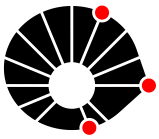

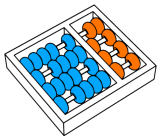

Soluções para Cloud Vendor Lock-In

M. M. Kobuchi *L. F. Bittencourt*

Relatório Técnico - IC-PFG-17-14
Projeto Final de Graduação
2017 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Soluções para Cloud Vendor Lock-In

Marcos Massayuki Kobuchi* Luiz Fernando Bittencourt*

Resumo

Este trabalho é o relatório de um projeto visando a construção de uma interface única de acesso a serviços de armazenamento, em que a AWS S3, Azure Blob Storage e OpenStack Swift são os mais conhecidos. Com o uso de framework já existente desenvolvido em .NET, implementado para a AWS S3, Azure Blob Storage e Google APIs, este trabalho busca iniciar os estudos necessários para a implementação de mais um serviço, o OpenStack Swift. Com base nesses estudos, constatamos a existência de um framework não oficial que permite a interação com a OpenStack API, entretanto, verificamos que se encontrava desatualizado, com uma branch de atualização já em andamento. Para contornar esse problema, provisoriamente, estudou-se as interações com a API através de requisições HTTP.

1 Introdução

O conceito de computação em nuvens trilhou um longo caminho desde que surgiu o primeiro computador, em 1946, pouco após o final da segunda segunda mundial [1]. J.C.R. Licklider [2], um dos responsáveis pelo desenvolvimento da ARPANET, o predecessor direto da Internet, já envisionava na década de sessenta o conceito de "Rede Galáctica", um conjunto globalmente interconectado de computadores através do qual todos poderiam acessar rapidamente dados e programas de qualquer site, um conceito bem parecido com o que chamamos de computação em nuvens hoje [3].

Mas como a internet até então era bastante restrita, o desenvolvimento deste conceito só começa a se tornar popular durante a década de 1990, quando se começa a oferecer *bandwidths* de conexão maiores. Em 1999, por exemplo, é fundada a empresa Salesforce.com [4], pelo antigo executivo da Oracle, Marc Benioff, junto com três desenvolvedores de software, Parker Harris, Dave Moellenhoff, e Frank Dominguez, que juntos lançam o primeiro de software de automação de vendas. Com esta plataforma, são entregues serviços empresariais através de um simples *website*, pavimentando o caminho para inúmeros outros serviços e aplicações providos pela internet.

Em 2002, a plataforma *Amazon Web Services* é lançada, "expondo tecnologia e dados de produtos da Amazon e de seus afiliados, permitindo que desenvolvedores criem aplicativos inovadores e empresariais por conta própria" [5]. Inicialmente, oferecia apenas algumas ferramentas e serviços, sendo publicamente reformulado em 2003, quando Chris Pinkham e Benjamin Black apresentaram um artigo descrevendo uma visão da infra-estrutura de

*Instituto de Computação, Universidade Estadual de Campinas (UNICAMP), Campinas, SP

computação da Amazon, completamente padronizada, completamente automatizada e que dependeria amplamente de *webservices*, como armazenamento [6]. Além disso, mencionavam a possibilidade de vender o acesso a servidores virtuais como um serviço, gerando uma nova fonte de recursos com o investimento nessa infra-estrutura. Em 2004, finalmente, é lançado para o público em geral a Amazon Simple Queue Service (SQS), um *hosted queue* altamente escalável para armazenamento de mensagens entre componentes de aplicativos distribuídos [7], e em 2006 a Amazon AWS é oficialmente reinaugurada, ganhando os serviços EC2 [8], S3 [9] e o já discutido SQS.

Seguindo o mesmo caminho, a Microsoft lança seu próprio serviço de nuvem, com o Windows Azure, posteriormente renomeado para Microsoft Azure, enquanto a Google lança o Google Apps, oferecendo serviços de email (Gmail), Google Docs, Sheets, Slides e até mesmo um serviço de armazenamento chamado Google Drive [10].

Hoje, com a mais recente escalada do uso de celulares smartphones, notadamente a partir do lançamento do primeiro iPhone [11], serviços providos através da internet tornam-se cada vez mais imprescindíveis, movimentando centenas de bilhões de dólares todos os anos.

Neste documento, estudamos conceitos de computação em nuvens, apresentando na seção dois os serviços mais comuns e conhecidos, enquanto na seção três comentamos sobre modelos de implementação, que variam conforme as necessidades do cliente. Ainda, na seção quatro, comentamos um conhecido problema, que é o foco deste trabalho, o lock-in. Na mesma seção, também apresentamos um framework disponível num repositório público, e na seção cinco buscamos estudar o OpenStack, software de código aberto para criação de nuvens públicas e privadas.

2 Tipologia

Desde que foi primeiramente trabalhada, a computação em nuvens vem evoluindo e cada vez mais serviços são oferecidos, cada um provendo soluções para diferentes tipos de problemas. Podemos, então, dividir esses serviços em algumas categorias:

- **IaaS** Infrastructure as a Service ou Infraestrutura como Serviço é a categoria mais básica entre todos os serviços, oferecendo uma infraestrutura de computação instantânea, provisionada e gerenciada através da internet. É possível escalar rapidamente, de acordo com as necessidades instantâneas do cliente.

- **PaaS** Platform as a Service ou Plataforma como Serviço é um ambiente sob demanda para desenvolvimento, teste, fornecimento e gerenciamento de aplicativos de software. Com ela, é possível criar rapidamente aplicativos móveis ou web, sem precisar se preocupar com a configuração da infraestrutura de servidores, armazenamento, rede e bancos de dados necessários.

- **SaaS** Software as a Service ou Software como Serviço permite aos usuários se conectar e usar aplicativos baseados em nuvem. Exemplos são o email, calendário e ferramentas de edição, como o Office e o Google Docs.

- **DaaS [12]** Data as a Service ou Dados como Serviço é um modelo baseado em SaaS, porém especializado em fornecer e distribuir informações no qual os arquivos de dados (texto, imagens, sons, vídeos) estão disponíveis para clientes em uma rede, geralmente a internet.

- **CaaS [13]** Communication as a Service ou Comunicação como Serviço é um serviço que traz as redes sociais, a computação em nuvem e os smartphones em conjunto, fornecendo tecnologias que permitem aos usuários se comunicarem via voz, texto e multimídia em qualquer dispositivo que desejarem utilizar.

- **DBaaS [14]** Data Base as a Service ou Banco de dados como Serviço, é um modelo baseado no PaaS, porém especializado em prover um serviço de banco de dados através de um conjunto comum de abstrações. Assim, pode-se por exemplo utilizar um MySQL ou um MongoDB, através da mesma interface, utilizando exatamente as mesmas chamadas de API, sem que o usuário tenha que se preocupar com a exata implementação do banco de dados.

- **SECaaS [15]** Security as a Service ou Segurança como Serviço, é um modelo baseado no SaaS, porém especializado em serviços de segurança de informações, beneficiando organizações através do custo reduzido, facilidade de gerenciamento e atualizações contínuas de anti-virus.

- **XaaS [16]** Anything as a Service ou Tudo como Serviço. É um serviço que oferece em um mesmo local outros serviços discutidos anteriormente nesta seção.

3 Modelos de Implementação

As nuvens podem se diferenciar de acordo com o processo de negócios, do tipo de informação e do nível de visão desejado. Por isso, o acesso de usuários pode ser restringido de acordo com as necessidades.

- **Pública** Uma nuvem pública é quando os serviços são apresentados ao público através de uma rede aberta, e os usuários de diferentes aplicações acessam as mesmas máquinas. Em outras palavras, isso significa que recursos computacionais são compartilhados entre várias contas. Graças a isso, os usuários compartilham os custos de manutenção, o que reduz drasticamente o investimento necessário. Entretanto, são necessárias políticas rigorosas de segurança, para que vulnerabilidades de uma conta não afete usuários de outros serviços.

- **Privada** Uma nuvem privada é construída com o intuito de atender um único usuário (normalmente empresas), e a infraestrutura pertence totalmente a ele (um datacenter, por exemplo), sendo normalmente acessada através de uma rede privada. Por isso, o gerente de T.I. possui total controle sobre como as aplicações são implementadas, porém isso acaba por demandar um maior investimento.

- **Híbrida** As nuvens híbridas são compostas por nuvens públicas e privadas, tornando-se uma tendência graças ao seu alto nível de flexibilidade. Pode ser adaptada facilmente a vários ambientes, e normalmente são utilizadas por empresas que possuem alguma restrição em levar todos seus serviços para a nuvem pública, como dificuldades de conectividade local à internet. Outra vantagem é a possibilidade de ter seus recursos ampliados a partir de uma reserva em uma nuvem pública, o que caracteriza a *computação em ondas* [17].

4 Cloud Lock-In

Apesar da melhora de interoperabilidade entre plataformas, APIs para computação em nuvem ainda são essencialmente proprietários, não possuindo uma padronização. Por isso, usuários não conseguem migrar dados e aplicações de um provedor para outro facilmente, o que os deixa muito vulneráveis para aumento de preços, problemas de confiabilidade ou até mesmo se provedores forem à falência [18].

4.1 TwentyTwenty.Storage

Um dos problemas de vendor lock-in mais conhecido é o armazenamento. Como exemplo, a Microsoft Azure oferece um serviço chamado BlobStorage, a Amazon AWS oferece o S3, enquanto o Google oferece o APIs Storage. Uma das soluções mais discutidas para se evitar esse problema é a existência de uma interface única para utilizar diferentes provedores. Para isso, a TwentyTwenty oferece um framework público [19] desenvolvido em .NET, em que é possível acessar qualquer um dos três serviços citados anteriormente, mais a possibilidade de se utilizar um armazenamento local, apenas indicando o caminho do diretório a ser utilizado.

Porém, este projeto, no momento do presente estudo, encontrava-se desatualizado, com diversos *packages* antigos, alguns inclusive em fase beta. Este problema é contornado com a atualização para as versões mais recentes disponíveis através do gerenciador de pacotes NuGet.

Para se utilizar o framework, basta criar um projeto e instanciar o respectivo provedor, com suas respectivas configurações iniciais:

```
IStorageProvider provider = new AmazonStorageProvider(new
    AmazonProviderOptions
{
    Bucket = "mybucketname",
    PublicKey = "mypublickey",
    SecretKey = "mysecretkey"
});
```

Com isso, é possível utilizar as funções genéricas salvar, atualizar, deletar *blobs*, entre diversas outras funções.

```
await _provider.SaveBlobStreamAsync(containerName, blobName, dataStream);
await _provider.GetBlobDescriptorAsync(containerName, blobName);
await _provider.DeleteBlobAsync(containerName, blobName);
```

Neste trabalho, buscamos iniciar a implementação de um acesso a mais uma nuvem bem conhecida, o OpenStack.

5 OpenStack

O OpenStack é um software de código aberto para criar nuvens privadas e públicas, capaz de controlar conjuntos de recursos computacionais, armazenamento e rede de um datacenter, gerenciados através de um painel ou através de uma API [20]. É considerado como uma plataforma, por oferecer APIs que, em conjunto, são capazes de controlar todos os recursos disponíveis: máquinas virtuais, rede, armazenadores, balanceadores de carga, entre inúmeros outros.

5.1 Configuração de Ambiente

O próprio guia de instalação recomenda instalar o software num servidor ou máquina virtual dedicada ao seu propósito, pois a instalação faz mudanças substanciais no sistema. Aqui, optamos por instalar um Ubuntu Server v16.04 numa máquina virtual utilizando a Oracle VirtualBox v5.2.2 para MacOS. Assim que tivemos nosso sistema instalado, configuramos a máquina virtual para utilizar um adaptador de rede conectado a um *Bridged Adapter*, de modo que podemos acessar o sistema via SSH.

Além disso, como o OpenStack fica associado a um IP, decidimos configurar o sistema para que utilizasse um IP estático, além de servidores DNS da Google:

```
# The primary network interface
auto enp0s3
iface enp0s3 inet static
    address 192.168.15.100
    netmask 255.255.255.0
    gateway 192.168.15.1
    dns-nameservers 8.8.8.8 8.8.4.4
```

Para que o servidor não entre em suspensão, desativamos essas opções através do comando:

```
$ sudo systemctl mask sleep.target suspend.target hibernate.target
   hybrid-sleep.target
```

Também optamos por atualizar todos os pacotes possíveis do ubuntu antes da instalação do openstack, através dos seguintes comandos:

```
$ sudo apt-get update
$ sudo apt-get upgrate
$ sudo apt-get dist-upgrate
```

Agora, com o sistema devidamente configurado e atualizado, iniciamos a instalação do OpenStack.

5.2 Instalação

Para a instalação do OpenStack, seguimos o tutorial na documentação oficial [21].

Adicionamos o usuário a lista de sudoers, e baixamos do repositório oficial a última versão.

```
$ echo "ubuntu ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/ubuntu
$ sudo su - ubuntu
$ git clone https://git.openstack.org/openstack-dev/devstack
$ cd devstack
```

Antes de iniciar a instalação, precisamos criar um arquivo local.conf. Para utilizar os serviços de armazenamento, utilizamos um exemplo disponível na internet [22], com algumas adaptações para facilitar a configuração de senhas e a ativação do serviço Swift, discutido posteriormente.

```
# Sample ‘‘local.conf‘‘ for user-configurable variables in ‘‘stack.sh‘‘

# NOTE: Copy this file to the root DevStack directory for it to work
    properly.

# ‘‘local.conf‘‘ is a user-maintained settings file that is sourced from
    ‘‘stackrc‘‘.
# This gives it the ability to override any variables set in ‘‘stackrc‘‘.
# Also, most of the settings in ‘‘stack.sh‘‘ are written to only be set
    if no
# value has already been set; this lets ‘‘local.conf‘‘ effectively
    override the
# default values.

# This is a collection of some of the settings we have found to be useful
# in our DevStack development environments. Additional settings are
    described
# in http://devstack.org/local.conf.html
# These should be considered as samples and are unsupported DevStack
    code.

# The ‘‘localrc‘‘ section replaces the old ‘‘localrc‘‘ configuration
    file.
# Note that if ‘‘localrc‘‘ is present it will be used in favor of this
    section.
[[local|localrc]]

# Minimal Contents
# ----------------
```

```
# While ‘‘stack.sh‘‘ is happy to run without ‘‘localrc‘‘, devlife is
    better when
# there are a few minimal variables set:

# If the ‘‘SERVICE_TOKEN‘‘ and ‘‘*_PASSWORD‘‘ variables are not set
# here you will be prompted to enter values for them by ‘‘stack.sh‘‘
# and they will be added to ‘‘local.conf‘‘.
SERVICE_TOKEN=azertytoken
ADMIN_PASSWORD=password
MYSQL_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
disable_service n-net
enable_service q-svc
enable_service q-agt
enable_service q-dhcp
enable_service q-l3
enable_service q-meta
enable_service neutron
enable_service swift
RECLONE=YES

# ‘‘HOST_IP‘‘ and ‘‘HOST_IPV6‘‘ should be set manually for best results
    if
# the NIC configuration of the host is unusual, i.e. ‘‘eth1‘‘ has the
    default
# route but ‘‘eth0‘‘ is the public interface.  They are auto-detected in
# ‘‘stack.sh‘‘ but often is indeterminate on later runs due to the IP
    moving
# from an Ethernet interface to a bridge on the host. Setting it here
    also
# makes it available for ‘‘openrc‘‘ to include when setting
    ‘‘OS_AUTH_URL‘‘.
# Neither is set by default.
HOST_IP=192.168.15.100
#HOST_IPV6=2001:db8::7


# Logging
# -------

# By default ‘‘stack.sh‘‘ output only goes to the terminal where it
    runs.  It can
# be configured to additionally log to a file by setting ‘‘LOGFILE‘‘ to
```

```
    the full
# path of the destination log file.  A timestamp will be appended to the
    given name.
LOGFILE=$DEST/logs/stack.sh.log
#HOST_IPV6=2001:db8::7


# Logging
# -------

# By default ‘‘stack.sh‘‘ output only goes to the terminal where it
    runs.  It can
# be configured to additionally log to a file by setting ‘‘LOGFILE‘‘ to
    the full
# path of the destination log file.  A timestamp will be appended to the
    given name.
LOGFILE=$DEST/logs/stack.sh.log

# Old log files are automatically removed after 7 days to keep things
    neat.  Change
# the number of days by setting ‘‘LOGDAYS‘‘.
LOGDAYS=2

# Nova logs will be colorized if ‘‘SYSLOG‘‘ is not set; turn this off by
    setting
# ‘‘LOG_COLOR‘‘ false.
#LOG_COLOR=False


# Using milestone-proposed branches
# ---------------------------------

# Uncomment these to grab the milestone-proposed branches from the repos:
#CINDER_BRANCH=milestone-proposed
#GLANCE_BRANCH=milestone-proposed
#HORIZON_BRANCH=milestone-proposed
#KEYSTONE_BRANCH=milestone-proposed
#KEYSTONECLIENT_BRANCH=milestone-proposed
#NOVA_BRANCH=milestone-proposed
#NOVACLIENT_BRANCH=milestone-proposed
#NEUTRON_BRANCH=milestone-proposed
#SWIFT_BRANCH=milestone-proposed
```

```
# Swift
# -----

# Swift is now used as the back-end for the S3-like object store. If
   Nova's
# objectstore (''n-obj'' in ''ENABLED_SERVICES'') is enabled, it will NOT
# run if Swift is enabled. Setting the hash value is required and you
   will
# be prompted for it if Swift is enabled so just set it to something
   already:
SWIFT_HASH=66a3d6b56c1f479c8b4e70ab5c2000f5

# For development purposes the default of 3 replicas is usually not
   required.
# Set this to 1 to save some resources:
SWIFT_REPLICAS=1

# The data for Swift is stored by default in (''$DEST/data/swift''),
# or (''$DATA_DIR/swift'') if ''DATA_DIR'' has been set, and can be
# moved by setting ''SWIFT_DATA_DIR''. The directory will be created
# if it does not exist.
SWIFT_DATA_DIR=$DEST/data

ENABLED_SERVICES+=,key,swift,mysql

# Tempest
# -------

# Install the tempest test suite
enable_service tempest

# Cinder - Block Device Service
# ENABLED_SERVICES+=,cinder,c-api,c-vol,c-sch,c-bak
```

Finalmente, basta utilizar o comando

```
$ ./stack.sh
```

E após cerca de quarenta minutos a instalação é completa.

5.3 Serviços

Com o software instalado, temos acesso a um dashboard disponível no ip configurado. Basta ir ao navegador e acessar 192.168.15.100, que é o IP estático configurado anteriormente. Podemos acessar o sistema utilizando o usuário *admin* e senha *password*.

Como administrador, temos acesso a outras funcionalidades do sistema, como por exemplo a administração de usuários e de projetos.

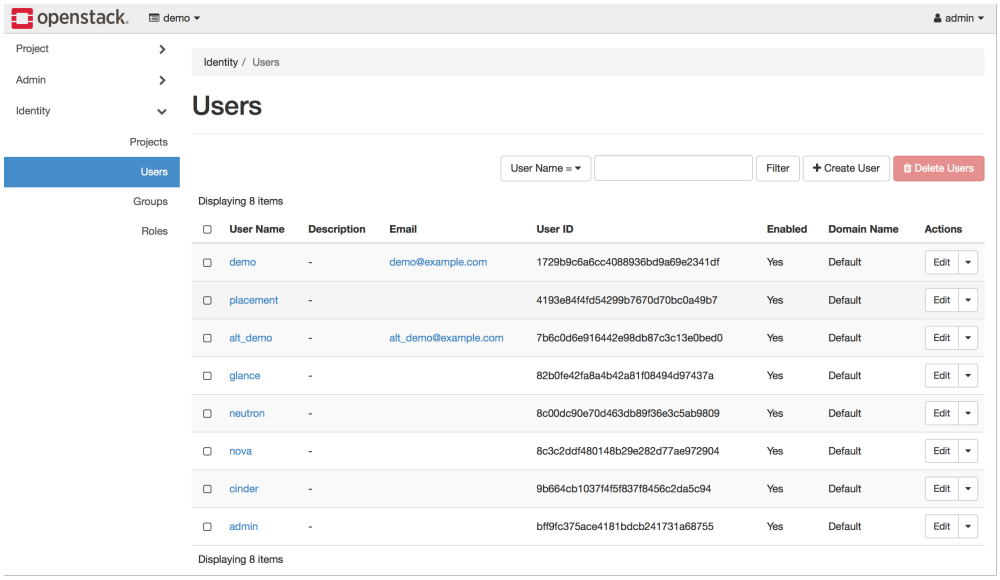

Figura 1: Administração de usuários.

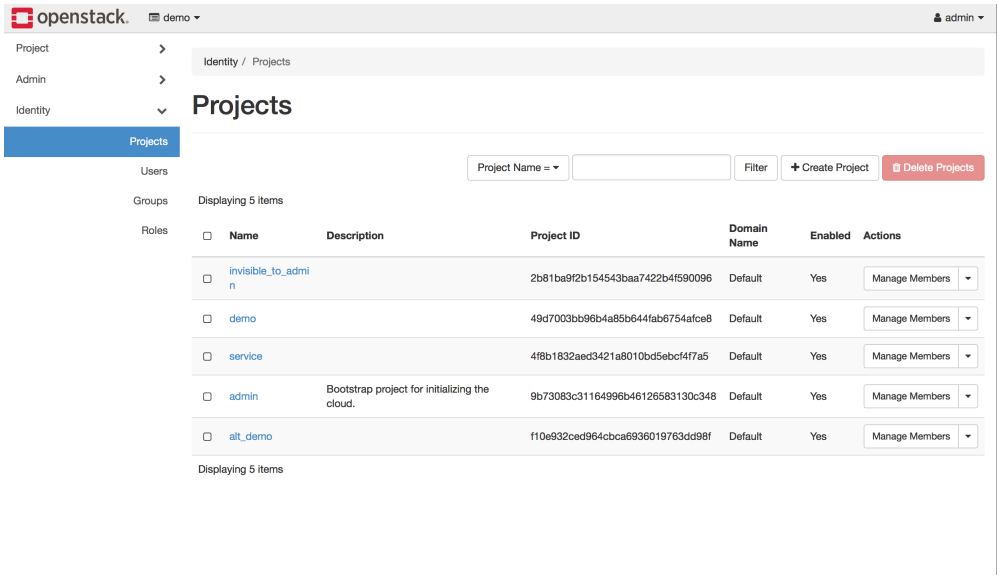

Figura 2: Administração de projetos.

5.4 Keystone

Para podermos utilizar a plataforma através da API, é necessário instalar o keystone [23], um serviço para autenticação de cliente de API e descoberta de serviços, utilizando o seguinte comando:

```
$ sudo apt-get install keystone
```

Após a instalação, podemos acessar a API no endereço 192.168.15.100 na porta 5000. Entretanto, após a instalação, não é mais possível utilizar o dashboard apresentado anteriormente.

5.5 openstack.net

Existe um framework não oficial chamado openstack.net, em que podemos através dele acessar recursos de uma plataforma openstack. Entretanto, ele utiliza uma versão anterior do serviço *Identity* [24], responsável por gerar tokens de autenticação que permite chamadas aos outros serviços da plataforma através do uso de APIs. Já existe uma branch no projeto que busca atualizar esse serviço, porém ainda não foi oficialmente lançado. Por isso, neste trabalho optamos por estudar os *endpoints* disponíveis, antes de implementar uma versão que utilize o framework citado.

5.6 Estudos de API

Através da API disponível no ip 192.168.15.100 configurado previamente, é possível realizar chamadas que manipulam os serviços desejados. Utilizando o Postman [25], uma cadeia de ferramentas que torna o trabalho com as APIs mais rápido e intuitivo através de requisições HTTP, buscamos estudar quais são os parâmetros necessários para cada requisição.

5.6.1 Autenticação

Para autenticar o acesso aos serviços OpenStack, devemos emitir um pedido de autenticação para o *Identity*, enviando um payload contendo as credenciais. As credenciais são geralmente uma combinação do nome de usuário, senha, e opcionalmente o nome ou ID do projeto. Também é possível obter um token, ao invés de um usuário e senha.

Como exemplo de requisição, observe a seguinte chamada, para o endpoint /v3/auth/tokens:

```
{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "domain": {
            "name": "Default"
```

```
          },
          "name": "admin",
          "password": "password"
        }
      }
    },
    "scope": {
      "project": {
        "domain": {
          "name": "Default"
        },
        "name": "demo"
      }
    }
  }
}
```

Assim, com credenciais válidas, obtemos no *X-Subject-Token* header o token de acesso, enquanto no *body* obtemos os detalhes do token (note que o token é gerado a cada requisição):

```
X-Subject-Token:
    gAAAAABaOA_9iE2EvXw9ARFOefOe6cxYt-S1eH9o6aPhG717Dr3jmzdWiY-_
UjRqgup-8CDh66Q5RIjWWVfzzfHbH4ErrHWEI_KoPXFfeWuu5fdeNM12AkEWvI1QZounhSl3b9xW65D09K6G
--A9KLsFnfOJZ0-vrQcdrHIdC1XnzpsJY6Ag984
```

```
{
    "token": {
        "is_domain": false,
        "methods": [
            "password"
        ],
        "roles": [
            {
                "id": "548bf4cf6bf14093a20eba6b5df366be",
                "name": "admin"
            }
        ],
        "expires_at": "2017-12-18T18:39:59.000000Z",
        "project": {
            "domain": {
                "id": "default",
                "name": "Default"
            },
            "id": "518448b3261f42138a44b31b68acf170",
            "name": "demo"
```

```
},
"catalog": [
    {
        "endpoints": [
            {
                "url": "http://172.20.10.12/placement",
                "interface": "public",
                "region": "RegionOne",
                "region_id": "RegionOne",
                "id": "b343ce67b6b54861a2fc1e28864e82cc"
            }
        ],
        "type": "placement",
        "id": "1569b3c150c140cfbae70b1b3fc0e499",
        "name": "placement"
    },
    {
        "endpoints": [
            {
                "url": "http://172.20.10.12/volume/v1/
                                518448b3261f42138a44b31b68acf170",
                "interface": "public",
                "region": "RegionOne",
                "region_id": "RegionOne",
                "id": "703780ab0c30493f892b939c4f3e3dab"
            }
        ],
        "type": "volume",
        "id": "2e4e3100ef504563b7097c03b689301d",
        "name": "cinder"
    },
    {
        "endpoints": [
            {
                "url": "http://172.20.10.12/volume/v2/
                                518448b3261f42138a44b31b68acf170",
                "interface": "public",
                "region": "RegionOne",
                "region_id": "RegionOne",
                "id": "ff871eef406945c38f7376192303d35e"
            }
        ],
        "type": "volumev2",
        "id": "57d855d4483f4093beab48143d39a1cc",
```

```
            "name": "cinderv2"
        },
        {
            "endpoints": [
                {
                    "url": "http://172.20.10.12:9696/",
                    "interface": "public",
                    "region": "RegionOne",
                    "region_id": "RegionOne",
                    "id": "af8c4f7a44bb4fe2bd2b9c67222ed5ef"
                }
            ],
            "type": "network",
            "id": "9f60f0e2e70a401abebe4becdd3ddb13",
            "name": "neutron"
        },
        {
            "endpoints": [
                {
                    "url": "http://172.20.10.12/compute/v2.1",
                    "interface": "public",
                    "region": "RegionOne",
                    "region_id": "RegionOne",
                    "id": "d5db9743272f469cbc4f14449c1b6b69"
                }
            ],
            "type": "compute",
            "id": "ab5be75f286140c9babeb5c3532caa25",
            "name": "nova"
        },
        {
            "endpoints": [
                {
                    "url": "http://172.20.10.12/image",
                    "interface": "public",
                    "region": "RegionOne",
                    "region_id": "RegionOne",
                    "id": "4eb37c9a48ed46ad9b8c6c908609f8c0"
                }
            ],
            "type": "image",
            "id": "affb2c6624c74e1c84f30a38b9c5a636",
            "name": "glance"
        },
```

```
{
    "endpoints": [
        {
            "url": "http://172.20.10.12/compute/v2/
                                518448b3261f42138a44b31b68acf170",
            "interface": "public",
            "region": "RegionOne",
            "region_id": "RegionOne",
            "id": "1dcb7b75737d42d987cf163f1b106588"
        }
    ],
    "type": "compute_legacy",
    "id": "d6a36ad2500840c1a095cb89ffbd3d54",
    "name": "nova_legacy"
},
{
    "endpoints": [
        {
            "url": "http://172.20.10.12:8080/v1/
                                AUTH_518448b3261f42138a44b31b68acf170",
            "interface": "public",
            "region": "RegionOne",
            "region_id": "RegionOne",
            "id": "2d9826b3a24b4a949d1222db3fcd6e7f"
        },
        {
            "url": "http://172.20.10.12:8080",
            "interface": "admin",
            "region": "RegionOne",
            "region_id": "RegionOne",
            "id": "5e4e6f7c97774fe7b3f6b2fe14aa992b"
        }
    ],
    "type": "object-store",
    "id": "de099264e3ec43808142e8cd7cde2122",
    "name": "swift"
},
{
    "endpoints": [
        {
            "url": "http://172.20.10.12/identity",
            "interface": "public",
            "region": "RegionOne",
            "region_id": "RegionOne",
```

```
                        "id": "daf0fff9de7d44cd90fc8a4352001ee6"
                    },
                    {
                        "url": "http://172.20.10.12/identity",
                        "interface": "admin",
                        "region": "RegionOne",
                        "region_id": "RegionOne",
                        "id": "dde850e6e0b04015839036e8e63e5e3c"
                    }
                ],
                "type": "identity",
                "id": "e143a7126fa94a40ad35a86adf651d5d",
                "name": "keystone"
            },
            {
                "endpoints": [
                    {
                        "url": "http://172.20.10.12/volume/v3/
                                        518448b3261f42138a44b31b68acf170",
                        "interface": "public",
                        "region": "RegionOne",
                        "region_id": "RegionOne",
                        "id": "313db6e9958f43fda45ab7a7f5ea7d90"
                    }
                ],
                "type": "volumev3",
                "id": "ebdf06f79afb4c199409117901d77632",
                "name": "cinderv3"
            }
        ],
        "user": {
            "password_expires_at": null,
            "domain": {
                "id": "default",
                "name": "Default"
            },
            "id": "c5f8fe775984474996617a74428f4110",
            "name": "admin"
        },
        "audit_ids": [
            "9Llj6onPQa6jR4qk0902hA"
        ],
        "issued_at": "2017-12-18T17:39:59.000000Z"
    }
```

```
}
```

Em seguida, quando enviarmos requisições para a API, devemos incluir o token recebido no *X-Auth-Token* header. Se utilizarmos diferentes serviços, devemos obter um token para cada. Um token é válido por um tempo limitado, após a expiração, é necessário autenticar-se novamente.

5.6.2 Swift

O serviço responsável por armazenamento é o Swift [26]. Ele é um textitobject/blob store altamente disponível, distribuído e eventualmente consistente. Assim, organizações podem utiliza-lo para armazenar dados de forma eficiente, segura e econômica, podendo criar, modificar e acessar objetos e metadados através de uma REST API. Através dessa interface, é possível fazer requisições HTTP para realizar operações, e com o protocolo SSL podemos interagir com os objetos.

Da documentação [27], a API de Armazenamento de Objetos organiza dados em forma de hierarquia:

- **Conta** Representa o nível superior da hierarquia. O provedor de serviços fornece uma conta para o usuário, e assim ele possui todos os recursos disponíveis nessa conta. A conta define um namespace para contêineres, e um contêiner pode ter o mesmo nome em duas contas diferentes. No ambiente OpenStack, a conta é equivalente ao projeto.
- **Container** Define um *namespace* para objetos. Um objeto com mesmo nome em dois contêineres diferentes representam objetos diferentes. Também é possível criar diversos contêineres em uma mesma conta.
- **Objeto** Armazena os dados, como documentos, imagens, vídeos, enfim. Também é possível armazenar metadados com um objeto.

Para acessar a API, podemos analisar a documentação [28] e verificar quais são os endpoints específicos para cada operação. Para acessar os dados descritos anteriormente, basta fazer requisições HTTP respectivamente para os seguintes endpoints:

```
/v1/{account}
/v1/{account}/{container}
/v1/{account}/{container}/{object}
```

É possível realizar requisições GET para acessar os dados e POST para atualizar metadados, por exemplo. É preciso trocar *account* pelo ID fornecido pelo provedor, *conteiner* pelo ID do contêiner e *object* pelo ID do objeto.

6 Conclusão

Com esse trabalho, estudamos as possíveis soluções para lock-in, e constatamos diversos estudos e soluções nesse sentido. O TwentyTwenty.Storage, implementado em .NET para

acessar a API da AWS S3, Google APIs e Azure Blob Storage, é uma das soluções encontradas. Apesar do projeto estar desatualizado, é possível modificá-lo para utilizar os frameworks mais recentes. Desejamos, então, incluir nesse projeto, o acesso a mais um provedor, o OpenStack, porém foram encontrados diversos desafios, já que não encontramos framework oficial e/ou projetos minimamente atualizados com as versões mais recentes do OpenStack. Buscando contornar esse problema, encontramos a possibilidade de realizar essa interação através de chamadas diretas a API, embora isso demandasse maior conhecimento em outras áreas do que se houvesse simplesmente um framework para essa função.

Referências

[1] ENIAC. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2017. Disponível em https://pt.wikipedia.org/wiki/ENIAC. Acesso em 16 dez. 2017.

[2] J.C.R Licklider. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2017. Disponível em https://en.wikipedia.org/wiki/J._C._R._Licklider. Acesso em 16 dez. 2017.

[3] A history of cloud computing. ComputerWeekly, 2017. Disponível em http://www.computerweekly.com/feature/A-history-of-cloud-computing. Acesso em 16 dez. 2017.

[4] Salesforce.com. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2017. Disponível em https://en.wikipedia.org/wiki/Salesforce.com. Acesso em 16 dez. 2017.

[5] Amazon Web Services Launches. In: Amazon Press Release. Amazon, 2017. Disponível em http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-newsArticle&ID=830816. Acesso em 16 dez. 2017.

[6] Amazon Web Services. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2017. Disponível em https://en.wikipedia.org/wiki/Amazon_Web_Services. Acesso em 16 dez. 2017.

[7] Introducing the Amazon Simple Queue Service. AWS, 2004. Disponível em https://aws.amazon.com/about-aws/whats-new/2004/11/03/introducing-the-amazon-simple-queue-service. Acesso em 16 dez. 2017.

[8] Amazon Elastic Compute Cloud. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2017. Disponível em https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud. Acesso em 16 dez. 2017.

[9] Amazon S3. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2017. Disponível em https://en.wikipedia.org/wiki/Amazon_S3. Acesso em 16 dez. 2017.

[10] G Suite. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2017. Disponível em https://en.wikipedia.org/wiki/G_Suite. Acesso em 16 dez. 2017.

[11] Timeline of Apple "iPhone" Rumors (1999-Present). In: FierceWireless. Questex LLC, 2017. Disponível em https://www.fiercewireless.com/wireless/timeline-apple-iphone-rumors-1999-present. Acesso em 16 dez. 2017.

[12] data as a service. TechTarget, 2017. Disponível em http://searchcloudapplications.techtarget.com/definition/data-as-a-service. Acesso em 16 dez. 2017.

[13] CaaS: Communications-as-a-Service. In: Dr. Dobb's. UBM, 2017. Disponível em http://www.drdobbs.com/web-development/caas-communications-as-a-service/217600340. Acesso em 16 dez. 2017.

[14] DAYS, Frank. What is Database as a Service?. In: Stratoscale. Stratoscale, 2017. Disponível em https://www.stratoscale.com/blog/dbaas/what-is-database-as-a-service/. Acesso em 16 dez. 2017.

[15] HUSSAIN, Mohammed; ABDULSALAM, Hanady. SECaaS: Security as a Service for Cloud-based Applications. In: ResearchGate. ResearchGate GmbH, 2017. Disponível em https://www.researchgate.net/publication/254004357. Acesso em 16 dez. 2017.

[16] X as a service (XaaS): What the future of cloud computing will bring. In: CloudTech. Cloud Tech News, 2017. Disponível em https://www.cloudcomputing-news.net/news/2014/aug/18/x-as-a-service-xaas-what-the-future-of-cloud-computing-will-bring/. Acesso em 16 dez. 2017.

[17] O que é cloud bursting?. Microsoft, 2017. Disponível em https://azure.microsoft.com/pt-br/overview/what-is-cloud-bursting/. Acesso em 18 dez. 2017.

[18] ARMBRUST, Michael; FOX, Armando; GRIFFITH, Rean; JOSEPH, Anthony D.; KATZ, Randy; KONWINSKI, Andy; LEE, Gunho; PATTERSON, David; RABKIN, Ariel; STOICA, Ion; ZAHARIA, Matei. A View of Cloud Computing. In: CloudTech. Cloud Tech News, 2017. Disponível em https://cacm.acm.org/magazines/2010/4/81493-a-view-of-cloud-computing/fulltext. Acesso em 16 dez. 2017.

[19] TwentyTwenty.Storage. Disponível em https://github.com/2020IP/TwentyTwenty.Storage. Acesso em 17 dez. 2017.

[20] OpenStack. Openstack.org, 2017. Disponível em https://www.openstack.org. Acesso em 17 dez. 2017.

[21] DevStack. Openstack.org, 2017. Disponível em https://docs.openstack.org/devstack/latest/. Acesso em 17 dez. 2017.

[22] https://github.com/poornimakshirsagar/devstack-cinder-backup/blob/master/local.conf. Acesso em 17 dez. 2017.

[23] Keystone. Openstack.org, 2017. Disponível em https://docs.openstack.org/keystone/. Acesso em 17 dez. 2017.

[24] Identity. Openstack.org, 2017. Disponível em https://docs.openstack.org/api-ref/identity/v3/index.html. Acesso em 17 dez. 2017.

[25] Postman. Postdot Technologies, Inc., 2017. Disponível em https://www.getpostman.com/. Acesso em 17 dez. 2017.

[26] Swift. Openstack.org, 2017. Disponível em https://docs.openstack.org/swift/latest/. Acesso em 18 dez. 2017.

[27] Object Storage API overview. Openstack.org, 2017. Disponível em https://docs.openstack.org/swift/latest/api/object_api_v1_overview.html. Acesso em 18 dez. 2017.

[28] Object Storage API. Openstack.org, 2017. Disponível em https://developer.openstack.org/api-ref/object-store/index.html. Acesso em 18 dez. 2017.