Mobilidade de microserviços em centros de dados suportada por Software Defined Networking

Microservices mobility in datacenters supported by Software Defined Networking

Daniel Jorge Valente
Department of Informatics
University of Minho
Braga, Portugal
djvalente@gmail.com

Pedro Sousa
Centro Algoritmi, Department of Informatics
University of Minho
Braga, Portugal
pns@di.uminho.pt

Resumo — **A crescente adoção de arquiteturas de software baseadas em microserviços, e consequentemente a proliferação da utilização de *containers* em ambientes com infraestruturas híbridas (centros de dados locais e operadores de clouds públicas), lança vários desafios do ponto de vista da integração deste novo paradigma nas soluções existentes. Neste contexto, este artigo centra-se inicialmente nos desafios de integrar uma plataforma de orquestração de *containers* (Kubernetes) com soluções já existentes no centro de dados, nomeadamente com a componente de rede (Cisco ACI – *Application Centric Infrastructure*), com o balanceador de carga (F5) e armazenamento (NFS). Num segundo momento é também abordada a integração com plataformas de orquestração alojadas numa cloud pública (Microsoft Azure). Esta integração vai permitir a movimentação de *containers* do centro de dados local para a cloud sem necessitar de reconfigurar a infraestrutura de rede ou as políticas de segurança em vigor.**

Palavras Chave - ACI, Cloud, Kubernetes, Microserviços, SDN.

Abstract — **The growing adoption of software architectures based on microservices, and consequently the proliferation of the use of *containers* in environments with hybrid infrastructures (local data centers and public cloud providers), poses several challenges in terms of integrating this new paradigm into already existing solutions. In this context, this article initially focuses on the challenges of integrating a *container* orchestration platform (Kubernetes) with existing solutions in the data center, namely: network infrastructure (Cisco ACI – *Application Centric Infrastructure*), load balancing (F5) and storage (NFS). In a second stage, this paper also addresses the issue of integration with orchestration platforms hosted in a public cloud (Microsoft Azure). This integration will allow the movement of containers from the local data center to the cloud without having to reconfigure the network infrastructure or the current security policy.**

Keywords - ACI, Cloud, Kubernetes, Microservices, SDN.

I. INTRODUÇÃO

As aplicações monolíticas são normalmente constituídas por vários componentes, mas que na prática são tratados como um só. Assim, alterações num componente obriga à reinstalação da aplicação completa, além disso verifica-se que a falta de limites e de mecanismos formais de integração entre os diferentes componentes aumenta a complexidade da solução e muitas vezes leva à deterioração da qualidade do sistema como um todo. Estes e outros problemas [1] forçam-nos a equacionar a divisão das atuais aplicações monolíticas em componentes mais pequenos e independentes, aos quais damos o nome de microserviços. De forma a conseguir isolar corretamente os ambientes de cada microserviço recorremos a uma tecnologia denominada de *Linux Containers* [2].

À medida que o número de *containers* aumenta na infraestrutura, a tarefa de os gerir convenientemente fica bastante complexa [3] o que rapidamente justifica a utilização de plataformas de orquestração como o *Kubernetes* [4]. Estas plataformas permitem criar uma abstração relativamente à infraestrutura física, simplificando o desenvolvimento, a implementação e a gestão [5]. No entanto, essa abstração só é possível se houver uma total integração com as soluções já existentes no centro de dados (e.g. rede, balanceadores, armazenamento). Numa cloud pública essa integração é assegurada pelo provedor do serviço, mas em centros de dados locais é necessário que as equipas internas da organização consigam assegurar essa integração. Relativamente à componente de rede propriamente dita, tendo em conta que as clouds públicas e os próprios orquestradores de *containers* já implementam soluções baseadas em software, pretende-se estender [6] as funcionalidades e os controlos da solução *Software Defined Networking* (SDN) existente no centro de dados local (neste caso, o Cisco ACI [7]) para todo o ecossistema, conseguindo desta forma a mobilidade pretendida.

Este documento é composto por 5 secções assim organizadas: a secção I apresentou uma introdução ao trabalho desenvolvido contextualizando a sua relevância, a secção II descreve a arquitetura e o problema que pretendemos resolver, na secção III apresentamos mais algum detalhe sobre a solução implementada, a secção IV demonstra

os resultados obtidos e finalmente na secção V são apresentadas as conclusões do trabalho desenvolvido.

II. ARQUITETURA DA SOLUÇÃO PROPOSTA

Uma das grandes vantagens que frequentemente se associa à utilização de *containers* é precisamente a possibilidade, do ponto de vista da infraestrutura de computação, de movimentar os *workloads* entre locais [8]. No entanto é necessário que a componente de rede se adapte de forma dinâmica a estas movimentações, mantendo políticas e garantindo que o utilizador final não tenha qualquer perceção sobre o local onde a aplicação está a ser executada.

A empresa onde se contextualiza este trabalho (SONAE MC), tendo concluído a primeira fase de transformação da rede dos seus centros de dados locais, através da implementação da solução Cisco ACI numa arquitetura *multi-pod* [9], obteve a capacidade de movimentar máquinas virtuais entre centros de dados sem qualquer preocupação com as definições de rede (endereçamento, rotas), ou com a politica de controlo de acessos. No entanto, uma vez que têm também o objetivo de acompanhar a evolução aplicacional em curso, adoção de *containers* nos centros de dados locais e em cloud publicas, pretende-se avaliar se esta mobilidade será também possível neste contexto, isto é, movimentar *containers* alojadas num cluster no centro de dados local para um cluster numa cloud pública, mantendo os endereços de rede e as políticas de segurança. A Figura 1 representa o cenário final pretendido.

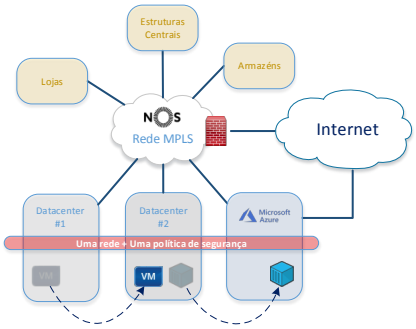

Figura 1. Representação da mobilidade entre locais considerados

Inicialmente será implementado no centro de dados principal um cluster de Kubernetes, com três *Workers* e um *Master* [10] devidamente integrado na rede do centro de dados. Para isso, utilizaremos o *Container Network Interface* (CNI) disponibilizado pela própria Cisco, permitindo que através do controlador SDN seja efetuada a gestão da rede e o controlo de acessos das máquinas físicas, virtuais e também dos containers. A Figura 2 representa a arquitetura nesta fase inicial da implementação.

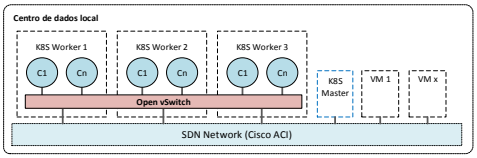

Figura 2. Representação da integração de rede K8S com ACI

Além da integração ao nível da rede, será necessário garantir que o cluster de Kubernetes consegue utilizar os serviços de armazenamento de informação local, e também tirar partido da solução de balanceamento de carga existente no centro de dados. Na Figura 3 é possível verificar a arquitetura no momento em que integramos com o serviço de armazenamento baseado em *Network File Storage* (NFS) e com o balanceador de carga existente do fabricante F5. Estas integrações permitirão ao Kubernetes orquestrar toda a criação/gestão dos recursos necessários à execução das aplicações.

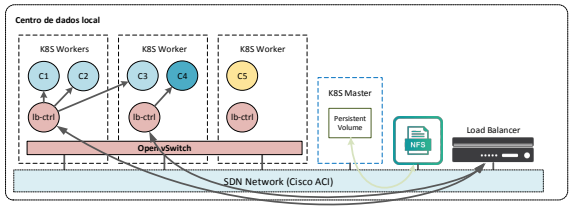

Figura 3. Integração Kubernetes com centro de dados local

Finalmente, integraremos o ambiente local descrito acima, com a solução *Platform as a Service* (PaaS) da cloud da Microsoft denominada *Azure Kubernetes Service* (AKS). O facto de não conseguirmos controlar a componente de rede do AKS, por se tratar de um serviço gerido, obriga a configurar a rede de forma a que os *containers* utilizem endereçamento da própria rede virtual do Azure, o que nos levou à eventual possibilidade de integrar o ACI diretamente com a rede do Azure, e desta forma endereçar os desafios propostos. A Figura 4 representa desta forma a integração pretendida.

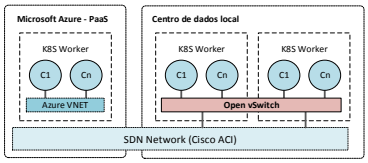

Figura 4. Integração do Kubernetes com solução PaaS (AKS)

III. BANCADA DE TESTES

Como mencionado anteriormente, nesta secção iremos detalhar um pouco mais o trabalho em torno da implementação da solução proposta. Iniciaremos por salientar os principais desafios relacionados com a instalação do cluster de Kubernetes, abordaremos os passos necessários à integração do ACI com o Kubernetes local, e por último descreveremos como foi feita a integração do ACI com a cloud Azure.

A. *Instalação do cluster Kubernetes*

Existem inúmeras formas [11] de implementar um cluster de Kubernetes, pelo que iremos apenas abordar os dois relevantes para a nossa solução.

- Num centro de dados com inúmeras máquinas
- Em clouds publicas utilizando clusters geridos

Atualmente existem duas opções para criar um cluster em Azure (AKS), via linha de comandos ou via o portal de gestão da cloud da Microsoft. Via linha de comandos é possível parametrizar mais algumas opções, nomeadamente qual o CNI a utilizar. No entanto, como o que pretendemos é que os *containers* utilizem a rede nativa do Azure, temos que forçosamente utilizar o Azure-CNI, permitindo utilizar o portal durante todo o processo, necessitando apenas de definir, para além de alguns aspetos gerais do Azure, a versão do Kubernetes que pretendemos utilizar, a quantidade e os recursos computacionais (processador e memória) dos nós de trabalho, o método de escalabilidade do cluster, configurações de autenticação e monitorização e finalmente de rede. Sobre a componente de rede, uma vez que estamos a usar o Azure-CNI, apenas parametrizamos as redes IP que vão ser utilizadas pelo cluster. Definimos a gama de endereços utilizada pelos próprios *pods (Cluster subnet*), o endereçamento para serviços internos *(Kubernetes services)* e o endereçamento de interligação entre a rede dos worker nodes e a rede de serviços *(Docker Bridge)*.

Os passos necessários à instalação inicial de um cluster de Kubernetes em máquinas virtuais Linux, está já bastante documentado [10], pelo que nos iremos apenas focar nos desafios de integração com os serviços já existentes no atual centro de dados.

1) *Integração com balanceador de carga (F5)*

O fabricante F5 disponibiliza dois modos de integração com clusters *Kubernetes*, sendo que a principal diferença está, mais uma vez, na componente de rede. A utilização do modo *Cluster* pressupõe que pretendemos integrar o balanceador diretamente com a rede interna do Kubernetes, seja através da criação de um *overlay* com VXLAN, ou através da utilização de protocolos de *routing* dinâmico (BGP), com vista a permitir uma conectividade direta entre o F5 e os *containers* dentro do cluster. Esta abordagem, apesar de apresentar inúmeras vantagens [12] teve de ser descartada, uma vez que obriga a utilização de CNIs específicos, comprometendo a integração com a solução de rede existente Cisco ACI.

O modo *Nodeport* não necessita de qualquer integração ao nível da rede, o que simplifica a solução e a sua implementação. Este modo é considerado inferior ao modo *Cluster* pelo facto de não permitir que o F5 tenha visibilidade direta sobre os *containers*, no entanto como a nossa solução pressupõe a integração com a solução Cisco ACI, que nos vai trazer precisamente esta capacidade, esta limitação não será relevante para a solução final que pretendemos implementar.

2) *Utilização de armazenamento persistente*

Os *containers* são efémeros por natureza, isto é, um container que "desaparece" é automaticamente substituído por um novo, conferindo uma enorme resiliência às aplicações, mas impossibilitando a utilização dos sistemas de ficheiros internos. Para endereçar este desafio o Kubernetes implementa um mecanismo de abstração ao nível do armazenamento que designamos por *volumes* [13]. Existe uma grande variedade de tipos de *volumes* que mapeiam diretamente com protocolos ou tipos de armazenamento suportados pelo Kubernetes, no entanto para a nossa solução apenas utilizamos armazenamento do tipo NFS. O primeiro passo é a criação de um *Persistent Volume*, que define o endereço do servidor, o caminho para a partilha de rede, o tipo de armazenamento e a política de retenção, posteriormente é necessário criar um *Persistent Volume Claim*, que definirá o tipo de acesso (leitura, leitura/escrita, etc.) e a capacidade de armazenamento. Após termos estes objetos criados no cluster, é apenas necessário que o *container* que pretende usar o armazenamento persistente o referencie no momento da sua criação.

3) *Integração da solução SDN da Cisco com Kubernetes*

A Cisco disponibiliza uma ferramenta que apoia a preparação da infraestrutura já existente no centro de dados local para a integração com plataformas externas de gestão de *containers*. Esta ferramenta, denominada *acc-provision*, além de preparar a integração cria também as definições aplicacionais que posteriormente aplicaremos ao Kubernetes para instalar o CNI. Este comando recebe como parâmetro um ficheiro YAML que detalha a informação da infraestrutura e retorna um novo ficheiro YAML que servirá de input para a parametrização do CNI no próprio cluster de Kubernetes. Na Figura 5 podemos verificar que o processo cria automaticamente objetos (*Application Profile*, com *End Point Groups* e *Contracts*) no ACI que permitirão a comunicação entre *containers*.

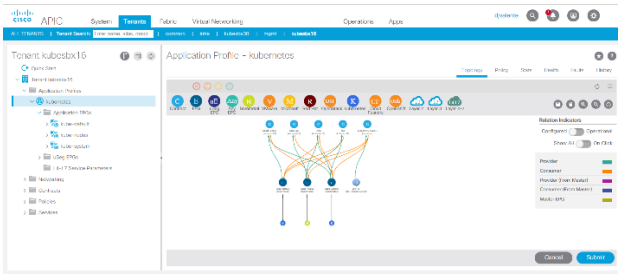

Figura 5. Verificação no ACI dos objetos criados para a integração

Após preparar o ACI é necessário garantir que os nós que compõem o *cluster* estão também corretamente parametrizados, nomeadamente a componente de rede. Cada nó físico terá de ter duas placas de rede, sendo que a primeira será apenas para gestão dos próprios nós, e a segunda deverá estar configurada para suportar *Virtual Private Local Area Networks* (VLAN). Uma VLAN será utilizada para garantir a extensão do *overlay*, isto é, o switch virtual do cluster será integrado diretamente no *fabric*, como se de um switch físico se tratasse. A outra VLAN será utilizada pelo próprio Kubernetes.

A integração fica concluída com a instalação do ACI-CNI no cluster, passando como input o ficheiro YAML criado pela ferramenta *acc-provision*.

B. *Integração da solução SDN da Cisco com Azure*

A integração do ACI com a cloud publica da Microsoft é uma funcionalidade apenas disponível a partir da versão 4.2, lançada no final de setembro de 2019. Esta funcionalidade tem como principal objetivo estender o controlo que o ACI tem sobre a rede do centro de dados local para a rede do Azure, permitindo replicar o modelo operativo. Esta integração obriga à adição de três novas componentes à arquitetura, nomeadamente: *i)* Multi-site Orchestrator (MSO); *ii)* Cloud APIC (CAPIC) e *iii)* Conectividade entre o centro de dados e o Azure.

O MSO é o ponto central de gestão que nos vai permitir obter uma visão unificada, e um único ponto de controlo de toda a infraestrutura. O CAPIC é a componente que vai gerir toda a política de rede da cloud, recorrendo ao modelo de políticas criado no MSO. O controlador instalado localmente no centro de dados instala a política diretamente nos *switchs* físicos, ao passo que o CAPIC do Azure, primeiro traduz a política criada no MSO para conceitos/objetos do Azure permitindo assim a sua aplicação na infraestrutura cloud.

A conectividade entre os dois locais (centro de dados local e a cloud Azure) pode ser decomposta em duas componentes: *underlay* e *overlay*. A componente *underlay* baseia-se em tuneis IPSec sobre a Internet. Os tuneis são estabelecidos entre um equipamento no centro de dados e dois *routers* CSR1000V instalados em Azure. Os routers da cloud, além de assegurarem a conectividade entre os sites, vão também ser uma peça fundamental para o dinamismo que se pretende obter relativamente ao endereçamento IP, isto é, será necessário garantir que redes criadas dentro do Azure são anunciadas para o centro de dados, o que tendo em conta que não conseguimos estabelecer adjacências de *routing* diretamente com o Azure, obriga-nos a recorrer ao objeto Virtual Network Gateway (VNG), que é precisamente o objeto que nos vai possibilitar este dinamismo, obrigando-nos a criar tuneis IPSec entre os CSR1000V e os VNG dentro da cloud. A componente *overlay* utiliza BGP-EVPN para a componente de controlo (control plane) e encapsulamento VXLAN para a componente de transporte (data plane). Na Figura 6 são visíveis todos os componentes que compõem a arquitetura da interligação.

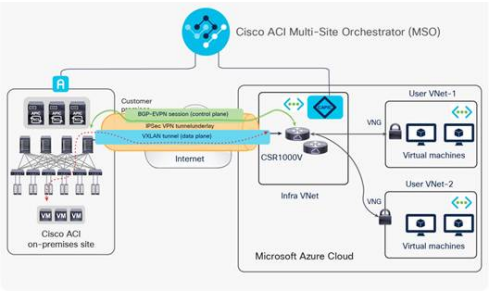

Figura 6. Arquitetura da ligação do ACI ao Azure

IV. TESTES E RESULTADOS

Nesta secção são apresentados os casos de uso utilizados para validar a solução implementada. Inicialmente demonstramos a integração do cluster Kubernetes com as soluções utilizadas no centro de dados local (rede e balanceador de tráfego) verificando que a própria política de segurança está incluída na componente de rede. Finalmente, comprovamos que a integração do ACI com a cloud Azure permite controlar *containers* alojadas em AKS da mesma forma que controla máquinas virtuais criadas em Azure.

A. *Integração com soluções existentes no centro de dados*

1) *Integração com a componente de Rede (Cisco ACI)*

Para validar o correto sincronismo entre o Kubernetes e o ACI, procedemos à instalação de uma aplicação de exemplo disponibilizada na página oficial do Kubernetes. Esta aplicação é composta por uma camada para *front-end* composta por três *containers*, uma *Redis Cache* composta por um *container* para *master* e dois para *slave*. Após a sua correta instalação no Kubernetes é possível constatar que a informação de todas as componentes surgiu no ACI, conforme vemos na Figura 7.

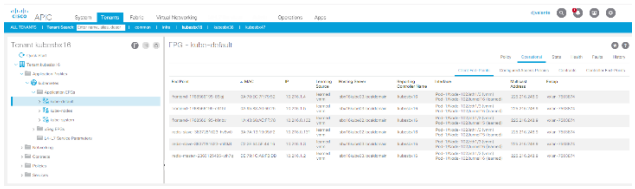

Figura 7. Validação no ACI da integração com o cluster

Para alem da visibilidade unificada, é também possível utilizar o ACI para controlar os acessos entre diferentes *containers* dentro do cluster, ou entre *containers* e recursos fora do cluster, como por exemplo máquinas virtuais. Na prática os *containers* ao serem criados são associados a *End Point Groups* do ACI, como se de máquinas virtuais se tratassem. Desta forma as funcionalidades de micro segmentação existentes no ACI podem ser também utilizadas para controlar os fluxos de e para os *containers* existentes no cluster. Numa aplicação com várias camadas, antes de criarmos os *containers* no cluster temos de criar no ACI os EPGs e os contratos que descriminam com quem pode "falar" cada grupo de *containers*, conforme vemos na Figura 8, finalmente no momento da criação do *container* é necessário acrescentar a que EPG este irá pertencer, ficando automaticamente associado à política criada no ACI.

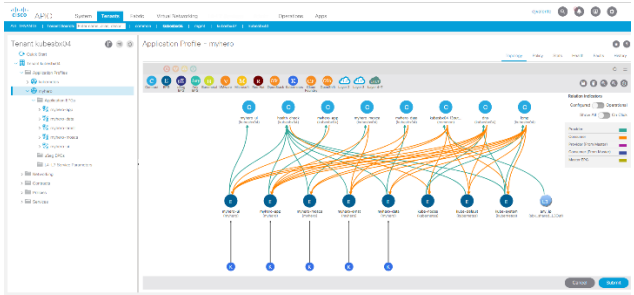

Figura 8. Application Profile da aplicação de validação

2) *Integração com balanceador de carga*

Além da já referida intenção de balancear tráfego para *containers* através da utilização do balanceador F5 já existente, verificamos que através das funcionalidades de *Policy Based Routing* (PBR) nativas do ACI é possível balancear tráfego utilizando apenas a componente de rede. Esta segunda opção, por se tratar de uma abordagem bastante mais simples pode ser interessante em algumas situações.

A integração do balanceador F5 no cluster obriga a algumas parametrizações do ponto de vista de autenticação, uma vez que na prática vai existir um *container*, que será responsável por monitorizar todo o cluster, interagindo automaticamente com o balanceador fora do cluster quando necessário. Podemos verificar o correto funcionamento da integração através da criação de um ambiente de teste disponibilizado pela própria F5. Na prática estamos a pedir ao Kubernetes que crie três *containers* e que os associe a um balanceador. Uma vez que o cluster está integrado com o F5, o balanceador será automaticamente criado no equipamento externo ao cluster, sem qualquer intervenção manual, como podemos ver na Figura 9.

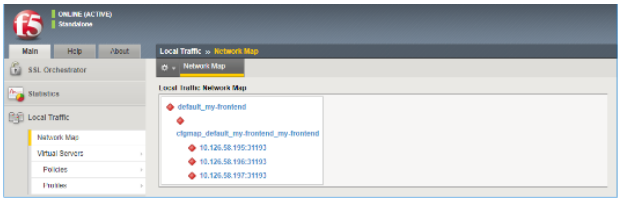

Figura 9. Criação de balanceador no F5

Em clusters integrados com ACI, mas sem integração com nenhum balanceador de carga específico, o próprio ACI vai assumir essa função recorrendo à já referida funcionalidade de PBR. Na prática a mesma indicação que no exemplo anterior despoletava a criação de um balanceador no F5, nestes casos despoletará a criação de várias configurações no ACI: *i)* Alteração ao objeto L3Out - permitir a entrada do tráfego no fabric; *ii)* Criação de um *Service Graph* – define o tipo de tráfego que é permitido (*filter)* e a política de redireccionamento (*policy based routing*) e *iii)* Associação do novo *Service Graph* ao contrato que permite a comunicação entre os EPGs. A Figura 10 pretende representar todo o processo de entrada de tráfego no cluster quando o cluster apenas está integrado com o ACI.

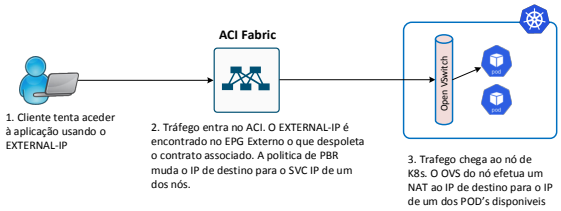

Figura 10. Representação do processo de entrada de tráfego

B. *Integração do ACI com a cloud Azure*

No sentido de validar se a componente *underlay* está corretamente operacional, podemos validar se no router do centro de dados existem os dois tuneis IPSec com os routers em Azure, e se o protocolo de *routing* que vai garantir que as redes da cloud sejam conhecidas no centro de dados está corretamente configurado. A Figura 11 mostra o output dos comandos que permitem precisamente validar os pressupostos acima descritos.

```
CSRv1#sh ip int bri
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       10.240.21.73    YES NVRAM  up                    up
GigabitEthernet2       10.240.254.2    YES NVRAM  up                    up
GigabitEthernet3       192.168.3.1     YES NVRAM  up                    up
Loopback10             unassigned      YES unset  up                    up
Loopback101            unassigned      YES unset  up                    up
Loopback1000           88.157.227.103  YES NVRAM  up                    up
Loopback1094           111.111.111.100 YES NVRAM  up                    up
Tunnel1                33.3.1.2        YES NVRAM  up                    up
Tunnel2                33.3.1.6        YES NVRAM  up                    up
CSRv1#sh ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
1.1.1.1           0   FULL/  -        00:00:37    192.168.3.2     GigabitEthernet3
13.74.181.86      0   FULL/  -        00:00:36    33.3.1.5        Tunnel2
168.63.46.2       0   FULL/  -        00:00:36    33.3.1.1        Tunnel1
CSRv1#
```

Figura 11. Validação dos tuneis e do protocolo de routing

Para validar a componente de *overlay*, temos de verificar o estado da EVPN diretamente num dos switchs do centro de dados. Na Figura 12 podemos confirmar o aparecimento dos dois equipamentos da cloud Azure.

```
SPINE201# show bgp l2vpn evpn summary  vrf all
BGP summary information for VRF overlay-1, address family L2VPN EVPN
BGP router identifier 192.168.1.101, local AS number 65500
BGP table version is 10318, L2VPN EVPN config peers 5, capable peers 3
151 network entries and 172 paths using 30808 bytes of memory
BGP attribute entries [14/2072], BGP AS path entries [0/0]
BGP community entries [0/0], BGP clusterlist entries [0/0]

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
22.2.0.1        4 65498       0       0        0    0    0     2w2d Idle
22.2.0.2        4 65498       0       0        0    0    0     2w2d Active
33.3.0.1        4 65499      12      12    10318    0    0 00:06:05 5
33.3.0.2        4 65499      12      12    10318    0    0 00:06:15 5
192.168.2.102   4 65500   23277   28808    10318    0    0     2w2d 26
SPINE201#
```

Figura 12. Validação do overlay

C. *Utilização do MSO para gerir fluxos entre máquinas virtuais e containers em Azure*

Conforme referimos anteriormente é no componente MSO que está centralizada a visão e o controlo unificado, pelo que se pretendemos interagir com a cloud Azure através da solução ACI, teremos de recorrer à interface do próprio MSO. A Figura 13 mostra um EPG, denominado SHARED_TEST2, criado no MSO devidamente associado ao Azure.

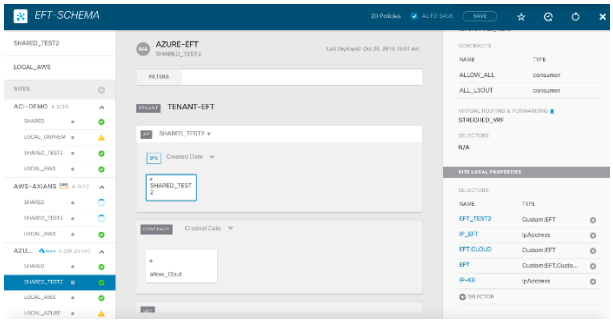

Figura 13. Criação de EPG via MSO

Ao concluir esta configuração as redes IP utilizadas no interior da cloud, são anunciadas dinamicamente para o centro de dados local, surgindo nas tabelas de *routing* dos switchs do centro de dados. Outra configuração importante visível na Figura 13 são os *Selectors*. Os *Selectors* definem as regras

pelas quais as máquinas virtuais vão ser colocadas nos EPGs do ACI, ficando automaticamente sujeitas à política implementada para aquele EPG. O recomendado pela Cisco é associar uma marcação *(tag)* à máquina que corresponda a um dos *Selectors* do EPG, e o CAPIC automaticamente vai aplicar a política existente. No entanto, apesar de termos verificado esse comportamento para máquinas virtuais, quando se trata de *containers* criados em AKS, não é possível adicionar *tags* diretamente aos *containers* o que complica a integração pretendida. Com vista a contornar esta limitação, adicionamos ao EPG um novo *Selector* que associa ao EPG qualquer objeto de Azure baseado no seu endereço IP. Ao colocarmos o IP propriamente dito do *container* criado dentro do AKS, foi possível comunicar com o *container* tendo como origem uma máquina alojada no centro de dados local.

Analisando a informação no APIC do centro de dados local, verificamos que o endereço IP do container foi automaticamente adicionado ao EPG SHARED_TEST2 conforme pretendíamos, permitindo desta forma a comunicação entre os dois mundos, independentemente de se tratar de uma máquina virtual ou de um *container*.O facto de verificarmos que é possível associar containers de AKS a EPGs específicos do ACI, confirma que será possível termos containers num cluster de Kubernetes local, a utilizar o ACI como balanceador de tráfego, e quando pretendido criar um novo container em AKS e associa-lo a qualquer componente da aplicação, visto que o endereçamento que o cliente final conhece é agnóstico da localização real dos containers.

V. Conclusões e Trabalho futuro

Este artigo apresenta uma solução para uma empresa que pretende adotar arquiteturas de software baseadas em microserviços, necessitando para isso de gerir *containers* em operadores de clouds públicas e nos seus centros de dados locais. A empresa pretende também conseguir movimentar containers entre os referidos locais, sem necessitar de alterar as definições de rede ou as políticas de segurança em vigor. Inicialmente já era possível movimentar máquinas virtuais entre centros de dados locais, através da solução SDN em produção nos centros de dados, mas a empresa pretendia estender esta flexibilidade para containers alojados em clusters de Kubernetes e para as clouds publicas. Para isso foi necessário integrar o cluster de Kubernetes com as soluções existentes. Na componente de rede integramos com a solução Cisco ACI, na componente de balanceamento integramos com a solução F5 e descrevemos a possibilidade de balancear tráfego diretamente na infraestrutura de rede, e ainda descrevemos a integração com soluções de armazenamento baseados em NFS. Num segundo momento, descrevemos a arquitetura que suportará a integração da solução ACI com a cloud Azure. Por fim, alguns testes ilustrativos foram apresentados por forma a corroborar a eficácia da solução.

Apesar de se planear, no ano de 2020, implementar todas as integrações descritas neste artigo, vai ser necessário aprofundar a integração com diferentes soluções de armazenamento. Será relevante para este tema analisar o trabalho recente desenvolvido em torno do *plug-in Container Storage Instance*, dado que pretende criar precisamente um grau de abstração adicional facilitando a utilização de múltiplas soluções de armazenamento de forma transparente.

Agradecimentos

This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020

Referências Bibliográficas

[1] Z. Avidan, H. Otharsson, "Accelerating the Digital Journey from Legacy Systems to Modern Microservices", CreateSpace Ind. Pub. Platform, 2018.

[2] D. Merkel, “Docker: Lightweight Linux Containers for Consistent Development and Deployment,” Linux Journal, 2014.

[3] T. Giovanni, B. et al., “An architecture for self-managing microservices,” *Proceedings of the 1st International Workshop on Automated Incident Management in Cloud,* pp. 19-24, 2015.

[4] B. Burns et al., “Borg, Omega, and Kubernetes,” *in ACM Queue vol. 14, no. 1,* pp. 70-93, 2016.

[5] M. Luksa, Kubernetes in Action, New York: Manning Publications (1st Edition), 2018.

[6] B. Pfaff, J. Pettit, T. Koponen, “Extending Networking into the Virtualization Layer,” Proc. of *HotNets,* 2014.

[7] “Application Centric Infrastructure (ACI), the policy driven data centre,” [Online]. Available: https://pt.slideshare.net/CiscoCanada/application-centric-infrastructure-aci-the-policy-driven-data-centre.

[8] V. L. a. C. P. Davide Taibi, “Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation,” *IEEE Cloud Computing,* vol. 201, nº 7, 2017.

[9] “Cisco Application Centric Infrastructure Multipod Configuration White Paper,” [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-739714.html .

[10] “K8S Getting started,” [Online]. Available: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/.

[11] “Production-Grade Container Orchestration,” [Online]. Available: http://kubernetes.io.

[12] “BIG-IP Controller Modes,” [Online]. Available: https://clouddocs.f5.com/containers/v2/kubernetes/kctlr-modes.html.

[13] “K8s Volumes,” [Online]. Available: https://kubernetes.io/docs/concepts/storage/volumes/.