Aurora Database (RDS) Cluster Cloudformation Definition

An Amazon DB cluster is made up of DB Engine instances and a cluster volume that represents data copied across the Availability Zones as a single, virtual volume. There are two types of instances in a DB cluster: a primary instance and Replicas.

The primary instance performs all of the data modifications to the DB cluster and also supports read workloads. Each DB cluster has one primary instance. AReplica supports only read workloads. Each DB instance can have up to 15 Replicas. You can connect to any instance in the DB cluster using an endpoint address.

To implement our Database cluster we are going to use the Aurora DB Engine, Amazon Aurora (Aurora) is a fully managed, MySQL-compatible, relational database engine that combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. It delivers up to five times the performance of MySQL without requiring changes to most of your existing applications.

The following diagram illustrates the relationship between the Amazon Aurora cluster volume and the primary and Aurora Replicas in the Aurora DB cluster.

Its time to build our Aurora DB cluster in our Yaml file. We started with our 4 main parts: Description, Parameters, Resources and Outputs.

The description will be a sentence as usual, Aurora Database Cluster with one replication instance and Multi Availability Zones.

Lets check our main architecture template to verify the parameters we passed to our DB Cluster template:

Almost every parameter we are going to need in our DB cluster come as an output from the VPC module, and only one parameter come from the stack setup, so, from the VPC output we have:

The Subnet one , the Subnet two, the Availability Zone 1 and the Availability Zone 2.

From the stack setup we will need the DB Password access.

Going back to our DB Cluster template file we will declare this 5 parameters with type String.

Next section is the Resource, The Aurora documentation specify that we will need a VPC with at least 2 subnets in at least 2 different availability zones, so the first resource we are going to need is the RDS Subnet group.

We are going to call this resource DBSubnetGroup and will be an AWS::RDS::DBSubnetGroup type, this one, creates an RDS database subnet group. Subnet groups must contain at least two subnet in two different Availability Zones in the same region. And we already guarantee this in the VPC module implementation.

The properties for this resource will be:

- DBSubnetGroupDescription, as the Description section, we just type in a description here, Database Subnet Groups.
- Next the SubnetIds, here we are going to list the The EC2 Subnet IDs for our Subnet Group. Referencing from our parameter section we have Subnet1 and Subnet2.

The next resource we need to declare is the DB cluster itself. We are going to named RDSCluster and will be an AWS::RDS::DBCluster creates a cluster, such as an Aurora for Amazon RDS (Amazon Aurora) DB cluster.

Next we declare the properties for our Database Cluster. The first we are going to need is the Database master username, we are going to type in here, admin.

Second we have the database master password, this will be a reference from our parameter section.

Third the database name, lets call it MyDB

Next we are going to set up our database engine, with the Engine property. Here we are going to type in Aurora.

Then we have the DBSubnetGroupName, at this point we just reference the resource we just created before this one, and the last property will be

The DBClusterParameterGroupName allows us to define parameters that apply to the whole cluster, there is also the possibility to implement parameters to specific instances inside the cluster. Cluster-level parameters are managed in this DB cluster parameter groups property and we are going to reference here the next resource we will create in the template. In other hand the Instance-level parameters are managed in the DB parameter groups.

Although each instance in an Aurora DB cluster is compatible with the MySQL database engine, some of the MySQL database engine parameters must be applied at the cluster level, and are managed using DB cluster parameter groups.

Our next resources are the DB instances, we are going to create 2 similars DB instance, but one we want to live in the Avalability Zone 1 and the other will be in the Availability zone 2, the one in the AZ1 will be our primary instance, and the other in the AZ2 will be our replica.

Lets start with the RDSDBInstance1: the type for this instance will be AWS::RDS::DBInstance, and the properties will be:
- DBSubnetGroupName, or the Database Subnet Group Name, as you are probably guessing, this will be a reference to the resource we created above.
- DBParameterGroupName, this one will be a reference to the resource we need to create next.
- We specify the Engine, and will type in Aurora.

- DBClusterIdentifier, here we have a reference for the DB cluster created before.
- We DONT want this DB Instance to be publicly Accesible, so we set this property to false, because we are going to be accesing only over the Subnet.
- Next the Availability Zone, here we are going to reference the AZ 1 and of course in the RDS instance here we will reference the AZ 2.
- And the last property will be the DB Instance Type, here we type in: db.r3.xlarge

As we mentioned before the next resource will be another DB instance, with the only different that the Availability zone , will be a reference to the AZ 2.

Then we have the last 2 resources the RDSDBClusterParameterGroup and the RDSDBParameterGroup both specify parameters for the whole cluster and instances, the only difference is that the parameters specified at the cluster level contemplate global parameters, like the time zone, and the other one are more related to the instances, for exampe the sql mode. Of course in both side we can specify the description for the cluster and the instances, and also the Engine family, in our case, Aurora 5.6

With this we finnsh our template, we are not going to add any outputs for this template, but as a homework you can add a DB cluster url output to use it on your application db connection set up.

In the next lecture we are going to create our Stack using the AWS Console and will try to analize all the module created.