INSTITUTO FEDERAL
Pernambuco

Automatizando a implantação de redes Blockchains em ambientes de contêineres

Automating the deployment of Blockchain networks in container environments

RAFAEL MIGUEL LUNA PONTES CAVALCANTI
rmlpc@discente.ifpe.edu.br
ANDERSON APOLÔNIO LIRA QUEIROZ
anderson.queiroz@paulista.ifpe.edu.br

RESUMO

A implantação de redes blockchain permissionadas, como as baseadas no Hyperledger Fabric, é um processo complexo que exige configurações detalhadas de infraestrutura, certificados e recursos de rede. Este trabalho propõe uma ferramenta chamada HLF AUTO, desenvolvida em Golang, que automatiza esse processo por meio da integração com o Hyperledger Fabric Operator em ambientes Kubernetes. A infraestrutura é provisionada com o uso de um módulo Terraform em um cluster EKS da AWS, enquanto a linguagem declarativa PKL é usada para gerar arquivos de configuração YAML e JSON de forma parametrizada. A proposta visa reduzir o esforço manual na criação de ambientes blockchain reutilizáveis e permitir personalizações rápidas na infraestrutura e na própria blockcvchain.

Palavras-chave: Blockchain; Hyperledger Fabric; Kubernetes; Terraform; Automation.

ABSTRACT

Deploying permissioned blockchain networks, such as Hyperledger Fabric, is a complex process that requires infrastructure configurations, certificates, and network resources. This work proposes a tool called HLF AUTO, developed in Golang, that automates this process through integration with the Hyperledger Fabric Operator in Kubernetes environments. The infrastructure is provisioned using a Terraform module on an AWS EKS (Elastic Kubernetes Service) cluster, while the declarative language PKL is used to generate parameterized YAML and JSON configuration files. The proposal aims to reduce the manual effort required to

create reusable blockchain environments, in addition to allowing rapid customizations in the infrastructure and blockchain.

Keywords: *Blockchain, Hyperledger Fabric, Kubernetes, Terraform, Automation*

1 INTRODUÇÃO

A tecnologia blockchain ganhou destaque nos últimos anos devido ao seu potencial de transformação em diversas áreas, como finanças e saúde. Baseada em um registro distribuído e imutável, a blockchain permite a realização de transações de maneira segura e transparente. Nesse sentido, o Hyperledger Fabric destaca-se como uma solução voltada para o mundo empresarial, porque oferece recursos para a criação de redes de blockchain permissionadas.

Apesar dos benefícios relacionados à segurança, integridade e desempenho, a complexidade de implementar essas redes é um grande desafio, visto que é necessária uma bagagem de conhecimento para implantar e administrar uma blockchain. Segundo o artigo "Leveraging Blockchain Technology for Enhanced Data Management for BI", as equipes técnicas precisam, em média, de 80 horas de treinamento especializado para entender o funcionamento básico da tecnologia.

Seguindo esse raciocínio, ao considerar a inserção dessa tecnologia em ambientes conteinerizados orquestrados com kubernetes, adiciona-se uma nova camada de abstração, relacionada tanto à operação do próprio orquestrador quanto à integração adequada entre o Kubernetes e os componentes do Hyperledger Fabric. Assim, as equipes precisam dominar não apenas os conceitos de blockchain, mas também os mecanismos de provisionamento, escalabilidade e segurança oferecidos por ambientes nativos em nuvem.

Nessa perspectiva, este trabalho tem como objetivo apresentar uma ferramenta para facilitar esse processo de inserção da blockchain em um ambiente com kubernetes. O foco é fornecer uma interface simples e personalizável para a criação de uma infraestrutura visando hospedar um Hyperledger fabric. Para isso, foram criados dois módulos, o primeiro modulo escrito com Terraform refere-se à infraestrutura, em que o usuário poderá criar um cluster EKS na AWS e será capaz de administrar quantidade de nós, definir versões de alguns componentes criados previamente, como o ingress nginx, cert-manager e o elastic block storage da aws.

Relacionado ao segundo módulo, o usuário poderá, por meio de uma CLI escrita em Golang, definir a estrutura da rede blockchain, isso significa que é possível modificar a quantidade de unidades certificadoras, quantidade de organizações, quantidade nós de pares e nós ordenadores, como também poderá utilizar as configurações pré-estabelecidas pelo próprio módulo, em que inclui três *templates* para testes locais.

Assim, com o intuito de facilitar a personalização tanto do cluster quanto da rede Hyperledger Fabric, o módulo de infraestrutura inclui ainda um pacote escrito em PKL, uma linguagem de programação capaz de renderizar arquivos nos formatos YAML e JSON. Essa abordagem permite abstrair a complexidade de configurações longas e repetitivas, visto que apenas algumas linhas de código, o usuário pode provisionar de forma automatizada toda a infraestrutura e a rede blockchain necessária ao seu ambiente. Além disso, contará com o benefício de definir sua infraestrutura como código sendo capaz de ser versionado, ou seja, uma característica importante em momentos de replicação de ambiente e resolução de desastres.

.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 BLOCKCHAIN

Blockchain pode ser compreendido como um livro-razão mantido por uma rede de nós, em que cada nó é responsável por guardar uma cópia do livro. Assim, para adicionar uma nova informação no livro, ou seja, um novo bloco, cada membro da rede deve concordar com a transação e deve ser guardada a sequência de transações até chegar no estado atual do livro-razão, que será irreversível após a conclusão do processo.

Além disso, as Blockchains podem ser classificadas em públicas, privadas, permissionadas ou construídas por um consórcio. Nesse sentido, as públicas são blockchains que qualquer pessoa pode participar, como o Bitcoin (Nakamoto, 2008), as privadas são controladas por uma entidade que determina quem pode entrar na rede (Buterin, 2015), por exemplo, a ethereum. As blockchains permissionadas são configuradas por uma empresa e o participante precisa de um convite para participar ou permissão (Oracle, [s.d.]), por fim, as blockchains construídas por um consórcio, são caracterizadas por ser mantidas por um grupo de empresas que determina quem pode participar.

2.2 CONTÊINERES E ORQUESTRADOS DE CONTÊINERES

Segundo Hightower, Burns e Beda (2022), contêineres são uma forma de empacotar uma aplicação e todas as suas dependências. Em outras palavras, conteinerização significa isolar a aplicação e entregar apenas um conjunto reduzido de recursos para garantir o melhor uso do hardware da máquina hospedeira. Para realizar a criação de contêineres, o Linux Kernel possui o Cgroups (control groups), usado para limitar e isolar o uso de CPU, memória, disco e rede. E o Namespace, responsáveis por isolar grupos de processos, de modo que eles não enxerguem os processos de outros grupos, ou em outros contêineres no sistema host. No caso dos Namespaces, alguns exemplos são: pid para isolamento de processo, net para controle de interface de rede e mnt para gestão de pontos de montagem. O Cgroups, nesse sentido, permite o administrador compartilhar recursos do host com os contêineres, podendo limitar ou aumentar recursos. O Union file System é outros recursos importante, principalmente, no docker, visto que é um sistema de arquivo que funciona por meio de criação de camadas (Romero, 2015), assim, útil para a criação dos arquivos de configuração.

Apesar do grande benefício dos contêineres em relação à performance, provisioná-los sem ajuda de um orquestrador ou uma ferramenta para criação de imagens, é uma tarefa não convencional, podendo conter erros humanos na hora de administrar recursos. Para resolver isso, ferramentas como Docker e Podman, são úteis para criação de imagens, e orquestradores de contêineres, responsáveis por administrar em larga escala, como kubernetes e docker swarm, tentam auxiliar na tarefa de provisionamento. Neste momento, é importante compreender que orquestrar significa cuidar do contêiner em todo seu ciclo de vida, isso significa, gerenciar os recursos necessários para mantê-lo ativo de maneira estável até o fim de sua vida. Por exemplo, os orquestradores administram a quantidade de réplicas ativas de contêiner, mantém a conexão com os volumes para acessar dados e fazem o balanceamento de carga, dentre outras funções.

2.3 INFRAESTRUTURA COMO CÓDIGO (IaC)

A prática de Infraestrutura como Código (IaC) surgiu como uma abordagem essencial para automatizar o provisionamento e a gestão de ambientes computacionais, principalmente em contextos de computação em nuvem. O processo de configuração manual de servidores, redes e demais recursos, a IaC permite que toda a infraestrutura seja definida por meio de arquivos de configuração versionados e executáveis por ferramentas automatizadas (Apriorit, [s.d.]).

Essa perspectiva promove diversos benefícios, como reprodutibilidade, escalabilidade, rastreabilidade de mudanças e redução de erros operacionais. Com o uso de ferramentas como Terraform, Ansible, CloudFormation, entre outras, as equipes de desenvolvimento e operações podem definir arquiteturas complexas como código, garantir ambientes consistentes entre desenvolvimento, testes e produção, e aplicar práticas de DevOps com maior eficiência.

Segundo Hummer et al. (2013), a automação da infraestrutura por meio de código contribui significativamente para a confiabilidade de sistemas distribuídos e a agilidade no processo de entrega de software, especialmente em arquiteturas baseadas em microsserviços e orquestração de contêineres.

2.4 PROVEDORES DE NUVEM

Provedores de nuvem podem ser compreendidos como empresas de TI que oferecem serviços online, tais serviços podem ser divididos nas três categorias principais: PaaS, IaaS e SaaS. Paas, significa plataforma como serviço, ou seja, o provedor irá oferecer toda a plataforma necessária para executar, por exemplo, um site. Na AWS, que é um dos provedores mais conhecidos atualmente, o AWS Elastic Beanstalk facilita a criação de serviços web, porque configura a infraestrutura de forma automática, após o desenvolvedor enviar o seu código. Em relação a IaaS, infraestrutura como serviço, é ofertado os recursos de infraestrutura, como máquinas virtuais, o EC2 da AWS é um exemplo de máquinas virtuais como serviço ofertado. Por fim, SaaS, o software como serviço, é oferecido ao cliente um software completo sem a necessidade de qualquer configuração, o Google Workspace é um exemplo de SaaS, porque oferece Gmail, docs e drive.

3 TRABALHOS RELACIONADOS

Em razão da complexidade de administrar redes blockchain e da necessidade de diminuir processos manuais ao provisionar um hyperledger, há alguns esforços para automatizar esse processo.

De início, foi realizada uma busca no Google Scholar, com foco em pesquisas publicadas entre 2021 e 2025. As investigações encontradas concentram-se na automação da implantação e gestão de redes blockchain permissionadas, com ênfase na simplificação, escalabilidade e integração com ambientes orquestrados por contêineres.

O framework NVAL (Network Validation and Automated Lifecycle) propõe um sistema para automatizar o deploy e a avaliação de redes blockchain privadas, utilizando meta modelagem da arquitetura e execução automática de scripts para provisionar diferentes configurações de rede (Bera et al., 2021). Tal abordagem facilita a execução de testes em larga escala, otimizando o desenvolvimento e a validação de arquiteturas blockchain.

No contexto de aplicações Ethereum, o projeto KATENA apresenta uma modelagem declarativa para implantação e gerenciamento, empregando a linguagem TOSCA em conjunto com o orquestrador xOpera. Essa solução reduz significativamente a quantidade de código necessário e simplifica a gestão das aplicações blockchain (Baresi et al., 2022).

Para redes baseadas em Hyperledger Fabric, destaca-se o Hyperledger Bevel, uma ferramenta da Linux Foundation que automatiza o provisionamento da infraestrutura e a implantação em ambientes Kubernetes. Utilizando Ansible, Helm e operadores Kubernetes customizados, o Bevel oferece uma forma declarativa e escalável de gerenciar redes Fabric (Hyperledger Foundation, 2023). Complementarmente, o Hyperledger Fabric Operator automatiza a criação e manutenção da rede dentro do Kubernetes, cuidando da geração de certificados, configuração dos nós e integração com o cluster (Viejo et al., 2022).

O presente trabalho propõe uma abordagem similar, combinando um módulo de infraestrutura baseado em Terraform para a criação e gestão de clusters EKS na AWS com um módulo de rede blockchain configurável via CLI escrita em Golang, que permite a customização de redes Hyperledger Fabric. Diferentemente das soluções anteriores, esta ferramenta utiliza uma linguagem de programação declarativa para renderizar configurações nos formatos YAML e JSON, possibilitando o provisionamento automático e simplificado da infraestrutura e da rede blockchain com poucas linhas de código.

4 MATERIAIS E MÉTODOS

Para alcançar o objetivo deste trabalho, será utilizado um orquestrador de contêineres, o kubernetes, em que será hospedado na AWS e provisionado por um módulo do terraform visando construir uma infraestrutura reutilizável. Além disso, a linguagem de configuração PKL será necessária para renderizar arquivos de configuração utilizados pelo módulo terraform e pela CLI escrita em Golang. Por fim, será utilizado o Hyperledger Fabric Operator, pois administra a geração de certificados assinados, a ferramenta em Golang proposta neste trabalho pode ser chamada de “HLF AUTO”. Nesse sentido, tal ferramenta fará a integração do Hyperledger Fabric Operator para gerar os certificados e criará os

recursos da blockchain com base no arquivo renderizado pelo PKL. Serão adicionados nesse fluxo templates criados previamente visando facilitar os testes por parte do usuário.

4.1 HYPERLEDGER FABRIC

O HyperLedger Fabric é uma plataforma de blockchain permissionada voltada para o uso empresarial. Seus principais componentes são: Peer Nodes, Orderer, Chaincode, Ledger, Channels, Client applications, Fabric CA e Gateway. Assim, Peer Nodes, exerce a função de endorser e committer, em que o primeiro realiza a validação e a assinatura das transações, e o committer armazena o ledger e atualiza o estado após o consenso ser alcançado. O Orderer realiza a ordenação das transações e forma os blocos que devem ser enviados aos peers para validação, a ordem é fundamental para a consistência das informações. O Chaincode (contrato inteligente), é lógica de negócio do HyperLedger, os quais podem ser escritos em Go, Java ou JavaScript.

O Ledger em si, é composto por blocos, que contém as transações em um determinado período, e o estado global é responsável por representar o estado atual das informações. Os channels, são as subdivisões da rede para garantir a comunicação privada entre os participantes. Logo em seguida, o Cliente applications, são as aplicações externas utilizadas para enviar transações ou consultar o ledger. Essas aplicações se comunicam com os peers para poder chamar os chaincodes. Por fim, o Fabric CA, emite os certificados digitais visando garantir identidade e segurança, e o Gateway faz o encaminhamento das transações ou consultas entre o cliente e os peers. Esse último está presente na versão 2.5 do Hyperledger, que será a utilizada neste trabalho.

4.2 KUBERNETES

O kubernetes é um orquestrador de contêineres, que permite uma vasta configuração de componentes. Os principais recursos são os PODs, Services, Ingress, replicaSet, PV, PVC, nós de controle e nós de trabalho. O POD, é um componente que permite agrupar mais de um contêiner e permite a comunicação entre eles por meio do localhost, como também em seu arquivo de configuração no formato yaml, é possível definir as variáveis de ambiente, a imagem utilizada e alguns outros recursos mais avançados, como o health check para o app implantado.

O service permite expor o contêiner ao acesso externo, em seu arquivo de configuração no formato yaml, é permitido definir qual porta, protocolo e a forma como será

exposto à aplicação, que pode ser utilizando o ip do nó, em que o pod está alocado, ou utilizando um recurso de proxy para encaminhar o tráfego para o contêineres, ou seja, o ingress. Nesse sentido, o Ingress permite o encaminhamento do tráfego externo para os contêiner utilizando um proxy, o mais comum é utilizar o servidor web NGINX para poder fazer esse encaminhamento, o qual é redirecionado utilizado DNS. O replicaSet se encaixa nessa configuração controlando a quantidade de pods existentes, por exemplo, se no deployment estiver definido que deve haver três pods, caso alguém delete um deles, o replicaset se encarregará de criar um novo pod idêntico aos outros.

Junto a isso, um operator é um tipo de recurso do kubernetes que encapsula a lógica de uma aplicação de forma que pode ser reutilizada por outros usuários. Por exemplo, o operator do MySql define os recursos e uma lógica mínima para implantar esse SGBD no kubernetes. De forma semelhante, o operator utilizado neste trabalho, abstrai a complexidade de administrar a criação de certificados assinados para os recursos do Hyperledger.

4.3 HELM CHART

O helm chart é um empacotador de manifesto para kubernetes, com essa ferramenta é possível abstrair configurações complexas e o usuário pode configurar seu aplicativo utilizando apenas um arquivo no formato yaml, isso facilita a replicação e reutilização de projeto. Neste trabalho, tal ferramenta será utilizada para configurar ferramentas previamente configuradas por seus administradores. Será por meio do terraform o principal uso, visto que é uma forma eficaz de implantar recursos importantes para um cluster kubernetes.

4.4 LINGUAGENS UTILIZADAS

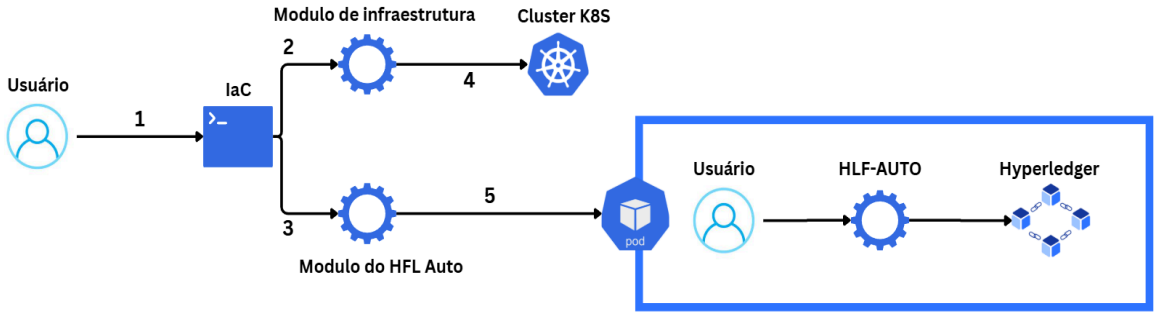
O Terraform permitirá a criação automatizada de recursos de infraestrutura na AWS, como o cluster EKS, VPC, sub-redes, Load Balancers e volumes EBS. Golang é utilizada na construção da interface de linha de comando (CLI) para definir e instanciar redes Hyperledger Fabric. Por fim, PKL, por ser uma linguagem declarativa usada para renderizar arquivos YAML e JSON, reduzindo a complexidade de configuração.

5 ARQUITETURA DA SOLUÇÃO

De início, o projeto pode ser dividido em três partes, na qual a primeira refere-se a infraestrutura construída para hospedar a blockchain, a segunda é a própria ferramenta de CLI construída neste trabalho para implantar o HyperLedger Fabric e, por fim, um módulo escrito

PKL, que permite renderizar configurações personalizadas para o cluster kubernetes e seus recursos, como também para a Blockchain. Abaixo, na figura , é possível visualizar como o fluxo de trabalho dos módulos criados.

Figura 1: Fluxo de trabalho da solução



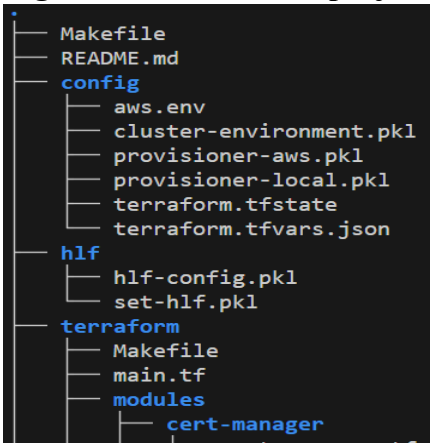
Fonte: Autoria própria, 2025.

No início do fluxo, conforme a imagem 1 apresenta, o usuário acessa a base de código e pode definir as configurações básicas no passo dois e três para a infraestrutura no modulo de infraestrutura e para a blockchain no modulo do HLF Auto, no passo quatro a infraestrutura é provisionada e no quinto passo o usuário precisa acessar o POD para poder aplicar a blockhchain previamente configurada no módulo do HFL Auto.

5.1 ARQUITETURA DA INFRAESTRUTURA

O módulo do terraform e do PKL fazem parte do mesmo repositório, abaixo há os principais diretórios para essa parte do projeto na figura 2 abaixo.

Figura 2: Diretórios do projeto

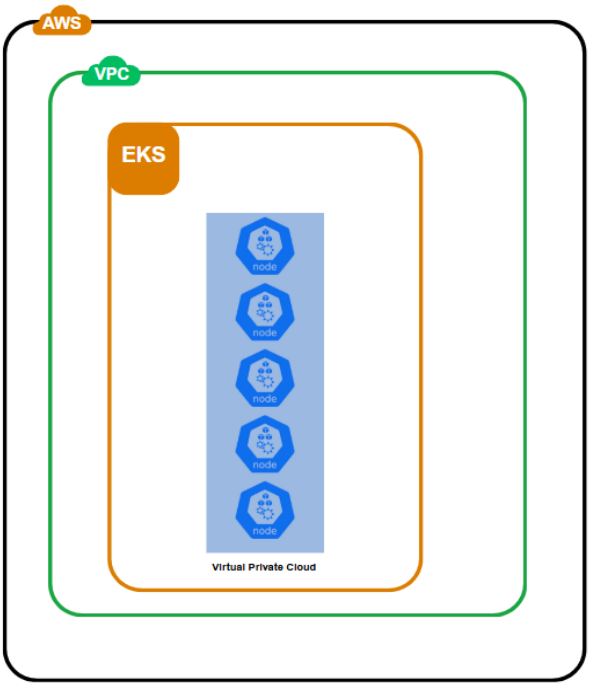


Fonte: Autoria própria, 2025.

No arquivo Makefile do diretório raiz, há entradas que facilitam a chamada de alguns scripts para construção da infraestrutura e para renderização de arquivos essenciais para definição das configurações do projeto. Além disso, será necessário que o usuário possua uma conta AWS com credenciais válidas e adicione “AWS_ACCESS_KEY_ID” e “AWS_SECRET_ACCESS_KEY ” no arquivo “/config/aws.env”. Seguindo o roteiro do projeto, dentro do diretório “config”, o usuário poderá definir a configuração do cluster EKS no arquivo “cluster-environment.pkl”. Por exemplo, quantidade de nós do EKS, versão do cert-manager, versão do nginx e do EBS.

Após definir ou optar pela configuração pré-estabelecida, o usuário pode utilizar a entrada “make aws_render” para renderizar o arquivo “terraform.tfvars.json” que será consumido pelos módulos definidos na pasta “modules” do projeto. Para aplicar tal configuração, o usuário poderá utilizar “make aws_apply” para aplicar a configuração ou “make aws_plan” para visualizar o que será criado em sua conta AWS. Abaixo há a representação do cluster EKS definido como configuração padrão utilizando cinco nós e uma VPC. Nesta figura 3 é abstraído outras definições como tabela de rotas, regras de firewall e entrada para internet.

Figura 3: Estrutura do cluster Kubernetes padrão



Fonte: Autoria própria, 2025.

Durante o processo de criação da infraestrutura, é importante ser citado a criação de um contêiner, que será utilizado como caixa de ferramentas para executar a CLI escrita em Golang. Esse contêiner contará com ferramentas, como Git, kubectl, Golang, um plugin do HLF Operator para poder definir configurações para a blockchain. Junto a isso, foi criada uma imagem base que foi construída previamente com intuito de instalar todas as dependências para o HLF Auto com intuito de diminuir o tempo de espera ao provisionar a infraestrutura com o Terraform. Pode-se observar esta parte no projeto na imagem-1, especificamente, no diretório "Deployments", onde possui alguns scripts escritos em Bash e a definição do POD no arquivo toolbox.yaml, um desses scripts clona o repositório contendo HLF Auto, responsável pela criação da blockchain. A imagem base pode ser verificada neste diretório: https://github.com/rafaellunaM/hlf-toolbox e está armazenada no Docker Hub para ser utilizada pelo contêiner. Outro ponto importante nessa fase de criação, é relacionada ao tempo de provisionamento, por padrão a criação de um cluster padrão no Amazon EKS leva, em média, cerca de dez minutos, conforme destacado pela própria AWS: *"Amazon EKS reduces control plane creation time by 40%"* (AMAZON WEB SERVICES, 2021).

5.2 ARQUITETURA DO HLF AUTO E DA REDE BLOCKCHAIN

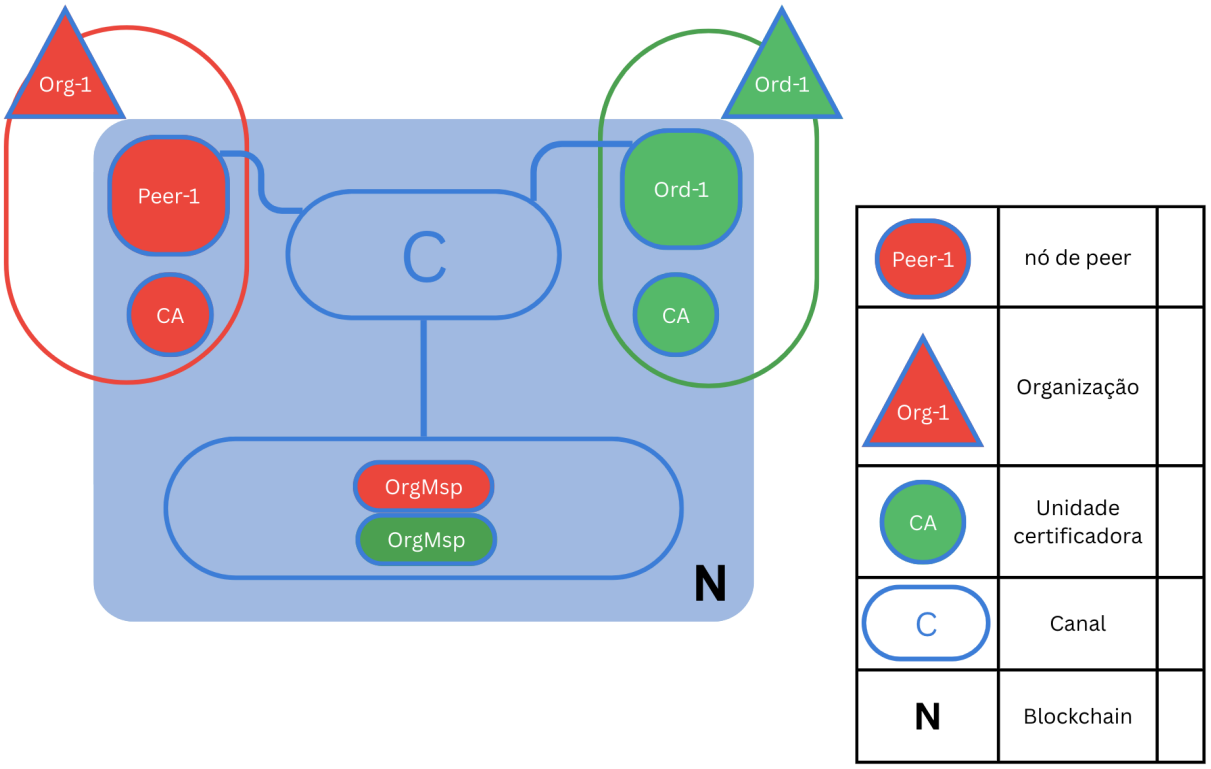
Após esse processo de provisionamento, o usuário poderá iniciar a criação de sua rede HyperLedger. Dentro do diretório "hlf" apresentado na imagem-1, é possível definir as configurações da rede dentro do arquivo "set-hlf.pkl", em que é possível definir o nome do canal principal, a quantidade de pares por organização, o nome desses pares, a quantidade nós ordenadores, a quantidade de nós ordenadores e o nome desses nós ordenadores. Esse módulo, especificamente, abstrai complexidades relacionadas a geração de certificados, usuários, senhas, tipo de banco e protocolo utilizado para ordenação das requisições para o HyperLedger Fabric.

Nesse sentido, o padrão da configuração, conta com o protocolo RAFT implementado nos nós ordenadores com intuito de fazer um consenso sobre a ordem das requisições, por exemplo, se existir três nós ordenadores será feito o quorum para confirmar a ordem de adição ou remoção dos dados de um livro razão em um dos nós de par.

Dessa forma, o usuário não precisa lidar com tal complexidade é apenas definir o tamanho de sua rede blockchain. Há também uma configuração padrão definida no módulo, que consiste em duas organizações, uma para os nós ordenadores e outra organização para nó que armazena o livro razão, cada organização contará com uma unidade certificadora para

validar a identidade dos nós e eles se comunicaram por um canal principal denominado ‘demo“, em razão da natureza do Operator do HyperLedger Fabric no Kubernetes, é necessário criar um canal para instalação do Chaincode, o qual é responsável por definir a regra de negócio de um projeto. Abaixo segue a figura 4 exemplificando essa configuração padrão.

Figura 4: Rede blockchain padrão



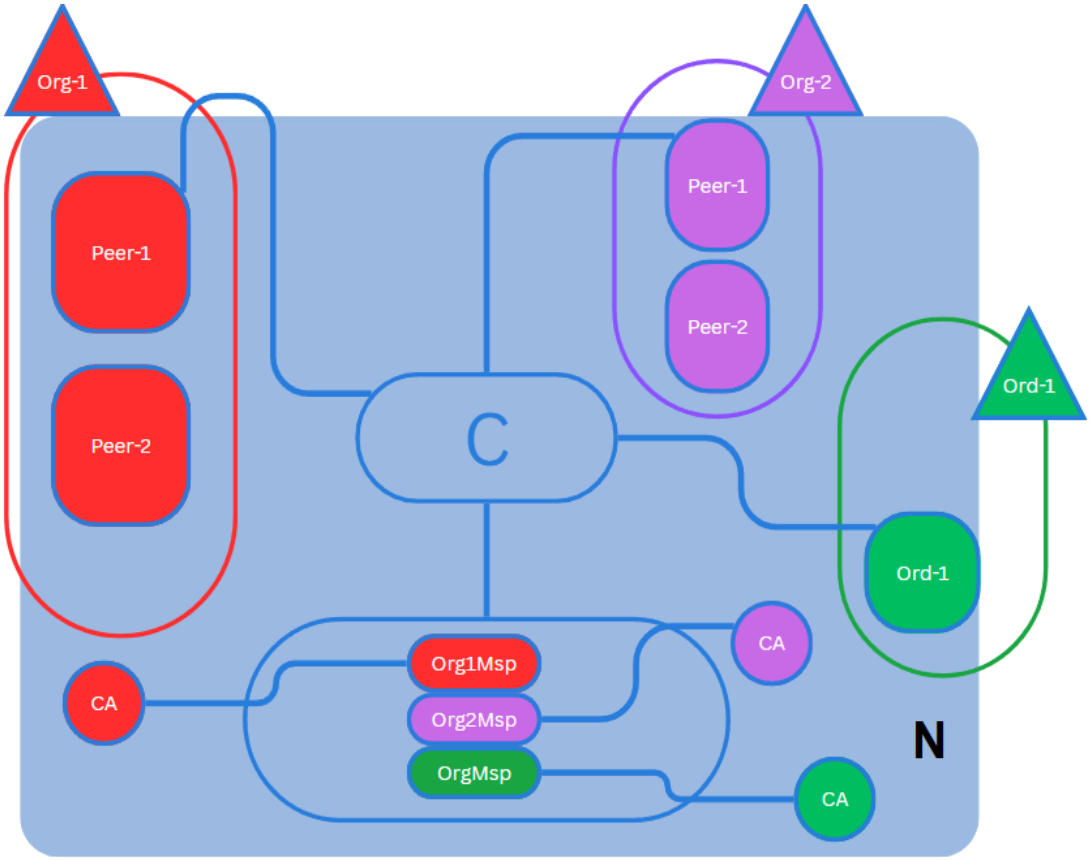
Fonte: Autoria própria, 2025.

Esse processo de definição da blockchain pode ser feito anteriormente ao provisionamento da infraestrutura ou pode ser personalizado posteriormente, mas o usuário irá precisar chamar a entrada do makefile “make hlf_apply”, em que deverá recriar o contêiner toolbox. Logo após esse processo, é possível acessar o contêiner utilizando “make hlf_access” e iniciar o processo de criação da blockchain por meio do comando “go run ”.

O HLF Auto após ser chamado pelo binário do GoLang apresenta algumas opções que podem ser divididas em definição de rede e chamadas dos scripts GoLang. A primeira categoria permite o usuário escolher a configuração definida no módulo PKL ou escolher dois templates implementadas diretamente na ferramenta. O primeiro template constrói uma rede com três organizações, em que duas delas contará com dois peers e uma unidade certificadora, e outra

será composta por um nó ordenador e uma unidade certificadora. Ademais, haverá o canal principal para todos os componentes se comunicarem. Abaixo segue a figura 5 deste template.

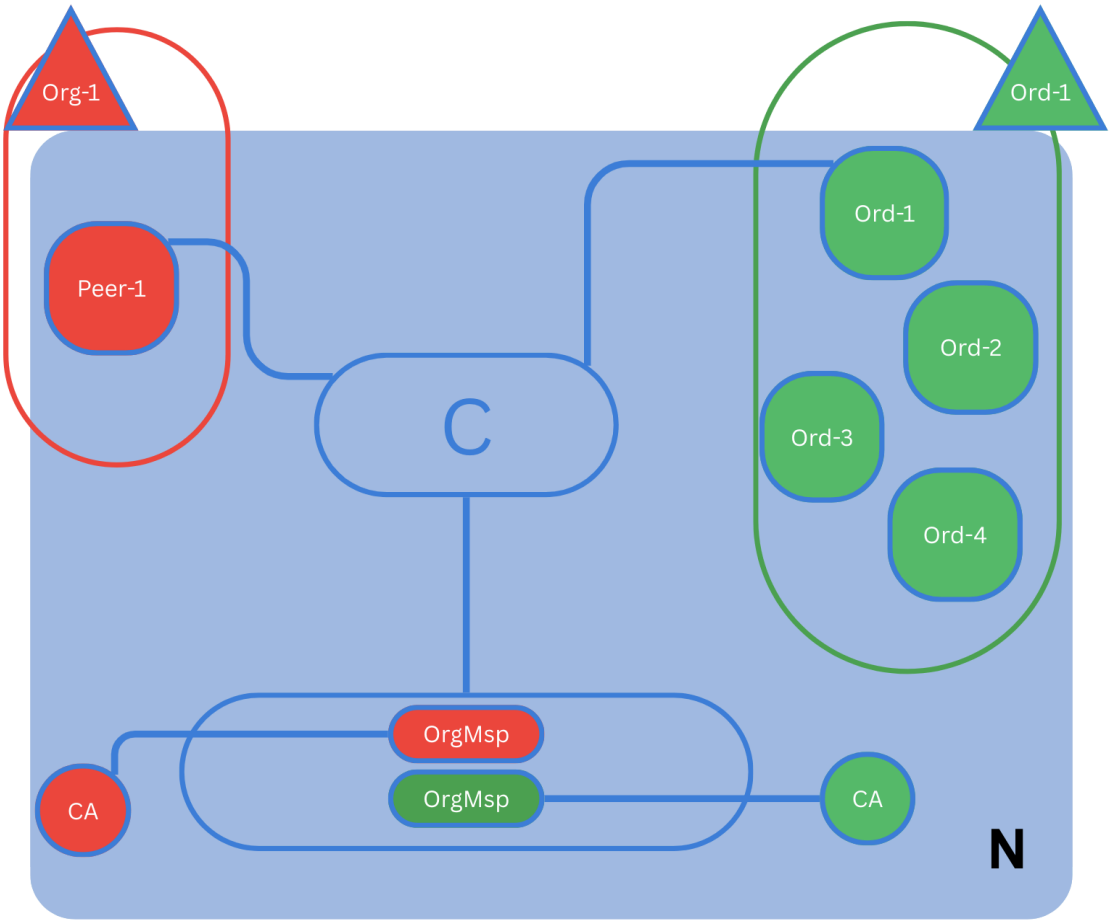
Figura 5: Rede blockchain com três organizações



Fonte: Autoria própria, 2025

Em relação à segunda possibilidade, o usuário pode escolher uma rede com arquitetura com duas organizações, sendo a primeira formada apenas por um peer, que armazenará o ledger, e uma autoridade certificadora e outra organização com quatro nós ordenadores e uma unidade certificadora e ambas organizações se comunicando por um canal principal chamado “demo”. Abaixo segue a figura 6 que faz a apresentação visual deste template.

Figura 6: Rede blockchain com quatro nós ordenadores



Fonte: Autoria própria, 2025

Após a escolha do modelo da Blockchain a ser criada, por meio da ferramenta CLI, é possível implementar todos os componentes utilizando “all”, visualizar o projeto criado utilizando “13” ou deletar todos os recursos criados com “12”. Há a possibilidade de implantar cada componente, basta seguir o sumário definido no README na raiz do projeto do HLF Auto. Por fim, há a possibilidade de instalar um Chaincode de teste para validar a rede HyperLedger e, consequentemente, fazer uma consulta no livro razão povoado.

6 ETAPAS DO DESENVOLVIMENTO

Relacionado às etapas de desenvolvimento, o processo iniciou na escolha do ambiente para construção da infraestrutura. A ferramenta EKS possibilitou a implementação com terraform utilizando o recurso chamado “provider” no terraform, o qual permite a comunicação entre o código e a API da AWS para a definição de recursos. O terraform, nesse sentido, é excelente, porque permite a criação da infraestrutura via código, o que possibilita o

versionamento e reutilização posterior para o usuário que utilizará o módulo. Além disso, foi necessário construir módulos que consumissem um arquivo de definição no formato json visando a personalização da infraestrutura. Recursos definidos, como NGINX, são essenciais para conexão de um usuário externo com um aplicativo hospedado no EKS, o Cert-manager é fundamental para aplicar o TLS nas URLs de acesso aos projetos e o EBS como solução de armazenamento para contêineres que necessitem de volumes. Para todos esses recursos foi necessário um módulo, relacionado a rede blockchain, foi adicionado um módulo para instalação do Operator do Hyperledger fabric, componente base para a criação da blockchain dentro dessa infraestrutura. Em cada um dos diretórios desses módulos, foi escrito um arquivo de variáveis preenchidas por um arquivo json construído posteriormente e um arquivo principal que define as características do recurso.

Relacionado ao módulo config escrito em PKL, foi dividido em duas partes, para a configuração principal da infraestrutura, é escrito o arquivo “provisioner-aws.pkl” encontrado no diretório config, é possível visualizar na imagem-1. Nesse arquivo, é definida cinco classes, uma para cada módulo do terraform relacionado a recursos da infraestrutura e uma classe que abstrai as outras cinco classes com o objetivo de entregar uma interface limpa para o usuário final utilizar no arquivo “cluster-environmment.pkl”. Há um exemplo de utilização no próprio arquivo.

Além disso, utilizando o módulo hlf também escrito em PKL, foi criado a lógica para renderização de um arquivo json, em que define as características da rede Hyperledger Fabric. Essa lógica é composta por cinco classes que definem: unidade certificadora, nó responsável pelo ledger, nó ordenador, canal de implementação de chaincode e canal principal para os componentes.

De início, características, como nome de usuários, senha dos usuários, tamanho do ledger, tipo do banco de dados são definidos e nome dos arquivos contendo as chaves públicas do componente, são definidos previamente. A lógica nesse módulo reduz essa complexidade e é entregue ao usuário apenas uma interface que define a quantidade de componentes da rede no arquivo “set-hlf.pkl”, porém, com um pouco mais de lógica é viável a reescrita das características definidas como padrão.

Seguindo essa linha de pensamento, o módulo HLF Auto, é escrito em GoLang e dividido em três diretórios: templates, internal e cmd. Esse layout é comum em projetos escritos nessa linguagem. O arquivo main.go contém a lógica de interface do usuário, ao chamar esse arquivo as opções que aparecem para o usuário foram construídas nos arquivos

dentro do diretório cmd, o qual é dividido por funções: administration, ca, chaincode, channels, nodes e scripts. No diretório administration, é encontrada a lógica para deletar os recursos criados pela CLI ou mostrar os recursos, na pasta “ca”, é montada a lógica para criação das unidades certificadoras, registro de usuários, validação de usuários e registro de unidade certificadora em um canal. No diretório “node”, são criados os scripts para a criação dos nós ordenadores e dos nós responsáveis pelo ledger. Ademais, na pasta channels é escrito como o main-channel é criado e o canal de configuração da rede, e na pasta scripts é criado o arquivo “execute-pem.go” para criação do PEM dos nós ordenadores para ser adicionado no canal principal. Por fim, na raiz desse módulo, o diretório “internal” contém o arquivo “commons.go”, que é responsável por organizar os dados definidos no json renderizado pelo módulo HLF escrito em PKL.

Após essa organização, os scripts escritos “cmd” chama as estruturas definidas em “internal” e consome os dados do arquivo json, para definir a rede é utilizado a CLI do operator do HyperLedger Fabric, o qual fica responsável por criar os recursos e administrar os certificados dos componentes da blockchain. Utilizando essa CLI, fica mais simples de definir a configuração da rede e replicar caso necessário, além de facilitar a depuração de erros, visto que toda a definição é visível de forma simples no arquivo json renderizado, o qual abstrai complexidade de criação de unidades certificadoras, ordenadores, organizações e tamanho dos bancos, por exemplo.

Relacionado a toolbox criada para este projeto, é utilizada a imagem base do ubuntu 22.04, é escrito um script em Bash para instalação das ferramentas necessárias para o HLF Auto. Após ser construído, é publicado no repositório público do Docker. Isso facilita a reutilização dessa imagem em outros ambientes e diminui a complexidade para utilizar a CLI escrita em Golang.

Dessa forma, é possível definir tanto a infraestrutura quanto a rede blockchain, mantiveram-se alguns processos manuais, como a chamada da CLI, visando o controle do processo de criação da infraestrutura e da blockchain e por serem contextos diferentes.

7 AMBIENTE DE TESTE

No ambiente local, foi necessário AWS CLI na versão 2.15, terraform na versão 1.12, ubuntu 24.04, e PKL 0.25.6. Para infraestrutura foi utilizado o EKS na versão 1.30, nginx na versão 4.10 do helm chart, EBS na versão 2.42 do helm chart. Para a CLI foi usada o GoLang na versão 1.24, AWS CLI na versão 2.15, kubectl na versão 1.30 e o Hyperledger Fabric

operator na versão 1.11 do helm chart, os códigos foram versionados no Github nos seguintes repositórios: https://github.com/rafaellunaM/IaC-tcc responsável pelos módulos Terraforme e PKL, https://github.com/rafaellunaM/hlf-module-tcc é relacionado a ferramenta CLI construída em GoLang, por fim, https://github.com/rafaellunaM/hlf-toolbox é a caixa de ferramentas criada para auxiliar na criação do Hyperledger Fabric.

Para validar essa solução, foi configurado um ambiente utilizando os módulos escritos previamente, como a parte de infraestrutura foi escrita para AWS, utilizou-se uma conta com credenciais válidas para criação do ambiente. Para o teste do módulo de infraestrutura, foi clonado o repositório https://github.com/rafaellunaM/IaC-tcc., foi adicionado as credenciais da conta AWS no arquivo aws.env, como é sugerido pelo README do projeto, e em seguida foi escrito o terminal o comando "make hlf_apply" para utilizar a configuração padrão da infraestrutura. Após o fim do provisionamento, foi utilizado o comando "make hlf_access" para acessar o contêiner e logo em seguida chamar o HLF Auto. Por fim, foi utilizada cada uma das templates e feito uma requisição após cada uma das configurações serem implementadas. Conforme esperado, a requisição mostrou os dados dos livros razões povoadas e definidos pelo Chaincode que a ferramenta define previamente. Abaixo é exemplificado o uso da templates hlf-config.json, o status de sucesso dos componentes e uma consulta invocando os dados da blockchain. nas figuras 7 e 8 na página a seguir.

Figura 7: Status dos PODs.

```
--- Pods ---
NAME                                                READY   STATUS    RESTARTS   AGE
asset-6bb7775fb6-5jchk                              1/1     Running   0          82s
hlf-toolbox-deployment-0                            1/1     Running   0          44m
kfs-hlf-operator-controller-manager-5d84b587cc-sfxfv   2/2     Running   0          44m
ord-ca-68d6b7f475-xwbrg                             1/1     Running   0          6m5s
ord-node1-5b9f5b4985-vrlcx                          1/1     Running   0          4m9s
org1-ca-54d5d9798-5qfq7                             1/1     Running   0          6m5s
org1-peer1-58fccd8fbd-j9trb                         1/1     Running   0          4m31s
```

Fonte: Autoria própria, 2025

Figura 8: Status de uma consulta padrão.

Fonte: Autoria própria, 2025.

8 RESULTADOS E DISCUSSÕES

Os resultados dos testes demonstram que a solução atende aos objetivos de automação e simplicidade para provisionar uma infraestrutura para o Hyperledger Fabric e construir a própria rede blockchain. Nesse sentido, o processo torna-se automatizado possibilitando a redução de erros humanos, permite a reutilização e facilita a manutenção devido ao versionamento no GitHub.

Apesar desse resultado positivo, o trabalho ainda tem algumas limitações e dividas técnicas. Relacionado a limitações, o primeiro ponto é relacionado aos módulos terraform, porque permitem a criação da infraestrutura apenas na AWS, caso o usuário trabalhe com outra cloud ou até mesmo possua uma cloud privada, não será possível reutilizar esses módulos. Outro ponto importante, é relacionado ao processo de precisar deletar o contêiner que executa o HLF Auto, visto que é preciso deletar o contêiner e, em seguida, aplicá-lo novamente para que seja transferido o arquivo JSON renderizado no módulo PKL para dentro do contêiner. Por fim, outro ponto de melhoria é o módulo PKL que renderiza o arquivo JSON para a ferramenta CLI, já que a definição de usuários, senhas e armazenamento podem ser complexas para o usuário final devido à falta de familiaridade com a linguagem PKL.

Em relação à dívida técnica, o processo de acessar o contêiner para executar o HLF Auto pode ser substituído por um caminho, em que o usuário executa o arquivo “” no próprio terminal, evitando a necessidade de acessar o contêiner para executar a ferramenta, ainda que isso seja simplificado pelos scripts escritos no Makefile na raiz do projeto. Outro ponto, é as permissões do contêiner criado, uma vez que é criado com permissões de administrador para executar, deletar, visualizar e atualizar recursos no kubernetes além do necessário. Ou seja, isso pode ter o escopo reduzido visando a segurança do cluster EKS. O módulo do Operator do Hypeledger Fabric para o kubernetes também pode ser revisitado, pois é necessário deixá-lo personalizável, isto é, o usuário deve ter a possibilidade de definir versão do helm chart, arquivo yaml de configuração, namespace utilizado no kubernetes e nome dentro do cluster. Junto a isso, a definição de chaincodes personalizados, por exemplo, o usuário deverá definir um arquivo e a CLI poderá consumir para aplicar na rede criada, atualmente, há apenas um chaincode para os templates.

Para trabalhos futuros, além das questões técnicas a serem resolvidas, a primeira sugestão é referente a possibilidade de utilizar outros ambientes para provisionar a infraestrutura. Por exemplo, para ambientes locais, é possível usar o K3S, que é uma versão para ambientes em desktop do kubernetes, a automação é facilitada, porque é definido via arquivo do tipo YAML, assim, pode ser criado a lógica no módulo PKL para chavear entre os ambientes de acordo com a entrada definida no Makefile do projeto. Em outras palavras, ao utilizar “make k3s_provider” deve ser renderizado um arquivo YAML capaz de ser consumido para a construção do K3S e adicionado também ao Makefile “make k3s_apply” para iniciar o processo de criação do kubernetes no ambiente local. Tal processo pode ser repetido para outra nuvem, como a Azure da Microsoft ou para a nuvem do Google.

Por fim, a outra sugestão, é implementar o Hyperledeger Caliper, framework responsável pela avaliação de desempenho de redes blockchain, o que permitiria realizar testes de carga, latência e throughput sobre a infraestrutura provisionada.

Dessa forma, espera-se que esses trabalhos futuros sugeridos, como o suporte a outros provedores, melhorias na usabilidade e integração com o Caliper framework, aperfeiçoem a ferramenta para ser mais robusta e acessível para diferentes cenários e usuário.

Os testes realizados demonstraram que a ferramenta HLF AUTO atingiu seu objetivo de automatizar o provisionamento da infraestrutura e a criação de redes Hyperledger Fabric. A solução apresentou vantagens em termos de simplicidade, reutilização e redução de erros manuais, alinhando-se às boas práticas de DevOps. Como limitações, destaca-se o suporte

exclusivo à AWS e a necessidade de melhorias na interface de personalização dos parâmetros da blockchain. Trabalhos futuros podem focar na adaptação da ferramenta para outros provedores de nuvem, na integração com o Hyperledger Caliper para análise de desempenho e na criação de uma interface gráfica amigável

REFERÊNCIAS

AMAZON WEB SERVICES. **Amazon EKS reduces control plane creation time by 40%**. 2021. Disponível em: https://aws.amazon.com/about-aws/whats-new/2021/03/amazon-eks-reduces-control-plane-creation-time-40-percent/. Acesso em: 11 jun. 2025.

BARESI, L. et al. **A declarative modelling framework for the deployment and management of blockchain applications**. Journal of Systems and Software, [S. l.], 2022.

BERA, S. et al. **NVAL: A framework for automating deployment and evaluation of blockchain network**. IEEE Access, [S. l.], 2021.

BUTERIN, V**. On public and private blockchains**. *Coindesk*, 2015. Disponível em: https://www.coindesk.com/markets/2015/08/07/vitalik-buterin-on-public-and-private-blockchains. Acesso em: 22 jan. 2025.

GOOGLE**. O que é um provedor de serviços em nuvem?** [S. l.], [s. d.]. Disponível em: https://cloud.google.com/learn/what-is-a-cloud-service-provider?hl=pt-BR. Acesso em: 22 jan. 2025.

HIGHTOWER, K.; BURNS, B.; BEDA, J. **Kubernetes: up and running: dive into the future of infrastructure**. 3. ed. Sebastopol: O'Reilly Media, 2022.

HUMMER, W. et al. **Testing idempotence for infrastructure as code. In: International Conference on Software Engineering**, 2013, San Francisco. *Proceedings*... [S. l.]: IEEE, 2013.

HYPERLEDGER FOUNDATION. **Hyperledger Bevel**. [S. l.], [s. d.]. Disponível em: https://www.hyperledger.org/use/bevel. Acesso em: 7 jun. 2025.

JAIN, R. **The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling.** New York: Wiley, 1991.

NAKAMOTO, Satoshi. **Bitcoin: a peer-to-peer electronic cash system**. 2008. Disponível em: https://bitcoin.org/bitcoin.pdf. Acesso em: 7 jun. 2025.

ORACLE. **Permissioned blockchain**. [S. l.], [s. d.]. Disponível em: https://www.oracle.com/developer/permissioned-blockchain/. Acesso em: 22 jan. 2025.

ROMERO, D. **Docker e contêineres: contêineres com Docker, do desenvolvimento à produção.** 1. ed. São Paulo: Casa do Código, 2015.

VIEJO, D. et al. **Hyperledger Fabric Operator: simplifying blockchain deployment on Kubernetes**. International Journal of Blockchain Research, [S. l.], 2022.