Deploy microservices with containers.

*While running microservices directly as processes is very efficient, it comes at a cost.*

* *The server must be meticulously maintained with the necessary dependencies and tools.*
* *A runaway process can consume all the memory or CPU.*
* *Deploying and monitoring the microservices is a brittle process.*

*All these shortcomings can be mitigated with containers. Containers are packages that contain everything a program needs to run. A container image is a self-contained unit that can run on any server without having to install any dependencies or tools first*

*Containers provides enough virtualization and brings below benefits:*

* *Isolation – contained processes are isolated from each other. All containers have their own filesystem hence dependency conflicts are not possible.*
* *Concurrency – We can run multiple instances per container image without conflict.*
* *Lightweight – We ne need to boot entire OS. Hence container is lightweight.*
* *No-installation – Containers is just a matter of download and run an image. There is no installation step require.*
* *Resource manages – We can define CPU and memory size as per requirement. Hence no destabilize issue.*

***Serverless Deployment -***

*Container as a Server provides a benefit to deploy an application serverless, where we no need to worry about managing server. We only need to create a container image and pass it onto the cloud provider, which will take further care: provide virtual machine, run docker script, start and monitor image.*

*benefits a managed container service has:*

* *No servers: there is no need to maintain or patch servers.*
* *Easy deployment: just build a container image and tell the service to use it.*
* *Autoscaling: the cloud provider can provide more capacity when demand spikes or stop all containers when there is no traffic.*

*Drawbacks –*

* *Vendor lock-in - this is the big one. Moving away from a managed service is always challenging, as the cloud vendor provides and controls most of the infrastructure.*
* *Limited resources: managed services impose CPU and memory limits that cannot be avoided.*
* *Less control: we don’t have the same level of control we get with other options. You’re out of luck if you need functionality that is not provided by the managed service.*