배포

1. **EC2 설정**
    - XShell 다운
    - 새 세션 만들기
    - **연결-호스트**에 AWS EC2 인스턴스의 **퍼블릭 도메인**을 입력
    - **연결-사용자인증**의 public key 설정 클릭
    - AWS IAM 설정 .pem 파일을 사용자 키로 지정하고 확인
    - 새 세션 더블클릭
    - EC2 instance에 맞춰서 유저이름 입력
    - 세션 연결 성공
2. **Java**
    - `sudo apt-get install java-1.8.0-openjdk`
3. **MySQL**
    - `sudo apt-get install mysql-server`
    - `sudo systemctl start mysql`
    - `sudo systemctl enable mysql`
4. **FrontEnd**
    - React 프로젝트 폴더로 이동
    - `sudo npm install`
    - `sudo npm run build`
5. **BackEnd**
    - SpringBoot 프로젝트 폴더로 이동
    - src/main/resources에 aws.properties와 datasource.properties 생성
    - src/main/resources에 아래 파일들 설정

        ▼ **application.properties**

        ```
        spring.jpa.hibernate.ddl-auto=update

        spring.jpa.show-sql=true

        spring.servlet.multipart.max-file-size=-1
        spring.servlet.multipart.max-request-size=-1
        ```

        ▼ **aws.properties**

        ```
        별도 파일 참고
        ```

▼ **datasource.properties**

```
별도 파일 참고
```

▼ **jwt.properties**

```
별도 파일 참고
```

- `sudo vim /etc/systemd/system/springboot.service`

```
#springboot.service
[Unit]
Description=Cocktailus Service
After=network.target

[Service]
ExecStart=/bin/bash -c "exec java -jar @build경로@"

User=root
Group=root

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

- `[Service]` 의 `ExecStart=/bin/bash -c "exec java -jar @build경로@"` 의 @build경로@에 SpringBoot build jar파일 경로 넣기
- SpringBoot 프로젝트 폴더에서 `sudo gradle build`
- `sudo systemctl start springboot.service`
- `sudo systemctl enable springboot`

6. **Nginx**

- `sudo apt-get install nginx`
- `sudo vim /etc/nginx/sites-enabled/default`

```
server {
        root @Frontend Build 경로@;

        index index.html index.htm index.nginx-debian.html;
        server_name i5a103.p.ssafy.io; # managed by Certbot



        location / {
                #root   /home/ubuntu/FE/0818/S05P13A103/frontend/build;
                #index  index.html index.htm;
                #;
                try_files $uri $uri/ /index.html;
        }

        location /api {
            proxy_pass http://localhost:8080;
            proxy_http_version 1.1;
            proxy_set_header Connection "";
            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-Proto $scheme;
            proxy_set_header X-Forwarded-Host $host;
            proxy_set_header X-Forwarded-Port $server_port;
        }

        listen [::]:443 ssl ipv6only=on; # managed by Certbot
        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/i5a103.p.ssafy.io/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/i5a103.p.ssafy.io/privkey.pem; # managed by Certbot
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = i5a103.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        #location / {
        #       try_files $uri $uri/ /index.html;
        #}
        listen 80 ;
        listen [::]:80 ;
    server_name i5a103.p.ssafy.io;
    return 404; # managed by Certbot


}
```

- **server**의 `root    @Frontend Build 경로@` 의 **@build경로@**에 React 프로젝트 build 폴더 넣기
- SPA의 경우 새로고침시 404에러를 막기 위해 **location /** 에 `try_files $uri $uri/ /index.html` 넣기
- **location /api** 에서 Backend와 Frontend의 http, https 불일치를 막기 위해 Proxy 적용

```
location /api {
    proxy_pass http://localhost:8080;
    proxy_http_version 1.1;
    proxy_set_header Connection "";
    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-Proto $scheme;
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Forwarded-Port $server_port;
}
```

- 나머지는 아래 **letsencrypt**의 `sudo certbot ceronly --standalone -d 'domain명'` 을 통해 적용

- `sudo service nginx start`

7. **letsencrypt**

- `sudo apt-get install snap`
- `sudo apt-get install letsencrypt -y`
- `sudo snap install --classic certbot`
- `sudo certbot certonly --standalone -d 'domain명'`