Fhir On Pi - 3

Mislukkingen

In mijn vrije tijd rommel ik al een paar jaar met dit soort “credit card computers”. Ik monitor ons stroomgebruik ermee (Domoticz op een Pi) en heb ik een display wat de tijdstippen van hoog- en laagwater op het strand aangeeft (Arduino). Ik heb een afstandsmeter gebouwd die de waterstand in de regenwaterput meet (Arduino), een per-Led-programmeerbare LED kleurenstrip en zo nog wat vrij nutteloze maar leuke dingen. Beroepsmatig heb ik al jaren te maken met HL7v3 als uitwisselingsprotocol voor informatie in de zorg en de laatste jaren is daar HL7 FHIR (uitspreken als "fire", vuur) bijgekomen.

Het leek me leuk om te kijken of het me zou lukken om een Raspberry PI als "FHIR-server" in te richten. Gewoon om te kijken of dat kan. Ik bekeek eerst de documentatie bij de HAPI Fhir-server, maar ik vond die wel erg ingewikkeld. Een artikel van Paul Bastide die de IBM Fhir-server op een Raspberry 4 had draaien leek veel simpeler. Dus de IBM Fhir-server was mijn eerste poging (en mijn 2e, 3e en 4e ook).

IBM Fhir server

Paul Bastide is een senior software engineer bij IBM die hetzelfde wilde als ik (zie deze link[1]). In mijn eerste optimisme dacht ik gewoon zijn instructies te kunnen volgen en daarmee binnen een paar uur een Raspberry FHIR-server te hebben.

```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-jar-plugin:3.1.2:jar (default-jar) on project fhir-persistence-schema: Error assembling JAR: Problem creating jar: Geen ruimte meer over op apparaat
```

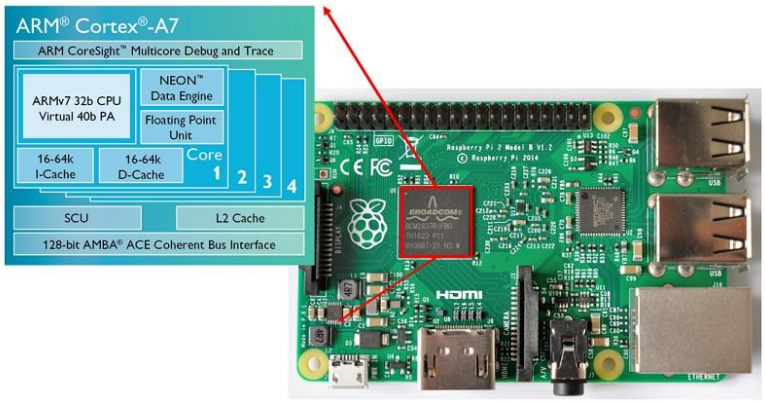
Dat viel erg tegen. Eén van de eerste problemen was het gebrek aan ruimte op de SD-kaart (poging 1, zie afbeelding). Het koppelen van een USB harddisk en verplaatsen van het project naar die disk lukte ook niet echt (poging 2). Maar een volgend probleem was dat de IBM Fhir server niet wilde opstarten (pogingen 3 en 4). Dit ondanks de aanpassingen die her-en-der in Github stonden (aanpassen van de maximale tijd voor het opstarten van de Derby database, verminderen van de vereiste geheugencapaciteit in jvm.options) . Na een keer of vier alles opschonen en opnieuw bouwen en proberen te starten (met voortdurend meldingen als “Server fhir-server start failed. Check server logs for details”) en nalopen van

```
Starting server fhir-server.

Server fhir-server start failed. Check server logs for details.
```

[1] https://ibm.github.io/FHIR/blog/raspberrypi4howto

de server-logs zonder énig idee wat ik nog meer moest veranderen besloot ik om een andere weg in te slaan. Die andere weg was nog lastig. Er worden veel cloud-oplossingen aangeboden (oa Microsoft, Google en Amazon) maar ik wilde nu juist een “eigen” Fhir-server, draaiend op mijn eigen Pi. Weer een andere open-source FHIR-server bleek alleen op Windows te draaien of in Linux-omgevingen op een 386 processor. (Een Raspberry heeft een ARM processor). Zo kwam ik dus toch weer terug bij de HAPI Fhirserver, die in het begin zo ingewikkeld leek (en dat ook was…) .



Aan de slag

Aan de vorige “Fhir on Pi” hebben we een draaiende Raspberry Pi 4 met geïnstalleerde Java JDK en JRE overgehouden. Bovendien kunnen we (tenzij je extra uitdagingen wilt) vanaf de PC inloggen op de Pi.

De volgende stap die we gaan zetten is dat we **Tomcat** gaan installeren en dat niet alleen, in de volgende aflevering installeren we ook de beheerschermen voor Tomcat.

(Apache) Tomcat is een met Java gebouwde HTTP webserver die op verschillende manieren *functionaliteit* via Internet-protocollen beschikbaar kan maken. Je kunt met Tomcat dus webportalen bouwen, applicaties die XML berichten ontvangen en verzenden, systemen maken die een database van buitenaf benaderbaar maken enzovoorts. De HAPI Fhir-server heeft iets als Tomcat nodig omdat de server wel een (in ons geval interne) database meebrengt en de afhandeling van het FHIR-specifieke deel doet, maar voor het generieke werk gebruik maakt van Tomcat.

Zoals wij de Raspberry gaan inrichten is de HAPI Fhir-server een “application” onder Tomcat, op dezelfde manier als waarop het Tomcat-beheer en de Tomcat-documentatie ook “applications” zijn. Er zijn andere manieren om de HAPI Fhir-server te laten draaien en één daarvan zal ik later ook noemen, maar die hebben nadelen.

Spoiler alert: het nadeel van die manier, die ik later zal beschrijven, is dat de database iedere keer opnieuw wordt opgebouwd. De ingevoerde gegevens verdwijnen daarmee bij iedere keer opstarten.Dat nadeel is er niet als we de Fhir-server binnen Tomcat onderbrengen.

Zoals veel opensource producten bestaat Tomcat weer uit verschillende onderdelen, de namen van die onderdelen kom je dan vaak weer tegen in configuratiebestanden, logfiles enzovoorts en dat kan verwarrend zijn.

De Tomcat versie die wij nu installeren bestaat weer uit:

- **Catalina** (een servlet container. Een servlet reageert op aanvragen vanuit het netwerk, zoals een http-request. De container handelt daarvoor een aantal gemeenschappelijke taken af);
- **Coyote** (een HTTP-connector)
- **Jasper** (een JSP-engine, kortom: een technische component die Java Server Pages bedient).

Zoals je bij de meeste opensource produkten zit er totaal geen eenduidigheid in de naamgeving. Catalina is een meisjesnaam en de naam van een eiland bij Los Angeles, een coyote is een prairiewolf en Jasper is een jongensnaam waar de letters JSP in voorkomen. Bij Bassie en Adriaan of Mevrouw Larietand heb ik meer consistentie aangetroffen (resp. B2/B100 en A en Be).

Maar goed, het installeren van Tomcat gaat als volgt:

1. Maak een nieuwe groep “tomcat” aan:

   sudo groupadd tomcat

2. Maak een nieuwe user “tomcat” aan in de groep “tomcat”:

   sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat

3. In de /tmp directory gaan we de Tomcat “tarball” (zip-file) neerzetten:

   cd /tmp

   en dan

   curl -O https://apache.newfountain.nl/tomcat/tomcat-9/v9.0.41/bin/apache-tomcat-9.0.41.tar.gz

4. We gaan de “tarball” uitpakken in een nieuwe directory onder /opt

   mkdir /opt/tomcat

   en dan

   sudo tar xzvf apache-tomcat-*tar.gz -C /opt/tomcat --strip-components=1

5. Nu de rechten goedzetten in de directory waarin we de boel hebben uitgepakt

   cd /opt/tomcat

   daarna:

sudo chgrp -R tomcat /opt/tomcat

daarna:

sudo chmod -R g+r conf

dan:

sudo chmod g+x conf

en tenslotte:

sudo chown -R tomcat webapps/ work/ temp/ logs/

6. **Nano, de editor**. We gaan nu zorgen dat tomcat kan worden opgestart. Hiervoor moeten we een bestand op de Pi gaan editten en dat doen we met “nano”. Als je nog niet gewend bent om daarmee te werken, lees daar dan eerst wat over en doe wat experimenten met bestanden die geen kwaad kunnen. Maar vóórdat we gaan editten moeten we wel weten waar Java staat op de Pi (JAVA_HOME). Dat doen we met:

   sudo update-java-alternatives -l (de laatste letter is een kleine L).

   De Pi antwoordt met iets als:

   java-1.11.0-openjdk-amd64 1081 /usr/lib/jvm/java-1.11.0-openjdk-amd64

   JAVA_HOME is het stuk tekst op het einde. (hier dus: /usr /lib/jvm/java-1.11.0-openjdk-amd64). Als je met Putty op een PC werkt heb je nu echt een voordeel want je kunt de tekst selecteren, in kladblok/notepad plakken en ergens bewaren.

   Nu gaan we editten:

   sudo nano /etc/systemd/system/tomcat.service

   In je scherm krijg je nu iets te zien als:

/etc/systemd/system/tomcat.service

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking

Environment=JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target
```

En hier vervang je de tekst achter “JAVA_HOME=” door jóuw JAVA_HOME. Je slaat de file op met <ctrl> o, dan geef je <return>, dan <ctrl>x om Nano te verlaten.

7. **Systemctl**. Nu gaan we aan de slag met het commando “systemctl”. Als de Pi wordt opgestart en de “kernel” is geladen kijkt het systeem welke extra toepassingen ook nog gestart moeten worden. Die extra toepassingen kunnen per systeem verschillen want bijvoorbeeld op een mailserver draaien andere toepassingen dan op een firewallsysteem. Systemctl is het tool waarmee het opstartsysteem van Linux wordt beheerd en dat gaan we nu gebruiken om tomcat op te starten.
   Eerst zorgen dat systemctl wordt bijgewerkt naar de actuele situatie:

   sudo systemctl daemon-reload

   Dan tomcat starten:

   sudo systemctl start tomcat

8. **Less**. Nu gaan we met systemctl kijken of alles goed gaat. Na het invoeren van de opdracht (hieronder) krijg je één of meer regels te zien in een tooltje dat “less” heet. “less” (en haar tegenpool “more”) gebruik je om op een (karakter-georiënteerde) terminal een bestand scherm-voor-scherm te kunnen bekijken. “less” en “more” zijn dus geen editor (zoals nano dat wel is), want je kunt het bestand niet veranderen, maar je kunt wel bladeren, zoeken enzovoorts. Je verlaat “less” met <ctrl> c of met alleen een “q”.

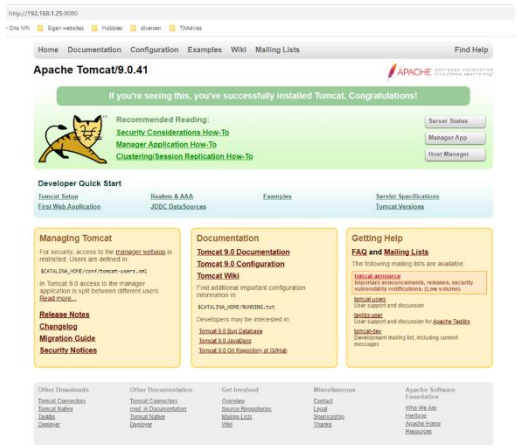
   sudo systemctl status tomcat

Je krijgt nu een paar regels te zien in “less” en als het goed is zie je op een regel met datum+tijdstip met “Tomcat started” en een regel met datum+tijdstip en “Started Apache Tomcat Web Application Container”. Je Tomcat installatie is geslaagd!

```
Dec 15 11:50:35 fhirpi systemd[1]: Starting Apache Tomcat Web Application Container...
Dec 15 11:50:35 fhirpi startup.sh[694]: Existing PID file found during start.
Dec 15 11:50:35 fhirpi startup.sh[694]: Removing/clearing stale PID file.
Dec 15 11:50:35 fhirpi startup.sh[694]: Tomcat started.
Dec 15 11:50:35 fhirpi systemd[1]: Started Apache Tomcat Web Application Container.
lines 1-14/14 (END)
```

Verlaat “less” met een druk op q of <ctrl>c.

9. Je denkt misschien dat het vandaag niet beter kan worden, maar dat is niet zo! Start nu op je PC een webbrowser op en type in de adresbalk het IP-adres van je Raspberry Pi, met direct daarachter :8080 . (We gebruiken dus poort 8080 op de Pi). Het adres van mijn Pi is 192.168.1.25 en ik type dus http://192.168.1.25:8080 in de adresbalk.



Als het goed is zie je nu de tomcat welkomstpagina op je scherm!

Je Pi is een webserver geworden! In de volgende aflevering gaan we de beheerschermen voor Tomcat installeren.