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

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD VLÁDY SLOVENSKEJ REPUBLIKY

OBSAH

Úvod ... 7
2 Monitoring v prostredí Cloud computing-u ... 8
2.1 Cloud Computing ... 8
2.2 Motivácia pre monitorovanie ... 8
2.2.1 Kapacita a plánovanie zdrojov ... 9
2.2.2 Kapacita a manažovanie zdrojov ... 9
2.2.3 Manažovanie dátového centra ... 10
2.2.4 Manažment SLA ... 10
2.2.5 Fakturovanie ... 10
2.2.6 Zisťovanie problémov ... 11
2.2.7 Manažovanie výkonu ... 11
2.2.8 Bezpečnosť ... 11
2.3 Pohľady na monitoring ... 11
3 Problematika zmluvy o úrovni služby v Cloud computing-u ... 14
3.1 Komponenty zmluvy o úrovni služby ... 14
3.1.1 Základné komponenty ... 14
3.1.2 Manažérske komponenty ... 15
3.1.3 Atribúty služby ... 15
3.1.4 SLA QoS metriky ... 15
3.1.5 Právne komponenty ... 16
3.2 SLA metriky pre IaaS Cloud Computing ... 16
3.3 Indikátory úrovne služby ... 17
3.4 Ciele úrovne služby ... 17
3.5 Zhrnutie problematiky SLA, SLI a SLO ... 18
4 Definovanie a problematika metrík ... 19
4.1 Metriky z pohľadu zákazníka ... 19
4.2 Metriky z pohľadu poskytovateľa ... 20
4.2.1 Metriky sieťovej vrstvy ... 21
4.2.2 Metriky hardvérovej vrstvy ... 22
4.2.3 Metriky vrstvy operačného systému ... 24
4.3 Prevedenie meraní vybraných metrík ... 25
4.3.1 Výpočtový procesor ... 26
4.3.2 Operačná pamäť ... 27
4.3.3 Úložisko ... 28
4.3.4 Sieťové prvky ... 29
5 Monitoring platformy *OpenStack* ... 31
5.1 OpenStack ... 31
5.2 Monitorovanie v rámci OpenStack ... 32
5.3 Problematika viac-klientskej architektúry v kontexte monitoringu ... 34
5.4 Prezentovanie nazbieraných dát pre zúčastnené strany ... 35
5.4.1 Reporty ... 35
5.4.2 Prístup k dátam z monitorovania – vykresľovanie metrík ... 37
5.4.3 Prístup k dátam z monitorovania – signalizácia ... 38

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD VLÁDY SLOVENSKEJ REPUBLIKY

6 Prieskum vhodných nástrojov na riešenie ... 40
6.1 Všeobecný prehľad nástrojov ... 40
6.1.1 Nagios ... 40
6.1.2 Graphite ... 40
6.1.3 Collectd ... 41
6.1.4 Grafana ... 41
6.2 Zabbix ... 42
6.3 Prometheus ... 42
6.4 Zhrnutie ... 43
7 Návrh technického riešenia ... 44
7.1 Požiadavky na riešenie ... 44
7.2 Výber technológií a hardvérové požiadavky ... 45
7.2.1 Prometheus server ... 46
7.2.2 Nástroje pre zber metrík ... 46
7.2.3 Vizualizácia a upozorňovanie ... 48
7.3 Architektúra riešenia ... 48
7.3.1 Centrálny server ... 48
7.3.2 Ostatné komponenty ... 51
7.3.3 Celková architektúra ... 52
7.4 Nasadenie ... 53
7.4.1 Prometheus server ... 54
7.4.2 Node exporter ... 55
7.4.3 Snmp exporter ... 55
7.4.4 Openstack exporter ... 57
7.4.5 Collectd ... 58
7.4.6 Grafana ... 59
7.4.7 Aplikácia ako služba ... 60
7.4.8 Konfigurácia *Prometheus* a overenie ... 61
7.5 Príprava pre vizualizáciu ... 63
7.5.1 Prepojenie Prometheus a Grafana ... 64
7.5.2 Importovanie prednastavených šablón ... 64
7.5.3 Priečinky ... 65
7.5.4 Vytvorenie používateľov ... 65
7.5.5 Vytvorenie obrazoviek a panelov ... 66
7.6 Monitorovacia obrazovka pre klientov ... 68
7.7 Monitorovacia obrazovka pre administrátora ... 70
7.7.1 Vizualizácia monitoringu *OpenStack* služieb ... 70
7.7.2 Vizualizácia rezervovaných a dostupných zdrojov pre cloud ... 72
7.7.3 Vizualizácia monitoringu fyzickej infraštruktúry ... 73
7.7.4 Ostatné prvky monitorovacej obrazovky ... 75
7.8 Upozorňovanie ... 76
7.8.1 Upozorňovacie pravidlá ... 76
7.8.2 Kontaktné body ... 77
7.8.3 Notifikačné politiky ... 78
7.9 Testovanie ... 79

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD VLÁDY SLOVENSKEJ REPUBLIKY

7.9.1 Vyťaženie zdrojov a dostupnosť 79
7.9.2 Služby OpenStack 83
7.9.3 Podozrivé správanie VM 83
Záver 85
Zoznam použitej literatúry 86
Prílohy 90

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD VLÁDY SLOVENSKEJ REPUBLIKY

ZOZNAM OBRÁZKOV

Obrázok 2.1 Pohľady na monitoring [5] 12
Obrázok 4.1 Sieťové výkonové metriky [13] 22
Obrázok 4.2 Meranie oneskorenia v slučke [30] 29
Obrázok 5.1 OpenStack mapa [33] 31
Obrázok 5.2 Prehľad zodpovednosti pri CC službách [46] 35
Obrázok 6.1 Prometheus architektúra [60] 43
Obrázok 7.1 Architektúra centrálneho servera 49
Obrázok 7.2 Komunikácia openstack-exporter s OpenStack API 50
Obrázok 7.3 Komunikácia snmp_exporter so sieťovými zariadeniami 51
Obrázok 7.4 Architektúra ostatných komponentov 51
Obrázok 7.5 Topológia monitorovaného cloudu 52
Obrázok 7.6 Prostredie monitorovacieho servera 52
Obrázok 7.7 Celková architektúra 53
Obrázok 7.8 Rozhranie snmp exporter 57
Obrázok 7.9 Začiatok výpisu metrík získaných pomocou *snmp exporter* 57
Obrázok 7.10 Webové rozhranie Grafana 60
Obrázok 7.11 *Prometheus* rozhranie - práce 63
Obrázok 7.12 Grafana pridanie zdroja dát 64
Obrázok 7.13 *Grafana* vytvorenie obrazovky - navigačný panel 66
Obrázok 7.14 Popis rozhrania obrazovky 67
Obrázok 7.15 Ukážka časti obrazovky pre klientov – informácie o prostredí 68
Obrázok 7.16 Ukážka časti obrazovky pre klientov – informácie o inštanciách 69
Obrázok 7.17 Detaily služby cinder 71
Obrázok 7.18 Vizualizácia monitoringu OpenStack služieb 72
Obrázok 7.19 Využité zdroje v cloude 73
Obrázok 7.20 Vizualizácia využitých zdrojov na uzloch 74
Obrázok 7.21 Prehľad zariadení v infraštruktúrre 74
Obrázok 7.22 Overenie projektu *Sochor_Monitoring_Test* 79
Obrázok 7.23 Pridanie výpočtového uzla 80
Obrázok 7.24 Vyťaženie systému - *htop* 80
Obrázok 7.25 Vyťaženie systému - Grafana 80
Obrázok 7.26 Vytvorené upozornenia - panel pre aktívne upozornenia 81
Obrázok 7.27 Vygenerované upozornenie - discord 81
Obrázok 7.28 Nedostupnosť uzla - monitorovacia obrazovka 82
Obrázok 7.29 Upozornenie - výpadok uzla 82
Obrázok 7.30 Výpadok služby cinder 83
Obrázok 7.31 Upozornenie výpadku služby cinder 83
Obrázok 7.32 Čakajúce upozornenie vyťaženia inštancie 84
Obrázok 7.33 Aktívne upozornenie vyťaženia inštancie 84

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD VLÁDY SLOVENSKEJ REPUBLIKY

ZOZNAM TABULIEK

Tabuľka 3.1 SLA QoS metriky pre IaaS [6], [9] 16
Tabuľka 4.1 *CloudWatch* metriky inštancie [11] 20
Tabuľka 5.1 Úrovne upozornení [52] 39
Tabuľka 7.1 Zoznam komponentov na centrálnom serveri 50
Tabuľka 7.2 Zoznam a umiestnenie nasadených exportérov 61
Tabuľka 7.3 Zoznam pravidiel 77

ZOZNAM SKRATIEK

API – Aplikačné programovacie rozhranie
CC – Cloud Computing
CLI – Príkazový riadok (*Command line interface*)
CPU – Centrálna riadiaca jednotka (procesor)
CS – Cloudová služba (*Cloud Service*)
CSP – Poskytovateľ cloudovej služby (*Cloud Service Provider*)
GB – Gigabajt
HTTP – Hypertextový prenosový protokol
IaaS – Infraštruktúra ako služba (*Infrastructure as a Service*)
IT – Informačné technológie
KPI – Kľúčový indikátor výkonu (*Key performance indicator*)
MIB – Databáza informácií manažérstva (Management Information Base)
MS – Monitorovací server
NPMs – Sieťové výkonové metriky (*Network Performance Metrics*)
OS – Operačný systém
QoS – Kvalita služby (*Quality of Service*)
RAM – Operačná pamäť
RU – Riadiaci uzol
SLA – Zmluva o úrovni služby (*Service Level Agreement*)
SLI – Indikátor úrovne služby (*Service Level Indicator*)
SLO – Cieľ úrovne služby (*Service Level Objective*)
SNMP – Jednoduchý protokol manažérstva siete
SZ – Sieťové zariadenia
URL – Jednotný Vyhľadávač Prostriedka (*Uniform Resource Locator*)
vCPU – Virtuálna centrálna riadiaca jednotka
VM – Virtuálny stroj (*Virtual Machine*)
VU – Výpočtový uzol

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
ÚRAD VLÁDY SLOVENSKEJ REPUBLIKY

ÚVOD

Hlavným cieľom tohto dokumentu je návrh a implementácia monitorovacieho systému pre cloud, postaveného na platforme *OpenStack*, ktorý poskytuje infraštruktúru ako službu. V rámci požiadaviek je nevyhnutné, aby systém umožnil meranie technických parametrov cloudu a sieťovej infraštruktúry. Tieto informácie by mali byť prezentované takým spôsobom, aby poskytovateľ dokázal vyhodnotiť prípadné ohrozenie poskytovania služieb. Systém by mal taktiež vhodným spôsobom prezentovať informácie o stave a využití zákazníckych prostredí.

Pre dosiahnutie nášho cieľa je nevyhnutné identifikovať motiváciu pre monitorovanie, rozdeliť rôzne pohľady na tento proces a zoznámiť sa s problematikou SLA. Rovnako by sme mali preskúmať spôsoby extrakcie a prezentovania významných metrík z prostredia infraštruktúry, a platformy *OpenStack*. Dôležitým faktorom je aj preskúmanie dostupných riešení, ktoré môžeme neskôr využiť v praktickej časti. V rámci samotnej implementácie je nevyhnutné navrhnúť architektúru, na základe ktorej budeme môcť nasadiť systém a otestovať jeho funkčnosť.

1 MONITORING V PROSTREDÍ CLOUD COMPUTING-U

Na úvod je vhodné definovať, čo to vlastne pojem *Cloud computing* reprezentuje, V tejto kapitole sa pokúsime identifikovať motiváciu za monitorovaním takéhoto systému a rozdelíme rôzne pohľady.

1.1 Cloud Computing

Pre vysvetlenie sa môžeme inšpirovať jedným z momentálnych lídrov v oblasti poskytovania CC služieb, spoločnosťou Amazon. Vo všeobecnosti platí, že CC je poskytovanie informačno-technologických (IT) zdrojov cez internet a na požiadanie. Je preň typická fakturácia založená na princípe zaplať za to, čo použiješ (*pay-as-you-go*). Zákazníci tým pádom nepotrebujú spravovať ich vlastnú fyzickú infraštruktúru, ale vedia pristupovať k výpočtovým prostriedkom, úložisku a inej potrebnej IT infraštruktúre, ktorú im poskytovateľ vie poskytnúť [1].

Medzi hlavné benefity CC sa uvádzajú agilnosť, elasticita a úspora nákladov. Na základe poskytovanej služby existujú tri hlavné typy modelov služieb, [1]. Vieme ich definovať ako:

- **Infraštruktúra ako služba (IaaS)** – Zvykne sa definovať aj ako základná vrstva CC. Typicky poskytuje prístup k sieťovým, výpočtovým a dátovým (úložisko) zdrojom, pričom dáva používateľovi najväčší level flexibility a riadenia IT zdrojov z pomedzi ostatných typov CC. Práve tento typ je z pohľadu práce najdôležitejší.
- **Platforma ako služba (PaaS) –** Odstraňuje zákazníkovi nutnosť spravovať infraštruktúru. Poskytovateľ v tomto prípade dodáva priamo platformu, na ktorej je možný vývoj aplikácií.
- **Softvér ako služba (SaaS) –** Pre bežného používateľa internetu je tento typ CC najznámejší. Jedná sa najmä koncové aplikácie (napríklad webový email klient) [1].

1.2 Motivácia pre monitorovanie

Záujem o monitorovanie CC prostredia má ako poskytovateľ, tak aj používateľ takejto platformy. Je zrejmé, že prevádzkovateľ služby potrebuje detegovať zlyhania, ako na fyzickej, tak vo virtuálnej infraštruktúre. Jeho záujmom by taktiež malo byť optimalizovať chod infraštruktúry [2].

Keďže vzťah medzi poskytovateľom a zákazníkom by mal byť formálne zadefinovaný, v tomto prípade v zmluve o úrovni služby (SLA), monitorovací systém by mal vedieť

upozorniť vlastníka systému na prípadné problémy, ktoré môžu ovplyvniť výkon služby, a tým spôsobiť finančné škody. SLA sa podrobnejšie venujeme v kapitole 2.

Zákazník má záujem monitorovať jemu poskytnuté zdroje, a to najmä aby mal prehľad, či potrebuje pridelené množstvo, keďže väčšie množstvo pridelených prostriedkov znamená vyššie náklady. Na druhej strane musí vedieť, či jemu pridelené zdroje stále postačujú požiadavkám jeho projektu, a či nie je potrebné ich navýšiť [2].

Existuje veľké množstvo procesov, ktoré sú pre beh CC platformy kritické, pričom pre každý z nich je ich monitorovanie kľúčovou úlohou. Dané procesy, ako aj rolu ich monitoringu uvádzame nižšie [3].

1.2.1 Kapacita a plánovanie zdrojov

Pred masovým adoptovaním CC bolo pre prevádzkovateľov a vývojárov aplikácii jednou z najväčších výziev plánovanie a manažovanie fyzických zdrojov, na ktorých daná aplikácia fungovala. Aby dokázali zaručiť istú kvalitu služby, potrebovali kvantifikovať, aké množstvo prostriedkov potrebujú pre hladký beh procesov. Vo všeobecnosti platí, že takýto odhad je možné vytvoriť na základe testovania, monitorovania a štatistickej analýzy, no reálne hodnoty môžu byť nepredvídateľné a vysoko variabilné. V tomto je pre poskytovateľov aplikačných služieb CC veľkou výhodou, keďže celý tento proces monitorovania a spravovania infraštruktúry prechádza na prevádzkovateľa CC systému. Tu sa dostávame k faktu, ktorý uvádzame aj vyššie v kapitole 1.2. Monitorovanie priľahlej infraštruktúry je z pohľadu poskytovateľa CC služieb kritické, pokiaľ chce sledovať a predpovedať vývoj parametrov zastúpených v procese zachovania kvality služby (QoS). Na základe výstupov z takéhoto monitorovacieho systému by mal vedieť správne naplánovať rozvoj fyzickej infraštruktúry a poskytovaných zdrojov tak, aby boli rešpektované SLA, uzatvorené zo všetkými zákazníkmi [3].

1.2.2 Kapacita a manažovanie zdrojov

Základným predpokladom funkčného monitorovacieho systému je schopnosť zachytiť aktuálny stav CC platformy. Hlavnou doménou v CC je virtualizácia, ktorá zakrýva heterogenitu priľahlej fyzickej infraštruktúry, a tým prináša vyššiu úroveň zložitosti pre poskytovateľa, ktorý tým pádom potrebuje manažovať ako fyzickú, tak aj virtuálnu infraštruktúru. Virtuálne zdroje môžu migrovať medzi fyzickými zariadeniami, čo môže spôsobovať variabilnosť dostupných zdrojov na jednotlivých fyzických zariadeniach. Správne nasadený monitorovací systém dokáže administrátora upozorniť, ak by dochádzalo k nedostatku výpočtových zdrojov na niektorom z uzlov. Okrem toho je pre veľa

zákazníkov dôležitý aspekt dostupnosti služby. Monitorovanie dokáže odhaliť anomálie v chode systému, a preto ho považujeme za potrebný v kontexte zaručenie čo najväčšej dostupnosti [3].

1.2.3 Manažovanie dátového centra

Cloudové služby bývajú poskytované na podklade veľkých dátových centier, ktorých riadenie je dôležitou úlohou pre bezproblémový chod komplexného systému. Aktivity riadenia dátového centra vieme rozdeliť do dvoch základných úloh:

- **Monitorovanie** – Sledovanie požadovaných hardvérových a softvérových metrík.
- **Dátová analýza** – Spracovanie nazbieraných metrík za účelom vyhodnotenia stavu systému.

Je dôležité, aby takýto monitorovací systém umožňoval veľké množstvo operácií v reálnom čase, keďže v dátovom centre je predpoklad výskytu veľkého množstva prvkov infraštruktúry. V tomto kontexte je hlavná motivácia pre monitoring úspora energie, čiže optimalizácia behu fyzických zariadení [3].

1.2.4 Manažment SLA

Monitoring je povinnou a nenahraditeľnou súčasťou procesu overovania dodržiavania SLA, napríklad keď sa vykonávajú audítorské aktivity pre zistenie dodržiavania regulácií (napr. v prípade, ak sa služba ponúka štátnym orgánom). Dáta z monitorovacieho systému môžu poskytovateľovi pomôcť naformulovať SLA viac realisticky a navrhnúť udržateľný model fakturovania [3].

1.2.5 Fakturovanie

Ako už bolo spomenuté vyššie, jednou zo základných charakteristík CC je poskytovanie služieb na báze „plať za to, čo použiješ“. To umožňuje zákazníkovi platiť proporcionálne k využívaniu informačných zdrojov. Základným predpokladom pre fakturáciu takýmto spôsobom je existencia procesu, ktorý sleduje množstvo využitých zdrojov zákazníkov. Tento proces môže byť, buď samostatný, alebo spojený v jedno s monitorovacím riešením [3].

1.2.6 Zisťovanie problémov

Infraštruktúra CC býva často komplexná a predstavuje výzvu v prípade presného určenia pôvodu vzniknutého problému. Pri výskyte problému je potrebné analyzovať niekoľko možných komponentov, ktoré mohli zlyhať. Ďalší fakt, ktorý pridáva na zložitosti, je aj to, že samotné komponenty sa môžu skladať z niekoľkých vrstiev (napr. fyzická alebo virtuálna vrstva, operačný systém (OS) hostiteľa alebo virtuálneho stroja (VM) a pod.). Zrozumiteľný a detailný monitorovací systém dokáže poskytovateľovi pomôcť rýchlejšie, a presnejšie diagnostikovať vzniknutý problém. Okrem toho by si mal zákazník vedieť overiť, či sa prípadná chyba vyskytuje na jeho, alebo prevádzkovateľovej strane [3], [4].

1.2.7 Manažovanie výkonu

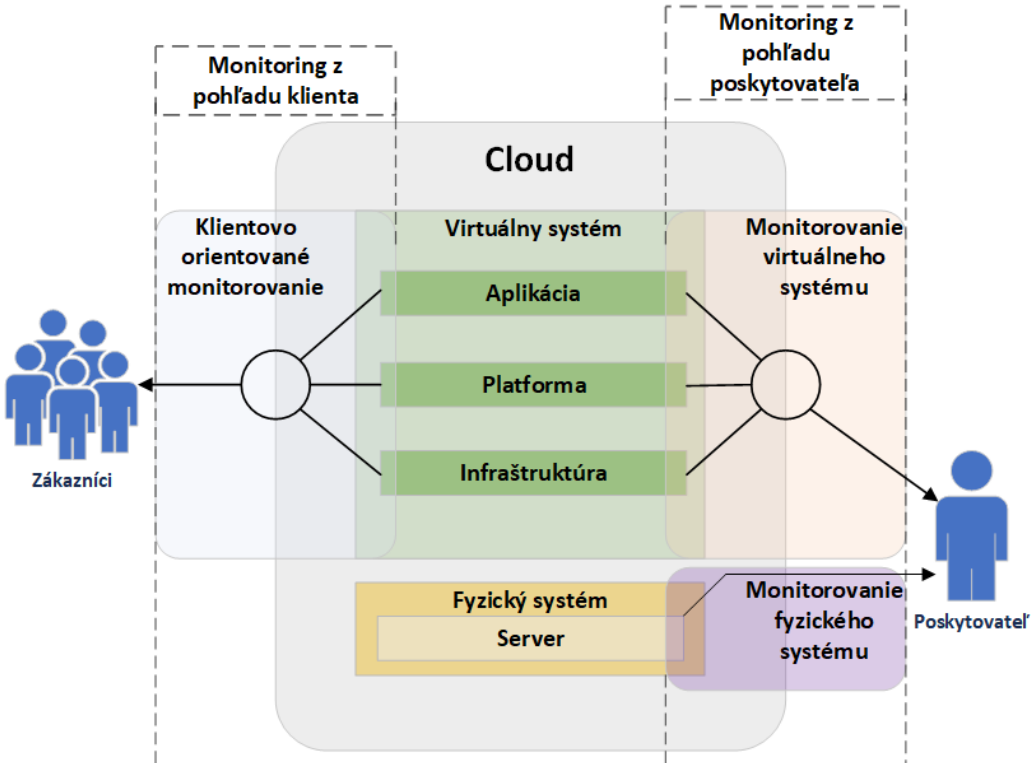
V praxi platí, že niektoré výpočtové uzly v CC poskytujú väčší výkon ako iné. Na bežného zákazníka toto nemá zásadný vplyv, keďže záťaž sa zvykne rovnomerne rozdeľovať a pokles výkonu nebýva zväčša významný. Môžu však existovať zákazníci, ktorí poskytujú s využitím CC kritické služby. Pre nich je každá zmena výkonu významná, pričom dostupnosť výpočtových zdrojov je kritická. Pri takýchto službách zvykne daný zákazník využívať niekoľko rôznych poskytovateľov, pričom si vyberá toho aktuálneho (na ktorom bude v tom momente bežať služba) na základe poskytnutého, vnímaného výkonu. Zákazník má v tomto prípade záujem monitorovať jeho vnímaný výkon služby, pričom poskytovateľ mu chce poskytnúť čo najspoľahlivejšie zdroje. Pre obidve strany je v tomto prípade monitorovanie nutnosťou [3].

1.2.8 Bezpečnosť

Bezpečnosť v CC je dôležitá z viacerých dôvodov. V prvom rade je to jeden z hlavných faktorov, ktorý zákazníka odrádza od migrovania jeho infraštruktúry do takéhoto prostredia. Pre manažovanie bezpečnosti v CC infraštruktúre alebo aplikáciách, je potrebný správny monitorovací systém. V prípade poskytovania služieb niektorým štátnym organizáciám musí CC spĺňať prísne kritériá týkajúce sa regulácií ohľadom správy dát. Toto je možné dosiahnuť pomocou monitorovacieho systému, ktorý vie vierohodne dokázať, že používateľove dáta neopustili hranice krajiny [3].

1.3 Pohľady na monitoring

Pre praktické nasadenie riešenia potrebujeme určiť informácie, ktoré bude daný systém zbierať. Primárne sa na problematiku pozeráme z dvoch pohľadov. Popisujeme ich nižšie.



Obrázok 1.1 Pohľady na monitoring [5]

Obrázok 1.1 graficky zobrazuje, čo by sa vo všeobecnosti malo monitorovať a komu dáta z jednotlivých častí prezentovať. Z pohľadu tejto práce nie je zaujímavá časť platformy, ani aplikácie. Môžeme vidieť, že poskytovateľ cloudovej služby (CSP) by mal disponovať informáciami o stave fyzického systému a virtuálnej infraštruktúry, čo v prípade IaaS platformy predstavuje najmä VMs. Zákazník nemá motiváciu sledovať fyzický systém, preto ho zaujíma len stav jeho virtuálnych zdrojov. Poskytovateľ mu môže nejakým spôsobom sprostredkovať takéto informácie.

Keďže model CC je vždy úzko spätý s garanciou kvality služby, užitočné monitorovacie dáta by mali byť späté s podmienkami definovanými v SLA (viď kapitolu 2). Definovanie metrík pre obidve formy monitoringu, ako z pohľadu klienta, tak poskytovateľa, je kritickým aspektom monitorovacieho systému. Exituje však niekoľko problémov, ktoré je potrebné v tomto kontexte brať do úvahy [5].

V prípade klientovo orientovaného monitoringu musia metriky poskytovať informácie pomocou rovnakých parametrov, ako je definované v SLA [5]. Pre príklad uvedieme scenár, kedy CSP garantuje pre každého zákazníka maximálnu kapacitu úložiska pre jeho VM

v GigaBajtoch (GB). V takom prípade musí CSP monitorovať úložisko každej VM u všetkých zákazníkov, a následne tieto dáta vhodne interpretovať v rovnakých jednotkách.

V prípade monitorovania virtuálneho systému platí rovnaký princíp. To, čo by sa malo monitorovať, je taktiež úzko späté s SLA. Avšak pre CSP by tieto dáta mali poskytnúť dodatočné informácie [5]. Pre lepšie pochopenie nadviažeme na príklad v prechádzajúcom odseku. Okrem monitorovania úložiska na každej VM by mal CSP zbierať dáta aj o celkovom virtuálnom úložisku v CC platforme. Mal by poznať celkové využitie úložiska všetkými klientami. Tieto informácie by mali slúžiť len pre CSP.

Prípad monitorovania priľahlého fyzického systému je rozdielny od predchádzajúcich dvoch, keďže sa týka nižšej úrovne infraštruktúry, na ktorej stojí celá CC platforma. To znamená, že potrebné metriky sú rovnaké ako pri bežných distribuovaných systémoch. Medzi tieto metriky patria z hardvérovej strany napríklad:

- využitie CPU,
- využitie pamäte RAM,
- využitie fyzického úložiska a
- sieťová prevádzka [5].

Zo strany metrík OS sem môžeme zaradiť napríklad:

- zaťaženie systému a
- využitie virtuálnej pamäte, keďže jej správu má na starosti OS [5].

Takto nazbierané informácie slúžia prioritne pre manažment celého systému. V tomto prípade je spojenie s SLA zakorenené vo vzťahu medzi nízko-úrovňovým fyzickým systémom a vysoko-úrovňovým virtuálnym systémom, ktorý stojí na danej infraštruktúre [5]. Metrikám sa v tejto práci budeme ešte venovať bližšie v kapitole 3.

2 PROBLEMATIKA ZMLUVY O ÚROVNI SLUŽBY V CLOUD COMPUTING-U

Keďže v predchádzajúcej kapitole sme identifikovali manažovanie SLA ako jednu z motivácií pre vytvorenie monitorovacieho systému, v tejto kapitole sa tejto problematike povenujeme podrobnejšie. SLA vieme definovať ako formát obsahujúci popis dohodnutých služieb, parametrov úrovne služby a garancií v rámci kvality služby (QoS). Dôležitou časťou tohto dokumentu je aj časť venujúca sa postihom v prípade porušenia ktorejkoľvek z dohodnutých častí. Postihy sú zväčša uvádzané kvantifikovateľným spôsobom. SLA tvorí významný kontrakt medzi poskytovateľom služby a zákazníkom, ktorý istým spôsobom interaguje s poskytovanou službou. Kľúčovým aspektom SLA je poskytnutie jasnej definície základných častí služby, ako je výkon, dostupnosť, fakturovanie a pod. Tento formát sa využíva v širokej škále odvetví. V IT priemysle je to napríklad poskytovanie webových služieb, riadenie dátového centra, sieťové odvetvia atď. Konkrétne znenie a opis SLA sa môže v rámci jednotlivých odvetví meniť, zatiaľ čo základná myšlienka zostáva rovnaká [6].

2.1 Komponenty zmluvy o úrovni služby

SLA obsahuje komponenty, ktoré pre zjednodušenie vieme zatriediť do jednotlivých kategórií. Vďaka nim dokáže byť spracovanie novej SLA systematickejšie, čo uľahčuje samotný proces tvorby takéhoto dokumentu. Zjednodušene vieme tieto komponenty roztriediť do piatich kategórií, ktoré uvádzame nižšie spolu s tými najdôležitejšími z hľadiska SLA v CC [7].

2.1.1 Základné komponenty

Medzi základne komponenty SLA sa radí:

- **Predmet SLA** – Definuje prečo sa spisuje daná SLA, definuje ktoré strany sú v nej zainteresované.
- **Dátum začiatku platnosti a exspirácie** – Časový interval počas ktorého je daný dokument platný. V prípade záujmu obidvoch strán je možné tento interval predlžovať.
- **Číslo verzie** – V prípade, ak sa dotknuté strany dohodnú na akejkoľvek zmene v znení SLA, je potrebné upraviť číslo verzie dohodnutým spôsobom, napríklad navýšením hodnoty [6], [7].

2.1.2 Manažérske komponenty

Medzi manažérske komponenty SLA sa radí:

- **Formát a periodicita reportovania** – V tejto časti sa definuje akým spôsobom, v akom formáte, ako detailne a ako často budú poskytované reporty, či už zo strany používateľa, alebo samotného zákazníka.
- **Platby a kompenzácie** – Zákazník musí uhrádzať platby v súlade s podmienkami stanovenými v tejto časti. Rovnako sa tu určujú kompenzácie od poskytovateľa v prípade porušenia SLA [7].

2.1.3 Atribúty služby

Medzi atribúty služby sa radí:

- **Identifikácia služby** – V tejto časti je jedinečne definovaná služba [6], [7].
- **Rozsah** – V tejto časti sa určuje doména alebo rozsah, v ktorej služba existuje. Taktiež sú tu spomenuté ciele úrovne služby (SLO) a kľúčové indikátory výkonu služby (KPI) [7].
- **Prostredie** – Táto časť popisuje prostredie, v ktorom služba funguje najefektívnejšie. Môže tu byť definované ako zmeny prostredia, prípadne konfiguračných parametrov, ovplyvňujú efektívnosť služby [7]. V prípade IaaS CC si ako jeden z atribútov prostredia vieme predstaviť kvalitu internetového pripojenia zákazníka. Ak by mal zákazník vysokú latenciu alebo nízku prenosovú rýchlosť pripojenia, môže to ovplyvniť jeho vnímanú kvalitu služby, aj keď u poskytovateľa nevznikli žiadne problémy. Preto je vhodné takéto atribúty uvádzať do kontraktu.
- **Výnimky z kvality služby** – V kontexte prevádzkovania služby môže nastať situácia, kedy poskytovateľ nie je schopný danú službu poskytovať. Toto môže zahŕňať napr. pravidelnú údržbu, výpadky v dodávkach energie a pod. Vopred známe situácie môžu byť uvedené v tejto časti zmluvy a v prípade takýchto výpadkov je poskytovateľ oslobodený od definovaných sankcií [7], [8].

2.1.4 SLA QoS metriky

- **Definícia metrík** – V tejto práci sa budeme zameriavať najmä na oblasti, ktoré sú definované v tejto časti SLA. Opisujú sa tu QoS indikátory pomocou metrík. Pre danú službu sú analyzované rôzne charakteristiky pomocou matematických metód, ktoré sa nazývajú aj ako metriky služby (*Service Metrics*) [7]. Z pohľadu tejto práce je vhodné zadefinovať také, ktoré majú opodstatnenie pre cloudovú službu IaaS. Tejto problematike sa venujeme bližšie v kapitole 2.2.

- **Metódy merania** – Použité metódy merania a mechanizmy výpočtov bývajú uvedené v tejto časti SLA [7].

2.1.5 Právne komponenty

- **Autorita** – Uvádza, na akú autoritu je možné sa obrátiť v prípade akýkoľvek sporov.
- **Postup vysporiadania –** Uvádza sa najmä pre prípad, že by došlo k bankrotu poskytovateľa služby.
- **Politika ukončenia kontraktu**
- **Utajenie –** Právne definície týkajúce sa ochrany citlivých údajov.
- **Kompenzácie za poškodenie**

2.2 SLA metriky pre IaaS Cloud Computing

Z pohľadu tejto práce nás zaujíma monitoring cloudovej služby IaaS. Za účelom určenia parametrov na monitorovanie potrebujeme pre túto cloudovú službu najprv zadefinovať užitočné metriky z pohľadu SLA. Vieme sa inšpirovať napríklad spoločnosťou Amazon, ktorá poskytuje mimo iných aj IaaS cloudové služby. Niekoľko zdrojov [6], [9] sa zhoduje, že medzi najdôležitejšie metriky pre túto cloudovú službu patria nižšie uvedené.

Parameter	Popis
Kapacita procesora (CPU)	Rýchlosť (a počet jadier) procesora pre VM
Veľkosť operačnej pamäte	Operačná pamäť RAM dostupná pre VM
Trvanie štartu	Doba kým je VM pripravená na použitie
Úložisko	Veľkosť úložiska pre krátkodobé alebo dlhodobé využitie
Škálovateľné nahor	Maximálny počet VMs pre jedného používateľa
Škálovateľné nadol	Minimálny počet VMs pre jedného používateľa
Čas škálovania nahor	Čas pre navýšenie určitého počtu VMs
Čas škálovania nadol	Čas pre zníženie určitého počtu VMs
Automatické škálovanie	Boolovská hodnota pre podporu auto škálovateľnej vlastnosti
Maximálny počet VMs	Maximálny počet VMs, ktoré môže zákazník spustiť
Dostupnosť	Zvyčajne percentuálna hodnota vyjadrujúca pomer času počas ktorého je služba dostupná
Doba odozvy	Trvanie prijatia a vykonania požiadavky

Tabuľka 2.1 SLA QoS metriky pre IaaS [6], [9]

2.3 Indikátory úrovne služby

Súčasťou takého dokumentu sú aj indikátory úrovne služby (SLI). Preto sa ďalej stručne povenujeme aj tejto problematike.

SLI definujeme ako kvantitatívne meranie istého aspektu úrovne služby. Zo samotného názvu vyplýva, že SLI indikuje tento aspekt. Výslednú hodnotu takéhoto merania považujeme za SLI [10].

Merania bývajú často agregované, čiže surové dáta sa zbierajú počas meracieho okna a následne sa premieňajú na pomer, priemer alebo percentil. V ideálnom prípade SLI priamo meria úroveň služby. V niektorých prípadoch to ale buď nie je možné, alebo získanie dát je náročné a nerentabilné. Do takejto kategórie vieme zaradiť meranie oneskorenia na strane klienta. Takéto oneskorenie je z pohľadu SLA oveľa významnejšia metrika, no v praxi je pre poskytovateľa meranie oneskorenia zvyčajne možné len na strane servera [10].

Veľmi dôležitým SLI je aj dostupnosť, inak aj pomer času, počas ktorého je služba dostupná voči celkovému času využívania služby. Pri každej službe je dôležité rátať aj s výpadkami, preto 100% dostupnosť nie je realistický predpoklad. V IT priemysle sa často garantujú dostupnosti na úrovni 99% a vyššie [10].

2.4 Ciele úrovne služby

Rovnako ako SLI je v kontexte SLA vhodné zadefinovať aj pojem cieľ úrovne služby (SLO). SLO predstavuje požadovanú hodnotu pre úroveň služby, ktorá je meraná pomocou SLI. To akým spôsobom zvoliť a na akú hodnotu nastaviť SLO je komplexný proces [10].

Určenie a zverejnenie SLO pre používateľov, im umožňuje nastaviť si očakávania, ako výkonná je daná služba. Vďaka takejto stratégii dokáže poskytovateľ zredukovať nepodložené sťažnosti, napríklad na pomalú rýchlosť služby. V prípade ak SLO nie sú explicitne zadefinované, používatelia si zvyknú vytvárať vlastné očakávania ohľadom výkonnosti služby, ktoré môžu byť zo strany poskytovateľa nerealizovateľné. Zo strany používateľa môžu vzniknúť dva protichodné fenomény na základe vyššie spomenutých skutočností. Na jednej strane môže dôjsť k prílišnému spoliehaniu sa na danú službu, čo môže mať negatívny následok v prípade výpadku. Na strane druhej môže používateľ nadobudnúť predstavu, že je služba málo spoľahlivá. Z reálnych udalostí je z pohľadu poskytovateľa horší fenomén prílišného spoliehania, preto v praxi funguje zaujímavé riešenie. Po nastavení SLO, napríklad dostupnosti, na určitú hodnotu (pre príklad

predpokladajme hodnotu 95% času) je aj v prípade 100% funkčnosti služba zámerne vypínaná, aby sa splnila predpísaná hodnota [10].

2.5 Zhrnutie problematiky SLA, SLI a SLO

Je náročné prevádzkovať službu bez porozumenia toho, ktoré atribúty najviac ovplyvňujú jej beh, ako ich merať a vyhodnocovať. Na základe intuície, skúseností a znalostí toho, čo používatelia od služby očakávajú, by mal poskytovateľ vedieť zostrojiť SLA s definovanými SLO, ktoré sa overujú pomocou SLI.

Inak povedané, poskytovateľ by mal stanoviť hodnoty, ktoré pre rôzne aspekty služby garantuje (SLO), vytvoriť merania týchto aspektov, ktorých výsledky (SLI) sa budú porovnávať s SLO. Na základe toho sa určí, či je splnená SLA [10].

V praktickej časti očakávame, že systém bude vykonávať merania niektorých metrík spomenutých v Tabuľka 2.1, najmä však dostupnosti. Z týchto meraní systém získa hodnotu SLI, ktorú bude prezentovať a aj porovnávať s prednastavenou hodnotou SLO. V prípade ak SLI → SLO bude systém informovať o možnom ohrození poskytovania služby v súlade s SLA. Ak SLI < SLO, bude systém prezentovať porušenie SLA.

3 DEFINOVANIE A PROBLEMATIKA METRÍK

V predchádzajúcej kapitole sme sa pokúsili definovať dôležité metriky z pohľadu SLA. Zákazníka však okrem toho môže zaujímať aj výkonnosť a využitie jeho prostriedkov, preto potrebujeme identifikovať metriky, ktoré by vedeli takéto informácie prezentovať. Okrem toho máme záujem identifikovať metriky prezentujúce zdravie, výkon a stav celého systému za účelom riadenia behu platformy. Preto potrebujeme najprv poznať z čoho sa systém skladá a následne pre jednotlivé súčasti identifikovať vhodné metriky.

3.1 Metriky z pohľadu zákazníka

Zákazník má, okrem garantovaných parametrov v SLA, motiváciu sledovať výkonnosť jemu pridelených zdrojov. My, ako poskytovateľ služby, mu chceme poskytnúť takéto informácie, ktoré však potrebujeme najprv identifikovať. V našom prípade sa budeme inšpirovať priemyselným riešením od spoločnosti Amazon, pozri [11]. Ich cloudová platforma (AWS) je celosvetovo využívaná a poskytuje zákazníkom širokú škálu cloudových služieb (CS).

Z pohľadu tejto práce nás zaujíma Amazon EC2 (*Amazon Elastic Compute Cloud*), čo predstavuje IaaS službu od spoločnosti Amazon. Poskytuje veľkú škálu virtuálnych inštancií s dostupnosťou rôznych typov CPU, úložiska, operačného systému a podobne [12]. Zákazníkovi umožňuje monitorovanie svojich inštancií pomocou riešenia s názvom *CloudWatch*.

Uvádzané priemyselné riešenie poskytuje niekoľko kategórií metrík, ktoré sú zákazníkovi k dispozícii. Väčšina týchto kategórií je v úzkom súvise s účtovacím systémom spoločnosti Amazon. Tieto kategórie nie sú z pohľadu riešenia tejto práce významné. Môžeme sa však inšpirovať kategóriou metrík VM inštancie, viď Tabuľka 3.1, kde sa nachádzajú najmä technické parametre. V prípade záujmu sú ostatné kategórie dostupné online v Amazon *CloudWatch* dokumentácii, pozri [11].

Môžeme konštatovať, že tabuľka nižšie nám poskytuje pohľad na to, aké metriky by mohol navrhovaný systém zbierať a prezentovať zákazníkovi navyše k tým, ktoré sme definovali so spojitosťou s SLA v kapitole 2.

Metriky inštancie		
Metrika	**Opis**	**Jednotka**
Využitie CPU	Percentuálne vyjadrenie priradených EC2 výpočtových jednotiek, ktoré inštancia využíva.	Percentá
Operácie čítania z disku	Dokončené operácie čítania zo všetkých diskov, ktoré má inštancia k dispozícii.	Počet
Operácie zápisu na disk	Dokončené operácie zapisovania na všetky disky, ktoré má inštancia k dispozícii	Počet
Objem čítania z disku	Množstvo bajtov prečítaných zo všetkých diskov, ktoré má inštancia k dispozícii.	Bajty
Objem zápisu na disk	Množstvo bajtov zapísaných na všetky disky, ktoré má inštancia k dispozícii.	Bajty
Objem prijatej prevádzky	Množstvo bajtov prijatých na každom z rozhraní inštancie.	Bajty
Objem odoslanej prevádzky	Množstvo bajtov odoslaných z každého rozhrania inštancie.	Bajty
Počet prijatých paketov	Počet paketov prijatých na každom z rozhraní inštancie.	Počet
Počet odoslaných paketov	Počet paketov odoslaných z každého rozhrania inštancie.	Počet

Tabuľka 3.1 *CloudWatch* metriky inštancie [11]

3.2 Metriky z pohľadu poskytovateľa

Z literatúry je pomerne náročné identifikovať konkrétne štandardy, procedúry a metriky pre monitorovanie CC ako celku. Pre začiatok si teda uvedieme vrstvy, ktorými vieme vymodelovať cloud. Keďže poskytovateľ má záujem poznať všetky aspekty jeho prevádzkovanej platformy, tieto vrstvy mu môžu poskytnúť pohľad, kde všade umiestniť monitorovacie stanice. Pre kompletnosť nižšie uvádzame všetkých sedem vrstiev, tak ako sú definované neziskovou organizáciou *Cloud Security Aliance* [3].

- **Objekty –** Do tejto vrstvy patria dátové centrá, v ktorých sú uložené výpočtové a sieťové prostriedky [3]. V tejto práci sa nebudeme bližšie zaoberať monitorovaniu tejto vrstvy, no v prípade komerčného poskytovania CS mimo akademického prostredie, by bolo vhodné rozšíriť monitorovací systém aj o túto časť.

- **Sieť –** V tejto vrstve vieme nájsť sieťové zariadenia a prepojenia, ako v rámci priľahlej fyzickej infraštruktúry, tak aj (ak je to možné) medzi cloudom a používateľom.
- **Hardvér –** Pre túto vrstvu sú typické fyzické komponenty, ako výpočtových tak aj sieťových zariadení.
- **Operačný systém –** Ako súčasť tejto vrstvy považujeme softvérové komponenty, ktoré tvoria operačný systém na fyzických hostiteľoch, prípadne na samotných virtuálnych inštanciách.
- **Sprostredkovací softvér (*middleware*) –** Je softvérová vrstva medzi operačným systémom a používateľskou aplikáciou typická pre SaaS a PaaS.
- **Aplikácia –** Rovnako typická pre SaaS a PaaS.
- **Používateľ –** Do tejto vrstvy zaraďujeme konečného používateľa CS a aplikácie bežiace mimo CC (napríklad webový prehliadač bežiaci na zariadení používateľa) [3].

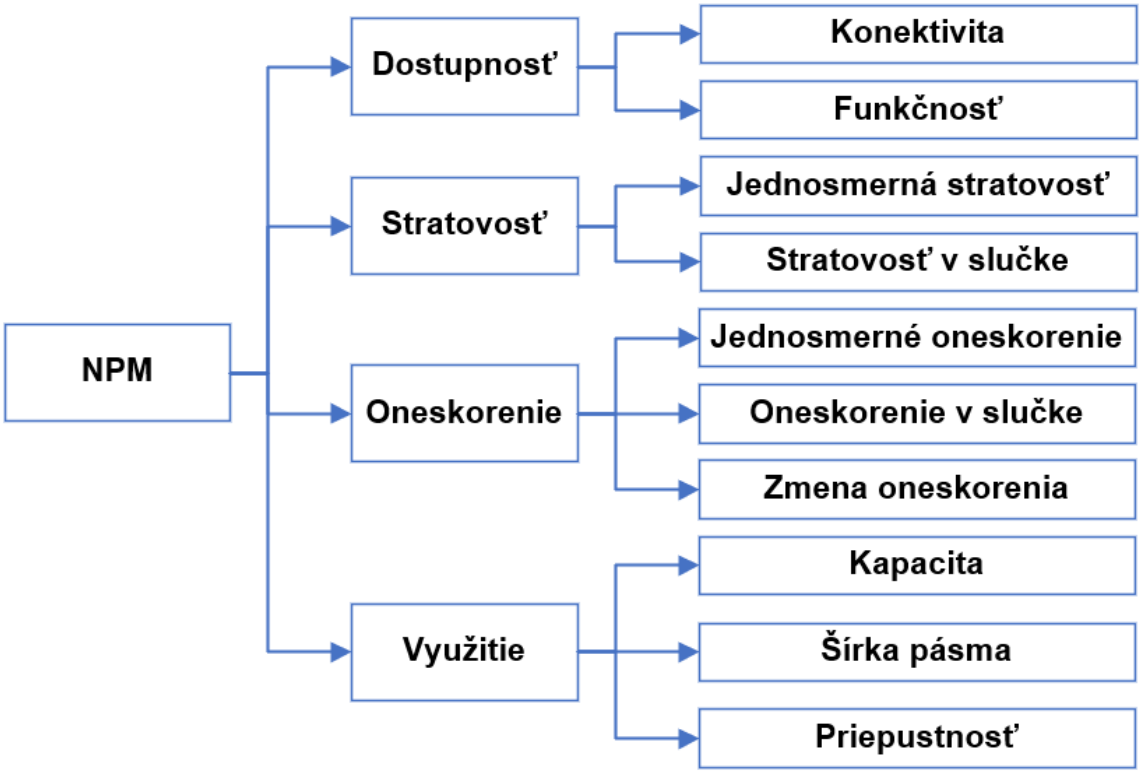
Z pohľadu poskytovateľa primárne IaaS služby považujeme za významné vrstvy objektu, siete, hardvéru a operačného systému. Preto sa ďalej pokúsime identifikovať významné metriky pre tieto vrstvy. Rovnako vieme tvrdiť, že dôležitou vrstvou je aj konečný používateľ, no praktické prevedenie takéhoto typu monitorovanie je ťažko uskutočniteľné.

3.2.1 Metriky sieťovej vrstvy

Sieťové výkonové metriky (NPM - *Network Performance Metrics*) zobrazené nižšie, zodpovedajú základným výkonovým metrikám vo vrstve sieťového riadenia. Dokážeme ich rozdeliť do štyroch typov nasledovne:

- **Dostupnosť –** Zahrňujeme sem funkčnosť fyzickej konektivity jednotlivých sieťových prvkov a sieťových zariadení.
- **Stratovosť –** Je v jednoduchosti pomer stratených k odoslaným paketom medzi odosielateľom a prijímateľom v určitom časovom intervale, zväčša vyjadrený v percentách. Definuje sa jednosmerné oneskorenie, ktoré vyjadruje tento pomer len smerom od odosielateľa k prijímateľovi a oneskorenie v slučke, ktoré zahŕňa aj cestu odpovede opačným smerom.
- **Oneskorenie –** Vyjadruje, aký veľký časový interval trvá paketu vykonať jedno alebo dvojsmernú cestu. Zväčša sa skladá z troch faktorov: jednosmerné oneskorenie, oneskorenie v slučke a zmena oneskorenia.

- **Využitie –** Je percentuálne vyjadrenie momentálneho vyťaženia linky v pomere ku maximálnej priepustnosti [13].



Obrázok 3.1 Sieťové výkonové metriky [13]

Pre cloud bývajú typické dátové centrá, preto nás zaujíma najmä problematika monitorovania ich infraštruktúry, pri ktorej býva bežnou praxou ideológia nadmerného odberu (ang. *oversubscription*). V jednoduchosti to znamená, že poskytovateľ zákazníkom poskytne v súčte väčšie množstvo zdrojov, ako je ich reálna maximálna kapacita. V praxi sa aplikuje najmä za účelom maximalizácie dostupných zdrojov a zvýšenia rentabilnosti. Vychádza z nízkej pravdepodobnosti výskytu udalosti, pri ktorej by všetci zákazníci využívali všetky svoje zdroje na sto percent. Z vyššie uvedeného vyplýva, že kľúčovým cieľom je schopnosť rýchlo identifikovať preťaženie a pokles výkonu. Pre tento účel nám najvýznamnejšie slúžia metriky oneskorenia a využitia CPU sieťových zariadení v dotknutej sieti [14].

3.2.2 Metriky hardvérovej vrstvy

Do tejto vrstvy zahŕňame fyzické zariadenia a ich komponenty, ktoré tvoria výpočtovú, prípadne sieťovú vrstvu nevyhnutnú pre beh CC platformy. Nižšie sa pokúsime identifikovať, ktoré metriky sú z pohľadu poskytovania služby a behu CC platformy významné.

Keďže koncept CC má svoje korene v *grid* sieťach, resp. v distribuovaných systémoch [2], platí, že dôležité metriky z pohľadu takýchto systémov sú podstatné aj v rámci CC. Kniha Google SRE [10] definuje štyri zlaté signály monitorovania, ktoré vyjadrujú najdôležitejšie faktory potrebné sledovať pri každom používateľsky orientovanom systéme. Tieto charakteristiky uvádzame nižšie:

- **Oneskorenie –** Čas potrebný pre dokončenie akcie. Sledovanie oneskorenia komponentov umožňuje určenie slabých miest, prípadne identifikovanie začínajúceho zlyhania.
- **Prevádzka –** Pod prevádzkou rozumieme zaneprázdnenie systému alebo systémového komponentu. Pri tejto charakteristike ide najmä o zachytenie záťaže alebo dopytu po systéme. Zjednodušene vieme tvrdiť, že ide o vyjadrenie toho, koľko práce momentálne systém vykonáva. Sledovaním prevádzky komponentov vieme určiť nedostatok, ale aj prebytok pracovnej kapacity pri danom komponente.
- **Chyby –** Sledovanie chýb je dôležité v kontexte znalosti celkového zdravia systému. Na základe dát o chybách vieme predpokladať slabiny systému a určiť, ako často jednotlivé komponenty zlyhávajú. V prípade zlyhania celého systému dokážu niektoré chyby indikovať, kde a prečo systém zlyhal.
- **Saturácia –** Meria aktuálne využitie daného zdroja. V tomto kontexte sa často využívajú pomery alebo percentá. Dáta ohľadom saturácie poskytujú informáciu o vyťažení zdrojov, ktoré služba alebo aplikácia potrebuje pre efektívne fungovanie [15]. Pri veľkej záťaži komponentov môže dôjsť k poklesu výkonu služby.

Vyššie uvedené signály vieme použiť ako smernicu, pričom potrebujeme zistiť, ako budú tieto metriky vyjadrené v hardvérovej vrstve systému. Momentálny technologický trend je abstrahovanie fyzických komponentov a nízkoúrovňového OS. Nepopierateľným faktom však je, že každá služba spolieha na základný hardvér a OS pri svojej funkčnosti. Pokiaľ chceme identifikovať, ktoré metriky je vhodné zbierať a vyhodnocovať, potrebujeme si najprv položiť fundamentálnu otázku. Ktoré individuálne komponenty tvoria základnú vrstvu pre beh platformy?

Vieme, že dôležitými súčasťami sú hardvérové komponenty serverov (CPU, pamäť, úložisko, sieťové zariadenia) a niektoré základné abstrakcie, ktoré poskytuje OS (procesy, deskriptory súborov). Pokiaľ sa na každý z týchto komponentov pozrieme v kontexte štyroch zlatých signálov, vieme identifikovať vhodné metriky pre monitorovanie.

V kontexte **CPU** sú vhodné nasledovné merania:

- **Oneskorenie –** Priemerné alebo maximálne oneskorenie v rámci CPU plánovača.
- **Prevádzka –** Celkové využitie CPU.
- **Chyby –** Špecifické chyby procesora, zlyhanie CPU.
- **Saturácia –** Dĺžka bežiaceho frontu.

V kontexte **pamäte** môžu jednotlivé signály vyzerať nasledovne:

- **Oneskorenie –** Pri pamäti je náročné nájsť vhodnú metódu merania.
- **Prevádzka –** Množstvo aktuálne využitej pamäte.
- **Chyby –** Chyby typu *Ouf of memory.*
- **Saturácia –** Využívanie SWAP.

Pre **úložisko** by sme mali brať do úvahy nasledovné merania:

- **Oneskorenie –** Priemerný čakací čas pre zápisy a čítanie.
- **Prevádzka –** I/O úrovne pre čítanie aj zápis.
- **Chyby –** Chyby súborového systému, chyby disku v */sys/devices*.
- **Saturácia –** Veľkosť I/O frontu.

Pre **sieťové zariadenia**:

- **Oneskorenie –** Dĺžka frontu sieťového ovládača.
- **Prevádzka –** Prichádzajúce a odchádzajúce bajty alebo pakety za sekundu.
- **Chyby –** Chyby sieťových zariadení, zahodené pakety.
- **Saturácia –** Zahodené pakety, opakovane vysielané segmenty.

Spolu s metrikami, ktoré reprezentujú stav fyzických zdrojov je vhodné zbierať aj metriky súvisiace s OS [15].

3.2.3 Metriky vrstvy operačného systému

V prípade CC platformy postavenej nad riešením *OpenStack* a poskytujúcej IaaS službu, považujeme za vhodné do tejto vrstvy zaradiť OS bežiace na fyzických uzloch, ale aj samotný cloudový OS *OpenStack*.

V kontexte vhodných metrík pre OS bežiace na fyzických zariadeniach je náročná ich presná identifikácia z literatúry. Vo všeobecnosti to záleží najmä od požiadaviek administrátorov. Pre začiatok sa môžeme inšpirovať metódou ***Využitie, Saturácia*** *a* ***Chyby*** (USE – *Utilization Saturation and Errors*), pomocou ktorej je možné analyzovať výkonnosť akéhokoľvek systému [16]. Touto metódou sme sa inšpirovali aj v predchádzajúcej kapitole,

pri hardvérových komponentoch. V rámci nej sa ako významné softvérové metriky uvádzajú:

- **Kapacita procesov** – Systém má obmedzený počet procesov aj vlákien. V tomto kontexte je vhodné sledovať ich momentálne využitie, počet čakajúcich procesov na spracovanie, prípadne monitorovať aj chyby alokácie.
- **Kapacita súborových deskriptorov** – Pri ktorých je rovnako vhodné sledovať ich momentálne využitie a prípadné chyby [16].

Okrem vyššie uvedených metrík považujeme zo softvérového pohľadu za dôležitú aj **dobu behu servera**, keďže tá dokáže indikovať, či sa systém nevypol neočakávanie bez vedomia administrátora. Tú vieme definovať ako množstvo času, počas ktorého je server schopný vykonávať požiadavky. Túto metriku vypočítame ako *Celkový čas – doba prestoja* [17].

Cloudovému OS *OpenStack* a jeho komponentom, ktoré by bolo vhodné monitorovať, sa venujeme podrobnejšie v kapitole 4.2.

3.3 Prevedenie meraní vybraných metrík

Je mimo rozsahu tejto práce venovať sa spôsobu merania všetkých metrík uvedených v doterajšom riešení. Preto nižšie definujeme podľa nás tie najdôležitejšie metriky, pričom uvádzame ich možné spôsoby merania a vyhodnotenia. Z doterajšieho kontextu je zrejmé, že najdôležitejšími komponentami sú CPU, pamäť, úložisko a sieťové prvky. Tu je potrebné ozrejmiť, že máme motiváciu zbierať informácie o týchto komponentoch z fyzickej aj virtuálnej vrstvy, pričom spôsob extrakcie sa v jednotlivých vrstvách môže líšiť. Pri fyzickom systéme sa spôsob získania dát potrebných pre výpočet metrík môže meniť v závislosti na type OS. V tejto práci o danej problematike pojednávame v kontexte systémov Linux/UNIX.

Z pohľadu funkčnosti monitorovacieho systému považujeme za dôležité, aby sme na virtuálne systémy nemuseli priamo pristupovať, preto potrebujeme vyššie spomenuté dáta získať iným spôsobom. Tu sme dospeli k záveru, že tieto informácie budeme získavať prostredníctvom komunikácie s hypervízorom. Pre tieto účely je vhodný program *virsh,* ktorý na komunikáciu využíva súpravu nástrojov *Libvirt* (virtualizačné API) napísanú v jazyku C. Viac informácií o programe a využitej súprave nástrojov je dostupných na oficiálnych manuálových stránkach *Libvirt,* viď [18].

Pri každom komponente sa pokúsime identifikovať, aké výsledky môžu predznamenávať problém, keďže neskôr budeme mať motiváciu informovať

administrátorov o takýchto udalostiach. Túto problematiku riešime len v kontexte fyzických systémov.

3.3.1 Výpočtový procesor

Pre CPU považujeme za najdôležitejšiu metriku jeho využitie, tj. množstvo času, počas ktorého vykonával činnosť, zväčša uvádzané v percentách.

Pri fyzickom systéme štatistiky o využívaní procesora ukladá jadro (*kernel*) do určeného súboru `/proc/stat`. Vysvetlenie formátu a významu jednotlivých hodnôt je k dispozícii v manuáli pre Linux, časť *proc(5)*, alebo online na [19]. Pre nás je dôležité, že dokážeme z týchto údajov identifikovať aký dlhý čas sa jadrá CPU venovali:

- Prostrediu používateľa.
- Jadru systému.
- Čakaniu na dokončenie I/O operácií.
- Čakaniu (procesor nevykonával žiadnu činnosť).
- Obsluhovaniu prerušení a
- ostatným činnostiam (viď [19]).

Na základe takto poskytnutých informácií dokážeme vypočítať, aký pomer času bolo jednotlivé jadro využívané. Pre výpočet využitia CPU, ako celku nám stačí aritmetický priemer využitia všetkých jeho jadier [20].

Pri virtuálnom systéme máme k dispozícii niekoľko spôsobov extrakcie informácií o využití virtuálneho CPU (vCPU) pomocou programu *virsh*. Pre viac informácií o konkrétnych možnostiach odkazujeme čitateľa na dokumentáciu tohto riešenia (viď [21]), odporúčame vyhľadať informácie o príkaze `domstats` a jeho argumentoch `cpu-total` a `vcpus`. Alternatívne je možné využiť aj príkaz `cpu-stats`. Pri ktorejkoľvek z uvedených možností však nevieme získať čas nečinnosti. Vieme však využiť čas práce vCPU.

Aké vysoké využitie môže spôsobovať problém? V tomto ohľade sa ako najkritickejšia časť prezentuje plánovač CPU, presnejšie teda úlohy s nízkou prioritou, ktoré sa pri vysokom využití nikdy nemusia dostať k výpočtovým prostriedkom. Vo všeobecnosti sa uvádza vrchný limit dlhodobého využitia medzi sedemdesiatimi a osemdesiatimi percentami. Pri procesoroch, ktoré sú sústavne zaťažované nad túto hranicu môže dochádzať k nepredvídateľnému správaniu. Pre systémových návrhárov sa zvykne odporúčať, aby pri návrhu nových systémov mierili na maximálne využitie procesora pod päťdesiat percent [22].

3.3.2 Operačná pamäť

Rovnako, ako pri CPU aj pri RAM pamäti vo fyzickom systéme považujeme za najdôležitejšiu metriku jej využitie. Proces extrakcie základných dát je taktiež veľmi podobný. Štatistiky o využívaní pamäte sa ukladajú do systémového súboru `/proc/meminfo`. Viac informácií je opäť dostupných v manuáli pre Linux, nižšie uvedieme len pár základných štatistík, ktoré nám takto systém ponúka.

- **Celková pamäť** – Celková použiteľná RAM.
- **Voľná pamäť** – Množstvo voľnej pamäte v systéme. Nezarátavame sem vyrovnávacie pamäte, ktoré môžu byť vyčistené.
- **Dostupná pamäť** – Voľná pamäť, do tejto hodnoty zarátavame aj vyrovnávacie pamäte.
- Využitie rôznych typov **vyrovnávacích pamätí**.

Z praktického hľadiska sú, zo všetkých ponúknutých štatistík, najdôležitejšie prvé dve spomenuté. Množstvo využitej pamäte vypočítame ako rozdiel celkovej a voľnej pamäte [19], [23].

Aj pre virtuálny systém budeme postupovať podobne ako v prípade CPU. *Virsh* nám ponúka príkaz `dommemstat`, ktorý vracia pamäťové štatistiky o zvolenej, bežiacej VM. Vďaka tomu dokážeme, mimo iného, získať údaje o celkovej, využitej aj voľnej pamäti. Všetky informácie, ktoré je možné pomocou tohto príkazu získať, ich jednotky a popis, je rovnako k dispozícii v oficiálnej dokumentácii [21].

Určiť, aké vysoké využitie operačnej pamäte môže spôsobiť problém je náročnejšie na identifikáciu, ako tomu bolo pri CPU. Odporúčania pre GitHub Enterprise server uvádzajú prvú formu hlásenia nižšej závažnosti (viac o závažnosti hlásení v 4.4.3) pri dlhodobom zaťažení nad úrovňou päťdesiatich percent. Pri vytrvalom zaťažení nad sedemdesiat percent odporúčajú hlásenie vyššej závažnosti [24]. Iné zdroje uvádzajú hlásenie problému pri využití nad sedemdesiatpäť alebo dokonca deväťdesiat percent [25], [26]. V rámci hlásení o využití pamäte je vhodné mať nastavené aj upozornenie pri využívaní vymieňania, tj. presunu dát z operačnej pamäte na disk za účelom uvoľnenia kapacity. Niektoré zdroje uvádzajú vhodnosť hlásenia až pri sedemdesiat percentom využívaní výmeny, no z našich praktický skúseností považujeme túto hranicu za príliš vysokú a odporúčame ju nastaviť okolo hodnoty dvadsiatich percent, keďže proces výmeny je náročný na systémové zdroje a opotrebováva SSD disky, ktoré majú obmedzený počet zapisovacích cyklov.

3.3.3 Úložisko

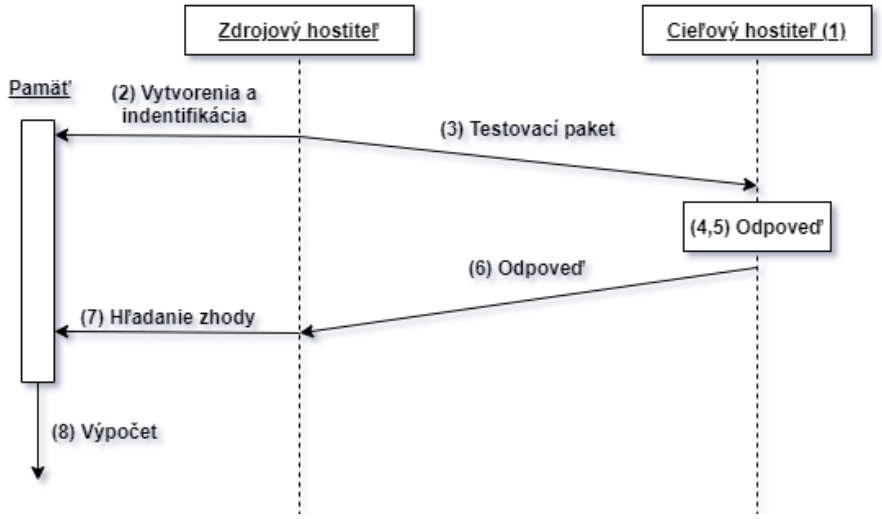
V linuxových systémoch slúži na zobrazenie informácií o využití disku jednotlivými súborovými systémami príkaz `df`. Tak, ako v predchádzajúcich prípadoch, jeho využitie je popísané v manuáli pre Linux (viď [27]). Zaujímavosťou je, že veľkosť úložiska nie je v tomto prípade jediným limitujúcim faktorom. Existuje ešte i-uzol, ktorý predstavuje dátovú štruktúru uchovávajúcu metadáta o adresároch a súboroch. Zjednodušene vieme tvrdiť, že predstavuje počet súborov, ktoré sú v danom súborovom systéme uložené. Ich množstvo je taktiež limitované, no v praxi málokedy dochádza k prekročeniu maxima tejto štruktúry pred dosiahnutím limitu úložnej kapacity. Príkaz `df` nám umožňuje extrahovať samotnú hodnotu *Available,* ktorá predstavuje zostávajúce voľné miesto v danom súborovom systéme [27]–[29].

Virsh nám umožňuje získať štatistiky aj o diskoch, ktoré sú priradené pre danú VM. Pre tento účel opäť slúži príkaz `domstats` s argumentom `block`. Pre detailnejší opis pozri [21].

Plnenie úložiska je pomalší proces, určite mu však netreba prikladať menšiu závažnosť. Nastavenie upozornenia je vhodné v prípade, že na disku ostáva menej ako 10% voľného miesta. Ak to monitorovací systém podporuje, je užitočné, aby na základe predchádzajúcich trendov dokázal odhadnúť kedy sa zariadeniu minie voľná kapacita a tento údaj reportovať administrátorovi [26].

3.3.4 Sieťové prvky

V kapitole 3.2.1 sme identifikovali najdôležitejšie metriky v rámci monitorovania sieťovej vrstvy, ktorými sú oneskorenie, vyťaženie liniek a využite CPU na zariadeniach v rámci siete datacentra. Ako prvému sa budeme venovať meraniu oneskorenia, následne prejdeme na problematiku sledovania CPU a ostatných komponentov na sieťových zariadeniach. ***Oneskorenie*** delíme na jednosmerné a v slučke. Druhé zo spomenutých považujeme za metriku poskytujúcu väčšiu informačnú hodnotu, preto sa budeme venovať tomuto typu. Poznáme dva spôsoby jeho merania, a to aktívny a pasívny. V jednoduchosti, pri aktívnom spôsobe generujeme novú prevádzku, ktorú následne monitorujeme. Pri pasívnom negenerujeme nič navyše, ale informácie získavame z už exitujúcej prevádzky. Na implementáciu je jednoduchší aktívny spôsob, no je potrebné dbať na to, aby objem generovanej prevádzky nemal významný vplyv na tú existujúcu. Oneskorenie v slučke definujeme ako rozdiel medzi časom odoslania prvého bitu žiadosti a časom prijatia posledného bitu odpovede na rovnakom hostiteľovi [30]. Postup merania nám graficky zobrazuje **Obrázok 3.2**.



Obrázok 3.2 Meranie oneskorenia v slučke [30]

Vo svete sieťových technológií je pre manažovanie a monitorovanie zariadení všeobecne rozšírený, a zaužívaný jednoduchý protokol manažérstva siete (SNMP). Tento protokol zatrieďuje podporované metriky do hierarchickej dátovej štruktúry zvanej databáza informácií manažérstva (MIB). Modul, ktorý je prítomný na riadenom elemente, inak nazývaný ako SNMP agent, odpovedá na dopyty riadiaceho prvku, SNMP manažéra. Odpoveďou na takýto dopyt sú hodnoty požadovaného objektu v MIB, korešpondujúce s jeho typom. Vďaka SNMP dokážeme zo sieťových zariadení extrahovať spomínané

využitie CPU, ale aj iné pasívne metriky, ako napríklad spomínané vyťaženie liniek alebo dĺžku frontu [31].

Keďže pri sieťových prvkoch sa jedná o fyzické zariadenia, nepotrebujeme riešiť v tejto časti problematiku virtuálneho prostredia. Spomenieme však, že by sme vedeli využiť SNMP alebo priame výstupy zo softvérových súčastí, ktoré emulujú takúto infraštruktúru vo virtuálnom prostredí. V kontexte *OpenStack* sa jedná o softvér s názvom *Neutron*. Pre viac informácií o monitorovaní virtuálnej sieťovej infraštruktúry odkazujeme čitateľa na [32].

V rámci upozorňovania platí pri CPU rovnaká filozofia, o akej sme pojednávali v kapitole 3.3.1. Pri oneskorení je problematické definovať presnú hranicu, od ktorej hovoríme o probléme. V jednoduchosti platí, že čím vyššie oneskorenie, tým horší výkon systému. Vhodné riešenie by mohlo byť sledovanie hodnoty oneskorenia isté časové okno, na základe údajov z takéhoto merania vyhodnotiť priemerné oneskorenie v systéme a pri prekročení dvoj alebo trojnásobku takejto hodnoty upozorniť na možný problém.

4 MONITORING PLATFORMY *OPENSTACK*

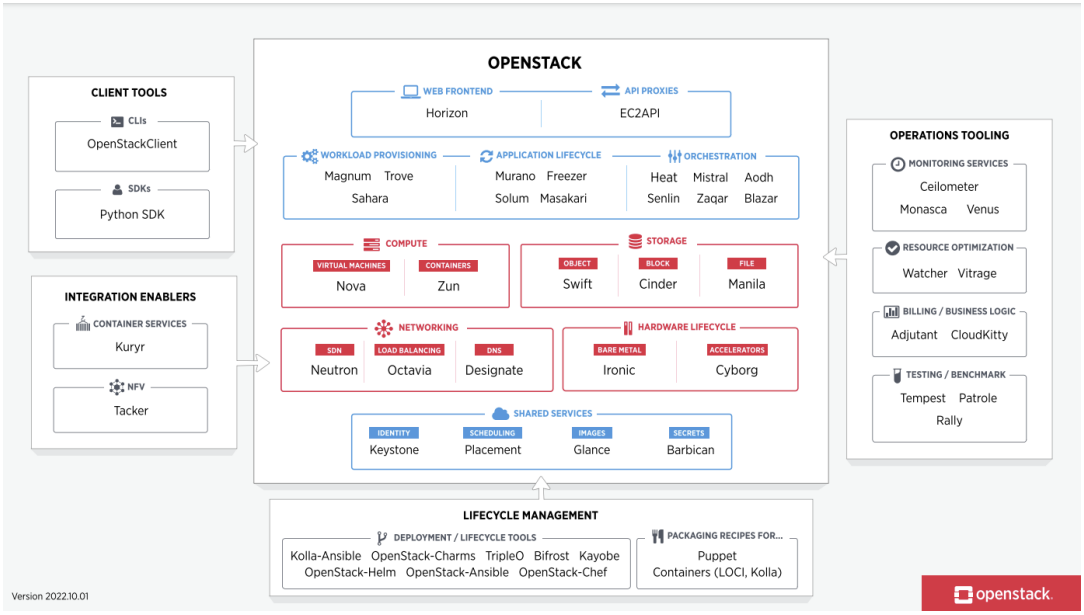
Pre správne pochopenie problematiky je nutné sa s ním na začiatok stručne oboznámiť.

4.1 *OpenStack*

OpenStack je cloudový operačný systém, ktorý dokáže ovládať veľké množstvo výpočtových, úložných a sieťových zdrojov. Na manažovanie týchto zdrojov využíva aplikačné programovacie rozhranie (API) s bežnými autentifikačnými mechanizmami. Okrem základného typu služby, ktorou je infraštruktúra, ponúka tento OS pomocou prídavných komponentov možnosti orchestrácie, prevencie zlyhania a riadenia služby pre zabezpečenie vysokej dostupnosti používateľských aplikácií [33].

Pôvodne boli súčasťou len dva základné moduly a to:

- **Nova** – Vyvinutý v NASA, poskytuje spôsob akým zabezpečiť vytvorenie, beh a zánik VM [34], [35].
- **Swift** – Vyvinutý spoločnosťou *Rackspace, pričom* riadi činnosť ukladania dát [34], [36].



Obrázok 4.1 OpenStack mapa [33]

So zapojením výrobcov serverov, sieťového vybavenia, úložísk a iných gigantov vo svete IT je *OpenStack* komunita veľmi aktívna, a celý systém sa rýchlym tempom vyvíja. Vo všeobecnosti oddeľuje používateľov od masívnych hardvérových zdrojov za sieťou, cez sieť. Je zodpovedný za interakciu s hypervízorom na fyzickom systéme [34]. Súčasťou

tohto systému je momentálne veľké množstvo komponentov, pričom každý z nich plní svoju úlohu v kontexte celého systému. Názvy najdôležitejších komponentov, s ich úlohami môžeme vidieť na *OpenStack* mape zobrazenej vyššie.

4.2 Monitorovanie v rámci OpenStack

Dáta nazbierané priamo z *OpenStack* sú z nášho pohľadu veľmi dôležité, keďže majú úzke prepojenie s SLA metrikami. O SLA sme už pojednávali v kapitole 2, tu sa pokúsime prepojiť danú problematiku s *OpenStack*.

Už vieme, že pri IaaS službe poskytuje CSP zákazníkovi výpočtové prostriedky, ktoré vieme rozdeliť do troch hlavných kategórií:

- **Výpočtové prostriedky –** Primárne do tejto kategórie zaraďujeme CPU a RAM. V *OpenStack* ma riadenie tejto kategórie za úlohu služba ***Nova.*** Práve táto služba nám umožní pohľad na SLA parametre, ako sú kapacita procesora, veľkosť operačnej pamäte a maximálny počet VM pre zákazníka.
- **Úložisko –** Riadením úložiska je v *OpenStack* poverená služba ***Cinder.*** Dokážeme z nej identifikovať ďalší SLA parameter, veľkosť prideleného úložiska.
- **Sieť –** Sieťové prvky spravuje služba ***Neutron.*** Síce nedokážeme priamo identifikovať žiadny z parametrov špecifikovaných pre SLA (viď Tabuľka 2.1), ktorý by sme vedeli extrahovať z tejto služby, no jeho funkčnosť je kritická, keďže pri jeho výpadku nie je zabezpečená žiadna komunikácia pre inštancie v cloude, čo sa môže zákazníkovi javiť ako výpadok dostupnosti jeho prostriedkov.

Okrem vyššie spomenutých kategórií identifikujeme ešte dva dôležité komponenty, a to ***Glance*** pre správu dostupných operačných systémov. Pre autentifikáciu a autorizáciu v rámci platformy slúži ***Keystone*** [37].

Čo je teda vhodné monitorovať z *OpenStack*? Z nášho pohľadu vidíme potrebu sledovať dostupnosť všetkých piatich komponentov uvedených vyššie. Dôvodom je, že pri výpadku ktoréhokoľvek z nich dochádza k narušeniu fungovania služby. V prípadnej SLA by dokonca mohli byť uvedené dostupnosti jednotlivých komponentov samostatne. Každý komponent sa navyše skladá zo súčastí, ktoré majú svoje osobité úlohy. Navrhujeme monitorovať aj tieto súčasti, ak je to možné. Prehľad najdôležitejších uvádzame nižšie, pri niektorých stručne popisujeme, aký vplyv na platformu môže mať ich výpadok.

- **Výpočtová služba (nova)**

- ***nova-api*** – Prijíma HTTP požiadavky, ktoré následne konvertuje a komunikuje ďalej s ostatnými súčasťami. Pri výpadku nebude môcť žiadny komponent komunikovať so službou *nova*.
- ***nova-scheduler*** – Rozhoduje, ktorý hostiteľ dostane pridelený ktorú inštanciu. Pri výpadku môže byť ovplyvnená tvorba nových inštancií.
- ***nova-conductor*** – Zaisťuje požiadavky, ktoré si vyžadujú koordináciu medzi ostatnými súčasťami alebo databázou. Výpadok môže znefunkčniť fungovanie komponentu ako takého.
- ***nova-compute*** – Riadi komunikáciu s hypervízorom a VM. Pri výpadku môže dôjsť k nefunkčnosti inštancií [38].

- **Úložná služba (cinder)**
  - ***cinder-volume*** – Spravuje dynamicky pripojiteľné blokové zariadenia. Pri jeho výpadku je znemožnená schopnosť vytvárať alebo upravovať blokové úložisko.
  - ***cidner-api*** – Má rovnakú úlohu ako nova-api, avšak pre úložný komponent.
  - ***cinder-scheduler*** – Rozhoduje, ktorý hostiteľ dostane pridelené jednotlivé zväzky [39].
- **Sieťová služba (neutron)**
  - ***neutron-server*** – Prijíma a smeruje API požiadavky k vhodným *OpenStack* sieťovým rozšíreniam na vykonanie akcie. Z toho vyplýva, že pri jeho nefunkčnosti dochádza k strate funkcionality sieťovej vrstvy cloudu.
  - ***OpenStack sieťové rozšírenia a agenti*** – Sú zodpovedné za pripájanie a odpájanie portov, vytváranie sietí a podsietí, poskytovanie IP adresácie a pod. Ich množstvo a typ závisí od poskytovateľa, a technológiách nasadených v danom prostredí. Medzi bežné sa zaradujú napríklad L3-agent a DHCP (dynamické prideľovanie IP adries) agent [40].

Samotný *OpenStack* ponúka pre monitorovanie vlastný projekt s názvom *Telemetry*. Jeho cieľom je spoľahlivo zbierať dáta o využití fyzických a virtuálnych zdrojov, rovnako tak vyššie spomenutých komponentov v rámci nasadeného cloudu. Taktiež ponúka možnosti analýzy a vyvolania akcií v prípade splnenia definovaných kritérií. Ako je uvádzané v oficiálnej dokumentácii, tento projekt sa skladá z ďalších menších projektov, ktorými sú:

- **Ceilometer** – zodpovedný za zber, normalizovanie a transformáciu dát vyprodukovaných súčasťami *OpenStack,* a
- **Aodh** – ktorý umožňuje vyvolať akciu na základe definovaných pravidiel [41].

Okrem vyššie spomenutých existuje aj riešenie s názvom *Monasca*, ktoré poskytuje monitorovanie ako službu, pričom dokáže integrovať aj s *OpenStack* [42].

4.3 Problematika viac-klientskej architektúry v kontexte monitoringu

Viac-klientska architektúra (ang. *Multi-tenant architecture*) je prirodzeným výsledkom snahy dosiahnuť pozitívny ekonomický efekt na báze úspory z rozsahu. V kontexte CC sa tento efekt dosahuje za využitia virtualizácie a následného zdieľania zdrojov [43]. Klient, v tomto kontexte nazývaný aj *tenant,* môže byť akákoľvek aplikácia, používateľ alebo skupina používateľov, ktorý požaduje vlastné bezpečné a exkluzívne virtuálne prostredie [44]. V prípade monitoringu CC platformy je veľkou výzvou identifikovať aký level prístupu k monitorovacím dátam garantovať jednotlivým klientom. Je potrebné myslieť na dva faktory, ktoré s touto architektúrou vznikajú v kontexte prístupu k nazbieraným dátam. Týmito faktormi sú:

- **Súbežnosť –** V niektorých prípadoch (najmä pri PaaS) má viacero klientov záujem pristupovať k rovnakým dátam. Pri takejto situácii musí poskytovateľ vhodným spôsobom poskytnúť viacerým klientom prístup k identickým informáciám z monitorovacieho systému [45]. Z pohľadu IaaS je z literatúry náročné presne identifikovať, kedy by mohla vzniknúť takáto situácia. Teoreticky by sme mohli predpokladať, že by viacero zákazníkov malo záujem o dáta z rovnakého fyzického uzla, na ktorom beží ich virtuálne prostredie.
- **Izolácia -** Tento faktor je pre monitorovací systém veľmi dôležitý, keďže tvorí veľkú časť dôvery zákazníka voči poskytovateľovi. Klienti nesmú mať prístup k informáciám zozbieraných z prostredí určených exkluzívne pre iných klientov [45]. V rámci tejto práce to znamená, že klient musí mať prístup k informáciám len o tých VM, ktoré mu aj reálne prislúchajú.

Pre kompletnosť a prepojenie pojmov CSP a *tenant* viď **Obrázok 4.2**. Ten poskytuje prehľad, ktorá strana by pri základných typoch CC služieb mala preberať zodpovednosť za jednotlivé súčasti. Medzi tieto súčasti zaraďujeme:

- **Infraštruktúru –** Hardvér a OS prislúchajúci k zariadeniam, na ktorých beží platforma.
- **OS –** Operačné systémy vo virtuálnom prostredí.
- **Aplikácie**
- **Dáta**

Typ cloudovej služby	Zodpovednosť			
	Infraštruktúra	OS	Aplikácia	Dáta
SaaS	CSP	CSP	CSP	Tenant
PaaS	CSP	CSP	Tenant	Tenant
IaaS	CSP	Tenant	Tenant	Tenant

Obrázok 4.2 Prehľad zodpovednosti pri CC službách [46]

Z obrázku vyššie si môžeme všimnúť, že *tenant* nesie veľkú mieru zodpovednosti pri IaaS službe, naopak pri služby typu SaaS nesie väčšiu zodpovednosť poskytovateľ. Vieme z toho vyvodiť, že pri monitorovacom systéme CC platformy potrebuje CSP monitorovať Tú časť, za ktorú je zodpovedný, ostatné je v kompetenciách jednotlivých klientov [46].

4.4 Prezentovanie nazbieraných dát pre zúčastnené strany

Ako sme už pojednávali v kapitole 4.3, do zúčastnených strán zaraďujeme poskytovateľa a používateľov (klientov). Spôsoby, akými im budú prezentované dáta sa podľa nás nemusia fundamentálne líšiť, rozdiel bude len v množstve a type prezentovaných dát. Na základe informácií od spoločnosti Google, vieme rozlíšiť dva hlavné spôsoby prezentovania informácií z monitorovacieho systému, ktorými sú reporty a forma prístupu k dátam z monitorovania [47].

4.4.1 Reporty

Nie je presne definované, aké rôzne typy a aké množstvo informácií musí byť v takýchto reportoch uvedené. Závisí to najmä na požiadavkách zákazníkov a schopnostiach CSP. V snahe identifikovať rôzne typy vhodných reportov sa môžeme inšpirovať technickou dokumentáciou cloudového systému W3500 od spoločnosti IBM [48]. Z pohľadu IaaS považujeme za významné reporty o aktivite strojov. Tieto slúžia na sledovanie fyzických a virtuálnych zdrojov, ktoré sú v rámci cloudu využívané. V rámci tejto kategórie môže existovať viacero typov hlásení:

- **Reporty o výpočtových uzloch –** Užitočné pre CSP. Môže obsahovať vzory využívania, priemery a potenciálne budúce trendy využitia CPU, a pamäte. Na základe takéhoto reportu by bolo možné identifikovať uzly, ktoré sú buď málo využívané, alebo preťažené. Pre tento typ môžu byť vhodné nasledovné informácie:
  - status,
  - rýchlosť CPU (MHz/GHz),
  - priemerné využitie CPU počas N-dní (%),
  - maximálne dosiahnuté využitie CPU počas N-dní (%),
  - kapacita pamäte,
  - priemerné využitie pamäte počas N-dní (%/GB),
  - maximálne dosiahnuté využitie pamäte počas N-dní (%/GB) a
  - počet inštancií bežiacich na výpočtovom uzle.
- **Reporty o využitých pridelených zdrojoch –** Užitočné pre zákazníkov. Tento typ reportu môže obsahovať rovnaké informácie ako typ vyššie, avšak ako celok sa nebude brať kapacita na výpočtovom uzle, ale kapacita pridelená danému zákazníkovi.
- **Využitie pridelených úložných zdrojov –** Užitočné pre zákazníkov. Pre CSP môže byť generovaný obdobný typ reportu, kde však za celok považujeme úložné zdroje na výpočtovom uzle alebo v celej platforme. Poskytuje informácie o využívaní úložiska. Medzi tieto informácie môžeme zaradiť:
  - pridelené úložisko (GB),
  - priemerné využitie úložiska počas N-dní (GB/%) a
  - maximálne dosiahnuté využitie úložiska počas N-dní (GB/%).
- **Využitie virtuálnych strojov –** Užitočné pre zákazníka. Z takéhoto typu reportu dokáže zákazník získať informácie o jeho VM. Nižšie uvádzame informácie, ktoré by mohol takýto report obsahovať:
  - identifikátor VM,
  - status,
  - informácie o pridelenom CPU:
    - počet jadier,
    - frekvencia.
  - Momentálne využitie CPU (%),
  - priemerné využitie CPU počas N-dní (%),
  - pridelená pamäť (GB),
  - pridelené úložisko (GB),

- momentálne využitie pamäte/úložiska (GB/%) a
- priemerné využitie pamäte/úložiska počas N-dní (GB/%).

- **Informácie o diskoch –** Užitočné pre CSP. Poskytuje CSP informácie o stave a využití jednotlivých diskov. Na základe takýchto podkladov môže byť schopný identifikovať ich vyťaženosť, prípadne zhoršenie výkonnosti, čo môže predznamenať blížiacu sa poruchu zariadenia. Môže obsahovať nasledovné informácie:
  - status,
  - veľkosť (GB),
  - priemerný počet vstupných/výstupných operácií čítania (*read IOPS*) za sekundu za posledných N-dní,
  - priemerný čas doby odozvy operácie čítania za posledných N-dní (ms),
  - priemerný čas doby odozvy operácie zápisu za posledných N-dní (ms),
  - výpočtový uzol v ktorom sa disk nachádza a
  - ak je to možné, tak zoznam inštancií, ktoré disk využívajú [48].

4.4.2 Prístup k dátam z monitorovania – vykresľovanie metrík

Pre lepšie pochopenie správania sa číselných ukazovateľov, je vhodné ich správnym spôsobom vykresliť. V praxi je zaužívané vizualizovať najdôležitejšie metriky na jednej obrazovke (*dashboard*), ktorá slúži ako efektívny vizualizačný nástroj pre sledovanie a zobrazovanie dát z rôznych zdrojov. Vďaka tomu poskytuje prehľad o stave a výkone sledovaného atribútu. Tomu, ako vykresľovať aké dáta sa venuje odbor dátovej analýzy. My tu spomenieme niekoľko zvyčajných vizualizačných techník:

- **Koláčový graf** – Rozdeľuje celok na sekcie. Poskytuje jednoduchý spôsob ako porovnať veľkosti jednotlivých komponentov v celku.
- **Čiarový graf** – Vhodne zobrazuje zmenu jednej alebo viacerých metrík v priebehu času. Vhodný spôsob pre rýchlu detekciu potenciálnych anomálií [49].
- **Miera (*gauge*) –** Možno si ju predstaviť ako ručičkový tachometer vo vozidle. Spolu s číselným ukazovateľom je vhodná na zobrazenie aktuálnej hodnoty metriky [50].

Vizuálna komunikácia by mala byť jednoduchá, aby umožnila cieľovej skupine rýchlo a jednoducho extrahovať z vizuálnych interpretácií požadované informácie. Preto sa uvádzajú odporúčané postupy pre vytvorenie užitočnej a jasnej vizualizácie.

- **Posúdenie cieľovej skupiny –** Pri tvorbe vizualizácií je potrebné zohľadniť niekoľko základných otázok. Najdôležitejšie sú, pre koho bude daná vizualizácia určená, aké sú jeho znalosti v problematike a akú informáciu sa z nej snaží získať. Ak je to možné, odporúčame vizualizáciu pred uvedením do produkcie najprv odprezentovať časti cieľovej skupiny pre získanie spätnej väzby.
- **Vybrať vhodné vizualizačné techniky –** Pre rôzne typy metrík je potrebné zohľadniť správny typ grafu. O tejto problematike sme už pojednávali v časti vyššie.
- **Udržať jednoduchosť –** Vo všeobecnosti je vhodné postupovať prístupom čo najmenšieho počtu vizualizovaných dát, poskytujúcich čo najväčšiu výpovednú hodnotu [49].

4.4.3 Prístup k dátam z monitorovania – signalizácia

Upozornenie na významné udalosti je jedna z najkritickejších a najdôležitejších súčastí monitorovacieho systému. Z tohto pohľadu je aj najnáročnejšia na presné definovanie, keďže každý systém je iný. To znamená, že jednotlivé ukazovatele budú mať rôznu váhu v rôznych systémoch. Existujú isté postupy, ktorými sa môžu administrátori monitorovacieho systému riadiť pre vytvorenie úspešnej stratégie varovných signálov. Spoločnosť Microsoft odporúča zohľadniť nižšie uvedené otázky, pre určenie, či je zistený symptóm vhodným kandidátom pre vytvorenie hlásenia. V rámci tejto stratégie sa však počíta s priemyselným prístupom, kedy je pre rôzne typy problémov (napr. hardvér/softvér) určený špecifický tím na jeho riešenie.

- **Je to urgentné? –** Vzniknutý problém nemusí hneď znamenať potrebu okamžitej odpovede od administrátora [51]. Je bežnou praxou zadefinovanie úrovní upozornení na základe ich závažnosti, ako uvádza **Tabuľka 4.1**.
- **Sú ovplyvnení zákazníci? –** Sú používatelia služby ovplyvnený vzniknutým problémom? Ak áno, je vhodné problém eskalovať na urgentný a informovať zákazníka, že problém je v štádiu riešenia.
- **Sú ovplyvnené závislé systémy? –** Môže vzniknutý problém vytvoriť ďalšie hlásenia zo závislých systémov [51]? Napríklad pri výpadku výpočtového uzla môžu vzniknúť následné hlásenia o nefunkčnosti, napríklad *OpenStack* komponentu Nova. Vo viac tímovom prostredí je pri takejto udalosti dôležité zabezpečiť, aby viacero tímov nepracovalo na rovnakom probléme.

Level	Názov úrovne	Popis
Závažnosť 0	Kritická	Výpadok kritického elementu služby, strata dostupnosti, výrazne znížená výkonnosť. Vyžaduje si okamžitú akciu.
Závažnosť 1	Chybová	Zníženie výkonnosti, strata dostupnosti časti služby. Vyžaduje si pozornosť, avšak nie okamžitú.
Závažnosť 2	Upozorňujúca	Problém, ktorý nezahŕňa zníženie výkonu alebo stratu dostupnosti, no má pri eskalovaní potenciál ich zapríčiniť.
Závažnosť 3	Informačná	Neindikuje problém, ale poskytuje zaujímavé informácie operátorovi.
Závažnosť 4	Úplná	Neindikuje problém, ale poskytuje detailné informácie operátorovi.

Tabuľka 4.1 Úrovne upozornení [52]

5 PRIESKUM VHODNÝCH NÁSTROJOV NA RIEŠENIE

Z doterajšieho riešenia práce je zrejmé, že nástroj, ktorý hľadáme by mal byť schopný zbierať metriky z fyzických serverov, sieťových zariadení aj hypervízorov. Taktiež by mal vedieť získať informácie priamo z prostredia *OpenStack*. Mal by ponúkať vyhodnocovanie a vizuálne prezentovanie nazbieraných dát, minimálne administrátorom platformy. Dôležitosť prikladáme aj možnosti vytvoriť pravidlá, na základe ktorých budú administrátori upozorňovaní v prípade prekročenia stanovených hodnôt. Výhodou by bola aj podpora viac-klientskej architektúry, kedy by sa používatelia vedeli dostať k monitorovacím dátam z ich virtuálnych prostredí.

Na trhu existuje veľké množstvo nástrojov určených, či už na generické monitorovanie systémov, alebo špecializované monitorovanie cloudu. Z týchto systémov je väčšina spoplatnených, my sa však sústredíme na tie s otvorenou licenciou. Nižšie uvedieme stručný prehľad niektorých z nich, pričom ďalej sa budeme primárne venovať porovnaniu riešenia *Zabbix*, a riešenia *Prometheus*.

5.1 Všeobecný prehľad nástrojov

5.1.1 Nagios

Vo svete monitorovacích riešení s otvorenou licenciou je často spomínaná služba *Nagios*. Vývojári tohto riešenia ponúkajú v otvorenej licencii *Nagios Core*, čo môžeme považovať za jadro služby, pri ktorom máme k dispozícii nástroj pre monitorovanie infraštruktúry, upozorňovanie administrátorov a množstvo bezplatných dodatkov vytvorených vývojármi, a komunitou. Vďaka nim je možné monitorovať aj služby bežiace v *OpenStack*. *Nagios* je však už starší softvér, čo sa prejavuje na jeho používateľskom rozhraní. V bezplatnej verzii nie je viac-klientska architektúra natívne podporovaná. V zhrnutí, s týmto riešením by sme boli schopní monitorovať stav fyzickej, sieťovej aj virtuálnej infraštruktúry, rovnako ako služby v rámci *OpenStack*. Taktiež vizualizovanie a upozorňovanie je natívne podporované [3], [37], [53].

5.1.2 Graphite

Graphite je monitorovací nástroj s otvorenou licenciou, ktorý sa vyznačuje ukladaním dát vo forme časových radov. Funguje formou centrálneho bodu, na ktorý dokážu používatelia zaslať metriky v definovanom tvare, nazbierané pomocou skriptov alebo iných riešení tretích strán, najčastejšie však *collectd*. Toto riešenie poskytuje celkovú viditeľnosť nad infraštruktúrou, podporuje vizualizáciu aj upozorňovanie administrátorov.

Problematické by mohlo byť získavanie údajov priamo z *OpenStack*. Pre vizualizovanie sa dá prepojiť s nástrojom *Grafana*, ktorá v istej forme podporuje viac-klientsku architektúru. Toto riešenie je do veľkej miery prispôsobivé, no jeho správa si vyžaduje hĺbkovú znalosť nástroja [54], [55].

5.1.3 Collectd

Tento démon, pozri aj [56], dokáže zbierať systémové aj aplikačné metriky a poskytuje mechanizmy na ich ukladanie, prípadne ich sprístupňuje cez sieť. Jeho prednosťami sú rýchlosť, keďže bol napísaný v jazyku C a optimalizovaný pre výkonnosť, a veľká dostupnosť rôznych rozšírení. Tento nástroj je neustále vyvíjaný a dobre zdokumentovaný. Má však aj svoje limitácie. Samostatne nedokáže zozbierané dáta vizualizovať, keďže nemá grafické používateľské rozhranie. Schopnosť upozorňovania je síce podporovaná, no výrazne limitovaná len na jednoduché overovanie prednastavených hodnôt [56]. Veľké plus vidíme v možnosti prepojení s riešeniami ako *Graphite* alebo *Prometheus*, keďže *Collecd* poskytuje rozšírenia, ktoré prevedú nazbierané dáta do požadovaného formátu a následne ich sprístupní pre zozbieranie.

5.1.4 Grafana

Grafana je interaktívna dátovo vizualizačná platforma. Aj keď sama o sebe nedokáže monitorovať, v spojení s monitorovacími riešeniami poskytuje silný nástroj pre vizualizáciu aj upozorňovanie. Dokáže pracovať s dátami z rôznych zdrojov ako *InfluxDB*, *Prometheus*, *Graphite* a pod. Je ponúkaná v troch variantoch:

- **S otvorenou licenciou** – Je bezplatná a ponúka celú základnú funkcionalitu.
- **Cloudové riešenie** – K dispozícii sú rôzne plány, pričom pri bezplatnom je zahrnutých desaťtisíc časových radov metrík v *Prometheus* formáte, päťdesiat GB úložiska pre logy atď. Na oficiálnej stránke sa uvádza ako najjednoduchšie riešenie pre začiatočníkov.
- **Pre firmy** – Platené riešenie s plnou podporou a funkcionalitou navyše, ako napríklad možnosť zasielania reportov, bezpečné oddelenie klientov a pod [57].

Za zmienku ešte stojí silné prepojenie riešenia *Prometheus* pre zber dát a *Grafana* pre ich vizualizáciu.

5.2 *Zabbix*

Zabbix sa prezentuje ako distribuované monitorovacie riešenie s otvorenou licenciou. Dokáže monitorovať veľkú sadu parametrov, od sieťových zariadení, cez fyzické servery, VM, aplikácie, služby a pod. Má vstavanú podporu pre SNMP, čo z neho robí výborný nástroj pre monitorovanie sieťovej infraštruktúry. Zbieranie dát je vykonávané buď priamo serverom, proxy serverom alebo pomocou agentov na koncových zariadeniach. *Zabbix* má dobrú podporu pre nastavovanie upozornení. Pre rôzne typy metrík je možné nastaviť rozdielne vyhodnocovacie pravidlá pre zaslanie upozornenia. Vstavané vizualizačné nástroje umožňujú jednoducho vizualizovať nazbierané dáta, ktoré sú ukladané v internej databáze. Komunita doposiaľ vytvorila veľké množstvo šablón pre rôzne typy monitorovania, napr. šablóny pre monitorovanie Cisco zariadení cez SNMP.

Zabbix architektúra sa skladá s nasledujúcich komponentov:

- **Server** – Centrálny komponent, kde sú uložené všetky konfigurácie a dáta.
- **Databázové úložisko** – *Zabbix* všetko ukladá do databázového úložiska.
- **Webové rozhranie** – Rozhranie umožňujúce užívateľom interakciu so systémom. Zväčša býva nasadené na rovnakom zariadení ako centrálny server.
- **Proxy** – Dokáže zbierať monitorovacie dáta namiesto centrálneho servera. Nie je povinnou súčasťou, ale vhodnou pri distribuovanom nasadení.
- **Agent** – Slúžia na monitorovanie lokálnych zdrojov na koncových zariadeniach [58].

Pre dosiahnutie viac-klientskej architektúry sa odporúča prepojenie s riešením *Grafana* [59].

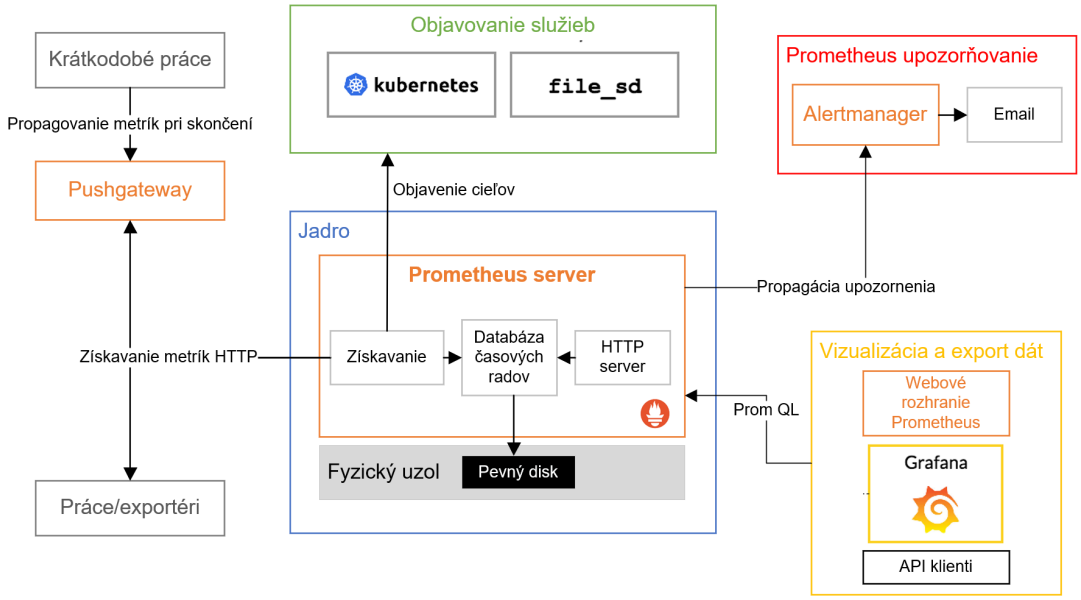
Pre monitorovanie fyzickej infraštruktúry sa *Zabbix* ukazuje ako najvhodnejšie riešenie. Avšak pre monitorovanie služieb *OpenStack*, ako aj VM pomocou komunikácie s hypervízorom by bolo nutné preveriť funkčnosť skriptov vytvorených komunitou pre tieto účely, prípade vytvoriť vlastné.

5.3 *Prometheus*

Za názvom *Prometheus* sa skrýva súprava nástrojov pre systémové monitorovanie a upozorňovanie. Je súčasťou *Cloud Native Computing Foundation*, ktorá združuje projekty s otvorenou licenciou natívne pre cloud. Zozbierané dáta ukladá ako časové rady, čo znamená, že každá metrika je uložená s časovou pečiatkou a voliteľnými pármi vo formáte kľúč-hodnota, nazývaných aj ako štítky. Medzi prednosti tohto nástroja sa radí práve spôsob

ukladania nazbieraných metrík, ako aj vlastný dopytovací jazyk *PromQL*, ktorý umožňuje efektívne využiť maticový spôsob ukladania dát. Za výhodu považujeme aj proces získavania dát, kde sa využíva *pull* model v kombinácii s hypertextovým prenosovým protokolom (HTTP). *Prometheus* bol navrhnutý najmä pre spoľahlivosť, keďže každý serverový komponent funguje samostatne a nie je závislý od žiadnych externých služieb. Pre jeho beh nie je potrebné vytvárať veľkú infraštruktúru. Zber dát sa vykonáva pomocou tzv. exportérov, ktoré môžu byť nasadené buď na koncovej, alebo inej stanici, v závislosti na type a spôsobe získavania dát [60]. Architektúru riešenia zobrazuje **Obrázok 5.1**.

5.4 Zhrnutie



Obrázok 5.1 Prometheus architektúra [60]

Pre praktické nasadenie sme vybrali riešenie postavené na súprave nástrojov *Prometheus* a vizualizačnej platforme *Grafana*. K výberu nás motivovala progresívnosť týchto riešení a ich silná vzájomná integrácia. Rovnako sme boli inšpirovaní prácou [34], kde sa autorom podarilo úspešne monitorovať stav *OpenStack* platformy aj hardvérových komponentov na fyzických uzloch. Avšak autori sa nevenovali problematike monitorovania VM vo vnútri cloudu, ani sieťových zariadení. Rovnako nekládli veľký dôraz na vizualizáciu a využili prednastavené šablóny. Pri našom riešení bude snahou vylepšiť monitorovací systém aj o tieto aspekty.

6 NÁVRH TECHNICKÉHO RIEŠENIA

V tejto kapitole sa budeme venovať návrhu a nasadeniu samotného monitorovacieho systému. Najprv stanovíme požiadavky na riešenie, následne sa pokúsime identifikovať hardvérové požiadavky pre takýto systém a vyberieme technológie potrebné pre realizáciu. Následne odprezentujeme navrhnutú architektúru a opíšeme jej nasadenie.

6.1 Požiadavky na riešenie

Ako sme pojednávali v kapitole 1.2, existuje niekoľko motivácií pre využívanie monitorovacieho systému. Hlavným cieľom je navrhnúť a nasadiť monitorovací systém pre CC platformu postavenú na riešení *OpenStack*. Pre vlastníka sú z nášho pohľadu, dôležité nasledovné procesy.

Kapacita a plánovanie zdrojov, a **kapacita a manažovanie zdrojov,** keďže bude prevádzkovaná fyzická infraštruktúra, ktorej zdravie, hladký beh a dostatok výpočtových zdrojov sú kritické z pohľadu funkčnosti celého systému.

Manažovaniu dátového centra nemusíme prikladať až takú významnosť, keďže veľkosť a zložitosť infraštruktúry nebude v dohľadnej budúcnosti dosahovať rozmery priemyselného dátového centra. Bude však vhodné myslieť na výkonnosť monitorovacieho systému v prípade škálovania infraštruktúry v budúcnosti.

Manažmentu SLA prikladáme vyšší stupeň dôležitosti, keďže je predpoklad poskytovania CC služieb ďalším zákazníkom.

Fakturovanie nie je predmetom tejto práce.

Zisťovanie problémov je kľúčovým dôvodom prečo prevádzkovať monitorovací systém, preto by naše riešenie malo byť prehľadné. Zároveň by malo umožniť rýchle detegovanie problému, ako aj upozorniť administrátorov, že došlo k problému.

Pri **manažovaní výkonu** by mal systém vedieť identifikovať potenciálne zníženie výkonu na jednotlivých uzloch a informovať o tom minimálne administrátorov.

V rámci **bezpečnosti** nepredpokladáme sledovanie zákazníckych dát pre garantovanie ich bezpečnosti.

Od riešenia teda požadujeme aby:

1. **Dokázalo monitorovať dostupnosť a výkonnosť** všetkých hardvérových prvkov, ktoré slúžia pre beh platformy. Medzi tieto prvky zaraďujeme sieťové a výpočtové zariadenia, tj. servery. V rámci výkonnosti sme identifikovali tri najkritickejšie komponenty, ktorými sú CPU, RAM a úložisko. V prípade sieťových zariadení sem môžeme zaradiť aj rozhrania. Nasadený systém musí byť schopný zbierať metriky obsahujúce také informácie, aby z nich bolo možné určiť využitie spomenutých komponentov s identifikovaním kritických medzí systému. Navyše, k hardvérovým zdrojom je vhodné, aby riešenie sledovalo množstvo využitých prostriedkov platformy *OpenStack*. Medzi takéto prostriedky radíme vCPU, RAM a úložisko z pohľadu platformy.
2. **Bolo schopné monitorovať stav poskytovaných služieb**. V rámci zadania práce sa venujeme službe IaaS, ktorá sa na cieľovej platforme, pre ktorú systém navrhujeme, skladá zo šiestich samostatných služieb: *nova*, *cinder*, *heat*, *keystone*, *neutron* a *glance*. V prípadnej SLA by stálo za zváženie uviesť tieto služby samostatne, keďže každá spravuje istý aspekt celku tak, ako je uvedené v kapitole 4.2. V tomto kontexte od systému požadujeme, aby vedel monitorovať, ukladať a vyhodnocovať ich dostupnosť v zadanom časovom okne, napr. jedného dňa, a v rámci sledovania SLA parametrov zbieral informácie o prostrediach klientov. Za takého prostredie považujeme *OpenStack* projekty. V rámci nich musí systém získavať dáta o veľkosti dostupného/využitého množstva pamäte a úložiska, využitom/maximálnom počte inštancií, a jadier procesora. Okrem toho, je potrebné monitorovať využitie CPU a RAM jednotlivými inštanciami. Vhodnými môžu byť aj ostatné metriky uvedené v Tabuľka 2.1.
3. **Nazbierané dáta vhodne vizualizovalo**, aby pri prípadnom vzniku problému bolo pre administrátora jednoduchšie identifikovať jeho pôvod.
4. **Spoľahlivo a včas upozornilo relevantné osoby** v prípade vzniku problému, napríklad administrátorov systému. Poskytovateľ vo všeobecnosti nemá záujem upozorňovať klientov na udalosti v ich prostrediach, keďže sa tým môže pripraviť o prípadný zisk. Môže však ponúkať takúto službu navyše.

6.2 Výber technológií a hardvérové požiadavky

V tejto kapitole opíšeme vybrané technológie *Prometheus* a *Grafana*, spolu s potrebnými súčasťami, pričom sa pokúsime definovať ich hardvérové požiadavky.

6.2.1 Prometheus server

Ako jadro celého riešenia nasadíme *Prometheus* server. Ten bude zodpovedný za zber, ukladanie a poskytovanie metrík ďalším komponentom. Skladá sa z troch základných prvkov, ako ukazuje **Obrázok 5.1**. Súčasť pre získavanie dát pristupuje na HTTP koncové body a získava z nich metriky. Následne ich posúva úložnému komponentu, ktorý sa stará o ich efektívne uloženie. Posledným komponentom je HTTP server, ktorý prijíma dotazy vo formáte *PromQL* a poskytuje požadované dáta. Tento server rovnako funguje ako webové používateľské rozhranie, v ktorom je možná základná vizualizácia a textový prehľad nazbieraných metrík, spolu so statusom nakonfigurovaných koncových bodov.

Určiť potrebné systémové zdroje pre plynulý beh, nie je jednoduchá úloha. Závisí to totiž najmä od pravidelnosti zberu a veľkosti metrík, rovnako aj od množstva a zložitosti dotazov na *Prometheus* server. Pre prostredie *Prometheus* + *Grafana* sa pre úvodné nasadenie zvykne odporúčať: aspoň dve jadrá procesora, štyri GB operačnej pamäte a dvadsať GB úložiska. Po nasadení je možné približne vypočítať množstvo potrebnej RAM pre beh *Prometheus* servera, kalkulačka s vysvetlením a opisom je dostupná online na [61].

6.2.2 Nástroje pre zber metrík

Riešenie systém zberu metrík je komplexný problém, na ktorý neexistuje jediné riešenie, a preto aj náš návrh bude využívať viaceré komponenty. Hardvérové požiadavky nižšie uvedených nástrojov nie je možné presne určiť z dostupných zdrojov. Platí však, že sú to veľmi jednoduché programy, ktoré nevykonávajú výpočtovo náročnú činnosť. Nemali by mať preto veľký vplyv na celkovú výkonnosť hostiteľského systému.

Z pohľadu požiadaviek na riešenie vieme využité nástroje zaradiť do oblastí monitorovania dostupnosti a výkonnosti, a monitorovania stavu poskytovaných služieb. O popise týchto požiadaviek sme už pojednávali v kapitole 6.1.

6.2.2.1 Dostupnosť a výkonnosť

V kontexte zberu metrík ***z fyzických uzlov*** vieme využiť exportér pre linuxové systémy s názvom *Node exporter* (komponent systému *Prometheus*). Ten je potrebné nainštalovať na každú koncovú stanicu, z ktorej máme záujem extrahovať informácie. Tento program bude následne získavať metriky, buď na základe systémových volaní, alebo prístupom k systémovým súborom, do ktorých jadro OS ukladá štatistiky používania. Takéto získavanie sa uskutočňuje za pomoci zberačov, pričom je ich k dispozícii viac ako päťdesiat, každý pre rôzne typy systémových štatistík. Ich kompletný zoznam je k dispozícii online, v dokumentácii k programu, viď [62]. Po zbere ich exportér prevedie do špecifického

formátu pre *Prometheus* a zverejní pomocou HTTP koncového bodu pre *Prometheus* server na čítanie.

V rámci ***sieťových zariadení*** potrebujeme na zber dát využiť protokol SNMP. *Prometheus* nemá natívnu podporu tohto protokolu, preto musíme nasadiť špecializovaný exportér. Ten nasadíme za účelom zberu dát cez SNMP a následného prevedenia do zrozumiteľného formátu, keďže SNMP využíva hierarchickú, zatiaľ čo *Prometheus* N-maticovú štruktúru prezentovania metrík. Tento exportér nesie názov *snmp_exporter,* a je jedným z oficiálne podporovaných spolu s vyššie uvedeným *Node exporter*. Skladá sa z dvoch hlavných komponentov: *snmp_exporter* a *generator*. Druhý spomenutý slúži na vytvorenie konfiguračného súboru, zatiaľ čo prvý je hlavným komponentom zodpovedným za zber, prevedenie a prezentáciu metrík. Odporúčané nasadenie je na jednom alebo viacerých uzloch, pričom si tento exportér môžeme predstaviť ako proxy pre *Prometheus* server. Fungovaniu tohto exportéra sa ešte budeme v práci bližšie venovať. Pre viac informácií viď [63].

6.2.2.2 Stav poskytovaných služieb

Pre ***zber informácií o službách a zdrojoch*** v rámci platformy *OpenStack* môžeme využiť exportér s názvom *openstack-exporter*. Toto komunitné riešenie, napísané v jazyku *Golang* dokáže získať metriky z bežiaceho *OpenStack* cloudu a zverejniť ich pre *Prometheus* server. Dáta zbiera pomocou API koncových bodov, ktoré poskytuje *OpenStack*. Presné umiestnenie tohto exportéra nie je definované. Pre jeho funkcionalitu však potrebuje mať konektivitu na API body *OpenStack* a rovnako s *Prometheus* serverom. Zoznam všetkých informácií, ktoré vie zbierať je pomerne rozsiahly, pričom je dostupný v dokumentácii, viď [64]. Okrem vyššie spomenutého existuje aj exportér vydaný pod spoločnosťou *Cannonical*. Je však horšie zdokumentovaný a v čase prieskumu sa javil zastaraný, pre čitateľa však uvádzame odkaz na *github* repozitár, pozri [65].

V záujme splnenia požiadaviek potrebujeme ešte získavať ***informácie o inštanciách v cloude***. Nami zvolený spôsob, je extrakcia týchto dát z hypervízorov na výpočtových uzloch. Pre tento účel nám môže poslúžiť riešenie s názvom *collectd*. Tento démon dokáže zo systému, na ktorom je nasadený, zbierať obrovské množstvo dát za pomoci vstavaných dodatkov. Nás v tomto kontexte zaujímajú dodatky *virt* a *write_prometheus*. Prvý z uvedených využíva pre zber štatistík o virtuálnych hostiteľoch bežiacich v systéme *libvirt* API, rovnako ako *virsh,* o ktorom sme pojednávali v kapitole 3.3. Druhý spomenutý dodatok sa stará o prevod metrík do vhodného formátu. Pre zaručenie funkčnosti musí byť tento démon nainštalovaný na všetkých výpočtových uzloch v cloude.

6.2.3 Vizualizácia a upozorňovanie

Aj keď *Prometheus* umožňuje obmedzenú formu vizualizovania zozbieraných dát, v samotnej dokumentácii sa pre tieto účely odporúča využiť nástroj *Grafana*. Tá nám umožní vytvoriť monitorovacie obrazovky, v ktorých budeme schopní nakonfigurovať panely zobrazujúce najdôležitejšie aspekty systému. Každý panel bude získavať potrebné informácie s využitím *PromQL* dotazov na *Prometheus* server. Je na zvážení administrátorov, či tento nástroj nasadiť na samostatný, alebo rovnaký server ako *Prometheus*. Veľké plus vidíme v možnosti integrácie s ďalšími systémami, ktoré môžu byť v budúcnosti pridané do tohto riešenia, čím sa vytvorí jeden centrálny prezentačný uzol. Ďalším benefitom je možnosť konfigurácie upozornení priamo v grafickom rozhraní *Grafana*, čo využijeme v ďalšom riešení práce.

V oficiálnej dokumentácii sú minimálne hardvérové požiadavky definované ako dvesto päťdesiatpäť MB operačnej pamäte a jedno jadro CPU. V prípade využitia funkcionalít, ako je napríklad upozorňovanie, sa odporúča navýšiť počet jadier CPU [66].

6.3 Architektúra riešenia

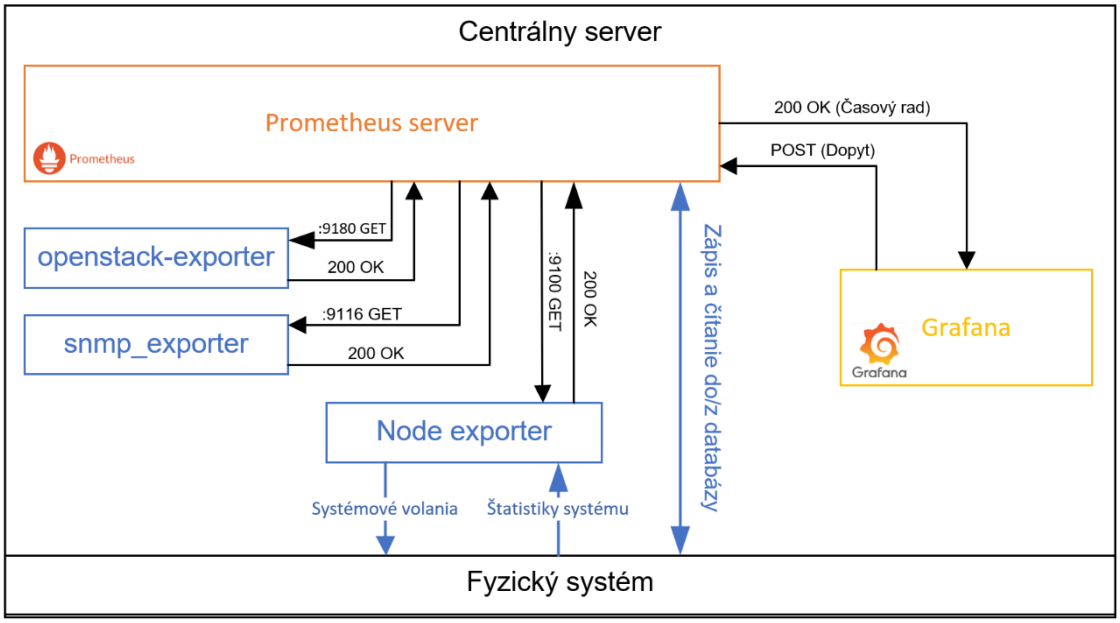
Za účelom realizácie projektu sme použili fyzický server, určený pre virtualizáciu, s operačným systémom *VmWare ESXI 6.7 Update 3*. V tomto prostredí môžeme nasadzovať VM podľa potreby. Pred samotným návrhom architektúry bolo potrebné určiť, či bude snahou oddeliť jednotlivé komponenty na samostatné VM, alebo zvolíme cestu centralizovaného riešenia, kedy bude existovať jeden centrálny bod. Aj keď centralizované riešenie prináša jediný bod zlyhania, rozhodli práve pre tento spôsob, a to pre jednoduchšiu administráciu do budúcna. Následne musíme brať do úvahy aj fyzickú infraštruktúru cloudu, keďže na výpočtové uzly máme záujem nasadiť dva typy exportérov a sieťové zariadenia budeme monitorovať s využitím SNMP.

6.3.1 Centrálny server

Centrálny server bude hlavným prvkom celého systému. Pre lepšiu názornosť uvádzame najprv grafickú vizualizáciu jeho komponentov a ich vzájomnú komunikáciu, pozri **Obrázok 6.1**. Základným komponentom je tu *Prometheus* server, ktorý komunikuje s exportérmi a zbiera z nich metriky. Táto komunikácie je znázornená čiernymi šípkami, pričom sa využíva HTTP protokol. Komunikáciu začína server zaslaním HTTP GET na localhost a špecifický port exportéra. Cesta k súboru s metrikami je v základnom nastavení pri každom exportéri `/metrics`. Porty nasadených exportérov sú uvedené v **Tabuľka 6.1**. Pokiaľ všetko správne funguje, exportér odošle odpoveď 200 OK, spolu s metrikami

a *Prometheus* server ich uloží do databázy. Ako sme už spomínali, na vizualizáciu budeme využívať nástroj *Grafana*, ktorý taktiež nasadíme na centrálny server. Ten bude komunikovať priamo s *Prometheus* serverom, za využitia *PromQL* dopytov, na ktoré server odpovedá požadovaným časovým radom.

Nižšie uvádzame obrázok architektúry a zoznam komponentov, ktoré budeme nasadzovať na centrálny server, s popisoch ich úloh a portom, na ktorých prijímajú spojenia.

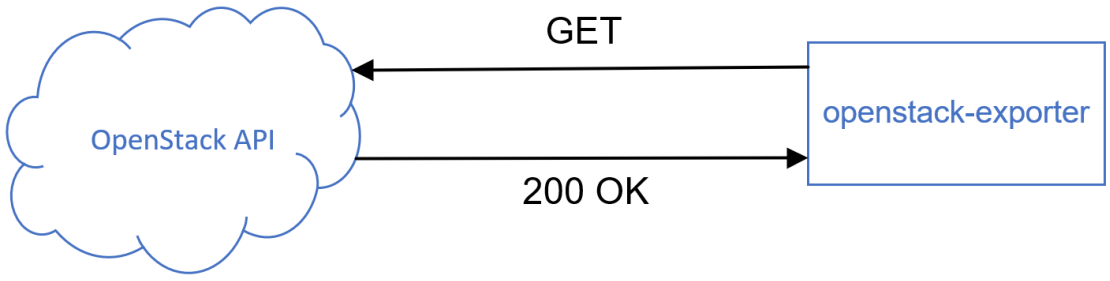


Obrázok 6.1 Architektúra centrálneho servera

Komponent	Port	Úloha
Prometheus server	9090	Zbieranie, ukladanie a poskytovanie metrík, poskytuje webový server pre užívateľskú interakciu.
Grafana	3000	Vizualizácia a upozorňovanie, poskytuje webový server pre užívateľskú interakciu.
openstack-exporter	9180	Zber metrík z *OpenStack* API
snmp_exporter	9116	Zber metrík zo sieťových zariadení za pomoci SNMP, prevedenie to správneho formátu
Node exporter	9100	Zber systémových informácií z centrálneho servera

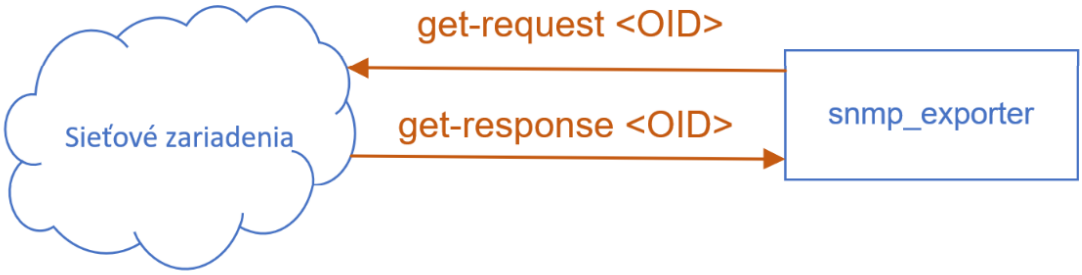
Tabuľka 6.1 Zoznam komponentov na centrálnom serveri

Openstack-exporter a ***snmp_exporter*** budú síce nasadené na centrálnom serveri, no metriky budú zbierať z externého prostredia cloudovej platformy. Preto považujeme za vhodné ozrejmiť fungovanie týchto komponentov. Najprv sa budeme venovať prvému spomenutému, pričom jeho komunikáciu s API koncovými bodmi reprezentuje **Obrázok 6.2**. Prvým krokom je autentifikácia voči *OpenStack* API, pričom všetky konfiguračné údaje ako meno, heslo a adresa autentifikačného bodu sa uvádzajú v konfiguračnom súbore *clouds.yaml* v špecifickom formáte *os-client-config,* pozri [64]. Program následne začne zasielať HTTP GET požiadavky na všetky dostupné koncové body, a tým získa štatistiky priamo z OpenStack. Koncové body nie je nutné jednotlivo konfigurovať, program ich sám objaví.



Obrázok 6.2 Komunikácia openstack-exporter s OpenStack API

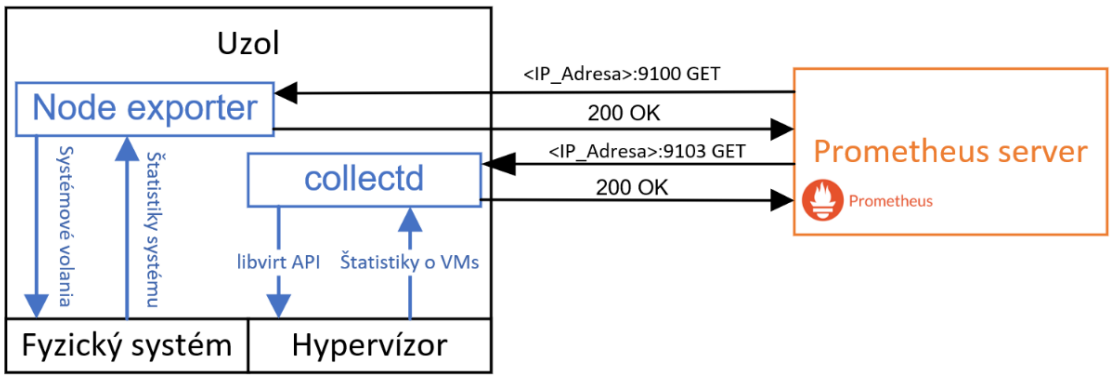
Snmp_exporter na získavanie metrík využíva SNMP. Túto komunikáciu znázorňuje **Obrázok 6.3**. Všetky OID, o ktoré máme záujem, musia byť uvedené v konfiguračnom súbore. Tento súbor sa generuje pomocou generátora. Túto problematiku budeme bližšie rozoberať pri samotnom nasadení.



Obrázok 6.3 Komunikácia snmp_exporter so sieťovými zariadeniami

6.3.2 Ostatné komponenty

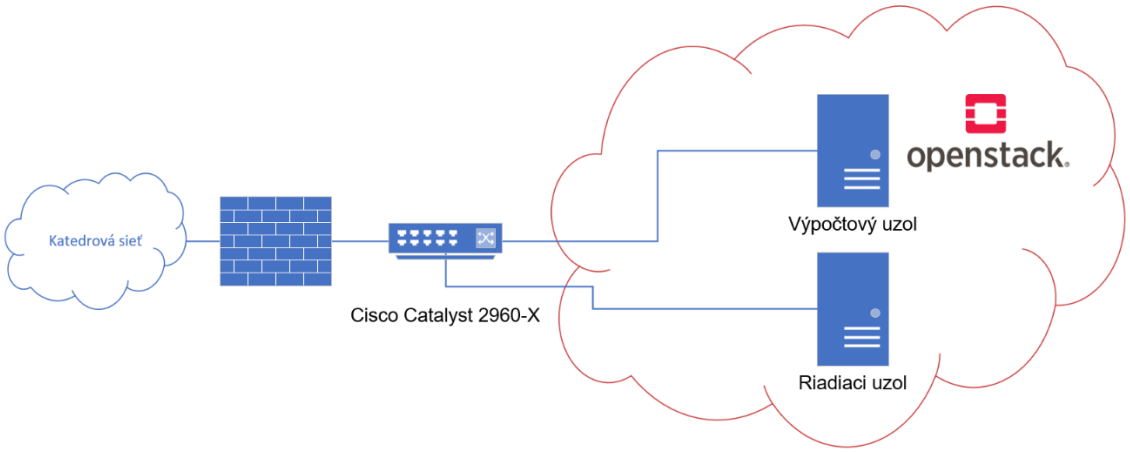
Okrem doteraz uvedených súčastí definujeme v rámci architektúry aj monitorovacích agentov na fyzických uzloch cloudovej platformy. Ako sme už spomínali v kapitole 6.2.2, na výpočtových uzloch sa jedná o exportér *Node exporter* a démona *collectd* s dodatkami *virt,* a *write_prometheus*. Na riadiacom uzle nie je *collectd* potrebný. Komunikácia s týmito agentami je totožná, ako pri exportéroch na centrálnom serveri a prezentuje ju **Obrázok 6.4**. Objavuje sa nám tu nový exportér, ktorý prijíma spojenia na porte 9103.



Obrázok 6.4 Architektúra ostatných komponentov

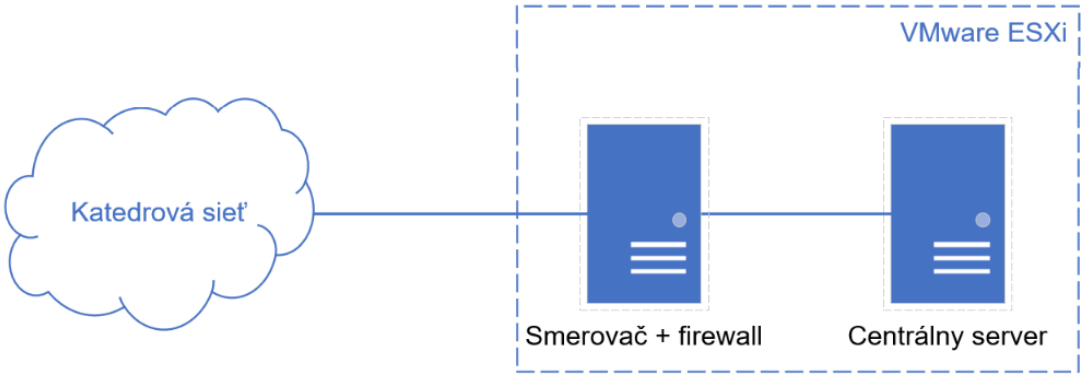
6.3.3 Celková architektúra

V tejto časti prepojíme systémovú architektúru s fyzickým svetom. Konzultáciou s administrátormi cloudovej platformy sme zistili zloženie infraštruktúry. Skladá sa z dvoch uzlov, pričom jeden je riadiaci a druhý výpočtový. Konektivitu medzi nimi a do katedrovej siete kde bol test zabezpečuje prepínač Cisco Catalyst 2960-X, konkrétne na portoch Gi1/0/25 až Gi1/0/30. Práve tieto tri zariadenia považujeme za fyzickú infraštruktúru, ktorú potrebujeme monitorovať. Znázorňuje ju **Obrázok 6.5**.



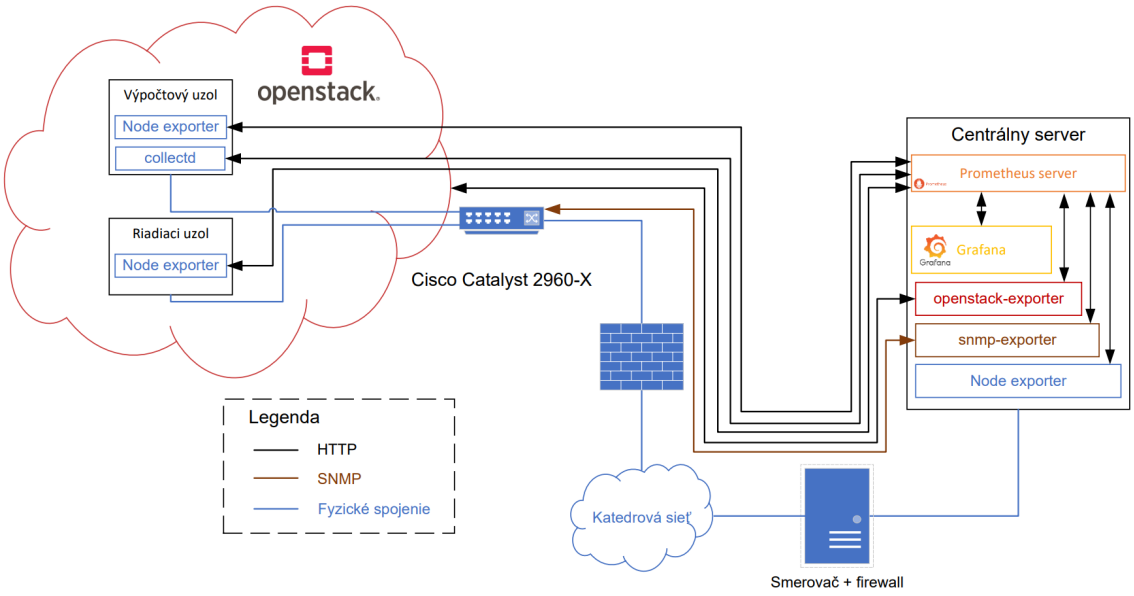
Obrázok 6.5 Topológia monitorovaného cloudu

Okrem platformy potrebujeme ozrejmiť aj prostredie monitorovacieho servera. To sa skladá z dvoch VM, pričom jeden slúži ako samotný server a druhý ako smerovač a základný firewall. Topológiu tohto prostredia ilustruje **Obrázok 6.6**.



Obrázok 6.6 Prostredie monitorovacieho servera

Spojením všetkých prvkov dokopy získavame pohľad na celkovú architektúru riešenia, pozri **Obrázok 6.7**.



Obrázok 6.7 Celková architektúra

6.4 Nasadenie

Na základe architektúry, ktorú sme definovali v predchádzajúcej kapitole, je potrebné toto monitorovacie riešenie nasadiť a uviesť do funkčného stavu. V tejto kapitole opíšeme proces nasadzovania jednotlivých komponentov a overíme ich funkčnosť. Komponenty budeme nasadzovať na tri rôzne stroje. Pre začiatok je vhodné uviesť ich špecifikácie, pričom pri uzloch v cloude máme záujem len o OS, naopak pri centrálnom serveri uvedieme aj pridelené výpočtové zdroje.

- **Riadiaci uzol:**
  - **OS:** CentOS Linux 7 (Core)
  - **Kernel:** Linux 3.10.0-1127.18.2.el7.x86_64
- **Výpočtový uzol:**
  - **OS:** CentOS Linux 7 (Core)
  - **Kernel:** Linux 3.10.0-1127.18.2.el7.x86_64
- **Centrálny server:**
  - **OS:** Ubuntu 20.04.5 LTS
  - **Kernel:** Linux 5.4.0-146-generic
  - **CPU:** 2 jadrá

- **RAM:** 4 GB
- **Úložisko:** 300 GB

6.4.1 Prometheus server

Keďže tento prvok považujeme za jadro celého riešenia, bude aj prvým nasadeným komponentom. Inštalovať ho budeme na centrálny server. Spôsob nasadenia, prvotnej konfigurácie aj používania je dobre popísaný v oficiálnej dokumentácii, viď [60]. Nižšie stručne popisujeme náš postup.

Prvým krokom je stiahnutie príslušného archívu. V čase nasadenia bola najnovšia verzia označená ako 2.37.1. Pre získanie odkazu na stiahnutie archívu *Prometheus* viď dokumentáciu, konkrétne časť *Introduction->First steps*. Pre stiahnutie odporúčame nástroj *wget*. Po stiahnutí je potrebné archív extrahovať, najvhodnejšie nástrojom *tar*. Je na zvážení administrátora, do akého priečinku chce uložiť výslednú zložku. My sme využili adresár `/opt`, ktorý by sa mal, podľa dokumentácie k linuxovému súborovému systému, využívať na inštaláciu všetkých aplikácií tretích strán. Do tohto adresára budeme mať záujem neskôr nainštalovať všetky komponenty. Výslednú zložku pre *Prometheus* sme nazvali `prometheus_server`, a okrem iného obsahuje dva najdôležitejšie súbory: binárny pre spustenie s názvom `prometheus` a konfiguračný `prometheus.yml`. V konfiguračnom súbore sa nachádza pár prednastavených riadkov, ktoré umožňujú spustiť server hneď po rozbalení, pričom monitoruje samého seba. Z dôvodu obmedzeného rozsahu práce toto nebudeme demonštrovať a v prípade záujmu odkazujeme čitateľa na oficiálnu dokumentáciu, kde je prvá interakcia s riešením detailne opísaná. Ako posledný krok odporúčame nastavenie aplikácie ako službu. Táto konfigurácia umožní, aby sa aplikácia automaticky spúšťala pri štarte systému a uľahčí interakciu s ňou. Keďže máme záujem takto nakonfigurovať všetky komponenty systému, uvádzame v kapitole 6.4.7 generický postup aj s odkazom na nami využité súbory. V našom nasadení využívame tri argumenty programu.

- **storage.tsdb.path** - Pre špecifikovanie cesty k priečinku, kde bude umiestnená databáza časových radov.
- **config.file** - Pre špecifikovanie cesty ku konfiguračnému súboru.
- **storage.tsdb.retention.time** - Pre nastavenie dĺžky uchovávania dát, ktorá je v základnom nastavení pätnásť dní, my požadujeme hodnotu tridsiatich dní.

6.4.2 Node exporter

Tento exportér budeme nasadzovať na tri stanice. Výhodou je, že tento proces sa pri daných OS nelíši. Ďalším benefitom je detailne spracovaný návod na inštaláciu v oficiálnej *Prometheus* dokumentácii, viď [60], časť *guides->monitoring linux host metrics with the node exporter*.

Postup je veľmi podobný nasadeniu predchádzajúceho servera. Ako prvé je potrebné stiahnutie archívu, jeho extrakcia a spustenie. Prirodzene dodržiavame na každom zariadení inštalačný adresár `/opt`. Opätovne odporúčame nakonfigurovať tento program ako službu. V prípade exportérov je jednoduché overiť ich správnu funkčnosť, a to pomocou klienta jednotného vyhľadávača prostriedku (URL). Keďže exportéri zverejňujú metriky na HTTP koncovom bode, dokážeme pomocou spomenutého klienta, nazývaného aj *curl*, pristúpiť k tomuto obsahu. Ako bolo spomenuté v 6.3.1, *node exporter* funguje na porte 9100, pričom cesta k obsahu je `/metrics`. Tieto informácie využijeme pri zadávaní príkazu: `#curl http://localhost:9100/metrics`. Ako výstup očakávame pomerne rozsiahly výpis metrík aj s ich hodnotami.

6.4.3 Snmp exporter

V kontexte nasadenie a konfigurácie exportérov považujeme tento za najnáročnejší na uvedenie do prevádzky, a to najmä pre proces jeho konfigurácie pomocou generátora konfiguračného súboru. Podobne, ako v predošlých prípadoch, je v rámci prvého kroku potrebné stiahnuť binárne súbory programu, v našom prípade na centrálny server. Za týmto účelom odkazujeme čitateľa na oficiálny repozitár, kde sa nachádza aj dokumentácia s opisom inštalácie a vygenerovania konfigurácie, pozri [63]. V čase nasadenia bola najnovšia dostupná verzia označená pod číslom 0.21.0. Do adresára `/opt` sme z *github* repozitára naklonovali dva nové projekty, pod názvami `snmp_exporter`, ktorý obsahuje základný konfiguračný a spustiteľný binárny súbor a `snmp_exporter_generator`, obsahujúci celý projekt aj s generátorom konfigurácie. Keďže vygenerovanie konfiguračného súboru je predpokladom pre správnu funkčnosť, opíšeme ako prvé tento proces.

V rámci stiahnutého projektu v zložke `snmp_exporter_generator` je najdôležitejším priečinkom `generator`. Ten obsahuje dva prvky, konfiguráciu generátora pod názvom `generator.yml` a priečinok `mibs`, do ktorého je potrebné stiahnuť všetky MIB súbory, buď priamo obsahujúce nami požadované OID, alebo od nich závisiace iné, nami využívané MIB. Práve identifikácia požadovaných OID a následne korešpondujúcich MIB súborov je časovo náročná úloha. Pre vyhľadávanie, prácu a sťahovanie MIB súborov

odporúčame nasledovné zdroje: [67], [68]. Problematiku nami požadovaných metrík, ich OID a k nim korešpondujúcich MIB súborov sa pokúšame zhrnúť nižšie.

Požadované metriky:

- **Doba behu systému**
  - Názov: *sysUpTime*
  - OID: 1.3.6.1.2.1.1
  - MIB: SNMPv2-MIB
- **Informácie o rozhraniach**, ako napr. názov, rýchlosť, aktuálna prevádzka, status a pod.
  - Názvy: *interfaces* a *ifXTable*
  - OID*:* 1.3.6.1.2.1.2, 1.3.6.1.2.1.31.1.1
  - MIB: IF-MIB
- **Štatistiky procesora**
  - Názov: *cpmCPU*
  - OID: 1.3.6.1.4.1.9.9.109.1.1
  - MIB: CISCO-PROCESS-MIB
- **Informácie o pamäti**
  - Názvy: *ciscoMemoryPoolUsed* a *ciscoMemoryPoolFree*
  - OID: 1.3.6.1.4.1.9.9.48.1.1.1.5 a 1.3.6.1.4.1.9.9.48.1.1.1.6
  - MIB: CISCO-MEMORY-POOL-MIB

Zoznam všetkých nami stiahnutých MIB súborov je k dispozícii v Príloha A: Zoznam MIB súborov pre generátor *snmp_exporter*, pričom niektoré súbory si generátor stiahne sám pri stavbe programu a niektoré je potrebné stiahnuť manuálne.

Keďže generátor je dynamicky viazaný na stiahnuté MIB súbory, je potrebné ho skompilovať a zostaviť pri snahe o vygenerovanie konfigurácie, tak ako je uvedené aj v dokumentácii na *github* [63]. Pred tým bolo potrebné upraviť konfiguráciu generátora, na základe ktorého vytvorí konfiguračný súbor pre exportér.

Po stiahnutí máme konfiguračný súbor generátora `generator.yml`, čiastočne prednastavený. V našom prípade sme potrebovali doplniť využívanie SNMP verzie dva, definovanie *community string* pre autentifikovanie a pridanie definície pre zber štatistík o procesore a pamäti. Tieto zmeny sme pre jednoduchosť vykonávali v konfigurácii modulu s názvom *if_mib*, ktorý je prednastavený na zbieranie informácií o rozhraniach. Jeho finálny obsah je k dispozícii v **Chyba! Nenašiel sa žiaden zdroj odkazov.**, súbor konfigurácie.

Keďže tento nástroj bol naprogramovaný v jazyku Go, je nutné tento jazyk doinštalovať do OS. Pre *Ubuntu* je dostupný inštalačný balíček, nanešťastie v čase nasadenia obsahoval Go verziu 1.13, pričom pre kompiláciu generátora je potrebná verzia minimálne 1.19. Preto sme museli inštaláciu vykonať ručne, pomocou inštrukcií v oficiálnej Go dokumentácii, dostupnej na [69].

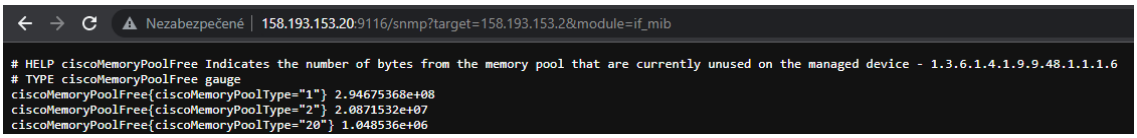
V rámci posledného kroku sme generátor zostavili s využitím príkazu `#make generator mibs` a konfiguráciu pre exportér vytvorili príkazom `#make generate`, čím sa nám vygeneroval súbor `snmp.yml`, ktorý sme presunuli do priečinka `snmp_exporter`, kde sa nachádza spustiteľný súbor exportéra. Tak ako v predchádzajúcich prípadoch sme vytvorili aplikáciu ako službu a spustili program.

Pre overenie poskytuje exportér používateľské rozhranie dostupné na porte 9116. Tu môžeme overiť zber metrík zadaním IP adresy cieľového zariadenia do položky *Target* a stlačením tlačidla *Submit*. **Obrázok 6.8** zobrazuje našu snahu získať metriky z prepínača 2960X, viď **Obrázok 6.5**.



Obrázok 6.8 Rozhranie snmp exporter

V našom prípade sme si overili funkčnosť, čo dokazuje výstup zobrazený na obrázku nižšie.



Obrázok 6.9 Začiatok výpisu metrík získaných pomocou *snmp exporter*

6.4.4 Openstack exporter

V dokumentácii sú uvedené až štyri rôzne spôsoby nasadenia tohto exportéra. My sme pre jednoduchosť zvolili inštaláciu pomocou dostupného balíčka *snap*. Inštaláciu je

potrebné vykonať ako privilegovaný používateľ *root* a jej postup zahŕňa jediný príkaz uvedený nižšie, čím nainštalujeme exportér z kanála *edge*, a získame prístup k najnovším funkcionalitám.

```
#snap install –channel edge golang-openstack-exporter
```

Ovládanie aplikácie je dobre popísané v dokumentácii, preto odkazujeme čitateľa na [64].

V rámci nasadenia považujeme za vhodné spomenúť problémy, s ktorými sme sa stretli. Prvý sa týka umiestnenie konfiguračného súboru `clouds.yaml`, ktorý je opísaný v dokumentácii. Tento súbor odporúčame umiestniť do adresára `/var`, keďže na základe našich skúseností pri inom umiestnení dôjde k blokovaniu prístupu k súboru aplikáciou *apparmour*. Pre definovanie cesty k súboru sme využili argument programu *os-client-config.* V tejto práci nebudeme uvádzať obsah nášho súboru, keďže obsahuje privátne informácie týkajúce sa prístupu k administrátorským častiam cloudu.

Overenie funkčnosti exportéra sme vykonali rovnakým spôsobom ako v 6.4.2, avšak s využitím portu 9180.

6.4.5 Collectd

Démona *collectd,* máme záujem inštalovať len na výpočtový uzol. Za týmto účelom je k dispozícii inštalačný balíček pre *collectd* aj požadované dodatky *virt* a *write_prometheus*, avšak len v repozitári EPEL, čo je repozitár určený pre OS *RedHat*. V rámci OS *CentOS* je možné tento repozitár aktivovať. Proces inštalácie uvádzame nižšie:

```
//inštalácia epel repozitára
#yum install epel-release
//inštalácia collectd s dodatkami z epel repozitára
# yum --enablerepo=epel install collectd collectd-virt collectd-
write_prometheus
//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)
```

V tomto prípade už máme vytvorenú aplikáciu ako službu, pre jej automatické zapínanie pri štarte systému nám ju stačí aktivovať pomocou `#systemctl enable collectd`.

Po inštalácii je nutné vykonať zmeny v konfiguračnom súbore, ktorý má v základnom nastavení väčšinu konfigurácie v tvare komentárov, začínajúcich znakom „#“. Toto následne uľahčuje zmeny, keďže často stačí vymazať tento znak zo začiatku riadku

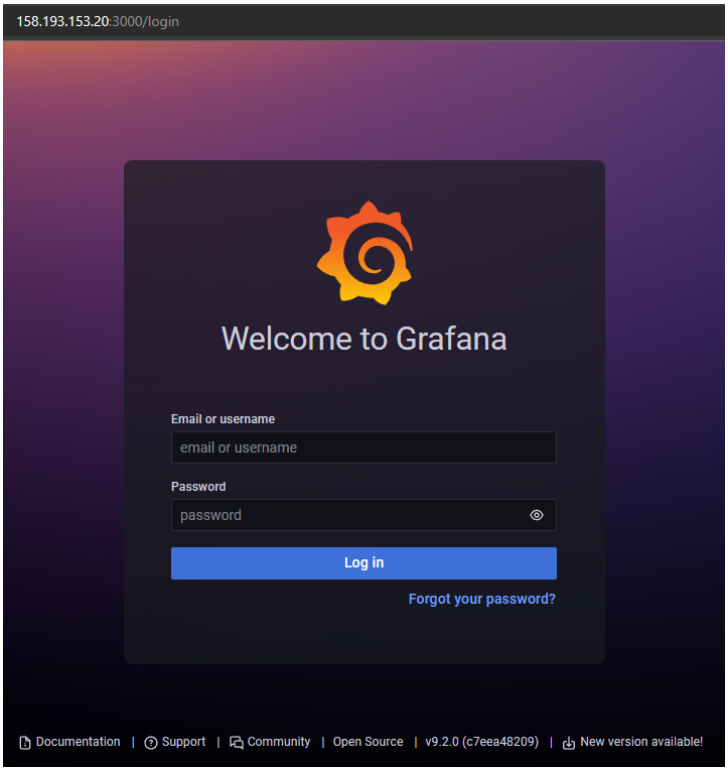
pre aktivovanie danej konfigurácie. Cesta k tomuto súboru bola pri našej inštalácii nasledovná: `/etc/collectd.conf`. Pre naše účely potrebujeme odstrániť znaky indikujúce komentár pre riadky obsahujúce *LoadPlugin virt* a *LoadPlugin write_prometheus*, pre načítanie dodatkov. Ďalej upraviť obsah medzi `<Plugin virt></Plugin>` a `<Plugin write_prometheus></Plugin>`. Pre opisy významu jednotlivých riadkov odporúčame oficiálny manuál, ktorý je detailne popísaný, pozri [70]. Naša konfigurácia je k dispozícii v **Chyba! Nenašiel sa žiaden zdroj odkazov.**, súbor konfigurácie.

Overenie funkčnosti sme opäť vykonali pomocou príkazu `#curl`, pričom *collectd* sprístupňuje metriky na porte 9103. Rovnako sme overili beh služby vypísaním jej statusu. Tu sme sa stretli s chybovým výpisom: `collectd[26847]: virt plugin: Array index out of bounds: tag_index = 10`. V snahe vypátrať dôvod výskytu tejto hlášky sme zistili nasledovné: keďže CentOS7 je už starší operačný systém, nie je možné naň nainštalovať najnovšiu verziu tohto démona, z dôvodu závislostí na knižniciach, ktoré nie sú pre tento OS dostupné. Najnovšia podporovaná verzia je 5.9, zatiaľ čo najnovšia celkovo je 5.12. Toto sa čiastočne prejavuje na množstve dát, ktoré sme schopní extrahovať z jednotlivých VM, keďže verzia 5.12 používa novšiu verziu *libvirt* API, ktorá poskytuje ďalšie funkcionality. Rovnako je to pôvodom spomínanej chybovej hlášky, ktorá je v novších verziách opravená. Z nášho pozorovanie nemá táto chyba žiadny vplyv na korektný beh démona.

6.4.6 Grafana

Proces inštalácie je detailne popísaný v oficiálnej dokumentácii, viď [66] časť *Setup->Install Grafana.* Pri našom nasadení sme využili inštaláciu pomocou balíčku `.deb`. Pri tejto forme je prípadné aktualizácie nutné vykonávať manuálne. Medzi momentom nasadenia a písania tejto práce vytvorili vývojári APT repozitár, ktorý je možné pridať do inštalačného manažéra *apt* na hostiteľskom systéme. Proces inštalácie a aktualizovania aplikácie je potom zautomatizovaný.

Po inštalácii môžeme pristúpiť k webovému rozhraniu *Grafana* na porte 3000, pozri **Obrázok 6.10**. Pre prvé prihlásenie využijeme používateľské meno aj heslo *admin.* Následne sme boli vyzvaní k zmene administrátorského hesla.



Obrázok 6.10 Webové rozhranie Grafana

6.4.7 Aplikácia ako služba

V prípade, ak potrebujeme mať aplikáciu aktívnu automaticky hneď po štarte systému, vieme ju nastaviť ako službu a aktivovať jej automatické zapínanie. Tento proces odporúčame vykonávať ako privilegovaný používateľ *root*. Pred samotnou konfiguráciou potrebujeme mať k dispozícii textový editor podľa vlastnej voľby, v našom prípade *nano* a poznať cestu k spustiteľnému súboru aplikácie.

Prvým krokom je vytvorenie konfiguračného súboru pre našu novú službu, ktorý musí byť umiestnený v adresári `/etc/systemd/system`. Jeho názov musí spĺňať nasledovnú konvenciu: `<NÁZOV_SLUŽBY>.service`. Ak by sme chceli vytvoriť *Prometheus* ako službu, konfiguračný súbor by mohol mať názov `prometheus.service`. Následne potrebujeme tento súbor adekvátne upraviť. Existuje veľké množstvo možností konfigurácie, no my v rámci nášho nasadenia využijeme len niektoré z nich. V prípade záujmu odkazujeme čitateľa na [71], kde je celá problematika takéhoto typu súborov zrozumiteľným spôsobom opísaná. Pre vytvorenie konfiguračného súboru pozri vzor alebo

nami použitý konfiguračný súbor pre daný program, ktorý je rovnako k dispozícii v prílohách práce.

Po dokončení konfiguračného súboru je potrebné ešte:

1. Inštruovať manažéra služieb, aby si opätovne načítal konfigurácie pomocou: `#systemctl deamon reload`.
2. Aktivovať automatické spúšťanie služby pri štarte systému s využitím: `#systemctl enable <NÁZOV_SLUŽBY>`.

Po vykonaní vyššie spomenutých krokov sa manažér služby, pri každom spustení systému, pokúsi spustiť aplikáciu. Okrem toho môžeme interagovať s programom za využitia štandardných príkazov, napríklad: `#systemctl [start|stop|restart|status]`. V prípade problémov odporúčame nahliadnuť do systémového žurnálu, ktorý často poskytne dôvod nefunkčnosti. Pristúpiť k nemu vieme pomocou: `#journalctl -xe`.

6.4.8 Konfigurácia *Prometheus* a overenie

Nasadením exportérov sme získali možnosť extrahovať metriky z oblastí záujmu. Tie však potrebujeme uchovávať a sprístupniť pre ďalšie prvky systému, na čom nám slúži práve *Prometheus*. Jeho konfiguráciou zadefinujeme, z ktorých miest, a ako často má zbierať metriky. Pre kompletnosť si najprv uvedieme stručný prehľad nasadených exportérov s ich umiestnením, pozri **Tabuľka 6.2**.

Exportér	Spustiteľný súbor	Konfiguračný súbor	URL koncového bodu s metrikami (z pohľadu Prometheus)
Centrálny server (158.193.153.20)			
openstack-exporter	/snap/golang-openstack-exporter/103/bin/openstack-exporter	/var/prometheusProject/clouds.yaml	http://localhost:9180/metrics
snmp-exporter	/opt/snmp_exporter/snmp_exporter	/opt/snmp_exporter/snmp.yml	http://localhost:9116/snmp
Node exporter	/opt/node_exporter/node_exporter	X	http://localhost:9100/metrics
Výpočtový uzol (192.168.1.21)			
Node exporter	/opt/node_exporter/node_exporter	X	http://192.168.1.21:9100/metrics
collectd	/usr/sbin/collectd	/etc/collectd.conf	http://192.168.1.21:9103/metrics
Riadiaci uzol (192.168.1.3)			
Node exporter	/opt/node_exporter/node_exporter	X	http://192.168.1.3:9100/metrics

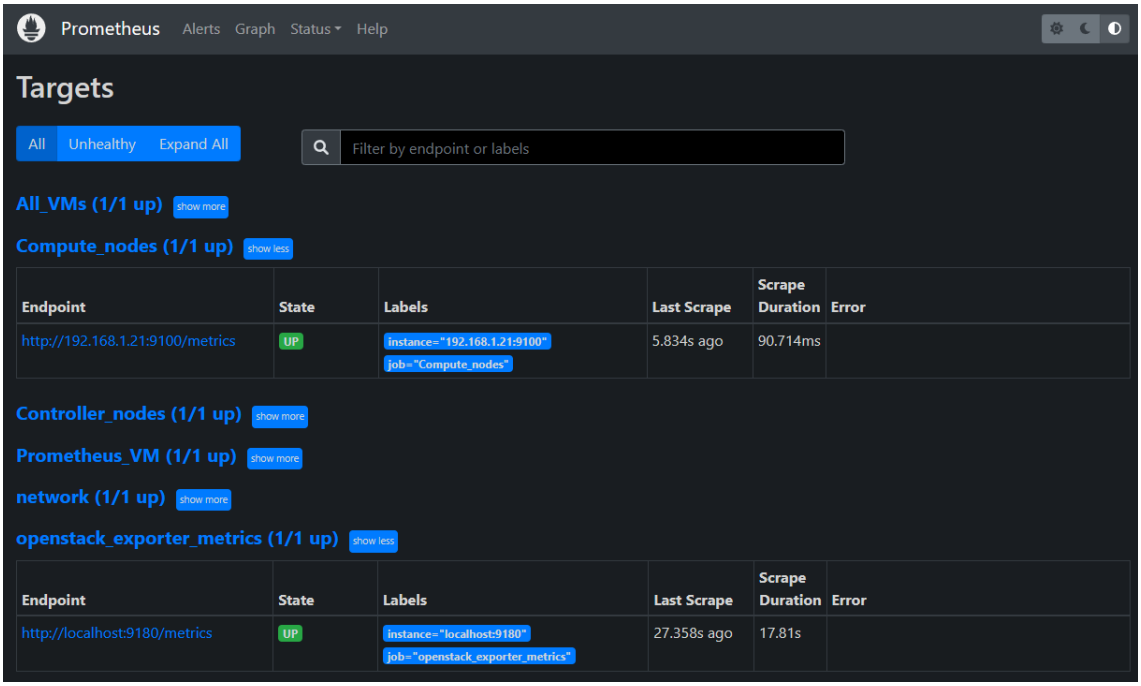
Tabuľka 6.2 Zoznam a umiestnenie nasadených exportérov

Syntax a forma konfigurácie je uvedená v oficiálnej dokumentácii [60], časť *Prometheus->Configuration*, preto ju v tejto práci nebudeme detailne opisovať. Z pohľadu zbierania metrík sa v konfigurácii definujú takzvané *jobs*, čo prekladáme ako práce. Práca predstavuje zbierku koncových bodov s rovnakým účelom, napr. koncové body výpočtových uzlov. Všetky metriky zozbierané v rámci takejto definície dostanú pridelený špeciálny štítok, uľahčujúci dopyty a vyhľadávanie konkrétnych metrík. V rámci tejto konfigurácie je možné zadefinovať rôzne parametre, ako napr. pravidelnosť zbierania. Najdôležitejšou definíciou sú však ciele, ktoré definujú konkrétne koncové body exportérov. V našom prípade sme nakonfigurovali nasledovné práce:

- **Prometheus_VM** – Určená pre metriky týkajúce sa centrálneho servera.
- **openstack_exporter_metrics** – Metriky z CC OS OpenStack.
- **Controller_nodes** – Práca určená pre metriky zo všetkých kontrolných uzlov. Aj keď momentálne má monitorovaný cloud len jeden takýto uzol, snažíme sa myslieť na škálovanie. V prípade pridania ďalšieho uzla sa jeho koncový bod jednoducho pridá do cieľov tejto definície.
- **Compute_nodes** – Rovnako ako predchádzajúca, ale pre výpočtové uzly.
- **All_VMs** – Práca určená pre metriky získané z virtuálnych inštancií.
- **network** – Dedikované pre metriky zo sieťových zariadení.

Kompletnú konfiguráciu prikladáme do príloh, pozri: súbor konfigurácie. Globálne sme nastavili interval zbierania metrík na pätnásť sekúnd. V prípade exportéra pre *OpenStack* sme túto hodnotu prepísali na jednu minútu. K tomuto kroku nás viedlo neskoršie zistenie, že dĺžka extrakcie metrík exportérom trvá v niektorých prípadoch aj dvadsať a viac sekúnd. Tieto hodnoty sú k nahliadnutiu pre všetky exportéri vo webovom rozhraní *Prometheus*, v časti S*tatus->Targets* položka *Last Scrape,* viď napr **Obrázok 6.11**.

Je samozrejmosťou, že po vykonaní konfigurácie musíme overiť jej funkčnosť, čo vykonáme prístupom na webové rozhranie *Prometheus* servera a navigáciou do časti S*tatus->Targets*. Toto zobrazuje **Obrázok 6.11**. Na obrázku si môžeme všimnúť nakonfigurované práce, pričom všetky ich ciele indikujú zdravý status slovom *up*. V prípade práce pre výpočtové uzly a *openstack-exporter* sme si nechali zobraziť aj detaily, kde môžeme vidieť ciele, status, aké štítky sú pridávané metrikám z týchto cieľov, kedy bol naposledy vykonaný zber a ako dlho trval, prípadne chybovú hlášku.



Obrázok 6.11 *Prometheus* rozhranie - práce

Overením tejto časti považujeme cieľ merania technických parametrov cloudu a sieťovej infraštruktúry za účelom prehľadu o stave za splnený.

6.5 Príprava pre vizualizáciu

Zozbierané metriky s možnosťou vyhľadávania poskytnú používateľom monitorovacieho systému malú pridanú hodnotu. Aby dokázal mať administrátor komplexný prehľad nad stavom služieb a platformy, je potrebné tieto dáta vizualizovať. Pre potreby rýchlej reakcie na problém potrebujeme zase nastaviť adekvátny proces upozorňovania. Práve týmto problémom sa budeme venovať v ďalšom priebehu práce, pričom sa budeme riadiť dvomi pohľadmi, tak ako sme pojednávali v kapitole 1.3. Z toho vyplýva, že máme záujem vytvoriť minimálne dva typy monitorovacích obrazoviek, ktoré budú slúžiť pre používateľa a administrátora, pričom sa budeme držať definovaných odporúčaní v kapitole 4.4.2.

V prípade používateľskej obrazovky bude snahou vizualizovať jeho dostupné prostriedky a ich využitie, spolu s metrikami uvedenými v Tabuľka 2.1.

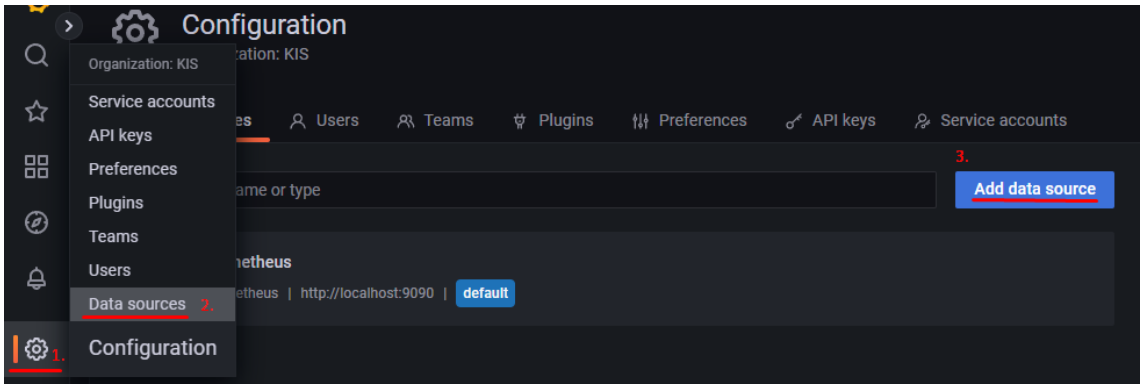
V prípade administrátorskej obrazovky bude snahou na jednom paneli zobraziť celkový stav služieb opísaných v kapitole 4.2 a stav fyzickej infraštruktúry so zameraním na prvky spomenuté v kapitole 3.3.

Problematike upozorňovania sa budeme venovať len v kontexte administrátorov.

V kapitole 4.4.1 sme pojednávali o vhodnosti zasielanie reportov a ich obsahu. Grafana poskytuje takúto funkcionalitu, avšak len v platenej *enterprise* verzii.

6.5.1 Prepojenie Prometheus a Grafana

Prvým krokom je nutnosť prepojenia monitorovacieho servera s vizualizačnou platformou. Tento proces sa začína prihlásením do *Grafana*, kde sa následne potrebujeme v navigačnom paneli na ľavej strane dostať do časti *Configuration->Data sources,* pročom volíme možnosť *Add data source,* pozri **Obrázok 6.12 Grafana pridanie zdroja**



Obrázok 6.12 Grafana pridanie zdroja dát

dátObrázok 6.12.

Otvorí sa nám ponuka všetkých podporovaných dátových zdrojov, my volíme *Prometheus* a následne vyplníme potrebné polia. V našom prípade sme tento zdroj nazvali *KIS-OpenStack-Old* a URL *Prometheus* servera sme nastavili na http://localhost:9090. V dolnej časti konfigurovanej obrazovky potvrdíme zadané parametre tlačidlom *Save & test*. Po dokončení môžeme postupne prejsť na vytváranie obrazoviek.

6.5.2 Importovanie prednastavených šablón

Pre začiatok práce s platformou odporúčame importovanie dostupných, prednastavených, monitorovacích obrazoviek k exportérom. My budeme síce vytvárať vlastné, špecializované obrazovky, zodpovedajúce požiadavkám riešenia, no tieto generické poskytujú dobrý štartovací bod pre ďalšiu prácu, keďže boli vytvorené systémovými administrátormi a poskytujú náhľad na dôležité metriky z exportérov. V našom prípade máme k dispozícii šablóny pre *Node exporter*, *openstack-exporter* a *snmp exporter*. Všetky dostupné obrazovky je možné prehliadať online na: https://grafana.com/grafana/dashboards/. Spôsob ich importu je veľmi jednoduchý a potrebujeme k nemu poznať len ID požadovanej šablóny. Krok za krokom je popísaný v [66], časť *Dashboards->Manage dashboards* odsek *Import a dasboard*. Proces je

intuitívny a skrátene sa potrebujeme dostať v navigačnom paneli do *Dashboards->Import*, zadať ID, nahrať, zvoliť zdroj dát a potvrdiť operáciu tlačidlom *Import*. *Grafana* nás automaticky presmeruje na novo vytvorenú obrazovku. Tieto šablóny sú často príliš komplexné, no odporúčame ich pre zoznámenie sa s platformou.

6.5.3 Priečinky

Skorej, ako prejdeme vytváraniu obrazoviek, objasníme koncept priečinkov. *Grafana* poskytuje pre logické oddeľovanie obrazoviek a upozornení priečinky, pričom ten továrenský má názov *General* a nie je možné ho odstrániť. Pre priečinky je možné nastaviť povolenia pre prístup k ich obsahu. Problematika povolení, rolí užívateľov a tímov je v *Grafana* pomerne obsiahla, preto odkazujeme čitateľa na oficiálnu dokumentáciu [66], časť *Administration-> Roles and Permisions.* Dôležité je, že každý užívateľ ma pridelenú jednu z troch dostupných rolí: *Admin, Editor* alebo *Viewer*, na základe ktorej riadime povolenia prístupu a úprav k obsahu priečinkov. V rámci nášho riešenia sme pre obrazovky vytvorili tri nové zložky:

- **Old-OpenStack** – Pre obrazovky a upozornenia určené administrátorom, týkajúce sa nami monitorovaného cloudu. Prístup do zložky len pre užívateľov s rolou *Admin* a *Editor*.
- **Templates** – Určené pre importované obrazovky. Povolený prístup pre role *Admin* a *Editor*.
- **Users** – Miesto pre obrazovky určené klientom. Povolenie prístupu aj pre rolu *Viewer*.

6.5.4 Vytvorenie používateľov

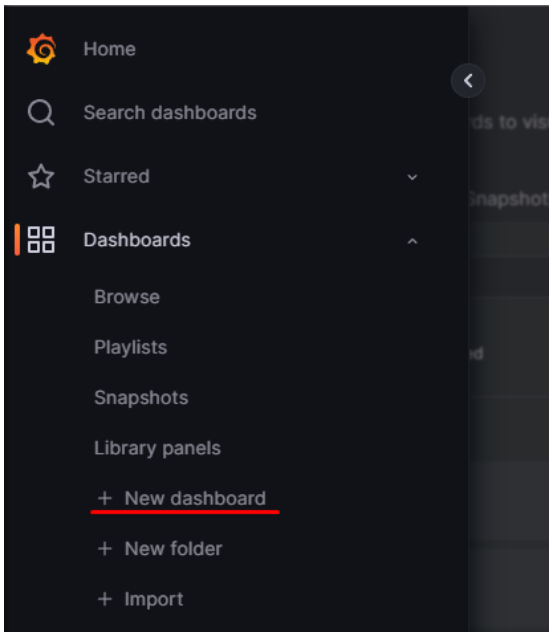
Pred vytváraním obrazovky pre klientov, potrebujeme ešte objasniť problematiku vytvárania takýchto užívateľov. Na začiatok spomenieme, že *Grafana* podporuje integráciu rôznych autentifikačných systémov, pričom všetky informácie sú dostupné v [66], časť *Setup->Configure security->Configure authentication*. My sme sa v rámci nášho riešenia takejto integrácii nevenovali. Užívateľov vytvárame a spravujeme manuálne v grafickom rozhraní, pozri [66] časť *Administration->User management*, alebo pomocou API, pozri [66] časť *Developers->HTTP API->Admin HTTP API*.

Pre klientov navrhujeme vytvoriť účet s takým prihlasovacím menom, aby zodpovedalo názvu jemu prislúchajúceho *OpenStack* projektu. Toto nám neskôr pri tvorbe obrazovky pre klientov umožní využiť globálnu *Grafana* premennú `${__user.login}`, pomocou ktorej zabezpečíme, že daný používateľ bude mať prístup len k informáciám

z jeho projektu. Rovnako predpokladáme, že klientsky účet bude mať najnižšiu možnú rolu, *viewer* a prístup len do zložky *Users*, ako sme definovali v kapitole 6.5.3. Zoznam všetkých globálnych premenných aj s popisom je dostupný v dokumentácii [66], časť *Dashboards->Variables->Manage variables.*

6.5.5 Vytvorenie obrazoviek a panelov

Monitorovacia obrazovka sa skladá z panelov, ktoré vizualizujú metriky pomocou jedného alebo viacerých dopytov na zdroj dát. To v našom prípade znamená dopyty v *PromQL* na *Prometheus* server. Novú obrazovku vytvoríme napríklad v navigačnom paneli v časti *Dashboards->New dashboard*, pozri **Obrázok 6.13**.

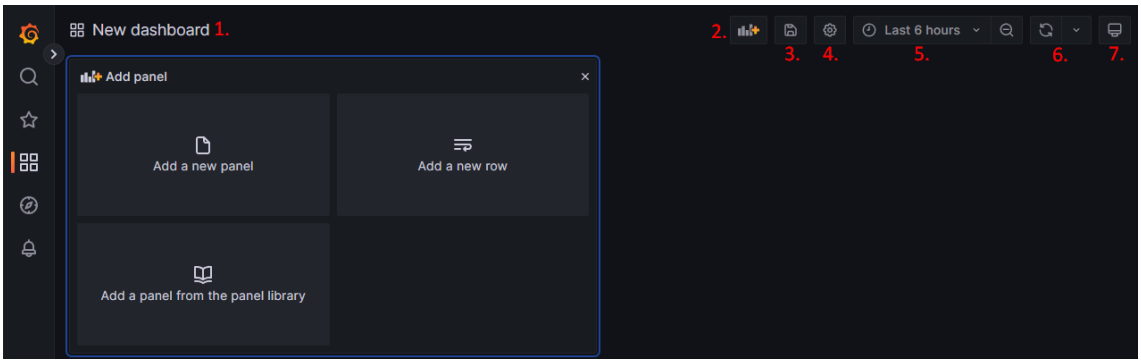


Obrázok 6.13 *Grafana* vytvorenie obrazovky - navigačný panel

Grafana nás presmeruje do novo vytvorenej obrazovky, kde môžeme začať s vytváraním panelov. **Obrázok 6.14** zobrazuje novo vytvorenú obrazovku, pričom číslami jedna až sedem označujeme jednotlivé časti rozhrania, ktorých úlohu popisujeme nižšie:

1. **Názov obrazovky** – Mal by pomôcť užívateľovi identifikovať čo je vizualizované.
2. **Pridať panel** – Tlačidlo pre pridanie nového panela. Vytvorí na obrazovke rovnakú entitu, ako sa v **Obrázok 6.14** nachádza pod názvom obrazovky.
3. **Uloženie** – Pri každej zmene je, pre jej uchovanie nutné obrazovku uložiť. Ukladanie funguje spôsobom verziovania, kedy sú k dispozícii predchádzajúce verzie na import. Tie sa nachádzajú v nastaveniach obrazovky, sekcia *Versions*. Pri ukladaní obrazovky určujeme jej cieľový priečinok.
4. **Nastavenia obrazovky** – V tejto časti sa nachádzajú všeobecné nastavenia pre celé prostredie obrazovky. Skladájú sa zo siedmich kategórií, pričom za najdôležitejšie považujeme základné nastavenia, napr. názov a premenné.
5. **Časový rozsah** – Používateľ si vie v tejto časti definovať, pre aké časové obdobie si želá vizualizovať metriky na obrazovke.

6. **Automatické obnovovanie** – V základnom nastavení je táto funkcia vypnutá. Pre dynamické obnovovanie obsahu na obrazovke v určitých časových intervaloch, je nutné zapnúť túto možnosť výberom intervalu. Bez tohto sa obsah obnoví len pri načítaní stránky.
7. **Zmena zobrazovacieho režimu** – Týmto komponentom má užívateľ možnosť prepnúť medzi tromi dostupnými režimami. Prvý je základný, pri druhom sa skryje navigačný panel a tretí je určený pre zobrazenie na celú obrazovku, napríklad pre dedikovaný monitor alebo televíziu.

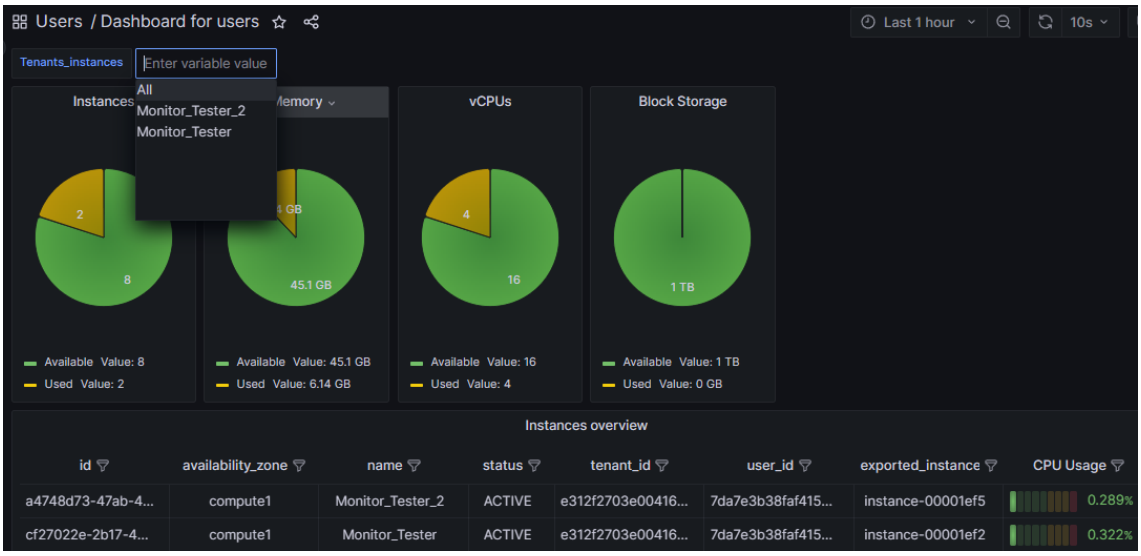


Obrázok 6.14 Popis rozhrania obrazovky

V rámci panelov je k dispozícii veľké množstvo vizualizačných možností, od jednoduchého zobrazenia hodnoty, cez koláčový graf, tabuľky, až po grafy zobrazujúce vývoj metriky v čase. Tieto nebudeme detailne opisovať. Viac informácií o vytváraní obrazoviek a panelov je možné nájsť v dokumentácii [66], časť *Dashboards->Build dashboards.*

6.6 Monitorovacia obrazovka pre klientov

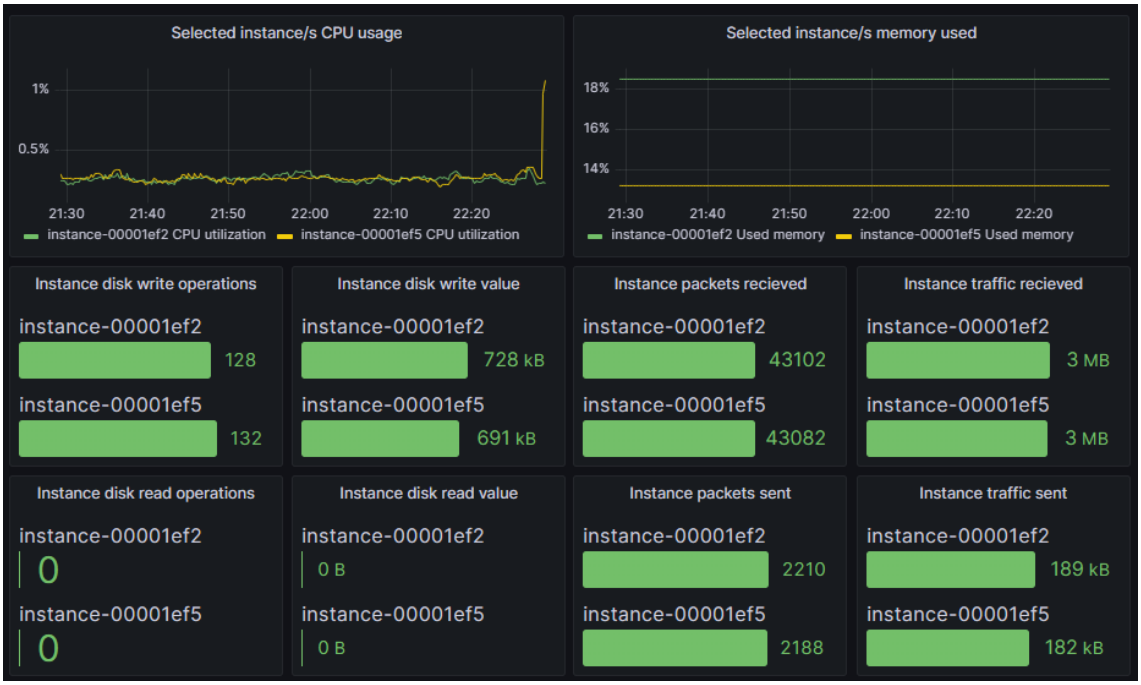
V predchádzajúcej kapitole sme pojednávali o potrebe vytvorenia monitorovacích obrazoviek, pričom jedna z nich by mala byť určená pre klientov. Túto dedikovanú obrazovku sme nazvali *Dashboard for users* a uložili do priečinka *Users*. Pri jej vytváraní sme mysleli na to, aby poskytovala všeobecné informácie o poskytnutom prostredí v cloude, ale aj detailné informácie o užívateľových inštanciách. Vizualizovať budeme pomer dostupných a využitých zdrojov v projekte, pomocou koláčových grafov. Medzi tieto zdroje zaraďujeme jadrá vCPU, počet inštancií, RAM a blokové úložisko. Rovnako poskytujeme užívateľovi tabuľku so zoznamom jeho vytvorených inštancií, kde k nim poskytujeme základné informácie, ako napríklad meno, stav, identifikátor a využitie vCPU, pozri **Obrázok 6.15**. Aby sa naše riešenie priblížilo aj tomu priemyselnému, vizualizujeme taktiež všetky metriky uvedené v Tabuľka 3.1, a to buď pre všetky, alebo klientom zvolenú inštanciu, pozri aj **Obrázok 6.16**. Na zvolenie inštancie slúži užívateľovi rozbaľovací zoznam v hornej časti obrazovky s názvom *Tenants_instrances,* čo ukazuje aj **Obrázok 6.15**.



Obrázok 6.15 Ukážka časti obrazovky pre klientov – informácie o prostredí

Jednotlivé panely sme vytvárali v grafickom rozhraní *Grafana*, no ako ďalšia možnosť je k dispozícii aj konfigurácia s využitím jazyka JSON. Z pohľadu rozsahu práce nie je možné, aby sme pojednávali o nastavovaní jednotlivých panelov. V rámci ich konfigurácie je časovo najnáročnejšou úlohou správne vytvorenie sady dopytov a korektné vizuálne nastavenie. Pre čitateľa uvádzame vygenerovaný JSON model celej obrazovky, pričom použité dopyty sa dajú dohľadať pod výrazom „*expr*“. Súbor je k dispozícii

v prílohách, pozri **Chyba! Nenašiel sa žiaden zdroj odkazov.**, súbor JSON model obrazovky pre klientov.



Obrázok 6.16 Ukážka časti obrazovky pre klientov – informácie o inštanciách

Významnou funkciou tejto obrazovky je aj dynamické získavanie dát na základe prihlasovacieho mena klienta. Tým myslíme, že každý klient po prihlásení vidí vizualizované dáta zo svojho prostredia. Pre dosiahnutie spomínanej funkcionality sme využili premenné obrazovky. Tento koncept je pomerne rozsiahly a náročný. Detailne je popísaný v oficiálnej dokumentácii [66], časť *Dashboards->Variables*. Nami vytvorené premenné sú typu dopyt (*query*) a nazvali sme ich *Tenant* a *Tenants_instances*.

Premenná *Tenant* pri načítaní obrazovky vykoná dopyt `openstack_identity_project_info{name="${__user.login}"}`, vracajúci metriku s informáciami o projekte, ktorého meno sa zhoduje s obsahom globálnej premennej `${__user.login}`. V tejto premennej je podľa dokumentácie uložené prihlasovacieho meno aktuálneho používateľa. Z metriky, pomocou regulárnych výrazov, extrahujeme zo štítkov meno a identifikátor projektu klienta. Ten sa uloží ako hodnota premennej *Tenant*, čo využívame pri dopytoch vo vizualizačných paneloch. Vďaka tomu sú klientom zobrazované len dáta z ich prostredí.

Premenná *Tenants_instances* využíva dopyt s vnorenou premennou *Tennant*, pre získanie zoznamu všetkých klientových VM. Nami vytvorený dopyt vyzerá nasledovne: `openstack_nova_server_status{tenant_id=~"$Tenant"}`. Práve na základe

hodnoty štítka *tenant_id* vieme vyfiltrovať inštancie patriace klientovi. Rovnako, ako pri predchádzajúcej premennej, extrahujeme z tohto zoznamu meno a identifikátor VM, ktorý uložíme ako hodnotu premennej, čo využívame vo vizualizačných paneloch. Snímky obrazovky nastavenia týchto premenných sú k dispozícii v prílohách, **Chyba! Nenašiel sa žiaden zdroj odkazov.** – súbor Prehľad panelov.

V priebehu riešenia sme zistili, že takto vytvorená obrazovka môže byť užitočná aj pre administrátora cloudu. V prípade, keď pri premennej *Tenant* odstránime podmienku zhodnosti s menom užívateľa, nám takýto dopyt vráti zoznam všetkých projektov v cloude. Tým sa v obrazovke vytvorí rozbaľovacie menu so všetkými projektami. Vďaka tomu, môže mať administrátor rýchly prehľad o všetkých prostrediach, pričom dokáže rýchlo identifikovať vlastníctvo inštancie, vykazujúcej podozrivú aktivitu. Takto upravenú obrazovku sme nazvali *Tenants overview dashboard* a uložili ju do priečinka *Old-OpenStack*. Jej JSON model prikladáme v **Chyba! Nenašiel sa žiaden zdroj odkazov.**.

Z pohľadu požiadaviek na riešenie sme vytvorením tejto časti splnili zber a vizualizáciu SLA parametrov z prostredia klientov. V kontexte cieľa práce sme splnili časť vhodnej interpretácie prehľadu stavu a vyťaženia prostredia používateľovi služby. Výstupom navyše je uľahčenie práce administrátorov v kontexte prehľadu o projektoch a inštanciách v cloude.

6.7 Monitorovacia obrazovka pre administrátora

V tejto časti sa budeme venovať vytvoreniu prehľadnej obrazovky pre administrátora. Jej forma a obsah bol konzultovaný s administrátormi na KIS FRI UNIZA. Vzhľadom na väčšie množstvo panelov budeme obrazovku prezentovať postupne.

6.7.1 Vizualizácia monitoringu *OpenStack* služieb

V rámci *OpenStack* služieb, pozri aj kapitolu 4.2, máme záujem vizualizovať ich dostupnosť a zdravie, keďže pri ich výpadku dochádza k zhoršeniu kvality poskytovanej služby. Konkrétne budeme vizualizovať SLI pre dostupnosť jednotlivých *OpenStack* služieb v časovom okne, ktoré si definuje administrátor. Toto umožní overiť hodnotu SLI pre požadované časové okná, čo môže byť užitočné z pohľadu spätnej kontroly dodržiavania SLA. Ďalej budeme slovne zobrazovať zdravie spomínaných služieb, aby dokázal administrátor rýchlo identifikovať ich správnu funkčnosť. Posledným aspektom záujmu je vizualizácia časových osí, na ktoré sa vykresľuje dostupnosť služieb v čase. Toto býva bežným spôsobom prezentovania dostupnosti online služieb. Z takejto prezentácie dokážeme presne vyčítať, kedy služba vypadla, a ako dlho trvalo jej zotavenie. Za týmto

účelom sme vytvorili niekoľko panelov, ktoré opisujeme nižšie. Spôsobu ich pridávania sme sa venovali v kapitole 6.5.5.

Prvému sa budeme venovať panelu SLI pre dostupnosť *OpenStack* služieb. Za účelom výpočtu SLI sú pre každú službu k dispozícii binárne metriky. Tie indikujú ich dostupnosť. Pre demonštráciu si ukážeme dopyt pre určenie dostupnosti komponentu *nova*: *avg_over_time(sum((openstack_nova_up or on() vector(0)))[[$__range:]) * 100*. Najviac vnorený výraz prezentuje metrika vyjadrujúca stav služby a nulový vektor, ktorý potrebujeme v prípade fyzického výpadku uzla, na ktorom beží služba. Pri takej udalosti *openstack-exporter* nie je schopný získať hodnotu metriky, čoho výsledkom je v danom časovom okne položka *NODATA*. Pre správny výpočet dostupnosti potrebujeme vyplniť tieto prázdne okná hodnotou nula, na čo nám slúži práve nulový vektor. Funkciou *sum* zlúčime nulový vektor s vektorom metriky, čím získame neprerušený, výsledný vektor. Nad ním vypočítame priemernú hodnotu v časovom intervale zadanom v rozhraní obrazovky, čo prezentuje premenná `$__range`. Pre získanie percentuálnej hodnoty výsledok na záver násobíme sto. Nastavenia dopytov pre všetky panely sú súčasťou **Chyba! Nenašiel sa žiaden zdroj odkazov.**, súbor Prehľad panelov.

Panely prezentujúce zdravie služieb fungujú na základe metrík `openstack_<SLUŽBA>_agent_state`, ktorá vracia zoznam súčastí danej služby a ich status prezentovaný binárne. V práci sme o týchto súčastiach už pojednávali v kapitole 4.2. V prostredí monitorovanej platformy dokážeme extrahovať takéto informácie zo služieb *nova, cinder* a *neutron*. V prípade, ak všetky súčasti fungujú korektne, mapujeme túto hodnotu na slovo *HEALTHY*, ak niektoré z nich nefungujú, prezentujeme to slovom *PROBLEM* a pri nefunkčnosti všetkých súčastí využívame slovo *DOWN*. Môžeme si všimnúť, že **Obrázok 6.18** zobrazuje pri komponente *cinder* problém. Na týchto paneloch sme vytvorili prepojenia na panely inej obrazovky, v ktorej zobrazujeme detaily komponentov, čo vyjadrujú malé okná v ľavých horných rohoch. Spôsob vytvorenia takýchto prepojení je popísaný v dokumentácii [66], konkrétne v *Dashboards->Build dashboards->Manage dashboard links*. Detaily služby *cinder* prezentuje **Obrázok 6.17**, kde si môžeme povšimnúť nefungujúcu súčasť *cinder-volume*. Toto korešponduje aj

Cinder

Time	adminState	hostname	job	service	zone	Status
2023-04-16 00:0...	enabled	block1@lvm	openstack_expor...	cinder-volume	nova	0
2023-04-16 00:0...	enabled	controller	openstack_expor...	cinder-scheduler	nova	1

Obrázok 6.17 Detaily služby cinder

s informáciami od administrátorov. Z toho vyplýva, že táto služba je síce dostupná, no nie je funkčná.

Pre posledný panel máme k dispozícii graf typu *state timeline*, do ktorého sme pridali binárne metriky prezentujúce dostupnosť služieb a *Grafana* zvyšok urobila automaticky.

Obrázok 6.18 zobrazuje výsledné spojenie všetkých spomenutých panelov. Týmto sme dokončili vizualizáciu dostupnosti služieb a pokračujeme ďalšími súčasťami platformy.

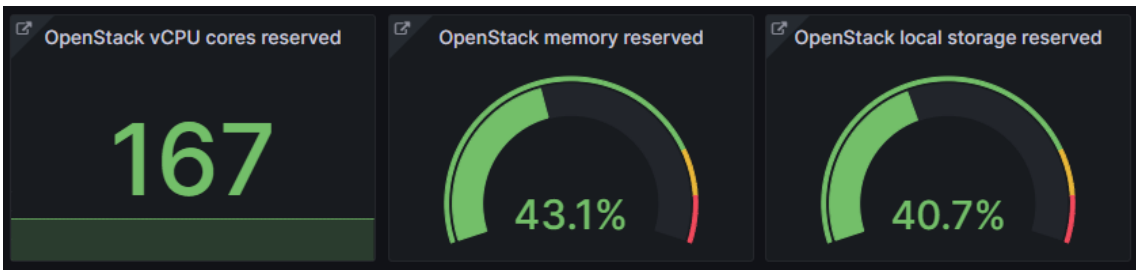


Obrázok 6.18 Vizualizácia monitoringu OpenStack služieb

6.7.2 Vizualizácia rezervovaných a dostupných zdrojov pre cloud

Aby mali administrátori prehľad o celkovom množstve poskytnutých zdrojov v cloude, máme záujem tieto informácie vizuálne prezentovať. Keďže sa venujeme cloudu, ktorý poskytuje primárne IaaS, medzi poskytované zdroje zaraďujeme vCPU, RAM a úložisko. Ako sme pojednávali v 6.6, klientom poskytujeme prehľad využitých zdrojov v ich projektoch. Z pohľadu administrátora je významné, aby videl celkový súčet využitých prostriedkov všetkými projektami. Na základe toho dokáže vyhodnotiť, či má jeho platforma dostatočnú kapacitu pre ďalších zákazníkov. Pre ujasnenie dodávame, že vizualizované dáta neprezentujú aktuálne, ale celkové využívané zdroje. To znamená, že pokiaľ by si klient vytvoril novú VM s jedným vCPU, celková hodnota využitých vCPU sa navýši o jedna. Ak by klient túto VM vypol, hodnota sa nezmení, keďže vCPU je stále mapované ako využité. K zníženiu celkového súčtu by došlo len pri vymazaní tejto VM.

Vizualizáciu týchto prvkov vykonávame pomocou troch panelov, zobrazujúcich celkový počet využitých vCPU, percentuálne využitie pamäte a úložiska, pozri **Obrázok 6.19**. Všetky informácie pre tieto panely pochádzajú zo služby *nova*. Pre počet využitých vCPU máme priamo metriku, vyjadrujúcu túto hodnotu. V prípade pamäte máme k dispozícii metriky vyjadrujúce množstvo využitých a dostupných bajtov. Dopyt v *PromQL*



Obrázok 6.19 Využité zdroje v cloude

pre získanie percentuálnej hodnoty využitia vyzerá nasledovne: `(openstack_nova_memory_used_bytes/(openstack_nova_memory_used_bytes+openstack_nova_memory_available_bytes)) * 100`. V prípade úložiska je dopyt totožný, akurát s korešpondujúcimi metrikami.

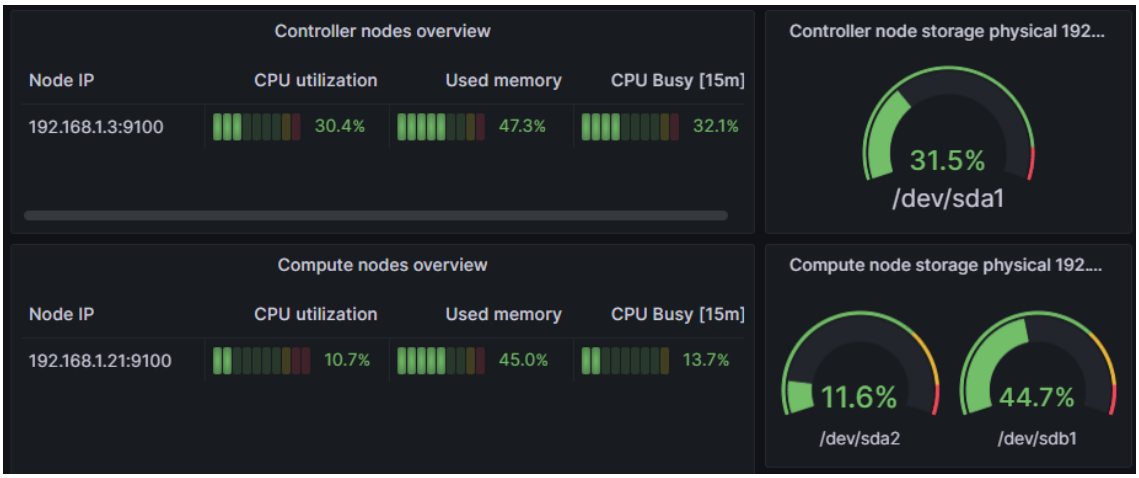
6.7.3 Vizualizácia monitoringu fyzickej infraštruktúry

Teraz prejdeme k vizualizácii dát z monitoringu fyzickej infraštruktúry, čo je aj jedným z hlavných cieľov tejto práce. *Grafana* je veľmi dobre navrhnutá pre vizualizovanie dynamických prostredí, čo sme sa snažili využiť aj my pri vytváraní panelov pre infraštruktúru. Benefitom takéhoto návrhu je automatická vizualizácia dát z novo pridaných zariadení.

Vizualizácia monitoringu fyzickej infraštruktúry tvorí väčšinovú časť obrazovky pre administrátora. Z dôvodu rozsiahlosti a množstva vytvorených panelov, a zároveň snahy o udržanie prehľadnosti tejto práce nebudeme uvádzať ukážky pre všetky vytvorené panely. Čitateľovi ich však sprístupňujeme v prílohách, pozri **Chyba! Nenašiel sa žiaden zdroj odkazov.**, súbor Prehľad panelov.

Čo všetko máme záujem vizualizovať z pohľadu fyzickej infraštruktúry? Ako sme pojednávali v kapitole 3.2, potrebujeme zobraziť využitie CPU, RAM a úložiska na fyzických uzloch a monitorovacom serveri. Administrátormi sme boli požiadaní aj o prehľad všetkých výpočtových uzlov a sieťových zariadení v infraštruktúre, pričom pod prehľadom rozumieme počet funkčných a nefunkčných zariadení spolu s ich zoznamom.

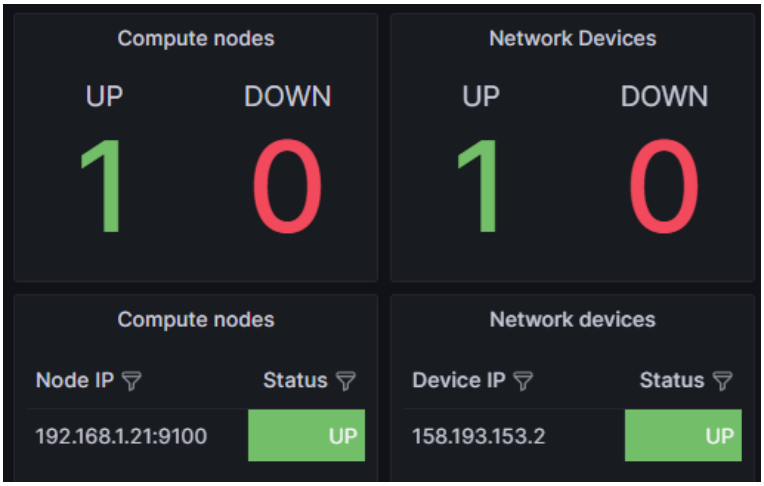
Administrátorom poskytujeme dve dynamické tabuľky, zobrazujúce všetky výpočtové a kontrolné uzly v rámci cloudu. V rámci takejto tabuľky prezentujeme informácie o IP adrese uzla, aktuálnom využití CPU, priemernom využití CPU za posledných pätnásť minút a využitej RAM. K tabuľkám poskytujeme navyše panely s informáciou o využití úložiska. V prípade existencie štandardizovaného systému názvoslovia súborových systémov, by bolo možné úložisko priradiť do dynamických tabuliek. Momentálne je však nutné pre každý uzol vybrať relevantné súborové systémy manuálne a nastaviť ich



Obrázok 6.20 Vizualizácia využitých zdrojov na uzloch

vizualizáciu. Opíšeme ešte dynamickosť spomínaných tabuliek, ktorá spočíva v tom, že po pridaní nového zariadenia do prislúchajúcej práce v konfigurácii *Prometheus*, viď 6.4.8, ho systém automaticky zaeviduje a pridá do tabuľky. Ukážku vizualizácie využitých zdrojov prezentuje **Obrázok 6.20**.

Pre prehľad všetkých zariadení v infraštruktúre sme vytvorili prehľadné panely, prezentujúce počet funkčných a nefunkčných zariadení, prčom sme oddelili výpočtové uzly



Obrázok 6.21 Prehľad zariadení v infraštruktúrre

od sieťových zariadení. Rovnako sme na základe požiadaviek vytvorili aj ich zoznam. Pri týchto prvkoch sme rovnako mysleli na ich dynamickosť, takže fungujú na rovnakom pripcípe, ako sme uvádzali pri tabuľkách vyššie. Pre lepšiu predstavu prezentuje **Obrázok 6.21** ukážku vytvorenej vizualizácie.

K hlavnej monitorovacej obrazovke sme vytvorili ešte dve dodatočné. Prepojenie k nim je priložené v pravom hornom rohu hlavnej monitorovacej obrazovky. Jedná sa o *Nodes info*, poskytujúcu detailný, vizualizovaný, prehľad všetkých zbieraných metrík exportérom *Node exporter* pre zvolený uzol. Obrazovka *Network devices* poskytuje rovnakú funkcionalitu, avšak pre sieťové zariadenia, tj. pre metriky zozbierané pomocou *snmp-exporter*. Pri týchto obrazovkách sme sa inšpirovali stiahnutými šablónami, pričom administrátorovi poskytujú komplexný prehľad o behu zariadení, čo môže byť užitočné pri zisťovaní pôvodu problémov.

6.7.4 Ostatné prvky monitorovacej obrazovky

Okrem doteraz pojednávaných komponentov sme do obrazovky pre administrátorov pridali dva panely navyše. Jedným je prehľad všetkých aktivovaných upozornení v rámci priečinka *Old-OpenStack*. Vďaka tomu administrátor prehľadne vidí, v ktorých častiach cloudu sú aktuálne problémy. O problematike upozorňovania budeme ešte pojednávať v kapitole 6.8. Druhý panel sme vytvorili na základe prosby administrátorov a poskytuje prehľad využitia vCPU všetkými inštanciami v cloude. Zámerom tohto panelu je poskytnúť spôsob identifikovania inštancie, ktorá po dlhšiu dobu vykazuje vysoké využitie vCPU, čo môže predznamenať neželané správanie.

Vytvorením prehľadnej obrazovky umožňujeme administrátorom jednoduchý prehľad o stave cloudu ako celku, pričom na základe farebnej interpretácie hodnôt dokáže vyhodnotiť, či dochádza k ohrozeniu poskytovaných služieb. Formálne by sme to mohli považovať za splnenie cieľa práce týkajúceho sa tejto problematiky, avšak považujeme za dôležité, aby bol nastavený proces aktívneho upozorňovania v prípade vzniku problémov, čomu sa venujeme v ďalšej časti.

6.8 Upozorňovanie

Grafana poskytuje sofistikovaný spôsob upozorňovania. Základnými prvkami tohto systému sú upozorňovacie pravidlá, kontaktné body a notifikačné politiky. Nižšie stručne uvedieme našu implementáciu týchto prvkov.

6.8.1 Upozorňovacie pravidlá

Vo všeobecnosti sa v tejto časti definujú podmienky, za ktorých sa vygeneruje upozornenie. V rámci nášho nasadenia sme vytvorili dvadsaťšesť pravidiel, ktoré sme si zadefinovali do siedmich vyhodnocovacích skupín. Tie zlučujú pravidlá, buď na základe vzájomnej súvislosti, alebo spoločného vyhodnocovacieho intervalu. Zoznam vytvorených pravidiel uvádza **Tabuľka 6.3**, pričom pre jej správne pochopenie si potrebujeme zadefinovať skratky pre výpočtový uzol (VU), sieťové zariadenia (SZ), riadiaci uzol (RU) a monitorovací server (MS). Pri pravidle využitia sieťových rozhraní vytvárame upozornenia len pre uvedené rozhrania, keďže na základe informácií od administrátorov len tie zahŕňame do fyzickej infraštruktúry cloudu.

Intervaly, ako aj podmienky pre vyhodnocovanie pravidiel sme určovali na základne skúseností získaných z praxe, keďže pri prieskume problematiky sme nenarazili na žiadnu formu štandardizácie.

Vyhodnovacia skupina	Interval vyhodnocovania	Pravidlá	Vyhodnotenie na základe
Dostupnosť	20 sekúnd	Dostupnosť VU	Posledná hodnota
		Dostupnosť SZ	
		Dostupnosť RU	
Vyťaženie hardvérových zdrojov	30 sekúnd	Vyťaženie CPU na VU	Priemer za posledných 5 minút viac ako 80% využitia
		Vyťaženie RAM na VU	
		Vyťaženie CPU na RU	
		Vyťaženie RAM na RU	
		Vyťaženie CPU na SZ	
		Vyťaženie RAM na SZ	
		Vyťaženie CPU na MS	
		Vyťaženie RAM na MS	
Využitie sieťových rozhraní	1 minúta	Využitie rozhraní Gi1/0/25-1/0/30	Priemer za posledných 5 minút viac ako 80% maximálnej priepustnosti rozhrania
Rezervované zdroje v OpenStack	10 minút	Rezervovaná RAM	Priemer za poslednú hodinu viac ako 80% z maxima
		Rezervované úložisko	
OpenStack služby	1 minúta	Dostupnosť služby Cinder	Posledná hodnota
		Dostupnosť služby Nova	
		Dostupnosť služby Heat	
		Dostupnosť služby Keystone	
		Dostupnosť služby Neutron	
		Dostupnosť služby Glance	
		Stav komponentov Cinder	
		Stav komponentov Nova	
		Stav komponentov Neutron	
Kontrola úložiska	30 minút	Využitie úložiska na VU	Priemer za poslednú hodinu viac ako 90% z maxima
		Využitie úložiska na RU	
Virtuálne inštancie	5 minút	Vyťaženie CPU na inštanciách	Priemer za posledenú hodinu viac ako 80% využitia

Tabuľka 6.3 Zoznam pravidiel

6.8.2 Kontaktné body

Kontaktné body definujú cieľovú destináciu a spôsob odoslania upozornení, pričom je podporovaná široká škála riešení, od jednoduchého emailu, cez firemné komunikačné platformy, až po *chat* aplikácie.

V našom prípade sme nakonfigurovali len jeden kontaktný bod, ktorý sme nazvali *Main alerts*. V rámci neho sme zadefinovali dva ciele, *Teams* pre administrátorov a *Discord* pre testovanie. Pri obidvoch platformách sme využili technológiu *webhook*, čo je funkcia postavená na HTTP, umožňujúca komunikáciu medzi dvomi API bodmi. V praxi funguje takým spôsobom, že v prípade vygenerovania upozornenia odošle *Grafana* na poskytnuté URL POST správu o výskyte udalosti. Táto požiadavka je následne spracovaná koncovou

platformou a užívateľovi príde správa s upozornením. Pre detailnejší popis všeobecného fungovanie tejto technológie odporúčame čitateľovi vysvetlenie od spoločnosti *RedHat*, pozri [72].

Proces vytvorenia kontaktného bodu pre nami využité platformy je veľmi jednoduchý. Je potrebné vygenerovať URL pre *webhook* na požadovaný kanál, následne v *Grafana* upraviť alebo vytvoriť nový kontaktný bod, zvoliť platformu a vložiť URL. V platforme *Teams* je možné takéto URL získať navigáciou do ďalších možností kanála, kde sa priamo nachádza táto možnosť. Nie je to však dostupné pre všetky typy kanálov, napr. vzdelávacie pre študentov.

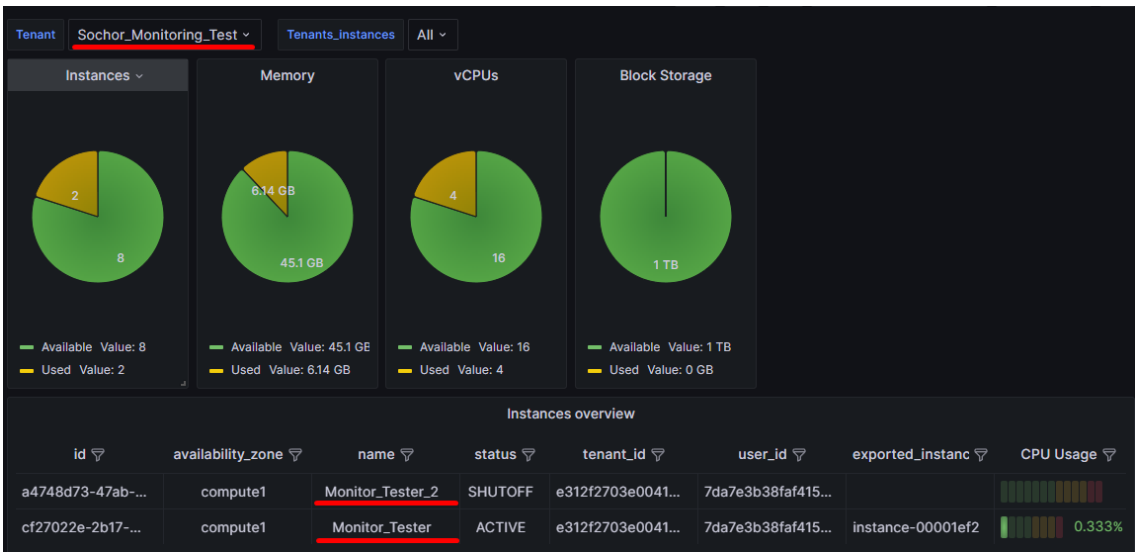
6.8.3 Notifikačné politiky

Tento koncept je užitočný najmä v prípade veľkých nasadení, kde sa predpokladá zasielanie upozornení rôznym administrátorským tímom na základe ich obsahu. Rozdelenie je možné vykonávať definovaním štítkov, ktoré je možné pridať v nastaveniach jednotlivých upozorňovacích pravidiel.

V tejto časti sa rovnako prepoja upozorňovacie pravidlá a komunikačné body. V základom nastavení je k dispozícii *root policy*, do ktorej budú smerovať všetky upozornenia. Z pohľadu nášho nasadenia nie je potrebné vytvárať žiadne ďalšie politiky, keďže rozsah nasadenia si to nevyžaduje. Nami vytvorený kontaktný bod sme teda pridali práve do spomínanej základnej politiky.

6.9 Testovanie

Po úspešnom nasadení komponentov, vytvorení obrazoviek a nastavení upozornení je nutné otestovať funkčnosť systému. Keďže monitorovaná platforma je aktívne využívaná, musíme byť pre jej testovaní opatrní. V rámci našich testov overíme vizualizáciu a zasielanie upozornení pre dostupnosť fyzických uzlov aj služieb *OpenStack*, vyťaženie zdrojov na serveroch, a podozrivé správanie na VM vo vnútri cloudu. Pre tieto účely sme si v monitorovanej platforme vytvorili projekt s názvom *Sochor_Monitoring_Test* a dve inštancie *Monitor_Tester,* a *Monitor_Tester_2,* na ktoré sme zároveň nainštalovali *node exporter*. Toto potvrdzujú aj dáta z monitorovacej platformy, keďže v obrazovke *Tenants overview dashboard* vieme v rámci premennej *Tenant* navoliť tento projekt, viď **Obrázok 6.22**. Na obrázku taktiež vidíme dve vytvorené inštancie, pričom jedna z nich indikuje stav SHUTOFF, čo korešponduje s realitou, keďže túto VM sme pred vytvorením snímku obrazovky vypli.

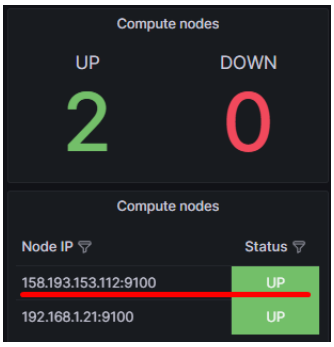


Obrázok 6.22 Overenie projektu *Sochor_Monitoring_Test*

6.9.1 Vyťaženie zdrojov a dostupnosť

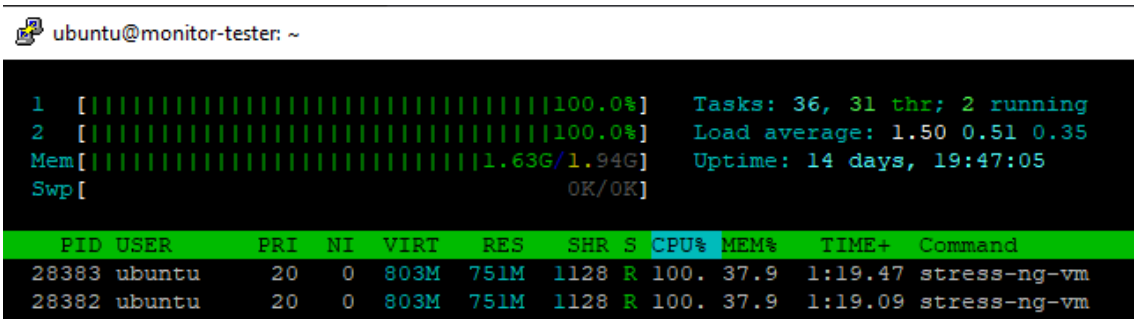
Za najkritickejšiu udalosť v rámci infraštruktúry môžeme považovať výpadok alebo preťaženie zariadenia. Práve preto je na mieste overiť, či nastavené upozornenie funguje, a za aký čas sa dostane k administrátorovi. Taktiež overíme, či nám nastavený panel bude vizualizovať tento výpadok. O dynamických paneloch sme už pojednávali v kapitole 6.7.3, rovnakým spôsobom fungujú aj pravidlá pre všetky zariadenia. Do konfiguračného súboru práce určenej pre výpočtové uzly sme pridali IP adresu inštancie *Monitor_Tester*, ktorá nám bude pre tieto účely simulovať výpočtový uzol. Akonáhle si *Prometheus* načíta konfiguráciu,

na monitorovacej obrazovke sa nám zmenia panely týkajúce sa výpočtových uzlov, viď **Obrázok 6.23**. K pôvodnému uzlu pribudol nový záznam a navýšil sa počet uzlov v stave UP.

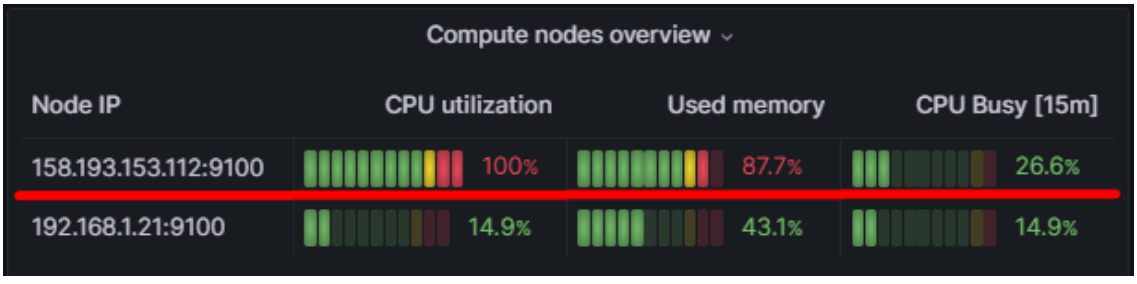


Obrázok 6.23 Pridanie výpočtového uzla

Ako prvé budeme simulovať vyťaženie CPU a RAM. Pre tieto účely využijeme nástroj *stress-ng*, vďaka ktorému dokážeme vyťažiť obidva komponenty naraz. VM *Monitor_Tester* má k dispozícii dva GB RAM a dve jadrá vCPU. Pre zaťaženie sme využili príkaz `#stress-ng --vm-bytes 1500M --vm-keep -m 2 --Timeou 420`, čím sme v systéme vytvorili dve VM pre zaťaženie jadier vCPU. Okrem toho sa v RAM alokovalo tisícpäťsto MB pamäte. Viac informácií o fungovaní nástroja a jeho argumentoch je k dispozícii v manuáli k programu. Testovanie sme spustili o 11:00 lokálneho času. Zaťaženie na systéme nám dokazujú aj informácie z nástroja *htop*, viď **Obrázok 6.24**, ale aj dáta priamo z monitorovacej obrazovky, viď **Obrázok 6.25**.

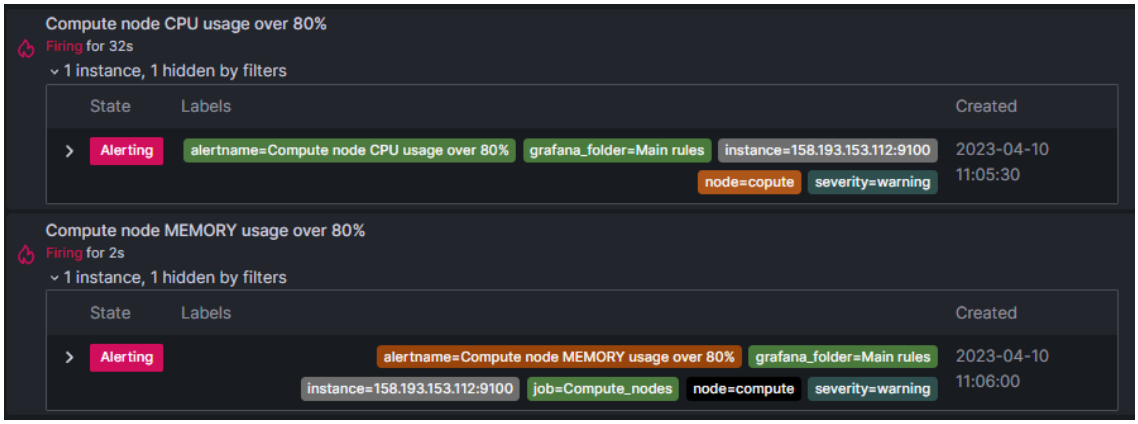


Obrázok 6.24 Vyťaženie systému - *htop*

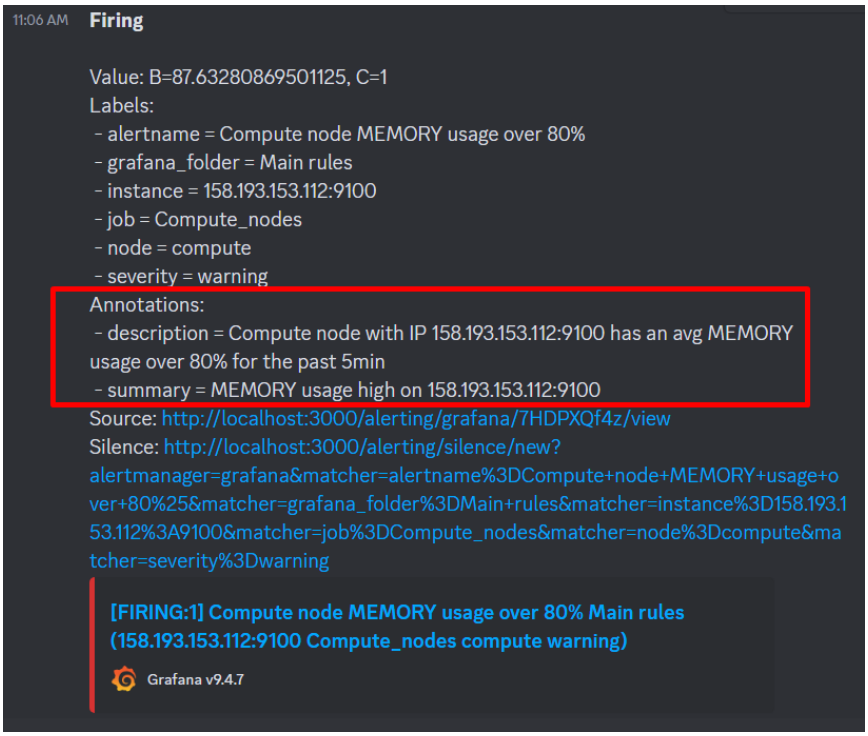


Obrázok 6.25 Vyťaženie systému - Grafana

Na základe nastaveného pravidla pre vyťaženie zdrojov musí priemer využitia za posledných päť minút stúpnuť nad osemdesiat percent. Pri našom testovaní sa upozornenia vytvorili 11:05:30 pre CPU a 11:06:00 pre RAM, čo môžeme vidieť aj na monitorovacej obrazovke, panel pre aktívne upozornenia, pozri **Obrázok 6.26**.



Obrázok 6.26 Vytvorené upozornenia - panel pre aktívne upozornenia

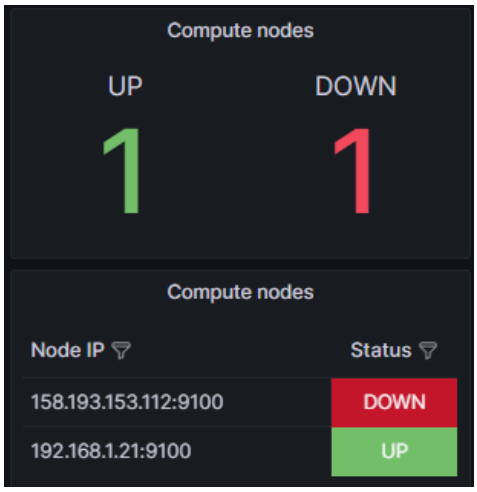


Obrázok 6.27 Vygenerované upozornenie - discord

Administrátori boli upovedomení o udalosti aj odoslaním textových upozornení na nastavené platformy. V prípade platforiem *Discord* a *Teams* sme hlásenia o udalostiach dostali medzi 11:06, a 11:07. Textové upozornenie pre vysoké využitie pamäte zobrazuje **Obrázok 6.27**, pričom vo zvýraznenej časti sa nachádzajú najdôležitejšie informácie typu:

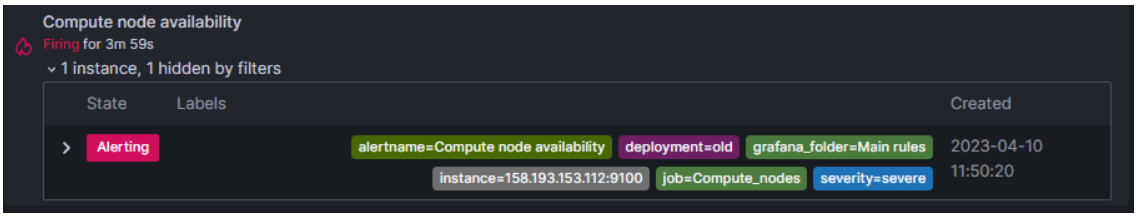
čo, a na akom zariadení sa stalo plus akého komponentu sa to týka. Po ukončení vyťaženia, keď kleslo priemerné využitie pod osemdesiat percent, sa upozornenia automaticky vyriešili a administrátori boli upovedomení správou *Resolved*. Pre prípady, kedy sa upozornenie nerieši dlhšiu dobu, sme nastavili interval opakovaného odosielania textových správ na štyri týždne. V praxi to znamená, že pokiaľ by bolo upozornenie aktívne dlhšiu dobu, správa je odoslaná pri vygenerovaní upozornenia a potom každý štvrtý týždeň. Tento interval je možné zmeniť v nastaveniach notifikačnej politiky, časť *Timing options->Repeat interval*.

Testovanie upozornenia pre dostupnosť budeme simulovať vypnutím VM, pričom budeme sledovať, za aký čas sa vygeneruje upozornenie. Vypnutie VM realizujeme o 11:50 lokálneho času, pričom do pätnástich sekúnd nám panely na obrazovke indikujú nedostupnosť uzla, pozri **Obrázok 6.28**.



Obrázok 6.28 Nedostupnosť uzla - monitorovacia obrazovka

Upozornenie sa vygenerovalo do dvadsiatich sekúnd od vypnutia VM, viď **Obrázok 6.29**, pričom textovú správu sme dostali do jednej minúty od výpadku.



Obrázok 6.29 Upozornenie - výpadok uzla

Po opätovnom spustení VM sa do dvadsiatich sekúnd vyriešilo upozornenie a obrazovka opäť indikovala funkčný stav obidvoch uzlov. Textovú správu o vyriešení problému sme obdržali do desiatich minút od zotavenia.

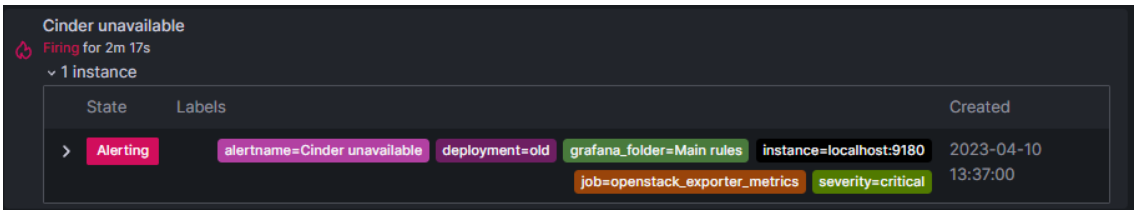
6.9.2 Služby OpenStack

Ďalšou súčasťou, ktorú máme záujem otestovať, je upozorňovanie na výpadok služieb *OpenStack*. V monitorovanom cloude je tento spôsob testovania náročný, keďže vypnutie služby ovplyvní ostatných používateľov. V našom prípade však môžeme využiť službu *cinder*, ktorá nie je pri tomto nasadení plne funkčná. Testovanie vykonáme jej vypnutím a zdokumentujeme proces vizualizácie, a upozornenia.

V rámci služby *cinder* vypíname pomocou `#service openstack-cinder-api stop` jej API súčasť, čím znemožníme akúkoľvek komunikáciu s týmto komponentom. Do jednej minúty od vypnutia nám obrazovka signalizuje problém slovom *DOWN*. Rovnako sa znižuje hodnota dostupnosti služby za zvolené časové okno, v prípade **Obrázok 6.30** posledné tri hodiny. Zároveň sa generuje upozornenie, pozri **Obrázok 6.31**, ktoré do dvoch minút od začiatku výpadku prijímame v textovej forme.



Obrázok 6.30 Výpadok služby cinder

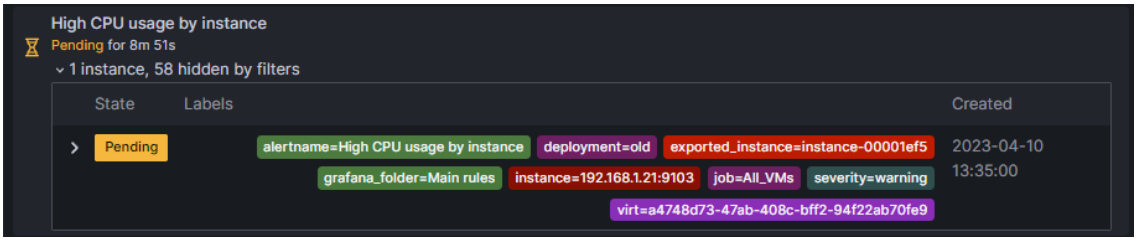


Obrázok 6.31 Upozornenie výpadku služby cinder

6.9.3 Podozrivé správanie VM

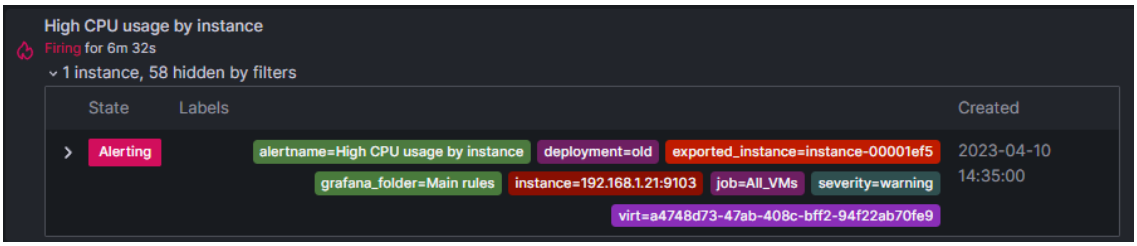
Od administrátorov bola požiadavka vytvoriť taký proces, aby boli informovaní v prípade dlho dobejšieho, vysokého využívania zdrojov inštanciami v cloude, keďže takýto stav môže indikovať zneužívanie poskytnutých zdrojov na nežiadúce účely. Na účely tohto testovania vyťažíme CPU na VM *Monitor_Tester_2* a zdokumentujeme následné reakcie. Na vyťaženie využívame príkaz `#stress-ng --vm-keep -m 2`, čím vyťažíme všetky dve jadrá vCPU. S testom začíname o 13:00 lokálneho času.

Tridsaťpäť minút od začiatku testu sa nám vygenerovalo čakajúce upozornenie, pozri **Obrázok 6.32**, indikátor *Pending*. To vznikne po splnení definovaného pravidla a do aktívneho upozornenia prejde po nastavenej čakacej dobe, pri tomto pravidle po jednej hodine.



Obrázok 6.32 Čakajúce upozornenie vyťaženia inštancie

Pokiaľ počas nastavenej hodiny nedôjde k náprave, vygeneruje sa aktívne upozornenie a odošle na nastavené platformy.



Obrázok 6.33 Aktívne upozornenie vyťaženia inštancie

ZÁVER

Cieľom dokumentu bolo navrhnúť a implementovať monitorovací systém pre cloud postavený na platforme *OpenStack*. V teoretickej rovine riešenia problematiky sme najprv preskúmali rôzne motivácie pre vytvorenie monitorovacieho systému a rozdelili pohľady na monitoring. Do jednej z motivácií zapadalo aj overovanie parametrov definovaných v SLA, preto sme sa venovali tejto problematike a zadefinovali významné metriky v jej rámci. Následne sme identifikovali dôležité výkonové metriky fyzických a virtuálnych zariadení, ktoré by mohli mať vplyv na poskytované služby a objasnili sme spôsob ich získavania. Keďže sme mali záujem monitorovať platformu *OpenStack*, bolo nutné sa s ňou oboznámiť a predstaviť dôležité súčasti pre budúce monitorovanie. K záveru teoretickej časti sme sa venovali vhodným spôsobom prezentovania získaných dát a predstaveniu softvérových nástrojov vhodných na riešenie.

V praktickej rovine sme stanovili požiadavky na riešenie. Na ich základe sme pre nasadenie systému zvolili súpravu nástrojov *Prometheus* a vizualizačnú platformu *Grafana*. Pokračovali sme návrhom architektúry, pozostávajúcej z centrálneho servera pre zber dát a exportérov, zbierajúcich metriky z koncových API bodov a zariadení. Na základe stanovenej architektúry sme systém nasadili. Proces pozostával z inštalácie a konfigurácie jednotlivých komponentov, vytvorenia vizualizačných obrazoviek, a nastavenia upozorňovacích pravidiel, ktoré by dokázali administrátorov upozorniť pri vzniku problémov. Medzi hlavné monitorované parametre sme zaradili využitie CPU, RAM a úložiska na fyzických uzloch, spolu s ich dostupnosťou. Rovnako monitorujeme dostupnosť služieb *OpenStack* a využívanie zdrojov jednotlivými inštanciami v cloude, za účelom ich následnej prezentácie aj užívateľom platformy. Vo finálnej časti sme otestovali funkčnosť nami nasadeného systému, čím sme splnili cieľ tejto práce. Taktiež sme sa v jednoduchosti venovali možnému smerovaniu nami vytvoreného riešenia do budúcnosti.

ZOZNAM POUŽITEJ LITERATÚRY

[1] "What is cloud computing?", 2012. https://aws.amazon.com/what-is-cloud-computing/ (cit jan. 20, 2023).

[2] C. B. Hauser a S. Wesner, "Reviewing Cloud Monitoring: Towards Cloud Resource Profiling", *IEEE Int. Conf. Cloud Comput. CLOUD*, roč. 2018-July, s. 678–685, 2018, doi: 10.1109/CLOUD.2018.00093.

[3] 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.

[4] L. Romano, D. De Mari, Z. Jerzak, a C. Fetzer, "A novel approach to QoS monitoring in the cloud", *Proc. - 1st Int. Conf. Data Compression, Commun. Process. CCP 2011*, s. 45–51, 2011, doi: 10.1109/CCP.2011.49.

[5] 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.

[6] E. Aljoumah, F. Al-Mousawi, I. Ahmad, M. Al-Shammri, a Z. Al-Jady, "SLA in cloud computing architectures: A comprehensive study", *Int. J. Grid Distrib. Comput.*, roč. 8, č. 5, s. 7–32, 2015, doi: 10.14257/ijgdc.2015.8.5.02.

[7] M. Singh, S. Bhushan, a S. Rani, "Investigation of SLA Management in Cloud Computing and Future Directions", *Proc. - 2021 2nd Int. Conf. Comput. Methods Sci. Technol. ICCMST 2021*, s. 78–83, 2021, doi: 10.1109/ICCMST54943.2021.00027.

[8] S. Frey, C. Lüthje, R. Teckelmann, a C. Reich, "Adaptable Service Level Objective Agreement (A-SLO-A) for Cloud services", *CLOSER 2013 - Proc. 3rd Int. Conf. Cloud Comput. Serv. Sci.*, s. 457–462, 2013, doi: 10.5220/0004356604570462.

[9] M. Alhamad, T. Dillon, a E. Chang, "Conceptual SLA framework for cloud computing", *4th IEEE Int. Conf. Digit. Ecosyst. Technol. - Conf. Proc. IEEE-DEST 2010, DEST 2010*, s. 606–610, 2010, doi: 10.1109/DEST.2010.5610586.

[10] N. R. M. (O'Reilly) Betsy Beyer, Chris Jones, Jennifer Petoff, *Site Reliability Engineering*. Google, Inc., 2016.

[11] "AWS EC2 Cloud Watch documentation". https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/viewing_metrics_with_cloudwatch.html (cit feb. 16, 2023).

[12] "AWS EC2". https://aws.amazon.com/ec2/ (cit feb. 14, 2023).

[13] H. Lee, M. Kim, J. Hong, a G. Lee, "QoS parameters to network performance metrics mapping for SLA monitoring", *KNOM Rev.*, č. March, s. 42–53, 2002, [Online]. Available at: http://mail.apnoms.org/knom/knom-review/v5n2/4.pdf.

[14] S. Lee, K. Levanti, a H. S. Kim, "Network monitoring: Present and future", *Comput. Networks*, roč. 65, s. 84–98, 2014, doi: 10.1016/j.comnet.2014.03.007.

[15] J. Ellingwood, "Gathering Metrics from Your Infrastructure and Applications", 2017. https://www.digitalocean.com/community/tutorials/gathering-metrics-from-your-infrastructure-and-applications (cit feb. 27, 2023).

[16] Brendan Gregg, "The USE Method". https://www.brendangregg.com/usemethod.html (cit mar. 06, 2023).

[17] V. Lam, "3 Key Server Monitoring Metrics To Track For System Health And

Performance". https://theqalead.com/test-management/server-monitoring-metrics/ (cit mar. 06, 2023).

[18] "Libvirt Manual Pages". https://www.libvirt.org/manpages/ (cit mar. 18, 2023).

[19] "proc(5) — Linux manual page". https://man7.org/linux/man-pages/man5/proc.5.html (cit mar. 18, 2023).

[20] B. Brazil, "Understanding Machine CPU usage". https://www.robustperception.io/understanding-machine-cpu-usage/ (cit mar. 18, 2023).

[21] "Manual virsh". https://www.libvirt.org/manpages/virsh.html (cit mar. 18, 2023).

[22] "How To Calculate CPU Utilization". https://www.embedded.com/how-to-calculate-cpu-utilization/ (cit mar. 19, 2023).

[23] "Working With Memory Metrics From Node Exporter". https://acloudxpert.com/working-with-memory-metrics-from-node-exporter/ (cit mar. 20, 2023).

[24] "GitHub Server Recommended alert thresholds". https://docs.github.com/en/enterprise-server@3.7/admin/enterprise-management/monitoring-your-appliance/recommended-alert-thresholds (cit mar. 24, 2023).

[25] "Tableau Hardware Monitoring". https://help.tableau.com/current/blueprint/en-us/bp_hardware_monitoring.htm (cit mar. 24, 2023).

[26] D. N, "Server Monitoring Best Practices". https://www.poweradmin.com/blog/monitoring-best-practices-2/ (cit mar. 24, 2023).

[27] "Linux manual". https://man7.org/linux/man-pages/ (cit mar. 23, 2023).

[28] "Check your disk space use with the Linux df command". https://www.redhat.com/sysadmin/Linux-df-command (cit mar. 20, 2023).

[29] "Filesystem metrics from the node exporter". https://www.robustperception.io/filesystem-metrics-from-the-node-exporter/ (cit mar. 23, 2023).

[30] European Telecommunications Standards Institute, "ETSI EG 202 765-3 V1.1.1 Speech and multimedia Transmission Quality (STQ); QoS and network performance metrics and measurement methods; Part 3: Network performance metrics", roč. 1, s. 1–33, 2009.

[31] G. Gardikis, K. Sarsembagieva, G. Xilouris, a A. Kourtis, "An SNMP agent for active in-network measurements", *Int. Congr. Ultra Mod. Telecommun. Control Syst. Work.*, s. 302–307, 2012, doi: 10.1109/ICUMT.2012.6459684.

[32] A. Ciuffoletti, "Monitoring a virtual network infrastructure: An IaaS perspective", *Comput. Commun. Rev.*, roč. 40, č. 5, s. 47–52, 2010, doi: 10.1145/1880153.1880161.

[33] "What is OpenStack?" https://www.openstack.org/software/ (cit feb. 04, 2023).

[34] 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.

[35] "OpenStack nova documentation". https://www.openstack.org/software/ (cit feb. 04, 2023).

[36] "OpenStack swift documentation". https://docs.openstack.org/swift/latest/ (cit feb. 04,

2023).

[37] “OpenStack Monitoring”. https://docs.openstack.org/operations-guide/ops-monitoring.html (cit mar. 16, 2023).

[38] “Nova System Architecture”. https://docs.openstack.org/nova/queens/user/architecture.html (cit mar. 16, 2023).

[39] “Cinder System Architecture”. https://docs.openstack.org/cinder/latest/contributor/architecture.html (cit mar. 18, 2023).

[40] “Networking service overview”. https://docs.openstack.org/neutron/latest/install/common/get-started-networking.html (cit mar. 18, 2023).

[41] “OpenStack telemetry”. https://wiki.openstack.org/wiki/Telemetry (cit apr. 01, 2023).

[42] “OpenStack Monasca”. https://wiki.openstack.org/wiki/Monasca (cit apr. 01, 2023).

[43] H. Aljahdali, A. Albatli, P. Garraghan, P. Townend, L. Lau, a J. Xu, “Multi-tenancy in cloud computing”, *Proc. - IEEE 8th Int. Symp. Serv. Oriented Syst. Eng. SOSE 2014*, s. 344–351, 2014, doi: 10.1109/SOSE.2014.50.

[44] S. Kajeepeta, “Multi-tenancy in the cloud: Why it matters”. https://www.computerworld.com/article/2517005/multi-tenancy-in-the-cloud-why-it-matters.html (cit mar. 09, 2023).

[45] D. Tovarnñák 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.

[46] “CSP vs. tenant - Understanding shared responsibilities”. https://www.kuppingercole.com/blog/reinwarth/csp-vs-tenant-understanding-shared-responsibilities (cit mar. 10, 2023).

[47] “Reporting & Monitoring Overview”. https://developers.google.com/maps/reporting-and-monitoring/overview (cit mar. 09, 2023).

[48] “IBM Cloud Pak System W3500 2.3.2.0”. https://www.ibm.com/docs/en/cloud-pak-system-w3500/2.3.2.0 (cit mar. 11, 2023).

[49] “What is data visualization?” https://www.ibm.com/topics/data-visualization (cit mar. 11, 2023).

[50] “Cloud monitoring”. https://cloud.google.com/monitoring/docs (cit mar. 11, 2023).

[51] “Cloud monitoring guide: Response”. https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/manage/monitor/response#successful-alerting-strategy (cit mar. 13, 2023).

[52] “Deploy Azure Monitor: Alerts and automated actions”. https://learn.microsoft.com/en-us/azure/azure-monitor/best-practices-alerts (cit mar. 13, 2023).

[53] R. Grati, K. Boukadi, a H. Ben-Abdallah, “Overview of IaaS monitoring tools”, *Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl. AICCSA*, roč. 2016-July, 2016, doi: 10.1109/AICCSA.2015.7507146.

[54] “What is Graphite Monitoring?”, 2022. https://www.metricfire.com/blog/what-is-graphite-monitoring/ (cit mar. 25, 2023).

[55] H. J, “Graphite Beginner’s Guide”. https://www.netadmintools.com/graphite-beginners-guide/#wbounce-modal (cit mar. 25, 2023).

[56] “collectd – The system statistics collection daemon”. https://collectd.org/ (cit mar. 26, 2023).

[57] “Grafana”. https://grafana.com/ (cit mar. 26, 2023).

[58] “Zabbix manual”. https://www.zabbix.com/documentation/6.0/en/manual (cit mar. 27, 2023).

[59] D. Lambert, “Multi-tenant monitoring: how to achieve that using free Zabbix open-source monitoring software”. https://blog.zabbix.com/multi-tenant-monitoring-how-to-achieve-that-using-free-zabbix-open-source-monitoring-software/12024/ (cit mar. 27, 2023).

[60] “Prometheus documentation”. https://prometheus.io/docs/introduction/overview/ (cit mar. 27, 2023).

[61] B. Brazil, “How much RAM does Prometheus 2.x need for cardinality and ingestion?”, 2019. https://www.robustperception.io/how-much-ram-does-prometheus-2-x-need-for-cardinality-and-ingestion/ (cit mar. 20, 2023).

[62] “Prometheus node exporter”. https://github.com/prometheus/node_exporter (cit mar. 29, 2023).

[63] “Prometheus snmp_exporter”. https://github.com/prometheus/snmp_exporter (cit mar. 30, 2023).

[64] “openstack-exporter”. https://github.com/openstack-exporter/openstack-exporter (cit mar. 30, 2023).

[65] “Cannonical prometheus-openstack-exporter”. https://github.com/canonical/prometheus-openstack-exporter (cit apr. 02, 2023).

[66] “Grafana documenation”. https://grafana.com/docs/grafana/latest/ (cit mar. 30, 2023).

[67] “Global OID reference database”. https://oidref.com/ (cit apr. 02, 2023).

[68] “circitor MIB files repository”. https://www.circitor.fr/Mibs/Mibs.php (cit apr. 02, 2023).

[69] “Go documentation”. https://go.dev/doc/ (cit apr. 02, 2023).

[70] “Manpage collectd.conf(5)”. https://collectd.org/documentation/manpages/collectd.conf.5.shtml# (cit apr. 02, 2023).

[71] “Understanding Systemd Units and Unit Files”. https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files (cit apr. 01, 2023).

[72] “What is a webhook?” https://www.redhat.com/en/topics/automation/what-is-a-webhook (cit apr. 07, 2023).

[73] “Grafana mimir”. https://grafana.com/oss/mimir/ (cit mar. 20, 2023).

PRÍLOHY

Príloha A: Zoznam MIB súborov pre generátor snmp_exporter

```
sochor@prometheus:/opt/snmp_exporter_generator/generator/mibs$ ls
7.2.50.0.18765.RELEASE-B100-MIB.txt      PAN-LC-MIB.md5
7.2.50.0.18765.RELEASE-CERTS-MIB.txt     PAN-LC-MIB.my
7.2.50.0.18765.RELEASE-IPVS-MIB.txt      PAN-PRODUCT-MIB.md5
7.2.50.0.18765.RELEASE-ONE4NET-MIB.txt   PAN-PRODUCT-MIB.my
AIRESPACE-REF-MIB                        PAN-TRAPS.md5
AIRESPACE-WIRELESS-MIB                   PAN-TRAPS.my
apc-powernet-mib                         PDU-MIB.txt
ARISTA-ENTITY-SENSOR-MIB                 PICO-IPSEC-FLOW-MONITOR-MIB.txt
ARISTA-SMI-MIB                           PICO-SMI-ID-MIB.txt
ARISTA-SW-IP-FORWARDING-MIB              PICO-SMI-MIB.txt
CISCO-MEMORY-POOL-MIB.mib                PRINTER-MIB-V2.txt
CISCO-PROCESS-MIB.txt                    servertech-sentry3-mib
CISCO-QOS-PIB-MIB.mib                    servertech-sentry4-mib
CISCO-SMI.mib                            SNMP-FRAMEWORK-MIB
CISCO-TC.mib                             SNMP-FRAMEWORK-MIB.mib
CyberPower.MIB                           SNMPv2-CONF.mib
ENTITY-MIB                               SNMPv2-MIB
ENTITY-SENSOR-MIB                        SNMPv2-SMI
ENTITY-STATE-MIB                         SNMPv2-SMI.mib
ENTITY-STATE-TC-MIB                      SNMPv2-TC
HCNUM-TC                                 SNMPv2-TC.mib
HCNUM-TC.mib                             SYNOLOGY-DISK-MIB.txt
HOST-RESOURCES-MIB                       SYNOLOGY-EBOX-MIB.txt
IANA-CHARSET-MIB.txt                     SYNOLOGY-FLASHCACHE-MIB.txt
IANA-IFTYPE-MIB.txt                      SYNOLOGY-GPUINFO-MIB.txt
IANA-PRINTER-MIB.txt                     SYNOLOGY-ISCSILUN-MIB.txt
IF-MIB                                   SYNOLOGY-ISCSITarget-MIB.txt
INET-ADDRESS-MIB                         SYNOLOGY-NFS-MIB.txt
Infrapower-MIB.mib                       SYNOLOGY-PORT-MIB.txt
IPV6-TC                                  SYNOLOGY-RAID-MIB.txt
ISDN-MIB                                 SYNOLOGY-SERVICES-MIB.txt
KEEPALIVED-MIB                           SYNOLOGY-SHA-MIB.txt
LIEBERT_GP_PDU.MIB                       SYNOLOGY-SMART-MIB.txt
LIEBERT_GP_REG.MIB                       SYNOLOGY-SPACEIO-MIB.txt
MIKROTIK-MIB                             SYNOLOGY-STORAGEIO-MIB.txt
NET-SNMP-MIB                             SYNOLOGY-SYSTEM-MIB.txt
NET-SNMP-TC                              SYNOLOGY-UPS-MIB.txt
PAN-COMMON-MIB.md5                       UBNT-AirFiber-MIB
PAN-COMMON-MIB.my                        UBNT-AirMAX-MIB.txt
PAN-ENTITY-EXT-MIB.md5                   UBNT-UniFi-MIB
PAN-ENTITY-EXT-MIB.my                    UCD-SNMP-MIB
PAN-GLOBAL-REG-MIB.md5                   VRRP-MIB
PAN-GLOBAL-REG-MIB.my                    VRRPv3-MIB
PAN-GLOBAL-TC-MIB.md5                    WIENER-CRATE-MIB-5704.txt
PAN-GLOBAL-TC-MIB.my
```