FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO


# Framework de gestão de notícias regionais em canais de televisão

**Diogo Santos Tavares**

MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Orientador: Maria Teresa de Andrade

Orientador externo: Pedro Santos

20 de Fevereiro de 2019

# Resumo

Nos últimos anos tem-se verificado uma tendência de hiperlocalização de notícias, por parte dos canais televisivos de caráter nacional, que visa uma aproximação mais rápida e eficaz do espetador às notícias que lhe são relevantes. Estes canais usam uma rede noticiosa que tem por base órgãos de comunicação regional, sendo que usam muitas vezes os conteúdos audiovisuais, recolhidos pelos jornalistas pertencentes a estes órgãos, nas suas transmissões. No entanto, estes profissionais encontram-se fora da cadeia de contribuições destes canais, surgindo assim uma debilidade no que toca à forma de comunicação entre estas identidades e na contribuição dos conteúdos.

A evolução ao nível do desempenho do hardware informático veio possibilitar a virtualização dos ambientes de produção de conteúdo televisivo e, ao usar as novas tecnologias abertas baseadas em nuvem permite a interligação de todos os jornalistas espalhados pelo território nacional, possibilitando a criação de uma rede de contribuições mais próximas dos locais das notícias.

Esta dissertação procura desenvolver uma solução de abstração e virtualização dos recursos de uma plataforma, em ambiente virtual, capaz de oferecer suporte para as fases de contribuição, produção, gestão e distribuição de conteúdo. Vai assim permitir a interligação dos vários jornalistas localizados em diferentes pontos do país e torna-se possível a redução de custos e a carga logística deste canais televisivos, para além de usar os recursos disponíveis de uma forma mais intuitiva.

O objetivo passa também por descentralizar o poder computacional da nuvem e executar parte do processamento dos dados mais próximo da fonte destes, sendo possível otimizar aspetos como a latência, custos de largura de banda ao enviar para a nuvem e ainda a privacidade dos dados. Trata-se de uma tendência registada neste últimos anos visto que, face ao grande volume de dados que hoje em dia são produzidos e transportados para a nuvem, as tecnologias e soluções utilizadas tornam-se ineficientes.

# Abstract

In recent years, it has been a news hyperlocalization trend by national news TV channels aiming a more rapid and effective approach by the viewer to the news that is relevant to him. These channels use a news network that is based on regional media, using audiovisual content collected by journalists belonging to these organs in their broadcasts. However, these professionals are outside the chain of these channels, resulting in a weakness in the form of communication between these identities and in the contribution of content.

The computer hardware performance evolution has made possible the television content production environments virtualization and the use of new open-source and cloud based technologies allows the interconnection of all the journalists spread across the national territory, making it possible to create a network of contributions that are closer to the news locations.

Thus, this paper porposes to develop a resources virtualization and abstraction solution, in order to support a platform capable of integrating the content contribution, production, management and distribution phases, allowing the interconnection of the various journalists located in different parts of the country. In this way, it becomes possible to reduce costs and logistic load of this television system, in addition to using available resources in a more intuitive way.

The goal is also to decentralize the computing power of the cloud and to perform some of the data processing closer to the source, which can optimize aspects such as latency, bandwidth costs when sending to the cloud and data privacy. This is a trend seen in recent years, given the large amount of data that is now being produced and transported to the cloud, the technologies and solutions used become inefficient.

# Agradecimentos

Gostaria de agradecer, em primeiro lugar, ao meus pais e ao meu irmão, pelo o apoio incondicional ao longo do meu percurso académico e também pelo esforço colocado para me proporcionarem as condições necessárias à minha formação. Aos meus amigos de sempre, que me acompanharam em cada etapa da minha vida.

À minha orientadora, Professora Maria Teresa de Andrade, por ter aceite este desafio. Estou grato pelo contributo a nível técnico e académico, e pela paciente revisão dos conteúdos desta dissertação.

A toda a instituição FEUP, pelo conhecimento que me foi transmitido, por tudo aquilo que representa e por todas as condições disponibilizadas ao longo do meu ciclo de estudos.

Ao Engenheiro Pedro Santos, pela orientação, disponibilidade e por todo o conhecimento transmitido, importante para a realização desta dissertação. Ao Engenheiro Alexandre Ulisses, pela proposta deste desafio.

Finalmente, aos meus colegas de estágio na MOG Technologies, pela camaradagem que demonstraram neste período, assim como a todos os colaboradores pela simpatia que sempre tiveram para comigo.

Diogo Santos Tavares

*"Many of life's failures are people who did not realize how close they were to success when they gave up"*

Thomas A. Edison

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Abreviaturas e Símbolos

| | |
|---|---|
| API | Application Programming Interface |
| AWS | Amazon Web Services |
| CDN | Content Delivery Network |
| CPU | Central Process Unit |
| DNS | Domain Name Service |
| GUI | Graphical User Interface |
| HTTP | HyperText Transfer Protocol |
| IaaS | Infrastructure as a Service |
| IoT | Internet of Things |
| IP | Internet Protocol |
| LAN | Local Area Network |
| MPEG | Moving Picture Experts Group |
| NAT | Network Address Translation |
| NIST | National Institute of Standards and Technology |
| NFV | Network Functions Virtualized |
| NTP | Network Time Protocol |
| OS | Operative System |
| REST | Representational State Transfer |
| RCP | Remote Procedure Calls |
| RTMP | Real Time Messaging Protocol |
| TCP | Transmission Control Protocol |
| UI | User Interface |
| UDP | User Datagram Protocol |
| USB | Universal Serial Bus |
| VM | Virtual Machine |
| VNF | Virtual Network function |
| WebRTC | Web Real Time Comunication |

# Capítulo 1

# Introdução

A presente dissertação insere-se nas área da computação da *cloud* e dos *media*. Tem por objetivo encontrar soluções baseadas na virtualização de recursos que permitam suportar as diferentes fases da cadeia de valor do conteúdo, nomeadamente a contribuição, produção, edição e gestão de conteúdos, em tecnologias baseadas em *cloud*.

A mesma será realizada em ambiente empresarial, na MOG Technologies, empresa de desenvolvimento de soluções multimédia para a indústria audiovisual, situada na Maia.

Parte do trabalho desenvolvido na dissertação não está circunscrito apenas à MOG Technologies, sendo que, a empresa faz parte do consórcio de um projeto europeu designado de 5G City. É da responsabilidade da MOG Technologies adquirir o conhecimento e estudar uma solução ao nível de abstração e virtualização dos recursos localizados no *edge* para uma contribuição na virtualização da infraestrutura do projeto 5G City, a ser implementada em 3 diferentes cidades europeias (Barcelona, Bristol e Lucca).

Em relação ao projeto 5G city, é um projeto internacional que conta com 19 empresas participantes de 7 países diferentes, o qual teve início em Junho de 2017 e conta com uma duração prevista de 30 meses.

O objetivo final deste projeto passa por maximizar o retorno do investimento para toda a cadeia do mercado digital (utilizadores, aplicações, fornecedores de *cloud*, ou seja, os próprios municípios, fornecedores de telecomunicações e de infra-estrutura). Para tal, o principal objetivo do 5G City é construir e instanciar uma plataforma aberta (PaaS) *multi-tenant*, que estende o modelo da cloud (centralizado) até a extremidade da rede (*edge*), com uma demonstração nas três cidades europeias. Podendo assim, desta forma, avançar o estado da arte para resolver os principais desafios no domínio de virtualização do *edge* baseado em 5G, incluindo a perspetiva de um *host* neutro em ambientes de implementação como cidades.

Assim, a 5G City projetará, desenvolverá, implementará e demonstrará, em condições operacionais, uma plataforma de *cloud* distribuída para os municípios e proprietários da infraestrutura que atuam como *hosts* neutros 5G.

## 1.1 Contexto

Atualmente, observa-se a enfatização, por parte dos canais televisivos, de uma tendência de hiperlocalização de notícias que corre no sentido de aproximar o espetador ao seu quotidiano e às notícias que lhe são relevantes. Pelo que, estes canais, de carácter nacional, para apresentarem notícias de origem regional recorrem a uma rede noticiosa que apresentam na sua base os órgãos de comunicação regional, sejam eles rádios, jornais ou televisões locais. No entanto, esta solução adotada por estes canais é bastante pobre. Recorrem à voz e à imagem recolhidas por órgãos de comunicação regional e a forma de comunicação das referidas entidades caracteriza-se pela sua debilidade, sendo que os jornalistas responsáveis pela notícias regionais se encontram fora da cadeia dos canais de caráter nacional.

Posto isto, apresenta-se a necessidade de criar umas melhores condições para que os mesmos jornalistas possam interagir diretamente no processo de contribuição e assim aumentar as suas receitas que, por conseguinte, ajudará a sustentabilidade dos meios de comunicação social através do envio regular de notícias audiovisuais para os canais nacionais de natureza generalista. Para tal, é preciso arranjar uma forma de contribuição e comunicação mais viável entre estas entidades, que passará por um sistema distribuído que permita aos jornalistas locais contribuir com conteúdos *live* ou *not live* e que os canais de TV tenham acesso a estes conteúdos.

A utilização de tecnologias abertas baseadas na *cloud* na fase de contribuição, produção, gestão e distribuição de conteúdos permite a interligação deste conjunto de jornalistas distribuídos ao nível do território nacional, alavancando o nascimento de uma rede de contribuições regionais próximas dos locais das notícias e dos potenciais consumidores das mesmas. Desta forma, permite, simultaneamente, diversificar os conteúdos difundidos pelas estações de televisão que servirão como agregadores dos mesmos, e com isso aumentar o volume de negócios destas últimas.

No ponto de vista de quem gere esses conteúdos (o canal de televisão) a escolha de uma solução fiável revela-se complexa. Sendo que, mais do que uma escolha de módulos individuais que executam determinados serviços e que estão otimizados para determinadas funções, a interligação e integração dos mesmos, a criação de uma cadeia completa de produção de *media* aponta-se como o mais importante. Assim, poderá referir-se que importa criar mecanismos de conexão entre o envio dos conteúdos dos jornalistas para a *cloud*, a sua curadoria em ambiente totalmente virtual e, posteriormente, o suporte de múltiplos utilizadores através de tecnologias de distribuição de media como são as CDNs (Content delivery network) dedicadas.

## 1.2 Motivação e objetivos

A dissertação em causa procura desenvolver então uma solução de abstração e virtualização de recursos que irá suportar uma *framework* capaz de criar uma comunicação *end to end* entre quem capta a notícia e o espetador final. Para isso, a dissertação visa a conceção e desenvolvimento de um conjunto de micro serviços que permita a interligação transparente de diversos módulos funcionais, bem como a criação a camada de abstração que permita a utilização dos mesmos em diversos

casos de uso e *workflows*. Com isto, aponta-se como objetivo o estudo de soluções tecnológicas existentes no mercado, a definição de funcionalidades chave e o desenho de uma arquitetura de referência que permita suportar os desenvolvimentos efetuados. O desenvolvimento desta camada de abstração de recursos abordará componentes de diversas áreas da cadeia de produção de conteúdos, como a virtualização, a cloud, os micro serviços, serviços web, transmissão de vídeo e redes de distribuição de conteúdos, tendo sempre em mente a modularidade, escalabilidade, fiabilidade, concorrência e a distribuição de carga.

## 1.3 Estrutura do relatório

Este documento apresenta uma estrutura dividida em seis capítulos principais. No primeiro, e, portanto presente capítulo, é apresentada uma introdução no âmbito da dissertação com o intuito de fornecer uma explicação ao nível do contexto, das motivações e dos objetivos da mesma.

Posteriormente, o segundo capítulo, refere-se ao estudo do estado da arte, onde se encontra realizada uma abordagem aos conceitos associados da dissertação e ao trabalho já desenvolvido neste âmbito.

No terceiro capítulo, é realizada uma análise à solução de *crowd jornalism*, assim como os módulo que a constituem.

Já o quarto capítulo encontra o seu fundamento na descrição do problema a tratar, bem como a solução proposta a resolver o mesmo.

Em relação ao quinto capítulo, descreve a ambiente de testes e os procedimentos, sendo apresentados ainda os resultados dos testes desenvolvidos.

Finalmente, no sexto e último capítulo, é feita uma conclusão de toda a dissertação, e ainda é analisado aquilo que pode ser um trabalho futuro desta dissertação.

# Capítulo 2

# Estado da arte

No presente capítulo são apresentados os resultados de uma pesquisa sobre as diferentes temáticas, tecnologias e metodologias que suportam o projeto em causa.

Para se criar uma plataforma baseada em tecnologia *cloud* que permita o envio de notícias distribuídas por parte de jornalistas localizados em diferentes regiões e que ofereça a esses jornalistas, serviços remotos que lhes perita editar e produzir notícias com base nos conteúdos que eles próprios enviaram. O objetivo final é o de incorporar essas notícias regionais na programação nacional, apresentando-as de uma forma mais rica do que aquela que é utilizada hoje em dia pelos canais televisivos nacionais e permitindo aos jornalistas regionais serem eles próprios a criar a notícia e beneficiar dessa criação, ou seja, a contribuir diretamente.

Ao usar uma rede de contribuições usando tecnologias baseadas na *cloud*, que assenta na virtualização, é importante começar por fazer um estudo sobre as diferentes tecnologias de virtualização existentes.

De seguida, é necessário abordar os diferentes níveis de computação (*cloud*, *fog* e *edge*), já que estamos a lidar com uma grande área geográfica é necessário dividir a carga computacional em cada um dos níveis com o objetivo de obter melhores resultados, tanto em aspetos de eficiência como económicos.

Como se trata de uma rede de contribuições com um potencial de fontes de conteúdo bastante elevada é importante fazer a gestão de todas estas fontes assim como o processamento do conteúdo. Para tal, vão ser os estudados alguns dos gestores de redes de nuvens existentes.

Por fim, é feito um estudo no mercado com base nas soluções que já existem. Para tal, realiza-se uma descrição do panorama atual para cada um deles, enquadrando-os no âmbito do projeto, de forma a serem a justificação para futuras escolhas e decisões.

## 2.1 Codificação dos conteúdos audiovisuais

Este projeto tem na sua base a ideia de um smartphone usar uma *web app* para captar os conteúdos audiovisuais através da câmera do mesmo e transmitir ao vivo para os servidores de vídeo, com o objetivo de uma posterior gestão dos conteúdos. Para este fim, revela-se necessário

estudar os vários temas que irão suportar este *workflow* como os *codecs* de vídeo, os *codecs* de áudio e as técnicas de *streaming*. De seguida, procede-se à abordagem dos referidos temas e das tecnologias que existem no mercado.

### 2.1.1 Formatos

A decisão do formato do conteúdo para streaming através da web apresenta a necessidade de ter em conta as possíveis limitações de rede dos vários utilizadores que irão contribuir com os conteúdos audiovisuais, assim como o sistema operativo do smartphone. A velocidade de upload média da internet móvel em Portugal, em Maio de 2018, foi de 9,37 Mbps, segundo o ranking global de velocidades da empresa de análise e testes da internet Ookla. [1]

Atualmente, os formatos mais importantes para dispositivos móveis, como o Iphone, Ipad, Ipod e smartphones, em geral, é o formato MP4 (MPEG-4 part 14, extensão .mp4).

O formato MP4 é um variável do MPEG. Pelo que, quando se trata de vídeos gravados ao vivo, ou streamings, é o formato mais indicado. Apresenta como vantagem o suporte nativo a legendas, codecs Xvid, DivX e H.264, um dos codecs mais usados, para vídeos. Para além de possuir, também, o suporte ao codec ACC para áudio.

Assim, quandos se pretende um equilíbrio entre qualidade e compactação, em princípio e em geral, o MP4 é o preferido.

### 2.1.2 Codecs de vídeo

Um codec é um pacote de software ou hardware destinado à codificação de um conteúdo multimédia através da aplicação de uma dada técnica.

Essa codificação poderá ser feita com ou sem compressão da informação inicial do conteúdo, sendo que havendo compressão, a mesma pode ser feita com perdas ou sem perdas.

Apontam-se como possíveis as seguintes formas de compressão:

- compressão sem perdas:
  - compressão sem perdas matematicamente: não há, de facto, nenhuma diferença entre o conjunto de bits inicial e o conjunto de bits que é obtido após a descodificação;
  - compressão sem perdas visualmente: há algumas perdas, no entanto as mesmas não são percetíveis ao sistema visual humano;
- compressão com perdas.

Os quatro codecs de vídeo com perdas mais eficientes e mais utilizados, atualmente, são o H.264 (ou AVC), o H.265 (ou HEVC), o VP8 e o VP9.

Outros codecs existentes são utilizados com alguma quota de relevância, mas antes de serem transmitidos pelas plataformas de streaming são recodificados num destes codecs.

#### 2.1.2.1 H.264

O H.264 caracteriza-se por ser o formato mais conceituado tanto em transmissões de canais de televisão como streaming via Internet e é suportado pela maioria dos navegadores de internet, sistemas de televisão e telemóveis atuais. Pelo que se poderá afirmar que o mesmo é utilizado pela maior parte dos sistemas de televisão digital terrestre de diversos países (debaixo da norma DVB-T) e plataformas como o Netflix, Youtube, Vimeo, entre muitos outros. O codificador x264 é o software open-source mais utilizado para codificar vídeo em H.264. Este foi desenvolvido em 2003 pelo ITU-T Video Coding Experts Group e pelo ISO/IEC JTC1 Moving Picture Experts Group. [2] [3]

#### 2.1.2.2 H.265

O H.265 foi lançado posteriormente do H.264, caracterizado por ser o seu sucessor, desenvolvido pelo mesmo grupo de trabalho, em 2013. Os seus objetivos passaram pela transmissão de conteúdos em HD e SD com menos *bitrate*, bem como agilizar a transmissão de conteúdos em Ultra HD (2K e 4K) via internet. A codificação pode ser feita com o software open-source x265. [4]

#### 2.1.2.3 VP8

O formato VP8 surgiu por meio da Google quando deu o início de um novo projeto de media aberta "WebM", dedicado ao desenvolvimento de um formato de media aberto e de alta qualidade para a web, que está disponível gratuitamente para todos. O formato VP8 foi originalmente desenvolvido por uma pequena equipa de pesquisa da On2 Technologies, Inc. como um sucessor de sua família VPx de codecs de vídeo. Ao comparar o VP8 com outros formatos de codificação de vídeo, verifica-se que este possui muitos recursos técnicos diversificados, o que lhe permite alcançar uma alta eficiência de compactação e baixa complexidade computacional para descodificação, em simultâneo. Entre estes recursos, destacam-se a necessidade de pouca largura de banda e a compatibilização com a maioria dos vídeos da web. Esta última, constata-se através da amostragem 4:2:0, profundidade de cor de 8 bits por canal, varredura progressiva, não entrelaçada, e dimensões da imagem até um máximo de 16383x16383 pixels.

Desde o anúncio do WebM, o VP8 não obteve apenas um forte apoio de uma extensa lista de importantes participantes do setor, como também começou a atrair um interesse na comunidade de pesquisa em codificação de vídeo, tanto da parat industrial como académica. [5]

#### 2.1.2.4 VP9

O VP9 é outro formato desenvolvido pela Google, o mais recente, que compete com o H.265. A biblioteca de software open-source desenvolvida para a codificação designa-se de libvpx e, atualmente, é bastante utilizada em vídeos do YouTube com resolução superior a 1080p. [6] O

H.265 e o VP9 ainda não são descodificáveis na maioria dos dispositivos comparativamente ao H.264, mas a sua adesão tem vindo a aumentar.

Os testes de eficiência de compressão de codecs de vídeo estão dependentes de vários fatores. Isto porque, há várias métricas que podem ser calculadas, e cada uma delas terá um resultado diferente em função de definições da codificação, bitrate do vídeo, estrutura do GOP, resolução do vídeo, entre outros fatores. No entanto, de uma forma genérica, num estudo de 2016 realizado por De Cock et al concluíram que com os codecs H.265 e VP9 apresenta-se a possibilidade de poupar até 40-50% de bitrate em resoluções até 1080p comparativamente ao codec H.264. O H.265 tem uma performance superior ao VP9 na maior parte dos cenários, mas decresce consoante o aumento da resolução do vídeo, revelando-se inferior perante o VP9 em vídeo 1080p. [6] [3] Uma grande parte dos avanços feitos pelo VP9 apresentam-se como uma evolução natural do VP8. O desafio técnico é o de garantir que todas as ferramentas de codificação trabalhem em conjunto de maneira altamente eficiente. [7]

### 2.1.3 Codecs de áudio

Os codecs de áudio mais recentes permitem sinais de alta qualidade até banda completa (20 kHz), o que geralmente é associado à largura de banda máxima audível. Na atual comunicação em tempo real baseada em IP, um dos codecs de áudio predominantes em banda larga é o Opus codec Internet.[8]

#### 2.1.3.1 OPUS

O Opus é um codec de áudio totalmente aberto, livre de royalties e altamente versátil. O Opus é incomparável para transmissão interativa de fala e música pela Internet, mas também é destinado a aplicações de armazenamento e streaming. Este é padronizado pela IETF (Internet Engineering Task Force) como RFC 6716, que incorpora a tecnologia do codec SILK do Skype e do codec CELT do Xiph.Org. [9] O mesmo é composto por uma camada baseada em predição linear e uma camada que se baseia na transformação discreta modificada do cosseno. Assim, um codec com duas camadas disponíveis pode operar numa faixa mais ampla do que qualquer um sozinho, bem como pode alcançar uma melhor qualidade através da sua combinação. [10]

O Opus pode lidar com uma ampla gama de aplicações de áudio, incluindo voz sobre IP, videoconferência, diálogo em jogos e até mesmo transmissões de música ao vivo.

As características deste codec são:

- Bitrates de 6 kb/s a 510 kb/s;
- Taxas de amostragem de 8 kHz (narrowband) a 48 kHz (fullband);
- Tamanho das frames de 2.5 ms a 60 ms;
- Suporte para bitrate constante (CBR) e bitrate variável (VBR);
- Suporte para fala e música;

- Suporte para som mono e estéreo;
- Suporte para mais de 255 canaiss (multistream frames);
- bitrate ajustável dinamicamente;
- Robusto a perdas.

#### 2.1.3.2 AAC

Outro codec de áudio predominante é o Advanced Audio Coding (AAC), projetado para ser o sucessor do formato MP3 e utiliza uma codificação para compressão com perda de dados de som digital.

O AAC foi padronizado pela ISO (Organização Internacional de Normalização) e pela IEC(Comissão Eletrotécnica Internacional ) como parte integrante das especificações MPEG-2 e MPEG-4. O padrão MPEG-2 contém vários métodos de codificação de áudio, incluindo o esquema de codificação MP3. A qualidade estéreo é satisfatória para requisitos modestos de 96 kbit/s em modo *joint stereo*. Os testes de áudio MPEG-2 demonstraram que o AAC atinge os requisitos de "transparência"para taxas de 128 kbit/s em estéreo e 320 kbit/s para áudio. [11]

O AAC é o formato padrão para o sistema operativo IOs e Android, um dado relevante a tomar em conta, visto que o projeto assenta na contribuição de conteúdos audiovisuais através dos *smartphones*.

## 2.2 *Streaming* de conteúdo multimédia

A escolha de uma tecnologia de streaming envolve várias considerações, o que inclui uma análise das vantagens e desvantagens do protocolo de streaming usado pela tecnologia. Esta secção tem como objetivo fazer a análise dos principais protocolos existentes, assim como apropriados para este projeto.

### 2.2.1 Técnicas de *streaming*

Os protocolos de comunicação ditam como os dados são comunicados, definem elementos como a estrutura dos cabeçalhos dos dados, a autenticação e o tratamento de erros. Existem dezenas de protocolos, pelo que aqui vão ser analisados o RTMP (Real Time Messaging Protocol) e o webRTC (Web Real Time Communication) juntamente com websocket.

#### 2.2.1.1 RTMP

O RTMP (*Real Time Messaging Protocol*) é um protocolo desenvolvido pela Adobe Systems que utiliza por padrão a porta 1935 e foi utilizado em primeiro lugar no servidor de comunicação da Flash.

Este protocolo recorre a um outro protocolo, o TCP/IP, com a finalidade de transmissão de pacotes e não é usado para RPC (Remote Procedure Calls). O RTMP mantém uma conexão persistente com o servidor e permite a comunicação em tempo real de dados do tipo: áudio, vídeo e objeto. Para ser realizada a comunicação RTMP, é necessário, além do servidor, um arquivo swf. Este arquivo deverá ser compartilhado pelos utilizadores finais e deve ser dotado de chamadas RTMP, normalmente codificadas em Action Script.

Os campos de aplicação mais comuns desta tecnologia são a videoconferência, VOD (Video on Demand), a transmissão de vídeos ao vivo e a chamada de vídeo. [12] [13]

### 2.2.2 RTP

O RTP (*Real Time Protocol*), fornece serviços de entrega *end-to-end* para dados em tempo real, como áudio e vídeo interativos. Esses serviços incluem a identificação do tipo de *payload*, numeração da sequência, registo de data e hora e monitorização da entrega. Aplicações normalmente executam RTP em cima de UDP para fazer uso de seus serviços de multiplexação e checksum; ambos Os protocolos contribuem com partes da funcionalidade do protocolo de transporte.

No entanto, a RTP pode ser usada com outra rede subjacente adequada ou protocolos de transporte (consulte a Seção 11). O RTP suporta a transferência de dados para vários destinos usando a distribuição multicast se fornecido pelo rede subjacente.

Observe que o próprio RTP não fornece nenhum mecanismo para garantir entrega ou fornecer outras garantias de qualidade de serviço, mas em serviços de camada inferior para fazer isso. Não garante a entrega ou impedir a entrega fora de ordem, nem presume que a garantia subjacente rede é confiável e entrega pacotes em sequência. A sequência números incluídos na RTP permitem ao receptor reconstruir o sequência de pacotes do remetente, mas os números de sequência também podem ser usados para determinar a localização adequada de um pacote, por exemplo, em vídeo decodificação, sem necessariamente descodificar pacotes em sequência.

Embora a RTP seja principalmente projetada para satisfazer as necessidades de participante de conferências multimídia, não se limita a aplicação particular. Armazenamento de dados contínuos, interativo simulação distribuída, crachá ativo e controle e medição As aplicações também podem considerar a RTP aplicável.

[14]

#### 2.2.2.1 WebRTC

O WebRTC é uma API em desenvolvimento elaborada pela World Wide Web Consortium (W3C) que providencia aos navegadores e às aplicações móveis a comunicação em tempo real (RTC) sem a necessidade de plugins. Esta iniciativa é suportada pela Google, Mozilla, Opera, entre outros. [15]

O WebRTC permite que os navegadores comuniquem em tempo real através de uma arquitetura *peer-to-peer*. Trata-se de uma comunicação segura de áudio / vídeo (e dados) entre navegadores HTML5. O WebRTC reúne dois campos historicamente separados, as telecomunicações e

o desenvolvimento da Web. Este apresenta como objetivo permitir o desenvolvimento de aplicações RTC de alta qualidade para o navegador, plataformas móveis e dispositivos IoT (*Internet of Things*), permitindo que todos se comuniquem por meio de um conjunto comum de protocolos. [16] [17]

## 2.3 Virtualização

A virtualização é uma técnica que torna possível a criação de múltiplos ambientes simulados ou recursos dedicados a partir de um único sistema de hardware físico. Esta apresenta na sua génese a obtenção de uma imagem virtual de algo como um servidor, um sistema operacional, um dispositivo de armazenamento ou uma rede de recursos para que estes possam ser usados em várias máquinas ao mesmo tempo. O principal objetivo da virtualização aponta-se como a gestão da carga de trabalho ao transformar a computação tradicional com vista a torná-la mais escalável, eficiente e económica. Ao aplicar-se a virtualização dos servidores que vão ser usados no projeto revelam-se benefícios como: a eficiência, a escalabilidade e a redução de custos. [18]

Existem três tecnologias utilizadas para disponibilizar a virtualização necessária à partilha de recursos em ambientes de *cloud computing*: Máquinas Virtuais, *Containers*, *Unikernel*. Ambas as tecnologias cumprem os requisitos de isolamento destas plataformas, no entanto existem algumas diferenças fundamentais na sua arquitetura (ver figura 2.1).

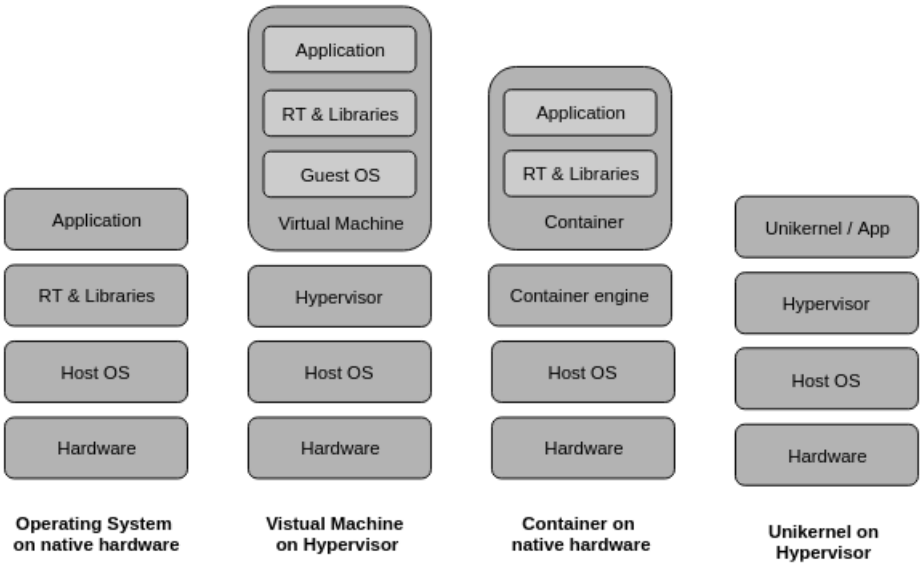



Figura 2.1: VMs vs *containers* vs *unikernels*

### 2.3.1 Máquinas virtuais

Uma máquina virtual (VM) caracteriza-se por ser equivalente a um sistema operativo emulado que é executado por cima de outro sistema operativo. O utilizador que usa uma máquina virtual tem a mesma experiência como se tivesse este mesmo hardware.

As VMs, através do hipervisor, podem ter acesso a qualquer número de recursos: poder de computação, ao CPU e à memória da máquina hospedeira (máquina física ou virtual onde este está instanciado) mas de uma forma controlada e assistida; um ou mais discos físicos ou virtuais para armazenamento; uma interface de rede virtual ou real; bem como quaisquer dispositivos, como placas de vídeo, dispositivos USB ou outros hardwares partilhados com a VM.

Os hipervisores são a tecnologia que permite a criação e a execução de VMs.

Esta tecnologia tem vindo a ser dividida em dois tipos:

- Tipo 1, ou hipervisores "*bare metal*", executam VMs *Guest* diretamente no hardware do sistema comportando-se como um sistema operativo. Requerem uma consola de gestão e são normalmente usados em *datacenters*. Exemplos deste tipo de hipervisores sao Oracle OVM for SPARC, ESXi, Hyper-V and KVM ;

- Tipo 2, ou hipervisores "*hosted*", comportam-se mais como as aplicações tradicionais que podem ser iniciadas e paradas como um programa normal, coordenam as chamadas para o CPU, memória, disco, rede e outros recursos através na máquina física hospedeira. Exemplos deste tipo de hipervisor são VMware Fusion, Oracle Virtual Box, Oracle VM for x86, Solaris Zones, Parallels and VMware Workstation.

Cada máquina virtual tem um sistema operativo instalado, o que permite maior flexibilidade no aprovisionamento de aplicações que necessitem de diferentes sistemas operativos ou de diferentes versões destes. Apesar desta flexibilidade, uma VM tem certas desvantagens em relação a um *container*. São mais difíceis de instanciar, devido ao trabalho acrescido da instalação de um sistema operativo e à fragmentação resultante do uso de diferentes versões dos mesmos. Têm geralmente pior desempenho e consomem mais recursos computacionais, devido ao *overhead* de correr um sistema operativo. No entanto, por outro lado, é uma tecnologia que executa a aplicação num ambiente bastante isolado do sistema operacional da máquina hospedeira, ao contrário dos *containers* que se trata de apenas de um processo isolado que partilha o mesmo *kernel* do sistema operacional da máquina hospedeira, o que traz benefícios a nível de proteção e segurança.

### 2.3.2 *Containers*

Ao contrário das máquinas virtuais, os *containers* não correm um sistema operativo completo, em vez disso partilham o da máquina *host*. [19] Por um lado, o aprovisionamento de aplicações neste tipo de tecnologia é mais limitado, ao não permitir a execução de diferentes OSs. No entanto, apresenta ganhos a nível de desempenho, dos quais o consumo de muito menos recursos e ter menos *overhead* computacional. É também mais ágil a nível de escalabilidade, tanto horizontal como vertical, ao ser mais simples a alocação dinâmica de recursos e ao ser consideravelmente mais rápido arrancar *containers* do que VMs. [20]

Uma grande vantagem da utilização de *containers* é a possibilidade de isolar cada módulo de uma aplicação, cada um com as suas configurações e dependências, evitando assim problemas de

incompatibilidade entre diferentes componentes. Esta abordagem permite maior controlo sobre o ambiente de implantação, tornando este processo mais previsível.

A tecnologia de *containers* Linux (LXC) surge em 2008 com a introdução, na versão de *kernel* 2.6.24, da funcionalidade cgroups, que permite limitar os recursos disponíveis aos processos (como CPU, memória ou rede) [Zap12]. Esta gestão de recursos, aliada à utilização de *namespaces* que isolam um grupo de processos dos restantes no sistema, permite a virtualização de aplicações sem ser necessário instanciar máquinas virtuais. Nos últimos anos tem-se assistido a um grande crescimento no mercado da tecnologia de *containers*.

### 2.3.3 Unikernels

*Unikernels* são um reaparecimento do conceito do sistema operacional da biblioteca, que se trata de um sistema operacional absolutamente básico que possui o número mínimo de bibliotecas necessárias para executar um aplicativo específico. Os sistemas operacionais de bibliotecas datam de meados da década de 1990, quando foram usados em sistemas operacionais académicos, como o MIT Exokernel[21] e o Nemesis. [19]

O objetivo do *unikernels* é reestruturar por inteiro as VMs, incluindo todo o código *kernel* e *userspace*, em componentes mais modulares que são flexíveis, seguros e reutilizáveis, como um sistema operacional de biblioteca, de acordo com Anil Madhavapeddy e David J. Scott num comunicado do artigo da ACM (Association for Computing Machinery). [22] Mais precisamente, os *unikernels* são, na verdade, imagens executáveis que podem ser executadas num hipervisor sem a necessidade de um sitema operativo "guest"como acontece com as VMs. Estas imagens contém o código da aplicação assim como as funções do sistema operativo requeridas pela aplicação. [23]

Os Unikernels geralmente são construídos usando compiladores que utilizam sistemas operacionais de bibliotecas, que são coleções de bibliotecas que representam os principais recursos de um sistema operacional. Isso permite que um desenvolvedor unikernel inclua seletivamente apenas os componentes de biblioteca necessários para fazer um aplicativo funcionar, com o código unikernel orquestrando esses drivers. As funções tradicionais do sistema operacional, como o gerenciamento de rede ou do sistema de arquivos, são compiladas no executável final, conforme necessário.

Os Unikernels usam uma fração dos recursos exigidos pelos sistemas operacionais completos como distribuições Linux ou Microsoft Windows Server.

Comparado com as VMs e os *containers*, o tamanho reduzido dos *unikernels* permite tempos de inicialização inferiores ao segundo e altas densidades de implantação incomparáveis. Além disso, o espaço ocupado pelas funções do sistema operacional da biblioteca e a ausência de utilitários tradicionais do sistema operacional reduzem bastante a vulnerabilidade a hackers. Alguns sistemas de compilação unikernel utilizam linguagens seguras de tipos, como Haskell ou Erlang, enquanto outros podem se ligar a linguagens mais comuns, como C, C ++ ou Java. [24] Mas embora o *unikernel* à primeira vista possa parecer uma excelente escolha em termos de consumo de recursos, rapidez de resposta, proteção a vulnerabilidades e por se apresentarem bastante mais leves, o facto de serem criados à volta da aplicação que se pretende executar pode representar

um grande desafio, nomeadamente de low-level programming, como referiu Joshua Woods, chefe de documentação do CoreOS (empresa de containers e Kubernetes), no Open Source Summit em Los Angeles. Outra adversidade advém de se basear muito na virtualização de hardware, mais precisamente os custos de paravirtualização, que acaba por afetar o balanço geral dos unikernels.

## 2.4 Níveis de computação

A computação na nuvem mudou radicalmente a maneira como vivemos e trabalhamos desde 2005, ano em que esta surgiu. [25] Serviços segundo o modelo *software* como serviço (SaaS), como a Google Apps, Twitter, Facebook e Flickr, têm sido bastante usados no nosso dia a dia.

A Internet das coisas (IoT) foi primeiramente introduzida em 1999 na gestão das cadeias de abastecimento, seguindo-se depois a introdução do conceito de "tornar uma informação sensível ao computador sem a ajuda da intervenção humana", que rapidamente foi adaptado a outros campos de aplicação, como saúde, casa, meio ambiente e transportes [26].

Até 2020 é estimado que haverá 50 biliões de "coisas"conectadas à Internet, segundo a Cisco Internet Business Solutions Group [27]. No final do ano 2021, os dados produzidos por pessoas, máquinas e "coisas" chegarão aos 847 zettabytes, segundo a cisco cloud index 2016-2021 [28] (ver figura 2.2).

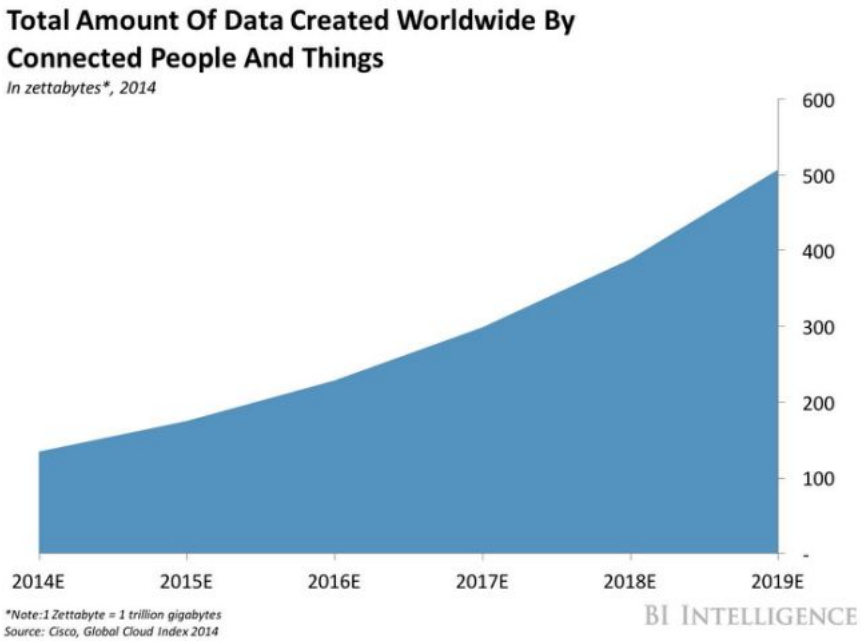



Figura 2.2: Evolução da quantidade de dados criados em todo o mundo [29]

.

Algumas aplicações IoT podem exigir tempos de resposta muito curtos, envolver dados privados e ainda produzir uma grande quantidade de dados, o que poderia representar uma carga pesada para a rede. A computação na nuvem não é eficiente o suficiente para suportar estas aplicações.

Com o impulso dos serviços de nuvem e da IoT, o *edge* da rede está a tornar-se para além de consumidor, um produtor de dados. O que nos traz a uma era pós-nuvem [30], onde há uma grande quantidade de dados gerados por "coisas"que fazem parte da nossa vida diária, e muitas aplicações foram e ainda continuam a ser instanciadas no *edge* da rede onde irão consumir esses dados.

Em 2019 45% dos dados criados pela IoT vão ser armazenados, processados e analisados perto ou no *edge* da rede, o que mostra que a utilização de tanto a computação *edge* como a computação *fog* tem vindo a crescer bastante.

Na figura 2.3 podemos observar estas 3 camadas de computação.

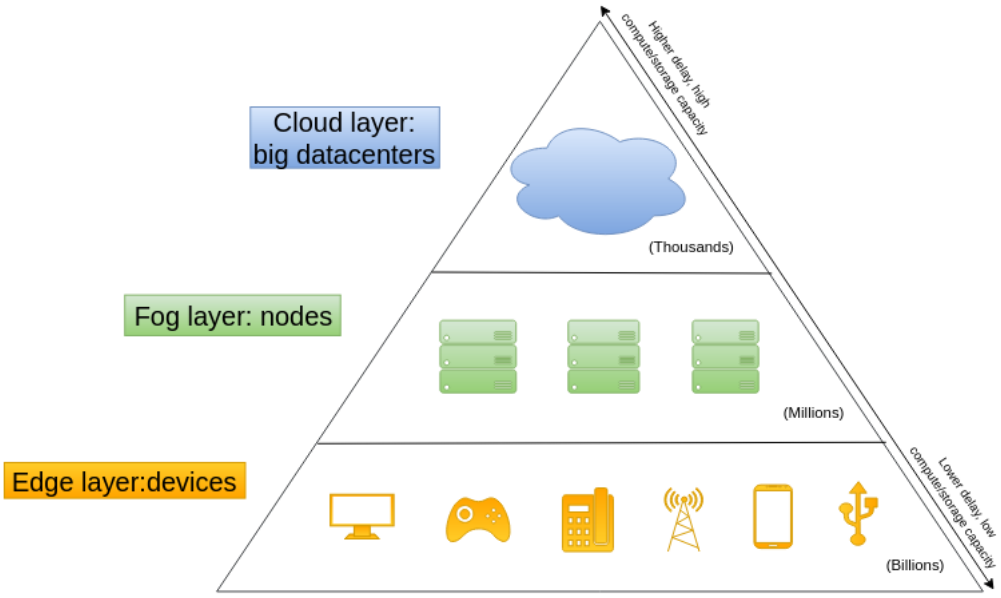



Figura 2.3: Várias camadas de computação existentes

### 2.4.1 Computação na *cloud*

Computação na cloud pode ser definido como um serviço que possui um grupo de recursos computacionais partilhados ( rede, servidores, armazenamento, aplicações e serviços) que são disponibilizados *on-demand*, ou seja, consoante o que o cliente necessita. Este serviço é acessível a todos os utilizadores que pretendam requisitá-lo, em qualquer sítio do mundo, de uma forma rápida e eficiente, sem muito esforço.

De acordo com a NIST [31], são cinco as características essenciais deste serviço:

- Serviço *self-service on-demand*: os recursos são providenciados de forma automática ao utilizador, sem intervenção humana por parte dos provedores dos serviços em questão.
- Amplo acesso de rede: os recursos são facilmente acessíveis na rede, em vários tipos de dispositivos (smartphones, tablets, laptops, etc.).

- Agrupamento de recursos: os recursos do provedor de serviços estão configurados de forma a servir múltiplos utilizadores (*multi-tenancy*), e são alocados de forma dinâmica de acordo com a procura de cada um.

- Rápida elasticidade: os recursos disponíveis para provisão podem escalar, em certos casos automaticamente, de acordo com a procura do utilizador. Podem aparentar ser ilimitados.

- Serviço monitorizado: os sistemas na nuvem automaticamente retiram métricas da utilização dos recursos, utilizando-as para optimizar os mesmos e reportando-as, dando assim maior transparência tanto ao provedor e utilizador.

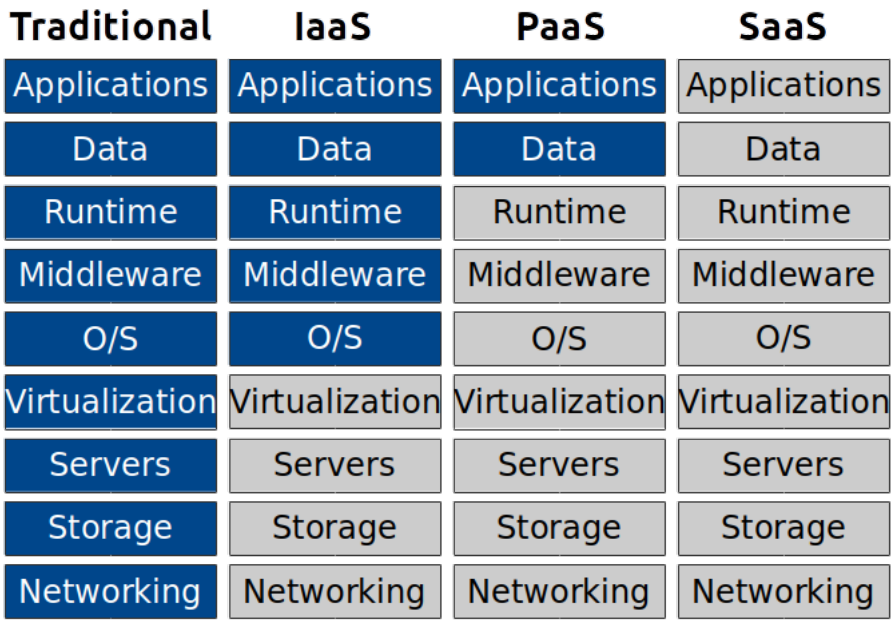



Figura 2.4: Diferentes modelos de serviço existentes da computação na nuvem.

Relativamente ao modelo do serviço, podemos identificar quatro diferentes tipos de modelos com níveis crescentes de abstração e virtualização. Estes modelos representam uma transformação na forma de disponibilização e consumo do software. A nuvem cada vez mais é utilizada para abstrair certas camadas inferiores de uma aplicação. Como podemos ver na figura 2.4, onde as camadas cinzentas representam as camadas providenciadas e as camadas azuis representam as camadas arbitrárias, os modelos são:

- Tradicional: toda a infraestrutura é gerida pelo utilizador.

- Infraestrutura como serviço (IaaS): os recursos computacionais básicos (processamento, rede ou armazenamento) são providenciados, e o utilizador pode executar software arbitrário, incluindo sistemas operativos e aplicações.

- Plataforma como serviço (PaaS): para além dos recursos básicos, os sistemas operativos também são providenciados. O utilizador continua a configurar as suas próprias aplicações, e pode ter algum controlo sobre as mesmas através de ficheiros de configuração.
- Software como serviço (SaaS): todas as camadas são abstraídas. O utilizador não tem controlo das aplicações, acedendo às mesmas através de diferentes dispositivos clientes.

Estão ainda definidos quatro modelos de instanciação, que dizem respeito aos utilizadores, entidade responsável e localização da cloud:

- Nuvem privada: destina-se ao uso interno de uma organização, pode ser gerida pela própria ou outra entidade, e pode existir dentro ou fora das instalações da mesma.
- Nuvem comunitária: destina-se ao uso por parte de uma comunidade com fins em comum e pode ser gerida e estar localizada nas instalações de uma ou mais das organizações intervenientes.
- Nuvem pública: destina-se ao uso do público geral, pode ser gerida por uma entidade comercial, académica ou governamental e está localizada nas instalações dessa entidade.
- Nuvem híbrida: é uma combinação de dois ou mais tipos de nuvens anteriores que, apesar de existirem separadamente, estão ligadas por tecnologias que permitem a portabilidade entre elas.

### 2.4.2 Computação *Fog*

Computação *fog* estende o paradigma da computação na nuvem.

Em relação à computação na nuvem tradicional, o processamento dos dados deixa de ser num *data center* local para ser feita num *data center* remoto providenciado por algum serviço de nuvem como AWS, Azure, Google Cloud ou Oracle. Enquanto que na computação *fog*, o processamento é afastado da nuvem, isto é, do *data center* privado. Passa a ser executado no *edge* da rede, aproximando assim o processamento da nuvem para mais próximo de onde os dados são gerados ( dispositivos finais ) para um nó *fog*. Deste modo, é possível economizar tempo e recursos visto que a necessidade de enviar os dados para o *data center* já não existe. [32]

É esta mudança das capacidades computacionais da nuvem, de algum *data center* virtualizado, para algo mais próximo dos end devices, algo mais real.

Em suma, trata-se de uma plataforma virtualizada que providencia serviços de computação, armazenamento e de rede entre os dispositivos finais e os *data centers* que compõem a tradicional computação na nuvem. Esta computação requer uma série de características, o que faz com que seja uma extensão não trivial da computação tradicional na nuvem [33]. Estas características são:

- Localização do *edge* e baixa latência. Desde o início da computação *fog* que o objetivo foi e continua a ser suportar *endpoints* com serviços localizados no *edge* da rede, como aplicações que requerem baixa latência (*streaming* de vídeo, *gaming*).

- Distribuição geográfica. Ao contrário do que acontece na computação na nuvem a qual ocupa uma posição mais centralizada a todos os serviços alvo, na computação *fog* como se localiza mais perto dos end devices é necessário que haja uma distribuição mais ampla dos nós de processamento, de forma a satisfazer todos os serviços pedidos.

- Número elevado de nós *fog*, como consequência do ampla geo-distribuição.

- Suporte para mobilidade. É essencial para muitos aplicativos *fog* para se comunicar diretamente com dispositivos móveis, e, portanto, apoiar técnicas de mobilidade, como o Protocolo LISP 1, que em vez de um só endereço divide este em dois, um para a identidade do *host* e outro para o local da identidade.

- Interação em tempo real. Aplicações que pretendem usar a computação *fog* envolvem interações em tempo real, em vez de processamento em lote.

- Predominância do acesso sem fios.

- Heterogeneidade. *fog* nodes podem ser implantados numa ampla variedade de ambientes.

- Interoperabilidade. O suporte de determinados serviços, como por exemplo o streaming de um vídeo, pode requerer a interação de diferentes fornecedores. Portanto, componentes *fog* devem ser capazes de interagir e comunicar com outros sistemas.

- Suporte para análise em tempo real e interação com a nuvem. O *fog* está posicionado para desempenhar um papel significativo na ingestão e processamento dos dados próximos à fonte.

### 2.4.3 Computação no *Edge*

O conceito *edge* computing surgiu quando houve a necessidade de processar parte dos dados na network *edge*, mais concretamente nos dispositivos finais, em vez de processar todos os dados na nuvem, ou num datacenter local. Ao contrário do que acontece na computação *fog*, em que as capacidades de processamento da nuvem são migradas para um datacenter local, na computação *edge* estas capacidades são migradas precisamente para o local onde dos dados são gerados, os dispositivos finais. Com a utilização do *edge* computing é possível obter uma otimização em aspetos como a latência, custo da bandwith, segurança, privacidade e no caso dos smartphones o tempo limitado da bateria. [34] Atualmente podemos afirmar que existe 3 grandes razões para a computação *edge* se tornar essencial.

A primeira razão é relativa aos serviços de nuvem. A mudança de todas as tarefas computacionais para a nuvem tem sido uma forma eficiente de processar os dados, pela simples razão de haver mais poder computacional na nuvem do que na network *edge*. No entanto, apesar desta velocidade de processamento ter aumentado rapidamente, a largura de banda das redes que transportam os dados da nuvem e para esta igualmente não tem aumentado o suficiente. Com o aumento dos dados

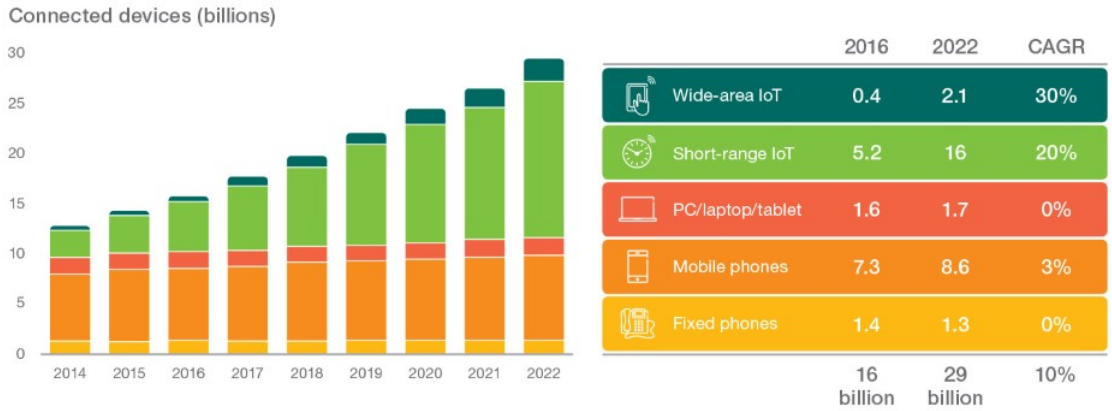



Figura 2.5: Evolução dos aparelhos conectados existentes [35].

que os aparelhos da *edge* geram, a rede está-se a tornar o bottleneck da computação na nuvem. Daí a importância de processar pelo menos parte desses dados na *edge*.

Outra razão está relacionada com a internet das coisas (IoT). O número de aparelhos que fazem parte desta IoT tem vindo a aumentar bastante, como podemos observar na figura 2.5, e consequentemente também o tamanho dos dados gerados por estes aparelhos. Computação de nuvem tradicional não vai ser eficiente o suficiente para lidar com todos os dados que irão ser gerados.

Por último, a transformação que se tem observado no *edge* dos consumidores de dados para produtores de dados. Na computação na nuvem, aparelhos como telemóveis no *edge* da rede tradicionalmente apenas consumiam dados. Nos dias de hoje, o cenário é diferente, os utilizadores produzem também os dados com os telemóveis, como fazer upload de fotos ou vídeos nas redes sociais. Esta mudança requer mais funcionalidade no *edge* da rede, como por exemplo, um vídeo que o utilizador queira carregar para uma rede social baseada na nuvem como Facebook ou Twitter. Este vídeo se for muito grande vai precisar de bastante largura de banda, então, neste caso, o aparelho em causa pode processar o vídeo de forma a reduzir a resolução, e assim baixar o tamanho deste antes de fazer upload para a cloud. Outro exemplo que pode ser dado, em vez de fazer upload de dados sensíveis, recolhidos por alguns dispositivos, para a nuvem através de uma rede vulnerável, estes dados podem ser processados por dispositivos equipados adequadamente no *edge*, o que permite uma melhor proteção e garantia de privacidade destes dados.

### 2.4.4 Segurança dos *data centers*

A ideia das camadas de computação *edge* e *fog* é colocar os serviços computacionais mais perto dos aparelhos finais em *data centers* mais próximos da população, ou mesmos nestes aparelhos.

Em relação à camada de computação *fog*, ao migrar estes serviços computacionais dos grandes *data centers* construídos fora dos centros populacionais para *data centers* de menor dimensão mas mais perto da população envolve questões de segurança. Ao adotarem esta localização menos protegida a segurança da infraestrutura corre mais riscos. Com tantos dados, desde informações pessoais até propriedade intelectual, a serem transmitidos e armazenados nestes *data centers*, proteger essa informação torna-se um grande desafio.

A segurança na entrada principal do *data center* é uma necessidade, regular o acesso ao data center com um sistema de segurança à entrada. No entanto, pode não ser suficiente adotar apenas este método de segurança. Ameaças internas, incluindo também erros humanos, são um risco que está sempre presente e que representa a maior causa dos *data centers* ficarem inativos [36].

Segundo Paul Mott [37], é necessário restringir o acesso apenas às pessoas de confiança, mas para além de focar em manter as pessoas fora, é necessário monitorizar o que acontece dentro do *data center*, nomeadamente acompanhar o acesso aos equipamentos do *data center* destas pessoas, desde recolher a informação de quem acedeu a o que é que fez.

## 2.5 Gestor de redes de nuvens privadas e ou públicas

Neste subcapítulo irão ser analisados alguns softwares que têm como campo de aplicação a criação e gestão de nuvens públicas ou privadas, o OpenStack e o Apache Cloudstack.

### 2.5.1 OpenStack

O OpenStack [38] é uma plataforma open-source gratuita que permite a criação de nuvens públicas e privadas, assim como a gestão destas.

Foi lançado em julho de 2010 pela NASA (agência espacial americana) juntamente com a Rackspace (empresa americana de computação na nuvem), e é, neste momento, gerida pela fundação OpenStack. Esta fundação é uma organização sem fins lucrativos criada em Setembro de 2012 com o objetivo de promover o desenvolvimento e a distribuição deste software. Atualmente o projeto OpenStack é suportado por mais de 500 empresas.

Esta plataforma é implementada segundo o modelo IaaS (infraestrutura como serviço), ou seja, os recursos computacionais básicos (como processamento, rede ou armazenamento) são providenciados, e o utilizador pode executar software arbitrário, incluindo sistemas operativos e aplicações, e facilita o controlo de largos recursos de computação, rede e armazenamento através de um *datacenter*, todos estes geridos a partir do *dashboard* ou da API (Aplication Program Interface). [39]

Em termos de arquitetura (ver figura 2.6 ) , este software divide-se em vários componentes que irão ser analisados de seguida.

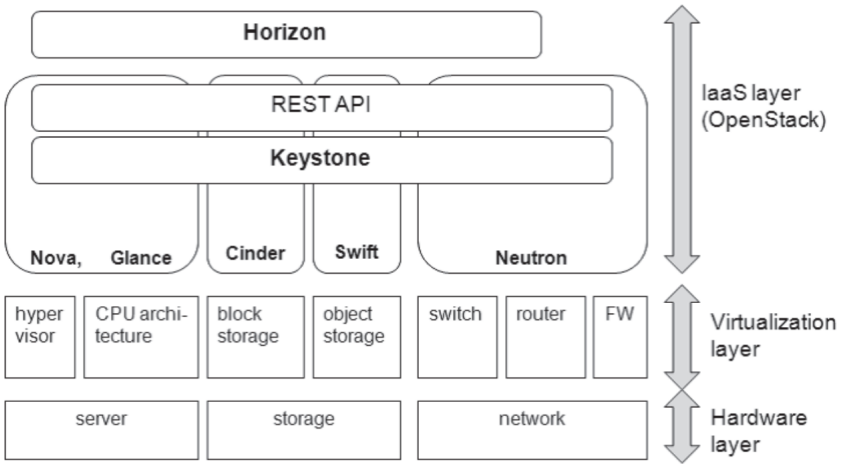



Figura 2.6: Arquitetura do OpenStack

#### 2.5.1.1 Componentes software do OpenStack

Este software consiste num grupo de 31 sub-projetos relacionados, em que cada um representa um dado serviço, entre eles serviços de rede, computação ou armazenamento. Todo este conjunto de projetos providencia todos os recursos e ferramentas necessárias para criar uma infraestrutura de serviço de nuvem.

Cada um destes projetos possui a sua própria API para facilitar a integração de todo o software.

A figura 2.7 ilustra a arquitetura de uma instalação com os serviços mínimos necessários do Openstack, assim como as relações que existem entre eles.

Como plataforma IaaS, o OpenStack contém as VMs no centro de toda esta lógica, provisionadas pelo serviço Nova. As VMs são suportadas por outros serviços como a conexão à rede pelo Neutron; imagens OS pelo Glance; armazenamento pelo Swift e pelo Cinder. O serviço Keystone é responsável pela autenticação de todo o software OpenStack enquanto que o Horizon providencia uma interface de gestão de todos os outros serviços baseada em web.

Alguns dos serviços providenciado pelo openstack mais relevantes a este caso de estudo são:

- **Serviço de computação**

  OpenStack Compute (Nova) foi projetado para gerir recursos computacionais e providenciar acesso a estes via GUI (Dashboard), ferramentas na linha de comando ou pelo conjunto de APIs.

  Este serviço, Nova, trabalha com as tecnologias de virtualização mais conhecidas, como o KVM (default) [40], o VMware [41], o Xen [42] ou o Hyper-V [43] assim como as tecnologias de Linux Container, como o LXC [44].

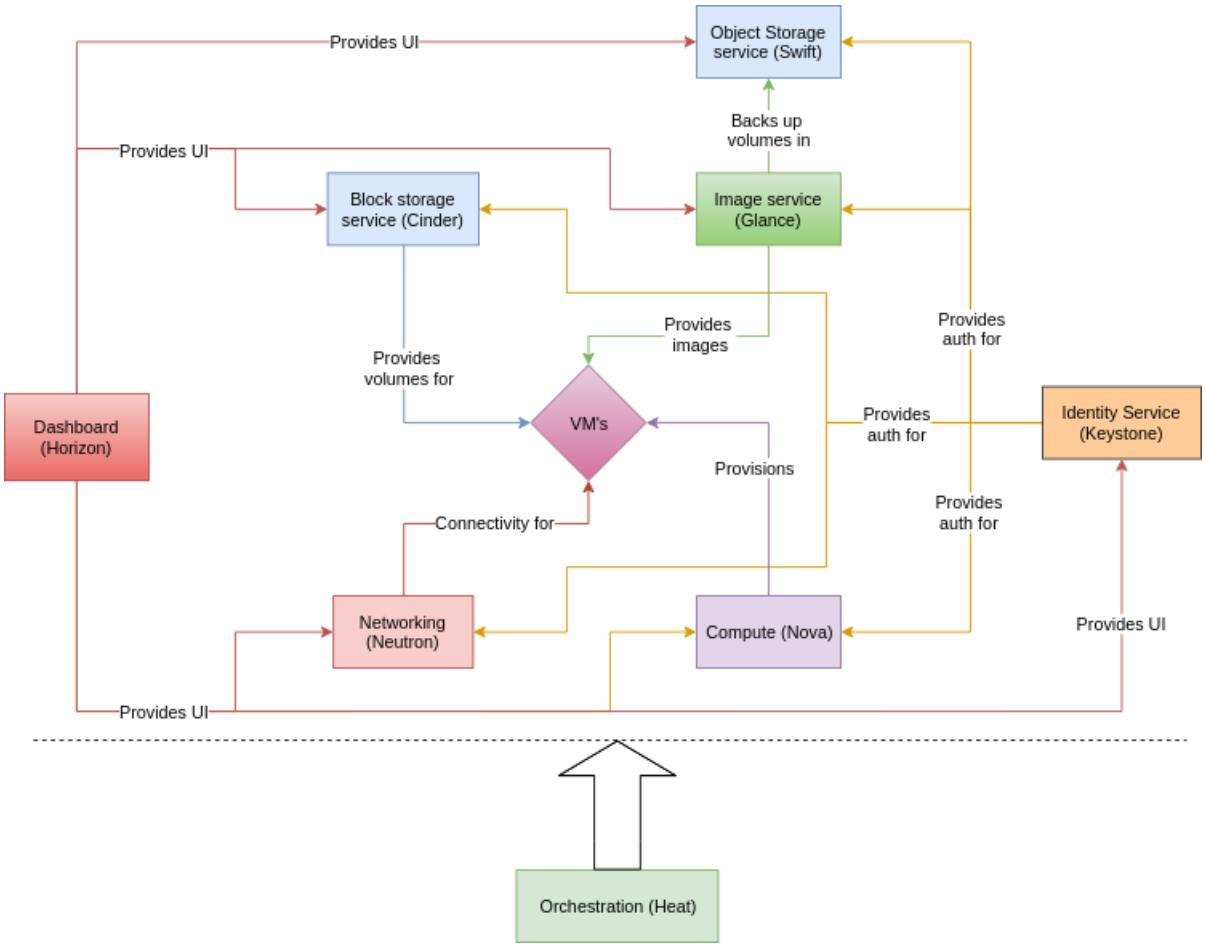



Figura 2.7: Diagrama de relações entre os serviços essenciais do OpenStack

A Nova pode ser considerada a parte principal de um sistema que segue o modelo IaaS, no qual os utilizadores da nuvem têm acesso a VMs hospedadas por nós que executem o serviço Nova. Na plataforma OpenStack, podem ser adicionados e integrados nós computacionais aos nós já existentes, tornando o conjunto de recursos do hardware escalonável horizontalmente.

- **Serviço de armazenamento**

  Para além da tecnologia de armazenamento tradicional (pertence aos recursos de computação geridos pelo Nova), o OpenStack suporta mais dois tipos de armazenamento, os quais são o Armazenamento de objetos e armazenamento de blocos.

  O armazenamento de objetos (Swift) é um sistema de armazenamento redundante escalável, no qual os objetos e os arquivos são armazenados, replicados e distribuídos por vários servidores no cluster.

  O armazenamento em bloco (cinder) gere (cria / anexa / desanexa) o armazenamento em bloco virtualizado e fornece aos utilizadores do OpenStack acesso a este armazenamento. O armazenamento em bloco é totalmente integrado nos serviços de computação (Nova)

e Dashboard (Horizon) via APIs, permitindo aos utilizadores consumir esses recursos de armazenamento, mesmo sem qualquer conhecimento da tecnologia dos dispositivos de armazenamento subjacentes.

- **Serviço de rede**

  O serviço de rede (Neutron) fornece uma abstração virtual da infraestrutura de rede (rede, sub-redes, portas, routers) e serviços (firewall, load balancer, rede virtual privada) em cluster baseado no OpenStack. O papel deste serviço é fornecer esta abstração às VMs (criadas e geridas pela Nova).

- **Serviço *Dashboard***

  O *dashboard* do OpenStack (Horizon) permite aos utilizadores aceder aos diferentes serviços do OpenStack através de uma interface gráfica Web.

  É possível observar as diferentes VMs instanciadas, as redes virtuais a que pertencem e os *routers* virtuais que conectam essas redes. Além do *dashboard*, é possível também interagir e executar tarefas através do conjunto de APIs nativas do OpenStack ou da compatibilidade com o EC2 APIs.

- **Serviços Partilhados**

  O OpenStack possui vários outros serviços que são usados pelos projetos principais acima, o que torna mais fácil implementar e operar a nuvem a ser usada. Estes serviços, como o de identidade, o de gestão de imagens e o da interface web, integram nos componentes do OpenStack uns com os outros, bem como sistemas externos para fornecer uma experiência unificada para utilizadores ao interagirem com os diferentes recursos da nuvem.

  **Serviço de identidade**. O serviço OpenStack Identity (Keystone) fornece um mecanismo central de autenticação e autorização para outros serviços do OpenStack. A Keystone também fornece endpoints para todos os serviços do OpenStack.

  **Serviço de imagem**. O serviço OpenStack Image (Glance) permite criar, armazenar e recuperar imagens de disco para as VMs, as quais são usadas pelo serviço de computação durante o aprovisionamento das VMs.

  **Serviço de telemetria**. O serviço OpenStack Telemetry (Ceilometer) visa medir e rastrear os serviços usados pelos utilizadores do OpenStack e fornece o faturamento de acordo com esse uso.

  **Serviço de orquestração**. O OpenStack Orchestration (Heat) fornece a capacidade de definir e automatizar a implantação da infra-estrutura, serviços e aplicações através do uso de modelos flexíveis. Pode aumentar ou diminuir o *cluster* do OpenStack.

### 2.5.2 Apache CloudStack

O CloudStack é um projeto de alto nível da Apache Software Foundation (ASF) [45]. O projeto foi iniciado por uma empresa chamada Cloud.com [46] que lançou a primeira versão do CloudStack em maio de 2010. A maior parte do software lançado estava sob a Licença Pública Geral GNU, versão 3 (GPLv3), sendo que uma pequena parte foi mantida proprietária. Enquanto isso, a Cloud.com foi comprada pela Citrix [47] que lançou o restante software proprietário da CloudStack sob a GPLv3 e geriu o projeto por quase um ano. Em abril de 2012, a Citrix reclassificou o CloudStack sob a Licença de Software Apache 2.0 (ASLv2), submeteu-o à Apache Incubator [48] e interrompeu seu envolvimento no projeto.

O CloudStack da Apache é uma plataforma de software open-source que utiliza recursos de computação para criar nuvens públicas, privadas e híbridas segundo o modelo IaaS (infraestrutura como serviço). Foi projetado para instanciar e gerir grandes redes de VMs, como uma plataforma de computação na nuvem altamente disponível e escalável. Atualmente, o CloudStack suporta os hipervisores mais populares, incluindo o VMware, o Oracle VM, o KVM, o XenServer e o Xen.

O CloudStack oferece uma computação em nuvem pública semelhante ao Amazon EC2, mas sobre o seu próprio hardware. Providencia também a orquestração dos vários recursos virtualizados para criar um ambiente homogéneo em que é possível ter acesso aos recursos de forma simplificada.

Possibilita gerir os ambientes de Cloud Computing através de três diferentes formas, uma interface Web simples e fácil de usar, da linha de comando e ou através de uma API RESTful com todos os recursos [49].

#### 2.5.2.1 Arquitetura

O CloudStack está focado na automação e gestão de vários data centers. Como resultado, a documentação disponível dá mais relevância à arquitetura e aos recursos gerais da plataforma de nuvem (por exemplo, organização da rede de data centers de conjuntos e clusters) do que a uma descrição detalhada de funções de software individuais (camadas). O software é inteiramente escrito em Java e está organizado nas seguintes camadas (ver figura 2.8):

- **Interface**: Esta camada é composta pela User Interface (UI) e pela API externa, que permite configurar, executar pedidos, provisionar e permitir o acesso à infraestrutura de IT (tecnologia de informação).
- **Lógica de negócios**: Esta camada contém funções de gestão que fornecem o controlo do acesso e autenticação, processam o fluxo de trabalho para melhorar a alta disponibilidade do serviço e a disponibilidade e/ou aplicabilidade dos recursos necessários, como redes e armazenamento.

- **Orquestração**: Esta camada é responsável pela gestão da infraestrutura virtual. Gere máquinas virtuais, armazenamento, rede e interage com a base de dados do Cloud Center (que armazena as informações de configuração de um cluster) para gerir e instanciar os templates, imagens e snapshots. Automatiza a distribuição dos recursos de computação, rede e armazenamento para os vários controladores e fornece à camada de negócios as informações necessárias para a definição de políticas de *load balancing*, segurança de dados e conformidades. Esta camada também possui um gestor de tarefas assíncrono para gerir o planeamento e a priorização de tarefas solicitadas.
- **Controladores**: Os controladores permitem a comunicação com os hipervisores subjacentes, bem como os recursos de rede e armazenamento. O CloudStack usa por padrão um router virtual como fornecedor de serviços de rede. Este router virtual implementa recursos como acesso remoto por VPN, proteção de firewall, NAT de fonte e estático; load balancing, encaminhamento de porta, DHCP, DNS e dados do utilizador. Os recursos de armazenamento são divididos em dois tipos:
  * Armazenamento primário - é compartilhado entre todos os hosts de um cluster e é usado para hospedar as instâncias de VMs Guest. Suporta SAN (por exemplo, iSCSI), NAS (por exemplo, NFS) ou Direct Attach Storage (DAS), por exemplo, o sistema Linux EXTended (EXT).
  * Armazenamento secundário - é usado para armazenar templates, imagens ISO e snapshots a serem instanciados na infraestrutura de TI. Cada centro de dados tem pelo menos um servidor de armazenamento secundário que é partilhado por todos os pods. Ao contrário do armazenamento primário usa apenas o NFS, como os servidores Swift, armazenamento NFS ou Linux NFS do armazenamento de objetos OpenStack.

#### 2.5.2.2 Deployment do software

A instalação do software CloudStack consiste em duas partes, o servidor de gestão e a infraestrutura da nuvem gerida.

O servidor de gestão é responsável pela orquestração, por alocar os recursos da nuvem instanciada, atribui armazenamento e endereços IP às instâncias da máquina virtual e geralmente é executado numa máquina dedicada ou como uma máquina virtual.

O servidor de gerenciamento tem como objetivo:

- Providenciar a interface da Web;
- Fornece as interfaces da API para a API do CloudStack, bem como a interface do EC2;
- Gere a atribuição das VMs guest a um recurso de computação específico;
- Gere a atribuição de endereços IP públicos e privados;

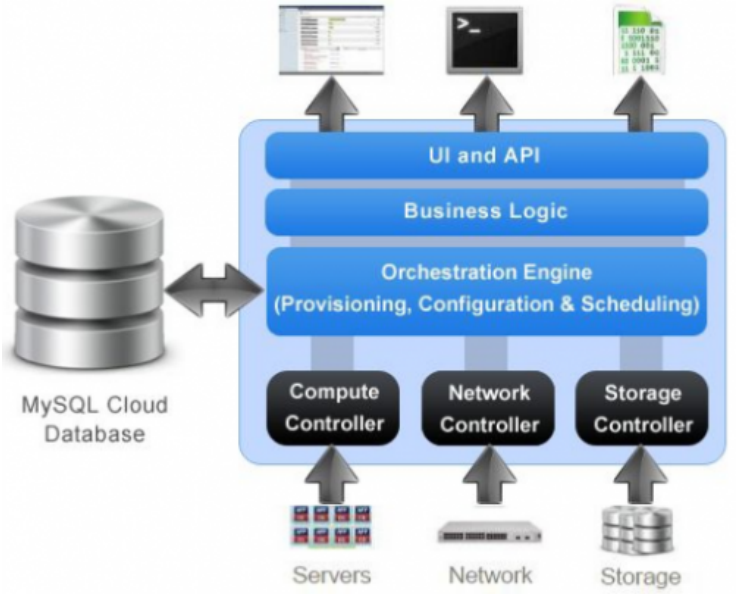



Figura 2.8: Arquitetura funcional do CloudStack [50].

- Aloca armazenamento durante o processo de instanciação das VMs;
- Gere snapshots, imagens de disco (modelos) e imagens ISO;
- Fornece um único ponto de configuração para sua nuvem;

Em relação à infraestrutura da nuvem, esta está organizada de forma hierárquica como podemos verificar na figura 2.9.

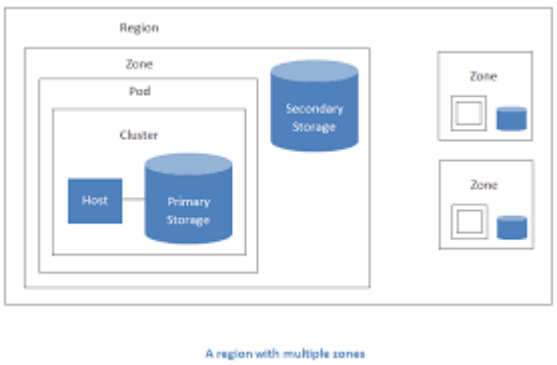



Figura 2.9: Infraestrutura do CloudStack [51].

A infraestrutura é constituída pelos seguintes componentes:

- Região: uma coleção de uma ou mais zonas geograficamente próximas geridas por um ou mais servidores de gestão.
- Zona: normalmente, uma zona é equivalente a um único datacenter. Uma zona consiste em um ou mais pods e armazenamento secundário.

- Pod: um pod geralmente é um rack de hardware que inclui um switch de camada 2 e um ou mais clusters.
- Cluster: um cluster consiste em um ou mais hosts com um tipo comum de hipervisor e armazenamento primário.
- Host: Um único nó de computação dentro de um cluster. Os hosts são onde os serviços de nuvem "reais"são executados em forma de máquinas virtuais guest.
- O armazenamento primário está associado a um cluster e armazena os volumes de disco de todas as VMs em execução nos hosts desse cluster.
- O armazenamento secundário está associado a uma zona e armazena modelos, imagens ISO e instantâneos de volume de disco.

### 2.5.3 Comparação

Foi feita uma breve comparação entre estas duas tecnologias, que pode ser observada na tabela 2.1.

## 2.6 Tecnologias para configuração de uma aplicação num servidor

### 2.6.1 Node.js

Node.js, também designado só por Node, é um JavaScript *runtime* baseado num modelo orientado a eventos assíncronos. Foi projetado para o desenvolvimento de aplicações de rede escaláveis. É implementado sobre o motor JavaScript do Google Chrome, mais conhecido como V8 Engine. [52]

Esta tecnologia surgiu quando os responsáveis pelo desenvolvimento do JavaScript estenderam este de algo que apenas corre no *brownser* para algo que pode correr numa máquina como uma aplicação independente. Com o Node o JavaScript agora tem a capacidade de fazer coisas que outras linguagens, como o python, têm.

Tanto o V8 e o Node focam-se na performance e no baixo consumo de memória, mas, enquanto o V8 suporta principalmente JavaScript no navegador, o Node tem como objetivo suportar processos do servidor.

Tanto o JavaScript dos navegadores quanto o Node.js são executados no mecanismo de tempo de execução do V8 JavaScript. Esse mecanismo usa o código JavaScript e o converte num código de máquina mais rápido. O código da máquina é um código de baixo nível que o computador pode executar sem precisar interpretá-lo primeiro. [53]

O Node.js é um tempo de execução de JavaScript criado no mecanismo JavaScript V8 do Chrome. O Node.js usa um modelo de I/O sem bloqueio orientado a eventos que o torna leve

| **Suporte** | **CloudStack** | **OpenStack** |
|---|---|---|
| **Camada de abstração dos recursos** | | |
| Computação | Libcloud | Nova |
| Armazenamento | Interno | Swift/Cinder |
| Volume | Interno | Nova-Volume |
| Rede | Interno | Neutron/Nova |
| **Camada do serviço principal** | | |
| Serviço de identidade | IAM API | Keystone |
| Agendamento | Interno | Nova-scheduler |
| Repositório de imagens | Interno | Glance |
| Faturamento | CloudStack Usage | Ceilometer |
| Logging | Interno | Interno |
| **camada de suporte** | | |
| Mensagens | internal/RabbitMQ | RabbitMQ |
| Base de dados | MySQL | MySQL/Galera/ MariaDB/ MongoDB |
| Serviço de transferência | Interno | Nova Object store/-cinder |
| **Ferramentas de gestão** | | |
| Ferramentas CLI | cloudmonkey | OpenStack (CLI) |
| APIs | Public cloud and Plugins | Public cloud and Plugins |
| Dashboard | Admin UI | Horizon(Admin UI) |
| Orquestrador | Cloudstack Cookbook | Heat |
| **Serviços com valor acrescentado** | | |
| Zonas de disponibilidade | Interno | Interno |
| Alta disponibilidade | Externo | Externo |
| Suporte híbrido | Amazon EC2 | HP Helion/ Amazon EC2/IBM |
| Live migration | Interno | Interno |

Tabela 2.1: Comparação entre OpenStack e CloudStack

e eficiente. O ecossistema de pacotes do Node.js, npm, é o maior ecossistema de bibliotecas de código aberto do mundo.

O modelo I/O refere-se a input/output. Pode ser qualquer coisa, desde ler / gravar arquivos locais até fazer uma solicitação HTTP para uma API. [54]

### 2.6.2 NGINX

O NGINX é um servidor HTTP e proxy reverso de alto desempenho, gratuito e de código aberto, bem como um servidor proxy IMAP / POP3. O NGINX é conhecido por seu alto

desempenho, estabilidade, rico conjunto de recursos, configuração simples e baixo consumo de recursos.

O NGINX é um dos poucos servidores escritos para resolver o problema do C10K. Ao contrário dos servidores tradicionais, o NGINX não depende de threads para lidar com solicitações. Em vez disso, ele usa uma arquitetura orientada a eventos (assíncrona) muito mais escalável. Essa arquitetura usa pequenas, mas mais importantes, quantidades previsíveis de memória sob carga. Mesmo que não seja esperado lidar com milhares de solicitações simultâneas, é possível beneficiar do alto desempenho e do baixo espaço ocupado pela memória do NGINX.

O NGINX escala em todas as direções: desde o menor VPS até grandes grupos de servidores. [55]

## 2.7 Estudo do mercado

Nesta secção vai ser feita uma análise às soluções que existem no mercado semelhantes à solução que esta dissertação tem como objetivo desenvolver. A nível nacional o que existe apenas é o "eu vi" da TVI e o "eu repórter CM" da CM TV, baseado em conteúdo de natureza informativa em tempo não real. Ambos funcionam da mesma forma, através de uma plataforma, os espetadores podem submeter à apreciação, tanto da TVI como da CMTV , e carregar um vídeo ou uma foto de algum acontecimento real. O ficheiro (foto ou vídeo) carregado é armazenado na infraestrutura da TVI ou da CMTV e posteriormente o conteúdo poderá ser exibido num espaço informativo, quando e na medida em que são considerados que os mesmos têm qualidade e relevância noticiosa, de acordo com os critérios.

O panorama internacional é bastante parecido ao panorama nacional, as estações de notícias mais populares como BBC News ou CNN oferecem um serviço em tempo não real idêntico ao serviço oferecido pela TVI ou a CMTV. O utilizador pode contribuir através de uma plataforma com fotos ou vídeos de eventos ocorridos, que irão ser mais tarde analisados por alguém da empresa responsável por essa tarefa. Na tabela 2.2 podemos comparar algumas características entre a solução oferecida por estas identidades e a solução estudada nesta dissertação.

| | Eu vi (TVI) | Eu repórter CM (CMTV) | Solução estudada |
|---|---|---|---|
| Real-Time | Não | Não | Sim |
| Sujeito a notícias falsas | Sim | Sim | Não |
| Possibilidade de edição de conteúdo | Não | Não | Sim |
| Potencial para substituir um repórter | Não | Não | Sim |
| Potencial para substituir as tradicionais notícias | Não | Não | Sim |
| Rápido a chegar ao público | Não | Não | Sim |
| Necessidade de uma pessoa a rever o conteúdo | Sim | Sim | Sim |

Tabela 2.2: Estudo de algumas soluções existentes no mercado por parte de *broadcasters*.

## 2.8 Conclusões

Neste capítulo foi revisto o estado da arte sobre virtualização, fazendo uma análise sobre 3 tipos de virtualização, máquinas virtuais, containers e unikernels. Foi feita uma comparação entre estas tecnologias e foram explicados alguns dos seus componentes mais importantes.

De seguida, foram analisados os vários níveis de computação existentes, entre os quais na cloud, no edge e ainda a fog. Foi explicada importância de cada um nos dias de hoje e ainda foi abordada a questão da segurança do hardware usado numa computação mais próximo do edge.

Quanto aos gestores de redes de nuvens, foi explicada a sua importância na gestão de uma cloud, tendo sido de seguida exploradas com algum detalhe algumas das mais importantes plataformas nesta área. Foi também feita uma comparação em termmos de propriedades e características entre as diferentes soluções.

Foi decidido utilizar VMs por se encontrarem mais de acordo com os requerimentos do caso de estudo em causa, pelo facto de serem completamente isoladas da máquina host, o que permite em causa de erro encerrar a VM instanciar uma nova sem qualquer complicação. Quanto ao gestor de nuvens, foi optado o OpenStack, apesar da maior dificuldade de configuração e utilização comparativamente ao openNebula, é um projeto em desenvolvimento muito ativo e em grande crescimento, para além de muito completo a nível das suas potencialidades, permitindo assim uma melhor adaptação a um cenário mais genérico.

# Capítulo 3

# Protótipo da *framework* de gestão de notícias regionais

Neste capítulo é feita uma introdução ao conceito da *framework* e, em seguida, é apresentada a arquitetura e as funcionalidades da mesma.

## 3.1 Introdução

A cadeia de valor da televisão pode ser entendida por compreender quatro fases sequenciais: produção; programação; distribuição; entrega e consumo. Isto é, um conteúdo de televisão é produzido, armazenado, e enviado para os *broadcasters* que distribuem esse conteúdo para o consumidor final. O problema identifica-se no respetivo *workflow*.

O registo do aumento da hiperlocalização de notícias, isto é, a existência de vários órgãos de comunicação regional para garantir uma maior cobertura e, por conseguinte, uma maior rapidez na captação da notícia e entrega ao consumidor final, comporta algumas debilidades na comunicação entre estas entidades, demonstrando ser pouco eficiente.

Como tal, aponta-se como necessário a criação de uma solução capaz de apresentar benefícios, quer seja de gestão do *workflow*, quer seja em termos económicos, e desta forma, garantir a sustentabilidade dos meios de comunicação social.

A *framework* constitui uma solução de *crowd journalism*, que consiste num sistema integrado baseado na *cloud* para as fases de captura, processamento da *cloud*, controlo e gestão dos conteúdos recebidos e pós-produção no estúdio, o qual dá a oportunidade do público geral, juntamente com jornalistas de profissão, contribuir com conteúdo televisivo para posterior transmissão por parte de *broadcasters*, através da captação de vídeo com um *smartphone*.

A finalidade é promover a migração dos sistemas atuais de captura, produção, gestão, distribuição de conteúdos para um paradigma assente na *cloud*. Desta forma, permite a criação de

fluxos de trabalho flexíveis, sistemas abertos e novos modelos de negócio que enderecem as necessidades de clientes atuais e as especificidades de futuros clientes, sendo possível atingir, também, a promoção de uma deslocalização de processos e uma escalabilidade natural das soluções encontradas. Para além de que revela-se, também, capaz de melhorar a qualidade do serviço e da experiência do consumidor final dos conteúdos, através do desenho e conceção de novos ambientes imersivos ou com uma melhoria significativa de qualidade, que aumentem o número de visualizações dos conteúdos produzidos. Por este meio, induzem à produção de novos conteúdos, re-alimentando, positivamente, a cadeia de valor e viabilizando a criação de novos empregos, bem como a identificação de novas oportunidades de negócio.

## 3.2 Arquitetura

A arquitetura da solução (ver figura 3.1) é composta por vários módulos que integram em dois tipos de VMs com funções distintas e específicas:

- $VM_A$:

  Responsável por indicar através de uma *web page* o endereço IP da $VM_B$ que irá executar a transcodificação da *stream* e após ser iniciado este processo, esta $VM_A$ começa a receber a *stream* transcodificada.

  Da mesma forma que recebe esta poderá estar a receber N *streams* provenientes de diferentes fontes. É procedido à comutação do conteúdo pretendido e este é enviado para para alguma CDN, broadcaster e/ou para armazenamento.

- $VM_B$:

  Responsável por receber, transcodificar e enviar para a $VM_A$, em tempo real, a *stream* proveniente do *smartphone*.

Como todos os nós têm um grande nível de abstração e a comunicação entre eles é idêntica existe uma grande escalabilidade, o que permite que as VMs sejam instanciadas mais do que uma vez e que a introdução de novos nós com outras funcionalidades seja simples. Nesta arquitetura, todos as componentes pertencem à mesma rede privada.

### 3.2.1 $VM_A$

Esta VM, responsável por gerir os pedidos de ligação por parte dos *smartphones* e também por receber as várias *streams* já transcodificadas, é configurada com um servidor HTTP NGINX a redirecionar os vários pedidos recebidos, como um servidor *in-front* ou *reverse proxy*, e um servidor Node.js responsável por processar o conteúdo dinâmico dos pedidos redirecionados para este.

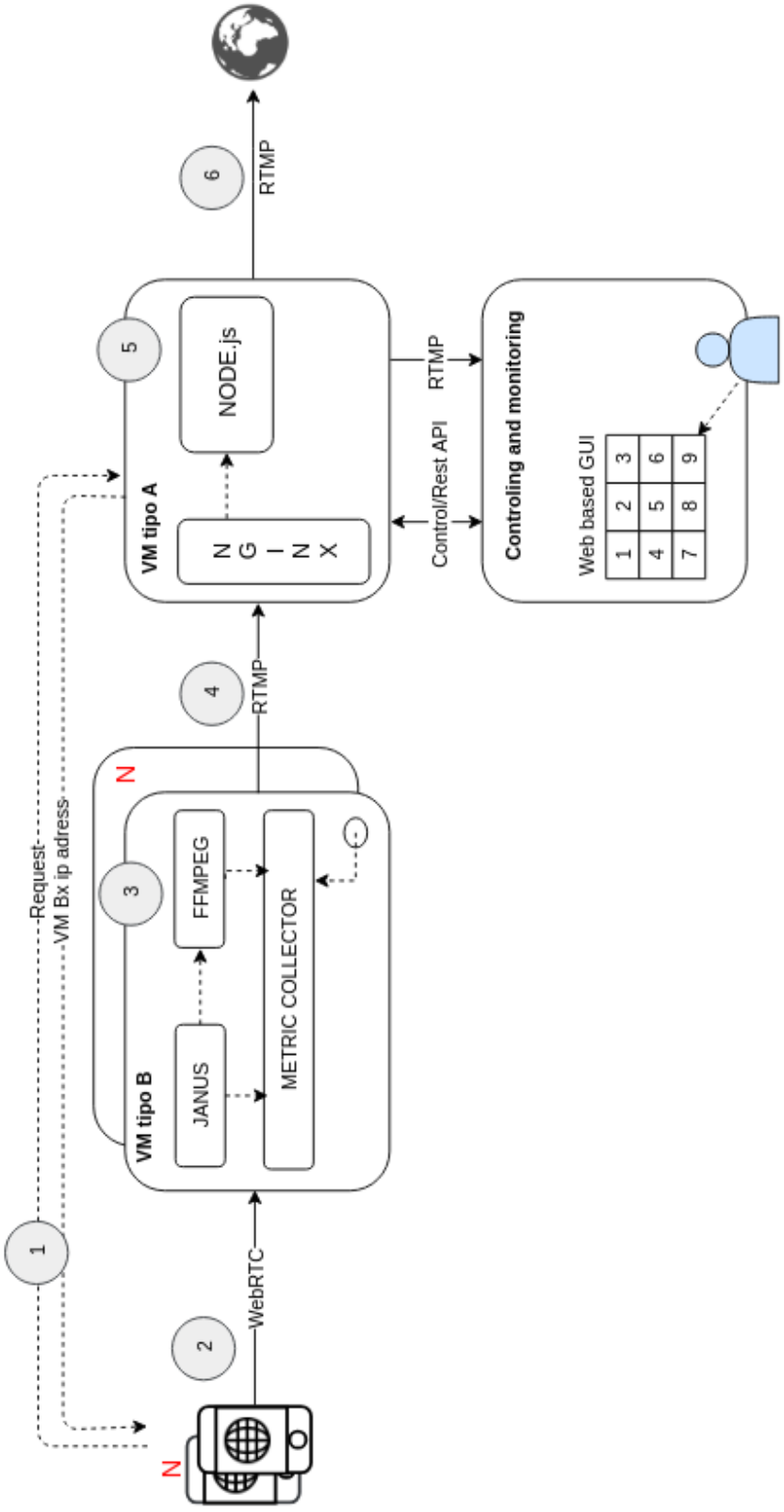



Figura 3.1: Arquitetura do protótipo

Um servidor NGINX funciona como um servidor web que após receber pedidos HTTP, processa-os e redireciona o pedido para o local certo. Funciona segundo um modelo orientado a eventos assíncronos, assim como o Node.js, sendo que a principal vantagem de usar este servidor como entrada inicial dos pedidos é a rapidez com que os processa, mesmo que sejam feitos milhares de pedidos simultaneamente.

Já o Node.JS é um ambiente de programação de javascript que permite o desenvolvimento de apps para servidores, sendo possível comunicar com bases de dados e gerar páginas dinâmicas, como por exemplo o PHP, mas tendo a diferença que inclui também um servidor HTTP capaz de gerir pedidos de forma independente. No entanto, este servidor mostrou-se ineficaz no que o atual tráfego requisita.

Não é obrigatoriamente necessário implementar um servidor NGINX juntamente com Node.js, mas, analisando alguns resultados, usar uma arquitetura que integre um servidor NGINX *in-front* permite melhorar a *performance* e eficiência, especialmente quando existe um grande tráfego de pedidos. Para além disso, é uma solução escalável, uma vez que é possível ter várias instâncias Node.js para aumentar a capacidade de processamento. Assim, o servidor NGINX é capaz de receber os pedidos e redirecioná-los para as diferentes instâncias. Desta forma torna-se mais fácil operar com maior tráfego. [56]

Concluindo, uma vez que a *framework* necessita de garantir eficiência e pouca latência no que toca a operar sobre uma grande quantidade de tráfego, aplicar esta arquitetura na $VM_A$ torna-se bastante importante para alcançar os objetivos.

### 3.2.2 $VM_B$

Em relação a esta VM, a qual é instanciada por cada pedido proveniente dos *smartphones*, tem como papel receber a *live stream* e realizar a transcodificação e envio da mesma para a $VM_A$, em tempo real. Para tal é usado o FFMPEG, uma *framework* de multimédia, capaz de descodificar, codificar, transcodificar, multiplexar, filtrar e reproduzir conteúdos de multimédia.

É também importante garantir a compatibilidade com os vários *codecs* usados hoje em dia nos *smartphones* para a codificação dos dados (ver tabela 3.1) , o que é conseguido ao usar o FFMPEG. Para além disso, a resolução das várias *streams* pode ser variável, portanto, é necessário também transcodificar para uma resolução única.

Para realizar esta operação não é necessário um sistema operativo muito complexo e pesado, sendo que foi decidido usar o Alpine, um sistema operativo derivado de Linux que é capaz de responder às necessidades impostas pelo propósito da VM.

| | **Codecs** |
|---|---|
| Áudio | AAC LC ; AAC+ ; AMR-NB ; AMR-WB ; FLAC ; PCM/Wave |
| Vídeo | H.263 ; H.264 AVC ; H.265 HEVC ; VP8 |

Tabela 3.1: *Codecs* de vídeo e áudio suportados por *smartphones* para *encoding*

Outro módulo importante a ser implementado para que seja feita a monitorização de todas as VMs instanciadas, é uma coletor de métricas que recolha dados desde o consumo de CPU ou memória RAM usada. Desta forma, é possível agregar a informação das VMs para estudo e análise.

## 3.3 *WorkFlow*

Como foi possível verificar a enumeração de etapas, na figura 3.1, este sistema segue um respetivo worflow:

- 1: Primeiramente, o *smartphone* através da aplicação web faz o pedido de ligação à VM A, sendo que esta responde com o endereço IP da máquina do VM do tipo B criada para o respetivo aparelho, através do corpo HTML da página.
- 2: Após o *smartphone* ter recebido o endereço da VM do tipo B, começa a transmitir o conteúdo *live*, via WebRTC.
- 3: O módulo JANUS, responsável por receber *stream* através de WebRTC, redireciona esta, via RTP, para o módulo FFMPEG que inicia o processo de transcodificação em tempo real.
- 4: A *stream* já transcodificada é transmitida, via RTMP, em tempo real, para VM A.
- 5: Na VM A são recebidas N *streams* com o conteúdo *live*, sendo que estas são apresentadas através de uma interface. Um operador irá estar a analisar estas imagens e vai ser comutada a pretendida entre estas.
- 6: Por fim, a *stream* selecionada é enviada para os *broadcasters*.

Na figura 3.2 é apresentado um cenário de utilização, no qual são criadas uma $VM_A$ e 2 $VM_B$ e existe um pedido de apenas um *smartphone* para iniciar *streaming live*.

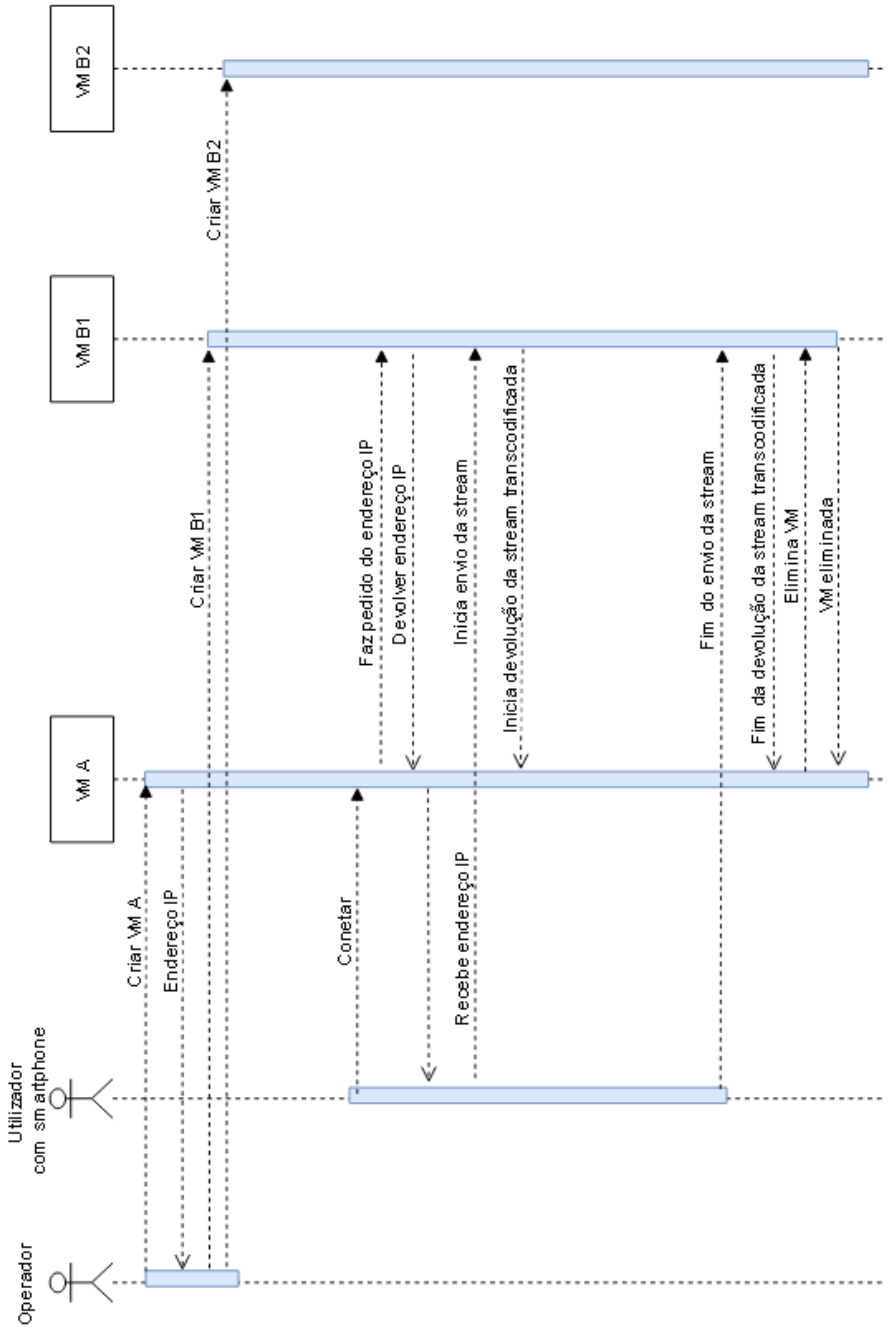



Figura 3.2: Exemplo de aplicação para um smartphone

## 3.4 Limitações

Por agora, a instanciação destes módulos é feita no ambiente da empresa, em máquinas próprias. E, apesar deste protótipo ter sido desenvolvido tendo em vista os conceitos de *cloud computing*, a instanciação dos módulos é ainda feita de forma pouco coesa e automatizada.

A instanciação das VMs ao ser realizada através de procedimentos manuais não se está a retirar todo o proveito da virtualização, por isso é importante estudar uma solução que permita a automatização da crição das VMs e ainda que seja possível a sua monitorização. No capítulo 4 irá ser analisada essa solução.

## 3.5 Aplicação ao projeto

Conforme referido no capítulo 1, a solução proposta nesta dissertação está a ser desenvolvida para ser implementada na infraestrutura providenciada pelo 5G City. No entanto, encontra-se ainda em fase de desenvolvimento nas instalações da MOG, e faz uso das máquinas da empresa.

O cenário utilizado como referência para a implementação deste trabalho consiste em uma $VM_A$, inicialmente por duas $VM_B$ e dois *smartphones* a fornecer uma *stream* de entrada. Sendo assim testados todos os módulos implementados até agora. Este caso procura simular a cobertura de um evento onde existem dois *smartphones* a transmitir as *streams* por WebRTC.

Inicialmente a máquina $VM_A$recebe o pedido de um *smartphone*, ao qual responde com o endereço IP da $VM_B$ que irá receber a *stream live*, executar a transcodificação e enviar, por RTMP, para a máquina $VM_A$. Nesta VM irão ser recebidas ambas as *streams* e é gerada uma pré-visualização dos sinais para que um realizador/operador possa fazer a comutação entre os sinais dos dois *smartphones*.

# Capítulo 4

# Desenvolvimentos efetuados

No respetivo capítulo será apresentado o problema identificado no protótipo, assim como uma breve descrição da solução encontrada e as tecnologias que a sustentam.

## 4.1 Definição Do Problema e Requisitos

De acordo com o que foi dito anteriormente, os *smartphones* que se encontram a captar imagem e som transmitem as *streams* por WebRTC, através de uma web app, para uma dada $Vm_B$. Esta VM iá proceder à transcodificação para o *codec* de vídeo H.264 e *codec* de áudio AAC, e, assim que começa a ser transcodificada, começa também a ser enviada para a $VM_A$. Esta VM estará a receber N *streams* de N $VMs_B$, as quais são apresentadas *live* numa interface GUI e analisadas por um operador que é responsável por decidir qual a *stream* a comutar. A *stream* escolhida será transmitida, como output do sistema, para entrega e consumo.

O problema que se procura resolver é a falta de escalabilidade e automatização por parte das VMs e dos respetivos módulos, os quais tem de ser neste momento instanciados através de procedimentos manuais, não tirando proveito de todas as vantagens da virtualização.

## 4.2 Solução proposta

A *framework* apresentada no capítulo anterior tem como objetivo final promover a migração dos sistemas atuais de captura, produção, gestão, distribuição de conteúdos para um paradigma assente na *cloud*. Desta forma, permitir a criação de fluxos de trabalho flexíveis, sendo possível atingir, também, a promoção de uma deslocalização de processos e uma escalabilidade natural das soluções encontradas. Para além de que revela-se, também, capaz de melhorar a qualidade do serviço e da experiência do consumidor final dos conteúdo. Por este meio, induzem à produção de novos conteúdos, re-alimentando, positivamente, a

cadeia de valor e viabilizando a criação de novos empregos, bem como a identificação de novas oportunidades de negócio.

Para resolver o problema acima mencionado na presente dissertação irá ser estudada a forma como a tecnologia OpenStack, anteriormente analisado juntamente com outras tecnologias, pode vir a colmatar as limitações identificadas para trazer maiores vantagens e regalias de toda a virtualização da cadeia de produção de conteúdo televisivo.

O objetivo desta dissertação é utilizar o OpenStack para criar e gerir uma *cloud* privada onde irá ser possível instanciar o sistema de VMs anteriormente criadas, e perceber de que forma é possível automatizar todo o *workflow*. A utilização desta *cloud* vai potenciar a escalabilidade da aplicação, de forma a lidar com tarefas como a transcodificação dos sinais em tempo real e a sua comutação, tarefas tradicionalmente feitas em hardware dedicado.

### 4.2.1 Use case do 5G City

Ao conseguir criar uma *cloud* num ambiente OpenStack de forma automatizada, através da virtualização dos recursos disponibilizados na MOG Technologies, é possível estudar de que forma se poderá aplicar à infraestrutura presente nas cidades incluídas no projeto 5G City.

Um use case do projeto é então usar a plataforma e virtualização e integrar a solução de *crowd journalism*, alvo de análise no capítulo anterior.

### 4.2.2 Infraestrutura

A infraestrutura do projeto 5G City ( ver figura 4.1), a qual é composta por *microcontrollers*, ARM SBCs (*Single Board Computer*), x86 SBCs e x86 *Servers*.

Este hardware está organizado por:

- ***Small City cells***:
  Constituídas por ARM SBCs, x86 SBCs e *microcontrollers*.
- ***Mobile edge Computing (MEC) Nodes***:
  Constituídos por ARM SBCs, x86 SBCs e x86 servers.
- ***Metro Datacenter (DC)***:
  Constituído por x86 *servers*.

O poder de computação é mais elevado no Metro DC, no entanto, este representa uma maior latência visto que se encontra mais afastado dos *end devices*. Já as *smart city cells* e os MEC *nodes* não apresentam tanto poder de computação mas representam menor latência, sendo portanto importante garantir e colocar o processamento nestas unidades de computação.

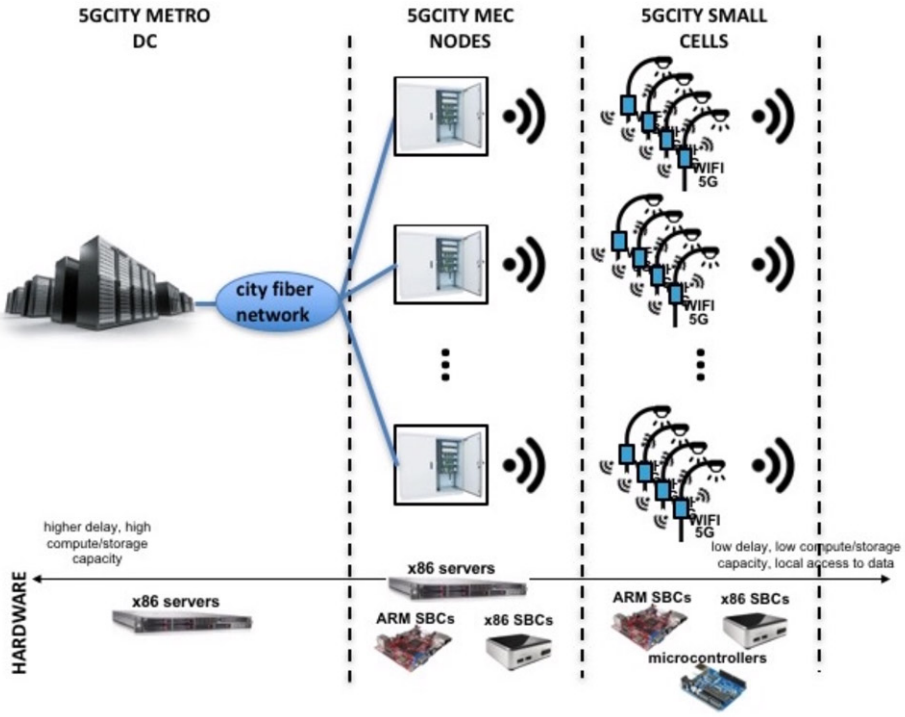



Figura 4.1: Infraestrutura. [57]

A arquitetura de alto nível pode ser observada na figura 4.2. Os serviços irão ser instanciados sobre esta estrutura, onde cada serviço é representado por uma *slice*, que consiste numa dada percentagem de poder de computação da infraestrutura disponível, e para fazer uso da mesma é necessário comunicar com a orquestração do 5G City através de uma API própria.

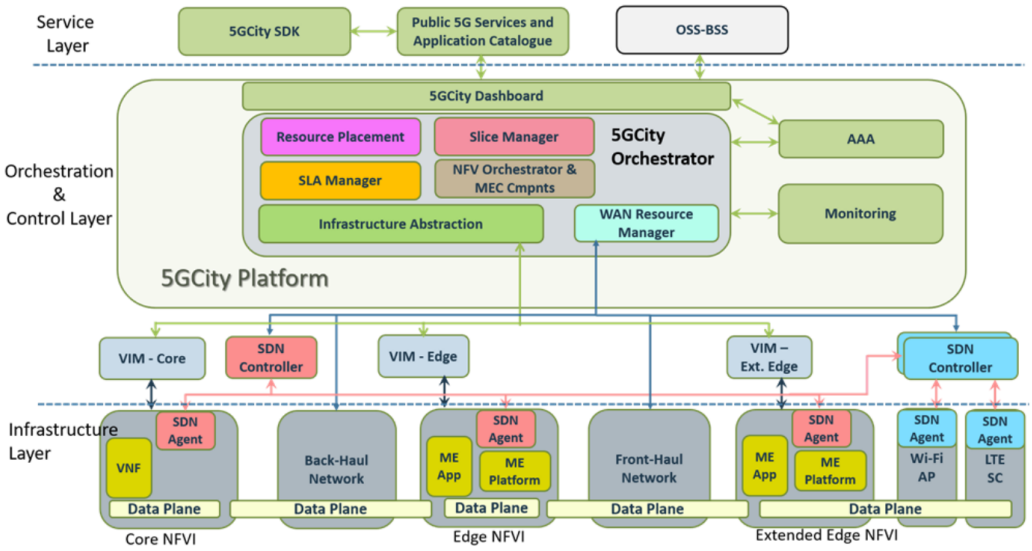



Figura 4.2: Arquitetura de alto nível do 5G City. [57]

### 4.2.3 Funcionalidades

A grande vantagem de integrar o sistema numa arquitetura assente no *cloud* OpenStack, são os serviços que esta tecnologia proporciona para criar um ecossistema totalmente escalável, virtualizado e automatizado, através da API fornecida.

Os modelos das VMs que são propostas instanciar não correspondem com aquelas que irão ser instanciadas futuramente, sendo que estas são apenas *mocks*, isto é, VMs de carga para ser possível realizar alguns testes às funcionalidades do OpenStack.

### 4.2.4 Requisitos

Os requisitos podem ser consultados na tabela 4.1.

| **Requisito** | |
|---|---|
| Criação de uma rede onde irão ser instanciadas as máquinas $VM_B$ com acesso à rede externa (net1) | R001 |
| Criação de uma rede onde irão ser instanciadas as máquinas $VM_A$ com acesso à rede externa (net2) | R002 |
| Criar um modelo $VM_B$ de acordo com as funcionalidades | R003 |
| Criar um modelo $VM_A$ de acordo com as funcionalidade | R004 |
| Instanciar 2 $VM_B$ (VM_b1 e VM_b2) com *interface* à rede net2 | R005 |
| Instanciar 1 $VM_A$ (VM_a) | R007 |
| $VM_B$ terá de suportar os *codecs* supra mencionados | R008 |
| Automatizar o *deployment* da solução | R009 |
| Gestão automatizada da *cloud* | R010 |

Tabela 4.1: Requisitos.

Na figura é representado um modelo da arquitetura que se pretende instanciar 4.3.

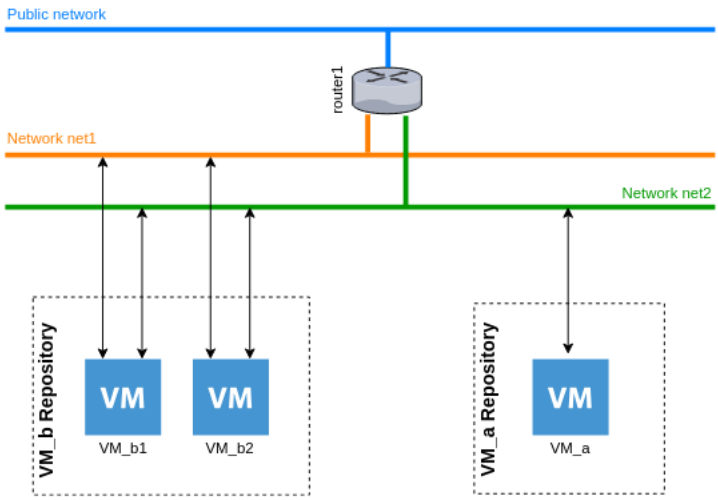



Figura 4.3: Arquitetura projetada.

## 4.3 *Cloud* privada

Como foi referido acima o objetivo é criar um cenário de uma *cloud* privada que contenha algumas VMs.

Para tal, foi procedido à instalação e configuração da tecnologia OpenStack e posterior instanciação da *cloud*.

### 4.3.1 Instalação OpenStack

A arquitetura de *deployment* do OpenStack pode variar bastante de caso para caso, visto que o OpenStack assim o permite, sendo bastante escalável horizontalmente. A arquitetura representada na figura 4.4, é constituída por três tipos de nós fundamentais para uma instalação mínima:

- **Nó de controlo**

  Responsável por:

  * Executar o serviço de identidade (Keystone);
  * Serviço de imagem (Glance);
  * Serviço de *dashboard* (Horizon);
  * Gestão do nó de computação;
  * Serviços de suporte, assim como base de dados SQL, serviço de mensagens e NTP;

  Opcionalmente, este nó pode ser responsável também pela totalidade ou parte dos serviços de armazenamento em blocos (Cinder) ou em objetos (Swift), serviço de monitorização e recolha de dados (Celiometer).

- **Nó de rede**

  Neste nó é executado o serviço de rede (Neutron), um serviço *standalone*, que normalmente instancia processos através de vários nós que interagem com outros processos ou serviços OpenStack. As componentes da rede OpenStack são:

  * Servidor Neutron, responsável por expor a API da rede OpenStack e as suas extensões. Para além disso, aplica o modelo de rede e mapeamento IP de cada porta. O servidor Neutron requer acesso indireto a uma base de dados, sendo que é feito através de plugins que comunicam com a base de dados usando AMQP (Advanced Message Queuing Protocol).
  * Plugin agent, é processado em cada nó de computação para gerir a configuração do switch virtual local (vswitch).
  * DHCP *agent*, fornece serviços DHCP (Dynamic Host Configuration Protocol) à rede. Requer acesso ao serviço de mensagens (message queue).

    * L3 *agent*, fornece encaminhamento L3 / NAT para haver acesso à rede externa nas VMs em redes do utilizador. Requer acesso à fila de mensagens. Opcional dependendo do plug-in.
    * Serviços do fornecedor de rede (servidores/serviços SDN, ou Software Defined Networking), providencia serviços adicionais à rede. Estes serviços comunica com outras componentes através de canis de comunicação como REST APIs.

– **Nó de computação**

  É neste nó que atua o serviço de computação Nova, onde as VMs são instanciadas, assim como o hipervisor que sustenta as VMs. Entre os hipervisores suportados estão KVM (Kernel-based Virtual Machine), QEMU (Quick EMUlator), VMware vSphere 5.1.0, Xen e Hyper-V.

  Neste nó também corre um agente de serviço da rede, o qual estabelece ligação das VMs às rede virtuais e fornece serviços de *firewall* às VMs através dos grupos de segurança, nos quais são definidas regras da rede.

É possível instanciar mais do que um nó em cada tipo, mediante os objetivos que se pretende alcançar ao usar o OpenStack.

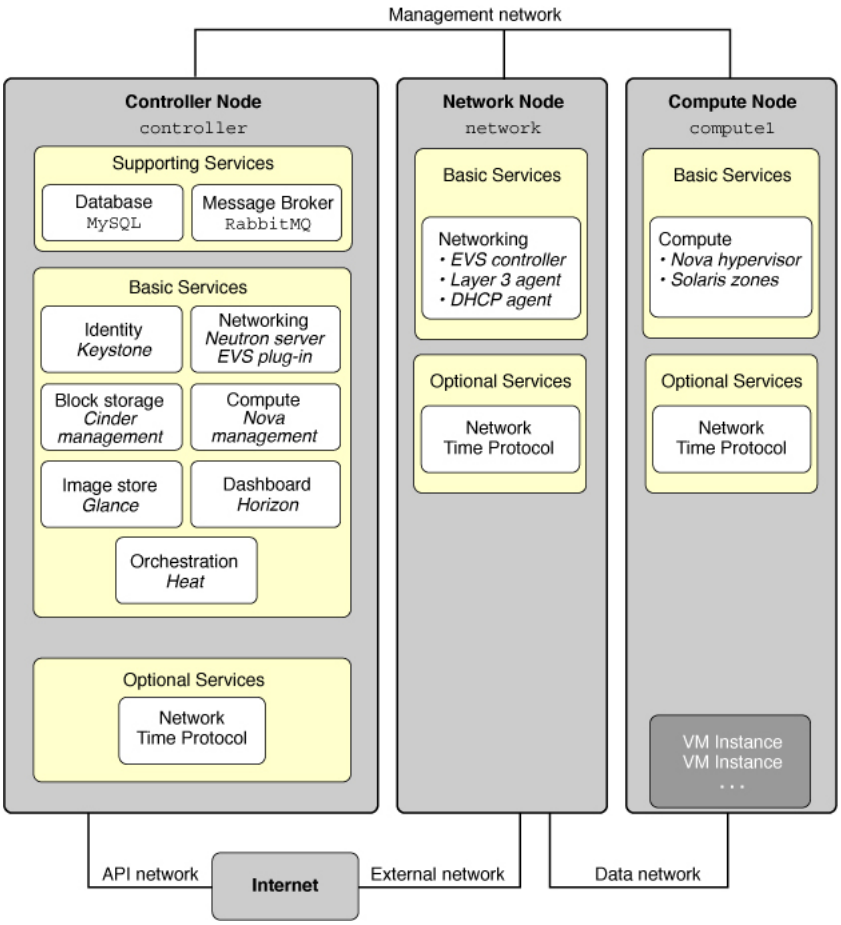



Figura 4.4: Arquitetura exemplo de *deployment* do OpenStack. [58]

Sendo que ainda é possível acrescentar, opcionalmente, dois tipos de nós:

- **Nó de armazenamento em bloco**

  Este nó opcional contém os discos que os serviços de armazenamento em bloco e sistema de arquivos partilhados provisionam para as VMs. De maneira simplificada, o tráfego de serviços entre os nós de computação e este nó usa a rede de gestão. De maneira a alcançar aumentos de desempenho e segurança é necessário implementar uma rede de armazenamento separada destinada a este propósito.

- **Nó de armazenamento em objeto**

  O nó contém os discos que o serviço Armazenamento de Objetos usa para armazenar contas, *containers* e objetos.

  Da mesma maneira que foi dito no nó de armazenamento em objeto para alcançar aumentos de desempenho e segurança é necessário implementar uma rede de armazenamento separada igualmente.

No entanto, para instalar o OpenStack foi decidido utilizar o Devstack (*Queens Realease*), uma série de *scripts* extensíveis usados para criar um ambiente OpenStack completo com base

nas versões mais recentes. Trata-se de uma instalação *All-In-One single machine* que junta os serviços pretendidos dos vários tipos de nós numa só máquina, isto é possível através dos *scripts* que instalam as dependências necessárias para que não seja necessário instalar de forma separada.

É usado de maneira interativa como um ambiente de desenvolvimento e como base para grande parte dos testes funcionais do projeto OpenStack.

Uma vez que o pretendido é apenas a criação, em pequena escala, de uma *cloud* privada que no máximo irá possuir apenas 3 VMs, o hardware usado vai ser apenas uma máquina (ver especificações na tabela 4.2), e, por isso, não se justifica uma instalação de vários nós.

| | **Máquina Host** |
|---|---|
| Processador | Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz |
| Cores | 4(8 Threads) |
| Memória RAM | 16 GB |
| Sistema Operativo | Ubuntu 18.04 LTS |

Tabela 4.2: Especificações da máquina virtual "Devstack".

Visto que a instalação do OpenStack modifica bastante as configurações do sistema, o mesmo foi instalado numa máquina virtual criada através da VirtualBox, cujas especificações encontram-se abaixo na tabela 4.3. Assim, torna-se mais simples, em caso de ocorrência de algum erro, desligar a VM e iniciar uma nova. É possível ainda criar *snapshots* da VM que seja do nosso interesse guardar para que posteriormente possam ser recuperados pela VM em causa, ou por uma outra qualquer.

Tal não seria possível se a instalação fosse realizada na máquina *host*.

Em relação ao sistema operativo a ser usado na máquina virtual a decisão passou por utilizar o Ubuntu 16.04, pelo motivo de ser o sistema operativo mais testado com esta tecnologia e por ser um OS relativamente leve.

| | **Máquina Virtual** |
|---|---|
| Software | Oracle VirtualBox |
| OS | Ubuntu 16.04 |
| Base Memory | 8 GB |
| Video Memory | 32 MB |
| Processor | 4CPUs |
| Execution Cap | 100% |
| Disk | 62 GB |

Tabela 4.3: Especificações da máquina virtual utilizada para a instalação do Openstack.

Através do ficheiro local.conf (ver anexo) é possível escolher quais os serviços a usar no OpenStack consoante as necessidades. A instalação realizada conta com os seguintes serviços do OpenStack:

- Keystone: serviço de identidade que fornece autenticação;
- Nova: serviço de computação que provisiona as VMs;
- Glance: serviço de imagem que provisiona as imagens às VMs;
- Cinder: serviço de armazenamento em bloco que fornece armazenamento Às VMs;
- Neutron: serviço de rede provisiona a rede às VMs;
- Horizon: seviço *dashboard* que providencia uma interface web;
- Celiometer: serviço de métricas que coleta dados;

### 4.3.2 Rede

Para implementar um sistema de redes capaz de satisfazer os requisitos, foram seguidos os seguintes passos (ver o script em anexo, assim como as especificações de cada componente criada, também em anexo):

- Em primeiro lugar foi definida uma rede virtual a qual vai ser uma interface para as VMs. Essa rede foi designada de net1;
- A esta rede foi associada um *subnet* designada de subnet1;
- O mesmo procedimento foi feito para implementar a rede "net2"e a *subnet* correspondente "subnet2";
- De seguida foi criado um *router* com três interfaces, a rede externa, a rede net1 e net2;

- O mesmo procedimento foi feito para implementar a rede "net2"e a *subnet* correspondente "subnet2", que irá constituir uma interface para o *router* "router1":

### 4.3.3 VMs

Primeiramente definiu-se como objetivo criar uma imagem própria baseada em Alpine para cada tipo de VM, A e B. É pretendido que essa imagem já contenha todas as ferramentas necessárias ao propósito da VM, poupando assim recursos e tempo. Assim, ao instanciar uma VM com a imagem esta já se vai encontrar pronta para executar as tarefas.

Para a $VM_A$ foi tido em conta os seguintes passos:

- Criar uma VM com a imagem do OS Alpine, à qual foi associada um bloco de 1GB com a opção *bootable* ligada.
- Foi feito o *setup* do Alpine no respetivo bloco.
- Foi instalado o NGINX e o NODE.JS no bloco.
- De seguida, foi feito o upload do bloco para uma imagem.

Para a $VM_B$ foi tido em conta os seguintes passos:

- Criar uma VM com a imagem do OS Alpine, à qual foi associada um bloco de 1GB com a opção *bootable* ligada.
- Foi feito o *setup* do Alpine no respetivo bloco.
- Foi instalado o FFMPEG no bloco
- De seguida, foi feito o upload do bloco para uma imagem.

De maneira a definir as regras da rede que vai ser interface às VMs foi criado um grupo de segurança com certas regras.

Tendo estas duas imagens *snapshots* do tipo QCOW2 no armazenamento de imagens agora é possível instanciar VMs com cada uma das imagens.

Para implementar as VMs capazes de satisfazer os requisitos, foram seguidos os seguintes passos (ver o script em anexo, assim como as especificações de cada componente criada, também em anexo):

- Tendo em conta as regras da rede de VMs foi definido um grupo de segurança com algumas regras a serem aplicadas a cada VM.

- Foram criadas duas $VM_B$ (VM_b1 e VM_b2), ambas com interfaces para a rede "net1"e "net2", e ainda atribuído o *floating* IP respetivo a cada uma.

- Foi criada a $VM_A$ designada de "VM_a", com interface à rede "net2"e atribuído o *floating* IP correspondente.

### 4.3.4 Monitorização

A monitorização da *cloud* pode ser vista na GUI do *dashboard* (ver figura 4.5) e é possível retirar métricas de quanta memória RAM a cloud está a consumir, quanto CPU usa, quantas componentes tem, entre outras.

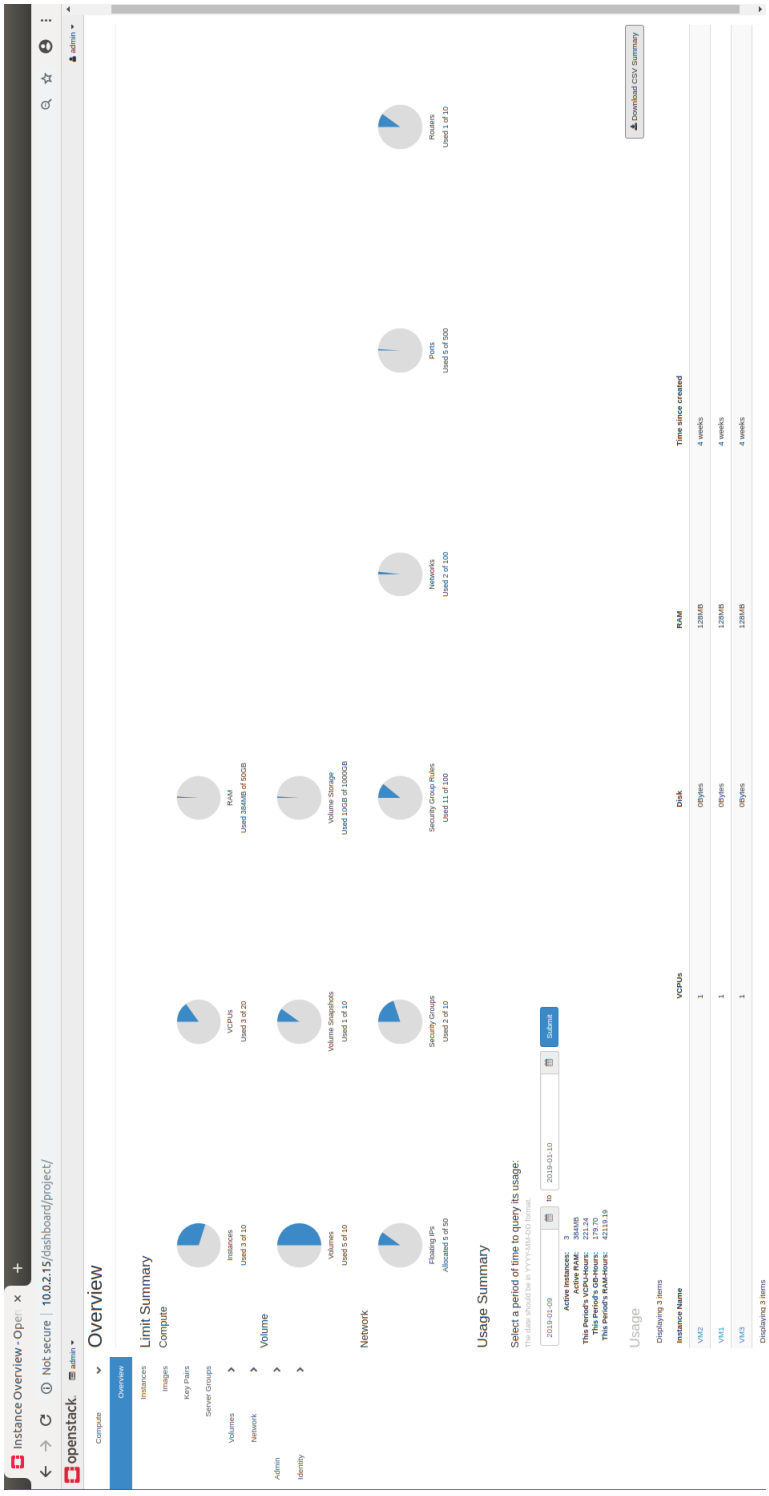



Figura 4.5: *Overview da cloud*

# Capítulo 5

# Validação

Neste capítulo são descritos os testes realizados a algumas funcionalidades das VMs A e B instanciadas na *cloud*.

Primeiramente foram executados testes à rede usando o software Iperf, uma ferramenta para realizar medições da largura de banda máxima alcançável em redes IP. Suporta o ajuste de vários parâmetros relacionados a temporização, *buffers* e protocolos (TCP, UDP, SCTP com IPv4 e IPv6). Para cada teste, ele informa a largura de banda, a perda e outros parâmetros.

Os testes servem para verificar a integridade da instalação do OpenStack. Essa verificação tornou-se a base para os testes.

Posteriormente foram realizados testes à receção de *streams* provenientes da rede externa, e transcodificação das mesmas, na $VM_B$.

Os dados são retirados do OpenStack, do software Iperf e da tecnologia FFMPEG usada na transcodificação.

## 5.1 Ambiente de teste e Metodologia

Todos os testes foram executados no ambiente da *cloud* privada descrita anteriormente e representada na figura 5.1, onde as VMs são geradas através de uma imagem *snapshot* (Ver recursos das VMs na tabela 5.1).

| **Recurso** | **VM_b1** | **VM_a** |
|---|---|---|
| RAM | 192 mB | 192 mB |
| CPU | 2 VCPU | 1 VCPUs |
| Free space | 591 mb (787 mb) | 2GB |
| Floating IP | 172.24.4.60 | 172.24.4.86 |

Tabela 5.1: Especificações dos recursos da VM A e da VMs B.

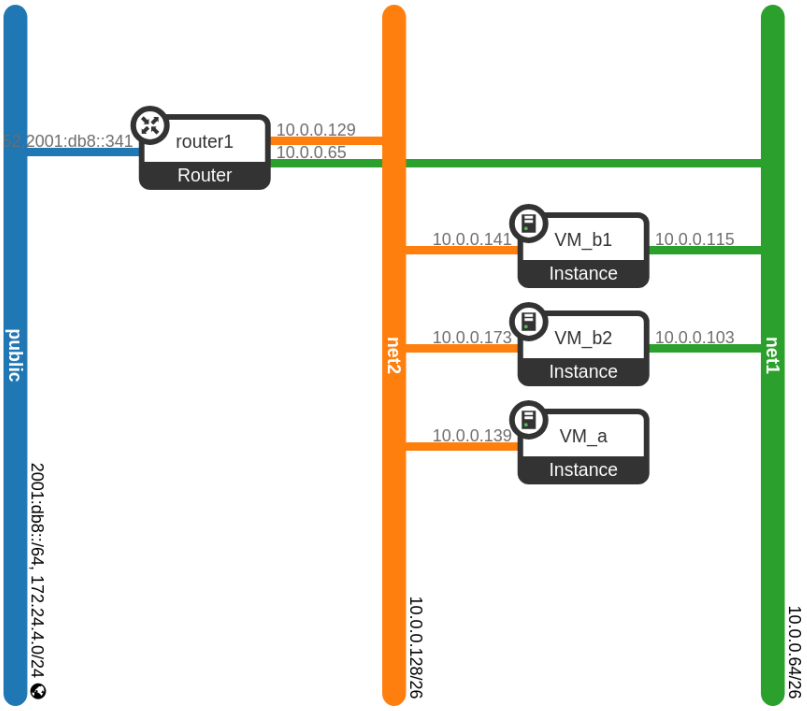



Figura 5.1: Cenário de teste (imagem retirada através do OpenStack *dashboard*).

### 5.1.1 Testes de rede

De seguida, são apresentados os resultados dos testes realizados ao sistema de redes criado do OpenStack. Os cenários testados foram:

- VM exterior (localizado na rede exterior) to $VM_A$ ;
- VM exterior para $VM_{B1}$;
- $VM_{B1}$ para $VM_A$;
- $VM_{B1}$ para $VM_{B2}$;

Primeiramente, são apresentados os resultados dos testes à rede nos vários cenários mencionados em cima, usando o protocolo TCP (ver tabela 5.2).

De seguida, foram realizados os mesmos testes, mas usando o protocolo UDP em vez de TCP (ver tabela 5.3).

| Comunicação | Largura de banda |
|---|---|
| VM exterior para $VM_A$ | 3.40 Gbits/s |
| VM exterior para $VM_{B1}$ | 3.54 Gbits/s |
| $VM_{B1}$ para $VM_A$ | 1.56 Gbits/s |
| $VM_{B1}$ para $VM_{B2}$ | 1.56 Gbits/s |

Tabela 5.2: Testes TCP na rede virtual criada.

| Capacidade de ligação | Largura de banda | Comunicação | Percentagem de perda erro |
|---|---|---|---|
| 500 Mb | 174 Mbits/s | $VM_{B1}$ para $VM_A$ | 24% |
| 500 Mb | 493 Mbits/s | VM exterior para $VM_{B1}$ | 20% |
| 1 G | 177 Mbits/s | $VM_{B1}$ para $VM_A$ | 23% |
| 1G | 770 Mbits/s | VM exterior para $VM_{B1}$ | 25% |

Tabela 5.3: Testes UDP na rede virtual criada.

## 5.2 Resultados

### 5.2.1 *Deployment* da *cloud* e *use cases*

Na figura 5.2 são apresentadas as várias camadas de *deployment* da *cloud*, o qual foi realizado através da API fornecida pelo OpenStack (ver fig. 5.3), tanto das componentes de rede como a das VMs.

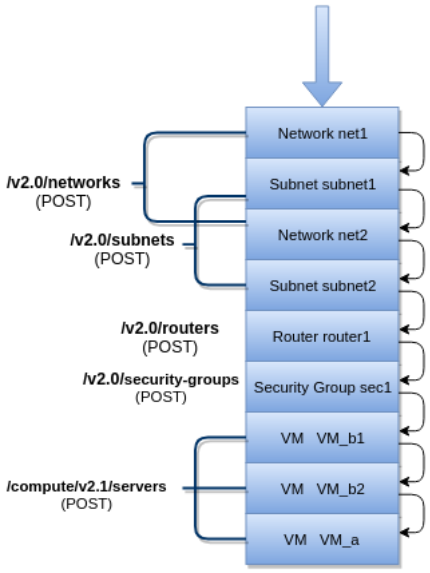
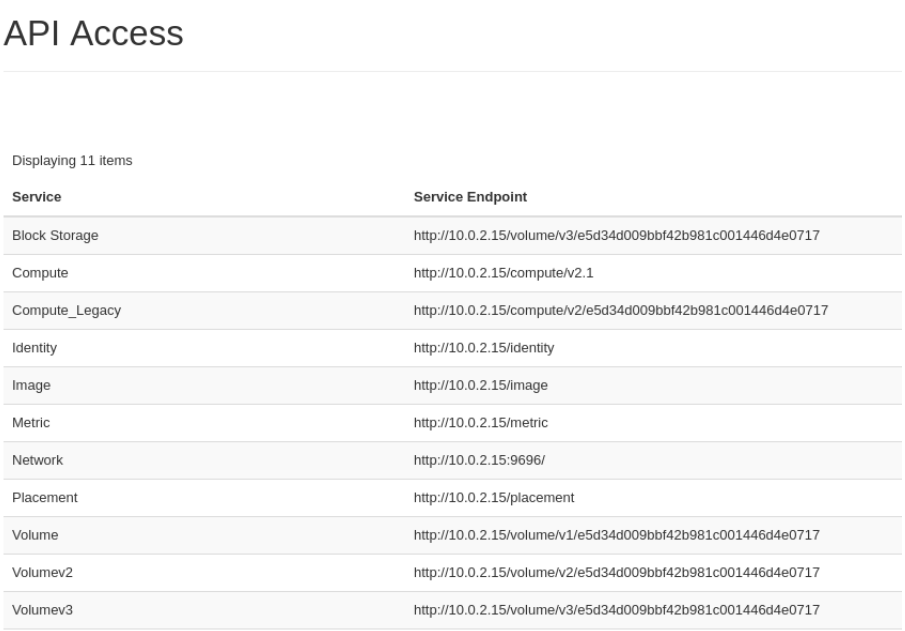



Figura 5.2: Camadas de *deployment* no *script* criado.

Na figura 5.4 apresentam-se os resultados do tempo decorrido em cada fase do processo de *deployment* do cenário. A duração total da operação foi em média cerca de 52 segundos.



API Access

Displaying 11 items

| Service | Service Endpoint |
|---|---|
| Block Storage | http://10.0.2.15/volume/v3/e5d34d009bbf42b981c001446d4e0717 |
| Compute | http://10.0.2.15/compute/v2.1 |
| Compute_Legacy | http://10.0.2.15/compute/v2/e5d34d009bbf42b981c001446d4e0717 |
| Identity | http://10.0.2.15/identity |
| Image | http://10.0.2.15/image |
| Metric | http://10.0.2.15/metric |
| Network | http://10.0.2.15:9696/ |
| Placement | http://10.0.2.15/placement |
| Volume | http://10.0.2.15/volume/v1/e5d34d009bbf42b981c001446d4e0717 |
| Volumev2 | http://10.0.2.15/volume/v2/e5d34d009bbf42b981c001446d4e0717 |
| Volumev3 | http://10.0.2.15/volume/v3/e5d34d009bbf42b981c001446d4e0717 |

Figura 5.3: API fornecida pelo OpenStack para cada serviço

Na etapa de instanciação das componentes da rede foram despendidos aproximadamente 24 segundos enquanto que a instanciação das respetivas VMs foi cerca de 29 segundos.

O tempo da instanciação varia razoavelmente, potencialmente devido ao fator de ser feita uma comunicação à API do OpenStack, podendo haver atrasos na rede. Por essa razão, os testes forma repetidos várias vezes e o valor apresentado é uma média dos mesmos.

Na tabela 5.4 é apresentado o tempo decorrido das funções mais relevantes e úteis, tendo em conta os objetivos propostos.

| **Ação** | **Tempo decorrido (s)** |
|---|---|
| Create network | 1,813 |
| Create and associate subnet | 2,129 |
| Create and configure router | 10,737 |
| Create and manage security group | 5,347 |
| Send VM to Openstack | 2,121 |
| Run VM | 7,826 |
| Stop VM | 16,346 |
| Restart VM | 7,945 |
| Replicate VM | 133,839 |
| Deploy two VMs | 12,271 |
| Remove VM | 4,234 |

Tabela 5.4: Métricas temporais de alguns *use cases*.

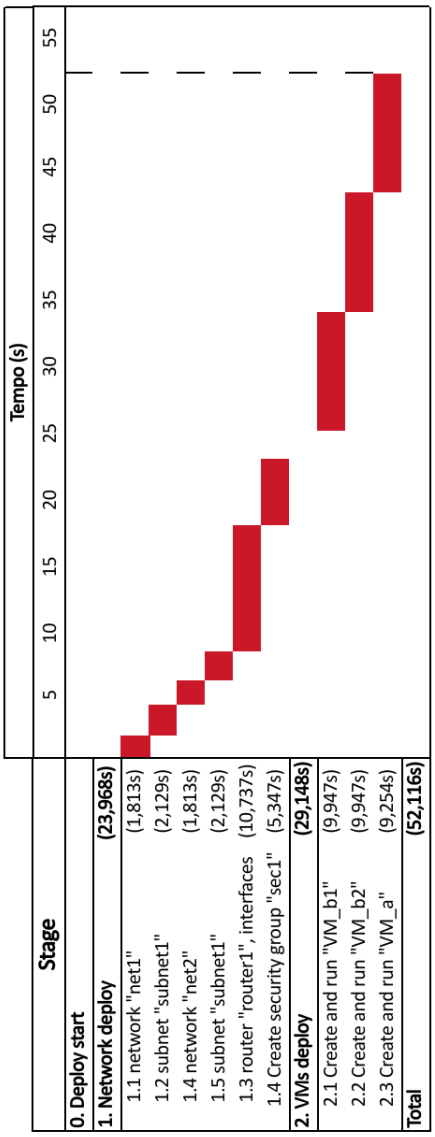



Figura 5.4: Tempo decorrido em cada etapa

### 5.2.2 Teste de streaming

Foram feitos vários testes às funcionalidades da máquina $VM_B$, as quais são receber a *stream* proveniente da rede pública, transcodificar o vídeo para os *codecs* de vídeo H.264 e áudio AAC, e enviar para a $VM_A$ a *stream* transcodificada na qual será armazenada (ver fig. 5.5).

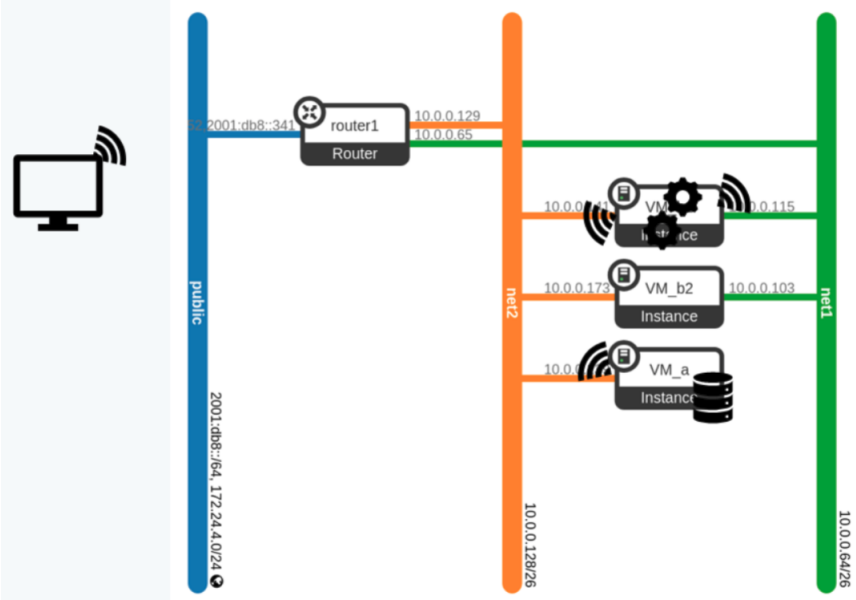



Figura 5.5: Teste de *streaming*

Os comandos FFMPEG usados para realizar os testes foram:

- Na VM instanciada na rede exterior:

```
1 ffmpeg -re -i nome_do_video.mp4 -codec copy -bsf:v
    h264_mp4toannexb -f rtp_mpegts rtp://172.24.4.60:7000
```

- Na VM_b1 :

```
1 ffmpeg -f mpegts -i rtp://172.24.4.60:7000 -vcodec libx264 -
    acodec aac -f rtp_mpegts rtp://172.24.4.86:7000
```

- Na VM_a :

```
1 ffmpeg -f mpegts -i rtp://172.24.4.86 -c copy output1.mp4
```

Para cada teste foi então utilizado um vídeo com características diferentes (ver tabela 5.5), sendo que algumas informações sobre a transmissão de cada vídeo para a $VM_B$, em cada teste, podem ser observadas nas tabelas 5.6, 5.7, 5.8.

| **Teste** | 1 | 2 | 3 |
|---|---|---|---|
| **Resolução** | 640x368 | 640x480 | 1280x720 |
| **Bitrate** | 1380 KB/s | 1376 KB/s | 2197 KB/s |
| **Duração** | 30,38 s | 180,32 s | 62,29 s |
| **Tamanho** | 5 MB | 30 MB | 10 MB |

Tabela 5.5: Características dos vídeos.

| VM | Unidade | Áudio | Vídeo | Total |
|---|---|---|---|---|
| VM exterior | pacotes | 1425 | 759 | 2184 |
| | kbytes | 1270,2 | 1051,5 | 2321,7 |
| VM_b1 | pacotes | 1422 | 756 | 2178 |
| | kbytes | 1266,8 | 1045,1 | 2311,9 |
| % de perda | | | | 0,27% |

Figura 5.6: Resultados no teste 1.

| VM | Unidade | Áudio | Vídeo | Total |
|---|---|---|---|---|
| VM exterior | pacotes | 8594 | 4582 | 13176 |
| | kbytes | 7644,8 | 5138,5 | 12783,3 |
| VM_b1 | pacotes | 8570 | 4568 | 13138 |
| | kbytes | 7615,1 | 5091,5 | 12706,6 |
| % de perda | | | | 0,29% |

Figura 5.7: Resultados no teste 2.

| VM | Unidade | Áudio | Vídeo | Total |
|---|---|---|---|---|
| VM exterior | pacotes | 2922 | 1557 | 4479 |
| | kbytes | 2602,5 | 4689,3 | 7291,8 |
| VM_b1 | pacotes | 2913 | 1552 | 4465 |
| | kbytes | 2593,6 | 4643,6 | 7237,2 |
| % de perda | | | | 0,31% |

Figura 5.8: Resultados no teste 3.

Foram também retiradas métricas de uso de CPU e memória RAM durante os testes realizados (ver tabelas 5.9, 5.10). Este consumo de recursos, como era esperado, aumentou consoante a qualidade dos vídeos utilizados.

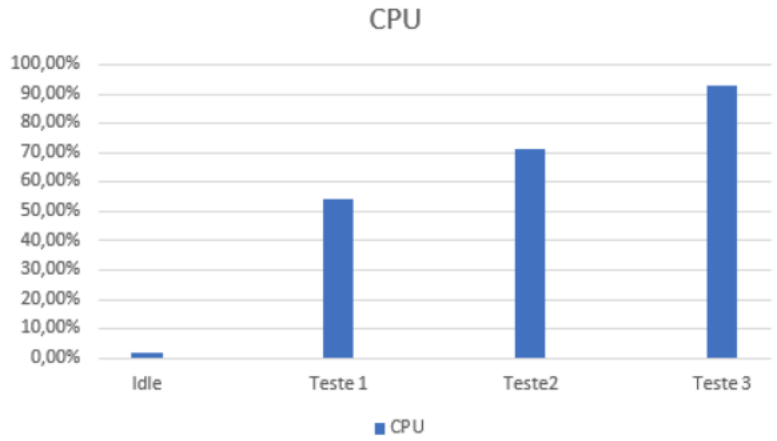



Figura 5.9: Uso de CPU durante os testes na VM_b1

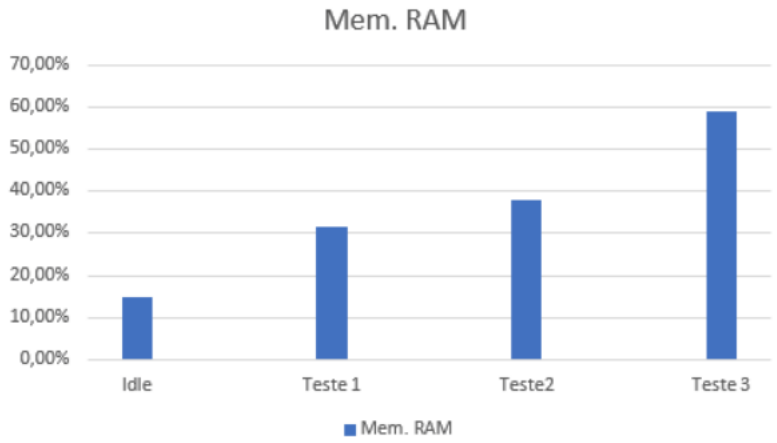



Figura 5.10: Uso de memória RAM durante os testes na VM_b1

# Capítulo 6

# Conclusões e Trabalho Fututro

## 6.1 Conclusões

As tecnologias de virtualização e de computação na *cloud* atualmente disponíveis permitem uma a rápida implementação e um fácil escalamento de aplicações e serviços, o que vem a possibilitar que haja um menor investimento inicial em infraestrutura. Por outro lado, a evolução em tecnologias como redes *ethernet* de 10 ou mais gigabit por segundo potenciam a utilização de soluções deste tipo para cenários de produção de conteúdo televisivo, com todos os benefícios da computação na *cloud* e ainda com o benefício de não serem necessários todos os equipamentos dispendiosos normalmente associados a esta atividade.

Nesta dissertação foram analisadas as tecnologias mais apropriadas à criação e configuração de uma *cloud* privada para o *deployment* de aplicações em máquinas virtuais.

Foram efetuados testes à solução apresentada, tendo sido validada a funcionalidade de reserva de recursos. Apesar de ter sido estudada, de forma atenta, a API fornecida pelo OpenStack, assim como as respetivas possibilidades de automatização do processo de instanciação das VMs, não o foi conseguido implementar na *framework*. Constitui assim, um trabalho a realizar no futuro e que certamente será continuado após a concretização da dissertação.

Assim, esta dissertação contribui para a proliferação das tecnologias de cloud computing, tornando-as mais acessíveis ao utilizador final que possa não ter um conhecimento aprofundado de aplicações e sistemas distribuídos.

## 6.2 Satisfação dos Objetivos

Grande parte dos objetivos inicialmente propostos foram satisfeitos. A solução implementada permitiu validar a aplicabilidade da arquitetura definida ao cenário da *framework*.

Apesar de ter sido escolhido um conjunto de soluções que se adequavam ao caso estudado e de estas satisfazerem os respetivos requisitos, seria possível a utilização da arquitetura definida com outras soluções de orquestração e para outros cenários de teste.

## 6.3 Trabalho Futuro

Tendo sido proposta uma arquitetura adaptável a diferentes cenários e ferramentas, esta solução pode ainda ser complementada por uma plataforma de orquestração, a qual visa fornecer métodos de expansão às suas funcionalidades, que permitem a criação de aplicações intermediárias e complementares, o que facilita o processo de instanciação de aplicações na *cloud*.

Apesar da dissertação ter findado, os trabalhos vão continuar, e o próximo objetivo é criar uma solução *end-to-end* NFV (funções de rede virtualizadas) juntando o OpenStack e o software Open Source MANO (OSM).

Network Functions Virtualization (NFV), ou funções de rede virtualizadas, oferecem uma nova forma de projetar, instanciar e gerir serviços de rede. O NFV separa as funções de rede, como a Network Adresses Translation (NAT), firewall, deteção de intrusos , DNS (Domain Name Service) e cache, para citar alguns, de dispositivos de hardware proprietários para que eles possam executar no software.

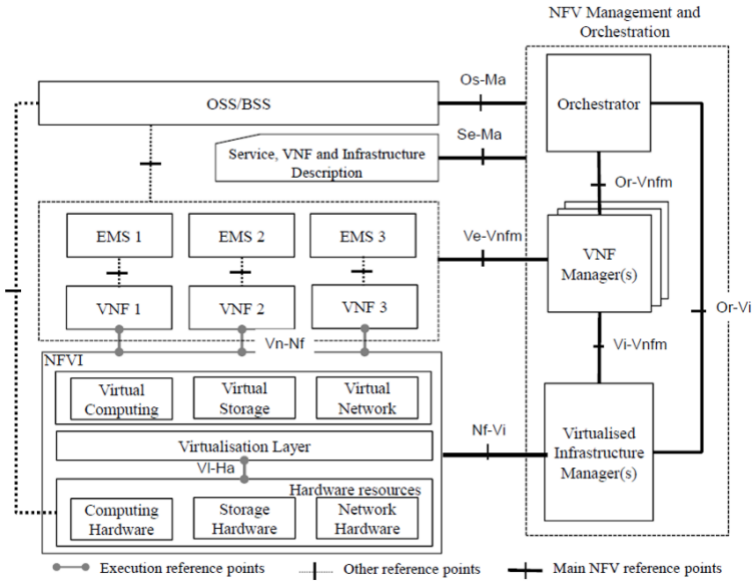



Figura 6.1: Arquitetura conjunta do OpenStack e OSM

A NFV foi projetado para consolidar e fornecer as componentes de rede necessárias para suportar uma infraestrutura totalmente virtualizada, incluindo servidores virtuais, armazenamento e até mesmo outras redes. Ele utiliza tecnologias de virtualização de TI padrão que são executadas em hardware de alto volume de serviços, comutadores e armazenamento para virtualizar funções de rede. É aplicável a qualquer plano de processamento de dados ou função de plano de controle em infra-estruturas de rede com e sem fio.

O OSM proporciona uma solução de gestão e orquestração open source alinhada com o modelo NFV da ETSI. Esta solução oferece qualidade na produção e encontra os requisitos de redes comerciais NFV.

Na figura 6.2 podemos observar como é que estas duas tecnologias podem trabalhar em conjunto e na figura 6.3 é possível observa a integração das duas tecnologias na arquitetura.

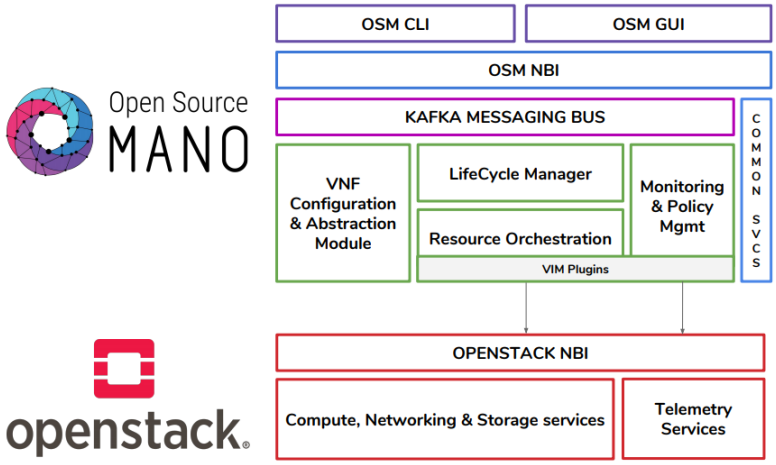



Figura 6.2: Arquitetura conjunta do OpenStack e OSM.[38]

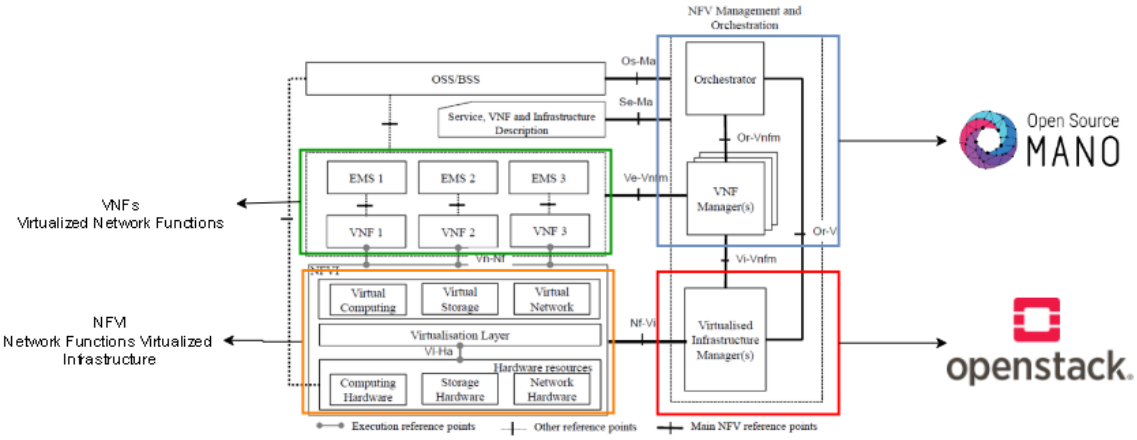



Figura 6.3: Interação entre o OpenStack e o OSM.

# Anexo A

# *Deployment* da solução

## A.1 Authenticate script

Listing A.1: bash version

```
#!/usr/bin/env bash
#
# source openrc [username] [projectname]
#
# Configure a set of credentials for $PROJECT/$USERNAME:
#   Set OS_PROJECT_NAME to override the default project 'demo'
#   Set OS_USERNAME to override the default user name 'demo'
#   Set ADMIN_PASSWORD to set the password for 'admin' and 'demo'

# NOTE: support for the old NOVA_* novaclient environment variables
   has
# been removed.

if [[ -n "$1" ]]; then
    OS_USERNAME=$1
fi
if [[ -n "$2" ]]; then
    OS_PROJECT_NAME=$2
fi

# Find the other rc files
RC_DIR=$(cd $(dirname "${BASH_SOURCE:-$0}") && pwd)

# Import common functions
source $RC_DIR/functions
```

```
25
26 # Load local configuration
27 source $RC_DIR/stackrc
28
29 # Load the last env variables if available
30 if [[ -r $RC_DIR/.stackenv ]]; then
31     source $RC_DIR/.stackenv
32     export OS_CACERT
33 fi
34
35 # Get some necessary configuration
36 source $RC_DIR/lib/tls
37
38 # The OpenStack ecosystem has standardized the term **project** as
      the
39 # entity that owns resources.  In some places **tenant** remains
40 # referenced, but in all cases this just means **project**.  We will
41 # warn if we need to turn on legacy **tenant** support to have a
42 # working environment.
43 export OS_PROJECT_NAME=${OS_PROJECT_NAME:-demo}
44
45 echo "WARNING: setting legacy OS_TENANT_NAME to support cli tools."
46 export OS_TENANT_NAME=$OS_PROJECT_NAME
47
48 # In addition to the owning entity (project), nova stores the entity
       performing
49 # the action as the **user**.
50 export OS_USERNAME=${OS_USERNAME:-demo}
51
52 # With Keystone you pass the keystone password instead of an api key
      .
53 # Recent versions of novaclient use OS_PASSWORD instead of
      NOVA_API_KEYs
54 # or NOVA_PASSWORD.
55 export OS_PASSWORD=${ADMIN_PASSWORD:-secret}
56
57 # Region
58 export OS_REGION_NAME=${REGION_NAME:-RegionOne}
59
```

```
60 # Set the host API endpoint. This will default to HOST_IP if
       SERVICE_IP_VERSION
61 # is 4, else HOST_IPV6 if it's 6. SERVICE_HOST may also be used to
       specify the
62 # endpoint, which is convenient for some localrc configurations.
       Additionally,
63 # some exercises call Glance directly. On a single-node installation
       , Glance
64 # should be listening on a local IP address, depending on the
       setting of
65 # SERVICE_IP_VERSION. If its running elsewhere, it can be set here.
66 if [[ $SERVICE_IP_VERSION == 6 ]]; then
67     HOST_IPV6=${HOST_IPV6:-::1}
68     SERVICE_HOST=${SERVICE_HOST:-[$HOST_IPV6]}
69     GLANCE_HOST=${GLANCE_HOST:-[$HOST_IPV6]}
70 else
71     HOST_IP=${HOST_IP:-127.0.0.1}
72     SERVICE_HOST=${SERVICE_HOST:-$HOST_IP}
73     GLANCE_HOST=${GLANCE_HOST:-$HOST_IP}
74 fi
75
76 # Identity API version
77 export OS_IDENTITY_API_VERSION=${IDENTITY_API_VERSION:-3}
78
79 # Ask keystoneauth1 to use keystone
80 export OS_AUTH_TYPE=password
81
82 # Authenticating against an OpenStack cloud using Keystone returns a
       **Token**
83 # and **Service Catalog**.  The catalog contains the endpoints for
       all services
84 # the user/project has access to - including nova, glance, keystone,
       swift, ...
85 # We currently recommend using the version 3 *identity api*.
86 #
87
88 # If you don't have a working .stackenv, this is the backup position
89 KEYSTONE_BACKUP=$SERVICE_PROTOCOL://$SERVICE_HOST:5000
90 KEYSTONE_AUTH_URI=${KEYSTONE_AUTH_URI:-$KEYSTONE_BACKUP}
91
```

```
92 export OS_AUTH_URL=${OS_AUTH_URL:-$KEYSTONE_AUTH_URI}
93
94 # Currently, in order to use openstackclient with Identity API v3,
95 # we need to set the domain which the user and project belong to.
96 if [ "$OS_IDENTITY_API_VERSION" = "3" ]; then
97     export OS_USER_DOMAIN_ID=${OS_USER_DOMAIN_ID:-"default"}
98     export OS_PROJECT_DOMAIN_ID=${OS_PROJECT_DOMAIN_ID:-"default"}
99 fi
100
101 # Set OS_CACERT to a default CA certificate chain if it exists.
102 if [[ ! -v OS_CACERT ]] ; then
103     DEFAULT_OS_CACERT=$INT_CA_DIR/ca-chain.pem
104     # If the file does not exist, this may confuse preflight sanity
        checks
105     if [ -e $DEFAULT_OS_CACERT ] ; then
106         export OS_CACERT=$DEFAULT_OS_CACERT
107     fi
108 fi
109
110 # Currently cinderclient needs you to specify the *volume api*
    version. This
111 # needs to match the config of your catalog returned by Keystone.
112 export CINDER_VERSION=${CINDER_VERSION:-3}
113 export OS_VOLUME_API_VERSION=${OS_VOLUME_API_VERSION:-
    $CINDER_VERSION}
```

## A.2 Deployment bash script

Listing A.2: bash version

```
1
2 #!/bin/bash
3
4 # get credentials as user "admin" in the project "admin"
5
6 . devstack/openrc admin admin
7
8 #Create network "net1"
9
10 openstack network create net1
11
```

```
12 #Associate subent "subnet1" to "net1"
13
14 openstack subnet create  --subnet-range 192.168.101.0/24 --dhcp --
      gateway 192.168.101.1 --dns-nameserver 8.8.8.8 --network net1
      subnet1
15
16 #Create network "net1"
17
18 openstack network create net2
19
20 #Associate subent "subnet2" to "net2"
21
22 openstack subnet create  --subnet-range 192.162.101.0/24 --dhcp --
      gateway 192.162.101.1 --dns-nameserver 8.8.8.8 --network net2
      subnet2
23
24 #Create router  "router1"  for one or more VMs_b network
25
26 openstack router create router1
27
28 #Attach extern, net1 e net2 interfaces
29
30 openstack router set router1 --external-gateway public
31
32 openstack router add subnet router1 subnet1
33
34 openstack router add subnet router1 subnet2
35
36 #Create the security group "sec1"
37
38 openstack security group create sec1
39
40 #Manage "sec1" rules
41
42 openstack security group rule create --protocol icmp --egress sec1
43
44 openstack security group rule create --protocol icmp --ingress sec1
45
46 openstack security group rule create --protocol tcp --ingress sec1
47
```

```
48 openstack security group rule create --protocol tcp --egress sec1
49
50 openstack security group rule create --protocol tcp --dst-port 22
      sec1
51
52 #Create VM "VM_b1"
53
54 openstack server create --image standard_alpine --flavor 2v.192m --
      security-group sec1 --key-name key3 --nic net-id=net1 VM_b1
55
56 #Create VM "VM_b2"
57
58 openstack server create --image standard_alpine --flavor 1v.192m --
      security-group sec1 --key-name key3 --nic net-id=net1 VM_b2
59
60 #Create VM "VM_a"
61
62 openstack server create --image standard_alpine --flavor 1v.192m --
      security-group sec1 --key-name key3 --nic net-id=net2 VM_a
63
64 #Set VMs floating IPs
65
66 openstack server add floating ip VM_b1 172.24.4.45
67
68 openstack server add floating ip VM_b1 172.24.4.86
69
70 openstack server add floating ip VM_a 172.24.4.60
```

## A.3 Especificações fornecidas pelo OpenStack das componentes

```
1
2  +---------------------------+----------------------------------------+
3  | Field                     | Value                                  |
4  +---------------------------+----------------------------------------+
5  | admin_state_up            | UP                                     |
6  | availability_zone_hints   |                                        |
7  | availability_zones        |                                        |
8  | created_at                | 2019-01-24T13:52:19Z                   |
9  | description               |                                        |
10 | dns_domain                | None                                   |
```

```
| id                        | 2e16c2ad-215b-4dc7-91ef-b8cca62a1380 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | False                                |
| is_vlan_transparent       | None                                 |
| location                  | None                                 |
| mtu                       | 1450                                 |
| name                      | net1                                 |
| port_security_enabled     | True                                 |
| project_id                | e5d34d009bbf42b981c001446d4e0717     |
| provider:network_type     | vxlan                                |
| provider:physical_network | None                                 |
| provider:segmentation_id  | 21                                   |
| qos_policy_id             | None                                 |
| revision_number           | 1                                    |
| router:external           | Internal                             |
| segments                  | None                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| updated_at                | 2019-01-24T13:52:19Z                 |
+---------------------------+--------------------------------------+
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allocation_pools  | 192.168.101.2-192.168.101.254        |
| cidr              | 192.168.101.0/24                     |
| created_at        | 2019-01-24T13:52:22Z                 |
| description       |                                      |
| dns_nameservers   | 8.8.8.8                              |
| enable_dhcp       | True                                 |
| gateway_ip        | 192.168.101.1                        |
| host_routes       |                                      |
| id                | 51e32743-033c-4274-8f10-74e32d4a753a |
| ip_version        | 4                                    |
| ipv6_address_mode | None                                 |
| ipv6_ra_mode      | None                                 |
| location          | None                                 |
| name              | subnet1                              |
```

```
| network_id        | 2e16c2ad-215b-4dc7-91ef-b8cca62a1380 |
| project_id        | e5d34d009bbf42b981c001446d4e0717     |
| revision_number   | 0                                    |
| segment_id        | None                                 |
| service_types     |                                      |
| subnetpool_id     | None                                 |
| tags              |                                      |
| updated_at        | 2019-01-24T13:52:22Z                 |
+-------------------+--------------------------------------+
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2019-01-24T13:52:24Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | 138d5241-e730-49d5-a393-a90c18347265 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | False                                |
| is_vlan_transparent       | None                                 |
| location                  | None                                 |
| mtu                       | 1450                                 |
| name                      | net2                                 |
| port_security_enabled     | True                                 |
| project_id                | e5d34d009bbf42b981c001446d4e0717     |
| provider:network_type     | vxlan                                |
| provider:physical_network | None                                 |
| provider:segmentation_id  | 6                                    |
| qos_policy_id             | None                                 |
| revision_number           | 1                                    |
| router:external           | Internal                             |
| segments                  | None                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| updated_at                | 2019-01-24T13:52:24Z                 |
```

```
+--------------------------+-------------------------------------+
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allocation_pools  | 192.162.101.2-192.162.101.254        |
| cidr              | 192.162.101.0/24                     |
| created_at        | 2019-01-24T13:52:25Z                 |
| description       |                                      |
| dns_nameservers   | 8.8.8.8                              |
| enable_dhcp       | True                                 |
| gateway_ip        | 192.162.101.1                        |
| host_routes       |                                      |
| id                | 9585473b-bacf-4fde-9288-e58b82efbc5c |
| ip_version        | 4                                    |
| ipv6_address_mode | None                                 |
| ipv6_ra_mode      | None                                 |
| location          | None                                 |
| name              | subnet2                              |
| network_id        | 138d5241-e730-49d5-a393-a90c18347265 |
| project_id        | e5d34d009bbf42b981c001446d4e0717     |
| revision_number   | 0                                    |
| segment_id        | None                                 |
| service_types     |                                      |
| subnetpool_id     | None                                 |
| tags              |                                      |
| updated_at        | 2019-01-24T13:52:25Z                 |
+-------------------+--------------------------------------+
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| created_at      | 2019-01-24T13:52:28Z                 |
| description     | sec1                                 |
| id              | aeabedb7-3f6d-4a91-8082-596c50bbe230 |
| location        | None                                 |
| name            | sec1                                 |
| project_id      | e5d34d009bbf42b981c001446d4e0717     |
| revision_number | 1                                    |
| rules           |                                      |
|                 |                                      |
| tags            | []                                   |
```

```
| updated_at      | 2019-01-24T13:52:28Z                 |
+-----------------+--------------------------------------+
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| created_at        | 2019-01-24T13:52:30Z                 |
| description       |                                      |
| direction         | egress                               |
| ether_type        | IPv4                                 |
| id                | 58afc977-72e7-4d83-a911-b7a4d789362d |
| location          | None                                 |
| name              | None                                 |
| port_range_max    | None                                 |
| port_range_min    | None                                 |
| project_id        | e5d34d009bbf42b981c001446d4e0717     |
| protocol          | icmp                                 |
| remote_group_id   | None                                 |
| remote_ip_prefix  | 0.0.0.0/0                            |
| revision_number   | 0                                    |
| security_group_id | aeabedb7-3f6d-4a91-8082-596c50bbe230 |
| updated_at        | 2019-01-24T13:52:30Z                 |
+-------------------+--------------------------------------+
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| created_at        | 2019-01-24T13:52:31Z                 |
| description       |                                      |
| direction         | ingress                              |
| ether_type        | IPv4                                 |
| id                | 3a5551d9-81a7-4b2b-a220-7ae4daac7d6c |
| location          | None                                 |
| name              | None                                 |
| port_range_max    | None                                 |
| port_range_min    | None                                 |
| project_id        | e5d34d009bbf42b981c001446d4e0717     |
| protocol          | icmp                                 |
| remote_group_id   | None                                 |
| remote_ip_prefix  | 0.0.0.0/0                            |
| revision_number   | 0                                    |
| security_group_id | aeabedb7-3f6d-4a91-8082-596c50bbe230 |
```

```
| updated_at        | 2019-01-24T13:52:31Z                 |
+-------------------+--------------------------------------+
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| created_at        | 2019-01-24T13:52:33Z                 |
| description       |                                      |
| direction         | ingress                              |
| ether_type        | IPv4                                 |
| id                | e98ce73f-907a-4239-9608-f61bf6c3e7f7 |
| location          | None                                 |
| name              | None                                 |
| port_range_max    | None                                 |
| port_range_min    | None                                 |
| project_id        | e5d34d009bbf42b981c001446d4e0717     |
| protocol          | tcp                                  |
| remote_group_id   | None                                 |
| remote_ip_prefix  | 0.0.0.0/0                            |
| revision_number   | 0                                    |
| security_group_id | aeabedb7-3f6d-4a91-8082-596c50bbe230 |
| updated_at        | 2019-01-24T13:52:33Z                 |
+-------------------+--------------------------------------+
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| created_at        | 2019-01-24T13:52:35Z                 |
| description       |                                      |
| direction         | egress                               |
| ether_type        | IPv4                                 |
| id                | 5c1b5a28-f4db-4cc5-b910-0ecfba7b3ca0 |
| location          | None                                 |
| name              | None                                 |
| port_range_max    | None                                 |
| port_range_min    | None                                 |
| project_id        | e5d34d009bbf42b981c001446d4e0717     |
| protocol          | tcp                                  |
| remote_group_id   | None                                 |
| remote_ip_prefix  | 0.0.0.0/0                            |
| revision_number   | 0                                    |
| security_group_id | aeabedb7-3f6d-4a91-8082-596c50bbe230 |
```

```
| updated_at        | 2019-01-24T13:52:35Z                 |
+-------------------+--------------------------------------+
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| created_at        | 2019-01-24T13:52:36Z                 |
| description       |                                      |
| direction         | ingress                              |
| ether_type        | IPv4                                 |
| id                | e493fa16-9f07-4c65-85f1-bd507ce34298 |
| location          | None                                 |
| name              | None                                 |
| port_range_max    | 22                                   |
| port_range_min    | 22                                   |
| project_id        | e5d34d009bbf42b981c001446d4e0717     |
| protocol          | tcp                                  |
| remote_group_id   | None                                 |
| remote_ip_prefix  | 0.0.0.0/0                            |
| revision_number   | 0                                    |
| security_group_id | aeabedb7-3f6d-4a91-8082-596c50bbe230 |
| updated_at        | 2019-01-24T13:52:36Z                 |
+-------------------+--------------------------------------+
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| admin_state_up          | UP                                   |
| availability_zone_hints |                                      |
| availability_zones      |                                      |
| created_at              | 2019-01-24T13:52:38Z                 |
| description             |                                      |
| distributed             | False                                |
| external_gateway_info   | None                                 |
| flavor_id               | None                                 |
| ha                      | False                                |
| id                      | 5247469b-0f46-4825-8ba2-e7972ac0c8a3 |
| location                | None                                 |
| name                    | router1                              |
| project_id              | e5d34d009bbf42b981c001446d4e0717     |
| revision_number         | 0                                    |
| routes                  |                                      |
```

```
| status                   | ACTIVE                                |
| tags                     |                                       |
| updated_at               | 2019-01-24T13:52:38Z                  |
+--------------------------+---------------------------------------+
+------------------------------+-----------------------------------+
| Field                        | Value                             |
+------------------------------+-----------------------------------+
| OS-DCF:diskConfig            | MANUAL                            |
| OS-EXT-AZ:availability_zone  |                                   |
| OS-EXT-SRV-ATTR:host         | None                              |
| OS-EXT-SRV-ATTR:hypervisor   | None                              |
| OS-EXT-SRV-ATTR:instance     |                                   |
| OS-EXT-STS:power_state       | NOSTATE                           |
| OS-EXT-STS:task_state        | scheduling                        |
| OS-EXT-STS:vm_state          | building                          |
| OS-SRV-USG:launched_at       | None                              |
| OS-SRV-USG:terminated_at     | None                              |
| accessIPv4                   |                                   |
| accessIPv6                   |                                   |
| addresses                    |                                   |
| adminPass                    | jaV4qmtKb6E5                      |
| config_drive                 |                                   |
| created                      | 2019-01-24T13:52:58Z              |
| flavor                       | 2v.192m (f2m)                     |
| hostId                       |                                   |
| id                           | e0e4-1363-44e9-875d-d94c5a        |
| image                        | standard_alpine                   |
| key_name                     | key3                              |
| name                         | VM_b1                             |
| progress                     | 0                                 |
| project_id                   | e5d34d009bbf42b981c001446d4e0717  |
| properties                   |                                   |
| security_groups              | name='sec1'                       |
| status                       | BUILD                             |
| updated                      | 2019-01-24T13:52:59Z              |
| user_id                      | 74dc6e4aa04a4fd9ab83bce419        |
| volumes_attached             |                                   |
+------------------------------+-----------------------------------+

+-----------------------------------+-------------------------+
```

```
| Field                          | Value                                |
+--------------------------------+--------------------------------------+
| OS-DCF:diskConfig              | MANUAL                               |
| OS-EXT-AZ:availability_zone    |                                      |
| OS-EXT-SRV-ATTR:host           | None                                 |
| OS-EXT-SRV-ATTR:hypervisor     | None                                 |
| OS-EXT-SRV-ATTR:instance       |                                      |
| OS-EXT-STS:power_state         | NOSTATE                              |
| OS-EXT-STS:task_state          | scheduling                           |
| OS-EXT-STS:vm_state            | building                             |
| OS-SRV-USG:launched_at         | None                                 |
| OS-SRV-USG:terminated_at       | None                                 |
| accessIPv4                     |                                      |
| accessIPv6                     |                                      |
| addresses                      |                                      |
| adminPass                      | cT9jdRxYBqC7                         |
| config_drive                   |                                      |
| created                        | 2019-01-24T13:53:20Z                 |
| flavor                         | 1v.192m (f1m)                        |
| hostId                         |                                      |
| id                             | ff30-f0f0-4498-8d18-dfa981           |
| image                          | standard_alpine                      |
| key_name                       | key3                                 |
| name                           | VM_b2                                |
| progress                       | 0                                    |
| project_id                     | e5d34d009bbf42b981c001446d4e0717     |
| properties                     |                                      |
| security_groups                | name='sec1'                          |
| status                         | BUILD                                |
| updated                        | 2019-01-24T13:53:20Z                 |
| user_id                        | 74dc6e4aa04a4fd9ab83bce419           |
| volumes_attached               |                                      |
+--------------------------------+--------------------------------------+
+--------------------------------+--------------------------------------+
| Field                          | Value                                |
+--------------------------------+--------------------------------------+
| OS-DCF:diskConfig              | MANUAL                               |
| OS-EXT-AZ:availability_zone    |                                      |
| OS-EXT-SRV-ATTR:host           | None                                 |
| OS-EXT-SRV-ATTR:hypervisor     | None                                 |
```

```
| OS-EXT-SRV-ATTR:instance    |                                      |
| OS-EXT-STS:power_state      | NOSTATE                              |
| OS-EXT-STS:task_state       | scheduling                           |
| OS-EXT-STS:vm_state         | building                             |
| OS-SRV-USG:launched_at      | None                                 |
| OS-SRV-USG:terminated_at    | None                                 |
| accessIPv4                  |                                      |
| accessIPv6                  |                                      |
| addresses                   |                                      |
| adminPass                   | CCiCkx8Y7ekV                         |
| config_drive                |                                      |
| created                     | 2019-01-24T13:53:40Z                 |
| flavor                      | 1v.192m (f1m)                        |
| hostId                      |                                      |
| id                          | b292-cf1c-474b-af97-d88c5d           |
| image                       | standard_alpine                      |
| key_name                    | key3                                 |
| name                        | VM_a                                 |
| progress                    | 0                                    |
| project_id                  | e5d34d009bbf42b981c001446d4e0717     |
| properties                  |                                      |
| security_groups             | name='sec1'                          |
| status                      | BUILD                                |
| updated                     | 2019-01-24T13:53:39Z                 |
| user_id                     | 74dc6e4aa04a4fd9ab83bce              |
| volumes_attached            |                                      |
+-----------------------------------------------------------------+
```

# Bibliografia


[1] *Speedtest Global Index*. disponível em mhttp://www.speedtest.net/global-index. Junho 2018.

[2] *H.264, Advanced video coding for generic audiovisual services*. disponível em http://www.itu.int/rec/T-REC-H.264. Junho 2018.

[3] Jan De Cock et al. "A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications". Em: *Applications of Digital Image Processing XXXIX*. Vol. 9971. International Society for Optics e Photonics. 2016, p. 997116.

[4] *High Efficiency Video Coding (HEVC)*. disponível em http://hevc.info/. Junho 2018.

[5] Jim Bankoski, Paul Wilkins e Yaowu Xu. "Technical overview of VP8, an open source video codec for the web". Em: *Multimedia and Expo (ICME), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1–6.

[6] *VP9, Google's Latest Open Standard for Video Compression*. disponível em https://www.encoding.com/vp9/. Maio 2018.

[7] Debargha Mukherjee et al. "The latest open-source video codec VP9-an overview and preliminary results". Em: *Picture Coding Symposium (PCS), 2013*. IEEE. 2013, pp. 390–393.

[8] Michael Maruschke et al. "Review of the opus codec in a WebRTC scenario for audio and speech communication". Em: *International Conference on Speech and Computer*. Springer. 2015, pp. 348–355.

[9] *Overview*. disponível em http://opus-codec.org/. Maio 2018.

[10] Jean-Marc Valin e Cary Bran. *WebRTC Audio Codec and Processing Requirements*. Rel. téc. 2016.

[11] *AAC*. disponível em https://en.wikipedia.org/wiki/Advanced_Audio_Coding. Junho 2018.

[12] Pengyu Zhao et al. "A mobile real-time video system using RTMP". Em: *Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on*. IEEE. 2012, pp. 61–64.

[13] *Protocolo RTMP*. disponível em https://rafow.wordpress.com/2008/05/26/protocolo-rtmp/. Junho 2018.

[14] Henning Schulzrinne et al. *RTP: A transport protocol for real-time applications*. Rel. téc. 2003.

[15] *webRTC*. disponível em https://en.wikipedia.org/wiki/WebRTC. Maio 2018.

[16] Salvatore Loreto e Simon Pietro Romano. *Real-Time Communication with WebRTC: Peer-to-Peer in the Browser*. "O'Reilly Media, Inc.", 2014.

[17] *webRTC*. disponível em https://webrtc.org. Maio 2018.

[18] Lakshay Malhotra, Devyani Agarwal e Arunima Jaiswal. "Virtualization in cloud computing". Em: *J. Inform. Tech. Softw. Eng* 4.2 (2014).

[19] Carlos Becker Westphall et al. "CLOUD COMPUTING 2017 Proceedings: The Eighth International Conference on Cloud Computing, GRIDs, and Virtualization". Em: (2017).

[20] Wes Felter et al. "An updated performance comparison of virtual machines and linux containers". Em: *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On*. IEEE. 2015, pp. 171–172.

[21] Dawson R Engler, M Frans Kaashoek et al. *Exokernel: An operating system architecture for application-level resource management*. Vol. 29. 5. ACM, 1995.

[22] *What is a unikernel, and why does it matter?* disponível em https://www.hpe.com/us/en/insights/articles/what-is-a-unikernel-and-why-does-it-matter-1710.html. Outubro 2017.

[23] Anil Madhavapeddy e David J Scott. "Unikernels: the rise of the virtual library operating system". Em: *Communications of the ACM* 57.1 (2014), pp. 61–69.

[24] Alfred Bratterud, Andreas Happe e Robert Anderson Keith Duncan. "Enhancing cloud security and privacy: the Unikernel solution". Em: *Eighth International Conference on Cloud Computing, GRIDs, and Virtualization, 19 February 2017-23 February 2017, Athens, Greece*. Curran Associates. 2017.

[25] Michael Armbrust et al. "A view of cloud computing". Em: *Communications of the ACM* 53.4 (2010), pp. 50–58.

[26] Harald Sundmaeker et al. "Vision and challenges for realising the Internet of Things". Em: *Cluster of European Research Projects on the Internet of Things, European Commision* 3.3 (2010), pp. 34–36.

[27] *The Internet of Things How the Next Evolution of the Internet Is Changing Everything*. disponível em https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf. Maio. 2017.

[28] *Cisco Global Cloud Index: Forecast and Methodology 2017-2021*. disponível em https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html. Maio. 2017.

[29] *Cisco Global Cloud Index 2014–2019, update 2015*. disponível em https://www.cisco.com/c/dam/m/en_us/service-provider/ciscoknowledgenetwork/files/547_11_10-15-DocumentsCisco_GCI_Deck_2014-2019_for_CKN__10NOV2015_.pdf. Maio 2018.

[30] Jayavardhana Gubbi et al. "Internet of Things (IoT): A vision, architectural elements, and future directions". Em: *Future generation computer systems* 29.7 (2013), pp. 1645–1660.

[31] Peter Mell, Tim Grance et al. "The NIST definition of cloud computing". Em: (2011).

[32] Ivan Stojmenovic e Sheng Wen. "The fog computing paradigm: Scenarios and security issues". Em: *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*. IEEE. 2014, pp. 1–8.

[33] Flavio Bonomi et al. "Fog computing and its role in the internet of things". Em: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM. 2012, pp. 13–16.

[34] Weisong Shi e Schahram Dustdar. "The promise of edge computing". Em: *Computer* 49.5 (2016), pp. 78–81.

[35] *Internet of Things Forecast*. disponível em https://www.ericsson.com. Maio 2018.

[36] *Cabinet Security*. disponível em https://www.belden.com/blog/data-centers/cabinet-security-more-important-than-you-might-think, note = Maio. 2017.

[37] *Data center physical hacks: How to safeguard equipment in cabinets*. disponível em https://www.datacenterdynamics.com/opinions/data-center-physical-hacks-how-safeguard-equipment-cabinets/. Maio. 2017.

[38] *Openstack*. disponível em https://www.openstack.org/. Outubro 2018.

[39] Omar Sefraoui, Mohammed Aissaoui e Mohsine Eleuldj. "OpenStack: toward an open-source solution for cloud computing". Em: *International Journal of Computer Applications* 55.3 (2012), pp. 38–42.

[40] *Kernel-based Virtual Machine*. disponível em http://www.linux-kvm.org/. Outubro 2018.

[41] *VMware*. disponível em https://www.vmware.com/. Outubro 2018.

[42] *Xen project*. disponível em http://www.xenproject.org/. Outubro 2018.

[43] *Hyper-V*. disponível em https://en.wikipedia.org/wiki/Hyper-V/. Outubro 2018.

[44] *Linux Containers*. disponível em https://linuxcontainers.org/lxc/introduction/. Outubro 2018.

[45] *Apache, The Apache Software Foundation*. disponível em http://www.apache.org/foundation/.

[46] *The Channel, Cloud.com takes on virty infrastructure*. disponível em http://www.channelregister.co.uk/2010/05/04/cloud_com_launch/,

[47] *Citrix Systems, Inc.* disponível em http://www.citrix.com/.

[48] *The Apache Software Foundation Incubator*. disponível em http://incubator.apache.org/.

[49] Lei Huang e Yonggao Yang. "Facilitating Education Using Cloud Computing Infrastructure". Em: *J. Comput. Sci. Coll.* 28.4 (2013), pp. 19–25. ISSN: 1937-4771. URL: http://dl.acm.org/citation.cfm?id=2458539.2458542.

[50] Tessema M Mengistu et al. "cuCloud: Volunteer Computing as a Service (VCaaS) System". Em: *International Conference on Cloud Computing*. Springer. 2018, pp. 251–264.

[51] *Concepts and Terminology*. disponível em http://docs.cloudstack.apache.org. Maio 2018.

[52] *Node.js*. disponível em https://www.nodejs.org/. Outubro 2018.

[53] Stefan Tilkov e Steve Vinoski. "Node. js: Using JavaScript to build high-performance network programs". Em: *IEEE Internet Computing* 14.6 (2010), pp. 80–83.

[54] *Npm*. disponível em https://www.npmjs.com/. Outubro 2018.

[55] *NGINX*. disponível em https://www.nginx.com/. Outubro 2018.

[56] Ioannis K Chaniotis, Kyriakos-Ioannis D Kyriakou e Nikolaos D Tselikas. "Is Node. js a viable option for building modern web applications? A performance evaluation study". Em: *Computing* 97.10 (2015), pp. 1023–1044.

[57] *5G city*. disponível em https://www.5gcity.eu, note = Outubro 2018.

[58] *Arquitetura exemplo de Deployment do OpenStack*. disponível em https://www.openstack.org/. Outubro 2018.