deployment disadv:

1. Pods share same volumes (not good with DB as it will make a corruption)
2. Pods randomly have hash charachters in their names so there is no persistent identity (in the DB cluster replication pods will not identify each other )
3. Deployment scaling does not help stateful workload.

Statefulset

1. Pods are individually addressable

<podname>-<ordinal>

Pod-0

Pod-1

If pod-0 destroyed and recreated it will be created with the same dns name

1. Pods have their own persistent volume.
2. In the scale down k8s will terminate the highst pod number

* Persistent volume

<https://aws.amazon.com/blogs/storage/persistent-storage-for-kubernetes/>

* headless service

A headless service in the context of Kubernetes refers to a service that doesn't have a cluster-specific IP address associated with it. Unlike a regular service in Kubernetes, which typically provides a stable endpoint (IP address) for clients to connect to, a headless service doesn't abstract the endpoints.

When you create a headless service in Kubernetes, the service is assigned a DNS entry, but instead of resolving to a single IP address, it resolves to the individual IPs of the pods backing that service. This means that each pod behind the headless service will have its own DNS record, allowing direct access to each pod without going through a load balancer or proxy.

Headless services are often used for scenarios where direct access to individual pods is necessary. For example:

1. StatefulSets: When deploying stateful applications where each pod has its own identity or state (like databases), a headless service can be useful. Each pod can be addressed individually by other services or applications.
2. Service Discovery: In scenarios where you want to discover and communicate with all instances of a certain type of pod, a headless service allows direct DNS lookups to each pod's IP.
3. Custom Load Balancing: Some applications might require custom load balancing or routing logic, and a headless service can be used to implement such custom behavior.

By leveraging headless services, Kubernetes allows for more granular control and direct communication with individual pods in a service without relying on a single, load-balanced endpoint.

Operators:

* Postgres operator

<https://www.kubegres.io/doc/getting-started.html>

Configure Liveness, Readiness and Startup Probes:

<https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/#before-you-begin>

* Deployment types

https://medium.com/tech-at-wildlife-studios/canary-deployment-in-kubernetes-how-to-use-the-pattern-b2e9c40d085d

<https://github.com/ianlewis/kubernetes-bluegreen-deployment-tutorial>

* Horizontal autoscaling

<https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/>

* Kubectl patch
* **kubectl patch** is a command-line utility in Kubernetes that allows you to update the fields of a resource without replacing the entire resource definition. This is particularly useful when you want to make small modifications to a resource without having to resubmit the entire YAML or JSON definition.
* The basic syntax for **kubectl patch** is as follows:
* bashCopy code
* kubectl patch RESOURCE\_TYPE RESOURCE\_NAME -p 'PATCH\_SPEC'
* Here, **RESOURCE\_TYPE** is the type of Kubernetes resource you want to patch (e.g., Deployment, Pod, Service), **RESOURCE\_NAME** is the name of the specific resource you want to patch, and **PATCH\_SPEC** is a JSON or YAML string that specifies the changes you want to make.
* For example, to update the image of a container in a Deployment, you can use the following command:
* bashCopy code
* kubectl patch deployment my-deployment -p '{"spec": {"template": {"spec": {"containers": [{"name": "my-container", "image": "new-image:tag"}]}}}}'
* This will update the image of the container named "my-container" in the "my-deployment" Deployment without having to resubmit the entire Deployment manifest.
* The **-p** flag is used to specify the patch data, and it can accept the patch data either as a JSON or YAML string directly or from a file using the **--patch** flag.
* Keep in mind that **kubectl patch** is a powerful tool, but it requires careful usage to avoid unintended consequences. Always review and test the changes before applying them in a production environment.