FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Vizualizace monitoringu virtuálních serverů pro OpenStack
Student:	Bc. Martin Pavelek
Vedoucí:	Ing. Tomáš Vondra
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

V privátním cloudu OpenStack existuje několik nástrojů pro sledování zátěže virtuálních strojů. Ten nejvíce používaný, Ceilometer, data pouze sbírá, ale nenabízí jejich vizualizaci. Systémy pro správu virtuálních serverů na OpenStacku tudíž převážně nenabízí funkci sledování zatížení.
- Nastudujte problematiku monitorování zátěže v cloudu OpenStack (Ceilometer, Gnocchi, Monasca) a prozkoumejte API těchto nástrojů.
- Srovnejte existující nástroje na kreslení grafů na webových stránkách.
- S použitím technologií Java s REST API na serveru a Javascript u klienta navrhněte a implementujte komponentu realizující vizualizaci monitoringu metrik dostupných u virtuálních strojů (alespoň CPU a I/O na disk a síť).
- Cílem je, aby komponenta byla integrována do prostředí pro správu virtuálních serverů, které bude požadovat vykreslení konkrétních grafů. Neměla by dovolovat uživateli přístup k libovolným metrikám.
- Řešení otestujte i vzhledem ke škálovatelnosti samotných služeb OpenStacku.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 30. ledna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Vizualizace monitoringu virtuálních serverů pro OpenStack

Bc. Martin Pavelek

Katedra softwarového inženýrství
Vedoucí práce: Ing. Tomáš Vondra

7. května 2018

Poděkování

Tímto bych chtěl poděkovat vedoucímu této práce Ing. Tomáši Vondrovi za možnost podnětných konzultací a ochotu při řešení problémů. Dále bych chtěl poděkovat své rodině a v neposlední řadě i přátelům za podporu během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 7. května 2018 .....

Odkaz na tuto práci

Pavelek, Martin. *Vizualizace monitoringu virtuálních serverů pro OpenStack.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato diplomová práce se zabývá problémem monitorování a vizualizace využití prostředků virtuálních serverů pod správou cloud softwaru OpenStack. Autor v úvodu nastiňuje aktuální situaci ve světě cloud computingu a platformě OpenStack. Zmiňuje také, že uživatelé nyní nemají žádnou vizualizaci k dispozici a proč by je dlouhodobé využití prostředků mělo zajímat.

V analýze se věnuje průzkumu technologií pro monitorování, úskalí jednotlivých částí a možnostmi vizualizace dat. Po důkladné analýze byla navržena komponenta a její zasazení do funkčního prostředí.

Výstupem práce je webová komponenta a popis nasazení navazujících projektů, které celkovou funkčnost umožňují. V závěru se autor věnuje vyhodnocení naplnění cílů a možnostem jak řešení dále rozšířit.

Klíčová slova OpenStack, cloud computing, virtualizace, monitorování, REST API, grafy

Abstract

The aim of this master thesis is to assess and analyze the problem of monitoring and visualizing the use of virtual server resources under OpenStack cloud management. In the introduction, the author describes the current situation in the world of cloud computing and OpenStack platform. He also mentions that no visualization is now available to users and why they should be interested in the usage of resources in the long term.

The analysis focuses on the exploration of monitoring technologies, the pitfalls of individual parts and the possibilities of data visualization. After a thorough analysis, the component was designed and put into a functional environment.

The output of the work is a web component and a description of the deployment of follow-up projects that enable the overall functionality. In conclusion, the author evaluates the objectives and opportunities for further expansion of the solution.

Keywords OpenStack, cloud computing, virtualization, monitoring, REST API, graphs

Obsah

Úvod **1**

1 Cíl práce **3**

2 Analýza **5**
2.1 Cloud computing 5
2.2 Co je OpenStack 8
2.3 Jak funguje OpenStack 8
2.4 Současný stav problematiky 11
2.5 Vizualizační nástroje 15
2.6 Analýza požadavků 19
2.7 Modelování případů užití 20

3 Návrh **25**
3.1 Zvolené řešení 25
3.2 Technologie 26

4 Realizace **29**
4.1 DevStack 29
4.2 Problémová funkčnost 30
4.3 Zvolené funkční verze 31
4.4 Nasazení 32
4.5 Implementace 35

5 Testování **47**
5.1 Výkon v běžném provozu 47
5.2 Výkon s velkým počtem instancí 48
5.3 Výkon vzhledem ke granularitě 49
5.4 Mobilní zařízení 49
5.5 Hodnocení výsledků 50

Závěr **53**

Literatura **55**

A Seznam použitých zkratek **59**

B Detailní architektura OpenStacku **61**

C Podrobný popis nasazení **63**
C.1 OpenStack 63
C.2 Port forwarding a donastavení Gnocchi 64
C.3 Aplikace 65

D Obsah přiloženého CD **67**

Seznam obrázků

2.1 Využití cloudu. 6
2.2 Přehled typů cloudových služeb. 7
2.3 Bare metal virtualizace. 9
2.4 Přidaná vrstva OpenStacku 9
2.5 Propojení projektů OpenStack 11
2.6 Monasca monitoring. 12
2.7 Ukázka přehledu objednaných prostředků a nákladů. 14
2.8 Ukázka Grafana. 15
2.9 Ukázka CanvasJS. 16
2.10 Ukázka Google Charts. 17
2.11 Ukázka ZingChart. 17
2.12 Ukázka MetricsGraphics.js. 18
2.13 Ukázka Chartist. 18
2.14 Seznam účastníků. 20
2.15 Use case – Kontrola přístupu. 21
2.16 Use case – Vizualizace zátěže serverů. 22

3.1 Architektura projektu Gnocchi. 27

4.1 Chybný příkaz ke stáhnutí verze Queens dle dokumentace. 30
4.2 Správný příkaz ke stáhnutí DevStacku verze Queens. 30
4.3 Chyba při instalaci DevStacku verze Queens. 30
4.4 Spuštění nové instance z obrazu. 34
4.5 Vypsání sledovaných metrik. 34
4.6 Inicializace Gnocchi a Ceilometru. 35
4.7 Architektura sběru dat pomocí Ceilometru. 36
4.8 Tok dat až k API. 37
4.9 Tok dat z Gnocchi API k uživateli. 38
4.10 Struktura aplikace. 38
4.11 Stránka se seznamem běžících instancí. 40

4.12 Stránka s detailem instance. 41
4.13 Grafy s různou granularitou. 43

5.1 Rozbor času nástrojem Pingdom. 48
5.2 Závislost rychlosti na granularitě. 49
5.3 Zobrazení na mobilním zařízení. 50

B.1 Detailní propojení projektů 62

C.1 Příprava systému a instalace OpenStacku. 63
C.2 Export nastavení pro příkazovou řádku. 64
C.3 Příklad spuštění nové instance. 64
C.4 Aktualizace Gnocchi a Ceilometeru. 65
C.5 Instalace Apache Tomcat 8 66
C.6 Instalace Apache Tomcat 8 rozšíření. 66

Seznam tabulek

2.1 Typické služby využité v OpenStacku. 10
2.2 Srovnání databází časových řad. 13
2.3 Srovnání knihoven ve výběru. 19

4.1 Verze použitých technologií 31
4.2 Potřebné porty a jejich aplikace. 34
4.3 Zajímavé metriky ke sledování. 44

Úvod

Většina lidí dnes využívá služeb veřejných cloudů aniž by si to uvědomovali. Pokud chtějí na počítači či chytrém zařízení zavolat známým, sledovat televizi, sdílet dokumenty či uložit fotografie z dovolené, využívají k tomu cloud poskytovatele dané aplikace. Takový veřejný cloud je celosvětově dostupný a umožňuje využívat stejnou aplikaci z domova, kanceláře či z hotelového pokoje. Za veškerou správu zodpovídá provozovatel dané služby. Jde tak o uživatelsky pohodlné řešení bez starostí.

Cloud computing je společnostmi hojně využíván ke službám jako je odesílání emailů, úprava dokumentů, ukládání souborů či vytváření celých aplikací a služeb. V případě privátního cloudu jde o využití výpočetních prostředků přes síť, se správou pouze vybraným uživatelům a využívání pro vnitřní účely či aplikace poskytované široké veřejnosti. Vnější uživatel ale nemá možnost manipulovat s prostředky cloudu či si vytvářet vlastní servery. Je tak vhodný pro nasazení vnitřních aplikací sloužící zaměstnancům společnosti. Firmám poskytuje mnoho výhod veřejného cloudu, včetně škálovatelnosti a elasticity, spolu s dalšími možnostmi řízení a přizpůsobení hostované výpočetní infrastruktury, dostupnými přes internet.

Termín cloud computing označuje dodávání výpočetních služeb, jako jsou servery, úložiště, databáze, sítě, software, analytické nástroje a další přes internet neboli cloud. Společnosti nabízející tyto výpočetní prostředky se nazývají poskytovatelé cloudu a obvykle se za služby cloud computingu platí na základě jejich využití. Zákazník tak zaplatí za množství využitých jednotek procesoru, paměti a úložiště. Nespornými výhodami cloud computingu jsou nízké náklady na provoz, flexibilní a rychlé nasazení, vysoký výkon a spolehlivost.

Všechny typy cloudů jsou většinou provozovány v datacentrech (poskytovateli cloudu), která poskytují vysokou úroveň zabezpečení a pracují na eliminaci rizik. Správcem infrastruktury je tak většinou provozovatel datacentra a z hlediska společnosti odpadá nutnost údržby. Ze zmíněných důvodů vyplývá, že jejich popularita bude stále narůstat.

S rozšiřováním prostředí se také zvyšuje poskytovatelům cloudů náročnost

správy velkého množství těchto virtuálních serverů. Vznikla tedy softwarová vrstva, která pomáhá spravovat řady výpočetních uzlů cloudů. Zástupcem světově využívaného softwaru (operačního systému) pro vytváření takových cloudů je OpenStack. Po nasazení v datovém centru může spravovat řadu výpočetních a úložných prostředků, které může nastavovat a přirazovat administrátor virtuálním strojům pomocí dashboardu ve webovém rozhraní.

Zákazník využívající cloud si zřídí účet, zaplatí a následně pracuje ve webovém rozhraní OpenStacku, kde si spustí virtuální stroje a nakonfiguruje další chování sítí či přístupů, jak by tomu bylo u fyzických zařízení. Má informace o množství vyhrazených kvót a svých nákladech. Bohužel ale nemá přehled o reálném využití prostředků. Je tak pouze na něm a jeho vlastních silách, aby dokázal zhodnotit, zda je využitelnost prostředků ideální nebo zda je jeho virtuální stroj dlouhodobě přetížen či nevyužit.

Dostupné systémy, ale převážně nenabízí funkci sledování zatížení virtuálních serverů, data pouze sbírají, ale neposkytují jejich vizualizaci uživateli. Cílem této práce je tedy prostudovat možnosti monitorování zátěže virtuálních serverů v OpenStacku a realizovat komponentu, integrovanou do správy virtuálních serverů zobrazující grafy využití jednotlivých strojů.

Kapitola 1

Cíl práce

Cílem práce je zjistit možnosti monitorování zátěže virtuálních serverů na platformě OpenStack a vytvořit komponentu do správy virtuálních serverů do zákaznického portálu, která umožní zobrazit grafy využití jednotlivých strojů. Výsledná komponenta bude integrována do prostředí správy a umožní zobrazení vybraných metrik o systému.

Dílčími cíli bude zprovoznění projektů zodpovědných za sběr dat o instancích a jejich efektivní ukládání. Dále průzkum grafových knihoven a implementace aplikace na straně serveru, která bude na vyžádání poskytovat data pro vykreslení touto knihovnou.

Přihlášeným uživatelům bude komponenta dostupná přes webové rozhraní. Uživatel nebude mít možnost komunikace přímo s rozhraním úložiště.

Kapitola 2

Analýza

2.1 Cloud computing

Obecný pojem cloud popisuje širokou škálu služeb a způsobů práce s nimi. Cloud computing je termín označující dodávání výpočetních služeb, jako jsou servery, úložiště, databáze, sítě, software, analytické nástroje a další, přes internet („cloud“). Společnosti nabízející tyto výpočetní prostředky se nazývají poskytovatelé cloudu a obvykle se za služby cloud computingu platí na základě jejich využití. Zákazník tak zaplatí za množství využitých jednotek procesoru, paměti a úložiště.[1]

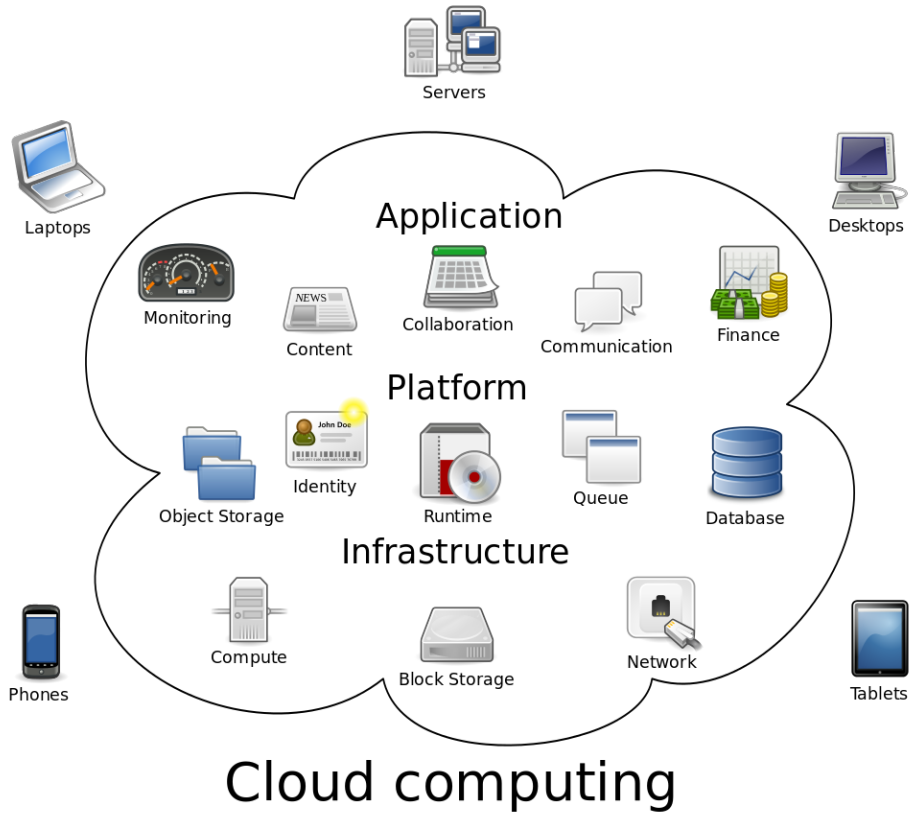
Cloud computing se běžně využívá v online službách jako je odesílání emailů, úprava dokumentů, sledování videí, hraní her, ukládání souborů či vytváření celých aplikací a služeb. Díky širokým možnostem použití jej dnes využívají firmy od nejmenších až po globální korporace či státní organizace.

Od tradičního pojetí ICT v organizacích představují cloudové technologie velkou změnu. Vyžadují trochu jiný přístup od správců a je klíčově důležité si s poskytovatelem dobře domluvit podmínky. Při následném využívání má cloud oproti vlastní spravované infrastruktuře nesporné výhody. Jenže cloud není stejně jako většina moderních technologií samospasitelný, ale je vždy dobré zvážit jeho nasazení pro konkrétní podmínky ve firmě a pro daný účel.[2]

Jak lze vidět na obrázku 2.7, možnosti využití jsou velmi široké a cloud je využitelný pro velkou škálu aplikací jak pro osobní použití, tak ve firmách. Dá se říci, že jakýkoli systém lze umístit do cloudu a provozovat jej tam bez citelného dopadu na uživatele.

2.1.1 Výhody využití cloudu

1. Náklady – Díky cloud computingu nejsou nutné velké investiční výdaje pro nákup celé infrastruktury. Dále zákazník platí pouze podle velikosti vyhrazených zdrojů.



Obrázek 2.1: Příklad široké škály využití cloudů.[3]

2. Flexibilita – V případě, že je potřeba zvýšit výpočetní výkon či paměťové prostředky, většinou stačí pouze požádat poskytovatele. Prostředky jsou ihned přiděleny. Stejně v případě žádosti o snížení. Případně může škálování prostředků probíhat automatizovaně.

3. Bezstarostná správa – Údržba serverů vyžaduje mnoho dalších pracovních zásahů, jako je nastavení hardwaru, softwaru či záplatování. Umístění služeb do cloudu zbaví zákazníka většiny těchto činností.

4. Spolehlivost – V případě problému je mnohem jednodušší zálohování dat, zotavení systémů a udržení kontinuity firemních procesů. Je to proto, že data lze zrcadlit na více redundantních míst na straně poskytovatele.

2.1.2 Nevýhody využití cloudu

1. Nejisté místo uložení – Data mohou být uložena na více geografických polohách a případně v zahraničí. To může znamenat i rozdílné právní

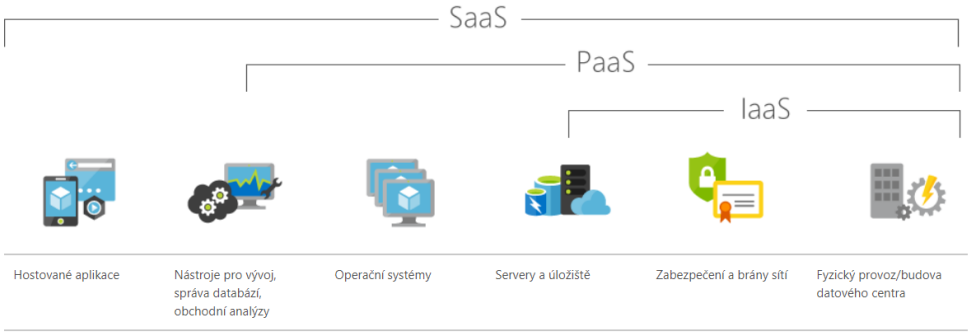
ustanovení při práci poskytovatelů a riziko zneužití.

2. Bez možnosti fyzického zásahu – Je nutné na poskytovatele spoléhat a to i v případě problému. Poskytovatel může reagovat se zpožděním na výzvy.

2.1.3 Typy cloudových služeb

Poskytované služby cloud computingu se dělí do tří kategorií, přičemž závisí jedna na druhé. Výběr správné úrovně je důležitý pro splnění požadavků, které má zákazník.[1]

- IaaS (Infrastructure as a Service) – Nejzákladnější úroveň v cloud computingu. Zákazník si pronajímá infrastrukturu jako jsou servery, virtuální počítače, úložiště či sítě.
- PaaS (Platform as a Service) – V této úrovni poskytovatel zaručuje kompletní prostředí pro vývoj a nasazení aplikací. Jde o operační systémy či běhová prostředí.
- SaaS (Software as a Service) – Na nejvyšší úrovni jde o dodávání celých aplikací. Uživatelé se na základě předplatného k aplikaci pouze připojují a využívají ji. Například email, podnikový systém či chatovací program.



Obrázek 2.2: Přehled typů cloudových služeb.

2.2 Co je OpenStack

OpenStack je software pro správu výpočetních, úložných a síťových prostředků v datových centrech. Je ovládán přes webové rozhraní nebo pomocí OpenStack API[1]. Open source[2] software, kterým OpenStack je, umožňuje uživatelům nahlížet do fungování a volně si software upravovat podle potřeby svého využití, či doplňovat své přídavné moduly.

Projekt byl vytvořen v roce 2010 společnými silami Rackspace a Anso Labs (NASA). Cílem bylo vytvořit cloud platformu, která splní požadavky veřejných i privátních cloudů a bude bez ohledu na velikost jednoduchá k implementaci a masivně škálovatelná.[4] Dnes je využívána po celém světě stovkami společností. (NASA, CERN, NSA, AT&T,...)

2.3 Jak funguje OpenStack

2.3.1 Virtualizace

Pro cloud computing je virtualizace typická. Ta umožňuje, aby na jednom fyzickém serveru (jednom hardware) běželo více oddělených serverů s vlastním operačním systémem. Fyzický server každému takovému virtuálnímu serveru emuluje virtuální hardware (procesor, paměť, disk, síťová karta, mechaniky, periferní zařízení a další).

To, že je server virtualizovaný, však zákazník na první pohled nepozná. Má svůj server s procesory, pamětí a dalšími komponentami, na tom mu běží nějaký operační systém dle jeho volby, má k němu plný přístup a pracuje s ním jako kdyby tento jeho operační systém běžel na vlastním hardware.

Celé je to samozřejmě o penězích – virtualizace oproti klasickému řešení přináší obrovské finanční úspory. Hlavní výhodou jsou samozřejmě peníze a poté snazší správa. Týká se to pořizovacích i provozních nákladů. Cílem je, aby více virtuálních serverů společně sdílelo fyzické prostředky.[5]

2.3.2 Škálovatelnost

Hypervisor[3] poskytne potřebnou virtualizaci a umožní proměnit jeden fyzický stroj na několik virtuálních. Není problém provozovat takto například 10 nezávislých serverů na čtyřech počítačích.

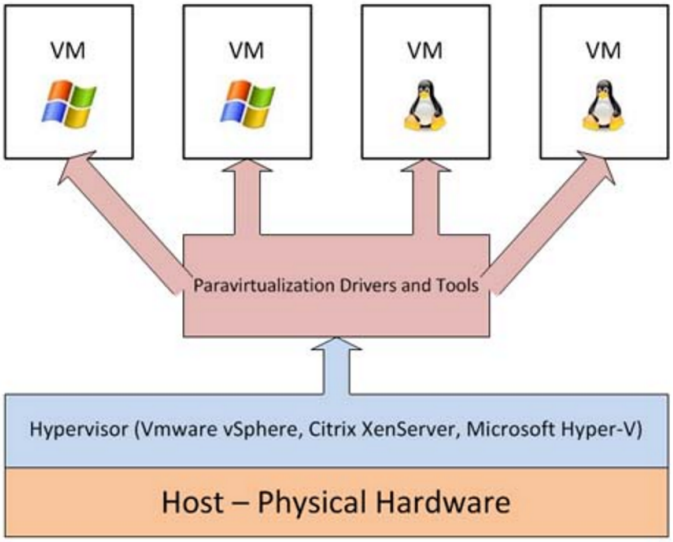
OpenStack volitelně obsahuje projekt Ironic, který využívá *bare metal* hypervisor. To znamená, že mezi hypervisorem a fyzickým hardwarem není žádná přidaná vrstva operačního systému. Hypervisor má tedy přímý přístup k hard-

[1] API – Application Programming Interface – Rozhraní pro programování aplikací.

[2] Open source – Celý kód je volně k dispozici pro čtení a úpravy.

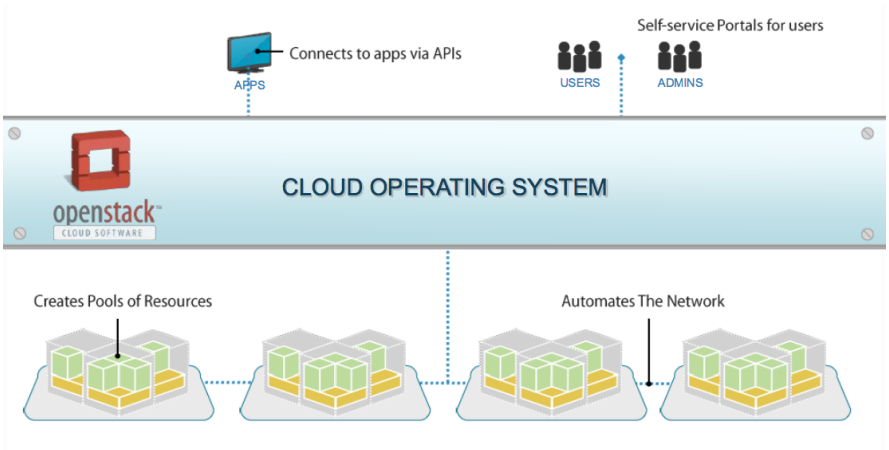
[3] Hypervisor – Software pro uskutečnění virtualizace hardwaru počítače. Umožňuje na jednom počítači spustit zároveň více operačních systémů.

waru a jsou možné i výpočetní úlohy, které by nemohly být virtualizovány.[6] Fungování je zobrazeno na obrázku 2.3.



Obrázek 2.3: Bare metal virtualizace.[7]

S rozšiřováním prostředí se ale pojí zvyšující se náročnost správy velkého množství virtuálních (i fyzických) serverů. Je zapotřebí vrstva, která by cloudy pomohla spravovat. To je právě příležitost a cíl platformy OpenStack.



Obrázek 2.4: Přidaná vrstva OpenStacku umožňuje nová využití.[8]

2.3.3 Architektura

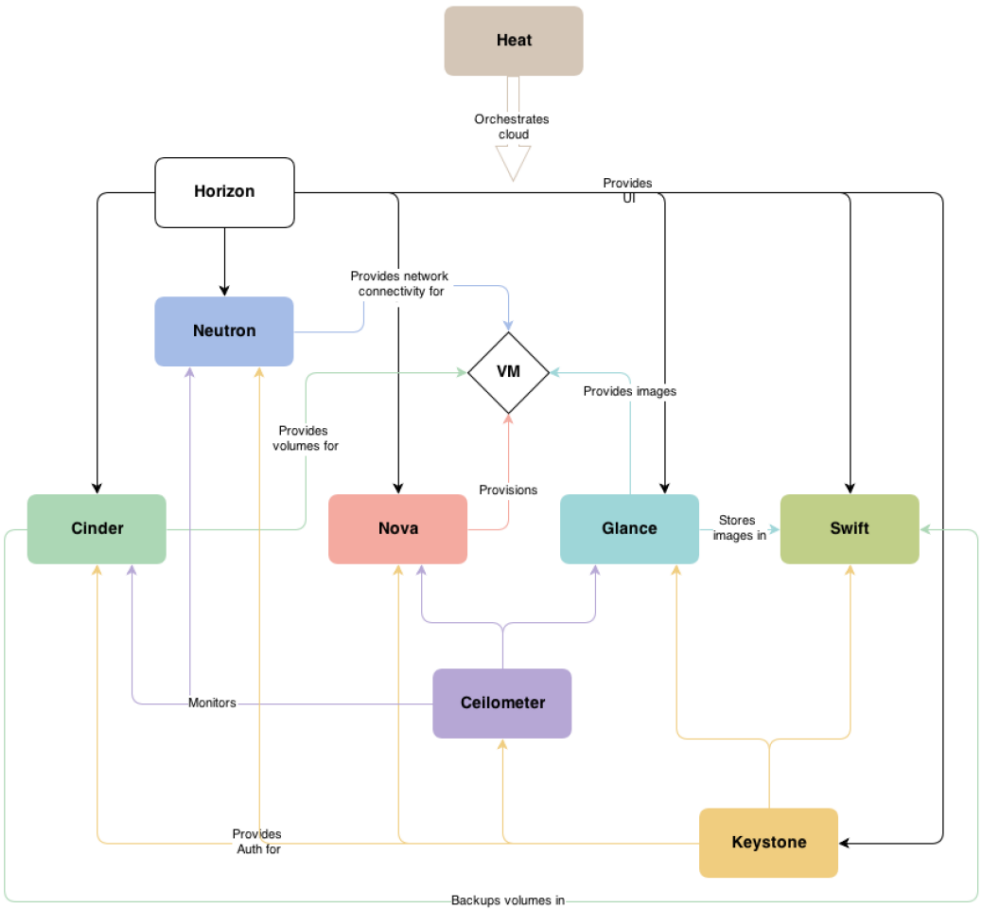
OpenStack poskytuje řešení řadící se do cloudu typu IaaS. Skládá se z množiny navzájem propojených služeb (projektů). Každá nabízí API, které usnadňuje integraci. Každý projekt poskytuje jinou funkčnost a tak pro různé využití cloudů a tedy samotného OpenStacku je potřeba zvolit různé projekty. Například cloud pro webové aplikace nemusí nutně obsahovat projekt Sahara určený pro big data procesování, ale spíše využije projekt Trove, poskytující Database as a Service. V tabulce jsou uvedeny nejvyužívanější projekty a popsány jejich přínosy.

Tabulka 2.1: Typické služby využité v OpenStacku.[9]

Název služby	Projekt	Popis poskytovaných služeb
Dashboard	Horizon	Webové GUI umožňující interagovat se všemi službami pomocí prohlížeče.
Identity	Keystone	Autorizace a autentizace všem službám.
Compute	Nova	Výpočetní služby. (Práce s hypervizory.)
Object storage	Swift	Ukládání souborů, replikace v clusteru.
Block storage	Cinder	Persistentní blokové úložiště.
Image	Glance	Registr obrazů operačních systémů.
Networking	Neutron	Síťová konektivita.
Monitoring	Ceilometer	Monitorování metrik.
Orchestration	Heat	Orchestrace cloud aplikací.

Na obrázku 2.5 je viditelné vzájemné propojení projektů v nasazeném OpenStacku. Je zde mimo jiné vidět, že Horizon je schopný ovládat jiné projekty. Také je důležité si povšimnout, kolik okolních projektů využívá samotný virtuální server.[4] Detailněji prostudovat architekturu je možné například z obrázku v příloze B, který zobrazuje i části zmíněných projektů.

[4]Na diagramu jako VM – virtual machine = virtuální stroj



Obrázek 2.5: Propojení zmíněných projektů v OpenStacku.

2.4 Současný stav problematiky

2.4.1 Technické pozadí

Ceilometer

Cloudový software OpenStack využívá vlastní monitorovací nástroj, tím je Ceilometer. Tento projekt je schopný kompletního monitorování instancí či obrazů. Umožňuje zaznamenávat velikosti přiřazených prostředků, reálné využití či velikost místa, které zabírají zálohy. To vše automaticky v průběhu času.

Dříve tedy měl za cíl ukládat časové řady a sbírat data. Nebylo přesně jasné, jak s nimi manipulovat a dotazovat se. Datový model byl tak velmi flexibilní. To bylo jednoduché při používání, ale vedlo k mizernému výkonu (terrible performance [10]) až do situace, kdy data obsahující informace o pár

týdnech bylo těžké ukládat bez kolapsu storage backendu[5].

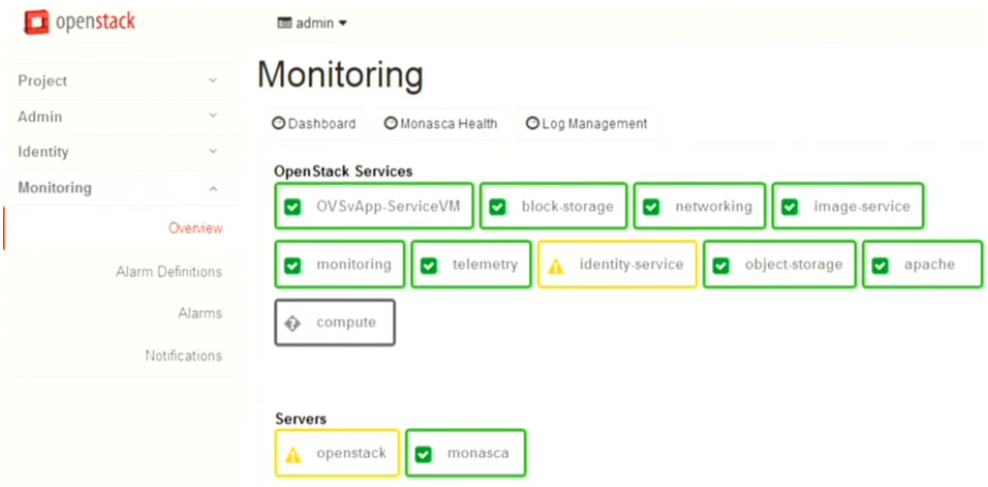
Flexibilní model je tedy výborný pro obecné použití, ale uživatelé chtěli hlavně jeden dotaz pořád dokola a případy užití byly jen malou podmnožinou datového modelu nebo byly příliš pomalé na běh (a tudíž nepoužitelné, jako by neexistovaly).

Sledování metrik

Pro sledování metrik o virtuálních strojích je možné využít různé projekty či jejich spojení. V předchozí kapitole jsou udány důvody, proč by se práce s metrikami neměla nechávat pouze na projektu Ceilometer.

Monitorovací nástroje Monasca a Gnocchi

Výběr navazujícího projektu je důležitý pro implementační záměr a cíl práce. Pro pochopení výběru je nutný průzkum funkcí a zaměření, jelikož oficiální popis je příliš stručný.



Obrázek 2.6: Ukázka přehledu stavu služeb v projektu Monasca.[11]

Projekt Monasca je velmi mladý a jeho cílem je především centrální správa logů a alarmovací systém. Monitorovací panel je přidán do prostředí Horizonu. Je perfektní pro snadné sledování všech součástí komplexního systému skládajícího se z více projektů.[12]

Každá komponenta OpenStacku je totiž schopna ukládat do logu informace o dění. Pokud je našim cílem eliminace budoucích problémů a včasné varování, Monasca nabízí přehledný panel, v jakém stavu se součásti nacházejí a škálu možností, jak se nechat upozornit na libovolnou nastálou událost. Přehled, který Monasca integruje do dashboardu Horizon je na obrázku 2.6.

[5]Storage backend – Datové úložiště ve formě souborů na disku nebo databáze.

Monasca umožňuje i sledování metrik o fyzickém serveru, úložiště pro sledovaná data a REST API. Bohužel, ale stav dokumentace nedovoluje dobré nahlédnutí do dalšího využití. Nejlepší řešení je napojení na vizualizační nástroj Grafana. Toto spojení může nabídnout přehled všech metrik v grafech. Jenže trpí stejnými bezpečnostními problémy Grafany jako napojení Gnocchi. Více v sekci o nástroji Grafana 2.5.1.

Projekt Gnocchi není sice oficiálním dílem OpenStacku, ale je společně s Ceilometrem typický jako uskupení zvané *Telemetry Service* – služba pro monitorování provozu. Je přímo určen ke sledování metrik, nabízí propracované REST API, aktualizovanou dokumentaci a především plnou funkcionalitu pro sledování virtuálních strojů. Více v sekci 3.2.2.

Je tedy jasnou volbou projekt Gnocchi pro vývoj vlastní komponenty v tomto projektu. Monasca přináší výhody pro administrátory serverů a infrastruktury.

2.4.2 Alternativy k Gnocchi

Gnocchi samozřejmě není jedinou databází, která si poradí s časovými řadami. Každá má svá úskalí. V tabulce 2.2 je přehled dostupných open source databází a srovnání jejich vlastností. Za zmínku stojí i databáze Graphite, ale ta kromě podpory Grafany nenabízí žádné výhody.

Tabulka 2.2: Srovnání databází časových řad.[13]

Vlastnost	Gnocchi	Prometheus	InfluxDB	OpenTSDB
Sběr metrik	Ne	Ano	Ne	Ne
Historie zdrojů	Ano	Ne	Ne	Ne
Více nájemníků	Ano	Ne	Ne	Ne
Rozhraní	REST API	REST API	HTTP	TCP
Vysoká dostupnost	Ano	Ne	Replikace	Ano
Škálovatelnost	Ano	Ne	Komerční	Ano
Upozorňování	Ne	Ano	Kapacitor	Ne
Podpora Grafany	Ano	Ano	Ano	Ano
Podpora collectd	Ano	Ano	Ano	Ano

Ze vzájemného porovnání vychází projekt Gnocchi skutečně nejlépe pro daný účel. Oproti ostatním má výhody v podpoře více uživatelů (*nájemníků*), zanechává historii zdrojů a oproti některým podporuje plnou horizontální škálovatelnost. Jediný detail, který by mohl uživateli chybět je služba upozorňování, která se uvažuje do budoucna.

Pomocí REST API může Gnocchi komunikovat a provádět všechny úkony. Oproti ostatním je také klasickým řešením pro telemetrii v OpenStacku.

2.4.3 Firemní pohled

Práce je vytvořena pro nasazení ve firmě 2V IT s.r.o. (společnost Home at Cloud), která se zabývá poskytováním ICT infrastruktury. Poskytován je Public cloud typu IaaS, provozován na otevřené platformě OpenStack. Zákazník si zřídí účet a následně pracuje ve webovém rozhraní OpenStacku, kde si spustí virtuální stroje a nakonfiguruje další chování sítí či přístupů.

Zákazník má ve svém účtu údaje o množství vyhrazených kvót a má tedy denní přehled o svých nákladech. Prostředky jsou účtovány hodinově a náklady jsou tedy za skutečně rezervované zdroje. Pokud si tedy zákazník spouští server pouze na pár hodin denně, pocítí to při platbě.

Obrázek 2.7: Ukázka přehledu objednaných prostředků a nákladů na provoz.

Ale ať už se jedná o jednu běžící instanci nebo stovky, je na samotném zákazníkovi, aby sledoval využití svých prostředků virtuálními stroji. Je tedy jeho vlastním zájmem vědět, zda virtuální stroj většinu dne beží nevyužitý s velkými přidělenými prostředky nebo naopak je na pokraji svých možností a hrozí přetížení.

Uživatel se tak musí sám rozhodovat, kolik prostředků bude potřebovat a zda je s aktuálním stavem spokojený. Monitorování a vizualizace je tedy na něm. Běžné nástroje (například Top[6]) umožňují vizualizaci prostředků v reálném čase, ale bez pohledu do historie. Zákazníka zajímá i dlouhodobý přehled využití, bez nutnosti instalace náročných nástrojů.

Ovšem každé řešení, které si zákazník vytvoří sám, vyžaduje nejen jeho iniciativu při řešení, ale také pokles výkonnosti jeho objednaných strojů nebo nutnou investici do dalších prostředků.

V případě řešení poskytovatelem cloudových služeb by bylo řešení dostupné všem bez rozdílu a bez vyžadování dalších kroků ze strany zákazníka.

[6]Top – Program pro Linux zobrazující zátěž počítače (doba běhu, počet přihlášených uživatelů), statistiku procesů (počet aktivních a běžících procesů), využití paměti, procesoru a v neposlední řadě také seznam procesů.[14]

2.5 Vizualizační nástroje

2.5.1 Grafana

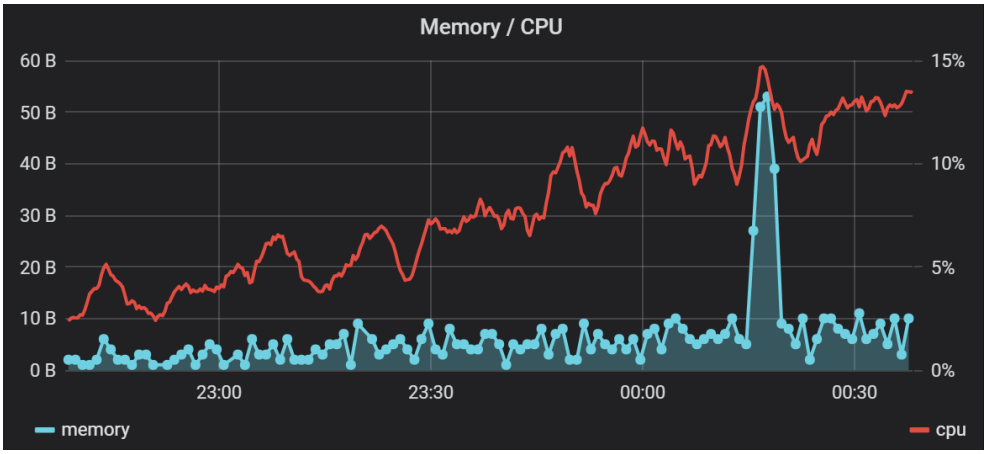
Teoreticky snadným a přímočarým řešením se může zdát Grafana. Otevřená platforma připravena pro napojení na různé datové zdroje a vytváření přehledných grafů. Může být hostována výrobcem na jejich hardwaru nebo instalována a provozována na vlastním serveru.

Pro Gnocchi jako zdroj dat existuje doplněk do Grafany. Ovšem nese s sebou několik nevýhod. Nejvýraznější je bezpečnost a oddělení uživatelů. Grafana totiž využívá jedno nastavení datového zdroje pro všechny uživatele. Je tedy možné vytvořit uživatelům v programu Grafana různá přístupová práva pro různé pohledy, ale není možné uživatele omezit, aby si z datového zdroje zobrazil jakákoli data a tedy případně i ta, která by mu měla být nedostupná.[15]

Mimo jiné Grafana sice umožňuje vkládání grafů do stránky, ale pouze pro předem připravené a publikované grafy. Je tedy možné dynamicky, pomocí skriptu automaticky vytvářet pohledy, ale už je nelze automaticky publikovat a vkládat do vlastních stránek.[16]

V neposlední řadě Grafana neposkytuje potřebnou flexibilitu při škálování Gnocchi. Pokud je použit Keystone, pro dotazování na přístupový token je nutné, aby i služba Gnocchi byla hostována na stejné HTTP adrese.[17]

Z těchto klíčových důvodů Grafana nemůže být zvolena jako technologie řešení práce.



Obrázek 2.8: Ukázka Grafana.[18]

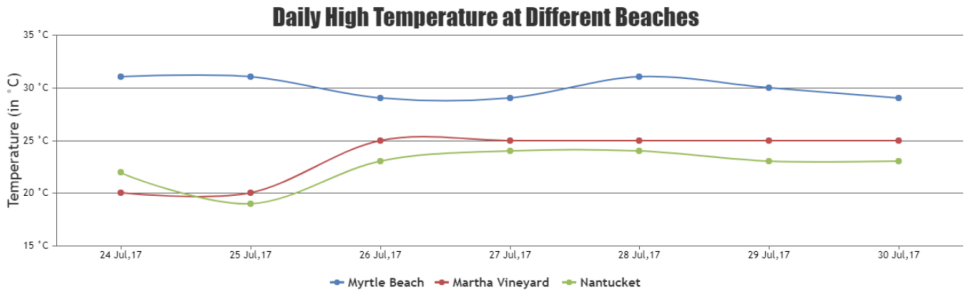
2.5.2 Nástroje pro kreslení grafů

V případě zvolení předávání dat k vykreslení grafu klientovi, jsou využívány Javascriptové knihovny. Existuje nepřeberné množství a je těžké zvolit jedinou, jelikož technicky jsou velmi podobné a jde většinou o subjektivní pocit z výsledku. Případně je důležité promyslet výšku zamýšlené investice.

CanvasJS

Tato knihovna sází na jednoduché rozhraní, skvělý výkon vykreslování pomocí elementu canvas a příkladným pojetím dokumentace. Výsledné grafy jsou působivé a mají širokou škálu přizpůsobení.

Je možné jej používat zdarma pro osobní použití, pro firmy je placená varianta. Pokud se vývojář ztotožní s vysokou pořizovací investicí, především díky perfektní dokumentaci a podpoře jde o skvělou volbu.

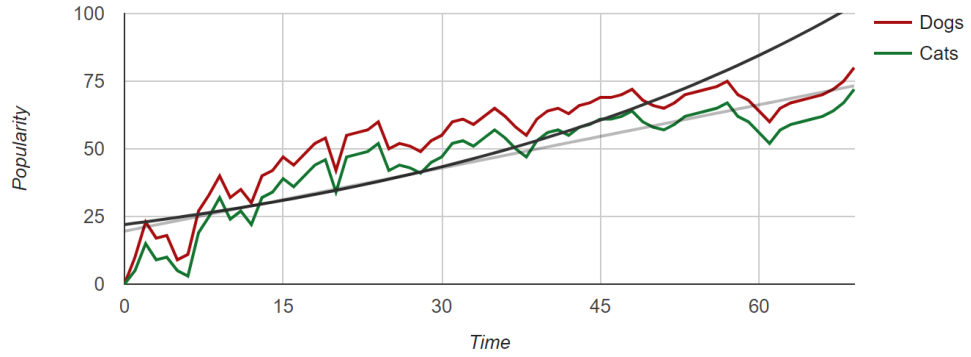


Obrázek 2.9: Ukázka CanvasJS.[19]

Google Charts

Knihovna od firmy Google poskytuje graficky velmi jednoduché a přehledné grafy. Knihovna má nádech minimalismu, je výkonná a úplně zdarma. Má širokou škálu nastavení a vzhled podporuje nastolený řád aplikací Google a aktuální Material Design používaný i pro zařízení Android.

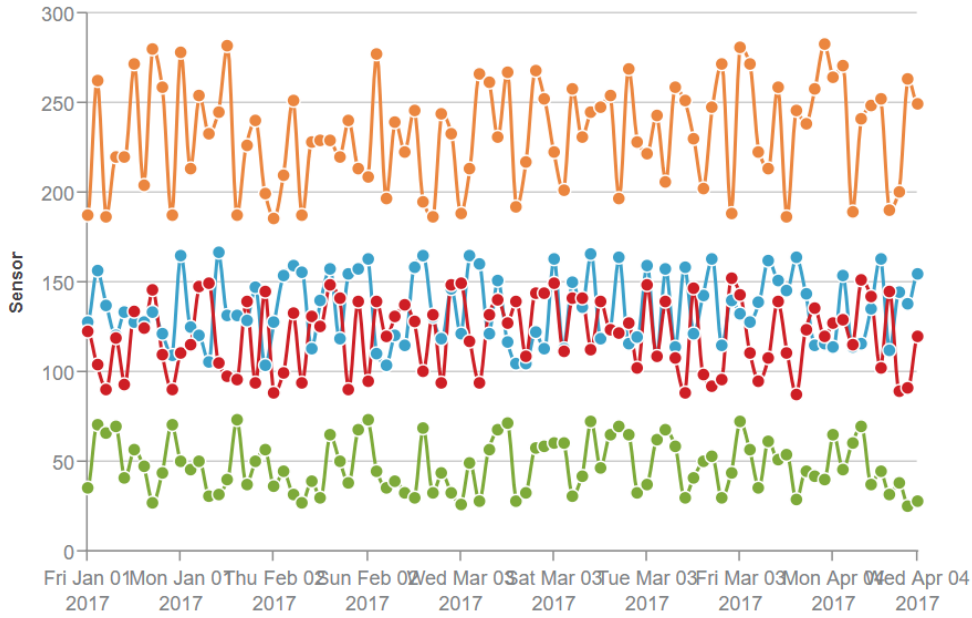
Využívá SVG a VML renderování a je podporována všemi moderními prohlížeči. Pokud by vývojář trval na bezplatném řešení, jde nejspíše o nejlepší volbu.



Obrázek 2.10: Ukázka Google Charts.[20]

ZingChart

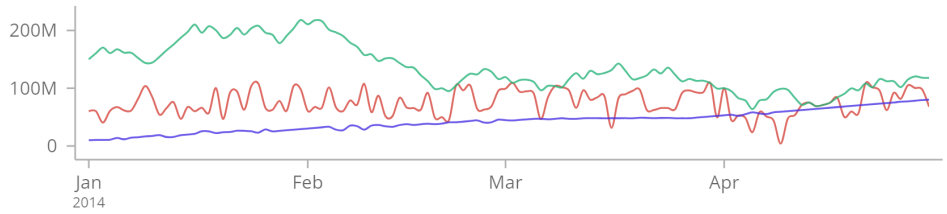
ZingChart je knihovna s širokým spektrem přizpůsobení. Je možné si s výslednými grafy velice pohrát a přizpůsobit si je do detailu, podle svých představ. Také se může pochlubit renderováním pomocí elementu canvas a podporou starších prohlížečů. V případě použití zdarma bude v grafu malý vodoznak.



Obrázek 2.11: Ukázka ZingChart.[21]

MetricsGraphics.js

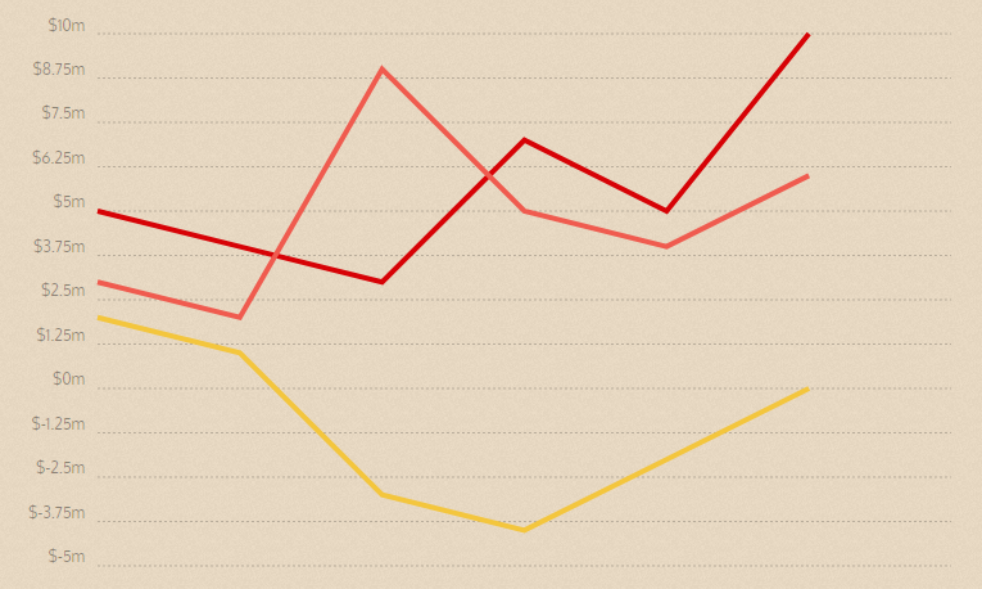
Jednoduchá knihovna zaměřující se na grafy vykreslující metriky o mnoha záznamech. Vhodná pro rychlý start a výkon, bohužel možnosti přizpůsobení jsou velmi malé. Je k dispozici zcela zdarma.



Obrázek 2.12: Ukázka MetricsGraphics.js.[22]

Chartist

Dalším zástupcem volně dostupné knihovny lehkého pojetí je Chartist. Chlubí se svou malou velikostí zdrojů a jak název napovídá, snaží se přidat i jakýsi umělecký prvek. Může jít o animace, rozdělení grafů, pohyby, přidání ikon a další. Nevýhodou může být nutnost připojení pluginů pro přidání dalších možností.



Obrázek 2.13: Ukázka Chartist.[23]

2.5.3 Výsledné srovnání

Tabulka 2.3: Srovnání knihoven ve výběru.

Knihovna	Zdarma	Dokumentace	Umístění
CanvasJS	ne	Příkladná	1.
Google Charts	ano	Dobrá	2.
Chartist	ano	Dobrá	3.
Zing Chart	ne	Dobrá	4.
MetricsGraphics.js	ano	Průměrná	5.

Jak již bylo řečeno, největším rozdílem je cena a subjektivní pohled. Pokud vývojář zvolí placené řešení, může se spolehnout na podporu. Ovšem placené knihovny nabízejí i trial verzi nebo verzi úplně zdarma pro studenty. Volba ale není nezvratným krokem. Jde o využitou externí knihovnu a je tedy možné nahradit ji jinou a transformovat data do podoby požadované knihovnou.

2.6 Analýza požadavků

Tato sekce definuje funkční i nefunkční požadavky pro výsledné řešení. Specifikace požadavků umožnuje popsat očekávanou funkčnost, kterou má systém splňovat, vyjasní zadání a zachytí omezení kladená na výsledné řešení. Požadavky vyplynuly ze zadání práce a rozhovorů s vedoucím práce.

2.6.1 Funkční požadavky

F1 – Vizualizace zátěže virtuálních strojů

Výsledné řešení poskytne data ve vizualizované formě, prezentované ve formě grafů. Uživateli bude umožněno zkoumat vybrané metriky aktivních virtuálních strojů. Uživatel nebude mít na výběr styl grafu.

F2 – Kontrola přístupu

Komponenta nebude dovolovat uživateli přístup k libovolným metrikám. Uživatel si bude smět zobrazit pouze informace o svých vlastních virtuálních strojích. Systém bude dostupný pouze přihlášeným uživatelům.

2.6.2 Nefunkční požadavky

N1 – Integrace do prostředí správy

Komponenta bude integrována do stávajícího prostředí pro správu virtuálních serverů.

N2 – Middleware v jazyce Java

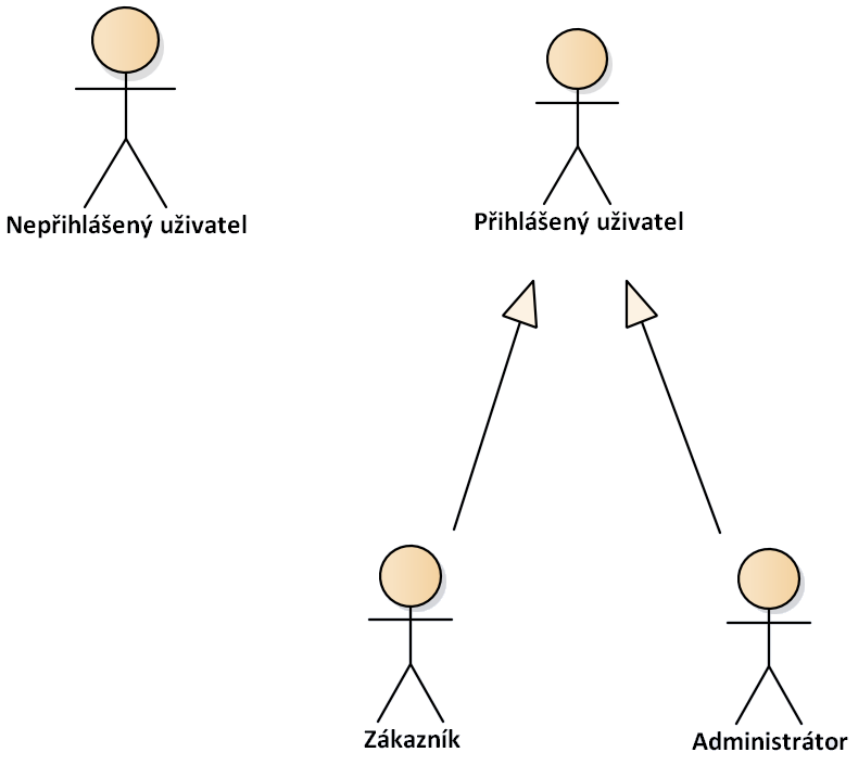
Implementovaný middleware bude napsaný v jazyce Java[7] a bude využívat REST API Gnocchi. Do prostředí správy serverů bude předávat data k vykreslení.

2.7 Modelování případů užití

2.7.1 Seznam účastníků

Seznam účastníků vyplývá z určení cílové skupiny využívající portál pro správu virtuálních serverů. Nepřihlášený uživatel nemá žádné možnosti, pouze se může přihlásit.

Přihlášený uživatel může vykonávat operace ve svém kontě. Takový uživatel je označený jako zákazník. Speciálním případem je přihlášený uživatel s právy administrátora, který může provádět akce i s cizím kontem.



Obrázek 2.14: Seznam účastníků.

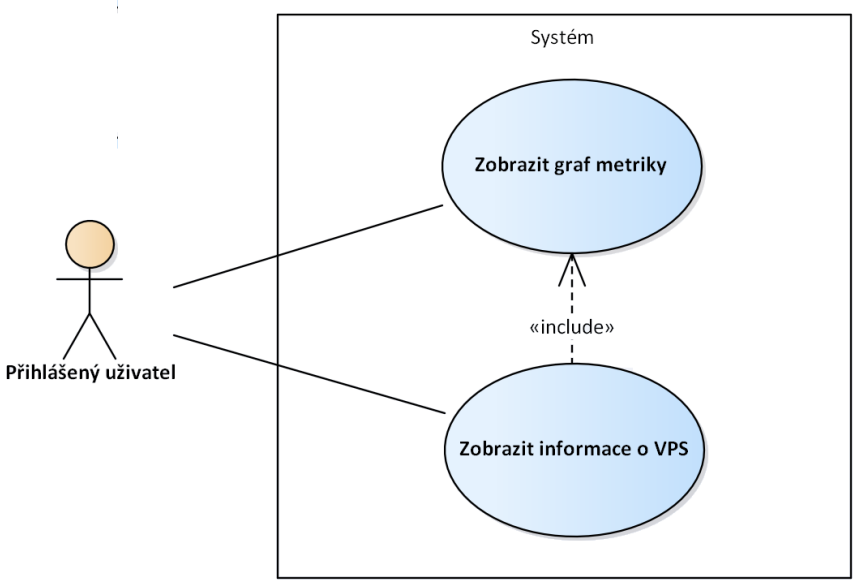
[7]Java – Objektově orientovaný programovací jazyk. Je přenositelný a výsledné aplikace mohou pracovat na různých počítačích a platformách.[24]

2.7.2 Případy užití

Jednotlivé scénáře užití definují vztahy mezi aktéry a případy užití.

2.7.2.1 Kontrola přístupu

Tento diagram plně pokrývá funkční požadavek F2 a definuje přístup k metrikám v aplikaci. Přihlášení do aplikace a změnu nepřihlášeného uživatele v přihlášeného zajišťuje zákaznický portál.



Obrázek 2.15: Use case – Kontrola přístupu.

Zobrazit graf metriky

Tento případ užití nastává ve chvíli, kdy přihlášený uživatel požaduje určité metriky o virtuálním serveru.

1. Uživatel zvolí zobrazení přehledu.
2. Systém zkontroluje oprávnění uživatele.
3. Systém využije integrovanou komponentu a zobrazí graf metriky.

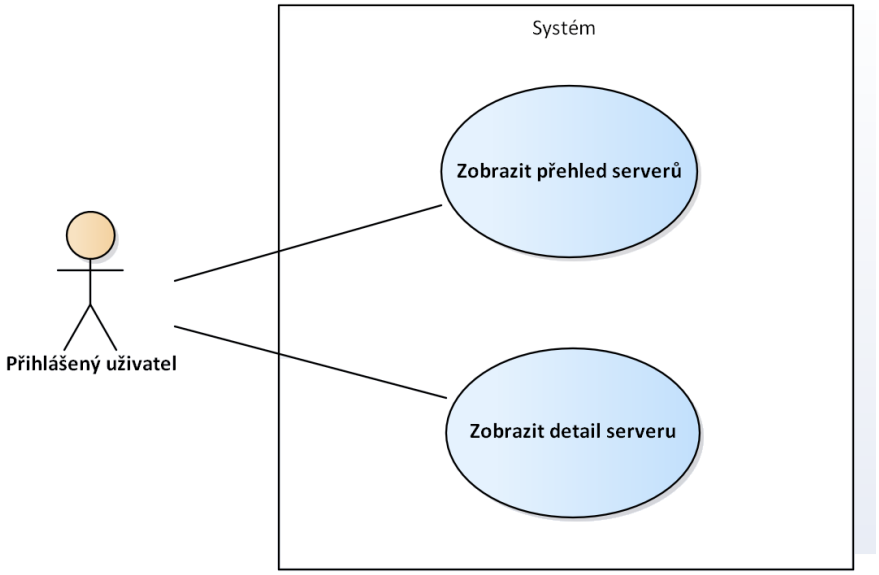
Zobrazit informace o VPS

Tento případ užití nastává ve chvíli, kdy si přihlášený uživatel žádá detailní informace o virtuálním serveru.

1. Uživatel zvolí zobrazení detailu pro virtuální server.
2. Systém zkontroluje oprávnění uživatele.
3. Systém využije integrovanou komponentu a vypíše informace o virtuálním serveru.
4. Include (Zobrazit graf metriky).

2.7.2.2 Vizualizace zátěže virtuálních serverů

Tento diagram plně pokrývá funkční požadavek F1 a definuje možné úkony v aplikaci. Tento diagram nepopisuje stávající portál, popisuje úlohy přidané komponenty.



Obrázek 2.16: Use case – Vizualizace zátěže serverů.

Zobrazit přehled serverů

Tento případ užití nastává ve chvíli, kdy požaduje přihlášený uživatel informace o stavu využití všech svých serverů.

1. Uživatel zvolí zobrazení přehledu serverů.
2. Systém zkontroluje oprávnění uživatele.
3. Systém využije integrovanou komponentu a zobrazí seznam serverů se základními metrikami pro orientaci.

Zobrazit detail serveru

Tento případ užití nastává ve chvíli, kdy si přihlášený uživatel přeje detailní informace o využití určitého serveru.

1. Include (Zobrazit přehled serverů)
2. Systém zkontroluje oprávnění uživatele.
3. Systém využije integrovanou komponentu a vypíše informace o virtuálním serveru.
4. Include (Zobrazit graf metriky).

Kapitola 3

Návrh

Výsledný návrh se odvíjí od kladených požadavků a z teoretických zjištění provedené analýzy.

3.1 Zvolené řešení

Autor pro implementaci řešení zvolil projekt Gnocchi v kombinaci s Ceilometrem. Takový postup totiž přináší výhody obou a eliminuje slabé stránky Ceilometru. Ceilometer má za úlohu pouze samotné sledování instancí a získávání dat.

Gnocchi je zodpovědné za shromažďování, ukládání a výpočet agregací sledovaných dat. Poskytuje také rozhraní pro předávání dat aplikační komponentě ke zpracování, která je bude následně předávat uživateli.

Pro přístup k uloženým datům je k dispozici REST API Gnocchi. Pomocí něj je možné spravovat zdroje, metriky, naměřené hodnoty, nastavení archivace či vyhledávat záznamy. Uživatel zákaznického portálu nebude mít přístup pro manipulaci s rozhraním Gnocchi. Veškerou komunikaci obstará komponenta nebo připraví administrátor fyzické infrastruktury.

Komponenta ve webovém rozhraní pro správu serverů bude realizována v jazyce JAVA s využitím zmíněného Gnocchi REST API. Výsledná grafová vizualizace ze získaných dat bude vykreslena pomocí jazyku Javascript s využitím grafové knihovny CanvasJS.

Cílem je podat uživateli detailní a přehledné informace o dlouhodobém stavu jeho serverů využívajících objednané prostředky.

3.2 Technologie

3.2.1 OpenStack

Cloud systém OpenStack je v době vytváření této práce vyvíjen ve verzi Rocky, která dosud nebyla vydána. Autor tedy zvolil předchozí verzi Queens, která v době tvorby byla vydána a označená jako stabilní.[8]

3.2.2 Ceilometer a Gnocchi

Ceilometer je schopný efektivního sběru dat dvou typů o využívání Openstacku. Jde o data typu událost, kdy jde o vytvoření instance, zálohy stavu a dalších akcí nebo o data typu metrika, která udávají stav sledovaných prostředků v čase (časové řady).[25] V rámci této práce se autor zabývá vizualizací metrik, na které je Gnocchi určené. Pro práci s událostmi by byl vhodný projekt Panko jako úložiště dokumentových dat a logů.

Vývojáři i uživatelé se shodli, že chtějí nadále Ceilometer, aby dlouhodobě data získával. Ale při zvážení množství dat, která nageneruje byla potřebná nová strategie přístupu k problému ukládání a dotazování.

Od OpenStacku verze Newton[9] dokonce není Ceilometer doporučený jako úložiště. Data by měla být uložena v Aodh(Alarming service)[10], Gnocchi a případně Panko.[26]

Dříve byl projekt Gnocchi součástí projektů OpenStacku. Dnes je ale vyvíjen odděleně a nezávisle. Spojení Ceilometer a Gnocchi je typické jako uskupení zvané *Telemetry Service* – služba pro monitorování provozu. Poslední stabilní verze byla v době tvorby 4.2, byla tedy autorem zvolena pro využití.

3.2.2.1 Architektura Gnocchi

Gnocchi se skládá z vícero služeb. HTTP REST API, démonů[11] `statsd` a `metricd` a úložiště. Data jsou získávána přes HTTP REST API nebo démona statsd. Démon metricd provádí operace nad získanými daty jako výpočty statistik. Těchto metricd démonů je typicky více a stejně jako úložiště je možné je v budoucnu přidávat pro možnosti škálování.

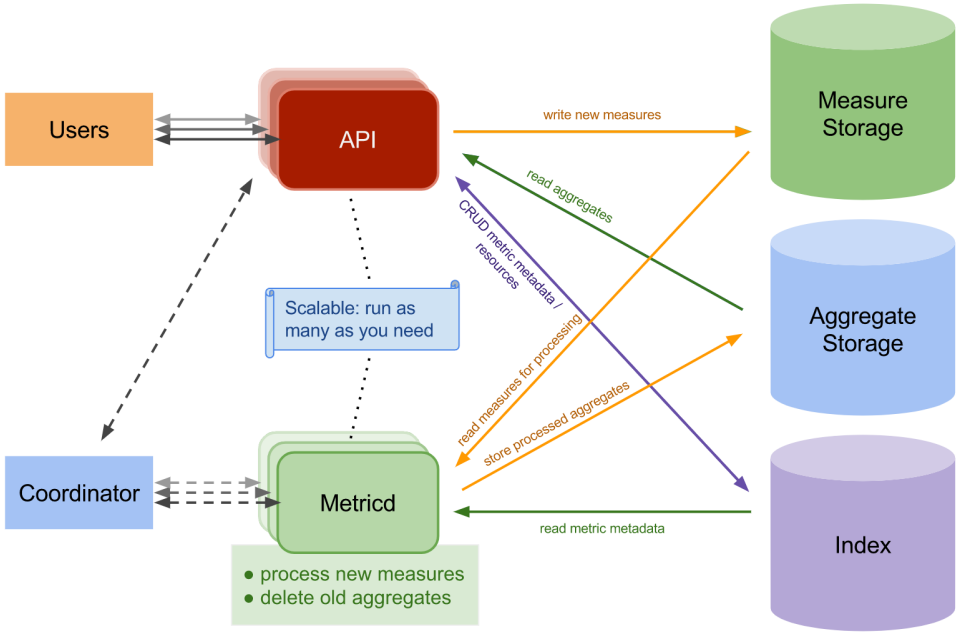
V této architektuře jsou využity také tři úložiště. Jsou univerzální, je možné vybrat preferované. Gnocchi může využívat různá úložiště pro metriky

[8] Stabilní verze – Anglicky *Stable* jsou v softwaru označovány verze, které jsou hotové a připravené pro plné použití či ostré nasazení do produkčního prostředí.

[9] Newton – Verze systému OpenStack mají své jmenné označení a vychází abecedně. S jistou tedy víme, že verze Queens bude novější vydanou vezí, než Newton.

[10] Aodh – Služba poskytující schopnost zachytit definovanou událost a provést automaticky požadované akce.

[11] Démon – V informatice jde o program nebo proces, většinou patřící k větší aplikaci, který v počítači nečinně spí a čeká na nějakou určitou událost. Jakmile událost nastane, vykonává přidělenou úlohu.[28]



Obrázek 3.1: Architektura projektu Gnocchi.

a naměřené hodnoty. (Measure storage a Aggregate Storage z obrázku architektury 3.1.) Dalším zástupcem je databáze pro indexování zdrojů, spojování s metrikami a ukládání archivačních politik. Obsahuje pro všechny záznamy také jejich definice a typy.[27]

Autor ponechal původní volbu ukládání metrik do souboru a jako databázi zvolil MySQL. Pomocí API jsou data předávána uživateli nebo jiným aplikacím. Výsledná komponenta tedy bude komunikovat výhradně s tímto bodem.

V případě, že by byl projekt Gnocchi provozován na více výpočetních uzlech, bylo by vhodné zvážit použití objektového úložiště Ceph místo ukládání do souborů kvůli lepší škálovatelnosti.

3.2.2.2 Archivace dat v podání Gnocchi

Způsob agregace a tedy archivace dat je dán archivační politikou přiřazenou k dané metrice. Ta udává, jak dlouho budou naměřená data uchována v metrice a jak budou agregována. V této práci použita politika *low*, která znamená granularitu dat 5 minut, v posledních 30ti dnech. Takové nastavení dostačuje a zároveň šetří ukládacím prostorem. Maximální velikost metriky je 406KiB.

Jinou možnou politikou pro detailnější přehled je například *high*, která má granularitu 1 sekundu poslední hodinu, 1 minutu poslední týden a 1 hodinu poslední rok. Maximální velikost takové použité metriky je 1057 KiB.[29]

3.2.3 Apache Tomcat

Běhovým prostředím pro provoz podnikových aplikací v jazyce Java EE[12] jsou aplikační servery. Apache Tomcat je v tomto směru typický zástupce pro vývoj webových aplikací Java Servlet a Java Servlet Pages.

Některé jiné aplikační servery sice nabízí funkčnost navíc, ale aplikace využívající specifické funkce, nemusí být tak ve výsledku přenositelné. Pro své jednoduché nasazení autor zvolil server Apache Tomcat ve verzi 8. Ten je vyvíjen jako open source projekt a je založen na Java servletech.

3.2.4 JavaScript

JavaScript je multiplatformní skriptovací jazyk, který umožňuje provádět operace v aplikaci na straně klienta. Skript běží v prohlížeči a umožňuje například verifikaci dat ještě před odesláním na server nebo dynamické chování stránek.[30]

3.2.4.1 CanvasJS

HTML element `<canvas>` je využíván pro vykreslování grafických elementů na webové stránce pomocí Javascriptu. Element je kontejner pro grafiku, která musí být vykreslena skriptem.[31]

CanvasJS je HTML5 Javascriptová knihova pro vykreslování grafů. Zvolena byla na základě moderního přístupu (využití elementu canvas) a subjektivně krásných výsledných grafů. V neposlední řadě se chlubí svým výkonem, zvládne vykreslit graf se 100 000 hodnotami za 219 ms.[32]

[12] Java EE – Java Platform, Enterprise Edition je součást platformy Java určená pro vývoj a provoz podnikových aplikací a informačních systémů.

Kapitola 4

Realizace

Na základě analýz a zpracovaného návrhu byla provedana implementace samotného řešení. Tato kapitola popisuje klíčové části řešení, popis fungování a nastálé problémy, které způsobily změny oproti návrhu.

4.1 DevStack

Nasazení kompletního softwaru OpenStack je velmi složitá operace. Je zde mnoho variant, jak postupovat a jednotlivé případy se od sebe liší použitím různých projektů, nastavení chování či rozložení komponent na více fyzických strojů. To přináší široké možnosti přizpůsobení pro dané nasazení, dobrý výsledný výkon a snadné škálování do budoucnosti.

V případě, že není potřeba rozložení na více strojů a OpenStack je potřebný pro vývoj, testování či seznámení, postačí instalace všech komponent na jedno zařízení. Pro tyto případy jednoduchého nasazení již existuje projekt DevStack.

DevStack je soubor skriptů, které umožní kompletní nasazení softwaru OpenStack. Jde o řešení, které na podporovaný operační systém Linux (Ubuntu, Fedora, CentOS, Debian nebo OpenSUSE) nainstaluje sám všechny požadované části. Je vyžadován pouze aktualizovaný operační systém a doplnění konfiguračního souboru o uživatelská hesla.[33] Skript stáhne a nainstaluje všechny návaznosti. Instalace trvá 15-60 minut v závislosti na internetové konektivitě a výkonu hardwaru.

Skript využívá GIT jako zdroje pro stahování a následnou instalaci všech projektů. V případě manuální instalace může uživatel také využít GIT nebo systémové repozitáře[13] Ubuntu či Debianu (podle zvoleného systému). Výsledná podoba automatické instalace by se tedy neměla lišit od systému vytvořeného ručně po částech.

[13]Repozitář – Server na internetu, který provozují většinou tvůrci distribuce a odkud je možné stahovat balíčky s aktualizacemi programů či systému, případně z něj rovnou instalovat aplikace.

4.2 Problémová funkčnost

Řídit se pouze oficiální dokumentací většinou nestačí. Je psaná odborně a přímočaře[14], především pro zkušenější vývojáře. Chybí ukázky použití, fotky obrazovek, nákresy datových modelů či sbírky vyskytujících se problémů.

4.2.1 DevStack

Řídit se samotnou dokumentací se nevyplácí už u samotné instalace DevStacku. Postup nezmiňuje, že pro starší stable verze je nutné přidat parametr.[34] Bez něj je vždy z GITu[15] stažena nejnovější verze.

```
git clone https://git.openstack.org/openstack-dev/devstack
```

Obrázek 4.1: Chybný příkaz ke stáhnutí verze Queens dle dokumentace.

```
git clone https://git.openstack.org/openstack-dev/devstack \
        -b stable/queens
```

Obrázek 4.2: Správný příkaz ke stáhnutí DevStacku verze Queens.

Minimální konfigurace nutná pro spuštění libovoné verze DevStacku taktéž není správně uvedená. V Testovaných verzích Rocky, Queens a Pike je nutné ke konfiguraci přidat minimálně záznam udávající IP adresu hosta.

```
[ERROR] /home/user/devstack/stackrc:821 Could not determine
host ip address. See local.conf for suggestions on setting HOST_IP.
```

Obrázek 4.3: Chyba při instalaci DevStacku verze Queens.

Za zmínku stojí také náročnost OpenStacku na výpočetní prostředky. Ve verzi Rocky i Queens z neznámého důvodu velmi vytěžoval procesor projekt Neutron a jeho úlohy. OpenStack má také velmi vysoké paměťové nároky. Při samotném běhu bez spuštěných instancí využívá systém přes 6 GB paměti. (Nároky na paměť nejsou zmíněny)

Toto nasazení OpenStacku bylo také velmi nestabilní i bez uživatelského zásahu. Projekt Keystone, který ověřuje uživatele během vývoje dvakrát ze dne na den úplně přestal fungovat. Také spolupráce Horizonu s Novou pro

[14] Pro určitý případ je nutné postupovat přesně příručkově – pokud uživatel neví, kterou cestou se vydat, nebo se v polovině návodu vyskytne problém, musí se uchýlit ke hledání na internetu, většinou na fórech komunity. Typické problémy nebo návrhy řešení chybí.

[15] GIT – Systém správy verzí. Opensource nástroj využíván při vývoji softwaru.

vytváření instancí nebyla spolehlivá. Po určitém čase nebylo možné vytvářet z webového rozhraní nové instance, pouze z příkazového řádku serveru. Pro nasazení v praxi i pro jednoduchý vývoj se DevStack jevil jako nevhodný jelikož důvody chování nejsou známy.

4.2.2 Spojení OpenStack, Ceilometer a Gnocchi

Zmíněné spojení projektů není nic výjimečného. Navazují jeden na druhý. Projekt Gnocchi by byl dokonce bez Ceilometru snad zbytečný. I když už není Gnocchi oficiálně spojen s OpenStackem, stále je uváděn jako typický zástupce pro ukládání dat.

Všechny části lze nainstalovat najednou při první instalaci OpenStacku. V případě instalace pomocí DevStacku stačí do konfiguračního souboru připsat povolení jednotlivých služeb (direktiva `enable_service`) a zásuvných modulů (direktiva `enable_plugin`). Pozdější instalace je náročnější a vyžaduje úplnou manuální konfiguraci. Znovuspuštění skriptu pro instalaci DevStacku už také není možné.

Bohužel projekty v navrhovaných verzích se nepodařilo zprovoznit. Neúspěšné byly i opakované instalace podle různých postupů či použití různých konfigurací. V OpenStacku verze *Queens* s Gnocchi verze `stable/4.2` projekt Ceilometer běžel, ale nefungovalo uložení dat do Gnocchi. – Problém byl vyřešen přechodem na starší verze.

4.3 Zvolené funkční verze

Všechny zmíněné technologie se velmi rychle vyvíjejí a chyby budou nejspíše v blízké době opraveny. Nicméně během vývoje práce byla otestována plná funkčnost pouze v níže uvedených verzích technologií.

Tabulka 4.1: Verze použitých technologií

Technologie	Verze
OpenStack	Pike
Ceilometer	Pike
Gnocchi	4.1
Ubuntu OS	16.04
Java	8 (update 162)
Apache	2.4.18
Apache Tomcat	8
JavaScript	ECMAScript 6
HTML	5
CanvasJS	2.0.1

4.4 Nasazení

Pro úspěšné nasazení celé práce je nutné přesné dodržení všech nastavení a zvolených verzí. Jak už bylo zmíněno, stačí malý rozdíl v použité verzi, nastavení či prohození kroků a výsledné chování není zaručeno. Taktéž dodržování postupu oficiální dokumentace nezaručuje správnou a očekávanou funkčnost.

V této kapitole je shrnut správný postup nasazení. Přesné znění příkazů a konfigurací se nachází v příloze C.

4.4.1 Systém

Pro instalaci DevStacku je vyžadována čistá instalace linuxového operačního systému. Využit byl doporučovaný – Ubuntu 16. Operační systém byl instalován do virtuálního prostředí. To obnáší nižší výkon a nutná donastavení přístupu, ale má výhodu rychlého nasazení v případě nutnosti přeinstalování (což se dělo často) a velkých poskytnutých výpočetních prostředků, které by autor nebyl schopný svými silami zajistit.

Operační systém byl nejprve aktualizován a byly doplněny užitečné programy, aby jejich pozdější instalace neovlivnily chod. Instalované programy:

- aptitude
- build-essential
- git
- ntp
- openssh-server
- python

Práce byla vyvíjena ve virtualizovaném prostředí serveru společnosti Homeatcloud.cz. Jde o přidanou úroveň virtualizace z důvodu snažšího vývoje odděleného od produkčního prostředí. Jednotlivé vrstvy jsou:

1. Fyzický server společnosti.
2. Produkční OpenStack společnosti.
3. Virtuální server Ubuntu.
4. OpenStack pro vývoj práce.

Využití OpenStacku společnosti bylo pouze uživatelské a nedošlo k žádným změnám v systému. Kdekoli v textu této práci byl zmíněn software OpenStack, jde o systém pro vývoj práce na virtuálním serveru Ubuntu.

4.4.2 DevStack

Prostředí OpenStack bylo instalováno pomocí skriptů DevStack. Stažen byl z oficiálního zdroje. Do konfiguračního souboru `local.conf` bylo nutné přidat vlastní heslo, `HOST_IP` a povolit volitelné služby a moduly. Explicitně povolené služby při instalaci:

- Cinder
- Ceilometer
- Gnocchi
- Gnocchi - Grafana
- Neutron
- Heat
- Aodh
- Nova
- Horizon

Pro využívání příkazové řádky ke správě je zapotřebí nastavit prostředí. Lze k tomu využít soubor dostupný z webového rozhraní nainstalovaného Horizonu, kde je k dispozici ke stažení `Admin-openrc.sh`. Po jeho importu do prostředí a zadání přístupového hesla je možné ovládat všechny části OpenStacku. (Pro ovládání Gnocchi je potřeba přidat do prostředí další dvě proměnné.)

4.4.3 Port forwarding

V tomto případě byl systém provozován virtuálně. I pokud některý proces naslouchá na určitém portu[16], stále není možné se s ním spojit. Důvodem je i virtuální síť, kterou je virtuální počítač propojen s fyzickou.

Virtuální sítě, stejně jako ty reálné, využívají routery, které umožňují směrování požadavků. Pokud je potřeba dopravit požadavek z vnější sítě (fyzická) do vnitřní (virtuální, spojená se serverem), je potřeba specifikovat všechna pravidla na routeru.

Pro virtuální stroj na kterém jsou provozovány všechny komponenty je nutné povolení následujících portů se směrováním na adresu počítače.

[16]Listening port – Stále běžící aplikace, čekající na data na určitém síťovém portu. [35] Typickým příkladem je protokol HTTP využívající port 80. Na jeho požadavek naslouchá webový server.

Tabulka 4.2: Potřebné porty a jejich aplikace.

Port	Aplikace
22	SSH
80	Apache
6080	VNC
8080	Apache Tomcat

4.4.4 Spouštění instancí

V této práci byly využívány účty Admin a Demo. Účet Admin disponuje vyššími právy, ale oba fungují jako oddělení uživatelé, kteří mohou spouštět své instance. Pro testovací účely byl využíván operační systém Linux CirrOS. Jde o minimální Linuxovou distribuci specializovanou pro běh v cloud prostředí.[36]

Instance lze spouštět z uživatelského rozhraní Horizon nebo z příkazového řádku. Při použití Horizonu jde o postupné zvolení obrazu, velikosti přidělených prostředků, výběr sítě a jiné. Použití je poměrně intuitivní. Instance je možné vytvářet i v příkazovém řádku. Jedním příkazem je nutné definovat vše.

```
nova boot --flavor m1.tiny --image cirros-0.3.5-x86_64-disk \
          --nic net-name=net1 --security-group default \
          My_CirrOS_server_007
```

Obrázek 4.4: Spuštění nové instance z obrazu.

4.4.5 Gnocchi a Ceilometer databáze

Po dokončené instalaci je možné si ověřit, zda běží procesy Ceilometru a Gnocchi. Pokud ano, lze vypsat například seznam sledovaných metrik.

```
gnocchi metric list
```

Obrázek 4.5: Vypsání sledovaných metrik.

Seznam bude prázdný a je potřeba znovu inicializovat indexovací nástroje a úložiště. Až poté je možné zobrazit sledované metriky.

4.4.6 Apache Tomcat

Instalace Apache Tomcat byla provedena pomocí repozitářů Ubuntu. Původním portem, který Tomcat používá je 8080, což bylo ponecháno.

```
gnocchi-upgrade
ceilometer-upgrade --skip-metering-database
```

Obrázek 4.6: Inicializace Gnocchi a Ceilometru.

4.4.7 Aplikace

Zdrojová složka aplikace je umístěna ve složce `webapps` serveru Apache Tomcat. Zbývá upravit konfiguraci požadavku pro vyžádání tokenu na správné uživatelské údaje.

Jelikož je aplikace na stejném serveru jako Gnocchi API, se kterým komunikuje, nejsou třeba další úpravy. Ukázková aplikace je dostupná na adrese `HOST_IP:8080/graph/`.

4.5 Implementace

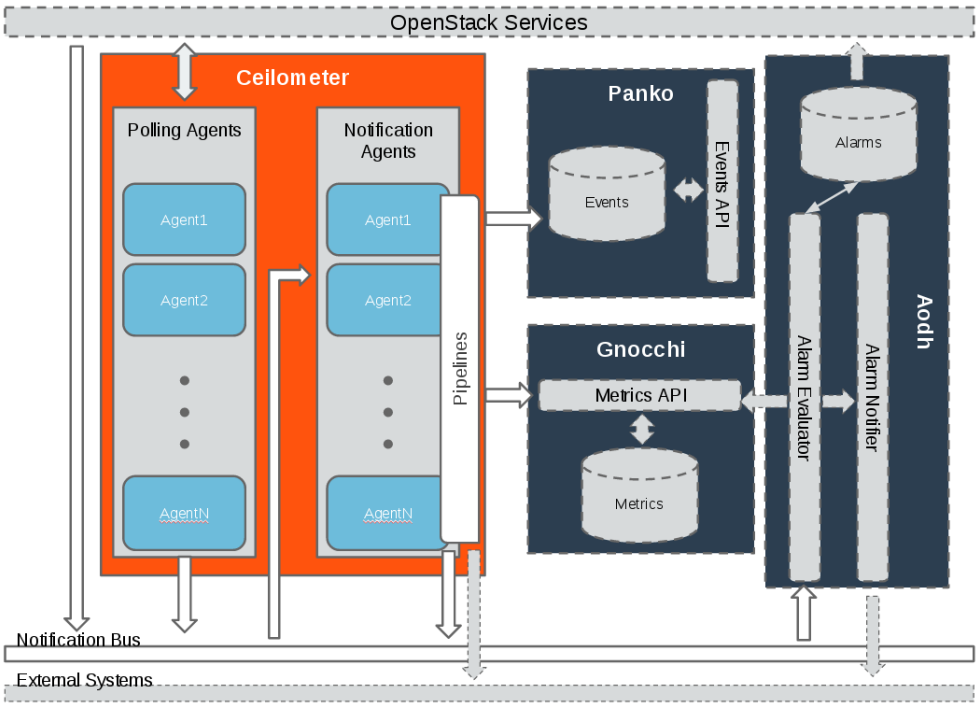
4.5.1 Architektura Ceilometer a Gnocchi

Projekt Ceilometer sleduje sice více služeb zároveň, ale zde bylo cílem získat informace o instancích. Démon Ceilometru funguje jako *polling agent*, tedy periodicky se dotazuje hypervisorů na informace o běžících instancích.

Pro takto získaná data o metrikách je připraven Gnocchi. Získané časové řady ze všech hypervisorů optimalizuje pro dotazování. Celkový pohled na Ceilometer a jeho spojení s Gnocchi je vyobrazen na diagramu 4.7.

Optimalizace provádí démoni Gnocchi, kteří nová data zapracovávají do stávajících. Řídí se zvolenou archivační politikou. (V práci zvolena politika low, více v sekci 3.2.2.2.) Data jsou ukládána do souboru a metadata jsou indexována v relační databázi MySQL.

Zpracovaná data jsou poskytována pomocí REST API Gnocchi. Pro zamezení přístupu k tomuto rozhraní není umožněno komunikovat na tomto portu z vnější sítě. Jelikož je aplikace na stejném serveru, Gnocchi API tedy využívat může.



Obrázek 4.7: Architektura sběru dat pomocí Ceilometru.[25]

4.5.2 Tok dat

Virtuální server je používán jako obvykle a nelze z něj nic poznat. Ceilometer zjistí data o používání a posílá je do Gnocchi, které je cetrálním úložištěm pro všechny instance, všechny metriky a tedy všechna data.

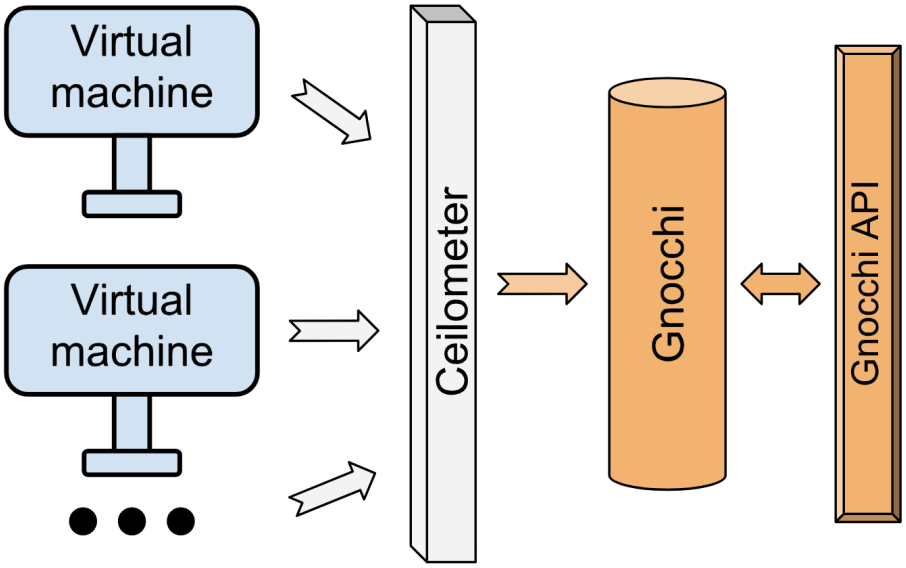
Na obrázku 4.8 lze vidět, jak jsou projekty zapojeny a jak data putují. Lze si také všimnout, že oproti obecné architektuře sběru dat na diagramu 4.7 chybí projekty pro ukládání událostí a Alarming service. Je to nejen z důvodu zjednodušení výsledné konfigurace, ale především je cílem prezentovat data o využití instancí.

Na obrázku 4.9 pokračuje tok dat z Gnocci API až k uživateli, kterému se zobrazí výsledné grafy o využívání jeho instance.

1. V prvním kroku klientův prohlížeč zažádá o zobrazení určité stránky aplikace, poskytující graf využití.

2. Aplikace na straně serveru odešle požadavek, kterým si ověří funkčnost přístupového tokenu k datům nebo získá token nový.

3. Na základě parametrů v požadavku jsou zažádány data o určité instanci.

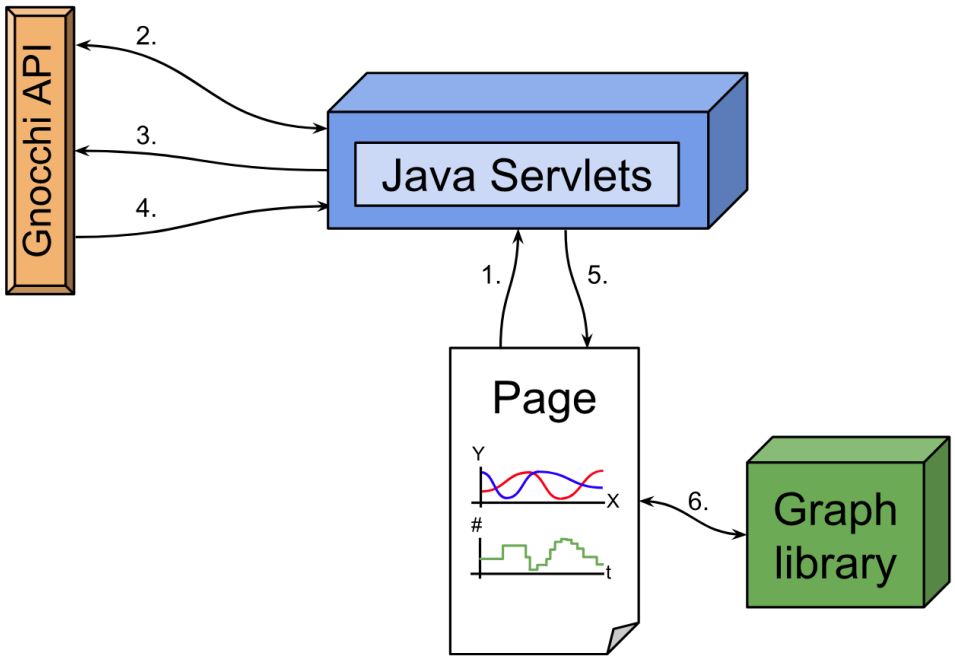


Obrázek 4.8: Tok dat až k API.

4. Gnocchi API vrátí požadovaná data. V případě potřeby nemusí být odeslána všechna data, mohou být převzorkována, aby ulehčila zobrazení, pokud není důležitá přesnost.

5. Webová stránka je vrácena prohlížeči uživatele. Stránka také obsahuje skripty obsahující data k zobrazení v grafech.

6. Javascript na stažené stránce využije vnější grafovou knihovnu a zobrazí graf k prezentaci.



Obrázek 4.9: Tok dat z Gnocchi API k uživateli.

4.5.3 Struktura aplikace

Zdrojová složka aplikace je umístěna ve standardní cestě pro Apache Tomcat, ve složce `webapps/graph`.(Podrobněji v příloze C.3.2) Celé rozvržení se řídí typickou strukturou pro Java Servlet aplikaci.[37]

webapps/graph

- `images` ..... Obrázky v aplikaci
- `src` ..... Zdrojové soubory aplikace
  - `configuration` ..... Soubory nastavení
  - `templates` ..... Šablony grafů
- `temp` ..... Složka pro dočasné soubory
- `WEB-INF` ..... Složka obsahující popis nasazení
  - `classes` ..... Přeložené třídy
  - `lib` ..... Použité knihovny .jar
  - `web.xml` ..... Popis nasazení v XML
- `index.jsp` ..... Počáteční soubor aplikace

Obrázek 4.10: Struktura aplikace.

4.5.4 Uživatelské rozhraní

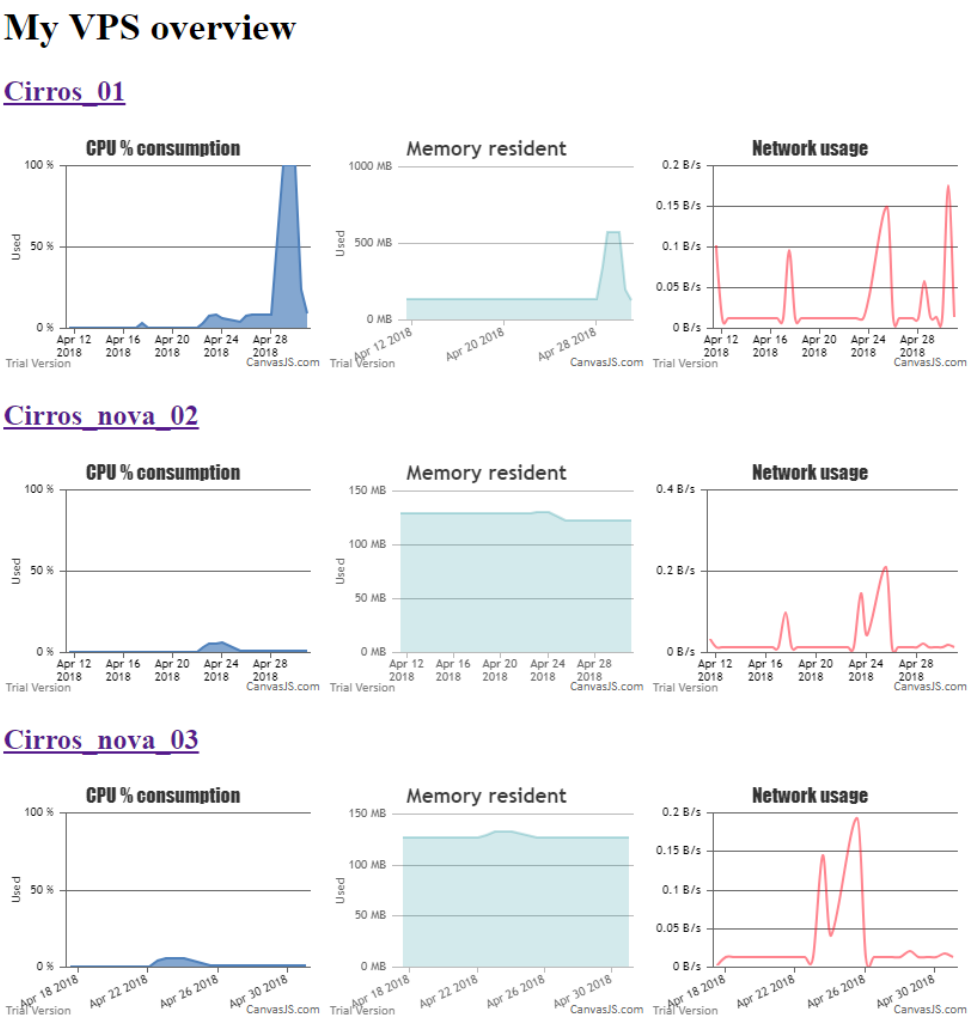
Práce byla vyvíjena pro nasazení v produkčním prostředí společnosti Homeatcloud.cz, bohužel v době vývoje používala společnost starší verzi cloudového softwaru OpenStack. Pro tyto verze nebyla funkčnost zaručována, jelikož i projekt Gnocchi příslušné verze byl spíše testovací nástroj. Nelze také opomíjet fakt, že kompletní nasazení celého řešení je uživatelsky náročná operace vyžadující velké zkušenosti. V případě neúspěchu by hrozil kolaps celého systému.

Z těchto důvodů není výsledná práce nasazena v produkčním prostředí společnosti. Byla tedy vyvíjena v odděleném prostředí a bude nasazena v budoucnu administrátory a programátory společnosti. Pro demonstraci své funkčnosti je vytvořeno na testovacím serveru pár stránek, představující typické využití komponent, jako by tomu bylo v ostrém provozu. Pro výběr uživatele k demonstrativním účelům slouží úvodní obrazovka umožňující zadání jeho OpenStack ID. Správu uživatelů, jejich autentizaci a autorizaci bude zajišťovat webová aplikace společnosti.

Obrazovka se seznamem běžících instancí poskytuje krátký přehled o využití jednotlivých VPS. Je zobrazen seznam serverů se základními metrikami, vždy pouze s hrubým výběrem dat. (Vysoká granularita.) Uživatel je tedy informaván o celkovém, dlouhodobém stavu svých serverů.

Na fotografii obrazovky 4.11 je seznam všech virtuálních serverů, které má uživatel ve své správě. Lze vidět, že kromě menšího síťového pohybu jsou servery *Cirros_nova_02* a *Cirros_nova_03* většinu času nevyužity. (Jediná zátěž procesoru kolem pěti procent 24. 4. 2018.) Aktivním serverem s krátkodobě velkým zatížením je *Cirros_01*, bylo by tedy vhodné aktivitu prozkoumat.

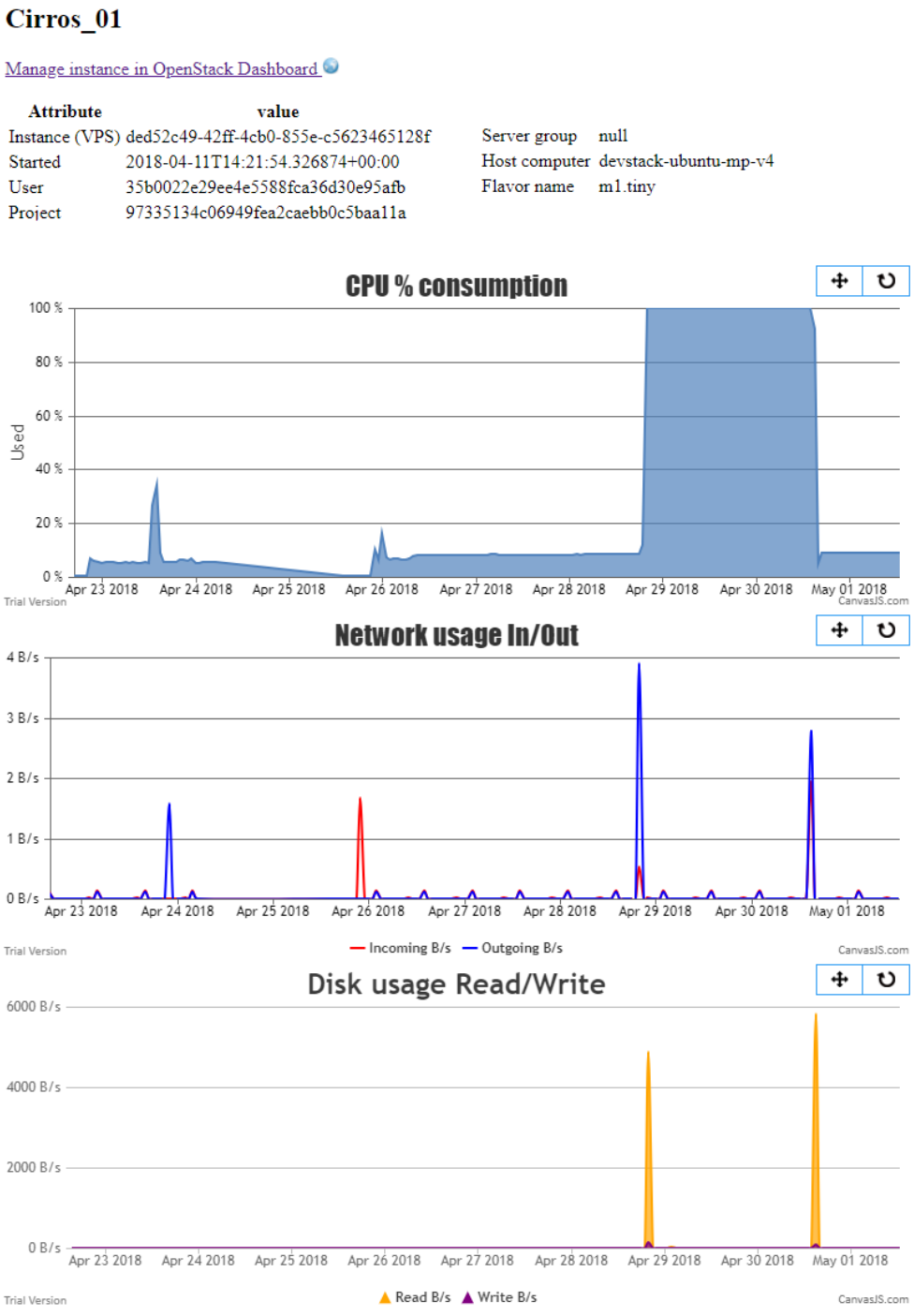
Pro detailní pohled lze přejít na detail instance. Stránka poskytuje široké detaily jako identifikační údaje, projekt do kterého server patří, kdy byl spuštěn, či na kterém fyzickém počítači je umístěn. Tato stránka obsahuje také velké grafy s nízkou granularitou a možností přiblížení. Je tak vhodná pro přesné analýzy provozu.



Obrázek 4.11: Stránka se seznamem běžících instancí.

Na fotografii obrazovky 4.12 je kompletní přehled aktuálního stavu instance s názvem *Cirros_01*. Je možné se podrobně věnovat rozboru využití. Z grafů lze vyčíst, že 23. 4. 2018 v poledne (po najetí myší je zobrazen přesný čas) byla vykonána akce, která stroj zatížila výpočetně a bylo odesláno pár bytů po síti.

Druhý důležitý moment je vidět 28. 4. 2018 ve 20:00 až do 30. 4. 2018 v 15:00. V daných časech proběhla jednorázová disková i síťová aktivita a v celém mezičase byl virtuální stroj výpočetně vytížen na 100 %. V případě reálného provozu by to byl důvod k prošetření, zde šlo o záměr autora pro ilustraci fungování.



Obrázek 4.12: Stránka s detailem instance.

4.5.5 Grafy

Aby byly výsledné grafy přehledné a dodržovaly přizpůsobení uživatele (správce), obsahuje aplikace sbírku Javascriptových souborů, které tvoří vykreslovací skripty pro grafovou knihovnu. Pro nejběžnější metriky jsou připraveny soubory přizpůsobených šablon. V případě použití jiných metrik jsou k dispozici základní šablony. Pokud uživatel bude chtít vykreslovat jiné metriky, stačí je zadat do požadavku. Vždy budou vykresleny.

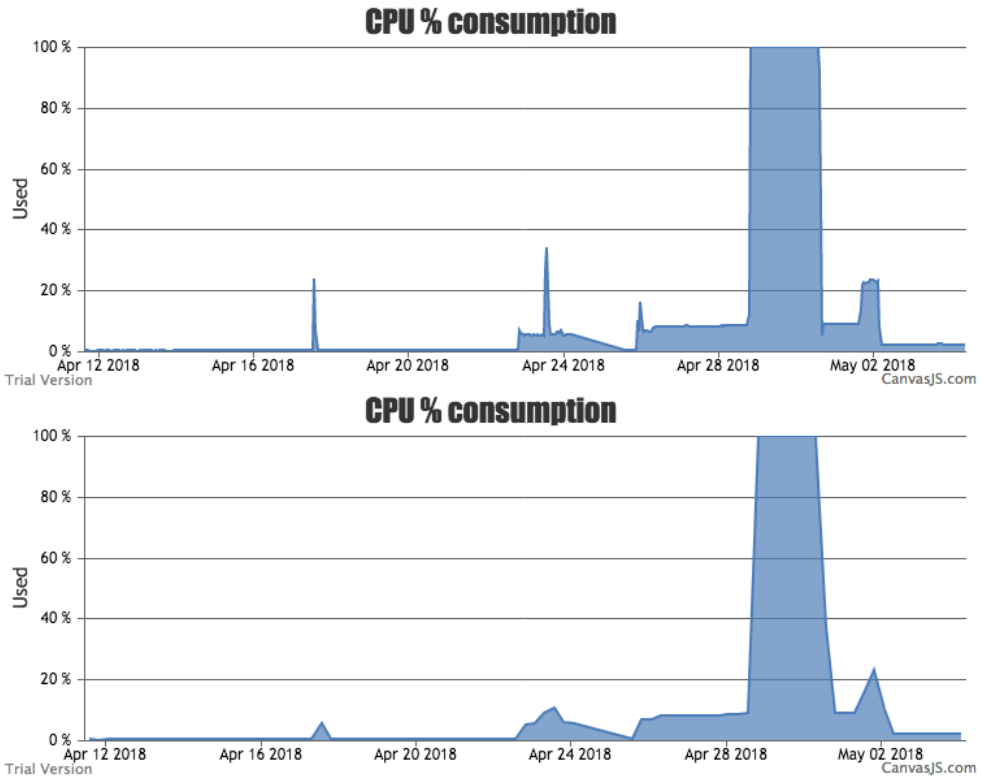
Pokud se mu výsledek ze šablony nezamlouvá, může vytvořit vlastní vzory nebo přepracovat šablonu podle dokumentace grafové knihovny. (Barvy, popisy, legenda, typ grafu,...)

Výsledný graf je velmi ovlivněn i prací s naměřenými daty. Velkým přizpůsobením je tak úprava dotazů, které komponenta provádí na API Gnocchi.

Granularita

Granularita označuje úroveň hloubky reprezentace dat, jinak řečeno jejich přesnost. Pokud jsou nyní Ceilometrem sbírána data každých 5 minut, zobrazení všech záznamů je nejpřesnější informace, jakou je možné získat. Ve výsledné komponentě jsou vybrána data s rozdílnou granularitou pro různé stránky. To přináší různou úroveň detailu i rozdílnou velikost zpracovávaných dat. Na obrázku v ukázce 4.13 je porovnání grafů stejné instance (stejné metriky) s rozdílnou granularitou dat.

Je možné pozorovat rozdílnou úroveň detailu. Graf nahoře s granularitou 5 minut je zubatější a zobrazuje přesnější detail. Graf dole s granularitou 6 hodin není tak ostrý a ztrácí část informace, ovšem pro přehled může být dostačující.



Obrázek 4.13: Grafy s různou granularitou.

Agregace

Agregace je spojování jednotlivých dat do jedné hodnoty. Jakým způsobem to bude prováděno záleží na agregační funkci. Pokud přijde požadavek na vypsání dat, je původním chováním Gnocchi vypsat agregovaná data metodou průměru.

Jeden záznam tedy určuje průměrnou hodnotu metriky mezi časovými body. Je možné přidat do požadavku parametr s jinou požadovanou agregací – například maximum.

Časové rozmezí

Je možné filtrovat data také podle času. Zadáním počátečního nebo koncového data lze vybrat pouze určitý časový úsek. Pokud v zadaném čase záznamy chybí, je odpověď prázdná.

4.5.6 Metriky

Ceilometer sbírá o instancích řadu metrik. Jejich název většinou dobře vypovídá o významu. Některé metriky jsou ale bohužel zavádějící, jelikož je jejich sledování technicky obtížné.

Využití CPU je jednoduché sledovat podle vyžádaných výpočetních prostředků. Naopak použitý hypervisor QEMU nevrací dobré informace o paměti. Když instance potřebuje paměť, dostane kolik vyžaduje až do výše své kvóty. Pokud ji ale už nevyužívá, zůstává nevyužita, ale instanci stále přidělena. Je to způsob fungování hypervisoru a snaha o šetření drahé, sdílené fyzické paměti. Toto chování lze pozorovat také na obecném přehledu instancí 4.11. (Když zátěž zmizela, využití paměti zůstalo.)

Ukázková stránka obsahuje vždy tři metriky pro každou běžící instanci. – využití procesoru, paměti a využití sítě (download). Po výběru určité instance zobrazuje detailní stránka využití procesoru, zátěž disku (zápis a čtení) a využití sítě (download a upload).

Celkový počet metrik, které jsou k dispozici závisí na použité verzi Ceilometru. Ve verzi Pike je k dispozici přes 30 metrik k běžící instanci. Další metriky, jako jsou například údaje o síťovém provozu, lze získat dodatečným dotazem. Síť je totiž síťovým zdrojem, ne instancí, takže je potřeba se dotazovat na zdroj typu `instance_network_interface`, který obsahuje parametr `instance_id` udávající, k jaké instanci síťový zdroj náleží. Výběr zajímavých metrik je v tabulce 4.3.

Pro zobrazení ukázkové stránky s detailním přehledem o instanci je tedy zapotřebí zjistit ID všech metrik. ID metrik o procesoru, zápisu na disk a čtení z disku jsou získány přímo. Pro síť se dohledají mezi síťovými zdroji podle instance. Odtud se získají ID pro download a upload.

Pro vykreslení grafů už jsou k dispozici všechny klíčové informace. Získají se následně data pro každou metriku a v případě využití disku a sítě se dvě metriky spojí do jednoho grafu pro lepší přehlednost.

Tabulka 4.3: Zajímavé metriky ke sledování.

Označení metriky	Jednotka	Význam
cpu_util	%	Využití procesoru.
disk.allocation	B	Velikost využitého místa na disku.
disk.capacity	B	Kapacita disku.
disk.read.bytes.rate	B/s	Množství čtení z disku.
disk.write.bytes.rate	B/s	Množství zápisů na disk.
network.incoming.bytes.rate	B/s	Množství příchozích dat.
network.outgoing.bytes.rate	B/s	Množství odchozích dat.
vcpus	vcpu	Množství alokovaných cpu.

Ne všechny metriky jsou ale správně použitelné. Především využití paměti (`memory.resident`) není možné dobře sledovat a jiné metriky neobsahují data – například `memory.usage`. Je tedy důležité vybrané metriky nejprve vyzkoušet, jejich podpora ve verzi Ceilometru nestačí.

Kapitola 5

Testování

Pro otestování funkčnosti komponenty byly vytvořeny demonstrační stránky, popsané v sekci 4.5.4. V části testování je zkoumán celkový dojem pro uživatele z pohledu použitelnosti.

5.1 Výkon v běžném provozu

Jelikož výsledná komponenta je využívána na ukázkové stránce, byla testována vždy celá stránka, tak jak se zobrazí uživateli. Výkon byl měřen webovými testovacími nástroji.

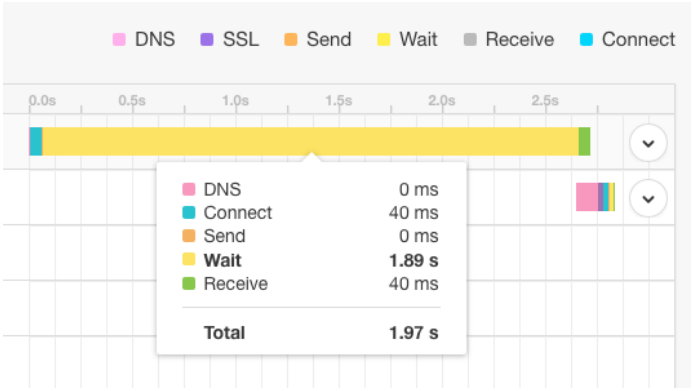
Pro testy byla použita stránka obsahující přehled všech serverů uživatele a stránka s detailem jednoho serveru. Přehledová stránka obsahuje stručný přehled tří metrik s granularitou 12 hodin. Stránka s detailem poskytuje informace o zařazení serveru (ID projektu, uživatele, hostitelský počítač a jiné.) a podrobné metriky s granularitou 1 hodina. V obou případech jsou zobrazena data za poslední měsíc používání serveru.

5.1.1 Pingdom

Pingdom je webový nástroj umožňující monitorování stavu a měření výkonu webových stránek. Zdarma poskytuje nástroj na testování, který po vložení odkazu provede dotaz na stránku a zobrazí statistiky. Testy byly prováděny ze serverů ve Švédsku.[38]

Přehled serverů

Dle náhledu je stránka zobrazena správně. Celková doba od odeslání požadavku do kompletního zobrazení stránky byla 2,34 sekund při celkové velikosti 122,9 kB. Z celkového času nejvíce zabralo čekání prohlížeče na data. (1,9 sekundy.) Na obrázku 5.1 je ilustrován rozbor času postřebný pro zobrazení stránky.



Obrázek 5.1: Rozbor času nástrojem Pingdom.

Detail serveru

Stránka s detailem serveru byla také zobrazena správně. Podrobnější grafy s touto granularitou nejsou problémem a stránka byla zobrazena za 1,8 sekundy při celkové velikosti 216,3 kB. Nejvíce času opět zabralo čekání na data. (888 ms)

5.2 Výkon s velkým počtem instancí

Demonstrační stránky byly podrobeny i náročnějšímu testu v podobě 22 spuštěných instancí zároveň. Všechny patří jednomu uživateli. Nastavení chování stránek nebylo nijak změněno.

5.2.1 Pingdom

Pro porovnání byl opět použit nástroj Pingdom odesílající požadavky ze Švédska.

Přehled serverů

Celková doba od odeslání požadavku do kompletního zobrazení stránky byla horších 7,49 sekundy při celkové velikosti 159,3 kB. Z celkového času samozřejmě nejvíce zabralo čekání prohlížeče na data. (6,6 sekund)

Detail serveru

Stránka s detailem serveru nebyla touto akcí dotčena, což je podle očekávání. Stránka byla zobrazena za 1,91 sekundy při celkové velikosti 221,9 kB. Čas čekání na data byl 900 ms.

5.3 Výkon vzhledem ke granularitě

Vybraná granularita dat silně ovlivňuje celkový výkon. Gnocchi API data předá téměř stejně rychle pro různou granularitu. Úzkým hrdlem je zde komponenta v Javě, která přijatá data musí transformovat do podoby pro grafovou knihovnu. Pro test byl využit servlet vykreslující zadanou metriku se všemi daty bez časového omezení. Měněna byla jen granularita dat.

5.3.1 Pingdom

Pro porovnání byl opět použit nástroj Pingdom odesílající požadavky ze Švédska. Jak je vidět na obrázku níže, postupné zvyšování granularity znamená snížení času potřebného k zobrazení stránky. Granularita je uváděna v hodinách a lze pozorovat, že při hodnotách hodiny a vyšších se čas zobrazení ustálí kolem 300 ms.

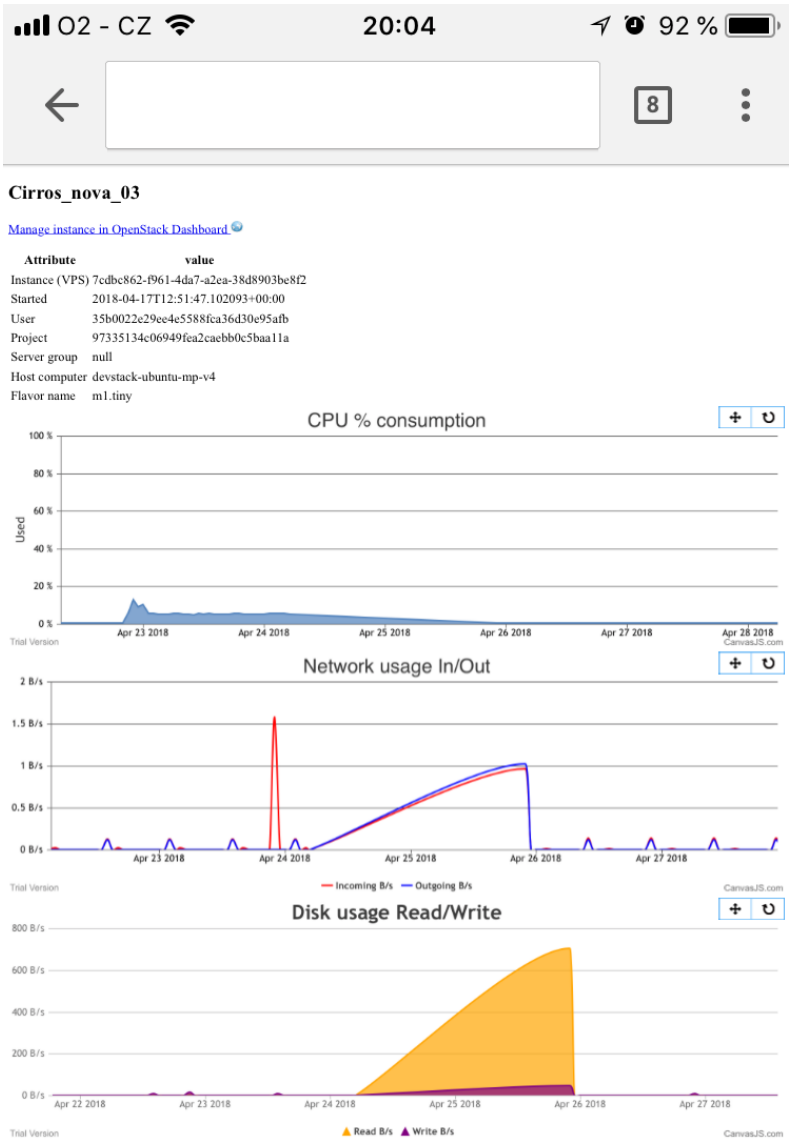
Obrázek 5.2: Závislost rychlosti na granularitě.

5.4 Mobilní zařízení

Testované stránky byly bez problému zobrazeny na všech testovaných zařízeních. Funkční je také přiblížení grafu (ruční výběr oblasti z grafu), pouze ovládací prvky pro návrat do původního zobrazení jsou příliž malé. Jelikož nebyly v zadání mobilní zařízení uvažovány, jde o zanedbatelný detail. Pro ukázku je na fotografii obrazovky 5.3 zobrazena stránka detailu s přiblíženými grafy.

Funkčnost byla testována na zařízeních:

- iPhone 7 – iOS 11.3.1, prohlížeč Safari.
- iPhone SE – iOS 11.3.1, prohlížeč Google Chrome 65.
- Samsung Galaxy S5 – Android 6.0.1, prohlížeč Google Chrome 66.
- Samsung Galaxy A3 – Android 7.0, prohlížeč Google Chrome 65.



Obrázek 5.3: Zobrazení na mobilním zařízení.

5.5 Hodnocení výsledků

Testování dopadlo dle očekávání. I přes několik úrovní virtualizace výsledného řešení (více v kapitole Realizace, v sekci 4.4.1), je doba odezvy velmi uspokojivá. Výsledná komponenta je tak schopna i vykreslování několika grafů v čase, který by pro samotný Ceilometer nedostačoval ani k základnímu dotazu.

Testování s velkým počtem instancí bylo ovlivněno malým množstvím nasbíraných dat o instancích, ale potvrdilo se, že doba odezvy především závisí na požadované granularitě dat a tedy na množství, které je zpracováváno

a odesíláno.

Pokud je preferencí výkon, je nejlepší variantou malý počet záznamů dodaných rozhraním Gnocchi za cenu nižší přesnosti. Pokud je požadována přesnost, je důležité si vyžádat více dat a čekat nutnou dobu na výsledek. Tak jako vždy je klíčové najít optimální poměr.

Závěr

Cílem této práce bylo vytvořit komponentu, integrovanou do prostředí správy serverů, která bude uživatelům služeb vykreslovat konkrétní grafy využití prostředků virtuálních serverů na platformě OpenStack. V úvodu této práce byly analyzovány možnosti sběru a ukládání dat o spuštěných instancích. Bylo zjištěno, jaké projekty je nutné zakomponovat do celkového řešení a jak zajistit prezentaci uživateli.

Byla navržena a implementována komponenta s možností vykreslování grafů využitím grafové knihovny. Dále byl zpracován popis nasazení všech navazujících projektů, které funkčnost umožňují.

Komponenta je dostupná přes webové rozhraní ve formě demonstračních webových stránek, které dokazují celkovou funkčnost, jelikož integrování do produkčního firemního řešení nebylo možné. Důvodem jsou rozdílné využívané verze softwaru OpenStack a návazných projektů.

V budoucnu je možné na práci navázat rozšiřováním funkcionality o vizualizaci událostí, dalšími parametry pro vykreslované grafy či idividualizací uživatelských preferencí.

Komponenta byla úspěšně realizována i přes velkou nestabilitu, nekompletní dokumentaci a uživatelskou nepřívětivost softwaru OpenStack. Z těchto důvodů bych software OpenStack označil spíše jako vhodný pro akademické a demonstrativní účely než jako klíčovou součást firemních systémů.

Realizace této diplomové práce pro mne byla velkým přínosem. Prošel jsem velké množství zdrojů a získal jsem nové poznatky o fungování cloud computingu. Také jsem se dozvěděl, jak fungují programy pod licencí OpenSource a přesvědčil se o síle komunity, která je s technologiemi neoddělitelně spojena. V neposlední řadě jsem si v praxi vyzkoušel fungování webových technologií a aplikačních rozhraní.

Literatura

[1] Co je cloud computing?: Průvodce pro začátečníky. In: Microsoft [online]. [cit. 2018-02-07]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-cloud-computing/

[2] Výhody a nevýhody cloudu – Jedox. In: NetMark [online]. 15.6.2016 [cit. 2018-02-08]. Dostupné z: http://www.netmark.cz/blog/item/4-vyhody-a-nevyhody-cloudu-jedox

[3] JOHNSTON, Sam. Cloud computing.svg. In: Wikimedia Commons [online]. 2009-03-03 [cit. 2018-02-15]. Dostupné z: https://commons.wikimedia.org/wiki/File:Cloud_computing.svg

[4] Introduction: A Bit of OpenStack History. In: OpenStack [online]. [cit. 2018-02-24]. Dostupné z: https://docs.openstack.org/project-team-guide/introduction.html

[5] Co je virtualizace?. In: WEDOS [online]. [cit. 2018-02-25]. Dostupné z: https://hosting.wedos.com/cs/virtual/co-je.html

[6] Bare Metal Service User Guide. In: OpenStack [online]. 2018-04-17 [cit. 2018-04-19]. Dostupné z: https://docs.openstack.org/ironic/pike/user/

[7] MESAROS, Relu. Difference between bare metal (hypervisor based) and host virtualization types. In: Stack Overflow [online]. 2016-07-03 [cit. 2018-03-25]. Dostupné z: https://stackoverflow.com/questions/38130453/difference-between-bare-metal-hypervisor-based-and-host-virtualization-types

[8] Cloud operating system. In: TWOTECHNOLOGYIBM [online]. [cit. 2018-03-25]. Dostupné z: http://twotechnologyibm.com/cloud-operating-system/

[9] VONDRA, Tomáš. Application deployment on OpenStack: A Workshop for LinuxDays 2017. In: Linux Days [online prezentace]. 2017, [cit. 2018-03-20]. Dostupné z: https://www.linuxdays.cz/2017/video/Tomas_Vondra-Vyuziti_Openstacku.pdf

[10] Gnocchi. In: OpenStack [online]. [cit. 2018-03-26]. Dostupné z: https://wiki.openstack.org/wiki/Gnocchi

[11] BEDYK, Witek a Roland HOCHMUTH. Log Management with Monasca. In: Youtube [online]. 2016-04-25 [cit. 2018-03-02]. Dostupné z: https://www.youtube.com/watch?v=ZPMkeNdM2d8

[12] MONASCA: monitoring. In: OpenStack [online]. [cit. 2018-03-23]. Dostupné z: https://www.openstack.org/software/releases/queens/components/monasca

[13] Comparisons To Alternatives. In: Gnocchi [online]. [cit. 2018-04-30]. Dostupné z: https://gnocchi.xyz/alternatives.html

[14] ŠÍN, Martin. Top, atop, htop a další kamarádi. In: LinuxExpress [online]. 2010-03-26 [cit. 2018-03-26]. Dostupné z: https://www.linuxexpres.cz/praxe/nastroje-top-atop-htop

[15] Permissions. In: Grafana Labs [online]. [cit. 2018-02-20]. Dostupné z: http://docs.grafana.org/administration/permissions/

[16] Sharing features. In: Grafana Labs [online]. [cit. 2018-02-20]. Dostupné z: http://docs.grafana.org/reference/sharing/

[17] Grafana Gnocchi datasource. In: Grafana Labs [online]. [cit. 2018-02-20]. Dostupné z: https://grafana.com/plugins/gnocchixyz-gnocchi-datasource

[18] Grafana Play Home. In: Grafana [online]. [cit. 2018-02-20]. Dostupné z: https://play.grafana.org/

[19] JavaScript Multi Series Charts & Graphs. In: CanvasJS [online]. [cit. 2018-03-26]. Dostupné z: https://canvasjs.com/javascript-charts/multi-series-chart/

[20] Line Chart. In: Google Charts [online]. 2017-04-14 [cit. 2018-03-26]. Dostupné z: https://developers.google.com/chart/interactive/docs/gallery/linechart

[21] Time Series Line Chart. In: Zing Chart [online]. [cit. 2018-03-26]. Dostupné z: https://www.zingchart.com/gallery/chart/#!time-series-line-chart

[22] ALMOSSAWI, Ali a Hamilton ULMER. Multiple lines. In: MetricGraphics.js [online]. [cit. 2018-03-26]. Dostupné z: https://www.metricsgraphicsjs.org/examples.htm#multilines

[23] KUNZ, Gion. CHARTIST - EXAMPLES. In: Chartist: SIMPLE RESPONSIVE CHARTS [online]. [cit. 2018-03-26]. Dostupné z: https://gionkunz.github.io/chartist-js/examples.html

[24] LOWE, Doug. Java all-in-one for dummies. 5th Edition. Hoboken, NJ: John Wiley, 2017. ISBN 11-192-4779-9.

[25] System Architecture. In: OpenStack [online]. 2018-04-02 [cit. 2018-04-16]. Dostupné z: https://docs.openstack.org/ceilometer/pike/contributor/architecture.html

[26] Telemetry best practices: Data storage. In: OpenStack [online]. 2018-04-02 [cit. 2018-04-11]. Dostupné z: https://docs.openstack.org/ceilometer/pike/admin/telemetry-best-practices.html

[27] Getting started. In: Gnocchi [online]. [cit. 2018-04-01]. Dostupné z: https://gnocchi.xyz/intro.html

[28] ROUSE, Margaret. Demon: definition. In: WhatIs.com [online]. 2010-11 [cit. 2018-04-01]. Dostupné z: https://whatis.techtarget.com/definition/demon

[29] Running Gnocchi: How to define archive policies. In: Gnocchi [online]. [cit. 2018-03-27]. Dostupné z: https://gnocchi.xyz/operating.html?highlight=archive

[30] 1. díl - Úvod do JavaScriptu. In: ITnetwork.cz [online]. [cit. 2018-04-07]. Dostupné z: https://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk

[31] HTML5 Canvas. In: W3schools.com: THE WORLD'S LARGEST WEB DEVELOPER SITE [online]. [cit. 2018-04-08]. Dostupné z: https://www.w3schools.com/html/html5_canvas.asp

[32] CanvasJS. In: CanvasJS [online]. [cit. 2018-04-08]. Dostupné z: https://canvasjs.com/

[33] DevStack. In: OpenStack [online]. 2018-04-04 [cit. 2018-04-12]. Dostupné z: https://docs.openstack.org/devstack/latest/

[34] DevStack. In: OpenStack [online]. 2018-04-13 [cit. 2018-04-14]. Dostupné z: https://docs.openstack.org/devstack/queens/

[35] COOPER, Stephen Byron. What Is a Listening Port?. In: It Still Works [online]. [cit. 2018-03-15]. Dostupné z: https://itstillworks.com/listening-port-8721837.html

[36] MOSER, Scott. Download. In: CirrOS [online]. 2011-10-07 [cit. 2018-03-22]. Dostupné z: https://launchpad.net/cirros

[37] Servlet Directory Structure: Directory Structure of Servlet Application. In: Sitesbay [online]. [cit. 2018-04-12]. Dostupné z: https://www.sitesbay.com/servlet/servlet-directory-structure

[38] Pingdom. In: Pingdom [online]. [cit. 2018-04-30]. Dostupné z: https://www.pingdom.com/

[39] Frequently Asked Questions. In: PageSpeed Tools [online]. 2018-01-10 [cit. 2018-04-30]. Dostupné z: https://developers.google.com/speed/docs/insights/

[40] Design. In: OpenStack [online]. 2018-04-16 [cit. 2018-04-19]. Dostupné z: https://docs.openstack.org/arch-design/design.html

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

CERN Conseil Européen pour la recherche nucléaire

CPU Central processing unit

GB Gigabyte

GUI Graphical user interface

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

IaaS Infrastructure as a service

ICT Information and Communication Technologies

ID Identifier

IP Internet Protocol address

IT Information Technology

JS JavaScript

JSP JavaServer Pages

KiB Kibibyte

NASA National Aeronautics and Space Administration

NSA National Security Agency

OS Operating System

Paas Platform as a service

PDF Portable Document Format

REST Representational State Transfer

SaaS Software as a Service

SSH Secure Shell

SVG Scalable Vector Graphics

SQL Structured Query Language

VM Virtual machine

VML Vector Markup Language

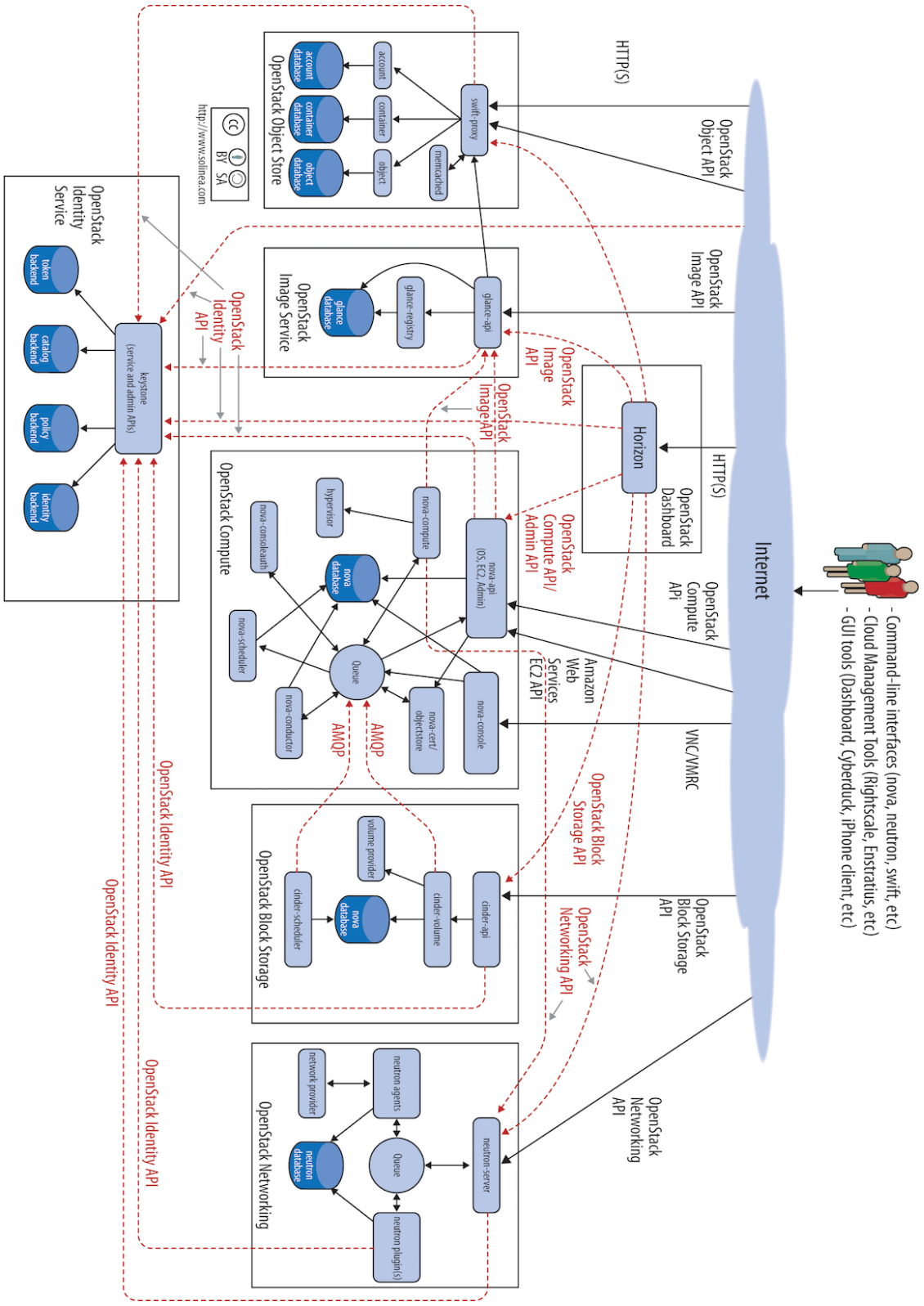
VNC Virtual Network Computing

VPS Virtuální privátní server

XML Extensible markup language

Příloha B

Detailní architektura OpenStacku



Obrázek B.1: Propojení projektů, včetně jejich součástí.[40]

Příloha C

Podrobný popis nasazení

C.1 OpenStack

1. Stažení obrazu Ubuntu Server 16 Xenial a instalace.
2. Update a upgrade systému a instalace programů. – aptitude, build-essential, git, ntp, ntpdate, openssh-server, python-dev, sudo, pip, htop.
3. Stažení devstacku z GITu.
4. Vytvoření nebo úprava souboru local.conf v domovském adresáři, ve složce devstack. Hlavní je přidání adresy na které bude OpenStack provozován a povolení dodatečných služeb a modulů. (Nebo použít soubor local.conf přiložený na CD.)
5. Po přípravě se spustí instalace pomocí skriptu. Jde o časově náročnou operaci maximálně na hodinu.

Toho lze docílit postupným použitím příkazů: (Na funkčním Ubuntu.) Po

```
$ cd ~
$ sudo apt-get update
$ sudo apt-get install aptitude build-essential git ntp ntpdate
      \ openssh-server python-dev sudo pip htop
$ git clone https://git.openstack.org/openstack-dev/devstack
      \ -b stable/pike
$ cd devstack
$ cp samples/local.conf local.conf
$ pico local.conf
$ ./stack.sh
```

Obrázek C.1: Příprava systému a instalace OpenStacku.

dokončení je vypsána celková doba instalace a některé přístupové body. Například projekt Horizon dostupný na `http://127.0.0.1/dashboard`, nebylo-li definováno v `HOST_IP` jinak.

Pokud je dashboard dostupný, OpenStack je připraven k použití. V případě, že by web dostupný byl, ale místo přihlašovací obrazovky by se zobrazovala chyba komprese, je nutné odmazat v souboru `/opt/stack/horizon/openstack_dashboard/templates/horizon/_scripts.html` Horizonu řádek obsahující `{% compress %}` a `{% endcompress %}`.

Pro práci v příkazové řádce instalovaného systému lze dodat proměnné prostředí s nastaveními pro používané příkazy:

1. Soubor obsahující tato nastavení je dostupný ke stažení v dashboardu Horizon po přihlášení uživatele (personalizované nastavení) v sekci */Project/ API access/ download OpenStack RC file (Identity API v3).*
2. Stažený soubor `admin-opensrc.sh` zkopírovat nebo vytvořit v systému.
3. Soubor se nahrává do prostředí příkazem níže, pro používání také Gnocchi z příkazové řádky se přidávají další dvě proměnné:

```
$ . admin-openrc.sh
$ export OS_AUTH_TYPE=password
$ export OS_TENANT_NAME=admin
```

Obrázek C.2: Export nastavení pro příkazovou řádku.

Při pokusu o export je vyžadováno heslo uživatele. Pokud je vše správně nastaveno, lze provádět všechny úkony správy z příkazové řádky. Vytvořit (a spustit) novou instanci lze příkazem:

```
$ nova boot --flavor m1.tiny --image cirros-0.3.5-x86_64-disk \
       --nic net-name=net1 --security-group default \
       My_CirrOS_server_007
```

Obrázek C.3: Příklad spuštění nové instance.

C.2 Port forwarding a donastavení Gnocchi

Pro virtuální stroj na kterém jsou provozovány všechny komponenty je nutné povolení následujících portů se směrováním na adresu počítače.

- Port 22 – SSH
- Port 80 – Apache
- Port 6080 – VNC
- Port 8080 – Apache Tomcat

C.2.1 OpenStack instalován na VM v OpenStacku

Práce byla vyvíjena ve virtualizovaném prostředí serveru společnosti Homeatcloud.cz. Jde o přidanou úroveň virtualizace z důvodu snažšího vývoje odděleného od produkčního prostředí. Jednotlivé vrstvy tedy jsou:

1. Fyzický server společnosti.
2. Produkční OpenStack společnosti.
3. Virtuální server Ubuntu.
4. OpenStack pro vývoj práce.

Na úrovni produkčního OpenStacku společnosti se nachází dashboard Horizon – v sekci *Compute/Access & Security/Seciruty groups/default/manage rules* lze přidat porty a směrování. Vždy pomocí tlačítka *Add rule* a příslušné aplikace. Port 22 volbou *Rule/SSH*, atd.

C.2.2 VirtualBox

V případě instalace v nástroji Oracle VM VirtualBox je přidání portu v nastavení vybraného počítače. V položce *síť/karta 1/pokročilé/předávání portů* lze přidat příslušné porty.

C.2.3 Start Gnocchi

Dodržený postup připraví všechny součásti. Bohužel ale nejspíše zůstanou ve stavu, kdy Gnocchi není zásobováno daty z Ceilometeru. (Otestovat výpisem metrik či naměřených hodnot.) Pro správnou funkci lze použít příkazy pro upgrade:

```
$ gnocchi-upgrade
$ ceilometer-upgrade --skip-metering-database
```

Obrázek C.4: Aktualizace Gnocchi a Ceilometeru.

Po provedení už ve výpisech metrik či zdrojů budou uvedeny všechny správné údaje.

C.3 Aplikace

C.3.1 Apache Tomcat

Instalace serveru Apache Tomcat vybraného pro spouštění Java Servletů bylo provedeno z repozitářů Ubuntu. Postup je pro uživatele jednoduchý. Stačí příkaz:

```
$ apt-get install tomcat8
$ apt-get install tomcat8-docs tomcat8-examples tomcat8-admin
```

Obrázek C.5: Instalace Apache Tomcat 8

Pokud by si uživatel přál i lokální dokumentaci, příklady používání nebo webové rozhraní pro administraci serveru Tomcat, může volitelně přidat:

```
$ apt-get install tomcat8-docs tomcat8-examples tomcat8-admin
```

Obrázek C.6: Instalace Apache Tomcat 8 rozšíření.

Funkčnost lze otestovat přístupem na adresu počítače, na defaultním portu 8080.

C.3.2 Nasazení aplikace

Celá zdrojová složka aplikace (název například *graph*) musí být umístěna v adresáři `/var/lib/tomcat8/webapps`. V aplikační složce, v cestě `src/configuration` se nalézá soubor `TokenRequestBody.txt`, kde je vzor požadavku o vystavení tokenu pro aplikaci – zde nastavit vlastní uživatelské jméno, heslo a ID projektu, které lze zjistit v OpenStacku.

Pro kompilaci dalších vlastních tříd nebo úpravy stávajících lze použít připravený sript `src/java_compile_servlet.sh` a přidat jako parametr název souboru k překladu.

Pokud je aplikace provozována na stejném serveru jako Gnocchi, není potřebná další úprava. Úvodní stránka aplikace bude poté dostupná na adrese `HOST_IP:8080/graph/`.

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `graph` ..... zdrojové kódy Java komponenty
    - `config` ..... konfigurační soubory OpenStack systému
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_Pavelek_Martin_2018.pdf` ..... text práce ve formátu PDF
  - `ZadaniDP_Pavelek_Martin_2018.pdf` .. zadání práce ve formátu PDF