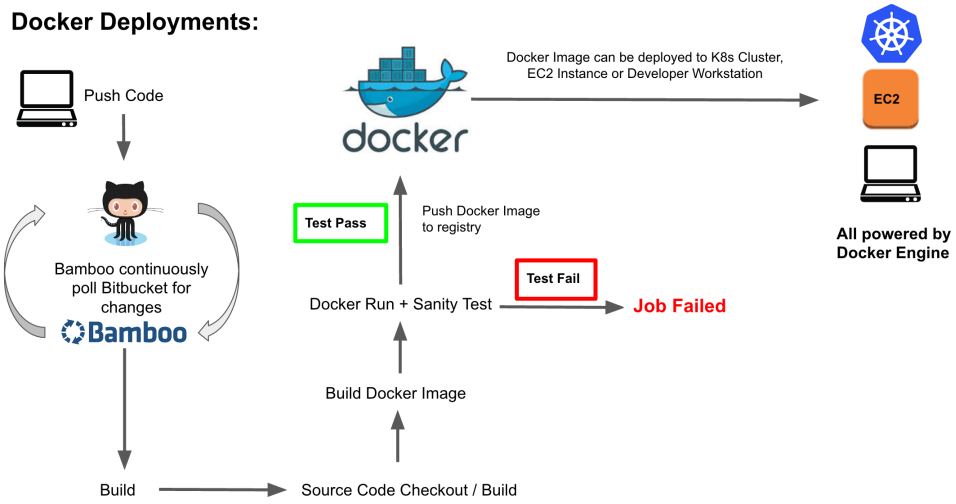
Project - Deploy to AWS EC2 (AWS Route 53 + AWS RDS Postgrest)



Web Deployment Preparation

- **Create A New GitHub Repository**
- **Create** `.gitignore`
- **Create** `requirements.txt`
- **Create** `.env`
- **Connect github**

- Create A New GitHub Repository

GitHub Repository

- Connect GitHub

```
-> git init
-> git add .
-> git commit -m "first commit"
-> git branch -M main
-> git remote add origin
git@github.com:ericarthuang/Copy_Learning_Django.git
-> git push -u origin main
```

Sepup Django Project

> ## - Setup Venv to Create Project
>
> *python -m venv venv*
> *venv\Scripts\activate.bat*

> ## - Create Project
>
> - pip install django
> - django-admin -> django-admin --version
> - django-admin startproject `djangotoec2_main`
> - chagnge root- `djangotoec2_main' to` DjangoToEC2_Project`
> - cd `djangotoec2_main` -> python manage.py runserver

> ## - Creating apps and Register App & URL
>
> - Creating app

-> py manage.py startapp blog_app

- Register App

-> In `djangotoec2_main/settings.py` , add `blog_app` in the `INSTALLED_APPS list`
-> `Open the file blog_app/views.py`

- Register URL

-> Create a URLconf file called `urls.py` in `blog_app folder`
-> Point the root URLconf at the `blog_app.urls` module
-> In `djangotoec2_main/urls.py` , add `import for` `django.urls.include` and insert an include() in the urlpatterns list

> ## - urls, views, templates, and static
>
> - In `djangotoec2_main/urls.py` , insert an path in the urlpatterns list

-> path('', include('blog_app.urls')),

- In `djangotoec2_main/views.py` , define the index(requrest) for index.html

-> Can define multiple html in `djangotoec2_main/views.py` to link htmlfiles in templates

templates

- Using `templates` folder to keep htmlfiles

-> Create `templates` folder in 'blog_app' folder
-> Create htmlfiles in the 'templates' folder
-> *Can create multiple folders for multiple apps -> `blog_app` folder -> `templates` folder -> `blog_app` folder -> htmlfiles*
-> *NOTICE: The Name of Folder should be sames as app**

- Create `base.html` for htmlfiles

static

- Using `static` folder to keep css, images, and other static files

-> Create `static` folder in 'blog_app' folder
-> Create `static` files in `static` folder
-> *Can create multiple folders for multiple apps*
-> *`blog_app` folder -> `static` folder -> `blog_app` folder -> css, images, and other static files*
-> *NOTICE: The Name of Folder should be sames as app**

- Put {% load static %} in `base.html`

-> `<link rel="stylesheet" href="{% static 'blog_app/main.css' %}">`

- Config Static and Media files for deployment

- **go to `settings.py`**

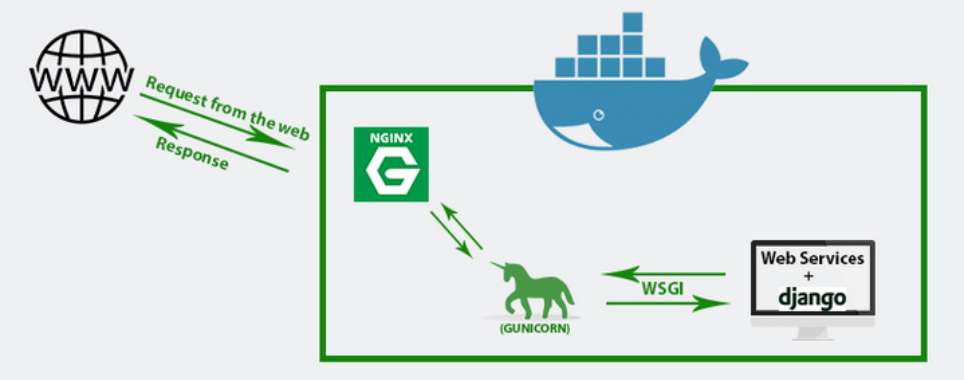
-> DEBUG = (os.getenv('DEBUG_VALUE') == 'True')
-> STATIC_ROOT = os.path.join(BADE_DIR, 'staticfiles')
-> python manage.py collectstatic

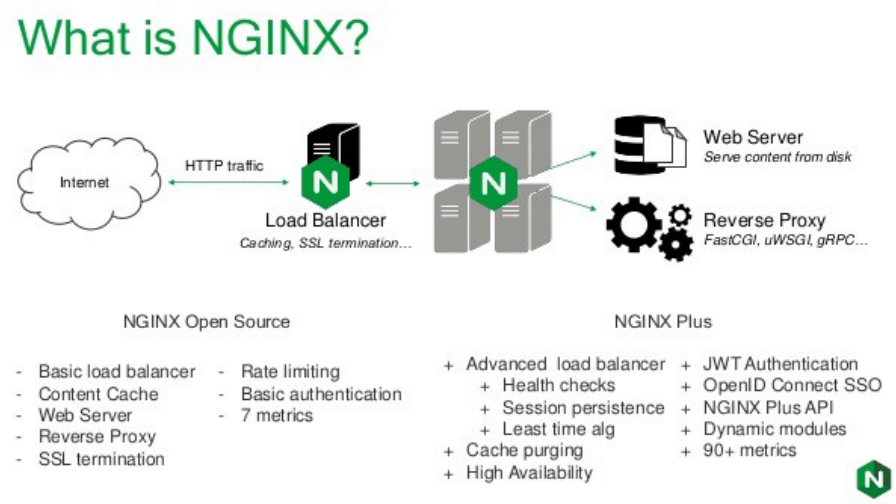
- Admin Page

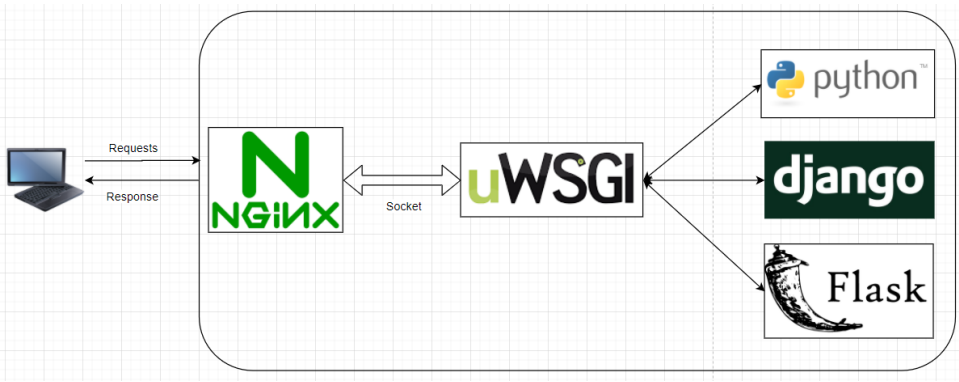
- CMD: python manage.py makemigrations
- CMD: python manage.py migrate
- CMD: python manage.py createsuperuser

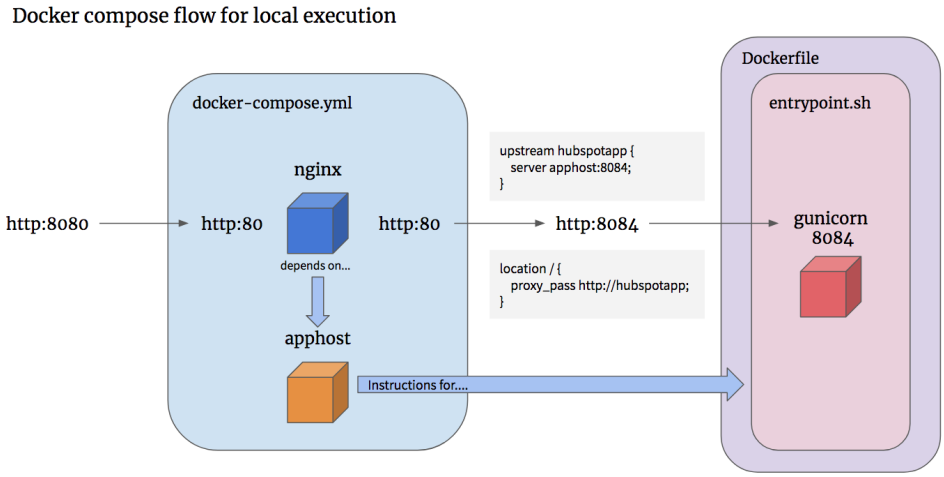
-> go to `http://127.0.0.1:8000/admin/` for logining
-> go to `http://127.0.0.1:8000/admin/auth/user/1/change/` to know the hassing password

Containerize Django and Nginx









- Create Dockerfile outside Project

```
FROM python:3.10.8-slim-buster
WORKDIR /app
COPY ./DjangoToEC2_Project ./
RUN pip install --upgrade pip --no-cache-dir && \
    pip install -r /app/requirements.txt --no-cache-dir
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
#CMD ["waitress-serve", "--listen=0.0.0.0:8000",
"djangotoec2_main.wsgi:application"]
#CMD ["gunicorn" "djangotoec2_main.wsgi:application", "--
bind", "0.0.0.0:8000"]
```

- Create Docker Image

- docker build -t djangotoec2:1.0 .

- Create Docker Repository and push Image to Repository

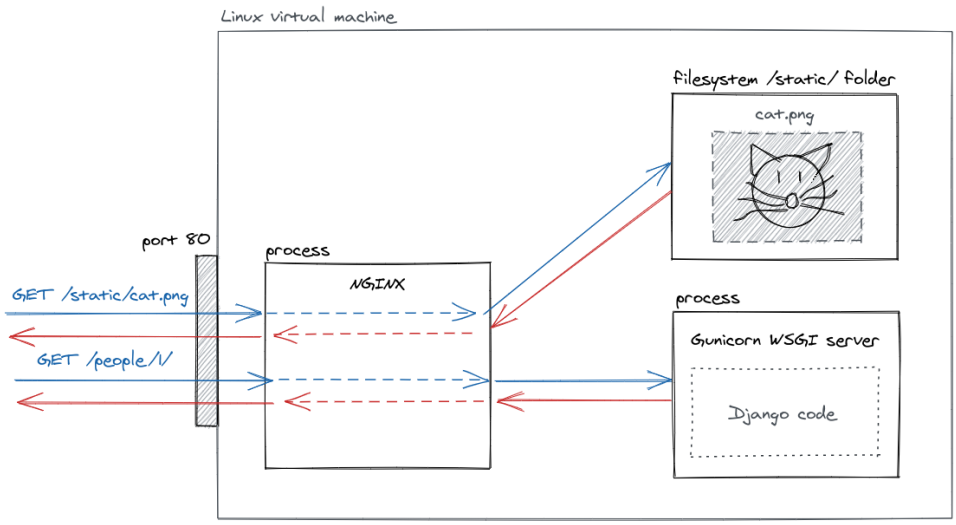
- Create New Reposity: ericarthaung/djangotoawsec2

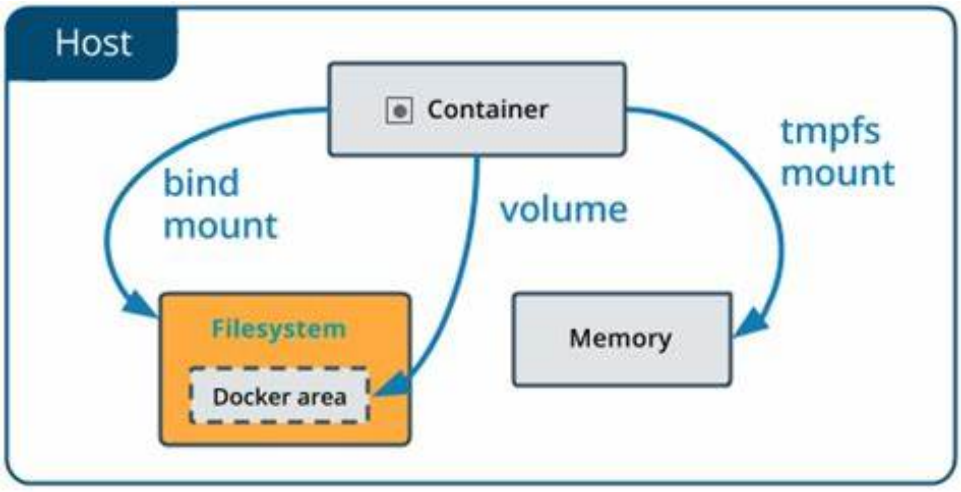
-> docker tag djangotoec2:1.0 ericarthaung/djangotoawsec2:1.0
-> docker push ericarthaung/djangotoawsec2:1.0
-> docker pull ericarthaung/djangotoawsec2:1.0

- Create Docker Container

-> docker run --name djangotoec2 -it -d -p 8000:8000 djangotoec2:1.0
-> docker run --name djangotoec2 -it -d -p 8000:8000
ericarthaung/djangotoawsec2:1.0

Create Container by using docker-compose.yml





- Create `nginx` folder outside Project
- Create `default.conf` in `nginx` folder
- Create `Dockerfile` in nginx folder

- Create `docker-compose.yml` outside Project

-> docker-compose build
-> docker-compose up'

- Setup `.env`

-> pip install python-decouple

- modify `settings.py`

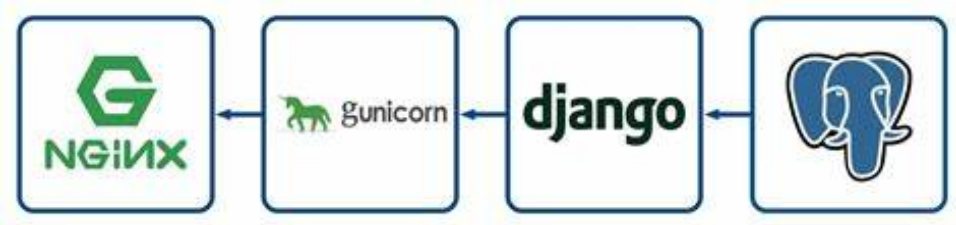
-> from decouple import config -> SECRET_KEY = config('SECRET_KEY') -> DEBUG = config('DEBUG', default=False, cast=bool)

- modify `docker-compose.yml`

```
services:
  django_app:
    env_file:
      - .env
```

- rebuild imanges and containers

AWS Service - RDS - Postgrest



- CMD: pip install psycopg2

- `Django` Example

-> Delete `db.sqlite3`
-> In `settings.py`

```
DATABASES = {
   'default': {
       'ENGINE': 'django.db.backends.postgresql',
       'NAME': '<DB_NAME>',
       'USER': '<DB_USER>',
       'PASSWORD': '<DB_PASSWORD>',
       'HOST': '<DB_HOST>',(database Connectivity & security-
Endpoint)
       'PORT': '<DB_PORT>',
   }
}
```

AWS RDS

- Ceteat Database

-> Choose a database creation method: Standard Create
-> Engine options: PostgreSQL
-> Templates: Free Tier
-> Settings: (link with `.env`)
-> Connectivity-Public access: yes
-> Connectivity- VPC security group(firewall): Create New: SG for DB;

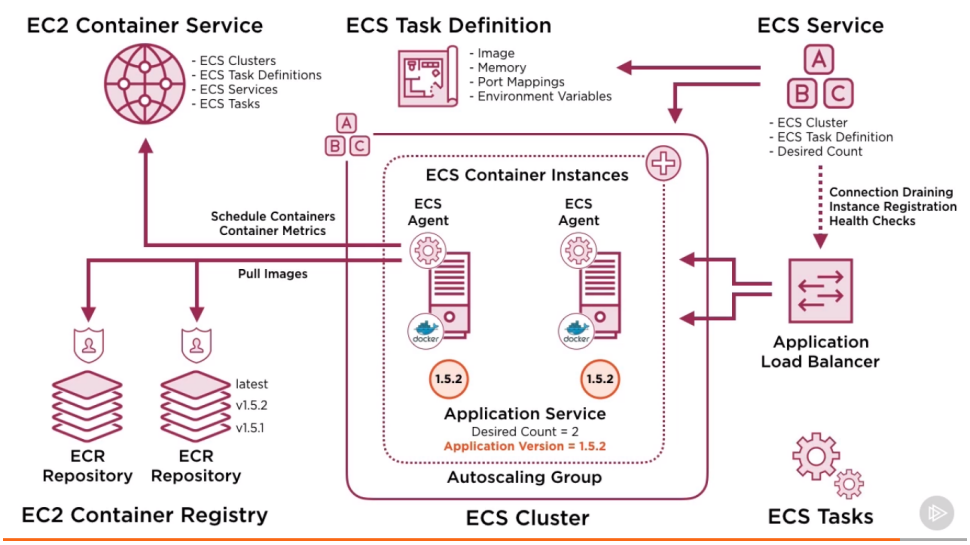
-> Connectivity- Additional configuration: 5432
-> Additional configuration - Initial database name:

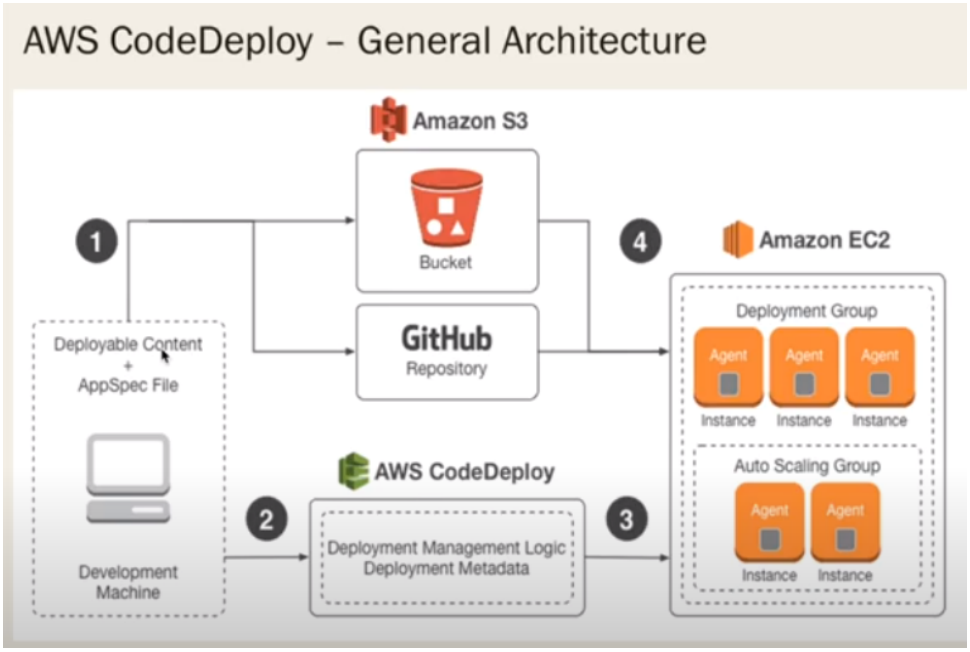
- CMD: python manage.py makemigrations
- CMD: python manage.py migrate

- CMD: python manage.py createsuperuser

AWS Service - AWS EC2

AWS EC2





- Launch instance: Create a virtual machine

- EC2 Dashboard

-> Name and tags
-> Application and OS Images (Amazon Machine Image - Amazon Linux)
-> Instance type
-> Key pair (login): Crerate new key pair(Keep the filename.pem)
-> Network settings
-> Configure storage
-> Launch Instance

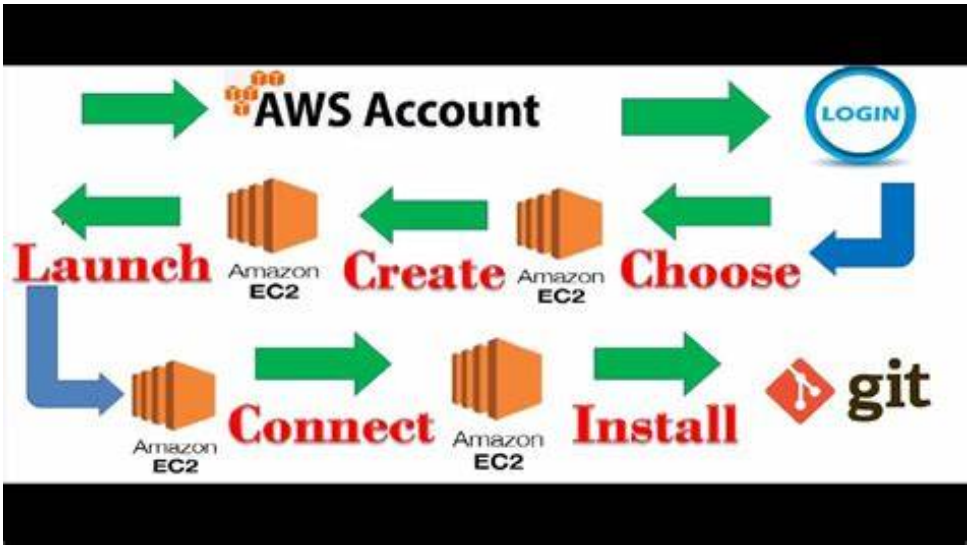
- View Instance

- Connect EC2

-> Choose Instance
-> Press Connect
-> `EC2 Instance Connect (browser-based SSH connection)`
-> CMD:chmod 400 DjangoToEC2.pem
-> CMD: ssh -i "DjangoToEC2.pem" ec2-user@ec2-175-41-205-149.ap-northeast-1.compute.amazonaws.com

- Install Git in AWS EC2 Instance

-> sudo yum update -y
-> sudo yum install git -y
-> git --version



- Install Docker in AWS EC2 Instance

-> sudo amazon-linux-extras install docker
-> sudo service docker start
-> sudo usermod -a -G docker ec2-user
-> sudo chkconfig docker on
-> sudo yum install -y git

-> sudo reboot
-> docker --version

- Connect RDS Postgrest with EC2 connection

- Running Web Service in AWS EC2

-> CMD: ssh -i "DjangoToEC2.pem" ec2-user@ec2-175-41-205-149.ap-northeast-1.compute.amazonaws.com

-> sudo curl -L https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
-> sudo chmod +x /usr/local/bin/docker-compose
-> docker --version
-> docker-compose version

- Install Docker-Compose in AWS EC2 Instance

-> sudo curl -L https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
-> sudo chmod +x /usr/local/bin/docker-compose
-> docker-compose version

- git clone GitHub Repository

-> git clone `https://github.com/ericarthuang/DjangoToEC2_Project.git`
-> ls -> cd DjangoToEC2_Project
-> ls -a
-> vi .env -> paste .env into vi .env -> docker-compose up --build

---

AWS Service - AWS Route 53

- Create hosted zone

-> Domain Name: djangotoes2.com
-> Type: Public hosted zone
-> Create hosted zone

- Go to Hosted zone details

-> Choose djangotoes2.com type: NS
-> Choose `Value` copy to `your web(I don't have)`

> - Go to EC2 `Network & Security` : `Elastic IPs`
>
> -> Allocate Elastic IP address -> Associate Elastic IP address: Instance -> Associate

> - Go to EC2 Instance
>
> -> `IPv4 Public IP`

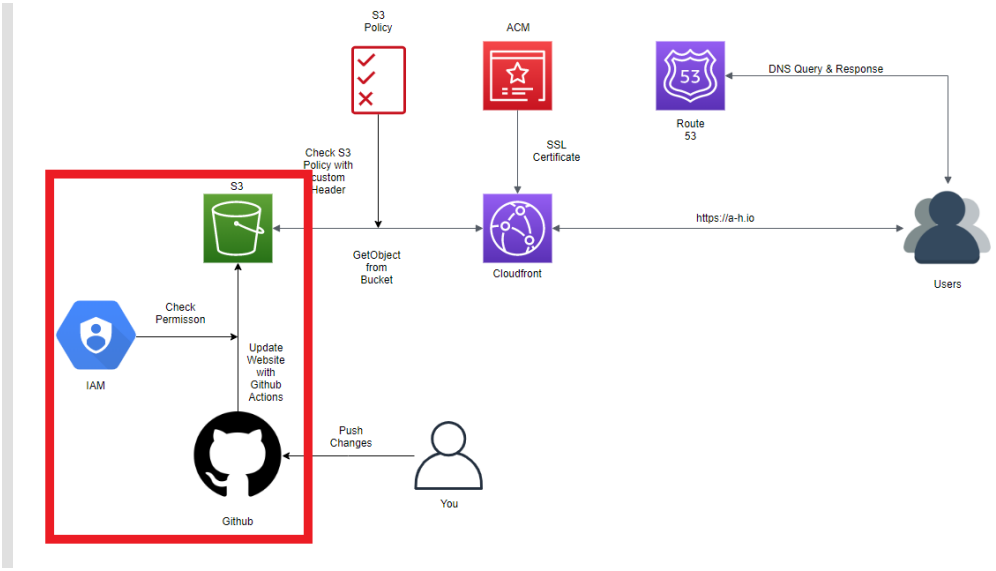
> - Go to Route 53
>
> -> Create Record -> Paste `IPv4 Public IP` in `value` field -> Create Records

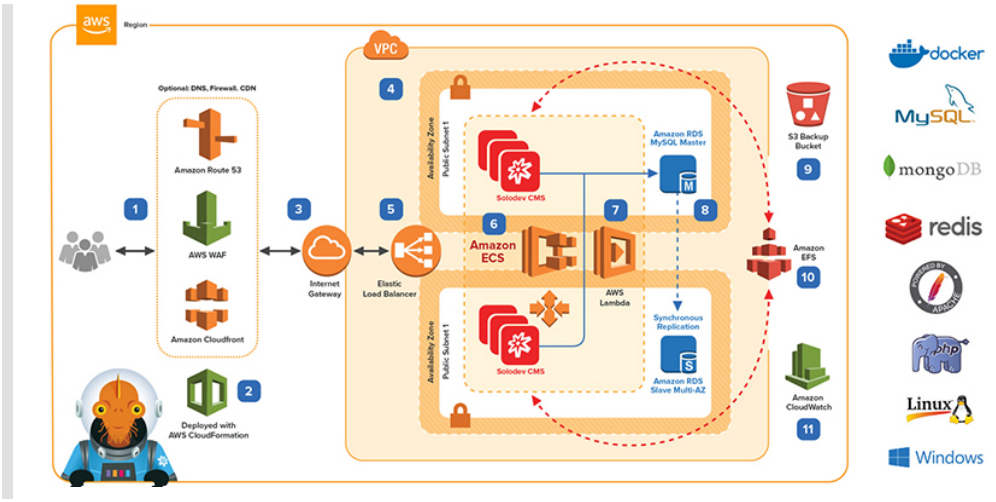
> -> Create Record -> Record Name: www -> Reocrd Type: CANME -> Paste `djangotoes2.com` in `value` field
> -> Create Records

AWS Service - AWS Certificate Manager

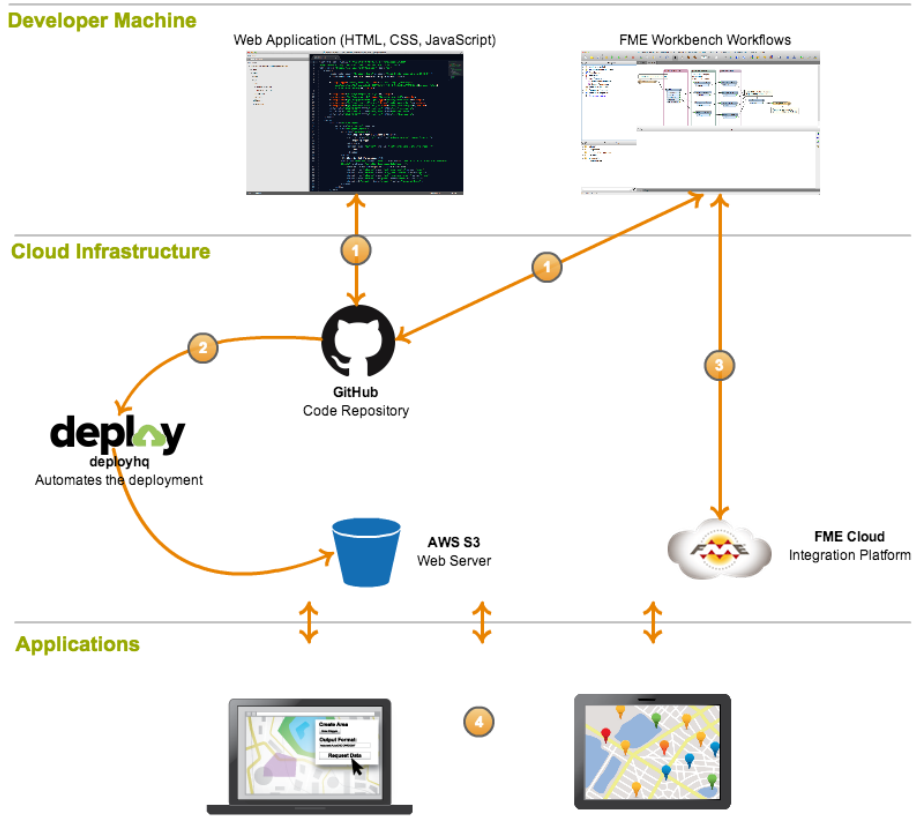
> - Request A Certificate
>
> -> Domain names: `djangotoes2.com`
> -> DNS validation - recommended
> -> Request
>
> - View Detail of Certificate
>
> -> Create Records in Route 53

> - Go to EC2 Instance
>
> -> Load Balancing: Load Balancers
> -> Create Load Balancer
> -> Application Load Balancer: Create -> Name: `my-lb`
> -> ...





Using AWS S3 for File Uploads



Create AWS S3 Bucket

AWS S3 Website

- Create AWS S3 Bucket

-> `django-learning-files`

- Permession
- CORS Configuration

```
[
  {
   "AllowedHeaders": [
     "*"
   ],
   "AllowedMethods": [
     "GET",
     "PUT",
     "POST",
     "DELETE"
   ],
   "AllowedOrigins": [
     "*"
   ],
   "ExposeHeaders": [
     "Access-Control-Allow-Origin"
   ]
```

```
 }
]
```

Create New User in AWS S3

- IAM
- Add Users

-> django_user(Select AWS credential type: Access key - Programmatic access)
-> `Attach existing policies directly`
-> `AmazonS3FullAccess`
-> Access key ID + Secret access key

Link Django with AWS3 and Using .env to Store the Scecrte Variables

- `pip install boto3`
- `pip install django-storages`
- `pip install python-dotenv`

```
import os
from dotenv import load_dotenv
load_dotenv()
os.getenv('ENV_VAR')
```

- `setup .env`

-> AWS_STORAGE_BUCKET_NAME=*******
-> AWS_ACCESS_KEY_ID=********
-> AWS_SECRET_ACCESS_KEY=********

- Go to `settings.py`

```
-> import os
-> from dotenv import load_dotenv
-> load_dotenv()
-> INSTALLED_APPS = `[stroages]`
-> AWS_STORAGE_BUCKET_NAME =
os.getenv('AWS_STORAGE_BUCKET_NAME')
-> AWS_ACCESS_KEY_ID = os.getenv('AWS_ACCESS_KEY_ID')
-> AWS_SECRET_ACCESS_KEY = os.getenv('AWS_SECRET_ACCESS_KEY')
-> AWS_S3_FILE_OVERWRITE = False
-> AWS_DEFAULT_ACL = None
-> DEFAULT_FILE_STORAGE =
'storages.backends.s3boto3.S3Boto3Storage'
```

- go to `user_app/models.py`

-> # can not use below code due to AWS S3 for resizing images

> - upload images to the AWS S3 BUCKET

> ## Upload and Download files to AWS S3
>
> Reference: Upload and Download files from AWS S3 Bucket using python

```
# .ENV VARS CONFIG
load_dotenv()
aws_bucket_name = os.getenv('AWS_STORAGE_BUCKET_NAME')
aws_access_key_id = os.getenv('AWS_ACCESS_KEY_ID')
aws_secret_access_key= os.getenv('AWS_SECRET_ACCESS_KEY')

# S3 BUCKET CONFIG
s3 = boto3.resource("s3")
my_bucket = s3.Bucket(aws_bucket_name)
my_bucket.upload_file(Key='index.html',
Filename='./index.html')
my_bucket.download_file(Key='index.html',
Filename='./index.html')
```

-- Memo End --