Automatizovaný deployment aplikací na základě scénáře

Marek Cigánek

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav bezpečnostního inženýrství

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Marek Cigánek**
Osobní číslo:	**A19825**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Bezpečnostní technologie, systémy a management**
Forma studia:	**Kombinovaná**
Téma práce:	**Automatizovaný deployment aplikací na základě scénáře**
Téma práce anglicky:	**Automated Playbook-based Deployment of Applications**

Zásady pro vypracování

1. Proveďte literární rešerši na dané téma.
2. Shromážděte požadavky pro automatizaci nasazování aplikací.
3. Navrhněte automatizované technické řešení nasazování aplikací.
4. Zdůvodněte výběr jednotlivých komponentů technického řešení.
5. Realizujte a otestujte výsledné technické řešení ve spolupráci s uživatelem.
6. Věnujte pozornost zabezpečení a práci s hesly.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. Ansible. Ansible [online]. 2021 [cit. 2021-10-01]. Dostupné z: https://www.ansible.com/
2. Vault. Vault [online]. 2021 [cit. 2021-10-01]. Dostupné z: https://www.vaultproject.io/
3. Design and use of virtualization technology in cloud computing [online], 2017. Hershey PA, USA: IGI Global [cit. 2021-10-01]. ISBN 9781522527862. Dostupné z: doi:10.4018/978-1-5225-2785-5
4. HOCHSTEIN, Lorin a René MOSER. Ansible: up and running: automating configuration management and deployment the easy way. Second edition. Beijing: O'Reilly, 2017. ISBN 978-1-4919-7980-8.
5. UPHILL, Thomas. Puppet 5 cookbook: jump-start your puppet 5.x deployment using engaging and practical recipes [online]. Fourth edition. Birmingham, London: Packt, 2018 [cit. 2021-10-01]. ISBN 978-1-78862-750-4. Dostupné z: https://ebookcentral.proquest.com/lib/natl-ebooks/detail.action?docID=5439841.

Vedoucí bakalářské práce: **doc. Ing. Jiří Vojtěšek, Ph.D.**
Ústav řízení procesů

Konzultant bakalářské práce: **Ing. Ivan Masár**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **17. ledna 2022**
Termín odevzdání bakalářské práce: **31. května 2022**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

Ing. Jan Valouch, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 17. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 24. 5. 2022

Marek Cigánek v.r.
podpis studenta

ABSTRAKT

Bakalářská práce se zabývá problematikou nasazování aplikací v rámci knihovny Univerzity Tomáše Bati ve Zlíně. Identifikuje a analyzuje jednotlivé aplikace používané v rámci knihovny, jejich současný stav. Zabývá se výběrem vhodné aplikace konfiguračního managementu, která by umožnila řešení automatizovaného nasazování aplikací. Automatizace nasazování aplikací umožní zjednodušení činnosti správce IT infrastruktury knihovny, nasazování aplikací ve vývojovém a testovacím prostředí. Práce se dále zabývá návrhem a realizací řešení jednotlivých komponent a možnosti jejího využití v rámci připravených scénářů pro vybrané aplikace. Dále se také soustředí na práci s hesly, současné možnosti centrální správy hesel a jejich využití v rámci zvoleného konfiguračního managementu.

Klíčová slova: ansible, aplikace, bezpečnost, nasazování aplikací, konfigurační management, scénář

ABSTRACT

The bachelor thesis deals with the issue of application deployment within the library of the Tomas Bata University in Zlín. It identifies and analyses the individual applications used within the library and their current status. It deals with the selection of a suitable configuration management application that would enable the solution of automated application deployment. Automation of application deployment will allow simplifying the activities of the library IT infrastructure administrator, deploying applications in the development and testing environment. The thesis also deals with the design and implementation of the solution of individual components and the possibility of its use within prepared scenarios for selected applications. It also focuses on the work with passwords, the current possibilities of price-permanent password management and their use within the selected configuration management.

Keywords: ansible, applications, security, application deployment, configuration management, playbook

Rád bych touto cestou poděkoval vedoucímu práce panu doc. Ing. Jiřímu Vojtěškovi, Ph.D. za odborné vedení a konzultace. Dále bych rád poděkoval panu Ing. Ivanu Masárovi z IT oddělení knihovny UTB za spolupráci a připomínky během vypracování této bakalářské práce. Také bych chtěl poděkovat rodině a přátelům za podporu během studia.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Automatizovaný deployment aplikací je stále aktuální téma v oboru informačních technologií. V minulosti se úkony řešily manuálně bez pomoci specializovaných aplikací. Automatizace může tuto činnost výrazně zjednodušit, aby nedocházelo tak často k chybám způsobeným během aktualizací produkčních systémů. Nasazení automatizace eliminuje chybovost lidského faktoru během provádění těchto změn. Usnadní a urychlí implementaci plánovaných změn do produkčního prostředí.

Cílem práce je analyzovat, navrhnout a vytvořit řešení, které usnadní nasazování aplikací do IT infrastruktury knihovny Univerzity Tomáše Bati. Knihovna má vlastní IT oddělení, které spravuje knihovní aplikace určené pro akademické pracovníky, zaměstnance, studenty univerzity a odbornou veřejnost.

V teoretické části práce jsou rozebírány jednotlivé aplikace konfiguračního managementu, které vedly ke vzniku jednotlivých nástrojů a historickým milníkům jejich vývoje. Dále se tato práce zabývá jejich architekturou a komponenty, které poskytují nástroje konfiguračního managementu. Dále pak shrnuje rozdíly mezi jednotlivými nástroji. Práce se též zabývá životním cyklem aplikací, jejichž součástí je automatizovaný deployment. V teoretické části zmiňuji běžně používané metody nasazování jednotlivých aplikací.

V praktické části je analyzován současný stav nasazování aplikací v prostředí IT oddělení knihovny. Jsou zkoumány požadavky jednotlivých aplikací na instalace a jejich zabezpečení. Na základě analýzy se vyhodnotí požadavky a provede se výběr vhodného řešení. Práce se dále zabývá komponenty a realizací celého řešení a jeho testováním. S ohledem na kybernetickou bezpečnost praktická část také řeší problematiku bezpečné komunikace s aplikacemi a správu hesel.

I. TEORETICKÁ ČÁST

1 KONFIGURAČNÍ MANAGEMENT

Kapitola se zabývá přehledem terminologie používané v konfiguračním managementu, rozebírá nejvýznamnější aplikace v této kategorii.

Následuje popis existujících řešení, jejich specifické rysy, historii, architekturu a bezpečnost. V závěru kapitoly se nalézá souhrn důležitých vlastností, výhody a nevýhody oproti ostatním řešením.

1.1 Přínos konfiguračního managementu

Konfigurační management zjednodušuje práce systémovým administrátorům a inženýrů při řešení opakujících se úkonů systémové administrace. Typickým příkladem mohou být různé instalace programových balíků, konfigurace systémových nebo programových nastavení. Cílem je eliminovat možné chyby způsobené lidským faktorem při provádění instalací nebo manuálních úpravách konfiguračních souborů.

Autor [1] uvádí, že " *Správa konfigurace je proces sledování a kontroly změn softwaru s ohledem na jeho požadavky, návrh, funkci a vývoj produktu.*[1]" (překlad autora z [1])

Firma Red Hat na svých webových stránkách [2] uvádí, že: "*Správa konfigurace je proces udržování počítačových systémů, serverů a softwaru v požadovaném, konzistentním stavu. Je to způsob, jak zajistit, aby systém fungoval tak, jak se od něj očekává, když se v průběhu času provádějí změny.*[2]" (překlad autora z [2])

Dříve se tyto úkony řešily často pomocí skriptovacích jazyků typických pro daný operační systém. Na operačním systému Linux systémoví administrátoři velmi často používají pro automatizaci opakujících se úkonů skriptovací jazyky typu shell, korn shell nebo bash dále v minulosti programovací jazyk Perl a v dnešní době velmi oblíbený a populární jazyk Python. K automatizaci jednotlivých kroků se v minulosti také používala na unixových a linuxových systémech aplikace expect.

---

[1] Configuration management is the process of tracking and controlling the changes in a software with respect to its requirement, design, function, and development of a product.

[2] Configuration management is a process for maintaining computer systems, servers, and software in a desired, consistent state. It's a way to make sure that a system performs as it's expected to as changes are made over time.

Na systémech Windows jsou velmi často používány pro automatizaci tzv. batch skripty (.bat) nebo moderní jazyk PowerShell. [3][4]

Jedná se také o značně manuální interakci s napsaným programem, kde programátor musí ošetřit možné chyby, podporu operačních systémů, kde bude program nasazován.

Dále administrátor musí řešit, jak se bude skript distribuovat na koncové stanice, jak se bude spouštět, zda manuálně nebo za pomocí nějakého plánovače úloh, jak se zajistí odeslání informace o výsledku zpět k administrátorovi systému.

Důvodem, proč použít specializovaný nástroj na automatizaci je zjednodušení práce administrátora, snížení chybovosti vykonávaných administrativních činností, možnost otestování aplikace nebo nové konfigurace v testovacím prostředí.

Z výše uvedených důvodů lze konstatovat, že konfigurační nástroje mají kladný přínos do oblasti systémové a aplikační administrace IT infrastruktury.

1.2 Základní termíny

Na začátku je nutné objasnit základní termíny používané v rámci konfiguračního managementu, které se dále vyskytují v textu.

1.2.1 Deklarativní vs imperativní programovací jazyk

Prvním důležitým termínem, který je nutný zmínit v rámci vytváření konfiguračních předpisů je rozdíl mezi deklarativním a imperativním programovacím jazykem. Ne všechny nástroje konfiguračního managementu podporují oba typy jazyků. Deklarativní programovací jazyky popisuje, jak by koncový stav nastavení měl vypadat.

Beneš v kurzu o funkcionálním programování na FEI VŠB (Fakulta Elektrotechniky a Informatiky Vysoká Škola Báňská) [5] uvádí, že:“*Deklarativní programovací jazyky vyjadřují hledané řešení obvykle ve formě výčtu vlastností, které by mělo řešení splňovat.*“ [5]

Následující příklad vytvoří adresář */tmp/my_dir* (řádek č.1) na operačním systému Linux, nastaví vlastníka na účet (řádek č.2) *root* a skupina *root* (řádek č.3). Na řádku č.4 se nastavují práva na tento adresář v numerické formě 0755, tzn. vlastník má nastavena práva na čtení, zápis a spuštění, skupina a ostatní mají nastavena práva na čtení a spuštění. Na řádku č.5 je typ akce, vytvoření adresáře a poslední řádek č.6 ukončuje tento předpis. [6]

```
1 directory ”tmp/my_dir” do
2     owner ”root”
```

```
3     group ″root″
4     mode 0755
5     action : create
6 end
```

Zpracováno z [6].

Na následujícím příkladu je pomocí imperativního jazyka popsána instalace balíku *tomcat6* řádky 1-3 a na řádcích 4-5 je definováno spuštění služby *tomcat6*. [6]

```
1 package "tomcat6" do
2      action : install
3 end
4 service "tomcat6" do
5     action [:start, :enable]
6 end
```

Zpracováno z [6].

Imperativní programovací jazyk Beneš [5] z VŠB definuje jako:“*Imperativní programovací jazyky jsou charakteristické zejména tím, že program má tvar posloupnosti příkazů, jejichž pořadí vyhodnocení je pevně stanoveno.*“ [5]

V následujícím příkladu je popsána instalace jazyka *php* pomocí imperativního jazyka v rámci interpreteru jazyka bash (řádek č.1). Blok 3-7 provede kroky pro rozbalení archivu zdrojových kódu jazyka php (řádek č.4), vytvoření konfiguračního souboru použitého pro kompilaci (řádek č.6). Na řádku č.7 dojde k překladu zdrojových souborů na binární a ukončení bloku na řádku č.8. K vykonání bloku 3-7 dojde pouze za předpokladu, když bude výsledek testu uvedeného na řádku č.8 negativní, příkaz *which php* vyhledá, zda příkaz existuje nebo nikoliv.[6]

```
1  bash "build php" do
2     cwd Config [: file cache path ]
3     code <<-EOF
4  tar -zxvf php-#{version}.tar.gz
5  cd php-#{version}
6  ./configure #{options}
7  make && make install
8  EOF
9     not_if "which php"
10 end
```

Zpracováno z [6].

1.2.2 Automatizace

Dalším důležitým pojmem je automatizace. Slovník [7] automatizaci definuje jako: "proces, při kterém je činnost člověka např. ve výrobě, službách nahrazována činností automat. tech. prostředků – automatů. Zbavuje člověka rutinní a namáhavé práce." [7]

V oblasti informačních technologií a této práci je termín automatizace použit ve spojení automatizovaný deployment aplikací. Tzn. provádění jednotlivých kroků za pomoci softwarového nástroje který usnadňuje výkon této činnosti.

1.2.3 Push vs Pull model

Pull a push modely patří mezi dva základní principy konfiguračního managementu a jejich komunikací mezi klientem nebo serverem. Metoda pull (táhnout) je metodou, kdy klient sám kontaktuje server, v daném časovém intervalu, aby zjistil, zda jeho konfigurace byla změněna nebo ne. Metoda push (tlačit) funguje obráceně. Server iniciuje spojení s klientem a aktualizuje jeho konfiguraci, pokud je nutná změna. [1]

1.2.4 Nasazování (Deployment)

Význam anglického výrazu deployment znamená nasazování. Tento termín je spojený s instalacemi operačních systémů a aplikací. Zabývá se jakým způsobem se aplikace nainstaluje a nakonfiguruje v daném prostředí operačního systému (Linux, Microsoft Windows). [8]

1.2.5 Idempotence

Význam slova idempotence pochází z matematiky a jeho význam je podle [9] následující: *"(v matematice) vlastnost operace taková, že vícenásobné provedení operace má vždy stejný výsledný efekt"*. [9]

V kontextu konfiguračního managementu toto znamená, že konfigurace může proběhnout opakovaně bez vedlejších účinků. To znamená, že konfiguraci, kterou se snažíme docílit určitého stavu, vykoná změny pouze v případě, že mají být provedeny. V ostatních případech nedojde k žádným změnám na provozovaném systému. [10]

2 KONFIGURAČNÍ NÁSTROJE

Kapitola se zabývá popisem jednotlivých konfiguračních nástrojů, jejich stručnou historií, důvody, které vedly autory k jejich vzniku. Dále popisuje jejich architekturu, jednotlivé komponenty a v neposlední řadě věnuje pozornost bezpečné komunikaci.

2.1 CFEngine

Prvním konfiguračním nástrojem, který je nutné zmínit je CFEngine. Jedná se o historicky nejstarší konfiguračním nástroj, ze kterého vychází konfigurační nástroj Puppet.

2.1.1 Historie

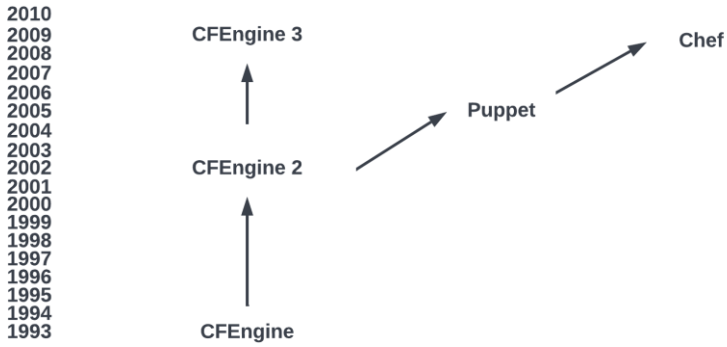
Zakladatelem projektu a hlavním autorem je emeritní profesor oboru síťové a systémové administrace na University College v Oslu, Dr. Mark Burgess. Je autorem knih, publikací a článků z oblasti síťové a systémové administrace, dále publikuje články o fyzice a fikci. Dále Burgess v [11] uvádí: *"V roce 1993 začal psát populární nástroj pro správu konfigurace CFEngine, který představil na konferenci HEPix v Paříži. V roce 1994 zaregistroval CFEngine u Free Software Foundation pod veřejnou licencí GNU.*[3]" (překlad autora z [11])

První verze CFEngine používal specializovaný jazyk umožňujícím vykonávat implicitní testy na bázi "if-then-else". Jazyk byl založený na třídách, určujících, jaký příkaz se má spustit na jakých systémech. [11][12]

Burgess ve verzi CFEngine 2 z roku 2002 aplikoval svůj výzkum na téma počítačové imunologie a konvergentní konfigurace. Dal dohromady myšlenku, že konfigurační management má provádět změny postupně a upravovat pouze to, co je nutné nastavit do předpokládaného stavu. Tato charakteristika ve velké míře zjednodušuje nasazování (deployment) aplikací a implementaci konfiguračního managementu. [12]

CFEngine 3 vydaný v roce 2009 byl Burgessem přepracován a implementoval zde jeho "teorii slibů" (promise theory) [14], tj. jak má systém nebo program fungovat podle námi definovaného stavu. Syntaxe jazyka byla také přepracována, aby odpovídala "teorii slibů". Vznik CFEngine 3 doprovázelo založení společnosti CFEngine AS poskytující komerční podporu toho programu a také její komerční verzi nazvané Enterprise. [12]

[3] He began writing the popular configuration management tool CFEngine in 1993, and presented it at the HEPix conference in Paris. In 1994 he registered CFEngine with the Free Software Foundation, under the GNU Public License.



Obrázek 1 Vývoj CFEngine a příbuzných nástrojů

[zdroj: upraveno z 13]

CFEngine 3 vydaný v roce 2009 byl Burgessem přepracován a implementoval zde jeho "teorii slibů" (promise theory) [14], tj. jak má systém nebo program fungovat podle námi definovaného stavu. Syntaxe jazyka byla také přepracována, aby odpovídala "teorii slibů". Vznik CFEngine 3 doprovázelo založení společnosti CFEngine AS poskytující komerční podporu toho programu a také její komerční verzi nazvané Enterprise. [12]

Příkladem slibu může být například to, že chceme, aby byl naistalován, nakonfigurován a následně spuštěn webový server Apache. Nepopisujeme dílčí kroky, ale pouze instalaci, konfiguraci a následné spuštění aplikačního serveru.

Community verze je volně dostupná ke stažení. Enterprise verze obsahuje rozšířenou podporu operačních systémů, REST API (Representational State Transfer Application Programming Interface), logování, monitorovaní a další funkce. [15]

CFengine je napsaný v jazyce C. Pro definování stavů používá DSL (Domain Specifc Language – Doménově specifické jazyky) jazyk, kde stavy mohou obsahovat instalace programů, správu uživatelských účtů, správu procesů, správu služeb atd. Jedná se o deklarativní jazyk. [16][17]

2.1.2 Architektura

CFEngine může fungovat zcela nezávisle bez nutnosti síťové komunikace nebo se používá jako kompletně distribuovaný systém řídící desítky, stovky nebo tisíce systémů.

Komponenty

Detailní komunikace mezi systémy označenými klient 1 a klient 2 a jednotlivými komponenty je zobrazena na Obr. 2.

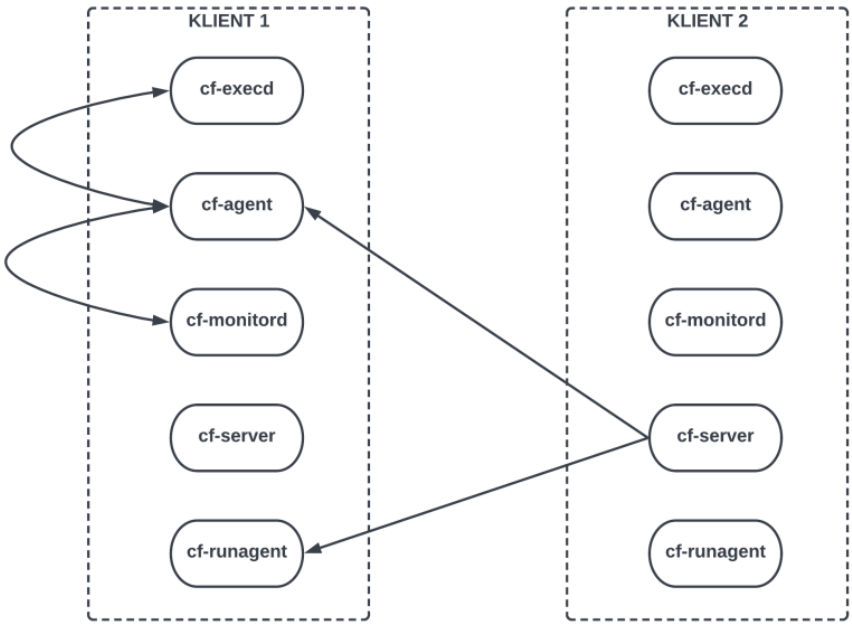
Jedná se o následující komponenty [18][19][20]:

cf-serverd

cf-serverd je server, který pracuje také jako souborový server. Dostává požadavky, jak vykonat politiku (policy) na jednotlivých klientech a spouští program cf-agent po přijetí spojení od cf-runagent. Službě cf-serverd se nepředávají žádná data.

cf-agent

cf-agent inicializuje změny nastavení podle příslibů (promises) na jednotlivých systémech. Může získávat data pro provedení změn od cf-serverd lokálně nebo ze vzdáleného systému.



Obrázek 2 Komunikace komponentů CFEngine [zdroj: upraveno z 20]

cf-execd

cf-execd je služba, démon, který spouští agenta cf-agent v pravidelných nebo uživatelsky definovaných intervalech. [20]

cf-runagent

cf-runagent je program, který může kontaktovat démona cf-serverd provozovaném na vzdáleném systému (komunikace zobrazena na Obr. 4), aby spustil agenta cf-agent. cf-runagent nemůže agenta cf-agent úkolovat.

cf-monitord

cf-monitord je program, který sbírá statistické informace o využití systémových prostředků na jednotlivých systémech. Tyto údaje slouží k monitorování a detekci případných anomálií.

cf-key

cf-key je služba, která generuje soukromý a veřejný SSL (Secure Socket Layer) certifikát pro zajištění bezpečné komunikace.

cf-report

cf-report vypisuje obsah databáze programu cf-agent v různých formátech.

2.1.3 Bezpečnost

CFEngine používá pro komunikaci proprietární protokol podobný protokolu OpenSSH. Je založen na vzájemné komunikaci typu výzva – odpověď (z anglického jazyka challenge – response) pomocí infrastruktury veřejných klíčů.

"*CFEngine Community Edition používá k ověřování šifrování veřejným klíčem RSA 2048. Ty se generují příkazem 'cf-key'. Pro přenos dat generuje 128bitový náhodný šifrovací klíč Blowfish. Odpověď na výzvu se ověřuje pomocí hashe MD5.*[4]" (překlad autora z [21])

2.2 Puppet

Puppet je dalším velmi známým a rozšířeným konfiguračním nástrojem, který je nasazován v rozsáhlých serverových infrastrukturách. Puppet vychází z konfiguračního nástroje CFEngine.

2.2.1 Historie

Autorem konfiguračního nástroje Puppet je Luke Kannies. Kannies začal pracovat na vývoji Puppet v roce 2001. A vedla ho tomu nespokojenost s nástroji pro konfiguraci systémů. Podle jeho blogu, než začal vyvíjet Puppet, nebyl vývojář, ale systémový administrátor, kdy napsal dohromady zhruba 5000 řádků kódu. Když v roce 2016 opouštěl firmu zdrojový kód obsahoval 130 000 řádků. Kennies založil v roce 2005 firmu Puppet Labs, kde byl do roku 2008 jediným zaměstnancem a přijal další dva zaměstnance, firma vydělávala 250 tisíc

[4] CFEngine Community Edition uses RSA 2048 public key encryption for authentication. These are generated by the 'cf-key' command. It generates a 128 bit random Blowfish encryption key for data transmission. Challenge response is verified by an MD5 hash.

dolarů ročně. Za sedm let měla firma pět set zaměstnanců a obrat 70 miliónů ročně. V roce 2016 Kennies oznámil odstoupení z pozice výkonného ředitele s tím, že firmu neopouští, ale zůstává ve vedení firmy. [22][23]

Puppet je volně dostupný k používání jako otevřený software pod licencí Apache nebo jako komerční produkt ve verzi Puppet Enterprise – PE. Puppet je dostupný pro operační systémy Linux, Windows, OS X, Solaris. Verze Enterprise nabízí oproti volné verzi podporu systémů AIX, F5, kontejnery Docker, virtuální stroje, reportování, RBAC (Role Based Access Control).

Puppet, stejně jako CFEngine používá deklarativní jazyk DSL. [24][25]

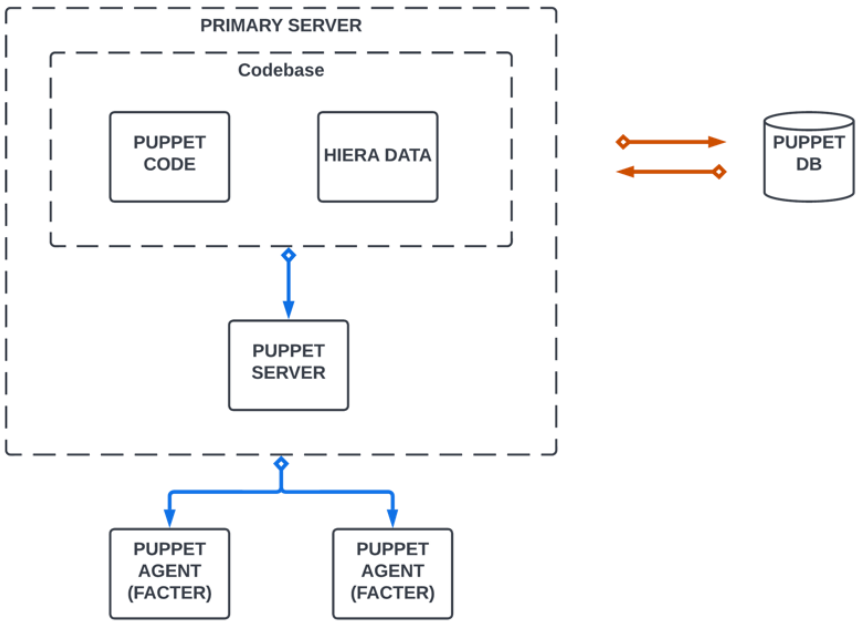
2.2.2 Architektura

Puppet je typickým představitelem architektury agent – server. Primární server komunikuje s agenty prostřednictvím protokolu HTTPS a SSL certifikátů. Součástí Puppet serveru je i integrovaná certifikační autorita – Puppet CA (Certification Authority), která podepisuje a spravuje SSL certifikáty. [26]

Komponenty

Na Obr. 3 jsou zobrazeny jednotlivé komponenty nástroje Puppet mezi, které patří:

- Puppet code (Manifests, Modules),
- Hiera,
- Facter,
- Puppet DB,
- Puppet Agent.



Obrázek 3 Komponenty nástroje Puppet [zdroj: upraveno z 27]

Manifesty (manifests)

Konfigurační předpisy definující požadovaný stav se v programu Puppet nazývají manifesty. [28]

Moduly (modules)

Manifesty se organizují do modulů, moduly obsahují třídy programu Puppet, definované typy, úlohy, plány úloh, funkce, typy, zprostředkovatele prostředků a zásuvné moduly (plugins), například vlastní typy nebo fakta.

Moduly mohou obsahovat například šablony souborů, které mají být uloženy na cílovém klientu. Pro tvorbu šablon se používá buď tzv. EPP (Embedded Puppet) nebo nativní šablony používané v jazyce Ruby tzv. ERB (Embeded Ruby). Manifesty se ukládají na primárním Puppet serveru. [29][30][31]

Hiera

Dalším důležitým komponentem je Hiera, která poskytuje pouze uchovávání dat, a lze s ní tvořit, jak název napovídá hierarchické struktury konfigurací. Příkladem může být nastavení od globální konfigurace pro všechny klienty až po individuální nastavení jednotlivých klientů. [32]

Facter

Dalším důležitým programem je Facter, program, který ukládá fakta o klientovi, jakými mohou být informace o hardware, systému, síťové nastavení a další. Facter umožňuje vytvářet vlastní data, která jsou dále k dispozici na klientovi a dají se použít v manifestech. [33]

Puppet DB

Puppet DB je databáze, kam se ukládají data z Puppet, daty mohou být například vygenerované reporty při běhu Puppet agentů na klientech. [34]

Agent

Puppet agent je program nainstalovaný na straně klienta, zodpovědný za nastavení klienta podle manifestů. Agent komunikuje s primárním Puppet serverem. Agent může běžet na klientovi jako služba, která aplikuje novou konfiguraci během pevně stanoveného intervalu například každých 15 minut. Tento parametr lze nastavit nebo lze agenta spouštět pomocí plánovače úloh v danou hodinu, minutu případně den v týdnu. Toho lze docílit na klientech s operačním systémem Linux pomocí programu plánovače úloh cronu, na operačních systémech Windows manuálně, případně vzdáleně pomocí programu Bolt. [35][36][37]

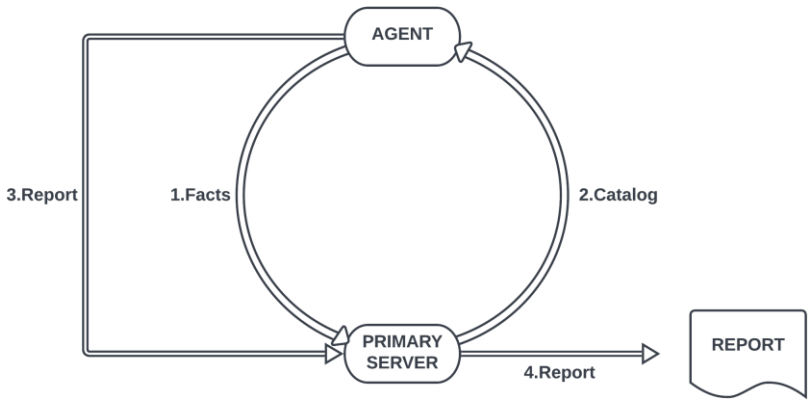
Na Obr. 4 je zobrazen průběh komunikačního cyklu mezi Puppet primárním serverem a klientem, na kterém je nainstalován a spuštěn Puppet agent.

Podle dokumentace projektu Puppet [38] agent pracuje následovně (Obr. 4):

1. Agent pošle fakta (Facts) primárnímu serveru a vyžádá si katalog. "*Katalog je dokument, který popisuje požadovaný stav systému pro jeden konkrétní počítač. Obsahuje seznam všech prostředků, které je třeba spravovat, a také všechny závislosti mezi těmito prostředky.*[5]" (překlad autora z [39])
2. Primární server přeloží katalog a pošle ho zpět agentovi. Agent následně zkontroluje nastavení každého prostředku (resource) a pokud není v požadovaném stavu, tak provede potřebnou změnu.
3. Agent zašle report o výsledku zpět Puppet serveru

[5] A catalog is a document that describes the desired system state for one specific computer. It lists all of the resources that need to be managed, as well as any dependencies between those resources.

4. Report je dále uložen například do Puppet DB. Puppet DB je databáze, která ukládá data jako jsou fakta a katalogy. [40]



Obrázek 4 Puppet Komunikace Server a Agent [zdroj: upraveno z 38]

Puppet Forge

Puppet Forge je platforma, která obsahuje moduly vytvořené společností Puppet, partnery a komunitou Puppet vývojářů. [41]

2.2.3 Bezpečnost

Puppet primární server, komunikuje s agenty prostřednictvím protokolu HTTPS a SSL certifikátů. Integrovaná certifikační autorita, Puppet CA, která je součástí Puppet serveru, podepisuje a spravuje SSL certifikáty. [26]

2.3 Chef

Dalším významným konfiguračním nástrojem je Chef a jeho podpůrné programové vybavení. Chef je specifický v pojmenování jednotlivých nástrojů podle kuchyně. Význam slova Chef přeložený do českého jazyka je šéfkuchař nebo vrchní šéf. Chef je napsán v jazyce Ruby a Erlang. Chef používá DSL jazyk. [42]

2.3.1 Historie

Adam Jacob spolu s jeho přáteli Nathanem Haneysmithem, Barryem Steinglassem, a jejich prvním zaměstnancem Joshua Timbermanem založil konzultační firmu, ze které později vznikla původní firma OpsCode, ze kterého vznikla později firma Chef. [43]

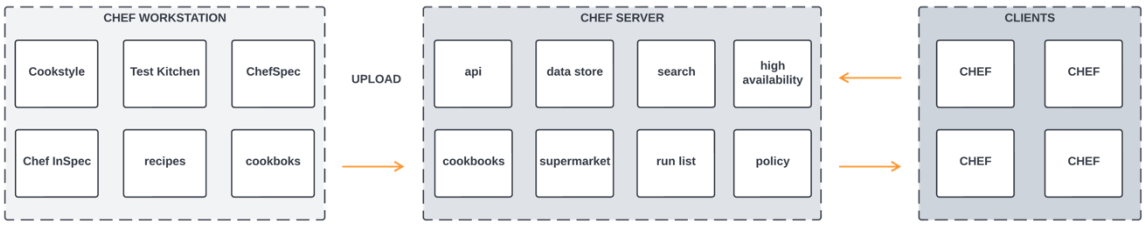
Jesse Robinsen oznámil 16.ledna 2008 konfigurační nástroj Chef.V roce 2020 firmu Chef koupila firma Progress. [44][45]

2.3.2 Architektura

Chef je stejně jako konfigurační nástroj Puppet představitelem architektury klient-server. Chef lze použít i samostatně bez složité infrastruktury prostřednictvím programu chef solo. Chef-solo je program, který se spouští lokálně a vykonává tzv. cookbooks (kuchařky). [46]

Komponenty

Mezi základní prvky patří Chef Workstation, Chef Server a klienti. Vztah mezi Chef Workstation a Chef Serverem a klienty je zobrazen na Obr. 5.



Obrázek 5 Chef komponenty [zdroj: upraveno z 47]

Chef Workstation

Chef Workstation je soubor nástrojů, který umožňuje práci s konfiguračním managementem Chef. V tomto prostředí se vytváří jednotlivé automatizační úlohy, tzv. recipe (recepty), které jsou organizovány do cookbooks (kuchařky). Chef Workstation obsahuje i testovací prostředí Test kitchen, které usnadňuje vývojáři testování jednotlivých cookbooks. Nástroj *knife* poskytuje mechanismus, jak nahrát cookbooks z lokálního chef repozitáře Chef Server. [48]

Recipe (recept)

Recipe je seznam jednotlivých úloh, které konfigurují klienta a ty se dále organizují do cookbook (kuchařek). [49]

Cookbook (kuchařka)

Cookbook obsahuje recipes (recepty), které mají být vykonány na Chef klientovi. Dále pak obsahují atributy, rozšiřující knihovny s funkcemi v jazyce Ruby, šablony a soubory a vlastní Ohai plugin moduly. [50]

Inspec

Nástroj Inspec slouží k testování a auditu aplikací. [51]

Knife (nůž)

Knife je další důležitou součástí Chef Workstation. Program poskytuje komunikaci s Chef Serverem, klienty (nodes) a nahrávání naprogramovaných cookbooks (kuchařek). Tato komunikace probíhá přes API. [52]

Test kitchen

Test kitchen je nástroj, který umožňuje testovat cookbooks (kuchařky) na libovolných platformách. Prostředí test kitchen poskytuje vývojáři možnost testování pomocí testů vytvořených v programu Inspec. [53]

Ohai

Ohai je nástroj, který shromažďuje informace o konfiguraci systému, informacemi jsou například síťová nastavení, velikost operační paměti, disku, procesor, doménové jméno atd. Tyto informace se dají využít v cookbooks (kuchařkách). [54]

Supermarket

Supermarket je aplikace, která slouží jako repozitář pro sdílení cookbooks (kuchařek). Chef Supermarket je dostupný ve veřejné formě nebo jako privátní server uvnitř organizace tzv. on-premise řešení. [55]

Chef Infra Server

Chef Infra Server spravuje centrálně konfigurační data. Na serveru se nachází cookbooks (kuchařky) a metadata klientů (nodes). API serveru je vytvořeno v jazyce Erlang. [56]

Node (klient)

Posledním důležitým komponentem je klient (v názvosloví Chef označovaný jako node) samotný, tím může být fyzické, virtuální, síťové nebo cloudové zařízení. Na klienta se aplikují cookbooks (kuchařky) podle definovaných atributů konfigurace. Klienti kontaktují Chef Intra Server, aby jim poskytl data o jejich konfiguraci – recipes (recepty), atributy, šablony souborů a soubory. [57]

2.3.3 Bezpečnost

Bezpečná komunikace mezi jednotlivými komponenty Chef ekosystému probíhá pomocí SSL certifikátů. [58][59]

2.4 Salt Stack

Dalším představitelem konfiguračního managementu je program Salt Stack napsaný v jazyce Python. Byl vyvinut s cílem poskytnout rychlé a bezpečné vykonávání příkazů na vzdálených klientech. [60]

Salt Stack na rozdíl od již zmíněných konfiguračních nástrojů jako je CFEngine, Puppet a Chef nepoužívá DSL, ale standardní datový formát YAML (YAML A‘int Markup Language ™ - Yaml není značkovací jazyk). [61]

Z tohoto pohledu není nutné znát specifický konfigurační jazyk nebo derivát skriptovacího jazyka Ruby jako v případě nástroje Chef. Salt Stack podporuje jak deklarativní, tak i imperativní styl vytváření konfiguračních úloh. [61]

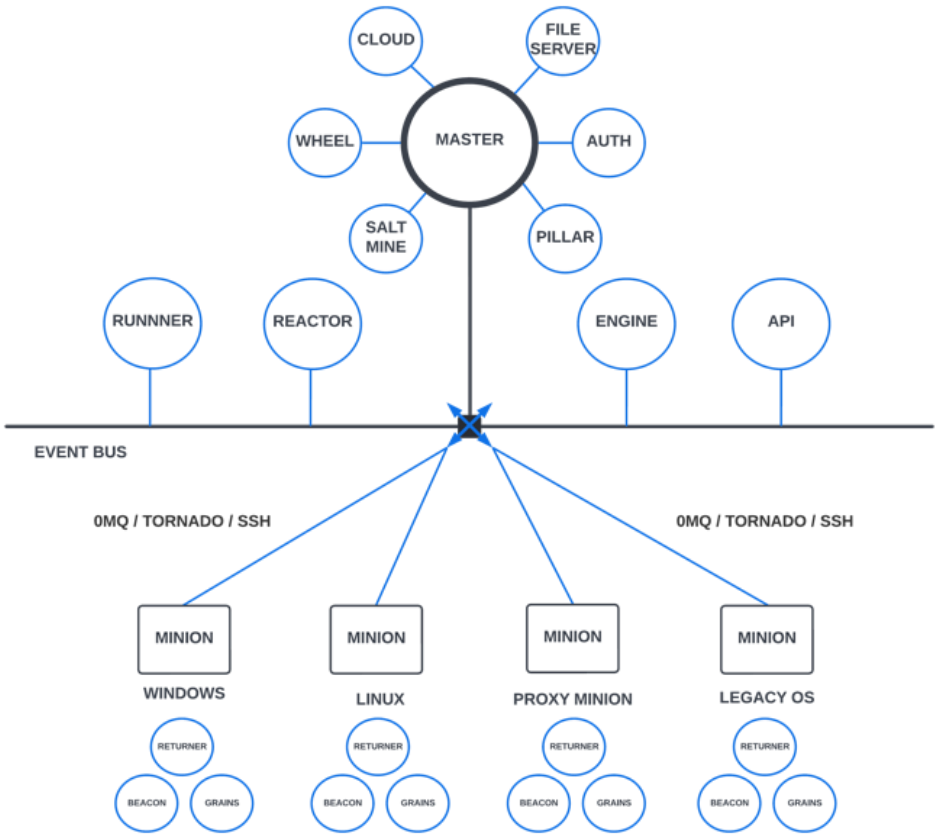
2.4.1 Historie

Autorem konfiguračního nástroje Salt Stack je Thomas Hatch, systémový inženýr a programátor. Hatch zveřejnil první ranou verzi 19. března 2011, další verze podporující Salt States zveřejnil 15. května 2011. Hatchovým cílem bylo vytvoření aplikace, jejíž předností bude rychlé vykonávání vzdálených příkazů. [62]

Salt Stack byl zveřejněn jako volně šiřitelný program (open source) pod licencí Apache 2.0.V srpnu roku 2020 došlo k akvizici s firmou VMware. [63]

2.4.2 Architektura

Architektura konfiguračního managementu Salt Stack je zobrazena na Obr. 6. Aplikace funguje v architektuře klient-server, server se nazývá master a klient je minion. Salt Stack může fungovat i zcela samostatně pomocí programů salt-ssh nebo salt-proxy. [64][65]



Obrázek 6 Architektura Salt Stack [zdroj: upraveno z 66]

Komponenty

Salt master

Salt master je server, který úkoluje minion klienty, minion po skončení úkolu zašle výsledné informace zpět serveru. Server používá pro komunikaci knihovnu ZeroMQ (Zero Message Queue), co je velmi výkonná knihovna pro asynchronní zasílání zpráv. [66]

Minions

Salt minion je klient Salt Stacku, který používá agenta salt-minion. Salt master zasílá příkazy agentu salt-minon jakou činnost má vykonat. [67]

States

States jsou konfigurační předpisy, které definují požadovaný stav (konfiguraci) koncového systému. Podle [68] „*Základem systému Salt State je soubor SLS neboli SaLt State. Soubor*

SLS je reprezentací stavu, ve kterém by se měl systém nacházet, a je nastaven tak, aby obsahoval tato data v jednoduchém formátu.[6]" (překlad autora z [68])

Grains (zrna)

Grains je programem, který sbírá informace o nastavení systému klienta (minions). Shromažďuje informace typu doménové jméno, IP adresa, informace o operačním systému, hardware, na kterém je Salt nainstalován. Jedná se hlavně o statické informace, které se mohou měnit, pokud dojde ke změně konfigurace systému nebo hardware. [69]

Pillars (pilíře)

Pilíře jsou typ data, definovaná na Salt Master serveru, která se používají v tzv. states (předpisy, které definují stav systému tzn. jeho konfiguraci). Data jsou předávána minions (klientům). [70]

2.4.3 Bezpečnost

Komunikace mezi serverem a klientem (minion), šifrovaní a dešifrování zpráv odeslaných z master server probíhá pomocí rotujícího klíče AES (Advanced Encryption Standard). Pro zvýšení bezpečnosti je to klient (minion), kdo inicializuje komunikaci s master serverem. [71]

2.5 Ansible

Ansible je v rodině konfiguračních nástrojů poměrně mladým nástrojem, který si během několika posledních let získal velkou popularitu mezi správci operačních systémů pro svou jednoduchost, rychlost, přehlednost, efektivnost a bezpečnost. Ansible používá pro popis jednotlivých úloh YAML a DSL. Konfigurační nástroj Ansible je volně dostupný ke stažení a provozování. [72][73]

2.5.1 Historie

Autorem projektu je Michael DeHaan, vývojář systémových nástrojů. Autor během svého působení u firmy Red Hat pracoval na projektu Cobbler. Začátek projektu se datuje na únor 2012 a nahradil tím automatizační nástroj Puppet. V ten samý rok byla zveřejněna první verze Ansible. [74][75]

---

[6] The core of the Salt State system is the SLS, or **Sa**l**L**t **S**tate file. The SLS is a representation of the state in which a system should be in, and is set up to contain this data in a simple format.

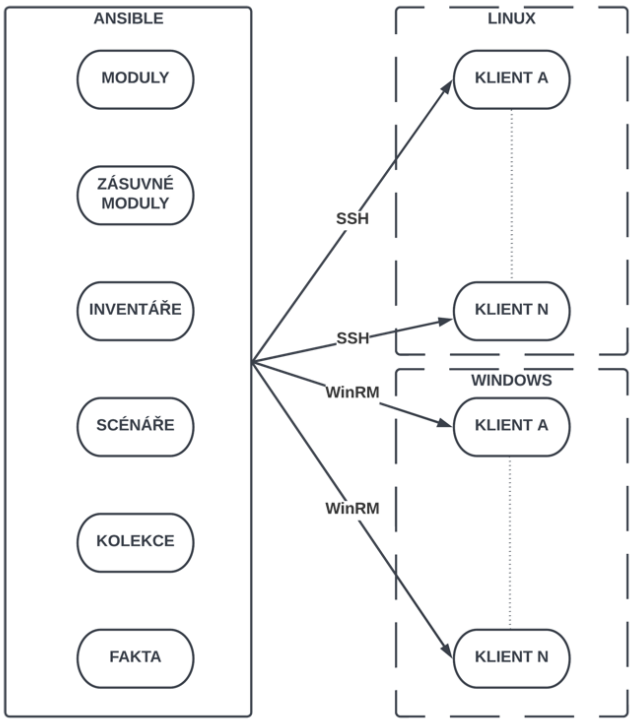
Autor uvádí v článku [74], že hlavními důvody vzniku Ansible byla frustrace z implementace nasazení nástroje Puppet u zákazníků. Dalším důvodem bylo zjednodušení nasazování automatizačních nástrojů. Dne 4. března 2013 oznámil založení firmy AnsibleWorks inc. Dne 16.října 2015 firma Red Hat oznámila v tiskové zprávě [77] akvizici firmy AnsibleWorks inc. [74][76]

2.5.2 Architektura

Ansible nevyžaduje komplikovanou architekturu jako je tomu v případě ostatních nástrojů konfiguračního managementu. Postačuje pouze lokálně instalovaný program Ansible a lokální účet na vzdáleném systému. Ansible komunikuje s vzdálenými systémy na bázi operačního systému Linux prostřednictvím protokolu OpenSSH (OpenBSD Secure Shell) a se systémy Microsoft Windows protokolu WinRM (Windows Remote management). Konfigurační management Ansible byl vytvořen v jazyce Python. [78]

Komponenty

Na Obr. 7 jsou zobrazeny základní komponenty a komunikace Ansible s klienty.



Obrázek 7 Architektura Ansible [zdroj: autor]

Inventories (inventáře)

Adresář obsahuje inventář systémů, které Ansible spravuje. Inventáře se vytváří v jazyce YAML nebo ve formátu INI souborů. [78][79]

Playbook (scénář)

Podle [80] je scénář automatizačních úloh, které provádějí úkoly bez zapojení lidského faktoru. Obsahuje předpisy tzv. plays, úlohy, které budou orchestrovat klienty serverové nebo síťové infrastruktury. [80]

Modules (moduly)

Podle [81] jsou moduly samostatné části kódu, které lze volat ze scénáře (playbook) nebo z příkazového řádku operačního systému.

Moduly podporují například:

- Operační systémy (Linux, Windows)
- Databáze MySQL, PostgreSQL
- Cloud (Amazon AWS, Microsoft Azure, Google cloud)
- Sítě (Cisco IOS, JunOS, F5)
- Úložiště dat (NetApp)

Plugins (zásuvné moduly)

Dokumentace projektu Ansible [78] uvádí, že: "*Zásuvné moduly nabízejí možnosti a rozšíření pro základní funkce Ansible – transformaci dat, protokolování výstupů, připojení k inventáři a další. Ansible se dodává s řadou praktických zásuvných modulů a můžete si snadno napsat vlastní*[7]". (překlad autora z [78])

Collections (kolekce)

Kolekce je určena pro distribuci obsahu Ansible, kterou mohou být scénáře, role, moduly a zásuvné moduly. [82]

[7] Plugins offer options and extensions for the core features of Ansible - transforming data, logging output, connecting to inventory, and more. Ansible ships with a number of handy plugins, and you can easily write your own.

Facts (fakta)

Fakta poskytuji informace o Ansible nebo vzdáleném systému. Tyto informace jsou přístupné pomocí proměnných a obsahují například data o operačním systému, IP adresy, připojených souborových systémech. Fakta se využívají ve scénářích nebo šablonách. [83]

Ansible Galaxy

Ansible Galaxy je webový portál pro sdílení rolí (role) nebo kolekcí (collection) v rámci komunity vývojářů Ansible. [84]

Ansible Automation Platform

Vlastník Ansible, firma Red Hat nabízí produkt Red Hat Ansible Automation Platform, který obsahuje integrovaný produkt Automation controller (dřive Ansible Tower) a podporu produktu. Automation controller nabízí webové rozhraní pro správu Ansible konfigurací a spouští naplánované scénáře. [85]

AWX

AWX je komunitní open source projekt sponzorovaný společností Red Hat, ze kterého vychází Ansible Tower. Tower umožňuje kontrolovat přístup, sdílet přihlašovací údaje SSH, grafickou správu inventářů. Podporuje integraci se serverem LDAP (Lightweight Directory Access Protocol), poskytuje přístup prostřednictvím REST API. [86]

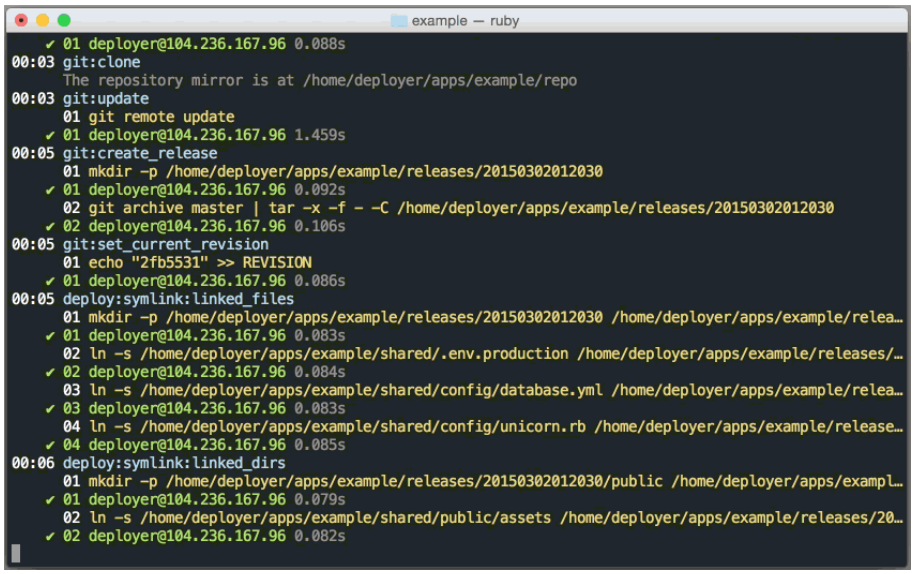
2.5.3 Bezpečnost

Komunikace mezi systémy probíhá pomocí bezpečného protokolu OpenSSH na operačních systémech typu Linux, pokud OpenSSH není dostupné použije Ansible modul jazyka Python – Paramiko. Pro komunikaci se systémy Windows používá protokol WinRM. [87][88]

2.6 Capistrano

Capistrano je aplikací, která je určena na nasazování webových aplikací na servery prostřednictvím bezpečné komunikace protokolem SSH. Tato aplikace se neřadí mezi nástroje konfiguračního managementu.

Používá se výhradně pro nasazování (deployment) aplikací na vzdálené servery. Nástroj Capistrano (Obr. 8) je naprogramován v jazyce Ruby, a i jeho syntaxe odpovídá struktuře tohoto jazyka. [89]



Obrázek 8 Capistrano [89]

2.7 Srovnání nástrojů konfiguračního managementu

Tabulka č.1 srovnává jednotlivé konfigurační nástroje podle jejich základních vlastností.

Tabulka 1 Srovnání nástrojů [zdroj:autor]

	CFEngine	Puppet	Chef	Ansible	SaltStack	Capistrano
DSL	ANO	ANO	ANO	ANO	NE	ANO
Deployment	ANO	ANO	ANO	ANO	ANO	ANO
Infrastruktura	ANO	ANO	ANO	NE	ANO	NE
Snadnost	NE	NE	NE	ANO	NE	ANO
Agent	ANO	ANO	ANO	NE	ANO	NE
Push	NE	NE	NE	ANO	ANO	NE
Pull	ANO	ANO	ANO	NE	ANO	NE
YAML[8]	NE	ANO[9]	ANO[10]	ANO	ANO	NE
Deklarativní jazyk	ANO	ANO	NE	ANO	ANO	ANO
Imperativní jazyk	NE	NE	ANO	NE	ANO	ANO
Jazyk	C	C++,Clojure, Ruby	Ruby	Python	Python	Ruby
Fakta	NE	Facter	Ohai	Facts	Grains	NE
Platforma pro sdílení	NE	Puppet Forge	Chef Supermarket	Ansible Galaxy	NE	NE

[8] Použití pro psaní konfiguračních scénářů
[9] Hiera [90] a Bolt [91]
[10] Od verze 16 [92]

3 APLIKACE

Kapitola se věnuje popisu životního cyklu aplikace, kterou je nutné zmínit pro bližší porozumění celkové problematice automatizovaného nasazování aplikací.

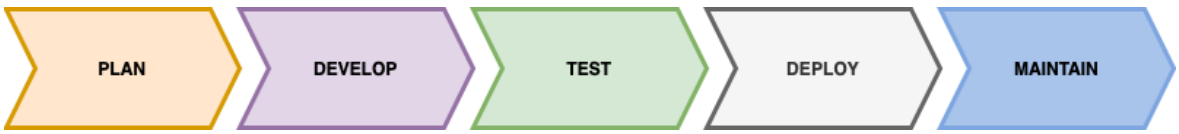
3.1 Životní cyklus aplikace

Každá aplikace má svůj vlastní životní cyklus – plánování, vývoj, testování, nasazování aplikace (deploy), její údržbou jako je instalace záplat (patch), opravy chyb (bugfix) a aktualizace (upgrade).

Zdroj [93] popisuje sedm částí vývojového cyklu (Obr. 9) následovně:

Obrázek 9 Sedm fázi životního cyklu vývoje software [zdroj: upraveno z 93]

Oproti tomu zdroj [94] (Obr. 10) uvádí vývojový cyklus aplikace v pěti fázích následovně:



Obrázek 10 Životní cyklus aplikace [zdroj: upraveno z 94]

V literatuře [95] autor uvádí fáze životního cyklu následovně:

- architektonický/vysokoúrovňový návrh (high-level design),
- detailní návrh (low-level design),
- vývoj (development),
- testování (testing),
- nasazování (deployment),
- údržba (maintenace).

Oproti tomu kniha [96] uvádí následující kroky:

- koncepce (conception),
- shromažďování požadavků, průzkum, modelování (requirements gathering/exploration/modeling),
- návrh (design),
- vývoj a ladění (coding and debugging),
- testování (testing),
- distribuce (release),
- údržba, vývoj programu (maintenance/software evolution),
- odstranění program (retirement).

Stephens v knize [95] rozděluje plánovací fázi na další dílčí kroky, který mi jsou high – a low-level design. High-level design, lze vysvětlil jako celkový pohled na produkt, přehled komponentů a základní komunikace mezi nimi. Low-level design se věnuje jednotlivým detailům komponentů a detailní komunikaci.

Oproti tomu webový portál [93] přidává mezi fázi plánování a vývoje software ještě definici požadavků a návrh a vývoj prototypů.

Jednotlivé body životní cyklu podle [94][95][96] lze popsat následovně:

Plánování (plan)

První fází životního cyklu aplikace je vždy plánování, které obsahuje úkoly, kterými jsou analýza požadavků na funkce aplikace, analýza jednotlivých komponentů aplikace a interakci mezi nimi až po návrh uživatelského rozhraní pro interakci s uživatelem a časový plán jednotlivých fází.

Vývoj (development)

Dalším nezbytným krokem je samotný vývoj aplikace, jednotlivých celků programu a jejich vzájemná interakce. Součástí vývoje aplikace by měl být i výše zmíněný návrh prototypů aplikace a návrhů.

Testování (test)

V další fázi se jednotlivé části programu testují, zda splňují požadavky pomocí testů. V dnešní době se využívají plně automatizované softwarové nástroje pro definování, spouštění a vyhodnocování jednotlivých testů.

Nasazování (deployment)

Po úspěšném průběhu testovací fáze dochází již k samotnému nasazování aplikací (deployment), který může být manuální nebo plně automatizovaný. Nasazování aplikací definuje, jak a kam se má aplikace nainstalovat, a jak má být nakonfigurována. K tomu to účelu je určen již zmiňovaný konfigurační management, který usnadní implementaci aplikací.

Automatizované nasazování aplikací zkracuje čas instalace a implementace a eliminuje chyby, které mohou nastat během manuálního procesu instalace.

Údržba (maintenance)

Předposlední fází životního cyklu aplikace je údržba. Údržba zahrnuje činnosti jako jsou instalace záplat (patch), odstranění chyb programu (bugfix) a instalace nových verzí (upgrade). Opět by se v případě údržby mělo postupovat, jak bylo popsáno ve fázi nasazování aplikací, tzn. nasazovat aplikace postupně po jednotlivých prostředích.

Odstranění program (retirement)

Poslední fází je odstranění programu. K tomuto kroku dochází z důvodů ukončení vývoje nových verzí nebo ukončení podpory aplikace pro určitou platformu nebo verzi operačního sytému. Dalším faktorem mohou být vysoké finanční náklady na licence aplikace, pokud se jedná a komerční produkt a její nahrazení například volně šiřitelným softwarem (open source) nebo nevyhovující vlastnosti programu.

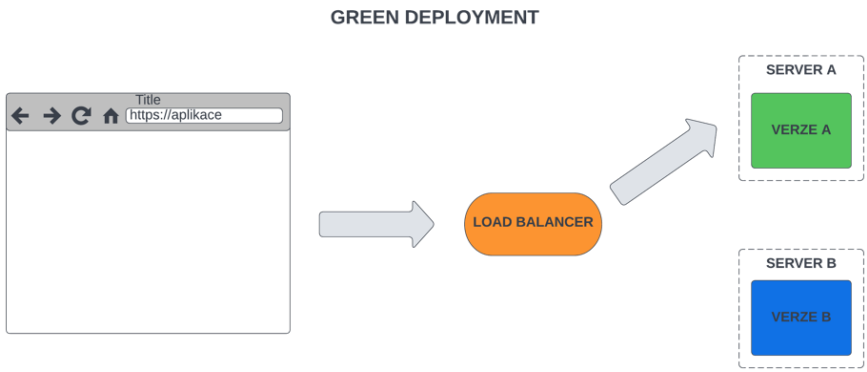
4 NASAZOVÁNÍ APLIKACÍ

V této kapitole jsou popsány metody nasazování aplikací. Metody se používají v rámci kontinuální integrace a kontinuálního nasazování aplikací – CI/CD (Continuous Integration/Continuous Delivery) neboli. Autoři knihy Continuous Delivery [97] uvádějí dva způsoby nasazování aplikací Prvním je *blue/green deployment* a druhým je *canary deployment*. Jedná se o dvě nejvíce používané metody v rámci kontinuálního nasazování aplikací.

Všechny následující metody počítají s nasazením load balanceru, tzn. síťového zařízení, které je schopné rozmístit zátěž provozu mezi více serverů. [98]

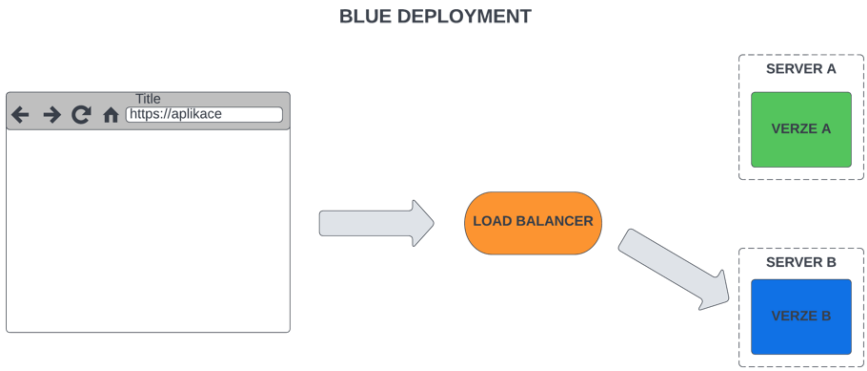
Blue/green deployment

Na Obr. 11 je zobrazen stav před změnou. Uživatel používá stále verzi aplikace označené jako verze A provozovanou na Serveru A.



Obrázek 11 Green deployment [zdroj: upraveno z 99]

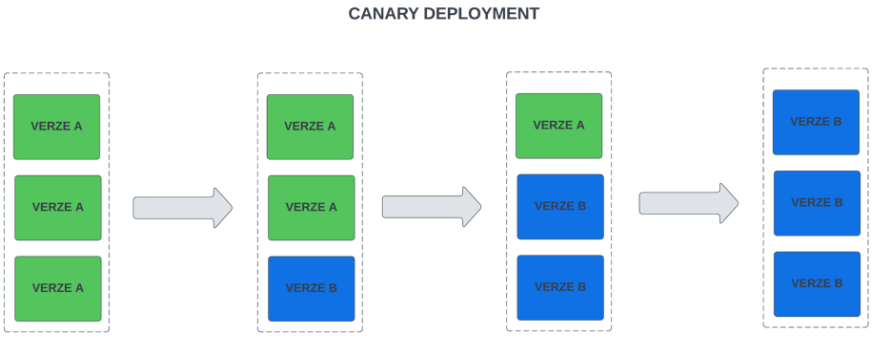
Na load balanceru dojde k přesměrování provozu ze serveru A na server B a aktualizovanou verzí B (Obr. 12). [97]



Obrázek 12 Blue deployment [zdroj: upraveno z 99]

Canary deployment

Během této strategie dojde k postupnému nahrazování aplikace verze A za verzi B po malých fázích (např. 25 %, 75 %, 100 %), jak je zobrazeno na Obr. 13. [97]



Obrázek 13 Canary deployment [zdroj: upraveno z 100]

Zdroje [99] a [100] uvádí kromě již zmíněných metod nasazování *blue/green* a *canary* další čtyři metody:

- recreate (znovuvytvoření),
- ramped (postupné nasazování),
- A/B testing (A/B testování),
- shadow (stínové nasazení).

Recreate (znovuvytvoření)

Během strategie recreate dojde k vytvoření nového serveru s aktualizovanou verzí aplikace. Stávající aplikace bude touto novou verzí nahrazena a zastavena. Tato strategie přináší množství nevýhod. Hlavní nevýhodou je nutná odstávka starého serveru se starou verzí a start nového s novou verzí. Během této doby bude aplikace nedostupná.

Ramped (postupné nasazování)

Ramped strategie znamená postupné nasazování nových verzí aplikace, které nahrazují stávající verze. Tato strategie se také nazývá inkrementální nebo průběžná (rolling-update). Nejprve se aplikace nasadí na první server, pokud je vše v pořádku přidá se další server a proběhne kontrola. Takto se postupuje dále až dojde k nasazení nové verze na všechny servery. Výhodou tohoto řešení je postupné nasazování nové verze aplikace, zatímco stávající verze je stálé v provozu. Nevýhodou je, že tento typ nasazování aplikace může být časově náročný.

A/B testing (A/B testování)

Během *A/B* strategie, obě verze aplikace A a B fungují současně. Pouze někteří uživatelé jsou postupně přesměrováni z verze A na novou verzi B. Mezi výhody této strategie patří současný běh několika verzí aplikace a kontrola nad provozem.

Shadow (stínové nasazení)

Strategie stínového nasazování aplikací znamená, že vedle současně provozované aplikace se nasadí nová verze. Stará verze postupně přesměruje provoz na novou verzi.

II. PRAKTICKÁ ČÁST

5 POŽADAVKY PRO AUTOMATIZACI NASAZOVÁNÍ APLIKACÍ

Kapitola se zabývá analýzou požadavků na nasazování aplikací z Knihovny UTB. Je zde popsána současná situace, jaké typy aplikací IT oddělení knihovny spravuje.

Knihovna provozuje zhruba 30 webových aplikací, jedná se o aplikace pro vyhledávání literatury, e-knih, e-časopisů, odborných publikací nebo dalších dokumentů.

Provozované aplikace využívají studenti a zaměstnanci a některé veřejně přístupné aplikace jsou k dispozici na síti Internet.

5.1 Analýza

Aplikace jsou nainstalovány nebo se instalují jako individuální softwarové balíky Linuxového operačního systému Debian pomocí standardních systémových nástrojů jako je program *apt*. Výhodou programu *apt* spočívá v řešení závislostí mezi jednotlivými balíčky programů, jako například knihovny nutné k instalaci.

V závislosti na typu aplikace probíhá distribuce software různými způsoby pomocí softwarových balíků určených pro operační systém Debian. Jedná se hlavně o webservery jako jsou například Apache a Tomcat, databázové servery MariaDB nebo MySQL a PostgreSQL, dále programovací jazyky typu Java ve formě JDK (Java Distribution Environment) nebo JRE (Java Runtime Environment), Perl, PHP a Solr.

Solr je open source platforma pro velmi rychlé vyhledávání postavená na knihovně Apache Lucene ™. Solr je často součástí instalačního archívu a není ho nutné dodatečně instalovat.

Další metodou používanou pro distribuci aplikací je instalace z veřejných úložišť software. Tyto úložiště podporují distribuovaný verzovací systém Git, který se používá pro ukládání volného software tzv. open source.

Aplikace jsou v závislosti na typu projektu k dispozici na webových stránkách projektů. Jsou distribuovány ve formě balíčků nebo ve zdrojové formě nebo jako softwarové archivy ve unixovém formátu *tar* a komprimovány pomocí aplikace *gzip* pro zmenšení objemu aplikace.

Poslední dvě metody distribuce aplikací vyžadují pokročilé znalosti operačního systému Linux na jejich instalaci. Jedná se o kompletně manuální instalační metodu, kde administrátor systému musí řešit jednotlivé závislosti software bez nebo s pomocí software *apt*.

Vzhledem k časové náročnosti manuální instalace aplikací, jejich množství a údržby přinese automatizace těchto kroků úsporu času, možnost reprodukovatelné instalace a v neposlední řadě kontrolu nad konfiguracemi jednotlivých aplikací a jejich verzování v systému Git.

Dalším požadavkem je snadné vytváření klonů jednotlivých prostředí. Tímto krokem je myšleno vytvoření instalace a konfigurace ve vývojovém prostředí a jednoduchá přenositelnost do testovacího prostředí a následně do produkce.

Pomocí nástroje na nasazování aplikací (deployment) by měl administrátor možnost snadno aktualizovat aplikaci na specifickou nebo nejnovější verzi.

5.2 Aplikace

Aplikace provozované knihovnou UTB jsou instalovány na samostatné virtuální servery provozovanými v prostředí Microsoft Hyper-V a pod operačním systémem Debian. Každá aplikace používá ve většině případů lokální instanci databázového serveru. Aplikace používají webové servery Apache nebo Tomcat.

5.2.1 Zabezpečení

Důležitým faktorem je samotné zabezpečení přístupu k jednotlivým aplikacím a bezpečná komunikace mezi uživatelovým klientem a serverem knihovny UTB. Bezpečná komunikace je docílena pomocí podpory protokolu HTTPS (Hyper Text Transfer Protocol) a SSL/TLS (Secure Socket Layer/Transport Security Layer) certifikátů na straně serveru. Tímto způsobem je zabezpečena šifrovaná komunikace a v případě odposlouchávání provozu nedojde k zachycení citlivých dat jakými jsou přihlašovací údaje (jméno a heslo) a jejich možné zneužití potenciálním útočníkem.

Řízení přístupu do aplikací je typicky řešeno pomocí Shibboleth SP (Service Provider), který zajišťuje autentifikaci a autorizaci ve spolupráci se vzdáleným Shibboleth IdP (Identify Provider) a lokální aplikací. Shibboleth je SSO (Single Sign On) systém, který podporuje protokoly SAML (Security Assertion Markup Language) pro komunikaci mezi Shibboleth SP a IdP a v tomto případě protokol LDAP (Lightweight Directory Access Protocol) pro komunikaci s lokálním zdrojem dat o uživatelích. [101]

5.2.2 Aplikace DSpace

Aplikaci DSpace používá knihovna pro provozování webového portálu Digitální knihovna UTB (https://digilib.k.utb.cz) a Publikační repozitář UTB (https://publikace.k.utb.cz/).

Domovská stránka DSpace [102] uvádí, že " *DSpace je software, který si vybírají akademické, neziskové i komerční organizace vytvářející otevřené digitální repozitáře. Je zdarma, lze jej snadno nainstalovat „out of the box" a zcela přizpůsobit potřebám jakékoli organizace.*[11] " (překlad autora z [102])

DSpace vyžaduje pro svojí instalaci:

- databázi (PostgreSQL),
- aplikační webový server (Tomcat),
- jazyk Java – JDK (Java Development Kit),
- Apache Maven,
- Ant,
- aplikaci na autorizaci Shibboleth.

Aplikace DSpace se distribuuje ve formě zdrojového archivu, který je nutné po rozbalení přeložit pomocí programu Apache Maven a dále nainstalovat programem Ant.

DSpace spolu s lokální databází PostgreSQL je instalován na samostatný virtuální server s operačním systémem Debian. Knihovna používá pro autorizaci uživatelů aplikace Shibboleth SP.

Bezpečná komunikace s okolním prostředím je zajištěna podporou protokolu HTTPS nakonfigurovanou ve virtuálním serveru s podporou SSL certifikátů.

5.2.3 Aplikace Gitea

Gitea je aplikace určená pro správu distribuovaného verzovacího software Git. Webové stránky aplikace uvádí [103], že tato aplikace je jednoduchá na instalaci, má nízké nároky na systémové zdroje, je multiplatformní a distribuuje se jako volně šiřitelný software.

Knihovna UTB využívá aplikaci Gitea jako správce repozitářů Git.

Gitea podporuje databáze typu:

- MySQL,

[11] DSpace is the software of choice for academic, non-profit, and commercial organizations building open digital repositories. It is free and easy to install "out of the box" and completely customizable to fit the needs of any organization.

- PostgreSQL,
- SQLite.

Gitea je také instalován na samostatný virtuální server s operačním systémem Debian. Aplikace používá pro uchovávání záznamu transakcí lokální databázový server MySQL.

Aplikace je dále zabezpečena pomocí podpory bezpečné komunikace HTTPS prostřednictvím SSL certifikátů.

5.2.4 Aplikace Koha

Webové stránky projektu Koha [104] uvádí, že: „*Koha je první svobodný softwarový balík pro automatizaci knihoven.*[12]“ (překlad autora z [104])

Koha obsahuje:

- plnohodnotný integrovaný knihovní systém – ILS (Integrated Library Systém),
- podpora jazykových mutací včetně českého jazyka,
- používá knihovní standardy pro kompatibilitu s ostatními knihovními systémy,
- webové rozhraní,
- software je šířen jako svobodný software pod licencí GPL (Free General Public License) verze 3,
- nezávislý na dodavateli. [104]

Aplikaci Koha používá knihovna pro provozování webového portálu Knihovního katalogu dostupného na adrese https://koha.k.utb.cz/.

Koha vyžaduje pro svojí instalaci:

- databázi (MySQL nebo MariaDB),
- webový server (Apache),
- jazyk Perl.

[12] Koha is the first free software library automation package.

Aplikace je dostupná ve formě Debian balíčku. Je provozována na samostatném virtuálním serveru a pro svůj provoz používá lokální databázi MySQL. Součástí je webový server Apache nakonfigurovaný s podporou protokolu HTTPS.

5.2.5 Aplikace OJS

OJS neboli Open Journal Systems je redakční systém pro správu a publikování vědeckých časopisů. Historie programu sahá do roku 2001, kdy byl vyvinut společností PKP pro zlepšení výzkumu. Během let se stal nejrozšířenějším volně šiřitelným softwarem pro vydávání časopisů. [105]

Knihovna UTB využívá tuto platformu jako redakční systém recenzovaných časopisů dostupných na adrese http://ojs.k.utb.cz/.

Aplikace vyžaduje pro svojí instalaci:

- databázi MySQL,
- jazyk PHP.

Aplikace OJS je dostupná ve formě archívu tgz.

OJS je další aplikací, která je provozována na samostatném virtuálním serveru s lokální databází MySQL. Bezpečná komunikace je stejně jako v předchozím případě zajištěna pomocí protokolu HTTPS s SSL certifikátů.

5.2.6 Aplikace WordPress

WordPress je volně šiřitelný program určený pro publikační činnosti jako jsou blogy nebo webové portály s možností přizpůsobení potřebám uživatele, firem nebo v případě knihovny UTB akademickým organizacím. [106]

Aplikaci WordPress používá knihovna pro provozování webových portálů:

- Portal Baťa a Baťův svět – https://tomasbata.org
- IVA Informační výchova
- Open Access portal
- Podcast UTB
- Akademická poradna UTB
- Nakladatelství UTB

- Bataproject

WordPress vyžaduje pro svojí instalaci:

- databázi (MySQL nebo MariaDB),
- webový server (Apache),
- jazyk PHP.

WordPress je distribuován jako archív programu ve formátu tgz.

Aplikace WordPress se spouští pod webovým serverem Apache, který umožňuje simultánní běh několika instancí současně. Knihovna provozuje WordPress jako jednotlivou instanci na samostatném serveru, ale také několik instancí na jednom serveru. V případě několika souběžných instancí, které se provozují pod webovým serverem Apache je každá instance nakonfigurována jako samostatný virtuální host.

Každá instance aplikace WordPress je nainstalována v separátním adresáři, a má k ní přístup pouze její administrátor pomocí protokolu SFTP (Secure File Transfer Protocol) a protokolu SSH. Jednotliví administrátoři mají přístup pouze do adresářové struktury své instance programu WordPress. Administrátoři mají tímto způsobem zajištěný přístup do své instance, kterou mohou nezávisle na ostatních instancích aplikace administrovat. Administrací je myšleno provádění údržby aplikace, jakou jsou nutné aktualizace, instalace a aktivace zásuvných modulů (plugins) nebo aktualizace webových stránek a podobné činnosti.

Do tohoto adresáře se také instaluje pomocný program wp-cli, který poskytuje administrátorovi možnosti aktualizací, instalací dalších podpůrných programů aplikace jako jsou zásuvné moduly atd.

Dalším bezpečnostním prvkem je použití aplikace Fail2ban a jejího zásuvného modulu (plugin) pro aplikaci WordPress, která blokuje neoprávněný přístup pomocí IP adresy na základě počtu neúspěšných pokusů o přístup.

5.2.7 Aplikace VuFind®

"*VuFind® je portál knihovních zdrojů navržený a vyvinutý pro knihovny knihovnami.*[13]" (překlad autora z [107]), který knihovna UTB využívá jako vyhledávací uživatelské rozhraní (frontend).

Aplikaci VuFind® používá knihovna pro provozování webového portálu VuFind® dostupného na adrese https://vufind.katalog.k.utb.cz/. Dále jako souborný katalog Zlínských knihoven – Knihovny Zlín na adrese http://knihovnyzlin.cz/.

VuFind® vyžaduje pro svojí instalaci:

- Databázi (MySQL nebo MariaDB),
- Webový server (Apache).

Aplikace VuFind® je dostupná jako Debian balík z oficiálního repozitáře distribuce. Jako ostatní aplikace je provozována na lokálním virtuálním serveru. Konfigurace virtuálního serveru v prostředí Apache podporuje protokol HTTPS a SSL certifikáty.

[13] VuFind® is a library resource portal designed and developed for libraries by libraries.

6 NÁVRH ŘEŠENÍ

Kapitola se zabývá výběrem vhodného řešení pro automatizované nasazování aplikací. Hodnotí jednotlivé nástroje, shrnuje jejich vlastnosti a posuzuje vhodnost pro nasazení v rámci knihovny UTB.

6.1 Výběr řešení

Výběr řešení byl omezen na aplikace Ansible, Capistrano, Chef a SaltStack, které poskytují vhodné nástroje pro řešení problematiky nasazování aplikací.

Aplikace Ansible, Capistrano nepotřebují ke své činnosti komplikovanou infrastrukturu. Chef a SaltStack pracují jak v režimu klient-server nebo samostatně.

Konfigurační nástroj Chef obsahuje nástroj Chef-solo, který funguje nezávisle na serverové části a lze ho použít pro nasazování aplikací. Chef-solo vyžadují pro vytváření konfiguračních předpisů znalost jazyka Ruby nebo DSL.

Aplikace Capistrano je primárně určena pouze na nasazování webových aplikací, tudíž jí nelze použít jako jiné nástroje ke konfiguracím operačních systémů.

Nástroj SaltStack obsahuje programy salt-ssh nebo salt-proxy. Vývoj konfiguračních předpisů tzv. salt states, vyžaduje počáteční velké úsilí k osvojení znalostí o tomto nástroji. Dokumentace projektu SaltStack není velmi přehledná.

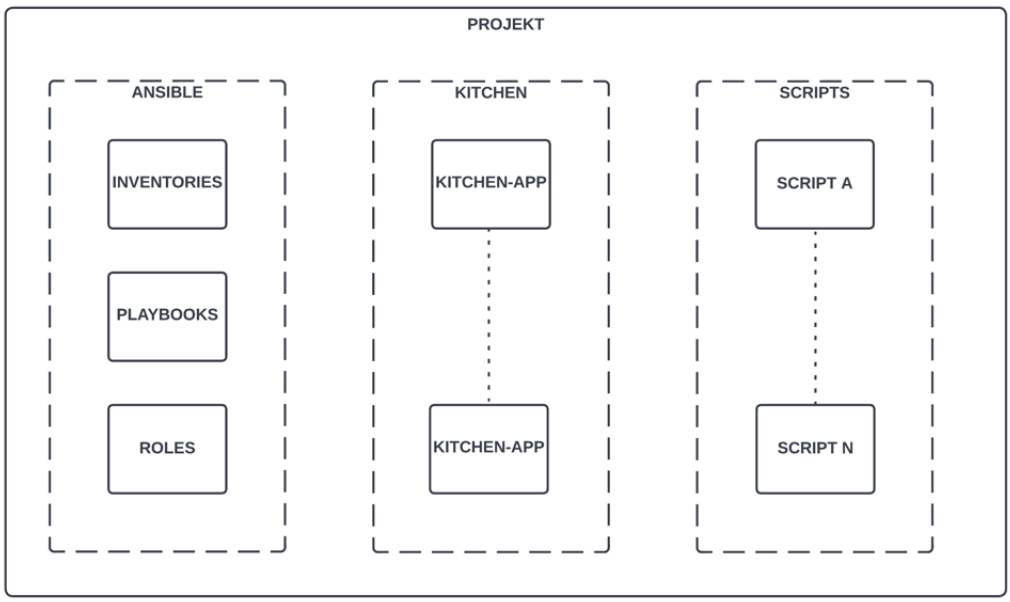
Poslední aplikací vhodnou pro nasazování aplikací je Ansible. Nepotřebuje ke svému provozu infrastrukturu, není nutná jakákoliv instalace agenta nebo samotného programu na systém, který se bude konfigurovat. Ansible používá jednoduchou syntaxi jazyka YAML pro definice konfiguračních předpisů. Konfigurační nástroje poskytují velké množství modulů, zhruba 3 400 v roce 2022. Moduly podporují konfiguraci operačních systémů (Linux, Windows), aplikací, databází, zařízení pro ukládání dat, poskytovatelé cloudových služeb, virtualizačních nástrojů a síťových prvků. Použití Ansible je velmi snadné, na lokálním systému stačí nainstalovat instalační balíček a nakonfigurovat SSH přístup na vzdálený systém nebo je možné spustit konfiguraci na lokálním systému.

Z těchto důvodů byl vybrán Ansible jako nejvhodnější nástroj pro řešení automatizovaného nasazování aplikací. Nástroj poskytuje kromě nasazování aplikací také možnosti řešení konfigurace pro nasazení při konfiguraci IT infrastruktury. Konfigurační nástroj Ansible je

vhodný pro konfiguraci serverových operačních systémů nebo konfiguraci síťových zařízení. [108][109]

6.2 Návrh projektu

Projekt bude obsahovat adresářovou strukturu jako je zobrazena na Obr. 14. Projekt by měl obsahovat adresář ansible, ve kterém se budou nacházet inventáře (inventories), scénáře aplikací (playbooks) a role (roles). Dále bude obsahovat adresář kitchen, zde budou umístěny konfigurační soubory aplikace Kitchen určené pro lokální vývojové prostředí. Projekt bude dále obsahovat adresář scripts, kde se budou nacházet pomocné skripty pro instalaci vývojového prostředí.



Obrázek 14 Návrh projektu [zdroj: autor]

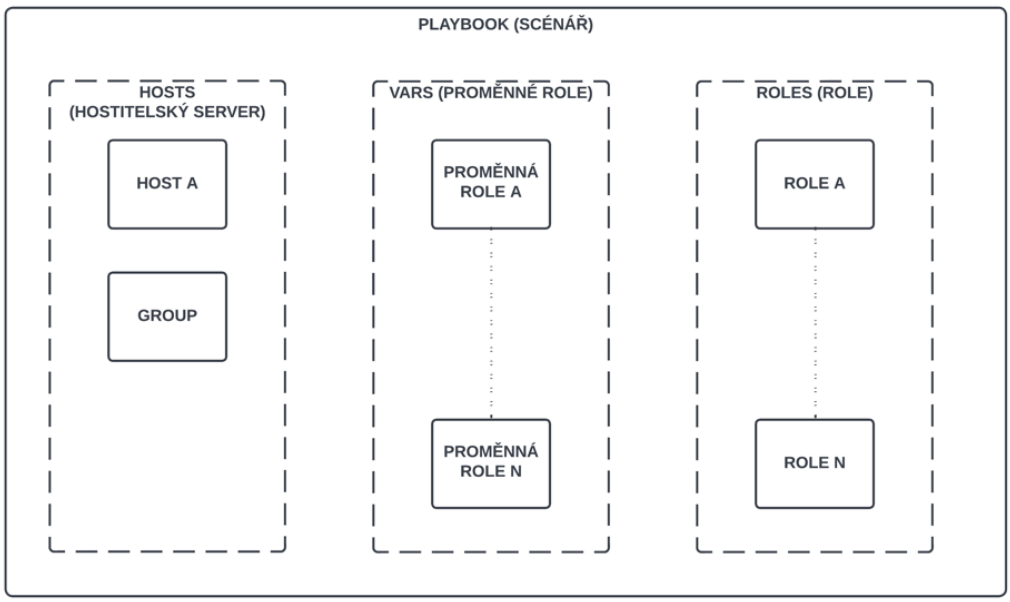
6.3 Návrh Ansible scénáře (playbook)

Vzhledem k počtu aplikací, které provozuje knihovna UTB by mělo být výsledné řešení co nejvíce modulární. Toto řešení umožňuje kombinovat již existující podpůrné role pro jednotlivé knihovní aplikace.

Ansible poskytuje tuto flexibilitu pomocí scénářů, které mohou existovat samostatně. Další možností je organizovat scénáře do rolí, které sdružují komplexnější činnosti.

Každá knihovní aplikace by měla mít svojí vlastní roli stejně jako podpůrné programy.

Na Obr. 16 je znázorněna architektura scénářů. Scénář obsahuje jméno cílového systému (hosts), proměnné role (vars), a jména rolí (roles).



Obrázek 15 Architektura scénáře [zdroj: autor]

Scénář zobrazený na Obr. 15 přepsaný do jazyka YAML by vypadal následovně:

```
1 ---
2 - host: host_A
3   roles:
4     - role_A
5     - role_N
6   vars:
7     roleA_var_1: foo
8     roleA_var_2: bar
```

Scénář je rozdělen na několik základních částí:

- hosts,
- roles,
- vars.

Část *hosts* (řádek č.2) obsahuje klienta nebo klienty, které bude Ansible konfigurovat.

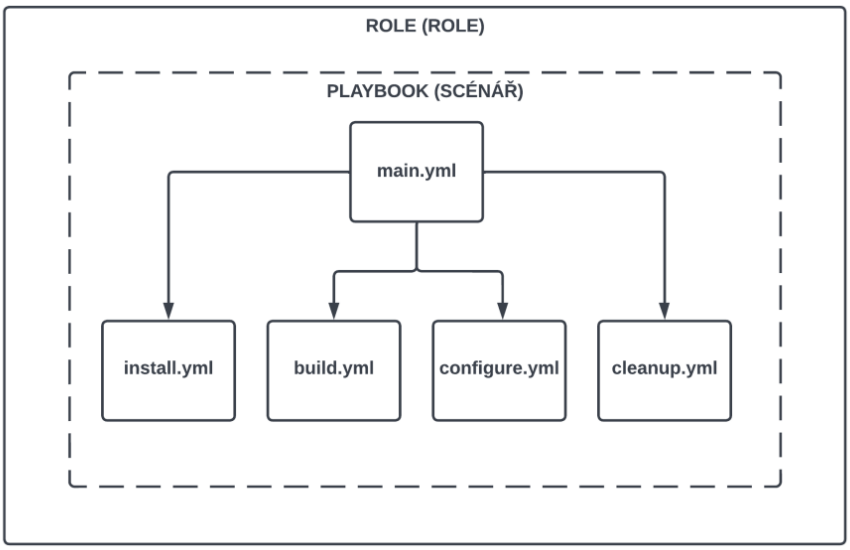
Část *roles* (řádek č.3–5) obsahuje seznam rolí.

Část *vars* (řádek č.6–8) obsahuje proměnné jednotlivých rolí, které jsou odlišné od výchozích hodnot nastavených v adresáři *roles/<jméno_role>/defaults*.

6.4 Návrh Ansible role (role)

Role poskytuje rozdělení scénářů do jednotlivých souborů, například aplikace se nejprve nainstaluje, pak následuje konfigurace, dále aplikace generuje soubory na základě šablon a na závěr zajistí, že bude aplikace automaticky spuštěna.

Na Obr. 16 je zobrazena role se vzorovým scénářem. Role má hlavní scénář (main.yml), tento scénář je povinný a zahrnuje ostatní scénáře pro danou roli, které stáhnou software potřebný pro instalaci (install.yml), překlad zdrojových souborů (build.yml), konfigurace (configure.yml) a odstranění dočasných souborů (cleanup.yml).



Obrázek 16 Architektura role [zdroj: autor]

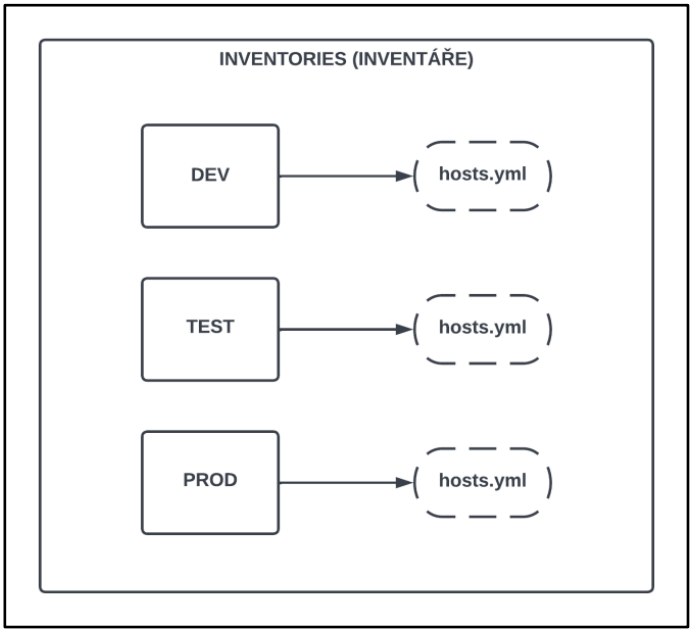
V případě potřeby by role měla být schopná vygenerovat konfigurační soubory, které jsou buď ve formě souborů nebo šablon. Na základě výchozích proměnných nebo uživatelsky definovaných proměnných by role měla mít schopnost vygenerovat soubory ze šablon. Role by měly být schopné komunikovat s démony služeb a při instalaci nastavit automatický start, při změně konfigurace restartovat službu.

6.5 Návrh Ansible inventáře (inventories)

Klienti Ansible by měly být logicky odděleny podle jednotlivých prostředí, ve kterých jsou umístěny. Nejvhodnějším řešením je oddělení prostředí prostřednictvím adresářové struktury, které by vypadala, jak je zobrazeno na Obr. 17:

- adresář DEV pro vývojové prostředí,

- adresář TEST pro testovací prostředí,
- adresář PROD pro produkční prostředí.



Obrázek 17 Architektura inventáře [zdroj: autor]

Každé prostředí by mělo obsahovat soubor nebo soubory s jmény a IP adresami klientů, které Ansible konfiguruje.

6.6 Návrh Ansible proměnných (vars)

Proměnné se vyskytují na všech úrovních (scénáře, role) a umožňují parametrizaci vykonávaných úloh.

Jednotlivé role by měly umožnovat změnu proměnných, které se mohou měnit v závislosti na prostředí. Například ve vývojovém prostředí bude nainstalována novější verze aplikace určená pro testování, než je verze v produkčním prostředí.

7 KOMPONENTY TECHNICKÉHO ŘEŠENÍ

Kapitola obsahuje popis jednotlivých komponent technického řešení automatizovaného nasazování aplikací.

7.1 Verzovací software

Celý projekt bude uložený ve verzovacím software Git v centrálním repozitáři knihovny UTB pod jménem playbooks.

7.2 Ansible program

Instalace Ansible je velmi jednoduchá. V závislosti na provozovaném operačním systému se nainstaluje pouze balíček ansible jak je uvedeno na následujícím příkladu na systému Debian.

```
sudo apt install ansible
```

Příkaz *apt* řeší závislosti balíčku např. nainstaluje chybějící moduly jazyka Python.

7.3 Ansible scénář (playbook)

Podle [80] je scénář souhrn automatizačních úloh, které provádějí úkoly bez zapojení lidského faktoru. Obsahuje předpisy, jaký software má být nainstalován a jak má být koncový systém nakonfigurován. Scénář se využije pro organizaci jednotlivých komponentů aplikace včetně aplikace samotné.

Ukázková struktura scénáře je následující:

```
1 ---
2  - hosts: my_host
3    become: yes
4    remote_user: ansible
5    gather_facts: yes
6
7    roles:
8     - apache
9
10   vars:
11      dspace_database_name: dspace
12      dspace_database_user: dspace
```

Na řádku č.2 parametr *hosts* označuje počítač nebo definovanou skupinu počítačů, na kterých se scénář spustí. Parametr *become* uvedený na řádku č.3 definuje, zda chceme použít privilegovaná práva. Na řádku č.4 parametr *remote_user* definuje jako jaký uživatel se bude Ansible na vzdálený systém přihlašovat. Uživatel musí existovat, jinak dojde k chybě provádění programu. Pokud vynecháme parametr *remote_user*, Ansible se bude přihlašovat jako aktuální uživatel, který spouští tento scénář. Parametr *gather_facts* (řádek č.5) definuje, zdali se budou shromažďovat informace o vzdáleném systému např. jméno serveru (hostname), IP adresa, verze operačního systému atd.

Na řádku č.7 se nalézá definice rolí *role*, která definuje, jaké role má scénář použít pro konfiguraci vzdáleného systému.

Na řádku č.10 definice *vars* obsahuje vlastní definice proměnných, které chceme změnit pro danou roli a jsou odlišné od proměnných deklarovaných v souborech *role/vars/main.yml* nebo *role/defaults/main.yml*.

7.4 Ansible role (role)

Ansible obsahuje program pro vygenerování šablony role, tímto programem je ansible-galaxy. Následující příkaz vygeneruje roli:

```
ansible-galaxy init <JMÉNO ROLE>
```

Struktura vygenerované role je následující:

```
role
├── README.md
├── defaults
├── files
├── handlers
├── meta
├── tasks
├── templates
├── tests
└── vars
```

7.4.1 Adresář defaults

Adresář defaults obsahuje výchozí proměnné role. Proměnné, které jsou zde definované mají nejmenší prioritu.

7.4.2 Adresář files

Adresář files obsahuje soubory, které chceme zkopírovat na cílový systém.

7.4.3 Adresář handlers

Adresář *handlers* obsahuje obslužné programy, které mají na starosti práci s jednotlivými démony programů. Jedná se o příkazy k nastartování démona, restartu, zastavení démona prostřednictvím moderního Linuxového systémového manažeru *systemd* a Ansible příkazu *service*, označení *ansible.builtin* znamená, že se jedná o interní modul Ansible. Existují i moduly vytvořené komunitou Ansible uživatelů a vývojářů a ty jsou označovány jako *community*. Na následujícím příkladu je ukázka předpisu pro službu databázového serveru mysql:

```
1 - name: start mysql
2   ansible.builtin.service:
3     name: mysql
4     state: started
```

Na řádku č.1 je pojmenování úlohy pomocí parametru *name*. Na řádku č.2 je volání interního modulu *service*, který řídí činnost služby – démona. Modul poskytuje několik možností pro konfiguraci. Na řádku č.3 je uvedeno jméno služby, kterou modul nastavuje. Na řádku č.4 je uveden stav této služby, kdy chceme, aby služba byla spuštěna (started). Stavy služby mohou být dále, kromě již uvedené, restartován– *restarted*, zastaven – *stopped*, nebo znovu načten – *reloaded*.

7.4.4 Adresář meta

Adresář meta obsahuje metadata role jako jsou autor, popis, licence, podporovaná verze Ansible, závislosti. Informace jsou důležité, pokud bude role publikována na portál Ansible Galaxy (https://galaxy.ansible.com).

7.4.5 Adresář tasks

Adresář tasks obsahuje seznam úkolů (tasks), které má program Ansible vykonat. Předpis úkolu je následující:

```
- name: Jméno úkolu
  <příkaz>:
    <parametry>:
```

Například pro nainstalování balíčku *vim* na systému Debian, by vypadal následovně:

```
1 - name: Install package vim
2   ansible.builtin.apt:
3     name: vim
4     state: present
```

Řádek č.1 označuje jméno úlohy, na řádku č.2 je volání modulu *apt*, Řádek č.3 obsahuje jméno balíčku, který má být nainstalován a na posledním řádku s parametrem *state* říkáme, že chceme, aby byl program nainstalován.

7.4.6 Adresář templates

Adresář templates obsahuje šablony souborů, kde chceme dynamicky měnit parametry nastavení na základě předem definovaných proměnných nebo proměnných prostředí. Šablony se vytvářejí pomocí jazyka Jinja, který primárně využívá programovací jazyk Python.

Na následujícím příkladu je demonstrován příklad využití šablony pro nastavení souboru:

```
1 {% if ansible_hostname == „dev01.lab.local“ %}
2     parameter_A = {{ var_A }}
3 {% else %}
4     parameter_A = {{ var_B}}
5 {% endif %}
```

Pokud je jméno serveru uvedené ve faktu *ansible_hostname* rovno jménu *dev01.lab.local* (řádek č.1) přidá se do souboru proměnná *parametr_A* a její hodnota bude rovna hodnotě definované v uživatelské nebo výchozí proměnné *var_A* (řádek č.1), jinak bude *parameter_A* roven hodnotě *var_B*.

Uvedený příklad předpokládá, že jsou definovány proměnné *var_A* a *var_B*.

7.4.7 Adresář test

Adresář test obsahuje soubor integračních testů pro danou roli.

7.4.8 Adresář vars

Adresář vars obsahuje proměnné pro roli. Hodnoty v adresáři vars mají vyšší prioritu nad hodnotami v adresáři defaults.

Proměnné v adresáři *vars/main.yml* pro nastavení uživatele by vypadaly následovně:

Konfigurace skupiny, uživatele, výchozího shellu a domovského adresáře:

```
Application_group: application
application_user: application
application_user_shell: /bin/bash
application_home: /path/to/application
```

Adresář pro ukládání dočasných souborů:

```
application_download_dir: /tmp/aplication
```

Verze instalované aplikace:

```
application_version: 1.0
```

Konfigurace GPG klíče a Debian repozitáře s instalačními balíky:

```
application_repository_apt_key: „https://path_to_gpg_key_location“
application_repository_url: „deb  http://repository stable main“
```

7.5 Ansible inventáře (inventories)

Adresář obsahuje inventář systémů, které Ansible spravuje. Je na administrátorovi systému, jak navrhne strukturu adresářů. Je vhodné logicky oddělit systémy podle jednotlivých systémů:

- *dev* – obsahuje systémy určené pro vývoj,
- *test* – obsahuje systémy určené pro testování,
- *prod* – obsahuje výhradně produkční systémy.

Struktura adresáře inventories je následující:

```
inventories
├── dev
│   └── hosts.yml
├── prod
│   └── hosts.yml
└── test
    └── hosts.yml
```

Soubor *inventories/dev/hosts.yml* pak obsahuje definici hostů [79]:

```
---
all:
```

```
  application_A
  application_B

application_A:
  hosts:
    server_A.example.net

application_B:
  hosts:
    server_B.example.net
```

7.6 Spuštění Ansible

Jednotlivé scénáře se spouštějí pomocí příkazu ansible-playbook, jak je uvedeno na následujícím příkladu:

```
ansible-playbook -i inventories/dev/hosts.yml minimal.yml -K
```

Program *ansible-playbook* s parametrem *-i <inventory>*, použije dříve jmenovaný inventarizační soubor, soubor *minimal.yml* je jméno playbooku popsaného v kapitole 5.3 a parametr *-K* je výzva k zadání hesla superuživatele root.

Výsledek běhu programu je následující:

```
BECOME password:

PLAY [application_A]

*********************************************************************

TASK [Gathering Facts]

*********************************************************************

ok: [server_A.example.net]

TASK [vim : install package]

*********************************************************************

ok: [server_A.example.net] => (item=vim)

PLAY RECAP
*********************************************************************

server_A.example.net              : ok=2    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

Na posledním řádku z příkladu se nachází výsledek běhu programu *ansible-playbook*, *ok=2*, znamená, že dva úkoly (task) proběhly úspěšně. V případě neúspěchu jedné z úloh by se

změnila hodnota parametru *failed*, pokud by došlo ke změně například konfigurace změnila by se hodnota parametru *changed*. Parametr *unreachable* udává počet nedostupných hostů v případě, když se scénářem konfiguruje vice klientů. Parametr *skipped* udává počet vynechaných úkolů. Parametr *rescued* udává počet volání rescue v blocích úkolů a poslední parametr *ignored* udává počet ignorovaných chyb v úkolech.

8 REALIZACE ŘEŠENÍ

Na základě požadavků uvedených v kapitole 5 a dle návrhu v kapitole 6 a kapitole 7 byly vytvořeny scénáře pro instalaci a konfiguraci jednotlivých aplikačních prostředí. Pro několik ukázkových aplikací byly vytvořeny role, které instalují a konfigurují jednotlivé aplikace do aplikačních prostředí.

Struktura projektu playbooks v repozitáři Git má následující strukturu:

```
playbooks
├── README.md
├── ansible
├── kitchen
└── scripts
```

README.md je soubor, který obsahuje dokumentaci ve formátu jazyka Markdown.

ansible je adresář, který obsahuje jednotlivé Ansible role a scénáře určené pro instalaci jednotlivých aplikací.

kitchen je adresář obsahující přepisy pro vytvoření lokálních testovacích prostředí pro jednotlivé aplikace.

scripts je adresář, který obsahuje skript pro instalaci lokálního vývojového prostředí.

8.1 Struktura adresáře kitchen

Adresář kitchen obsahuje adresáře, které obsahují konfigurační soubory pro lokální vývojové prostředí na vytváření Ansible rolí.

```
kitchen
├── kitchen-dspace
├── kitchen-gitea
├── kitchen-koha
├── kitchen-ojs
├── kitchen-vufind
├── kitchen-wordpress
└── kitchen-sample.yml
```

kitchen-dspace je adresář, který obsahuje konfiguraci kitchen pro aplikaci DSpace.

kichen-gitea je adresář, který obsahuje konfiguraci kitchen pro aplikaci Gitea.

kitchen-koha je adresář, který obsahuje konfiguraci kitchen pro aplikaci Koha.

kitchen-ojs je adresář, který obsahuje konfiguraci kitchen pro aplikaci OJS.

kitchen-vufind je adresář, který obsahuje konfiguraci kitchen pro aplikaci VuFind ®.

kitchen-wordpress je adresář, který obsahuje konfiguraci kitchen pro aplikaci WordPress.

kitchen-sample.yml je soubor s ukázkovou konfigurací pro Chef Kitchen.

8.2 Struktura adresáře ansible

Adresář ansible obsahuje následující soubory*:*

```
ansible
├── README.md
├── inventories
├── playbooks
└── roles
```

README.md je soubor, který obsahuje dokumentaci ve formátu jazyka Markdown.

inventories je adresář, který obsahuje seznam Ansible klientů.

playbooks je adresář, který obsahuje jednotlivé scénáře aplikací.

roles je adresář obsahující role aplikací a podpůrných programů.

8.2.1 Struktura adresáře inventories

Adresář inventories obsahuje inventář systémů, které Ansible spravuje. Je na administrátorovi systému, jak navrhne strukturu adresářů. Je vhodné logicky oddělit systémy podle jednotlivých systémů:

- *dev* – obsahuje systémy určené pro vývoj,
- *test* – obsahuje systémy určené pro testování,
- *prod* – obsahuje výhradně produkční systémy.

Struktura adresáře *inventories* je následující:

```
inventories
├── dev
│   └── hosts.yml
├── prod
│   └── hosts.yml
└── test
    └── hosts.yml
```

Soubor *inventories/dev/hosts.yml* s hosty obsahuje [79]:

```
---
wordpress:
  hosts:
    192.168.1.180

dspace:
  hosts:
    192.168.1.182
```

8.2.2 Struktura adresáře roles

Adresář roles obsahuje jednotlivé role, které byly vytvořeny pro knihovní aplikace, nebo podpůrné programy. Některé role, jak je dále uvedeno v textu byly převzaty z portálu Ansible Galaxy.

Databázové role

Pro instalace databázových serverů byly zvoleny již existující Ansible role, které jsou volně šiřitelné a dostupné z portálu Ansible Galaxy.

Role geerlingguy.mysql

Pro databázovou aplikaci, byla použita již existující role geerlingguy.mysql od autora knihy Ansible for DevOps [73], Jeffa Geerlinga, dostupná na portálu Ansible Galaxy.

Role poskytuje konfigurační možnosti pro konfiguraci MySQL nebo MariaDB databáze, pro aplikaci WordPress byla použita pro konfigurace hesla uživatele root, vytvoření databáze uživatele a vytvoření uživatele.

Role geerlingguy.postgresql

Stejně jako v případě role pro MySQL, byla pro konfiguraci databázového serveru použita existující role od Jeffa Geerlinga, geerlingguy.postgresql. Role poskytuje konfigurační možnosti pomocí deklarování proměnných.

Webové servery

V případě řešení webových serverů byla vytvořena role pro instalaci a konfiguraci webového serveru Apache, která používá specifickou konfiguraci virtuálních serverů a podpory SSL. Dále byla převzata role geerlingguy.apache z portálu Ansible Galaxy pro konfiguraci některých scénářů. Jako poslední role byla vytvořena role pro webový server Tomcat.

Role Apache

Role apache nainstaluje webový server Apache z repozitáře operačního systému Debian. A vygeneruje konfiguraci virtuálního hostu, jedná se o specifickou konfiguraci knihovny UTB. Role také umožňuje nakonfigurovat, které Apache moduly mají být povoleny při běhu služby. Dále po provedení změn v konfiguraci, role ovládá službu Apache a restartuje ji.

Role geerlingguy.apache

Pro konfiguraci webového serveru použita existující role geerlingguy.apache z portálu Ansible Galaxy. Role poskytuje konfigurační možnosti pomocí deklarování proměnných. Role byla použita pro konfiguraci některých scénářů.

Role Tomcat

Role tomcat instaluje webový aplikační server Tomcat. Automaticky nastaví spuštění služby Tomcat.

Programovací jazyky

Jednotlivé role pro instalaci programovacích jazyků byly vytvořeny, aby odpovídaly specifickým požadavkům některých aplikací.

Role jdk

Role jdk nainstaluje Debian balík openjdk-jdk. Instalaci lze pomocí proměnné parametrizovat, jaká verze má být nainstalována.

Role jre

Role jre nainstaluje Debian balík *openjdk-jre*. Instalaci lze pomocí proměnné parametrizovat, jaká verze má být nainstalována.

Role php

Role nainstaluje Debian balík php a příslušné podpůrné moduly.

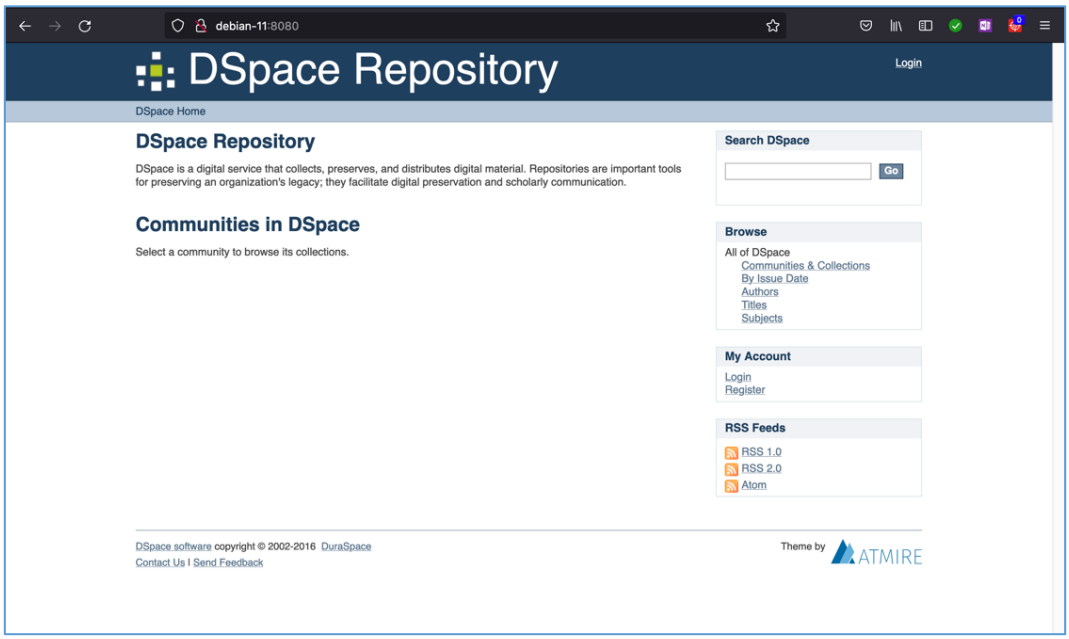
Knihovní aplikace

Pro jednotlivé aplikace byly vytvořeny samostatné role, které je nainstalují, nakonfigurují a spustí.

Role dspace

Role stáhne archív aplikace, nainstaluje a nakonfiguruje aplikaci DSpace (Obr. 18). Následně přidá uživatele a skupinu dspace, nutné pro běh aplikace. Dále vytvoří adresářovou

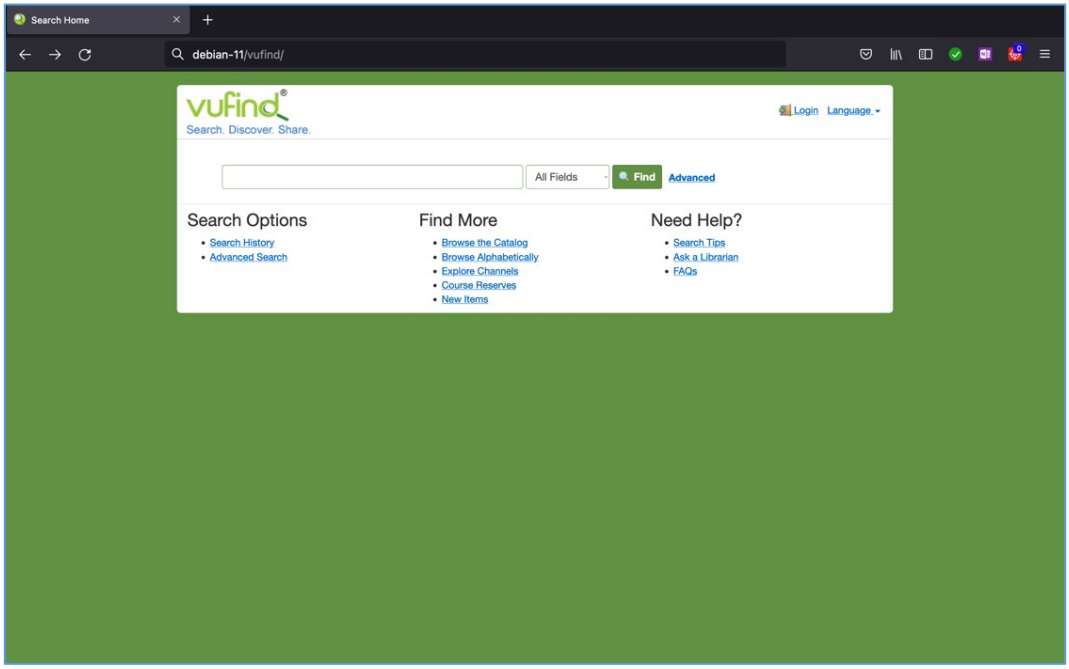
strukturu, ze zdrojových programů přeloží aplikaci pomocí programu Maven, nainstaluje jí programem Ant a zkopíruje aplikaci a její podpůrné programy do adresářové struktury programu Tomcat. Role následně stáhne a nainstaluje lokalizační programy do aplikace DSpace. Po provedení všech kroků spustí webový server Tomcat.



Obrázek 18 Aplikace DSpace [zdroj: autor]

Role vufind

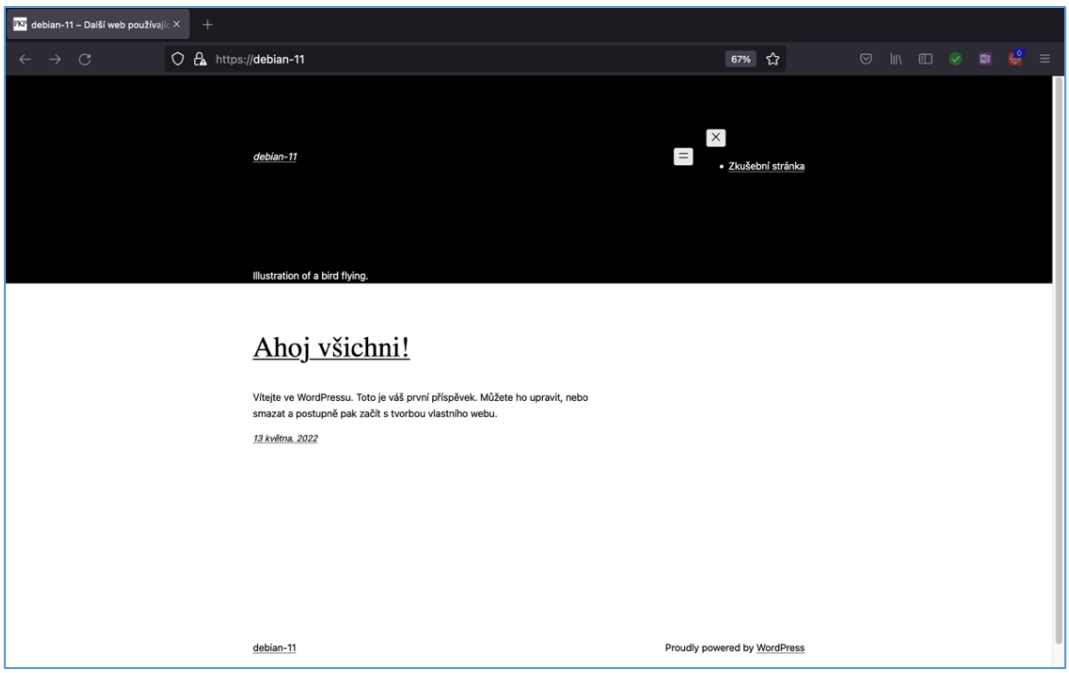
Role stáhne archív aplikace, nainstaluje a nakonfiguruje aplikaci VuFind® (Obr. 19), přidá uživatele a skupinu vufind, nutnou pro instalaci a konfiguraci aplikace. Role nainstaluje server Apache s podporou protokolu HTTPS.



Obrázek 19 Aplikace VuFind® [zdroj: autor]

Role wordpress

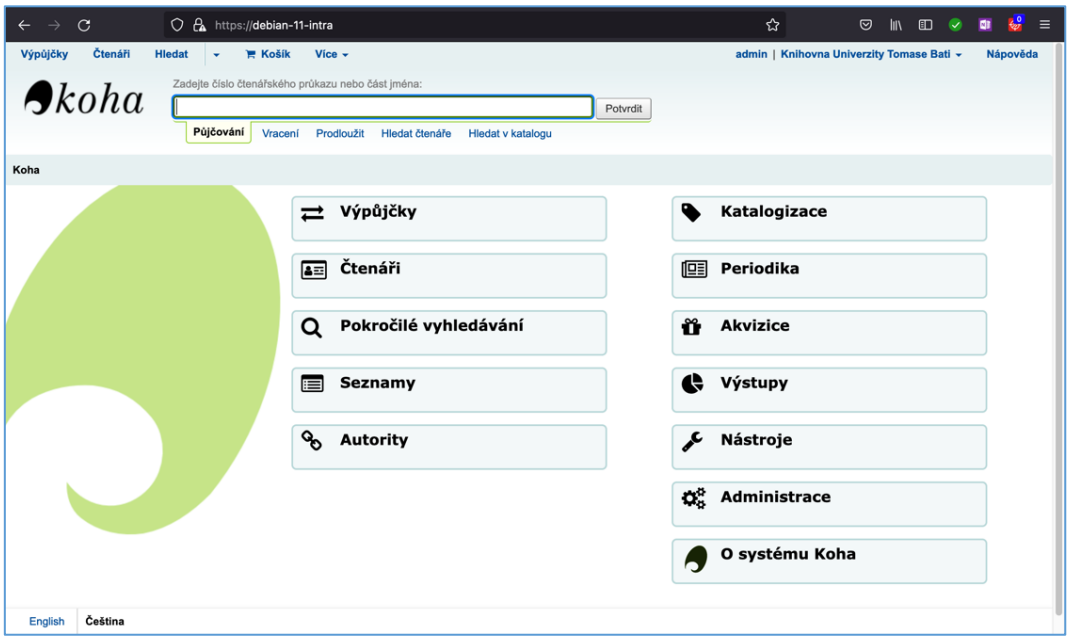
Role wordpress stáhne českou verzi archívu aplikace, nainstaluje a nakonfiguruje aplikaci WordPress (Obr. 20) podle předem stanovených požadavků popsaných v kapitole č.5.



Obrázek 20 Aplikace WordPress [zdroj: autor]

Role koha

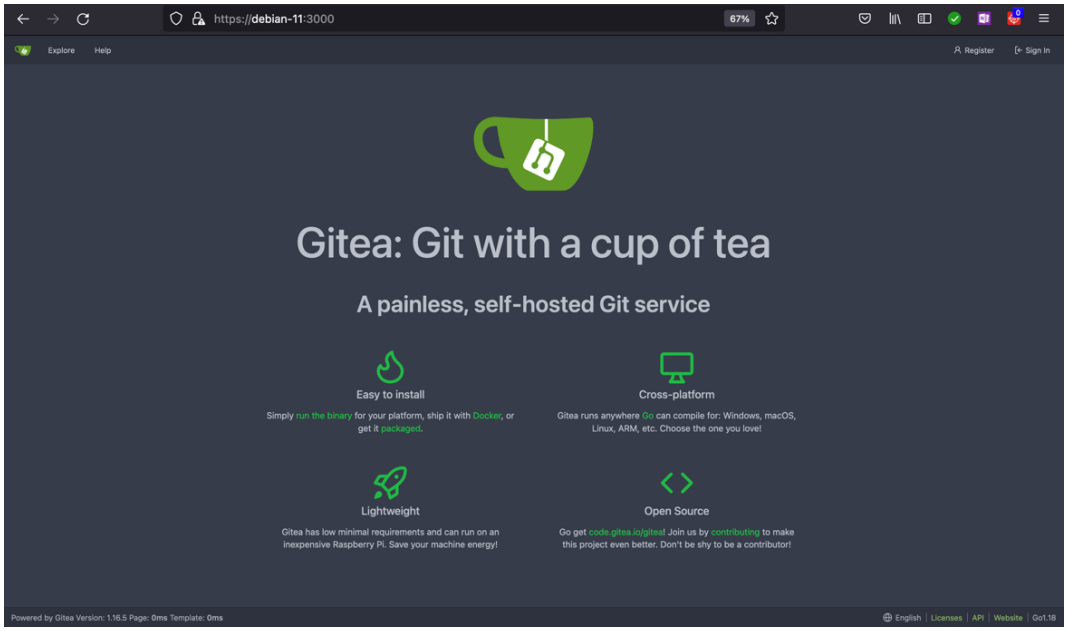
Role koha přidá repozitář aplikace pro distribuci Debian, ze kterého se nainstaluje aplikace Koha (Obr. 21). Vytvoří databázi na základě parametrů, dále nainstaluje aplikaci a podporu českého jazyka. Server Apache je nakonfigurován s podporou protokolu HTTPS.



Obrázek 21 Aplikace Koha [zdroj: autor]

Role gitea

Role gitea přidá systémového uživatele a skupinu git a vytvoří adresářovou strukturu pro instalaci binárního souboru. Role stáhne binární soubor aplikace Gitea do předem definovaného adresáře, vygeneruje soubor ze šablony na automatický start služby gitea démonem systemd. Aplikace Gitea je zobrazena na Obr. 22.



Obrázek 22 Aplikace Gitea [zdroj: autor]

Role ojs

Role ojs vytoří uživatele ojs a skupinu, která bude dále použita pro konfiguraci adresáře. Role stáhne instalační archiv aplikace OJS a nainstaluje ho do předem určeného adresáře. Dále vygeneruje konfigurační soubory pro server Apache s podporou SSL certifikátů a nastaví automatický start webového serveru Apache. Aplikace OJS je zobrazena na Obr. 23 až Obr. 26.

Ostatní

Další role obsahují podpůrný software, který se používá pro konfiguraci aplikací.

Role shibboleth

Role shibboleth nainstaluje základní instalační soubory z Debian repozitáře pro podporu Shibboleth SP.

Role solr

Role nainstaluje vyhledávací software Apache Solr, který některé aplikace potřebují ke svému provozu.

8.3 Ukázková instalace aplikace OJS

V následující ukázce je zobrazen zkrácený průběh instalace aplikace OJS prostřednictvím Ansible pomocí vytvořeného scénáře. Detail tohoto scénáře je zobrazen v příloze č. II.

Spuštění *ansible-playbook* z příkazové řádky operačního systému Linux (parametry jsou popsány v kapitole 7.6).

```
ansible-playbook -i inventories/dev/hosts.yml playbook-ojs.yml -K
```

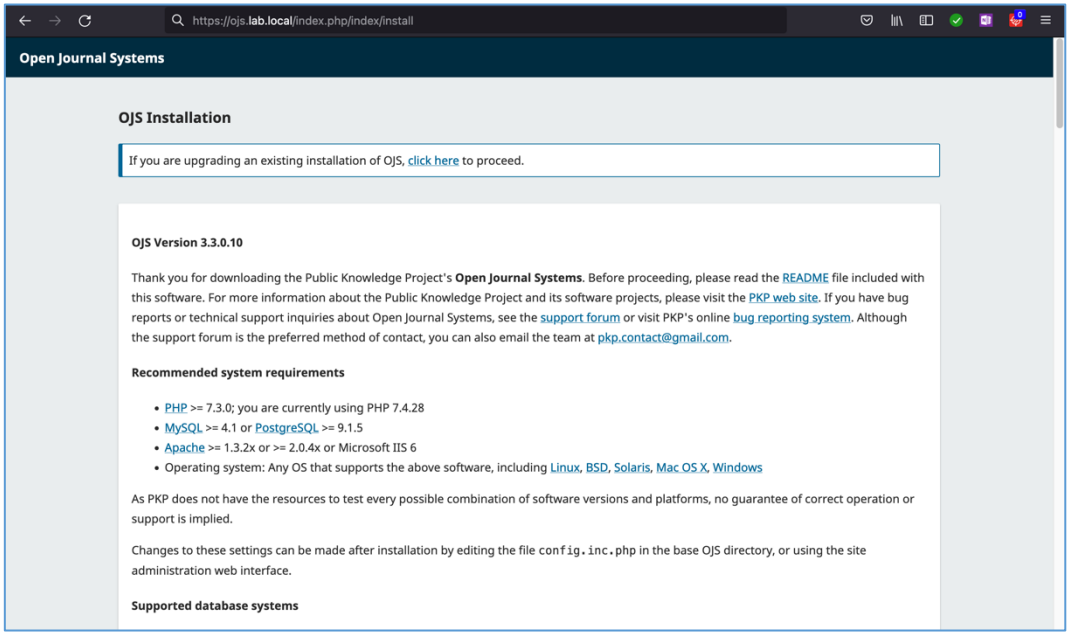
Průběh běhu scénáře (zkráceno):

```
BECOME password:

PLAY [ojs]
*********************************************************************

TASK [Gathering Facts]
*********************************************************************

ok: [192.168.1.184]

TASK [community/geerlingguy.apache : Include OS-specific variables.]
*********************************************************************

ok: [192.168.1.184]

TASK [community/geerlingguy.apache : Include variables for Amazon Linux.]
*********************************************************************

skipping: [192.168.1.184]

TASK [community/geerlingguy.apache : Define apache_packages.]
*********************************************************************

ok: [192.168.1.184]

.
```

Ukončení běhu scénáře a informace o výsledku (význam jednotlivých polí je popsán v kapitole 7.6)

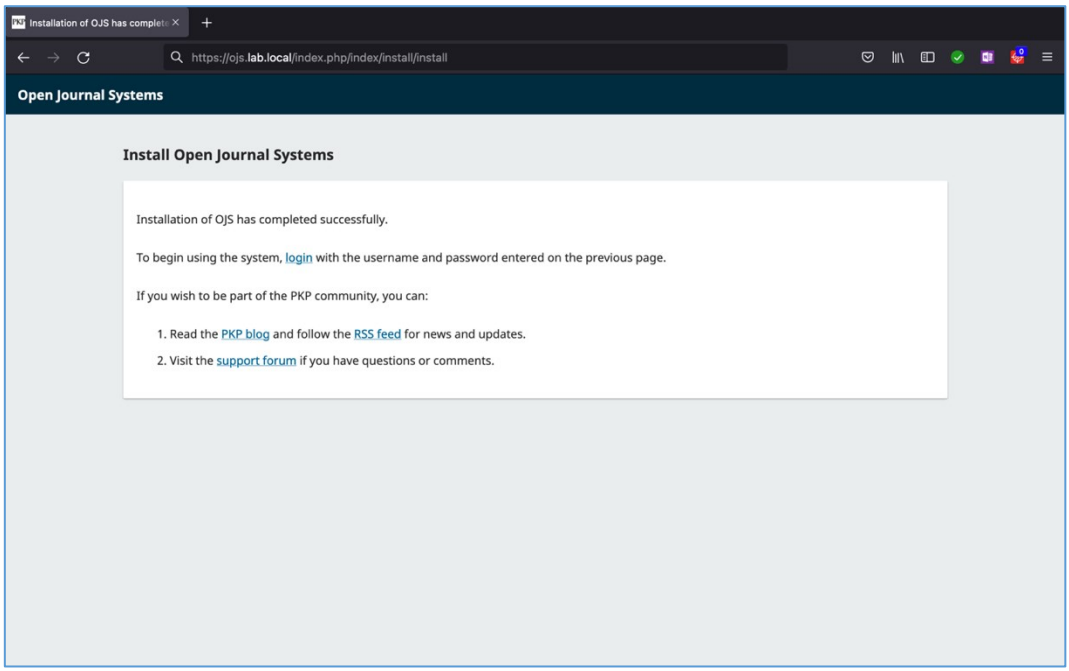
```
.
PLAY RECAP
*********************************************************************

192.168.1.184              : ok=85   changed=33   unreachable=0    failed=0
skipped=32   rescued=0    ignored=0
```

Výsledkem je nainstalovaná aplikace OJS připravená ke konfiguraci (Obr. 23).

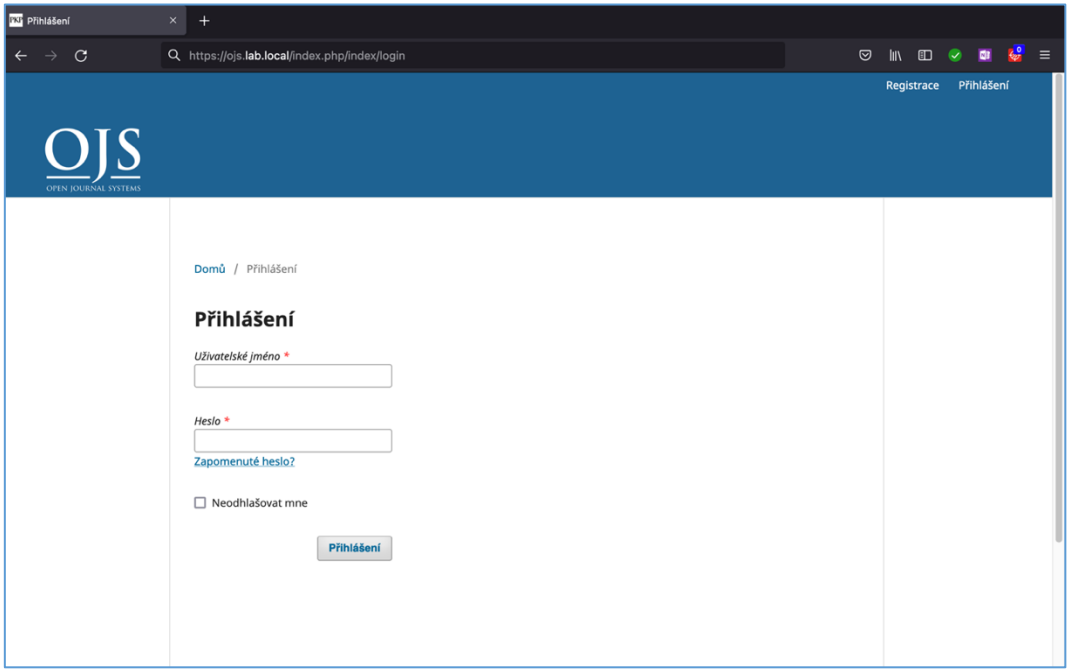


Obrázek 23 Aplikace OJS konfigurace [zdroj: autor]

Po vyplnění všech údajů (nastavení administrátora, databáze, jazyku) a potvrzení instalátor zobrazí stránku (Obr. 24) o úspěšné instalaci. Je zde odkaz na přihlášení do systému (login). Administrátor aplikace nebo uživatel se přihlašuje na stránce zobrazené na Obr. 25.

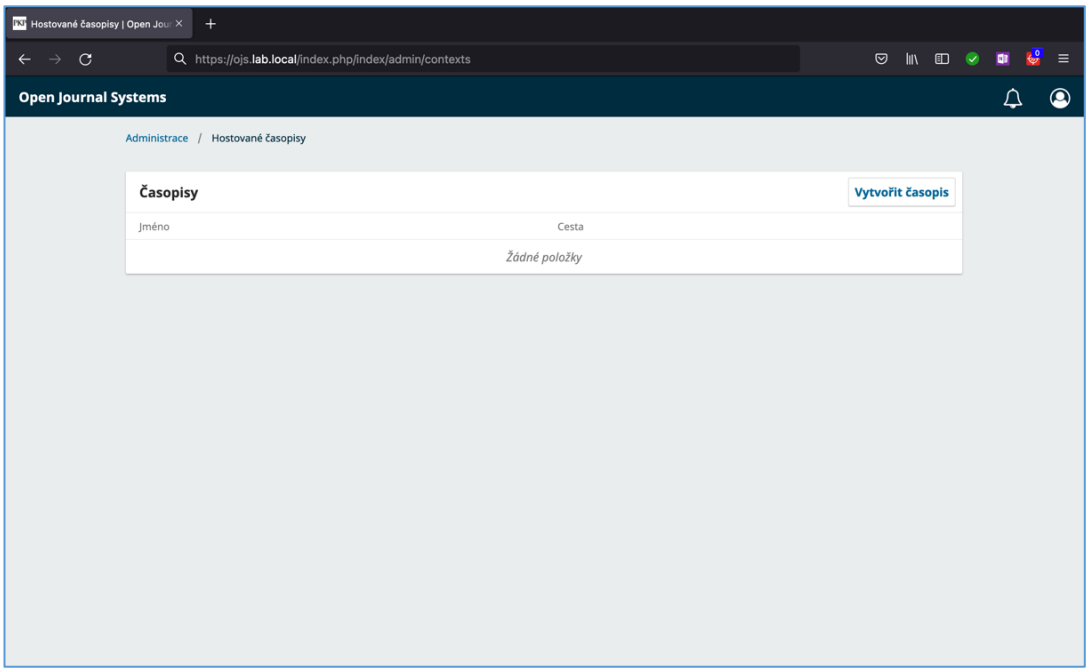


Obrázek 24 Aplikace OJS dokončení konfigurace [zdroj: autor]



Obrázek 25 Aplikace OJS přihlášení [zdroj: autor]

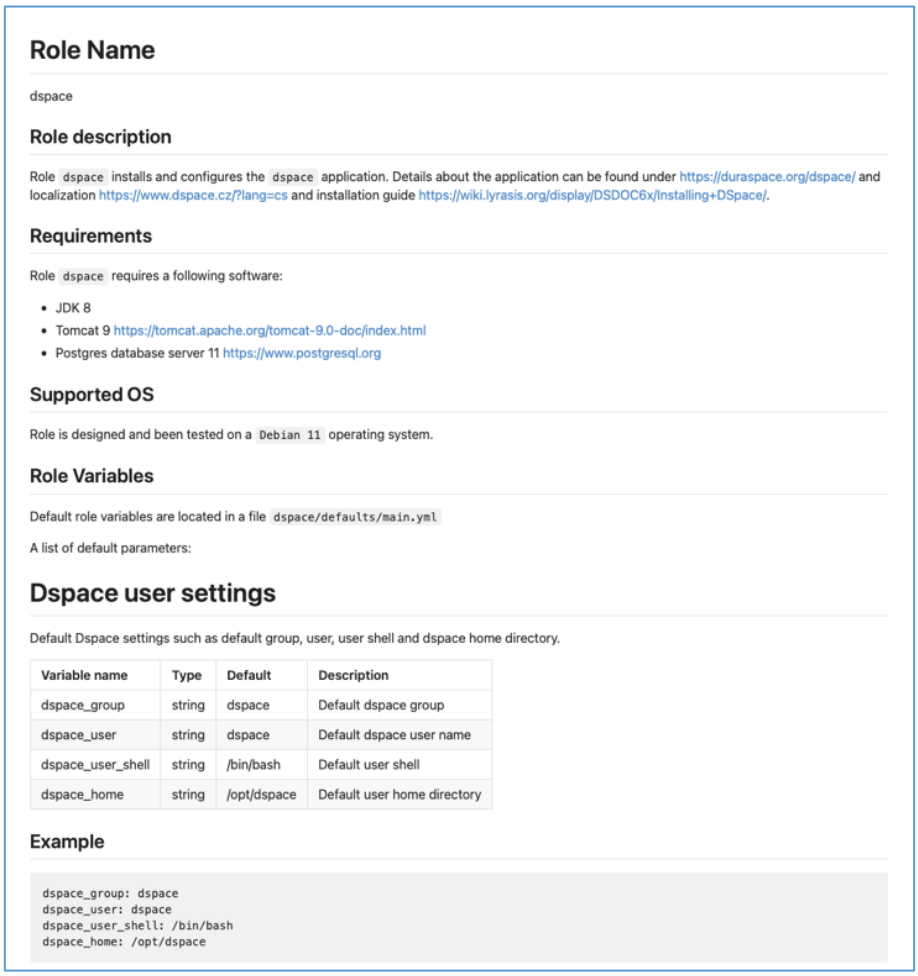
Po úspěšném přihlášení administrátora se zobrazí stránka určená pro správu aplikace OJS (Obr.26).



Obrázek 26 Aplikace OJS administrace [zdroj: autor]

8.4 Dokumentace

Dokumentace je součástí celého projektu i jednotlivých rolí, je vytvořena v jazyce Markdown. Každá role obsahuje soubor README.md (Obr. 27), kde jsou popsány jednotlivé proměnné a jejich výchozí parametry.



Role Name

dspace

Role description

Role `dspace` installs and configures the `dspace` application. Details about the application can be found under https://duraspace.org/dspace/ and localization https://www.dspace.cz/?lang=cs and installation guide https://wiki.lyrasis.org/display/DSDOC6x/Installing+DSpace/.

Requirements

Role `dspace` requires a following software:

- JDK 8
- Tomcat 9 https://tomcat.apache.org/tomcat-9.0-doc/index.html
- Postgres database server 11 https://www.postgresql.org

Supported OS

Role is designed and been tested on a `Debian 11` operating system.

Role Variables

Default role variables are located in a file `dspace/defaults/main.yml`

A list of default parameters:

Dspace user settings

Default Dspace settings such as default group, user, user shell and dspace home directory.

Variable name	Type	Default	Description
dspace_group	string	dspace	Default dspace group
dspace_user	string	dspace	Default dspace user name
dspace_user_shell	string	/bin/bash	Default user shell
dspace_home	string	/opt/dspace	Default user home directory

Example

```
dspace_group: dspace
dspace_user: dspace
dspace_user_shell: /bin/bash
dspace_home: /opt/dspace
```

Obrázek 27 Dokumentace role [zdroj: autor]

9 TESTOVÁNÍ

Ve spolupráci s administrátorem IT oddělení knihovny UTB, byly otestovány Ansible scénáře určené pro nasazování aplikací.

9.1 Testovací prostředí

Automatizované instalace aplikací byly otestovány v lokálním vývojovém prostředí, pro které byla zvolena kombinace Oracle VirtualBox, HashiCorp Vagrant, Chef Workstation a Ansible. Prostředí Chef Workstation je primárně určeno pro vývoj Chef cookbooks, ale prostřednictvím doplňkového modulu ho lze použít pro vývoj Ansible rolí a scénářů. Kombinace těchto aplikací tvoří robustní lokální vývojové prostředí bez nutnosti interakce s produkčními servery.

Ansible scénáře byly také úspěšně otestovány v produkčním prostředí virtualizačního software Microsoft Hyper-V, který používá knihovna UTB.

9.2 Zpracování komentářů

Veškeré komentáře od administrátora byly zpracovány a opraveny. Správce IT systémů knihovny UTB byl proškolen na konfigurační management Ansible a konfiguraci vytvořených rolí a scénářů. Pro jednotlivé role a scénáře byla vytvořena dokumentace, která je součástí vytvořeného kódu.

9.3 Integrační testy

Byly vytvořeny ukázkové integrační testy, které by měly být součástí jednotlivých rolí pro možnou automatizaci celého procesu testování jednotlivých komponentů. Tyto testy byly vytvořeny v programu Inspec, který může být přidán jako další prvek do lokálního testovacího prostředí.

Následuje ukázka integračního testu, který otestuje, zda byl nainstalován softwarový balík programovacího jazyka PHP. Na řádku č.1 se nachází popis testu, který začíná klíčovým slovem *describe*, na řádku č.2 je popsána akce pomocí klíčového slova *it*. Balík by měl být nainstalován. Poslední řádek č.3 označuje konec testu pomocí klíčového slova *end*.

```
1 describe package('php') do
2    it { should be_installed }
3 end
```

9.4 Kontrola syntaxe

Jednotlivé role a scénáře (playbooks) byly zkontrolovány pomocí aplikace ansible-lint, která provede kontrolu syntaxe a kontrolu stylu.

Zkrácená ukázka výstupu programu *ansible-lint*:

```
1 $ ansible-lint main.yml
2   WARNING  Listing 2 violation(s) that are fatal
3   fqcn-builtins: Use FQCN for builtin actions.
4   main.yml:4 Task/Handler: Add group
5
6   fqcn-builtins: Use FQCN for builtin actions.
7   main.yml:9 Task/Handler: Add user
```

Na řádku č.1 je spuštěn program, na následujícím řádku č.2 program vypsal upozornění na dva nalezené problémy v rámci scénáře main.yml. Obě chyby znamenají, že úlohy (tasks) neobsahují *ansible.builtin.user and ansible.builtin.group*. Po opravě chyb proběhne další kontrola pomocí programu ansible-lint bez varování.

10 ZABEZPEČENÍ

Aplikace, které se instalují pomocí konfiguračního managementu obsahují velmi často citlivá data jako jsou uživatelská jména nebo hesla, nebo přístupové tokeny a SSL certifikáty. Je proto velmi důležité tyto údaje chránit proti možnému zneužití. Hesla by z tohoto důvodu neměla být ukládána ve své prosté formě bez jakéhokoliv zabezpečení.

10.1 Ansible Vault

Součásti Ansible je vlastní správce hesel, Ansible Vault (trezor). Ansible Vault, umožňuje zašifrovat hesla na úrovni proměnné nebo souboru. Hodnota proměnné může být součástí scénáře v zašifrované podobě. [110]

Jako například:

```
postgresql_db_password:$ANSIBLE_VAULT;1.1;AES256
61653139626661623936326638393030636132386262646165616431393539313566306532313537
35643266363438396636633063646431316263333161376 30a643165326436356164376334353237
32366434626564343933616262323565306238306433366263363637623462333961373834656132
30353933633963656 20a6335656564643862366630623362646131633132373266666664666616138
39323931643232656664626530633661643533663833343864366163666237323930
```

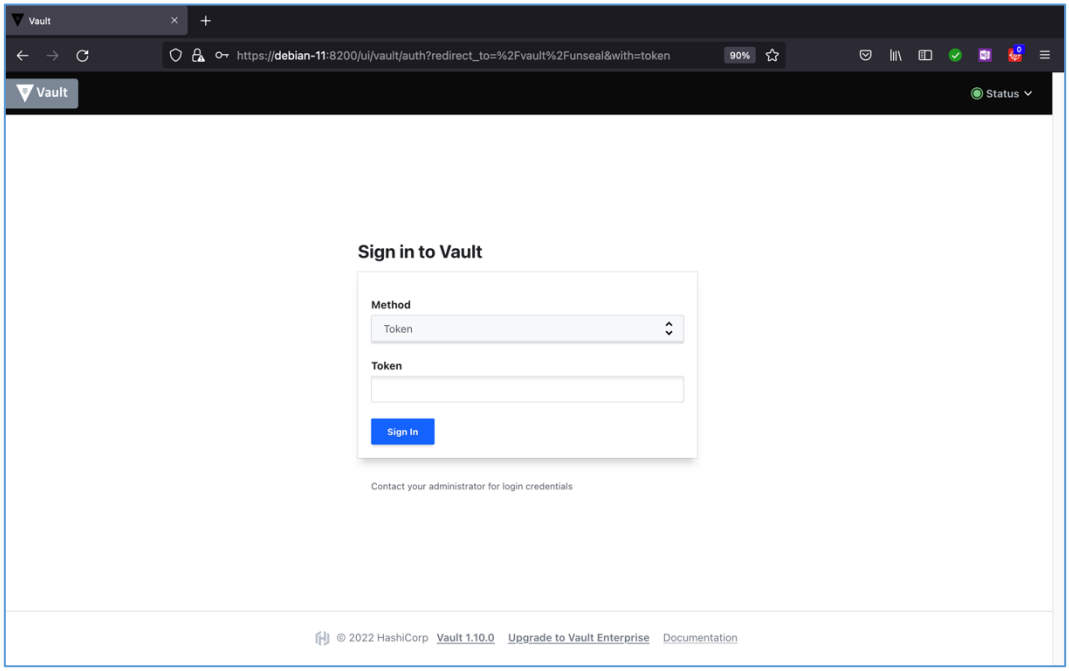
Program *ansible-playbook* by se volal s parametrem *–ask-vault-pass*

```
ansible-playbook playbook-dspace.yml –ask-vault-pass
```

Nevýhodou programu Ansible Vault je omezení pouze pro Ansible. Nelze ho použít jako centrální server pro správu hesel s různými přístupy k heslům pomocí přístupových práv.

10.2 HashiCorp Vault

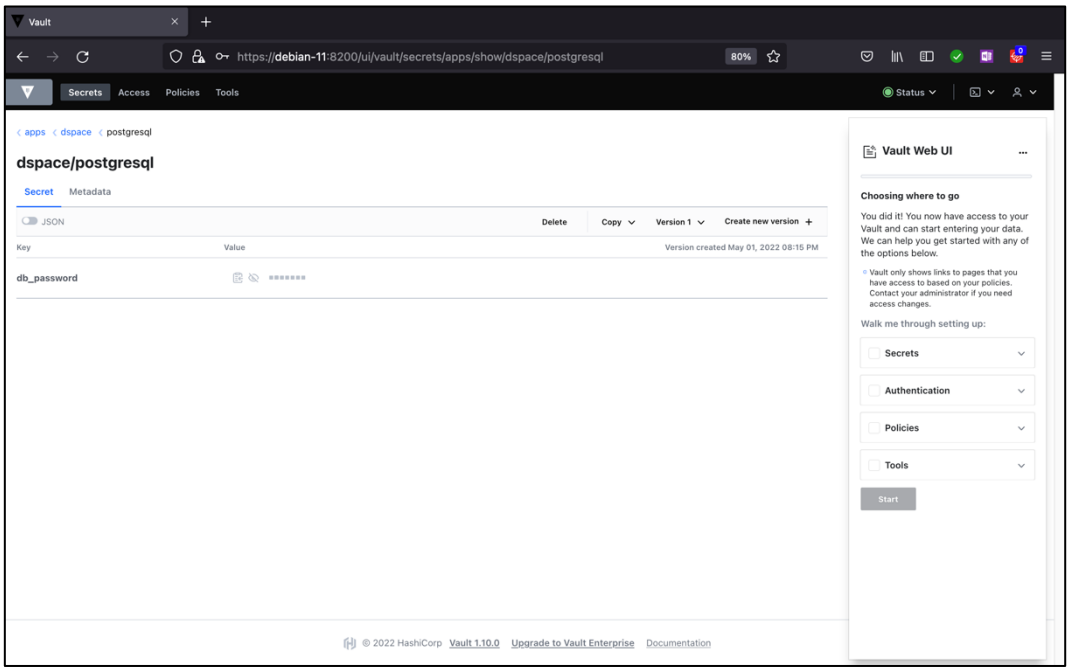
Dalším významným správcem hesel je software Vault od firmy HashiCorp (HC) (Obr. 28). HC Vault se nabízí ve volné formě, verzi pro cloud a ve verzi Enterprise. Verze pro cloud nepotřebuje vlastní serverovou infrastrukturu a nabízí stejné možnosti jako verze Entreprise prémiovou podporu. Obě verze nabízí škálovatelnou infrastrukturu s podporou více serverů. [111][112]



Obrázek 28 Aplikace HashiCorp Vault [zdroj: autor]

HC Vault podporuje ukládání hesel ve formě KV – Key:Value (klíč:hodnota), PKI Certifikátů, podporuje poskytovatele cloudových služeb jako jsou Amazon AWS (Amazon Web Services), Google Cloud nebo Microsoft Azure.

Na následujícím Obr. 29 je zobrazen příklad použití modulu KV (Key/Value).



Obrázek 29 Aplikace HashiCorp Vault KV [zdroj: autor]

V rámci Ansible scénáře pro aplikaci DSpace by se k heslům uloženým v aplikaci HC Vault přistupovalo následovně pomocí tzv. tokenu.

```
1 vars:
2    vault_server: debian-11
3    vault_secret_path: apps/data/dspace
4    vault_token: „{{ lookup(,env‘,‘VAULT_TOKEN‘) | de-
fault(‘XXXXXXXXXXXXXXXX‘,true) }}“
5    postgresql_databases:
6      - name: dspace
7        login_password: „{{ lookup(,community.hashi_vault.hashi_vault‘,
,secret={{ vault_secret_path }}/postgresql:db_password token={{ vault_to-
ken }} url=https://{{ vault_server }}:8200‘, errors=‘ignore‘) | de-
fault(,utb1234‘,true) }}“
```

Klíčové slovo *vars* definuje sekci definic proměnných, řádky č.2-4 definují jednotlivé proměnné. Řádek č. 4, proměnná *vault_token* definuje token k přístupu k HashiCorp Vault serveru. Token se nahraje buď z proměnné prostředí nebo je zde i výchozí hodnota pomocí proměnné *default*. Výchozí hodnota je zde uvedena pouze jako příklad, pro budoucí použití a neměla by být v žádném případě uložena ve verzovacím systému kvůli případnému nežádoucímu zneužití nepovolanou osobou.

Na řádku č.7 je uveden příklad, jak se přistupuje k uloženému heslu v HC Vault serveru pomocí Ansible modulu *community.hashi_vault*. Je zde také uvedena výchozí hodnota proměnné *default*.

K datům uloženým v HC Vault lze přistupovat pomocí lokální instalace programu vault a na linuxových operačních systémech nebo systému macOS. Další možností je přístup pomocí REST API, které lze stejně jako lokální instalace programu vault využít ve skriptovacích jazycích.

Příklad použití programu *vault* na příkazové řádce:

```
vault kv get -field=user apps/dspace
user1234
```

HC Vault lze nasadit v rámci celé IT infrastruktury jako centrální server pro správu hesel a SSL certifikátů. Produkt není omezen pouze pro program Ansible. HC Vault podporuje správu uživatelů, skupin a úrovně přístupu k uloženým datům.

HC Vault podporuje jako backend pro ukládání dat úložiště HC Consul. Použití HC Consul poskytuje možnost pro využití vysoké dostupnosti pro HC Vault.

ZÁVĚR

Cílem bakalářské práce bylo vytvořit scénáře pro automatizované nasazování (deployment) aplikací spravovaných IT oddělením knihovny Univerzity Tomáše Bati. Po důkladné analýze požadavků a aplikací byla vybrána volně šiřitelná verze software pro správu konfigurací – Ansible. Tento nástroj splňuje požadavky na snadné psaní automatizačních úloh a snadnou implementaci bez nutnosti vytváření složité infrastruktury.

Vytvořené scénáře fungují nezávisle na použitém virtualizačním nástroji, jakým je například Oracle VirtualBox použitý pro testovací prostředí nebo Microsoft Hyper-V, který používá knihovna v produkčním prostředí.

V prostředí Ansible byly vytvořeny aplikační role pro nasazování aplikací DSpace, Koha, Gitea, OJS, VuFind® a WordPress. Dále bylo vytvořeno několik podpůrných rolí určených pro již zmíněné aplikace. Aplikační scénáře a role se mohou dále modifikovat podle potřeb správců aplikací knihovny UTB.

Pro zvýšení bezpečnosti uchovávání hesel byla otestována aplikace pro centrální správu hesel Vault od firmy HashiCorp. Pro tuto aplikaci byla vytvořena role a scénář. Scénáře pro nasazování knihovních aplikací byly otestovány s i bez použití aplikace Vault.

Na základě zkušeností s nasazováním aplikací lze doporučit následující varianty postupu.

Pro vývoj scénářů a rolí aplikací použít prostředí s programy Oracle VirtualBox nebo Microsoft Hyper-V, HashiCorp Vagrant a Ansible. Dále pak program Chef Workstation pro orchestraci virtuálních serverů.

První variantou pro nasazovaní aplikací do produkčního prostředí je použití programu Chef Workstation, kterého součástí je Chef Kitchen, s podporou pro Microsoft Hyper-V. Program Chef Kitchen bude spouštět Ansible scénáře.

V druhé variantě program Ansible komunikuje přímo s virtuálním serverem v prostředí Hyper-V ze systému určeného pro administraci serverů.

Poslední možností je řešení s použitím programu Packer, kterého součástí je modul pro podporu Ansible. Modul (provisioner) umožňuje vykonávat Ansible scénáře. Problematikou nasazení programu Packer v prostředí knihovny UTB se zabývala diplomová práce [113].

SEZNAM POUŽITÉ LITERATURY

[1] KRISHNA, Vijaya. Comparing Configuration Management Tools: Chef vs. Puppet vs. Ansible. *GSPANN* [online]. Milpitas: GSPANN Technologies, c2022 [cit. 2022-05-03]. Dostupné z: https://www.gspann.com/resources/blogs/puppet-vs-chef-vs-ansible/

[2] What is configuration management?. *Red Hat* [online]. Red Hat, c2022, July 11, 2019 [cit. 2022-05-03]. Dostupné z: https://www.redhat.com/en/topics/automation/what-is-configuration-management

[3] HOFFMAN, CHRIS. How to Write a Batch Script on Windows. *Tutorialspoint* [online]. LifeSavvy Media, 2017, JUL 5, 2017 [cit. 2022-05-04]. Dostupné z: https://www.howtogeek.com/263177/how-to-write-a-batch-script-on-windows/

[4] WHEELER, Sean, Hananya JACOBSON a Shawn KOON. What is PowerShell?. *Microsoft* [online]. Microsoft, 2022, 02/16/2022 [cit. 2022-05-04]. Dostupné z: https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.2

[5] BENEŠ, Miroslav. Funkcionální programování – Kapitola 1. Funkcionální jazyky: 1.1. Klasifikace programovacích jazyků. *Katedra informatiky FEI VŠB-TU* [online]. Ostrava: Katedra informatiky FEI VŠB-TU [cit. 2022-05-03]. Dostupné z: http://www.cs.vsb.cz/navrat/vyuka/flp/texty/fp/ch01s01.html

[6] HUMMER, Waldemar, Florian ROSENBERG, Fábio OLIVEIRA a Tamar EILAM. Testing Idempotence for Infrastructure as Code. EYERS, David a Karsten SCHWAN, ed. *Middleware 2013* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, 2013, s. 368-388 [cit. 2022-05-24]. Lecture Notes in Computer Science. ISBN 978-3-642-45064-8. Dostupné z: doi:10.1007/978-3-642-45065-5_19

[7] *Technický slovník naučný*. Praha: Encyklopedický dům, 2005. ISBN 80-860-4420-3.

[8] Seznam.cz deployment: Slovník. *Seznam.cz* [online]. Seznam.cz, c1996–2022 [cit. 2022-05-03]. Dostupné z: https://slovnik.seznam.cz/preklad/anglicky_cesky/deployment

[9] SCS.ABZ.CZ: ABZ slovník cizích slov. *ABZ.cz: ABZ slovník cizích slov* [online]. Ostrava: ABZ knihy, c2005-2022 [cit. 2022-05-03]. Dostupné z: https://slovnik-cizich-slov.abz.cz/web.php/hledat?cizi_slovo=idempotentni&typ_hledani=prefix

[10] MAAS, Mark. WHITEPAPER: ANSIBLE IN DEPTH. *Ansible.com* [online]. Red Hat, c2017 [cit. 2022-05-03]. Dostupné z: https://www.ansible.com/hubfs/pdfs/Ansible-InDepth-WhitePaper.pdf

[11] BURGESS, Mark. Biographical information. *Homepage mark burgess* [online]. [cit. 2022-02-06]. Dostupné z: http://markburgess.org/bio.html

[12] ZAMBONI, Diego. *Learning CFEngine 3*. Sebastopol (California): O'Reilly Media, c2012. ISBN 978- 1449312206.

[13] TSALOLIKHIN, Aleksey. Relative Origin of Cfengine, Puppet and Chef. *Vertical Sysadmin* [online]. Vertical Sysadmin, 2010 [cit. 2022-02-06]. Dostupné z: http://verticalsysadmin.com/blog/relative-origins-of-cfengine-chef-and-puppet/]

[14] BURGEES, Mark. Promise Theory—What Is It?. *LINUX JOURNAL* [online]. Slashdot Media, c2022, October 24, 2014 [cit. 2022-05-03]. Dostupné z: https://www.linuxjournal.com/content/promise-theory—what-it

[15] Find the offering that fits your needs. *CFEngine* [online]. Palo Alto (California): Northern.tech AS, c2021 [cit. 2022-02-06]. Dostupné z: https://cfengine.com/community-vs-enterprise-comparison/

[16] How does CFEngine work?. *CFEngine* [online]. Palo Alto (California): Northern.tech AS, c2021 [cit. 2022-02-06]. Dostupné z: https://docs.cfengine.com/docs/3.19/guide-faq-how-does-cfen- ine-work.html

[17] Promises Available in CFEngine. *Cfengine* [online]. Palo Alto (California): Northern.tech AS, c2021 [cit. 2022-02-06]. Dostupné z: https://docs.cfengine.com/docs/3.19/guide-writing-and-serving-po- icy-promises-available-in-cfengine.html

[18] RAJNEESH. *CFEngine 3: Beginner's Guide*. Birmingham: Packt Publishing, c2011. ISBN 978-- -84951-498-9.

[19] The CFEngine Components. *CFEngine* [online]. Palo Alto (California): Northern.tech AS, 2021 [cit. 2022-02-06]. Dostupné z: https://docs.cfengine.com/docs/3.5/manuals-components.html]

[20] BURGESS, Mark a Diego ZAMBONI. *Modern Infrastructure Engineering with CFEngine 3* [online]. Berkeley: The USENIX Association, 2012 [cit. 2022-05-24]. ISBN 978-1-931971-98-0. Dostupné z:

https://www.usenix.org/system/files/lisa/books/usenix_26_cfengine3_frontmatter.pdf

[21] ARCHITECTURE AND SECURITY. *CFEngine* [online]. Palo Alto (California): Northern.tech AS, 2021 [cit. 2022-02-06]. Dostupné z: https://docs.cfengine.com/docs/archive/manuals/st-security.html

[22] KANIES, Luke. Entrepreneur, Stage 1: Bootstrapping, Burnout, and Babies. *WRITING BY LUKE KANIES: Advisor, writer, and founder focused on design, strategy, and products*[online]. 2020, JULY 7, 2020 [cit. 2022-05-03]. Dostupné z: https://lukekanies.com/entrepreneur-stage-1-bootstrapping-burnout-and-babies/

[23] KANIES, Luke. Handing over the reins. *Puppet* [online]. Puppet, c2022, 26 September 2016 [cit. 2022-05-03]. Dostupné z: https://puppet.com/blog/handing-over-reins/

[24] Supported operating systems. *Puppet* [online]. Puppet, c2021 [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/pe/2021.2/supported_operating_systems.html

[25] Configuration automation. *Puppet* [online]. Puppet, c2022 [cit. 2022-05-03]. Dostupné z: https://puppet.com/products/puppet-enterprise/open-source-comparison/#compare

[26] Certificate authority and SSL. Puppet [online]. Puppet, c2022 [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/7/ssl_certificates.html

[27] Puppet overview. *Puppet* [online]. Puppet, c2022 [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/7/puppet_overview.html#puppet_overview

[28] The Puppet language. *Puppet* [online]. C2022 [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/7/puppet_language.html

[29] Modules overview. Puppet [online]. Puppet, c2022 [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/7/modules_fundamentals.html

[30] Creating templates using Embedded Puppet. *Puppet* [online]. Puppet, c2022 [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/6/lang_template_epp.html#lang_template_epp

[31] Creating templates using Embedded Ruby. Puppet [online]. Puppet, c2022 [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/6/lang_template_erb.html#lang_template_erb

[32] About Hiera. Puppet [online]. C2022: Puppet [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/7/hiera_intro.html#hiera_intro

[33] Facter: Core Facts. Puppet [online]. Puppet, c2022 [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/6/core_facts.html

[34] Overview and requirements. *Puppet* [online]. Puppet, c2022 [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppetdb/6/overview.html]

[35] Puppet agent on *nix systems. *Puppet* [online]. Puppet, c2022 [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/6/services_agent_unix.html

[36] Puppet agent on Windows. *Puppet* [online]. C2022: Puppet [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/6/services_agent_windows.html#services_agent_windows

[37] SARTI, Glenn. Combining PowerShell, Bolt and Puppet Tasks – Part 1. *Puppet* [online]. Puppet, c2022, 20 July 2018 [cit. 2022-05-03]. Dostupné z: https://puppet.com/blog/combining-powershell-bolt-and-puppet-tasks-part-1/

[38] What is Puppet?. *Puppet* [online]. Puppet, c2022 [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/7/what_is_puppet.html

[39] Overview of Puppet's architecture. *Puppet* [online]. Puppet, c2021 [cit. 2022-05-23]. Dostupné z: https://puppet.com/docs/puppet/5.5/architecture.html

[40] PuppetDB. *Puppet* [online]. Puppet, c2022 [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/6/puppetdb_overview.html

[41] Welcome to Puppet Forge. *Forge* [online]. Puppet [cit. 2022-05-03]. Dostupné z: https://forge.puppet.com

[42] MCLELLAN, Bryan. Chef 11 Released!. *Progress Chef* [online]. c2022 [cit. 2022-05-24]. Dostupné z: https://www.chef.io/blog/chef-11-released

[43] JACOB, Adam. Happy 10th Anniversary Chef. *Progress Chef* [online]. Progress Software Corporation, 2018, September 12, 2018 [cit. 2022-05-03]. Dostupné z: https://www.chef.io/blog/happy-10th-anniversary-chef

[44] ROBBINS, Jesse. Announcing Chef. *Progress Chef* [online]. Progress Software Corporation, 2009, January 16, 2009 [cit. 2022-05-03]. Dostupné z: https://www.chef.io/blog/announcing-chef

[45] BAKER, Kim. Progress Announces Acquisition of Chef. *Progress* [online]. Progress Software Corporation, 2020, Sep 08, 2020 [cit. 2022-05-03]. Dostupné z:

https://investors.progress.com/news-releases/news-release-details/progress-announces-acquisition-chef

[46] Chef-solo. *Progress Chef* [online]. Progress Software Corporation, 2022, March 7, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/chef_solo/].

[47] Chef Infra Overview. *Progress Chef* [online]. Progress Software Corporation, 2022, April 14, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/chef_overview/

[48] About Chef Workstation. *Progress Chef* [online]. Progress Software Corporation, 2022, March 16, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/workstation/

[49] About Recipes. *Progress Chef* [online]. Progress Software Corporation, 2022, April 21, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/recipes/

[50] About Cookbooks. *Progress Chef* [online]. Progress Software Corporation, 2022, April 21, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/cookbooks/

[51] An Overview of Chef InSpec. *Progress Chef* [online]. Progress Software Corporation, 2022, April 8, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/inspec/

[52] About Knife. *Progress Chef* [online]. Progress Software Corporation, 2022, April 8, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/workstation/knife/

[53] Test Kitchen. *Progress Chef* [online]. Progress Software Corporation, 2021, February 16, 2021 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/workstation/kitchen/

[54] About Ohai. *Progress Chef* [online]. Progress Software Corporation, 2022, March 25, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/ohai/

[55] Chef Supermarket. *Progress Chef* [online]. Progress Software Corporation, 2022, January 20, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/supermarket/

[56] Chef Infra Server Overview. *Progress Chef* [online]. Progress Software Corporation, 2021, October 13, 2021 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/server/

[57] About Nodes. *Progress Chef* [online]. Progress Software Corporation, 2022, April 21, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/nodes/

[58] Security. *Progress Chef* [online]. Progress Software Corporation, 2021, October 13, 2021 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/server/server_security/

[59] Chef Infra Client Security. *Progress Chef* [online]. Progress Software Corporation, 2022, February 5, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/chef_client_security/

[60] INTRODUCTION TO SALT. *SALTSTACK* [online]. SaltStack, c2022 [cit. 2022-05-03]. Dostupné z: https://docs.saltproject.io/en/latest/topics/index.html#the-30-second-summary

[61] SEBENIK, Craig a Thomas HATCH. *Salt essentials*. Sebastopol, California: O'Reilly, 2015. ISBN 978-1491900635.

[62] HATCH, Thomas. Salt Air 1 – Salt History, New Developments, & Salt Community Love. *SALTSTACK* [online]. SaltStack, 2012, November 9, 2012 [cit. 2022-05-03]. Dostupné z: https://saltproject.io/salt-air-1-salt-history-new-developments-salt-community-love/

[63] Vmware and SaltStack. *Vmware* [online]. Vmware, c2022 [cit. 2022-05-03]. Dostupné z: https://www.vmware.com/company/acquisitions/saltstack.html

[64] SALT SSH. *SALTSTACK* [online]. SaltStack, 2022, May 03, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.saltproject.io/en/latest/topics/ssh/index.html

[65] SALT PROXY MINION. *SALTSTACK* [online]. SaltStack, 2022, May 03, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.saltproject.io/en/latest/topics/proxyminion/index.html

[66] Salt overview. *SALTSTACK* [online]. Vmware, c2021 – 2022 [cit. 2022-05-03]. Dostupné z: https://docs.saltproject.io/salt/user-guide/en/latest/topics/overview.html

[67] CONFIGURING THE SALT MINION. *SALTSTACK* [online]. SaltStack, 2022, May 03, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.saltproject.io/en/latest/ref/configuration/minion.html

[68] HOW DO I USE SALT STATES?. *SALTSTACK* [online]. SaltStack, 2022, May 03, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.saltproject.io/en/latest/topics/tutorials/starting_states.html

[69] GRAINS. *SALTSTACK* [online]. SaltStack, 2022, May 03, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.saltproject.io/en/latest/topics/grains/index.html

[70] PILLAR WALKTHROUGH. *SALTSTACK* [online]. SaltStack, 2022, May 03, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.saltproject.io/en/latest/topics/tutorials/pillar.html

[71] Salt security. *SALTSTACK* [online]. VMware, c2021 - 2022 [cit. 2022-05-03]. Dostupné z: https://docs.saltproject.io/salt/user-guide/en/latest/topics/security.html

[72] HOCHSTEIN, Lorin a René MOSER. *Ansible: up and running: automating configuration management and deployment the easy way*. Second edition. Beijing: O'Reilly, 2017. ISBN 978-1491979808.

[73] GEERLING, Jeff. *Ansible for DevOps: Server and configuration management for humans*. Leanpub, c2014-2016. ISBN 978-0-9863934-1-9.

[74] DEHAAN, Michael. The Inside Playbook: The Origins of Ansible. *Red Hat Ansible* [online]. Ansible, 2013, December 8, 2013 [cit. 2022-05-03]. Dostupné z: https://www.ansible.com/blog/2013/12/08/the-origins-of-ansible

[75] ROMETSCH, Ben. How Ansible got started and grew. *Opensource.com* [online]. Red Hat, 2021, February 16, 2021 [cit. 2022-05-03]. Dostupné z: https://opensource.com/article/21/2/ansible-origin-story

[76] DEHAAN, Michael. The Inside Playbook: Introducing AnsibleWorks!. *Red Hat Ansible* [online]. Ansible, 2013, March 4, 2013 [cit. 2022-05-03]. Dostupné z: https://www.ansible.com/blog/2013/03/04/introducing-ansibleworks

[77] Red Hat to Acquire IT Automation and DevOps Leader Ansible. *Red Hat* [online]. RALEIGH: Red Hat, 2015, October 16, 2015 [cit. 2022-05-03]. Dostupné z: https://www.redhat.com/en/about/press-releases/red-hat-acquire-it-automation-and-devops-leader-ansible

[78] Ansible architecture. *Ansible Documentation* [online]. Ansible project contributors, 2022, Apr 27, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.ansible.com/ansible/latest/dev_guide/overview_architecture.html

[79] How to build your inventory. *Ansible Documentation* [online]. Ansible project contributors, 2022, Apr 27, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

[80] What is an Ansible playbook?. *Red Hat* [online]. Red Hat, 2019, January 8, 2019 [cit. 2022-05-03]. Dostupné z: https://www.redhat.com/en/topics/automation/what-is-an-ansible-playbook

[81] Introduction to modules. *Ansible Documentation* [online]. Ansible project contributors, 2022, Apr 27, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.ansible.com/ansible/latest/user_guide/modules_intro.html

[82] Using collections. *Ansible Documentation* [online]. Ansible project contributors, 2022, Apr 27, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.ansible.com/ansible/latest/user_guide/collections_using.html

[83] Discovering variables: facts and magic variables. *Ansible Documentation* [online]. Ansible project contributors, 2022, Apr 27, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.ansible.com/ansible/latest/user_guide/playbooks_vars_facts.html

[84] Galaxy Documentation: About Galaxy. *Ansible Galaxy* [online]. Red Hat, 2021, Sep 23, 2021 [cit. 2022-05-03]. Dostupné z: https://galaxy.ansible.com/docs/

[85] Red Hat Ansible Automation Platform. *Red Hat: Red Hat Ansible Automation Platform* [online]. Red Hat, c2022 [cit. 2022-05-14]. Dostupné z: https://www.redhat.com/en/technologies/management/ansible

[86] Ansible Tower. *Ansible Documentation* [online]. Ansible project contributors, 2020, Dec 01, 2020 [cit. 2022-05-03]. Dostupné z: https://docs.ansible.com/ansible/2.5/reference_appendices/tower.html

[87] Connection methods and details. *Ansible Documentation* [online]. Ansible project contributors, 2022, Apr 27, 2022 [cit. 2022-05-03

[88] Windows Remote Management. *Ansible Documentation* [online]. Ansible project contributors, 2022, Apr 27, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.ansible.com/ansible/latest/user_guide/windows_winrm.html

[89] What is Capistrano?: Capistrano is a remote server automation tool. *Capistrano* [online]. [cit. 2022-05-04]. Dostupné z: https://capistranorb.com/documentation/overview/what-is-capistrano/

[90] Configuring Hiera. *Puppet* [online]. C2022: Puppet [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/puppet/6/hiera_config_yaml_5.html

[91] Writing plans in YAML. *Puppet* [online]. C2022: Puppet [cit. 2022-05-03]. Dostupné z: https://puppet.com/docs/bolt/latest/writing_yaml_plans.html

[92] Chef Infra Client Release Notes. *Progress Chef* [online]. Progress Software Corporation, 2022, January 18, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.chef.io/release_notes_client/#yaml-recipes

[93] JEVTIC, GORAN. What is SDLC? Phases of Software Development, Models, & Best Practices. *PhoenixNAP* [online]. 2019, MAY 15, 2019 [cit. 2022-05-04]. Dostupné z: https://phoenixnap.com/blog/software-development-life-cycle

[94] What is Application Lifecycle Management?. *Inflectra* [online]. Inflectra Corporation, c2022 [cit. 2022-05-04]. Dostupné z: https://www.inflectra.com/SpiraTeam/Highlights/Understanding-ALM-Tools.aspx

[95] STEPHENS, Rod. *Beginning Software Engineering*. Indianapolis: John Wiley, 2015. ISBN 978-1-118-96914-4.

[96] DOOLEY, John F. *Software Development, Design and Coding: With Patterns, Debugging, Unit Testing, and Refactoring.* 2nd ed. 2017. Berkeley, CA: Apress, 2017. ISBN 978-1-4842-3152-4.

[97] HUMBLE, Jez a David FARLEY. *Continuous delivery: reliable software releases through build, test, and deployment automation*. Upper Saddle River, NJ: Addison-Wesley, 2010. ISBN 978-0321601919.

[98] What Is Load Balancing?. *NGINX: Part of F5* [online]. F5 [cit. 2022-05-09]. Dostupné z: https://www.nginx.com/resources/glossary/load-balancing/

[99] TREMEL, Etienne. Six Strategies for Application Deployment. *The New Stack* [online]. The New Stack, c2022, 21 Nov 2017 [cit. 2022-05-21]. Dostupné z: https://thenewstack.io/deployment-strategies/

[100] Intro To Deployment Strategies: Blue-Green, Canary, And More. *Harness* [online]. harness, c2022, February 23, 2022 [cit. 2022-05-21]. Dostupné z: https://harness.io/blog/continuous-verification/blue-green-canary-deployment-strategies/

[101] PAVLÍK, Jiří. Představení Shibboleth. *Itlib: Informačné technológie a knižnice* [online]. Informačné technológie a knižnice, c2021 [cit. 2022-05-04]. Dostupné z: https://itlib.cvtisr.sk/Články/clanek1510/

[102] About DSpace. *LYRASIS* [online]. DSpace, c2022 [cit. 2022-05-04]. Dostupné z: https://duraspace.org/dspace/about/

[103] What is Gitea?. *Gitea* [online]. The Gitea Authors, c2022 [cit. 2022-05-04]. Dostupné z: https://docs.gitea.io/en-us/

[104] About. *Koha* [online]. CC-BY-SA Koha Library Software Community, c2022 [cit. 2022-05-04]. Dostupné z: https://koha-community.org/about/

[105] Open Journal Systems. *PKP: PUBLIC KNOWLEDGE PROJECT* [online]. Simon Fraser University Library, c2014 [cit. 2022-05-04]. Dostupné z: https://pkp.sfu.ca/ojs/

[106] Democratize Publishing: The freedom to build. The freedom to change. The freedom to share. *Wordpress.org* [online]. Wordpress.org [cit. 2022-05-04]. Dostupné z: https://wordpress.org/about/

[107] *About VuFind®* [online]. Villanova University's Falvey Memorial Library, 2020, Sep 02, 2020 [cit. 2022-05-04]. Dostupné z: https://vufind.org/vufind/about.html

[108] LANE, RYAN. Moving away from Puppet: SaltStack or Ansible?. *RYAN D LANE: A blog with rants about DevOps.* [online]. 2014, AUGUST 4, 2014 [cit. 2022-05-05]. Dostupné z: https://blog.ryandlane.com/2014/08/04/moving-away-from-puppet-saltstack-or-ansible/

[109] All modules. *Ansible Documentation* [online]. Ansible, 2021, Oct 11, 2021 [cit. 2022-05-03]. Dostupné z: https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html

[110] Encrypting content with Ansible Vault. *Ansible Documentation* [online]. Ansible, 2022, May 05, 2022 [cit. 2022-05-03]. Dostupné z: https://docs.ansible.com/ansible/latest/user_guide/vault.html

[111] WHY HASHICORP: Solve your most important cloud infrastructure challenges. *HashiCorp* [online]. HashiCorp [cit. 2022-05-06]. Dostupné z: https://www.hashicorp.com/why

[112] What is Vault?. *HashiCorp Vault* [online]. HashiCorp, 2022 [cit. 2022-05-06]. Dostupné z: https://www.vaultproject.io/docs/what-is-vault

[113] CIBULKA, Milan. *Automatizovaný deployment linuxových serverů a aplikací na platformě Hyper-V*. Zlín: Univerzita Tomáše Bati ve Zlíně, 2021, 71 s. Dostupné také z: http://hdl.handle.net/10563/46105. Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky, Ústav informatiky a umělé inteligence. Vedoucí práce Sysel, Martin.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AES	Advanced Encryption Standard
API	Application Programming Interface
APT	Advanced Package Tool
AWS	Amazon Web Services
BSD	Berkley Software Distribution
CA	Certificate Authority
CI	Continuous Integration
CD	Continuous Development
DB	Database
DEV	Development
DSL	Domain Specific Language
EOF	End Of File
EPP	Embedded Puppet
ERB	Embedded Ruby
FEI	Fakulta Elektrotechniky a Informatiky
FIPS	Federal Information Processing Standards
FQDN	Fully Qualified Domain Name
GIT	Global Information Tracker
GNU	GNU Not Unix
GPL	General Public License
HC	HashiCorp
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IdP	Identity Provider
ILS	Integrated Library System

IT	Informační Technologie
IP	Internet Protocol
IOS	Internetwork Operating System
JDK	Java Development Kit
JRE	Java Runtime Environment
KV	Key/Value
LDAP	Lightweight Directory Access Provider
MD	Message Digest
MQ	Message Queue
OJS	Open Journal Systems
OPS	Operations
OS	Operating System
PE	Puppet Enterprise
PHP	PHP: Hypertext Preprocessor
PKI	Public Key Interchange
PROD	Production
RSA	Rivest-Shamir-Adleman
RBAC	Role Based Access Control
REST	Representational State Transfer
SAML	Security Assertion Markup Language
SHA	Secure Hash Algorithm
SFTP	Simple File Transfer Protocol
SLS	Salt State
SP	Service Provider
SQL	Structured Query Language
SSH	Secure Shell

SSL	Secure Socket Layer
SSO	Single Sign On
SVN	Subversion
TLS	Transport Socket Layer
YAML	YAML A'int Markup Language
YUM	Yellowdog updater modifier
URL	Universal Resource Locator
UTB	Univerzita Tomáše Bati
VŠB	Vysoká Škola Báňská
WinRM	Windows Remote management

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: CD

Přiložené CD obsahuje:

README.md - Dokumentace

ansible – Adresář se zdrojovými kódy

kitchen – Adresář se soubory pro testovací prostředí Kitchen

scripts – Adresář s podpůrnými skripty

PŘÍLOHA P II: UKÁZKOVÝ SCÉNÁŘ ROLE OJS

```
---
- hosts: ojs
  become: yes
  remote_user: ansible
  gather_facts: yes
  roles:
    - community/geerlingguy.apache
    - community/geerlingguy.mysql
    - php
    - ojs


  vars:
    vault_server: debian-11
    vault_secret_path: kv/data/ojs
    vault_token: "{{ lookup('env','VAULT_TOKEN') | default('hvs.ynNTjx-
fzp9UeoRI6yHnXIKsy',true) }}"
    ojs_version: 3.3.0-10


    apache_mods_enabled:
        - rewrite.load
        - ssl.load
        - socache_shmcb.load
    # do not create virtual hosts
    apache_create_vhosts: false
    # remove 000-default file
    apache_remove_default_vhost: true


    mysql_root_password: "{{ lookup('community.hashi_vault.hashi_vault',
'secret={{ vault_secret_path }}/mysql:db_root_password token={{ vault_to-
ken }} url=https://{{ vault_server }}:8200  validate_certs=false', er-
rors='ignore') | default('utb1234',true) }}"
    mysql_bind_address: "{{ ansible_hostname }}"
    mysql_databases:
      - name: ojs
        collation: utf8_general_ci
        encoding: utf8
        replicate: 1
    mysql_users:
      - name: ojs
        host: "%"
```

```
        password: "{{ lookup('community.hashi_vault.hashi_vault',
'secret={{ vault_secret_path }}/mysql:db_user_password token={{ vault_to-
ken }} url=https://{{ vault_server }}:8200  validate_certs=false', er-
rors='ignore') | default('utb1234',true) }}"
        priv: "*.*:ALL"
```