label3.334Otevřené porty na proxy serverutable.3.3

Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

Infrastruktura pro centralizovanou automatickou instalaci serverů

Jan Sokol

Vedoucí práce: Ing. Josef Dostál

29. dubna 2017

Poděkování

Doplňte, máte-li komu a za co děkovat. V opačném případě úplně odstraňte tento příkaz.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 29. dubna 2017 .....

Odkaz na tuto práci

Sokol, Jan. *Infrastruktura pro centralizovanou automatickou instalaci serverů.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato bakalářská práce se zabývá průzkumem a implementací systému na instalaci serverů bez lidského dozoru. Cílem práce je využití některého již použitého systému a s úpravami nasazení do produkce. V teoterické části práce je popsána analýza X vybraných nástrojů. Mezi kritéria výberu těchto nástrojů patřilo z ekonomických a praktických důvodů open-source řešení, dále rozvinutá komunita, jež může pomoci s řešením případných problémů s nástrojem. Na základě analýzy z předchozí části byl vybrán nástroj, který nejlépe vyhovoval stanoveným požadavkům: jednoduše použitelné grafické rozhraní, podpora instalace široké škály operačních systémů (podmínkou alespoň OS Debian a CentOS) a možnost rychlých úprav v konfiguracích instalovaných serverů. Následovně je popsáno nasazení vyhovujícího nástroje Foreman, jak hlavního uzlu, tak proxy serverů v oddělených lokalitách. V další části je popsán vyvoj a nasazení pluginu v prostředí Foreman, který v jeho grafickém rozhraní zobrazuje grafy z dat sesbírané démonem collectd.

Klíčová slova linux, foreman, provisioning, server

Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

Keywords linux, foreman, provisioning, server

Obsah

Seznam obrázků

Úvod

Jakýkoliv hardware spojovaný s počítači, dříve než začne vykonávat svoji činnost, vyžaduje mít nainstalovaný a správně nakonfigurovaný software. Příkladem může být obyčejný stolní počítač, který potřebuje v sobě mít nainstalovaný operační systém spolu s ovladači. Tato instalace spolu s následující konfigurací může být časově poměrně náročná. Pokud ale těchto počítačů, potažmo serverů bez nainstalovaného operačního systému (pozn. v angličtině bare metal machine) máme desítky, dokonce stovky, čas vložený do instalace těchto strojů se může vyšplhat na neúnosné číslo. Takovouto mechanickou a nijak záživnou práci je systémový administrátor, či osoba pracující v serverovně nucena pravidelně vykonávat. Mnoho chytrých hlav nad problémem automatizace zmíněných úkonů produmalo dlouhé večery. Z tohoto úsilí na svět přišla řada nástrojů na zavedení operačního systému do čistého serveru. Tyto nástroje jsou hojně využívány jak v komerčním prostředí, tak v akademickém.

Životní cyklus serveru se skládá ze tří částí:

- provisioning, tedy chvíle, kdy se snažíme oživit server. Mezi to patří konfigurace DHCP, DNS a dalších částí, které potřebujeme, aby stroj komunikoval se sítí. Pro slovo provisioning není v češtině jednoduchý překlad. Vysvětlil bych ho takto: vytvoření a nakonfigurování nového stroje (ať už virtuálního, nebo s opravdovým hardware) tak, že bude obsahovat nainstalovaný operační systém a nakonfigurovaný přístup k síti, aby bylo možné s ním komunikovat,

- konfigurace, tj. část práce, kdy se snažíme server dostat do pro nás použitelné podoby – instalujeme dodatečné balíčky a nastavíme je,

- reportování, což je část cyklu, kdy je na serveru již vše nastaveno a monitorujeme správnou funkci.

Cíl práce

Bakalářská práce má za úkol sestavit infrastrukturu pro plně automatizovanou instalaci serverů. Předchází tomu popis jednotlivých standardů a protokolů využívaných pro dané úkony. Další částí práce je analýza jednotivých frameworků – od více k méně populárním. V práci jsou zahrnuty pouze open-source řešení. Existují také placené frameworky s profesionální podporou, o nich ale práce nepojednává. Z této analýzy je cílem vybrat nejvhodnější framework pro naše požadavky.

Vzniklý systém s vybraným frameworkem má mít možnost škálování. Tím je myšleno přídání dalších datacenter, tj. přídání dalších oddělených LAN sítí do správy infrastruktury.

Kapitola 1

Teorie zavedení operačního systému

Pro celou analýzu a následnou stavbu infrastruktury, o které práce uvažuje, je důležité teoretické pochopení jednotlivých kroků probíhajících od zapnutí serveru až po zavedení operačního systému. Tato kapitola se věnuje popsání jednotlivých standardů a protokolů využívaných při zavádění obrazů disků.

1.1 Bare metal server

S přesunem IT infrasktruktury do „cloudu“ se v našem průmyslu začínají objevovat nové výrazy, mezi které patří i bare metal. Nejjednodušeji řečeno, bare metal server je klasický dedikovaný server s novým, pěkným jménem.

Ukáži na příkladu: pokud si zákazník objedná bare metal server, přístup k hardwaru má pouze on a s nikým ho nesdílí. Šířka pásma na síťové kartě tedy patří jen jemu a celou ji může využít.

1.2 Provisioning a frameworky pro tento účel

Pokud v IT oblasti potkáme výraz „provisioning“, obyčejně tím myslíme sérii kroků, během kterých připravíme server s přeurčeným operačním systémem, daty a dalším vybraným softwarem. Následně na serveru připravíme připojení k síti. Mezi typické kroky patří:

- vybrání serveru z databáze volných strojů,
- nainstalování odpovídajícího softwaru (operační systém, ovladače pro hardware, aplikace),
- konfigurace nainstalovaného softwaru z předchozího kroku (nastavení připojení k síti, tj. přiřazení IP adresy, brány, atp.).

Po provedení kroků se systém restartuje a načte nový operační systém. Server je zde již připraven k použití.

Pro provisioning existuje řada nástrojů (frameworků), které se tuto činnost snaží zautomatizovat. Nástroje se většinou ve své funkčnosti překrývají, můžeme je tedy poměrně dobře porovnat. Porovnání se bude věnovat další kapitola.

1.3 Zavedení kódu s PXE

PXE (Pre eXecution Environment) je metodou zavedení nějakého kódu (klasicky zavaděče) na koncovém zařízení pouze pomocí jeho síťové karty. PXE je definován na základě standardů internetových protokolů a služeb široce využívaných v průmyslovém světě. Jmenovitě to jsou TCP/IP, DHCP, TFTP, které standardizují formu komunikace mezi serverem a klienty.[**?**] Metoda byla prvně popsána v roce 1999 [**?**].

PXE standard pracuje v následující krocích:

1. Klient zahájí komunikaci zasláním DHCPDISCOVER paketu na broadcast adresu. K tomu se využije standartního portu DHCP - 67 na UDP. Tento paket obsahuje rozšíření, podle kterého server pozná, že paket přichází od klienta s implementovaným PXE protokolem. Konkrétně tedy DHCP Option 60, nastavena na PXEClient:Arch:xxxxx:UNDI:yyyzzz. (Předpokládáme, že DHCP server toto rozšíření podporuje.)

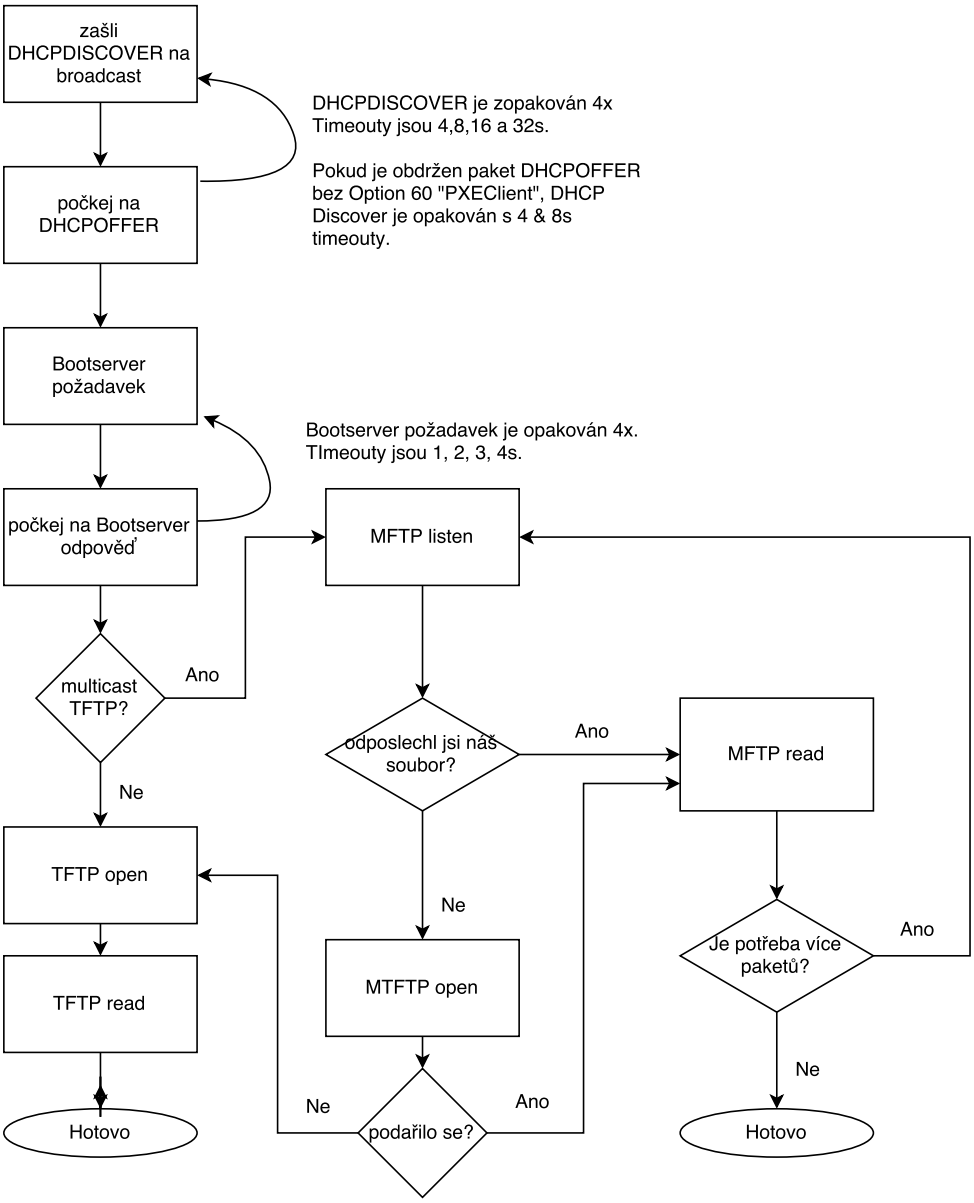
2. Odpoví na standardním DHCP reply portu (UDP 68) paketem DHCPOFFER. Každá zpráva obsahuje klasické DHCP parametry - IP adresu klienta a další možnosti nastavené administrátorem serveru.

3. Z DHCPOFFER paketu si klient zaznamená:

   - IP adresu klienta,
   - seznam Boot serverů z PXE tagu,
   - Discovery Control Options,
   - IP adresu Multicast Discovery.

4. Pokud si klient vybere IP adresu nabídnutou DHCP službou, pak musí dokončit standartní DHCP průběh zasláním požadavku na adresu (paketu DHCPREQUEST) a poté počkáním na potvrzení požadavku od služby – obdržením paketu DHCPACK.

5. klient si vybere a objeví Boot server. Tento objevovací paket může být zaslán na broadcast (port 67), multicast (port 4011) nebo na unicast (port 4011). Toto záleží na nastavení obdržené v předchozím přijatém DHCPOFFER paketu obsahující rozšiřující PXE tagy. Paket je stejný



Obrázek 1.1: Diagram popisující průběh dle PXE

jako počáteční DHCPDISCOVER v kroku 1, jen je organizován jako DHCPREQUEST a obsahuje:

- IP adresu přiřazenou klientovi z DHCP,
- tag pro identifikátor klienta (UUID),
- tag pro klientovo UNDI,

- tag pro klientovu achitekrutu serveru,
- DHCP Option 60 - nastavenou na PXEClient:Arch:xxxxx:UNDI:yyyzzz.

6. Boot server vyšle na unicast klientovi paket DHCPACK na klientův zdrojový port. Tento paket obsahuje:

   - jméno zaváděcího souboru
   - konfigurační parametry MTFTP [1]

7. Klient stáhne spustitelný soubor buď pomocí standardního TFTP (port 69), nebo MTFTP (potom přiřazený port nalezneme v Bootserver ACK paketu). Kam se server uloží záleží na architektuře procesoru klienta.

8. Klient rozhodne, zda je potřebný test staženého souboru. Pokud je test zapotřebí, klient zašle další DHCPREQUEST boot serveru požadující identifikační údaje staženého souboru. Tyto identifikační údaje stáhne a provede test, zda je soubor správný.

9. Konečně, pokud podmínka kroku 8 proběhla pozitivně, PXE klient začne vykonávat kód právě staženého souboru.

1.4 Zavaděč operačního systému

Pomocí výše uvedené série kroků do počítače chceme nahrát určitý kód. Tím kódem je zavaděč systému – pro naše využití bylo využito bootloaderu pxelinux (patřící do kolekce zavaděčů syslinux). Zavaděčem nazýváme krátký kód, uložený v tabulce MBR [2], nebo boot sektoru jednoho z oddílů disku. Účelem zavaděče je, aby do operační paměti počítače nakopíroval další, větší program – tím je např. jádro operačního systému. Následně přeskočí na zkopírovaný kód a tím mu předá řízení počítače.

Zavaděče lze rozdělit do dvou kategorií [**?**]:

1. na primární (first-stage) – jsou přímo obsaženy na hardware počítače. Na IBM-kompatibilních PC se tento zavaděč nazývá BIOS,

2. a sekundární (second-stage), které jsou zavolány primárním zavaděčem. Mezi ně patří právě syslinux, nebo dobře známý GRUB [**?**].

[1] Multicast Trivial File Transfer Protocol
[2] Master Boot Record

1.4.1 Syslinux (pxelinux)

V porovnání s GRUB je syslinux velmi jednoduchým zavaděčem [**?**]. Ve skutečnosti syslinux je kolekcí zavaděčů, každý zavaděč pro jiný účel:

1. syslinux je určen pro bootování ze souborového systému FAT
2. isolinux, pro bootování z ISO 9660 filesystému, který je používán na CD
3. pxelinux je určen pro bootování ze síťového serveru (tedy naše řešení)
4. extlinux – bootování ze souborového systému ext2/ext3

1.5 Automatizace instalace linuxových distribucí

Následující sekce popisuje způsoby instalace linuxových distribucí bez interakce člověka. Bohužel tento proces pro všechny distribuce nebyl unifikován. Je tedy nutné použít více instalátorů v jednom prostředí. Příští strany se budou věnovat instalátorům pro operační systémy Debian (instalátor pod názvem Preseed) a Red Hat Enterprise Linux (Kickstart).

1.5.0.1 Automatizace instalace OS Debian pomocí Preseed

Při instalaci operačního systému Debian a distribucí na něm založených, preseeding (pozn. v překladu do češtiny přednastavení) nám umožňuje odpovědět na otázky při instalaci OS, které bychom jinak byli nuceni vyplnit ručně. Díky tomu můžeme plně automatizovat většinu instalací. Preseeding dokonce obsahuje nějaké funkce, které přes manuální instalaci nejsou dostupné. Celá tato konfigurace je obsah skriptu předložený instalaci. Následující odstavce ukazují, jak skript může být předložen a co by měl obsahovat.

Způsoby preseedingu Existují tři metody, jak skript instalaci předložit. Těmi jsou: initrd, soubor a předložení po síti.

- `initrd` – preseed konfiguraci předáme instalaci už v ram disku; načtení skriptu se provede hned na začátku instalace. Můžeme přeskočit všechny otázky. Podmínkou je soubor `preseed.cfg` uložený v kořeni initrd.
- `soubor` - v tomto případě je konfigurace uložena na bootovacím médiu (tj. CD nebo USB flash disk). Načtení skriptu se provede hned po připojení média, tj. hned po dotazech na direktivy (otázky) o jazyku instalace a rozvržení klávesnice.
- `ze sítě` - načtení preseed skriptu proběhne hned po automatické konfiguraci síťových rozhraní. Boot parametr obsahující řetězec, odkud je soubor stažen, se nazývá `preseed/url=http://server/preseed.cfg`.

V případě této bakalářské práce je využito třetí možnosti, tj. stažení konfiguračního souboru pomocí HTTP protokolu. I když načítání preseed konfigurace z initrd se zdá jako nejzajímavější způsob, generování initrd instalátoru je poměrně komplexní proces.

Preseed soubor Preseed soubor je plain text soubor, ve kterém každý řádek obsahuje jednu odpověď na jednu `debconf` direktivu. Jeden řádek obsahuje čtyři pole oddělené mezerou (nebo tabulátorem). Na příkladu

```
d-i mirror/suite string stable
```

- `d-i` – první pole je tzv. vlastník direktivy; `\uv{d-i}` je značka pro direktivy spojené s instalátorem,
- `mirror/suite` je identifikátor a typ otázky oddělené lomítkem,
- `string stable` jsou jsou datový typ a hodnota odpovědi. Pokud řádek obsahuje další mezeru, považuje se za část odpovědi.

1.5.0.2 Automatizace instalace CentOS/RHEL pomocí Kickstart

Kickstart je způsob automatizované instalace od společnosti RedHat. Kvůli této vazbě je tedy kompatibilní s operačnímy systémy vázanými na tuto společnost, jmenovitě Red Hat Enterprise Linux (zkráceně RHEL), CentOS, Fedora.

Základem Kisktart instalace jsou tři prvky: malý bootovací obraz disku, konfigurační soubor a repozitář s jednotlivými balíčky. Díky PXE instalaci obraz disku a kickstart soubor nemusí být uložen na záznamovém médiu připojeném k serveru. Kickstart nabízí širokou škálu možností, jak si instalaci přizpůsobit vlastní potřebě. Pěkným a užitečným způsobem, jak toto nastavit je jeden plain text soubor (`ks.cfg`).

Soubor obsahuje část s příkazy odkazující na jednotlivé kroky při instalaci. V souboru předáme stejné parametry, které bychom provedli, pokud bychom instalovali interaktivně.

Kickstart soubor `ks.cfg` se skládá z těchto částí:

- nastavení klávesnice a jazyka operačního systému,
- způsob autentifikace,
- diskové rozdělení,
- výběr bootloaderu,
- seznam balíčků, které se na stroj budou instalovat,

- a konečně vlastní příkazy, které se mají provést v okamžiku, co skončí instalace (tzv. post-install skript).

Další ze zajímavých vlastností ks.cfg je částečně automatizovaná instalace, případně aktualizace. Instalátor se poté zeptá jen na otázky, které mu nebyly podsunuty v konfiguračním souboru (mezi toto může například patřit rozdělení disku, které můžeme chtít u každého serveru jiné). Některé možnosti aktualizace operačního systému pokmocí kickstartu jsou ale omezené – například není možné aktualizovat verze balíčků.

Rozdělení disku Následující část pojednává o možnostech kickstart instalace při rozdělení disků. Příkazy níže je možné vložit do konfiguračního souboru (`ks.cfg`).

`autopart` - automaticky vytvoří diskové oddíly - 1 GB a více pro root oddíl (`/`), swap oddíl a boot oddíl podle příslušící architektury. Tento příkaz též přijímá parametry `--encrypted` a `--passphrase=PASS`.

`ignoredisk` - zapříčiní, že disk nebude využit. Parametr poté je `--drives=disk1,disk2,...` Druhým využitím příkazu je parametr `--use-only=disk1`, pomocí kterého určíme, pouze jaké disky máme použít.

`raid` vytvoří software RAID. Struktura příkazu vypadá takto:

```
raid <mntpoint> --level=<level> --device=<mddevice> <partitions*>
```

`Mntpoint` je místo, kde bude vytvořený raidový oddíl připojen. Podmínkou je, že oddíl připojený na `/boot` musí být RAID typu 1 (To samé platí i pro root oddíl, pokud nemáme zvláštní boot oddíl). Parametr level nám říká typ RAIDu (0, 1, 4, 5, 6, nebo 10).

`part / partition` - vytvoří diskový oddíl na disku. K zajímavým parametrům patří `<mntpoint>`, `--fstype` (případně `--noformat`), `--size`.

`bootloader` - specifikuje, jak by měl být bootloader instalován. Pomocí parametru `--location` nastavíme oddíl (případně pozici Master Boot Recordu, pokud nastavíme `--location=mbr`), kde má být záznam zapsán. Parametr `--driveorder` říká, který disk je první v pořadí v nastavení BIOSu.

`clearpart` odstraní všechny diskové oddíly na discích specifikovaných v parametru `--drives`. S pomocí `--initlabel` vytvoříme label standardní k naší architektuře, tedy GPT, či MBR.

sgdisk

parted

1.5.0.3 Disková rozvržení

Před úspěšnému dokončení instalace operačního systému na server si musíme zodpovědět několik otázek. Před kopírováním systémových souborů musí být cílové médium (většinou hard disk) nastaveno. Nastavením je myšlena série kroků, do které patří

- partitioning, neboli vytvoření oddílů – jednotlivé oddíly disků jsou části disku (ať už fyzického nebo logického), se kterými můžeme libovolně nezávisle manipulovat. Diskový oddíl obsahuje svůj vlastní souborový systém, připadně pododdíly,
- naformátování disku, při kterém na oddíl zavedeme souborový systém,
- vytvoření zavaděče disku,
- nakopírování souborů na disk.

1.6 Formáty diskových oddílů

Na počítačích s procesorem z rodiny x86 (IBM PC kompatibilné) je proces bootování obyčejně řešen jedním ze dvou způsobů. Tím prvním a zároveň starším je BIOS-MBR, nebo novější UEFI-GPT. Následující odstavce přiblíží, jak metody fungují.

1.6.1 Standardní proces bootování s MBR

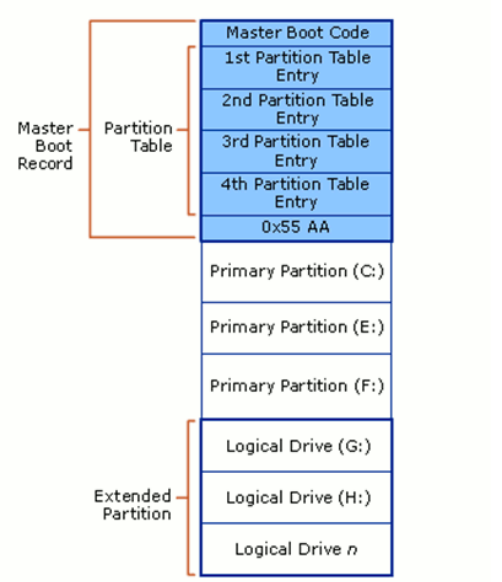
MBR je starým standardem pro rozdělování oddílů disku a stále je ve velké míře používán. MBR (master boot record), se nachází na úplném začátku disku a obsahuje metadata o tom, jak jsou na zařízení logické oddíly organizovány. Master Boot Record též obsahuje spustitelný kód, který je schopen prohledávat oddíly a najít na nich operační systém. A následně spustit spustitelný kód/proceduru operačního systému.

Na MBR disku je možné mít pouze 4 oddíly. Pro vytvoření více oddílů je potřeba nastavit tu čtvrtou jako „rozšířenou“ (tj. „extended“), ve které je možné vytvořit pod-oddíly (též logické disky). Tím, že MBR využívá 32 bitů na adresu oddílu, je maximální velikost jednoho do 2TB. Takto nějak vypadá typické rozdělení MBR:

Jak vidíme na obrázku výše, MBR má několik nevýhod. Vedle již zmíněné maximální velikosti oddílu 2TB a maximálně 4 logických oddílů, to je též fakt, že MBR je jediné místo, kde se nachází informace o diskovém rozdělení. Pokud se tedy někdy poruší, celý disk se stane nečitelným.

1.6.2 Standardní proces bootování s GPT

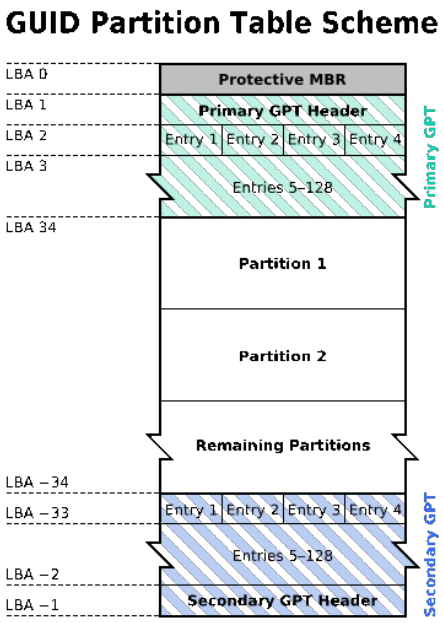
GPT je posledním standardem pro rozdělování oddílů na disku, současně je částí UEFI standardu. Využívá globálně unikátních identifikátorů (GUID), pomocí kterých definuje oddíl. Pro systém založený na UEFI tedy je nezbytné, aby využíval GPT. Je teoreticky možné vytvořit neomezený počet oddílů – na většině operačních systémů je ale toto číslo omezeno na 128. Narozdíl od MBR, které je omezeno na 2TB, jeden oddíl může být velký až $2\hat{6}4$ bloků (64 bit



Obrázek 1.2: MBR rozvržení disku

adresa). Při bloku velkém 512 bajtů je to 9.44 ZB (1 ZB je miliarda TB). Na diagramu 1.3 vidíme rozvržení disku s GPT.

Jak ukazuje obrázek 1.3, primární GPT se nachází na začátku disku a sekundární na konci. To je jedna z funkcionalit, co dělá GPT účinnější než MBR. GPT ukládá záložní hlavičku a tabulku oddílů na konci disku, takže pokud se nějakým způsobem primární poškodí, může být zachráněna a obnovena ze sekundární. Také obsahuje CRC32 opravné součty pro detekování chyb v hlavičce anebo tabulce oddílů. Dále můžeme vidět "Protective MBR"v prvním sektoru disku. Pomocí tohoto hybridního nastavení můžeme počítač s biosem zavést z disku s GPT rozdělením, se zavaděčem uloženým v části "Protective MBR". Tato funkcionalita je též ochranou před poškozením od nástrojů, které o GPT neví.



Obrázek 1.3: GPT rozvržení disku

Kapitola 2

Analýza provisioning frameworků

V této kapitole rozeberu jednotlivé frameworky, pomocí kterých je možné nainstalovat operační systém. Jednou z nutných podmínek je schopnost instalace operačních systémů CentOS a Debian. Ostatní operační systémy jsou výhoda, ale nejsou nezbytné. Kapitola též popisuje stroje, na kterých bude celý experiment testován spolu s jejich konfigurací. Služby jsou konfigurovány tak, aby byly na oddělených virtuálních serverech a nemohly se vzájemně ovlivňovat. Na konci kapitoly vybereme framework Foreman z důvodů níže uvedených.

2.1 Metodika porovnání

Než začnu zkoumání a hodnocení jednotlivých frameworků, je třeba si stanovit hodnotící kritéria, která mi umožní frameworky objektivně porovnávat a vybrat kandidáta pro kapitolu **??**. Pokud to bude alespoň trochu možné, tak pro kritérium stanovím stupnici, na základě které bude možné frameworky mezi sebou porovnat.

Zhodnocení jednotlivých frameworků pro provisioning čistého hardwaru bylo provedeno jak z pohledu kvality, tak i kvantity. Jakmile je framework nastaven a rozvržen, může být složité klientské servery zmigrovat z jednoho na druhý. Kritéria zvolené při porovnáváni jsou následující:

- uzavřenost systému (licence),
- dospělost projektu,
- počet aktivních uživatelů,
- složitost instalace frameworku,
- stabilita,

- složitost údržby,
- hardwarová náročnost,
- počet volitelných vlastností.

Následující podkapitoly přiblíží frameworky ke srovnání. Dále popíši jednotlivé parametry, podle kterých frameworky budeme porovnávat.

2.2 Testovací laboratoř

2.2.1 Hardware

2.2.1.1 Master server

Celý experiment byl testován na Intel x86 stroji se základovou kartou X10SLM-F od výrobce SuperMicro. Na kartě je zapojen BMC modul pro vzdálené ovládání. Další fakta o serveru jsou:

- Intel® Xeon® E3-1200 v3,
- 16 GB RAM,
- 1x 10Gbps Intel GE síťová karta,
- 2x512 GB SSD disky zapojené jako JBOD. [3]

2.2.1.2 Instalovaný stroj

Instalovaný stroj má tyto parametry:

- základní deska X10DDW-i,
- Intel® Xeon® E3-1200 v3,
- 16 GB RAM,
- 1x 10Gbps Intel GE síťová karta,
- 1x 1Gbps Intel XE síťová karta,
- 1x 4TB HDD.

Koncept instalace serverů je navržen tak, že hlavní, tedy 10 Gbit síťová karta je zapojena do podsítě připojené do internetu. IP adresa na této síťové kartě přiřazena staticky na stroji. 1 Gbit port je zapojen do interní sítě oddělené od internetu, na které bude celý proces instalace probíhat. IP adresa je na tomto portu přiřazována pomocí DHCP. Připojení k internetu při instalaci probíhá pomocí http-proxy, její adresa je zaslána serveru v Kickstart/preseed konfiguračním souboru.

[3] Just a bunch of disks - v překladu jen hromada disků

2.2.1.3 Přepínač mezi stroji

gotta redo this text

Jako přepínač mezi stroji byl využit Cisco SF550X-24.

Na portu master serveru je povolen VLAN trunking. Netagované pakety patří do VLANy 1, kdy je povoleno tagovat i 222.

Na instalovaném stroji je konfigurace nasledující:

1Gbps karta do vlany 222 bez trunkování. 10Gbps karta do vlan 1 bez trunkování.

2.2.2 Konfigurace strojů

V následujících podkapitolách popíši, jaká nastavení na serverech byla provedena.

2.2.2.1 Master

Na master stroji budeme všechny naše testované projekty mít v kontejnerech. Přesněji řečeno to nebudou kontejnery, ale budeme využívat KVM (Kernel-based Virtual Machine) virtualizace, díky které můžeme jednotlivé části úplně oddělit od hypervisoru. Toto je míněho hlavně z bezpečnostního hlediska. Jako operační systém, nad kterým budou jednotlivé služby spouštěny je vybrán Debian. Mezi důvodu patří nadstandardní stabilita, jedna z největších komunit v oblasti GNU/Linux a také velký výběr již vytvořených balíčků pro jakékoliv potřeby. Na síťovém portu jsou povoleny VLAN 1 a 222, přičemž pokud pokud ethernetový rámec tag s informací o VLANě neobsahuje, bude nastavena na 1. Tuto konfiguraci sítě pro operační systém Debian vidíme níže:

```
# /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto br0
iface br0 inet dhcp
bridge_ports enp5s0
up /usr/sbin/brctl stp br0 off

auto enp5s0.222
iface enp5s0.222 inet static
        vlan-raw-device enp5s0

auto br1
iface br1 inet static
        address 192.168.4.4
```

```
netmask 255.255.255.0
bridge_ports enp5s0.222
up /usr/sbin/brctl stp br1 on
```

Jak vidíme výše, rozhraní br0 je VLAN 1 (dle nastavení na přepínači) a br1 je VLAN 222. Tato terminologie bude využívána i v následujícím textu.

2.3 Testované frameworky

Kapitola níže popíše frameworky, které jsme k naší analýze vybrali.

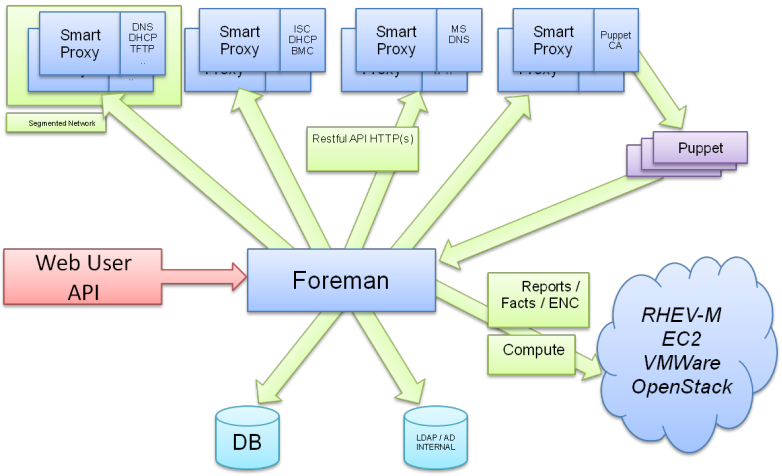
2.4 Foreman

Tento open source projekt spatřil světlo světa v roce 2009, kdy byl založen programátory Ohadem Levym a Paulem Kellym. Foreman je komplexní nástroj pro správu celého životního cyklu jak hardhare strojů, tak těch virtuálních. Mezi hlavní části a funkce patří: provisioning, configurace serveru a poté jeho monitoring. Umožňuje automatizaci úkolů, které se opakují během prvotního nastavení infrastruktury. Dále Foreman nabízí správu konfigurací serverů, následovaných monitorováním a zobrazování trendů ze získaných dat. Celý projekt je zaštítěn společností RedHat a jako jazyk byl vybrán Ruby on Rails. Databázi, do které budou ukládána data, si můžeme při instalaci vybrat -patří mezi ně PostgreSQL, SQLite, MySQL a další. Foreman se dá také provozovat na jiných databázích, ovšem bez oficiální podpory.

Během provisioningu, Foreman z velké části závisí na Smart Proxy, která je ve výchozím nastavení nainstalována na stejný server, jako Master uzel. Smart proxy hraje roli prostředníka v kominikaci mezi Foremanem a externími službami. V současnosi Smart Proxy obsahuje podporu pro služby: TFTP, DNS, DHCP, Puppet & Puppet CA. Další služby je možné nainstalovat pomocí pluginů. Na obrázku XXX můžeme vidět komunikaci mezi Foremanem, Smart Proxy a službami.

Provisioning serveru s námi vybraným operačním systémem a požadovanou konfigurací je několika krokový proces. Prvně jě třeba vybrat operační systém s přiřazeným instalačním médiem. Foreman již má několik operačních systémů založených na UNIXu v sobě nakonfigurován a připraven přímo k instalaci na stroje. Dalším krokem je vybrání rozdělení disků a šablony pro provisioning (pro příklad Kickstart nebo Preseed).

Když náš nový server nabootuje pomocí PXE protokolu, vyšle broadcast zprávu pro DHCP server schopný přijímat DHCP žádosti. Smart Proxy pracující jako DHCP server odpoví a přidělí IP adresu novému hostu. PXE server je kontaktován a přesměruje nový stroj na TFTP server obsahující bootovací obraz disku. Nový stroj se pokusí o získání obrazu, spustí ji a začne instalaci s parametry, které byly obsažené v šabloně ve Foremanu. Následovně, pokud



Obrázek 2.1: Architektura Foremanu

je tak povoleno, proběhne jeden běh Puppetu. Foreman spoléhá na službě Puppet pro správu konfigurací a pro sbírání faktů o serverech. Celý proces je zobrazen na obrázku XXXX.

==== obrazek ===

Foreman je schopen zavést široké spektrum operačních systémů založených na UNIXu. Mezi operační systémy, které se podle diskusních fór povedlo nainstalovat, patří: RHEL, Fedora, CentOS, Ubuntu, Debian, Solaris 8 and 10, OpenSUSE. Každopádně oficiální seznam operačních systémů, které Foreman podporuje, je poněkud kratší. Patří mezi ně: RHEL 6 and 7, CentOS 6 and 7, Fedora 19, Debian 7, Ubuntu 12.04 and 14.04 CITE THIS.

Foreman nabízí grafické rozhrací pro jednoduché využívání i méně schopnými uživateli. Je též dostupné RESTful API. Další variantou ovládání Foremanu je jeho nástroj pro ovládání z příkazové řádky - zvaný Hammer. Pomocí něj je možné jednotlivé dílčí kroky skriptovat. Foreman je jednoduše rozšiřitelný pomocí Rails Engines. To ukazuje široká škála již dostupných rozšíření. Mezi ně patří například rozšíření Azure, Digital Ocean a další - pro přímý provisioning virtuálních serverů ve zmíněných cloudech.

2.4.1 Foreman Smart Proxy

Jádrem Foremanova řídícího systému pro vzdálené části infrastruktury jsou Smart Proxy, což jsou malé a modulární aplikace vytvořené v Ruby. Zjednodušeně, pokud něco spravovat, spustíme jednu z těchto ruby aplikací v místě, kde potřebujeme. Například: Místo toho, abychom manuálně zašli k DHCP serveru, podívali se na zápůjčky IP adres, přímo se pomocí REST API DHCP serveru zeptáme na volnou IP adresu, kterou nám DHCP server vrátí. Kód těchto proxy serverů je cíleně velmi krátký - okolo 1000 řádků, díky čemuž

není tak těžké jednotlivým částem porozumět a dopsat libovolný modul.

Mezi tyto smart proxy moduly patří:

- DHCP, DNS, TFTP
- BMC/IPMI
- Puppet, Puppet CA
- MCollective

Foreman obsahuje koncept zvaný Compute Resources. Tento nápad cílí na provisioning virtuálních serverů (také bare metal) na veřejných a soukromých cloudech. Podporuje tedy například VMWare infrastrukturu, nebo EC2.

2.5 Razor

Provisioning like its 1999

2.6 Stacki

Stacki je nabízen ve dvou základních plánech. V plánu zdarma, ten ale nenabízí webové rozhraní. Placený plán sice přes webovou stránku ovládat lze, skrývá ale v sobě jiná omezení. Mezi tato omezení paří například nemožnost instalace distribucí založených na Debianu (tedy i Ubuntu, které je jedním z našich požadavků), dále neobsahuje podporu pro UEFI. UEFI v našem případě problém není, protože SuperMicro základové karty v nastavení umožňují změnu na Legacy Boot (BIOS), absence podbory Debian distribucí je ale pro nás závážná, až klíčová.

2.6.1 Edice zdarma

Základní verzi Stacki můžeme nalézt zdarma, bohužel je ořezaná o určité vlastnosti. Jednou z možností je stažení rpm balíčku pro CentOS. Po instalaci balíčku dostaneme skript, který server přetransformuje na Stacki frontend. Druhou možností je, stejně jako u placené verze, připraný ISO obraz pro instalaci na stroj bez operačního systému.

2.6.2 Pro plán

Placená edice Stacki softwaru je nabízena ve 14-ti denní zkušební verzi, tu jsem také pro tento experiment použil. Po registraci na webu produktu je odeslán e-mail s odkazem na stažení iso souboru. Obraz je ve své podstatě CentOS linux (můžeme si vybrat CentOS 6, či 7) přípravený pro vypálení na CD, nebo zkopírování na USB disk pro následovné zavedení systému a instalaci

na čistý stroj. Tento postup je odlišný od ostatní opensource konkurence, jež nabízejí svoje provision frameworky většinou jako balíčky pro linuxové disribuce. Cena placené edice je 100 USD/rok na jeden uzel, hlavní výhodou je již zmíněná podpora UEFI a možnost instalace Ubuntu a v neposlední řadě placená podpora.

2.7 OpenStack Ironic

OpenStack je škálovatelná, open-source platforma pro stavbu veřejných či privátních cloudů. Jako distribuční model převažuje IaaS, česky přeloženo jako "Infrastruktura jako služba". V takovémto modelu se poskytovatel služeb zavazje poskytnout infrastrukturu - ať už virtualizované, tak i hardware servery. Příklady komerčních IaaS jsou např. Amazon Web Services, nebo Microsoft Azure. OpenStack je stavěný na velké výpočetní, datové i síťové zdroje v datacentrech, u čehož je vě ovládáno přes webové rozhraní.

Všechny OpenStack služby mezi sebou komunikují na bázi RESTful API, k tomuto využívají HTTP protokol pro výměnu informací a dat. Jednou komponentou je právě Ironic pro instalaci bare metal serverů.

OpenStack je orientován na obrovské clustery. Věřím, že v určitých případech může být správnou volbou i pro provisioning, v našem případě ale taková volba připadá jako kolos. Už jen instalace celého frameforku na jeden systém (v mém případě vybraném DevStacku) trvala něco přes půl hodinu. Z důvodů nevhodnosti pro naše využití jsem se s testováním tohoto frameworku dále nezabýval.

2.8 Spacewalk

Spacewalk is an open source based Linux server administration tool. It is registered under the open source license GPLv2. Spacewalk is community driven software from which few commercial products as Red Hat Satellite and Novell SUSE Manager have derived. Spacewalk started in June 2008 when it became officially an open source project. It is based on Red Hat Network which was started in 2001 and it later spawned into a stand-alone Red Hat Satellite product. [1.]

Spacewalik je open source nástrojem pro administraci Linux serverů. Licencí, pod kterou je distribuován, je GPLv2. Spacewalk je

pacewalk has many features that allow Linux administrators to manage hardware and software information, install and update systems, manage systems as a group, provision systems, configuration file management and deployment and monitoring. Spacewalk works with virtualization platforms such as VMware and Xen to create and configure virtual guests and manage systems efficiently in multiple geographical locations. Spacewalk is managed through a web interface where all the information about servers is located. Adminis-

tration software supports many Linux distributions such as Fedora, CentOS, SLES and Debian with both versions of architecture, 32-bit and 64-bit. [1.]

Spacewalk obsahuje mnoho funkcí nabízejících administrát

2.8.1 Licence

V této práci se budu zabývat pouze open-source frameworky, tedy pokud nějaký framework chceme zvažovat, musíme nejdříve zjistit, zda nám to licence projektu dovoluje. Licence frameworku může ovlivnit licenci díla, kde framework použijeme (tedy i tuto bakalářskou práci). Důvody pro open-source jsou zřejmé:

nulové počáteční náklady - získáme zdarma produkt, krerý by jinak společnost vyvíjela několik měsíců. Díky tomu je ušetřeno na počátečních nákladech,

bezpečnost - oprava 0 Day [**?**] zranitelností a dalších chyb je díky komunitě skoro okamžitá,

žádné proprietární uzamčení a lepší kvalita kódu.

Licence se dají rozdělit podle stupně přísnosti. Níže uvedené rozdělení je od nejméně přísného až po nejstriktnější:

- Public Domain,
- permisivní,
- LGPL,
- copyleft,
- AGPL.

Všechny analyzované frameworky se nacházejí pod určitou open-source licencí, nebo alespoň mají ekvivalentní open-source edici. OpenStack Ironic, Foreman, Cobbler i MaaS jsou kompletně open-source a zdarma.

Tabulka 2.1: Frameworky: License

-	Foreman	Ironic	Razor	Stacki	Spacewalk	Cobbler

2.8.2 Stáří projektu

„Nedospělý“, či netestovaný kód, který se může jednoduše rozbít, přímo koreluje se stářím frameworku. Je proto důležité se ujistit, že projekt, který vybereme bude použitelný a dospělý. V naší analýze je dospělost spočítána pomocí počtu let, jak dlouho je framework vyvíjen.

Počet uživatelů či počet nasazení systému nám může ukázat určité náznaky, jako jaká je populárnost systému u určité společnosti a které důvody se za tím skrývají. Tato hodnota, aby byla přesná, není jednoduše zjistitelná. Určité metriky jsou ale veřejně dostupné – mezi ty patří například počet forků na GitHubu, počet uživatelů v mailing listu, počet velkých firem, které se rozhodly pro nasazení frameworku (zde vycházím z předpokladu, že ve velké společnosti tým na takový úkol bude mít větší počet pracovníků a tím i přispěvatelů do projektu). Tyto hodnoty nám můžou nastínit určitý odhad, kolik nasazení bylo. Toto nám také v určité míře ukazuje, jaká rychlost bude při opravách různých chyb, či rychlost přidávání dalších funkcí.

Tabulka 2.2: Frameworky: Stáří projektu

-	Foreman	Ironic	Razor	Stacki	Spacewalk	Cobbler

2.8.3 Složitost instalace

Dalších z hlavních bodů, které musíme při výběru frameworku zvažovat, je složitost instalace takovéhoto systému pro provizi serverů. Jedním

2.8.4 Stabilita

2.8.5 Složitost užití

Všechny analyzované frameworky mají webové grafické rozhraní, pomocí kterého je možné servery provizovat. Díky tomu by měla být uživatelská přívětivost na slušné úrovni. Výjimkou je Razor, u kterého je grafické rozhraní pouze placené verzi (tj. Puppet Enterprise). Objevil jsem grafické rozhraní [**?**], které by zmíněný problém mělo vyřešit, není ale součástí analýzy.

2.8.6 HW požadavky

Dříve, než začneme instalovat jakýkoliv software, musíme se ujistit, zda hadware, na který program chceme nainstalovat, je dostatečný. I přes to, že hardware dostupný pro experiment se zdál dostatečný, v případě instalace Ironic byl nedostatečný. Pro jeho provozování je třeba sestavit celý cluster – pro experiment tedy místo OpenStacku byl vybrán DevStack [**?**], který bohužel svojí rychlostí neoslnil. Všechny ostatní frameworky s poskytnutým výpočetním výkonem a pamětí problém neměly.

2.8.7 Podpora discovery

Je důležité, aby vybraný framework podporoval tzv. discovery – tedy objevení a zjištění základních informací o serverech, které které jsou nové a ještě v data-

bázi nejsou. Většina frameworků má tuto funkcionalitu vytvořenou způsobem linuxové image, která se načte místo operačního systému na neznámých strojích. Tento malý linux si poté pokusí nastavit připojení k síti, DNS, a pokud se zdaří, odešle frameworku zpět informace o novém serveru. Mezi informace může patřit například počet procesorů, velikost paměti, nebo typ síťové karty. Ve frameworku Foreman se tato funkcionalita nazývá Discovery [**?**], v Razoru zase EL Microkernel [**?**]. Tabulka 2.3 ukazuje, zda testovaný nástroj objevování nových strojů podporuje.

Tabulka 2.3: Frameworky: Podpora funkcionality discovery

-	Foreman	Ironic	Razor	Stacki	Spacewalk	Cobbler

2.8.8 Podpora operačních systémů

Jedním z prvních požadavků bylo, že daný framework bude schopný instalovat několik různých operačních systéml vedle sebe. Například MaaS podporuje pouze Ubuntu – je vydáván společností Canonical (společnost zaštitující Ubuntu). Proto mnoho funkcionalit MaaS je již zakomponováno do Ubuntu. Cobbler i Ironic mají podporu instalace pouze linuxových distribucí. Foreman zvládne jak různé linuxové distrubuce, tak BSD systémy, i Windows. V tabulce 2.4 jsou vidět frameworky a jejich podporu instalovaných operačních systémů:

Tabulka 2.4: Frameworky: Podpora operačních systémů

-	Foreman	Ironic	Razor	Stacki	Spacewalk	Cobbler

2.9 Závěr

V této kapitole popíši a shrnu jednotlvé frameworky v jejich silných i slabých bodech podle poznatků, které jsem získal při testování systémů a podle posbíraných statistických dat.

2.9.1 Foreman

Z výše testovaných frameworků po analýze byl vybrán Foreman. Hlavní výhodou bylo jeho grafické rozhraní, pomocí kterého může administrátor vykonat reinstalaci serveru na jakémkoliv zařízení s webovým prohlížečem. To samé

se týká i serverového technika, který je schopen práci vykonat na svém telefonu, stojící vedle serveru v serverovně. Jeho instalace byla v porovnání s ostatními frameworky stejně obtížná, z důvodu, že implicitní konfigurace není ideální a je třeba systém hluboce nastudovat, pokud ho nevyužíváte na uživatelské úrovni. Výhodou je už vytvořené balížky pro master server, Smart Proxy, i nějaké pluginy do operačního systému Debian, což konkurence v podobě Spacewalk nenabízí. Mezi další důvody mého výběru Foremanu patří jedna z největších komunit v této oblasti a také časté vydávání nových verzí. Nasazením frameworku se věnuje příští kapitola.

2.9.2 Ironic

2.9.3 Ostatní frameworky

V této části práce následuje přehledná tabulka popisující, jak si frameworky v jednotivých kategoriích vedly. Podrobnější tabulky o každém kritériu byly uvedeny v příslušných kapitolách.

Tabulka 2.5: Srovnání frameworků

-	Foreman	Ironic	Stacki	Razor	Cobbler
Uzavřenost systému					
Dospělost projektu					
Počet aktivních uživatelů					
Složitost instalace frameworku					
Stabilita					
Složitost údržby					
Podpora discovery					
Hardwarová náročnost					
Počet features					

Tabulka 2.6: Informace o frameworcích

Framework	Webová stránka	Testovaná verze
Foreman	https://www.theforeman.org/	1.14.3
Ironic	https://wiki.openstack.org/wiki/Ironic	1:5.1.2-0ubuntu1
Razor	https://github.com/puppetlabs/razor-server	
Stacki	https://github.com/StackIQ/stacki	
Spacewalk	https://spacewalk.redhat.com/	
Cobbler	http://cobbler.github.io/	2.8.0

Kapitola 3

Nasazení Foremanu

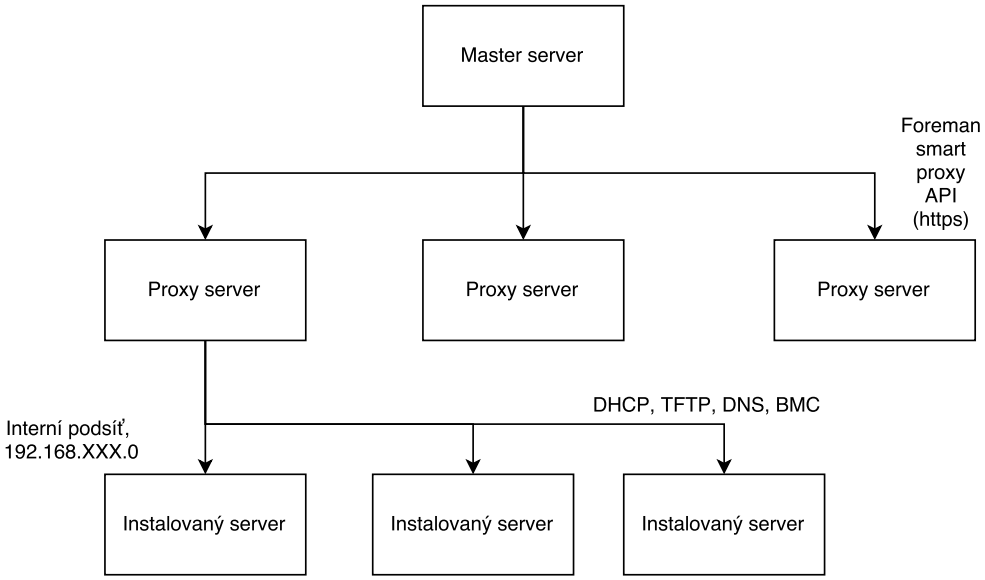
Cílem této kapitoly je popsání nasazení frameworku Foreman, který jsme vybrali v minulé kapitole. V první části popíši topologii infrastruktury. Dále budou postupně popsány jednotilivé servery a služby, které na nich běží. Na konci kapitoly je popsán Ansible a jeho playbooky, pomocí kterých je možné infrastrukturu jednoduše nasadit.

3.1 Topologie infrastruktury

Jedním ze stavebních kamenů virtualizace je vymezení běhu operačního systému do virtualizovaného prostředí, přičemž jsou mu poskytnuty částečné či plné zdroje virtualizačním nástrojem. Díky tomu můžeme na jednom fyzickém stroji provozovat více virtuálních počítačů. Dává nám to možnost lépe a efektivněji využít výpočetního výkonu stroje (tzv. hypervisoru) s dalšími výhodami, mezi které patří např. možnost hostování více operačních systémů na jednom serveru. Výhodou virtualizovaných prostředí je dozajista oddělenost a jednoduché znovunasazení instancí. V našem případě budeme jednotlivé části infrastuktury virtualizovat pomocí KVM.

3.1.1 KVM

KVM je virtualizační infrastruktura pro linuxové jádro, které ho přemění na hypervisor. Aplikaci začala vyvíjet společnost Qumranet, později odkoupená RedHatem. V linuxovém kernelu je obsaženo od verze 2.6.20. Podporovaná je pouze plná virtualizace, plnou virtualizací ale lze spouštět i proprietární operační systémy, jako macOS, Windows, Solaris. Ke své funkčnosti vyžaduje nejméně podporu virtualizace procesoru první generace a to buď procesory od společnosti AMD pod označením AMD-VTM(1) či od společnosti Intel s registrovanou ochranou známkou IntelR VT.



Obrázek 3.1: Infrastruktura

3.2 Master server

Master server je instalován na operačním systému Debian, na serveru virtualizovaném pomocí infrastruktury KVM. Hlavní službou je nástroj Foreman, jehož instalace je dále popsána.

3.2.1 Foreman

Před samotnou instalací je třeba přidat repozitáře pro balíčkovací utilitu apt-get, jelikož verze chtěných aplikací nemusí být v oficiálních repozitářích aktuální.

```
echo "deb http://deb.theforeman.org/ jessie 1.14" > /etc/apt/sources.list.d/foreman.
echo "deb http://deb.theforeman.org/ plugins 1.14" >> /etc/apt/sources.list.d/forema
apt-get -y install ca-certificates
wget -q https://deb.theforeman.org/pubkey.gpg -O- | apt-key add -
```

Poté co jsou přidány repozitáře je možné nainstalovat `foreman-installer`.

```
apt-get update && apt-get -y install foreman-installer
```

Instalátor je možné spustit buď v interaktivním módu, kdy je uživatel dotázán na nastavení na obrazovce. Druhou možností je vyplnění konfiguračního souboru `/etc/foreman-installer/scenarios.d/foreman-answers.yaml`, ze

Tabulka 3.1: Podpora diskových rozdělení

	Kickstart	Preseed
NO RAID	GPT & MBR	GPT & MBR
RAID1	GPT & MBR	GPT & MBR
RAID0	GPT & MBR	GPT & MBR

kterého si instalátor sebere odpovědi automaticky. V příloze práce je tento soubor vyplněn. (Vytvořen jako šablona v Ansible).

V interaktivním módu je vygenerováno heslo pro uživatele admin, pomocí kterého je možné se přihlásit do uživatelského webového rozhraní. Pokud použijeme Ansible z přílohy práce, je třeba toto heslo v konfiguračním souboru `defaults/defaults.yaml` upravit a to bude aplikováno.

3.2.1.1 Foreman Discovery plugin

3.2.1.2 Disková rozdělení

V teoretické části práce byly popsány výhody a nevýhody formátování oddílů disku pomocí MBR či GPT. V instalovaných serverech touto infrastrukturou se bude používat MBR pro malé disky (do 1TB) a to z důvodu kompatibility základních desek. Při větších kapacitách již využijeme GPT rozdělení z důvodů popsaných v teoretické části práce.

Práce obsahuje šablony pro operační systémy Debian a CentOS (a operační systémy z nich odvozené) a to pro:

3.2.2 Zálohování

V případě této práce je v konfiguraci užita relační databáze PostgreSQL. V databázi jsou uloženy informace o serverech, statistiky a metadata Foremanu. Díky tomu, že krátký čas odstávky vadit nebude, je možné si dovolit udělat zálohu databáze metodou dump. Obecně dump je soubor, který obsahuje strukturu tabulek a dále data obsažená v těchto tabulkách. Dle rady v manuálu [**?**] dále zálohujeme adresáře

- `/etc/foreman` obsahující všechny konfigurační soubory Master Foremanu
- `/var/lib/puppet/ssl` – obsahující certifikáty

Celé zálohování blíže popisuje shellový skript uložený v `/usr/local/bin/foreman-backup.sh`:

```
#!/usr/bin/env bash

# Backup The Foreman, following the advice at
# http://theforeman.org/manuals/1.7/index.html#5.5.1Backup
```

```
set -e # If any command fails, stop this script.
ME=$(basename $0)
main () {

  DATE=$(date '+%Y%m%d.%H%M')
  BACKUPDIR=/data/backups/backup-$DATE
  mkdir $BACKUPDIR
  chgrp postgres $BACKUPDIR
  chmod g+w $BACKUPDIR

  cd $BACKUPDIR

  # Backup postgres database
  su - postgres -c "pg_dump -Fc foreman > $BACKUPDIR/foreman.dump"

  # Backup config ifles

  tar --selinux -czf $BACKUPDIR/etc_foreman_dir.tar.gz /etc/foreman
  tar --selinux -czf $BACKUPDIR/var_lib_puppet_dir.tar.gz /var/lib/puppet/ssl
  tar --selinux -czf $BACKUPDIR/tftpboot-dhcp.tar.gz /var/lib/tftpboot /etc/dhcp/ /v

  ls -lh *tar.gz foreman.dump

}

main 2>&1 | /usr/bin/logger -t $ME
```

Tento skript je spouštěn každý den ve 2 hodiny ráno. Opakované spouštění je řešeno pro UNIX klasickým způsobem, tedy pomocí CRON démona. Řádek v CRON tabulce pro uživatele root (otevřeme pomocí příkazu `crontab -e`) vypadá následovně:

```
2 0 * * *  /usr/local/bin/foreman-backup.sh
```

Následovně ještě celý filesystem serveru zálohujeme pomocí utility backuppc [**?**]. Jediné co je pro klienta potřebné spustit, je rsync v démon módu. Nainstalujeme tedy rsync a vytvoříme služby pro systemd.

Je potřeba vytvořit soubor `/etc/systemd/system/rsyncd.socket` s obsahem:

```
[Unit]
Description=Rsync Server Activation Socket
ConditionPathExists=/etc/rsyncd.conf
```

```
[Socket]
ListenStream=873
Accept=true

[Install]
WantedBy=sockets.target
```

Soubor se systemd službou, nacházející se v `/etc/systemd/system/rsyncd@.service`:

```
[Unit]
Description=Rsync Server
After=local-fs.target
ConditionPathExists=/etc/rsyncd.conf

[Service]
# Note: this requires /etc/rsyncd.conf
ExecStart=/usr/bin/rsync --daemon
StandardInput=socket
```

Dříve, než můžeme tuto službu použít, je ještě potřeba vytvořit soubor `/etc/rsyncd.conf`. V nejjednodušší konfiguraci bude obsahovat:

```
lock file = /var/run/rsync.lock
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid

[foreman]
    path = /home/foreman
    uid = rsync
    gid = rsync
    read only = no
    list = yes
    auth users = rsyncclient
    secrets file = /etc/rsyncd.secrets
    hosts allow = 192.168.1.0/255.255.255.0
```

Soubor `/etc/rsyncd.secrets` obsahuje uživatelská jména a hesla oddělená dvojtečkou. Následovně je třeba nastavit na souboru správná přístupová práva, aby ostatní uživatelé nebyli schopni soubor číst, či nijak upravovat.

```
# chmod 600 /etc/rsyncd.secrets
```

Tabulka 3.2: Otevřené porty na hlavním uzlu

Port	Protokol	Potřebné pro
53	TCP & UDP	DNS Server
67, 68	UDP	DHCP Server
69	UDP *	TFTP Server
443	TCP * HTTP & HTTPS	přístup do Foreman UI / provisioning templates - using Ap
3000	TCP HTTP	přístup do Foreman UI / provisioning templates - using sta
3306	TCP	v případě oddělené PostgreSQL databáze
5910 - 5930	TCP	Server VNC Consoles
5432	TCP	v případě oddělené PostgreSQL databáze
8140	TCP	Puppet Master
8443	TCP	Smart Proxy, otevřená pouze pro Foreman

3.2.3 Zabezpečení serveru

Jedním z prvků zabezpečení je nastavení firewallu na serveru. Protože používáme operační systém Debian, jako firewall máme k dispozici iptables. Na serveru zakážeme všechny porty mimo:

Dalším krokem je vynucení SSL při připojení do webového rozhraní Foremanu. V posledních letech byl představen projekt Let's Encrypt pod záštitou neziskové instituce Internet Security Research Group (ISRG). Let's Encrypt otevřená certifikační autorita (CA), nabízející digitální certrifikáty potřebné k HTTPS (SSL/TLS) zdarma.

V našem případě, kdy pro frontend využíváme webový server Apache (httpd) je vytvoření a obnova certifikátu velmi jednoduchá. Před vygenerováním certifikátu je nutné nastavit DNS A záznam tak, aby mířil na IP adresu přiřazenou našemu serveru. Tento A záznam musí být stejný jako hostname nastavený na serveru. Postup instalace certifikátu a jeho obnovení je následovný:

Nainstalujeme balíček s názvem certbot:

```
# apt-get install python-certbot-apache
```

Aktuálně už můžeme certbot klient použít. Vygenerování SSL certifikátu pro Apache je poměrně přímočaré. Klient automaticky získá a nainstaluje nový SSL certifikát pro validní domény v Apache konfiguraci.

Interaktivní instalaci certifikátů vyvoláme pomocí příkazu:

```
# certbot --apache
```

Certbot projde získanou Apache konfiguraci a najde domény, které by měli být v žádosti o certifikát obsaženy. Interaktivně je možné jakoukoliv doménu odebrat. Utilita se dotáže na email potřebný při ztrátě klíče a dá nám na výběr ze dvou možností:

- povolený http a https přístup, pokud je nějaká potřeba pro nezašifrovanou komunikaci,
- nebo přesměrování z http na https, které my využijeme.

Když instalace proběhne, vytvořené certifikáty spolu se soukromým klíčem jsou k dispozici v adresáři `/etc/letsencrypt/live`.

Zda je certifikát validní je možné zjistit například pomocí následujícího odkazu:

```
https://www.ssllabs.com/ssltest/analyze.html?d=example.com\&latest
```

Certifikáty vydávané autoritou Let's Encrypt jsou validní na 90 dní, proto je potřeba je po uplynulé době obnovovat. Doporučená doba k tomu je 60 dní. Aplikace ertbot obsahuje parametr renew, pomocí kterého certifikát obnovíme.

Praktickým způsobem, jak na linuxovém stroji vykonávat nějakou činnost opakovaně, je cron démon. Protože obnovovací příkaz zjišťuje pouze datum expirace a obnoví certifikát pouze pokud do expirace zbývá méně, než 30 dní, není problém tuto činnost spustit například jednou týdně.

Úpravu CRON tabulky uživatele root provedeme následovně:

```
# crontab -e
```

Příkaz otevřel soubor `/var/spool/cron/crontabs` pomocí našeho editoru. Do souboru přidáme řádek a uložíme:

```
30 2 * * 1 /usr/bin/certbot renew >> /var/log/le-renew.log
```

Tato část automaticky spustí aplikaci `/usr/bin/certbot renew` každé pondělí ve dvě ráno, v době, kdy by server neměl být tak těžce zatížen. Informace o průběhu se uloží do logu v adresáři `/var/log`.

3.3 Proxy server

3.3.1 Nastavení sítě

Tento virtuální server má přístup ke dvoum odděleným sítím. První z nich je připojena do internetu (toto síťové rozhraní nazvěme eth0) a má IP adresu přiřazenou z veřejného rozsahu. Adresa je nastavena staticky na stroji. Druhá síť je interní. Na tomto proxy serveru adresu přiřadíme staticky, u dalších strojů se adresa přiřadí pomocí DHCP (DHCP server bude spuštěn na tomto serveru).

Zde definujeme rozsah sítě pro každou lokalitu:

```
10.22.X.0/24,
```

kde X je číslo lokality, ve které se proxy server nachází. Maska podsítě `/24` nám říká, že síť bude schopna obsáhnout 255 počítačů. Proxy server v lokalitě č. 1 tedy bude mít vlastní adresu `10.22.1.1/24`.

3.3.2 Foreman Smart Proxy

Smart Proxy je projektem nabízejícím restové API jednotlivým subsystémům. Je tedy rozhraním mezi automatizačními nástroji vyšší úrovně (např. Foreman) a službami nižší úrovně (jako je DHCP, DNS, TFTP, atd.).

Pro instalaci balíčku v operačním systému Debian je třeba přidat repozitáře Foremanu z adresy `deb.theforeman.org`. [**?**] Postup přidání repozitářů je stejný jako u master serveru, proto zde není popsán. Poté nainstalujeme balíček pomocí `apt-get` a aplikaci nastavíme. Jednou z klíčových částí konfigurace je nastavení komunikace s Foreman master serverem pomocí SSL. Toto je blíže popsáno v kapitole Zabezpečení serveru.

3.3.3 konfigurace DHCP

Smart proxy potřebuje pro svou funkci správně fungující PXE stack, do kterého patří i DHCP server. Osobně jsem využil serveru od ISC, kvůli obrovské uživatelské podpoře a celkové vyladěnosti. V repozitářích Debianu lze nalézt pod jménem `isc-dhcp-server`. Ukázková konfigurace je níže:

```
omapi-port 7911;
max-lease-time 86400;

option domain-name-servers 8.8.8.8;

allow booting;
allow bootp;

option fqdn.no-client-update    on;  # set the "O" and "S" flag bits
option fqdn.rcode2            255;
option pxegrub code 150 = text ;

# PXE Handoff.
next-server SERVER_IP;
filename "pxelinux.0";

log-facility local7;

include "/etc/dhcp/dhcpd.hosts";

subnet 192.168.2.0 netmask 255.255.255.0 {
```

```
range 192.168.2.10 192.168.2.255;
  option subnet-mask 255.255.255.0;
  option routers SERVER_IP;
```

Výše vidíme nakonfigurované direktivy. Lease time (čas propůjčky IP adresy) můžeme nastavit libovolně. Povolení bootp a booting direktiv je nutná podmínka k funkčnosti PXE. Řádka `filename "pxelinux.0"`; značí soubor, který bude instalovanému stroji podsunut. Lokace souboru na proxy serveru je v adresáři `/var/lib/tftpboot`. Implicitně se soubor nachází v adresáři `/usr/share/syslinux/`, odkud je možné ho přesunout. Též je třeba, aby adresář obsahoval soubor `menu.c32`. Nastavení podsítě je znovu libovolné, zde nastaveno podle kapitoly XX. Proměnná `SERVER_IP` obsahuje IP adresu Smart Proxy.

3.3.4 konfigurace TFTP

Pro nastavení TFTP serveru využiji balíčku xinetd. Jeho nastavení je poměrně triviální; konfigurace TFTP se nachází v adresáři `/etc/xinetd.d/tftp`:

```
service tftp
{
 protocol = udp
 port = 69
 bind = SERVER-IP-BIND
 socket_type = dgram
 wait = yes
 user = nobody
 server = /usr/sbin/in.tftpd
 server_args = /srv/tftp
 disable = no
}
```

3.3.5 Zabezpečení serveru

Komunikaci mezi Smart proxy a hlavním uzlem vykonáváme přes zabezpečené spojení SSL. Prvním krokem k tomu je vytvoření certifikátu podepsaným Master serverem. Na hlavním uzlu tedy vykonáme příkaz (parameter SERVER-HOSTNAME je FQDN [4] nově vytvořeného proxy serveru):

```
$ /opt/puppetlabs/bin/puppet cert --generate SERVER-HOSTNAME
```

Výstupem příkazu jsou 3 soubory:

[4] Fully Qualified Domain Name; FQDN přesně určuje umístění počítače ve stromové struktuře DNS.

- certifikát nově vytvořené proxy
- soukromý klíč nově vytvořené proxy
- certifikát hlavního uzlu

Tyto soubory přesuneme na proxy server a po změníme čtecí práva pouze pro uživatele foreman-proxy.

Druhým bodem zabezpečení serveru je firewall. Stejně jako na hlavním uzlu máme operační systém Debian, použijeme tedy iptables.

Povolené porty jsou následující:

Tabulka 3.3: Otevřené porty na proxy serveru

Port	Protokol	Potřebné pro
53	TCP & UDP	DNS Server
67, 68	UDP	DHCP Server
69	UDP	TFTP Server
5910 - 5930	TCP	Server VNC Consoles
8140	TCP	Puppet Master
8443	TCP	Smart Proxy, otevřená pouze pro Foreman

3.4 Collectd plugin

Cílem našeho rozšíření je mít v grafickém rozhraní systému Foreman jednoduché grafy, které zobrazují informace o nainstalovaných serverech. Mezi tyto informace patří například vytížení procesoru, využití paměti, nebo přenos dat na síťovém rozhraní. Takových systémů na sběr dat existují stovky, proto v rozsahu bakalářské práce nemá smysl vyvíjet další. Na pomoc jsem si vzal démon na sběr statistických dat, collectd. Nabízí mnoho doplňků, díky kterým můžeme o serverech sbírat skoro jaékoliv informace nás napadnou.

3.4.0.1 bootstrap na nově instalovaných serverech

Na serverech, které jsou přes foreman instalovány se po prvotní provizi operačního systému nainstaluje collectd a poté nakonfiguruje v klient režimu. V příloze je skript, který collectd nastaví následovně:

```
LoadPlugin contextswitch
LoadPlugin cpu
LoadPlugin df
LoadPlugin disk
LoadPlugin entropy
LoadPlugin interface
```

```
LoadPlugin load
LoadPlugin memory
LoadPlugin processes
LoadPlugin uptime
LoadPlugin users
LoadPlugin vmem
LoadPlugin tcpconns

<Plugin df>
  FSType "rootfs"
  IgnoreSelected true
  ReportInodes true
</Plugin>
LoadPlugin network
  <Plugin network>
  ReportStats true
    <Server "ctd-server-ip" >
      SecurityLevel "Encrypt"
      Username "username"
      Password password
    </Server>
</Plugin>
```

V této chvíli jsou již metriky (ty, co jsou vybrané mezi načtenými pluginy) odesílány na server s IP adresou server-ip.

3.4.0.2 server s uloženými daty

Všechna statistická data jsou uložena na dalším odděleném serveru. Tento počítač ma dle předchozí kapitoly IP adresu "ctd-server-ip". Aplikace collectd zde běží v server režimu a naslouchá na portu 0.0.0.0:25826 na UDP. Tomu napovídá níže přiložená konfigurace v collectd.conf:

```
Hostname FIXME-HOSTNAME
FQDNLookup false
Interval 30
ReadThreads 1
LoadPlugin syslog
<Plugin syslog>
        LogLevel info
</Plugin>
LoadPlugin rrdtool
LoadPlugin network
```

```
## Server config
<Plugin network>
  Listen "0.0.0.0" "25826"
  ReportStats true
        SecurityLevel "Sign"
  AuthFile "/etc/collectd/auth_file"
</Plugin>

<Plugin rrdtool>
        DataDir "/var/lib/collectd/rrd"
</Plugin>
```

Jak můžeme vidět, uživatelská jména a hesla jsou ukládána do souboru /etc/collectd/auth_file. Konfigurační soubor má formát

```
username1: pass1
username2: pass2
```

3.4.0.3 Collectd Graph Panel

Pro poskytnutí grafů na serveru s uloženými daty využijeme vynikajícího projektu Collectd Graph Panel. Tento grafický frontend je vytvořený v jazyce PHP a distribuovaný pod licencí GPL 3. Pro instalaci potřebujeme nainstalovaný `rrdtool`, který je možné operačním systému Debian získat následovně:

```
# apt-get install rrdtool
```

Dále webový server s podporou PHP verze alespoň 5.0, použijeme tedy nám známý Apache server (httpd) s pro nás postačujícím `mod_php`.

```
# apt-get install apache2 libapache2-mod-php

# a2enmod mod_php
```

Instalace Collectd Graph Panelu je velmi jednoduchá. Vezmeme oficiální git repozitář, který naklonujeme do `DocumentRootu` webového serveru.

```
$ git clone https://github.com/pommi/CGP
```

V této podobě již CGP zobrazuje grafy. Pro naše potřeby je ale nutné ještě nakonfigurovat zabezpečené připojení na webovém serveru a omezit přístup do aplikace pouze z určitých IP adres. Před nastavením zabezpečeného připojení potřebujeme povolit ssl modul v Apache web serveru:

```
# a2enmod ssl
```

Získání certifikátu od certifikační autority Let's Encrypt je popsáno na straně 30.

3.4.1 Vývoj pluginu

3.4.1.1 Název

Dle doporučení v dokumentaci Foremanu má název začínat předponou "foreman_", pro lepší identifikaci a asociaci pluginu s projektem. Dále pokud je název víceslovný, jednotlivá slova oddělujeme podtržítkem. Protože jsou pluginy publikovány jako gemy na stránce rubygems.org, je také potřeba zjistit, zna námi zvolené jméno je stále volné (tomu pomůže i zmíněný prefix).

Námi zvolené jméno je tedy "foreman_colletctd_graphs".

3.4.1.2 Příprava prostředí

Na GitHubu projektu je již fungující ukázkový plugin, který je možné využít pro jakékoliv naše potřeby. Obsahuje mnoho typů chování, které je nám libo využít - mezi ně patří přidání nových modelů, přepisování pohledů, přídávání uživatelských práv, položek do menu, atp. Soubor README v ukázce obsahuje seznam aktuálního chování.

Se základními znalostmi programu git si projekt naklonujeme na lokální stroj, kde budeme projekt vyvíjet:

```
$ git clone https://github.com/theforeman/foreman_plugin_template foreman_colletctd_
```

Tímto stáhneme aktuální verzi z master větve do adresáře foreman_colletctd_graphs. Repozitář obsahuje skript na změnu jména na námi zvolené, to provedeme následovně:

```
$ ./rename.rb foreman_colletctd_graphs
```

3.4.1.3 Instalace pluginu

Instalace pluginu ve vývojovém prostředí je jednodužším řešením, protože kód náčítá za běhu a nené potřeba instalovat plugin jako gem. Vytvořením vývojové instance Foremanu jsme se zabývali v kapitole ZZ.

Plugin je možné rovnou spustit (a prozkoumat jeho chování) úpravou souboru `Gemfile.local.rb`. Také je možné vytvořit soubor `foreman\_colletctd\_graphs.rb` v adresáři `bundler.d` a vložit do něj tento řádek:

```
# Gemfile.local.rb
gem 'foreman_colletctd_graphs', :path => 'path_to/foreman_colletctd_graphs'
```

Potom instalujeme "preface"bundle příkazem:

```
$ bundle install
```

A znovu restartujeme Foreman. Nový plugin můžeme vidět v záložce plugin tab na About Page.

3.4.1.4 Vývoj

Prvně upravíme soubor `foreman_colletctd_graphs.gemspec` a do něj vložíme meta informace, jako jméno pluginu, autora, web stránku projektu, verzi. Dále do souboru `lib/foreman_colletctd_graphs/engine.rb` vložíme následující:

```
# lib/foreman_colletctd_graphs/engine.rb
initializer 'foreman_colletctd_graphs.register_plugin', :before => :finisher_hook do
  Foreman::Plugin.register :foreman_colletctd_graphs do
    # the code of our plugin
  end
end
```

3.4.1.5 Struktura pluginu

The project is a Rails::Engine [27] as described in the instructions for Foreman plugins in the official documentation [54]. The most adequate choice was an engine with a semi-isolated namespace [46], since I wanted to avoid namespace pollution resulting in name conflicts but I needed an access to the classes in Foreman and Katello. The plugin has a standard Rails project structure with minor modifications mirroring the conventions in Foreman and Katello projects.

3.4.1.6 Deface

Pro úpravu HTML stránek, do kterých budeme vkládat grafy, použijeme knihovnu Deface. Umožňuje nám upravit HTML pohledy bez zásahu do spodního Ruby pohledu. Použití knihovny je možné dvěma způsoby, já zde popíši jen jeden z nich. Pro více informací je dostupná dokumentace zde [**?**].

3.4.1.7 Content Security Policy

`Content-Security-Policy` je HTTP hlavička vytvořená s hlavním cílem snížení XSS rizik deklarováním, jaký obsah je povolen k načtení. V současnosti hlavičku podporují prohlížeče Google Chrome (od verze 25+), Firefox (31+), Safari (7+), i Microsoft Edge.

Ve frameworku Foreman je tato hlavička vynucena – pomocí gemu Secure Headers. Pokud grafy vygenerované z rrd souborů chceme načítat z jiného serveru, do konfigurace pluginu je třeba přidat adresu serveru s uloženými daty. Konfigurace se provádí v souboru `/usr/share/foreman/config/initializers/secure_headers.rb` a to následovně:

3.4.2 Závěr

Ačkoliv výsledný plugin neobsahuje hodně kódu, veškerá jeho funkcionalita dle zadání je naplněna. Na obrázku je možné vidět aktuální podobu.

Po naprogramování práce jsem objevil již vytvořený plugin foreman-colly [**?**], též generující grafy sbírané démonem collectd, ale fungující na odlišném způsobu. Doplněk od Lukáše Zapletala aktivně naslouchá paketům s metrikami, můj plugin pouze zobrazuje již vygenerované grafy. Výhledově by bylo možné tyto dvě funcionality spojit do jednoho většího doplňku.

3.4.2.1 Vytvoření Gemu

http://bundler.io/v1.12/guides/creating$_gem.html$

3.4.2.2 Balíček pro OS Debian

https://softwareengineering.stackexchange.com/questions/195633/good-approaches-for-packaging-php-web-applications-for-debian

3.4.3 Nasazení pluginu

V produkčním prostředí je nasazení pluginu velmi jednoduché. Náš vytvořený .deb balíček pomocí rsync/scp technologie přesuneme na server. V adresáři obsahující balíček napíšeme:

```
$ rsync -avh foreman_colletctd_graphs.deb $UNAME@$SERVERNAME:
```

a následovně nainstalujeme nástrojem standartně obsaženým v Debianu:

```
$ dpkg -i foreman_colletctd_graphs.deb
```

Dalším krokem je restartovat Foreman server, pokud máme systemd jako náš init systém, tento krok vykonáme následovně:

```
# systemctl restart foreman.service
```

V této chvíli plugin máme aktivní a funkční.

3.5 Ansible

Ansible je nástroj určený k automatickému nastavení strojů podle předem určených parametrů. Součástí bakalářské práce jsou konfigurační soubory pro Ansible, tzv. playbooky, pomocí kterých je možné jednotlivé části infrastruktury nakonfigurovat během několika minut.

Ansible v porovnání s konkurenčními nástroji, jako je např. Chef nebo Puppet, nevyžaduje žádnou instalaci agenta na koncových zařízeních. Pro připojení ke koncovým zařízením se tak nejčastěji používá SSH (Secure Shell). V

terminologii Ansiblu se tato zařízení označují jako uzly (angl. nodes). Informace o jednotlivých uzlech jsou uvedeny v inventáři (angl. Inventory), kde je možné definovat parametr uzlů.

3.5.1 Inventář

Důležitým souborem v Ansible je tzv. Inventář (Inventory), který obsahuje informace o nastavovaných uzlech. Formát souboru je INI [**?**], kde by měly být specifikovány jména uzlů, jejich IP adresy, uživatelská jména, hesla, porty, na kterých se chceme připojit, atp. Soubor nám umožňuje jednotlivé servery seskupovat.

```
$ cat servers_list
[master]
host1  ip_addr=10.10.1.2

[proxy]
host2  ip_addr=10.10.2.2
host3  ip_addr=10.10.3.3

[collectd]
host4  ip_addr=10.10.1.3
```

Pro přidání další lokality do infrastruktury je tedy nutné pouze přidat server s jeho IP adresou do Inventářového souboru (je nutné mít v novém proxy serveru SSH klíč Ansible serveru). Při vytváření (úpravě) inventářového souboru je třeba dbát na určité požadavky:

- server označený jako master musí být pouze jeden,
- collectd server též pouze jeden,
- proxy serverů může být libovolně (role proxy musí být zprovozněna alespoň na jednom serveru – klidně na stejném, jako master, nebo jiném).

3.5.2 Spouštění příkazů v Ansible

Při použití Ansible jsou možné dva různé způsoby vzdáleného spuštění příkazů na nastavovaných serverech. První metodou je tzv. Ad-Hoc [**?**]. Tímto způsobem je možné interaktivně (z příkazové řádky) spustit příkaz a okamžitě vidět výsledek činnosti. Příkladem je ping modul, pomocí kterého je možné zjistit, zda servery jsou po síti dostupné:

```
$ ansible -m ping -u deployer servers_list
```

Po vykonání příkazu Ansible s pomocí ping modulu vyzkouší všechny servery definované v inventářovém souboru `servers_list`, zda jsou dostupné k připojení přes SSH. Případem užití je většinou rychlé nasazení oprav na více serverů najednou. Těchto modulů je v současnosti přes 500 [**?**] a s každým vydáním nové verze jich přibývá.

Jiným přístupem při definování serverů je konfigurace pomocí Ansible Playboků. Playbooky jsou skripty psané v jazyce YAML. Viz krátká ukázka:

```
---
- name : Install and configure foreman
    hosts : servers_list
    remote_user : user
    sudo : yes

  tasks :
  - name : ( os = Debian ) Install Foreman
    apt: name=foreman state=present update_cache=yes
    sudo: yes
```

Při vykonávání Playbooku se Ansible pokusí vytvořit SSH připojení se všemi servery definovanými v souboru `servers_list`. Pokud konfigurovaný server obsahuje OS Debian, nainstaluje se z repozitářů balíček Foreman. Playbooky v příloze práce jsou složitější.

3.5.3 Playbook pro Foreman a proxy servery

Scénáře Ansiblu mohou být velice strukturovatelné. V našem případě je rozdělíme do více souborů, a poté je budeme ovládat nadřazeným scénářem. Tato volba by měla pomoci udržet scénáře přehledné. Vytvoříme tedy role – sady příkazů pro provedení určitých změn. Role jsou reprezentovány adresářem v adresáři roles. Inspirací pro konfigurační soubory v příloze bakalářské práce jsou již vytvořené playbooky pro instalaci instancí Foremanu od společnosti Adfinis Sysgroup [**?**], volně dostupné v jejich GitHub repozitáři.

Playbooky obsahují několik rolí, které lze na cílový server nainstalovat:

- Foreman pomocí Foreman instalátoru,
- dodatečná konfigurace Foremanu (webserver, šablony obsažené ve Foremanu, instalace pluginu),
- nastavení isc-dhcp-server,
- nastavení TFTP serveru,
- foreman-proxy,
- nastavení collectd serveru pro sběr metrik.

Před spuštěním playbooku je třeba splnit určité požadavky. Pokud požadavky nejsou splněny, instalace nemusí být zakončena správně. Mezi tyto nároky patří:

Master server:

- správně nastavené FQDN (DNS A záznam musí mířit na tento server),
- vypnutý SELinux,
- jelikož je s Foremanem instalován i puppet, je třeba, aby stroj měl alespoň 2GB RAM paměti,
- ve firewallu je třeba, aby porty 67, 69, 80, 443 (a další dle kapitoly Zabezpečení serveru) byly otevřené,
- byl na konfigurovaném serveru přístup k internetu a repozitářům Debianu.

  Na proxy serveru se předpokládá, že síťové rozhraní eth0 je použito pro veřejnou síť, eth1 pro interní síť.

  Playbooky jsou dostupné jak v příloze práce, tak v mém GitHub repozitáři [**?**].

Závěr

Literatura

[1] RFC 951 - Bootstrap Protocol. Online, Duben 2017. Dostupné z `https://tools.ietf.org/html/rfc951`.

[2] Berners-Lee T., Fielding R. & Frystyk H. (zobrazeno 11.12.2016.) Pre-boot execution environment (PXE) specification version 2.1. Tech. rep. URL: ftp://download.intel.com/design/archives/wfm/downloads/pxespec.pdf.

[3] RFC 783 - TFTP Protocol (revision 2). Online, Duben 2017. Dostupné z `https://tools.ietf.org/html/rfc783`.

[4] The GNU General Public License v3.0 - GNU Project - Free Software Foundation. Online, Duben 2017. Dostupné z `https://www.gnu.org/licenses/gpl-3.0.en.html`.

[5] Installation instructions - Smart Proxy - Foreman. Online, Duben 2017. Dostupné z `http://projects.theforeman.org/projects/smart-proxy/wiki/Installation_{}instructions`.

[6] Foreman :: Manual. Online, Duben 2017. Dostupné z `http://theforeman.org/manuals/1.7/index.html#5.5.1Backup`.

[7] PRAŽÁK, Ondřej. Foreman plugin for Jenkins CI. Brno, 2015. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Grác Marek.

[8] A Comparative Study of Baremetal Provisioning Frameworks http://www.pdl.cmu.edu/PDL-FTP/associated/CMU-PDL-14-109.pdf

[9] ŠAMALÍK, Adam. Extension of OpenStack Modules for Ansible Platform. Brno, 2016. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Hruška Martin.

[10] Intro to Playbooks mdash; Ansible Documentation. Online, Duben 2017. Dostupné z http://docs.ansible.com/ansible/playbooks_intro.html.

[11] GitHub - adfinis-sygroup/foreman-ansible: Ansible playbook to deploy a complete Foreman instance within minutes.. Online, Duben 2017. Dostupné z https://github.com/adfinis-sygroup/foreman-ansible.

[12] INI file - Wikipedia. Online, Duben 2017. Dostupné z https://en.wikipedia.org/wiki/INI_file.

[13] GitHub - lzap/foreman_colly: Foreman plugin for collectd. Online, Duben 2017. Dostupné z https://github.com/lzap/foreman_colly.

[14] BackupPC: Open Source Backup to disk. Online, Duben 2017. Dostupné z http://backuppc.sourceforge.net/.

[15] GitHub - spree/deface: Rails 3 plugin that allows you to customize ERB views in a Rails application without editing the underlying view.. Online, Duben 2017. Dostupné z https://github.com/spree/deface.

[16] Zero-day (computing) - Wikipedia. Online, Duben 2017. Dostupné z https://en.wikipedia.org/wiki/Zero-day_(computing).

[17] Bad Economy Is Good for Open Source. Online, Duben 2017. Dostupné z http://www.cmswire.com/cms/web-cms/bad-economy-is-good-for-open-source-004187.php.

[18] Spacewalk still a viable option? : linuxadmin. Online, Duben 2017. Dostupné z https://www.reddit.com/r/linuxadmin/comments/5muwsl/spacewalk_still_a_viable_option/.

[19] Foreman vs Puppet. Online, Duben 2017. Dostupné z https://www.upguard.com/articles/foreman-vs.-puppet.

[20] Installation · puppetlabs/razor-server Wiki · GitHub. Online, Duben 2017. Dostupné z https://github.com/puppetlabs/razor-server/wiki/Installation#installing-packages.

[21] Puppet Management GUI Comparison | OlinData. Online, Duben 2017. Dostupné z https://www.olindata.com/en/blog/2014/01/puppet-management-gui-comparison.

[22] GitHub - sodabrew/puppet-dashboard: The Puppet Dashboard is a web interface providing node classification and reporting features for Puppet, an open source system configuration management tool. Online, Duben 2017. Dostupné z https://github.com/sodabrew/puppet-dashboard.

[23] LiveCD - 1 (úvod, isolinux). Online, Duben 2017. Dostupné z http://www.abclinuxu.cz/clanky/system/livecd-1-uvod-isolinux.

[24] grub2 - What is the difference between GRUB and SYSLINUX? - Ask Ubuntu. Online, Duben 2017. Dostupné z https://askubuntu.com/questions/651902/what-is-the-difference-between-grub-and-syslinux.

[25] GitHub - puppetlabs/razor-el-mk: The discovery kernel for razor-server. Online, Duben 2017. Dostupné z https://github.com/puppetlabs/razor-el-mk.

[26] Foreman :: Plugin Manuals. Online, Duben 2017. Dostupné z https://theforeman.org/plugins/foreman_discovery/2.0/.

[27] Backend Installation · StackIQ/stacki Wiki · GitHub. Online, Duben 2017. Dostupné z https://github.com/StackIQ/stacki/wiki/Backend-Installation#discovery.

[28] Introduction To Ad-Hoc Commands mdash; Ansible Documentation. Online, Duben 2017. Dostupné z http://docs.ansible.com/ansible/intro_adhoc.html.

[29] GitHub - j-sokol/foreman_collectd_graphs_plugin. Online, Duben 2017. Dostupné z https://github.com/j-sokol/foreman_collectd_graphs_plugin.

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS