Развертывание iQChannels в режиме кластера

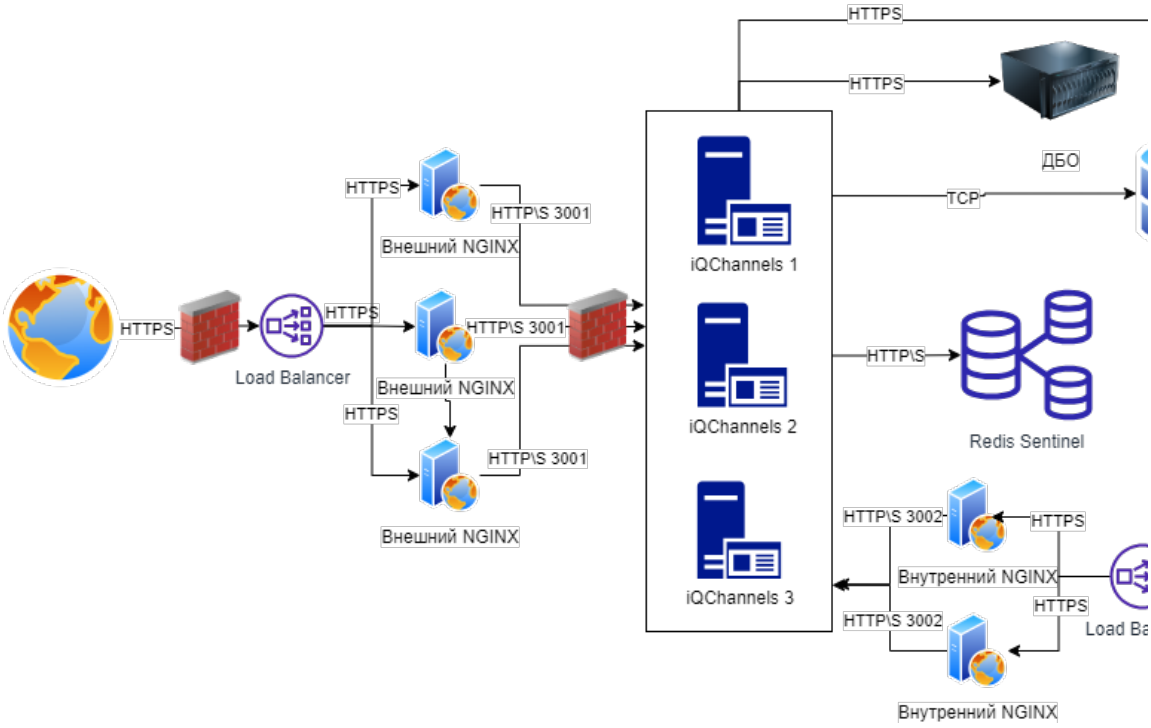
IQ4

iQChannels поддерживает работу в режиме кластера для обеспечения отказоустойчивости и горизонтальной масштабируемости сервера приложений.

Особенности кластера:

- В кластере может быть несколько нод с серверным приложением iQChannels (рекомендуется до 3–5).
- Все ноды в кластере равноправны, нет Single Point of Failure.
- База данных iQChannels работает в режиме standby.
- Балансировка входящих подключений между нодами, как со стороны клиентов, так и со стороны операторов, происходит с помощью Nginx или любого другого прокси-сервера.
- Ноды взаимодействуют между собой через Redis Sentinel.
- В случае выхода из строя одной или нескольких нод Nginx автоматически распределяет подключения между работающими нодами.
- Конфигурация нод должна быть одинаковой, в конфигурации каждой ноды должны быть указаны адреса всех нод кластера.

Пример развертывания кластера:



Конфигурации оборудования:

https://wiki.iqstore.ru/pages/viewpage.action?pageId=2950032

Запуск iQChannels в режиме кластера:

1. Создать файл docker-compose.yml, в котором запускаются docker-образ iQChannels и Redis с поддержкой кластеризации. Пример такого файла docker-compose (2).yaml

2. В docker-compose прописать через volumes путь до файла с конфигом и папки, где будут хранится логи.

```
volumes:
      - /     /config.yaml:/iqchannels/config/config.yaml
      - /   :/iqchannels/var/log/
```

3. Контейнеры будут брать за основу config.yaml на вашем сервер из пункта 2. В config.yaml есть отдельный раздел - Clusterization. Для включения режима кластера пропишите:

```
cluster_mode:                     # cluster mode is disabled by default
  enabled: true                   # to enable clusterization you also need to provide redis
settings bellow
```

Для работы в кластерном режиме iQChannels использует Redis. Redis обязателен для работы кластеров. Redis может работать в трех режимах: single,cluster,sentinel. Примеры для настройки всех 3 режимов:

Single:

```
redis:
  mode: single # single, sentinel or cluster
  network: tcp # The network type, either tcp or unix. Default is tcp.
  addr: localhost:49153 # host:port address. Addr string for single or sentinel instance
  db: 1 # Database to be selected after connecting to the server.
  user: default # user from redis instance
  password: redispw # pass
  cluster_addrs: # list of cluster nodes in format host:port, only for cluster mode
  max_retries: 3 # // Maximum number of retries before giving up. Default is 3 retries; -1
(not 0) disables retries.
  dial_timeout: 5 # default 5 seconds
  min_idle_conns: 1 # default 0
  max_idle_conns: 10 # default 0
  min_retry_backoff: 8 #Minimum backoff between each retry in milliseconds. Default is 8
milliseconds; -1 disables backoff.
  max_retry_backoff: 512 #Maximum backoff between each retry. Default is 512 milliseconds;
-1 disables backoff.
  pool_fifo: true # Type of connection pool. true for FIFO pool, false for LIFO pool. Note
that FIFO has slightly higher
                  #overhead compared to LIFO,but it helps closing idle connections faster
reducing the pool size.
  pool_size: 10 # Maximum number of socket connections. Default is 10 connections per every
available CPU as reported by runtime.GOMAXPROCS.
```

Sentinel:

```
redis:
  mode: sentinel
  master_name: master # The master name.
  sentinel_addrs: # A seed list of host:port addresses of sentinel nodes.
    -
    -
  client_name: client # ClientName will execute the `CLIENT SETNAME ClientName` command for
each conn.
  sentinel_username: sentinel # // If specified with SentinelPassword, enables ACL-based
authentication (via AUTH <user> <pass>)
  sentinel_password: pass # // Sentinel password from "requirepass <password>" (if enabled)
in Sentinel configuration, or, if SentinelUsername is also supplied, used for ACL-based
authentication.
  route_randomly: false # Allows routing read-only commands to the random master or replica
node.
  route_by_latency: true # Allows routing read-only commands to the closest master or
replica node.
  replica_only: false #Route all commands to replica read-only nodes.
  min_retry_backoff: 8 #Minimum backoff between each retry in milliseconds. Default is 8
milliseconds; -1 disables backoff.
  max_retry_backoff: 512 #Maximum backoff between each retry. Default is 512 milliseconds;
-1 disables backoff.
  db: 1 # Database to be selected after connecting to the server.
  user: default # user from redis instance
  password: redispw # pass
  max_retries: 3 # // Maximum number of retries before giving up. Default is 3 retries; -1
(not 0) disables retries.
  dial_timeout: 5 # default 5 seconds
  min_idle_conns: 1 # default 0
  max_idle_conns: 10 # default 0
  pool_fifo: true # Type of connection pool. true for FIFO pool, false for LIFO pool. Note
that FIFO has slightly higher
  #                   #overhead compared to LIFO,but it helps closing idle connections
faster reducing the pool size.
  pool_size: 10 # Maximum number of socket connections. Default is 10 connections per every
available CPU as reported by runtime.GOMAXPROCS.
```

Cluster:

```
redis:
  mode: cluster
  network: tcp # The network type, either tcp or unix. Default is tcp.
  addr: localhost:49153 # host:port address
  db: 1 # Database to be selected after connecting to the server.
  user: default # user from redis instance
  password: redispw # pass
  cluster_addrs: # list of cluster nodes in format host:port, only for cluster mode
  # -
  # -
  max_retries: 3 # // Maximum number of retries before giving up. Default is 3 retries; -1
(not 0) disables retries.
  dial_timeout: 5 # default 5 seconds
  min_idle_conns: 1 # default 0
  max_idle_conns: 10 # default 0
  min_retry_backoff: 8 #Minimum backoff between each retry in milliseconds. Default is 8
milliseconds; -1 disables backoff.
  max_retry_backoff: 512 #Maximum backoff between each retry. Default is 512 milliseconds;
-1 disables backoff.
  pool_fifo: true # Type of connection pool. true for FIFO pool, false for LIFO pool. Note
that FIFO has slightly higher
  ##                   #overhead compared to LIFO,but it helps closing idle connections
faster reducing the pool size.
  pool_size: 10 # Maximum number of socket connections. Default is 10 connections per every
available CPU as reported by runtime.GOMAXPROCS.
```

4. Пропишите нужное кол-во сервисов iQChannels и Redis через docker-compose или командную строку docker и запустите.