
# 어디까지 할 수 있는지 한계를 확인하는 Full Stack 도전기

## Week 2

from h01010 Company

컴퓨터공학과 201802106 **서연주**

## *WHAT I DID*

- [x] 인스턴스 생성 및 연동
- [ ] ~~GCP SQL Instance 연동하기~~
- [x] 버튼 생성 및 라우팅 수정
- [x] 다른 컴포넌트로 데이터 전달하기

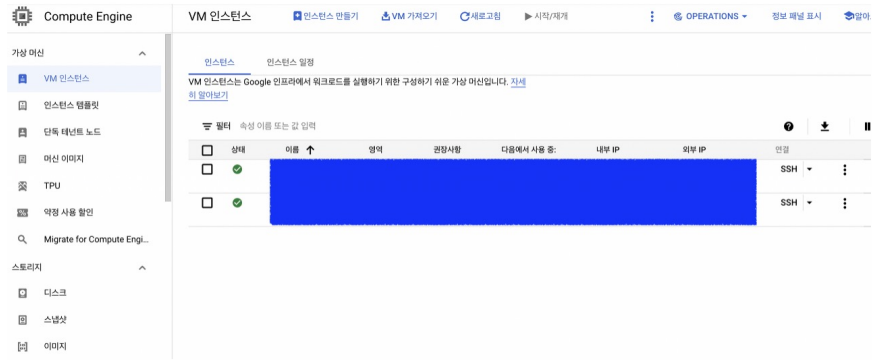
# 인스턴스 생성 및 연동

## 월요일

- 진행한 것

### 1. 인스턴스 생성 및 연결

오늘은 GCP에서 Compute Engine을 생성하여, 맥에서 접속할 수 있도록 세팅하는 일을 진행하였다.




한 인스턴스가 이상하여 총 두개를 만들었다.

위에서 생성한 인스턴스를 맥에서 원격 접속하기 위해 rsa key를 생성해주는 작업을 수행하였다.
* 참고 : https://ruuci.tistory.com/6

키를 생성하여 맥에서 접속 가능하도록 만들었고, 인스턴스의 외부 IP를 사용해서 ssh로 접속을 성공하였다.

### 2. 도커 세팅

세팅한 인스턴스에서 도커를 세팅하기로 하였다.

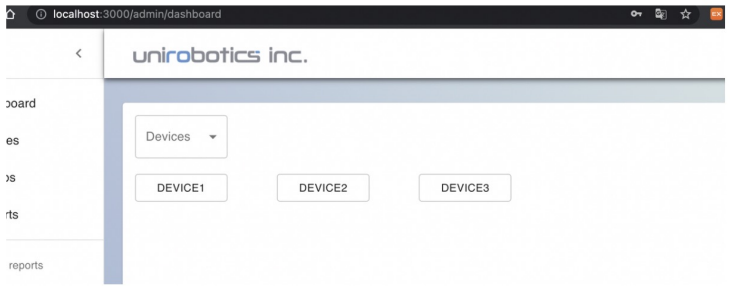
도커가 아예 설치되어있지 않아, 도커 설치 -> 도커 유저 승인 -> 도커 실행 테스트를 순서대로 수행하였다.
* 도커 설치 참고 : https://ahniverson.tistory.com/14
* 도커 실행 에러 해결 참고 : https://github.com/occidere/TIL/issues/116

도커 설치도 성공!
(ip가 어디서 유출될지 몰라 캡쳐는 생략하였다.)

## 버튼 생성 및 라우팅 수정

### 수요일

- 진행한 것




지금 에러에러

**현재 문제**

위 사진에서 버튼을 누르면 다른 주소로 라우팅이 돼야하는데, 계속 /admin/dashboard로 새로고침 됨

문제가 있는 코드 부분은 아래와 같음

```
  <Route path="/admin/dashboard" exact component={Dashboard} />
<Route path="/admin/devices" exact component={Devices} />
<Route path="/admin/groups" exact component={Groups} />
<Route exact path="/admin/groups">
<Groups
//user_id={user_id}
/>
</Route>
<Route path="/admin/reports" exact component={Reports} />
<Redirect from="/admin" to="/admin/dashboard" />
```

아마 Route와 Redirect 부분이 잘못 맞물려 그런 것 같았다.

그래서 차라리 버튼에 따라 url을 다르게 주지 말고, 일단은 props로 줘야겠다는 생각이 들었다.

**먼저 해야할 일(아직 해결까진 아님)**

```
export default function Groups() {
    const classes = useStyles();
    const [deviceCate, setDeviceCate] = React.useState('');
    const fixedHeightPaper = clsx(classes.paper, classes.fixedHeight);

    const handleChange = (event) => {
        setDeviceCate(event.target.value);
    };

    return (
        <>
            {/* for test */}
            <Grid item>
            <FormControl variant="outlined" className={classes.formControl}>
                <InputLabel id="demo-simple-select-outlined-label">Devices</InputLabel>
                <Select
                    labelId="demo-simple-select-outlined-label"
                    id="demo-simple-select-outlined"
                    value={deviceCate}
                    onChange={handleChange}
                    label="Devices"
                >
```

1

1. 먼저 위처럼 Select에서 값이 바뀌면 handleChange 함수를 통해 "deviceCate"라는 state를 업데이트

```
{DeviceList.map((data) => {
    if (data.deviceCate === "deviceCate") {
        return (
            <Grid item xs={2}>
                <Button variant="outlined" size="medium" className={classes.deviceButton}
                    key={data.id} component={Link} to={data.path}>{data.deviceName}</Butt
            </Grid>
        )
    }
})}
```

2. map안에 조건문을 넣어 가져온 data.deviceCate가 1번에서 업데이트해 준 deviceCate라는 state와 같아야만 보여주도록 수정

이제 select별로 버튼들을 잘 가져오는 것을 볼 수 있다.

내일은 버튼별로 다른 값을 가리키도록 바꾸겠다!

# 다른 컴포넌트로 데이터 전달하기

## 목요일

수요일에 한 것과 이어서, 작업을 진행하였다.

**위 문제 해결 (결론)**
구조를 요약해서 말하자면,
상위 컴포넌트 : DashBoard > 하위 컴포넌트 : Device, Chart 이다.
그래서 이제 Device -> Chart로 버튼을 통해 선택한 값을 넘겨주기 위해서는 상위 컴포넌트인 DashBoard를 거쳐야한다.

이를 위해 props 함수를 사용하였다.

```
const selectedDevice = (event) => {
    props.selectDevice(event.target.innerHTML);
    console.log(event.target.innerHTML)
};
```

위 함수는 버튼을 누르면 실행되는 함수이다.
안에 코드를 보면 props.selectDevice를 통해 selectDevice라는 이름의 props 함수를 통해 'event.target.innerHTML' 파라미터를 전달해주었다.

```
export default function Dashboard() {
  const classes = useStyles();
  const [selectedDevice, setSelectedDevice] = React.useState('');
  const fixedHeightPaper = clsx(classes.paper, classes.fixedHeight);

  return (
    <>
      <Grid container spacing={2}>

        <Grid item xs={12}>
          <Paper className={fixedHeightPaper}>
            <Devices selectDevice={
            function(_selectedDevice){
              setSelectedDevice(_selectedDevice)
            }.bind(this)}></Devices>
          </Paper>
        </Grid>
```

상위 컴포넌트 파일인 DashBoard.js에 있는 Device 컴포넌트 선언 부분이다.

여기서 selectDevice 함수를 정의해주었는데, 받아오는 파라미터를 가지고, selectedDevice라는 state(지금 드래그한 부분)을 업데이트 해주고 있다.
*리액트에서는 state를 업데이트해주기위해 Hooks라는 것을 사용하여, 해당 state만을 업데이트하는 함수를 지정해줄 수 있는데, 여기서 selectedDevice state를 업데이트 해주는 것이 바로 setSelectedDevice이다.

상위 컴포넌트인 DashBoard가 가지고 있는 selectedDevice 값을 이제 Chart로 넘겨줘야한다.

```
: <Chart title={selectedDevice}/>}
```

상위 컴포넌트 파일인 DashBoard.js에 있는 Chart 컴포넌트 선언 부분이다.

여기서 selectedDevice를 title이라는 props로 넘겨줄 수 있었다.

ARGOS

# Full Stack Challenge | For Week 3

## *To Do List*

- [ ] device 테이블 만들기
- [ ] GCP SQL Instance 연동하기 (진짜 해야하는데…)
- [ ] backend 파일 재구조화
- [ ] device 테이블 테스트하기


# THANK YOU

컴퓨터공학과 201802106 **서연주**