Eclipse Zenoh router deployment on VM using Docker tool and Docker Compose file

1. Configuration

1.1. VM

- IP address
- Zenoh REST port - default is **8000**
- Zenoh TCP port - default is **7447**
- Docker container name - **zenoh_zenoh_1**

1.2. Docker Compose

1.2.1. Docker commands

container build up

```
docker-compose -f docker-compose.yml up --build
```

stop container

```
docker stop zenoh_zenoh_1
```

start container

```
docker start zenoh_zenoh_1
```

show container logs

```
docker logs zenoh_zenoh_1
```

1.2.2. Files required

- **docker-compose.yml**
- **entrypoint.sh**
- **router-config.json5**

Files need to include your own configuration, for example: path to router config file in docker compose .yml file.

docker-compose.yml

```
version: "3.3"
services:
  zenoh:
    image: eclipse/zenoh
    restart: always
    ports:
      - 8447:7447 # remapped Zenoh default ports
      - 8000:8000
    volumes:
      - /home/docker/zenoh/data:/root/.zenoh
      - /your/path/to/router-config.json5:/etc/zenoh/router-config.json5
      - /your/path/to/entrypoint.sh:/entrypoint.sh:ro
    environment:
      - RUST_LOG=debug
    entrypoint: /entrypoint.sh # Specify entrypoint file
```

entrypoint.sh

```
#!/bin/ash
echo " * Starting: /zenohd --config /etc/zenoh/router-config.json5 $*"
exec /zenohd --config /etc/zenoh/router-config.json5 $*
```

config-router.json5

```
{
  metadata: {
    name: "your name",
    location: "your location"
  },
  mode: "router",
  connect: {
    endpoints: []
  },
  listen: {
    endpoints: ["tcp/[::]:7447"]
  },
  scouting: {
    timeout: null,
    delay: null,
    multicast: {
      enabled: true,
      address: null,
      interface: null,
      autoconnect: null,
      listen: null
    },
    gossip: {
      enabled: null,
      multihop: null,
      autoconnect: null
    }
  },
  timestamping: {
    enabled: null,
    drop_future_timestamp: null
  },
  queries_default_timeout: null,
  routing: {
    router: {
      peers_failover_brokering: null
    },
    peer: {
```

```
    mode: null
  }
},
aggregation: {
  subscribers: [],
  publishers: []
},
transport: {
  unicast: {
    accept_timeout: 10000,
    accept_pending: 100,
    max_sessions: 1000,
    max_links: 1,
    lowlatency: false
  },
  multicast: {
    join_interval: 2500,
    max_sessions: 1000
  },
  qos: {
    enabled: true
  },
  link: {
    protocols: null,
    tx: {
      sequence_number_resolution: "32bit",
      lease: 10000,
      keep_alive: 4,
      batch_size: 65535,
      queue: {
        size: {
          control: 1,
          real_time: 1,
          interactive_high: 1,
          interactive_low: 1,
          data_high: 2,
          data: 4,
          data_low: 2,
          background: 1
        },
        backoff: 100
      },
      threads: 1
    },
    rx: {
      buffer_size: 65535,
      max_message_size: 1073741824
    },
    tls: {
      root_ca_certificate: null,
      server_private_key: null,
      server_certificate: null,
      client_auth: null,
      client_private_key: null,
      client_certificate: null,
      server_name_verification: null
    },
    unixpipe: {
      file_access_mask: null
    },
    compression: {
      enabled: false
    }
  },
  shared_memory: {
    enabled: false
  },
  auth: {
    usrpwd: {
      user: null,
      password: null,
```

```
        dictionary_file: null
      },
      pubkey: {
        public_key_pem: null,
        private_key_pem: null,
        public_key_file: null,
        private_key_file: null,
        key_size: null,
        known_keys_file: null
      }
    }
  },
  adminspace: {
    permissions: {
      read: true,
      write: false
    }
  },
  "plugins_search_dirs": [],
  "plugins": {
    "rest": {
      "__required__": true,
      "http_port": "8000"
  },
  "storage_manager": {
     "storages": {
        "demo": {
          "key_expr": "demo/example/**",
          "volume": "memory"
        }
      }
    }
  }
}
```

2. Interaction using HTTP

Zenoh router information

```
curl http://<vm_ip>:<zenoh_rest_port>/@/router/local
```

Information about router storage

```
curl "http://<vm_ip>:<zenoh_rest_port>/@/router/local/status/plugins/storage_manager/storages/*"
```

GET key/value

```
$ curl http://<vm_ip>:<zenoh_rest_port>/demo/example/test-hello

[
{ "key": "demo/example/test-hello", "value": "Hello World!", "encoding": "text/plain", "time": "2024-01-
18T12:35:37.781402476Z/678ef664139c1214c3ba3844b5542b08" }
]
```

Delete key/value

```
curl -X DELETE http://<vm_ip>:<zenoh_rest_port>/demo/example/test
```

3. Interaction using Zenoh-python

TCP get

```
$ python3 zenoh-python/z_get.py -e tcp/<vm_ip>:<zenoh_tcp_port> -s demo/example/**

Opening session...
Sending Query 'demo/example/**'...
>> Received ('demo/example/test-hello': 'Hello World!')
```

z_get.py

```
#
# Copyright (c) 2022 ZettaScale Technology
#
# This program and the accompanying materials are made available under the
# terms of the Eclipse Public License 2.0 which is available at
# http://www.eclipse.org/legal/epl-2.0, or the Apache License, Version 2.0
# which is available at https://www.apache.org/licenses/LICENSE-2.0.
#
# SPDX-License-Identifier: EPL-2.0 OR Apache-2.0
#
# Contributors:
#   ZettaScale Zenoh Team, <zenoh@zettascale.tech>
#

import sys
import time
import argparse
import json
import zenoh
from zenoh import config, QueryTarget

# --- Command line argument parsing --- --- --- --- --- ---
parser = argparse.ArgumentParser(
    prog='z_get',
    description='zenoh get example')
parser.add_argument('--mode', '-m', dest='mode',
                    choices=['peer', 'client'],
                    type=str,
                    help='The zenoh session mode.')
parser.add_argument('--connect', '-e', dest='connect',
                    metavar='ENDPOINT',
                    action='append',
                    type=str,
                    help='Endpoints to connect to.')
parser.add_argument('--listen', '-l', dest='listen',
                    metavar='ENDPOINT',
                    action='append',
                    type=str,
                    help='Endpoints to listen on.')
parser.add_argument('--selector', '-s', dest='selector',
                    default='demo/example/**',
                    type=str,
                    help='The selection of resources to query.')
parser.add_argument('--target', '-t', dest='target',
                    choices=['ALL', 'BEST_MATCHING', 'ALL_COMPLETE', 'NONE'],
                    default='BEST_MATCHING',
                    type=str,
                    help='The target queryables of the query.')
parser.add_argument('--value', '-v', dest='value',
                    type=str,
                    help='An optional value to send in the query.')
parser.add_argument('--config', '-c', dest='config',
                    metavar='FILE',
                    type=str,
                    help='A configuration file.')
```

```
args = parser.parse_args()
conf = zenoh.Config.from_file(
    args.config) if args.config is not None else zenoh.Config()
if args.mode is not None:
    conf.insert_json5(zenoh.config.MODE_KEY, json.dumps(args.mode))
if args.connect is not None:
    conf.insert_json5(zenoh.config.CONNECT_KEY, json.dumps(args.connect))
if args.listen is not None:
    conf.insert_json5(zenoh.config.LISTEN_KEY, json.dumps(args.listen))
selector = args.selector
target = {
    'ALL': QueryTarget.ALL(),
    'BEST_MATCHING': QueryTarget.BEST_MATCHING(),
    'ALL_COMPLETE': QueryTarget.ALL_COMPLETE(),
}.get(args.target)

# Zenoh code  --- --- --- --- --- --- --- --- --- --- ---

# initiate logging
zenoh.init_logger()

print("Opening session...")
session = zenoh.open(conf)

print("Sending Query '{}'...".format(selector))
replies = session.get(selector, zenoh.Queue(), target=target, value=args.value, consolidation=zenoh.
QueryConsolidation.NONE())
for reply in replies.receiver:
    try:
        print(">> Received ('{}': '{}')"
              .format(reply.ok.key_expr, reply.ok.payload.decode("utf-8")))
    except:
        print(">> Received (ERROR: '{}')"
              .format(reply.err.payload.decode("utf-8")))


session.close()
```