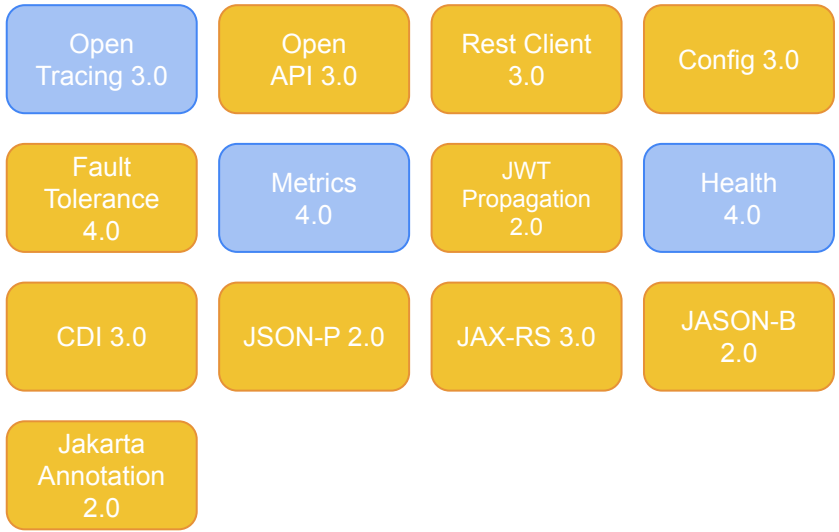
Microservice Observability

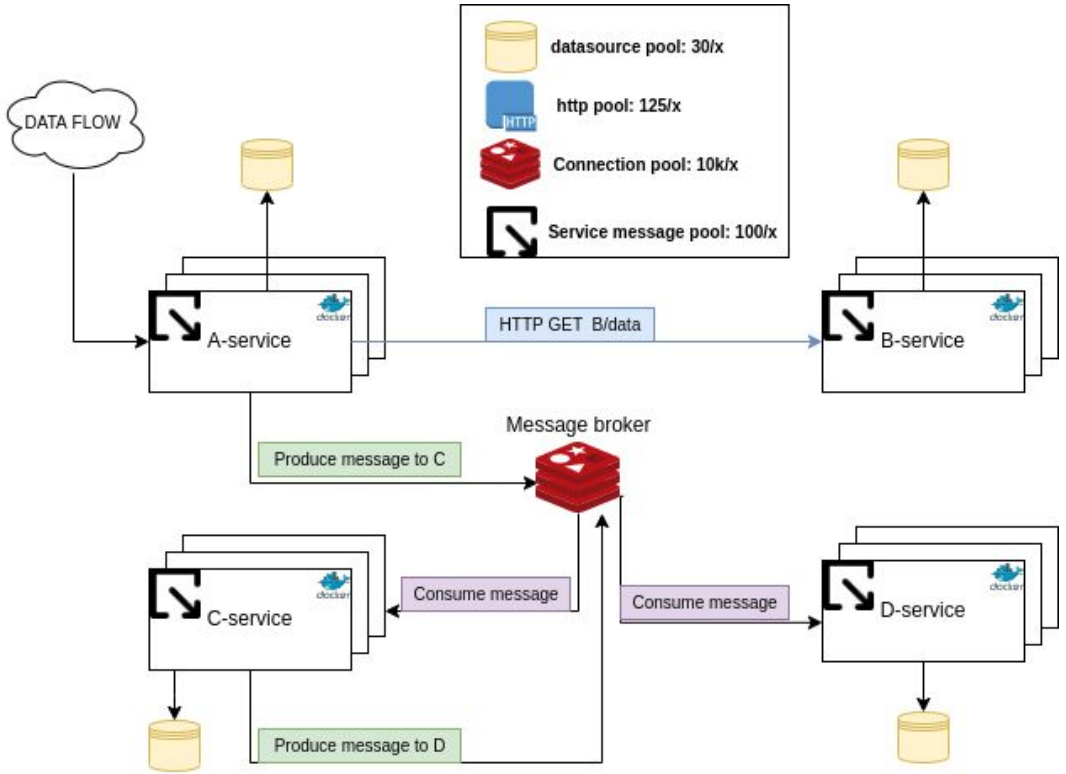
Observability

- Megfigyelhetőség
- Microservice világban dinamikusan változik a szolgáltatások száma, a skálázódás változatos terhelést okoz az erőforrásokra
- Egyes szolgáltatások különböző komplexitással és terheléssel rendelkeznek, erőforrás igényt kell tudni meghatározni
- Nagyon sok log, amiket elemezni kell, egy üzleti folyamat több szolgáltatáson keresztül halad, log aggregáció
- Techek:
  - Microprofile Metrics
  - Microprofile Health
  - Microprofile OpenTracing
    - Jaeger
    - Prometheus
    - Grafana
    - ELK stack
      - Elastic
      - Logstash
      - Kibana



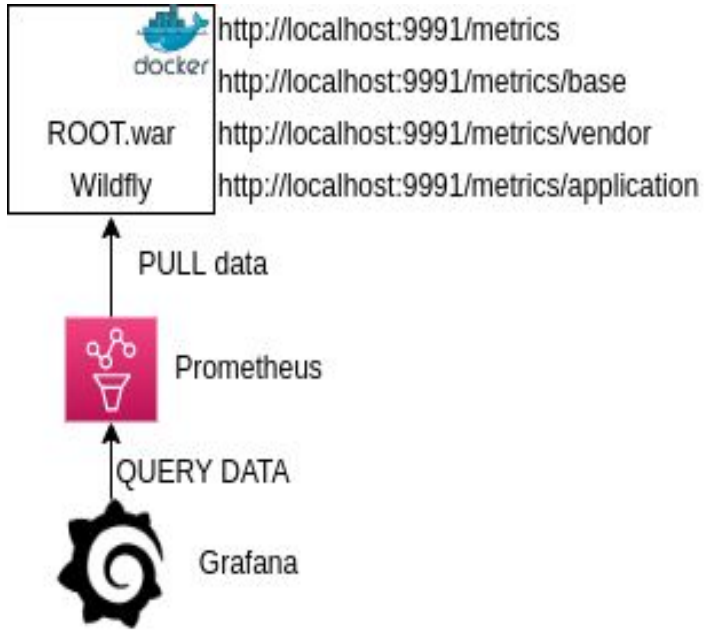
Minta Arch

- Vegyünk A,B,C,D microservicet
  - Tartalmaznak x db consumert
  - Adatbázisból kérnek le adatot
  - Resten egymást hívják
  - Event-et küldenek egymásnak
- Vegyünk néhány tulajdonságot
  - Rest http szálak service példányonként: 125
  - Consumer szám: 50
  - Datasource pool méret: 30
  - Messaging pool: 100
- Skálázzunk!
  - “A” microservice 5000/sec message flow
  - “A” 10 peldany
  - “B” 2 példány
  - “C” 1 példány
  - “D” 1 Példány
- Milyen problémák jöhetnek elő?
  - Mennyi erőforrásra van szükség? (CPU, RAM, poolok)
  - “A” service-t skálazzuk, mit okoz? (B service?)
  - Központi broker? (connection)
  - 1 service 1 db, de 10 service is ugyan az a db instance?
  - Torlódás esetén hogy azonosítjuk hol vagyunk lassúak?



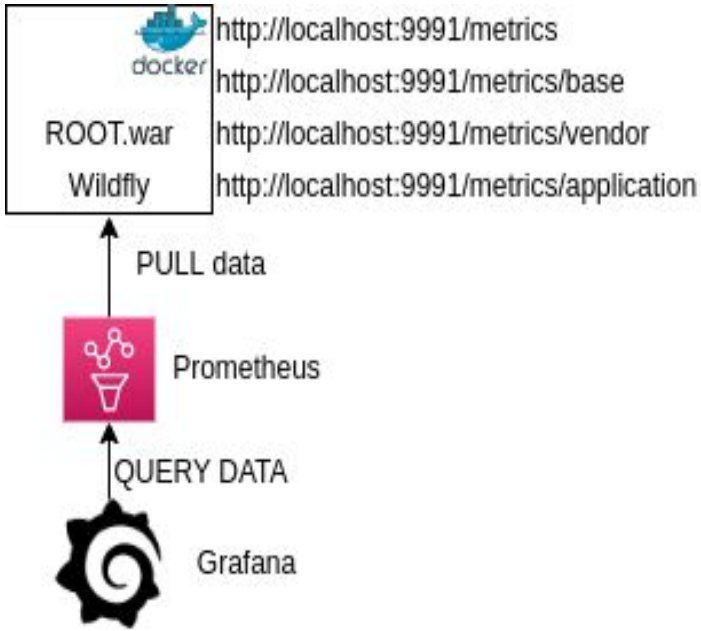
Microprofile Metrics I.

- Metrika szerepe: standard specifikációt határoz amivel monitorozhatóvá válik az alkalmazás performanciája, erőforrás használata.
- Meghatározható hogy egy service miként reagál a skálázásra, és pontosabb erőforrás igényt lehet vele mutatni.. (több példányra van szükség, pool méretek optimalizálása, CPU, RAM használat)
- 3 scope-ot határoz meg az alkalmazás szempontjából ami implementálja
    - **Base**(required) - azok az adatok amiket a service-nek szolgáltatnia kell
        - Cpu adatok (elérhető magok, használt cpu time), jvm adatok (GC time, jvm uptime, memory adatok…)
    - **Vendor** - wildfly által nyújtott adatok, jdbc datasource pool hasznalát, jpa statistics (updatek, lockok), jms statistics
    - **Application** - alkalmazás metrika
        - Messaging (redis, amq, rabbit..) connection használat
        - Rest metrikak, melyik endpoint-ot hányszor hívták
        - Üzleti metrikák (üzleti entitások létrejötte az adatbázisban)



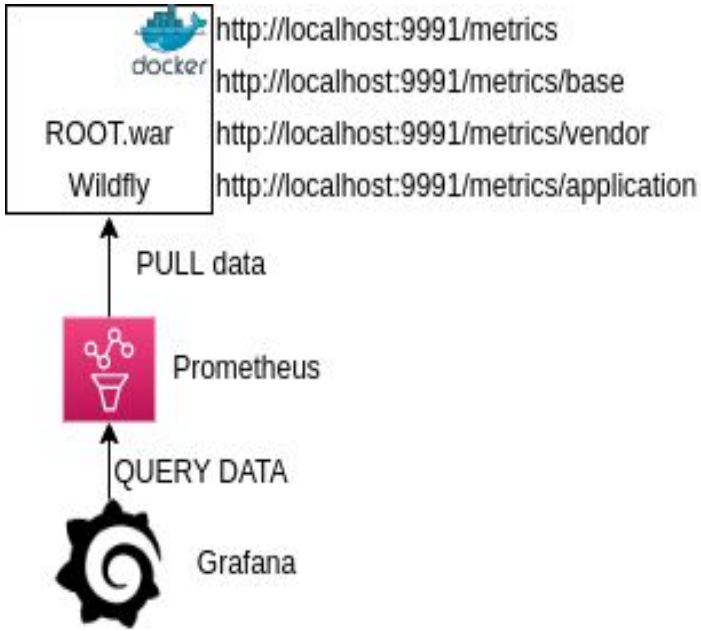
Microprofile Metrics II.

- Integráció
  - Prometheus
    - Open source
    - CNFC - Cloud Native Computing Foundation tag
    - Monitoring eszköz
    - Time series adatbázis
  - Grafana
    - Open source
    - CNFC - Cloud Native Computing Foundation tag
    - Dashboardokat biztosít metrika adatokhoz
    - Táblázatokat, grafikonokat jelenít meg, vizualizáció
    - Adatforrásokat kell megadni ahonnan az adatokat lekérdezi



Microprofile Metrics III.

- Dependency: <artifactId>microprofile-metrics-api</artifactId>
  - org.eclipse.microprofile.metrics.MetricRegistry
- Wildfly metrics
  - standalone.xml config - cli sciptek
  - Management port
- Prometheus config
  - prometheus.yml
    - Target: service elérhetőség - localhost
    - Metrics_path: /metrics metric adatok rest
    - Scrape_interval: milyen gyakran kérjen metrika adatot
- Grafana
  - Provision -> dashboard.yml, datasource.yml
    - Datasouce-ok -> prometheus
    - Dashboard-ok:
      - Json formában adhatók meg
      - Micro service dashboard (jvm, cpu, datasource)



Microprofile Metrics demo

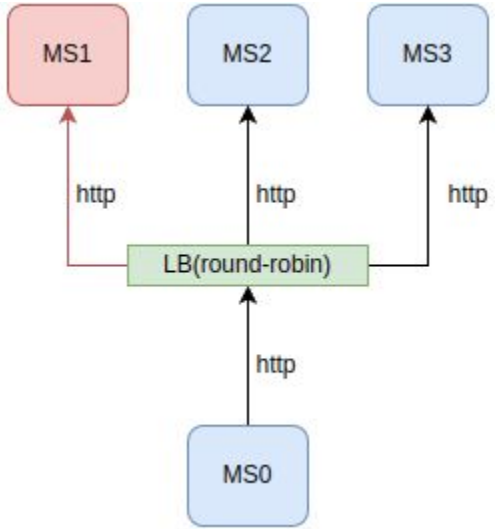
Microprofile Health I.

- Futtató környezet számára biztosít adatokat, az döntéseket tud hozni ennek megfelelően
  - K8S probe-ok építhetnek rá logikákat
    - Forgalomirányítás
    - Service újraindítása
    - Új példány indítása


```
{
  "checks": [
    {
      "data": {
        "etcdURL": "http://hubphq-vdr-docker-s001.icellmobilsoft.hu:2379",
        "nodeName": "44abcbe28a36"
      },
      "name": "etcd",
      "status": "UP"
    },
    {
      "data": {
        "URL": "hubphq-vdr-docker-s001.icellmobilsoft.hu:6379/0",
        "nodeName": "44abcbe28a36"
      },
      "name": "redis.auth",
      "status": "UP"
    },
    {
      "data": {
        "nodeName": "44abcbe28a36",
        "oracleURL": "jdbc:oracle:thin:@//oradev06.icellmobilsoft.hu:1521/icsandpdb"
      },
      "name": "oracle",
      "status": "UP"
    }
  ],
  "status": "UP"
}
```

Microprofile Health II.

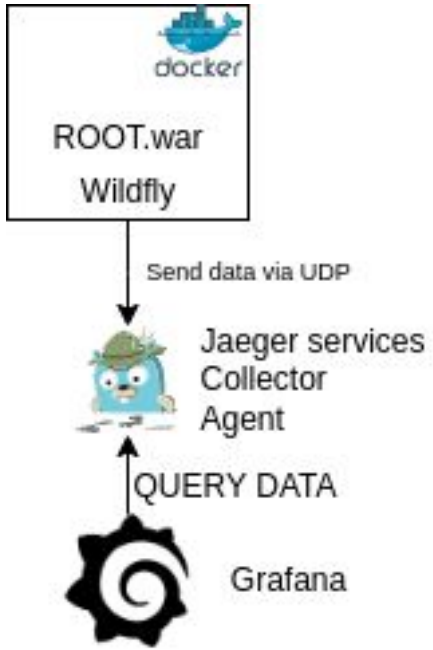
- **/health/live**
  - Wildfly Up értéket ad vissza ha fut a szerver
  - Kubernetes probe fel tudja használni, hogy ellenőrizze él a szolgáltatás
- **health/startup**
  - Deployment állapota
- **/health/ready**
  - Lényegében meg kell tudnia mondania hogy a service képes forgalmat fogadni
  - Erőforrás pingek
    - Redis példányok elérhetősége
    - Message broker elérhetősége
    - Adatbázis elérhetőség
    - Etcd elérhetőség
  - Erőforrások terheltsége
    - Pool 90% felett
  - K8s probe eldöntheti irányít felé forgalmat vagy sem



Microprofile Health demo

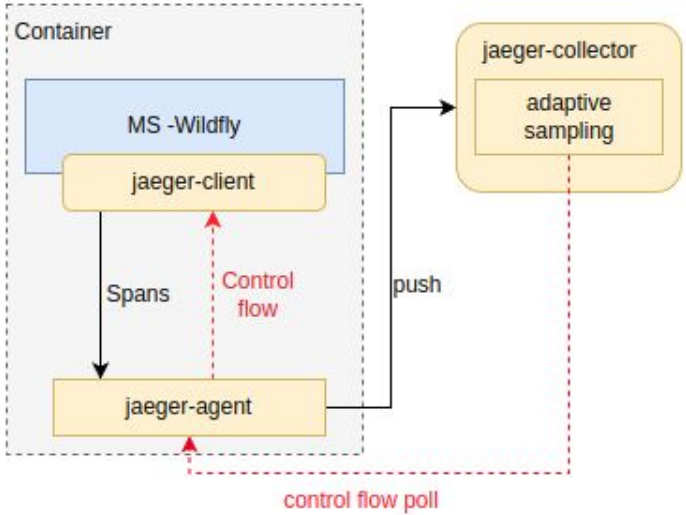
Microprofile OpenTracing I. Jaeger

- Szerepe hogy nyomon követhető legyen a microservicek és egyéb komponensek közötti kommunikáció
  - Rest hívások
  - Messaging
  - Metódus szintű trace
- Van saját felülete Jaeger UI
- Grafana integráció
  - Datasource 8-as grafanától már lehet Jaeger datasource
  - Complex Dashboard készíthető vele, egy dashboardon metrika és trace adat
  - Teljesítménymérés, rendszer kapcsolatok felderítése, kereshetünk vele lassú tranzakciokat



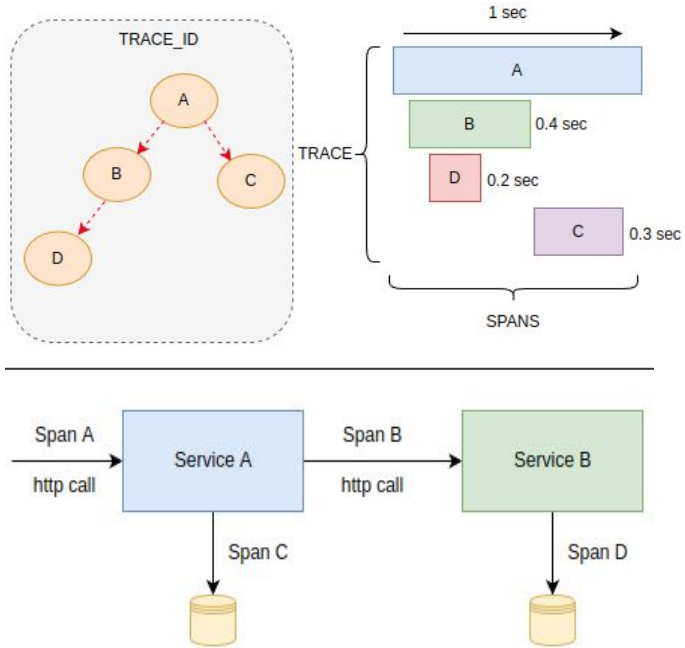
Microprofile OpenTracing II. Jaeger

- Elosztott nyomkövetést tesz lehetővé
- CNFC része 2017 óta (Cloud Native Computing Foundation)
- Open source
- Mintavételezési konfiguráció szerint működik
- Adaptive sampling (centrális paraméterezhetőség)
    - Nagy tranzakciós számú rendszernél, nem mindig van lehetőség 100%-os rate beállításra, mert már csak a trace adat szerzés lassítana mindent
    - Az adaptive sampling lehetőséget ad hogy ne az alkalmazás oldalon keljen konfigurálni a mintavétel értékét, hanem a jaeger agenten keresztül
    - Különböző műveletek mintavételezési értéke megadható
    - Ha szeretnénk változtatni a mintavételezésen, akkor nem kell az alkalmazáshoz nyúlni



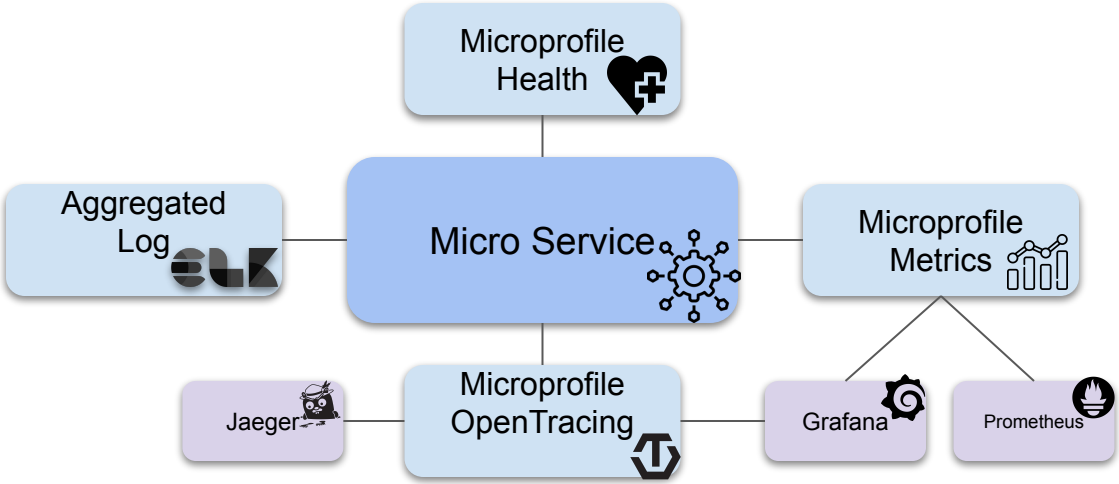
Microprofile OpenTracing III. Jaeger

- Span: Munka egységet jelent
  - name, start time, duration
- Trace: Span-ek összessége, egy rendszerszintű tranzakciót reprezentál
- {trace-id}:{span-id}:{parent-span-id}:{flags}
- Uber-trace-id:[a19991e9f059d2f6:0707ea8f50bda30b:a19991e9f059d2f6:1]]
- “A” span -> Rest hívás indul “A” szolgáltatáson
  - B -> Rest hívás “B” szolgáltatásban
    - D -> “B” szolgaltatas adatbázis művelet
  - C -> “A” szolgáltatás adatbázis művelete



Microprofile OpenTracing demo

Összegzés


Microprofile
Health
Aggregated
Log
Micro Service
Microprofile
Metrics
Jaeger
Microprofile
OpenTracing
Grafana
Prometheus