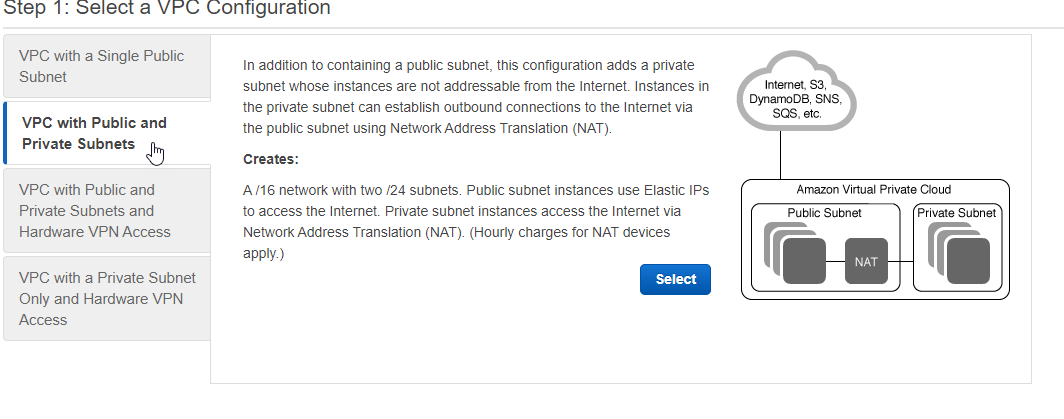
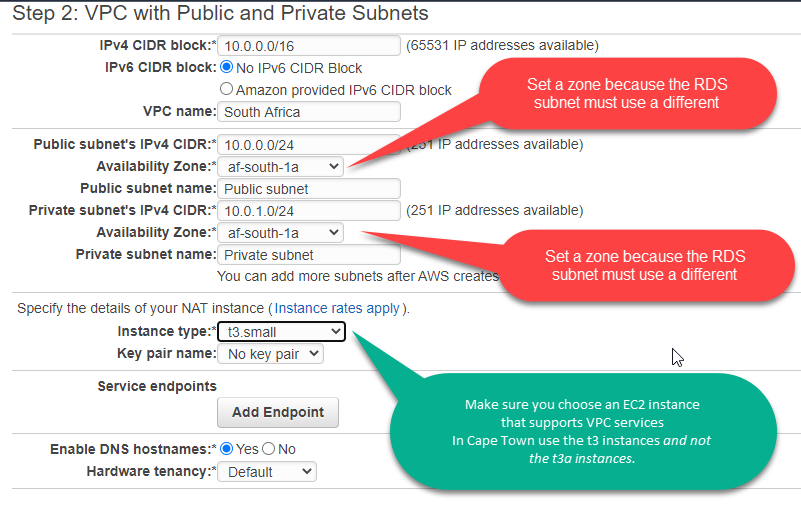
SETTING UP AN AWS VIRTUAL PRIVATE CLOUD (VPC)  
  
Choose **VPC Dashboard** and choose **Launch VPC Wizard**

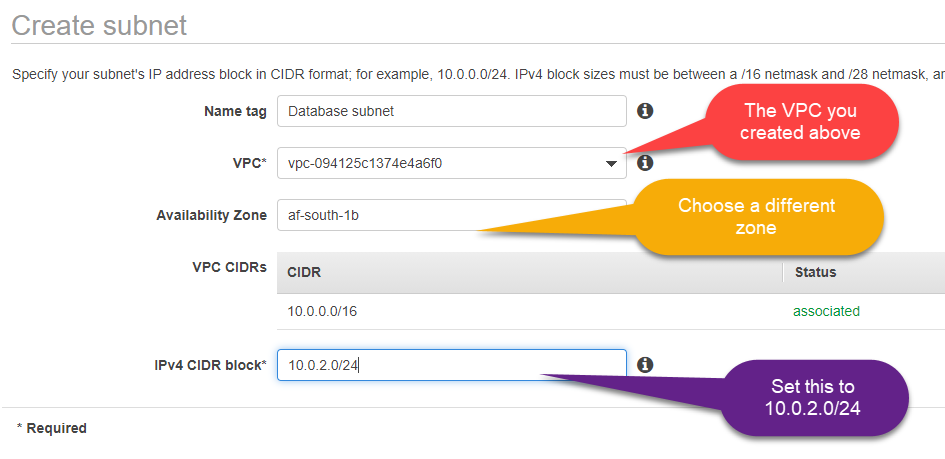




# CREATE A PRIVATE SUBNET FOR THE RDS

To add the second private subnet to your VPC, choose **VPC Dashboard**, choose **Subnets**, and then choose **Create subnet**.

Use 10.0.2.0/24 as the Ipv4 range



## **Create a VPC security group for a public web server**

Next you create a security group for public access. To connect to public instances in your VPC, you add inbound rules to your VPC security group that allow traffic to connect from the internet.

Choose **VPC Dashboard**, choose **Security Groups**, and then choose **Create security group**

# 

## **Create a VPC security group for a private DB instance**

To keep your DB instance private, create a second security group for private access. To connect to private instances in your VPC, you add inbound rules to your VPC security group that allow traffic from your web server only.

# 

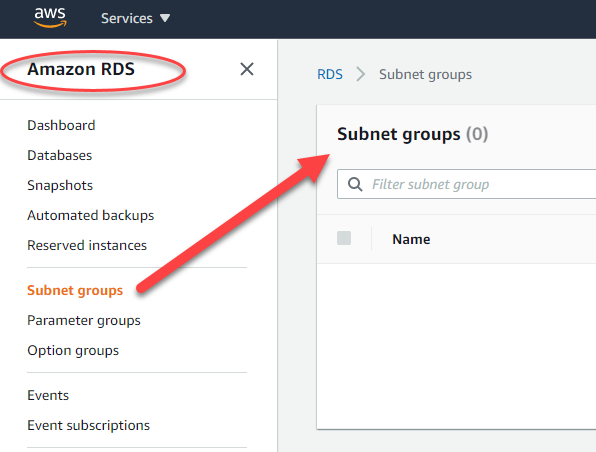
## **Create a DB subnet group**

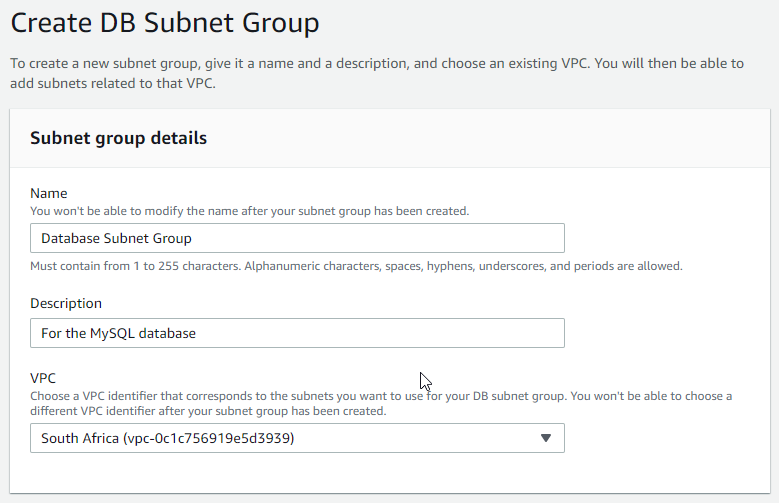
A DB subnet group is a collection of subnets that you create in a VPC and that you then designate for your DB instances. A DB subnet group allows you to specify a particular VPC when creating DB instances.

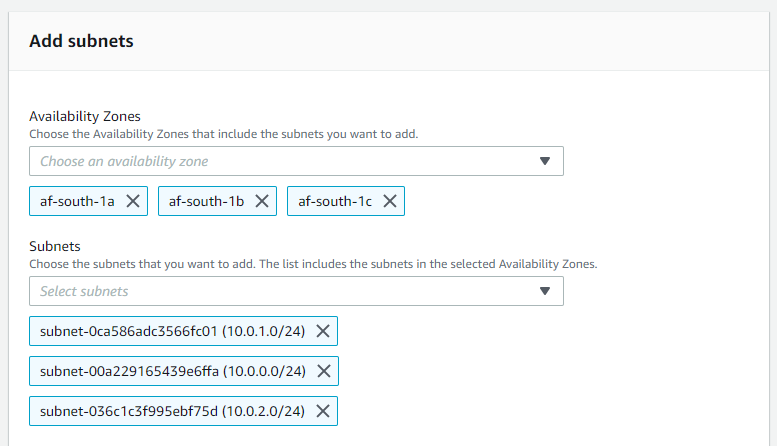
**To create a DB subnet group**

Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.

Make sure you connect to the Amazon RDS console, not to the Amazon VPC console.



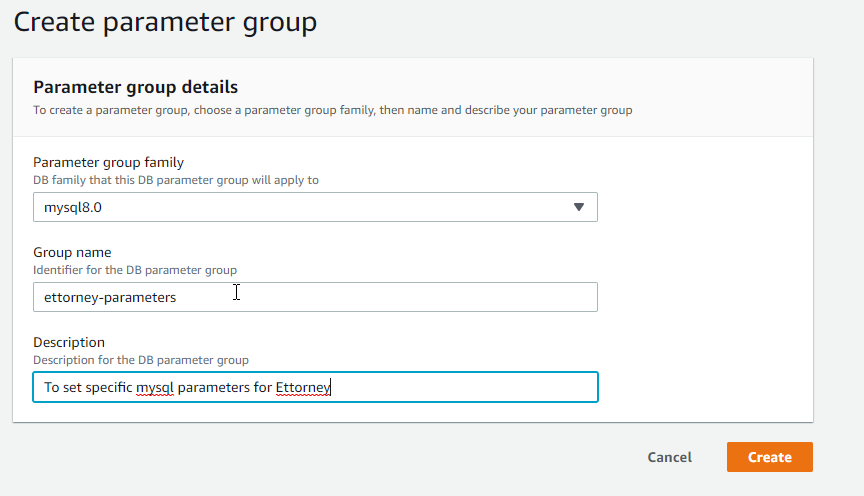


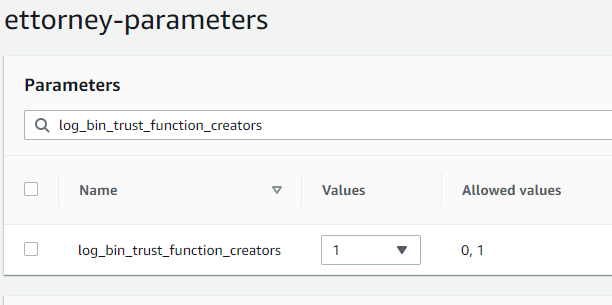


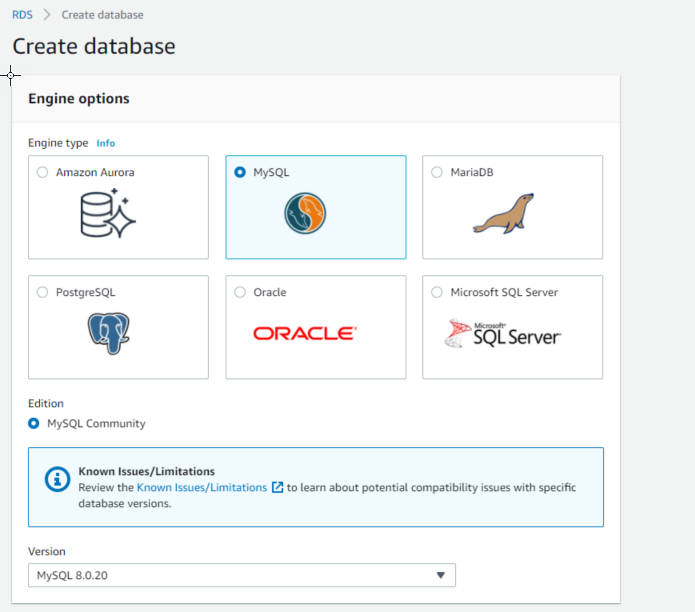
# CREATE THE RDS

**Step 1**

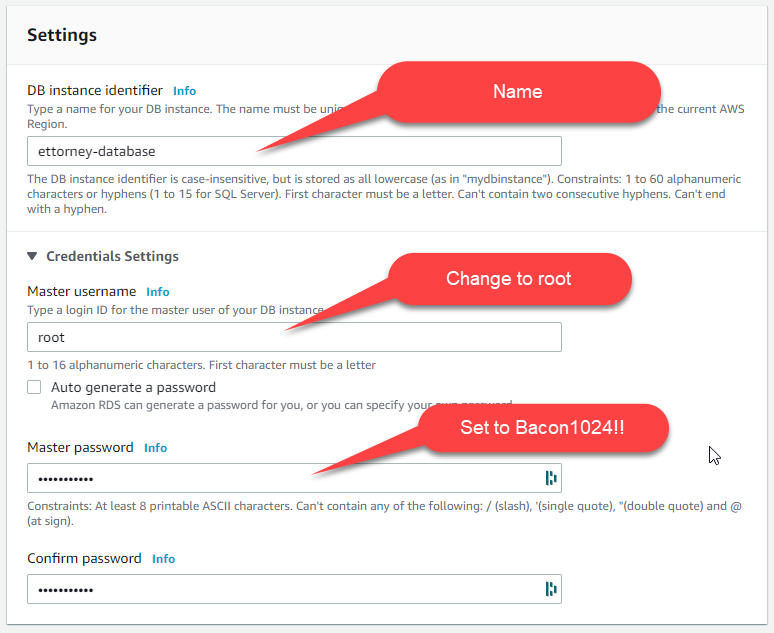
Create a parameter Group (so we can change some parameters)



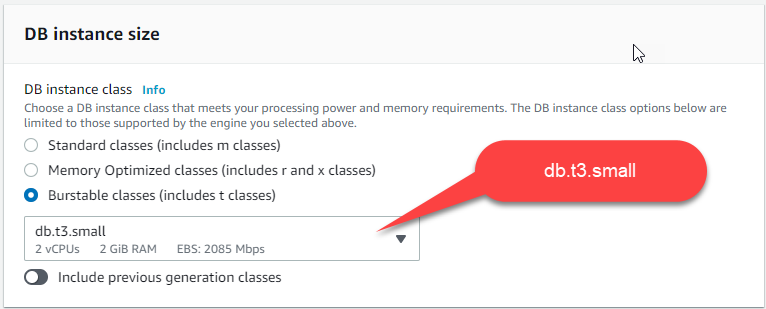


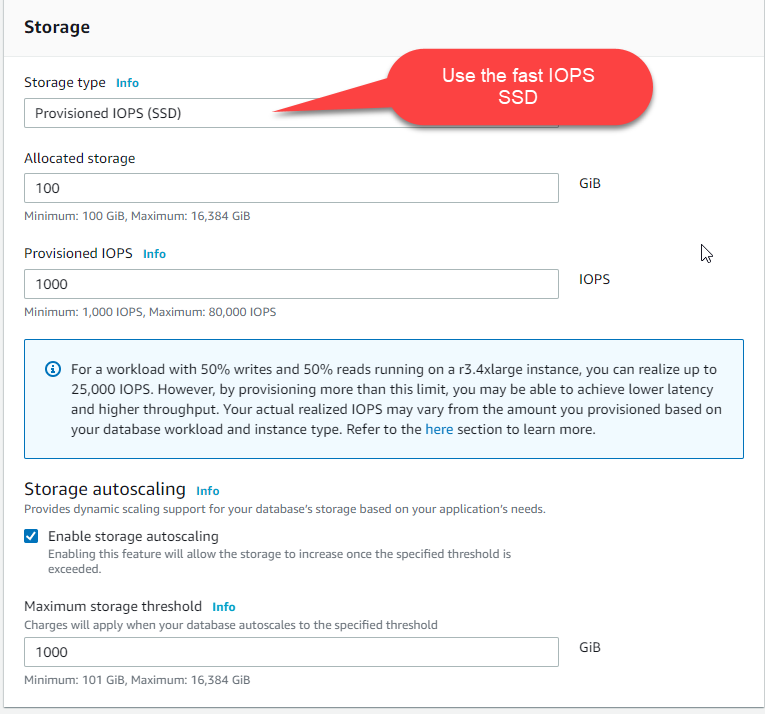


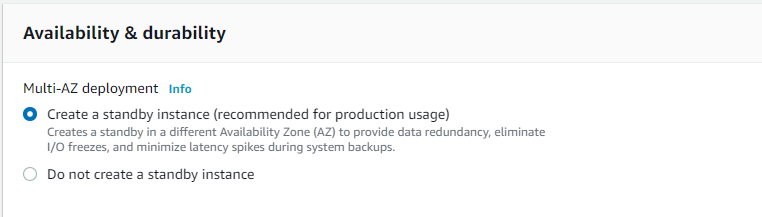
**Step 1**

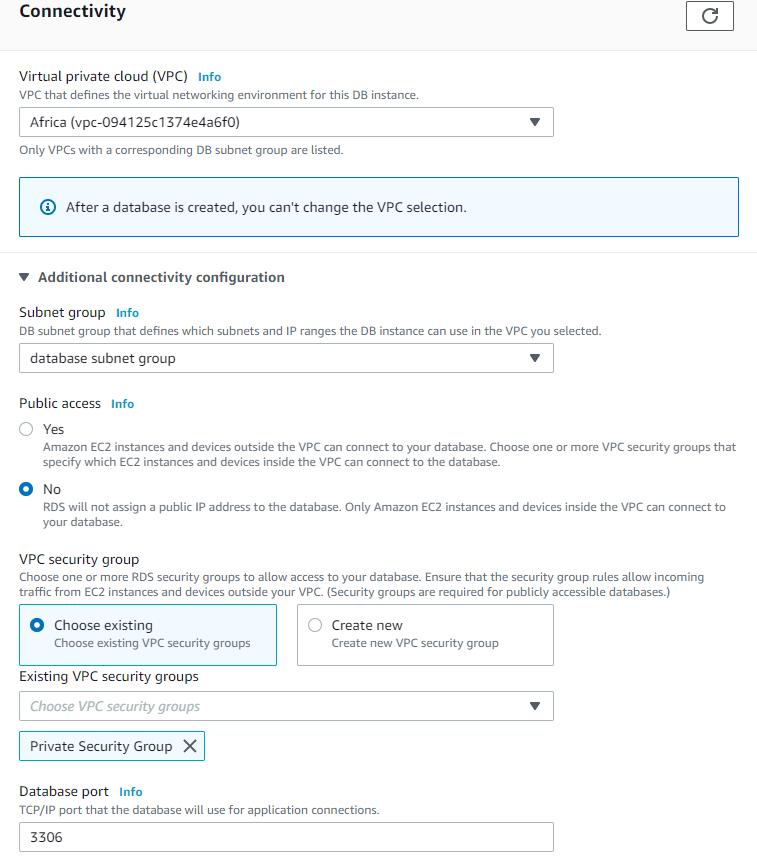


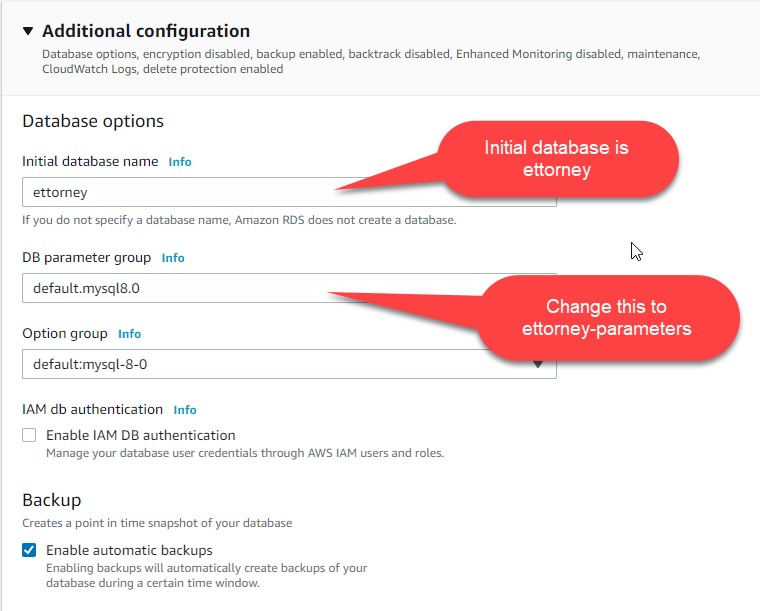
Instance Size  
  
Start on a T3 database instance which is an affordable way to kick off a project and build in some buffering for spikes in usage.

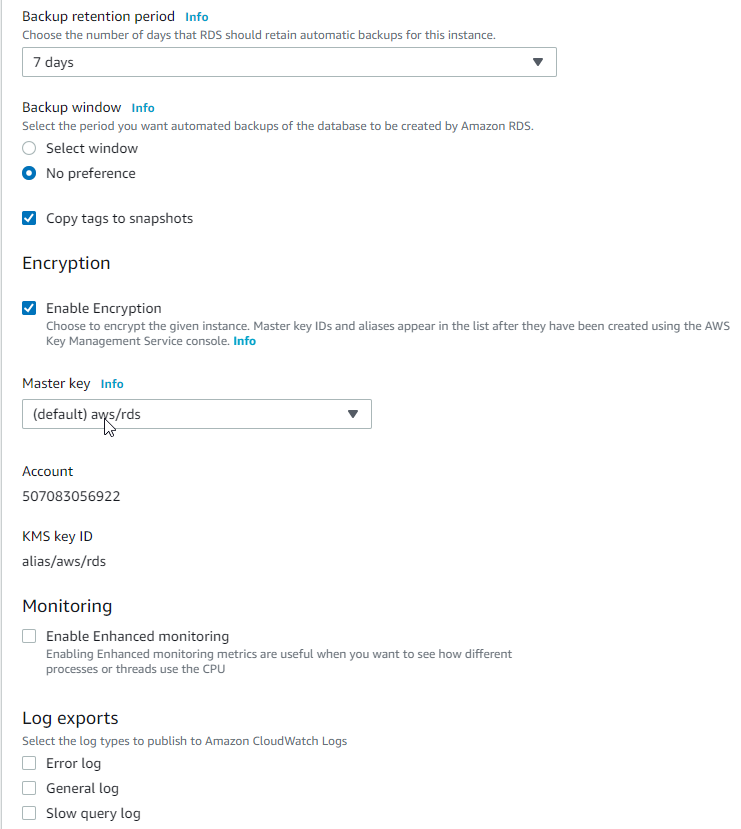
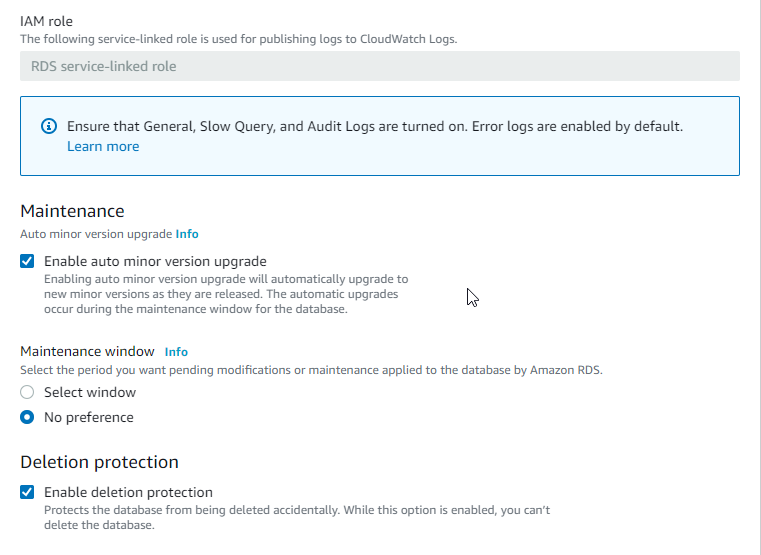










**Set parameter “log\_bin\_trust\_function\_creators” = 1**

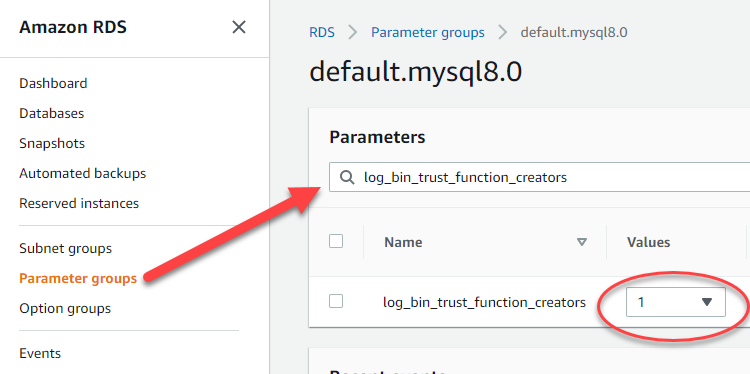
Note: The CREATE FUNCTION and INSERT statements are written to the binary log, so the slave will execute them. Because the slave SQL thread has full privileges, it will execute the dangerous statement. Thus, the function invocation has different effects on the master and slave and is not replication-safe.

You will get this error when trying to create a TRIGGER in the migrations:  
  
ERROR 1419 (HY000): You do not have the SUPER privilege and

binary logging is enabled (you \*might\* want to use the less safe

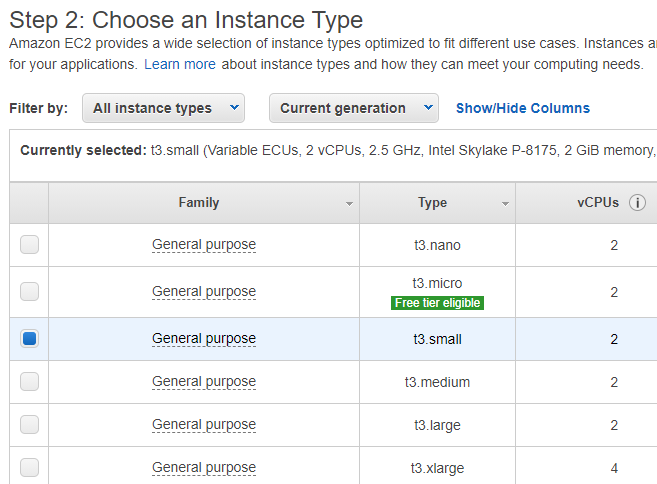
log\_bin\_trust\_function\_creators variable)

To solve this,set the global log\_bin\_trust\_function\_creators system variable to 1.

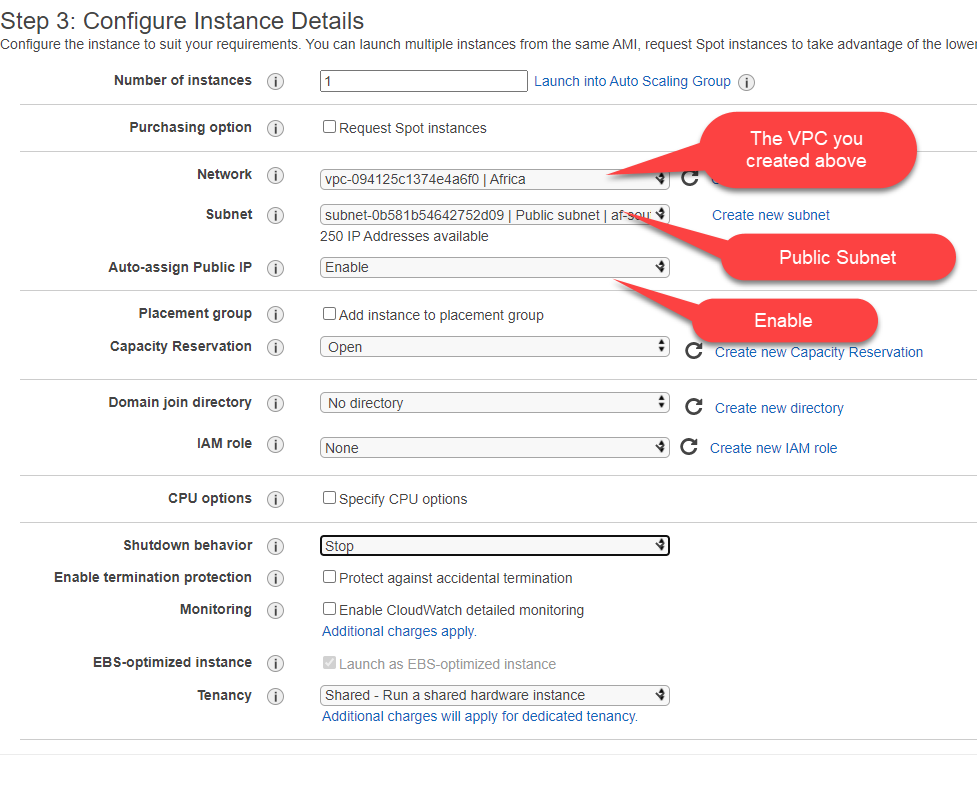


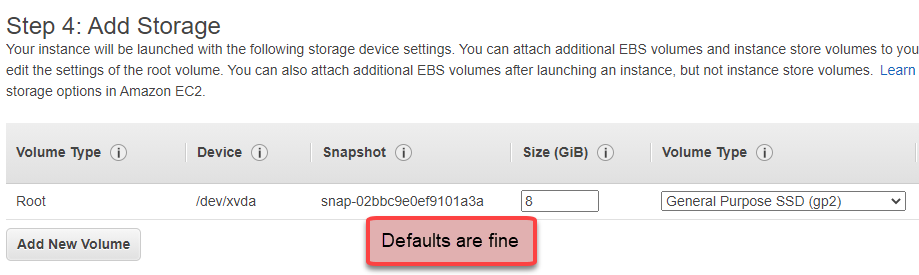
# **EC2 Server Setup – As Per Turorial**

Step 1  
  
Go to EC2 and choose **Launch Instance  
  
Choose Amaxon Linux 2 AMI**

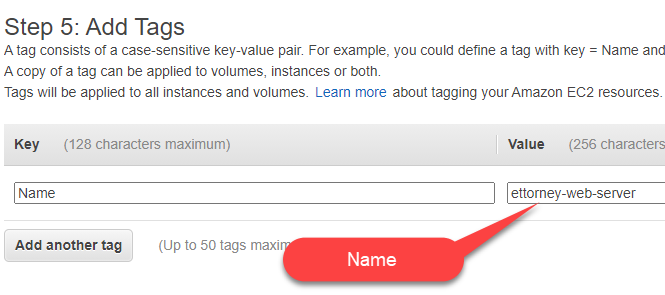


Click on **Next: Configure Instance Details**

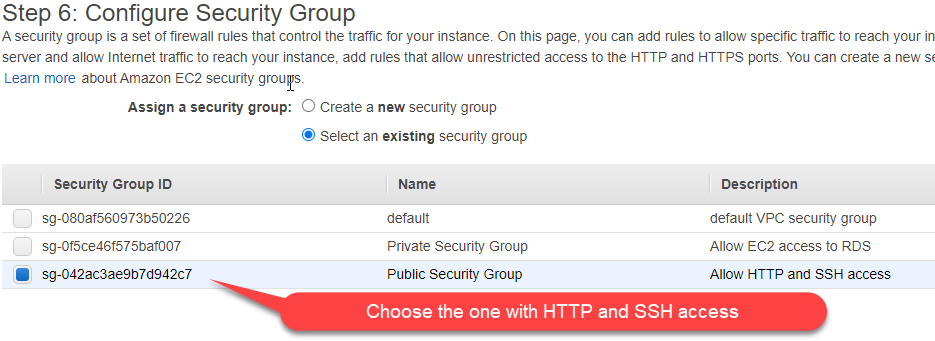


Click on **Next: Add Storage**

Click on **Next: Add Tags**



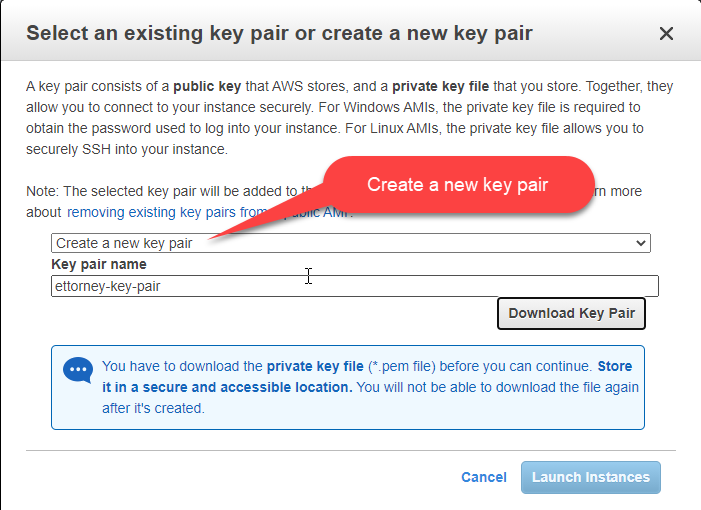
Click on **Next: Configure Security Group**



Click **Review and Launch**

Click **Launch**

**Step 7**

**Create a Key Pair to enable SSH**

# 

**Adding Apache to the EC2**

SSH into the EC2 Server  
  
**ssh -i \Users\Rick\.ssh\ettorney-key-pair.pem ec2-user@** **13.244.109.111**

**Tip**  
  
When running big updates you can start a **screen** session in your shell window. Sometimes you may experience a network interruption that can disconnect the SSH connection to your instance. If this happens during a long software update, it can leave the instance in a recoverable, although confused state. A screen session allows you to continue running the update even if your connection is interrupted, and you can reconnect to the session later without problems. See: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-updates.html>

Execute the screen command to begin the session.  
  
[ec2-user ~]$ **screen  
  
If your session is disconnected, log back into your instance and list the available screens.**[ec2-user ~]$ **screen -ls**

There is a screen on:

17793.pts-0.ip-12-34-56-78 (Detached)

1 Socket in /var/run/screen/S-ec2-user.

Reconnect to the screen using the screen -r command and the process ID from the previous command.

[ec2-user ~]$ **screen -r 17793**

When you are finished using screen, use the exit command to close the session.

[ec2-user ~]$ **exit**

Run the following:  
  
**Update Yum Repository**

To ensure that all of your software packages are up to date, perform a quick software update on your instance. This process may take a few minutes, but it is important to make sure that you have the latest security updates and bug fixes.  
  
**sudo yum update -y**  
**Install Apache**

**sudo yum install httpd -y**

**Install mod\_ssl to enable TLS (for https)**

**sudo yum install mod\_ssl -y**

**Change the Document Root to html/public**

Make ec2-user the owner of httpd.conf so WinSCP can edit it.

**sudo chown ec2-user /etc/httpd/conf/httpd.conf**

Use WinSCP or vim to edit httpd.conf and add this at the bottom of the file:

<VirtualHost \*:80>

DocumentRoot /var/www/html/public

<Directory /var/www/html>

AllowOverride All

</Directory>

</VirtualHost>

To change the Document Root for HTTPS

Add this to the bottom of **/etc/httpd/conf.d/ssl.conf**

DocumentRoot /var/www/html/public

<Directory /var/www/html>

AllowOverride All

</Directory>

To edit in WinSCP change the owner first: **sudo chown ec2-user /etc/httpd/conf.d/ssl.conf**

**Disable autoindexing from apache – to fix icons directory giving 404**

Apache redirects the icons directory for FancyIndexed directory listings. This makes /public/icons throw a 404 in Ettorney  
  
To disable FancyIndexing, you can rename /etc/httpd/conf.d/autoindex.conf  
  
**sudo mv /etc/httpd/conf.d/autoindex.conf /etc/httpd/conf.d/autoindex.txt**

**Create a certificate for HTTPS**

See: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/SSL-on-amazon-linux-2.html>

and

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/SSL-on-amazon-linux-2.html#letsencrypt>

Your instance has the following files that you use to configure your secure server and create a certificate for testing:

/etc/httpd/conf.d/ssl.conf

This is the configuration file for mod\_ssl. It contains *directives* telling Apache where to find encryption keys and certificates, the TLS protocol versions to allow, and the encryption ciphers to accept.

/etc/pki/tls/certs/make-dummy-cert

This is a script to generate a self-signed X.509 certificate and private key for your server host. This certificate is useful for testing that Apache is properly set up to use TLS. Because it offers no proof of identity, it should not be used in production. If used in production, it triggers warnings in Web browsers.

Run the script to generate a self-signed dummy certificate and key for testing.

**cd /etc/pki/tls/certs**

**sudo ./make-dummy-cert localhost.crt**

This generates a new file localhost.crt in the /etc/pki/tls/certs/ directory. The specified file name matches the default that is assigned in the **SSLCertificateFile** directive in /etc/httpd/conf.d/ssl.conf.

This file contains both a self-signed certificate and the certificate's private key.

**Note**Apache requires the certificate and key to be in PEM format

The file names and extensions are a convenience and have no effect on function. For example, you can call a certificate cert.crt, cert.pem, or any other file name, so long as the related directive in the ssl.conf file uses the same name.

**Note**When you replace the default TLS files with your own customized files, be sure that they are in PEM format.

To edit in WinSCP change the owner first: **sudo chown ec2-user** **/etc/httpd/conf.d/ssl.conf**

Open the /etc/httpd/conf.d/ssl.conf file and comment out the following line, because the self-signed dummy certificate also contains the key. If you do not comment out this line the Apache service will fail to start.

SSLCertificateKeyFile /etc/pki/tls/private/localhost.key

How to get a proper certificate:  
  
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/SSL-on-amazon-linux-2.html#letsencrypt>

**Install PHP & Extensions**  
  
**sudo amazon-linux-extras enable php7.4**

**sudo yum install php**

**sudo yum install php-pdo php-fpm php-json php-mysqlnd php-mbstring php-simplexml php-gd php-dom php-xml php-cli**

Make sure *allow\_url\_fopen* is enabled in php.ini

**Install wkhtmltopdf (used to create pdfs in Ettorney)**

1) Download it

**wget -c https://github.com/wkhtmltopdf/packaging/releases/download/0.12.6-1/wkhtmltox-0.12.6-1.amazonlinux2.x86\_64.rpm**

2) Install it  
 **sudo yum install wkhtmltox-0.12.6-1.amazonlinux2.x86\_64.rpm -y**

**Install pdftk (used by wkhtmltpodf)**

1) Use WinSCP to copy these 2 files from the dump folder on Ettorney to /home/ec2-user  
  
pdftk

libgcj.so.10

2) Move them to these folders

**sudo mv pdftk /usr/local/bin**

**sudo mv libgcj.so.10 /usr/local/share**

3) Make pdftk executable  
  
**sudo chmod +x /usr/local/bin/pdftk**  
4) Test if pdftk works

**LD\_LIBRARY\_PATH=/usr/local/share pdftk –version**  
  
5) After running composer install, change line 20 in vendor\mikehaertl\php-pdftk\src\Command.php  
  
~~protected $\_command = 'pdftk';~~

protected $\_command = 'LD\_LIBRARY\_PATH=/usr/local/share pdftk';

**Note**

**This is a bit of a hack** but we either have to change to CENTOS because Amazon Linux does not support the GCJ library used by pdftk. See <https://forums.aws.amazon.com/thread.jspa?threadID=96919>   
  
This hack it works fine – but if we run further composer updates in the future it could overwrite this change.

Maybe we fork the <https://github.com/mikehaertl/php-pdftk> github repository and use that instead?

**Install libreoffice (used to convert Word documents to PDF in Ettorney)**

**sudo amazon-linux-extras enable libreoffice && sudo yum install libreoffice**

Note: Can maybe use libreoffice-headless.x86\_64

**Install Git**  
  
**sudo yum install git -y**

**Install composer**

**curl -sS https://getcomposer.org/installer | php**

**sudo mv composer.phar /usr/bin/composer**

**Give composer execution rights**  
  
**sudo chmod +x /usr/bin/composer**

**Install NPM**

1. Download Node Version Manager  
     
   **curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash**
2. Install Node Version Manager  
     
   **. ~/.nvm/nvm.sh**
3. Install Node

**nvm install node**

**Clone the Ettorney Program from Git**

**cd /var/www/html**

**sudo chown -R ec2-user:apache /var/www**

**git clone https://github.com/LegalSuite/Ettorney .**

**Remove the dump folder**

There is sensitive data in there!!

**Install Composer Packages**

**composer install --no-interaction --prefer-dist --optimize-autoloader --no-dev**

Don’t forget to change line 20 in vendor\mikehaertl\php-pdftk\src\Command.php

**Install Node Modules**

**npm install**

**Set up the environment file**

1) Copy the example env to .env

**cp .env.example .env**

2) Edit the .env

**vi .env**

**Set permissions to various folders**

**Permissions explained:**  
  
Permissions can be represented by a string or a number:  
  
***String permissions*** are a string of 10 characters: one character that represents the file type, then nine characters that represent the file's read (r), write (w), and execute (x) permissions in order of owner, group, and others. If not permitted, the dash symbol (-) is used.  
  
-rwxr-xr--  
  
Owner can read, write & execute  
Group can read and execute  
Others can only read

A ***numeric permission*** is a string of three digits that each represent user, group, and other permissions, respectively. Each digit can range from 0 to 7, and each digit's value is obtained by summing the class's permissions:

0 means no permissions allowed.  
+1 if the class can execute the file  
+2 if the class can write to the file  
+4 if the class can read the file.

In other words, the meaning of each digit value ends up being:

0: No permission  
1: Execute  
2: Write  
3: Write and execute  
4: Read  
5: Read and execute  
6: Read and write  
7: Read, write, and execute

So -rwxr-xr-- would be 754 in numeric notation.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

METHOD #1  
  
From: <https://stackoverflow.com/questions/30639174/how-to-set-up-file-permissions-for-laravel>

**Webserver as owner (the way most people do it, and the Laravel doc's way):**

Make the apache user and apache group own the laravel root directory.

**sudo chown -R apache:apache /var/www**

if you do that, the webserver owns all the files, and is also the group, and you will have some problems uploading files or working with files via FTP, because your FTP client will be logged in as you, not your webserver, so add your user to the webserver user group:

**sudo usermod -a -G apache ec2-user**

Then you set all your directories to 755 and your files to 644... SET file permissions

**sudo find /var/www -type f -exec chmod 644 {} \;**

SET directory permissions

**sudo find /var/www -type d -exec chmod 755 {} \;**

24 Oct 2020:The seemed to work – but still having problem with LibreOffice permissions. Try add a Document Template in Ettorney. Check the laravel error log.

**Update: This may work  
  
cd /var/www/html**

**sudo chmod 777 -R storage**

**sudo chmod 777 -R bootstrap/cache**  
**LibreOffice permissions**

See:  
<https://superuser.com/questions/627266/convert-file-to-pdf-using-libreoffice-under-user-apache-i-e-when-using-php>

The apache user does not have a $HOME so libreoffice cannot run if there is no $HOME defined it does not have access to the system **/tmp** directory. A web server normally runs in a restricted environment and does not have full access to the file system for very valid security reasons.

So, you need to   
1) give apache's user a home and   
2) give it a directory it has access to to write in.   
  
So, create a tmp directory in the same folder where you store your webpage and then run the following php code:

<?php

shell\_exec('export HOME=/tmp && libreoffice --headless -convert-to pdf --outdir ./tmp /tmp/ayb/document\_34.doc');

?>

I just tested and it works perfectly on my machine. Make sure your ./tmp has its permissions set to 777.

**cd /var/www/html**

**sudo md tmp  
  
sudo chmod 777 tmp**

METHOD #2

From: <https://stackoverflow.com/questions/30639174/how-to-set-up-file-permissions-for-laravel>

I prefer to own all the directories and files (it makes working with everything much easier), so I do:

**sudo chown -R ec2-user:apache /var/www/html**

Then I give both myself and the webserver permissions:

**sudo find /var/www/html -type f -exec chmod 664 {} \;**

**sudo find /var/www/html -type d -exec chmod 775 {} \;**

**Then give the webserver the rights to read and write to storage and cache**

Whichever way you set it up, then you need to give read and write permissions to the webserver for storage, cache and any other directories the webserver needs to upload or write too (depending on your situation), so run the commands   
  
**cd /var/www/html**

**sudo chgrp -R apache storage bootstrap/cache**

**sudo chmod -R ug+rwx storage bootstrap/cache**Add ec2-user to the apache group (so we can FTP files and edit files as ec2-user from WinSCP)  **sudo usermod -a -G apache ec2-user**

Now, you're secure and your website works, AND you can work with the files fairly easily

See note about LibreOffice permissions and /tmp directory

METHOD #3

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateWebServer.html>

**To set file permissions for the Apache web server**

Add the www group to your EC2 instance with the following command.

I think the webserver group is *apache* and not *www*

**sudo groupadd www**

Add the ec2-user user to the www group.

**sudo usermod -a -G www ec2-user**

Log out to refresh your permissions and include the new www group.

**exit**

Log back in again and verify that the www group exists with the groups command.

**groups**

ec2-user wheel www

Change the group ownership of the /var/www directory and its contents to the www group.

**sudo chgrp -R www /var/www**

Change the directory permissions of /var/www and its subdirectories to add group write permissions and set the group ID on subdirectories created in the future.

**sudo chmod 2775 /var/www**

**sudo find /var/www -type d -exec sudo chmod 2775 {} +**

Recursively change the permissions for files in the /var/www directory and its subdirectories to add group write permissions.

**sudo find /var/www -type f -exec sudo chmod 0664 {} +**

This throws and error: ”unable to execute /bin/chmod: Argument list too long”  
  
Trying this:  
  
**sudo find /var/www/html -type f -exec chmod 664 {} \;**

Seems to work better – but now getting 500 error!!

**Generate an app key for Laravel**  
  
**sudo php artisan key:generate**

**Fix error with font-awesome font path**

The FontAwesome $fa-font-path was causing sass-loader errors in **node\_modules/font-awesome/scss/\_variables.scss**

Either edit it with WinSCP  
  
Or use Vim…  
  
vi node\_modules/font-awesome/scss/\_variables.scss

~~$fa-font-path: "../fonts" !default;~~

$fa-font-path: "//netdna.bootstrapcdn.com/font-awesome/4.7.0/fonts" !default;   
  
**Create the tables in the Ettorney database**

**php artisan migrate**  
  
**Generate the runtime javascript**  
  
**npm run prod**

**Start Apache**  
  
**sudo service httpd start**

**Set up automatic start after reboot**

**sudo chkconfig httpd on**