# SUMMARY:

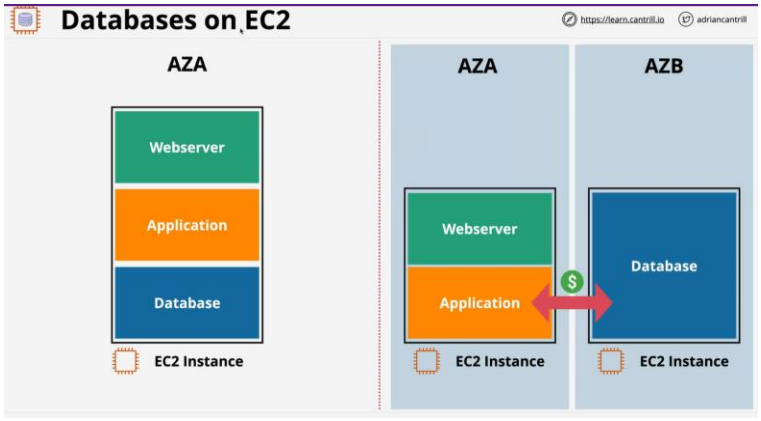
This Lab covers RDS – Migrating from monolithic, MariaDB, Muti-AZ, into Aurora, into Aurora Serverless

Contents

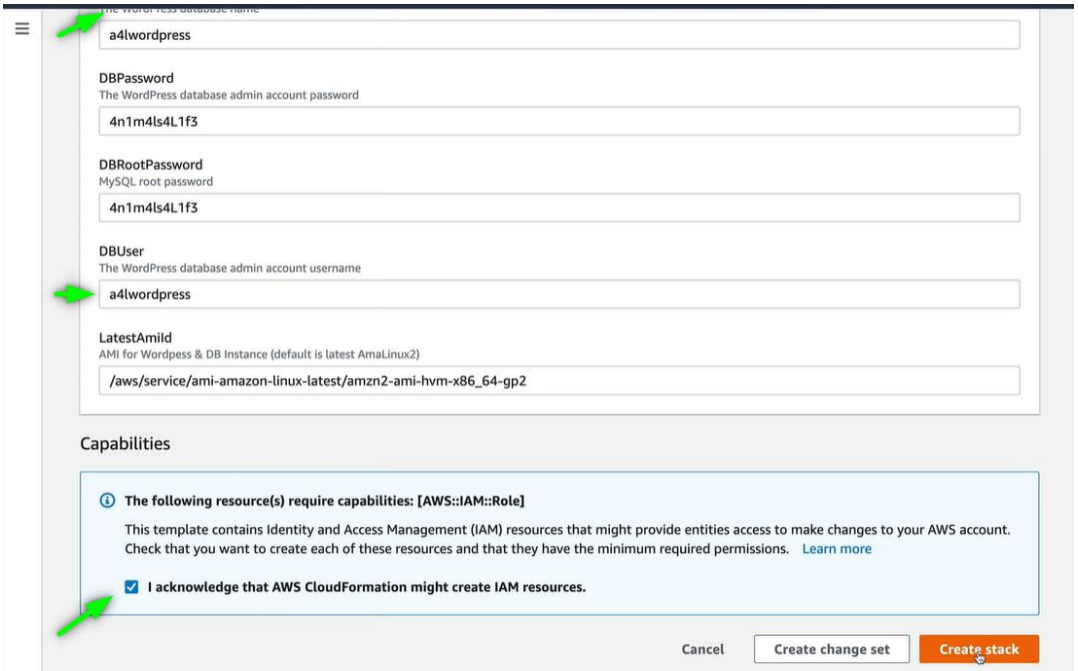


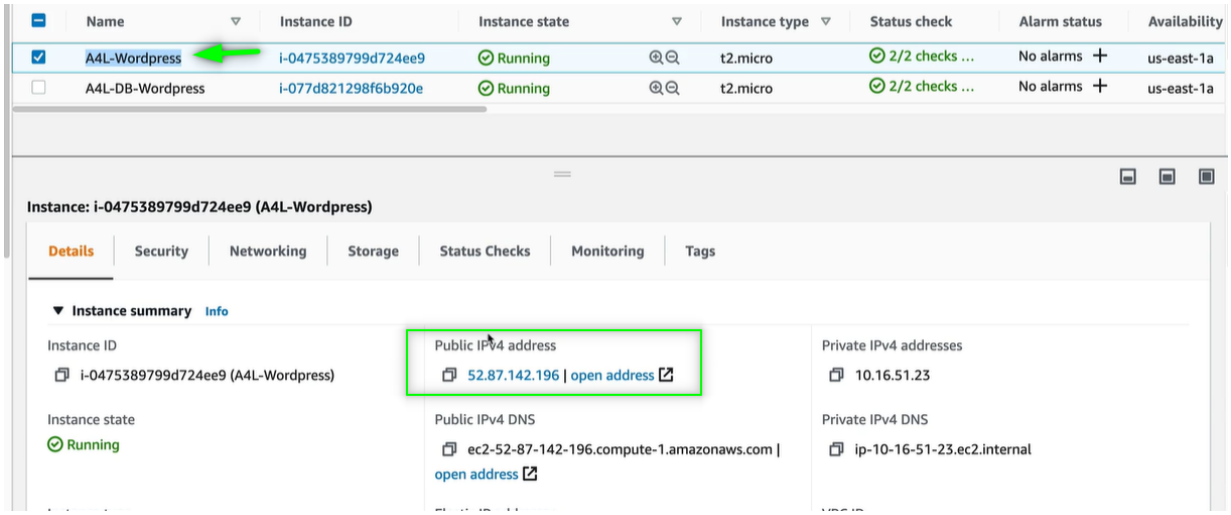
## 1. Migrating the Wordpress Monolith to a dedicated EC2 DB

The outcome of this lesson is to understand the process for performing a simple database migration manually. Taking a monolith architecture and splitting the DB out to an EC2:

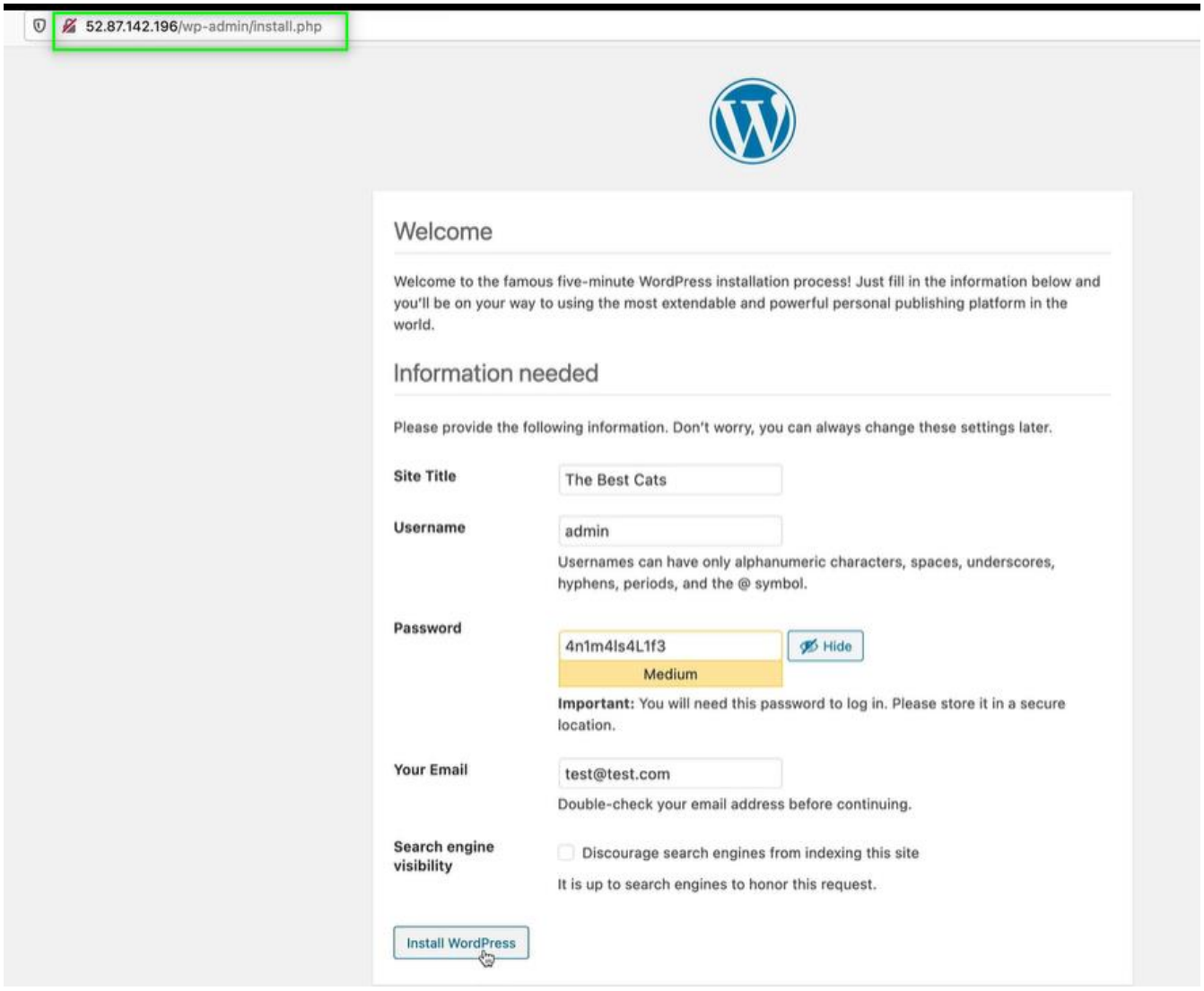



Create wordpress and db stack:





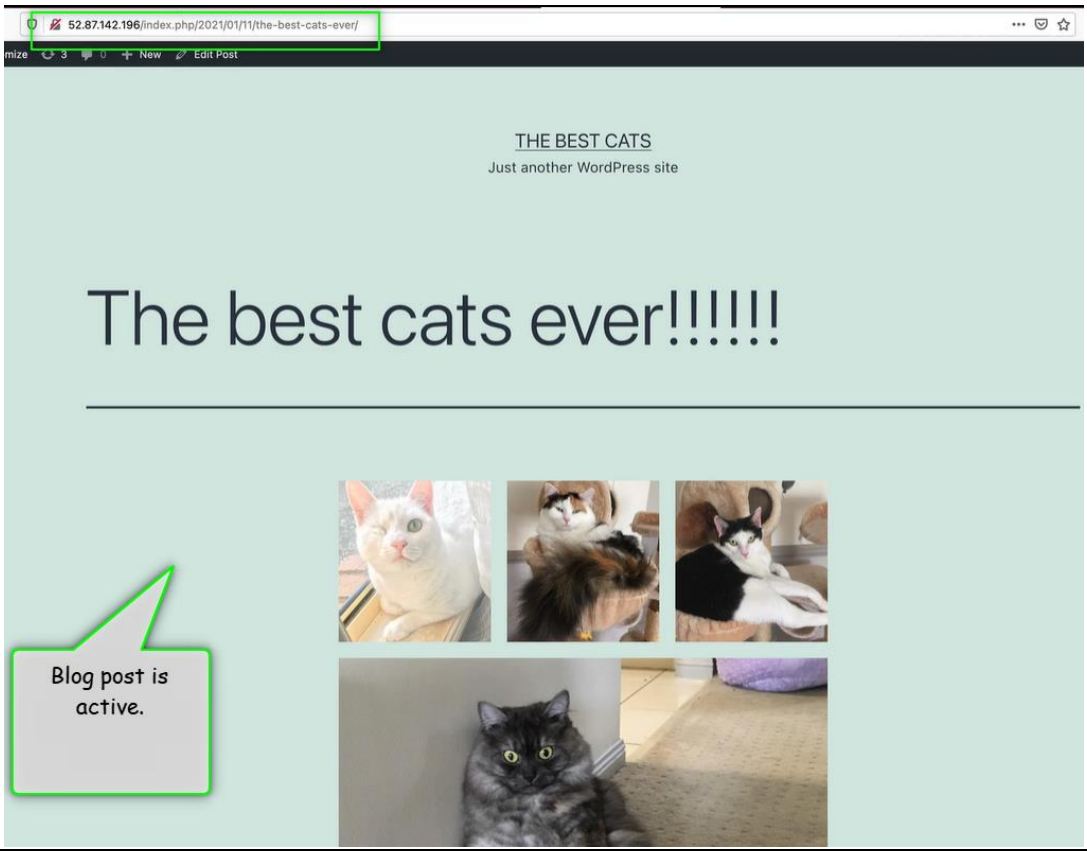

Use public IP to see the wordpress site, configure and add blogpost with images from demo file:



Login into DB and configure with private IP:

```
 GNU nano 2.9.8                                                wp-config.php

 * @link https://wordpress.org/support/article/editing-wp-config-php/
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'a4lwordpress' );

/** MySQL database username */
define( 'DB_USER', 'a4lwordpress' );

/** MySQL database password */
define( 'DB_PASSWORD', '4n1m4ls4L1f3' );

/** MySQL hostname */
define( 'DB_HOST', '10.16.60.102' );

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}
 * You can change these at any point in time to invalidate all existing cookies. This will force all users to have to log in again.


[ec2-user@ip-10-16-51-23 html]$
```

Check blog post:



## 2. Migrating MariaDB database into RDS

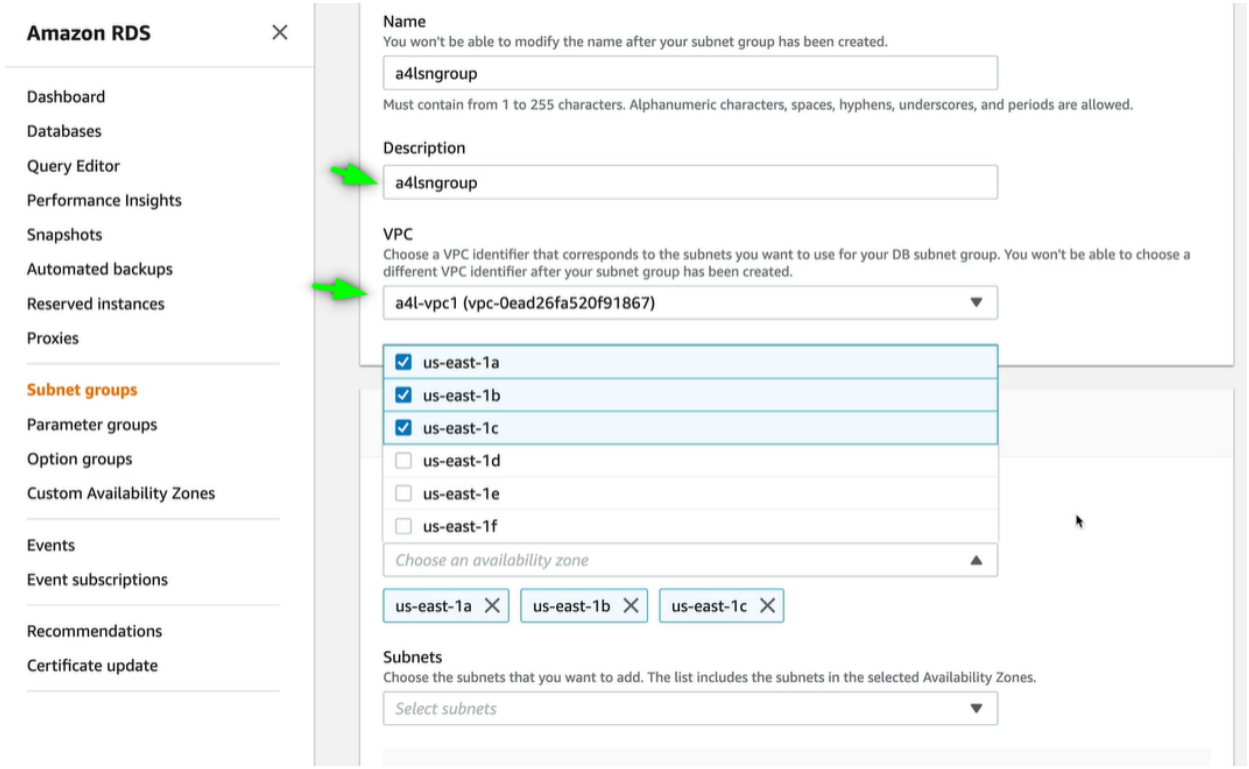
In this [DEMO] Lesson you will create a MySQL RDS instance and migrate the Wordpress Database from the self-managed MariaDB server running on EC2 into this RDS instance.

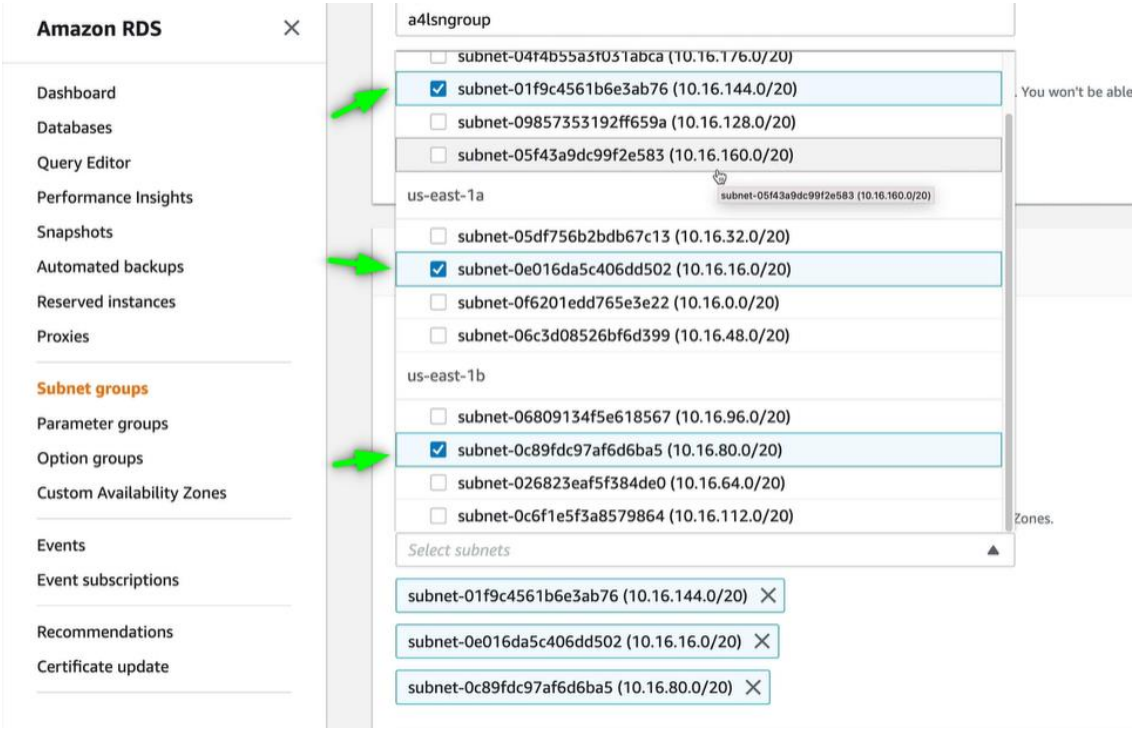
Due to the time required to work with RDS please ensure you have at least 1 hour to work through this demo.

Commands for demo:

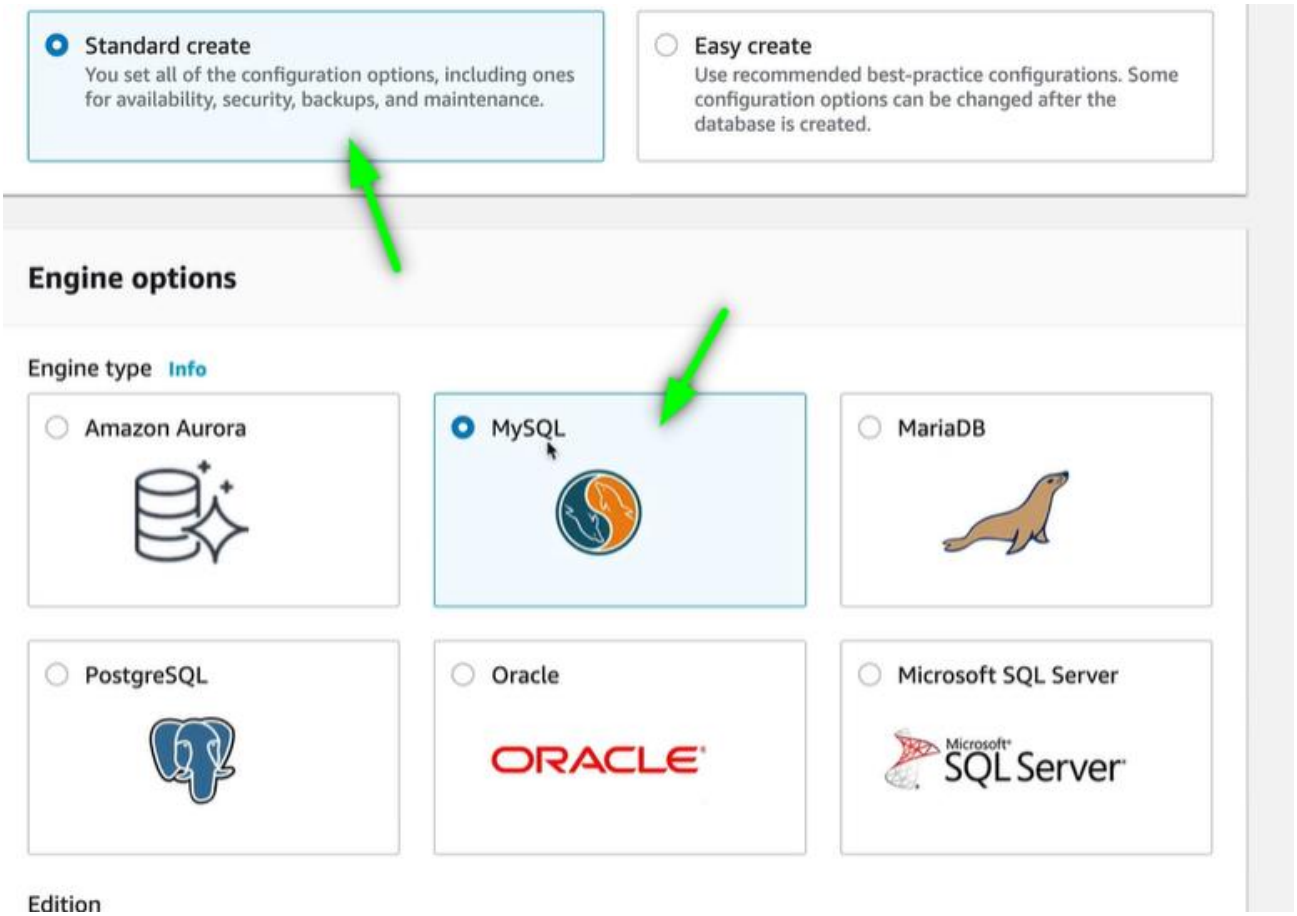
```
# Backup of Source Database
mysqldump -h PRIVATEIPOFMARIADBINSTANCE -u a4lwordpress -p a4lwordpress > a4lwordpress.sql


# Restore to Destination Database
mysql -h CNAMEOFRDSINSTANCE -u a4lwordpress -p a4lwordpress < a4lwordpress.sql

# Change WP Config
cd /var/www/html
sudo nano wp-config.php

replace
/** MySQL hostname */
define('DB_HOST', 'PRIVATEIPOFMARIADBINSTANCE');

with
/** MySQL hostname */
define('DB_HOST', 'REPLACEME_WITH_RDSINSTANCEENDPOINTADDRESS');
```

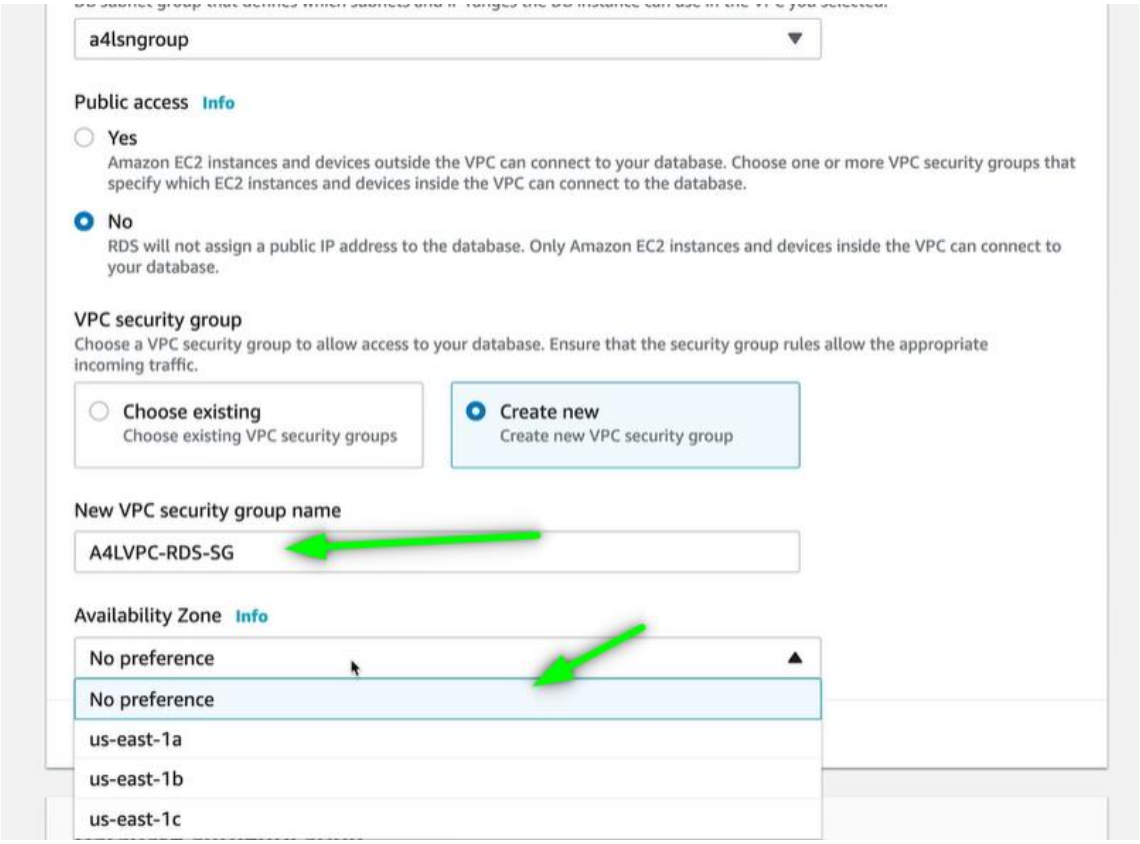
Do same steps in 1 to get wordpress stack, except create DB subnet group:

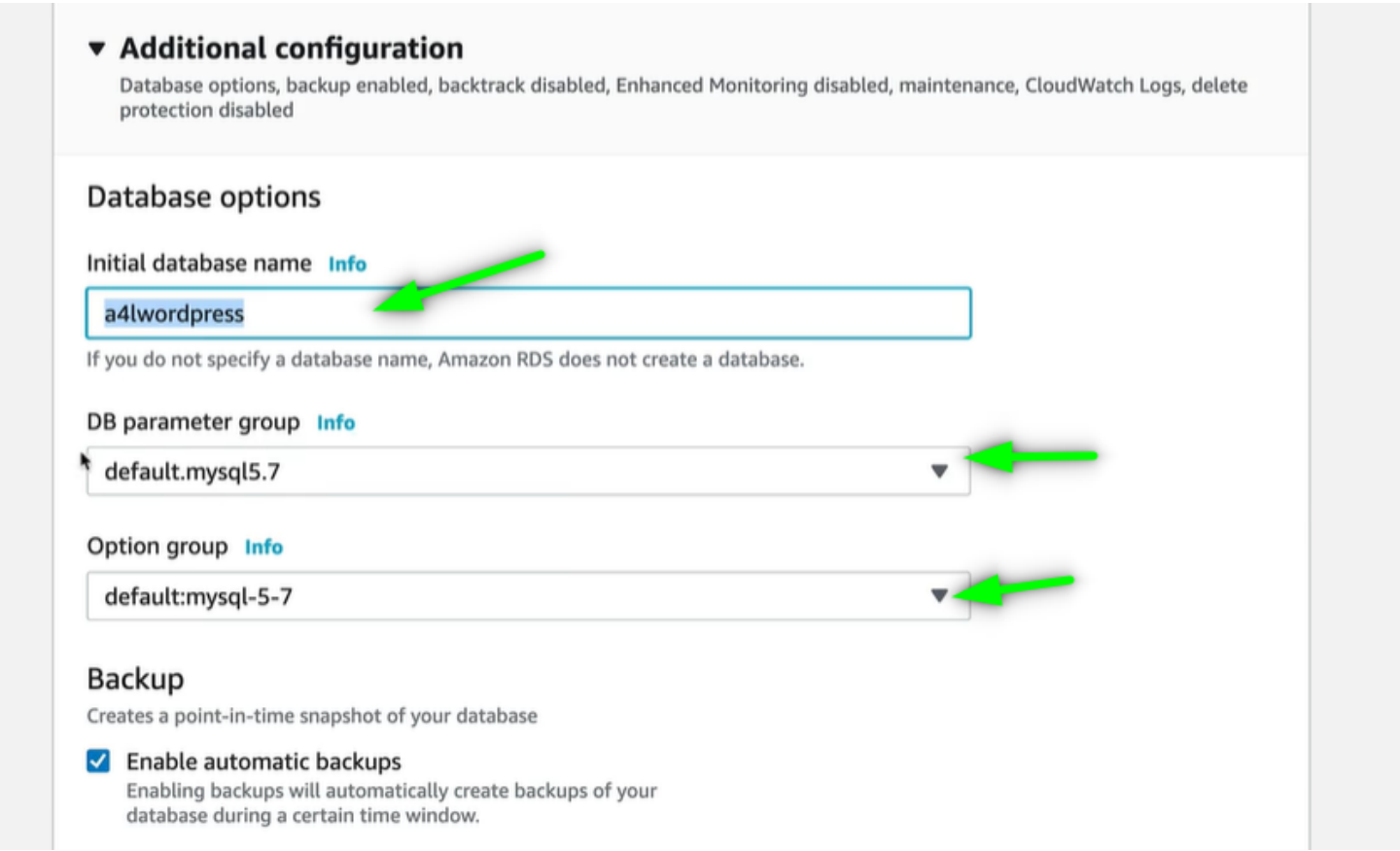


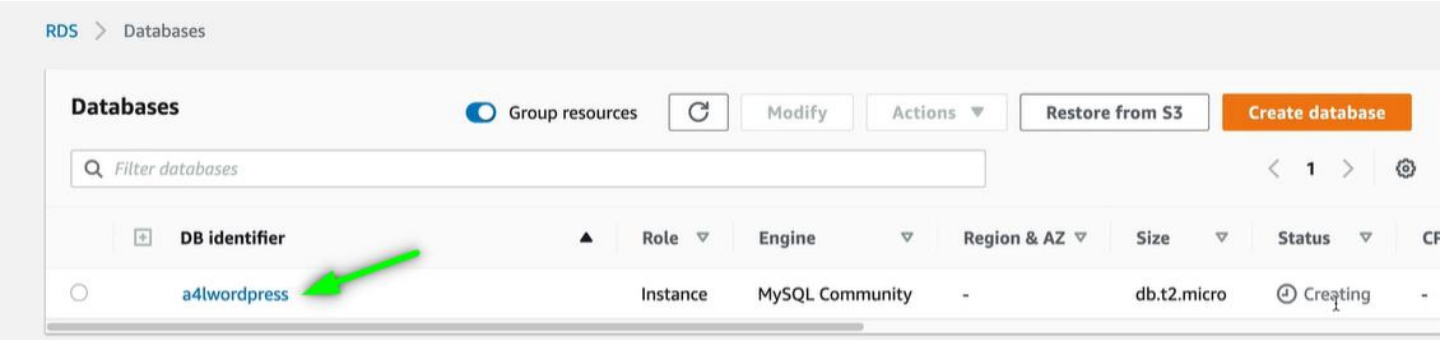


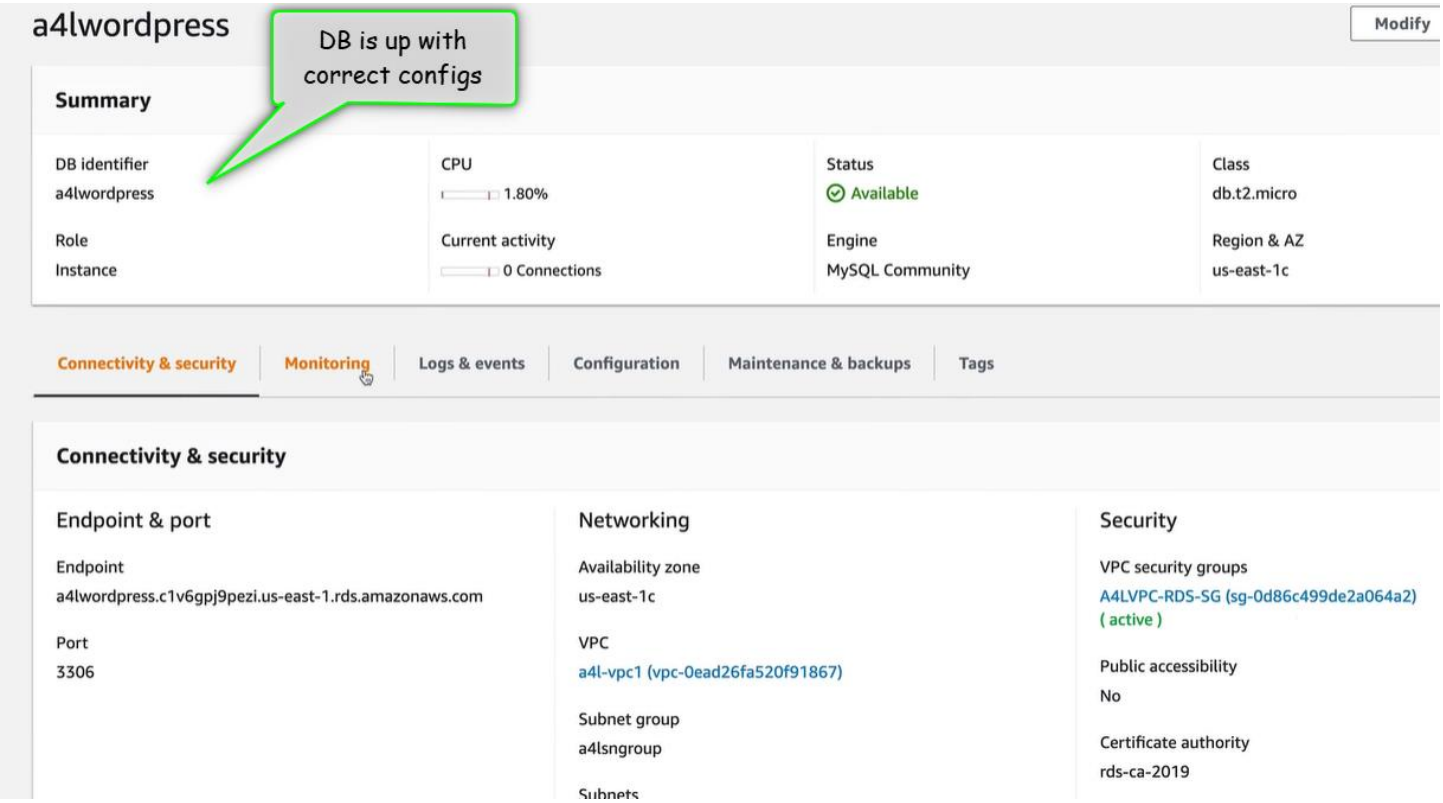

Create MySQL dvb and configure:




a4lsngroup
Public access Info
Yes
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.
No
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.
VPC security group
Choose a VPC security group to allow access to your database. Ensure that the security group rules allow the appropriate incoming traffic.
Choose existing
Choose existing VPC security groups
Create new
Create new VPC security group
New VPC security group name
A4LVPC-RDS-SG
Availability Zone Info
No preference
No preference
us-east-1a
us-east-1b
us-east-1c


Additional configuration
Database options, backup enabled, backtrack disabled, Enhanced Monitoring disabled, maintenance, CloudWatch Logs, delete protection disabled
Database options
Initial database name Info
a4lwordpress
If you do not specify a database name, Amazon RDS does not create a database.
DB parameter group Info
default.mysql5.7
Option group Info
default:mysql-5-7
Backup
Creates a point-in-time snapshot of your database
Enable automatic backups
Enabling backups will automatically create backups of your database during a certain time window.



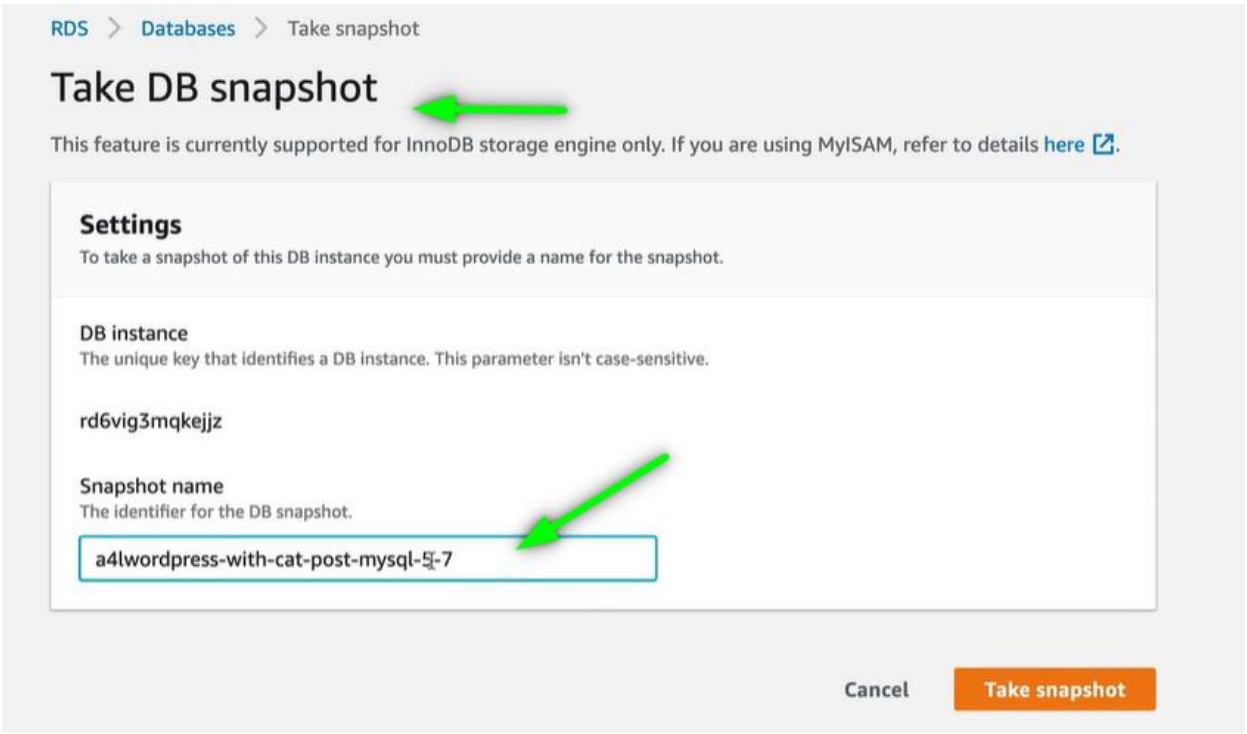

Connect to wordpress config and point DB connection to the new RDS using the Endpoint address:

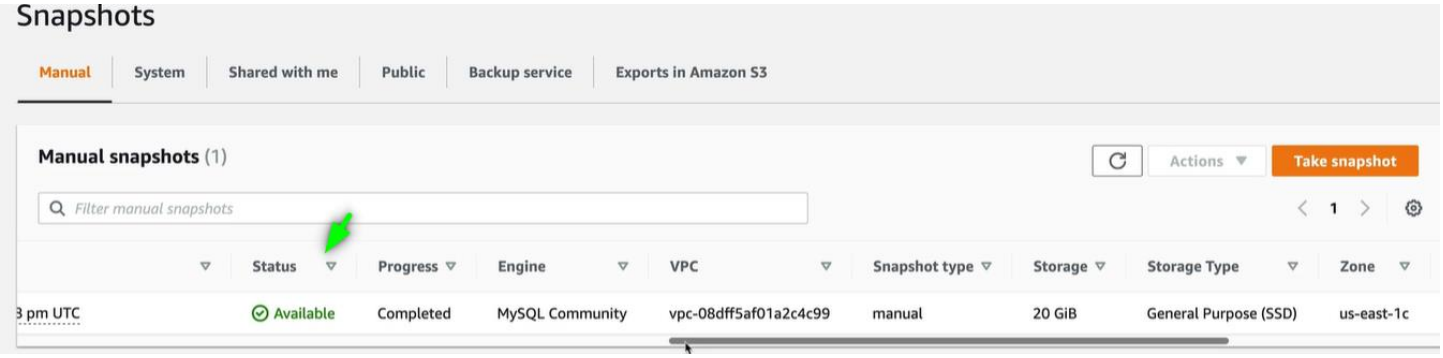
```
 GNU nano 2.9.8                                    wp-config.php

 * @link https://wordpress.org/support/article/editing-wp-config-php/
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'a4lwordpress' );

/** MySQL database username */
define( 'DB_USER', 'a4lwordpress' );

/** MySQL database password */
define( 'DB_PASSWORD', '4n1m4ls4L1f3' );

/** MySQL hostname */
define( 'DB_HOST', 'a4lwordpress.c1v6gpj9pezi.us-east-1.rds.amazonaws.com' );

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}
 * You can change these at any point in time to invalidate all existing cookies. This will force all users to have to log in again.

[ec2-user@ip-10-16-60-130 html]$
```

## 3. MultiAZ & Using a snapshot to recover from data corruption

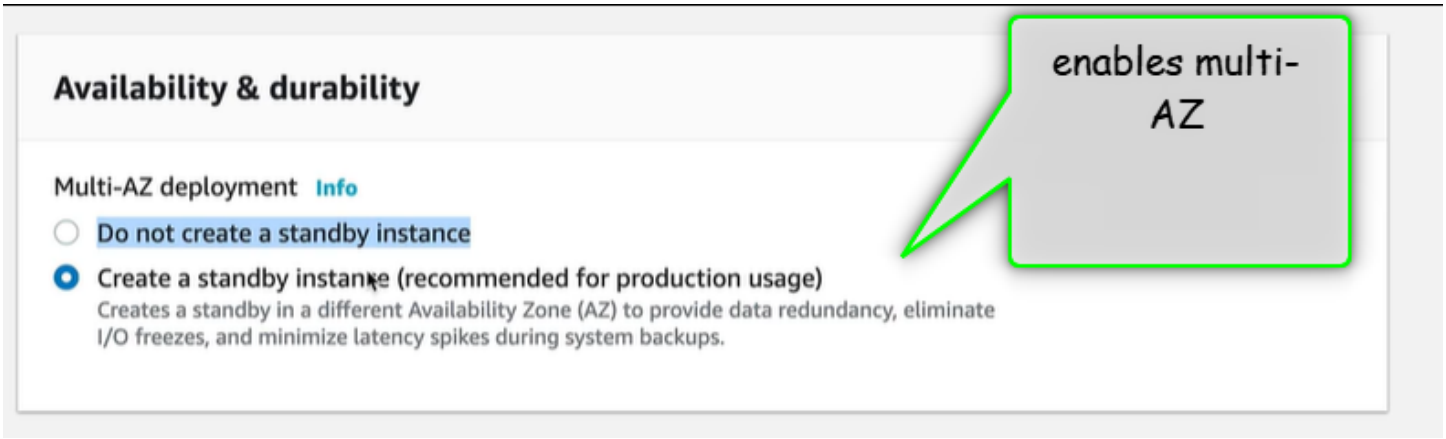
In this [DEMO] lesson you will gain experience how how snapshots and restores can be used to recover from data corruption issues.

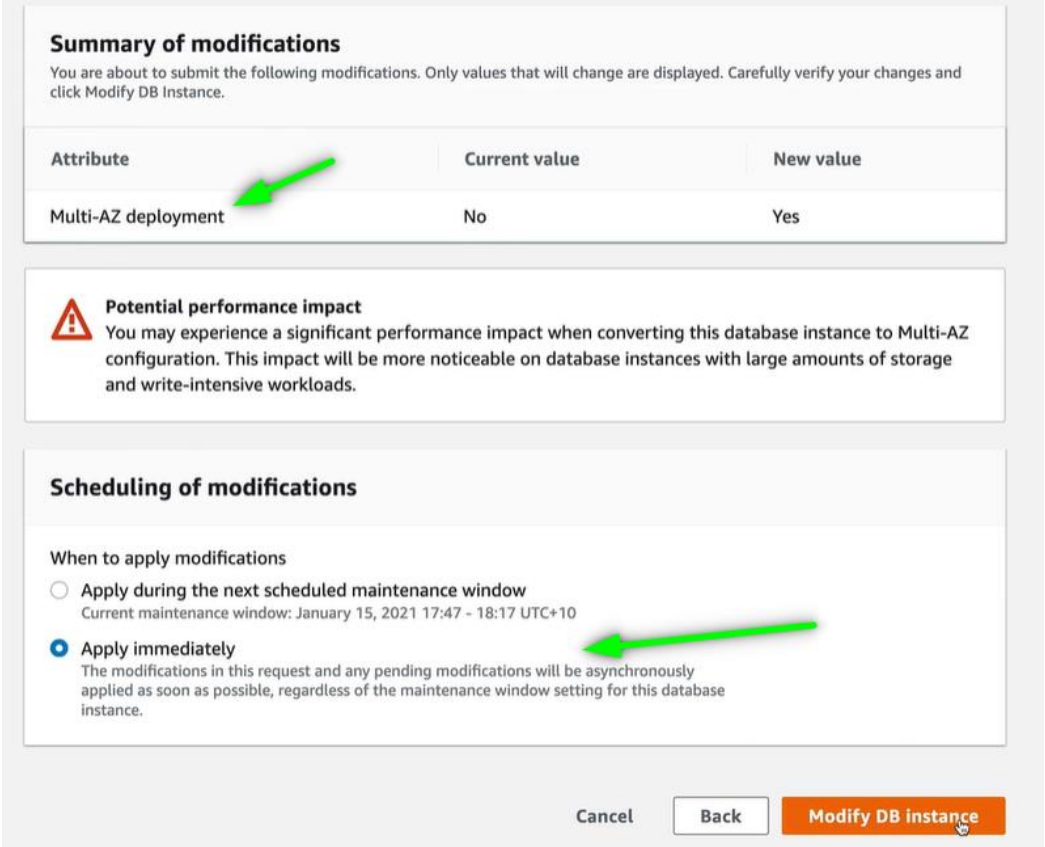
Use stack from previous, create snapshot:



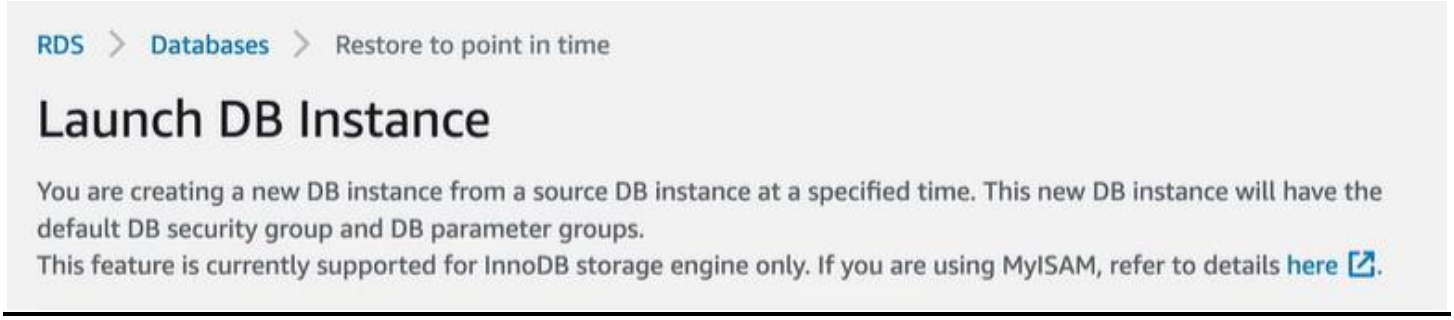



Configure DB with multi-AZ:



Restoring done here:

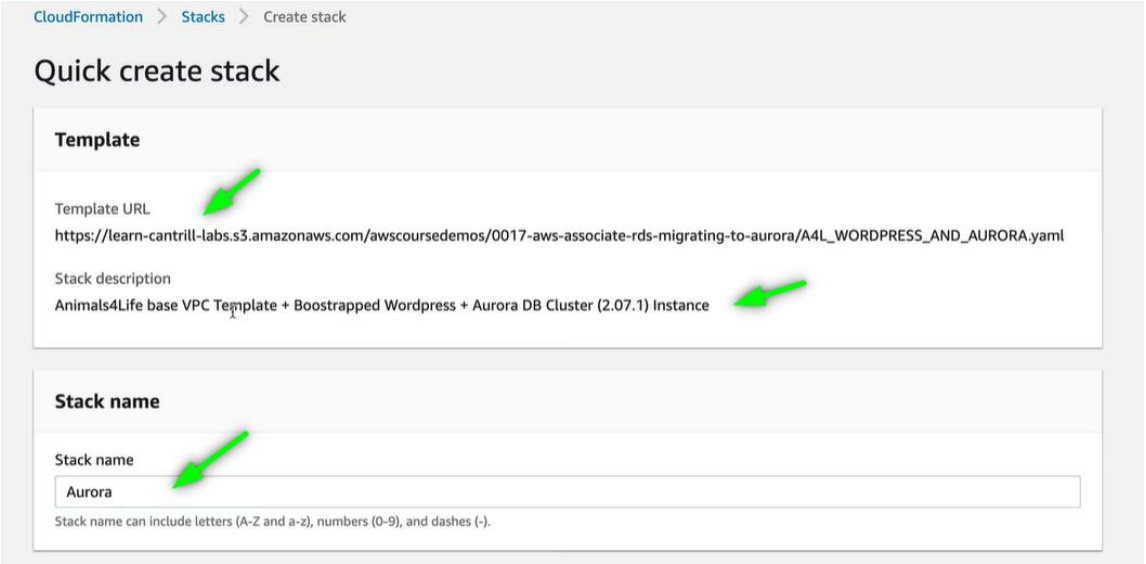



## 4. Migrating Wordpress onto Aurora

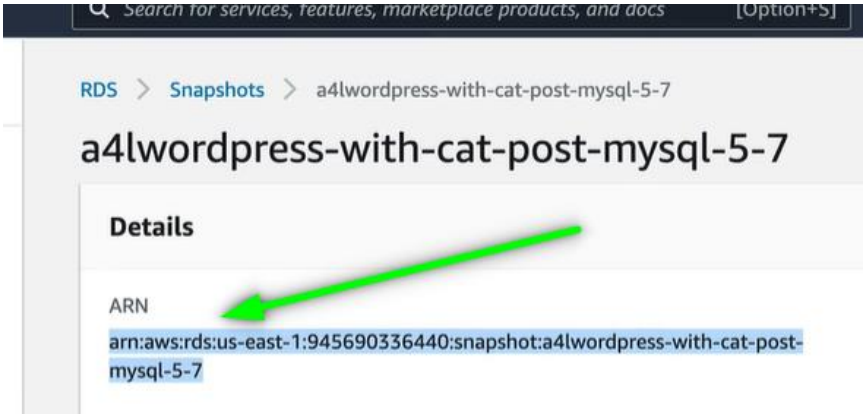
In this [DEMO] lesson you will migrate an RDS Snapshot created earlier in the course into an Aurora Cluster and verify access by provisioning the Wordpress EC2 instance.

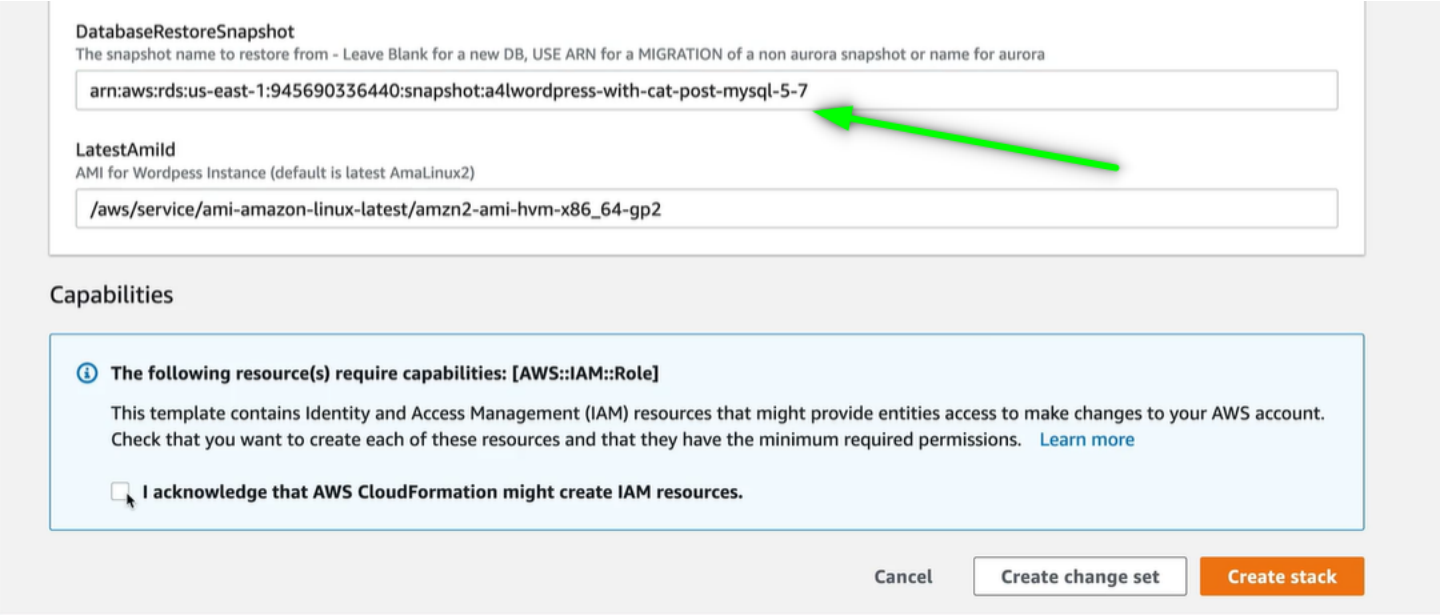
The lesson also introduces the next architectural problem which the course will resolve ... local media storage.

Use Stack to create Aurora cluster:

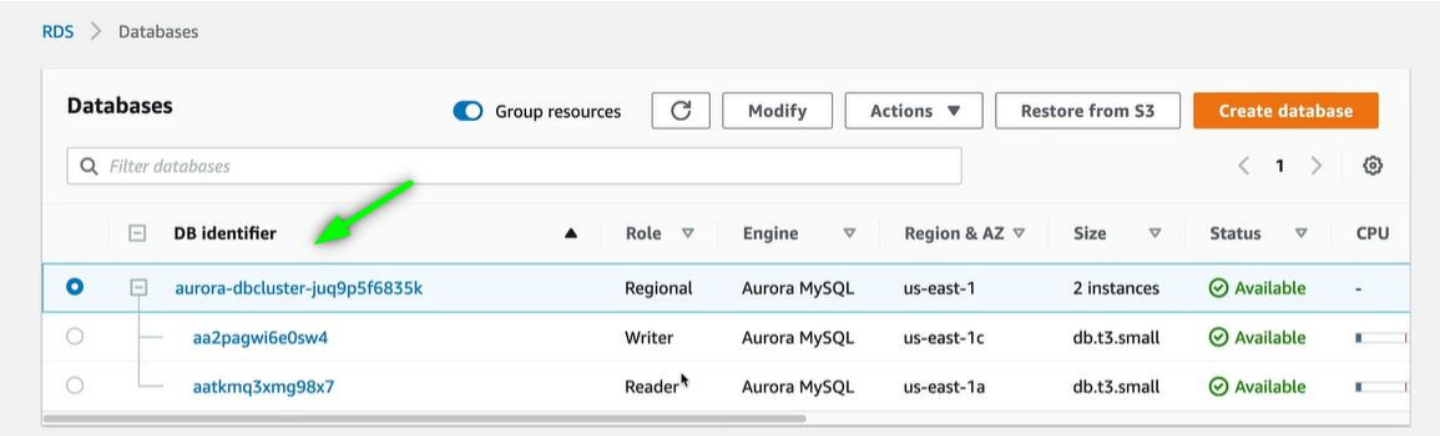



Use ARN from previous:

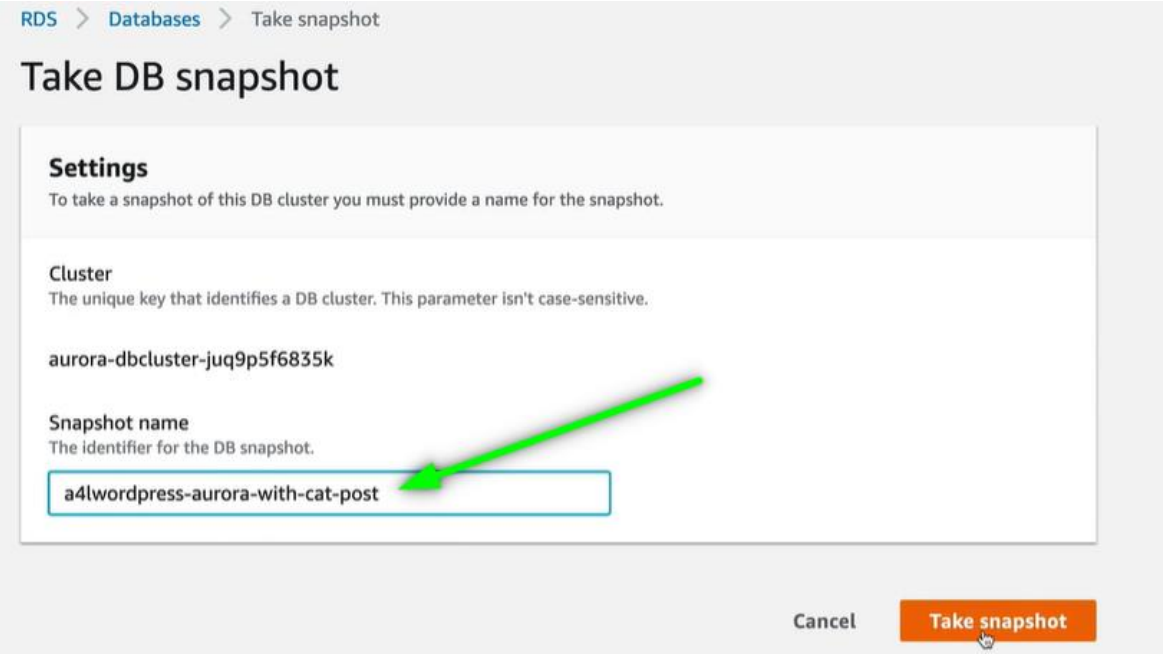




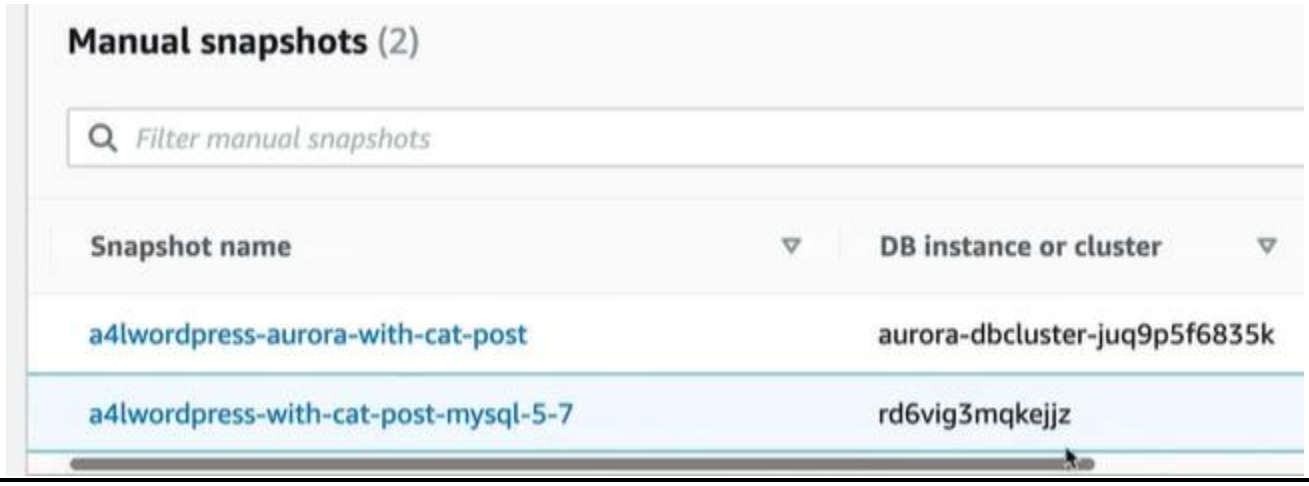

Aurora Cluster:




Take aurora snapshot:



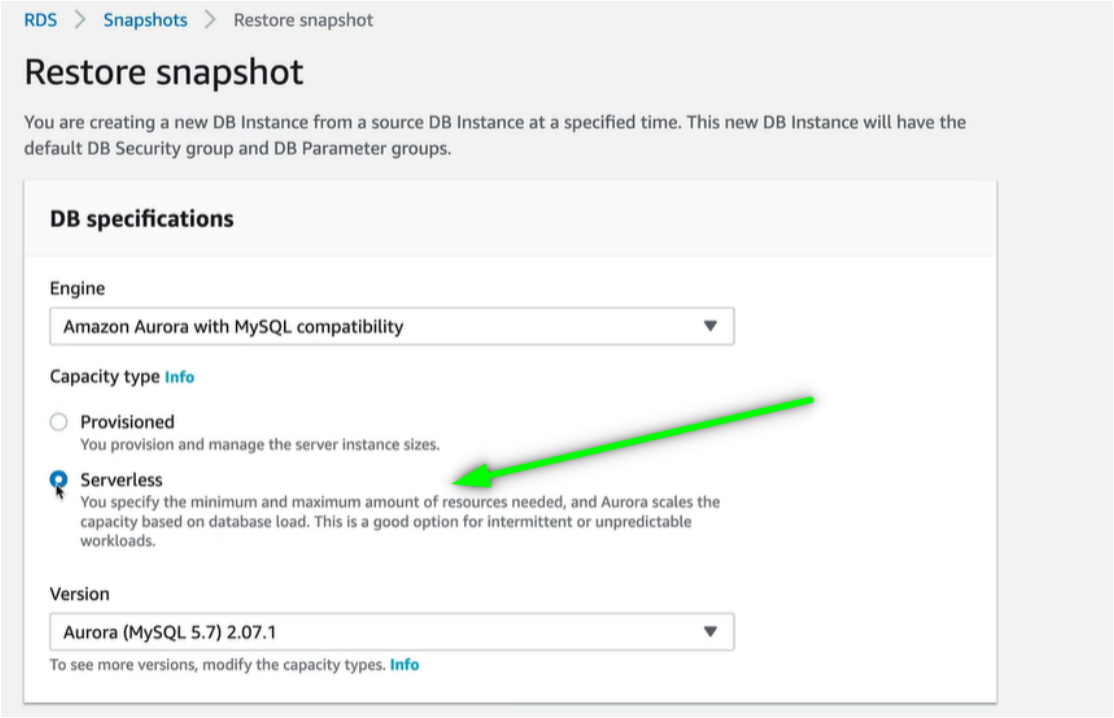

Confirm both snapshots are present – MySQL + Aurora:



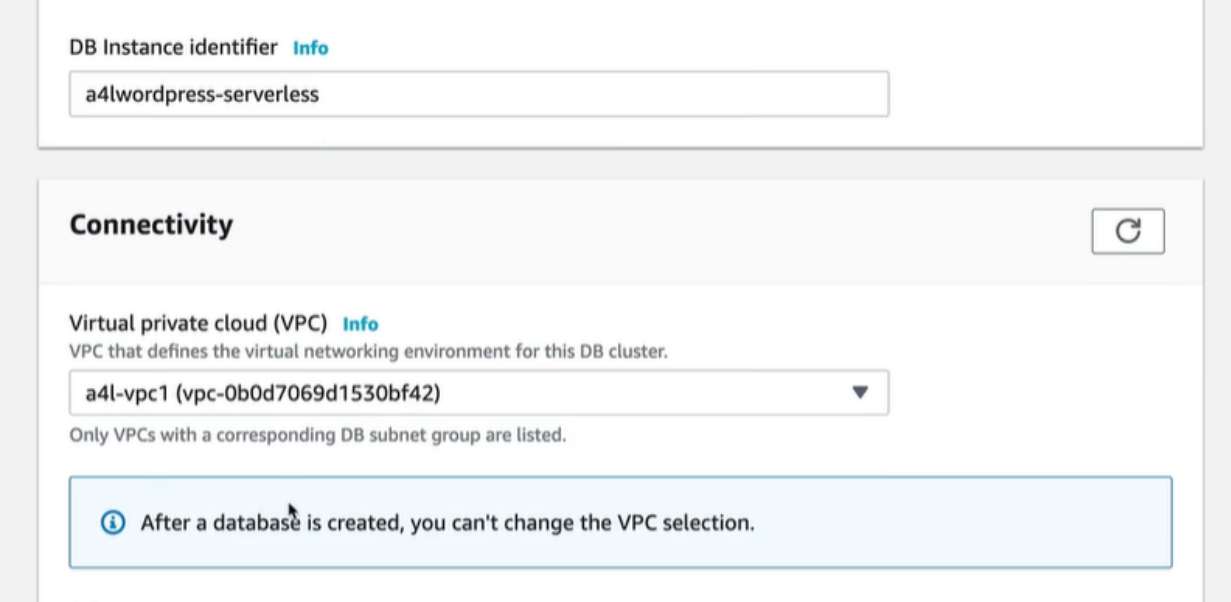

## 5. Migrating to Aurora Serverless

n this [DEMO] lesson you will experience how to migrate an Aurora provisioned snapshot into an Aurora serverless cluster. In addition you will see how an aurora serverless cluster can scale down to 0 ACU and pause - meaning the cluster costs will be for storage only. You will experience how when incoming load reaches the serverless cluster it will unpause and allocate ACU to begin servicing requests.

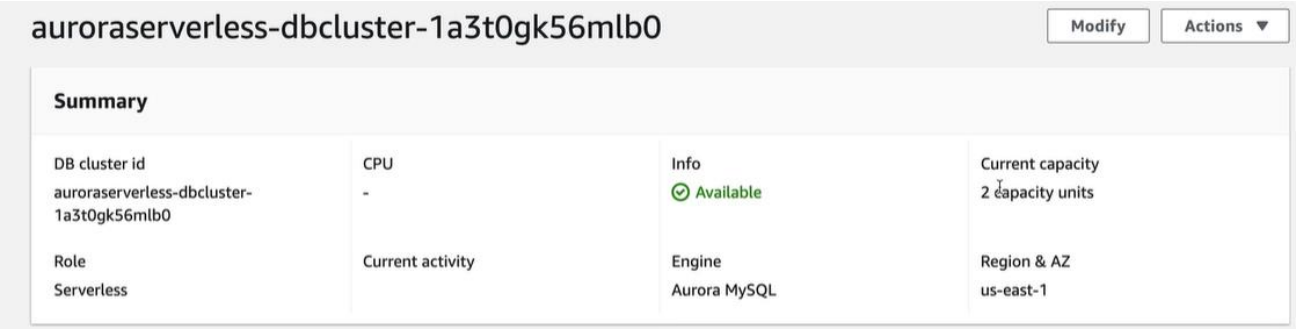
Use same stack, point to Aurora snapshot – restore to a serverless:




Establish DB instance identifier and connectivity:




Explore serverless options:




Big advantage is pay model changes to 'pay what you use'