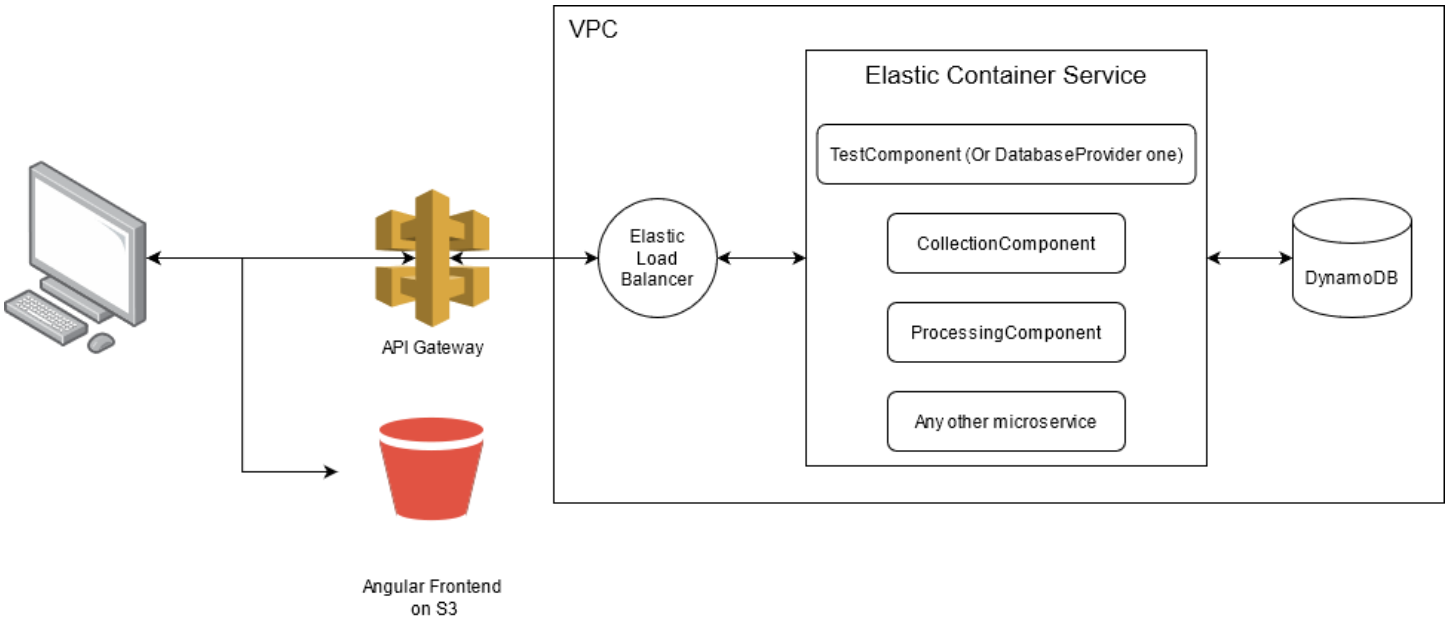
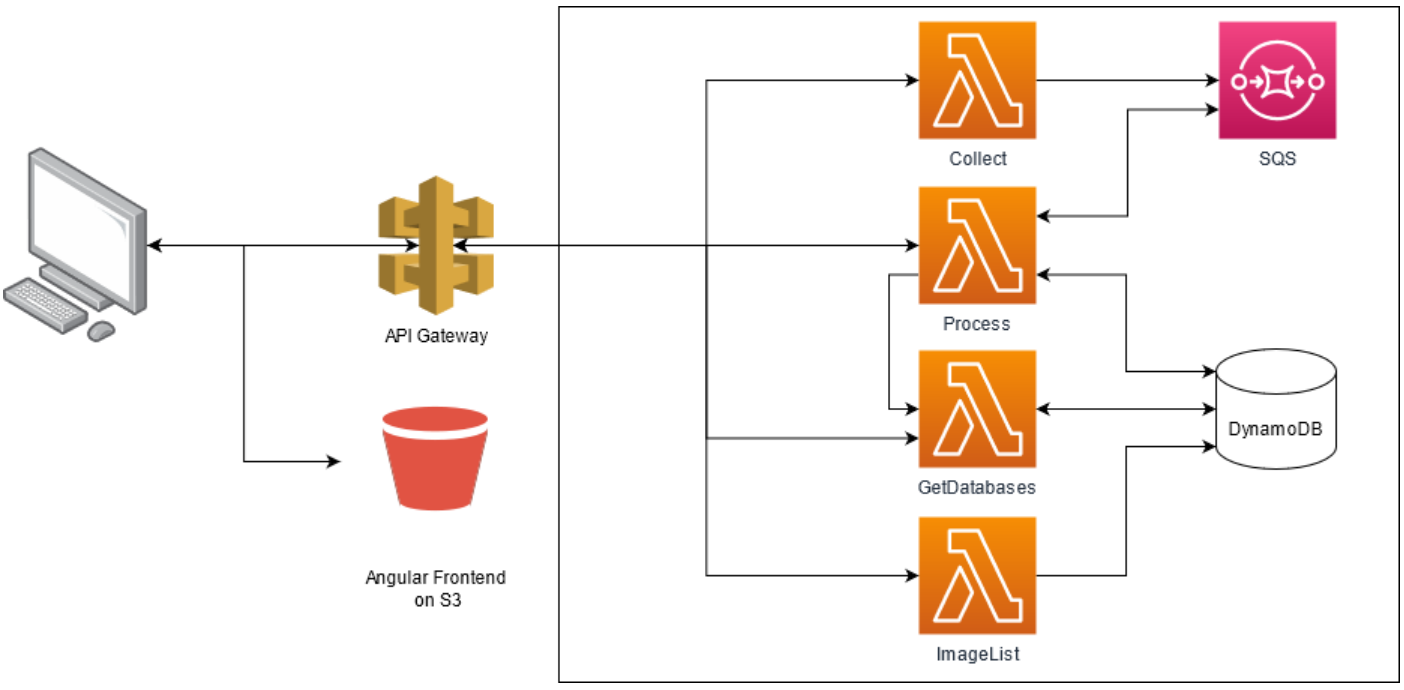
Solution overview – Héctor Humanes

The picture below suggests how to deploy the proposed exercise solution in AWS:



The S3 bucket contains both the html and javascript files which will be accessed from the clients. On the other part, the API Gateway provides the unified entry point for all the endpoints created in the different microservices. Behind the API Gateway, an ELB is placed since the exercise instructions specified that the service must handle large volume of request, and that justifies the existence of the ELB (properly configured) to admit large volume of requests. After the ELB, there is a ECS Cluster containing all the microservices which conforms the application itself. ELB has been chosen over BeanStalk since the application is fully dockerized and the workflow will be easier when working with ELB. Finally, a DynamoDB database is placed as the storage option for the application.

By the way, in my opinion, there is another option to develop the proposed application. By using it, we can benefit from the serverless architecture strong points since the proposed exercise application is really function-oriented. In addition to this switching from the synchronous HTTP communication to an asynchronous Event-Driven approach (with the help of a communication bus such as SQS) will also bring great benefits. The serverless approach would be:



In this architecture example, the Lambda functions act as endpoints for the collect, process and imageList as in the previous architecture proposal. The main difference is that the all the previous containers are just now a Lambda function, with a small footprint in resources comparing to the Spring Boot application in a Docker container. In addition to this, the collect component now gets all the images from a website and publish them to a queue asynchronously, so it's not aware of the availability status of the Processing component. The Processing component is also decoupled from the Collection component and only try to process the images as they pop into the queue. Using the queue system is also very interesting given the database unavailability constraint, since when this problem occurs, images can just be returned to the queue and the component can try to process them later.

If you have any question or want to go deeper into any aspect of these architectural proposal, I'll be more than happy to solve any doubts you may have.