Timebox 4 – AWS-Deployment

Oversigt

OpgaveNavn	Implementering af hjemmesiden på AWS-server		
Implementering af krav	N/A		
Udført af	Marc og Jan	**Dato**	11-10-2021
Timebox	4	**Område**	Website

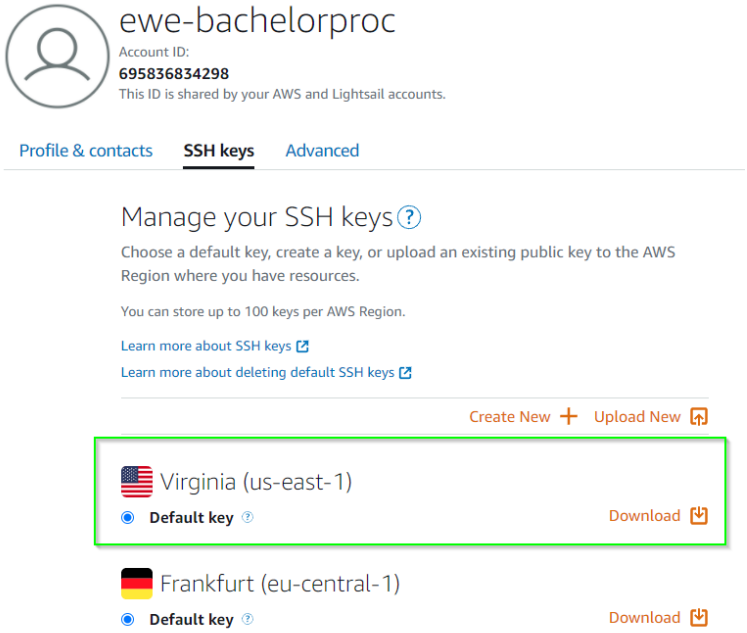
Contents

Introduktion

Dette dokument omhandler hjemmesiden, som skal implementeres på en AWS-server.

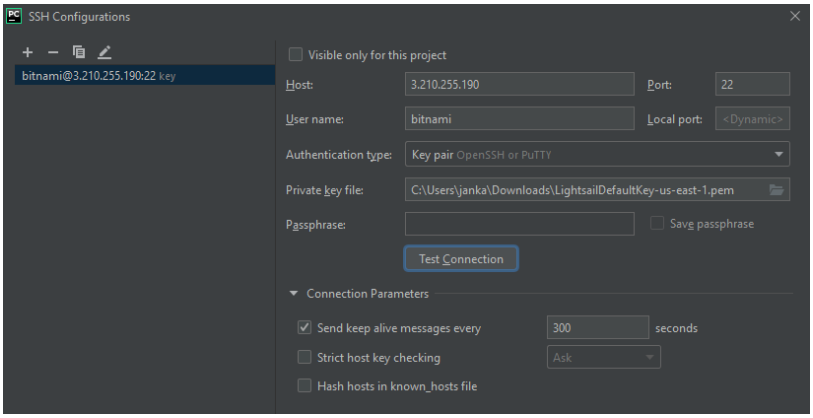
Analyse og Design

For at kunne SSH ind på AWS-serveren, skal der hentes en PEM-fil [1], se Figur 1, som skal bruges gennem Pycharm [2] for at forbinde til AWS-serven.



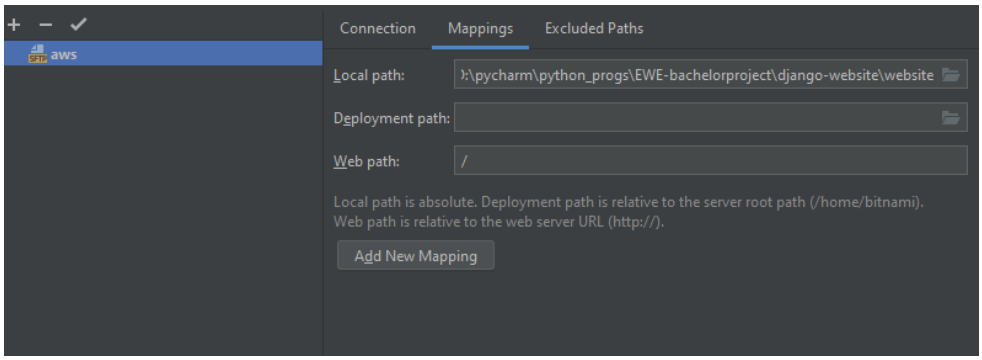
Figur 1 PEM-fil

Det er muligt at ”remote deploy” hjemmesiden gennem Pycharm [3]. opsætning af forbindelsen ses på Figur 2.



Figur 2 Opsætning for forbindelsen

Opsætning af mapping ses på Figur 4.



Figur 3 Opsætning for mapping

For at kunne tilgå hjemmesiden gennem et domænenavn, som fx ”www.eksempel.com”, skal der købes et domænenavn, hvilket bliver gjort igennem godaddy.com [4] DNS (domain name service) udbyderen.

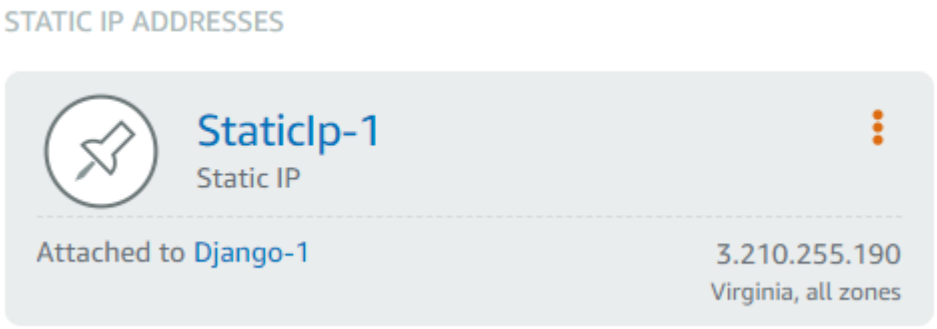
For at bruge det nye domænenavn, vil der skulle sættes NGINX [5] op på AWS-serveren, som forbinder http requests til serveren på domænenavnet, til en Gunicorn website server som vil drive hjemmesiden, da Djangos server ikke er anbefalet som produktions server, i stedet for Djangos egen server.

Implementering

Der skal laves ændringer i settings.py filen, hvor hjemmesidens public IP og domænenavn skal tilføjes, se Figur 4, derud over skal IP'en laves til static, så den ikke ændrer sig, hvilket bliver gjort på AWS lightsail kontrol siden, hvilket kan ses på Figur 5.

```
ALLOWED_HOSTS = ['www.control-center.xyz', '3.210.255.190']
```

Figur 4 ændring i setttings.py



Figur 5 statisk ip

Derefter skal der sættes NGINX op, på serveren, hvilket gøres ved at downloade NGINX gennem package manageren.
Efter NGINX er downloadet skal der ændres i NGINX settings filen, der kan findes under "/etc/nginx/sites-enabled/default", der kan dig også sættes en settings fil op, specifikt for hjemmesiden.

I settings filen skal der sættes hvad domænenavnet er, under "server_name", samt for at domænet bliver vist i browserens url bar, skal det sendes upstream, ved hjælp af "proxy_set_header", for til sidst at sende http requestene til serverens localhost ip og port, som Gunicorn serveren vil bruge, hvilket gøres med "proxy_pass,"
Disse settings kan ses i Figur 6, under "location /".

For at sidens static filer kan vises, skal der fortælles til NGINX, hvor disse filer kan findes, hvilket gøres gennem "location /static/", som kan ses på Figur 6.

```
server_name www.control-center.xyz control-center 3.69.102.33;

location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        ##try_files $uri $uri/ =404;
#       proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $server_name;
        proxy_pass http://0.0.0.0:8000;
}
location /static/ {
        alias /home/bitnami/Django-final-server/static/;
}
```

Figur 6 nginx settings

Efter NGINX er sat op, skal der laves en static mappe, hvilket nemt kan gøres gennem Djangos kommandoer, ved at lave en static mappe, i projektets root directory, for derefter at bruge kommandoen:
$ python manage.py collectstatic

Gunicorn kan installeres med pip, ved at give kommandoen:
$ pip3 install gunicorn

Efter Gunicorn er installeret, kan hjemmesiden startes op med kommandoen:
$ gunicorn website.wsgi:application –bind 0.0.0.0:8000 --access-logfile /var/log/web_log/gunicorn-access.log --error-logfile /var/log/web_log/gunicorn-error.log

Hvilket fortæller Gunicorn at den skal starte op på localhost port 8000, og bruge website projektets WSGI-fil.

Verifikation

Test af deployment af hjemmesiden på AWS-serveren er som på Tabel 1.

Tabel 1: Tests til verifikation af opgave

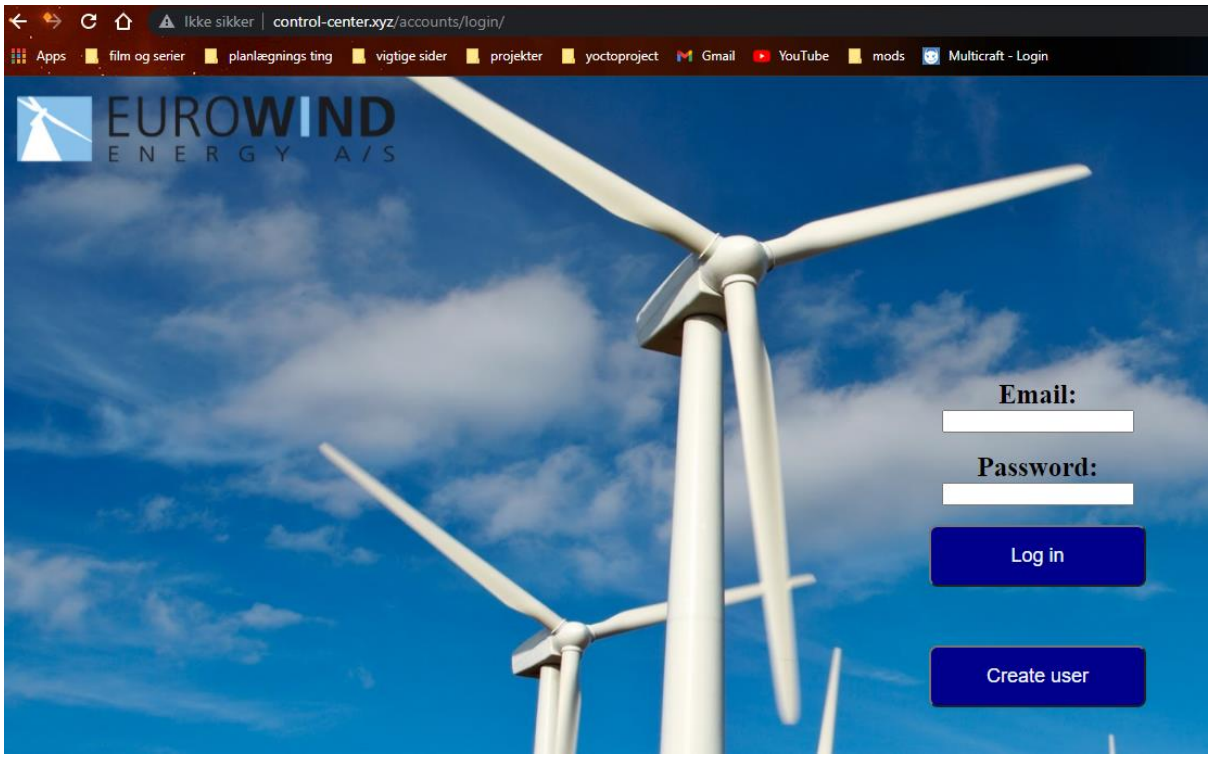
Test	Test Steps	Pre-requisites	Pass-betingelser	Resultat
Test af hjemmesiden på AWS	1. indtast i en Browser ” http://3.69.102.33:8000”	1. Hjemmesiden kører inde på AWS-serven	• Kan tilgå hjemmesiden, se Figur 7	Bestået
Test af DNS	1. indtast i en Browser ” http://www.control-center.xyz”	1. Hjemmesiden kører inde på AWS-serven	• Kan tilgå hjemmesiden, se Figur 8	

Testresultat

Se testresultatet på Figur 7.

Figur 7 hjemmesiden via cloud

Se testresultatet på Figur 8.



Figur 8 DNS

Konklusion

Hjemmesiden er nu deployed på en AWS-server, hvor det hele virker som forventet, udover lidt problemer med DNS records, som kan blive fikset ved at bruge AWS nameservers, samt skulle det være muligt at gå fra http til https, da nginx nu er sat op.

Referencer

[1]	Lightsail SSH-dokumentation https://lightsail.aws.amazon.com/ls/docs/en_us/articles/understanding-ssh-in-amazon-lightsail (sidst besøgt 11-10-2021)
[2]	Pycharms guide til SSH https://www.jetbrains.com/help/pycharm/create-ssh-configurations.html (sidst besøgt 11-10-2021)
[3]	Pycharms guide til remote deploy https://www.jetbrains.com/help/pycharm/creating-a-remote-server-configuration.html#config (sidst besøgt 11-10-2021)
[4]	GoDaddy https://dk.godaddy.com/ (sidst besøgt 15-10-2021)
[5]	Nginx https://nginx.org/en/ (sidst besøgt 15-10-2021)
[6]	Gunicorn https://gunicorn.org/ (sidst besøgt 15-10-2021)