PROJEKT

Multitenantné operačné centrum kybernetickej bezpečnosti riešené
ako otvorená cloudová služba s prvkami strojového učenia

Previazané s balíkom KPB1 – Návrh virt. riešenia

Previazané s výstupom
V3 – Popis vybraného riešenia spĺňajúceho stanovené kritéria

typ – dokumentácia

Monitoring IaaS CC platformy postavenej nad OpenStack - experiment

OBSAH

Zoznam obrázkov

ÚVOD

Svet informačných technológií sa neustále vyvíja. V posledných rokoch v tejto sfére dominuje pojem cloud computig (CC). Hlavným prvkom, ktorý umožnil vznik a následný rozvoj CC bola virtualizácia. CC umožnil firmám aj jednotlivcom prechod z lokálnej infraštruktúry na virtualizované zdroje nasadené v cloude. Tento prístup je výhodný hlavne z pohľadu cenovej efektivity, kde sa uplatňuje ekonomika z rozsahu. CC sa ukázal prospešný aj vo vzdelávacej sfére, keďže otvoril nové možnosti pre študentov, či už pri vyvíjaní softvéru alebo pri realizovaní rôznych projektov, ktoré majú rozdielne nároky na hardvérové požiadavky.

V tejto práci preskúmame problematiku monitorovania CC prostredia. Pokúsime sa definovať motiváciu pre takýto monitoring. Budeme sa venovať aj problematike viacerých používateľov platformy a spôsobu ako im reprezentovať zozbierané dáta. Preskúmame metriky, ktoré je vhodné v takomto prostredí monitorovať. Následne sa pokúsime nájsť konkrétne riešenie, ktoré by bolo vhodné pre nasadenie takéhoto monitorovacieho systému.

Motiváciou pre prácu na tomto projekte je vznik novej CC platformy na KIS FRI UNIZA, ktorej ambíciou je slúžiť zamestnancom, študentom, prípade aj ďalším subjektom najmä v doméne poskytovania spoľahlivých a výkonných výpočtových zdrojov na požiadanie.

Výsledkom práce by malo byť experimentálne nasadenie vybraného riešenia. Postup nasadenia by mal byť dôkladne zdokumentovaný. Týmto nasadením overíme funkčnosť daného riešenia, ktoré môže byť v neskoršej fáze, napr. v rámci záverečnej práce, nasadené do reálneho CC systému.

1 ANALÝZA

1.1 POPIS TÉMY

V tomto dokumente sa venujeme experimentu s monitoringom cloudu bežiaceho na platforme *OpenStack*, ktorý ponúka najmä IaaS službu.. Na začiatok je potrebné zadefinovať, čo konkrétne je dôležité monitorovať, či už z pohľadu administrátora cloudu, ale aj z pohľadu používateľov. Zaujíma nás hlavne výkon cloudu a priľahlej infraštruktúry, aby sme dokázali zistiť, či nedochádza k ohrozeniu poskytovania služieb. Po určení konkrétnych monitorovacích parametrov preskúmame možnosti monitoringu v platforme *OpenStack*, poprípade možnosti, ktoré ponúkajú tretie strany. Snahou bude vytvoriť návrh riešenia.

1.2 MOTIVÁCIA PRE MONITORING CLOUDU

Monitorovanie cloudu má veľký význam ako pre poskytovateľa, tak aj pre koncového používateľa. Z pohľadu poskytovateľa mu monitoring umožňuje spravovať a kontrolovať ako hardvérovú, tak aj softvérovú infraštruktúru. Sústavné monitorovanie cloudu a jeho SLA (*Service Level Agreement* – Zmluva o úrovni služby) parametrov poskytuje ako poskytovateľovi, tak aj zákazníkovi rôzne druhy informácie, ako napríklad generovaná záťaž, výkonnosť a kvalitu služby (QoS – *Quality of Service*). Na základe týchto informácií dokáže poskytovateľ nasadiť také mechanizmy, ktoré zabránia alebo dokážu napraviť porušenia z pohľadu dohodnutej SLA, či už z pohľadu zákazníka (prekročenie zazmluvnených parametrov), alebo z pohľadu poskytovateľa (nedostatočná výkonnosť služby). Monitorovanie je kritické pre veľké množstvo procesov v cloude. Tieto procesy si rozoberieme nižšie a zhodnotíme rolu monitoringu pri jednotlivých procesoch [1].

1.2.1 Plánovanie zdrojov

Pred masovejším adoptovaním cloudových služieb bolo pre vývojárov služieb a aplikácií náročné odhadnúť aké veľké množstvo výpočtových prostriedkov budú potrebovať pre plynulých chod ich produktu. Poskytovatelia cloudových služieb zvyčajne garantujú istú úroveň QoS. Do tohto by spadala výkonnosť a kvantita výpočtových zdrojov, ktoré sa vopred dohodnú pri vytváraní SLA. Pre vývojárov, ktorý majú svoju službu v cloude, to znamená, že v situácii, kedy by pre chod ich služby potrebovali viac výpočtových zdrojov ako predpokladali, nepotrebujú nakupovať nové zdroje, jednoducho stačí požiadať o navýšenie zdrojov v rámci SLA. Pre poskytovateľa cloudu to však znamená, že musí mať nasadený monitorovací systém tak, aby vedel sledovať a predpovedať vývoj všetkých parametrov, ktoré sa podieľajú na procese garantovania QoS za účelom korektného plánovania rozšírenia cloudovej infraštruktúry [1].

1.2.2 Spravovanie zdrojov

Pre správu komplexného systému, ktorým je aj cloud, je potrebné mať nasadený monitorovací systém, ktorý dokáže presne zachytiť jeho stav v reálnom čase. Oproti správe klasických, čisto hardvérových systémov prináša cloud novú výzvu pre administrátorov. Táto výzva vychádza z jedného zo základných pilierov cloud computingu, virtualizácie. Virtualizácia umožňuje skryť vysokú heterogenitu hardvérových prostriedkov, čo považujeme za benefit, no pramení z toho potreba spravovania ako hardvérových, tak aj virtuálnych zdrojov. Je potrebné si uvedomiť, že virtuálne zdroje môžu kedykoľvek migrovať medzi fyzickými strojmi. Správne nasadený monitorovací systém umožňuje zvládnuť kolísanie dostupných zdrojov a rýchlo sa meniace podmienky v sieti tak, aby boli dodržané všetky zmluvne dohodnuté parametre ponúkaných služieb [1].

1.2.3 Spravovanie SLA

Monitorovací systém je základ a nutnosť pri zisťovaní dodržiavania SLA parametrov. Z pohľadu poskytovateľa monitoring umožňuje viac realisticky naformulovať znenie SLA pre jednotlivých klientov, ako aj poskytnúť lepší fakturačný model na základe vedomostí o výkone infraštruktúry [1].

1.2.4 Fakturácia

Jedna zo základných charakteristík cloud computingu je ponuka tzv. meraných služieb (*measured services*). V praxi to umožňuje zákazníkovi platiť presne za to, čo využil. V prípade typu služby IaaS sa do fakturácie zväčša zahŕňa počet virtuálnych strojov, pričom sa berú do úvahy aj ich špecifikácie ako množstvo vCPUs, veľkosť pamäte a pod. Pre tento proces je monitorovanie kľúčové, keďže poskytovateľ vďaka nemu získa potrebné informácie pre zostavenie faktúry a zákazník si dokáže overiť správnosť naúčtovaných poplatkov, a sledovať spotrebu [1].

1.2.5 Riešenie problémov

Komplexná infraštruktúra cloudu predstavuje výzvu pri snahe lokalizovať a vyriešiť prípadný vzniknutý problém. Ak by v infraštruktúre nebol nasadený monitorovací systém, schopný identifikovať problém v konkrétnej sekcii (napr. na základe logovacích záznamov), museli by sa manuálne prehľadávať početné komponenty (napr. sieť, host), pričom niektoré z týchto komponentov sú ešte tvorené niekoľkými rôznymi vrstvami (napr. skutočný a virtuálny hardvér). Správne nasadený monitorovací systém nám umožní vyhnúť sa tomuto zdĺhavému procesu a rýchlo detegovať a reagovať na vzniknuté chyby, čo v konečnom dôsledku znamená kratšie výpadky a lepšiu službu [1].

1.3 ZÁKLADNÉ KONCEPCIE PRE MONITORING CLOUDU

1.3.1 Cloudové vrstvy

Na základe práce *Cloud Security Alliance* dokážeme cloud rozčleniť do siedmych základných vrstiev. Tieto vrstvy sú vymenované a popísané nižšie.

- **Vybavenie:** táto vrstva sa skladá z celej fyzickej infraštruktúry zahŕňajúc hardvérové stroje hosťujúce výpočtové prostriedky, ako aj sieťové zariadenia.
- **Sieť:** do tejto vrstvy zahŕňame linky a cesty ako v cloude, tak aj medzi cloudom a koncovým používateľom.
- **Hardvér:** jednotlivé hardvérové komponenty výpočtových aj sieťových zariadení zaraďujeme do tejto vrstvy. Môžu to byť napríklad procesory, pamäte, matičné dosky a pod.
- **Operačný systém (OS):** do tejto vrstvy zaraďujeme softvérové komponenty OS hostiteľa (OS bežiace na fyzických strojoch) aj OS používateľa (OS bežiace na virtuálnych strojoch).
- **Middleware:** táto vrstva je relevantná iba z pohľadu niektorých typov služieb, presnejšie SaaS a PaaS. Zaraďujeme sem softvérovú vrstvu medzi OS a užívateľskou aplikáciou.
- **Aplikácia:** sem zaraďujeme aplikáciu, ktorá bola spustená užívateľom cloudového systému.
- **Používateľ:** do tejto poslednej vrstvy zaraďujeme finálneho používateľa cloudového systému a ním spustené aplikácie bežiace mimo cloudu (napríklad používateľom spustený webový prehliadač na vlastnom zariadení).

Z pohľadu monitoringu môžeme tieto vrstvy, spomenuté vyššie, chápať ako miesta, kam by sme mali nainštalovať jednotlivé monitorovacie sondy [1].

1.3.2 Úrovne monitorovania

V cloudovom prostredí môže byť nasadené vysoko aj nízko úrovňové monitorovanie. V prípade dobre fungujúceho monitorovacieho systému je vhodné, aby boli nasadené obidve úrovne monitorovania. Vysoko úrovňové monitorovanie sa zaoberá stavom samotnej virtuálnej platformy. Na druhú stranu nízko úrovňové monitorovanie sa viac zaoberá stavom fyzickej infraštruktúry. Pri typoch služby PaaS a SaaS je toto nízkoúrovňové monitorovanie čisto v záujme poskytovateľa a konečný používateľ nemá prístup k týmto dátam, zatiaľ čo pri IaaS sú obidve úrovne monitorovania v záujme ako zákazníka, tak aj poskytovateľa. Z pohľadu IaaS je dôležitejšie práve nízko úrovňové monitorovanie, ktoré zbiera informácie z rôznych cloudových vrstiev. Príklady pre jednotlivé vrstvy sú uvedené nižšie:

- **Hardvérová vrstva:** rôznorodé informácie o CPU (teplota, frekvencia, zaťažnie), pamäte (dostupná, využívaná, teplota), teploty na ostatných komponentoch, napätie, záťaž ostatných komponentov.
- Vrstva operačného systému a middleware: chyby a softvérové zraniteľnosti.
- **Sieťová vrstva:** bezpečnosť infraštruktúry pomocou firewall-u, *Intrusion Detection System* (IDS) a *Intrusion Prevetion System* (IDS).
- **Úroveň vybavenia:** sem by sme mohli zahrnúť fyzické monitorovanie dátových centier za použitia videokamier [1].

1.3.3 Monitorovacie testy a zbierané metriky

Monitorovacie testy vieme rozdeliť do dvoch hlavných kategórií, ktoré nazveme sieťová a výpočtová kategória. Teraz si popíšeme aké metriky je vhodné zbierať v jednotlivých kategóriách.

Pre výpočtovú kategóriu je vhodné nasadiť testy pre nasledovné metriky.

- **Serverová priepustnosť** definovaná ako počet požiadaviek za sekundu
- Rýchlosť procesora
- Čas spracovania jednej požiadavky procesorom
- Využitie procesora
- **Počet I/O operácií** definovaných v počte za sekundu
- Priepustnosť pamäte
- Doba odozvy
- Čas potrebný na spustenie VM
- Trvanie spustenia VM

Pre sieťovú kategóriu je vhodné nasadiť testy na monitorovanie týchto metrík.

- Round trip time (RTT)
- Kolísanie oneskorenia (jitter)
- Priepustnosť
- Stratovosť
- Dostupná šírka pásma

Všetky tieto metriky vieme vyhodnocovať pomocou klasických štatistických indikátorov ako je priemer, medián atď [1].

Práca [2] sa na metriky pozerá podobným spôsobom. Taktiež rozdeľuje monitorovacie metriky do štyroch kategórií na základe vrstiev cloud modelu, ktoré sme spomínali v 1.3.1. Jednotlivé kategórie ako aj dané metriky, ktoré k nim patria popisujeme nižšie.

a) **Založené na pracovnej záťaži (*Workload-based*)** – Táto kategória sa týka najmä cloudovej služby SaaS, keďže jednotlivé metriky, ktoré sa v tomto prípade nazývajú aj ako kľúčové výkonové indexy (*Key Performance Index* - KPI) a zbierajú sa na aplikačnej vrstve. Do tejto kategórie vieme zaradiť napríklad dobu odozvy medzi službami, dobu odozvy v medziprocesnej komunikácii, počet spustených vlákien a pod. Tieto metriky môžeme následne zlúčiť do jednej abstraktnej metriky, na základe ktorej vieme priradiť jednotlivým virtuálny strojom (VM) váhu zaťaženia. Toto vieme neskôr pretaviť na QoS výkonnostnú metriku pre danú VM [2]–[4].
b) **Založené na výpočtových prostriedkoch (*Compute-based*)** - Táto kategória je dôležitá hlavne z pohľadu IaaS a PaaS typov služieb. Radíme sem metriky ako doba prevádzkyschopnosti systému, priepustnosť disku, využitie CPU a RAM, alokácia pamäte a doba vytvorenia VM [3], [5].
c) **Založené na sieťových prostriedkoch (*Network-based*) –** Do tejto kategórie môžeme zaradiť metriky, ktoré sú úzko zviazané s celkovým statusom sieťovej konektivity. Konkrétne sa jedná o počty paketov, priepustnosť liniek a stav sieťových rozhraní [2], [3], [5].
d) **Založené na udalostiach (*Events*) –** Medzi udalosti, ktoré je prospešné sledovať z pohľadu cloud monitoringu patria najmä udalosti generované jednotlivými komponentami cloudu na IaaS a PaaS vrstve. Samotné informácie od komponentov nie sú dostatočné, preto sa ich odporúča využívať v korelácii s dátami nazbieranými z metrík spomenutých vyššie. Pri takomto využití vieme rýchlejšie a presnejšie identifikovať vzniknuté chyby a nedostatky, ktoré sa môžu počas behu systému vyskytnúť [2], [5], [6].

Keďže my sa venujeme najmä monitoringu cloudovej platformy poskytujúcej primárne službu IaaS, tak je vhodné si zhrnúť čo nás z tohto pohľadu zaujíma. Zdroje poskytované IaaS službou sú primárne vo forme virtuálnych strojov. Virtuálny stroj potrebuje pre svoje fungovanie procesné výpočtové zdroje a úložisko. Z toho vyplýva, že v prípade cloudovej platformy poskytujúcej IaaS je vhodné nasadiť monitorovacie riešenie, ktorého cieľom je monitorovanie základných metrík ako napríklad využitie procesora a zaťaženie siete.

1.3.4 Pohľady monitoringu

Monitorovací systém dokáže poskytnúť informácie o rôznych aspektoch výkonu, správania a celkového stavu systému. Tu je však dôležité si uvedomiť, že to ako sa s týmito informáciami bude ďalej narábať nezávisí len na tom, z akej cloudovej vrstvy sme informácie získali, ale aj na tom, kto tieto informácie ďalej získa a za akým účelom. V prípade IaaS cloudovej služby sa zákazníkom reportuje stav ich virtuálneho stroja, čo zahŕňa systémovú záťaž VM, využitie pamäte a celkový výkon. Poskytovateľ takejto služby však musí monitorovať každú takúto VM v jeho infraštruktúre, aby si bol istý že neustále poskytuje službu v rámci dohodnutých SLA parametrov. Taktiež musí monitorovať fyzickú infraštruktúru, aby dokázal kontrolovať celkovú systémovú záťaž, prideľovanie zdrojov novým VM prípadne migráciu už vytvorených. Z toho vyplýva, že pre rôzne entity podieľajúce sa na celkovej službe prináležia rôzne nazbierané dáta, keďže majú rôzny pohľad na danú službu. Vo všeobecnosti poznáme dva pohľady, ktoré uvádzame nižšie [7].

- **Pohľad klienta –** Z tohto pohľadu je cloud vnímaný ako abstraktná entita, schopná poskytnúť požadované výpočtové služby. Informácie poskytované v rámci tohto pohľadu by mali umožniť klientovi pochopiť charakteristiky danej služby, prípade optimalizovať jej využívanie. To, aké konkrétne informácie (ktoré metriky) budú poskytované závisí na znení SLA [7].

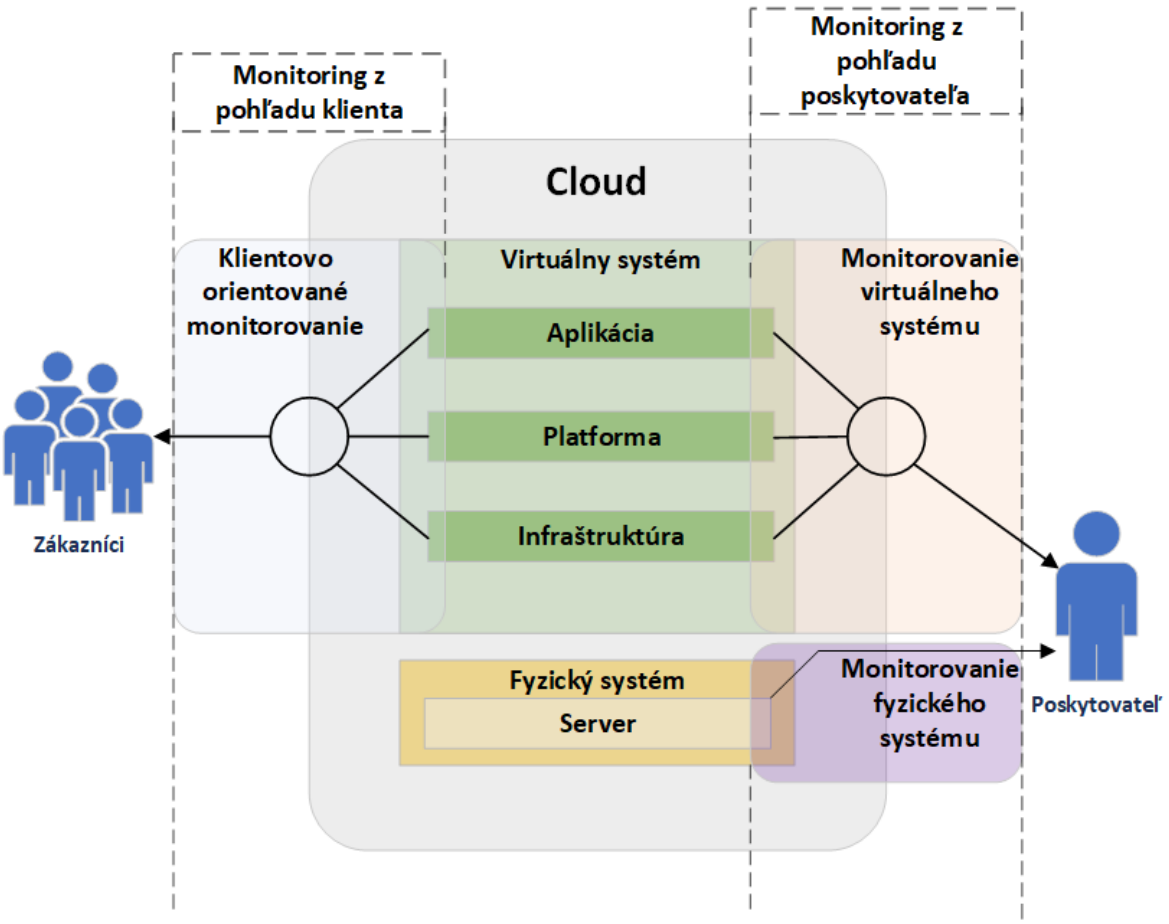
- **Pohľad poskytovateľa –** Z tohto pohľadu je cloud komplexná distribuovaná infraštruktúra s početnými hardvérovými aj softvérovými elementami, ktoré sú spolu prepojené takým spôsobom, aby dokázali poskytovať určité sety služieb. Informácie poskytované v rámci tohto pohľadu by mali prevádzkovateľovi získať prehľad o fungovaní jednotlivých cloudových elementov, ich stave, výkone a pod. Poskytovateľ by mal z týchto informácií vedieť odkontrolovať garanciu SLA parametrov alebo môžu byť použité na optimalizáciu riadenia systému a využívania zdrojov [7].

Pre lepšie pochopenie vyššie spomenutých pohľadov si uvedieme príklad z praxe. Predstavme si situáciu, kde figurujeme ako poskytovateľ cloudovej služby IaaS, pričom prevádzkujeme hybridný cloud. Privátnu časť hybridného cloudu máme postavenú nad platformou *OpenStack*, pričom je spustená na nami vlastnených hardvérových prostriedkoch. Ako verejnú časť hybridného cloudu využívame Amazon EC2. V prípade monitorovacieho systému sa dokážeme na privátnu časť pozerať z pohľadu poskytovateľa, keďže vieme zbierať informácie z celej infraštruktúry. Na verejnú časť (Amazon EC2) sa už nevieme pozerať z rovnakého pohľadu, keďže tu figurujeme ako zákazník spoločnosti Amazon, ktorá nám poskytuje len informácie o našich VM. Preto sa na verejnú časť pozeráme z pohľadu zákazníka [7].

V prípade ak spojíme monitoring jednotlivých vrstiev cloudu so zadefinovanými pohľadmi, vieme monitoring cloudu rozdeliť na tri základné typy. Tieto typy definujeme ako

1. Klientovo orientované monitorovanie
2. Monitorovanie virtuálneho systému
3. Monitorovanie fyzického systému

Pre pochopenie je lepšie si to zobraziť graficky, preto uvádzame Obrázok 1.1.



Obrázok 1.1 Spojenie pohľadu a úrovne monitoringu cloudu

1.4 CENTRALIZOVANÝ ZBER LOGOVACÍCH ZÁZNAMOV

Táto práca sa nebude dopodrobna zaoberať problematikou centralizovaného zberu logovacích záznamov. Namiesto toho budú v tejto podkapitole zosumarizované najdôležitejšie časti z diplomových prác [8][9], ktoré považujeme za dôležité v kontexte návrhu monitoringu cloudového riešenia.

Jedným z najpoužívanejších spôsobov zaznamenávania udalostí na rôznych zariadeniach je za použitia riešenia s menom Syslog. Ten je používaný najmä na UNIXových systémoch a funguje na princípe client-server modelu. Na unixových zariadeniach sa logovacie záznamy nachádzajú zvyčajne v adresári /var/log/.

Pri zbere väčšieho množstva logovacích záznamov je potrebné riešiť aj ich systém archivácie. Z nášho pohľadu budeme riešiť optimálne riešenia archivácie a zberu v operačnom systéme Linux, kde sa ako najlepšie riešenie javí použitie služby logrotate, ktorej zrozumiteľná dokumentácia sa nachádza v Linux manuály. Ďalšou časťou, ktorá by pre nás mohla byť v ďalšom riešení zaujímavá je schopnosť vizualizácie zozbieraných dát (logov), čo by sme sa snažili dosiahnuť pomocou riešenia s otvorenou licenciou. V prípade, ak by sme potrebovali aj v našom riešení realizovať istý centralizovaný zber logov, nám práve práce [1] a [2] ponúkajú 2 rôzne riešenia, ktorými by sme to vedeli zrealizovať. Keďže sa tomu vyššie spomenuté práce venujú podrobne, tu len pomenujeme riešenia, ktoré boli využité. Jedno z riešení je vytvorenie centrálneho zberu

logov pomocou LXC kontajnery a následne ich vizualizácia pomocou nástroja Graylog. Ako ďalšie z riešení bola predstavená kolekcia troch produktov s otvorenou licenciou a to Elasticsearch, Logstash a Kibana. Toto druhé riešenie je z nášho pohľadu zaujímavejšie, keďže produkt Kibana (vizualizačný nástroj) často figuruje aj v spojení *OpenStack* monitoringu za pomoci riešenia Prometheus, ktorému by sme sa chceli neskôr v tejto práci venovať.

1.5 SYSTÉM MONITORINGU *OPENSTACK* CLOUDOVEJ PLATFORMY POSTAVENÉ NA RIEŠENÍ PROMETHEUS

Monitoring je dôležitou súčasťou cloudových platforiem, vďaka ktorému vieme efektívne analyzovať sieť, spravovať systém, predpovedať rôzne udalosti, detegovať chyby a vykonávať zotavujúce operácie [10].

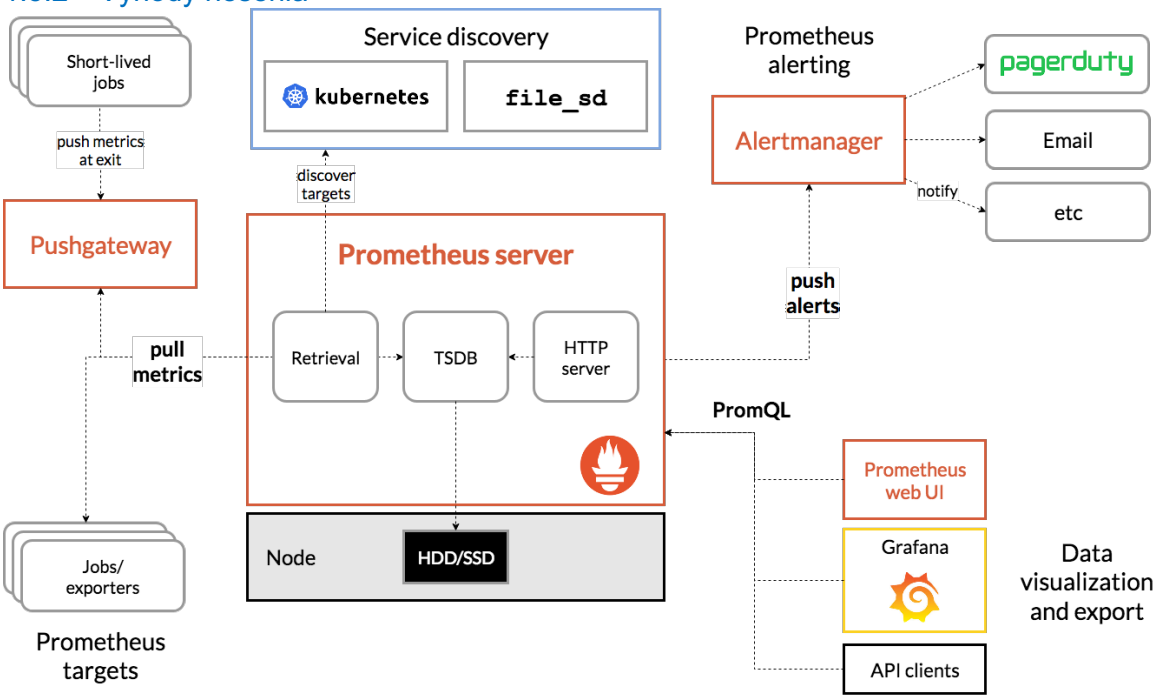
1.5.1 Prometheus

Samotné riešenie na monitorovanie s názvom Prometheus sa skldá z niekoľkých modulov. Medzi tieto moduly patrí:

- Prometheus server
  - Získava požadované dáta, ktoré následne ukladá na definované miesto
- Exporters
  - Reportuje dáta Prometheus serveru. Tento modul je spustený popri aplikácii, z ktorej chceme získať požadované metriky.
- Pushgateway
  - Plní funkciu kvázi buffera, kde sa uchovávajú namerané dáta pred tým, ako si ich Prometheus server vyžiada.
- PromQL
  - Prometheus Query Language – získavanie dát z databázy.
- Alertmanager
  - Slúži na informovanie a upozorňovanie administrátora.
- WebUI
  - Slúži najmä prezentáciu nameraných dát.

Všetky tieto moduly sú navzájom prepojené a vytvárajú jeden funkčný celok [10]. Architektúru riešenia Prometheus reprezentuje **Chyba! Nenašiel sa žiaden zdroj odkazov.**.

1.5.2 Výhody riešenia



Obrázok 1.2 Architektúra Prometheus

Riešenie Prometheus sa prezentuje niekoľkými výhodami. Tieto výhody sú bližšie popísané nižšie.

- Poskytnutie kvalitných dát
  - Prometheus uchováva dáta ako časové rady, ktoré sú presne definované názvom.
- Elastické získavanie dát z databázy
  - Vďaka vlastnému *query* jazyku, dokáže používateľ získať z databázy presne tie dáta, ktoré potrebuje.
- Vizualizácia
  - Poskytuje niekoľko vzorov, ktorými dokáže vizualizovať namerané dáta.
- Efektívne ukladanie
  - Namerané dáta sa ukladajú v špecifickom a efektívnom formáte.
- Jednoduchá operácia
  - Každý nasadený server je nezávislý. Všetky binárne súbory sú staticky pripojené a jednoduché na nasadenie.
- Presné upozornenia
  - Vďaka architektúre riešenia vie Prometheus v krátkom čase informovať administrátora o vzniknutých situáciách na základe vopred nakonfigurovaných upozorňovacích pravidiel [3] [4].

1.5.3 Prometheus a SNMP

V prípade monitoringu zariadení, ktorý je postavený na protokole Simple Network Management Protocol (SNMP) je taktiež možné využiť riešenie Prometheus. Základným predpokladom je mať funkčnú inštaláciu Promethea, prípadne aj vizualizačného nástroja Grafana, pričom by mal byť Prometheus nakonfigurovaný ako zdroj dát pre Grafanu.

Ďalšia nutnosť je prítomnosť SNMP agenta na zariadeniach, ktoré chceme monitorovať takýmto spôsobom.

Aby dokázal Prometheus zbierať dáta cez SNMP je nutné mať nainštalovaný a správne nakonfigurovaný snmp_exporter na zariadení (napríklad na zariadení, kde beží samotný Prometheus). Tento exporter nám umožní zbierať spomenuté dáta cez SNMP. Pre konfiguráciu exportera je možné manuálne upravovať konfiguračný súbor, ktorý je vo formáte yml. Manuálna úprava súboru sa však neodporúča, namiesto toho je vhodné využiť snmp_exporter configuration generator, ktorý je k dispozícii na stiahnutie priamo so samotným exporterom z lokality github, viac informácií je dostupných na: https://github.com/prometheus/snmp_exporter/tree/main/generator [12].

1.5.4 Vizualizácia

Na analýzu a vizualizáciu dát využíva riešenie Prometheus nástroj s názvom Grafana, ktorá sa prezentuje ako medzi platformový nástroj určený na analýzu a vizualizáciu nameraných dát. Má niekoľko základných charakteristík, medzi ktoré patria:

- Zobrazovanie dát
  - Grafana poskytuje veľké množstvo spôsobov, ktorými je možné dáta vizualizovať. Oficiálna knižnica obsahuje veľké množstvo rozšírení z ktorých je možné si vybrať (tepelné mapy, rôzne typy grafov ako čiarový, koláčový atď.)
- Zdroje dát
  - Riešenie dokáže čerpať a vizualizovať dáta z veľkého množstva zdrojov (databáz). Medzi tieto zdroje patrí napríklad Graphite, OpenTSDB, Prometheus, Elasticsearch, CloudWatch, KairosDB, atď.
- Zasielanie upozornení
  - Vizuálne zobrazuje nastavené hranice pre zasielanie upozornení, pre najdôležitejšie indikátory.
- Zmiešané grafy
  - Dokáže zobraziť niekoľko rôznych zozbieraných metrík, z rôznych datasetov, v jednom grafe [10].

1.5.5 Možný spôsob nasadenia

V rámci literatúry, ktorú sme študovali sme narazili na konkrétny spôsob nasadenia riešenia Prometheus v *OpenStack* platforme. V tomto riešení boli v cloude nasadené 2 výpočtové uzly, jeden úložiskový uzol a jeden kontrolný uzol. Prometheus bol spustený na externom stroji, ktorý spravoval monitoring. Na každom uzle bol nasadený takzvaný node exporter, ktorý zbieral a odosielal najmä systémové informácie o každom uzle. Na centrálnom uzli bol navyše nasadený *OpenStack* exporter. Ten zodpovedal za zber základných dát z *OpenStack* platformy. Zbierané parametre sú zobrazené v tabuľke nižšie [10].

Node exporter		***OpenStack* exporter**	
Názov parametra	Bližší popis	Názov parametra	Bližší popis
ARP	Zbiera ARP štatistiky	(*Meminfo*) Informácie o pamäti	Zbiera dáta o operačnej pamäti
CPU	Zbiera štatistiky o využití procesora	VCPUs	Zbiera dáta o virtuálnych procesoroch
(*Diskstats*) Štatistiky disku	Zbiera informácie o I/O	RAM	Zbiera štatistiky operačnej pamäte
(*Filesystem*) Súborový systém	Monitoruje využitie disku	Disk	Zbiera dáta ako sú voľné miesto na disku a jeho využitie
(*Loadavg*) Záťaž systému	Zbiera informácie o záťaži systému	Inštancie	Zbiera štatistiky o virtuálnych strojoch, napríklad ich počet
(*Meminfo*) Informácie o pamäti	Zbiera dáta o operačnej pamäti	Swift	Zbiera dáta o module Swift

Riešenie Prometheus sa pokúsime nasadiť v našom experimentálnom prostredí, aby sme si overili jeho funkčnosť a funkcionalitu. Nasadeniu sa bude venovať ďalšia kapitola.

2 VÝSLEDKY PRÁCE - EXPERIMENTÁLNE NASADENIE PROMETHEUS + GRAFANA

Naša testovacia infraštruktúra sa nachádza v katedrovej sieti za firewallom. Pre vstupný smerovač nám bola pridelená verejná IP adresa 158.193.153.30. V našej vnútornej sieti za vstupným smerovačom sme sa rozhodli využívať privátny IPv4 rozsah 192.168.0.0/24. Riešenie bude pozostávať z implementácie vstupného smerovača Prometheus servera. Z počiatku bude Prometheus monitorovať len samého seba (vzorová konfigurácia), pre monitorovanie nášho cloudu budeme musieť vykonať prislúchajúcu konfiguráciu. Našou snahou bude monitorovať poskytnutý nasadený *OpenStack* na KIS.

2.1 INŠTALÁCIA A NASADENIE VSTUPNÉHO SMEROVAČA

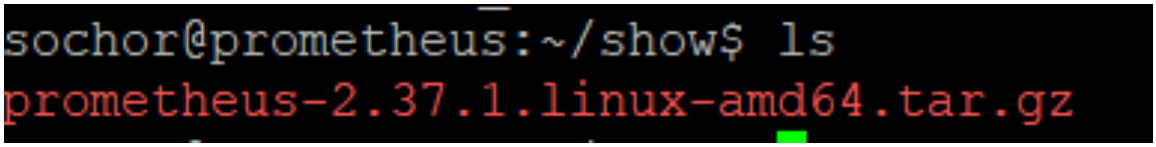
Ako smerovač sme sa rozhodli nasadiť serverovú distribúciu Ubuntu 20.04 LTS, najmä pre jednoduchosť ovládania, ktoré vyplýva z predchádzajúcich skúseností s prácou na tomto OS. Smerovacie politiky riešime pomocou nftables.

2.1.1 Inštalácia Prometheus

Monitorovacie riešenie sa skladá z troch samostatných nástrojov (Prometheus, Grafana a *OpenStack* exporter for Prometheus). Všetky tieto nástroje budeme pre experimentálne nasadenie inštalovať na jednu VM. Je však dôležité podotknúť, že inštalácia na jeden stroj nie je podmienkou funkčnosti. Každý nástroj môže byť inštalovaný na samostatnom systéme za predpokladu, že máme medzi jednotlivými zariadeniami sieťovú konektivitu na špecifikovaných portoch. V našom prípade, rovnako ako pri smerovači, využívame OS Ubuntu 20.04 LTS. Je dôležité mať tento systém aktualizovaný pre zaručenie správnej funkcionality. Ako prvé budeme nasadzovať Prometheus podľa nižšie uvedeného postupu.

Je veľa dostupných spôsobov ako nainštalovať Prometheus, ako napríklad z prekompilovaných binárnych súborov, priamo zo zdrojového kódu alebo pomocou Docker obrazu. My sme pri našom nasadení využili prekompilovaný binárny súbor. Tieto súbory, nie len pre Prometheus, ale aj pre jeho ďalšie priame nadstavby, sú dostupné na webovej lokalite https://prometheus.io/download/. Inštalácia je možná pre veľké množstvo operačných systémov (windows, linux, darwin), ako aj rôzne architektúry procesora. My sme samozrejme volili operačný systém linux s architektúrou amd64 a LTS verziu 2.37.1. Pre stiahnutie binárneho súboru využívame nástroj wget. Tento nástroj by mal byť nainštalovaný spolu s OS Ubuntu, no v inom prípade je možné ho doinštalovať pomocou `#sudo apt-get install wget.` Pre stiahnutie potrebujeme ešte URL odkaz na binárny súbor. V našom prípade sťahujeme do domovského priečinku prihláseného používateľa pomocou: `#wget https://github.com/prometheus/prometheus/releases/download/v2.37.1/prometheus-2.37.1.linux-amd64.tar.gz`
Proces sťahovania budeme vidieť podobne ako na nasledovnom obrázku.

Teraz si môžeme jednoducho overiť, či sa súbor stiahol pomocou príkazu `#ls`. Vo výpise by sme mali vidieť náš stiahnutý súbor podobne ako na obrázku nižšie.



Obrázok 2.2 Zložka show s Prometheus.tar

Po stiahnutí potrebujeme extrahovať archív, aby sme mohli pristúpiť k samotnej inštalácii. Toto vykonáme pomocou nástroja tar, konkrétne za použitia prepínača -xvzf. Význam prepínača je nasledovný: X pre extrahovanie archívu, V pre zrozumiteľný výpis, Z keďže na konci súboru máme .gz a f pre vytvorenie zložky s rovnakým názvom ako má archív. V našom prípade vyzeral príkaz pre extrahovanie a jeho následný výpis aj s výpisom obsahu aktuálneho adresára nasledovne.

Obrázok 2.1 Proces sťahovania Prometheus

```
sochor@prometheus:~/show$ tar -xvzf prometheus-2.37.1.linux-amd64.tar.gz
prometheus-2.37.1.linux-amd64/
prometheus-2.37.1.linux-amd64/consoles/
prometheus-2.37.1.linux-amd64/consoles/index.html.example
prometheus-2.37.1.linux-amd64/consoles/node-cpu.html
prometheus-2.37.1.linux-amd64/consoles/node-disk.html
prometheus-2.37.1.linux-amd64/consoles/node-overview.html
prometheus-2.37.1.linux-amd64/consoles/node.html
prometheus-2.37.1.linux-amd64/consoles/prometheus-overview.html
prometheus-2.37.1.linux-amd64/consoles/prometheus.html
prometheus-2.37.1.linux-amd64/console_libraries/
prometheus-2.37.1.linux-amd64/console_libraries/menu.lib
prometheus-2.37.1.linux-amd64/console_libraries/prom.lib
prometheus-2.37.1.linux-amd64/prometheus.yml
prometheus-2.37.1.linux-amd64/LICENSE
prometheus-2.37.1.linux-amd64/NOTICE
prometheus-2.37.1.linux-amd64/prometheus
prometheus-2.37.1.linux-amd64/promtool
sochor@prometheus:~/show$ ls
prometheus-2.37.1.linux-amd64  prometheus-2.37.1.linux-amd64.tar.gz
```

Obrázok 2.3 Extrahovanie Prometheus archívu

Ako vidíme na obrázku vyššie pribudla nám v adresári nová zložka, v ktorej sa nachádzajú všetky potrebné súbory pre beh programu. Keď sa presunieme do novo vytvoreného priečinku jeho obsah vyzerá ako je uvedené na nasledujúcom obrázku.

```
sochor@prometheus:~/show/prometheus-2.37.1.linux-amd64$ ls
console_libraries  consoles  LICENSE  NOTICE  prometheus  prometheus.yml  promtool
```

Obrázok 2.4 Obsah priečinku Prometheus

Teraz prejdeme k samotnému spusteniu, pričom máme dva rôzne spôsoby ako sa k tomuto kroku postaviť. Jednoduchší spôsob je spustenie samotného spustiteľného súboru *prometheus* pomocou `#./prometheus`. Toto riešenie je však jednorazové a po každom reštarte systému bude potrebné opätovne spúšťať Prometheus manuálne. Z tohto dôvodu odporúčame vytvoriť Prometheus ako službu, vďaka ktorej ho budeme vedieť ovládať za pomoci nástroja *systemctl,* a taktiež budeme jednoducho vedieť nastaviť jeho spúšťanie pri štarte systému.

2.1.2 Prometheus ako služba

Nastavenie aplikácie ako službu je pomerne jednoduché a priamočiare. Postup je nasledovný:

1. Vytvoríme si nový súbor v adresári **/etc/systemd/system/**. Súbor musí byť pomenovaný v tvare **<*NÁZOV_SÚBORU*>.service.** V našom prípade sme súbor vytvorili pomocou príkazu:
   `#touch /etc/systemd/system/prometheus.service`.
   V našom prípade sme si vytvorenie zatiaľ prázdneho súboru overili tak, ako je uvedené na obrázku nižšie.

```
sochor@prometheus:~$ cd /etc/systemd/system/
sochor@prometheus:/etc/systemd/system$ ls | grep prometheus
prometheus.service
```

Obrázok 2.5 Overenie vytvorenia súboru prometheus.service

2. Teraz potrebujeme vyplniť vytvorený súbor adekvátnou konfiguráciou. Nižšie uvádzame našu vzorovú konfiguráciu. Riadky začínajúce sa znakmi „#“ a „;“ sú v konfiguračných súboroch pre systemd ignorované a preto je ich možné využiť pre komentovanie, podobne ako v našom prípade nižšie.

```
[Unit]
#Popis informačného charakteru
Description=Prometheus Server

#Odkaz na dokumentáciu k aplikácii
Documentation=https://prometheus.io/docs/introduction/overv
iew/

#Špecifikuje, že aplikáciu spustí až po spustení sieťových
#služieb
After=network-online.target

[Service]
#Špecifikuje, že službu spustí pod používateľom root
User=root

#V prípade zlyhania sa pokúsi o reštart služby
Restart=on-failure

#Cesta ku spustiteľnému súboru aplikácie spolu s potrebnými
#prepínačmi (priečinok na ukladanie nazbieraných dát,
#umiestnenie konfiguračného súboru (viď 2.1.4))
ExecStart=/home/sochor/prometheus-2.37.1.linux-
amd64/prometheus --
storage.tsdb.path=/var/lib/prometheus/data/ --
config.file=/home/sochor/prometheus-2.37.1.linux-
amd64/prometheus.yml

[Install]
#Definuje, že služba sa spustí pri spustení systému
WantedBy=multi-user.target
```

3. Ďalej je potrebné reštartovať systemctl démona. Tento krok vykonáme zadaním príkazu
   `#sudo systemctl daemon-reload`
4. Nakoniec zapneme prometheus službu pomocou
   `#sudo systemctl start prometheus.`
   Overenie stavu služby vieme jednoducho vykonať zadaním
   `#sudo systemctl status prometheus.`
5. Aby sa aplikácia spúšťala spolu s OS je potrebné aktivovať túto službu pomocou
   `#sudo systemctl enable prometheus`.

Pri našom pokuse o vytvorenie služby sme narazili na problém, ktorý je aj s riešením bližšie popísaný v kapitole 2.1.8. K obsluhe a konfigurácii tohto riešenia je dostupná detailná dokumentácia na webovej lokalite https://prometheus.io/docs/introduction/overview/.

2.1.3 Inštalácia vizualizačného nástroja Grafana

Po inštalácii riešenia Prometheus odporúčame nasadiť aj vizualizačný nástroj Grafana, ktorý dokáže prispôsobiteľným spôsobom zobrazovať zozbierané dáta či už Prometheom alebo aj z iných databázových riešení. Existuje niekoľko spôsobov ako tento nástroj nainštalovať. V našom prípade sme sa v rámci experimentálneho nasadenia

rozhodli pre inštaláciu .deb súboru, keďže pracujeme na OS Ubuntu. Rôzne metódy inštalovania sú dostupné online na https://grafana.com/docs/grafana/v9.0/setup-grafana/installation/debian/. K dispozícii sú dve rôzne edície: OSS a Enterprise. Obidve je možné využívať bezplatne, pričom vývojármi odporúčaná je enterprise edícia. OSS je pod AGPLv3 licenciou, zatiaľ čo enterprise pod Grafana Labs License. Na webovej stránke sa uvádza, že enterprise edícia obsahuje všetku funkcionalitu OSS edície, avšak je možné ju aktivovať na „full Enterprise feature set", čo obsahuje aj podporu pre enterprise rozšírenia. Keďže uvažujeme nasadenie na akademickej pôjde, vybrali sme OSS edíciu. V čase nasadzovania bola najnovšia stabilná verzia aplikácie 9.2.0, ktorú sme inštalovali.

Návod pre stiahnutie rôznych typov súborov pre konkrétne OS je dostupný na https://grafana.com/grafana/download/9.2.0?edition=oss&platform=linux. V našom prípade sme postupovali nasledovne:

```
#sudo apt-get install -y adduser libfontconfig1
#wget https://dl.grafana.com/oss/release/grafana_9.2.0_amd64.deb
#sudo dpkg -i grafana_9.2.0_amd64.deb
```

Týmto inštalačným spôsobom sa nám automaticky vytvorila grafana ako služba. Pre spustenie aplikácie zadáme nasledovnú sekvenciu príkazov, pričom očakávame status active (running):

```
#sudo systemctl daemon-reload
#sudo systemctl start grafana-server
#sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
     Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled;
vendo>
     Active: active (running) since Thu 2022-10-13 09:15:21 UTC; 6 days
ago
       Docs: http://docs.grafana.org
   Main PID: 886 (grafana-server)
      Tasks: 18 (limit: 19103)
     Memory: 132.4M
```

Podobne ako pri riešení Prometheus, aj tu je vhodné aktivovať službu pre štart spolu s OS.

```
#sudo systemctl enable grafana-server.service
```

2.1.4 Nastavenie a prístup na Prometheus

Po úspešnej inštalácii si môžeme overiť funkčnosť prístupom na grafické rozhranie a nastaviť testovaciu konfiguráciu.

V rámci prvotnej konfigurácie nastavíme, aby Prometheus monitoroval sám seba. Tento krok je uvedený aj v oficiálnej dokumentácii a aj keď nie je veľmi užitočný, poskytne nám základný prehľad, či riešenia správne funguje. Konfiguračný súbor by mal byť pomenovaný prometheus.yml a uložený do dostupného priečinka. My sme tento súbor vytvorili priamo v priečinku so spustiteľným súborom prometheus. Vzorová konfigurácia pre nastavenie monitorovania samého seba vyzerá nasledovne:

```
global:
scrape_interval:     15s # Vyžaduj nové dáta každých 15s – toto je
# prednastavená hodnota

# Tieto značky budú pridelené pre časové záznamy a upozornenia pri
# komunikácii s externými systémami (vzdialené úložiská, Alermanager)
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
```

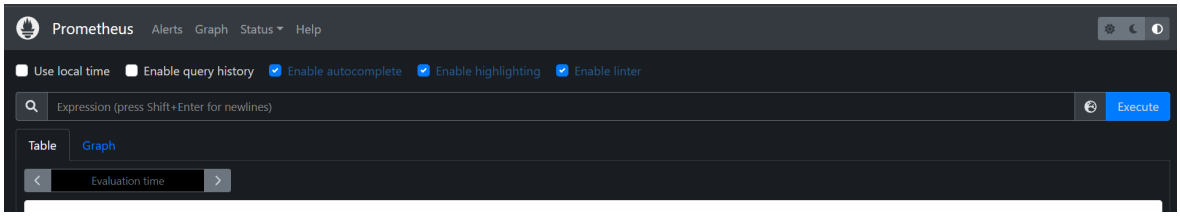
```
    monitor: 'codelab-monitor'

# V tejto časti sa uvádzajú koncové body určené pre monitorovanie.
# V tomto prípade to obsahuje len samotný Prometheus.
scrape_configs:
  # Nastavenie značkovania pre túto prácu
  - job_name: 'prometheus'

    # Prepisuje prednastavený časový interval pre zber dát
    scrape_interval: 5s

    static_configs:
      - targets: ['localhost:9090']
```

Po vytvorení konfiguračného súboru potrebujeme reštartovať aplikáciu, prípadne pomocou prepínača určiť, ktorý konfiguračný súbor sa má využiť. Ak už máme vytvorený Prometheus ako službu, skontrolujeme v súbore /etc/systemd/system/prometheus.service, časť ExecStart prepínač `-config.file`. Tu musí byť uvedená aktuálna cesta ku konfiguračnému súboru. Následne už len vykonáme reštart služby `#sudo systemctl restart prometheus.service`.

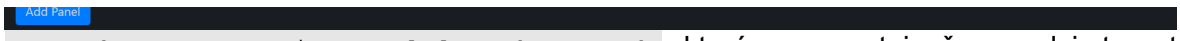
Po vykonaní reštartu je vhodné počkať pár minút kým sa nazbiera isté množstvo dát. Teraz si popíšeme prístup na grafické rozhranie aplikácie.

Ako prvú krok odporúčame overiť status prometheus služby, pre overenie či nevznikol počas behu problém. Očakávame status active (running). Pri momentálnej, továrenskej, konfigurácii máme prístupné grafické rozhranie využitím http na porte 9090. V našom experimentálnom prostredí bolo potrebné nakonfigurovať presmerovanie tohto portu, pre docielenie požadovaného výsledku. Prístup na webové rozhranie http://[IP_ADRESA]:9090, kde sa nám zobrazí vstavaný prehliadač výrazov, ako vidíme na obrázku nižšie.

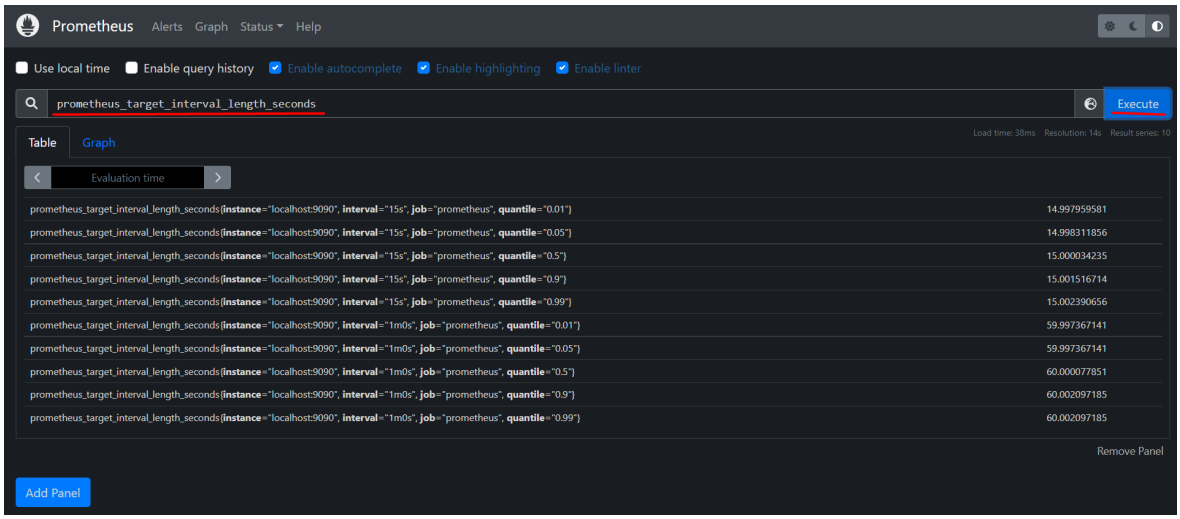
Tento prehliadač slúži na vyhľadávanie nazbieraných dát za použitia výrazového jazyka, ktorý je dostupný online na https://prometheus.io/docs/prometheus/latest/querying/basics/. Vhodné je aj overenie, či Prometheus zbiera dáta sám o sebe a zároveň tu budeme demonštrovať funkcionalitu prehliadača. Jedna z metrík, ktoré o sebe Prometheus zbiera sa nazýva



Obrázok 2.6 Prehliadač výrazov Prometheus

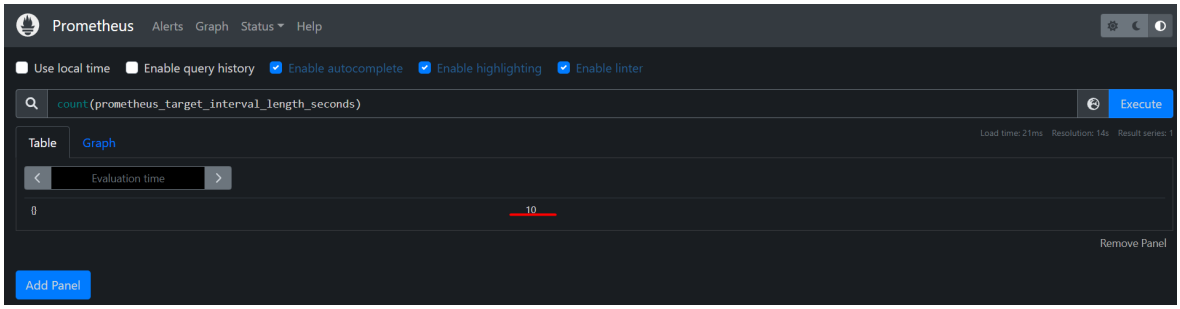


`prometheus_target_interval_length_seconds`, ktorá reprezentuje čas medzi „target scrapes“, čo by sme mohli v slovenčine reprezentovať ako akt získania dát z cieľového hostiteľa. Zadáme názov metriky do vyhradeného miesta a vykonáme dopyt na dáta z databázy, tak ako uvádzame na ďalšom obrázku.



Obrázok 2.7 Prvotné vyhľadanie dát

Výsledkom dopytu je množina časových radov pre metriku so zadaným názvom, ale s rozdielnymi značkami (*labels*). Pre demonštráciu práce s vyhľadávačom si ešte ukážeme ako by sme dostali počet vrátených výsledkov dopytu. Využijeme výraz `count(prometheus_target_interval_length_seconds)`, jeho výstup uvádzame na nasledujúcom obrázku.

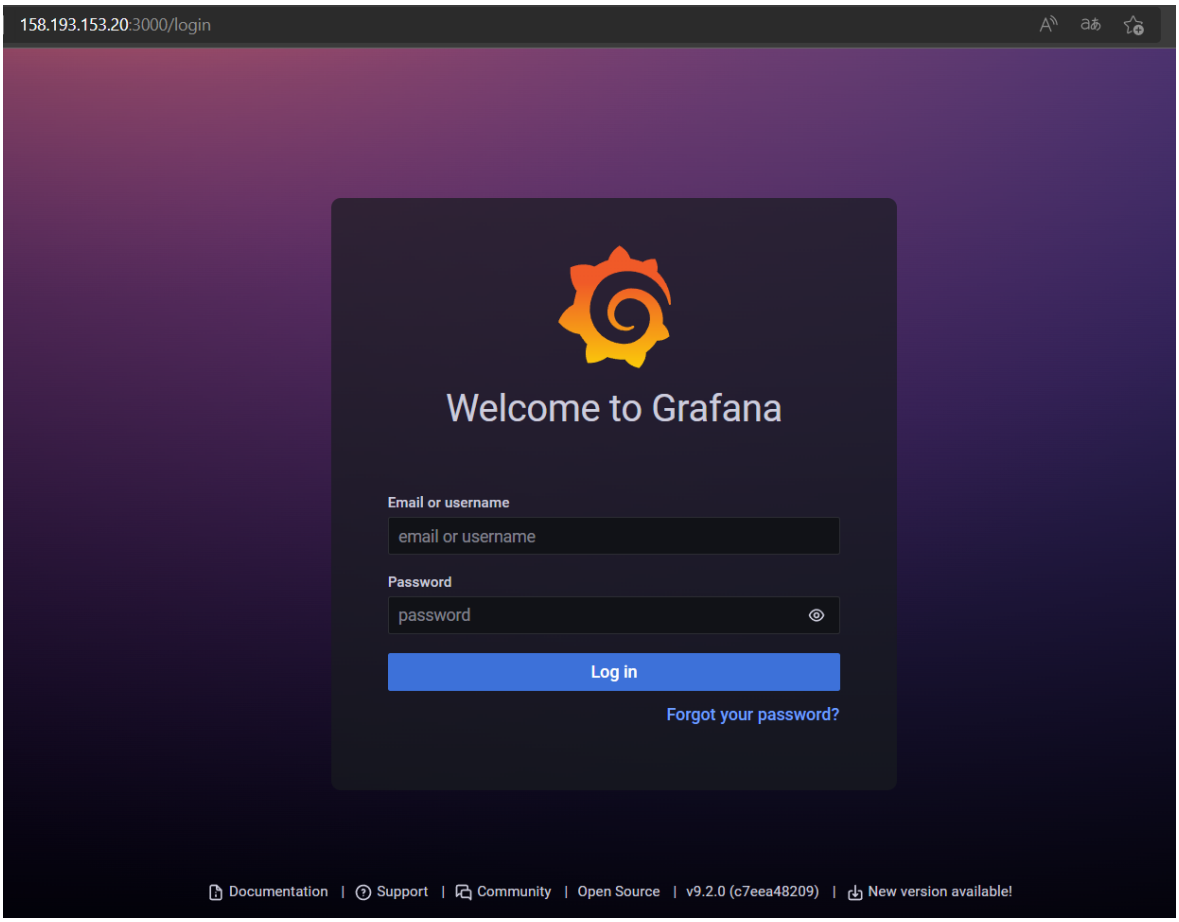


Obrázok 2.8 Vrátenie počtu získaných odpovedí na dopyt

Webové rozhranie ponúka aj istú formu grafickej vizualizácie nazbieraných časových radov, avšak v oficiálnej dokumentácii sa uvádza, že na vizualizáciu je vhodné využiť iné riešenia, ako napríklad Grafana.

2.1.5 Nastavenie a prístup na Grafanu

Na rozdiel od riešenia Prometheus, pri tomto vizualizačnom nástroji nepotrebujeme momentálne vykonávať žiadnu dodatočnú konfiguráciu pre zabezpečenie základnej funkcionality. Továrensky prednastavený port, na ktorom beží webová služba je http/3000. Teraz sa pokúsime dostať do webového rozhrania aplikácie. Využijeme http://[IP_ADRESA]:3000/. Očakávame zobrazenie prihlasovacej obrazovky, ako uvádzame nižšie.

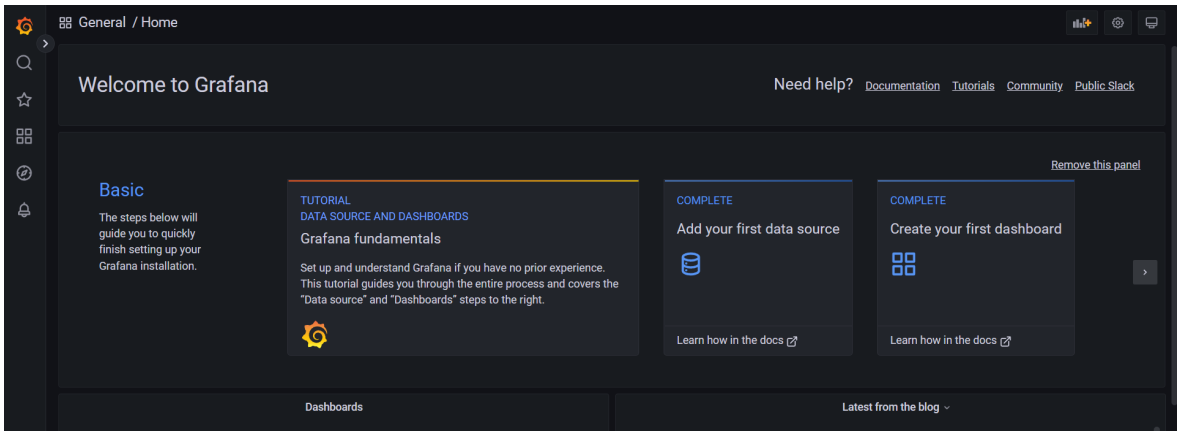


Obrázok 2.9 Prihlasovacia obrazovka Grafana

Továrensky prednastavený administrátorský účet má prihlasovacie údaje:

- Meno: **admin**
- Heslo: **admin**

Po úspešnom overení nás aplikácia vyzve na zmenu prednastaveného hesla, čo je vysoko odporúčaný krok. Následne sa dostávame do domovskej obrazovky. Momentálne nemáme vytvorenú žiadnu obrazovku s vizualizáciou dát, tomu sa budeme venovať až po úspešnom zbere dát z *OpenStack*-u.



Obrázok 2.10 Domovská obrazovka Grafana

2.1.6 Nasadenie *OpenStack* exporter

Veľkou výhodou monitorovacieho riešenia Prometheus je aj veľká ponuka rozšírení, ktoré nám ušetria čas pri nasadení a konfigurácii monitorovania parciálnych častí. Pre *OpenStack* existuje rozšírenie *OpenStack exporter for Prometheus*, dostupné online na [13]. Základnou úlohou tohto rozšírenia je extrahovať údaje o stave vnútorných súčastí *OpenStack* a sprístupniť ich vo formáte kompatibilnom s riešením *Prometheus*. Je vhodné spomenúť, že nástroj funguje za využitia RESP API, pomocou ktorého komunikuje s jednotlivými modulmi. Preto je možné exporter spustiť na ľubovoľnom zariadení za predpokladu, že má toto zariadenie potrebnú konektivitu s monitorovaným cloudom aj so zariadením bežiacom Prometheus.

V rámci nasadenia máme od vývojárov k dispozícii štyri rôzne spôsoby inštalácie:

- Využitím kolla-ansible
- Využitím helm charts
- Za pomoci Docker obrazu z repozitára
- Pomocou snapd

Všetky vyššie spomenuté metódy inštalácie sú popísané aj s ďalšími externými odkazmi (kolla-ansible, helm charts) v git repozitári [13]. V našom experimentálnom nasadení sme exporter inštalovali na rovnaké zariadenie ako Prometheus aj Grafanu. Z pohľadu výkonnosti a škálovateľnosti to nie je optimálne a odporúčame nasadenie týchto nástrojov na samostatné výpočtové entity, avšak v našom prípade nám ide najmä o overenie funkcionality jednotlivých komponentov, ako aj celku v čo najkratšom čase, preto sme volili prístup inštalácie na jedno zariadenie.

Pre jednoduchosť budeme *OpenStack* exporter inštalovať pomocou snapd, pričom si z tohto nástroja taktiež urobíme službu, aby bolo zabezpečené jeho zapínanie spolu so štartom systému.

Inštaláciu začneme nainštalovaním snapu pomocou:

```
#sudo snap install --channel stable golang-openstack-exporter
```

Tento exporter dokáže pracovať v dvoch rôznych módoch, *legacy* (cielený jeden cloud) a *multi cloud* (cielených viacero cloudov). Pre naše nasadenie využívame *legacy* mód, keďže našou snahou je monitorovať jednu cloudovú entitu.

Pre ovládanie aplikácie z konzoly začíname príkazom `#golang-openstack-exporter.openstack-exporter.` Napríklad pre výpis všetkých dostupných prepínačov a možností vieme využiť prepínač –help ako je ukázané nižšie:

```
sochor@prometheus:~$ golang-openstack-exporter.openstack-exporter --help
usage: openstack-exporter [<flags>] [<cloud>]

Flags:
  -h, --help                     Show context-sensitive help (also try
                                 --help-long and --help-man).
      --log.level="info"         Log level: [debug, info, warn, error,
fatal]
```

Z pohľadu riešenia, je pre nás ovládanie aplikácie z konzoly nevhodné, keďže neposkytuje možnosť automatického štartu po reštarte systému. Z tohto dôvodu potrebujeme vytvoriť tento snap ako službu, kde si aj popíšeme dôležité súčasti exportera, ktoré nakonfigurujeme.

Prvým krokom je vytvorenie súboru **[NÁZOV].service** v adresári `/etc/systemd/system`. Následne potrebujeme zistiť umiestnenie spustiteľného súboru pre náš snap. Za týmto účelom sme využili postup, ako je uvedené nižšie.

```
sochor@prometheus:/etc/systemd/system$ which golang-openstack-
exporter.openstack-exporter
/snap/bin/golang-openstack-exporter.openstack-exporter
```

Z výpisu vyššie je zrejmé, že ku spustiteľnému súboru vedie cesta /snap/bin/golang-openstack-exporter.openstack-exporter.

Zo všetkých ponúkaných prepínačov aplikácie budeme zatiaľ využívať len jeden povinný `(--os-client-config)`, a to cestu ku konfiguračnému súboru `clouds.yaml.` Ako prvé si však potrebujeme tento súbor vytvoriť. Experimentálne sme zistili, že najvhodnejšie je vytvárať tento súbor v adresári **/var**. V iných prípadoch (domovský adresár používateľa a adresár /etc) mala aplikácia problém s prístupom k súboru, ktorý bol blokovaný aplikáciou *apparmour*, ktorá je súčasťou balíčka *snapd*. Nami odporúčaný postup uvádzame ako kombináciu všeobecne popísaných krokov s priloženými príkazmi, ktoré sme pri nasadení použili:

1. Vytvorenie súboru clouds.yaml v adresáru /var/[VOLITEĽNÝ_NÁZOV].
   `#mkdir /var/prometheusProject`
   `#touch /var/prometheusProject/clouds.yaml`
2. Vyplniť vytvorený súbor správnou konfiguráciou. Viac informácií o syntaxe je dostupných online na https://docs.openstack.org/python-openstackclient/pike/configuration/index.html. V tomto prípade, tu nebudeme uvádzať našu podobu vyplneného súboru, pretože sa v ňom nachádzajú neverejné informácie. Spomenieme len, že náš cloud sme nazvali default.

Teraz už máme k dispozícii všetky potrebné komponenty, aby sme vytvorili openstack-exporter ako službu. Nižšie uvádzame vzorovú konfiguráciu súboru, v našom prípade node-exporter.service.

```
[Unit]
Description=Openstack exporter
After=network-online.target

[Service]
User=root
```

```
Restart=on-failure

ExecStart=/snap/bin/golang-openstack-exporter.openstack-exporter
--os-client-config=/var/prometheusProject/clouds.yaml default

[Install]
WantedBy=multi-user.target
```

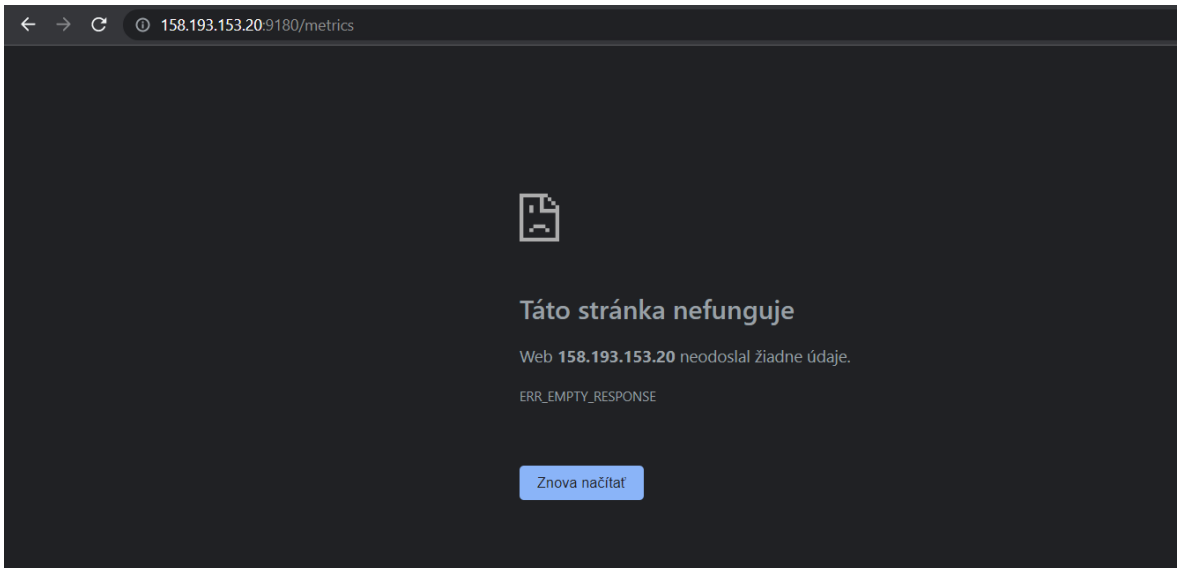
Vykonáme potrebný reload systemctl démona.

```
#sudo systemctl daemon-reload
```

Zapneme našu službu a overíme jej status.

```
#sudo systemctl start node-exporter.service
#sudo systemctl status node-exporter.service
● node-exporter.service - Node exporter
     Loaded: loaded (/etc/systemd/system/node-exporter.service; enabled;
vendor preset: enabled)
     Active: active (running) since Tue 2022-11-01 11:20:33 UTC; 24s ago
   Main PID: 54524 (openstack-expor)
      Tasks: 0 (limit: 19103)
     Memory: 812.0K
     CGroup: /system.slice/node-exporter.service
             ▸ 54524 /snap/golang-openstack-exporter/97/bin/openstack-
exporter --os-client-config=/var/prometheusProject/clouds.yaml default
```

Funkčnosť overíme prístupom na zverejnené metriky. Openstack-exporter zbiera dáta priamo z OpenStacku a sprístupňuje ich na porte 9180 na hostiteľskom zariadení. Vyskúšame teda pristúpiť na toto zariadenie, na konkrétny port a očakávame zobrazenie textovej reprezentácie nazbieraných metrík.



Obrázok 2.11 Prvotný test openstack-exporter

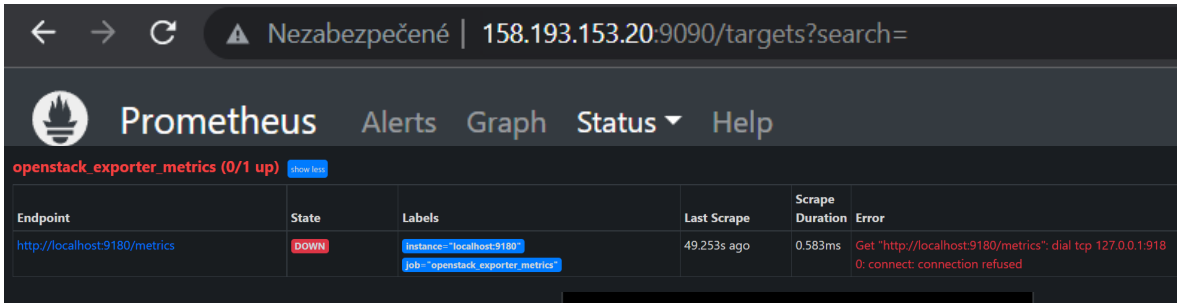
Ako vidíme na obrázku vyššie, nedosiahli sme očakávaný výsledok.

2.1.7 Nastavenie nového zdroja pre Prometheus

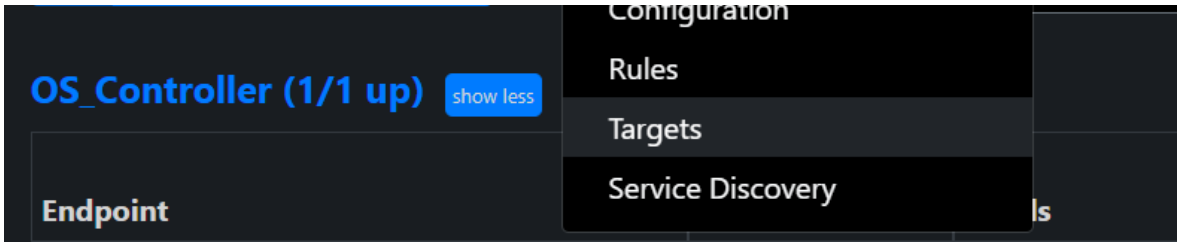
Nakonfigurujeme Prometheus, aby zbieral dáta aj z vyššie spomenutého exportera. Tam si tiež budeme vedieť overiť stav tejto služby. Postup je priamočiary, do konfiguračného súboru promethea (prometheus.yml) pridáme novú prácu (job). V konfiguračnom súbore v časti *scrape_configs:* pridáme nasledovnú konfiguráciu:

```
scrape_configs:
  - job_name: "openstack_exporter_metrics"
    scrape_interval: 60s
    scrape_timeout: 60s
    metrics_path: "/metrics"
    static_configs:
      - targets: ["localhost:9180"]
```

Prometheus si dokáže načítať zmeny v konfigurácii aj počas behu, je to uvedené v oficiálnej dokumentácii aj s postupom, no my sme pre jednoduchosť len reštartovali službu pomocou `#sudo systemctl restart prometheus.service`. Teraz keď pristúpime na grafické rozhranie Prometheus si vieme v časti STATUS->TARGETS (viď Obrázok 2.12) overiť stav novo nakonfigurovaného exportera.



Obrázok 2.13 Prvotný stav openstack-exporter



Obrázok 2.12 Prometheus status->targets

Ako zobrazuje Obrázok 2.13, stav služby je down. Experimentálne sme tento problém vyriešili a postup s popisom uvádzame nižšie.

Aj keď status služby signalizuje stav active (running), experimentálne sme zistili, že po istom počte prístupov na daný port služba prestane bežať. Vieme dokonca povedať, že služba korektne nebežala vôbec. Skúmaním sme zistili, že chyba je v ceste ku spustiteľnému súboru. Necháme si ešte raz vypísať status danej služby.

```
#sudo systemctl status node-exporter.service
● node-exporter.service - Node exporter
     Loaded: loaded (/etc/systemd/system/node-exporter.service; enabled;
vendor preset: enabled)
     Active: active (running) since Tue 2022-11-01 11:20:33 UTC; 24s ago
   Main PID: 54524 (openstack-expor)
      Tasks: 0 (limit: 19103)
     Memory: 812.0K
```

```
     CGroup: /system.slice/node-exporter.service
             ▸ 54524 /snap/golang-openstack-exporter/97/bin/openstack-
exporter --os-client-config=/var/prometheusProject/clouds.yaml default
```

Na konci výpisu vyššie si môžeme všimnúť, že sa spustil proces s PID 54524, ktorý beží spustením súboru na danej ceste s danými prepínačmi. Toto sa však nezhoduje s cestou, ktorú sme v konfigurácii služby uviedli my. Experimentálne sme zistili, že ak do konfigurácie služby uvedieme kompletnú cestu, akú nám poskytol výpis stavu služby, bude exporter fungovať korektne. Preto v našom prípade v súbore /etc/systemd/system/node-exporter.service meníme cestu pri *ExecStart.* V našom prípade zmenený riadok vyzerá nasledovne: `ExecStart=/snap/golang-openstack-exporter/97/bin/openstack-exporter --os-client-config=/var/prometheusProject/clouds.yaml default.`

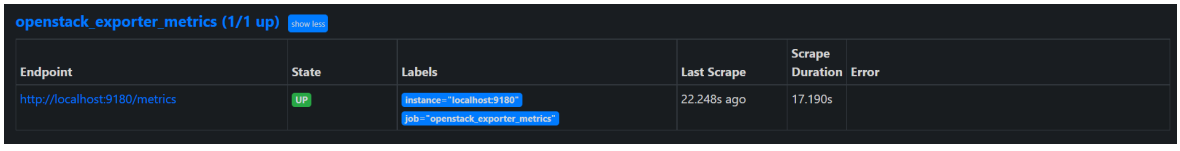
- Opätovne potrebujeme reštartovať systemctl démona. `#sudo systemctl daemon-reload`
- Teraz spustíme službu a vypíšeme si jej stav

```
#sudo systemctl status node-exporter.service
● node-exporter.service - Node exporter
Loaded: loaded (/etc/systemd/system/node-exporter.service; enabled;
vendor preset: enabled)
     Active: active (running) since Tue 2022-11-01 11:31:56 UTC; 33s ago
   Main PID: 55504 (openstack-expor)
      Tasks: 11 (limit: 19103)
     Memory: 20.4M
     CGroup: /system.slice/node-exporter.service
             └─55504 /snap/golang-openstack-exporter/97/bin/openstack-
exporter --os-client-config=/var/prometheusProject/clouds.yaml default
```

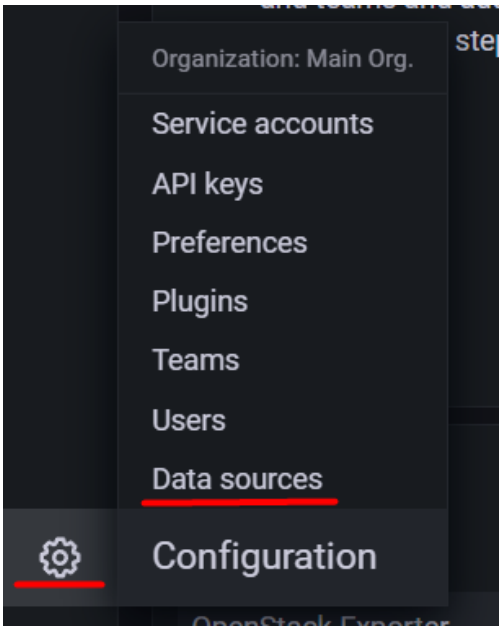
- Overíme stav cieľa v grafickom rozhraní Prometheus. (voliteľne môžeme pristúpiť aj na metriky exportera, na porte 9180)

2.1.8 Spojenie Grafana+Prometheus a nasadenie panelu pre openstack-exporter

Veľkou výhodou riešenia Grafana je dostupnosť obrovského množstva pred vytvorených šablón. Vďaka tomu vieme ušetriť množstvo konfigurácie a nastavovania pri manuálnom vytváraní obrazoviek (dashboardov).



Obrázok 2.14 Funkčný stav openstack-exporter



Obrázok 2.15 Grafana Data sources

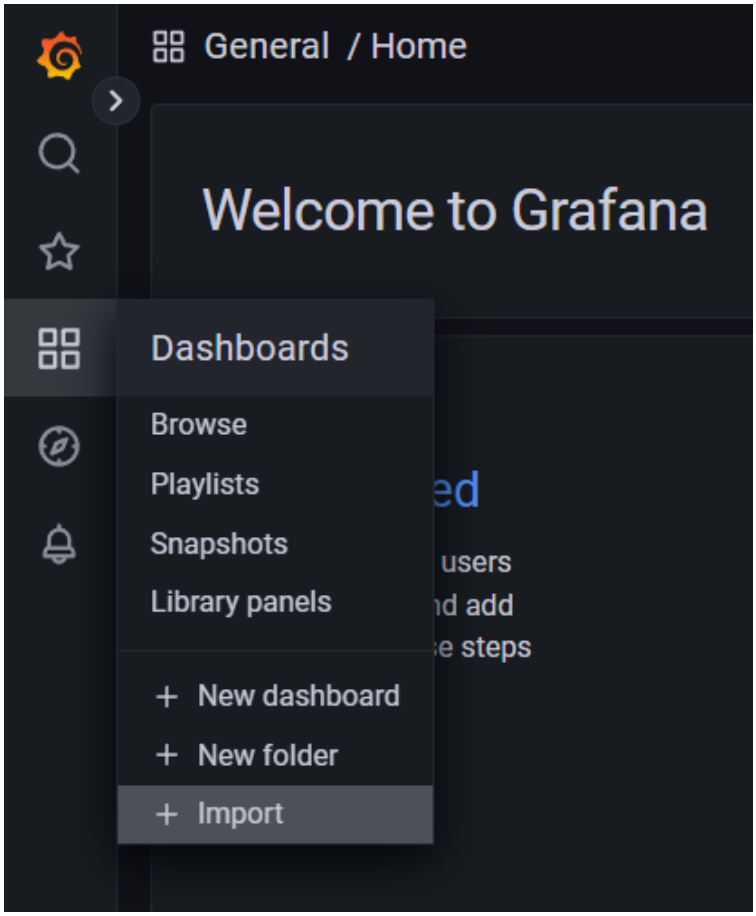
Ako prvé je potrebné nakonfigurovať Grafanu tak, aby sa vedela pripojiť k nášmu Prometheovi. Po prihlásení do Grafany (viď 2.1.5) volíme na ľavom paneli symbol ozubeného kolieska a možnosť „*Data sources*“. Tento krok je zobrazený na obrázku vpravo. Následne sa nám otvorí nové menu, kde volíme modré tlačidlo „*Add data source*“. Zobrazí sa nám ponuka veľkého množstva podporovaných zdrojov. My volíme zdroj Prometheus, ktorý bol v našom prípade uvedený hneď ako prvý. Po kliknutí na možnosť sa nám vytvorí nový zdroj a zobrazí sa konfiguračná obrazovka. Pre jej veľkosť ju sem nebudeme uvádzať ako obrázok. Je však dôležité nakonfigurovať všetky potrebné polia. V našom prípade sme pri testovacom nasadení konfigurovali iba meno a URL nasledovne.

- Meno: Prometheus_first test
- URL: http://localhost:9090

Na samom spodku konfiguračnej obrazovky potom nájdeme tlačidlo „*Save & test*“. Týmto sa otestuje, či má Grafana prístup na nakonfigurovaný zdroj. Očakávame výpis „*Data source is working*“. Týmto sme úspešne nakonfigurovali náš nový zdroj pre získavanie dát na vizualizáciu. Prejdeme teda k nastaveniu monitorovacej obrazovky pre náš modul OpenStack exporter.

Pre monitorovaciu obrazovku využije vopred vytvorenú šablónu, ktorú nakonfiguroval sám autor tohto modulu. Grafana ponúka na svojej webovej stránke veľké množstvo predpripravených šablón pre monitorovacie obrazovky. Začneme otvorením webovej lokality https://grafana.com/grafana/dashboards/14854-openstack-exporter/. K dispozícii máme dve možnosti, a to stiahnutím konfiguračného JSON súboru alebo jednoduchšie skopírovaním ID. My volíme možnosť pomocou ID, pretože je jednoduchšia.

Po skopírovaní ID sa musíme v Grafane znavigovať na bočnom paneli do ponuky Dashboards->import, tak ako zobrazuje Obrázok 2.16.

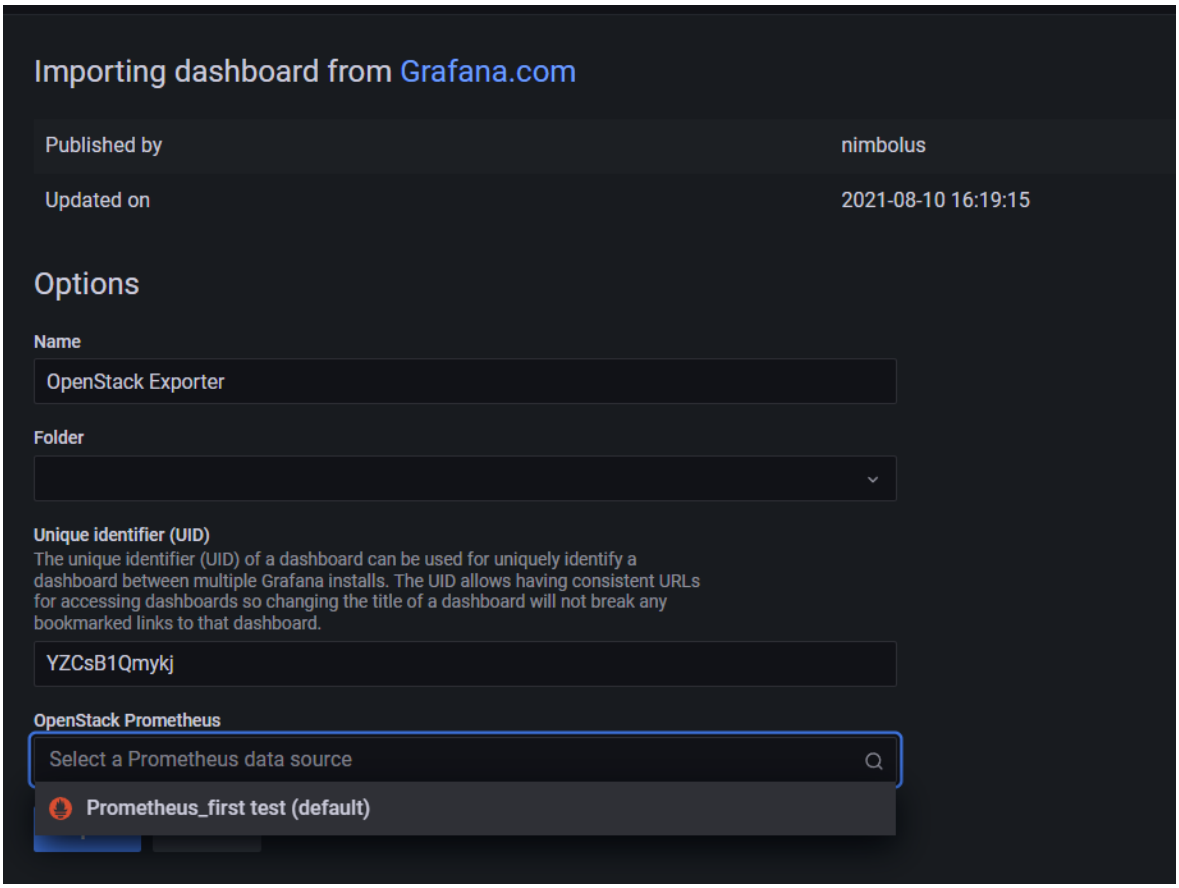


Obrázok 2.16 Importovanie monitorovacej obrazovky p1

Obrázok 2.16 Importovanie monitorovacej obrazovky p1

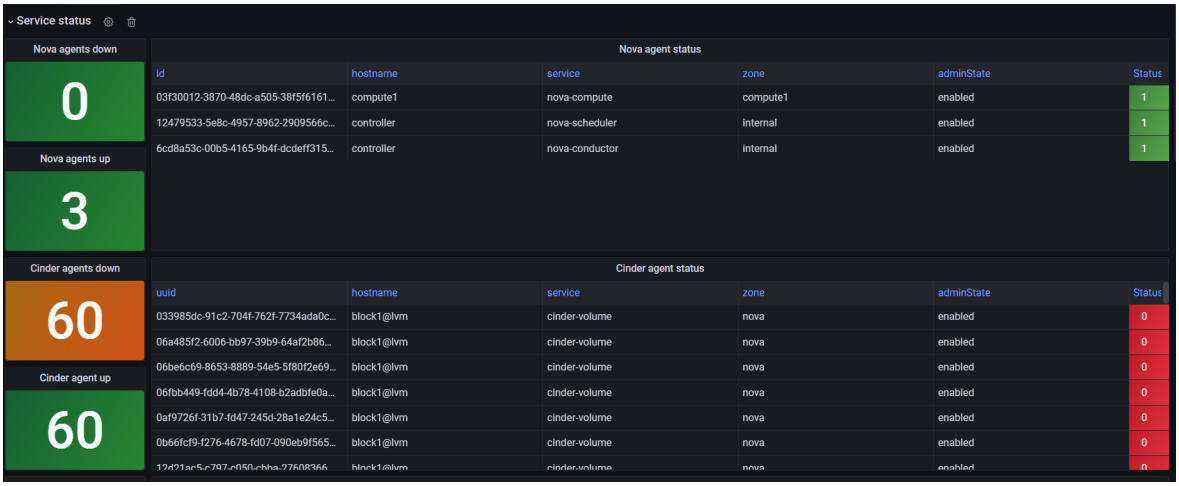
Následne vložíme skopírované ID do miesta na to určeného (import via grafana.com) a klikneme na Load. Upozorňujeme, že pri tomto kroku predpokladáme, že server na ktorom je Grafana nasadená má prístup k internetu. Bez takéhoto prístupu je nutné využiť spôsobom pomocou konfiguračného súboru JSON.

Objaví sa importovacia obrazovka, v ktorej môžeme zmeniť a špecifikovať parametre ako sú, meno obrazovky, priečinok, UID a to najdôležitejšie, zdroj z ktorého sa majú do obrazovky načítavať dáta. V našom prípade to bude nakonfigurovaný zdroj Prometheus. Nižšie uvádzame aj obrázok pre ukážku.



Obrázok 2.18 Importovacia obrazovka pre dashboard

Klikneme na import a proces je hotový. Ak sme všetko správne nastavili, mali by sme mať v ponuke nový dashboard s informáciami o našej OpenStack platforme, podobne ako na Obrázok 2.17.

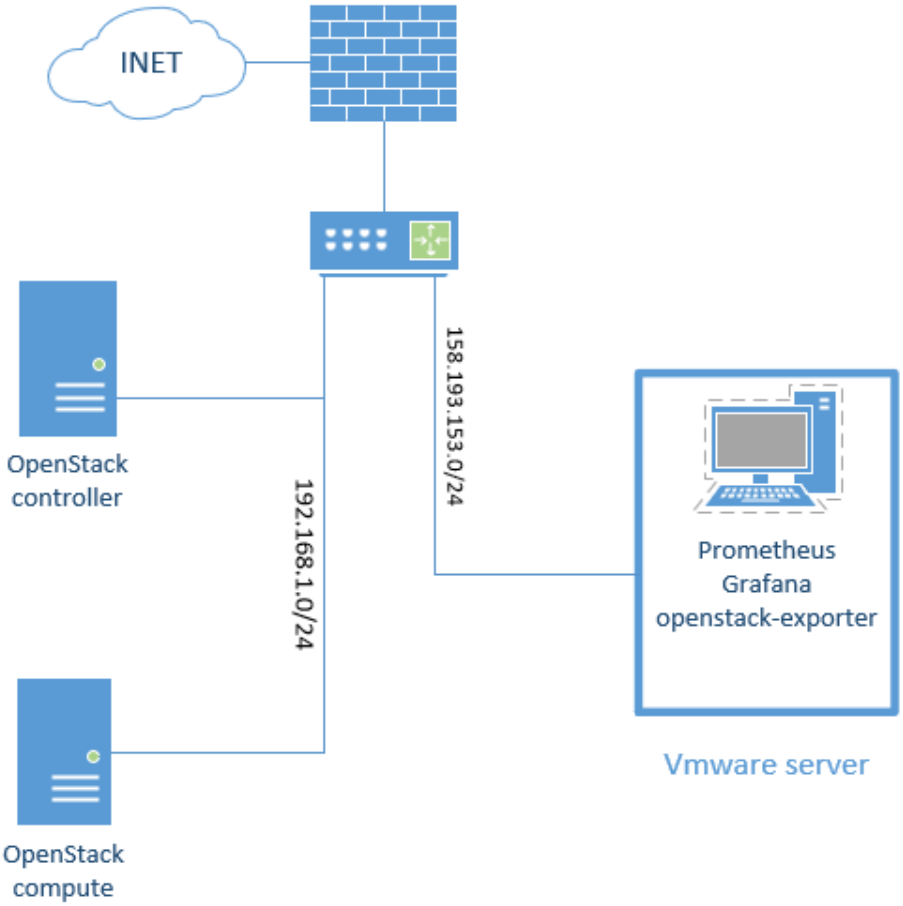


Obrázok 2.17 OpenStack exporter dashboard

3 NASADENIE FUNKČNÉHO RIEŠENIA

Pri funkčnom riešení predpokladáme nie len s monitorovaním samotnej platformy OpenStack, ale aj s monitorovaním priľahlej fyzickej infraštruktúry. Vhodné je aj navrhnúť riešenie, ako monitorovať samotné VM v CC platforme, nie len platformu ako celok.

Začneme s navrhnutou topológiou riešenia. Budeme pracovať s jedným kontrolným a jedným výpočtovým uzlom, ktoré sú v rámci jednej samostatnej siete. Ďalej máme k dispozícii jeden server, ktorý sme v predchádzajúcej kapitole konfigurovali a na ktorom máme nasadený Prometheus spolu s Grafanou. Topológiu uvádzame na obrázku nižšie.



Obrázok 3.1 Topológia funkčného nasadenia

Na topológii vyššie je vidieť, že náš monitorovací server sa nachádza v inej sieti ako uzly OpenStack. Medzi týmito sieťami sa nachádza firewall. Je dôležité poznamenať, že pre správnu funkčnosť je potrebné aby mali medzi sebou tieto siete konektivitu a povolenú minimálne službu http a následne rôzne porty podľa potreby technológie Prometheus.

3.1 POŽIADAVKY A PRVKY SYSTÉMU

Požadujeme aby systém monitoroval:

1. **Stav fyzickej infraštruktúry –** toto budeme vykonávať pomocou pluginu pre systém Prometheus s názvom **node exporter.** Tento plugin funguje pre Linuxové systémy, pričom zverejňuje širokú škálu hardvérových a kernelových metrík z hostiteľského systému. Štandardne zverejňuje metriky na porte 9100, toto je však možné prekonfigurovať na základe potreby.
2. **Stav platformy OpenStack –** toto riešime tiež pomocou systému Prometheus. Nasadenie je popísané v predchádzajúcej kapitole, keďže na tomto nasadení sme demonštrovali konfiguráciu Prometheus + Grafana.
3. **Stav virtuálnych strojov v CC platforme –** pre toto riešenie predpokladáme využitie nástroja *collectd,* spolu s pluginmi *virt* a *write_prometheus.* Za použitia týchto pluginov budeme schopní zbierať metriky z hypervízora na výpočtovom uzle a následne ich zverejňovať pre Prometheus na zber.

3.2 MONITOROVANIE STAVU FYZICKEJ INFRAŠTRUKTÚRY

Monitoring fyzickej infraštruktúry je bežnou praxou pre prevádzkovateľa akéhokoľvek typu zariadení. Prirodzene má motiváciu vedieť v akom stave sa jeho systémy nachádzajú. V našom prípade použijeme na monitorovanie uzlov Prometheus spolu s pluginom **node-exporter.** Tento bude zbierať veľké množstvo informácií o behu a stave daného systému, zverejní ich pomocou http na porte 9100 (je možné zmeniť). Následne potrebujeme Prometheovi nastaviť nový zdroj, aby tieto nazbierané dáta uložil pre ich vizualizáciu.

V našom prípade inštalujeme spomenuté plugin-y na obidva CC uzly. Postup inštalácie je veľmi dobre zadokumentovaný v oficiálnej dokumentácii, preto ho tu už uvádzať nebudeme. Dostupný je na webovej lokalite: https://prometheus.io/docs/guides/node-exporter/. Po inštalácii je potrebné pridať do systému Prometheus nový zdroj dát. Tento postup sme už dokumentovali v 2.1.7, preto ho už nebudeme podrobne dokumentovať. Pre vizualizáciu dát vieme pre jednoduchosť využiť detailnú, prednastavenú šablónu pre Grafana, ktoré je dostupná na webovej lokalite https://grafana.com/grafana/dashboards/1860-node-exporter-full/. Podobne ako v predchádzajúcom prípade, postup pridania monitorovacej obrazovky sme dokumentovali v 2.1.8. Je potrebné postupovať rovnakým spôsobom. Výsledkom, by mala byť monitorovacia obrazovka ako je ukázané na obrázku nižšie.

Obrázok 3.2 Monitorovacia obrazovka Node-Exporterr

Po tomto kroku zbierame a vieme aj vyhodnocovať metriky z fyzických uzlov. Platforma poskytuje aj nástroj pre upozorňovanie administrátorov v prípade, ak metrika prekročí vopred nastavenú hodnotu. Tomuto prvku sme sa však už v rámci projektu nestihli venovať, je však veľmi dôležitým aspektom monitorovacieho systému, preto pri ostrom nasadení je táto funkcionalita potrebná.

3.3 MONITOROVANIE STAVU VMS V CC PLATFORME

Požiadavka od administrátorov systému bola, aby mali prehľad aj čo sa odohráva vo vnútri platformy a v akom stave sú VM vo vnútri. Motiváciou je napríklad identifikovanie nadbytočných, už nepoužívaných strojov alebo detekcia takých, ktoré zbytočne využívajú priveľa prostriedkov a mohlo by vzniknúť podozrenie, že ich používatelia využívajú na inú činnosť, ako bola VM poskytnutá.

Pre riešenie tohto problému sme sa po preskúmaní riešení rozhodli pre využitie démona pre zber systémových štatistík **collectd.** Tento démon využijeme spolu s pluginmi:

- **Virt –** Tento plugin využíva virtualizačné API *libvirt*, ktoré bolo vyvinuté spoločnosťou RedHat. Jeho úlohou je zbierať štatistiky o virtualizovaných hostiteľoch v systéme. Pomocou neho je možné zbierať využitie CPU, sieťových rozhraní a blokových zariadení (úložiská), bez potreby inštalovania akéhokoľvek softvéru na tieto hostiteľské systémy. Dáta sú zbierané priamo z hypervízora. Podporované sú Xen, Qemu a KVM back-endy [14].
- **Write Prometheus –** Plugin vytvorí webový server na porte 9103 (nastaviteľné) a akceptuje žiadosti od Prometheus, pre zber dát z tohto koncového bodu [15].

3.3.1 Nasadenie riešenia

Keďže VM bežia len na výpočtovom uzle, budeme collectd démona inštalovať len v rámci toho systému. Opäť existuje niekoľko spôsobov akým je možná inštalácia. Na danom uzle beží operačný systém CentOS7, pre ktorý je dostupný inštalačný balíček, preto sme sa v našom prípade rozhodli pre túto formu. Všetky ostatné formy inštalácie, ako aj celá dokumentácia je dostupná online na https://collectd.org/wiki/index.php/Main_Page.

Pri našom nasadení sme sa nevyhli väčšiemu množstvu problémov. Niektoré pochádzali zo samotného OS a našej slabšej znalosti pri práci s ním. Pre budúcnosť odporúčame pre nasadenie Debian/Ubuntu based OS. Náš postup inštalácie aj s problémami popisujeme nižšie. Pre spresnenie, celú inštaláciu sme vykonávali ako super používateľ root.

```
#inštalácia collectd, ako prvé potrebujeme aktivovať epel repozitár
yum install epel-release
```

V našom prípade vznikol problém v DNS, kedy yum nevedel nájsť potrebné mirrors. Riešením bola úprava súboru resolv.conf, v ktorom sme pridali ďalší DNS server (8.8.8.8). V prípade úspešnej inštalácie pokračujeme na samotnú inštaláciu démona.

```
#Predpokladaný príkaz
yum install collectd
```

Po zadaní tohto príkazu sme sa stretli s hláškou „No package found“. Po skúmaní sme prišli na riešenie. Potrebné je v danom príkaze špecifikovať, aby sa použil epel repozitár.

```
#Inštalácia collectd s využitím epel repozitára
yum --enablerepo=epel install collectd
#Overenie inštalácie
systemctl status collectd

collectd.service - Collectd statistics daemon
   Loaded: loaded (/usr/lib/systemd/system/collectd.service; disabled;
vendor preset: disabled)
   Active: active (running) since Sun 2023-01-29 15:12:55 CET; 3 days ago
     Docs: man:collectd(1)
           man:collectd.conf(5)
```

Teraz máme k dispozícii nainštalovaného démona, ku ktorému je potrebné zmeniť konfiguráciu. Na základe tej si sám aktivuje požiadané pluginy. Tu je dôležité spomenúť, že niektoré pluginy sú závislé na rôznych knižniciach a balíčkoch (toto je vždy uvedené v collectd dokumentácii ku pluginom). V prípade, ak tieto nie sú nainštalované, je potrebné ich pred tým doinštalovať pre správnu funkčnosť. Nižšie uvádzame bloky rozsiahleho konfiguračného súboru (*/etc/collectd.conf* v našom prípade), ktoré sme upravili tak, aby démon aktivoval požadované pluginy a zbieral dáta z hypervízora.

V sekcii LoadPlugin je potrebné od-komentovať (vymazať znak #) nasledovné riadky.

```
#LoadPlugin virt
#LoadPlugin write_prometheusň
```

V sekcii Plugin Configuration najprv upravíme konfiguráciu pluginu virt. V našom prípade vyzerá nasledovne:

```
<Plugin virt>
        Connection "qemu:///system"
        RefreshInterval 60
        BlockDeviceFormat "target"
        HostnameFormat "uuid"
        InterfaceFormat "address"
        PluginInstanceFormat "uuid"
        ExtraStats "cpu_util disk_err domain_state job_stats_background
perf vcpupin"
</Plugin>
```

Objasnenie významu jednotlivých riadkov je dostupné online na: https://collectd.org/documentation/manpages/collectd.conf.5.shtml#plugin_virt.

Teraz upravíme konfiguráciu pre plugin Write Prometheus. V našom prípade sme konfiguráciu len od-komentovali, základne nastavený port sme nemali dôvod meniť.

```
<Plugin write_prometheus>
        Port "9103"
</Plugin>
```

Pre aktivovanie pluginov teraz potrebujeme reštartovať démona zadaním nasledovného príkazu.

```
systemctl restart collectd
```

Očakávame stav active (running). V našom prípade pozorujeme nižšie uvedené chybové hlásenie.

```
compute1 collectd[24579]: virt plugin: Array index out of bounds:
tag_index = 10
```

Démon však funguje podľa očakávania a doteraz sa nám nepodarilo identifikovať ani dôvod problému a ani, či to má nejaký vplyv na beh démona.

Posledným krokom je pridanie ďalšieho zdroja do systému Prometheus, tak ako už niekoľkokrát pred tým v rámci tejto práce. Pre tento plugin neexistuje vhodná prednastavená šablóna na vizualizáciu nazbieraných metrík, preto budeme musieť monitorovaciu obrazovku vytvoriť ručne. Tomu sa však v tejto práci z dôvodu nedostatku času už nebudeme venovať.

ZÁVER

V rámci tohto experimentu sa nám podarilo preskúmať problematiku monitoringu CC platformy primárne postavenej na technológii *OpenStack*. Definovali sme motiváciu pre monitoring, preskúmali sme metriky vhodné na sledovanie. Ďalej sme sa pokúsili prepojiť problematiku zmluvy o úrovni služby so samotným monitoringom. Popísali sme vybrané riešenie *Prometheus* spolu s vizualizačnou platformou *Grafana*.

V praktickej časti práce sa nám podarilo preskúmané riešenie *Prometheus* aj v praxi experimentálne nasadiť. Monitorovali sme testovaciu *OpenStack* platformu na KIS. Nasadili sme samotný *Prometheus* server na jednom stroji spolu s *Grafana* vizualizačným riešením. Na monitorovanie celkového stavu CC platformy sme nasadili plugin *openstack exporter*. Monitorovanie fyzickej infraštruktúry sme riešili pomocou pluginu *node exporter*. Pri tomto nasadení sme dané pluginy prepojili s serverom *Prometheus* a nazbierané dáta sme vizualizovali v platforme *Grafana*. Pre monitorovanie stavu VM v CC platforme sme využili démona *collectd*, ktorý s príslušnými pluginmi zbieral štatistiky priamo z hypervízora na výpočtovom uzle. Dáta z tohto démona sme taktiež prepojili so serverom *Prometheus*, avšak dáta sme pre nedostatok času už nedokázali vizualizovať.

Výsledkom tejto práce je experimentálne nasadenie, ktoré sme dôkladne zdokumentovali tak, aby bolo reálne nasadenie na novú CC platformu jednoduchšie a rýchlejšie. Dokumentácie postupov v tejto práci môžu ďalej slúžiť ako pomôcka pri nasadzovaní monitorovacích riešení nie len pre CC platformy.

ZOZNAM LITERATÚRY

[1] G. Aceto, A. Botta, W. De Donato, a A. Pescapè, “Cloud monitoring: A survey”, *Comput. Networks*, roč. 57, č. 9, s. 2093–2115, 2013, doi: 10.1016/j.comnet.2013.04.001.

[2] S. Kundu, “Hybrid Cloud Workload Monitoring as a Service Hybrid Cloud Workload Monitoring as a Service A Project Report The Faculty of the Department of Computer Science San José State University Of the Requirements for the Class CS 298”, 2021.

[3] D. Tovarnňák a T. Pitner, “Towards multi-tenant and interoperable monitoring of virtual machines in cloud”, *Proc. - 14th Int. Symp. Symb. Numer. Algorithms Sci. Comput. SYNASC 2012*, s. 436–442, 2012, doi: 10.1109/SYNASC.2012.55.

[4] J. Shao, H. Wei, Q. Wang, a H. Mei, “A runtime model based monitoring approach for cloud”, *Proc. - 2010 IEEE 3rd Int. Conf. Cloud Comput. CLOUD 2010*, č. July 2010, s. 313–320, 2010, doi: 10.1109/CLOUD.2010.31.

[5] A. Kertesz *et al.*, “Enhancing Federated Cloud Management with an Integrated Service Monitoring Approach”, *J. Grid Comput.*, roč. 11, č. 4, s. 699–720, 2013, doi: 10.1007/s10723-013-9269-0.

[6] G. Da Cunha Rodrigues *et al.*, “Monitoring of cloud computing environments: Concepts, solutions, trends, and future directions”, *Proc. ACM Symp. Appl. Comput.*, roč. 04-08-Apri, č. March 2018, s. 378–383, 2016, doi: 10.1145/2851613.2851619.

[7] J. Montes, A. Sánchez, B. Memishi, M. S. Pérez, a G. Antoniu, “GMonE: A complete approach to cloud monitoring”, *Futur. Gener. Comput. Syst.*, roč. 29, č. 8, s. 2026–2040, 2013, doi: 10.1016/j.future.2013.02.011.

[8] M. Drozdík, “Diplomová práca - Implementácia nástrojov na prácu so systémovými logmi a ich vizualizácia v prostredí katedry KIS”, s. 79, 2017.

[9] M. Dočár, “Diplomová práca - Zber a archivácia logov na KIS UNIZA”, s. 81, 2017.

[10] L. Chen, M. Xian, a J. Liu, “Monitoring System of OpenStack Cloud Platform Based on Prometheus”, *Proc. - 2020 Int. Conf. Comput. Vision, Image Deep Learn. CVIDL 2020*, č. Cvidl, s. 206–209, 2020, doi: 10.1109/CVIDL51233.2020.0-100.

[11] “Prometheus documentation”. https://prometheus.io/docs/introduction/overview/.

[12] “Prometheus SNMP”. https://linuxhint.com/monitor-network-devices-prometheus/.

[13] “OpenStack exporter for Prometheus”. https://github.com/openstack-exporter/openstack-exporter#readme.

[14] “Collectd virt plugin documentation”. https://collectd.org/wiki/index.php/Plugin:virt (cit feb. 01, 2023).

[15] “Collectd Write Prometheus plugin documentation”. https://collectd.org/wiki/index.php/Plugin:Write_Prometheus (cit feb. 01, 2023).