**3. StatefulSet:**

A **StatefulSet** is like a **ReplicaSet** but with added functionality to manage stateful applications.

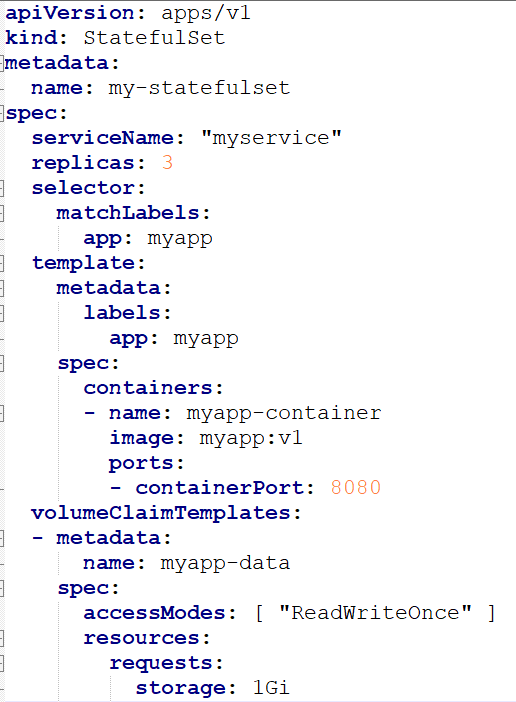
StatefulSets ensure that the given pods are unique, persistent identities across rescheduling and scaling events. This makes StatefulSets suitable for applications like databases or distributed systems that require consistent and predictable network identities, stable storage, and ordered deployment or scaling.

**Key Features:**

* **Stable, unique network identities** for each pod (e.g., podname-0, podname-1, etc.).
* **Stable, persistent storage** (by using Persistent Volume Claims).
* **Ordered deployment and scaling** (pods are created/terminated in a specific order).
* Supports **rolling updates** (you can update StatefulSet pods one at a time).

**Use Cases:**

* **Stateful applications**: Where the application requires **stable network identities**, **persistent storage**, and **ordered deployment**. For example:
  + **Databases** like **MySQL**, **PostgreSQL**, **Cassandra**, **MongoDB**.
  + **Distributed systems** like **Zookeeper** or **Kafka**.
  + **Message queues**: Redis



Here we don’t specify PVC separately, instead we use **VolumeClaimTemplates** to claim the volumes. **volumeClaimTemplates** ensures that each pod gets its own PersistentVolumeClaim.

Each pod in the StatefulSet will have a unique identity(myapp-0, myapp-1 etc..) and a stable storage volume (e.g., myapp-data-0(1GB), myapp-data-1(1GB), etc.) and volumes mappings to Pods also done in orderly like (myapp-0 🡪ayapp-data-0, etc..)