Lesson 03: Kubernetes Pods, Deployments, Statefull Set, ReplicaSet, DaemonSet

Lữ Thanh Tùng

Pods

- Pods là đơn vị triển khai nhỏ nhất, có thể tạo và quản lý trong Kubernetes
- Pod là tập các container cùng chia sẻ tài nguyên lưu trữ và mạng, và các thông tin cho cách chạy các container. Các container này được thực thi trên cùng một máy ảo, máy vật lý hoặc một máy chủ logic (cloud)

Tạo pods

- Để tạo pod đơn giản, ta có thể tạo 1 file yaml lưu cấu hình pod và thực hiện lênhsau để tạo pod:
  - kubectl apply -f tên-file.yaml

```
tung@tung-ideapad-5-pro:~$ kubectl apply -f hello-world.yaml
pod/hello-world-pod created
tung@tung-ideapad-5-pro:~$ kubectl logs hello-world-pod

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

tung@tung-ideapad-5-pro:~$ kubectl get pods
NAME              READY   STATUS             RESTARTS        AGE
hello-world-pod   0/1     CrashLoopBackOff   8 (2m34s ago)   18m
```

Tài nguyên workload cho quản lý pods

- Thông thường, chúng ta không cần phải tạo pods trực tiếp, ngay cả các pod đơn lẻ. Thay vào đó, chúng ta sẽ sử dụng các tài nguyên workload như Deployment, Job để khởi tạo chúng
- Có hai cách chính để sử dụng Pod trong kubernetes:
  - **Pod chỉ chạy một container:** Mô hình "one-container-per-Pod" là trường hợp sử dụng phổ biển nhất. Trong trường hợp này, Pod sẽ là lớp bao của một container, Kubernetes sẽ quản lý pods thay vì container.
  - **Pod chạy một tập các container làm việc cùng nhau:** Một Pod đóng gói một ứng dụng bao gồm nhiều container cùng vị trí được liên kết chặt chẽ và chia sẻ tài nguyên với nhau.
- Một Pod chạy một phiên bản của một ứng dụng. Nếu muốn scale horizontal, ta sẽ sử dụng nhiều Pod, mỗi Pod một phiên bản. Trong Kubernetes, điều này được gọi là replication.
- Replicated Pods được được tạo và quản lý theo nhóm bởi tài nguyên workload và bộ điều khiển của nó.

Chia sẻ tài nguyên và giao tiếp

- Pod cho phép chia sẻ dữ liệu và giao tiếp giữa các cointainer
- Tất cả các containers trong Pod có thể truy cập các ổ đĩa được chia sẻ điều này cho phép các containers chia sẻ dữ liệu. Các ổ đĩa này cũng cho phép dữ liệu trong Pod tồn tại khi một container cần khởi động lại
- Mỗi pod được gán một địa chỉ IP duy nhất
- Mỗi cointainer trong Pod chia sẻ không gian mạng (địa chỉ IP, cổng mạng) bằng cách sử dụng localhost.

Pod lifetime

- Pod được tạo, gán một ID duy nhất (UID) và được lên lịch cho các node đang tồn tại (node kết thúc khi khởi đông lại hoặc bị xóa) . Nếu một Node chết, các Pod đã được lên lịch cho node đó sẽ được xóa sau một khoảng thời gian chờ.
- Pod không thể tự phục hồi do đó nếu một Pod được lên lịch mà sau đó thất bại, thì Pod sẽ bị xóa. Tương tự như vậy, một Pod sẽ không tồn tại sau khi bị thu hồi do thiếu tài nguyên hoặc Node bảo trì. Kubernetes sử dung một controller để xử lý công việc quản lý các phiên bản Pod dùng một lần.
- Một Pod nhất định (như được xác định bởi UID) không bao giờ được "lên lịch lại", thay vào đó, Pod đó có thể được thay thế bằng một Pod mới, gần giống hệt, thậm chí có cùng tên nếu muốn, nhưng có UID khác.
- Nếu Pod bị xóa vì bất kỳ lý do gì và ngay cả khi một thay thế giống hệt được tạo, thì thứ liên quan với pod (như ổ lưu trữ) cũng bị hủy và được tạo lại.

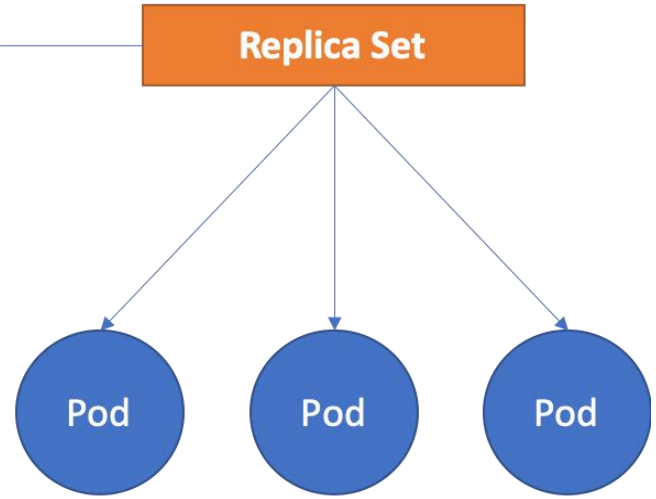
Các loại tài nguyên dùng cho deploy app trong kubernetes

- Deployment
- ReplicaSet
- StatefulSets
- DaemonSet

ReplicaSet

- Tạo ra các bản sao của Pod trong khi triển khai
- Làm cho số bản sao luôn chạy như trong khai báo

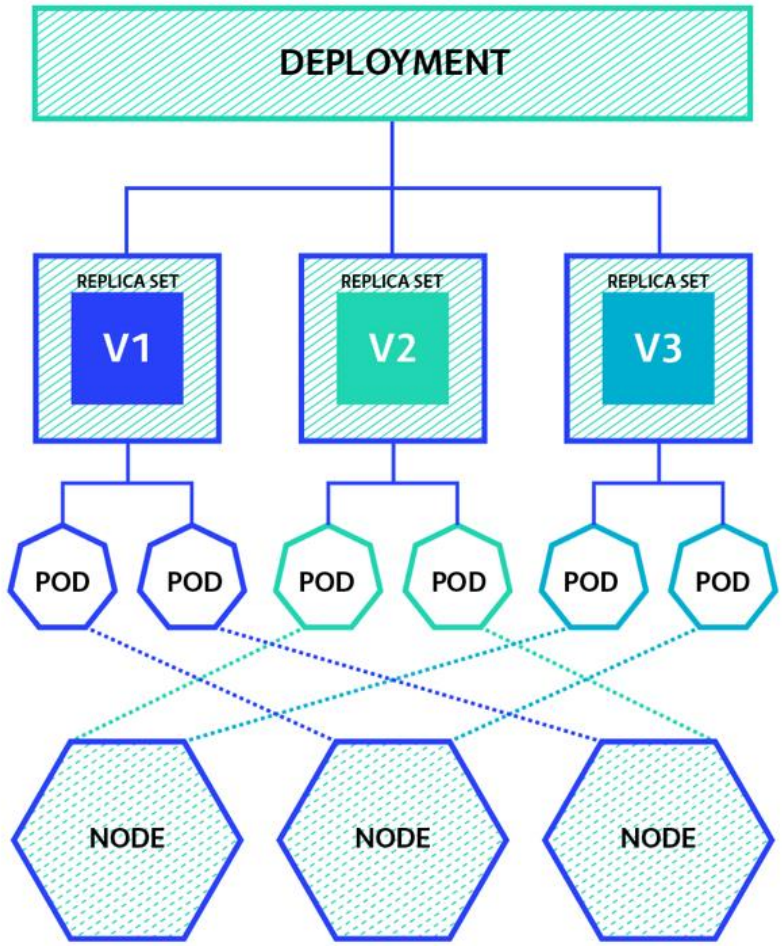
```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata: …
spec:
  replicas: 3
  selector: …
  template: …
```



Deployments

- Deployment giống ReplicaSet tuy nhiên nó có các cơ chế để quản lý các pod một cách hiệu quả, trong khi ReplicaSet thì không.

- Deployment cho phép tạo, cập nhật, hủy cập nhật các phiên bản khác nhau của các ứng dụng trong Kubernetes
- Cung cấp các cơ chế: rolling updates, rollbacks, and self-healing capabilities



Các trường hợp sử dụng

- Tạo một Deployment để triển khai một ReplicaSet
- Khai báo một trạng thái mới của Pod: bằng cách triển khai các ReplicaSet khác thay thế
- Phục hồi(Rollback) một phiên bản Deployment gần nhất
- Mở rộng (Scale) Deployment để đáp ứng nhiều tải hơn
- Tạm dừng rollout Deployment
- Dọn dẹp những ReplicaSet cũ

Tạo một Deployment

- Trong ví dụ bên ta viết 1 file yaml để tạo deployment
  - Trường .metadata.name mô tả tên của Deployment. Tên này sẽ trở thành tên của ReplicaSets and Pods nếu được tạo sau đó
  - Trường .spec.replicas mô tả số ReplicaSet được tạo ra
  - Trường .spec.selector field định nghĩa cách các ReplicaSet được tạo ra sẽ quản lý những Pod nào.
  - Trường .template field gồm những trường con
    - Trường .metadata.labels mô tả nhãn của Pod
    - Trường .template.spec mô tả các thông số của Pod. Trong ví dụ trên: Pods có một container, nginx, image tồn tại trên nginx Docker Hub ở phiên bản 1.14.2
- **Kết quả chạy:**

```
tung@tung-ideapad-5-pro:~$ kubectl apply -f nginx.yaml
deployment.apps/nginx-deployment created
tung@tung-ideapad-5-pro:~$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           16s
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Updating a Deployment

- Ta sẽ cập nhật nginx image lên nginx:1.16.1 thay cho nginx:1.14.2
  - kubectl set image deployment.v1.apps/nginx-deployment nginx=nginx:1.16.1
  - Trong đó deployment.v1.apps/nginx-deployment là tên của deplyment
  - nginx là tên của container cần cập nhât

```
tung@tung-ideapad-5-pro:~$ kubectl set image deployment.v1.apps/nginx-deployment nginx=nginx:1.16.1
deployment.apps/nginx-deployment image updated
tung@tung-ideapad-5-pro:~$ kubectl rollout status deployment/nginx-deployment
deployment "nginx-deployment" successfully rolled out
tung@tung-ideapad-5-pro:~$ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-66f8758855   3         3         3       60s
nginx-deployment-85996f8dbd   0         0         0       14m
tung@tung-ideapad-5-pro:~$ kubectl describe deployments
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Thu, 29 Jun 2023 11:52:02 +0700
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision: 2
Selector:               app=nginx
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:1.16.1
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  nginx-deployment-85996f8dbd (0/0 replicas created)
NewReplicaSet:   nginx-deployment-66f8758855 (3/3 replicas created)
Events:
```

Rolling Back a Deployment

- Khi update một deployment, đôi khi chúng ta cần quay lại phiên bản trước đó (ví du: hệ thống khi cập nhật không ổn định).

Để rollback ta cần làm

- Kiểm tra lịch sử Deployment bằng lệnh
  - kubectl rollout history deployment/nginx-deployment
  - Để thêm Change-Cause vào phiên bản hiện tại, ta dùng lệnh
    - kubectl annotate deployment/nginx-deployment kubernetes.io/change-cause="image updated to 1.16.1"
  - Ta có thể kiểm tra thay đổi chi tiết các phiên bản:
    - kubectl rollout history deployment/nginx-deployment --revision=2

```
tung@tung-ideapad-5-pro:~$ kubectl rollout history deployment/nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         <none>

tung@tung-ideapad-5-pro:~$ kubectl annotate deployment/nginx-deployment kubernet
es.io/change-cause="image updated to 1.16.1"
deployment.apps/nginx-deployment annotate
tung@tung-ideapad-5-pro:~$ kubectl rollout history deployment/nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         image updated to 1.16.1
```

Rolling Back a Deployment

- Quay về phiên bản trước đó:
  - Sử dụng kubectl rollout undo deployment/nginx-deployment để quay lại phiên bản trước
  - Sử dụng kubectl rollout undo deployment/nginx-deployment --to-revision=2 để chỉ định số phiên bản mà mình muốn quay lại

```
tung@tung-ideapad-5-pro:~$ kubectl annotate deployment/nginx-deployment kubernet
es.io/change-cause="image updated to 1.16.1"
deployment.apps/nginx-deployment annotate
tung@tung-ideapad-5-pro:~$ kubectl rollout history deployment/nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         image updated to 1.16.1

tung@tung-ideapad-5-pro:~$ kubectl rollout undo  deployment/nginx-deployment
deployment.apps/nginx-deployment rolled back
tung@tung-ideapad-5-pro:~$ kubectl rollout history deployment/nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
2         image updated to 1.16.1
3         <none>

tung@tung-ideapad-5-pro:~$ kubectl rollout undo  deployment/nginx-deployment --t
o-revision=2
deployment.apps/nginx-deployment rolled back
tung@tung-ideapad-5-pro:~$ kubectl rollout history deployment/nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
3         <none>
4         image updated to 1.16.1
```

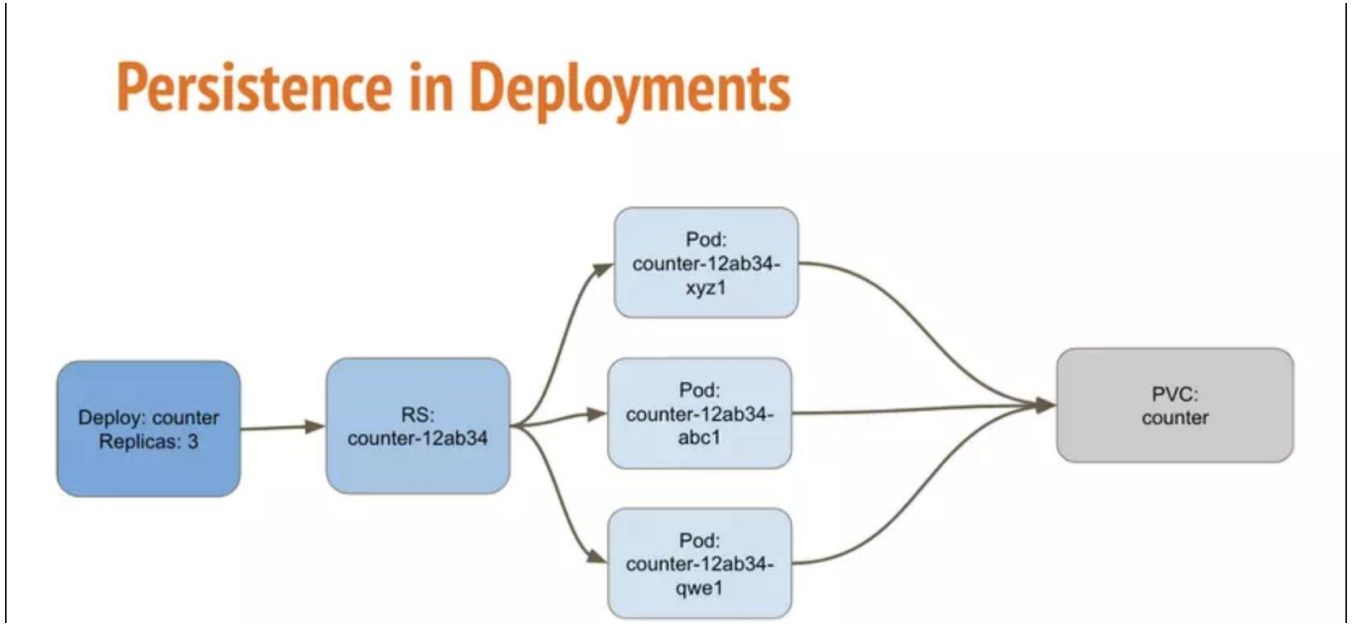
Scale Deployment

- Ta có thể mở rộng Deployment bằng lệnh
  - kubectl scale deployment/nginx-deployment --replicas=10

```
tung@tung-ideapad-5-pro:~$ kubectl scale deployment/nginx-deployment --replicas=10
deployment.apps/nginx-deployment scaled
tung@tung-ideapad-5-pro:~$ kubectl get deploy
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   10/10   10           10          3d13h
```

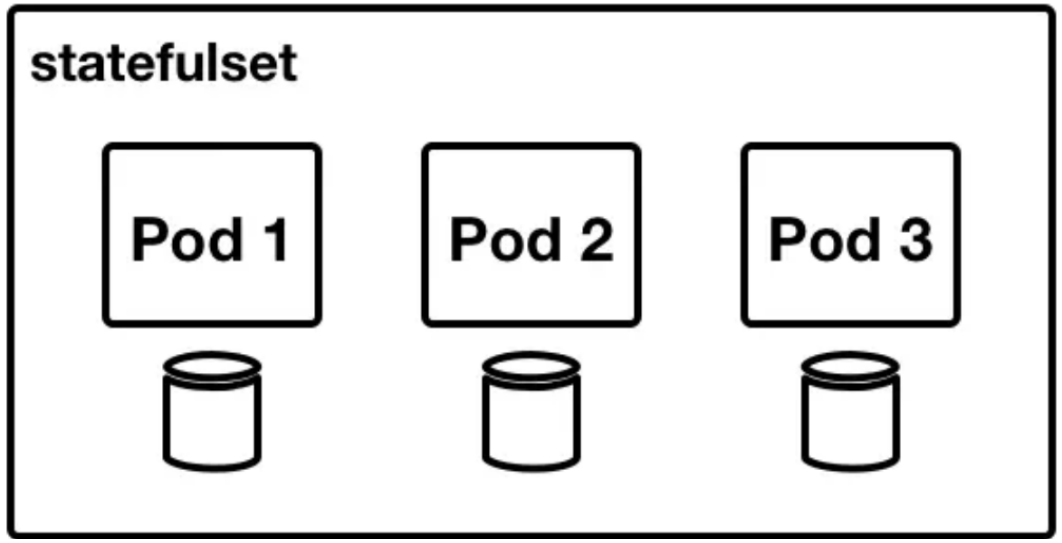
Deployment UseCase

- Deployment thường được sử dụng cho các ứng dụng stateless
- Ta cũng có thể lưu trạng thái triển khai bằng cách gắn cho nó một Persistent Volume và làm cho nó trở thành stateful. Nhưng lưu ý là ở đây tất cả các Pod của bạn sẽ cùng chia sẻ một Volume và dữ liệu của chúng cũng sẽ tương tự nhau.



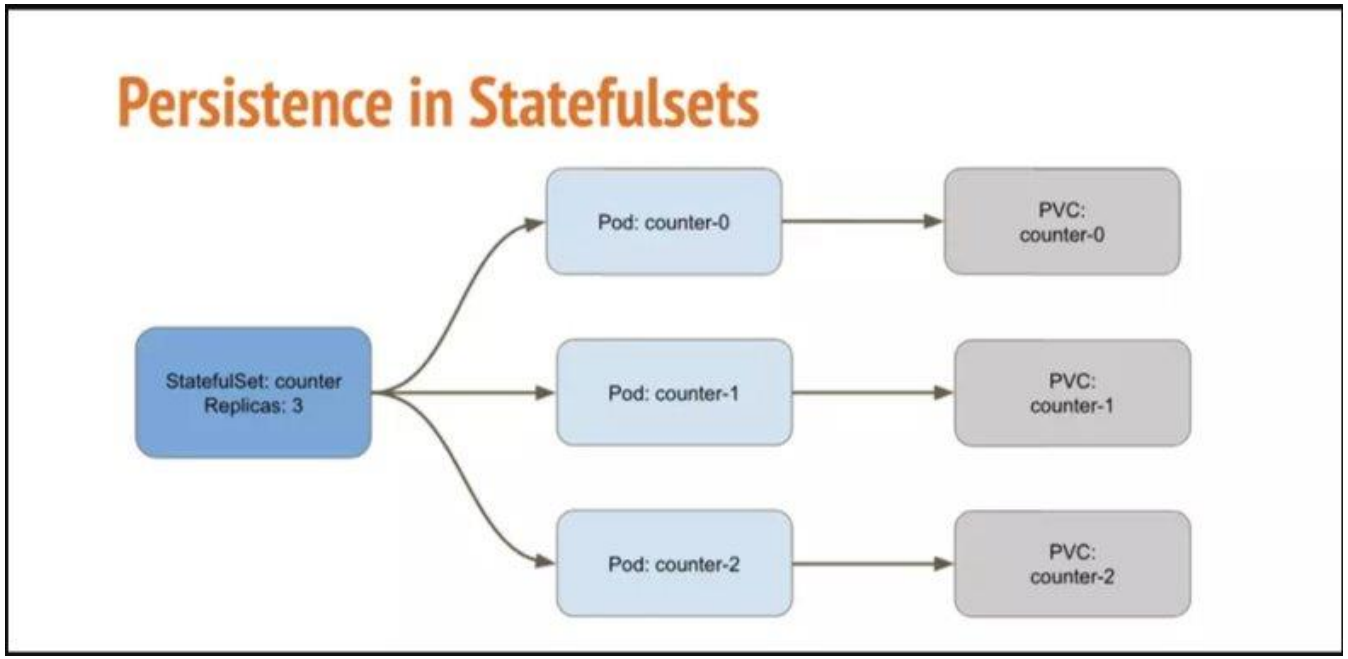
StatefulSets

- StatefulSet cũng là một Controller nhưng không giống như Deployments, nó không tạo ReplicaSet mà chính nó tạo Pod với quy ước đặt tên duy nhất. Ví dụ: Nếu bạn tạo StatefulSet với bộ đếm tên (counter), nó sẽ tạo một pod với tên *counter-0* và cho nhiều bản sao của một statefulset, tên của chúng sẽ tăng lên như *counter-0, counter-1, counter-2,* v.v.
- Mỗi bản sao của StatefulSet sẽ có trạng thái riêng và nếu sử dụng PersistentVolume thì mỗi Pod sẽ tạo PVC (Persistent Volume Claim) riêng. Vì vậy, một StatefulSet có 3 bản sao sẽ tạo ra 3 Pod, mỗi Pod có Volume riêng.



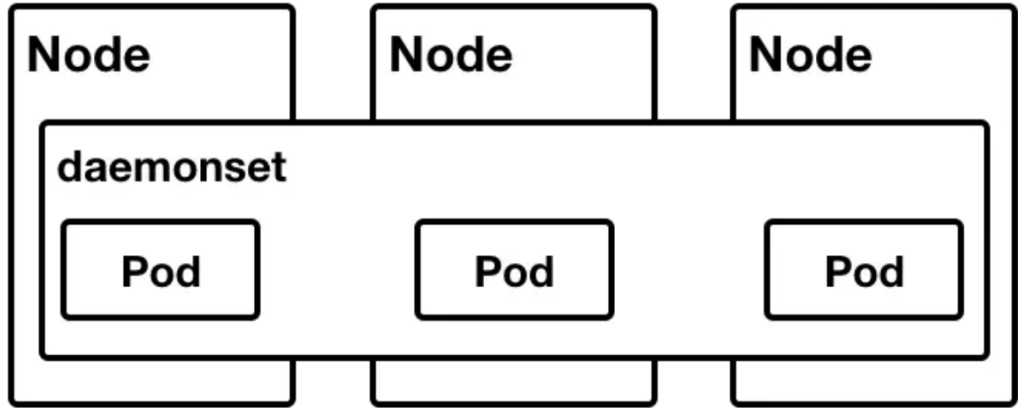
StatefulSets UseCase

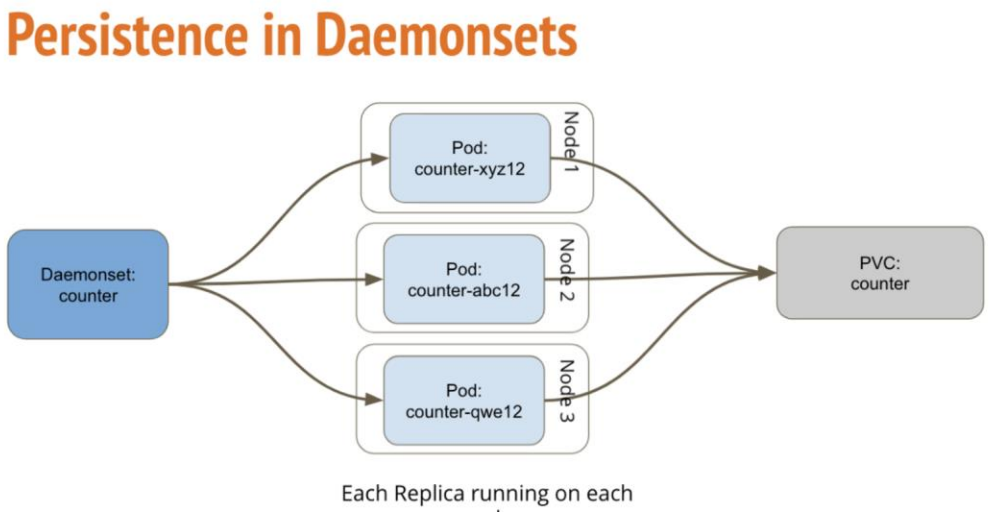
- Các dịch vụ dữ liệu (Database, lưu trữ key-value,v.v..)
- Các hệ thống nhạy cảm với việc định danh (consensus-systems, leader-election clustering,v.v..)
- Và bất cứ kiến trúc nào cần việc slow roll-out và liên kết theo cụm.



DaemonSets

- DaemonSets là một controller đảm bảo rằng Pod của bạn sẽ được chạy trên tất cả các node trong cụm. Và nếu một node được thêm/xoá khỏi cụm thì DaemonSets cũng sẽ tự động thêm/xoá Pod
- DaemonSets sẽ triển khai các Pod bằng số lượng node. Tuy nhiên, về mặt hoạt động, nó sẽ hoạt động tương tự như một Deployment, tức là tất cả các Pod cũng sẽ cùng chia sẻ một Persistent Volume.
- **Lưu ý**: Một chút lưu ý ở đây là DaemonSets sẽ không chạy trên các node có một *taint* (ví dụ: Master).





DaemonSets UseCase

- Monitoring Exporters: Chẳng hạn bạn muốn giám sát tất cả các node trong cụm của mình vì vậy bạn cần một trình giám sát chạy trên tất cả các node, ví dụ như Node-Exporter.
- Logs Collection Daemon: Tương tự, bạn cũng muốn thu được log từ tất cả các node thì bạn cũng cần một DaemonSet của log collector như Fluentd trên mỗi node.