Домашняя работа: Helmfile + Istio

Примеры API запросов

```
sudo kubectl port-forward -n istio-system svc/istio-ingressgateway 80:80
```

muffin-wallet API

```
curl http://wallet.example.com/actuator/health

# Посмотреть все кошельки
curl http://wallet.example.com/v1/muffin-wallets

# Создать кошелёк Jack
curl -X POST http://wallet.example.com/v1/muffin-wallets \
  -H "Content-Type: application/json" \
  -d '{"owner_name": "Jack"}'

# Создать кошелёк Bob
curl -X POST http://wallet.example.com/v1/muffin-wallets \
  -H "Content-Type: application/json" \
  -d '{"owner_name": "Bob"}'

# Получить кошелёк по ID = 101
curl http://wallet.example.com/v1/muffin-wallet/101

# Транзакция
curl -X POST http://wallet.example.com/v1/muffin-
wallet/{from_id}/transaction \
  -H "Content-Type: application/json" \
  -d '{"to_muffin_wallet_id": "{to_id}", "amount": 10}'

# Поиск по имени владельца
curl "http://wallet.example.com/v1/muffin-wallets?owner_name=Bob"
```

muffin-currency API (через port-forward)

```
# Через port-forward (без istio)
kubectl port-forward svc/muffin-currency 8083:8083
# Получить курс обмена
curl "http://localhost:8083/rate?from=PLAIN&to=CHOKOLATE"

# Или из muffin-wallet
```

```
kubectl exec deploy/muffin-wallet -c istio-proxy -- \
  curl -s "http://muffin-currency:8083/rate?from=PLAIN&to=CHOKOLATE"
```

Установка Istio

```
istioctl install --set profile=demo -y

# Включить sidecar
kubectl label namespace default istio-injection=enabled

# Посмотреть поды istio-system
kubectl get pods -n istio-system

# Посмотреть label
kubectl get namespace default --show-labels
```

Istio Ingress Gateway

Проверить что Gateway и VirtualService созданы

```
kubectl get gateway

# Результат
NAME                    AGE
muffin-wallet-gateway   43h
```

```
kubectl get virtualservice

# Результат
NAME               GATEWAYS                    HOSTS
AGE
muffin-wallet-vs   ["muffin-wallet-gateway"]   ["wallet.example.com"]
43h
```

Доступ через Istio Gateway

```
# Теперь к приложению можно обращаться так:
curl http://wallet.example.com/v1/muffin-wallet/101
```

У меня был запущен `sudo ... port-forward`, чтобы не писать `wallet.example.com:8080`, иначе можно сделать так:

```
kubectl port-forward -n istio-system svc/istio-ingressgateway 8080:80

# К приложению образаться так:
curl http://wallet.example.com:8080/actuator/health
```

Observability (метрики и трейсинг)

Установка Kiali, Prometheus + бонусом Grafana

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-
1.20/samples/addons/prometheus.yaml
kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-
1.20/samples/addons/grafana.yaml
kubectl apply -f https://raw.githubusercontent.com/istio/istio/release-
1.20/samples/addons/kiali.yaml

# Проверка, что всё запустилось
kubectl get pods -n istio-system
```

Генерация трафика для метрик

Генерация трафика:

```
curl -s http://wallet.example.com/actuator/health > /dev/null
curl -s http://wallet.example.com/ > /dev/null

# + Запрос к muffin-currency
kubectl exec deploy/muffin-wallet -c istio-proxy -- \
  curl -s "http://muffin-currency:8083/rate?from=PLAIN&to=CHOKOLATE"
```

Kiali — граф сервисов и трафик

```
istioctl dashboard kiali
```

Результат:

СКРИН

Prometheus — метрики

```
kubectl port-forward -n istio-system svc/prometheus 9090:9090
```

```
# Далее открываем в браузере http://localhost:9090 (почему-то istioctl
dashboard prometheus у меня не работает)
```

Примеры запросов:

```
# Количество запросов к muffin-wallet
istio_requests_total{destination_service_name="muffin-wallet"}

# Количество запросов к muffin-currency
istio_requests_total{destination_service_name="muffin-currency"}

# Error rate за последние 5 минут
rate(istio_requests_total{
  destination_service_name="muffin-wallet",
  response_code!="200"
}[5m])

# RPS к muffin-wallet
rate(istio_requests_total{destination_service_name="muffin-wallet"}[1m])

# RPS к muffin-currency
rate(istio_requests_total{destination_service_name="muffin-currency"}
[1m])
```

Grafana — dashboards с графиками

```
istioctl dashboard grafana
```

Переходим в **Dashboards**, открываем **Istio Service Dashboard**, в **Service** выбираем нужный сервис.

Увидим такой результат:

СКРИН

Security (mTLS + AuthorizationPolicy)

Проверка, что ресурсы созданы

```
kubectl get peerauthentication

# Результат
NAME                                      MODE     AGE
peerauthentication.security.istio.io/default   STRICT   3s
```

```
kubectl get authorizationpolicy

# Результат
NAME                                                                ACTION
AGE
authorizationpolicy.security.istio.io/muffin-currency-policy   ALLOW
3s
```

```
kubectl get serviceaccount muffin-wallet

# Результат
NAME                             SECRETS   AGE
serviceaccount/default           0         46h
serviceaccount/muffin-wallet     0         3s
```

Проверка mTLS в Kiali

```
istioctl dashboard kiali
```

Видим замочки на линиях между сервисами — это mTLS. Вот как это выглядит:

СКРИН

Проверка mTLS через обращения к сервису

```
# Попытка доступа к muffin-currency из другого namespace
kubectl create namespace no-istio
kubectl run test-no-sidecar -n no-istio --image=curlimages/curl --rm -it
--restart=Never \
  -- curl -v --max-time 3 "http://muffin-currency.default:8083/rate?
from=PLAIN&to=CHOKOLATE"

# Результат:
curl: (56) Recv failure: Connection reset by peer
```

Проверка AuthorizationPolicy

```
kubectl run test-wallet --image=curlimages/curl --rm -it --restart=Never
\
  --overrides='{"spec":{"serviceAccountName":"muffin-wallet"}}' \
  -- curl -s "http://muffin-currency:8083/rate?from=PLAIN&to=CHOKOLATE"
```

```
# Результат
{"from":"PLAIN","to":"CHOKOLATE","rate":0.013}

# Удаляем лишний namespace
kubectl delete namespace no-istio
```

Попробуем обратиться к muffin-currency без serviceAccount

```
kubectl run test-default --image=curlimages/curl --rm -it --
restart=Never \
  -- curl -s "http://muffin-currency:8083/rate?from=PLAIN&to=CHOKOLATE"

# Результат
RBAC: access denied
```

Resilience (Circuit Breaker, Retry, Timeout)

Что настроено

Компонент	Описание
Circuit Breaker	Выбрасывает нездоровые инстансы после 5 ошибок 5xx
Retry	3 повторные попытки при сбоях (5xx, reset, connect-failure)
Timeout	Максимум 5 секунд на запрос к muffin-currency
Connection Pool	Лимиты: 100 соединений, 1000 активных запросов

Проверить что ресурсы созданы

```
kubectl get destinationrule

# Результат
NAME                 HOST              AGE
muffin-currency-dr   muffin-currency   2s
```

```
kubectl get virtualservice

# Результат
NAME                 GATEWAYS                    HOSTS
AGE
muffin-currency-vs                               ["muffin-currency"]
24s
muffin-wallet-vs     ["muffin-wallet-gateway"]   ["wallet.example.com"]
44h
```

Circuit Breaker

Просмотр статистики

```
kubectl exec deploy/muffin-wallet -c istio-proxy -- \
  pilot-agent request GET stats | grep "muffin-currency" | grep -E "
(ejections|outlier)"
```

Retry

Просмотр статистики

```
kubectl exec deploy/muffin-wallet -c istio-proxy -- \
  pilot-agent request GET stats | grep -E "retry|upstream_rq"
```

Timeout

Просмотр статистики

```
kubectl exec deploy/muffin-wallet -c istio-proxy -- \
  pilot-agent request GET stats | grep "timeout"
```

Все значения лежат в values.yaml

```
istioResilience:
  enabled: true
  muffinCurrency:
    timeout: 5s
    retries:
      attempts: 3
      perTryTimeout: 2s              # timeout на каждую попытку
      retryOn: "5xx,reset,connect-failure,retriable-4xx"
    connectionPool:
      maxConnections: 100
      http1MaxPendingRequests: 100   # очередь запросов
      http2MaxRequests: 1000         # макс активных запросов
      maxRetries: 3
    circuitBreaker:
      consecutive5xxErrors: 5        # ошибок до выброса
      interval: 10s                  # интервал анализа
      baseEjectionTime: 30s          # время выброса
      maxEjectionPercent: 50         # макс % выброшенных
```

Gateway, VirtualService, ServiceEntry

Gateway

Написан в файле `istio-gateway.yaml`

```
kubectl get gateway

# Результат
NAME                     AGE
muffin-wallet-gateway    45h
```

VirtualService

Внешняя маршрутизация

Написан в файле `istio-gateway.yaml`. Позволяет использовать хост `wallet.example.com`

То есть благодаря нему можно делать так:

```
curl http://wallet.example.com/actuator/health

# Результат
{"status":"UP",...
```

Внутренняя маршрутизация (между сервисами)

Написан в файле `istio-resilience.yaml`. Управляет маршрутизацией из `muffin-wallet` в `muffin-currency`, работой Retry, Timeout

Проверка:

```
kubectl run test-wallet --image=curlimages/curl --rm -it --restart=Never
\
  --overrides='{"spec":{"serviceAccountName":"muffin-wallet"}}' \
  -- curl -s "http://muffin-currency:8083/rate?from=PLAIN&to=CHOKOLATE"

# Результат
{"from":"PLAIN","to":"CHOKOLATE","rate":0.013}
```

3. ServiceEntry — взаимодействие с внешней БД

Написан в файле `istio-serviceentry.yaml`.

Насколько понимаю благодаря ServiceEntry Postgres теперь явно зарегистрирован в Service Mesh, то есть Istio полностью отслеживает весь трафик. + Гарантированно виден в Kiali как `postgres-external`.

```
kubectl get serviceentry

# Результат
NAME                HOSTS                                        LOCATION
RESOLUTION   AGE
postgres-external   ["postgres.default.svc.cluster.local"]
MESH_INTERNAL   DNS          36m
```

И весь трафик можно ещё отследить так:

```
kubectl logs -l app=muffin-wallet -c istio-proxy --tail=20 | grep
postgres
```

Или через Kiali:

СКРИН