Real-Time Healthcare IoT Network Monitoring (Software-only)

Goal: Fully self-contained dev/project scaffold you can run locally (Docker) to simulate IoT healthcare sensors, SNMP devices, Zabbix monitoring + alerts, and Grafana visualization — *no hardware required*.

What this delivers

- `docker-compose.yml` to run:
- MySQL database for Zabbix
- Zabbix server
- Zabbix web UI (nginx + PHP)
- Grafana (with Zabbix plugin pre-installed)
- SNMP simulator (snmpsim) to simulate network devices
- Python sensor simulator `sensor_simulator.py` that:
- Simulates patient vitals (heart rate, temperature, SPO2)
- Sends metrics to Zabbix using `pyzabbix` (Zabbix Sender protocol)
- `zabbix_auto_config.py`: script that uses the Zabbix API to create a host, items (trapper), and triggers (alerts) automatically so you get graphs/alerts without manual web UI steps.
- `requirements.txt` for Python dependencies.
- README with run instructions and notes for Grafana dashboard import and Zabbix credentials.

How to use (summary)

1. Install Docker & Docker Compose.
2. Clone this project files into a folder.
3. `docker compose up -d` (or `docker-compose up -d`) to start services.
4. Wait ~1–2 minutes for Zabbix to initialize (DB migrations) and Grafana to be ready.
5. Run `python3 zabbix_auto_config.py` (edit credentials at top) to create host/items/triggers.
6. Start `python3 sensor_simulator.py` to push simulated patient telemetry to Zabbix every 5s.
7. Open Zabbix frontend at `http://localhost:8080` (default login: `Admin` / `zabbix`) and Grafana at `http://localhost:3000` (default `admin` / `admin`).

Files (included below)

1. `docker-compose.yml`
2. `sensor_simulator.py`
3. `zabbix_auto_config.py`
4. `requirements.txt`
5. `README.md` (this doc)

`docker-compose.yml`

```
version: '3.7'
services:
  mysql:
    image: mysql:8.0
    container_name: zbx_mysql
    environment:
      MYSQL_ROOT_PASSWORD: zabbix
      MYSQL_DATABASE: zabbix
      MYSQL_USER: zabbix
      MYSQL_PASSWORD: zabbix
    volumes:
      - mysql-data:/var/lib/mysql
    ports:
      - "3306:3306"

  zabbix-server:
    image: zabbix/zabbix-server-mysql:6.4-latest
    container_name: zabbix-server
    depends_on:
      - mysql
    environment:
      DB_SERVER_HOST: mysql
      MYSQL_DATABASE: zabbix
      MYSQL_USER: zabbix
      MYSQL_PASSWORD: zabbix
    ports:
      - "10051:10051"

  zabbix-web:
    image: zabbix/zabbix-web-nginx-mysql:6.4-latest
    container_name: zabbix-web
    depends_on:
      - zabbix-server
      - mysql
    environment:
      DB_SERVER_HOST: mysql
      MYSQL_DATABASE: zabbix
      MYSQL_USER: zabbix
      MYSQL_PASSWORD: zabbix
      ZBX_SERVER_HOST: zabbix-server
    ports:
      - "8080:8080"

  grafana:
    image: grafana/grafana:10.2.0
    container_name: grafana
    environment:
      - GF_INSTALL_PLUGINS=alexanderzobnin-zabbix-datasource
```

```
      - GF_SECURITY_ADMIN_PASSWORD=admin
    volumes:
      - grafana-data:/var/lib/grafana
    ports:
      - "3000:3000"

  snmpsim:
    image: snmpsim/snmpsim
    container_name: snmpsim
    command: ["snmpsimd.py", "--data-dir=/data", "--agent-udpv4-
endpoint=0.0.0.0:1161"]
    volumes:
      - ./snmpsim-data:/data
    ports:
      - "1161:1161/udp"

volumes:
  mysql-data:
  grafana-data:
```

Notes: this compose uses official Zabbix images (server & web). Zabbix server listens on port 10051 for active checks and Zabbix sender. Grafana has the Zabbix plugin installed (datasource) allowing you to query data from Zabbix.

`requirements.txt`

```
pyzabbix==1.6.2
requests
```

`sensor_simulator.py`

```
"""Simulate healthcare IoT sensors and send metrics to Zabbix using pyzabbix
ZabbixSender.

This script creates metrics for multiple 'patients' and sends trapper values
to Zabbix server at port 10051.

Configure ZABBIX_SERVER, HOSTNAME below to match the host created by
zabbix_auto_config.py
"""
import time
import random
from pyzabbix import ZabbixMetric, ZabbixSender

# CONFIG
ZABBIX_SERVER = '127.0.0.1'
```

```
ZABBIX_PORT = 10051
HOSTNAME = 'sim-patient-1'  # must match the host created in
zabbix_auto_config.py

sender = ZabbixSender(zabbix_server=ZABBIX_SERVER, zabbix_port=ZABBIX_PORT)

def gen_vitals():
    # realistic-ish ranges
    heart_rate = random.randint(55, 110)            # bpm
    temperature = round(random.uniform(36.0, 39.0), 1)  # Celsius
    spo2 = random.randint(88, 100)                  # %
    return heart_rate, temperature, spo2

if __name__ == '__main__':
    print('Starting sensor simulator. Sending metrics to', ZABBIX_SERVER)
    try:
        while True:
            hr, temp, spo2 = gen_vitals()
            metrics = [
                ZabbixMetric(HOSTNAME, 'vitals.heart_rate', hr),
                ZabbixMetric(HOSTNAME, 'vitals.temperature', temp),
                ZabbixMetric(HOSTNAME, 'vitals.spo2', spo2),
            ]
            result = sender.send(metrics)
            print('Sent:', hr, temp, spo2, '->', result)
            time.sleep(5)  # send every 5 seconds
    except KeyboardInterrupt:
        print('Stopped simulator')
```

`zabbix_auto_config.py`

```
"""Create host, items and triggers in Zabbix via the Zabbix API.

Run after Zabbix server + web are up. Update ZABBIX_API_URL, USER, PASS as
needed.
"""
import time
from pyzabbix import ZabbixAPI

# CONFIG - change if necessary
ZABBIX_API_URL = 'http://127.0.0.1:8080/zabbix'
ZABBIX_USER = 'Admin'
ZABBIX_PASS = 'zabbix'
HOSTNAME = 'sim-patient-1'

zapi = ZabbixAPI(ZABBIX_API_URL)
print('Connecting to Zabbix API...')
zapi.login(ZABBIX_USER, ZABBIX_PASS)
```

```
print('Connected. Zabbix API version', zapi.api_version())

# create host group (if not exists)
group_name = 'Simulated Patients'
try:
    group = zapi.hostgroup.get(filter={'name': group_name})
    if not group:
        group = zapi.hostgroup.create(name=group_name)
        groupid = group['groupids'][0]
    else:
        groupid = group[0]['groupid']
except Exception as e:
    print('Error ensuring hostgroup:', e)
    raise

# create host
interfaces = [{
    'type': 1,  # agent interface (not used for trapper items but required)
    'main': 1,
    'useip': 1,
    'ip': '127.0.0.1',
    'dns': '',
    'port': '10050'
}]

try:
    existing = zapi.host.get(filter={'host': HOSTNAME})
    if existing:
        hostid = existing[0]['hostid']
        print('Host already exists with id', hostid)
    else:
        h = zapi.host.create(
            host=HOSTNAME,
            interfaces=interfaces,
            groups=[{'groupid': groupid}],
            tags=[{'tag': 'sim', 'value': '1'}]
        )
        hostid = h['hostids'][0]
        print('Created host', HOSTNAME, 'id', hostid)
except Exception as e:
    print('Error creating host:', e)
    raise

# Create trapper items (use type=2 for Zabbix trapper)
items = [
    {'key_': 'vitals.heart_rate', 'name': 'Heart Rate', 'value_type': 3},
    {'key_': 'vitals.temperature', 'name': 'Temperature', 'value_type': 0},
    {'key_': 'vitals.spo2', 'name': 'SpO2', 'value_type': 3},
]

for it in items:
```

```
    try:
        existing_items = zapi.item.get(hostids=hostid, filter={'key_':
it['key_']})
        if existing_items:
            print('Item exists:', it['key_'])
            continue
        created = zapi.item.create(
            name=it['name'],
            key_=it['key_'],
            hostid=hostid,
            type=2,  # Zabbix trapper
            value_type=it['value_type'],
            delay='5s'
        )
        print('Created item', it['key_'])
    except Exception as e:
        print('Error creating item', it['key_'], e)

# Create a trigger: high temperature
try:
    trig = zapi.trigger.create(
        description='High temperature for {HOST.NAME}',
        expression=f"{{{HOSTNAME}:vitals.temperature.last()}}>37.5",
        priority=4,
        tags=[{'tag': 'alert', 'value': 'temp'}]
    )
    print('Created trigger')
except Exception as e:
    print('Trigger may already exist or error:', e)

print('Done. You can now start sensor_simulator.py to send values to
Zabbix.')
```

Notes: `zabbix_auto_config.py` uses the pyzabbix `ZabbixAPI`. It creates a host with trapper-type items so the sensor simulator can push values via Zabbix Sender. The trigger expression above is a simple example (temp > 37.5°C). Adjust thresholds to your needs.

SNMP simulator data

- Put SNMP simulation data files (snmpsim data) into `./snmpsim-data`. I included a small README inside `snmpsim-data/` to generate or copy sample MIB datasets. For simple monitoring you can also create an SNMP host in Zabbix pointing to `127.0.0.1` port `1161` and apply standard SNMP templates.

Grafana setup (visualization)

1. Login to Grafana at `http://localhost:3000` (admin/admin).
2. Add a new Data Source -> choose `Zabbix` (datasource plugin installed by docker-compose). Configure API URL: `http://zabbix-web:8080/zabbix` or `http://host.docker.internal:8080/zabbix` depending on your Docker environment. In simple local Docker on Linux you may need to set API URL to `http://host.docker.internal:8080/zabbix` — if that doesn't exist, use `http://<your-host-ip>:8080/zabbix`.
3. Explore dashboards — create a panel using `zabbix` data source and query for the items `vitals.*` created earlier.

I can export a sample Grafana dashboard JSON for you on request.

Alerts

- **Zabbix**: triggers created by `zabbix_auto_config.py` will show under `Monitoring -> Triggers`. You can configure actions/notifications to send email or webhook. Zabbix has built-in action mechanisms.
- **Grafana**: Grafana can also create alert rules based on metrics (if desired).

Troubleshooting & Tips

- If Zabbix web UI shows no data: ensure `sensor_simulator.py` `HOSTNAME` matches the host created by `zabbix_auto_config.py`.
- If Grafana cannot connect to Zabbix API from inside Docker, try using the host IP rather than `localhost` in the Grafana datasource config. On Linux, `host.docker.internal` may not exist; use your machine's IP.
- To send SNMP-based items to Zabbix instead of trapper, create SNMP host in Zabbix (host interface type SNMP) and use SNMP OIDs from your snmpsim dataset.

Next steps (optional enhancements I can add immediately if you want):

- Provide a ready-made Grafana dashboard JSON (panels for HR, Temp, SpO2) you can import.
- Add Dockerized `pyzabbix` script runner so simulator runs in a container and auto-starts.
- Add user-friendly web UI to start/stop simulated patients and adjust their baseline vitals.
- Configure Zabbix actions to send email / webhook (Slack/Telegram) on trigger.

If you want, I can now:

1. Add the Grafana dashboard JSON and a `docker-compose` service to auto-run the Python simulator, or
2. Generate the exact `snmpsim` dataset and example Zabbix SNMP host templates, or

3. Provide a single `setup.sh` that wires everything together and runs the Python scripts in a container.

Tell me which of those you'd like me to add and I'll update the project files here.