Ignite 2023

Wordpress Stack;

Tsuyoshi
January 27, 2025

Contents

Contents

Versão	Data	Responsável	Descrição
V0.1	19-Jun-2024	Henrique Yara	Estrutura da documentação
V0.2	26-Jun-2024	Henrique Yara	Terraform e ambiente
V0.3	08-Jul-2024	Henrique Yara	Ansible
V0.4	26-Jul-2024	Henrique Yara	Kubernetes
V1.0	08-August-2024	Henrique Yara	Documentação v1.0

ESCRITA

Palavras em negrito: Nome de ferramentas e comandos
Palavras em itálico: Palavras em inglês
Palavras em colorido: Nome de arquivos ou variáveis

REQUERIMENTOS

2.1 Desafio (Individual)

WordPress rodando como *microservices* em **Kubernetes**, o provisionamento deve ser feito com **Terraform** e as configurações com **Ansible** e **Helm**, a arquitetura e implementação deve ser efetuada com boas práticas de cada provedor, a aplicação deve escalar horizontalmente.

2.2 Objetivo

Avaliar a proatividade, criatividade, resiliência e entrega de cada Ignite.

2.2.1 Ferramentas

- Ansible
- Helm
- Terraform
- AWS, Azure, GCP
- Kubernetes
- Wordpress

2.3 Entrega

- Infraestrutura
- Código IaC
- Aplicação disponível
- Aplicação Escalável
- Documentação

2.4 Desafio (grupo)

Gerenciamento de *logs* com **ELK** centralizado, o *log* de todos os ambientes devem ser gerenciados por uma única plataforma, ingestão dos *logs*, transformação e visualização.

2.5 Entregas adicionais

A forma de apresentação do desafio e entregas adicionais foi deixada em aberto.

2.6 Prazo

Data de entrega: 09/08/2024

DEPLOY

3.1 Requisitos

3.1.1 Sistema operacional

O ambiente foi testado dentro do **WSL (Windows Subsystem for Linux)** na versão 2.2.4.0.

O *WSL* está usando a distribuição **Ubuntu** versão 22.04.4 LTS com as seguintes configurações:

```
uname -a
# Linux henriqueatos 5.15.153.1-microsoft-standard-WSL2 #1
    SMP Fri Mar 29 23:14:13 UTC 2024 x86_64 x86_64 x86_64 GNU/
    Linux
```

3.1.2 Aplicações

É necessário ter as seguintes aplicações instaladas:

- terraform:1.9.0-1
- python3-pip:22.0.2+dfsg-1ubuntu0.4
- ansible:10.1.0
- ansible-core:2.17.1
- make:4.3-4.1build1 (opcional)

Módulos do **ansible-galaxy**:

- community.mysql:v3.9.0
- kubernetes.core:v3.2.0

3.1.3 Instalar requisitos

Para instalar os requisitos use o comando abaixo:

```
sudo apt install terraform=1.9.0-1 python3-pip=22.0.2+dfsg-1
    ubuntu0.4 make=4.3-4.1build1
pip install ansible==10.1.0 ansible-core==2.17.1 --break-
    system-packages
ansible-galaxy collection install community.mysql:3.9.0
    kubernetes.core:3.2.0
```

Caso tenha problemas para instalar terraform siga o tutorial oficial da Hashicorp.
Lembre se de adicionar os binários instalados pelo **pip** no PATH:

```
PATH=$PATH:$HOME/.local/bin/
```

3.1.4 Configuração

Primeiro, configure seu acesso na nuvem **AWS** colocando suas credenciais no arquivo ∼/.aws/credentials ou usando o comando abaixo:

```
aws configure
```

No código terraform existe duas credenciais. O perfil workload definido no arquivo terraform/main.tf, e o perfil route53 definido no arquivo validate_terraform/main.tf.
O perfil route53 possui um *DNS* pré-configurado.
O arquivo development.tfvars possui as variáveis usadas pelo **terraform**.
As variáveis que precisam ser alteradas são:

- bastion_config.source_cidr: IP da sua máquina que vai acessar a instância bastion.
- bastion_config.public_key_path: Caminho da chave pública usada no bastion.
- bastion_config.private_key_path: Caminho da chave privada para autenticação.

O terraform está usando um **bucket S3** como *backend*. Garanta que seu usuário também tenha acesso ao bucket.
Verifique também se o segredo do **stormforge** existe na **AWS**, caso não exista crie esse segredo com o nome **stormforge-credentials**.

3.2 Instruções (sem make)

Entre na pasta terraform.
Inicialize o terraform:

```
terraform init
```

Para criar apenas o certificado execute:

```
terraform apply --auto-approve -var-file='development.tfvars'
    -target=aws_acm_certificate.website_cert
```

Obtenha os valores para validar o domínio:

```
terraform output cert_validation
```

Faça a validação do certificado.
É importante que a validação do certificados seja feita antes do cluster ser criado, pois o **Application Load Balancer** vai precisar o certificado.
Execute o resto do código terraform:

```
terraform apply --auto-approve -var-file='development.tfvars'
    -parallelism=10
```

Faça os apontamentos do DNS para o load balancer.

3.3 Instruções (make)

Observação: Se for utilizar o **make** é necessário ter a conta route53 configurada.
Entre na pasta terraform.
Para execute no **WSL** o seguinte comando:

```
make all
```

3.4 Makefile

O arquivo Makefile tem as opções:

- all: Executa o **setup**, **start** e **records**.
- setup: Inicializa o **terraform**.
- start: Aplica as configurações dos arquivos **terraform**.
- records: Cria os apontamentos *DNS*.
- stop: Destrói o ambiente que foi criado pelo **terraform**.
- ansible: Força a execução do código ansible dentro do **terraform**.
- ssh: Cria uma conexão **ssh** com a instância **bastion**.
- clean: Apaga os arquivos principais do **terraform**.

3.5 Acesso do ambiente

Durante a execução do **terraform** o terminal irá mostrar as senhas para o grafana, wordpress e um token temporário para acessar o kubernetes-dashboard:

```
"msg": [
        "--- Wordpress ---",
        "User: eks_wp_admin",
        "Password: xxxxxxx",
        "--- Grafana ---",
        "User: grafana_admin",
        "Password: xxxxxxx",
        "--- Kubernetes Dashboard ---",
        "Token: XXX"
]
```

Também é possível acessar as senhas pelo **AWS Secret Manager**.

3.6 Retorno esperado

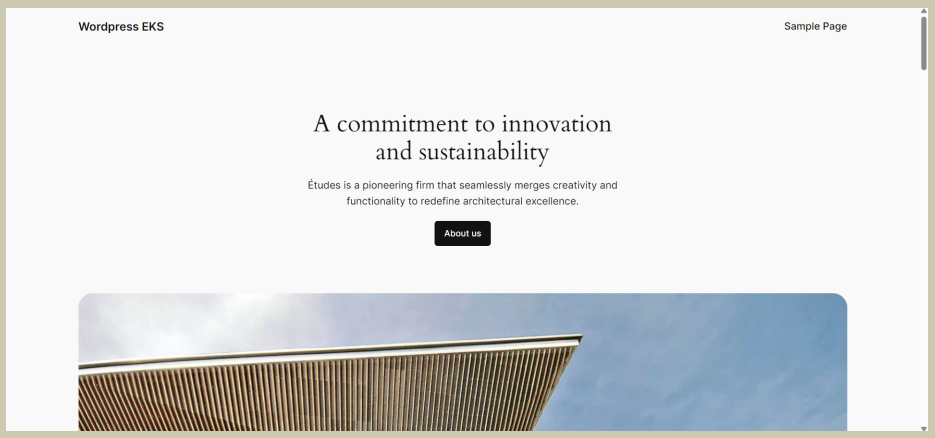

Figure 1: Retorno de `https://wordpress.htsuyoshiy.online`

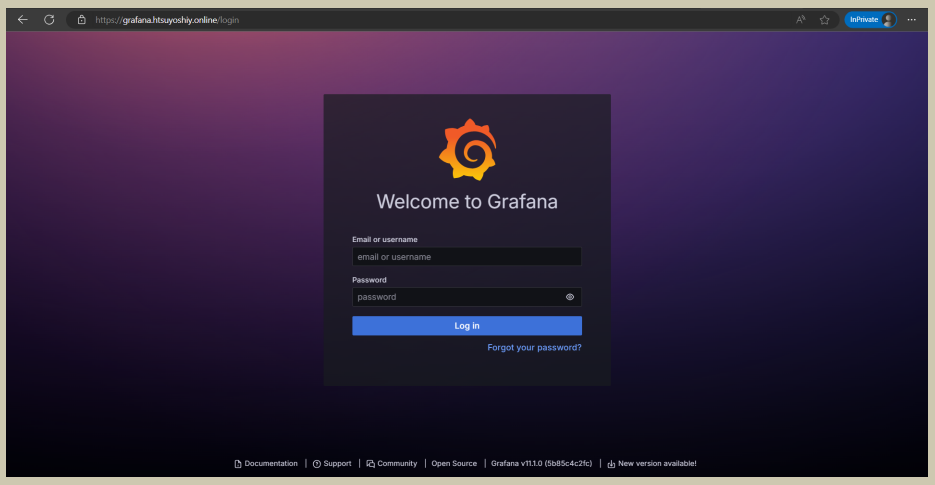

Figure 2: Retorno de `https://grafana.htsuyoshiy.online`

3.6 RETORNO ESPERADO

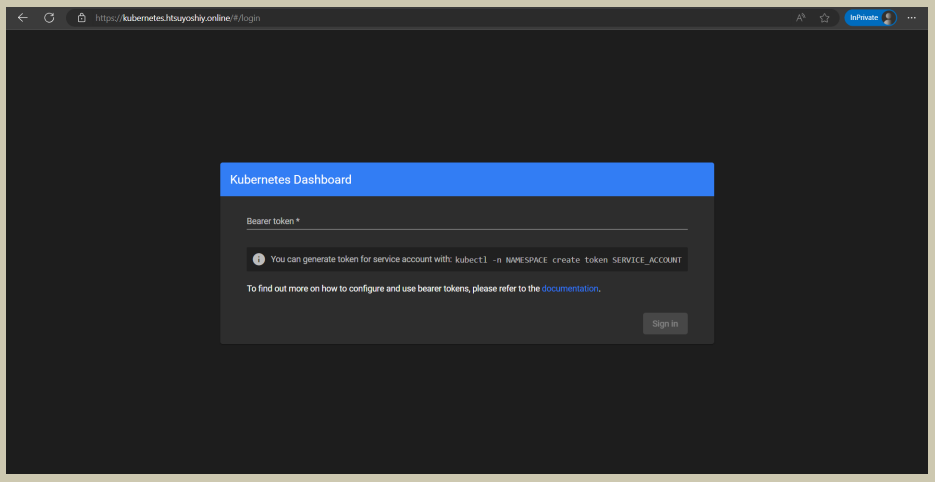

Figure 3: Retorno de `https://kubernetes.htsuyoshiy.online`

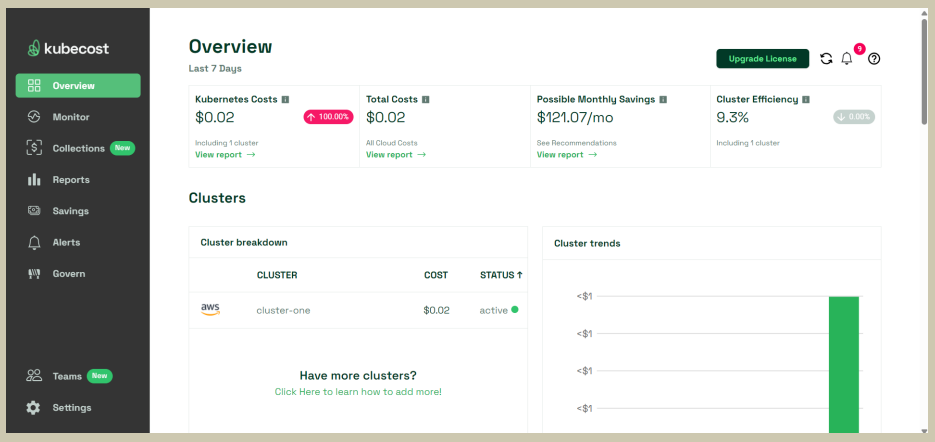

Figure 4: Retorno de `https://kubecost.htsuyoshiy.online`

3.7 Destruir o ambiente

Para destruir o ambiente faça login no **bastion** e delete todos os componentes criados pelo **ansible** no cluster.

Caso o ambiente seja destruído sem todas as aplicações do cluster tiverem sido destruídas, alguns componentes da **aws** podem ficar sobrando.

- O **Application Load Balancer** gerenciado pelo **ingress controller** pode impedir o ambiente de ser destruído pelo terraform.
- Os **Volumes EBS** gerenciado pelo **ebs controller** podem ficar sobrando na conta aws.
- As instâncias **Karpenter EC2** gerenciado pelo **karpenter**
- Existem **Secrets** que foram criados manualmente como o **stormforge**.

Depois execute:

```
terraform destroy --auto-approve -var-file='development.tfvars'
```

ARQUITETURA

4.1 Arquitetura

Arquitetura desenvolvida na nuvem **AWS**:

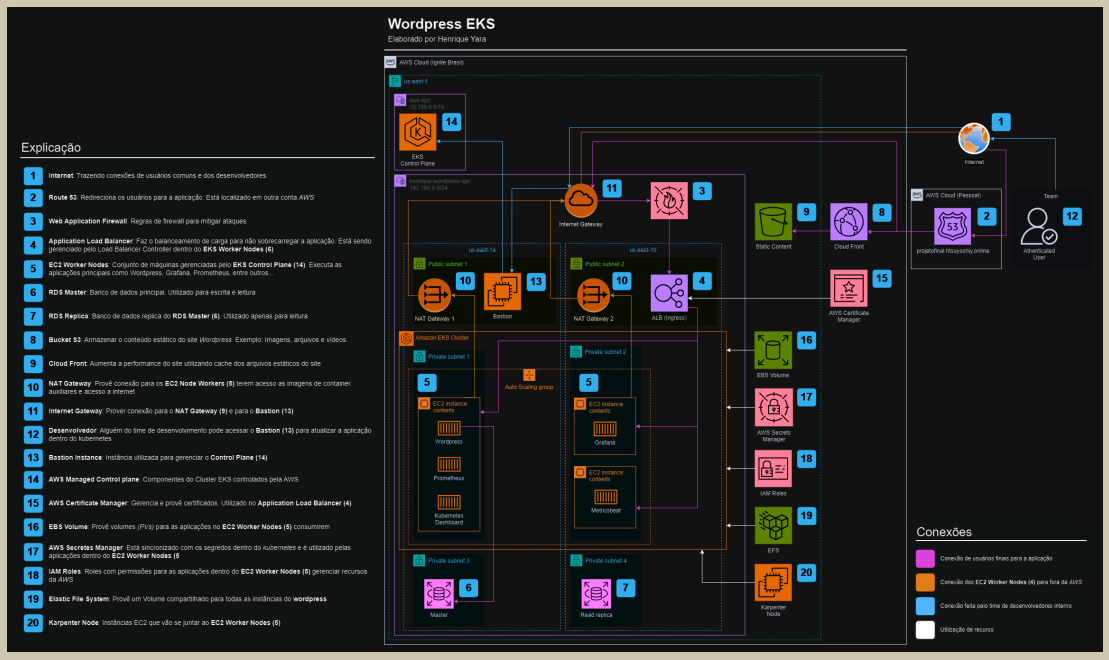

Figure 5: Arquitetura completa v1.4

4.1 ARQUITETURA

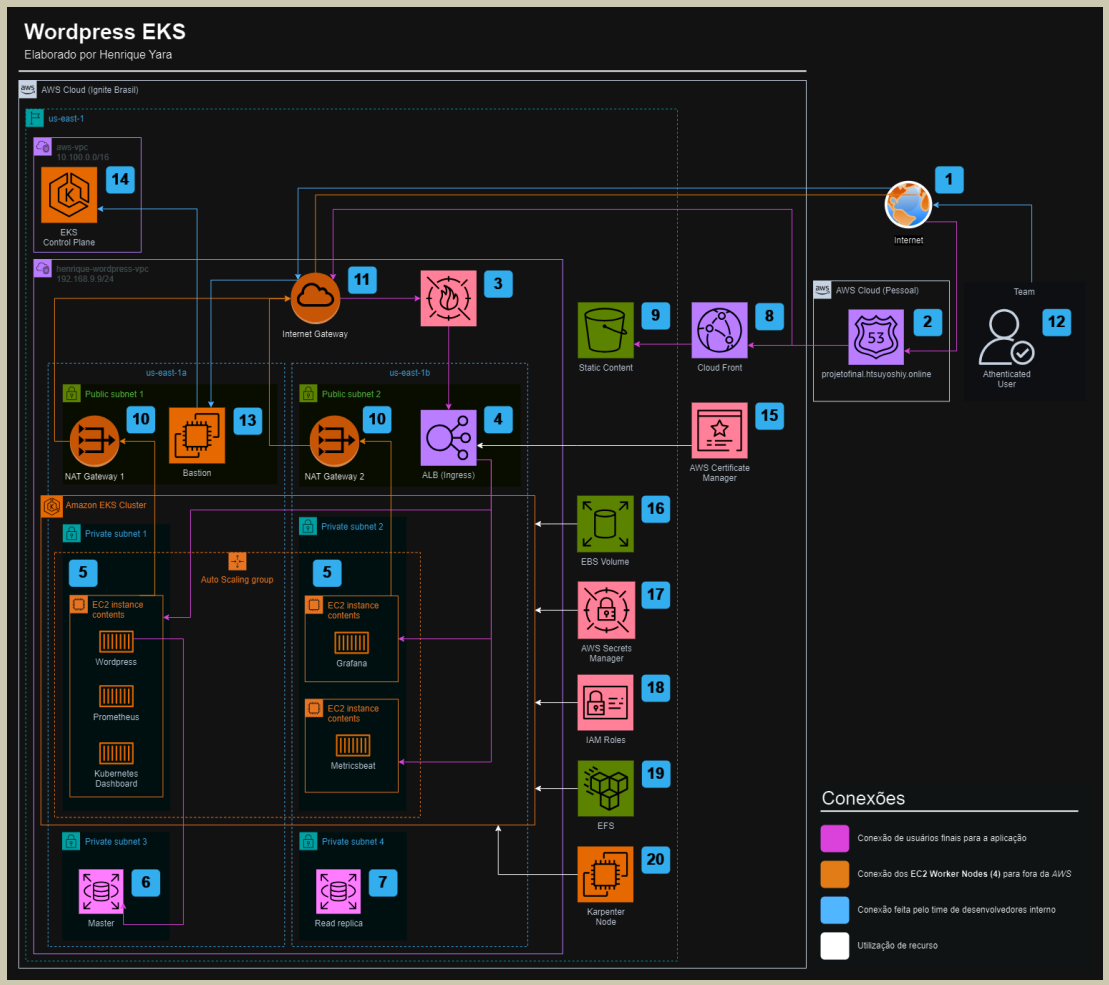

Figure 6: Conexões - Arquitetura v1.4

Figure 7: Explicação - Arquitetura v1.4

AWS

5.1 Usuários

5.1.1 workload

A conta de workload da AWS é gerenciada pelo time HCI da Atos Brasil
Conta: XXX
Grupo: XXX
Permissão: Administrador
Região: us-east-2

5.1.2 route53

A conta de route53 da AWS é externa.
Conta: route53
Grupo: add-record-route-53
Permissão: Modificar registros do **Route 53**
Região: us-east-1

5.2 Recursos

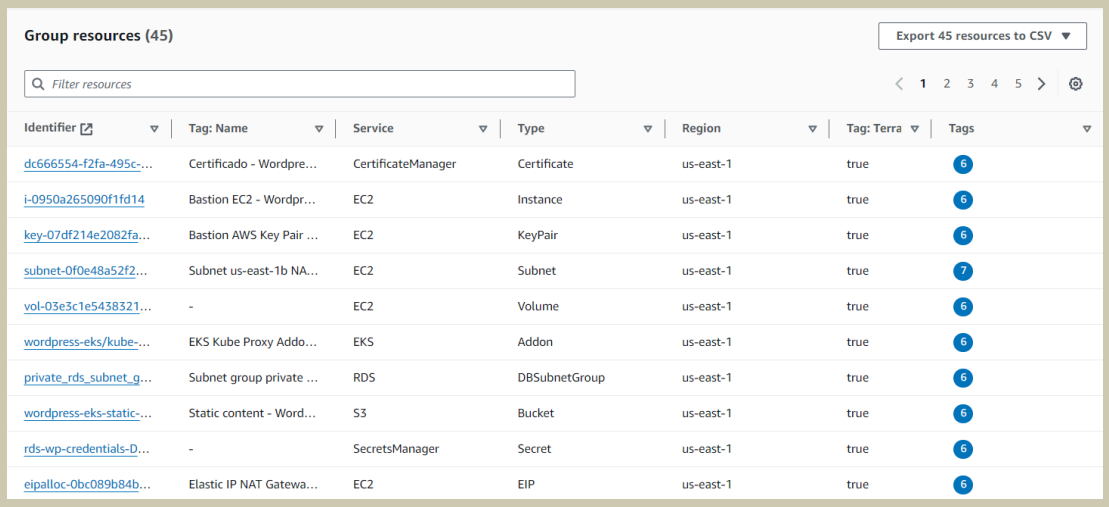

Figure 8: Recursos criados na AWS

5.2.1 Tags

Tag	Valor	Explicação
Application	Wordpress EKS	Nome da aplicação
Terraform	true	Indica se o objeto foi criado pelo terraform
Environment	Development	Ambiente da aplicação
Owned by	Henrique Yara	Responsável por criar o objeto

Table 1: Tags padrão: Aplicado em TODOS os recursos criados

Tag	Valor	Explicação
Name	EKS Cluster - EKS Wordpress	Nome do recurso
Type	EKS	Categoria do recurso

Table 2: Tags específicas: separa cada tipo de serviço e suas dependências em grupos

5.3 Roles

5.3.1 CustomerServiceRoleForBastion

Permissão para acessar segredos do **Secrets Manager**, acessar informações do **EKS**.

- Policies:
  - BastionPolicy (Customer Managed)

5.3.2 CustomerServiceRoleForEKSCluster

Permissões de acesso para o **Cluster EKS**.

- Policies:
  - AmazonEKSClusterPolicy (AWS Managed)
  - AmazonEKSVPCResourceController (AWS Managed)

5.3.3 CustomerServiceRoleForEKSNodeGroup

Permissões de acesso para os **Node Workers**.

- Policies:
  - ☐ AmazonEKSWorkerNodePolicy (AWS Managed)
  - ☐ AmazonEKS_CNI_Policy (AWS Managed)
  - ☐ AmazonEC2ContainerRegistryReadOnl (AWS Managed)

5.3.4 CustomerServiceRoleForKarpenterController

Permissão para alocar nós para o *cluster*.

- Trust policy: Federation
- Service account: karpenter:karpenter
- Policies:
  - ☐ AWSKarpenterControllerPolicy (Customer Managed)

5.3.5 CustomerServiceRoleForKarpenterNode

Permissão para alocar nós para o *cluster*.

- Trust policy: Federation
- Service account: karpenter:karpenter
- Policies:
  - ☐ AmazonEKSWorkerNodePolicy (AWS Managed)
  - ☐ AmazonEKS_CNI_Policy (AWS Managed)
  - ☐ AmazonEC2ContainerRegistryReadOnl (AWS Managed)
  - ☐ AmazonSSMManagedInstanceCore (AWS Managed)

5.3.6 CustomerServiceRoleForEKSEBSCSIDriver

Permissão para criar volumes **EBS** para **StatefulSets**.

- Trust policy: Federation
- Service account: kube-system:ebs-csi-controller–sa
- Policies:
  - AmazonEBSCSIDriverPolicy (AWS Managed)

5.3.7 CustomerServiceRoleForEKSEFSCSIDriver

Permissão para criar **EFS** para as instâncias do **wordpress**.

- Trust policy: Federation
- Service account: kube-system:efs-csi-controller–sa
- Policies:
  - AmazonEFSCSIDriverPolicy (AWS Managed)

5.3.8 CustomerServiceRoleForLoadBalancerController

Permissão para gerenciar **ALB** e criar *endpoints* do **Ingress**.

- Trust policy: Federation
- Service account: kube-system:alb-ingress-controller-sa
- Policies:
  - AWSLoadBalancerControllerIAMPolicy (Customer Managed)

5.3.9 CustomerServiceRoleForGrafanaSecretsAccess

Acesso para todos os segredos do **Secret Manager**.

- Trust policy: Federation
- Service account: kube-system:secrets-csi-sa
- Policies:
  - SecretsCSIControllerIAMPolicy (Customer Managed)

5.3.10 CustomerServiceRoleForGrafanaSecretsAccess

Acesso para os segredos do **Grafana** no **Secret Manager**.

- Trust policy: Federation
- Service account: monitoring:prometheus-grafana
- Policies:
  - AWSSecretsAccessGrafana (Customer Managed)

5.3.11 CustomerServiceRoleForWordpressOffloadMedia

Acesso para o bucket de conteúdo estático do wordpress e segredos armazenados no Secret Manager.

- Trust policy: Federation
- Service account: wordpress:wp-offload-media-sa
- Policies:
  - WordpressS3AccessPolicy (Customer Managed)
  - SecretsAccessRDSWp (Customer Managed)

5.4 Secrets

Segredos:

- rds (Gerenciado pela AWS): Senha do banco **RDS**
- rds-wp-credentials: Usuário e senha do **wordpress** para acessar o **RDS**
- rds-grafana-credentials: Usuário e senha do **grafana** para acessar o **RDS**
- wp-credentials: Usuário e senha do **wordpress**
- grafana-credentials: Usuário e senha do **grafana**
- stormforge-credentials: ID e segredo do **stormforge**

5.5 Rede

Subnet	CIDR	Acesso	Explicação
Control Plane	10.100.0.0/16	Privado	CIDR do Control Plane
VPC	192.168.0.0/16	Privado	CIDR da VPC
EKS Subnet 1	192.168.0.0/18	Privado	EKS Worker Nodes
EKS Subnet 2	192.168.64.0/18	Privado	EKS Worker Nodes
RDS Subnet 1	192.168.128.0/24	Privado	Banco de dados principal
RDS Subnet 2	192.168.129.0/24	Privado	Réplica do banco de dados
Subnet Pública 1	192.168.130.0/24	Público	Bastion e NAT Gateway
Subnet Pública 2	192.168.131.0/24	Público	NAT Gateway

Table 3: Separação de rede

5.5.1 Security Groups

Bastion Security Group					
Ingress			Egress		
Source	Ports	Action	Destination	Ports	Action
IP do desenvolvedor	22	Allow	0.0.0.0/0	all	allow

Table 4: Bastion Security Group

Worker Node Access EKS Control Plane Security Group					
Ingress			Egress		
Source	Ports	Action	Destination	Ports	Action
192.168.0.0/18	443	Allow	0.0.0.0/0	all	allow
192.168.64.0/18	443	Allow	-	-	-

Table 5: EKS Worker Node Security Group

Bastion Access EKS Control Plane Security Group					
Ingress			Egress		
Source	Ports	Action	Destination	Ports	Action
bastion_sg	443	Allow	0.0.0.0/0	all	allow

Table 6: EKS Bastion Security Group

RDS Security Group					
Ingress			Egress		
Source	Ports	Action	Destination	Ports	Action
a	3306	Allow	0.0.0.0/0	all	allow

Table 7: RDS Security Group

EFS Security Group					
Ingress			Egress		
Source	Ports	Action	Destination	Ports	Action
a	2049	Allow	0.0.0.0/0	all	allow

Table 8: EFS Security Group

5.6 Certificate Manager

Certificado com os seguintes domínios:

- htsuyoshiy.online
- kubernetes.htsuyoshiy.online
- kubecost.htsuyoshiy.online
- wordpress.htsuyoshiy.online
- grafana.htsuyoshiy.online

5.7 Elastic Kubernetes Service

Versão: v1.30.0
Add-ons:

- Amazon VPC CNI:v1.16.0-eksbuild.1
- kube-proxy:v1.29.0-eksbuild.1
- CoreDNS:v1.11.1-eksbuild.9
- Amazon EBS CSI Driver:v1.32.0-eksbuild.1
- Amazon EFS CSI Driver:v2.0.5-eksbuild.1

5.8 EKS - Node Group

Tipo de instância: t3.medium
Tamanho do disco: 20 Gb
Tipo de nó: ON_DEMAND
Número desejado de nós: 3
Mínimo de nós: 3
Máximo de nós: 3
Máximo indisponível: 1
Responsável por executar o **karpenter**, **metrics-server** e outros **controllers**

5.9 Elastic File System

Nome: EFS - Wordpress EKS

5.10 Cloud Front

Responsável por enviar o conteúdo estático armazenado no **S3**

5.11 S3

5.11.1 Conteúdo estático - Wordpress

Prefixo: wordpress_eks_static_content_

5.11.2 terraform-backend

Nome: henrique-s3-terraform-backend

5.12 SQS

Responsável por interceptar eventos de desligamento de instâncias **spot** provisionadas pelo **karpenter**.

5.13 Web Application Firewall

O **WAF** foi utilizado para mitigar ataques na aplicação. Ele fica na linha de frente do **Load Balancer**.

Nome do **WAF WACL**: wafv2-web-acl

Regras:

- AWSManagedRulesCommonRuleSet
- AWSManagedRulesLinuxRuleSet
- AWSManagedRulesAmazonIpReputationList
- AWSManagedRulesAnonymousIpList
- AWSManagedRulesKnownBadInputsRuleSet
- AWSManagedRulesUnixRuleSet
- AWSManagedRulesWindowsRuleSet

BASTION

6.1 Especificações

Sistema Operacional: Ubuntu 24.04
Plataforma: Linux/UNIX
Arquitetura: x86_64
Virtualização: HVM
Armazenamento: ssd-gp3
AMI-ID: ami-04a81a99f5ec58529

6.2 Ansible

O **ansible** automatiza a configuração do banco de dados e do **kubernetes** por meio de uma instância **EC2**.

As automatizações feitas pelo ansible instalam dos seguintes programas na instância **EC2**:

- Dependências de **playbooks** do **ansible**:
  - unzip:6.0-28ubuntu
  - python3-pip:24.0+dfsg-1ubuntu1
  - python3-kubernetes:22.6.0-2
  - python3-yaml:6.0.1-2build2
  - python3-jsonpatch:1.32-3
- Binários para interagir com **kubernetes** e **AWS**:
  - awscli:latest
  - kubectl:v1.30.0
  - helm:v3.15.2
  - eksctl:v0.183.0

RELATIONAL DATABASE SERVICE

Engine: mariadb
Engine version: 10.11.6
Instance class: db.t3.micro
Port: 3306
Allocated storage: 5
MaX allocated storage: 10
Storage encrypted: false
Manage master user password: true
Publicly accessible: false
Multi az: false
Database: wordpress

7.1 Databases

Database:

- wordpress
- defectdojo

7.2 Users

7.2.1 wordpress_user

wordpress_user@% (Deactivated)

- Privillege: *.*:ALL

7.2.2 wp_user

wp_user@192.168.0.0/255.255.192.0
wp_user@192.168.64.0/255.255.192.0

- Privillege: wordpress.*:ALL

7.2.3 grafana_user

grafana_user@192.168.0.0/255.255.192.0
grafana_user@192.168.64.0/255.255.192.0

- Privillege:
  - wordpress.wp_statistics_events:SELECT
  - wordpress.wp_statistics_exclusions:SELECT
  - wordpress.wp_statistics_historical:SELECT
  - wordpress.wp_statistics_pages:SELECT
  - wordpress.wp_statistics_search:SELECT
  - wordpress.wp_statistics_useronline:SELECT
  - wordpress.wp_statistics_visit:SELECT
  - wordpress.wp_statistics_visitor:SELECT
  - wordpress.wp_statistics_visitor_relationships:SELECT

KUBERNETES

Visão geral da arquitetura desenvolvida dentro do **kubernetes**:

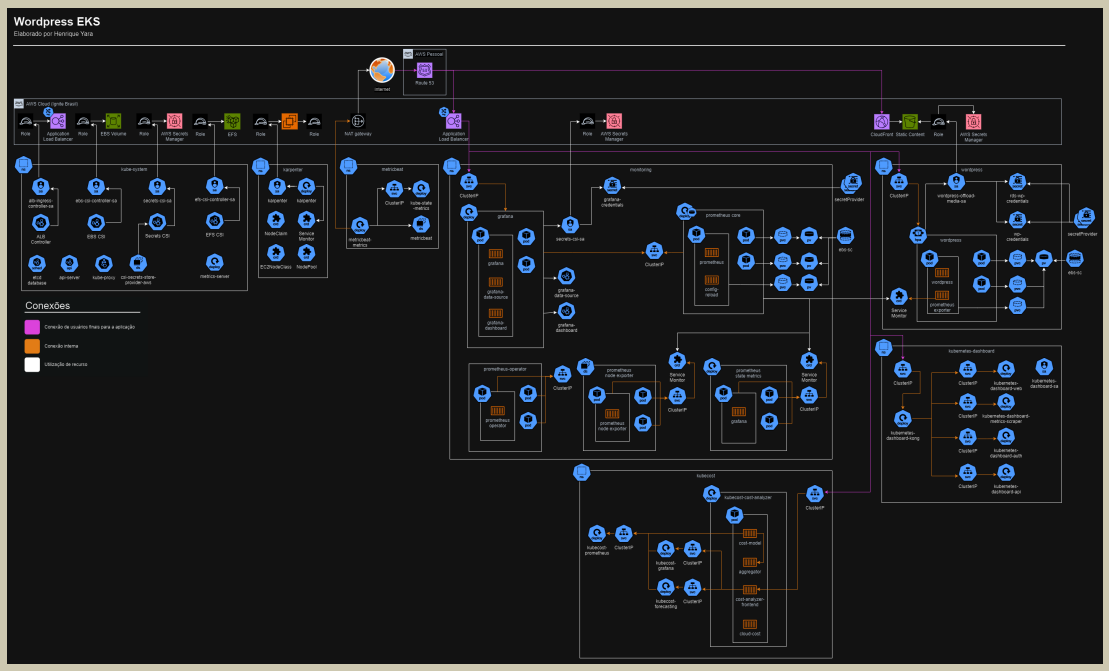

Figure 9: Kubernetes arquitetura v1.4

8.1 kube-system

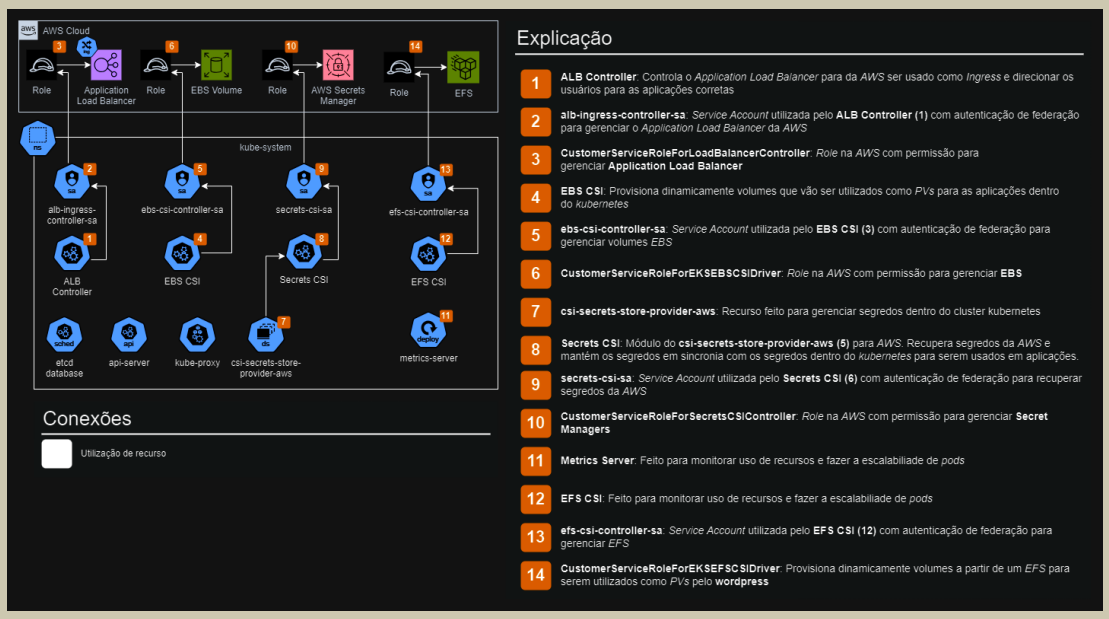

Figure 10: kube-system namespace

- Application Load Balancer Controller: Alocação de **Application Load Balancer** para servir de ponto de entrada para as aplicações dentro do *cluster*
  - Service Account: alb-ingress-controller-sa
  - Role: CustomerServiceRoleForLoadBalancerController
- AWS EBS CSI: Alocação de **Elastic Block Storage** para servir como **Persistent Volume** para aplicações **StatefulSet**
  - Service Account: ebs-csi-controller–sa
  - Role: CustomerServiceRoleForEKSEBSCSIDriver
- AWS EFS CSI: Alocação de **Elastic FileSystem** para servir como **Persistent Volume** para o **wordpress**
  - Service Account: efs-csi-controller–sa
  - Role: CustomerServiceRoleForEKSEFSCSIDriver
- AWS Secrets CSI: Sincronização de segredos do **Secret Manager** da **AWS** com os objetos **secrets** dentro do **kubernetes**

- Service Account: secrets-csi-sa
- Role: CustomerServiceRoleForSecretsCSIController

- Metric Server: Responsável por prover informações para a escalabilidade do **HorizontalPodAutoscaling**
  - Toleration: CriticalAddonsOnly:Exists

8.2 wordpress

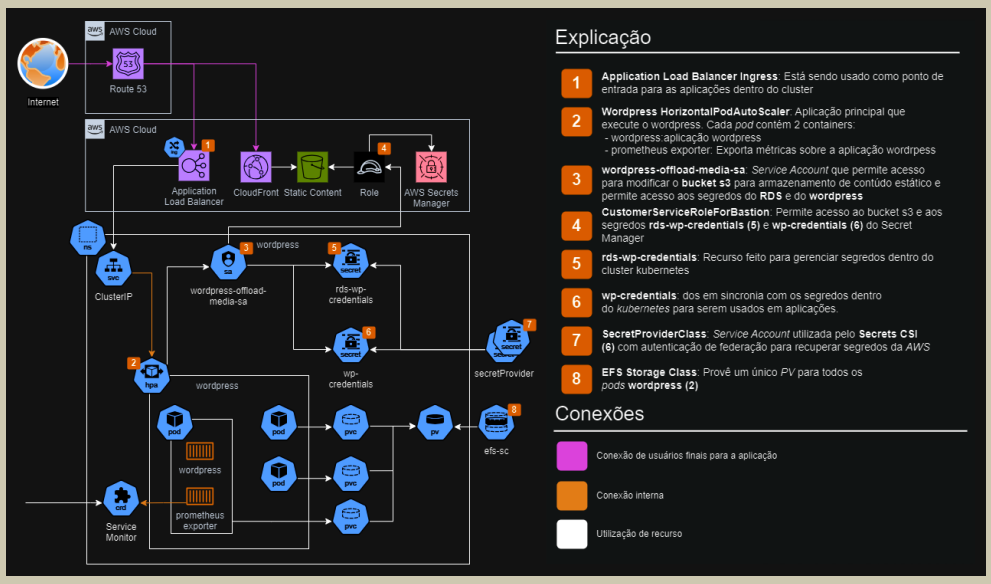

Figure 11: wordpress namespace

Release: wordpress
Namespace: wordpress
Chart: link
Hostname: https://wordpress.htsuyoshiy.online
Descrição: Aplicação principal desenvolvida para o projeto
Service Account: CustomerServiceRoleForWordpressOffloadMedia
Ingress:

```
annotations:
  alb.ingress.kubernetes.io/scheme: 'internet-facing'
  alb.ingress.kubernetes.io/target-type: 'ip'
  alb.ingress.kubernetes.io/load-balancer-attributes:
    idle_timeout.timeout_seconds=60
  alb.ingress.kubernetes.io/target-group-attributes:
    stickiness.enabled=true,
    stickiness.type=lb_cookie,
    stickiness.lb_cookie.duration_seconds=86400
  alb.ingress.kubernetes.io/ssl-redirect: '443'
```

8.2.1 Horizontal Pod Autoscaling

- Min replicas: 3
- Max replicas: 7
- Target cpu: 70%
- Target memory: 70%

8.2.2 Wordpress Plugins

- amazon-s3-and-cloudfront
- wp-statistics

Configurações do **amazon-s3-and-cloudfront**:

```
define( 'AS3CF_SETTINGS', serialize( array(
  'provider' => 'aws',
  'use-server-roles' => true,
  'bucket' => '{{ wp_offload_media_bucket }}',
  'region' => '{{ region }}',
  'copy-to-s3' => true,
  'enable-object-prefix' => true,
  'object-prefix' => 'wp-content/uploads/',
  'use-yearmonth-folders' => true,
  'remove-local-file' => true,
  'object-versioning' => true,
  'serve-from-s3' => true,
  'enable-delivery-domain' => true,
  'domain' => 'cloudfront',
  'delivery-domain' => '{{ wp_offload_media_cf }}',
  'domain-rewrite' => true,
  'path-rewrite' => true
)))
```

8.3 monitoring

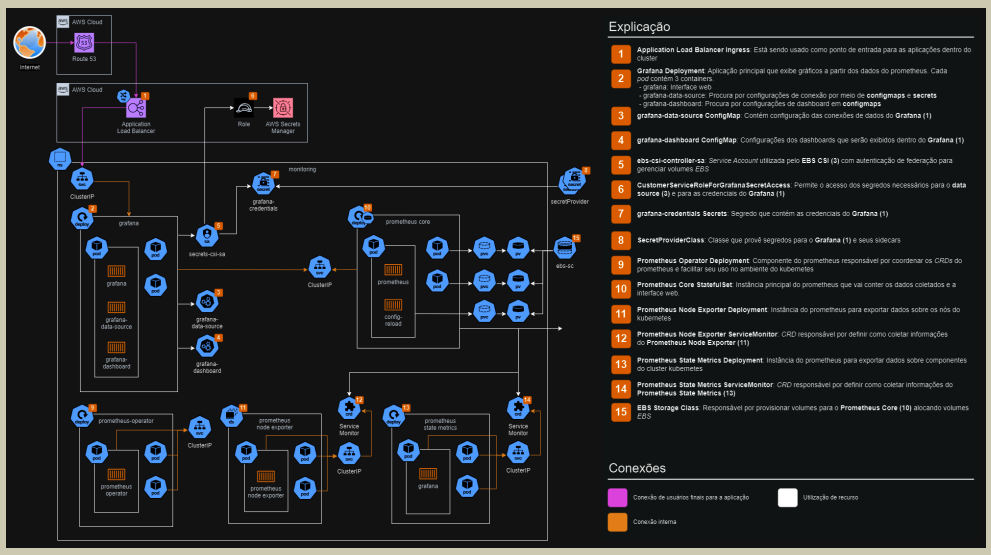

Figure 12: monitoring namespace

Release: prometheus
Namespace: monitoring
Chart: link
Hostname: https://grafana.htsuyoshiy.online
Descrição: Visualizar métricas do da aplicação **wordpress**, métricas específicas do **karpenter** e métricas do *cluster*,
Ingress:

```
annotations:
  alb.ingress.kubernetes.io/scheme: 'internet-facing'
  alb.ingress.kubernetes.io/target-type: 'ip'
  alb.ingress.kubernetes.io/load-balancer-attributes:
    idle_timeout.timeout_seconds=60
  alb.ingress.kubernetes.io/target-group-attributes:
    stickiness.enabled=true,
    stickiness.type=lb_cookie,
    stickiness.lb_cookie.duration_seconds=86400
  alb.ingress.kubernetes.io/ssl-redirect: '443'
```

8.4 kubernetes-dashboard

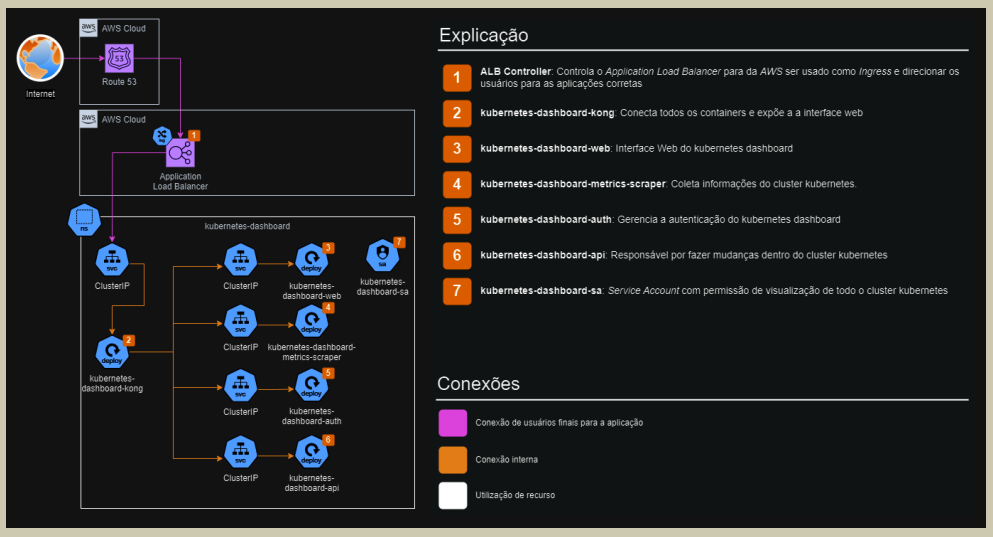

Figure 13: kubernetes-dashboard namespace

Release: kubernetes-dashboard
Namespace: kubernetes-dashboard
Chart: link
Hostname: https://kubernetes.htsuyoshiy.online
Descrição: Possibilita o gerenciamento do *dashboard* por uma interface web, é possível também ver o estado do *cluster*.
Ingress:

```yaml
annotations:
  alb.ingress.kubernetes.io/scheme: 'internet-facing'
  alb.ingress.kubernetes.io/target-type: 'ip'
  alb.ingress.kubernetes.io/load-balancer-attributes:
    idle_timeout.timeout_seconds=60
  alb.ingress.kubernetes.io/target-group-attributes:
    stickiness.enabled=true,
    stickiness.type=lb_cookie,
    stickiness.lb_cookie.duration_seconds=86400
  alb.ingress.kubernetes.io/ssl-redirect: '443'
  alb.ingress.kubernetes.io/backend-protocol: 'HTTPS'
```

8.5 metricbeat

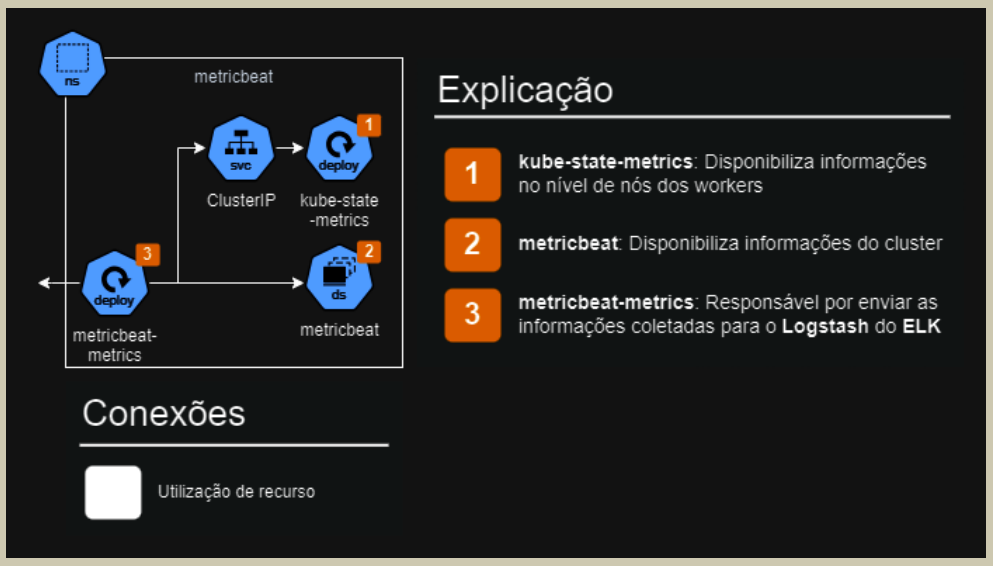

Figure 14: metricbeat namespace

Release: metricbeat
Namespace: metricbeat
Chart: link
Descrição: Responsável por enviar informações do status do *cluster* para o **ELK**.
Link para o **ELK**: `http://elk.htsuyoshiy.online`

8.6 kubecost

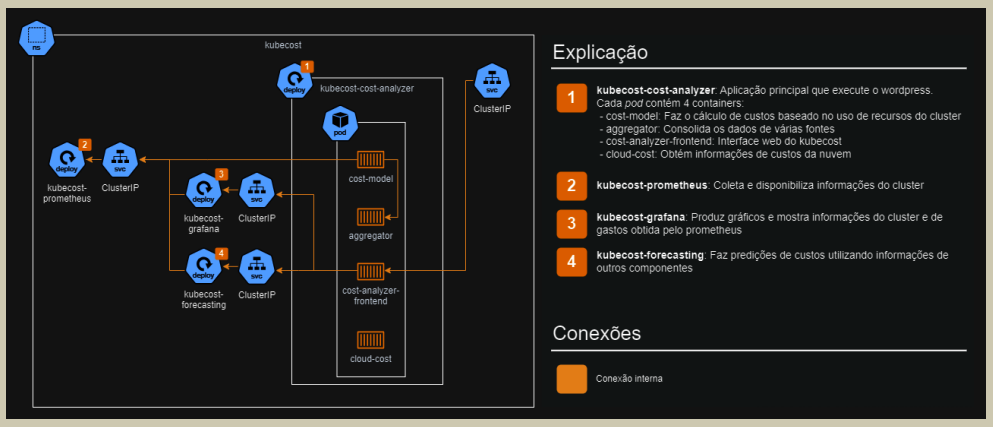

Figure 15: kubecost namespace

Release: kubecost
Namespace: kubecost
Chart: link
Hostname: https://kubecost.htsuyoshiy.online
Descrição: Visualizar gastos do *cluster* e a eficiência de uso dos recursos do *cluster*.
Ingress:

```
annotations:
  alb.ingress.kubernetes.io/scheme: 'internet-facing'
  alb.ingress.kubernetes.io/target-type: 'ip'
  alb.ingress.kubernetes.io/load-balancer-attributes:
    idle_timeout.timeout_seconds=60
  alb.ingress.kubernetes.io/target-group-attributes:
    stickiness.enabled=true,
    stickiness.type=lb_cookie,
    stickiness.lb_cookie.duration_seconds=86400
  alb.ingress.kubernetes.io/ssl-redirect: '443'
```

8.7 karpenter

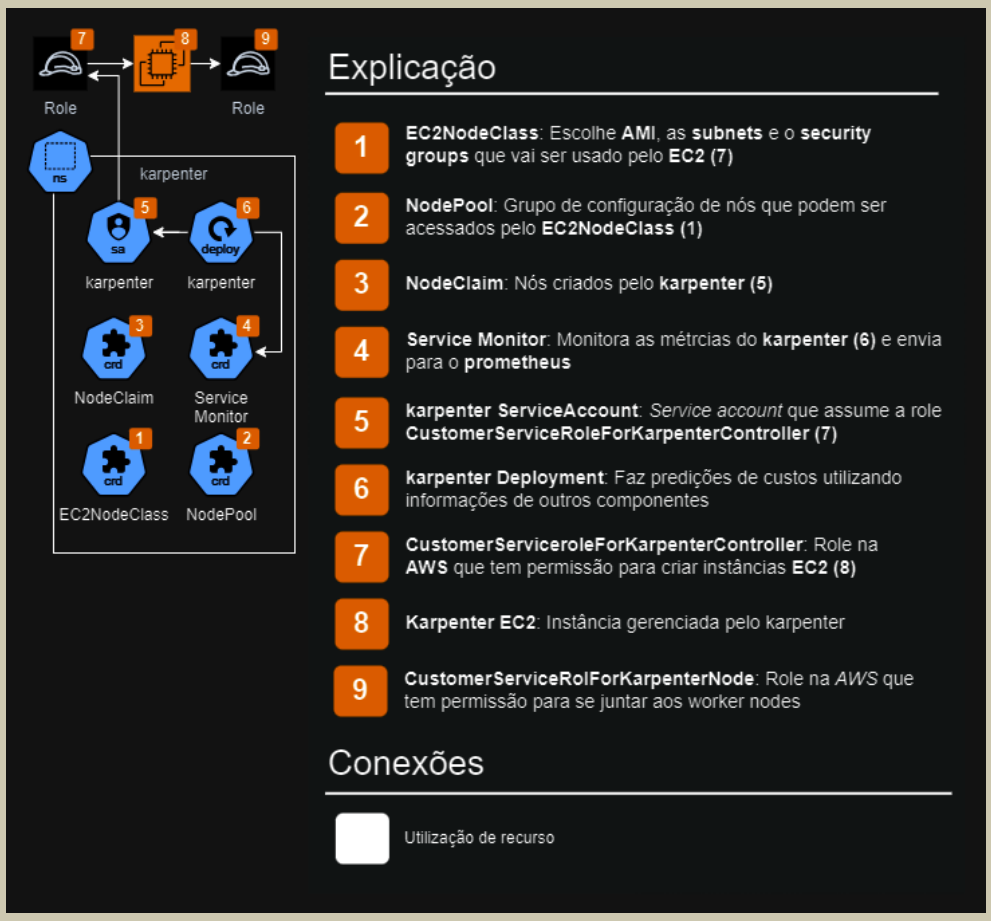

Figure 16: karpenter namespace

Release: karpenter
Namespace: karpenter
Chart: link
Descrição: Responsável por fazer o *autoscaling* dos nós do cluster **EKS**. O **karpenter** vai ser executado nos nós do **EKS Auto Scaling Groups** e vai provisionar nós para as outras aplicações.

8.7.1 EC2NodeClass

CRD do **karpenter** que define qual **Role**, **Security Groups**, **Availability Zone** e **AMI** vai ser utilizado pelos nodes do **karpenter**

- AMI: amazon-eks-node-1.30-v20240729
- Role: CustomerServiceRoleForKarpenterNode
- Subnet com Tag: karpenter.sh/discovery: wordpress-eks
  - EKS Subnet 1: 192.168.0.0/18
  - EKS Subnet 2: 192.168.64.0/18
- **Security Group** com **Tag**: karpenter.sh/discovery: wordpress-eks
  - Mesmo **Security Groups** criado pelo **EKS**

8.7.2 NodePool

Definem quais configurações as instâncias **EC2** que vão ser alocadas pelo **karpenter** vão ter.

NodePool criados:

- Propósito geral
  - t3.small, t3.medium
- Propósito geral (Spot)
  - t3.small, t3.medium

8.7.3 NodeClaim

CRD utilizado pelo **karpenter** para alocar uma instância **EC2.**

CUSTOS

9.1 Calculadora AWS

Link para calculadora AWS: link

Service	Uso	Custo
NAT Gateway	730 horas × 2	65.7
NAT Gateway data	50 Gb	4.50
EKS Cluster	730 horas	73.00
EKS Node Group (2 t3.small)	730 horas × 3	33.57
Karpenter Node (3 t3.medium)	730 Horas	31.22
EKS Node Group Storage (gp2)	60 Gb	4.80
Secrets Manager	4 segredos	1.60
Secrets Manager	100.000 chamadas de API	0.50
RDS (on-demand, 1 AZ, db.t3.micro)	730 horas	12.41
RDS Storage (gp2)	730 horas	2.30
EFS	10Gb	0.90
SQS	1M	0.00
Cloud front (Free tier)	1M	0.00
S3 bucket	1M	0.46
WAF Web ACL	2M	1.20
Custo mensal aproximado		227.44

Table 9: Custos da calculadora AWS