Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky



Automatizace procesů user managementu na zabezpečeném Hadoop clusteru

DIPLOMOVÁ PRÁCE

Studijní program: Aplikovaná informatika

Studijní obor: Informační systémy a technologie

Autor: Bc. Jan Kodet

Vedoucí diplomové práce: Ing. et Ing Soňa Karkošková, Ph.D.

Praha, duben 2019

Prohlášení

Prohlašuji, že jsem diplomovou práci „Automatizace procesů user managementu na zabezpečeném Hadoop clusteru“ vypracoval samostatně za použití v práci uvedených pramenů a literatury.

V Praze dne 29. dubba 2019 .....

Bc. Jan Kodet

Poděkování

Tímto bych rád poděkoval hlavně paní Ing. et Ing. Soně Karkoškové, Ph.D. za cenné rady, připomínky a vstřícný přístup při vedení diplomové práce a společnosti Adastra s.r.o. za poskytnutí prostředí pro realizaci praktické části této práce.

Abstrakt

Tato diplomová práce se zabývá automatizací uživatelských procesů probíhající na zabezpečené Hadoop platformě. Teoretická část práce poskytuje přehled o Frameworku pro zpracování velkého množství dat Apache Hadoop a jeho komponentách. Jedna z kapitol teoretické části práce je věnována i možnosti zabezpečení platformy, na které je Apache Hadoop nainstalovaný a formulováním Best Practise v oblasti bezpečnosti.

Na základě opakovaných požadavků uživatelů Hadoop platformy jsou v praktické části identifikovány procesy, jež lze automatizovat. Následně je vytvořen návrh automatizace definovaných procesů, který je i implementován do zabezpečeného Hadoop prostředí.

Poslední část práce má za cíl ověřit funkčnost řešení, zformulovat získané zkušenosti a navrhnout možná rozšíření výsledného řešení. Přínosem práce je snížení časové náročnosti řešení častých uživatelských požadavků.

Klíčová slova

Big Data, Apache Hadoop, Uživatelské procesy, Automatizace, Archimate, HDFS, LDAP

Abstract

This diploma thesis deals with the automation of user processes that take place on secured Hadoop platform. The main goal of this thesis is to reduce the time required to solve frequent user requests. Theoretical part provides an overview of Apache Hadoop framework for processing big amount of data, including its components. In addition to the basic theory, one of the chapters is dedicated to security options for the platform, on which Apache Hadoop is installed as well as formulating Best Practices from the security area.

In the practical part, processes that can be automized are identified based on frequent Hadoop platform's users' requests. Then, concept for automatization of defined processes is designed and is implemented in secured Hadoop environment.

The aim of the final part of this thesis is to verify solution functionality, formulate the acquired experience and suggest possible extensions of the solution obtained.

Keywords

Big Data, Apache Hadoop, User Management, Automation, Archimate, HDFS, LDAP

Obsah

Úvod 10
1.1 Cíle práce 10
1.2 Metoda dosažení určených cílů 12
1.3 Struktura práce 13
1.4 Rešerše informačních zdrojů 13
1.4.1 Závěrečné práce 13
1.4.2 Odborná literatura 14
2 Big Data 16
2.1 Definice pojmu 16
2.1.1 Velikost Big Data 17
2.1.2 Big Data systémy 17
2.2 Shrnutí 18
3 Apache Hadoop 19
3.1 Místo Apache Hadoop v podnikové infrastruktuře 20
3.2 Hadoop distribuce 21
3.2.1 Cloudera 22
3.2.2 Cloudera Manager 22
3.2.3 Cloudera Distribution Hadoop 24
3.2.4 Cloudera Navigator 24
3.3 Shrnutí 25
4 Hlavní komponenty Hadoop 26
4.1 MapReduce 26
4.1.1 Tolerance k výpadku 27
4.2 Hadoop Distributed File System (HDFS) 28
4.2.1 Architektura HDFS 28
4.3 YARN 30
4.4 Shrnutí 31
5 Ostatní komponenty Hadoop 32
5.1 HIVE 32
5.2 Impala 33
5.3 HBASE 34
5.4 Hue 35
5.5 Pig 36

5.6 Shrnutí 37
6 Možné zabezpečení Hadoop platformy 38
6.1.1 Autentizace 38
6.1.1.1 Active Directory 38
6.1.1.1.1 Fyzická struktura služby AD DS 39
6.1.1.1.2 Logická struktura služby AD DS 39
6.1.1.2 LDAP 39
6.1.1.3 Kerberos 40
6.1.2 Autorizace 42
6.1.2.1 Linux/HDFS práva na složky 42
6.1.2.2 ACL – Access Control List 43
6.1.2.3 Sentry 43
6.1.3 Audit 44
6.1.4 Zabezpečení dat 44
6.1.4.1 TLS šifrování 45
6.1.5 Shrnutí 46
7 User management 47
7.1 Vymezení user management procesů 47
7.2 Návrh řešení 47
7.3 Popis prostředí 49
7.4 Návrh databáze 51
7.4.1 Konceptuální návrh 51
7.4.2 Logický návrh 52
7.4.3 Fyzický návrh 53
7.4.4 Realizace 54
7.5 Založení uživatele na platformě 55
7.5.1 Zadání uživatelského jména uživatele 56
7.5.2 Založení uživatelských struktur 57
7.5.2.1 Založení Hive databáze 58
7.5.2.2 Založení domovského adresáře na HDFS 58
7.5.2.3 Založení domovského Linux adresáře 58
7.5.3 Založení přístupových práv uživatele 59
7.5.3.1 Vygenerování privátního klíče 59
7.5.3.2 Vygenerování veřejného klíče 60
7.5.3.3 Přesun privátního klíče do domovského adresáře HDFS 60

7.5.3.4 Autorizace veřejného klíče ... 60
7.5.3.5 Vygenerování Keytabu ... 60
7.5.3.6 Založení role v Sentry ... 61
7.5.3.7 Shrnutí ... 61
7.6 Odstranění neaktivních uživatelů z platformy ... 62
7.6.1 Identifikace neaktivních uživatelů ... 63
7.6.2 Odstranění uživatelských struktur ... 63
7.6.2.1 Odstranění Hive databáze ... 63
7.6.2.2 Odstranění domovského adresáře HDFS ... 64
7.6.2.3 Odstranění domovského adresáře ... 64
7.6.3 Odstranění práv uživatele ... 64
7.6.4 Odstranění uživatele z MariaDB ... 65
7.7 Shrnutí ... 65
7.8 Synchronizace uživatelů platformy ... 66
7.9 Založení Workspace ... 67
7.9.1 Požadavek na založení Workspace ... 67
7.9.2 Vložení názvu Workspace a přiřazení uživatelů ... 68
7.9.3 Založení požadavku na vytvoření skupin v AD/LDAP ... 69
7.9.4 Schválení požadavku ... 70
7.9.5 Vytvoření skupin a přiřazení uživatelů ... 70
7.9.6 Kontrola LDAP ... 70
7.10 Vytvoření Workspace ... 71
7.10.1 Založení struktur pro Workspace ... 72
7.10.1.1 Založení Hive databáze ... 72
7.10.1.2 Založení HDFS složky pro Workspace ... 72
7.10.1.3 Založení Linux složky ... 73
7.10.2 Založení práv pro Workspace ... 73
7.10.2.1 Založení Sentry role ... 73
7.10.2.2 Přidělení role skupinám ... 74
7.10.2.3 Definice ACL pro Workspace HDFS/Linux složku ... 74
7.10.2.4 Předání vlastnictví HDFS/Linux složky ... 75
7.10.3 Shrnutí ... 75
7.11 Odstranění Workspace ... 75
7.12 Založení PoC Workspace ... 78
7.12.1 Založení struktur PoC Workspace ... 80

7.12.2 Přiřazení uživatelů do PoC Workspace 80
7.12.3 Shrnutí 81
Závěr 82
Použitá literatura 85
Přílohy I

Úvod

Doba, kdy pojem Big Data byl Buzzwordem je pryč. Objemy dat se neustále zvětšují a zahlcují nás ze všech stran v různých formátech. Každou minutou vznikají miliony gigabytů nových dat. Zpracování ohromného množství dat se v průběhu posledních pár let stalo pro konkurenceschopnou firmu každodenní záležitostí. Klasické relační databázové systémy na tyto problémy nejsou schopné reagovat v přijatelném čase za přijatelnou cenu, což vedlo k nástupu nového přístupu ke zpracování dat a vývoji nových technologií, které jsou schopné pojmout mnohonásobně větší objem dat. Tyto technologie jsou již natolik vyspělé, že větším problémem se stává všechna data efektivně využít a analyzovat, než je uložit a zpracovat. Nejznámější Framework nabízející možnost zpracování a analýzy velkého množství dat je Apache Hadoop.

Trendy v analýze dat se také vyvíjejí velkou rychlostí a od klasického reportingu, kdy pomocí čísel a grafů vyjadřujeme dění v minulosti se posunuli až k predikci toho, co nastane v budoucnosti. Dalším trendem, který se dostává do popředí je zpracování a analýza dat v reálném čase. Například pro oddělení risku nebo detekce podvodů tato schopnost znamená obrovský krok vpřed.

Důkazem těchto trendů je rostoucí popularita IT pozice zvané Data Scientist. Datový vědec patří k nejlépe vydělávající pozici v oblasti informačních technologií. Má za úkol analyzovat firemní data a přicházet s výstupy, které pomáhají strategickému i operativnímu rozhodování firmy. Na cestě ke splnění tohoto úkolu musí překonat hned několik překážek. Ačkoliv to zní paradoxně, velkou překážkou bývá často sama firma, která datové vědce platí. Ti potřebují pro svou práci příslušné prostředky a prostředí. Založení prostředí pro práci je poměrně snadný proces, který lze vyřídit za pár minut, nicméně v korporátních podnicích tento proces může zabrat i několik týdnů. Vše musí být schváleno v mnoha různých systémech, čímž je ztraceno velké množství času.

V drtivé většině případů jsou požadavky na Hadoop administrátora stejné, jen málokdy se liší v maličkostech. Automatizace procesů spojených s požadavky uživatelů ušetří mnoho času jak analytikům, tak hlavně administrátorům, kteří je vyřizují.

1.1 Cíle práce

Cílem teoretické části je představit čtenáři pojem Big Data, seznámit ho s Frameworkem Hadoop, jeho hlavními komponentami, funkcionalitami a možnostmi jeho zabezpečení. Jsou vyjmenovány a detailně popsány komponenty integrované do prostředí Hadoop platformy. Přínosem teoretické části je znalostní příprava čtenáře za účelem lepší orientace v pojmech použitých v praktické části. Pro splnění tohoto cíle byly definovány dílčí cíle:

- Uvést do problematiky Big Data
- Seznámit s Frameworkem Apache Hadoop
- Vyjmenovat a popsat hlavní komponenty Frameworku

- Vyjmenovat a popsat další komponenty Frameworku, se kterými se bude pracovat v praktické části této práce
- Popsat zásady zabezpečení platformy, se kterým se bude pracovat v praktické části této práce

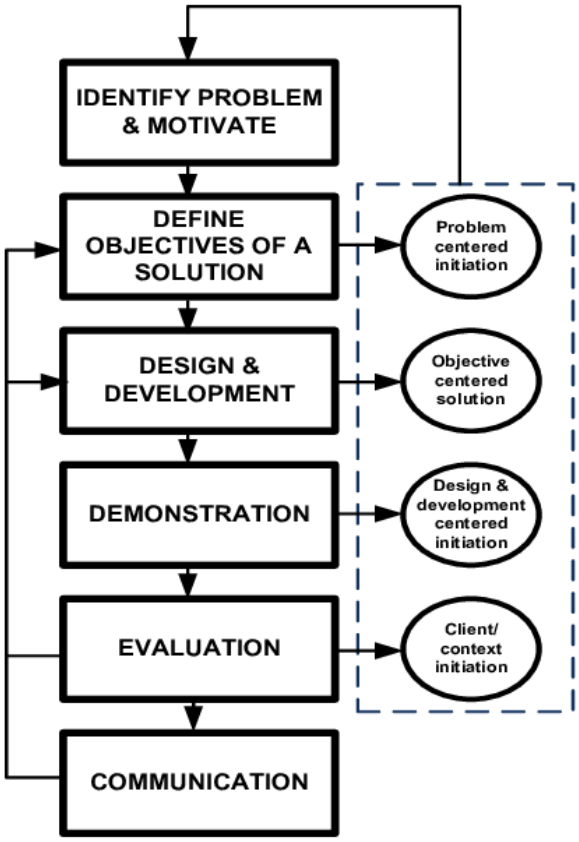
Hlavním cílem praktické části práce je automatizace procesů user managementu, které jsou opakovaně prováděny na zabezpečené Hadoop platformě. Automatizace těchto procesů usnadní práci Hadoop administrátorům a sníží čas potřebný na přípravu pracovního prostředí pro datové analytiky.

Pro dosažení hlavního cíle byly stanoveny dílčí cíle:

- Popsat prostředí, na kterém bude automatizace procesů probíhat
- Definovat základní user management procesy, které jsou pro automatizaci vhodné
- Tyto procesy detailně popsat
- Navrhnout business řešení
- Navrhnout technické řešení
- Implementace navrženého řešení do popsaného prostředí
- Ověření řešení

1.2 Metoda dosažení určených cílů

Pro dosažení cílů této práce byla využita vědecká metoda Design Science Research Methodology (Peffers, a další, 2008). Metoda se skládá z šesti posloupných kroků, jejichž plnění vede k vytyčeným cílům. Všechny kroky jsou vidět z následujícího obrázku:



Obrázek 1 - Design Science Research Methodology, zdroj: (Peffers, a další, 2008)

Prvním krokem metody je definice problému a motivace jeho řešení. Druhým krokem je určení cílů řešení, po kterém následuje návrh a vývoj řešení. Poté přichází na řadu ověření funkčnosti řešení a jeho vyhodnocení. Výsledek by měl být komunikován ve vědeckých publikacích.

Výstupem metody jsou implementované procesy usnadňující správu uživatelských procesů probíhajících na zabezpečené Hadoop platformě. Prvním krokem je identifikace řešeného problému. Při detailnějším prozkoumání trhu s nástroji Big Data lze říci, že žádná z firem nabízejících Hadoop distribuci se nevěnuje automatizaci procesů user managementu na platformě. Těmito procesy Hadoop administrátoři i uživatelé žádající prostředky ztratí velké množství času, přitom jde téměř vždy o stejné požadavky řešené stejnými příkazy. Definice cílů zazněla již v předchozí kapitole (1.1) společně s vytyčením dílčích cílů pro vyřešení problému. Praktická část práce se poté zabývá návrhem a vývojem řešení (kapitola 7 a dále). Pro ověření řešení byly procesy implementovány na laboratorní cluster společnosti Adastra, kde proběhlo testování administrátory Hadoop platformy. Vyhodnocení probíhalo formou zpětné vazby a individuálního hodnocení každého z nich. Samotná práce ani její výsledek

nebyly dosud nikde publikovány, a proto poslední krok metody Data Science Research nelze považovat za splněný.

1.3 Struktura práce

Práce je rozdělena do osmi větších kapitol. První kapitola je věnována formálnímu úvodu práce, rešerši zdrojů, stanovení dílčích a hlavních cílů práce. Ve druhé kapitole je uvedeno vysvětlení samotného pojmu Big Data a jsou zmíněny specifika Big Data systémů. Třetí, čtvrtá a pátá kapitola rozebírá podrobně nejznámější Framework pro zpracování velkého množství dat Apache Hadoop. Postupně je detailně popsán Framework samotný, jsou vyjmenovány jeho hlavní komponenty zajišťující kritické funkcionality a následně i ty, které jsou spojeny s prací uživatelů na platformě.

Celá praktická část práce je popsána v kapitole sedm. Jsou zde definovány opakovaně prováděné uživatelské procesy a následně navrhnuté řešení jejich automatizace. Návrh obsahuje vizualizaci celého procesu a návrh pomocné databáze, která je pro řešení využívána. Poté je provedena implementace jednotlivých procesů dle dřívějšího návrhu. V závěru je shrnuto implementované řešení na základě zpětné vazby administrátorů, kteří řešení využívali v rámci pilotního režimu. Také se uvádí možné rozšíření a doporučení pro zlepšení.

1.4 Rešerše informačních zdrojů

V této kapitole je uvedena rešerše literatury, která se zabývá problematikou Big Data a Frameworkem Hadoop. Jsou uvedeny závěrečné práce studentů vysokých škol a následně i odborná literatura.

Před úplným začátkem psaní této práce byla provedena důkladná rešerše, jejímž účelem bylo zjistit, jestli byly již dříve zpracovávány bakalářské či diplomové práce na podobné téma a zvážit možnost čerpání informací, respektive vyhnout se stejnému tématu. I přes to, že je pojem Big Data a relativně nový pojem, existuje již celá řada prací, které se ho dotýkají.

1.4.1 Závěrečné práce

Hlavním zdrojem hledání byla databáze závěrečných prací theses, ve které lze dohledat práce ze všech vysokých škol po celé české republice.

První nalezenou prací je práce bývalého studenta Vysoké školy ekonomické Josefa Kernera na téma Hadoop a Business Intelligence z roku 2015. Ve své práci se věnuje základním principům a pojmům spojených s pojmy Big Data a Hadoop. Vybírá základní komponenty Hadoop Frameworku a uvádí jejich možné využití při integraci s klasickými datovými sklady. Praktická část je poté zaměřena na návrh možnosti využití Hadoopů ve aktuální architektuře datového skladu.

Druhou prací na velmi podobné téma je Hadoop as an Extension of the Enterprise Data Warehouse od Aleše Hejmalíčka ze stejného roku. Autor v teoretické části práce představuje zásady datových skladů a Business Intelligence a rovněž popisuje Framework Hadoop. V praktické části je poté popisována integrace Hadoop do datových skladů se zaměřením na import faktů a dimenzí.

Další prací na téma Nové trendy v Business Intelligence – Zaměření na Big Data a Hadoop zpracoval Josef Korkisch v roce 2014. Cílem práce je zprostředkovat průvodce do světa Business Intelligence a Hadoop. Opět jsou vyjmenovány hlavní komponenty Frameworku Hadoop, jedna z kapitol je poté věnována porovnání nástrojů Business Intelligence, které umí pracovat s Big Data.

Poslední a zároveň nejnovější práce byla napsána v roce 2017 Michaelou Růžičkovou, která si vybrala praktické téma– Řešení vybrané úlohy v oblasti Big Data v bankovní společnosti. Kromě Apache Hadoop je nově popisován i další Framework pro zpracování Big Data Apache Spark a jedna z jeho komponent pro strojové učení. Tuto komponentu zároveň používá ve své praktické části práce, a to konkrétně při vytvoření návrhu segmentace právnických osob na základě ukazatelů vybraných a vypočtených z finančních výkazů pomocí algoritmu K-Means.

Při pohledu na vyjmenované závěrečné prácí je vidět popularita tématu Big Data. Cíle prací jsou často velmi podobné. V roce, kdy byl Hadoop novinkou na trhu se autoři zaměřují hlavně na integraci s klasickými nástroji datového skladu, s postupem času a značným technologickým posunem jsou cíle směřované na využiti Big Data technologií v praxi pro určité případy.

1.4.2 Odborná literatura

Zdrojem vyhledávání odborné literatury byl internet, ve kterém byly hledány klíčová slova uvedená na začátku této práce. Pro vyhledávání odborných publikací byl použit webový portál Google Scholar.

Základní přehled o pojmu Big Data přináší kniha Big Data a NoSQL databáze vydaná v Praze (Holubová, a další, 2015). V knize jsou uvedeny základní principy nerelačních databází a způsoby zpracování velkého množství dat.

Ve stejný rok byla vydána i kniha Hadoop: The Definitive Guide (White, 2015), která popisuje velmi podrobně a technicky funkcionalitu Hadoop Frameworku. Ve své další části uvádí možné praktické využití Hadoop platformy.

Zajímavým zdrojem, který se zabývá zabezpečením Hadoop platformy je poté kniha s názvem Hadoop Security (Spivey, 2015). Jde se o praktickou knihu, která poskytuje Hadoop administrátorům, respektive bezpečnostním architektům návod, jak platformu efektivně chránit proti neautorizovaným přístupům a zároveň jak minimalizovat škody v případě, že se útočníkovi i tak podaří na platformu proniknout.

Pro úvod do modelování procesů organizace, které bude použito v praktické části této práce je vhodná publikace Mastering Archimate Edition III.TCI (Wierda, 2017). Poskytuje základy modelovacího jazyku Archimate, který se používá pro návrh procesů v téměř jakémkoliv odvětví organizace (ať jde o infrastrukturu nebo business) a ukazuje klíčové aspekty správného modelování.

Poslední knihou, která je v rešerší uvedena má za cíl pomoci s praktickou částí a samotnou automatizací úloh v Unix prostředí. Learning the bash Shell: Unix Shell Programming (O'Reilly, 2009) je jednou z nejprodávanějších výukových knih Linuxové příkazové řádky. Své čtenáře naučí základní práci s příkazovou řádkou, i techniky debugování vytvořených skriptů pro usnadnění a zefektivnění práce.

Z knižních zdrojů je opravdu co vybírat, a proto je potřeba vybrat pouze nejrelevantnější zdroje. Všechny zmíněné knižní zdroje poskytují pevný znalostní základ pro účely této práce.

2 Big Data

První kapitola se věnuje obecnému vymezení pojmu Big Data a uvádí vlastnosti, které mají. Je rozebírána velikost Big Data, otázku, kdy jde o Big Data a kdy ne, nelze odpovědět konkrétním číslem, pro každého může tento pojem znamenat něco jiného. V poslední části kapitoly je naznačeno, jak se s velkým množstvím dat zachází, a jak zajistit, aby obrovské datové objemy bylo možné analyzovat v přijatelném čase.

2.1 Definice pojmu

V této kapitole je sepsán obecný úvod do problematiky Big Data. Jsou představeny výhody Big Data systémů, jeho vlastnosti a je přiblíženo, jak velká Big Data vlastně jsou. Dále je detailněji rozveden jeden z nejznámějších Big Data Frameworků, který je důležitý pro praktickou část této práce.

V dnešní době se mnoho firem potýká s problémem velkých dat. Využívání obrovského množství dat je pro konkurenceschopnou firmu již povinnost. Obecně řečeno, jde o obrovské množství dat, které přesahuje možnosti běžně používaných softwarových nástrojů pro sběr, ukládání a zpracování dat v krátkém časovém období. Do relačních databází lze uložit pouze data strukturovaná, v oblasti Big Data lze již uložit i nestrukturovaná data. Zatímco samotný pojem Big Data je relativně nový, proces sbírání a ukládání dat pro následné analýzy žádnou novinkou není. Tento koncept však dostal nové rozměry na začátku 21. století, kdy analytik Doug Laney ze společnosti Gartner zformuloval dnes již notoricky známou definici Big Data jako tři písmena „V“ (odvozené z anglických slov). Definuje Big Data jako data, jejichž velikost (anglicky volume), rychlost nárůstu (anglicky velocity) a různorodost (variety) dosud známé a ověřené technologie nestíhají zpracovávat. První V – Volume značí velikost dat. (Objem dat vskutku narůstá i díky dnešnímu fenoménu IoT[1], kdy jsou data produkovány některými stroji (například senzory), které jsou schopné data posílat i každou vteřinu. S tím souvisí i druhé V – Velocity neboli rychlost nárůstu. Za posledních 5 let lidstvo vyprodukovalo více dat než za celou svoji předchozí historii. Pod posledním V z definice Gartneru se skrývá různorodost. Zdrojová data mohou pocházet z vícero zdrojů v různých formách. Nestrukturovaná data se jen těžko ukládají do relačních databázových systémů. (Gartner, 2013)

S přibývajícími zkušenostmi s Big Data se začali do této definice přidávat i jiné pojmy a dnes můžeme najít i definice, které uvádějí až deset vlastností Big Data, začínajících na písmeno V. Jedním z takových pojmů je věrohodnost (veracity) nebo přidaná hodnota (value). Takovéto hesla mohou vycházet i z marketingu a každá firma si pojem Big Data definuje po svém. Nicméně za zmínku stojí určitě poslední uvedený pojem hodnota. Každá investice má mít i svou návratnost. Hodnota dat je to, co mění velká data na velké peníze a

[1] Internet of Things

skladovat terabajty zbytečných dat není vhodná strategie žádného podniku. Naopak při troše kreativity a s dobrým nápadem lze vytěžit cenné informace i z menšího množství dat. (*The V's of Big Data*, 2014)

Ještě před pár lety byl pojem Big Data označován jako buzzword[2], vzhledem k exponenciálnímu růstu objemu dat jsou však nástroje pro jejich zpracování jedny z nejrychleji rozvíjejících se technologií na světě. Tato oblast informatiky se pomalu, ale jistě stává každodenním problémem, který musejí firmy řešit. Technologický pokrok umožnil zpracování mnohonásobného množství dat, problémem tak spíše začíná být, jak všechna tato data využít, respektive využít tak, aby to nebylo v rozporu s GDPR[3], případně jinými regulacemi ohledně využití osobních dat. (*Dutcher*, 2014)

2.1.1 Velikost Big Data

Velikost dat lze objektivně měřit. V současné době lze pořídit pevné disky o velikosti až několik terabytů[4] (TB). Odhaduje se, že v roce 2020 bude 6 miliard lidí na zemi vlastnit mobilní telefon a denně vyprodukují 2,8 kvintiliónů (2,8 x 10^{18} bytů) dat. Odhady celkového objemu dat se pohybují okolo 40–44 zettabytů (44 x 10^{21} bytů). Pro lepší představu takové množství dat je ekvivalent miliardy pevných disků o velikosti 1 TB. Dalším dobrým příkladem pro lepší pochopení jsou sociální sítě. V roce 2015 měla sociální síť Twitter zhruba 200 miliónů uživatelů, kteří každý měsíc vyprodukují 400 milionů „twítů" (krátkých komentářů). Další ze známých komunikačních kanálů Facebook má aktuálně kolem 1,5 miliardy uživatelů, kteří denně průměrně sdílí nějaký obsah až 4,75 miliardy krát a nahrají přes 300 miliónů fotek. Pro účely uložení a zpracování těchto dat disponoval Facebook v roce 2008 sítí o velikost 10 tisíců serverů, přičemž odhady hovoří o zhruba 180 tisících serverů za rok 2012 – tedy 18krát více serverů za pouhé 4 roky.
Takovéto objemy dat jsou extrémní, a ne každá firma generuje takové množství. Při pohledu na situaci v české republice vedou v generování velkého množství dat hlavně mobilní operátoři nebo banky, které evidují veškeré provedené transakce, ať již jde o výběry z bankomatu nebo platbu kartou. V obou případech se však jedná o řádově desítky fyzických serverů, které jsou schopné uložit řádově stovky TB.
Data objektivně změřit lze, určit však, kdy se jedná o velká data a kdy, ne je již složitější a záleží na subjektivním vnímání. V uvedených případech Facebooku nebo Twitteru asi nelze pochybovat, nicméně i při skromnějším objemu uchovávaných dat k nim lze přistupovat jako k velkým datům a využívat příslušné nástroje. Naopak relační databázové systémy také nezůstávají pozadu a lze na jejich příslušný hardware uložit desítky TB dat. Otázkou však poté je, jestli se to finančně vyplatí, jelikož nástroje pro velká data jsou designované tak, aby běžela na komoditním[5] hardwaru, jehož cena je nižší. (*Facebook na konci roku zastavil pokles uživatelů*, 2019)

2.1.2 Big Data systémy

Pro zpracování takto velkých dat bylo třeba přijít s řešením, které si dokáže s takovým množstvím poradit v přijatelném čase. Odpovědí na tento úkol je takzvaný Cluster

[2] Buzzword – slovo nebo fráze, která se za krátký časový úsek stala populární
[3] GDPR – právní rámec ochrany osobních údajů
[4] Terabyte - 10^{12} bajtů (B)
[5] Běžně dostupném

Computing[6]. Cluster je množina vzájemně propojených serverů, kterým se v Big Data terminologii říká node nebo česky uzel. Díky většímu množství nodů v jednom clusteru lze využít paralelizaci a distribuované zpracování. Vykonávaná úloha není závislá na výpočetním výkonu jednoho serveru. Je rozdělena do několika menších částí, které mohou probíhat paralelně na jiných nodech v clusteru. Zátěž je rozložena na několik dílů. Díky tomu lze operovat i s opravdu velkým množství dat v přijatelném čase. Hlavní silou clusteru je snadná škálovatelnost. V případě nedostatečného výpočetního výkonu, respektive úložiště, lze cluster jednoduše rozšířit zakoupením dalšího nodu. (Holubová a další, 2015)

Big Data systémy mají zpravidla následující vlastnosti:

- Používají lokální úložiště
- Je vybírán komoditní hardware
- Všechny služby jsou open source
- Na nodech v clusteru běží Linux distribuce – převážně Centos[7]
- Disponují Big Data frameworkem

Framework – v softwarovém inženýrství je aplikační rámec, jenž má za cíl podporu vývojářů při vývoji něčeho, co již bylo dříve vyvinuto. Mezi frameworky zabývajících se Big Data problematikou se řadí Apache Hadoop, Spark, Flink nebo Storm. Další kapitola je věnována právě prvnímu zmíněnému frameworku Apache Hadoop.

2.2 Shrnutí

Big Data technologie přináší nové přístupy ke zpracování dat. Čím dál více firem se s nimi setkává. Objem, rychlost nárůstu a různorodost jsou vlastnosti, které Big Data charakterizují. Byla diskutována otázka, jak velká jsou Big Data. Na tuto otázku neexistuje přesná odpověď, jedná se o subjektivní pohled na problematiku. V poslední části byly uvedeny charakteristiky Big Data systémů a jejich typické rysy.

[6] Shlukování více serverů, které se chovají jako jeden
[7] Volně dostupná Linuxová distribuce

3 Apache Hadoop

Tato kapitola se zaměří na Framework Apache Hadoop, který vznikl poměrně nedávno. Nejprve je vysvětleno, co vlastně Hadoop je a následně jsou představeny jeho hlavní části. Poté je probíráno zařazení do podnikové infrastruktury a nastíněno možné využití Hadoop platforem v praxi.

Poslední část kapitoly zmiňuje nejznámější Hadoop distribuce dostupné na trhu a poté je podrobněji popsána distribuce od firmy Cloudera.

Nejznámějším Frameworkem pro zpracování velkého množství strukturovaných a nestrukturovaných dat je Apache Hadoop. Apache Hadoop (dále jen Hadoop), jak již název napovídá byl vyvinut firmou Apache a jeho první oficiální datum vydání je uveden desátý prosinec roku 2011. Jedná se o open source projekt, jehož cílem je vyvinout software pro spolehlivé, škálovatelné a zároveň distribuované výpočty. (*Holubová a další*, 2015)

Definice hadoopu z oficiálních stránek projektu uvádí:

„*Apache Hadoop software je framework, který umožňuje distribuované zpracování velkých datových objemů přes clustery počítačů pomocí jednoduchých programovacích modelů. Je navržen tak, aby ho bylo možné jednoduše škálovat až na tisíce strojů, přičemž každý z nich nabízí svůj výpočetní výkon a úložiště. Zároveň nespoléhá pouze na hardwarové vybavení pro poskytování vysoké dostupnosti, samotný framework je navržen tak, aby detekovaa a zpracovával selhání na aplikační vrstvě, a tak poskytoval vysoce dostupnou službu, která je odolná vůči jakémukoliv hardwarovému selhání.*“ *(Hadoop* , 2019)

Projekt obsahuje následující komponenty:

- Hadoop Common – společné služby, které jsou využívány v ostatních modulech
- Hadoop Distributed File System – Distribuovaný souborový systém dostupný ze všech nodů v clusteru. Poskytuje úložiště aplikačních dat.
- Hadoop YARN – Framework pro časování úloh a rozdělování zdrojů celého clusteru
- Hadoop MapReduce – Programovací model pro paralelní zpracování velkého objemu dat.
- Hadoop Ozone – Nejnovější modul Hadoop projektu. Jde o úložiště dat ve formátu klíč-hodnota.

S projektem Hadoop jsou dále spojovány i jiné projekty. Pro tuto diplomovou práci jsou důležité jen některé z nich, které budou v dalších kapitolách rozebírány. (*Hadoop*, 2019)

- Ambari – Webová aplikace pro zajištění chodu a monitorování Hadoop clusteru. Veškeré aplikace běžící na clusteru jsou vizuálně znázorněny a je možno je řídit.
- Hue – Obdoba Ambari
- Avro – Systém serializace dat – datový formát pro uložení na Hadoop

- Cassandra – NoSQL databáze navržená pro provoz na celém clusteru.
- Chukwa – Systém pro monitorování velkých distribuovaných systémů
- HBase – Škálovatelná, distribuovaná databáze, která podporuje ukládání strukturovaných dat pro obrovské tabulky.
- Hive – Aplikace datového skladu poskytující SQL dotazy nad daty z Hadoopu.
- Mahout – Machine learning a data mining knihovna
- Pig – Platforma pro analýzu velkých datových sad.
- Spark – Výpočetní engine pro data uložená na Hadoopu. Poskytuje širokou škálu možností od tvorbu ETL až po machine learning nebo streamování dat.
- Tez – Programovací framework pro tvorbu datových toků postavený na Hadoop YARN.
- ZooKeeper – Nástroj pro koordinaci distribuovaných aplikací. (*Hadoop*, 2019)

Moduly důležité pro tuto práci jsou zmíněny v dalších kapitolách.

3.1 Místo Apache Hadoop v podnikové infrastruktuře

Ve většině podnicích existuje již zaběhnutý systém sběru a správy firemních dat, postavený na relačních databázových systémech. Je zcela jasné, že nahrazování tohoto systému by bylo velice náročné jak časově, tak finančně. Nahrazení relačních databázových systémů zároveň vůbec není účel, za kterým byl Hadoop vyvinut. Spíše, než nahrazení je Hadoop využíván jako doplněk relačních databázových systémů, jejichž kapacita a výkon jsou draze zaplaceny. Při aktuální nabídce relačních databázových systémů (například Teradata, Oracle nebo Microsoft) vychází vybudování a správa Hadoop clusteru se stejnou kapacitou několikanásobně levněji. (*How Hadoop is Used in Organizations*, 2016)

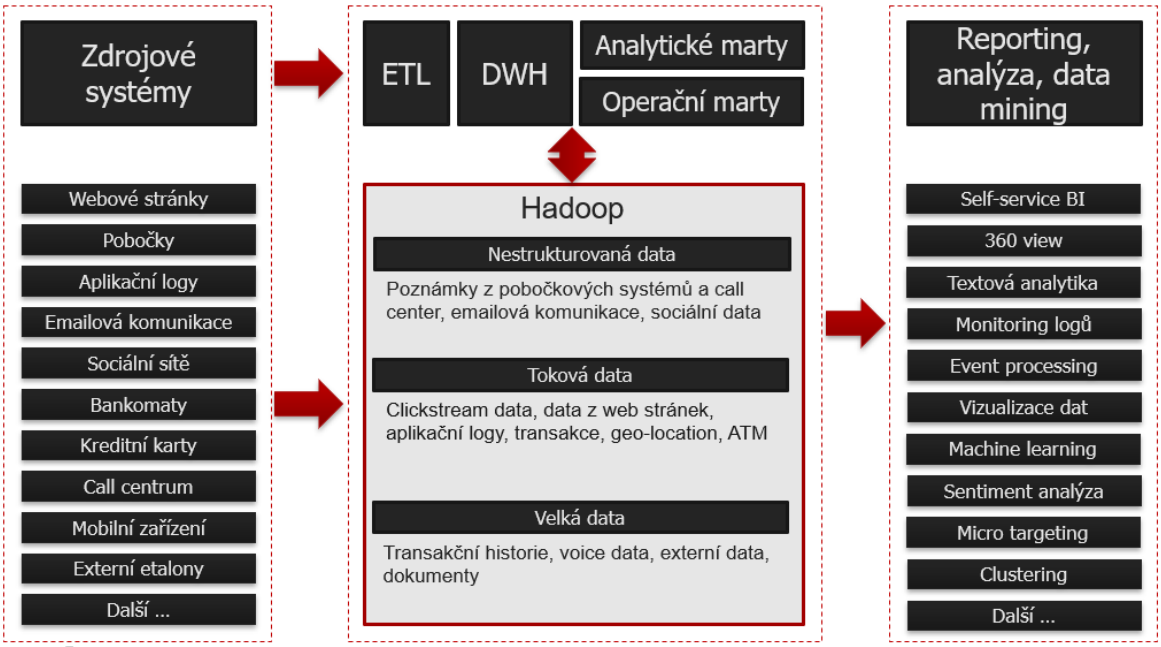
Nejčastěji bývá Hadoop využíván jako:

- Rozšíření fungujícího datového skladu – díky velkému úložišti dat Hadoop clusteru lze udržovat historii dat až o několik let déle
- Analytická platforma – široká škála analytických služeb kompatibilních s Hadoopem představuje silnou analytickou platformu, na které mohou operovat datoví vědci a tvořit přesnější analýzy na základě většího množství zdrojových dat

Integrace dat mezi Hadoopem a datovým skladem je oboustranný proces. Proto je možné Hadoop využívat i jako takzvané operativní úložiště v případě nutnosti rychlých reportů či analýz. Data jsou jednoduše přenesena z datového skladu na Hadoop, kde je vykonána transformace či analýza a následně vrácena do datového skladu napojeného na ověřené nástroje Bussiness Inteligence[8] pro potřeby koncových uživatelů, kteří jsou již na tyto nástroje zvyklí. (*How Hadoop is Used in Organizations*, 2016)

[8] Business Inteligence – souhrn analytických a vykazovacích podnikových nástrojů

Využití Hadoopu jako analytické platformy demonstruje následující obrázek:



Obrázek 2 - Integrace Hadoopu do infrastruktury podniku, zdroj: (*Adastra*, 2017)

V levé části obrázku jsou zobrazeny možná zdrojová data. Zatímco do tradičních datových skladů lze uložit pouze strukturovaná data ze zdrojových systémů, na Hadoop platformu lze ukládat i data nestrukturovaná. Kromě dat z webových stránek nebo poboček to může být například i emailová komunikace, data z bankomatů nebo záznamy hovorů z call centra. Přenos dat se řeší pomocí dávek nebo tokových dat (takzvaný stream). Zatímco dávkové zpracování dat je spouštěno v určitý čas, toková data umožňují data zpracovávat ihned v reálném čase. To je ideální například v případě sledování a monitorování podezřelých transakcí. (*Adastra*, 2017)

Velmi často řešenou úlohou bývá právě přenos dat mezi datovým skladem a Hadoop platformou. Pro tyto účely byl vyvinut další nástroj, který je nazývá Sqoop. Pomocí jednoduchých příkazů lze přenášet jedním či druhým směrem celé databáze, případně jednotlivé tabulky. Tento nástroj se však netýká user management procesů, a tudíž nebude dále zmiňován. (*Apache Sqoop* , 2019)

3.2 Hadoop distribuce

Apache Hadoop se skládá z velkého množství komponent. Udržet všechny komponenty funkční a správně nakonfigurované je velmi složité. Na základě toho vznikly postupem času společnosti, které se poskytují jednotné softwarové balíčky právě pro Apache Hadoop. Takový balíček se nazývá distribuce. Mezi největší společnosti nabízející Hadoop distribuce

se řadí Cloudera[9], Hortonworks[10] a MapR Technologies[11]. Cloudera se však v aktuálním roce (2019) spojila s Hortonworks a momentálně probíhá restrukturalizace firmy. Právě Cloudera Hadoop distribuce bude dále popsána, jelikož se objevuje i v praktické části práce. (Hortonworks, 2019)

3.2.1 Cloudera

Společnost Cloudera se na trhu objevila roku 2008 a zapsala se do historie jako první komerční Startup zabývající se Hadoop Frameworkem. Hlavním obchodním modelem je nabídka hybridního open source řešení pro Hadoop, což v překladu znamená že část nabízených komponent je vlastnictvím firmy a není volně dostupná, zbytek je open-source. Cloudera využila zaváhání firmy Intel, která nezvládla dokončit vlastní distribuci Hadoopu a aktuálně je lídrem na trhu. Jak bylo zmíněno v předchozí kapitole, dokonce koupila velkého konkurenta Hortonworks. (*Cloudera* merge Hortonworks, 2019)

Nabídku Cloudera distribucí tvoří dvě úrovně. Cloudera Express a Cloudera Enterprise. Cloudera Express je ke stažení zdarma na webových stránkách Cloudery. Obsahuje CDH (Cloudera Distribution Hadoop) a Cloudera Manager s omezenou funkcionalitou. Cloudera Enterprise poté obsahuje CDH, Cloudera Manager, Cloudera Navigator a technickou podporu, která představuje ochotu společnosti Cloudera řešit veškeré problémy spojené s Hadoopem. (*CDH Cloudera*, 2019)

3.2.2 Cloudera Manager

Cloudera Manager je první ze tří produktů dodaných při zakoupení Cloudera Enterprise edice. Jde o grafické uživatelské rozhraní (GUI), které slouží k celkové správě a monitoringu Hadoop clusteru. Umožňuje administraci prostředí bez hlubších znalostí příkazové řádky. Zároveň umožňuje automatizovat deployment, instalaci, konfiguraci a distribuci nových služeb do celého clusteru. Zároveň shromažďuje logy z celé platformy a je možné snadno dohledat jakékoliv záznamy ze všech nodů. (*CDH Cloudera*, 2019)

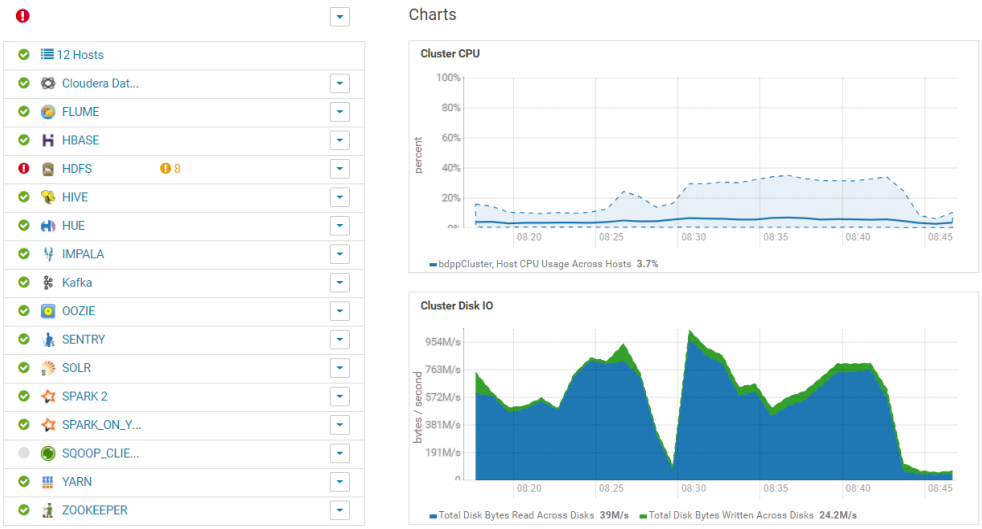
Hlavními výhodami, pro které je Cloudera Manager hojně využíván jsou:

- Možnost spouštění, zastavování a restartování služeb v clusteru (na pozadí spouští příkazy, které by si musel administrátor pamatovat a složitě spouštět z příkazové řadky na všech nodech zvlášť)
- Instalace nových služeb, případně jejich pozdější odstranění
- Monitoring celkového vytížení clusteru a jednotlivých nodů, diagnostika problémů v clusteru
- Pro přístup lze integrovat s LDAP protokolem pro řízení přístupů
- Možnost procházet soubory na HDFS a vyhledávat soubory podle názvu

[9] www.cloudera.com
[10] https://hortonworks.com/
[11] https://mapr.com/

- Zjednodušuje upgrade služeb na vyšší verze

Prostředí Cloudera Manageru ukazuje následující obrázek:

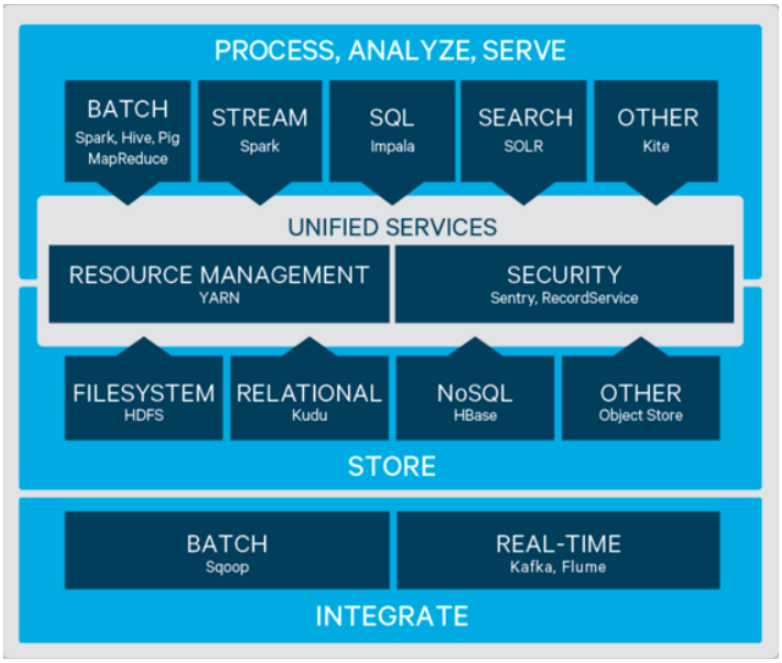


Obrázek 3 - Cloudera Manager, zdroj: (autor, 2019)

Na obrázku vidíme celkový přehled všech služeb, které jsou na clusteru nainstalované. Vpravo je poté vidět procentuální zátěž všech CPU, jimiž cluster disponuje a objem dat zapisovaných na disky v určitém časovém intervalu. Tyto dvě metriky jsou však pouze jedny z mnoha, které lze v Cloudera Manageru sledovat. Podle těchto metrik lze zpětně určit přesné dění na jednotlivých nodech. Jsou sbírané však pouze data, které produkují služby běžící v rámci Cloudera distribuce. V případě, že Cloudera Manager ukazuje stav jednoho z nodu jako dobrý, neznamená to nutně, že se na serveru nic neděje. Uživatelské aplikace spouštěné na jednotlivých nodech nejsou v Cloudera Manageru vůbec vidět.

3.2.3 Cloudera Distribution Hadoop

Cloudera Distribution Hadoop (dále jen CDH) představuje nejpoužívanější distribuci projektu Hadoop, která zároveň integruje i ostatní projekty spojené s ním. Představuje podnikové řešení, které je snadné implementovat a může ihned fungovat bez složitějších instalací. V době tvorby této práce byla nejnovější verze CDH 6.2, nicméně pro účely praktické části je využita verze CDH 5.14, jejíž hlavní komponenty zobrazuje následující obrázek:



Obrázek 4 - Komponenty CDH, zdroj: (Cloudera, 2019)

3.2.4 Cloudera Navigator

Cloudera Navigator je kompletní řešením pro Data Governance[12], audit a dalších spojených úkolů data managementu, který je plně integrovaný s Hadoop platformou. Umožňuje audit přístupu k datům, ověřování uživatelských přístupů k datům, monitoring přístupu k citlivým datům nebo analýzu frekvence přístupu k nim. Obecně lze říci, že Cloudera Navigator poskytuje rychlé odpovědi auditorům, respektive DPO[13]. (*CDH Cloudera*, 2019)

[12] celkové řízení podmínek, obsahu, dostupnosti a kvality dat v organizaci tak, aby vše bylo v souladu s potřebami procesů a s účelem organizace
[13] Data Protection Officer – osoba pověřená kontrolovat manipulaci s citlivými daty. Tato pozice vznikla na základě normy GDPR, za účelem ochrany osobních dat

3.3 Shrnutí

V této kapitole byl představen nejznámější Framework pro práci s Big Data. Byly vyjmenovány i další projekty, které mají s Hadoop projektem něco společného a následně popsány možná využití Hadoop platformy v podnikové praxi. Velká část kapitoly byla věnována společnosti Cloudera, která poskytuje distribuci Hadoop a je lídrem na trhu ve své oblasti podnikání. Byly zmíněny dvě edice, které Cloudera nabízí a hlavní součásti těchto edic. Cloudera Expres je zdarma ke stažení, avšak poskytuje pouze CDH a Cloudera Manager s omezenou funkcionalitou. Pro účely praktické části je využita edice Cloudera Enterprise s verzí CDH 6.2.

4 Hlavní komponenty Hadoop

Třetí kapitola podrobněji rozebírá části Frameworku Hadoop, které byly zmíněny v předchozí kapitole. Jsou jimi programovací model MapReduce, Hadoop Distributed File System a YARN. V rámci těchto částí je vysvětlena hlavní myšlenka Hadoopu a pojmy jako je distribuované zpracování a paralelizace. Důraz je kladený hlavně na odolnost vůči výpadkům a způsob uložení dat.

4.1 MapReduce

MapReduce je jeden ze základních a zároveň nejdůležitějších modulů Hadoop frameworku. Umožňuje uložení a přístup k datům. Základní myšlenka programovacího modelu MapReduce vychází z jednoduchého programovacího modelu „rozděl a panuj“. Může být použitý i pro lokální zpracování, jeho síla se však projeví obzvláště při distribuovaném zpracování většího množství dat. Automaticky provádí úlohy paralelně a zajišťuje odolnost vůči chybám. (*What is Map Reduce*, 2018)

Skládá se ze dvou funkcí:

- funkce Map
- funkce Reduce

Funkce Map je použita ke zpracování každého objektu ze vstupních dat, přičemž výstupem této funkce je jedna nebo více dvojic ve tvaru klíč, hodnota pro každý z objektů. Každá Map úloha dostane přidělenou část zadaného souboru. Výchozí velikost těchto částí je 64 MB, stejně jako je velikost jednoho bloku na HDFS. Celkový počet bloků, do kterého je vstupní soubor rozdělen je roven počtu Map úloh, které jsou Name nodem rozděleny na dostupné workery v clusteru. (*Holubová a další*, 2015)

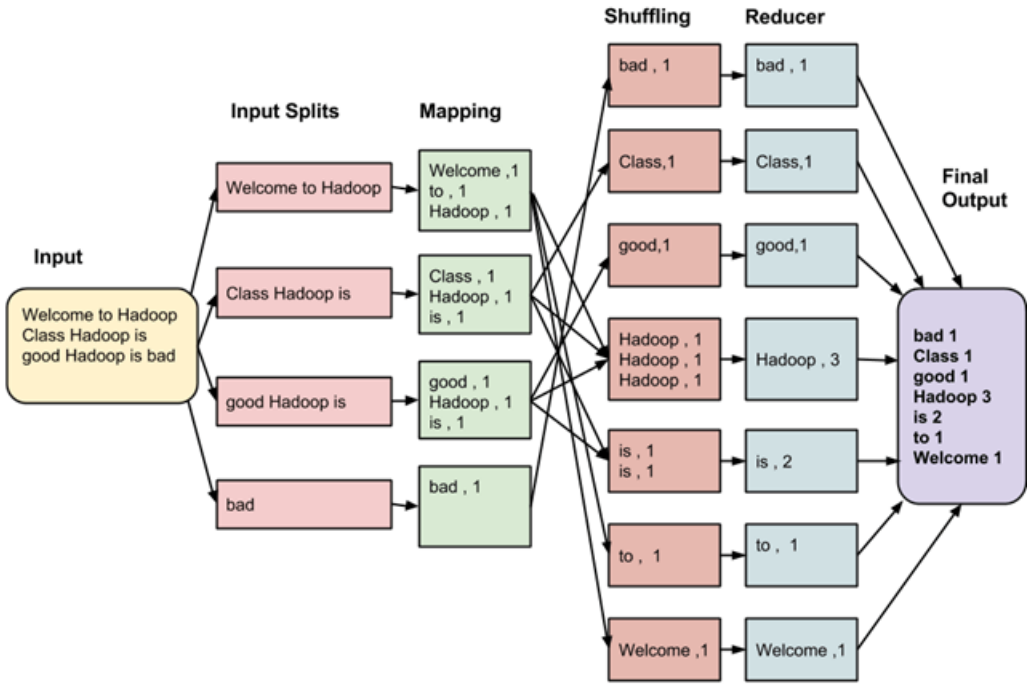
Po skončení funkce Map následuje funkce Reduce, která je rozdělena do tří fází.

Shuffle – Načte data z Map funkce ze všech workerů.

Sort – Seskupuje záznamy se stejným klíčem

Reduce – Aplikuje uživatelem definovanou reduce funkci na každý klíč a příslušnému seznamu hodnot.

Klasickým příkladem MapReduce úlohy je získání počet jednotlivých slov ze souboru. Průběh takové MapReduce úlohy je znázorněn následujícím obrázkem:



Obrázek 5 - průběh MapReduce úlohy, zdroj: (*MapReduce model*, 2015)

Na vstupu stojí soubor se třemi věty. Tento soubor je rozdělen do bloků a ty jsou následně rozděleny workerům pro zpracování. Workery vstupní data transformují do podoby (klíč hodnota). Poté dojde k seřazení a aplikaci uživatelsky definované funkci Reduce. Na výstupu opět stojí jeden výsledek v podobě (klíč hodnota), tentokrát však již s agregovanými daty, v našem případě (slovo, počet celkových výskytů). (*MapReduce model*, 2015)

Součástí MapReduce modelu může být i funkce Combine. Pokud není upravena uživatelem, pouze předá svůj vstup beze změny na výstup. Je podobná funkci Reduce, ale je prováděna pouze nad lokálními výstupy jednoho uzlu (workeru) ve fázi Map. Uvažujme příklad z obrázku 1 – pokud se slovo „Hadoop“ v dokumentu opakují, výsledkem Map fáze budou tři záznamy ve formě klíč hodnota (Hadoop, 1). Takový výstup je komprimovatelný. Funkce Combiner redukuje takovéto záznamy na dvojice (slovo, počet výskytů) – v našem případě (Hadoop, 3). Typicky je funkce Combine využívána pro redukci množství dat, které jsou předávány mezi fází Map a Reduce. (*Holubová a další*, 2015)

4.1.1 Tolerance k výpadku

Při zpracování velkého množství dat je nutné, aby měl MapReduce model zabudovaný robustní mechanismus pro řešení chyb nebo selhání hardwaru. Selhání některého z workerů, kterému byla přiřazena nějaké úloha Map nebo Reduce řeší Master node. Ten posílá pravidelně dotazy na aktuální stav. V případě, že se některý worker neozve do

stanoveného časového limitu, worker je označen jako nedostupný, všechny jeho nedokončené úlohy jako čekající a následně jsou přiřazeny jinému, dostupnému workeru.

Selhání jednoho z workerů tedy není žádný velký problém. Nastává však otázka, jak vyřešit selhání master uzlu, který řídí ostatní uzly v clusteru. Selhání master uzlu se řeší jedním ze dvou přístupů. V pesimistickém případě je zapisován stav clusteru v pravidelně opakujících se intervalech do logů (informace o dostupných uzlech, dokončených a čekajících MapReduce úlohách atd.). Při selhání master uzlu je tak možné okamžitě nastartovat novou verzi masteru, která bude pokračovat v řízení od posledních stavů zmíněných v log souborech.

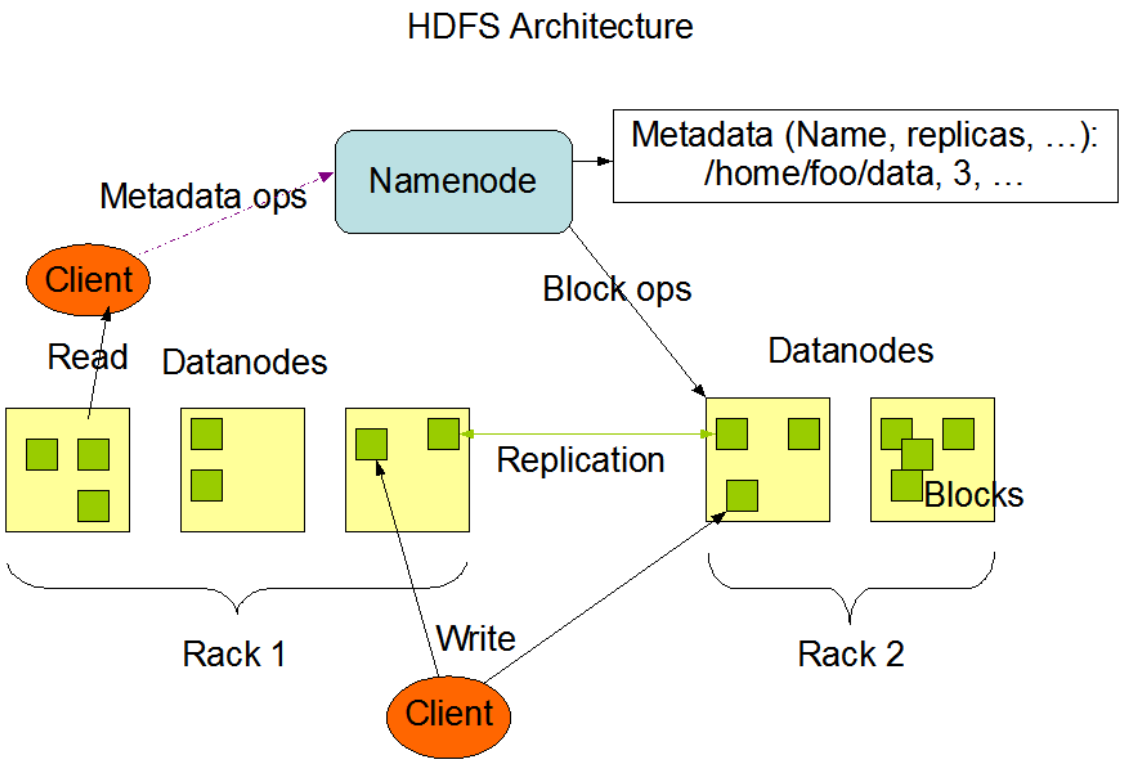
V druhém případě se předpokládá, že je master natolik spolehlivý, že je jeho selhání vysoce nepravděpodobné. Je kladen důraz na kvalitu a sílu hardwaru, na kterém master běží. Pokud však master přeci jen selže, je nutné zajistit odstávku celého clusteru a po opravě pustit znovu všechny nedokončené MapReduce úlohy. (*Holubová a další*, 2015)

4.2 Hadoop Distributed File System (HDFS)

Druhým z důležitých modulů Hadoop Frameworku je Hadoop Distributed File System. Hadoop Distributed File System (dále jen „HDFS“) je distribuovaný platformě nezávislý škálovatelný souborový systém. Má mnoho podobných rysů jako ostatní distribuované souborové systémy, nicméně je navržený tak, aby běžel na komoditním hardwaru. Hlavním rysem, podobně jako u MapReduce modelu je odolnost vůči chybám. Hardwarové selhání je považováno spíše za standard než za něco neobvyklého. Poskytuje dostupnost dat zvláště aplikacím, které pracují s obrovskými datasety (v řádech GB až TB). Splňuje i několik POSIX požadavků pro stream přístup přímo do systému. Původně byl navržen jako základ pro webový vyhledávací engine Apache Nutch. Je optimalizovaný hlavně pro dávkové zpracování, místo interaktivního uživatelského přístupu. Dalším předpokladem je, že soubor, který je jednou uložen není dále modifikován nebo alespoň není výrazně modifikován. HDFS poskytuje uživatelům jmenný prostor (Namespace), který je vytvořen hierarchií adresářů a souborů, stejně jako klasické souborové systémy. Práce s distribuovaným systémem se nápadně podobá Linuxovému souborovému systému, zásadní rozdíl je však v tom, jak je tato funkcionalita implementována.(Hadoop, 2018)

4.2.1 Architektura HDFS

HDFS je založený na master/slave architektuře. Cluster obsahuje jeden master node – tzv. NameNode, který spravuje veškerá metadata Filesystemu a reguluje přístup klientů k nim. Ostatním uzlům v clusteru, které slouží k samotnému ukládání dat se říká DataNody. Data jsou ukládány do bloků, jejichž výchozí velikost je 64 MB. Velikost jednoho bloku se dá upravit a doporučená velikost je 128 MB. V případě uložení souboru o velikosti 640 MB, je soubor rozdělen do pěti bloků. Každý DataNode pravidelně reportuje NameNodu seznam bloků, které fyzicky obsahuje. NameNode zde hraje zásadní roli, protože je jediný, kdo ví přesně, kde je jaký blok uložen a odkud bloky vzít, aby mohl být zobrazen klientem požadovaný soubor. (*HDFS Guide*, 2018)



Obrázek 6 - Architektura HDFS, zdroj: (*Hadoop*, 2018)

DataNody jsou rozděleny do racků[14].Každý z DataNodu obsahuje fyzicky jednotlivé bloky, které dohromady tvoří jeden soubor. Tyto bloky jsou replikovány a rozděleny na DataNody z jiných racků pro zajištění dostupnosti souboru i při výpadku. Výchozí hodnota replikace bloků je krát tří, nicméně tato hodnota jde upravit. Doporučená hodnota je však v tomto případě výchozí.

Rychlému zotavení po výpadku pomáhají HDFS dvě techniky:

- NameNode ukládá v pravidelných intervalech na lokální disk obraz celého systému (Checkpoint), který tvoří celý jmenný prostor, všechny metadata a další nastavení. Také drží log transakcí, které byly v souborovém systému provedeny od posledního kontrolního bodu (Checkpointu). Po neočekávaném výpadku lze obnovit velice snadno poslední známý stav. Nejprve je načten poslední uložený obraz systému a poté aplikovány operace z logu transakcí.
- Druhou technikou pro ulehčení obnovy systému po výpadku jsou tzv. heartbeaty. Jedná se o zprávu, kterou opět v pravidelném časovém intervalu posílají DataNody NameNodu a informují ho o svém momentálním stavu. V případě, že NameNode z některého DataNodu zprávu v zadaném časovém intervalu nedostane, označí bloky dat, které jsou fyzicky na daném DataNodu za nedostupné a neposílá mu

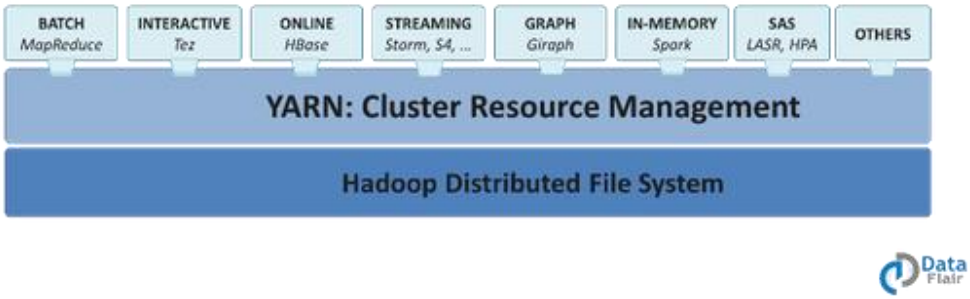
[14] Systém pro montáž a propojování jednotlivých serverových komponent do jedné perfektně uspořádané struktury

požadavky na načtení, respektive uložení dat. Vzhledem k replikaci nedojde ke ztrátě dat, protože bloky jsou uloženy ještě dvakrát na jiném DataNodu v jiném racku. Po označení DataNodu za nedostupný dojde zároveň k dodatečné replikaci nedostupných bloků na jiný, dostupný DataNode.

4.3 YARN

Zkratka YARN je zkratka z anglických slov „Yet Another Resource Negotiator“, česky „další vyjednavač zdrojů“. Jak tato zkratka napovídá, YARN v clusteru plní roli rozdělovače zdrojů. Byl představen společně s Hadoop 2.x a umožňuje různorodé zpracování dat – interaktivní zpracování, dávkové nebo stream – uložených na HDFS. Kromě rozdělování zdrojů však má i roli spouštěče pravidelných úloh, které jsou v clusteru připravené. Obecně lze říci, že YARN pomáhá rozšířit sílu Hadoopu a umožňuje ostatním aplikacím využívat jeden z nejspolehlivějších úložných systémů – HDFS. Zároveň umožňuje fungování i ostatních Frameworků pro zpracování velkých dat na stejném hardwaru, na kterém je nasazený Hadoop. Analýza a popis ostatních Frameworků není předmětem této práce, nicméně za zmínku určitě stojí hlavně Apache Spark. (*YARN*, 2016)

Zařazení do Hadoop ekosystému znázorňuje následující obrázek:



Obrázek 7 - Zařazení YARN v Hadoop, zdroj: (*YARN*, 2016)

Apache YARN je od Hadoop verze 2.0 komponentou, která zodpovídá za řízení a přidělování zdrojů celého clusteru a skládá se z hlavního (master) démona, označovaného jako Resource Manager (správce zdrojů), vedlejšího (slave) démona tzv. Node Manager (správce uzlů) a Application Master (pro každou aplikaci jeden). Dále ho tvoří Capacity Scheduler (plánovač kapacit) a Container (kontejner). (*YARN Container*, 2016)

Resource manager – Hlavní démon YARNu. Řídí globální rozdělování zdrojů (CPU a paměti) aplikacím v celém clusteru. Má dvě hlavní komponenty

- Scheduler – Je zodpovědný za přidělování zdrojů běžícím aplikacím. Neposkytuje monitoring, ani se nestará o nedokončené aplikace, které selžou. (*YARN Resource Manager* , 2012)

- Application manager – Spravuje Application mastery v celém clusteru. Zodpovídá za jejich monitoring a restart při selhání některého z nich. **Node manager –** Agent běžící na každém uzlu v clusteru. Má za úkol spravovat jednotlivé worker nody. Prakticky to znamená, že provádí monitoring na každém z nich a předává informace o něm Resource mangeru. Hlavními sledovanými metrikami jsou zdraví workeru, využití zdrojů (CPU a RAM paměti) jednotlivých kontejnerů, logování a dále. (*YARN*, 2012)

Application manager – Ke každé aplikaci je přidělen jeden Application Manager. Má za úkol řídit životní cyklus aplikace. Komunikuje s Node Managerem a obstarává zdroje pro jednotlivé aplikace. (Hortonworks, 2012)

Container – Výpočetní jednotka Frameworku YARN. Je to místo, kde se odehrávají výpočty aplikací. Každý Container má přidělené zdroje (CPU a RAM paměti). V rámci jednoho data nodu jich může běžet hned několik, nelze je však přesouvat mezi nodama. Za přidělování Containeru je zodpovědný Node Manager. (*YARN Container*, 2018)

Capacity Scheduler – Komponenta přidaná až do Hadoop verze 2.0, která zajišťuje garantovanou dostupnost zdrojů pro určitou (definovanou) skupinu uživatelů. V rámci Capacity Scheduler komponenty běží i Fair Scheduler, který zase zajišťuje, že v rámci definované skupiny jsou garantované zdroje rozděleny rovnoměrně. Později byla přidána i funkce Preemption, která má za úkol uvolnit kontejnery běžících aplikací, v případě, že je dostupnost zdrojů je menší než garantovaná hodnota definovaná v Capacity Scheduleru. (*YARN Capacity Scheduler*, 2017)

4.4 Shrnutí

V této kapitole byly podrobněji popsány hlavní komponenty Frameworku Apache Hadoop. Byl popsaný proces přístupu k datům pomocí programovacího modelu MapReduce, jeho hlavní silné stránky, jako jsou paralelní zpracování a odolnost vůči výpadkům, které jsou považovány za standard, se kterým se musí počítat. Stejné vlastnosti má i distribuovaný souborový systém HDFS, který rozděluje data do bloků. Jednotlivé bloky jsou v clusteru replikovány 3x, aby byla zajištěna dostupnost při výpadku některého z data nodů.

O rozdělování dostupných prostředků do clusteru se stará YARN, jehož hlavní části byly popsány v poslední části této kapitoly. Ten je od Hadoop verze 2.0 zodpovědný za rozdělování zdrojů, což z něj dělá jednu z nejdůležitějších komponent, bez které nemůže Hadoop Framework fungovat.

5 Ostatní komponenty Hadoop

Následující kapitola uvádí ostatní komponenty takzvaného Hadoop ekosystému, které jsou velmi důležité z pohledu práce uživatel na platformě. Tyto komponenty mají za cíl usnadnit a zpříjemnit uživateli práci. Všechny tyto služby jsou open-source a na jejich dalším vývoji se podílejí nejlepší vývojáři z celého světa. Zároveň se tyto komponenty vyvíjí závratnou rychlostí a jejich nové verze jsou vydávány ob pár měsíců.

5.1 HIVE

Apache Hive je systém datového skladu pro Apache Hadoop Framework. Umožňuje dotazování nad daty uložených v HDFS pomocí syntaxe HiveQL, což je dotazovací jazyk nápadně se podobající SQL (využívá SQL-92 standard). Hlavní silou služby Hive je právě možnost tvorby každodenních analýz nad obrovským množstvím dat bez znalosti MapReduce frameworku nebo programování. Hlavní přidanou hodnotou je překlad HiveQL dotazů do MapReduce, Apache Tez nebo Spark úloh. Každá z těchto úloh poté může být spuštěna přes YARN. Hive podporuje dotazování nejen nad daty uložené v HDFS, ale je kompatibilní i s populárními souborovými systémy, jako je například Amazon S3 nebo Alluxio. Data ukládá v různých datových typech, který lze specifikovat při DDL[15] dotazu. (*Bansal a další*, 2015)

Podporuje ukládání dat do následujících datových formátů:

- Textový soubor – je výchozím formátem pro ukládání. Také je často preferovaným formátem. Data jsou uložena do řádků, kde každý řádek je jeden záznam.
- Sekvenční soubor – speciální soubor „ploché“ struktury pro Hadoop, který ukládá hodnoty v binární formě klíč-hodnota a je možné ho dělit do části. Jeho hlavní výhodou je možnost spojit několik sekvenčních souborů do jednoho.
- RC soubor (Row Columnar File) – Řádkovo-sloupcový formát souboru. Nabízí vysokou míru komprese na úrovní jednoho řádku. Stejně jako sekvenční soubor ukládá data formou klíč-hodnota a je dobré ho používat v případě, že je nutné zpracovávat více řádků najednou.
- AVRO soubor – AVRO je samotný open-source projekt, který zajišťuje serializaci[16] a výměnu dat v rámci Hadoop. Je možné posílat data z Hadoopu do aplikace napsané v jakémkoliv programovacím jazyce. Jedná se o jeden z velmi oblíbených formátů.

[15] Data Definition Language – SQL příkazy pro definici nových objektů v databázi
[16] Převedení datové struktury na posloupnost bitů, kterou lze uložit na nějaké úložiště, případně přenést do sítě a následně opět rekonstruovat

- Parquet – Sloupcový formát. Řádky ukládá po sloupcích. Umožňuje efektivní kompresi dat a urychluje dotazování na data. (*Hive formats* , 2019)

Databáze a tabulky jsou ukládány na HDFS a jejich metadata do Hive Metastore. Hive Metastore je relační databáze (většinou MySQL), která musí být pro cluster dostupná. Hive do ní ukládá data o tabulkách, například místo uložení, definici, název nebo vlastníka. Díky Metastoru lze s tabulkami pracovat jako s relačními tabulkami. Hive umožňuje vytvářet dva typy tabulek – externí a interní. Při zakládání jsou v obou případech uložena metadata tabulky (schéma, definice atd.). Tabulka je fyzicky uložena na HDFS do složky /user/hive/warehouse (dá se změnit při definici tabulky). Rozdíl mezi tabulkami je vidět při jejich mazání. V případě mazání interní tabulky dochází ke smazání všech metadat z Metastoru a zároveň všech dat ze složky /user/hive/warehouse. Oproti tomu, když je mazána externí tabulka, jsou smazána všechna metadata z Metastoru, data však zůstávají ve složce dále, nicméně Hive je již nevnímá jako tabulku. (*Hive Metastore*, 2016)

Hive je určený hlavně pro dávkové zpracování. Není vhodný pro real-time[17] zpracování dat, protože používá full-table-scan[18], proto v případě menších dotazů bude výkon stejný nebo horší než u komplexních dotazů. V roce 2013 byla založena iniciativa Stinger založená tehdy ještě samostatnou firmou Hortonworks, jejímž cílem bylo výrazné zrychlení Hive. Ten podporuje překlad do MapReduce úloh, které mohou být časově poměrně náročné. Založená iniciativa znamenala přímý souboj dvou největších firem nabízející Hadoop distribuci, jelikož společnost Cloudera roku 2012 představila svůj produkt Apache Impala, který měl za cíl odstranit hlavní nedostatek Hive – pomalu vykonávané analytické dotazy. (*Hive Metastore,* 2016)

5.2 Impala

Apache Impala je Engine pro datovou analýzu nad daty z HDFS pomocí standardního dotazovacího jazyku SQL. Kromě dat z HDFS však umí pracovat i s daty uloženými v HBase (viz. další kapitola). Byla vytvořena roku 2012 společností Cloudera. Je vhodná pro analýzu velkých datových objemů. Byla navržen tak, aby odstranila nedostatky v té době dostupných nástrojů pro analýzy. Cílem bylo snížit dobu odezvy a zároveň se přiblížit uživatelské přívětivosti, který poskytují relační databázové systémy a o který se úspěšně snaží i Hive. Dále se snaží umožnit práci souběžně většímu počtu uživatelů, rozdělování zdrojů pro výpočty řídí YARN. Impala se snaží co nejvíce operací provádět lokálně (mezivýsledky jsou ukládány na disk), a její výkon se ještě zvýší, v případě, že pracuje se soubory optimalizovanými pro čtení po sloupcích (viz. kapitola 5.1 - podporované datové formáty). Přináší spolehlivou a škálovatelnou databázovou technologii s nízkou latencí. Impala verze 1.0 byla vydána na jaře roku 2013. Jak již bylo zmíněno, technologie zabývající se zpracováním velkého množství dat jsou jedny z nejrychleji rozvíjejících se, a tak za několik měsíců po vydání byly vydány verze 1.2, 1.3 a 1.4, které přinesly mnoho vylepšení, jež z Impaly udělaly kvalitní uživatelský nástroj. Představeny byly nové možnosti pro správu

[17] Data zpracovávána v reálném čase
[18] Sekvenční čtení všech řádků v tabulce

uložených dat, vylepšení všech dílčích komponent systému a dotazovací jazyk byl rozšířen o další funkcionalitu. (*What new in Impala*, 2019)

Ve vývoji Impaly je významný rok 2014, kdy byla představena verze 2.0. Rozšířeno bylo ukládání mezivýsledků na disk (takzvaný disk spilling) a byly vylepšeny vnořené SQL dotazy. Přidány byly některé statistické funkce a v neposlední řadě proběhlo rozšíření možnosti řízení přístupu uživatelů.

Nejnovější verze je aktuálně (v době psaní této práce) 3.2, které se soustředí na vylepšení výkonu metadat. Je prováděna jejich automatická synchronizace z Hive Metastore. Dále umožňuje nastavení výchozího souborového formátu a nově je podporována klauzule *if not exists* do příkazu *alter table*. (*What new in Impala* , 2019)

Ačkoliv byl prvotní vývoj Impaly prováděn výhradně firmou Cloudera, momentálně se na vývoji podílí open-source komunita a Impalu nabízejí nyní již pouze dvě (kvůli spojeni Hortonworks a Cloudery) společnosti nabízející Hadoop distribuci – MapR a Cloudera.

Apache Impala je kompatibilní se standardem SQL-92 a zároveň je téměř totožná s HiveQL. Oproti relačním databázovým systémům, Impala nepodporuje vytváření primárních a cizích klíčů, unikátních záznamů, indexů, auto inkrementů a povinných sloupců. Podporuje následující datové typy (*Cloudera*, 2019):

- Celá čísla – INT, TYNYINT, SMALLINT, BIGINT
- Textové – VARCHAR, CHAR, STRING
- Desetinné – FLOAT, DOUBLE
- Booleovské – BOLEAN
- Komplexní – ARRAY, MAP, STRUCT
- Časové – TIMESTAMP

5.3 HBASE

Apache HBase představuje alternativu pro uložení dat v clusteru k HDFS. Jedná se o distribuovanou NoSQL databázi, která ukládá data formou key-value. Sama je však postavena nad HDFS. Je kompatibilní s Impalou a může pro ni sloužit jako zdroj dat. HBase je určena pro rychlé vyhledávání položek na základě definovaného klíče. V případě, že jí bude Impala využívat, je vhodné konstruovat dotazy s klauzulí *where,* kde v podmínce je uvedena právě hodnota klíče. Výhodou HBase oproti HDFS je vkládání hodnot příkazem *INSERT,* který je zde rozdíl od samotného HDFS vhodný a umožňuje data i měnit, v případě že je v příkazu uveden existující klíč. Oproti Hive, který je určen hlavně pro dávkové zpracování je HBase ideální volbou pro real-time zpracování dat, kde se záznamy rychle mění. (*Hbase,* 2019)

5.4 Hue

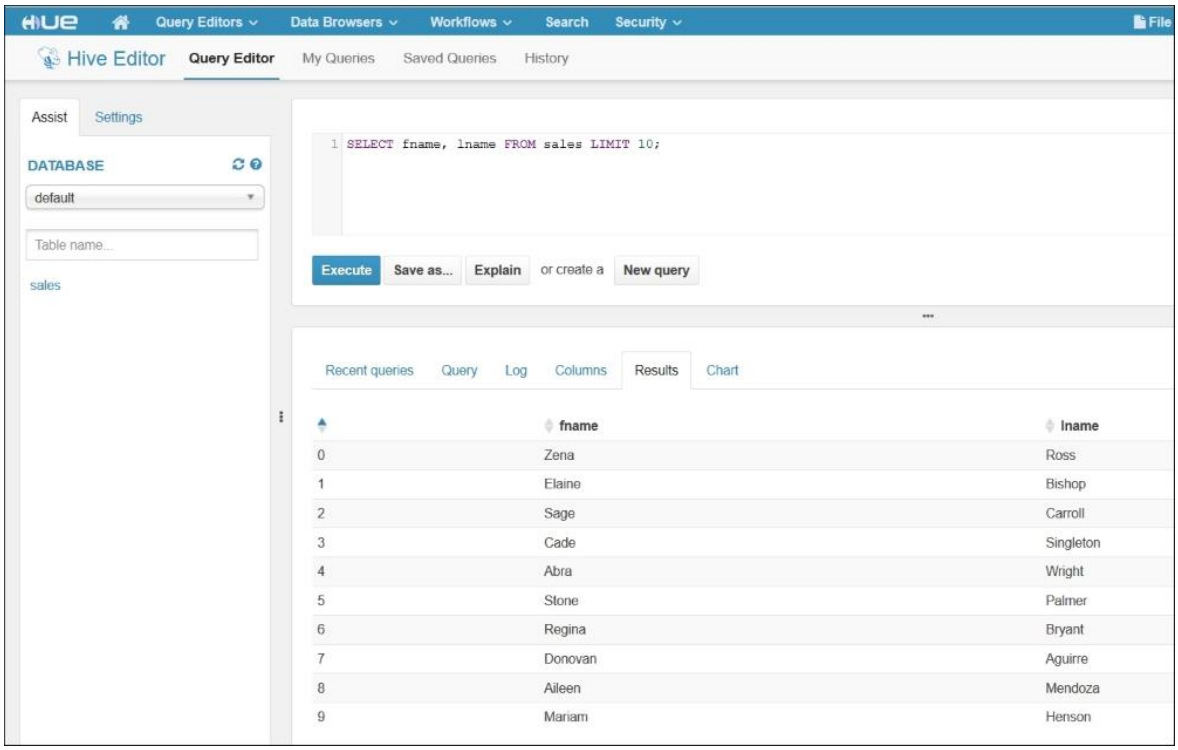
Hue – zkratka anglického „Hadoop User Experience“ – je open source webové rozhraní pro analýzu dat pomocí Hadoop dalších komponent patřících do Hadoop ekosystému. Je snadno konfigurovatelné a umožňuje spojení hned několika z nich do jedné graficky a uživatelsky přívětivé webové aplikace. Hlavním cílem Hue je umožnit uživatelům snadné použití potenciálu Hadoop služeb bez starostí o složitost a bez použití příkazové řádky.

Do hue lze připojit následující nativní Hadoop služby:

- Prohlížeč HDFS
- Prohlížeč běžících MapReduce úloh pomocí služby YARN
- Editor příkazů Apache Hive
- Editor příkazů Apache Pig
- Prohlížeč služby Apache ZooKeeper
- Prohlížeč Apache HBase databáze

Uživatelské účty do HUE lze zakládat a odebírat buď lokálně (přímo ve službě, na konkrétním serveru) nebo pomocí LDAP. Lokální zakládání účtů není doporučený postup, centrální správa účtů má své výhody a řadí se mezi Best Practice při používání služby. HUE umožňuje hlavně business uživatelům provádět vše, co potřebují při práci na clusteru a jedná se o jednu z nejdůležitějších služeb pro uživatele. (*Hadoop*, 2019)

Webové rozhraní HUE vypadá takto:



Obrázek 8 - ukázka HUE, zdroj: (autor, 2019)

Vlevo je vidět seznam databází, do kterých má uživatel přístup. Do textového pole pod výběrem databáze lze zadat text a vyhledat konkrétní tabulku ve vybrané databázi. V pravé horní části je poté prostor pro zadávání SQL dotazů, jejichž výsledek se zobrazí hned pod ním.

5.5 Pig

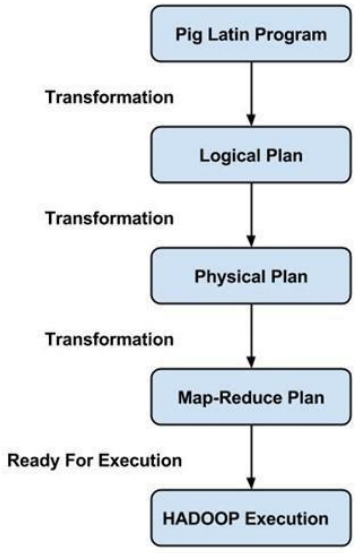
Apache Pig je vysoko úrovňový skriptovací jazyk používaný v rámci Hadoop a je výsledkem snažení vývojářů společnosti Yahoo. Jeho záměr je stejný jako v případě Apache Hive. Umožňuje uživatelům komplexní datové transformace bez jakékoliv znalosti Javy, díky čemuž se uživatelé mohou soustředit na konkrétní analýzu, místo řešení složité syntaxe při vytváření úloh v clusteru. Umí zpracovat data z mnoha zdrojů (jak strukturovaná, tak i nestrukturovaná). (*Apache Pig*, 2019)

Skládá se ze dvou komponent:

- Pig Latin, samotný skriptovací jazyk
- Runtime prostředí pro spouštění Pig programů

Konkrétní Pig program je složen ze série operací nebo transformací aplikovaných na vstupní data, které vedou k požadovanému výstupu. Tyto operace popisující tok dat jsou přeloženy do formy, která je spustitelná v prostředí pro spouštění Pig aplikací. Následně je celý proces přeložen do série MapReduce úloh – podobně jako u služby Hive – které jsou spuštěny. (*Apache Pig*, 2017)

PigLatin jazyk je poměrně snadný a používá klíčová slova, na která jsou uživatelé zvyklí v souvislosti s manipulací s daty, například *join*, *group* nebo *filter*.



Obrázek 9 - Architektura PIG, zdroj: (*Apache Pig*, 2017)

Pig má dva módy pro spuštění:

- Lokální mód – Pig běží v jedné JVM a využívá lokální file systém konkrétního serveru, na kterém je spuštěn. Tento mód je vhodný pouze pro analýzy, případné transformace pouze menších datových sad.
- MapReduce mód – V tomto módu jsou finální MapReduce úlohy spuštěny distribuovaně (je využitý výkon celého clusteru) a proto je tento mód vhodný pro analýzy případně transformace velkých datových sad.

I přesto, že jde v podstatě o alternativu služby Apache Hive, není Pig uživatelsky moc oblíben a v oblasti uživatelského zpracování dat jasně vede Hive, který má značnou výhodu v podobnosti dotazovacímu jazyku SQL, který je rozšířený po celém světě a naučit se ho je poměrně snadné.

5.6 Shrnutí

V této kapitole bylo vyjmenováno několik dalších komponent Frameworku Hadoop. Všechny z vyjmenovaných služeb s výjimkou HBase mají za cíl zjednodušovat práci uživatele při analytických úlohách. V praxi nejpoužívanějšími nástroji jsou Apache Hive a Apache Impala, které implementují oblíbený standard SQL do světa Big Data. Naopak méně uživatelsky oblíbený až téměř nepoužívaný je Apache Pig, pro svojí atypickou syntaxi. Apache HBase poté představuje alternativu k uložení dat na HDFS a pro svoje aplikační rozhraní je hlavně pro vývojáře také oblíbeným nástrojem.

6 Možné zabezpečení Hadoop platformy

Vzhledem k velkému množství důležitých firemních dat, které se na Hadoop platformě nachází, je třeba řešit zabezpečení. Všechny uzly v clusteru jsou ve stejné síti, a proto musí být tato síť zabezpečená, aby v ní nebylo žádné slabé místo. Prvním principem v IT bezpečnosti je přísná kontrola spojení clusteru s vnějším světem. Každý z uzlů má povolené určité porty a služby. Jedním z ověřených přístupů je vymezení jednoho uzlu pro komunikaci s vnějším prostředím. Takovému uzlu se říká edge server a plní roli prostředníka. (*Bmc,* 2018)

Řízení bezpečnosti v clusteru probíhá v několika vrstvách.

6.1.1 Autentizace

Proces, který má za úkol ověřit identitu uživatele snažícího se do systému dostat. Hadoop poskytuje dva módy autentizace. První, takzvané jednoduché ověření, zkrátka důvěřuje uživateli, že je skutečně ten, za kterého se vydává. Druhý mód využívá protokol Kerberos a poskytuje plně zabezpečený cluster, do kterého se nikdo nepovolaný není schopen dostat. V rámci Best Practices se doporučuje kromě Kerberosu využívat i některé z adresářových služeb jako jsou například LDAP nebo Active Directory, aby všechny objekty v clusteru bylo možné řídit z jednoho centrálního místa. Vytváření lokálních uživatelů a skupin se důrazně nedoporučuje a ve větších organizacích bývá správa uživatelů a skupin kontrolována jako první při auditní kontrole. (*Hadoop security,* 2018)

6.1.1.1 Active Directory

Active Directory je řešení adresářových služeb od společnosti Microsoft pro spravování síťových služeb. Pod pojmem adresářová služba si můžeme představit aplikaci či celou skupinu aplikací, které ukládají a organizují data o uživatelích a zdrojích v síti. Adresářová služba má přístup k adresáři, který je občas označován jako adresářový repositář[19]. Adresářová služba funguje také jako centrální autentizační autorita, která vykonává bezpečné ověřování uživatelů, služeb nebo počítačů. Data o všech objektech v síti jsou ukládány a následné jsou poskytovány administrátorům nebo jiným aplikacím, které je potřebují. (*Active Directory,* 2014)

Hlavní službou Active Directory je Domain Services (AD DS – doménové služby), která ukládá informace o složkách a zprostředkovává vzájemnou interakci mezi uživatelem a doménou. AD DS ověřuje přístup, když se uživatel snaží přistoupit na server přes síť

[19] Datové úložiště, odkud mohou být staženy a nainstalovány softwarové balíčky

6.1.1.1.1 Fyzická struktura služby AD DS

Fyzickou strukturu tvoří veškerý hardware a síťová infrastruktura, která je nutná pro správné fungování služby. Patří sem (*Active Directory,* 2014):

- Domain Controllers (Doménové řadiče) - Doménový řadič je fyzický nebo virtuální server, který má nainstalovanou roli AD DS. Na souborovým systému má fyzicky uloženou AD databázi. Pokud tato databáze obsahuje data o účtech v jednotlivých doménách, je to kritická komponenta celé služby.
- Active Directory Site (Active Directory lokalita) - Lokalita reprezentuje fyzickou síťovou strukturu organizace. Sdružuje skupinu serverů, které jsou propojeny rychlou síťovou linkou.
- Active Directory Partition (Active Directory rozdělení) - AD databáze je rozdělena na jednotlivé části. Každá z částí obsahuje specifická data a má svůj vlastní replikační rozsah.

6.1.1.1.2 Logická struktura služby AD DS

Logická struktura je nezávislá na fyzické struktuře. Patří sem (*Active Directory,* 2014):

- Domain (Doména) -Logický soubor objektů (například počítače nebo uživatelů) uložených v jedné AD databázi, které sdílejí stejnou bezpečnostní politiku
- Tree (Strom) - Doménový strom slouží jako logické uskupení jednotlivých domén a reprezentuje unikátní jmenný prostor systému doménových jmen (DNS namespace)
- Forest (Les) – Uskupení doménových stromů, které mají mezi sebou navázanou vzájemnou důvěru
- Organization Unit (Organizační jednotka) - Organizačné jednotky jsou objekty, které umožňují sdružování jednotlivých objektů, jako jsou skupiny, uživatelé, počítače nebo tiskárny
- Groups (Skupiny) - Jedna z nejdůležitějších komponent logické struktury pro řízení oprávnění. Jednotliví uživatelé jsou přiřazováni do skupin, což zjednodušuje s zpřehledňuje proces řízení oprávnění k jednotlivým prostředkům v síti

6.1.1.2 LDAP

LDAP, (z anglického Lightweight Directory Access Protocol) je jednoduchý aplikační protokol pro adresářové informační služby, který komunikuje skrze TCP/IP[20]. Je optimalizovaný pro čtení záznamů ve standardizované formě, např. dvojice jméno a heslo u autentifikační služby. Umožňuje komukoliv nalézt zaměstnance, zřízení nebo skupiny v síti. LDAP je méně robustní verzí DAP (Directory Access Protocol) protokolu, který je součástí standardu X.500 pro adresářové služby v síti.

[20] TCP/IP-primární přenosový protokol/protokol síťové vrstvy

V síti adresář udává, kde přesně se vybraný objekt nachází. V TCP/IP síti DNS (Domain Name System) je adresář, který systém využívá pro párování doménového jména s konkrétní, unikátní síťovou adresou. I když není známé doménové jméno, LDAP umožňuje snadné vyhledávání. LDAP adresář používá jednoduchou stromovou, hierarchickou strukturu. Ten obsahuje kořenový adresář (označován jako kořen stromu), který se dále rozvětvuje na země. Každá ze zemí se poté větví na organizace. Organizace na organizační jednotky, které vše zakončují rozdělením na jednotlivce (kromě zaměstnanců se sem řadí i soubory nebo tiskárny). Každý záznam v této struktuře se skládá z (*LDAP*, 2019):

- jedinečného jména záznamu (Distinguished Name – DN)
- atributů, které popisují záznam
- objektů (object class), které určují typ záznamu

LDAP adresář může být distribuován mezi více serverů. Každý ze serverů může mít replikovanou verzi celého adresáře a jeho stav je periodicky synchronizován.

V praxi je velmi oblíbený a je možné ho integrovat hned s několika službami, běžícími na Hadoop platformě. S LDAP lze velmi snadno integrovat například Hue, Cloudera Manager nebo Cloudera Navigator.

6.1.1.3 Kerberos

Kerberos je síťový autentizační protokol, navržený jako bezpečný nástroj pro autentizaci aplikací s architekturou klient–server přes nezabezpečenou síť. Byl vyvinu v rámci projektu Athena na MIT (Massachusetts Institute of Technology). Jeho první veřejná verze byla Kerberos V4. Kerberos V5 byl představen roku 1996, nicméně tato verze se používá ještě dnes. Jméno protokolu vzniklo na základě řecké mytologie, kde je Kerberos tříhlavý pes chránící vstup do podsvětí. (*Kerberos*, 2019)

Autentizace pomocí tohoto protokolu umožňuje využívat princip jednotného přihlášení (SSO – single Sign-On) mezi různými systémy. Metoda SSO dovoluje použit pouze jedno přihlášení do více aplikací, například ve Windows se uživatel ověřuje pouze jedou vůči Active Directory a jeho identita mu po celou dobu přihlášení zůstává. Vychází z předpokladu, že když se uživatel ověřuje u stejného zdroje, není nutné vyplňovat stejné údaje. Samotný protokol Kerberos používá silnou kryptografii. Využívá metodu klíčů, takzvaných tiketů. Při autentizaci klienta na serveru využívá důvěryhodnou třetí stranu, takže se klient neověřuje přímo na serveru. Třetí důvěryhodnou stranou je myšleno takzvané Key Distribution Center (dále jen KDC), v překladu centrum pro distribuci klíčů. Tento centrální autentizační prvek zvyšuje bezpečnost a zároveň může obsluhovat naráz více aplikací. Nevýhoda tohoto přístupu je, že v případě výpadku KDC přestane fungovat ověření do všech služeb, které Kerberos využívají. KDC má uložené šifrovací klíče všech klientů a ti se ověřují pouze vůči němu. Ověření pomocí protokolu Kerberos probíhá v následujících krocích (*Varonis,* 2018)

1. Autentizace uživatele – Při přihlášení do uživatelského stanice zadává uživatel svoje přihlašovací údaje. Klientská stanice následně provede jednosměrnou hashovací

funkci nad zadaným heslem a vygeneruje uživatelův Secret Key. Poté klient odešle žádost na Authentication Server (AS), což je součást KDC, která obsahuje uživatelské údaje. (*KRB,* 2018)

2. Získání tiketu – V rámci AS je ověřeno, zda uživatel skutečně existuje nebo ne (AS má přístup do Active Directory/LDAP) a v případě, že ano, vytvoří si vlastní Secret Key. Jako odpověď klientovi zašle takzvaný Session Key, který je využíván pro šifrování budoucí komunikace mezi KDC a klientem a je zašifrovaný Secret Key uživatele. Společně s ním mu odešle i unikátní klíč Ticket-Granting Ticket (dále jen TGT) obsahující Session Key. Fakt, že je klient schopen Session Key rozšifrovat potvrzuje, že je klient opravdu ten, za koho se vydává. TGT obsahuje i další údaje jako jsou jméno klienta, adresa nebo doba platnosti. TGT má omezenou dobu platnosti (výchozí hodnota je 10 hodin). TGT i Session Key jsou ukládány do ticket cache. (*KRB,* 2018)
3. Získání Service tiketu – Když se nyní chce klient přihlásit k nějaké službě v síti, použije získaný TGT a požádá KDC o takzvaný Service Ticket. Žádost o Service Tiket je posílána na Ticket Granting Service (dále jen TGS), což je další část KDC a je rozdělena do dvou zpráv. Jedna zpráva obsahuje ID služby a klientský TGT. Druhá zpráva slouží jako ověřovač, skládá se z identifikace uživatele a časové známky. Zároveň je šifrovaná pomocí Session Key. Dohromady tyto dvě zprávy tvoří Service Principal Name (dále jen SPN), tedy unikátní identifikátor služby běžící na serveru. Pokud TGS nalezne SPN v Active Directory (a jsou splněny další autentizační podmínky), pak odešle v odpovědi Service Ticket, který je zašifrovaný pomocí Secret Key. (*KRB,* 2018)
4. Autentizace u služby – Ověření klienta na síťové službě (serveru)
   Pomocí protokolu, který používá aplikační server, zašle klient na server data obsahující Service Ticket. Server poté rozšifruje Ticket pomocí svého klíče a dozví se z něj údaje o uživateli. V případě zapnuté obousměrné autentizace, odešle server klientovi šifrovanou časovou známku pomocí Session Key a dojde k ověření serveru klientem. (*KRB,* 2018)

Kerberos je stále hojně využíván pro následující vlastnosti (Kebreos advantages, 2017):

- Uživatelské heslo ani heslo se nikdy nepřenáší v síti – posílají se pouze šifrované tikety
- Využívá symetrickou kryptografii – podporuje různé šifry
- Jeho architektura je rozšiřitelná – umožňuje použít další nebo jiné šifry
- Podporuje obousměrnou autentizaci – lze nastavit ověřování serveru vůči klientům
- Chrání proti odposlechu nebo opakovanému použití stejných rámců
- Podporuje metodu SSO – umožňuje uživatelům přistupovat k různým službám bez toho, aby museli zadávat heslo. Zároveň tím zjednodušuje administraci, vše je spravováno centrálně.
- Funguje v modelu klient/server a používá pro ověřování důvěryhodnou třetí stranu KDC, to je zároveň slabina, jelikož KDC musí být stále dostupné a v případě výpadku selže ověřování do několika služeb naráz.
- Klient se vůbec neověřuje na serveru, ověření provádí KDC a odesílá serveru pouze šifrované uživatelovo jméno, které může rozšifrovat pouze server.

- Podporuje impersonation (česky zosobnění) – sám uživatel vůbec nepřistupuje ke zdrojům, místo něj to provádí systém.
- Je považován za bezpečný pro svůj systém dočasných Ticketů. Chrání proti řadě různých útoků a zneužití.

V Hadoop clusteru se Kerberos používá pro zabezpečení služeb Impala, Hive, HDFS nebo Kafka. Prakticky je celý proces pro uživatele, jenž se snaží ověřit vůči některé z vyjmenovaných služeb, velmi jednoduchý. Jedním příkazem požádá o Ticket (ten je uložen na straně klienta do ticket-cache), který následně platí do všech zmíněných služeb.

V praktické části se bude využívat několik pojmů souvisejících s protokolem Kerberos, a proto je potřeba vysvětlit :

Keytab – V češtině tabulka klíčů je soubor, který obsahuje SPN a odpovídající Secret Key. Využívá se pro rozšifrování komunikace při SSO a tím i klienta žádajícího o přístup k nějaké službe. (*KRB,* 2018)

Realm – Logická entita, ve které se nacházejí všechny účty. Význam je téměř totožný s významem pojmu doména ve Windows světě. Většinou se používá jméno realmu úplně stejné jako je DNS domény převedeným na velká písmena. (*KRB,* 2018)

Principal –Zkráceně SPN neboli Service Principal Name. Unikátní identita klienta žádajícího o přidělení Ticketu. Principal je ve formě *(uživatelské jméno)@(realm).* (*KRB,* 2018)

6.1.2 Autorizace

Proces, který specifikuje přístupy uživatelů a systému v určitém prostoru. Hadoop poskytuje řízení přístupu k souborům, podobně jako souborový Linux systém, pomocí oprávnění přiřazené přímo na soubor nebo takzvaným ACL (Access Control List), což je seznam pro řízení přístupů, který určuje, která skupina nebo samotný uživatel má k souboru přístup. Kromě přístupu k souborům je potřeba i řídit přístupy do databází. Ty jsou řízeny opět centrálně skrze nástroj Apache Sentry.

6.1.2.1 Linux/HDFS práva na složky

Linux je rozdílný od ostatních operačních systémů. Jeho odlišnost spočívá ve schopnosti mít více uživatelů. Byl navržen tak, aby více uživatelů mělo přístup do systému v jeden moment. Aby bylo možné tento přístup udržet, je nutné mít zajištěné, aby si uživatelé navzájem nemohli uškodit. Zde vstupují do hry uživatelské práva, které se nastavují na objekty v souborovém systému, konkrétně složky a soubory. Linux rozlišuje tři hlavní práva na svoje objekty (*Linux permissions,* 2019).

- Read – možnost čtení souboru. Umožňuje zobrazit obsah daného souboru.
- Write – možnost upravovat a následné ukládat obsah souboru.
- Execute – možnost spouštění. Právo execute nastavené na souboru umožňuje soubor spustit. Jedná se převážně o skripty nebo připravené programy. Stejné

právo nastavené na složku umožňuje uživateli přistoupit do dané složky a mít jí jako aktuální pracovní adresář.

Tyto práva se označují písmeny – r (read), w (write) a x (execute).

Dále je třeba určit, komu budou práva nastaveny. Linux opět rozlišuje tři základní skupiny v souvislosti s nastavením práv. Nově vzniklý soubor na Linuxu patří vždy tomu, kdo ho vytvořil a jeho výchozí osobní Linux skupině. Poslední pomyslnou skupinou jsou ostatní, jinými slovy kdokoliv jiný, kdo není vlastníkem ani Linuxovou skupinou. Tyto skupiny jsou označovány prvním písmenem do angličtiny přeložených slov uživatel, skupina a ostatní neboli u (user), g (group) a o (others). (*Linux permissions,* 2019).

Na základě výše vyjmenovaných skupin a práv, lze řídit přístupy k objektům. Tento přístup určuje práva pro jednoho konkrétního uživatele, jednu konkrétní skupinu a všem ostatním. Neumožňuje však definovat práva pro více jednotlivců.

HDFS používá naprosto totožný systém řízení práv, pouze s jinou syntaxí, aby bylo jasné, že budou práva nastavována na HDFS, a ne v Linuxovém souborovém systému

6.1.2.2 ACL – Access Control List

Řešením zmíněného nedostatku tradičního řízení práv Linuxového souborového systému je ACL. Zkratka od anglického „Access Control List“ neboli seznam pro řízení přístupu. Jedná se skutečně o seznam oprávnění, který je vázaný k nějakému objektu. V našem případě k souboru nebo celé složce. Tento seznam říká operačnímu systému, jaká přístupová práva mají jednotliví uživatelé nebo skupiny k souboru či složce. Používá se při specifikaci nějakých výjimečných práv, to znamená v případě, kdy je potřeba k nějakému objektu zajistit přístup jednotlivce nebo omezené skupiny uživatelů. Kromě přístupu konkrétních uživatelů nebo skupin se také přes ACL specifikuje konkrétní právo, které daná entita bude mít. Na Linuxovém souborovém systému jde o práva read (jen čtení), write (přepisovat soubor/složku) a execute (spouštění souboru). (*ACLs,* 2019)

HDFS používá znovu téměř stejné příkazy jako Linuxový souborový systém. I do HDFS je ACL implementován a lze skrze něj řídit přístupy ke složkám, respektive souborům. Opět je nastavování práv totožné jako v Linuxu, pouze s rozdílnou syntaxí pro specifikaci toho, kde bude ACL měněno.

6.1.2.3 Sentry

Apache Sentry je autorizační modul pro Hadoop poskytující granulární autorizaci založenou na rolích, které umožňují uživatelům vidět pouze ty objekty, na které mají přiřazené právo v clusteru. S jeho vývojem začala společnost Cloudera, nicméně na jeho vývoji se podíleli vývojáři z ostatních firem, například IBM, Oracle nebo Intel. Představen byl v roce 2013. Umožňuje spravovat přístupy – podobně jako AD a LDAP – centrálně, pro služby běžící v clusteru. Vytváření rolí a oprávnění lze spravovat skrze uživatelské rozhraní Hue nebo pomocí SQL dotazů spuštěných v nástroji Hive nebo Impala. Řídí přístupy k různým objektům v Hadoop platformě – databázím, tabulkám, URI a serverům. Role je přiřazována

vždy na skupinu, nikoliv jednotlivce. Sentry je integrovaná do celé platformy a je schopna řídit oprávnění k následujícím službám (*Sentry roles,* 2019):

- Apache Hive
- Hive metastore
- Apache Solr[21]
- Impala
- HDFS
- Apache Spark[22]

Všechny údaje o přístupech k objektům i o rolích jsou ukládány, což je dobré pro audit. Sentry umožňuje ukládat oprávnění ve dvou módech.

- Uložení do databáze – informace o právech jsou uložené v konkrétní databázi s názvem Sentry. Tento mód je preferovaný.
- Uložení do souboru – Všechny role a jí přidělené přístupy jsou zapisovány a čteny z konkrétního souboru (global-policy.ini). Tento způsob je však zastaralý a nepoužívá se.

Vytváření rolí a jejich následné přiřazování skupinám vykonává administrátor platformy, nicméně díky možnosti spravovat tyto role skrze Hive, respektive SQL dotazy je možnost tyto procesy automatizovat. (*Sentry roles ,* 2019)

6.1.3 Audit

Další z často řešených problémů z oblasti bezpečnosti na Hadoop platformě (a nejen na ní) je audit. Služby běžící na Hadoop platformě logují do logovacích souborů uložených na serverech (na kterých je daná služba spuštěna) všechny operace, které na clusteru provádí. Proto není těžké dohledat, kdy, který uživatel využíval, jakou službu a jaké operace prováděl. Pomocí adresářových služeb LDAP nebo AD lze zjistit, kteří uživatelé mají do clusteru přístup, díky Sentry poté kdo má přímo v clusteru přístup, ke kterým objektům. Společnost cloudera pro ještě snadnější audit a zároveň pro kompletní Data Governance vyvinula nástroj Cloudera Navigator, který shromažďuje všechny logy z celého clusteru na jedno místo, díky čemuž se dá snadno vyhledat auditní informace za určitý čas. Tento nástroj je však bohužel komerční, a proto ne všechny firmy ho využívají. (*Cloudera, 2019)*

6.1.4 Zabezpečení dat

Cílem zabezpečení dat je zajistit, aby k nim mohli přistupovat ti uživatelé, kteří jsou k tomu oprávněni. Je další vrstvou zabezpečení proti potenciálnímu útočníkovi. Úplné zabezpečení dat se skládá ze dvou částí. Zabezpečit data, když jsou zapisovaná na disk nebo jiné úložné zařízení a zabezpečit data během přenosu z jednoho procesu nebo systému na druhý. Cloudera Manager umožňuje jednoduché nastavení zabezpečení dat, které se v clusteru

[21] Indexovací systém pro vyhledávání ve velkém množství dat
[22] Framework pro zpracování Big Dat

přenášejí pomocí TLS šifrování. V praktické části této práce se s TLS zabezpečením dat počítá, nicméně žádný vliv na řešení nemá. (*Cloudera,* 2019)

6.1.4.1 TLS šifrování

Zkratka TLS (Transport Security Layer) je kryptografický protokol považovaný za nástupce protokolu SSL (Secure Socket Layer). SSL verze 3.0 se od TLS verze 1.0 prakticky neliší s výjimkou nepatrných rozdílů. Proto často mluvíme o TLS/SSL zabezpečení. TLS oproti SSL umožňuje začít komunikaci v nešifrovaném formátu a nasadit šifrované zabezpečení na ty servery, které hostují více domén na jedné IP adrese. V Hadoop clusteru se pohybuje při autorizaci a autentizaci velké množství citlivých dat. Mnoho služeb také ukládá hesla ve svých konfiguračních souborech. Ukradení těchto dat by mohlo mít fatální následky pro celou platformu. (*SSSL,* 2019)

Pro zabezpečení přenosu dat je nutné nahrát certifikáty na každý node v clusteru a označit jej za důvěryhodný. Dále je potřeba vygenerovat dvojici klíčů (privátní a veřejný klíč) spojených s certifikátem. Pro správu klíčů a certifikátů se používá Java Keystore, což je zabezpečené místo s certifikáty a klíči. Je uložený na souborovém systému a je chráněný heslem. Každý záznam (myšleno klíč nebo certifikát) je identifikován aliasem. Java Keystore však nelze použít pro všechny služby běžící v clusteru, například Hue nebo Impala vyžadují pouze podepsaný certifikát a příslušný privátní klíč, vzhledem k tomu, že nejsou napsané v Javě. (*IBM,* 2019)

Aby byl certifikát označený jako důvěryhodný, musí být podepsaný jedním z následujících způsobů :

- Podpis veřejnou certifikační autoritou – Společnost Cloudera tento způsob označuje za doporučený. Veřejná certifikační autorita je důvěryhodná třetí strana. Získání podepsaného certifikátu z některé z nich (například Symantec[23]nebo Comodo[24]) usnadňuje instalaci certifikátu, protože je již připravená bezpečnostní infrastruktura (výchozí Java Keystore již obsahuje certifikáty některých autorit). (*Cloudera,* 2019)
- Podpis interní certifikační autoritou – Organizace disponuje vlastní certifikační autoritu a má zavedenou politiku pro podpis certifikátů. Tuto možnost nabízí například Microsoft Active Directory Certificate Services[25] nebo OpenSSL Certificate Authority[26]. (*Cloudera,* 2019)
- Vlastní podpis certifikátu – Metoda nedoporučovaná do produkčního prostředí. Spočívá v generování vlastního certifikátu pomocí nástroje OpenSSL, kterým je možné podepisovat další certifikáty a tím je označit za věrohodné. Takový certifikát je označován jako root certifikát. Je potřeba ho následně nainstalovat do

[23] www.symantec.com
[24] www.comodo.com
[25] Platforma pro vydávání a správu certifikátů
[26] Open source nástroj pro implementaci kryptografických funkcí

důvěryhodných certifikátů jednotlivých stanic, aby byl přenos dat správně zabezpečen. (*Cloudera,* 2019)

6.1.5 Shrnutí

Všechny z vyjmenovaných metod zabezpečení vychází z Best Practices a jsou důrazně doporučovány pro provoz Hadoop clusteru. Zároveň však představují výzvu pro všechny, kteří se na zabezpečení podílejí. Do procesu vstupuje více stran, ať už je to LDAP administrátora, který má za úkol integrovat LDAP s Kerberosem nebo administrátor Hadoop platformy, který musí zajistit správné spojení služeb (které podporují vzdálenou autentizaci) s LDAP. V neposlední řadě je také třeba zavést proces auditu a vyřešit podepisování TLS/SSL certifikátů.

I přes veškeré překážky a výzvy, které přinášejí zmíněné technologie, je zabezpečení nedílnou součástí vývoje Hadoop platformy a je bráno jako standardní část projektu, jehož cílem je vybudovat analytickou platformu.

7 User management

7.1 Vymezení user management procesů

Pod pojmem user management si lze představit činnost nebo skupinu činností, při které se spravují uživatelské účty a jejich přístupy do sítí, služeb nebo aplikací. Tyto činnosti se zpravidla neliší, případně pouze minimálně, jejich automatizace je tedy zcela na místě a může ušetřit mnoho času. Další velkou výhodou automatizace těchto procesů je zároveň zavedení monitorování a audit toho, kdo na platformu přistupuje, případně jaké objekty se na ní vyskytují. Veškeré procesy user managementu na platformě jsou prováděny výhradně administrátorem. Před každou změnou (založení uživatele, složek, databází) se provádí schvalovací proces a veškeré změny jsou auditovány pomocí logů, případně jiných auditních nástrojů. V této práci není nijak řešen proces schvalování, každá organizace řeší schvalovací procesy po svém. Procesy user managementu nejsou nikde pevně stanovené. Lze je uzpůsobit potřebám firmy. Na základě zkušeností a pomoci od kolegů, kteří administrovali nějakou Hadoop platformu byly stanoveny obecné procesy, které lze využít ve většině Hadoop prostředích:

- Založení uživatele na platformě
- Odstranění uživatelů z platformy
- Založení Workspace
- Odstranění Workspace
- Založení PoC Workspace

Tyto procesy jsou nejčastěji vykonávané v korporátním prostředí a v následujících kapitolách budou popsány podrobněji. Zároveň je však potřeba znovu zdůraznit, že nejsou pevně dané, na jejich úpravě a vývoji lze dále pracovat.

7.2 Návrh řešení

Před samotným návrhem řešení je třeba ještě vysvětlit pár pojmů. Po zkoumání vhodné notace, respektive diagramů pro přehlednou vizualizaci procesů v organizaci zůstali v úvahu tři modelovací jazyky/notace – UML, Archimate a BPMN 2.0. Konečnou volbou se stal Archimate, který z uvedených modelovacích jazyků nejnovější. Sledovanou metrikou při rozhodování byla jednoduchost a přehlednost.

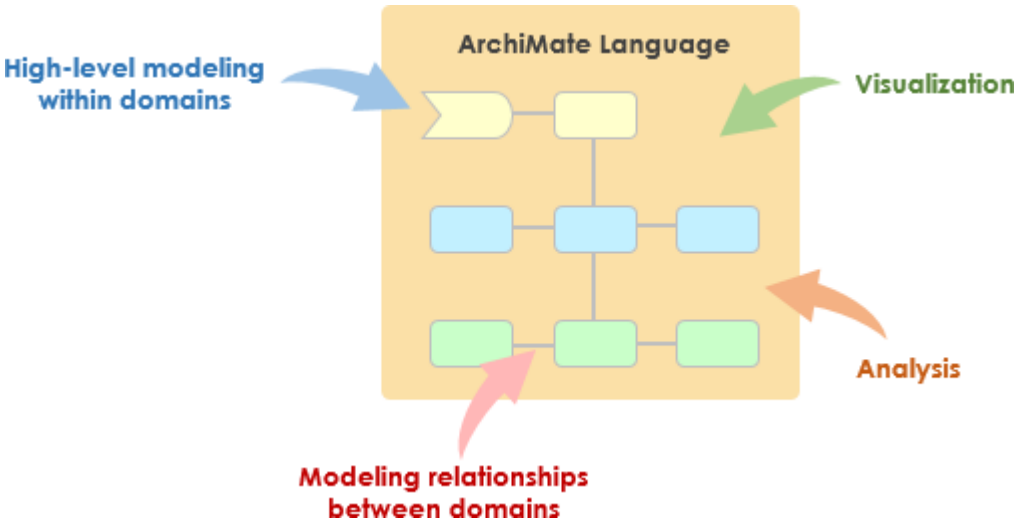
Archimate je otevřený a nezávislý modelovací jazyk podporovaný různými dodavateli nástrojů a poradenskými firmami. Archimate poskytuje nástroje pro podporu podnikových architektů při popisu, analýze a vizualizaci vztahů mezi různými doménami architektury jednoznačným způsobem. (*Visual-Paradigm,* 2018)

Poskytuje jednotný jazyk pro konstrukce a operace v:

- Business procesech
- Organizační struktuře
- Informačních tocích
- IT systémů
- Technické architektuře

Vhled do problematiky pomáhá všem zúčastněným stranám navrhovat, posuzovat nebo zkoumat následky některých rozhodnutí a změn mezi business doménami.

Pro business hrají modely důležitou roli již dlouhou dobu. Procesní model, informační, datové modely, strategické nebo operativní modely. Organizace se snaží propojit všechny svoje modely, aby získali pohled z různých pohledů. Modely jsou typicky povrchnější, když se soustředí na podstatu organizace. Tento trend vyústil ve vývoj Archimate jazyku, který poskytuje obecné modelování v rámci domén, ale zároveň umožňuje modelování vztahů mezi nimi. Příklad diagramu modelovaném v jazyce Archimate je vidět na obrázku 10. (*Visual-Paradigm*, 2018)



Obrázek 10 - Jazyk ArchiMate, zdroj: (*Visual-Paradigm,* 2018)

Archimate diagramy lze v tomto případě využít ve všech fázích pro všechny zúčastněné strany. Business vrstva – žlutou barvou – je vždy horní vrstva diagramu (viz. obrázek 10). Modrá barva značí aplikační vrstvu, zelená poté technologickou. Krása Archimate diagramu spočívá v tom, že každá ze zúčastněných stran (business, analytik, vývojář) si z něho odnesou to, co je pro ně zrovna důležité.

7.3 Popis prostředí

Implementace řešení proběhne na laboratorním clusteru firmy Adastra. Přístup do clusteru je řízen pomocí LDAP, pro přihlášení na jednotlivé nody je potřeba vygenerovat dvojici klíčů. Autentizace probíhá veřejným klíčem, nikoliv heslem. V clusteru je připojeno jedenáct nodů z toho 7 z nich jsou worker nody. Dále je v něm jeden takzvaný Edge server, který slouží jako přístupový bod uživatele do clusteru a běží na něm nejvíce komponent Hadoop služeb. Typicky jsou všichni uživatelé přesměrováváni na tento typ serveru, proto by měl mít i lepší hardwarové parametry. Dalším typem nodu je Head server. Ty bývají v clusteru zpravidla dva, jeden jako záloha, protože je na nich nainstalován Hive Metastore, jehož výpadek má za následek výpadek celého clusteru. Posledním typem serveru je management, který je pro tuto práci velmi důležitý, jelikož je na něm nainstalován webový server Apache. Následující tabulka shrnuje všechny nody v clusteru a jejich názvy.

Číslo serveru	Název server	Role
1	Edge1	Edge server
2	Head1	Head server
3	Head2	Head server
4	Worker1	Worker server
5	Worker2	Worker server
6	Worker3	Worker server
7	Worker4	Worker server
8	Worker5	Worker server
9	Worker6	Worker server
10	Worker7	Worker server
11	Management	Mangement server

Obrázek 11 - Popis prostředí, zdroj: (autor, 2019)

Z bezpečnostních důvodů nejsou uvedeny IP adresy serverů.

Servery vyznačené šedou barvou jsou pro řešení této práce kritické. Na jiné servery přímo přistupováno nebude. Jak již bylo zmíněno, na serveru Management je nainstalovaný webový server Apache, který je použitý pro realizaci webového uživatelského rozhraní a

budou z něj spouštěné všechny potřebné skripty. Head2 server je poté vyznačen, protože se na něm nachází KDC a bude na něm probíhat generování Keytabu pro ověření u služeb.

Celý cluster je zabezpečen protokolem Kerberos. Kerberos je třeba se ověřit u následujících služeb

- Impala
- Hive
- HDFS

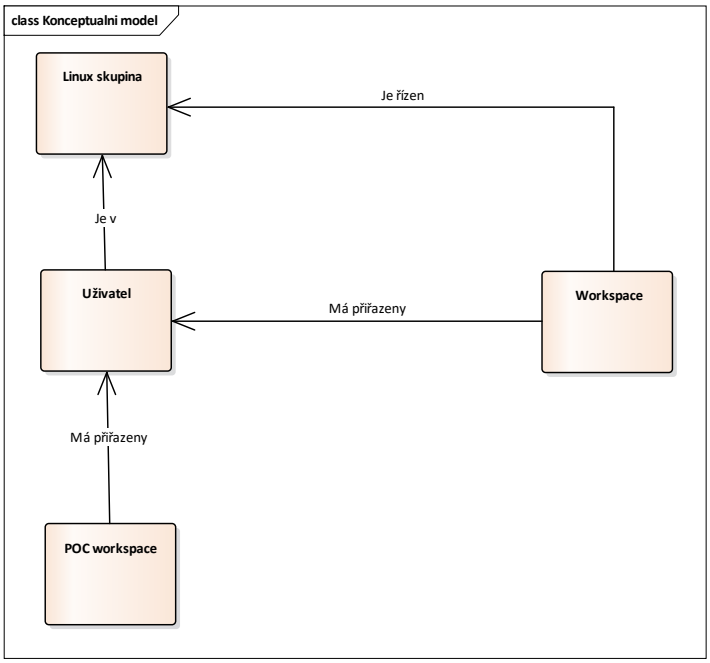
V clusteru běží CDH 6.2.

7.4 Návrh databáze

Pro celkové řešení je potřeba vytvořit databázi objektů, které na platformě budou vznikat. Ta pomůže jednak při implementaci navrhnutého řešení a jednak bude poskytovat přehled toho, co se na platformě děje. Využita bude MariaDB, která v clusteru běžet musí pro správný chod celého Hadoop. Je nainstalovaná na třech serverech v clusteru, všechny instance však o sobě navzájem vědí a data synchronizují mezi sebou. Služby běžící v rámci Cloudera distribuce ukládají data do té, která je na serveru Head1. Proto je vhodné využít jiný server, aby nebylo ohroženo fungování platformy pro přetížení.

7.4.1 Konceptuální návrh

Při pohledu na definované procesy určené k automatizaci je vidět o jakých entitách bude potřeba udržovat informace a jaké mezi nimi jsou vztahy. Nejdůležitější entitou pro user management procesy je samozřejmě samotný uživatel. Uživatel může být přiřazen do dvou dalších entit. Do Workspace nebo PoC Workspace. Je proto nutné držet informace i o těchto entitách. Workspace a PoC Workspace budou mít téměř totožné atributy, rozdíl je v jejich relacích. Workspace má přiřazeny uživatele a vstup do něj je řízen Linuxovými skupinami. Členství uživatele v Linuxové skupině, která řídí přístup do Workspace opravňuje uživatele do Workspace přistupovat. Například když je uživatel přiřazený do CRM Workspace, musí být členem některé z Linuxových skupin, které vzniknou na základě požadavku o vytvoření Workspace. PoC Workspace má též přiřazené uživatele, není však řízen žádnou Linuxovou skupinou. Konceptuální návrh byl vytvořen pomocí nástroje Enterprise Architect:



Obrázek 12 - konceptuální návrh, zdroj: (autor, 2019)

7.4.2 Logický návrh

Na základě konceptuálního návrhu byl vytvořen logický návrh definující konkrétní atributy jednotlivých entit vztahy mezi nimi. Hlavní entitou celého návrhu je uživatel. O uživateli jsou ukládány následující atributy:

- UserID – unikátní identifikátor uživatel
- Username – uživatelské jméno v LDAP
- First_name – křestní jméno uživatele
- Last_name – přijmení uživatel
- Dt_created – datum vložení záznamu

Další tabulkou je Workspace. O Workspace postačí udržovat:

- WorkspaceID – unikátní identifikátor Workspace
- Workspace_name – název Workspace
- Is_created – příznak toho, jestli je Workspace založený

Vztah mezi uživatelem a Workspace je typu many to many (M:N). Jeden uživatel může být v jednom nebo více Workspace a naopak. Uživatel však nemusí být členem žádného Workspace. Workspace však musí mít alespoň jednoho uživatele.

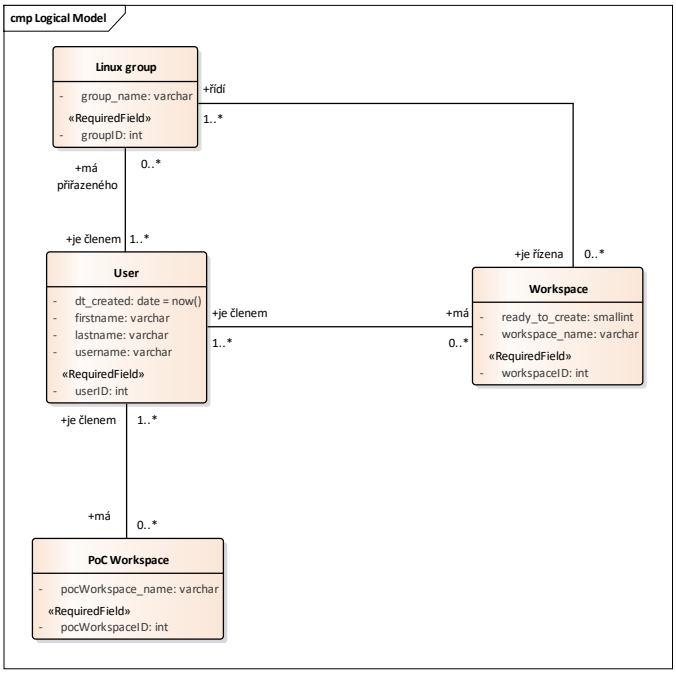
Stejný vztah je i mezi uživatelem a PoC Workspace. Pro vytvoření PoC Workspace musí být přiřazený aspoň jeden uživatel, ten však do žádného PoC Workspace patřit nemusí. O PoC Workspace budou udržováno pouze:

- PocWorkspaceID – unikátní identifikátor PoC Workspace
- PocWorkspace_name – název PoC Workspace

Poslední vytvořenou tabulkou je Linux skupina, která má vazbu na uživatele a na Workspace. Při založení Workspace musí zároveň vzniknout alespoň jedna Linuxová skupina. Linuxová skupina nemusí řídit žádný Workspace.

Posledním vztahem je vztah mezi uživatelem a Linuxovou skupinou. Uživatel může být členem N skupiny, ale nemusí. Technicky je možné, aby skupina neměla přiřazené žádné uživatele, nicméně jak bylo řečeno, při založení Workspace je nutné založit alespoň jednu Linuxovou skupinu a přiřadit minimálně jednoho uživatele. Proto je databáze navržena tak, že Linuxová skupina musí mít přiřazeného alespoň jednoho uživatele.

Logický návrh databáze je vidět na obrázku 12.

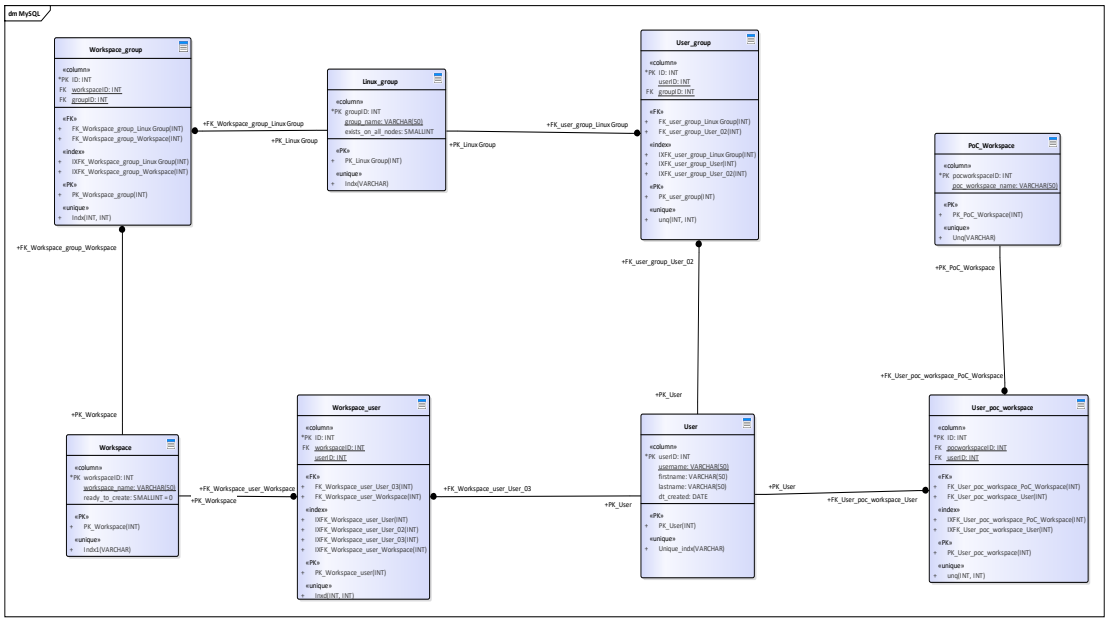


Obrázek 13 - Logický návrh databáze, zdroj: (autor, 2019)

Mezi všemi entitami navrhnuté databáze jsou vztahy many to many (M:N), proto bude třeba ve fyzickém návrhu tyto vztahy vyřešit vazebními tabulkami.

7.4.3 Fyzický návrh

Do fyzického návrhu databáze jsou přidány vazební tabulky pro realizaci vzájemných vztahů a definovány všechna omezení v čele s primárními a cizími klíči.



Obrázek 14 -Fyzický návrh databáze, zdroj: (autor, 2019)

Mezi všemi entitami byly vytvořeny vazební tabulky kvůli vztahu typu many to many (M:N). Ty obsahují vždy jednoznačné identifikátory spojovaných tabulek. Kombinace těchto jednoznačných identifikátorů je vždy nastavená jako unikátní index, aby bylo zabráněno duplicitám při vkládání. Unikátní indexy jsou nastaveny rovněž na atributech, které uchovávají název entit. Konkrétně to jsou:

- Workspace_name (tabulka Workspace)
- Username (tabulka User)
- Poc_workspace_name (tabulka PoC _Workspace)
- Group_name (tabulka Linux Group)

Pro všechny vazební tabulky v databázi je nastavena možnost *ON DELETE Cascade*, která zajistí, že budou z vazebních tabulek smazány všechny záznamy, propojené se svým rodičovským záznamem.

Primární klíče všech tabulek jsou jednoduše ID – jednoznačné identifikátory, které jsou nastaveny jako auto inkrement, což zajišťuje automatické přidávání ID hodnoty bez její specifikace.

7.4.4 Realizace

Z vytvořeného fyzického návrhu byl vygenerován DDL skript (viz. příloha 1), který byl spuštěn v připravené databázi běžící v clusteru.

7.5 Založení uživatele na platformě

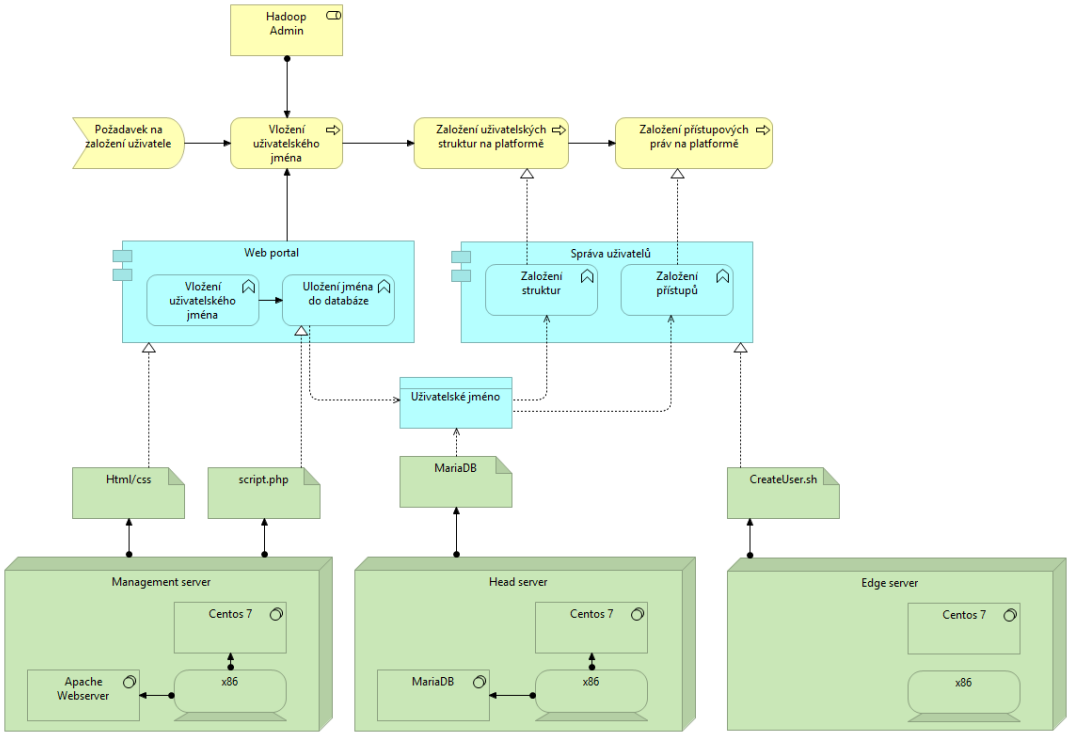
Prvním procesem, který se provádí velice frekventovaně a bývá prvním krokem ke všem následujícím akcím, je založení uživatele na platformě. Pro přístup na platformu je nutné, aby byl účet uživatele založen v AD/LDAP serveru, který je propojený s platformou. Tato činnosti však bývá prováděna hned při nástupu zaměstnance do firmy, a proto považujeme tento předpoklad také za splněný. Při zakládání uživatele na Linux prostředí vzniká i jeho osobní unikátní skupina, která je potřeba dále například při zakládání role v Sentry pro přistup do osobní databáze. Vzhledem k tomuto faktu lze proces založení uživatele provést okamžitě, tzn. po potvrzení v GUI bude akce provedena. Jako první akce v procesu je provedena administrátorem. Jde o vyplnění uživatelského jména uživatele v připraveném webovém grafickém rozhraní na základě schválené žádosti v některém z tracking systému (například Jira). Jakmile administrátor potvrdí založení, dochází současně ke třem akcím, které jsou na sobě časově nezávislé –

- Založení Hive databáze
- Založení domovského adresáře na Linux serverech
- Založení domovského adresáře na HDFS

Následně na založení Hive databáze navazuje založení role v Sentry, díky které se uživatel do své osobní databáze dostane a umožní mu vkládat do domovské složky na HDFS.

Po založení domovského adresáře poté následuje vygenerování klíče pro ssh přístup. Zde však záleží na zabezpečení dané platformy. Některé mají ssh přístup podmíněný „pouze“ heslem a není zde tento krok potřeba. V našem případě však probíhá přihlášení pomocí privátního a veřejného klíče, proto je třeba privátní klíč vygenerovat a vložit ho do domovského adresáře.

Jakmile se všechny zmíněné akce provedou, proces je hotový. Návrh celého proces znázorňuje následující diagram:



Obrázek 15 - Proces vytvoření uživatele, zdroj: (autor, 2019)

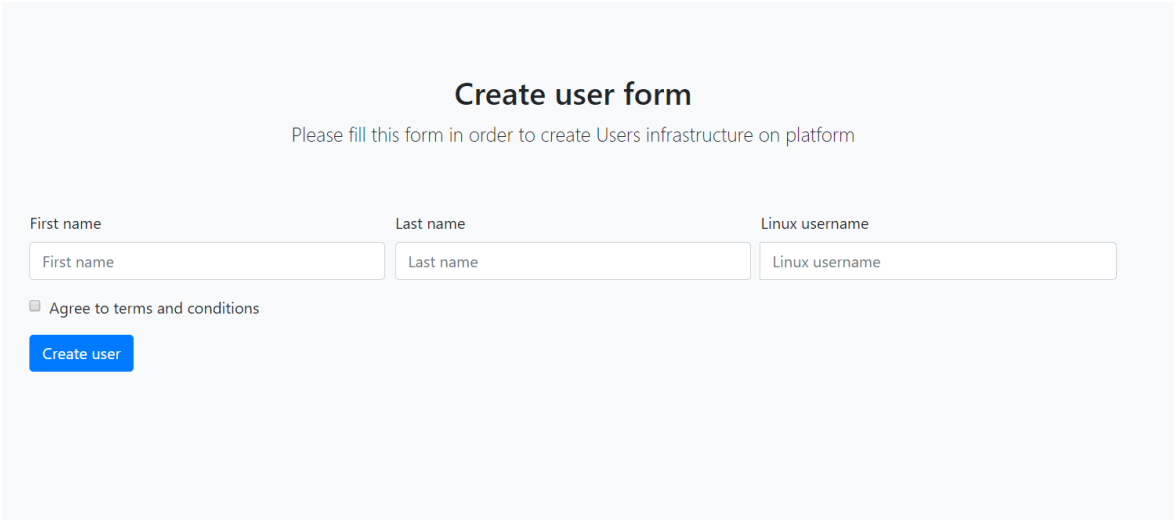
Žlutá vrstva znázorňuje business pohled na proces, modrá a zelená vrstva je poté určena technicky zdatným a společně představují technický návrh řešení včetně plánovaných technologií. Za zmínku stojí uložení uživatele z webového portálu do databáze. Při zakládání uživatele nehraje žádnou roli, nicméně má svůj význam v dalších procesech.

7.5.1 Zadání uživatelského jména uživatele

Jak je vidět z diagramu, prvním krokem je zadání jména uživatele, kterého chceme na platformě založit do webové aplikace běžící na management serveru. Uživatel musí existovat v AD/LDAPU, aby bylo možné celý proces úspěšně dokončit. I přes to, že očekáváme, že uživatel opravdu v AD/LDAPU existuje, může zde dojít k uživatelské chybě (například překlepu pří zadávání uživatelského jména) a proto je potřeba takové chyby ošetřit. Proto hned při potvrzení založení uživatele proběhne kontrola existence zadaného uživatele. V případě, že uživatel neexistuje, pravděpodobně jde o překlep a zadávající bude na chybu upozorněn.

Pokud uživatel existuje, může proces pokračovat. Pro přehled uživatelů založených na platformě jsou všechny vyplněné informace uloženy do databáze. Ukládáno je jméno uživatele, příjmení a jeho Linuxové jméno, které nesmí obsahovat žádné speciální znaky s výjimkou tečky, která však bude stejně v následujících krocích nahrazena podtržítkem, jelikož založit Linuxovou skupinu, ani Hive databázi s tečkou v názvu nelze.

Formulář ve webové aplikaci pro založení uživatele na platformě je vidět na následujícím obrázku:

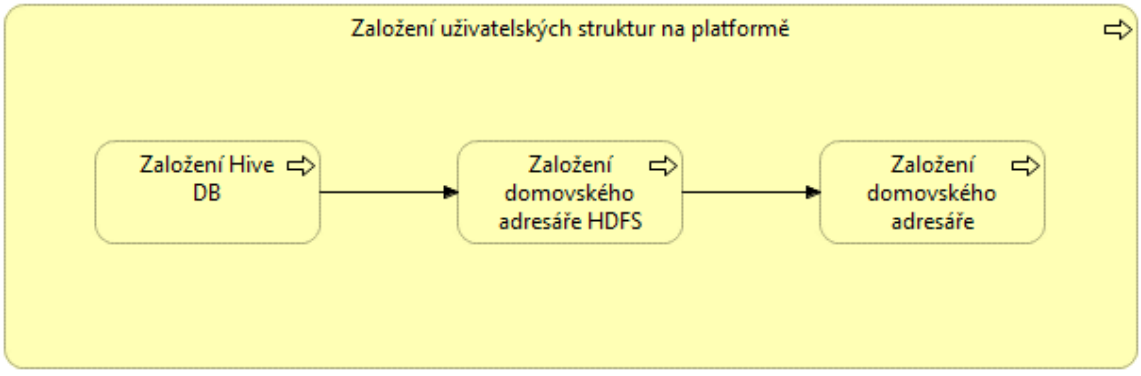


Obrázek 16 - Formulář pro založení uživatele, zdroj: (autor, 2019)

Pro uložení údajů o uživateli platformy je nutné vytvořit databázi obsahující tabulku uživatelů. Uchovávanými informacemi o uživatelích jsou křestní jméno, příjmení a uživatelské jméno uvedené v LDAP.

7.5.2 Založení uživatelských struktur

Poté co jsou všechna potřebná data uložena je následně zavolán Shell skript, který zakládá uživatele, respektive uživatelský prostor na platformě. První část tohoto skriptu zakládá strukturu uživatelského prostředí.



Obrázek 17 - Zakládání uživatelských struktur, zdroj: (autor, 2019)

Struktura uživatelského prostoru je založena v případě, že existuje osobní Hive databáze uživatele, domovský adresář na HDFS a domovský adresář na všech Linux serverech.

7.5.2.1 Založení Hive databáze

Skript pro zakládání uživatelských struktur začíná tvorbou Hive databáze, která se bude jmenovat **usr_(uživatelské_jméno)**. Zároveň, jak již bylo řečeno v uživatelským jménu by neměli být tečky kvůli tomu, že si s ní některé služby neporadí, a proto probíhá nahrazení tečky podtržítkem. Syntaxe pro založení je stejná jako v relačních databázích.

Založení databáze proběhne příkazem:

Create database if not exists (název databáze);

Lze založit i příkazem:

Create database if not exists (název databáze) comment (koment k databázi) with dbproperties (název vlastnosti = vlastnost);

Tímto způsobem je možné udržovat metadata o databázi, nicméně není to povinné. Klauzule „if not exists" zajišťuje, že celý skript nespadne, v případě, že databáze již existuje.

7.5.2.2 Založení domovského adresáře na HDFS

HDFS používá téměř totožné příkazy jako Linux, jediný rozdíl je, že před příkaz, který chceme na HDFS vykonat přidáme:

Hdfs dfs nebo *Hadoop fs*.

Příkaz pro vytvoření složky na HDFS vypadá takto:

Hdfs dfs – mkdir – p /cesta/ke/složce

Stejně jako u Linuxu i zde parmetr – p zajišťuje, že se nic nestane v případě, že složka již jednou založena byla. Poté co složka existuje, je potřeba jí přivlastnit samotnému uživateli příkazem:

hdfs dfs -chown (uživatelské jméno):(skupina uživatele) /user/(uživatelské jméno)

Což zajistí možnost uživateli se svojí složkou manipulovat. Pro nahrávání souborů do ní však je ještě zapotřebí, aby byla nastavena správná ACL (viz. Kapitola ACL) a to pomocí příkazu:

hdfs dfs -setfacl -m user:(uživatelské jméno):rwx /user/(uživatelské jméno)

A následně byla přiřazena role, která má oprávnění all na URI směřující na domovskou složku uživatele. Vytváření a přiřazení role však je prováděno až v druhé části skriptu.

7.5.2.3 Založení domovského Linux adresáře

Poslední akcí první části skriptu pro založení uživatele na platformě je založení uživatelského domovského Linux adresáře na všech uzlech v clusteru. Příkaz pro založení složky v Linux prostředí vypadá následovně:

mkdir -p /cesta/ke/složce

Obdovně jako u zakládání HDFS složky je provedeno přivlastnění složky uživateli neboli jejímu vlastníkovi. Toho bylo docíleno takto:

chown (uživatelské jméno):(uživatelská skupina) /cesta/ke/složce

Pro založení na všechny servery je potom použitý příkaz ssh. V kombinaci s předchozím vypadá příkaz pro založení domovské složky na všech serverech v clusteru následovně:

Servers = („server1“ „server2“ „server3“)

for server in ${servers[@]}; do

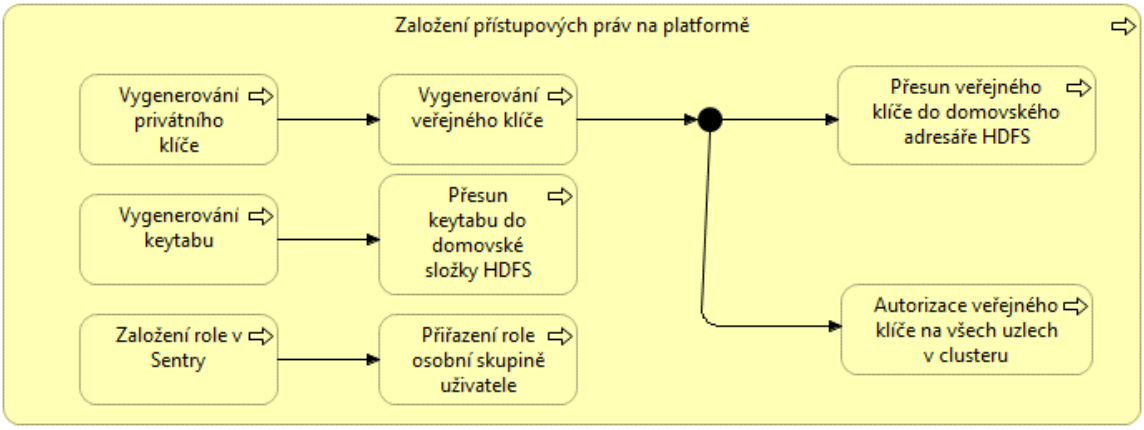
ssh -i Jan.Kodet.pem ${server} bash -c 'mkdir -p ${home_folder}$1 chown user:$1 ${}'

done

Poté co je založena složka uživatele na všech uzlech v clusteru, končí první část zakládání uživatele. Všechny struktury jsou připraveny, zbývá do nich přiřadit uživatele.

7.5.3 Založení přístupových práv uživatele

Ve druhé části zakládacího skriptu probíhá zakládání přístupových práv do nově vzniklých struktur (složek a databáze). Prováděny jsou následující operace:



Obrázek 18 - Založení přístupových práv, zdroj: (autor, 2019)

7.5.3.1 Vygenerování privátního klíče

Jak již bylo řečeno, pro přístup na platformu je zapotřebí mít vlastní privátní a veřejný SSH klíč. Jedná se však o specifický příklad, ne všechny platformy vyžadují dvojici klíčů. Lze se ověřit i pouze pomocí uživatelského jména a hesla. V našem případě však ověřování probíhá pomocí klíčů, a tak je nutné tuto operaci do procesu zakomponovat.

Výchozím algoritmem pro generování SSH klíčů je RSA – podle jeho autorů, Rivest, Shamir a Adelman. Pro vygenerování veřejného SSH klíče RSA algoritmem v Linux prostředí se používá příkaz:

ssh-keygen – t rsa

Při generování klíče algoritmem RSA je přepínač – t zbytečný, protože RSA algoritmus je výchozí.

7.5.3.2 Vygenerování veřejného klíče

Následně je z veřejného klíče vygenerován privátní klíč, který má u sebe pouze uživatel a je předkládán při každém přihlášení uživatele. Privátní klíč je možné generovat nástrojem PuttyGen, příkazem:

puttygen /cesta/k/veřejnému/klíči -o /cesta/kam/klíč/nahrát

Tím je vytvořena dvojice klíčů.

7.5.3.3 Přesun privátního klíče do domovského adresáře HDFS

Privátní klíč je poté nahrán do domovské HDFS složky uživatele, aby ho měl uživatel dostupný (může si ho stáhnout skrze Hue, kde se ověřuje pomocí LDAP). Operace je provedena příkazem:

hdfs dfs -put /Linux/cesta/ke/klíči /hdfs/cílová/cesta

7.5.3.4 Autorizace veřejného klíče

Po vygenerovaní veřejného klíče se musí tento klíč označit za věrohodný a přesunout do souboru **/home/(uživatelské_jméno)/.ssh/authorized_keys**. Pro přesun souborů v Linuxovém prostředí se používá příkaz:

cp /zdrojová/cesta/k/souboru /cílová/cesta/souboru

Tato akce se musí provést na všech uzlech clusteru, na které má uživatel mít přístup. To zajistí příkaz scp, který umožňuje přenos souborů mezi servery. Příkaz scp vypadá takto:

scp/cesta/k/lokálnímu/souboru uživatelské_jméno@název_serveru:/cesta/k/vzdálenému/souboru

7.5.3.5 Vygenerování Keytabu

Ověřování vůči cloudera službám se provádí pomocí Keytabu. Jeho vygenerování je o trošku složitější než generování privátního a veřejného klíče v tom, že se musí dělat na serveru, na kterém je nainstalovaný Kerberos. Na tomto serveru je třeba zadat příkaz:

kadmin.local

a následně spustit samotné generování Keytabu príkazem:

ktadd -norandkey -k (název.Keytabu) (uživatelské_jméno)@(realm)

Vygenerovaný Keytab poté je opět přesunut do domovské HDFS složky uživatele, aby k němu měl přístup.

7.5.3.6 Založení role v Sentry

Přístupy do databáze Hive jsou řízeny rolemi nástrojem Sentry. Po vytvoření Hive databáze do ní nikdo, kromě administrátorů nemá přístup. Role se podobně jako u některých relačních databázových systému vytvoří příkazem:

create role název_role;

V tuto chvíli však ani tato role samotná nemá žádné oprávnění. Proto je třeba pro ni specifikovat oprávnění příkazem:

grant oprávnění on (název databáze).(název tabulky) to role (název role);

V případě, že musí být uděleno právo na celou databázi, je použitý příkaz:

grant (oprávnění) on (název databáze). to role (název role).*

Oprávnění může být select, drop nebo all.

Role v Sentry nemohou být přiřazeny konkrétnímu uživateli. Přiřazení role lze provést pouze na konkrétní skupinu. V případě zakládání uživatele je proto využito toho, že každý uživatel má svojí osobní a zároveň unikátní skupinu, které je role s patřičnými oprávněními přiřazena a to příkazem:

grant (název role) to group (název skupiny);

Tím je role přiřazena konkrétní skupině, jejíž členové mají nyní přístup do dané databáze.

7.5.3.7 Shrnutí

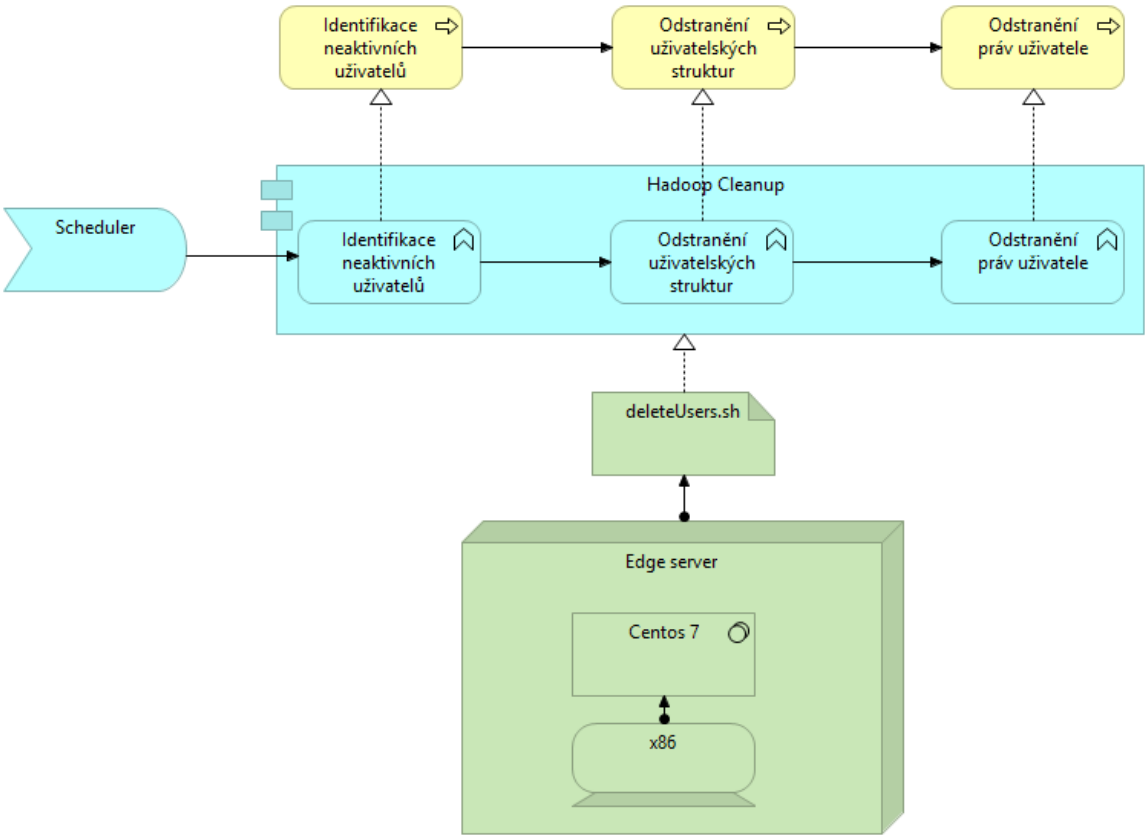
Po úspěšném skončení všech procesů je vše připraveno k tomu, aby uživatel platformu začal plně využívat.

Skrze LDAP může přistoupit do své nově vytvořené domovské HDFS složky, ve které již má připraveny Keytab a privátní klíč. Pomocí Keytabu se může ověřit u některých služeb běžících v clusteru (vyjmenované v kapitole 1.10.2.3), pomocí privátního klíče se zase dostane přímo na určené uzly v clusteru přes ssh příkaz. Zároveň má založenou databázi, kde může libovolně zakládat a mazat tabulky díky založené roli v Sentry s nastavenými právy all. V neposlední řadě také může ukládat soubory na HDFS do své domovské složky.

7.6 Odstranění neaktivních uživatelů z platformy

Při odchodu zaměstnance nebo ukončení práce externisty na projektu nemá cenu uživatele na platformě dále udržovat. Odstranění neaktivních uživatelů z LDAP bývá většinou ve firmách zaběhlý proces probíhající v určitém intervalu a stejně bude probíhat i odstranění uživatele z Hadoop platformy, respektive všech neaktivních uživatelů, kteří již v ani LDAP nejsou.

Obecný pohled na celý proces vypadá takto:



Obrázek 19 - Odstranění uživatelů, zdroj: (autor, 2019)

Celý proces začíná scheduler, který představuje pravidelnou úlohu spouštějící se v pravidelných intervalech. V tomto případě není potřeba lidský vstup, ani grafické uživatelské rozhraní. Proces odstranění uživatelů určitě není nikterak kriticky důležitý pro chod organizace, a proto jeho provedení dvakrát měsíčně postačí. Je vykonáván jedním shell skriptem, který má tři pomyslné fáze.

7.6.1 Identifikace neaktivních uživatelů

Prvním krokem pro odstranění neaktivních uživatelů získání listu všech uživatelů, kteří aktivně na platformu přistupují nebo přistupovali. To je zjištěno podle toho, zda mají založený domovský adresář na Linux serverech. Pomocí příkazu

ls -C /home

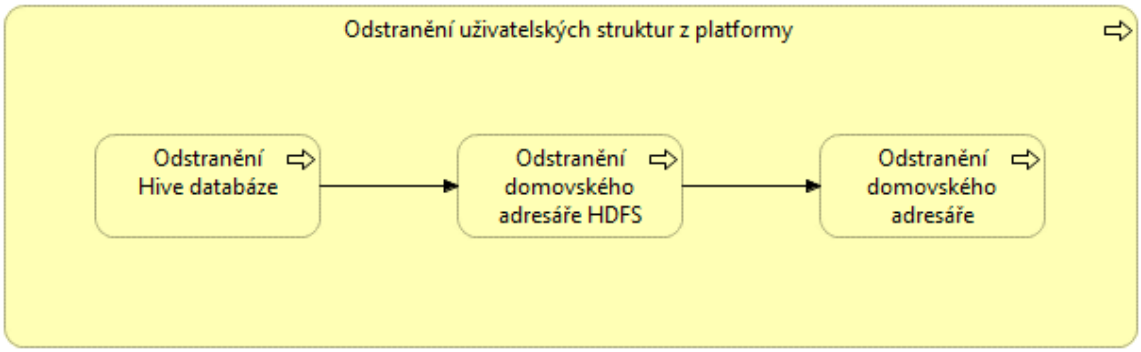
jsou do listu uloženi všichni uživatelé platformy, ať již jsou aktivní nebo ne. Následně je na každého z nich aplikován příkaz:

getent passwd (jméno uživatele)

a v případě, že je návratová hodnota dotazu 0, je jasné, že uživatel již není v LDAP, ani neexistuje lokálně, tudíž je možné vymazat všechny jeho objekty z platformy. Takovýto uživatelé jsou přidáni do nového listu, který označuje všechny uživatelé, kteří mohou být smazáni. Jeden po druhém jsou následně mazáni.

7.6.2 Odstranění uživatelských struktur

Z předchozí fáze skriptu byli identifikování uživatelé, kteří se již nenachází v LDAP, což v překladu znamená, že již ve firmě nepracují a jejich uživatelské prostory na platformě budou kompletně smazány. Při odstraňování uživatelských struktur dochází u každého z uživatelů určeného ke smazání k přesnému opaku toho, co se děje při jejich zakládání. Tyto operace znázorňuje následující obrázek:



Obrázek 20 - Odstranění uživatelských struktur, zdroj: (autor, 2019)

Úplné odstranění uživatelské struktury je dokončeno po úspěšném provedení těchto tří operací.

7.6.2.1 Odstranění Hive databáze

Odstranění Hive databáze je vykonáno příkazem:

drop database if exists (název databáze) cascade

Klauzule if exists opět jako při zakládání způsobuje, že příkaz neselže v případě, že je databáze například již smazaná nebo nikdy neexistovala. Při použitá klauzule cascade na konci příkazu budou nejprve smazány všechny tabulky, které se v databázi nachází a až poté bude vymazána celá databáze.

7.6.2.2 Odstranění domovského adresáře HDFS

Odstranění domovského adresáře z HDFS je vykonáno velmi snadno příkazem:

hdfs dfs -rm -r /cesta/ke/složce

Zde není žádný parametr pro pokračování v případě, příkaz selže, nicméně je ošetřeno, že celý skript neselže v případě, že je složka již smazána nebo nikdy neexistovala.

7.6.2.3 Odstranění domovského adresáře

Pro kompletní odstranění domovského adresáře uživatele je použit příkaz:

rm -r /cesta/ke/složce

Při procesu zakládání byla domovská složka uživatele založena na všech serverech v clusteru, a proto je také potřeba jí ze všech serverů i smazat. Stejně jako při zakládání je operace provedena pomocí příkazu

ssh

v kombinaci s prvním uvedeným. Finální příkaz pro smazání domovského adresáře uživatele potom vypadá takto:

```
Servers = ( „server1“ „server2“ „server3“ „serverN“ )

for server in ${servers[@]}; do

    ssh ${server} bash -c ‘rm -r /cesta/ke/složce‘

done
```

Po úspěšném provedení je odstraněn domovský adresář uživatele ze všech serverů v clusteru.

7.6.3 Odstranění práv uživatele

Odebrání práv uživateli je téměř vyřešené tím, že byly smazány domovské složky uživatele. V jeho HDFS složce se nacházel privátní klíč a Keytab, kterým se ověřoval vůči službám v clusteru. V případě, že by se někdo před smazáním domovského Linux adresáře dostal k privátnímu klíči, je mu i tak k ničemu, vzhledem k tomu, že veřejný klíč patřící k odcizenému privátnímu klíči byl přidán na seznam autorizovaných klíčů přímo v domovské složce, která již neexistuje.

Jediným oprávněním, které tedy zůstává je založená role v Sentry. V případě odstranění skupiny uživatele, která byla na roli navázána z LDAP, dojde i k přerušení vazby mezi rolí a skupinou. I tak však role zůstává evidována, jen není nikomu přiřazena. Vzhledem k tomu, že v předchozí fázi došlo i k odstranění celé Hive databáze, nemá tedy role přiřazeny ani žádná práva, proto je nepoužitelná a nemá cenu jí nadále držet. Její odstranění proběhne po spuštění Hive/Impala dotazu:

drop role (název role);

Po úspěšném dokončení tohoto dotazu je uživatel kompletně vymazán a není dále schopen platformu využívat.

7.6.4 Odstranění uživatele z MariaDB

Potom, co jsou všechny struktury i přístupy uživatele odstraněny z platformy, zbývat ho odstranit z databáze MariaDB, kde je vedena evidence všech objektů založených na platformě a jejich provázání. V případě, že je odmazán uživatel, který je stále přiřazen (a evidován) v nějakém Workspace, je třeba vymazat všechny jeho záznamy, nejen z tabulky uživatelů.

7.7 Shrnutí

Při odstraňování neaktivních uživatelů probíhá mazání všeho, co vzniklo v procesu zakládání uživatele na platformě. Zásadní rozdíl mezi procesy spočívá v tom, že zakládání uživatele vyžaduje schválení správce platformy a také probíhá jednotlivě. Naproti tomu mazání neaktivních uživatelů probíhá masově (jsou sesbíráni všichni neaktivní uživatelé a postupně mazáni) a není potřeba žádné schvalování. Tento proces také nevyžaduje žádný lidský vstup a je prováděn dvakrát za měsíc (dle uvážení Hadoop administrátora může být i častěji).

7.8 Synchronizace uživatelů platformy

Předpokládá se, že implementace procesů pro zakládání a mazání uživatelů probíhá až potom, co je platforma hotová. To znamená, že je platforma již využívaná, nicméně, v tabulce uživatelů žádný uživatel ještě není. Proto je potřeba tyto uživatele synchronizovat s tabulkou uživatelů, díky které získáme přehled o uživatelích, kteří na platformu přistupují. Tato činnost bude prováděna opět shell skriptem a bude pomocí Cronu nastavena, aby byla spouštěna jednou denně (1:00 v noci). Zároveň zamezíme tomu, že někdo celý proces založení obejde, případně o něm nebude vědět a zažádá si sám o založení. V tu chvíli by uživatel na platformě fyzicky existoval, ale nešlo by ho přiřadit do žádného Workspace pomocí implementovaných procesů, protože co není v tabulce uživatelů, jako by na platformě vůbec neexistovalo.

V prostředí, ve kterém jsou user management procesy implementovány mají možnost se přihlásit všichni uživatelé, kteří jsou v LDAP, ale ne všichni to udělají. Za aktivní uživatele platformy jsou proto považováni všichni uživatelé, kteří se přihlásili alespoň jednou, a tudíž mají založený domovský adresář na serveru, na který se přihlásili. Tento server je téměř vždy Edge server, který je považován za bránu do okolního světa a server pro uživatelský přístup. Nejprve jsou tedy sesbíráni všichni uživatelé pomocí příkazu:

ls -C /home

A poté je pro každého z nich poslán dotaz na existenci uživatele příkazem:

getent passwd (jméno uživatele)

V případě, že návratová hodnota tohoto příkazu je 0, znamená to, že uživatel stále na platformě existuje. Podle UID[27] je dále poznat, jestli byl uživatel založen lokálně nebo v LDAP. Uživatelé s nízkým UID jsou typicky zakládaní lokálně, naopak vysoké UID uživatele je známkou toho, že byl uživatel založen skrze LDAP. Odkud však byl uživatel založen není pro tuto úlohu nutné vědět, důležité je to, že uživatel na platformu někdy přistoupil a stále existuje. Posledním krokem synchronizace je uložení všech identifikovaných uživatelů do databáze. To je provedeno příkazem pro:

mysql -u user -p -e "insert into role_based_access.user (username) values ('${user}')

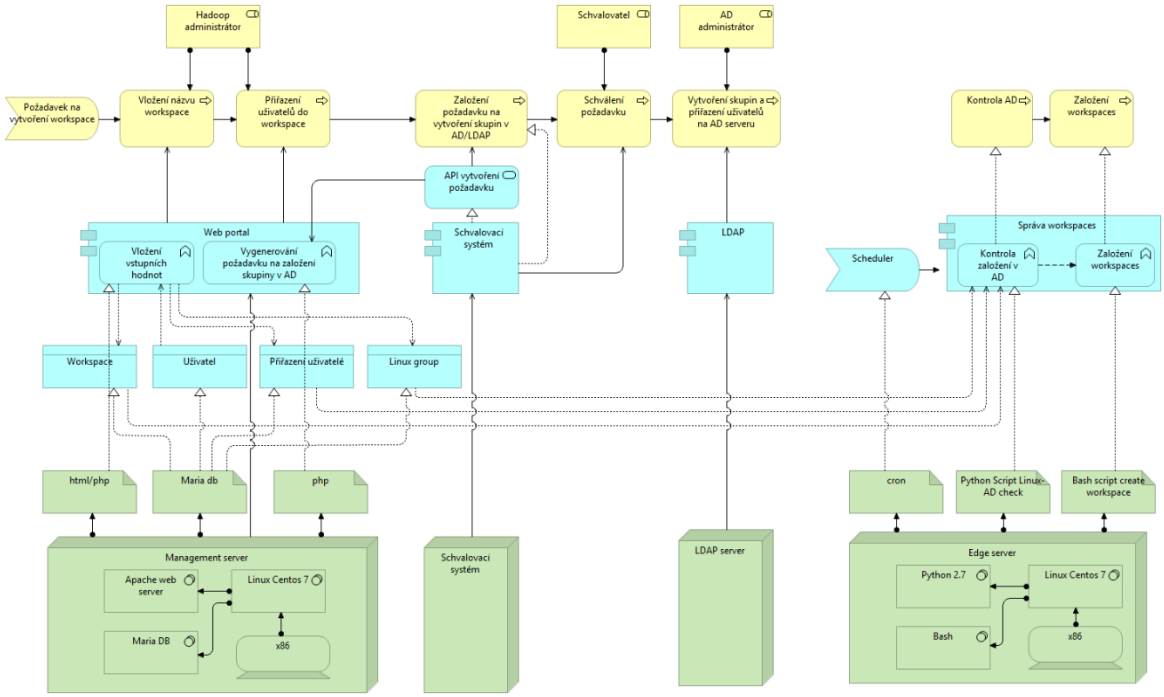
pro každého uživatele zvlášť. Tímto procesem je zajištěn všeobecný přehled o uživatelích přistupujících na platformu.

[27] Identifikátor uživatele v Unixových operačních systémech

7.9 Založení Workspace

Proces založení Workspace je o trochu složitější než založení uživatele na platformě. Nejprve je potřeba definovat pojem Workspace. V překladu „prostor pro práci“ je skutečně nějaký prostor, v tomto případě na Hadoop platformě, který je určený pro konkrétní skupinu uživatelů. Typicky je tato skupina uživatelů definována jako konkrétní oddělení firmy, například CRM. Oddělení CRM bude vytvořeno prostředí, do kterého si budou moci zaměstnanci sdílet všechny potřebné dokumenty, zakládat databázové tabulky, vytvářet analýzy a zároveň budou vědět, že se k těmto artefaktům nedostane nikdo nepovolaný.

Při zakládání Workspace probíhají stejné operace jako při zakládání uživatele. Nejprve je založena struktura celého Workspace (Linux složka, HDFS složka a Hive databáze) a poté jsou vytvořeny práva pro členy Workspace. Přiřazování práv je však zde o něco složitější. Celý proces obecně znázorňuje následující diagram:



Obrázek 21 - Založení Workspace, zdroj: (autor, 2019)

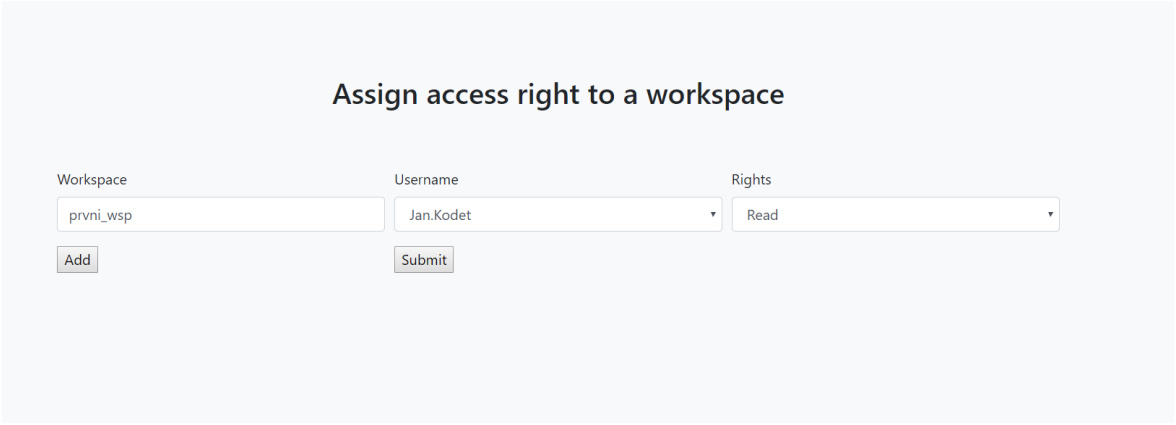
7.9.1 Požadavek na založení Workspace

Celý proces začíná schválením požadavku na založení Workspace. Kromě vlastníka platformy musí tento požadavek schválit i vlastník Workspace, který potvrzuje jeho přiřazené členy. Například CRM Workspace schvaluje i vedoucí pracovník tohoto oddělení. Požadavek je vložen do libovolného schvalovacího systému, v diagramu je znázorněná

Jira[28], která je velmi oblíbeným nástrojem nejen pro schvalování požadavků, ale také evidenci pracovních úkolů. Značnou výhodou tohoto systému je také REST[29] API, kterým disponuje.

7.9.2 Vložení názvu Workspace a přiřazení uživatelů

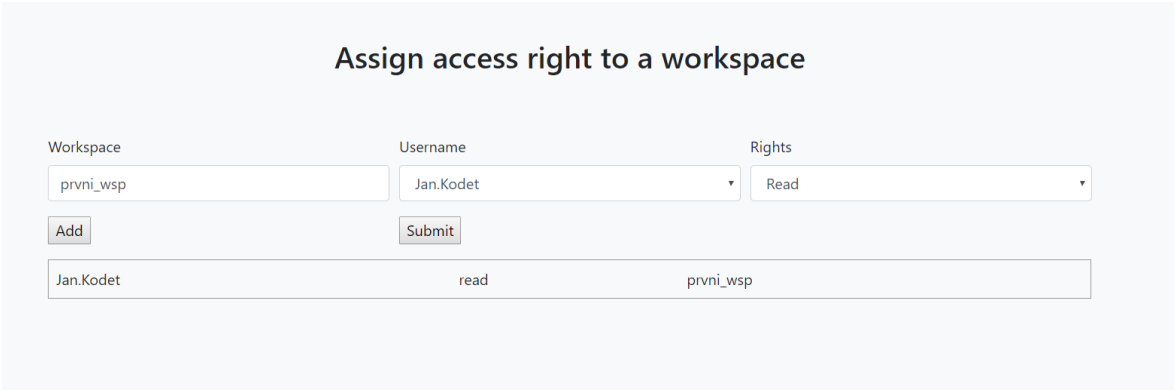
Na základě schválené žádosti o Workspace zadává administrátor Hadoop platformy název Workspace a přiřazuje do něj vybrané uživatele na webovém portálu. Výběr uživatelů probíhá ze seznamu, který je generován dynamicky ze seznamu uživatelů registrovaných v předchozím procesu založení uživatele. Tato činnost je jediným vstupem administrátora Hadoop platformy a pro něj potvrzením zadaných informací končí. Formulář pro vyplnění údajů týkajících se Workspace vypadá následovně:



Obrázek 22 - Formulář pro založení Workspace, zdroj: (autor, 2019)

[28] Schvalovací systém
[29] Representational State Transfer – architektura rozhraní, navržená pro distribuované prostředí, používané pro výměnu dat pomocí http volání

Hadoop administrátor u přiřazených uživatelů určí, jaká práva mají mít. Vybírá ze dvou možností – read, write. Stisknutím tlačítka Add přidá vybraného uživatele do tabulky pod tlačítkem Submit. Zde je shrnuto, jaké záznamy se budou ukládat:



Obrázek 23 - Přiřazení uživatele do Workspace, zdroj: (autor, 2019)

Jakmile vybere všechny uživatele a jejich práva, zmáčkne tlačítko „Submit“. Poté, co je založení Workspace potvrzeno, následují další operace, které se dějí automaticky:

- Uložení názvu Workspace do databáze
- Uložení názvu vygenerovaných Linuxových skupin do databáze
- Uložení údaje o přiřazení uživatele do Workspace do databáze
- Uložení údaje o přiřazení Linuxové skupiny k Workspace
- Uložení údaje o přiřazení uživatele do Linuxové skupiny na základě vybraných práv

Tyto informace jsou dále zpracovávány v další fázi procesu.

7.9.3 Založení požadavku na vytvoření skupin v AD/LDAP

Poté co jsou všechny údaje automaticky uloženy do databáze, probíhá generování a zakládání požadavku v schvalovacím systému na založení skupin (které jsou také automaticky generovány). Kromě vytvoření skupin je potřeba i přiřadit konkrétní uživatele do nich, na což se musí také vygenerovat žádost.

Tyto žádosti generuje webový portál. Všechny úlohy, které má webový portál vykonávat budou napsané v programovacím jazyku Php nebo Javascript.

Tento krok bude v této práci zjednodušen na pouhé vygenerování textového souboru, ve kterém budou sepsané operace, které se musí provést v LDAP. Bohužel v prostředí, na kterém návrh a implementace proběhla není k dispozici žádný schvalovací systém. Tento bod je však poměrně důležitý pro maximální automatizaci procesu zakládání Workspace, proto není možné ho vynechat úplně.

7.9.4 Schválení požadavku

Po automatickém vygenerování žádosti o založení skupin v LDAP následuje další lidský vstup do procesu a tím je schvalování. Žádost o založení a přiřazení uživatelů (v našem případě textový soubor) schvaluje administrátor LDAP serveru, který se o všechny uživatele a skupiny stará centralizovaně.

7.9.5 Vytvoření skupin a přiřazení uživatelů

Jak již víme, práva do databáze jsou řízeny rolemi, které se zakládají v Sentry. Roli poté lze přiřadit pouze na celou Linux skupinu, nikoliv jednotlivce. Proto je potřeba tyto skupiny založit, pro řízení přístupů uživatelů. Pro řízení přístupu do Workspace musí vzniknou dvě Linuxovém skupiny. Jedna s právem zápisu se sufixem _write. Druhá s právem pouze na čtení se sufixem _read. Na základě schváleného požadavku poté vzniknou skupiny na AD/LDAP serveru a do nich jsou přiřazení uživatelé. Ti uživatelé, kteří mají mít práva na zápis do Hive databáze, HDFS a Linux složek, dostanou přidělenou skupinu se sufixem write, uživatelé, kteří mají z Workspace pouze číst (typicky analytické role) mají poté přidělenou skupinu se sufixem _read, ke které jsou navázané příslušná práva. Jakmile jsou všechny skupiny vytvořeny a uživatele přiřazeni do příslušných skupin, proces zdánlivě nepokračuje. Čeká se na spuštění pravidelné úlohy, která má za cíl zkontrolovat připravenost pro samotné založení Workspace a následně fyzicky založit připravené Workspace na platformě.

7.9.6 Kontrola LDAP

Hlavní částí celého procesu, kdy Workspace fyzicky na platformě vzniká je spouštěna cron úlohou, která je nastavena tak, aby se spouštěla dvakrát denně nezávisle na předchozích akcích prováděných na webovém portále nebo v LDAP. Tato úloha je napsána ve skriptovacím jazyce Python. Jeho verze 2.7, běží na každé Linuxové distribuci v rámci operačního systému, proto není třeba řešit žádné problémy s kompatibilitou.

V rámci této úlohy probíhá kontrola, zda je připravené prostředí pro samotné založení Workspace. Prostředí pro Workspace je připravené když:

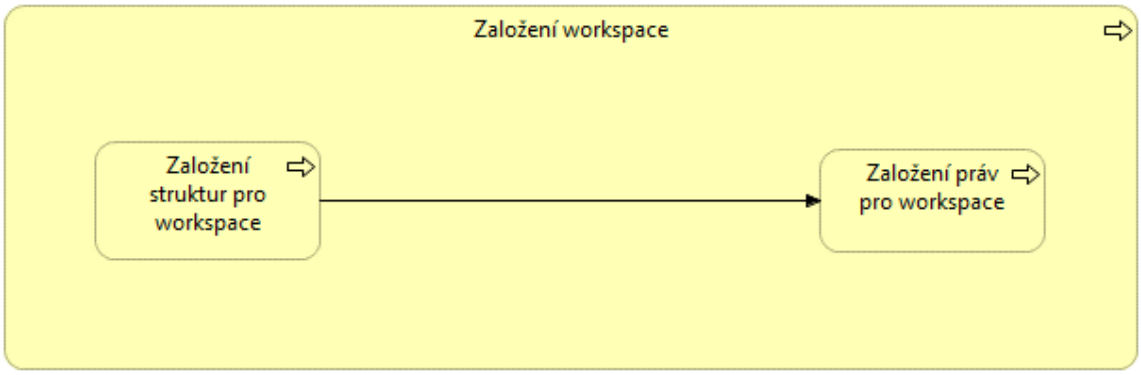
- Existují dvě Linuxové skupiny – (název Workspace)_read a (název Workspace)_write
- Všichni uživatelé, kteří mají do Workspace zapisovat mají přidělenou skupinu se sufixem _write a uživatelé, ti kteří mají pouze číst mají přidělenou skupinu se sufixem _read

V případě, že je Workspace připravený k založení, je označen atributem „is_ready“ v databázi (v tabulce Workspace).

7.10 Vytvoření Workspace

Poté co proběhne kontrola připravenosti Workspace, následuje samotné zakládání, které je realizováno Shell skriptem. Zakládání Workspace je podobně jako mazání uživatelů prováděno masově. První je z databáze SQL dotazem vytažen seznam názvů Workspace, které jsou připraveny k založení (označení připravenosti Workspace probíhá v předchozí fázi – kontrola LDAP). Pro každý Workspace zvlášť následují podobné operace jako při zakládání uživatele – založení struktur Workspace na platformě a založení přístupových práv pro skupiny, ve kterých jsou přiřazeni uživatelé, kteří mají mít do Workspace přístup.

Obecně proces založení Workspace vypadá následovně:



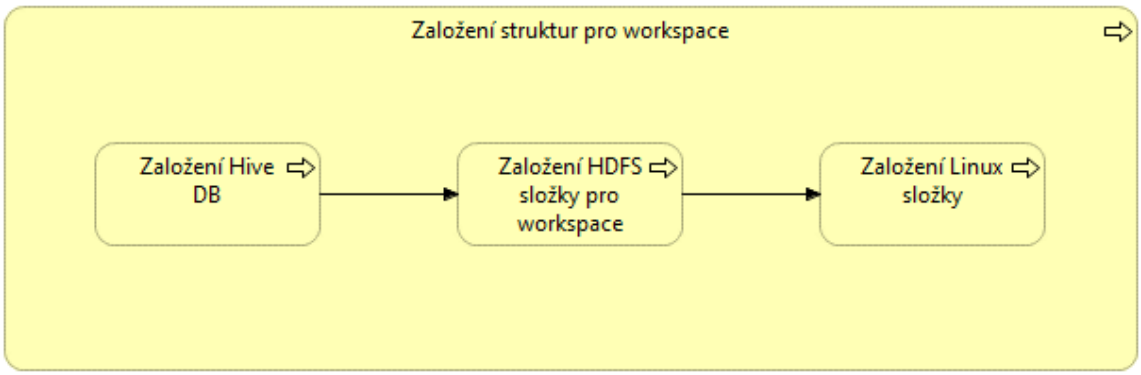
Obrázek 24 - proces založení Workspace, zdroj: (autor, 2019)

Jakmile jsou obě operace úspěšně provedeny, lze Workspace považovat za hotový a přístupný koncovým uživatelům.

7.10.1 Založení struktur pro Workspace

Pro Workspace jsou zakládány stejné objekty jako pro uživatele – Hive databáze, HDFS složka a Linux složka na všech serverech v clusteru.

Celý proces založení struktur pro Workspace znázorňuje následující obrázek:



Obrázek 25 - Založení struktur pro Workspace, zdoj: (autor, 2019)

Jako první probíhá založení Hive databáze pro celý Workspace. Do této databáze mohou uživatelé s právem na zápis libovolně zakládat databázové tabulky, vkládat do nich data a zprostředkovávat je ostatním členům Workspace. Konzumenti těchto dat jsou poté členové Workspace, kteří nemají právo na zápis, a tudíž mohou posílat pouze select dotazy. Druhým krokem je poté založení HDFS složky pro Workspace jejíž vlastníkem bude HDFS (pro administrátorská práva na HDFS je třeba kinit jako HDFS) a skupina, která ponese název Workspace se sufixem _write. Pomocí ACL je zajištěn přístup dvou vzniklých skupin, s tím, že skupina se sufixem _write a tudíž její členové má opět práva na zápis a úpravu souborů uvnitř, skupina se sufixem _read je schopná ze složky pouze číst. Obdobně je založena i Linux složka s jediným rozdílem, že vlastníkem po založení zůstane root.

7.10.1.1 Založení Hive databáze

Hive databáze byla založena již v procesu zakládání uživatele. V tomto případě není syntaxe příkazu jiná, jediný rozdíl je ve jménu. Místo prefixu usr je použitý prefix wsp pro příznak toho, že jde o Workspace. Založení databáze tedy probíhá Hive/Impala dotazem:

create database if not exists wsp_(název Workspace);

Po dokončení dotazu je databáze úspěšně založená nebo již dříve existovala.

7.10.1.2 Založení HDFS složky pro Workspace

HDFS složka je založena taktéž naprosto stejně jako pro uživatele, ovšem s rozdílným jménem a cestou ke složce. HDFS složka pro Workspace je založena příkazem:

hdfs dfs -mkdir -p /Workspaces/(název Workspace)

do složky Workspaces. Další struktura složky není určena a může být měněna uživateli, kteří mají právo zápisu. Po úspěšném provedení příkazu je HDFS složka založena anebo již byla dříve založena.

7.10.1.3 Založení Linux složky

Pro Linuxovou složku Workspace platí to samé jako pro složku na HDFS. Je zakládána do složky /wsp a její další struktura není definována. Po úspěšném provedení příkazu:

mkdir -p /wsp/(název Workspace)

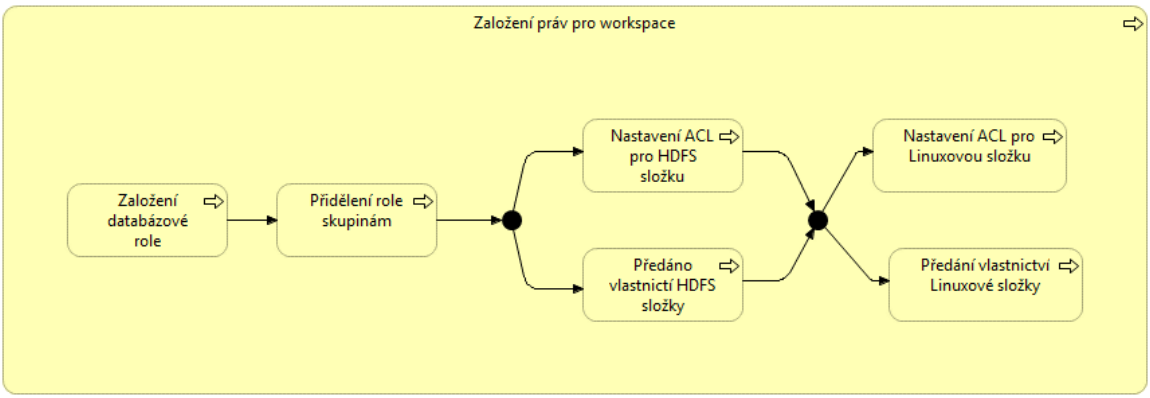
je složka založena anebo se na souborovém systému již nacházela (díky parametru -p).

Vytvoření složky na všech serverech v clusteru opět zajišťuje příkaz SSH:

ssh server1 mkdir -p /wsp/(název Workspace)

7.10.2 Založení práv pro Workspace

Po založení struktur pro Workspace je třeba zajistit, aby se do nich dostali jeho uživatelé. První jsou zakládány databázové role řídící přístup do databáze, zápis do složky dané databáze na HDFS a zároveň i zápis do složky Workspace na HDFS. Následuje přidělení vytvořených rolí skupinám. Tím jsou připraveny přístupy pro uživatele do sdílené Workspace databáze, zbývá vyřešit přístupy do složek na HDFS a Linux serveru. Oproti zakládání uživatelských složek, jsou zde práva do nich řešeny nejen změnou vlastníka složky, ale i definicí ACL. Celý proces zakládání přístupových práv pro Workspace zobrazuje následující diagram:



Obrázek 26 - založení práv pro Workspace, zdroj: (autor, 2019)

7.10.2.1 Založení Sentry role

Pro rozlišení rolí mají všechny role spojené s některým Workspace prefix wsp. Každý Workspace musí mít dvě role kvůli rozdílným právům. Jak již bylo několikrát řečeno, jedna role umožňuje zápis, druhá pouze čtení. Pro vytvoření těchto dvou rolí jsou použity Hive/Impal dotazy:

Create role wsp_(název Workspace)_read;

Create role *wsp_*(název Workspace)_write;

Samotné vytvoření rolí ještě nic neznamená, důležitá jsou práva, která jsou s rolí spojená. Pro přiřazení práva pouze čtení do prvně vzniklé role je použitý Hive/Impala příkaz:

Grant select on wsp_(název Workspace). to role wsp_(název Workspace)_read;*

Kde *wsp_(název Workspace).** znamená právo na všechny objekty v definované databázi. Pro nastavení práv

Pro přiřazení práva zápisu do druhé role je použitý Hive/Impala:

Grant all on wsp_(název Workspace). to role wsp_(název Workspace)_write;*

Pro možnost zápisu do složky Workspace je poté puštěn Hive/Impala dotaz:

Grant all on URI hdfs://Workspaces/(název Workspace) to role wsp_(název Workspace)_write;

Po úspěšném spuštění těchto příkazů jsou role připravené, zbývá je přiřadit konkrétním skupinám.

7.10.2.2 Přidělení role skupinám

Role wsp_(název Workspace)_read je poté přiřazena stejnojmenné skupině pomocí Hive/Impala dotazu:

Grant role wsp_(název Workspace)_read to group wsp_(název Workspace)_read;

To samé pro roli se sufixem _write:

Grant role wsp_(název Workspace)_write to group wsp_(název Workspace)_write;

Tímto je umožněno uživatelům přistupovat do Workspace databáze podle toho, jakou skupinu mají přiřazenou. Stále se však ještě nedostanou do žádné z vytvořených složek určených pro Workspace.

7.10.2.3 Definice ACL pro Workspace HDFS/Linux složku

Pro přístup uživatelů do složky je třeba definovat ACL. Opět jsou nastavena práva pro dvě skupiny podle jejich sufixu. Pro zápis do HDFS složky je v příkazové řádce spuštěn příkaz:

hdfs dfs -setfacl -m group:wsp_(název Workspace)_write:rwx /Workspaces/(název Workspace)

Do Linuxové složky:

setfacl -m group:wsp_(název Workspace)_write:rwx /Workspaces/(název Workspace)

Pro čtení z HDFS složky poté:

hdfs dfs -setfacl -m group:wsp_(název Workspace)_write:rwx /Workspaces/(název Workspace)

Linux složka:

setfacl -m group:wsp_(název Workspace)_write:rwx /Workspaces/(název Workspace)

7.10.2.4 Předání vlastnictví HDFS/Linux složky

Příkaz pro předání vlastnictví složky je stejné na HDFS jako v Linux prostředí, s jediným rozdílem. Pro změnu vlastníka složky v Linux prostředí je použitý příkaz:

chgrp (název skupiny) /cesta/ke/složce

Zatímco pro HDFS platí příkaz:

hdfs dfs -chgrp (název skupiny) /cesta/ke/složce

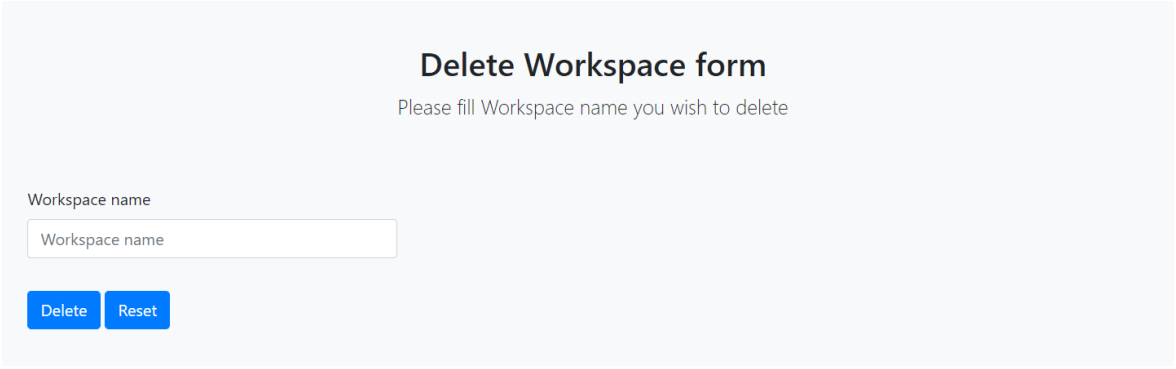
Pouze je před daný příkaz přidáno „hdfs dfs“, čímž je serveru řečeno, aby akci provedl na HDFS.

7.10.3 Shrnutí

V této kapitole byl řešený proces založení Workspace. Nejprve bylo třeba definovat pojem Workspace, určit, jaké objekty ho tvoří a navrhnout řešení automatizace procesu. Objekty byly definovány stejně jako při zakládání uživatele, respektive uživatelského prostoru – Databáze Hive, HDFS složka a Linuxová složka. Zásadní rozdíl, který však bylo třeba vyřešit bylo přiřazení více uživatelů do založených objektů a zároveň rozlišit přístupová práva. Někteří uživatelé mají pouze právo na čtení, zatímco jiní mohou i zapisovat. To je možné rozlišit pomocí členství ve skupinách, a proto bylo zapotřebí zapojení další entity – LDAP. Do procesu tedy vstoupilo i generování žádostí (přímo do schvalovacího systému) na založení dvou Linuxových skupin, které rozlišují uživatelský přístup do Workspace. Generování těchto žádostí však bylo implementováno pouze formou generování textového souboru, ve kterém je napsaný sled událostí, které musí být provedeny v LDAP, a to z důvodu absence některého ze schvalovacích nástrojů (například JIRA). Nezávisle na tom, kdy budou požadované operace v LDAP vykonány, probíhá nově na platformě pravidelná kontrola, která jednak kontroluje založení skupin a přiřazení uživatelů do nich a jednak spouští samotný Shell skript pro založení Workspace, v případě, že je vše připraveno.

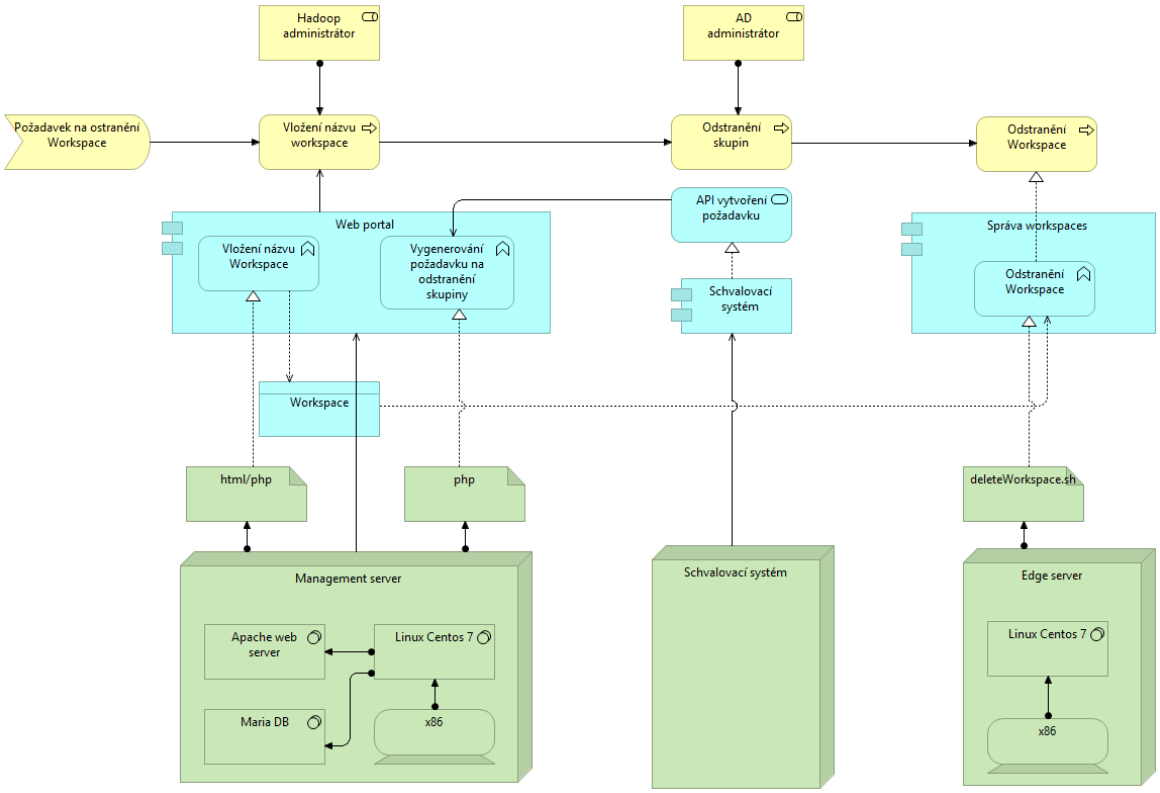
7.11 Odstranění Workspace

Proces odstranění Workspace je stejně jako proces odstranění uživatele opak toho, co se děje při zakládání. Na rozdíl od mazání neaktivních uživatelů, začíná odstranění Workspace schválenou žádostí, neprobíhá automaticky v pravidelném intervalu. Po schválení žádosti o odstranění Workspace zadává administrátor název Workspace.



Obrázek 27 - Vložení názvu Workspace, zdroj: (autor, 2019)

Po potvrzení formuláře proběhne proces mazání, který je znázorněný následujícím diagramem:



Obrázek 28 - Odstranění Workspace, zdroj: (autor, 2019)

Aby byl Workspace úplně smazán z platformy, je potřeba smazat jeho celou strukturu, která vznikla při jeho zakládání:

- Hive databáze
- Linuxová složka
- HDFS složka

Pro odstranění těchto struktur jsou použity stejné příkazy jako v kapitolách 7.6.2.1, 7.6.2.2 a 7.6.2.3. Poté, co jsou tyto struktury smazány, je potřeba založit žádost o smazání dvou skupin Workspace – (název Workspace)_read a (název Workspace)_write. Posledním krokem pro úplné odstranění Workspace je odstranění dvou rolí v Sentry s opět stejným názvem (název Workspace)_read a (název Workspace)_write.

Tím je Workspace zcela odstraněn z platformy.

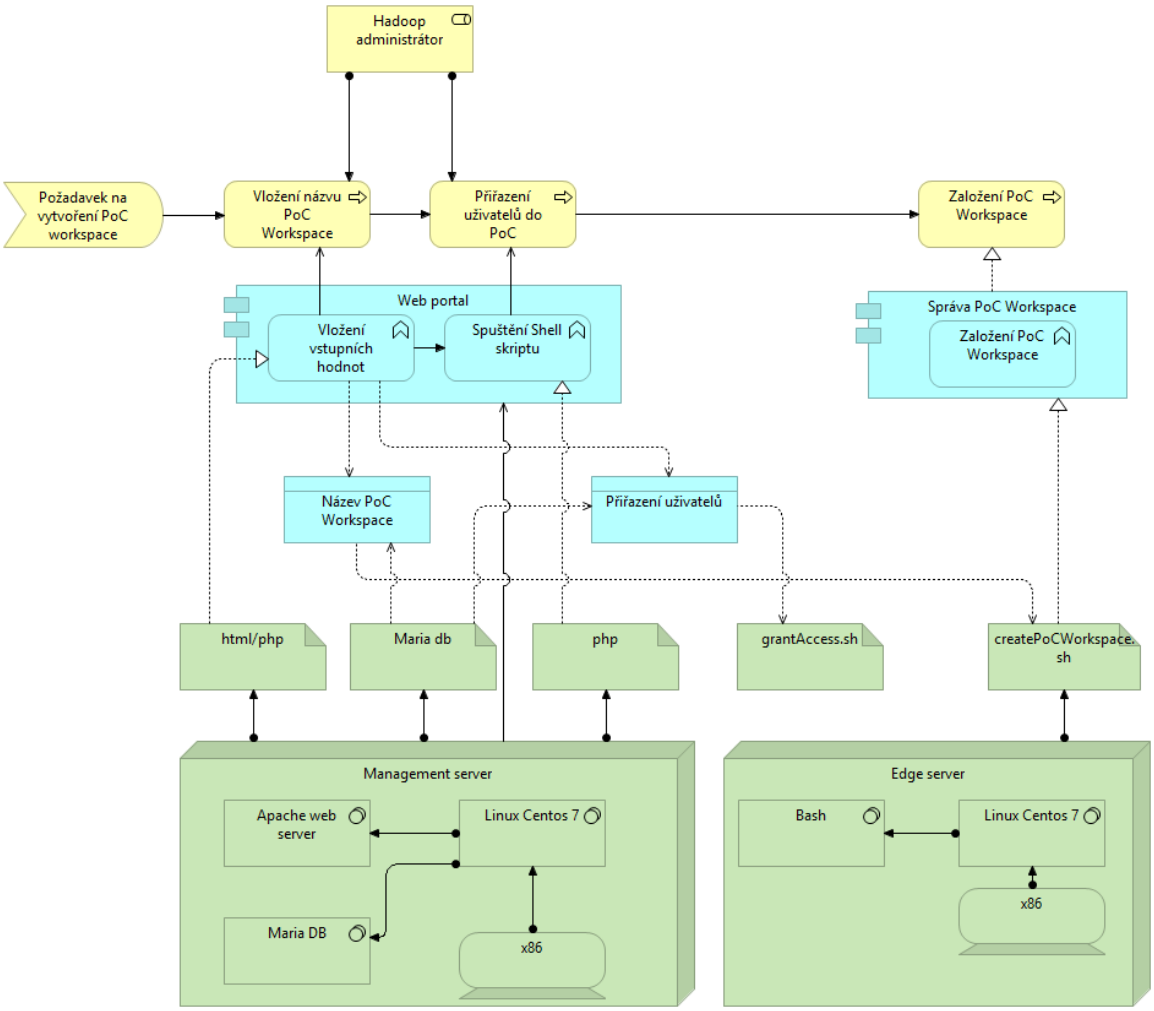
7.12 Založení PoC Workspace

Nevýhodou zakládání Workspace je, že i přes snahu maximální automatizace procesu je při něm potřeba zásah člověka při schválení žádosti o přidělení uživatelů do skupin. Nejistý je také proces, jakým zakládání objektů v LDAP probíhá. V některých organizacích je tento proces prováděn automaticky hned po schválení ve schvalovacím systému. Ne všichni však tento proces automatizují, a tudíž se na to nelze spoléhat. Kvůli zmíněným faktům zůstává proces vytvoření Workspace otázkou minimálně několika dní.

Pro naléhavé případy, kdy je potřeba začít práci na platformě co možná nejdříve byl definován takzvaný PoC Workspace. PoC (Proof of Concept) Workspace je speciální druh Workspace. Jeho struktura a definice je stejná jako u klasického Workspace. Je to místo určené pro skupinovou práci uživatelů platformy. Zásadní rozdíl je však v tom, že je primárně určený menšímu počtu lidí pro účely malých projektů nebo PoC řešení, které je často třeba začít řešit co možná nejrychleji.

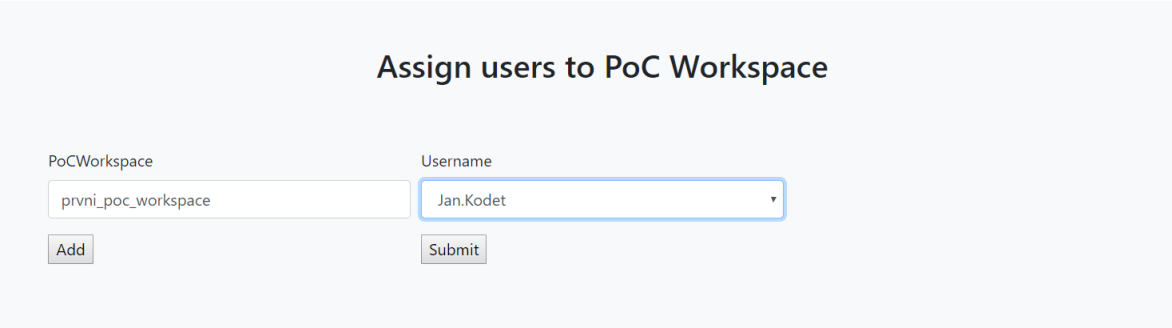
PoC Workspace nepotřebuje schvalování, Hadoop administrátor je schopný ho založit na počkání. Proto z celého procesu vypadává operace spojené se schvalovacím systémem a kontrolou připravenosti Linuxového prostředí. Uživatelé, kteří jsou do něj přiřazeni mají všichni bez výjimky právo na zápis a mohou libovolně měnit tabulky v databázi, respektive strukturu složek. Všechny práva jsou spojeny s unikátní skupinou uživatele. Je však třeba brát v povědomí, že takovou skupinu nemusí mít uživatelé na každé platformě. Zpravidla se však unikátní skupina pro každého uživatele vytváří

Nákres procesu vypadá následovně:



Obrázek 29 - Založení PoC Workspace, zdroj: (autor, 2019)

Proces začíná na webovém portálu, kam Hadoop administrátor zadává název PoC Workspace a přiřazuje do něho uživatele.

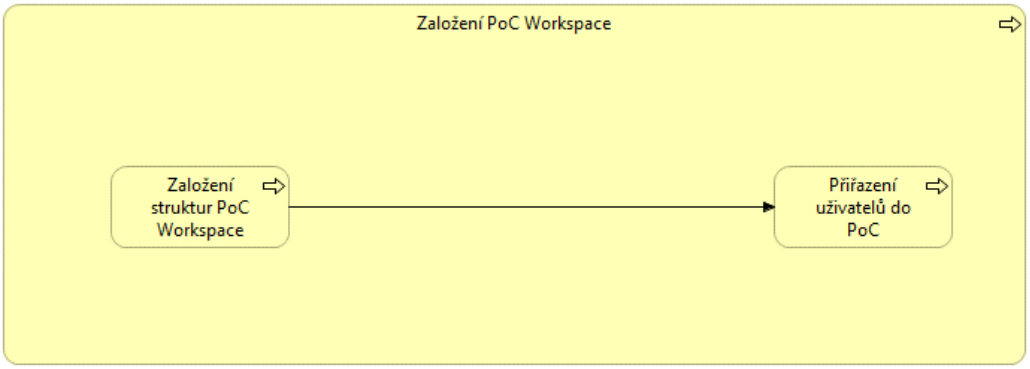


Obrázek 30 - Přiřazení uživatelů do PoC Workspace, zdroj: (autor, 2019)

Po potvrzení výběru se na pozadí automaticky provedou následující operace:

- Název PoC Workspace se uloží do tabulky PoC_Workspace
- Do tabulky User_poc_Workspace se uloží ID PoC Workspace a všech vybraných uživatelů
- Rovnou se spustí Shell skript, který založí PoC Workspace a přiřadí do něj uživatele

Pro znázornění akcí, které vykonává zakládací skript si lze vypůjčit obrázky z procesu založení Workspace, jen přidat slovo PoC.



Obrázek 31 - Založení PoC Workspace, zdroj: (autor, 2019)

Jako první je založena struktura, následně jsou přiřazení jednotlivý uživatele.

7.12.1 Založení struktur PoC Workspace

Pro založení struktury PoC Workspace jsou založené stejné objekty jako při zakládání Workspace (viz. kapitola 7.8.18):

- Hive databáze
- Linuxová složka
- HDFS složka

Pro založení jsou opět použité stejné příkazy jako v kapitolách 7.9.7.2, 7.9.7.3 a 7.9.7.4.

7.12.2 Přiřazení uživatelů do PoC Workspace

Hlavním rozdílem mezi založením Workspace a PoC Workspace je způsob přiřazování uživatelů. Proces založení PoC Workspace je zjednodušen tím, že není potřeba zakládat Linuxovou skupinu, která přístupy řídí. Místo toho je využitý fakt, že každý uživatel má svou unikátní skupinu, která se odvíjí od uživatelského jména. Například uživatel platformy *Jan.Kodet* má unikátní skupinu *jan_kodet*, ve které je úplně sám, a tudíž jde o unikátní identifikátor uživatele. Přístup do Linuxové a HDFS složky je tudíž řízen pomocí definice

ACL, kam jsou definovány plné přístupy konkrétnímu uživateli. Zde však zmíněná unikátní skupina ještě potřeba není, ACL lze definovat i na uživatelské jméno. Při zakládání přístupu do Hive databáze už ale potřebná je. Pro PoC Workspace je vytvořena jedna role s právy na zápis a do té jsou jednotlivý uživatelé postupně přidáni pomocí svých unikátních skupin.

Je nutné brát v úvahu, že proces vytvoření PoC Workspace je závislý na unikátních skupinách uživatelů. Ze zkušenosti však vychází, že v některém prostředí se unikátní skupiny uživatelů nevytváří. V tu chvíli nelze použít koncept PoC Workspace a musí být navrhnuto úplně jiné řešení pro rychlý přístup k platformě.

7.12.3 Shrnutí

Koncept PoC Workspace se snaží odstranit čekací dobu na založení potřebného prostoru pro skupinovou práci na platformě. Úplně vynechává část se zakládáním Linuxových skupin, které v klasickém Workspace řídí přístup k vytvořeným objektům a také je vynechána kontrola založení těchto skupin. Oproti Workspace, Shell skript pro založení PoC Workspace spouští webový portál, a ne kontrolní skript běžící v pravidelných intervalech.

Závěr

Cílem teoretické části práce bylo představení pojmu Big Data, seznámení s Frameworkem Apache Hadoop, jeho hlavními komponentami, funkcionalitami a možnosti zabezpečení. Všechny tyto pojmy byli v rámci teoretického úvodu detailně vysvětleny a popsány.

Hlavním cílem praktické části práce je automatizace procesů user managementu, které jsou opakovaně prováděny na zabezpečené Hadoop platformě, aby byla usnadněna práce Hadoop administrátorům a aby se snížil čas potřebný na přípravu pracovního prostředí pro datové analytiky.

Prostředí, na kterém probíhala implementace bylo popsáno v kapitole 7.3. Procesy, které je vhodné automatizovat byly definovány na základě častých požadavků od uživatelů. Následně byl vytvořen návrh pro jejich automatizaci za pomoci modelovacího jazyku Archimate. Mezi vybranými procesy byly:

- Založení uživatele na platformě
- Odstranění uživatelů z platformy
- Založení Workspace
- Odstranění Workspace
- Založení PoC Workspace

Všechny výše uvedené procesy byly následně implementovány do reálného Hadoop prostředí. Pro ověření řešení byl spuštěn pilotní režim, do kterého se zapojilo celkem šest lidí operující na dané Hadoop platformě. Sledované metriky byly čas potřebný k vytvoření uživatele/Workspace/PoC Workspace a uživatelská přívětivost pro administrátory. Uživatelská přívětivost se měří těžko, proto vyhodnocení v tomto případě probíhalo formou zpětné vazby. Čtyři lidé uvedli, že jim webové rozhraní, ve kterém operuje uživatel připadá snadné a přehledné. Zbylí dva si stěžovali na formulář pro vytvoření Workspace. V případě chybného přiřazení uživatele do Workspace není k dispozici odebrání uživatele z návrhu přiřazení a je potřeba stránku aktualizovat, čímž je zrušeno i zařazení všech ostatních uživatelů.

Co se týče času, procesy, které nevyžadují schvalování trvají samozřejmě kratší dobu než ty, které ho vyžadují. Následující tabulka porovnává průměrné doby trvání jednotlivých procesů:

Název procesu	**Doba trvání – manuálně**	**Doba trvání – automaticky**
Založení uživatele	10–15 minut	1 minuta
Odstranění uživatele	5–10 minut	1 minuta
Založení PoC Workspace	15–20 minut	3 minuty
Založení Workspace	2 dny	0,5 dne
Odstranění Workspace	1 den	0,5 den

Obrázek 32 – Vyhodnocení, zdroj: (autor, 2019)

U procesu založení Workspace nejde jasně říct, jak dlouho proces trvá. Zvláště při absenci schvalovacího systému nebylo tento proces možné objektivně otestovat. Předpokládáme, že všechny žádosti o LDAP skupiny budou založeny okamžitě, jakmile je schválen požadavek na Workspace. Jejich vyřízení může trvat půl dne a následující den si teprve Hadoop administrátor všimne, že jsou všechny skupiny založeny a začne Workspace vytvářet. Odhadovaná doba uvedená v předchozí tabulce je velice optimistická. To samé platí i o mazání, kde je třeba Linuxové skupiny pouze z LDAP smazat, nicméně i zde vstupuje do procesu jiný proces – schvalování smazání LDAP skupin.

User management na Hadoop platformě je široká oblast a jak bylo řečeno dříve, lze definovat další procesy, které bude možné efektivně zautomatizovat. Kvůli časové náročnosti nebyl implementován proces odstranění PoC Workspace, pro který byl však vytvořen automatický proces zakládání. Vzhledem k jeho vlastnostem je považován za krátkodobý objekt, a proto se předpokládá, že jeho odstranění bude často řešeným problémem. Dalším prostorem pro zlepšení je Webový portál, na jehož uživatelskou přívětivost bylo upozorněno v průběhu pilotního režimu. Konkrétně přidání možnosti měnit zařazení uživatelů do Workspace.

Dalším kritickým bodem pro user management je napojení na schvalovací systém organizace. V tomto bodě dochází k největším prodlevám, jelikož zadávání požadavků pro schválení a jejich následné vyřizování není zábavná činnost a často bývá odkládána. Proto je nutné, aby byly žádosti zakládány samy a zároveň hned po jejich schválení, respektive vyřízení probíhala navazující činnost. V této práci nebylo toto napojení řešeno (pouze formou vypsání potřebných akcí) pro absenci schvalovacího systému schopného komunikace s aplikacemi.

Následující procesy jsou navrhovány pro automatizaci:

- Založení uživatele v MySQL (případně jiné) databázi běžící v clusteru a následné informování příslušného uživatele o založení formou emailu
- Úprava Workspace – Přidání nebo odebrání členů Workspace. Je potřeba vygenerovat žádost o zařazení/vyřazení požadovaného uživatele ze skupiny vytvořené pro Workspace
- Úprava PoC Workspace – Dodatečné zařazení nebo odstranění uživatelů do/z Poc Workspace

Při pohledu na doby trvání jednotlivých procesů, které byly úspěšně implementovány je jasně viditelná úspora času. Hlavní cíl práce byl splněn.

Použitá literatura

1) ACLs [online]. [cit. 2019-04-20]. Dostupné z: https://linuxconfig.org/how-to-manage-acls-on-linux
2) Active Directory [online]. [cit. 2019-04-20]. Dostupné z: http://lide.uhk.cz/fim/student/pokormi2/tnpw1/struktura-ad.html
3) Apache Hadoop Ozone. Apache Hadoop [online]. [cit. 20.04.2019]. Dostupné z: https://hadoop.apache.org/ozone
4) Apache Hue [online]. [cit. 2019-04-20]. Dostupné z: https://www.cloudera.com/products/open-source/apache-hadoop/hue.html
5) Apache Pig Intro [online]. [cit. 2019-04-01]. Dostupné z: https://www.guru99.com/introduction-to-pig-and-hive.html
6) Apache Pig [online]. [cit. 2019-04-01]. Dostupné z: https://hortonworks.com/tutorial/beginners-guide-to-apache-pig
7) Apache Sqoop [online]. [cit. 2019-04-20]. Dostupné z: https://sqoop.apache.org
8) Archimate [online]. [cit. 2019-04-01]. Dostupné z: https://www.visual-paradigm.com/guide/archimate/what-is-archimate
9) BANSAL, Hanish, Saurabh CHAUHAN a Shrey MEHROTRA. Apache Hive Cookbook. 1. 2016. ISBN 1782161090.
10) Facebook na konci roku zastavil pokles uživatelů [online]. [cit. 2019-04-01]. Dostupné z: https://www.mediaguru.cz/clanky/2019/01/facebook-na-konci-roku-zastavil-pokles-uzivatelu
11) CDH. Cloudera. [Online] [Citace: 20. 04 2019.] https://www.cloudera.com/documentation/enterprise/latest/topics/cdh_intro.html.
12) Cloudera merge Hortonworks [online]. [cit. 2019-04-20]. Dostupné z: https://hortonworks.com/blog/welcome-brand-new-cloudera/
13) Data Management [online]. [cit. 2019-04-01]. Dostupné z: https://www.systemonline.cz/business-intelligence/sprava-dat-v-ere-big-data.htm
Dostupné z: https://hadoop.apache.org/ozone
14) DUTCHER, Jenna. 2014. What Is Big Data?. In: Datascience@berkeley Blog [online]. [cit. 2019-04-40]. Dostupné z:http://datascience.berkeley.edu/what-is-big-data/
15) GARTNER. 2013. Big Data Analytics - Free Gartner Research: IT Glossary [online]. [cit. 2019-04-01]. Dostupné z:http://www.gartner.com/it-glossary/big-data/
16) Hadoop security [online]. [cit. 2019-04-20]. Dostupné z: https://www.bmc.com/guides/hadoop-security.html
17) Hadoop [online]. [cit. 2019-04-01]. Dostupné z: https://hadoop.apache.org/
18) HBase [online]. [cit. 2019-04-20]. Dostupné z: https://hbase.apache.org/
19) HDFS Guide. Apache Foundation. [Online] [Citace: 20. 04 2019.] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#FS+Shell
20) Hive formats [online]. [cit. 2019-04-20]. Dostupné z: http://dwgeek.com/hive-different-file-formats-text-sequence-rc-avro-orc-parquet-file.html/

21) Hive Metastore [online]. [cit. 2019-04-20]. Dostupné z: https://community.hortonworks.com/questions/23797/how-the-hive-metastore-works.html
22) HOLUBOVÁ, Irena, Karel MINAŘÍK, David, NOVÁK a Jiří KOSEK. Big Data a NoSQL databáze. Praha: Grada, 2015. ISBN 978-80-247-5466-6. – MapReduce
23) How Hadoop is Used in Organizations [online]. [cit. 20.04.2019]. Dostupné z: https://acadgild.com/blog/hadoop-used-organizations
24) Impala Datatypes [online]. [cit. 2019-04-01]. Dostupné z: https://www.cloudera.com/documentation/enterprise/5-14-x/topics/impala_datatypes.html
25) Import SSL cert [online]. [cit. 2019-04-20]. Dostupné z: https://www.ibm.com/support/knowledgecenter/en/SS6JTX/waveadmin/wave_appf.html
26) Kebreos advantages [online]. [cit. 2019-04-20]. Dostupné z: https://www.computerweekly.com/news/1280096066/Kerberos-advantages
27) Kerberos [online]. [cit. 2019-04-01]. Dostupné z: http://www.hadoopadmin.co.in/kerberos
28) KRB [online]. [cit. 2019-04-20]. Dostupné z: https://www.varonis.com/blog/kerberos-authentication-explained/
29) LDAP [online]. [cit. 2019-04-20]. Dostupné z: http://www.benak.net/ldap
30) Linux permissions [online]. [cit. 2019-04-20]. Dostupné z: https://www.pluralsight.com/blog/it-ops/linux-file-permissions
31) MapReduce model [online]. [cit. 2019-04-20]. Dostupné z: https://www.guru99.com/introduction-to-mapreduce.html
32) Sentry roles [online]. [cit. 2019-04-01]. Dostupné z: https://mapr.com/docs/archive/mapr50/Sentry_28213732.html
33) The V's of Big Data: Velocity, Volume, Value, Variety, and Veracity. Home | XSi [online]. [cit. 29.04.2019]. Dostupné z: https://www.xsnet.com/blog/bid/205405/the-v-s-of-big-data-velocity-volume-value-variety-and-veracity
34) Using Hive [online]. [cit. 2019-04-01]. Dostupné z: https://docs.microsoft.com/cs-cz/azure/hdinsight/hadoop/hdinsight-use-hive
35) What is Map Reduce [online]. [cit. 2019-04-01]. Dostupné z: https://www.quora.com/What-is-Map-Reduce
36) What is YARN in Hadoop? – Quora A place to share knowledge and better understand the world [online]. Dostupné z: https://www.quora.com/What-is-YARN-in-Hadoop
37) What new in Impala [online]. [cit. 2019-04-01]. Dostupné z: https://impala.apache.org/docs/build/html/topics/impala_new_features.html#new_features
38) YARN Capacity Scheduler [online]. [cit. 2019-04-20]. Dostupné z: https://hortonworks.com/blog/yarn-capacity-scheduler/
39) YARN Container [online]. [cit. 2019-04-20]. Dostupné z: https://www.quora.com/What-is-yarn-container
40) YARN guide [online]. [cit. 2019-04-20]. Dostupné z: https://data-flair.training/blogs/hadoop-yarn-tutorial/

41) YARN resourcemanager [online]. [cit. 2019-04-20]. Dostupné z: https://hortonworks.com/blog/apache-hadoop-yarn-resourcemanager/
42) YARN [online]. [cit. 2019-04-01]. Dostupné z: https://hortonworks.com/blog/apache-hadoop-yarn-nodemanager

Seznam obrázků

Obrázek 1 - Design Science Research Methodology, zdroj: (Peffers, a další, 2008) 12
Obrázek 2 - Integrace Hadoopu do infrastruktury podniku, zdroj: (Adastra, 2017) 21
Obrázek 3 - Cloudera Manager, zdroj: (autor, 2019) 23
Obrázek 4 - Komponenty CDH, zdroj: (Cloudera, 2019) 24
Obrázek 5 - průběh MapReduce úlohy, zdroj: (Guru99, 2015) 27
Obrázek 6 - Architektura HDFS, zdroj: (Hadoop, 2018) 29
Obrázek 7 - Zařazení YARN v Hadoop, zdroj: (Data-flair, 2016) 30
Obrázek 8 - ukázka HUE, zdroj: (autor, 2019) 35
Obrázek 9 - Architektura PIG, zdroj: (Guru99, 2017) 36
Obrázek 10 - Jazyk ArchiMate, zdroj: (*Visual-Paradigm,* 2018) 48
Obrázek 11 - konceptuální návrh, zdroj: (autor) 51
Obrázek 12 - Logický návrh databáze, zdroj: (autor) 53
Obrázek 13 -Fyzický návrh databáze, zdroj: (autor) 53
Obrázek 14 - Proces vytvoření uživatele, zdroj: (autor) 56
Obrázek 15 - Formulář pro založení uživatele, zdroj: (autor) 57
Obrázek 16 - Zakládání uživatelských struktur, zdroj: (autor) 57
Obrázek 17 - Založení přístupových práv, zdroj: (autor) 59
Obrázek 18 - Odstranění uživatelů 62
Obrázek 19 - Odstranění uživatelských struktur, zdroj: (autor) 63
Obrázek 20 - Založení Workspace, zdroj: (autor) 67
Obrázek 21 - Formulář pro založení Workspace, zdroj: (autor) 68
Obrázek 22 - Přiřazení uživatele do Workspace, zdroj: (autor) 69
Obrázek 23 - proces založení Workspace 71
Obrázek 24 - Založení struktur pro Workspace, zdoj: (autor) 72
Obrázek 25 - založení práv pro Workspace 73
Obrázek 26 - Vložení názvu Workspace, zdroj: (autor) 76
Obrázek 27 - Odstranění Workspace, zdroj: (autor) 77
Obrázek 28 - Založení PoC Workspace, zdroj: (Autor, 2019) 79
Obrázek 29 - Přiřazení uživatelů do PoC Workspace, zdroj: (autor) 79
Obrázek 30 - Založení PoC Workspace, zdroj: (autor) 80

Přílohy

7.13 Příloha 1

```
SET FOREIGN_KEY_CHECKS=0
;

/* Drop Tables */

DROP TABLE IF EXISTS `Linux_group` CASCADE
;

DROP TABLE IF EXISTS `PoC_Workspace` CASCADE
;

DROP TABLE IF EXISTS `User` CASCADE
;

DROP TABLE IF EXISTS `User_group` CASCADE
;

DROP TABLE IF EXISTS `User_poc_Workspace` CASCADE
;

DROP TABLE IF EXISTS `Workspace` CASCADE
;

DROP TABLE IF EXISTS `Workspace_group` CASCADE
;

DROP TABLE IF EXISTS `Workspace_user` CASCADE
;

/* Create Tables */

CREATE TABLE `Linux_group`
(
          `groupID` INT NOT NULL AUTO_INCREMENT,
          `group_name` VARCHAR(50) NULL,
          `exists_on_all_nodes` SMALLINT NULL,
          CONSTRAINT `PK_Linux Group` PRIMARY KEY (`groupID` ASC)
)
```

```
;

CREATE TABLE `PoC_Workspace`
(
	`pocWorkspaceID` INT NOT NULL AUTO_INCREMENT,
	`poc_Workspace_name` VARCHAR(50) NULL,
	CONSTRAINT `PK_PoC_Workspace` PRIMARY KEY (`pocWorkspaceID` ASC)
)

;

CREATE TABLE `User`
(
	`userID` INT NOT NULL AUTO_INCREMENT,
	`username` VARCHAR(50) NULL,
	`firstname` VARCHAR(50) NULL,
	`lastname` VARCHAR(50) NULL,
	`dt_created` DATE NULL COMMENT 'now',
	CONSTRAINT `PK_User` PRIMARY KEY (`userID` ASC)
)

;

CREATE TABLE `User_group`
(
	`ID` INT NOT NULL AUTO_INCREMENT,
	`userID` INT NULL,
	`groupID` INT NULL,
	CONSTRAINT `PK_user_group` PRIMARY KEY (`ID` ASC)
)

;

CREATE TABLE `User_poc_Workspace`
(
	`ID` INT NOT NULL AUTO_INCREMENT,
	`pocWorkspaceID` INT NULL,
	`userID` INT NULL,
	CONSTRAINT `PK_User_poc_Workspace` PRIMARY KEY (`ID` ASC)
)

;

CREATE TABLE `Workspace`
```

```
(
        `WorkspaceID` INT NOT NULL AUTO_INCREMENT,
        `Workspace_name` VARCHAR(50) NULL,
        `ready_to_create` SMALLINT NULL DEFAULT 0,
        CONSTRAINT `PK_Workspace` PRIMARY KEY (`WorkspaceID` ASC)
)

;

CREATE TABLE `Workspace_group`
(
        `ID` INT NOT NULL AUTO_INCREMENT,
        `WorkspaceID` INT NULL,
        `groupID` INT NULL,
        CONSTRAINT `PK_Workspace_group` PRIMARY KEY (`ID` ASC)
)

;

CREATE TABLE `Workspace_user`
(
        `ID` INT NOT NULL AUTO_INCREMENT,
        `WorkspaceID` INT NULL,
        `userID` INT NULL,
        CONSTRAINT `PK_Workspace_user` PRIMARY KEY (`ID` ASC)
)

;

/* Create Primary Keys, Indexes, Uniques, Checks */

ALTER TABLE `Linux_group`
 ADD CONSTRAINT `Indx` UNIQUE (`group_name` ASC)
;

ALTER TABLE `PoC_Workspace`
 ADD CONSTRAINT `Unq` UNIQUE (`poc_Workspace_name` ASC)
;

ALTER TABLE `User`
 ADD CONSTRAINT `Unique_indx` UNIQUE (`username` ASC)
;

ALTER TABLE `User_group`
 ADD CONSTRAINT `unq` UNIQUE (`userID` ASC, `groupID` ASC)
```

```
;

ALTER TABLE `User_group`
 ADD INDEX `IXFK_user_group_Linux Group` (`groupID` ASC)
;

ALTER TABLE `User_group`
 ADD INDEX `IXFK_user_group_User` (`userID` ASC)
;

ALTER TABLE `User_group`
 ADD INDEX `IXFK_user_group_User_02` (`userID` ASC)
;

ALTER TABLE `User_poc_Workspace`
 ADD CONSTRAINT `unq` UNIQUE (`pocWorkspaceID` ASC, `userID` ASC)
;

ALTER TABLE `User_poc_Workspace`
 ADD INDEX `IXFK_User_poc_Workspace_PoC_Workspace` (`pocWorkspaceID` ASC)
;

ALTER TABLE `User_poc_Workspace`
 ADD INDEX `IXFK_User_poc_Workspace_User` (`userID` ASC)
;

ALTER TABLE `Workspace`
 ADD CONSTRAINT `Indx1` UNIQUE (`Workspace_name` ASC)
;

ALTER TABLE `Workspace_group`
 ADD CONSTRAINT `Indx` UNIQUE (`WorkspaceID` ASC, `groupID` ASC)
;

ALTER TABLE `Workspace_group`
 ADD INDEX `IXFK_Workspace_group_Linux Group` (`groupID` ASC)
;

ALTER TABLE `Workspace_group`
 ADD INDEX `IXFK_Workspace_group_Workspace` (`WorkspaceID` ASC)
;

ALTER TABLE `Workspace_user`
 ADD CONSTRAINT `Inxd` UNIQUE (`userID` ASC, `WorkspaceID` ASC)
;
```

```
ALTER TABLE `Workspace_user`
 ADD INDEX `IXFK_Workspace_user_User` (`userID` ASC)
;

ALTER TABLE `Workspace_user`
 ADD INDEX `IXFK_Workspace_user_User_02` (`ID` ASC)
;

ALTER TABLE `Workspace_user`
 ADD INDEX `IXFK_Workspace_user_User_03` (`userID` ASC)
;

ALTER TABLE `Workspace_user`
 ADD INDEX `IXFK_Workspace_user_Workspace` (`WorkspaceID` ASC)
;

/* Create Foreign Key Constraints */

ALTER TABLE `User_group`
 ADD CONSTRAINT `FK_user_group_Linux Group`
         FOREIGN KEY (`groupID`) REFERENCES `Linux_group` (`groupID`) ON DELETE Restrict
ON UPDATE Restrict
;

ALTER TABLE `User_group`
 ADD CONSTRAINT `FK_user_group_User_02`
         FOREIGN KEY (`userID`) REFERENCES `User` (`userID`) ON DELETE Cascade ON UPDATE
Restrict
;

ALTER TABLE `User_poc_Workspace`
 ADD CONSTRAINT `FK_User_poc_Workspace_PoC_Workspace`
         FOREIGN KEY (`pocWorkspaceID`) REFERENCES `PoC_Workspace` (`pocWorkspaceID`) ON
DELETE Cascade ON UPDATE Restrict
;

ALTER TABLE `User_poc_Workspace`
 ADD CONSTRAINT `FK_User_poc_Workspace_User`
         FOREIGN KEY (`userID`) REFERENCES `User` (`userID`) ON DELETE Cascade ON UPDATE
Restrict
;

ALTER TABLE `Workspace_group`
 ADD CONSTRAINT `FK_Workspace_group_Linux Group`
```

```
         FOREIGN KEY (`groupID`) REFERENCES `Linux_group` (`groupID`) ON DELETE Cascade ON
UPDATE Restrict
;

ALTER TABLE `Workspace_group`
 ADD CONSTRAINT `FK_Workspace_group_Workspace`
         FOREIGN KEY (`WorkspaceID`) REFERENCES `Workspace` (`WorkspaceID`) ON DELETE
Cascade ON UPDATE Restrict
;

ALTER TABLE `Workspace_user`
 ADD CONSTRAINT `FK_Workspace_user_User_03`
         FOREIGN KEY (`userID`) REFERENCES `User` (`userID`) ON DELETE Cascade ON UPDATE
Restrict
;

ALTER TABLE `Workspace_user`
 ADD CONSTRAINT `FK_Workspace_user_Workspace`
         FOREIGN KEY (`WorkspaceID`) REFERENCES `Workspace` (`WorkspaceID`) ON DELETE
Cascade ON UPDATE Restrict
;

SET FOREIGN_KEY_CHECKS=1
;
```