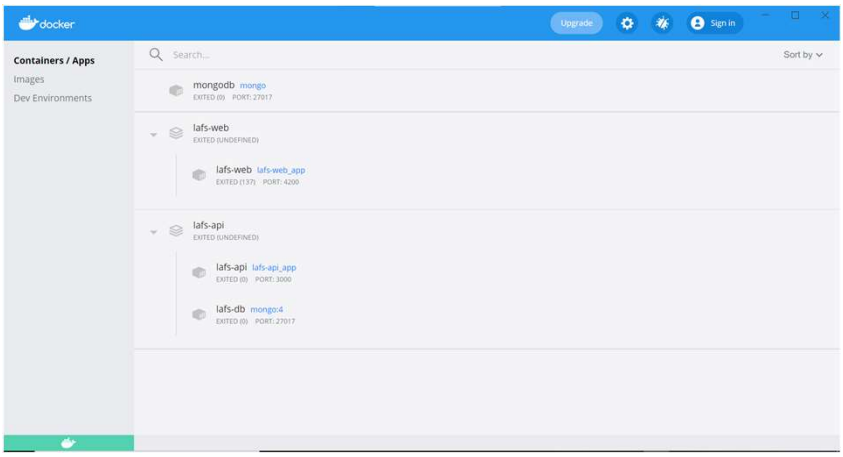
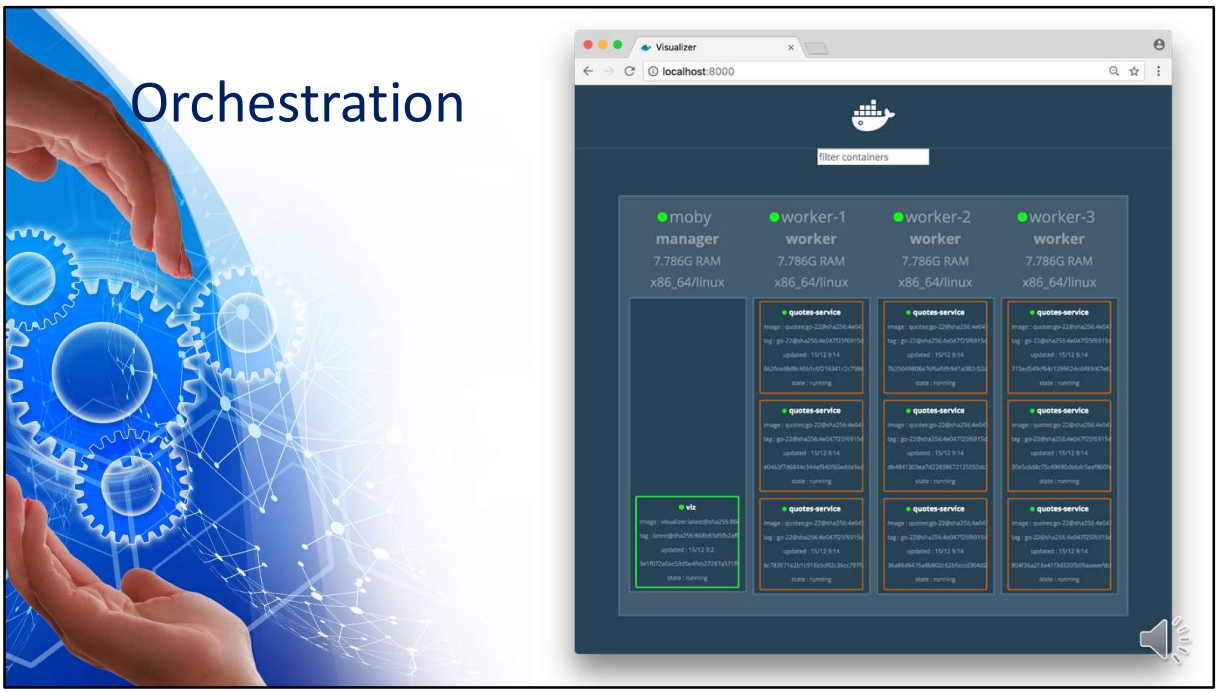
Video Link: https://youtu.be/Voltnh7W_t4

Hello, and welcome to my presentation on cloud development. Computers have always been a great source of fascination to me. I was a biomedical equipment technician in the Army before working as an electrical engineer for the past two and a half years where I spend my time designing enclosures in CAD, circuits, and writing embedded code and the applications that interface between the embedded systems and the UI. So, most of my experience is in embedded systems, but I've found there is some overlap in cloud development that gives me a unique perspective on the field. I hope I can leverage that experience to articulate some of the intricacies of cloud development without getting too technical.

Containerization

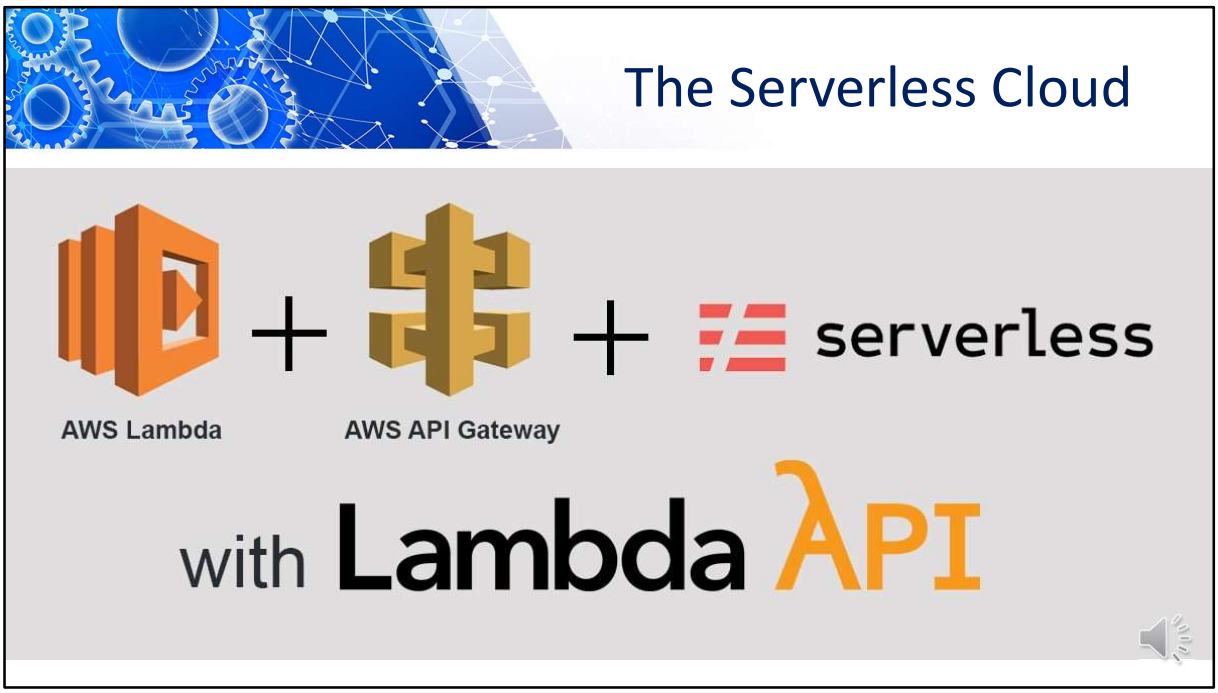


Containerization is one of the most pervasive models for cloud migration. It allows administrators to allocate just enough resources for the application without the overhead of allocating for an OS for every instance. Many containers can exist and operate on a single system. In this model, scaling becomes an issue because you would need to continuously expand the size of your single server system to allow more containers to operate. A better solution would use many different servers, potentially in completely different physical locations to host your application. This would allow you to scale the system by increasing the number of servers rather than expanding the capability of a single server. Enter: container orchestration. Container orchestration provides a hidden layer that seamlessly connects different containers in a single application with containers across physically distinct systems. The tool I've found most useful in implementing container orchestration is Docker Compose.

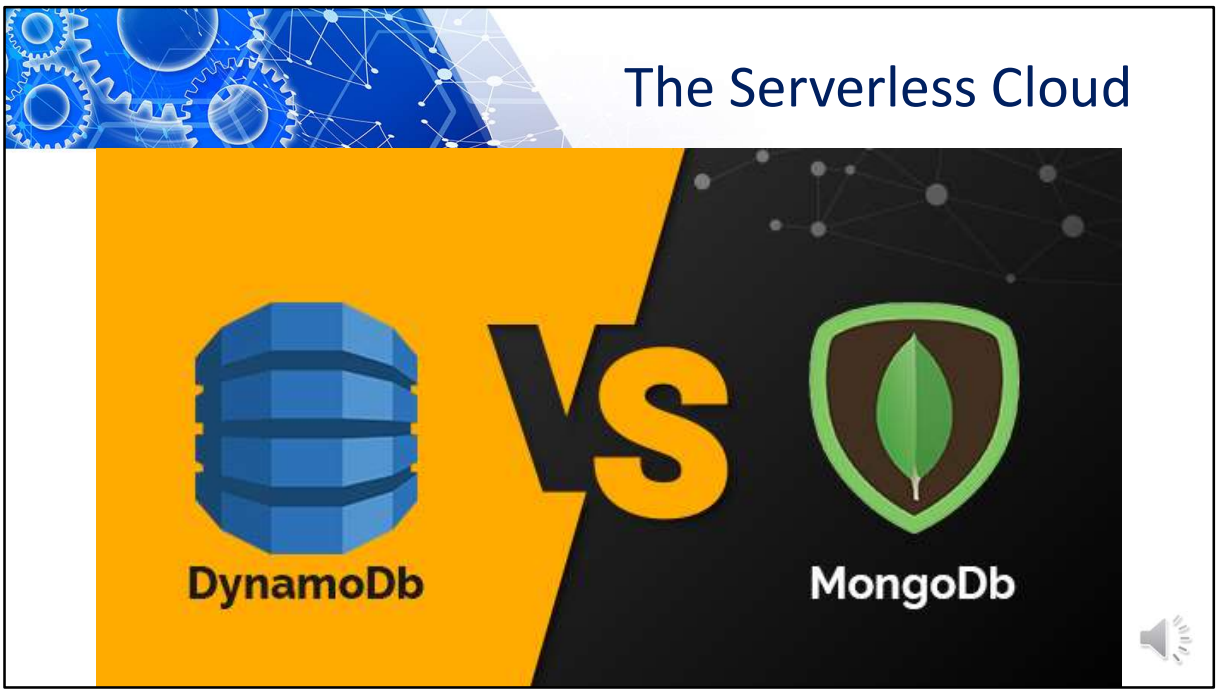


Docker Compose requires very little configuration to get your application up and running with container orchestration. It also allows you to visually represent your orchestration structure which can highlight shortcomings in your application design. If there is a path from one container to another that is slowing down your system, you can more easily see that using tools like “docker-compose-viz.”

Serverless computing allows you to scale your application on demand rather than having to lease or host an entire server which has a fixed number of resources. Using container orchestration, your application can expand to the needs of user's by dynamically expanding or decreasing the number of instances that can be spread across many different servers. This is the heart of the benefit of using container orchestration. It facilitates the use of serverless computing which allows a more efficient consumption and allocation of resources by minimizing overhead.



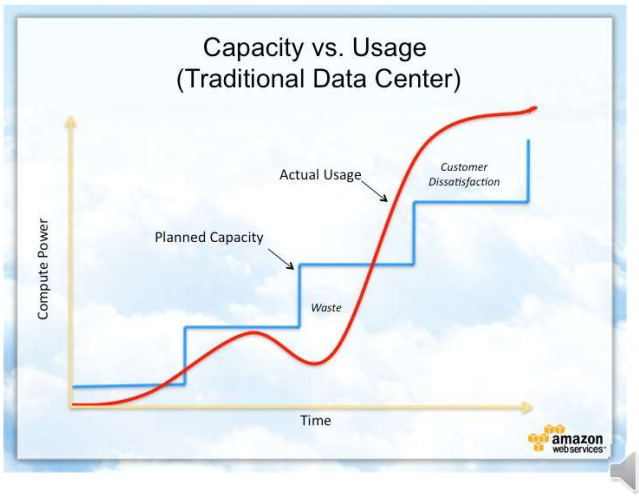
A serverless API allows you to call complex functions and only pay for the resources consumed solely by the function call. AWS offers the Lambda service which allows you to implement a serverless API which only consumes resources when it is called. Functions can be as complex as your application demands and can support dependencies as needed. To produce a Lambda function, I would start by writing the code in the IDE of your choosing and then upload it to the lambda service. You can add test scripts to verify the operation of your code and check that it is able to communicate with other resources in your application. If the function accesses other components of your application, you'll need to make sure you add permissions for each call.



There are some different options available for database hosting software and structures. For this presentation I will be comparing MongoDB and DynamoDB. A key difference is that Mongo can be applied to any server system while Dynamo only runs on AWS servers. Dynamo can be integrated with other AWS modules very easily and requires much less effort than Mongo. If above all things you crave freedom to choose your provider, Mongo may be a better option in this regard.
Mongo uses a document-based storage system that can result in very different query speeds depending on the complexity of the query. Dynamo uses a key-value storage table which can improve and homogenize the speed of queries. Which storage system is best for your application depends on the applications requirements. Mongo is schema-free but allows you to configure a schema if your application demands it. Dynamo doesn't allow you to set a schema.

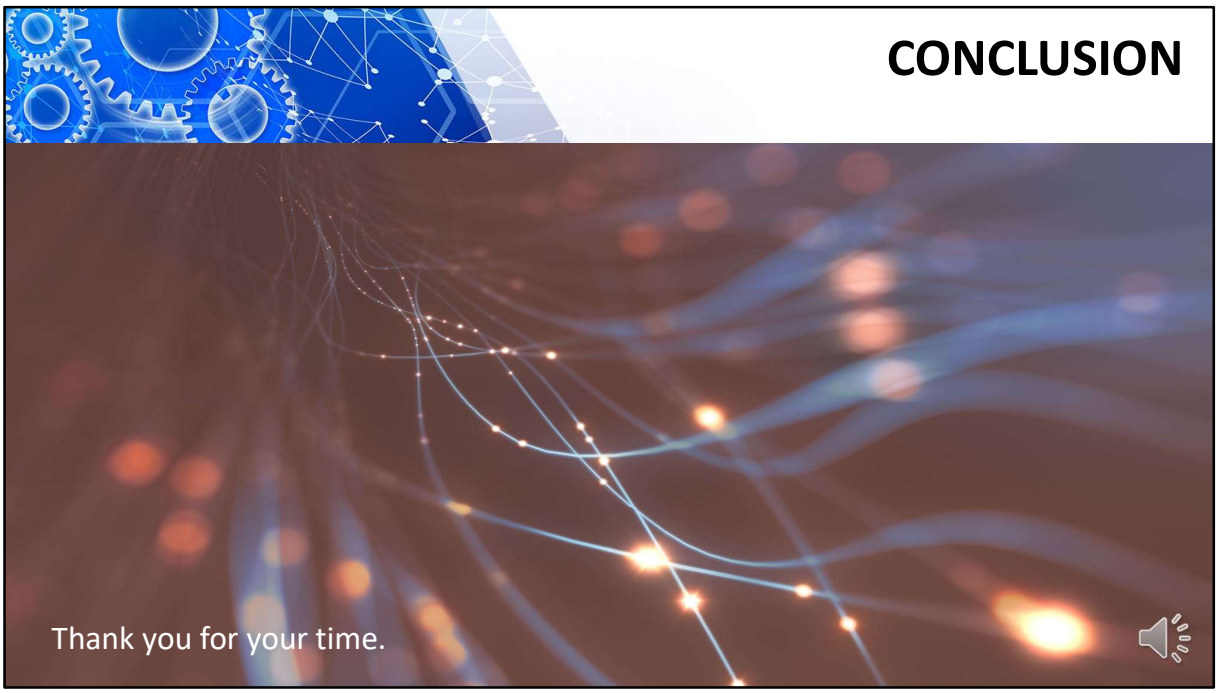
Cloud-Based Development Principles

- Elasticity
- Pay-for-use model



Elasticity is a term which refers to the dynamic response of a serverless cloud to allocate resources only as needed. A typical server model will have a fixed platform that determines the resources available such that the responsiveness of the application is maximized if the load is below the planned capacity. The graph above illustrates this concept. The servers have fixed operating costs so any demand that is below what the system is capable of at maximum capacity are wasted resources and money. If the usage curve exceeds the capacity of the server, the application will become unresponsive to users. Elasticity in serverless computing solves this problem by dynamically allocating resources across a large server pool which results in a usage and capacity curve that closely track.

To prevent unauthorized access to parts of your website, you should setup a hierarchy of privileges that each type of user will need to complete their tasks. You can use authentication tokens to achieve a tiered privilege hierarchy that creates the desired separation of privileges. Roles define the different use cases and policies determine what a user with a certain role can access or modify. For the full-stack application designed for this presentation, I created a policy that allowed my lambda functions to modify or read from my Dynamo database. This highlights an important point about how authentication roles aren't just important for users but can also provide separation between functions within your application which ensures a secure API. Communications between different resources should have just enough privileges to accomplish what they need.



Serverless computing offers a dynamic and cost-effective solution to migrate an application to the cloud. By using containers and container orchestration, your application will be able to produce a responsive user experience while keeping overhead to a minimum. Going with a cloud solution offloads all the technical requirements and duties to the cloud service provider which allows you to focus on the quality and experience for your application with greater focus. I hope this presentation has been of some small benefit to you. Thank you for your time.