Univerzita Hradec Králové
Fakulta informatiky a managementu
katedra informačních technologií

BAKALÁŘSKÁ PRÁCE

Implementace redakčního systému Wordpress pomocí konfiguračního managementu

Autor: David Sucharda

Studijní obor: Aplikovaná Informatika

Vedoucí práce: Ing. Aleš Komárek

Hradec Králové duben 2015

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a uvedl jsem všechny použité prameny a literaturu.

V Hradci Králové dne 26. dubna 2015 David Sucharda

Poděkování

Rád bych zde poděkoval Ing. Aleši Komárkovi za odborné vedení práce, podnětné rady a čas, který mi věnoval.

Anotace

Ve své práci se zabývám softwarovým konfiguračním managementem. S jeho pomocí je nainstalován webový redakční systém. V první části práce je popsán úvod do tématu a důvod vyběru práce. Ve druhé části jsou teoretické informace o konfiguračním managementu a redakčnímu systému, který je nasazován. V další části jsou praktické informace a popisy vybraných použitých zdrojových kódů. Následovat bude provedení měření rychlosti a spolehlivosti nasazení redakčního systému. Závěr obsahuje shrnutí informací a možnosti rozšíření stávajícího řešení.

Annotation

Title: Implementation of cms Wordpress using configuration management

In my thesis I'm dealing with system configuration management. With its help is deployed a web content management system. The first chapter is introduction to the subject and the reason for selecting this thesis. In the second chapter are theoretical information about configuration management and content management system, which will be deployed. In the following chapter are practical information and description of selected source codes. Next chapter is about performing tests of speed and reliability of cms deployment. Final chapter contains summary information and the possibility of expanding existing solutions.

Obsah

1 Úvod **1**

1.1 Důvod výběru tématu bakalářské práce 1

1.2 Shrnutí práce 2

2 Teoretická část **3**

2.1 Konfigurační management 3

2.1.1 Historie Konfiguračního managementu 4

2.1.2 Software management 4

2.1.3 Hardware management 5

2.1.4 Information Technology Infrastructure Library (ITIL) 6

2.2 Použitý software konfiguračního managementu 6

2.2.1 Salt 8

2.2.2 Ansible 9

2.2.3 Puppet 10

2.3 Wordpress 12

2.3.1 Rozšíření Wordpressu 12

2.3.2 Šablony Wordpressu 13

2.3.3 WP-CLI 13

2.3.4 PHP a Apache 13

2.3.5 MySQL 14

2.4 Pracovní prostředí 14

2.4.1 Git 15

2.4.2 Reclass 15

Yaml 16

2.4.3 OpenSSH 16

3 Praktická část **17**

3.1 Vytvoření prostředí pro práci 17

3.2 Informace o Implementaci 18
3.2.1 Wordpress 18
3.3 Implementace pomocí Salt 18
3.3.1 Vytvoření souborů Wordpressu 21
3.3.2 Úprava souboru wp-config 22
3.3.3 Instalace nástroje WP-CLI 22
3.3.4 Instalace databáze 24
3.3.5 Instalace rozšíření, šablon a aktualizací 25
3.4 Implementace pomocí Ansible 28
3.4.1 Instalace WP-CLI 30
3.4.2 Instalace jádra WP 31
3.4.3 Vytvoření souboru wp-config 32
3.4.4 Instalace databáze 33
3.4.5 Instalace rozšíření a šablon 35
3.4.6 Instalace aktualizací 36
3.4.7 Nasazení s využitím Gitu 37
3.5 Puppet 38
3.5.1 Vytvoření Wordpress souborů 40
3.5.2 Vytvoření souboru wp-config 41
3.5.3 Instalace databáze 41
3.5.4 Rozšíření a šablony 43

4 Testování **44**
4.1 Rychlost nasazení 44
4.2 Bezpečnost nasazení 45

5 Závěr **46**
5.1 Možnosti rozšíření 46

Literatura **51**

Seznam obrázků

2.1 KM – proces konfiguračního managementu (zdroj: [42]) 4
2.2 Salt – master/minion (zdroj: [11]) 9
2.3 Ansible – workflow nasazení plabooku (zdroj: [41]) 10
2.4 Puppet – diagram zpracování změn (zdroj: [19]) 11
2.5 PHP a Apache komunikace (zdroj: [46]) 14
2.6 Git - verzování a větve (zdroj: [40]).) 15

3.1 OpenStack – použité stroje 17

Seznam tabulek

2.1 Programy KM – zabezpečení (zdroje: [16],[20],[13],[20],[43],[44],[45]) 7
2.2 Programy KM – podpora platforem (zdroje: [16],[20],[13],[20],[43],[44],[45]) . 7

4.1 Test rychlosti WP-CLI vs Git 44
4.2 Rychlost jednotlivých nástrojů 45

1 Úvod

S využitím konfiguračního managementu se zrychluje nasazení, aktualizace a úprava softwarových aplikací na více než jednom stroji. I když je tento postup velice bezpečný a rychlý, tak ho zatím moc firem nevyužívá. Místo toho používají vlastní konfigurace a skripty, které fungují na omezené části projektů. Tím pádem vznikají konfigurační odchylky, problémy s bezpečností. Díky těmto odchylkám jsou systémy těžko sledovatelné, škálovatelné a udržovatelné. Čím je větší projekt, tím více má členů a právě díky tomu také vzniká více problémů a zpomaluje se vývoj.

Ve středních a větších firmách, které se zabývají vývojem softwarových aplikací, je potřeba zrychlit a automatizovat jejich nasazení. Šetří to čas pro práci zaměstnanců a tím samozřejmě i náklady firmy.

Pokud by se například zpracovávalo pět webových stránek, které mají fungovat jako blog, tak je dobré zpracovat pouze jednu a zbylé zkopírovat. Bez automatizace by se musely takové weby nasazovat a nastavovat každá zvlášť. Díky tomu se také zamezí různým odchylkám a problémům.

1.1 Důvod výběru tématu bakalářské práce

Tímto tématem jsem se začal zabývat teprve nedávno, když jsem se dostal do většího týmu programátorů a přebral sekci webových projektů na redakčním systému Wordpress. Jeden z úkolů bylo vytvořit soutěžní stránku a tu poté rozkopírovat na dalších šest domén. Vytvoření jako takové nebyl problém, ten nastal až při nasazování.

Kopírování souborů na jednotlivé servery pomocí FTP je často velice zdlouhavá práce. Další problém je následná úprava, protože webové stránky nemůžete nechat jen tak být. Je třeba sledovat jejich využití, funkčnost a dostupnost. Pokud nastane problém musí se co nejrychleji opravit a pokud máte pod dohledem desítky webů, tak tyto úpravy mohou zabrat mnoho času.

Z tohoto důvodu jsem se rozhodl otestovat nasazení webového redakčního systému Wordpress pomocí konfiguračního managementu.

1.2 Shrnutí práce

Cílem této práce je zjistit, jak náročné je nasazení Wordpressu pomocí konfiguračního managementu a také jaký nástroj z vybraných je k tomu nejlepší. Nasazování Wordpressu je realizováno pomocí tří vybraných nástrojů softwarového konfiguračního managementu. Vybrány byly SaltStack, Ansible a Puppet, protože jsou v této době nejrozšířenejší. Součástí tohoto nasazení je také stažení pluginů (rozšíření) do Wordpressu a jeho kompletní aktualizace.

Každý z těchto nástrojů využívá jiný způsob nasazení a také jiné programovací jazyky. Proto bude potřeba vytvořit konfiguraci a postup nasazení pro všechny tyto nástroje. Postupy by měly být co nejvíce univerzální, aby se mohly použít na jakýkoliv typ webové stránky fungující na Wordpressu.

Prvním úkolem je nasadit a zprovoznit Wordpress pomocí každého nástroje. Dalším je instalace jednotlivých rozšíření redakčního systému a v poslední části se zpracuje aktualizace všech součástí.

2 Teoretická část

V této kapitole je popsán konfigurační management a jeho historie. Dále jsou zde informace o vybraných nástrojích konfiguračního managementu a také o redakčním systému Wordpress.

2.1 Konfigurační management

Konfigurační management je velice rozsáhlá oblast IT managementu. Hlavním cílem je co nejvíce usnadnit a automatizovat procesy nasazování, sledování i aktualizace nových hardwarových či softwarových řešení pro celou společnost.[3][1] Konfigurační management je jednou ze součástí ITIL, který se zabývá podporou IT oblasti v organizaci. ITIL (Information Technology Infrastructure Library) je soubor praxí pro plánování a využívání informačních technologií nejen ze strany dodavatelů IT služeb ale i zákazníků.[4]

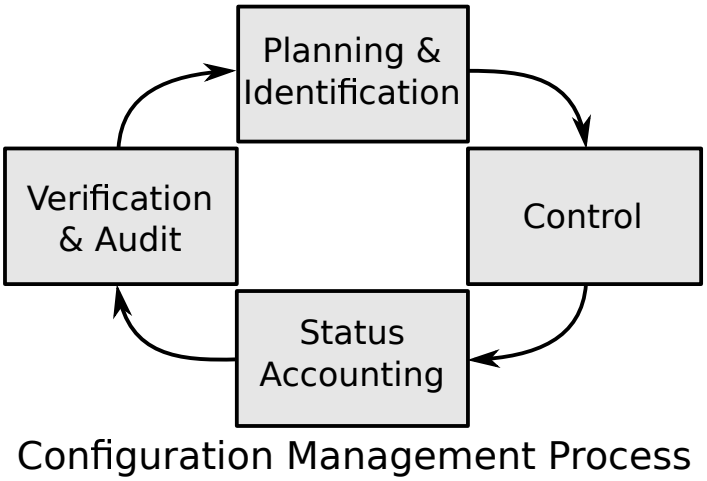
Konfigurační management se zabývá:

1. Sledováním, řízením, aktualizací systémových a hardwarových prostředků všech prvků sítě společnosti.
2. Řízením vývoje komplexních aplikací, aby nepřetěžovaly hardware ani software a přitom si udržely svoji použitelnost i rychlost.
3. Nasazením aplikací a jejich nastavení na více nebo všechny prvky bez problémů s ohledem na zachování funkčnosti.

Konfigurační management se také dělí podle částí, které sleduje a udržuje. Hlavní části jsou software management a hardware management.[7]

- Software management se zabývá vývojem a správou operačních systémů, aplikací, konfigurací i dat pro celou společnost.
- Hardware management se zabývá hlavně plánováním a nasazením fyzických prostředků pro správný chod softwaru a celé sítě společnosti.

Postup konfiguračního managementu:



Obrázek 2.1: KM – proces konfiguračního managementu (zdroj: [42])

2.1.1 Historie Konfiguračního managementu

Termín konfigurační management a s ním spojená technická disciplína se začal objevovat na konci 60. let. Vznikl díky Americkému Ministerstvu Obrany (DoD), a to když se hardware standardy začaly používat nově na software. Dříve se totiž využívalo jen hardware standardů. V 70. letech vznikly vojenské standardy, zabývající se konfiguračním managementem, zvané „480 series", které byly v 1991 spojeny do jednoho vojenského standardu MIL–STD–973.

V této době již konfigurační management není pouhou součástí vojenského standardu. Díky tomu vzniklo mnoho termínů s ním spojených, například: product lifecycle, systems engeneering nebo integrated logistics system.

Zajímavý je product lifecycle, který je jeden z nejvyužívanějších v konfiguračním managementu. Zabývá se zpracováním celé životnosti produktu od jeho návrhu, výroby, prodeje až po jeho likvidaci. Poskytuje kompletní informace o celém životním cyklu jako potřebné materiály na výrobu, personál a další důležitá data.[5]

2.1.2 Software management

Pod pojmem software se dá představit, například: operační systém, aplikace nebo data. Hlavním rozdílem softwaru je, že je vytvořen tak, aby se dal lehce nahradit pomocí aktualizace nebo záměnou za jiný. Nedá se nijak masově vyrábět jako hardware a díky tomu je práce na software náročnější na množství lidí než je tomu u hardware. Proto mohou být velké

rozdíly v cenách softwarových aplikací, podle toho kolik lidí na nich pracovalo.[6][7]

Další rozdíly od hardware managementu:

1. Náročnější na testování – aplikace jsou většinou tvořeny z různých funkcí (částí) a otestovat se musí každá znich. Testování se provádí, jak při nasazení, tak i během vývoje, aby se daly chyby odladit ještě před nasazením. Oproti hardware, který se testuje jako celek, pokud pracují všechny jeho části, tak se testování ukončí.

2. Provádění oprav – když u softwarové aplikace nefunguje jistá funce, musí se celá aplikace prověřit a upravit nejen porouchaná funkce, ale i ostatní. Protože funkce jsou na sobě závislé a využívají kód jiných funkcí. Tak je to i u hardware, ale u toho se vymění poškozená součást a vše by mělo fungovat v pořádku.

Jednotlivé části:

- Sledování a aktualizace systémů - pomocí sledování se určuje, které systémy jsou nejnáročnější na údržbu, jaké potřebují vylepšit a jestli by mohly způsobit problémy. Při aktualizaci je třeba si dávat pozor na to, že v průběhu nebo po nasazení aktualizace může vzniknout chyba, a proto je třeba mít předešlé systémy zálohované pro rychlé obnovení provozu.

- Vývoj komplexních aplikací - při vytváření softwaru se spoléhá na komunikaci všech prvků společnosti od vůdců firmy, programátorů až po testery a uživatele aplikací. Když komunikace není správně vyřešena, může docházet k problémům, jako je odchylka od hlavního cíle nebo velká náročnosti na hardwarové prostředky.

- Nasazení nových aplikací - díky softwarovému managementu se nasazují aplikace na vybrané nebo všechny části sítě, což ušetří mnoho času. Pomocí sledování softwarových prostředků také testuje jejich náročnost a v mnoha případech se musí stáhnout z živého provozu, aktualizovat nebo vypnout. Znovu se pak musí dát pozor na možnost vzniku chyb.

2.1.3 Hardware management

Hardware se zabývá servery, koncovými počítači, sítěmi a jejich komponenty. Jde nejen o vytvoření a udržování hardwarových prostředků, ale také, jak budou servery zaměřeny. Některé mohou podporovat webové aplikace, a proto potřebují výkonější sítové prostředky.

Na jiných serverech budou zálohy, tak musí obsahovat větší diskový prostor a na dalších mohou běžet náročné výpočtové aplikace, takže mají vyšší paměť a výpočetní rychlost.

Hardware management jako takový se hlavně zabývá plánováním a nasazením hardware prostředků tak, aby celková struktura pracovala co nejrychleji a nejspolehlivěji.[7] Je také součástí softwaru pro vytváření cloudů, jako je například OpenStack. Jeho hlavní součástí je správa hardwarových zdrojů a jejich využití.

2.1.4 Information Technology Infrastructure Library (ITIL)

Tento rámec vznikl během 80. let ve Velké Británii. Slouží jako standard řízení a správy IT služeb v organizaci. Zaměřuje se také na jejich neustálé zlepšování a sledování, jak z pohledu společnosti, tak i zákazníka. ITIL není normou, ale souborem doporučení a nejlepších postupů.[8][4]

Hlavní rysy ITIL:

1. Nezávislost na platformě
2. Jednoznačná terminologie – aby se zabránilo nedorozuměním kvůli využití jednoho termínu ve více významech.
3. Proces řízení – tento proces je sled činností pro změnu vstupu na výstup, je také monitorován, měřen, vyhodnocován a vylepšován.

2.2 Použitý software konfiguračního managementu

Softwarových aplikací pro konfigurační management je nespočet. Většina z nich je zadarmo a open source. Open source je možnost stažení zdrojového kódu a jeho úpravy pro vlastní potřeby.

Jedna z nejdůležitějších informací o těchto aplikacích je jaké podporují možnosti zabezpečení a platformy. Další důležitá informace je, v jakém jazyce jsou napsané a jaký princip komunikace využívají. Některé nástroje využívají strukturu master/minion, při které probíhá komunikace většinou mezi ovládajícím strojem a ovládaným. Jiné nástroje zase mohou používat komunikaci se všemy stroji v síti. [9]

Jsou tři hlavní části zabezpečení, kde první dvě jsou podporovány skoro ve všech programech:

1. Oboustranná autentifikace – klient ověřuje server a server ověří klienta.
2. Šifrování - zajišťuje aby se data po síti neposílala v textové podobě.
3. Ověřovací mód (Verify mode).

Tabulka 2.1: Programy KM – zabezpečení (zdroje: [16],[20],[13],[20],[43],[44],[45])

	Programovací jazyk	Oboustranná autentifikace	Šifrování	Ověřovací mód	První vydání
Ansible	Python	Ano	Ano	Ano	2012
Puppet	Ruby	Ano	Ano	Ano	2005
Salt	Python	Ano	Ano	Ano	2011
Quattor	Perl, Python	Ano	Ano	Ne	2005
SmartFrog	Java	Ano	Ano	Ne	2004
STAF	C++	Ne	Částečně	Ne	1998

Další programy, které podporují všechna zabezpečení jsou – CFEngine, Chef, Bcfg2 , Rundeck, Synctool.

Tabulka 2.2: Programy KM – podpora platforem (zdroje: [16],[20],[13],[20],[43],[44],[45])

	AIX	HP-UX	Linux	Mac OS X	Solaris	Windows	Další
Ansible	Ano	Ano	Ano	Ano	Ano	Ano	Ne
Puppet	Ano	Ano	Ano	Částečně	Ano	Ano	Ano
Salt	Ne	Ne	Ano	Ano	Ano	Ano	Částečně
Quattor	Ne	Ne	Ano	Částečně	Ano	Ne	Ne
SmartFrog	Ne	Ano	Ano	Ano	Ano	Ano	Ne
STAF	Ano	Ano	Ano	Ano	Ano	Ano	Ano

Při výběru správného programu je vhodné dávat pozor na typ komunikace a jaké platformy podporuje. Většina podporuje všechny hlavní platformy jako Linux, Windows, Mac

OS X nebo Solaris alespoň částečně.

V této práci jsou použity programy Puppet, Ansible a Salt, protože jsou v dnešní době jedny z nejrozšířenějších.

2.2.1 Salt

Salt, jinak známý jako SaltStack, je také jeden z novějších aplikací na konfigurační management s první stabilní verzí z roku 2011. Tento program vznikl z programů na rychlý sběr a zpracování dat v administrativním prostředí. SaltStack umí také pracovat v cloudu a umožňuje vytvořit privátní cloud nebo automatizovat virtuální servery.[10]

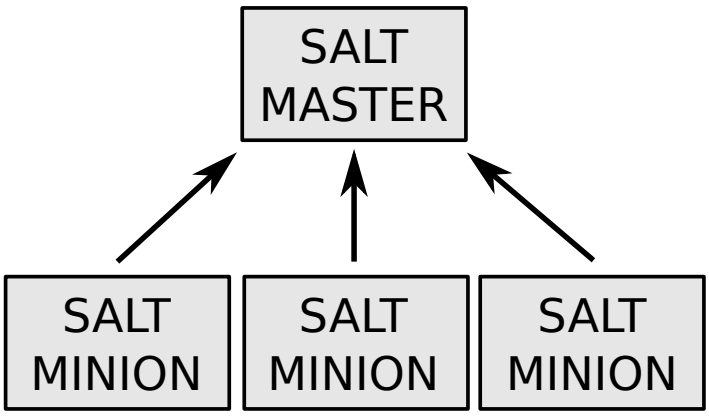
Jeho hlavní zaměření je na modulovost, rozšířitelnost, jednoduchost a hlavně rychlost. Moduly jsou vytvářeny v jazyce Python a dají se oddělit od Saltu. Díky tomu si administrátor může program upravit přesně, jak potřebuje. Hlavním cílem je také jednoduché propojení s různými druhy jiných aplikací.

Salt je založený na architektuře master/minion, kde master ovládá své miniony. Minioni mají za úkol zpracovávat příkazy a posílat odpovědi na mastera. Tato komunikace je velice rychlá, protože Salt využívá pro ukládání dat jednoduché soubory a RAM pamět místo databáze.[14]

Základní typy modulů [11]:

- Execution – tyto moduly jsou hlavním jádrem Salt, obsahující cross platformní informace, aby se Salt mohl používat pro více platforem.
- State – části, které tvoří kostru a spouští jednotlivé moduly Saltu, které musí být spuštěny. Nastavují konfiguraci na klientovi.
- Grains – hledají statické informace o systému a ukládají je do RAM paměti pro rychlé využití.
- Returners – zachytávají odpovědi na úpravy z klientů, schromažďují je a zpracovávají.

Dokumentace pro SaltStack je hlavně v textové podobě na webových stránkách. Pro naučení, natrénování programu je možné využít týmové kurzy nebo kurzy v určitých zeměpisných oblastech (nejvíce v USA). Je zde také možnost zapsat se na certifikaci na SaltStack Inženýra. Firmy, které využívají Salt, jsou například: Apple, U.S. Department of Defence, Harvard University, LindedIn a další.[12][13]



Obrázek 2.2: Salt – master/minion (zdroj: [11])

Díky tomu, že Salt funguje na principu master/minion je nezávislný na platformě.

2.2.2 Ansible

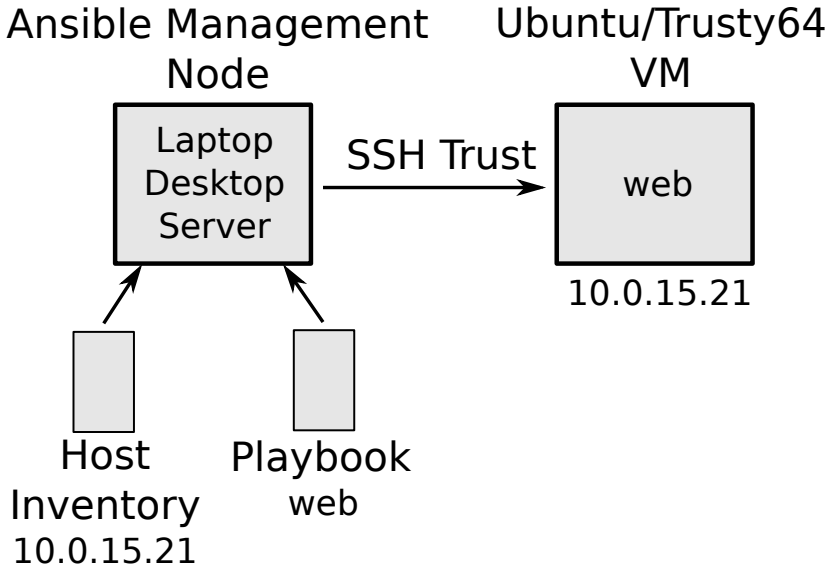
Ansible je také poměrně nový software konfiguračního management. Celý tento program se zakládá na jednoduchosti použití už od první verze z roku 2012. Hlavním cílem je, aby se administrátor po čase nepoužívání nemusel znovu učit, jak se s touto aplikací pracuje.

Ansible je založen na způsobu zpracování malých programů (Ansible Modules), které jsou napsány tak, aby se klient dostal do požadovaného stavu. Ansible tyto programy spustí nejčastěji pomocí SSH a po dokončení programu je smaže. Aplikace nevyžaduje žádné servery nebo databáze.[17][15]

Základní body Ansible:

- SSH – Ansible je zaměřen na bezpečnost, podporuje hesla a další způsoby přihlášení, ale jako hlavní se využívá SSH autentifikace.
- Textové soubory – pro zvýšení jednoduchosti při nastavování klienta pracuje s jednoduchými textovými soubory, stejně jako například webové servery s html, php nebo css soubory.

Ansible jako takový podporuje možnost vlastních rozšíření, kde má každý možnost takové rozšíření vytvořit. Navíc u nich nezáleží v jaké jazyce jsou napsány. Jedinné co programovací jazyk musí umožnovat je vracet data kódované jako JSON.



Obrázek 2.3: Ansible – workflow nasazení plabooku (zdroj: [41])

Obrázek popisuje, jakým způsobem probíhá nasazení určitého playbooku na miniona pomocí nástroje Ansible. S využitím host souboru, ve kterém je IP adresa ovládaného stroje, je zajištěna komunikace. Ansible poté spustí playbook na ovládaném stroji a po dokončení playbook ze stroje smaže a ukončí spojení.

Ansible dokumentace se hlavně zaměřuje na internetově šířené textové dokumenty a videa. Některé firmy které využívají Ansible - Twitter, Electronic Arts, GoPro nebo Motorola. [15][16]

2.2.3 Puppet

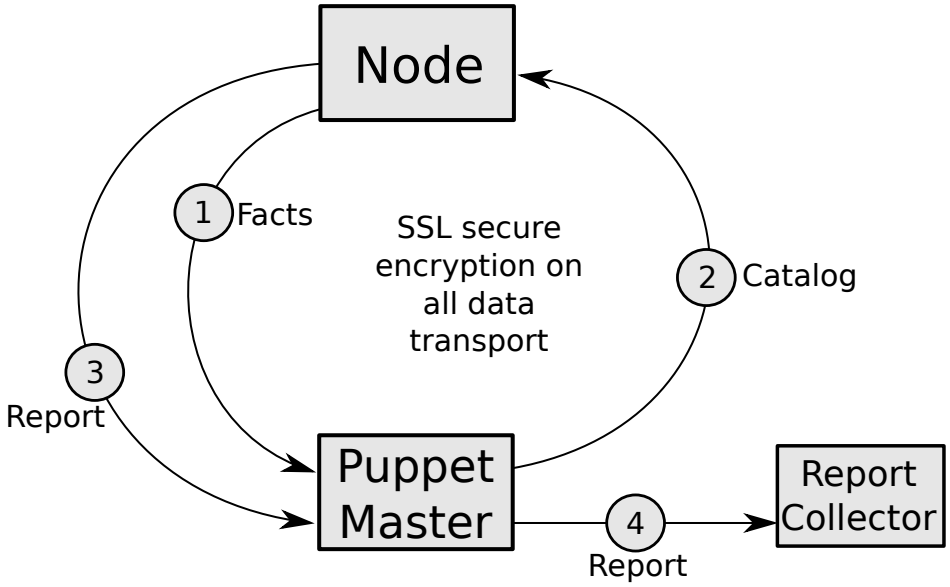
Puppet je open source program, který je vyvíjen společností Puppet labs od roku 2005. Puppet nahrazuje psaní kódů na jedno použití i další manuální práce pro systémového administrátora a zároveň zajišťuje konzistenci přes celou infrastrukturu.

Existuje i placené rozšíření Puppet Enterprise. Tato edice je rozdělena na tzv. node. Každý z nich má svojí vlastní funkci a dají se různě dokupovat, takže si každý systémový administrátor může nastavit Puppet, jak potřebuje. U Enterprise edice je také možné kontaktovat přímo systémové administrátory z Puppet labs a požádat je o konzultaci. Puppet pracuje jako deklarativní jazyk, to znamená, že se mu řekne (napíše) co přesně má udělat a on to tak zpracuje.

Základní funkce Puppet:

- Definuje požadovaný stav infrastruktury.
- Simuluje změny nastavení před jejich nasazením.
- Nasazuje požadovaný stav automaticky a kontroluje nastavení.
- Hlásí rozdíly mezi stávajícím a požadovaným stavem, zaznamenává také změny, které byly provedeny, aby infrastruktura byla v požadovaném stavu.

Puppet dostává systém do požadovaného stavu podle následujícího diagramu, kde každé zařízení v infrastruktuře by mělo mít nainstalovaný Puppet klient nebo Puppet master (administrátorský program).[19][18]



Obrázek 2.4: Puppet – diagram zpracování změn (zdroj: [19])

Popis každé části diagramu:

1. Fakta – Puppet klient pošle z každého zařízení jeho normalizovaná data na zařízení s nainstalovaným Puppet master.
2. Katalog – Puppet master zpracovává fakta a upravuje je podle požadovaného nastavení, rozesílá úpravy na zařízení, aby Puppet klient změnil nastavení.
3. Hlášení – klient posílá hlášení o provedených změnách na mastera.
4. Sběr Hlášení – Puppet sbírá všechna hlášení z klientů. Je možné je exportovat pro sdílení informací s jinými týmy nebo aplikacemi.

Puppet má také velice dobře zpracovanou dokumentaci a je zde i možnost zapsat se na placené kurzy. Známé firmy, které využívají Puppet jsou: Cisco, Github, Intel, NASA, Spotify, Sony a mnoho dalších.[18][20]

2.3 Wordpress

Wordpress je open source redakční systém pro tvorbu a správu webových stránek. Je to systém se základním jádrem a jeho funkčnost je možné rozšířit za pomoci rozšíření (plugin). V této době existuje obrovské množství rozšíření od eshopu až po rezervaci stolů pro restaurace. Další jeho součást je šablonový (theme) systém, díky kterému je možné mít různý počet vzhledů a upravit je podle libosti.

Od začátku vývoje měl Wordpress své bezpečností problémy, ale díky velké komunitě se postupem času většina těchto chyb opravila. Nejlepšího zabezpečení lze dosáhnout pomocí ochranných rozšíření a také je vhodné využít zabezpečení přes htaccess soubor, ve kterém je možné definovat různá zabezpečení jako přesměrování, blokované ip adresy a povolené soubory.

Základní Wordpress využívá programovací jazyk PHP a MySQL databázi. Použití MySQL se dá vyhnout pomocí rozšíření, které umožňuje ukládat data do souborů na server místo do databáze. V této práci je standardní nasazení s PHP i MySQL a samozřejmě datový prostor pro uložení souborů Wordpressu.[21][22][2]

Každé rozšíření může mít svoje vlastní nastavení, a proto se čas na kompletní nastavení Wordpressu s každým rozšířením zvyšuje. I díky tomu se může čas na nastavení může pohybovat kolem 12 hodin čisté práce (bez ohledu na vzhled). Z tohoto důvodu se nasazuje Wordpress s jednoduchou šablonou a k tomu pouze několik důležitých rozšíření, mělo by jich být kolem 20. Tento způsob nasazování zrychlí implementaci webů a aktualizaci veškerých součástí.

2.3.1 Rozšíření Wordpressu

V této době již existuje obrovské množství rozšíření a šablon pro úpravu Wordpressu, a proto dobře zpracované stránky nemusejí vůbec vypadat jako by fungovaly na tomto systému. Na začátku roku 2014 bylo zaznamenáno přes 30 000 rozšíření a Wordpress využívalo 22% z 10 milionů nejlepších stránek na světě.[24][25]

Pro tuto práci je vytvořený seznam rozšíření využívaných na stávajících stránkách, které jsou v mojí správě. Je jich kolem 100, ale pro jednotlivé implementace se použije 19 vybra-

ných, které jsou vhodné pro jednotlivé účely webů. Od potřeb e-shopu až po blogy a komentáře. Použití velkého množství rozšíření zbytečně zpomaluje načítání stránek, proto je vhodné využívat pouze pár klíčových.

2.3.2 Šablony Wordpressu

Většina šablon pro Wordpress je placená, protože ve velkém měřítku upravují funkčnost webu. Jsou zaměřeny na vzhled webu a díky tomu by měly mít dobré grafické zpracování.

Programátor má možnost bezpečně upravit jakoukoliv šablonu přes tzv.: child-theme. Taková šablona pouze upravuje některé funkce a zbytek se načítá z té hlavní. Pokud se vytvoří soubor se stejným jménem jako je v hlavní šabloně, potom se bude načítat soubor z child-theme a díky tomu se dají upravit určité komponenty šablony.[26][27]

Pro tuto práci budou využity základní vzhledy Wordpressu a jako možnost několik vybraných od skupiny ElegantThemes, která vytváří velice propracované šablony s kvalitním vzhledem.

2.3.3 WP-CLI

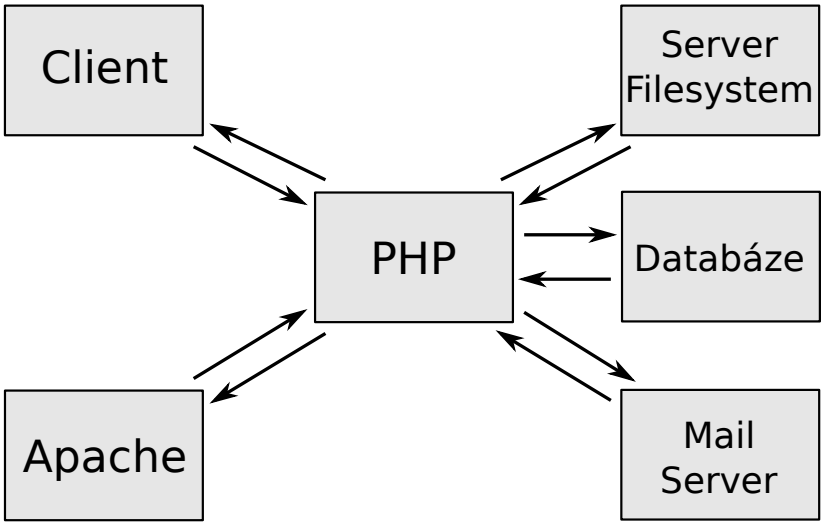
WP-CLI je nástroj, díky kterému se dá celý Wordpress ovládat pomocí příkazové řádky. Tento nástroj také umožnuje instalaci Wordpressu i všech jeho dalších součástí. Navíc umožňuje i kompletní aktualizaci všech součástí Wordpressu. Takže je vhodné tento nástroj využít pro instalaci Wordpressu a jeho následovné aktualizaci. Obě tyto možnosti budou implementovány pomocí jednoho příkazu pro všechny weby na Wordpressu.[28]

2.3.4 PHP a Apache

PHP je jeden z nejpoužívanějších programovacích jazyků pro tvorbu dynamických webových stránek. První z jeho výhod je, že jeho kód je zpracováván na serveru, a proto není přístupný uživatelům stránek. Další výhodou je možnost vkládání kódu z jiných souborů, proto stačí kód napsat pouze jednou a pomocí příkazu ho vložit do více stránek. Díky tomu se nemusí nic vytvářet vícekrát na každé stránce zvlášť.[36]

Protože je PHP zpracováváno na serveru, je potřeba mít serverového klienta, například: Apache. Apache je HTTP serverový klient, který zpracovává HTTP požadavky z prohlížeče a po následném zpracování vrací HTML kód. Apache podporuje v dnešní době mnoho rozšíření a jazyků jako například: PHP, Python nebo Perl. Jedna z jeho výhod je, že je také open source, a proto si ho může kdokoliv upravit.[38]

Komunikace mezi PHP a Apche:



Obrázek 2.5: PHP a Apache komunikace (zdroj: [46])

2.3.5 MySQL

Náročnější aplikace vyžadují možnost ukládat informace a nastavení do databáze, aby se s nimi mohlo následně lépe a přehledněji pracovat. Databáze je pouze několik tabulek plných informací a určitých pravidel. Tyto informace jsou uloženy v souboru a proto byl vytvořen jazyk SQL, aby k nim byl snadný přístup pomocí příkazů.

MySQL je variace SQL, která se velmi často používá s jazykem PHP pro vytvoření webových stránek. MySQL pracuje s daty pouze jako textové řetězce a proto je na PHP, aby tento text správně zpracoval do vhodné podoby. MySQL databáze je nejčastěji zabezpečena pomocí uživatelského jména a hesla. Databáze jako taková by také měla být nejzabezpečenějším místem webových stránek, protože může obsahovat citlivé informace o webu a jeho uživatelých. [37]

2.4 Pracovní prostředí

Prostředí ve kterém je práce zpracována se nazývá OpenStack. Je to soubor softwarových nástrojů pro vytvoření a správu cloudové platformy. V cloud platformě se dají vytvořit virtuální stroje, které dokáží zpracovávat mnoho různých funkcí. Tyto virtuální stoje využívají hardwarové prostředky více reálných strojů pro zrychlení jejich výkonu.

Výhoda OpenStack je, že je open source a díky tomu si kdokoliv může stáhnout jeho zdrojový kód a upravit ho. I díky tomu má velkou komunitu vývojářů, která se snaží, aby jejich

produkty byly co nejlepší a nejbezpečnější.[30][1]

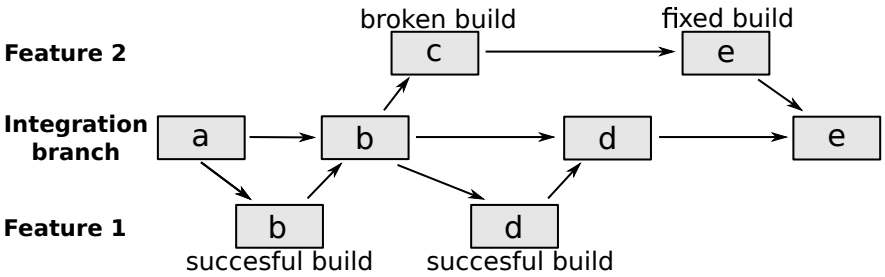
V tomto prostředí byly poskytnuty čtyři virtuální stroje, na kterých je operační systém Linux. První stroj je master (hlavní), který ovládá zbylé tři stroje. Tento master stroj zpracovává aktualizace součástí z Gitlabu a implemetuje je na ostatní stroje. Další tři stroje jsou určeny pro nasazení instancí Wordpressů přes vybraný software konfiguračního managementu. Na těchto strojích je tedy vynucena i základní podpora PHP a MySQL, které Wordpress potřebuje pro svojí funkčnost.

Všechny z následujících součástí pracovního prostředí jsou již zpracovány, a proto jsou nasazovány automaticky při vytváření prostředí.

2.4.1 Git

Jedna z hlavních součástí je také nástroj Git, který se stará o přesuny a aktualizaci souborů. Git je nástroj pro version control (správu verzí). Díky kterému je možné mít více verzí jednoho projektu. Různé verze umožnují sledování veškerých změn provedených na projektu a jejich vrácení nebo zapracování do hlavní části projektu. Další výhoda verzování je možnost práce více lidí na jednom projektu.[31]

Tento nástroj umožňuje ukládat data na servery soukromé i veřejné. Největším zástupcem veřejných serverů je v této době Github a na soukromý server se dá zakoupit verze Gitlab, která je placená. V této době Gitlab využívá přes 100 000 organizací. Jedny z těchto organizací jsou: Nasa, IBM nebo AVG.[32]



Obrázek 2.6: Git - verzování a větve (zdroj: [40]).)

Obrázek popisuje, jak funguje větvení v nástroji Git. Integration brach je hlavní větev, do které se vkládají funkce z jiných větví, pokud fungují správně.

2.4.2 Reclass

Reclass umožňuje sjednotit konfigurace z různých nástrojů konfiguračního managementu. Díky tomu je možné vytvořit jednotné nastavení, které funguje v Puppetu, Saltu i Ansible.

Reclass umožňuje součásti definovat jako třídy, které mají svou hierarchii a také zpracovává přepsání detailů napříč hierarchií. Jazyk pro nastavení, který je využívaný v tomto případě, je Yaml.

V prostředí, ve kterém je práce zpracovávána, je reclass funkční pro Salt a Ansible, proto budou použity jako první.[33]

Yaml

Yaml je jazyk pro serializaci dat, který je human-friendly (dobře čitelný pro člověka). Tento jazyk se používá hlavně na konfiguraci a díky snadné úpravě je možná rychlá úprava a změna funkcí aplikace.

Yaml byl inspirován jazyky C, Perl a Python, ale formát dat převzal z jazyka XML. Proto byl v začátcích použitelný pouze pro tyto jazyky, ale dnešní době se dá použít téměř pro jakýkoliv programovací jazyk.[34]

2.4.3 OpenSSH

OpenSSH je nástroj, který umožňuje zabezpečenou komunikaci přes Internet. Není na mysli lidská komunikace, ale přenosy dat a vzdálené ovládání počítačů. Tento nástroj šifruje jak hesla, tak komunikaci mezi počítači. Díky tomu je velice odolný proti hackingu. Tento nástroj využívá například Ansible, který přes něj komunikuje s ostatními stroji. Jedny ze společností využívající OpenSSH jsou: Cisco, Apple nebo NETFLIX.

Tento nástroj je tedy automaticky použit v případě Ansible. Pro připojení na ovládající stroj se ale využije nástroj Putty. Tento nástroj podporuje stejný protokol pro komunikaci jako OpenSSH, kterým je SSH, a je také velice lehce použitelný.[35]

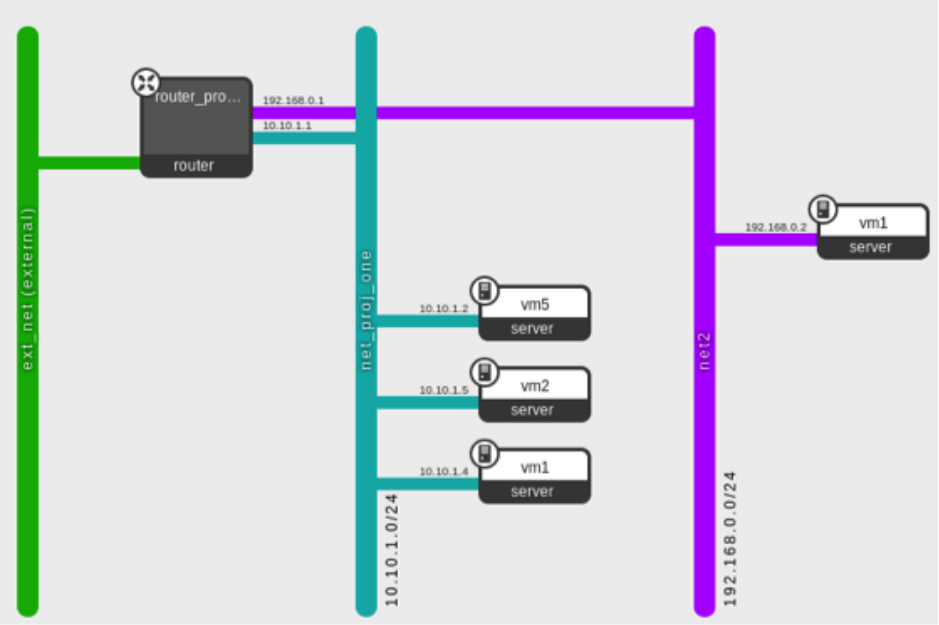
3 Praktická část

V této kapitole jsou popsány postupy pro implementaci Wordpressu pomocí vybraných nástrojů konfiguračního managementu.

3.1 Vytvoření prostředí pro práci

První krok je vytvoření routeru, sítě a strojů v cloudovém prostředí OpenStack. Vytvořený router je spojením mezi internetem a vnitřní sítí, ve které jsou stroje. Na těchto strojích jsou jednotlivé nástroje konfiguračního managementu a také instance Wordpressu. Díky tomuto routeru se dá ovládat jednotlivé stroje přes internet, a proto není potřeba být ve stejné síti jako je cloud.

Obrázek použitých strojů z prostředí OpenStack:



Obrázek 3.1: OpenStack – použité stroje

3.2 Informace o Implementaci

Po zpracování prostředí je na čase implementovat Wordpress a WP-CLI. Tyto části jsou instalovány na jednotlivé stroje pomocí různých nástrojů konfiguračního managementu. Pro instalaci Wordpressu a aktualizace se využívá hlavně nástroj WP-CLI. Pokud ale nebude možné tento nástroj nainstalovat, tak bude instalace dokončena pomocí jiných součástí, jako je Git nebo Linuxové příkazy.

3.2.1 Wordpress

Implementace Wordpressu se skládá z kopírování základních souborů na server a vytvoření databáze. Dalším krokem je instalace Wordpressu, která se dá provést pomocí přístupu přes prohlížeč, kde se nastavují přihlašovací údaje k databázi a poté k Wordpressu jako takovému. Tento krok je automatizován, aby stránky fungovaly okamžitě po nasazení. Proto je potřeba vytvořit konfigurační soubor (wp-config.php), který se nahraje do složky s Wordpressem. Následující krok je automatizace instalace databázových tabulek a jejímu naplnění pomocí základních (defaultních) hodnot. Posledním krokem v nasazování je instalace vybraných rozšíření, šablon a zpracování aktualizace celého systému.

Proto je postup práce takový:

- Vytvoření složek a nakopírování souborů Wordpressu.
- Nahrání upraveného souboru wp-config.
- Vytvoření databáze a vložení základních dat.
- Instalace rozšíření a témat + aktualizace.

3.3 Implementace pomocí Salt

Salt funguje na bázi master/minion, proto je potřeba na hlavním stroji vynutit součásti, jak pro ovládání, tak i pro ovládané stroje. Některé další součásti budou Git, Linux, OpenSSH a reclass. A poslední vynucované části jsou formule, které zaručí kompletní nastavení jednotlivých součástí. Veškeré konfigurační soubory jsou zapsány v jazyce Yaml.

Příklad nastavení součástí na masteru:

```
classes:
- system.git.client
- system.linux.system.single
- system.openssh.client.cluster.cloudlab
- system.openssh.server.team.tcpcloud
- system.salt.minion.master
- system.salt.master.single
- system.salt.master.formula.linux
- system.salt.master.formula.saltstack
- system.salt.master.formula.wordpress
- system.reclass.storage.salt
- system.reclass.storage.system.webapp_stg
```

Ukázka kódu 3.1: Master součásti (Salt)

Detailní nastavení dalších tří strojů je pomocí `webapp_stg.yml`, ve kterém jsou definované součásti pro každý stroj. K součástem se také mohou přidat jednotlivé parametry, které jsou potřebné k jejich správnému fungování.

Parametry na masteru:

```
parameters:
  _param:
    reclass_data_repository:
       git@repo1.robotice.cz:reclass-models/dels-salt-model.git
    reclass_data_revision: master
    reclass_config_master: 192.168.11.100
    salt_master_host: 127.0.0.1
    salt_master_environment_name: dev
    salt_master_environment_repository:
       git@repo1.robotice.cz:saltstack-formulas
    salt_master_environment_revision: develop
```

Ukázka kódu 3.2: Master parametry (Salt)

Tyto parametry nastavují url odkaz do Gitlabu, ze kterého má master stáhnout reclass a Salt. K tomuto stahování se také uvádí verze (revision) jakou má Git stahovat. U reclassu je důle-

žité nastavit ip adresu hlavního stroje, tato adresa je nastavena na ovládaných strojích, aby byla možná komunikace strojů. Poslední nastavení je verze pro Salt a v jakém prostředí se pracuje, jestli ve vývojovém nebo produkčním.

Po nastavení mastera se přejde k vytvoření konfigurace pro Salt miniona, ve které se vynucuje Linux, openssh a Wordpress formule.

Nastavení součástí jednoho miniona:

```
node:
  wordpress_web_salt_stg:
    name: web01
    domain: wordpress.dls.vpc.cloudlab.cz
    classes:
    - system.linux.system.single
    - system.openssh.client.cluster.cloudlab
    - system.openssh.server.team.tcpcloud
    - system.wordpress.server.multi
    - system.wordpress.server.app.devel
    params:
      salt_master_host: ${_param:reclass_config_master}
      apache_wordpress_devel_host: wordpress-salt.robotice.cz
      mysql_wordpress_devel_password: password
```

Ukázka kódu 3.3: Minion součásti (Salt)

V této části se nastavují jak součásti, tak nutné parametry pro správnou funkčnost a komunikaci mezi ovládaným a ovládajícím strojem. Jeden z těchto parametrů musí být IP adresa mastera, na kterou ovládaný stroj zasílá informace. Při zpracování webových stránek je také důležité, aby byly přístupné z internetu pomocí url adresy, proto je zde i toto nastavení.

Po těchto nastaveních se může přejít k vytvoření Wordpress formule, díky které se instance Wordpressu instaluje na ovládaný stroj. Instalace jako taková se povede příkazem na serveru `salt 'web01*' state.sls wordpress`. Tento příkaz říká Saltu, aby aktualizoval formuli Wordpress na stroji se jménem začínajícím web01.

3.3.1 Vytvoření souborů Wordpressu

Před vytvořením souborů Wordpressu se musí vytvořit složka, ve které budou soubory umístěny. K tomu se využije Sakt funkce file.directory, která zajišťuje, že složka existuje a má správně nastavená přístupová práva. Pokud neexistuje, tak ji vytvoří.

Vytvoření složky pro soubory Wordpressu:

```
/srv/wordpress/sites/{{ app_name }}:
  file.directory:
  - user: www-data
  - group: www-data
  - mode: 770
  - makedirs: true
```

Ukázka kódu 3.4: Vytvoření složky WP

Parametr `app_name` je nastaven v konfiguračním yml souboru a nejčastěji má stejnou hodnotu jako url odkaz stránek, které se vytváří, ale není to nutné.[39]

Následujícím krokem je stažení Wordpress souborů do nově vytvořené složky. K tomuto úkolu se využije nástroj Git, který stáhne všechny soubory a umístí je do vytvořené složky.

```
wordpress_{{ app_name }}_git:
  git.latest:
  - name: {{ server.git_source }}
  - rev: {{ app.version }}-branch
  - target: /srv/wordpress/sites/{{ app_name }}/root
  - user: www-data
  - require:
    - pkg: git_packages
```

Ukázka kódu 3.5: Stažení WP

Paramter `app_name` je v tomto případě stejný jako v předchozím případě. Do části name se vkládá zdrojová url adresa, ze které Git stahuje soubory Wordpressu. Díky rev (revision) se můžou stahovat různé verze Wordpressu. Je důležité aby se tato funkce spuštěla pod uživatelem www-data, který je vlastníkem vytvořené složky. Je to z důvodu, aby se zajistilo, že má práva pro zápis do této složky. Pomocí tohoto uživatele se také instalují rozšíření a aktualizují součásti Wordpressu.

3.3.2 Úprava souboru wp-config

Soubor wp-config je před nahráním do určené složky upraven podle informací, které jsou uvedeny v yml soubu. Díky těmto úpravám je možné vytvářet rozdílné databáze a různé stránky fungující na systému Wordpress.

S využitím příkazu file.managed, se přesune tento soubor do složky Wordpressu s využitím nástroje Git. Tento příkaz tedy nejdříve soubor upraví podle zadaných informací a poté ho přesune.

Přesun souboru wp-config:

```
/srv/wordpress/sites/{{ app_name }}/root/wp-config.php:
  file.managed:
  - source: salt://wordpress/files/wp-config.php
  - template: jinja
  - mode: 644
  - require:
    - git: wordpress_{{ app_name }}_git
  - defaults:
    app_name: "{{ app_name }}"
```

Ukázka kódu 3.6: Přesun souboru wp-config

Díky možnosti template jsou do tohoto souboru nastavena data z yml souboru. Poté je volána součást Git jako required (vyžadovaná) součást pro přesun souboru.

3.3.3 Instalace nástroje WP-CLI

Před dokončením instalace Wordpressu a jeho dalších součástí je potřeba nainstalovat WP-CLI, protože s jeho pomocí se vytváří potřebná základní data a vkládají se do databáze.

Nejdříve se nakopíruje soubor, ve kterém je zapsán instalační script pro instalaci dokončování příkazů po WP-CLI. Kód pro tento úkol je skoro stejný jako pro přesun souboru wp-config. Jediný rozdíl je v použitých jménech a složce. Tento kód využívá složku pro dočasné soubory, aby se instalační soubor po provedení smazal.

Hlavním krokem této části je instalace WP-CLI, která je provedena pomocí standardního scriptu pro Linux. Tento script stáhne soubor s WP-CLI a nainstaluje ho s nastavenými právy. Následující část scriptu spustí instalaci pro dokončování příkazů WP-CLI.

Spuštění instalace WP-CLI pomocí cmd:

```
install_wpcli:
  cmd.script:
    - name: wpcli-install
    - source: salt://wordpress/files/wpcli-install.sh
    - cwd: /
    - user: root
    - require:
      - git: wordpress_{{ app_name }}_git
    - unless: wp cli version --allow-root
```

Ukázka kódu 3.7: Spouštění instalace WP-CLI

Pomocí příkazu cmd.script se spustí tento instalační script a není tak potřeba ho kopírovat na ovládaný stroj. Kód probíhá pod uživatelem root, protože potřebuje práva na zápis do složek do kterých uživatel www-data nemá přístup. Možnost unless zajišťuje, aby se WP-CLI neinstalovalo opakovaně, pokud již existuje.

Instalační script WP-CLI:

```
#!/bin/bash
curl -O https://raw.githubusercontent.com/wp-cli/builds/
        gh-pages/phar/wp-cli.phar

chmod +x wp-cli.phar
sudo mv wp-cli.phar /usr/local/bin/wp

chmod +x /tmp/wpcli-tab.sh
source /tmp/wpcli-tab.sh
```

Ukázka kódu 3.8: Instalační script WP-CLI

Nejdříve se musí určit, že se jedná o Linuxový bash script. Poté se s pomocí curl stáhne soubor wp-cli a dalšími řádky zajistí, že se WP-CLI může spustit a přesune jej do složky ve které jsou přítomny další funkce Linuxu. Soubor se díky tomu i přejmenuje na wp, aby se nemuselo pokaždé psát wp-cli.phar místo wp. Poslední řádky tohoto scriptu umožní spuštění instalačního scriptu pro dokončování příkazů.

3.3.4 Instalace databáze

Při implementaci databáze jsou dva postupy, které jsou zpracovány. Prvním je s podporou WP-CLI, protože umí vytvořit tabulky v databázi přesně podle verze Wordpressu. Tento postup je vhodnější, protože funguje v propojení s danou verzí Wordpressu.

Instalace databáze a dat pomocí WP-CLI:

```
{%- if salt['cmd.retcode']('wp core is-installed
   --path="'+web_path+'" --allow-root') == 1 %}
wp_install:
  cmd.run:
    - name: wp core install --url='{{ app.core_install.url }}'
       --title='{{ app.core_install.title }}' --admin_user='{{
       app.core_install.admin_user }}' --admin_password='{{
       app.core_install.admin_password }}' --admin_email='{{
       app.core_install.admin_email }}'
    - cwd: {{ web_path }}
    - user: www-data
{%- endif %}
```

Ukázka kódu 3.9: Instalace databáze s WP-CLI

Nejdříve se musí zabezpečit, aby se databáze neinstalovala pokaždé, protože by se tabulky pokaždé přepisovaly. Proto funkce if zjišťuje, jestli jsou tabulky nainstalované pomocí příkazu WP-CLI a rozhoduje, zda nainstalovat databázi nebo ne. Pokud tedy tabulky neexitují, tak se spustí příkaz wp core install, který je vytvoří. Tento příkaz využívá instalační informace z yml souboru, které by se jinak musely zadávat při přístupu z webového prohlížeče.

Druhá možnost je instalace datábáze bez WP-CLI. Jde o instalaci pomocí Linuxového příkazu, který nahraje základní data ze souboru sql. Tento soubor obsahuje základní Wordpress databázi z verze 4.1, takže pokud by se použil na nižší verzi, je možné že by Wordpress nefungoval správně. Je náročné tento soubor správně upravit, aby nepůsobil problémy. Wordpress sice disponuje možností aktualizace databáze, ale ne vždy toto řešení pomůže.

Nejdříve se pomocí Gitu zkopíruje soubor sql na ovládaný stroj, proto se znovu využije funkce file.managed. Kód je znovu téměř identický jako v případě wp-config. Když je tento soubor přesunut, je poté využit pomocí Linuxového příkazu pro přesun do MySQL databáze. To zpracuje kód ze souboru a vytvoří tabulky s daty.

Instalace databáze a dat pomocí příkazu mysql:

```
create_db:
  cmd.run:
    - name: mysql -u {{ app.database.user }} -p{{
       app.database.password }} < /tmp/init.mysql
    - require:
      - service: mysql
      - file: /tmp/init.mysql
```

Ukázka kódu 3.10: Instalace DB bez WP-CLI

Po těchto krocích je nainstalovaný základní Wordpress a připravený k použití.

3.3.5 Instalace rozšíření, šablon a aktualizací

Instalování rozšíření a jejich aktualizace je velice podobná záležitost, kde jediná změna je ze slova install na update. Aktualizace jako taková je zpracována pro všechny součásti Wordpressu i jeho jádro.

Stažení a instalace rozšíření nebo šablon probíhá cyklem, aby se nainstalovaly všechny součásti a ne jenom jedna. Veškeré tyto kódy jsou uzavřeny v tomto cyklu a jsou kontrolovány podmínkami, aby proběhla pouze jedna akce pro každé rozšíření.

Obalení cyklem for:

```
{%- for plugin_name, plugin in app.plugin.iteritems() %}
% Kód pro Aktualizaci / Instalaci %
{%- endfor %}
```

Ukázka kódu 3.11: For pro rozšíření

Díky tomuto cyklu je také možné se dostat k nastaveným informacím pro každé rozšíření, jako je jeho verze a způsob instalace. Tyto informace jsou použity v následujících kódech pro kontroly instalace nebo aktualizace.

Instalování rozšíření umožňuje verzování, čili stažení určité nastavené verze. Je zde možnost i stažení nejnovější verze. Díky verzování je možné rozšíření jak aktualizovat, tak i downgradovat (nainstalovat starší verzi). V Salt řešení jsou dvě možnosti aktualizace a to přes WP-CLI nebo nástroj Git.

Instalace/aktualizace rozšíření pomocí WP-CLI:

```
{%- if plugin.source.engine == 'http' %}
{{ plugin_name }}_install:
  cmd.run:
{%- if plugin.version == 'latest' %}
    - name: wp plugin install {{ plugin_name }}
{%- else %}
    - name: wp plugin install {{ plugin_name }} --version='{{
       plugin.version }}'
{%- endif %}
    - cwd: {{ web_path }}
    - user: www-data
```

Ukázka kódu 3.12: Instalace/aktualizace rozšíření s WP-CLI

Tento kód se provede pouze, pokud je nastaveno, aby se toto určité rozšíření aktualizovalo přes WP-CLI (http). Před provedením tohoto kódu se musí nejdříve zjistit, je-li dané rozšíření nainstalované, aby se neinstalovalo znovu. Jak již bylo řečeno, kód pro aktualizaci má stejný kód pouze se změnou slova v příkazech z install na update.

Instalace/aktualizace rozšíření pomocí Gitu:

```
{%- elif plugin.source.engine == 'git' %}
{{ plugin_name }}_git_update:
  git.latest:
    - name: {{ plugin.source.address }}
{%- if plugin.version != 'latest' %}
    - rev: {{ plugin.version }}
{%- else %}
    - rev: 'master'
{%- endif %}
    - target: {{ web_path }}/wp-content/plugins/{{ plugin_name }}
    - user: www-data
    - force: true
    - require:
      - git: wordpress_{{ app_name }}_git
```

```
{%- endif %}
```

Ukázka kódu 3.13: Instalace/aktualizace rozšíření s Gitem

Při instalaci pomocí Gitu je potřeba vědět adresu, na které jsou přístupné soubory rozšíření a tuto informaci uvést do name části. Pomocí rev (revision) se určuje, jaká verze rozšíření se má stáhnout. Jedna z dalších informací je také cesta, do které mají být rozšíření nainstalovány. Tato cesta se nastavuje pomocí target.

Aby tento kód fungoval musíme použít force: true, protože Git by jinak odmítl přepsat cílové soubory. Proto toto není úplně nejlepší řešení. Vhodnějším by bylo stahovat určité verze rozšíření do vlastního Gitlabu a tam kontrolovat úpravy v souborech. Díky tomu se také dá snadněji oddělit produkční a vývojové prostředí. V produkčním prostředí se nahrávají pouze otestovaná rozšíření a vývojové prostředí je určeno k testování nejnovějších verzí rozšíření, které se poté implementují na produkční prostředí.

Aktualizace jádra je určena pouze proměnnou ano/ne, jestli se má jádro aktualizovat. A jádro jako takové je jediná část v tomto redakčním systému, proto tento kód nemusíme obalovat do žádného cyklu.

Kód pro aktualizaci jádra:

```
{%- if app.do_update.core_update %}
wp_core_update:
  cmd.run:
    - name: wp core update
    - cwd: {{ web_path }}
    - user: www-data
    - unless: wp core check-update
{%- endif %}
```

Ukázka kódu 3.14: Aktualizace jádra

Aktualizace se provede, pokud je v yml nastavena proměnná aktualizace jádra (`core_update`) na true. A díky unless části se aktualizace provede pouze, pokud je k dispozici nová verze systému.

Kód pro aktualizaci šablon je skoro stejný jako pro aktualizaci jádra:

```
{%- if app.do_update.theme_update %}
wp_theme_update:
  cmd.run:
    - name: wp theme update --all
    - cwd: {{ web_path }}
    - user: www-data
{%- endif %}
```

Ukázka kódu 3.15: Aktualizace šablon

Aktualizování šablon je možné rozpracovat do stejné podoby jako u rozšíření, ale díky nutnosti platit za většinu šablon není možné využít WP-CLI a Git pouze pro některé z nich. Proto je tento postup zatím řešený jako jediný.

3.4 Implementace pomocí Ansible

Ansible jako jediný z použitých nástorojů nefunguje na bázi master/minion, ale pouze posílá soubor příkazů na ovládaný stroj, který se provede a poté se smaže. Díky tomuto je nasazení snadnější, protože není důležité jaký stroj je master a jaký minion. V tomto případě se storoje stále nazývají master a minion, ale v definici Ansible toto nemá žádný význam.

Ansible součást se nasazuje, jak na master stroj, tak na miniona (web03), na kterém budou soubory Wordpressu. Aby spolu mohly stroje komunikovat pomocí Ansible, je nutné aby měly seznam IP adres ostatních strojů ve svém host souboru, proto je tento soubor přítomný na všech strojích.

Proto se přidají další součásti do konfiguračního souboru mastera:

```
classes:
- system.salt.master.formula.ansible
- system.ansible.client.single
- system.ansible.server.single
- system.ansible.server.playbook.wordpress
```

Ukázka kódu 3.16: Další master součásti (Ansible)

Ansible client/server single obsahuje různá nastavení jako SSH klíče, zapnutí service a odkazu na playbook. Důležitá část je Ansible formule, která se stará o instalaci tohoto nástroje

a vytvoření uživatelů. Playbook Wordpress na serveru obsahuje umístění playbooku pro miniona a také různé parametry jako IP adressy strojů, které se vkládají do host souboru. Playbook je postup, jakým způsobem se má Wordpress nainstalovat, něco jako formule u nástroje Salt.

Je potřeba přidat i parametry na master:

```
parameters:
  _param:
    ansible_server_environment_repository:
       git@repo1.robotice.cz:ansible-playbooks
    ansible_server_environment_revision: develop
```

Ukázka kódu 3.17: Další master parametry (Ansible)

Podobné parametry jako u Salt, které se starají o nastavení, kde má Git hledat playbooky pro Ansible a také jaké jejich revision má stahovat. Po doplnění nastavení mastera je na čase zprovoznit samotného miniona, podobně jako tomu bylo u Saltu.

Nastavení součástí Ansible miniona:

```
node:
  wordpress_web_ha_ansible_stg:
    name: web03
    domain: wordpress-ha.dls.vpc.cloudlab.cz
    classes:
    - system.linux.system.single
    - system.openssh.client.cluster.cloudlab
    - system.openssh.server.team.tcpcloud
    - system.ansible.client.single
    params:
      salt_master_host: ${_param:reclass_config_master}
      apache_wordpress_devel_host: wordpress-ansible.robotice.cz
```

Ukázka kódu 3.18: Minion součásti (Ansible)

Salt master host je zde pouze pro možnost komunikací se Salt masterem, v tomto případě není nutný. Přidává se také host, aby se zajistil přístup na webové stránky pomocí url aresy. Tyto úpravy umožnily komunikaci hostů pomocí Ansible a také přístupnost playbooku pro

instalaci Wordpressu, proto se přejde k dalšímu kroku a to k vytvoření playbooku a jeho následné instalaci na miniona. Instalaci playbooku se spouští na masteru spustit pomocí příkazu "ansible-playbook název-playbooku.yml".

Ansible využívá pro svoji práci hlavně soubory yml. V předchozím případě se používal soubor sls, ve kterém se mohly definovat proměnné, funkce i cykly. To v tomto případě ale není možné, protože yml je jazyk pro serializaci dat. V Ansible se ale využívají specifické názvy v yml souborech, které zajistí stejné možnosti jako v souboru sls.

3.4.1 Instalace WP-CLI

V případě Ansible se první instaluje WP-CLI, protože se tento nástroj použije ve větším měřítku než v případě Saltu. Následující kód zpravovává jak stažení WP-CLI, tak jeho instalaci.

Stažení a instalace WP-CLI:

```
- name: install (wp-cli)
  get_url:
    url: https://raw.githubusercontent.com/wp-cli/builds/
         gh-pages/phar/wp-cli.phar
    dest: "{{ wordpress_wp_cli_dir }}/wp"
    force: true
    owner: root
    group: root
    mode: 0755
  tags: [configuration, wordpress, wordpress-wp-cli,
     wordpress-wp-cli-install]
```

Ukázka kódu 3.19: Instalace WP-CLI

Možnost `get_url` umožňuje stahovat soubory pomocí HTTP, HTTPS nebo FTP. Pro stažení je potřebné nastavit url odkaz na soubor, který se má stáhnout. Proměnná dest určuje, kam se má soubor stáhnout. Do této proměnné se vloží informace z `wordpress_wp_cli_dir`, ve které uložena cesta do složky bin, ze které jsou v Linuxu spouštěny uživatelské programy. Stažený soubor se uloží pod jménem wp a díky tomu je v konzoli přístupný pomocí příkazu wp.

Jako další se určuje uživatel, pod kterým se tento soubor ukládá a určují se jeho práva pro přístup. Další proměnné je force, která umožňuje stahovat soubor pokaždé, i když už je v učené složce přítomný. Pro malé soubory opětovné stahování nedělá v Ansible žádný problém. A poslední je možnost tags, která umožňuje spustit pouze jednu určitou část z celého playbooku. Tato možnost se hodí například pro spuštění pouze aktualizace Wordpressu.

3.4.2 Instalace jádra WP

Při instalaci jádra se zajišťuje i stažení Wordpressu, proto se kontroluje, jestli jsou soubory už přítomné v dané složce. Pokud soubory existují na správném místě, pak se nebude instalace provádět znovu. Jak již bylo řečeno, nedají se využít funkce if nebo for, a proto je vytvořen úkol, který se stará o tuto kontrolu.

Kontrola Wordpress souborů:

```
- name: check if wp is downloaded
  shell: "ls {{ item.path }} | grep -q 'wp-'"
  register: check_download
  failed_when: False
  changed_when: False
  with_items: wordpress_installs
  tags: [configuration, wordpress, wordpress-core,
     wordpress-is-downloaded]
```

Ukázka kódu 3.20: Kontrola WP souborů

Pro přehlednost pro uživatele je tento úkol pojmenován pomocí name. Nejdůležitější informace v tomto úkolu jsou Linuxový příkaz na kontrolu souborů, `with_items` a také register. Linuxový příkaz ls vypisuje soubory v určité složce, jaká je tato složka, je nastaveno v proměnné item.path. K proměnným se přistupuje pomocí již zmíněného `with_items`, který umí také přistoupit k proměnným jiných úkolů. Poslední důležitá informace register umožňuje nastavit název úkolu, pod kterým je přístupný jiným úkolům. Register je nastaven hlavně z důvodu, aby byly přístupné proměnné tohoto úkolu, protože nahrazuje kontrolu if.

Další parametry jsou `failed_when`, `changed_when` a také tags, který je popsán u minulého kódu. Díky možnosti `failed_when` je možné definovat, kdy Ansible vrátí informaci, že se úkol neprovedl. Pak je také potřeba někdy negovat zpáteční hodnotu, což zajiťuje parametr `changed_when`. Ani jedna z těchto možností v tomto případě využita není, a proto

mají nastavenou hodnotu false.

Všechny následující úkoly budou využívat příkazu wp, který poskytuje WP-CLI. Pokud nejsou stažené soubory Wordpressu, tak se musí stáhnout. K tomu je využit Linuxový příkaz wp core download.

Stažení souborů Wordpressu:

```
- name: download wp files
  shell: "wp core download"
  args:
    chdir: {{ item.item.path }}
  with_items: check_download.results
  when: wordpress_installs and item.rc != 0
  sudo: yes
  sudo_user: {{ item.item.user }}
  tags: [configuration, wordpress, wordpress-core,
     wordpress-downloaded]
```

Ukázka kódu 3.21: Stažení WP souborů

Tento úkol se spustí pouze, pokud předchozí kontrola vrátí hodnotu 1 (true). V Linuxu jsou významy false a true prohozeny, takže příkaz ls vrátí hodnotu 1, pokud nenajde žádné soubory obsahující "wp-"v určené složce. Pro přístup k této proměnné se musí použít `with_items` s názvem, který byl nastaven pomocí register. Přerušení tohoto úkolu se zajistí pomocí části when, která určuje, kdy se má úkol přerušit.

Aby se nemusel u příkazu wp core využívat parametr –path, dá se použít Ansible argument chdir, který změní umístění na určenou složku. Tato složka je také nastavena v proměnných. Jedno z posledních nastavení je sudo a `sudo_user`, které umožňuje změnit uživatele z root (administrátor) na uživatele zadaného v proměnných. Díky tomu se použije uživatel, který je určený na správu Wordpressu a také se nemusí využívat parametr –allow-root pro WP-CLI příkazy.

3.4.3 Vytvoření souboru wp-config

Po úspěšné instalci jádra je třeba vytvořit soubor wp-config. V případě Saltu se využíval soubor, do kterého se vložily proměnné z yml, a poté se tento soubor přesunul na miniona. V tomto případě se používá druhá možnost a to konfigurace pomocí WP-CLI. Výhodou je,

že se tento soubor vytváří přesně podle šablony dané verze Wordpressu, ale nevýhodou je, že automaticky ho není možné upravit více než dovoluje WP-CLI.

Vytváření souboru wp-config:

```
- name: create wp-config
  shell: "wp core config --quiet --dbname='{{ item.dbname }}'
     --dbuser='{{ item.dbuser }}' --dbpass='{{ item.dbpass }}'
     --dbhost='{{ item.dbhost | default('localhost') }}'"
  args:
    creates: "{{ item.path }}/wp-config.php"
    chdir: {{ item.path }}
  with_items: wordpress_installs
  sudo: yes
  sudo_user: {{ item.item.user }}
  tags: [configuration, wordpress, wordpress-core,
     wordpress-configure]
```

Ukázka kódu 3.22: Soubor wp-config

Přikaz wp core config vytvoří soubor wp-config a naplní ho proměnnými, které jsou uložené ve `wordpress_installs`. Je vhodné tomuto úkolu nastavit také argument creates, aby nebyl soubor wp-config vytvářen pokaždé. Znovu vytvářením by nejen zpomaloval další úkoly, ale také by mohl přepisovat úpravy provedené Wordpressem nebo administrátorem. Tomuto úkolu se nenastavuje hodnota register, protože na tomto úkolu nejsou nutně závislé žádné další úkoly.

3.4.4 Instalace databáze

Při instalaci databáze se musí kontrolovat, jestli už není naistalována. Pokud by kód pro její vytvoření projížděl pokaždé, tak by se do databáze vkládaly výchozí (defaultní) data. Vkládáním výchozích dat by se mazaly změny provedené v administraci Wordpressu. Kód pro kontrolu je velice podobný jako byl kód pro kontrolu souborů Wordpressu.

Kontrola před instalací databáze:

```
- name: check if database is already installed
  shell: "wp core is-installed --allow-root"
  args:
    chdir: {{ item.path }}
  register: check_installation
  failed_when: False
  changed_when: False
  with_items: wordpress_installs
  tags: [configuration, wordpress, wordpress-core,
     wordpress-is-installed]
```

Ukázka kódu 3.23: Kontrola databáze

Příkaz wp core is-installed kontroluje, jestli jsou Wordpress tabulky přítomné v databázi. Tento příkaz kontroluje pouze hlavní tabulky Wordpressu, nikoliv tabulky přidané pro rozšíření.

Pokud tedy nejsou tabulky v databázi, tak se musí vytvořit. Do tabulek je také potřeba vložit výchozí hodnoty, které automaticky vytvoří WP-CLI z dodaných parametrů. Jedny z těhto parametrů jsou administrátorský uživatel a jeho heslo.

Instalace databáze:

```
- name: install wp database
  shell: "wp core install --allow-root --url='{{ item.item.url }}'
     --title='{{ item.item.title }}' --admin_name='{{
     item.item.admin_name | default('admin') }}' --admin_email='{{
     item.item.admin_email }}' --admin_password='{{
     item.item.admin_password }}'"
  args:
    chdir: {{ item.item.path }}
  with_items: check_installation.results
  when: wordpress_installs and item.rc != 0
  tags: [configuration, wordpress, wordpress-core,
     wordpress-install]
```

Ukázka kódu 3.24: Instalace databáze

Díky příkazu wp core install a jeho parametrům se nainstaluje jádro Wordpressu. V tomto případě se nepřistupuje k proměnným přímo, ale přes kontrolu instalace, proto je v příkazu využito item.item místo item. Po instalaci je možné znovu použít příkaz wp core is-installed, aby se zkontrolovalo, jestli byl Wordpress správně nainstalován. Pokud někde v instalaci nastala chyba, pak se další úkoly neprovádí a vypíše se chybová hláška.

3.4.5 Instalace rozšíření a šablon

Kód pro instalaci rozšíření a šablon je kompletně stejný až na klíčové slovo ve WP-CLI příkazu. Prvním krokem je kontrola, jestli je rozšíření nainstalováno, aby se mohlo určit, jestli se jedná o instalaci nebo aktualizaci.

Kontrola přítomnosti rozšíření/šablony:

```
- name: identify installation plugin
  shell: "wp plugin is-installed --allow-root {{ item.1 }}"
  args:
    chdir: {{ item.0.path }}
  register: check_installation_plugins
  failed_when: False
  changed_when: False
  with_subelements:
    - wordpress_installs
    - plugins
  when: item.1
  tags: [configuration, wordpress, wordpress-plugins,
     wordpress-is-installed-plugin]
```

Ukázka kódu 3.25: Kontrola rozšíření/šablony

Kontroly probíhají pomocí příkazu wp plugin is-installed a wp theme is-installed. V tomto případě se přistupuje nejen k proměnným ale také k poli proměnných, které je podřazené jiné proměnné. K tomu se využívá možnost `with_subelements` a pro přístup k proměnným se využije klíčové slovo item a číslo subelementu, ve kterém je informace uložena (začíná od 0). Pomocí when se také určuje, aby se kód neprováděl, pokud není žádné další rozšíření definovano a mohlo se pokračovat dál.

Po kontrole se může přejít k instalaci nebo aktualizaci. Pokud rozšíření není přítomno, tak se nainstaluje pomocí příkazu wp plugin install a wp theme install. Pokud rozšíření nebo šablona s definovaným jménem neexistuje, pak se nenainstaluje a příkaz selže. Existence jména ve WP-CLI se kontroluje pouze se seznamem na Wordpress.org, který obsahuje pouze pluginy zdarma ke stažení.

Kontrola přítomnosti rozšíření/šablony:

```
- name: install plugin
  shell: "wp plugin install {{ item.item.1 }} --allow-root"
  args:
    chdir: {{ item.item.0.path }}
  with_items: check_installation_plugins.results
  when: check_installation_plugins is defined and item.item.1 and
     item.rc != 0
  tags: [configuration, wordpress, wordpress-plugins,
     wordpress-install-plugin]
```

Ukázka kódu 3.26: Kontrola rozšíření/šablony

Tento kód se provede, pouze pokud předchozí kontrola vrátí hodnotu 1 (true) a rozšíření je definováno. Znovu se také přistupuje k proměnným pomocí předchozího úkolu, a proto se zadává item.item. Po tomto kódu se znovu projíždí kontrola, jestli bylo rozšíření nainstalováno, aby se zamezilo dalším chybám a vynutila se kontrola konfigurace.

3.4.6 Instalace aktualizací

Aktualizace jádra, rozšíření i šablon se provádí pouze, pokud je nastavena proměnná pro jejich aktualizace na true a jsou-li nainstalované, aby nevznikaly chyby v konfiguraci.

Kód pro aktualizaci jádra:

```
- name: update wp core
  shell: "wp core update --allow-root"
  args:
    chdir: {{ item.item.path }}
  with_items: check_installation.results
```

```
  when: wordpress_installs and item.rc == 0 and
     item.item.core_update == 1
  tags: [configuration, wordpress, wordpress-core,
     wordpress-update]
```

Ukázka kódu 3.27: Aktualizace jádra (Ansible)

Díky modulu when se všechny kontroly před spuštěním spojí do jedné podmínky. Přístup k proměnným je znovu pomocí item.item, jediná možnost jak použít pouze jeden item je vložit do `with_items` další položku, ale pak by přístup k těmto proměnným byl pomocí item.0 a item.1.

Aktualizace rozšíření a šablon:

```
- name: update plugin
  shell: "wp plugin update {{ item.item.1 }} --allow-root"
  args:
    chdir: {{ item.item.0.path}}
  with_items: check_installation_plugins.results
  when: check_installation_plugins is defined and item.item.1 and
     item.rc == 0 and item.item.0.plugins_update == 1
  tags: [configuration, wordpress, wordpress-plugins,
     wordpress-update-plugin]
```

Ukázka kódu 3.28: Aktualizace rozšíření a šablon (Ansible)

Všechny části tohoto kódu již byly popsány v předchozích příkladech. Toto řešení je v tomto stavu kompletní a plně použitelné. Další možnost je využít Git pro lepší kontrolu nad celým projektem.

3.4.7 Nasazení s využitím Gitu

Pro Git je nejlepší vytvořit kopii stávajícího řešení s jiným názvem. Pokud by byly úkoly pro Git a WP-CLI v jednom řešení, pak by to velice zpomalovalo nasazení. Důvodem je to, že by se prováděly kontroly obou řešení místo jednoho. Pro toto nasazení má Ansible modul pro komunikaci s Gitem.

Ansible s využitím Gitu:

```
- name: download wp files via git
  git: repo={{ item.item.core_path_git }} dest={{ item.item.path
     }} version={{ item.item.core_version }}
  with_items: check_download_git.results
  when: wordpress_installs_git and item.rc != 0
  sudo: yes
  sudo_user: {{ item.item.user }}
  tags: [configuration-git, wordpress-git, wordpress-core-git,
     wordpress-downloaded-git]
```

Ukázka kódu 3.29: Instalace jádra Wordpressu pomocí Git

Pro modul Git jsou dvě hlavní informace. První je repo, která určuje odkud se budou soubory stahovat. Druhá je cíl (dest), kam se soubory mají stáhnout. Vedlejší informace verze (version) určuje, jakou verzi má Git stahovat a díky tomu umožní jak aktualizaci, tak i stažení předchozí verze (downgrade).

Další úkoly budou pouhé kombinace předchozích s využitím nástroje Git.

3.5 Puppet

Nasazení pomocí Puppetu probíhá pomocí funkcí tohoto nástroje s využitím Linuxových příkazů, které stahují a pracují se soubory. Toto řešení obsahuje pouze implementaci Wordpressu bez rozšíření. Je tomu tak z technických důvodů, které jsou problémy s cloud rozhraním a není plná funkčnost reclassu pro Puppet. Proto se tato implementace skládá z vytvoření databáze a Wordpress souborů.

Jako v předchozích případech se nejdříve musí přidat součásti do master stroje:

```
classes:
- system.puppet.master.single
- system.puppet.master.module.wordpress
- system.hiera.storage.puppet
parameters:
  _param:
```

```
    hiera_data_repository:
       git@repo1.robotice.cz:hiera-models/dels-puppet-model.git
    hiera_data_revision: master
```

Ukázka kódu 3.30: Další master součásti (Puppet)

Součást hiera se využívá jako částečná náhrada za reclass, a proto potřebuje podobné informace jako reclass. Znovu se zadává i odkaz na modul, který nainstaluje Wordpress na miniona. Po dokončení nastavení mastera je načase přidat součásti minionovi.

Nastavení součástí Puppet miniona:

```
node:
  wordpress_web_ha_puppet_stg:
    name: web02
    domain: wordpress-ha.dls.vpc.cloudlab.cz
    classes:
    - system.linux.system.single
    - system.openssh.client.cluster.cloudlab
    - system.openssh.server.team.tcpcloud
    - system.puppet.agent.master
    - system.ansible.client.single
    params:
      salt_master_host: ${_param:reclass_config_master}
      puppet_agent_master_host: ${_param:reclass_config_master}
      puppet_agent_report_host: ${_param:reclass_config_master}
```

Ukázka kódu 3.31: Minion součásti (Puppet)

Většina využitých tříd je stejná jako v předchozích případech. Jediná nová je pro Puppet agenta (miniona). Pro miniona je také důležité zajistit možnost komunikace s master strojem, proto je potřeba nastavit IP adresu master stroje. Tato adresa je stejná jako v případě Saltu a díky tomu se nemusí definovat dvakrát.

Tyto úpravy umožnily nastavení součástí pro oba stroje. Další úkol je vytvořit modul, který nainstaluje Wordpress na miniona. Puppet využívá definování tříd podobně jako PHP, a proto se musí nejdříve takové třídy vytvořit jak pro Wordpress, tak pro databázi.

Třída pro Wordpress v Puppetu:

```
define wordpress::instance::app ( $parametry ) {
- Zde budou kontroly parametrů a vytvoření WP souborů.
}
```

Ukázka kódu 3.32: Vytvoření třídy pro Wordpress (Puppet)

Stejný způsob by se použil pro vytvoření tříd pro rozšíření a šablony. Po vytvoření tříd se zpracuje jejich plnění parametry a dalšími funkcemi. Instalace jako taková se provede pouze zavoláním třídy s parametry.

3.5.1 Vytvoření Wordpress souborů

Tato část se nachází ve třídě pro Wordpress a obsahuje kontrolu, jestli jsou všechny důležité parametry nastaveny a poté zajistí samotné vytvoření Wordpress složky se soubory. Kontroluje se také, jestli má proměnná správný formát. Některé proměnné mohou být pouze pravda/nepravda (boolean), jiné třeba textové (string). Po kontrolách se přechází ke stažení Wordpress souborů.

Stažení souborů s nástrojem Puppet:

```
exec { "Download Wordpress
   ${install_url}/wordpress-${version}.tar.gz to ${install_dir}":
  command => "wget ${install_url}/wordpress-${version}.tar.gz",
  creates => "${install_dir}/wordpress-${version}.tar.gz",
  require => File[$install_dir],
  user    => $wp_owner,
  group   => $wp_group,
}
```

Ukázka kódu 3.33: Stažení Wordpressu (Puppet)

Stažení probíhá pomocí Linuxového příkazu wget, který stahuje z HTTP, HTTPS nebo FTP. Tomuto kódu se definuje možnost creates, podobně jako v Ansible, protože lokálně vytváří stažený soubor. Další proměnné určují uživatele, pod kterým je soubor vytvořen a jestli má práva pro zápis do určené složky. A stejně jako v Ansible je možné tuto funkci pojmenovat, aby bylo jasné, že se právě zpracovává.

Soubor, který se stahuje, má koncovku tar.gz, což říká, že je to archiv. Když je to archiv, tak se musí nejdříve rozbalit. K tomu se využije Linuxová funkce tar. Následně se změní vlastník souborů pomocí příkazu chown. Po těchto úpravách jsou připravené Wordpress soubory ve složce, která byla určena v proměnných, a proto se může přejít k vytvoření souboru wp-config.

3.5.2 Vytvoření souboru wp-config

V Pupetu se využívá šablonovací systém, proto je vytvořena šablona souboru wp-config, stejně jako tomu bylo v případě Saltu. Tato šablona se poté naplní informacemi z proměnných. Díky tomu se nemusí soubor nastavovat ručně a je možné ho velice snadno upravit.

Vytvoření souboru wp-config pomocí šablony v Puppetu:

```
concat::fragment { "${install_dir}/wp-config.php body":
  target => "${install_dir}/wp-config.php",
  content => template('wordpress/wp-config.php.erb'),
  order  => '20',
}
```

Ukázka kódu 3.34: Vytvoření souboru wp-config

S použitím concat::fragment se dá složit více souborů do jednoho kompletního a přesunout do umístění ve kterém má být. Samotné vložení informací do souboru zajišťuje funkce template. Pokud se skládá více souborů dohromady je potřeba určit v jakém pořadí se mají složi. K tomuto řazení se nastavuje proměnná order. Po dokončení těchto funkcí jsou kompletní soubory Wordpressu a jediné co zbývá je instalace databáze.

3.5.3 Instalace databáze

Pro vyvoření MySQL databáze se může využít Linuxových příkazů, ale Puppet má pro tuto práci rozšíření, které usnadňuje práci s databází. Tento modul má mnoho funkcí, ale v tomto případě se použije pouze pro vytvoření uživatele a databáze.

Vytvoření databáze pomocí Puppet modulu:

```
mysql_database { "${db_host}/${db_name}":
  name => $db_name,
  charset => 'utf8',
}
```

Ukázka kódu 3.35: Vytvoření databáze

Jediné co je potřeba nastavit v tomto příkazu je jméno databáze, které se musí shodovat se jménem uvedeným v souboru wp-config.

Vytvoření uživatele do databáze se používá nejčastěji pro omezení práv, aby uživatel mohl upravovat jen svoji přidělenou databázi a nezasahoval do žádné jiné. V tomto případě ale není problém vytvořit uživatele s právy administrátora, protože web je pouze jeden na serveru a také je pouze pro testování, nikoliv pro produkční využití.

Vytvoření uživatele databáze a nastavení jeho práv:

```
mysql_user { "${db_user}@${db_host}":
    password_hash => mysql_password($db_password),
}

mysql_grant { "${db_user}@${db_host}/${db_name}.*":
  table      => "${db_name}.*",
  user       => "${db_user}@${db_host}",
  privileges => ['ALL'],
}
```

Ukázka kódu 3.36: Vytvoření uživatele

Po spuštění tohoto kódu by měl být na určeném stroji plně funkční Wordpress. Pro dokončení instalace je ovšem třeba naplnit databázi základními daty. Toho lze dosáhnout pomocí zadání url adresy webu do prohlížeče nebo vložením dat ze souboru pomocí Linuxového příkazu, jako tomu bylo v Saltu.

3.5.4 Rozšíření a šablony

Instalace rozšíření a šablon tímto způsobem je velice podobná instalaci pomocí WP-CLI, proto by se tohoto rozšíření dalo využít a ovládat přes Linuxové příkazy podobně jako v případě Ansible. Šlo by tedy o vytvoření tříd, kontrolu proměnných a volání funkcí WP-CLI.

4 Testování

Testuje se hlavně rychlost nasazení pro jednotlivé nástroje. Bezpečnost nasazení je u všech nástrojů velice podobná, proto není nijak testována. Z důvodu problémů s cloud prostředím je část testování provedena na lokálním virtuálním Linux serveru, stejném jako je v cloud prostředí.

4.1 Rychlost nasazení

První byl proveden test rychlosti instalace rozšíření pomocí WP-CLI a Gitu. V tomto testu se vícekrát instalovalo jedno rozšíření do určené složky. Je jasně vidět, že instalace pomocí

Tabulka 4.1: Test rychlosti WP-CLI vs Git

	WP-CLI	Git
První test	37 sekund	1 minuta 32 sekund
Druhý test	21 sekund	1 minuta 16 sekund
Třetí test	26 sekund	1 minuta 22 sekund
Průměr	28 sekund	1 minuta 23 sekund

WP-CLI je třikrát rychlejší než pomocí Gitu. Na druhou stranu Git umožňuje lepší práci s verzemi a je možné využít soukromý Gitlab. Dalším testem je rychlost nasazení pomocí jednotlivých nástrojů. Budou provedeny tři různé druhy testů a všechny budou využívat kódu fungujícím pomocí WP-CLI příkazů, protože je rychlejší.

Druhy prováděných testů:

1. Instalace Wordpressu bez rozšíření a šablon.
2. Instalace s 19 vybranými rozšířeními.
3. Aktualizace 10 vybraných rozšíření.

Výsledky jsou zapsány v tabulce níže:

Tabulka 4.2: Rychlost jednotlivých nástrojů

	Salt	Ansible	Puppet
Instalace bez rozšíření	42 sekund	55 sekund	1 minuta
Instalace s rozšířeními	3 minuty 36 sekund	4 minuty 3 sekundy	x
Aktualizace 10 rozšíření	2 minuty	2 minuty 21 sekund	x

Testy pro Puppet bohužel nejsou kompletní, ale i tak je jasné, že nasazení pomocí Saltu je nejrychlejší. Ansible je velice zpomalovaný jeho nutností provádět dva úkoly pro instalaci a aktualizaci jedntlivých rozšíření. Salt provádí pouze jeden takový úkol.

4.2 Bezpečnost nasazení

Komunikace a autentifikace počítačů v síti probíhá pomocí SSH klíčů, takže v této bezpečnosti není problém. Navíc jsou stroje uzavřeny ve své vlastní síti a přístupné jsou pouze soubory webu a ovládající stroj.

Pár problémů s bezpečností může nastat a to při špatném vyplnění dat do konfiguračních yml souborů nebo u chyby při instalaci Wordpressu. Pokud například instalace nedokončí plnění dat do databáze, pak se zobrazí pouze bílá stránka nebo stránka s konfigurací Wordpressu. Tyto chyby pak mohou způsobit problém s bezpečností.

5 Závěr

V této části jsou shrnuty zjištěné informace a také možnosti dalších rozšíření stávajícího řešení.

Z dřívějších prací vím, jak náročné a unavující je kopírování stejných webů, jejich kompletní kontrola a zprovoznění. Proto jsem chtěl najít jednodušší možnost a nasazení Wordpressu bylo určeno z důvodu, protože s ním mám největší zkušenost.

První otázka byla, jestli se vůbec dá nasadit Wordpress pomocí konfiguračního managementu. Odpověď byla jasná téměř hned, protože již byly zpracované nějaké postupy, jak tento systém nasadit. Další otázka byla jaké nástroje softwarového konfiguračního managementu by bylo vhodné využít. Po pár hodinách hledání informací a konzultacích bylo jasné, že tyto nástroje budou Salt, Ansible a Puppet.

Pro vybrané nástroje se poté musely vytvořit postupy, jak se má Wordpress nainstalovat a jaké prostředky k tomu použít. Nejlepší možnost je využít nástrojů Git a WP-CLI, které dokáží zpracovat celou instalaci do pár minut.

Poslední otázka byla, jak rychle tyto nástroje nasazují Wordpress. Pro nasazení jednoho webu to sice není moc důležité, ale pokud se jich bude nasazovat více, pak je vhodné použít co nejrychlejší nástroj. Podle testů i z vlastní zkušenosti mohu říci, že se Saltem se pracuje nejlépe.

5.1 Možnosti rozšíření

Možnost rozšíření je hlavně ve využití nástroje Git a Gitlabu. Je možné využít Gitlab a stahovat do něj nejnovější verze rozšíření a šablon. Díky tomu se dají rozdělovat verze, kontrolovat aktualizované soubory a hlavně využít placené rozšíření a šablony. Pokud jsou rozšíření placené, pak nejsou dostupné na Githubu a jsou k dispozici pouze pro ruční stažení po registraci. Proto by se stahovaly a následně vkládaly na Gitlab, který je soukromý.

V tomto řešení pouze Ansible umožňuje spustit jednu součást celého kódu, jako je aktualizace. Bude vhodné tuto možnost zapracovat i do ostatních nástrojů.

Literatura

[1] KOMÁREK, Aleš a SOBĚSLAV, Vladimír. *OpenSource Automation in Cloud Computing*. In *Proceedings of the 4th International Conference on Computer Engineering and Networks*. Springer International Publishing, 2015, s. 805-812, ISBN 978-3-319-11103-2.

[2] HEDENGREN, Thord Daniel. *Smashing WordPress: Beyond the Blog, 4th Edition*. John Wiley and Sons, 2012, ISBN 978-1-118-60075-7.

[3] ELOS Technologies s.r.o. *Správa konfigurací* [online]. 2014 [cit. 2014-10-23]. Dostupné z: `http://www.elostech.cz/cm`

[4] ČERMÁK, Miroslav. *ITIL tajemství zbavený* [online]. 2009 [cit. 2014-08-25]. Dostupné z: `http://www.cleverandsmart.cz/itil-tajemstvi-zbaveny/`

[5] BROUSE, Peggy S. *Configuration management* [online]. 2002 [cit. 2014-08-25]. Dostupné z: `http://www.eolss.net/sample-chapters/c15/E1-28-03-02.pdf`

[6] Carnegie Mellon University. *Configuration management* [online]. 2009-2012 [cit. 2014-08-25]. Dostupné z: `http://www.sei.cmu.edu/productlines/frame_report/config.man.htm`

[7] FARAG, Joe. *What Is the Real Difference between Software Configuration Management and Hardware Configuration Management?* [online]. 2013 [cit. 2014-08-25]. Dostupné z: `http://www.cmcrossroads.com/article/what-real-difference-between-software-configuration-management-and-hardware-configuration`

[8] SmallCart Systems LLC. *What is ITIL v3?* [online]. 2008 [cit. 2014-08-25]. Dostupné z: `http://www.smallcart.com/itilv3.html`

[9] BRYANT, Christian. *A Guide to Open Source Cloud Computing Software* [online]. 2014 [cit. 2014-08-25]. Dostupné z: `http://www.tomsitpro.com/articles/open-source-cloud-computing-software,2-754-10.html`

[10] SCHALLER, Byron. *Infrastructure Management with SaltStack: Part 1 – The Setup* [online]. 2014 [cit. 2014-08-27]. Dostupné z: http://vbyron.com/blog/infrastructure-management-saltstack-part-1-getting-started/

[11] SCHALLER, Byron. *Infrastructure Management with SaltStack: Part 2 – Grains, States, and Pillar* [online]. 2014 [cit. 2014-08-27]. Dostupné z: http://vbyron.com/blog/infrastructure-management-saltstack-part-2-grains-states-pillar/

[12] SaltStack Inc. *SaltStack* [online]. 2015 [cit. 2014-08-27]. Dostupné z: http://saltstack.com/

[13] SaltStack Inc. *SaltStack Dokumentace* [online]. 2015 [cit. 2014-08-27]. Dostupné z: http://docs.saltstack.com/en/latest/

[14] HATCH, Tom. *Systems and Configuration Management Tools with SaltStack* [online]. 2013 [cit. 2014-08-27]. Dostupné z: https://www.youtube.com/watch?v=yphLKSjnSU8

[15] Ansible, Inc. *Ansible* [online]. 2015 [cit. 2014-08-27]. Dostupné z: http://www.ansible.com/home

[16] Ansible, Inc. *Ansible Dokumentace* [online]. 2015 [cit. 2014-08-27]. Dostupné z: http://docs.ansible.com/

[17] OGENSTAD, Patrick.*What is Ansible?* [online]. 2015 [cit. 2014-08-27]. Dostupné z: http://networklore.com/ansible/

[18] Puppet Labs. *PuppetLabs* [online]. 2015 [cit. 2014-08-27]. Dostupné z: https://puppetlabs.com/

[19] KANIES, Luke.*Puppet Introduction* [online]. 2012 [cit. 2014-08-27]. Dostupné z: http://www.aosabook.org/en/puppet.html

[20] Puppet Labs. *Puppet Dokumentace* [online]. 2015 [cit. 2014-08-27]. Dostupné z: https://docs.puppetlabs.com/

[21] WordPress Foundation. *Wordpress About* [online]. 2003-2015 [cit. 2014-08-28]. Dostupné z: https://wordpress.org/about/

[22] WELLER, Nathan B.*How To Start A WordPress Management Business In Less Than A Day* [online]. 2014 [cit. 2014-08-28]. Dostupné z: http://www.elegantthemes.com/blog/tips-tricks/how-to-start-a-wordpress-management-business-in-less-than-a-day

[23] WordPress Foundation. *Wordpress Blog* [online]. 2003-2015 [cit. 2014-08-28]. Dostupné z: https://wordpress.org/news/

[24] Q-Success. *WordPress Plugin Repository Now Hosts Over 30,000 Plugins* [online]. 2014 [cit. 2014-08-28]. Dostupné z: http://wptavern.com/wordpress-plugin-repository-now-hosts-over-30000-plugins

[25] *Usage of content management systems for websites* [online]. 2014 [cit. 2014-08-28]. Dostupné z: http://w3techs.com/technologies/overview/content_management/all

[26] Automattic Inc. *How to modify wordpress themes the smart way* [online]. 2013 [cit. 2014-08-28]. Dostupné z: http://themeshaper.com/modify-wordpress-themes/

[27] WordPress Foundation. *Wordpress Dokumentace* [online]. 2003-2015 [cit. 2014-08-28]. Dostupné z: https://codex.wordpress.org/

[28] SiteGround.com. *Tutorial on How to Use WP-Cli* [online]. 2004-2015 [cit. 2014-08-28]. Dostupné z: https://www.siteground.com/tutorials/wordpress/wp-cli.htm

[29] BACHHUBER, Daniel.*WP-CLI Informace a Dokumentace* [online]. 2015 [cit. 2014-08-28]. Dostupné z: http://wp-cli.org/

[30] Opensource.com. *What is OpenStack?* [online]. 2014 [cit. 2015-03-29]. Dostupné z: http://opensource.com/resources/what-is-openstack

[31] BRUCE, James. *What Is Git and Why You Should Use Version Control If You're a Developer* [online]. 2012 [cit. 2015-04-03]. Dostupné z: http://www.makeuseof.com/tag/git-version-control-youre-developer/

[32] GitLab B.V. *About Git* [online]. 2011-2015 [cit. 2015-04-20]. Dostupné z: https://about.gitlab.com/

[33] KRAFFT, Martin f. *Reclass — Recursive external node classification* [online]. 2013 [cit. 2015-03-29]. Dostupné z: http://reclass.pantsfullofunix.net/

[34] BIRK, Volker. *YML – Why a Markup Language?!* [online]. 2007-2013 [cit. 2015-03-29]. Dostupné z: `http://fdik.org/yml/`

[35] OpenBSD *About OpenSSH* [online]. 1999-2009 [cit. 2015-03-29]. Dostupné z: `http://www.openssh.com/`

[36] Tvorba-webu.cz. *PHP /základy/* [online]. 2003-2008 [cit. 2015-04-08]. Dostupné z: `http://www.tvorba-webu.cz/php/`

[37] Tvorba-webu.cz. *MySQL, SQL a PHP* [online]. 2003-2008 [cit. 2015-04-08]. Dostupné z: `http://www.tvorba-webu.cz/php/mysql.php`

[38] Apache Software Foundation. *PHP /základy/* [online]. 1997-2015 [cit. 2015-04-08]. Dostupné z: `http://httpd.apache.org/`

[39] SaltStack Inc. *Operations on regular files, special files, directories, and symlinks* [online]. 2015 [cit. 2015-03-18]. Dostupné z: `http://docs.saltstack.com/en/latest/ref/states/all/salt.states.file.html`

[40] Alexis. *Git, Feature Branches, and Jenkins – or how I learned to stop worrying about broken builds* [online]. 2011 [cit. 2015-04-20]. Dostupné z: `http://twasink.net/2011/09/20/git-feature-branches-and-jenkins-or-how-i-learned-to-stop-worrying-about-broken-builds/`

[41] WEISSIG, Justin. *Episode 43. - 19 Minutes With Ansible (Part 1/4)* [online]. 2015 [cit. 2015-04-20]. Dostupné z: `https://sysadmincasts.com/episodes/43-19-minutes-with-ansible-part-1-4`

[42] Novo Solutions. *Service Desk Software* [online]. 2012 [cit. 2015-04-20]. Dostupné z: `http://www.novosolutions.com/service-desk-software/`

[43] Quattor.org *Quattor Dokumentace* [online]. 2015 [cit. 2015-04-24]. Dostupné z: `http://www.quattor.org/documentation/`

[44] HP Labs *SmartFrog* [online]. 2015 [cit. 2015-04-24]. Dostupné z: `http://www.smartfrog.org/display/sf/SmartFrog+Home`

[45] Slashdot Media. *Software Testing Automation Framework (STAF)* [online]. 2015 [cit. 2015-04-24]. Dostupné z: `http://staf.sourceforge.net/current/STAFUG.htm`

[46] FROGCP *WHAT IS PHP?* [online]. 2015 [cit. 2015-04-25]. Dostupné z: `https://www.frogcp.com/php-hosting-with-mysql`

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Rokitanského 62, 500 03 Hradec Králové, tel: 493 331 111, fax: 493 332 235

Zadání k závěrečné práci

Jméno a příjmení studenta: **David Sucharda**

Obor studia: Aplikovaná informatika
Jméno a příjmení vedoucího práce: **Aleš Komárek**

Název práce:
Implementace redakčního systému Wordpress pomocí Konfiguračního Managementu

Název práce v AJ:
Implementation of CMS Wordpress using the Configuration Management

Podtitul práce:

Podtitul práce v AJ:

Cíl práce: Cílem bakalářské práce je zjistit, jestli je možné nasazovat redakční systémy pomocí konfiguračního managementu a rychlost i spolehlivost implementace.

Osnova práce:
1. Úvod
2. Vymezení základních pojmů
3. Seznámení s konfiguračním managementem
4. Nasazení aplikace pomocí konfiguračního managementu
5. Diskuze a závěr
6. Použitá literatura

Projednáno dne:

Podpis studenta Podpis vedoucího práce