Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Automatizace provozu akademické autentizační proxy

2023

Pavel Vyskočil

Vedoucí práce:
Mgr. Radek Janoštík, Ph.D.

Studijní program: Informační technologie,
kombinovaná forma

Bibliografické údaje

Autor:	Pavel Vyskočil
Název práce:	Automatizace provozu akademické autentizační proxy
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2023
Studijní program:	Informační technologie, kombinovaná forma
Vedoucí práce:	Mgr. Radek Janoštík, Ph.D.
Počet stran:	48
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Pavel Vyskočil
Title:	Automating the operation of the academic authentication proxy
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2023
Study program:	Information Technologies, combined form
Supervisor:	Mgr. Radek Janoštík, Ph.D.
Page count:	48
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Cílem bakalářské práce bylo provést analýzu akademické autentizační proxy využívané spolu s IdM systémem Perun a navrhnout vhodnou architekturu prostředí s důrazem na automatizaci provozu. Výstupem práce je ukázkový předpis pro nasazení akademické autentizační brány v režimu vysoké dostupnosti při splnění požadavků na modulárnost a přizpůsobitelnost řešení pro různé případy nasazení do konkrétních projektů nebo organizací. O funkčnosti navrženého a implementovaného řešení svědčí aktuální stav integrace do produkčního řešení ekosystému Perun, kdy již několik instancí akademické autentizační proxy je spravováno pomocí tohoto předpisu.

Synopsis

The aim of the bachelor thesis was to analyze the academic authentication proxy used with the IdM system Perun and to propose a suitable architecture of the environment with an emphasis on the automation of the operation. The output of the thesis is a sample guideline for deploying an academic authentication gateway in high availability mode while meeting the requirements for modularity and customizability of the solution for different deployment cases in specific projects or organizations. The functionality of the designed and implemented solution is evidenced by the current state of integration into the Perun ecosystem production solution, where several instances of the academic authentication proxy are already managed using this guideline.

Klíčová slova: autentizace; autorizace; AAI; automatizace; nasazení; Perun

Keywords: authentication; authorization; AAI; automation; deployment; Perun

Na tomto místě bych chtěl poděkovat vedoucímu bakalářské práce Mgr. Radku Janoštíkovi, Ph.D. za cenné rady, doporučení a připomínky. Dále bych rád poděkoval svým kolegům za odborné rady a připomínky, a v neposlední řadě také své přítelkyni za podporu.

Odevzdáním tohoto textu jeho autor/ka místopřísežně prohlašuje, že celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

1 Úvod

V rámci distribuovaného akademického prostředí existuje velké množství zdrojů (služeb), které je třeba zpřístupnit vědeckým pracovníkům a uživatelům z různých institucí. Zaměstnanci, studenti a popřípadě další uživatelé jednotlivých organizací jsou v rámci dané organizace zpravidla vybaveni přístupovými údaji (přihlašovací jméno a heslo), pomocí kterých je jim umožněn přístup do interních systémů dané organizace. Výše zmíněné zdroje jsou však provozovány různými organizacemi a je třeba je zpřístupnit i pracovníkům a uživatelům jiných organizací. Jedním z možných řešení je vybudování federovaného prostředí za použití ekosystému Perun, tedy obecně Autentizační a Autorizační Infrastruktury (dále jen AAI), která uživatelům umožnuje přistupovat ke zdrojům provozovaných různými organizacemi za pomoci přihlašovacích údajů ze své domovské instituce.

Ekosystém Perun je budován ve spolupráci sdružení CESNET[1] a národního centra CERIT-SC[2] spolu s Masarykovou univerzitou za účelem poskytování služeb správy identit a řízení přístupu uživatelů ke zdrojům a službám. Celý ekosystém je modulární, což v případě potřeby umožňuje nahradit jednotlivé komponenty alternativní implementací. To umožňuje integraci ekosystému i do prostředí, kde je již nějaká z částí AAI řešení dostupná.

Jednou z hlavních funkcionalit ekosystému je řízení přístupu uživatelů ke službám, kterou zajišťuje komponentna tzv. autentizační proxy. Tato část je velmi komplexní a jsou na ni kladeny velké nároky v oblasti dostupnosti a také bezpečnosti. Jde totiž o centrální komponentu a v případě jejího výpadku není umožněno uživatelům přihlášení na žádnou ze služeb připojených do AAI infrastruktury. S narůstajícími nároky na tuto komponentu a také s rozšiřujícím se počtem projektů, ve kterých je provozována, se dosavadní řešení manuální správy jednotlivých instancí začalo jevit jako nedostatečné a neefektivní.

Cílem této bakalářské práce je analýza autentizační proxy a dalších relevantních komponent ekosystému Perun použitých pro řízení přístupu, a dále pak návrh vhodné architektury prostředí s důrazem na posílení automatizace provozu. Navržené řešení bude následně implementováno pomocí vhodného nástroje. Zároveň je však třeba zachovat modulárnost a přizpůsobitelnost celého řešení, aby jej bylo možné integrovat do různých prostředí.

Text práce je rozdělen celkem do sedmi kapitol. Druhá kapitola se zaměřuje na popis provozu AAI v akademickém prostředí. Třetí kapitola se zabývá analýzou celého ekosystému Perun a jeho jednotlivých komponent. Ve čtvrté kapitole je popsána dosavadní architektura prostředí autentizační proxy a její jednotlivé komponenty. Tato kapitola se také zaměřuje na zajištění vysoké dostupnosti provozu této komponenty. Návrh nové architektury prostředí autentizační proxy je poté popsán v rámci páte kapitoly, přičemž implementací těchto změn se zabývá šestá kapitola.

[1] Sdružení vysokých škol a Akademie věd České republiky provozující národní e-infrastrukturu pro vědu, výzkum a vzdělávání. [1]

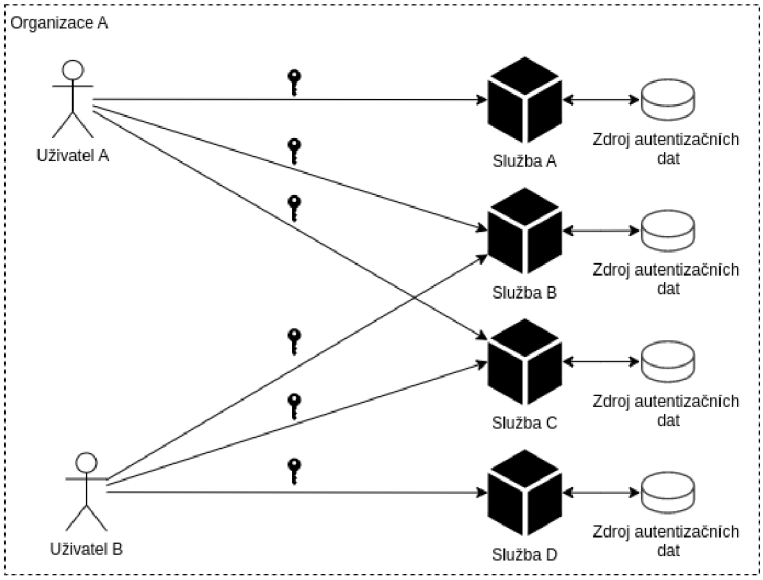
[2] Národní centrum provozující výpočetní a datovou infrastrukturu pro výzkum a vývoj. [2]

2 Popis prostředí

V rámci této kapitoly bude popsáno akademické prostředí se zaměřením na autentizační a autorizační infrastrukturu (AAI). Primárně zde budou vysvětleny role, které jednotlivé subjekty mohou zastávat. Dále zde budou popsány služby federace identit a federace služeb.

Akademické prostředí je složeno z velkého množství subjektů. Na jedné straně zde máme uživatele s formálním vztahem k organizaci (zaměstnanec, student, vědecký pracovník, registrovaný uživatel, atd.), a na druhé straně stojí služby a zdroje, které mají být uživatelům zpřístupněny. Tyto zdroje nejsou zpravidla přístupné volně a je nutné se před přístupem k tomuto zdroji autentizovat. Některé zdroje mohou vyžadovat další podmínky, které je nutné splnit před přístupem na službu. Tento proces je nazýván autorizace.

V rámci akademického prostředí se již zpravidla nevyužívá lokální autentizace (viz diagram 1), kdy má uživatel vytvořeny různé přístupové údaje ke každé službě/zdroji, ale využívá se konceptu tzv. poskytovatelů identit a poskytovatelů služeb za pomoci standardizovaných protokolů. Výsledkem aplikace tohoto přístupu vzniká tzv. federované prostředí.



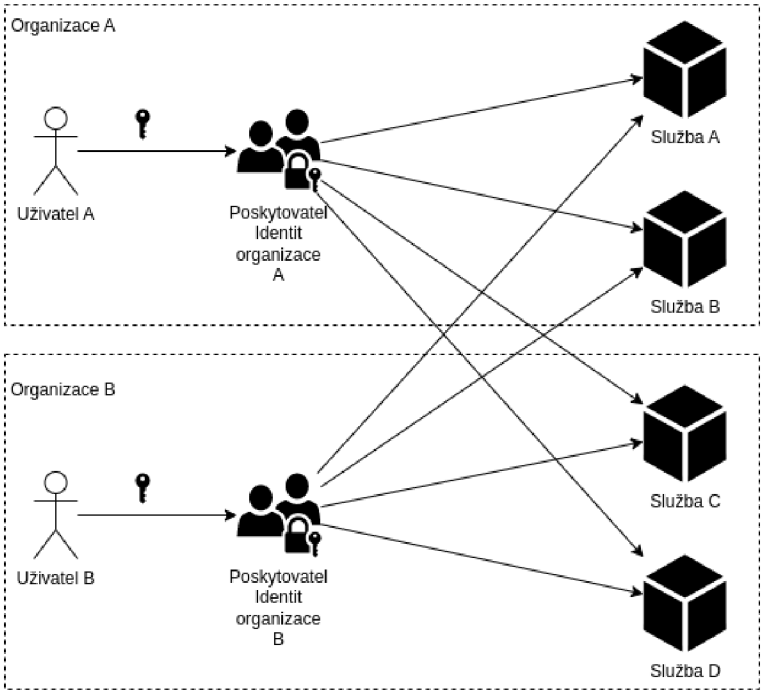
Obrázek 1: Diagram lokální autentizace k více zdrojům

Organizace, která spravuje účty svých uživatelů, může poskytovat službu tzv. poskytovatele identit[3]. Její uživatelé jsou vybaveni přístupovými údaji (nejčastěji uživatelské jméno a heslo), pomocí kterých ověřují svoji identitu vůči svému poskytovateli identit, a ten poté může žadeteli o autentizaci vydat informaci o úspěchu anebo neúspěchu této operace. Spolu s touto informací mohou být předány také některé další informace o uživateli. Organizace, ve které má uži-

[3]Označován jako Identity provider (IdP)

vatel takto spravovaný uživatelský účet, je poté nazývána domovská organizace uživatele, přičemž každý uživatel může mít i více domovských organizací.

Organizace poskytující služby, které mají být zpřístupněny uživatelům zastávají roli tzv. poskytovatelů služeb[4]. Tyto entity poté nezajišťují autentizaci uživatele přímo, nicméně tento požadavek delegují na poskytovatele identit. Na základě dat získaných od poskytovatele identit následně mohou vyhodnotit, zda se uživatel úspěšně autentizoval a také zda je oprávněn k danému zdroji přistoupit.



Obrázek 2: Diagram autentizace uživatelů ke zdrojům s užitím konceptu poskytovatele identit a poskytovatele služeb

Tento systém poskytovatelů identit a služeb má mnohé výhody pro všechny zmíněné subjekty. Pro správce služeb odpadá nutnost zřizovat a spravovat přístupové údaje jednotlivých uživatelů. Uživatelé nejsou nuceni vytvářet si pro každou jednotlivou službu nové přístupové údaje a ke všem službám je mu umožněn přístup pomocí jedněch přístupových údajů z jeho domovské organizace. Uživatel dále může čerpat výhodu z tzv. Single Sign-On[3] přihlášení (SSO přihlášení), kdy je mu umožněno přistoupit k více službám na základě jednoho ověření přístupových údajů u jeho poskytovatele identit. Současně jde také o bezpečnější řešení, jelikož jedno centrální místo pro správu a ověřování přístupových údajů je zpravidla lépe zabezpečeno, než kdyby byly přístupové údaje spravovány pro každou službu samostatně.

Současně však tento systém přináší i nové aspekty, jež mohou způsobit pro-

[4]Označován jako Service provider (SP)

blémy. Prvním z nich je nutnost vzájemné výměny metadat[5] mezi jednotlivými poskytovateli identit a služeb. Poskytovatel identit totiž musí mít informace, které služby u něj mohou žádat o autentizaci, popřípadě jaké atributy (informace o uživateli) daný poskytovatel služby vyžaduje. Stejně tak poskytovatel služby musí mít seznam dostupných poskytovatelů identit, pomocí nichž se uživatel může vůči dané službě autentizovat.

Pro ulehčení sdílení těchto metadat se jednotlivé subjekty začaly seskupovat do tzv. federací, které umožnují jednoduché získání metadat všech subjektů z dané federace, ať už se jedná o poskytovatele identit nebo o poskytovatele služeb. Federace nejčastěji vznikají na úrovni jednotlivých států, nicméně mohou vznikat i v rámci různých uskupení nebo zájmových skupin. V rámci České republiky tak vznikla např. Česká akademická federace eduID.cz[6]. Jednotlivé národní federace mohou být dále zapojeny do tzv. interfederace eduGAIN[7].

Další aspekt představuje nutnost umožnit uživateli výběr domovské organizace při přístupu na službu za předpokladu, že služba umožňuje přístup skrze více poskytovatelů identit. Tento problém se nejčastěji řeší pomocí služby tzv. rozcestníku[8], který může být provozován jednotlivými službami, popřípadě i centrálně na úrovni federací.

3 Ekosystému Perun

Ekosystém Perun je v rámci několika organizací a infrastruktur využíván pro správu uživatelů, jejich identit a pro řízení přístupu ke zdrojům. Tato funkcionalita je zajištna pomocí dvou komponent, a to Identity management systému (dále jen IdM) a autentizační a autorizační proxy (dále jen AAI proxy).

3.1 IdM Perun

IdM systém Perun[4] zajišťuje správu uživatelů, jejich identit a atributů. Základní jednotkou systému je uživatel (uživatelský účet), který je digitální reprezentací fyzického uživatele v systému. Ke každému uživatelskému účtu je přiřazena jedna či více digitálních uživatelských identit. Ty představují vazbu daného fyzického uživatele a některého z poskytovatelů identit. Na úrovni datového modelu je tato vazba reprezentována jako kombinace identifikátoru poskytovatele identity a identifikátoru uživatele v rámci organizace poskytovatele identit. Každý uživatel může mít přiřazen různý počet digitálních uživatelských identit v závislosti na tom, kolik různých uživatelských účtů má u různých poskytovatelů identit. Ke každé digitální uživatelské identitě mohou být přiřazeny další atributy, které slouží pro uchování dat získaných o uživateli od poskytovatele dané uživatelské identity.

[5] Technické informace potřebné pro propojení poskytovatele identit a poskytovatele služby.
[6] https://www.eduid.cz/cs/index
[7] https://edugain.org
[8] Discovery service/WAYF - Where Are You From

Uživatelé mohou být v systému seskupováni do tzv. virtuálních organizací[9]. V rámci jedné virtuální organizace mohou být uživatelé dále členěni do skupin, případně podskupin. Tyto skupiny mohou být následně použity např. pro autorizaci přístupu ke koncovým službám (viz kapitola 3.4).

IdM systém umožňuje k jednotlivým objektům v systému ukládat dodatečné informace, a to za pomoci tzv. atributů. Tento způsob ukládání dat pomocí atributů byl navržen v diplomové práci s názvem Správa atributů systému Perun[5]. Jednotlivé atributy jsou předem definovány atributovou definicí, která se skládá především z názvu atributu, jeho popisu a z definice datového typu. Součástí definice může být vazba na tzv. atributový modul. Ten může sloužit ke kontrole hodnoty atributu, popřípadě definovat pravidla pro získání hodnoty atributu na základě atributu jiného. V závislosti na konkrétní atributové definici může být hodnota atributu uložena přímo v objektu daného atributu, popřípadě získána výpočtem dle předem stanovené funkce.

3.2 AAI proxy

AAI proxy tvoří integrační prvek mezi poskytovateli identit a poskytovateli služeb. Jejím primárním úkolem je identifikovat a autentizovat uživatele za pomoci jeho poskytovatele identit, a následně uživateli umožnit přístup k chráněnému zdroji, tedy službě. Kromě pouhé autentizace uživatele zajišťuje tato komponenta také předání údajů o uživateli službě. Tyto údaje jsou zpravidla získány od poskytovatele identit a mohou být doplněny nebo nahrazeny daty získanými z jiných zdrojů. Další funkcionalitou, kterou AAI proxy poskytuje je autorizace, tedy rozhodnutí, zda je uživatel oprávněn ke koncové službě přistoupit či nikoli, a to na základě předem specifikovaných pravidel.

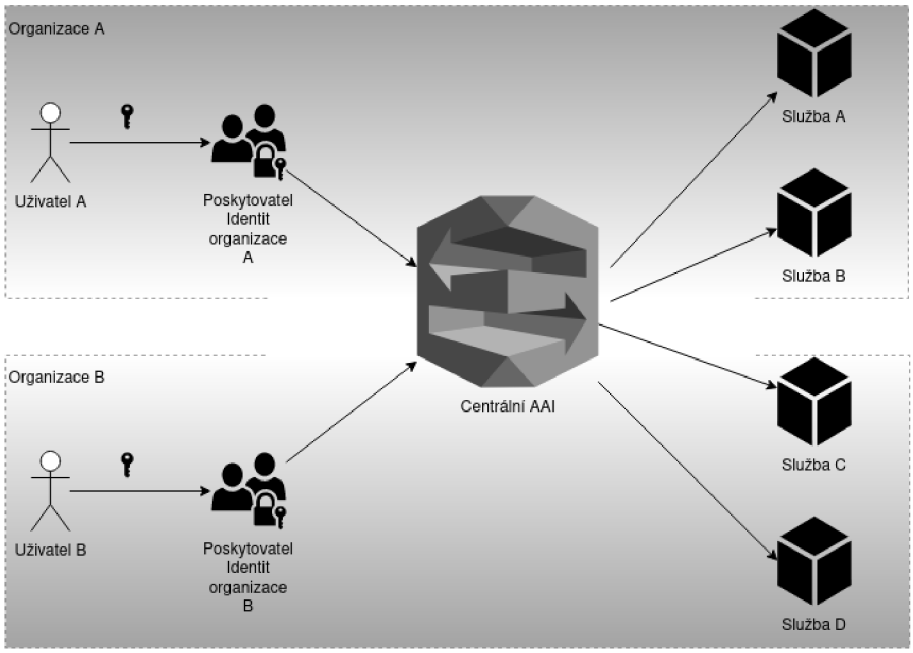
Tato centrální AAI proxy je v souladu se specifikací AARC Blueprint Architecture (AARC BPA)[10], která je doporučovaným metodickým pokynem pro zajištění kompatibility nezávislých AAI v rámci Evropského Open Science Cloudu (EOSC) a dalších akademických spoluprací.

AARC BPA obsahuje pět sekcí reprezentujících tematické seskupení komponent v infrastruktuře: Identitu uživatele, Atributové služby, Překladové služby přístupových protokolů, Autorizaci, a Koncové služby.[6] Propojení jednotlivých vrstev je znázorněno diagramem 4.

- Identita uživatele - obsahuje služby pro identifikaci a autentizaci uživatele
- Atributové služby - obsahuje komponenty zabývající se správou a poskytováním informací o uživateli
- Překladové služby přístupových protokolů - řeší požadavek na podporu více autentizačních technologií a vzájemný překlad mezi protokoly

[9] Virtuální organizace nemusí mít právní základ a tudíž mohou sdružovat uživatele z různých organizací, přičemž členy virtuální organizace mohou být i uživatelé, kteří nejsou členy žádné reálné organizace.

[10] https://aarc-project.eu/architecture



Obrázek 3: Zjednodušené schéma propojení komponent v rámci AAI proxy

- Autorizace - obsahuje komponenty pro řízení přístupu k prvkům sekce koncových služeb
- Koncové služby - obsahuje zdroje (služby), ke kterým chtějí uživatelé přistupovat

Sekce identity uživatele zahrnuje všechny služby primárně určené pro identifikaci a autentizaci uživatele. V rámci výzkumného a akademického prostředí se jedná o jednotlivé poskytovatele identit využívajících výhradně autentizačních protokolů SAML 2.0[11], OAuth 2.0[12] a OpenID Connect[13]. Během procesu autentizace uživatele dochází k vydání informací o uživateli poskytovatelem identit. V rámci proxy AAI tato část AARC BPA není příliš zajímavá, jelikož tyto služby standardně dodávají jednotlivé organizace spravující své uživatele a proxy AAI jen zajišťuje jejich integraci pomocí sekce překladových služeb.

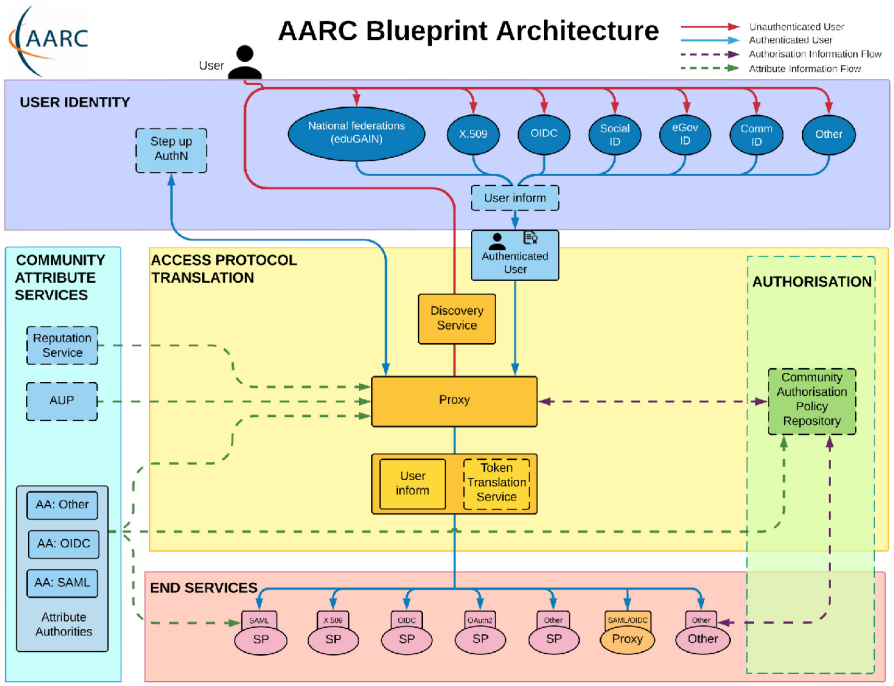
Oblast atributových služeb obsahuje ty komponenty, které jsou primárně určeny pro správu a poskytování informací o uživateli. Standardně jde o informace nad rámec dat získaných během autentizace od poskytovatele identit. Do této oblasti architektury spadá zejména tzv. atributová autorita[14], kterou v rámci proxy

[11]SAML 2.0 (Security Assertion Markup Language) je otevřený standard vytvořený za účelem poskytování SSO přihlášení mezi více doménami; https://saml.xml.org/saml-specifications

[12]Moderní autorizační protokol; https://oauth.net/2/

[13]OpenID Connect (označován také jako OIDC) je rozšíření autorizačního protokolu OAuth 2.0 o autentizaci a o rozhraní pro získávání informací o uživateli; https://openid.net/developers/how-connect-works/

[14]Technická jednotka řízená organizací, která je oprávněna činit určitá prohlášení o enti-



Obrázek 4: AARC Blueprint Architecture (AARC-BPA-2019)[6]

AAI zajišťuje IdM Perun. Ten mimo jiné slouží k centrální správě uživatelů v infrastruktuře, jejich skupin a také k evidenci koncových služeb. V rámci systému Perun jsou také generovány, popřípadě dopočítávány některé údaje o uživateli (např. jednoznačný identifikátor uživatele v rámci e-infrastruktury).

Sekce překladových služeb přístupových protokolů zahrnuje komponenty potřebné pro podporu více autentizačních technologií. Patří sem zejména integrační prvek mezi poskytovateli identit a poskytovateli služeb (tzv. SP-IdP-Proxy), služby zajišťující překlad autentizačních dat mezi různými technologiemi, rozcestník pro výběr ověřujícího poskytovatele identit uživatele a také služba pro informování uživatele o zpracování jeho osobních údajů. Služby spadající do této oblasti jsou základními komponentami proxy AAI.

Služby zajišťující řízení přístupu ke koncovým službám spadají pod oblast autorizace. AARC BPA umožňuje část řízení přístupu ke koncovým službám delegovat na centrální komponenty a tím snížit složitost správy autorizačních politik a jejich vyhodnocování pro jednotlivé služby. Správa autorizačních politik je v rámci proxy AAI zajištěna IdM systémem Perun, přičemž proxy AAI je zodpovědná za vyhodnocování daných autorizačních politik pro jednotlivé služby.

Sekce koncových služeb sdružuje všechny poskytovatele služeb, ke kterým mohou uživatele přistupovat a k nimž má být umožněn přístup pouze autentizovaným uživatelům. Může se jednat jak o služby webové, tak i o služby newebové

tách a přiřazovat jim atributy. Atributová autorita může být autoritativním zdrojem těchto prohlášení, nebo může jít o prohlášení získaná z jiných zdrojů. [7]

(např. autentizace k aplikačnímu rozhraní). Koncové služby jsou dodávány poskytovateli těchto služeb a proxy AAI zajišťujě primárně jejich integraci.

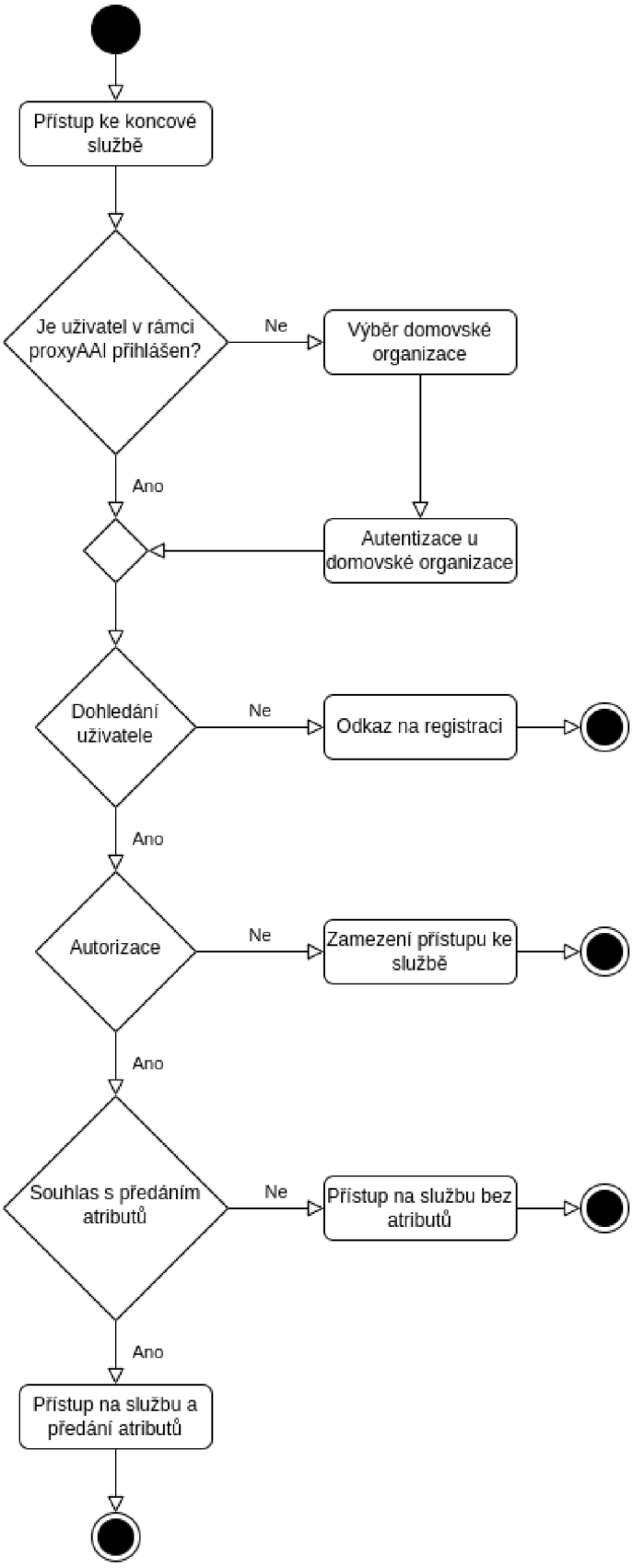
Jednotlivé služby z výše uvedených seskupení částí infrastruktury mohou být poskytovány jak různými komponentami, tak mohou být provozovány různými organizacemi. Kompatibilita je zajištěna jasnou specifikací, jaké služby mají být dodávány v rámci jaké sekce při zajištění kompatibility s dalšími částmi infrastruktury. Je tedy možné nahradit služby jedné sekce jiným technickým řešením, které je v souladu s AARC BPA, při zachování stejné funkcionality.

3.3 Proces přihlášení uživatele pomocí proxy AAI

Při přístupu uživatele ke službě dojde k odeslání autentizačního požadavku komponentě proxy AAI. Ta vyhodnotí, zda je již uživatel v rámci AAI proxy přihlášen či nikoli. Pokud není, je uživatel odkázán na rozcestník, kde mu je umožněno dohledat svoji domovskou organizaci. Proces pokračuje odesláním autentizačního požadavku poskytovateli identit zvolené organizace. Uživatel se autentizuje u zvoleného poskytovatele identit a tato informace spolu s informacemi o uživateli je v rámci odpovědi poskytnuta AAI proxy. Na základě přijatých dat je poté uživatel identifikován v IdM Perun, kde mohou být současně data o uživateli aktualizována na základě přijatých dat z poskytovatele identit, zejména informace vztahující se k použité digitální identitě. Proces pokračuje vyhodnocením, zda uživatel splňuje všechny podmínky pro přístup ke koncové službě definované v IdM systému, tedy autorizace. Splňuje-li uživatel tyto podmínky, je následně přesměrován jako autentizovaný na koncovou službu. Tato kromě pouhé informace o provedené autentizaci může získat také další dodatečné informace o uživateli, a to jak získané od poskytovatele identit, tak i z centrální evidence uživatelů. Toto vydání atributů může být podmíněno souhlasem uživatele a je technicky realizováno v závislosti na použitém integračním protokolu.

Podmínky pro přístup uživatelů ke službě jsou různé. Některé služby umožňují přístup všem autentizovaným uživatelům a proces autorizace na straně proxy AAI se nerealizuje (samotná autorizace může být zajištěna na straně koncové služby), jiné umožňují přístup vybraným skupinám uživatelů, popřípadě může být zvoleno jiné autorizační pravidlo. Nesplňuje-li uživatel daná autorizační pravidla koncové služby, je v procesu přihlášení zastaven a odkázán na stránku informující o nesplnění všech kritérií nutných pro přístup ke koncové službě.

Jestliže uživatel není v IdM systému dohledán, je nutné, aby se do centrální evidence uživatelů nejprve zaregistroval. V takovém případě dojde k přesměrování uživatele na registrační komponentu IdM systému spolu s údaji o daném uživateli (atributy získané od poskytovatele identit). Tato registrace je nutná, jelikož během ní dojde k zavedení uživatele do IdM systému a vygenerování identifikátorů pro jeho uživatelský účet. V rámci této registrace je možné od uživatele vyžádat doplnění některých údajů, ať už z důvodu nepřeposlání těchto dat od poskytovatele identit, nebo z důvodu nutnosti vyplnit některé údaje vyžadované provozovatelem infrastruktury nad rámec základních informací.



Obrázek 5: Diagram procesu přístupu uživatele ke službě

Přistupuje-li ke koncové službě uživatel, který je již v rámci proxy AAI přihlášen, dojde k přeskočení výběru domovské organizace uživatele a proces pokračuje vyhodnocením autorizace uživatele ke koncové službě. Proces přihlášení uživatele je znázorněn na obrázku 5.

3.4 Funkcionalita proxy AAI

Jak bylo popsáno, Proxy AAI tvoří integrační prvek mezi poskytovateli identit a služeb. Navíc zajišťuje některé funkcionality, které by v případě přímého propojení mezi poskytovateli identit a poskytovateli služeb musela zajistit sama služba, popřípadě by nebyly dostupné.

Mezi první poskytované funkcionality můžeme zařadit samotnou **integraci jednotlivých poskytovatelů identit** a také poskytování **rozcestníku**. Bez použití proxy AAI se služba musí integrovat spolu s každým poskytovatelem identit samostatně, popřípadě může využít integraci celé federace. Současně je také úlohou služby umožňit uživateli volbu své domovské organizace, pomocí které se bude k dané službě autentizovat. V případě proxy AAI je tento krok pro službu výrazně jednodušší, jelikož služba samotná se musí integrovat pouze s jedním poskytovatelem identit, a to právě s proxy AAI. Integrace jednotlivých poskytovatelů identit poté probíhá v rámci centrální komponenty, stejně jako zajištění dostupnosti rozcestníku. Tento může být provozován jako součást proxy AAI, a nebo může být použito i některé z již dostupných externích rešení (např. v rámci jednotlivých federací).

Jak již bylo zmíněno výše, velká část poskytovatelů identit je založena na autentizačním protokolu SAML 2.0, menší část poté na protokolu OpenID Connect. V rámci obou protokolů je však možné integrovat pouze koncové služby, které využívají shodný autentizační protokol jako poskytovatel identity, a jen malá část poskytovalů identit umožňuje integrovat služby z obou protokolů. V takovém případě je třeba mezi poskytovatele identity a koncovou službu vložit komponentu, která zajistí překlad mezi autentizačními protokoly. Touto komponentou může být právě proxy AAI, která takto umožňuje **odstínění technické implementace poskytovatelů identit a služeb**.

Integrace koncových služeb k jednotnému AAI má nesporné výhody především pro uživatele, kterým je takto zajištěno **jednotné workflow pro přístup ke službám** a také je jim umožněno využívat již zmiňované Single Sign-on přihlašování. Ke všem integrovaným službám se uživatelé přihlašují přes jednu centrální komponentu, proxy AAI, která ve všech případech působí vizuálně na uživatele jednotně.

Po úspěšné autentizaci uživatele koncová služba obdrží informace o tom, zda autentizace byla úspěšná spolu s množinou atributů, které o uživateli vydal poskytovatel dané identity. Tyto vydávané množiny atributů se však mohou lišit na základě zvoleného poskytovatele identit, a je dále na službě se s tímto problémem vypořádat. Výborným příkladem jsou atributy pro jméno uživatele. Existuje více standardizovaných atributů pro předávání jména uživatele, které

se liší tím, zda je vydáváno celé jméno, zda obsahuje i tituly, nebo zda je rozděleno na části (křestní jméno, prostřední jméno a příjmení). Proxy AAI zajišťuje tzv. **harmonizaci atributů**, tedy zpracování atributů získaných od jednotlivých poskytovatelů identit a následné vydání jednotné harmonizované sady atributů poskytovateli služby.

Proxy AAI nevydává koncové službě přímo data získaná od poskytovatele identit, ale využívá data uložená v IdM systému Perun, přičemž samotné vydání atributů službě může být podmíněno souhlasem uživatele. Každý uživatel tedy musí mít v systému Perun zaveden uživatelský účet. Ten představuje digitální reprezentaci uživatele, tedy osoby samotné. V rámci uživatelského účtu má uživatel připojenu minimálně jednu uživatelskou digitální identitu. Digitální identita jednoznačně identifikuje uživatele v rámci některého z poskytovatelů identit a jde o kombinaci identifikátoru poskytovatele identity a identifikátoru uživatele. Systém Perun spolu s proxy AAI umožňují uživateli tzv. **konsolidaci identit**, tedy přidání vícero digitálních identit k jednomu uživatelskému účtu. Uživatelský účet může v systému vznikat více způsoby. Prvním z nich je samotná registrace uživatele do systému, druhým poté synchronizace uživatelů z externích systémů (např. personální systém organizace). K uživatelskému účtu jsou dále vázány atributy, které byly do systému zadány (např. ověřený email uživatele získaný při registraci), popřípadě mohou být systémem Perun vypočteny (např. jednoznačný identifikátor uživatele vygenerovaný při vytváření uživatele, seznam všech identifikátorů uživatele, apod.).

Proxy AAI během každého přihlášení provádí aktualizaci dat dané digitální identity na základě právě přijatých dat od poskytovatele identit. Součástí této aktualizace dat je také uložení informace o použití dané digitální identity, a to celé za účelem práce s co nejaktuálnějšími daty o uživateli. Tato aktualizace může vynutit nový výpočet některých atributů uživatele, které jsou vypočteny systémem Perun a jsou závislé na atributech digitálních identit. Kromě samotné evidence uživatelů a jejich atributů umožňuje systém Perun již zmíněné seskupování uživatelů do hierarchického stromu skupin a podskupin. Členství uživatele v těchto skupinách může být také vydáváno koncovým službám jako jeden z atributů uživatele.

Další funkcionalitou, kterou poskytuje proxy AAI je **autorizace**. Je-li nutné před přístupem na službu uživatele nejdříve autorizovat, je v případě prostředí bez proxy AAI tento proces plně v režii služby samotné. Ta autorizaci provádí na základě dat přijatých od poskytovale identit, a to buď přímo na základě získaných dat (vhodné např. pro služby umožňující přístup osobám s jistým vztahem k organizace - např. pro studenty či zaměstnance), popřípadě identifikací uživatele ve vlastní databázi uživatelů, kteří mají mít k dané službě přístup. V případě integrace služby k proxy AAI může být tento způsob autorizace uživatelů na základě přijatých atributů rovněž používán, nicméně je možné autorizaci přenechat i na proxy AAI samotné. V takovém případě jsou předem nadefinována autorizační pravidla, na základě kterých proxy AAI rozhoduje, zda uživatele na danou službu pustí či nikoli. Jednou z možností definice autorizačních pravidel je kon-

trola členství ve skupinách. V případě, kdy uživatel daná pravidla splňuje, je přesměrován zpět na službu spolu s požadovanými atributy. V opačném případě je uživateli zobrazena stránka s informací o zamezení přístupu, která může být obohacena o informace, na koho se obrátit pro získání přístupu.

Jelikož je proxy AAI integračním prvkem mezi poskytovateli identit a služeb, je také vhodným místem pro sběr **auditovacích dat** a **statistik**. Všechna přihlášení totiž musí být provedena právě přes proxy AAI. Během každého přihlášení jsou zaevidována data o autentizaci uživatele, která mohou být použita pro oba zmíněné účely. V případě, kdy je proxy AAI pověřena také autorizací uživatele je součástí evidovaných dat také informace, zda byl uživatel k dané koncové službě autorizován či nikoli, popřípadě na základě jakých pravidel autorizován nebyl. Provádí-li koncová služba autorizaci sama, je třeba tato data evidovat i na straně koncové služby. Sběr auditovacích logů centrálně na jednom místě umožňuje efektivněji dotrasovat akce uživatele např. v případě zneužití přístupových údajů. Využití centrální komponenty také umožňuje blokaci uživatele v rámci IdM systému Perun, kdy uživateli bude zamezeno přistupovat ke všem službám integrovaným k proxy AAI. Dalším využitím může být také sběr souhlasů uživatelů s podmínkami infrastruktury nebo služby.

3.5 Nové problémy

Využívání proxy AAI má mnohé výhody, avšak přináší také nové problémy a nedostatky. Ty vychází především ze skutečnosti, že se jedná o centrální komponentu, která zajišťuje veškerou autentizaci, a často také autorizaci uživatelů ke všem koncovým službám v rámci dané infrastruktury.

Největší problém představuje samotná **kritičnost centrální komponenty proxy AAI**. Jde o jeden z nejvíce kritických systémů v celé infrastruktuře, jelikož výpadek proxy AAI způsobí nemožnost autentizace a autorizace uživatelů ke koncovým službám. Tento problém bývá obecně označován jako *Single point of failure*, kdy výpadek jedné komponenty způsobí výpadek, popřípadě nedostupnost, mnoha dalších komponent. Z tohoto důvodu je tedy třeba zajistit maximální dostupnost dané komponenty, viz kapitola 4.1.

Další problém může představovat požadavek na **uživatelskou zkušenost**[15] a to především v rámci procesů autentizace a autorizace uživatele. Problém opět vychází ze skutečnosti, že proxy AAI je centrální komponenta mezi poskytovateli identit a poskytovateli služeb, která nejen v rámci procesu autentizace a autorizace uživatele vyžaduje interakci uživatele (např. výběr domovské organizace, souhlas s vydáním atributů uživatele službě, sběr souhlasu s novou verzí podmínek infrastruktury, zobrazení chybových stránek v případě přístupu uživatele nesplňujícího podmínky služby, atd.). Nejen v rámci těchto vyjmenovaných procesů je třeba klást důraz na jednoznačnost, jednoduchost a srozumitelnost

[15] Uživatelská zkušenost (angl. *User Experience* - UX) se zabývá analýzou, návrhy, testováním a optimalizací zkušeností uživatele se službou, či produktem s cílem zvyšovat pozitivní zkušenost uživatele.

jednotlivých procesů pro uživatele, aby se v jejich rámci neztrácel a zároveň, aby prolínání procesů působilo co nejméně rušivě. Tento důraz je kladen nejen na jednotlivé procesy, ale také na správný návrh zobrazovaných rozhraní včetně vhodných a konzistentních formulací zobrazovaných textů.

Některým službám může způsobovat potíže také skutečnost, že proxy AAI vydává jiné hodnoty atributů, než jaké by obdržely přímo od poskytovatele identit. Například, proxy AAI vydává službám generované identifikátory uživatele. To však činí potíže převážně službám, které byly do prostředí proxy AAI integrovány již po nějaké době provozu s přímým připojením k poskytovatelům identit a současně používající identifikátor uživatele pro persistentní uložení dat na straně služby. Tento problém však lze relativně snadno řešit více způsoby. Prvním z nich je jednorázové přemapování na straně služby, kdy správce služby provede manuální změnu identifikátorů uživatelů z hodnot získaných od poskytovatele služby na hodnoty získané z proxy AAI. Tento proces jednorázové migrace probíhá v rámci integrace služby k proxy AAI a zpravidla nevyžaduje další změny na straně služby. V některých případech však tento proces není možné použít, například pokud někteří uživatelé nejsou zavedeni v IdM systému Perun, nebo v případě kdy by změna na straně služby byla moc nákladná nebo nemožná. V takovém případě je možné migraci uživatelských identifikátorů provádět průběžně. Proxy AAI pro takové služby vydává dodatečné atributy uživatele, mezi které patří také seznam všech identifikátorů uživatele. Po přihlášení uživatele může služba na základě tohoto seznamu uživatele dohledat ve své databázi a zpřístupnit mu tak jeho předchozí data, a také provést aktualizaci identifikátoru na hodnotu identifikátoru uživatele v infrastruktuře. V rámci této průběžné migrace mohou nastat situace, na které musí být služba připravena správně zareagovat. Patří mezi ně například situace, kdy služba nedokáže uživatele dle seznamu identifikátorů ve svých datech dohledat, popřípadě kdy díky konsolidaci více identit k jednomu uživatelskému účtu v rámci infrastruktury služba ve svých datech dohledá více záznamů. První situaci je možné vyřešit na straně uživatele konsolidací identity, pomocí které dříve přistupoval ke koncové službě ke svému uživatelskému účtu. Druhá situace je komplikovanější a vyžaduje manuální zásah na straně koncové služby, kde je nutné zajistit úpravu dat takovým způsobem, aby byl na základě přijatých dat uživatele dohledán jen jeden záznam.

Za další nedostatek můžeme považovat samotnou skutečnost, že proxy AAI vystupuje vůči koncovým službám v roli poskytovatele identit. Ten zajišťuje autentizaci a autorizaci uživatele spolu s vydáním atributů uživatele koncové službě ve chvíli, kdy uživatel ke koncové službě přistupuje. Na základě předaných atributů uživatele mohou být na straně služby uchovávána uživatelská data, a v některých případech mohou služby uživateli poskytovat i jiná rozhraní chráněná jinými způsoby (např. pomocí aplikačních tokenů). U takových služeb je obvykle potřeba zajistit, aby v případě, kdy uživatel ztratí nárok přistupovat k dané koncové službě, popřípadě když jsou mu odebrána některá oprávnění, byla služba o těchto změnách informována a mohla na ně adekvátně zareagovat (např. revokací aplikačních tokenů). Tento proces bývá označován jako deprovisioning dat.

Jelikož však proxy AAI vydává data o uživateli pouze během přístupu uživatele ke službě, nedokáže tyto informace o změnách službě poskytnout bez interakce uživatele, popřípadě je poskytnout včas. Těmto službám je tedy třeba tyto infomace o změnách předávat jiným způsobem, například pomocí pravidelné propagace dat z IdM systému Perun. Ta může zahrnovat seznam uživatelů obohacený o uživatelské atributy, ale také např. oprávnění jednotlivých uživatelů v dané službě, jejich role, atd.

4 Stávající architektura proxy AAI

Ekosystém Perun je včetně autentizační a autorizační proxy využíván v řadě organizací a projektů. Komponenta proxy AAI je však v rámci nasazení upravována tak, aby co nejvhodněji plnila požadavky dané infrastruktury. Samotná komponenta není sestavena jako monolitická aplikace, nýbrž se jedná o soustavu aplikací a dalších komponent, které společně poskytují služby proxy AAI. Následující kapitola se bude zabývat nejprve obecnou architekturou proxy AAI obsahující základní prvky pro poskytování funkcionality. Další podkapitoly se poté zaměří na zajištění vysoké dostupnosti a také na jednotlivé komponenty proxy AAI z technického hlediska. Specifika jednotlivých infrastruktur, ve kterých je proxy AAI nasazeno nebudou v rámci práce popsána.

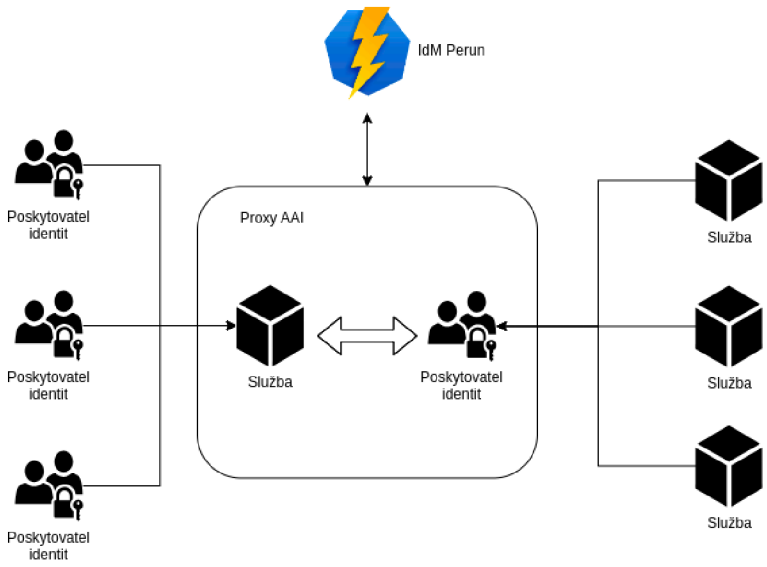
Jak již bylo zmíněno výše, proxy AAI je integrační prvek mezi poskytovateli identit a služeb. Z obecného pohledu je možné proxy AAI vnímat jako komponentu, která současně zastává roli poskytovatele identit i poskytovatele služeb. Pro poskytovatele identit představuje proxy AAI jednoho z poskytovatelů služeb, a naopak pro poskytovatele služeb představuje jednoho z poskytovatelů identit.

Jednotliví poskytovatelé identit i služeb mohou využívat různých autentizačních protokolů. Proxy AAI umožňuje integrovat poskytovatele identit pomocí autentizačních protokolů SAML 2.0, OAuth 2.0 a OpenID Connect. To umožňuje k proxy AAI integrovat velkou řadu poskytovatelů identit včetně využívání identit od poskytovatelů sociálních sítí. V některých případech je žádoucí, aby uživatelé měli přiděleny také autentizační údaje spravované na úrovni infrastruktury, které mohou být uchovávány např. v systémech poskytující rozhraní pomocí protokolů Kerberos[16] nebo LDAP[17]. Z pohledu obecné architektury se však jedná o další poskytovatele identit. Pro koncové služby proxy AAI představuje poskytovatele identit, který umožňuje integrovat koncové služby pomocí protokolů SAML 2.0, OAuth 2.0 a OpenID Connect.

Jelikož je k proxy AAI zpravidla integrováno více poskytovatelů identit, je v rámci infrastruktury potřeba také komponenta rozcestníku, který během procesu autentizace uživateli zobrazí seznam všech dostupných poskytovatelů identit. Z nich si uživatel zvolí jednoho, který následně provede samotnou autentizaci

[16] Síťový autentizační protokol; http://web.mit.edu/kerberos/

[17] The Lightweight Directory Access Protocol - protokol pro ukládání a přístup k datům na adresářovém serveru [8]



Obrázek 6: Zjednodušená architektura proxy AAI

uživatele. Tako komponenta nemusí být nutně součástí proxy AAI, ale může být využito i externí řešení nabízené např. v rámci federací identit.

Proxy AAI je úzce spojena s IdM systémem Perun, který slouží jako autoritativní zdroj dat o uživatelích a jejich členství v jednotlivých skupinách, popřípadě také jako zdroj autorizačních pravidel pro řízení přístupu. Z pohledu zjednodušené architektury se jedná o externí komponentu. Tato zjednodušená architektura je znázorněna na obrázku 6. V následující podkapitolách bude popsáno zajištění vysoké dostupnosti, dále budou rozebrány jednotlivé komponenty proxy AAI.

4.1 Vysoká dostupnost

Z popisu výše je patrné, že proxy AAI je velmi kritickou službou v rámci celé infrastruktury. Je tedy žádoucí zajistit, že její služby budou dostupné po maximální možnou dobu, v ideálním případě neustále, bez významných výpadků, snížení výkonu, nebo dalších dopadů na uživatele. Pro označení takových systémů se používá pojem vysoká dostupnost (angl. *High Availability* - HA).

Pro měření, popřípadě porovnávání, vysoké dostupnosti systémů a služeb existuje celá řada metrik, přičemž pravděpodobně nejpoužívanější je tzv. procento dostupnosti. To vyjadřuje poměr času, kdy byla služba dostupná k celkovému času běhu služby a udává se v procentech. Systémy poté zpravidla rozdělujeme do několika kategorií na základě roční procentuální dostupnosti, viz tabulka 1, přičemž běžné systémy jsou schopny dosahovat dostupnosti kolem 99 % a to bez dalších doplňujících technik pro zvýšení dostupnosti systému. Systémy, které dosahují roční procentuální dostupnosti přesahující 99,9 % bývají označovány jako vysoce dostupné.

Aby bylo možné měřit procentuální dostupnost systému, je nutné nejdříve zaspecifikovat co znamená, že je systém dostupný či nikoli. V případě komponenty

Tabulka 1: Tabulka procentuální dostupnosti systémů

Dostupnost [%]	Maximální doba výpadku po dobu 1 roku
90 %	36,53 dnů
99 %	3,65 dne
99,9 %	8,77 hodin
99,99 %	52,6 minut
99,999 %	5,26 minuty
99,9999 %	31,56 sekund

Tabulka 2: Tabulka procentuální dostupnosti jednotlivých instancí proxy AAI

Měsíc	login.bbmri-eric.eu	login.elixir-czech.org	login.ceitec.cz	login.cesnet.cz
03. 2021	99,895	99,895	99,998	99,899
04. 2021	99,986	100	99,998	100
05. 2021	99,998	99,992	99,993	100
06. 2021	99,990	99,713	99,983	99,992
07. 2021	100	99,937	99,179	100
08. 2021	99,978	99,912	99,990	100
09. 2021	99,974	99,969	99,937	100
10. 2021	99,980	99,912	99,965	99,993
11. 2021	99,994	99,982	99,982	99,976
12. 2021	99,956	99,707	99,955	100
01. 2022	99,976	99,988	99,909	99,989
02. 2022	99,385	99,369	99,372	99,971
Průměr	**99,925**	**99,847**	**99,855**	**99,985**

proxy AAI se jako ideální definice dostupnosti systému nabízí schopnost uživatelů přihlásit se ke koncovým službám, jelikož proces současně zahrňuje základní funkcionality této komponenty.

Pro měření dostupnosti komponenty proxy AAI je využíváno simulované přihlášení uživatele pomocí vlastního poskytovatele identit k testovací službě a měření, zda dané přihlášení bylo úspěšné či nikoli, popřípadě zda netrvalo příliš dlouho. Toto měření probíhá každou minutu pomocí monitorovacího systému. Na základě historických dat z monitorovacího systému bylo zjištěno, že v období 03. 2021 – 02. 2022 dosahovala proxy AAI dostupnosti průměrně téměř 99,91 %.[18] Přehled dostupnosti vybraných instancí v daném období je uveden v tabulce 2.

[18]Rozdíly v dostupnosti jednotlivých instancí jsou způsobeny převážně závislostí na jiném provozovateli virtualizační platformy a sítě.

4.1.1 Zajištění vysoké dostupnosti

Ve stávajících nasazení proxy AAI je tato komponenta provozována standardně v clusteru[19] se třemi uzly. Jeden z těchto uzlů je vždy aktivní a zajišťuje tak poskytování funkcionality, tedy autentizaci a autorizaci uživatelů ke službám, zbylé uzly jsou záložní pro případ výpadku aktivního uzlu. V případě, kdy aktivní uzel již není schopen poskytovat patřičnou funkcionalitu, je možné jej vyřadit z provozu, tedy označit jej jako neaktivní, a následně ze zbylých záložních uzlů zvolit nový aktivní uzel.

Způsob zajištění redundantního provozu a případný způsob volby, který z uzlů bude zodpovědný za poskytování funkcionality je silně závislý na dostupnosti podpory v rámci infrastruktury, ve které bude cluster provozován. Ve stávajících instancích byly využívány dva nástroje pro zajištění redundance proxy AAI, a to *Pacemaker*[20] a *ExaBGP*[21] v závislosti na prostředí provozu. Proxy AAI je vždy přístupná pomocí jedné veřejné IP adresy (a příslušného doménového jména), přičemž uzel poskytující funkcionalitu proxy AAI se může v čase měnit v závislosti na jeho dostupnosti. Zjednodušená architektura clusteru je znázorněna schématem 8 na konci této kapitoly.

V případě využívání nástroje *Pacemaker* je zmíněná IP adresa označována jako virtuální a v jeden časový okamžik je přidělena právě jednomu uzlu. V případě výpadku nebo manuálním zásahem správce dochází k přesunu této virtuální IP adresy mezi uzly. Využívání nástroje *ExaBGP* vyžaduje složitější technickou podporu v rámci prostředí, kde bude využíváno. Přináší však lepší možnosti při rozdělení jednotlivých uzlů mezi více datových center, které mohou být také geograficky odděleny. U nástroje *ExaBGP* nedochází k přidělování IP adresy jednotlivým uzlům, nýbrž k dynamické modifikaci směrovacích tabulek směrovačů v příslušné síti, které poté určují, na který uzel bude v danou chvíli provoz směrován. Také je umožněno mít současně více aktivních uzlů, nicméně je nutné mít také více veřejných IP adres pomocí kterých je daná instance proxy AAI dostupná uživatelům. Podrobný popis obou nástrojů a protokolů, které tyto nástroje využívají není předmětem této práce.

Redundantní provoz proxy AAI kromě vysoké dostupnosti také představuje značnou výhodu při práci s jednotlivými uzly (aktualizace komponent, instalace systémových záplat, atd.), kdy je možné tyto provozní zásahy uskutečnit bez dopadů na dostupnost služby. Aby však bylo možné provozovat proxy AAI redundantně, je nutné některé komponenty uzpůsobit pro takový provoz, popřípadě je nahradit za komponenty, které jsou pro redundantní provoz vhodnější.

[19] *Cluster* je označení pro množinu propojených počítačů nebo serverů (označovány jako *uzly*), které pracují společně jako jeden systém, ať už za účelem zvýšení výkonu, popřípadě dostupnosti a spolehlivosti systému.

[20] Software používaný pro zajištění vysoké dostupnosti systému. Slouží k řízení a monitorování clusteru a zajišťuje, že v případě výpadku nebo poruchy jednoho uzlu převezme jiný uzel jeho funkci; https://clusterlabs.org/pacemaker/

[21] Software pro řízení směrování v počítačových sítích, jehož hlavním účelem je umožnit konfiguraci a dynamické oznámení směrovacích informací mezi síťovými zařízeními; https://github.com/Exa-Networks/exabgp

Jde zejména o komponenty sloužící k uchování stavu a dat (jako jsou databáze, úložiště sezení, atd.), kdy je nutné zajistit konzistenci a dostupnost těchto dat napříč jednotlivými uzly.

Redundantní provoz má také dopady při řešení incidentů (např. v případě kdy se uživatel nemůže z důvodu chyby přihlásit ke službě), kdy je nutné se vypořádat se skutečností, že jednotlivé uzly standardně ukládají provozní logy odděleně, přičemž není zaručeno, který uzel je v danou dobu aktivní. Je tak případně nutné prohledat logy ze všech uzlů. S tímto problémem je možné se vypořádat např. využitím společného logovacího serveru, který může umožnit prohledání provozních logů ze všech instancí na jednom místě. Ten v závislosti na prostředí může být provozován pouze pro potřeby proxy AAI, případně je možné využít centrálního logovacího serveru, je-li v rámci infrastruktury dostupný.

Dalším opatřením pro zajištění vysoké dostupnosti v dosavadním řešení je minimalizace závislostí na externích komponentách, popřípadě minimalizace dopadů v případě výpadku některé z externích komponent. Největší závislost proxy AAI je na IdM systému Perun, který slouží jakožto zdroj informací o uživatelých, jejich identitách, a také jako zdroj autorizačních pravidel. Tato data jsou však nutná pro provoz Proxy AAI, a výpadek IdM systému Perun tedy vede k výpadku i proxy AAI. IdM systém Perun není v celém rozsahu uzpůsoben k provozu ve vysoké dostupnsoti a je tak třeba se s danou situací vhodné.

4.2 Komponenty proxy AAI

Jak již bylo zmíněno, v případě proxy AAI nejde o systém s monolitickou architekturou, který by zajišťoval všechnu funkcionalitu, nýbrž jde o kombinaci více komponent, které ji společně zajišťují. Jednotlivé komponenty bychom mohli rozdělit do kategorií na hlavní, podpůrné a doplňkové. Architektura jednoho uzlu je znázorněna na obrázku 7.

4.2.1 Hlavní komponenty

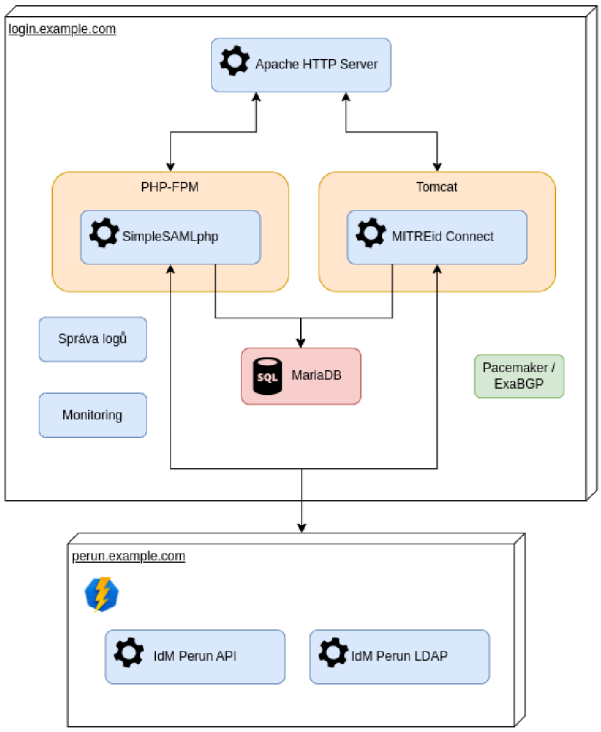
Hlavní komponenty přímo zajišťují požadovanou funkcionalitu. V našem případě se jedná o aplikace *SimpleSAMLphp*[22] a *MITREid Connect*[23], které vhodným propojením umožňují propojit jednotlivé poskytovatele identit a poskytovatele služeb a zajistit tak uživateli možnost se ke službě autentizovat, a případně také autorizovat.

4.2.1.1 SimpeSAMLphp

SimpleSAMLphp je open-source aplikace v jazyce PHP, která poskytuje nativní podporu protokolu SAML 2.0. Na základě konfigurace je možné aplikaci provozovat v režimu poskytovatele služby nebo poskytovatele identit, popřípadě v obou

[22] https://simplesamlphp.org

[23] https://github.com/mitreid-connect/OpenID-Connect-Java-Spring-Server/wiki



Obrázek 7: Schéma stávající architektury jednoho uzlu

režimech současně (tzv. SP-IdP Proxy) jako proxy AAI. Výhodou aplikace SimpleSAMLphp je její jednoduchost, široká konfigurovatelnost a také modulárnost. Aplikaci je možné upravit, popřípadě rozšířit pomocí vlastních modulů, které mohou jak měnit vizuální stránku aplikace, tak přinášet novou funkcionalitu.

V rámci komponenty proxy AAI zajišťuje nástroj SimpleSAMLphp napojení všech dostupných poskytovatelů identit a autentizaci uživatele (ať už pomocí externího poskytovatele identit nebo v rámci lokálního ověření přístupových údajů) pro všechny koncové služby. Pro koncové služby využívající protokolu SAML 2.0 dále poskytuje i atributové služby a autorizaci.

4.2.1.2 MITREid Connect

Jde o referenční implementaci OpenID Connect autorizačního serveru postavenou na frameworku Spring[24] v jazyce Java. Pro svůj provoz vyžaduje doplnění o databázi a webový aplikační server pro provozování Java aplikací. Ten je zajištěn pomocí nástroje Tomcat[25].

V rámci proxy AAI zajišťuje komponenta MITREid Connect autentizaci a autorizaci uživatelů pro koncové služby připojené pomocí protokolů OAuth 2.0

[24] Open-source framework pro usnadnění vývoje robustních a škálovatelných aplikací. Spring nabízí mnoho modulů a funkcionalit, které umožňují efektivní správu závislostí, řízení přístupu k datům, implementaci webových služeb a mnoho dalšího; https://spring.io

[25] https://tomcat.apache.org

a OpenID Connect, přičemž samotná autentizace uživatelů je delegována na komponentu SimpleSAMLphp (MITREid Connect server v danou chvíli vystupuje jako koncová služba pro komponentu SimpleSAMLphp).

4.2.2 Podpůrné komponenty

Mezi podpůrné komponenty můžeme zařadit všechny komponenty, které samy o sobě nezajišťují funkcionalitu proxy AAI, ale jsou nutné pro provoz hlavních komponent. V případě proxy AAI jde zejména o *webový server*, *databázi*, *komponentu rozcestníku* a také o *systém správy identit a skupin Perun*.

4.2.2.1 Webový server

Webový server slouží pro příjem a zpracování HTTP a HTTPS požadavků od klientů. V rámci stávajících instancí je využíván nástroj Apache HTTP Server[26]. Je zodpovědný za přesměrovávání HTTP/HTTPS požadavků na PHP-FPM[27] (požadavky směrující na SimpleSAMLphp), nebo na aplikační webový server (požadavky pro aplikaci MITREid Connect), a za poskytování dalšího obsahu pomocí zmíněných protokolů.

4.2.2.2 Databáze

Databáze je jednou z nejdůležitějších podpůrných komponent, jelikož slouží pro uchování persistentních dat jednotlivých komponent. Aby celá proxy AAI mohla svoji funkci zajišťovat s vysokou dostupností, je třeba, aby i databáze byla provozována s vysokou dostupností. V rámci proxy AAI je využívána databáze *MariaDB Galera Cluster*[28], která umožňuje provoz multi-master[29] databázového clusteru.

4.2.2.3 Rozcestník

Rozcestník je jednou z nutných komponent proxy AAI, jelikož umožnuje uživateli zvolit pomocí kterého poskytovatele identit bude autentizován. V rámci proxy AAI je využíváno především vestavěné řešení v komponentě SimpleSAMLphp, popřípadě je využíván rozcestník provozovaný v rámci České akademické federace identit eduID.cz[30].

[26] https://httpd.apache.org

[27] PHP-FPM je samostatný procesový manažer pro interpretaci a zpracování PHP skriptů založený na protokolu FastCGI;https://www.php.net/manual/en/install.fpm.php

[28] https://mariadb.com/kb/en/galera-cluster/

[29] Označení clusteru, který v jednu chvíli má více aktivních uzlů, přičemž všechny jsou schopny poskytovat požadovanou funkcionalitu.

[30] https://www.eduid.cz/cs/tech/wayf

4.2.2.4 IdM systém Perun

Základní funkce IdM systému Perun již byly popsány v kapitole 3.1. Jak již také bylo zmíněno, systém není přímo uzpůsoben pro provoz ve vysoké dostupnosti. Pro komunikaci se systémem je možné využít dvě rozhraní.

První z nich je RPC API rozhraní, které umožňuje práci se všemi objekty systému (čtení i zápis). Jde o primární rozhraní, které je určeno pro systémy, které požadují přístup ke všem objektům systému, případně nevyžadují jeho vyšší dostupnost. Druhým dostupným rozhraním je LDAP, které je určené pouze ke čtení a neobsahuje všechny objekty systému. Z uvedeného důvodu je využíváno především interními komponentami. V případě výpadku IdM systému Perun zůstává LDAP rozhraní zpravidla dostupné. LDAP rozhraní může být také provozováno redundantně.

Proxy AAI využívá primárně LDAP rozhraní pro čtení dat. Pro zápisové operace poté využívá RPC API rozhraní. Kombinací obou těchto rozhraní je možné zajistit základní provoz proxy AAI s vyšší dostupností. Všechny potřebné informace o již registrovaných uživatelích, jejich identitách i autorizačních pravidlech pro řízení přístupu ke koncovým službám jsou totiž dostupné právě v LDAP rozhraní. V případě výpadku IdM systému (tedy RPC API rozhraní) lze stále poskytovat základní služby proxy AAI. V danou chvíli však není možné provádět aktualizaci informací o uživatelých a jejich identitách, registrovat nové uživatele, popřípadě měnit autorizační pravidla pro přístup ke koncovým službám.

4.2.3 Doplňkové komponenty

Mezi doplňkové komponenty můžeme zařadit ostatní komponenty nebo aplikace, které nepatří mezi základní nebo podpůrné komponenty. Jedná se zejména o komponenty, které mohou přispívat k vyšší bezpečnosti či lepší operativě. V rámci proxy AAI do této kategorie můžeme zařadit systém pro správu logů nebo systém pro monitoring jednotlivých komponent.

Systém pro správu logů zajišťuje sběr a správu logů z jednotlivých aplikací. Jednotlivé logované zprávy se zpravidla rozdělí dle zdroje, tedy aplikace, která danou zprávu vytvořila. To umožňuje v případě potřeby pohodlnější vyhledávání a také umožňuje aplikovat různá pravidla pro uchování logovacích zpráv dle typů zpráv. Jednotlivé záznamy také mohou být odesílány na centrální logovací server. Zprávy z jednotlivých zdrojů jsou dále rotovány[31] a je na ně aplikována politika pro uchování logů. V rámci proxy AAI jsou využívány nástroje *rsyslog*[32] pro kategorizaci logů a *logrotate*[33] pro rotaci logů.

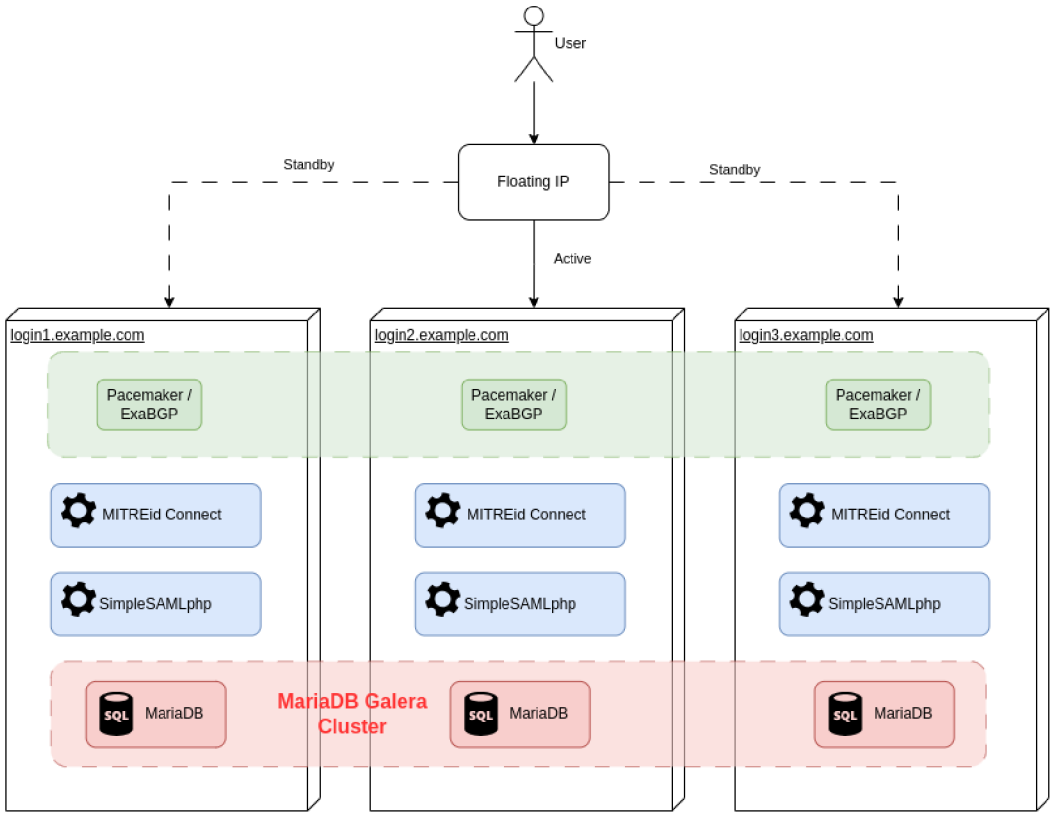
Monitoring jednotlivých komponent je jedním ze základů pro poskytování služeb ve vysoké dostupnosti. Jelikož se však nepodílí na poskytování základ-

[31] Proces, při kterém dochází k nahrazení starších záznamů novými. Rotace logů může probíhat na základě času (např. každý den), po dosažení určité velikosti souboru, popřípadě dle počtu záznamů.

[32] https://www.rsyslog.com

[33] https://github.com/logrotate/logrotate

ních funkcí proxy AAI, a jelikož bývá nejčastěji provozován centrálně v rámci infrastruktury, byl zařazen mezi doplňkové komponenty. Monitorovací systém je zodpovědný za kontrolu stavu jednotlivých uzlů a komponent, a případné notifikace jejich správcům. Mezi základní monitorované prvky patří například dostupnost síťového připojení, zbývající volné místo na disku, volná paměť RAM, atd. V případě proxy AAI je využíván monitorovací systém poskytovaný v rámci České Národní Gridové Infrastruktury Metacentrum, který je doplněn o vlastní monitorovací sondy (např. kontrola stavu jednotlivých komponent proxy AAI, kontrola dostupnosti přihlášení, atd.). Správně nastavený monitoring umožňuje předcházet velkému množství výpadků díky včasné detekci chybných stavů, popřípadě umožňuje zkrátit dobu při výpadku. Správci jsou o daných problémech neprodleně informováni a velké množství monitorovacích sond může usnadnit detekci, na jaké části infrastruktury došlo k problému. Je však důležité neprovádět pouze monitoring svých komponent, nicméně i všech externích komponent na kterých je proxy AAI závislá.



Obrázek 8: Zjednodušené schéma stávající architektury clusteru

5 Návrh správy konfigurace a nové architektury prostředí autentizační proxy

Jednotlivé instance proxy AAI byly doposud instalovány a spravovány manuálně. Tento způsob správy jednotlivých instancí však byl velmi náročný a neefektivní, jelikož požadované změny se vždy musely postupně aplikovat na jednotlivé uzly v clusteru. S ohledem na narůstající množství instancí a počet změn se stávala téměř neudržitelnou. Manuální zásahy, které bylo třeba postupně aplikovat na všechny uzly v clusteru, také zvyšovaly pravděpodobnost, že dojde k výpadku v důsledku chybné aplikace změn, popřípadě nekonzistence mezi uzly.

Tato kapitola se zabývá návrhem nahrazení manuální správy konfigurace autentizační proxy za pomoci některého z vhodných nástrojů pro automatizaci a správu konfigurace. Budou zde popsány zvolené technologie a dále také navržené změny v rámci architektury autentizační proxy.

5.1 Volba nástroje pro správu konfigurace

V dnešní době se pro automatizovanou správu infrastruktury používají nástroje tzv. *Infrastructure as Code (IaC)*. „*Infrastructure as Code je přístup k automatizaci správy infrastruktury založený na postupech z vývoje softwaru. Klade důraz na konzistentní, opakovatelné rutiny pro poskytování a změnu systémů a jejich konfiguraci. Provedete změny v kódu a poté pomocí automatizace otestujete a použijete tyto změny ve svých systémech.*“[9]

V publikaci *Infrastructure as Code* jsou uvedeny tři základní principy pro implementaci Infrastructure as Code: definovat vše jako kód, průběžně testovat a dodávat probíhající práci, a budovat malé, jednoduché, části, které je možné nezávisle měnit. Definování všeho jako kódu je základním principem pro spolehlivé provádění změn, a to především z důvodu znovupoužitelnosti[34], konzistence[35] a transparentnosti[36].

Princip průběžného testování a pravidelného dodávání práce je založen na myšlence spíše budovat kvalitu, než ji následně testovat. Využívá k tomu procesy známé jako Continuous Integration (CI)[37] a Continuous Delivery (CD)[38]. Poslední princip, budování malých a jednoduchých částí se zaměřuje na způsob budování systému jako celku. Vychází z předpokladu, že provádění změn do vel-

[34]Kód je možné použít pro budování více instancí. V případě potřeby je možno kód jednoduše upravit/opravit

[35]Vše sestavené za pomoci stejného kódu je postaveno stejným způsobem

[36]Samotný kód popisuje, jak je daná věc postavena. Slouží tedy i jako dokumentace. Na základě historie aplikovaných změn může také pomoci odhalit případné chyby.

[37]Continous Integration je postup automatizace integrace změn kódu od více přispěvatelů do jediného softwarového projektu, po kterém se spouští automatizovaná sestavení a testy s cílem rychleji najít a řešit chyby, zlepšit kvalitu softwaru a zkrátit dobu potřebnou k ověření a vydání nových verzí softwaru [10]

[38]Continuous Delivery je postup vývoje softwaru, kdy se jednotlivé změny automaticky připravují na vydání, před kterým však projdou standardizovaným testovacím procesem [11]

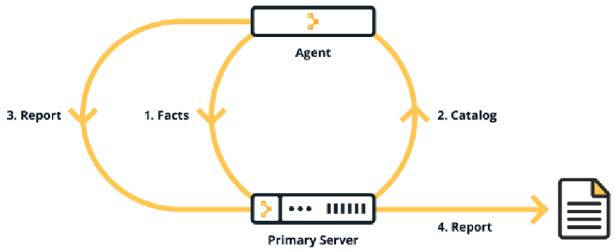
kých monolitických systémů je složité a často může dojít k rozbití systému nebo jeho části, a je tedy vhodné celý systém budovat z menších částí.

Nástroje spadající pod Infrastructure as Code je možné rozdělit do dvou kategorií: na nástroje pro konfigurační management a na tzv. orchestrátory. Nástroje pro konfigurační management jsou určeny pro automatizovanou konfiguraci jednotlivých serverů a jejich komponent. Veškeré změny konfigurace jsou prováděny pomocí změn v kódu, které se následně pomocí nástrojů konfiguračního managementu provedou na jednotlivých serverech/clusterech. Tyto nástroje se však nezabývají vytvářením a rušením jednotlivých serverů, alokací zdojů, atd. Zjednodušeně bychom tedy mohli říct, že se jedná o nástroje zabývající se pouze konfigurací již předem vytvořených serverů. Orchestrační nástroje se však zabývají i automatizací poskytování zdrojů, tedy vytváření serverů, alokace zdrojů, atd. Orchestrační nástroje jsou využívány především v rámci cloudového prostředí, potažmo ve velmi dynamickém prostředí.

V rámci jednotlivých infrastruktur, ve kterých jsou provozovány stávající autentizační proxy, je k dispozici virtualizační platforma umožňující běh virtuálních serverů postavená na technologii VMware vSphere[39], přičemž jednotlivé infrastruktury poskytují více geograficky oddělených lokalit. To je dobrým předpokladem pro provoz stabilní služby ve vysoké dostupnosti. Z tohoto důvodu, a také z důvodu, že autentizační proxy není přizpůsobena pro běh v cloudovém prostředí, se dále budeme zabývat pouze nástroji pro konfigurační management. Nejznámějšími jsou: Ansible a Puppet, nicméně existují i další (např. Salt, CFEngine, Chef). Pomocí jednotlivých nástrojů je možné dosáhnout stejných výsledků. Liší se však převážně způsobem zápisu a způsobem komunikace mezi řídícím strojem a jednotlivými spravovanými servery.

Puppet[40] je open-source nástroj (existuje i verze Puppet enterprise[41], která již není open-source) pro správu a automatizaci konfigurace serverů. Zápis je prováděn pomocí tzv. Puppet Domain-Specific Language (DSL), který je podobný jazyku Ruby. Kód je deklarativní, tedy popisuje požadovaný stav systému, nikoli kroky potřebné k dosažení tohoto stavu. Puppet je založen na architektuře klient-server, kde primární řídící server (dále Puppet server) udržuje kód definující požadovaný stav systému. Puppet server poté spouští jednotlivé agenty, s kterými komunikuje pomocí protokolu HTTPS. Puppet agent, který je nainstalován na spravovaném serveru, nejdříve sesbírá základní informace o serveru (hostname, IP adresa, operační systém, atd.) a předá je Puppet serveru. Puppet server provede kompilaci katalogu, dokumentu, popisující cílový stav uzlu. Puppet agent si následně stáhne z Puppet serveru katalog vtahující se k uzlu na kterém běží, přeloží jeho kód do příkazů, které poté provede na daném uzlu. Po dokončení konfigurace odešle Puppet agent záznam o provedených akcích Puppet serveru. Celý tento proces je ilustrován na obrázku 9.

[39] https://www.vmware.com
[40] https://www.puppet.com/community/open-source
[41] https://www.puppet.com/products/puppet-enterprise



Obrázek 9: Schéma komunikace Puppet serveru a Puppet agenta[12]

Ansible[42] je open-source nástroj napsaný v jazyce Python pro správu a automatizaci serverů. Jeho hlavním cílem je jednoduchost a snadné použití. Zaměřuje se také na bezpečnost a spolehlivost. Zápis je prováděn pomocí jazyka YAML[43], a to také deklarativně. Je založen na architektuře typu klient-server (server je označován jako řídící uzel, klienti jsou poté spravované uzly), přičemž komunikace probíhá pomocí protokolu SSH (Ansible umožňuje i jiné formy komunikace mezi řídícím a spravovaným uzlem[44]). Výhodou Ansiblu je skutečnost, že na straně klienta není nutný běh žádné aplikace. Řídící uzel musí mít pouze povolen přístup na daný spravovaný uzel pomocí protokolu SSH.

Popis cílové konfigurace systému je uložen na straně řídícího uzlu ve formě tzv. playbooků [45]. Řídící uzel se připojuje na jednotlivé spravované uzly a synchronně či asynchronně (dáno konfigurací) provede jednotlivé úkoly, a to za pomoci tzv. Ansible modulů[46]. Tyto moduly jsou řídícím uzlem přesunuty na spravované uzly, kde jsou spuštěny. Po provedení požadované akce modul zaznamená výstup a výsledek operace, a tyto informace vrátí řídícímu uzlu. Jednotlivé moduly jsou po vykonání úkolu ze spravovaných uzlů odstraněny.

Oba nástroje, jak Puppet tak i Ansible, jsou tedy vhodné pro konfigurační management autentizační proxy. Na základě výše uvedeného popisu obou nástrojů byl nakonec vybrán Ansible, a to z několika důvodů. Hlavním z nich byla jednoduchost a srozumitelnost zápisu,a také využívání obecně známého formátu YAML. To usnadní integraci nového způsobu správy autentizační proxy správcům, jelikož se nebudou muset učit nový jazyk pro správu a bude pro ně také jednodušší dané řešení dále udržovat a rozvíjet. Mezi další důvody můžeme zařadit skutečnost, že není třeba instalovat na spravované uzly žádné agenty, a také zvolení Ansiblu jako nástroje pro automatizaci a správu konfigurace IdM systému Perun, což umožní sdílení cenných zkušeností i některých částí kódu.

[42] https://www.ansible.com

[43] Lidsky čitelný formát pro serializaci strukturovaných dat; https://yaml.org

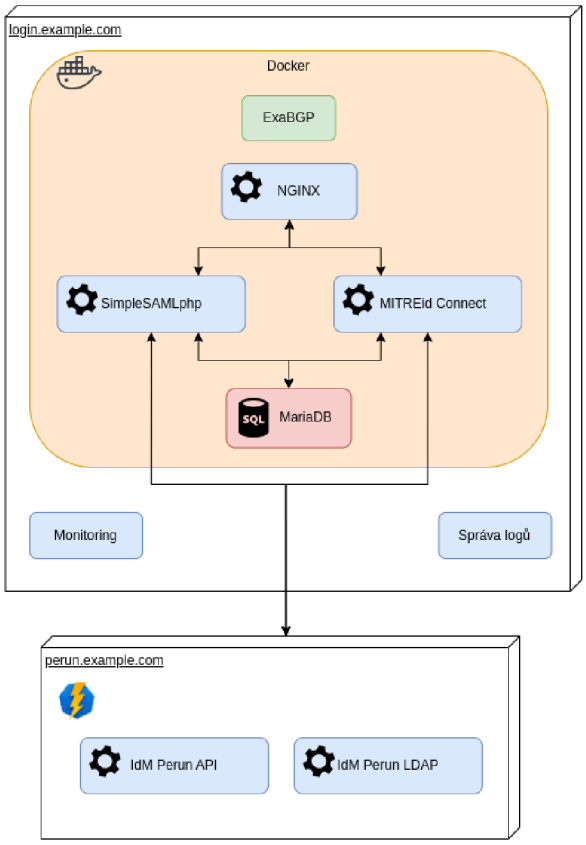
[44] https://docs.ansible.com/ansible/latest/network/getting_started/network_differences.html

[45] Soubor obsahující instrukce pro provádení úkolů (tzv. Ansible task) na spravovaných zařízeních;https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html

[46] Spustitelné skripty nebo aplikace implementující funkcionalitu odpovídající danému úkolu.

5.2 Změny v architektuře autentizační proxy

Předchozí analýza ukázala kromě výše uvedené změny způsobu konfigurace jednotlivých serverů také další možnosti, kde by bylo možné architekturu autentizační proxy upravit tak, aby odpovídala modernímu způsobu provozu systémů, popřípadě kde by bylo možné využít některé z výhod, které architektonická úprava přináší. Tyto jednotlivé návrhy změn budou popsány v následujících podkapitolách. Změny navržené v rámci architektury jednoho uzlu jsou také znázorněny schématem 10.



Obrázek 10: Schéma návrhu architektury jednoho uzlu

5.2.1 Kontejnerizace

Původní architektura autentizační proxy byla založena na klasickém způsobu nasazování a provozu systému, kdy jednotlivé aplikace (subsystémy) jsou spuštěny pod operačním systémem virtuálního serveru. Tento způsob provozu systémů je však již značně zastaralý a má mnohé nevýhody, které jsou znatelné především u systému sestávajících z mnoha komponent. Mezi tyto nevýhody patří zejména nutnost instalace a správy všech závislostí, které jsou často odlišné dle volby operačního systému, či konkrétní verze operačního systému. V určitých případech

také může docházet ke konfliktům, kdy více aplikací závisí na stejné knihovně, avšak na jiné verzi.

V dnešní době se často používá tzv. kontejnerizace. Jde o metodu zapouzdření zdrojového kódu aplikace spolu se všemi závislostmi (knihovny, frameworky, atd.) do jednoho izolovaného kontejneru. Ty je pak možné přesouvat a spouštět konzistentně v jakémkoli prostředí nezávisle na operačním systému.[13]

Jedním ze systémů pro kontejnerovou virtualizaci je Docker[47]. Jde o open-source platformu, která umožňuje kontejnery vytvářet, distribuovat a také spouštět. Docker využívá klient-server architekturu. Na virtuálním stroji je spuštěn Docker daemon (někdy označován jako Docker Engine), který umožňuje vytvářet, spravovat a spouštět kontejnery (Docker containers). K Docker Engine se připojuje některý z klientů (např. Docker CLI, který umožňuje správu dockerových kontejnerů pomocí příkazové řádky). Docker Engine i klient mohou běžet na stejném serveru, jeden klient může být připojen k více serverům současně. Docker Engine je dále zodpovědný za stahování a správu Dockerových obrazů (Docker Images) na lokálním úložišti.

Docker Image je statická šablona obsahující všechny komponenty (včetně kódu, závislostí, konfigurací a operačního systému) potřebné pro běh aplikace. Obrazy jsou často založeny na jiných obrazech, které rozšiřují o další aplikace a závislosti. Jednotlivé obrazy se skládají z několika vrstev a nejčastěji se vytváří na základě tzv. Dockerfile[48] souboru. Vytvořené obrazy je možné uložit do Docker Registry[49], přičemž existuje výchozí registr Docker Hub[50]. V rámci něj je dostupné velké množství obrazů a tento registr slouží jako výchozí při hledání a stahování obrazů.

Docker kontejner je poté spustitelnou instancí obrazu, přičemž každý kontejner běží odděleně od ostatních a má své souborové systémy, procesy, uživatelské adresní prostory a síťová rozhraní. Ve výchozím nastavení je kontejner relativně dobře izolován od ostatních kontejnerů i od hostitelského systému, což znamená, že jednotlivé kontejnery neovlivňují ostatní aplikace a nemohou být ovlivňovány změnami v okolním prostředí. Kontejner může být připojen k jedné nebo více sítím (tzv. Docker network)[51]. Ke kontejneru je možné také připojit úložiště, které zajistí zachování dat i po zrušení kontejneru. Po odstranění kontejneru dojde ke smazání všech změn stavů oproti základnímu obrazu, které nebyly uloženy v trvalém úložišti[52].

V rámci nové architektury autentizační proxy bude využita platforma Docker,

[47] https://docs.docker.com/get-started/overview/

[48] Textový soubor obsahující seznam instrukcí pro sestavení obrazu. Umožňuje specifikovat základní obraz (tzv. base image), definovat závislosti, kopírovat soubory do obrazu, nastavit proměnné prostředí a určit spouštěcí příkaz aplikace;https://docs.docker.com/engine/reference/builder/

[49] Registr určený pro ukládání a distribuci obrazů.

[50] https://hub.docker.com

[51] Síťový mechanismus umožňující komunikaci mezi kontejnery popřípadě mezi kontejnerem a hostitelským systémem nebo jinými externími sítěmi;https://docs.docker.com/network/

[52] https://docs.docker.com/storage/

přičemž v rámci Docker kontejnerů by měly být provozovány všechny hlavní a podpůrné komponenty. Doplňkové komponenty nebudou provozovány jako kontejnery. Např. aplikace pro správu logů, rsyslog a logrotate, jsou distribuovány jako součást využívaného operačního systému. Současně tak bude zajištěno, že správa logů nebude závislá na dostupnosti a stavu Docker Engine. Obdobně to platí i pro monitoring.

5.2.2 Vysoká dostupnost

Jak již bylo popsáno v kapitole 4.1.1, u stávajících instancí autentizační proxy jsou využívány dva různé nástroje pro zajištění vysoké dostupnosti. Oba nástroje jsou značně rozdílné a vyžadují jisté znalosti pro pochopení, jak dané nástroje a protokoly fungují, a také jak je správně nakonfigurovat. Z uvedeného důvodu bude v rámci nové architektury využíván pouze nástroj ExaBGP. Ten sice vyžaduje síťovou podporu na straně infrastruktury, ve které je provozován (ve stávajících infrastrukturách zajištěno), ale přináší značné výhody při provozu clusteru. Podrobný popis nástroje ExaBGP i protokolu BGP[53] je nad rámec této práce.

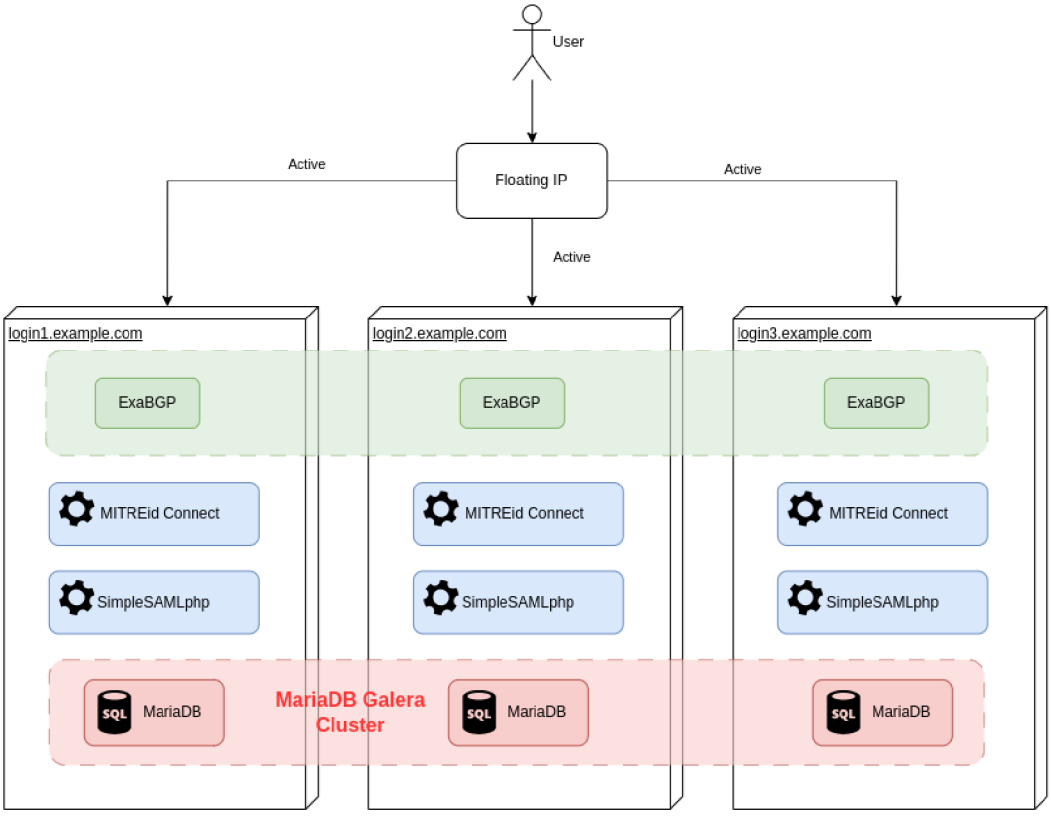
ExaBGP není nástroj určený přímo pro zajištění vysoké dostupnosti, nicméně jde o nástroj pro dynamické řízení směrování v počítačových sítích. Toho je možné využít jak při budování systému s vysokou dostupností, tak i pro rozložení zátěže mezi více uzlů, tzv. Load Balancingu. V rámci každého uzlu je spuštěn nástroj ExaBGP, který v pravidelných intervalech oznamuje směrovačům (tzv. Route Reflector), zda na něj může být směrován síťový provoz, popřípadě s jakou prioritou. Ty následně poté rozhodují, na který uzel bude provoz směrován.

Dosavadní instance autentizační proxy jsou provozovány v rámci clusterů o třech uzlech (jeden aktivní a dva záložní). Celkový počet uzlů je v tuto chvíli dostatečný, nově však bude možné provoz distribuovat mezi více uzlů. Pro použití rozložení zátěže mezi více uzly je však nutné mít k dispozici větší počet veřejných IP adres. Tento počet odpovídá maximálnímu počtu uzlů, které mají být v jeden okamžik aktivní (dále jen aktivní uzly). Pro autentizační proxy budeme počítat se situací, kdy je možné provoz rozložit mezi všechny tři uzly. Celkový počet uzlů, popřípadě počet současně aktivních uzlů se však může časem měnit na základě předpokládané zátěže jednotlivých instancí. V rámci nástroje ExaBGP je možné monitorovat stav jednotlivých komponent daného uzlu a následně se rozhodnout, zda je na daný uzel možné směrovat provoz či nikoli. Změny navržené v rámci nástroje pro zajištění vysoké dostupnosti jsou znázorněny schématem 11.

5.2.3 Webový server

Původní architektura využívala jako webový server Apache HTTP Server. V rámci nové architektury však byla vybrána alternativa ve formě nástroje NGINX[54], který je oproti Apache HTTP Serveru vhodnějším nástrojem pro přesměrovávání požadavků na další aplikační servery (tzv. Reverse proxy). Je efektivnější

[53] A Border Gateway Protocol ; https://www.rfc-editor.org/rfc/rfc4271
[54] https://www.nginx.com



Obrázek 11: Zjednodušené schéma návrhu architektury clusteru

při zpracovávání většího množství dotazů a spotřebovává menší množství operační paměti. Mezi další výhody také můžeme zařadit způsob konfigurace.

6 Implementace správy konfigurace a nové architektury prostředí autentizační proxy

V rámci předchozí kapitoly byly popsány navržené změny správy konfigurace autentizační proxy, které s sebou přináší i určité změny v rámci architektury jednotlivých serverů. Vývoj a implementace těchto změn probíhala na serverech provozovaných na virtualizační platformě Masarykovy Univerzity[55], ve které jsou provozovány některé instance autentizační proxy, a to konkrétně na serverech s operačním systémem Debian[56]. Proces implementace jednotlivých změn a také shrnutí aktuálního stavu nasazení jsou popsány v následujících podkapitolách.

[55] https://it.muni.cz/sluzby/virtualizacni-platforma-mu
[56] https://www.debian.org

6.1 Příprava obrazů pro docker kontejnery

Jak již bylo popsáno v předcházejících kapitolách, v rámci původní architektury byly provozovány jednotlivé aplikace přímo na serveru. Nově budou některé aplikace (převážně hlavní a podpůrné komponenty) provozovány v rámci izolovaných Docker kontejnerů. Aby však bylo možné dané aplikace v kontejnerech provozovat, bylo nutné připravit Docker obrazy, popřípadě najít vhodné Docker obrazy v rámci veřejných registrů.

V rámci navržené architekury, popsané na obrázku 10 se jedná o Docker obrazy pro aplikace Nginx, SimpleSAMLphp, MITREid Connect, ExaBGP a MariaDB. Obraz aplikace Nginx[57] je dostupný v rámci registru Docker Hub jakožto oficiální Docker obraz a byl tedy využit. Obrazy pro ostatní aplikace v rámci registru nejsou dostupné, již nejsou udržované, a nebo měly některé nedostatky, kvůli kterým musely být nahrazeny obrazy vlastními. Vlastní obrazy jsou aktuálně uchovávány v rámci interní platformy GitLab, která současně zajišťuje automatizované vytváření jednotlivých obrazů, které jsou následně distribuovány pomocí Gitlab Container registry[58]. Automatické vytváření Docker obrazů bylo připraveno a implementováno kolegou autora této práce a proto se tímto tématem práce nebude zabývat. Při vzniku této práce byly vytvořeny předpisy pro sestavení Docker obrazů jednotlivých aplikací. Tyto obrazy však od vzniku práce prošly značnými úpravami na základě požadavků z provozu, přičemž autor práce není jediným autorem těchto změn. U jednotlivých obrazů tedy bude uveden rozsah prací v rámci daného obrazu. V příloze poté budou uvedeny aktuální verze předpisů pro sestavení jednotlivých Docker obrazů, spolu s ukázkou příkazu pro sestavení.

6.1.1 Obraz aplikace SimpleSAMLphp

Jedná se o relativně komplikovaný obraz, který využívá více fází při sestavování Docker obrazu (tzv. Multi-stage builds[59]), kdy výstupem každé fáze je samostatný Docker image, který může být poté použit v některé z dalších fází.

Dockerfile se skládá celkem z pěti fází. V první fázi dojde k přípravě obrazu obsahující PHP spolu s nutnými závislostmi. Druhá fáze připraví dočasný obraz balíčkovacího nástroje Composer[60]. V třetí fázi je jako předloha využit obraz z první fáze a nástroj Composer z druhé fáze. V této fázi dojde k nainstalování nástroje SimpleSAMLphp spolu se všemi moduly nutnými pro provoz dané instance. Během čtvrté fáze dojde k nainstalování všech dalších závislostí pomocí NodeJS[61]. Poslední fáze využívá jako předlohu výstup z první fáze, do které je překopírována aplikace SimpleSAMLphp spolu se všemi závislostmi.

Komplikovanost tohoto obrazu je dána především skutečností, že jednotlivé

[57] https://hub.docker.com/_/nginx
[58] https://docs.gitlab.com/ee/user/packages/container_registry/
[59] https://docs.docker.com/build/building/multi-stage/
[60] https://getcomposer.org
[61] https://nodejs.org/en/about

finální instance vyžadují jinou množinu instalovaných modulů. Z tohoto důvodu je Docker obraz sestavován vždy pro konkrétní instanci dle použití. Současně je umožňěno v rámci první a třetí fáze vyvolat na počátku a na konci dané fáze skript, který může například pro konkrétní instanci provést instalaci požadovaných knihoven.

V rámci tohoto obrazu autor práce vytvořil návrh a prvotní implementaci předpisu pro sestavení obrazu, včetně více fázového sestavení a možnosti volat již zmíněné skripty.

6.1.2 Obraz aplikace MITREid Connect

Jde o obraz obsahující aplikační server Tomcat spolu s aplikací MITREid Connect. Mimo tuto aplikaci je Docker obraz doplňen o aplikaci sloužící k synchronizaci dat o koncových službách (tzv. klientech) využívající protokoly OAuth 2.0 a OpenID Connect z IdM systému Perun do aplikace MITREid Connect. V rámci původního návrhu bylo součástí sestavení obrazu i samotné sestavení obou aplikací pomocí nástroje Maven[62]. V současné době je toto sestavení prováděno automaticky po vydání nové verze dané aplikace a při sestavování obrazu tedy dojde jen ke stažení již sestavené verze aplikace. U tohoto obrazu autor práce vytvořil původní návrh a implementaci obrazu.

6.1.3 Obraz aplikace MariaDB spolu s možností Galera Clusteru

V rámci registru Docker Hub existuje oficiální obraz obsahující databázi MariaDB[63]. Ten však není přímo uzpůsoben pro provoz jakožto MariaDB Galera Cluster, tedy na více uzlech s možností více aktivních uzlů současně. Z uvedeného důvodu byl na jeho základě vytvořen vlastní obraz s upraveným skriptem pro spuštění v režimu Galera Cluster. Dále je tento obraz rozšířen o skript pro monitoring daného uzlu, a také o skript sloužící k vytváření záloh databáze a uchovávání těchto záloh po předem specifikovaný počet dní. U tohoto obrazu autor práce vytvořil doplňující skripty pro monitoring a zálohování daného uzlu, a dále provedl některé úpravy v rámci spouštěcího skriptu. Základ obrazu byl převzat od kolegů z Masarykovy Univerzity.

6.1.4 Obraz aplikace ExaBGP

Tento obraz vychází ze základního obrazu obsahující Python, do kterého je doinstalována aplikace ExaBGP ve formě Python balíku[64] a dalších závislostí. Součástí obrazu je také instalace klienta pro Docker, který poté umožňuje uvnitř kontejneru provádět monitoring ostatních Docker kontejnerů a na základě těchto výsledků případně provádět úpravy ve směrovácích tabulkách sítě. Autor práce vytvořil předpis pro sestavení tohoto obrazu.

[62] https://maven.apache.org
[63] https://hub.docker.com/_/mariadb
[64] https://pypi.org/project/exabgp/

6.2 Vytvoření Ansible playbooku

Jakmile byly připraveny Docker obrazy jednotlivých aplikací, bylo možné postoupit k přípravě Ansible předpisu a s tím také k přípravě jednotlivých Ansible rolí. Jak již bylo zmíněno v kapitole zabývající se výběrem vhodného nástroje pro konfigurační management, jedním z důvodů, proč byl vybrán právě nástroj Ansible, bylo možné sdílení vědomostí a také již vytvořených rolí. V rámci výsledného předpisu jsou tedy využívány i role, které vznikly v rámci automatizace nasazení IdM systému Perun. Obdobně některé role vyvinuté v rámci této bakalářské práce byly dále rozšiřovány kolegy autora práce a jsou, popřípadě dále mohou být, používány i pro automatizaci nasazení jiných nástrojů. V části popisu jednotlivých Ansible rolí bude opět uvedeno, které role, popřípadě jaké části předpisu byly vytvořeny autorem práce.

6.2.1 Popis Ansible playbooku a základní role *cesnet.proxyaai*

Samotný Ansible předpis pro instalaci a konfiguraci autentizační proxy, *playbook.yml*, je velmi jednoduchý, jelikož ve skutečnosti pouze specifikuje, že bude spuštěna Ansible role s názvem *cesnet.proxyaai*. Ta obsahuje specifikaci všech úkolů, které budou na cílovém serveru vykonány, přičemž úkolem je často načtení dalšího souboru s úkoly, popřípadě jiné Ansible role. Načítání dalších rolí, popřípadě souborů s úkoly je výhodné nejen z důvodu větší přehlednosti v rámci celého Ansible předpisu, ale umožňuje také sdílení Ansible rolí i u jiných projektů. Samotnou roli *cesnet.proxyaai* bychom mohli rozdělit na dvě části, přičemž první část se zabývá konfigurací serveru jako takového a přípravou prostředí, zatímco druhá část se již zaměřuje na konfiguraci jednotlivých komponent a nástrojů.

V rámci první části dojde k instalaci potřebných balíků, konfigurace přístupů k danému serveru (přidání SSH klíčů pro přímý přístup na superuživatele, vytvoření uživatelských účtů, nastavení práv uživatelů pro povýšení práv na superuživatele pomocí ověření osobním hardwarovým klíčem Yubikey[65], zakázání autentizace k serveru pomocí hesla), konfiguraci nástroje crontab[66], přípravu prostředí systému, konfiguraci synchronizace času, konfiguraci firewallu a nastavení automatické aktualizace systémových balíků. V této části předpisu jsou využívány role *cesnet.ntp*[67], *cesnet.firewall*[68], *cesnet.yubikeys*[69] a *cesnet.unattended_upgrades*[70],

[65] Hardwarový bezpečnostní token ve formě USB klíčenky. Po dotyku tlačítka vygeneruje jednorázové heslo; https://www.yubico.com

[66] Proces běžící na pozadí v systému Linux/Unix pro zajištění spouštění plánovaných úkolů v přesný čas nebo v pravidelných intervalech.

[67] Ansible role pro nastavení synchronizace času pomocí NTP protokolu;https://github.com/CESNET/ansible-role-ntp

[68] Anisble role pro nastavení firewallu pomocí nástroje iptables; https://github.com/CESNET/ansible-role-firewall

[69] Ansible role pro nastavení přihlášení na superuživatele pomocí hardwarového tokenu; https://github.com/CESNET/ansible-role-yubikeys

[70] Ansible role pro nastavení automatických aktualizací systémových balíků;https://github.com/CESNET/ansible-role-unattended-upgrades

které nejsou dílem autora práce. Tyto Ansible role jsou také dostupné v rámci jednoho společného repozitáře *perun-ansible-roles*[71].

Druhá část této role se zabývá již samotnou instalací a konfigurací jednotlivých komponent pro poskytování služeb autentizační proxy, popřípadě podpůrných komponent či nástrojů, které jsou nutné k provozu. Tato část role je uvedena v rámci ukázky zdrojového kódu 1. Na počátku, a následně také na konci, je umožňěno načíst soubor s dalšími úkoly, které je nutno vykonat specificky pro některou ze spravovaných instancí, popřípadě na konkrétním serveru. Následuje instalace a konfigurace nástroje Docker pomocí role *cesnet.proxyaai_docker* a upravení konfigurace pro odesílání emailů z Docker kontejnerů pomocí role *cesnet.sendmail_for_docker*[72]. Dále jsou nainstalovány a nakonfigurovány doplňkové nástroje pro monitoring a správu logů (role *cesnet.metacentrum_monitoring*, *cesnet.rsyslog* a *cesnet.logrotate*). Poté již následuje instalace a konfigurace jednotlivých podpůrných komponent ExaBGP, webového serveru (Nginx), databáze a následně hlavních komponent SimpleSAMLphp a MITREid Connect. Některé z těchto komponent nejsou instalovány a konfigurovány automaticky, ale je nutné je pomocí dodatečných parametrů povolit. Toto rozhodnutí bylo učiněno především z důvodu dosažení větší modularity. Popis jednotlivých rolí bude uveden dále.

Zdrojový kód 1: Ansible role cesnet.proxyaai - druhá část

```
---
- name: "Custom pre tasks"
  when: site_specific_pre_tasks is defined
  include_tasks:
    file: "files/{{ site_specific_pre_tasks }}"
    apply:
      tags:
        - site_specific
        - site_specific_pre_tasks
  tags:
    - site_specific
    - site_specific_pre_tasks

- name: "DOCKER"
  import_role:
    name: cesnet.proxyaai_docker
  tags:
    - docker

- name: "SENDMAIL - SMTP setup for Dockerized environment"
  import_role:
    name: cesnet.sendmail_for_docker
  tags:
    - sendmail

- name: "Monitoring"
  import_role:
    name: cesnet.metacentrum_monitoring
  when: components_monitoring == "enabled"
  tags:
```

[71] https://github.com/CESNET/perun-ansible-roles

[72] Ansible role pro konfiguraci odesílání emailů z Docker kontejnerů. Autorem této role není autor práce; https://github.com/CESNET/ansible-role-sendmail-for-docker/

```
31      - monitoring
32
33  - name: "RSYSLOG"
34    import_role:
35      name: cesnet.rsyslog
36    tags:
37      - rsyslog
38
39  - name: "LOGROTATE"
40    import_role:
41      name: cesnet.logrotate
42    tags:
43      - logrotate
44
45  - name: "ExaBGP"
46    import_role:
47      name: cesnet.exabgp_docker
48    when: components_exabgp == "enabled"
49
50  - name: "REVERSEPROXY"
51    import_role:
52      name: cesnet.reverseproxy_docker
53    tags:
54      - reverseproxy
55
56  - name: "MARIADB"
57    import_role:
58      name: cesnet.mariadb_docker
59    when: components_mariadb == "enabled"
60    tags:
61      - mariadb
62
63  - name: "SIMPLESAMLPHP"
64    import_tasks: "simplesamlphp.yml"
65    tags:
66      - simplesamlphp
67
68  - name: "MITREID"
69    import_tasks: "mitreid.yml"
70    when: components_mitreid == "enabled"
71    tags:
72      - mitreid
73
74  - name: "Custom post tasks"
75    when: site_specific_post_tasks is defined
76    include_tasks:
77      file: "files/{{ site_specific_post_tasks }}"
78      apply:
79        tags:
80          - site_specific
81          - site_specific_post_tasks
82    tags:
83      - site_specific
84      - site_specific_post_tasks
```

6.2.2 Popis jednotlivých částí Ansible předpisu

Jak bylo uvedeno v předchozí kapitole, druhá část Ansible předpisu je rozdělena do několika jednotlivých úkolů. Ty jsou často vykonáním konkrétní Ansible role, popřípadě načtením souboru s dalšími úkoly, které budou dále popsány v následujících podkapitolách.

6.2.2.1 Ansible role pro instlaci a konfiguraci Dockeru

V rámci role *cesnet.proxyaai_docker* dojde k instalaci Docker Engine a jeho závislostí a také nástroje Docker Compose[73], pomocí kterého jsou následně jednotlivé Docker kontejnery spouštěny. Tato role dále zajišťuje připojení k Docker registrům, konfiguraci sítě a vytváření tzv. Docker volumes. Poslední částí této role je nastavení automatického mazání již nepotřebných kontejnerů a obrazů.

6.2.2.2 Ansible role pro instalaci a konfiguraci ExaBGP

Komponenta ExaBGP je provozována jako Docker kontejner, jehož instalaci a konfiguraci zajišťuje role *cesnet.exabgp_docker*. Jelikož je komponenta ExaBGP odpovědná za dynamické směrování síťového provozu na jednotlivé uzly clusteru, bylo nutné tuto roli rozšířit také o možnost vzdáleného ovládání této komponenty. Pomocí parametrů při spuštění Ansible playbooku je možné tuto komponentu zapnout, vypnout, popřípadě restartovat. Současně i samotné spuštění komponenty je nutno vyvolat pomocí parametru při spuštění Ansible playbooku. Pospaná implementace byla realizováná z důvodu, aby nedocházelo k modifikaci směrovacích tabulek během provádění konfigurace jednotlivých komponent na daném uzlu, a tedy bylo umožněno jednoduché odstavení uzlu ze seznamu aktivních uzlů.

6.2.2.3 Ansible role pro instlaci a konfiguraci webového serveru

Instalaci a konfiguraci webového serveru, konkrétně Nginx, je zajištěna pomocí role *cesnet.reverseproxy_docker*. Ta je dostupná také pomocí GitHub repozitáře[74]. Jak je patrné z názvu této role, komponenta Nginx je provozována jako Docker kontejner.

6.2.2.4 Ansible role pro instlaci a konfiguraci databáze MariaDB

Databáze MariaDB je provozována také ve formě Docker kontejneru. Instalaci a konfiguraci zajišťuje role s názvem *cesnet.mariadb_docker*, která kromě samotné konfigurace a spuštění databáze umožňuje také vytvoření jednotlivých databází, import dat, vytvoření uživatelů a správné nastavení přístupových práv k jednotlivým databázím a tabulkám.

V rámci této role bylo nutno vyřešit dva problémy. První z nich představuje nutnost vyčkat na nastartování databázového prostředí před vytvářením jednotlivých databází, přičemž start databáze díky provozu MariaDB Galera Clusteru může trvat delší dobu. Toto bylo vyřešeno periodickou kontrolou, zda je již možné se připojit k databázi. Druhý problém opět představuje provoz databáze v clusteru, jelikož je nutné zajistit, že nedojde k vypnutí nebo restartu databáze na všech uzlech současně. Z uvedeného důvodu tato role neprovádí restart kontejneru po úpravě konfigurace, nicméně upozorňuje na nutnost manuálního restartu.

[73] https://docs.docker.com/compose/
[74] https://github.com/CESNET/ansible-role-reverseproxy-docker

6.2.2.5 Sady úkolů pro instlaci a konfiguraci nástrojů SimpleSAMLphp a MITREid Connect

V rámci obou sad úkolů dojde k postupné konfiguraci a spušťění aplikací SimpleSAMLphp (sada úkolů *simplesamlphp.yml*) a MITREid Connect (sada úkolů *mitreid.yml*) a to uvnitř Docker kontejnerů. V rámci obou sad úkolů je též možné vyvolat některé specifické úkoly pouze pro specifické instance nebo servery.

6.2.2.6 Ansible role pro instalaci a konfiguraci doplňkových komponent

Tato část zahrnuje instalaci a konfiguraci nástrojů pro správu logů a komponenty pro monitorování, a to pomocí rolí *cesnet.metacentrum_monitoring*[75], *cesnet.rsyslog*[76] a *cesnet.logrotate*[77]. Tyto role byly vytvořeny autorem před vznikem této práce a dále rozšiřovány kolegy autora.

6.2.3 Doplňkové Ansible předpisy

V rámci finálního řešení byly kromě Ansible předpisu pro sestavení a konfiguraci autentizační proxy vytvořeny i další, doplňkové předpisy. Jsou zpravidla určeny pro usnadnění práce a jde o předpisy např. pro vygenerování žádostí o serverový certifikát, instalaci systémových aktualizací a případný restart serveru, je-li vyžadován, restart Docker kontejneru s databází nebo pro přípravu síťových rozhraní. Vyjma těchto předpisů byl vytvořen také jednoduchý předpis, který umožní vytvoření centrálního logovacího serveru, na který poté mohou jednotlivé uzly clusteru odesílat své logy.

[75] Ansible role pro konfiguraci monitorovací komponenty;https://github.com/CESNET/ansible-role-metacentrum-monitoring

[76] Ansible role pro konfiguraci nástroje rsyslog; https://github.com/CESNET/ansible-role-rsyslog

[77] Ansible role pro konfiguraci nástroje logrotate; https://github.com/CESNET/ansible-role-logrotate

Závěr

Cílem této bakalářské práce bylo provést analýzu akademické autentizační proxy ekosystému Perun, návrh vhodné změny architektury prostředí s důrazem na posílení automatizace provozu a následnou implementaci navržených změn.

Nová architektura přinesla nahrazení dosavadního způsobu provozu aplikací, kdy jednotlivé aplikace jsou spuštěny přímo pod operačním systémem daného serveru. Toto řešení bylo nahrazeno modernějším způsobem, kdy jednotlivé aplikace jsou uzavřeny a provozovány uvnitř nezávislých Docker kontejnerů. Součástí architektonických změn bylo nahrazení nástroje používaného jako webový server a sjednocení používaného nástroje pro zajištění vysoké dostupnosti. Zvolený nástroj nyní umožňuje provoz více aktivních uzlů současně oproti původnímu řešení provozu jediného aktivního uzlu, což umožňuje rozložení zátěže mezi aktivní uzly clusteru. Zvolený nástroj také umožňuje provádení předem specifikovaných kontrol pro automatické odstavení uzlu v případě výpadku některé komponenty.

Jednotlivé servery již nejsou spravovány a konfigurovány manuálně, ale je využito nástroje pro správu konfigurace Ansible. Tato změna přinesla mnohé výhody, mezi které můžeme zařadit minimalizaci manuálních zásahů u jednotlivých serverů, lepší dokumentaci, a také urychlila celý proces nasazení nových verzí, případně úprav konfigurace. Současně eliminovala potencionální problémy a výpadky, ke kterým by mohlo dojít důsledkem chybné aplikace změn. Využití tohoto nástroje také umožnuje v případě potřeby jednoduché rozšíření jednotlivých clusterů o nové uzly. Vytvoření nové instance autentizační proxy je nově také výrazně jednodušší.

Navržené a implementované řešení bylo průběžně integrováno do produkčního prostředí a některé z provozovaných instancí autentizační proxy jsou již spravovány tímto novým způsobem. Současně byly pomocí tohoto řešení již vytvořeny také nové instance, a to v různých konfiguracích. Modularita a přizpůsobitelnost výsledného řešení umožnila jeho využití také pro automatizovanou správu několika instancí v režimu pouze poskytovatele idenit nebo poskytovatele služeb.

Zkušenosti z provozu instancí autentizační proxy s více aktivními uzly také ukazují nedostatky komponenty MariaDB Galera Cluster. Tyto nedostatky se projevují především při vyšším zatížení jednotlivých uzlů, kdy dochází k replikaci jednotlivých databázových změn na ostatní uzly a projevují se sníženou odezvou databáze. Při současných změnách na více uzlech však může dojít ke konfliktům, které vedou k zastavení databázového clusteru a vyžadují manuálního zásahu. Z těchto důvodů by v rámci dalšího rozšíření bylo ideální vyčlenění databázového clusteru mimo autentizační proxy, popřípadě využití databázového clusteru od externího poskytovatele.

Od počátku integrace do produkčního prostředí se také objevily nové požadavky na rozšíření tohoto řešení, zejména o správu nových komponent. Tyto byly do řešení postupně zakomponovány, ovšem svým charakterem neodpovídají tématu této práce a proto nejsou explicitně popsány. Tím byla i ověřena možnost rozšiřitelnosti výsledků práce v budoucnosti.

Conclusions

This thesis aimed to perform an analysis of the academic authentication proxy of the Perun ecosystem, create a proposal of a new architecture for the deployment environment with an emphasis on improving the automation of operations and, finally, implementation of the proposed changes.

The new architecture replaces the previously used approach to application operations when particular applications have been run directly under a server operating system environment. This solution has been replaced with a more modern method of encapsulating applications into separated and loosely coupled Docker containers. Proposed architectural changes include replacing a tool suite used as the web server and unifying the components ensuring high availability across various deployments. Chosen software solutions allow the operation of multiple cluster nodes being active simultaneously, in contrast to having just a single active node in the previously used approach. In addition, the replacement software components allow spreading the operation load across the active nodes and performing specific checks for automated disemployment of a node in case of an outage of one of the components running on the particular node.

Configuration management of servers has been upgraded from manually executed work on each node separately to fully automated change management using the Ansible tool. This change brought many advantages, like minimalization of manual changes needed to be conducted on each node separately, self-documentation of the deployment schema, and speeding up the process of deploying new application versions or making configuration changes. Apart from the mentioned changes, usage of this tool eliminated potential problems and outages possibly caused by the faulty application of changes in the manual approach. Choosing Ansible, in addition, enabled easy process of extending the cluster by deploying new nodes and rapidly speeded up the process of creating a new instance of the authentication proxy.

The designed and implemented solution has been continuously integrated into a production environment when some of the instances of the authentication proxy are managed using the proposed approach. Also, several new deployments have been established using the new solution in various configurations. The modularity and customizability of the final solution also allowed us to reuse it for automated management of several instances running in the mode as only Identity Providers or Service Providers.

Operations of authentication proxy instances with multiple active nodes already in production mode have already pointed out some disadvantages of the MariaDB Galera Cluster component. These flaws appear when nodes experience heavy load when the replication of database changes to other nodes lowers the performance of the whole cluster. When an update in the database occurs on multiple nodes simultaneously, a conflict between the modifications might appear, leading to the clusters' failure with the need of manual intervention to restore its proper operation. Because of these reasons, in the follow-up extension

of the proposed solution, a database cluster hosted outside of the authentication proxy or usage of a completely outsourced database cluster is foreseen.

From the beginning of integration into the production environment, several new requirements have been declared, mainly focused on the management of the new component. Implementation of declared additional functionalities has been continuously incorporated into the solution. However, as they do not correspond with the topic of this thesis, they are not described. Despite that, the possibility of extending the final solution has been verified by implementing mentioned requirements.

A Obsah elektronických dat

`text/`
: Adresář s textem práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech (textových) příloh, a všechny soubory potřebné pro bezproblémové vytvoření PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu a příloh, vložené obrázky, apod.

`README.md`
: Textový soubor s instrukcemi k použití Ansible předpisu pro sestavení jedné instance v režimu proxy AAI v minimální konfiguraci (bez aplikaci MIT-REid Connect a napojení na IdM systém) a jedné instance v režimu poskytovatele služby, a to pomocí Ansible předpisu s ukázkovými daty v adresáři `ansible_proxyaai/`.

`vyjadreni.pdf`
: Vyjádření vedoucího pracoviště Bezpečnosti digitálních identit Ústavu výpočetní techniky Masarykovy univerzity, Mgr. Slávka Licehammera, ke stavu integrace výsledného řešení automatizované správy akademické autentizační proxy v produkčním prostředí ekosystému Perun.

`docker_images.zip`
: Adresář (ZIP archiv) obsahující jednotlivé Dockerfile předpisy pro sestavení Docker obrazů. V rámci adresáře je doplněn `README.md` soubor s instrukcemi pro sestavení obrazů a ukázkou příkazů pro sestavení jednotlivých obrazů.

`ansible_roles.zip`
: Adresář (ZIP archiv) obsahující jednotlivé ansible role, které byly vytvořeny autorem této práce. V rámci adresáře je doplněn `README.md` soubor s popisem jednotlivých rolí.

`ansible_proxyaai.zip`
: Adresář (ZIP archiv) obsahující kompletní Ansible předpis spolu s ukázkovými daty pro sestavení jedné instance v režimu proxy AAI (součástí je vytvoření poskytovatelů identit – možná autentizace pomocí lokálních účtů a dále pomocí účtů Google a Microsoft), jedné instance v režimu poskytovatele služby. Dále také data pro doplňující Ansible předpis pro sestavení logovacího serveru.

Literatura

[1] *O nás*. [online]. [cit. 2022-12-10]. Dostupný z: ⟨https://www.cesnet.cz/sdruzeni/⟩.

[2] *About CERIT-SC*. [online]. [cit. 2022-12-10]. Dostupný z: ⟨https://www.cerit-sc.cz⟩.

[3] Kukic, Ado. *The Definitive Guide to Single Sign-On* [online]. [cit. 2023-1-15]. Dostupný z: ⟨https://auth0.com/resources/whitepapers/definitive-guide-to-single-sign-on⟩.

[4] Procházka, Michal; Licehammer, Slavek; Matyska, Luděk. Perun — Modern approach for user and service management. In. *2014 IST-Africa Conference Proceedings*. 2014, s. 1–11. Dostupný také z: ⟨http://dx.doi.org/10.1109/ISTAFRICA.2014.6880654⟩.

[5] Licehammer, Slávek. *Správa atributů systému Perun [online]*. 2012 [cit. 2023-03-23]. SUPERVISOR: RNDr. Michal Procházka, Ph.D. Dostupný také z: ⟨https://theses.cz/id/9jikpa/⟩.

[6] members, AARC Community; members, AppInt. *AARC Blueprint Architecture 2019 (AARC-G045)*. 2019. Dostupný také z: ⟨https://doi.org/10.5281/zenodo.3672785⟩.

[7] Groep, David Leo; Collier, Ian; Dack, Tom aj. *Guidelines for Secure Operation of Attribute Authorities and issuers of statements for entities*. 2022. Dostupný také z: ⟨https://doi.org/10.5281/zenodo.5927799⟩.

[8] Carter, Gerald. *LDAP system administration*. 1st ed. Sebastopol, CA: O'Reilly, c2003. ISBN 1-56592-491-6.

[9] Morris, K.; Safari, an O'Reilly Media Company. *Infrastructure as Code, 2nd Edition*. 2020. Dostupný také z: ⟨https://www.oreilly.com/library/view/infrastructure-as-code/9781098114664/⟩. ISBN 9781098114664.

[10] *What is Continuous Integration?* [online]. [cit. 2023-7-22]. Dostupný z: ⟨https://aws.amazon.com/devops/continuous-integration/⟩.

[11] *What is Continuous Delivery?* [online]. [cit. 2023-7-22]. Dostupný z: ⟨https://aws.amazon.com/devops/continuous-delivery/⟩.

[12] *What is Puppet?* [online]. [cit. 2023-7-22]. Dostupný z: ⟨https://www.puppet.com/docs/puppet/8/what_is_puppet.html⟩.

[13] *What is containerization?* [online]. [cit. 2023-7-26]. Dostupný z: ⟨https://www.redhat.com/en/topics/cloud-native-apps/what-is-containerization⟩.