Stateless vs. Stateful Kubernetes

At a very basic level, as the name suggests, the term “stateless” means that no past data nor state is stored or needs to be persistent when a new container is created.

Stateless applications tend to include containerized microservices apps, CDN, print services, or any short-term workers. and are easy for both deploying and managing resources.

Stateful applications typically involve some database, such as **Cassandra, MongoDB, or MySQL and processes a read and/or write to it.**

Typically, it will involve requests being processed based on the information provided.

The prior request history impacts the current state; hence, the server must access and hold onto state information generated during the processing of the earlier request, which is why it is called stateful.

Containerized Stateful Application Use Cases and Their Challenges

Containerized applications need statefulness, as they are commonly deployed in hybrid and edge-to-core-to-cloud workloads, as well as CI/CD use cases. Here are some of the common use cases for containerized application deployments

Data analytics processing and need to go over massive amounts of data repeatedly.

– Single-instance databases like MySQL, PostgreSQL, MariaDB

– NoSQL databases like Cassandra and MongoDB

– In-memory databases like Redis and MemSQL and KDB+

– Messaging apps like Kafka

– Business critical apps like Oracle, SQL server, and SAP

Kubernetes Storage and Stateful Applications

In Kubernetes, basic storage building blocks are known **as volumes**.

A volume is attached to a pod. This means if the pod is eliminated by the cluster, the volume gets to be stored in the node. What if the node disappears, what do we do?

A volume is like **local storage** (you can attach it to the node), and there is no **persistence** to it. A volume gets released when **a pod is destroyed**. As such, a regular volume **lacks persistence, portability, and scalability.**

Persistent storage, as the name suggests, **retains, or stores the data generated by an application, making it suitable for stateful applications.** Unlike local storage or a regular volume, a persistent volume is **managed by clusters**, and it's not dependent on the pod lifecycle; therefore, the data can be retained and reused.

When creating a persistent volume for Kubernetes clusters, the storage file system, and its configuration (IDs, access modes, size, names etc.) needs to be specified in a Storage Class.

There are 3 steps involved in creating a persistent volume and attaching it to a container in a pod:

- Create a StorageClass
- Create a PersistentVolumeClaim
- Define the volume