CLOUD ARCHITECTURE AND SECURITY

CA-1

Sneha Sivaram

X23192054

URL to presentation: https://youtu.be/M0-FFFNkyGg

Introduction

The aim of this project is to demonstrate a detailed approach to secure a web application hosted on a cloud infrastructure. A variety of tools are used here to secure the infrastructure and pluggins for the web application. Furthermore, a thorough monitoring and testing is done on both to check for any possible vulnerabilities as well. And it is concluded with discussion of possible methods that can be carried out in future for securing the web application and the infrastructure.

The cloud infrastructure chosen here is the AWS IAAS platform, where an EC2 instance is created. And then configuring of LAMP stack (Linux, Apache, MariaDB, PHP) is done to deploy Wordpress as the web application. Following this, the security measures are taken in order to secure both the instance and the Wordpress application.

This report will cover the following objectives:
1. Launching the EC2 instance on AWS.
2. Configuring LAMP stack and Wordpress.
3. Securing and monitoring the instance.
4. Securing the Wordpress site.
5. Testing and validation.

1. Launching the EC2 instance

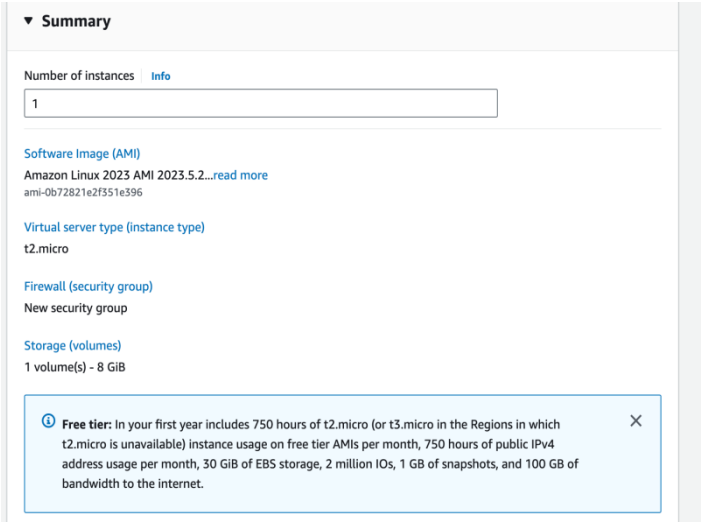
The chosen cloud platform here is AWS, where an EC2 instance is launched by setting up a proper name, choosing the OS, creating a new key pair, instance type and security groups. The ports 443, 80 were configured to accept connections from any source and the port 22 was configured to only accept from the home device's IP.

After launching the instance, a connection was established to the instance from the main device through ssh using this command:

ssh -i "newkey.pem" ec2-user@ec2-54-144-123-185.compute-1.amazonaws.com

After establishing the connection, first it is important to upgrade and update the required packages to ensure that everything is properly set up to date: *sudo dnf update -y*

The following is the details of the instance created:



2. Configuring LAMP stack and Wordpress

LAMP Stack

After establishing the connection to the instance, the initial step was to configure the LAMP stack which is also known as (Linux, Apache, MariaDB, PHP). This LAMP server can be used to host a static website or application which read and write info to the database [1].

After ensuring that all software are up to date using update command the first step was to install the latest versions of Apache webserver. PHP and MariaDB[1]:

sudo dnf install -y httpd wget php-fpm php-mysqli php-json php php-devel
sudo dnf install mariadb105-server

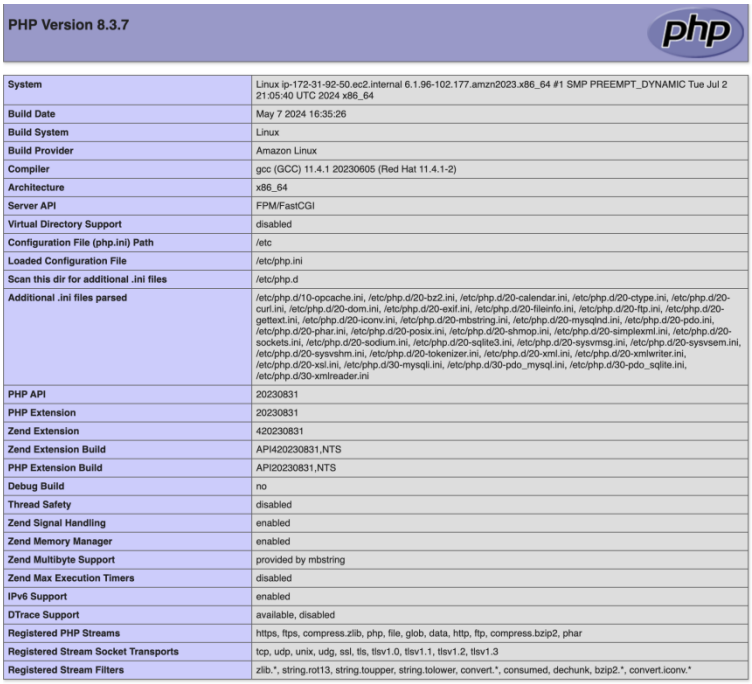
The Apache webserver is then started and a command called systemctl is used to ensure that the Apache starts automatically every time the system boots [1]. After setting up Apache, the public DNS address of the instance is accessed to ensure that it working properly. Furthermore, some of the important files of Apache are stored in a root directory, so the permissions are changed from root to the ec2 user so that it allowed to manipulate any required files and so the user is added to the Apache group and the [1]:

sudo usermod -a -G apache ec2-user
sudo chown -R ec2-user:apache /var/www
sudo chmod 2775 /var/www && find /var/www -type d -exec sudo chmod 2775 {} \;
find /var/www -type f -exec sudo chmod 0664 {} \;

Testing the LAMP server:
The first step is creating a PHP file in the Apache Document root.

echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php

The phpinfo file can be accessed through the web browser using the public DNS address of the instance: *http://54.144.123.185/phpinfo.php*
The PHP file looks like this:



PHP Version 8.3.7

System	Linux ip-172-31-92-50.ec2.internal 6.1.96-102.177.amzn2023.x86_64 #1 SMP PREEMPT_DYNAMIC Tue Jul 2 21:05:40 UTC 2024 x86_64
Build Date	May 7 2024 16:35:26
Build System	Linux
Build Provider	Amazon Linux
Compiler	gcc (GCC) 11.4.1 20230605 (Red Hat 11.4.1-2)
Architecture	x86_64
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/10-opcache.ini, /etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-dom.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-mbstring.ini, /etc/php.d/20-mysqlnd.ini, /etc/php.d/20-pdo.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-posix.ini, /etc/php.d/20-shmop.ini, /etc/php.d/20-simplexml.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-sodium.ini, /etc/php.d/20-sqlite3.ini, /etc/php.d/20-sysvmsg.ini, /etc/php.d/20-sysvsem.ini, /etc/php.d/20-sysvshm.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/20-xml.ini, /etc/php.d/20-xmlwriter.ini, /etc/php.d/20-xsl.ini, /etc/php.d/30-mysqli.ini, /etc/php.d/30-pdo_mysql.ini, /etc/php.d/30-pdo_sqlite.ini, /etc/php.d/30-xmlreader.ini
PHP API	20230831
PHP Extension	20230831
Zend Extension	420230831
Zend Extension Build	API420230831,NTS
PHP Extension Build	API20230831,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
Zend Max Execution Timers	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, convert.*, consumed, dechunk, bzip2.*, convert.iconv.*

For safety reasons the PHP file is deleted in order to protect the sensitive data from any possible unauthorized access.

Securing the Database server (MariaDB)
The server is initially started and a secure installation is done to change the root password and remove anonymous accounts, disable root login and removing the test database [1].
The LAMP stack is now configured and the next step is to host the wordpress site.

Wordpress

After configuring the LAMP server the next step is to host the Wordpress site. Initially several important packages and downloaded and installed [2]:

dnf install wget php-mysqlnd httpd php-fpm php-mysqli mariadb105-server php-json php php-devel -y

Then the latest wordpress is installed using the wget command and the installation package is unzipped to a folder:

wget https://wordpress.org/latest.tar.gz
tar -xzf latest.tar.gz

It is important to have a database for the wordpress site to store data like blog posts, users, comments etc. So, a user is created and a database is assigned and authorized to the user in order to read and save the data [2].

Firstly the mariadb is started and logged into as root with the password set during the lampstack configuration:

sudo systemctl start mariadb httpd

mysql -u root -p

A user is then created along with a password and later on a database is created and assigned to the user after granting the privileges:

CREATE USER 'clouduser'@'localhost' IDENTIFIED BY 'cloud123';

CREATE DATABASE `wordpress-db`;

GRANT ALL PRIVILEGES ON `wordpress-db`. TO "clouduser"@"localhost";*

FLUSH PRIVILEGES;

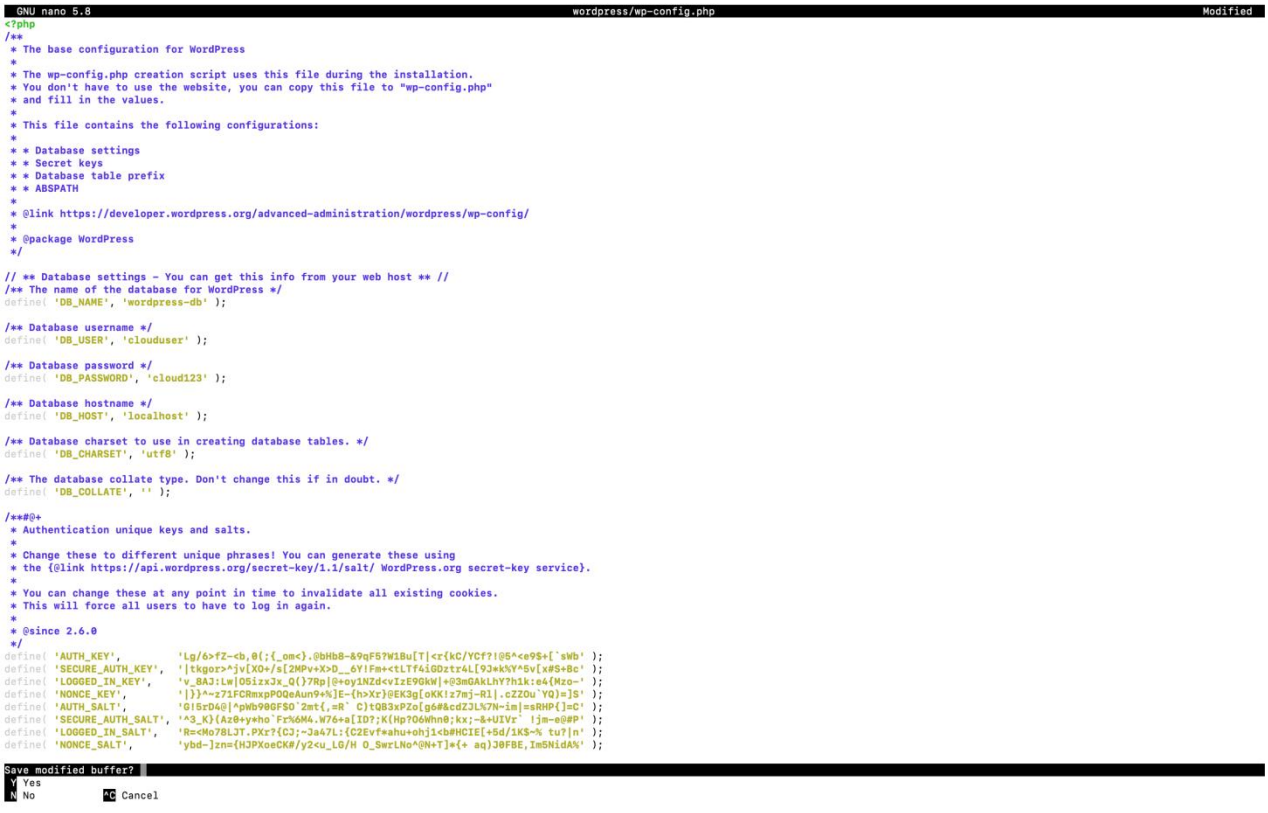
Exit;

The next step is to copy a sample configuration file called wp-config-sample.php which is present in the wordpress installation folder to edit and configure according to requirements [2].

cp wordpress/wp-config-sample.php wordpress/wp-config.php

nano wordpress/wp-config.php

In the .php file, the the database, username and password are replaced with actual ones that were created in the previous steps. And for more safety the section called Authentication Unique Key and salts values are replaced with actual generated values to ensure the safety of website [2].



In the next step, the wordpress files are installed under the Apache Document root:

cp -r wordpress/ /var/www/html/*

To allow the wordpress to use permalinks [2]:

sudo nano /etc/httpd/conf/httpd.conf

Under a section called <Directory "var/www/html"> the line AllowOverride None is changed to AllowOverride All. This is done so that the wordpress works properly [2].

The file permissions are granted to the apache user since some features in wordpress require the write access. These commands were used to grant file ownership, group ownership and change d directory permissions [2].

sudo chown -R apache /var/www
sudo chgrp -R apache /var/www
sudo chmod 2775 /var/www
find /var/www -type d -exec sudo chmod 2775 {} \;
find /var/www -type f -exec sudo chmod 0644 {} \;

Wordpress installation

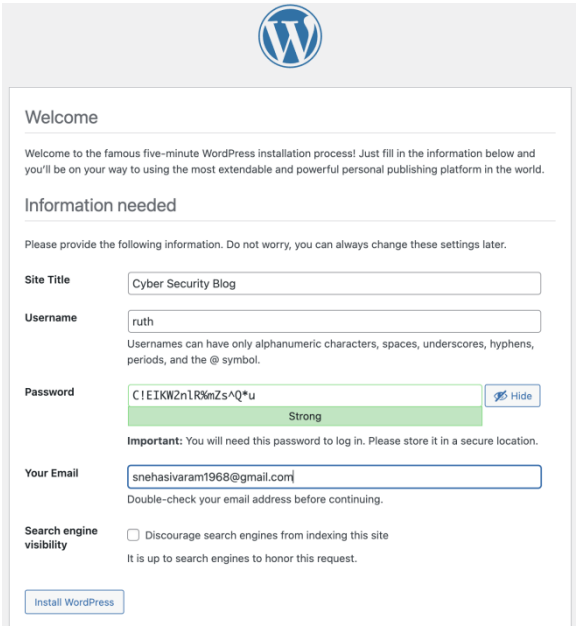
Using chkconfig command to ensure that httpd and database services start at system boot:

sudo chkconfig httpd on && sudo chkconfig mariadb on

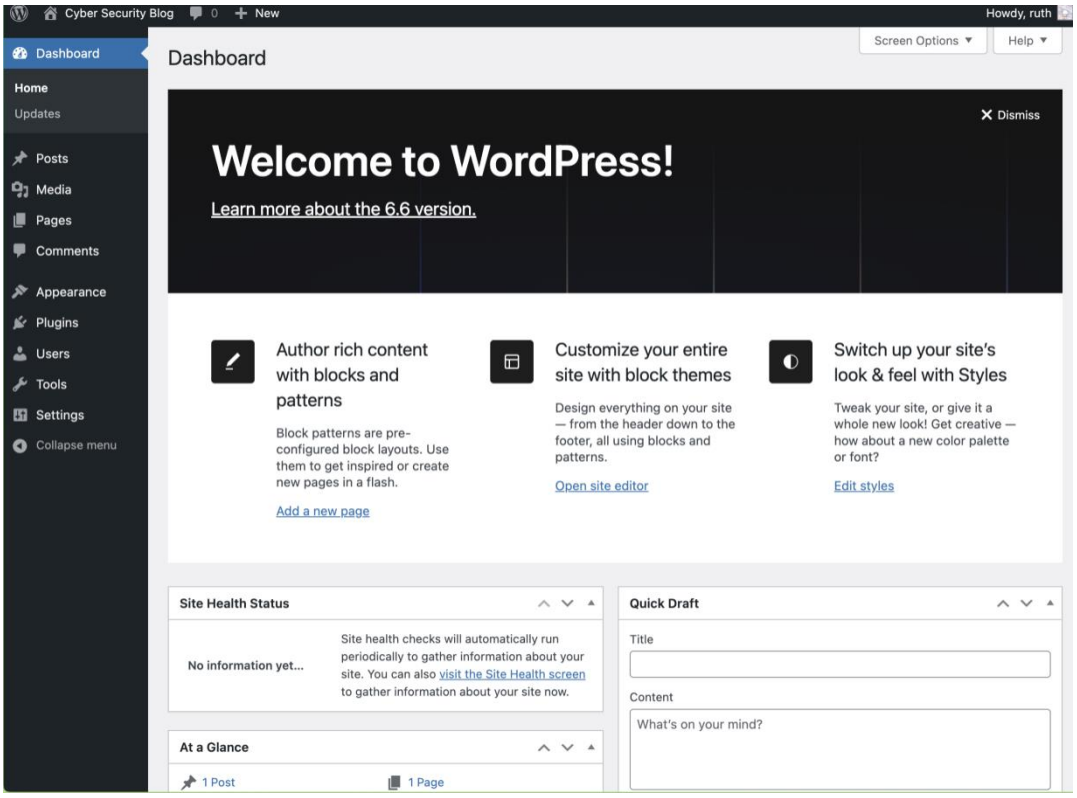
Verifying the database server iand Apache server (httpd) is running:

sudo service mariadb status
sudo service httpd status

After these steps, the public DNS address of the instance is accessed through the browser which will take to the wordpress installation page:



After setting up the user and password. Login to the admin page to access the dashboard.



3. Securing and Monitoring the Instance

The most important step in securing the instance always will be regularly updating the operating system and installed packages up to date. Keeping it always updated and patches to fix any possible vulnerabilities is one of the best practices.

And using SSH key pairs is a better and the best option than using password to connect to the instance. This method increases the security by making it difficult for attackers to gain access through brute force attacks.

Firewalld

Firewalld is a dynamic firewall management tool which is can be customized according to the specific needs. It supports varieties of firewall types, rules and settings which makes it useful for protecting an instance [3].

This tool mainly supports linux based distribution and uses zones and services to manage incoming and outgoing traffic and meets various security requirements [3].

Installation of firewalld:

sudo yum install firewalld

Starting and enabling firewalld:

sudo systemctl start firewalld
sudo systemctl enable firewalld

```
[ec2-user@ip-172-31-47-178 ~]$ sudo yum install firewalld
sudo systemctl start firewalld
sudo systemctl enable firewalld
Last metadata expiration check: 6:36:30 ago on Sat Jul 27 10:52:51 2024.
Dependencies resolved.
================================================================================
 Package                  Architecture  Version                  Repository  Size
================================================================================
Installing:
 firewalld                noarch        1.2.3-1.amzn2023         amazonlinux 452 k
Installing dependencies:
 firewalld-filesystem     noarch        1.2.3-1.amzn2023         amazonlinux  11 k
 gobject-introspection    x86_64        1.73.0-2.amzn2023.0.3    amazonlinux 255 k
 ipset                    x86_64        7.11-1.amzn2023.0.3      amazonlinux  40 k
 ipset-libs               x86_64        7.11-1.amzn2023.0.3      amazonlinux  67 k
 iptables-libs            x86_64        1.8.8-3.amzn2023.0.2     amazonlinux 401 k
 iptables-nft             x86_64        1.8.8-3.amzn2023.0.2     amazonlinux 183 k
 libnetfilter_conntrack   x86_64        1.0.8-2.amzn2023.0.2     amazonlinux  58 k
 libnfnetlink             x86_64        1.0.1-19.amzn2023.0.2    amazonlinux  30 k
 libnftnl                 x86_64        1.2.2-2.amzn2023.0.2     amazonlinux  84 k
 nftables                 x86_64        1:1.0.4-3.amzn2023.0.2   amazonlinux 400 k
 python3-firewall         noarch        1.2.3-1.amzn2023         amazonlinux 357 k
 python3-gobject-base     x86_64        3.42.2-2.amzn2023.0.3    amazonlinux 178 k
 python3-gobject-base-noarch noarch     3.42.2-2.amzn2023.0.3    amazonlinux 154 k
 python3-nftables         x86_64        1:1.0.4-3.amzn2023.0.2   amazonlinux  18 k
Installing weak dependencies:
 libcap-ng-python3        x86_64        0.8.2-4.amzn2023.0.2     amazonlinux  30 k

Transaction Summary
================================================================================
Install  16 Packages

Total download size: 2.7 M
Installed size: 11 M
Is this ok [y/N]: y
Downloading Packages:
(1/16): firewalld-filesystem-1.2.3-1.amzn2023.noarch.rpm  137 kB/s |  11 kB  00:00
(2/16): gobject-introspection-1.73.0-2.amzn2023.0.3.x86_64.rpm  2.5 MB/s | 255 kB  00:00
(3/16): firewalld-1.2.3-1.amzn2023.noarch.rpm  3.9 MB/s | 452 kB  00:00
(4/16): ipset-7.11-1.amzn2023.0.3.x86_64.rpm  911 kB/s |  40 kB  00:00
(5/16): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm  11 MB/s | 401 kB  00:00
(6/16): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm  5.9 MB/s | 183 kB  00:00
(7/16): libcap-ng-python3-0.8.2-4.amzn2023.0.2.x86_64.rpm  1.3 MB/s |  30 kB  00:00
(8/16): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm  2.5 MB/s |  58 kB  00:00
(9/16): ipset-libs-7.11-1.amzn2023.0.3.x86_64.rpm  793 kB/s |  67 kB  00:00
(10/16): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64.rpm  1.3 MB/s |  30 kB  00:00
(11/16): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm  3.3 MB/s |  84 kB  00:00
(12/16): nftables-1.0.4-3.amzn2023.0.2.x86_64.rpm  5.9 MB/s | 400 kB  00:00
(13/16): python3-gobject-base-3.42.2-2.amzn2023.0.3.x86_64.rpm  2.9 MB/s | 178 kB  00:00
(14/16): python3-firewall-1.2.3-1.amzn2023.noarch.rpm  4.6 MB/s | 357 kB  00:00
(15/16): python3-nftables-1.0.4-3.amzn2023.0.2.x86_64.rpm  891 kB/s |  18 kB  00:00
(16/16): python3-gobject-base-noarch-3.42.2-2.amzn2023.0.3.noarch.rpm  2.1 MB/s | 154 kB  00:00
--------------------------------------------------------------------------------
Total                                          6.9 MB/s | 2.7 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : libnftnl-1.2.2-2.amzn2023.0.2.x86_64                   1/16
  Installing       : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64              2/16
  Installing       : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64     3/16
```

Checking the status:

sudo firewall-cmd --get-zones

Allowing services:

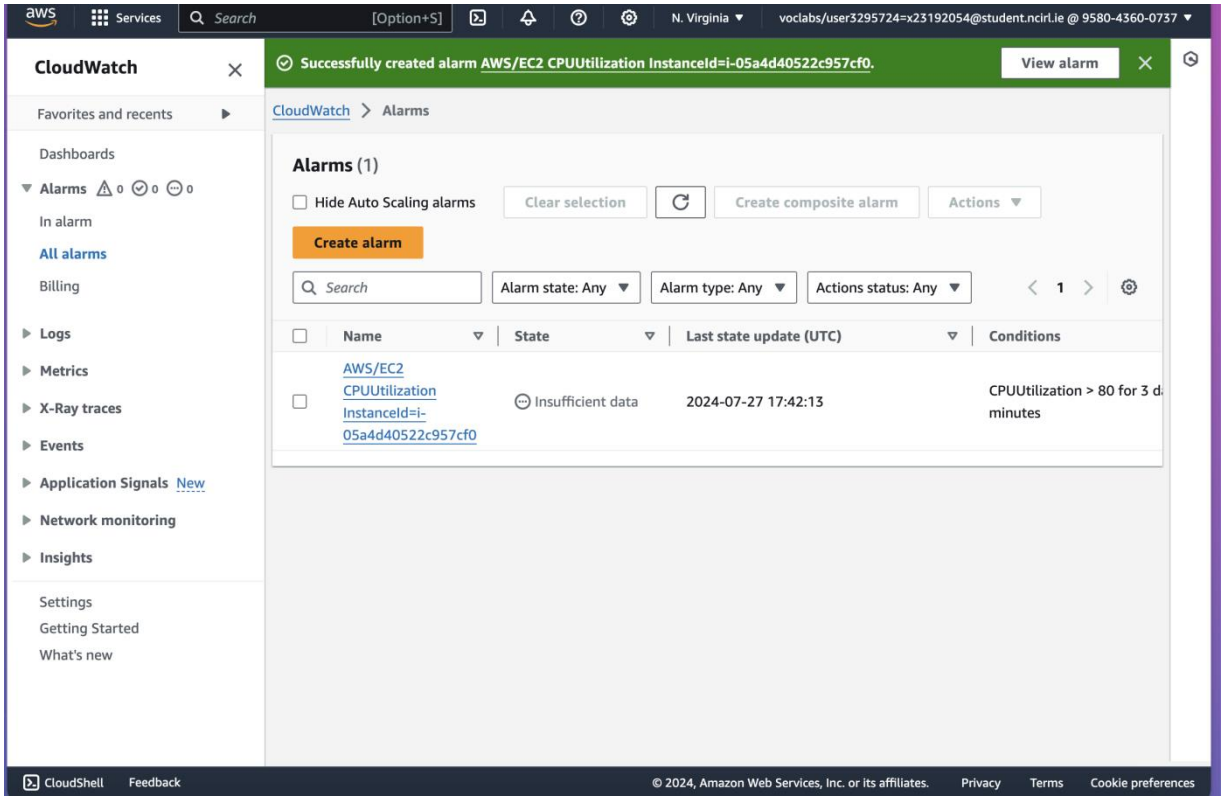
sudo firewall-cmd --zone=public --permanent --add-service=http
sudo firewall-cmd --zone=public --permanent --add-service=https
sudo firewall-cmd --zone=public --permanent --add-service=ssh

```
Complete!
[ec2-user@ip-172-31-47-178 ~]$ sudo systemctl enable firewalld
[ec2-user@ip-172-31-47-178 ~]$ sudo systemctl start firewalld
[ec2-user@ip-172-31-47-178 ~]$ sudo firewall-cmd --zone=public --add-service=http --permanent
sudo firewall-cmd --zone=public --add-service=https --permanent
sudo firewall-cmd --zone=public --remove-service=ssh --permanent
sudo firewall-cmd --reload
success
success
success
success
```

Cloudwatch

AWS has a tool called cloudwatch which is used to monitor instance's health and can log its behaviour. Cloudwatch can track metrics like CPU utilization, disk I/O, network traffics and is also able to send alerts to the user in case any of the metrics crosses the normal allotted threshold range [4].

Its a great tool for monitoring the health of instance and study its behaviour.



So this alarm will be monitoring the instance and will report quickly in case anything abnormal occurs.

4. Securing the Wordpress site

Maintaining the security of website requires multiple techniques such as generating a SSL certificate and installing plugins etc. It is important to do this so that the website is secure from any possible attacks.

Generating a SSL/TLS certificate for the website

This SSL/TLS encryption ensures that the data transmitted between the user and the server is completely encrypted without any form of eavesdropping.So here, this tool called certbot was used to generate the certificate.

Certbot is a tool provided by the EEF (Electronic Frontier Foundation) which automates the process

of generating and renewing the 'Let's Encrypt' SSL certificates [5]. Its completely free to use and very useful to have a secure website.

Installation of certbot:

sudo dnf install certbot python3-certbot-apache -y

Obtaining the SSL certificate:

sudo certbot --apache

There will series of prompts asking for details like domain name, email address etc. For the public DNS of the instance where the wordpress is installed, a domain name was assigned to it using the online platform GoDaddy which was used for generating the SSL certificate.

Plugins for Wordpress:

1. Limit Login Attempts Reloaded

This plugin limits the number of login attempts from one single IP address thereby preventing brute force attacks [6]. Its basically easy to install and setup in the wordpress plugin section. It can be configured to set the number of allowed attempts and other metrics [6].

2. Login Lockdown

This plugin records IP addresses and timestamps of failed login attempts and locks down login if there are too many bad login attempts [7]. This can be easily set up as plugin and configured to set the number of login attempts and lockdown duration as well [7].

3. Akismet

This plugin was preinstalled in the wordpress, when the site was first hosted. It is known to be a powerful anti-spam plugin that checks comments and contact form submissions against a global database of spam [8]. This plugin is activated by obtaining the API key from the official website.

4. Solid Security Basic

Its a great plugin which has a wide range of features such as two factor authentication, file change detection and security logs [9]. It can be customized to specific needs and the 2FA authentication can be assigned to specific roles which are required as well [9]. It is easy to install and activate.

5. Updraft plus

Its a backup plugin that allows to schedule backups at specific times. It is very useful and greatly helps in case anything happens to website [10]. It also has the features of storing backups in remote and safer locations [10].

6. Wordfence Security

Its a very useful plugin which provides complete security solutions such as a firewall, malware scanner, traffic and login attempts monitoring etc. It has a separate dashboard and has a wide variety

of features that can be used. It is easy to install and set up the security features [11].

5. Testing of wordpress

After implementing all these plugins, it is important to always the monitor the website. Since the plugins may not always guarantee a hundred percent safety. So, some of the important tools were used as part of testing the website to check for any possible issues that could be noted down.

Here, two tools were mainly chosen: Nikto and Nmap. These two tools were used in a virtual machine Kali Linux, and was able to yield some important results regarding the website.

Nikto:
It is a powerful web server scanning tool that tests for various security issues, outdated software, insecure files and other problems [12].

The command to use nikto in kali linux:

nikto -h http://54.144.123.185

The following results were found:

It was able to detect HTTP Trace method enabled which must be disabled to prevent any Cross-site Scripting attack. And files containing sensitive data related to the website were detected which must be restricted of any access.

The website can be protected by regularly monitoring and testing with tools to find any possible issues.

Nmap:
Nmap is a powerful network scanning tool which is used to discover open hosts, ports and services to detect any possible vulnerabilities [13]. It can also detect the software versions and is widely used by security professionals for network scanning.

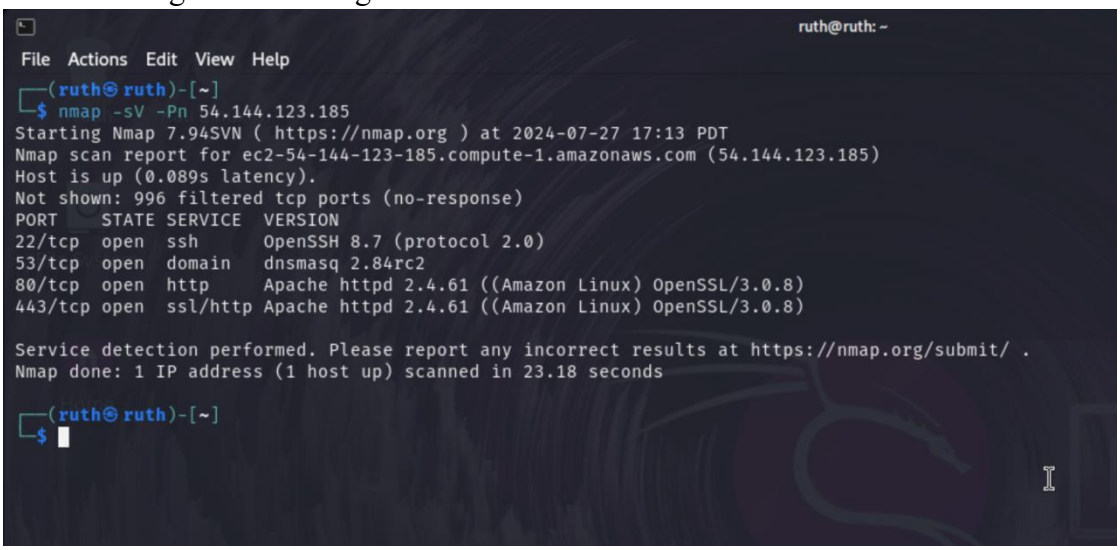
The command to use Nmap in Kali Linux is:

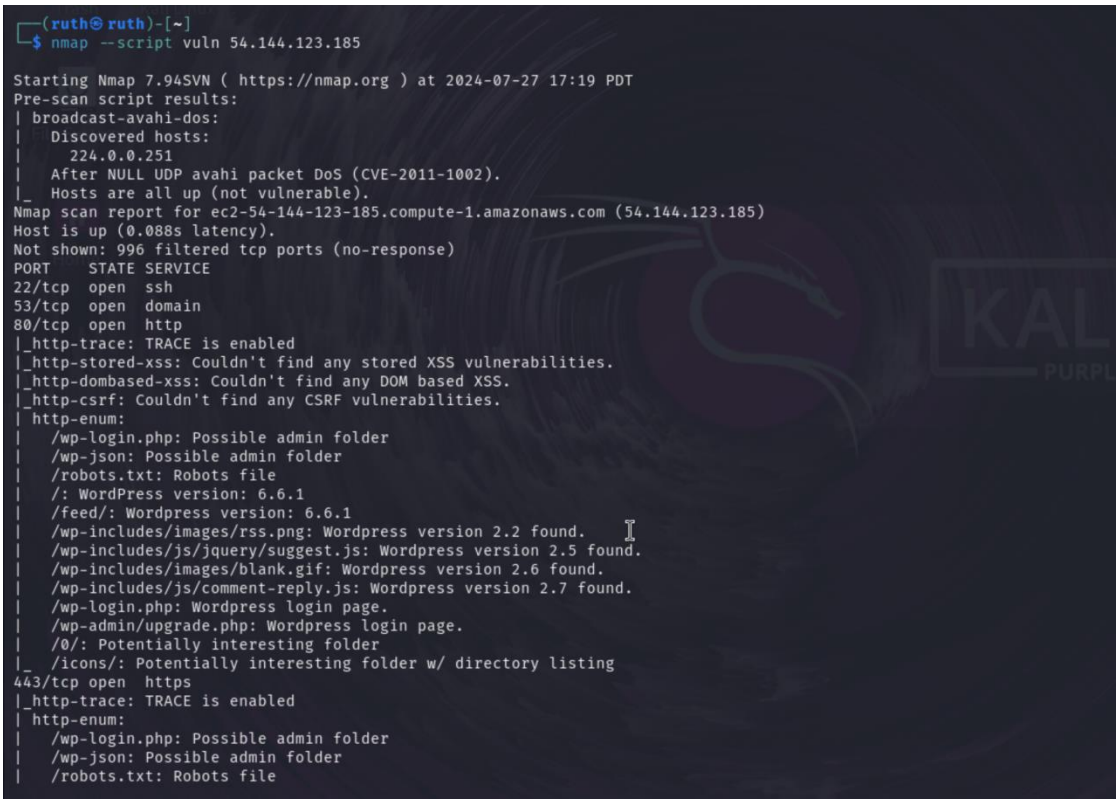
nmap -sV -Pn 54.144.123.185

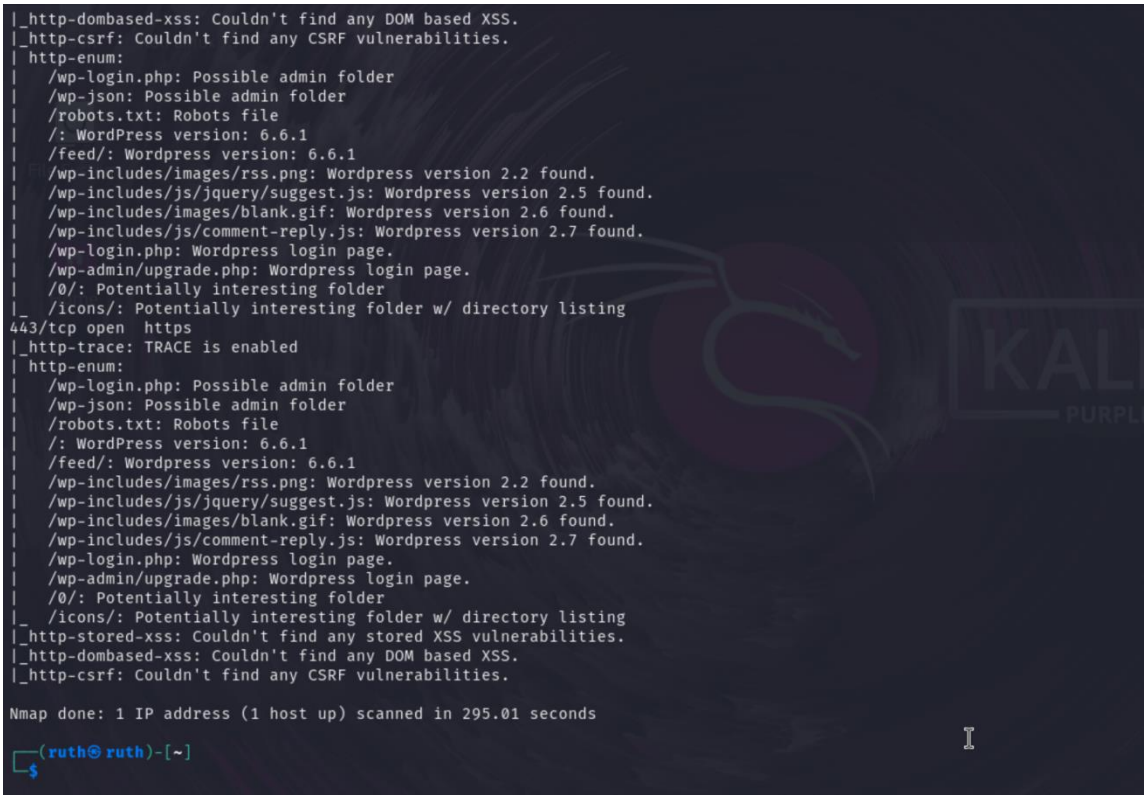
For scanning vulnerabilities:

nmap --script vuln 54.144.123.185

The following results were generated:







It was able to discover the open ports: 22, 53, 80 and 443. And it also detected that TRACE was enabled on both port 80 and port 443. Some of the admin folders were also detected.

These issues can be resolved by restricting access to certain ports, and deleting unwanted files. And installing more secure plugins as well.

It is important to always monitor the wordpress and instance and check for any unusual behaviour and the software must be always be kept up to date. These are the best practices that could be followed to maintain the security of Instance and the web application. To conclude, more advanced tools can be used to maintain the security of both very well.

REFERENCES:

[1] AWS, “Tutorial: Install a LAMP server on AL2023,” 2023. [Online]. Available: https://docs.aws.amazon.com/linux/al2023/ug/ec2-lamp-amazon-linux-2023.html#test-lamp-server-2023 [Accessed on: July 10, 2024].

[2] AWS, “ Tutorial: Host a WordPress blog on AL2023,” 2023. [Online]. Available: https://docs.aws.amazon.com/linux/al2023/ug/hosting-wordpress-aml-2023.html [Accessed on: July 10, 2024]

[3] Firewalld.org, “Firewalld,” 2023. [Online]. Available: https://firewalld.org/ [Accessed on: July 10, 2024]

[4] AWS, “What is Amazon CloudWatch?,” 2021. [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html [Accessed on: July 10, 2024]

[5] EFF, “Certbot,”. [Online]. Available: https://certbot.eff.org/ [Accessed on: July 10, 2024]

[6] Wordpress, “Limit Login Attempts Reloaded,” 2024. [Online]. Available: https://wordpress.org/plugins/limit-login-attempts-reloaded/ [Accessed on: July 10, 2024]

[7] Wordpress, “Login Lockdown and Protection,” 2024. [Online]. Available: https://wordpress.org/plugins/login-lockdown/ [Accessed on: July 10, 2024]

[8] Wordpress, “Akismet Anti-spam: Spam Protection,” 2024. [Online]. Available: https://wordpress.org/plugins/akismet/ [Accessed on: July 11, 2024]

[9] Wordpress, “Solid Security – Password, Two Factor Authentication, and Brute Force Protection,” 2024. [Online]. Available: https://wordpress.org/plugins/better-wp-security/ [Accessed on: July 11, 2024]

[10] Wordpress, “UpdraftPlus: WP Backup & Migration Plugin,” 2024. [Online]. Available: https://wordpress.org/plugins/updraftplus/ [Accessed on: July 11, 2024]

[11] Wordpress, “Wordfence Security – Firewall, Malware Scan, and Login Security,” 2024. [Online]. Available: https://wordpress.org/plugins/wordfence/ [Accessed on: July 11, 2024]

[12]CIRT.net, “Nikto 2.5,” 2024. [Online]. Available: https://www.cirt.net/Nikto2 [Accessed on: July 11, 2024]

[13] Nmap.org, “Nmap 7.95,” 2024. [Online]. Available: https://nmap.org/ [Accessed on: July 11, 2024]

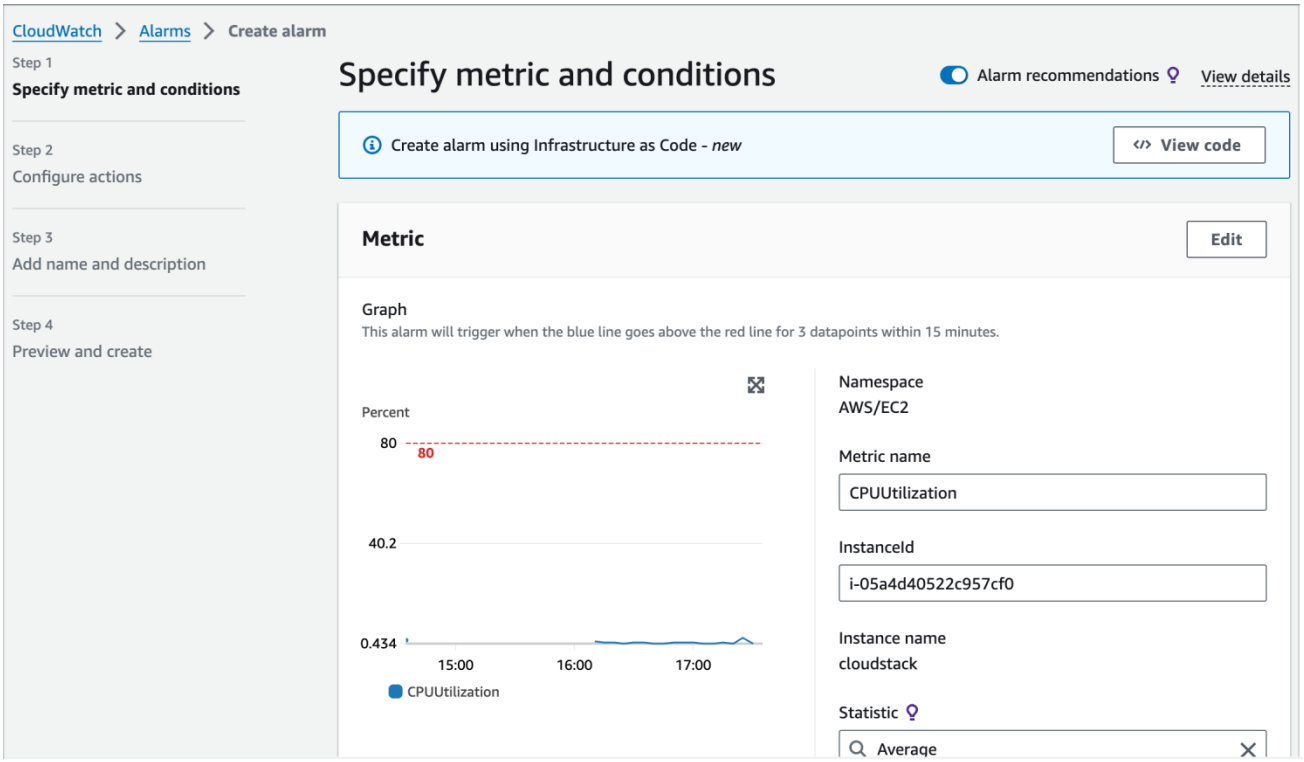
APPENDIX

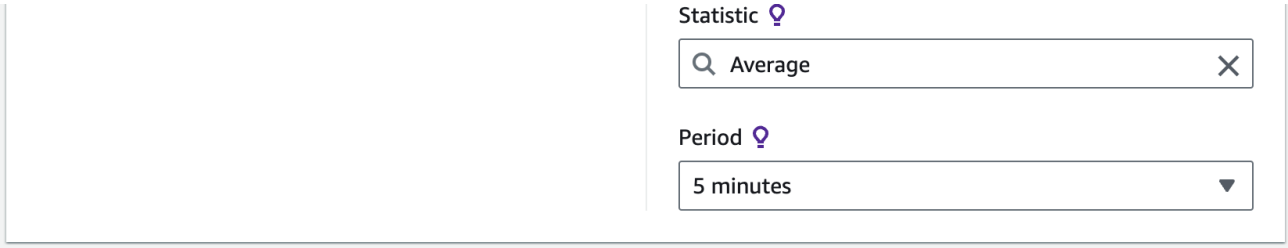
Setting up MariaDB

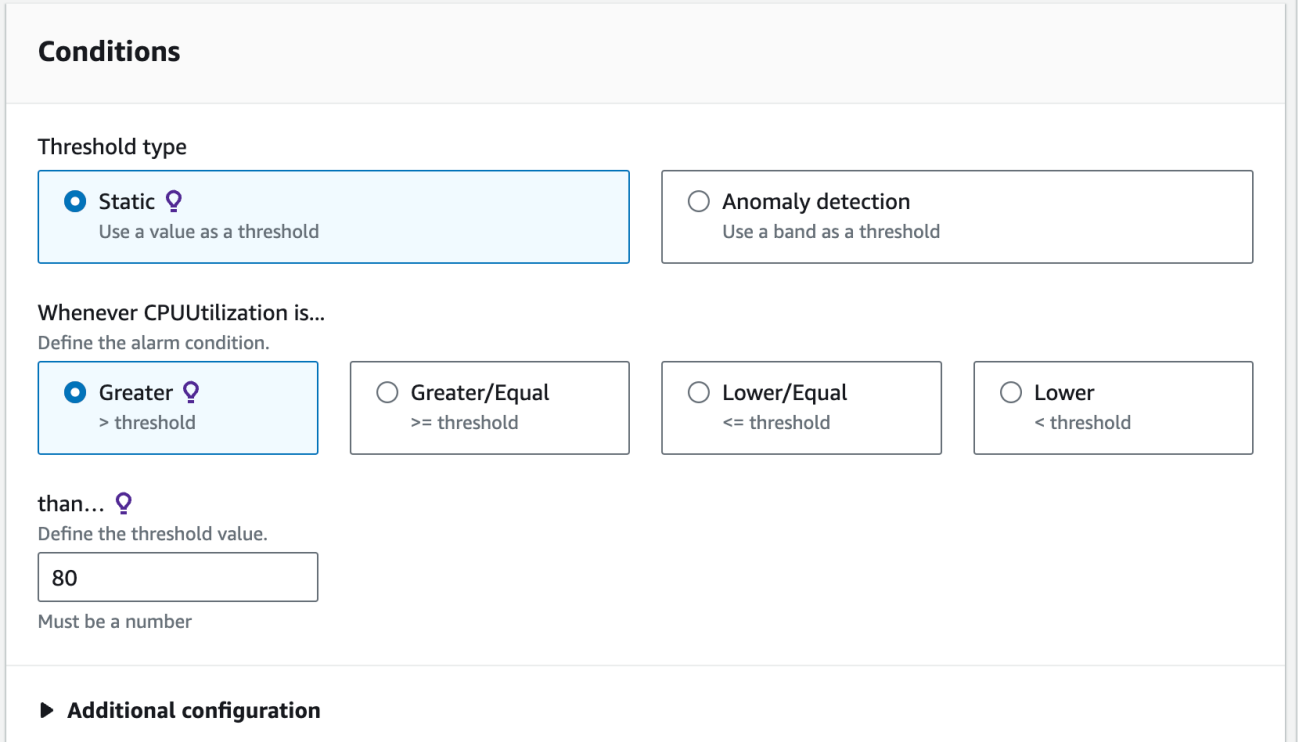
```
 You already have your root account protected, so you can safely answer 'n'.

[Change the root password? [Y/n] y
[New password:
[Re-enter new password:
 Password updated successfully!
 Reloading privilege tables..
  ... Success!


 By default, a MariaDB installation has an anonymous user, allowing anyone
 to log into MariaDB without having to have a user account created for
 them.  This is intended only for testing, and to make the installation
 go a bit smoother.  You should remove them before moving into a
 production environment.

[Remove anonymous users? [Y/n] y
  ... Success!

 Normally, root should only be allowed to connect from 'localhost'.  This
 ensures that someone cannot guess at the root password from the network.

[Disallow root login remotely? [Y/n] y
  ... Success!

 By default, MariaDB comes with a database named 'test' that anyone can
 access.  This is also intended only for testing, and should be removed
 before moving into a production environment.

[Remove test database and access to it? [Y/n] y
  - Dropping test database...
  ... Success!
  - Removing privileges on test database...
  ... Success!

 Reloading the privilege tables will ensure that all changes made so far
 will take effect immediately.

[Reload privilege tables now? [Y/n] y
  ... Success!

 Cleaning up...

 All done!  If you've completed all of the above steps, your MariaDB
 installation should now be secure.

 Thanks for using MariaDB!
 [ec2-user@ip-172-31-92-50 ~]$
```

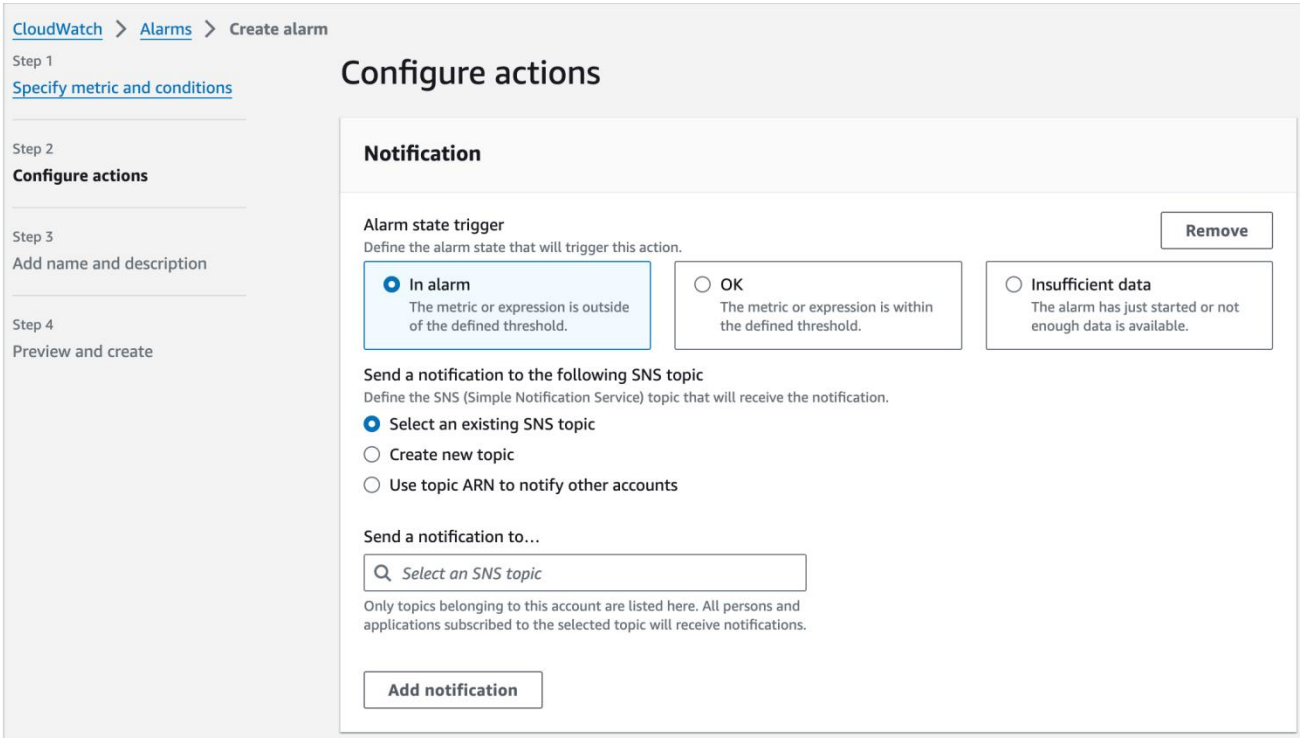
```
[ec2-user@ip-172-31-92-50 ~]$ dnf install wget php-mysqlnd httpd php-fpm php-mysqli mariadb105-server php-json php php-devel -y
 Error: This command has to be run with superuser privileges (under the root user on most systems).
[ec2-user@ip-172-31-92-50 ~]$ tar -xzf latest.tar.gz
[ec2-user@ip-172-31-92-50 ~]$ sudo systemctl start mariadb httpd
[ec2-user@ip-172-31-92-50 ~]$ mysql -u root -p
[Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 13
Server version: 10.5.23-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

[MariaDB [(none)]> CREATE USER 'clouduser'@'localhost' IDENTIFIED BY 'cloud123';
Query OK, 0 rows affected (0.004 sec)

[MariaDB [(none)]> CREATE DATABASE `wordpress-db`;
Query OK, 1 row affected (0.000 sec)

[MariaDB [(none)]> CREATE USER 'clouduser'@'localhost' IDENTIFIED BY cloud123;
 ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'cloud123' at line 1
[MariaDB [(none)]> CREATE USER 'clouduser'@'localhost' IDENTIFIED BY 'cloud123';
 ERROR 1396 (HY000): Operation CREATE USER failed for 'clouduser'@'localhost'
[MariaDB [(none)]> GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "clouduser"@"localhost";
Query OK, 0 rows affected (0.001 sec)

[MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.001 sec)

[MariaDB [(none)]> exit
Bye
```

Setting up Cloudwatch









SSL/TLS certificate generation

```
lines 1-21/21 (END)
[ec2-user@ip-172-31-92-50 ~]$ sudo dnf install certbot python3-certbot-apache -y
Last metadata expiration check: 1:16:30 ago on Mon Jul 22 09:30:59 2024.
Dependencies resolved.
================================================================================
 Package                Arch    Version                      Repository    Size
================================================================================
Installing:
 certbot                noarch  2.6.0-4.amzn2023.0.1         amazonlinux   49 k
 python3-certbot-apache noarch  2.6.0-4.amzn2023.0.1         amazonlinux  287 k
Installing dependencies:
 augeas-libs            x86_64  1.13.0-1.amzn2023.0.2        amazonlinux  408 k
 fontawesome-fonts      noarch  1:4.7.0-11.amzn2023.0.2      amazonlinux  205 k
 mod_ssl                x86_64  1:2.4.59-2.amzn2023          amazonlinux  112 k
 python3-acme           noarch  2.6.0-4.amzn2023.0.1         amazonlinux  161 k
 python3-augeas         noarch  1.1.0-10.amzn2023            amazonlinux   34 k
 python3-certbot        noarch  2.6.0-4.amzn2023.0.1         amazonlinux  677 k
 python3-configargparse noarch  1.7-1.amzn2023               amazonlinux   45 k
 python3-josepy         noarch  1.13.0-6.amzn2023            amazonlinux   61 k
 python3-parsedatetime  noarch  2.6-10.amzn2023              amazonlinux   80 k
 python3-pyOpenSSL      noarch  21.0.0-1.amzn2023.0.2        amazonlinux   92 k
 python3-pyrfc3339      noarch  1.1-16.amzn2023              amazonlinux   19 k
 sscg                   x86_64  3.0.3-76.amzn2023            amazonlinux   45 k
Installing weak dependencies:
 python-josepy-doc      noarch  1.13.0-6.amzn2023            amazonlinux   20 k

Transaction Summary
================================================================================
Install  15 Packages

Total download size: 2.2 M
Installed size: 9.5 M
Downloading Packages:
(1/15): certbot-2.6.0-4.amzn2023.0.1.noarch.rpm 420 kB/s |  49 kB     00:00
(2/15): fontawesome-fonts-4.7.0-11.amzn2023.0.2 1.4 MB/s | 205 kB     00:00
(3/15): augeas-libs-1.13.0-1.amzn2023.0.2.x86_6 2.6 MB/s | 408 kB     00:00
(4/15): python-josepy-doc-1.13.0-6.amzn2023.noa 725 kB/s |  20 kB     00:00
(5/15): mod_ssl-2.4.59-2.amzn2023.x86_64.rpm    1.4 MB/s | 112 kB     00:00
(6/15): python3-augeas-1.1.0-10.amzn2023.noarch 410 kB/s |  34 kB     00:00
(7/15): python3-acme-2.6.0-4.amzn2023.0.1.noarc 1.2 MB/s | 161 kB     00:00
(8/15): python3-configargparse-1.7-1.amzn2023.n 1.2 MB/s |  45 kB     00:00
(9/15): python3-certbot-apache-2.6.0-4.amzn2023 4.0 MB/s | 287 kB     00:00
(10/15): python3-certbot-2.6.0-4.amzn2023.0.1.n 4.6 MB/s | 677 kB     00:00
(11/15): python3-josepy-1.13.0-6.amzn2023.noarc 1.8 MB/s |  61 kB     00:00
(12/15): python3-pyrfc3339-1.1-16.amzn2023.noar 775 kB/s |  19 kB     00:00
(13/15): python3-parsedatetime-2.6-10.amzn2023. 1.4 MB/s |  80 kB     00:00
(14/15): sscg-3.0.3-76.amzn2023.x86_64.rpm      1.9 MB/s |  45 kB     00:00
(15/15): python3-pyOpenSSL-21.0.0-1.amzn2023.0. 1.3 MB/s |  92 kB     00:00
--------------------------------------------------------------------------------
Total                                           4.7 MB/s | 2.2 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : python3-pyrfc3339-1.1-16.amzn2023.noarch               1/15
  Installing       : python3-pyOpenSSL-21.0.0-1.amzn2023.0.2.noarch         2/15
  Installing       : sscg-3.0.3-76.amzn2023.x86_64                          3/15
  Installing       : mod_ssl-1:2.4.59-2.amzn2023.x86_64                     4/15
  Installing       : python3-parsedatetime-2.6-10.amzn2023.noarch           5/15
  Installing       : python3-configargparse-1.7-1.amzn2023.noarch           6/15
  Installing       : python-josepy-doc-1.13.0-6.amzn2023.noarch             7/15
```

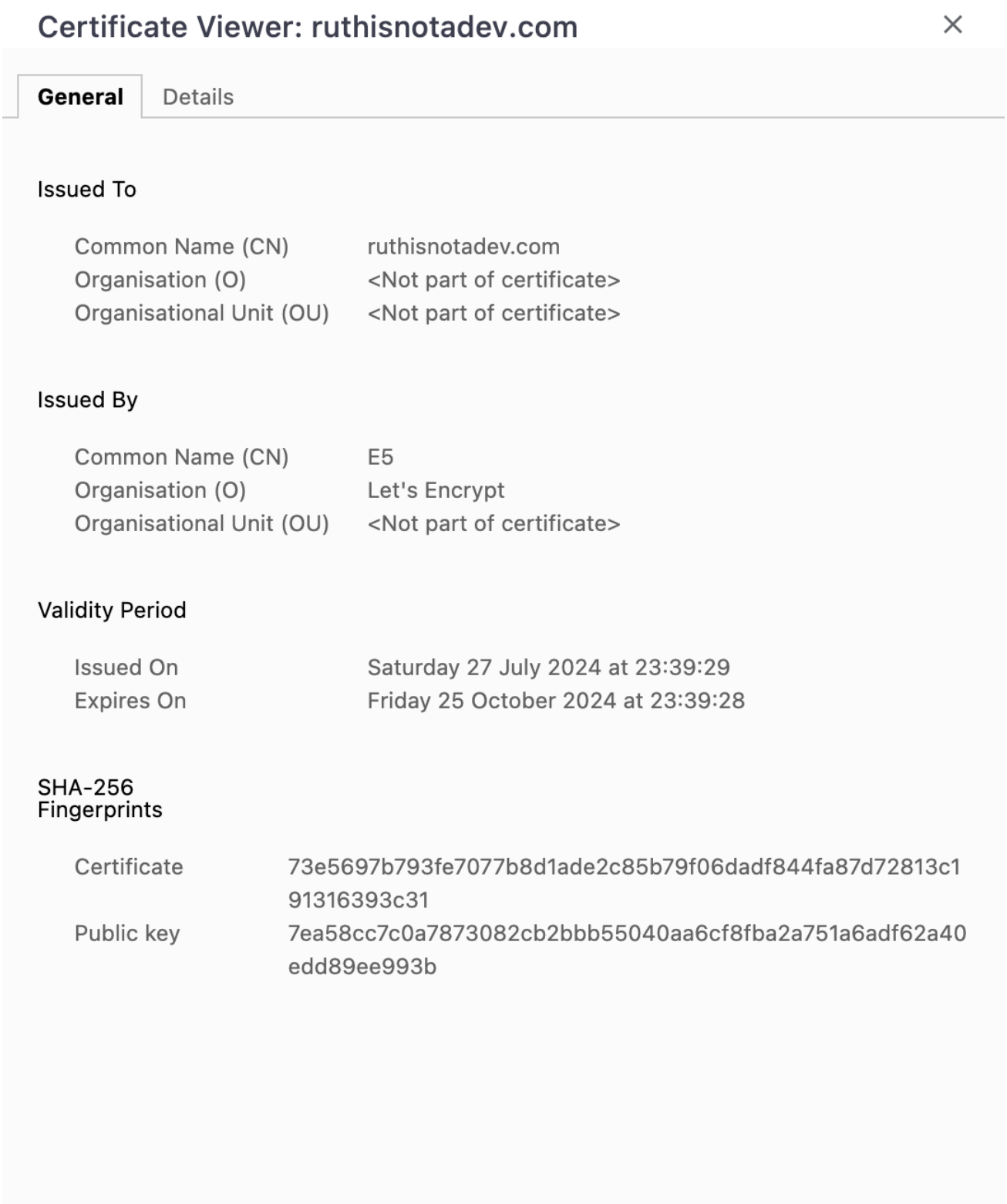
```
[ec2-user@ip-172-31-92-50 ~]$ sudo certbot --apache -v -d ruthisnotadev.com
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator apache, Installer apache
Requesting a certificate for ruthisnotadev.com
Performing the following challenges:
http-01 challenge for ruthisnotadev.com
Waiting for verification...
Cleaning up challenges

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/ruthisnotadev.com/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/ruthisnotadev.com/privkey.pem
This certificate expires on 2024-10-20.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.

Deploying certificate
Created an SSL vhost at /etc/httpd/conf/httpd-le-ssl.conf
Deploying Certificate to VirtualHost /etc/httpd/conf/httpd-le-ssl.conf
Enabling site /etc/httpd/conf/httpd-le-ssl.conf by adding Include to root configuration
Successfully deployed certificate for ruthisnotadev.com to /etc/httpd/conf/httpd-le-ssl.conf
Redirecting vhost in /etc/httpd/conf/httpd.conf to ssl vhost in /etc/httpd/conf/httpd-le-ssl.conf
Congratulations! You have successfully enabled HTTPS on https://ruthisnotadev.com
Subscribe to the EFF mailing list (email: snehasivaram1968@gmail.com).

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
[ec2-user@ip-172-31-92-50 ~]$
```

The certificate:

```
    </Directory>

    ErrorLog /var/log/httpd/your-domain-error.log
    CustomLog /var/log/httpd/your-domain-access.log combined

Include /etc/letsencrypt/options-ssl-apache.conf
SSLCertificateFile /etc/letsencrypt/live/ruthisnotadev.com/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/ruthisnotadev.com/privkey.pem
</VirtualHost>
</IfModule>
[ec2-user@ip-172-31-66-245 conf.d]$ cd
[ec2-user@ip-172-31-66-245 ~]$ cd /etc/letsencrypt/live/ruthisnotadev.com/fullchain.pem
-bash: cd: /etc/letsencrypt/live/ruthisnotadev.com/fullchain.pem: Permission denied
[ec2-user@ip-172-31-66-245 ~]$ sudo cat /etc/letsencrypt/live/ruthisnotadev.com/fullchain.pem
-----BEGIN CERTIFICATE-----
MIIDnDCCAyKgAwIBAgISAy1RvBN7VUEI0hAVXG4+HQoWMAoGCCqGSM49BAMDMDIx
CzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MQswCQYDVQQDEwJF
NTAeFw0yNDA3MjcyMjM5MjlaFw0yNDEwMjUyMjM5MjhaMBwxGjAYBgNVBAMTEXJ1
dGhpc25vdGFkZXYuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEznkZTArY
qN4Qx9sIQToyOvEvSPSVg/3DH1zUxkYYMdayz05CEu0VVPs4crsRLdEBtl5t2xIH
IYzGHTxXzhwSFqOCAiwwggIoMA4GA1UdDwEB/wQEAwIHgDAdBgNVHSUEFjAUBggr
BgEFBQcDAQYIKwYBBQUHAwIwDAYDVR0TAQH/BAIwADAdBgNVHQ4EFgQUsSSeJYgG
l+S7VkBAqi7mZrs6IvEwHwYDVR0jBBgwFoAUnytfzzwhT50Et+0rLMTGcIvS1w0w
VQYIKwYBBQUHAQEESTBHMCEGCCsGAQUFBzABhhVodHRwOi8vZTUuby5sZW5jci5v
cmcwIgYIKwYBBQUHMAKGFmh0dHA6Ly9lNS5pLmxlbmNyLm9yZy8wMwYDVR0RBCww
KoIRcnV0aGlzbm90YWRldi5jb22CFXd3dy5ydXRoaXNub3RhZGV2LmNvbTATBgNV
HSAEDDAKMAgGBmeBDAECATCCAQYGCisGAQQB1nkCBAIEgfcEgfQA8gB3AD8XS0/X
IkdY1B11HIS+DRLtkDd/H4Vq6BG/KIXs+GRuAAABkPaQSMwAAAQDAEgwRgIhAO65
rwl8fTlQFv6AxmvZIXtDkEifgYKFUhDg/nOgOhiCAiEA9JwH9SUdkYFmVT9IpKvg
MFy97XfDFeNgQAh63raCsyEAdwAZmBBxCfDWUi4wgNKeP2S7g24ozPkPUo7u385K
Pxa0ygAAAZD2kEjkAAAEAwBIMEYCIQCD/C/uqmkSg0qoWrXczQ/EODDBzkmoN+Hw
Unhm9hsPMQIhAMC94+rFkXtMQlHkmbmLnFEq4iNZ/FhN2HmkkidejD71MAoGCCqG
SM49BAMDA2gAMGUCMQDhFLHZ0+5niRtFyEDF4UgexJeQ7400yWnDOJNrAEHrPewo
HtsCJ4ok0GYkSReI9l8CMAR4wIBUT+v0lFppcEIImcf1I6qtlODV5QOM/4mMunvl
ygxucNVlCCInhDKLCXxcNg==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEvzCCAj+gAwIBAgIRAIOPbGPOsTmMYgZigxXJ/d4wDQYJKoZIhvcNAQELBQAw
TzELMAkGA1UEBhMCVVMxKTAnBgNVBAoTIEludGVybmV0IFNlY3VyaXR5IFJlc2Vh
cmNoIEdyb3VwMRUwEwYDVQQDEwxJU1JHIFJvb3QgWDEwHhcNMjQwMzEzMDAwMDAw
WhcNMjcwMzEyMjM1OTU5WjAyMQswCQYDVQQGEwJVUzEWMBQGA1UEChMNTGV0J3Mg
RW5jcnlwdDELMAkGA1UEAxMCRTUwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAAQNCzqK
a2GOtu/cX1jnxkJFVktj9mZhSAouWXW0gQI3ULc/FnncmOyhKJdyIBwsz9V8UiBO
VHhbhBRrwJCuhezAUUE8Wod/Bk3U/mDR+mwt4X2VEIiiCFQPmRpM5uoKrNijgfgw
gfUwDgYDVR0PAQH/BAQDAgGGMB0GA1UdJQQWMBQGCCsGAQUFBwMCBggrBgEFBQcD
ATASBgNVHRMBAf8ECDAGAQH/AgEAMB0GA1UdDgQWBBSfK1/PPCFPnQS37SssxMZw
i9LXDTAfBgNVHSMEGDAWgBR5tFnme7bl5AFzgAiIyBpY9umbbjAyBggrBgEFBQcB
AQQmMCQwIgYIKwYBBQUHMAKGFmh0dHA6Ly94MS5pLmxlbmNyLm9yZy8wEwYDVR0g
BAwwCjAIBgZngQwBAgEwJwYDVR0fBCAwHjAcoBqgGIYWaHR0cDovL3gxLmMubGVu
Y3Iub3JnLzANBgkqhkiG9w0BAQsFAAOCAgEAH3KdNEVCQdqk0LKyuNImTKdRJY1C
2uw2SJajuhqkyGPY8C+zzsufZ+mgnhnq1A2KVQOSykOEnUbx1cy637rBAihx97r+
bcwbZM6sTDIaEriR/PLk6LKs9Be0uoVxgOKDcpG9svD33J+G9Lcfv1K9luDmSTgG
6XNFIN5vfI5gs/lMPyojEMdIzK9blcl2/1vKxO8WGCcjvsQ1nJ/Pwt8LQZBfOFyV
XP8ubAp/au3dc4EKWG9MO5zcx1qT9+NXRGdVWxGvmBFRAajciMfXME1ZuGmk3/GO
koAM7ZkjZmleyokP1LGzmfJcUd9s7eeu1/9/eg5XlXd/55GtYjAM+C4DG5i7eaNq
cm2F+yxYIPt6cbbtYVNJCGfHWqHEQ4FYStUyFnv8sjyqU8ypgZaNJ9aVcWSICLOI
E1/Qv/7oKsnZCWJ926wU6RqG1OYPGOi1zuABhLw61cuPVDT28nQS/e6z95cJXq0e
K1BcaJ6fJZsmbjRgD5p3mvEf5vdQM7MCEvU0tHbsx2I5mHHJoABHb8KVBgWp/lcX
GWiWaeOyB7RP+OfDtvi2OsapxXiV7vNVs7fMlrRjY1joKaqmmycnBvAq14AEbtyL
sVfOS66B8apkeFX2NY4XPEYV4ZSCe8VHPrdrERk2wILG3T/EGmSIkCYVUMSnjmJd
VQD9F6Na/+zmXCc=
-----END CERTIFICATE-----
[ec2-user@ip-172-31-66-245 ~]$
```

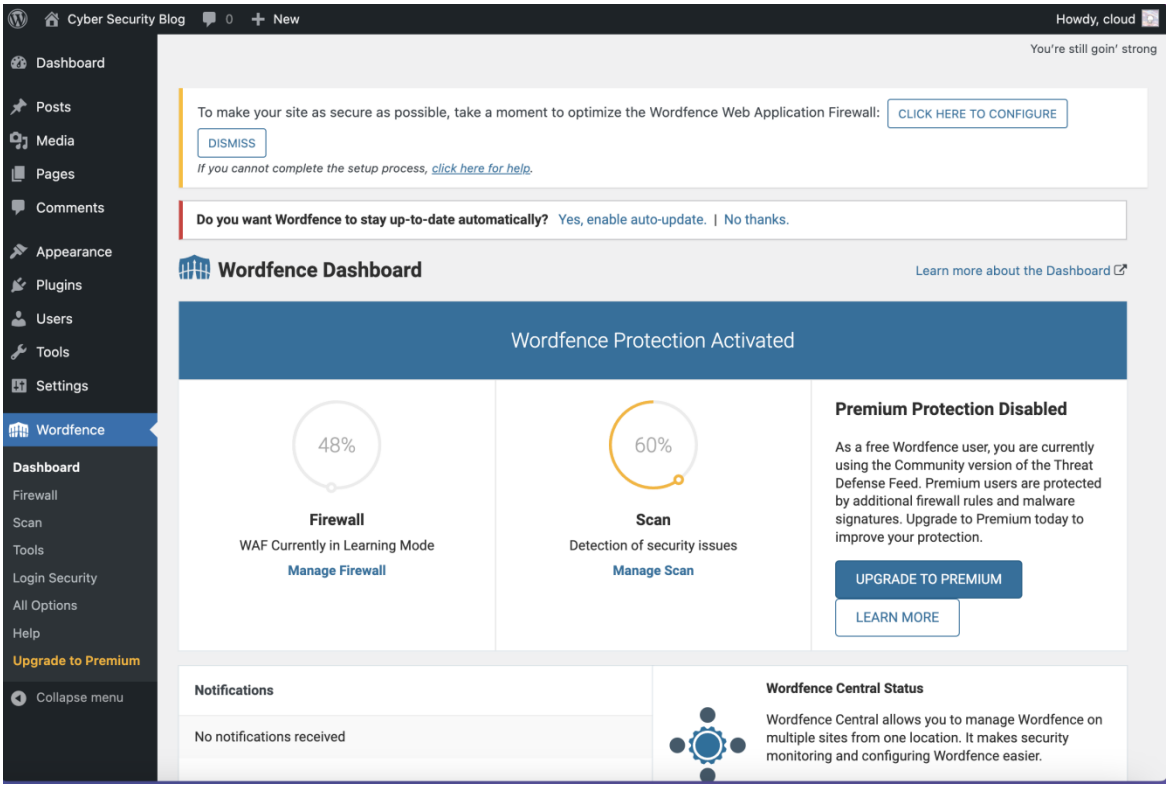
```
[ec2-user@ip-172-31-66-245 ~]$ sudo cat /etc/letsencrypt/live/ruthisnotadev.com/privkey.pem
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgaSMsX7tMkR70AZbJ
83J1cdaXc5T4CYXCHDJ2eOZumLShRANCAATOeRlMCtio3hDH2whBOjI68S9I9JWD
/cMfXNTGRhgx1rLPTkIS7RVU+zhyuxEt0QG2Xm3bEgchjMYdPFfOHBIW
-----END PRIVATE KEY-----
[ec2-user@ip-172-31-66-245 ~]$
```

The Certificate Details:

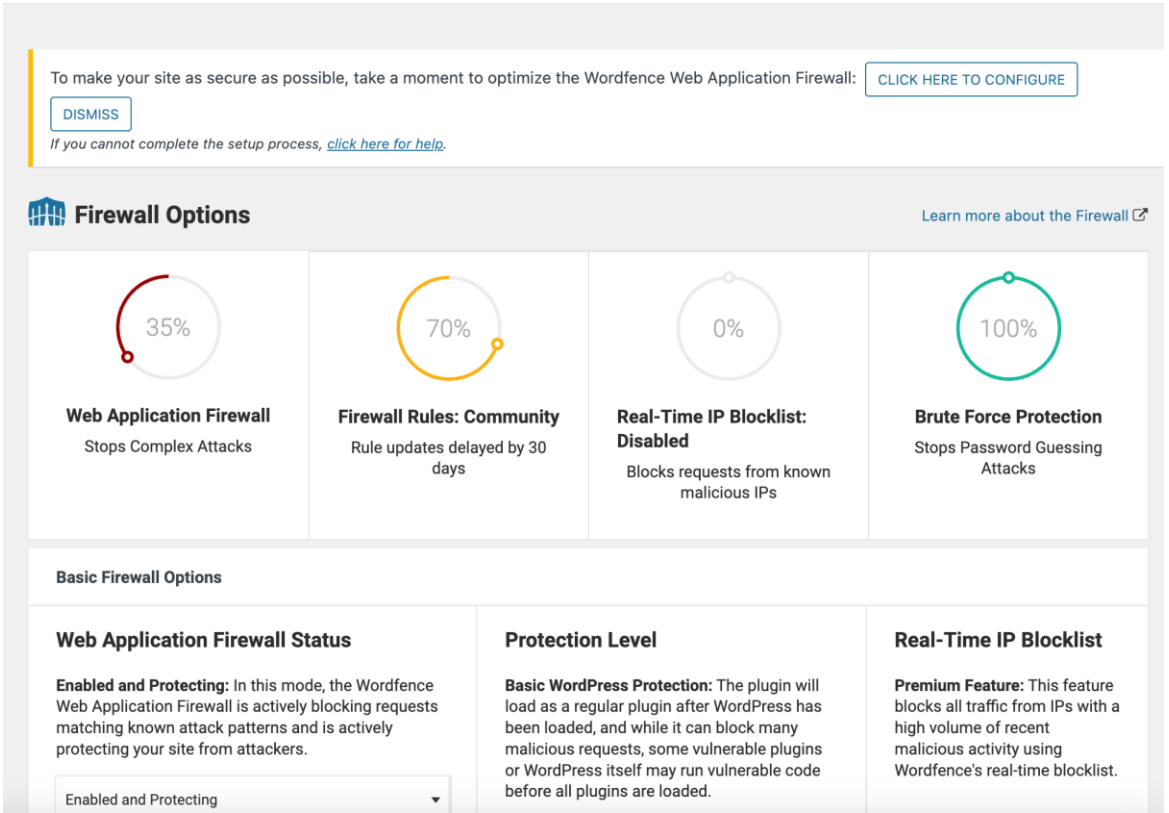


Wordpress plugins

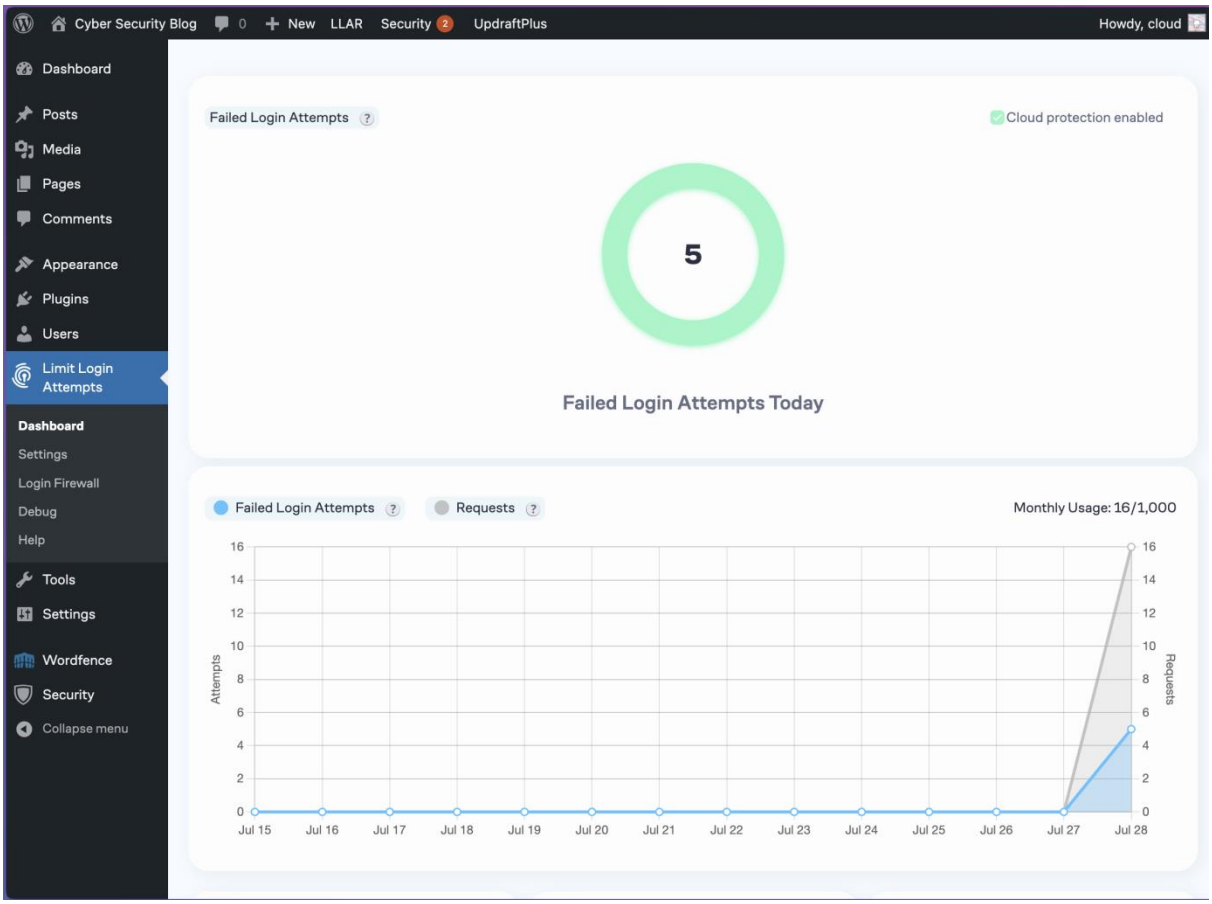
Wordfence

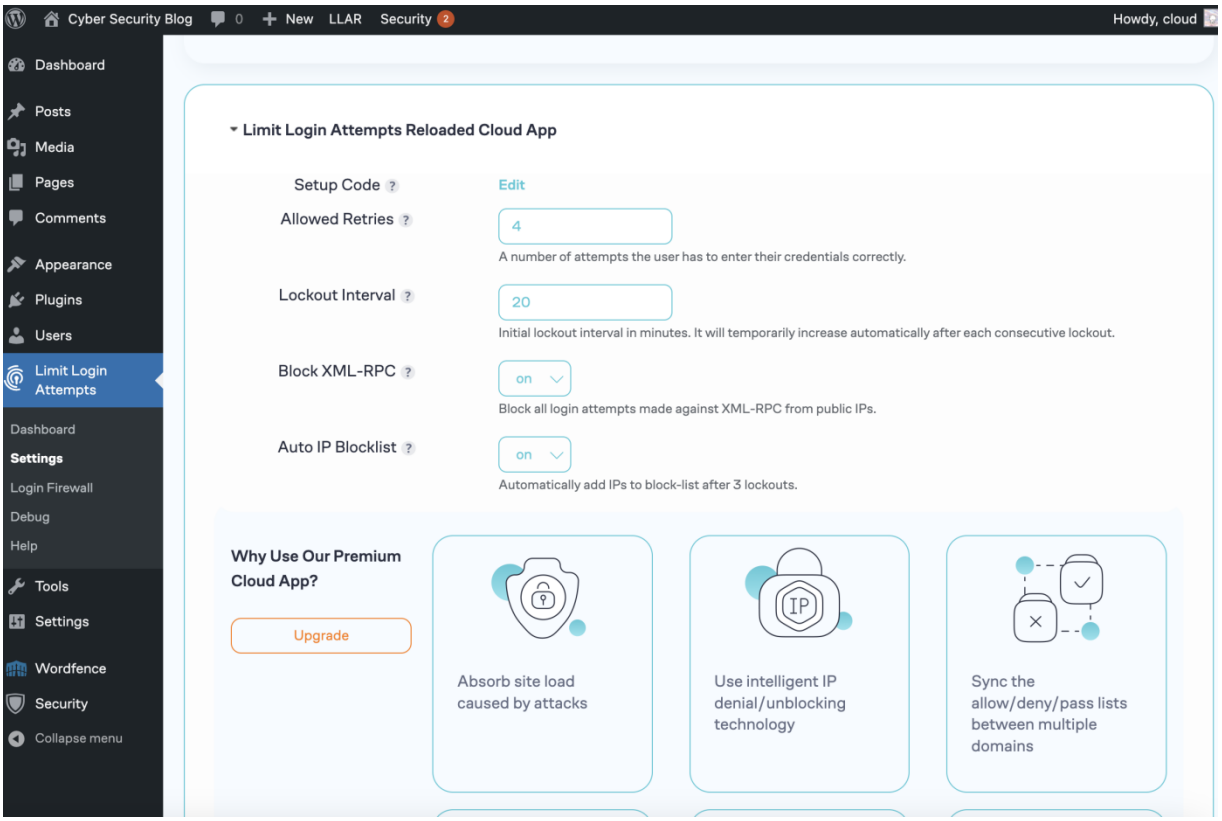


Firewall enabled

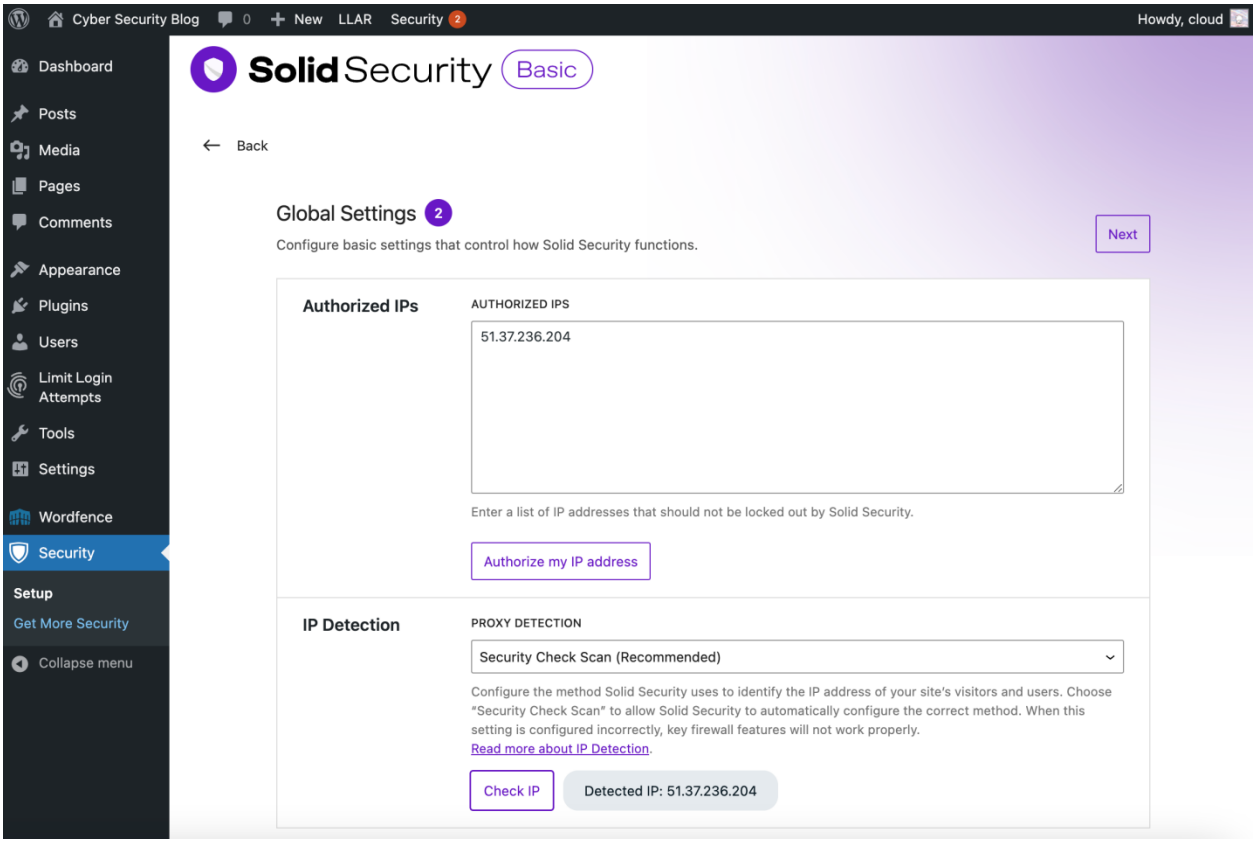


Limit Login Attempts

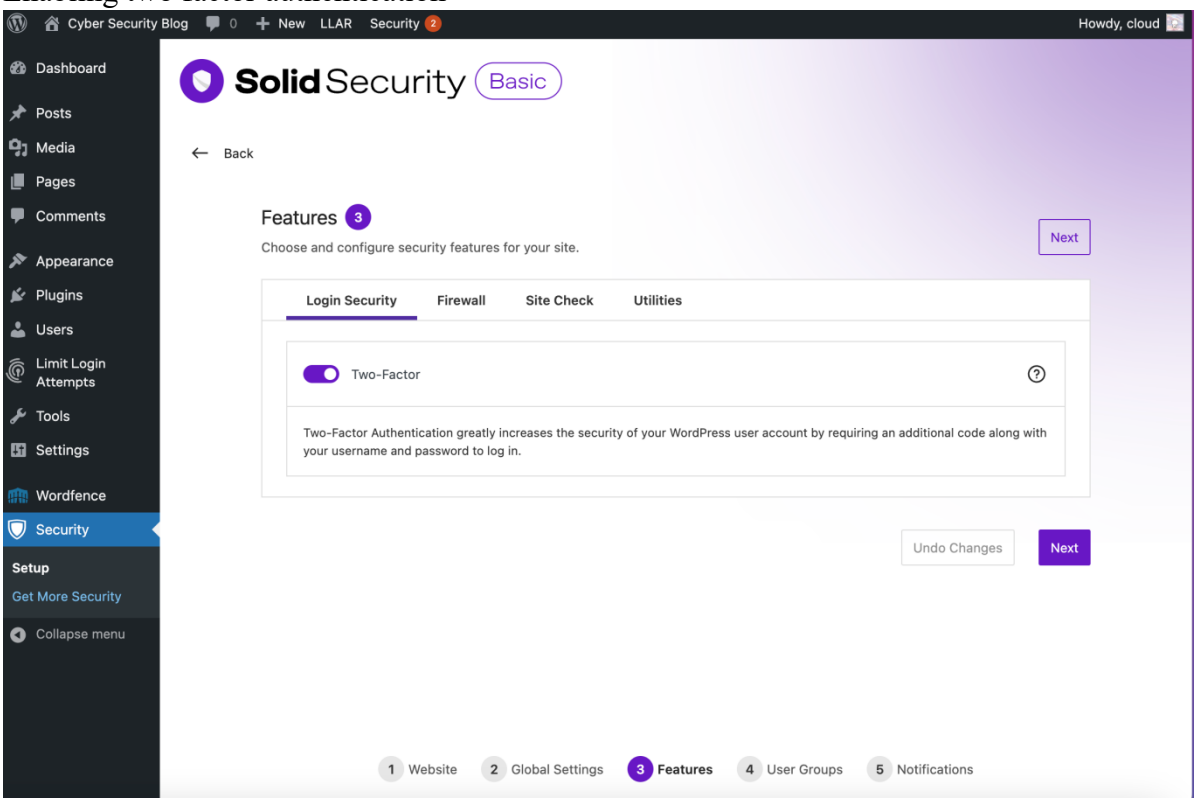




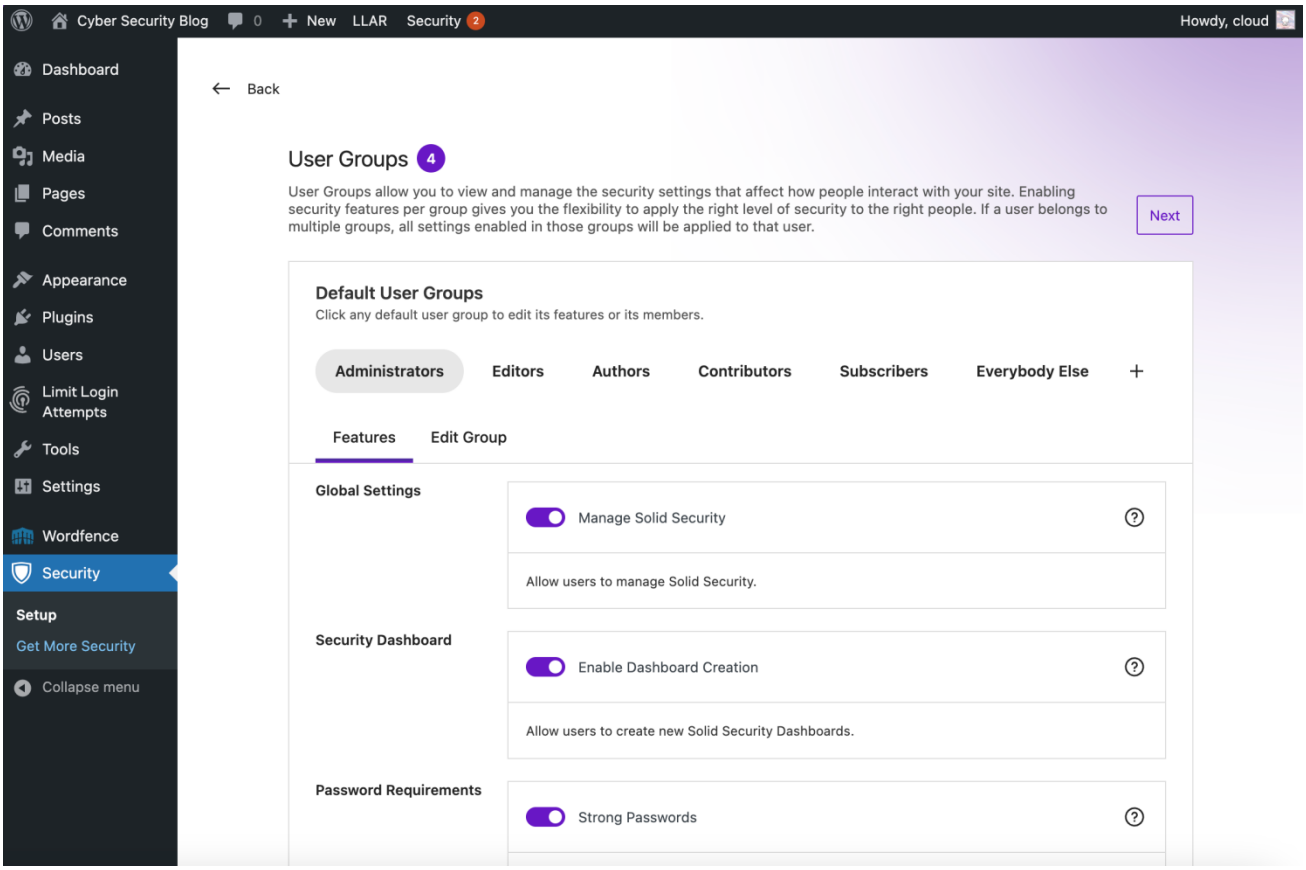
Solid Security



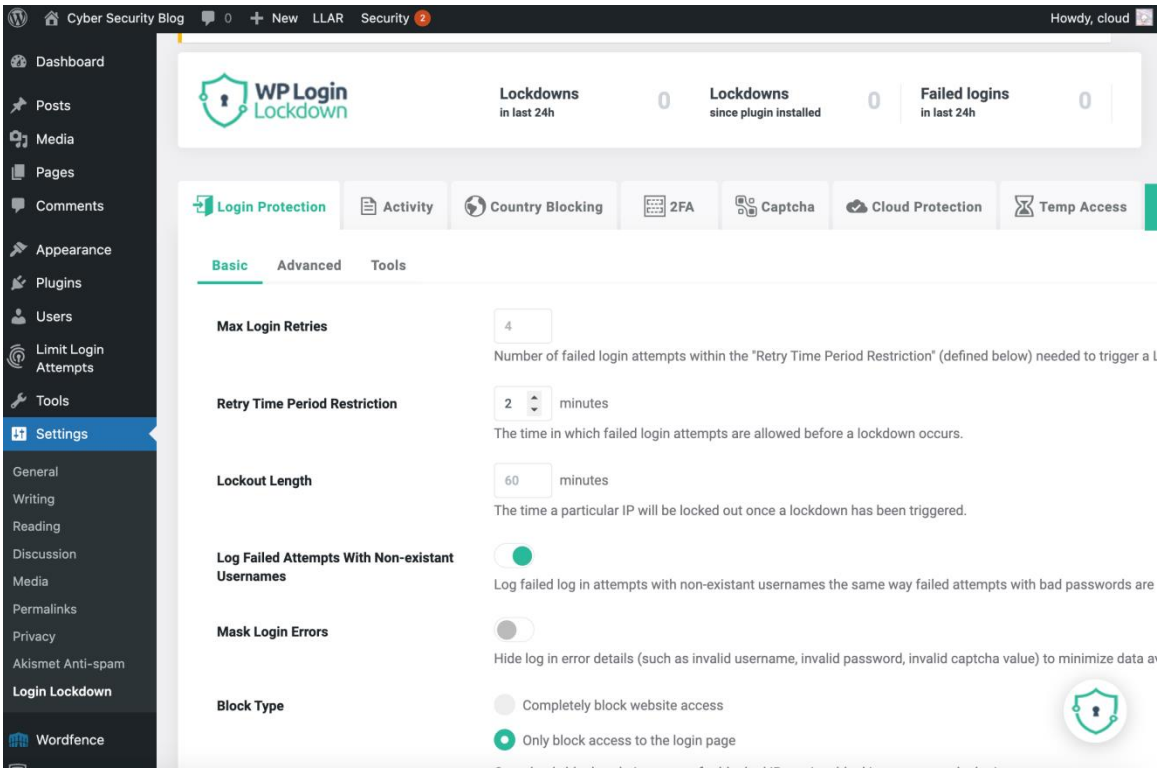
Enabling two factor authentication



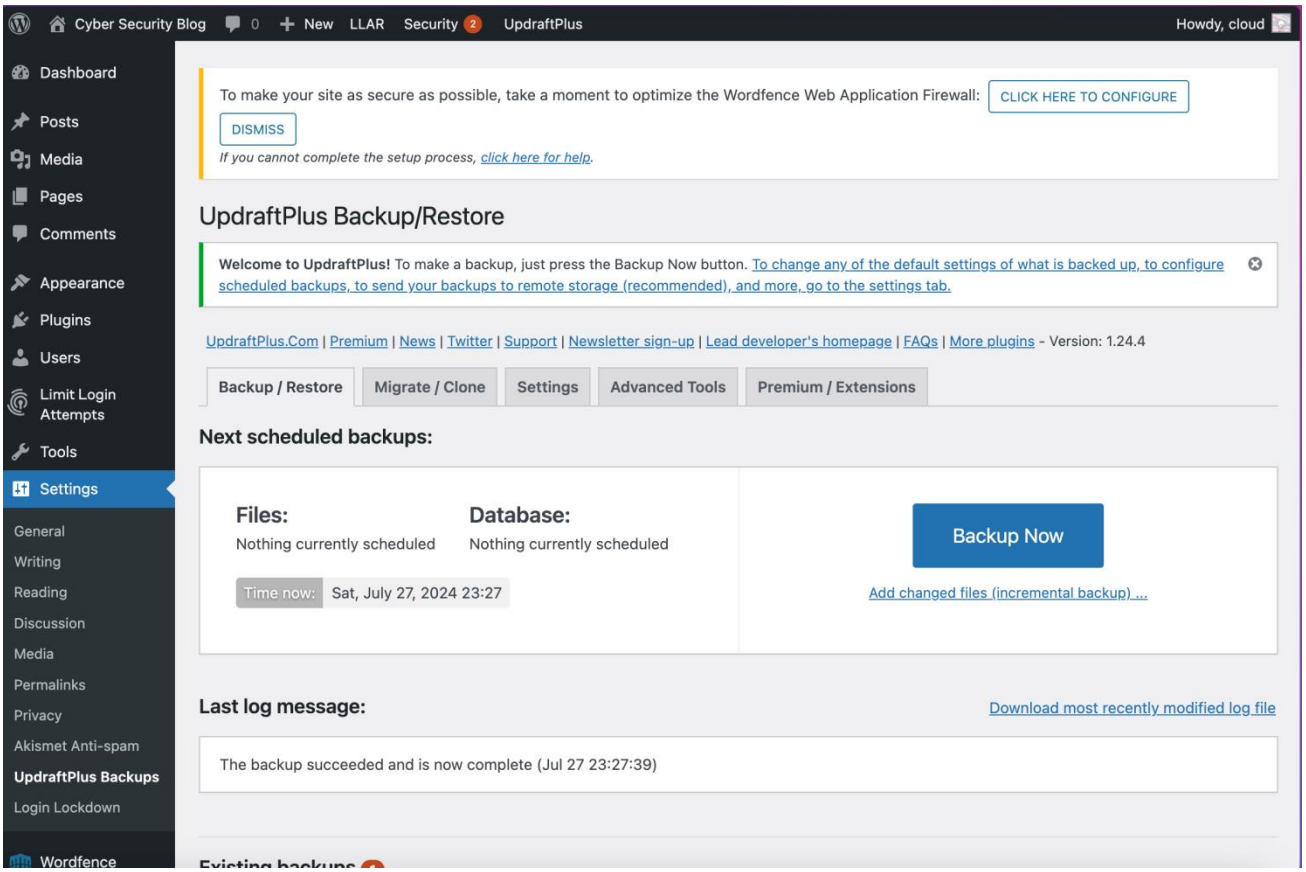
The user groups



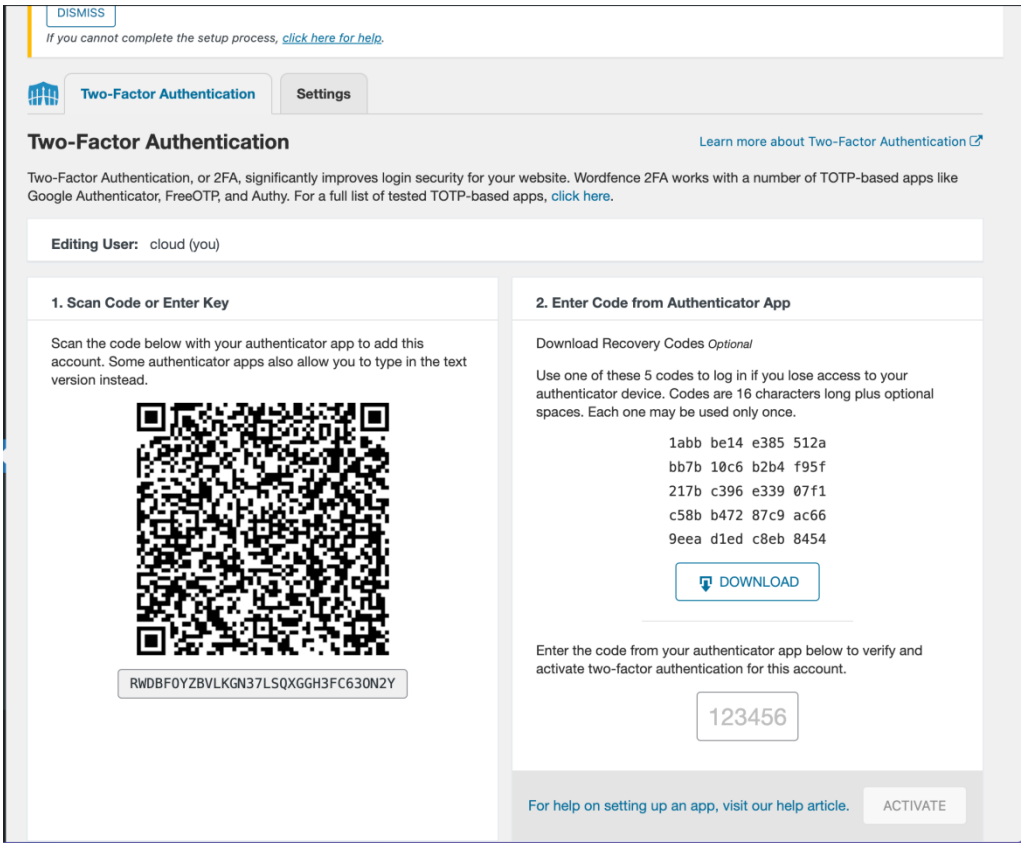
Login Lockdown



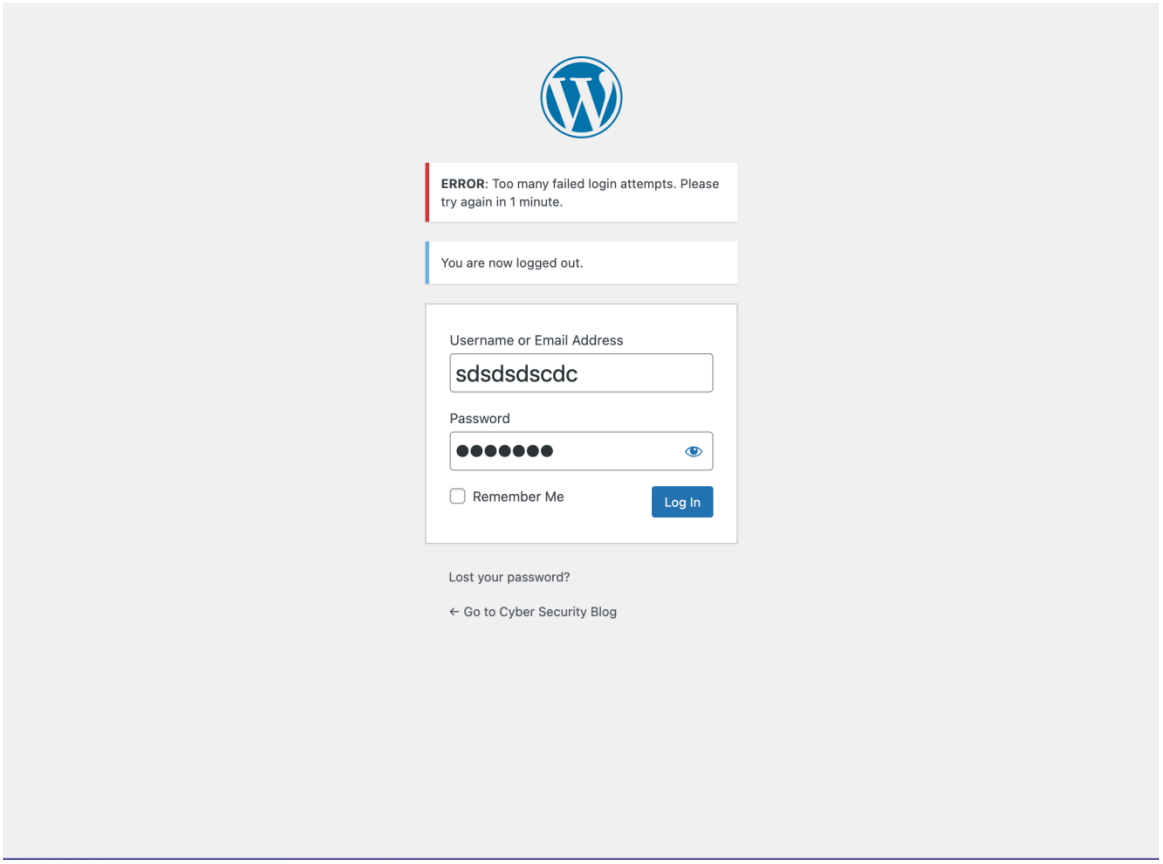
Updraft Plus Backup



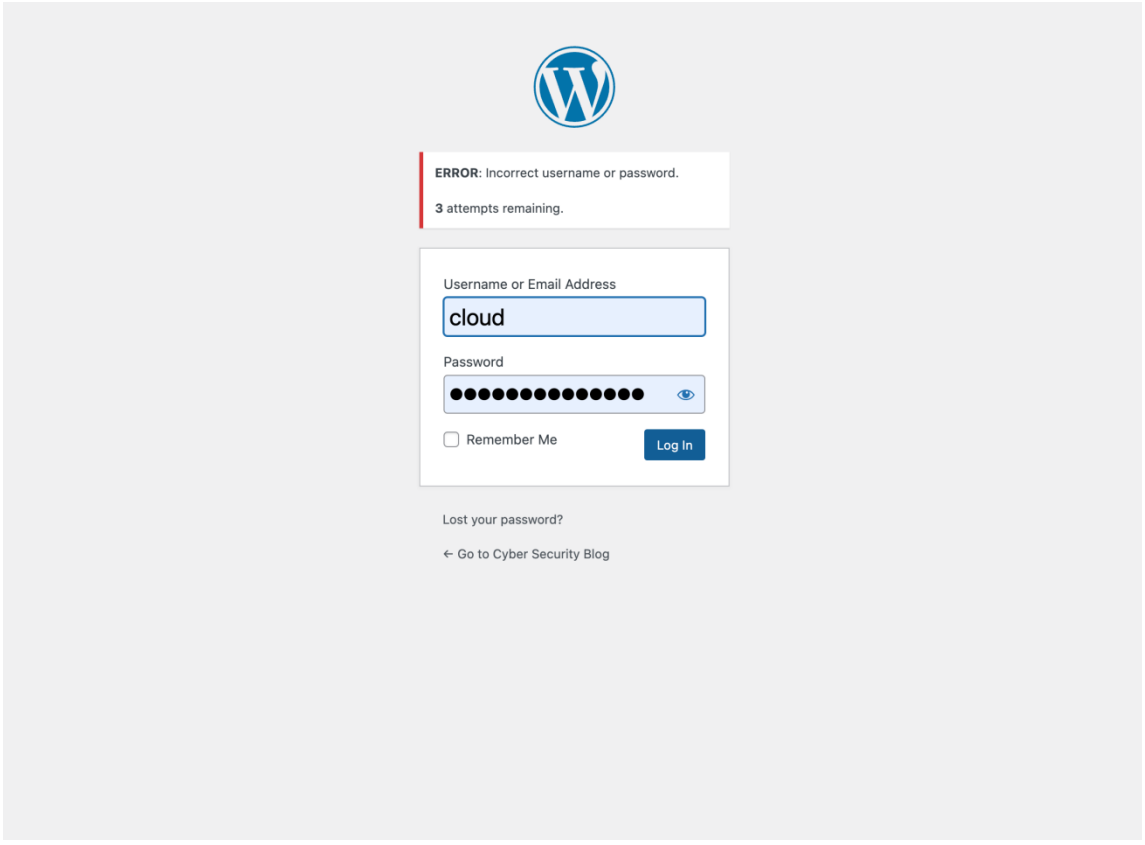
Enabling Two factor authentication for admin



Limiting Login Attempts



Limited Attempts



List of plugins