Rastreabilidade de alocação e desalocação de contêineres usando Docker Swarm com base em blockchain Hyperledger consorciado

Marco A. Marques
UDESC
marco.marques@edu.udesc.br

Maurício A. Pillon
UDESC
mauricio.pillon@udesc.br

Guilherme P. Koslovski
UDESC
guilherme.koslovski@udesc.br

Charles C. Miers
UDESC
charles.miers@udesc.br

ABSTRACT

Container-based virtualization technology enables dynamic allocation, addressing needs of scalability and fault tolerance. This feature provides flexibility in the use of computing resources, but turns monitoring a challenge due to the large flow of calls and allocations. Recording these operations in a Blockchain allows to not only audit the resources employed but also the chronology analysis of the performed operations. In addition, the use of blockchain distributes the credibility of record veracity between provider, end user, and developer of the container-based solution.

KEYWORDS

Container, Blockchain, Hyperledger, Docker

1 INTRODUÇÃO

A arquitetura de microsserviços permitiu aumentar o desempenho de aplicações, fragmentando-as em partes independentes de desenvolvimento, versionamento, provisionamento e escalabilidade. Contêineres são considerados o padrão de microsserviços na nuvem [1], pela sua rapidez e facilidade na alocação, gerenciamento escalável e resiliência [2].

Tal fato conduziu ao surgimento de plataformas de orquestração de contêineres, projetadas para gerenciar a implantação de aplicações conteinerizadas em aglomerados de larga escala, sendo capazes de executar centenas de milhares de trabalhos em diversas máquinas [3]. Tais características tornaram o monitoramento deste ambiente descentralizado um desafio, devido ao elevado número de recursos e consequentemente de informações geradas e trafegadas. Sobretudo, a análise de rastros de aplicações largamente distribuídas (microsserviços) para encontrar problemas ou realizar otimizações requer um sequenciamento cronológico das atividades realizadas, um desafio comum em ambientes distribuídos. Desta forma, tornou-se necessária uma ferramenta para o registro dos eventos relacionados ao ciclo de vida dos contêineres, permitindo a auditabilidade da utilização dos recursos computacionais.

Este trabalho apresenta o uso da tecnologia Blockchain como repositório seguro e auditável de registro de eventos de criação, pausa e destruição de contêineres. Um fato que contribui para o uso de Blockchains é de estes serem um livro-razão distribuído, que consiste em uma cadeia de blocos interligados através do uso de um ou mais *hashes* (dependendo da implementação) do bloco anterior [4]. Além disso, os Blockchains são bancos de dados compartilhados com considerável resistência a violações, nos quais todos os participantes podem acrescentar e ler transações, mas o controle total sobre a cadeia é consideravelmente dificultado pela natureza dos mecanismos de consenso tipicamente empregados [5]. Em situações nas quais uma autoridade central possui o controle sobre os recursos de informação, a confiança depositada nesta autoridade para manter registros corretos e precisos é baixa, pois não existe mecanismo que permita que entidades externas possam verificar a validade dos registros. A tecnologia Blockchain soluciona o problema de confiança por manter registros e transações dos recursos de informação em uma rede distribuída [6]. A implementação proposta prevê a utilização de uma Blockchain composta pelas entidades envolvidas na criação e manutenção das aplicações, serviços e ambientes para que, desta forma, haja consenso entre as partes quanto à validade dos eventos registrados.

2 SOLUÇÃO PROPOSTA

2.1 Definições e Detalhamento

O ciclo de vida dos contêineres é composto pelos seguintes estados: em execução, pausado, parado ou deletado. A transição entre estes estados é realizada via linha de comando ou pelo orquestrador, e está representada na Figura 1.

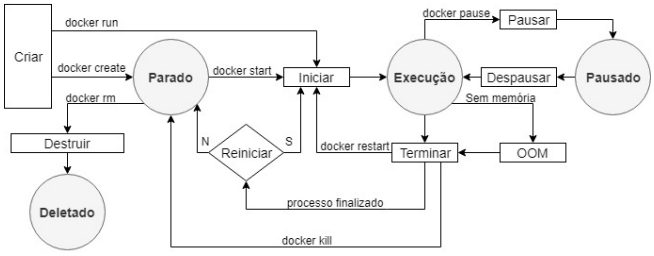

Figura 1: Ciclo de vida contêiner Docker [7]

Existem diferentes recursos para realizar a coleta de eventos de contêineres e serviços do ambiente Docker. A interface de linha de comando da *engine* Docker oferece o recurso *Docker events*, que

apresenta em tempo real os eventos relacionados aos contêineres, imagens, volumes, serviços, nós, configurações, além do próprio *daemon* Docker [7]. A solução proposta será desenvolvida em linguagem Python, utilizando a biblioteca docker-py, que permite que aplicações comuniquem-se com a *engine* Docker e realizem todas as ações disponíveis na Docker CLI [8], e a biblioteca Fabric-SDK, que oferece recursos para comunicação e interação com a Blockchain. A aplicação é composta de dois agentes (Figura 2) para comunicação com a *engine* Docker e com a Blockchain, respectivamente. O agente de coleta é o responsável pela coleta dos eventos em tempo real e armazenamento em repositório temporário para que o agente de transação utilize-os para gerar as transações e enviar para o nó validador da Blockchain.

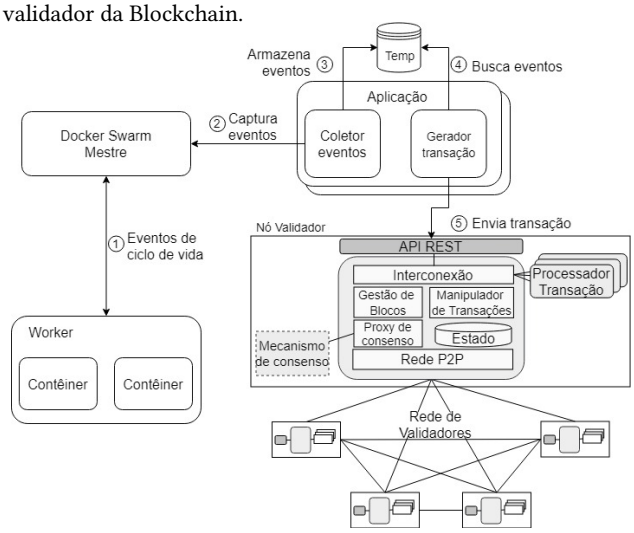

Figura 2: Modelo de solução proposta.

O foco do trabalho é a coleta e o registro dos eventos *start, stop, pause, unpause e restart*. A aplicação será implementada no ambiente como um serviço executado em todos os nós tipo *Worker* do orquestrador Docker Swarm. A Blockchain utilizada nesta proposta será o modelo Hyperledger Fabric, por atender a critérios como possibilidade de operar em modo consórcio e permitir a implementação de mecanismo de consenso que não demande alto custo computacional de processador, o *Practical Byzantine Fault Tolerance* (PBFT).

2.2 Ambiente de Execução

O protótipo experimental da proposta está sendo desenvolvido para a plataforma de virtualização em contêineres Docker, sendo utilizado o Docker Swarm como orquestrador. A escolha deve-se à relevância da ferramenta nas implementações de contêineres [9].

O orquestrador Docker Swarm possui dois tipos de nós: Mestre e *Worker* (Figura 2). Os nós Mestres são responsáveis por manter o estado do aglomerado, agendar serviços e responder aos chamados da API HTTP. Através da utilização do mecanismo de consenso Raft, os mestres mantém a consistência do orquestrador e serviços em execução. O Raft é um algoritmo que gerencia um *log* replicado e permite obter o consenso elegendo um líder responsável pelo gerenciamento deste *log*. O líder recebe, dos clientes, entradas a serem adicionadas ao *log* e as replica para os outros servidores, informando quando é seguro aplicá-las. Este algoritmo mantém-se completamente funcional enquanto a maioria dos seus nós estiverem operacionais e puderem comunicar-se com os clientes. Tendo em vista a necessidade de maioria para obtenção de consenso, sugere-se a utilização de número ímpar de nós tipo Mestre, com número máximo sugerido de 7 elementos [10]. A solução proposta selecionará o número de Mestres dentro do escopo apresentado, de acordo com a carga de trabalho da plataforma. A Blockchain de testes será composta por 8 nós validadores (Figura 3 representa o modelo simplificado).

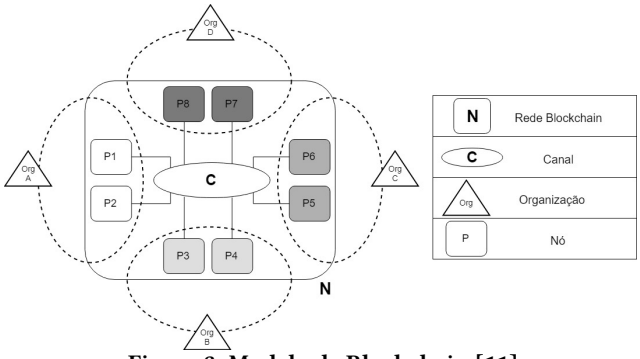

Figura 3: Modelo de Blockchain [11]

Neste modelo a Blockchain é composta por 4 organizações, possuindo 2 nós validadores cada. A arquitetura da solução visa garantir o consenso entre elas.

CONSIDERAÇÕES & TRABALHOS FUTUROS

O presente trabalho estuda a aplicação de Blockchain para a monitoração de informações de microsserviços, aproveitando os benefícios do livro-razão distribuído para auxiliar o processo. O protótipo experimental baseado em Docker Swarm e Hyperledger Fabric será implementado seguindo as especificações elencadas, com o objetivo de testar a aplicação responsável pela coleta e registro dos eventos. Considerando que em um ambiente de produção podem ocorrer milhares de eventos por segundo, deve ser avaliado também como a aplicação lida com tal demanda, afim de garantir que não ocorrerão eventos sem o devido registro na Blockchain.

Agradecimentos: Os autores agradecem o apoio do LabP2D/UDESC e a FAPESC.

REFERÊNCIAS

[1] S. Vaucher, R. Pires, P. Felber, M. Pasin, V. Schiavoni, and C. Fetzer, "Sgx-aware container orchestration for heterogeneous clusters," in *Proc. of 38th ICDCS*, IEEE 38th ICDCS, 2018.
[2] S. Newman, *Building Microservices.* OReilly, 1 ed., 2015.
[3] M. A. Rodriguez and R. Buyya, "Container-based cluster orchestration systems: Taxonomy and future directions," *CoRR*, vol. abs/1807.06193, 2018.
[4] N. El Ioini and C. Pahl, "Trustworthy orchestration of container based edge computing using permissioned blockchain," in *Proc. of 5th IOTSMS*, 10 2018.
[5] S. Helmer, M. Roggia, N. El Ioini, and C. Pahl, "Ethernitydb – integrating database functionality into a blockchain," in *Proc. of European Conference on Advances in Databases and Information Systems*, pp. 37–44, 09 2018.
[6] A. Sutton and R. Samavi, "Blockchain enabled privacy audit logs," in *Proc. of ISWC*, pp. 645–660, 10 2017.
[7] D. Docs, "Docker events." https://docs.docker.com/engine/reference -/commandline/events/, out 2019.
[8] D. py Docs, "Docker-py." https://docker-py.readthedocs.io/en/1.2.3/, out 2019.
[9] Diamanti, "Container adoption benchmark survey." http://www.diamanti.com, out 2019.
[10] D. Docs, "How swarm works." https://docs.docker.com/engine/ swarm/how-swarm-mode-works/nodes/, out 2019.
[11] H. fabric Docs, "Peers and organizations.." https://hyperledger-fabric.readthedocs.io/en/release-1.4/peers/peers.html, out 2019.