Sveučilište u Rijeci
TEHNIČKI FAKULTET

OpenStack vježbe za kolegij :

Inženjerstvo kompleksnih programskih sustava

-Python vježbe koristeći REST API-

Nositelj: izv. prof. dr. sc. Tihana Galinac Grbac

Autor: Rino Rodin

Srpanj 2018. Rijeka

1. Python skripte koristeći REST API

1.1 Autentifikacija

U skripti openstack_api_utils.py implementirane su metode za dohvat autentifikacijskog tokena i endpointa (URL-a) određenog OpenStack servisa.

1.2 Dohvaćanje autentifikacijskog tokena.

Dohvaćanje se vrši pomoću metode get_auth_token(). Temelj metode get_auth_token() je slanje POST zahtjev na adresu: http://10.30.1.2:5000/v3/auth/tokens.
Post zahtjev koristi zaglavlja (headers) u kojima je definiran tip sadržaja zahtjeva, u ovom slučaju to je: application/json. Podaci (sadržaj) su definirani u json obliku unutar varijable „data“. Podaci koji se prenose definiraju korisničko ime, lozinku te opseg (scope) dohvata tokena. U ovom slučaju token se dohvaća na razini admin projekta(tenant-a) jer želimo administratorska dopuštenja. Inače, token je moguće dohvatiti i za neku drugu razinu dopuštenja. Dio skripte openstack_api_utils.py zadužen za dohvaćanje tokena prikazan je na slici 1.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ukljucivanje potrebnih knjiznica knjiznica za slanje http zahtjeva
import requests
# knjiznica za rad s json formatom
import json


# ovo su podaci koji su potrebni Identity servisu da nam izda token najvaznije je
# dobro zadati lozinku i ime treba paziti da je ime odabranog projekta tocno
data = {
 "auth": {
    "identity": {
      "methods": ["password"],
      "password": {
        "user": {
          "name": "admin",
          "domain": {"id": "default"},
          "password": "AkX3UrB9"
        }
      }
    },
    "scope": {
      "project": {
        "name": "admin",
        "domain": {"id": "default"}
      }
    }
  }
}

# funkcija koja se poziva na pocetku drugih skripti kako bi se izvrsila autentikacija
# i kako bi se dobio auth token koji je potreban za daljni rad s Openstack API-jem
def get_auth_token():
    # definira se zaglavlje zahtjeva, podatci se salje u obliku json datoteke pa je
    # onda potrebno u zaglavlju Content-Type postaviti na slijedecu vrijednost
    headers = {'Content-Type': 'application/json'}
    # salje se POST zahtjev sa gore definiranim podacima na Identity servis koji je zaduzen za autentikaciju
    r = requests.post("http://10.30.1.2:5000/v3/auth/tokens", headers=headers, data=json.dumps(data))
    print("Creating new auth token...")
    # ispis vracenog statusnog koda kao povratna informacija korisniku da je sve u redu
    print(r.status_code, r.reason)
    # kao sto je navedeno u dokumentaciji uspjesni zahtjev vraca token u zaglavlju
    # ovdje se taj token vraca tamo gdje se funkcija pozvala
    return r.headers.get("x-subject-token")
```

Slika 1. Dohvaćanje autentifikacijskog tokena

1.3 Dohvaćanje endpointa (URL-a) servisa

Kako bi korisnik mogao zatražiti usluge od određenog servisa OpenStack-a potebno je znati URL (endpoint) željenog servisa na koji se onda može poslati korisnički zahtjev. Dohvaćanje URL-a željenog servisa odvojili smo u metodu get_endpoint() koja se također nalazi unutar skripte openstack_api_utils.py. Metoda se koristi od strane svake skripte koja želi komunicirati s nekim od servisa. Prilikom poziva metode potrebno joj je proslijediti dva parametra:

1. Naziv željenog servisa (npr. glance, nova, neutron)
2. Autentifikacijski token dobiven pomoću prethodno opisane metode get_auth_token()

Metoda kao povratni parametar vraća URL (endpoint) onog servisa čiji je naziv proslijeđen metodi kao parametar. Dio skripte openstack_api_utils.py zadužen za dohvaćanje URL-a (endpoint-a), tj. metoda get_endpoint(...) prikazana je na slici 2.

```
48  # funkcija koja dohvaca endpoint (URL) na kojoj se nalazi pojedini servis
49  # kao argument prima ime tog servisa i auth token
50  def get_endpoint(service_name, auth_token):
51      # u zaglavlju se salje auth token
52      headers = {'X-Auth-Token': auth_token}
53      # kao sto pise u dokumentaciji salje se GET zahtjev na navedeni url
54      r = requests.get("http://10.30.1.2:5000/v3/auth/catalog", headers=headers)
55      # pretvaranje rezultata u json format
56      results_json = r.json()
57
58      # iteriranje po json datoteci po svakom izlistanom servisu da bi se izvukao URL modula cije je ime zadano kao argument
59      for catalog_entry in results_json["catalog"]:
60          # if uvjet usporedjuje odgovara li ime servisa onome iz argumenta
61          if catalog_entry["name"] == service_name:
62              # ako odgovara potrebno je proci kroz sva njegova sucelja i pronaci ono koje je javno
63              for endpoint in catalog_entry["endpoints"]:
64                  if endpoint["interface"] == "public":
65                      # kada smo nasli njegovo javno sucelje spremimo njegov url u varijablu koju na kraju vracamo
66                      service_endpoint = endpoint["url"]
67      return service_endpoint
```

Slika 2. Dohvaćanje endpointa (URL-a servisa)

2. Skripta 1.py

Na početku skripte Skripta 1.py dohvaća se token za autentifikaciju te glance endpoint (URL). Glance je OpenStack servis koji omogućava dodavanje i uređivanje slika OS-a. Naziv spremnika, kao i putanju do .img datoteke korisnik unosi kroz terminal. POST zahtjevom kreira se spremnik (engl. container) s karakteristikama navedenim u „data“ varijabli. Karakteristike su: format spremnika, format .img datoteke koju će spremink sadržavati , te naziv spremnika. Spremnik se kreira POST zahtjevom na glance endpoint (URL) uz dodatak „/v2/images“. Nakon kreacije spremnika u njega je potrebno podignuti odgovarajuću .img datoteku. To se čini pomoću PUT zahtjeva koji se šalje na glance endpoint (URL) uz dodatak "/v2/images/" + image_id + "/file" gdje varijabla „image_id“ označava ID spremnika koji smo prethodno dohvatili iz odgovora dobivenog nakon kreacije samog spremnika. Nužno je napomenuti da se zaglavlja za podizanje .img datoteke razlikuju od zaglavlja koja ćemo korstiti ubuduće jer je tip sadržaja tretnutno "application/octet-stream" a ne "application/json" kao što će to kasnije biti slučaj. Sadržaj je u ovom slučaju binary datoteka te ju je potrebno učitati pomoću python metode open() koja prima 2 parametra:

1. Putanja do binary datoteke
2. Oznaka da se radi o čitanju binary datoteke ('rb'-read binary)

Nakon učitavanja, binary datoteku potrebno je i pročitati. To se čini pozivanjem read() metode na objektu koji kreira open() metoda. Podizanjem binary datoteke završava kreacija slike OS-a, skripta ispisuje poruku o uspješnoj kreaciji slike OS-a, te sliku tada možemo iskoristiti za stvaranje nove instance. Skripta 1.py prikazana je na slici 3.

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# dohvacanje vanjskih funkcija i knjiznica funkcija
from openstack_api_utils import get_auth_token, get_endpoint
import json
import requests

#dohvacanje autentifikacijskog tokena
auth_token = get_auth_token()

#specificiranje zaglavlja
headers = {'X-Auth-Token': auth_token}
#dohvacanje glance endpointa (URL-a)
glance_endpoint = get_endpoint("glance", auth_token)

#trazimo korisnika da unese naziv nove slike OS-a i lokaciju image datoteke koju zeli ucitati
image_name = raw_input("\nEnter new image name: ")
image_path = raw_input("\nEnter image file location: ")

#specifikacija podataka o containeru image datoteke
data = {
    "container_format": "bare",
    "disk_format": "qcow2",
    "name": image_name
}
#specifikacija zaglavlja
headers={
    'X-Auth-Token': auth_token,
    "Content-Type": "application/json"
}
#saljemo POST zahtjev za stvaranje containera
r = requests.post(glance_endpoint + "/v2/images", headers=headers, data=json.dumps(data))
print "Creating image..."
#iz odgovora dohvacamo id containera
results_json = r.json()
image_id = results_json["id"]

#otvaramo binary datoteku (willy-server-cloudimg-amd64-disk1.img)
data = open(image_path, 'rb').read()

#specifikacija zaglavlja
headers={
    'X-Auth-Token': auth_token,
    "Content-Type": "application/octet-stream"
}
#PUT zahtjev za pridruzivanje .img datoteke containeru na OpenStacku
r = requests.put(glance_endpoint + "/v2/images/" + image_id + "/file", headers=headers, data=data)
if r.status_code == 204
    # ispis da je proces zavrsen
    print "Image Created!"
else:
    #ispis poruke o gresci
    print "Image creation failed"
```

Slika 3. Skripta 1.py Upload .img datoteke u odgovarajući container na OpenStacku

3. Skripta 2.py

Na početku skripte 2 dohvaćamo glance endpoint (URL). Na glance endpoint dodajemo nastavak + "/v2/images" te na taj način dobivamo potpunu adresu na koju možemo poslati get zahtjev. Get zahtjev poslan na tu adresu vraća odgovor unutar kojeg se nalazi lista svih slika OS-a. Nakon provjere je li odgovor valjan odgovor pretvaramo u lako čitljivi json format te iteriramo po takvim podacima i ispisujemo sve slike koje su u njima sadržane. Zatim od korisnika tražimo da unese naziv slike (image OS-a) te koristimo uneseni naziv kako bismo dohvatili ID slike čiji je naziv korisnik unio.
Ovoga puta get zahtjev šaljemo na glance endpoint uz dodatak + "/v2/images?name=" + name gdje je „name“ ime koje je korisnik unio.
Odgovor ponovno pretvaramo u json format te iz njega dohvaćamo ID.
ID se na kraju ispisuje na ekran. Skripta 2 prikazana je na slici 4.

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# knjiznice koje su potrebne za rad skripte
# authentication je py modul koji smo kreirali da provodi autentikaciju
from authentication import get_auth_token, get_endpoint
# knjiznica za slanje http zahtjeva
import requests
# knjiznica za rad s json datotekama
import json

# dohvaca auth token pomocu funkcije iz authenticate.py
# i sprema ga u varijablu auth_token
auth_token = get_auth_token()
# dohvaca se URL glance servssa pomocu funkcije pomocne knjiznice
glance_endpoint = get_endpoint("glance", auth_token)
# ispis teksta na terminal
print "List of all images by name and size:"

# kreairanje zaglavlja zahtjeva, zaglavlje mora sadrzavati token u ovom obliku
headers = {'X-Auth-Token': auth_token}
# slanje GET zahtjeva na url od glance servisa sa ukljucenim zaglavljem
r = requests.get(glance_endpoint + "/v2/images", headers=headers)
# ispis koda koji smo dobili kao povratnu informaciju
print(r.status_code, r.reason)

# pretvaranje rezultata u json format
json_data = r.json()
# petlja u kojoj se iterira po svim slikama koje se dobiju
for image in json_data["images"]:
    # ispis podataka (ime i velicina) pojedine slike uz formatiranje
    print repr(image["name"]).ljust(50) + repr(image["size"]).rjust(15)

# trazenje unosa preko terminala od strane korisnika
name = raw_input('\nSearch for image by name: ')

# pitaj korisnika za unos imena po kojem se pretrazuje
print('\nLooking for %s...\n' % name)


# dohvacanje specificne OS slike putem njezinog imena
# salje se GET zahtjev s query parametrom unesene rijeci
r = requests.get(glance_endpoint + "/v2/images?name=" + name, headers=headers)
# ponovno se rezultat pretvara u json format
results_json = r.json()
# iz rezultata se dohvaca polje svih slika koje je glance servis vratio
images = results_json["images"]
# provjera postoji li rezultat pretrage
if images:
    # ako postoji za svaku sliku ispis njen ID
    for image in images:
        print('Images found, id is:%s' % image["id"])
else:
    # ako upit ne vrati nikakav rezultat
    print "Image Not Found"
```

Slika 4. Skripta 2.py Ispis slika OS-a te dohvaćanje ID-a slike putem njenog imena

4. Skripta 3.py

Skripta se sastoji od dva dijela, prvi se dio odnosi na kreiranje mreže te dohvaćanje ID-ja kreirane mreže dok se u drugom dijelu kreiranoj mreži, na temelju ID-a, dodjeljuje određena podmreža (subnet). Prvi dio skripte prikazan je na slici 5.

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # dohvacanje vanjskih funkcija i knjiznica funkcija
5  from openstack_api_utils import get_auth_token, get_endpoint
6  import json
7  import requests
8
9  # dohvacanje autentifikacijskog tokena pomocu metode iz skripte openstack_api_utils.py
10 auth_token = get_auth_token()
11
12 #dohvacanje URL-a (endpointa) servisa neutron kojem ce se proslijediti zahtjev za kreiranje mreze
13 neutron_endpoint = get_endpoint("neutron", auth_token)
14
15 #trazimo od korisnika unos imena nove mreze koju cemo kreirati
16 network_name = raw_input("\nEnter new network name: ")
17
18
19 #definicija zaglavlja POST zahtjeva
20 headers={
21     'X-Auth-Token': auth_token,
22     "Content-Type": "application/json"
23 }
24
25 #varijabla data kojom se prenose podaci, tocnije ime mreze koju cemo kreirati
26 data = {
27     "network": {"name": network_name}
28 }
29
30 #slanje post zahtjeva sa specificiranim zaglavljima i podacima na URL (endpoint) neutrona
31 # te primanje odgovora (response) u varijablu r
32 r = requests.post(neutron_endpoint + "/v2.0/networks", headers=headers, data=json.dumps(data))
33
34 #provjera je li mreza uspjesno kreirana,  status code 201 oznacava uspjesnu kreaciju
35 if r.status_code != 201:
36     sys.exit("Network creation error!")
37 else:
38     #pretvaranje odgovora u json format
39     results_json = r.json()
40     #iscitavanje ID-a novokreirane mreze iz odgovora koji smo dobili od servisa neutron
41     net_dict = results_json['network']
42     network_id = net_dict['id']
43     #ispis potvrde kreacije i ID-a novokreirane mreze
44     print('Network %s created' % network_id)
```

Slika 5. Stvaranje mreže u skripti skripta 3.py.

Na početku skripte se koristi metoda get_auth_token() iz skripte openstack_api_utils.py koja dohvaća autentifikacijski token. Token se prosljeđuje metodi get_endpoint(...) koja dohvaća neutronov endpoint (URL) te ga sprema kao string u varijablu neutron_endpoint. Endpoint je adresa (URL) na kojoj servis neutron poslužuje klijente. Iako to nije vidljivo iz skripte, dohvaćeni neutron endpoint bit će jednak http://10.30.1.2:9696/ . Na liniji 16 tražimo od korisnika unos naziva nove mreže. Zatim slijedi definicija zaglavlja te podataka(zadržaja). U sadržaju (varijabla „data“) specificiramo naziv mreže te koristimo varijablu „network_name“ u koju je pohranjen korisnički unos. Na neutronov endpoint dodajemo „/v2.0/networks „ te time dobivamo potpunu adresu servisa kojemu možemo proslijediti zahtjev za kreiranje nove mreže. Na potpunu adresu upućujemo novi post zahtjev koji u svom tijelu sadrži json podatke („data“) koji specificiraju naziv mreže. Naziv mreže student mijenja po potrebi , a u ovom slučaju odabran je naziv StudentNet. Iz odgovora (response) koji dobivamo od servisa neutron dohvaćamo ID novokreirane mreže. Taj ID koristit demo prilikom kreiranja podmreže (subnet) u drugom dijelu skripte. Kreiranje podmreže prikazano je na slici 6.

```
50  #creating subnets
51
52  #trazimo od korisnika unos imena nove podmreze koju cemo kreirati
53  subnet_name = raw_input("\nEnter new subnet name: ")
54
55  #trazimo od korisnika unos cidr-a (Classless Inter-Domain Routing) nove podmreze koju cemo kreirati
56  subnet_cidr = raw_input("\nEnter subnet cidr: ")
57
58  #varijabla data kojom se prenose podaci, tocnije ime podmreze koju cemo kreirati
59  #cidr podmreze, verzija ip adrese, ID mreze kojoj se dodjeljuje podmreza
60  data = {
61      "subnet":{"name":subnet_name,
62                "cidr":subnet_cidr,
63                "ip_version":4,
64                "network_id":network_id}
65  }
66
67  #definicija zaglavlja POST zahtjeva
68  headers={
69      'X-Auth-Token': auth_token,
70      "Content-Type": "application/json"
71  }
72
73  #slanje post zahtjeva sa specificiranim zaglavljima i podacima na URL (endpoint) neutrona
74  # te primanje odgovora (response) u varijablu r
75  r = requests.post(neutron_endpoint + "/v2.0/subnets", headers=headers, data=json.dumps(data))
76
77  #provjera je li podmreza uspjesno kreirana,  status code 201 oznacava uspjesnu kreaciju
78  if r.status_code != 201:
79      sys.exit("Subnet creation error!")
80  else:
81      #pretvaranje odgovora u json format
82      results_json = r.json()
83      #iscitavanje imena novokreirane podmreze iz odgovora koji smo dobili od servisa neutron
84      subnet_dict = results_json['subnet']
85      subnet_name = subnet_dict['name']
86      #ispis potvrde kreacije i naziva novokreirane podmreze
87      print('Subnet %s created' % subnet_name)
```

Slika 6. Kreiranje podmreže unutar skripte skripta3.py

U drugom dijelu skripte stvaramo podmrežu (subnet) te ga pomoću ID-ja mreže dodjeljujemo odgovarajućoj mreži. Stvaranje mreže također se vrši pomoću post zahtjeva, a u tijelu zahtjeva („data“) specificiramo naziv podmreže, u ovom slučaju StudentSubnet , cidr(*Classless Inter-Domain Routing*) podmreže, ip verziju te ID mreže kojoj se subnet dodjeljuje. Parametre naziv podmreže i cidr korisnik unosi preko treminala. Na neutronov URL(endpoint) ovoga puta dodajemo „/v2.0/subnets“ te na taj način dobivamo potpunu adresu servisa neutron na koju možemo proslijediti zahtjev za kreiranje podmreže. Ukoliko je kod odgovora koji dobivamo od neutrona jednak kodu uspješne kreacije podmreže (kod 201) i odgovora dohvaćamo naziv subneta te ispisujemo poruku o uspješnoj kreaciji podmreže.

5. Skripta 4.py

Radi preglednosti skripta je podijeljena na tri dijela. Prvi dio odnosi se na dohvaćanje svih potrebnih parametara koji će u drugom dijelu biti korišteni za stvaranje rutera i porta. Treći dio skripte prikazuje dodjeljivanje porta ruteru. Prvi dio skripte, dohvaćanje potrebnih parametara prikazan je na slici 7.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

#dohvacanje vanjskih funkcija i knjiznica funkcija
import requests
import json
from openstack_api_utils import get_auth_token, get_endpoint
import sys

# dohvacanje tokena za autentifikaciju korsnika
auth_token = get_auth_token()
headers = {'X-Auth-Token': auth_token}

# dohvacanje URL-a(endpointa) neutron servisa
neutron_endpoint = get_endpoint("neutron", auth_token)

# trazi unos naziva mreze od korisnika
name = raw_input('\nEnter exact network name: ')

# dohvaca id mreze cije ime je korisnik unio
r = requests.get(neutron_endpoint + "/v2.0/networks?name=" + name + "&fields=id", headers=headers)
print("Finding network...")
print(r.status_code, r.reason)
results_json = r.json()

# provjera ima li rezultata pretrage, ako ih nema rezults_json ce biti prazan (len ce biti nula)
#ako rezultata pretrage postoji spremamo ga u varijablu network_id
if len(results_json["networks"]) > 0:
    net_dict = results_json['networks'][0]
    network_id = net_dict['id']
    print('Found network %s with id %s' % (name, network_id))
else:
    sys.exit("Network not found.")

# dohvacanje id-ja javne mreze te id-ja subneta javne mreze
r = requests.get(neutron_endpoint + "/v2.0/networks?name=admin_floating_net", headers=headers)
print(r.status_code, r.reason)
results_json = r.json()

# provjera ima li rezultata pretrage, ako ih nema rezults_json ce biti prazan (len ce biti nula)
#ako ima rezultata pretrage spremamo ih u varijable public_network_id i public_subnet_id
if len(results_json["networks"]) > 0:
    net_dict = results_json['networks'][0]
    public_network_id = net_dict['id']
    public_subnet_id = net_dict['subnets'][0]

    print('Found public network with id %s' %  public_network_id)
    print ('Found public subnet id %s' % public_subnet_id)
else:
    sys.exit("Public network not found.")
```

Slika 7. Dohvaćanje potrebnih parametara za kreaciju rutera i porta - Prvi dio skripte Skripta 4.py

Na početku skripte Skripta 4.py dohvaćamo autentifikacijski token te endpoint neutron servisa na isti način kao i u skripti Skripta 3.py. Zatim od korisnika tražimo unos naziva privatne mreže koju preko rutera želi povezati s javnom mrežom. Uneseno ime koristimo za dohvaćanje ID-ja privatne mreže putem GET zahtjeva. Get zahtjev izvršava se u jednoj liniji koda te se ime mreže definira kao string unutar URL-a na koji šaljemo GET zahtjev (linija 21). U zaglavlju GET zahtjeva prosljeđujemo autentifikacijski token. Na linijama 28-33 provjeravamo je li mreža uspješno pronađena te ako je spremamo njen ID. Zatim slijedi dohvaćanje ID-ja javne mreže te ID-ja podmreže javne mreže. Dohvaćanje se vrši po istom principu kao i dohvaćanje ID-ja privatne mreže samo što je ime mreže u GET zahjevu eksplicitno navedeno kao „admin_floating_net“ jer je to jedina javna mreža u našem okruženju. Nakon što smo dohvatili sve potrebne parametre možemo prijeći na drugi dio skripte, a to je kreiranje rutera i porta pomoću dohvaćenih parametara. Drugi dio skripte prikazan je na slici 8.

```
52  #trazimo od korisnika unos imena rutera koji cemo kreirati
53  router_name = raw_input('\nEnter new router name: ')
54
55  #specifikacija zaglavlja
56  headers = {
57      'X-Auth-Token': auth_token,
58      'Content-Type': 'application/json'
59  }
60
61  #specifikacija podataka o novom ruteru, naziv rutera, id javne mreze, fiksna javna adresa
62  #id podmreze
63  data = {
64      "router": {
65          "name": router_name,
66          "external_gateway_info": {
67              "network_id": public_network_id,
68              "enable_snat": "true",
69              "external_fixed_ips": [
70                  {
71                      "ip_address": "10.30.2.171",
72                      "subnet_id": public_subnet_id
73                  }
74              ]
75          },
76          "admin_state_up": "true"
77      }
78  }
79
80  #slanje post zahtjva za kreaciju novog rutera servisu neutron
81  r = requests.post(neutron_endpoint + "/v2.0/routers", headers=headers, data=json.dumps(data))
82  print "Creating router..."
83  # ispis statusnog koda o kreaciji te spremanje ID-ja rutera iz odgovora dobivenog od neutrona
84  print(r.status_code, r.reason)
85  results_json = r.json()
86  router_id = results_json["router"]["id"]
87
88
89  #specifikacija podataka za kreiranje novog porta
90  #id privatne mreze, fiksna ip adresa
91  data = {
92          'port': {
93              'admin_state_up': True,
94              'network_id': network_id,
95              'fixed_ips': [{"ip_address": "10.20.0.126"}]
96          }
97  }
98
99  #slanje post zahtjeva za stvaranje porta servisu neutron
100 r = requests.post(neutron_endpoint + "/v2.0/ports", headers=headers, data=json.dumps(data))
101 print "Creating port..."
102 print(r.status_code, r.reason)
103 results_json = r.json()
104 #provjera uspjesnosti kreacije porta i ispis poruke
105 #dohvacanje id-ja novokreiranog porta koji ce se kasnije dodijeliti ruteru
106 if len(results_json["port"]) > 0:
107     net_dict = results_json['port']
108     port_id = net_dict['id']
109     print('Created port %s with id %s' % (name, port_id))
```

Slika 8. Kreiranje rutera i porta te dodjeljivanje porta ruteru – Drugi dio skripte Skripta 4.py

U drugom dijelu od korisnika tražimo upis naziva novog rutera koji demo kreirati (linija 53). Specificiramo zaglavlja i sadržaj koje ćemo prenijeti servisu neutron putem POST zahtjeva. Putem sadržaja (varijabla „data“) prosljeđujemo naziv rutera, ID javne mreže, parametar kojim omogućujemo SNAT, fiksnu javnu adresu te ID podmreže javne mreže. Na neutronov endpoint (URL) dodajemo "/v2.0/routers" čime dobivamo potpunu adresu na koju prosljeđujemo zahtjev za kreiranje novog rutera. Nakon kreacije rutera spremamo njegov ID jer ćemo ga kasnije koristiti za dodavanje porta. Prije dodavanja, port je potrebno kreirati. Na liniji 91 specificiramo podatke o portu, točnije ID privatne mreže te fiksnu IP adresu. Specificirane podatke predajemo, na liniji 100, servisu neutron. Neutronovom endpointu (URL-u) dodajemo "/v2.0/ports" budući da se radi o kreaciji porta. Ukoliko je port uspješno kreiran spremamo njegov ID te ispisujemo poruku o uspješnoj kreaciji. Spremljeni ID koristi ćemo u idućem koraku, a to je dodjeljivanje porta prethodno kreiranom ruteru. Dodjeljivanje porta ruteru prikazano je na slici 9.

```
117  #specifikacija id-ja porta koji dodjeljujemo ruteru
118  data = {
119          "port_id":port_id
120  }
121
122  #slanje put zahtjeva za dodavanje porta ruteru
123  r = requests.put(neutron_endpoint + "/v2.0/routers/"+router_id+"/add_router_interface", headers=headers, data=json.dumps(data))
124  print "Adding interface to router ..."
125
126  #ispis statusnog koda o dodavanju porta ruteru
127  print(r.status_code, r.reason)
128  results_json = r.json()
129  print json.dumps(results_json, indent=4)
```

Slika 9. Dodjeljivanje porta ruteru – Tredi dio skripte Skripta 4.py

6. Skripta 5.py

U skripti 5 koristimo sve resurse stvorene u prethodnim vježbama kako bi kreirali novu instancu.

Radi preglednosti skripta 5 podijeljena je na tri dijela. U prvom dijelu dohvaćamo endpointe tri OpenStack servisa (nova, glance i neutron). Prvo koristimo glance endpoint i šaljemo get zahtjev na adresu glance_endpoint + "/v2/images" kako bi korisniku izlistali sve ponuđene slike OS-a. Zatim tražimo od korisnika unos imena one slike OS-a koju želi koristiti prilikom kreacije nove instance. Zatim koritimo ime te ga prosljeđujemo na adresu glance_endpoint+"/v2/images?name=" + name te iz odgovora dohvaćamo ID odabrane slike OS-a. ID ćemo koristiti kasnije prilikom stvaranja instance. Prvi dio skripte 5 prikazan je na slici 10.

```
import requests
import json
from openstack_api_utils import get_auth_token, get_endpoint
import sys

# dohvacanje tokena za autentifikaciju korisnika sa pomocnom funkcijom
auth_token = get_auth_token()
# dodavanje auth tokena u zaglavlje koje ce se poslati
headers = {'X-Auth-Token': auth_token}

# iduce tri linije su pomocne funkcije za dohvacanje URL-a (endpointa) svih potrebnih servisa
# ove funkcije su uvezene iz pomocne knjiznice
nova_endpoint = get_endpoint("nova", auth_token)
glance_endpoint = get_endpoint("glance", auth_token)
neutron_endpoint = get_endpoint("neutron", auth_token)

print "List of all images by name and size:"

# slanje zahtjeva na glance url da se dohvate sve slike
r = requests.get(glance_endpoint + "/v2/images", headers=headers)
# ispis statusnog koda koji je glance servis vratio
print(r.status_code, r.reason)
# pretvaranje rezultata u json format
json_data = r.json()

# ispis svih slika koje smo dobili
for image in json_data["images"]:
    # ljust i rjust poravnavaju ispis u lijevo i u desno kako bi
    # ga korisnik mogao lakse citati
    print repr(image["name"]).ljust(50) + repr(image["size"]).rjust(15)

# trazenje unosa preko terminala od strane korisnika
name = raw_input('\nEnter image name to be used for the instance: ')

# pitaj korisnika za unos imena po kojem se pretrazuje
print('\nLooking for %s...\n' % name)

# dohvacanje specificne slike putem njezinog imena koje je korisnik unio
r = requests.get(glance_endpoint+"/v2/images?name=" + name, headers=headers)
# pretvaranje rezultata u json format
results_json = r.json()
images = results_json["images"]
# provjera ima li rezultata naseg upita
if images:
    # ako ima ispisi sve slike koje su proandjene
    for image in images:
        print('Images found, id is:%s' % image["id"])
        image_id = image["id"]
# ako nema ispisi poruku i zavrsi izvrsavanje programa
else:
    # izvrsava se ukoliko nema rezultata pretrage
    sys.exit("Image Not Found")
```

Slika 10. Prvi dio skripte Skripta 5.py- Dohvaćanje ID-a slike OS-a

U drugom dijelu skripte tražimo od korisnika unos imena mreže koju želi koristiti te na temelju imena dohvaćamo ID mreže. ID ćemo koristiti kasnije prilikom stvaranja instance. Zahtjev za dohvat ID-a šalje se get zahtjevom na adresu neutron_endpoint + "/v2.0/networks?name=" + name + "&fields=id".

Također, u drugom dijelu, koristimo endpoint servisa nova kako bi korisniku izlistali flavor-e koje može odabrati za svoju novu instancu. Nakon što korisnik odabere flavor dohvaća se id flavora te se on sprema za kasnije.
Zahtjev za izlistavanjem falvora šalje se na adresu nova_endpoint + "/flavors" .
Drugi dio skripte 5 prikazan je na slici 11.

```
60  # trazi unos naziva mreze od korisnika
61  name = raw_input('\nEnter exact network name: ')
62
63  # dohvaca id mreze cije ime je korisnik unio
64  r = requests.get(neutron_endpoint + "/v2.0/networks?name=" + name + "&fields=id", headers=headers)
65  print("Finding network...")
66  # ispis statusnog koda koji nam vrati neutron servis
67  print(r.status_code, r.reason)
68  results_json = r.json()
69
70  # provjera ima li rezultata pretrage, ako ih nema rezults_json ce biti prazan (len ce biti nula)
71  if len(results_json["networks"]) > 0:
72      # ako postoje rezultati pretrage dohvati prvi i spremi ga u rjecnik (dictionary)
73      net_dict = results_json['networks'][0]
74      network_id = net_dict['id']
75      # ispis podataka o pronadjenoj mrezi
76      print('Found network %s with id %s' % (name, network_id))
77  else:
78      # u slucaju da nema rezultata pretrage ispisuje se odgovarajuca poruka i zavrsava se program
79      sys.exit("Network not found.")
80
81
82
83
84
85  # slanje GET zahtjeva za popisom dostupnih flavora
86  r = requests.get(nova_endpoint + "/flavors", headers=headers)
87  print(r.status_code, r.reason)
88  json_data = r.json()
89
90  # ispis liste svih dostupnih flavora sa formatiranje da bi bilo citljivije
91  for flavor in json_data["flavors"]:
92      print repr(flavor["id"]).ljust(40) + repr(flavor["name"]).rjust(15)
93
94
95  # trazenje unosa od korisnika
96  flavor_id = raw_input('\nEnter flavor ID: ')
97  # slanje GET zahtjeva da se dohvate podaci o flavoru sa odredjenim ID-om
98  r = requests.get(nova_endpoint + "/flavors/" + flavor_id, headers=headers)
99  print(r.status_code, r.reason)
100 if r.status_code != 200:
101     # ako se dobije statusni kod razlicit od 200 (OK) znaci da se flavor nije
102     # uspio dohvatiti te se ispisuje poruka i zavrsava program
103     sys.exit("Invalid flavor id")
```

Slika 11. Drugi dio skripte Skripta 5.py- Odabir mreže i flavor-a

U trećem dijelu skripte 5 tražimo od korisnika unos imena nove instance te sve dosad prikupljene parametre te uneseno ime prosljeđujemo putem post zahtjeva na adresu nova_endpoint + "/servers". Na toj adresi servis nova prima zahtjeve za stvaranje nove instance. Nakon poruke o uspješnosti kreacije nove

instance još jednom koristimo nova endpoint kako bi izlistali sve aktivne instance. Treći dio skripte 5 prikazan je na slici 12.

```
# trazenje unosa od korisnika, ako unese prazan string onda ga se pita da unese opet
instance_name = raw_input('\nEnter name for new instance: ')
while instance_name == "":
    print "Name cannot be empty\n"
    instance_name = raw_input('\nEnter name for new instance: ')

# podaci za kreiranje nove instance, sastoje se od podataka koje je korisnik unio
data = {
        "server": {
                "name": instance_name,
                "imageRef": image_id,
                "flavorRef": flavor_id,
                "networks": [
                    {
                        "uuid": network_id
                    }
                ],
                "availability_zone": "nova",
                "security_groups": [
                    {
                        "name": "default"
                    }
                ]
            }
}
# u zaglavlje je potrebno dodatio Content Type jer se salju
# podaci pa je potrebno definirati kojeg su tipa
headers = {
    'X-Auth-Token': auth_token,
    'Content-Type': 'application/json'
}

# slanje POST zahtjeva za kreiranje nove isntance sa gore definiranim podacima
# json.dumps(data) je potrebno napraviti kako bi se osiguralo da su poslani podaci
# u json formatu
r = requests.post(nova_endpoint + "/servers", headers=headers, data=json.dumps(data))
print("Creating instace...")
print(r.status_code, r.reason)
if r.status_code != 202:
    sys.exit("Unable to create instace")
else:
    print "Instance created successfully"

# slanje zahtjeva da se izlistaju sve instance kako bi mogli vidjeti je li
# kreirana nasa nova instanca
r = requests.get(nova_endpoint + "/servers", headers=headers, data=json.dumps(data))
json_data = r.json()
for server in json_data["servers"]:
    print server["name"]
```

Slika 12. Treći dio skripte 5- Odabir imena instance , kreiranje nove instance na temelju svih prikupljenih parametara te izlistavanje svih aktivnih instanci