АРХИТЕКТУРА НА ОБЛАЧНИ ОПЕРАЦИОННИ СИСТЕМИ

докторант Невян Нейков, Икономически университет – Варна
e-mail: nevyan_neykov@ue-varna.bg

1. Въведение

Облачните изчисления подпомагат създаването на големи, еластични системи за обработка на данни. В случаите, когато изискванията на потребителите са ниски, системите автоматично намаляват консумираните от тях ресурси, като при нужда ги разрастват за да отговорят на необходимите възникнали потребности. Обикновено практическото използване на облачни ресурси изисква създаването на специална мрежа (Grid) от виртуални системи. Изграждането на подобни системи се извършва, чрез използване на процеса виртуализация, с помощта на технологията хипервайзор, позволяваща изграждането и стартирането на множество виртуални системи на един и същи физически сървър. Облачните изчислителни системи консумират ресурси само при необходимост, които в последствие биват освободени. Това ги прави удобни за употреба на разнороден хардуер, като минимизирайки загуби произтичащи от излишък на ресурси. Все повече доставчици на облачни услуги предлагат предварително тестване на продуктите, което дава на потребителите възможност за тяхното опознаване.

2. Същност на облачните изчисления

Виртуализация – софтуерна технология, която чрез споделяне на ресурси, позволява създаването на виртуални версии на: хардуерни устройства и софтуер. Това прави възможно стартирането на множество операционни системи и приложения на един сървър по едно и също време.

Виртуална машина – софтуерна абстракция на хардуер, управлявана от операционна система, използваща в основата си хипервайзори. Различават се два

вида виртуални машини – гост и домакин. Домакина представлява първоначално инсталираната(първична) операционна система, госта е виртуалната машина, която е инсталирана върху първичната операционна система.

Хипервайзор – форма на виртуализация, емулираща ресурси на хардуерно ниво, даваща възможност да се създават и стартират виртуални машини, на които се инсталират множество операционни системи. Тези операционни системи споделят виртуализирани хардуерни ресурси. Примери: Xen, VirtualBox, KvmM, Wmware, Hyper-V.

Контейнер – форма на виртуализация, емулираща ресурси на ниво на операционна система(ядро). По този начин се използват по-малко ресурси, което от своя страна позволява опаковането и работата на много приложения на един сървър. Контейнерите не правят абстракция на целия хардуер, като използват само нужните за приложението ресурси, останалите остават налични за другите контейнери. Това води до повишаване на тяхната скорост на стартиране и работа. Примери: Docker, LXD

Основни разлики между понятията клъстер, грид и облак: Клъстерът представлява група от еднородни с подобни хардуерни конфигурации компютри свързани посредством локална мрежа. При грида и облака тази мрежа може да бъде разнородна, там машините са разпространени в по-широк мащаб и са географски разпръснати. Облачните изчисления имат централизиран модел на разпределение на ресурсите, докато при грида модела е децентрализиран измежду множество административни домейни. При облачните изчисления компютрите се притежават от една организация, докато при в грида те са собственост на много лица.

Облачните изчисления включват в себе си, както приложенията, предоставяни като услуги, така и хардуерните и софтуерните ресурси намиращи се в центровете за данни. Обикновено всички тези услуги се предлагат публично и се наричат Софтуер като услуга (SaaS). В тях се включват облачно-базирани операционни системи, които представляват вид операционна система,

конструирана за работа в облачни и виртуализирани среди. Тя управлява действието и изпълнението и процеси на виртуалните машини, виртуални сървъри и виртуална инфраструктура, както и хардуерни и софтуерни ресурси. Примери за подобни облачно-базирани ОС са: Microsoft Windows Azure и Google Chrome OS. Техните ресурси са достъпни за потребителите. Облачните операционни системи се наричат още Web OS, защото с помощта на браузър могат да бъдат управлявани, да се стартират апликации или дори цялостни операционни системи. Данните им се съхраняват в Интернет и са налични по всяко време и място. Обикновено публичните облачно базирани операционни системи предоставят параметри за ползване като: 1GБ до 30GB свободно място и възможност за връзка с други доставчици на услуги за съхранение като: DropBox, Google Drive, SkyDrive, 4Shared и др. От приложенията са налични инструменти за редакция на текст, снимки, документи, таблици и съобщения, календар, бележки, презентации, файлов мениджър, ftp клиент, аудио плеър, емейл. Допълнително се предлага синхронизация на файлове с водещи операционни системи. Облачно базираните ОС са полезни за хора, които са в постоянно движение и работят с различни компютърни и мобилни устройства. Самата web операционна система не изисква много ресурси за стартирането си, като това я прави приложима за използване от нетбуци, таблети и по-стари компютри, поради извършването на операции в облака и връщането на техният резултат. Примери за публични облачно базирани системи са:

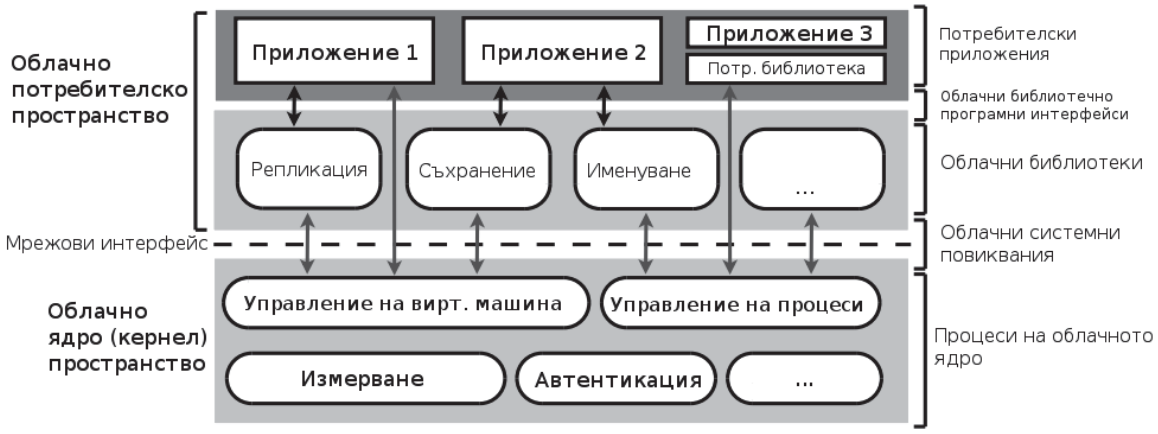
MS Office online: Word, Excel, PowerPoint, OneNote запис на данни в OneDrive

Google: Gmail, Calendar, Docs, Sheets, Slides – запис на данни в Google Drive

Apple iCloud: mail, contacts, calendar: Pages, Numbers., Keynopte – запис на данни в iCloud

хибридни – DropBox – локално и облачно съхранение на данни

Ще разгледаме подробно архитектурата на една облачно базирана операционна система и нейното функциониране – фиг. 1. При облачните операционни системи дадена компютърна машина се обозначава с понятието възел и се идентифицира с произволен идентификатор с цел минимизиране на риска й от колизии с други машини.



Фиг. 1. *Логическа архитектура на облачна операционна система*

Облачен обект – съвкупност от локални процеси принадлежащи на операционната система.

Облачен процес – набор от облачни обекти, от които в последствие се изгражда едно приложение. Трябва да се има в предвид, че в една облачна операционна система изчислителните облачни процеси са винаги активни и работещи.

В рамките на пространството на ядрото се управляват процеси като: физическа алокация, контрол на достъп и измерване на ресурси. Всички останали процеси се обхващат от потребителското пространство.

Облачни процеси, изпълнени директно от потребители се наричат потребителски приложения. Облачните библиотеки от своя страна представляват

облачни процеси, извиквани от потребителски приложения или от други библиотеки.

Приложенията комуникират с библиотеките и ядрото на облачните процеси чрез мрежа, посредством стандартни интерфейси наречени облачни системни повиквания. Обектите намиращи се в облачното потребителско пространство имат достъп до управлението на тези облачни системни повиквания, като прихващат сигнали от облачната операционна система и са достъпни през мрежови интерфейс за извършване на процеси по управление.

Ядрото на облачните процеси и по-конкретно управлението на процеси и виртуални машини управляват връзките между имената на обектите, техните мрежови адреси и портове, като резултатната информация става достъпна из целия облак посредством библиотека на имената. Тя проследява връзките между процесите на потребителските облачни приложения и съставните им обекти.

Автентикационното ядро на облачните процеси раздава и проверява правата за достъп необходими на операциите по управление.

Измерване на ресурси: OS поддържа броя на наличните облачни ресурси, чрез извършване на локални измервания на всеки един облачен възел. Вземат се в предвид променливи като: латентност, ширина на мрежова лента, загуба на пакети, които са ценни ресурси на знание и се използват директно от приложенията.

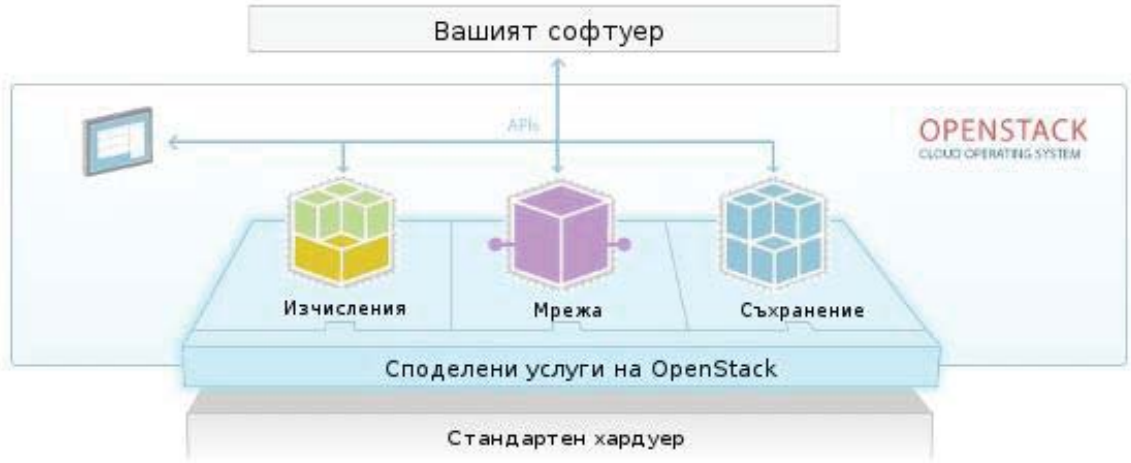
Разпределено управление на процеси и приложения: Облачната операционна система създава и управлява всеки един съществуващ обект в облачните възли. За тази цел, като обща практика се използват виртуални машини, доставящи абстракция, която гъвкаво отделя логическите изчислителни ресурси от изграждащите ги физически облачни възли. Виртуализацията предоставя характеристики, нужни за облачната среда, като: поддръжка на множество операционни платформи работещи върху един и същ компютър и до известна степен изолация между процеси стартирани от различни виртуални машини, намиращи се на един и същи хардуер. Оптимизационните изисквания,

като: изчислителна еластичност и балансиране на товар внасят нужда от динамично алокиране на ресурси. Например прехвърлянето на един работещ процес между два облачни възела, се осъществява или на ниво на облачни процеси – мигриране на единични процеси между възли или на ниво виртуални машини – съхранение и възстановяване на цялостното състояние на виртуалната машина върху друг възел. Операционната система 'гост' се намира на слой над хипервайзора, който посредством файлово-системен интерфейс достъпва функциите предоставяни от операционната система 'домакин'.

Облачната операционна система, допълнително предоставя на потребителите интерфейс за управление на процеси, посредством абстракция с цел обединение на всички различни изчислителни ресурси в едно. Облачните библиотеки предоставят стандартни API интерфейсни характеристики като: достъп до обекти из целия облак, именуване на процеси посредством DNS, разпределена и надеждна функционалност по съхранение, автоматизирано изграждане на облачни приложения, хоризонтално скалиране и управление на жизнен цикъл, висока достъпност, поддръжка при грешки с възможност за репликиране на изпълнение на процеси. Облачните библиотеки дават на разработчиците контрол върху нужното им ниво на репликация, консистентност и достъп, както и начини за справяне в случай на провал, когато изискванията на приложението не са задоволени. По този начин разработчика на приложения концентрира вниманието си върху определени черти на приложението, знаейки, че системата ще направи най-доброто за да осигури заявените изисквания. Ползите са двустранни: от една страна разработчика, не е задължен да конструира собствени библиотеки специално за приложението, а само да настройва ниво на поддръжка от което има нужда посредством API интерфейс на облачната библиотека. По този начин се намалява сложността и настъпването на грешки при създаването на приложения. От друга страна може повторно да се използват компоненти на облачни приложения, като се адаптират за задоволяване на различни спецификации, обновявайки много малка част от параметрите на самата облачна библиотека.

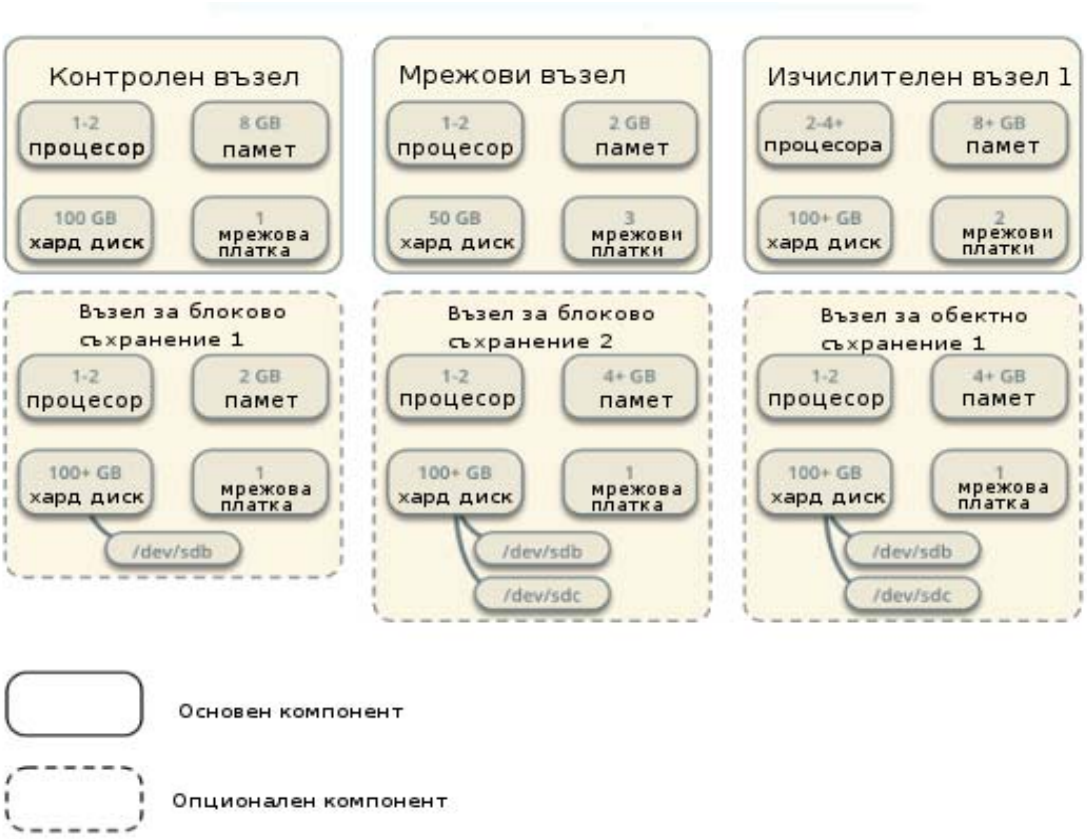
3. OpenStack – инфраструктурен софтуер за облачни изчисления, използващ Apache лиценз

OpenStack е основан от NASA и RackSpace, представлява софтуер управляващ облачни изчисления, който се използва от големи фирми, университети и институции. Факт е, че 55% от OpenStack продукционните облаци по света работят под Линукс Убунту. OpenStack не доставя само услуги, обслужвайки системи или центрове за данни, а представлява софтуер с отворен код за създаване на публични, частни и хибридни облаци. По този начин продуктът позволява лесно преместване на данни и услуги от една към друга фирма, както и разпределяне и използване на общи услуги конкурентно от различни фирми. Концептуалната архитектура на софтуерния продукт е представена на фиг. 2.



***Фиг. 2** Концептуална архитектура на OpenStack*

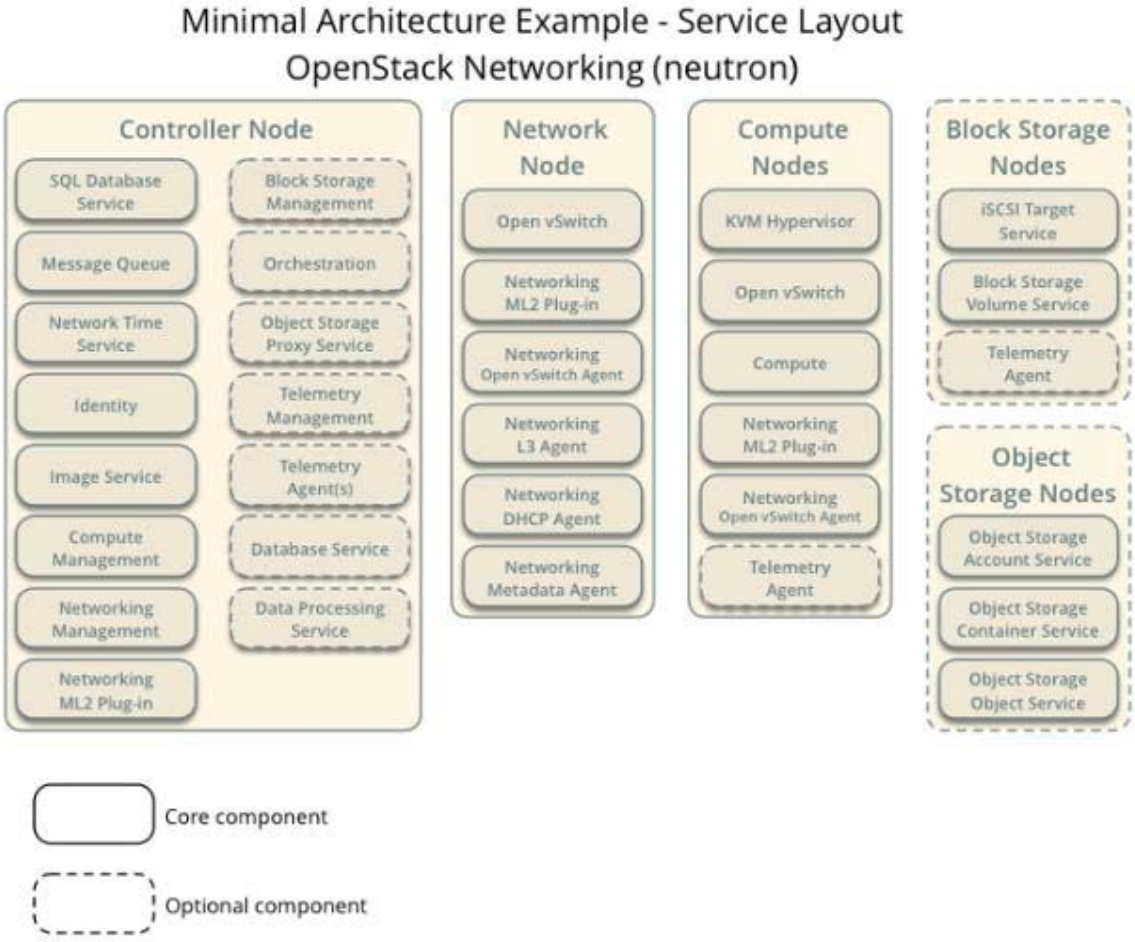
Следва минималната хардуерна конфигурация, която се изисква за стартиране на продукта, включваща по един компютър за контролен, мрежови и



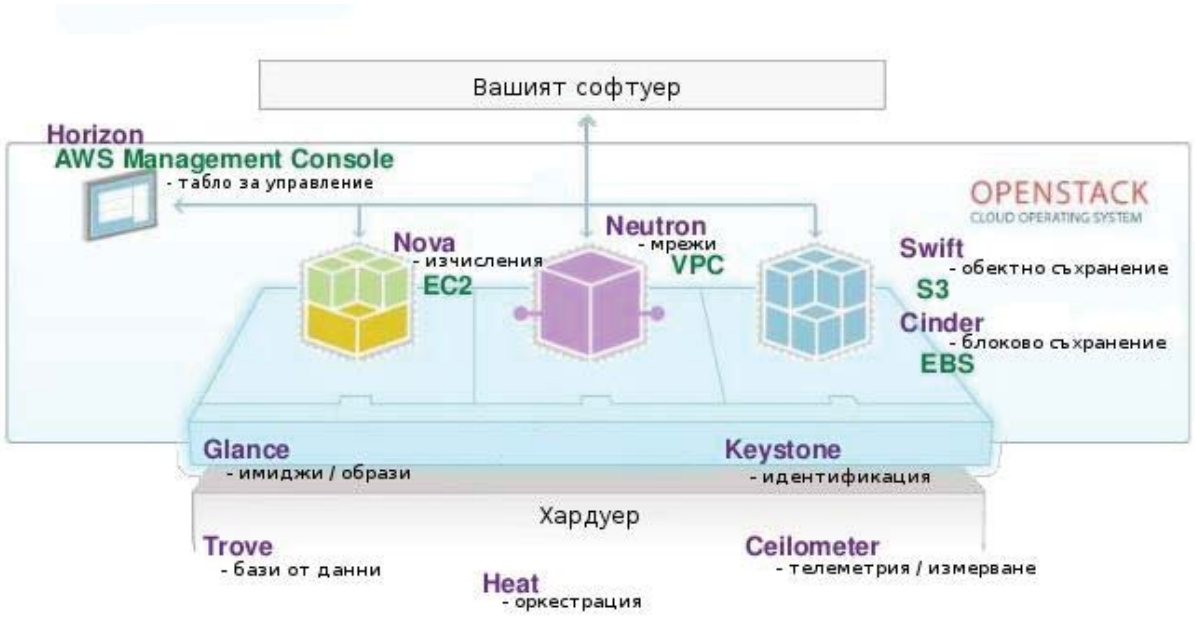
изчислителен възел - фиг. 3.

Фиг. 3 *Минимални хардуерни изисквания на архитектурата на Neutron*

Проектът OpenStack доставя облачни решения от тип Инфраструктура като услуга (IaaS) чрез различни услуги. Всяка една услуга се предлага за достъп чрез включен софтуерен програмен интерфейс (API), който улеснява интеграцията им. OpenStack е конфигуруем, за да може да посрещне нуждите от различни компютърни, мрежови и дискови опции. Стартирането на виртуална машина или т.нар. инстанция включва в себе си взаимодействия измежду множество услуги. Следва описание на основните услуги предлагани от неговата софтуерна платформа представени на фиг. 4 и фиг.5.



Фиг. 4. *Компоненти и услуги на OpenStack базиран на Neutron*



Фиг. 5. *OpenStack услуги*

Навигационна среда: Horizon: Доставя уеб базиран портал за самообслужване, който взаимодейства с услугите предлагани от OpenStack, като стартиране на инстанция, назначаване на IP адреси и конфигуриране на контрол на достъп.

Изчисления: Nova: Управлява цикъла на живот на отделните изчислителни инстанции (компютри). Отговаря за създаването, разпределението и разрушаването на виртуални машини според предварително определени изисквания. *Компютърният възел* стартира хипервайзорната (KVM) част от Compute, която от своя страна стартира виртуални машини или инстанции. Тук се стартира също и мрежовия плъгин, който не само свързва мрежи, но и доставя услуги, като защитни стени. Може да се стартира повече от един Compute възел. Допълнително, Compute възела включва агент по Телеметрия, който събира метрики, както и трети мрежови интерфейс към отделна мрежа, за да подобри производителността на услугите по съхранение на информация.

Мрежова свързаност: Neutron: Позволява свързаност, като услуга към други услуги на OpenStack. Доставя API за потребителите за определяне на мрежи и прикачвания към тях. Има pluggable архитектура, която поддържа множество мрежови технологии. *Мрежовия възел* включва в себе си мрежови плъгин и различни агенти, които дават възможност за превключване, рутиране, преадресиране, както и услуги по динамично раздаване на IP адреси. Този възел също се занимава с външната свързаност между отделни виртуални инстанции.

Съхранение на информация

Запазване на обекти: Swift: Запазва и извлича произволно неструктурирани обекти посредством REST HTTP API. Доставя висока толерантност при репликация на грешки и мащабиране на архитектурата. Опционалния Object Storage възел включва дискове за съхранение на акаунти, контейнери и обекти. Минималната архитектура изисква съществуването на повече от два възела.

Блоково съхранение: Cinder: Доставя непрекъснато блоково съхранение към стартираните инстанции. Неговият драйвер ускорява създаването и управлението на блокови устройства за съхранение на информация. Опционалния Block Storage възел включва дискове, които се използват от включените виртуални машини. Може да се стартира повече от един от този възел.

Споделени услуги

Идентифицираща услуга: Keystone: Доставя автентикация и оторизация към други услуги на OpenStack, както и каталог от крайни точки съдържащ всички предлагани услуги.

Имидж услуга: Glance: Запазва и извлича дисковите имиджи на виртуалните машини. Използва се от Compute при стартирането на услуги.

Телеметрия: Celiometar: Прави мониторинг и измерва облака за производителност, мащабиране, заплащане и статистически цели.

Услуги от високо ниво

Оркестрация: Heat: Оркестрира множество съставни облачни апликации, като използва шаблонни формати захранвайки REST API на OpenStack

Бази от данни: Trove: Доставя мащабиращи и надеждни услуги предаващи функционалност по бази от данни.

Концепция за идентификация

Идентификацията представя пред OpenStack даден човек, система или

Фиг. 6. *Процес на идентификация в OpenStack – Keystone*

услуга, използващи облачните услуги, като валидира входящите заявки на потребителя, който твърди че прави запитването. Услугата по идентификация проследява потребителите и техните позволения, като доставя каталог съдържащ наличните им услуги и техните крайни API точки за достъп. Всяка една услуга е нужно да се инсталира посредством Identity service, която дава възможност да се проследят всички инсталирани услуги и тяхното месторазположение в мрежата. Към потребителите се добавят тоукъни, за да осъществят достъп до ресурси. Допълнително потребителите, могат директно да се присъединяват, към определен ползвател за придобиване на поведение (пример: достъп до ресурси), присъщо за този тип ползвател. Следва описание на използваните понятия предлагани от платформата за идентификация:

Права за достъп

Това да данни, които потвърждават идентичността на даден потребител. Например: потребителско име, API ключ и парола, или автентикиращ токън доставен от Услугата по Идентичност.

Автентикация

Процес при който се потвърждава идентичността на даден потребител. Услугата Identity потвърждава входящите заявки, валидирайки определен набор от credentials подадени от потребителя, които биват: потребителско име и парола или потребителско име и API ключ. След тяхната валидация, OpenStack Identity издава автентикационен токън, който потребителя представя в последващи заявки – фиг. 6 .

Токън

Низ, състоящ се от букви и цифри, с който се осъществява достъп до API и ресурсите на OpenStack. Той може да бъде премахнат по всяко едно време и е валиден за определен краен момент от време.

Ползвател

Контейнер, който се използва за да групира или изолира ресурси или обекти по идентичност. В зависимост от оператора на услугата даден ползвател може да бъде прикачен към клиент, акаунт организация или проект.

Услуга

Услугите в OpenStack са: Compute (nova), Object Storage (swift), Image Service (glance). Те доставят една или повече крайни точки, през които потребителите имат достъп до ресурси и могат да извършват операции.

Крайна точка

Достъпен през мрежата адрес, от където може да извика услуга, обикновено представлява URL адрес.

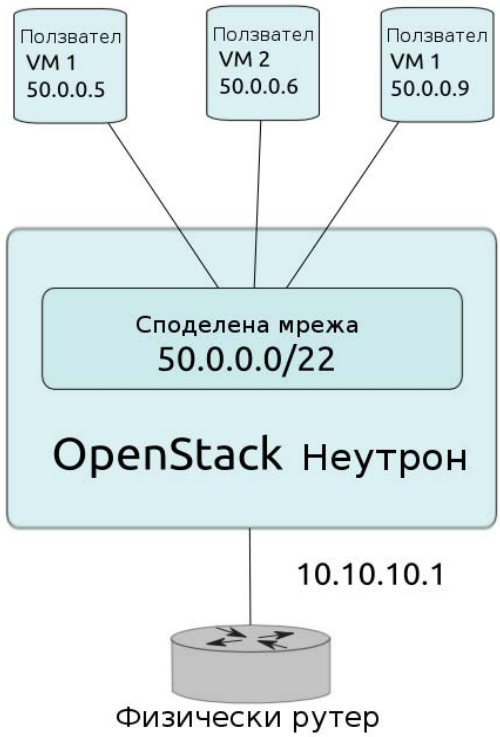
Роля

Персоналност с дефиниран набор от правила или привилегии, служеща за изпълнението на определен брой операции. Услугата Identity, издава токън за даден потребител, който включва в себе си списък с роли. Услугите, които се извикват от даден потребител определят: как да интерпретират неговия набор от роли и по-точно за кои операции и ресурси всяка роля му дава достъп.

Мрежови концепции

Мрежовата свързаност на OpenStack Neutron управлява всички мрежови аспекти от виртуалната и нивата на достъп на физическата мрежова инфраструктури – фиг. 7. Тази свързаност дава възможност на ползвателите да създават сложни виртуални мрежови топологии, включващи услуги, като: защитни стени, балансиране на натоварване и виртуални частни мрежи – фиг. 8.

Фиг. 7. *Мрежова архитектура управлявана от Neutron*



Фиг. 8. *Примерна мрежова архитектура - Neutron*

Мрежовата свързаност предоставя абстракции на мрежи, под-мрежи, рутери и обекти. Всяка една подобна абстракция притежава приличаща на физическата функционалност, където от мрежи се създават подмрежи, а чрез рутери се пренасочва трафик между различни подмрежи. Всеки един рутер има gateway, който е свързан към дадена основна мрежа, както и множество интерфейси свързани към подмрежи. По този начин подмрежите имат възможност за достъп до машини от други подмрежи, към които рутерът е свързан.

Всяка една инсталирана мрежа съдържа в себе си поне една външна мрежа. Важно е да се знае, че тя не е виртуална, а представлява гледка към реална физическа, външна мрежа, достъпна извън инсталацията на OpenStack. Поради това, че нейните IP адреси са достъпни от всеки, който се намира извън

инсталацията на OpenStack, при нея протокола за раздаване на динамични адреси DHCP е забранен.

Освен това, всяка една мрежова свързаност съдържа една или повече вътрешни мрежи. Към тези софтуерно-дефинирани мрежи директно се свързват виртуални машини (ВМ). За достъп от външна мрежа до виртуалните машини и обратно са нужни рутери между мрежите. Виртуалните машини от една мрежа са директно достъпни единствено от ВМ на всяка вътрешна мрежа или от подмрежи свързани през интерфейси на рутери. Всеки един рутер притежава gateway, който е свързан към мрежата и множество интерфейси, свързващи подмрежите. Подобно на физически рутер, ВМ от една подмрежа имат достъп до ВМ на друга подмрежа, свързана към същия рутер. Чрез gateway-а интерфейса на рутера ВМ имат достъп до външни мрежи.

Всяка една връзка към дадена подмрежа се нарича 'порт'. Допълнително, може да се разпределят IP адреси от портове на външни към вътрешни мрежи, както и да се асоциират външни мрежови IP адреси с определени портове към виртуални машини. По този начин компютри от външната мрежа могат да имат достъп до виртуалната машина.

Групи по сигурност

Те дават право на администраторите да дефинират правила на защитни стени и да ги запазват в групи. Една виртуална машина може да принадлежи към една или повече групи за сигурност. Частта Neutron Networking прилага правила в тези групи по сигурност, които блокират или отварят портове или тип трафик специфичен за тази виртуална машина.

4. Заключение

Облачните услуги навлизат в живота на всеки Интернет потребител. До този момент те представляват предизвикателство не само към ползвателите, но и към хората разработващи подобни системи, които желаят с тяхна помощ плавно да заменят стандартните съществуващи архитектури. Описаната примерна архитектура на облачно-изчислителна инфраструктура е изградена на база на софтуерната платформа OpenStack. Възприетия в нея архитектурен модел позволява функционален достъп и обработка на данни при поискване, използвайки ресурси в зависимост от възникнали потребности. Това от своя страна води до оптимизация на цялостната производителност и употреба на дадена услуга, наред с намаляване на съпътстващите я финансови разходи.

Използвана литература

1. Hemke M., Ubuntu Unleashed 2015 Edition: Covering 14.10 and 15.04, SAMS, Pearson Education
2. OpenStack Installation Guide for Ubuntu 14.04, OpenStack Foundation, docs.openstack.org/juno/install-guide/install/apt/openstack-install-guide-apt-juno.pdf, 7-08-2015
3. Doyle J., Shorten S., O'Mahony D., Stratus: Load Balancing the Cloud for Carbon Emissions Control, IEEE: Transactions on Cloud Computing, http://www.tara.tcd.ie/bitstream/handle/2262/67834/Stratus_v2.pdf
4. Poelker C., http://www.computerworld.com/article/2473557/data-center/will-cloud-computing-kill-the-storage-area-network-.html, 11-12-2012
5. Ubuntu in the cloud, http://www.ubuntu.com/cloud
6. Cloud computing for business: What is cloud, http://www.opengroup.org/cloud/cloud/cloud_for_business/what.htm
7. Matthews R., How Environmentally Sustainable is Cloud Computing and Storage?, 09-12-2013,

http://globalwarmingisreal.com/2013/09/12/sustainable-cloud-computing/

8. Pillai S., Difference between Hypervisor Virtualization and Container Virtualization http://www.slashroot.in/difference-between-hypervisor-virtualization-and-container-virtualization, 10-21-2014
9. Pianese, Fabio, et al. "Toward a cloud operating system." Network Operations and Management Symposium Workshops (NOMS Wksps), IEEE, 2010, http://www.moritzsteiner.de/papers/CloudOS.pdf