Documentação Projeto Compass UOL “Atividade Linux”

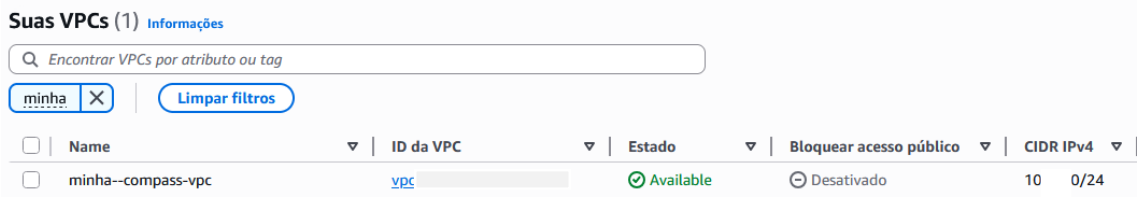
Esse relatório trata a respeito do primeiro projeto do programa de bolsas *Compass UOL*, para a execução dele é aplicado conceitos de servidor *Linux* e computação em nuvem da *AWS* (*Amazon Web Services*). As etapas realizadas são apresentadas nas sessões seguintes.

O objetivo desse projeto é configurar um ambiente em nuvem *AWS* (ambiente de rede e sistema operacional) para que então seja realizado a configuração do servidor *web* Nginx e, com uso de *Webhook* criar logs e monitorar o status de disponibilidade da página *web* avisando ao administrador se a página ficar indisponível.

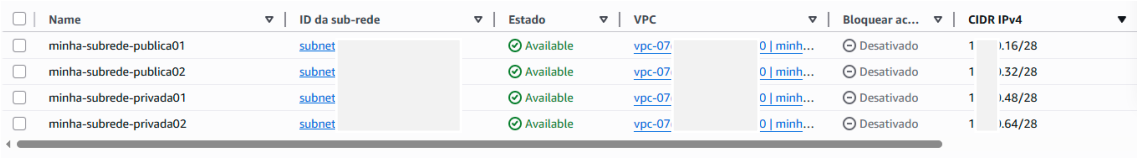
Primeira Etapa

Inicialmente, foi criada a *VPC* (*Virtual Private Cloud*) na *AWS*, utilizando o bloco CIDR /24, que corresponde a 256 endereços IP (*Internet Protocol)* possíveis. Em seguida, foram definidas duas sub-redes públicas (para acesso externo) e duas sub-redes privadas, cada uma com bloco CIDR /28, o que resulta em 16 endereços IP por sub-rede, conforme os requisitos do projeto. A Figura 1 retrata essas configurações.

Figura 1 – VPC e as Sub-redes criadas.



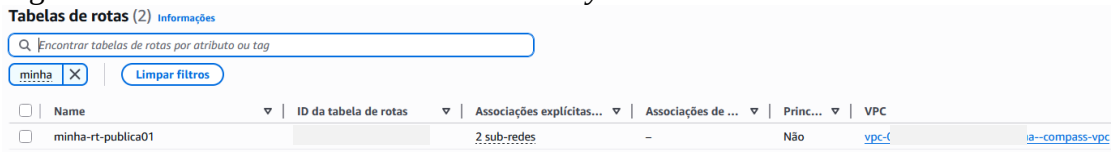
[1] VPC



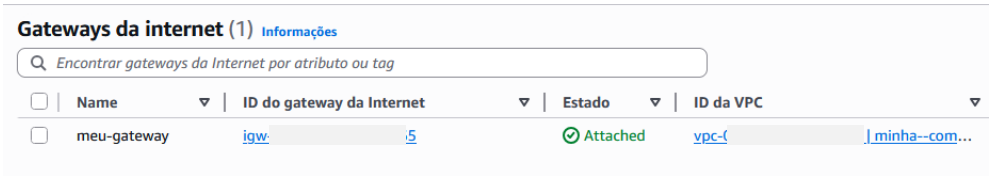
[2] Sub-redes

Na sequência, foi criado um *Internet Gateway*, *sendo ele* anexado à VPC. Em seguida, a tabela de rotas das sub-redes públicas foram configuradas para direcionar o tráfego de saída (0.0.0.0/0) para esse gateway. Essa etapa é essencial para permitir que a instância que será criada em nuvem AWS possa acessar a internet e ser acessada externamente, como no caso de um servidor web. Na figura 2 se pode observar as capturas de telas criadas nesse momento.

Figura 2 – Tabela de Rotas e *Internet Gateway*.



[1] Tabela de Rotas.



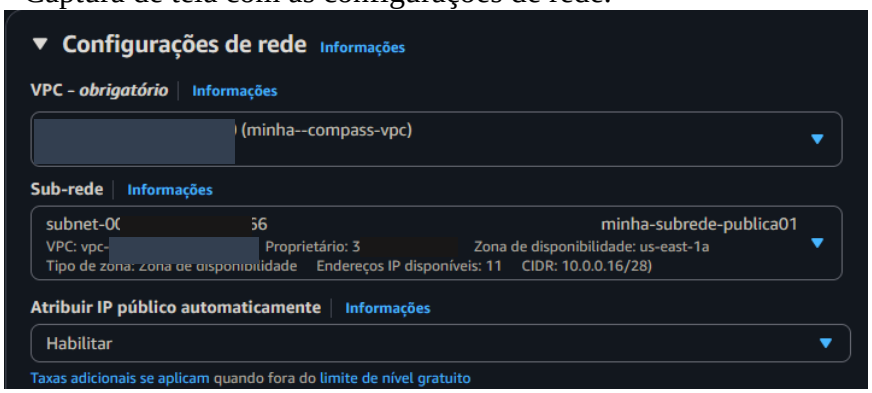
[2] *Internet Gateway.*

Após a criação e configuração da VPC, foi configurada uma instância EC2 (*Elastic Compute Cloud) "t2.micro"* com Linux (Ubuntu) com armazenamento de 8 GiB. Nela é necessário inserir os nomes e *tags* da organização, sem essas informações não é possível ser criada a instância.

A instancia EC2 foi associada a VPC criada previamente, sendo vinculada à primeira sub-rede pública (figura 3[1]). As portas 443, 80 e 22 foram liberadas no grupo de segurança para permitir o tráfego HTTPS, HTTP e SSH, respectivamente, com acesso aberto para todos os IPs (figura 3[2]).

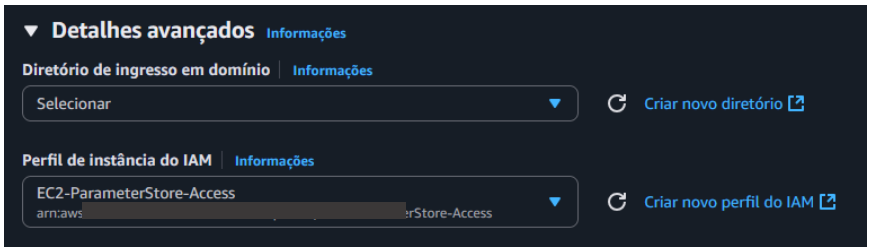
Na seção "detalhes avançados", deve ser selecionado o perfil de instância do IAM (figura 3[3]), esse procedimento será explicado com mais detalhes na seção "Terceira Etapa". A transmissão de dados utiliza a porta 443.

Figura 3 – Captura de tela com as configurações de rede.



[1] Configurações de rede.

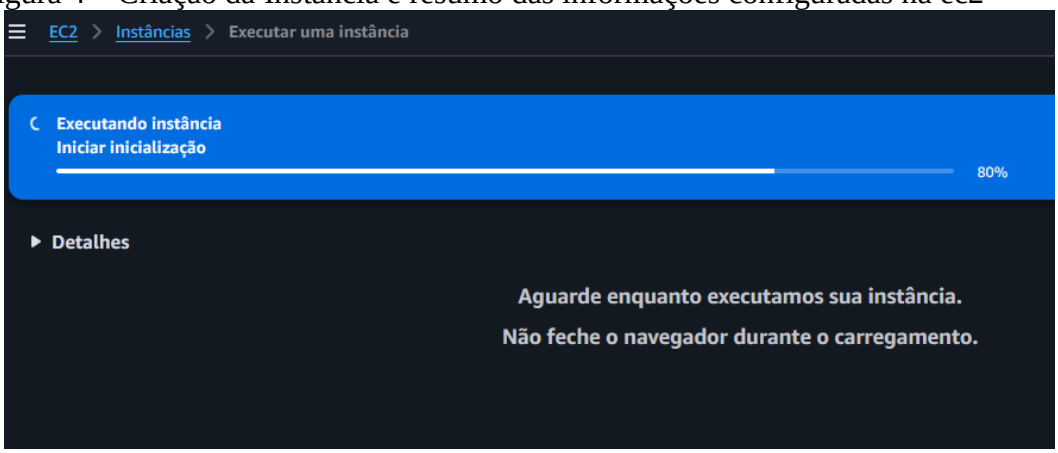
[2] Regras do grupo de segurança.



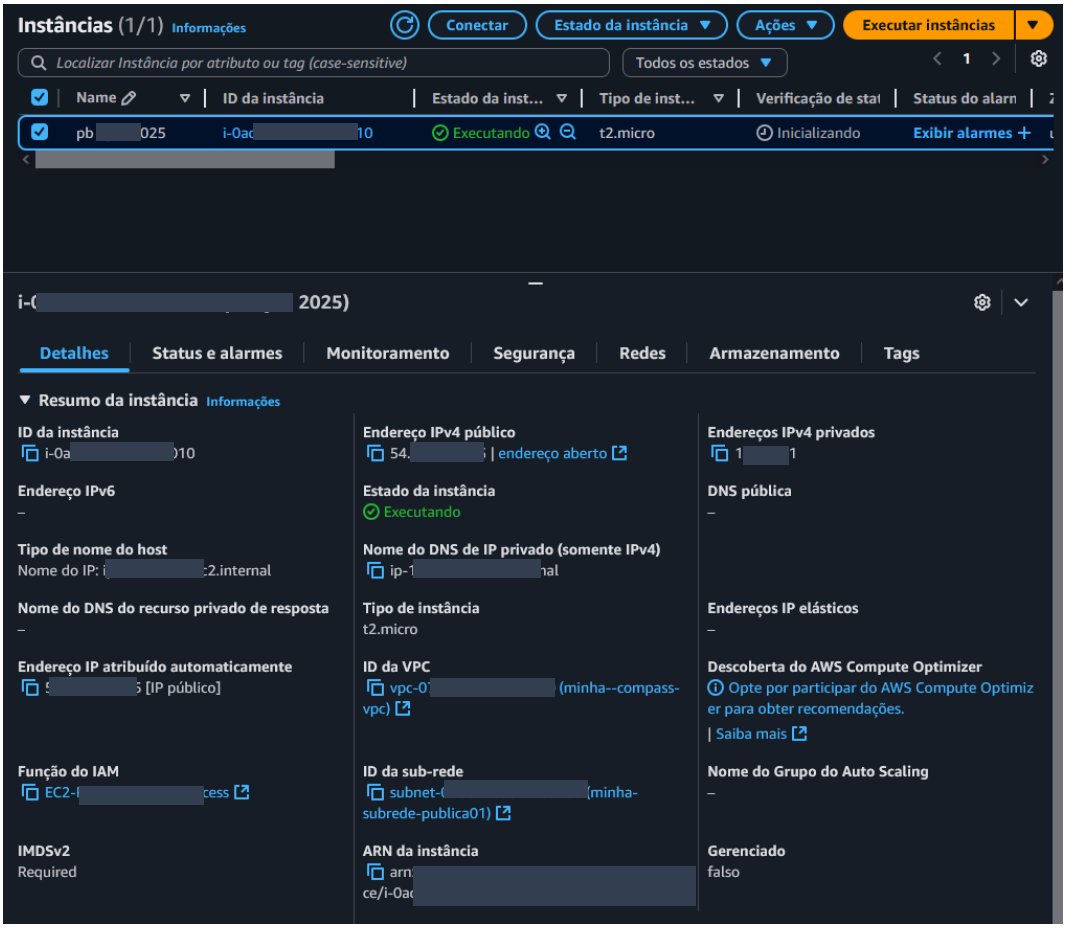
[3] Perfil da instância do IAM para acesso aos segredos armazenados.

Dessa maneira, com as configurações de criação realizadas a instância EC2 foi executada, figura 4[1]. Após a criação, ela é inicializada automaticamente e assim é possível realizar o acesso remoto.

Figura 4 – Criação da Instância e resumo das informações configuradas na ec2



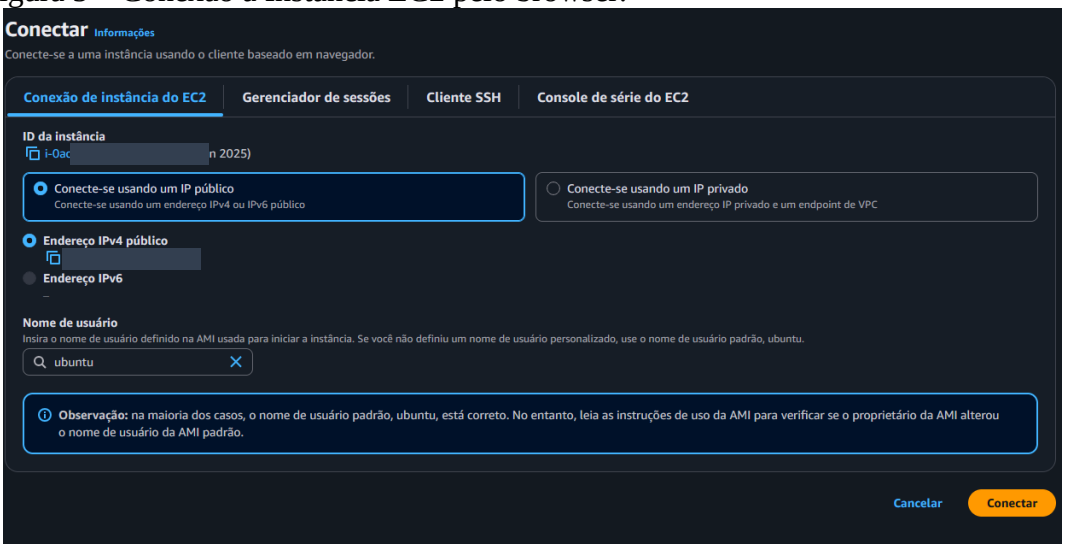
[1] Progresso da Criação da Instância EC2.



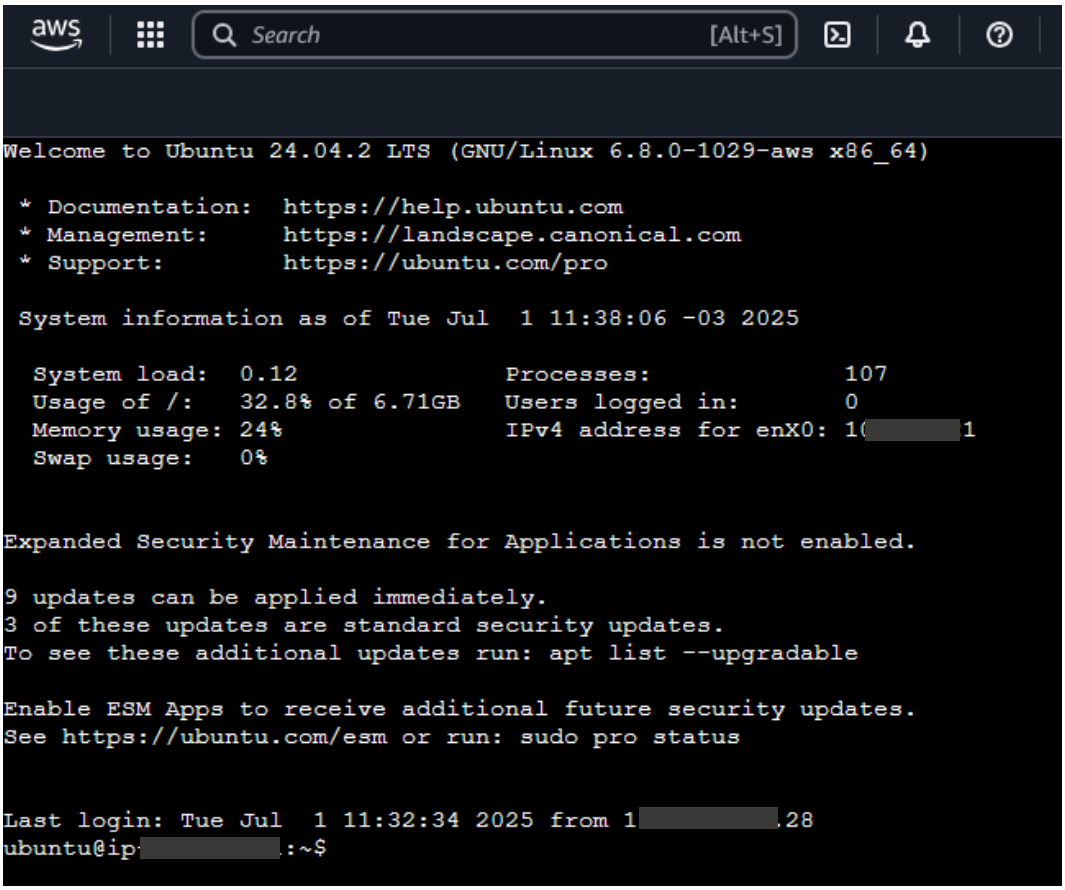
[2] Instância criada e em Execução

O acesso remoto da máquina criada pode ocorrer pelo navegador, selecionando a opção “Conectar” (Figura 4[2]), nessa opção abre uma nova aba, figura 5[1] com mais algumas informações da instância disponível e então se for selecionado novamente a opção “Conectar” é carregado a máquina, já inicializada no usuário padrão, conforme observado na figura 5[2].

Figura 5 – Conexão à Instância EC2 pelo browser.



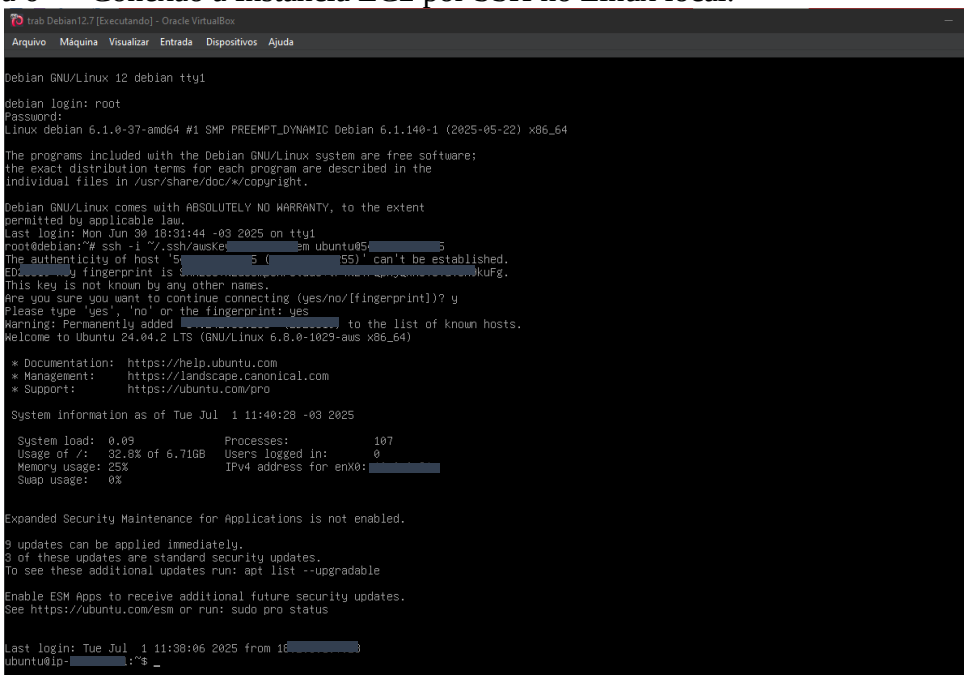
[1] Tela com informações para a conexão com o navegador.



[2] Tela da Instância conectada no navegador.

E uma outra forma de realizar o acesso remoto à instância é estabelecendo uma conexão remota com chave SSH no Linux (ou outro Sistema Operacional) utilizando para isso o endereço de rede público da instância criada, conforme observado na figura 6.

Figura 6 - – Conexão à Instância EC2 por SSH no Linux local.



Para realizar a conexão com uso de *SSH* é necessário instalar o servidor *SSH*, "*apt-get install openssh-server*", e também iniciar e habilitar o serviço SSH com o *"systemctl"*. A conexão é realizada pelo comando digitado no terminal Linux "*ssh –i /'caminho para chave'/'chave.pem' usuario@ip_publico_EC2"*, no caso o usuário padrão da instância EC2 Linux Ubuntu é "ubuntu".

Segunda Etapa

Na figura 7, tem-se a instalação padrão do servidor web *Nginx* (*Engine X*) no Linux se deu pelo comando "*apt-get install nginx –y*", porém antes de executar a instalação é recomendado usar o "*apt-get update*" e "*apt-get upgrade*" para atualizar os endereços dos repositórios e atualizar os pacotes que já estão instalados no Linux. O *Nginx* na sua instalação padrão costuma já ser iniciado imediatamente após a instalação e também habilitado para iniciar automaticamente no *boot* do sistema operacional. Portanto a página padrão básica do servidor fica disponível para acesso no browser pelo endereço de rede do computador.

Figura 7 – Atualização de pacotes e instalação Nginx

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-[redacted]:~$ sudo apt-get update -y; sudo apt-get upgrade -y;
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [920 kB]
```

(...)

```
ubuntu@ip[redacted]~$ sudo apt-get install -y nginx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  nginx-common
Suggested packages:
  fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
  nginx nginx-common
0 upgraded, 2 newly installed, 0 to remove and 5 not upgraded.
Need to get 564 kB of archives.
After this operation, 1596 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 nginx-common all 1.24.0-2ubuntu7.4 [43.4 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 nginx amd64 1.24.0-2ubuntu7.4 [521 kB]
Fetched 564 kB in 0s (14.8 MB/s)
Preconfiguring packages ...
Selecting previously unselected package nginx-common.
(Reading database ... 70681 files and directories currently installed.)
Preparing to unpack .../nginx-common_1.24.0-2ubuntu7.4_all.deb ...
Unpacking nginx-common (1.24.0-2ubuntu7.4) ...
Selecting previously unselected package nginx.
Preparing to unpack .../nginx_1.24.0-2ubuntu7.4_amd64.deb ...
Unpacking nginx (1.24.0-2ubuntu7.4) ...
Setting up nginx-common (1.24.0-2ubuntu7.4) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
Setting up nginx (1.24.0-2ubuntu7.4) ...
 * Upgrading binary nginx
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for ufw (0.36.2-6) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...
```

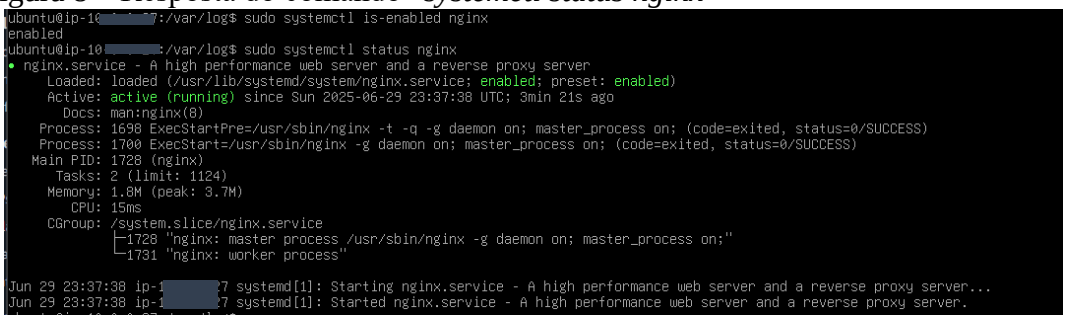
E para garantir que o sistema funcionará conforme é desejado, é também inserido manualmente os comandos habilitar o serviço após a instalação e para fazê-lo iniciar automaticamente na inicialização do sistema operacional. E após a inserção dos comandos é verificado se o serviço está habilitado para iniciar automaticamente, os comandos utilizados no terminal podem ser observados na tabela 1, na sequência.

Tabela 1 – Comandos para habilitar e inicializar o nginx.

Comando	Função
systemctl start nginx	Inicializa o serviço Nginx imediatamente;
systemctl enable nginx	Cria um link simbólico do serviço nos diretórios de inicialização automática.
systemctl status nginx	Verifica o status atual do Nginx (Se está ativo).
systemctl is-enabled nginx	Verifica se o Nginx está habilitado para inicialização automática.

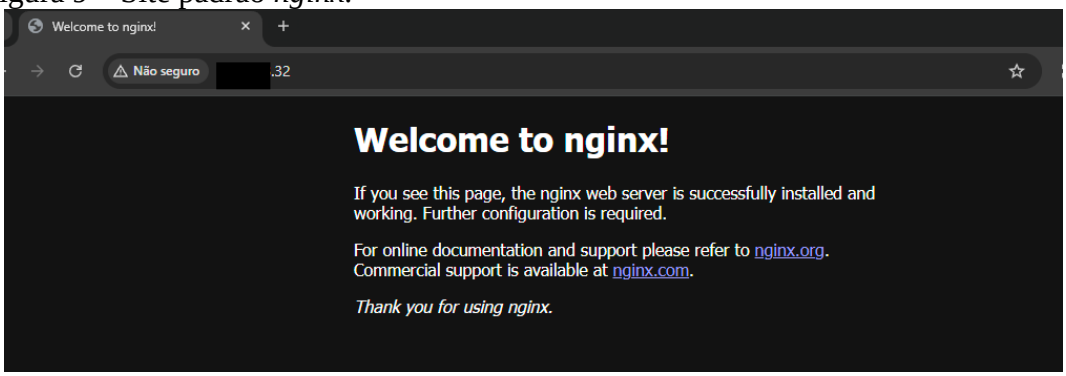
Assim, foi executado o comando "*systemctl status nginx*" para confirmar que o servidor *web* está funcionando corretamente e sem nenhum conflito. E conforme pode ser observado na figura 8, o mesmo encontra-se ativo e funcionando corretamente.

Figura 8 – Resposta do comando "*systemctl status nginx"*



Após na instalação do *nginx*, já é criada uma página *HTML* (*HyperText Markup Language)* básica, sendo possível acessá-la pelo endereço de rede público da instância EC2. Na figura 9, abaixo, é mostrado a captura de tela desse site padrão do *nginx*.

Figura 9 – Site padrão *nginx.*



Verificado que o servidor está funcionando corretamente é necessário modificar o site padrão para um que contenha as informações necessárias do projeto. No diretório "/etc/nginx/sites-avaliable/" está o arquivo de configuração padrão "default" nele contém dados de qual porta está escutando e também onde está localizado o site padrão (figura 10), no caso, não há necessidade de alteração.

Figura 10 – Arquivo de configuração padrão dos sites do servidor web.

```
# Default server configuration
#
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }
```

E no diretório "/var/www/html/" está localizado o site básico do *nginx*, apresentado acima (Figura 10). O nome dele é "*index.nginx-debian.html*" e conforme observado no arquivo de configuração do servidor web, ele serve a primeira página disponível seguindo a ordem apresentada. Dessa maneira assim que for criado o arquivo "index.html" ele passará a ser servido pelo *nginx.*

O novo site para mostrar que o servidor está funcionando corretamente foi carregado com o nome "index.html", o código *HTML* e *CSS* da página pode ser observado na figura 11.

Esse site foi criado externamente, compactado em *zip* e então salvo no repositório do *Github* para que fosse realizado o *download* de dentro da máquina EC2. O download, descompactação e configuração dos arquivos da página foi realizado com os comandos presentes na tabela 2.

Figura 11 – Site index.html, servido pelo nginx.

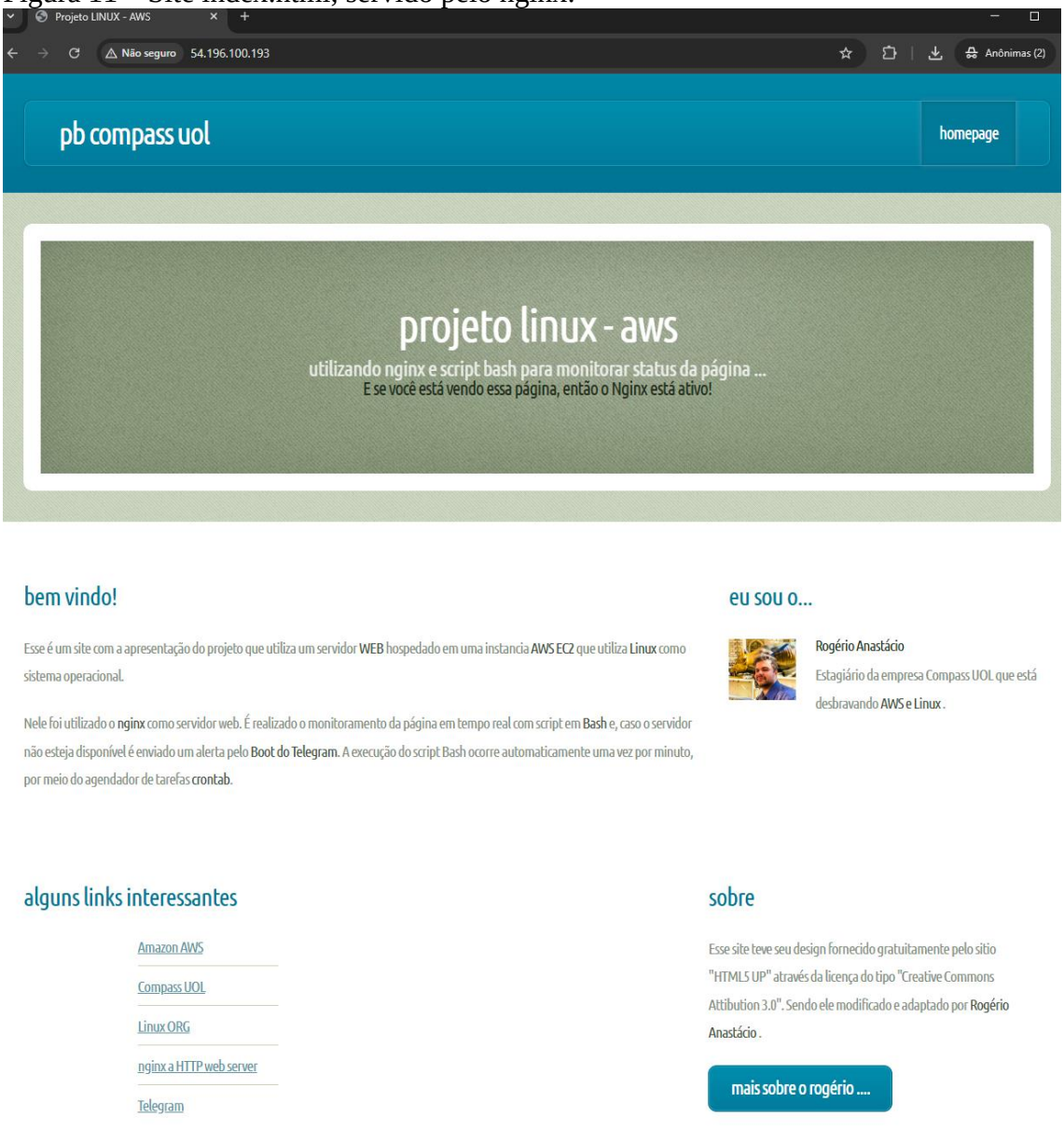


Tabela 2 – Comandos realizar download, descompactar e configurar as permissões de acesso aos arquivos do site.

Comando	Função
apt-get install unzip -y	Instala o pacote Unzip.
curl -L "https://github.com/master-rogerio/HelloWorld/raw/refs/heads/main/site.zip" -o "/tmp/site.zip"	Faz *download* do arquivo e o armazena no diretório /tmp/.
unzip -q -o "/tmp/site.zip" -d "/var/www/html/"	Descompacta todos os arquivos na pasta /var/www/html/
rm -f "/tmp/site.zip"	Remove o arquivo baixado sem exibir erro (caso ele não exista).
chown -R www-data:www-data /var/www/html	
find /var/www/html -type d -exec chmod 755 {} \;	
find /var/www/html -type f -exec chmod 644 {} \;	

Terceira Etapa

Na terceira etapa é apresentado como foi realizado o monitoramento e o envio das notificações de disponibilidade do *site* para o *Telegram*.

O monitoramento foi realizado através da execução de um *script Bash* que chama o programa de linha de comando *"curl"*. A cada vez que *script* é executado ocorre uma consulta ao endereço de rede do *site* para saber o *status* do mesmo. Esse script é armazenado em *"/usr/local/bin/"* com o nome de *"monitor_site.sh"*. As configurações de acesso ao script é alterada com o "*chmod"* para ser executável ("*+x")*.

Nessa consulta, caso seja retornado o código HTTP igual a 200, corresponde que o site está online e disponível, é salvo um registro com a data e o código retornado. E nesse script se for retornado um código diferente é considerado site indisponível além de salvar o registro com data do evento é disparado uma mensagem de notificação para um *bot* do *Telegram*. E para que o script seja executado periodicamente é criado uma tarefa, no agendador de tarefas *cron*.

Na figura 12, abaixo, pode ser observado o script usado verificação de disponibilidade, armazenamento de log e disparo de mensagem. O primeiro ponto e que para enviar as notificações no *Telegram* é necessário colocar o código do *token* e o código identificador do *chat.* Por esses dados serem sensíveis é utilizada a ferramenta da *AWS* (*AWS Systems Manager*) "Armazém de Parâmetros" para armazenar as chaves e o *IAM (Identity and Access Management)* para gerenciar identidades e controlar o acesso aos recursos digitais de forma segura.

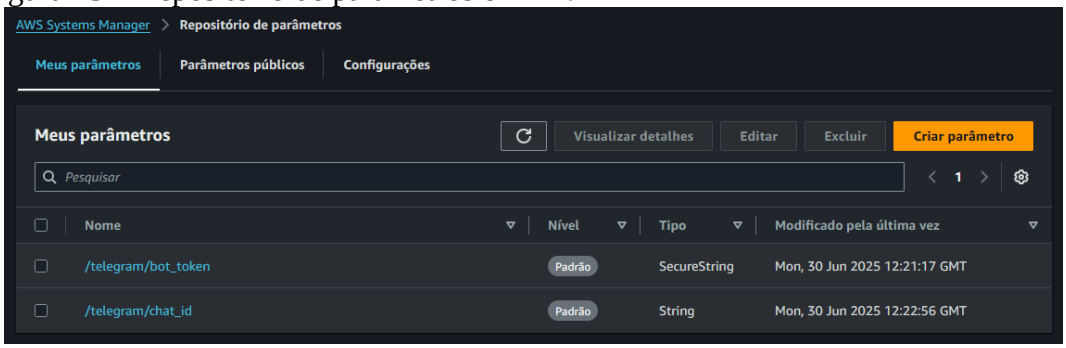
Figura 12 – Script Bash (monitor_site.sh) para verificar disponibilidade do site.

```
#!/bin/bash

#Busca Parametros armazenados no Repositorio de Parametros AWS (IAM)
aws configure set region us-east-1
TELEGRAM_BOT_TOKEN=$(aws ssm get-parameter --name "/telegram/bot_token" --with-decryption --query "Parameter.Value" --output text)
TELEGRAM_CHAT_ID=$(aws ssm get-parameter --name "/telegram/chat_id" --query "Parameter.Value" --output text)

# Configurações do Telegram
BOT_TOKEN="$TELEGRAM_BOT_TOKEN"
CHAT_ID="$TELEGRAM_CHAT_ID"

SITE_URL="http://$(curl -s https://checkip.amazonaws.com/)" # Busca Ip Publico

response=$(curl -s -o /dev/null -w "%{http_code}" $SITE_URL) # Verifica resposta do site

timestamp=$(date "+%Y-%m-%d %H:%M:%S") #Variavel que armazena Data e Hora atual

if [ "$response" -eq 200 ]; then
    echo "[$timestamp] Site $SITE_URL está respondendo normalmente. Código: $response" >> /var/log/monitor_site.log
else
    echo "[$timestamp] ALERTA: $SITE_URL não está respondendo. Código: $response" >> /var/log/monitor_site.log

    # Envia mensagem para o Telegram
    message="🚨 *ALERTA DE MONITORAMENTO* 🚨

    O site está Offline!

    *IP:* $SITE_URL
    *Código HTTP:* $response
    *Hora:* $timestamp"

    curl -s -X POST \
        "https://api.telegram.org/bot$BOT_TOKEN/sendMessage" \
        -d chat_id="$CHAT_ID" \
        -d text="$message" \
        -d parse_mode="Markdown" >> /dev/null
fi
```

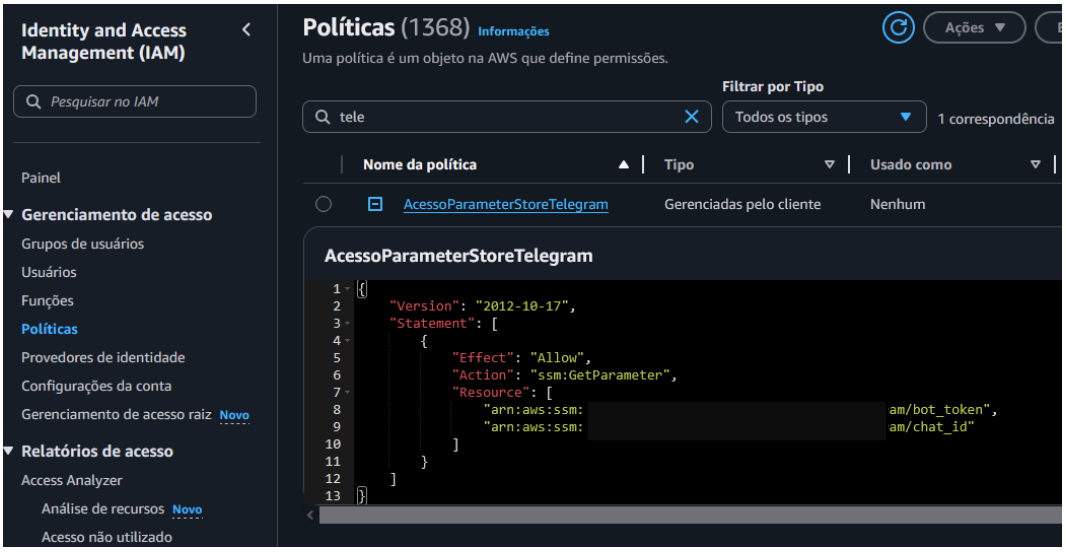
Assim na sequência (figura 13), é apresentado as capturas de telas relacionadas ao serviço de armazenamento seguro das informações, na figura 13[1] é apresentado a forma que os parâmetros do *token ("bot_token")* e *chat("chat_id")* foram armazenados no

repositório de parâmetros. Enquanto na figura 13[2] tem-se a política de acesso para acessar as chaves pela EC2.

Figura 13 – Repositório de parâmetros e IAM.



[1] Repositório de parâmetros.



[2] Parâmetro no IAM para acessar pela EC2.

Agora para acessar as chaves dentro da instância EC2, é utilizado a funcionalidade do *AWS CLI (Amazon Web Services Command Line Interface*), essa é uma ferramenta que permite a interação com os serviços *AWS* usando comandos no *shell* de linha de comando.

É necessário instalá-lo, e o primeiro passo é realizar o *download* do pacote. Esse pacote vem compactado no formato *.zip. A linhas de comandos que foram digitadas no terminal para a instalação do *AWS CLI* podem ser observadas na tabela 3.

Dessa forma, realizando esses passos já é possível acessar as informações sensíveis armazenadas no repositório de parâmetros da *AWS*. E assim para que o script seja executado periodicamente é necessário instalar o agendador de tarefas “cron” (*apt-get install cron –y).* Após a instalação é executado o *“crontab”* com a linha de comando observado na tabela 4, o agendamento é realizado para que o *cron* execute o *script* “*monitor_site.sh*” uma vez por minuto.

Tabela 3 – Comandos para instalar AWS CLI.

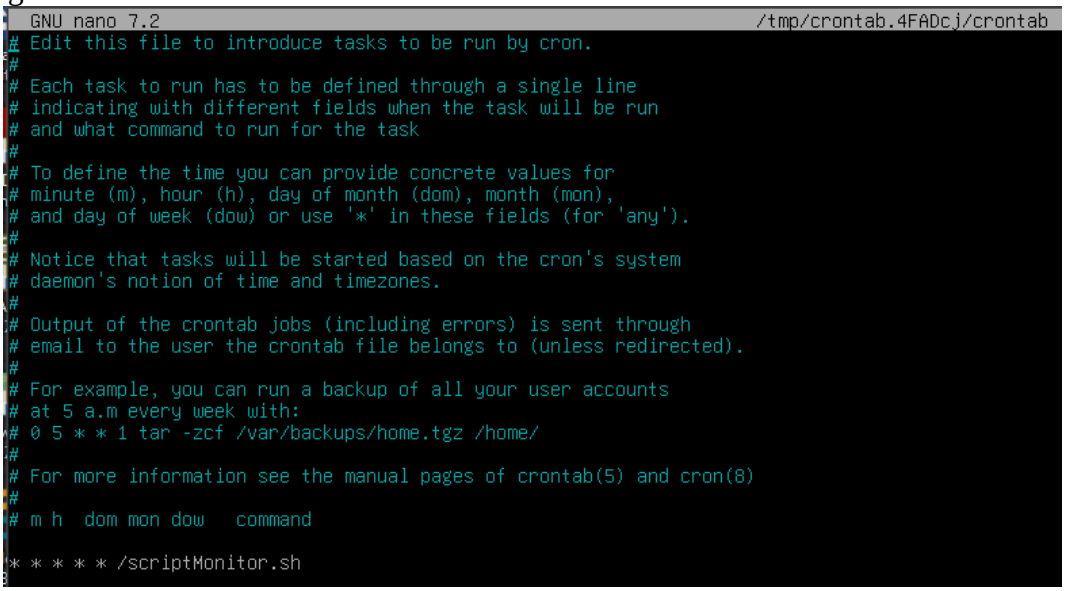
Comando	Função
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "/tmp/awscliv2.zip"	Faz *download* do pacote e armazena no diretório /tmp/.
sudo unzip -q /tmp/awscliv2.zip -d /tmp	Descompacta o arquivo na pasta /tmp/
/tmp/aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --update	(Explicado por partes abaixo)
/tmp/aws/install	Local onde se encontra o script para instalação do *AWS CLI*.
--bin-dir /usr/local/bin	Local onde será colocado o Executável AWS (para acesso pelo terminal)
--install-dir /usr/local/aws-cli	Diretório onde será instalado o AWS CLI.
--update	Se houver versão mais antiga, realiza a atualização

Após o agendamento da tarefa é possível acessar o editor do "*cron"* pelo comando "*crontab -e*", observado na figura 14, ou então enviar o comando "crontab –l" para listar todas as tarefas agendadas para o usuário atual.

Tabela 4 – Comandos execução do *crontab*.

Comando	Função
(crontab -l 2>/dev/null; echo "* * * * * /usr/local/bin/monitor_site.sh") | crontab -	(Explicado por partes abaixo)
crontab -l 2>/dev/null;	Lista tarefas atuais, se não tiver suprime o erro.
echo "* * * * * /usr/local/bin/monitor_site.sh"	Cria uma nova linha de agendamento, executando o monitor.site.sh a cada minuto.
crontab -	Envia a nova lista atualizada para o crontab

Figura 14 – Editor de tarefas *cron.*



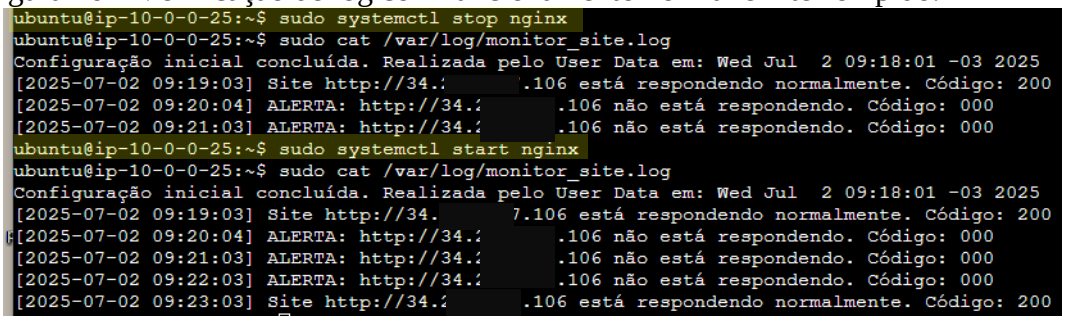
Quarta Etapa

Após a realização das etapas, é verificado inicialmente se a página está sendo servida corretamente pelo *nginx*. Na figura 15 foi realizado a captura de tela do acesso à página *HTML* quando inserido o endereço público de rede da instância EC2.

Figura 15 – Página *HTML* acessada pelo IP público da instância EC2.

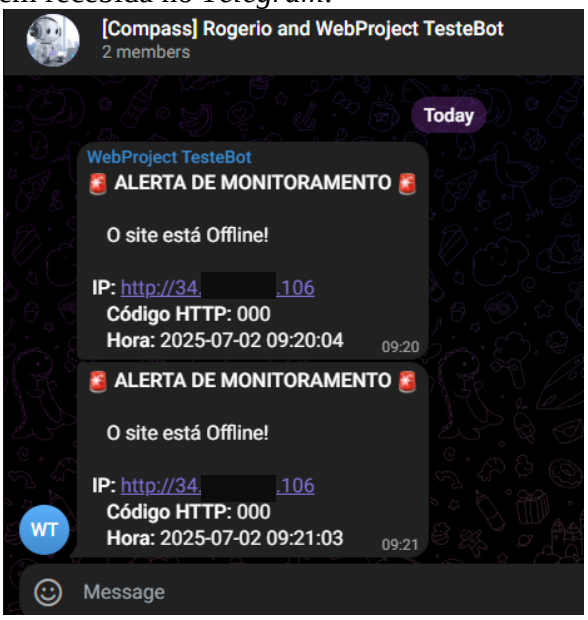
Na sequência foi verificado se o script estava criando os registros (logs) com o estado da página. O próximo teste realizado foi a interrupção intencional do servidor *nginx* para observar a alteração do log (Figura 16) e disparo de mensagem pelo *bot* do *Telegram (*(Figura 17).

Figura 16 – Verificação de log com funcionamento normal e interrompido.



Assim, conforme a figura 16, pode ser observado que o funcionamento normal da página (serviço ativo) o código de estado *HTML* recebido é ‘200’. E quando o serviço é interrompido o código disponibilizado é ‘000’, significando que nenhuma resposta foi recebida.

Figura 17 – Mensagem recebida no *Telegram*.



Desafio Extra

Foi implementado o desafio extra que realizasse a inserção de toda as configurações relacionadas a instalação dos pacotes, *download* do site, acesso as chaves armazenadas no perfil do *IAM*, configuração das permissões, criação de script de monitoramento e logs utilizando de um script inserido no campo *"Dados de Usuário"* na criação da instância EC2 (Figura 18).

E quando inserido o script no campo reservado para essa finalidade a instância funciona da mesma forma após ser inicializada, servindo o site criado e monitorando seu estado de operação.

Figura 18 – Script de automação com “Dados de Usuário”

```
#!/bin/bash
# Atualiza o sistema, instala o Nginx e dependências adicionais.
sudo apt-get update -y; sudo apt-get upgrade -y; sudo apt-get install -y nginx curl cron unzip;

# Inicia e habilita o Nginx
sudo systemctl start nginx
sudo systemctl enable nginx

# Troca o Timezone para Brasil
sudo timedatectl set-timezone America/Sao_Paulo


# ------- Baixa Site para usar no Nginx -------

sudo curl -L "https://github.com/master-rogerio/CompassUol_PB_2025/raw/refs/heads/main/Sprint01/site.zip" -o "/tmp/site.zip"

sudo unzip -q -o "/tmp/site.zip" -d "/var/www/html/"

sudo rm -f "/tmp/site.zip"

# Ajusta Permissões
sudo chown -R www-data:www-data /var/www/html
sudo find /var/www/html -type d -exec chmod 755 {} \;
sudo find /var/www/html -type f -exec chmod 644 {} \;

##-------- Fim Site --------


# --- AWSCLI Config Install ----
# Baixa o instalador oficial AWSCLI
sudo curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "/tmp/awscliv2.zip"

# Descompacta
sudo unzip -q /tmp/awscliv2.zip -d /tmp

# Instala AWSCLI
sudo /tmp/aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --update

sudo rm -rf /tmp/aws /tmp/awscliv2.zip
## --------- FIM Configuração AWSCLI --------


## Cria SCRIPT de monitoramento Telegram
cat <<'EOF' > /usr/local/bin/monitor_site.sh
#!/bin/bash

aws configure set region us-east-1

TELEGRAM_BOT_TOKEN=$(aws ssm get-parameter --name "/telegram/bot_token" --with-decryption --query "Parameter.Value" --output text)
TELEGRAM_CHAT_ID=$(aws ssm get-parameter --name "/telegram/chat_id" --query "Parameter.Value" --output text)

# Configurações do Telegram
BOT_TOKEN="$TELEGRAM_BOT_TOKEN"
CHAT_ID="$TELEGRAM_CHAT_ID"
SITE_URL="http://$(curl -s https://checkip.amazonaws.com/)" # Busca Ip Publico

# Verifica resposta do site
response=$(curl -s -o /dev/null -w "%{http_code}" $SITE_URL)
timestamp=$(date "+%Y-%m-%d %H:%M:%S")

if [ "$response" -eq 200 ]; then
    echo "[$timestamp] Site $SITE_URL está respondendo normalmente. Código: $response" >> /var/log/monitor_site.log
else
    echo "[$timestamp] ALERTA: $SITE_URL não está respondendo. Código: $response" >> /var/log/monitor_site.log

    # Envia mensagem para o Telegram
    message="🚨 *ALERTA DE MONITORAMENTO* 🚨

    O site está Offline!

    *IP:* $SITE_URL
    *Código HTTP:* $response
    *Hora:* $timestamp"

    curl -s -X POST \
        "https://api.telegram.org/bot$BOT_TOKEN/sendMessage" \
        -d chat_id="$CHAT_ID" \
        -d text="$message" \
        -d parse_mode="Markdown" >> /dev/null
fi
EOF


# Tornar o script executável
chmod +x /usr/local/bin/monitor_site.sh

## Configuração do cron job para executar a cada minuto
(crontab -l 2>/dev/null; echo "* * * * * /usr/local/bin/monitor_site.sh") | crontab -

# Mensagem final
echo "Configuração inicial concluída. Realizada pelo User Data em: $(date)" > /var/log/monitor_site.log
```

Bibliografia

AMAZON WEB SERVICES, Inc. *Amazon Elastic Compute Cloud (EC2)*. AWS, 2025. Disponível em: <https://aws.amazon.com/pt/ec2/>. Acesso em: 24 jun. 2025.

AMAZON WEB SERVICES, Inc. *AWS Command Line Interface (CLI)*. AWS, 2025. Disponível em: <https://aws.amazon.com/pt/cli/>. Acesso em: 30 jun. 2025.

AMAZON WEB SERVICES, Inc. *AWS Identity and Access Management (IAM)*. AWS, 2025. Disponível em: <https://aws.amazon.com/pt/iam/>. Acesso em: 30 jun. 2025.

AMAZON WEB SERVICES, Inc. *AWS Systems Manager Parameter Store*. AWS Documentation, 2025. Disponível em: <https://docs.aws.amazon.com/pt_br/systems-manager/latest/userguide/systems-manager-parameter-store.html>. Acesso em: 30 jun. 2025.

AMAZON WEB SERVICES, Inc. *Visão geral da AWS*. AWS Whitepapers, 2025. Disponível em: <https://docs.aws.amazon.com/pt_br/whitepapers/latest/aws-overview/introduction.html>. Acesso em: 27 jun. 2025.

F5, Inc. *EC2 instances for NGINX*. NGINX Documentation, 2024. Disponível em: <https://docs.nginx.com/nginx/deployment-guides/amazon-web-services/ec2-instances-for-nginx/>. Acesso em: 01 jul. 2025.

TELEGRAM MESSENGER LLP. *Telegram Bot API*. Telegram Core, 2025. Disponível em: <https://core.telegram.org/bots/api>. Acesso em: 27 jun. 2025.