UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FELIPE VASCONCELOS MARINHO

DEVSPACES:
FERRAMENTA PARA CRIAÇÃO DE AMBIENTES NA NUVEM USANDO INSTÂNCIAS PREEMPTIVAS

CAMPINA GRANDE - PB

2023

FELIPE VASCONCELOS MARINHO

DEVSPACES:
FERRAMENTA PARA CRIAÇÃO DE AMBIENTES NA NUVEM USANDO INSTÂNCIAS PREEMPTIVAS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : João Arthur Brunet Monteiro

CAMPINA GRANDE - PB

2023

FELIPE VASCONCELOS MARINHO

DEVSPACES:
FERRAMENTA PARA CRIAÇÃO DE AMBIENTES NA NUVEM USANDO INSTÂNCIAS PREEMPTIVAS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

João Arthur Brunet Monteiro

Orientador – UASC/CEEI/UFCG

Franklin de Souza Ramalho

Examinador – UASC/CEEI/UFCG

Melina Mongiovi Sabino

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 17 de Novembro de 2023.

CAMPINA GRANDE - PB

RESUMO

No processo de desenvolvimento de software, a aquisição e manutenção de hardware adequado para as necessidades de programação podem resultar em altos custos de investimento de capital. A alternativa de uso de recursos em nuvem oferece flexibilidade, porém o gerenciamento desses recursos pode ser complexo e oneroso, requerendo conhecimentos especializados em operações em nuvem. O problema consiste em gerenciar um ambiente de desenvolvimento na nuvem de forma eficiente, evitando altos custos de aquisição e manutenção de hardware próprio, além de simplificar o gerenciamento de recursos ao alugar máquinas na nuvem, buscando minimizar despesas e eliminar a necessidade de expertise complexa em operações em nuvem. Propomos o desenvolvimento de uma ferramenta de linha de comando, destinada a simplificar o gerenciamento do ambiente de desenvolvimento de software. Essa ferramenta terá a capacidade de criar, configurar e gerenciar recursos na nuvem de forma automatizada e eficiente. Uma característica diferencial é a utilização de instâncias preemptivas oferecidas por provedores de nuvem, permitindo aproveitar recursos ociosos a custos ainda mais baixos, sem comprometer a qualidade do ambiente de desenvolvimento. Espera-se que o usuário seja capaz de criar ambientes de desenvolvimento utilizando a ferramenta proposta integrando-a com outras soluções já existentes para desenvolvimento de código. Ao oferecer uma solução intuitiva, nossa abordagem visa otimizar o ambiente de desenvolvimento, maximizando a economia e eliminando a necessidade de conhecimentos avançados em operações em nuvem por parte da equipe de desenvolvimento. Ao final deste trabalho, a usabilidade da ferramenta foi validada e demonstrou ser eficaz na simplificação do gerenciamento dos ambientes. A maioria dos participantes conseguiu gerenciar ambientes com sucesso, destacando a facilidade de uso e a utilidade da documentação fornecida.

DEVSPACES:
TOOL FOR CREATING CLOUD ENVIRONMENTS USING PREEMPTIBLE INSTANCES

ABSTRACT

In the software development process, purchasing and maintaining hardware suitable for programming needs can result in high capital investment costs. The alternative of using cloud resources offers flexibility, but managing these resources can be complex and costly, requiring specialized knowledge in cloud operations. The problem consists of managing a development environment in the cloud efficiently, avoiding high costs of purchasing and maintaining your own hardware, in addition to simplifying resource management when renting machines in the cloud, seeking to minimize expenses and eliminate the need for complex expertise in cloud operations. We propose the development of a command-line tool, intended to simplify the management of the software development environment. This tool will have the ability to create, configure and manage cloud resources in an automated and efficient way. A differentiating feature is the use of preemptible instances offered by cloud providers, allowing you to take advantage of idle resources at even lower costs, without compromising the quality of the development environment. It is expected that the user will be able to create development environments using the proposed tool, integrating it with other existing solutions for code development. By offering an intuitive solution, our approach aims to optimize the development environment, maximizing cost savings and eliminating the need for advanced cloud operations expertise on the part of the development team. At the end of this work, the usability of the tool was validated and demonstrated to be effective in simplifying the management of environments. The majority of participants were able to successfully manage environments, highlighting the ease of use and usefulness of the documentation provided.

DevSpaces - Ferramenta para criação de ambientes na nuvem usando instâncias preemptivas

Felipe Vasconcelos Marinho
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
felipe.marinho@ccc.ufcg.edu.br

João Arthur Brunet Monteiro
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
joao.arthur@computacao.ufcg.edu.br

RESUMO

No processo de desenvolvimento de software, a aquisição e manutenção de hardware adequado para as necessidades de programação podem resultar em altos custos de investimento de capital. A alternativa de uso de recursos em nuvem oferece flexibilidade, porém o gerenciamento desses recursos pode ser complexo e oneroso, requerendo conhecimentos especializados em operações em nuvem. O problema consiste em gerenciar um ambiente de desenvolvimento na nuvem de forma eficiente, evitando altos custos de aquisição e manutenção de hardware próprio, além de simplificar o gerenciamento de recursos ao alugar máquinas na nuvem, buscando minimizar despesas e eliminar a necessidade de expertise complexa em operações em nuvem. Propomos o desenvolvimento de uma ferramenta de linha de comando, destinada a simplificar o gerenciamento do ambiente de desenvolvimento de software. Essa ferramenta terá a capacidade de criar, configurar e gerenciar recursos na nuvem de forma automatizada e eficiente. Uma característica diferencial é a utilização de instâncias preemptivas oferecidas por provedores de nuvem, permitindo aproveitar recursos ociosos a custos ainda mais baixos, sem comprometer a qualidade do ambiente de desenvolvimento. Espera-se que o usuário seja capaz de criar ambientes de desenvolvimento utilizando a ferramenta proposta integrando-a com outras soluções já existentes para desenvolvimento de código. Ao oferecer uma solução intuitiva, nossa abordagem visa otimizar o ambiente de desenvolvimento, maximizando a economia e eliminando a necessidade de conhecimentos avançados em operações em nuvem por parte da equipe de desenvolvimento. Ao final deste trabalho, a usabilidade da ferramenta foi validada e demonstrou ser eficaz na simplificação do gerenciamento dos ambientes. A maioria dos participantes conseguiu gerenciar ambientes com sucesso, destacando a facilidade de uso e a utilidade da documentação fornecida.

PALAVRAS-CHAVE

Desenvolvimento de software, Recursos em nuvem, Gerenciamento eficiente, Automação de recursos, Otimização de custos.

1. INTRODUÇÃO

1.1 Contextualização

O desenvolvimento de software é uma disciplina que se insere no epicentro da transformação digital, tornando-se a base essencial para a criação de soluções tecnológicas que impulsionam o mundo moderno [7]. Nesse contexto, uma premissa crucial emerge: a necessidade de uma infraestrutura computacional sólida para permitir a execução, teste e desenvolvimento eficaz das aplicações.

A complexidade crescente das aplicações de software [8] demanda recursos computacionais robustos para viabilizar a criação e aprimoramento de códigos de forma eficiente. Uma máquina adequada é o alicerce sobre o qual se edifica a capacidade de compilação, depuração e execução de aplicações, possibilitando uma análise precisa dos resultados. Ao contar com uma máquina adequada, os desenvolvedores têm a vantagem de agilizar a detecção de erros, a realização de testes e a validação do desempenho das aplicações em cenários reais.

Tradicionalmente, a aquisição e manutenção de hardware físico têm sido uma opção para desenvolvedores ou empresas, mas essa abordagem pode ser onerosa e trazer diversas preocupações, como os altos custos de aquisição (*CAPEX* alto), despesas periódicas de manutenção, a rápida obsolescência das peças e a depender do hardware utilizado, altos gastos com energia. [9] Além disso, no contexto de trabalho remoto, há ainda o desafio adicional de transportar o hardware para diferentes locais.

A crescente demanda por desenvolvimento de software tem impulsionado a busca por soluções eficientes e flexíveis para criação de ambientes de desenvolvimento. Através da adoção da cloud, já se sabe que hoje é possível reduzir significativamente esses custos [9].

Recentemente, na indústria de desenvolvimento de software, tem havido uma crescente popularização de alternativas que visam facilitar a programação em ambientes remotos. Plataformas como JetBrains Spaces[1], Github Codespaces[2] e Gitpod.io[3] têm ganhado destaque [10] [11]. O que é notável é a integração dessas soluções com os editores de código mais renomados do mercado, como Visual Studio Code[4] (VSCode) e Intellij IDEA[5], que agora oferecem suporte nativo para desenvolvimento de código em máquinas remotas e até dentro de contêineres [2]. Essa tendência oferece maior flexibilidade e colaboração, permitindo que os desenvolvedores trabalhem em ambientes personalizados e compartilhem facilmente projetos em ambientes controlados.

Além disso, uma variedade de modelos de precificação tem emergido nesse espaço. Enquanto algumas soluções adotam uma abordagem de assinatura mensal fixa, ou compra de créditos, outras seguem o modelo "*pay-per-use*", onde a cobrança é feita por cada hora de uso das máquinas. Sob o último modelo, nos serviços supracitados os valores cobrados podem incluir todos os custos associados à instância, como transferência de dados em rede e armazenamento em um único preço, ou cobrá-los de forma individual.

Esses preços podem variar dependendo do número de CPUs e da quantidade de memória da máquina desejada, proporcionando aos desenvolvedores a capacidade de escolher recursos que atendam às suas necessidades específicas. Essa diversidade de opções oferece aos desenvolvedores a liberdade de escolher a solução que melhor se adapte ao seu estilo de trabalho e orçamento.

1.2 Problema

Ao compararmos os preços das soluções de desenvolvimento remoto disponíveis com os custos associados às instâncias na Amazon Web Services[6] (AWS), notamos diferenças significativas. A AWS é uma plataforma líder de serviços em nuvem, oferecendo uma gama diversificada de recursos para empresas e desenvolvedores. Um desses serviços é o Amazon Elastic Compute Cloud[7] (EC2), que possibilita o provisionamento flexível de instâncias virtuais em nuvem, ajustáveis às necessidades de capacidade e aplicação.

Além disso, é importante destacar que aproveitar instâncias preemptivas, conhecidas como EC2 Spot instances, pode ter um impacto significativo na redução dos custos de computação. Por meio delas, os provedores de nuvem oferecem capacidade de processamento excedente a preços substancialmente reduzidos, com a condição de que essas instâncias possam ser interrompidas quando os recursos forem necessários para cargas de trabalho prioritárias.

Seu uso pode consideravelmente diminuir os custos de computação, muitas vezes reduzindo-os em até 90%. Contudo, é relevante considerar que, na AWS, é essencial também abranger os custos associados ao armazenamento e à transferência de dados. Para armazenamento, a AWS oferece o serviço Amazon Elastic Block Store (EBS), que possibilita a alocação de volumes de armazenamento persistentes para instâncias EC2, podendo ser associados a diferentes tipos de instâncias, como as gp3.

Para critério de comparação, assumimos que um desenvolvedor trabalha 170 horas mensais em um mês de 730 horas. O cálculo do custo total pode ser obtido através da seguinte fórmula:

```
(horas_uso * custo_maquina) + (custo_mes *
armazenamento)
```

Utilizando os valores praticados por cada serviço, é possível compará-los entre si utilizando-se como base uma máquina com 40GB de armazenamento de dados, 4vCPUs e pelo menos 8GB de memória RAM.

Provedor	Máquina	Custo (h)	Armazenamento	Custo (GB/mês)	Total/mês
Codespaces	4 vCPUS/8GB RAM	$ 0.36	40GB	$ 0.07	$ 64
Jetbrains Spaces	4 vCPUS/8GB RAM	$ 0.40	40GB	$ 0.1825[8]	$ 75.3
AWS - EC2 On-Demand	t3.xlarge (4 vCPUS/16GB RAM)	$ 0.167	40GB	$ 0.08	$ 31.59
AWS - EC2 Spot	m5.xlarge (4 vCPUS/16GB RAM)	$ 0.07	40GB	$ 0.8	$ 15.1

Tabela 1 - Comparativo de preços entre os principais provedores do mercado.

Uma análise atenta dos custos associados aos provedores de ambientes de desenvolvimento revela um cenário em que a criação direta de recursos na nuvem se apresenta como uma opção mais econômica e vantajosa. Ao avaliar as opções oferecidas por serviços como Codespaces e Jetbrains Spaces, em comparação com a implantação direta em infraestruturas de nuvem como AWS EC2 On-Demand e AWS EC2 Spot, fica evidente que as soluções do mercado podem se mostrar substancialmente mais dispendiosas.

[1] https://www.jetbrains.com/space/features/dev-environments.html
[2] https://github.com/features/codespaces
[3] https://www.gitpod.io/
[4] https://code.visualstudio.com/
[5] https://www.jetbrains.com/idea/
[6] https://aws.amazon.com/
[7] https://aws.amazon.com/pt/ec2/

[8] O serviço cobra $0.01 por hora para cada 40GB, o valor mensal foi obtido assumindo um mês com 730 horas.

No quadro acima, podemos observar a comparação detalhada dos custos mensais associados a diferentes provedores e tipos de máquinas. Ao considerar máquinas com características semelhantes, com o mínimo de 4 vCPUs e 8GB de RAM, constata-se que os preços cobrados por serviços como Codespaces e Jetbrains Spaces ultrapassam significativamente os valores esperados ao provisionar recursos diretamente na nuvem. A diferença é ainda mais expressiva ao comparar com instâncias Spot, como a m5.xlarge da AWS, cujo custo mensal é reduzido a uma fração das opções mencionadas anteriormente.

Contudo, é fundamental ponderar alguns aspectos. Enquanto a economia é notória, as instâncias preemptivas podem ser interrompidas com antecedência, à medida que os recursos são redirecionados para cargas de trabalho prioritárias. Portanto, sua aplicabilidade se destina, preferencialmente, a tarefas que podem ser temporariamente interrompidas [4]. Embora a vantagem financeira seja substancial, é crucial considerar essa dinâmica ao planejar as cargas de trabalho e a arquitetura de sistemas em ambientes que envolvem instâncias preemptivas.

Além disso, ao optar pelo provisionamento direto na AWS, uma série de desafios complexos e técnicos emerge. A configuração e gerenciamento de instâncias EC2, seleção e personalização de AMIs, criação de Launch Templates, alocação e gerenciamento de discos EBS, requisições de instâncias Spot e até mesmo a administração minuciosa de elementos como Security Groups e VPCs, são apenas alguns dos aspectos que compõem essa complexa equação. Coordenar a interação entre esses componentes e garantir um ambiente seguro e bem estruturado requerem conhecimentos detalhados, que muitas vezes são atribuídos a profissionais certificados como "AWS Solutions Architects".

A gestão dessa intrincada rede de elementos não apenas exige expertise técnica, mas também consome tempo valioso que poderia ser direcionado para atividades de desenvolvimento mais produtivas. É compreensível que muitas empresas e desenvolvedores, em face dessa complexidade, prefiram pagar um custo mais elevado por serviços de ambientes de desenvolvimento comerciais. A opção de investir em soluções prontas, apesar do custo adicional, elimina a necessidade de se familiarizar e lidar com as minúcias desses componentes, permitindo que as equipes permaneçam focadas no desenvolvimento de suas aplicações sem o ônus de gerenciar a intrincada infraestrutura subjacente. No entanto, essa escolha muitas vezes implica em um pagamento substancialmente superior, o que reforça ainda mais a necessidade de explorar alternativas mais econômicas e eficientes para o desenvolvimento em ambientes remotos.

Nesse contexto, este presente projeto surge como uma proposta inovadora para abordar os desafios enfrentados por desenvolvedores que buscam alternativas mais econômicas e escaláveis para seus processos de trabalho, como o provisionamento direto. Dado que o gerenciamento da infraestrutura na nuvem pode se tornar complexo e oneroso, especialmente com o uso de instâncias EC2 Spot, para garantir escalabilidade vertical e manter a persistência dos dados para uso futuro.

Com o propósito de enfrentar esses obstáculos, propomos uma solução na forma de uma ferramenta de linha de comando, projetada para gerenciar de maneira eficiente a criação, escalonamento e persistência dos dados das instâncias preemptivas na nuvem, com foco na minimização dos custos.

2. DEVSPACES

A ferramenta DevSpaces[9] oferece uma solução promissora ao permitir que os desenvolvedores trabalhem em ambientes de desenvolvimento remotos, eliminando a necessidade de investimento em hardware físico e reduzindo significativamente os custos associados. Ao utilizar instâncias preemptivas, ou seja, servidores virtuais que usam recursos de computação sobressalentes e não reservados [1], é possível aproveitar recursos disponíveis na nuvem de forma econômica, pagando apenas pelos recursos utilizados, sem a necessidade de aquisição de máquinas dedicadas.

A escalabilidade vertical proporcionada pela nuvem, combinado com a simplicidade de uso da ferramenta é um recurso valioso, pois permite que os desenvolvedores adaptem o ambiente de trabalho de acordo com as necessidades específicas de cada projeto. Por exemplo, é possível mudar para uma máquina com mais recursos conforme a necessidade. Com a capacidade de dimensionar recursos conforme a demanda, os desenvolvedores podem utilizar instâncias com mais recursos em momentos de maior exigência de desempenho e economizar quando a demanda for menor.

Neste trabalho, serão apresentados estudos detalhados sobre o funcionamento e metodologia de desenvolvimento da ferramenta "DevSpaces", e sua integração com práticas de desenvolvimento de software. Pretende-se, ao final deste trabalho, oferecer uma solução de código aberto eficiente e viável para o provisionamento de ambientes remotos, reduzindo significativamente custos e simplificando o processo de criação e gerenciamento de ambientes de desenvolvimento.

Ao eliminar e abstrair preocupações relacionadas à implantação de recursos na nuvem, espera-se que a ferramenta "DevSpaces" permita que os desenvolvedores se concentrem exclusivamente na tarefa de desenvolvimento, proporcionando maior produtividade e eficiência ao processo de criação de software.

2.1 Funcionamento

A ferramenta opera como uma Interface de Linha de Comando (CLI) que utiliza o Software Development Kit (SDK) da AWS para estabelecer comunicação com a API da nuvem. Desenvolvido em linguagem Golang, a escolha desta linguagem se baseou no amplo suporte oferecido tanto para a

[9] https://github.com/felipemarinho97/dev-spaces

criação de CLIs eficazes quanto para a construção de integrações fluidas com a nuvem.

Para utilizar a ferramenta, é imperativo estar autenticado com as credenciais de acesso à alguma conta AWS. A filosofia subjacente à API da interface foi intencionalmente projetada para ser o mais intuitiva possível, especialmente para usuários menos familiarizados com conceitos de nuvem. Contudo, ao mesmo tempo, a ferramenta também oferece meios para que usuários mais experientes possam personalizar suas configurações por meio de parâmetros opcionais.

Seguindo esse raciocínio, um princípio de design central adotado na criação da ferramenta é o foco na simplicidade e eficácia - inspirado pela filosofia Unix de "fazer uma coisa e fazê-la bem". [13] Esse princípio guia a funcionalidade principal da ferramenta, garantindo que ela se concentre em uma única tarefa específica, mas a execute de maneira excepcional. Isso implica que, em muitos casos, abstrações excessivas foram evitadas em nome da flexibilidade. Entretanto, esse enfoque possibilita a construção de soluções personalizadas e complementares em cima da funcionalidade principal da ferramenta, tornando-a flexível e adaptável.

2.1.1 Virtualização

O ponto central para o funcionamento da ferramenta é a utilização estratégica da virtualização, permitindo que instâncias preemptivas atendam às necessidades de cargas de trabalho stateful que demandam consistência de dados e minimização das interrupções. Ao ser criada, uma instância Spot possui disco, cpu e memória voláteis, podendo ser excluídas a qualquer momento. Na nossa abordagem, alocamos um armazenamento não-volátil por meio do Amazon Elastic Block Store (EBS), garantindo a persistência e segurança dos dados. Esse disco, contendo uma raiz de instalação Linux, serve como a base de nossas instâncias preemptivas, oferecendo a estabilidade necessária.

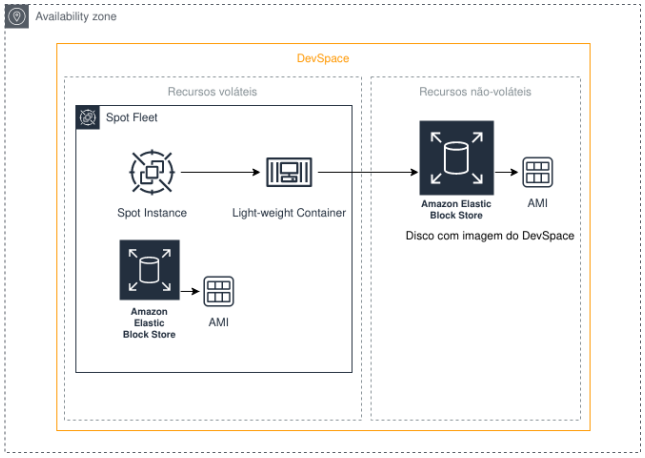

Figura 1 - O gráfico acima ilustra a estratégia utilizada para a criação de ambientes. A instância preemptiva possui seu próprio disco e AMI, porém o disco e a imagem do “DevSpace” são recursos provisionados à parte, com ciclo de vida independentes do ciclo da instância.

Ao iniciar uma instância Spot, um processo customizado de boot é acionado, permitindo que a raiz de instalação não-volátil seja acessada por meio de virtualização, normalmente isso pode ser feito através do utilitário “chroot” [6], porém na nossa abordagem é usado o “systemd-nspawn”.

> “O systemd-nspawn pode ser usado para executar um comando ou sistema operacional em um contêiner de namespace leve. De muitas maneiras, é semelhante ao chroot, mas mais poderoso, pois virtualiza totalmente a hierarquia do sistema de arquivos, bem como a árvore de processos, os vários subsistemas IPC e o host e o nome do domínio.” [5]

Isso permite que serviços systemd possam ser iniciados dentro desse ambiente de contêiner. Recentemente, a maioria das distribuições Linux adotou o systemd, substituindo outros sistemas init, como o SysV init. Exemplos de serviços systemd incluem docker, ssh, cron dentre outros. Essa abordagem, garante que, além da ampla compatibilidade de serviços, no caso de uma interrupção da instância, uma substituta possa ser rapidamente provisionada, mantendo a continuidade das operações e minimizando o tempo de inatividade.

De modo geral na nossa abordagem, a instância spot, seu disco e sua imagem associada serão chamados de **máquina hospedeira**. O disco não-volátil, sua AMI e o contêiner serão o nosso **ambiente de desenvolvimento**.

2.1.2 Script de inicialização

O script desempenha um papel fundamental na automação das tarefas necessárias para o funcionamento do ambiente de desenvolvimento do DevSpace. Ele é executado de forma sistemática após a inicialização da máquina hospedeira, permitindo a configuração adequada para as operações subsequentes.

O processo começa com a verificação e garantia de que o utilitário de virtualização "systemd-nspawn" e os recursos de gerenciamento de rede estejam devidamente instalados na AMI da máquina hospedeira. Após essa etapa, o script entra em um loop de espera, monitorando continuamente a associação de um disco EBS à instância. No contexto da ferramenta, esse disco é sempre associado ao dispositivo /dev/sdf.

Após detectar um novo disco associado à instância, o script prossegue com o terceiro passo, que envolve a identificação do sistema de arquivos presente neste disco e sua subsequente montagem no ponto de montagem "/devspace". Uma vez concluída essa etapa, o script realiza modificações no arquivo de configuração SSH do ambiente de desenvolvimento.

Essas modificações incluem a alteração da porta SSH para 2222, a fim de evitar conflitos com a porta SSH padrão 22 da máquina hospedeira. Além disso, o script adiciona as chaves públicas associadas à máquina hospedeira ao arquivo "authorized_keys" de todos os usuários presentes na imagem

do DevSpace. Isso garante que o ambiente de desenvolvimento possa ser acessado com as mesmas chaves que foram definidas no momento da criação da máquina hospedeira.

O último passo do processo é a instanciação do ambiente virtualizado, realizado através do utilitário "systemd-nspawn". Neste estágio, algumas medidas de segurança são desativadas, uma vez que é desejável que o DevSpace tenha acesso a todos os recursos da máquina virtualizada.

São utilizadas capacidades especiais, definidas através do parâmetro "--capability=all", e são especificados o nome do ambiente, o ponto de montagem e a opção de boot. O utilitário "systemd-nspawn" é responsável por inicializar a AMI presente no disco não-volátil. Dependendo da imagem, esse processo pode levar apenas alguns segundos para ser concluído.

Uma vez que a inicialização esteja completa, o sistema de inicialização da imagem da máquina virtual deve iniciar o serviço SSH, tornando o ambiente acessível por SSH na porta 2222. Isso permite que os usuários acessem o DevSpace de forma eficiente e segura, estando prontos para prosseguir com suas tarefas de desenvolvimento.

Esse script é essencial para garantir que o DevSpace seja configurado e funcione de maneira adequada, possibilitando a criação de ambientes de desenvolvimento coesos e eficazes.

2.2 Funcionalidades

2.2.1 Criação de ambientes

Esta função tem como objetivo a configuração de ambientes de desenvolvimento remoto, requerendo a escolha de uma imagem específica de distribuição Linux. Ao executar o comando "create", é instanciado um disco Elastic Block Store (EBS), associado à AMI Linux selecionada através do parâmetro "--ami". Cabe ressaltar que "AMI" se refere à "Amazon Machine Image", ou seja, uma imagem pré definida contendo um sistema operacional e, opcionalmente, outros softwares. O comando "create" também requer atributos adicionais, como "name", utilizado para fazer referência ao ambiente, e "key-name", referente à chave SSH empregada para a conexão com a instância.

A funcionalidade do comando "create" também oferece a capacidade de definir o tamanho do disco durante a criação do "devspace", atendendo às especificidades do projeto. Além disso, a utilização do parâmetro "--ami" se estende ao suporte de filtros, tais como "name", "owner" e "arch" (arquitetura). Estes filtros permitem a busca por AMIs com base em atributos específicos, possibilitando uma seleção personalizada da imagem desejada. A inclusão do caractere "*" é suportada no filtro "name" amplia ainda mais a abrangência da busca por AMIs. A **Figura 2** abaixo ilustra o uso da funcionalidade.

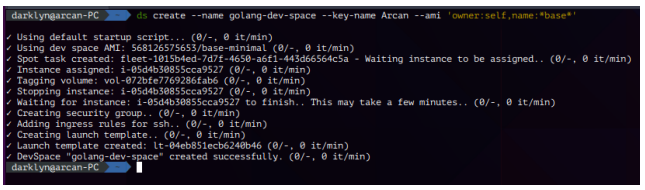

Figura 2 - Exemplo de criação de ambiente utilizando filtro no parâmetro “--ami”.

Após a conclusão da criação do disco EBS com a imagem selecionada, o comando "create" procede à geração de um "Launch Template". Este componente é fundamental para a subsequente ação de inicialização da instância. O "Launch Template" possui algumas definições necessárias, como o Security Group, responsável por determinar as regras de abertura de portas e filtragem de tráfego, a subrede de alocação da instância, zona de disponibilidade, a imagem da máquina hospedeira e o tamanho do disco correspondente. Também é adicionado o script de inicialização customizado, contendo automações a serem executadas sempre que a instância for iniciada, permitindo a configuração e carregamento automatizado dos ambientes de desenvolvimento.

2.2.2 Destruição de ambientes

A funcionalidade de destruição de ambientes, acessível através do comando "destroy", faz parte do grupo de comandos de administração, e é uma das funcionalidades essenciais da ferramenta. Este comando requer o fornecimento obrigatório do nome do ambiente de desenvolvimento através do parâmetro “--name”.

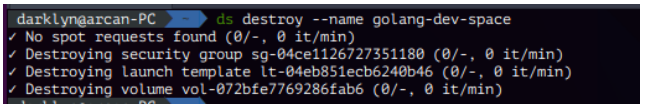

Figura 3 - Exemplo de destruição do ambiente “golang-dev-space”

Ao ser invocado, o comando busca todos os recursos gerenciados pela a ferramenta e efetua a deleção dos mesmos (a figura acima ilustra esse processo). É possível saber exatamente quais recursos foram criados pela ferramenta e quais estão associados ao dev-space através do emprego de Tags.

O recurso de "tagging" é uma prática inerente à infraestrutura em nuvem, particularmente na AWS, onde recursos podem ser categorizados por tags específicas. Essa prática é inclusive encorajada, pois fornece maior controle à quem está gerenciando os recursos na nuvem. Na nossa abordagem todos os recursos possuem ao menos as seguintes tags: "managed-by=dev-spaces" e "dev-spaces:name=<nome-do-ambiente>".

Isso garante que, ao executar a destruição do ambiente, a ferramenta seja capaz de eliminar seletivamente somente os recursos que ela própria gerencia, e os que estão diretamente

associados ao ambiente especificado no parâmetro “name”. Essa metodologia fornece um controle refinado sobre o processo de destruição e reforça a segurança do gerenciamento de recursos em nuvem.

2.2.3 Iniciação de ambientes

A funcionalidade de iniciar ambientes, disponível por meio do comando "start", faz parte do grupo de comandos de controle de ciclo de vida. Esse comando requer o fornecimento obrigatório do nome do ambiente de desenvolvimento através do parâmetro “--name”. Além dessa especificação central, outros parâmetros opcionais desempenham um papel significativo na personalização da inicialização, tais como:

- --min-cpus ou -c: Define o número mínimo de vCPUs exigido. Se não informado, o padrão é zero, ou seja, a ferramenta irá instanciar a máquina com a menor quantidade de vCPUs disponível.
- --min-memory ou -m: Especifica a quantidade mínima de memória RAM em GB. (padrão: 0GB)
- --max-price: Determina o preço máximo por hora (em dólares estadunidenses) que a instância preemptiva pode custar.
- --timeout ou -t: Define um limite de tempo máximo para a execução da instância, evitando gastos desnecessários.

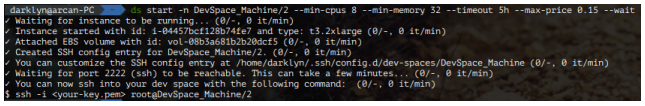

Figura 4 - Exemplo de iniciação para uma máquina com ao menos 8 vCPUs e 32GB de memória RAM (t3.2xlarge).

O comando "start" (**Figura 4**) inicia a criação de uma frota de instâncias preemptivas, conhecida como "Spot Fleets", que contém exclusivamente uma única instância alinhada com as especificações mínimas declaradas. Essas especificações, são em parte fornecidas pelos parâmetros do comando “start”, e complementadas por informações previamente gravadas no Launch Template durante a execução do comando "create".

A AWS seleciona a instância adequada com base nas especificações escolhidas. Durante a seleção, existem duas estratégias de alocação: "menor preço", ou "maior capacidade". A primeira opção priorizará a escolha da instância com o menor preço que atenda as especificações, já a segunda opção também vai priorizar instâncias com preço baixo, porém com uma menor probabilidade de sofrerem interrupções. Por padrão, nossa ferramenta utiliza a estratégia "menor preço", porém é possível alterar isso através do arquivo de configuração.

Após fazer a solicitação da frota, a ferramenta aguarda até que uma instância seja alocada. Posteriormente à alocação da instância, ocorre a conexão do disco EBS com a AMI do dev-space. O User Script, inserido na máquina hospedeira por meio do Launch Template, está em execução aguardando a montagem de um novo disco. Ao ser detectado, esse disco é montado no ponto de montagem "/devspace" e, em seguida, o utilitário "systemd-nspawn" é invocado para estabelecer um contêiner, o qual inicializa a partir do mencionado ponto de montagem.

A operação de inicialização é geralmente rápida e, assim que o ambiente dentro do contêiner é ativado, incluindo os serviços SSH, a máquina está pronta para estabelecer conexões. No contexto da máquina hospedeira, é possível visualizar o contêiner por meio do utilitário "machinectl", proporcionando uma camada adicional de controle e gestão. A ferramenta "dev-spaces" apresenta, assim, um processo estruturado e altamente controlado para iniciar ambientes de desenvolvimento de maneira eficiente e otimizada.

2.2.4 Encerramento de ambientes

A funcionalidade de encerrar ambientes, disponível por meio do comando "stop" (**Figura 5**), faz parte do grupo de comandos de controle de ciclo de vida. Quando invocado, o comando lista todas as requisições de Frotas Spot gerenciadas pela ferramenta. Essas requisições são então filtradas com base no nome do "devspace" através do emprego de Tags, caso seja especificado no parâmetro "--name" pelo usuário. Caso esse parâmetro seja omitido, o comando "stop" irá cancelar todas as requisições de frotas que estão sendo gerenciadas pela ferramenta. Após a filtragem, a ferramenta abre uma requisição de cancelamento de frota para cada frota spot atualmente ativa que atenda aos parâmetros informados.

Uma vez concluída a execução do comando, as requisições de cancelamento de frota ficam agendadas e a infraestrutura da nuvem automaticamente realiza o processo de desmontagem dos discos EBS associados ao "dev-space" e encerramento das máquinas EC2. A instância hospedeira e seu disco EBS correspondente são inteiramente removidos durante a execução desse comando por tratarem-se de recursos voláteis. É importante ressaltar que essa ação não implicará na perda de dados armazenados no "DevSpace", pois esses dados são mantidos seguros no próprio disco não-volátil do ambiente, como exemplificado na Figura 1.

Figura 5 - Exemplo de execução do comando “stop” sem uso de parâmetros. Nesse caso, dois ambientes que estavam ativos foram encerrados.

2.2.5 Acompanhamento de ambientes

A funcionalidade de monitoramento de status e listagem de ambientes é acessada por meio dos comandos "list" e "status" na ferramenta. O primeiro comando, quando acionado, realiza a enumeração de todos os ambientes criados pela ferramenta. Internamente, essa operação consiste em uma requisição que

busca os nomes de todos os *Launch Templates* criados pela própria ferramenta.

A lista resultante inclui informações como o nome do ambiente, sua versão, ID e data de criação. Um exemplo de execução pode ser visto na **Figura 6** abaixo. O uso do parâmetro "-o wide" permite que o comando "list" apresente informações adicionais, como o ID da instância provisionada, seu estado (*RUNNING* ou *TERMINATED*), endereço IP e DNS públicos, chave SSH necessária para conexão e a zona de disponibilidade em que o ambiente está localizado.

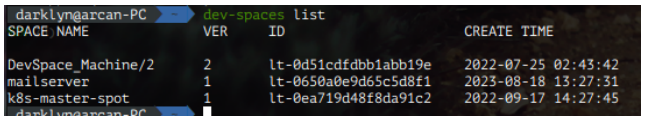

Figura 6 - Exemplo de execução do comando “list” sem o uso do parâmetro “-o wide”.

O outro comando, denominado "status", oferece uma alternativa para verificar o estado da frota Spot de um ambiente específico. Essa funcionalidade pode ser particularmente útil em cenários nos quais a nuvem requisitou a devolução da capacidade da instância spot. Ao usar o comando "status", é possível acessar informações que indicam se a frota foi encerrada (tanto pela nuvem, quanto pelo próprio usuário) ou se ainda está em execução. Essa funcionalidade complementa a funcionalidade de listagem ao oferecer insights instantâneos sobre o estado operacional dos ambientes.

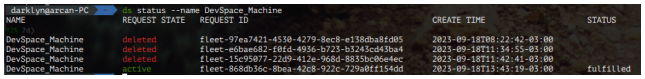

Figura 7 - Exemplo de execução do comando “status”. Aqui é utilizado o filtro informando o nome do ambiente.

No exemplo acima, é utilizado o nome do ambiente como argumento do parâmetro “--name” para filtrar os resultados do comando “status”. Também é possível omitir esse parâmetro, nesse cenário, a ferramenta irá retornar o estado de todas as últimas requisições de frotas spot dentro das últimas 48 horas.

2.2.6 Cópia de ambientes

Esta funcionalidade está disponível através do comando “copy” que é um subcomando do comando “tools”, eles fazem parte dos comandos de administração. Para usá-lo é necessário fornecer o nome do ambiente remoto através do parâmetro “--name”, a região e a zona de disponibilidade destino através dos parâmetros “--new-region” e “--availability-zone”, respectivamente.

Antes de executá-lo, deve-se usar a ferramenta a partir de uma região onde um ambiente já criado se encontra. Por exemplo, para copiar um ambiente de us-east-2 (Ohio) para sa-east-1 (São Paulo) deve-se primeiro utilizar a ferramenta na região de Ohio e indicar a região e zona de disponibilidade em São Paulo através dos parâmetros.

Este comando é útil em cenários onde os preços Spot de uma mesma categoria de máquinas está mais barato em uma região A do que na região B. Nesse caso, basta copiar o ambiente para uma região mais barata (desde que isso não afete significativamente a latência) para disfrutar de preços mais competitivos. Outro cenário seria o caso em que um desenvolvedor muda de localidade. Para manter a latência baixa, bastaria fazer a cópia do ambiente para uma região mais próxima, e posteriormente a destruição do ambiente utilizando a funcionalidade exposta na seção **2.2.2** na antiga região. Sendo assim, é possível derivar a funcionalidade de “mover” o ambiente ao utilizá-lo conjuntamente com o comando “destroy”.

Quando a ferramenta recebe uma solicitação para copiar um ambiente, ela emprega a API do EC2 para realizar essa operação. O primeiro passo é criar um snapshot do disco não volátil do ambiente de desenvolvimento existente. Após a conclusão bem-sucedida da criação desse snapshot, um novo volume EBS é gerado na região de destino, usando o snapshot como base. Para manter a consistência e o controle, esse novo volume é devidamente etiquetado com as informações relevantes.

Além disso, a replicação do ambiente envolve não apenas o disco, mas também outros elementos cruciais, como os grupos de segurança (Security Groups) e os modelos de lançamento (Launch Templates). Esses componentes são duplicados para garantir que o ambiente copiado possua todas as configurações e restrições necessárias, permitindo que ele funcione de forma independente e segura no novo contexto.

2.2.7 Scaling vertical de ambientes

Esta funcionalidade está disponível por meio do comando "scale", que faz parte dos comandos de administração sob o subcomando "tools," tem como objetivo permitir a expansão das capacidades de um ambiente remoto. Este comando requer o nome do ambiente especificado pelo parâmetro "--name", bem como as especificações desejadas para a nova máquina, representadas pelos parâmetros "--min-memory" (memória) e "--min-cpus" (vCPUs). O parâmetro opcional "--max-price" também pode ser fornecido para estabelecer um limite de preço para a nova instância.

A operação em si envolve a criação de uma nova solicitação de frota spot, considerando as novas especificações de recursos. Uma vez que a solicitação é atendida, uma máquina com as características desejadas é inicializada. O User Script desta nova máquina hospedeira é responsável por aguardar a conexão do disco não volátil do ambiente de desenvolvimento e iniciá-lo o quanto antes. Assim que a nova máquina atinge o estado "RUNNING," a ferramenta encerra o contêiner systemd-nspawn em execução na máquina anterior por meio do utilitário machinectl, além de desmontar o disco não volátil.

Após a desmontagem e desconexão do disco, a frota spot da máquina anterior é cancelada, e o disco é anexado à nova máquina hospedeira, agora configurada com as novas especificações. Essa abordagem assegura que o tempo de inatividade associado à escalabilidade vertical de um DevSpace seja mínimo, uma vez que a máquina antiga é desligada apenas quando a nova máquina está totalmente operacional e pronta para uso. Isso garante uma transição suave e eficaz durante o processo de escalabilidade.

3. ARQUITETURA

A ferramenta em questão é essencialmente uma Interface de Linha de Comando (CLI) desenvolvida em Golang, projetada para interagir com a infraestrutura na nuvem AWS por meio da biblioteca AWS SDK v2, que, por sua vez, efetua solicitações à API web do serviço EC2. Contudo, o que diferencia essa ferramenta é a sua arquitetura modular.

O módulo central, denominado "core," desempenha um papel crítico, pois é responsável pela integração com o AWS SDK. Ele estabelece um contrato claro para entrada e saída de dados, proporcionando uma abstração completa dos conceitos subjacentes de infraestrutura. Essa modularização contribui para a manutenção do código e facilita a expansão da funcionalidade da ferramenta.

No contexto deste projeto específico, para a parte de apresentação foi implementada uma interface de linha de comando, encapsulada no módulo "cli". Esse componente atua como a ponte entre o usuário e o módulo "core," orquestrando o fluxo de entrada de dados e comunicação com a ferramenta. Através da CLI, os usuários podem interagir de forma intuitiva com a ferramenta, emitindo comandos para criar, gerenciar e personalizar seus ambientes de desenvolvimento na AWS, com toda a complexidade da infraestrutura oculta por trás dessa interface amigável.

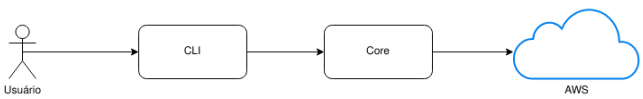

Figura 8 - Representação gráfica da arquitetura da ferramenta.

É válido ressaltar que a arquitetura modular não se restringe apenas à organização dos arquivos e a divisão de pastas. É possível importar cada módulo separadamente a partir de um outro projeto em Golang e estender as suas funcionalidades. Desse modo, seu código pode ser reaproveitado para outros cenários, como por exemplo um servidor web que integra-se ao módulo “core” para criar e excluir ambientes remotos através de uma API Rest.

3.1 Módulo core

O módulo central, denominado "core", é responsável por toda a lógica do domínio do problema na ferramenta, contendo a implementação das funcionalidades para criar, destruir, iniciar e parar ambientes de desenvolvimento. Sua estrutura interna é projetada para oferecer as operações básicas que asseguram o gerenciamento eficiente e automatizado desses ambientes na nuvem, da forma mais genérica possível.

Esse módulo expõe uma interface com métodos para contemplar as funcionalidades descritas na seção 2. Os seguintes métodos exibidos na **Figura 9** são suportados por esta interface.

```
type Handler interface {
    Create(ctx context.Context, opts CreateOptions) (CreateOutput, error)
    Destroy(ctx context.Context, opts DestroyOptions) error
    Start(ctx context.Context, opts StartOptions) (StartOutput, error)
    Stop(ctx context.Context, opts StopOptions) (StopOutput, error)
    ListSpaces(ctx context.Context, opts ListOptions) ([]ListItem, error)
    EditSpec(ctx context.Context, opts EditSpecOptions) (EditOutput, error)
    Copy(ctx context.Context, opts CopyOptions) (CopyOutput, error)
}
```

Figura 9 - Métodos presentes na interface exposta pelo módulo “core”.

Note o uso *structs* como o objetos de entrada e saída dessas interfaces. Esses objetos encapsulam os parâmetros necessários para utilização de cada método dessa interface. Algumas das vantagens de utilizar essa abordagem é que na linguagem Go é possível utilizar metadados denominados *struct tags*. A partir destas tags, é possível adicionar facilmente funcionalidades de validação de entrada de dados, ou serialização em formatos específicos como json ou yaml.

```
type EditSpecOptions struct {
    // Name of the Dev Space
    Name string `validate:"required"`
    // MinMemory is the amount of memory in MiB
    MinMemory int `validate:"min=0"`
    // MinCPUs is the amount of cpus
    MinCPUs int `validate:"min=0"`
    // MaxPrice is the maximum price for the instance
    MaxPrice string `validate:"required"`
    // SSHKey is the path of the SSH key
    SSHKey string `validate:"required"`
}

You, 14 months ago | 1 author (You)
type EditOutput struct {
    // InstanceID is the ID of the instance
    InstanceID string `json:"instance_id"`
    // InstanceIP is the Public IP of the instance
    InstanceIP string `json:"instance_ip"`
    // InstanceType is the type of the instance
    InstanceType string `json:"instance_type"`
    // FleetRequestID is the ID of the FleetRequest
    FleetRequestID string `json:"fleet_request_id"`
}
```

Figura 10 - Utilização de struct tags nos objetos de entrada e saída do método EditSpec.

Para validação dos structs, foi utilizado a biblioteca “go-playground/validator/v10”.

Como dito anteriormente, a função desse módulo é abstrair toda a comunicação com a nuvem AWS. Para usá-lo, é necessário criar uma instância de um objeto “*AWSHandler*” que implementa a interface exibida na **Figura 9**. Ao ser criada, essa

instância utiliza a biblioteca AWS SDK v2 para se comunicar com a API do serviço EC2 e cuida de todo o domínio da lógica para prover os métodos necessários da interface.

Além de facilitar a manutenção e entendimento do código, o uso de interfaces permite que futuramente, caso haja necessidade, a implementação do DevSpaces em outro provedor de cloud (como Azure, Oracle ou Google Cloud Platform) seja possível.

3.2 Módulo CLI

Este módulo em particular desempenha um papel crucial na camada de apresentação da ferramenta, onde ocorre o processamento da entrada fornecida pelo usuário por meio da Interface de Linha de Comando (CLI). A sua implementação se baseia na utilização da biblioteca urfave/cli, a qual se destaca por habilitar a construção de CLIs de forma declarativa. Essa abordagem simplifica significativamente a manutenção do código, tornando-o mais legível e de fácil compreensão.

Ao criar a estrutura de definição do CLI com a biblioteca urfave/cli, um dos benefícios é a geração automática de páginas de ajuda que podem ser acessadas por meio do parâmetro "--help". Isso significa que os usuários da ferramenta têm à disposição uma documentação instantânea que descreve os comandos disponíveis, suas opções e como utilizá-los. Isso é especialmente valioso, pois torna a ferramenta mais amigável e acessível, permitindo que os usuários a explorem e compreendam com facilidade.

```
{
    Name:        "create",
    Description: "Create a the dev space environment automatically.",

    Category: ADM,
    Action:   commands.CreateCommand,
    Flags: []cli.Flag{
        &cli.StringFlag{
            Name:     "name",
            Aliases:  []string{"n"},
            Required: true,
        },
        &cli.StringFlag{
            Name:     "key-name",
            Aliases:  []string{"k"},
            Required: true,
            Usage:    "Name of the SSH key pair to use",
        },
        &cli.StringFlag{
            Name:     "ami",
            Aliases:  []string{"i"},
            Required: true,
```

Figura 11 - Trecho de código utilizando a abordagem declarativa para criação do CLI.

Outro ponto positivo na adoção desta biblioteca é a capacidade de definir valores padrão para os parâmetros do CLI. Isso simplifica a experiência do usuário, pois ele não precisa fornecer valores para todos os parâmetros a cada execução. Os valores padrão podem ser configurados de acordo com as preferências e necessidades da ferramenta, garantindo que a execução dos comandos seja conveniente e eficiente. Em resumo, seu uso nesse módulo não apenas simplifica a construção e manutenção do CLI, mas também aprimora a usabilidade da ferramenta, proporcionando aos usuários uma experiência mais intuitiva e documentação acessível.

Adicionalmente, para cada comando disponível na Interface de Linha de Comando (CLI), é designada uma função específica que desempenha o papel de extrair os parâmetros fornecidos pelo usuário e traduzi-los para o formato esperado pelas interfaces expostas pelo módulo "core". Em situações em que filtros são utilizados, este módulo executa o processamento e a conversão dos filtros em objetos estruturados por meio do emprego de expressões regulares.

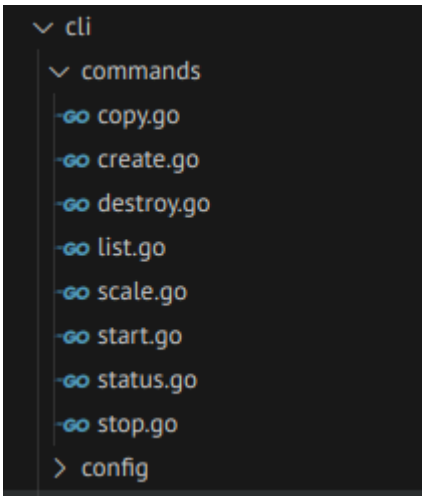

Figura 12 - Organização do código do módulo CLI.

A implementação de cada um dos comandos foi estruturada de forma a criar arquivos individuais dedicados a cada funcionalidade. Essa abordagem visa aprimorar a manutenibilidade da ferramenta, uma vez que a divisão resulta em arquivos menores, contendo menos linhas de código e com um propósito altamente específico. Cada um desses arquivos assume a responsabilidade de capturar os parâmetros fornecidos pelo usuário e, em seguida, chamar os métodos apropriados no módulo "core" para a conclusão bem-sucedida do comando.

Além do pré-processamento da entrada do usuário, este módulo também oferece uma implementação de “logger” (um mecanismo de registro de eventos) destinado a satisfazer a interface requisitada pelo módulo "core". Essa implementação adota o padrão de design Strategy. Essa implementação, chamada de "CLI logger" , é concebida tendo em mente a experiência do usuário e, como tal, frequentemente inclui elementos visuais, como indicadores de progresso ("spinners") ou outros recursos visuais em conjunto com os *logs* de execução.

Por padrão, essa implementação de "logger" suprime a exibição de registros em nível DEBUG, a fim de manter a saída de log mais limpa. No entanto, é possível habilitar a exibição de registros em nível DEBUG definindo a variável de ambiente DEBUG=true. Dessa forma, os usuários têm flexibilidade para controlar o detalhamento das informações de log conforme suas necessidades específicas, ao mesmo tempo em que desfrutam

de uma experiência mais visual e amigável durante a interação com a ferramenta por meio da CLI.

4. AVALIAÇÃO

Foi desenvolvido um questionário com o intuito de coletar feedback e opiniões dos usuários. A avaliação teve como objetivo entender a experiência dos usuários ao utilizar a ferramenta. A metodologia adotada para o questionário baseou-se em uma abordagem qualitativa, centrada na avaliação da usabilidade da ferramenta em relação aos casos de uso específicos, como a criação, inicialização, parada e destruição de ambientes de desenvolvimento. E tópicos qualitativos como integração com fluxo de trabalho, clareza dos status e feedback, documentação, percepção de desempenho e facilidade geral de uso. Cada pergunta foi elaborada para capturar as percepções e experiências dos usuários ao interagir com a ferramenta.

O questionário é composto por uma série de perguntas de múltipla escolha, permitindo que os participantes selecionem a alternativa que melhor represente suas opiniões ou experiências. Contando também com um espaço para comentários e sugestões abertas, de modo a obter um feedback mais detalhado. Esses resultados forneceram insumo tanto para avaliação como para aprimoramento futuro da ferramenta, de modo a garantir uma experiência mais intuitiva e produtiva aos mesmos.

A avaliação da ferramenta apresentou um desafio devido à sua natureza, que envolve a interação direta com a Amazon Web Services (AWS). Para utilizar a ferramenta, é necessário possuir uma conta na AWS e associar um cartão de crédito válido a essa conta. Essa exigência se deve ao fato de que a AWS, como um serviço de computação em nuvem, geralmente cobra pelo uso de recursos, como instâncias de computação e armazenamento.

A necessidade de um cartão de crédito associado à conta criou um obstáculo para algumas pessoas que desejavam avaliar a ferramenta. Além disso, o processo de avaliação poderia, de fato, implicar em gastos financeiros por parte dos avaliadores, dependendo do uso que fizerem da ferramenta. Isso significa que os avaliadores precisam estar cientes de que a avaliação não é isenta de custos e que os gastos associados ao uso da AWS podem variar dependendo do tamanho e da complexidade dos ambientes de desenvolvimento que criarem.

Essa dificuldade financeira potencial limitou a disponibilidade de pessoas dispostas a avaliar a ferramenta, tornando importante considerar os custos envolvidos no processo de avaliação. Portanto, durante a abordagem de avaliação foi feito um alerta e conscientização dos custos associados à utilização da AWS como parte do processo.

O questionário foi criado utilizando a plataforma Google Forms. Um total de seis participantes responderam ao questionário. Todos os participantes conseguiram concluir a instalação da ferramenta com sucesso.

4.1 Tarefa 1: Criar e Iniciar um Ambiente

Na primeira tarefa, os participantes foram instruídos a criar um ambiente remoto e iniciá-lo, sendo fornecidos os links para a documentação do projeto no GitHub. Dos participantes, 83,3% conseguiram criar o ambiente e iniciá-lo com sucesso, enquanto 16,7% não tiveram sucesso nessa tarefa.

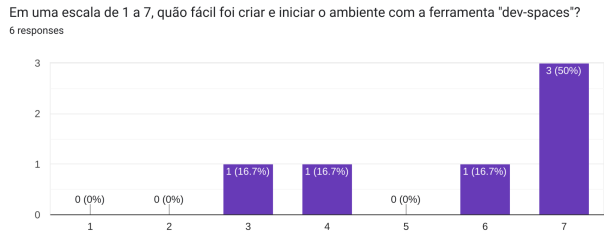

Figura 13 - Pergunta sobre facilidade de uso dos comandos create e start.

Em relação à facilidade de uso da ferramenta, as respostas foram variadas. Foi fornecida uma escala de 1 (Muito difícil) a 7 (Muito fácil) para classificar a facilidade de uso. Cerca de 50% dos participantes classificaram a ferramenta com a pontuação máxima de 7, indicando que a consideraram "Muito fácil". As respostas restantes se dividiram entre as opções 3 (um pouco difícil), 4 (neutro) e 6 (fácil).

Quando questionados sobre a ocorrência de erros ao utilizar os comandos durante essa atividade, 33,3% dos participantes relataram que não encontraram nenhum tipo de erro, indicando uma experiência livre de problemas. No entanto, os restantes 66,7% dos participantes tiveram alguns contratempos e foram solicitados a especificá-los.

Dentre esses últimos, um participante não detalhou quais erros encontrou, mas mencionou que conseguiu resolvê-los utilizando a documentação fornecida. Os outros dois participantes relataram terem recebido mensagens de erro relacionadas a “tentativas máximas excedidas” ao executar o comando "create" durante a criação da frota Spot. Um deles tentou novamente e conseguiu criar o ambiente com sucesso, enquanto o outro não tentou novamente, resultando na impossibilidade de concluir as outras tarefas subsequentes.

Após uma investigação mais aprofundada, ficou claro que o erro de tentativas máximas ocorreu devido a verificações adicionais feitas pela AWS em contas novas ou em situações em que um serviço é utilizado pela primeira vez (nesse caso, o EC2). Geralmente, a AWS envia um e-mail informando que estão sendo realizadas validações adicionais na conta do

usuário. Após a conclusão dessas verificações, a utilização da AWS é normalizada. Para evitar problemas futuros, uma observação foi adicionada à documentação oficial do projeto para alertar os usuários sobre essa situação e fornecer informações sobre como proceder.

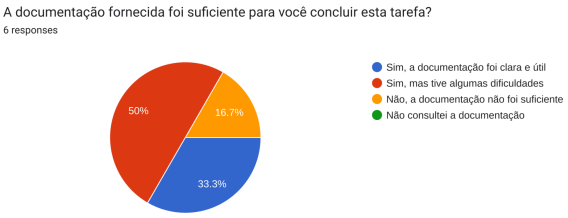

Figura 14 - Pergunta sobre qualidade da documentação.

Em relação à qualidade da documentação (Figura 14), os resultados indicam uma percepção variada entre os participantes. Metade dos respondentes concordou que a documentação foi suficiente para a conclusão da tarefa, mas encontrou algumas dificuldades (50%). Um grupo menor, representando 33% dos participantes, afirmou que a documentação foi clara e útil para eles. No entanto, 16,7% dos participantes indicaram que a documentação não foi suficiente para concluir a tarefa. Essas respostas destacam a importância de continuar aprimorando e esclarecendo a documentação para melhor atender às necessidades dos usuários da ferramenta.

4.2 Tarefa 2: Integrar Ambiente com VSCode

Nesta segunda tarefa, os participantes foram convidados a realizar a integração de um ambiente de desenvolvimento criado com a ferramenta "dev-spaces" com o Visual Studio Code (VSCode), uma das IDEs mais populares entre os desenvolvedores. O objetivo era permitir que os avaliadores experimentassem a integração de suas configurações de desenvolvimento com a ferramenta e explorassem como as funcionalidades do VSCode poderiam ser aplicadas ao ambiente remoto.

A atividade exigia que os participantes seguissem as instruções fornecidas na documentação oficial, que incluía etapas para configurar o VSCode e se conectar ao ambiente remoto criado anteriormente. Os participantes foram encorajados a usar extensões e recursos do VSCode, para explorar a funcionalidade completa do ambiente de desenvolvimento. Os resultados revelaram que a grande maioria (83,3%) conseguiu concluir com êxito essa integração, enquanto 16,7% enfrentaram dificuldades que os impediram de realizar a integração.

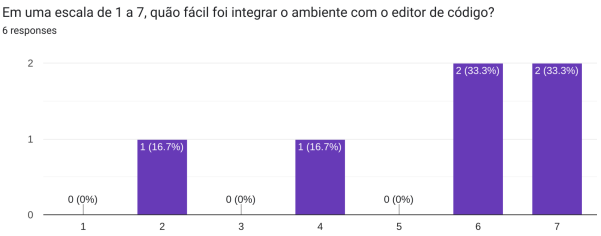

Figura 15 - Pergunta sobre facilidade de integração.

Ao avaliar a facilidade de integração do ambiente de desenvolvimento com o editor de código, os participantes classificaram sua experiência em uma escala de 1 a 7, onde 7 representava a maior facilidade e 1 a menor. Os resultados revelam que a maioria dos participantes (33,3%) atribuiu a pontuação máxima, indicando que a integração foi percebida como muito fácil. Um número igual de participantes (33,3%) classificou a integração com uma nota 6, também demonstrando um alto nível de satisfação com o processo. Por outro lado, 16,7% dos participantes classificaram a integração com uma nota 4, o que sugere uma avaliação neutra. Por fim, 16,7% dos participantes deram a nota 2, indicando que a integração foi percebida como relativamente difícil por esse grupo.

Aqueles que encontraram obstáculos durante a integração descreveram brevemente suas experiências. Um dos participantes relatou que enfrentou problemas para inicializar o ambiente na tarefa anterior, o que afetou sua capacidade de integração. Outro participante destacou que a documentação referente ao parâmetro "IdentityFile" apresentava diferenças entre o link na documentação oficial do projeto e o código comentado dentro do arquivo de configurações SSH. Esse participante recomendou uma padronização para facilitar o processo. Por fim, um participante mencionou que a integração com o editor de código foi fácil, mas devido ao uso do sistema Windows, precisou fazer edições manuais no arquivo de configuração SSH. No entanto, destacou que a documentação foi útil nesse processo.

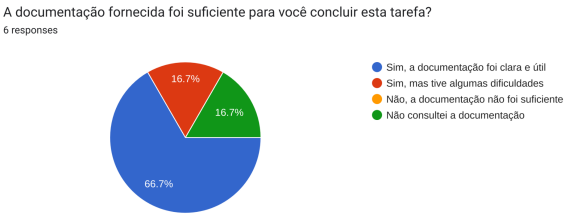

Figura 16 - Pergunta sobre qualidade da documentação.

Em relação à qualidade da documentação, 66,7% dos participantes concordaram que ela foi clara e útil, indicando que a documentação oferecida facilitou a conclusão da tarefa. No entanto, 16,7% dos participantes afirmaram que, embora

tenham conseguido concluir a tarefa, enfrentaram algumas dificuldades. Notavelmente, nenhum dos participantes indicou que a documentação foi insuficiente para a conclusão da tarefa, e 16,7% relataram que não precisaram consultar a documentação.

Essa segunda tarefa demonstrou que a integração do ambiente de desenvolvimento com o Visual Studio Code foi viável para a maioria dos participantes, mas algumas questões específicas em relação à documentação e problemas relacionados a tarefas anteriores afetaram a experiência de alguns deles. Esses insights podem contribuir para aprimoramentos futuros na documentação e na experiência geral do usuário.

4.3 Tarefa 3: Parar e destruir um Ambiente

Nesta etapa, os participantes foram instruídos a utilizar os comandos "stop" e "destroy" da ferramenta "dev-spaces" para parar e destruir o ambiente de desenvolvimento. Os resultados demonstram que todos os participantes (100%) conseguiram realizar essa tarefa com sucesso, indicando um alto grau de eficácia da ferramenta na execução dessas operações.

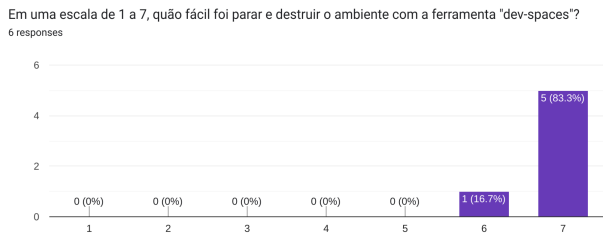

Figura 17 - Pergunta sobre a facilidade de finalização do ambiente.

Quando questionados sobre a facilidade dessas ações, a maioria dos participantes (83,3%) classificou a experiência com a nota máxima, ou seja, atribuíram a pontuação 7, indicando que parar e destruir o ambiente foi percebido como muito fácil. Os restantes 16,7% dos participantes classificaram a tarefa com a nota 6, o que ainda representa uma avaliação positiva.

É relevante notar que nenhum participante encontrou erros significativos ao tentar parar ou destruir o ambiente. As respostas coletadas enfatizaram a facilidade da tarefa, com alguns participantes mencionando explicitamente que essa foi a parte mais fácil do processo.

Quanto à documentação, todos os participantes (100%) consideraram que ela foi clara e útil para concluir essa tarefa. Essa avaliação positiva da documentação reflete a eficácia das informações fornecidas para orientar os usuários na execução dessas ações.

Esses resultados demonstram que a tarefa de parar e destruir um ambiente usando a ferramenta "dev-spaces" foi realizada com sucesso, com a maioria dos participantes classificando-a como muito fácil e encontrando a documentação suficiente e útil para orientá-los nesse processo.

4.4 Feedback Geral

Nesta seção, os participantes foram convidados a fornecer feedback geral sobre a ferramenta "dev-spaces" e a experiência de uso. As respostas revelam uma avaliação geral positiva em relação à documentação, desempenho, opções de personalização e facilidade de uso da ferramenta.

Em relação à documentação, metade dos participantes (50%) classificou-a como "muito útil" (pontuação 7 em uma escala de 1 a 7), destacando a utilidade das informações fornecidas para compreender as funcionalidades da ferramenta e como usá-la efetivamente. Outros 33,3% a consideraram "útil" (pontuação 6), e 16,7% a avaliaram de forma neutra (pontuação 4).

Quando questionados sobre problemas de desempenho ou lentidão durante a gestão dos ambientes na nuvem, a grande maioria dos participantes (83,3%) relatou não ter enfrentado problemas significativos nesse sentido. No entanto, 16,7% mencionaram que não puderam gerenciar os ambientes na nuvem devido a dificuldades na inicialização.

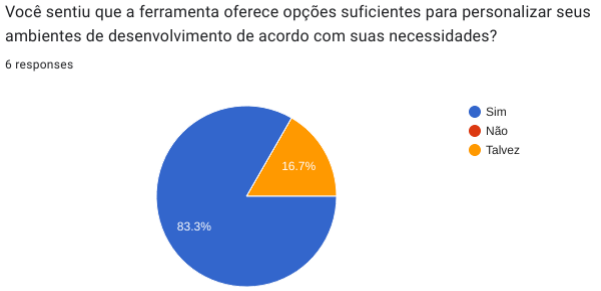

Figura 18 - Pergunta sobre flexibilidade da ferramenta.

Sobre as opções de personalização, a maioria dos participantes (83,3%) concordou que a ferramenta oferece opções suficientes para personalizar os ambientes de desenvolvimento de acordo com suas necessidades, enquanto 16,7% expressaram uma resposta mais cautelosa, indicando que talvez haja margem para melhorias nessa área.

A avaliação da facilidade de uso e usabilidade geral da ferramenta "dev-spaces" revelou uma experiência geralmente positiva. Um terço dos participantes (33,3%) classificou a facilidade de uso como "muito fácil" (pontuação 7), metade dos participantes (50%) a considerou "fácil" (pontuação 6) e (16.7%) considerou “um pouco fácil” (pontuação 5). Os demais participantes não atribuíram pontuações mais baixas, indicando que a ferramenta foi percebida de modo geral como de fácil utilização em sua experiência.

Os participantes ainda compartilharam feedback valioso e sugestões para aprimorar ainda mais a usabilidade da

ferramenta "dev-spaces". Dois participantes enfatizaram a importância de adicionar exemplos adicionais e revisar a documentação para torná-la ainda mais amigável, incluindo links para os tipos de instâncias da AWS, mais assistência na solução dos erros comuns, especialmente para aqueles com conhecimento mínimo no assunto.

Outro participante expressou a necessidade de melhorar o suporte ao ambiente Windows, e que poderia ser criada uma extensão do VSCode provendo as funcionalidades da ferramenta. Além disso, ele mencionou a importância de considerar o suporte a outras nuvens, como GCP e Azure, para cobrir cenários de testes e benchmarking que envolvem criação de máquinas em múltiplas nuvens. Além disso, ele sugeriu explorar funcionalidades adicionais relacionadas ao gerenciamento de redes, auto scaling e load balancing. A integração com arquivos de configuração de infraestrutura como código (IaC), como CloudFormation e Terraform, também foi sugerida.

Além disso, houve sugestões para automatizar o processo de importação de pares de chaves SSH quando o usuário passa como parâmetro o caminho para uma chave SSH local ao invés do nome de uma chave criada manualmente, como é feito hoje.

Os participantes também forneceram feedback positivo, elogiando a ferramenta por sua facilidade de uso e seu potencial para otimizar o gerenciamento de instâncias. A agilidade proporcionada pela ferramenta foi ressaltada como um ponto forte.

Em resumo, o feedback e as sugestões dos participantes destacam oportunidades para aprimorar a documentação, expandir o suporte da ferramenta para diferentes plataformas e nuvens, e simplificar a experiência de uso, de modo a torná-la ainda mais acessível e valiosa para um público diversificado.

5. TRABALHOS FUTUROS

Em relação aos trabalhos futuros, existem várias áreas de aprimoramento e expansão para a ferramenta "dev-spaces". Uma das melhorias mais destacadas pelos usuários é o aprimoramento do suporte para Windows, tornando a ferramenta ainda mais acessível a desenvolvedores que utilizam essa plataforma.

Além disso, há um interesse claro em expandir os recursos de gerenciamento da rede. A possibilidade de abrir e fechar portas diretamente no *SecurityGroup* do ambiente a partir do CLI seria uma adição valiosa para a ferramenta.

A capacidade de escalar os volumes, aumentando seu tamanho de maneira simples, é outra área que pode ser explorada. Isso proporcionaria maior flexibilidade aos desenvolvedores na gestão dos recursos em nuvem.

Para tornar a ferramenta mais amigável e acessível a diversos tipos de desenvolvedores, a ideia de implementar uma versão baseada na web pode ser considerada. Isso permitiria que aqueles sem experiência em AWS ou que não possuam uma conta AWS gerenciem ambientes de desenvolvimento de forma mais intuitiva.

Além disso, expandir o suporte para outros provedores de nuvem, como GCP (Google Cloud Platform) e Azure, é uma perspectiva interessante. Isso possibilitaria que os usuários escolhessem o provedor de nuvem de acordo com suas preferências ou necessidades específicas.

Em resumo, os trabalhos futuros envolvem melhorias na usabilidade, expansão de recursos e maior flexibilidade para acomodar diferentes perfis de desenvolvedores e preferências de plataforma de nuvem.

6. REFERÊNCIAS

[1] O que é uma instância em computação em nuvem? – Explicação sobre instâncias de nuvem – AWS. Disponível em: <https://aws.amazon.com/pt/what-is/cloud-instances/>. Acesso em: 23 out. 2023.

[2] Visual Studio Code Remote Development. Disponível em: <https://code.visualstudio.com/docs/remote/remote-overview>.

[3] About billing for GitHub Codespaces. Disponível em: <https://docs.github.com/en/billing/managing-billing-for-github-codespaces/about-billing-for-github-codespaces#pricing-for-paid-usage>. Acesso em: 23 out. 2023.

[4] How Spot Instances work - Amazon Elastic Compute Cloud. Disponível em: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/how-spot-instances-work.html>. Acesso em: 23 out. 2023.

[5] systemd-nspawn. Disponível em: <https://www.freedesktop.org/software/systemd/man/systemd-nspawn.html>. Acesso em: 23 out. 2023.

[6] chroot(1) - Linux manual page. Disponível em: <https://man7.org/linux/man-pages/man1/chroot.1.html>. Acesso em: 23 out. 2023.

[7] Ebert, C. and Duarte, C.H.C., 2018. Digital transformation. *IEEE Softw.*, *35*(4), pp.16-21

[8] S. Yu and S. Zhou, "A survey on metric of software complexity," *2010 2nd IEEE International Conference on Information Management and Engineering*, Chengdu, China, 2010, pp. 352-356, doi: 10.1109/ICIME.2010.5477581.

[9] Karunakaran, S. (2013). Impact of Cloud Adoption on Agile Software Development. In: Mahmood, Z., Saeed, S. (eds) Software Engineering Frameworks for the Cloud Computing Paradigm. Computer Communications and Networks. Springer, London. https://doi.org/10.1007/978-1-4471-5031-2_10

[10] David J. Malan, Jonathan Carter, Rongxin Liu, and Carter Zenke. 2023. Providing Students with Standardized, Cloud-Based Programming Environments at Term's Start (for Free). In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 1183. https://doi.org/10.1145/3545947.3569611

[11] David J. Malan. 2022. Standardizing Students' Programming Environments with Docker Containers: Using

Visual Studio Code in the Cloud with GitHub Codespaces. In Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 2 (ITiCSE '22). Association for Computing Machinery, New York, NY, USA, 599–600. https://doi.org/10.1145/3502717.3532164
[12] New T3 Instances – Burstable, Cost-Effective Performance | AWS News Blog. Disponível em: <https://aws.amazon.com/blogs/aws/new-t3-instances-burstable-cost-effective-performance/>. Acesso em: 23 out. 2023.
[13] Bell System Technical Journal, 57: 6. July-August 1978 pp 1899-1904. UNIX Time-Sharing System: Forward. (McIlroy, M.D.; Pinson, E.N.; Tague, B.A.)