

**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.



Slika 1. Dohvaćanje autentifikacijskog tokena

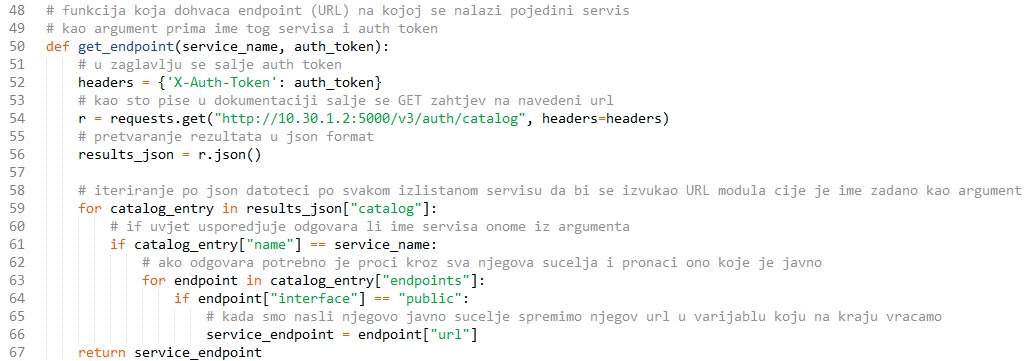
1.3 Dohvaćanje endpointa (URL-a) servisa

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.



Slika 2. Dohvaćanje endpointa (URL-a servisa)

1. **Skripta 1.py**

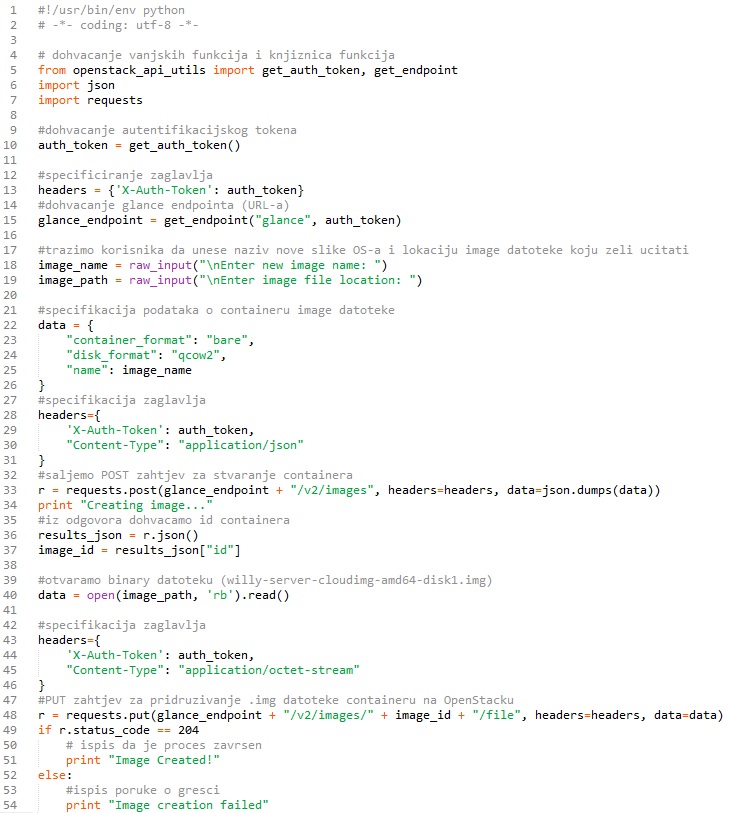
Na početku skripte Skripta 1.py dohvaća se token za autentifikaciju te glance endpoint (URL). Glance je OpenStack servis koji omogućava dodavanje i uređivanje slika OS-a. Naziv spremnika, kao i putanju do .img datoteke korisnik unosi kroz terminal. POST zahtjevom kreira se spremnik (engl. container) 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.



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

1. **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.



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

1. **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.



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.



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.

1. **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.



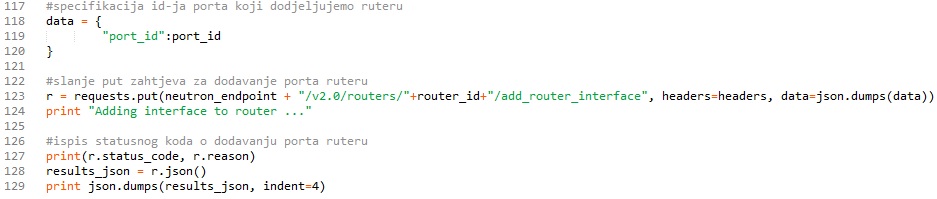
Slika 7. Dohvaćanje potrebnih parametara za kreaciju rutera i porta - Prvi dio skripte Skripta 4.py

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.



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.

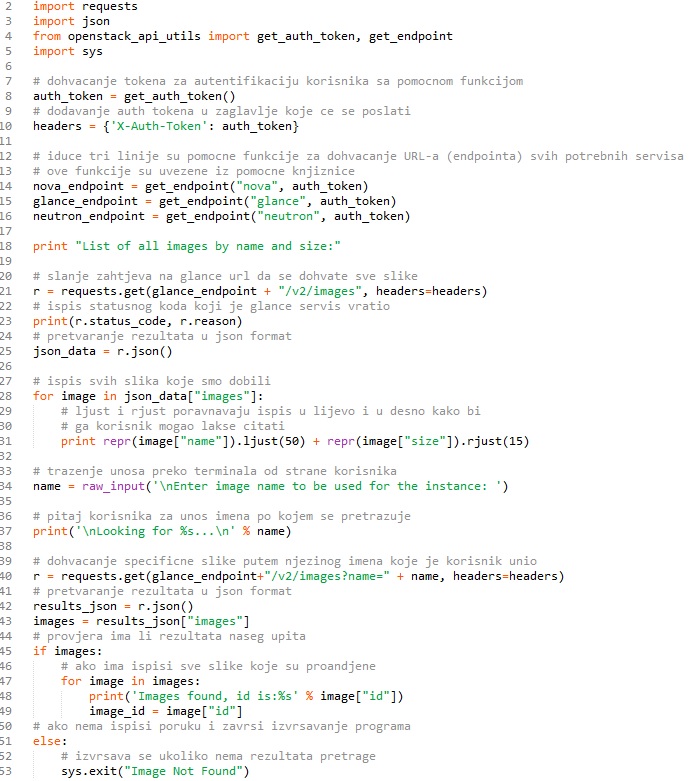


Slika 9. Dodjeljivanje porta ruteru – Tredi dio skripte Skripta 4.py

1. **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.



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.



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.



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