Tweaking the app and architecture.

So we already built our architecture and pipeline. Let's now tweak it a little bit to get a better performance and nicer parameters handling in our app.

Lets open our app index file and check the 3rd and 5th line this one defines our Db cluster url and our Db password

Lets delete both values and set up our yaml files to fill in this 2 values during the codepipeline execution.

The first thing to set up our yaml files:

Lets open our database cluster yaml file and add the outputs section at the end of the file

We will need to output our Db Cluster Endpoint so we type in dbClusterURL:

Set a description Database Cluster URL Endpoint and finally use the Get Attribute function with the attribute Endpoint.Address

With this we will have access to our Db Cluster endpoint URL in our main architecture yaml file

Lets go there and add the new parameters to our CodePipeline module

First we will need to add the parameter DbPassword, this one we already have it from our parameters section and the DbClusterEndpoint parameter referenced as an output from the DatabaseCluster output we just created
Its time for our Deployment Pipeline yaml file this is the one capable to do the whole set up we need

First we are going to add our 2 new parameters in the parameters section
Lets add DbClusterEndpoint: with the string type and we are going to do the same process with DbPassword parameter

Lets go now to the CodeBuildProject resource

We want to add this 2 new parameters into our index file replacing the line 3 and 5
Lets do this using the commands in the prebuild stage

We are going to be using the Environment variables property to setup our Dbcluster url and DB password and used in our commands

So lets type in first the commands and then will setup the environment variable property needed

We are going to use a simple sed command to replace the whole line 3 in our index.php file

We are going to add a new commandafter the second printf and type in
sed -i "3s/.*/ define('DB_SERVER', '"$DB_CLUSTER_URL"'); /" sources/index.php
With this command we are replace the third line with the string define('DB_SERVER',
and the environment variable DB_CLUSTER_URL in the files index.php inside the sources folder

we will do the same with the 5th line
sed -i "5s/.*/ define('DB_PASS', '"$DB_CLUSTER_PASS"'); /" sources/index.php

but of course the environment variablewill be a different one.

We are almost done we just need to setup our 2 environment variables to achieve it
we go to the EnvironmentVariables property and add DB_CLUSTER_URL referencing the DbClusterEndpoint declared in the parameters section and add DB_CLUSTER_PASS referencing the DbPassword also declared in our parameters section

Lets now test everything in our stack

First we upload everything the 3 files modified to the S3 bucket, do not forget to also upload the zip file with all the files in it.

Then we go to our Cloudformation Service screen click Actions and select the Create Change set option

Then we type in our main architecture url and click next

Once here we add a name and a description for this newly created change set and click next until we get to the review screen

here we can check all the modules affected and click the execute button

After some minutes we will have everything ready to work we go to our shell console and push our changes

After some seconds our codepipeline will start to work

Passing first for the Source stage

Then the Build stage, here we want to check the log details to see whats happening

The first thing we notice is the pre-build commands executing then the docker build execution doing each step we declare in our dockerfile

finally everything is pushed to the ECR repository

Lets go back to our codepipeline and check the Deploy stage

We can also go in details here

Lets go to the EC2 container service screen, click our Service then the service name and open the events tab

Here we can check our Deploy working, starting 2 new task, registering 2 targets into the targetgroup, deregistering the old 2 targerts, drainning the connections to this old targets stoping the 2 task and finally reaching a steady state with the new task

We can now open our loadbalancer url from the outputs in our cfn service screen and will see our application running smoothly

Let go back to the ECS screen and open the metrics tab

Here we can monitor our cluster execution with the CPU and memory utilization graph

We can also go to the events tab and click in our target group link above

This will open our target group definition

Here open the Monitoring tab and you can verify all the functioning details for our application

Healthy and unhealthy host registered

Average latency
Requests
200 and 300 code answers
500 and 400 code errors and more

This is a pretty cool diagnostic of our app running

Lets check more metrics

Go to the Cluster link and check the cluster metrics

here we can check the cpu utilization and reservation and the memory utilization and reservation

Lets get an even better memory reservation definition

In our service definition yaml file

we defined a Memory property with a 490 value this one is telling cloudformation to create the task with 490 MiB of memory to reserve for the container, but if the container attempts to exceed the allocated memory, the container is terminated.

We don’t want our container to be terminated instead, we want to utilize all the memory available in the instance, because our architecture is not scaling inside the container, instead our architecture is scaling in more ec2 instances.

For this reason we need our container to be able to take everything he needs from the machine hosting it.

To achieve it, we just want to change this property to MemoryReservation with this we are stablishing a memory reservation and not a memory limit.

Lets save everything into our S3 bucket and update the Service module through the cfn service screen

We follow the same process we just did before...

And its now ready, our app and architecture have been tweaked for a better performance
and parameters handling

We also checked how easy is to modify a module in our architecture definition and of course how easy is to modify our codepipeline definition.

The next lecture will be our final lecture there, we will talk about the conclusions
the courses coming and more...