": "qfx5100-48t-6q", "role": "leaf", "deployStatus":
       "deploy"},
       {"name": "test12321-leaf-3", "family": "qfx5100-48t-6q", "role": "leaf", "deployStatus":
       "deploy"}
    ]
  }
}
```

Successful execution of the POST request is reflected by the corresponding response codes at both the client and the server.



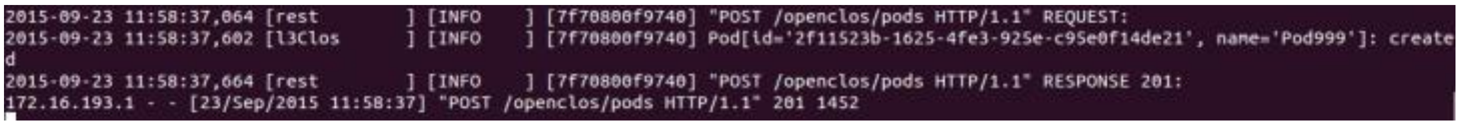
Status	**201** Created Loading time: 1033 ms
Request headers	**User-Agent**: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.85 Safari/537.36 **Origin**: chrome-extension://hgmloofddffdnphfgcellkdfbfbjeloo **Content-Type**: application/json **Accept**: */* **Accept-Encoding**: gzip, deflate **Accept-Language**: en-US,en;q=0.8
Response headers	**Date**: Wed, 23 Sep 2015 18:33:58 GMT **Server**: WSGIServer/0.1 Python/2.7.6 **Content-Length**: 1452 **Content-Type**: application/json **Location**: http://172.16.193.134:8080/openclos/pods/71c03d55-d4ef-4d46-b5a6-d42e0021c84e



Status	**201** Created Loa
Request headers	**User-Agent**: Mozilla/5.0 … .0.2454.85 Safari/537.36 **Origin**: chrome-extensio **Content-Type**: applicati **Accept**: */* **Accept-Encoding**: gzip, **Accept-Language**: en-U
Response headers	**Date**: Wed, 23 Sep 2015 18:33:58 GMT **Server**: WSGIServer/0.1 Python/2.7.6 **Content-Length**: 1452 **Content-Type**: application/json **Location**: http://172.16.193.134:8080/openclos/pods/71c03d55-d4ef-4d46-b5a6-d42e0021c84e

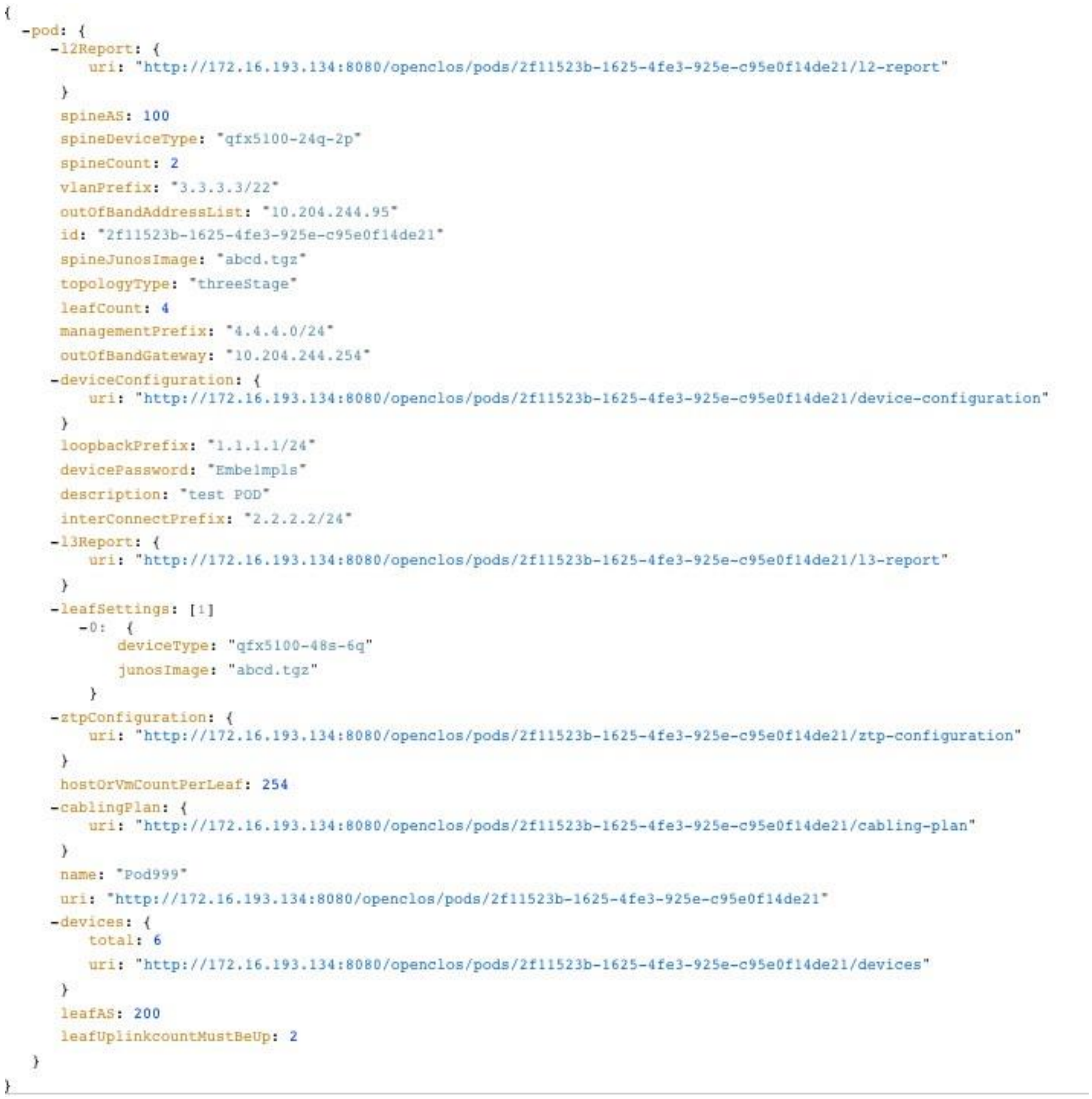
Status Code: 201 - Created

The request has been fulfilled and resulted in a new resource being created.



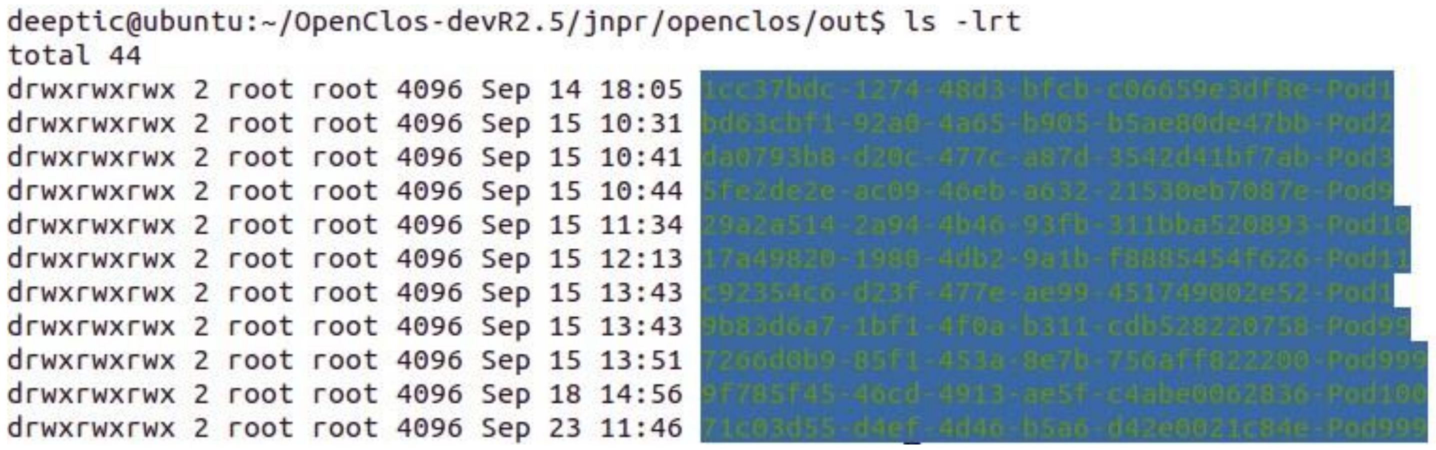
```
2015-09-23 11:58:37,064 [rest       ] [INFO    ] [7f70800f9740] "POST /openclos/pods HTTP/1.1" REQUEST:
2015-09-23 11:58:37,602 [l3Clos     ] [INFO    ] [7f70800f9740] Pod[id='2f11523b-1625-4fe3-925e-c95e0f14de21', name='Pod999']: create
d
2015-09-23 11:58:37,664 [rest       ] [INFO    ] [7f70800f9740] "POST /openclos/pods HTTP/1.1" RESPONSE 201:
172.16.193.1 - - [23/Sep/2015 11:58:37] "POST /openclos/pods HTTP/1.1" 201 1452
```

The created POD information is also returned as a response after the successful POST execution. From the output below a new POD – POD999 with the ID “2f11523b-1625-4fe3-925e-c95e0f14de21” has been created with the attributes as specified by the JSON input schema.



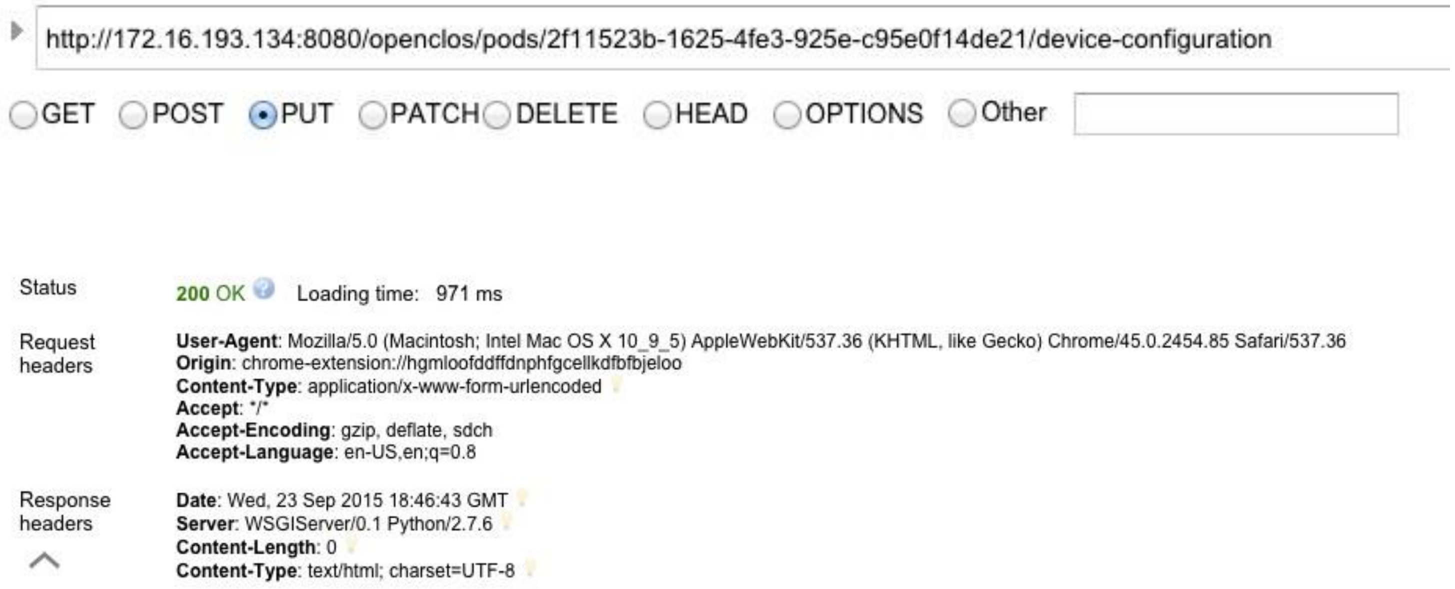
```
{
  -pod: {
    -l2Report: {
        uri: "http://172.16.193.134:8080/openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/l2-report"
     }
     spineAS: 100
     spineDeviceType: "qfx5100-24q-2p"
     spineCount: 2
     vlanPrefix: "3.3.3.3/22"
     outOfBandAddressList: "10.204.244.95"
     id: "2f11523b-1625-4fe3-925e-c95e0f14de21"
     spineJunosImage: "abcd.tgz"
     topologyType: "threeStage"
     leafCount: 4
     managementPrefix: "4.4.4.0/24"
     outOfBandGateway: "10.204.244.254"
    -deviceConfiguration: {
        uri: "http://172.16.193.134:8080/openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/device-configuration"
     }
     loopbackPrefix: "1.1.1.1/24"
     devicePassword: "Embe1mpls"
     description: "test POD"
     interConnectPrefix: "2.2.2.2/24"
    -l3Report: {
        uri: "http://172.16.193.134:8080/openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/l3-report"
     }
    -leafSettings: [1]
       -0:  {
            deviceType: "qfx5100-48s-6q"
            junosImage: "abcd.tgz"
        }
    -ztpConfiguration: {
        uri: "http://172.16.193.134:8080/openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/ztp-configuration"
     }
     hostOrVmCountPerLeaf: 254
    -cablingPlan: {
        uri: "http://172.16.193.134:8080/openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/cabling-plan"
     }
     name: "Pod999"
     uri: "http://172.16.193.134:8080/openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21"
    -devices: {
        total: 6
        uri: "http://172.16.193.134:8080/openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/devices"
     }
     leafAS: 200
     leafUplinkcountMustBeUp: 2
  }
}
```

At this point, however the configuration files are not yet.



```
deeptic@ubuntu:~/OpenClos-devR2.5/jnpr/openclos/out$ ls -lrt
total 44
drwxrwxrwx 2 root root 4096 Sep 14 18:05 1cc37bdc-1274-48d3-bfcb-c06659e3df8e-Pod1
drwxrwxrwx 2 root root 4096 Sep 15 10:31 bd63cbf1-92a0-4a65-b905-b5ae80de47bb-Pod2
drwxrwxrwx 2 root root 4096 Sep 15 10:41 da0793b8-d20c-477c-a87d-3542d41bf7ab-Pod3
drwxrwxrwx 2 root root 4096 Sep 15 10:44 5fe2de2e-ac09-46eb-a632-21530eb7087e-Pod9
drwxrwxrwx 2 root root 4096 Sep 15 11:34 29a2a514-2a94-4b46-93fb-311bba520893-Pod10
drwxrwxrwx 2 root root 4096 Sep 15 12:13 17a49820-1980-4db2-9a1b-f8885454f626-Pod11
drwxrwxrwx 2 root root 4096 Sep 15 13:43 c92354c6-d23f-477e-ae99-451749002e52-Pod1
drwxrwxrwx 2 root root 4096 Sep 15 13:43 9b83d6a7-1bf1-4f0a-b311-cdb528220758-Pod99
drwxrwxrwx 2 root root 4096 Sep 15 13:51 7266d0b9-85f1-453a-8e7b-756aff822200-Pod999
drwxrwxrwx 2 root root 4096 Sep 18 14:56 9f785f45-46cd-4913-ae5f-c4abe0062836-Pod100
drwxrwxrwx 2 root root 4096 Sep 23 11:46 71c03d55-d4ef-4d46-b5a6-d42e0021c84e-Pod999
```

For the configuration generation to complete, a PUT request to the URI as returned by the above JSON response, needs to be completed.

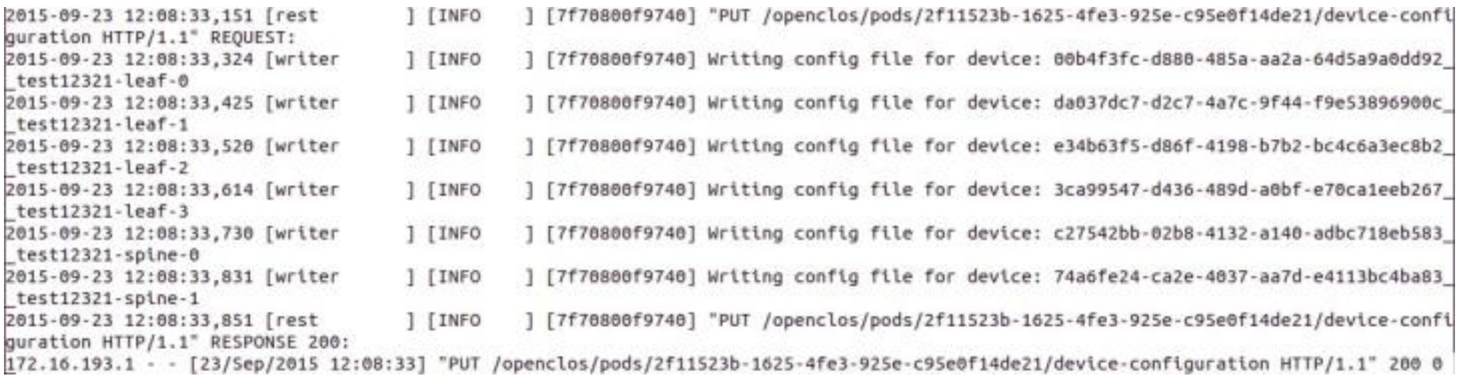


http://172.16.193.134:8080/openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/device-configuration

GET POST [✓ PUT] PATCH DELETE HEAD OPTIONS Other

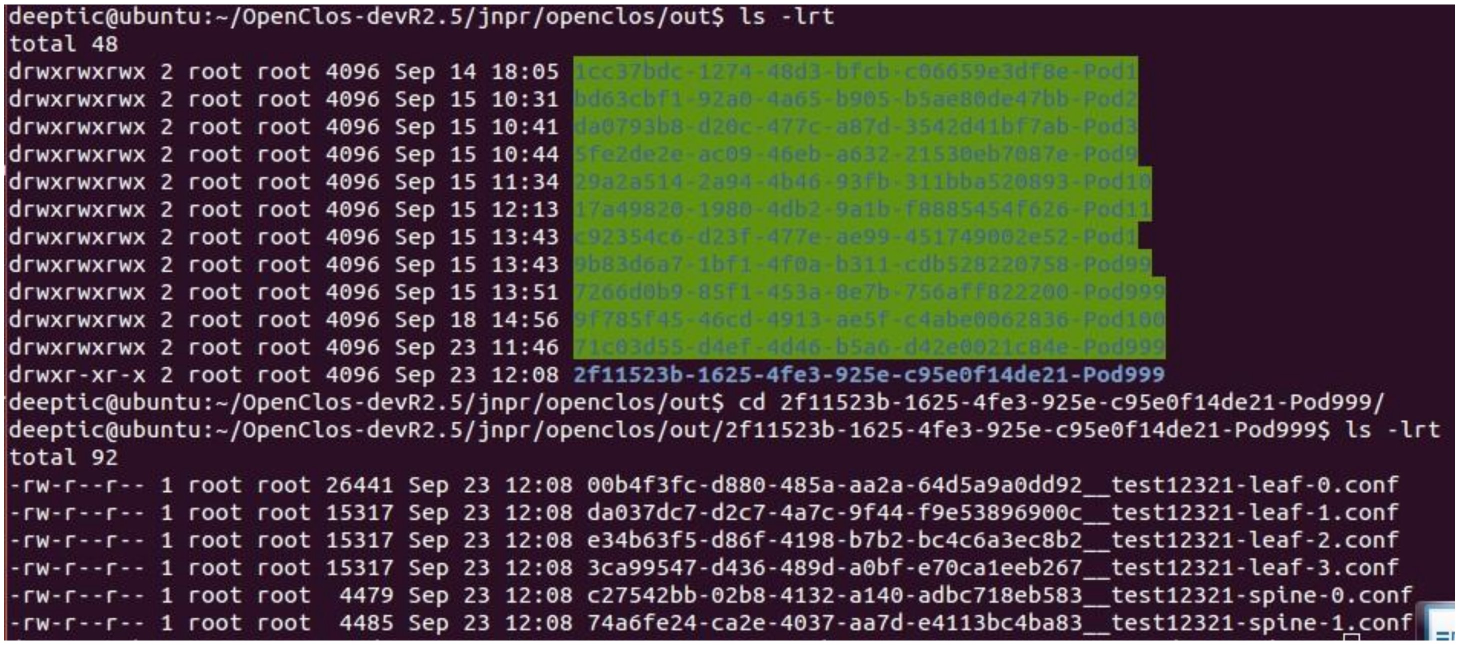
```
Status            200 OK   Loading time:   971 ms

Request           User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.85 Safari/537.36
headers           Origin: chrome-extension://hgmloofddffdnphfgcellkdfbfbjeloo
                  Content-Type: application/x-www-form-urlencoded
                  Accept: */*
                  Accept-Encoding: gzip, deflate, sdch
                  Accept-Language: en-US,en;q=0.8

Response          Date: Wed, 23 Sep 2015 18:46:43 GMT
headers           Server: WSGIServer/0.1 Python/2.7.6
                  Content-Length: 0
                  Content-Type: text/html; charset=UTF-8
```

Configuration files being written into the out folder can also be seen by the messages displayed at the ReST server.



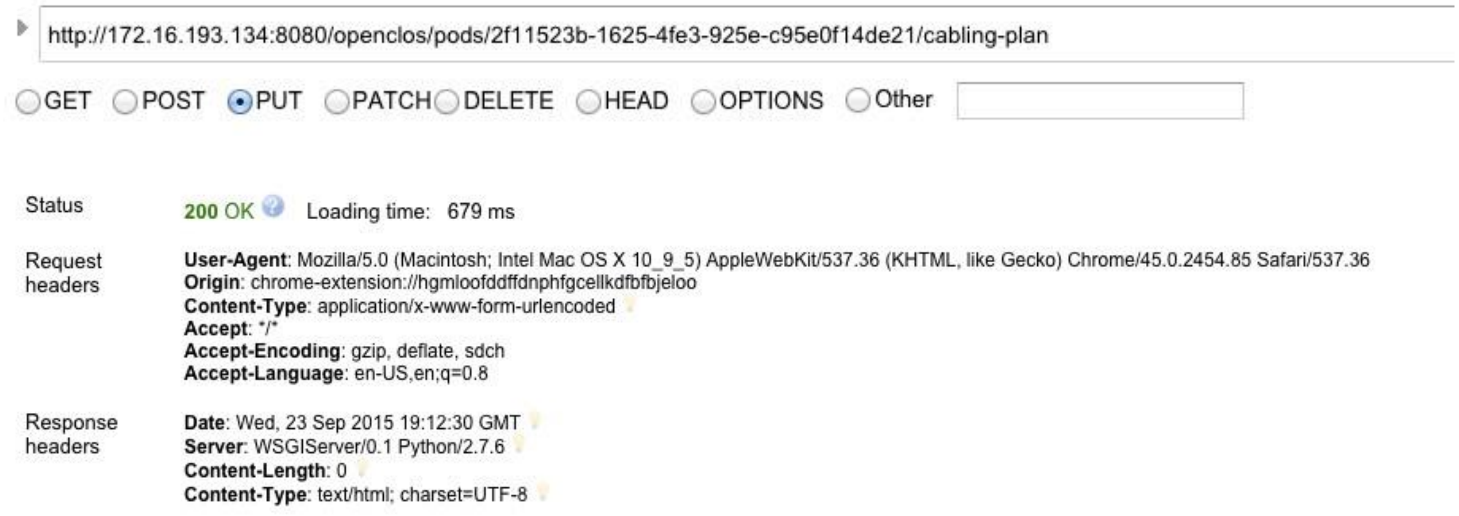
```
2015-09-23 12:08:33,151 [rest        ] [INFO    ] [7f70800f9740] "PUT /openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/device-confi
guration HTTP/1.1" REQUEST:
2015-09-23 12:08:33,324 [writer      ] [INFO    ] [7f70800f9740] Writing config file for device: 00b4f3fc-d880-485a-aa2a-64d5a9a0dd92_
_test12321-leaf-0
2015-09-23 12:08:33,425 [writer      ] [INFO    ] [7f70800f9740] Writing config file for device: da037dc7-d2c7-4a7c-9f44-f9e53896900c_
_test12321-leaf-1
2015-09-23 12:08:33,520 [writer      ] [INFO    ] [7f70800f9740] Writing config file for device: e34b63f5-d86f-4198-b7b2-bc4c6a3ec8b2_
_test12321-leaf-2
2015-09-23 12:08:33,614 [writer      ] [INFO    ] [7f70800f9740] Writing config file for device: 3ca99547-d436-489d-a0bf-e70ca1eeb267_
_test12321-leaf-3
2015-09-23 12:08:33,730 [writer      ] [INFO    ] [7f70800f9740] Writing config file for device: c27542bb-02b8-4132-a140-adbc718eb583_
_test12321-spine-0
2015-09-23 12:08:33,831 [writer      ] [INFO    ] [7f70800f9740] Writing config file for device: 74a6fe24-ca2e-4037-aa7d-e4113bc4ba83_
_test12321-spine-1
2015-09-23 12:08:33,851 [rest        ] [INFO    ] [7f70800f9740] "PUT /openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/device-confi
guration HTTP/1.1" RESPONSE 200:
172.16.193.1 - - [23/Sep/2015 12:08:33] "PUT /openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/device-configuration HTTP/1.1" 200 0
```

The /jnpr/openclos/out path hosts all the files generated for each device – 2 spine devices and 4 leaf devices as specified by the input JSON schema.



```
deeptic@ubuntu:~/OpenClos-devR2.5/jnpr/openclos/out$ ls -lrt
total 48
drwxrwxrwx 2 root root 4096 Sep 14 18:05 1cc37bdc-1274-48d3-bfcb-c06659e3df8e-Pod1
drwxrwxrwx 2 root root 4096 Sep 15 10:31 bd63cbf1-92a0-4a65-b905-b5ae80de47bb-Pod2
drwxrwxrwx 2 root root 4096 Sep 15 10:41 da0793b8-d20c-477c-a87d-3542d41bf7ab-Pod3
drwxrwxrwx 2 root root 4096 Sep 15 10:44 5fe2de2e-ac09-46eb-a632-21530eb7087e-Pod9
drwxrwxrwx 2 root root 4096 Sep 15 11:34 29a2a514-2a94-4b46-93fb-311bba520893-Pod10
drwxrwxrwx 2 root root 4096 Sep 15 12:13 17a49820-1980-4db2-9a1b-f8885454f626-Pod11
drwxrwxrwx 2 root root 4096 Sep 15 13:43 c92354c6-d23f-477e-ae99-451749002e52-Pod1
drwxrwxrwx 2 root root 4096 Sep 15 13:43 9b83d6a7-1bf1-4f0a-b311-cdb528220758-Pod99
drwxrwxrwx 2 root root 4096 Sep 15 13:51 7266d0b9-85f1-453a-8e7b-756aff822200-Pod999
drwxrwxrwx 2 root root 4096 Sep 18 14:56 9f785f45-46cd-4913-ae5f-c4abe0062836-Pod100
drwxrwxrwx 2 root root 4096 Sep 23 11:46 71c03d55-d4ef-4d46-b5a6-d42e0021c84e-Pod999
drwxr-xr-x 2 root root 4096 Sep 23 12:08 2f11523b-1625-4fe3-925e-c95e0f14de21-Pod999
deeptic@ubuntu:~/OpenClos-devR2.5/jnpr/openclos/out$ cd 2f11523b-1625-4fe3-925e-c95e0f14de21-Pod999/
deeptic@ubuntu:~/OpenClos-devR2.5/jnpr/openclos/out/2f11523b-1625-4fe3-925e-c95e0f14de21-Pod999$ ls -lrt
total 92
-rw-r--r-- 1 root root 26441 Sep 23 12:08 00b4f3fc-d880-485a-aa2a-64d5a9a0dd92__test12321-leaf-0.conf
-rw-r--r-- 1 root root 15317 Sep 23 12:08 da037dc7-d2c7-4a7c-9f44-f9e53896900c__test12321-leaf-1.conf
-rw-r--r-- 1 root root 15317 Sep 23 12:08 e34b63f5-d86f-4198-b7b2-bc4c6a3ec8b2__test12321-leaf-2.conf
-rw-r--r-- 1 root root 15317 Sep 23 12:08 3ca99547-d436-489d-a0bf-e70ca1eeb267__test12321-leaf-3.conf
-rw-r--r-- 1 root root  4479 Sep 23 12:08 c27542bb-02b8-4132-a140-adbc718eb583__test12321-spine-0.conf
-rw-r--r-- 1 root root  4485 Sep 23 12:08 74a6fe24-ca2e-4037-aa7d-e4113bc4ba83__test12321-spine-1.conf
```

A cabling plan can also be generated to visually represent the topology for which the device configurations have been generated. The POD UID again is referenced in this transaction to identify the relevant topology.



```
http://172.16.193.134:8080/openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/cabling-plan
```

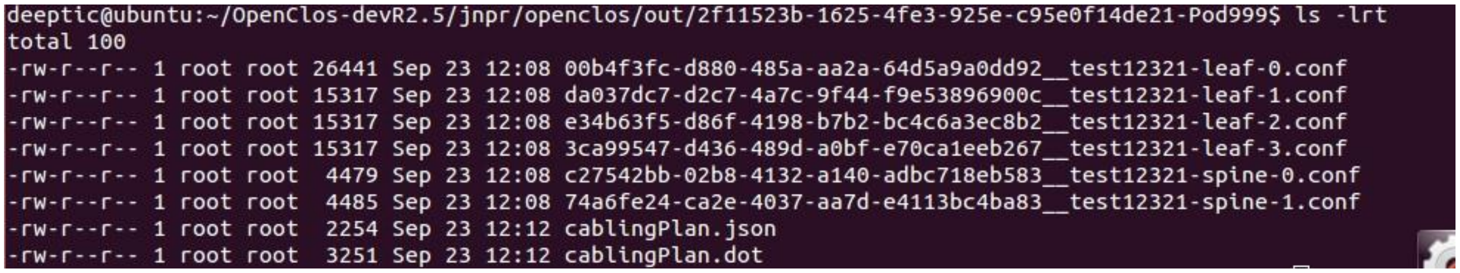
GET POST [✓ PUT] PATCH DELETE HEAD OPTIONS Other

Status 200 OK Loading time: 679 ms

Request headers
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.85 Safari/537.36
Origin: chrome-extension://hgmloofddffdnphfgcellkdfbfbjeloo
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8

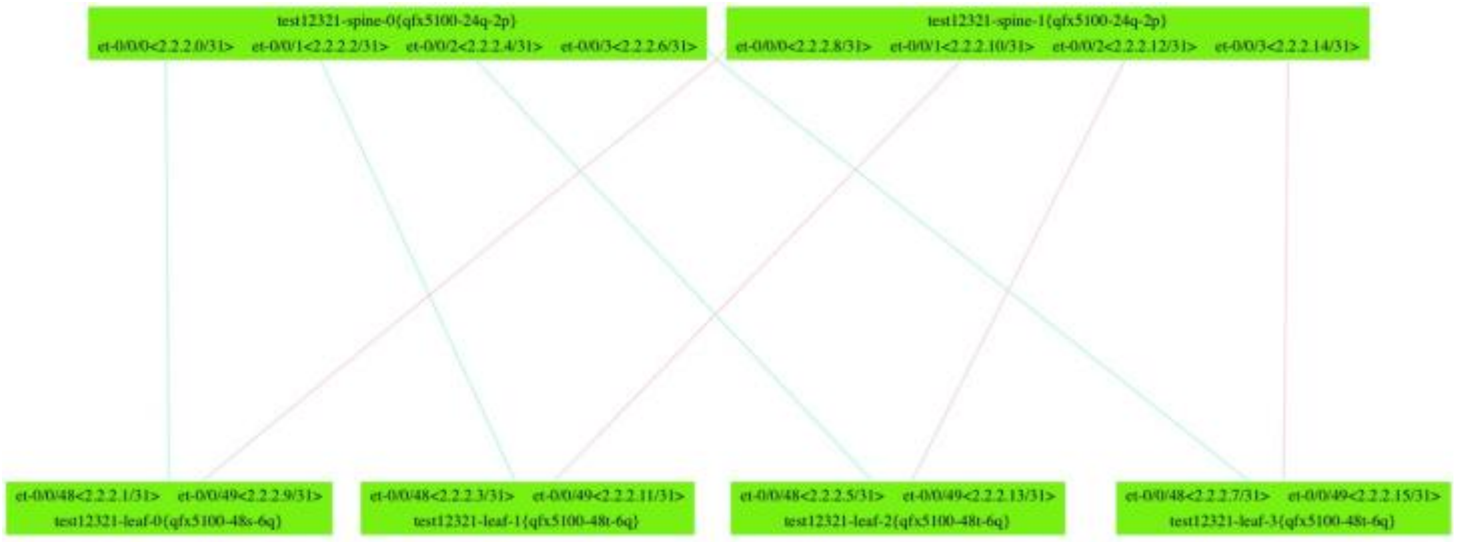
Response headers
Date: Wed, 23 Sep 2015 19:12:30 GMT
Server: WSGIServer/0.1 Python/2.7.6
Content-Length: 0
Content-Type: text/html; charset=UTF-8

The result is the creation of cablingPlan.json and cablingPlan.dot files, which can be further used as desired.



```
deeptic@ubuntu:~/OpenClos-devR2.5/jnpr/openclos/out/2f11523b-1625-4fe3-925e-c95e0f14de21-Pod999$ ls -lrt
total 100
-rw-r--r-- 1 root root 26441 Sep 23 12:08 00b4f3fc-d880-485a-aa2a-64d5a9a0dd92__test12321-leaf-0.conf
-rw-r--r-- 1 root root 15317 Sep 23 12:08 da037dc7-d2c7-4a7c-9f44-f9e53896900c__test12321-leaf-1.conf
-rw-r--r-- 1 root root 15317 Sep 23 12:08 e34b63f5-d86f-4198-b7b2-bc4c6a3ec8b2__test12321-leaf-2.conf
-rw-r--r-- 1 root root 15317 Sep 23 12:08 3ca99547-d436-489d-a0bf-e70ca1eeb267__test12321-leaf-3.conf
-rw-r--r-- 1 root root  4479 Sep 23 12:08 c27542bb-02b8-4132-a140-adbc718eb583__test12321-spine-0.conf
-rw-r--r-- 1 root root  4485 Sep 23 12:08 74a6fe24-ca2e-4037-aa7d-e4113bc4ba83__test12321-spine-1.conf
-rw-r--r-- 1 root root  2254 Sep 23 12:12 cablingPlan.json
-rw-r--r-- 1 root root  3251 Sep 23 12:12 cablingPlan.dot
```

Below a png file has been created using the dot file which displays the device topology.

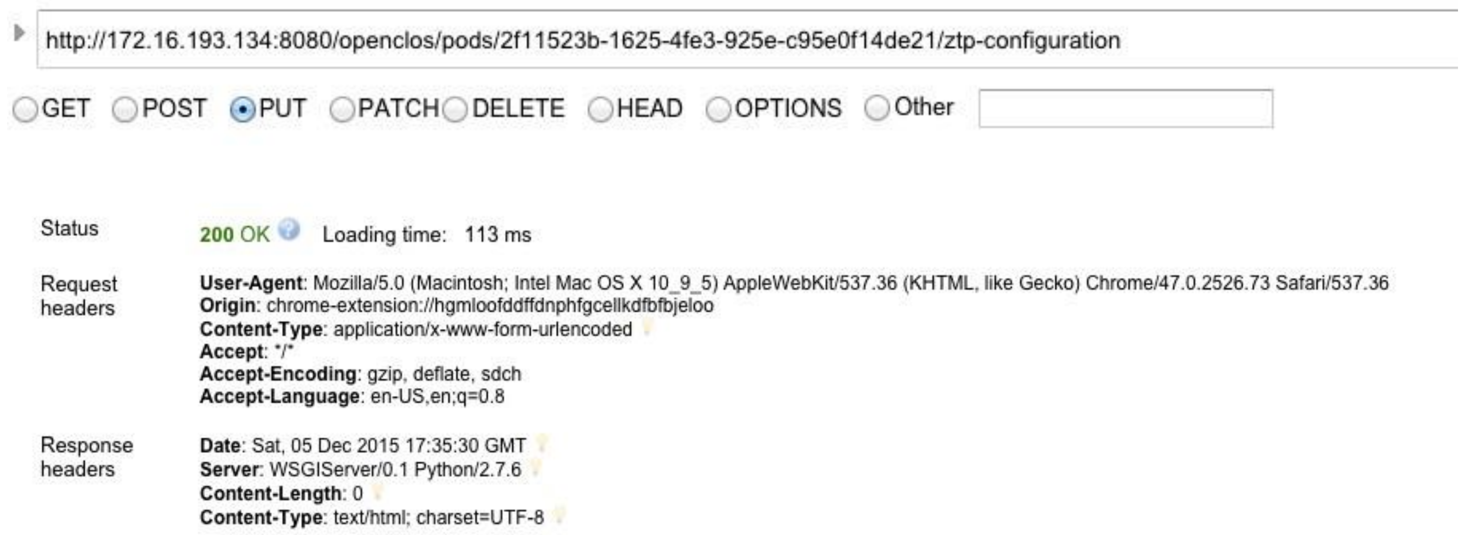


OpenClos for Zero Touch Provisioning

Zero Touch Provisioning (ZTP) offers a better solution to provision networking equipment in data centers automatically without any manual intervention. Devices can be provisioned simultaneously by interacting with the ZTP server. Network devices use information that is configured on a DHCP server to load a specified software version and configuration file, making the provisioning process quick and efficient. Commonly, a Linux Server running the Internet System Consortium (ISC) DHCP server, is used as the ZTP server. The ZTP process uses the DHCP vendor options set in the DHCP ACK from the DHCP server to proceed with the ZTP process. More details on the ZTP process for Junos switches can be found here.

OpenClos can be used to facilitate ZTP by generating configuration files to be used by both the ZTP server and the Junos devices to be provisioned. The steps described above can be used to generate the configuration files to be used for provisioning network equipment.

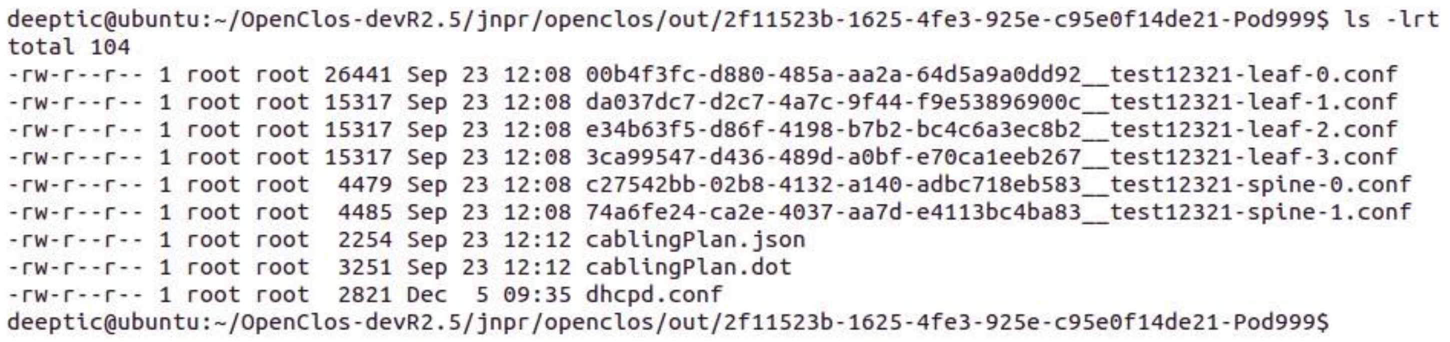
To configure the ZTP server, user defined parameters can be used to modify /openclos/conf/openclos.yaml. With the POD ID specified, the ReST call can be used to generate the 'dhcpd.conf' file for the ZTP server.



http://172.16.193.134:8080/openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/ztp-configuration

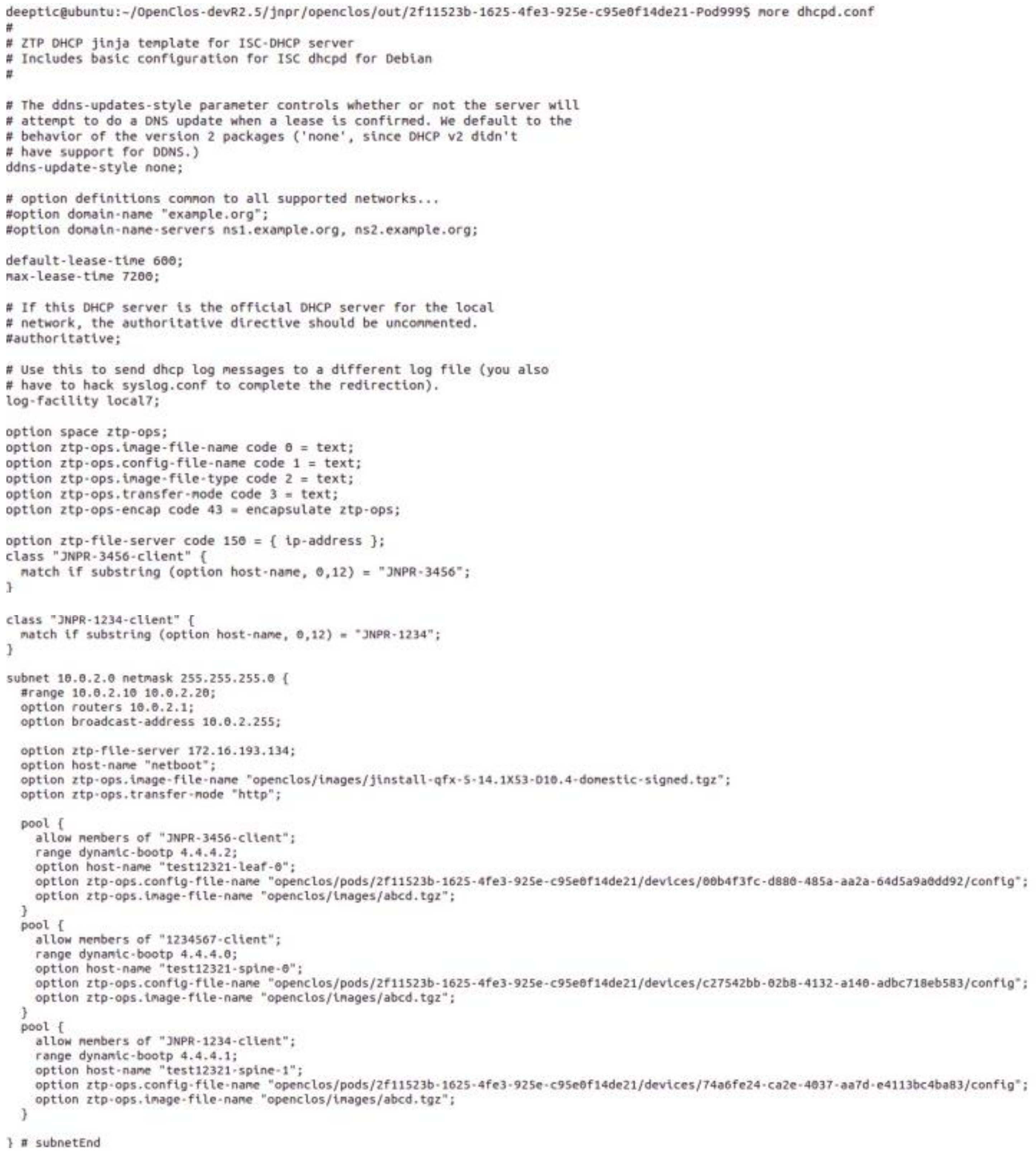
GET POST [✓ PUT] PATCH DELETE HEAD OPTIONS Other

Status	200 OK Loading time: 113 ms
Request headers	**User-Agent**: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.73 Safari/537.36 **Origin**: chrome-extension://hgmloofddffdnphfgcellkdfbfbjeloo **Content-Type**: application/x-www-form-urlencoded **Accept**: */* **Accept-Encoding**: gzip, deflate, sdch **Accept-Language**: en-US,en;q=0.8
Response headers	**Date**: Sat, 05 Dec 2015 17:35:30 GMT **Server**: WSGIServer/0.1 Python/2.7.6 **Content-Length**: 0 **Content-Type**: text/html; charset=UTF-8



```
deeptic@ubuntu:~/OpenClos-devR2.5/jnpr/openclos/out/2f11523b-1625-4fe3-925e-c95e0f14de21-Pod999$ ls -lrt
total 104
-rw-r--r-- 1 root root 26441 Sep 23 12:08 00b4f3fc-d880-485a-aa2a-64d5a9a0dd92__test12321-leaf-0.conf
-rw-r--r-- 1 root root 15317 Sep 23 12:08 da037dc7-d2c7-4a7c-9f44-f9e53896900c__test12321-leaf-1.conf
-rw-r--r-- 1 root root 15317 Sep 23 12:08 e34b63f5-d86f-4198-b7b2-bc4c6a3ec8b2__test12321-leaf-2.conf
-rw-r--r-- 1 root root 15317 Sep 23 12:08 3ca99547-d436-489d-a0bf-e70ca1eeb267__test12321-leaf-3.conf
-rw-r--r-- 1 root root  4479 Sep 23 12:08 c27542bb-02b8-4132-a140-adbc718eb583__test12321-spine-0.conf
-rw-r--r-- 1 root root  4485 Sep 23 12:08 74a6fe24-ca2e-4037-aa7d-e4113bc4ba83__test12321-spine-1.conf
-rw-r--r-- 1 root root  2254 Sep 23 12:12 cablingPlan.json
-rw-r--r-- 1 root root  3251 Sep 23 12:12 cablingPlan.dot
-rw-r--r-- 1 root root  2821 Dec  5 09:35 dhcpd.conf
deeptic@ubuntu:~/OpenClos-devR2.5/jnpr/openclos/out/2f11523b-1625-4fe3-925e-c95e0f14de21-Pod999$
```

Below are some snippets of ISC DHCP server file with ZTP options set.



```
deeptic@ubuntu:~/OpenClos-devR2.5/jnpr/openclos/out/2f11523b-1625-4fe3-925e-c95e0f14de21-Pod999$ more dhcpd.conf
#
# ZTP DHCP jinja template for ISC-DHCP server
# Includes basic configuration for ISC dhcpd for Debian
#

# The ddns-updates-style parameter controls whether or not the server will
# attempt to do a DNS update when a lease is confirmed. We default to the
# behavior of the version 2 packages ('none', since DHCP v2 didn't
# have support for DDNS.)
ddns-update-style none;

# option definitions common to all supported networks...
#option domain-name "example.org";
#option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
#authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

option space ztp-ops;
option ztp-ops.image-file-name code 0 = text;
option ztp-ops.config-file-name code 1 = text;
option ztp-ops.image-file-type code 2 = text;
option ztp-ops.transfer-mode code 3 = text;
option ztp-ops-encap code 43 = encapsulate ztp-ops;

option ztp-file-server code 150 = { ip-address };
class "JNPR-3456-client" {
  match if substring (option host-name, 0,12) = "JNPR-3456";
}

class "JNPR-1234-client" {
  match if substring (option host-name, 0,12) = "JNPR-1234";
}

subnet 10.0.2.0 netmask 255.255.255.0 {
  #range 10.0.2.10 10.0.2.20;
  option routers 10.0.2.1;
  option broadcast-address 10.0.2.255;

  option ztp-file-server 172.16.193.134;
  option host-name "netboot";
  option ztp-ops.image-file-name "openclos/images/jinstall-qfx-5-14.1X53-D10.4-domestic-signed.tgz";
  option ztp-ops.transfer-mode "http";

  pool {
    allow members of "JNPR-3456-client";
    range dynamic-bootp 4.4.4.2;
    option host-name "test12321-leaf-0";
    option ztp-ops.config-file-name "openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/devices/00b4f3fc-d880-485a-aa2a-64d5a9a0dd92/config";
    option ztp-ops.image-file-name "openclos/images/abcd.tgz";
  }
  pool {
    allow members of "1234567-client";
    range dynamic-bootp 4.4.4.0;
    option host-name "test12321-spine-0";
    option ztp-ops.config-file-name "openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/devices/c27542bb-02b8-4132-a140-adbc718eb583/config";
    option ztp-ops.image-file-name "openclos/images/abcd.tgz";
  }
  pool {
    allow members of "JNPR-1234-client";
    range dynamic-bootp 4.4.4.1;
    option host-name "test12321-spine-1";
    option ztp-ops.config-file-name "openclos/pods/2f11523b-1625-4fe3-925e-c95e0f14de21/devices/74a6fe24-ca2e-4037-aa7d-e4113bc4ba83/config";
    option ztp-ops.image-file-name "openclos/images/abcd.tgz";
  }

} # subnetEnd
```

Conclusion

ReST API for OpenClos can be used as an interface to automate the creation of JSON templates, for IP fabric configuration generation. It is lightweight and flexible and can be an ideal provisioning tool in datacenter environments