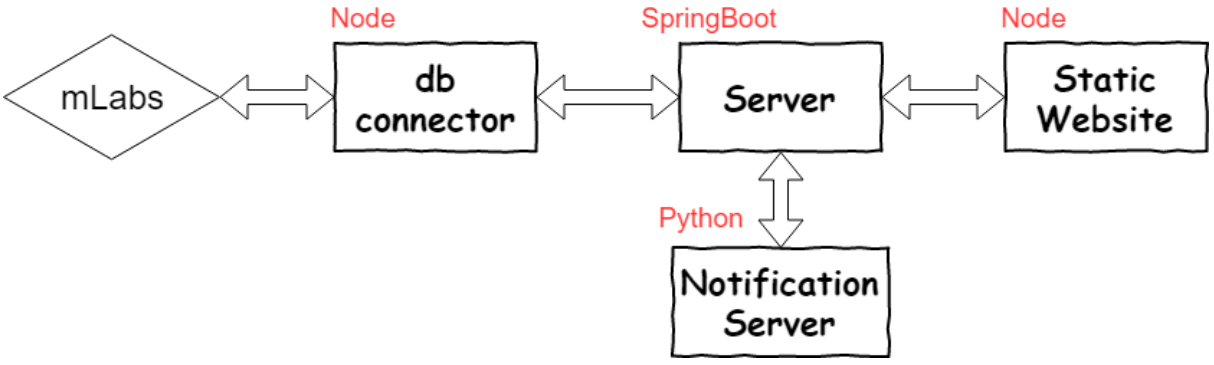
Little Anchorage Financial Bank (LAFB) - Cloud implementation

Original architecture



The previous software architecture in use by LAFB had a number of issues and potential improvements. Firstly, the system architecture as a whole was very monolithic – what does this mean? Each module in the architecture is highly dependent and tightly coupled with other components; if any of them malfunction or are removed the entire system will break. Thus it is very difficult to introduce new versions of any of the applications, introduction of new bugs can have fatal effects, and any mainaintance will mean extended periods of downtime for the entire system. None of these are desirable.

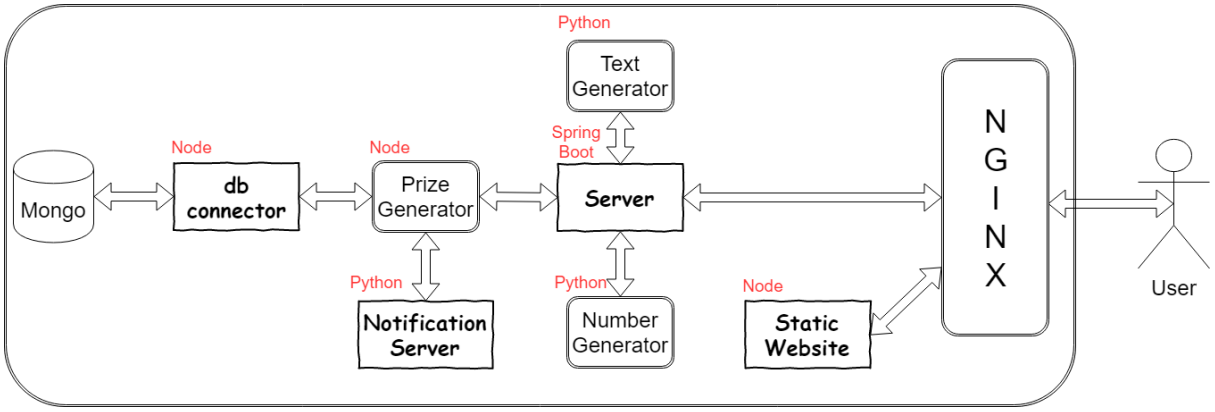
Scalability is another issue with the previous architecture; sudden high throughput into the system can lead to a sluggish system and longer wait times for users – both of which can mean lost customers if wait times are intolerable. Under the old architecture scaling up (improving the power of the existing server) or out (acquiring more servers) would have been a solution to this, albeit an expensive and inflexible one. There is however no elasticity in this solution – if traffic were to fall the company would still have been stuck with these extra servers or underutilised hardware, along with all the associated running and maintenance costs.

The on premises nature of the previous architecture's servers ensures its own problems and expenses. Security of servers is a massive issue; controlling who has access to the physical server space is always a concern – malicious actions are a very real possibility. Vulnerabilities are not limited to the human aspect; the threat of fire, flooding or other physical dangers would lead to the loss of the entire system, potentially irreparably. As evinced earlier, the costs of

running on an on premises server can be huge – with certain employees solely dedicated to it's functionality.

So in what ways have we improved the old architecture and what technologies did we use the achieve them? Cloud computing and containerisation.

New Architecture



Why move to the cloud?

Cloud computing is defined as 'the practice of using a network of remote servers hosted on the internet to store, manage and process data, rather than a local server or a personal computer'. Essentially this means hiring out computing resources at an external location for a price to host our system architecture. Why is this a good alternative to the current on premises server? Cloud computing provides a solution to all of the previously explained problems.

The scalability and elasticity issues are elegantly solved by creating more or less virtual machines and/or instances of applications to meet the demands currently asked of the system – and this scaling is abstracted from us, the system dynamically changes in response to higher or lower traffic. Thus the end user, your customers, should never experience increase wait times.

Security issues are also dampened by switching to a cloud computing solution. No on premises physical server entirely removes the associated risks – noone can unexpectedly access it, it cannot be damaged and nobody has to maintain it. The system will exist somewhere within a data centre located at a desired region (for example, at one of two regions in the UK).

Costs can be massively decreased by switching to a cloud solution. The obvious source of cost reduction lies in no longer having to maintain any physical server hardware and the employees tasked with doing so. Secondary cost reduction comes via only paying for the computing resources currently in use; scaling with increased or decreased customer demand. There are no unnecessary overheads.

System downtime is mitigated via cloud computing – cloud providers uphold Service Level Agreements (SLA), which specify the level of uptime users of the service can expect (upwards of 99.9999%). The responsibility for any downtime due to hardware failure is thus removed from LAFB. Another source of system downtime – failure or changes to individual applications within the system architecture is mitigated through the use of a technology known as containerisation. To achieve this we made use of Docker and Kubernetes.

Why use Docker?

Docker is a piece of software that makes the experience of running, maintaining and deploying applications much easier. This can be installed inside any VM that you create in azure.

When making the application it is common practice to have multiple services that handle one aspect of the deployment, this way, should an error occur, the error can be quickly found and the user experience should not be affected.

Docker utilizes these individual services by making something called a container from them. A container is running instance of a particular service inside the application and each services container can communicate to one another for a smooth experience

As well as containers, docker also utilizes a feature called images. An image is a basic blueprint for how a container should look and every time something is altered with the service this change is reflected in the base image and therefore in the container also.

These images can be stored inside a local repository similar to git to ensure speedy build times when deploying an application. Also like git, docker has its global repository called docker hub where local images can be stored on the internet and accessed by anyone should you wish the image to be public.

Why use Kubernetes?

Before considering why kubernetes has been chosen we need to consider what a container orchestration system is and also what means to achieve this are available. Currently, the two mainstream methods available to us on the market is kubernetes and docker swarm.

When people compare these two methods they don't seem to understand that the two methods are not that different from one another. Both are used to manage deployed containers spread across multiple VM's that when combined act as one big machine, however, they do this in very subtly different ways.

Docker is a platform and tool for building, distributing and running Docker containers. It offers its native clustering tool that can be used to orchestrate and schedule containers on machine clusters.

Kubernetes is a container orchestration system for Docker containers that is more extensive than Docker Swarm and is meant to coordinate clusters of nodes at scale in production in an efficient manner. It works around the concept of pods, which are scheduling units (and can contain one or more containers) in the Kubernetes ecosystem, and they are distributed among nodes to provide high availability.

While kubernetes does run from the groundwork of docker it is not a complete solution and does require additional plugins to run. With that being said kubernetes is still considered the market leader for container orchestration.