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

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 aprisionado à fornecedores ou soluções fechadas (*vendor lock in*). Entretanto, 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 "over-provision"sua infraestrutura virtualizada e isto rezulta 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 aborda 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.

Este estudo descreve como podemos desenhar padrões de provisionamento em ambiente *multi-cloud* que ofereçam:

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.

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 oversubscrição 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

2.1.1. Initial Multi-Cloud General Deplyment Fault Tolerance Challenges

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

2.1.3. Cost Efficiency Problems

2.1.4. Security Problems

2.1.5. Application Deployment in Multi-Cloud Environment

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

2.2.1. title

2.3. Cost efficiency patterns

2.3.1. title

2.4. security related patterns

2.4.1. title

2.5. multi-cloud control plane framework

3. experimental validation

4. 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.