
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV INTELIGENTNÍCH SYSTÉMŮ**
**DEPARTMENT OF INTELLIGENT SYSTEMS**


# VYUŽITÍ DISKOVÝCH OBRAZŮ V PROJEKTU BEAKER PRO PLATFORMU OPENSTACK

**IMAGE-BASED PROVISIONING FOR BEAKER USING OPENSTACK**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** **PETER HOSTAČNÝ**
**AUTHOR**

**VEDOUCÍ PRÁCE** **Ing. ALEŠ SMRČKA, Ph.D.**
**SUPERVISOR**



**BRNO 2017**

*Zadání bakalářské práce/19149/2016/xhosta03*

**Vysoké učení technické v Brně - Fakulta informačních technologií**

Ústav inteligentních systémů Akademický rok 2016/2017

# Zadání bakalářské práce

Řešitel: **Hostačný Peter**
Obor: Informační technologie
Téma: **Využití diskových obrazů v projektu Beaker pro platformu OpenStack**
**Image-Based Provisioning for Beaker Using OpenStack**
Kategorie: Operační systémy

Pokyny:

1. Nastudujte projekt Beaker pro integrační testování systémů. Seznamte se s platformou OpenStack a s automatizací administrace cloudových služeb.
2. Analyzujte požadavky pro automatizovanou přípravu testovacího systému pomocí projektu Beaker s využitím cloudových služeb. Navrhněte řešení automatizované přípravy testovacího prostředí v cloudové platformě OpenStack založené na klonování diskových obrazů.
3. Implementujte vaše řešení jako rozšíření projektu Beaker.
4. Diskutujte řešení s vývojovým týmem projektu Beaker. Demonstrujte správnou funkcionalitu řešení na netriviálních případech použití.

Literatura:

- Dokumentace k projektu Beaker. https://beaker-project.org/docs/
- Dokumentace k platformě OpenStack. http://docs.openstack.org/
- Kolektiv autorů komunity OpenStack. Adding Speed and Agility to Virtualized Infrastructure with OpenStack. 2015. Dostupné na: http://www.openstack.org/assets/pdf-downloads/virtualization-Integration-whitepaper-2015.pdf

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Smrčka Aleš, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
*vedoucí ústavu*

# Abstrakt

Integračné testovanie prebieha v projekte Beaker vždy v čistom prostredí, na čerstvo nainštalovanom operačnom systéme. Inštalácia zaberá veľké percento z doby prípravy testovacieho stroja. Táto práca sa snaží o urýchlenie daného procesu prípravy pomocou klonovania obrazov diskov virtuálnych strojov na platforme OpenStack. Pre beh testov na tejto platforme je nutné aby mal Beaker vlastný OpenStack účet, nainštalované knižnice potrebné pre komunikáciu, aby boli obrazy diskov vopred nahraté do cloudu a aby testovacia sada vyžadovala beh na procesorovej architektúre x86_64. Očakáva sa, že toto vylepšenie prinesie urýchlenie prípravy testovacích strojov a tým pádom aj urýchlenie testovania a zvýšenie efektivity využitia výpočetných zdrojov. Výsledok práce má praktické využitie pre firmu Red Hat, ktorá ho plánuje používať pri pravidelnom testovaní balíčkov.



# Abstract

Integration testing in Beaker project is always performed in a clear environment, on a freshly installed operating system. The installation takes up a large percentage of the test machine preparation time. This work attempts to accelerate the process of preparation by cloning virtual machine images in OpenStack cloud platform. In order to run tests on this platform, Beaker must have it's own OpenStack account, some extra libraries installed on the Beaker server side, images must be uploaded into cloud in advance and the tests must require x86_64 cpu architecture. This feature is expected to accelerate the preparation of test machines and thus accelerate testing and increase the efficiency of computing resources usage. The result of this work is planned to be used by Red Hat for regular package testing.



# Kľúčové slová

Integračné testovanie, Beaker, OpenStack

# Keywords

Integration testing, Beaker, OpenStack


# Citácia

HOSTAČNÝ, Peter. *Využití diskových obrazů v projektu Beaker pro platformu OpenStack.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Smrčka Aleš.

# Využití diskových obrazů v projektu Beaker pro platformu OpenStack

## Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Aleša Smrčku, Ph.D. Ďalšie informácie a pomoc mi poskytli členovia Beaker teamu z firmy Red Hat (Brisbane). Uviedl som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Peter Hostačný
1. augusta 2017


## Poďakovanie

V prvom rade by som rád poďakoval svojmu vedúcemu práce Ing. Alešovi Smrčkovi, Ph.D. za pohotové reakcie a pomoc počas semestra a za konzultácie aj v čase dovolenky. Ďalej by som rád poďakoval celému Beaker teamu z Red Hatu. Menovite sú to hlavne Dan Callaghan a Róman Joost, ktorí mi vždy ochotne pomohli pri riešení rôznych chýb softvéru, na ktoré som počas práce narážal.

# Obsah

# Kapitola 1

# Úvod

Jeden z mnohých typov testovania softvéru je integračné testovanie. Táto práca sa zameriava na vylepšenie jedného z nástrojov, ktorý sa na integračné testovanie používa - projekt Beaker. Je to komunitný projekt firmy Red Hat a jeho zdrojové kódy, dokumentácia a ďalšie informácie sú voľne dostupné na jeho domovskej stránke [2]. V práci sú ďalej využité cloudové služby projektu OpenStack, ktorého integrácia už bola v Beakri počas jej písania v nejakej miere implementovaná (viac v kapitole 2.2). Oba projekty sú stručne popísané v kapitole 2.

Existuje viacero problémov, ktoré sa musia pri (nielen integračnom) testovaní riešiť. Jeden z nich je napríklad voľba, na akom stroji a v akom prostredí testy pobežia. Keďže testy môžu byť deštruktívneho charakteru (či už kvôli chybe v teste alebo v samotnom testovanom software) alebo môžu systém zanechať nie v pôvodnom stave, musí QA [1] inžinier riešiť ako zabezpečiť autentické chovanie testov a testovanej aplikácie - t.j. ako systém pripraviť tak, aby výsledky testov neboli nečakane ovplyvnené tým, čo sa na stroji dialo pred spustením testovacej sady. Pravdepoobne najjednoduchšie riešenie je spúšťať testy v čistom prostredí - tj. určiť stroj, na ktorý sa pred spustením testov nainštaluje operačný systém a nastaví sa podľa potrieb testovacej sady. Keďže vo veľa prípadoch (firmách) je nutné zdieľať zdroje, je potrebné mať softvér, ktorý automatizovane rieši management testovacích strojov. Vo firme Red Hat sa o toto stará nástroj Beaker, ktorý si udržuje inventár strojov (fyzických aj virtuálnych) a testeri majú pomocou užívateľského rozhrania možnosť si ich rezervovať, inštalovať na ne operačný systém podľa potreby, nastaviť si ich a spúšťať na nich testy alebo celé testovacie sady. Po skončení rezervácie sú stroje vracané do spoločného priestoru a sú opäť k dispozícii.

Celý tento prístup inštalácie operačného systému pre každý beh testovacej sady je obrovským spomalením celého procesu testovania. Častokrát sa stáva, že inštalácia systému zaberie niekoľkonásobne dlhšiu dobu ako samotný beh testovacej sady.

Hlavný prínos tejto práce je urýchlenie procesu prípravy testovacích strojov, konkrétne časti obsahujúcej inštaláciu operačného systému, čo vedie k zrýchleniu testovania. Toto sa dá teoreticky dosiahnúť pomocou využitia klonovania obrazov disku v platforme OpenStack, kedy sa inštalácia operačného systému preskočí a miesto toho sa na vytvorenie virtuálneho stroja použije vopred pripravený diskový obraz. Takýto prístup sa samozrejme dá použiť iba za určitých podmienok, ktoré sú vymenované a vysvetlené v podkapitole 3.1. Celý proces spolu s architektúrou systému testovania je bližšie popísaný v kapitole 3.

[1] quality assurance

Implementačné detaily sú obsiahnuté v kapitole 4 (rozšírenie/editácia kódu sa týka iba projektu Beaker). Overenie správnosti a demonštrácia výsledkov práce je v kapitole 5.

# Kapitola 2

# Testovanie systému v rámci projektu Beaker

V nasledujúcich podkapitolách bude vysvetlené:
2.1 Čo znamená integračné testovanie a na aké integračné testovanie sa používa Beaker.
2.2 Čo je to OpenStack a stručný popis jeho súčastí použitých v tejto práci.
2.3 Čo je to Beaker, z akých častí sa skladá a ako v ňom prebieha testovanie - tj. sekvencia krokov, ku ktorým dochádza od odoslania uživateľského požiadavku na spustenie testov až po reportovanie výsledkov späť užívateľovi.
2.3.1 Stav OpenStack integrácie v projekte Beaker pred mojím príspevkom do projektu.

## 2.1 Integračné testovanie

Integračné testovanie testuje integráciu alebo rozhrania medzi komponentami, interakciu s rôznymi súčasťami systému (ako je operačný systém, súborový systém, hardware) alebo rozhrania medzi systémami (preložené z [18]).

Zoznam niektorých typov integračného testovania, ktoré prebiehajú alebo môžu prebiehať v Beakri:

- testovanie jadra operačného systému na rôznych procesorových architektúrach (ppc64, s390, x86_64,...),
- testovanie kombinácií rôznych verzií webových serverov, databázových serverov a programovacích jazykov (napr. Apache + MySQL + PHP; Nginx + MariaDB + PHP a desiatky ďalších kombinácií líšiacich sa kľudne iba verziou komponenty),
- testovanie aplikácii využívajúcich rôzne knižnice (napr. v jazyku Python)
- a mnoho ďalších...

## 2.2 OpenStack

V tejto podkapitole sú vysvetlené termíny z prostredia OpenStack, ktoré budem v práci používať.

OpenStack je open-source cloudová platforma, ktorá užívateľom umožňuje využívať zdieľané výpočetné zdroje. Celý projekt pozostáva z viacerých kľučových služieb, ktoré sú spolu

prepojené a tvoria jeden celok. Asi najdôležitejšie z nich sú tieto (celý dizajn je možno nájsť v dokumentácii projektu OpenStack [14]):

- Identity service (Keystone) - služba poskytujúca jednotné rozhranie pre autentifikáciu.
- Image service (Glance) - služba slúžiaca na správu obrazov diskov virtuálnych strojov.
- Compute service (Nova) - služba pre vytváranie, správu a mazanie virtuálnych strojov (spomedzi všetkých služieb je najviac používaná).
- Networking service (Neutron) - správa sietí a sieťových rozhraní.
- Dashboard (Horizon) - graifcké webové uživateľské rozhranie.

Niektoré termíny z OpenStack terminológie spolu s vysvetlením ich významu:

- *Instancia* - (niekedy nazývaná aj Server) je označenie pre virtuálny stroj v OpenStacku.
- *Projekt* - (občas označovaný aj ako tenant) je v OpenStacku označenie pre množinu zdrojov, ktoré sú zdieľané v rámci určitej skupiny užívateľov. Dá sa chápať aj ako pomenovaná kvóta, ktorú môžu užívatelia (s prístupom do nej) využívať. Príklad: 10 instancií, 25 virtuálnych procesorov, 30 GB RAM, 5 verejných IP a pod.
- *Keystone trust* - sa dá chápať ako token, slúžiaci na delegáciu práv v určitom projekte druhému užívateľovi. Ten potom môže v mene užívateľa, ktorý *keystone trust* vytvoril, vytvárať, spravovať a mazať jeho instancie [11].
- *Flavor* - je pojem určený na pomenovanie veľkosti instancie. Pri vytváraní instancii užívateľ nevolí každý hardvérový parameter zvlášť, ale vyberá zo skupiny vopred definovaných veľkostí (flavors), pričom každá veľkosť má administrátorom definovaný počet CPU, veľkosť disku, pamäte RAM a podobne [13].

  príklad: m1.medium = {2 VCPUs, 40 GB disk, 4096 MB RAM}
- *Floating IP* - je špeciálny druh sieťového rozhrania poskytovaný službou Neutron, vďaka ktorému sú vytvorené instancie sieťovo dostupné mimo OpenStack. Floating IP nie je záležitosťou DHCP servera ani inak staticky prideľovaná vnútri instancie, takže instancia 'nevie o tom', že má takúto adresu pridelenú [6].

## 2.3 Beaker

Beaker je opensource nástroj pre správu a automatizáciu testovacích strojov, spúšťanie testov na týchto strojoch a zobrazovanie/archiváciu ich výstupov. Zoznam strojov (či už virtuálnych alebo fyzických) spolu s popisom ich hardvérového vybavenia si Beaker udržuje v databáze. Takisto si vedie list linuxových distribúcií, ktoré je na počítače možno inštalovať. Aktuálne sú podporované iba distribúcie založené na RPM (primárne Fedora a Red Hat Enterprise Linux) ktoré sa inštalujú pomocou inštalátora Anaconda [2].

Beaker (serverová časť) sa skladá z viacerých častí, ktoré spolu komunikujú, pričom každá môže bežať na inom počítači:

- *webserver* - poskytujúci webové rozhranie pre komunikáciu so serverom (v produkcii Apache, vo vývojovej verzii Gunicorn).

- *Scheduler* - (inak nazývaný aj beakerd) riadi celé plánovanie, výber strojov, vytvára kickstart súbory pre inštalátor Anaconda a komunikuje s lab controllermi (pojem vysvetlený nižšie) [2].
- *Database* - sú v nej uložené informácie o užívateľoch, skupinách, úlohách, ktoré bežali alebo práve budú bežať na testovacích strojoch, história aktivít, inventár testovacích strojov a veľa ďalších.
- *Lab Controller* - slúži na ovládanie (správu) testovacích strojov. K jednému beaker serveru môže byť pripoejných niekoľko lab controllerov. Každý stroj na testovanie musí byť pridelený práve jednému lab controlleru a každý lab controller môže mať pridelených x testovacích strojov. Skladá sa zo štyroch démonov (služieb) [8, 17]:
  - beaker-proxy - počúva na neprivilegovanom TCP porte (predvolene 8000) a plní úlohu XMLRPC servera. Spravidla iba presmerováva prichádzajúce požiadavky od testovacích strojov beaker serveru.
  - beaker-watchdog - predchádza zablokovaniu stroja tým, že preruší vykonávanie Recipe (pojem vsvetlený nižšie), ak doba jeho vykonávania presiahla určený časový limit.
  - beaker-provision - počas prípravy stroja má na starosti vytvorenie a nahratie bootloader image na TFTP server. Ďalej sa stará o príkazy napájania (reštart, vypnutie...) testovacieho stroja.
  - beaker-transfer - nahráva výstupy testov na archivačný server (je voliteľnou súčasťou). Defaultne sú tieto výstupy uložené lokálne v lab controlleri, dostupné cez webserver Apache.

Niektoré termíny (nielen) z Beaker terminologie, potrebné pre pochopenie zbytku práce, sú vysvetlené v nasledujúcich odstavcoch:

*Kickstart* - označuje sa ním metóda automatizovanej inštalácie operačných systémov Fedora a Red Hat Enterprise Linux (RHEL) [12]. V beaker terminológii sa tento pojem používa pre označeniu súboru, ktorý definuje parametre inštalácie. Súbor obsahuje napr. repozitáre pre inštaláciu, nastavenie jazyka, klávesnice, root hesla, post-inštalačné skripty a veľa ďalších možností.

*Test harnesses* - nástroj, ktorý na testovacom stroji spúšťa úlohy a reportuje výsledky späť do Beakru (konkrétne do lab controlleru). Základná test-harness sa nazýva *beah* a podporuje iba inštaláciu a spúšťanie úloh vopred importovaných do Beakru v podobe RPM balíčkov. Alternatívne harness (napr. *restraint*) podporujú aj iné spôsoby spúšťania úloh - napr. inštaláciu priamo z git repozitára.

*Job* - je označenie pre komplexné zadanie práce od užívateľa Beaker serveru. Zadáva sa pomocou XML súboru cez web-UI alebo cez Beaker clienta (konzolový klient pre Beaker server). Jeho schéma je popísaná na stránkach dokumentácie [10] kde je možné nájsť aj odkaz na aktuálny formálny zápis jeho schémy v jazyku RELAX NG [16]. Jeden Job môže obsahovať viacero "recipe sets"(pojem vysvetlený nižšie). Ak jeden z recipe setov skončí so so stavom Fail, celý Job je označený ako Failed.

*Recipe set* - obsahuje niekoľko (spravidla jeden) objektov Recipe, ktoré majú byť vykonané ako skupina. To znamená, že budú bežať v rámci jedného labu, že sa spustia až keď bude všetkým pridelený stroj, a že stroje, ktoré im boli pridelené sa uvoľnia až keď všetky Recipes skončia.

*Recipe* - obsahuje hardware požiadavky na stroj, na ktorom bude Recipe bežať, rovnako ako distribúciu, ktorá bude použitá na inštaláciu. Môže obsahovať dodatočné úpravy kickstart súboru, ktorý bude predložený Anaconde. Každý Recipe obsahuje minimálne jednu úlohu (Task).

*Task* - definuje aká úloha sa má spustiť a s akými parametrami. Konkrétny formát zadávania úloh je určený aj podľa použitej test-harness. Pre Recipes používajúce základnú test harness (beah) by každý Task mal odkazovať na jednu položku z Task Library.

*Task Library* - zoznam úloh (RPM balíčkov), ktoré je možno na strojoch inštalovať a spúšťať.

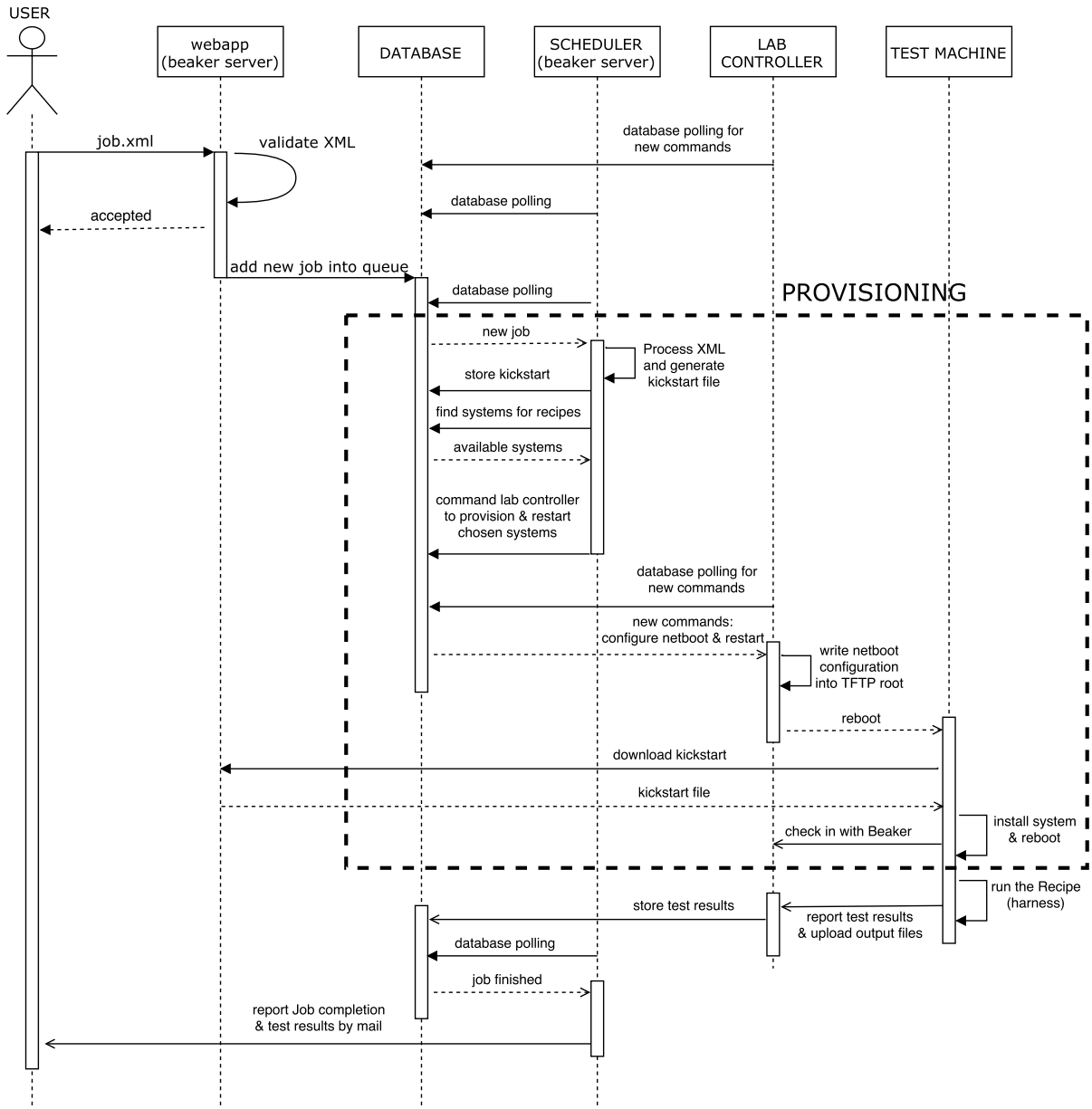
*distroRequires* - je element v schéme job.xml. Definuje sa pomocou neho distribúcia operačného systému pre testovací stroj. Každý recipe má svoju distroRequires sekciu.

*hostRequires* - takisto je to jeden elementov, ktoré obsahuje recipe v job.xml. Definujú sa pomocou neho harvérové požiadavky na testovací stroj.

*System* - v Beakri sa sytémom rozumie testovací stroj. V databáze sú o ňom uložené hardvérové detaily, stav (automated, broken, manual, removed), história aktivity a zoznam + štatistiky z behov Recipes.

Predchodzie definície sú inšpirované dokumentáciou - Beaker's capabilities [3].

Na nasledujúcom sekvenčnom diagrame (obr. 2.1) je zobrazená sekvencia krokov, ku ktorým dochádza od odoslania uživateľského požiadavku na spustenie testov, až po reportovanie výsledkov späť užívateľovi (zoznam krokov prebiehajúcich od výberu stroja až po jeho inštaláciu je podrobnejšie rozpísaný v Beaker dokumentácii - sekcia *Provisioning process* [15]).




Obr. 2.1: Sekvenčný diagram: spúšťanie testov v Beakri

Na obrázku 2.2 je ukážka bežného job.xml súboru, ktorý sa posiela Beakru. Konkrétne tento ukážkový job.xml bol vygenerovaný Beakrom pri administrátorskom príkaze na skenovanie hardvéru stroja s hostname="beaker-test-vm1.beaker". Job v ňom obsahuje jeden Recipe set, v ktorom sa nachádza jeden Recipe s požiadavkou na inštaláciu distribúcie CentOS 6 a s vynútením použitia konkrétneho stroja (parameter force v elemente hostRequires). Recipe obsahuje dve úlohy (Task), pričom v parametre druhej je použitý hostname stroja, ktorý by mal slúžiť pre účely identifikácie stroja počas reportovania výsledkov skenovania.

```
<job retention_tag="scratch">
  <whiteboard>Update Inventory for beaker-test-vm1.beaker</whiteboard>
  <recipeSet priority="Urgent">
    <recipe whiteboard="Update Inventory for beaker-test-vm1.beaker"
        role="RECIPE_MEMBERS" ks_meta="" kernel_options=""
        kernel_options_post="">
      <autopick random="false"/>
      <watchdog panic="None"/>
      <packages/>
      <ks_appends/>
      <repos/>
      <distroRequires>
        <and>
          <distro_family op="=" value="CentOS6"/>
          <distro_variant op="=" value=""/>
          <distro_name op="=" value="CentOS-6.9"/>
          <distro_arch op="=" value="x86_64"/>
        </and>
      </distroRequires>
      <hostRequires force="beaker-test-vm1.beaker"/>
      <partitions/>
      <task name="/distribution/install" role="STANDALONE"/>
      <task name="/distribution/inventory" role="STANDALONE">
        <params>
          <param name="HOSTNAME" value="beaker-test-vm1.beaker"/>
        </params>
      </task>
    </recipe>
  </recipeSet>
</job>
```

Obr. 2.2: Príklad job.xml súboru

### 2.3.1 Integrácia platformy OpenStack

OpenStack sa dá ako náhrada za fyzické stroje použiť iba v prípade, že nám stačí testovať na procesorovej architektúre x86_64 alebo i386. Sú to zatiaľ jediné architktúry, ktoré OpenStack podporuje [7].

Prvá podpora platformy OpenStack bola v projekte Beaker implementovaná vo verzii 0.17 [20]. Následne bola rozšírená vo verzii 24 (číslovanie sa medzitým zmenilo, preto ten skok) o viaceré kľúčové vylepšenia, ktorých zoznam je vypísaný v poznámkach k vydaniu danej verzie [19].

Pred začatím tejto práce bolo možné zapnúť OpenStack integráciu v projekte Beaker, ak na serverovej strane boli nainštalované potrebné python knižnice (zoznam je na stránkach dokumentácie [9]). Pre toto zapnutie bolo potrebné editovať Beaker databázu, nakonfigurovať odkazy na OpenStack Identity API (url), OpenStack Dashboard (url) a prístupove údaje do OpenStacku (užívateľské meno + heslo). Ďalej bolo potrebné nahrať špeciálny iPXE obraz disku do Glance (pojem vysvetlený v podkapitole 2.2). Celý návod je dostupný v dokumentácii Beakru dostupnej online [9].

Poskytovanie strojov z platformy OpenStack fungovalo následovne: Beaker pri každom Recipe skontroloval hardvérové požiadavky. Ak OpenStack dokázal uspokojiť tieto požiadavky a užívateľ mal vo svojom účte v Beakri vytvorený OpenStack trust, Beaker sa pokúsil

o vytvorenie virtuálneho stroja (v mene užívateľa, ktorý trust vytvoril). Ako základ pre virtuálny stroj sa použil vopred nahratý iPXE obraz disku. Ak do tohto bodu vytváranie z akejkoľvek príčiny zlyhalo, Beaker použil fallback ku klasickému poskytovaniu strojov. Po úspešnom vytvorení virtuálneho stroja sa ďalej sa pokračovalo bežnou inštaláciou operačneho systému pomocou inštalátora Anaconda.

# Kapitola 3

# Návrh využitia diskových obrazov v projekte Beaker

Integrácia platformy OpenStack do projektu Beaker bola prvým krokom a nutným základom pre implementáciu klonovania diskov s už nainštalovaným operačným systémom. Keďže virtuálne stroje, ktoré zdieľajú hardvér, môžu byť mierne pomalšie ako fyzické stroje, poskytovanie virtuálnych strojov z prostredia OpenStack malo výhodu hlavne v zlepšení dostupnosti testovacích strojov s procesorovou architektúrou x86_64. Je to z dôvodu lepšej škálovateľnosti, ktorú cloud ponúka. Testovací proces to však dokáže urýchliť iba iba tým spôsobom, že v prípade veľkej vyťaženosti testovacích strojov čaká Recipe kratšiu dobu od naplánovania po pridelenie testovacieho stroja. Celkový čas od odoslania požiadavku na testovanie až po obdržanie výsledkov testov sa teda týmto krokom mohol skrátiť, ale nie nutne.

Možnosť klonovania obrazov diskov prináša urýchlenie testovania, pri ktorom sa sa úplne preskakuje časť inštalácie operačného systému. Prináša to síce povinnosť dané obrazy do cloudu najskôr nahrať, ale pri opakovanom testovaní na rovnakej verzii operačného systému a hromadnom použití sa čas mnohonásobne vráti.

## 3.1 Analýza požiadavkov pre prípravu testovania

Pre implementáciu klonovania obrazov diskov použiteľnú v projekte Beaker bolo najväčším problémom nahradiť kickstart, pomocou ktorého Anaconda nielen inštalovala operačný systém, ale aj nastavovala stroj pre potreby testovania. To znamená inštaláciu balíčkov vyžadovaných v úlohách v danom Recipe, ohlásenie sa späť Beakru (nahlásenie ukončenia inštalácie, svojho hostname), vytvorenie konfiguračých súborov pre test harness, nastavenie služby pre synchronizáciu času, nakonfigurovanie repozitárov, inštalácia test harness, reštartovanie stroja po ukončení inštalácie a mnoho ďalších. Vzhľadom na to, že pri vytváraní virtuálnych strojov z vopred pripravených obrazov diskov sa Anaconda vôbec nespúšťa, bolo treba kickstart niečím nahradiť. Asi najlepšie riešenie, ktoré ponúka možnosť automatizácie je použitie *cloud-init* scriptov.

### 3.1.1 Cloud-init

Cloud-init [4] je open-source projekt, ktorý je určený na konfiguráciu a prispôsobenie instancii v cloude. Je to balíček dostupný vo väčšine známych linuxových distribucií a jeho hlavný účel je spustenie konfigurácie systému pri prvom štarte instancie (pri prvom bootovaní).

Okrem OpenStacku má podporu aj v iných cloudových platformách. Pri spúšťaní instancie sa script zadáva pomocou parametru *user data*, pričom väčšinou sa jedná o textový reťazec alebo odkaz na súbor so scriptom (platí pre vytváranie instancií cez web-ui, cli-clienta, aj python knižnicu). Najpoužívanejšie formáty sú shell script (začínajúci shebangom #!) alebo cloud config file (začínajúci #cloud-config) [5].

Použitie cloud-init je pre úžívateľa v Beakri úplne odtienené, tj. užívateľ s pojmom cloud-init vôbec nepríde do styku. Beaker si cloud-init script generuje automaticky z pripraveného kickstartu.

### 3.1.2 Predpoklady

Nasledujúci zoznam zhŕňa predpoklady pre beh testov na klonovanom obraze disku v OpenStacku:

1. Splnenie požiadavkov pre zapnutie OpenStack integrácie [9]:
   (a) Balíčky, ktoré musia byť nainštalované na stroji, kde beží Beaker scheduler (beakerd):
      - python-keystoneclient >= 0.11.0
      - python-novaclient >= 2.20.0
      - python-glanceclient >= 0.15.0
      - python-neutronclient >= 2.3.9

   (b) Konfigurácia následovných položiek v server.cfg
      - openstack.identity_api_url
      - openstack.dashboard_url
      - openstack.username
      - openstack.password

   (c) V beaker databáze sa musí nachádzať tabuľka openstack_region a v nej záznam obsahujúci ID lab controllera, s ktorým budú stroje z OpenStacku komunikovať (cudzí kľúč do tabuľky lab_controller).

2. Požiadavky na job.xml:
   (a) Nepoužiť `force` atribút v XML prvku `hostRequires` (`force` atribút vynucuje použitie konkrétneho stroja pomocou zadania jeho hostname).

   (b) Použiť architektúru x86_64 (definuje sa v prvku `distroRequires` - viď. element `distro_arch` v obrázku 2.2)

   (c) Pridať prvok `openstack_image` s atribútom `value` obsahujúcim názov obrazu disku do `hostRequires`. Ak existuje viacero obrazov s rovnakým menom, použitý je vždy ten najnovší so statusom 'active'.
   Príklad: `<openstack_image value="CentOS-7.2-x86_64-GenericCloud"/>`

   (d) Nutnosť definovať linuxovú distribúciu v `distroRequires` - rovnakú akú obsahuje obraz disku, ktorý bude použitý na klonovanie (kvôli vygenerovaniu správneho kickstart súboru, z ktorého sa generuje cloud-init script a takisto kvôli prípadnému fallback v prípade chyby počas vytvárania instancie).

3. Z účtu, ktorý zadáva job Beakru, musí byť vytvorený keystone trust. Vytvára sa cez web-ui v nastaveniach účtu: `http://BEAKER_URL/prefs/#keystone-trust` (OpenStack integrácia už musí byť zapnutá). Pri vytváraní keystone trust užívateľ zadáva svoje prístupové údaje do OpenStacku spolu s názvom projektu, do ktorého má prístup.

4. (voliteľné) V OpenStacku by mal bežať NTP server, ktorého adresu má poskytovať DHCP. Ak tam NTP server z akejkoľvek príčiny nebeží, je poreba vypnúť synchronizáciu času v testovacom stroji pomocou zadania atribútu `ks_meta="no_clock_sync"` v prvku Recipe v job.xml. Toto je voliteľný požiadavok - ak ho užívateľ vynechá, pri súčasnom kóde sa test harness bude asi 10 minút snažiť synchronizovať čas a potom sa spustí.

5. Požadovaný obraz disku musí existovať v OpenStacku. Užívateľ má možnosť vybrať si zo zoznamu verejných obrazov diskov (public), ktoré sú prístupné vo všetkých projektoch alebo si môže vytvoriť/nahrať vlastné obrazy (private). Nahrávanie vlastných obrazov diskov a ich následné využitie v Beakri bolo otestované iba pomocou účtov s prístupom do jedného projektu.

Po splnení všetkých povinných požiadavkov uvedených vyššie by mal Beaker poskytnúť virtuálny stroj z OpenStacku a miesto inštalácie by mal naklonovať užívateľom uvedený obraz disku.

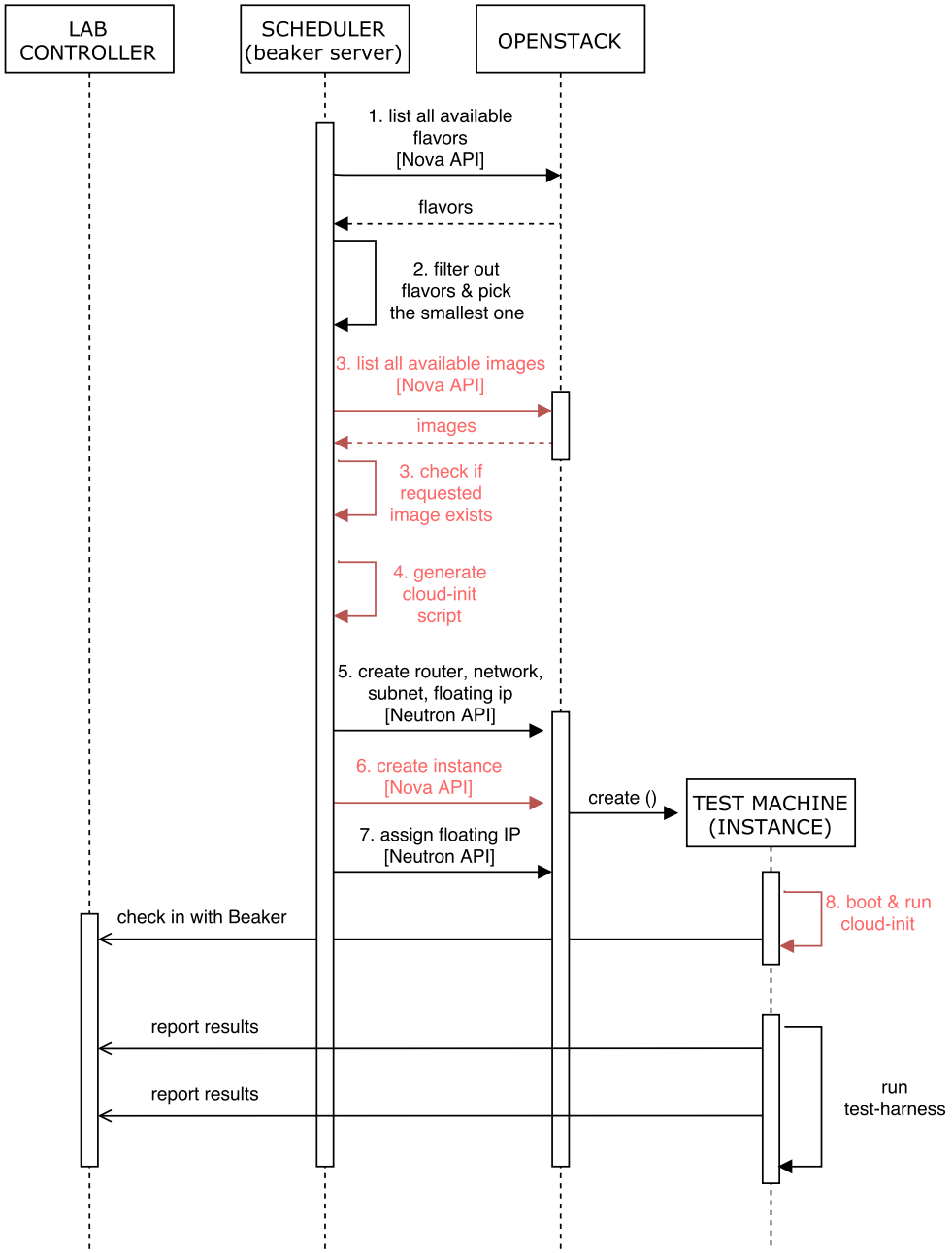
## 3.2 Architektúra systému testovania

Na nasledujúcom sekvenčnom diagrame (obr. 3.1) sú zobrazené jednotlivé kroky, ku ktorým dochádza počas poskytovania virtuálneho stroja z OpenStacku, za pomoci klonovania obrazu disku (tzn. časť z predchodzieho sekvenčného diagramu na obr. 2.1 označná ako PROVISIONING, bez komunikácie s databázou). Časová os začína v bode, kedy Beaker detekoval, že Recipe môže bežať na virtuálnom stroji. Predpokladá sa, že boli splnené všetky požiadavky z predchádzajúcej podkapitoly 3.1.2. Červené časti boli implementované/zmenené ako súčasť tejto práce.

Popis k diagramu:

1. Beaker si od OpenStacku cez python-novaclient knižnicu vyžiada zoznam všetkých flavors.
2. Pomocou požiadavkov zo sekcie hostRequires z job.xml z tohto zoznamu vyfiltruje všetky, ktoré nespĺňajú podmienky. Následne sú všetky flavors zoradené podľa veľkosti pamäte a najmenší z tohto zoznamu sa zvolí pre vytvorenie VM.
3. Ďalej si Beaker vyžiada zoznam všetkých dostupných obrazov diskov a skontroluje, či sa v ňom nachádza obraz, ktorý požaduje užívateľ. Keďže užívateľ obraz disku definuje pomocou názvu a v OpenStacku tento parameter obrazu nie je unikátny, Beaker sa snaží vybrať posledný aktualizovaný a označený ako aktívny.
4. Po výbere obrazu disku sa vygeneruje cloud-init script. Na jeho generáciu sa ako základ použije už vygenerovaný kickstart súbor.
5. Následne sa pomocou python-neutronclient knižnice v OpenStacku vytvorí nový router, sieť, subnet a floating IP.

6. Vytvorí sa instancia. Na jej vytváranie sa použije vopred vybratý flavor, obraz disku a ako 'userdata' parameter sa použije vygenerovaný cloud-init script.
7. Instancii sa pridelí floating IP, takže sa na ňu užívateľ v prípade rezervácie bude môcť prihlásiť.
8. Virtuálny stroj sa naštartuje, cloud-init spustí konfiguráciu a na konci stroj reštartuje (súčasť skriptu). Ďalej je testovacie workflow nezmenené, okrem malého rozdielu - o vypnutie a zmazanie instancie sa stará scheduler priamo, bez účasti lab controlleru.




Obr. 3.1: Sekvenčný diagram: poskytovanie stroja z OpenStacku

# Kapitola 4

# Implementačné detaily rozšírenia projektu beaker

Editované súbory:

`./Server/bkr/server/config/app.cfg`

Táto úprava pridáva ďalšie lokality, z ktorých Gunicorn (webserver použitý ako náhrada Apache počas vývoja) staticky poskytuje súbory.

`./Common/bkr/common/schema/beaker-job.rng`

Úprava súboru potrebná pre zmenu XML schémy, pomocou ktorej Beaker overuje správnosť zadaného job.xml súboru.

`./IntegrationTests/src/bkr/inttest/server/selenium/*`
`./IntegrationTests/src/bkr/inttest/server/test_update_status.py`
`./Server/bkr/server/tests/data_setup.py`

Refactoring kódu kvôli oddeleniu funkcionality do dvoch metód (generovanie kickstart súboru a provisioning).

`./Server/bkr/server/dynamic_virt.py`

Vytvorenie metódy pre generovanie cloud-init scriptu. Komunikácia s OpenStackom pomocou knižníc python-novaclient. Doplnenie funkcionality pre výber obrazu disku.

`./Server/bkr/server/model/scheduler.py`

Doplnenie metódy typu getter pre vyňatie názvu obrazu disku z job.xml súboru. Rozdelenie metódy `Recipe::provision()` do dvoch metód: `Recipe::generate_kickstart()` a `Recipe::provision()`. Dôvodom bola nutnosť vygenerovať kickstart súbor skôr ako sa volala metóda `VirtManager::create_vm()`, pre účely generácie cloud-init scriptu.

`./Server/bkr/server/needpropertyxml.py`

Doplnenie `XmlOpenStackImage` triedy do stromu, ktorý vzniká z job.xml súboru. Vytvorenie jednej metódy pre parser, ktorý filtruje OpenStack flavors na základe HW požiadavkov a druhej, ktorá detekuje, či je daný HW požiadavok možný splniť v OpenStacku.

`./Server/bkr/server/tools/beakerd.py`

Zmena poradia volania funkcii kvôli predávaniu parametrov.

# Kapitola 5

# Overenie správnosti a demonštrácia

V nasledujúcich podkapitolách sa nachádzajú výsledky testov ktoré overujú pridanú funkcionalitu do projektu Beaker. V podkapitole 5.2 sú zobrazené tabuľky s ukážkou redukcie času prípravy testovacieho stroja v prípade použitia klonovania disku miesto inštalácie.

## 5.1 Overenie správnej funkcionality

Táto podkapitola obsahuje 3 testy - ich vstupy, výstupy a výsledok.

V prvom teste (podkapitola 5.1.1) bude otestované, či sa po splnení predpokladov z kapitoly 3.1.2 naplánuje beh testov na stroji z OpenStacku a či sa pri príprave stroja naklonuje obraz disku uvedený užívateľom miesto inštalácie operačného systému.

V druhom teste (podkapitola 5.1.2) sa zmeria doba potrebná na prípravu testovacieho stroja klonovaním a doba potrebná na inštaláciu operačného systému. Obe merania budú obsahovať aj konfiguračnú časť prípravy, keďže pri inštalácii sa táto konfigurácia vykonáva v rámci behu inštalátora Anaconda.

V treťom teste (podkapitola 5.1.3) sa porovná beh rovnakej testovacej sady na dvoch strojoch, pričom jeden bude bežať na naklonovanom obraze disku a druhý na inštalovanom.

### 5.1.1 Test - splnenie požiadavkov

Nasledujúci test overil, či sa po zadaní job.xml na obrázku 5.1 Beakru naplánuje beh testov na klonovanom obraze disku.

Zoznam predpokladov, ktoré boli splnené z kapitoly 3.1.2:

1. Zapnutie OpenStack integrácie.
   (a) Balíčky nainštalované na stroji, kde beží Beaker scheduler (beakerd):

```
python-glanceclient-0.17.3-2.el7ost.noarch
python-keystoneclient-1.3.0-2.el7ost.noarch
python-neutronclient-2.4.0-2.el7ost.noarch
python-novaclient-2.23.0-2.el7ost.noarch
```

(b) Položky v konfiguračnom súbore pre Beaker:

```
openstack.identity_api_url = "http://$OPENSTACK_URL:5000/v3"
openstack.dashboard_url = "http://$OPENSTACK_URL/dashboard/"
openstack.username = "$USERNAME"
openstack.password = "$PASSWD"
```

(c) V beaker databáze sa nachádza tabuľka openstack_region a v nej záznam obsahujúci ID lab controllera - 1.

2. Požiadavky na job.xml - viď. obrázok 5.1:

   (a) `force` atribút sa nenachádza v `hostRequires`.

   (b) Prvok `distroRequires` obsahuje požiadavku na architektúru x86_64.

   (c) V `hostRequires` sa nachádza prvok `openstack_image` s atribútom `value` obsahujúcim názov obrazu disku - `CentOS-7.2-x86_64-GenericCloud`.

   (d) V `distroRequires` je definovaná distribúcia CentOS 7 - rovnaká ako v požadovanom image, ktorý sa bude klonovať.

3. Z účtu, ktorý zadáva job Beakru, bol vytvorený keystone trust - obrázok 5.2.

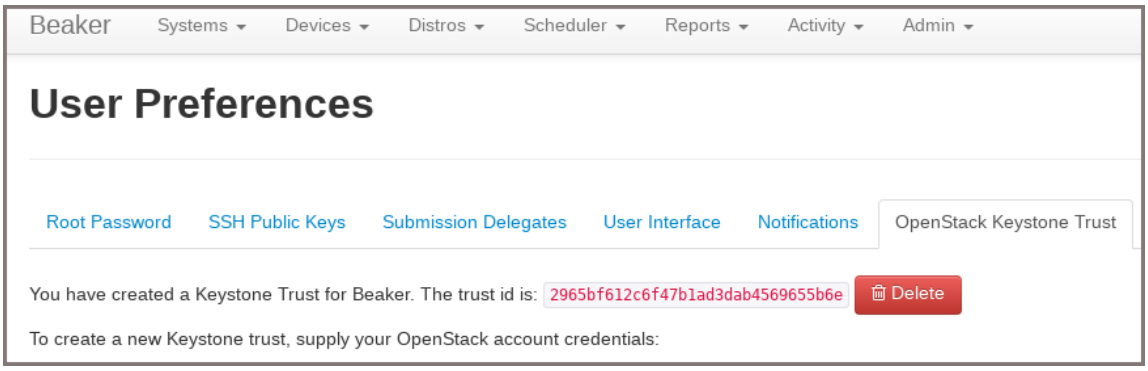
4. Synchronizáca času je vypnutá - v prvku Recipe v job.xml (obrázok 5.1) sa nachádza atribút `ks_meta` s hodnotou `"no_clock_sync"`.

5. Požadovaný obraz disku existuje v OpenStacku - obrázok 5.3.

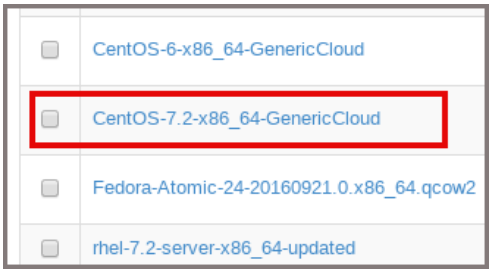
Na obrázku 5.4 je vidieť screenshot z Beaker GUI - Recipe s číslom 120 bežal na OpenStack instancii s UID=887f5a5c-1450-4dd3-b490-2f4780e7e712. Na obrázku 5.5 je screenshot z OpenStack webového rozhrania, na ktorom je vidieť, že ako základ pre vytvorenie instancie s vyššie uvedeným UID bol použitý image `CentOS-7.2-x86_64-GenericCloud`. Vzhľadom na to, že vo webovom rozhraní Beakru sú výsledky z úloh, bežiacich na tejto instancii, znamená to, že nastavenie testovacieho stroja (test-harness) prebehlo úspešne. Ukázalo sa tým, že po splnení predpokladov z kapitoly 3.1.2 Beaker miesto inštalácie stroja spustí klonovanie a pomocou cloud-init v testovacom stroji prebehne nastavenie prostredia potrebného pre spúšťanie testov.

```
<job retention_tag="scratch">
  <whiteboard>OpenStack - CentOS 7, test Job, cloning image</whiteboard>
  <recipeSet priority="Normal">
    <recipe whiteboard="OpenStack, CentOS 7, test Recipe, cloning image"
         role="None" ks_meta="no_clock_sync" kernel_options=""
        kernel_options_post="">
      <autopick random="false"/>
      <watchdog panic="None"/>
      <packages/>
      <ks_appends/>
      <repos/>
      <distroRequires>
        <and>
          <distro_name op="=" value="CentOS Linux-7"/>
          <distro_arch op="=" value="x86_64"/>
        </and>
      </distroRequires>
      <hostRequires>
        <system_type value="Machine"/>
        <openstack_image value="CentOS-7.2-x86_64-GenericCloud"/>
      </hostRequires>
      <partitions/>
      <task name="/distribution/install" role="STANDALONE"/>
      <task name="/distribution/command" role="None">
        <params>
          <param name="CMDS_TO_RUN" value="netstat -lautnp"/>
        </params>
      </task>
      <task name="/distribution/reservesys" role="STANDALONE">
        <params>
          <param name="RESERVE_IF_FAIL" value="True"/>
          <param name="RESERVETIME" value="10800"/>
        </params>
      </task>
    </recipe>
  </recipeSet>
</job>
```

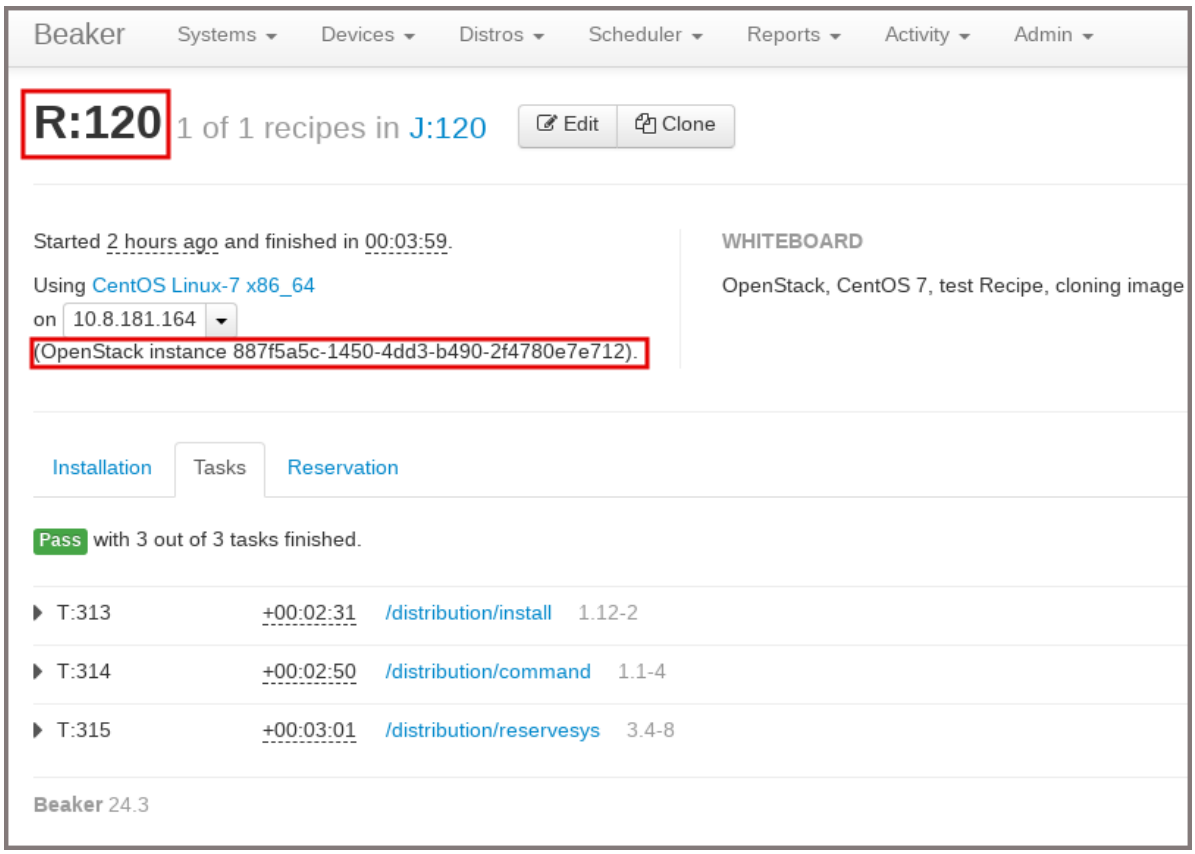
Obr. 5.1: Príklad job.xml s Recipe vyžadujúcim klonovanie disku



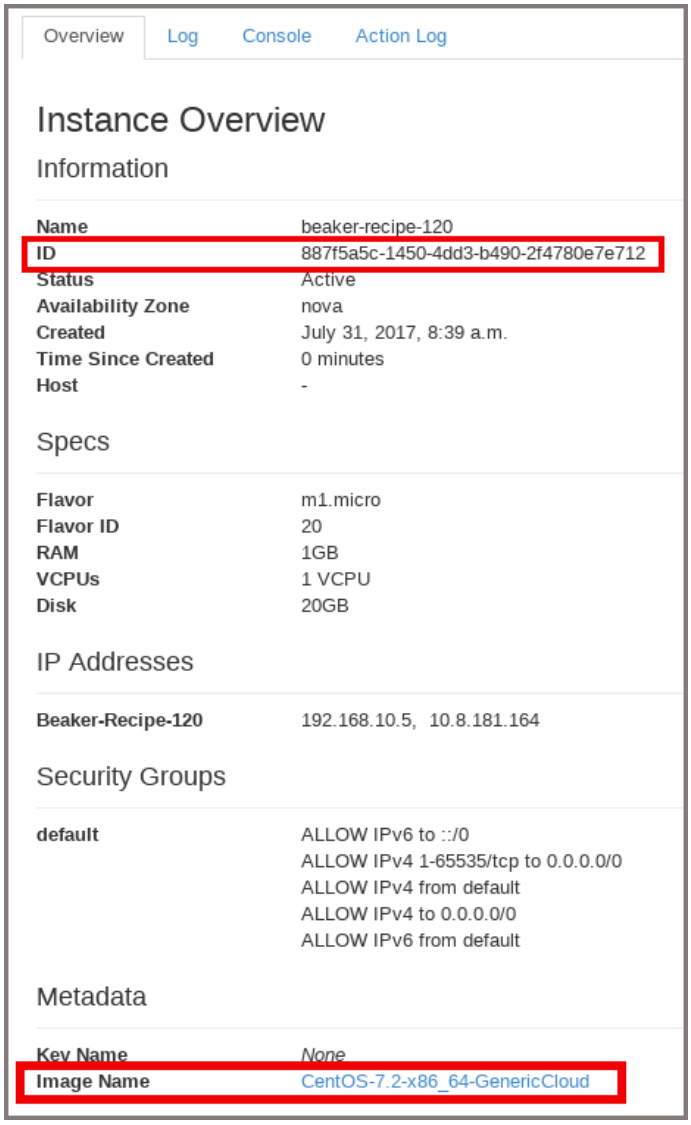

Obr. 5.2: Beaker web-ui: keystone trust vytvorený v nastaveniach účtu




Obr. 5.3: OpenStack zoznam verejných obrazov diskov




Obr. 5.4: Screenshot Recipe 120 z Beaker GUI




Obr. 5.5: Screenshot instancie v OpenStacku počas behu Recipe 120

### 5.1.2 Test - porovnávanie času prípravy testovacieho stroja

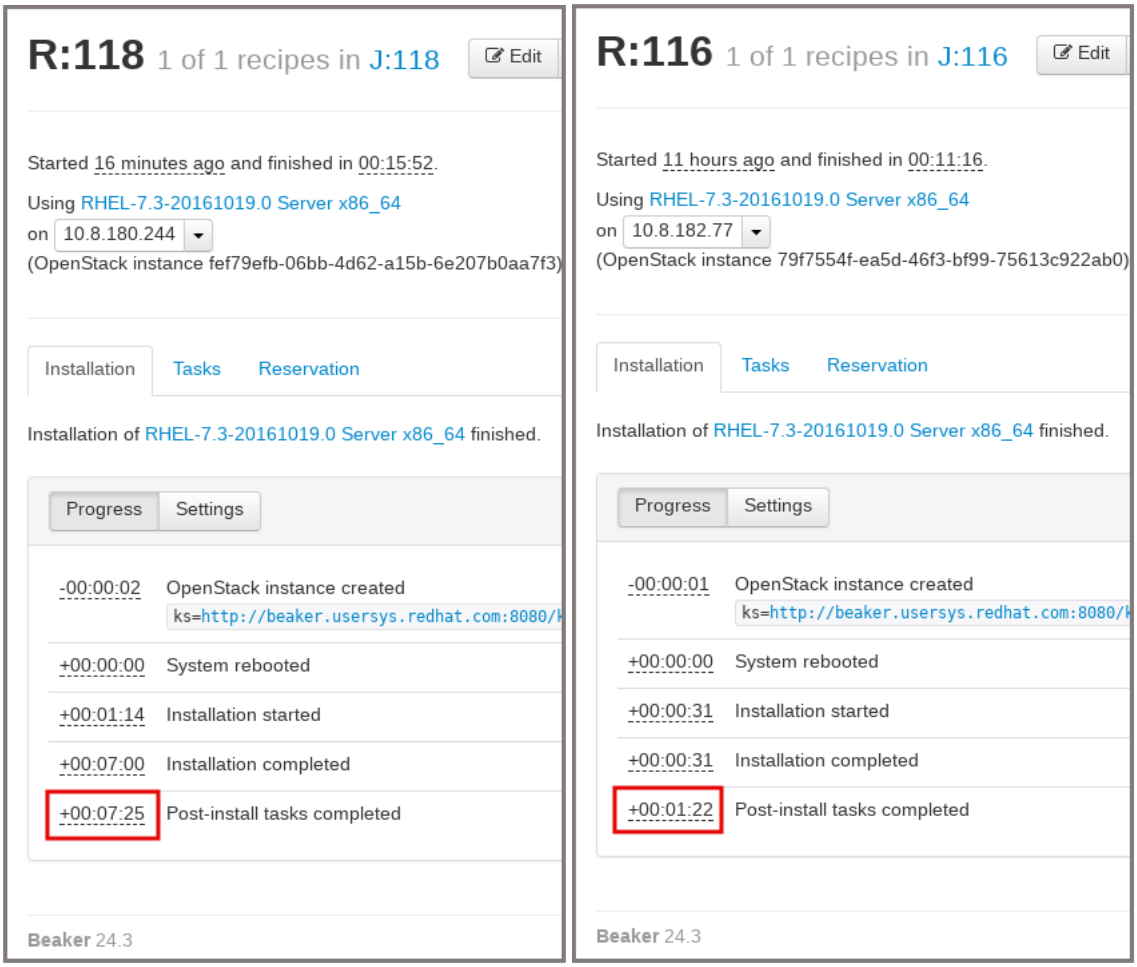
Na nasledujúcom teste sa overilo, že príprava stroja pomocou klonovania obrazu disku je naozaj rýchlejšia ako keď sa operačný systém inštaluje.

Porovnávanie prebehlo medzi inštaláciou Red Hat Enterprise Linux (RHEL) verzie 7.3 a klonovaním obrazu disku, ktorý obsahoval RHEL inštalovaný z rovnakého repozitára.

Na obrázku 5.6 vidíme čas nutný na prípravu stroja dosiahnutý:

- pri inštalácii: 7 minút a 25 sekúnd,
- pri klonovaní: 1 minúta a 22 sekúnd.

Vzhľadom na to, že pri klonovaní disku nedochádza k žiadnej inštalácii operačného systému, treba podotknúť, že správy zobrazujúce kernel command line (prvá správa vo výpise, obsahujúca odkaz na kickstart súbor), `Installation started` a `Installation completed`




(a) Inštalácia (b) Klonovanie

Obr. 5.6: Porovnanie časov inštalácie a klonovania disku s operačným systémom Red Hat Enterprise Linux 7

sú mierne zavádzajúce, pretože na klonovaných instanciách k žiadnej inštalácii operačného systému nedochádza. Takisto správa `Post-install tasks completed` pri klonovaní označuje koniec cloud-init scriptu a nie post-install sekcie v kickstarte. Tieto správy sú tam pretože sú obsiahnuté v samotnom kickstart súbore, z ktorého je generovaný cloud-init script. Instancie pomocou nich (cez XMLRPC volania) dávajú Beakru najavo v akom stave sa aktuálne nachádzajú. Pre ich nahradenie bude potreba viacero zmien v XMLRPC serveri, ktoré bude nutné vykonať pred nasadením riešenia do produkcie.

### 5.1.3 Test - porovnávanie výsledkov úloh (testov) v Beakri

Test v tejto podkapitole ukázal, že testy sa na instancii, ktorá vznikla klonovaním obrazu disku, správali rovnako, ako na nainštalovanom stroji. Boli použité rovnaké Recipes ako v predchádzajúcom teste (5.1.2) - tj. recipe 116 prebehol na klonovanej instancii a recipe 118 na inštalovanej. Jediný rozdiel v job.xml použitom na vytvorenie oboch jobov bol element pridaný v recipe 116 - `<openstack_image value="rhel-7.3-server-x86_64-released"/>`

Testy sa spúšťajú cez špeciálny wrapper, ktorý je používaný na reálne testovanie balíčka v Beakri. Do Beakru bol pridaný ako RPM balíček pomocou funkcionality 'Add new Task', takže sa nachádza v Task Library pod názvom `/CoreOS/abrt/Regression/Upstream`. Testy sú z upstream verzie balíčka abrt v repozitári na GitHube (vetva RHEL7) [1].

Wrapper dostáva názov testu, ktorý má spustiť, pomocou parametru v job.xml:

```
<task name="/CoreOS/abrt/Regression/Upstream" role="None">
  <params>
    <param name="ABRTTEST" value="ccpp-plugin-java"/>
  </params>
</task>
```

Boli spustené nasledujúce testy:

```
ccpp-plugin-java
kernel-vmcore-harvest
dbus-argument-validation
cli-authentication
rhts-problem-report-api
upload-watcher-stress-test
ureport-machineid
```

Výsledky testov sa zhodovali v oboch prípadoch - tj. prvý test zlyhal a ostatné skončili úspešne. Výstupy testov sa taktiež zhodujú.

## 5.2 Prínos

Vyhotovenie tejto práce prinieslo výsledky zobrazené v nasledujúcej tabuľke 5.7:

| | Čas potrebný na prípravu stroja [sekundy] | | | |
|---|---|---|---|---|
| | virtuálny stroj lokálne | OpenStack | OpenStack | |
| **Operačný systém** | inštalácia | inštalácia | klonovanie | |
| Red Hat Enterprise Linux 7.3 | 2465 | 473 | 82 | Server variant |
| Centos 7.2 | 1086 | 480 | 65 | |
| Red Hat Enterprise Linux 6.9 | 2391 | 457 | 65 | Workstation variant |
| Fedora 24 | 635 | 420 | 67 | |

Obr. 5.7: Porovnanie časov prípravy testovacieho stroja na rôznych platformách a s rôznymi operačnými systémami

Treba podotknúť, že obraz disku niekto musí vopred vytvoriť a nahrať do platformy OpenStack, čo tiež zaberie určitý čas. Keď sa ale tento čas porovná s časom, ktorý sa ušetrí pri klonovaní – ak zoberieme do úvahy, že testovanie sa spúšťa opakovane – zistíme, že ušetrený čas ďaleko presiahne čas prípravy obrazu.

Test poukázal aj na ďalšiu dôležitú vec - doba prípravy testovacieho stroja je pri inštalácii z repozitára v sieti silne ovplyvnená rýchlosťou sieťového pripojenia a aktuálnym

stavom siete. Je to kvôli veľkému objemu dát, ktoré je počas inštalácie potrebné stiahnúť. Odľahčenie siete je teda ďalší aspekt, ktorý sa klonovaním obrazov diskov v OpenStacku zlepší. V prvom stĺpci možno vidieť veľký nepomer prípravnej doby, keďže pri testovaní bežal Beaker server lokálne (v laptope v ČR) a operačné systémy boli sťahované z rôznych úložísk (RHEL z USA a CentOS+Fedora z ČR).

# Kapitola 6

# Záver

Cieľom tejto práce bolo analyzovať a urýchliť proces automatizovanej prípravy testovacích systémov v projekte Beaker s využitím cloudovej platformy OpenStack.

Toto sa podarilo pomocou zmien v príprave testovacieho stroja tým, že sa miesto inštalácie operačného systému klonuje obraz disku, v ktorom je operačný systém už nainštalovaný. V opakovaných testoch a v testoch využívajúcich rovnaký operačný systém tak môže dôjsť k badateľnému zrýchleniu. Toto taktiež závisí od charakteristiky testovacej sady - ak je beh samotných testov krátky, dobu ušetrenú klonovaním disku bude poznať oveľa viac ako pri testovacích sadách, ktoré majú exekučnú dobu niekoľko hodín. Klonovanie obrazu disku môže byť ako náhrada za inštaláciu použitá iba v OpenStacku, čo znamená urýchlenie prípravy stroja iba pre testy na architektúre x86_64 a s takými hardvérovými požiadavkami, ktoré nebránia použitiu virtuálnych strojov.

Z pohľadu ďalšieho vývoja projektu by bolo vhodné vytvoriť všeobecný cloud-init script, aby užívateľ nebol nútený zadávať distribúciu operačného systému odpovedajúcu obrazu disku - tj. aby bol cloud-init schopný toto detekovať sám a podľa toho prispôsobiť príkazy na nastavovanie stroja. Ďalej by bolo treba upraviť xmlrpc server (beaker-proxy kód) a webovú aplikáciu, aby užívateľ nevidel staré hlásenia o začiatku a konci inštalácie ale začiatok a koniec nastavovanie stroja cloud-init scriptom.

# Literatúra


[1] abrt: Automatic bug detection and reporting tool. Červenec 2017, original-date: 2013-02-01T14:38:04Z.
URL https://github.com/abrt/abrt

[2] Beaker architecture · Beaker 24.3. 2017.
URL https://beaker-project.org/docs/admin-guide/architecture.html

[3] Beaker's capabilities · Beaker 24.3. 2017.
URL
https://beaker-project.org/docs/architecture-guide/capabilities.html

[4] cloud-init in Launchpad. 2017.
URL https://launchpad.net/cloud-init

[5] Configuring instances at boot time. 2017.
URL https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/4/html/End_User_Guide/user-data.html

[6] Difference between Floating IP and private IP — RDO. 2017.
URL https://www.rdoproject.org/networking/difference-between-floating-ip-and-private-ip/

[7] HeterogeneousInstanceTypes – OpenStack. 2017.
URL https://wiki.openstack.org/wiki/HeterogeneousInstanceTypes

[8] Installation · Beaker 24.3. 2017.
URL https://beaker-project.org/docs/admin-guide/installation.html

[9] Integration with OpenStack · Beaker 24.3. 2017.
URL https://beaker-project.org/docs/admin-guide/openstack.html

[10] Job XML · Beaker 24.3. 2017.
URL https://beaker-project.org/docs/user-guide/job-xml.html#job-xml

[11] Keystone/Trusts – OpenStack. 2017.
URL https://wiki.openstack.org/wiki/Keystone/Trusts

[12] Kickstart Documentation — Pykickstart 3.7 documentation. 2017.
URL http://pykickstart.readthedocs.io/en/latest/kickstart-docs.html#what-are-kickstart-installations

[13] Manage flavors. 2017.
URL https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/4/html/Administration_User_Guide/cli_manage_flavors.html

[14] OpenStack Docs: Design. 2017.
URL https://docs.openstack.org/arch-design/design.html

[15] Provisioning process · Beaker 24.3. 2017.
URL https://beaker-project.org/docs/architecture-guide/provisioning-process.html

[16] RELAX NG home page. 2017.
URL http://relaxng.org/

[17] Source code walk-through · Beaker development. 2017.
URL https://beaker-project.org/dev/guide/source-walkthrough.html

[18] What is Integration testing? 2017.
URL http://istqbexamcertification.com/what-is-integration-testing/

[19] What's New in Beaker 24? · Beaker 24.3. 2017.
URL https://beaker-project.org/docs/whats-new/release-24.html

[20] Nick Coghlan: OpenStack Based Dynamic Virtualization · Beaker development. 2017.
URL https://beaker-project.org/dev/proposals/dynamic-virtualization.html

# Prílohy