

ДЗ. Отладка и тестирование в Kubernetes

Выполнение ДЗ | План работы

- Работа выполняется в ветке `kubernetes-debug`
- Не забудьте про ведение файла `README` и оформление PR
- В тех случаях, где в ДЗ обозначены проблемы, можно попытаться решить их самостоятельно, не забегая на следующие слайды
- Используемые файлы манифестов доступны в [репозитории со сниппетами](#)

kubectl debug

На лекции показывали работу `kubectl debug` и вызов `strace` для получения сведений о запущенном в поде процессе.

Но после установки `kubectl-debug` и запуска `strace` мы получаем такую картинку:

```
1  bash-5.0# strace -c -p1
2  strace: test_ptrace_get_syscall_info: PTRACE_TRACEME: Operation not permitted
3  strace: attach: ptrace(PTRACE_ATTACH, 1): Operation not permitted
```

kubectl debug | Задание

- Установите в ваш кластер `kubectl debug`:
 - Выполнив `brew install aylei/tap/kubectl-debug`
 - Или скачав [отсюда](#) архив с исполняемым файлом и добавив его в `$PATH`
- Запустите в кластере поды с агентом `kubectl-debug` из [этого](#) манифеста

kubectl debug | Задание

- Проверьте работу команды `strace` на любом поде (можно использовать Web-сервер из предыдущих заданий)
- Определите, почему не работает `strace` и почините
- Необходимые для выполнения задания файлы положите в папку `kubernetes-debug/strace`
 - Если есть shell-скрипт, он должен называться `run.sh`
 - Если есть файлы с манифестами - можно просто сложить их в папку
 - Инструкция по запуску должна быть в README

kubectl debug | Подсказки

- Возможность запуска трассировки определяется наличием у процесса capability `SYS_PTRACE`
- Когда запущен debug-контейнер (не агент!) - посмотрите наличие нужных прав у него используя `docker inspect`
- Посмотрите на исходный код агента `kubectl-debug` - выставляет ли он нужные capabilities при создании debug-контейнера и историю изменений файла.
- Проверьте, какая версия кода используется в запущенном у вас поде с агентом `kubectl-debug`

iptables-tailer

Один из полезных инструментов, который был упомянут, но не показан на лекции - это `kube-iptables-tailer`. Он предназначен для того, чтобы выводить информацию об отброшенных iptables пакетах в журнал событий Kubernetes (`kubectl get events`)

Основной кейс - сообщить разработчикам сервисов о проблемах с NetworkPolicy.

Итак, приступим...

iptables-tailer

Чтобы выполнить домашнее задание нам потребуется следующее:

- Кластер с установленным и запущенным Calico (для GKE - это просто включенные галки Network Policy)
- Для нод K8s лучше использовать Ubuntu 18.xx или новее (Fedora, CentOS 7/8 тоже можно, если заведете)
- Тестовое приложение
- Инсталляция `kube-iptables-tailer`
- Результаты работы должны быть в папке `kubernetes-debug/kit`

iptables-tailer | Тестовое приложение

Для нашего задания в качестве тестового приложения мы возьмем `netperf-operator`

- Это Kubernetes-оператор, который позволяет запускать тесты пропускной способности сети между нодами кластера
- Сам проект - не очень production-grade, но иногда выручает

iptables-tailer | Тестовое приложение

- Установите манифесты для запуска оператора в кластере (лежат в папке `deploy` в репозитории проекта):
 - Custom Resource Definition - схема манифестов для запуска тестов Netperf
 - RBAC - политики и разрешения для нашего оператора
 - И сам оператор, который будет следить за появлением ресурсов с `Kind: Netperf` и запускать поды с клиентом и сервером утилиты NetPerf

```
1 kubectl apply -f ./deploy/crd.yaml
2 kubectl apply -f ./deploy/rbac.yaml
3 kubectl apply -f ./deploy/operator.yaml
```

iptables-tailer | Тестовое приложение

Теперь можно запустить наш первый тест, применив манифест

`cr.yaml` из папки `deploy`

```
1  #> kubectl apply -f ./deploy/crd.yaml
2
3  #> kubectl describe netperf.app.example.com/example
4  Name:          example
5  Namespace:     default
6  Labels:        <none>
7  Annotations:   ...
8  API Version:   apps/v1
9  Kind:          Netperf
10 Metadata: ...
11 Spec: ...
12 Status:
13   Client Pod:    netperf-client-42010aa6009b
14   Server Pod:    netperf-server-42010aa6009b
15   Speed Bits Per Sec: 915.42
16   Status:        Done
17   Events:        <none>
```

iptables-tailer | Тестовое приложение

Если в результатах `kubectl describe` вы увидели `Status: Done` и результат измерений, значит все прошло хорошо (обычно на тест нужно 1-2 минуты).

Теперь можно добавить сетевую политику для Calico, чтобы ограничить доступ к подам Netperf и включить логирование в `iptables`.

iptables-tailer | Сетевые политики

```
1  apiVersion: crd.projectcalico.org/v1
2  kind: NetworkPolicy
3  metadata:
4    name: netperf-calico-policy
5    labels:
6  spec:
7    order: 10
8    selector: app == "netperf-operator"
9    ingress:
10     - action: Allow
11       source:
12         selector: netperf-role == "netperf-client"
13     - action: Log
14     - action: Deny
15    egress:
16     - action: Allow
17       destination:
18         selector: netperf-role == "netperf-client"
19     - action: Log
20     - action: Deny
```

iptables-tailer | Сетевые политики

Теперь, если повторно запустить тест, мы увидим, что тест висит в состоянии *Starting*. В нашей сетевой политике есть ошибка.

Проверьте, что в логах ноды Kubernetes появились сообщения об отброшенных пакетах:

- Подключитесь к ноде по SSH
- `iptables --list -nv | grep DROP` - счетчики дропов ненулевые
- `iptables --list -nv | grep LOG` - счетчики с действием логирования ненулевые
- `journalctl -k | grep calico`

iptables-tailes | Сетевые политики

```
1  root@gke-node01:~# journalctl -k | grep
2  calico-pack
3
4  Sep 02 15:33:33 gke-node01 kernel: cali
5  packet: IN=cali4c143f2e8e0 OUT=eth0
6  MAC=ee:ee:ee:ee:ee:ee:de:ad:be:ef:ca:fe
7  SRC=10.48.1.7 DST=10.48.0.12 LEN=60 TOS
8  PREC=0x00 TTL=63 ID=42011 DF PROTO=TCP
9  SPT=35823 DPT=12865 WINDOW=28400 RES=0x
10  URGP=0
11
12  Sep 02 15:33:34 gke-node01 kernel: cali
13  packet: IN=cali4c143f2e8e0 OUT=eth0
    MAC=ee:ee:ee:ee:ee:ee:de:ad:be:ef:ca:fe
    SRC=10.48.1.7 DST=10.48.0.12 LEN=60 TOS
    PREC=0x00 TTL=63 ID=42012 DF PROTO=TCP
    SPT=35823 DPT=12865 WINDOW=28400 RES=0x
    URGP=0
    Sep 02 15:33:36 gke-node01 kernel: cali
    packet: IN=cali4c143f2e8e0 OUT=eth0
    MAC=ee:ee:ee:ee:ee:ee:de:ad:be:ef:ca:fe
    SRC=10.48.1.7 DST=10.48.0.12 LEN=60 TOS
    PREC=0x00 TTL=63 ID=42013 DF PROTO=TCP
    SPT=35823 DPT=12865 WINDOW=28400 RES=0x
    URGP=0
    Sep 02 15:33:40 gke-node01 kernel: cali
    packet: IN=cali4c143f2e8e0 OUT=eth0
    MAC=ee:ee:ee:ee:ee:ee:de:ad:be:ef:ca:fe
    SRC=10.48.1.7 DST=10.48.0.12 LEN=60 TOS
    PREC=0x00 TTL=63 ID=42014 DF PROTO=TCP
    SPT=35823 DPT=12865 WINDOW=28400 RES=0x
```

iptables-tailes | Сетевые политики

Вроде бы, проблемы с сетевыми политиками можно диагностировать таким образом. Но...

Очевидно, что давать всем подряд доступ по SSH к нодам кластера и чтению kernel log - неудачная идея

Кроме того, из этих сообщений непонятно, с какими подами у нас проблема.

iptables-tailer to the rescue! 

iptables-tailes | Установка

- Попробуем запустить `iptables-tailer` используя манифест из репозитория проекта
- Проверим логи запущенного пода
- В зависимости от степени везения, мы можем увидеть в логе кучу ошибок, связанных с тем сервис не имеет права на листинг подов с дефолтным `ServiceAccount`
- Это исправляется созданием отдельного сервис-аккаунта с правами на просмотр информации о подах и созданием Event-ресурсов


iptables-tailes | Проверка

- Теперь можно снова запустить тесты NetPerf (удалив и снова применив манифест `cr.yaml`)
- Проверим логи пода `iptables-tailer` и события в кластере (`kubectl get events -A`)
- И опять, мы ничего не увидим. А жаль...
- В манифесте с DaemonSet была переменная, которая задавала префикс для выбора логов `iptables`
 - В ней указан префикс `calico-drop` , а по умолчанию Calico логирует пакеты с префиксом `calico-packet`

Исправим...

iptables-tailes | Проверка

Исправить это можно или исправив префикс в манифесте DaemonSet или указав его в сетевой политике. В целом, любой способ сработает.

Снова запустим тесты NetPerf. И скорее всего, снова ничего не увидим... 

iptables-tailer | Проверка

В манифесте из репозитория `kube-iptables-tailer` настроен так, что ищет текстовый файл с логами `iptables` в определенной папке.

Но во многих современных Linux-дистрибутивах логи по умолчанию не будут туда отгружаться, а будут складываться в журнал `systemd`.

К счастью, `iptables-tailer` умеет работать с systemd journal - для этого надо передать ему параметр `JOURNAL_DIRECTORY` б указав каталог с файлами журнала (по умолчанию, `/var/log/journal`). Задайте его в манифесте.

iptables-tailes | Проверка

После применения манифеста опять что-то пошло не так.

Если посмотрим на логи - то увидим, что необходимо пересобрать образ с `kube-iptables-tailer`, включив опцию C-Go (для связывания с C-библиотекой, которая обеспечивает чтение журнала systemd).

К счастью, можно просто пролистать до следующего слайда...

iptables-tailes | Изи

Создадим DaemonSet используя манифест из **репозитории со
сниппетами**

- Проверим логи одного из созданных подов DaemonSet - теперь контейнеры должны нормально запуститься
- Снова запустим тесты NetPerf и проверим события в кластере Kubernetes:
 - `kubectl get events -A`
 - `kubectl describe pod --selector=app=netperf-operator`

iptables-tailes | Иззи

Должно получиться как-то так:

```
1  << SKIPPED >>
2  Events:
3      Type          Reason          Age    From          Message
4      ----          -
5      Normal        Scheduled       73s    default-scheduler    Successfully assigned
6      default/netperf-server-42010aa60024 to gke-debug-hw-default-pool
7      Normal        Pulled         72s    kubelet, gke-debug-hw-default-pool    Container image
8      "tailoredcloud/netperf:v2.7" already present on machine
9      Normal        Created        72s    kubelet, gke-debug-hw-default-pool    Created container
      Normal        Started        72s    kubelet, gke-debug-hw-default-pool    Started container
      Warning       PacketDrop     70s    kube-iptables-tailer    Packet dropped when
      receiving traffic from 10.48.0.14
```

1 Теперь мы видим проблемы с сетевыми политиками, когда смотрим описание нашего Pod

Задание со

- Исправьте ошибку в нашей сетевой политике, чтобы Netperf снова начал работать
- Поправьте манифест DaemonSet из репозитория, чтобы в логах отображались имена Podов, а не их IP-адреса

Проверка ДЗ

- Результаты вашей работы должны быть добавлены в ветку **kubernetes-debug** вашего GitHub репозитория `<YOUR_LOGIN>_platform`
- В **README.md** рекомендуется внести описание того, что сделано
- Создайте Pull Request к ветке **master** (описание PR рекомендуется заполнять)
- Добавьте метку `kubernetes-debug` к вашему PR

Проверка ДЗ

Данное задание будет проверяться в полуавтоматическом режиме. Не пугайтесь того, что тесты в итоге завершатся неуспешно.

При этом смотрите в лог **Travis**, чтобы понять, действительно ли они дошли до “правильной ошибки”, говорящей о том, что дальнейшая проверка будет производиться вручную.

Проверка ДЗ

Структура репозитория после выполнения домашнего задания:

```
1  ...
2  └─ kubernetes-debug
3      ...
4      └─ strace
5          └─ ...
6      └─ kit
7  ...
```