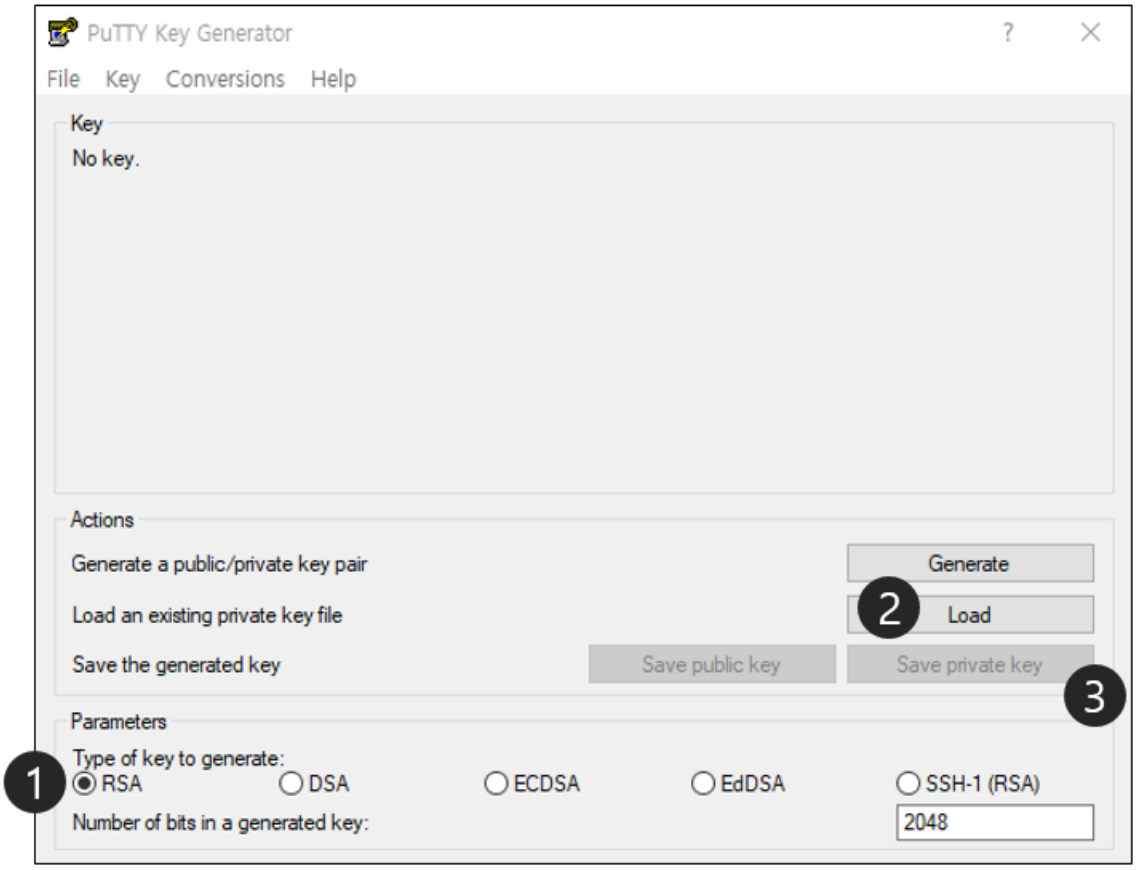
# 소피의 책방 배포 가이드

## 1. AWS EC2에 로그인하기

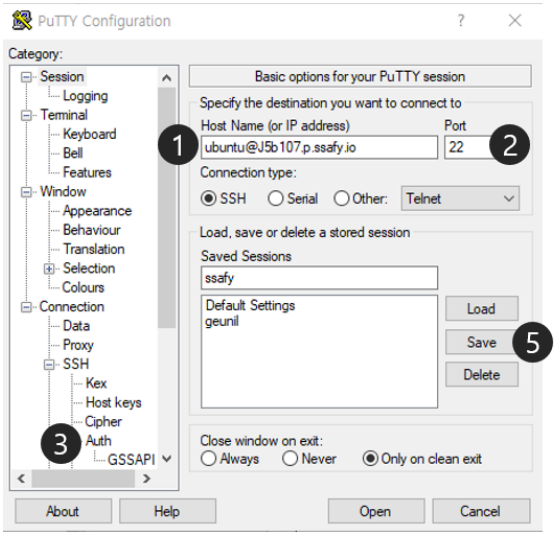
> SSAFY에서는 AWS Console 로그인을 하도록 하는게 아니고 미리 만들어둔 다음에 pem 파일을 나눠줬다.

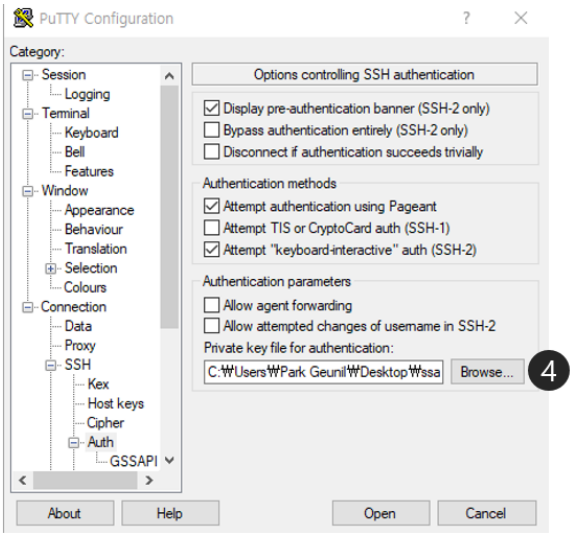
### 1. Puttygen으로 ppk 파일 만들기




1. RSA로 설정한다.
2. Load를 눌러서 받은 pem key를 가져오고
3. Save private key를 눌러서 저장한다. (저장할 때 이름은 상관없다.)

### 2. Putty로 로그인하기




1. Host Name에는 EC2 인스턴스의 public IP를 입력한다. (ubuntu는 인스턴스를 만들 때 설정한 것이다.)
2. Port 번호는 22번으로 고정이다.
3. Connection > SSH > Auth를 눌러서 오른쪽 그림의 창을 연다.
4. Browse를 눌러서 Puttygen으로 만들어 놓은 ppk 파일을 넣는다.
5. 이름을 마음대로 정하고 Save를 하고 Open을 눌러서 연다.

# 2. Docker 기본 설정.

## 1. Docker 깔기

```
# ubuntu 설정 update
sudo apt-get update

# curl 명령어를 설치한다.
# curl은 CLI에서 web에 직접적으로 요청하기 위한 명령어
sudo apt-get install curl

# -fsSL은 fail silent show-error location을 합친 것
# [참고] https://explainshell.com/explain?cmd=curl+-fsSL+example.org#
# su 명령어는 현재 사용자를 로그아웃하지 않은 상태에서 다른 사용자의 계정으로 전환하는 명령어이다.
curl -fsSL https://get.docker.com/ | sudo sh
```

## 2. Docker에 권한 부여하기

docker는 Linux의 root 유저 권한이 필요하여 설정함. 후 putty로 재로그인해야 권한 설정이 잘 됨

```
# 현재 사용자에게 root 권한 부여
# usermod: 사용자 계정에 관련된 설정을 변경
# -G 옵션: 사용자 계정의 2차 그룹 설정
# -a 옵션: -G 옵션과 같이 사용하는 옵션으로 현재 사용자가 가지고 있는 2차 그룹 외에 추가로 2차 그룹을 지정할 때 사용
sudo usermod -aG docker $USER
```

### 3. Docker version 확인

docker version을 입력했을 때 버전 정보가 나오면 다운로드 성공한 것

```
docker version
```

### 4. Docker hub 로그인

Docker hub에 회원가입한 후 아래의 명령어를 실행했을 때 username과 password를 입력해 로그인함.

```
docker login
```

## 3. build 파일을 만든다.

### 1. React

```
npm run build
```

### 2. Spring

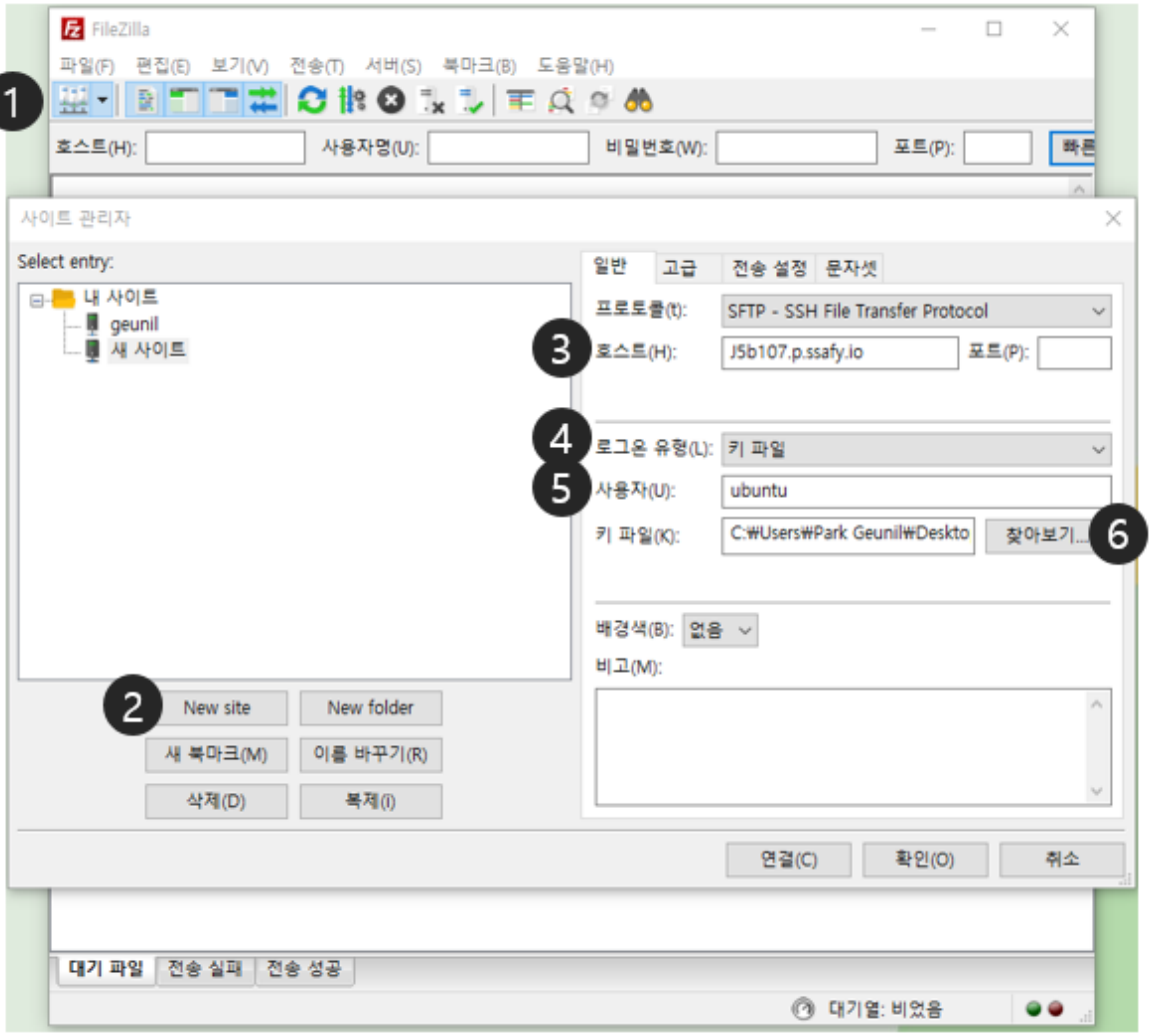
먼저 src 폴더의 test 폴더를 삭제한다.

```
gradlew build
```

### 3. Django

장고는 따로 anaconda에서 build 파일을 만들지 않고 통채로 올렸다.

# 4. 빌드 파일 EC2로 옮기기




1. 사이트 관리자 창을 연다
2. New site를 만들어서 내 사이트에 새롭게 만든다.(이름 설정 가능)
3. 내 EC2 호스트를 입력한다. 포트는 비워둔다.
4. 로그온 유형은 키 파일로 바꾼다.
5. 사용자는 ubuntu이다. (EC2 인스턴스로 만든 사용자 입력)
6. key file은 받은 pem 키

# 5. 각 컨테이너마다 Dockerfile을 설정한다.

## 1. 폴더구조

- docker-compose.yml
- FE ( **react build한 파일 폴더 채로 가져오기** )
  - index.html
  - nginx.conf
  - .....

- DB
  - Dockerfile
- SpringBE
  - Dockerfile
  - sopy-0.0.1-SNAPSHOT.jar ( **spring boot build 파일** )
- djangoBE
  - AI ( **django 폴더 통채로** )
    - ……
  - Dockerfile
- nginx
  - nginx.conf
- certbot-etc
  - options-ssl-nginx.conf
  - ssl-dhparams.pem
  - …..

## 2. docker-compose.yml

docker-compose up -d 명령어를 수행하기 위한 파일

```
# 현재 docker-version임, 바꿀일이 없음
version: "3"

services:
  nginxproxy:
    # 아래의 것들을 모두 하고 나서 proxy설정을 켤 것.
    depends_on:
      - nginx
      - db
      - spring
      - django
    # pull할 image
    image: nginx:latest
    # container 이름을 설정할 수 있다.
    container_name: proxy
    # 외부에서 들어올 수 있는 port를 여러개 열 수 있다.
    ports:
      - "80:80"
    # 혹시 꺼지게 되면 다시 restart할 수 있다.
    restart: always
    # volume 설정 <연결하고자 하는 EC2 파일 혹은 폴더>:<컨테이너 내부의 파일 혹은 폴더>
    volumes:
      # FE
      - ./FE:/usr/share/nginx/html
      # 프록시 설정 파일
      - ./proxy/nginx.conf:/etc/nginx/nginx.conf

  nginx:
    image: nginx:latest
    container_name: front
```

```
    restart: always
    volumes:
      - ./FE:/usr/share/nginx/html
      # 새로고침을 위한 설정으로 바꾸기 위해서 default.conf를 nginx.conf로 변경
      - ./FE/nginx.conf:/etc/nginx/conf.d/default.conf

  db:
    image: mysql:5.7
    container_name: mydb
    restart: always
    # db삭제되지 말라고 미리 연결해둠.
    volumes:
      - /home/ubuntu/mysqldata:/var/lib/mysql
    # MYSQL 환경설정
    environment:
      # ROOT 비밀번호 설정
      MYSQL_ROOT_PASSWORD: 092812
      # 내가 쓸 DATABASE, 없을 경우 알아서 자동으로 schema 생성해줌
      MYSQL_DATABASE: ssafy_db
    # command는 컨테이너 내에서 설정하는 것
    command:
      # 한글 파일 설정
      - --character-set-server=utf8
      - --collation-server=utf8_general_ci
    ports:
      - "3306:3306"

  spring:
    # db가 켜져야 spring을 빌드할 수 있기 때문에 depends_on으로 설정
    depends_on:
      - db
    # build는 Dockerfile이 있을 때 사용
    build:
      # context는 dockerfile이 있는 위치
      context: ./SpringBE
      # dockerfile은 실제 사용할 Dockerfile
      dockerfile: Dockerfile
    container_name: spring
    ports:
      - "5000:5000"

  django:
    depends_on:
      - spring
    build:
      context: ./djangoBE
      dockerfile: Dockerfile
    container_name: django
    ports:
      - "5001:5001"

volumes:
  mydb:
```

## 3. Nginx Proxy 설정

Nginx proxy용 nginx.conf 설정

```
user nginx;
worker_processes auto;

error_log /var/log/nginx/error.log warn;
pid       /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" "$request_uri"
"$uri"'
                      '"$http_user_agent" "$http_x_forwarded_for"';
    access_log  /var/log/nginx/access.log  main;
    sendfile on;
    keepalive_timeout 65;

    # docker-web으로 왔을 때 연동할 서버를 설정
    upstream docker-web {
        # nginx로 간다.
        server nginx:80;
    }

    server {
        listen 80;
        # 요청을 보내는 서버 이름
        server_name j5b107.p.ssafy.io;
        # 설정을 하기 위한 것
        location / {
            # 443으로 왔을 경우 proxy_pass로 upstream docker-web으로 가라
            proxy_pass         http://docker-web;
            proxy_redirect     off;
            proxy_set_header   Host $host;
            proxy_set_header   X-Real-IP $remote_addr;
            proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header   X-Forwarded-Host $server_name;
        }
    }
}
```

## 3. FE

FE용 nginx.conf 설정

```
server {
  # FE의 port
  listen 80;
  listen [::]:80;
  server_name j5b107.p.ssafy.io;

  location / {
    # 연결할 html이 들어있는 폴더
    root    /usr/share/nginx/html/;
    # 읽을 index 파일의 확장자명, 간혹 htm으로 빌드되는 것 때문에 이렇게 쓴다고 함
    index   index.html index.htm;
    # 새로고침 했을 때 다시 index.html파일로 try하라는 것
    try_files $uri $uri/ /index.html;
  }
}
```

## 3. DB

mysql은 따로 Dockerfile을 만들 필요가 없음(모든 DB 관련 설정은 docker-compose.yml에 있음)

## 4. Spring Dockerfile

1. spring Dockerfile

```
# jdk를 가져온다.
FROM openjdk:8-jdk-alpine
# volume을 가져온다.
VOLUME /tmp
# 변수로 설정할 수 있음, JAR_FILE이라는 이름은 현재 위치의 모든 jar 확장자
ARG JAR_FILE=*.jar
# 왼쪽의 파일을 app.jar이라는 이름으로 컨테이너 내부로 카피함
COPY ${JAR_FILE} app.jar
# ENTRYPOINT는 해당 컨테이너가 실행됐을 때 실행되는 명령어
# 아래 명령어는 jar 파일로 서버를 켜는 명령어임
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

## 5. Django Dockerfile

django Dockerfile

```
FROM continuumio/anaconda3

# AI에 있는 모든 파일을 만든다.
COPY ./AI .

# pip install
RUN /bin/bash -c "pip install -r requirements.txt"
# migrate 필요함
RUN python manage.py migrate
# 컨테이너 실행하면서 runserver 함
ENTRYPOINT ["python", "manage.py", "runserver", "0.0.0.0:5001"]
```

# 5. Jenkins 배포 설정

Jenkins는 다음 기회에 !

# 😢취소 명령어

아마도 제일 많이 쓴 명령어

```
# 구동중인 컨테이너들을 종료시킴
docker stop $(docker ps -a -q)
# 컨테이너들 삭제
docker rm $(docker ps -a -q)
# 이미지들 삭제
docker rmi $(docker images -a -q)
# 각종 volume 설정들 삭제
docker system prune -a --volumes
```