Kubernetes StatefulSet - Examples & Best Practices

Bharathiraja Shanmugam

November 23, 2021 13 Minute Read

Table of Contents

Containers were initially designed to be stateless, so building stateful applications in th tricky. Luckily, Kubernetes Statefulset solves this problem.

It manages a set of replica pods, all with a unique identity. This makes it the perfect too managing applications that need persistent storage or a unique network identity.

This article covers what Kubernetes Statefulsets are, when to use them, and how to dep applications with them. It also covers Statefulset concepts using the MySQL database. T includes creating a fixed Pod name for each MySQL replication and accessing the replica using the Services object.

Key Takeaway

- Kubernetes Statefulset is a tool for managing, deploying, and scaling stateful applica
- Unlike deployments, StatefulSet guarantees the order of pod creation and terminati makes it better for building stateful applications.

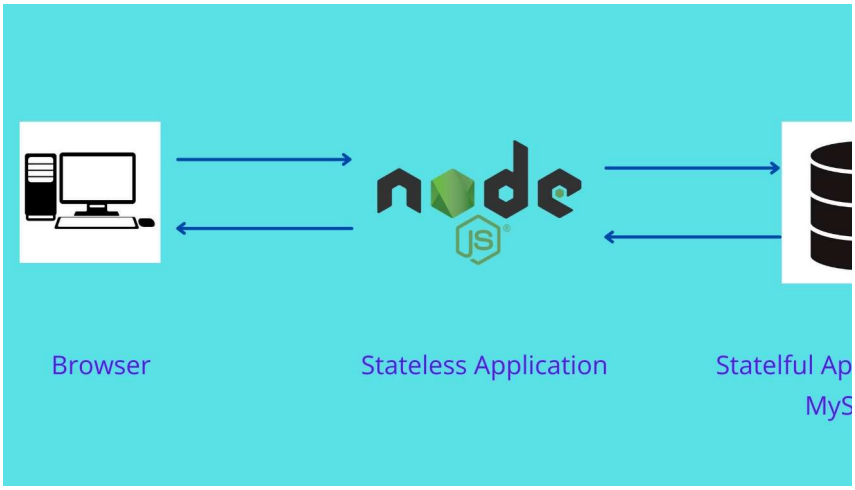
- However, they fall short in certain areas, such as volume deletion and storage manag

What Are Stateful Applications?

Stateful applications receive data and store and track them. MySQL, Oracle, and Postgre some popular stateful applications.

Stateless applications are the opposite. They do not store data; they only process it whe receive it. Node.js and Nginx are examples of stateless applications.

A modern web application uses both types of applications to serve user requests. For ex Node.js, a stateless application, receives new data on each user request. It then connect stateful application like MySQL to process the data. MySQL then stores and updates the on the user's request.



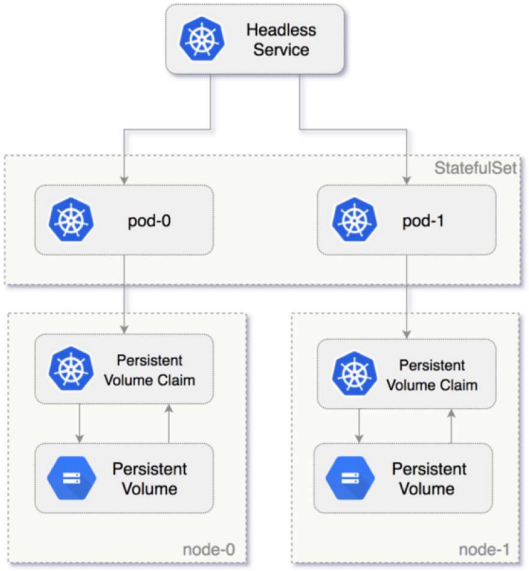
What Are StatefulSets?

Kubernetes StatefulSets are controllers that run stateful applications as containers (Po assign a sticky identity—an ordinal number starting from zero—to each Pod. They do this assigning random IDs to each replica Pod.

The system creates a new Pod by cloning the previous Pod's data. If the previous Pod re the pending state, then the system will not create the new Pod.

If you delete a Pod, the system will delete it in reverse order, not random order. For exar had four replicas and scaled down to three, it would delete the Pod numbered 3.

The diagram below shows how the Pod is numbered from zero and how Kubernetes pers volume is attached to the Pod in the StatefulSets.



When to Use StatefulSets

There are several reasons to consider using StatefulSets. Here are two examples:

- Say you deployed a MySQL database in the Kubernetes cluster and scaled it to three and a frontend application wants to access the MySQL cluster to read and write data request will be forwarded to three Pods. However, the write request will only be forw the first (primary) Pod, and the data will be synced with the other Pods. You can achi using StatefulSets.
- Deleting or scaling down a StatefulSet will not delete the volumes in the stateful app This gives your data safety. If you delete the MySQL Pod or if the MySQL Pod restarts, have access to the data in the same volume.

Deployment vs. StatefulSets

You can also create Pods (containers) using the Deployment object in the Kubernetes cl allows you to easily replicate Pods and attach a storage volume to the Pods. The same t done by using StatefulSets. What then is the advantage of using StatefulSets?

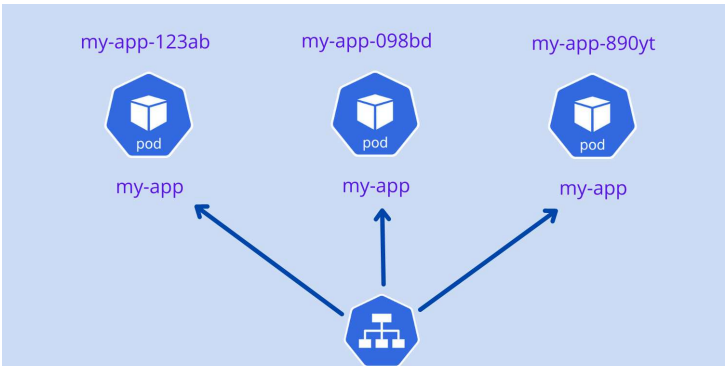
Well, the Pods created using the Deployment object are assigned random IDs. For exam creating a Pod named “my-app”, and you are scaling it to three replicas. The names of t created like this:

```
my-app-123ab my-app-098bd my-app-890yt
```

After the name “my-app”, random IDs are added. If the Pod restarts or you scale it down again, the Kubernetes Deployment object will assign different random IDs for each Pod. restarting, the names of all Pods appear like this:

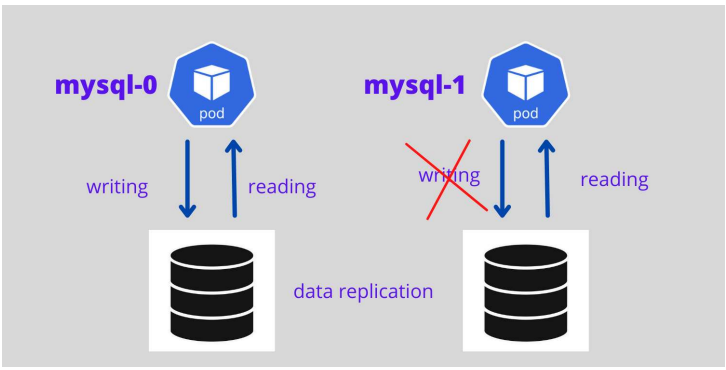
```
my-app-jk879 my-app-kl097 my-app-76hf7
```

All these Pods are associated with one load balancer service. So in a stateless applicatio in the Pod name are easily identified, and the service object easily handles the random and distributes the load. This type of deployment is very suitable for stateless applicati

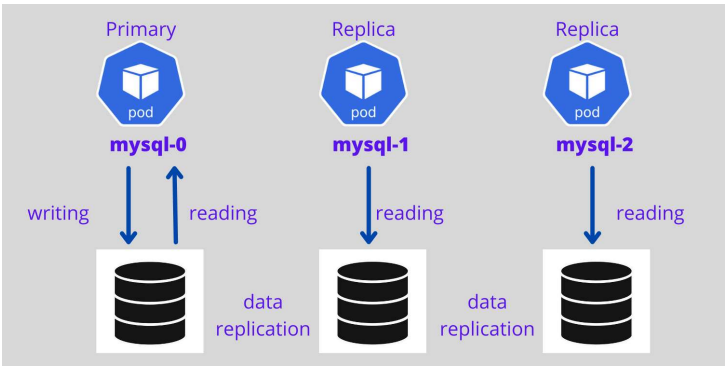


However, stateful applications cannot be deployed like this. The stateful application ne identity for each Pod because replica Pods are not identical Pods.

Take a look at the MySQL database deployment. Assume you are creating Pods for the M database using the Kubernetes Deployment object and scaling the Pods. If you are writi one MySQL Pod, do not replicate the same data on another MySQL Pod if the Pod is rest is the first problem with the Kubernetes Deployment object for the stateful application.

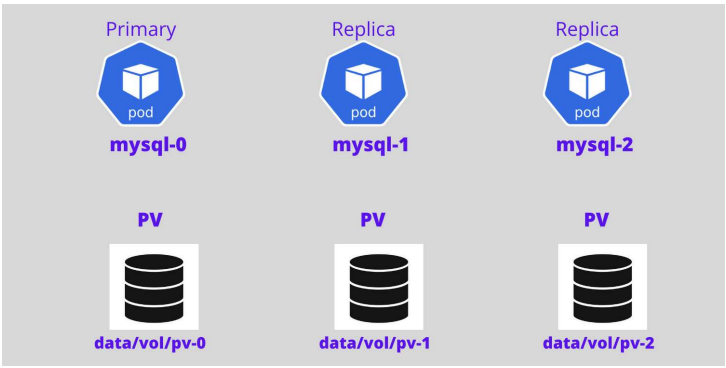


Stateful applications always need a sticky identity. While the Kubernetes Deployment ol random IDs for each Pod, the Kubernetes StatefulSets controller offers an ordinal numb Pod starting from zero, such as mysql-0, mysql-1, mysql-2, and so forth.

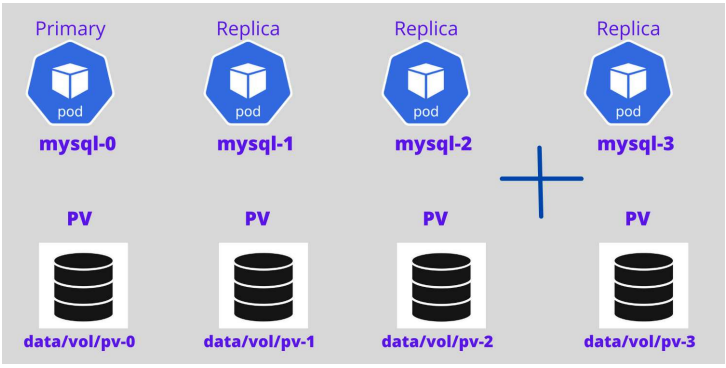


For stateful applications with a StatefulSet controller, it is possible to set the first Pod a and other Pods as replicas—the first Pod will handle both read and write requests from and other Pods always sync with the first Pod for data replication. If the Pod dies, a new created with the same name.

The diagram below shows a MySQL primary and replica architecture with persistent volu data replication architecture.



Now, add another Pod to that. The fourth Pod will only be created if the third Pod is up a running, and it will clone the data from the previous Pod.



In summary, StatefulSets provide the following advantages when compared to Deploym

- Ordered numbers for each Pod
- The first Pod can be a primary, which makes it a good choice when creating a replica database setup, which handles both reading and writing
- Other Pods act as replicas
- New Pods will only be created if the previous Pod is in running state and will clone t Pod's data
- Deletion of Pods occurs in reverse order

How to Create a StatefulSet in Kubernetes

Let's go through how to create a Pod for MySQL database using StatefulSet.

Create a Secret

First, create a Secret for the MySQL app. This will store sensitive information, such as us and passwords.

Here, I am creating a simple Secret. However, in a production environment, using the Ha Vault is recommended. Use the following code to create a Secret for MySQL:

```
apiVersion: v1 kind: Secret metadata: name: mysql-password type: opac
```

Save the code using the file name `mysql-secret.yaml`. Then, run this command on you Kubernetes cluster:

```
kubectl apply -f mysql-secret.yaml
```

Get the list of Secrets:

```
kubectl get secrets
```

Create a MySQL StatefulSet Application

Before creating a StatefulSet application, check your volumes by getting the persistent v

```
kubectl get pv NAME CAPACITY ACCESS MODES RECLAIM STATUS pvc-e0567 1
```

Next, get the persistent volume claim list:

```
kubectl get pvc NAME STATUS VOLUME CAPACITY ACCESS mysql-store-mysql-
```

Last, get the storage class list:

```
kubectl get storageclass NAME PROVISIONER RECLAIMPOLICY linode-block-
```

Then use the following code to create a MySQL StatefulSet application in the Kubernete

```
apiVersion: apps/v1 kind: StatefulSet metadata: name: mysql-set spec:
```

Here are a few things to note:

1. The `kind` is a StatefulSet. `kind` tells Kubernetes to create a MySQL application with t feature.
2. The password is taken from the Secret object using the `secretKeyRef`.
3. The Linode block storage was used in the `volumeClaimTemplates`. If you are not me any storage class name here, then it will take the default storage class in your cluste
4. The replication count here is 3 (using the replica parameter), so it will create three P `mysql-set-0`, `mysql-set-1`, and `mysql-set-2`.

Next, save the code using the file name `mysql.yaml` and execute using the following co

```
kubectl apply -f mysql.yaml
```

Now that the MySQL Pods are created, get the Pods list:

```
kubectl get pods NAME READY STATUS RESTARTS AGE mysql-set-0 1/1 Runni
```

Create a Service for the StatefulSet Applicat

Now, create the service for the MySQL Pod. Do not use the load balancer for a stateful a
create a headless service for the MySQL app using this code:

```
apiVersion: v1 kind: Service metadata: name: mysql labels: app: mysql
```

Save the code using the file name `mysql-service.yaml` and execute using the followin
command:

```
kubectl apply -f mysql-service.yaml
```

Get the list of running services:

```
kubectl get svc
```

Create a Client for MySQL

To access MySQL, you will need a MySQL client tool. Deploy a MySQL client using the foll
manifest code:

```
apiVersion: v1 kind: Pod metadata: name: mysql-client spec: container
```

Save the code using the file name `mysql-client.yaml` and execute using the following

```
kubectl apply -f mysql-client.yaml
```

Then enter this into the MySQL client:

```
kubectl exec --stdin --tty mysql-client -- sh
```

Finally, install the MySQL client tool:

```
apk add mysql-client
```

Access the MySQL Application Using the My
Client

Next, access the MySQL application using the MySQL client and create databases on the

If you are not already in the MySQL client Pod, enter it now:

```
kubectl exec -it mysql-client /bin/sh
```

To access MySQL, you can use the same standard MySQL command to connect with the M server:

```
mysql -u root -p -h host-server-name
```

For access, you will need a MySQL server name. The syntax of the MySQL server in the Ku cluster is given below:

```
stateful_name-ordinal_number.mysql.default.svc.cluster.local #Example
```

Connect with the MySQL primary Pod using the following command. When asked for a pa enter the one you made in the “Create a Secret” section above.

```
mysql -u root -p -h mysql-set-0.mysql.default.svc.cluster.local
```

Next, create a database on the MySQL primary, then exit:

```
create database erp; exit;
```

Now connect the other Pods and create the database like above:

```
mysql -u root -p -h mysql-set-1.mysql.default.svc.cluster.local mysql
```

> Remember that while Kubernetes helps you set up a stateful applic you will need to set up the data cloning and data sync by yourself. T cannot be done by the StatefulSets.

Best Practices

If you plan to deploy stateful apps, like Oracle, MySQL, Elasticsearch, and MongoDB, use StatefulSets. They're a great option.

You need to consider the following points while creating stateful applications:

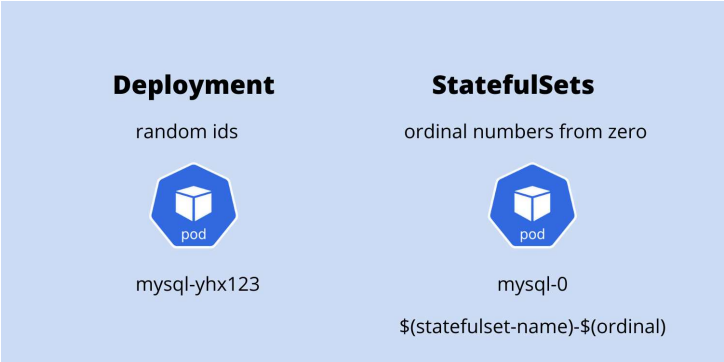
1. Create a separate namespace for databases.
2. Place all the needed components for stateful applications, such as ConfigMaps, Secr Services, in the particular namespace.
3. Put your custom scripts in the ConfigMaps.
4. Use headless service instead of load balancer service while creating Service objects.
5. Use the HashiCorp Vault for storing your Secrets.

6. Use the persistent volume storage for storing the data. Then your data won't be dele the Pod dies or crashes.

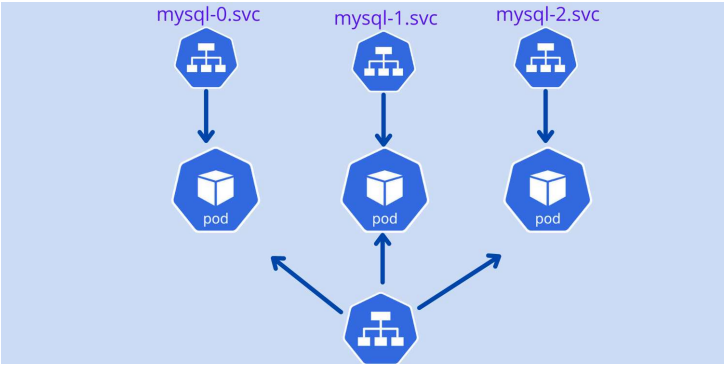
Deployment objects are the controllers most commonly used to create pods in Kuberne can easily scale these Pods by mentioning the replication count in the manifest file.

For stateless applications, Deployment objects are most suitable. For example, assume planning to deploy your Node.js application and want to scale it to five replicas. In this Deployment object is well suited.

The diagram below shows how Deployment and StatefulSets assign names to the Pods.



StatefulSets create ordinal service endpoints for each Pod created using the replica opt diagram below shows how the stateful Pod endpoints are created with ordinal numberi they communicate with each other.



Final Words

This article covered Kubernetes's two main controllers for creating Pods: Deployments StatefulSets. A Deployment object is best for stateless apps, while a StatefulSet is best apps.

The StatefulSets controller assigns each Pod an ordinal number, starting at zero. It helps apps set up a primary replica architecture. If a Pod dies, a new Pod is re-created using t name. This helpful feature does not break the chain of stateful application clusters. If y scaling down, it will be deleted in reverse order.

Use the StatefulSets controller in the Kubernetes cluster to deploy stateful applications Oracle, MySQL, Elasticsearch, and MongoDB. While cloning and syncing data must still b completed manually, StatefulSets eases the complexity involved in deploying stateful a

Photo by Photoholgic on Unsplash

Frequently Asked Questions

What are stateless and stateful Kubernetes

Kubernetes is a deployment and scaling platform for containerized applications. Kuberr applications usually have two states: stateful and stateless.

A stateful Kubernetes application receives data, stores it, and tracks it. A stateless Kube application receives data but does not store or track it.

What are the limitations of StatefulSets in Kubernetes?

While Statefulsets is a great tool, it has issues that could affect your experience. Here ar should know about

- Resource management: Statefulsets have two ways of storing resources: manual stor Persistent volume provisioner. If you use another approach, you could run into some You will likely have difficulty connecting your preferred storage to pods using Statefu
- Headless Service: To use a headless service with StatefulSet, you must set it up yours don't, it would be difficult to work with network identities.
- Volume Deletion: Normally, Statefulset lets you set the order in which pods stop if re But that changes if you delete a Statefulset completely. This could lead to potential l storage.

When should I use a StatefulSet?

Use Statefulset when deploying an application that needs stable identities for its pods. need to use that if that's not the case.

What is the impact of node failure on Statef in Kubernetes?

If a Kubernetes node fails, the application running on the Statefulset will not function w happens, you'll have to mark the node out-of-service. This will forcefully delete the pod node, allowing it to recover on another node quickly.

What is the purpose of a headless service in StatefulSet?

Pods in a StatefulSet have a unique identity. The Headless service allows them to comm directly.

Additional Articles You May Like:

- Kubernetes Development Environments - A Comparison
- Development in Kubernetes - Local vs. Remote Clusters
- Docker Compose Alternatives for Kubernetes
- Docker Compose to Kubernetes: Step-by-Step Migration
- 5 Key Elements for a Great Developer Experience with Kubernetes
- Kubernetes Multi-Tenancy – A Best Practices Guide
- Kubernetes Network Policies: A Practitioner's Guide
- Kubernetes RBAC: Basics and Advanced Patterns
- Kubernetes Readiness Probes - Examples & Common Pitfall
- 10 Essentials For Kubernetes Multi-Tenancy
- A Guide to Using Kubernetes for Microservices
- Kubernetes StatefulSet - Examples & Best Practices
- Kubernetes Service Account: What It Is and How to Use It

- Kubernetes NGINX Ingress: 10 Useful Configuration Options
- A Complete Guide to Kubernetes Cost Optimization
- Advanced Guide to Kubernetes Ingress Controllers
- GitOps + Kubernetes Explained
- Why Platform Engineering Teams Should Standardize on Kubernetes
- Platform Engineering on Kubernetes for Accelerating Development Workflows

Guides Kubernetes Insights Tools Tutorials Platform Engineering

Related blog posts

View all posts

December 18, 2024 • 7 Minute Read

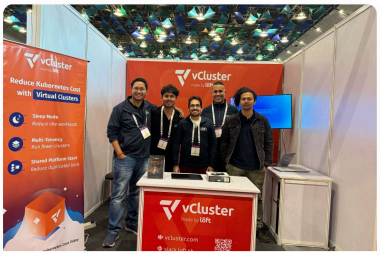
Kubernetes v1.32: Key Features, Updates, and What You Need to Know

The Kubernetes v1.32 release introduces significant advancements in resource management, security, scheduling, and observability. With a mix of 13 graduating features, 12 beta enhancements, and 19 got added in alpha, Kubernetes continues to refine its capabilities for production-...

December 17, 2024 • 6 Minute Read

Securing Multi-Tenant Kubernetes Clusters with Falco

We all agree that Kubernetes is running almost everywhere now, and nearly every other company has adopted it. The next step companies are taking is virtualizing this layer and creating multiple virtual Kubernetes clusters on a single host cluster. This is what we call multi-tenan...



December 15, 2024 • 3 Minute Read

KubeCon India 2024 Recap

So, it finally happened! The first-ever KubeCon in India took place at Yashobhoomi, Delhi, on the 11th and 12th of December! Loft Labs was proud to be one of the sponsors for this historic event. The excitement and enthusiasm I witnessed were next level. There were so many WOW mo...

Sign up for our newsletter

Be the first to know about new features, announcements and industry insights.

your@email.com*

Subscribe

Join our newsletter to stay up to date.

your@email.com*

Subscribe

By subscribing you agree to with our Privacy Policy and provide consent to receive updates from our company.

Resources

- Customer Stories
- Slack Community
- Blog

Product

- Getting Started
- Documentation
- Pricing
- Cost Optimization

Company

- About Us
- Careers
- Press & Media

Trending Topics

- Kubernetes Multi-tenancy
- Kubernetes + GitOps
- Kubernetes Cost Optimization