Pod & containers

Pod (под)

Pod это уровень абстракции. Он включает:

- совместное использование ресурсов
- управление планировщиком
- тестирование на рабочеспособность
- политики перезапуска
- политики безопасности
- контроль завершения
- тома

Он также абстрагирует детали рабочей нагрузки (контейнер, виртуальная машина, Wasm) но для некоторых могут потребоваться дополнительные модули (например KubeVirt для виртуалок).

Под смертен (после завершения или ошибки он удаляется без возможности перезапуска) и постоянны (для изменения нужно удалить старый и создать новый).

Настройка ноды для пода

NodeSelectors - список меток нодов. Простейший случай.

Affinity and anti-affinity - продвинутый способ Позволяет
attract
• Anti-affinity rules repel
• Hard rules must be obeyed
• Soft rules are only suggestions

Topology spread constraints - ограничения на топологию
Resource requests and resource limits

Теория процесса запуска pod

1. Создать YAML манифест
2. Отправить манифест на API сервер
3. Запрос будет аутентифицирован и авторизован
4. Спецификация будет проверена
5. Планировщик отфильтрует ноды на основе ограничений
6. Под будет привязан к удовлетворяющей требованиям ноде
7. Сервис kubelet на ноде получит задание

8. Сервис kubelet скачает спецификацию и сформирует задачу для исполнения
9. Сервис kubelet мониторит статус поды и в случае изменения направляет информацию на планировщик

Запуск пода

Есть два варианта: непосредственно через манифест на ноде или через контроллер. Первый вариант быстрый но без большинства возможностей (статический под, типа docker container). Второй вариант используется обычно.

Кластер создает сеть подов и автоматически подключает все поды к ней. Это одноуровневая L2 overlay сеть

Статусы пода

Pending под еще не создан, поиск ноды для запуска
Running нода найдена, запуск произведен
Init:X/Y исполнение инит контейнеров, завершено X из Y

Политики перезапуска настраиваются для контейнера. Поэтому пока идет перезапуск контейнера, считается что под еще работает. При обновлении под удаляется и создается новый. Это необходимо учитывать при разработке архитектуры.

Структура YAML файла

Верхний уровень

Параметр	Описание
Kind	Тип определяемого объекта. В данном случае Pod
apiVersion	Версия API
metadata	Метаданные
spec	Спецификация контейнеров

Metadata: метаданные пода

Параметр	Описание
name	Имя пода. Используется в качестве hostname у всех контейнеров для этого пода. Поэтому должно быть валидным DNS именем
labels	Метки пода

Spec: описание параметров контейнера, томов, внутри для каждого контейнера - name: имя_контейнера

Параметр	Описание
containers	Обычные контейнеры spec: initContainers: - name: ... containers: - name: ...
initContainers	Контейнеры, запускаемые до старта обычных контейнеров. Обычные запускаются только после завершения init контейнеров.
volumes:	Тома volumes: - name: html emptyDir: {}

Параметры контейнера:

Параметр	Описание
image	Образ image: nigelpoulton/k8sbook:1.0 Для использования другого хаба добавить URL перед именем образа
ports	Порты

Параметр	Описание

resources	ограничения на ресурсы
	`resources:`
	`  requests: <<==== Minimums for scheduling`
	`  cpu: 0.5`
	`  memory: 256Mi`
	`  limits: <<==== Maximums for kubelet to cap`
	`  cpu: 1.0`
	`  memory: 512Mi`
	Если нода с нужными ресурсами не найдена - под в статусе Pending. Для всех контейнеров в поде.

env	Переменные окружения.
	`env:`
	`- name: GIT_SYNC_REPO`
	`  value: https://github.com/nigelpoulton/ps-sidecar.git`

volumeMounts	Тома
	`volumeMounts:`
	`- name: html <<==== Mount shared volume`
	`  mountPath: /usr/share/nginx/`
	Настройки тома проводятся отдельно

Пример 1.

```
kind: Pod
apiVersion: v1
metadata:
  name: hello-pod
  labels:
    zone: prod
    version: v1
spec:
  containers:
  - name: hello-ctr
    image: nigelpoulton/k8sbook:1.0
    ports:
```

```
    - containerPort: 8080
    resources:
      limits:
        memory: 128Mi
        cpu: 0.5
```

Пример 2. Init контейнер.

```
apiVersion: v1
kind: Pod
metadata:
  name: initpod
  labels:
    app: initializer
spec:
  initContainers:
  - name: init-ctr
    # Pinned to 1.28 as newer versions have a sketchy nslookup command that doesn't work. Can also use a non-busybox image here
    image: busybox:1.28.4
    command: ['sh', '-c', 'until nslookup k8sbook; do echo waiting for k8sbook service; sleep 1; done; echo Service found!']
  containers:
    - name: web-ctr
      image: nigelpoulton/web-app:1.0
      ports:
        - containerPort: 8080
```

Пока образы в init контейнере не завершились, статус в Init:0/1

```
kubectl get pods --watch
NAME                         READY   STATUS     RESTARTS   AGE
initpod                      0/1     Init:0/1   0          16s
```

Пример 3. Дополнительный контейнер, обновляющий папку при изменении репозитория

```
# Some network drivers and laptop VM implementations cause issues with Service port mapping
# Minikube users may have to `minikube service svc-sidecar` to be able to access on `localhost:30001`
# Other users may have to run `kubectl port-forward service/svc-sidecar 30001:80`
apiVersion: v1
kind: Pod
```

```
metadata:
  name: git-sync
  labels:
    app: sidecar
spec:
  containers:
  - name: ctr-web
    image: nginx
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/
  - name: ctr-sync
    image: k8s.gcr.io/git-sync:v3.1.6
    volumeMounts:
    - name: html
      mountPath: /tmp/git
    env:
    - name: GIT_SYNC_REPO
      value: https://gitverse.ru/bobrobot/k8s_pods.git
    - name: GIT_SYNC_BRANCH
      value: master
    - name: GIT_SYNC_DEPTH
      value: "1"
    - name: GIT_SYNC_DEST
      value: "html"
  volumes:
  - name: html
    emptyDir: {}
---
apiVersion: v1
kind: Service
metadata:
  name: svc-sidecar
spec:
  selector:
    app: sidecar
  type: NodePort
  ports:
  - port: 80
    nodePort: 30001
```

При обращении на адрес кластера или ноды будет отображаться страница.

Основные команды

Команда	Доп. пар.	Описание
kubectl explain pods --recursive		Вывод всех параметров, доступных для конфигурирования Pod
kubectl get pods		список контейнеров
	-o yaml	расширенная информация о подах
kubectl apply	-f file.yml	Создание под из file.yml
kubectl describe	pod pod_name	Описание пода
kubectl logs pod_name		логи на поде. По умолчанию первого контейнера в поде
	--container cont_name	логи конкретного контейнера
kubectl exec		Выполнение команд внутри контейнера Два варианта
	pod_name -- command	Выполняет command на pod_name и возвращает результат в консоль. По умолчанию на первом контейнере. -- container для указания контейнера.
	-it pod_name -- command	Подключается в интерактивном режиме на контейнер и выполняет команду. `kubectl exec -it hello-pod -- sh`
kubectl edit pod pod_name		Редактирование под (для nano - в части дополнительные удобства Тестовый k8s) Пока не получилось.
kubectl delete	pod	Удаление подов. Имена подов через пробел
	svc	Удаление сервисов.
	-f	Удаление с использованием yaml файлов.

Container (контейнер)

Паттерны мультиконтейнеров

Init контейнеры - специальный тип контейнеров, для которых K8s гарантирует единственный запуск и завершение, перед остальными контейнерами. Пример: есть приложение и внешнее API с которым обязательно должно быть взаимодействие при старте. Вместо нагрузки на основную логику, можно процесс проверки вывести в init контейнер.

Slidecar контейнеры - выполняют периферийные задачи. Пока что бета.

Revision #10
Created 19 March 2025 18:20:25 by Admin
Updated 21 March 2025 14:26:38 by Admin