Topic	StatefulSet	Deployment
Updates & Scaling	▸ **Rolling Updates** ▸ **Scaling** ▸ **Roll Back** – Custom process	▸ **Rolling Updates** ▸ **Scaling** ▸ **Rolling Back**
Persisted Volumes	▸ **Volume per pod –** Using volumeClaimTemplates ▸ **Volume shared between pods** – Using Persistent Volume Claims	**Volume shared between pods** – using Persistent Volume Claims
Number of pods running	K8s assures that no more than the maximum number of pods are running, unless it is overridden.	The number of pods can be higher or lower than the number of replicas in specified (normally higher)
Single resilient container	**Partial** – On node failure the container is not automatically restarted.	Container restarted by K8s after configurable delay.

Many controllers allow the specification of affinity and anti-affinity rules, and constraint of object behavior with other objects with particular labels. For example, you can prevent a pod from being deployed on a particular node. The affinity/anti-affinity feature specifies a wide range of constraints that can be expressed:

Rules can be “soft”/“preference” rather than a hard requirement. So, if the scheduler cannot satisfy it, the pod will still be scheduled constrain against labels on other pods running on the node are allowed, rather than against labels on the node itself, which allows rules about which pods can and cannot be colocated

The list in this section is not intended to be exhaustive. Instead, it provides a reference for the K8s concepts that are used in the practical chapters of this Redbooks publication. A more detailed description of these concepts is provided in the K8s public documentation:
https://kubernetes.io/docs/concepts/#kubernetes-objects

4.5 Cloud-native is not for everyone, nor for everything

All of the preceding benefits sound valuable. Who wouldn't want them, right? Who wouldn't want to be able to innovate at the speed of the business, optimize the cost of infrastructure, ensure that their applications had the highest possible availability, and not be locked into any particular vendor or technology? It's the CIO’s, and indeed the CEO’s dream.

However, any good CIO, and more so the CFO, knows that all *improvements* come at a cost. Moving to a cloud-native approach costs in time and money. For sure, it has the potential to provide enormous benefits in the long term enabling the business to remain competitive, and indeed innovate at a pace that brings them to the front of the pack. But it requires investment up front, with returns being seen only somewhere in the middle. The more dramatic gains are seen after the system is well established.

Therefore, it should be clear that there are very few companies that adopt cloud-native wall to wall. This is all the more true for large long-standing enterprises with a large IT landscape. Therefore, we need to clearly recognize *where* we need the benefits of cloud-native, *which* benefits we need, and then plan accordingly, applying the improvements that make the right difference for our particular context.

This likely means adopting these practices in pockets of the enterprise only, especially at first. Applying it to specific applications, initiatives, technology areas and indeed teams.   It also means considering what are our priorities, and indeed more granularly, what are the specific