Multi-Cloud

Alexandre Canalle[1], **Ariel Frozza**[1]

[1]Instituto de Informática – Universidade Católica do Paraná (PUC-PR)
Caixa Postal 15.064 – 91.501-970 – Curitiba – PR – Brazil

arielfrozza@gmail.com, roxsnd@gmail.com

Abstract. *This meta-paper describes the style to be used in articles and short papers for SBC conferences. For papers in English, you should add just an abstract while for the papers in Portuguese, we also ask for an abstract in Portuguese ("resumo"). In both cases, abstracts should not have more than 10 lines and must be in the first page of the paper.*

Abstract. *Neste trabalho exploraremos os conceitos de Multi-Cloud e Infrastructure as Code, bem como suas possíveis utilizações em ambientes corporativos. Também descreveremos as dificuldades na implementação de Multi-Cloud e elencaremos possíveis soluções para as dificuldades descritas. Por fim, apresentaremos um roteiro de uso das ferramentas OpenSource Terraform e Ansible para demonstrar a implementação de um serviço web em duas Nuvens simultaneamente, uma pública (AWS) e uma privada (Openstack).*

1. Introdução

Atualmente, o uso de serviços e recursos a partir de múltiplos provedores de Cloud simultaneamente é devido pela necessidade de seus consumidores expressas em requerimentos tais como qualidade de serviço e custo. As organizações que estão migrando para a Nuvem, seja para estender ou substituir sua infraestrutura on-premises, buscam soluções confiáveis, seguras e, se possível, sem ficar aprisionados à fornecedores ou soluções fechadas (*vendor lock in*). Entretanto, a navegação entre os diversos serviços ofertados e orquestração de recursos em múltiplas clouds (usando Infraestrutura como Código, por exemplo) ainda está distante da maturidade. Alguns obstáculos devem ser transpostos para que isso ocorra, já que, há riscos reais que devem ser mitigados; todos os provedores de Cloud experimentam, eventualmente, períodos de indisponibilidade que podem causar impactos a negócios [2], além disso, performance variável também é um problema por que a maioria dos provedores "overprovision"sua infraestrutura virtualizada e isto resulta em degradação de performance e qualidade de serviço [42]. Outro ponto importante é que a maioria dos provedores (inclusive os lideres de mercado) de Nuvem usam/disponibilizam Software e APIs proprietárias e tanbém não são aderentes à padrões de Cloud API como a Tosca [39] ou OASIS CAMP [41].

Este artigo pretende descrever alguns desafios do uso em multi cloud e levantar quais melhorias são necessárias aplicarmos às atuais soluções para atender as necessidades dos usuários de multiplas clouds. Primeiramente, a definição e a necessidade de Multi-clouds é discutida, em seguida, um modelo padrão de arquitetura para Multi-cloud é apresentado. Por último, algumas soluções (software) que tornam possíveis a Infraestrutura como Códio em Multi cloud são apresentados e discutidos. Por último, ilustraremos

os processos, requisitos e desafios na implementação de IaaS em multi cloud usando Infre as code.

Este artigo não apresenta uma abordagem inovadora, Noentanto, ele intenta apontar algumas lacunas/dificuldades a serem sanadas por desenvolvedores de Multi-cloud, em especial no que diz aos frameworks e padrões de provisionamento em Nuvem (pública, privada ou híbrida) que ofereçam softwares e APIs não proprietárias mas que disponibilizam funcionalidades compatíveis/equivalentes com os softwares e serviços ofertados pelos principais provedores de Nuvem para que o provisionamento/orquestração/uso de vários provedores de Núvens diferentes seja possível.

Outro benefício importante deste esstud é a enfase em padrões "fault tolerant"que implicam na garantia que sistemas atinjam uptime e resiliencia à stress e falhas adequadas.

Por fim, este estudo (as soluções definidas/proostas) garante que os padrões funcionam também em cloud privadas, pois:

1) Custos de longo prazo para cloud publica podem ficar mais altos que on-premises devido ao OPEX da clou publica eventualmente ultrapassa o CAPEX da cloud privada quando servidores e outros equipamentos são comprados.

2) Por causa de segurança ou motivos regulatórios, pode ser necessário manter parte do workload on-premises.

3) Performance on-premises pode ser melhor por ue você controla a taxa de over-subscrição e cargas submetidas a uma mesma instância de virtualização.

Neste artigo vamos identificar e categorizar os principais desafios para atingir o objetivo principal. Em seguida, vamos identificar os atributos principais de qualidade desejados para o deploy dos padrões em Cloud. Depois, proporemos soluções de padrão de qualidade para cada padrão apresentado.

Todos os padrões serão unificados em um framework Multi-Cloud coeso.

Por fim, validaremos oos padrões chaves com experimentos para a "feasibility"dos casos de "fault-tolerance"e "self-healing"usando padrões open-source de implementação.

Implementação Open-source da plataforma com padrão neutro (vendor-neutral) será desenvolvida/apresentada para validar padrões e guias de adoção. Usaremos open-source software para demonstrar como os padrõe podem ser impleentados fazem este estudo mais util/completo/específico que padrões enéricos como os descrito em http://www.cloudpatterns.org. O resultado deste estudo será um catálogo de Padrões de Cloud "Fault Tolerant", Seguro e de alta performance com as soluções mapeadas com software open-source com exemplos do deploy (e alguns templates/pseudo-codigo) que podem ser usados para implementar um stack completo de infraestrutura que será compativel com a especificação OASIS CAMP (Cloud Application Management for Platforms) e mais tarde, com OASIS TOSCA (Topology and Orchestration Specification for Cloud Applications).

Em seguida, todos os padrões serão validados usando software Open Source. A maioria das ofertas em cloud não seria possível sem uso de software open source (e.g. hypervisors Xen e KVM). Como selecionamos os melhores projetos open-source? We need to selec the projects with a large and active community, quantos "commiters"estão

ativos, quantos contribuidores são de diferentes organizações ou este é um projeto open-source de uma única empresa (algumas empresas desenvolvem open-source software mas não aceitam contribuições externas - Read Only Opensource). Idealmente, melhor ainda se a solução open-source tiver alguma empresa mantenedora/responsável por oferecer serviços de consultoria e bug-fixes (caso a empresa não tenha todo o expertise in-house).

2. Design Patterns and Frameworks for Effective Multi-Cloud Deployment

2.1. Multi-Cloud Foundation Patterns focused on Fault Tolerant Deployment Solutions

Clouds podem ser usadas de forma serial, por exemplo, quando migramos de uma Cloud para outra, ou simultanea, quando usamos simultaneamente recursos de duas ou mais Clouds diferentes. O senário mais comum para o caso de uso simultâneo é a Cloud Ibrida, quando alguns serviços permanecem um uma nuvem (Privada) e outros seriços estão em uma núvem Pública.

Os motivos que justificam o uso de multiplas clous são inúmeros e dependem da natureza do negócio de cada usuário, mesmo assim é possível citar algumas vantagems do uso simultâneo de dois ou mais provedores de Cloud:

1) **Tolerancia à falhas** (faul-tolerance) por envolver uso de mais de um provedor de Nuvem simultaneamente.

2) **Neutralidade de fornecedor** como sendo a possibilidade de implementar IaaS nos principais proveodres de Nuvens usando soluções open-source para evitar o *vendor lock-in*.

3) **Performance** como abilidade para alterar/ajustar cargas para outros provedores de Cloud em caso de degradação de performance ou qualidade de serviço em um ou mais provedores.

4) **Segurança**

5) **Eficiencia de custos** como abilidade de aproveitar os melhores preços de recursos computacionais e serviços em um ambiente multi-cloud.

Os custos de operação em um determinado provedor Cloud variam em função dos requisitos de uso e da época de contratação dos serviços (. . . e.g. AWS spot instance e nos últimos 2 anos o custo por hora do EC2 baixou cerca de 30%) assim como os custos de trocar de provedor podem ficar muito altos caso a infraestrutura e/ou aplicações tenham que ser refeitos/revistos inteiramente. Com o uso de padrões *vendor neutral*, o objetivo seria ser capaz de mudar de provedor de cloud ou priorizar a execução on-premises de acordo com a conveniência, sem ter que alterar o software stack.

Muitos termos são encontrados na literatura para designar o uso simultâneo de duas ou mais clouds. A citar os mais recorrentes [artigo multicloud]; Multi-cloud, Cloud-federation, Inter-cloud, Hybryd Cloud, Cloud of Clouds, Sky Computing, Aggregated Clouds, Fog Computing, Distributed Clouds, etc.

Sendo assim, achamos útil delimitar o conceito de Multi-cloud abordado neste artigo de outros modelos de uso combinado de Clouds.

De acordo com [ref 6 do art multicloud], temos dois modelos de entraga de

serviços em múltiplas clouds; Federated Cloud e Multi-Cloud. A diferença entre estes modelos seria o grau de colaboração entre os Provedores de Cloud e pelo mode pelo qual os usuários interagem com as Clouds. No primero modelo há o acordo de uso compartilhado dos recursos entre os Provedores de Cloud. Os usuários de uma núvem federada não ficam sabendo, na maioria dos casos, de qual dos provedores um determinado recurso está sendo consumido. No caso do Multi-cloud, o usuário está ciente e é restponsável pela alocação de recursos em um ou outro provedor e não há nuência dos provedores para uso compartilhado de recursos entre os provedores.

Sky Computing, Aggregated Clouds, Multi-tier Clouds ou Cross-Cloud são casos particulares de Federated Clouds e portanto não serão abordadas neste artigo.

Um dos maiores problemas é a interoperabilidade entre diferentes Cloudse a portabilidade de aplicações entre diferentes provedores.

O tipo mais comum de Multi-cloud é a Cloud-Hibrida na qual involve duas ou mais clouds, por exemplo, uma Cloud Privada e uma Pública. Comumente [artigo multicloud], esse modelo de Cloud Híblrida é usado para ”Cloud Bursting”onde os recursos são espandidos para a Cloud Pública quando os recursos da Cloud privada chegam nos níveis máximos pré-definidos. A migração de uma Cloud para outra, mesmo que apenas uma vez, é outro exemplo de uso de (one-time) Multi-cloud.

A seguir descreveremos um framework para o uso de Multi-cloud envolvendo duas Clouds, uma Privada e outra Pública.

3. Design Patterns e Framework for Effective Multi-Cloud Deployment

A seguir vamos explorar um padrão de implementação de multi-cloud que propõem ajudar a atender dois casos comuns de uso de multi-cloud:

1) Public Cloud Busting Pattern: em algumas situações, a capacidade on premises de uma empresa podem exaurir e pode-se querer usar a capacidade em uma cloud pública para suprir uma determinada demanda.

2) Alta Disponibilidade: Eventualmente, um provedor de Cloud pode ficar (parcial ou totalmente) indisponível ou com degradação na qualidade dos serviços por diversos motivos, pode-se querer migrar de um provedor para outro os serviços e recursos para não impactar o usuário final. Por exemplo, em caso de falha ou manutenção do datacenter que hospeda a Cloud privada, pode-se migrar serviços para uma Cloud pública, mesmo que temporariamente.

3) Cost Efficiency Multi-clourd Pattrn: Em alguns casos, provedores oferecem descontos ou modificam seus preçõs em função da oferta e demanda, pode-se migrar uso de recursos e serviços de uma Cloud para outra sem que o cucuário final seja impactado.

A [Figura 1] ilustra as atividades e processos envolvidos para atingir os objetivos descritos acima. Outros benefícios, tais como Tolerância à Falhas, Segurança, Validação experimental, etc, também podem ser alcançados usando este mesmo padrão ou após serem feitas pequenas modificações, os quais não serão diretamente abordados neste artigo, mas podem ser entendidos de forma mais completa em [3 a fisher].

1- **Multi-cloud Telemetry**: coleta e agrega dados de capacidade das Clouds integrantes da Multi-cloud. Na Figura estao representadas uma Cloud Privada e outra Publica,

mas poderiamos expandir essa mesma arquitetura para um conjunto maior de diferentes provedores. As informações coletadas são repassadas para o **"SLA Enforcer Rules Engine"**, que valida a condição para a migração de recursos de uma Cloud Para outra, como exemplo, se a capacidade de uma das clouds chegar a 80%, faz-se deploy de novos recursos apenas na outra Cloud. O **"SLA Enforcer Rules Engine"** pode avaliar um conjunto de condições que se faça adequado para cada negócio ou situação, por exemplo ele pode avaliar condições de custo ou tempo de resposta, além da capacidade. A avaliação deste processo dispara o gatilho para iniciar as novas implementações através do **"multi Cloud Deployer/orquestrator"**. Além disso, o **"Multi-cloud Telemetry + Log Aggregator"**, também gera subsídio para a bilhetagem através de métricas de utilização que são contabilizadas e disponibilidadas para o usuário através do processo **Multi-cloud Billing**. Também os alertas de erros são tratados e processados pelo processo **Sistema de Alertas e Notificações**.

2 e 3- **"multi Cloud Deployer/orquestrator"**: utiliza as definições de implementações armazenadas como código em texto em controle de versão, GIT por exemplo. Estas definições de implementações são plicadas sobre as imagens selecionadas do Pipeline de Imagens binárias que podem ser VMs (em formato OVS, por exemplo) ou Containers (Docker, por exemplo).

4- Toda informação sensível, em especial senhas ou chaves são obtidas do cofre de senhas, que por motivos de segurança e de facilidade de acesso, geralmente está on premises e não em uma das nuvens que fazem parte da Multi-cloud.

5-

6-

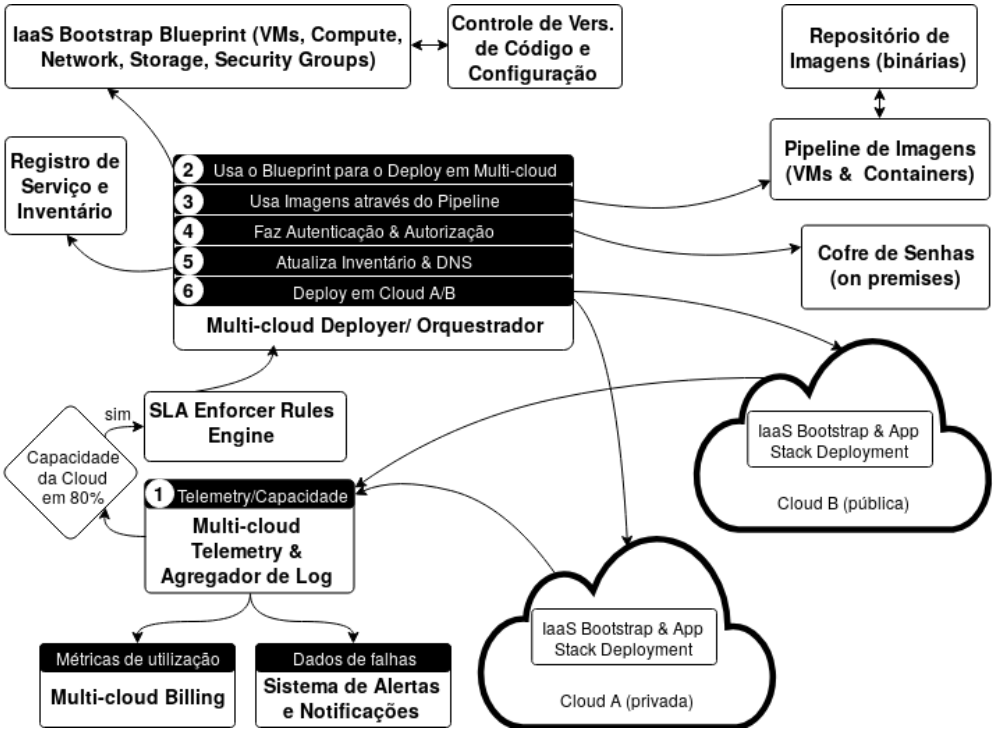

Figura 1. legenda longa

O cenário descrito acima inicia em 1 - Telemetry, assim possibilitando situações automatizadas de orquestração dos recursos na Multi-cloud, entretanto, um humano também poderia iniciar esse processo, passando instruções diretamente ao box **Multi-cloud Deployer/Orquestrator**, porém, na Figura1 essa situação não é representada.

Os processos descritos acima e esquematizados na Figura1 podem ser bastante complexos e envolver outros subprocessos e também um conjunto diversificado de ferramentas e softwares. Como exemplo, exploremos em mais detalhes o **Pipeline de Imagens**:

— check location 2363

3.0.1. Initial Multi-Cloud General Deplyment Fault Tolerance Challenges

3.0.2. Multi-Cloud Management after initial deployment and Dealing with Failures

3.0.3. Cost Efficiency Problems

3.0.4. Security Problems

3.0.5. Application Deployment in Multi-Cloud Environment

3.1. Multi-cloud management after initial Deployment and dealing with failures in automated way

3.1.1. title

3.2. Cost efficiency patterns

3.2.1. title

3.3. security related patterns

3.3.1. title

3.4. multi-cloud control plane framework

4. experimental validation

5. Conclusion

Conteúdo da sub-seção conteúdo da sub-seção conteúdo da sub-seção conteúdo da sub-seção. Conteúdo da sub-seção conteúdo da sub-seção conteúdo da sub-seção conteúdo da sub-seção.

Referências

[1] J. Bond. *The Enterprise Cloud: Best Practices for Transforming Legacy IT*. O'Reilly Media, 2015.

[2] J. Doe. Testing bibliography.

[3] A. Fisher. *Multi-Cloud Patterns, Best Practicies Framework: Focusing on Deployment Management, Fault Tolerance, Security, Self-Healing, Cost Efficiency and Vendor Neutrality*. O'Reilly Media, 2018.

[4] K. Morris. *Infrastructure as Code: Managing Servers in the Cloud*. O'Reilly Media, 2016.

[5] M. Zann. Testing bibliography again.