HPE HELION OPENSTACK
LAB GUIDE

ČÁST PRVNÍ- ZÁKLADY

HPE Helion OpenStack 2.0
Listopad 2015
Tomáš Kubica
Dokument verze 2.00
www.cloudsvet.cz

Obsah

1. Úvod do Helion OpenStack

1.1. HPE Helion OpenStack jako produkt

HP Helion OpenStack je produkt postavený na open source projektu OpenStack a je zaměřený na automatizaci IT infrastruktury, tedy Infrastructure as a Service (IaaS). Spojuje virtualizaci serverů, sítě i storage do jednoho orchestračního rámce a můžete tak velmi rychle z jednoho místa získat jednoduchou i složitou infrastrukturu na vyžádání.

1.2. HPE a open source OpenStack

HPE není pouze v roli někoho, kdo sesbírá open source komponenty a zabalí je do produktového provedení, které jednoduše nasadíte včetně potřebné podpory, ale je i jedním z nejaktivnější vývojářů v rámci celého projektu. Z veřejně dostupných zdrojů (www.stackalytics.com) se můžete přesvědčit o aktuálních příspěvcích různých firem – HPE je obvykle na prvním nebo druhém místě. Zákazník tak získává nejen OpenStack v odladěné a supportovatelné formě, ale řešení od firmy, která přímo provádí vývoj. Dostáváte tedy skutečný produkt vytvořený na základě hlubokých znalostí samotného kódu – žádné experimenty. HPE má v OpenStack komunitě téměř 200 svých vývojářů a předsedá řadě projektů. HPE Helion OpenStack nenahrazuje open source komponenty svým proprietárním kódem – získáváte skutečnou otevřenost a přenositelnost aplikací, znalostí, zkušeností i skriptů.

1.3. HPE Helion portfolio

Kromě HPE Helion OpenStack je v Helion rodině k dispozici i řada dalších nástrojů:

- HPE Helion CloudSystem – integrované řešení zahrnující Helion OpenStack a komerční nadstavby jako je HPE Cloud Service Automation a HPE Operation Orchestration. Toto řešení navíc přináší interface pro laické uživatele, servisní katalog, velkou orchestraci (včetně procesních záležitostí typu schvalování zdrojů nadřízeným apod.) a skutečně hybridní vlastnosti s podporou nejen Helion OpenStack, ale i OpenStack třetích stran, Amazon, Azure nebo VMware
- HPE Helion Stackato – platforma jako služba pro vývoj moderních aplikací postavená na Docker kontejnerech a Cloud Foundry PaaS, kterou si můžete nainstalovat kamkoli – do OpenStack, na bare metal, do VMware i Amazonu.
- HPE Development Platform – PaaS nadstavba pro Helion OpenStack přinášející prostředí pro vývoj a provoz moderních aplikací (HPE Helion Stackato) společně s integrovanými OpenStack projekty pro PaaS jako je DBaaS nebo MSGaaS a Helion Cloud Engine pro integrovaný CI/CD vývoj a provoz aplikací
- HPE Helion Eucalyptus – open source on-premise cloud postavený na Amazon API (vaše lokální kopie Amazonu)
- HPE Helion Content Depot – integrované řešení hardware+software pro objektovou storage postavenou na OpenStack Swift
- HPE Helion Rack – integrované řešení hardware+software pro rychlé získání kompletní Helion OpenStack + Development Platform cloudu pro vaše datové centrum
- HPE Helion CloudSystem pro HPE ConvergedSystem – integrované řešení hardware+software pro kompletní enterprise IaaS+PaaS řešení včetně 3PAR storage

1.4. HPE Helion v kombinaci s dalšími produkty HPE

HPE Helion je možné kombinovat s další nabídkou HPE produktů. Kromě logického možnosti použít precizní HPE hardware pro storage, compute i networking se nabízí řada dalších návazností:

- HPE OneView (nástroj pro řízení fyzické Composable Infrastructure) je integrovaný s Helion CloudSystem
- HPE Cloud Service Automation a Operation Orchestration umí ovládat HPE Helion OpenStack
- HPE Distributed Cloud Networking (overlay SDN pro náročné zákazníky) podporuje integraci s Helion OpenStack
- HPE Virtualization Perfomance Viewer podporuje sledování výkonu vašeho OpenStack prostředí

2. Základní práce s Helion OpenStack GUI

2.1. IP adresy a přístup do labu

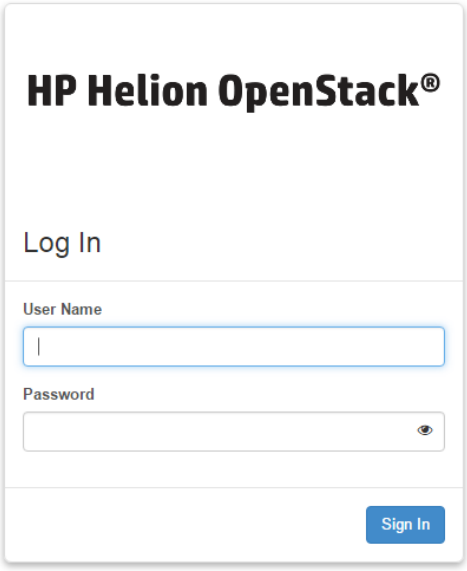
Pro lab budete potřebovat jméno a heslo do svého projektu a dále počítač s webovým prohlížečem a programem Putty (nebo jiným SSH klientem).

Aktuální adresace je následující:

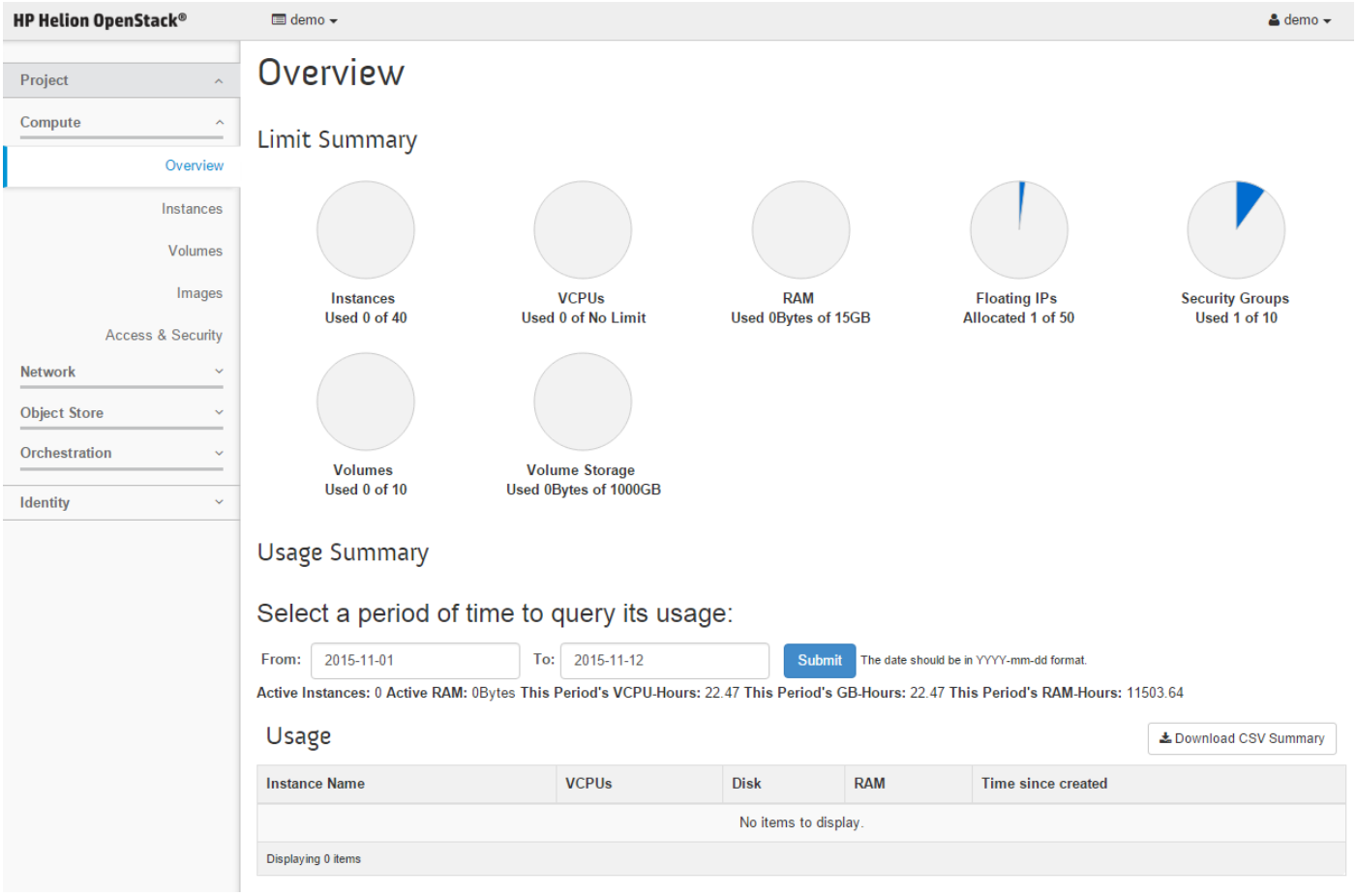
Helion OpenStack – https://16.21.188.203
labServer (SSH) – 16.21.188.201 na portu 9022
labServer (RDP) – 16.21.188.201

2.2. Získejte VM během chvilky

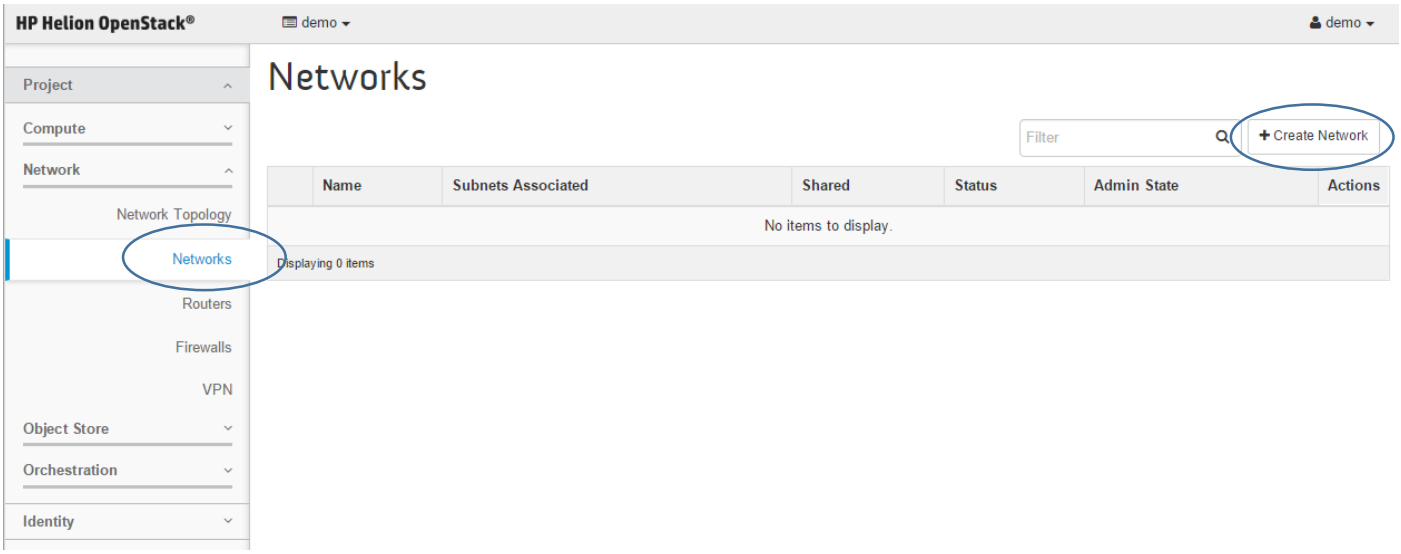
Otevřete prohlížeč, zadejte URL z části lab guide 2.1 a přihlaste se do Helion OpenStack jménem a heslem, které v rámci labu dostanete. Ten je namapován na projekt (někdy se používá terminologie „tenant“, ale jde o totéž).



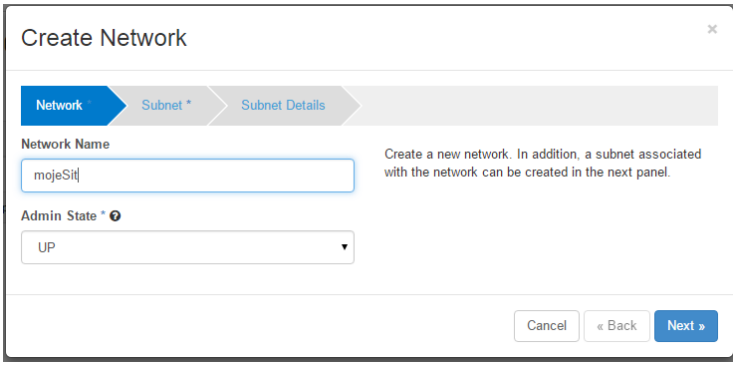
Přivítá vás dashboard s přehledem vašeho projektu. V levé části jsou dvě základní položky menu – Project, kde se budeme pohybovat celou dobu a Identity, kde v první části labu nemáte práva něco měnit. V rámci části Project je součástí Helion OpenStack položka Compute, Network, Object Store a Orchestration. Ostatní položky (pokud tam budou) patří do Helion Development Platform (dodatečná nadstavba), která se zaměřuje na Platform as a Service a není předmětem dnešního labu.



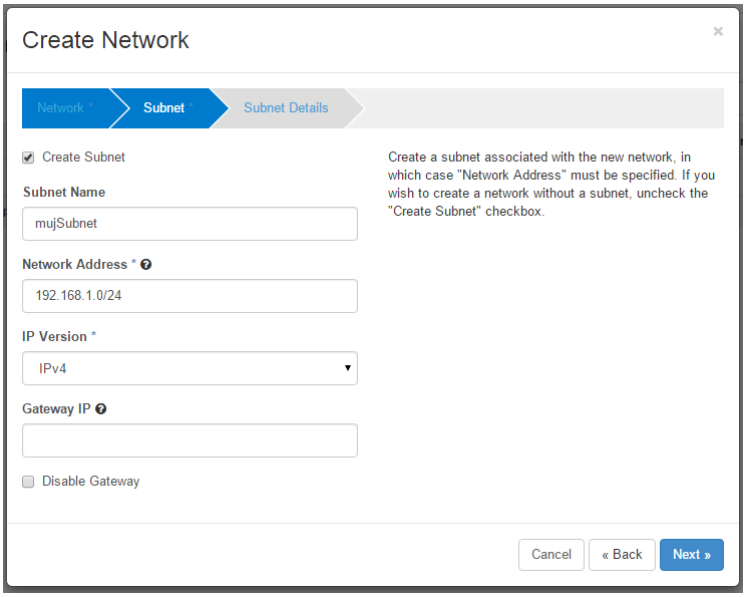
Nejprve si vytvořme nějakou privátní síť, která bude jen pro nás. Jděte do záložky Network a klikněte na Networks.



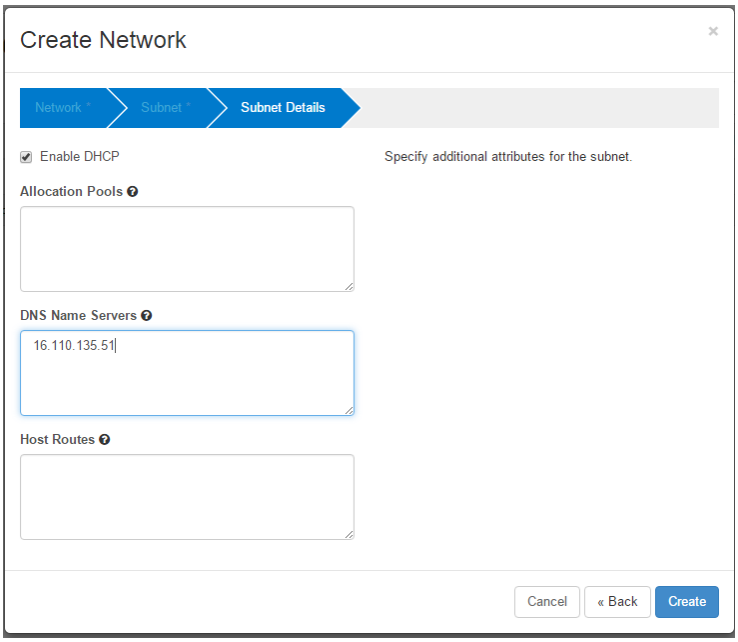
Klikněte na Create Network. Dejte síti název a klikněte na Next.



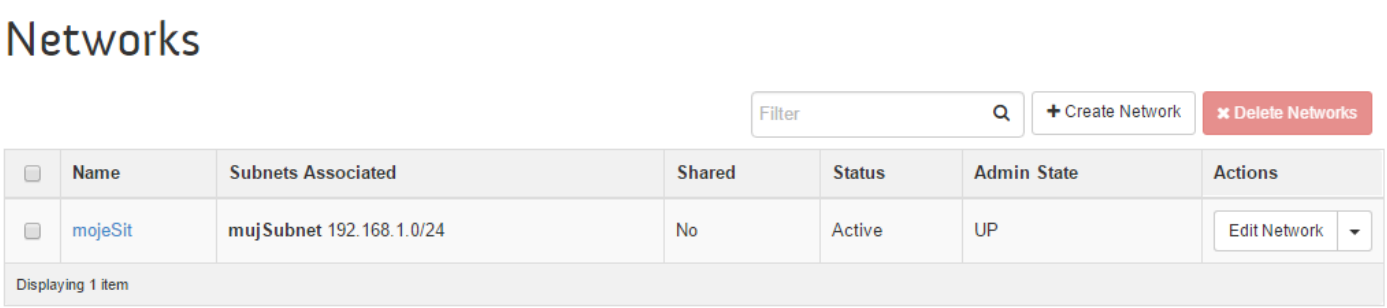
Pojmenujte nový subnet a zadejte jeho IP rozsah. Protože jde o vaší privátní síť, můžete volit zcela libovolně nezávisle na ostatních projektech/tenantech. Pak klikněte na Next.



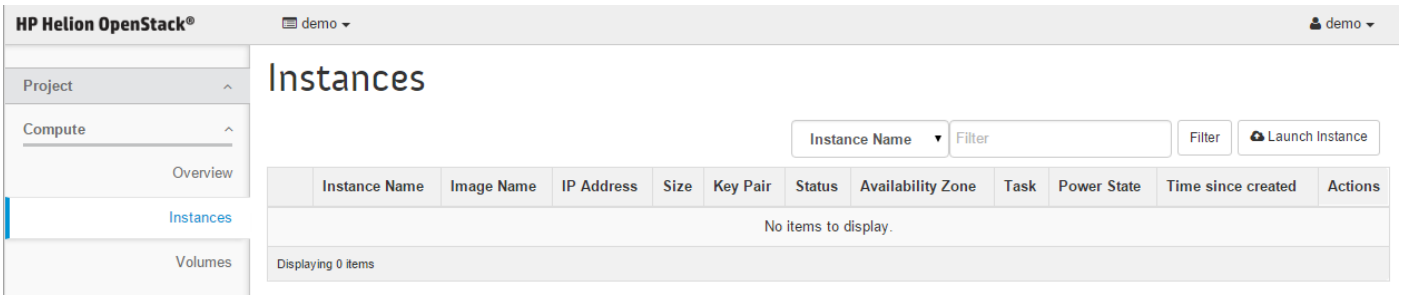
Můžeme definovat pouze určité rozsahy, posílat VM speciální routy, ale my pouze definujeme DNS server na adrese 16.110.135.51. Klikněte na Create.



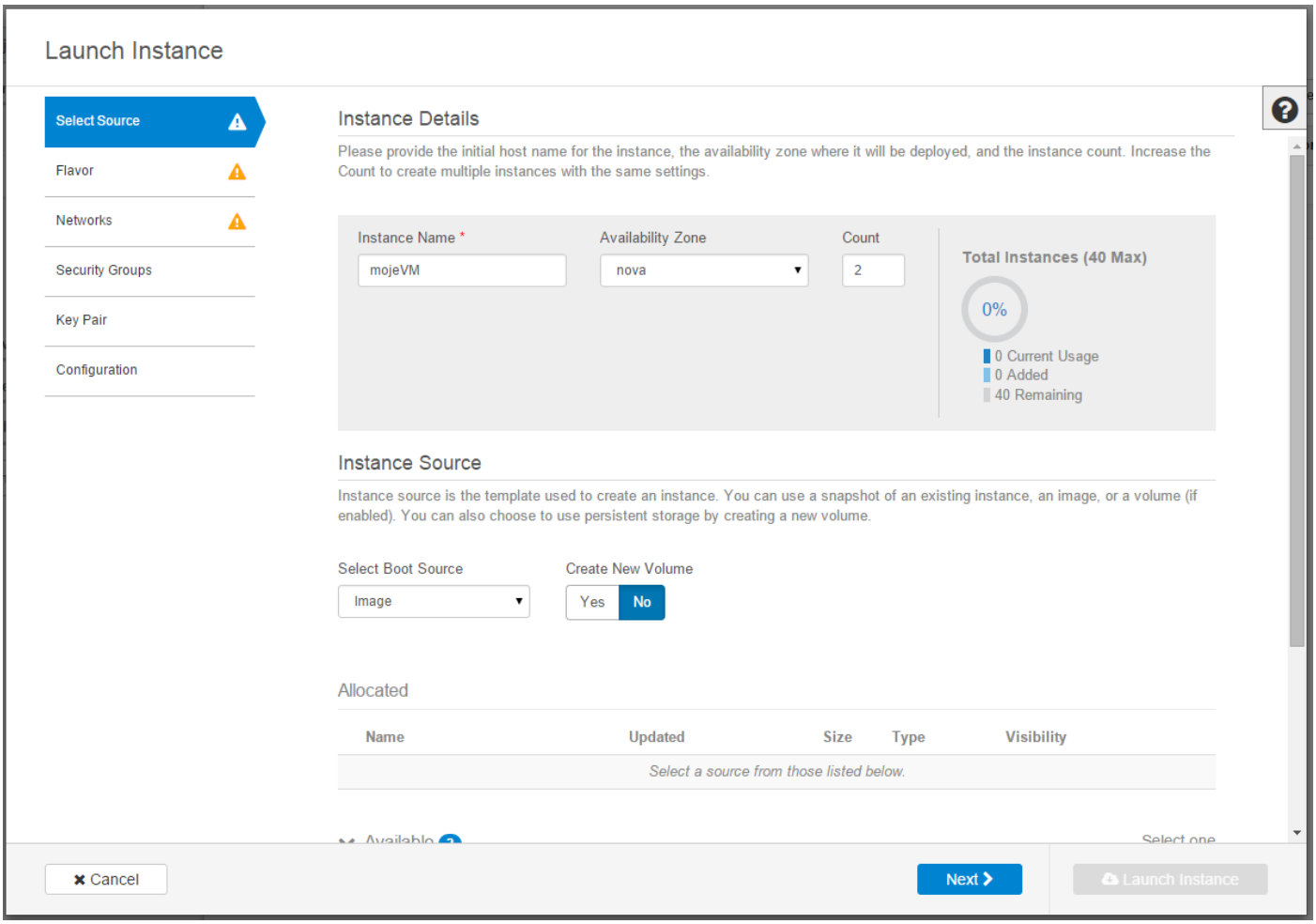
Síť máme připravenu.



Teď už můžeme vytvořit nějaké instance VM. Jděte do záložky Compute, Instances a klikněte na tlačítko Launch Instance.



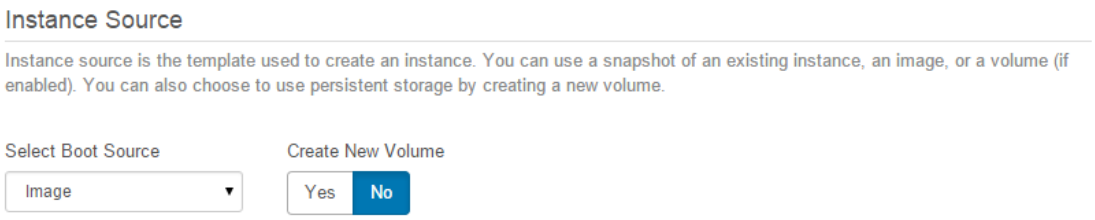
V průvodci zadejte název instance a vytvoříme si rovnou dvě.



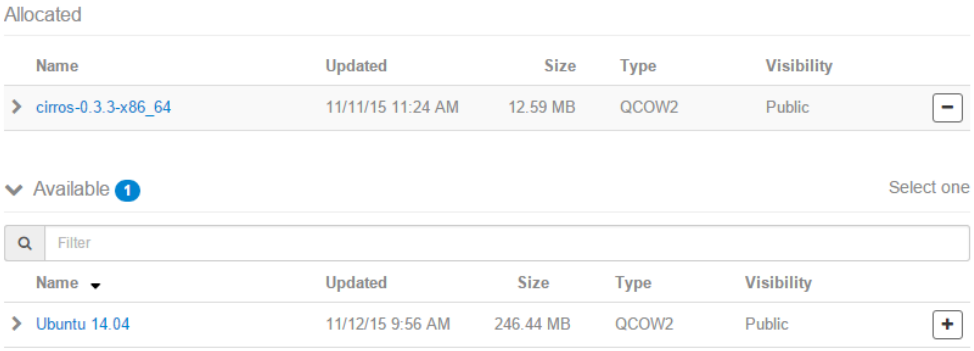
Můžeme se vybrat, zda chceme instanci vytvořit z Image, Instance snapshotu, z existující Volumu (tedy boot ze storage) nebo Volume snapshotu. Přepínačem také můžeme říct, zda se má v jednom kroku vytvořit bootovací

Volume ve storage, nebo zda chceme nastartovat VM z image, který se lokálně vytvoří v compute nodu, který OpenStack vybere.

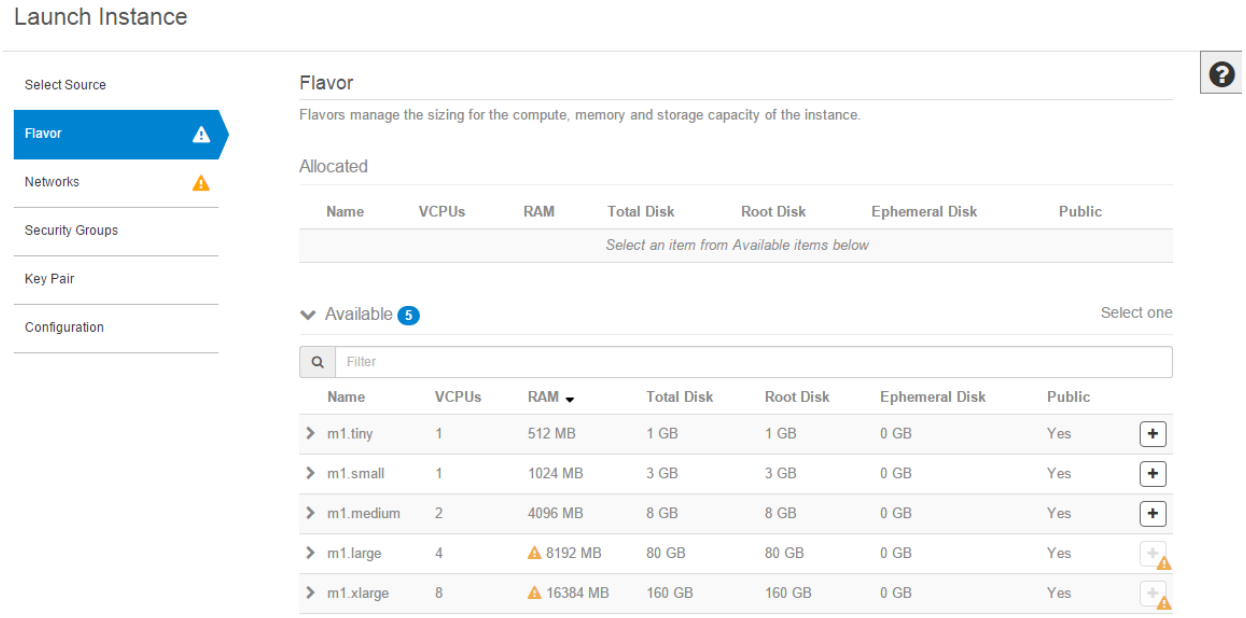
Použijeme výchozí hodnoty – tedy boot z Image, která se dostane lokálně do compute nodu.



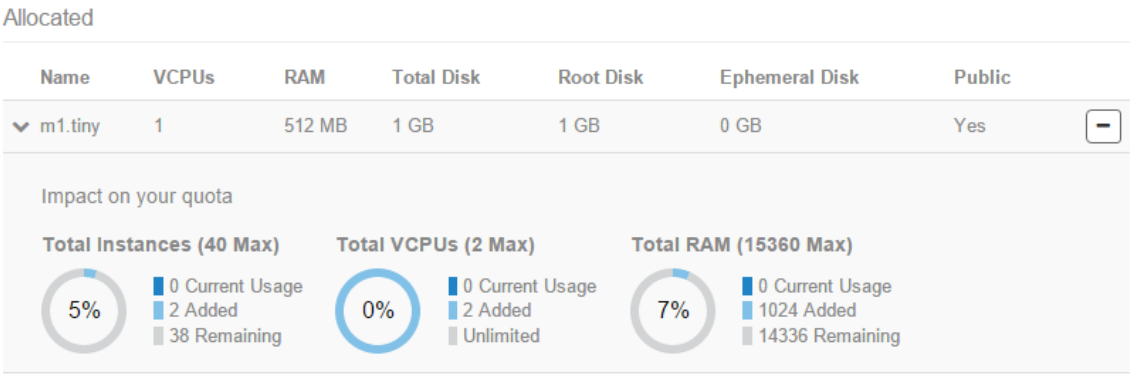
Vyberte si šablonu OS, kterou chcete nabootovat (přidejte ji kliknutím na symbol +). V našem případě to bude Cirros (mini-Linux vhodný pro první zkoušení).



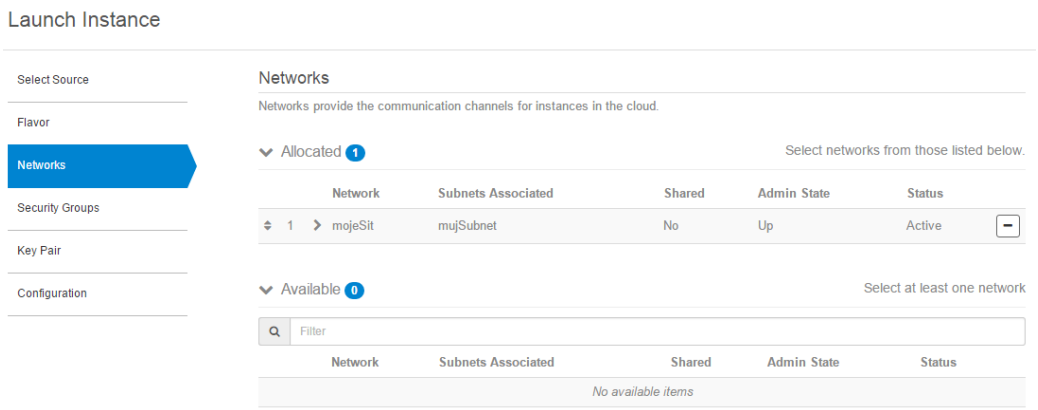
Teď už můžeme kliknout na Next.



V tomto kroku si vyberte jak velká má VM být. To co vidíte bylo pro vás, tedy uživatele v nějakém projektu/tenantu, definováno administrátorem. OpenStack vám také ukazuje, která velikost se ještě vejde do zdrojů, které vám zbývají (administrátor všemu projektu přidělil nějakou kvótu na různé zdroje – vCPU, paměť, storage, sítě, ...). Vyberte m1.tiny kliknutím na symbol +.

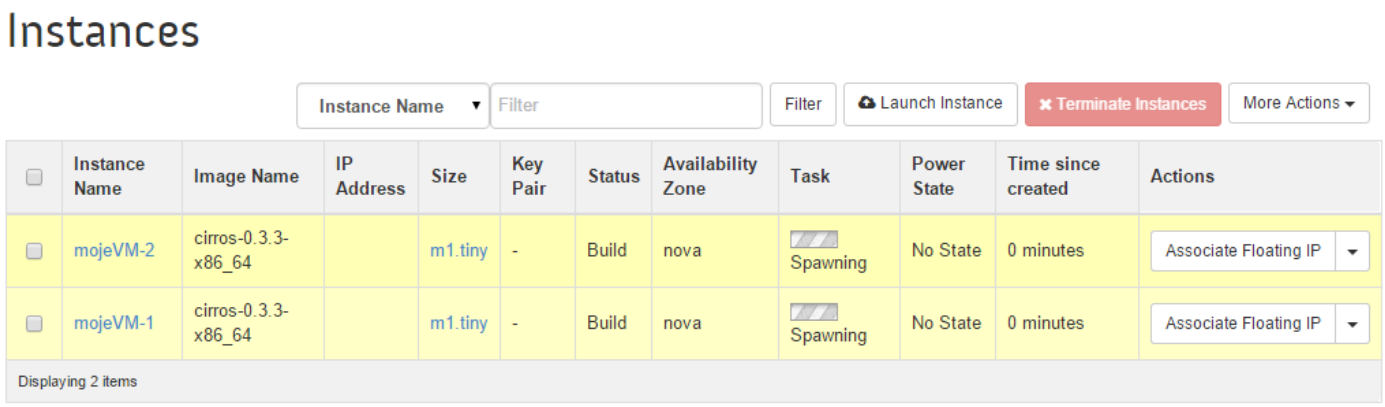


Můžeme kliknout na Next. V následujícím dialogu vyberte naší novou síť.



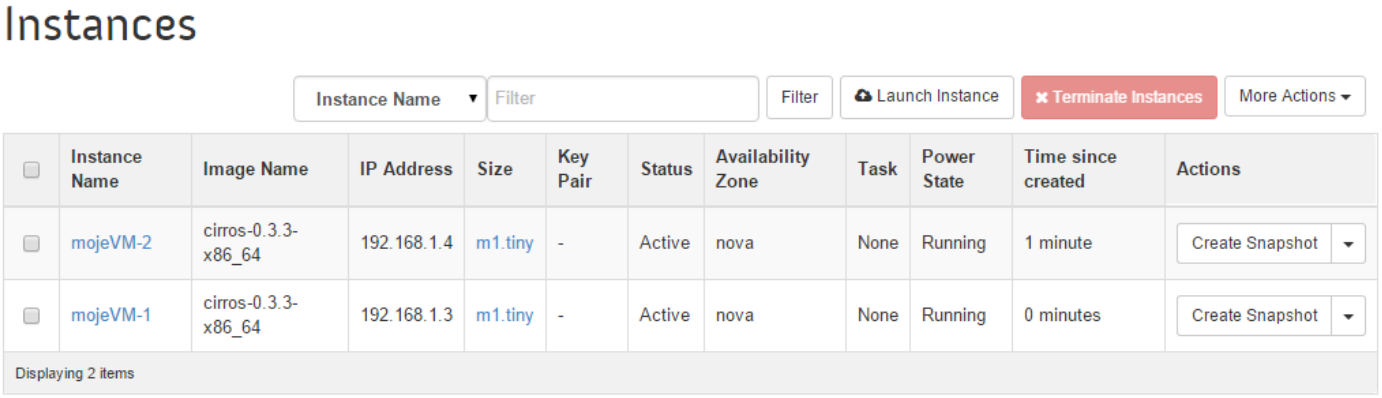
Další položky nejsou povinné a budeme se jim věnovat později. Teď už tedy můžeme kliknout na zelené tlačítko Launch Instance.

Instance se začínají vytvářet, než budou plně spuštěné, pozadí bude žluté.

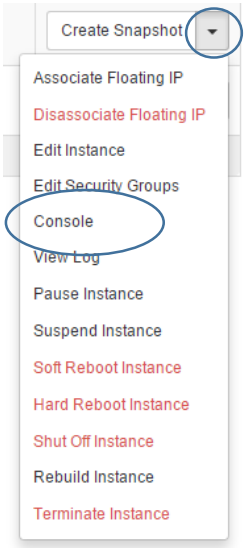


OpenStack scheduler si v našem případě vybral vhodný compute node, kde je dostatek zdrojů, pak překopíroval image soubor na jeho lokální disk a z toho image nabootovala VM.

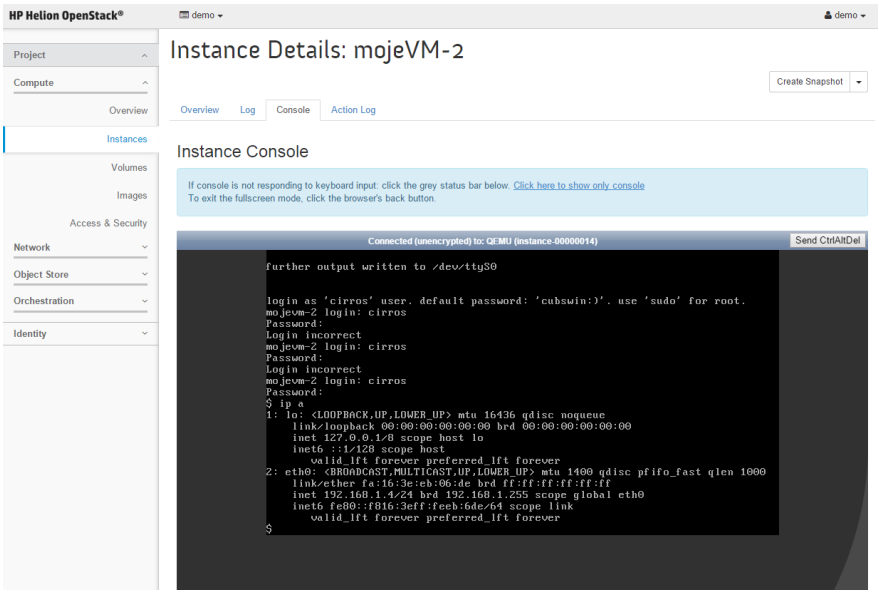
Po nějaké době budou VM připravené.



Všimněte si IP adres. Podívejme se do konzole jedné z VM. Klikněte na symbol šipky a vyberte Console.



V konzoli se můžete přihlásit účtem cirros s heslem cubswin:)



Příkazem „ip a“ se můžete podívat svojí IP adresu.

Co se to vlastně právě teď stalo?

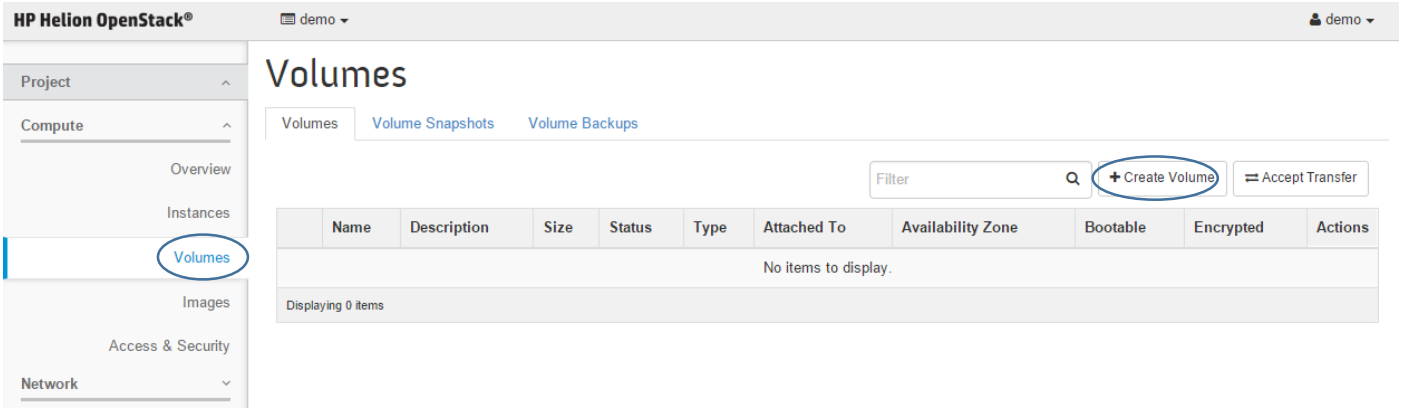
- Helion OpenStack vzal v úvahu potřeby zdrojů (příchuť) a našel vhodný compute node (tedy železo) pro vytvoření VM
- Námi vybraný image (Cirros) se nakopíroval na compute node, aby z něj VM mohla nabootovat
- Vytvořila se příslušná VM v compute node (v našem labu KVM hypervisor)
- Použila se naše privátní síť aniž by bylo nutné něco nastavovat ve fyzické síti (VXLAN enkapsulace je transparentní vůči fyzické síti kterou spolu mohou komunikovat tyto dvě VM i když jsou třeba na dvou různých compute nodech)
- Virtuální síťová karta VM kouká do této privátní sítě

2.3. Bloková storage, Volume a diskové obrazy

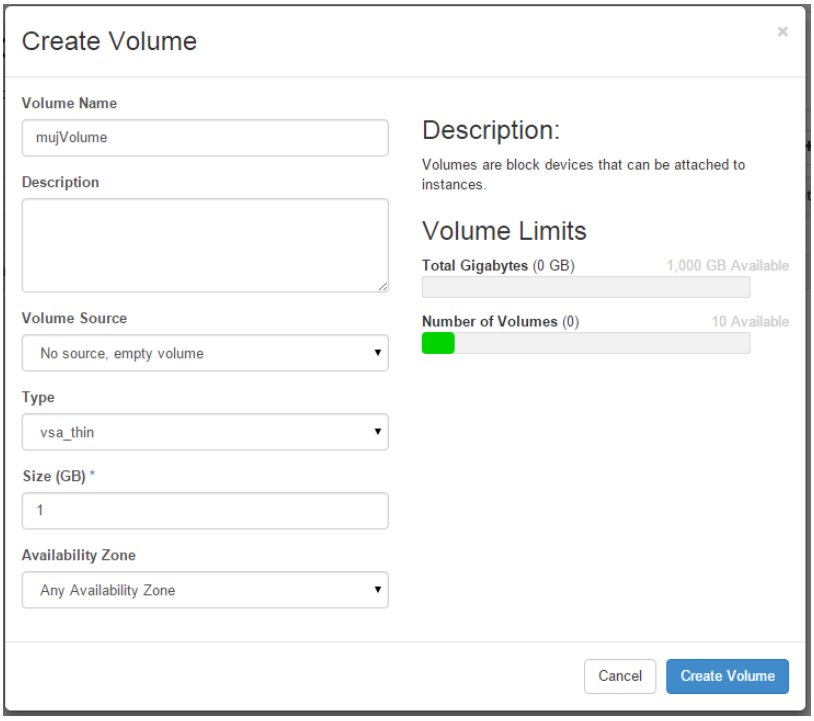
2.3.1. Bloková storage jako datový disk

Instance z předchozího příkladu používá ephemeral disky, tedy „jak přijde, tak odejde“. V okamžiku, kdy instanci zrušíte, vygumují se všechny zabrané zdroje a vrátí se k dalšímu využití. Z toho je patrné, že můžeme chtít aplikační stav ukládat na nějaké trvalejší místo – databázi, objektové úložiště nebo blokový volume na storage. A to si právě teď vyzkoušíme na příkladu StoreVirtual VSA (ale ovládání je identické při použití 3PAR nebo storage třetí strany).

Navštivte záložku Volumes a klikněte na Create Volume

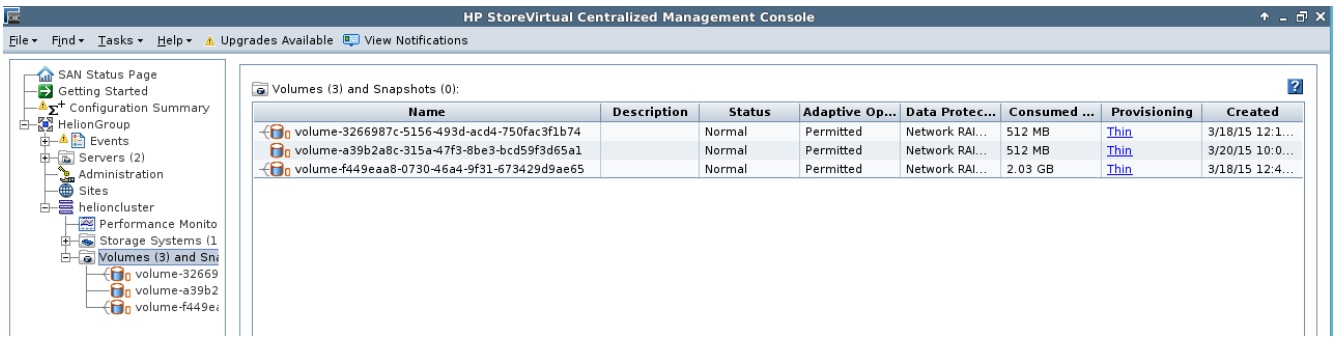


Dejte disku jméno, jako zdroj nic nepoužívejte (zatím necháme volume prázdný) a zvolte Typ (ten připravil administrátor HP Helion OpenStack – v našem případě jde o Volume na StoreVirtual s tenkým provosioningem a bez adaptivní optimalizace – víc o tom, jaké storage lze takto nabídnout v pozdější části labu). Ještě si v pravé části všimněte, že váš projekt má určitá omezení, která byla definována administrátorem Helion OpenStack.

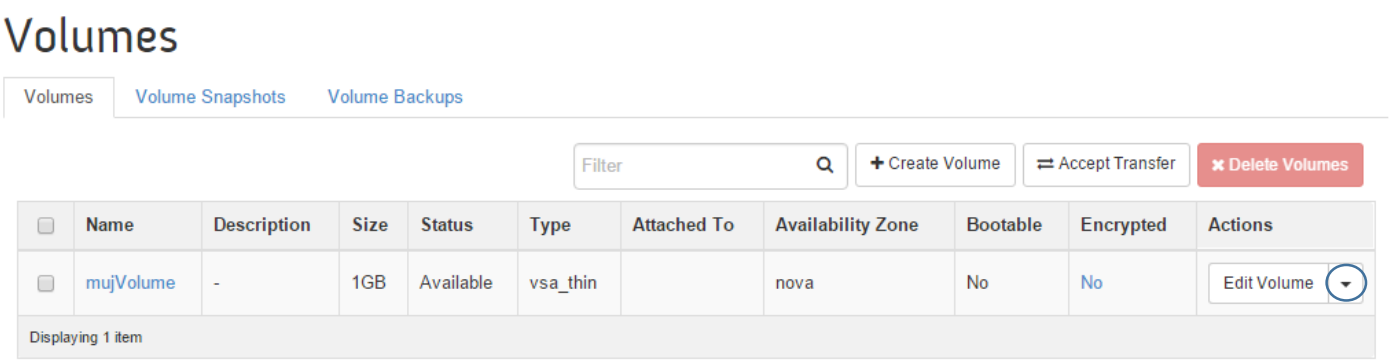


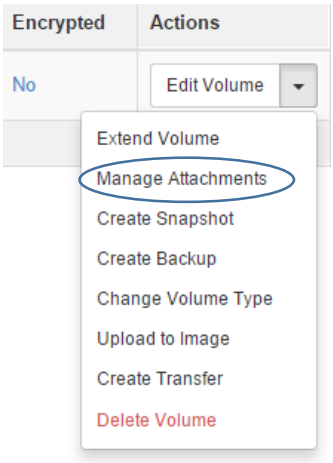
Klikněte na Create Volume

To bychom měli … v rámci labu nemáte přístup do StoreVirtual konzole, ale klidně požádejte o nahlédnutí – uvidíte zhruba toto (je to ten nejmladší a nepřipojený):

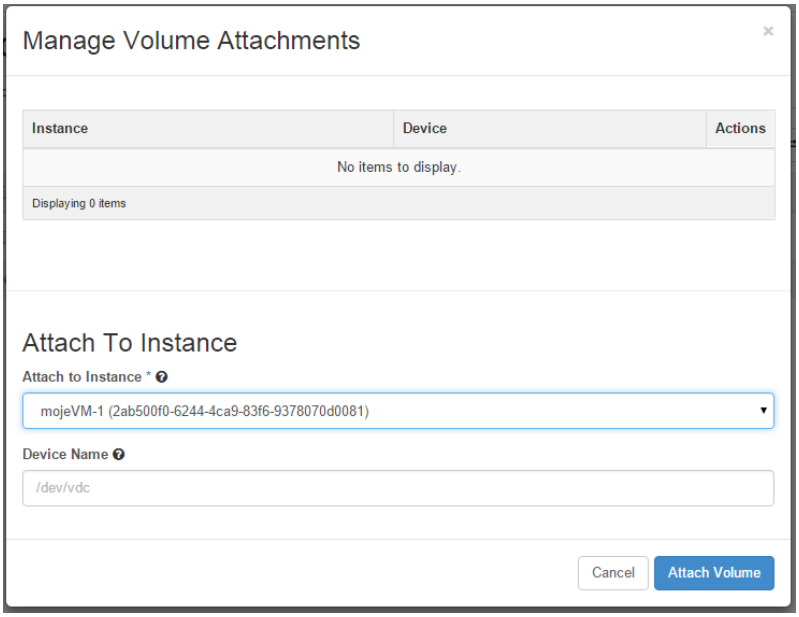


Pojďme ho tedy připojit k instanci. Klikněte na Edit Atachements

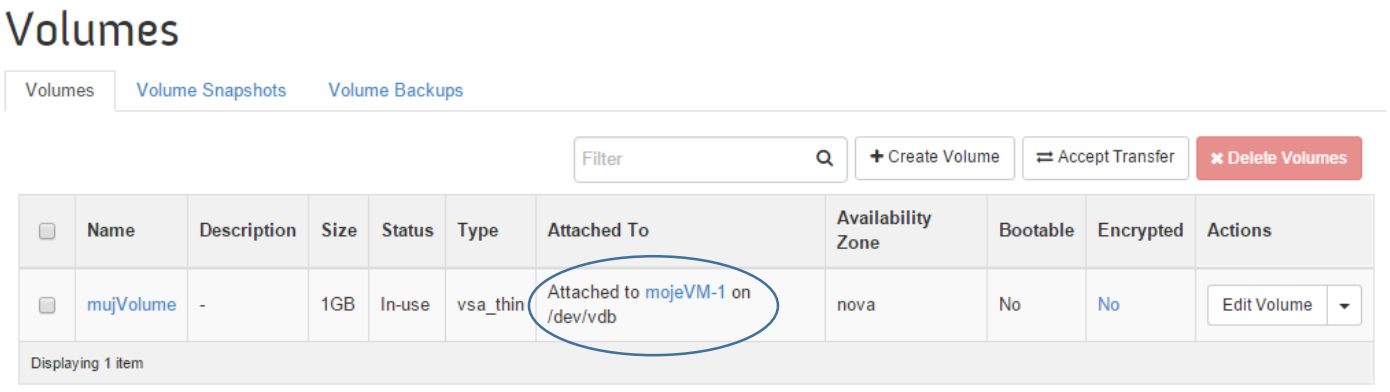




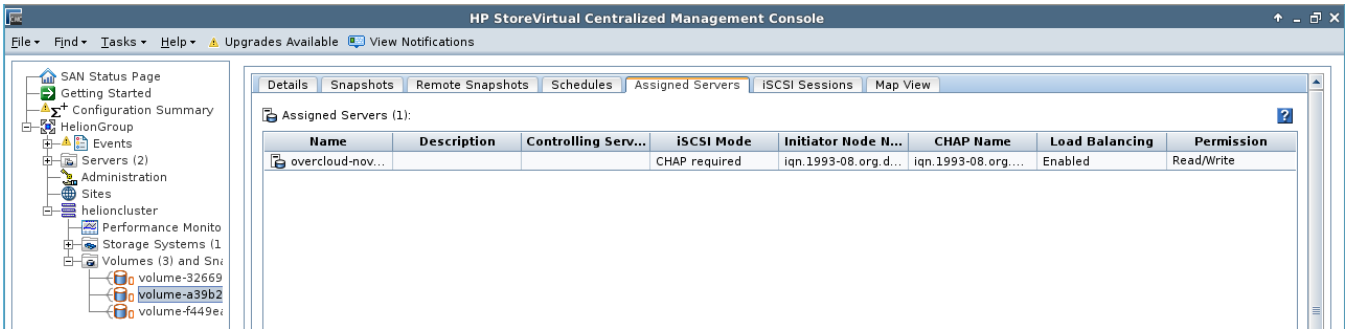
Vyberte jednu z našich instancí a klikněte na Attach Volume



Prohlédněte si výsledek



Podobně to vypadá ve StoreVirtual konzoli (tam v rámci labu přístup nemáte, ale váš průvodce vám výsledek může ukázat).



Pojďme teď na disk něco zapsat. Skočte do konzole příslušné VM, vytvořte na disk file systém, přimountujte si ho, vytvořte na něm soubor a vypište si obsah disku.

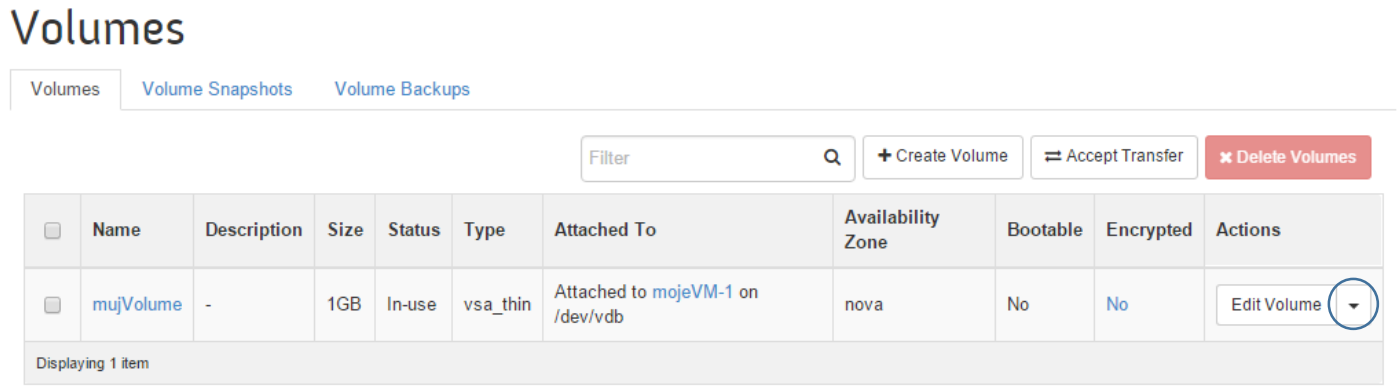
```
sudo fdisk /dev/vdb
```

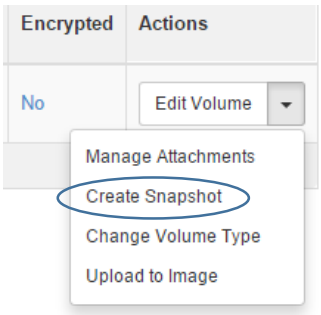
následně použijte volbu pro vytvoření nové partition – zmáčněte n + enter a na následující otázky pouze odklepněte výchozí hodnotu. Nakonec zmáčkněte w + enter, čímž se změny zapíší.

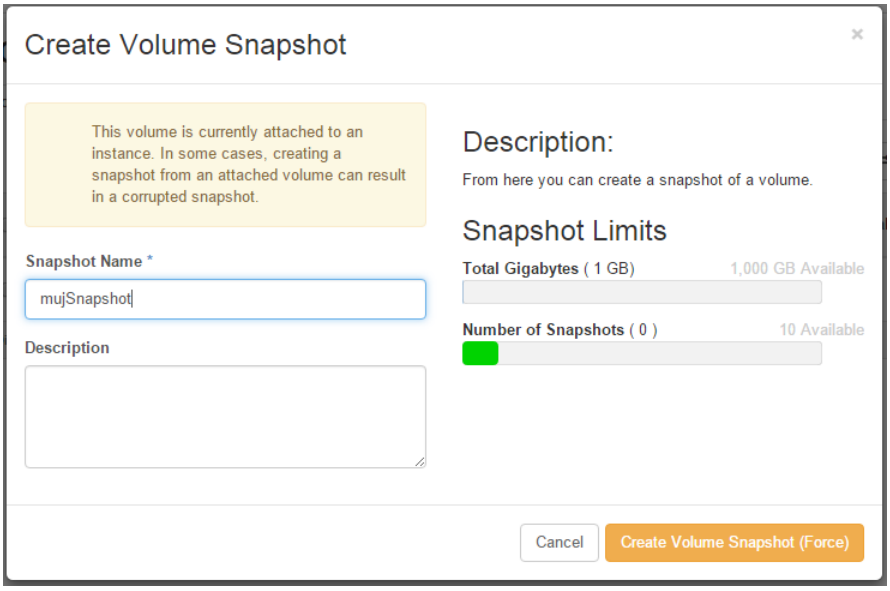
Pokračujeme dál – vytvoříme filesystem, přimountujeme si ho, vytvoříme nějaký soubor a přesvědčíme se, že tam je.

```
sudo mkfs.ext3 /dev/vdb
sudo mkdir /mujdisk
sudo mount /dev/vdb /mujdisk
sudo touch /mujdisk/mujsoubor.txt
ls /mujdisk
```

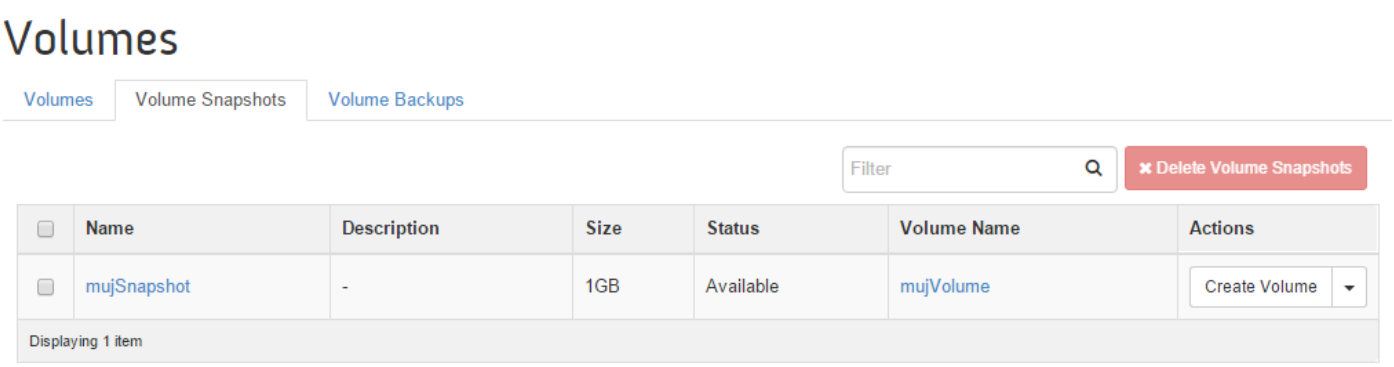
Udělejte snapshot – správně bychom měli disk odpojit, aby byla zajištěna aplikační konzistence dat. V našem případě tam ale aplikace neběží a pole pro nás udělá crash konzistentní snapshot, což nám v tuto chvíli stačí.



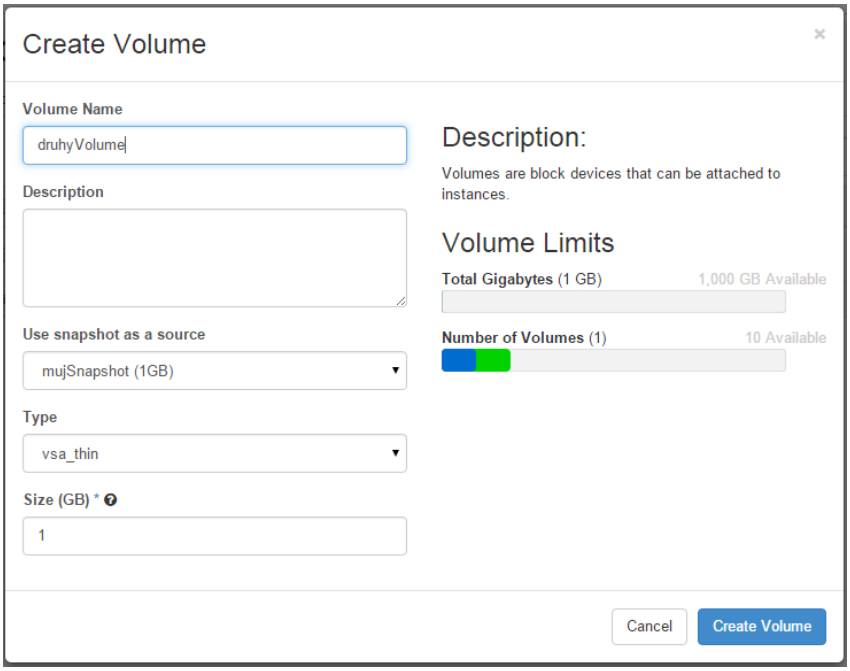




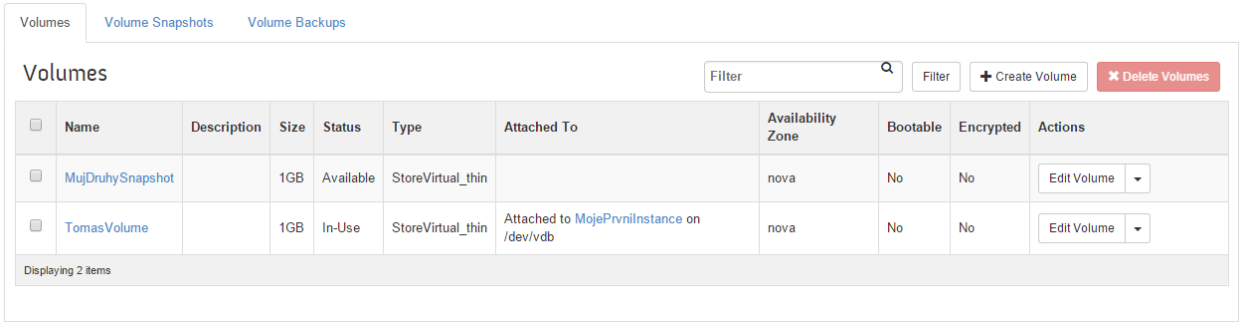
Klikněte na Create Volume Snapshot. Dostanete se na seznam snapshotů.



Vytvoříme z něj další volume



Klikněte na Create Volume



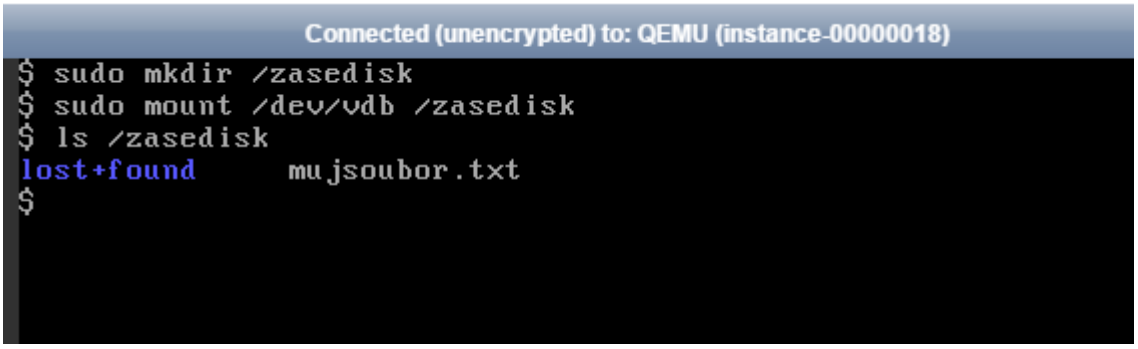
Všimněte si, že s nepřipojeným Volume lze dělat ještě další věci – například ho zvětšit nebo provést jeho backup (to se uloží jako objekt do objektové storage – o tom později), převést do jiného projektu apod. Další možnost je z Volume vytvořit nový startovací image, ale o tom později.

Připojte Volume k naší druhé VM – už víte jak na to.

Name	Description	Size	Status	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
druhyVolume	-	1GB	In-use	vsa_thin	Attached to mojeVM-2 on /dev/vdb	nova	No	No	Edit Volume
mujVolume	-	1GB	In-use	vsa_thin	Attached to mojeVM-1 on /dev/vdb	nova	No	No	Edit Volume

Displaying 2 items

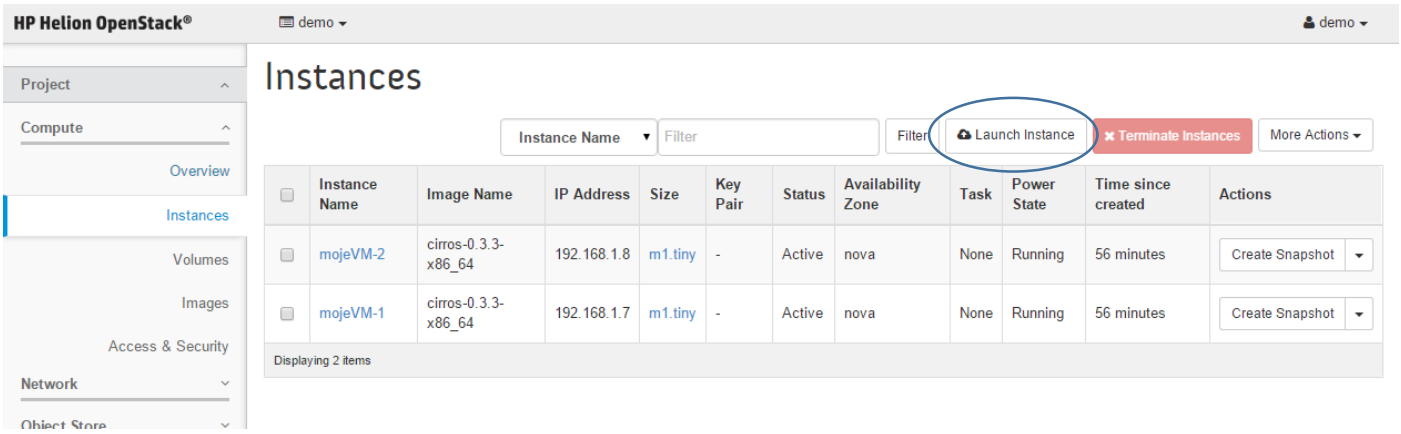
Jděte do konzole, přimapujte si file systém a podívejte se, jestli tam je náš soubor.



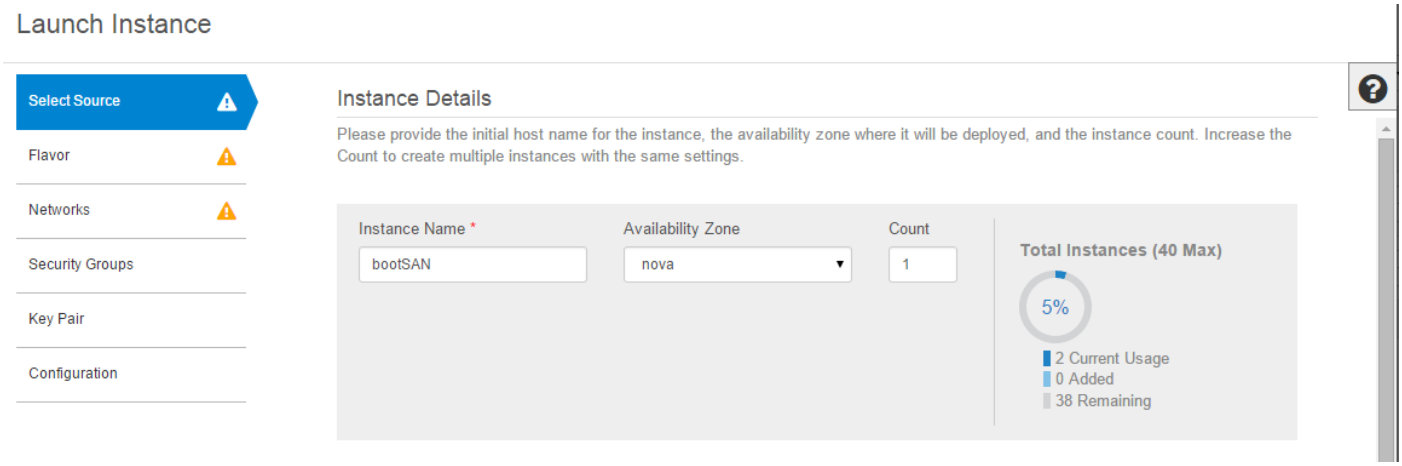
2.3.2. Bootování VM ze storage

Vytvořme si další VM a tentokrát budeme chtít, aby se vytvořil bootovací volume v naší storage, obraz se do něj nakopíroval a VM z něj nabootovala.

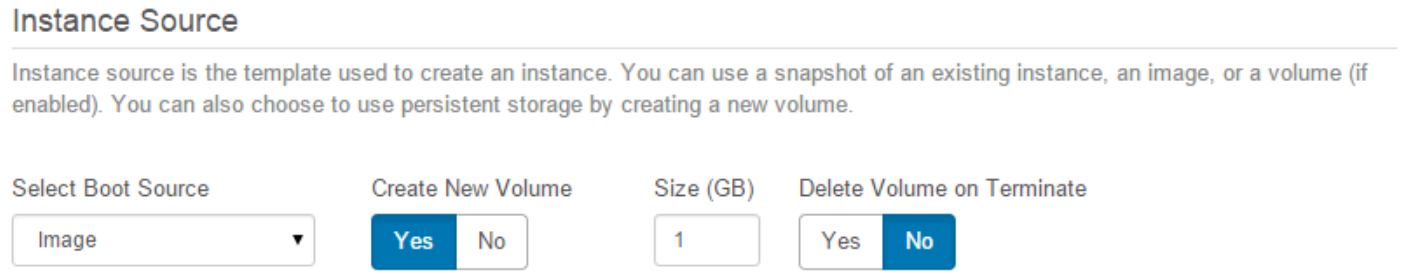
Jděte do záložky Compute, Instances a klikněte na Launch Instance.



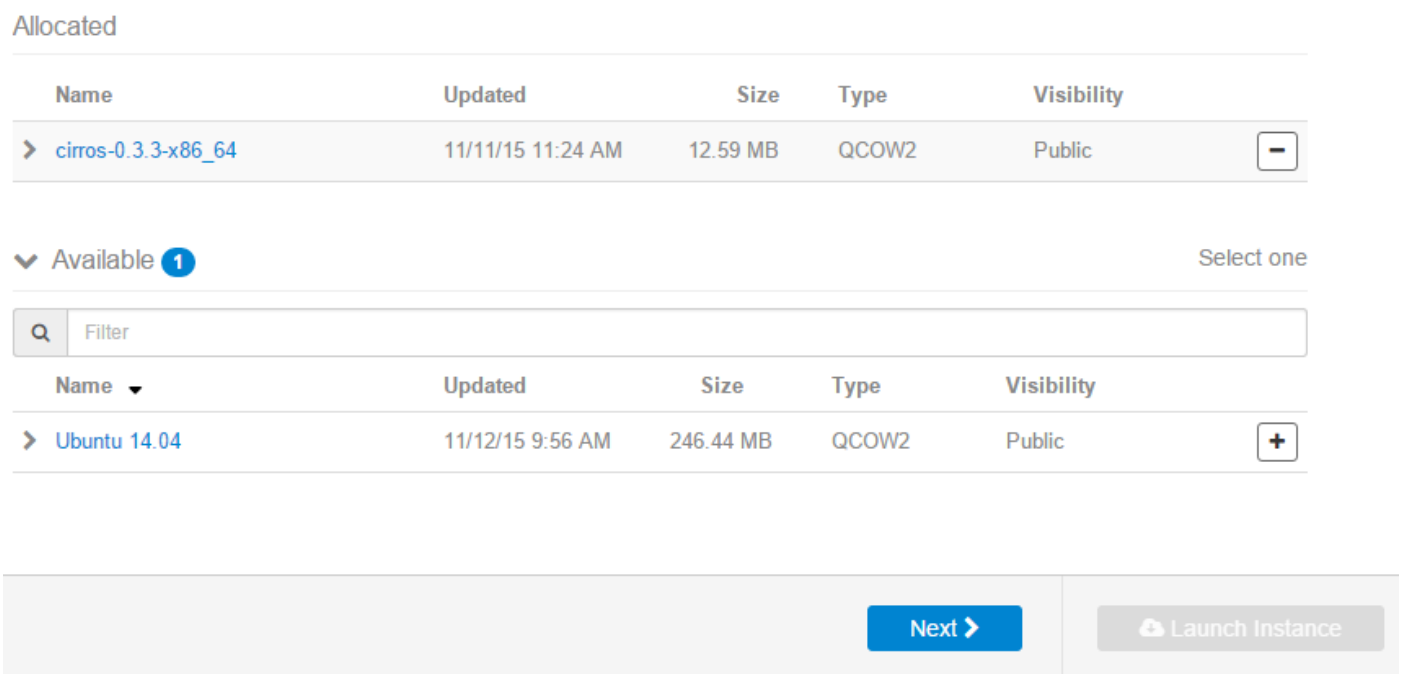
Pojmenujte vaší novou VM.



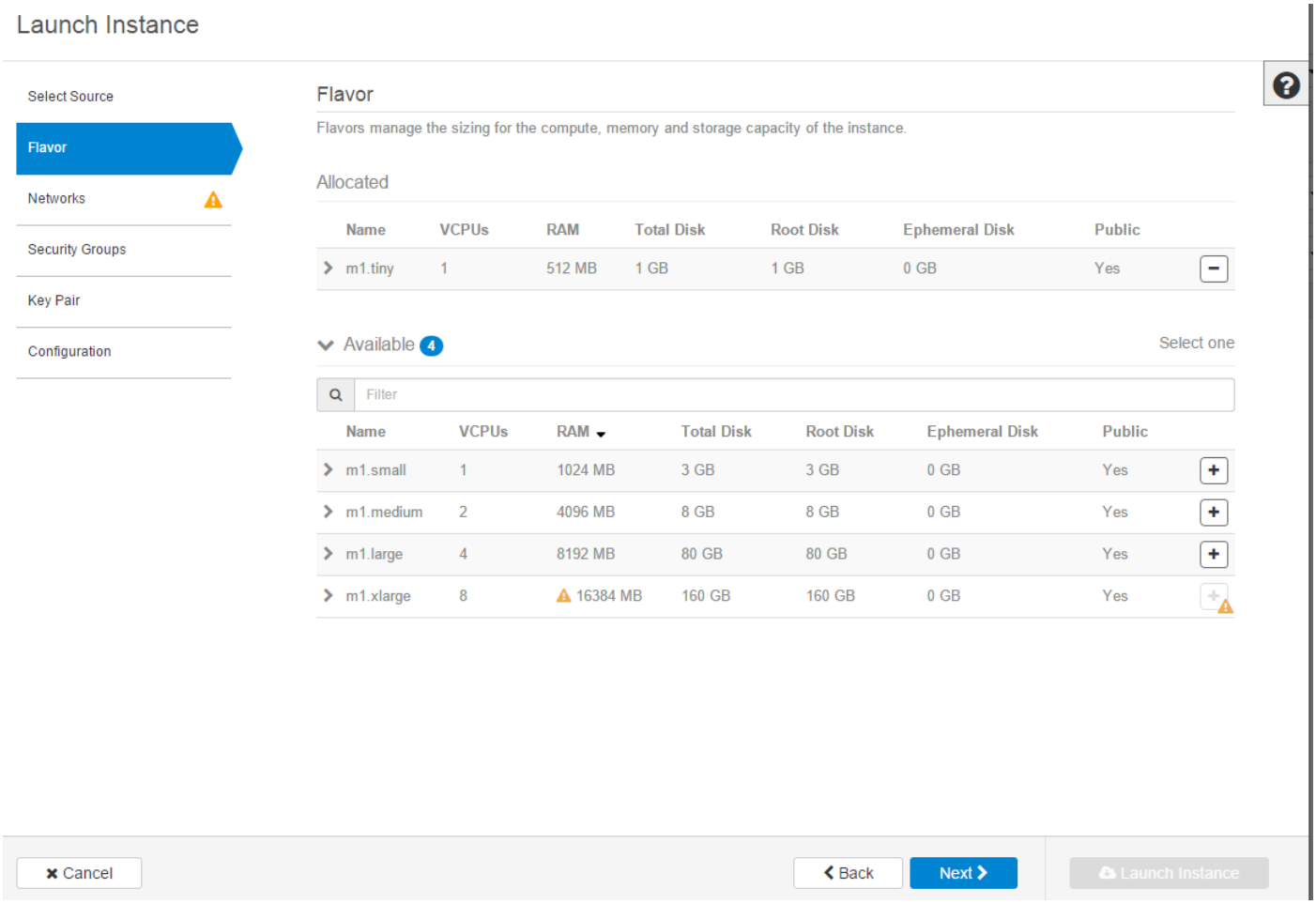
Budeme chtít bootovat z image, ale tak, že se vytvoří Volume v naší storage. Dále můžeme definovat, zda se má Volume automaticky zrušit v případě, že smažeme naší instanci (VM).



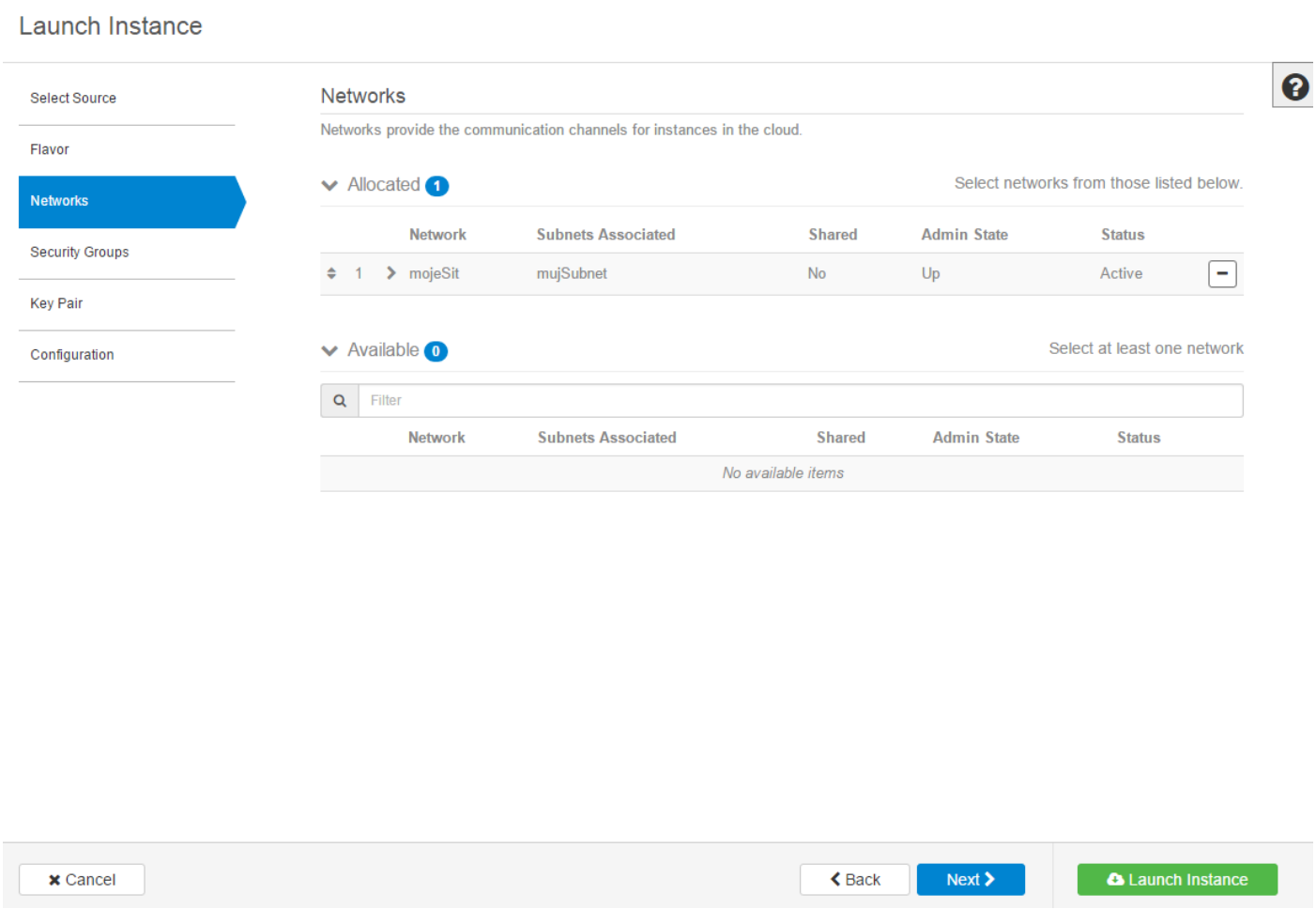
Vyberte si Cirros a klikněte Next



Použijte velikost m1.tiny a klikněte na Next



Přidejte síť a klikněte na zelené Launch Instance.



Počkejte až VM nabootuje.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
bootSAN	cirros-0.3.3-x86_64	192.168.1.9	m1.tiny	-	Active	nova	None	Running	1 minute	Create Snapshot
mojeVM-2	cirros-0.3.3-x86_64	192.168.1.8	m1.tiny	-	Active	nova	None	Running	1 hour, 2 minutes	Create Snapshot
mojeVM-1	cirros-0.3.3-x86_64	192.168.1.7	m1.tiny	-	Active	nova	None	Running	1 hour, 2 minutes	Create Snapshot

Displaying 3 items

Podívejte se na Volume, který vám průvodce vyrobil.

Volumes

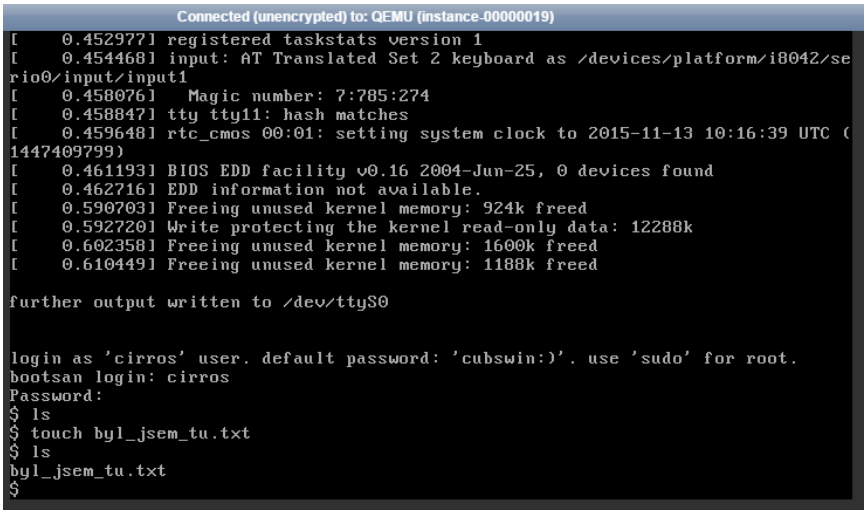
Volumes | Volume Snapshots | Volume Backups

Filter | + Create Volume | ⇄ Accept Transfer | ✖ Delete Volumes

Name	Description	Size	Status	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
46cf7f6d-cd62-44a0-a6f1-bc930e50602c	-	1GB	In-use	vsa_thin	Attached to bootSAN on /dev/vda	nova	Yes	No	Edit Volume
druhyVolume	-	1GB	In-use	vsa_thin	Attached to mojeVM-2 on /dev/vdb	nova	No	No	Edit Volume
mujVolume	-	1GB	In-use	vsa_thin	Attached to mojeVM-1 on /dev/vdb	nova	No	No	Edit Volume

Displaying 3 items

Skočte do konzole této nové VM. Teď bychom si mohli třeba něco nainstalovat nebo jinak si operační systém upravit. My děláme jedinou věc – vytvoříme v něm nějaký soubor.



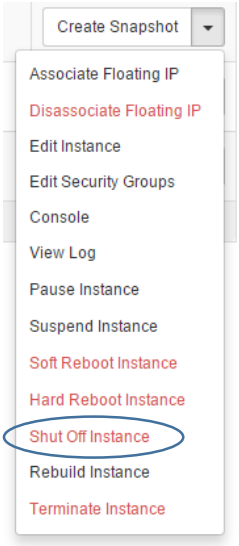
Instanci teď vypněte.

Instances

Instance Name | Filter | Filter | Launch Instance | ✖ Terminate Instances | More Actions

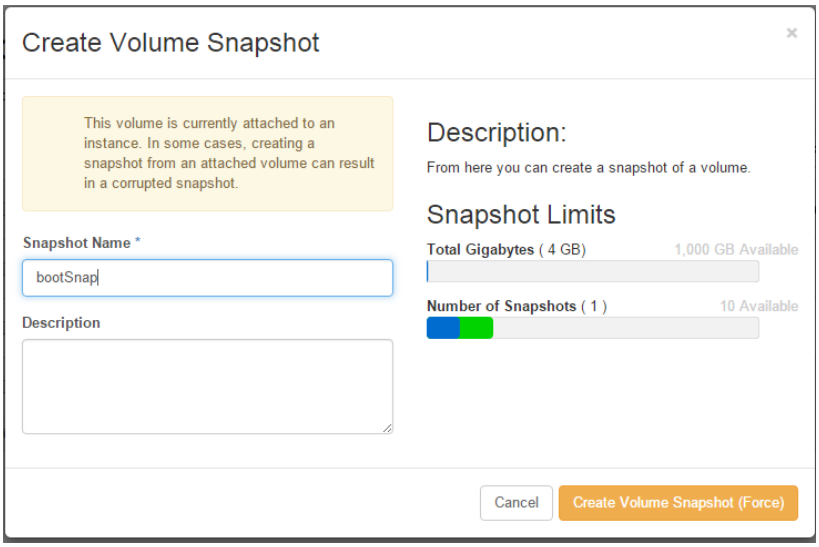
Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
bootSAN	cirros-0.3.3-x86_64	192.168.1.9	m1.tiny	-	Active	nova	None	Running	9 minutes	Create Snapshot
mojeVM-2	cirros-0.3.3-x86_64	192.168.1.8	m1.tiny	-	Active	nova	None	Running	1 hour, 12 minutes	Create Snapshot
mojeVM-1	cirros-0.3.3-x86_64	192.168.1.7	m1.tiny	-	Active	nova	None	Running	1 hour, 12 minutes	Create Snapshot

Displaying 3 items

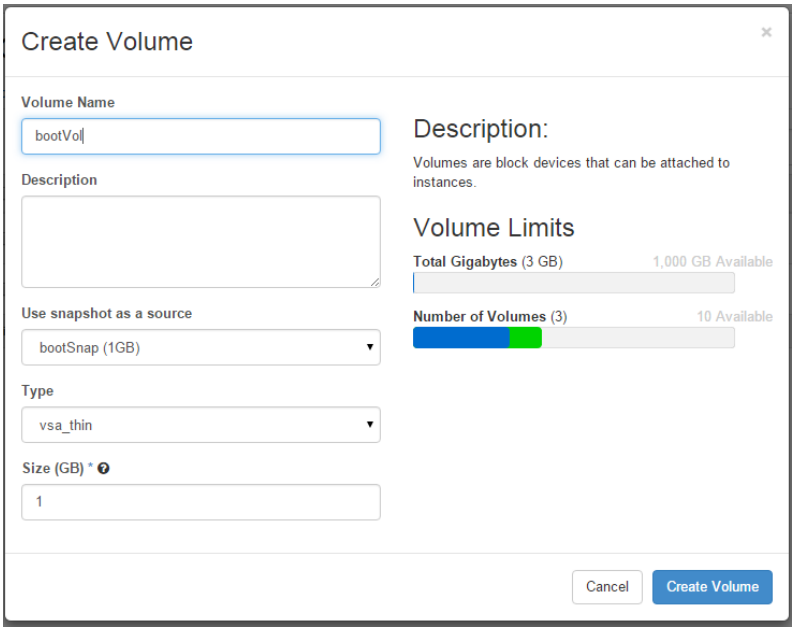


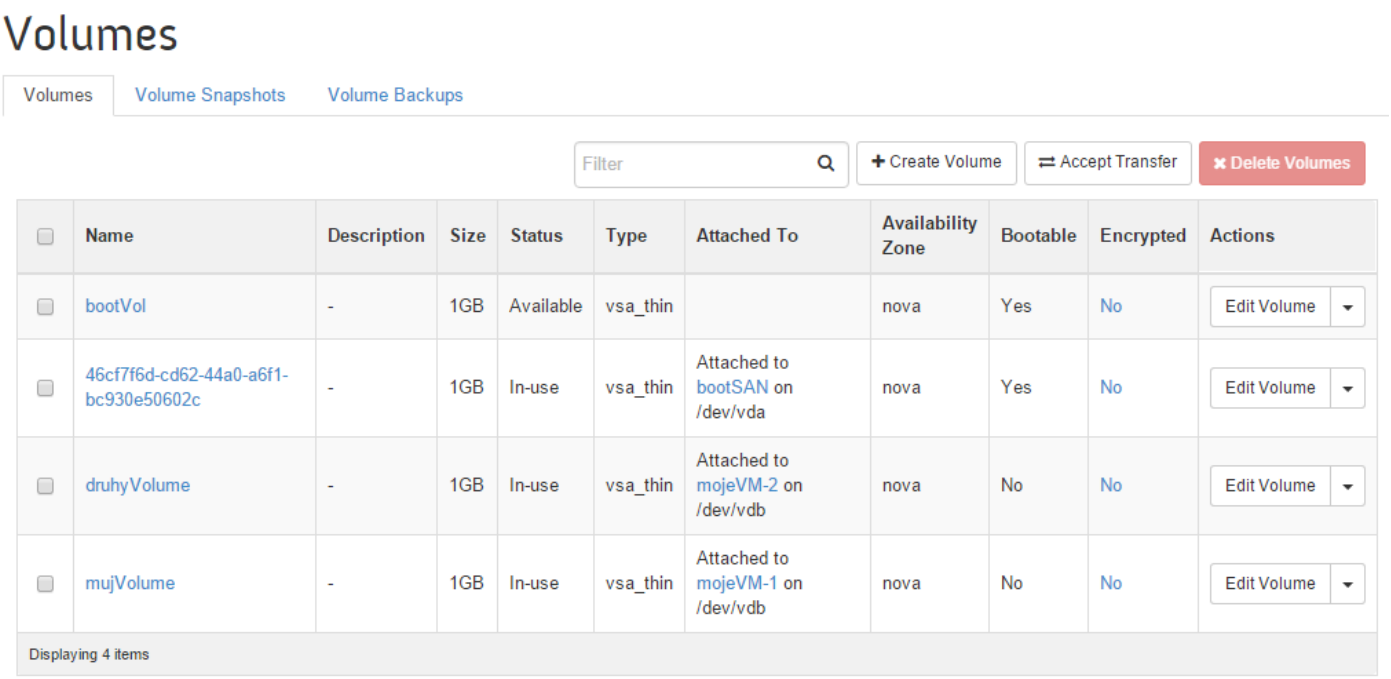
První co si zkusíme je udělat snapshot tohoto Volume a z něj vyvoříme nový bootovatelný Volume – už víte jak na to.

Vytvořte snapshot.

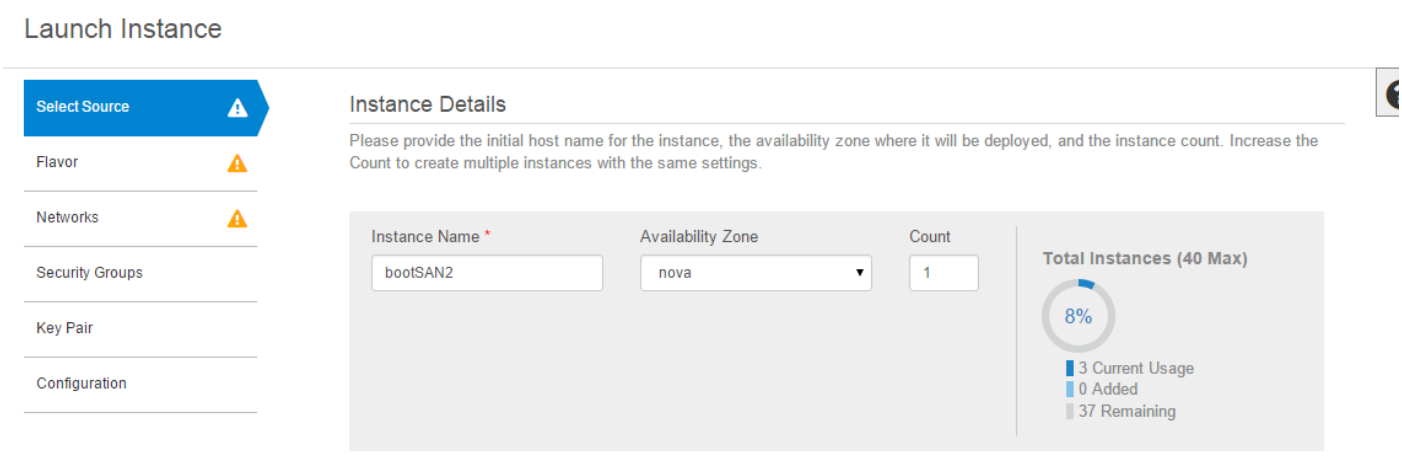


A ze snapshotu další Volume.

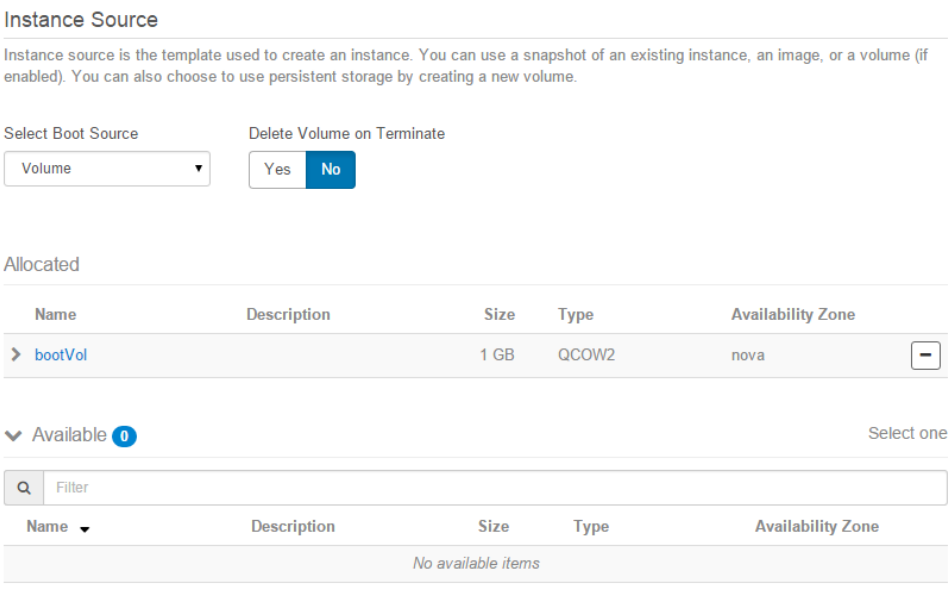




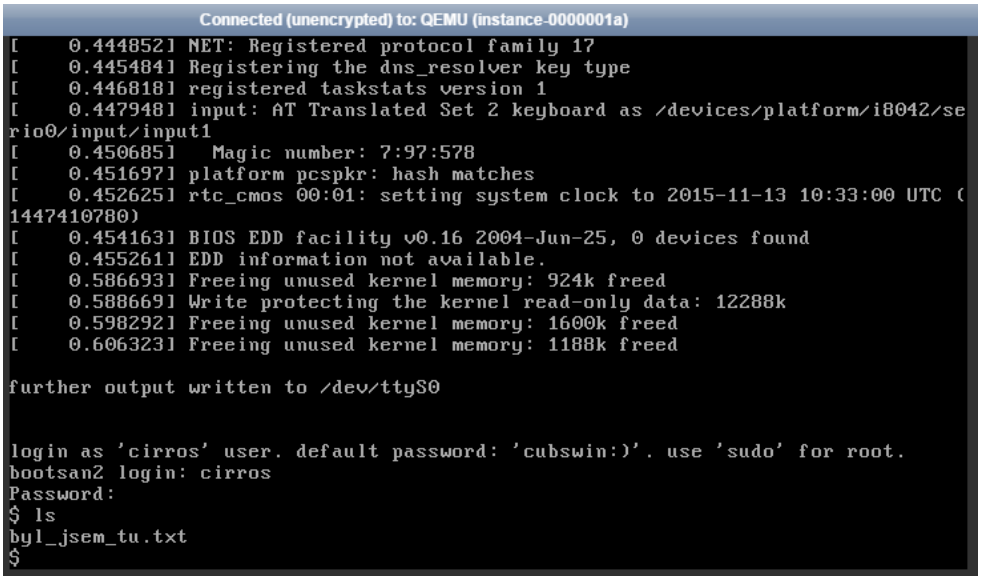
Spustíme si novou instanci, která bude bootovat z tohoto připraveného Volume ve storage. Jděte do Compute, Instances, Launch Instance.



Uveďte, že chcete startovat z Volume a vyberte ho.

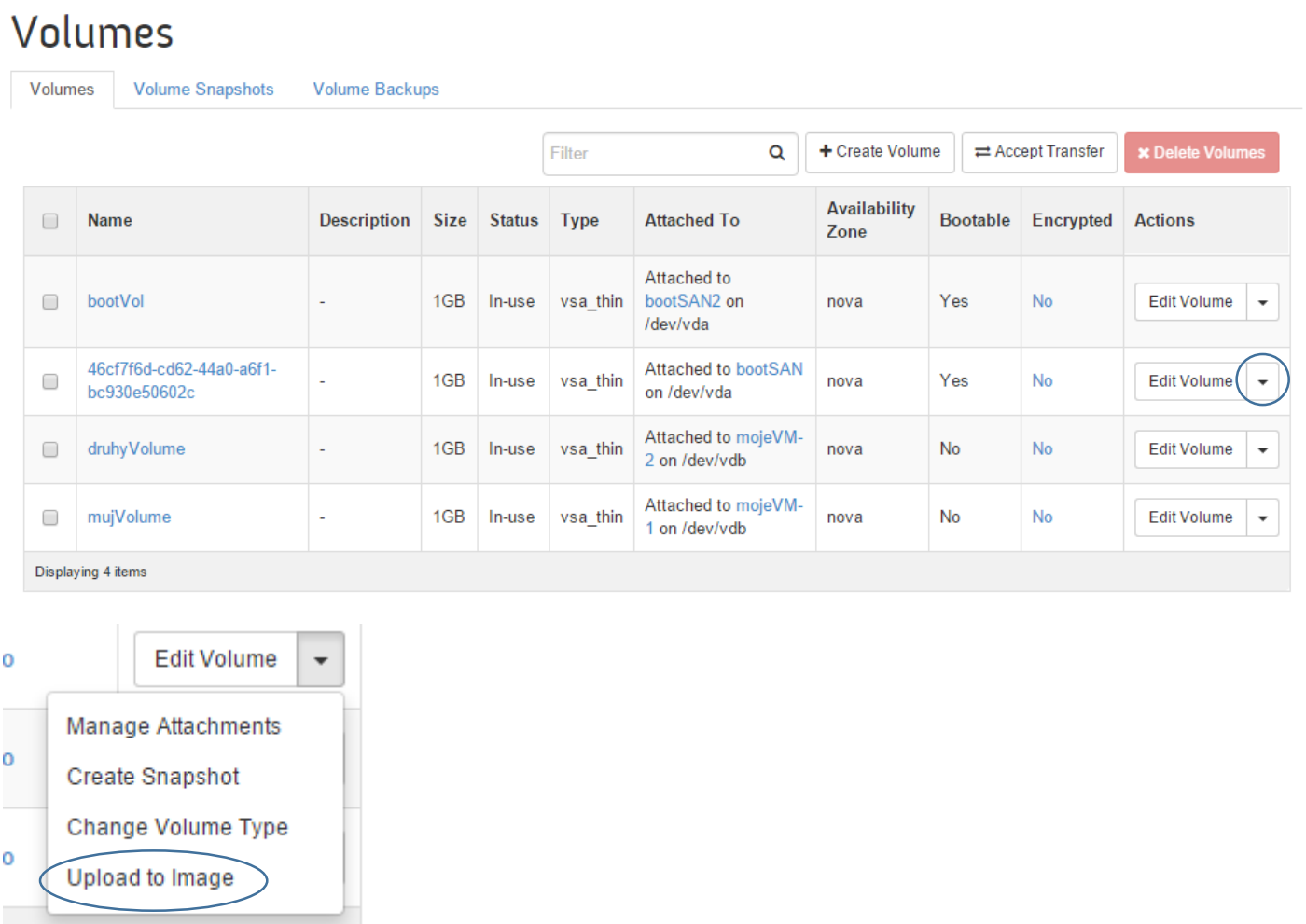


Dokončete průvodce tak jako v předchozích částech. Po nabootování se podívejte do konzole – je tam náš soubor?

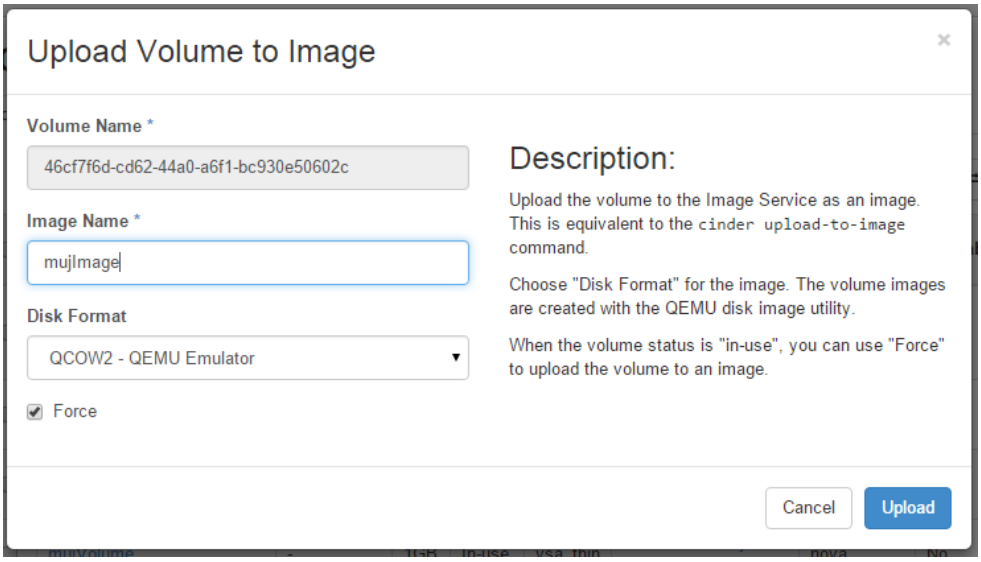


2.3.3. Nový Image z Volume

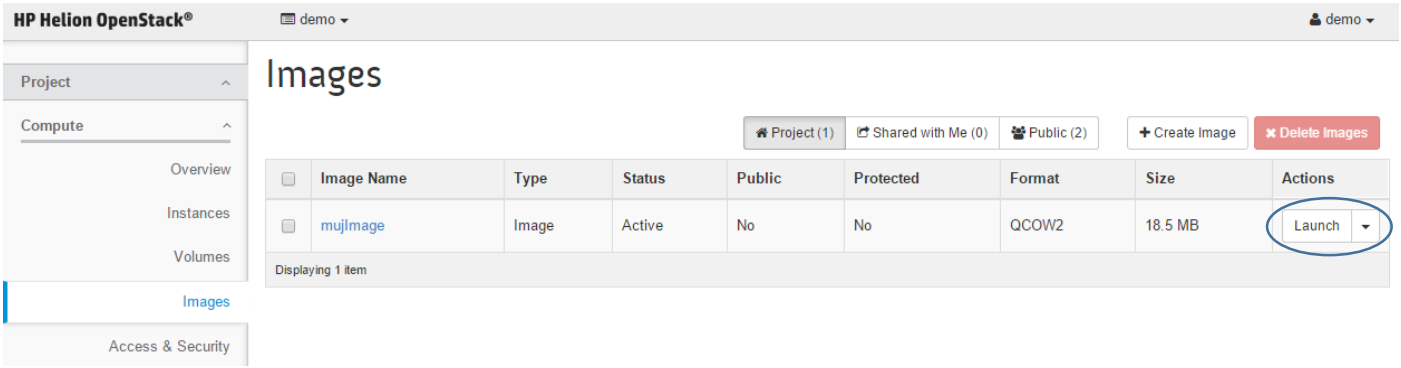
Tak to se nám povedlo. Dokonce tak, že bychom z toho chtěli vytvořit nový image, který můžeme používat dál – bez ohledu na to, jestli bude ve storage nebo lokálně. Způsobů, jak to udělat, je víc – zvolíme možnost vytvoření obrazu z Volume.



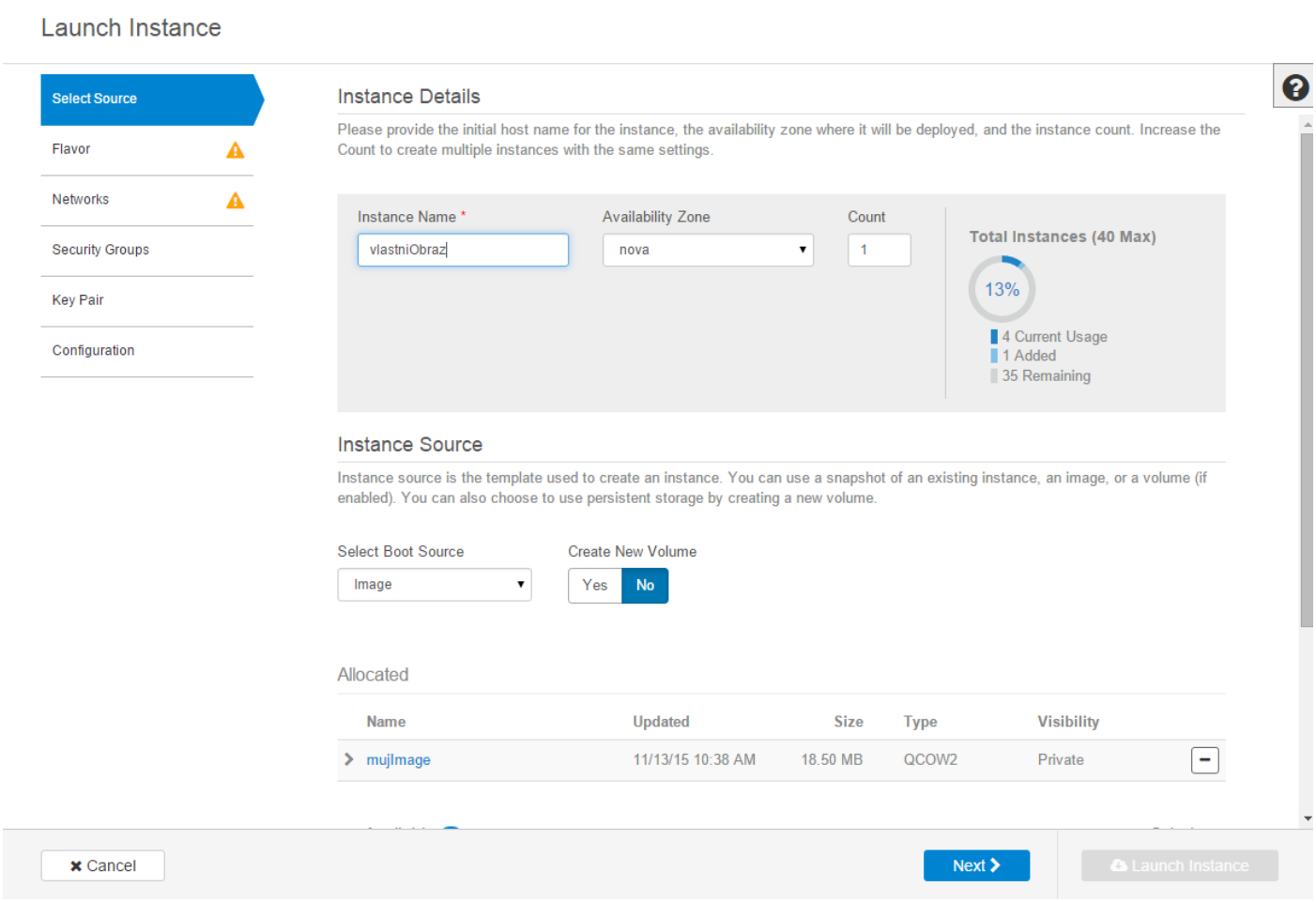
Dejte mu nějaké hezké jméno. Bylo by lepší disk odpojit z VM, ale protože víme, že je stejně vypnutá, uděláme to na sílu (a zaškrtneme Force).



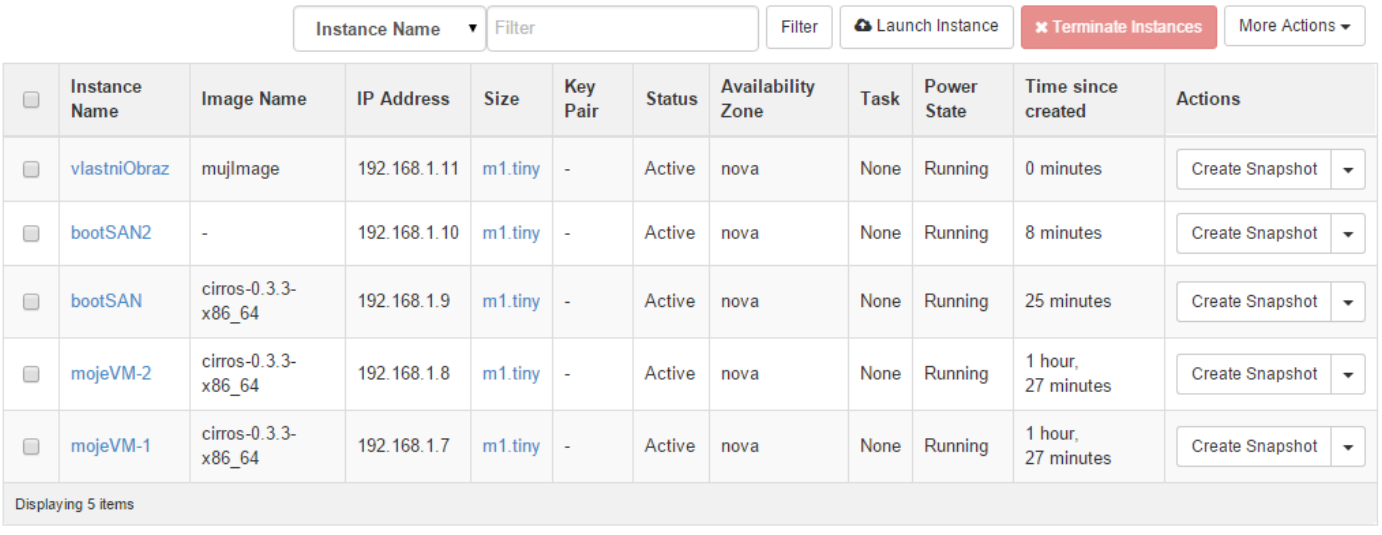
Podívejte se do Compute, Images. Rovnou odtud můžeme spustit průvodce vytvořením instance – klikněte na Launch.



Už víte dobře jak se to dělá – vytvořte instanci z tohoto obrazu a až nastartuje, připojte se na konzoli.

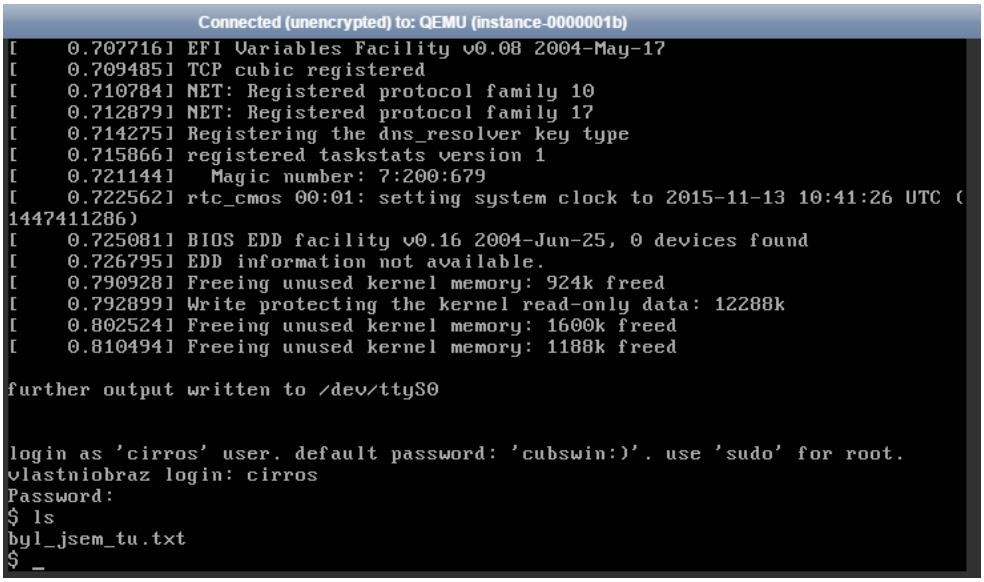


Instances



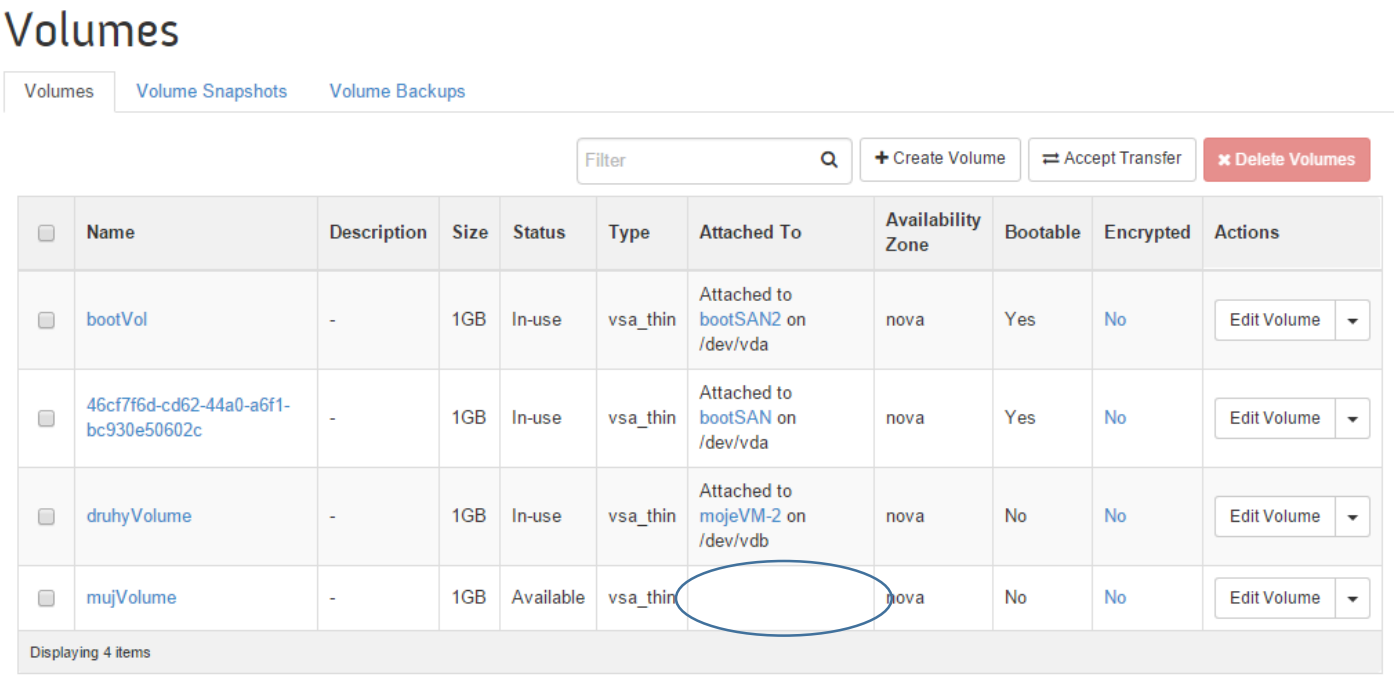
Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
vlastniObraz	mujImage	192.168.1.11	m1.tiny	-	Active	nova	None	Running	0 minutes	Create Snapshot
bootSAN2	-	192.168.1.10	m1.tiny	-	Active	nova	None	Running	8 minutes	Create Snapshot
bootSAN	cirros-0.3.3-x86_64	192.168.1.9	m1.tiny	-	Active	nova	None	Running	25 minutes	Create Snapshot
mojeVM-2	cirros-0.3.3-x86_64	192.168.1.8	m1.tiny	-	Active	nova	None	Running	1 hour, 27 minutes	Create Snapshot
mojeVM-1	cirros-0.3.3-x86_64	192.168.1.7	m1.tiny	-	Active	nova	None	Running	1 hour, 27 minutes	Create Snapshot

Displaying 5 items

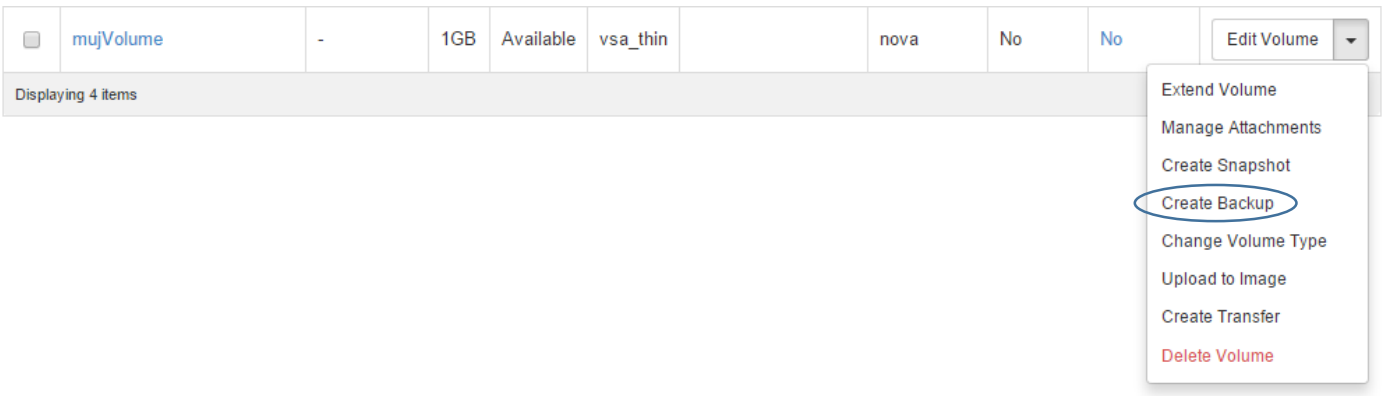


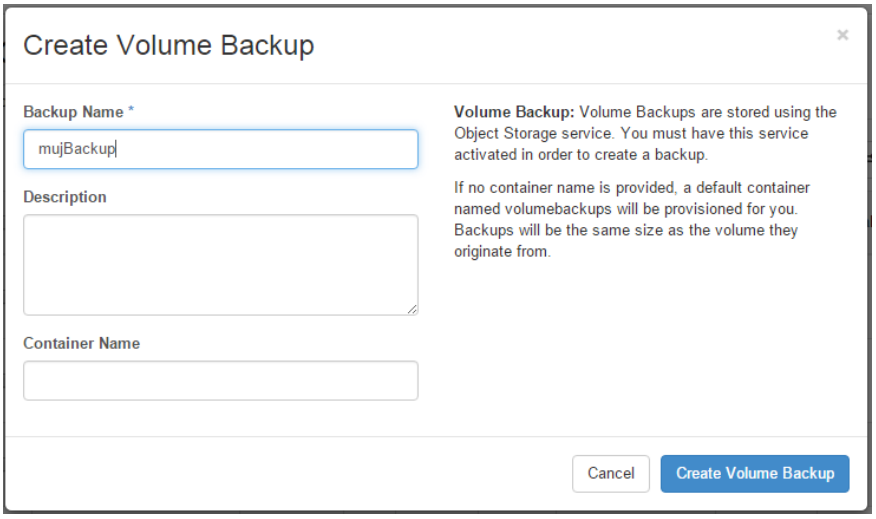
2.3.4. Volume backup

Na závěr ještě odpojte náš úplně první volume s názvem mujVolume – určitě už víte jak.

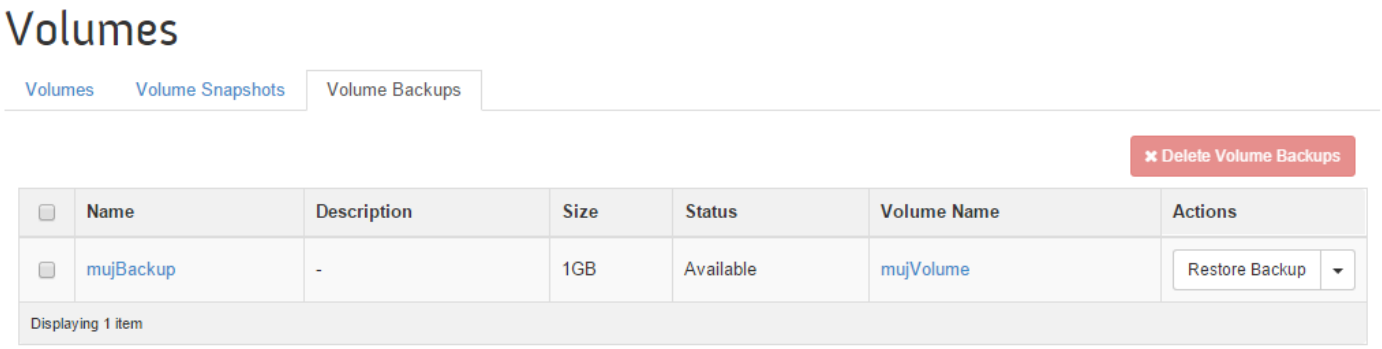


Udějte jeho backup.

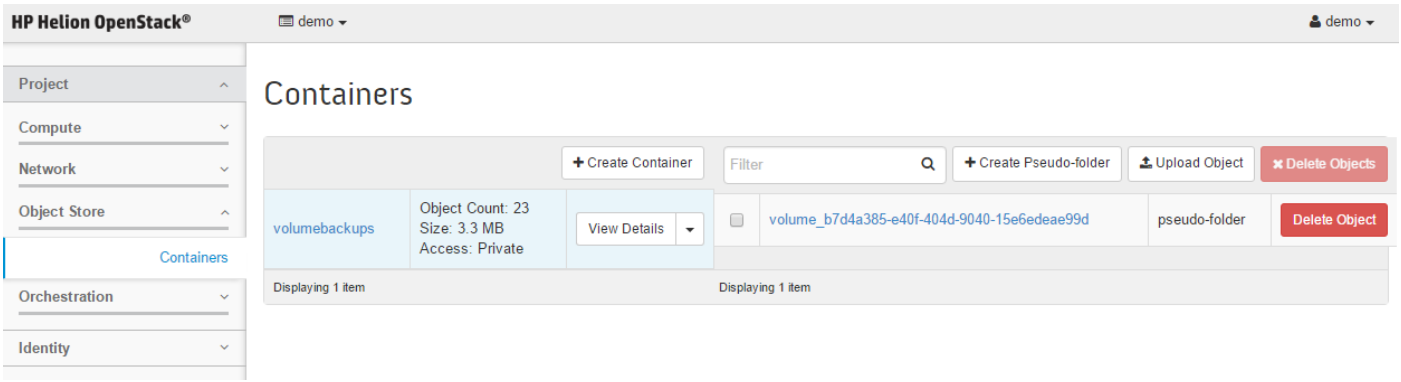




Backup bude po nějaké době hotový.

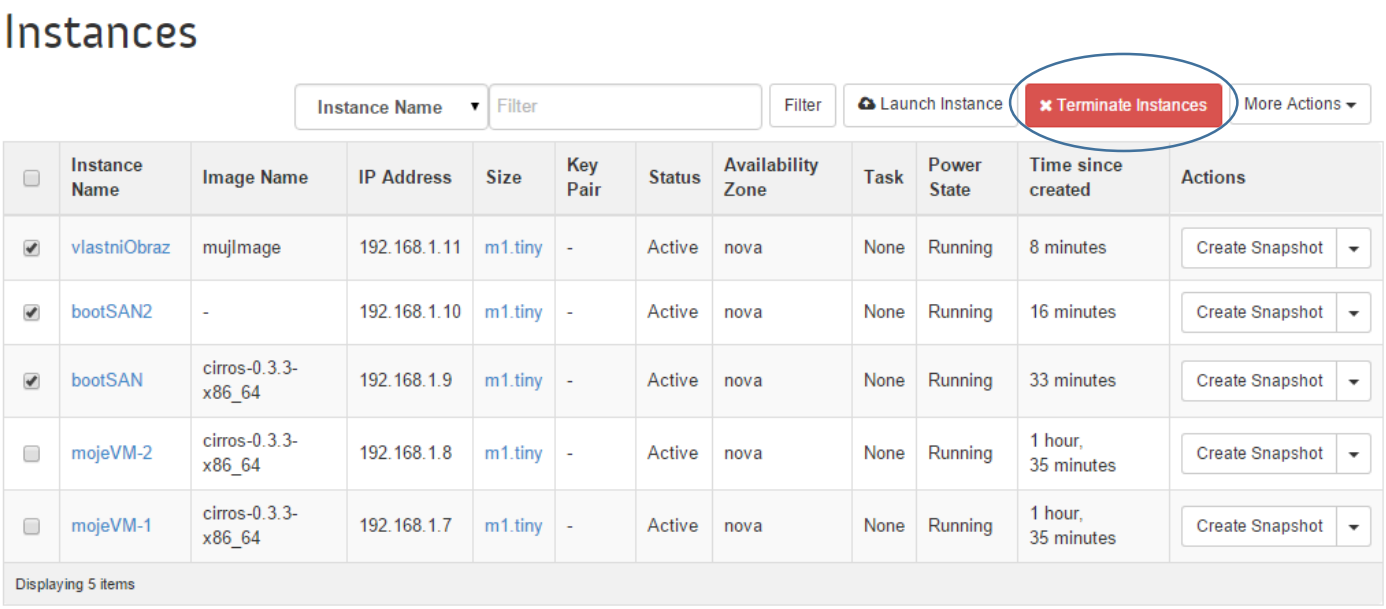


Můžete si ho i stáhnout k sobě – soubor se uložil do objektové storage, která je součástí OpenStack (podrobněji později). Najdete je v Object Store, Containers a v kontejneru volumebackups.



Proč použít Backup místo Snapshot? V případě Snapshot jde o velmi rychlou operaci, ale Volume, z kterého vznikl, není možné smazat (a nechat si jen Snapshot). Možná potřebujete disk skutečně zazálohovat a nechcete, aby příslušné Volume dále existovaly – přesto chcete mít možnost třeba za půl roku ze zálohy Volume znovu vytvořit.

Jsme na konci této části. Pojďme si prostředí trochu vyčistit, ať máme dostatek zdrojů pro další práci. Z instancí ponechejte jen ty první dvě, které jsme vytvořili úplně na začátku. Ostatní zaškrtněte a klikněte na Terminate Instances.



Dále vymažte všechny Volume Backup, Volume Snapshot a nakonec samotné Volume (nejdřív je odpojte).

2.4. Networking, směrování a firewalling

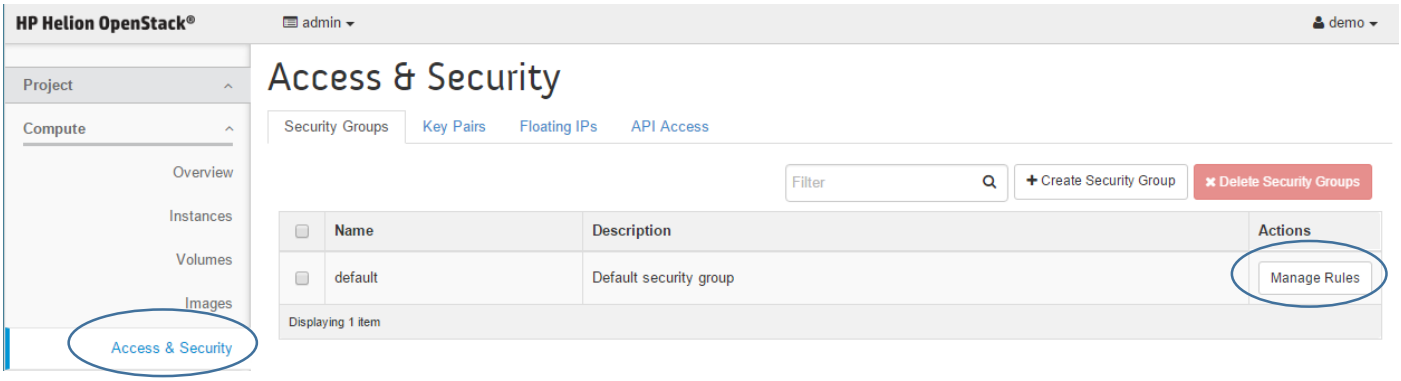
V rámci našeho projektu potřebujeme vytvářet virtuální sítě a máme následující požadavky:

- Nejsme nějak zásadně omezeni na počet sítí, které takto vytvoříme
- Není potřeba žádná dohoda s vlastníky jiných projektů – můžeme mít jakékoli IP rozsahy, neřešíme čísla VLAN ani nic podobného
- Není potřeba žádná interakce s fyzickou sítí, tedy všechny popsané kroky musí být možné realizovat zcela bez zásahu do nastavení fyzické sítě (tedy bez zavolání síťaře)
- Chceme mít možnost směrovat mezi těmto sítěmi, tedy mít router
- Chceme pro svůj projekt dostat seznam IP adres, které mají obecnou platnost ve firmě nebo na Internetu (tedy chceme spojení s reálným světem)
- Můžeme chtít používat firewallová pravidla na úrovni jednotlivých VM, ale i subnetů a routerů
- Můžeme potřebovat řešit VPN připojení poboček do aplikací běžících v OpenStack prostředí
- Možná budeme potřebovat load balancer, který pod jednou virtuální IP bude rozhazovat zátěž na jednotlivé virtuální servery

To vše je součástí HPE Helion OpenStack. Vyzkoušejme si to.

2.4.1. Per-VM stavový firewall

Nejprve si připravme mikrosegmentační pravidla (Security Groups). Jde o stavový firewall, který je implementovaný přímo v compute node a aplikuje se hned, jak provoz s VM pokračuje dál. Podívejme se na výchozí Security Group. Jděte do Compute, Access & Security, kde uvidíte výchozí SG default. Klikněte na manage rules a podívejte, jaké tam jsou.



Výchozí SG povoluje provoz iniciovaný z VM směrem ven pro IPv4 a IPv6 – instance tak například může stahovat soubory z Internetu, protože to je provoz iniciovaný zevnitř. Další pravidla říkají, že do VM může vstupovat jakýkoli provoz z jiných VM, které jsou sučástí stejné Security Group a to zase pro IPv4 i IPv6.

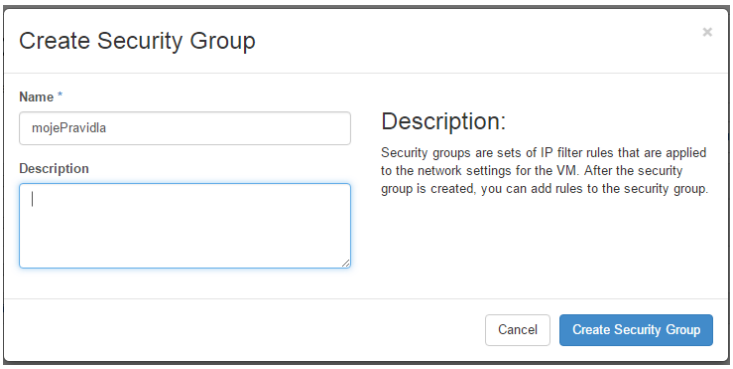
Manage Security Group Rules: default (73162dde-5c71-4e37-a302-964e6fd9876b)

+ Add Rule ✖ Delete Rules

	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
☐	Ingress	IPv6	Any	Any	-	default	Delete Rule
☐	Ingress	IPv4	Any	Any	-	default	Delete Rule
☐	Egress	IPv4	Any	Any	0.0.0.0/0	-	Delete Rule
☐	Egress	IPv6	Any	Any	::/0	-	Delete Rule

Displaying 4 items

My si ale budeme chtít trochu pohrát, takže vytvoříme jinou security group. Jděte zpět a klikněte na Create Security Group a zadejte nějaké jméno.



Klikněte na Manage Rules

	Name	Description	Actions
☐	default	Default security group	Manage Rules
☐	mojePravidla		Manage Rules

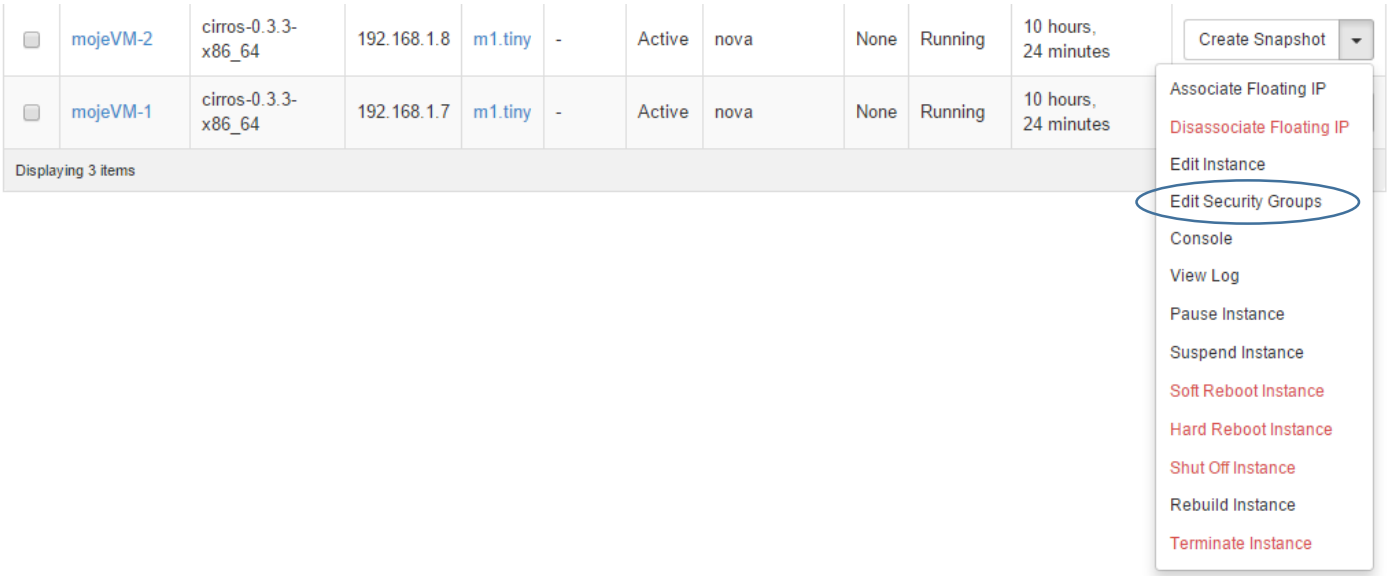
Displaying 2 items

Povolena je komunikace ven, ale směrem dovnitř vůbec nic. Takhle to pro začátek budeme chtít.

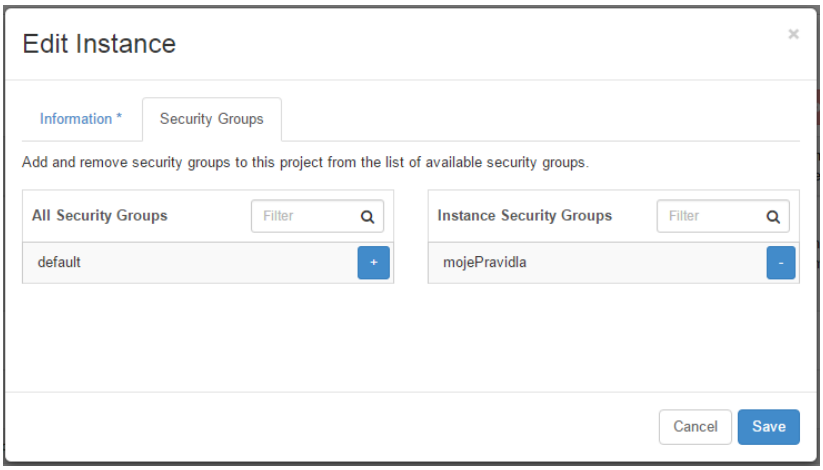
	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
☐	Egress	IPv4	Any	Any	0.0.0.0/0	-	Delete Rule
☐	Egress	IPv6	Any	Any	::/0	-	Delete Rule

Displaying 2 items

U našich běžících VM teď budeme chtít změnit přiřazenou Security Group. Jděte do Compute, Instances a klikněte na Edit Security Groups.

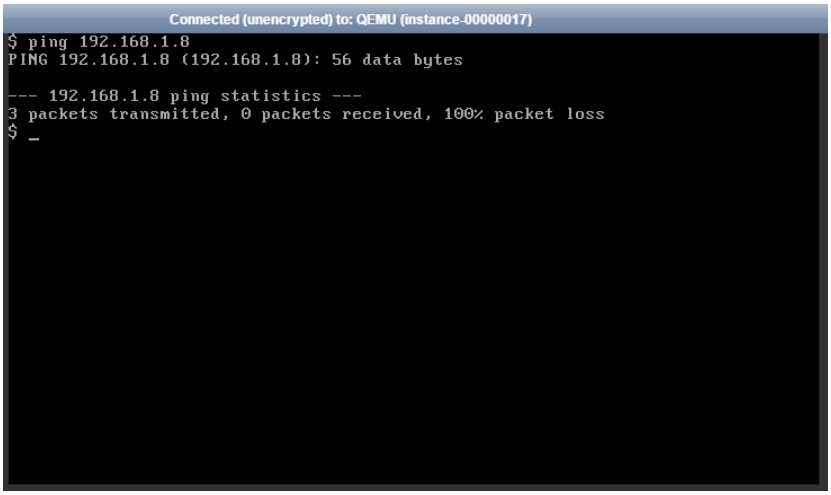


default dejte pryč a naopak přiřaďte mojePravidla.

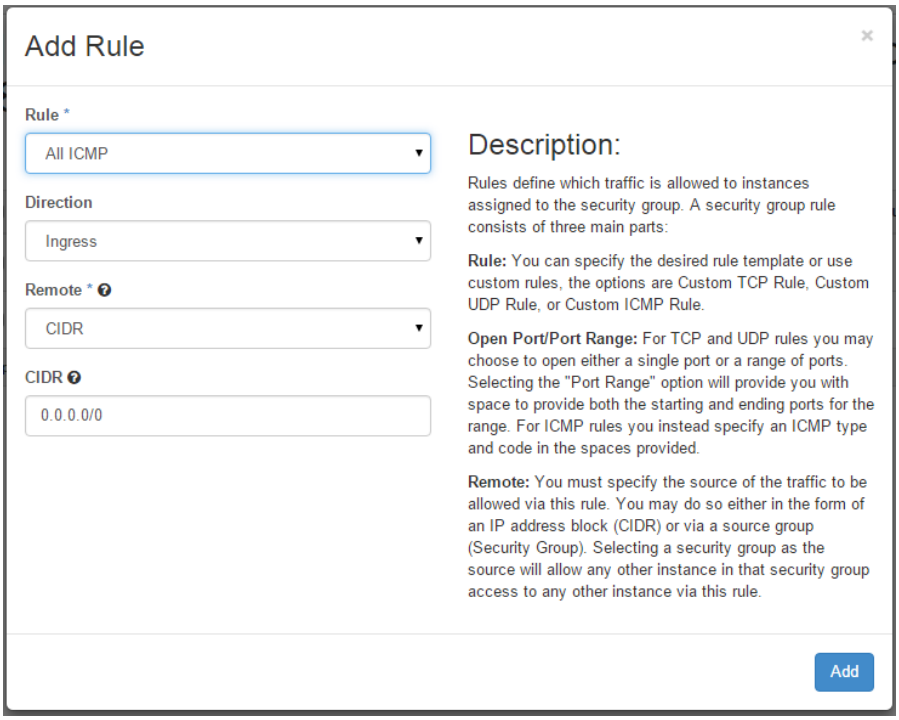


Zopakujte totéž i pro druhou VM.

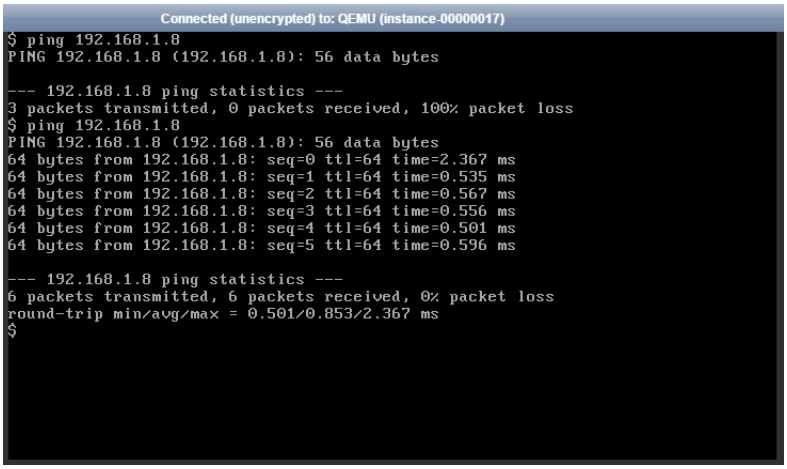
Skočte teď do konzole jedné z VM a zkuste ping na tu druhou. Neprochází.



Pojďme do Manage Rules naší Security Group a přidejme pravidlo povolující ICMP (tedy ping).



Zkuste ping znovu – tentokrát projde.

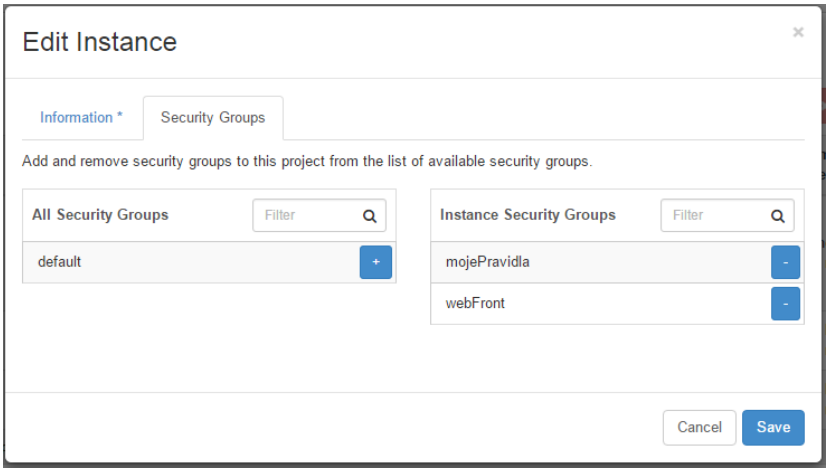


Jedna VM může být součástí více Security Group. Tak například vytvořte Security Group, kterou označíme Web frontend a pravidla budou vypadat nějak takhle.

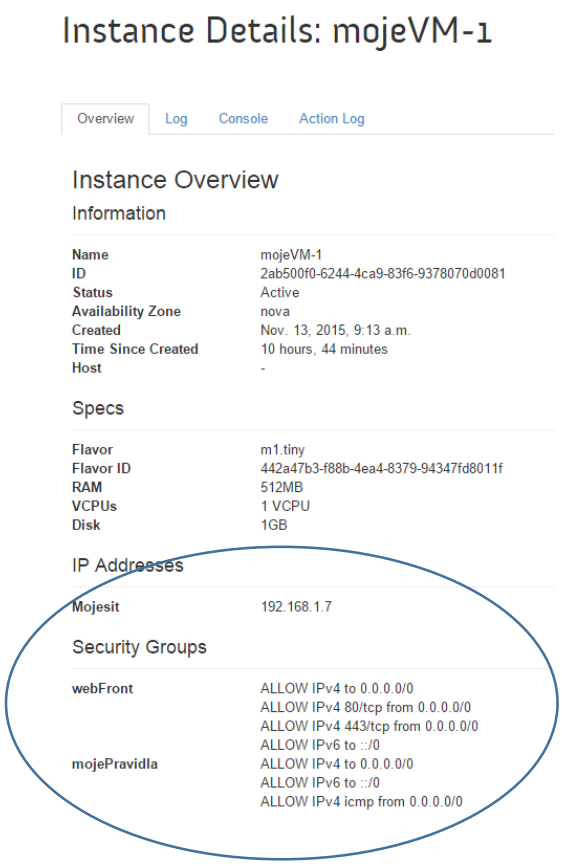
	Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
☐	Egress	IPv4	Any	Any	0.0.0.0/0	-	Delete Rule
☐	Egress	IPv6	Any	Any	::/0	-	Delete Rule
☐	Ingress	IPv4	TCP	80	0.0.0.0/0	-	Delete Rule
☐	Ingress	IPv4	TCP	443	0.0.0.0/0	-	Delete Rule

Displaying 4 items

Přiřaďte k jedné z VM ještě tuto další Security Group (Compute, Instances, Edit Security Groups).



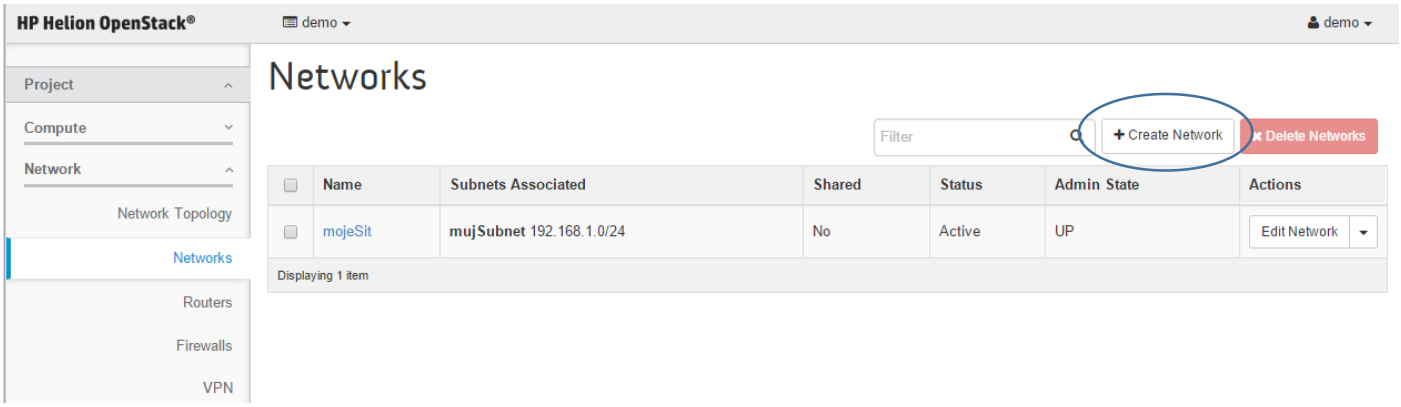
Na rozdíl od ACL je Security Group řešena jako výčet toho, co je povoleno a neobsahuje explicitní zákazy. Díky tomu je možné bez problémů spojit více Security Group s jedou VM. Když si rozkliknete detaily vaší instance kliknutím na její název uvidíte, že pravidla jsou tam všechna.



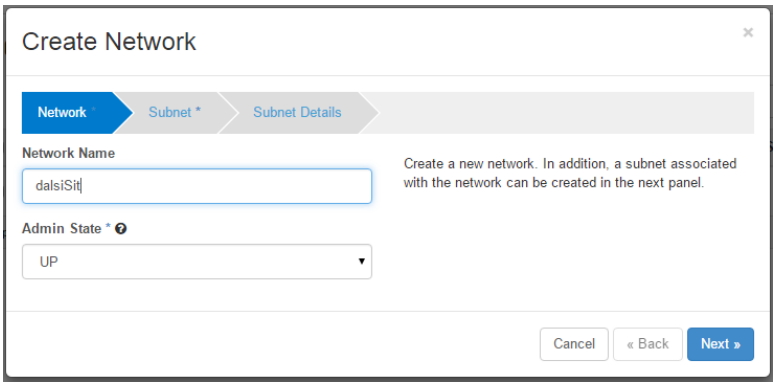
2.4.2. Sítě a směrování

Jděte do záložky Network, Networks, kde najdete síť, kterou jsme společně vytvořili na začátku labu. Dejme tomu, že v této síti provozujeme webové servery a teď chceme přidat servery aplikační a vytvořit pro ně další síť.

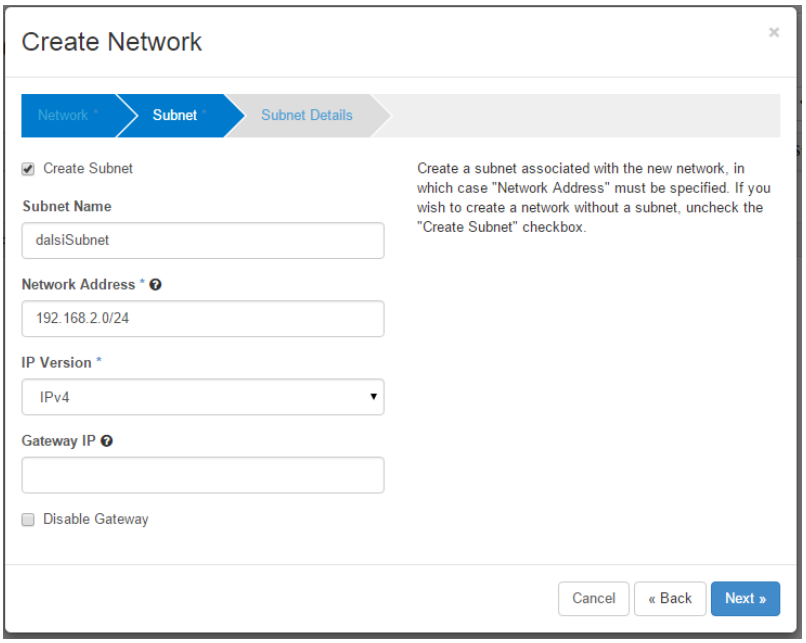
Pro založení další sítě klikněte na Create Network.



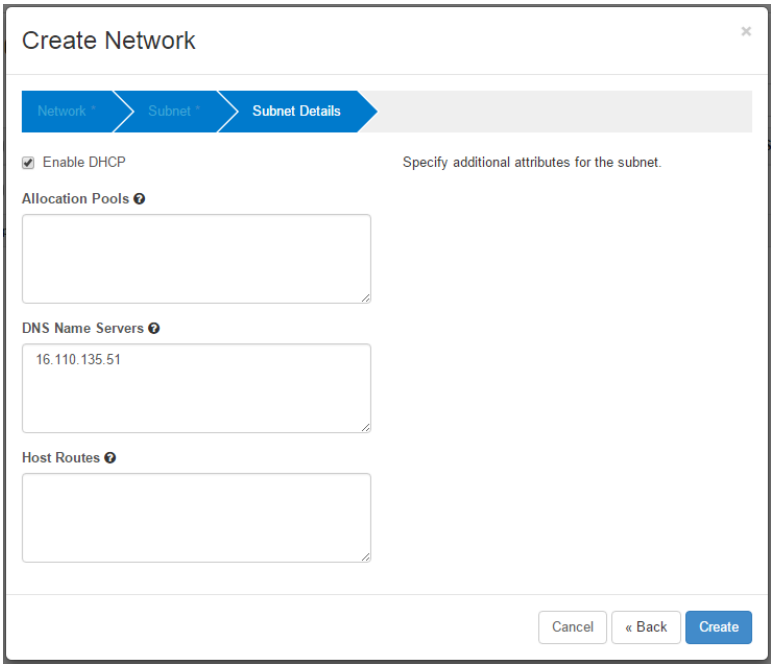
Pojmenujte ji a klikněte na Next.



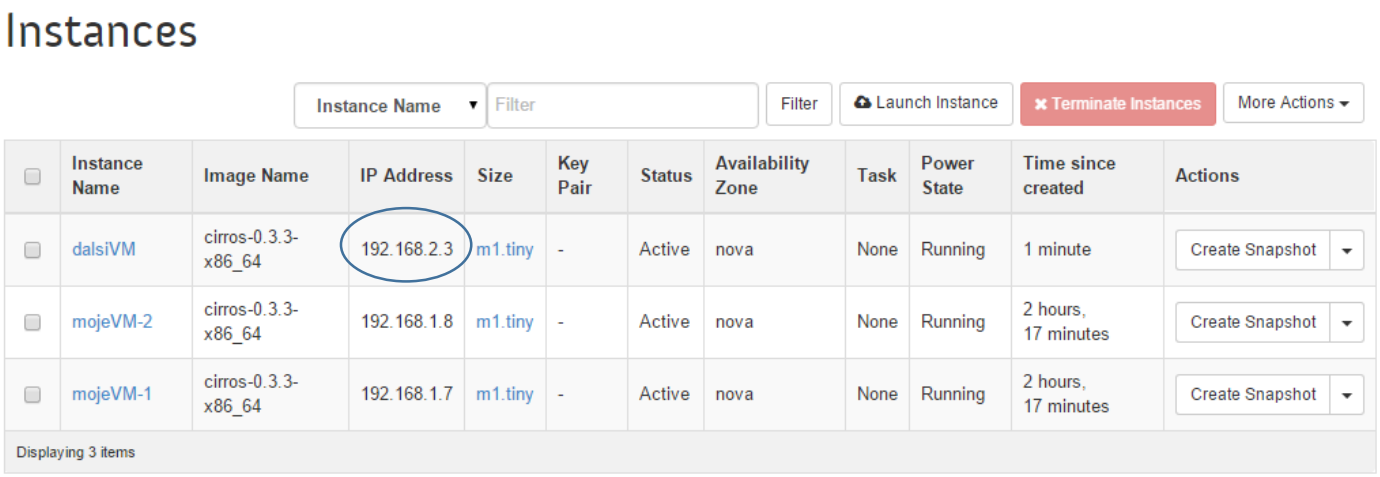
Vytvořte subnet – jiný, než ve vaší první síti (nicméně nezávisle na ostatních tenantech/projektech, kteří mohou klidně používat adresy úplně stejné). Klikněte na Next.



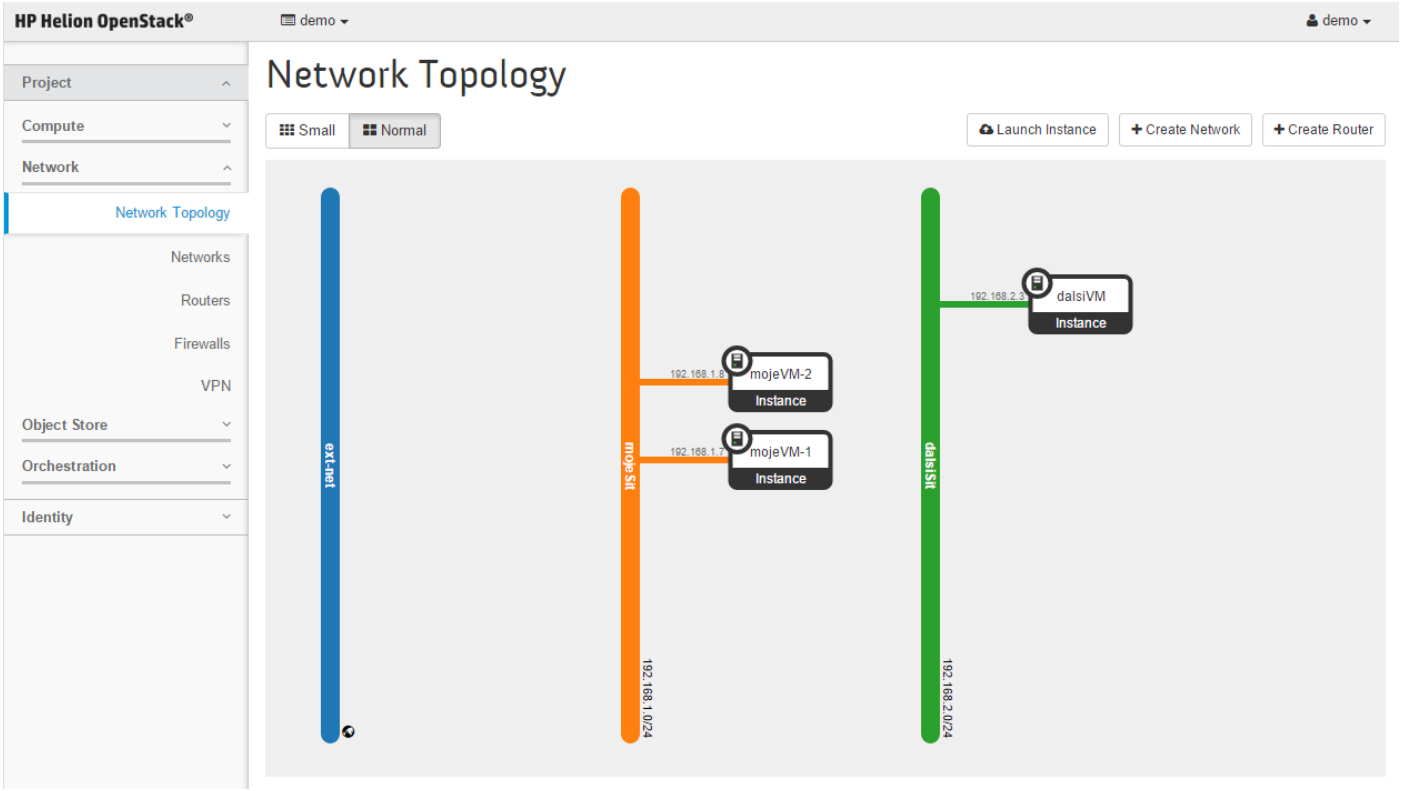
Vyplňte DNS a klikněte na Create.



Vytvořte jednu další instanci s Cirros image a přiřaďte ji do této nové sítě a jako Security Group použijte mojePravidla. Už znáte vše potřebné, takže si ukažme jen výsledek.

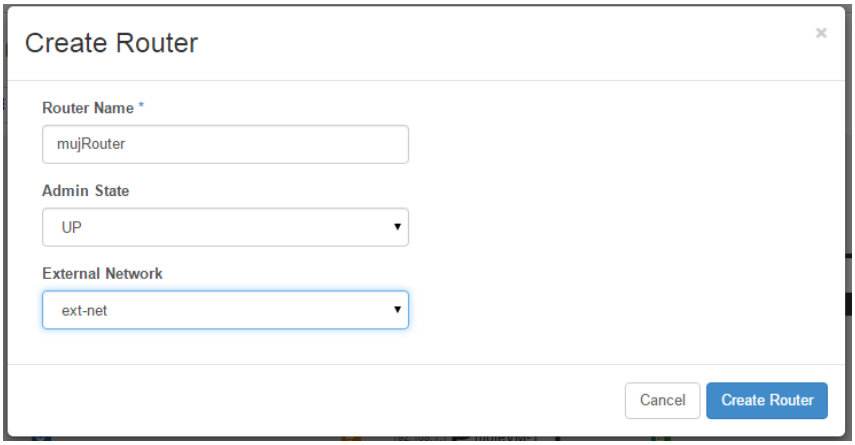


Podívejme se na obrázek topologie – Network, Network Topology, Normal

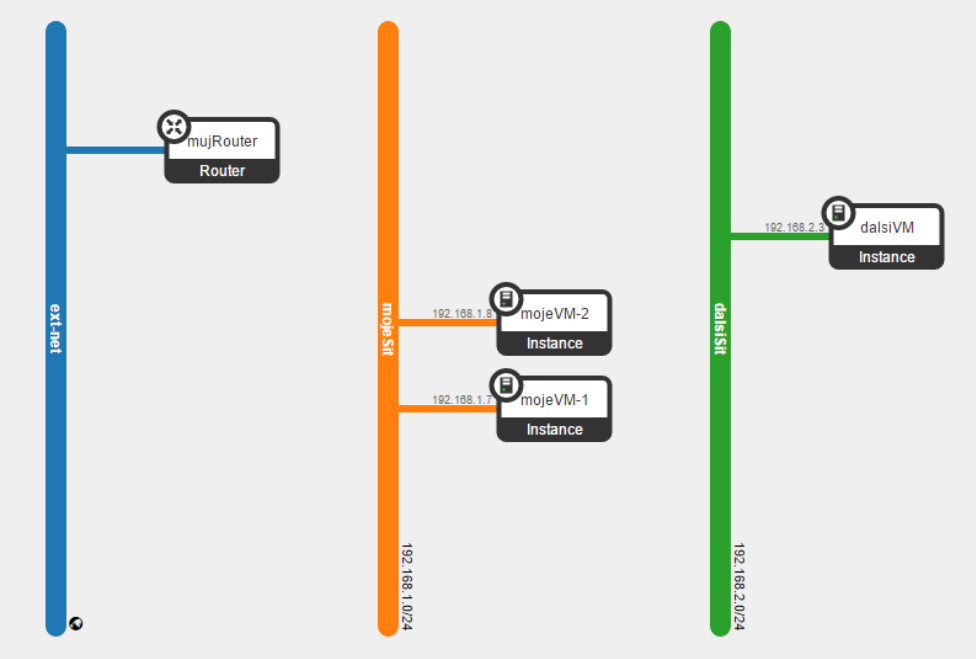


Máme co jsme chtěli – naše dvě sítě jsou zcela oddělené a nemají přístup kamkoli ven. V našem případě to ale není co potřebujeme. Budeme chtít obě sítě propojit routerem a také je obě připojit do venovní sítě, aby bylo možné stahovat třeba nějaké balíčky z Internetu apod. (tedy chceme, aby viděli ven a router prováděl SNAT, tedy překlad adres tak, jak to třeba dělá váš domácí Internetový router).

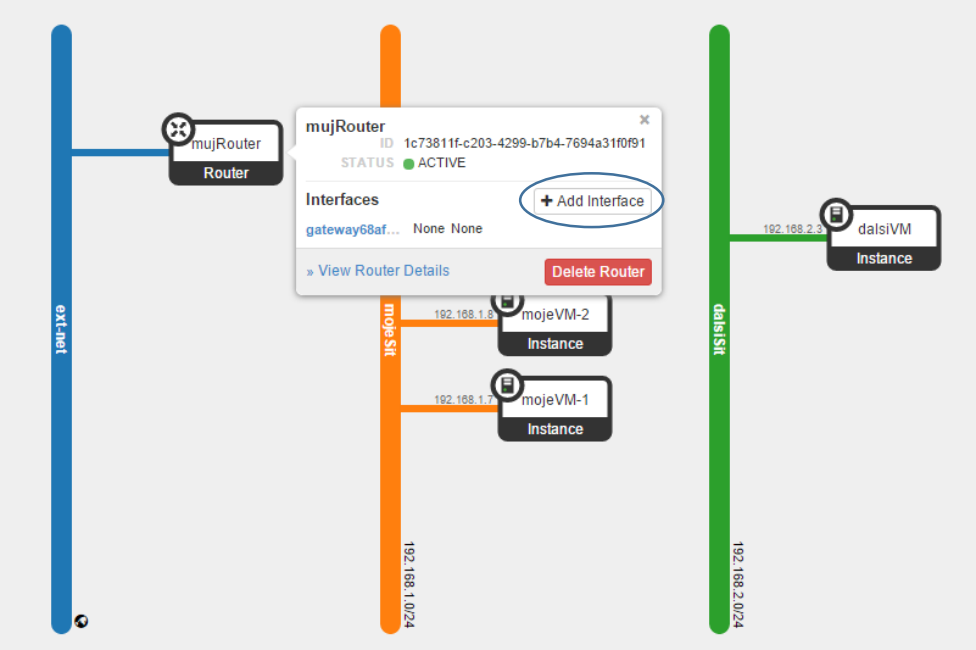
Buď jděte do záložky Routers a nebo přímo z topologie klikněte na Create Router. Dejte mu nějaké jméno a také vyberte externí síť. Tím zařídíme, že router bude NATovat provoz směrem ven do této sítě, takže se VM třeba dostanou na Internet. Potvrďte kliknutím na Create Router.



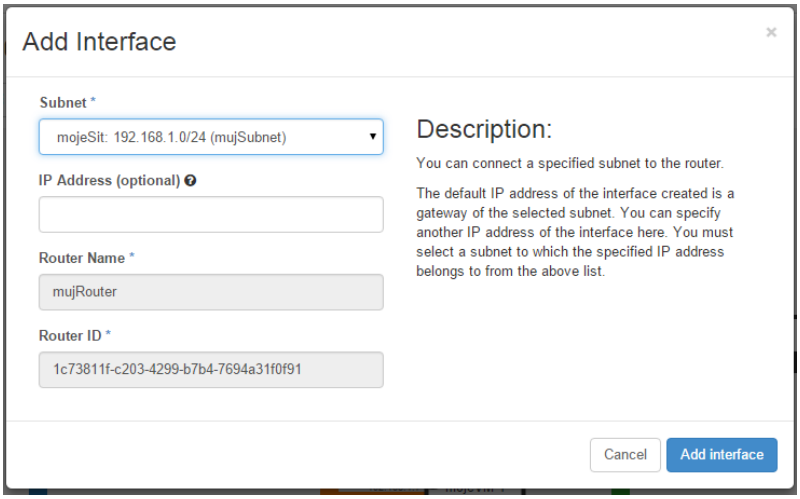
Jak se obrázek změnil? Máme nový virtuální router a ten je připojen na venkovní síť.



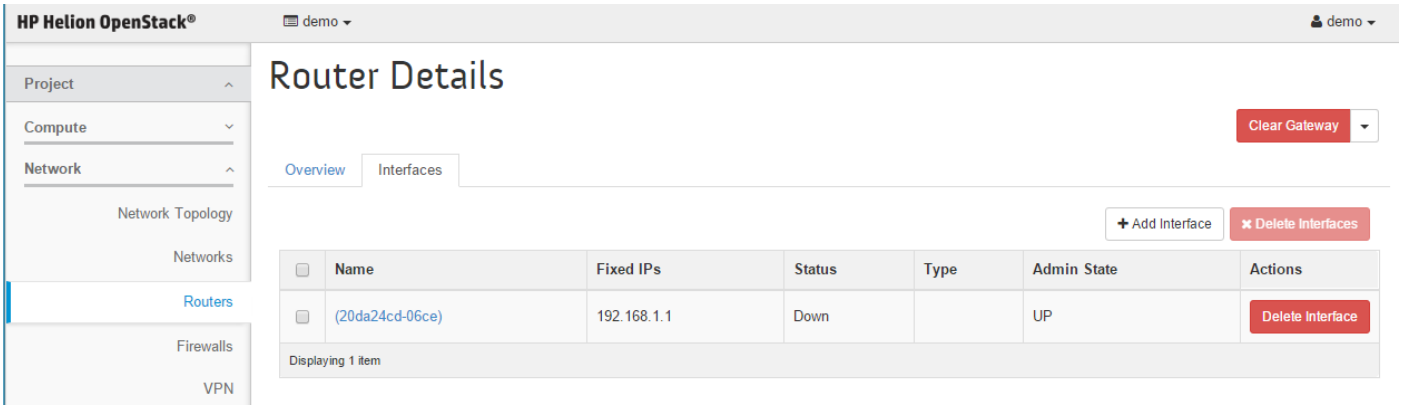
Připojme do něj naše vnitřní sítě. Potřebujeme vytvořit na routeru porty (interface) a do něj je připojit. Můžete buď jít do záložky Routers a kliknout na něj a udělat to tam. Nebo v topologii klikněte myší na router a pak na Add Interface.



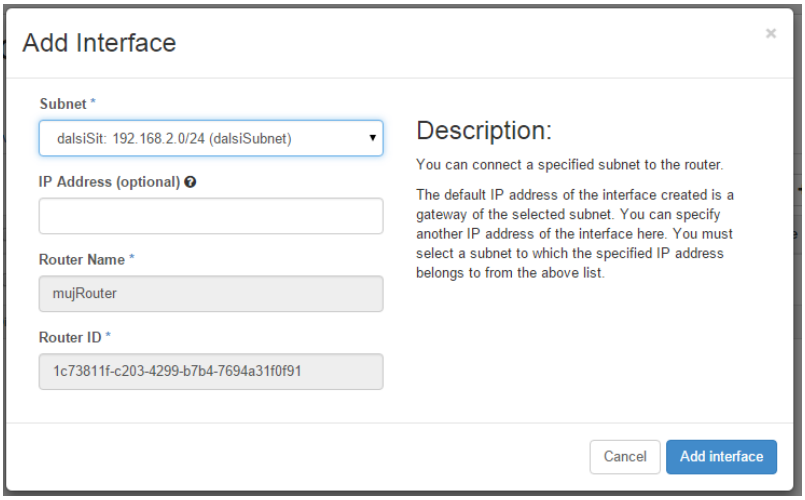
Připojte první síť.



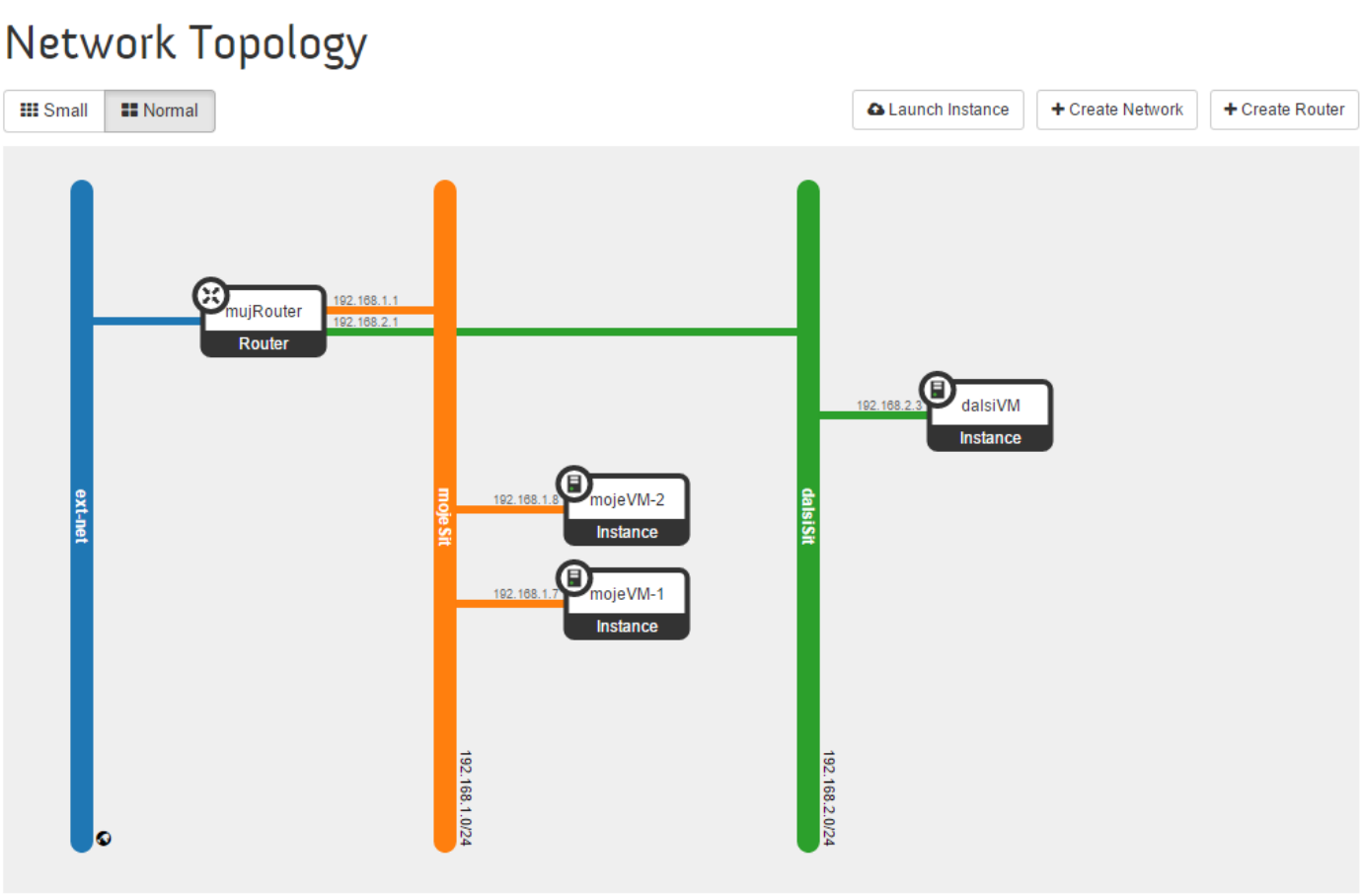
Druhou pro změnu přidejte třeba ze záložky Interfaces na Routeru kliknutím na tlačítko Add Interface.



Přidejte druhou síť.



Podívejme se teď na výslednou topologii.

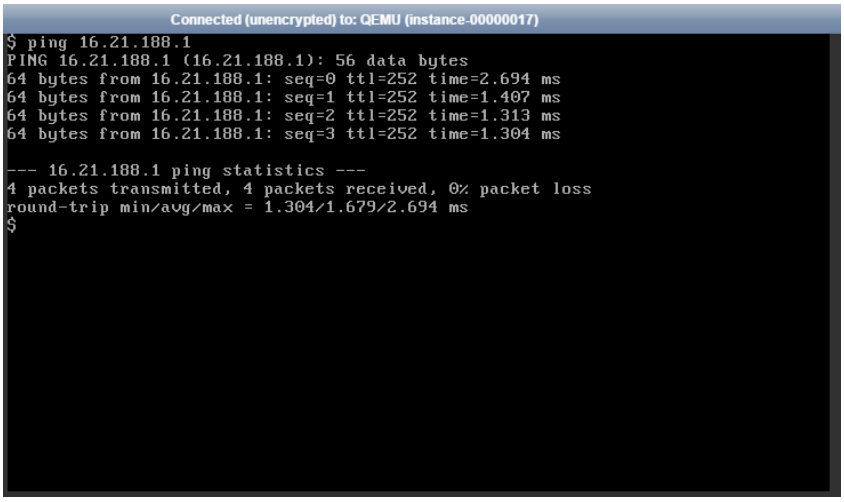


Teď už to stačí jen vyzkoušet. Skočte do konzole jedné z prvotních VM a pingněte do naší nové sítě. Zjistíte, že směrování skutečně funguje.

Tato komunikace je řešena distribuovaným virtuálním routerem. Nejedná se tedy o centralizovaný router, který by byl úzkým hrdlem. Funkce směrování je distribuovaná přes všechny compute nody, kde je nějaká VM tohoto tenantu.

Vyzkoušíme take přístup do skutečné externí sítě. V našem případě půjde o SNAT, tedy dojde k překladu na jednu adresu z externího subnetu (například Internetu nebo firemní sítě). Tento provoz je v aktuální verzi OpenStack řešen centrálně na řídících nodech.

Jděte do konzole a zkuste ping na adresu fyzického routeru, který je mimo váš OpenStack cloud (např. 16.21.188.1).

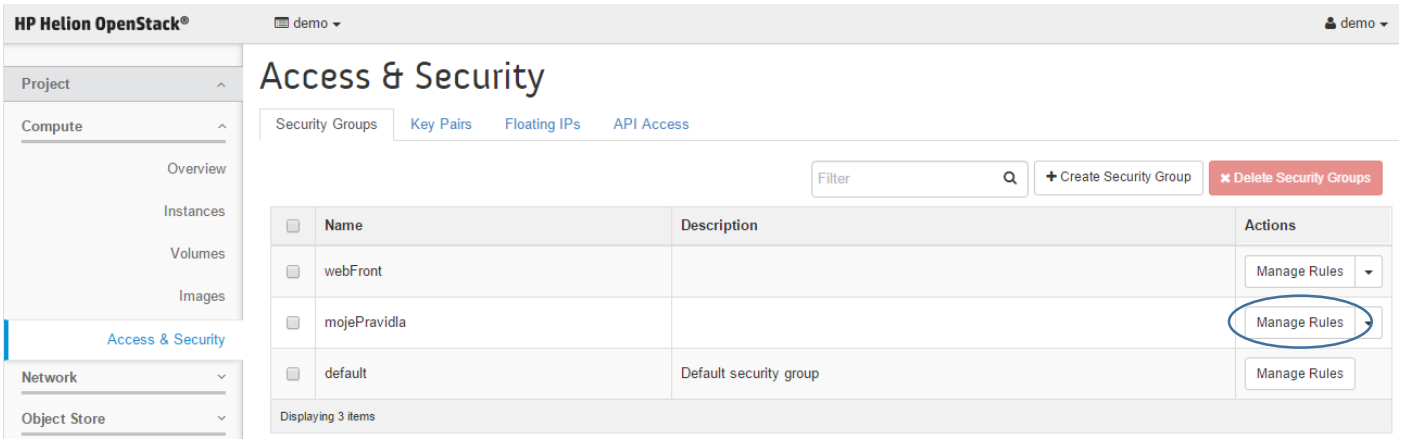


2.4.3. Přístup ke službám v cloudu z venku (Floating IP)

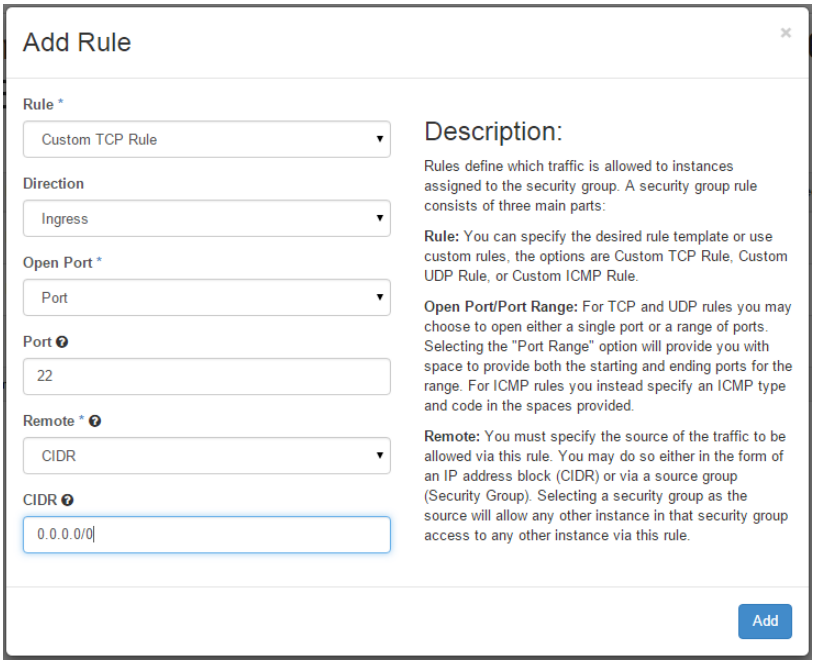
V předchozí části labu jsme našemu routeru přiřadili jednu z externích sítí, kterými jsou existující sítě v infrastruktuře, DMZ nebo Internet apod. Vyzkoušeli jsme si, že je možné se z našich VM dostat ven díky SNAT překladu adres na routeru). To nám může stačit pro stahování balíčků, ale pro front end servery to bude málo. Pokud je to možné, budeme chtít držet backend komunikaci uvnitř tenantu (databáze, aplikační server apod.), nicméně například k web serveru potřebujeme dát přímo přístup našim uživatelům (externím sítím).

Helion OpenStack toto řeší překladem 1:1, tedy Floating IP. V rámci tenantu/projektu si zažádáte o přidělení reálné IP adresy z vybrané externí sítě a tuto svážete s nějakou z vašich VM. Jinak řečeno vaše instance vnitřně nemá tuto IP adresu nastavenou na své síťové kartě. Tam zůstává interní adresa virtuální sítě, ale při komunikaci ven dojde k jejímu přeložení na externí (a při cestě od uživatelů naopak). Toto se odehrává distribuovaným způsobem (Distributed Virtual Router), takže přímo na compute node, kde je VM hostovaná (provoz nemusí jít do centrálního routeru, přímo z compute node je přeadresován a odchází do externí sítě – podle toho jak to administrátor cloudu udělal třeba nějakou specifickou VLAN nebo dedikovanou síťovou kartou). Protože adresa není nastavena přímo uvnitř VM je označována za plovoucí (Floating) – můžete ji přidělit jedné z VM, ale pak změnit názor a této ji odejmou a dát jiné (můžete tak například bokem instalovat novou verzi aplikace, zkoušet si ji a až budete spokojeni, rychle a snadno přehodit IP adresy).

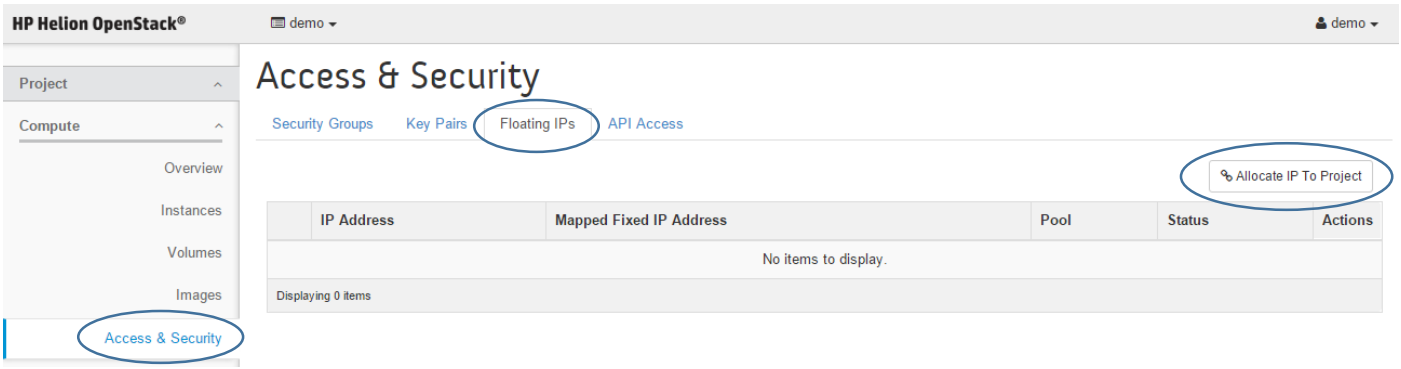
Abychom mohli efektivně zkoušet, povolme si přístup do našich VM přes SSH. Jděte do Compute, Access & Security a klikněte na správu pravidel v naší SG mojePravidla.



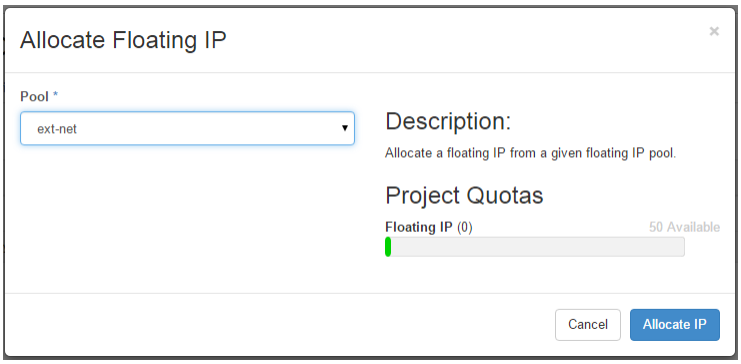
Přidejte pravidlo povolující SSH protokol, tedy TCP port s číslem 22.



Po přidání jděte zpět a podívejte se na záložku Floating IPs. Tady si můžeme pro náš projekt vzít nějaké reálné IP adresy z vnějšího světa, jejichž maximální počet je dán kvótou, kterou pro náš tenant/projekt určil administrátor. Klikněte na Allocate IP to Project.

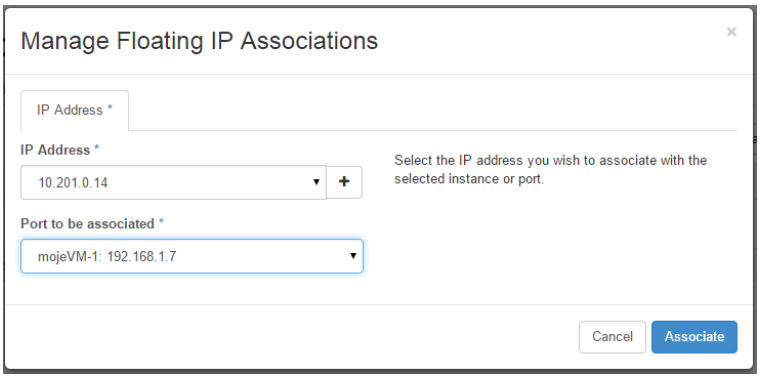


Vyberte si externí subnet a klikněte na Allocate IP

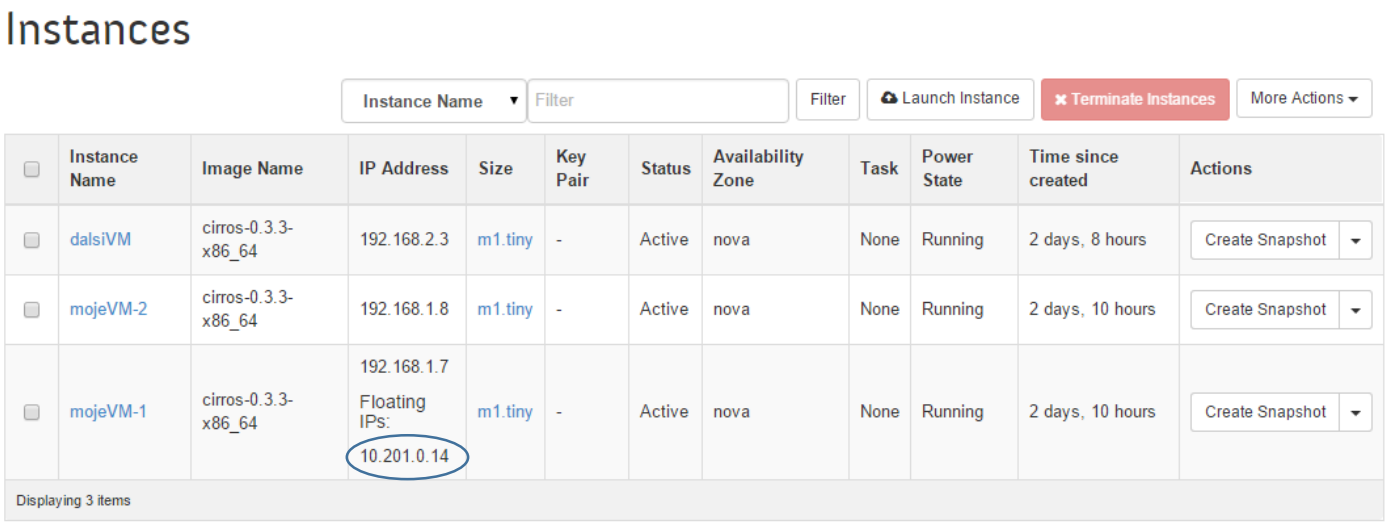


Výborně, adresu jsme dostali. Můžeme ji vrátit zpět nebo přiřadit nějaké VM, což lze rovnou udělat. Klikněte na Associate.

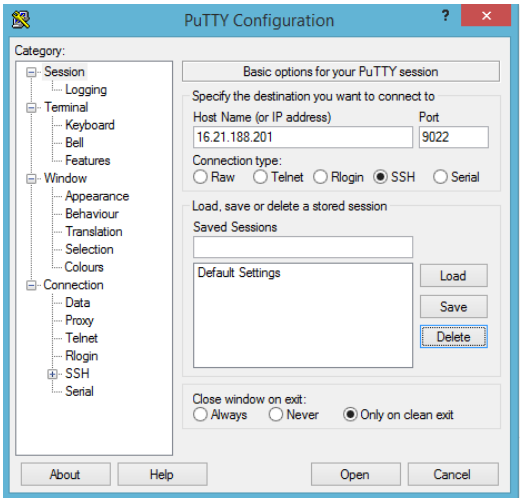
Přiřaďte externí plovoucí IP k jedné z našich VM a klikněte na Associate.



Jděte do Compute, Instances a uvidíte informaci o přiřazené plovoucí adrese.



Vyzkoušejme si to. Protože tato venkovní síť je zamčená uvnitř našeho lab prostředí, nejprve se připojte na labServer. Jeho IP adresu a port najdete na začátku tohoto dokumentu v části 2.1. Použijte Putty nebo jiného svého oblíbeného SSH klienta.

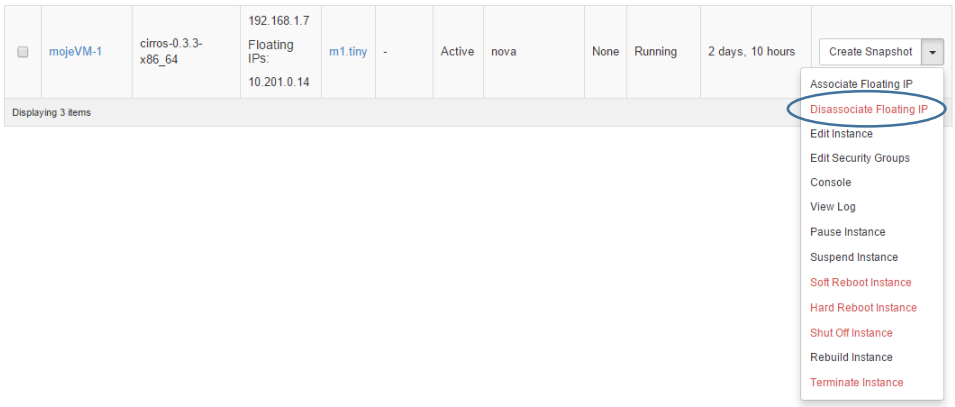


Přihlašovací jméno a heslo je stejné jako do konzole Helion OpenStack.

Jakmile budete uvnitř, zkouste pingnout vaší plovoucí IP. Pokud se to podaří, zkuste SSH s uživatelem cirros - heslo je, jak už víte, „cubswin:)“. Pravděpodobně budete dotázáni na uložení veřejného klíče. Až budete uvnitř, napište hostname, aby byla jistota, že jste ve VM, kterou očekáváte. Pak se odpojte.

```
tomas@labserver:~$ ping 10.201.0.14
PING 10.201.0.14 (10.201.0.14) 56(84) bytes of data.
64 bytes from 10.201.0.14: icmp_seq=1 ttl=61 time=1.09 ms
64 bytes from 10.201.0.14: icmp_seq=2 ttl=61 time=0.646 ms
^C
--- 10.201.0.14 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.646/0.870/1.095/0.226 ms

tomas@labserver:~$ ssh cirros@10.201.0.14
The authenticity of host '10.201.0.14 (10.201.0.14)' can't be established.
RSA key fingerprint is b0:44:1f:85:7e:ad:57:e0:23:54:c0:a6:73:80:89:fb.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.201.0.14' (RSA) to the list of known hosts.
cirros@10.201.0.14's password:
$ hostname
mojevm-1
$ exit
Connection to 10.201.0.14 closed.
tomas@labserver:~$
```

Můžeme teď plovoucí IP server odebrat a dát ji jinému. Jděte do Compute, Instances a klikněte na Disassociate IP.



Uvolněnou IP přiřaďte někomu jinému a zkuste se opět připojit. Protože jde o jinou VM, public key se změnil, takže pro SSH jej budete muset nejprve odstranit. Samozřejmě v případě služby jako je web server nic takového řešit nemusíte.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
dalsiVM	cirros-0.3.3-x86_64	192.168.2.3	m1.tiny	-	Active	nova	None	Running	2 days, 8 hours	Create Snapshot
mojeVM-2	cirros-0.3.3-x86_64	192.168.1.8 Floating IPs: 10.201.0.14	m1.tiny	-	Active	nova	None	Running	2 days, 10 hours	Create Snapshot
mojeVM-1	cirros-0.3.3-x86_64	192.168.1.7	m1.tiny	-	Active	nova	None	Running	2 days, 10 hours	Create Snapshot

Displaying 3 items

```
tomas@labserver:~$ ssh cirros@10.201.0.14
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
cd:ec:f1:13:c6:d6:e3:06:8e:79:29:56:fc:80:64:f5.
Please contact your system administrator.
Add correct host key in /home/tomas/.ssh/known_hosts to get rid of this message.
Offending RSA key in /home/tomas/.ssh/known_hosts:1
  remove with: ssh-keygen -f "/home/tomas/.ssh/known_hosts" -R 10.201.0.14
RSA host key for 10.201.0.14 has changed and you have requested strict checking.
Host key verification failed.

tomas@labserver:~$ ssh-keygen -f "/home/tomas/.ssh/known_hosts" -R 10.201.0.14
# Host 10.201.0.14 found: line 1 type RSA
/home/tomas/.ssh/known_hosts updated.
Original contents retained as /home/tomas/.ssh/known_hosts.old
tomas@labserver:~$ ssh cirros@10.201.0.14
The authenticity of host '10.201.0.14 (10.201.0.14)' can't be established.
RSA key fingerprint is cd:ec:f1:13:c6:d6:e3:06:8e:79:29:56:fc:80:64:f5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.201.0.14' (RSA) to the list of known hosts.
cirros@10.201.0.14's password:
$ hostname
mojevm-2
$ exit
Connection to 10.201.0.14 closed.
tomas@labserver:~$
```

2.4.4. Provider sítě

Pro naprostou většinu situací doporučuji používat virtuální networking tak, jak jsme to dělali v tomto labu. Tedy vytvořit virtuální sítě, virtuální routery a ven se dostávat přes SNAT pro back-end a Floating IP pro servery, ke kterým mají přistupovat uživatelé. Směrování ven (včetně překladu na Floating IP) provádíte na externí sítě, kterých můžete míc víc (například jedna externí síť může být Intranet, druhá DMZ). Tímto způsobem získáte maximální flexibilitu a výhody IaaS.

Mohou ale být situace, zejména při pozvolné migraci z pouhé virtualizace k IaaS, kdy chcete VM rovnou přímo napojit do externí sítě, tedy do nějaké reálné VLAN ve vašem datovém centru. Podobně jako u běžné virtualizace tedy řeknete – provoz této VM v okamžiku, kdy opouští compute node, má mít VLAN tag 1001 (a předpokládáte, že váš síťař takovou VLAN skutečně má). Bez multitenance, bez virtualizace, bez flexibility volby IP rozsahů apod.

Tato možnost není přístupná přímo z GUI, resp. nejprve administrátor musí provést deployment této VLAN na compute nody (na to má Helion Lifecycle Manager) a zavést tuto Provider VLAN v administrátorské části OpenStack. Pak mohou uživatelé (nebo konkrétní projekt) tuto přímou VLAN využívat.

V našem labu to zkoušet nebudeme a v praxi doporučuji používat spíše vyjímečně. Nicméně je to relevantní možnost a z pohledu uživatele jednoduchá a snadno pochopitelná.

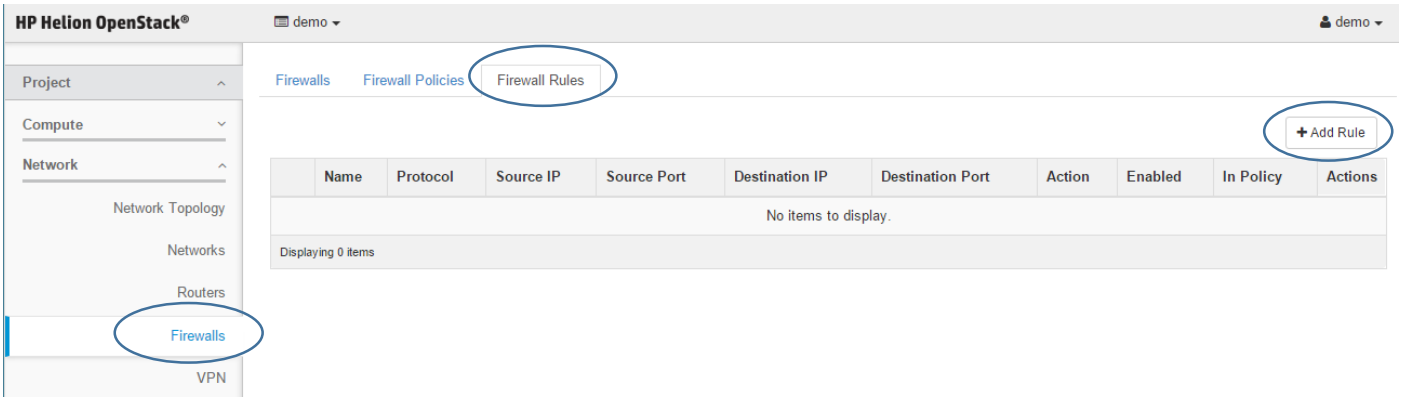
2.4.5. Firewall as a Service

V předchozí části jsme si ukazovali mikrosegmentaci formou Security Group. Šlo o malé stavové firewally implementované přímo na urovni hypervisoru (resp. vSwitche) a aplikovali se per-VM. To je možné doplnit ještě službou FWaaS, tedy firewallem aplikovaným na celý tenant/projekt.

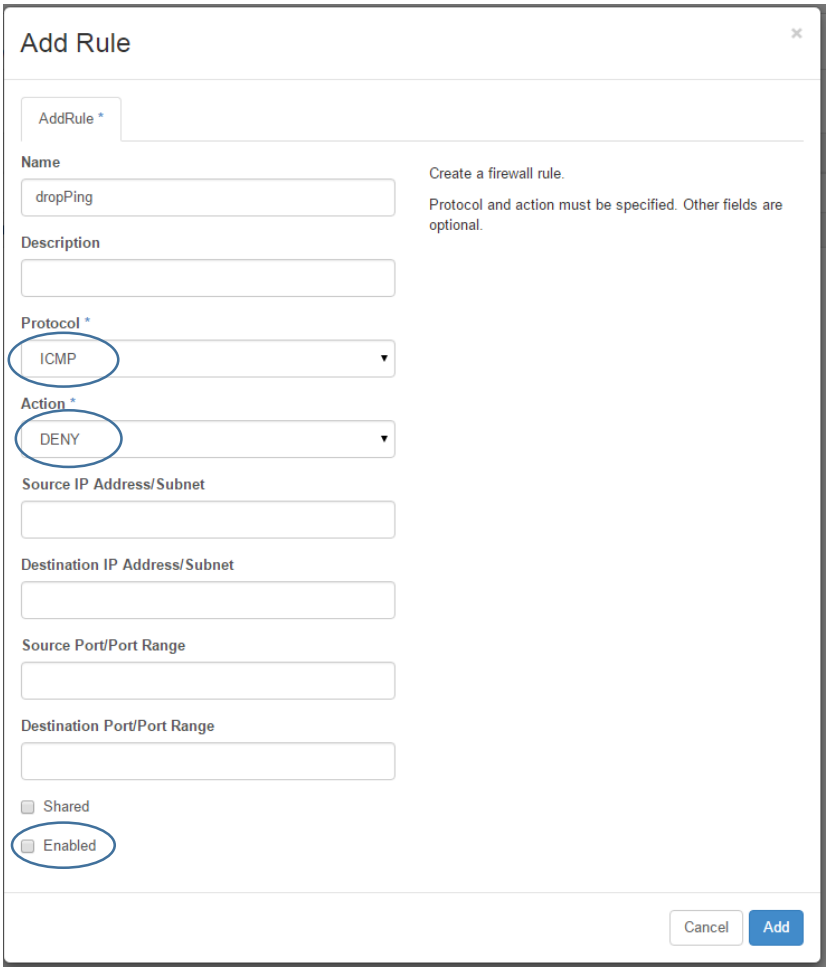
Jednou ze zajímavých výhod FWaaS je možnost použít různé implementace či chcete-li drivery. Pokud váš dodovatel firewallů nabízí kvalitní ovladače, můžete je začlenit do vaší Helion OpenStack instalace a uvedeným způsobem ovládat třeba přímo fyzické firewally. Součástí Helion OpenStack je softwarová open source implementace postavená na IPtables – tu si vyzkoušíme. Implementace využívá Distributed Virtual Routingu, tedy v případě FloatingIP jsou pravidla implementována na jednotlivých compute nodech bez nutnosti centralizace v nějakém nedistribuovaném virtuálním routeru. Z pohledu ovládání ale není rozdíl mezi touto a jinou implementací.

V čem je tedy smysl? Per-VM firewall je velmi vhodný pro mikrosegmentaci, ale můžete mít potřebu řídit celé své prostředí nebo projekt „centrálním“ firewallem. Jasně tak definuje co a jakým způsobem může komunikovat s vaším projektem.

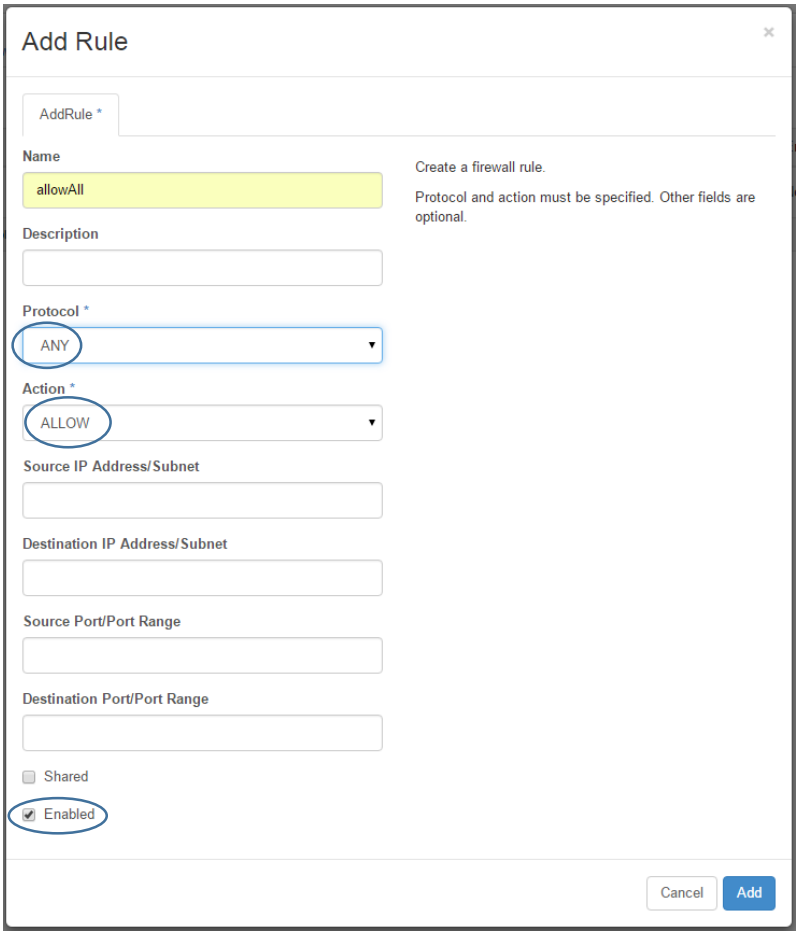
Jděte do záložky Network, Firewalls. Nahoře klikněte na Firewall Rules a pak na tlačítko Add Rule.



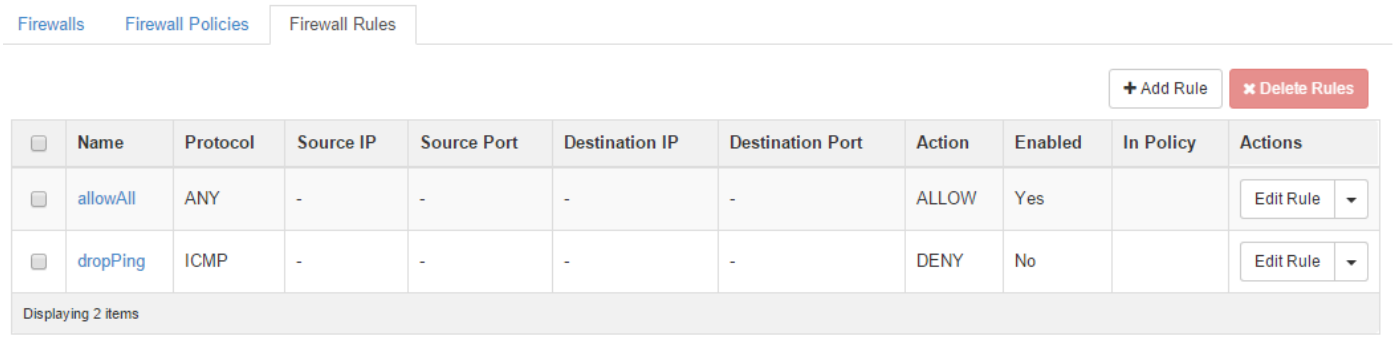
Přidejte pravidlo pro blokování Ping, ale zrušte jeho aktivitu (zrušte zaškrtnutí Enabled) a klikněte na Add.



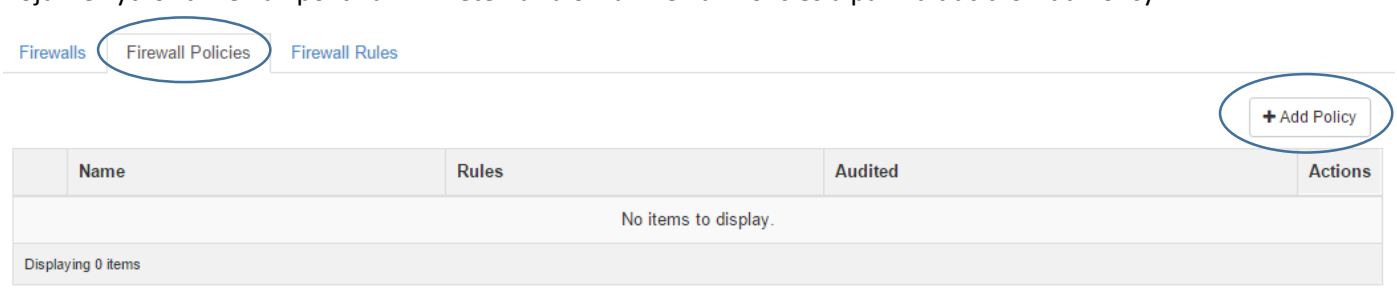
Přidejte druhé pravidlo povolující všechno.



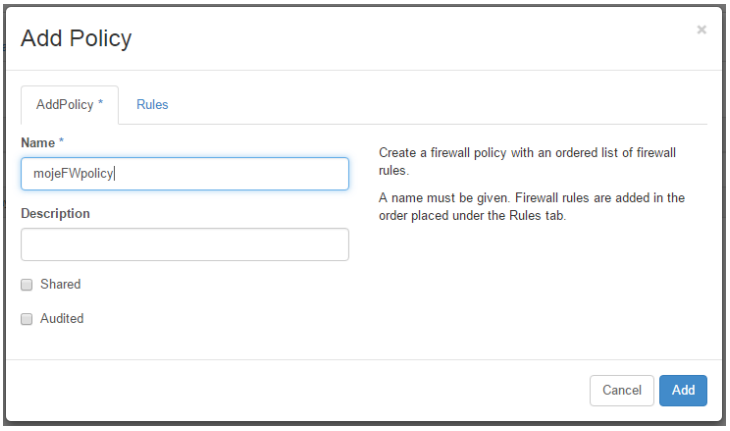
Výsledek by měl vypadat takhle.



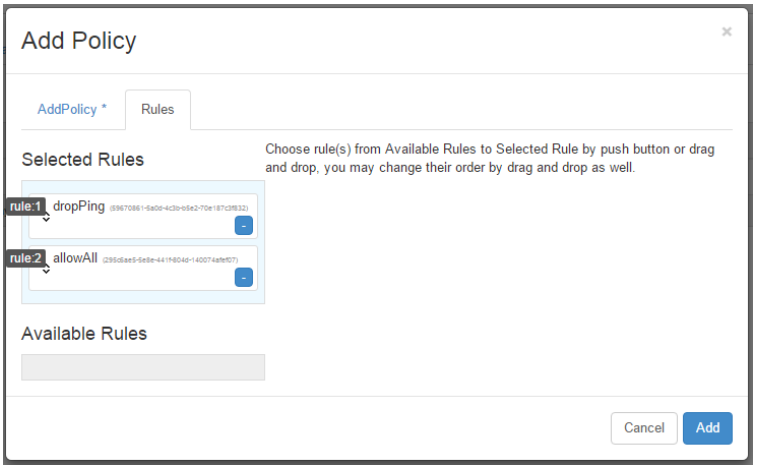
Pojďme vytvořit firewall politiku. Klikněte na záložku Firewall Policies a pak na tlačítko Add Policy.



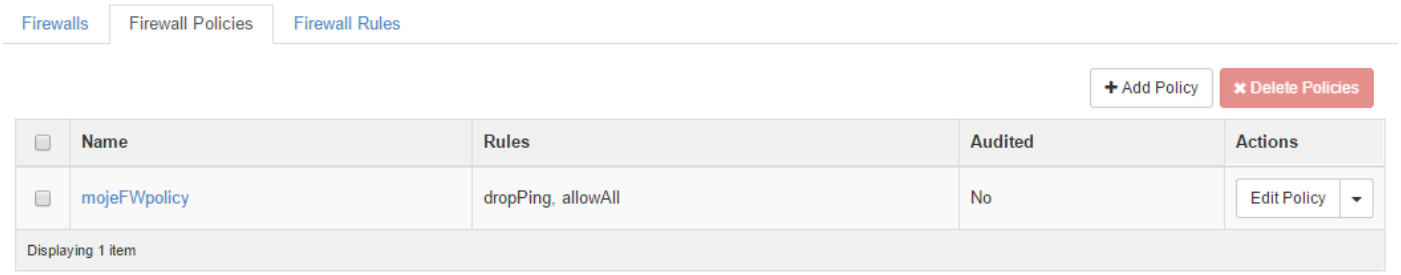
Pojmenujte vaší novou politiku a pak klikněte na Rules.



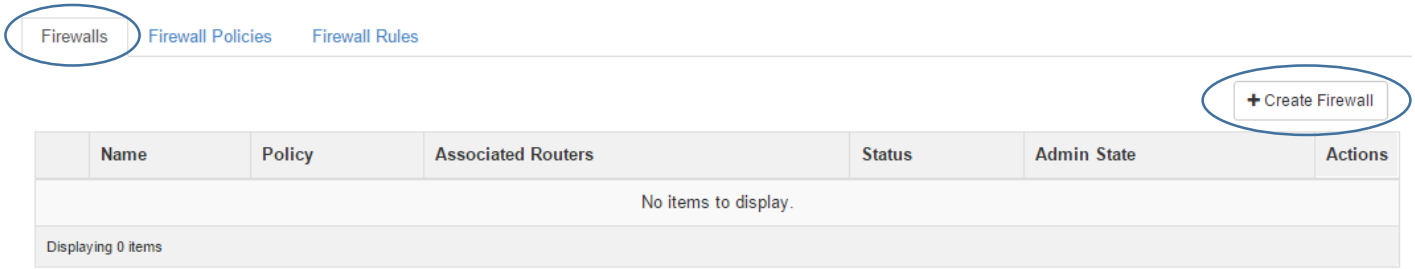
Přidejte pravidla ve správném pořadí, tedy nejprve zakaž ping a pak povol vše. Nakone klikněte na Add.



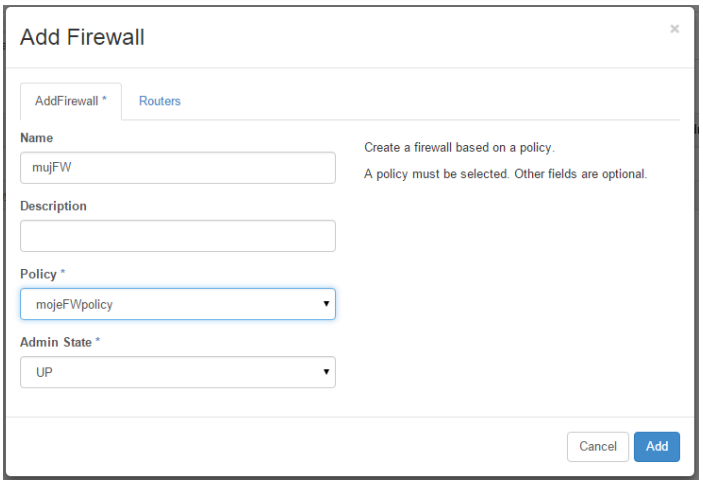
Výsledek bude vypadat takhle.



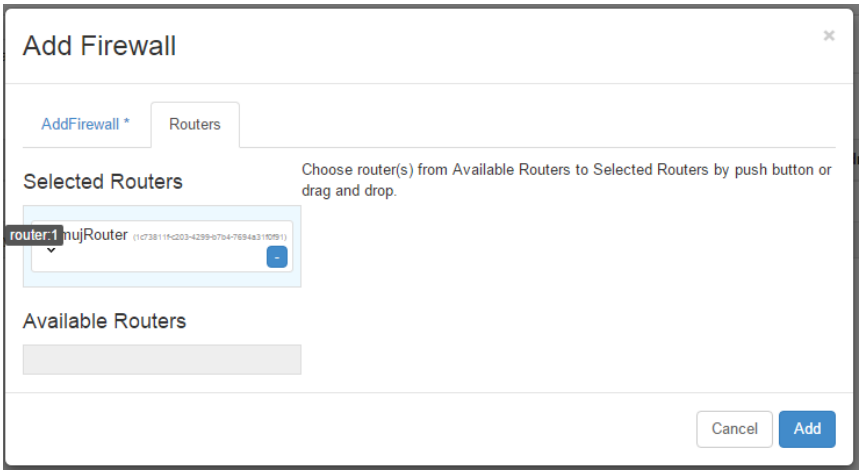
Teď už můžeme vytvořit firewall pro náš projekt. Klikněte na záložku Firewalls a tlačítko Create Firewall.



Pojmenujte svůj firewall a vyberte mu policy. Pak klikněte na Routers.



Přidejte svůj router a klikněte na Add.

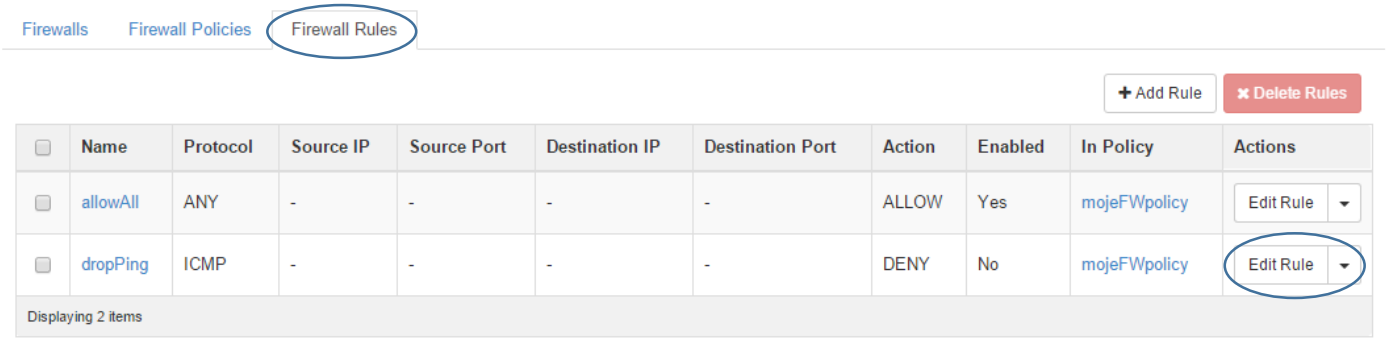


Připojte se na labServer a zkuste ping na FloatingIP, kterou ve svém projektu používáte. Ping bude procházet.

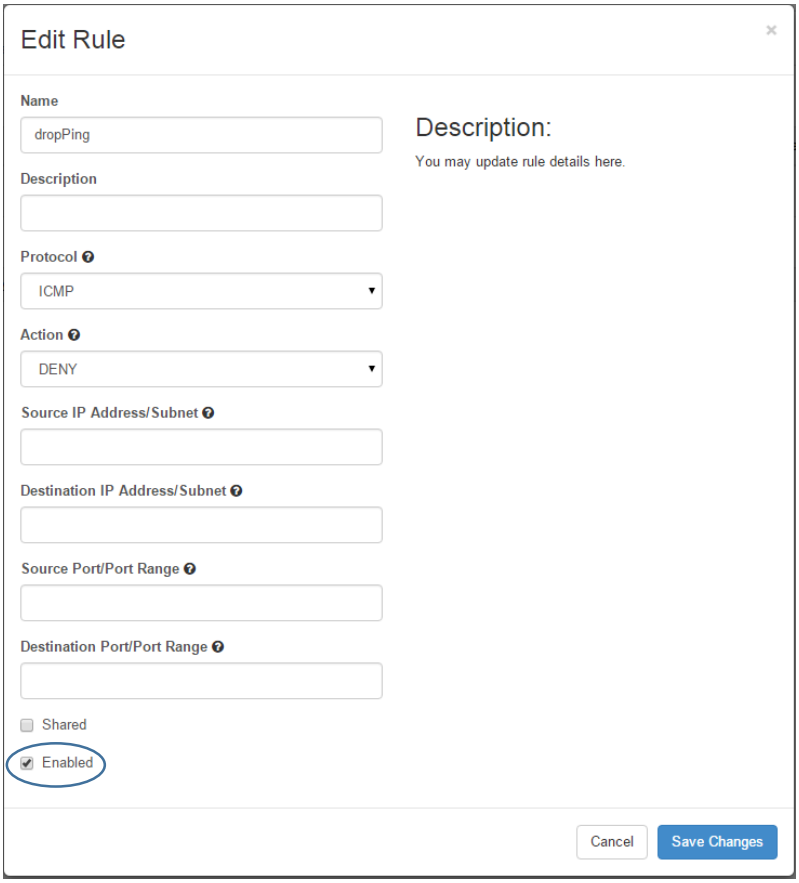
```
tomas@labserver:~$ ping 10.201.0.14
PING 10.201.0.14 (10.201.0.14) 56(84) bytes of data.
64 bytes from 10.201.0.14: icmp_seq=1 ttl=60 time=1.61 ms
64 bytes from 10.201.0.14: icmp_seq=2 ttl=60 time=0.972 ms
64 bytes from 10.201.0.14: icmp_seq=3 ttl=60 time=0.555 ms
```

```
^C
--- 10.201.0.14 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.555/1.047/1.616/0.438 ms
tomas@labserver:~$
```

Teď zapneme naše per-projekt firewallové pravidlo. Najděte si pravidlo dropPing a klikněte na Edit Rule.



Klikněte na Enabled a uložte změny.



Výsledek bude vypadat takhle.

Zkuste ping – nebude procházet.

```
tomas@labserver:~$ ping 10.201.0.14
PING 10.201.0.14 (10.201.0.14) 56(84) bytes of data.
^C
--- 10.201.0.14 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4031ms

tomas@labserver:~$
```

FWaaS nám funguje. Tímto způsobem můžete přehledně definovat jak váš projekt komunikuje se svým okolím, tedy uživateli. Součástí Helion OpenStack je open source implementace, ale můžete použít i drivery třetích stran od vašeho dodavatele firewallu.

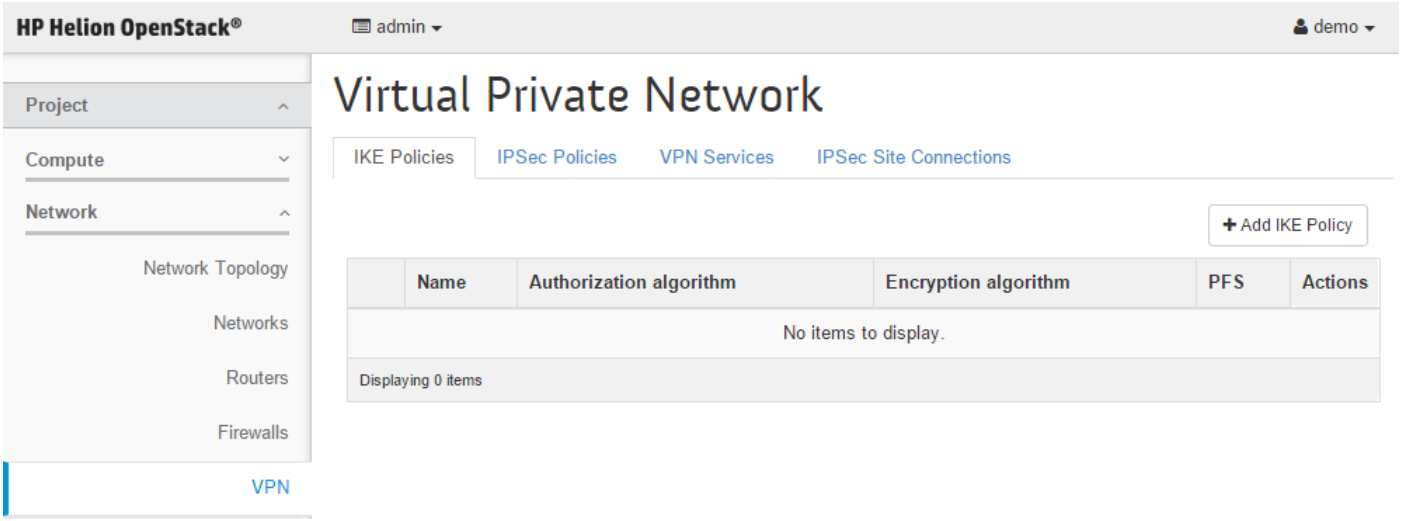
2.4.6. VPN as a service

Helion OpenStack od verze 2.0 nabízí VPN jako službu. Jde o sadu funkcí (API, CLI, GUI), které se potom implementují driverem. Existují ovladače pro komerční VPN koncentrátory a routery jako je Brocade Vyatta nebo Cisco (Helion OpenStack vám umožní tyto nainstalovat a podporovat co do správného volání jejich funkcí, nicméně konkrétní implementace samotného driveru je na externím subjektu). Součástí Helion OpenStack od verze 2.0 je implementace VPNaaS postavená na Strongswan, tedy plně open source, s podporou a v ceně produktu.

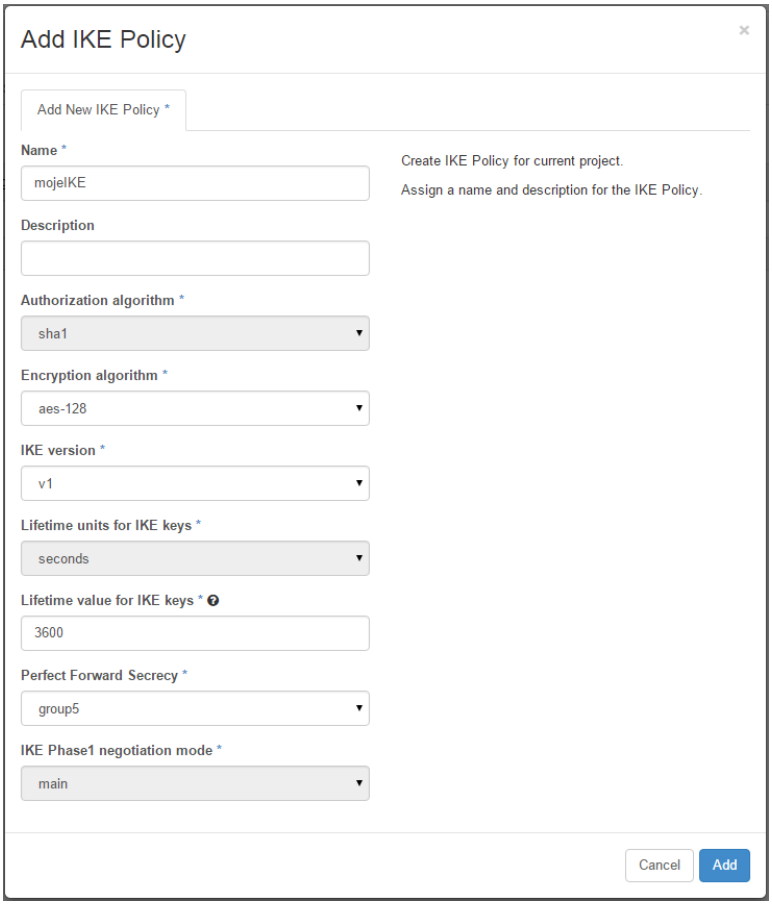
Jak to funguje? V našem labu jsme si vytvořili privátní síť mojeSit. Jak jsme schopni k ní přistupovat zvenčí? Vytvořili jsme router s bránou do externí sítě a službám pro uživatele přiřadili Floating IP z externího subnetu. Takto mohou uživatelé službu využívat. VPNaaS přichází s další možností. Můžete nabídnout IPSec přístup (tedy VPN) do tohoto subnetu. Například tedy IPSec router na vaší pobočce může vytvořit šifrovaný tunel do vašeho IaaS on-premise prostředí a dostat se do privátní sítě s instancemi. Výhodou je, že to funguje jako služba – každý tenant/projekt si to může ovládat sám, vše z jednotného prostředí přímo z OpenStack, s podporou multi-tenancy apod.

V našem labu aktuálně nemáme jak VPNaaS zkoušet jednoduše, protože je potřeba něco jako pobočkový router pro připojení do IaaS (připravujeme na později). Pojďme se ale podívat na samotné nastavení.

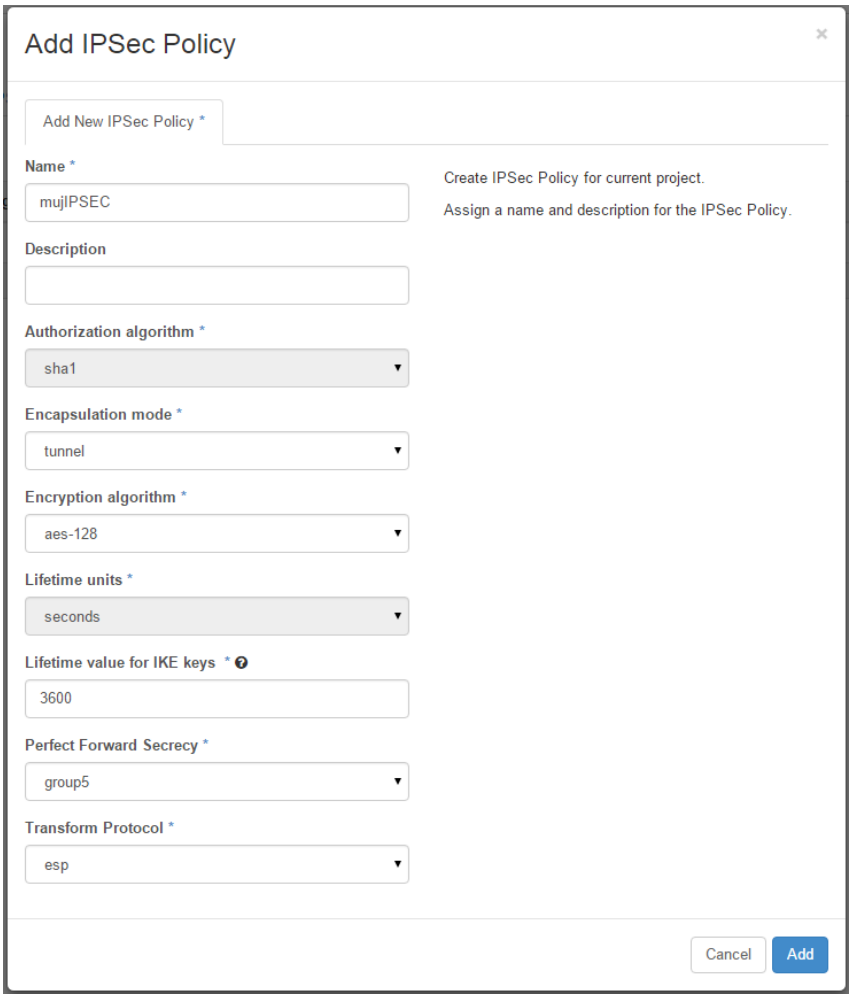
Nastavení najdete v záložce Network, VPN. V horní části pak definujete IPSec velmi podobně, jako u běžného routeru nebo firewallu – IKE politiku, IPSec politiku a pak to dáte dohromady.



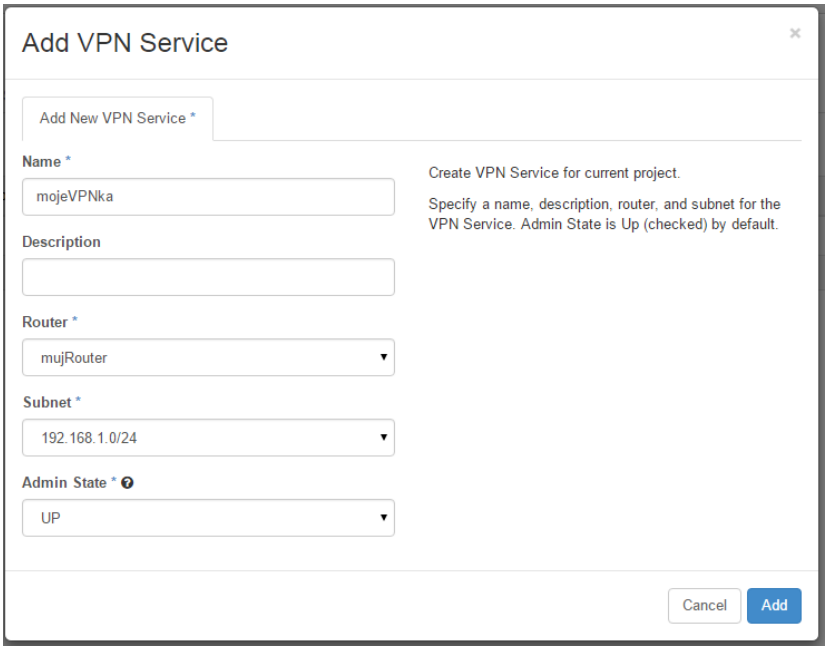
Nejprve nastavíte parametry IKE politiky



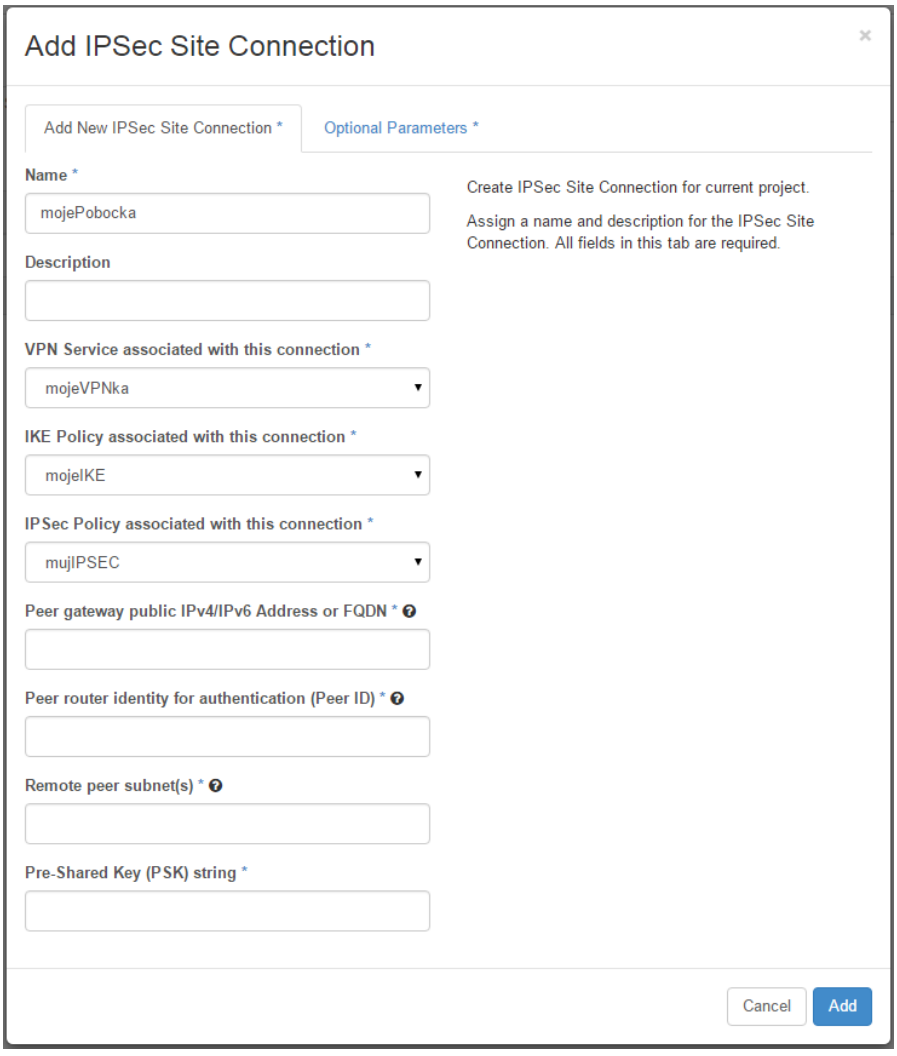
Takhle se nastavuje IPSec politika



Řekněte jak má vaše VPNka vypadat – tedy na jakém virtuálním routeru se implementuje a do jaké cílové privátní sítě vede.



Nakonec spojte všechny politiky dohromady do VPN služeb třeba pro pobočky.



2.4.7. Load-balancer as a service

Helion OpenStack od verze 2.0 podporuje službu load-balancingu. Typicky ji použijete pro zvýšení výkonu a dostupnosti webové farmy, rozdělení zátěže RESTful API na více nodů a tak podobně. Ve výchozím stavu instaluje Helion LBaaSv2. Tato novější generace má řadu výhod – například v brzké době bude v Helionu podporován OpenStack Octavia (škálovatelný open source balancer dokonce s podporou terminace TLS/SSL) – dnešní implementace je postavena na open source HAproxy. Pokud chcete používat referenční open source implementaci, která je v ceně řešení, nebo plánujete nasadit komerční balancer výrobce s podporou LBaaSv2 driverů, rozhodně zůstaňte u této verze. Jejím jediným nedostatkem je prozatimní absence GUI, nastavování tedy bude z příkazového řádku (ale to se brzy změní). Pokud trváte na GUI nebo váš dodavatel komerčního balanceru má zatím pouze drivery starší generace, můžete při instalaci Helion OpenStack zvolit LBaaSv1.

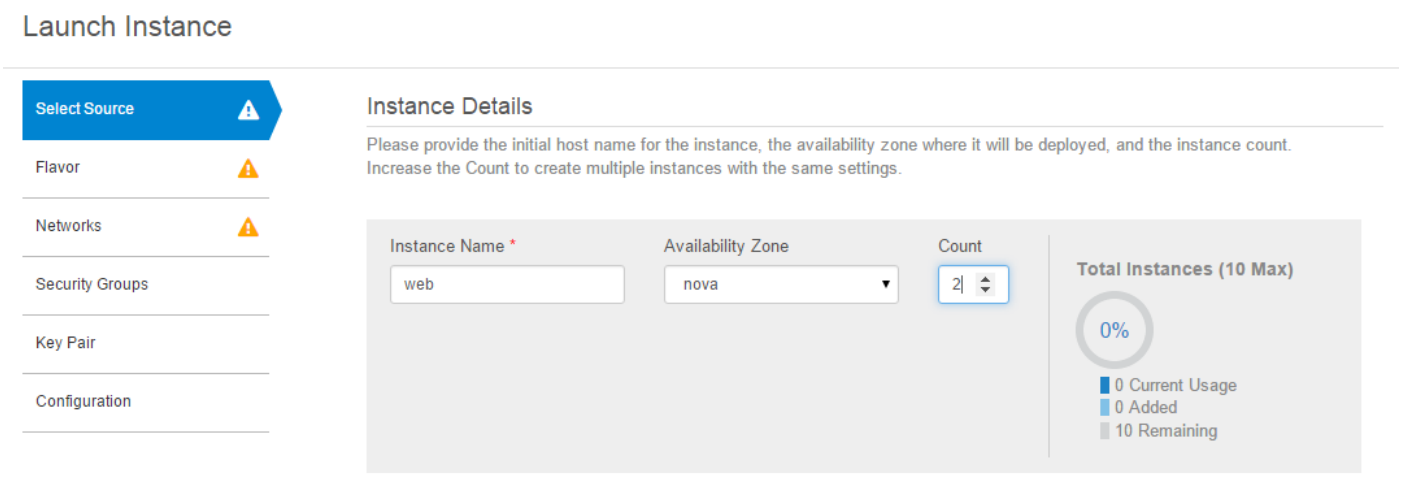
Vyzkoušejme si balancer v praxi.

Ujistěte se, že vaše Security Group povoluje port 80. Pokud ne, přidejte pravidlo.

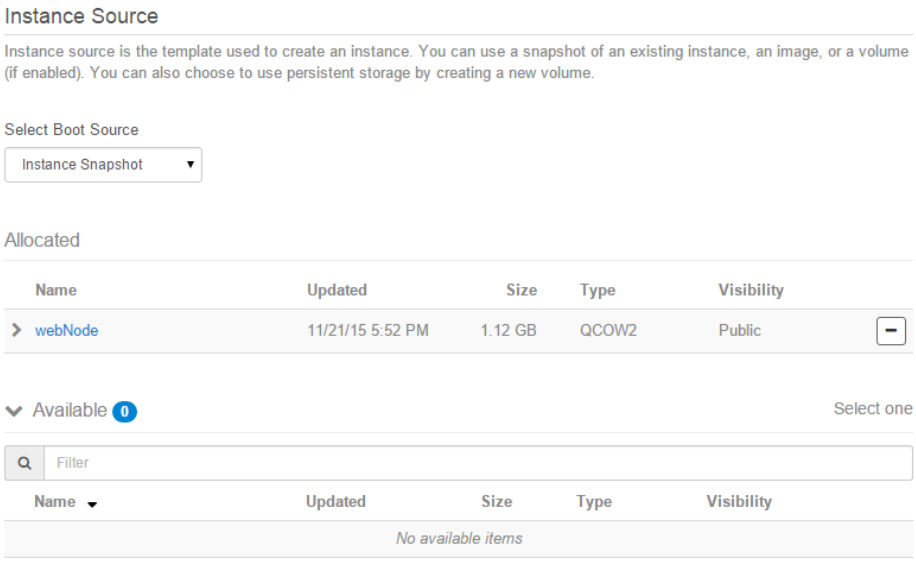
Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Actions
Ingress	IPv6	Any	Any	-	default	Delete Rule
Egress	IPv6	Any	Any	::/0	-	Delete Rule
Ingress	IPv4	Any	Any	-	default	Delete Rule
Egress	IPv4	Any	Any	0.0.0.0/0	-	Delete Rule
Ingress	IPv4	TCP	22	0.0.0.0/0	-	Delete Rule
Ingress	IPv4	TCP	80	0.0.0.0/0	-	Delete Rule

Displaying 6 items

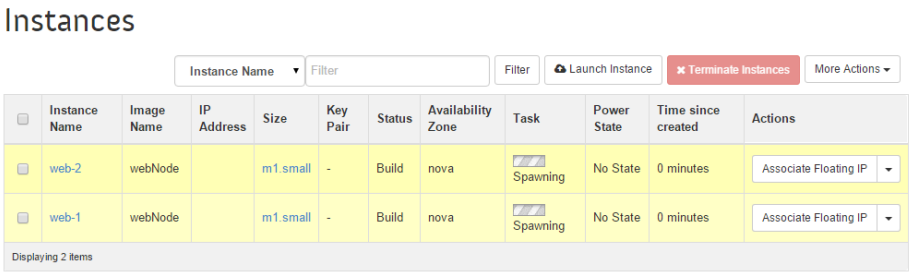
Vytvořte dvě instance webového serveru.



Startujeme z hotové Instance snapshot (s běžícím web serverem).



Použijte velikost m1.small a spusťte instance.



Přiřaďte těmto instancím Floating IP (už víte jak).

Připojte se na labServer, z kterého použijeme „textový prohlížeč“ curl. Ten se připojí na náš web server a zobrazí jeho odpověď. Na serverech běží jednoduchá služba, která vrátí text a privátní IP (takže dokážeme rozlišit kdo nám odpovídá).

```
tomas@labserver:~$ curl 10.201.0.10
Ja jsem web server 192.168.1.3
tomas@labserver:~$ curl 10.201.0.11
Ja jsem web server 192.168.1.4
tomas@labserver:~$
```

Webovou farmu máme připravenou, můžeme tedy začít balancovat provoz.

Helion OpenStack 2.0 v případě použití LBaaSv2 (výchozí možnost, velmi doporučuji tuto novější generaci) zatím nepodporuje grafické rozhranní. Musíme tedy do příkazové řádky. Podrobné informace o používání CLI získáte v druhé pokročilé části labu, pro tentokrát se jednoduše připojte do labServeru a opište níže uvedené příkazy.

Nejprve si načtěte komunikační proměnné (vysvětlíme v druhé části labu)

```
tomas@labserver:~$ source stack
```

Vytvořte nový load-balancer. Na konci musíte specifikovat název subnetu tak, jak jste ho v tomto labu vytvářeli (poud si nepamatujete, doporučoval jsem mujSubnet, nebo si najděte v GUI či příkazem neutron subnet-list).

```
tomas@labserver:~$ neutron lbaas-loadbalancer-create --name mujlb mujSubnet
Created a new loadbalancer:
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| description         |                                      |
| id                  | 9d4cd7a8-2d52-40f7-b1d0-45119246e59c |
| listeners           |                                      |
| name                | mujlb                                |
| operating_status    | OFFLINE                              |
| provider            | haproxy                              |
| provisioning_status | PENDING_CREATE                       |
| tenant_id           | d20fb4c9c0d645b4962484390a3be701     |
| vip_address         | 192.168.1.6                          |
| vip_port_id         | 8ebf59e6-5e97-4b93-866c-51374e5e7b97 |
| vip_subnet_id       | fab467ff-0767-4c8f-aa7d-a211f82360e3 |
+---------------------+--------------------------------------+
```

V následujícím kroku založíme listener (tedy jakou službu/protokol bude balancer nabízet ven).

```
tomas@labserver:~$ neutron lbaas-listener-create --loadbalancer mujlb --protocol HTTP --protocol-port=80 --name mujlis
Created a new listener:
+---------------------------+------------------------------------------------+
| Field                     | Value                                          |
+---------------------------+------------------------------------------------+
| admin_state_up            | True                                           |
| connection_limit          | -1                                             |
| default_pool_id           |                                                |
| default_tls_container_id  |                                                |
| description               |                                                |
| id                        | c443cfe3-977d-49fb-a6fd-64e8895d4cb0           |
| loadbalancers             | {"id": "9d4cd7a8-2d52-40f7-b1d0-45119246e59c"} |
| name                      | mujlis                                         |
| protocol                  | HTTP                                           |
| protocol_port             | 80                                             |
| sni_container_ids         |                                                |
| tenant_id                 | d20fb4c9c0d645b4962484390a3be701               |
+---------------------------+------------------------------------------------+
```

Jak budeme rozhazovat zátěž? Vytvořme pool a zvolíme jednoduchý balanční mechanismus round robin.

```
tomas@labserver:~$ neutron lbaas-pool-create --lb-algorithm ROUND_ROBIN --listener mujlis --protocol HTTP --name mujpool
Created a new pool:
+---------------------+------------------------------------------------+
| Field               | Value                                          |
+---------------------+------------------------------------------------+
| admin_state_up      | True                                           |
| description         |                                                |
| healthmonitor_id    |                                                |
| id                  | 92ace8aa-4683-42a3-b61c-ac6f8c2507f1           |
| lb_algorithm        | ROUND_ROBIN                                    |
| listeners           | {"id": "c443cfe3-977d-49fb-a6fd-64e8895d4cb0"} |
| members             |                                                |
| name                | mujpool                                        |
| protocol            | HTTP                                           |
| session_persistence |                                                |
| tenant_id           | d20fb4c9c0d645b4962484390a3be701               |
+---------------------+------------------------------------------------+
```

Přidejme teď jednotlivé servery identifikované jejich privátní IP adresou.

```
tomas@labserver:~$ neutron lbaas-member-create --subnet mujSubnet --address 192.168.1.3 --protocol-port 80 mujpool
Created a new member:
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| address        | 192.168.1.3                          |
| admin_state_up | True                                 |
| id             | 7bf74eac-7f14-4804-ab5e-f16214ba2168 |
| protocol_port  | 80                                   |
```

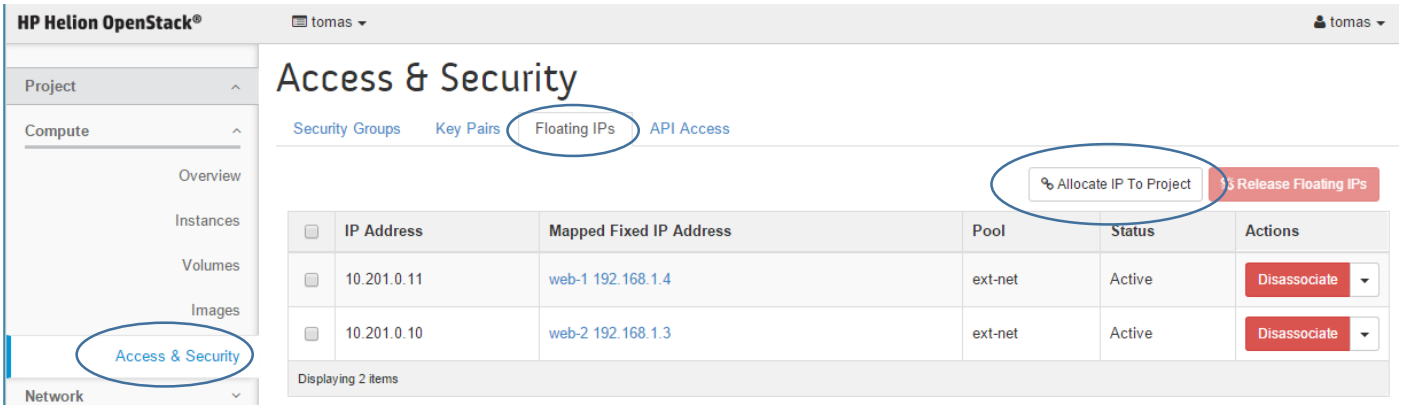
```
| subnet_id       | fab467ff-0767-4c8f-aa7d-a211f82360e3 |
| tenant_id       | d20fb4c9c0d645b4962484390a3be701     |
| weight          | 1                                    |
+-----------------+--------------------------------------+
tomas@labserver:~$ neutron lbaas-member-create --subnet mujSubnet --address 192.168.1.4 --protocol-port 80 mujpool
Created a new member:
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| address         | 192.168.1.4                          |
| admin_state_up  | True                                 |
| id              | fc8f968d-715f-49ae-b26a-5252bef7797b |
| protocol_port   | 80                                   |
| subnet_id       | fab467ff-0767-4c8f-aa7d-a211f82360e3 |
| tenant_id       | d20fb4c9c0d645b4962484390a3be701     |
| weight          | 1                                    |
+-----------------+--------------------------------------+
```

Máme skoro hotovo, balancer běží, ale nemá přiřazenu externí adresu. Vypište si informace o vašem balanceru.

```
tomas@labserver:~$ neutron lbaas-loadbalancer-show mujlb
+---------------------+------------------------------------------------+
| Field               | Value                                          |
+---------------------+------------------------------------------------+
| admin_state_up      | True                                           |
| description         |                                                |
| id                  | 9d4cd7a8-2d52-40f7-b1d0-45119246e59c           |
| listeners           | {"id": "c443cfe3-977d-49fb-a6fd-64e8895d4cb0"} |
| name                | mujlb                                          |
| operating_status    | ONLINE                                         |
| provider            | haproxy                                        |
| provisioning_status | ACTIVE                                         |
| tenant_id           | d20fb4c9c0d645b4962484390a3be701               |
| vip_address         | 192.168.1.6                                    |
| vip_port_id         | 8ebf59e6-5e97-4b93-866c-51374e5e7b97           |
| vip_subnet_id       | fab467ff-0767-4c8f-aa7d-a211f82360e3           |
+---------------------+------------------------------------------------+
```

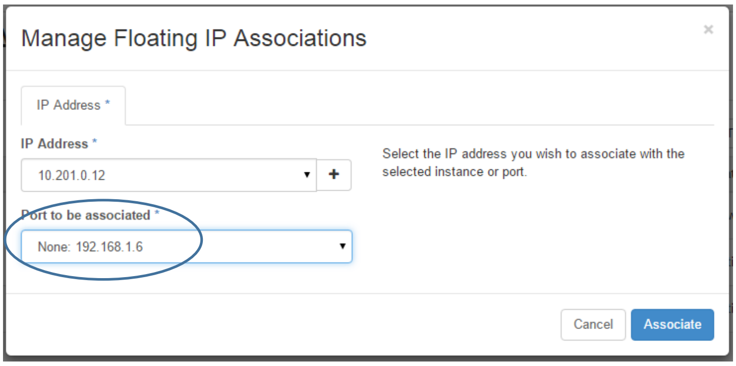
Poznamenejte si vip_address (virtuální IP balanceru).

Zažádejme si o novou Floating IP



Až ji budete mít, klikněte na Associate.

Přiřaďte Floating IP virtuální IP našeho balanceru (pamatujete z předchozího výpisu?).



	IP Address	Mapped Fixed IP Address	Pool	Status	Actions
	10.201.0.12	Load Balancer VIP 192.168.1.6	ext-net	Down	Disassociate
	10.201.0.11	web-1 192.168.1.4	ext-net	Active	Disassociate
	10.201.0.10	web-2 192.168.1.3	ext-net	Active	Disassociate

Displaying 3 items

Jděte do labServer a vyzkoušejte si curl na venkovní VIP. Zkuste to několikrát. Jak vidíte, provoz se nám rozkládá!

```
tomas@labserver:~$ curl 10.201.0.12
Ja jsem web server 192.168.1.4
tomas@labserver:~$ curl 10.201.0.12
Ja jsem web server 192.168.1.3
tomas@labserver:~$ curl 10.201.0.12
Ja jsem web server 192.168.1.4
tomas@labserver:~$ curl 10.201.0.12
Ja jsem web server 192.168.1.3
tomas@labserver:~$ curl 10.201.0.12
Ja jsem web server 192.168.1.4
tomas@labserver:~$ curl 10.201.0.12
Ja jsem web server 192.168.1.3
tomas@labserver:~$ curl 10.201.0.12
Ja jsem web server 192.168.1.4
tomas@labserver:~$ curl 10.201.0.12
Ja jsem web server 192.168.1.3
tomas@labserver:~$ curl 10.201.0.12
Ja jsem web server 192.168.1.4
tomas@labserver:~$ curl 10.201.0.12
Ja jsem web server 192.168.1.3
```

Samozřejmě jednotlivé nody nemusí mít přiřazenou externí Floating IP, to jsme dělali jenom proto, abychom vyzkoušeli, že web server funguje v pořádku.

Zbývá dořešit poslední věc. Pokud jeden ze serverů vypadne, balancer to nepozná a bude na něj některé požadavky i nadále směrovat. Potřebujeme tedy ještě zapnout health check našich serverů.

```
tomas@labserver:~$ neutron lbaas-healthmonitor-create --type HTTP --pool mujpool --delay 1 --max-retries 1 --timeout 1
Created a new healthmonitor:
+----------------+------------------------------------------------+
| Field          | Value                                          |
+----------------+------------------------------------------------+
| admin_state_up | True                                           |
| delay          | 1                                              |
| expected_codes | 200                                            |
| http_method    | GET                                            |
| id             | b784c85c-5b9a-4385-b6e6-2a44470a5c34           |
| max_retries    | 1                                              |
| pools          | {"id": "92ace8aa-4683-42a3-b61c-ac6f8c2507f1"} |
| tenant_id      | d20fb4c9c0d645b4962484390a3be701               |
| timeout        | 1                                              |
```

```
| type            | HTTP                                                |
| url_path        | /                                                   |
+-----------------+-----------------------------------------------------+
```

Teď můžete rozjet curl – například napište do řádky jednoduchou smyčku, která bude přistupovat na web každou vteřinu. Provoz se balancuje na oba servery. Pak jednu z web instancí zapauzujte nebo vypněte. Uvidíte, že balancer už na tento server nebude dál nic posílat.

```
tomas@labserver:~$ while true; do curl 10.201.0.12; sleep 1; done
Ja jsem web server 192.168.1.4
Ja jsem web server 192.168.1.3
Ja jsem web server 192.168.1.4
Ja jsem web server 192.168.1.3
Ja jsem web server 192.168.1.4
Ja jsem web server 192.168.1.3
Ja jsem web server 192.168.1.4
Ja jsem web server 192.168.1.3
Ja jsem web server 192.168.1.4
Ja jsem web server 192.168.1.3
Ja jsem web server 192.168.1.4
Ja jsem web server 192.168.1.4
Ja jsem web server 192.168.1.4
Ja jsem web server 192.168.1.4
Ja jsem web server 192.168.1.4
Ja jsem web server 192.168.1.4
Ja jsem web server 192.168.1.4
Ja jsem web server 192.168.1.4
```

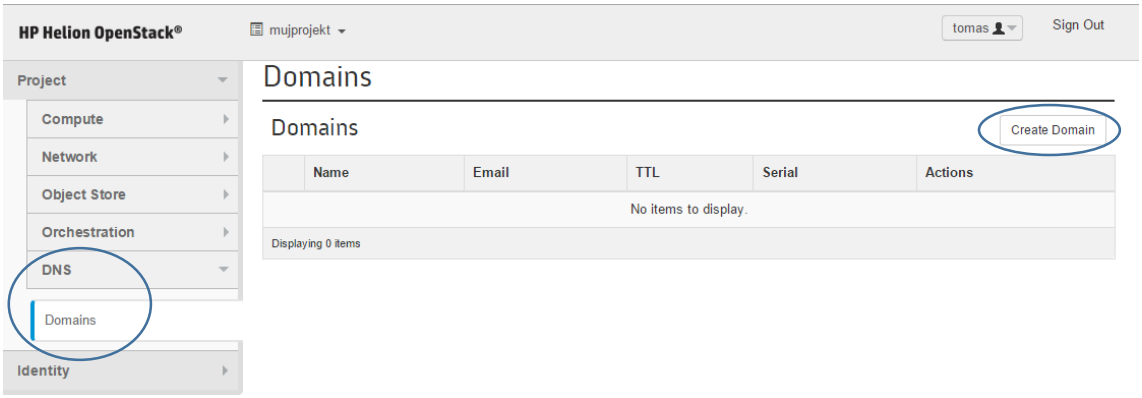
Tohle je tedy LBaaS, velmi příjemná funkce. Aktuální implementace je postavena na HAproxy, ale v blízké době se přejde na OpenStack Octavia (větší škála a výkon, možnost TLS terminace). Řada výrobců komerčních balancerů nabízí svoje drivery, které můžete v Helion OpenStack využít a ovládat tak komerční balancery jako je F5, Citrix, Brocade, Radware nebo A10.

2.4.8. DNS as a service

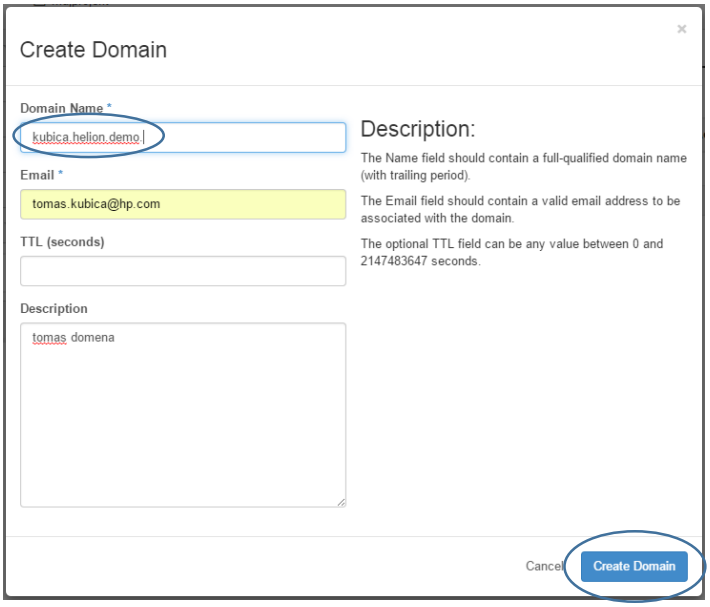
Následující návod je zatím z verze Helion OpenStack 1.1, update čekejte v další verzi lab guide.

Udělali už jsme hodně práce a na jejím konci máme připravenou infrastrukturu včetně startovacích diskových obrazů – můžeme začít instalovat a updatovat aplikace a jejich komponenty (o automatizaci těchto kroků částečně později v tomto labu a v plně krásé v druhém pokročilejším labu). Nicméně zbývá nám ještě jeden nešvar – komunikovat budeme přes IP adresu, ale pro reálné použití by se více hodilo DNS jméno. Jak to zařídit? Hledáme způsob, jak může tenant vytvářet DNS záznamy sám a to konzistentně bez ohledu na to, jaký DNS server je v infrastruktuře využíván (to je mimo dikci tenanta). Helion OpenStack pro toto nabízí DNSaaS, na jejímž backendu může být PowerDNS (Linux), Microsoft DNS z těch on-premise nebo DNS služba od cloud poskytovatele (podporovaná je DynECT a Akamai).

Otevřete záložku DNS a klikněte na Create Domain



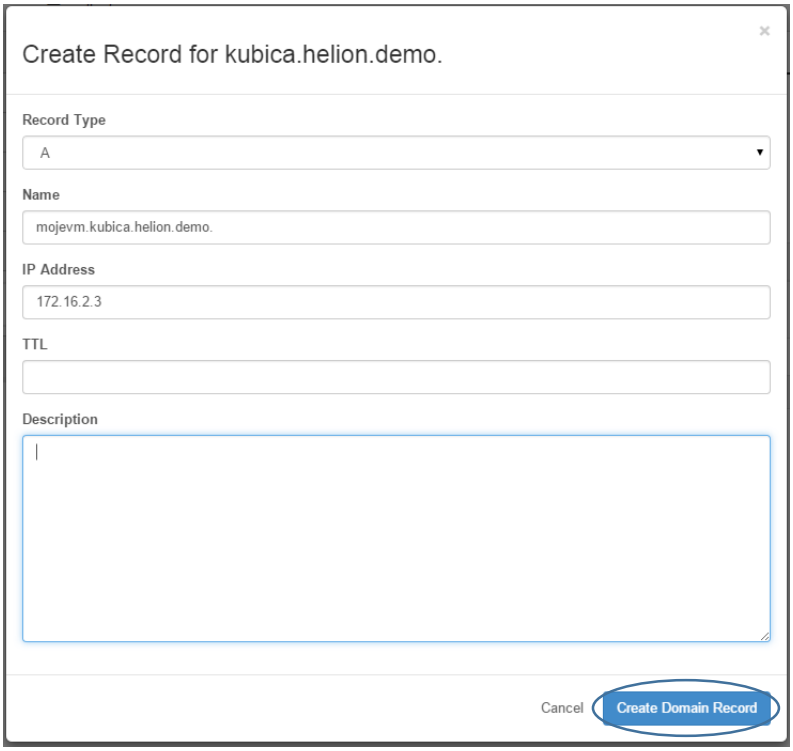
V rámci labu máme doménu helion.demo – přidejte si před ní svoje příjmení a vytvořte si tak svůj strom.
Nezapomeňte ukončit tečkou.

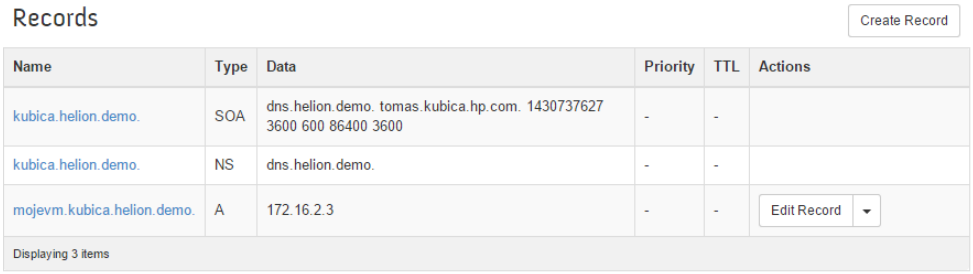


Přidáme si záznam, který namíříme na naší Floating IP. Klikněte na Manage Records.

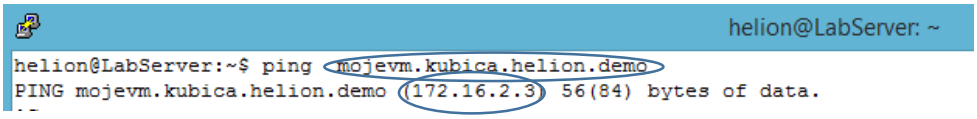
Root záznamy byly vytvořeny automaticky. Klikněte na Create Record

Vytvořte nový záznam s vaší Floating IP (u jména nezapomeňte na tečku na konci)





Přihlašte se na server v labu (jako v předchozí části) a zkuste ping (nebo nslookup či dig podle vaší libosti). Překládá se vám doménové jméno?



Takto tedy funguje DNSaaS. V další části labu už ho nebudeme potřebovat. Abychom šetřili zdroje labu, smažte teď záznam a doménu.

2.5. Efektivní práce s Image

Jak vypadají vaše image (nebo chcete-li šablony) ve vaší klasické virtualizaci? Jedna z nejčastějších otázek v diskusích kolem OpenStack s KVM jako hypervisorem je, jak mám konvertovat svoje VMware obrazy. Odpovědi na to sice existují, ale že je užitečnější zamyslet se nad celou koncepcí. Nasazení IaaS jako je OpenStack a potenciální snížení závislosti na jediném hypervisoru (docela dobré a typické nasazení je v produkci KVM a vSphere a v dev/test KVM a VirtualBox s Vagrantem) je dobrou příležitostí přemýšlet o vaší strategii kolem tvorby a používání diskových obrazů.

Přemýšlení o obrazech vašich VM začneme v tomto labu, ale pokročilé postupy rozeberem až v jeho druhé části. Dnes se zaměříme na to, jak to udělat, aby v šabloně nebyla uložena hesla. Tedy aby přístupové údaje nebyly vlastnost samotné image, ale dosadili se až v okamžiku vytvoření instance. Dále si vyzkoušíme jednoduché spouštění iniciačního skriptu – jednoduché dokonfigurování VM po vytvoření instance.

Jak můžeme přistupovat k tvorbě a používání diskových obrazů či chcete-li šablon?

- Image vznikají tak, že něco ve VM ručně kutáme a výsledek uložíme jako nový obraz
    - Výhoda: Nemusíme se nic učit, takhle se to dělalo i v prehistorii

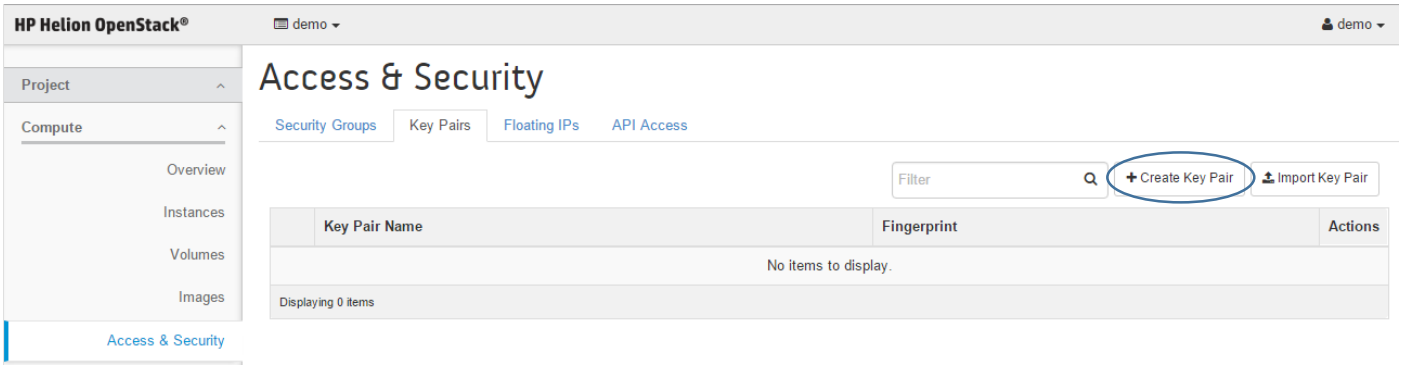
    - Nevýhoda: Každý image je unikát, vše trvá dlouho, vzniká velké množství chyb, opakovatelnost je velmi problematická, dokumentace prakticky neexistuje nebo je špatně, závislost (více či méně) na hypervisor platformě (a s tím spojená špatná přenositelnost a problém nekonzistencí mezi vývojem, QA a produkcí)
- Máme základní diskový obraz a konfigurační nástroje, které ho dostanou do stavu požadovaného aplikací
    - Výhoda: Vše je automatizované a dopadne vždy stejně, výborná a živá dokumentace, nezávislost na hypervisor platformě a přenositelnost
    - Nevýhoda: Příprava služby (aplikace) trvá dlouho, troubleshooting vyžaduje znalosti
- Používáme immutable servery, tedy image je read only a pro každou verzi aplikace vždy vzniká nový fixní obraz
    - Výhoda: Rychlý start, vysoká bezpečnost (lze třeba i vypnout SSH přístup apod.), pro provoz jednoduché nasazení, při použití vhodných nástrojů lze jedním postupem vygenerovat funkčně identické obrazy pro různé hypervisory a platformy (přenositelnost)
    - Nevýhoda: Nutnost změnit development strategii a naučit se používat nové nástroje předávání údajů aplikacím místo tradičních konfiguračních souborů

Tato témata jsou pokročilejší a v dnešním labu se jich dotkneme jen okrajové. V druhé pokročilé části si ukážeme nástroje jako Server Automation, Ansible, Chef nebo Packer.

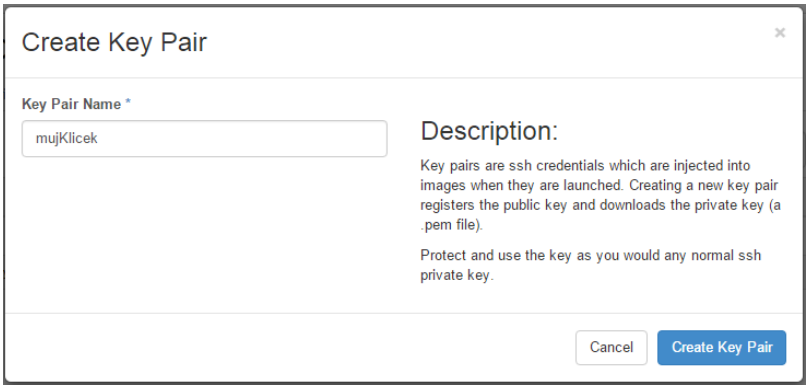
2.5.1. Pryč s hesly ze šablony, používejme Key Pair při startu

V automatizovaném prostředí samozřejmě můžete mít image virtuálních strojů se zabudovanými přihlašovacími hesly, ale není to ideální situace. Pokud jde o snapshot nějaké hotové aplikace, tak proč ne, ale jde-li o operační systém připravený pro další použití, znamenalo by to bezpečnostní riziko. Image upravený pro cloudové prostředí by měl být schopen se přizpůsobit situaci (v případě Linux to spočívá především v implementaci cloud-init balíčku při startu). Nechme stranou konkrétní mechanismus – místo hesla můžeme v Helion OpenStack vygenerovat SSH klíče (což je podstatně pohodlnější a bezpečnější) a ty použijeme místo hesel. Při startu instance zajistí OpenStack „vsunutí“ správného klíče do vašeho image, klíče, který patří konkrétnímu projektu, ne univerzální heslo.

Vygenerujeme si nové klíče (můžeme jich v projektu používat hned několik). Jděte do záložky Compute - Access & Security a pak Key Pairs.



Klikněte na Create Key Pair a vytvořte nový klíč



Klíč si stáhněte k sobě (to co získáváte je jeden klíč z páru, ten druhý se vsune do instance – vysvětlení detailů fungování PKI jsou nad rámec dnešního labu). Otevřete soubor a zkopírujte do schránky.

Připojte se na labServer. Nejprve si tento privátní klíček uložíme do souboru. Napište příkaz cat přesměrovaný do souboru a vložte zkopírovaný obsah klíče (u většiny klientů ho vložíte pravým tlačítkem). Nakonec zmáčněte CTRL+D.

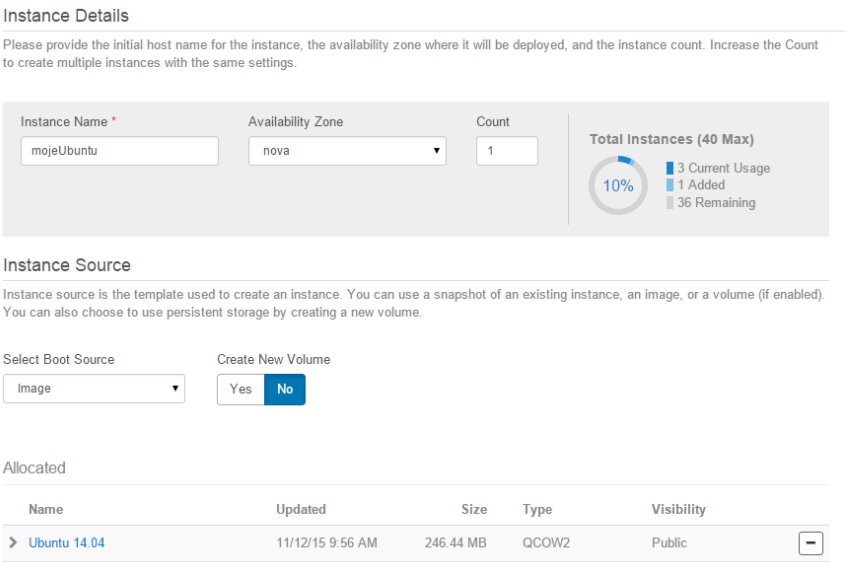
```
tomas@labserver:~$ cat > mujKlicek.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA71NJTyY+boPuDYSIUaBeosap0jrKIgjkQi+8Qr1atKK3VDvq
0OJgPUXl4O9JCOvdQiDz3aTY9gEbhP9AO2U2MYNYdF8qJ9MSi6ifn7ErqbU2a2OT
fbRspTnctSEaqYv80Vw4USZN58oqU62FgcepzV/kx5Dvnx36tH5RR+8z8zgHdFPi
W95HGgLx1W+6iOfY1I769z+4Q3T5bvC9LMvc2W1P384yCT96TBHT0gMeUqM1ojzN
xndHP+uPpoHMABGqIrrNN4SYYmSB+eBuzKHMiX35HlXWjQXIPyLqhOTYFIFBJeDs
+x5mCzhNxPe3vhht/a0Y7M5Dgwt4fH1nyPCd/QIDAQABAoIBADX0aeedMKALwERt
56m3ZP5/mVObC20G4icFygSl2eg1cu1boMG894N42a2PZMDNJBG/ihsjCgLUFxcx
0JJTbBdXjD6YIdHepSS1PF9tOvHEt+MYDO1fGstZMyfmsbMdqz6r8spgv1mNW2OI
EDxE/kQd5V8UjuEpihbdD4gPJoAjZvCO4c7tDXhy2ra2tEXSeIBL20hPDqujv5yM
GONjxni1II2cmHP8DlNDSS3w3Y9AW8V9repkxN1RsEFuHtmVPk8yeIVYf7OfN0+u
bDEkZWdthQDCI4nyD6VgtkmHDTEPDTpvWd5FmPtwFg77LrduuSwH2DYzEA40qJ8o
6trNNqECgYEA/QBfVH9AJNT/dcJt/phFT84TyvM1ifRrAZSclnW42M1AMe8cAWlg
rSNpwubD3XpWjY8o68nfBZ3OlstrmorF/tm+6h5rNEF7Q+JoonoRx0+ICGv5rK+b
ErqsCsgM4gvxac1b+D+/k2z1rNuTU/VqarLFsHkIJX9CBYXSVwtg4AUCgYEA8ilr
ZAV+VS8FRDiL07zQyqUiZW5DV3J/yNLL9PZOA+IdQjWRCwh6s1HDCS/2pXVQbu8i
coYNRoj/VW35jCr69rwSHzlykfSKCMMk6PiKd2lZpar59/9iQsGIxzL4Ykm0ntpN
eJf20wQfOMciI2NCfbWDleG5OMwI9DSeYdD7v5kCgYEAkY3NSoeLF6WS8uTQ81AX
UDp3GKOjgaKkjVw6WjWQCurKq++sZQODIxjkl8S7mofvk7FxEXYqYMjROd/+IAMG
tf//3iFx+7ZQfFWdbRxdbhVLZcz472h4BuZuZCWDg+jrErua1c+XH/HnxXLt57eh
aZFAOq7nCOuVyCedQ4bATSECgYBJ7itDFgpDp19MPJczxWlY9KFTph4ZDHPGs9Rg
rPGUbevQ0tm9LJGJPWT14RbD3NT5iThTDmnvJtQNGM4e5OBJg5Fkxv0bYjTiB/G0
zmw3mIot8czu0aEGEF/ZsM3z89yYwrz0HDDWq2N8yg66Dwu1pTzO/WK23FO/enEA
G/U/wQKBgQC+R8cZN6HcfVIZUwr8aUYp9ZaxDpjoiS2ErZt3MuV0FoUC7jicr4Rz
xpkHWOuaQo0253AKRo8E5QIyo8fatM997hkdPHhpymjnIf+DnHvpe3LbIy58DA0s
txOhvxGhAHLwNgeETER8LahQWBPDAj75PoAJulvR+rKFvIzpFe0/Sg==
-----END RSA PRIVATE KEY-----
tomas@labserver:~$
```

Aby bylo možné soubor používat, musíme nastavit správná práva.

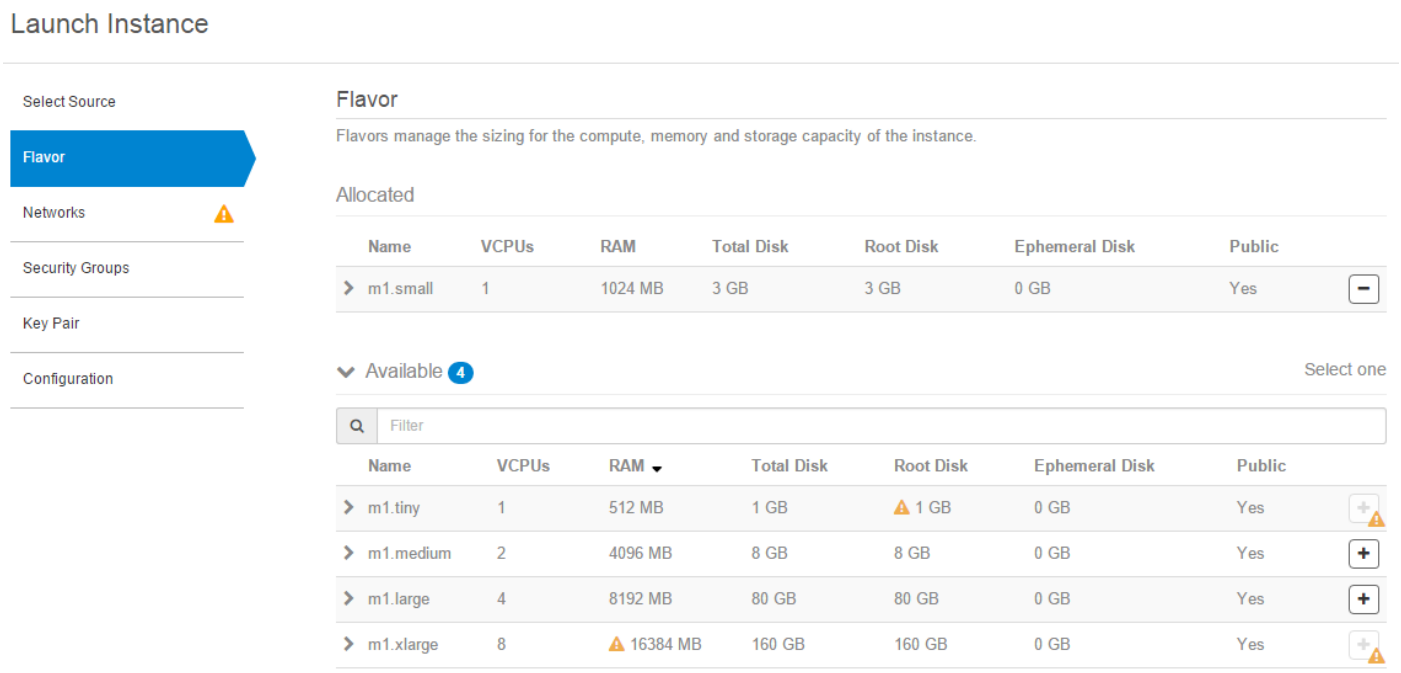
```
tomas@labserver:~$ chmod 600 mujKlicek.pem
```

To máme připravené. Spustíme teď instanci Ubuntu. Jde o tzv. cloud image, tedy hotový Ubuntu obraz, který je připravený pro cloudové nasazení. To se pozná především tak, že obsahuje utilitku cloud-init, která se spouští hned po startu a dokáže komunikovat s OpenStack. Tímto způsobem se do VM po startu dostanou potřebná nastavení, v našem případě SSH klíč pro výchozího uživatele (tím je user „ubuntu“).

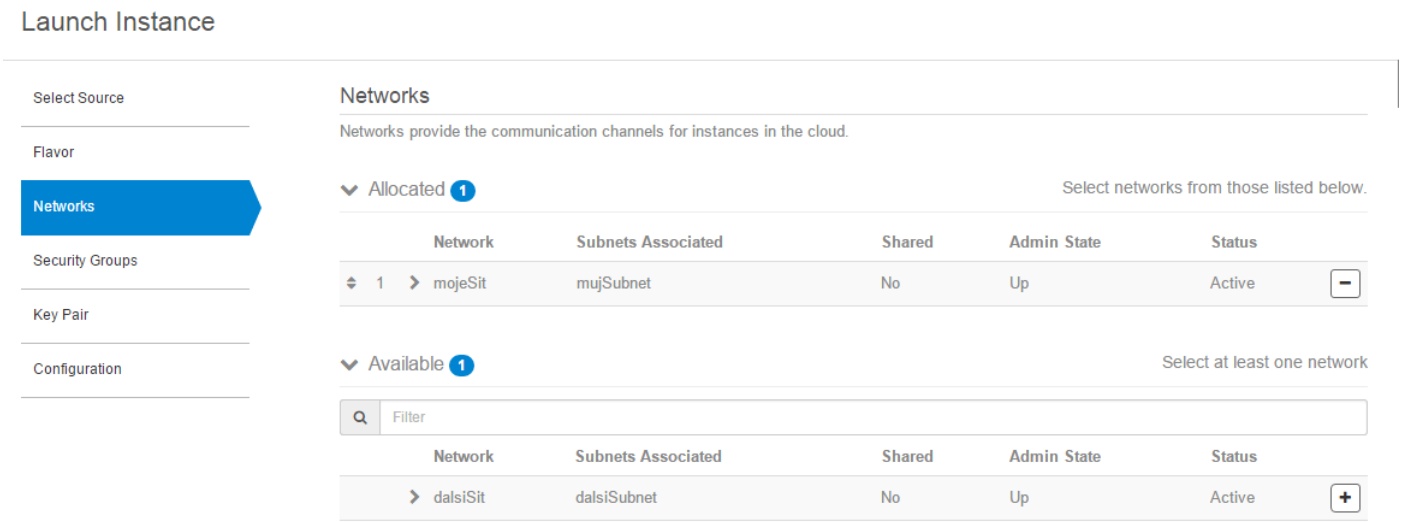
Vytvořte instanci Ubuntu



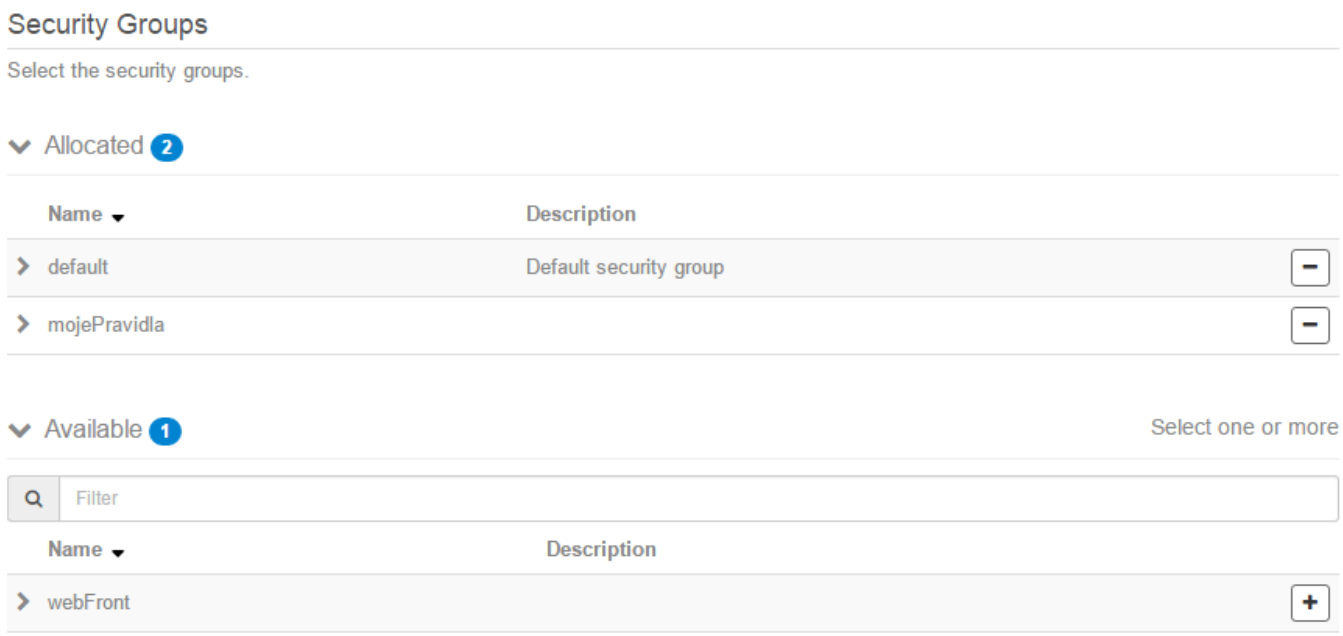
Vyberte si Flavor. m1.tiny má příliš malý disk pro tento image, na což vás průvodce upozorňuje. Zvolte tedy m1.small.



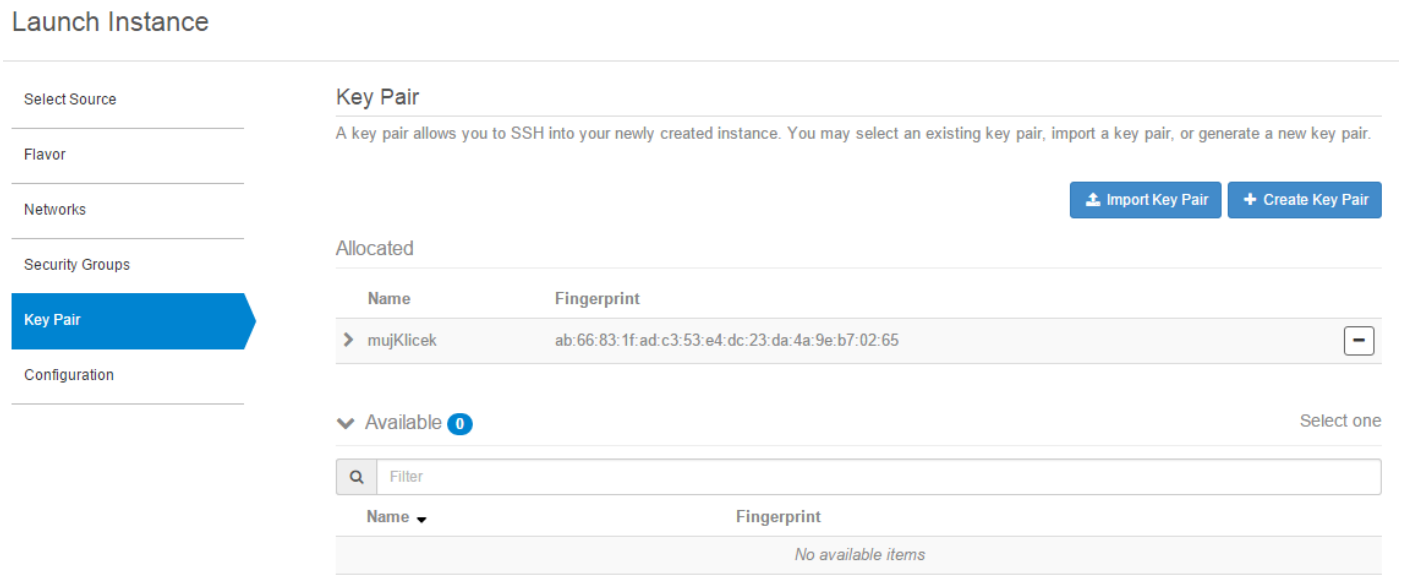
Alokujte síť.



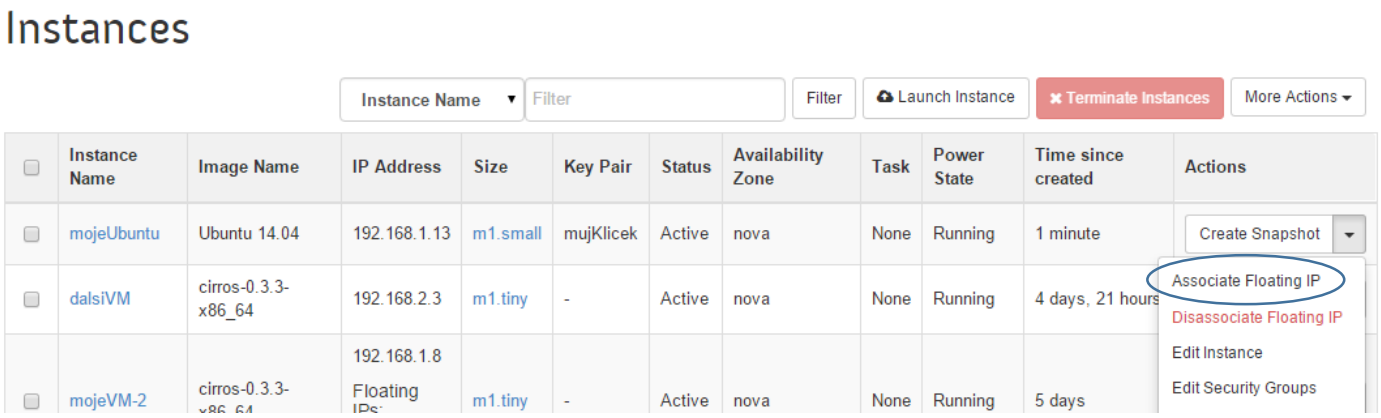
Přidejte Security Group povolující SSH přístup.



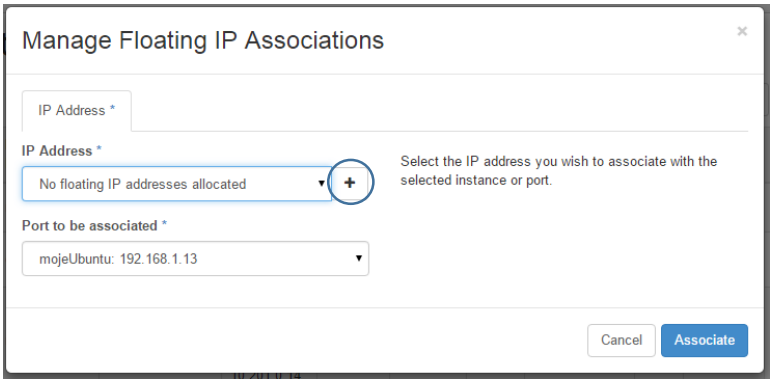
A teď to nejdůležitější – přiřaďte váš bezpečnostní klíč.

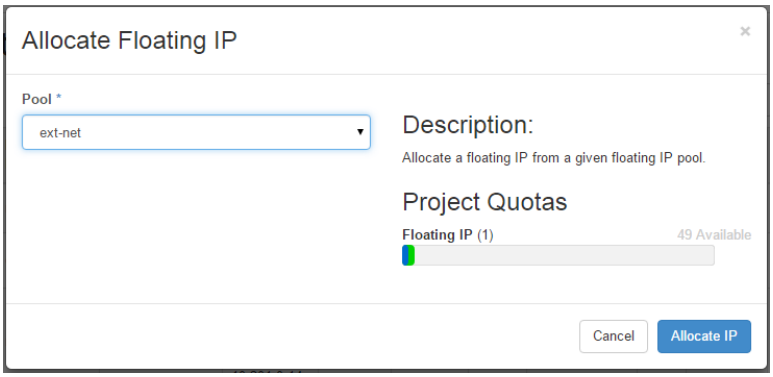


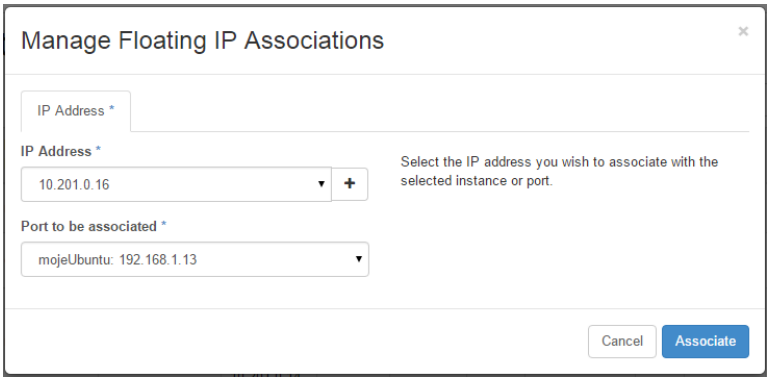
Teď už můžete instanci spustit. Přiřaďte jí novou FloatingIP.



Zážádejte o novou IP kliknutím na symbol +







Jděte do labServeru a pokuste se připojit bez vašeho klíčku.

```
tomas@labserver:~$ ssh ubuntu@10.201.0.16
The authenticity of host '10.201.0.16 (10.201.0.16)' can't be established.
ECDSA key fingerprint is 10:fe:42:6b:42:94:a3:96:1d:f7:7a:0f:8c:a3:46:51.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.201.0.16' (ECDSA) to the list of known hosts.
Permission denied (publickey).
```

Neúspěch a ani nejste dotázáni na heslo (je to v cloud image ve výchozím stavu zakázané, navíc pro účet ubuntu žádné neexistuje). Zkuste to znovu s využitím vašeho klíčku.

```
tomas@labserver:~$ ssh -i mujKlicek.pem ubuntu@10.201.0.16
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-66-generic x86_64)

...

ubuntu@mojeubuntu:~$
```

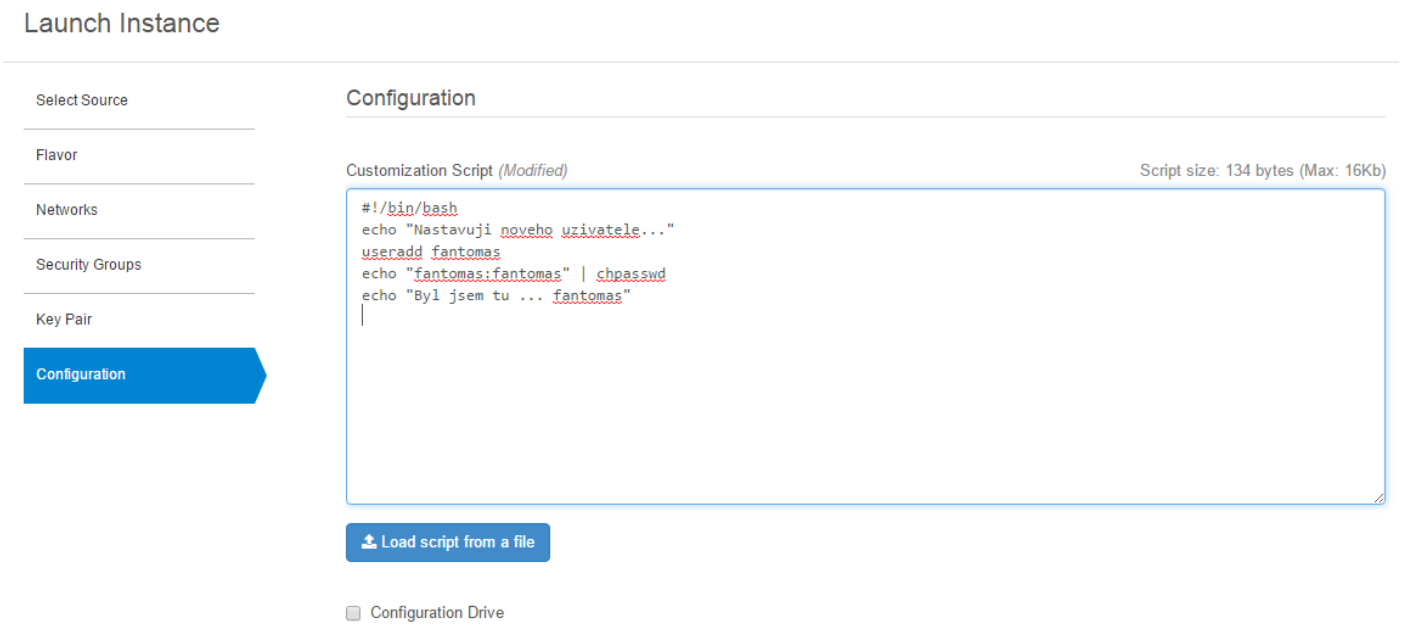
Jsme tam!

2.5.2. Iniciační skripty

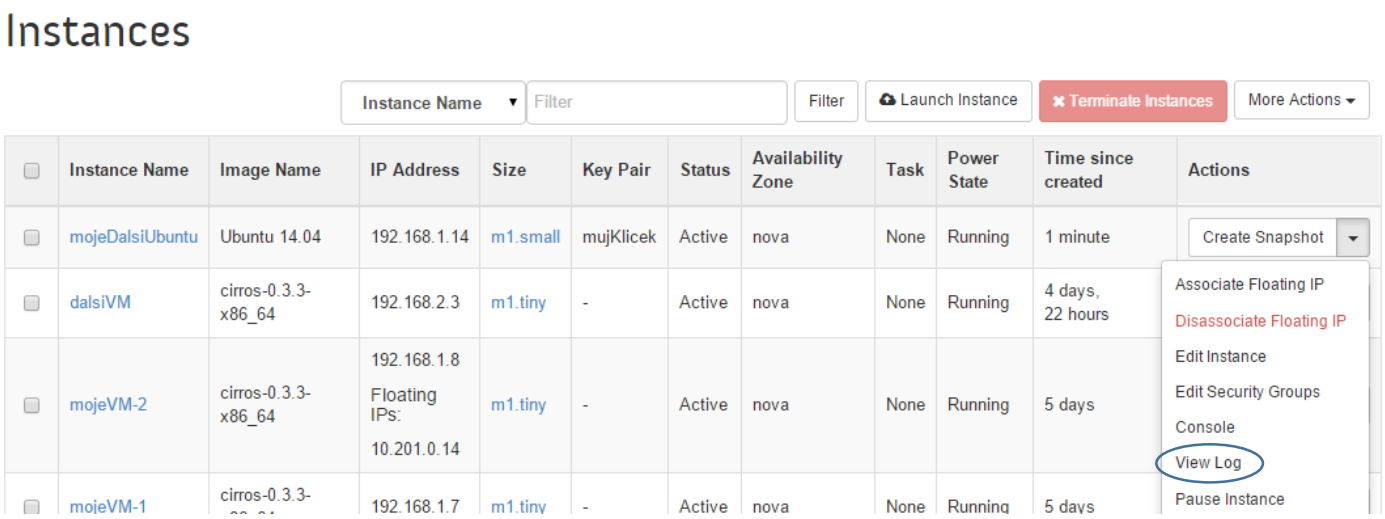
Pokud používáte image se zabudovaným cloud-init (tedy něco, co se obvykle označuje jako cloud-ready image), můžete při startu instance spustit vámi definovaný skript. Buď jednoduchý tak, jak to v rámci labu uděláme my (do 16KB kódu), nebo i komplexnější ve formě konfiguračního disku.

Zlikvidujte předchozí instanci Ubuntu image a vytvořte novou. V průvodci postupujte úlně stejně jako v předchozím labu, ale dojděte až na záložku Configuration. Do okénka Customization Script napište následující skript. Ten vytvoří nového uživatele fantomas s heslem fantomas.

```
#!/bin/bash
echo "Nastavuji noveho uzivatele..."
useradd fantomas
echo "fantomas:fantomas" | chpasswd
echo "Byl jsem tu ... fantomas"
```



Spusťte instanci a po té co naběhne se podívejte do logu. Klikněte na View Log



Zjistili jste přítomnost Fantomase?

Instance Details: mojeDalsiUbuntu

Create Snapshot

Overview | Log | Console | Action Log

Log Length 35 Go View Full Log

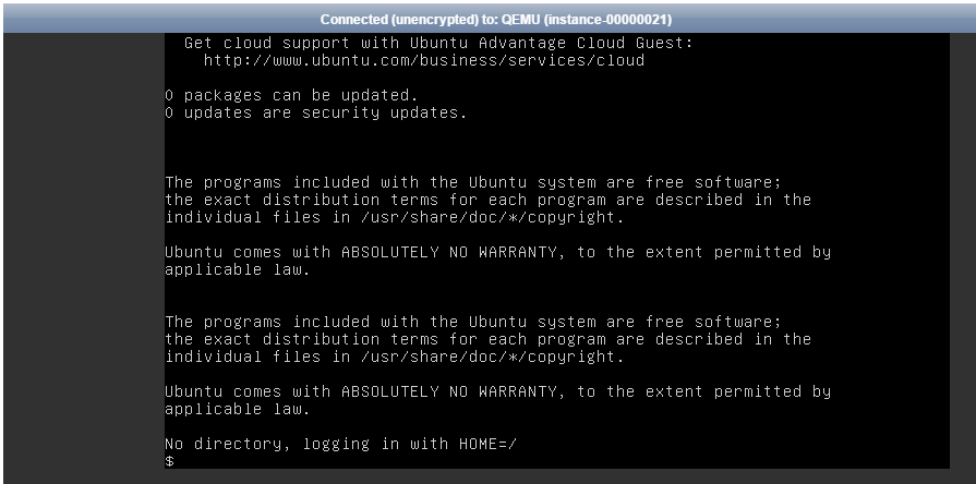
Instance Console Log

```
  landscape-client is not configured, please run landscape-config.
 * Restoring resolver state...       [80G [74G[ OK ]
Cloud-init v. 0.7.5 running 'modules:config' at Wed, 18 Nov 2015 09:32:52 +0000. Up 28.46 seconds.
 * Stopping System V runlevel compatibility[74G[ OK ]

Ubuntu 14.04.3 LTS mojedalsiubuntu ttyS0

mojedalsiubuntu login: Generating locales...
  en_US.UTF-8... up-to-date
Generation complete.
Cloud-init v. 0.7.5 running 'modules:final' at Wed, 18 Nov 2015 09:32:58 +0000. Up 34.35 seconds.
Nastavuji noveho uzivatele...
Byl jsem tu ... fantomas
ci-info: ++++++++++Authorized keys from /home/ubuntu/.ssh/authorized_keys for user ubuntu++++++++++
ci-info: +---------+-------------------------------------------------+---------+-------------------+
ci-info: | Keytype |                Fingerprint (md5)                | Options |      Comment      |
ci-info: +---------+-------------------------------------------------+---------+-------------------+
ci-info: | ssh-rsa | ab:66:83:1f:ad:c3:53:e4:dc:23:da:4a:9e:b7:02:65 |    -    | Generated-by-Nova |
ci-info: +---------+-------------------------------------------------+---------+-------------------+
ec2:
ec2: #############################################################
ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
ec2: 1024 10:ea:c7:3a:34:58:21:75:a1:ca:44:1b:34:3e:78:55  root@mojedalsiubuntu (DSA)
ec2: 256 e2:e2:6d:98:36:66:e1:37:3f:34:0f:b2:8e:dc:c5:ea  root@mojedalsiubuntu (ECDSA)
ec2: 256 c6:44:55:78:ab:db:bb:c4:96:d5:33:77:af:d2:7a:10  root@mojedalsiubuntu (ED25519)
ec2: 2048 f3:77:0e:86:e9:b1:f0:bf:c7:65:61:4c:f0:a5:5e:5b  root@mojedalsiubuntu (RSA)
ec2: -----END SSH HOST KEY FINGERPRINTS-----
ec2: #############################################################
-----BEGIN SSH HOST KEY KEYS-----
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBOT1CyWZblGrNk3Vavq9mvUAEHHW+gcGf7GVO5YyQWi4wJifaF5MdHE+IncQuO1Q3EACHB5/kF8wXk
/O2ZXEggs= root@mojedalsiubuntu
ssh-ed25519 AAAAC3NzaC11ZDI1NTE5AAAAIHMaTF5eoQcqjMZyBF1k28aLJXV4Nwq6D/0A2183E+vP root@mojedalsiubuntu
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDwUxZsd78xhNDwVs+mu0yUgoepLM3L6qSySebk4O97VeZvKKOf1aY1j3Cvj+fAegcp2mNYXN04MBB+LW4ZSyZVXieecnbNimLNaXx/SFuRGdhNlP
5r5RHC4EnN4cIs4MCIy4W01MTE1nL191BmMI24nXuj4XE4Crq4zXVyWOwNFHrsWTvqOwFzyjIFWxVorfZv3IsP1Oo9MJ4DXvP1YrOzPwjgE7oOLcJfiKDjmKoCHbNwOF38gEHPXe3dLj140rQtDjqd
Rm4CPqJT7VFJ723wR1aapgq5tfHEBLbajs9DtG5ZSZE9O4yNNe4dizdr3ZPViuP66Moepp+IqbVy7QFp root@mojedalsiubuntu
-----END SSH HOST KEY KEYS-----
Cloud-init v. 0.7.5 finished at Wed, 18 Nov 2015 09:32:59 +0000. Datasource DataSourceOpenStack [net,ver=2].  Up 36.06 seconds
```

Připojte se do konzole a přihlaste se tímto účtem.

Instance Console

If console is not responding to keyboard input: click the grey status bar below. Click here to show only console
To exit the fullscreen mode, click the browser's back button.



Gratuluji – právě jste se naučili upravit výsledný OS přímo při vytváření instance. Tímto způsobem můžete například nastavit potřebné parametry pro to, aby se do OS dostala nějaká další služba či nástroj pro složitější kroky, jako je HPE Server Automation, Chef, Puppet apod. Můžete také třeba nastavit http proxy, nainstalovat nějaký balíček apod.

2.5.3. „Dotahování" VM aneb immutable image

Základem této strategie je úvaha, že je riskantní nacházet se v nějakém mezistavu. Možná to znáte z některých svých činností – někdy je rychlejší a jednodušší začít od začátku, než se pokoušet napravit něco, co se vám nebo někomu jinému zrovna nepovedlo. Někdy to třeba rychlejší a jednodušší není, ale je tu jedna jistota. Pokud vyjdete ze stejných startovních podmínek a budete opakovat identický postup, dojdete ke stejnému výsledku. Aby byla zajištěna přesnost v postupu, bude ho dělat robot (ve formě vhodného nástroje). Pokud tedy potřebujete obraz s nainstalovanými komponentami jako je web server, aplikační server, knihovny a tak podobně, začnete ze základního obrazu s operačním systémem a pak si stáhnete všechny potřebné aktualizace a komponenty dle předpisu. Mohl by to vlastně být nějaký skript, ale to je strašně špatně udržitelné a museli byste pokaždé vymýšlet všechno znova. Sáhněte tedy raději po vhodném nástroji, který to udělá přehledně, předvídatelně, spoustu věcí vyřeší za vás, existují k němu rozsáhlé knihovny hotových modulů a příkladů a ideálně dokonce podporuje metodu desired state (tedy definujete cílový stav a na pořadí operací nezáleží).

Co použít? Nejstarší CFEngine bych dnes už doporučoval nechat odpočívat, ale můžete vyzkoušet jednoho ze dvou soupeřících dospěláků – Puppet a Chef. První je napsaný v Ruby, zaměřuje se na desired state princip (tedy deklarativní přístup), na serverech používá agenta a je velmi oblíbený. Ten druhý je také v Ruby (a část v Erlang), má řadu zajímavých vychytávek, je spíše řešen imperativně (více se podobá programování) a kromě konfigurace serverů exceluje i v dalších zajímavostech (testování softwaru nebo compliance check). Možná vás ale zaujmou i mladí, kteří na to jdou bez agentů a pro většinu lidí jsou jednodušší na seznámení. Tím jsou Ansible a SaltStack, oba napsaní v Python, oba hojně využívající nádherně čitelné předpisy (YAML a Jinja šablony). Já osobně pro začátek doporučuji Ansible, ale pak není špatné zkusit třeba Chef, který je velmi rozšířený (ale všechno je otázkou osobního vkusu a potřeb – v této čtyřce neuděláte chybu). Více o Ansible a Chef najdete v mém českém lab guide pro Helion OpenStack 2.0 (v sekci ke stažení bude už brzy, dám vědět). Pokud hledáte něco, co si poradí se vším včetně třeba UNIX systémů a má velmi bohatou knihovnu, nabízí GUI, compliance a je zavedené a prověřené v enterprise, zkuste komerční HPE Server Automation.

Co tedy potřebujete? Používáte pouze základní image – Okna 2012 R2, Ubuntu 14.04, CentOS 7 a to je třeba všechno. Pak máte ale připravené role (recepty, scénáře, …) pro jednotlivé typy serverů. Potřebujete novou verzi knihovny, jiné sestavení vaší aplikace nebo změnu nastavení? Rozhodně ji neprovedete ručně, ale zadáte ji do předpisu vámi vybraného nástroje. Většina z nich vám umožní stejnou sestavu spustit na vašem už běžícím serveru (na to existuje krásné slovo – nástroj je idempotentní), protože co už tam je se pouze ověří, že je v požadovaném stavu, a co tam chybí nebo nesedí se opraví. Stejně tak ale můžete vzít původní image a vybudovat všechno znovu (pokud jde o rychlý security patch, může být fajn to udělat na existujícím VM, ale ve většině případů bych volil cestu vybudovat si vše znova – ostatně takto stávající VM vůbec nemusíte vypínat a omezovat její dostupnost pro uživatele, až bude druhá nahoře funkční, můžete uživatele překlopit nějakým mechanismem – pro jednoduchost zatím řekněme DNS, ale na cloudsvet si povíme podstatně lepší metody).

Tohle si vyzkoušíte v druhé pokročilejší části labu Helion OpenStack.

2.5.4. Zapouzdřené aplikace aneb immutable server

Dotahování image je sice výborné z pohledu opakovatelnosti, dokumentace, přenositelnosti a plné automatizace, ale může trvat dlouho. Pokaždé musíte stáhnout hromady balíčků (někdy je dobrý nápad mít lokální cache), upravit a nakopírovat spousty souborů, instalovat služby. Všechno sice dělá precizní automat, ale zkrátit čas se mu nemusí podařit. Pokud by v předpisu byla chyba, je nutné ji zachytit, a protožo takový deployment už má blízko k Operations týmu, musí v něm být potřebné znalosti. Ze všech těchto důvodů se dá jít ještě dál a připravovat zlaté obrazy, tedy golden image a immutable server.

Myšlenka spočívá v tom, že vyjdete z předchozího scénáře, protože ten vám dává všechny příjemné vlastnosti desired state způsobu práce. Jeho výsledky ovšem přetavíte v hotový obraz – a samozřejmě automatizovaně. Můj

nejoblíbenější nástroj je v tomto ohledu jednoznačně Packer (už brzy si ho vyzkoušíte v cloudsvět Helion OpenStack 2.0 labu). Packer je jednoúčelový nástroj – stejných výsledků dosáhnete i v rámci univerzálnější orchestrace typu HPE Operation Orchestration či jeho open source varianty CloudSlang. Začínáte tím, že Packeru předhodíte výchozí obraz s OS. V předchozím kroku byly nástroje nezávislé na hypervisoru, tady to tak není – nicméně můžete jedním krokem a způsobem vyrobit obrazy pro různé platformy. Do předpisu tedy dáme třeba Debian obraz ve vSphere, OpenStack s KVM a Amazon AMI – tyto obvykle nemusíme vytvářet, existují už hotové. Packer pak každý tento image spustí v příslušném prostředí a provede „dotažení" – shell skript, Ansible, Chef a mnohé další podle vaší chutě. Následně z výsledku udělá nový image a ten připraví ke spuštění. Tak například v OpenStack to funguje tak, že Packer si sám na pozadí vytvoří instanci výchozího image, zajistí spuštění dotahovačů, pak instanci vypne, udělá z ní nový image, který je v OpenStack k dispozici pro spuštění. Totéž platí pro Amazon nebo vSphere a kromě toho Packer umí i čisté QEMU/KVM nebo VirtualBox včetně přípravy Vagrant boxů. A mimochodem dnes už podporuje i Docker kontejner, takže se stává docela dobrou migrační cestou pro ty, co kromě VM zkouší i kontejnery.

Výsledkem je, že provoz získává image s instancí aplikace, kterou stačí spustit v požadovaném počtu. Je dokonce možné v zapouzdření jít tak daleko, že necháte Packer na konci procesu zlikvidovat management přístup do VM, tedy zakázat SSH či RDP. Řešení je velmi spolehlivé, jednoduché pro provozáky a vše se spouští velmi rychle a dají se tak použít vychytávky jako je autoscaling cloud native aplikací. Je tu ale jedna změna – v předchozím případě nám konfigurační nástroj mohl zajistit i aplikační konfiguraci (tedy sdělit VM s aplikací kde má DB a jak se do ní nalogovat). V tomto scénáři něco takového není vhodné – naše zlaté obrazy by neměly mít pevně danou konfiguraci v sobě. Nastavení a objevování nodů by tedy mělo být ideálně mimo vlastní obraz. Na to existují nástroje jako je Etcd, Consul, Zookeper nebo doozerd – ale o těch jindy. Immutable servery jsou perfektní, ale už po vás vyžadují trochu víc znalostí a přemýšlení – ale odměna je sladká.

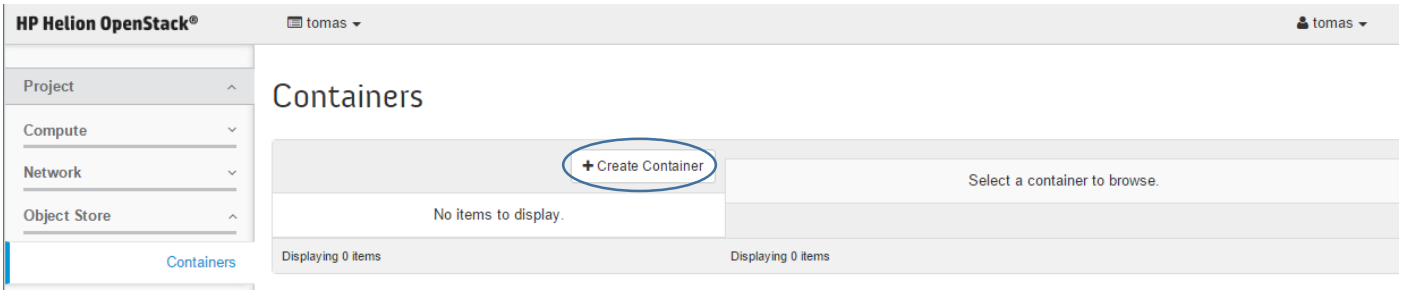
Tohle si vyzkoušíte v druhé pokročilejší části labu Helion OpenStack a některé funkce jako je Etcd nebo Consul v připravovaném labu zaměřeném na moderní vývoj aplikací, kontejnery, Helion Stackato a Helion Development Platform.

2.6. Objektová storage

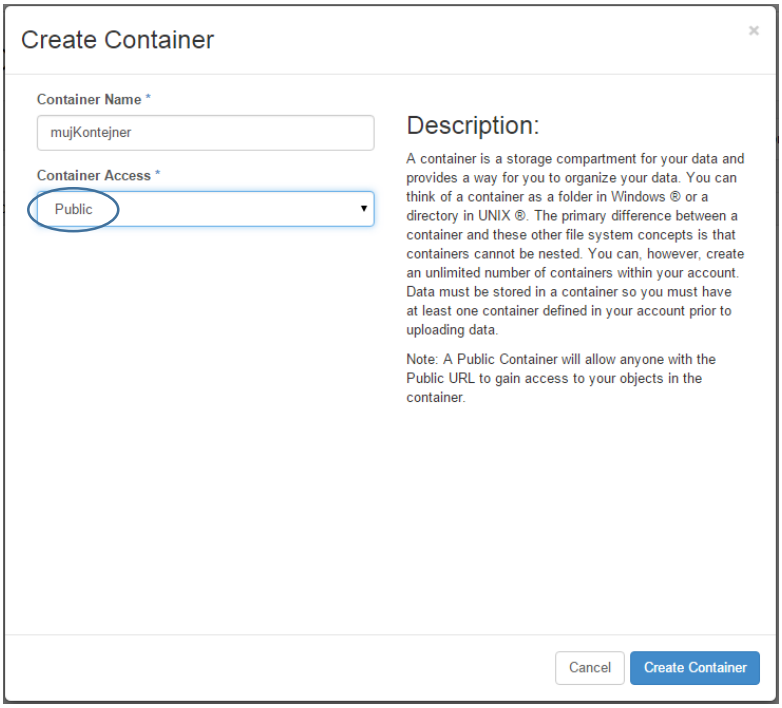
Součástí projektu OpenStack je implementace objektové storage. Běžná instalace používá pouze základní redundantní variantu a je určena především pro interní použití samotného OpenStack. Například veškeré image či zálohy volume se ukládají právě sem. Objektovou storage ale můžete v instalátoru vytvořit i ve scale-out verzi třeba na několik desítek uzlů a vytvořit tak masivní plně redundantní prostředek ukládání objektů. Dokonce můžete nainstalovat Swift bez cloudu (Helion Lifecycle Management podporuje i stand-alone object storage instalaci). Kapacita i výkon je čistě záležitostí počtu nodů, které mohou být jakýkoli server s nějakými disky. Typické nastavení ukládá každý objekt na třech různých místech a v případě havárie nodu se automaticky celé prostředí dosynchronizuje zcela bez výpadku, propadu výkonu nebo jakéhokoli manuálního zásahu. Chcete-li například ukládat velké soubory, diskové obrazy, zálohy nebo filmy, objektová storage může být perfektní volbou. Vězte také, že řešení si nejen můžete sestavit sami s využitím HPE Helion OpenStack, ale existuje i před připravená a odladěná varianta vybraného vhodného hardware, integrace, návazností, OpenStack a finančního modelu (aka per pay use) pod názvem HPE Helion Content Depot.

Práce s objektovou storage je velmi snadná. V našem labu využijeme základní instalace zaměřené na interní použití OpenStackem, ale pokud to nebudete přehánět, můžete si pohrát i s tou. Smyslem je, že uploadnete soubor a získáte jeho URL, pod kterou ho vždy najdete … a to je všechno, velmi jednoduché, extrémně škálovatelné.

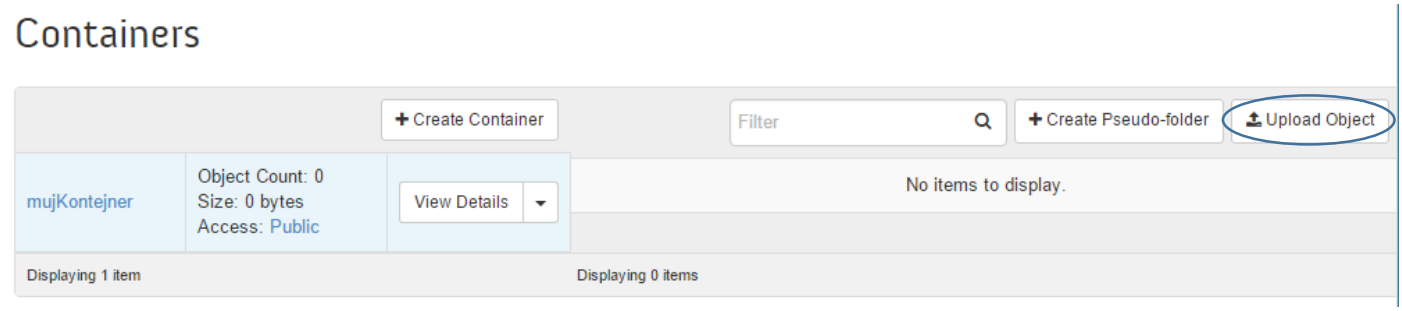
Jděte do záložky Object Store, Containers a klikněte na Create Container



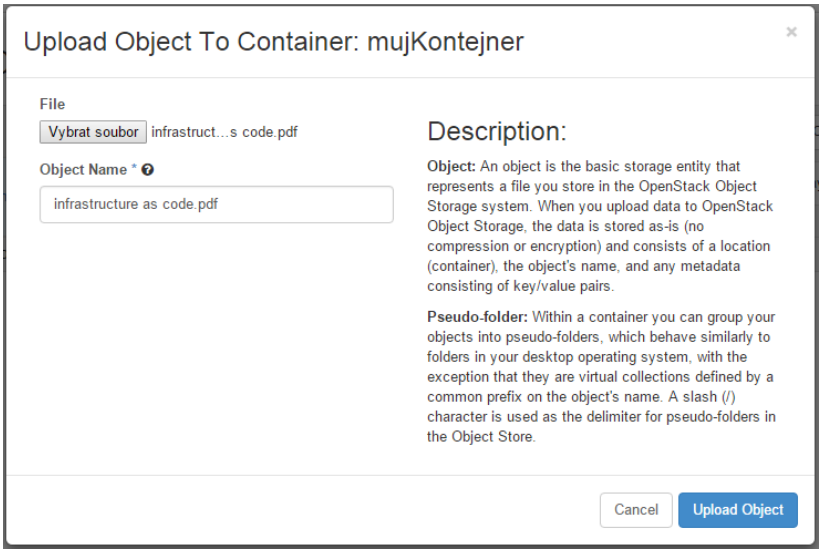
Dejte mu jméno a určete, jestli je viditelný jen pro váš projekt, nebo běžně dostupný a klikněte na Create Container



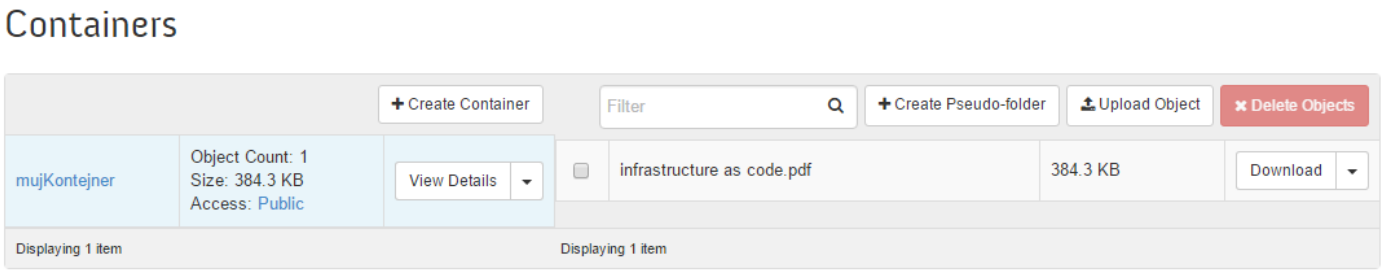
Velmi brzo bude hotovo a můžete kliknout na Upload Object



Zvolte nějaký soubor (mějte prosím ohledy na zdroje v labu, tedy něco rozumně velkého) a nahrajte do objektové storage kliknutím na Upload Object



Po úspěšném nahrání uvidíte objekt (v našem případě application/pdf typ) jako dostupný



Klikněte na Download a stáhnete si svůj objekt.

Daleko víc si se Swift objektovou storage pohrajete v druhé pokročilejší části lab guide, stáhněte si z www.cloudsvet.cz. Co dalšího umí OpenStack Swift?

- Samozřejmě přirozené vlastnosti scale-out object storage - velká horizontální škálovatelnost, použití komoditního hardware, vstup a výstup HTTP protokolem atd.
- Erasure-coding (v budoucí verzi Helion OpenStack) umožňuje místo ukládání celých objektů na nodu a jejich replikaci k několika dalším využít rozsekání velkého objektu na dílky, ty rozprostřít po nodech a přidat k nim kontrolní součty. Za cenu snížení výkonu a zvýšení latence lze snížit cenu za úložný prostor
- Řízení přístupu k vašim objektům (kdo smí?)
- Streamované vkládání, tedy schopnost vkládat objekt aniž by byla dopředu známa jeho délka
- Verzování zápisů, takže URL vede vždy na nejnovější verzi, ale systém drží a umožní získat ty předchozí
- Přímý upload z klienta přes HTTP POST, tedy webová aplikace dovoluje prohlížeči vzít lokální soubor a poslat přímo do object storage, nemusí se jít přes web server
- Časově omezené URL (dočasné odkazy)
- Časově omezené objekty (sami se odmažou)
- Velmi flexibilní storage politiky - dá se například podle uživatele nebo typu dat preferovat SSD, omezit fyzikální umístění dat na určitý region nebo naopak vynutit celoplanetární repliku
- Multi-regionální nastavení (v budoucí verzi Helion OpenStack) s read/write affinity (lokální čtení a zápis)

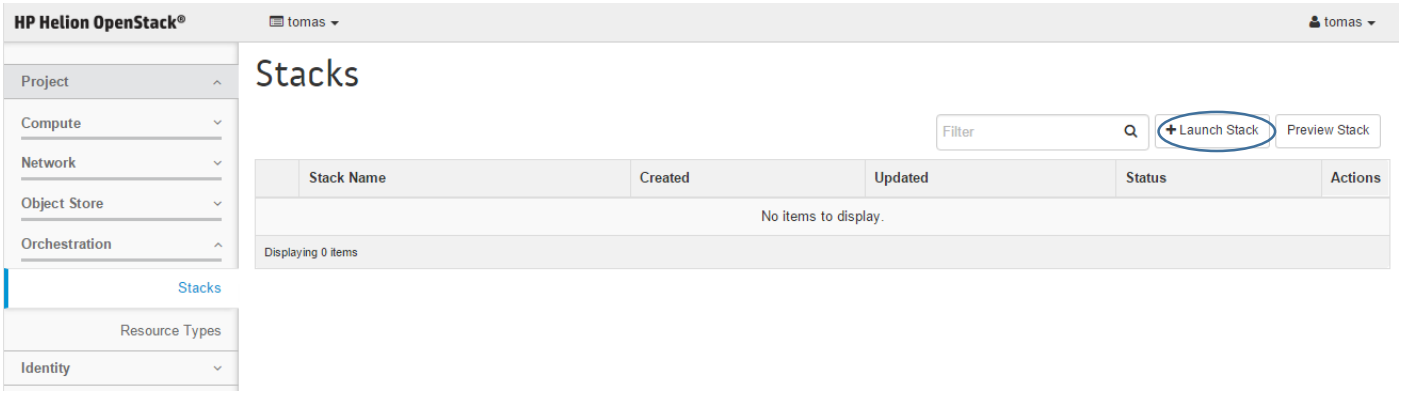
2.7. Infrastrukturní šablony

OpenStack nabízí velmi mocný nástroj pro tvorbu šablon a automatizaci celých prostředí. Tak jak znáte například šablonu pro VM, která zahrnuje operační systém, velikost RAM apod., šablona v OpenStack zahrnuje celou infrastrukturu. Tedy VM, privátní sítě a subnety, Floating IP, per-VM firewally, load-balancery, firewally, VPN přístupy, storage volume, diskové obrazy, přístupové SSH klíče či autokonfigurační skripty. Kromě toho umí

orchestrátor provádět auto-scale (automatické přidávání a ubírání VM dle zátěže) nebo automatizovat instalaci aplikací (instalační skripty, vyvolání Ansible nebo Chef apod.).

Jste líní opisovat? Všechny šablony najdete také na labServeru v adresáři heat-sablony – můžete si je vypsat (cat nebo more) a přes schránku kopírovat do GUI.

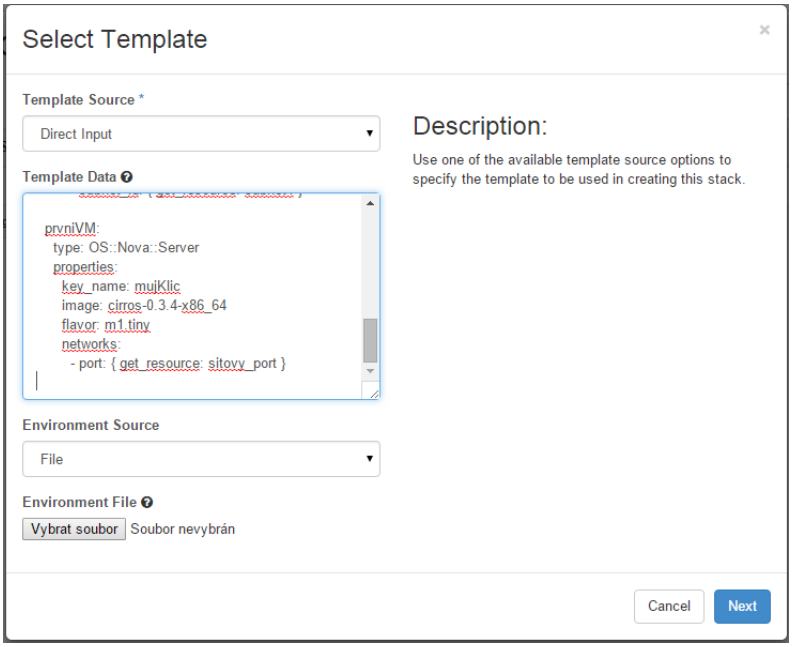
Připojte se do GUI a jděte na záložku Orchestration, Stacks a klikněte na Launch Stack



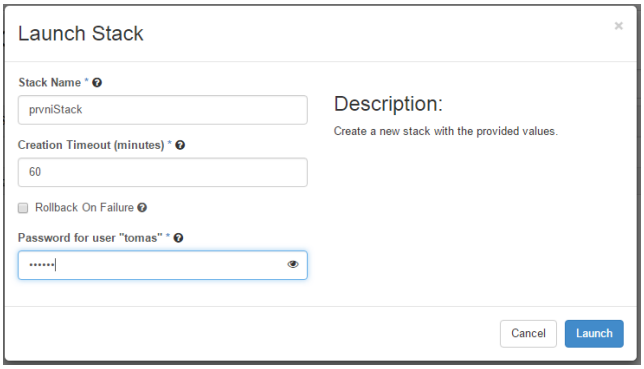
Šablonu můžete zadat přímo do okna, nebo ji nahrajete ve formě souboru. Obsah šablony používá YAML strukturu a pro začátek bude vypadat takto:

```
heat_template_version: 2013-05-23

description: Nase prvni sablona

resources:
  sit1:
    type: OS::Neutron::Net
    properties:
      name: stackSit1

  subnet1:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: sit1 }
      cidr: 192.168.10.0/24
      allocation_pools:
        - start: 192.168.10.100
          end: 192.168.10.200

  sitovy_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_resource: sit1 }
      fixed_ips:
        - subnet_id: { get_resource: subnet1 }

  prvniVM:
    type: OS::Nova::Server
    properties:
      key_name: mujKlic
      image: cirros-0.3.4-x86_64
      flavor: m1.tiny
      networks:
        - port: { get_resource: sitovy_port }
```

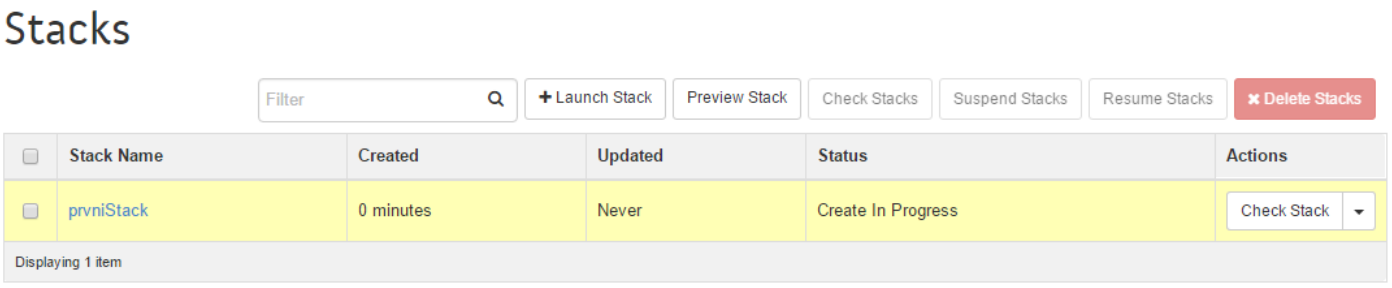
Vložte do okna (nebo přes soubor) a klikněte na Next

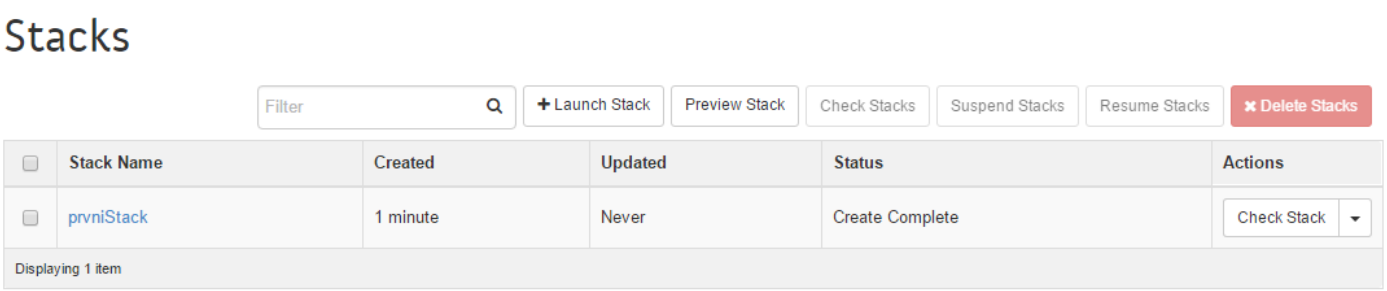


Použijte nějaké jméno, zadejte heslo a klikněte na Launch

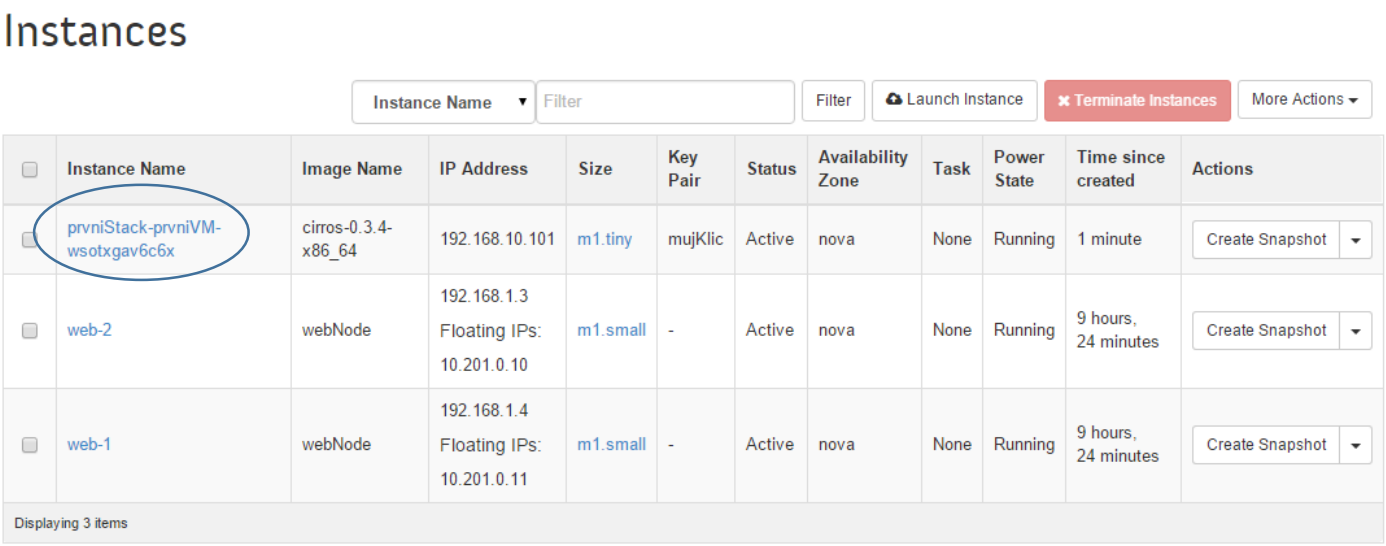


Počkejte až všechno doběhne

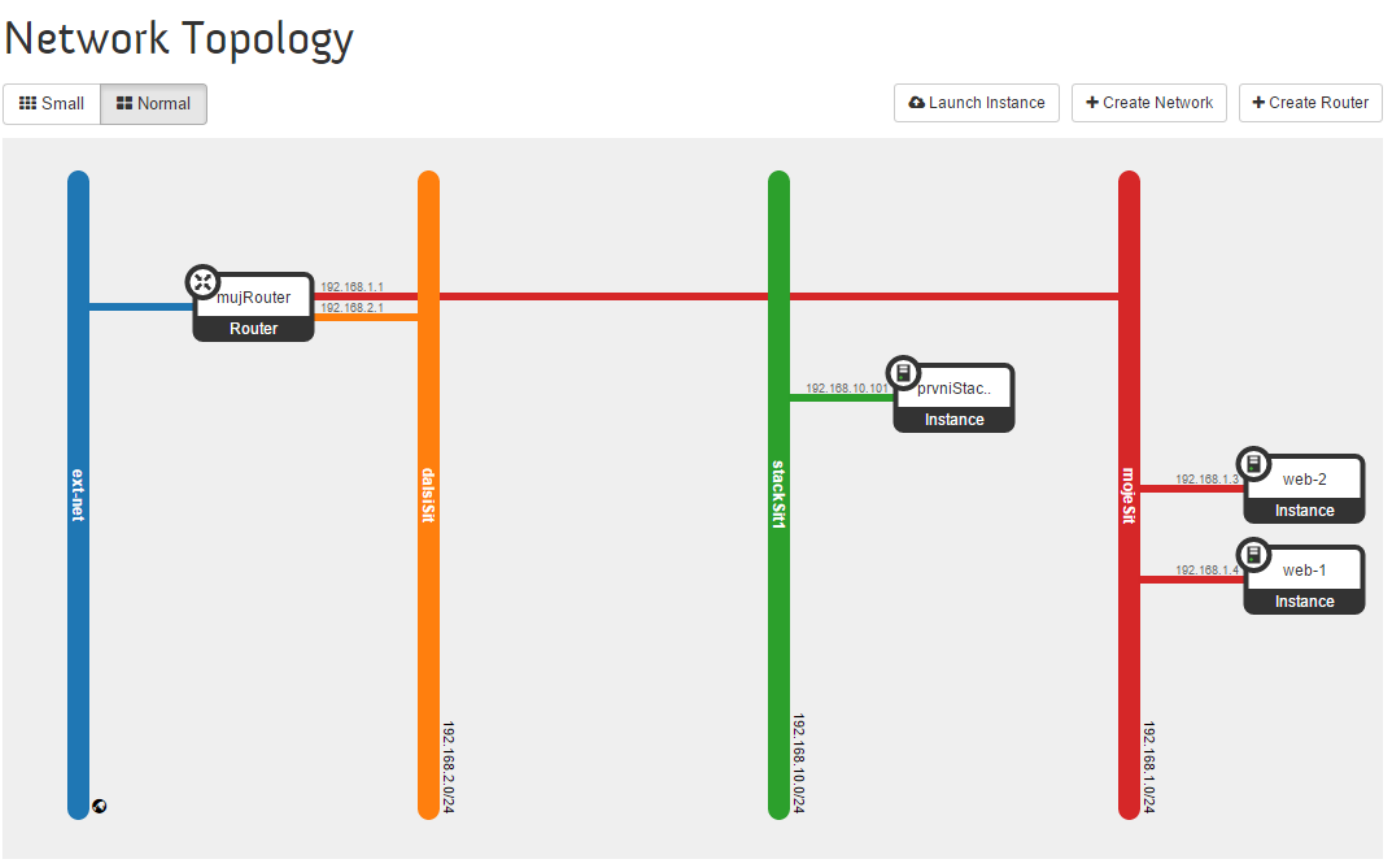




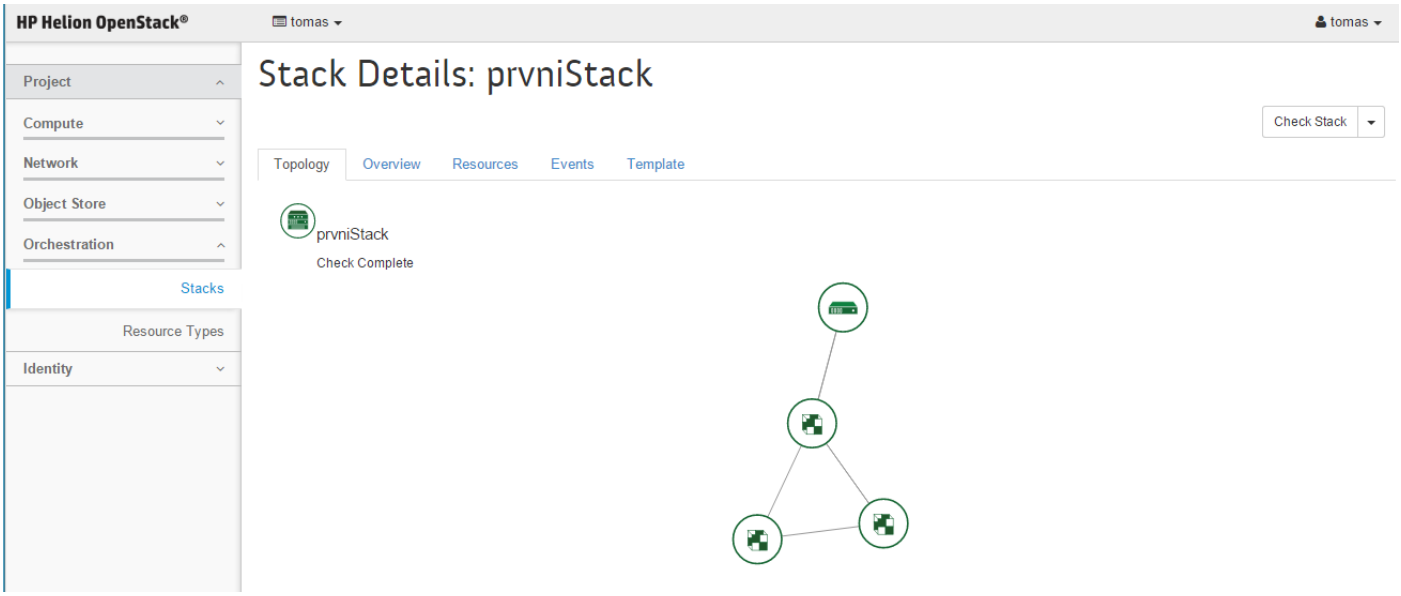
Koukněte se do instancí – je tam něco nového?



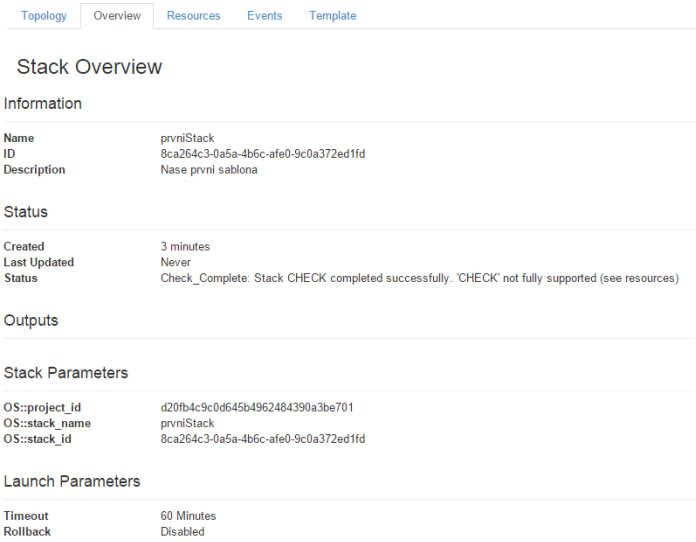
A co v síťové topologii?



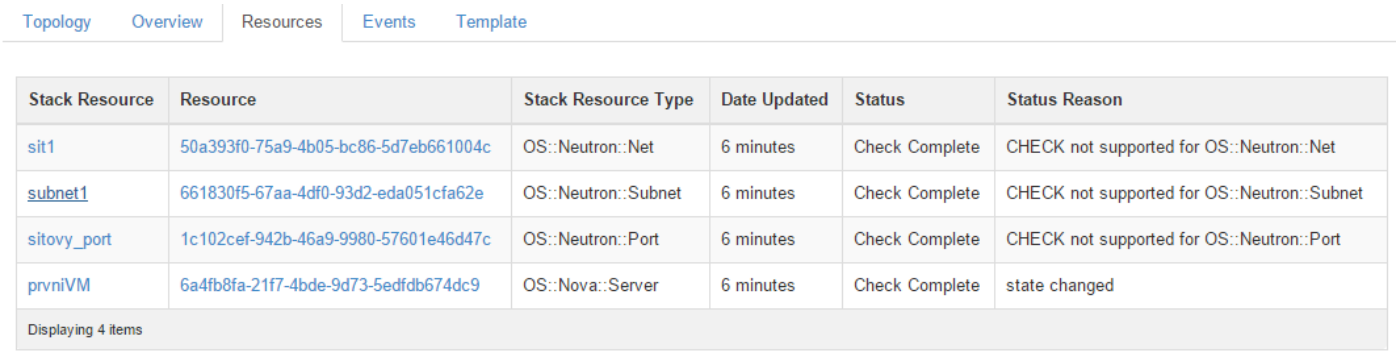
Rozklikněte si „stack“, tedy aplikovanou šablonu. Podíváme se, co se o ní můžeme dozvědět. Na první záložce máte vizualizaci návazností. Pro jednoduché šablony jako je tato může být dobrou pomůckou (pokud se do ni podíváte při průběhu vytváření uvidíte, jak vám jednotlivé zdroje zelenají tak, jak je engine vytváří).



V přehledu dostanete základní informace.



Velmi užitečná je záložka, kde jsou vidět zdroje, které v rámci šablony vznikly.



Stack Resource	Resource	Stack Resource Type	Date Updated	Status	Status Reason
sit1	50a393f0-75a9-4b05-bc86-5d7eb661004c	OS::Neutron::Net	6 minutes	Check Complete	CHECK not supported for OS::Neutron::Net
subnet1	661830f5-67aa-4df0-93d2-eda051cfa62e	OS::Neutron::Subnet	6 minutes	Check Complete	CHECK not supported for OS::Neutron::Subnet
sitovy_port	1c102cef-942b-46a9-9980-57601e46d47c	OS::Neutron::Port	6 minutes	Check Complete	CHECK not supported for OS::Neutron::Port
prvniVM	6a4fb8fa-21f7-4bde-9d73-5edfdb674dc9	OS::Nova::Server	6 minutes	Check Complete	state changed

Displaying 4 items

Co se všechno muselo stát, aby se šablona aplikovala?

Topology | Overview | Resources | Events | Template

Stack Resource	Resource	Time Since Event	Status	Status Reason
prvniStack		4 minutes	Check Complete	Stack CHECK completed successfully. 'CHECK' not fully supported (see resources)
prvniStack		4 minutes	Check Complete	Stack CHECK completed successfully
prvniVM		4 minutes	Check Complete	state changed
prvniVM		4 minutes	Check In Progress	state changed
sitovy_port		4 minutes	Check Complete	CHECK not supported for OS::Neutron::Port
subnet1		4 minutes	Check Complete	CHECK not supported for OS::Neutron::Subnet
sit1		4 minutes	Check Complete	CHECK not supported for OS::Neutron::Net
prvniStack		4 minutes	Check In Progress	Stack CHECK started
prvniStack		7 minutes	Create Complete	Stack CREATE completed successfully
prvniVM		7 minutes	Create Complete	state changed
prvniVM	-	7 minutes	Create In Progress	state changed
sitovy_port		7 minutes	Create Complete	state changed
sitovy_port	-	7 minutes	Create In Progress	state changed
subnet1		7 minutes	Create Complete	state changed
subnet1	-	7 minutes	Create In Progress	state changed
sit1		7 minutes	Create Complete	state changed
sit1	-	7 minutes	Create In Progress	state changed
prvniStack		7 minutes	Create In Progress	Stack CREATE started

Displaying 18 items

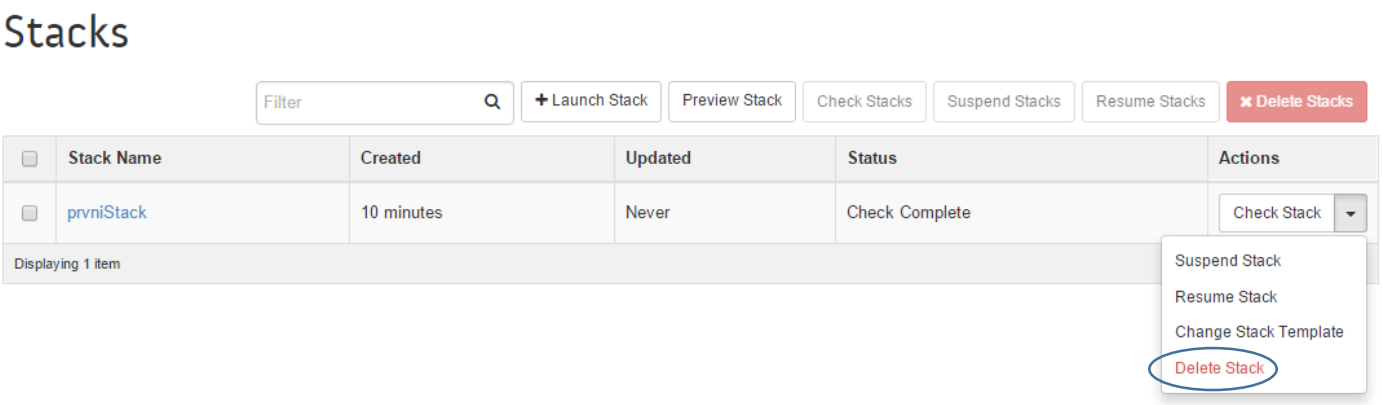
A jakou šablonu to vlastně máme?

Topology | Overview | Resources | Events | Template

Stack Template

```
description: Nase prvni sablona
heat_template_version: '2013-05-23'
resources:
  prvniVM:
    properties:
      flavor: m1.tiny
      image: cirros-0.3.4-x86_64
      key_name: mujKlic
      networks:
      - port: {get_resource: sitovy_port}
    type: OS::Nova::Server
  sit1:
    properties: {name: stackSit1}
    type: OS::Neutron::Net
  sitovy_port:
    properties:
      fixed_ips:
      - subnet_id: {get_resource: subnet1}
      network_id: {get_resource: sit1}
    type: OS::Neutron::Port
  subnet1:
    properties:
      allocation_pools:
      - {end: 192.168.10.200, start: 192.168.10.100}
      cidr: 192.168.10.0/24
      network_id: {get_resource: sit1}
    type: OS::Neutron::Subnet
```

Už celý stack nechceme? Nemusíme mazat jednotlivé zdroje, ale zlikvidujte to rovnou celé.

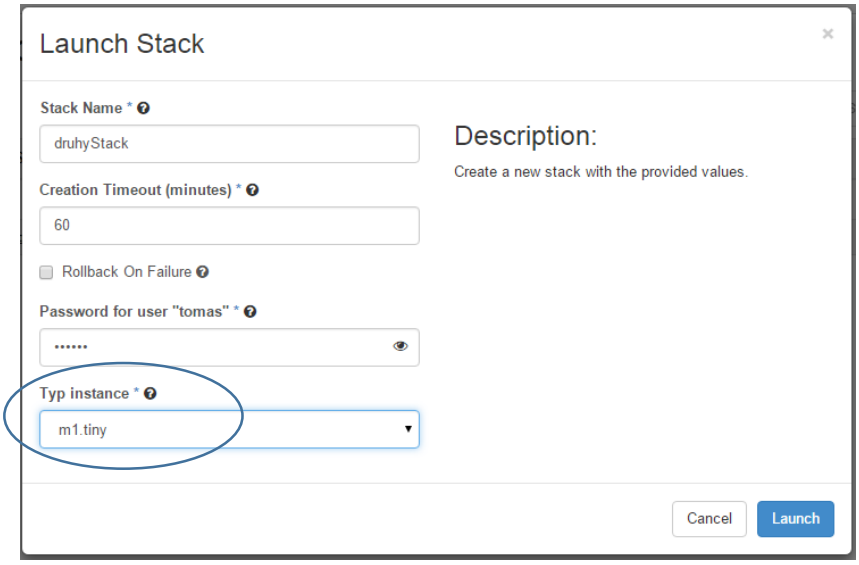


Šablona může být interaktivnější a některé parametry si nechat zadat uživatelem až v okamžiku jejího deploymentu. Následující příklad dává možnost definovat typ (velikost/flavor) serveru, ale chceme dát na výběr jen ze dvou možností. Dále využijeme možnosti formovat výstup, ve kterém předáme dál některé informace z instalace šablony, v našem případě jakou IP adresu VM dostala (teď se vám to bude zdát zbytečné, ale až se dostaneme k možnosti navázat šablonu na další operace v Ansiblu nebo jinému nástroji, uvidíte, jak je předání parametrů dál důležité). Použijte tuto šablonu:

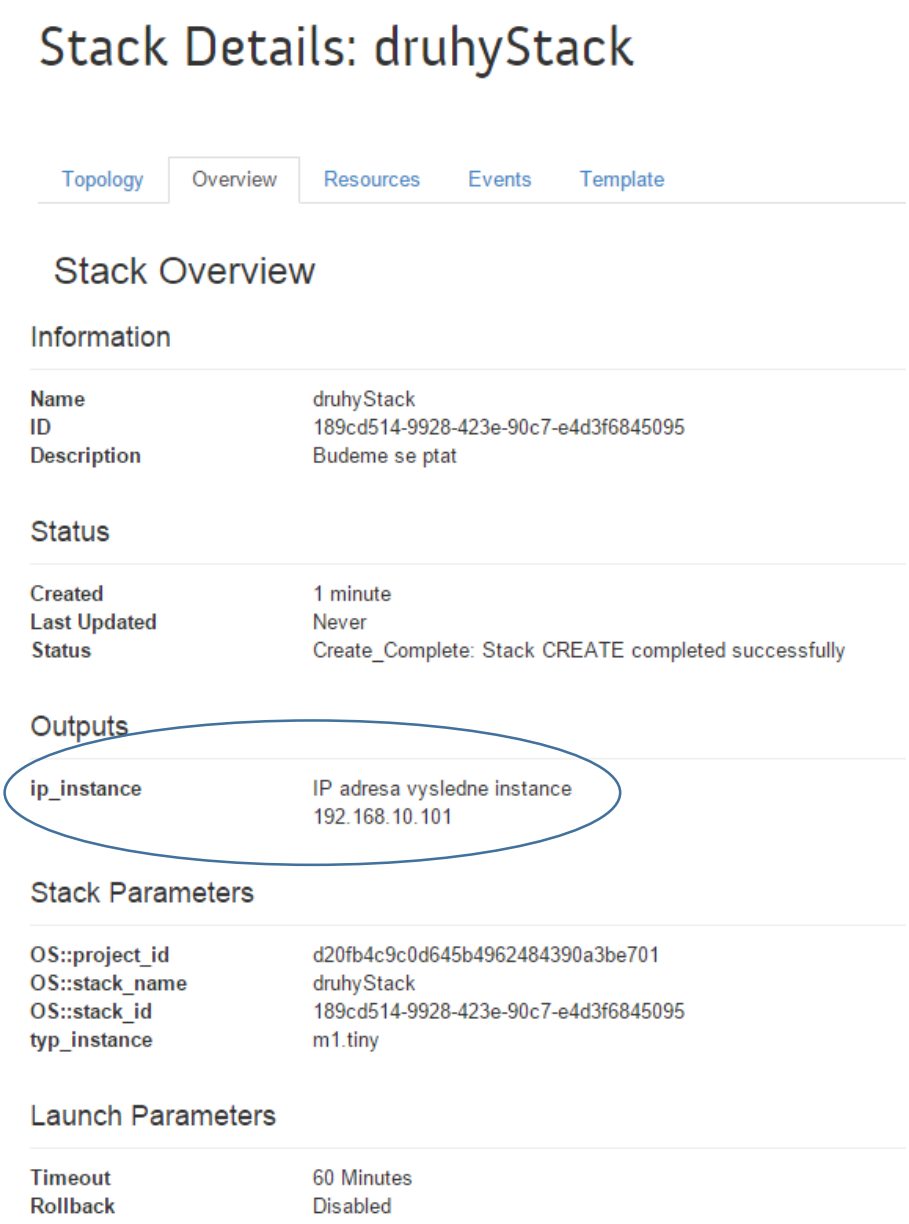
```
heat_template_version: 2013-05-23

description: Budeme se ptat

parameters:
  typ_instance:
    type: string
    label: Typ instance
    description: Vyberte si flavor m1.tiny nebo m1.small
    constraints:
      - allowed_values: [ m1.tiny, m1.small ]
        description: Pripustne hodnoty jsou m1.tiny nebo m1.small

resources:
  sit1:
    type: OS::Neutron::Net
    properties:
      name: stackSit1

  subnet1:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: sit1 }
      cidr: 192.168.10.0/24
      allocation_pools:
        - start: 192.168.10.100
          end: 192.168.10.200

  sitovy_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_resource: sit1 }
      fixed_ips:
        - subnet_id: { get_resource: subnet1 }

  prvniVM:
    type: OS::Nova::Server
    properties:
      key_name: mujKlic
      image: cirros-0.3.4-x86_64
      flavor: { get_param: typ_instance }
      networks:
        - port: { get_resource: sitovy_port }
```

```
outputs:
  ip_instance:
    description: IP adresa vysledne instance
    value: { get_attr: [prvniVM, first_address] }
```

Založte tento Stack a klikněte na Next. Všimněte si, že GUI po nás chce doplnění našich parametrů.



Po nastartování stacku se podívejte do Overview – najdete tam náš požadovaný výstup.

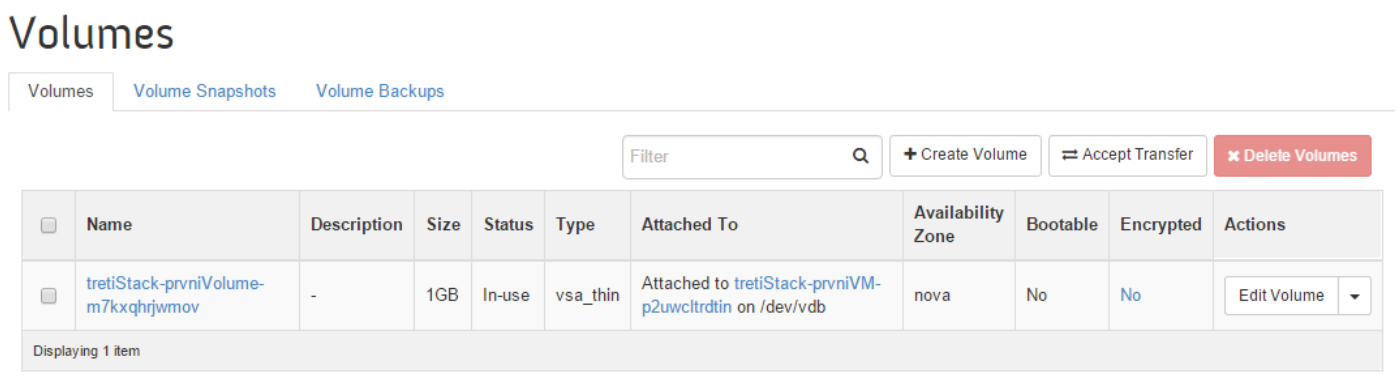


Zrušte tento stack, zkusíme nějake zas o kousek složitější. Co pro naší instanci udělat datový Volume ve storage a připojit? Takhle bude šablona vypadat – spusťte ji.

```
heat_template_version: 2013-05-23

description: Zkusime i volume

parameters:
  typ_instance:
    type: string
    label: Typ instance
    description: Vyberte si flavor m1.tiny nebo m1.small
    constraints:
      - allowed_values: [ m1.tiny, m1.small ]
        description: Pripustne hodnoty jsou m1.tiny nebo m1.small

resources:
  sit1:
    type: OS::Neutron::Net
    properties:
      name: stackSit1

  subnet1:
    type: OS::Neutron::Subnet
    properties:
```

```
      network_id: { get_resource: sit1 }
      cidr: 192.168.10.0/24
      allocation_pools:
        - start: 192.168.10.100
          end: 192.168.10.200

  sitovy_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_resource: sit1 }
      fixed_ips:
        - subnet_id: { get_resource: subnet1 }

  prvniVM:
    type: OS::Nova::Server
    properties:
      key_name: mujKlic
      image: cirros-0.3.4-x86_64
      flavor: { get_param: typ_instance }
      networks:
        - port: { get_resource: sitovy_port }

  prvniVolume:
    type: OS::Cinder::Volume
    properties:
      size: 1

  vol1_att:
    type: OS::Cinder::VolumeAttachment
    properties:
      instance_uuid: { get_resource: prvniVM }
      volume_id: { get_resource: prvniVolume }
      mountpoint: /dev/vdb

outputs:
  ip_instance:
    description: IP adresa vysledne instance
    value: { get_attr: [prvniVM, first_address] }
```

Přesvěčte se, že se Volume vytvořil a připojil.



Smažte tento stack, zkusíme ho dále obohacovat.

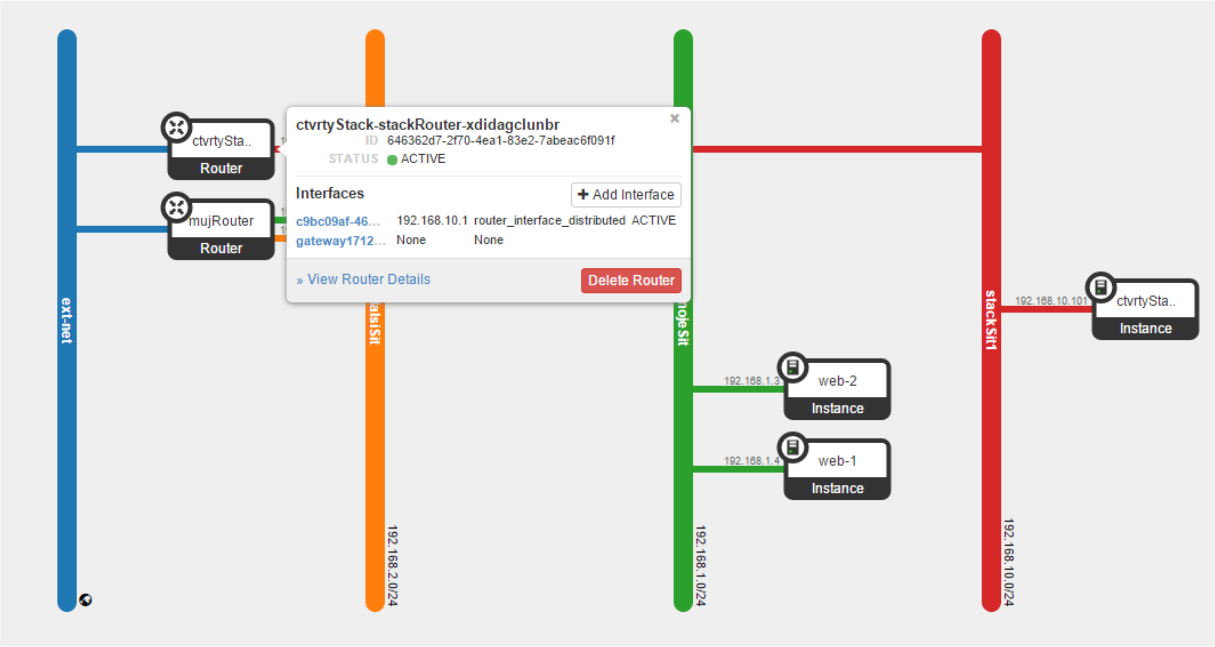
Zatím nemáme vytvořen router ani přiřazenu Floating IP. Co kdybychom teď vytvořili nový router a síť do něj připojili a dali naší prvniVM Floating IP adresu?

Tohle je náš čtvrtý stack, zkuste ho:

```
heat_template_version: 2013-05-23

description: Ted i se sitarinou

parameters:
  typ_instance:
    type: string
    label: Typ instance
    description: Vyberte si flavor m1.tiny nebo m1.small
    constraints:
```

```
      - allowed_values: [ m1.tiny, m1.small ]
        description: Pripustne hodnoty jsou m1.tiny nebo m1.small

resources:
  sit1:
    type: OS::Neutron::Net
    properties:
      name: stackSit1

  subnet1:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: sit1 }
      cidr: 192.168.10.0/24
      allocation_pools:
        - start: 192.168.10.100
          end: 192.168.10.200

  stackRouter:
    type: OS::Neutron::Router
    properties:
      external_gateway_info:
        network: ext-net

  stackRouter_interface:
    type: OS::Neutron::RouterInterface
    properties:
      router_id: { get_resource: stackRouter }
      subnet_id: { get_resource: subnet1 }

  sitovy_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_resource: sit1 }
      fixed_ips:
        - subnet_id: { get_resource: subnet1 }

  prvniVM_floating_ip:
    type: OS::Neutron::FloatingIP
    properties:
      floating_network: ext-net
      port_id: { get_resource: sitovy_port }

  prvniVM:
    type: OS::Nova::Server
    properties:
      key_name: mujKlic
      image: cirros-0.3.4-x86_64
      flavor: { get_param: typ_instance }
      networks:
        - port: { get_resource: sitovy_port }

  prvniVolume:
    type: OS::Cinder::Volume
    properties:
      size: 1

  vol1_att:
    type: OS::Cinder::VolumeAttachment
    properties:
      instance_uuid: { get_resource: prvniVM }
      volume_id: { get_resource: prvniVolume }
      mountpoint: /dev/vdb

outputs:
  ip_instance:
    description: IP adresa vysledne instance
    value: { get_attr: [prvniVM, first_address] }
  float_ip_instance:
    description: Venkovni IP adresa vysledne instance
    value: { get_attr: [prvniVM_floating_ip, floating_ip_address] }
```

Ověřte výsledek v topologii a výstupech.



Outputs

float_ip_instance	Venkovni IP adresa vysledne instance 10.201.0.14
ip_instance	IP adresa vysledne instance 192.168.10.101

Smažte i náš čtvrtý stack.

Jdeme do finále (alespoň co se tohoto labu týče – k orchestračním šablonám se vrátíme i v druhém pokročilém díle lab guide Helion OpenStack). Vytvořme teď další síť s web servery, tyto spustíme, přidáme k nim Security group povolující web provoz, vytvoříme load-balancer, který bude rozdělovat jejich zátěž a balanceru dáme Floating IP. Ve verzi Helion OpenStack 2.0 v našem labu vám ještě šablona spustit nepůjde (nepodporuje zatím LBaaSv2, pouze starší verzi) – nicméně v pozdějších aktualizacích už to možné bude.

```
heat_template_version: 2013-05-23

description: Webova farma

parameters:
  typ_instance:
    type: string
    label: Typ instance
    description: Vyberte si flavor m1.tiny nebo m1.small
    constraints:
      - allowed_values: [ m1.tiny, m1.small ]
        description: Pripustne hodnoty jsou m1.tiny nebo m1.small

resources:
  sit1:
    type: OS::Neutron::Net
    properties:
      name: stackSit1

  subnet1:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: sit1 }
      cidr: 192.168.10.0/24
      allocation_pools:
        - start: 192.168.10.100
          end: 192.168.10.200

  stackRouter:
    type: OS::Neutron::Router
```

```
  properties:
    external_gateway_info:
      network: ext-net

stackRouter_interface:
  type: OS::Neutron::RouterInterface
  properties:
    router_id: { get_resource: stackRouter }
    subnet_id: { get_resource: subnet1 }

sitovy_port:
  type: OS::Neutron::Port
  properties:
    network_id: { get_resource: sit1 }
    fixed_ips:
      - subnet_id: { get_resource: subnet1 }

prvniVM_floating_ip:
  type: OS::Neutron::FloatingIP
  properties:
    floating_network: ext-net
    port_id: { get_resource: sitovy_port }

prvniVM:
  type: OS::Nova::Server
  properties:
    key_name: mujKlic
    image: cirros-0.3.4-x86_64
    flavor: { get_param: typ_instance }
    networks:
      - port: { get_resource: sitovy_port }

prvniVolume:
  type: OS::Cinder::Volume
  properties:
    size: 1

vol1_att:
  type: OS::Cinder::VolumeAttachment
  properties:
    instance_uuid: { get_resource: prvniVM }
    volume_id: { get_resource: prvniVolume }
    mountpoint: /dev/vdb

web_security_group:
  type: OS::Neutron::SecurityGroup
  properties:
    description: Webovy pristup
    name: web-SG
    rules:
      - remote_ip_prefix: 0.0.0.0/0
        protocol: tcp
        port_range_min: 80
        port_range_max: 80
      - remote_ip_prefix: 0.0.0.0/0
        protocol: icmp

web_sit:
  type: OS::Neutron::Net
  properties:
    name: webSit

web_subnet:
  type: OS::Neutron::Subnet
  properties:
    network_id: { get_resource: web_sit }
    cidr: 192.168.20.0/24
    allocation_pools:
      - start: 192.168.20.100
        end: 192.168.20.200

stackRouter_interface2:
  type: OS::Neutron::RouterInterface
  properties:
    router_id: { get_resource: stackRouter }
```

```
    subnet_id: { get_resource: web_subnet }

webVM1_interface:
  type: OS::Neutron::Port
  properties:
    network_id: { get_resource: web_sit }
    fixed_ips:
      - subnet_id: { get_resource: web_subnet }
    security_groups: [{ get_resource: web_security_group }]

webVM2_interface:
  type: OS::Neutron::Port
  properties:
    network_id: { get_resource: web_sit }
    fixed_ips:
      - subnet_id: { get_resource: web_subnet }
    security_groups: [{ get_resource: web_security_group }]

webVM1:
  type: OS::Nova::Server
  properties:
    key_name: mujKlic
    image: webNode
    flavor: m1.small
    networks:
      - port: { get_resource: webVM1_interface }

webVM2:
  type: OS::Nova::Server
  properties:
    key_name: mujKlic
    image: webNode
    flavor: m1.small
    networks:
      - port: { get_resource: webVM2_interface }

health:
  type: OS::Neutron::HealthMonitor
  properties:
    type: HTTP
    delay: 1
    max_retries: 1
    timeout: 1

pool:
  type: OS::Neutron::Pool
  properties:
    lb_method: ROUND_ROBIN
    protocol: HTTP
    subnet_id: {get_resource: web_subnet}
    monitors: [{get_resource: health}]
    vip:
      protocol_port: 80

lb_member_webVM1:
  type: OS::Neutron::PoolMember
  properties:
    pool_id: {get_resource: pool}
    address: {get_attr: [webVM1, first_address]}
    protocol_port: 80

lb_member_webVM2:
  type: OS::Neutron::PoolMember
  properties:
    pool_id: {get_resource: pool}
    address: {get_attr: [webVM2, first_address]}
    protocol_port: 80

balancer:
  type: OS::Neutron::LoadBalancer
  properties:
    protocol_port: 80
    pool_id: {get_resource: pool}

balancer_floating:
```

```
    type: OS::Neutron::FloatingIP
    properties:
      floating_network: ext-net
      port_id: {get_attr: [pool, vip, port_id]}

outputs:
  ip_instance:
    description: IP adresa vysledne instance
    value: { get_attr: [prvniVM, first_address] }
  float_ip_instance:
    description: Venkovni IP adresa vysledne instance
    value: { get_attr: [prvniVM_floating_ip, floating_ip_address] }
  float_ip_lb:
    description: Virtualni IP balanceru
    value: { get_attr: [balancer_floating, floating_ip_address] }
```

Podařilo se nám vytvořit už docela komplikovanou šablonu a na tomto místě v základním lab guide skončíme. Pokračovat budeme v druhém pokročilejším labu, kde budeme iniciovat i instalaci a konfiguraci VM v rámci šablony.

Co tedy OpenStack orchestrace (komponenta Heat) umí?

- Už dříve jsme si vyzkoušeli, že šablona může zahrnovat mnoho vstupních parametrů, takže dokument může být poměrně univerzální a při použití šablony si ji uživatel doladí dle reálných potřeb
- Šablona je v jednoduchém textovém formátu a sama o sobě může sloužit jako dokumentace (Infrastructure as code)
- Jednoduchý textový formát můžete prohnat libovolným versioning systémem jako je třeba Git a snadno zjistíte rozdíly mezi verzemi šablony a udržujete přehled o tom kdo jaké změny provedl
- Kromě vyzkoušených věcí můžete používat konfigurační skripty – k tomu se ještě později dostaneme
- Výsledkem může být předání parametrů do dalšího systému, například něco, co zajistí konfiguraci OS a instalaci aplikace (Ansible, Chef, ...)
- Heat šablony podporují automatické škálování

2.8. Ukončení této části labu

Pohráli jsme si, pojďme teď uvolnit alokované zdroje, především běžící instance a volume. Proveďte prosím následující kroky:

1. Terminujte běžící instance
2. Smažte volume backupy a snapshoty
3. Smažte volume
4. Pokud jste vložili něco do objektové storage, zrušte to včetně kontejneru
5. Klikněte na router a zrušte jeho interfacy
6. Zrušte router
7. Zrušte vámi vytvořené sítě

Uff, nebylo to lepší dělat nějakým chytřejším skriptem? No jasně a nejen proto pokračujte do druhé části labu, tentokrát pro pokročilé. Stahujte na www.cloudsvet.cz.

3. Shrnutí a závěr

Právě jste dokončili první část labu zaměřenou na uživatelskou zkušenost v HPE Helion OpenStack. Vyzkoušeli jste si na vlastní kůži jak je možné orchestrovat celé prostředí a získávat infrastrukturu na zavolání, tedy IaaS. Ukázali jsme si jak pracovat s compute, storage a networking zdroji, jak vytvářet a používat image a snapshoty, jak orchestrovat ovládání hypervisorů, StoreVirtual VSA a dalších komponent.

Pokud se považujete za poučený sales, technický sales nebo presales na ne-automatizační oblasti, je pro vás úroveň detailu v tomto labu možná tak akorát. Pro solution architekty, přemýšlivce a automatizátory je zaměřena druhá část labu, kde ochutnáte mnoho dalších moderních technologií a postupů:

- Zrychlíte svojí práci díky použití standardního OpenStack CLI
- Naučíte se pracovat s OpenStack API a psát vlastní orchestrační aplikace
- Naučíte se instalovat OS balíčky a aplikace s Ansible, Chef a HPE Server Automation
- Budeme budovat immutable server s Packer nejen pro OpenStack

V plánu máme lab guidy i na další oblasti jako jsou:

- Orchestrace celého prostředí, spojení nového a tradiční IT a vytváření servisního katalogu s HP Cloud Service Automation
- Open source orchestrátor CloudSlang a jeho komerční varianta (HPE Operation Orchestration)
- Moderní vývoj aplikací v PaaS s HPE Helion Development Platform a Stackato
    - Kontejnery pod kapotou PaaS
    - Databáze jako služba
    - Message-queue jako služba
    - CI/CD integrace s Helion Cloud Engine

Pro implementátory pak připravujeme ještě další část zaměřenou na instalaci a administraci automatizovaného prostředí.

Gratuluji k absolvování této procházky světem automatizace s HPE Helion OpenStack ! Mnoho zdaru v praxi a stáhněte si další díl tohoto lab guide pro pokročilé.

Tomáš Kubica, Enterprise architect

www.cloudsvet.cz

4. Další zdroje

Blog o novém IT a nejnovější verze těchto lab guide:

www.cloudsvet.cz

Helion dokumentace je webová a často aktulizovaná, ideální primární zdroj informací:

http://docs.hpcloud.com/

HP nadstavby jako je Cloud Service Automation, Server Automation nebo Executive Scorecard:

http://www.hpe.com/software

Open source projekty:

http://www.openstack.org/

http://cloudfoundry.org/