Defining our app & Dockerfile

Lets build a simple php app to use with our Pipeline and Architecture.

This one will be a simple to do list able to connect to the database Cluster, insert a new task to the todo list, mark the task done, and of course list the todo items.

We want this app to run in our ECS cluster so we will need to define a docker configuration.

We want Docker to build the image with the parameters specified in that docker configuration document.

This docker configuration is called Dockerfile

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

Besides our app and its dependencies I want to be able to check and admin the Database through a web view for that purpose I'm going to use adminer

Adminer is a full-featured database management tool written in PHP.

It consist of a single file ready to deploy to the target server.

So our App will have the adminer and the files needed for our app to work.

Lets do a briefly check of our app.

Everything is in one file
First we see all the functions and database interactions defined in php language
Then we have some html forms and some php function calls

As you can see we defined the database parameters connection at the beginning of our file we are going to set this parameters manually But in the next lecture we are going to set this parameters automatically during the build stage, using shell commands

So the first parameters is the
DB_SERVER, we go to the DB cluster service
Copy the endpoint value and paste here.
We know the DB password, and just type it here

With this our app is ready to connect to our DB cluster.

In this lecture we are going to use the adminer to add all the tables needed for our app

In the next lecture we are going to add this validation and creation in our PHP application.

Now we want to build our dockerfile

The first we need is the PHP official image this one includes the Apache webserver and the PHP version 7 Then we want apt-get to update and install git, apache mod rewrite, the pdo mysql extension and grants all the privileges to the apache user.

Next we want to copy our source code to the /var/www/html directory this is the default Apache root folder

Next we want to able to handle user sessions in our app the actual app is not using user login but this will be great for your applications So we create a folder out of the html folder and give it write permissions

Lastly we want the Apache user default to be the owner of everything inside the /var/www/html

So now lets check the Docker magic and run this app locally

First we are going to build the Dockerfile
Then we are going to run it at this point we can check locally our app

Opening a browser in localhost pointing to port 8080

You will be able to check the adminer running locally but the app will try to connect to our private DB Cluster an eventually will give you an error.

Lets check the magic here, we are using docker to run this container like a some kind of virtual machine under our linux and will have exactly the same container running in our AWS ECS cluster

Lets create a change set for our stack and add our Deployment pipeline to our architecture lets go to our Cloudformation service screen and there click the Action Create a Change set for our stack

We already upload our main-arch and Deployment pipeline yaml files to the S3 that contains all our architecture files

In this screen we just type in our S3 url pointing to the Main architecture file and click continously until we get to the briefing screen

here we can check all the changes applied and click the execute button

Here you will start checking the changes and will appear the new Deployment Pipeline module

Ok, now we have our Deployment pipeline ready for our application

Lets push the app to our codecommit repository

You can check the application folder structure

The dockerfile is on the root and the application files are inside the sources folder.

To push our code we use the command

git add -A
git commit -m “Initial commit”
git push origin

We previously clone this codecommit repo into our local machine

After some minutes

You will see our Codepipeline working
First the source stage
This one prepare our Dockerfile, source code and the architecture templates files
and pass to the build stage

Once in the build stage CodeBuild will generate our image from the Dockerfile
and push it to the ECR repository

Next will take place the deploy stage
Here we are going to add our app to ECS cluster. Once this one finnished we are ready to test our app.

Just open the browser pointing to the Loadbalancer URL and start adding task to our todo list

So our app is ready and running in our architecture

In the next lecture we are going to tweak our app and architecture for a better performance and nicer parameters handling.