# Vejledning til opsætning af systemet

Vi har lavet en vejledning til at få vores system op og køre lokalt på en ubuntu 14.04 maskine. I produktionsmiljøet kører de fleste af systemerne på hver deres maskine, men her sætter vi det op, så det kører på den samme. Alle værktøjerne vi bruger kører også på Windows og OS X, men vi har valgt at lave en vejledning til ubuntu, da vores servere også kører ubuntu.

Denne guide tager udgangspunkt i en helt nyinstalleret ubuntu 14.04

## Opdater systemet

Sørg for at systemet er opdateret inden nedenstående kommandoer køres ved at køre:

*sudo apt-get upgrade*

*sudo apt-get update*

## Installation af postgres:

*sudo apt-get install postgresql-9.3*

## Installation af Java 8:

*sudo add-apt-repository ppa:webupd8team/java*

*sudo apt-get update*

*sudo apt-get install oracle-java8-installer*

## Installation af git:

*sudo apt-get install git*

## Installation af Maven

*sudo apt-get install maven*

## Opsætning af backend

Kopier eller hent via git vores backend community-ws ind i en mappe og gå ind i mappen vha terminalen.

*mkdir ~/code && cd ~/code*

Git:

*git clone* [*https://github.com/mspe87/community-ws.git*](https://github.com/mspe87/community-ws.git) *&& cd community-ws*

### Opsætning af database:

Skift rolle til bruger postgres:

*sudo su postgres*

åbn postgreSQL:

*psql*

sæt password på postgres brugeren så det stemmer overens med vores config fil skriv:

*\password postgres*

Sæt herefter password til 1qazXSW”

Først vil vi nu køre vores creation script, hvorefter vi vil indsætte data vha. Testdata;

*\i sql/'Create Database.sql'*

*\i sql/Testdata.sql*

Databasen er nu sat op og klar til brug.

### Opsætning af webservice:

Gå ind i mappen community-ws, som din egen bruger (ikke postgres).

*cd ~/code/community-ws*

Skriv:

*mvn install*

Alle dependencies bliver hentet ned og projektet bliver bygget.

For at køre projektet på en jetty server skriv:

*mvn jetty:run*

Projektet skulle gerne køre på port 8080 nu. Test eventuelt ved at gå ind på <http://localhost:8080/api/v1/application.wadl> i din browser. Denne giver en liste over alle kald der kan laves.

Til at skrive vores java kode, har vi brugt eclipse, som kan hentes fra https://eclipse.org/. For at lave projektet om til et eclipse projekt, som direkte kan importeres kan der skrives:

*mvn eclipse:eclipse*

## Opsætning af frontend

Installer Node.js:

*sudo apt-get install nodejs*

Installer Node package manager:

*sudo apt-get install npm*

Kopier frontenden mvSite ind i en mappe. Og gå ind i den mappe via en terminal.

*cd ~/code*

Git:

*git clone* [*https://github.com/mspe87/mvSite.git*](https://github.com/mspe87/mvSite.git) *&& cd mvSite*

For at kunne bygge og køre frontenden skal der installeres nogle node.js pakker og ruby gems. Skriv:

*sudo npm install -g grunt-cli*

*sudo npm install grunt --save-dev*

*sudo ln -s /usr/bin/nodejs /usr/bin/node*

*sudo npm install bower -g*

Installer dependencies fra package.json:

*sudo npm install*

Installer projektets dependencies med bower fra bower.json:

*bower install*

*sudo apt-get install ruby 1.9.3*

*sudo gem install compass*

*sudo apt-get install php5-cgi*

Alle værktøjer til at bygge og køre frontenden er nu installeret. Grunt bruges til dette, hvor der i Gruntfile.js er defineret hvilke tasks der kan køres. Skriv

*grunt build* for at bygge projektet. Html, css, scripts vil blive minificeret. Billeder vil blive optimeret, så de fylder mindre og blive postfixet med tilfældig streng, for at kunne cache i længere tid. Det kompilerede projekt ligger i mappen dist

*grunt serve* for at køre projektet på en node.js server med livereload. Dette har været rigtig smart til at udvikle i, da der bliver injectet et script der livereloader siden hvis der bliver lavet ændringer i en fil. (Sørg for at jetty kører, da backenden bliver brugt). GruntFile.js er sat op så den proxier ligesom vores .htaccess

*grunt serve --allow-remote* samme som grunt serve, men kører på 0.0.0.0:9000 så der kan forbindes til den udefra.

### Apache

Installer apache og php:

*sudo apt-get install apache2*

*sudo apt-get install php5*

*sudo apt-get install libapache2-mod-php5*

*sudo /etc/init.d/apache2 restart*

apache bruger som standard /var/www/html til siden der bliver vist når man går ind på host. Det vi har gjort, er at vi har slettet html mappen:

*cd /var/www*

*sudo rm -r html*

Og lavet et symlink til dist mappen:

*sudo ln -s ~/code/mvSite/dist html*

Vi skal tillade at .htaccess filen kan lave ændringer:

*sudo nano /etc/apache2/apache2.conf*

Under <Directory /var/www> ændres AllowOverride til **All** i stedet for none og gem filen.

Installer moduler:

*sudo a2enmod rewrite proxy\_http headers && sudo service apache2 restart*

Apache og backenden skulle nu køre som ønsket. Hvis <http://localhost> besøges, skulle der gerne blive vist en skærm om at blive logget ind (backend skal køre). Dette er mv-nordics login skærm. Det er kun muligt at logge ind igennem denne, på deres eget domæne, så vi har lavet en ”snyde login”. Gå ind på <http://localhost/api/v1/debug/login/2/redirect> og du vil blive logget ind som bruger med id 2.

## Opsætning af Varnish

*sudo apt-get install curl*

Vi vil køre følgende som root:

*sudo su*

*apt-get install apt-transport-https*

*curl https://repo.varnish-cache.org/ubuntu/GPG-key.txt | apt-key add -*

*echo "deb https://repo.varnish-cache.org/ubuntu/ precise varnish-4.0" >> /etc/apt/sources.list.d/varnish-cache.list*

*apt-get update*

*apt-get install varnish*

Varnish kører nu på port 6081

Skift tilbage til din normale bruger med ctrl+d

*cd ~/code*

Udskift varnish configuration default.vcl:

*git clone https://github.com/mspe87/varnish-conf.git && sudo cp varnish-conf/default.vcl /etc/varnish/default.vcl*

Genstart varnish:

*sudo service varnish restart*

Varnish er nu oppe og køre på port 6081, med apache på port 80 som backend. Test dette ved at går ind på <http://localhost:6081>

Alle systemerne snakker sammen, så sørg for at jetty serveren kører.

For at se headere der bliver sat på, kan følgende køres:

*curl --head* [*http://localhost:6081*](http://localhost:6081)

Her kan der ses om vi rammer cachen ved headeren X-Cache.