Stuff to stay in an interview,

Kuberetes is an orchestration tool for containers – convert isolated containers running on different hardware in to a cluster.

Docker abstracts the app from the image

Kubernetes abstracts the container from the infrastructure.

And istio abstracts the networking polcies for access polices.

Serverless and containers

far less overhead and more flexibility than applications hosted on traditional servers or virtual machines.

The gold standard is serveless

As serverless applications are more scalable and usually more cost-effective

Lambda you only pay for etc,

But,

What happends if the application is stateful.

With a container you dump everything into container at first then break it up picemeal.

'contains' both an application and all the elements the application needs to run properly, including system libraries, system settings, and other dependencies. Like a 'just add water' pancake mix, containers only need one thing – to be hosted and run – in order to perform their function.

If you

#### What are the key differences between serverless computing and containers?

##### Physical machines

'Serverless' computing actually runs on servers, but it is up to the serverless vendor to provision server space as it is needed by the application; no specific machines are assigned for a given function or application. On the other hand, each container lives on one machine at a time and uses the operating system of that machine, though they can be moved easily to a different machine if desired.

##### Scalability

In a container-based architecture, the number of containers deployed is determined by the developer in advance. In contrast, in a serverless architecture, the backend inherently and automatically scales to meet demand.

To continue the shipping container metaphor, a shipping company could try to forecast an increase in demand for a certain product and ship more containers to the destination to meet that demand, but it could not snap its fingers and produce more containers full of goods if demand were to exceed expectations.

Serverless architecture is a way to do exactly that. When it comes to computing power, serverless computing is like a water supply in a modern home: by turning on the tap, consumers can acquire and use as much water as they need at any time, and they only pay for what they use. This is far more scalable than attempting to buy water one bucket, or one shipping container, at a time.

##### Cost

Containers are constantly running, and therefore cloud providers have to charge for the server space even if no one is using the application at the time.

There are no continued expenses in a serverless architecture because application code does not run unless it is called. Instead, developers are only charged for the server capacity that their application does in fact use.

##### Maintenance

Containers are hosted in the cloud, but cloud providers do not update or maintain them. Developers have to manage and update each container they deploy.

From a developer's perspective, a serverless architecture has no backend to manage. The vendor takes care of all management and software updates for the servers that run the code.

##### Time of deployment

Containers take longer to set up initially than serverless functions because it is necessary to configure system settings, libraries, and so on. Once configured, containers take only a few seconds to deploy. But because serverless functions are smaller than container microservices and do not come bundled with system dependencies, they only take milliseconds to deploy. Serverless applications can be live as soon as the code is uploaded.

IF YOU USE Kubernetes,

Fault –tolerance

Auto healing

Isolation

Scaling autoscaling

Load balancing