Раздел `Events` особенно полезен для отладки контейнеров, которые не работают должным образом, поскольку в нем записан каждый этап жизненного цикла контейнера вместе с любыми возникшими ошибками.

Работа с ресурсами

До сих пор мы использовали `kubectl` для выполнения запросов и вывода информации, а также для применения декларативных YAML-манифестов с помощью `kubectl apply`. Но у `kubectl` также есть полный набор *императивных* команд: операций для непосредственного создания или редактирования ресурсов.

Императивные команды kubectl

Один из примеров был показан в подразделе «Запуск демонстрационного приложения» на с. 61, где мы использовали команду `kubectl run`, которая автоматически создает развертывание для выполнения в заданном контейнере.

Большинство ресурсов можно создавать явным образом с помощью команды `kubectl create`:

```
kubectl create namespace my-new-namespace
namespace "my-new-namespace" created
```

Аналогично `kubectl delete` и удалит ресурс:

```
kubectl delete namespace my-new-namespace
namespace "my-new-namespace" deleted
```

Команда `kubectl edit` дает вам возможность просматривать и модифицировать любые ресурсы:

```
kubectl edit deployments my-deployment
```

Этим вы откроете свой стандартный текстовый редактор с файлом манифеста в формате YAML, который представляет заданный ресурс.

Это хороший способ получить детальную информацию о конфигурации любого ресурса, но также вы можете в редакторе внести и необходимые изменения. После сохранения файла и выхода из редактора `kubectl` обновит ресурс так, словно вы применили к манифесту ресурса команду `kubectl apply`.

Если вы допустили какие-либо ошибки (например, некорректный формат YAML), `kubectl` об этом сообщит и откроет файл, чтобы вы могли исправить проблему.

Когда не следует использовать императивные команды

В этой книге мы постоянно подчеркиваем важность использования *декларативной* инфраструктуры в качестве кода. Поэтому неудивительно, что мы не советуем прибегать к императивным командам `kubectl`.

Они могут быть крайне полезными для быстрого тестирования или проверки новых идей, но их основная проблема в том, что у вас нет единого *источника истины*. Вы не можете определить, кто и когда запускал те или иные команды в кластере и к каким результатам это привело. Как только вы выполните императивную команду, состояние вашего кластера перестанет быть синхронизированным с файлами манифестов, хранящимися в системе контроля версий.

В следующий раз, когда кто-нибудь применит YAML-манифесты, все изменения, которые вы внесли императивным образом, будут перезаписаны и утеряны. Это может привести к неожиданным последствиям и вызвать потенциально нежелательные эффекты для критически важных сервисов.

> Во время дежурства Элис неожиданно происходит большой скачок нагрузки на сервис, которым она занимается. Элис использует команду kubectl scale, чтобы увеличить количество реплик с пяти до десяти. Несколькими днями позже Боб редактирует YAML-манифест в системе контроля версий, чтобы перейти на новый образ контейнеров, и при этом не замечает, что текущее число реплик в файле равно пяти, хотя в промышленной среде их десять. Боб выкатывает изменения, чем уменьшает число реплик вдвое и вызывает немедленную перегрузку или перебой в работе.
>
> *Келси Хайтауэр и др. Kubernetes Up & Running*

Элис забыла обновить файлы в системе контроля версий после внесения императивных изменений. Такую ошибку легко допустить, особенно под давлением (см. подраздел «Дежурство не должно быть пыткой» на с. 368): в реальной жизни не все проходит гладко.

Так же и Боб перед повторным применением файлов манифеста должен был проверить изменения с помощью команды `kubectl diff` (см. подраздел «Сравнение ресурсов» на с. 166). Ведь если вы не ожидаете, что изменения были, несложно и упустить это из виду. К тому же Боб, наверное, не читал нашу книгу.

Лучший способ избежать подобного рода проблем — всегда вносить изменения путем редактирования и применения файлов ресурсов в системе контроля версий.

Рекомендуемый подход

Не используйте императивные команды kubectl, такие как create или edit, в промышленных кластерах. Вместо этого всегда управляйте ресурсами с помощью YAML-манифестов в системе контроля версий и применяйте их командой kubectl apply (или используя чарты Helm).

Генерация манифестов ресурсов

Несмотря на то что мы не советуем использовать `kubectl` в интерактивном режиме для внесения изменений в ваш кластер, императивные команды могут сэкономить немало времени при создании с нуля YAML-файлов для Kubernetes.

Вместо того чтобы набирать огромные шаблонные фрагменты в пустом файле, вы можете сгенерировать YAML-манифест с помощью `kubectl`:

```
kubectl run demo --image=cloudnatived/demo:hello --dry-run -o yaml
apiVersion: extensions/v1beta1
kind: Deployment
...
```

Флаг `--dry-run` говорит `kubectl` о том, что вместо создания самого ресурса `kubectl` следует вывести его манифест. Флаг `-o yaml` позволяет отобразить манифест ресурса в формате YAML. Вы можете сохранить этот вывод в файл, отредактировать его (если требуется) и применить для создания ресурса в кластере:

```
kubectl run demo --image=cloudnatived/demo:hello --dry-run -o yaml
    >deployment.yaml
```

Итак, внесите какие-нибудь правки с помощью предпочитаемого вами редактора, сохраните и примените результат:

```
kubectl apply -f deployment.yaml
deployment.apps "demo" created
```

Экспорт ресурсов

`kubectl` может помочь с созданием манифестов не только для новых, но и для уже существующих ресурсов. Представьте, к примеру, что вы создали развертывание с помощью интерактивной команды (`kubectl run`), затем отредактировали его, установив подходящие настройки, и теперь вам нужно написать для него декларативный манифест в формате YAML, который можно будет добавить в систему контроля версий.

Чтобы это сделать, укажите для команды `kubectl get` флаг `--export`:

```
kubectl run newdemo --image=cloudnatived/demo:hello --port=8888
--labels app=newdemo
deployment.apps "newdemo" created
kubectl get deployments newdemo -o yaml --export >deployment.yaml
```

Этот вывод имеет формат, позволяющий сохранить данный манифест вместе с другими манифестами, а затем обновить его и применить с помощью команды `kubectl apply -f`.

Если до сих пор вы использовали для управления своим кластером интерактивные команды `kubectl`, но хотите перейти на рекомендуемый нами декларативный стиль, есть отличный способ: экспортируйте все ресурсы своего кластера в файлы манифестов, используя команду `kubectl` с флагом `--export` (как мы показывали в нашем примере), и все будет готово.

Сравнение ресурсов

Прежде чем применять манифесты Kubernetes с помощью команды `kubectl apply`, было бы крайне полезно увидеть, что же на самом деле изменится в кластере. Для этого предусмотрена команда `kubectl diff`:

```
kubectl diff -f deployment.yaml
-  replicas: 10
+  replicas: 5
```

Благодаря этому выводу можно убедиться, что вносимые вами изменения приведут именно к тем результатам, которых вы ожидали. Вы также будете предупреждены, если состояние активного ресурса рассинхронизировано с YAML-манифестом — возможно, с того момента, когда вы применяли манифест в последний раз, кто-то отредактировал его императивным способом.

Рекомендуемый подход

Чтобы проверить потенциальные изменения перед тем, как применить их к своему промышленному кластеру, используйте kubectl diff.

Работа с контейнерами

Большая часть работы кластера Kubernetes происходит внутри контейнеров, поэтому, если что-то идет не так, разобраться бывает непросто. Ниже представлено несколько способов работы с запущенными контейнерами с помощью `kubectl`.