INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS PIRIPIRI
TECNÓLOGO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PEDRO HENRIQUE VALENTINO SILVA SOUSA

ANÁLISE COMPARATIVA DE SERVIÇOS BÁSICOS DE INFRAESTRUTURA COMO SERVIÇO (IAAS) NOS PLANOS GRATUITOS DA AWS, DO AZURE E DA GCP

PIRIPIRI/PI
2026

PEDRO HENRIQUE VALENTINO SILVA SOUSA

ANÁLISE COMPARATIVA DE SERVIÇOS BÁSICOS DE INFRAESTRUTURA COMO SERVIÇO (IAAS) NOS PLANOS GRATUITOS DA AWS, DO AZURE E DA GCP

Trabalho de Conclusão de Curso (artigo científico) apresentado como exigência parcial para obtenção do diploma do Curso de Tecnólogo em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Piripiri.

Orientador: Prof. Me. Jonathas Jivago de Almeida Cruz.

PIRIPIRI/PI

2026

ANÁLISE COMPARATIVA DE SERVIÇOS BÁSICOS DE INFRAESTRUTURA COMO SERVIÇO (IAAS) NOS PLANOS GRATUITOS DA AWS, DO AZURE E DA GCP

Pedro Henrique Valentino Silva Sousa[1]
Prof. Me. Jonathas Jivago de Almeida Cruz[2]

RESUMO

Este trabalho compara serviços básicos de infraestrutura como serviço disponibilizados nos planos gratuitos da Amazon Web Services, Microsoft Azure e Google Cloud Platform. A pesquisa foi aplicada, experimental e comparativa, com abordagem predominantemente quantitativa. Uma mesma aplicação de servidor foi implantada em máquinas virtuais de entrada nas três plataformas e submetida a três rodadas padronizadas, compostas pelas etapas de aquecimento, carga controlada e recuperação. Foram analisados o tempo médio de resposta, a utilização do processador, a taxa de erro, o tráfego de rede, a disponibilidade de métricas nativas, a granularidade do monitoramento e o esforço operacional necessário para implantação, atualização e reversão. Todas as plataformas processaram a carga leve sem erros nas requisições. No cenário avaliado, a Amazon Web Services apresentou o menor tempo médio de resposta; a Microsoft Azure ofereceu o conjunto mais abrangente de métricas nativas e o menor tempo de reversão; e a Google Cloud Platform apresentou o menor tempo de implantação inicial. Concluiu-se que nenhuma plataforma foi superior em todos os critérios, pois cada uma apresentou vantagens contextuais que podem orientar estudantes, desenvolvedores e docentes conforme as prioridades do projeto ou da atividade prática.

Palavras-chave: computação em nuvem; infraestrutura como serviço; máquinas virtuais; monitoramento; planos gratuitos.

ABSTRACT

This study compares basic infrastructure-as-a-service resources available in the free tiers of Amazon Web Services, Microsoft Azure, and Google Cloud Platform. The research was applied, experimental, and comparative, with a predominantly quantitative approach. The same backend application was deployed on entry-level virtual machines across the three platforms and subjected to three standardized rounds consisting of warm-up, controlled load, and recovery phases. The analysis considered average response time, processor utilization, error rate, network traffic, availability of native metrics, monitoring granularity, and the operational effort required for deployment, update, and rollback. All platforms processed the light workload without request errors. In the evaluated scenario, Amazon Web Services achieved the lowest average response time; Microsoft Azure provided the most comprehensive set of native metrics and the shortest rollback time; and Google Cloud Platform achieved the shortest initial deployment time. The results indicate that no platform outperformed the others across all criteria, as each presented contextual

---

[1] Discente do curso de Análise e Desenvolvimento de Sistemas (IFPI).
[2] Docente orientador do IFPI - Campus Piripiri.

advantages that may guide students, developers, and instructors according to the priorities of a project or practical activity.

Keywords: cloud computing; infrastructure as a service; virtual machines; monitoring; free tier.

Data de aprovação: 06/07/2026

1 INTRODUÇÃO

A computação em nuvem consolidou-se como uma abordagem amplamente utilizada para disponibilização de recursos computacionais sob demanda, permitindo o uso remoto de infraestrutura, plataformas e aplicações por meio de redes de comunicação. Esse modelo é sustentado por recursos como virtualização, provisionamento dinâmico, compartilhamento de infraestrutura e acesso remoto aos serviços, características que favorecem maior flexibilidade no uso de recursos computacionais (MELL; GRANCE, 2011; ERL; PUTTINI; MAHMOOD, 2013).

Entre os modelos de serviço da computação em nuvem, a Infraestrutura como Serviço (*Infrastructure as a Service* – IaaS) corresponde ao fornecimento de recursos computacionais fundamentais, como máquinas virtuais, armazenamento e rede. Nesse modelo, o usuário mantém controle sobre o sistema operacional e sobre as aplicações implantadas, enquanto a infraestrutura física permanece sob responsabilidade do provedor (MELL; GRANCE, 2011; ERL; PUTTINI; MAHMOOD, 2013). Esse recorte é relevante para estudantes, desenvolvedores e pequenos projetos, pois permite a implantação de aplicações sem a aquisição de servidores próprios e com maior controle sobre o ambiente computacional utilizado.

No contexto da nuvem pública, Amazon Web Services (AWS), Microsoft Azure e Google Cloud Platform (GCP) oferecem serviços de infraestrutura com finalidades semelhantes, mas organizados por meio de nomenclaturas, recursos, mecanismos de configuração e ferramentas de monitoramento próprios. Estudos comparativos apontam que esses provedores apresentam diferenças quanto à organização dos serviços, à arquitetura, aos modelos de operação e à forma como os recursos são disponibilizados ao usuário (SARASWAT; TRIPATHI, 2020; HYSENI; IBRAHIMI, 2017). Tais diferenças podem influenciar a experiência prática de implantação, acompanhamento e manutenção de aplicações.

Apesar da ampla disponibilidade de documentação técnica, as informações fornecidas pelos provedores geralmente são apresentadas de forma isolada, seguindo a estrutura e a terminologia de cada plataforma. Essa característica pode dificultar comparações diretas entre serviços funcionalmente semelhantes, especialmente quando se consideram ambientes gratuitos ou de entrada. Além disso, a avaliação de desempenho em nuvem envolve fatores como compartilhamento de recursos, configuração das instâncias, localização geográfica, granularidade das métricas e metodologia de medição, aspectos que exigem experimentação prática e interpretação cuidadosa dos resultados (KHANGHAHI; RAVANMEHR, 2013; CASALE, 2023).

Diante desse contexto, o problema central desta pesquisa consistiu em analisar o comportamento de serviços básicos de IaaS da AWS, do Azure e da GCP em seus planos gratuitos, considerando um cenário experimental controlado. A investigação concentrou-se em três dimensões: desempenho de máquinas virtuais de entrada, monitoramento nativo e esforço operacional associado ao ciclo de vida de uma API backend. No desempenho, foram avaliados o tempo médio de resposta, a utilização média e máxima de CPU, a taxa de erro HTTP e o tráfego de rede observado durante as rodadas experimentais.

A realização deste estudo justifica-se pelo impacto que a escolha de um provedor de nuvem pode exercer sobre aspectos técnicos, operacionais e financeiros de um projeto. Essa escolha não envolve apenas o custo de uso, mas também a facilidade de configuração, a disponibilidade de métricas, a granularidade do monitoramento, o comportamento da aplicação sob carga, o esforço necessário para implantação e a possibilidade de atualizar ou reverter versões. Em projetos acadêmicos, protótipos e aplicações de pequeno porte, esses fatores tornam-se especialmente relevantes, pois os recursos disponíveis nos planos gratuitos podem orientar decisões iniciais de adoção tecnológica.

O objetivo geral deste trabalho foi comparar, de forma prática, serviços básicos de IaaS oferecidos nos planos gratuitos da AWS, do Azure e da GCP. Especificamente, buscou-se avaliar o desempenho e a utilização de recursos de máquinas virtuais de entrada, analisar as métricas disponibilizadas pelas ferramentas nativas de monitoramento e comparar os procedimentos de implantação, atualização e reversão de uma API backend. Dessa forma, o estudo pretende contribuir com uma análise experimental acessível, documentada e

reproduzível, capaz de auxiliar estudantes, desenvolvedores e pequenos projetos na compreensão das diferenças práticas entre os provedores avaliados.

A pesquisa foi aplicada, experimental e comparativa, com abordagem predominantemente quantitativa. A mesma API backend foi implantada nos três provedores e submetida a rodadas controladas de requisições, enquanto as métricas de infraestrutura foram coletadas pelos serviços nativos de monitoramento de cada plataforma. Os resultados foram analisados considerando as limitações de equivalência entre os ambientes, uma vez que as instâncias, regiões, mecanismos de agregação e granularidade das métricas não eram absolutamente idênticos.

2 REFERENCIAL TEÓRICO

Este referencial teórico apresenta os conceitos necessários para fundamentar a comparação realizada neste trabalho. Inicialmente, são discutidos os princípios da computação em nuvem e seus principais modelos de serviço e implantação. Em seguida, são descritas as plataformas analisadas e os estudos relacionados que contribuem para compreender a avaliação de desempenho, monitoramento e esforço operacional em ambientes de nuvem pública.

2.1 COMPUTAÇÃO EM NUVEM

A computação em nuvem pode ser compreendida como um modelo de fornecimento de recursos computacionais por meio da rede, permitindo o acesso sob demanda a serviços como processamento, armazenamento, aplicações e infraestrutura. Esse modelo reduz a necessidade de aquisição direta de equipamentos físicos e permite que usuários e organizações utilizem recursos provisionados e administrados por provedores especializados (MELL; GRANCE, 2011; ERL; PUTTINI; MAHMOOD, 2013).

Do ponto de vista arquitetural, a computação em nuvem é sustentada por tecnologias e mecanismos como virtualização, abstração de recursos, provisionamento automatizado, compartilhamento de infraestrutura e acesso remoto aos serviços. Esses elementos permitem que recursos físicos sejam organizados em ambientes lógicos, disponibilizados a diferentes usuários de forma dinâmica e escalável (ERL; PUTTINI; MAHMOOD, 2013).

Segundo Mell e Grance (2011), a computação em nuvem apresenta cinco características essenciais: autosserviço sob demanda, amplo acesso pela rede, agrupamento de recursos, elasticidade rápida e serviço mensurado. O autosserviço permite que o usuário provisione recursos sem interação humana direta com o provedor. O amplo acesso pela rede possibilita a utilização dos serviços por diferentes dispositivos e mecanismos padronizados de comunicação.

O agrupamento de recursos refere-se ao compartilhamento da infraestrutura física e virtual entre diversos consumidores, com alocação dinâmica conforme a demanda. A elasticidade possibilita ampliar ou reduzir rapidamente a capacidade disponível, enquanto o serviço mensurado permite monitorar, controlar e registrar o consumo de recursos, como processamento, armazenamento e tráfego de rede (MELL; GRANCE, 2011).

Entre os principais benefícios da computação em nuvem estão a redução da necessidade de investimento inicial em infraestrutura, a rapidez de provisionamento, a flexibilidade operacional e a possibilidade de ajustar os recursos conforme a demanda. Esse modelo também favorece a experimentação de tecnologias e a implantação de aplicações por estudantes, desenvolvedores e organizações que não dispõem de infraestrutura própria de grande porte (ERL; PUTTINI; MAHMOOD, 2013).

Entretanto, sua adoção também envolve limitações e riscos. A dependência de conectividade, a possibilidade de indisponibilidade do provedor, a variação de desempenho em recursos compartilhados, a dificuldade de estimar custos e a dependência tecnológica de serviços específicos podem influenciar a escolha e a utilização de uma plataforma. Além disso, aspectos relacionados à segurança, à privacidade e à localização dos dados devem ser considerados conforme as características da aplicação (KHANGHAHI; RAVANMEHR, 2013; CASALE, 2023).

Dessa forma, a avaliação de plataformas de computação em nuvem deve considerar não apenas a disponibilidade de recursos, mas também fatores como desempenho, observabilidade, facilidade de implantação, esforço operacional e limitações do ambiente utilizado. No caso deste trabalho, essa avaliação é delimitada aos serviços básicos de infraestrutura disponibilizados nos planos gratuitos da AWS, do Azure e da GCP.

2.2 MODELOS DE SERVIÇO E DE IMPLANTAÇÃO

Os serviços de computação em nuvem podem ser classificados conforme o nível de abstração oferecido ao usuário e a divisão de responsabilidades entre cliente e provedor. Entre os modelos mais consolidados estão a Infraestrutura como Serviço (*Infrastructure as a Service* – IaaS), a Plataforma como Serviço (*Platform as a Service* – PaaS) e o Software como Serviço (*Software as a Service* – SaaS) (MELL; GRANCE, 2011; ERL; PUTTINI; MAHMOOD, 2013).

No modelo IaaS, o provedor disponibiliza recursos fundamentais de infraestrutura, como processamento, armazenamento e rede. O usuário pode implantar sistemas operacionais, configurar o ambiente e executar suas aplicações, mantendo maior controle sobre a máquina virtual e seus componentes lógicos, embora não administre a infraestrutura física subjacente. Serviços como Amazon EC2, Azure Virtual Machines e Google Compute Engine são exemplos desse modelo, pois permitem a criação e o gerenciamento de máquinas virtuais em ambientes de nuvem pública (Amazon Web Services, 2026b; Microsoft, 2026b; Google Cloud, 2026b).

No modelo PaaS, o provedor oferece uma plataforma pronta para desenvolvimento, implantação e execução de aplicações. Nesse caso, o usuário concentra-se principalmente no código da aplicação, enquanto o provedor gerencia a infraestrutura, o sistema operacional, o ambiente de execução e parte das configurações necessárias para manter o serviço funcionando. Em termos práticos, esse modelo pode ser associado a plataformas nas quais o desenvolvedor envia o código e a própria plataforma cuida da execução da aplicação, como Heroku, Render, Railway, Vercel e Netlify. Também existem serviços PaaS oferecidos pela Azure, GCP e AWS, como Azure App Service, Google App Engine e AWS Elastic Beanstalk.

No modelo SaaS, aplicações completas são disponibilizadas ao usuário final por meio da internet. O provedor administra a infraestrutura, a plataforma, a manutenção e o próprio software, enquanto o consumidor utiliza as funcionalidades oferecidas, normalmente por navegador ou aplicativo cliente. Exemplos comuns de SaaS incluem ferramentas como Microsoft 365, Google Workspace e Salesforce.

Além dos modelos clássicos, a evolução das arquiteturas em nuvem originou categorias mais específicas. A Função como Serviço (*Function as a Service* – FaaS) permite executar funções acionadas por eventos sem que o usuário gerencie diretamente servidores ou ambientes de execução. Esse modelo é frequentemente

associado a arquiteturas *serverless*, nas quais o provedor realiza o provisionamento, a escalabilidade e parte da administração do ambiente de execução (BALDINI et al., 2017).

Os Contêineres como Serviço (*Containers as a Service* – CaaS) oferecem ambientes para implantação, execução e gerenciamento de aplicações empacotadas em contêineres. Esse modelo situa-se entre IaaS e PaaS, pois abstrai parte da infraestrutura, mas preserva ao usuário maior controle sobre o empacotamento, a configuração e a execução das aplicações (SENEL et al., 2023).

Quanto à forma de implantação, a nuvem pública é disponibilizada para uso amplo e operada por um provedor externo. A nuvem privada é destinada exclusivamente a uma organização, podendo ser administrada internamente ou por terceiros. A nuvem comunitária atende a organizações com requisitos compartilhados, enquanto a nuvem híbrida combina duas ou mais infraestruturas distintas, permitindo a integração e a movimentação de dados e aplicações entre elas (MELL; GRANCE, 2011; ERL; PUTTINI; MAHMOOD, 2013).

O presente estudo concentra-se no modelo IaaS e em ambientes de nuvem pública. Essa delimitação decorre da utilização de máquinas virtuais, armazenamento, recursos de rede e ferramentas de monitoramento disponibilizados por AWS, Microsoft Azure e GCP. Embora a API backend seja utilizada para geração de carga controlada, o objeto principal da análise corresponde à infraestrutura computacional e aos procedimentos operacionais associados à sua utilização.

2.3 PLATAFORMAS DE COMPUTAÇÃO EM NUVEM ANALISADAS

Amazon Web Services (AWS), Microsoft Azure e Google Cloud Platform (GCP) são provedores de nuvem pública que disponibilizam diferentes serviços de infraestrutura, plataforma, armazenamento, monitoramento, banco de dados, inteligência artificial e desenvolvimento de aplicações. Embora possuam portfólios amplos, o presente estudo não compara as plataformas em sua totalidade, mas apenas serviços básicos associados ao modelo de Infraestrutura como Serviço (IaaS), especialmente máquinas virtuais e ferramentas nativas de monitoramento.

Na AWS, o serviço de computação analisado foi o Amazon Elastic Compute Cloud (Amazon EC2), que permite criar e gerenciar servidores virtuais denominados instâncias. O tipo de instância escolhido define características como processamento,

memória, armazenamento e capacidade de rede. As instâncias podem ser associadas a imagens de sistema operacional, volumes de armazenamento, pares de chaves e grupos de segurança. O monitoramento dos recursos foi realizado por meio do Amazon CloudWatch, serviço que coleta métricas e permite acompanhar a utilização e o estado operacional da infraestrutura (Amazon Web Services, 2026b; Amazon Web Services, 2026a).

Na Microsoft Azure, o serviço de computação analisado foi o Azure Virtual Machines, que permite criar máquinas virtuais Linux ou Windows com diferentes configurações de processamento, memória, armazenamento e rede. Sua implantação envolve recursos associados, como grupo de recursos, interface de rede, endereço IP público, disco e regras de segurança por meio do Network Security Group. O monitoramento foi realizado com o Azure Monitor, serviço que reúne métricas e informações relacionadas ao desempenho e à integridade dos recursos implantados (Microsoft, 2026b; Microsoft, 2026a).

Na Google Cloud Platform, o serviço de computação analisado foi o Compute Engine, que permite criar instâncias de máquinas virtuais a partir de tipos de máquina, imagens de sistema operacional, discos e recursos de rede. As instâncias são implantadas em regiões e zonas selecionadas pelo usuário, podendo utilizar diferentes configurações de processamento e memória. O acompanhamento das métricas foi realizado por meio do Google Cloud Monitoring, serviço que permite consultar séries temporais, métricas e informações operacionais dos recursos da plataforma (Google Cloud, 2026b; Google Cloud, 2026a).

A escolha dessas plataformas decorreu da disponibilidade de serviços funcionalmente comparáveis de máquinas virtuais, da existência de ferramentas nativas de monitoramento e da possibilidade de utilização de recursos básicos nos respectivos planos gratuitos. Ainda assim, os serviços não são idênticos entre si. Cada provedor adota nomenclaturas, estruturas de configuração, modelos de instância, formas de agregação de métricas e mecanismos de controle próprios, o que justifica a realização de uma comparação experimental sob procedimentos padronizados.

Dessa forma, a análise desenvolvida neste trabalho deve ser compreendida como uma comparação entre serviços básicos de infraestrutura utilizados para hospedar e monitorar uma API backend, e não como uma avaliação ampla de todos os recursos oferecidos por AWS, Azure e GCP.

2.4 ESTUDOS RELACIONADOS

A literatura sobre avaliação de plataformas de computação em nuvem apresenta diferentes abordagens, incluindo comparações descritivas entre provedores, estudos experimentais de desempenho e trabalhos voltados à definição de métricas e desafios metodológicos. Essas pesquisas contribuem para compreender características técnicas das plataformas, mas nem sempre combinam, em um mesmo estudo, implantação prática de aplicação, monitoramento nativo e procedimentos operacionais.

No âmbito experimental, Kaushik, Sharma e Gupta (2021) compararam AWS, Microsoft Azure e Google Cloud com base em serviços e desempenho, evidenciando a viabilidade de análises quantitativas entre provedores comerciais. Esse tipo de estudo contribui para a definição de critérios de comparação e para a observação de diferenças práticas entre plataformas. Entretanto, abordagens centradas apenas em testes de desempenho ou benchmarks tendem a avaliar aspectos específicos do ambiente, sem necessariamente contemplar o ciclo completo de implantação, atualização, reversão e monitoramento de uma aplicação em operação.

Em uma perspectiva descritiva e arquitetural, Saraswat e Tripathi (2020) analisaram provedores de computação em nuvem considerando características, serviços e modelos de oferta. De modo semelhante, Hyseni e Ibrahimi (2017) discutiram diferenças entre plataformas quanto à organização dos serviços e à estrutura disponibilizada aos usuários. Esses trabalhos auxiliam na compreensão geral dos provedores, mas se concentram principalmente na comparação conceitual e descritiva, sem executar uma aplicação sob condições experimentais controladas.

Trabalhos de natureza metodológica também são relevantes para este estudo. Khanghahi e Ravanmehr (2013) discutem desafios da avaliação de desempenho em ambientes de nuvem, destacando que fatores como compartilhamento de recursos, configuração dos ambientes e variabilidade da infraestrutura podem dificultar a interpretação dos resultados. Casale (2023), por sua vez, ressalta a importância de aproximar o ensino e a prática de avaliação de desempenho das condições contemporâneas de computação em nuvem, considerando métricas, monitoramento e comportamento de sistemas reais.

A partir desses estudos, observa-se que a comparação entre provedores de nuvem pode ser realizada por diferentes perspectivas. Comparações descritivas

ajudam a compreender a organização dos serviços, enquanto estudos experimentais permitem observar o comportamento dos recursos sob carga. Entretanto, ainda há espaço para trabalhos que combinem uma aplicação funcional, recursos básicos disponíveis em planos gratuitos, ferramentas nativas de monitoramento e procedimentos relacionados ao ciclo de vida da aplicação.

Nesse contexto, o presente trabalho diferencia-se por comparar serviços básicos de Infraestrutura como Serviço (*Infrastructure as a Service* – IaaS) da AWS, do Azure e da GCP a partir da implantação de uma API backend em máquinas virtuais de entrada. A análise considera não apenas métricas de desempenho, como tempo de resposta, utilização de CPU, taxa de erro e tráfego de rede, mas também a disponibilidade de métricas nativas, a granularidade do monitoramento e o esforço operacional necessário para implantação, atualização e reversão da aplicação. Dessa forma, busca-se complementar os estudos anteriores com uma avaliação prática, acessível e delimitada aos recursos gratuitos utilizados no experimento.

3 METODOLOGIA

A metodologia adotada neste trabalho possuiu caráter aplicado, experimental e comparativo, com abordagem predominantemente quantitativa. O estudo fundamentou-se na execução de testes práticos em ambientes reais de computação em nuvem, utilizando serviços básicos de Infraestrutura como Serviço (IaaS) disponibilizados nos planos de *free tier* da AWS, do Azure e da GCP. A comparação foi delimitada a recursos de entrada, selecionados conforme disponibilidade gratuita, viabilidade de implantação da API backend e possibilidade de coleta de métricas por ferramentas nativas de monitoramento.

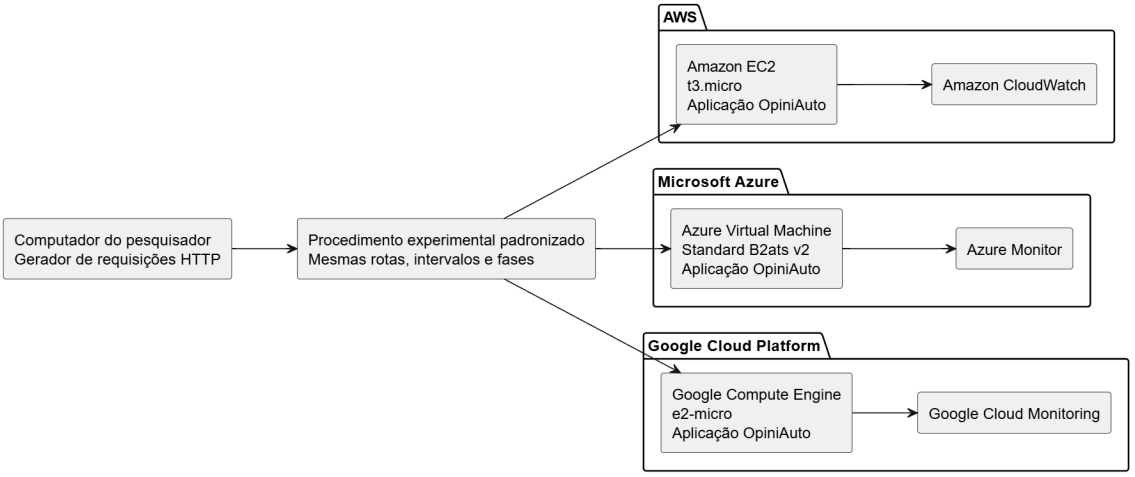
3.1 DEFINIÇÃO DO ESCOPO EXPERIMENTAL

O estudo foi delimitado a três dimensões: serviço de computação, monitoramento nativo e gerenciamento do ciclo de vida de uma API backend. A Figura 1 apresenta a organização geral do ambiente experimental. O computador do pesquisador foi utilizado para gerar requisições HTTP padronizadas, enquanto a API OpiniAuto foi implantada separadamente em máquinas virtuais da AWS, do Azure e da GCP. As métricas de infraestrutura foram coletadas pelos respectivos serviços nativos de monitoramento.

A dimensão de computação correspondeu ao uso de máquinas virtuais de entrada para hospedagem da API. A dimensão de monitoramento considerou a disponibilidade de métricas de CPU, memória, rede e disco, bem como a granularidade temporal e a necessidade de configurações adicionais. A dimensão operacional avaliou os procedimentos de implantação inicial, atualização e reversão da aplicação, considerando tempo de execução, macroetapas, configurações manuais e ocorrência de erros ou retrabalho.

Foram excluídos do escopo serviços avançados, como bancos de dados gerenciados, soluções de *big data*, aprendizado de máquina, funções *serverless*, contêineres, mecanismos pagos de escalabilidade automática e análises de custo além dos limites do *free tier*. Essa delimitação buscou preservar a simplicidade, a acessibilidade e a comparabilidade do experimento, mantendo o foco nos serviços básicos de IaaS utilizados para hospedar, monitorar e operar a API.

Figura 1 – Estrutura geral do ambiente experimental utilizado na comparação entre AWS, Microsoft Azure e Google Cloud Platform



Fonte: Elaborado pelo autor (2026).

3.2 PLATAFORMAS ANALISADAS

Foram analisadas Amazon Web Services (AWS), Microsoft Azure (Azure) e Google Cloud Platform (GCP), selecionadas por sua ampla adoção e pela disponibilidade de serviços de computação, monitoramento e implantação compatíveis com o escopo da pesquisa. A comparação considerou recursos de entrada acessíveis nos planos gratuitos e as limitações de cota, região e disponibilidade existentes no momento da criação dos ambientes.

3.3 SERVIÇO DE COMPUTAÇÃO

O serviço de computação correspondeu ao recurso de infraestrutura utilizado para hospedar a API backend em cada provedor. Em todos os ambientes, foram utilizadas máquinas virtuais de entrada, selecionadas conforme a disponibilidade nos planos de *free tier*, a viabilidade de execução da aplicação e a possibilidade de coleta de métricas por meio das ferramentas nativas de monitoramento.

A caracterização dos ambientes computacionais utilizados é apresentada na Tabela 1. As instâncias escolhidas possuíam porte aproximado, com cerca de 1 GiB de memória, mas não apresentavam equivalência absoluta. Essa diferença decorreu das próprias opções disponibilizadas por cada provedor em seus planos gratuitos, que não oferecem configurações idênticas de processamento, armazenamento, sistema operacional, região e zona de disponibilidade.

Na AWS, foi utilizada a instância t3.micro; na Azure, a instância Standard B2ats v2; e, na GCP, a instância e2-micro. Enquanto AWS e Azure apresentaram instâncias com 2 vCPUs, a GCP utilizou uma instância com processamento compartilhado. Essa diferença é relevante para a interpretação dos resultados, pois pode influenciar a utilização de CPU e a ocorrência de picos durante a fase de carga. Portanto, a comparação de desempenho deve ser compreendida como uma análise entre recursos básicos disponíveis ou compatíveis com os planos gratuitos, e não como uma comparação entre máquinas virtualmente idênticas.

Também foram observadas diferenças no armazenamento e no sistema operacional. A AWS utilizou um volume gp3 de 8 GiB, a Azure empregou um disco Standard HDD LRS e a GCP utilizou um disco permanente padrão de 10 GB. Além disso, foram utilizadas versões distintas do Ubuntu, conforme disponibilidade e configuração do ambiente no momento da criação das instâncias. Para reduzir os efeitos dessas diferenças, manteve-se a mesma API backend, a mesma versão do Node.js, as mesmas dependências, as mesmas rotas experimentais e o mesmo roteiro geral de execução.

Tabela 1 – Caracterização das máquinas virtuais utilizadas no experimento

Característica	**AWS**	**Microsoft Azure**	**GCP**
Serviço	Amazon EC2	Azure Virtual Machines	Google Compute Engine
Instância	t3.micro	Standard B2ats v2	e2-micro
Processamento	2 vCPUs	2 vCPUs	Compartilhado
Memória	1 GiB	1 GiB	1 GiB
Disco	8 GiB, gp3	Standard HDD LRS	10 GB, padrão
Sistema operacional	Ubuntu 26.04 LTS	Ubuntu Server 24.04 LTS	Ubuntu 26.04 LTS Minimal

Região	us-east-1	North Central US	us-east1
Zona	us-east-1c	Não selecionada	us-east1-b

Fonte: Elaborado pelo autor (2026).

As instâncias foram implantadas em regiões dos Estados Unidos, porém em localizações distintas. A AWS foi configurada na região us-east-1, zona us-east-1c; a Azure, na região North Central US; e a GCP, na região us-east1, zona us-east1-b. No caso da Azure, não foi definida manualmente uma zona de disponibilidade específica durante o provisionamento, mantendo-se a configuração regional adotada no ambiente utilizado. Essa diferença geográfica foi considerada na interpretação do tempo de resposta, pois a métrica incluiu o percurso de rede entre o computador do pesquisador e cada provedor.

Dessa forma, as diferenças entre as máquinas virtuais não invalidam o experimento, mas delimitam o alcance da comparação. Para o público-alvo deste estudo, composto principalmente por estudantes, desenvolvedores e docentes interessados em atividades práticas com nuvem, essa condição também representa um cenário realista: ao utilizar planos gratuitos, o usuário normalmente precisa escolher entre os recursos de entrada efetivamente disponíveis em cada plataforma, mesmo que eles não sejam perfeitamente equivalentes.

3.4 MONITORAMENTO NATIVO

A coleta das métricas de infraestrutura foi realizada exclusivamente pelos serviços nativos das plataformas: Amazon CloudWatch, Azure Monitor e Google Cloud Monitoring. Não foram utilizados agentes externos ou ferramentas de terceiros, permitindo avaliar a observabilidade oferecida diretamente por cada provedor.

A análise considerou a disponibilidade de métricas de CPU, memória, rede e disco, a granularidade temporal dos dados e a necessidade de configurações adicionais. Também foram observadas a organização das informações e a facilidade de acesso aos indicadores nos respectivos painéis.

Na AWS, o Amazon CloudWatch disponibilizou métricas de CPU, rede e disco com granularidade de 5 minutos, mas não apresentou a métrica de memória sem configuração adicional. Na GCP, o Google Cloud Monitoring forneceu métricas de CPU, rede e disco com granularidade de 1 minuto, enquanto a memória também exigiria configuração complementar. Na Azure, o Azure Monitor apresentou CPU,

memória, rede e disco com granularidade de 1 minuto, sem instalação de agente adicional no ambiente utilizado.

A padronização das séries temporais em intervalos de 5 minutos seria matematicamente possível caso estivessem disponíveis os pontos numéricos completos de todas as plataformas. Entretanto, essa transformação não foi aplicada porque os dados exportados não possuíam o mesmo nível de detalhamento para AWS, Azure e GCP. Assim, reagregar apenas parte das séries, especialmente as de 1 minuto da Azure e da GCP, poderia gerar uma comparação estatisticamente incompleta.

Dessa forma, os indicadores foram analisados conforme a granularidade originalmente disponibilizada por cada serviço. Essa decisão foi considerada na interpretação dos resultados, principalmente nos valores máximos de CPU, pois picos de curta duração tendem a ser mais visíveis em séries de 1 minuto e podem ser suavizados em séries de 5 minutos.

Pelo mesmo motivo, não foram calculadas medidas de dispersão, como variância e desvio padrão, para todas as métricas. O cálculo dessas medidas exigiria séries brutas completas e comparáveis entre as três plataformas. Como essa condição não foi atendida de maneira uniforme, optou-se por utilizar medidas descritivas mais simples, como média, máximo, taxa de erro e volume de rede observado.

3.5 PROCESSO DE IMPLANTAÇÃO DA API

O gerenciamento do ciclo de vida da aplicação foi avaliado por meio da implantação inicial da versão 1.0, da atualização para a versão 1.1 e da reversão posterior para a versão original. Em todas as plataformas, foi utilizada a mesma aplicação backend e um roteiro equivalente de execução.

Foram registrados o tempo total de cada procedimento, o número de macroetapas da implantação inicial, as configurações manuais necessárias e a ocorrência de erros ou retrabalho. Esses indicadores foram utilizados para representar o esforço operacional observado em cada ambiente.

As operações foram executadas manualmente, com configurações mínimas para o funcionamento da aplicação e respeito às limitações dos planos gratuitos. A validação das versões implantadas foi realizada por meio das rotas experimentais da

API, confirmando a ativação da versão correspondente após a implantação, a atualização e a reversão.

3.6 PROCEDIMENTOS EXPERIMENTAIS

Os experimentos foram realizados com a API backend OpiniAuto, desenvolvida com Node.js, Fastify e TypeScript. O código-fonte da aplicação e as instruções gerais de execução estão disponíveis em repositório público[3].

A aplicação utilizada não se limitou ao fornecimento de páginas estáticas. Para o experimento, foram utilizadas rotas específicas de validação e geração de carga, permitindo verificar a disponibilidade da API, identificar a versão implantada e acionar processamento controlado no servidor. Dessa forma, a aplicação funcionou como instrumento para exercitar a infraestrutura das máquinas virtuais, e não como objeto principal de avaliação funcional.

Foram utilizadas as versões v1.0-experiment e v1.1-experiment. A versão 1.0 constituiu o estado inicial da aplicação e disponibilizava as rotas necessárias para geração de carga e verificação de funcionamento. A versão 1.1 introduziu a rota /experiment/info, utilizada para confirmar a atualização. Após a reversão, o retorno da aplicação à versão 1.0 foi validado pela rota /experiment/version e pela indisponibilidade de /experiment/info.

As rotas experimentais possuíam finalidades distintas. A rota /experiment/health foi utilizada nas fases de aquecimento e recuperação, permitindo verificar a disponibilidade da aplicação sem gerar carga computacional significativa. A rota /experiment/cpu foi utilizada durante a fase de carga controlada, acionando processamento no backend. A rota /experiment/version identificou a versão ativa da aplicação, enquanto /experiment/info confirmou a funcionalidade adicionada na versão 1.1.

Foram executadas três rodadas completas em cada plataforma. Cada rodada foi composta por 2 minutos de aquecimento, 5 minutos de carga controlada e 2 minutos de recuperação, conforme apresentado na Figura 2. Durante a fase de carga, as requisições foram enviadas à rota /experiment/cpu em intervalos regulares de 1 segundo, resultando em uma distribuição uniforme de requisições ao longo dos 5 minutos de teste.

---

[3]Disponível em: <https://github.com/Pedrovallentino/opiniauto-backend>. Acesso em: 26 jun. 2026.

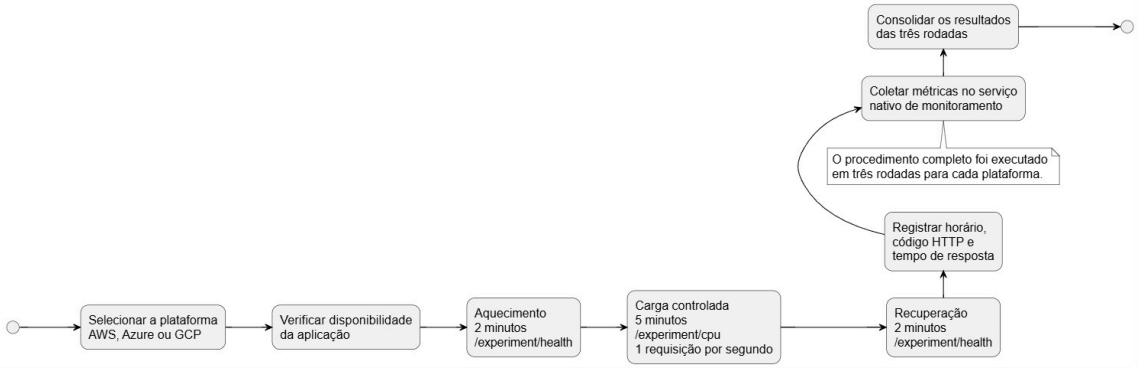
Durante os testes, foram registrados o horário, o código de status HTTP e o tempo de resposta de cada requisição. A taxa de erro foi calculada com base nas respostas diferentes de HTTP 200, enquanto o tempo médio foi obtido a partir das requisições realizadas nas fases de carga das três rodadas válidas.

Antes da definição do intervalo de 1 segundo, foi conduzido um teste preliminar na AWS com requisições contínuas. A ocorrência de respostas HTTP 429 indicou interferência do mecanismo de limitação de requisições da própria aplicação. Esse comportamento não foi desabilitado, pois a alteração poderia modificar a lógica da API utilizada no experimento. Por esse motivo, o teste preliminar foi descartado e as rodadas válidas foram padronizadas com uma requisição por segundo, evitando a interferência desse limite.

As instâncias permaneceram ativas por aproximadamente 24 horas antes da execução dos testes. Na AWS, foi necessário reiniciar manualmente o processo Node.js antes das rodadas, enquanto GCP e Azure permaneceram respondendo normalmente. Essa ocorrência foi registrada como parte da experiência operacional, mas não integrou as métricas de desempenho das rodadas.

Os procedimentos de implantação, atualização e reversão foram executados manualmente. Foram registrados os tempos, as macroetapas, as configurações necessárias, a ocorrência de erros ou retrabalho e a confirmação funcional das versões implantadas. O downtime não foi monitorado continuamente e, por isso, os tempos obtidos representam o esforço operacional dos procedimentos, e não a duração exata de indisponibilidade da aplicação.

Figura 2 – Fluxo das rodadas de testes experimentais



Fonte: Elaborado pelo autor (2026).

4 RESULTADOS E DISCUSSÃO

Nesta seção são apresentados e discutidos os resultados obtidos nos experimentos realizados nas plataformas Amazon Web Services, Microsoft Azure e Google Cloud Platform. A análise foi organizada de acordo com os três eixos definidos na metodologia: desempenho das máquinas virtuais, monitoramento nativo e gerenciamento do ciclo de vida da aplicação. Os resultados são interpretados de forma comparativa, considerando as condições específicas do experimento e as limitações de equivalência entre os ambientes utilizados.

4.1 DESEMPENHO DAS MÁQUINAS VIRTUAIS

A avaliação considerou a utilização média e máxima de CPU, o tempo médio de resposta, a taxa de erro e o tráfego total de rede observado durante as três rodadas válidas. Os resultados são apresentados na Tabela 2.

Tabela 2 – Métricas de desempenho e utilização de recursos

Plataforma	CPU média (%)	CPU máxima (%)	Resposta (ms)	Erro (%)	Rede (MB)
AWS	0,25	1,26	230,00	0	2,02
Azure	0,30	1,34	305,34	0	6,61
GCP	3,77	49,78	366,52	0	2,19

Fonte: Elaborado pelo autor (2026).

Todas as plataformas processaram a carga sem falhas HTTP, registrando taxa de erro igual a 0%. Neste trabalho, a expressão carga leve refere-se ao envio de uma requisição por segundo durante 5 minutos em cada rodada, totalizando aproximadamente 300 requisições por rodada e 900 requisições por plataforma. Esse nível de carga foi adotado para observar o comportamento inicial das máquinas virtuais de entrada sem acionar mecanismos de limitação da aplicação ou sobrecarga artificial do ambiente.

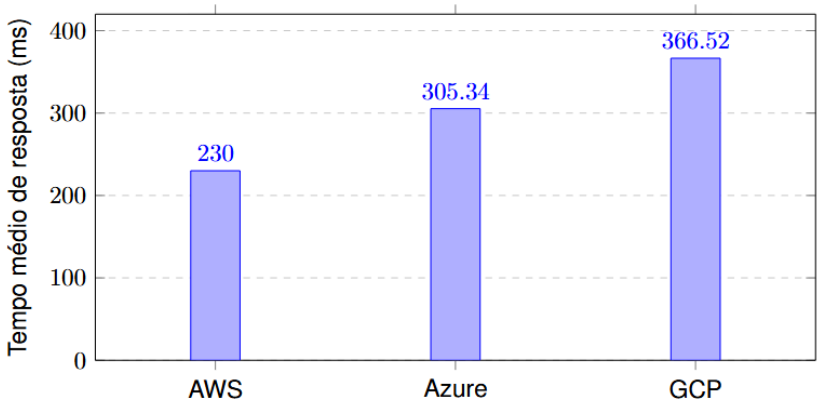
Os horários foram considerados no horário de Brasília. Na AWS, a janela de monitoramento analisada no Amazon CloudWatch foi de 11h23 a 13h03. Na Azure, as rodadas de teste ocorreram entre 16h54 e 17h34. Na GCP, as rodadas tiveram início às 08h32, com métricas de CPU analisadas no Google Cloud Monitoring entre 08h25 e 09h40. Esses registros foram utilizados para associar as requisições às métricas coletadas em cada plataforma, sem considerar o horário como variável independente de comparação.

A AWS apresentou o menor tempo médio de resposta, com 230,00 ms, seguida pela Azure, com 305,34 ms, e pela GCP, com 366,52 ms. No cenário

avaliado, a AWS apresentou tempo aproximadamente 24,7% inferior ao da Azure e 37,2% inferior ao da GCP. A Azure, por sua vez, apresentou tempo aproximadamente 16,7% inferior ao da GCP. A comparação visual é apresentada na Figura 3.

Os tempos registrados correspondem ao percurso completo das requisições, incluindo o processamento da aplicação e a comunicação de rede entre o computador do pesquisador e cada provedor. Portanto, os valores não representam exclusivamente o desempenho interno das máquinas virtuais.

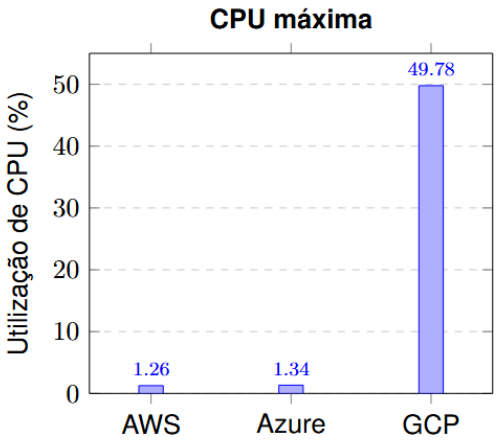
Figura 3 – Tempo médio de resposta das requisições por plataforma



Fonte: Elaborado pelo autor (2026).

Em relação à utilização de CPU, AWS e Azure apresentaram comportamento semelhante. A AWS registrou média de 0,25% e máxima de 1,26%, enquanto a Azure apresentou média de 0,30% e máxima de 1,34%. A GCP registrou média de 3,77% e máxima de 49,78%, demonstrando maior variação durante a fase de carga controlada, conforme apresentado na Figura 4.

Figura 4 – Utilização máxima de CPU por plataforma



Fonte: Elaborado pelo autor (2026).

A maior utilização de CPU observada na GCP deve ser interpretada considerando a característica da instância e2-micro, que utiliza processamento compartilhado. Essa diferença pode ter contribuído para a maior média e para o pico de CPU registrado nessa plataforma, especialmente quando comparada às instâncias da AWS e da Azure, que apresentaram 2 vCPUs no ambiente testado. Ainda assim, os dados não permitem atribuir o resultado exclusivamente a esse fator, pois também podem ter influenciado aspectos como região, compartilhamento de recursos, granularidade das métricas e condições momentâneas do ambiente.

A comparação dos valores máximos de CPU também deve considerar a diferença de granularidade entre os serviços de monitoramento. Enquanto a AWS apresentou pontos em intervalos de 5 minutos, Azure e GCP disponibilizaram dados com granularidade de 1 minuto. Dessa forma, picos de curta duração, como o registrado na GCP, possuem maior probabilidade de serem identificados nas séries mais detalhadas, enquanto podem ser suavizados em intervalos temporais mais amplos. O valor máximo deve, portanto, ser interpretado como um comportamento observado no ambiente e na resolução disponível, e não como uma medida diretamente equivalente entre os provedores.

No tráfego de rede, a Azure apresentou o maior volume total, com 6,61 MB, seguida pela GCP, com 2,19 MB, e pela AWS, com 2,02 MB. Esses valores devem ser interpretados de forma indicativa, pois AWS e Azure forneceram totais agregados de entrada e saída, enquanto o resultado da GCP foi estimado a partir de taxas médias exportadas pelo serviço de monitoramento. Assim, a comparação evidencia diferenças observadas no experimento, mas não representa equivalência absoluta entre os mecanismos de contabilização dos provedores.

De forma geral, a AWS apresentou o menor tempo médio de resposta e baixa utilização de CPU; a Azure obteve desempenho intermediário, com utilização de CPU próxima à AWS; e a GCP apresentou o maior tempo de resposta e o maior valor máximo de CPU registrado. Para o público-alvo deste estudo, esses resultados indicam vantagens contextuais: a AWS mostrou-se mais favorável quando a prioridade é menor tempo médio de resposta; a Azure apresentou comportamento equilibrado nesse eixo; e a GCP, embora tenha concluído as rodadas sem erros, exige maior cautela na interpretação da CPU devido ao processamento compartilhado da instância utilizada. Portanto, os resultados não indicam

superioridade geral de uma plataforma, mas ajudam a orientar escolhas conforme a prioridade do usuário e as limitações do plano gratuito.

Como não estavam disponíveis séries brutas uniformes para as três plataformas, não foram calculadas medidas de dispersão, como variância ou desvio padrão. Essa limitação reduz a profundidade estatística da comparação, mas preserva a consistência metodológica ao evitar cálculos baseados em dados incompletos ou não equivalentes. Esses resultados reforçam a importância de avaliações práticas realizadas sob condições controladas (KAUSHIK; SHARMA; GUPTA, 2021; KHANGHAHI; RAVANMEHR, 2013).

4.2 MONITORAMENTO NATIVO DAS PLATAFORMAS

A avaliação do monitoramento nativo considerou a disponibilidade de métricas de CPU, memória, rede e disco, a granularidade temporal dos dados e a necessidade de configurações adicionais. Os resultados consolidados são apresentados na Tabela 3.

Tabela 3 – Avaliação do monitoramento nativo das plataformas

Critério	AWS	Azure	GCP
Métrica de CPU disponível	Sim	Sim	Sim
Métrica de memória disponível	Não	Sim	Não
Métricas de rede disponíveis	Sim	Sim	Sim
Métricas de disco disponíveis	Sim	Sim	Sim
Granularidade temporal	5 minutos	1 minuto	1 minuto
Necessidade de configuração adicional	Sim, para memória	Não	Sim, para memória

Fonte: Elaborado pelo autor (2026).

As três plataformas disponibilizaram nativamente métricas de CPU, rede e disco. A principal diferença esteve relacionada à memória. Na AWS, o Amazon CloudWatch não apresentou esse indicador no painel padrão da instância utilizada, sendo necessária configuração adicional. Situação semelhante ocorreu na GCP, cuja métrica de memória também exigiria instalação ou ativação complementar.

Na Azure, as métricas de memória foram disponibilizadas diretamente no namespace Virtual Machine Host, sem necessidade de ativação do VM Insights, instalação manual de agente ou configuração adicional. Dessa forma, no ambiente avaliado, a Azure apresentou o conjunto mais abrangente de métricas nativas.

Também foram observadas diferenças na granularidade temporal. O Amazon CloudWatch apresentou dados em intervalos de 5 minutos, enquanto Azure Monitor

e Google Cloud Monitoring disponibilizaram pontos com granularidade de 1 minuto. Intervalos menores favorecem a identificação de variações de curta duração, enquanto agregações mais amplas podem suavizar oscilações ocorridas durante as fases dos testes.

Em relação à rede, as três plataformas forneceram dados de entrada e saída, mas adotaram formas distintas de apresentação. AWS e Azure disponibilizaram valores totais agregados, enquanto a GCP apresentou taxas médias exportadas, exigindo estimativa do volume total no período analisado. Assim, embora os indicadores fossem funcionalmente equivalentes, o tratamento dos dados não foi idêntico entre os provedores.

As métricas de disco também estavam disponíveis nas três plataformas, ainda que com diferenças de nomenclatura e organização. Essas variações demonstram que a observabilidade não depende apenas da existência de um serviço de monitoramento, mas também da abrangência das métricas, da granularidade temporal e do esforço necessário para obtê-las.

De forma geral, a Azure apresentou maior abrangência nativa no ambiente testado, ao disponibilizar CPU, memória, rede e disco com granularidade de 1 minuto. A GCP também ofereceu granularidade de 1 minuto, mas exigiria configuração adicional para memória. A AWS disponibilizou os principais indicadores, porém com granularidade de 5 minutos e sem memória no painel padrão. Esses resultados reforçam o caráter multidimensional da avaliação de plataformas de nuvem discutido por (KHANGHAHI; RAVANMEHR, 2013) e (CASALE, 2023).

4.3 IMPLANTAÇÃO INICIAL DA APLICAÇÃO

A análise da implantação inicial considerou o tempo necessário para disponibilizar a versão 1.0 da aplicação, o número de macroetapas, as configurações manuais exigidas e a ocorrência de erros ou retrabalho. Os resultados são apresentados na Tabela 4.

Os procedimentos seguiram um roteiro equivalente nas três plataformas, incluindo criação da máquina virtual, acesso por SSH, atualização do sistema, instalação das ferramentas necessárias, configuração do ambiente, clonagem do repositório, seleção da versão experimental, instalação das dependências,

compilação e inicialização da aplicação. Para fins comparativos, essas atividades foram organizadas em nove macroetapas. Como as operações foram executadas manualmente, os tempos registrados podem ter sido influenciados pela familiaridade do pesquisador com cada ambiente, pela velocidade de execução dos comandos e pela navegação nos painéis das plataformas.

Tabela 4 – Processo de implantação inicial da aplicação

Plataforma	Tempo (min)	Etapas	Configurações manuais	Erros ou retrabalho
AWS	20	9	Grupo de segurança, chave SSH, variáveis de ambiente e liberação da porta da aplicação	Sim
Azure	20	9	Grupo de recursos, Network Security Group, chave SSH, regra para a porta 3000 e variáveis de ambiente	Não
GCP	18	9	Regra de firewall, tag de rede, variáveis de ambiente e liberação da porta da aplicação	Não

Fonte: Elaborado pelo autor (2026).

A GCP apresentou o menor tempo de implantação, com 18 minutos, enquanto AWS e Azure demandaram 20 minutos. A diferença de 2 minutos correspondeu a uma redução de 10% em relação às demais plataformas, indicando vantagem temporal pequena no cenário avaliado.

Apesar dos tempos próximos, foram observadas diferenças na configuração da infraestrutura. Na AWS, o processo envolveu grupo de segurança, chave SSH, variáveis de ambiente e liberação da porta da aplicação. Na Azure, foram utilizados grupo de recursos, Network Security Group, chave SSH, regra para a porta 3000 e variáveis de ambiente. Na GCP, foram configuradas regra de firewall, tag de rede, variáveis de ambiente e porta de acesso.

A Azure apresentou maior quantidade de componentes de infraestrutura explicitamente organizados no portal, como grupo de recursos, rede virtual, interface de rede, endereço IP público e Network Security Group. Entretanto, parte desses recursos foi criada ou associada pelo próprio assistente de provisionamento, sem aumentar o tempo total em relação à AWS.

A AWS foi a única plataforma em que ocorreu retrabalho. A aplicação foi inicialmente executada antes da geração do arquivo compilado, resultando em erro pela ausência de dist/server.js. A falha foi corrigida com a execução do comando de compilação e a reinicialização da aplicação. Como o problema decorreu de uma inversão manual das etapas, não foi atribuído a uma limitação da plataforma.

Na GCP e na Azure, a implantação foi concluída sem repetição de etapas. Em todos os ambientes, a conclusão foi validada pela rota /experiment/version, que retornou a versão 1.0.0.

De forma geral, as três plataformas apresentaram processos de implantação tecnicamente viáveis e tempos próximos. A GCP obteve o menor tempo absoluto, enquanto AWS e Azure apresentaram resultados equivalentes. Para estudantes, desenvolvedores e docentes que utilizem ambientes gratuitos em atividades práticas, esses resultados indicam que o esforço inicial de implantação foi semelhante entre os provedores, embora cada plataforma organize recursos de rede, acesso e segurança de forma própria. Assim, os tempos devem ser interpretados como medida do esforço operacional observado no experimento, e não como uma métrica definitiva de facilidade de uso das plataformas (SARASWAT; TRIPATHI, 2020; HYSENI; IBRAHIMI, 2017).

4.4 ATUALIZAÇÃO E REVERSÃO DA APLICAÇÃO

A análise do ciclo de vida da aplicação considerou o tempo necessário para atualizar a versão 1.0 para a versão 1.1 e realizar a reversão posterior, além da ocorrência de erros, retrabalho e informações sobre indisponibilidade. A Tabela 5 apresenta os resultados.

Tabela 5 – Processos de atualização e reversão da aplicação

Plataforma	Tempo de atualização (min)	Tempo de reversão (min)	Downtime	Erros ou retrabalho
AWS	11	5	Não mensurado	Não
Azure	6	3	Não mensurado	Não
GCP	6	4	Não mensurado	Não

Fonte: Elaborado pelo autor (2026).

Na atualização, Azure e GCP apresentaram o menor tempo, com 6 minutos, enquanto a AWS demandou 11 minutos. Assim, o tempo registrado na AWS foi aproximadamente 83,3% superior ao observado nas demais plataformas. Nenhum dos procedimentos apresentou erros ou necessidade de retrabalho.

A atualização envolveu a interrupção da aplicação, a mudança do repositório para a versão 1.1, a verificação das dependências, a geração dos artefatos, a compilação e a reinicialização do processo Node.js. A conclusão foi validada pelas rotas /experiment/version e /experiment/info, que confirmaram a ativação da nova versão e da funcionalidade adicionada.

Na reversão, a Azure apresentou o menor tempo, com 3 minutos, seguida pela GCP, com 4 minutos, e pela AWS, com 5 minutos. Em relação à AWS, o procedimento foi 40% mais rápido na Azure e 20% mais rápido na GCP. A operação consistiu no retorno do repositório à versão 1.0, na recompilação e na reinicialização da aplicação. A validação confirmou novamente a versão 1.0.0 e a indisponibilidade da rota /experiment/info.

Os tempos registrados correspondem à duração total dos procedimentos manuais e podem ter sido influenciados pela execução dos comandos, pela resposta da conexão SSH e pelo tempo de compilação. Dessa forma, os resultados representam o esforço operacional observado, e não uma medida definitiva da eficiência técnica dos provedores.

O downtime não foi monitorado continuamente. Portanto, não foi possível determinar o período exato em que a aplicação permaneceu indisponível durante as transições. O tempo total de execução não deve ser interpretado como equivalente ao tempo de indisponibilidade percebido pelo usuário.

De forma geral, Azure e GCP apresentaram maior agilidade na atualização, enquanto a Azure também obteve o menor tempo de reversão. A AWS concluiu corretamente os dois procedimentos, porém apresentou os maiores tempos. Para o público-alvo deste estudo, esses resultados indicam que, em atividades práticas com ambientes gratuitos, a atualização e a reversão podem variar não apenas pela aplicação utilizada, mas também pela resposta da conexão SSH, pelo tempo de compilação, pela organização dos comandos e pela experiência do usuário com cada plataforma. Assim, os tempos registrados devem ser interpretados como esforço operacional observado no experimento, e não como medida exata de indisponibilidade ou como prova definitiva de superioridade técnica de um provedor. Ainda assim, os resultados oferecem uma referência prática para estudantes, desenvolvedores e docentes que desejem comparar o ciclo básico de atualização e reversão de uma API backend em máquinas virtuais de entrada.

4.5 SÍNTESE COMPARATIVA DOS RESULTADOS

A análise conjunta demonstrou que AWS, Azure e GCP foram capazes de hospedar a aplicação, processar a carga sem erros e permitir os procedimentos de

implantação, atualização e reversão. Entretanto, cada plataforma apresentou vantagens em dimensões distintas.

No desempenho das requisições, a AWS obteve o menor tempo médio de resposta, com 230,00 ms, seguida pela Azure, com 305,34 ms, e pela GCP, com 366,52 ms. AWS e Azure também apresentaram baixa utilização de CPU, enquanto a GCP registrou valores superiores, principalmente no pico máximo. Esses resultados devem ser limitados ao cenário experimental, pois foram influenciados pelas características das instâncias, pelo compartilhamento dos recursos e pelo percurso de rede entre o gerador de carga e os provedores.

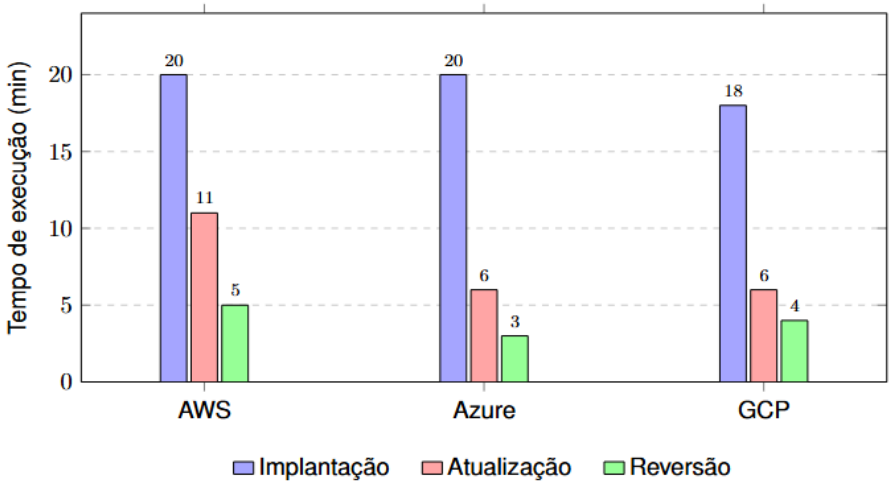
Quanto ao monitoramento nativo, a Azure disponibilizou o conjunto mais abrangente de métricas, incluindo memória e granularidade de 1 minuto, sem configuração adicional. A GCP também apresentou granularidade de 1 minuto, mas exigiria configuração complementar para memória. A AWS forneceu CPU, rede e disco, porém com granularidade de 5 minutos e sem memória no painel padrão.

Na implantação inicial, os tempos foram próximos: 18 minutos na GCP e 20 minutos na AWS e na Azure. As três plataformas demandaram nove macroetapas, embora apresentassem diferenças na organização e configuração dos recursos. O retrabalho registrado na AWS decorreu de uma inversão manual entre compilação e inicialização, não sendo atribuído ao provedor.

Azure e GCP apresentaram o menor tempo de atualização, com 6 minutos, enquanto a AWS demandou 11 minutos. Na reversão, a Azure registrou 3 minutos, a GCP 4 minutos e a AWS 5 minutos. Como o downtime não foi monitorado continuamente, esses valores representam o esforço operacional dos procedimentos, e não a duração exata de indisponibilidade.

A Figura 5 apresenta os tempos registrados nos processos de implantação, atualização e reversão da aplicação em cada plataforma. As três etapas foram representadas por cores distintas.

Figura 5 – Tempos dos processos de implantação, atualização e reversão por plataforma



Fonte: Elaborado pelo autor (2026).

Os resultados indicam que o processo de implantação apresentou duração semelhante nas três plataformas, com 20 minutos na AWS e na Azure e 18 minutos na GCP. A diferença mais relevante foi observada nas operações posteriores à implantação. Na AWS, a atualização exigiu 11 minutos, enquanto Azure e GCP concluíram o mesmo procedimento em 6 minutos. Na reversão, a Azure apresentou o menor tempo, com 3 minutos, seguida pela GCP, com 4 minutos, e pela AWS, com 5 minutos. Esses resultados sugerem que, embora o provisionamento inicial tenha demandado esforço próximo entre os provedores, os procedimentos de atualização e reversão foram mais ágeis na Azure e na GCP no ambiente experimental adotado. Essa interpretação, contudo, está relacionada às configurações e aos procedimentos específicos empregados no estudo, não devendo ser generalizada para todos os cenários de implantação.

A síntese dos resultados permite identificar vantagens contextuais entre as plataformas avaliadas. No cenário experimental deste trabalho, a AWS mostrou-se mais favorável quando a prioridade é o menor tempo médio de resposta. A Azure destacou-se quando a prioridade é a abrangência do monitoramento nativo e a agilidade na reversão. A GCP apresentou menor tempo de implantação inicial e tempo de atualização equivalente ao da Azure, embora a interpretação da utilização de CPU exija cautela devido ao processamento compartilhado da instância utilizada. Para estudantes, desenvolvedores e docentes interessados em atividades práticas com nuvem, esses resultados oferecem uma referência inicial para escolha da plataforma conforme o objetivo da atividade ou do projeto. Ainda assim, a decisão não deve ser generalizada para todos os cenários, pois os resultados estão

condicionados às instâncias, regiões, configurações, carga aplicada e limitações dos planos gratuitos utilizados no experimento (KAUSHIK; SHARMA; GUPTA, 2021; SARASWAT; TRIPATHI, 2020; HYSENI; IBRAHIMI, 2017).

5 CONSIDERAÇÕES FINAIS

Este trabalho comparou serviços básicos de Infraestrutura como Serviço (IaaS) disponibilizados nos planos gratuitos da AWS, do Azure e da GCP, considerando três dimensões: desempenho de máquinas virtuais de entrada, monitoramento nativo e esforço operacional associado à implantação, atualização e reversão de uma API backend. A utilização da mesma aplicação, das mesmas rotas experimentais e de procedimentos padronizados permitiu identificar diferenças técnicas e operacionais entre os provedores no cenário avaliado.

As três plataformas processaram a carga controlada sem erros e permitiram concluir os procedimentos relacionados ao ciclo de vida da aplicação. Entretanto, nenhuma apresentou os melhores resultados em todos os critérios. No cenário experimental avaliado, a AWS obteve o menor tempo médio de resposta; a Azure destacou-se pela maior abrangência das métricas nativas e pelo menor tempo de reversão; e a GCP apresentou o menor tempo de implantação inicial, além de tempo de atualização equivalente ao da Azure. Esses resultados indicam vantagens contextuais, úteis para orientar escolhas conforme a prioridade do usuário, sem caracterizar superioridade geral de uma plataforma sobre as demais.

Os resultados devem ser interpretados considerando as limitações do delineamento experimental. Não foi possível estabelecer equivalência absoluta entre as máquinas virtuais devido às diferenças de arquitetura, processamento, armazenamento, sistemas operacionais, disponibilidade regional e restrições dos planos gratuitos. AWS e Azure utilizaram instâncias com 2 vCPUs, enquanto a instância e2-micro da GCP adotava processamento compartilhado. Embora a aplicação, as dependências e os procedimentos tenham sido padronizados, essas diferenças podem ter influenciado o comportamento dos ambientes.

Também foi utilizada uma única API backend, submetida a uma carga leve, definida como uma requisição por segundo durante 5 minutos em cada rodada. O tempo de resposta incluiu o percurso de rede entre o computador do pesquisador e cada provedor, não representando exclusivamente o processamento interno das

máquinas virtuais. Além disso, os dados de rede foram apresentados e agregados de formas distintas pelas plataformas, razão pela qual foram tratados como indicadores comparativos, e não como medidas absolutamente equivalentes.

A granularidade das métricas de monitoramento também não foi uniforme. A AWS disponibilizou dados em intervalos de 5 minutos, enquanto Azure e GCP apresentaram métricas com granularidade de 1 minuto. Como não estavam disponíveis séries numéricas completas das três plataformas, não foi possível reagregar os dados em uma mesma resolução temporal nem calcular uniformemente medidas de dispersão, como variância e desvio padrão. Essa condição limita principalmente a comparação dos valores máximos de CPU e de oscilações de curta duração.

A memória não pôde ser comparada quantitativamente em todos os ambientes sem configurações adicionais, e o downtime não foi monitorado continuamente durante a atualização e a reversão. Os tempos dos procedimentos operacionais também podem ter sido influenciados pela execução manual dos comandos, pela conexão SSH, pela instalação de dependências e pela duração das compilações. Assim, esses valores representam a experiência operacional observada no cenário estudado, e não medidas definitivas da capacidade técnica dos provedores.

Como trabalhos futuros, recomenda-se ampliar o número de rodadas e repetir os experimentos em diferentes períodos, regiões e zonas de disponibilidade. A utilização de máquinas virtuais com configurações mais equivalentes, incluindo processamento, memória, armazenamento e sistema operacional, permitiria reduzir parte das diferenças existentes entre os ambientes avaliados.

Também se recomenda preservar e exportar as séries temporais completas de todas as métricas e requisições. Esses registros possibilitariam padronizar a resolução temporal dos dados, calcular medidas de dispersão e realizar análises estatísticas mais abrangentes. A instalação dos agentes oficiais de monitoramento também permitiria comparar o consumo de memória e outras métricas não disponíveis nativamente em todos os ambientes.

Experimentos posteriores podem utilizar cargas com maior concorrência, banco de dados, operações intensivas de processamento e entrada e saída, além de mecanismos externos de verificação contínua para mensurar o downtime durante atualizações e reversões. A inclusão de custos, segurança, contêineres, automação

de infraestrutura e escalabilidade também permitiria ampliar a comparação entre desempenho, observabilidade, esforço operacional e impacto financeiro. Essas ampliações podem tornar o estudo ainda mais útil para desenvolvedores, estudantes e docentes interessados em utilizar ambientes de nuvem em projetos práticos ou atividades de ensino.

REFERÊNCIAS

AMAZON WEB SERVICES. **What is Amazon CloudWatch?** 2026. Disponível em: <https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html>. Acesso em: 26 jun. 2026.

AMAZON WEB SERVICES. **What is Amazon EC2?** 2026. Disponível em: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>. Acesso em: 26 jun. 2026.

BALDINI, I. et al. Serverless computing: Current trends and open problems. In: **Research Advances in Cloud Computing**. [S.l.]: Springer, 2017. p. 1–20.

CASALE, G. Performance evaluation teaching in the age of cloud computing. **Performance Evaluation Review**, v. 51, n. 2, p. 45–52, 2023.

ERL, T.; PUTTINI, R.; MAHMOOD, Z. **Cloud Computing**: Concepts, Technology & Architecture. Upper Saddle River: Prentice Hall, 2013.

GOOGLE CLOUD. **Cloud Monitoring overview.** 2026. Disponível em: <https://cloud.google.com/monitoring/docs/monitoring-overview>. Acesso em: 26 jun. 2026.

GOOGLE CLOUD. **Compute Engine overview.** 2026. Disponível em: <https://cloud.google.com/compute/docs/overview>. Acesso em: 26 jun. 2026.

HYSENI, F.; IBRAHIMI, L. Comparison of the cloud computing platforms provided by amazon and google. In: **International Conference on Business Informatics Research**. [S.l.]: Springer, 2017. p. 190–203.

KAUSHIK, S.; SHARMA, M.; GUPTA, R. Cloud computing and comparison based on service and performance between amazon aws, microsoft azure and google cloud. **International Journal of Computer Applications**, v. 174, n. 16, p. 1–7, 2021.

KHANGHAHI, N.; RAVANMEHR, R. Cloud computing performance evaluation: Issues and challenges. In: **International Conference on Cloud Computing and Big Data**. [S.l.]: IEEE, 2013. p. 423–428.

MELL, P.; GRANCE, T. **The NIST Definition of Cloud Computing.** Gaithersburg, 2011.

MICROSOFT. **Azure Monitor overview.** 2026. Disponível em: <https://learn.microsoft.com/en-us/azure/azure-monitor/fundamentals/overview>. Acesso em: 26 jun. 2026.

MICROSOFT. **Virtual machines in Azure.** 2026. Disponível em: <https://learn.microsoft.com/en-us/azure/virtual-machines/overview>. Acesso em: 26 jun. 2026.

SARASWAT, M.; TRIPATHI, A. Cloud computing: Comparison and analysis of cloud service providers—aws, microsoft and google. **International Journal of Engineering Research and Technology**, v. 9, n. 6, p. 227–232, 2020.

SENEL, B. C. et al. Multitenant containers as a service for clouds and edge clouds. **arXiv preprint** arXiv:2304.08927, 2023.