UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA

CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

ISAIAS DO CARMO CARNEIRO

KUBESILENT: PROTÓTIPO DE UMA SOLUÇÃO WEB PARA MONITORAMENTO E ANÁLISE DE ATIVIDADES DE CRYPTOJACKING EM AMBIENTES AKS

SOBRAL

2024

ISAIAS DO CARMO CARNEIRO

KUBESILENT: PROTÓTIPO DE UMA SOLUÇÃO WEB PARA MONITORAMENTO E ANÁLISE DE ATIVIDADES DE CRYPTOJACKING EM AMBIENTES AKS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Orientador: Prof. Me. Erick Aguiar Donato

SOBRAL

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C288k Carneiro, Isaias do Carmo.
Kubesilent: Protótipo de uma solução web para monitoramento e análise de atividades de cryptojacking em ambientes AKS / Isaias do Carmo Carneiro. – 2024.
55 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2024.
Orientação: Prof. Me. Erick Aguiar Donato.

1. Computação em nuvem. 2. Cryptojacking. 3. Azure Kubernetes Service. 4. Segurança. 5. Monitoramento. I. Título.

CDD 621.39

ISAIAS DO CARMO CARNEIRO

KUBESILENT: PROTÓTIPO DE UMA SOLUÇÃO WEB PARA MONITORAMENTO E ANÁLISE DE ATIVIDADES DE CRYPTOJACKING EM AMBIENTES AKS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Aprovada em: 19 de Setembro de 2024

BANCA EXAMINADORA

Prof. Me. Erick Aguiar Donato (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Me. David Nascimento Coelho
Universidade Federal do Ceará (UFC)

Prof. Dr. Wendley Souza da Silva
Universidade Federal do Ceará (UFC)

À minha família, por sua capacidade de acreditar e investir em mim em momentos de dificuldades. Aos meus pais, pela educação que me deram ao longo da vida e pelos sacrifícios que fizeram para que eu pudesse frequentar a faculdade. A minha esposa, por sempre estar ao meu lado quando preciso.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão à minha mãe, Eronilde do Carmo Carneiro, que sempre me apoiou e me deu forças para seguir um caminho honesto. Ela foi o alicerce fundamental para que eu chegasse onde estou hoje. Graças a ela, pude aprender um instrumento, frequentar uma boa escola e, finalmente, cursar a faculdade.

Ao Prof. Erick Aguiar Donato, agradeço pela orientação e paciência nos momentos em que mais precisei. Sua presença constante e disposição para me guiar foram essenciais.

Aos meus familiares, agradeço por me incentivarem nos momentos difíceis e por compreenderem minha ausência enquanto me dedicava à faculdade.

À minha esposa, Taís, sou imensamente grato por seu apoio e motivação nos períodos mais desafiadores do curso, especialmente em tempos difíceis.

Aos professores, agradeço por todos os conselhos, pela ajuda e pela paciência com que guiaram meu aprendizado.

Por fim, agradeço à empresa Lanlink pela oportunidade de estágio, que me proporcionou um aprendizado valioso sobre o tema que envolve o presente trabalho

RESUMO

O advento da computação em nuvem trouxe inúmeras vantagens, mas também novos desafios de segurança, como o *cryptojacking*, que é o uso não autorizado de recursos computacionais de uma organização para minerar criptomoedas. O *Azure Kubernetes Service* (AKS), uma solução popular para a orquestração de contêineres em nuvem da Microsoft, é especialmente alvo desses ataques devido à sua complexidade e natureza escalável. Este Trabalho propõe o desenvolvimento de uma solução de monitoramento específica para *Kubernetes* no *Azure*, que visa identificar e alertar sobre atividades anômalas que possam indicar tentativas de *cryptojacking*. A pesquisa envolve uma revisão da literatura sobre computação em nuvem, *Azure*, *Kubernetes*, *Azure Kubernetes Service* e *cryptojacking*, além da discussão de trabalhos relacionados ao tema. Com base nisso, foi proposta e implementada uma solução de monitoramento específica para o *Kubernetes* na nuvem da *Microsoft*, com o intuito de detectar atividades suspeitas indicativas de *cryptojacking*. A solução se mostrou eficaz ao monitorar o ambiente, identificando anomalias que sugeriam tais atividades. Ademais, foi possível validar, na plataforma, a presença de processos maliciosos relacionados ao *cryptojacking*, comprovando a eficácia da abordagem adotada.

Palavras-chave: Computação em nuvem, *Cryptojacking*, *Azure Kubernetes Service*, Segurança, Monitoramento, Web.

ABSTRACT

The advent of cloud computing has brought numerous advantages but also new security challenges, such as cryptojacking—the unauthorized use of an organization's computational resources to mine cryptocurrencies. Microsoft's Azure Kubernetes Service (AKS), a popular solution for cloud container orchestration, is particularly susceptible to these attacks due to its complexity and scalable nature. This study proposes the development of a specific monitoring solution for Kubernetes on Azure, aimed at identifying and alerting about anomalous activities that may indicate cryptojacking attempts. The research includes a literature review on cloud computing, Azure, Kubernetes, Azure Kubernetes Service, and cryptojacking, as well as a discussion of related work on the topic. Based on this, a specific monitoring solution for Kubernetes on Microsoft's cloud was proposed and implemented, with the goal of detecting suspicious activities indicative of cryptojacking. The solution proved effective in monitoring the environment, identifying anomalies that suggested such activities. Additionally, it was possible to validate on the platform the presence of malicious processes related to cryptojacking, demonstrating the effectiveness of the adopted approach.

Keywords: Cloud Computing, Cryptojacking, Azure Kubernetes Service, Security, Monitoring, Web

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AD3	*Alternating Directions Dual Composition*
AKS	*Azure Kubernetes Service*
AWS	*Amazon Web Services*
CPU	*central processing unit*
HDD	*Hard Disk Drive*
NIST	*National Institute of Standards and Technology*
SSD	*Solid State Drive*
VCPU	*Virtual Central Processing Unit*
VM	*Virtual Machines*

SUMÁRIO

1 INTRODUÇÃO

Com o advento da computação em nuvem, tornou-se possível o acesso a poderosos recursos computacionais, facilitando a inovação e a transformação digital (VERAS MANOEL, p. 16). No entanto, junto com esses benefícios, surgem novos desafios de segurança. À medida que as organizações aceleram sua migração para ambientes de nuvem, a superfície de ataque para potenciais ameaças cibernéticas cresce proporcionalmente. Uma dessas ameaças emergentes é o *cryptojacking*, que, em ambientes de *cloud*, abusa de recursos ao comprometer ambientes empresariais para minerar criptomoedas (MICROSOFT, 2023).

O objetivo do *cryptojacking* é gerar lucro com pouco esforço, transferindo os custos operacionais, como energia e infraestrutura, para as vítimas. Em um ambiente de nuvem, essas atividades maliciosas podem passar despercebidas e, dado o poder de processamento que a nuvem entrega, essa ameaça pode elevar o consumo de recursos e, consequentemente, perda financeira para as empresas (ARIFFIN *et al.*, 2022).

Dentro do ecossistema de nuvem, o *Azure Kubernetes Service* (AKS) tem se destacado como uma solução popular para a orquestração de *containers* , ele permite que as empresas implementem, escalem e gerenciem suas aplicações de maneira eficiente. Com isso, a complexidade e a natureza dinâmica dos ambientes *Kubernetes* podem facilitar a exploração de vulnerabilidades, tornando-os alvos atrativos para ataques de *cryptojacking*. Ao usar conceitos de nuvem como escalabilidade, esse serviço é alvo de investidas de *cryptojacking*, afim de usar da capacidade de escalar recursos computacionais para minerar criptomoedas.

Este Trabalho propõe-se a abordar a problemática do *cryptojacking* em ambientes *kubernetes*, com foco no serviço de *Kubernetes* gerenciados da *Microsoft* AKS, explorando métodos e técnicas para a detecção dessa ameaça. A pesquisa visa desenvolver o protótipo de uma solução de monitoramento que possa identificar esse ataque no *Azure Kubernetes Service*, proporcionando visibilidade sobre atividades suspeitas de mineração de criptomoedas com base em indicadores conhecidos de *cryptojacking*.

A estrutura deste trabalho abrange uma revisão da literatura sobre computação em nuvem, *Kubernetes* e *cryptojacking*. Além disso, inclui uma análise sobre trabalhos relacionados ao tema, bem como o desenvolvimento e teste de uma solução prática. Esta pesquisa visa oferecer uma contribuição significativa ao campo de segurança em nuvem, apresentando uma solução *open-source* inovadora e eficiente para monitorar ambientes AKS contra atividades de *cryptojacking*.

1.1 Justificativa

Computação em nuvem foi um avanço significativo para a área de tecnologia. As empresas podem utilizar recursos de computação sem a necessidade de comprar e gerenciar servidores, já que agora tudo pode ser feito sob o modelo de consumo sob demanda, pagando apenas pelo processamento computacional que utilizam. Em meio a esses benefícios, surgem novos desafios quanto a segurança em ambientes de nuvem. Um desses desafios é o *cryptojacking* uma atividade que era comum em navegadores e *desktop* de usuários, porém vem tendo como foco plataformas de computação nos últimos anos (TEKINER *et al.*, 2021). Ataques de *cryptojacking* abusam de recursos de computação em nuvem e isso pode resultar em perdas financeiras para as organizações devido às altas taxas de computação que podem ser usadas com essa atividade. Nos ataques observados pela *Microsoft*, as organizações vitimas tiverem um prejuízo de mais de US$ 300.000 (MICROSOFT, 2023).

Diante dessa problemática crescente, o presente trabalho propõe o desenvolvimento de uma solução de monitoramento para detectar atividades de *cryptojacking* em ambientes AKS. O foco será no serviço de *kubernetes* da *Microsoft*, tanto por se tratar de uma plataforma amplamente utilizada em ambientes de nuvem quanto pelo fato de o *kubernetes* ser um alvo cada vez mais frequente de ataques de *cryptojacking*, conforme indicado em relatórios recentes (GRAP, 2023).

Nesse contexto, o desenvolvimento de uma solução de monitoramento específica para atividades de *cryptojacking* em *Kubernetes* é justificado pela necessidade de garantir a observabilidade e segurança dos ambientes de contêineres. Embora existam ferramentas semelhantes no mercado, como soluções disponíveis para AKS (*Azure Kubernetes Service*), muitas dessas exigem configurações complexas ou aquisição de licenças para funcionalidade completa. O diferencial da solução proposta é que ela é totalmente *open-source* e oferece uma implementação pronta para ser facilmente implantada no ambiente, permitindo a detecção rápida de comportamentos anômalos, como o uso excessivo de *central processing unit* (CPU) e a execução de processos maliciosos. Além disso, ajuda as organizações a manter conformidade com políticas de segurança, minimizando os impactos operacionais sem a necessidade de múltiplas ferramentas adicionais.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo deste trabalho é propor uma solução web que monitore e detecte atividades de *cryptojacking* no serviço de *kubernetes* da nuvem da *Microsoft* (*Azure Kubernetes Services*).

1.2.2 Objetivos específicos

- Conceituar Computação em nuvem, *Azure*, AKS e *Cryptojacking*;
- Compreender o entendimento do fluxo do ataque de *Cryptojacking* e a motivação dos agentes para realizar tal atividade;
- Definir os requisitos da solução;
- Codificar a solução;
- Criar ambiente de teste para simular atividade de *cryptojacking*;
- Testar e validar o comportamento da aplicação.

2 TRABALHOS RELACIONADOS

Nesta seção, são apresentados e discutidos os principais trabalhos relacionados que fundamentaram a escolha da solução proposta. Esses trabalhos serviram como referência, fornecendo orientação para a definição da abordagem adotada neste projeto.

(PASSOS *et al.*, 2024) trás um estudo focado na detecção de *cryptojacking*, baseado em algumas ferramentas de mercado. Além de conceituar *cryptojacking*, é feito um trabalho prático de comparação de eficacias dessas ferramentas junto ao *cryptojacking*. Apesar do estudo ser centrado em atividades de mineração de criptomoedas em navegadores, é apresentado brevemente uma ferramenta que usa o consumo de CPU do usuário como métrica de identificação de *cryptojacking*.

Com um tema mais próximo desse trabalho, (ARIFFIN *et al.*, 2022) apresenta um método para detectar atividades ilícitas de mineração de criptomoedas em plataformas de computação em nuvem. O trabalho usa métricas como CPU, memória e consumo de disco. O trabalho trás o uso do algoritmo *Alternating Directions Dual Composition* (AD3) para detectar os padrões maliciosos de consumo de recursos do sistema. Através de testes de simulação, o trabalho relaciona atividades de pico de CPU na nuvem quando o software de mineração está em uso, provando a eficácia do algoritmo.

Outro trabalho relevante sobre *cryptojacking* é apresentado por (BASAVARAJAPPA, 2016). Este trabalho trás uma análise abrangente da detecção de *cryptojacking* com base no uso de CPU por processos anômalos. É utilizado o *Xmrig*, uma aplicação de mineração de criptomoedas, para simular o ataque. O objetivo principal é identificar e avaliar os recursos do *Kubernetes* afetados por *malware* de mineração. Neste estudo a detecção é realizada por meio de ferramentas de código aberto, comparando o processo de mineração com um software legítimo de alto consumo de CPU para destacar diferenças de comportamento.

(JAYASINGHE; PORAVI, 2020) analisa ataques de *cryptojacking* contra infraestrutura de nuvem. Ele começa com uma breve visão geral do *cryptojacking* e seu impacto em ambientes de nuvem. Esse trabalho trás uma análise detalhada de alguns ataques distintos de *cryptojacking*. O trabalho categoriza alguns ataques de acordo com suas características, ferramentas usadas, vulnerabilidades exploradas e vítimas. Nessa resumo é citado o serviço de *cryptojacking* da *Amazon Web Services* (AWS). O documento destaca várias tendências interessantes. Por exemplo, ele mostra que o *XMRig* é a ferramenta mais comumente usada e que os ataques geralmente aproveitam ferramentas legítimas do sistema para evitar a detecção.

Outro ponto forte do trabalho é que ele examina as abordagens existentes para detectar ataques de *cryptojacking*, destacando suas limitações. Os autores observam que muitos sistemas de detecção atuais dependem de técnicas de análise estática, que são facilmente contornadas por invasores sofisticados. Por fim, o trabalho identifica uma necessidade de sistemas de detecção melhores e baseados em comportamento.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Computação em Nuvem

Na literatura pode-se encontrar diversas definições para o paradigma de computação em nuvem. Computação em nuvem é um modelo que permite, de forma conveniente, o acesso sob demanda de rede, de um conjunto compartilhado de recursos computacionais configuráveis como, por exemplo, redes, servidores, armazenamento, aplicações e serviços que podem ser rapidamente provisionados e liberados com um esforço de gerenciamento mínimo ou interações com o provedor de serviços (MELL *et al.*, 2011).

No modelo tradicional de infraestrutura, as empresas precisam de um grande investimento financeiro e técnico para executar as suas aplicações. Isso envolve a parte de *hardware*, *software*, sistemas operacionais, ou seja, os custos monetários, de gerenciamento e manutenção são elevados (COLLIER; SHAHAN, 2015, pag 17). Além disso, é muito comum se ter uma grande porcentagem de recursos computacionais que ficam ociosos quando se usa uma infraestrutura *on-premises*. Felizmente, com a computação em nuvem é possível ter recursos de TI de uma maneira escalável. Sendo assim, a infraestrutura passa a ser gerenciada na nuvem, permitindo que as organizações tenham uma otimização das aplicações, mais mobilidade e redução de custos (VERAS, 2012, pag).

Os dados armazenados em nuvem podem ser acessados através da *internet*. Além disso, sistemas, plataformas e infraestruturas podem ser acessadas por meio desse serviço, oferecido por empresas especializadas, que cobram conforme o uso, ou seja, é pago somente o que se é consumido. O *National Institute of Standards and Technology* (NIST) define *cloud* como uma tecnologia que deve ter o seguintes características:

- Auto-atendimento sob demanda (*On-Demand Self-Service*): Os usuários tem a capacidade de ajustar, personalizar e configurar os serviços necessários para atender às suas necessidades a qualquer momento através de um portal de autoatendimento. Isso não requer interação humana com o provedor de serviço.
- Amplo acesso a rede (*Broad network access*): O amplo acesso a rede significa que os serviços de *cloud* são acessíveis de qualquer plataforma. São fornecidos mecanismos padrões que promovem o uso de plataformas heterogêneas. Portanto, é possível ter acesso ao conteúdo de um celular, notebook, ou qualquer outra plataforma.

- *Pool* de Recursos (*Resource Pooling*): Os recursos computacionais da nuvem ficam reunidos graficamente. O cliente não possui controle sobre a localização dos recursos implantados, tendo somente uma informação mais genérica como o pais em que se encontra, o estado ou Data center. Os tipos de recursos são: armazenamento, memória e CPU e banda de rede.
- Elasticidade Rápida (*Rapid Elasticy*): Capacidade de alocar mais ou menos recursos no momento em que for necessário e com agilidade.
- Serviços Mensuráveis (Measured Service): Todos os serviços são monitorados e controlados automaticamente pela nuvem, de maneira que fique tudo transparente tanto para o provedor quanto para o cliente. Isso permite que o consumidor otimize sua utilização da nuvem de acordo com sua produção, e ajuda o provedor na hora da cobrança dos recursos.

3.1.1 Modelos de serviço de nuvem

Computação em nuvem normalmente é classificada em três modelos principais, os quais serão detalhados a seguir:

3.1.1.1 Software as a service (SaaS)

SaaS é um software hospedado e gerenciado centralmente para o cliente final. Nesse modelo, o cliente não se preocupa com gerenciamento de infraestrutura ou aplicação, apenas adquire e começa a se beneficiar do serviço. Tem-se como exemplos nesse modelo o *Microsoft One Drive*, *Dropbox*, *WordPress*, e *Amazon Kindle* (COLLIER; SHAHAN, 2015, p. 19).

3.1.1.2 Plataform as a Service (PaaS)

Com o modelo *PaaS*, não é mais necessário gerenciar a infraestrutura subjacente (geralmente hardware e sistemas operacionais. Com isso, o foco do cliente se mantém na implantação e gerenciamento de aplicativos. Dessa maneira, o desenvolvimento de aplicativos fica mais eficiente, pois não é mais necessário se preocupar com aquisição de recursos, planejamento de capacidade, manutenção de *sofware*, correções ou qualquer outro tipo de trabalho genérico redundante necessário para a execução de aplicativos (ANDERSSON, 2023).

3.1.1.3 Infrastructure as a Service (IaaS)

O IaaS contém os componentes básicos da TI na nuvem. Normalmente, o IaaS oferece acesso a recuros de rede, computadores (virtuais ou em hardware dedicado) e espaço de armazenamento de dados. Um fornecedor de nuvem IaaS executa e gerencia infraestrutura que executam software de virtualização, permitindo criar *Virtual Machines* (VM) que são executadas na infraestrutura do provedor de Nuvem (COLLIER; SHAHAN, 2015, p. 19). IaaS oferece o mais alto nível de flexibilidade e controle de gerenciamento sobre recursos de TI na nuvem.

3.1.2 Modelos de implantação de computação em nuvem

Quando se fala em computação em nuvem, existem quatro principais modelos de implantação, os quais serão descritos a seguir.

3.1.2.1 Nuvem Privada

Entende como nuvem privada quando o modelo de nuvem é quase sempre gerenciado e operado pela própria organização do cliente. Nesse modelo, os serviços não ficam disponíveis publicamente para uso geral, apenas para a organização (VERAS, 2012).

3.1.2.2 Nuvem pública

Serviço de nuvem disponibilizado publicamente através do modelo pague-por-uso. Geralmente são oferecidas por organizações públicas ou por grandes industrias que possuem a capacidade de poder computacional (VERAS, 2012). Nesse modelo temos líderes como a *Amazon, Microsoft e Google*.

3.1.2.3 Nuvem comunitária

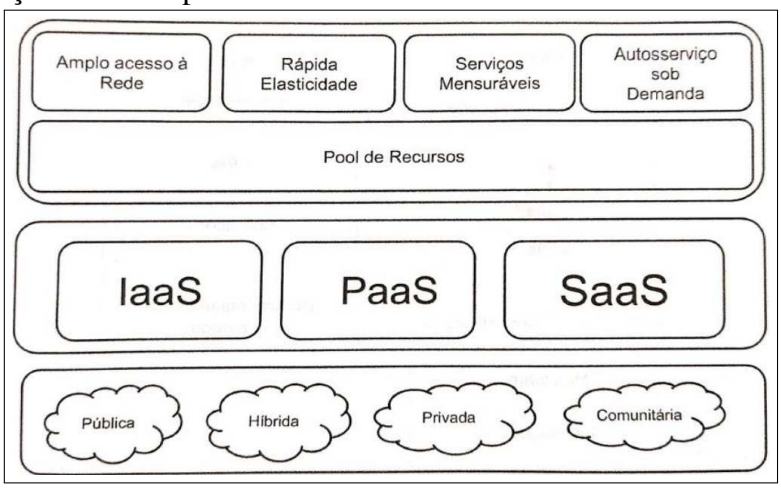
Neste cenário, a infraestrutura de computação em nuvem é compartilhada por várias organizações, suportando uma comunidade com interesses comuns. A nuvem comunitária pode ser gerida pelas próprias organizações que fazem parte dessa comunidade ou por terceiros, e pode estar localizada tanto dentro quanto fora das instalações das organizações (VERAS, 2012).

3.1.2.4 Nuvem híbrida

Essa infraestrutura é uma combinação de duas ou mais nuvens (privadas, públicas ou comunitárias) que permanecem como nuvens distintas, mas estão conectadas por meio de tecnologias proprietárias ou padronizadas, permitindo a portabilidade de dados e aplicações. A nuvem híbrida exige uma coordenação adicional para o uso conjunto das nuvens privadas e públicas (VERAS, 2012).

A fiura 1 resume os conceitos referentes a *CLOUD COMPUTING* consolidados pelo NIST.

Figura 1 – Definição do NIST para CLOUD COMPUTING



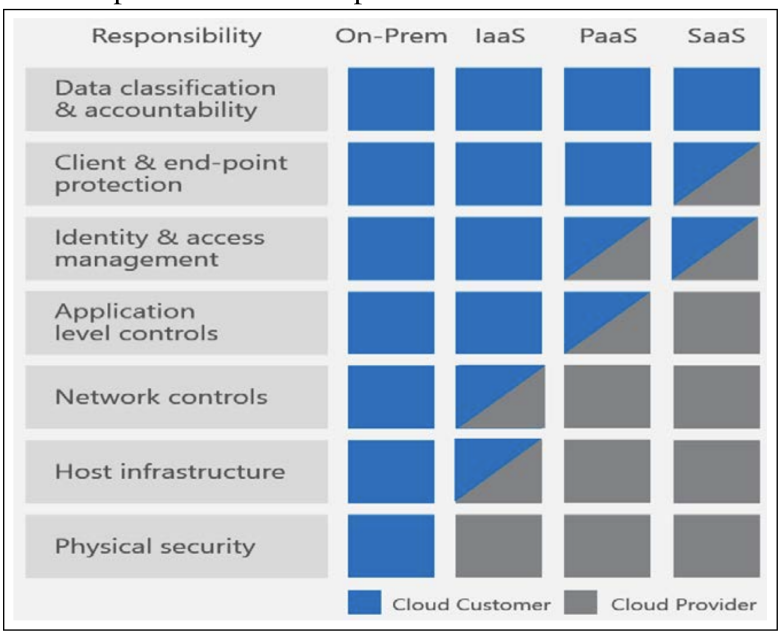
Fonte: (VERAS, 2012)

3.1.3 Modelo de responsabilidade compartilhada

Uma discussão muito comum quando se trata de nuvem é sobre a segurança. A segurança na nuvem não é uma responsabilidade única, mas sim uma responsabilidade compartilhada que envolve tanto o provedor de nuvem quanto a organização ou cliente. Com isso, o provedor é responsável pela segurança "da"infraestrutura da nuvem, incluindo segurança física, infraestrutura de rede e serviços fundamentais, enquanto a organização/cliente é responsável pela segurança "na"nuvem, isso inclui a governança de dados, gestão de acesso de usuários, segurança de aplicações e proteção de *endpoints* (LANE *et al.*, 2017). As responsabilidades variam de acordo com o modelo de serviço em nuvem (SaaS, PaaS, IaaS) e o modelo de implantação (pública, privada, híbrida, comunitária). A figura 2 ilustra os componentes dos serviços de computação em nuvem. É possível notar que o gerenciamento e a segurança vai alterando

também. No modelo SaaS a responsabilidade pelo gerenciamento da infraestrutura de nuvem é toda do provedor, que é o responsável pelo serviços de segurança. No modelo IaaS o cliente passa a ser responsável pela implantanção e pelo gerenciamento de segurança desde o host.

Figura 2 – Modelo de responsabilidade compartilhada



Fonte: (LANE *et al.*, 2017)

3.1.4 Benefícios da Nuvem na Tecnologia da Informação

A computação em nuvem permite que desenvolvedores, engenheiros e profissionais de TI criem, construam, testem e implantem soluções técnicas de forma produtiva, eficaz e segura. As equipes que ainda trabalham em ambientes locais podem enfrentar problemas de tecnológia e perder os benefícios do desenvolvimento em sistemas de computação em nuvem, como maior velocidade de desenvolvimento, testes, manutenção, automação e escalabilidade. Data centers são a base da computação em nuvem, oferecendo infraestrutura necessária para fornecer serviços baseados na nuvem, incluindo armazenamento, poder de computação e conectividade de rede. Eles são projetados para garantir alta disponibilidade, segurança e eficiência energética, com medidas rigorosas de segurança, auditáveis por empresas terceiras, tudo isso para proteger os recursos e ativos na nuvem. Além disso, é considerada uma opção estratégica para negócios e organizações devido à sua velocidade, confiabilidade, economia, produtividade, eficiência, segurança e desempenho. A crescente demanda e evolução das inovações tecnológicas fazem da computação em nuvem uma opção popular, impulsionada também pela missão de aumentar a

sustentabilidade (ANDERSSON, 2023, pag 3-4).

3.2 Microsoft Azure

O *Microsoft Azure*, que é o foco deste trabalho, é uma nuvem pública. A Ela é uma das principais plataformas de computação em nuvem disponíveis no mercado global, como aponta os relatórios do (WRIGHT DENNIS SMITH, 2023). Este serviço oferece uma ampla variedade de serviços em categorias como inteligência artificial, *machine learning, analytics*, computação, *containers, serverless*, bancos de dados, ferramentas de desenvolvimento, *DevOps*, gerenciamento de identidade, integração, *IoT*, gerenciamento e governança de nuvem, mídia e comunicação, *Azure Hybrid*, migração, realidade mista, *mobile*, redes, segurança, armazenamento, web, *Windows Virtual Desktop*, além de outros (ANDERSSON, 2023, pag 17-18). Como afirma em (WRIGHT DENNIS SMITH, 2023):

> A *Microsoft* é uma Líder neste Quadrante Mágico. O *Microsoft Azure* é forte em todos os casos de utilização, mas é particularmente adequado para organizações centradas na *Microsoft* com uma estratégia de nuvem híbrida. O seu vasto canal de parceiros globais também faz da *Microsoft* uma escolha lógica para clientes de pequena e média dimensão em todo o mundo.

A *Microsoft* possui *datacenter* do *Azure* em muitas regiões globais, facilitando para empresas de qualquer tamanho implantar serviços próximos aos seus clientes. O *Azure* é excelente provedor de nuvem para *startups*, permitindo iniciar com baixos custos e aumentar rapidamente sem grandes investimentos iniciais. Ele também oferece flexibilidade para configurar e desativar rapidamente ambientes de desenvolvimento e teste, mantendo os custos baixos e a manutenção mínima (COLLIER; SHAHAN, 2015, pag 18).

3.2.1 Visão geral dos principais serviços do Azure

A *Microsoft* está sempre criando novas soluções e atualizando o serviços existentes. Abaixo temos um breve descrição dos principais serviços hoje no *Azure*.

3.2.1.1 Serviços de Computação no Azure

Os serviços de computação hospedados no *Azure* oferecem recursos de computação como sistemas operacionais, redes, discos, processadores e memória. Esses recursos são disponibilizados de forma rápida e sob demanda para os usuários. Com eles é possível construir aplicações web e móveis, implantar e gerenciar máquinas virtuais, criar aplicativos em *containers*

na nuvem, entre outros. A Tabela 1 ilustra os principais serviços de computação no *Azure*, bem como sua funcionalidade.

Tabela 1 – Serviços de Computação do Azure

Serviço de Computação do Azure	**Para que serve?**
Azure App Service	Criar e desenvolver aplicativos web e móveis em um ambiente de nuvem totalmente gerenciado
Azure Static Web Apps	Desenvolver rapidamente aplicativos web modernos com *Azure* a partir de um repositório de código
Azure Virtual Machines	Provisionamento rápido e gerenciável de máquinas virtuais (VMs do Azure) em diferentes sistemas operacionais como Windows ou Linux
Azure Virtual Machine Scale Sets	Criar e provisionar múltiplas máquinas virtuais (VMs do Azure) com vantagem de alta disponibilidade
Azure Spot Virtual Machines	Economizar dinheiro ao provisionar capacidade de computação que você não usa para suas cargas de trabalho
Azure Functions	Desenvolver aplicativos sem servidor, orientados a eventos e fluxos de trabalho com estado
Azure Container Apps	Criar e implantar aplicativos modernos e microsserviços totalmente gerenciados usando contêineres sem servidor
Azure Kubernetes Service (AKS)	Serviço de *Kubernetes* gerenciado na nuvem
Service Fabric	Criar microsserviços e realizar orquestração de contêineres em diferentes sistemas operacionais como *Windows* e *Linux*

Fonte: Adaptado de (ANDERSSON, 2023)

3.2.1.2 Serviços de Rede

Os serviços de rede são essenciais para proteger a infraestrutura de nuvem, tanto privada quanto pública. Eles permitem que os usuários criem suas configurações de rede na nuvem e gerenciem seus recursos de rede conforme a demanda, isso trás flexibilidade e segurança no ambiente de nuvem. A Tabela 2, demostra os serviços de rede comuns do *Azure*.

Tabela 2 – Serviços de Rede do Azure

Serviço de Rede do Azure	**Para que serve?**
Azure Virtual Network	Conectar recursos do *azure* e comunicação externa
Azure Bastion	Acesso seguro e fácil às suas máquinas virtuais usando *RDP* e *SSH* privados, totalmente gerenciados
Azure Private Link	Acessar serviços hospedados na nuvem Azure com privacidade
Azure Firewall	Proteger seus recursos na nuvem com um *firewall* de alta disponibilidade e baixa manutenção
Azure Load Balancer	Balancear a carga das conexões e solicitações de seus aplicativos, tanto de entrada quanto de saída
Azure ExpressRoute	Criar conexões de rede privadas entre data centers Azure e infraestrutura *on-premises*
Azure Traffic Manager	Roteamento do tráfego de rede para melhor desempenho
Azure VPN Gateway	Criar conexões de rede privadas e seguras na nuvem *VPN*

Fonte: Adaptado de (ANDERSSON, 2023)

3.2.1.3 Principais serviços de Armazenamento

Os serviços de armazenamento no *Azure* são projetados para atender a uma variedade de necessidades de armazenamento de dados. Eles suportam desde o armazenamento de objetos de dados, como arquivos e imagens, até discos de máquinas virtuais e sistemas de mensagens. Esses serviços garantem alta disponibilidade, durabilidade e segurança dos dados, além de serem acessíveis e fáceis de gerenciar. A Tabela 3 apresenta alguns dos serviços de armazenamento mais comuns oferecidos pelo *Azure*, destacando suas funcionalidades.

3.2.2 Conceitos Básicos de Arquitetura e Gerenciamento de Recursos no Microsoft Azure

Há quatro níveis para organizar os recursos da empresa que utiliza *Azure*. Como ilustra a Figura 3, temos *Azure Managment groups*, *Azure Subscriptions*, *Resources groups* e *resources*.

3.2.2.1 Azure Managment Groups

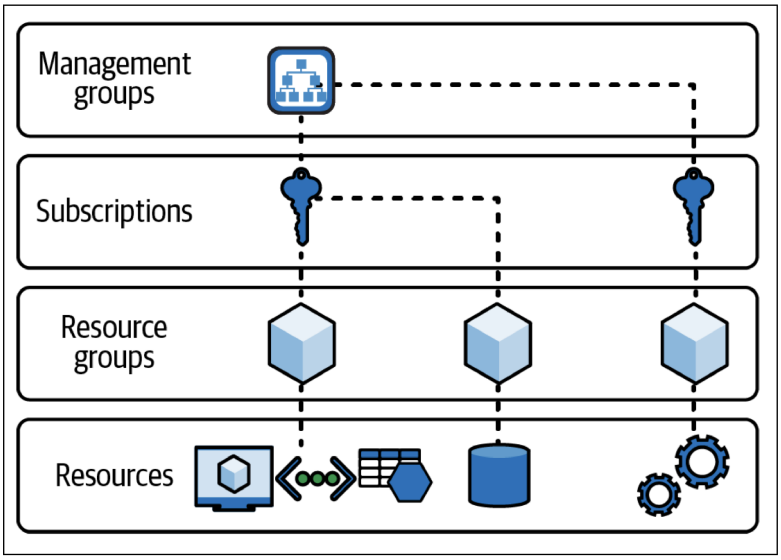
Os grupos de gerenciamento do *Azure* são o nível mais alto da estrutura central para gerenciar seus recursos na nuvem no *Azure*. As assinaturas abaixo de um grupo de gerenciamento

Tabela 3 – Serviços de Armazenamento do Azure

Serviço de Armazenamento do Azure	**Para que serve?**
Azure Blobs	Armazenar dados binários escaláveis, texto ou dados para análises de *big data* com o *Data Lake Storage Gen2*
Azure Files	Compartilhamentos de arquivos totalmente gerenciáveis para implantações *on-premises* ou na nuvem. Acessível em qualquer lugar através do protocolo **Server Message Block (SMB)**
Azure Queues	Usando Filas, você pode coletar grandes mensagens que são acessadas via chamadas HTTP autenticadas
Azure Managed Disks	Armazenar volumes em nível de bloco para Máquinas Virtuais do *Azure*

Fonte: Adaptado de (ANDERSSON, 2023)

Figura 3 – Quatro níveis dos recursos no Azure



Fonte: Adaptado de (LANE *et al.*, 2017)

herdam automaticamente as configurações, condições e restrições aplicadas nele.

O controle de acesso do *Azure* para todos os recursos é suportado nos grupos de gerenciamento raiz. Qualquer pessoa na organização com qualquer função no *Azure* pode ser atribuída a um grupo de gerenciamento do *Azure*. Quem tiver acesso e direitos ao grupo de gerenciamento podem agrupar assinaturas do *Azure*, visualizar os grupos de gerenciamento e a hierarquia da organização e, principalmente, controlar o acesso a qualquer serviço ou recurso do *Azure*, criando e aplicando controles de governança e políticas (ANDERSSON, 2023, pag 44).

3.2.2.2 Azure Subscription

Assinaturas no *Azure* são como um grande *container* que serve para agrupar contas e recursos. As assinaturas são como as organizações controlam os custos e cobrança mensal na nuvem da *Microsoft*. Usando as assinaturas, as organizações podem controlar o acesso aos recursos pelos usuários como criar, atualizar ou deletar esses recursos.

3.2.2.3 Azure Resource Groups

Os grupos de recursos no *Azure* são utilizados como um *container* lógico para agrupar os recursos no *Azure*. Usuários podem usar o grupo de recursos para organizar serviços de um mesmo projeto, deixando seu ambiente segregado e organizado.

3.2.2.4 Azure Resources

Banco de dados, servidores, máquinas virtuais, aplicativos web são criados recursos no *Azure*. Todos os recursos que uma organização cria precisam ser agrupados em um grupo de recurso. Ao criar um recurso do *Azure*, é necessário selecionar uma assinatura do *Azure*, um grupo de recursos e a região do Azure, onde deseja adicionar seus recursos (ANDERSSON, 2023, pag 45). Existem recursos que são globais, não sendo necessários escolher uma região.

3.3 Kubernetes

Kubernetes é um sistema de orquestração de *containers* de código aberto que foi criado pela *Google* e doado para a *Cloud Native Computing Foundation (CNCF)* em março de 2016. Ele gerencia aplicativos em *containers* em vários *hosts* para implantação, monitoramento e dimensionamento de *containers* (KUBERNETES, 2019).

Kubernetes, também conhecido como "k8s", permite que o usuário defina o como um aplicativo deverá ser implantado. Por exemplo, o usuário pode especificar que o aplicativo deverá ter três instâncias de uma aplicação web *Tomcat* em execução. O *Kubernetes* gerencia o início e o monitoramento contínuo dos *containers*, oferecendo recursos como reinicialização automática, reagendamento e replicação para garantir que o aplicativo se mantenha no estado desejado. O *Kubernetes* pode ser instalado de forma independente ou através de diversas distribuições, como *Red Hat OpenShift* (KUBERNETES, 2019). Além disso, pode ser gerenciado em ambientes de

nuvem, como o *Azure Kubernetes Services*.

3.3.1 Conceitos Básicos do Kubernetes

Recursos do *Kubernetes* pode ser criados via linha de comando, mas normalmente são através de arquivos *YAML (Yet Another Markup Language (YAML)*.

3.3.1.1 Containerização

Conteinerização é o processo de empacotar uma aplicação e suas dependências com uma imagem e roda-la como um *container* (POULTON, 2023, pag 3). Dessa forma, tem-se um ambiente consistente e portátil para a aplicação, independentemente de onde é executada. De maneira geral, imagens de *containers* são hospedadas em registro de *containers*, o mais famoso registro de imagens é o *docker hub*.

3.3.1.2 POD

Como afirma (POULTON, 2023) "Cada *pod* é um ambiente de execução compartilhado para um ou mais *containers*". O ambiente de execução inclui toda a parte de rede, volumes, memória, entre outros.

3.3.1.3 Deployments

Um *deployment* fornece escalonamento de *pod* e atualizações contínuas. O *Kubernetes* irá garantir que o número desejado de *pods* esteja em execução e, em uma atualização contínua, substituirá as instâncias do *pod*, uma por vez, permitindo atualizações de aplicativos sem tempo de inatividade (KUBERNETES, 2019).

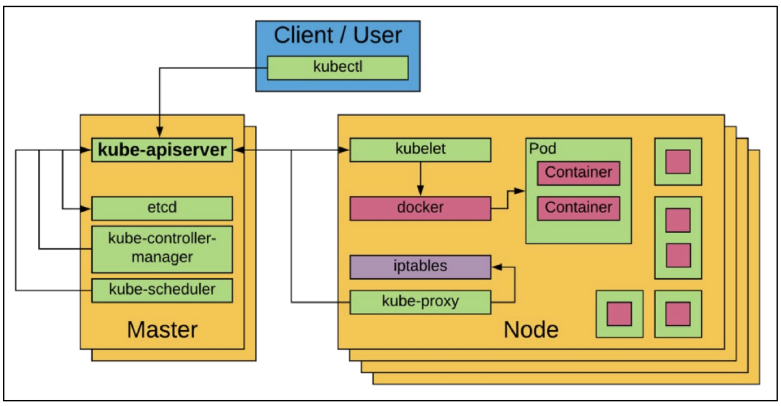
3.3.1.4 DaemonSet

Um *DaemonSet* garante que todos ou alguns Nodes executem uma cópia de um *Pod*. À medida que novos *Nodes* são adicionados ao *cluster*, os *Pods* são implantados neles automaticamente. Quando *Nodes* são removidos do *cluster*, os *Pods* correspondentes são excluídos. Ao deletar um *DaemonSet*, todos os *Pods* que ele criou são removidos (DAEMONSET, 2024).

3.3.2 Arquitetura do Kubernetes

Como descrito em (KUBERNETES, 2019), o *Kubernetes* usa uma arquitetura cliente-servidor. Um *cluster Kubernetes* é um conjunto de máquinas físicas ou virtuais e outros recursos de infraestrutura usados para executar aplicativos. A Figura 4 demostra a arquitetura do *Kubernetes*, onde as máquinas que gerenciam o *cluster* são chamadas de *Masters*, e as máquinas virtuais que executam os *containers* são chamadas de *Nodes*.

Figura 4 – Arquitetura do Kubernetes



Fonte: Adaptado de (KUBERNETES, 2019)

3.3.2.1 MASTER

O *Master* mantém os serviços responsáveis pela gestão do *cluster*. Entre eles, o mais crucial é o *kube-apiserver*, que serve como o principal ponto de acesso para clientes e *nodes* que desejam consultar ou modificar recursos no *cluster*. O servidor de *API* opera em conjunto com: *etcd*, que é um armazenamento distribuído de chave-valor utilizado para manter o estado do *cluster*; *kube-controller-manager*, que monitora o sistema e determina quais ajustes são necessários quando há mudanças nos recursos; e *kube-scheduler*, que aloca *pods* aos nós apropriados com base na disponibilidade e configuração dos mesmos (KUBERNETES, 2019).

3.3.2.2 NODE

NODE é uma máquina física ou virtual com os serviços necessários para executar os *containers*. Em um *cluster Kubernetes*, deve se ter muitos nodes afim de atender todos os *pods* necessários. Como mostra a figura 4, cada *node* tem dois serviços *kubernetes*: *kubelet*, o qual é responsável por garantir que os *pods* estejam rodando no *node*; e o *kube-proxy*, o qual gerencia

toda a parte de regras de rede dentro do *node*. Além disso, cada *node* executa múltiplos *pods*, onde cada pode irá conter um ou mais *containers* (KUBERNETES, 2019).

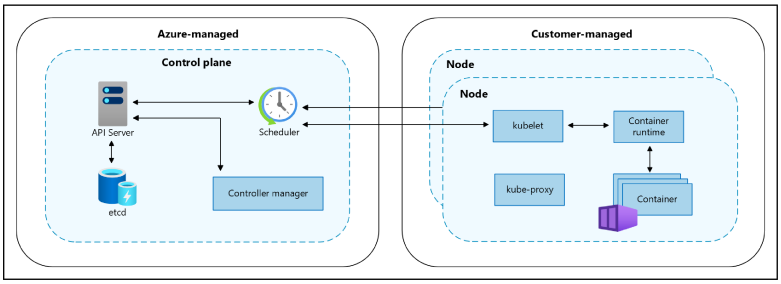
3.4 Azure Kubernetes Services

O *Azure Kubenetes Service (AKS)* é o serviço de *Kubernetes* gerenciado da *Microsoft*. Ele simplifica a implantação, o gerenciamento e o escalonamento de aplicativo que rodam em *containers*, através do *Kubernetes*. O AKS trás grande vantagem ao reduzir a complexidade e a sobrecarga operacional de ter que gerenciar uma infraestrutura física para o *kubernetes* (SCHAFFERERIN TYNEVI, 2024). Olhando para a Figura 2 teríamos o *Azure Kubernetes Servic*e como um serviço de *IaaS*, transferindo boa parte do gerenciamento da infraestrutura para o *Azure*.

3.4.1 Componentes de um cluster ASK

O *cluster* do AKS é basicamente o mesmo que o apresentado na Figura 4. A principal diferença se da nas responsabilidades de gerenciamento e nomenclatura. A Figura 5 demostra os dois componentes do AKS:

Figura 5 – Arquitetura do Azure Kubernetes Service



Fonte: (SCHAFFERERIN TYNEVI, 2024)

3.4.1.1 Painel de controle

O painel de controle é totalmente gerenciado pelo *Azure*. Ele é equivalente ao *Master* na Figura 4. A Tabela a seguir descreve o componente adicional quando se trata de *kubernetes* em nuvem.

Tabela 4 – Descriçao do cloud-controller-manager no Control plane

Componente	Descrição
cloud-controller-manager	O gerenciador de controladores da nuvem (*cloud-controller-manager*) integra a lógica de controle específica para nuvem, executando controladores específicos para o provedor de nuvem.

Fonte: Adaptado de (SCHAFFERERIN TYNEVI, 2024)

3.4.1.2 Node

Assim como na sessão 3.3.2, o *Node* no AKS é uma máquina virtual, nesse caso, uma máquina virtual do Azure responsável por executar os componentes do *Node* do *Kubernetes*.

3.4.1.3 Condiguração de Controle Plane e Node

(SCHAFFERERIN TYNEVI, 2024) menciona que "O tamanho da VM do *Azure* para seus nós define as CPU, memória, tamanho e tipo de armazenamento disponíveis, como *Solid State Drive* (SSD) de alto desempenho ou *Hard Disk Drive* (HDD) comum". Além disso, o AKS é compativel com sistemas operacionais ubuntu e *Azure linux*. Os tamanhos de VM do *Azure* são projetados para fornecer uma ampla gama de opções para hospedar serviços na nuvem.

3.5 Cryptojacking

3.5.1 Mineração de criptomoedas

Criptomoeda é um ativo digital registrado em *blockchain* que usa criptografia para garantir a rastreabilidade e a integridade das transações. Diferente de moedas tradicionais, elas não são emitidas ou regulamentadas por uma autoridade central, mas operam em um sistema distribuído que registra as transações e a emissão de novas unidades. Essa tecnologia tem se tornado atraente devido ao alto lucro resultante de sua valorização (PASSOS *et al.*, 2024).

A mineração de criptomoedas é o método utilizado para criar novas unidades, o que envolve a resolução de desafios matemáticos complexos com o uso de computadores altamente potentes. Mineradores competem entre si para resolver esses desafios, e aqueles que têm sucesso são recompensados com novas unidades da criptomoeda, além das taxas de transação geradas pela criação de novos blocos na rede. Nos últimos anos, a mineração de criptomoedas tornou-se uma atividade altamente lucrativa, atraindo cibercriminosos que começaram a praticar o *cryptojacking* (PASSOS *et al.*, 2024).

3.5.2 Definição de cryptojacking

(TEKINER *et al.*, 2021) define *cryptojacking* como:

> *Cryptojacking* é o ato de usar o poder computacional da vítima sem consentimento para minerar criptomoedas. Esta operação de mineração não autorizada custa um consumo extra de eletricidade e diminui drasticamente a eficiência computacional do *host* vítima. Como resultado, o invasor transforma esse poder computacional não autorizado em criptomoeda. Na literatura, o *malware* utilizado para esse fim é conhecido como *cryptojacking*.

As plataformas de nuvem são alvos atraentes devido aos seus vastos recursos, ampla superfície de ataque e capacidades robustas de conexão com a internet, permitindo períodos prolongados de mineração. O número de *malwares* de *cryptojacking* direcionados à infraestrutura em nuvem está aumentando anualmente, impactando principalmente grandes empresas. Como exemplo de empresas afetadas, tem-se clientes de *clusters* do *Azure Kubernetes*, principalmente devido a configurações inadequadas de segurança (TEKINER *et al.*, 2021).

3.5.3 Plataformas alvo de Cryptojacking

A seguir é descrito as plataformas mais comuns alvos de atividades de *cryptojacking*.

3.5.3.1 Navegador

Os navegadores são a plataforma de vítima mais comum porque oferecem um caminho direto para a CPU do usuário. Eles são facilmente acessíveis e os *scripts* de *cryptojacking* podem ser facilmente incorporados em páginas da web ou distribuídos por meio de extensões de navegador (TEKINER *et al.*, 2021).

3.5.3.2 Servidores On-premise (local)

Os servidores, localizados fisicamente nas organizações oferecem um bom poder de processamento que pode ser explorado para mineração de criptomoedas (TEKINER *et al.*, 2021).

3.5.3.3 Servidores em Nuvem

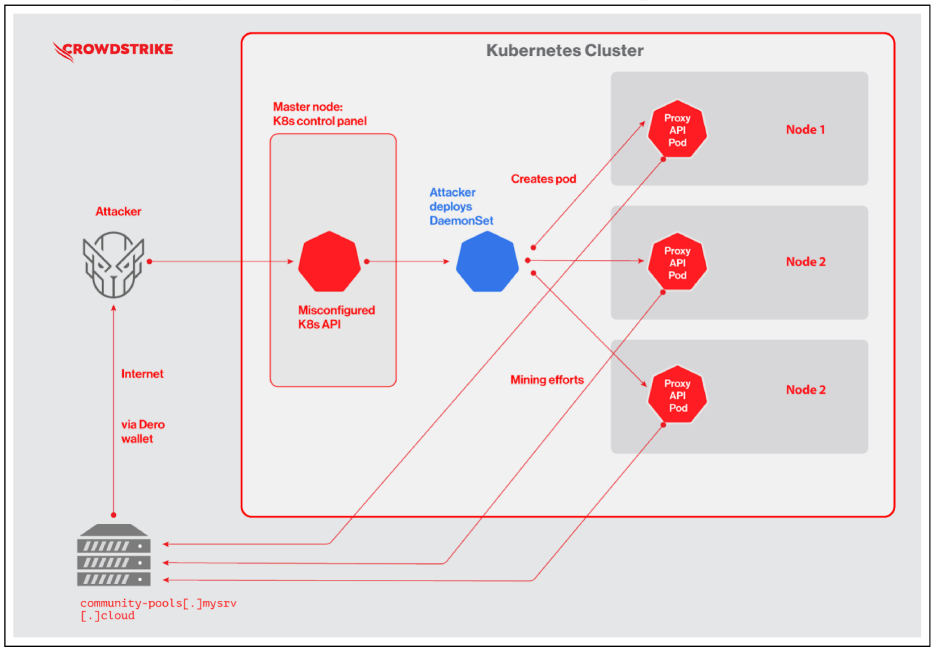
A quantidade de atividades de *cryptojacking* visando infraestrutura em nuvem está aumentando a cada ano, especialmente plataformas de Infraestrutura como Serviço (*IaaS*). Entre os motivos para o aumento de atividades de mineração de criptomoedas estão os recursos virtual-

mente infinitos, a conexão confiável com a Internet e o período mais longo de mineração/lucro devido às capacidades baseadas no *host* (TEKINER *et al.*, 2021).

3.5.4 Fluxo de ataque do Criptojacking em ambientes kubernetes

A Figura 6 ilustra o fluxo do ataque de *cryptojacking* em *clusters kubernetes* de maneira genérica.

Figura 6 – Fluxo de ataque de *cryptojacking* em *Kubernetes* para minerar *Dero*



Fonte: (GRAP, 2023)

A ameaça de *cryptojacking* em *clusters Kubernetes* pode variar em cada ação, mas seu comportamento geral é consistente. Conforme ilustrado na Figura, o atacante precisa obter um acesso inicial ao *cluster*. Após isso, ele cria *pods* no *cluster* que contêm imagens personalizadas para iniciar a execução de atividades de mineração de criptomoedas. No relatório de (GRAP, 2023), o ataque visa implantar um *DaemonSet* que cria um *pod* malicioso em cada *node* do *cluster Kubernetes*, permitindo que os invasores utilizem os recursos de todos os nodes simultaneamente para realizar uma operação de *cryptojacking*. Essa campanha de ataque visa a mineração da criptomoeda *Dero*.

3.5.5 Detectar atividades de cryptojacking

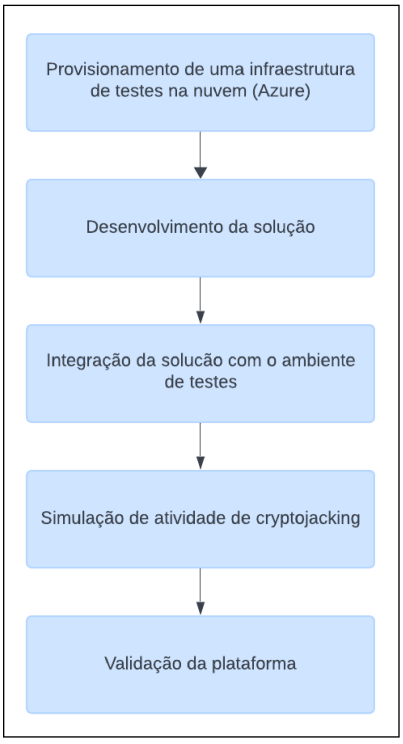
Conforme descrito por (TEKINER *et al.*, 2021), o alto uso de CPU continua sendo o principal indicador para a detecção de atividades de *cryptojacking*, uma vez que a mineração de

criptomoedas depende muito desse recurso. Além disso, o uso elevado de memória também é um sinal relevante para identificar possíveis atividades de *cryptojacking*. A análise de processos e binários que indicam essas atividades, também são métodos usados para validar se há atividades de mineração no sistema (GRAP, 2023). Entre os meios mais eficazes para detecção, destacam-se a análise de pacotes de rede e as chamadas de sistema. Todos esses recursos demonstram uma eficácia ainda maior quando combinados com técnicas de aprendizado de máquina.

4 METODOLOGIA

Esta seção apresenta a metodologia utilizada para o desenvolvimento de uma solução de monitoramento para detectar atividades de *cryptojacking* no *Azure Kubernetes Service (AKS)*. O método proposto neste trabalho para identificar mineração de criptomoedas foi testado em um ambiente de nuvem. Durante a execução do experimento, foram coletados dados primários na forma de métricas do sistema, que foram utilizados na solução web desenvolvida. As etapas da metodologia de pesquisa adotada para atingir os objetivos deste estudo é ilustrada na Figura 7

Figura 7 – Fluxo da metodologia utilizada



Fonte: Elaborado pelo autor.

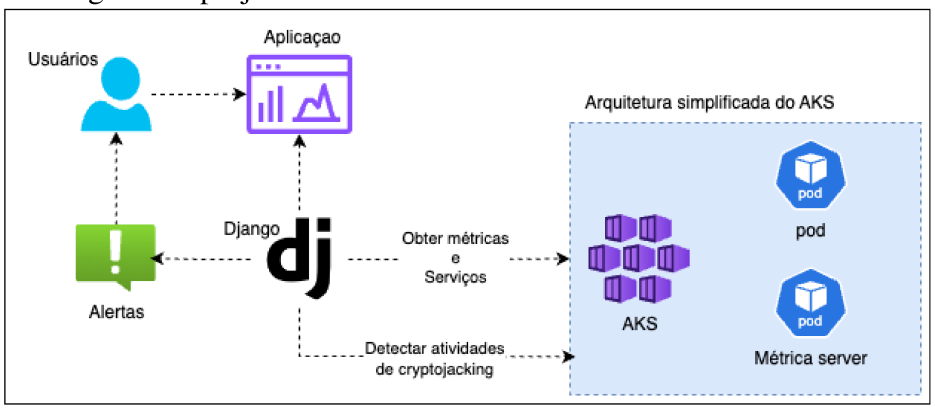
4.1 Arquitetura do sistema

4.1.1 Visão geral do projeto

A Figura 8 demostra a arquitetura geral deste trabalho. Nela os usuários tem acesso a aplicação web para visualizar métricas e recursos do *Azure Kubernetes Service*. A aplicação web, desenvolvida com *Django*, atua como a interface principal para os usuários, exibindo métricas detalhadas de uso de CPU em porcentagem e de memória ao longo do tempo. Esses dados ficam sendo representados graficamente com a biblioteca *Charts* e são renderizados em tempo

real na página web, permitindo aos usuários monitorar os recursos do AKS. A aplicação se comunica com o AKS para coletar esses dados de desempenho, como o uso de CPU e memória do *node* e dos *pods* para exibir na aplicação, que são atualizados dinamicamente com o auxílio de *Javascript*. Alertas são emitidos quando o consumo anormal de CPU no *Node* for detectado. O usuário, através da aplicação, consegue investigar processos que possam indicar atividades de *cryptojacking*.

Figura 8 – Visão geral do projeto



Fonte: Elaborado pelo autor.

4.1.2 Componentes principais

4.1.2.1 Azure Kubernetes Service

A primeira etapa envolve a criação e configuração do ambiente para testes. Nessa etapa é provisionado o serviço do *Azure Kubernetes Service*. Foi utilizada uma assinatura que a *Microsoft* disponibiliza para estudantes. Foi considerado um AKS com 1 *Node* com configuração de 2 *Virtual Central Processing Unit* (VCPU) e 7 gigas de RAM. A Figura 9 demonstra o serviço do AKS criado.

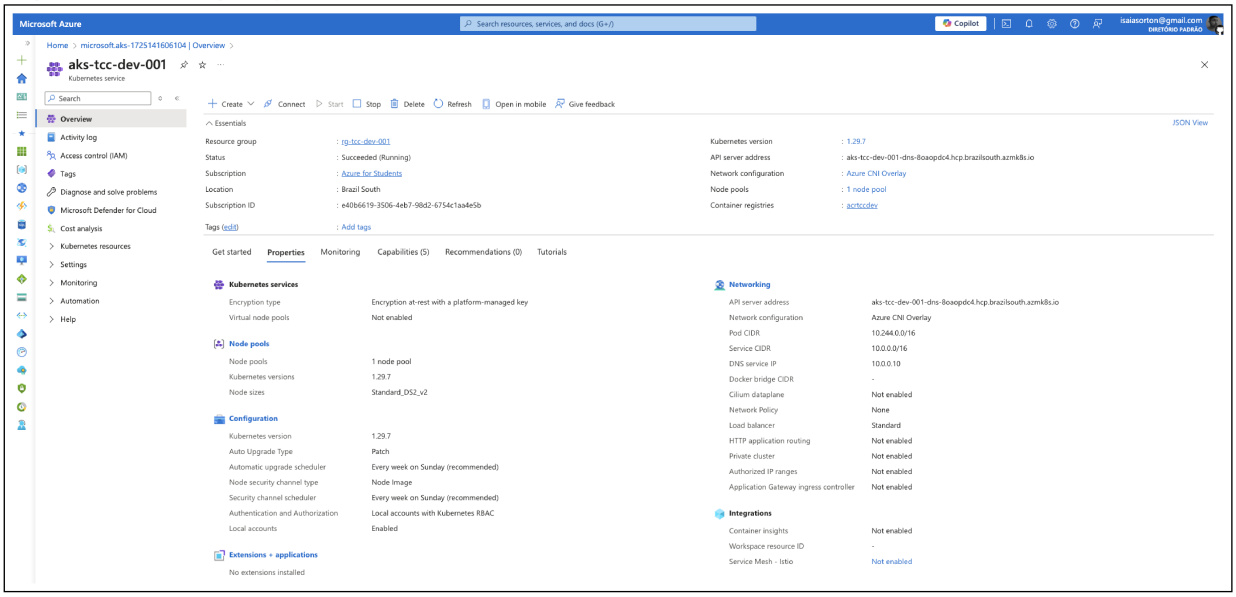
4.1.2.2 Framework Django

O *Django*[1] é um *framework* para criação de aplicações Web escrito em *Python*. Ele usa a arquitetura *model-template-view*:

- *Model*: *Django* vem com uma solução para mapeamento objeto-relacional (ORM,do inglês *Object-Relational Mapping*) no qual o esquema do banco de dados é descrito em

[1] **Django Web Framework (Versão 4.2.15)**. Disponível em: <https://www.djangoproject.com/>. Acesso em: 2 setembro 2024

Figura 9 – Configuração do *Azure Kubernetes Service*



Fonte: Elaborado pelo autor.

código *Python* (SANTIAGO *et al.*, 2020). Além disso, é possível fazer consultas e fazer atualizações no banco sem a necessidade de consultas diretas ao banco de dados.

- *Template*: Esta é a camada de apresentação web, é onde as informações são visualizadas pelos usuários. Um *template* nada mais é que de arquivos estáticos, como o *HTML* (SANTIAGO *et al.*, 2020).
- *Views*: As *views* no *Django* são responsáveis por receber as requisições que vêm do lado do cliente (SANTIAGO *et al.*, 2020). De maneira geral, depois de processar ou recuperar os dados necessários, a *view* prepara uma resposta e devolve ao cliente, no cenário desse trabalho, ela devolve dados para serem renderizados nos *templates*.

O *Django* foi considerado para o desenvolvimento da solução, por usa facilidade em construir aplicações web, além de ter diversas funcionalidades prontas, como autenticação, gerenciamento de usuários, formulários e administração. Isso acelera o desenvolvimento, permitindo o foco na lógica de monitoramento do que em detalhes técnicos.

4.1.2.3 Outras tecnologias utilizadas

Para a construção visual da aplicação, foi aplicado as seguintes ferramentas:

1. *Bootstrap*[2]: Um *framework* gratuito que permite a criação de sites e aplicações responsivas, de forma rápida e simples. Ele é uma ferramenta para desenvolvimento de *HTML, CSS e JS*, que oferece diversas vantagens como a agilidade para construir componentes web.

[2] **Bootstrap (Versão 4.6.2)**. Disponível em: <https://getbootstrap.com/>. Acesso em: 2 setembro 2024

2. *Kubernetes Python Client*[3]: Essa biblioteca facilita a interação com a *API* do *Kubernetes*. Ela lida automaticamente com autenticação e autorização, podendo utilizar credenciais obtidas a partir de uma conta de serviço *Kubernetes* ou de um arquivo *kubeconfig*, dependendo de onde o código está sendo executado (dentro ou fora de um *cluster*) (KUBERNETES, 2024).
3. *Chart.js*[4]: Uma das mais populares bibliotecas *Javascript* para criação de gráficos.
4. *SQLite3*: É uma biblioteca que implementa um mecanismo de banco de dados SQL transacional independente, sem servidor e sem configuração.
5. *Javascript*: Uso da linguagem para funcionalidades dinâmicas na aplicação, como comportamento dos gráficos em tempo real.
6. *Mailtrap*: O *mailtrap*[5] é uma plataforma de entrega de e-mail que oferece uma caixa de entrada virtual para capturar e visualizar e-mails enviados por uma aplicação.

4.2 Desenvolvimento da Solução

Os tópicos abaixo descrevem o processo para a construção da aplicação de monitoramento e como ela se integra ao AKS e emite alertas.

4.2.1 Requisitos Funcionais

Requisitos funcionais são as funcionalidades e serviços do sistema descritos explicitamente, documentando como o sistema deve reagir a entradas específicas, como deve se comportar em determinadas situações e o que o sistema não deve fazer (FIGUEIREDO, 2011).

4.2.1.1 Tela de Login e Cadastro

A aplicação deve incluir uma interface para login e cadastro de usuários. Na tela de cadastro, os usuários poderão criar uma conta ao fornecer nome de usuário, *e-mail* e senha. Após o registro, o acesso à aplicação será feito mediante a autenticação usando as credenciais de nome de usuário e senha. Além disso, as interfaces de login e cadastro devem estar interligadas, permitindo a transição entre elas de forma intuitiva, por meio de um botão, para facilitar a

[3] **Kubernetes Python Client (Versão 30.1.0)**. Disponível em: <https://github.com/kubernetes-client/python>. Acesso em: 2 setembro 2024
[4] **Chart.js**. Disponível em: <https://www.chartjs.org/>. Acesso em: 2 setembro 2024
[5] **Mailtrap**. Disponível em: <https://mailtrap.io/>. Acesso em: 2 setembro 2024

navegação do usuário.

4.2.1.2 Integração com Azure Kubernetes Service

O sistema deve integrar-se de forma transparente com o serviço do AKS, utilizando *APIs* do *Kubernetes* para obter informações sobre os *nodes* e *pods*. O sistema deve ser capaz de operar em tempo real, garantindo que as métricas e os dados apresentados na interface sejam atualizados frequentemente.

4.2.1.3 Monitoramento de Métricas dos Nodes e Pods

O sistema deve coletar e exibir métricas de uso de CPU e memória dos nodes no *cluster AKS*. Além disso, deve listar todos os *pods* em execução no *cluster*, exibindo suas respectivas métricas de uso de CPU e memória.

4.2.1.4 Listagem de Pods

O sistema deve exibir uma lista de todos os *pods* em execução no *cluster*, mostrando informações como nome do *pod*, *namespace*, uso de CPU e memória. Essa listagem deve ser atualizada periodicamente para refletir o estado atual dos *pods* no *cluster*.

4.2.1.5 Visualização de Dados

O sistema deve gerar e exibir gráficos dinâmicos e atualizados em tempo real, representando o uso de CPU e memória dos *nodes*. Os gráficos devem ser acessíveis via interface web e devem permitir ao usuário visualizar a evolução do uso de recursos ao longo do tempo.

4.2.1.6 Alertas

O sistema deve notificar o usuário via *e-mail* caso o alto uso de CPU seja identificado no *Node*. Esse processo se alinha como a primeira linha de detecção de atividades de *cryptojacking*. Foi usado para os testes da aplicação para o envio de *e-meails* o *mailtrap*.

4.2.1.7 Detecção Manual de Cryptojacking

O sistema deve fornecer um botão ao lado de cada *pod* na interface, permitindo ao usuário iniciar manualmente uma verificação de processos que possam indicar atividades de *cryptojacking* em execução dentro do *pod*. Ao clicar no botão, o sistema deve comparar os processos em execução no *pod* com uma base de dados de processos conhecidos por estarem associados a *cryptojacking*. O sistema deve permitir que o usuário visualize os detalhes dos processos maliciosos identificados, incluindo o nome do processo, o *hash* do binário e caminho do binário.

4.2.1.8 Atualização da base de dados de processos conhecidos

O sistema deve permitir que sejam adicionados novas assinaturas de *cryptojacking* a base de dados de atividades de *cryptojacking* conhecidos. Essa base é composta por nome do processo, caminho do binário do arquivo, *hash* do arquivo.

4.2.2 Requisitos Não Funcionais

Os requisitos não-funcionais definem propriedades e restrições do sistema, como segurança, desempenho e espaço em disco. Esses requisitos podem se aplicar ao sistema como um todo ou a partes específicas dele (FIGUEIREDO, 2011).

4.2.2.1 Desempenho

O sistema deve ser capaz de coletar e exibir métricas em tempo real, com um atraso mínimo, não superior a 5 segundos entre a coleta e a exibição dos dados. O tempo de resposta para a detecção manual de *cryptojacking*, desde o clique no botão até a exibição do resultado, deve ser inferior a 5 segundos.

4.2.2.2 Segurança

O acesso ao sistema deve ser restrito a usuários autenticados e autorizados, com autenticação baseada em padrões seguros.

4.2.2.3 Confiabilidade e Disponibilidade

O sistema deve ter uma alta disponibilidade, com um tempo de *uptime* de igual ao do serviço de AKS o qual ele estará configurado, garantindo que os usuários possam acessar o monitoramento de forma consistente.

4.2.2.4 Manutenibilidade

Para facilitar a manutenção e continuidade do sistema, ele deve usar boas práticas de software como uso de programação modular. Isso garante uma facilidade para atualizar adicionar novos recursos ao sistema.

4.2.2.5 Portabilidade

A configuração do sistema deve ser gerenciada via arquivos de configuração, permitindo fácil adaptação a diferentes ambientes de AKS. Dessa forma, a instalação do sistema deve funcionar em qualquer serviço de *Azure Kubernetes Service*.

4.3 Integração da solução com o ambiente de testes

Para se integrar ao serviço do *Azure Kubernetes Service*, o sistema usa a biblioteca *kubernetes* para *python*. Ela permite a integração com o a *API* do *Kubernetes*, passando pelo processo de autenticação e autorização. Esse processo pode ser feito a partir da leitura do arquivo de configuração do *kubernetes*. Isso permite que, ao implantar o sistema em qualquer AKS, ele consiga se comunicar e realizar as operações.

4.3.1 Coleta de Métricas

Para o processo de coleta de métricas, foi usado o *Kubernetes Metrics Server*, que é um agregador de dados de uso de recursos no *cluster*. Ele pode ser implantado na hora da configuração do *cluster AKS* ou usando uma implantação via arquivos *YAMl*. Através dele, é obtido as métricas dos *pods* rodando no *cluster*. O sistema, através das *Views* no *Django*, interage com a *API* do *kubernetes*, recupera as métricas e as renderiza através dos *templates* no *Django*.

4.3.2 Alertas

Para o teste do protótipo do sistema, usa-se a regra de observar as métricas do uso de Node. Quando o uso de CPU for superior a 80% por mais de cinco minutos, é enviado um *e-mail* para o destinatário do sistema, indicado que há uma anomalia de consumo de CPU. Este primeiro passo servirá para a análise manual de atividades de *cryptojacking* no ambiente.

4.3.3 Mecanismo de Detecção de Cryptojacking

Para detectar atividades suspeitas, além da visão gráfica dos *pods* rodando no AKS e do monitoramento de CPU e memória, o sistema permite que seja feita uma analise manual aos *pods* rodando. Essa analise verifica por processos suspeitos, *hashs* de binários rodando no *container* e o caminho desses binários. Ao se conectar com os *pods*, a aplicação busca processos em execução no *container* e compara com o banco de dados para validar se existe algo registrado na base de dados que possa indicar uma atividade de *cryptojacking*. O processo de validar *hash* de binário e o caminho onde ele esta sendo executado, funciona da mesma maneira.

4.3.4 Simulação de atividade de cryptojacking

Para o proposito educacional, na pesquisa foi utilizado o *XMRig*[6] que é um software de mineração de criptomoedas voltado principalmente para *Monero*, mas também compatível com outras moedas. Foi instalada a versão com configurações para execução exclusiva em CPU no ambiente de testes no AKS. O código a seguir representa um exemplo para implantação de uma imagem para minerar *monero* no *Kubernetes*.

```
1    apiVersion: v1
2    kind: Namespace
3    metadata:
4      name: xmrig
5      labels:
6        name: xmrig
7    ---
8    apiVersion: apps/v1
9    kind: Deployment
```

[6] **XMRig (Versão 6.22.0)**. Disponível em: <https://github.com/xmrig/xmrig>. Acesso em: 2 setembro 2024

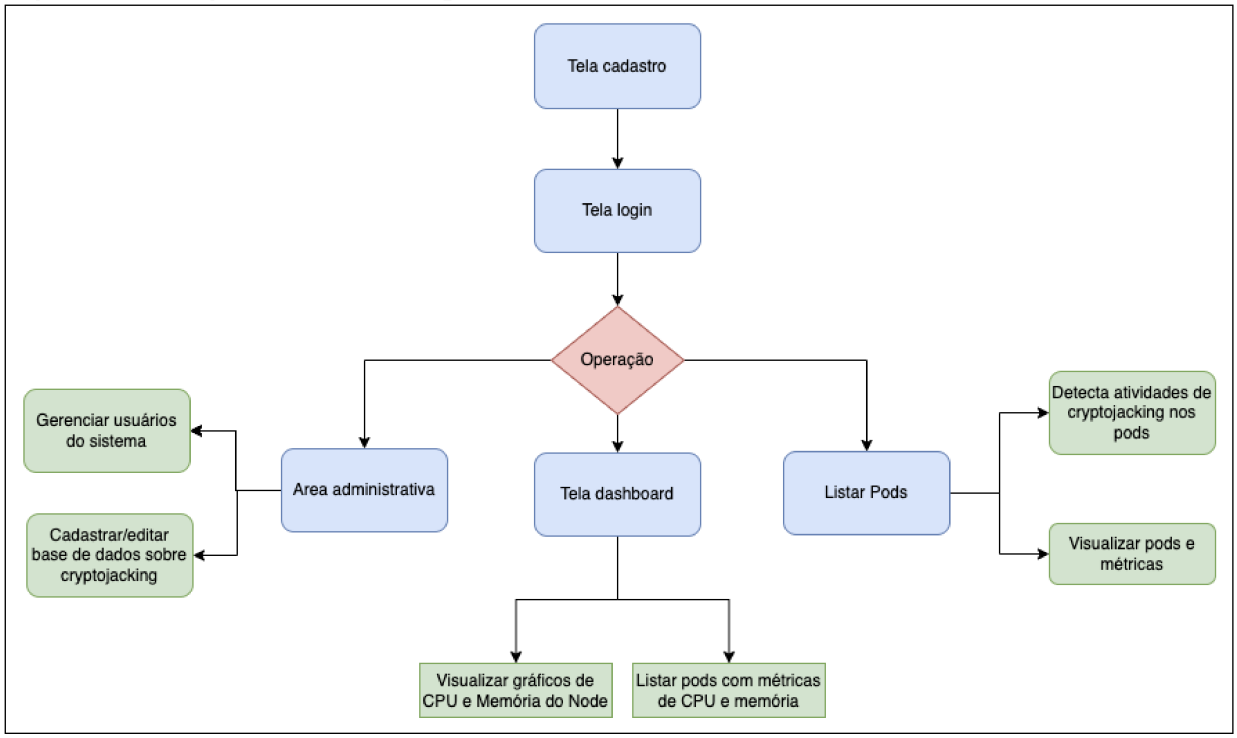
```
10    metadata:
11      namespace: xmrig
12      name: xmrig-deployment
13      labels:
14        app: xmrig
15    spec:
16      replicas: 9
17      selector:
18        matchLabels:
19          app: xmrig
20      template:
21        metadata:
22          labels:
23            app: xmrig
24        spec:
25          containers:
26            - name: xmrig
27              image: imagem/alpine-xmrig:v6.22.0
28              imagePullPolicy: Always
29              command:
30                - ./xmrig
31                - --donate-level
32                - '1'
33                - --max-cpu-usage
34                - '95'
35                - -o
36                - xmr.pool.minergate.com:45700
37                - -u
38                - aluno@email.com
39                - -p
40                - x
41                - -k
```

5 RESULTADOS E DISCUSSÕES

5.1 Diagrama geral do sistema Kubesilent

Como resultado de desenvolvimento do sistema *kubeSilente*, o diagrama na Figura 10, demostra os passos de acesso disponíveis no protótipo do sistema *kubesilent*. As telas desenvolvidas são detalhadas a partir do fluxograma ilustrado.

Figura 10 – Diagrama de fluxo da plataforma



Fonte: Elaborado pelo autor.

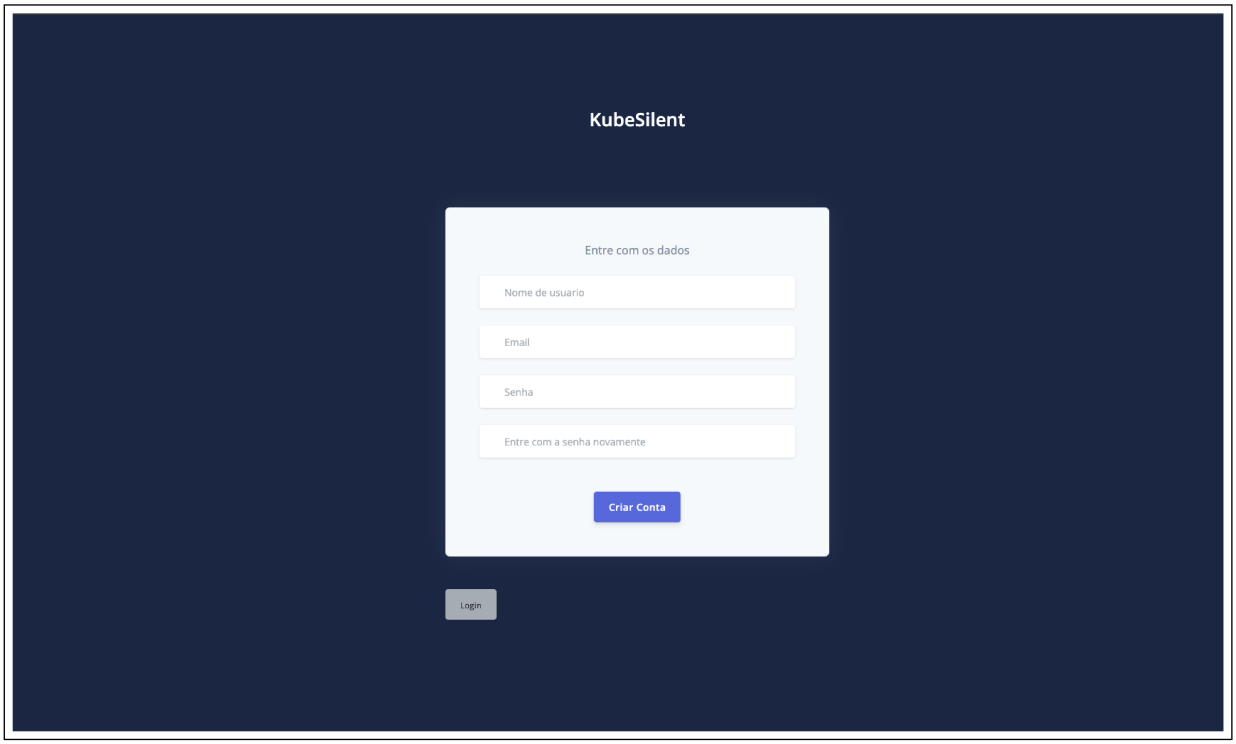
5.1.1 Cadastro e login

A tela inicial de cadastro (Figura 11) deve permitir ao usuário se registrar na plataforma mediante informações de nome de usuário, *e-mail* e senha. Após o cadastro, o usuário poderá logar através da tela de login (Figura 12). As telas são facilmente alternadas entre elas através de um botão.

5.1.2 Dashboard

Após a autenticação na plataforma, o usuário segue para a tela de *dashboard*. Nessa tela são exibidos os gráficos de porcentagem de uso de CPU e RAM em tempo real do *node*

Figura 11 – Tela de cadastro do kubesilent



Fonte: Elaborado pelo autor.

do *Azure Kubernetes Service*. Além disso, são mostrados os *pods* em execução no *cluster*, bem como as informações de métricas de CPU e memória dos *pods* individuais. A Figura 13 mostra a interface da tela de *dashboard* do sistema *kubesilent*.

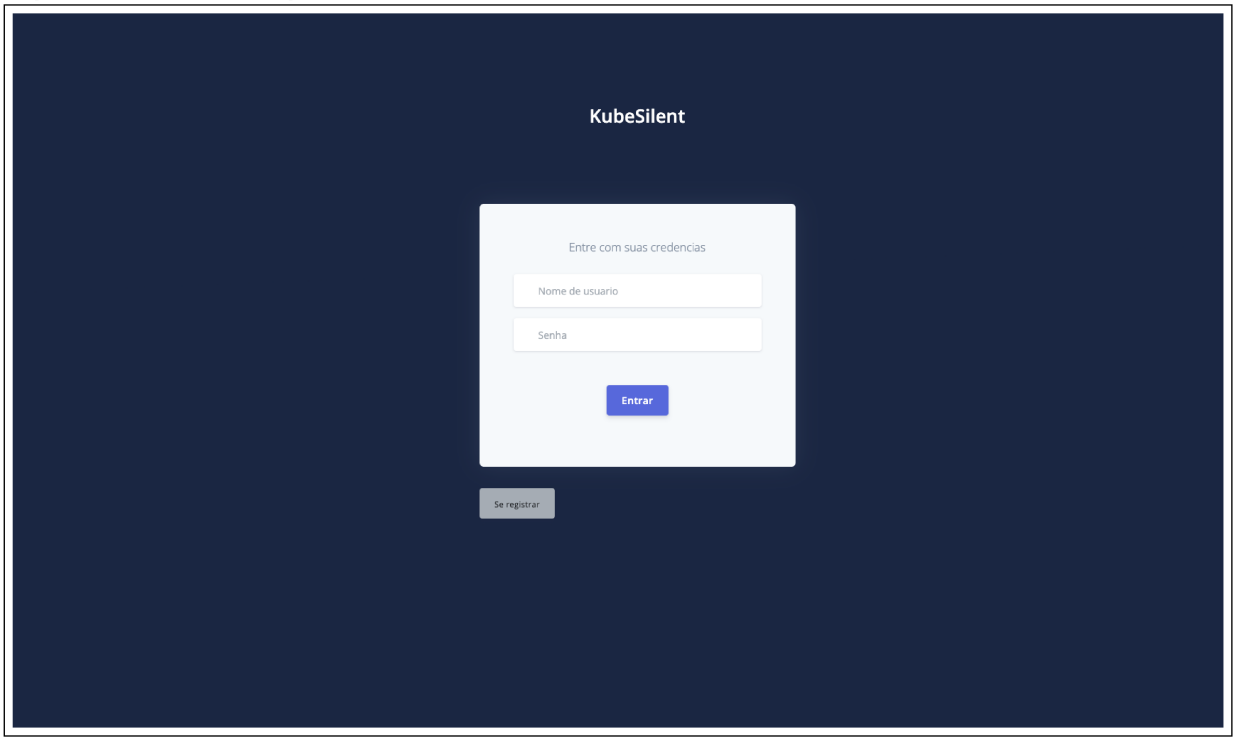
5.1.3 Tela de Nodes

A tela de nodes basicamente exibe as informações dos nodes disponíveis no AKS. Ela mostra o nome do node, a quantidade de *pods* em execução no node, o status e a versão do *kubernetes*.

5.1.4 Tela de pods

Na tela de *pods* do *kubesilente*, é onde o sistema interage com os *containers* em execução no *kubernetes*. Através de uma análise manual, ao clicar em "Ver processos", a aplicação busca no *container*, processos que possam indicar atividades de *cryptojacking*. Esse processo é baseado na busca de cadastros de atividades de *cryptojacking* feitas no banco de dados do sistema. A Figura 15 demostra a tela de *pods* do sistema. Já a Figura 16 demostra o comportamento da aplicação para análise de atividades de *cryptojacking* em um *pod*.

Figura 12 – Tela de login do kubesilent



Fonte: Elaborado pelo autor.

5.1.5 Tela de administração

Na área de administração, é realizado o cadastro e gerenciamento de atividades relacionadas ao *cryptojacking*. Conforme descrito em seções anteriores, o protótipo utiliza nomes de processos conhecidos de *cryptojacking*, caminhos comuns de binários envolvidos nessas atividades e *hashs* dos arquivos desses binários. A interface de administração foi configurada utilizando o painel administrativo nativo do *Django*, como ilustrado na Figura 17.

5.2 Execução dos Testes

Nesta seção serão mostrados o comportamento da aplicação, frente a uma atividade de mineração de criptomoedas.

5.2.1 Simulação do ataque

Como descrito no item 4.3.4, após a instalação do *software* de mineração rodando em *pod*, foi realizado um monitoramento sistemático da aplicação para observar seu comportamento. A aplicação que simula *cryptojacking* foi executada por mais de 5 minutos, enquanto o sistema *Kubesilent* validava, a cada minuto, a média de uso de CPU. Alertas eram enviados a cada minuto

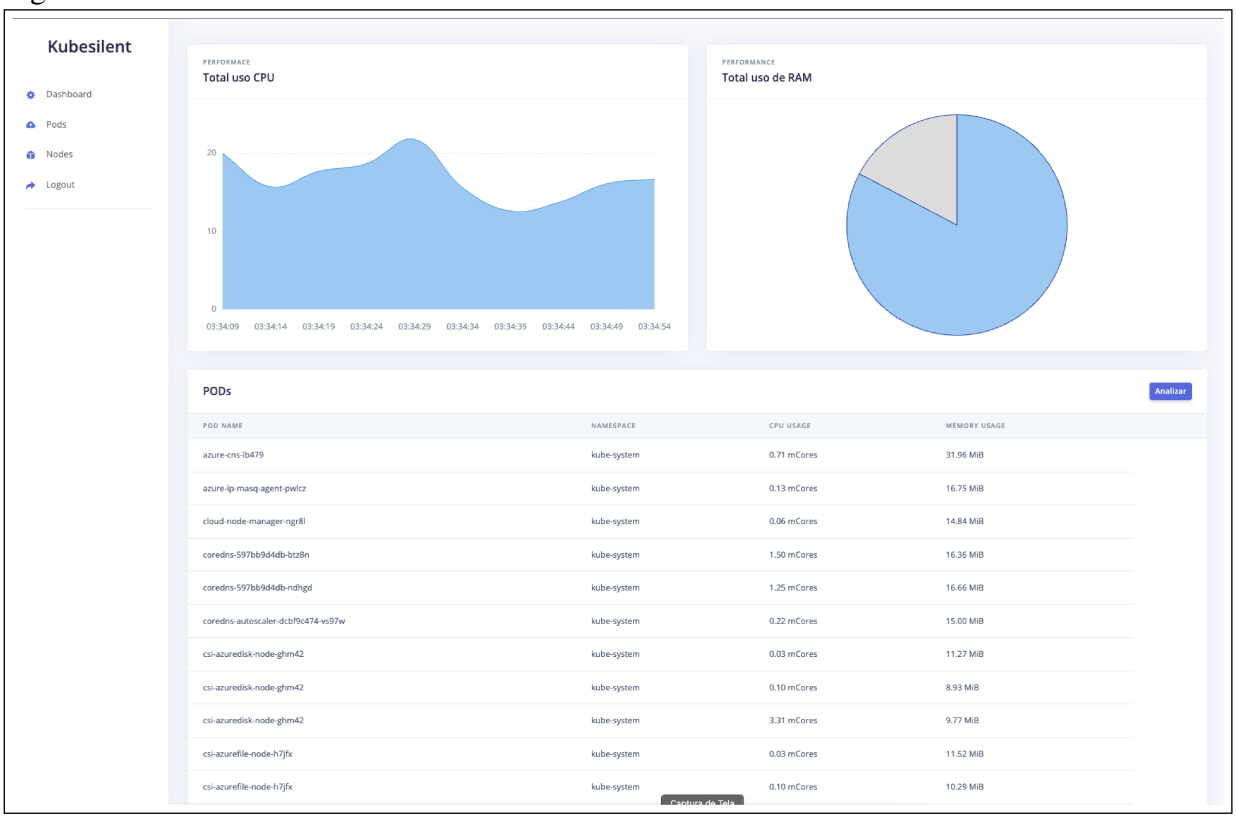
caso o uso de CPU ultrapassasse 80%. As Figuras 18 e 19 demostram os alertas recebido devido ao alto consumo de CPU.

Após os alertas recebidos, foi verificado a plataforma para análise de atividades suspeitas. A Figura 20 mostra o uso de CPU enquanto o software de mineração de criptomoedas está em execução.

Após os alertas, foi avaliado a aplicação para identificar atividades que indicassem a presença de mineração de criptomoedas no AKS. A Figura 21 ilustra *pods* suspeitos rodando, com alto consumo de CPU e memória no AKS.

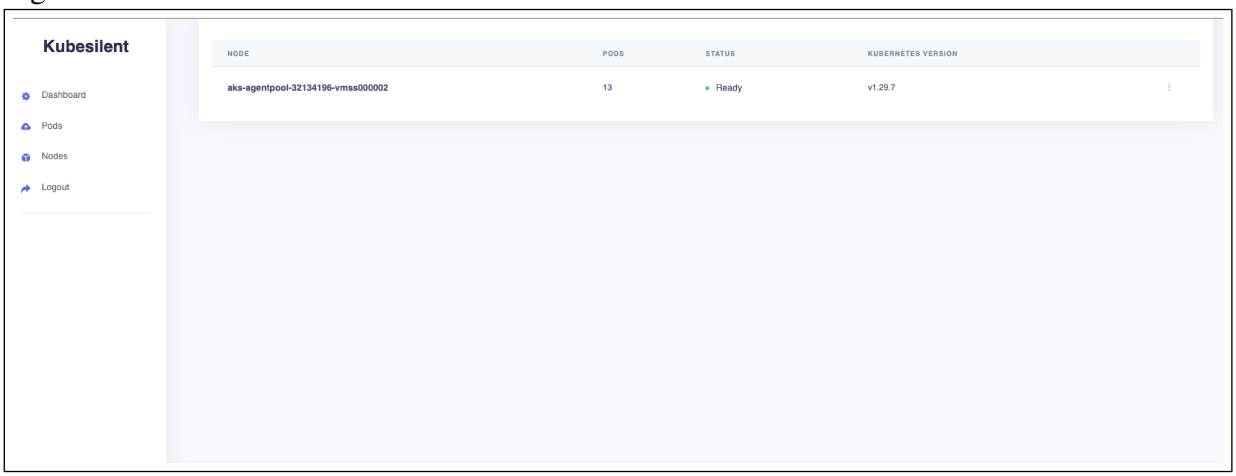
Dessa forma, verificou-se se havia processo que confirmasse a presença de atividades de *cryptojacking* nesses *pods*. A Figura 22, mostra o resultado da validação de processos nos *pods* suspeitos.

Figura 13 – Dashboard do kubesilent



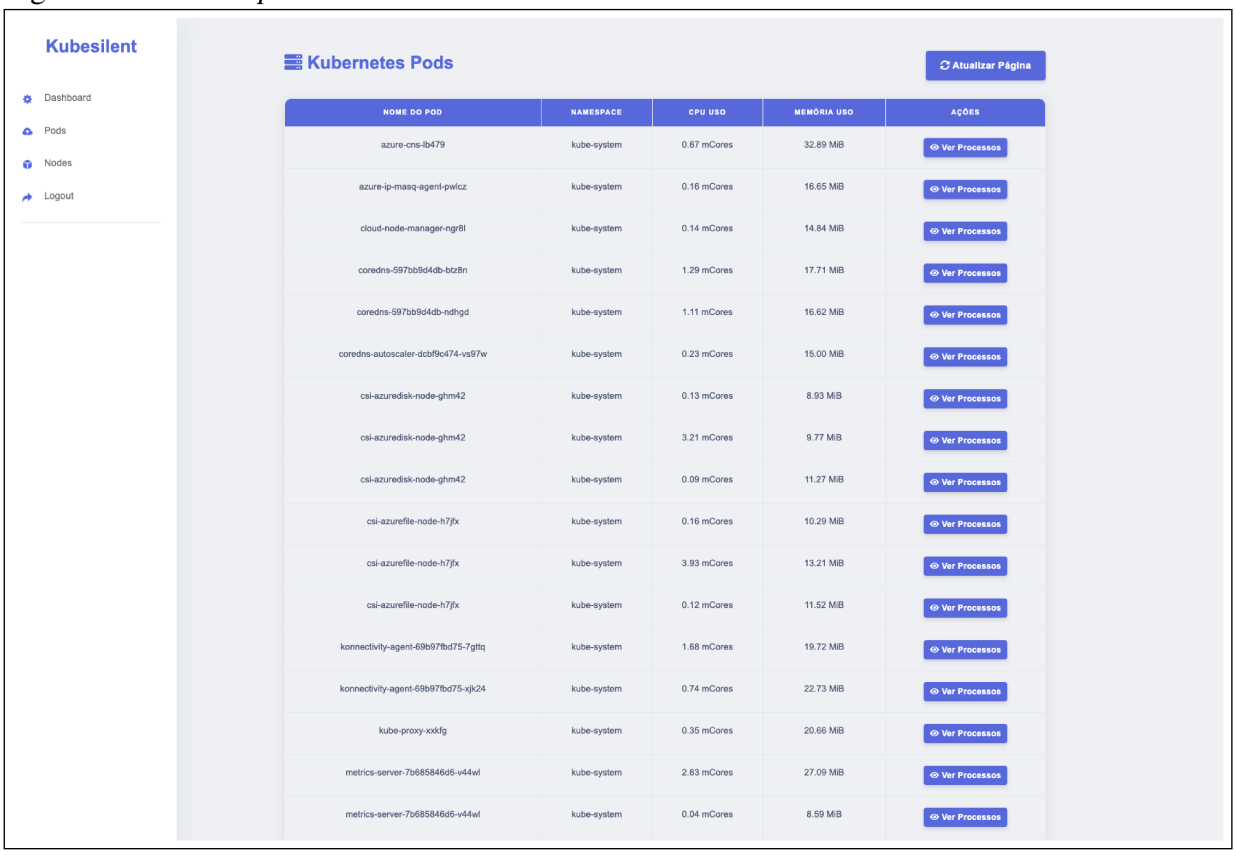
Fonte: Elaborado pelo autor.

Figura 14 – Tela de nodes do kubesilent



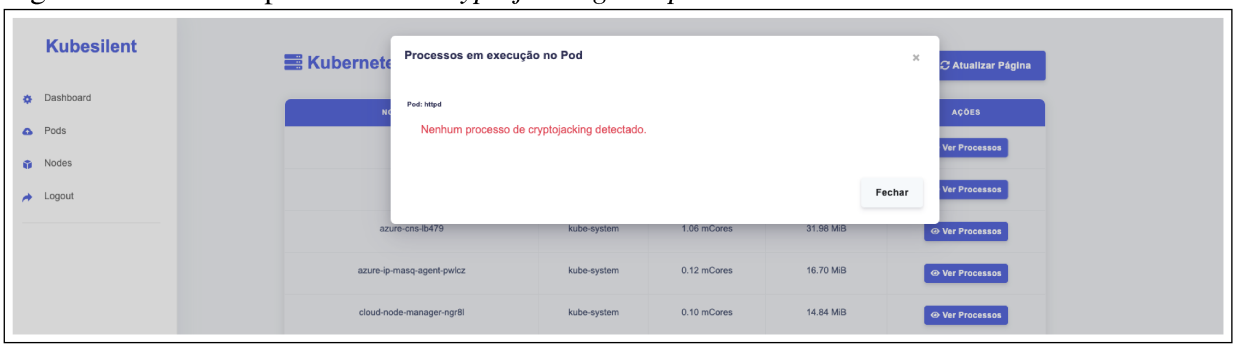
Fonte: Elaborado pelo autor.

Figura 15 – Tela de *pods* do *kubesilent*



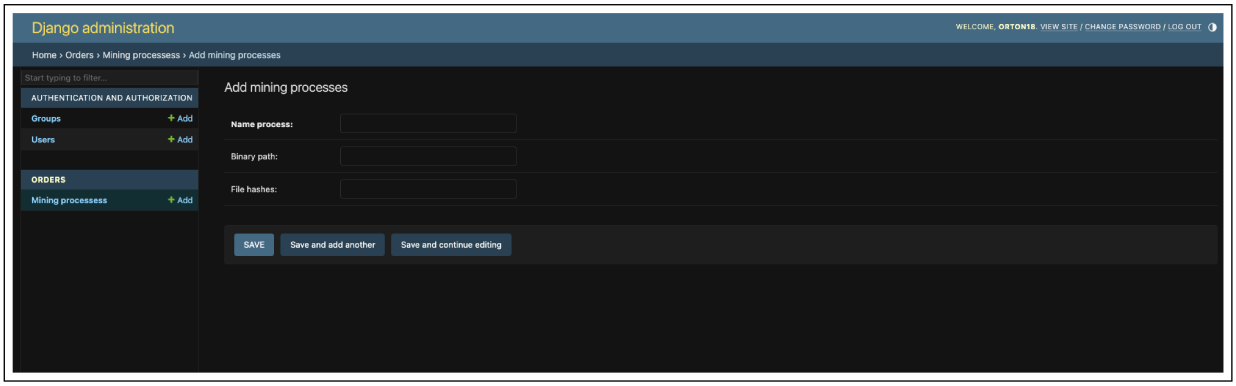
Fonte: Elaborado pelo autor.

Figura 16 – Análise processos de *cryptojacking* nos *pods*



Fonte: Elaborado pelo autor.

Figura 17 – Painel de administração para cadastro de novos indicadores de *cryptojacking*

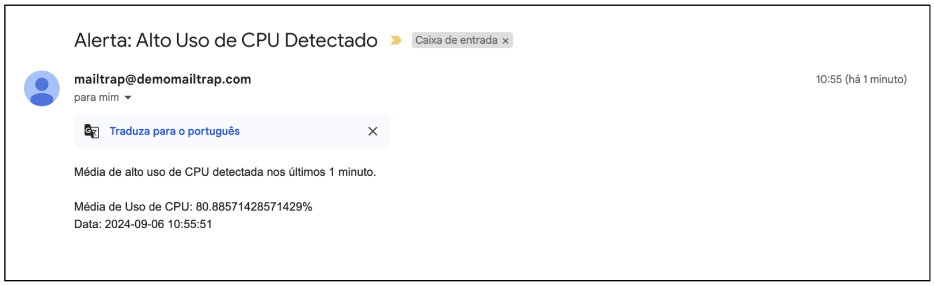


Fonte: Elaborado pelo autor.

Figura 18 – Alertas recebidos

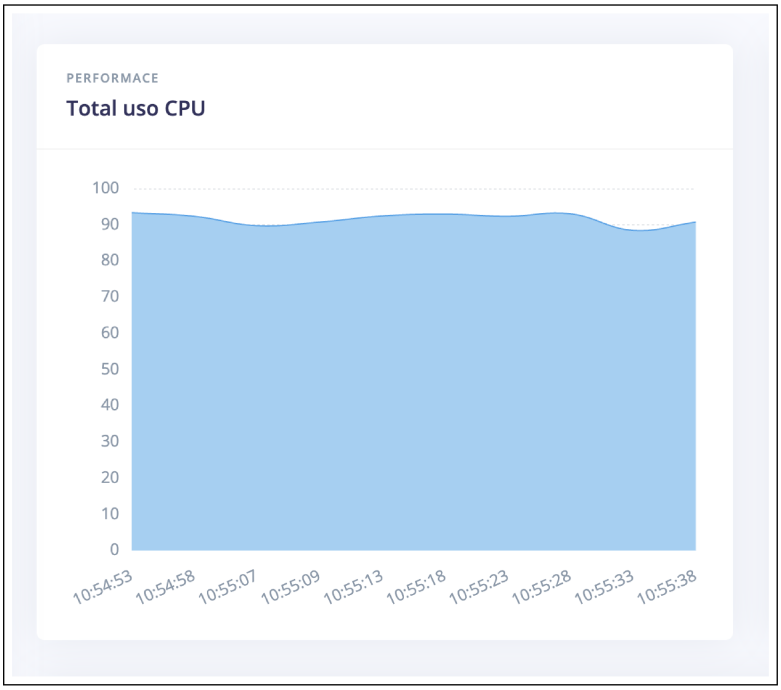
Fonte: Elaborado pelo autor.

Figura 19 – Detalhes do alerta recebido



Fonte: Elaborado pelo autor.

Figura 20 – Consumo de CPU no Node



Fonte: Elaborado pelo autor.

Figura 21 – *Pods* com alto uso de CPU

Fonte: Elaborado pelo autor.

Figura 22 – Análise de processos de *cryptojacking* nos *pods* suspeitos

Fonte: Elaborado pelo autor.

6 CONCLUSÃO E TRABALHOS FUTUROS

O presente trabalho se propôs a criar um protótipo de uma solução de monitoramento para detectar atividades de *cryptojakcing* no serviço do *azure kubernets service*. Os testes foram realizados com auxilio de softwares *Open-source* para simular um ataque de *cryptojacking*. De maneira geral, a ferramenta permite o monitoramento em tempo real das métricas de CPU e memória dos *nodes* e *pods*, além de oferecer uma solução prática para a detecção manual de processos maliciosos associados ao *cryptojacking*.

Ao longo do processo de criação da infraestrutura, desenvolvimento da solução, integração com o ambiente de testes e validação da plataforma, foram enfrentados e superados diversos desafios técnicos e operacionais. Os testes evidenciaram que o sistema pode ser uma boa ferramenta na identificação de atividades de mineração de criptomoedas. É importante enfatizar que o projeto se encontra em um protótipo inicial e que ainda tem funcionalidades a serem adicionadas e melhoradas na plataforma.

Entre as possíveis melhorias e direções para trabalhos futuros, podemos destaca-ser:

- Melhora a interface para a visualização de *cluster* com mais de um *Node*;
- Fazer testes e para validar o comportamento da aplicação, frente a falsos positivos;
- Automação da detecção de processos de *cryptojacking* e a implementação de alertas em tempo real;
- Uso de aprendizado de máquina para aprimorar a identificação de anomalias no uso de CPU, evitando falsos positivos, e para a detecção de atividades de mineração de criptomoedas;
- Uso de inspeção de trafego de redes, como uma forma mais eficaz de detectar *cryptojacking*.
- Expansão da solução para atuar não apenas com *cryptojacking*, mas como uma solução de segurança para *kubernetes*.

REFERÊNCIAS

ANDERSSON, J. C. **Learning Microsoft Azure**. [S.l.]: "O'Reilly Media, Inc.", 2023.

ARIFFIN, M. A. M.; DARUS, M. Y.; TAIB, A. M.; OSMAN, R.; MAT, C. M. A. C. Detecting illicit cryptocurrency mining activity in cloud computing platform. **Journal of Positive School Psychology**, v. 6, n. 3, p. 8611–8622, 2022.

BASAVARAJAPPA, R. N. Secure kubernetes resources from cpu based cryptojacking. **Global journal of Business and Integral Security**, 2016.

COLLIER, M.; SHAHAN, R. **Microsoft azure essentials-fundamentals of azure**. [S.l.]: Microsoft Press, 2015.

DAEMONSET. **Kubernetes**. 2024. Disponível em: <https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/>, . Acesso em: 14 ago. 2024.

FIGUEIREDO, E. Requisitos funcionais e requisitos não funcionais. **Icex, Dcc/Ufmg**, 2011.

GRAP, M. A. B. **CrowdStrike Discovers First-Ever Dero Cryptojacking Campaign Targeting Kubernetes**. 2023. Disponível em: <https://www.crowdstrike.com/blog/crowdstrike-discovers-first-ever-dero-cryptojacking-campaign-targeting-kubernetes/>, . Acesso em: 14 ago. 2024.

JAYASINGHE, K.; PORAVI, G. A survey of attack instances of cryptojacking targeting cloud infrastructure. In: **Proceedings of the 2020 2nd Asia pacific information technology conference**. [S.l.: s.n.], 2020. p. 100–107.

KUBERNETES. **Kubernetes API Concepts**. 2024. Disponível em: <https://kubernetes.io/docs/reference/using-api/api-concepts/>, . Acesso em: 26 ago. 2024.

KUBERNETES, T. Kubernetes. **Kubernetes. Retrieved May**, v. 24, p. 2019, 2019.

LANE, M.; SHRESTHA, A.; ALI, O. Managing the risks of data security and privacy in the cloud: a shared responsibility between the cloud service provider and the client organisation. **Bright Internet Global Summit 2017**, University of Southern Queensland, 2017.

MELL, P.; GRANCE, T. *et al.* The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National . . . , 2011.

MICROSOFT. **Cryptojacking: Understanding and defending against cloud compute resource abuse**. 2023. Disponível em: <https://www.microsoft.com/en-us/security/blog/2023/07/25/cryptojacking-understanding-and-defending-against-cloud-compute-resource-abuse/>, . Acesso em: 14 ago. 2024.

PASSOS, I. B.; SOUZA, G. P. de; LAZARIN, N. M. Identificação de cryptojacking: Uma análise de ferramentas. **Anais do Computer on the Beach**, v. 15, p. 312–314, 2024.

POULTON, N. **The kubernetes book**. [S.l.]: NIGEL POULTON LTD, 2023.

SANTIAGO, C. P.; VERAS, N. L.; ARAGÃO, A. P. de; CARVALHO, D. A.; AMARAL, L. A. Desenvolvimento de sistemas web orientado a reuso com python, django e bootstrap. **Sociedade Brasileira de Computação**, 2020.

SCHAFFERERIN TYNEVI, h. c. h. **O que é o AKS (Serviço de Kubernetes do Azure)?** 2024. Disponível em: <https://learn.microsoft.com/pt-br/azure/aks/what-is-aks>, . Acesso em: 02 ago. 2024.

TEKINER, E.; ACAR, A.; ULUAGAC, A. S.; KIRDA, E.; SELCUK, A. A. Sok: cryptojacking malware. In: IEEE. **2021 IEEE European Symposium on Security and Privacy (EuroS&P)**. [S.l.], 2021. p. 120–139.

VERAS, M. **Cloud Computing: nova arquitetura da TI**. [S.l.]: Brasport, 2012.

WRIGHT DENNIS SMITH, K. J. M. A. B. A. G. S. B. D. **Quadrante Mágico para Serviços Estratégicos da Plataforma de Nuvem**. 2023. Disponível em: <https://www.gartner.com/technology/media-products/reprints/amazon/1-2FTPMD93-PTB.html?trk=44f67619-4f3b-42e8-93b9-32ad8a123845&sc_channel=el>. Acesso em: 02 ago. 2024.