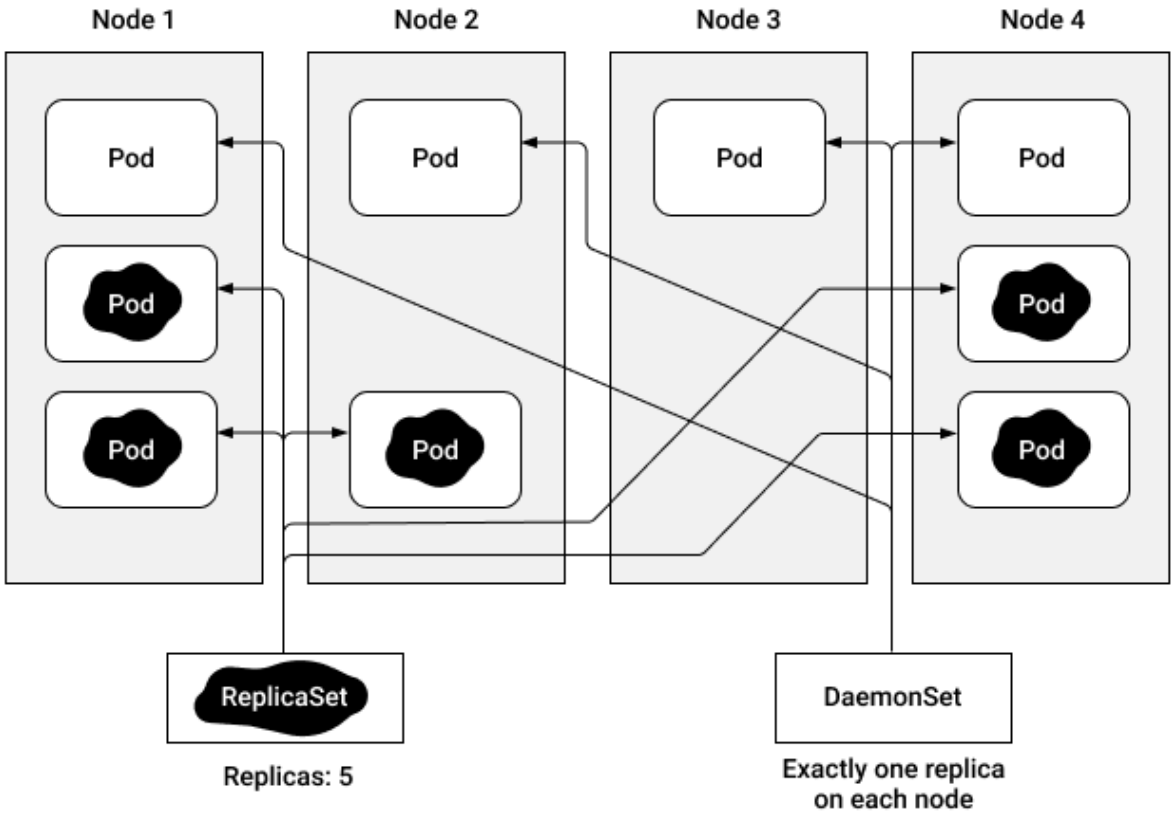
K8S 47: Kubernetes. Resources. DaemonSet

Описание:

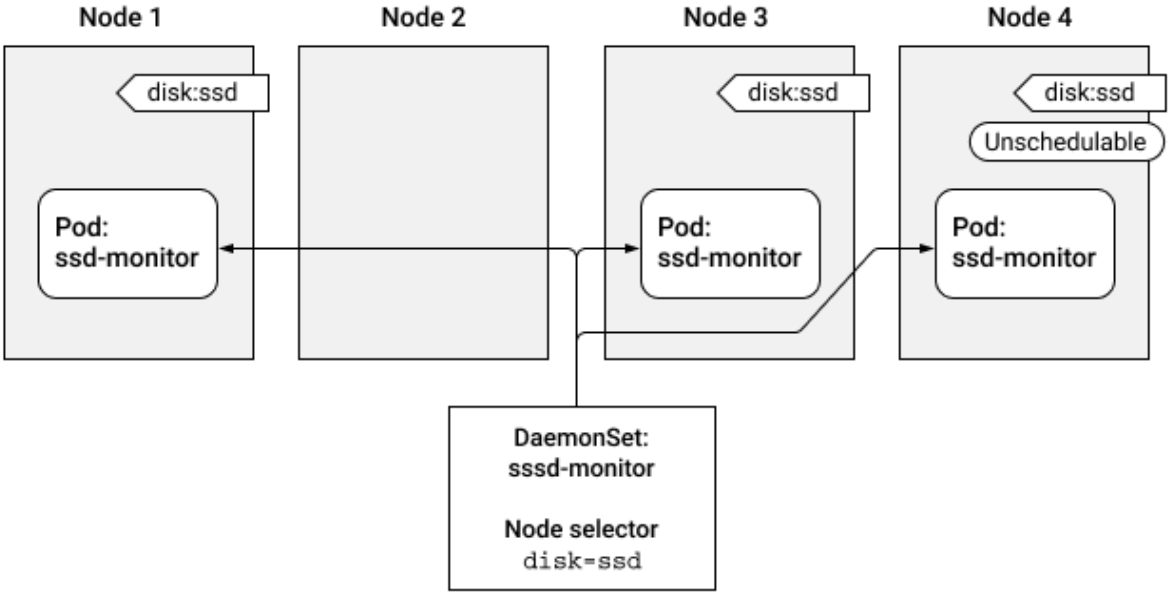
На данный момент мы познакомились с таким типом ресурсов, как Deployment, в котором можно определить в каком количестве (replicas) и где (nodeSelector) pods должны быть запущены. Но, допустим, нам нужно запускать приложение на каждой ноде? К примеру, для сбора метрик с ноды, конфигурации параметров хоста или запуска какого-то сервиса, который привязан к определенным нодам, но мы не хотим думать, сколько реплик нам нужно. Есть нода — контейнер должен быть на нем запущен. Для таких задач существует тип ресурсов под названием DaemonSet.
DaemonSet — это ресурс, который позволяет именно то, что мы описали, — запускает требуемую нагрузку на всех нодах, которые подходят по nodeSelector, в единичном экземпляре и следит за тем, чтобы она была запущена всегда, даже если она решила умереть.
В отличие от того же StatefulSet, с которым мы познакомились ранее, система именования подов похожа на Deployment — префикс, полученный из имени DaemonSet + случайная строка (5 символов), но управление подами производит сам DaemonSet, а не ReplicaSet, как в случае с Deployment.
Важной особенностью DaemonSet является то, что если у нас появляется новая нода, которая удовлетворяет nodeSelector, то на нем запустится под DaemonSet без необходимости вмешательства со стороны владельца кластера — Scheduler все сделает сам.
Схем обновления подов у DaemonSet 2:

- OnDelete — обновление будет произведено только при ручном удалении контейнеров. А до этого будет работать старый под.
- RollingUpdate — уже известная нам политика из Deployment.

Например, так работает kube-proxy. Один экземпляр должен быть запущен на каждом узле кластера.



При помощи nodeSelector можно заставить DaemonSet запускать pod только на определенных узлах кластера.



Пример описания DaemonSet:

```
apiVersion: apps/v1beta2
kind: DaemonSet
```

```
metadata:
  name: ssd-monitor
spec:
  selector:
  matchLabels:
    app: ssd-monitor
  template:
    metadata:
      labels:
      app: ssd-monitor
    spec:
      nodeSelector:
        disk: ssd
      containers:
      - name: main
        image: ifra/ssd-monitor
```

Полезные ссылки:

- DaemonSet (official docs)
- Perform a Rolling Update on a DaemonSet (official docs)
- DaemonSet (Google Kubernetes Engine docs)

Задание:

1. Напишите манифест DaemonSet, который будет запускать образ nginx:stable на каждой ноде с nodeSelector task: nodeSelector и назовите его daemonset-nodeSelector.
2. Выведите список нод с label task: nodeSelector (команду и вывод сохраните).
3. Примените данный манифест и выведите список подов, относящихся только к созданному Daemonset (одной командой без Pipeline) с опцией -o wide, чтобы видеть, на какой ноде запущен под (команду и вывод сохраните).
4. Скопируйте созданный ранее манифест, переименуйте его в daemonset-no-nodeSelector, удалите из него nodeSelector и добавьте политику обновления OnDelete.
5. Примените данный манифест и выведите список подов, где запущены поды DaemonSet (команду и вывод сохраните).
6. Измените в манифесте daemonset-no-nodeSelector образ на nginx:stable-alpine.

7. Примените данный манифест и выведите список подов, где запущены поды DaemonSet и через describe выведите описание одного из этих подов, чтобы проверить, что используется старый образ (команду и вывод сохраните).
8. Удалите один из подов DaemonSet и через describe выведите описание нового пода, который заменил удаленный, чтобы проверить, что используется новый образ (команду и вывод сохраните).
9. На проверку отправьте манифесты и все сохраненные выводы.