TLS For WordPress Site (Sandbox) Findings:

Background:

The wordpress sites's database, RDS MySQL, is not TLS enabled and is reported that the traffic is not encrypted. When a WordPress site's connection to its RDS MySQL database is not TLS enabled, all data exchanged between them travels across the network in plaintext. This means the information is completely unencrypted and readable to anyone who can intercept the traffic. Without TLS, the data packets containing sensitive information are exposed. An attacker positioned on the network between the WordPress server (EC2 instance) and database (RDS) can use simple tools to "sniff" or capture this traffic. For this reason, in this doc we are sandboxing for making the already running system into TLS compliant.

But first it is highly recommended to go through this set of documents for a complete clarification of the setup.

- DB Parameter Group changes
- Encrypting all connections with TLS for MySQL

Overview of steps:

1. Installed wordpress
2. Connection with RDS from wp-config.php
   This doc covers process from here:
3. Optional- TLS check using a custom plugin (The Plugin code is in the doc)
4. Creating custom RDS parameter group and associating it into the instance
5. Configuring the wp-config file for accepting TLS traffic from RDS
6. Reviewing configurations and restarting the RDS database
7. Consecutive option- TLS check using the custom plugin.

Detailed Steps:

Note: The following steps outline the requirements that are needed after the wordpress application is already active and running well. The purpose of this document is to only enable TLS into the already running system.

1. Creating The SSL status plugin (optional):

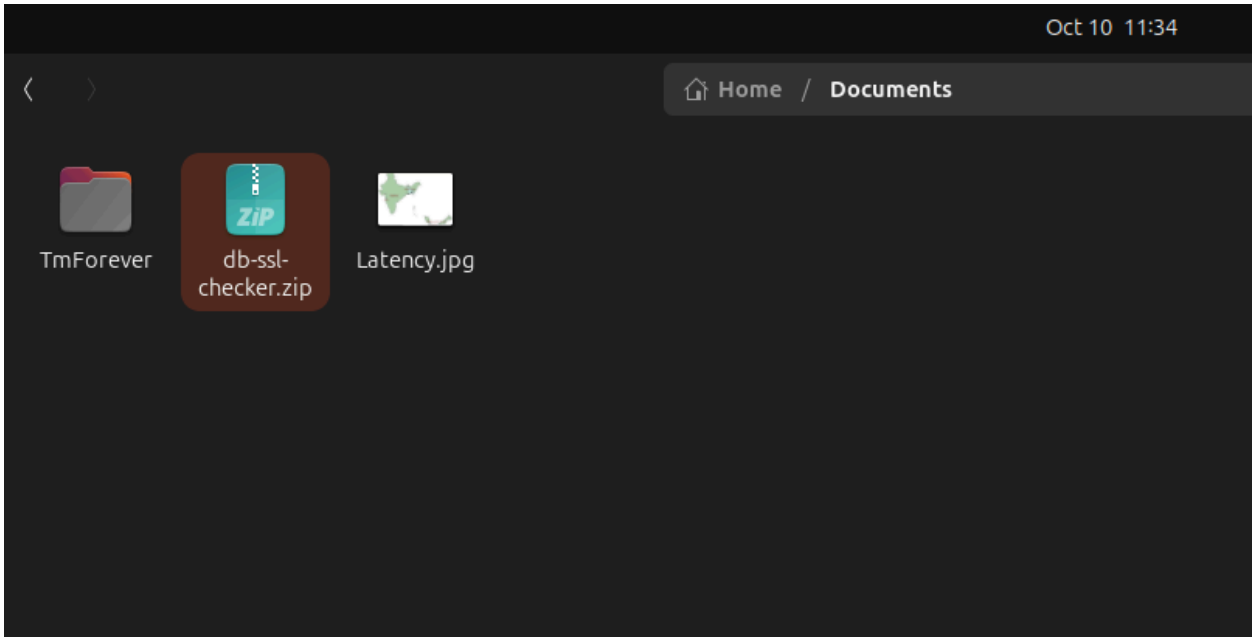
This is **completely optional**. You may proceed from the 2nd step.
For ease, I have created a wordpress plugin that can check if the wordpress application on the frontend is using TLS or not.

```
<?php
/**
 * Plugin Name: Database SSL Status Checker
 * Description: Adds an admin menu page to show the current SSL status of the WordPress
database connection.
 * Version: 1.0
 * Author: Manoj Gautam
 */

// It is recommended  allow direct access to this file.
if (!defined('ABSPATH')) {
    exit;
}

// 1. Add a new page to the admin menu.
function dbs_ssl_add_admin_menu() {
    add_menu_page(
        'Database SSL Status',      // Page Title
        'DB SSL Status',            // Menu Title
        'manage_options',           // Capability required
        'database-ssl-status',      // Menu Slug (URL)
        'dbs_ssl_render_status_page', // Function to render the page content
        'dashicons-lock',           // Icon
        6                           // Position
    );
}
add_action('admin_menu', 'dbs_ssl_add_admin_menu');

// 2. Render the content for the admin page.
function dbs_ssl_render_status_page() {
    global $wpdb; // Access the WordPress database object.

    // Query the database for the current connection's SSL cipher.
    $ssl_cipher_status = $wpdb->get_row("SHOW STATUS LIKE 'Ssl_cipher'", ARRAY_A);
    $ssl_cipher_value = isset($ssl_cipher_status['Value']) ? $ssl_cipher_status['Value'] : '';
```

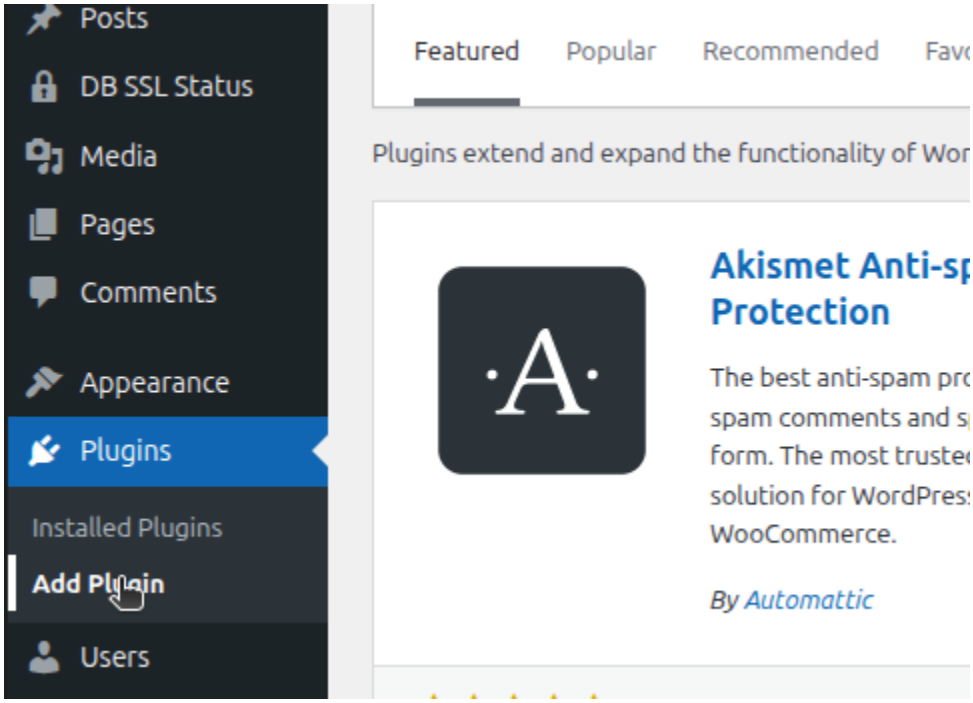
```
    ?>
    <div class="wrap">
        <h1>Database Connection SSL Status</h1>
        <hr>
        <?php
        // Check if the Ssl_cipher value is not empty.
        if (!empty($ssl_cipher_value)) {
            // SUCCESS: Connection is encrypted.
            ?>
            <div style="border-left: 4px solid #4CAF50; padding: 10px 20px; background-color:
#f1f8e9;">
                <h2> Connection is Secure (Using SSL/TLS)</h2>
                <p>WordPress is communicating with your database over an encrypted
connection.</p>
                <p><strong>SSL Cipher in use:</strong> <?php echo esc_html($ssl_cipher_value);
?></p>
            </div>
            <?php
        } else {
            // FAILURE: Connection is not encrypted.
            ?>
            <div style="border-left: 4px solid #F44336; padding: 10px 20px; background-color:
#ffebee;">
                <h2> Connection is NOT Secure</h2>
                <p>WordPress is communicating with your database over an unencrypted,
plain-text connection.</p>
            </div>
            <?php
        }
        ?>
    </div>
    <?php
}
```

For installing this, import this php script into a notepad in your local computer. Save this file as a php file. Then compress this and make it a zip folder.

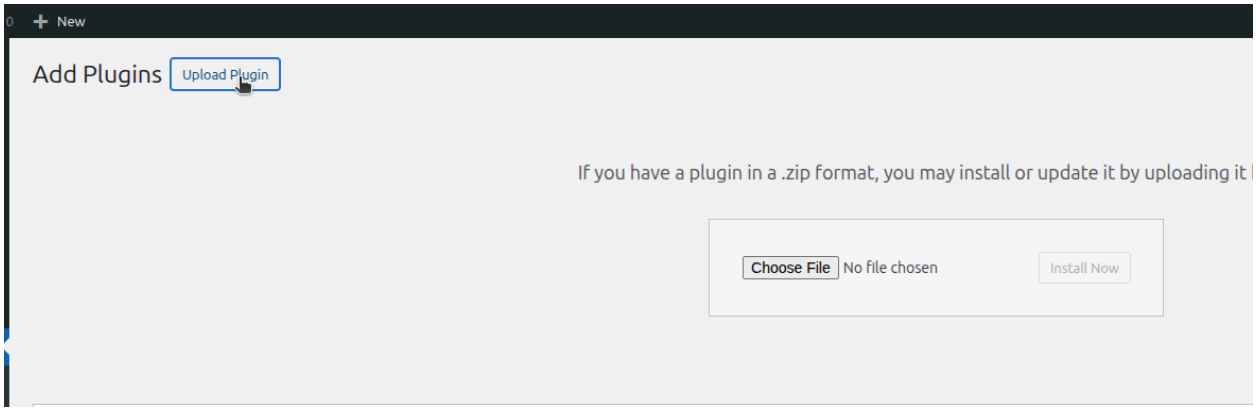
.zip folder



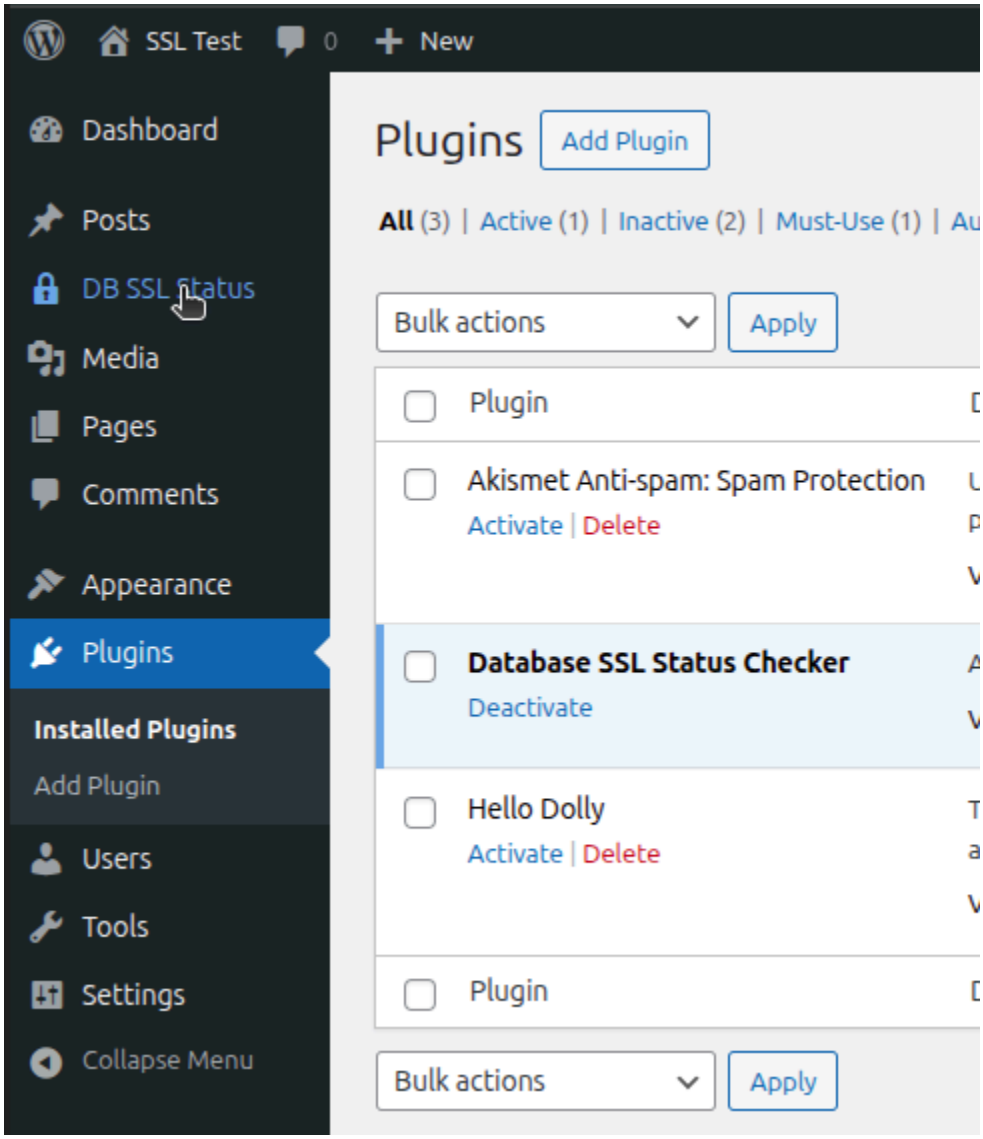
Head over to the WP admin dashboard and hit add plugin



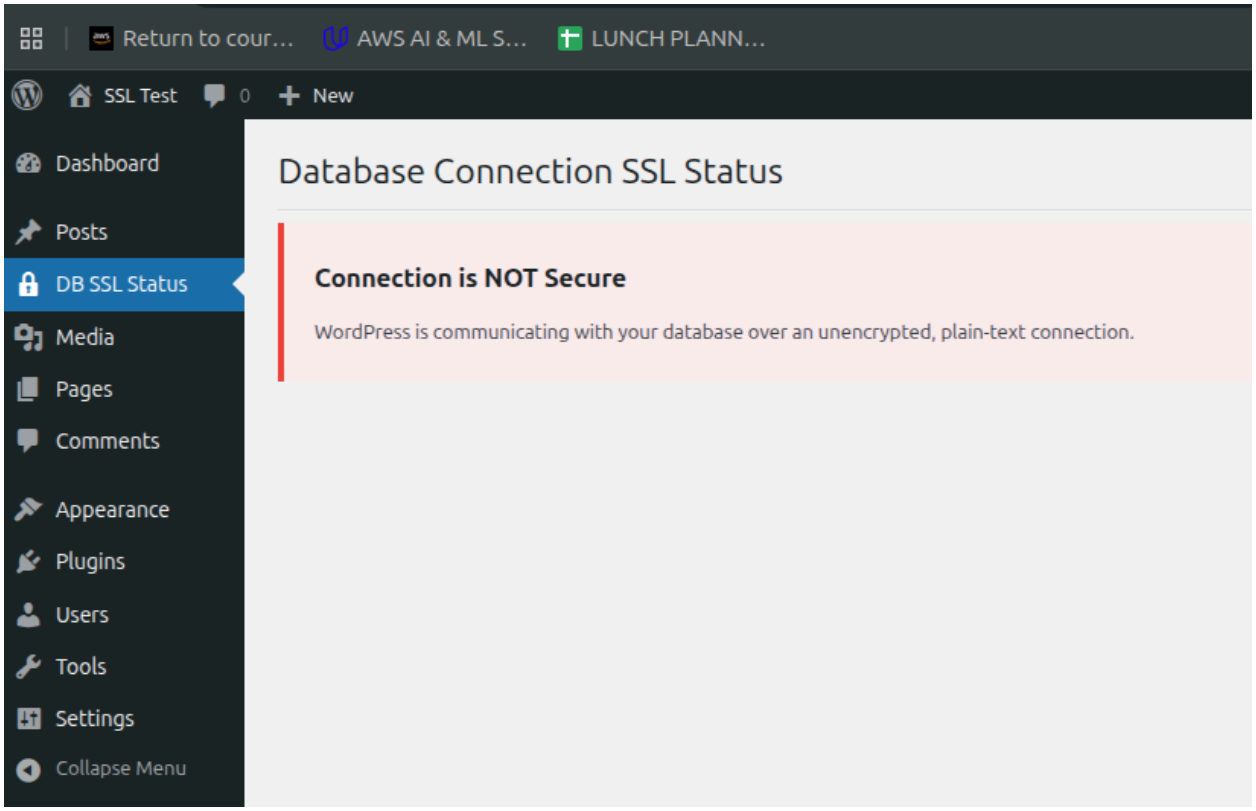
Then on the top of the page, click on upload plugin and upload the .zip folder.



Activate the plugin and on the left menu you will see a new menu named DB SSL Status that pops up.



If the system is not configured with TLS, it will show like this:



The plugin is reporting perfectly fine in my use case. Apologies if the plugin is not working correctly, the plugin has not been extensively tested and was made just for this specific purpose and lies only to the scope of this document.

In any case, we can continue with our setup from step 2.

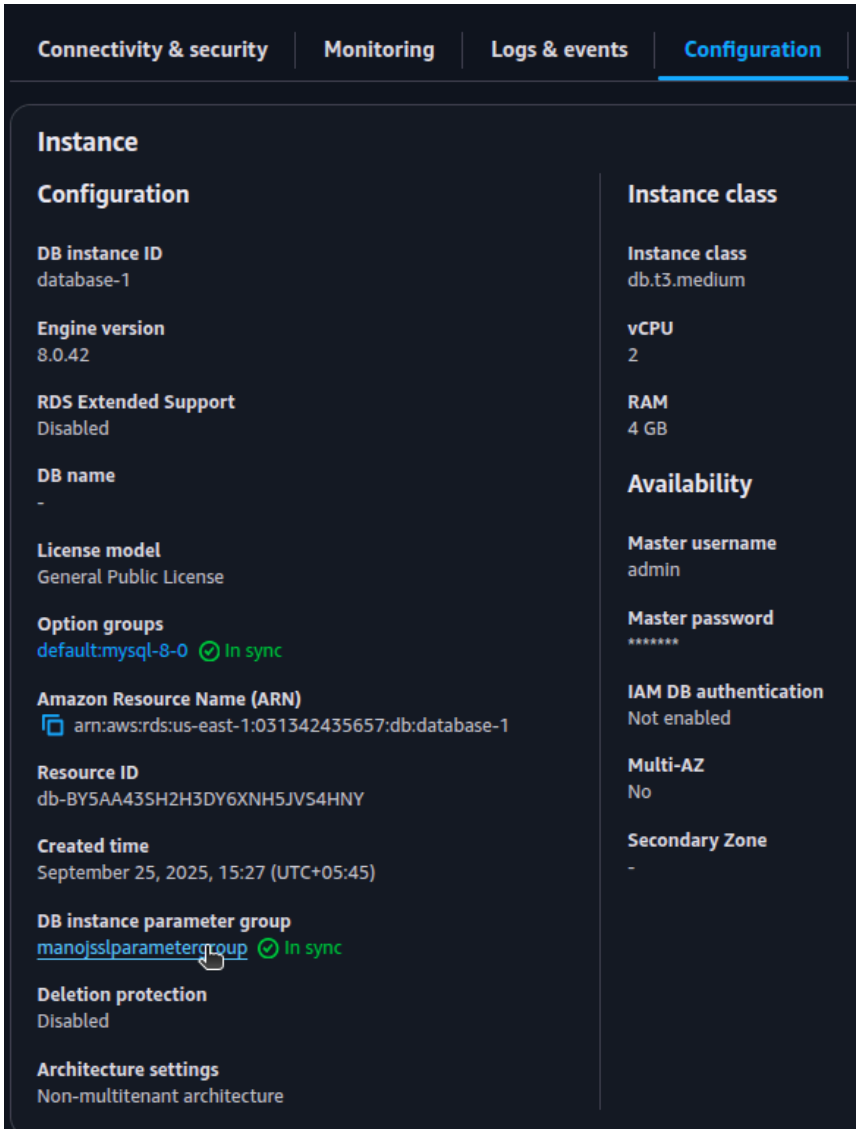
2. Configuring RDS for TLS:

I referred to [this document](#) (mentioned in the first section of this document) for the process of enabling TLS in the RDS MySQL database.

To summarize,

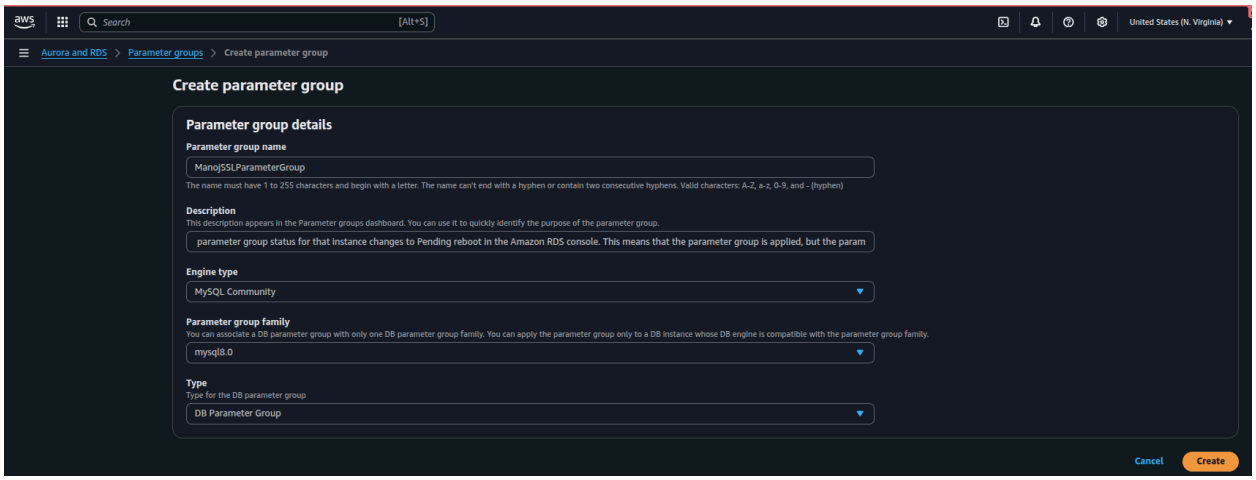
- The *require_secure_transport* parameter group of the database is to set true.
- Then the client application is to be modified to use the global AWS certificate for decrypting the encrypted traffic.

But, our database is using default parameter group as seen in the screenshot below:

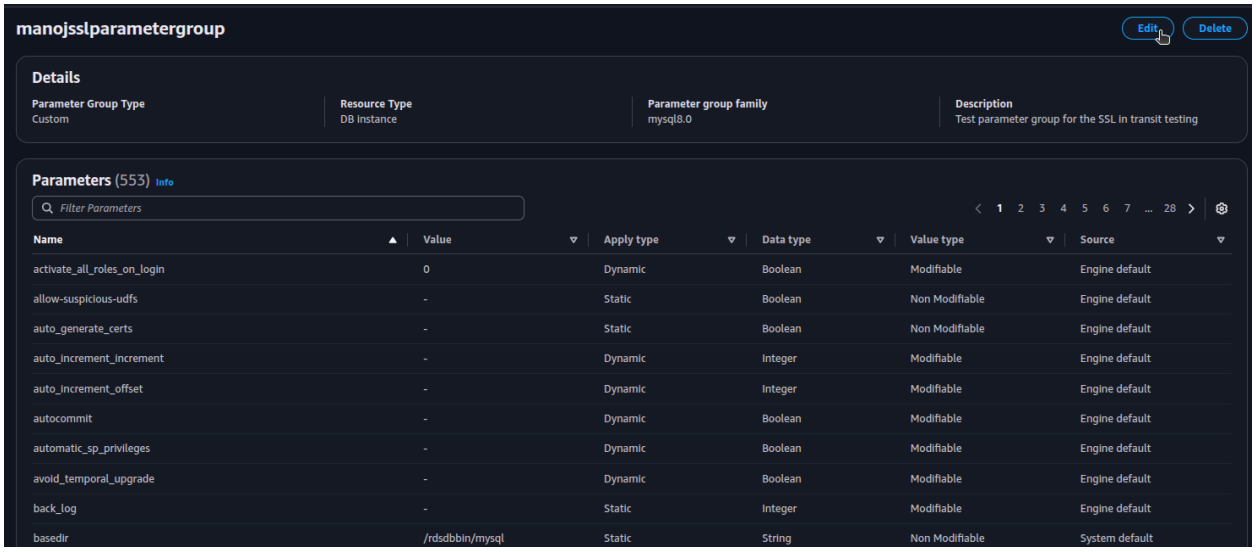


But, the default parameter groups cannot be changed. We have to create a new custom parameter group and then associate it to our DB instance. For applying the parameter group change, we then have to restart the DB instance (This is the downtime that we will face in the scenario)

Creating a custom parameter group. Hit on create with these configurations for MySQL default parameters.



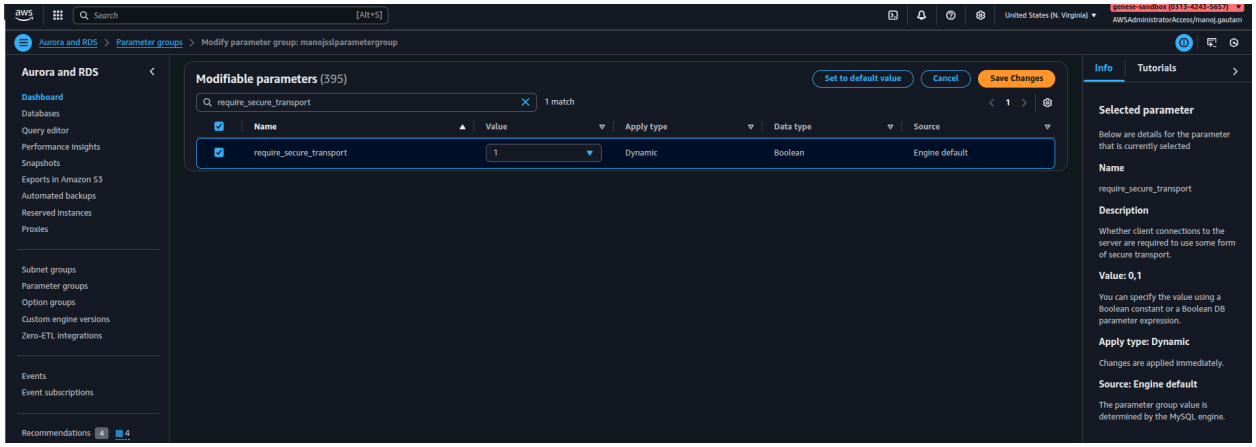
Now we only need to change the require_secure_transport parameter. Hence modify the just created parameter group.



Name	Value	Apply type	Data type	Value type	Source
activate_all_roles_on_login	0	Dynamic	Boolean	Modifiable	Engine default
allow-suspicious-udfs	-	Static	Boolean	Non Modifiable	Engine default
auto_generate_certs	-	Static	Boolean	Non Modifiable	Engine default
auto_increment_increment	-	Dynamic	Integer	Modifiable	Engine default
auto_increment_offset	-	Dynamic	Integer	Modifiable	Engine default
autocommit	-	Dynamic	Boolean	Modifiable	Engine default
automatic_sp_privileges	-	Dynamic	Boolean	Modifiable	Engine default
avoid_temporal_upgrade	-	Dynamic	Boolean	Modifiable	Engine default
back_log	-	Static	Integer	Modifiable	Engine default
basedir	/rdsdbbin/mysql	Static	String	Non Modifiable	System default

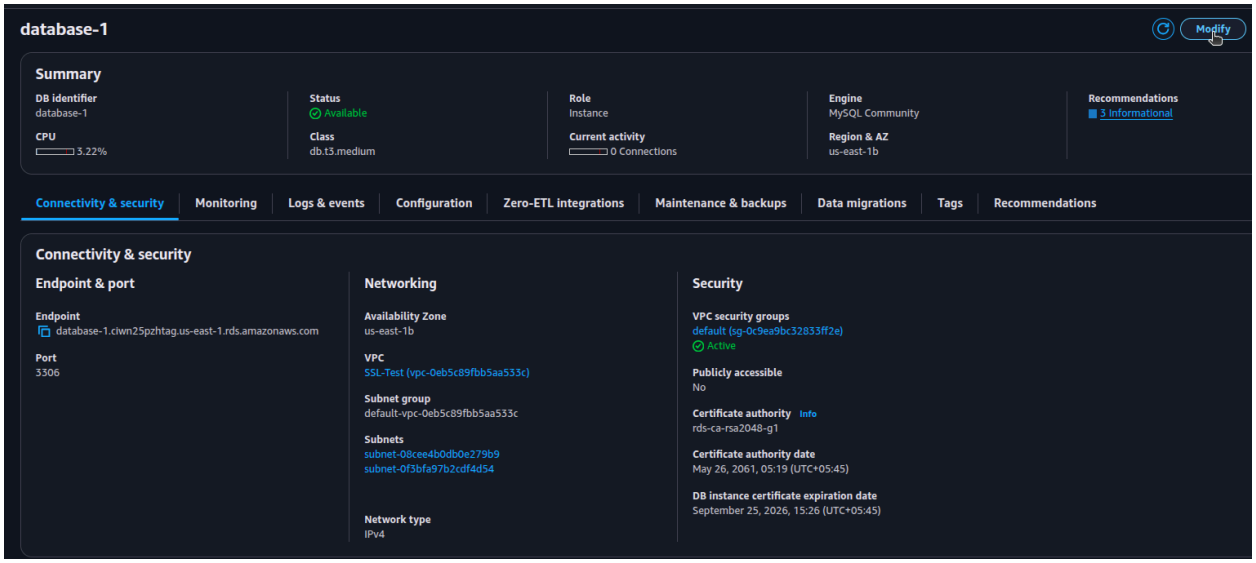
Search for the parameter require_secure_transport and change from 0 to 1.

Now the require_secure transport is checked, value was changed from 0 to 1 then press save changes.

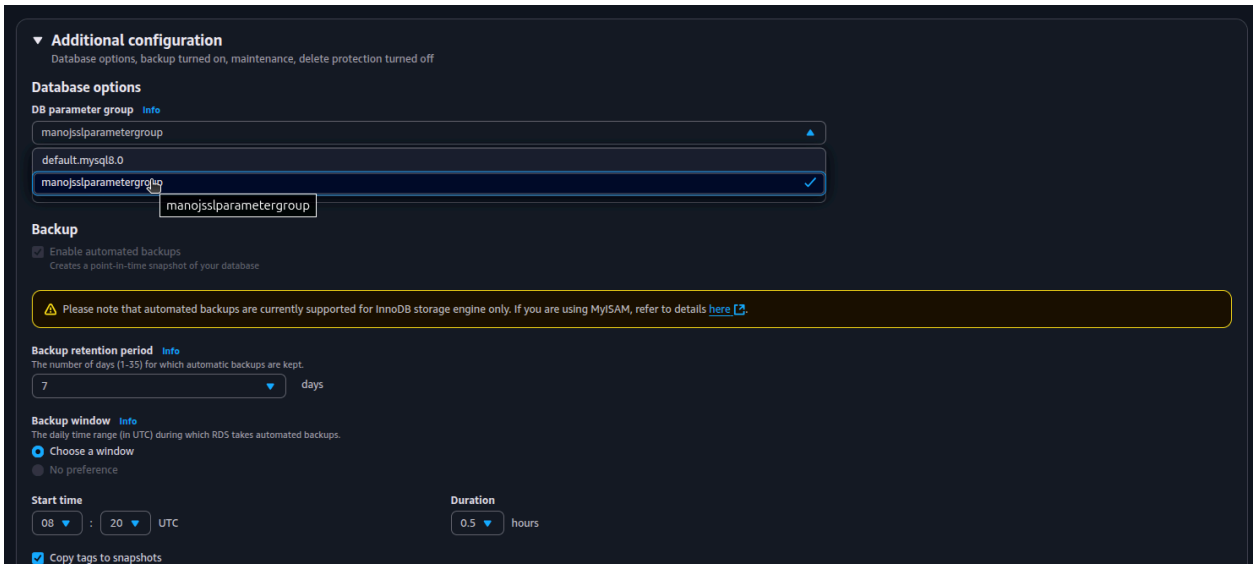


Now for the final change, modify the database instance to use this custom parameter group.

Head over to modify



Scroll down to additional configuration to select the custom parameter group that we have just created.



Save this modification.

Note that the parameter group change is only applied after a database instance reboot. So this change is not applied until a reboot.

3. Application level changes:

Till the previous step, forcing TLS is similar for all scenarios. Now we need some specific application level changes for a two way encryption and decryption.

For using the official AWS specific certs, we need to first download to the instance that is running the wordpress application. We'll then make modifications in the wp-config file then make a plugin to point to the location of downloaded CA certs. *Making a plugin just to point to the certification location seems outrageously complex, but was found to be the standard way in the wordpress environment.*

Download and move global certs:

```
ubuntu@ip-10-0-0-9:/tmp$ wget https://truststore.pki.us-east-1.amazonaws.com/global/global-bundle.pem
--2025-10-09 11:30:45--  https://truststore.pki.us-east-1.amazonaws.com/global/global-bundle.pem
Resolving truststore.pki.us-east-1.amazonaws.com (truststore.pki.us-east-1.amazonaws.com)... failed: Name or service not known.
wget: unable to resolve host address 'truststore.pki.us-east-1.amazonaws.com'
ubuntu@ip-10-0-0-9:/tmp$ wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
--2025-10-09 11:33:09--  https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
Resolving truststore.pki.rds.amazonaws.com (truststore.pki.rds.amazonaws.com)... 18.165.98.93, 18.165.98.60, 18.165.98.84, ...
Connecting to truststore.pki.rds.amazonaws.com (truststore.pki.rds.amazonaws.com)|18.165.98.93|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 165408 (162K) [binary/octet-stream]
Saving to: 'global-bundle.pem'

global-bundle.pem                   100%[===================================================================================================>] 161.53K  --.-KB/s    in 0.002s

2025-10-09 11:33:09 (99.8 MB/s) - 'global-bundle.pem' saved [165408/165408]

ubuntu@ip-10-0-0-9:/tmp$ sudo mv global-bundle.pem /etc/ssl/certs/aws-rds-global-bundle.pem
ubuntu@ip-10-0-0-9:/tmp$
```

Here's the script:

```
wget https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem
sudo mv global-bundle.pem /etc/ssl/certs/aws-rds-global-bundle.pem
```

Now we make changes to the **wp-config file**:

Lets head over to the config file:

```
sudo nano /var/www/html/mywebsite/wp-config.php
```

In the wp config file we add just a single line before the `/* That's all, stop editing!`

```
define('MYSQL_CLIENT_FLAGS', MYSQLI_CLIENT_SSL);
```

```
 *
 * Change this to true to enable the display of notices during development.
 * It is strongly recommended that plugin and theme developers use WP_DEBUG
 * in their development environments.
 *
 * For information on other constants that can be used for debugging,
 * visit the documentation.
 *
 * @link https://developer.wordpress.org/advanced-administration/debug/debug-wordpress/
 */
define( 'WP_DEBUG', false );

/* Add any custom values between this line and the "stop editing" line. */

define('MYSQL_CLIENT_FLAGS', MYSQLI_CLIENT_SSL);

/* That's all, stop editing! Happy publishing. */

/** Absolute path to the WordPress directory. */
if ( ! defined( 'ABSPATH' ) ) {
        define( 'ABSPATH', __DIR__ . '/' );
}
```

Now, we used the MYSQL_CLIENT_FLAGS, but to define path
we need to tell WordPress where to find the certificate file. The best way to do this is with a small, automatically-loaded plugin

```
ubuntu@ip-10-0-0-9:/tmp$
ubuntu@ip-10-0-0-9:/tmp$
ubuntu@ip-10-0-0-9:/tmp$ sudo mkdir -p /var/www/html/wp-content/mu-plugins
ubuntu@ip-10-0-0-9:/tmp$
ubuntu@ip-10-0-0-9:/tmp$
ubuntu@ip-10-0-0-9:/tmp$ sudo nano /var/www/html/wp-content/mu-plugins/db-ssl-verify.php
ubuntu@ip-10-0-0-9:/tmp$
```

And inside the php file we have:

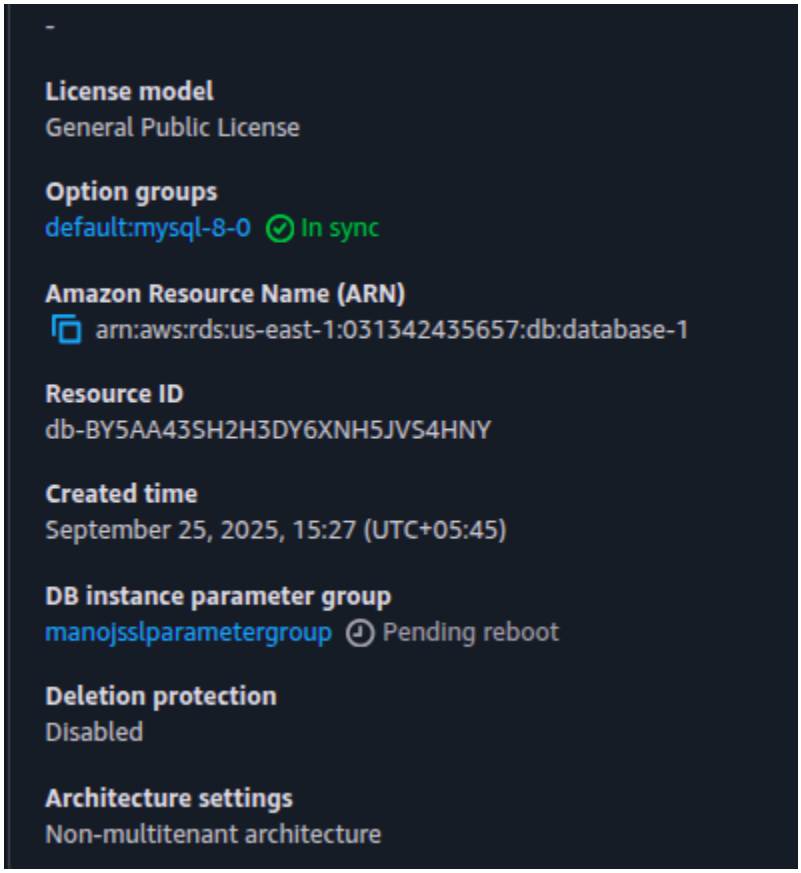
```
  GNU nano 7.2                                                    /var/www/html/wp-content/mu-plugins/db-
<?php
/**
 * Plugin Name: Force SSL Verification for DB Connection
 * Description: Forces WordPress to use a specific SSL CA certificate to verify the database server's identity
 */
add_action( 'pre_wpdb_init', function( $wpdb ) {
    if ( defined( 'MYSQL_CLIENT_FLAGS' ) && MYSQL_CLIENT_FLAGS === MYSQLI_CLIENT_SSL ) {
        $wpdb->dbh->ssl_set(
            null,
            null,
            '/etc/ssl/certs/aws-rds-global-bundle.pem',
            null,
            null
        );
    }
}, 1, 1 );
```


```
<?php
/**
 * Plugin Name: Force SSL Verification for DB Connection
 * Description: Forces WordPress to use a specific SSL CA certificate to verify the database server's identity.
 */
add_action( 'pre_wpdb_init', function( $wpdb ) {
    if ( defined( 'MYSQL_CLIENT_FLAGS' ) && MYSQL_CLIENT_FLAGS === MYSQLI_CLIENT_SSL ) {
        $wpdb->dbh->ssl_set(
            null,
            null,
            '/etc/ssl/certs/aws-rds-global-bundle.pem',
            null,
            null
        );
    }
}, 1, 1 );
```

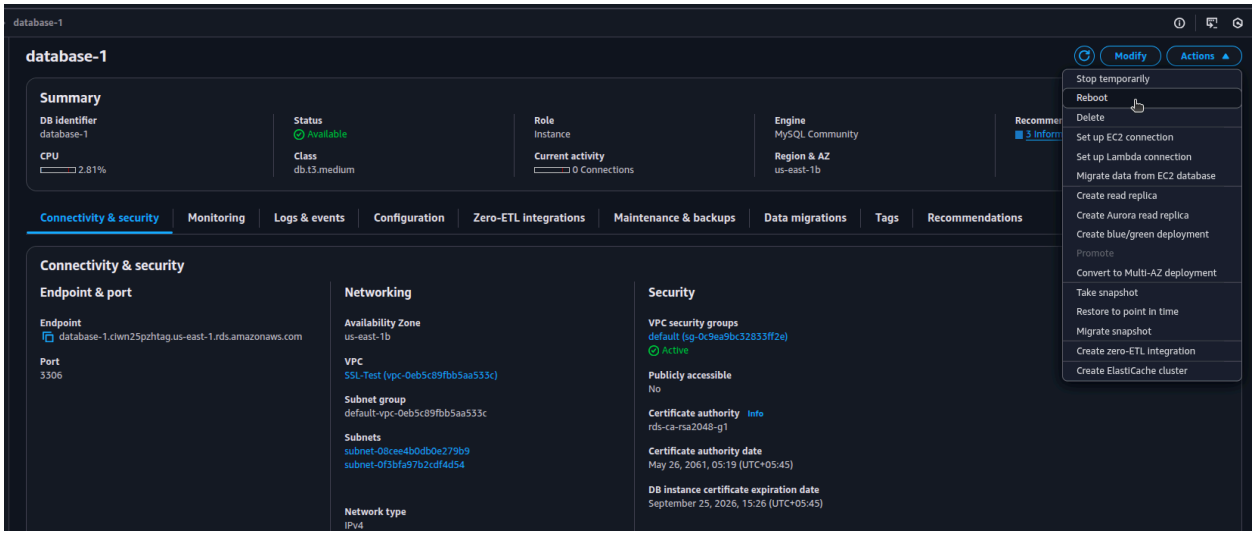

With these modifications, our application level changes are complete !

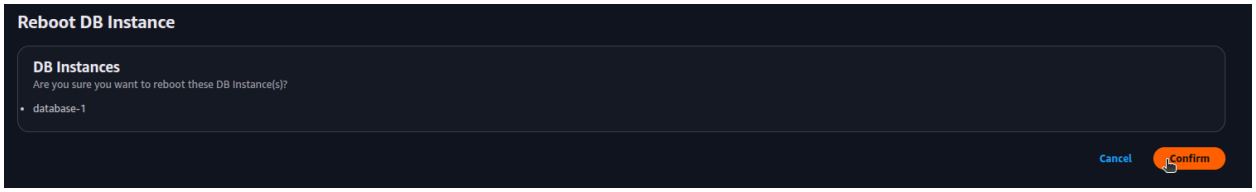
4. Restarting DB instance:

As we have mentioned before the parameter group changes are not applied until we do an instance reboot.



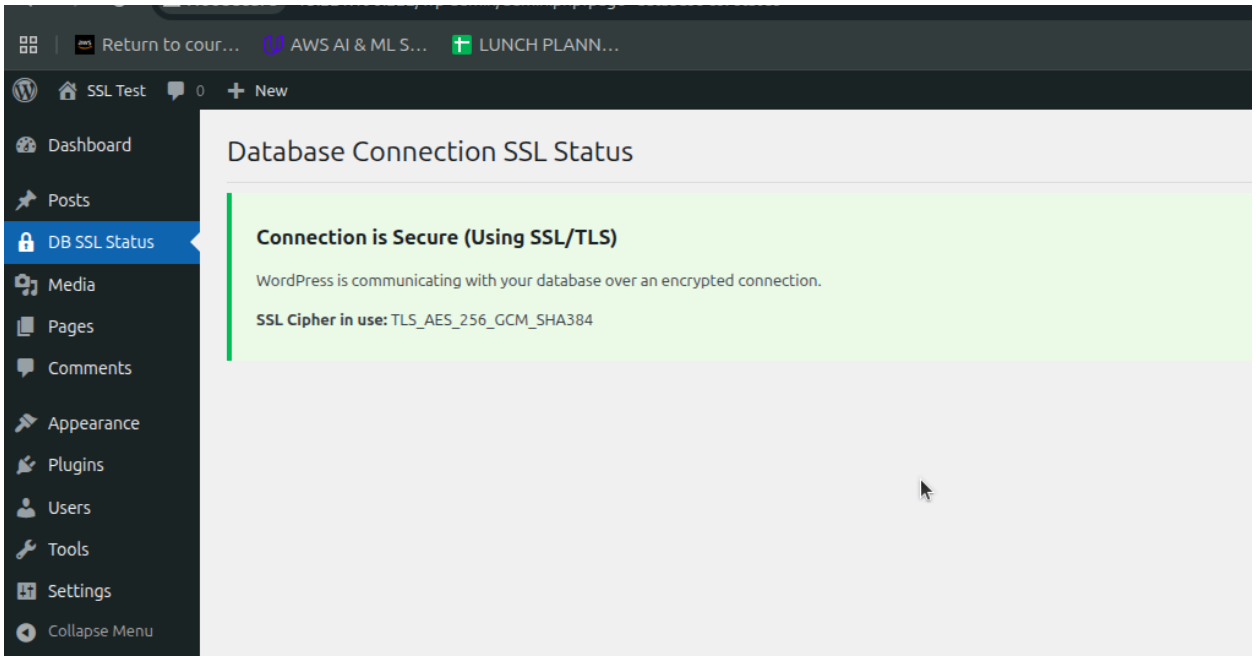
So, we head over to the AWS console and do a reboot.



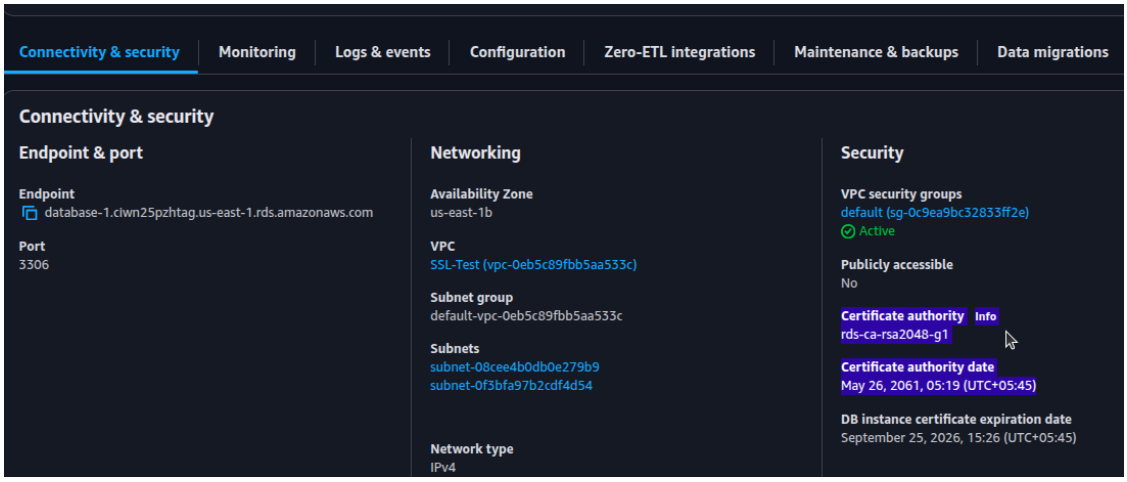


Note that the wordpress site will not work when DB is in reboot state (This is our only down time during the whole process)

When the DB shows available state, go to the WP admin page and click on the plugin that we have made:



Even if the plugin is not installed, we will see this on the console:



End of the document !