

# 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.



A primeira captura de tela mostra a interface 'Suas VPCs (1)' com uma barra de busca e filtros. Abaixo, há uma tabela com uma única linha para a VPC 'minha--compass-vpc', que está no estado 'Available' e tem o acesso público desativado. A segunda captura de tela mostra uma tabela com quatro linhas de sub-redes: duas públicas ('minha-subrede-publica01' e 'minha-subrede-publica02') e duas privadas ('minha-subrede-privada01' e 'minha-subrede-privada02'). Todas as sub-redes estão no estado 'Available' e associadas à VPC 'vpc-07'. Cada sub-rede possui um endereço CIDR IPv4 específico.

[1] VPC

Name	ID da sub-rede	Estado	VPC	Bloquear ac...	CIDR IPv4
minha-subrede-publica01	subnet-...	Available	vpc-07	Desativado	10.0.16/28
minha-subrede-publica02	subnet-...	Available	vpc-07	Desativado	10.0.32/28
minha-subrede-privada01	subnet-...	Available	vpc-07	Desativado	10.0.48/28
minha-subrede-privada02	subnet-...	Available	vpc-07	Desativado	10.0.64/28

[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*.

**Tabelas de rotas (2)** Informações

Q Encontrar tabelas de rotas por atributo ou tag

minha X Limpar filtros

<input type="checkbox"/>	Name	ID da tabela de rotas	Associações explícitas...	Associações de ...	Princ...	VPC
<input type="checkbox"/>	minha-rt-publica01		2 sub-redes	-	Não	vpc- <a href="#">vpc-</a> <a href="#">ia--compass-vpc</a>

[1] Tabela de Rotas.

**Gateways da internet (1)** Informações

Q Encontrar gateways da Internet por atributo ou tag

<input type="checkbox"/>	Name	ID do gateway da Internet	Estado	ID da VPC
<input type="checkbox"/>	meu-gateway	igw- <a href="#">igw-</a> <a href="#">5</a>	Attached	vpc- <a href="#">vpc-</a> <a href="#"> minha--com...</a>

[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.

**▼ Configurações de rede** Informações

**VPC – obrigatório** Informações

[\(minha--compass-vpc\)](#)

**Sub-rede** Informações

subnet-0C [36](#) [minha-subrede-publica01](#)

VPC: vpc-[vpc-](#) Proprietário: 3 Zona de disponibilidade: us-east-1a

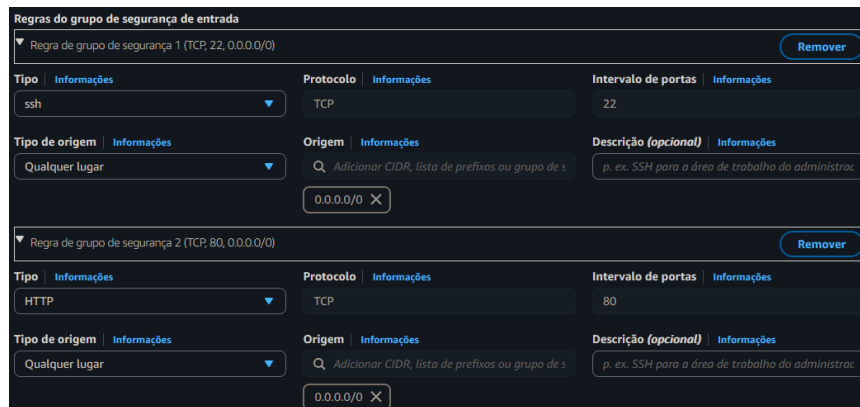
Tipo de zona: zona de disponibilidade Endereços IP disponíveis: 11 CIDR: 10.0.0.16/28

**Atribuir IP público automaticamente** Informações

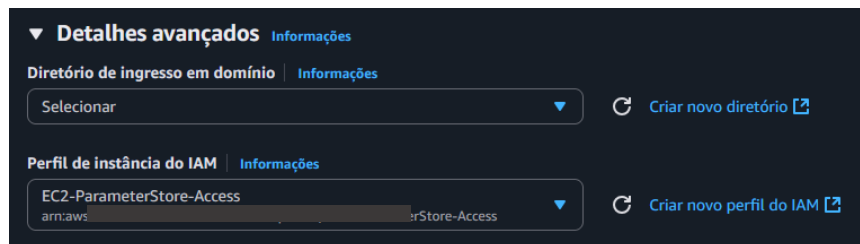
[Habilitar](#)

Taxas adicionais se aplicam quando fora do limite de nível gratuito

[1] Configurações de rede.



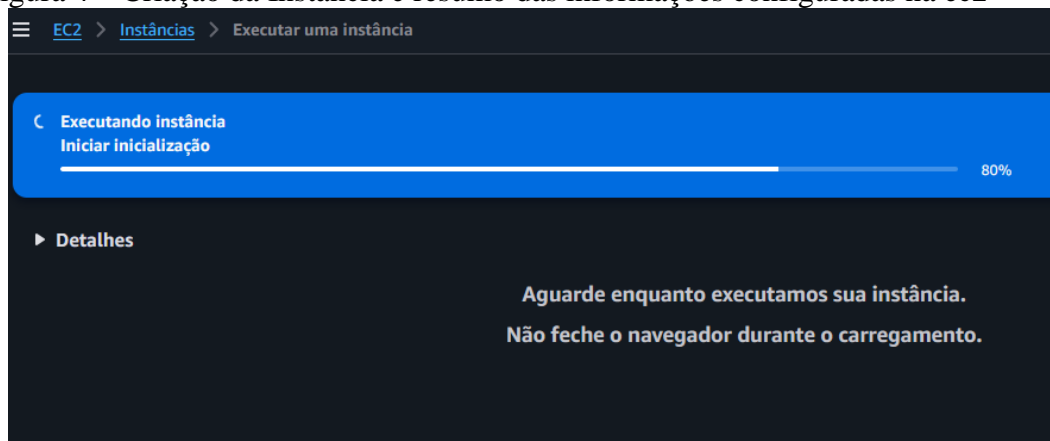
[2] Regras do grupo de segurança.



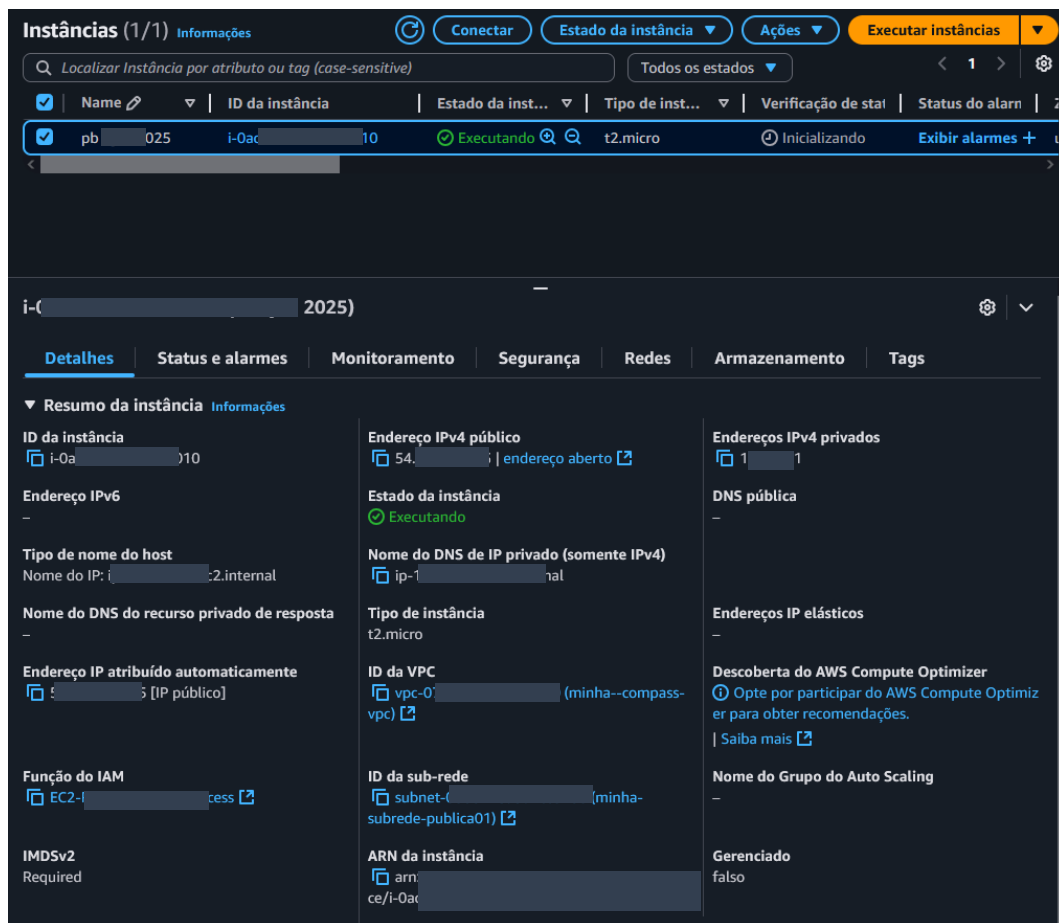
[3] Perfil da instância do IAM para acesso aos segredos armazenados.

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.



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-...:~$ 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-...:~$ 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
<code>systemctl start nginx</code>	Inicializa o serviço Nginx imediatamente;
<code>systemctl enable nginx</code>	Cria um link simbólico do serviço nos diretórios de inicialização automática.
<code>systemctl status nginx</code>	Verifica o status atual do Nginx (Se está ativo).
<code>systemctl is-enabled nginx</code>	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.

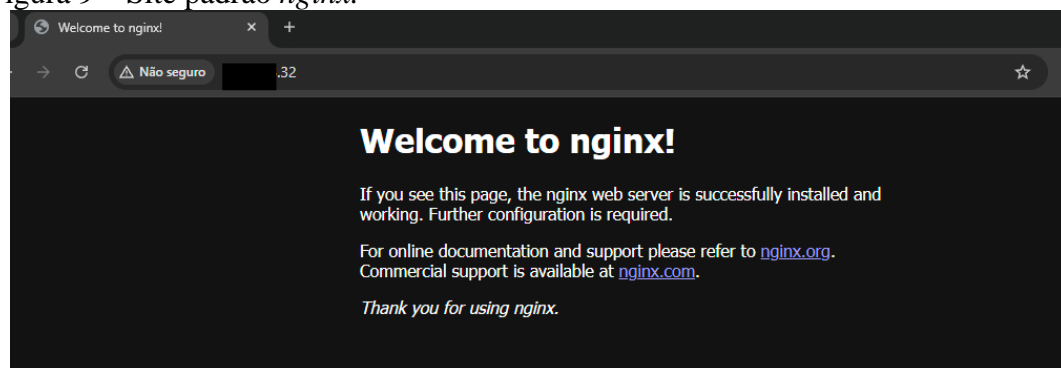
**Figura 8 – Resposta do comando “*systemctl status nginx*”**

```
ubuntu@ip-10-10-10-10:~$ sudo systemctl is-enabled nginx
enabled
ubuntu@ip-10-10-10-10:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-06-29 23:37:38 UTC; 3min 21s ago
     Docs: man:nginx(8)
  Process: 1698 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 1700 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 1728 (nginx)
    Tasks: 2 (limit: 1124)
   Memory: 1.8M (peak: 3.7M)
      CPU: 15ms
   CGroup: /system.slice/nginx.service
           └─1728 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─1731 "nginx: worker process"

Jun 29 23:37:38 ip-10-10-10-10:~$ systemctl status nginx
Jun 29 23:37:38 ip-10-10-10-10:~$ 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-available/*” 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.



### bem vindo!

Esse é um site com a apresentação do projeto que utiliza um servidor **WEB** hospedado em uma instancia **AWS EC2** que utiliza **Linux** como sistema operacional.

Nele foi utilizado o **nginx** como servidor web. É realizado o monitoramento da página em tempo real com script em **Bash** e, caso o servidor não esteja disponível é enviado um alerta pelo **Boot do Telegram**. A execução do script Bash ocorre automaticamente uma vez por minuto, por meio do agendador de tarefas **crontab**.

### eu sou o...



**Rogério Anastácio**  
Estagiário da empresa Compass UOL que está desbravando **AWS** e **Linux**.

### alguns links interessantes

- [Amazon AWS](#)
- [Compass UOL](#)
- [Linux.ORG](#)
- [nginx a HTTP web server](#)
- [Telegram](#)

### sobre

Esse site teve seu design fornecido gratuitamente pelo sítio "HTML5 UP" através da licença do tipo "Creative Commons Attribution 3.0". Sendo ele modificado e adaptado por **Rogério Anastácio**.

[mais sobre o rogerio ....](#)

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/CompassUol_PB_2025/raw/refs/heads/main/Sprint01/site.zip" -o "/tmp/site.zip"	Faz <i>download</i> 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	Altera a propriedade dos arquivos e diretórios dentro de "/var/www/html/" para o usuário "www-data"
find /var/www/html -type d -exec chmod 755 { } \;	Atribui aos diretórios dentro de "/var/www/html/" a propriedade de

	leitura, escrita, execução ao dono e leitura e execução para os outros usuários.
<code>find /var/www/html -type f -exec chmod 644 { } \;</code>	Atribui a todos os arquivos dentro de “/var/www/html/” onde o dono possa realizar a leitura e escrita enquanto os outros usuários possam somente ler.

## 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*.

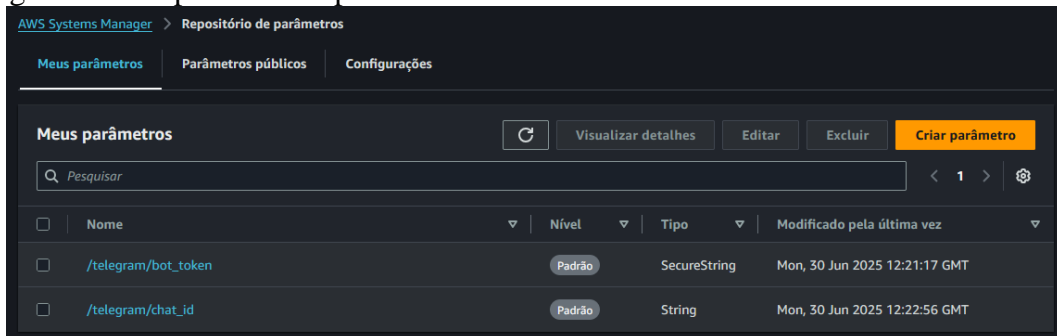
Na figura 12, abaixo, pode ser observado o script usado verificação de disponibilidade, armazenamento de log e disparo de mensagem. O primeiro ponto é que para enviar as notificações no *Telegram* é necessário colocar o código do *token* e o código identificador do *chat*. Por esses dados serem sensíveis 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.

```
1 #!/bin/bash
2
3 #Busca Parametros armazenados no Repositorio de Parametros AWS (IAM)
4 aws configure set region us-east-1
5 TELEGRAM_BOT_TOKEN=$(aws ssm get-parameter --name "/telegram/bot_token" --with-decryption --query "Parameter.Value" --output text)
6 TELEGRAM_CHAT_ID=$(aws ssm get-parameter --name "/telegram/chat_id" --query "Parameter.Value" --output text)
7
8 # Configurações do Telegram
9 BOT_TOKEN="$TELEGRAM_BOT_TOKEN"
10 CHAT_ID="$TELEGRAM_CHAT_ID"
11
12 SITE_URL="http://$(curl -s https://checkip.amazonaws.com/) " # Busca Ip Publico
13
14 response=$(curl -s -o /dev/null -w "%{http_code}" $SITE_URL) # Verifica resposta do site
15
16 timestamp=$(date "+%Y-%m-%d %H:%M:%S") #Variavel que armazena Data e Hora atual
17
18 if [ "$response" -eq 200 ]; then
19     echo "[$timestamp] Site $SITE_URL está respondendo normalmente. Código: $response" >> /var/log/monitor_site.log
20 else
21     echo "[$timestamp] ALERTA: $SITE_URL não está respondendo. Código: $response" >> /var/log/monitor_site.log
22
23     # Envia mensagem para o Telegram
24     message="🚨 *ALERTA DE MONITORAMENTO* 🚨
25
26     O site está Offline!
27
28     *IP:* $SITE_URL
29     *Código HTTP:* $response
30     *Hora:* $timestamp"
31
32     curl -s -X POST \
33         "https://api.telegram.org/bot$BOT_TOKEN/sendMessage" \
34         -d chat_id="$CHAT_ID" \
35         -d text="$message" \
36         -d parse_mode="Markdown" >> /dev/null
37 fi
```

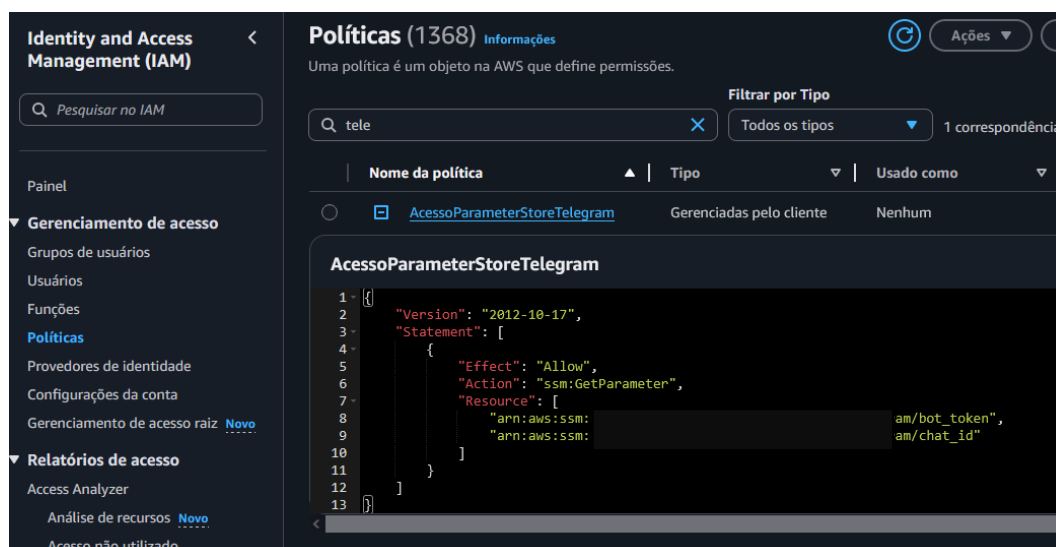
Assim na sequência (figura 13), é apresentado as capturas de telas relacionadas ao serviço de armazenamento seguro das informações, na figura 13[1] é apresentado a forma que os parâmetros do *token* (“*bot\_token*”) e *chat*(“*chat\_id*”) foram armazenados no repositório de parâmetros. Enquanto na figura 13[2] tem-se a política de acesso para acessar as chaves pela EC2.

Figura 13 – Repositório de parâmetros e IAM.



	Nome	Nível	Tipo	Modificado pela última vez
<input type="checkbox"/>	/telegram/bot_token	Padrão	SecureString	Mon, 30 Jun 2025 12:21:17 GMT
<input type="checkbox"/>	/telegram/chat_id	Padrão	String	Mon, 30 Jun 2025 12:22:56 GMT

[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
<code>curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "/tmp/awscliv2.zip"</code>	Faz <i>download</i> do pacote e armazena no diretório /tmp/.
<code>sudo unzip -q /tmp/awscliv2.zip -d /tmp</code>	Descompacta o arquivo na pasta /tmp/
<code>/tmp/aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --update</code>	(Explicado por partes abaixo)
<code>/tmp/aws/install</code>	Local onde se encontra o script para instalação do <i>AWS CLI</i> .
<code>--bin-dir /usr/local/bin</code>	Local onde será colocado o Executável <i>AWS</i> (para acesso pelo terminal)
<code>--install-dir /usr/local/aws-cli</code>	Diretório onde será instalado o <i>AWS CLI</i> .
<code>--update</code>	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
( <i>crontab -l 2&gt;/dev/null; echo "* * * * * /usr/local/bin/monitor_site.sh"</i> ) <i>  crontab -</i>	(Explicado por partes abaixo)
<i>crontab -l 2&gt;/dev/null;</i>	Lista tarefas atuais, se não tiver suprime o erro.
<i>echo "* * * * * /usr/local/bin/monitor_site.sh"</i>	Cria uma nova linha de agendamento, executando o <i>monitor.site.sh</i> a cada minuto.
<i>crontab -</i>	Envia a nova lista atualizada para o <i>crontab</i>

Figura 14 – Editor de tarefas *cron*.

```
GNU nano 7.2 /tmp/crontab.4FADcj/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
* * * * * /scriptMonitor.sh
```

### 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.



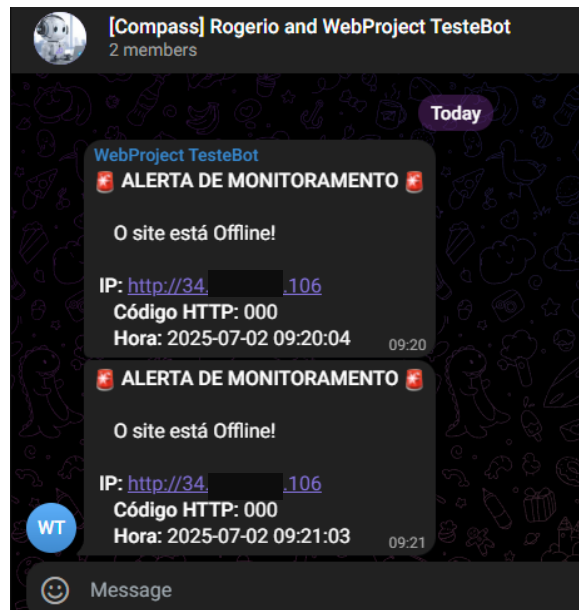
Na sequência foi verificado se o script estava criando os registros (logs) com o estado da página. O próximo teste realizado foi a interrupção intencional do servidor *nginx* para observar a alteração do log (Figura 16) e disparo de mensagem pelo *bot* do *Telegram* ((Figura 17).

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

```
ubuntu@ip-10-0-0-25:~$ sudo systemctl stop nginx
ubuntu@ip-10-0-0-25:~$ sudo cat /var/log/monitor_site.log
Configuração inicial concluída. Realizada pelo User Data em: Wed Jul 2 09:18:01 -03 2025
[2025-07-02 09:19:03] Site http://34.: .106 está respondendo normalmente. Código: 200
[2025-07-02 09:20:04] ALERTA: http://34.: .106 não está respondendo. Código: 000
[2025-07-02 09:21:03] ALERTA: http://34.: .106 não está respondendo. Código: 000
ubuntu@ip-10-0-0-25:~$ sudo systemctl start nginx
ubuntu@ip-10-0-0-25:~$ sudo cat /var/log/monitor_site.log
Configuração inicial concluída. Realizada pelo User Data em: Wed Jul 2 09:18:01 -03 2025
[2025-07-02 09:19:03] Site http://34.: .106 está respondendo normalmente. Código: 200
[2025-07-02 09:20:04] ALERTA: http://34.: .106 não está respondendo. Código: 000
[2025-07-02 09:21:03] ALERTA: http://34.: .106 não está respondendo. Código: 000
[2025-07-02 09:22:03] ALERTA: http://34.: .106 não está respondendo. Código: 000
[2025-07-02 09:23:03] Site http://34.: .106 está respondendo normalmente. Código: 200
```

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*.



Portanto, conforme demonstrado nas imagens, foi possível verificar que o sistema funcionou conforme o esperado, registrando mensagens no log de forma personalizada, de acordo com a disponibilidade da página. Além disso, o envio de notificações pelo *bot* foi corretamente acionado sempre que a página se encontrava indisponível.

## 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”

```
1 #!/bin/bash
2 # Atualiza o sistema, instala o Nginx e dependências adicionais.
3 sudo apt-get update -y; sudo apt-get upgrade -y; sudo apt-get install -y nginx curl cron unzip;
4
5 # Inicia e habilita o Nginx
6 sudo systemctl start nginx
7 sudo systemctl enable nginx
8
9 # Troca o Timezone para Brasil
10 sudo timedatectl set-timezone America/Sao_Paulo
11
12
13 # ----- Baixa Site para usar no Nginx -----
14
15 sudo curl -L "https://github.com/master-rogerio/CompassUol_PB_2025/raw/refs/heads/main/Sprint01/site.zip" -o "/tmp/site.zip"
16
17 sudo unzip -q -o "/tmp/site.zip" -d "/var/www/html/"
18
19 sudo rm -f "/tmp/site.zip"
20
21 # Ajusta Permissões
22 sudo chown -R www-data:www-data /var/www/html
23 sudo find /var/www/html -type d -exec chmod 755 {} \;
24 sudo find /var/www/html -type f -exec chmod 644 {} \;
25
26 ##----- Fim Site -----
27
28
29 # --- AWSCLI Config Install ---
30 # Baixa o instalador oficial AWSCLI
31 sudo curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "/tmp/awscliv2.zip"
32
33 # Descompacta
34 sudo unzip -q /tmp/awscliv2.zip -d /tmp
35
36 # Instala AWSCLI
37 sudo /tmp/aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --update
38
39 sudo rm -rf /tmp/aws /tmp/awscliv2.zip
40 ## ----- FIM Configuração AWSCLI -----
41
42
43 ## Cria SCRIPT de monitoramento Telegram
44 cat <<'EOF' > /usr/local/bin/monitor_site.sh
45 #!/bin/bash
46
47 aws configure set region us-east-1
48
49 TELEGRAM_BOT_TOKEN=$(aws ssm get-parameter --name "/telegram/bot_token" --with-decryption --query "Parameter.Value" --output text)
50 TELEGRAM_CHAT_ID=$(aws ssm get-parameter --name "/telegram/chat_id" --query "Parameter.Value" --output text)
51
52 # Configurações do Telegram
53 BOT_TOKEN="$TELEGRAM_BOT_TOKEN"
54 CHAT_ID="$TELEGRAM_CHAT_ID"
55 SITE_URL="http://$(curl -s https://checkip.amazonaws.com/)" # Busca Ip Publico
56
57 # Verifica resposta do site
58 response=$(curl -s -o /dev/null -w "%{http_code}" $SITE_URL)
59 timestamp=$(date "+%Y-%m-%d %H:%M:%S")
60
61 if [ "$response" -eq 200 ]; then
62     echo "[$timestamp] Site $SITE_URL está respondendo normalmente. Código: $response" >> /var/log/monitor_site.log
63 else
64     echo "[$timestamp] ALERTA: $SITE_URL não está respondendo. Código: $response" >> /var/log/monitor_site.log
65
66     # Envia mensagem para o Telegram
67     message="🚨 *ALERTA DE MONITORAMENTO* 🚨
68
69     O site está Offline!
70
71     *IP:* $SITE_URL
72     *Código HTTP:* $response
73     *Hora:* $timestamp"
74
75     curl -s -X POST \
76         "https://api.telegram.org/bot$BOT_TOKEN/sendMessage" \
77         -d chat_id="$CHAT_ID" \
78         -d text="$message" \
79         -d parse_mode="Markdown" >> /dev/null
80 fi
81 EOF
82
83
84 # Tornar o script executável
85 chmod +x /usr/local/bin/monitor_site.sh
86
87 ## Configuração do cron job para executar a cada minuto
88 (crontab -l 2>/dev/null; echo "* * * * * /usr/local/bin/monitor_site.sh") | crontab -
89
90 # Mensagem final
91 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](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](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.