UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Damjan Pipan

# Vaje za učenje OpenNebule v oblaku

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Mojca Ciglarič

Ljubljana, 2012

# I Z J A V A  O  A V T O R S T V U

## diplomskega dela

Spodaj podpisani/-a Damjan Pipan,

z vpisno številko 63020331,

sem avtor diplomskega dela z naslovom:

Vaje za učenje OpenNebule v oblaku

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom (naziv, ime in priimek)

  doc. dr. Mojca Ciglarič

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja: _____

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

*Tržaška 25*
*1001 Ljubljana, Slovenija*
*telefon: 01 476 84 11*
*01 476 83 87*
*faks: 01 426 46 47*
*01 476 87 11*
*www.fri.uni-lj.si*
*e-mail: dekanat@fri.uni-lj.si*

Št. naloge: 01798/2012
Datum: 02.02.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DAMJAN PIPAN**

Naslov: **VAJE ZA UČENJE OPENNEBULE V OBLAKU**
**EXERCISES FOR LEARNING OPENNEBULA IN A CLOUD**

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Pojasnite osnovne pojme v zvezi z računalništvom v oblaku. Opišite virtualni laboratorij v oblaku, ki ga uporablja Laboratorij za računalniške komunikacije za pedagoške namene. Preučite, kako bi se lahko virtualni laboratorij uporabil za učenje o postavljanju oblaka. Preverite možnosti gnezdene virtualizacije. V okviru možnosti virtualnega laboratorija pripravite serijo vaj, ki bi jih bodoči administratorji oblaka lahko uporabili za učenje oblačne platforme OpenNebula, pri čemer bi za učenje uporabili virtualno oblačno infrastrukturo znotraj obstoječega virtualnega laboratorija. Za vsako vajo pripravite ustrezne slike virtualnih računlanikov, vaje ustrezno pojasnite in opremite z navodili, v zaključku pa komentirajte prednosti takšnega načina učenja.

Mentor:
M. Ciglarič
doc. dr. Mojca Ciglarič

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

# Zahvala

Za mentorstvo, posvečen čas in nasvete pri izdelavi diplomskega dela se zahvaljujem mentorici doc. dr. Mojci Ciglarič. Zahvala tudi Mihi Groharju za pomoč pri delu v laboratoriju.

Zahvaljujem se moji družini, ker so mi omogočili študij v tujini, me med njim podpirali in verjeli vame. Še posebej se zahvaljujem moji punci Ines za spodbudo in potrpežljivost v času pisanja diplomskega dela.

Na koncu se zahvaljujem tudi vsem prijateljem in sošolcem, ki so mi med študijem stali ob strani.

# Kazalo vsebine

# Kazalo slik

# Seznam uporabljenih kratic in simbolov

**IaaS** (ang. Infrastructure as a Service) - storitveni model oblaka, ki ponuja infrastrukturo

**PaaS** (ang. Platform as a Service) - storitveni model oblaka, ki ponuja platformo na kateri lahko izvajamo aplikacije

**SaaS** (ang. Software as a Service) - storitveni model oblaka, ki ponuja programsko opremo

**NFS** (ang. Network File System) - Mrežni datotečni sistem

**iSCSI** (ang. Internet Small Computer System Interface) - vmesnik za povezovanje pomnilniških naprav, ki uporablja TCP/IP protokol

**VCL** (ang. Virtual Computing Lab) - Odprtokodna programska oprema, ki omogoča oskrbovanje z računalniški viri

**EC2** (ang. Elastic Compute Cloud) - Amazonova spletna storitev, ki nudi razširljivo računalniško infrastrukturo v javnem oblaku

**API** (ang. Application programming interface) - Programski vmesnik, ki zagotavlja, da ima računalniški program na razplago funkcije drugega računalniškega programa

**SSH** (ang. Secure Shell) - varen protokol za upravljanje računalnika na daljavo

**OCCI** (ang. Open Cloud Computing Interface) - vmesnik za upravljanje s storitvami oblaka

**IM** (ang. Information Manager driver) - gonilnik za upravnika informacij

**VM** (ang. Virtualization Manager driver) - gonilnik za upravnika virtualizacije

**TM** (ang. Transfer Manager driver) - gonilnik za upravnika prenosa

**VNC** – (angl. Virtual Network Computing) - grafični sistem za oddaljen dostop do računalnikov

# Povzetek

Računalništvo v oblaku je zelo priljubljena tema zadnjih nekaj let. Razširilo se je v skoraj vse veje računalništva. Zaradi vse večje popularnosti in hitrega razvoja se je veliko podjetij že odločilo za spremembo načina ponudbe svojih storitev. Danes obstaja veliko orodij, ki sistemskim administratorjem pomagajo pri izdelavi oblakov. Težave se pojavijo pri učenju o postavljanju oblaka. V večini primerov je težko priti do ustrezne infrastrukture, na kateri se lahko administrator uči postavljanja ali testiranja takega okolja.

Kot rešitev se lahko uporabi virtualizirano okolje, na katerem se izvedejo vaje za učenje. Eno izmed njih je Virtual Computing Lab (VCL), katerega uporablja Laboratorij za računalniške komunikacije. Eden izmed ciljev te diplomske naloge je preučiti možnosti uporabe virtualnega laboratorija za učenje postavljanja oblaka. V nadaljevanju se bo pripravila serija vaj s podrobnim opisom postopka namestitve orodja OpenNebula na virtualni infrastrukturi ter njegova uporaba za izdelavo in nadzorovanje oblaka.

Pred samo namestitvijo orodja je potrebno preveriti, ali infrastruktura ustreza zahtevam. Poleg tega je potrebno še ustrezno nastaviti določena orodja, ki bo OpenNebula uporabljala pri delovanju. Ko je infrastruktura pripravljena, se lahko namesti orodje in izdelovanjem oblaka se lahko začne. Za večjo preglednost in lažje razumevanje je celoten postopek razdeljen na več vaj, ki, poleg podrobnih navodil, vsebujejo tudi teoretično osnovo in delujoče testne vzorce.

# Ključne besede

Računalništvo v oblaku, OpenNebula, VCL, Privatni oblak, Vaje

# Abstract

Cloud computing is a very popular topic over the last few years. It has expanded in almost all branches of the computer science. Due to the increasing popularity and rapid development, many companies have already decided to change the way of providing their own services. Today, there are many tools that help the system administrators in making clouds. The problem starts in learning about setting up a cloud. In most cases, it is difficult to get adequate infrastructure which could be used by administrators to learn and test that environment.

As a solution, a virtualized environment could be used for carrying the exercises. One of them is the Virtual Computing Lab (VCL), used by the Computer communications lab. One of the goals of this thesis is to examine the use of virtual lab for learning how to set up a cloud. After that, a series of exercises will be prepared, with detailed descriptions of the OpenNebula installing process on a virtual infrastructure and its use for creating and managing the cloud.

Before the installation of the tool, it should be checked if the infrastructure meets the requirements. Moreover, it is required to set up certain tools and libraries, which will be used by OpenNebula. Once the infrastructure is prepared, the tool can be installed and setting up the cloud may begin. In order to increase transparency and achieve a better understanding, the whole process is divided into several exercises. Besides the detailed instructions, they contain the theoretical basis and functional test templates.

# Keywords

Cloud computing, OpenNebula, VCL, Private cloud, Exercises

# 1 Uvod

Način upravljanja s strojno in programsko opremo se je zelo spremenil v zadnjih nekaj letih. Zaradi tega se danes v računalništvu velikokrat naleti na besedo »oblak« ali »računalništvo v oblaku«.

V današnjem času podjetja uporabljajo oblake za ponujanje lastnih javnih storitev ali za zasebne namene. Primerov uporabe oblaka je veliko, od uporabe elektronske pošte in spletnih aplikacij do najema infrastrukture. Uporablja se ga tudi za izvajanje bolj zahtevnih operacij, kjer lokalna infrastruktura ni dovolj zmogljiva, ali pa v industriji računalniških iger.

Zaradi vseh prednosti, ki jih prinaša uporaba oblaka, se veliko podjetij odloča za spremembo svojih sistemov. Eni se za oblak odločajo zaradi lažje ponudbe svojih storitev, drugi zaradi lažjega dela znotraj posameznih skupin. Podjetja, katera imajo lastno infrastrukturo, se lahko odločijo, da jo uporabijo za gostovanje oblaka. Sam postopek izdelave zahteva poznavanje delovanja oblaka ter ustrezno nastavitev strežnikov, ki jih bo uporabljal oblak.

Poleg priprave infrastrukture in nastavljanja strežnikov je potrebno tudi določiti uporabnike in njihove pravice. Izkušeni sistemski administratorji ne bodo potrebovali veliko časa za pridobitev potrebnega znanja za nameščanje celotnega sistem, vendar se lahko srečajo z drugimi težavami, kot so recimo neustrezna infrastruktura ali nezaupanje v delovanje novega sistema s strani uporabnikov. Problem imajo tudi manj izkušeni administratorji ali začetniki, ki bi se tega radi naučili, vendar nimajo ustrezne infrastrukture ali dovolj jasnih virov za učenje.

Kot rešitev tem problemom se lahko uporabi virtualni laboratorij, ki se lahko prilagodi potrebam uporabnikov. Določeni skupini uporabnikov se dodelijo navidezni strežniki, ki jih le-ta lahko uporabi za učenje postavljanja oblaka.

Pri učenju izdelovanja oblaka obstaja več različnih pristopov. Izkušeni sistemski administratorji se v večini primerov odločajo za pregled dokumentacije ter se znajdejo z uporabo lastnega znanja in izkušenj.

Problemi se ponavadi pojavijo pri neizkušenih uporabnikih ali recimo študentih, ki nimajo predznanja. Branje dokumentacije v tem primeru lahko predstavlja problem, ker uporabnik sploh ne razume delovanja sistema, ne zna odpraviti morebitne napake ali ne more slediti navodilom.

Zaradi tega se je pojavila potreba za izdelovanje vaj, ki bodo začetnikom omogočale enostavnejše učenje z opisanimi postopki in koraki ter preverjenimi primeri.

Celoten postopek učenja, kako izdelati oblak je opisan v sklopu vaj. Vaje vsebujejo teoretično osnovo, ki opisuje delovanje oblaka, potrebno infrastrukturo in orodja, ki se lahko uporabljajo. Poleg teorije vsebujejo tudi natančna navodila, kako nastaviti strežnike in namestiti ustrezna orodja ter jih učinkovito uporabljati.

# 2 Teoretična razlaga pojmov in tehnologij

## 2.1 Računalništvo v oblaku

Računalništvo v oblaku uporabnikom ponuja računanje, programsko opremo, dostop in shranjevanje podatkov ter podobne storitve prek spleta. Te storitve so dostopne od kjerkoli, koderkoli ter v skoraj neomejeni količini. Ni jim potrebno vedeti, kje se strežniki fizično nahajajo ali kakšne so nastavitve sistemov, ki jim te informacije ponujajo. Podobno logiko se lahko najde pri električnih omrežjih, katera dostavljajo elektriko različnim gospodinjstvom, kjer uporabniki ne potrebujejo informacij, kaj to elektriko proizvaja ali na kakšen način je dostavljena.

Fizično oblak tvorijo medsebojno povezani centri, kateri vsebujejo veliko fizičnih strežnikov. Strežniki v mnogih primerih poganjajo virtualne stroje (navidezne vire), ki so ponujeni končnemu uporabniku. Tak koncept je podoben scenariju, ko podjetja uporabljajo navidezne stroje za boljši izkoristek strojne opreme. Razlika je v obsežnosti celotnega sistema (pri oblakih obstaja občutek neskončnosti virov), ceni (ni potrebe po investiranju v strojno opremo) ter časovni porabi virov (lahko se porabi več tisoč procesorjev le za dobo nekaj ur) [1].

Osnovni storitveni modeli oblaka so:

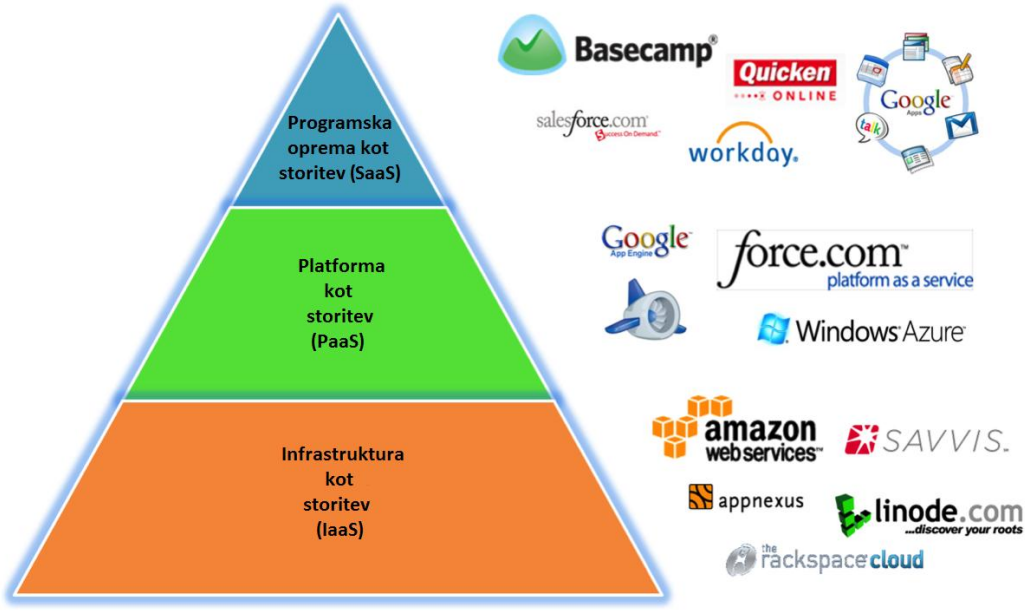
- Programska oprema kot storitev (ang. Software as a Service – SaaS)
- Platforma kot storitev (ang. Platform as a Service – PaaS)
- Infrastruktura kot storitev (ang. Infrastructure as a Service – IaaS)

Programsko plast se uporablja, ko uporabnik dostopa do storitve brez informacije, na katerem računalniku se aplikacija izvaja. Pri tem uporabniku ni potrebna namestitev ali zagon aplikacije. Za namestitev, nastavitev in vzdrževanje te programske opreme skrbi ponudnik (kot primer se lahko omeni Gmail). Prednosti takega dostopa so zaračunavanje glede na uporabo aplikacij ter zelo nizki stroški vzdrževanja.

Pri plasti Platforma kot storitev se posreduje operacijski sistem ali navidezni stroj z določeno procesorsko močjo, pomnilnikom in prostorom. Vsak primerek je popolnoma izoliran od ostalih, ki delujejo na isti strojni opremi, kjer se viri enostavno razporedijo glede na potrebe. Uporabnik

lahko razvija lastne aplikacije brez potrebe po nakupu in vzdrževanju lastne programske ali strojne infrastrukture. Primera v tej plasti sta Google App Engine in Microsoft Azure Services Platform.

Infrastruktura kot storitev je »najnižja« plast. Zahteva namestitev več različnih komponent (kot recimo procesorski čas, pomnilnik, diskovni prostor, omrežje, uporabniški dostop). Uporabniku omogoči popolno zunanje izvajanje (*angl. Outsourcing*) IT infrastrukture. Operacijski sistem, aplikacije in strežnik lahko uporablja na enak način, kot bi jih imel doma ali v podjetju, vendar lahko zakupljene kapacitete spreminja glede na potrebe. Primer je Amazonov Elastic Compute Cloud (EC2) [2,3].




**Slika 1: Osnovni storitveni modeli oblaka**

Razen osnovnih storitvenih modelov obstajajo tudi Hardware as a Service (HaaS), Computing as a Service (CaaS), Storage as a Service (STaaS), Aplication as a Service (AaaS) itn.

## 2.2 O orodju OpenNebula

Obstajajo primeri, ko se uporabnik ne more nanašati na zunanje ponudnike oblakov ali so posamezne storitve predrage. Takrat se lahko uporabi OpenNebula, odprto-kodno orodje za izdelavo privatnih, javnih in hibridnih oblakov, prilagodljivo v različnih okoljih.

Za razliko od ostalih odprto-kodnih orodij le-ta ne zajema le določenega hipervizorja. Nima posebnih infrastrukturnih zahtev ter se dobro prilagaja obstoječim okoljem, omrežjem, diskovnim prostorom ali pravilom uporabniškega upravljanja (ang. *user-management policies*).

OpenNebula je implementirana na način, ki sistemskim oskrbnikom omogoči, da prilagodijo vse vidike, vključno z virtualizacijo, skladiščenjem, informacijo, avtentikacijo, avtorizacijo in oddaljenimi storitvami oblaka. Celotno delovanje je upravljano s strani t.i. »*bash*« skript, ki se lahko nastavijo ali dodajo drugim skriptam ali programski opremi v kateremkoli jeziku in operacijskem sistemu.

Za komunikacijo med OpenNebulo in strežnikom, na katerem se nahaja, skrbijo gonilniki. Obstajajo trije tipi gonilnikov:

- Gonilniki za prenos (ang. *Transfer drivers*): Upravljajo s slikami na sistemu za skladiščenje ki je lahko v skupni (NFS ali iSCSI) ali zasebni rabi (navadno kopiranje čez SSH protokol).
- Gonilniki za virtualne stroje (ang. *Virtual Machine drivers*): Odvisni so od hipervizorja in so uporabljeni za upravljanje z virtualnimi stroji na strežniku.
- Gonilniki za informacije (ang. *Information drivers*): Uporabljeni so za pridobitev trenutnega stanja virtualnih strojev in strežnikov. Prav tako so odvisni od hipervizorja in delajo na vsakemu fizičnemu strežniku čez SSH protokol

Vse informacije zbrane s fizičnih strežnikov in virtualnih strojev ter vse nastavitve vsakega virtualnega stroja, kataloga dostopnih slik in omrežja, so shranjene v podatkovni bazi. Tam se jih lahko spreminja s pomočjo skript ali aplikacij.

Če je potrebno, lahko uporabnik uporabi API vmesnik orodja OpenNebula, s katerim dostopa do vseh funkcij, ki so mu na voljo ter lahko naredi lastne procedure. Na voljo je tudi »*hook«* sistem, ki uporabnikom omogoča poganjanje skript po določenih dogodkih. Primer bi bilo obveščanje po elektronski pošti pri spremembi nastavitev ali obremenitvi sistema, lahko pa se tudi izvaja ponovni zagon ali sprememba virov[4].

Poskrbljeno je tudi za dobro varnost. Strežniki komunicirajo izključno s pomočjo varnih povezav, zaščitenih z SSH RSA ključi in SSL plastjo. Vsako virtualno omrežje znotraj oblaka je lahko izolirano s požarnim zidom (npr. *ebtables*) [5,6].

Z OpenNebulo se lahko naredijo trije tipi oblakov, ki se razlikujejo po načinu uporabe.

Prvi je zasebni oblak. Pri njem ponavadi ne obstaja poseben API vmesnik in vsi viri so uporabljeni za notranje, zasebno delo (znotraj določene množice ljudi). Kot primer se lahko uporabi manjše podjetje, laboratorij ali skupina ljudi.

Naslednji je javni oblak. Za razliko od zasebnega oblaka so v javnem oblaku viri dostopni zunanjim uporabnikom čez prej definirani API vmesnik.

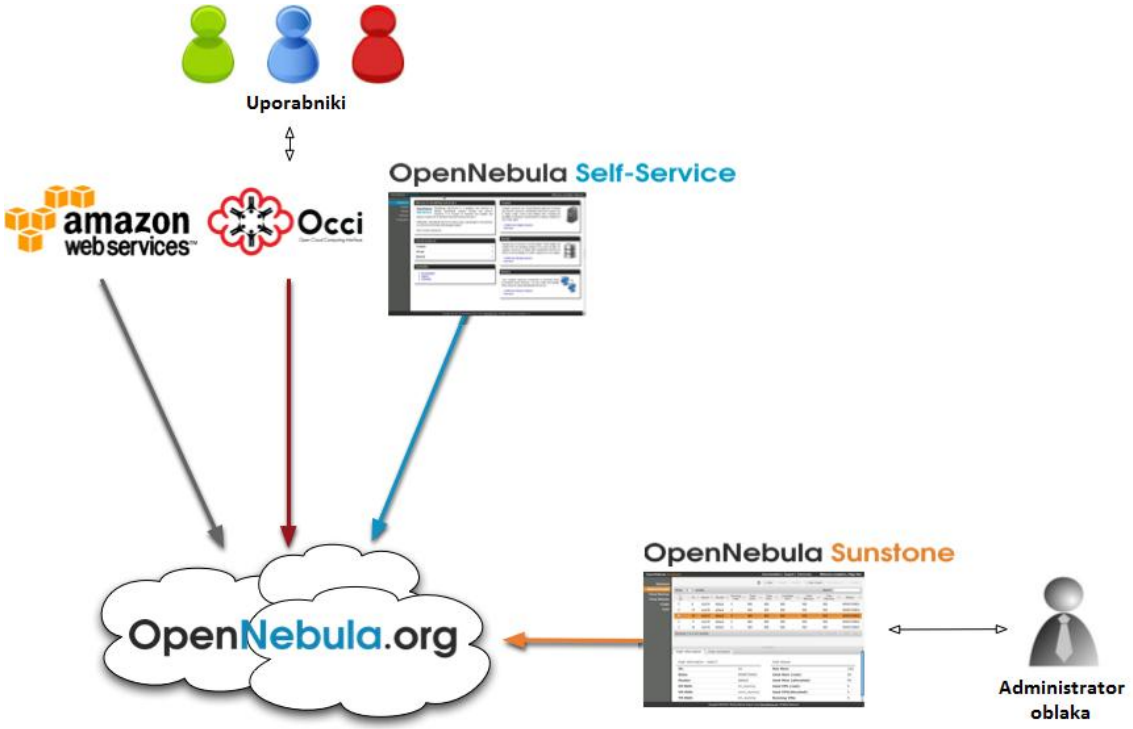
V primeru, ko se za izboljšavo lastnega oblaka uporabljajo zunanji viri ali uporabniki ponujajo lastne vire zunanjim uporabnikom (tretjim osebam), se govori o hibridnem oblaku.

Pri delu z oblaki se pogosto sreča z naslednjimi značilnostmi:

- upravljanje z uporabniki: Lahko se nastavi več uporabnikov in njihov dostop, ali omeji vire glede na posamezni uporabniški račun
- upravljanje s slikami virtualnih strojev: Vsaka slika diska je zabeležena in upravljana s centraliziranim katalogom slik
- upravljanje z virtualnim omrežjem: Znotraj oblaka se lahko definira več omrežij povezanih z različnimi fizičnimi vmesniki, s statičnim ali dinamičnim dodeljevanjem IP naslovov
- upravljanje z virtualnimi stroji: Vsak virtualni stroj ima lastne nastavitve (CPE, pomnilnik, diskovni prostor in virtualno omrežje) in se lahko poganja z različnih hipervizorjev
- upravljanje s storitvami: Skupino virtualnih strojev se lahko grupira da se naložijo istočasno, ter nastavi pri zagonu brez dodatnega dela na vsakemu primerku posebej
- upravljanje z infrastrukturo: Fizične strežnike lahko grupiramo v neodvisne gruče (koristno v heterogenem okolju)
- upravljanje s skladiščenjem: Podprto je veliko načinov skladiščenja v podatkovnih centrih kot so SAN (FibreChannel, iSCSI, AoE protokoli) ali NAS (NFS protokol)
- upravljanje z informacijami: Vsak strežnik in virtualni stroj so opazovani vsakih par sekund, lahko se jih tudi opazuje s pomočjo drugih orodij (Ganglia)
- upravljanje s časovnim razvrščanjem: Virtualni stroji so optimalno postavljene na strežnike glede na uporabniške zahteve in zahteve po virih
- uporabniški vmesnik: Vsebuje orodja za ukazno vrstico ki omogočajo upravljanje z vsako funkcijo OpenNebule (stanje gruče, stanje virtualnih strojev, shramba slik, ...)
- operacijski center: Večina informacij in nalog dostopnih čez ukazno vrstico je dostopna tudi prek spletnega vmesnika

Poleg zgoraj navedenih značilnosti, pri hibridnih oblakih imamo še naslednje:

- Cloud-bursting: Možnost da se lokalno infrastrukturo okrepi z močjo zunanjih virov če je v določenih trenutkih preveč obremenjena
- Zveza: Sposobnost kombiniranja različnih gruč, ki se nahajajo na različnih fizičnih lokacijah




**Slika 2: OpenNebula in njeni vmesniki**

Pri javnih oblakih je glavna značilnost izpostavljanje virov zunanjim uporabnikom s pomočjo enega ali več vmesnikov (primer OCCI in EC2 standardni API vmesniki)[7].

### Hipervizorji

Hipervizor je virtualizacijska platforma, ki podpira istočasno delovanje več gostujočih operacijskih sistemov in ponuja virtualizacijsko plast, ki vsakemu gostujočemu operacijskemu sisteme omogoča delovanje na istem fizičnem strežniku brez znanja o ostalih operacijskih sistemih. Vsak operacijski sistem je fizično zaščiten od ostalih operacijskih sistemov, nanj prav tako ne vpliva nestabilnost ali okvara drugih sistemov.

OpenNebula podpira tri tipe hipervizorjev: Xen, KVM in VMware.

Xen je bil nekaj let zapored vodilni odprtokodni hipervizor. Veliko podjetij ga še vedno uporablja danes. Najbolj znani so HP, IBM, Intel, AMD, Cisco, Dell, Fujitsu, Novell, Red Hat, Samsung in Citrix. Poleg uporabe kot hipervizor pri OpenNebuli, uporablja se tudi kot samostojen sistem, brez dodatnega Linux sistema (pri Amazon EC2, Linode in Rackspace Cloud) [8].

KVM (Kernel-based Virtual Machine) je bil direktno integriran v kernel izvire od leta 2007 in je bil dostopen v vsaki GNU/linux distribuciji od te točke naprej. Leta 2008 ga je kupil Red Hat in je danes aktivno vzdrževan s strani Lunux razvijalcev. Dizajn je drugačen kot pri Xenu. V bistvu

je samo vmesnik katerega se kliče prek posebne sistemske datoteke (gonilnike). Pri virtualizaciji uporablja Quick Emulator (QEMU). QEMU mu doda veliko značilnosti kot so podpora za serijski port, omrežno kartico, PCI-ATA vmesnik, USB krmilnik itn.

Tretji tip hipervizorjev. ki jih uporablja OpenNebula je VMware hipervizor. Obstaja več različic tega hipervizorja:

- VMware ESXi: brezplačen in najbolj enostaven hipervizor. Vsebuje samo vmesnik za ukazno vrstico in lasten kernel (ni Linux-ov). Podpira omejen/optimiziran seznam strojne opreme.
- VMware ESX: glavni pred ESXi. Vsebuje Java spletni vmesnik in je dostopen samo pod komercialno licenco.
- VMware Server: brezplačen odprto-kodni hipervizor, dostopen za namestitev na Linux in Windows operacijskih sistemih. Vsebuje isti Java spletni vmesnik kot ESX, ampak z nekaj manj lastnostmi.

VMware hipervizorji ponujajo večjo zmogljivost in so boljše integrirani z Windows virtualnimi stroji [9,10].

## 2.3 VCL – Virtualni laboratorij

Kot je že bilo omenjeno predstavlja največji problem pri učenju postavljanja oblaka pridobitev ustrezne infrastrukture. V podjetjih so strežniki ponavadi ves čas v uporabi in niso na voljo za »eksperimentiranje«, nabava novih zgolj za učenje pa predstavlja nepotreben strošek.

Zaradi teh razlogov je uporaba virtualnega laboratorija idealna rešitev, saj se z virtualizacijo lahko pripravi število navideznih strežnikov, ki so lahko na voljo tistim, ki se učijo.

VCL (ang. *Virtual Computing Lab*) je sistem za oskrbovanje z računalniškimi viri. Sestavljen je iz zasebnega oblaka, ki s pomočjo virtualizacijske tehnologije VMware omogoča oskrbovanje z virtualnimi stroji. Preko spletnega vmesnika uporabnikom omogoča rezervacije virov in administracijo sistema [11,12].

Razen virtualiziranega okolja ponuja VCL bolj enostaven pristop k izvajanju vaj. Kot je opisano v naslednjih poglavjih je celoten postopek učenja razdeljen na nekaj vaj. Za vsako vajo obstaja posnetek sistema, ki uporabnikom omogoča uporabo pripravljenega okolja glede na vajo, na kateri se nahaja. Hkrati omogoča vračanje na prejšnji korak v primeru, da se med izvajanjem vaje napake ne da odpraviti ali pride do sesutja celotnega sistema. Zaradi tega uporabniku ni potrebno začeti od znova, temveč lahko vaje nadaljuje tam, kjer je ostal.

Za izvajanje vaj iz te diplomske naloge je potrebno vnaprej pripraviti vsaj 2 strežnika za vsakega uporabnika. Prvi bo uporabljen kot čelni strežnik za OpenNebulo, drugi pa kot hipervizor.

# 3 Nastavitev OpenNebule in hipervizorjev

Celoten postopek učenja je razdeljen na 5 vaj. Ker gre za učenje je velika verjetnost, da uporabnik naredi kakšno večjo napako, ki je ne bo znal odpraviti. Zaradi tega so za vsako vajo pripravljeni posnetki sistema, ki omogočajo vračanje na začetek vaje ali preskok na naslednjo vajo. Posnetki so uporabni tudi v primeru, če uporabnik želi poskusiti kakšen drugačen način namestitve brez, da bi celotno vajo moral ponoviti od začetka.

Na začetku vsake vaje je kratek opis, zatem pa sledita postopek in zaključek. Med vajami se bo uporabnik najprej naučil, kako preveriti infrastrukturo, namestiti in nastaviti čelni strežnik in hipervizorje ter komunikacijo med njimi. Ko je okolje pripravljeno, se bo naučil, kako nastaviti določene komponente, ki tvorijo oblak. Poleg izdelave oblaka se bo na koncu tudi naučil, kako ta oblak nadzorovati ter ukrepati v primeru težav.

## 3.1 Vaja 1 – Preverjanje infrastrukture in namestitev OpenNebule

**Cilj vaje je uporabiti dani sistem kot čelni strežnik ter na njemu namestiti orodje OpenNebulo.** Naloge za 1. vajo so namestiti in nastaviti orodje OpenNebula ter nastaviti ustrezne gonilnike za delo s hipervizorji.

V prvi polovici vaje so opisane zahteve in postopek preverjanja, če se na obstoječi infrastrukturi lahko postavi oblak. Zahteve se nanašajo na oba strežnika (čelni in hipervizor) in je pomembno, da se uporabnik nauči preverjati določene komponente. Zatem sledi postopek namestitve orodja OpenNebula, katerega bo uporabnik uporabil na lastnem strežniku.

Uporabnikom je voljo na novo nameščen sistem, ki se ga bo uporabljalo kot čelni strežnik. Če uporabnik v postopku namestitve naredi večjo napako, se lahko znova naloži posnetek sistema.

### 3.1.1 Zahteve strojne in programske opreme

**Zahteve strojne opreme za hipervizorje**

**Centralna procesna enota**

Pri strežnikih je pomembno, da njihove centralne procesne enote (v nadaljevanju CPE) podpirajo virtualizacijo. Ta podatek se lahko preveri v datoteki */proc/cpuinfo*. Potrebno je paziti na zastavico *vmx* pri Intelovih procesorjih oziroma *svm* pri AMDjevih procesorjih.

Do istega podatka se lahko pride direktno z ukazom: *$ egrep '(vmx|svm)' /proc/cpuinfo|wc –l*

Če ukaz izpiše vrednost 0 pomeni, da CPE ne podpira virtualizacijo in je to potrebno omogočiti v BIOS nastavitvah. Če virtualizacijo podpira, ukaz vrne število CPE jeder z osnovno virtualizacijsko podporo.

**Pomnilnik**

Poleg virtualizacijske podpore je potrebno preveriti, ali imajo strežniki dovolj pomnilnika. To je predvsem odvisno od tipa in števila aplikacij ali sistemov, ki bodo v oblaku.

Pri manjših virtualnih strojih, ki imajo manj zahtevne spletne aplikacije, zadošča 256 MB po virtualnem stroju. Pri strojih srednje velikosti, ki bi naj bile za podatkovni strežnik z majhnim predpomnilnikom za strežbo zahtev, zadostuje 1 GB po virtualnem stroju. Pri večjih virtualnih strojih, ki poganjajo Java strežnike, je ponavadi potrebno 2 GB po virtualnem stroju.

Za izdelavo enostavnega oblaka bo zadoščalo 4 GB pomnilnika, kasneje se ga lahko brez težav poveča, če želimo delati z več primerki virtualnih strojev. Z ukazom *free -m* se preveri trenutno razpoložljiv pomnilnik (parameter *-m* vrača vrednost v MB, lahko se uporabi tudi *-k* za KB ali -g za GB).

**Disk**

Pri diskih je v virtualizacijskem okolju največkrat ozko grlo pri diskovnem vhodu/izhodu. Ko imamo veliko število vzporedno delujočih virtualnih strojev, se zelo hitro doseže maksimalno število vhodno/izhodnih operacij na sekundo (IOPS – Input/Output Operations Per Second). Zaradi tega je priporočljivo za začetek imeti vsaj tip diska SATA-2 (SATA – Serial Advanced Technology Attachment), ki je dovolj hiter za aplikacije, ki jih poganjal bo glavni strežnik. Če se pri SATA-2 diskih doseže maksimum, je naslednji po zmogljivosti SAS (Serial Attached SCSI) ali podoben. Imajo slabše razmerje med ceno in velikostjo, vendar je zmogljivost dosti večja. Danes so na trgu dostopni tudi hibridni SATA-2 diski, kateri kombinirajo značilnosti HDD in SSD (Solid-state Drive) v eno enoto. V bistvu vsebujejo trdi disk z manjšim SSD predpomnilnikom za izboljšanje zmogljivosti in časa dostopa do pogosto uporabljenih datotek

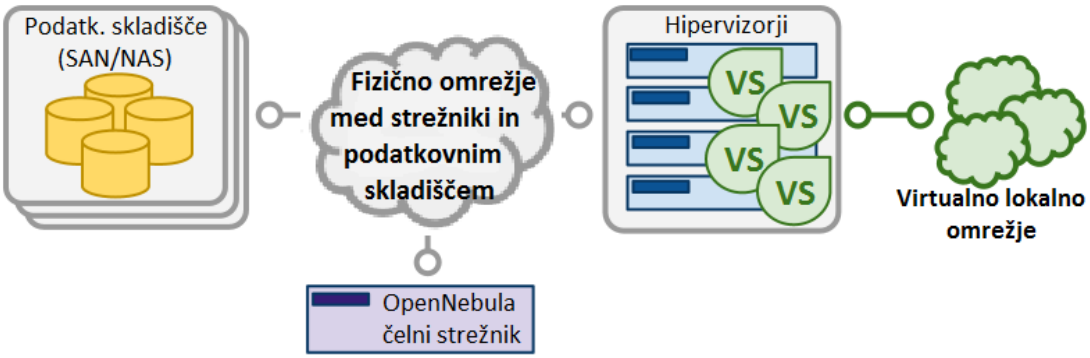
Poleg samega fizičnega diska se za večjo zmogljivost in zanesljivost pogosto uporablja diskovno polje RAID.

### Omrežna kartica

Izbira omrežne kartice je predvsem odvisna od planirane obremenitve. Če je planirana uporaba skupnega skladišča (NFS, iSCSI, AoE), je priporočljiva gigabitna mrežna kartica.

## Zahteve programske opreme

Za uspešno delovanje oblaka potrebuje OpenNebula en strežnik, ki je nastavljen kot čelno vozlišče (ang. *frontend node*). Ostali strežniki so vozlišča v gruči (ang. *clusted nodes*). Pri manjši infrastrukturi se lahko čelno vozliče uporabi tudi kot vozlišče v gruči. Pri večjih infrastrukturah lahko čelni strežnik zahteva veliko CPE moči, pomnilnika in diskovnega prostora in zaželeno je, da je postavljeno kot posebno vozlišče.




**Slika 3: Osnovne komponente sistema**

Osnovne komponente sistema so:

- Čelni strežnik: Vozlišče (strežnik), ki ga poganjajo storitve orodja OpenNebula
- Delovni strežniki: Vozlišča z vlogo hipervizorjev
- Podatkovno skladišče: Vsebuje glavne slike virtualnih strojev
- Fizično omrežje: Zagotavlja virtualizirano lokalno omrežje za virtualne stroje

### Zahteve za čelni strežnik

Strežnik, ki poganja storitve OpenNebule, se imenuje čelni strežnik (ang. *frontend server*). Imeti mora možnost komunikacije z vsemi ostalimi strežniki ter dostop do skupnih omrežnih skladišč.

Storitve OpenNebule vsebujejo proces za upravljanje (Management daemon (*oned*), časovni razporejevalnik virtualnih strojev (*mm_sched*), proces za nadzor, opazovanje in obračun (*onecctd*), strežnik za spletni vmesnik (*Sunstone*) in strežnik za API vmesnik od oblaka).

Komponente komunicirajo s pomočjo XML-RPC vmesnika, ki je lahko nameščen na strežnike. Čelni strežnik mora imeti nameščen Ruby 1.8.7.

**Zahteve za strežnike – hipervizorje**

Ostali strežniki so fizični stroji, ki bodo poganjali virtualne stroje. Z njimi upravlja OpenNebula s procesi, ki delujejo na čelnem strežniku z uporabo SSH protokola pri komunikaciji.

Strežnikom ni potrebno imeti nameščeno OpenNebulo. Programske zahteve za njih so:

- SSH strežnik z omogočenim overjanjem javnega ključa
- Nameščen hipervizor
- Programski jezik Ruby 1.8.7

**Podatkovno skladišče**

OpenNebula ima lastno shrambo slik, do katere dostopa preko čelnega strežnika z uporabo NAS (Network Attached Storage), SAN (Storage Area Network), direktne povezave ali drugega načina. Shranjevanje slik lahko deluje s skupnim skladiščem med strežniki. V tem primeru se lahko maksimalno izkoristi zmožnosti hipervizorjev, kot recimo Live Migration. Lahko deluje tudi brez skupnega skladišča med strežniki, kjer se slike premeščajo iz strežnika na strežnik pred zagonom ali ko so pavzirane (Cold Migration).

Shramba slika mora biti dovolj velika za shranjevanje vseh slik, ki bodo uporabljene. Virtualni stroji se lahko namestijo, da delajo s »kloni« glavne slike ali da direktno dostopajo do slike v shrambi.

Na primer, 64-jedrna gruča strežnikov ponavadi poganja okoli 80 virtualnih strojev, kjer vsaka potrebuje vsaj 10 GB diskovnega prostora. To pomeni, da potrebuje 800 GB v mapi /var/lib/one. V takem primeru je zaželeno imeti 10-15 glavnih slik, kar vzame dodatnih 200 GB prostora v /var/lib/images mapi. Skupaj je to 1 GB diskovnega prostora za nemoteno delovanje.

**Omrežje**

Pri hipervizorjih je potrebno ustrezno namestiti Ethernet most za vsako fizično omrežje, ki je z njimi povezano. Ponavadi so strežniki povezani z dvema fizičnima omrežjema - enim za javne IP naslove ter drugim za privatne LAN IP naslove.

### 3.1.2 Priprava sistema in namestitev orodja OpenNebula

Ko je enkrat ugotovljeno, da programska in strojna oprema ustrezata zahtevam, se lahko prične z namestitvijo OpenNebule. Obstajata dva načina namestitve. Prvi je s pomočjo prej zgrajene namestitvene datoteke, drugi pa namestitev iz izvirne kode.

Za začetnike se priporoča prvi način, ker je zelo poenostavljen in praviloma zadošča zahtevam uporabnika. Drugi primer se največ uporablja, če je potrebna kakšna sprememba pri načinu namestitve orodja, če se orodje ažurira ali briše. Za potrebe te vaje se priporoča prvi način. Tisti, ki imajo že nekaj izkušenj, lahko uporabijo drugi način.

**Namestitev s pomočjo prej zgrajene namestitvene datoteke**

Ta način namestitve je priporočen za začetnike in uporabnike kateri uporabljajo le eno različico OpenNebule. Na spletni strani OpenNebule [13, 22] se izbere različica namestitvene datoteke in se jo prenese na računalnik. Pri sistemih Ubuntu bo namestitvena datoteka imela ime oblike *Ubuntu-različica-opennebula-3.x.0-1_amd64.deb*. Namestitev se zažene z ukazom

*$ sudo dpkg –i ime_namestitvene_datoteke.deb*

V primeru, da sistemu manjkajo določeni paketi ali knjižnice, se jih lahko namesti z ukazom

*$ sudo apt-get install –f –y*

Na koncu je še potrebno namestiti Ruby knjižnice: *$ sudo ~/one/share/install_gems*

**Namestitev/prevajanje iz izvirne kode**

Pred samim prevajanjem iz izvirne kode je potrebno namestiti določene knjižnice. Kot je prej omenjeno, je v tej vaji opisan postopek namestitve v operacijskem sistemu Ubuntu. Informacije za ostale sisteme se lahko dobijo na uradni spletni strani OpenNebule [13, 22].

Za Ubuntu so potrebne knjižnice g++ (prevajalnik), libxmlrpc-c3-dev, scons, libsqlite3-dev, libmysqlclient-dev, libxml2-dev, libssl-dev ter ruby.

Namesti jih se z ukazom:

*$ sudo apt-get install g++ libxmlrpc-c3-dev scons libsqlite3-dev libmysqlclient-dev libxml2-dev libssl-dev ruby*

Pred namestitvijo je potrebno zgraditi paket z ukazom *$ scons [opcija=vrednost]*

Od opcij se lahko izbere:  sqlite_db (v vrednost se vpiše lokacija za namestitev), sqlite (vrednost »no« za izklop SQLite podpore), mysql (vrednost »yes« za MySQL podporo), xmlrpc (v vrednost se vpiše lokacija za namestitev) in parsers (vrednost »yes« za namestitev flex/bison datotek)

Glavni opciji, ki se uporabljajo v večini primerov sta MySQL ali SQLite podpora. Za SQLite podporo se uporabi ukaz *$ scons* , za MySQL podporo ukaz *$scons mysql=yes*.

Ko je enkrat vse zgrajeno, je potrebno zagnati namestitveno skripto z ukazom *$ ./install.sh [opcija] [vrednost]*.

Na izbiro so naslednje opcije:

- -u (uporabnik kateri bo zagnal OpenNebulo)
- -g (skupina uporabnika kateri bo zagnal OpenNebulo)
- -k (v primeru nadgradnje, ta opcija pusti obstoječo nastavitev)
- -d (lokacija za namestitev; če je postavljena pomeni zaprto namestitev, kjer se sistem uporablja še za druge stvari razen OpenNebule, ali več različic OpenNebule; če ni postavljena, bo namestitev v celotnem sistemu, primerna če se strežnik uporablja le za OpenNebulo)
- -c (le za namestitev orodja za odjemalca: OpenNebula CLI, OLLI in EC2)
- -r (za brisanje OpenNebule)
- -h (izpis pomoči za namestitev)

Najbolj pomembna opcija je -d. Poleg že omenjene razlike bo pri namestitvi v celotnem sistemu (angl. system-wide) uporabljena struktura map Unix sistema (/etc za nastavitev, /usr/bin za zagonske datoteke, /usr/lib za knjižnice). Pri taki namestitvi so potrebne root pravice. Zaprta namestitev (self-contained) kopira datoteke v mapo po izbiri, kjer root pravice niso potrebne, ter lahko obstaja več različic OpenNebule.

Za delo z OpenNebulo je potrebno definirati poseben uporabniški račun in njegovo delovno mapo. Ta račun bo uporabljen tudi za komunikacijo OpenNebule in hipervizorja. Ukaz za izdelovanje računa:

*$ sudo adduser –home /var/lib/one oneadmin*

Ko je račun narejen, za namestitev OpenNebule se uporabi ukaz *$ ./install.sh –d $HOME/one*.

Lahko se tudi določi posebna mapa ter se uporabniku da pravico nad njo:

*$ sudo mkdir /srv/cloud/one*
*$ sudo chown oneadmin:oneadmin /srv/cloud/one*
*$ ./install.sh –d /srv/cloud/one*

Potrebno je tudi namestiti določene Ruby knjižnice: Ruby 1.8.7, Sqlite3, MySQL, Curb, Nokogiri in Xmlparse. OpenNebula ponuja skripto za namestitev teh knjižnic. Pri namestitvi v celotnem sistemu se nahaja v mapi /usr/share/one/install_gems. Pri zaprti namestitvi je v mapi lokacija_opennebule/share/install_gems.

V Ubuntu sistemih, pred zagonom skripte je potrebno namestiti rubygems, ruby-dev in rake z ukazom *$ sudo apt-get install rubygems libopenssl-ruby ruby-dev rake libxslt1-dev*

Za tem je potrebno še zagnati skripto z ukazom *$ sudo ~/one/share/install_gems* pri zaprti namestitvi, ali *$ sudo /usr/share/one/install_gems* pri namestitvi v celotnem sistemu [14, 22].

### 3.1.3 Nastavitev orodja OpenNebula

Po namestitvi OpenNebule se lahko začne s nastavitvijo s pomočjo uporabnika, določenega za delo z OpenNebulo (oneadmin). Ukazi so privzeti iz dokumentacije orodja OpenNebula [14, 21].

Za lažje delo, je potrebno omogočiti uporabniku enostavno komunikacijo z vozlišči s pomočjo SSH protokola. Pri namestitvi iz izvirne kode je potrebno generirati RSA par ključev za uporabnika *oneadmin* z ukazom *$ ssh-keygen*.

Pred zagonom procesa je potrebno tudi določiti uporabnika *oneadmin* kot glavnega administratorja pri OpenNebuli. To se naredi z ukazi:

```
$ mkdir –p ~/.one
$ echo »oneadmin:oneadmin« > ~/.one/one_auth
$ chmod 600 ~/.one/one_auth
```

Uporabnik *oneadmin* je zdaj glavni OpenNebula uporabnik z vsemi pravicami potrebnimi za dodajanje novih uporabnikov, določanje njihovih pravic ter upravljanje s celo infrastrukturo.

Tisti, ki so se v prejšnjem delu vaje odločili za drugačen tip namestitve, morajo še definirati globalne spremenljivke. Ostali lahko preskočijo naslednji del ter nadaljujejo od »**Preverjanje delovanja**«.

Pri zaprti namestitvi, globalno spremenljivko ($ONE_LOCATION) se bo uporabilo kot lokacijo namestitve. Lahko se jo namesti kot pri namestitvi v celotnem sistemu z ukazom:

```
$ echo export ONE_LOCATION=$HOME/one | sudo tee –a /etc/profile.d/one
```

Če se uporablja samo uporabnik *oneadmin* je zadosti uporabiti ukaz:

```
$ echo export ONE_LOCATION=$HOME/one >> ~/.profile
```

Ukaz se lahko preveri, če se uporabnik odjavi, znova prijavi in poskusi ukaz *$ echo $ONE_LOCATION*. Izpis mora pokazati namestitveno lokacijo OpenNebule.

Za lažje delo se lahko v $PATH doda *bin* datoteko od OpenNebule (pri zaprti namestitvi) v izogib vpisovanju polne lokacije pri zagonskih datotekah OpenNebule:

*$ echo –e PATH=«\$ONE_LOCATION/bin:\$PATH» >> ~/.profile*

Ukaz lahko preverimo če se odjavimo ter znova prijavimo, in zaženemo ukaz *$ one –h*. Izpis bo pokazal pomoč za ukaz *one*.

**Preverjanje delovanja**

Na vrsti je preverjanje če je namestitev in osnovna nastavitev uspešna. Z uporabnikom oneadmin je potrebno vnesti ukaz *$ one start*. Po navadi se pojavi sporočilo da je proces že zagnan. Lahko ga se preveri z ukazom *$ ps ux | grep oned* .

V primeru težav, ali samo za informacijo se lahko preveri tudi log datoteko. Pri namestitvi v celotnem sistemu se uporabi ukaz *$ tail –f /var/log/one/oned.log*. Pri zaprti namestitvi se uporabi ukaz *$ tail –f $ONE_LOCATION/var/oned.log* .

Če vse deluje pravilno, bo ukaz *$ onevm list* izpisal prazno polje z glavo id, user, group, name, stat, cpu in mem.

Ko je proces uspešno zagnan ter deluje s privzetimi nastavitvami, se lahko nadaljuje z nastavitvijo OpenNebule. Ker se bodo v nadaljevanju spreminjale nastavitve, je potrebno vstaviti *oned* proces. To se naredi z ukazom *$ one stop*.

Naslednja naloga je nastavljanje vmware gonilnikov. To se naredi v nastavitveni datoteki, katera, razen gonilnikov, vsebuje tudi nastavitve za vse ostale komponente, ki jih uporablja OpenNebula. V nadaljevanju je opisana struktura te datoteke.

Nastavitvena datoteka (*oned.conf*) se nahaja v mapi */etc/one* pri nastavitvi v celotnem sistemu, ter *$ONE_LOCATION/etc* pri zaprti nastavitvi. V njej se nahajajo vse nastavitve, ki jih uporablja proces *oned* [14, 21].

Prvi del nastavitvene datoteke vsebuje naslednje opcije:

- MANAGER_TIMER: Interval bujenja za glavne funkcije *oned* procesa (v sekundah)
- HOST_MONITORING_INTERVAL: čas pri katerem se opazuje delovanje strežnika (v sekundah)
- HOST_PER_INTERVAL: število opazovanih strežnikov v intervalu
- VM_POLLING_INTERVAL: čas opazovanja virtualnih strojev

- VM_PER_INTERVAL: število opazovanih virtualnih strojev v intervalu
- VM_DIR: lokacija virtualnih strojev pri deljenem skladiščenju
- SCRIPTS_REMOTE_DIR: lokacija na strežnikih na katero čelni strežnik naloži skripte za opazovanje in nadzor virtualnih strojev
- PORT: port na katerem *oned* proces posluša XML_RPC klice
- DB: nastavitveni atributi za podatkovno bazo v ozadju
- VNC_BASE_PORT: VNC port za vsak virtualni stroj
- DEBUG_LEVEL: nivo obsega vpisovanja v log datoteko
- NETWORK_SIZE: privzeta velikost za virtualna omrežja (*netmask*)
- MAC_PREFIX: privzeti MAC prefiks za generiranje MAC naslovov
- DEFAULT_IMAGE_TYPE: privzeti tip slike; lahko je OS, CDROM ali DATABLOCK
- DEFAULT_DEVICE_PREFIX: hd – IDE vmesnik, sd – SCSI vmesnik, xvd – xen disk, vd – KVM disk

Če se uporabnik namesto SQLite odloči uporabljati MySQL, lahko uporabi vzorec v prilogi 1.

V tej vaji se bodo uporabile privzete nastavitve, tako da uporabniku ni potrebno spreminjati ta del nastavitvene datoteke. V nadaljevanju sledi nastavitev gonilnikov.

Nastavitvena datoteka vsebuje primere nastavitev za vse podprte hipervizorje. V privzetih nastavitvah so uporabljeni KVM gonilniki. Uporabnik jih mora zamenjati odvisno od hipervizorja, katerega bo uporabljal (vmware). To naredi na način, da jih vključi v komentar (da oznako # pred nastavitveno vrstico) ter isto oznako pobriše pred vmware gonilniki.

Spodaj so opisi opcij za posamezne gonilnike:

Pri namestitvi gonilnika za upravnika informacij (ang. *Information Manager driver*) je potrebno določiti:

- *name*: ime gonilnika
- *executable*:  pot do izvršljive datoteke/skripte gonilnika za upravljalnika informacij
- *argument*: argumenti za izvršljivo datoteko

Pri namestitvi gonilnika za upravnika virtualizacije (ang. *Virtualization Manager driver*) so potrebni:

- *name*: ime gonilnika
- *executable*: pot do izvršljive datoteke/skripte gonilnika za upravljalnika informacij
- *argument*: argumenti za izvršljivo datoteko
- *type*: tip gonilnika; Xen, KVM ali VMware
- *default*: pot do privzetih nastavitev parametrov za gonilnik

Pri namestitvi gonilnika za upravnika prenosa (ang. *Transfer Manager driver*) je potrebno določiti iste atribute kot pri gonilniku za upravljalnika informacij (*name*, *executable*, *argument*).

Pri namestitvi gonilnika za upravljanje s slikami (ang. *Image Manager driver*) je potrebno določiti:

- *executable*: pot do izvršljive datoteke/skripte gonilnika za upravljalnika informacij
- *argument*: argumenti za izvršljivo datoteko

V primeru da gonilniki niso pravilno nastavljeni, se bo pri zagonu *oned* procesa v ukazni vrstici izpisala napaka ter se bo v *log* datoteko vpisala vrsta napake.

Na koncu nastavitvene datoteke se nahaja še nastavitev »*hook*« sistema. Nastavitev »*hook*« sistema ni obvezna, vendar je priporočljiva. Kot je bilo že omenjeno, ima OpenNebula »*hook«* sistem, ki aktivira zagon določenih skript ob različnih dogodkih. Sam sistem je omogočen že pri privzeti nastavitvi, lahko pa se dodatno vklopijo posamezni deli/skripte.

Primer je HOST_HOOK, ki avtomatično znova zažene virtualne stroje, če se strežnik ne odziva, lahko jih tudi prenese na drugi strežnik. Drugi koristen primer je VM_HOOK, ki znova zažene virtualni stroj, če slučajno pride do sesutja.

Uporabnik lahko sam izbere, ali bo vklopil dodaten »hook«.

Za vse prej omejene nastavitve se priporoča nastavitev datoteke *oned.conf* ,opisana v prilogi 2.

**Nastavitev NFS strežnika**

Nastaviti je potrebno tudi NFS strežnik. Na njega se bo povezoval ESXi hipervizor in pobiral slike virtualnih strojev. Namesti ga se z ukazom *$ sudo apt-get install nfs-kernel-server*. Ko se namestitev konča, je potrebno v */etc/exports* dodati naslednje vrstice:

```
/var/lib/one/var/datastores/0
*(rw,sync,no_subtree_check,no_root_squash,anonuid=10000,anongid=10000)
/var/lib/one/var/datastores/100
*(rw,sync,no_subtree_check,no_root_squash,anonuid=10000,anongid=10000)
```

Z ukazom *$ id oneadmin* se preberejo *uid* in *gid* uporabnika *oneadmin* ter se te vrednosti vpišejo namesto 10000 pri *anonuid* in *anongid*. Datoteko se shrani te ponovno zažene nfs strežnik: *$ sudo /etc/init.d/nfs-kernel-server start* [14,15]

### 3.1.4 Upravljanje z uporabniki in skupinami ter kvote

Poleg glavnega uporabnika (*oneadmin*) v OpenNebuli se lahko določi tudi druge uporabnike in skupine. Ukaza za delo z uporabniki ali skupinami sta:

*$ oneuser <ukaz> [<argument>] [<opcija>]*
*$ onegroup <ukaz> [<argument>] [<opcija>]*

Za ogled trenutnih uporabnikov/skupin se lahko uporabi ukaz *$ oneuser/onegroup list*. Glavni privzeti skupini pri OpenNebuli sta *oneadmin* in *users*. Uporabniki *oneadmin* skupine imajo dostop do vseh pravic, uporabniki *users* skupine lahko dostopajo samo do skupnih ali lastnih objektov (virtualnih strojev, slik, omrežij). Od ostalih ukazov se lahko še omeni *create* za izdelavo novih uporabnikov oz. *show* za prikazovanje podrobnosti za uporabnika. Obstaja tudi možnost določanja lokalnih ali oddaljenih uporabnikov, odvisno od lastnih potreb.

V primeru ko več uporabnikov uporablja OpenNebulo, se jim lahko omeji dostop do virov. V privzetih nastavitvah ni določene kvote, lahko se pa določi na dva načina:

- privzete kvote: za vsakega uporabnika (:*defaults*:)
- kvote po uporabniku: za določene uporabnike

Obstajajo 4 različne kategorije za kvoto:

- *cpu* – enota je *float*; delež CPE dovoljen za VM (na primer 2.0 pomeni 2.0 GHz)
- *memory* – enota je MB; količina RAM-a (na primer 1024 pomeni 1 GB)
- *num_vms* – enota je *integer*; število zagnanih VM (na primer 4 pomeni 4 VM)
- *storage* – enota je MB; velikost prostora (na primer 20480 pomeni 20 GB)

Za oba načina se lahko nastavi poljubno število atributov, razlika je le v nastavitvi. Pri privzeti kvoti velikosti atributov se nastavijo v */etc/auth/quota.conf* datoteki. Pri kvoti po uporabniku se uporablja *onequota* pomožni program (primeri uporabe: *onequota set pips num_vms 8, onequaota list, onequota show pips –f).*

Kvoto se omogoči v *oned.conf* nastavitvah pri Auth Manager gonilniku.

V tej vaji ni potrebno uporabljati kvot, ker se ne bo ukvarjalo s samo uporabo oblaka.

### 3.1.5 Zaključek vaje

Kot je prej omenjeno je v prvem delu vaje bil cilj naučiti uporabnika, kako preveriti infrastrukturo. Zelo pomembno je, da uporabnik dobi občutek, kaj vse je potrebno za izdelovanje oblaka, kakšno infrastrukturo potrebuje in kako lahko preveri, kaj ima na voljo.

V nadaljevanju vaje je uporabnik lahko postopoma pripravil in nastavil sistem ter na koncu namestil OpenNebulo. Razen množice ukazov so bile opisane tudi druge možnosti nastavitve, saj uporabnik mora razumeti, kaj spreminja ter kako lahko to vpliva na delovanje sistema. To znanje bo v korist tudi pri odpravljanju morebitnih težav.

Na koncu sem kot končni »produkt« vaje dobil nameščen sistem, pripravljen za dodajanje delovnih strežnikov (hipervizorjev). Če ima uporabnik željo uporabiti drugačen operacijski sistem na čelnemu strežniku, je potrebno narediti določene spremembe tudi pri vajah (zaradi razlik v operacijskih sistemih in različicah OpenNebule).

## 3.2 Vaja 2 - Nastavitev in dodajanje hipervizorja

**Cilj vaje je nastaviti hipervizor strežnik in ga povezati s čelnim strežnikom.**

Po uspešni pripravi čelnega strežnika in namestitvi OpenNebule je na vrsti nastavitev delovnih strežnikov, ki bodo imeli vlogo hipervizorjev. Za potrebe vaje je uporabnikom na voljo en ESXi hipervizor. Podobno kot pri 1. vaji je sistem nameščen že vnaprej.

Za vajo 2 sta pripravljena dva posnetka sistema. Prvi je posnetek sistema čelnega strežnika, ki ima pravilno nameščeno in nastavljeno OpenNebulo. Drugi vsebuje »svež« ESXi hipervizor. Vaja je razdeljena na 2 dela. Prvi se nanaša na nastavitev hipervizorja, drugi na nastavitev čelnega strežnika in povezovanje s hipervizorjem.

### 3.2.1 Nastavitev VMware ESXi hipervizorja

Za upravljanje s ESXi hipervizorjem je potrebno prenesti VMware vSphere client aplikacijo katero se zažene z Windows sistema [16].


**Slika 4: VMware vSphere client aplikacija**

Naslednji korak po namestitvi je nastavitev pravic za *oneadmin* uporabnika. Ta korak je pomemben ker OpenNebula s pomočjo *oneadmin* uporabniškega računa upravlja s ESXi hipervizorjem.

Po prijavi na ESXi hipervizor (vpiše se naslov strežnika, uporabniško ime in geslo) je potrebno dodati oneadmin uporabniški račun ter mu dodeliti administratorske pravice. To se naredi z izvajanjem naslednjih korakov:

1. Klik na »*Local Users & groups*«.
2. Desni klik na seznam ter potem na »*Add*«.
3. V polja se vnesejo podatki »*User: oneadmin*«, »*UID: id* uporabnika *oneadmin* na čelnem strežniku« in »*Group membership: root*«.
4. Pri zavihku »*Permissions*«, desni klik na »*Add permission*«.
5. Za uporabnika *oneadmin* se nastavi »*Assigned Role*« na »*Administrator*« ter potrdi s klikom na »*OK*«.

VMware vSphere client aplikacijo se lahko uporablja tudi za nadzor delovanja hipervizorja in virtualnih strojev, ampak je omejena le na enega hipervizorja, ter ne podpira istočasen ogled ostalih.

V hipervizorju je tudi potrebno nastaviti povezavo do podatkovnega skladišča katero se nahaja na čelnem strežniku. Na ta način bo hipervizor imel dostop do slik virtualnih strojev. V vSphere clientu je potrebno kliknuti na »*Configuration*«, potem na »*Storage*« ter na »*Add datastore*«. Pri tipu skladišča se izbere NFS ter se vnesejo naslednji podatki:

*Server: IP naslov čelnega strežnika*
*Folder: /var/lib/one/datastores/100*
*Name: 100*

Za tem se postopek ponovi še enkrat ampak se uporabi ime mape in ime skladišča »0«.

OpenNebula prek čelnega strežnika komunicira s hipervizorji s pomočjo varnega ssh protokola. Za nemoteno komunikacijo je potrebno kopirati ssh ključ *oneadmin* uporabnika s čelnega strežnika na hipervizor [14, 20].

S pomočjo ssh se je potrebno prijaviti na ESXi hipervizor (*root* račun) ter izvesti naslednje ukaze:

*$ mkdir /etc/ssh/keys-oneadmin*
*$ chmod 755 /etc/ssh/keys-oneadmin*
*$ chown -R oneadmin /etc/ssh/keys-oneadmin*
*$ scp oneadmin@"IP naslov frontend strežnika":.ssh/id_rsa.pub /etc/ssh/keys-oneadmin/authorized_keys*

Delovanje se lahko preveri s poskusom povezovanja na hipervizor. Pri ukazu *ssh oneadmin@IP_hipervizorja* ne bo potrebno vpisovat gesla.

### 3.2.2 Nastavitev programske opreme na frontend strežniku

Za upravljanje z VMware hipervizorjem je na čelnem strežniku potrebno namestiti:

- libvirt, zgrajen z *–esx* zastavico katere uporabljajo OpenNebula gonilniki za virtualne stroje
- če ni lokalnega DNS strežnika je potrebno v */etc/hosts* datoteki dodati naslove strežnikov (na primer 192.168.66.99 esx01)
- v nastavitvah ssh je potrebno onemogočiti povpraševanje za dodajanjem strežnika v *known_hosts* datoteko: v *~/.ssh/config* se doda *Host * StrictHostKeyChecking no*

Za namestitev libvirt orodja je potrebno prenesti zadnji *libvirt-x.x.x.tar.gz* paket s uradne spletne strani. Najprej je potrebno namestiti pakete in knjižnici katere on zahteva. To se naredi z ukazom:

*$ sudo apt-get install build-essential gnutls-dev libdevmapper-dev python-dev libcurl4-gnutls-dev libnl-dev libxml2-dev*

Potem se še libvirt nastavi, zgradi in namesti z ukazi:

*$ ./configure –with-esx*
*$ make*
*$ sudo make install*
*$ sudo ldconfig*

V namestitveni datoteki libvirt orodja (*/etc/one/vmwarerc*) je potrebno dodati uporabniško poverilnico (*credential*):

*# Libvirt configuration*
*:libvirt_uri: »'esx://@HOST@/?no_verify=1&auto_answer=1'«*
*# Username and password of the VMware hipervizor*
*:username: »oneadmin«*
*:password: »opennebula«*

Za uporabo določenega strežnika s pomočjo OpenNebule je potrebno ta strežnik prej registrirati. To se naredi z ukazom *$ onehost create hostname –im im_mad –vmm vmm_mad -vn vnm_mad*. Prvi parameter, *hostname* se nanaša na ime strežnika, ki mora biti ustrezno nastavljeno v /etc/*hosts* datoteki ali DNS strežniku. Ostali parametri v tem ukazu določajo skripte, ki bodo uporabljene pri tem strežniku; *im_mad* za upravljalnika slik; *vmm_mad* za upravljalnika virtualnih strojev/virtualizacije in zadnji parameter, *vnm_mad* se nanaša na gonilnik za upravljanje z omrežjem (na primer *vmware*, *iptables*, *ebtables* in *vlan*).

V primeru, ko strežnik ali množica strežnikov ne bo več uporabljana, se jo lahko pobriše z ukazom *$ onehost delete range|hostid_list*. Opcijo *range* se definira z listo IDjev, kjer sta določena prvi in zadnji ID (na primer 1..8). Opcijo *hostid_list* se definira z listo IDjev, med sabo ločenih z vejico (na primer 1,2,5,8).

Obstajajo primeri, ko strežnika ni potrebno izbrisati, temveč samo onemogočiti (na primer zaradi nadgradnje, popravila). S tem se izogiba opazovanju in preprečuje zagon novih primerkov virtualnih strojev na strežniku. Za omogočanje/onemogočanje delovanja strežnika se uporablja ukaz *$ onehost enable/disable range|hostid_list*.

Poleg teh ukazov obstajajo še ukaz za zagon urejevalnika nastavitev (*$ onehost update hostid*), sinhronizacijo nadzornih skript (*$ onehost sync*), prikazovanje liste registriranih strežnikov (*$ onehost list*), prikazovanje detajlov za posameznega strežnika ($ *onehost show hostid*) in nadzor določenega števila strežnikov (primer za prikazovanje liste strežnikov, ki se osveži na vsake 3 sekunde: *$ onehost top –d 3*).

**Nastavitev omrežnega gonilnika**

Kot je bilo prej omenjeno je zadnji parameter pri registraciji novega strežnika omrežni gonilnik, ki se uporablja vsakič, ko se zažene novi virtualni stroj.

Na voljo so naslednji omrežni gonilniki:

- *dummy*: privzeti gonilnik kateri ne uveljavlja nobena posebna pravila. Vsi virtualni stroji povezani z istim fizičnim mostom lahko komunicirajo med sabo.

- *fw*: avtomatično naredi *iptables* pravila na strežniku ki poganja virtualne stroje. Ta gonilnik se uporablja za filtriranje različnih TCP/UDP vhodov in ICMP za vsak virtualni stroj
- *ebtables*: gonilnik ki avtomatično naredi *ebtables* pravila na strežniku za omogočanje omrežne izolacije med dvemi virtualnimi stroji zagnanimi na istem mostu, z različnimi /24 omrežjem.
- *802.1Q*: gonilnik uporabljen za omogočanje omrežne izolacije čez VLAN upravljan s strani strežnika s pomočjo *802.1Q* standarda
- *ovswitch*: popolno omrežno stikalo ki podpira VLAN, filter prometa, QoS in opazovanje s pomočjo standardnega vmesnika (na primer NetFlow, sFlow, SPAN in RSPAN)
- *vmware*: poseben VMware-ov gonilnik uporabljen za omrežno izolacijo izmed virtualnih strojev in *802.1Q* VLAN ko se uporablja ESXi strežnik. [14, 20]

Ko je vse nastavljeno se lahko registrira ESXi strežnik z ukazom:

*$ onehost create esx01 –im im_vmware –vmm vmm_vmware –vn vmware*

Če je vse poteklo brez težav, z ukazom *$ onehost list* se lahko preveri če je strežnik uspešno registriran.

Včasih se prvi prvem preverjanju ne pokažejo viri strežnika. V tem primeru se lahko poskusi povezovati na strežnik s čelnega strežnika s pomočjo ukaza:

*$ virsh –c esx://esx01/?no_verify=1*

Po vnosu uporabniškega imena in gesla, na zaslonu se bodo izpisali podatki o strežniku.

### 3.2.3 Zaključek vaje

V 2. vaji se je od uporabnika zahtevalo, da namesti hipervizorja ter ga poveže z OpenNebulo. Vaja je najprej zahtevala ustrezno namestitev hipervizorja s pomočjo posebne aplikacije. Potem je še bilo potrebno pripraviti čelni strežnik ter ga povezati s hipervizorjem.

V tej vaji se je uporabnik prvič srečal s komunikacijo med dvema strežnikoma, najprej v nastavitvah, kasneje tudi pri testih. Tukaj je potrebno biti posebej pozoren na spremembe v omrežju, saj je v primeru spremembe kakšnega od IP naslovov potrebno narediti ustrezne spremembe tudi na strežnikih. Ta detajl bo pomemben tudi v nadaljevanju, posebej, če se prekine vajo ali naredi večjo pavzo med izvajanjem naslednjih vaj.

Po uspešnem izvajanju vaje ima uporabnik dva med sabo povezana strežnika; čelni strežnik z nastavljeno OpenNebulo in ESXi hipervizor, pripravljen na delo z virtualnimi stroji.

# 4 Priprava vzorcev in osnove dela z OpenNebulo

Najbolj pomembno delo je opravljeno: čelni strežnik in hipervizor sta nastavljena in povezana. S tem je narejena osnova za delovanje oblaka. V naslednji vaji uporabnika še čaka nastavljanje ostalih komponent, kot so virtualno omrežje, skladišče slik in vzorci virtualnih strojev.

## 4.1 Vaja 3 - Upravljanje z virtualnim omrežjem in slikami

**Cilj vaje je nastaviti testno virtualno omrežje, pripraviti vzorce slik in virtualnih strojev ter jih dodati v OpenNebulo.**

Najprej je potrebno nastaviti virtualno omrežje, katere bodo virtualni stroji uporabljali. Potem je potrebno pripraviti slike in vzorce virtualnih strojev. Kot je na samem začetku teksta omenjeno, podpira OpenNebula število različnih načinov nastavljanja komponent. Vsako komponento nastavimo odvisno od zmogljivosti infrastrukture ali zahtev uporabnikov.

Skozi vajo se bo uporabnik naučil nastavljati omenjene komponente, pripraviti vzorce ter jih uporabiti v izdelavi oblaka. Pred začetkom vaje se pričakuje, da sta oba strežnika ustrezno nastavljena in povezana ter je za vsak strežnik pripravljen posnetek sistema.

Za lažje razumevanje so najprej opisani detajli za pripravo vzorcev za vsako komponento, potem pa ukazi za dodajanje teh vzorcev v OpenNebulo.  Primeri katere bo uporabnik lahko preizkusil v vaji, so v prilogah.

### 4.1.1 Upravljanje z virtualnimi omrežji

Vsak strežnik je lahko povezan na eno ali več fizičnih omrežij. Virtualni stroji lahko dostopajo do njih s pomočjo mostov. Možno jih je nastaviti tudi v primeru ko ni fizičnih naprav, kar pomeni da bodo virtualni stroji komunicirali samo med sabo. Virtualno omrežje je lahko povezano z enim ali več fizičnih omrežij, na primer LAN ali DMZ.

**Vzorec za obsežno omrežje (angl. ranged network)**

Pri obsežnem omrežju sta pomembna 2 parametra, *NETWORK_SIZE* in *NETWORK_ADDRESS*. Prvi se nanaša na velikost omrežja, drugi pa na začeten naslov.

V vzorcu morajo biti naslednji podatki:

| | |
|---|---|
| *NAME = »ran_lan«* | - ime omrežja |
| *TYPE = RANGED* | - tip omrežja, obsežno ali fiksno |
| *PUBLIC = YES* | - javno omrežje lahko uporabijo vsi uporabniki OpenNebule; privatno lahko samo tisti ki ga je naredil |
| *BRIDGE = VMNetwork* | - ime mosta kateri je narejen na fizičnem strežniku |
| *NETWORK_SIZE = 24* | - velikost omrežja |
| *NETWORK_ADDRESS = 192.168.66.0* | - začeten naslov |
| *# Custom Attributes* | |
| *GATEWAY = 192.168.66.1* | - naslov omrežnega prehoda |
| *DNS = 192.168.66.1* | - naslov DNS strežnika |

**Vzorec za fiksna omrežja**

Fiksna omrežja se uporabljajo ko se želi omejiti dostop za samo določene IP naslove. V tem primeru je potrebno dodati parameter LEASES ter obrisati parametra *NETWORK_SIZE* in *NETWORK_ADDRESS*. Pri parametru LEASES se lahko doda tudi MAC naslov pri določenem IP naslovu. Primer vzorca:

```
NAME = »fix_lan«
TYPE = FIXED
BRIDGE = fixed0
LEASES = [IP=130.10.0.1]
LEASES = [IP=130.10.0.2, MAC=50:20:20:20:20:21]
# Custom Attributes
GATEWAY = 130.10.0.1
DNS = 130.10.0.1
```

Ko je vzorec pripravljen se lahko na čelnem strežniku naredi virtualno omrežje z ukazom:

```
$ onevnet create <ime_vzorca>
```

Razen *create*, obstajata še dva ukaza. Ukaz *list* za izpis vse virtualnih omrežij ter ukaz *show* za izpis detajlov za določeno virtualno omrežje.

V prilogi 3 se nahaja primer vzorca kateri se lahko uporabi v vaji.

### 4.1.2 Upravljanje s diskovnimi slikami

Kot je prej omenjeno, OpenNebula uporablja shrambo slik za lažje delo z različnimi diskovnimi slikami. Vsaka slika lahko vsebuje operacijski sistem, podatke za aplikacije ali kateri namestitveni CD (na primer Matlab). Za slike predhodno rabi pripraviti določene vzorce katere se kasneje uporabi v OpenNebuli.

**Vzorec za slike s operacijskim sistemom**

Pri delu s slikami s operacijskim sistemom je pomembno da je sam operacijski sistem že v naprej nameščen. To se lahko naredi s različnimi orodji kot so virt-manager, VirtualBox ali VMware vSphere. Primer vzorca bi bil:

*NAME = »Ubuntu server«*
*PATH = /tmp/ubuntu.img*
*TYPE = **OS***
*PUBLIC = YES*
*DESCRIPTION = »Ubuntu server basic installation«*

Pri slikah je pomembno  da so v pravilnem formatu da jih hipervizor lahko zazna. KVM hipervizor uporablja formate *.raw*, *.qcow2*, *.vmdk*. Xen uporablja *.raw* in *.vmdk*. VMware uporablja *.vmdk*. Obstajajo tudi orodja katera omogočajo spremembo formata kot recimo qemu-img ali VboxManage converthd.

**Vzorec za slike s podatkih za aplikacije**

Pri podatkih za aplikacije se lahko definira prazen disk kateri se avtomatično formatira ob zagonu sistema. Primer vzorca:

*NAME = »Podatki za aplikacijo X«*
*TYPE = **DATABLOCK***
*# PATH ni nastavljen kar pomeni da bo slika zagnana kot novi prazni disk*
*SIZE = 20480*
*FSTYPE = ext3 #Tip datotečnega sistema (angl. File system type)*
*PUBLIC = NO*
*PERSISTANT = yes*
*DESCRIPTION = »Shramba za aplikacijo X«*

*»PERSISTANT«* nastavljen na *yes* pomeni da se bodo podatki vzdrževali med izdelavo in brisanje primerka. *»PUBLIC«* nastavljen na *no* pomeni da slika ne bo vidna drugim uporabnikom.

**Vzorec za sliko CDROM naprave**

Možno je tudi narediti vzorec slike katera bo namenjena samo branju. Primer bi bil namestitveni CD programa Matlab:

> *NAME = »MATLAB install CD«*
> *TYPE = **CDROM***
> *PATH = /tmp/matlab.iso*
> *DESCRIPTION = »Contains the MATLAB installation files. Mount it to install MATLAB on new OS images.«*

Pri uporabi VMware hipervizorja je zelo pomembno da je parameter *PATH* definiran kot:

> *PATH=vmware:///neposredno/ime/poti/do/mape*

V tej mapi se mora nahajati *disk.vmdk* datoteka.

Ko se pripravi vzorec, lahko ga se uporabi pri OpenNebuli s pomočjo *oneimage* orodja. Dodamo ga z ukazom:

> *$ oneimage create <vzorecslike.one>*

Obstajata še ukaza *list* za izpis vseh slik ter *show* za izpis detajlov za določeno sliko.

Razen ta tri osnovna ukaza obstajajo še ukazi za urejanje posameznih atributov pri že oddanimi vzorci. Ti atributi so:

- *persistant <id>*: naredi sliko trajnom ter shrani spremembe na sliki
- *non-persistant <id>*: v tem primeru spremembe ne bodo shranjene
- *disable <id>*: onemogoči določeno sliko da jo nobeden ne more uporabljati
- *enable <id>*: omogoči onemogočeno sliko
- *chtype <id> <type>*: spremeni tip slike (*OS*, *DATABLOCK* ali *CDROM*)
- *unpublish <id>*: razveljavi objavo; privatne slike lahko uporabljajo samo tisti kateri so jih naredili
- *publish <id>*: objavi sliko; javne slike lahko uporabljajo vsi uporabniki
- *chown <id> <userid> [<groupid>]*: spremeni lastnika in grupo slike
- *chgrp <id> <groupid>*: spremeni grupo za določeno sliko

V prilogi 4 se nahaja primer vzorca kateri se lahko uporabi v vaji.

## 4.2 Upravljanje z virtualnimi stroji

Za uspešen zagon virtualnega stroja je potrebno v naprej pripraviti vzorce kateri bodo vsebovali nastavitve na njegovo delovanje. Pri izdelavi vzorca virtualnega stroja je na izbiro veliko atributov. Večina primerov bo delovala tudi pri uporabi privzetih nastavitev. Potrebno je biti pozoren na omejitve infrastrukture, ki je na voljo ter zahteve virtualnih strojev, ki bodo uporabljani.

V nadaljevanju bodo opisani posamezni atributi ter možnosti nastavitev vsakega od njih. Zaradi večje preglednosti atributi v vzorcu so razdeljeni na več delov: zmogljivost, operacijski sistem in zagon, diskovna slika, omrežje, vhodno/izhodne naprave, zahteve in kontekst. Po opisu atributov sledi celoten primer nastavitve atributov za VMware, opis delovanja orodja *onetemplate* ter opis stanj katerih se virtualni stroj lahko nahaja. Na koncu je še opis delovanja orodja *onevm*, katero se uporablja za delo z virtualnimi stroji. Podatki v tabelah so prevzeti iz dokumentacije za OpenNebulo [14, 20].

### Zmogljivost

Pri atributih za zmogljivost se določajo viri kateri bodo na voljo posameznem virtualnem stroju:

| **Atribut** | **Opis** | **Privzeta vrednost** |
|---|---|---|
| NAME | Ime virtualnega stroja | *one-id* |
| MEMORY | Količina RAM pomnilnika za virtualni stroj, v MB | Obvezna |
| CPU | Odstotek CPE dodeljen virtualnem stroju | 1.0 |
| VCPU | Število virtualnih CPE dodeljenih virtualnem stroju | 1 |

### Operacijski sistem in zagon

S temi atributi se določajo nastavitve zagona kot so *kernel*, *initrd* in *bootloader*.

| Atribut | Opis | XEN vrednost | KVM vrednost | VMware vrednost |
|---|---|---|---|---|
| ARCH | CPE arhitektura | | I686/x86_64 | I686/x86_64 |
| KERNEL | Pot do kernel-a | Obvezna | Neobvezna | |
| INITRD | Pot do *initrd* slike | Neobvezna | Neobvezna | |
| ROOT | *Root* naprava | Neobvezna | Neobvezna | |
| KERNEL_CMD | Argumenti za *kernel* | Neobvezna | Neobvezna | |
| BOOTLOADER | Pot do bootloader-ja | Obvezna | Neobvezna | |
| BOOT | Način zagona; *hd*, *cdrom*, omrežje, *fd* | | *hd* | *hd* |

Primer za VMware: *OS = [ boot = hd, arch = x86_64]*

### Diskovna slika

Diskovna slika se lahko definira na več načinov. Atributi za uporabo slike iz shrambe slik:

| Atribut | Opis | Privzeta vrednost |
|---|---|---|
| IMAGE_ID | ID slike | Obvezna |
| BUS | Tip diska (*IDE*, *SCSI*, *Virtio*) | IDE |
| TARGET | Za spremembo vrstnega reda uporabljenega diska | Avtomatično določena |
| DRIVER | Določen tip diskovne slike | Odvisno od hipervizorja |

Lahko se uporabi tudi slika ki ni prej registrirana v shrambi slik. Na voljo so atributi:

| Atribut | Opis | Privzeta vrednost |
|---|---|---|
| TYPE | Tip diska: *floppy*, *disk*, *cdrom*, *swap*, *fs* in *block* | Odvisno od hipervizorja |
| SOURCE | Pot ali http naslov do datoteke | |
| SIZE | Velikost pri tipih *swap*, *fs* in *block*, v MB | |
| FORMAT | Format pri *fs* tipu (*ext2*, *ext3*, *ext4*, *vfat*, *ntfs*) | |
| TARGET | Map disk (*sda*, *sdb*, …) | Obvezna |
| CLONE | *yes* bo kopirala izvir, *no* bo uporabila izvir | *yes* |
| SAVE | Shrani sliko ali jo obriši po vstavljanju virtualnega stroja | *no* |
| READONLY | Koristno za *.iso* slike | *no* |
| BUS | Tip diska za emulacijo (*IDE*, *SCSI*, *Virtio*) | *IDE* |
| DRIVER | Določen tip diskovne slike | Odvisno od hipervizorja |

Primer iz shrambe slik: *DISK = [ IMAGE_ID = 2 ]*

Primer za swap napravo: *DISK = [ TYPE = swap, SIZE = 1024 ]*

### Omrežje

Omrežna naprava je določena s NIC zaporedjem (angl. Network Interface Card array). Obstajajo naslednji atributi:

| Atribut | Opis | Privzeta vrednost |
|---|---|---|
| NETWORK_ID | ID virtualnega omrežja | Obvezna če atribut *BRIDGE* ni določen |
| BRIDGE | Ime fizičnega mosta | Obvezna če atribut *NETWORK_ID* ni določen |
| IP | IP naslov dodeljen virtualnem stroju | Naključna |
| MAC | MAC naslov dodeljen virtualnem stroju | |
| TARGET | Ime naprave za uglasitev (angl. *tuning device*) | Avtomatično dodeljena |
| SCRIPT | Pot do skripte za zagon po izdelavi naprave za uglasitev | |
| MODEL | Strojna oprema katera bo emulirana | Odvisna od hipervizorja |
| WHITE_PORTS_TCP | Lista portov preko katerih je dovoljen dostop do virtualnega stroja za TCP protokol | Vsi dovoljeni |
| BLACK_PORTS_TCP | Lista portov preko katerih je prepovedan dostop do virtualnega stroja za TCP protokol | Nobeden prepovedan |

| | | |
|---|---|---|
| WHITE_PORTS_UDP | Lista portov preko katerih je dovoljen dostop do virtualnega stroja za UDP protokol | Vsi dovoljeni |
| BLACK_PORTS_UDP | Lista portov preko katerih je dovoljen dostop do virtualnega stroja za UDP protokol | Nobeden prepovedan |
| ICMP | Blokada ali sprejem ICMP povezav | Sprejem |

Primer NIC nastavitve za most: *NIC = [ BRIDGE = lan0]*

Primer NIC nastavitve za določeno omrežje s posebnim IP naslovom:

*NIC = [ NETWORK_ID = 1, IP = 102.168.66.66 ]*

### Vhodno/izhodne naprave

V določenih primerih je potrebno dodatno nastaviti vhodno/izhodne naprave kot recimo miško ali zaslon. Za to obstajajo atributi:

| Atribut | Opis | Privzeta vrednost |
|---|---|---|
| TYPE | Tip naprave, miška ali tablet | |
| BUS | Vhod prek katerega je naprava priključena | |

Lahko se dodajo tudi oddaljene naprave kot recimo VNC zaslon:

| Atribut | Opis | Privzeta vrednost |
|---|---|---|
| TYPE | Tip grafične naprave, *vnc* ali *sdl* (ne dela z VMware-om) | |
| LISTEN | Naslov za povezavo za VNC strežnik | 127.0.0.1 |
| PORT | Port za VNC strežnik | *KVM*: 5900 *Xen*: 0 |
| PASSWD | Geslo za VNC strežnik | |
| KEYMAP | Nastavitev tipkovnice | *Server locale* |

Primer: *GRAPHIC = [ TYPE = »vnc«, LISTEN = »0.0.0.0«, PASSWD = »mojegeslo«]*

### Zahteve

Obstajata dva atributa za spremembo delovanja razvrščevalnika ki določa na katerem strežniku bo deloval virtualni stroj:

| Atribut | Opis | Privzeta vrednost |
|---|---|---|
| REQUIREMENTS | *Boolean* vrednost za filtriranje strežnikov | Vsi so uporabljeni |
| RANK | Izraz uporabljen za sortiranje primernih strežnikov | Naključna izbira |

Primeri:

```
REQUIREMENTS = »NAME = \'thor*\'«     #strežniki katerih ime začne z »thor«
REQUIREMENTS = »HYPERVISOR = \'vmware\'«          #vmware hipervizorji
REQUIREMENTS = FREECPU > 1.9          #strežniki ki imajo več kot 2 proste CPE
RANK = FREEMEMORY – FREECPU*100          #sortiraj po strežnikih s več pomn.
                                         in manjšo CPE obremenitvijo
```

**Kontekst**

Kontekst predstavlja informacije prenesene virtualnem stroju prek ISO slike naložene kot particija.

Atributi katere vsebuje:

| Atribut | Opis | Privzeta vrednost |
|---|---|---|
| $VARIABLE | Poljubna spremenljivka | |
| FILES | Lista poti razdeljena s presledki | |
| TARGET | Naprava za priklop kontekstne ISO slike | sdb |

Primer:

```
CONTEXT = [
  hostname    = »$NAME«,
  ip_public   = »172.17.0.18«,
  files       = »init.sh /home/scorp/.ssh/id_rsa.pub«,
  target      = »hdc«,
  root_pubkey = »id_rsa.pub«,
  username    = »user«,
  user_pubkey = »id_rsa.pub«
]
```

Kot je prej omenjeno virtualni stroji bodo v večini primerov delovali z privzetimi nastavitvami (razen obveznih podatkov), lahko jih tudi nastavi po volji za bolj optimalno delo.

Primer celotnega vzorca za VMware hipervizor:

```
NAME = vmware-example
CPU = 0.1
MEMORY = 256
OS = [ BOOT = hd ]
DISK = [ IMAGE_ID = »3« ]
NIC = [ NETWORK_ID = »2« ]
```

Če bi se uporabil vzorec diskovne slike atributa *NAME* in *PATH* bi bila:

*NAME = MyVMwareDisk*
*PATH = vmware:///home/oneadmin/one/var/vmware-vm*

V mapi omenjeni z atributom *PATH* se mora nahajati *disk.vmdk* datoteka.

V prilogi 5 se nahaja primer vzorca kateri se lahko uporabi v vaji.

**Shramba vzorcev**

Za lažje delo z virtualnimi stroji, posebej če se planira posamezne vzorce razdeliti določenim grupam uporabnikov, se lahko uporabi orodje *onetemplate*.

Na voljo so naslednji ukazi:

*$ onetemplate create <datoteka>* # registrira novi vzorec
*$ onetemplate list* # izpis vseh vzorcev
*$ onetemplate update <id>* # odpre tekstualni urejevalnik za vzorec
*$ onetemplate instantiate <id> -m <število>* # zagon več istih primerkov
*$ onetemplate delete <id>* # obriše vzorec
*$ onetemplate show <id>* # izpiše detajle za vzorec
*$ onetemplate chgrp <id> <ime_grupe>* # daje pravice za uporabo in urejevanje -
*$ onetemplate chmod <id> <število>* # - grupi uporabnikov

**Opisi stanj za določen virtualni stroj**

Med zagonom in delovanjem, virtualni stroj spreminja svoje trenutno stanje. V nadaljevanju je opis vseh možnih stanj v katerih se virtualni stroj lahko nahaja:

- *Pending (pend)*: VS čaka na strežnik da ga zažene; stanje ne spremeni dokler ga uporabnik ali razvrščevalnik ne postavijo na strežnik
- *Hold (hold)*: Uporabnik lahko vstavi avtomatično postavljanje virtualnega stroja na strežnik
- *Prolog (prol)*: Upravljalnik prenosa prenosi virtualni stroj na strežnik
- *Running (runn)*: Virtualni stroj deluje brez napak
- *Migrate (migr)*: Virtualni stroj se seli z enega strežnika n drugi
- *Epilog (epil)*: Sistem čisti strežnika kateri je poganjal virtualni stroj; diskovne slike bodo shranjene nazaj na frontend
- *Stopped (stop)*: Virtualni stroj je vstavljen; njegovo stanje je shranjeno ter je skupaj s diskovnimi slikami prenesen nazaj na frontend
- *Suspended (susp)*: Virtualni stroj je pavziran, datoteke so če zmeraj na strežniku

- *Failed (fail)*: Zagon virtualnega stroja ni uspel
- *Unknown (unknown)*: Ni mogoče priti do informacij o virtualnem stroju; možno da se je sesul, ali se je njegov strežnik znova zagnal
- *Done (done)*: Virtualni stroj je končal z delovanjem; ne bo se več prikazoval na seznamu (*onevm list*)

**Delo z virtualnimi stroji**

Za delo z virtualnimi stroji se uporablja orodje *onevm*. Vsebuje veliko ukazov katere se uporablja odvisno od stanja v katerem se virtualni stroj nahaja. V primeru da uporabnik uporabi ukaz ki ni namenjen stanju v katerem se virtualni stroj nahaja, bo prejel opozorilo. Ukazi orodja *onevm* so:

- *onevm create <vm-id>*: naredi virtualni stroj kateri gre v stanje *Pending*
- *onevm list*: izpiše seznam vseh virtualnih strojev
- *onevm top*: izpisuje rezultate opazovanja vseh virtualnih strojev
- *onevm show <vm-id>*: izpiše detajle določenega virtualnega stroja
- *onevm deploy <vm-id> <host-id>*: Zažene virtualni stroj na določenem strežniku
- *onevm shutdown <vm-id>*: vstavi virtualni stroj
- *onevm livemigrate <vm-id> <host-id>*: slika se prenese s enega strežnika na drugi
- *onevm migrate <vm-id> <host-id>*: vstavi virtualni stroj in ga zažene na drugem strežniku
- *onevm hold <vm-id>*: nastavi virtualni stroj v *Hold* stanje
- *onevm release <vm-id>*: spremeni stanje virtualnega stroja iz *Hold* v *Pending*
- *onevm stop <vm-id>*: vstavi virtualni stroj in ga vrne nazaj na frontend
- *onevm cancel <vm-id>*: takoj uniči virtualni stroj (če *shutdown* ne deluje)
- *onevm suspend <vm-id>*: pavzira virtualni stroj
- *onevm resume <vm-id>*: virtulani stroj nadaljuje z delovanjem
- *onevm saveas <vm-id> <disk-id> <img-name>*: shrani določen disk virtualnega stroja kot novo sliko
- *onevm delete <vm-id>*: takoj uniči virtualni stroj; slika ki je mogla biti shranjena bo v *Error* stanju
- *onevm restart <vm-id>*: prisili hipervizorja da znova zažene virtualni stroj ki se vstavi v *Unknown* ali *Boot* stanju
- *onevm resubmit <vm-id>*: znova nastavi virtualni stroj v *Pending* stanje

Slika vseh možnih stanj ter ukazov kateri jih lahko spremenijo se nahaja v prilogi 6.

## 4.3 Zaključek vaje

Po uspešni izdelavi vzorcev virtualnega omrežja, slik in virtualnih strojev ter njihovem dodajanju v OpenNebulo je uporabnik uspešno naredil privatni oblak.

Skozi vajo je spoznal, kako spremeniti ali dodati posamezen detajl za določeni vzorec in kaj vse OpenNebula podpira. Pri tej vaji se same nastavitve OpenNebule ali ESXi strežnika ne spreminjajo, kar lahko da uporabniku več poguma za testiranje sistema pri različnih nastavitvah.

Če je uporabnik uspešno končal to vajo, je pridobil dovolj znanja za uspešno izdelavo oblaka. Trenutno lahko uporablja le ukazno vrstico, ampak vseeno lahko upravlja s celotnim sistemom. V nadaljevanju bo prikazano delovanje pripomočkov in vmesnikov, ki jih vsebuje OpenNebula za lažje upravljanje z oblakom.

# 5 Spletni vmesnik in upravljanje z oblakom

## 5.1 Vaja 4 - Spletni vmesnik Sunstone

**Cilj vaje je nastaviti spletni vmesnik Sunstone in ga uporabiti pri delu z oblakom.**

V prejšnjih poglavjih je bil opisan način uporabe orodja OpenNebula s pomočjo ukazne vrstice. Obstaja tudi drugi način uporabe, ki je večini uporabnikov bolj zanimiv. To je način uporabe, ki poteka prek grafičnega vmesnika Sunstone. Prek njega oskrbniki lahko na enostaven način upravljajo z viri, opravljajo operacije nad njimi ali opazujejo stanje celotne infrastrukture.

Prvi prvem delu vaje bo opisan način, kako nastaviti spletni vmesnik Sunstone. Zatem sledi par primerov uporabe.

### 5.1.1 Nastavitev spletnega vmesnika Sunstone

Spletni vmesnik Sunstone se avtomatično namesti z OpenNebulo. Za svoje delovanje potrebuje Ruby gems-e (*json, rack, sinatra* in *thin*). V prejšnjih vajah je že opisan način, kako jih namestiti zato ni potrebe po ponovnem opisovanju.

Kot je bilo prej omenjeno podpira OpenNebula tudi spletni grafični pregledovalnik operacijskih sistemov (VNC – angl. Virtual Network Computing). Z njim se lahko direktno poveže na virtualni stroj v oblaku. V določenih primerih se oskrbniki želijo izogniti namestitvi VNC odjemalca ter uporabiti spletni odjemalec. Za to je potrebno namestiti še dodatno knjižnico, *noVNC* (angl. VNC HTML5 web-socket client) na odjemalcu in VNC namestnik na strežniku.

Pri zaprti namestitvi skripto za namestitev lahko najdemo v mapi *$ONE_LOCATION/share* ter jo zaženemo z ukazom *$ ./install_novnc.*sh. Pri namestitvi v celotnem sistemu se skripta nahaja v */usr/share/one* mapi.

Na koncu je še potrebno preveriti nastavitev v *sunstone-server.conf* namestitveni datoteki [19].

```
# OpenNebula server contact information
:one_xmlrpc: http://localhost:2633/RPC2
# Server Configuration
:host: 0.0.0.0
:port: 9869
:auth: basic
# VNC Configuration
:vnc_proxy_base_port: 29876
:novnc_path: /var/cloud/one/share/noVNC
```

Ostal je še zadnji korak in to je zagon Sunstone strežnika z ukazom *$ sunstone-server start*.

### 5.1.2 Uporaba spletnega vmesnika Sunstone

Uporaba Sunstone vmesnika je zelo preprosta. Najprej je potrebno odpreti spletno stran vmesnika http://localhost:9869 ter se prijaviti z *oneadmin* uporabniškim imenom. V primeru težav s prijavo se lahko v nastavitveni datoteki preveri geslo ali port za stran (9869 je privzeti port).

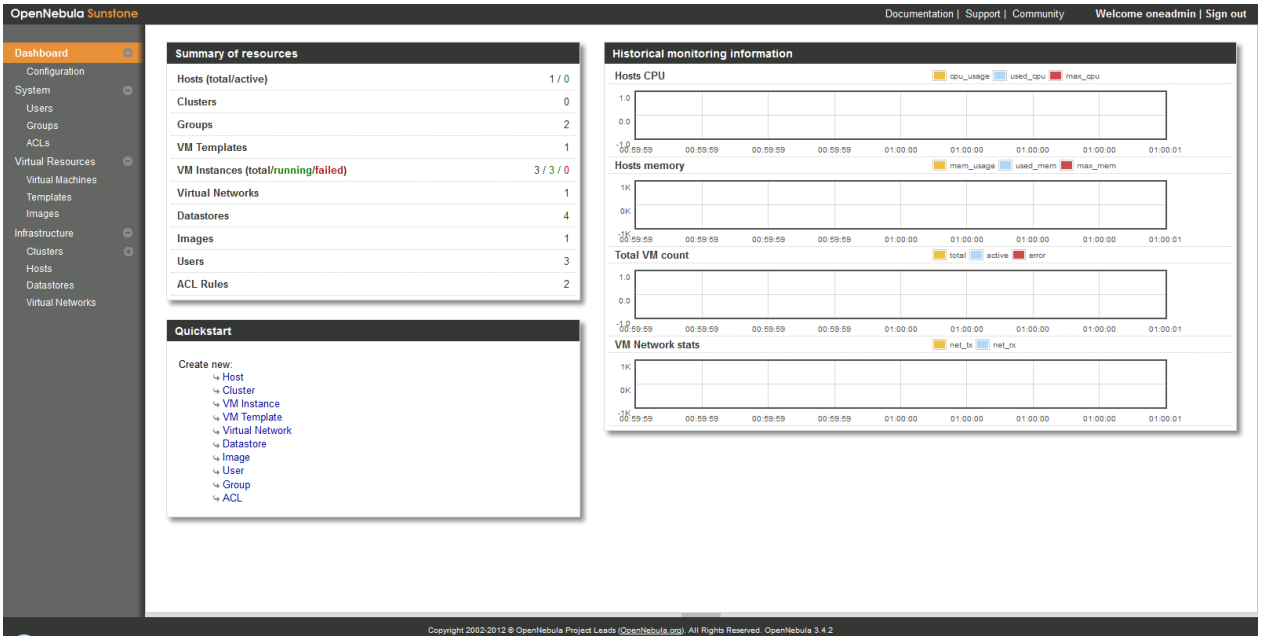

**Slika 5: Sunstone vmesnik - stran za prijavo**

Po uspešni prijavi se odpre pregledna plošča, ki ponuja veliko informacij. Tukaj so povezave na spletno stran OpenNebule, kjer se lahko dobi dodatna dokumentacija ali pomoč. Nižje na strani je okno s pregledom vseh virov, ki so na voljo. Poleg tega obstaja del za hitro dodajanje novih

virov. Na desni strani je okno z grafi, ki prikazujejo stanje CPE, pomnilnika, število virtualnih strojev in omrežno statistiko.

Grafi se generirajo s pomočjo podatkov, ki jih zbira OpenNebula s pomočjo procesov za obračun in statistiko. Zažene se ga lahko na frontendu s pomočjo ukaza *$ oneacct start*.

Na levi strani je meni, ki je razdeljen v kategorije, kot so strežniki (angl. hosts), virtualni stroji (angl. virtual machines), vzorci virtualnih strojev (angl. VM templates), virtualna omrežja (angl. virtual networks), diskovne slike (angl. disk images) ter uporabniki in grupe. Z izbiro posamezne kategorije se odprejo podstrani, ki omogočajo delo z določeno komponento.




**Slika 6: Sunstone vmesnik - Glavna stran**

Način dela je zelo podoben delu z ukazno vrstico. Namesto ukazov in urejevanja vzorcev s pomočjo tekstualnega urejevalnika se uporablja spletni vmesnik ter podatke vnaša v za to določena polja. Vmesnik vsebuje že vnaprej definirane namestitvene vzorce, ampak uporabniku omogoča tudi urejevanje ali dodajanje lastnih.

### 5.1.3 Zaključek vaje

V 4. vaji je uporabnik uspešno nastavil vmesnik Sunstone. Kot je vidno iz primera uporabe, Sunstone omogoča bolj elegantno delo z oblakom. Grafični prikaz virov omogoča lažji nadzor celotnega sistema brez uporabe dodatnih ukazov.

Tudi delo z oblakom je bolj enostavno. Vmesnik je zelo enostaven za uporabo ter omogoča dodajanje, spreminjanje ali brisanje komponent na enak način kot pri uporabi ukazne vrstice. Podobno kot pri 3. vaji lahko uporabnik sam preizkusi delovanje sistema, brez da bi ga pri tem skrbelo, ali se bo sistem sesul.

## 5.2 Vaja 5 – Opazovanje in testiranje oblaka, odpravljanje napak

**Cilj vaje je opazovati stanje oblaka in ustrezno ukrepati pri določenih dogodkih**.

V prejšnjih vajah se je uporabnik naučil, kako zgraditi oblak na infrastrukturi, kaj vse je potrebno narediti na strežnikih ter kako ustrezno nastaviti komponente, da delujejo v oblaku.

Za zadnjo vajo se lahko določi več različnih pod-nalog. Ko je uporabnik naredil oblak, se lahko simulirajo različni dogodki, ki zahtevajo njegovo ukrepanje. Ta vaja bi v bistvu predstavljala preizkus znanja, ki ga je uporabnik osvojil pri učenju.

### 5.2.1 Opazovanje in testiranje delovanja oblaka

V prvem delu vaje uporabnik opazuje oblak ter na njega poskuša dodati določeno število virtualnih strojev. Vzorce se lahko spremeni, se jim dodeli več virov in doda v OpenNebulo. Med izvajanjem celotne vaje mora uporabnik spremljati obnašanje sistema.

Druga naloga pri tej vaji je določiti število dodatnih uporabnikov, ki bodo imeli kvote na svojih uporabniških računih ter jih dodati v sistem skupaj z določenimi omejitvami. Potem je potrebno še narediti nove virtualne stroje in jih dodeliti tem uporabnikom.

Odvisno od števila uporabnikov in obremenitve VCLja se lahko uporabnikom omogoči še kakšen dodaten ESXi strežnik, katerega morajo nastaviti ter dodati v oblak. V nadaljevanju vaje se lahko doda še virtualne stroje, kateri bodo delovali na več hipervizorjih.

Za konec vaje se lahko povzroči sesutje enega strežnika in spremlja ukrepanje uporabnika. Najprej je potrebno narediti vse potrebne korake za reševanje virtualnih strojev, šele potem

ugotoviti, kaj je s strežnikom narobe. Zaželena je tudi uporaba »hook« sistema z vklopljenimi možnostmi za migracijo virtualnih strojev, ponovni zagon strežnika in obveščanje o napakah preko elektronske pošte. Pri odpravljanju napak se uporabnik lahko vrne na določeno vajo ter preveri, katere nastavitve je potrebno spremeniti, da bo sistem deloval.

### 5.2.2 Zaključek vaje

Odvisno od znanja, ki so ga uporabniki osvojili med izvajanjem prejšnjih vaj, se lahko hitro vidi, kako bodo znali upravljati z lastnimi oblaki. Izvajanje zadnje vaje je predvsem odvisno od izvajalca, ki vaje organizira. On lahko določi dodatne naloge za preverjanje znanja in ukrepanje uporabnikov pri naključnih dogodkih. Poleg tega je potrebno še upoštevati število uporabnikov, ki se učijo ter zmogljivost infrastrukture, ki je na voljo.

# 6 Zaključek

Eden izmed prvih ciljev diplomske naloge je bil ugotoviti, kako bi se lahko virtualni laboratorij uporabil za učenje o postavljanju oblaka. Na začetku se je bilo potrebno naučiti, kako sploh oblak deluje, kaj vsebuje in potrebuje za delo. Po izbiri hipervizorjev se je bilo potrebno odločiti, katero orodje bo upravljalo z oblakom ter preveriti, če bo dovolj močno za opravljanje dane naloge. V množici orodij, ki so bili na voljo, smo v ožji izbor dali OpenNebulo, OpenStack in Eucalyptus. V tem času je OpenNebula po razvoju bila pred ostalimi, zaradi česar smo jo izbrali kot orodje za učenje o postavljanju oblaka. Danes se OpenStack čedalje bolj uveljavlja in zna se zgoditi, da bi se ga lahko uporabilo namesto OpenNebule.

Po izbiri orodja je sledilo učenje, kako postaviti zasebni oblak in z njim upravljati. V tem času se je OpenNebula razvijala naprej, zaradi česar je prišlo do težav z novimi različicami. Velikokrat se je zgodilo, da se niso popolnoma ujemale s knjižnicami in drugimi orodji, katera smo uporabljali že prej. S časom so bile odpravljene tudi te težave, tako da smo se počasi bližali prvem cilju. Preveriti smo morali še, če obstajajo težave z gnezdeno virtualizacijo, saj je bil naš načrt postaviti zasebni oblak znotraj oblaka virtualnega laboratorija. Virtualni laboratorij v LRK za virtualizacijsko okolje uporablja VMware ESXi hipervizor, preko katerega poganja virtualne stroje. Kot hipervizor za izdelavo oblaka z OpenNebulo je bil tudi izbran ESXi in potrebno je bilo preveriti, če ESXi strežnik deluje kot virtualni stroj na ESXi hipervizoriju. Prišli smo do ugotovitve, da je to možno z dodatnim nastavljanjem ugnezdenega ESXi strežnika in da bo gnezdena virtualizacija delovala.

Kot naslednji cilj naloge je bilo potrebno pripraviti vaje, ki bodo pomagale študentom ali bodočim administratorjem pri učenju postavljanja oblaka. Pri uporabi dokumentacije so se pojavljale težave, ker so opisani postopki zahtevali veliko predznanja, zaradi česar so posamezni koraki bili prezahtevni za začetnike. Drugi problem je bil v količini dela, saj je bilo potrebno nastaviti vsaj 2 strežnika, OpenNebulo in vse komponente, ki jih potrebuje. Ker se gre za neizkušene uporabnike, težave so bile tudi pri vnosu ukazov. Na primer, pri napačnem vrstnem redu so se pogosto pojavljale napake, ki bi negativno vplivale na celoten sistem in celo onemogočale nadaljnje delo. Na podlagi lastnih izkušenj pri učenju smo ugotovili, da bi bilo najbolje, da bi vaje naredili sistematično, dodali teorijo ter delujoče primere. Med delom je potrebno sistem testirati ter narediti posnetke. S pomočjo teoretične osnove bi se uporabniki

naučili, zakaj sploh uporabljajo določene ukaze in kaj jim je potrebno spremeniti, če bi imeli drugačno infrastrukturo. Razen porazdeljene vsebine so med vsako vajo narejeni posnetki sistema. To uporabnikom omogoča, da imajo na začetku vaje delujoč sistem ter jim ni potrebno skrbeti, če so preskočili kakšen korak. Daje jim tudi dodaten pogum, ker jim v primeru večjih težav ni potrebno začeti od znova, temveč lahko nadaljujejo od zadnje vaje. Lahko tudi testirajo več različnih nastavitev ter spremljajo odziv sistema. S tem so odpravljene vse težave, s katerimi se lahko srečajo, zaradi česar so vaje primerne začetnikom.

V teoriji lahko vaje opravlja poljubno število uporabnikov/ bodočih administratorjev. Vsakemu se dodeli 2 strežnika in so pripravljeni za učenje. V praksi se pojavi težava zaradi zmogljivosti infrastrukture virtualnega laboratorija. ESXi strežniki zahtevajo vsaj 2 GB pomnilnika in če se doda še 1 GB za čelni strežnik, potem je to 3 GB pomnilnika na osebo. Kar se diskovnega prostora tiče je dovolj, če ima oseba na voljo 5 -10 GB, posebej če se med vajami naložijo vnaprej pripravljeni posnetki sistema. Virtualni laboratorij v LRK se med študijskim letom uporablja tudi za vaje pri predmetih ter bi sočasno izvajanje vaj za administratorje dodatno obremenilo celotni sistem. To pomeni, da se istočasno ne more učiti veliko uporabnikov, ampak se morajo prilagoditi odvisno od zasedenosti virov. Kratkoročna rešitev bi bila dobra časovna organizacija pri uporabi virov, za daljše obdobje pa ostaneta dve možnosti. Prva je nadgradnja infrastrukture, kar je priporočljivo samo v primeru, če se bo maksimalno izkoriščala med celotnim letom. Druga je uporaba hibridnega oblaka. V tem primeru bi se lahko samo v določenem času uporabili dodatni, zunanji viri (npr. v času izvajanja vaj).

Kar se vsebine vaj tiče, če se izkaže, da je uporabljeni pristop učinkovit, se odpira veliko novih možnosti za nadaljevanje dela. Lahko se pripravi nadaljevalni sklop vaj, kjer se lahko uporabnike uči o dodatnih nastavitvah, uporabi novih orodij, drugačnih strežnikov, itd. Z uporabo oblaka kot podlage za učenje lahko »preskočimo« omejitve fizične infrastrukture ter maksimalno izkoristimo možnosti, ki nam jih virtualizirano okolje ponuja.

# 7 Priloge

## 7.1 Priloga 1 – Postopek za nastavitev mysql baze

Pri uporabi mysql namesto sqlite je potrebno:

- namestiti MySQL strežnik: *$ sudo apt-get install mysql-server*
- nastaviti uporabniško ime in ustrezno bazo za uporabnika OpenNebulo:
  *$ mysql –u root –p*
  *mysql> create database one;*
  *mysql> CREATE USER 'one'@'localhost' IDENTIFIED BY 'onepassword';*
  *mysql> GRANT ALL ON one.* TO 'one'@'localhost';*
  *mysql> quit*
- v *oned.conf* datoteki pri DB je potrebno nastaviti:
  *DB = [ backend = »mysql«,*
  *server = »localhost«,*
  *port = 3306,*
  *user = »one«,*
  *passwd = »onepassword«,*
  *db_name = »one« ]*

## 7.2 Priloga 2 – Vzorec za nastavitev OpenNebule

Vrstice ki so v komentarjih so v tem vzorcu obrisane. Kot je že razloženo v vaji, nastavitve drugače ni treba brisati, lahko jih se samo da v komentarje.

```
HOST_MONITORING_INTERVAL = 600
VM_POLLING_INTERVAL      = 600
SCRIPTS_REMOTE_DIR=/var/tmp/one
PORT = 2633
```

```
DB = [ backend = "sqlite" ]
VNC_BASE_PORT = 5900
DEBUG_LEVEL = 3
NETWORK_SIZE = 254
MAC_PREFIX   = "02:00"
DATASTORE_LOCATION = /vmfs/volumes
DEFAULT_IMAGE_TYPE    = "OS"
DEFAULT_DEVICE_PREFIX = "hd"
IM_MAD = [
   name       = "im_vmware",
   executable = "one_im_sh",
   arguments  = "-t 15 -r 0 vmware" ]
VM_MAD = [
name       = "vmm_vmware",
   executable = "one_vmm_sh",
   arguments  = "-t 15 -r 0 vmware",
   default    = "vmm_exec/vmm_exec_vmware.conf",
   type       = "vmware" ]
TM_MAD = [
   executable = "one_tm",
   arguments  = "-t 15 -d dummy,shared,qcow2,ssh,vmware,iscsi" ]
DATASTORE_MAD = [
   executable = "one_datastore",
   arguments  = "-t 15 -d fs,vmware,iscsi"
]
HM_MAD = [
   executable = "one_hm" ]
AUTH_MAD = [
   executable = "one_auth_mad",
   arguments  = "--authn ssh,x509,ldap,server_cipher,server_x509"
]
SESSION_EXPIRATION_TIME = 900
VM_RESTRICTED_ATTR = "CONTEXT/FILES"
VM_RESTRICTED_ATTR = "DISK/SOURCE"
VM_RESTRICTED_ATTR = "NIC/MAC"
VM_RESTRICTED_ATTR = "NIC/VLAN_ID"
VM_RESTRICTED_ATTR = "RANK"
IMAGE_RESTRICTED_ATTR = "SOURCE"
```

## 7.3 Priloga 3 – Vzorec za nastavitev omrežja pri vaji

```
NAME    = "TestnoOmrezje"
TYPE    = "ranged"
BRIDGE  = "vSwitch0"
NETWORK_ADDRESS=172.16.33.0
NETWORK_SIZE=C
IP_START=172.16.33.2
IP_END=172.16.33.200
```

Ta vzorec se shrani v datoteko *test_omrezje.conf*. Potem se ga doda z ukazom *$ onevnet create test_omrezje.one*.

## 7.4 Priloga 4 – Vzorec za nastavitev slike pri vaji

```
NAME=TinyCore-TestnaSlika
DESCRIPTION="Tinycore linux za test pri izdelavi oblaka"
PATH=/home/oneadmin/images/TinyCore
```

Ta vzorec se shrani v datoteko *test_slika.one*.

Razen slike je potrebno narediti tudi vzorec za podatkovno skladišče.

V datoteko *datastore.one* se vpiše:

```
NAME = VMwareDS
DS_MAD = vmware
TM_MAD = shared
```

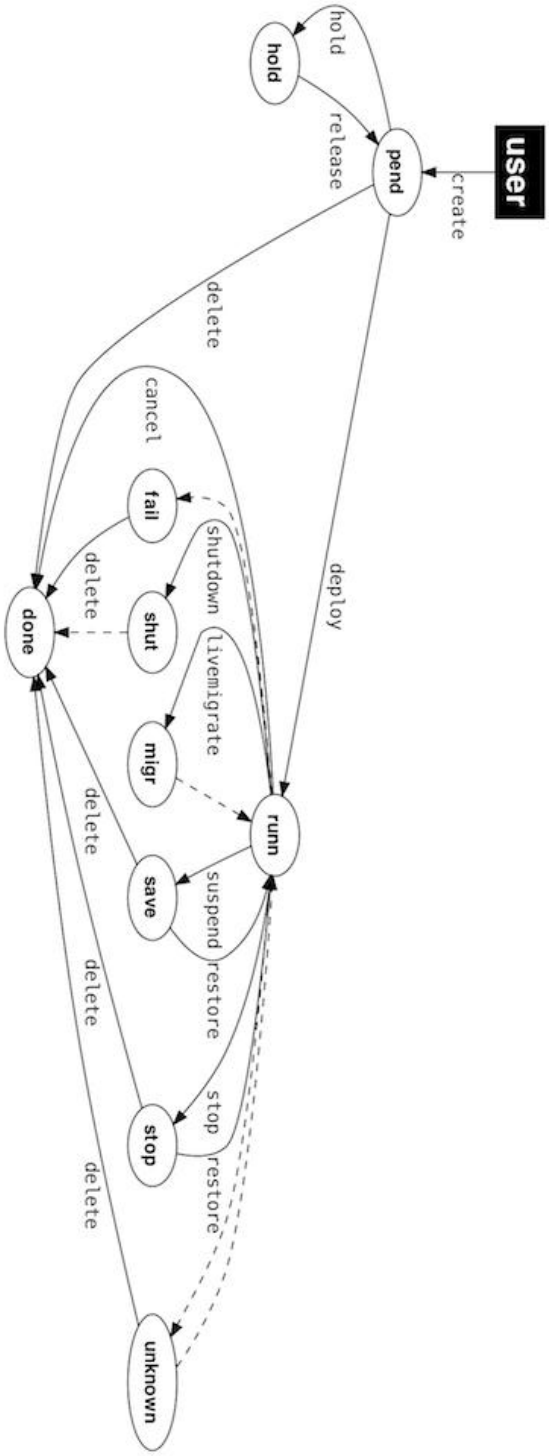
Potem se ga doda z ukazom *$ onedatastore create datastore.one*. Ko je skladišče dodano se bo izpisal njegov id. Zdaj se lahko registrira slika in poveže s tem skladiščem: *$ oneimage create test_slika.one –d ID_skladisca*

## 7.5 Priloga 5 – Vzorec za nastavitev virtualnega stroja pri vaji

```
CPU="1"
DISK=[
  BUS="scsi",
  IMAGE="TinyCore-TestaSlika",
  IMAGE_UNAME="oneadmin",
  TARGET="hda" ]
GRAPHICS=[
  LISTEN="0.0.0.0",
  TYPE="vnc" ]
MEMORY="1024"
NAME="VS-Vzorec-za-vajo"
NIC=[
  NETWORK=" TestnoOmrezje ",
  NETWORK_UNAME="oneadmin" ]
OS=[
  ARCH="i686" ]
RAW=[
  TYPE="vmware" ]
TEMPLATE_ID="0"
VCPU="1"
```

Ta vzorec se shrani v datoteko *testvm.one*. Uporabimo ga z ukazom *$ onevm create testvm.one*.

## 7.6 Priloga 6 – Ukazi in sprememba stanj pri virtualnih strojih



# 8 Literatura

[1] M. Ambrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, »A view of cloud computing«, *Communications of the ACM*, zv. 53, št. 4, str. 50-58, apr. 2010

[2] E. Bauer, A. Randee, »Reliability and availability of cloud computing«, John Wiley & Sons, Inc., New Jersey, 2012

[3] M. Ambrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, »Above the Clouds: A Berkely View of Cloud Computing«, UC Berkely Reliable Adaptive Distributed Systems Laboratory, ZDA, feb. 2009

[4] C. Yang, S. Wang, C. Liao, »On Construction of Cloud Virtualization with Integration of KVM and OpenNebula«, Springer-Verlag, Berlin, 2011

[5] Ebtables, http://ebtables.sourceforge.net

[6] T. Mather, S. Kumaraswamy, S. Latif, »Cloud Security and Privacy«, O'Reilly Media, Sebastopol, CA, 2009

[7] »An Overview of OpenNebula«, http://opennebula.org/documentation:archives:rel3.4:intro

[8] Xen Hypervisor Project, http://www.xen.org/products/xenhyp.html

[9] ESXi, http://www.vmware.com/files/pdf/ESXi_architecture.pdf

[10] G. Toraldo, »OpenNebula 3 Cloud Computing«, Packt publishing, maj 2012

[11] M. Grohar, »Integracija hibridnega oblaka z virtualnim laboratorijem«, FRI, diplomsko delo, str. 13-24, 2011

[12] M. Vouk, S. Averitt, M. Bugaev, A. Kurth, A. Peeler, H. Shaffer, E. Sills, S. Stein, J. Thomson, »Powered by VCL – Using Virtual Computing Laboratory (VCL), *Proc. 2nd International Conference on Virtual Computing (ICVCI)*, maj 2008

[13] OpenNebula, http://opennebula.org/

[14] OpenNebula – Dokumentacija, http://opennebula.org/documentation:rel3.6

[15] NFS protokol, http://rsync.tools.ietf.org/html/rfc5661

[16] vSphere, http://www.vmware.com/products/datacenter-virtualization/vsphere/index.html

[17] Ubuntu Server Guide, https://help.ubuntu.com/12.04/serverguide/index.html

[18] CentOS, http://www.centos.org/docs/5/

[19] C12G Labs S.L., »Advanced Setups for your Cloud Infrastructure, OpenNebula 3.6«, C12G Labs, 2012

[20] C12G Labs S.L., »Setting up and Managing your Virtual Resources, OpenNebula 3.6«, C12G Labs, 2012

[21] C12G Labs S.L., »Setting up and Managing your Cloud Infrastructure, OpenNebula 3.6«, C12G Labs, 2012

[22] C12G Labs S.L., »Designing and Installing your Cloud Infrastructure, OpenNebula 3.6«, C12G Labs, 2012