Странный NodePort

Дано

- Мы настроили `Deployment` и `Service` типа `NodePort` для сервиса UI
- Применили манифесты

Но ничего не работает

```
curl http://our.ip.goes.here:32530
curl: (7) Failed to connect to our.ip.goes.here port 32530: Connection refused
```

¯_(ツ)_/¯

Разбираемся

Connection refused

обычно означает, что мы пробились к серверу через облачные и локальные firewall

но на сервере никто не слушает этот порт

Проверим состояние нужного нам NodePort-сервиса:

```
1    $ kubectl get svc -n reddit-entry
2    NAME    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
3    ui      NodePort    10.10.255.169   <none>        9292:32530/TCP   15h
```

Уже неплохо. Сервис в рабочем состоянии, NodePort назначен.

Разбираемся

Может быть, сервис создан, но не знает, куда перенаправлять трафик?

```
1   kubectl -n reddit-entry describe service ui
2
3   Name:                     ui
4   Namespace:                reddit-entry
5   Labels:                   <none>
6   Annotations:              kubectl.kubernetes.io/last-applied-configuration:
7                               {"apiVersion":...
8   Selector:                 app=ui
9   Type:                     NodePort
10  IP:                       10.10.255.169
11  Port:                     <unset>  9292/TCP
12  TargetPort:               9292/TCP
13  NodePort:                 <unset>  32530/TCP
14  Endpoints:                10.10.16.3:9292,10.10.16.6:9292,10.10.17.5:9292 + 1 more...
15  Session Affinity:         None
16  External Traffic Policy:  Cluster
17  Events:                   <none>
```

И тут все хорошо... 😓

Разбираемся

На всякий случай, проверим состояние Deployment (хотя знаем, что если заполнены Endpoints, то с подами все хорошо):

```
kubectl -n reddit-entry describe deploy ui

Name:                   ui
...
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Containers:
   ui:
    Image:      gcr.io/cd-k8s-236617/user-10-ui:f5393576
    Port:       <none>
    Host Port:  <none>
...
```

Разбираемся

Применим “секретное оружие” - `kubectl port-forward`. Эта команда позволяет задействовать `kube-apiserver` в качестве прокси и подключиться с нашего рабочего места к поду внутри кластера. Это как SSH port-forwarding, но для Kubernetes.

> Краткий синтаксис: `kubectl port-forward TYPE/NAME [options] LOCAL_PORT:REMOTE_PORT`

Разбираемся

Несколько оговорок про `port-forward`:

- Весь трафик port-forward идет через API-сервер Kubernetes. Поэтому - это инструмент для диагностики и отладки. **Никакого high-load**, пожалуйста!
- Несмотря на то, что в аргументах можно указать Deployment, Service или конкретный под - эта команда просто **выберет один конкретный под** и подключится к нему. Переподключаться к другому поду она не будет.

> Мы указываем любой локальный порт, какой захотим, и порт **который слушает приложение в поде**. Не NodePort и не ClusterIP!

Разбираемся

```
kubectl port-forward -n reddit-entry svc/ui 9292:9292
Forwarding from 127.0.0.1:9292 -> 9292

curl http://127.0.0.1:9292
curl: (7) Failed to connect to 127.0.0.1 port 9292: Connection refused
```

Как мы видим - никто в поде не ответил на порту 9292.

Штирлиц насторожился... 👱

Разбираемся

Теперь сходим внутрь нашего пода и посмотрим, что там творится... Для этого у нас есть другое “секретное оружие”...

```
kubectl -n reddit-entry get pods -l app=ui
NAME                  READY   STATUS    RESTARTS   AGE
ui-5875f9bb88-cjjf6   1/1     Running   0          46m
ui-5875f9bb88-sq6wl   1/1     Running   0          46m

kubectl -n reddit-entry exec -ti ui-5875f9bb88-cjjf6 -- /bin/sh

/app # ps aux
PID   USER     TIME  COMMAND
    1 root      0:01 python3 post_app.py
   10 root      0:00 /app_runtime/venv/bin/python3 post_app.py
   16 root      0:00 /bin/sh
   21 root      0:00 ps aux
```

Внезапно! Внутри пода оказался сервис `post`, который слушает порт 5000

Разбираемся

Осталось понять, как так получилось... Откроем репозиторий сервиса `post` на GitLab и посмотрим на код пайплайна:

```
1   build:
2   ...
3     script:
4       - docker build -t gcr.io/reddit_repo/user-XX-ui:$CI_COMMIT_SHORT_SHA ./
5       - docker push gcr.io/reddit_repo/user-XX-ui:$CI_COMMIT_SHORT_SHA
6
7   release:
8   ...
9     script:
10      - docker pull gcr.io/reddit_repo/user-XX-ui:$CI_COMMIT_SHORT_SHA
11      - docker tag gcr.io/reddit_repo/user-XX-ui:$CI_COMMIT_SHORT_SHA
12  gcr.io/reddit_repo/user-XX-ui:$CI_COMMIT_TAG
        - docker push gcr.io/reddit_repo/user-XX-ui:$CI_COMMIT_TAG
```

Теперь ясно - образы сервиса `post` были названы и протегированы как `ui`. И именно они приехали в наш кластер.

Как сделать, чтоб так не было

- Задавать для подов `livenessProbe`, `readinessProbe`.
  - Мы бы сразу заметили, что поды в нерабочем состоянии (но не для перепутанных `ui` и `comment`).
  - Деплой такого ошибочного образа завершился бы ошибкой (и в кластере осталась бы работоспособная версия)
- Использовать “типовые” пайплайны с параметрами (например, `APP_NAME`) вместо хард-кода имен.

Как сделать, чтоб так не было

- Запретить изменяемые тэги образов в репозитории (без автоматизации выкатки - это жестоко).
  - С изменяемыми тэгами стоит помнить про кэширование образов на нодах Kubernetes (`imagePullPolicy` - поможет)
- Разграничить доступ команд к репозиториям, чтобы `post`-команда не могла пушить в `ui`-репозиторий.

Другие варианты диагностики

- `kubectl logs` - беглый взгляд на логи подов показал бы, что внутри “не то пальто”

Конец