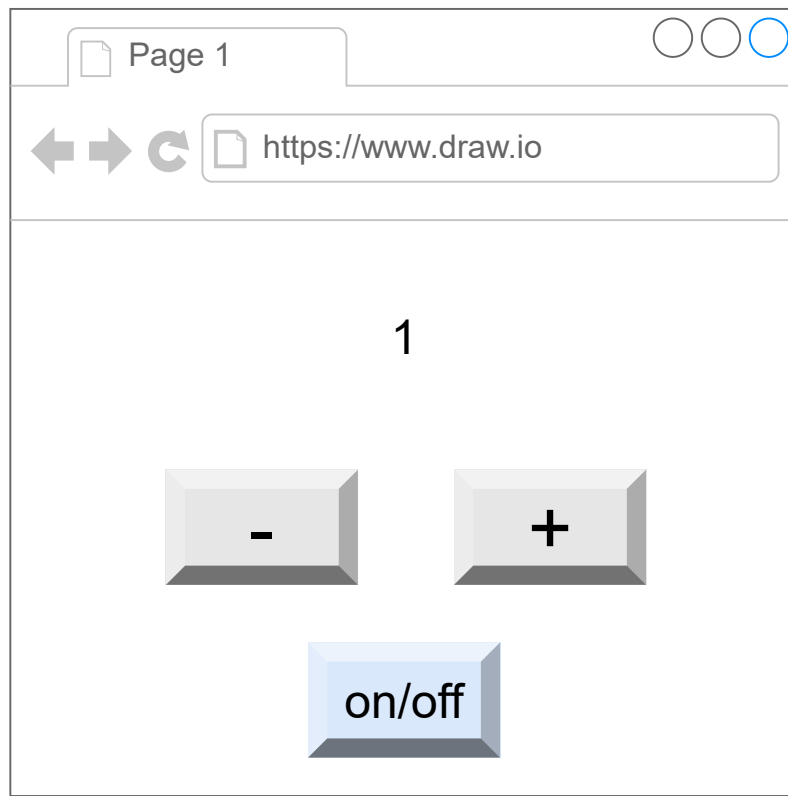


## TDD를 이용해서 만들 앱 소개



+, -, 그리고 on/off 버튼이 있습니다. 거기서 +를 누르면 숫자가 올라가고 -를 누르면 내려갑니다. 그리고 on/off 버튼(푸른색)을 누르면 +, - 버튼이 작동을 안하고 색깔이 변하는 간단한 앱을 만들어 보겠습니다.

```
import { render, screen, fireEvent } from "@testing-library/react";
import App from "../App";

test("the counter starts at 0", () => {
});

test("minus button has correct text", () => {
});

test("plus button has correct text", () => {
});

test("When the + button is pressed, the counter changes to 1", () => {
});

test("When the - button is pressed, the counter changes to -1", () => {
});

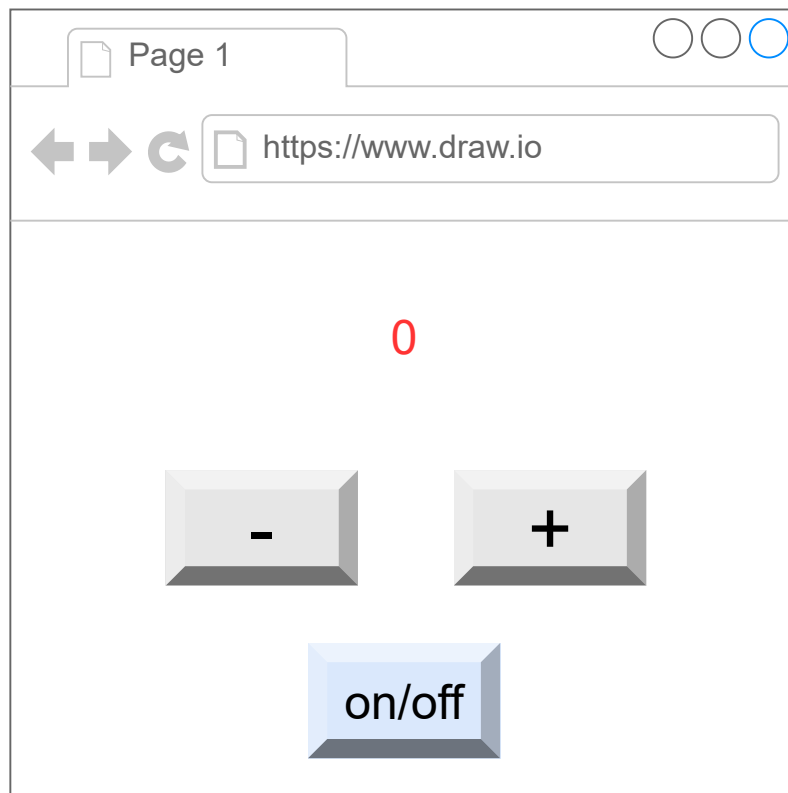
test("on/off button has blue color", () => {
});

test("Prevent the -, + button from being pressed when the on/off button is
```

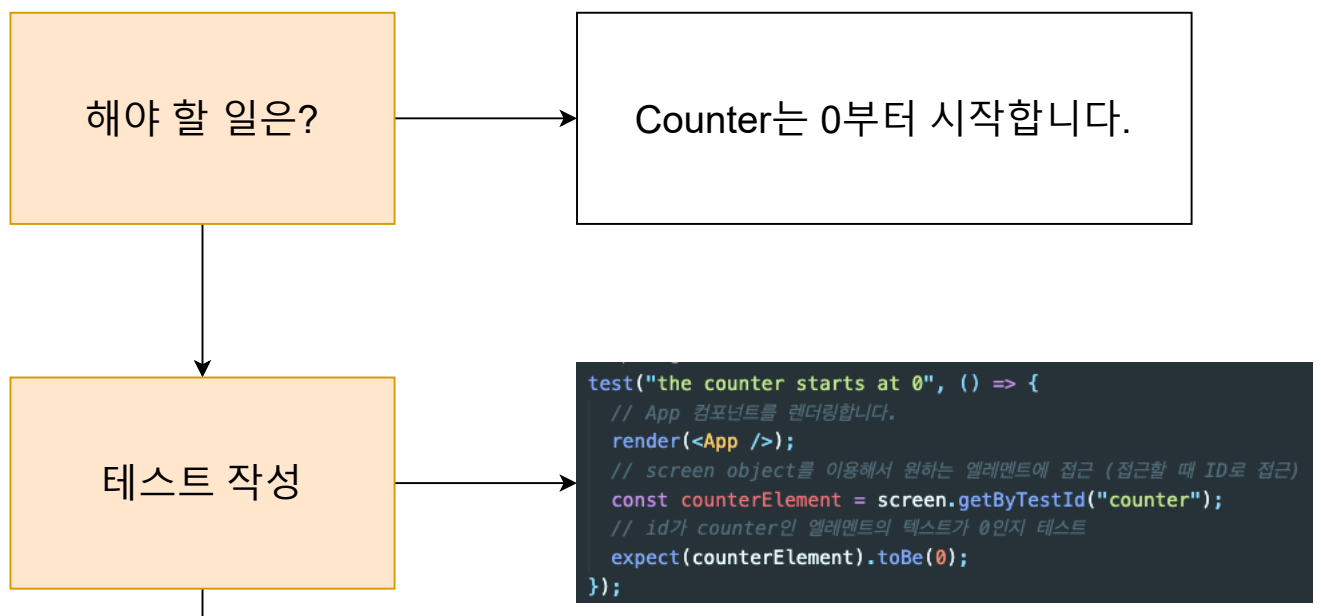
```
clicked", () => {  
});
```

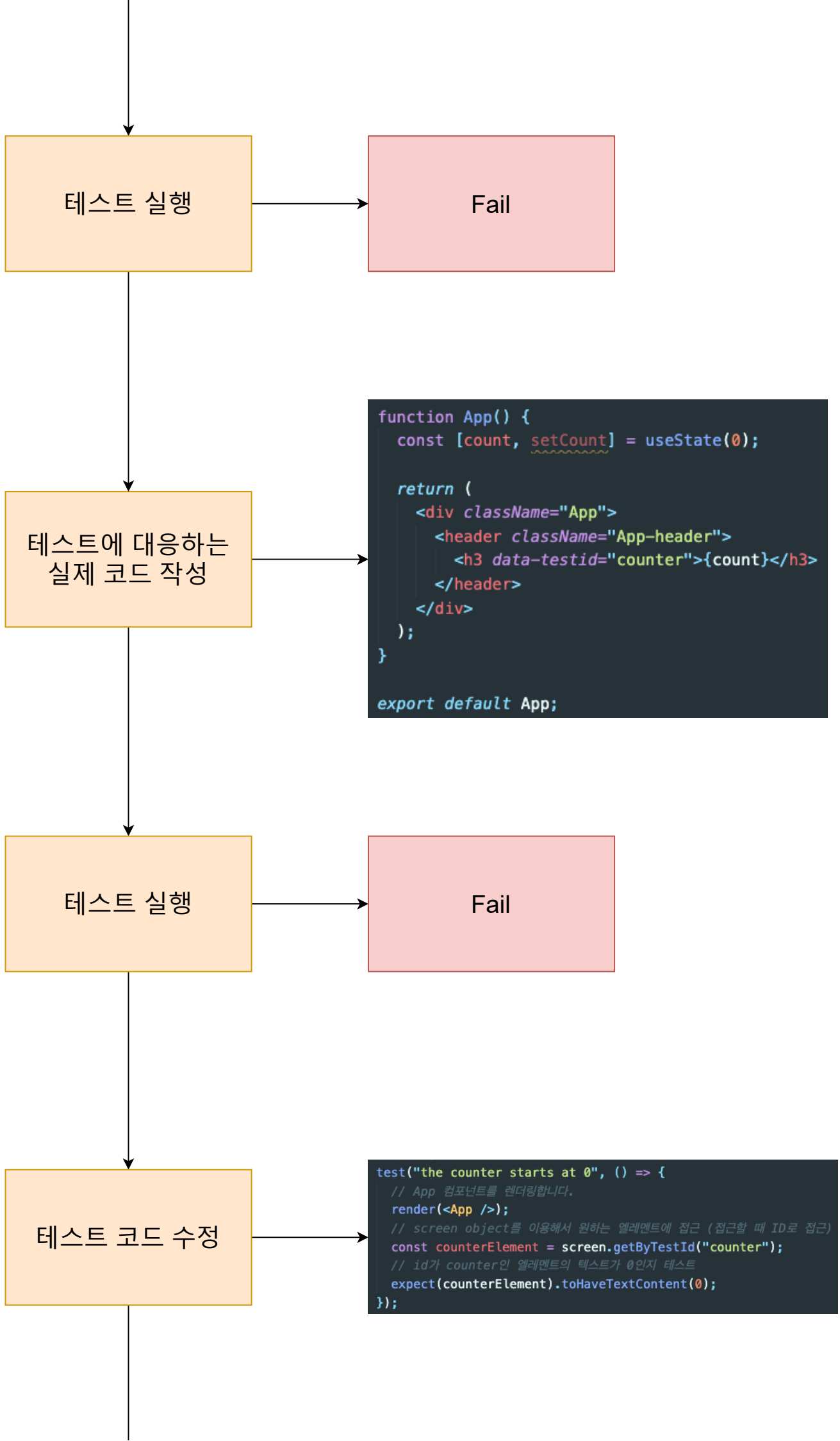
## 앱 만들기 시작

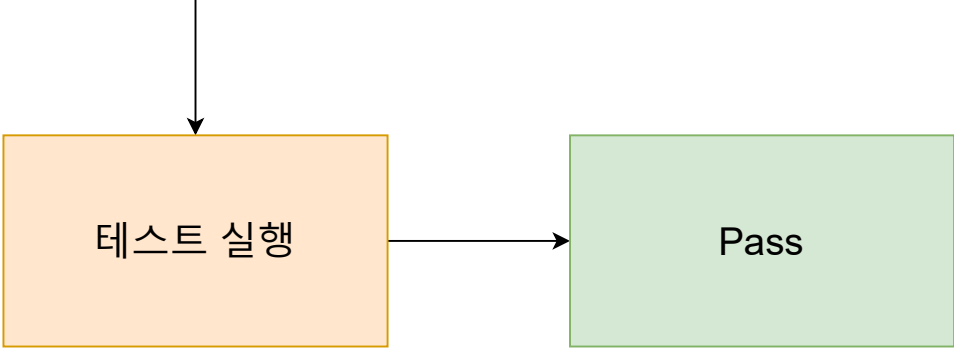
테스트 주도 개발을 할 것이기 때문에 먼저 테스트 코드 부터 작성해보겠



## Counter 생성

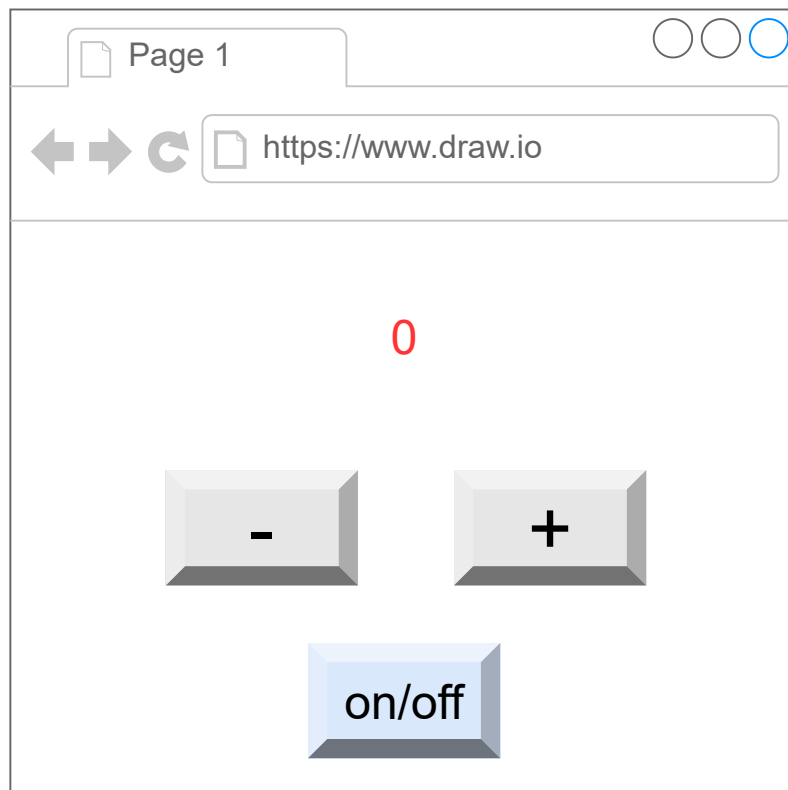






## 플러스, 마이너스 버튼 생성

카운터를 올리고 내릴 수 있는 버튼을 생성해보겠습니다.

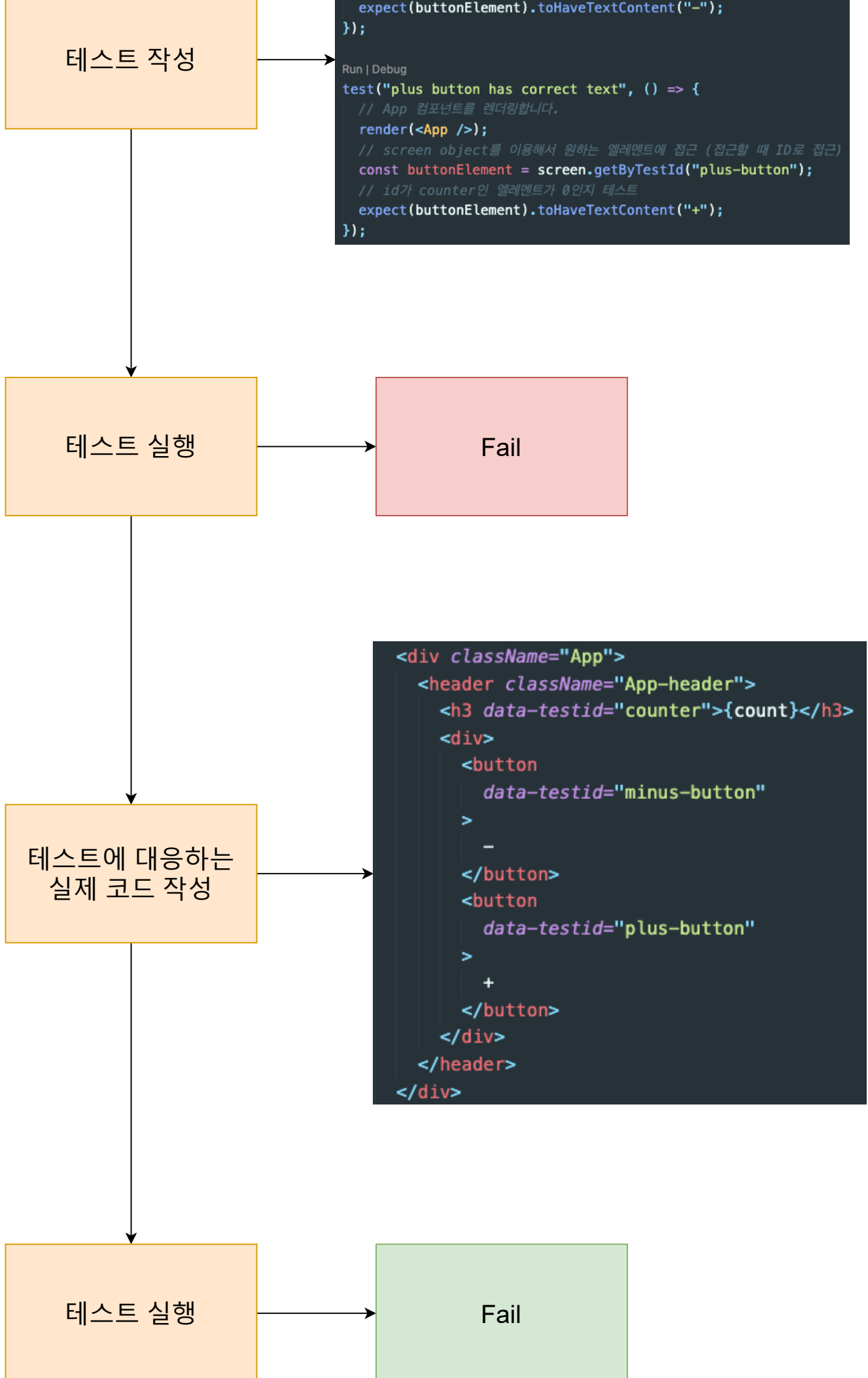


## 버튼 생성

해야 할 일은?

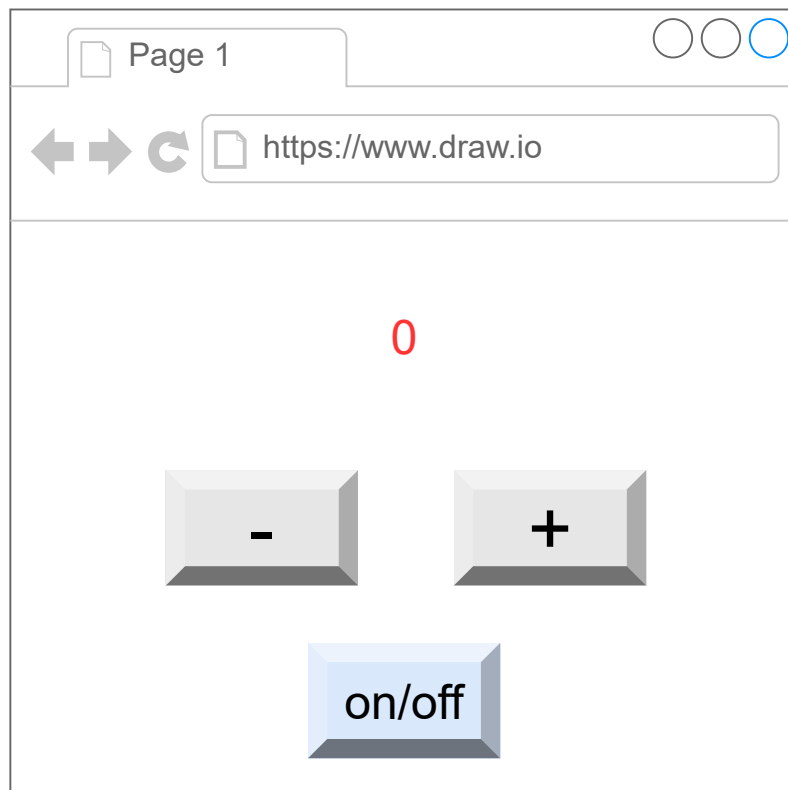
+, - 버튼 두개를 생성합니다.

```
test("minus button has correct text", () => {  
  // App 컴포넌트를 렌더링합니다.  
  render(<App />);  
  // screen object를 이용해서 원하는 요소에 접근 (접근할 때 ID로 접근)  
  const buttonElement = screen.getByTestId("minus-button");  
  // id가 counter인 요소가 0인지 테스트
```



플러스, 마이너스 버튼 기능 넣기(fire event)

카운터를 올리고 내릴 수 있는 버튼의 기능을 넣어서 카운터를 변화시켜주겠습니다.



## FireEvent API

유저가 발생시키는 액션(이벤트)에 대한 테스트를 해야 하는 경우 사용합니다.

<https://testing-library.com/docs/dom-testing-library/api-events/>

## 버튼 생성

해야 할 일

+ 버튼을 누르면



해야 할 일은?

카운터가 1로 변하게 됩니다.

테스트 작성

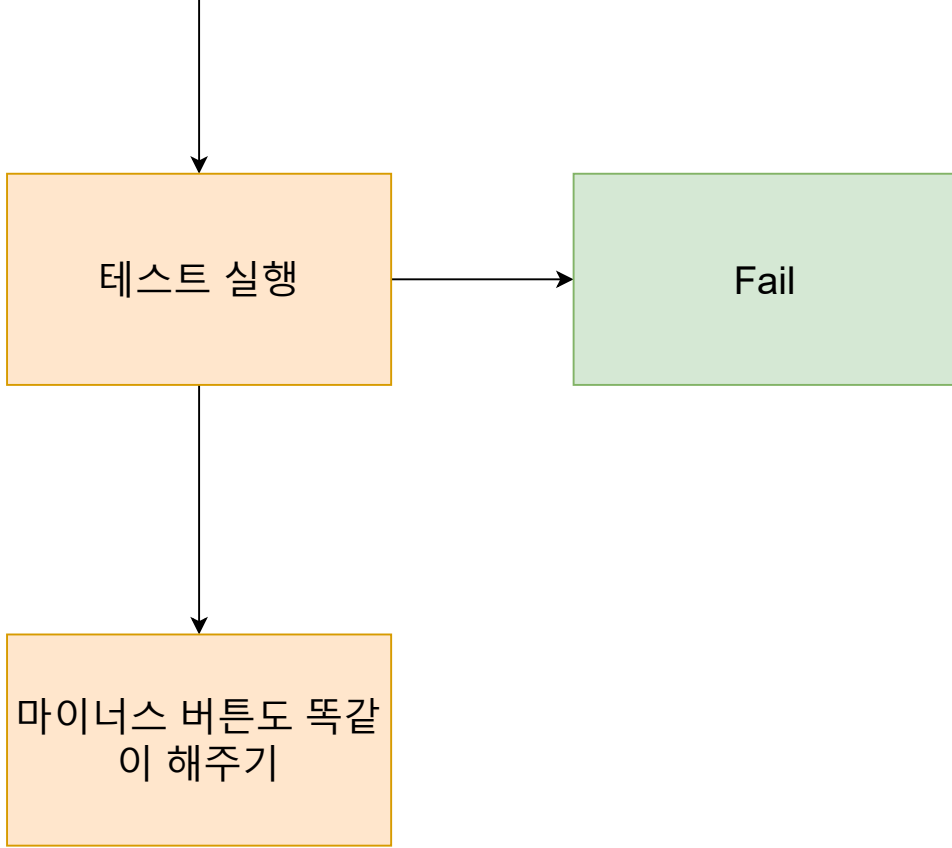
```
test("When the + button is pressed, the counter changes to 1", () => {  
  // App 컴포넌트를 렌더링합니다.  
  render(<App />);  
  // screen object를 이용해서 원하는 엘리먼트에 접근 (접근할 때 ID로 접근)  
  const buttonElement = screen.getByTestId("plus-button");  
  // click plus button  
  fireEvent.click(buttonElement);  
  // 카운터가 0에서 +1 돼서 1이 됩니다.  
  const counterElement = screen.getByTestId("counter");  
  expect(counterElement).toHaveTextContent(1);  
});
```

테스트 실행

Fail

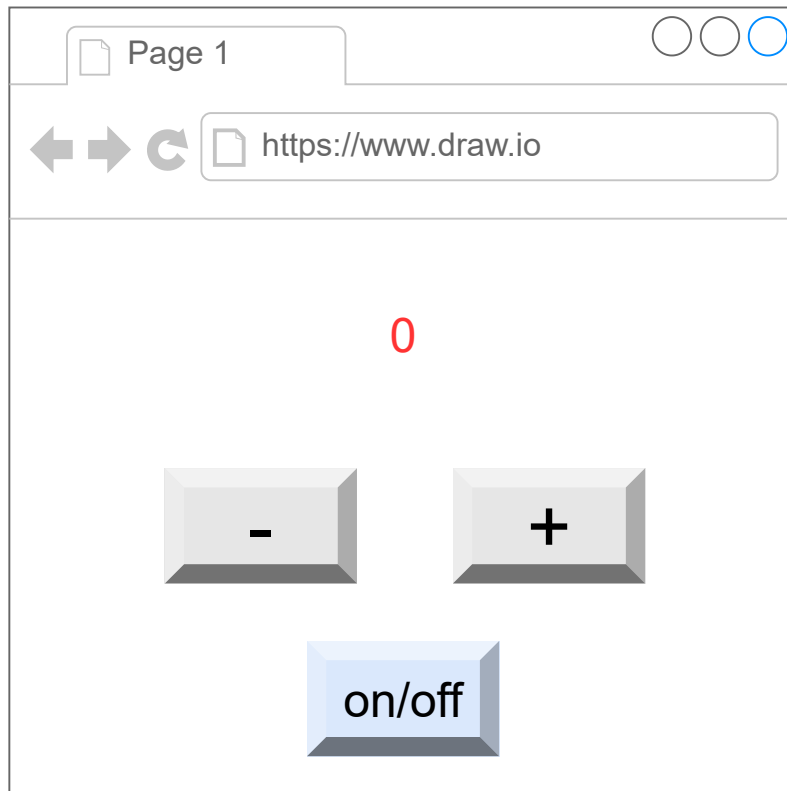
테스트에 대응하는  
실제 코드 작성

```
<button  
  data-testid="plus-button"  
  onClick={() => setCount((count) => count + 1)}  
>  
  +  
</button>
```



## on/off 버튼 만들기(toHaveStyle)

on/off 버튼을 만들어보겠습니다



## on/off 버튼 생성

해야 할 일은?

on/off 버튼을 만드는데 이 버튼은 파란색으로 스타일을 주겠습니다.

<https://github.com/testing-library/jest-dom>

```
test("on/off button has blue color", () => {  
  // App 컴포넌트를 렌더링합니다.  
  render(<App />);  
})
```

테스트 작성

```
render(<App />);  
// screen object를 이용해서 원하는 엘리먼트에 접근 (접근할 때 ID로 접근)  
const buttonElement = screen.getByTestId("on/off-button");  
// on/off 버튼 색깔을 블루색으로 ...  
expect(buttonElement).toHaveStyle({ backgroundColor: "blue" });  
});
```

테스트 실행

Fail

테스트에 대응하는  
실제 코드 작성

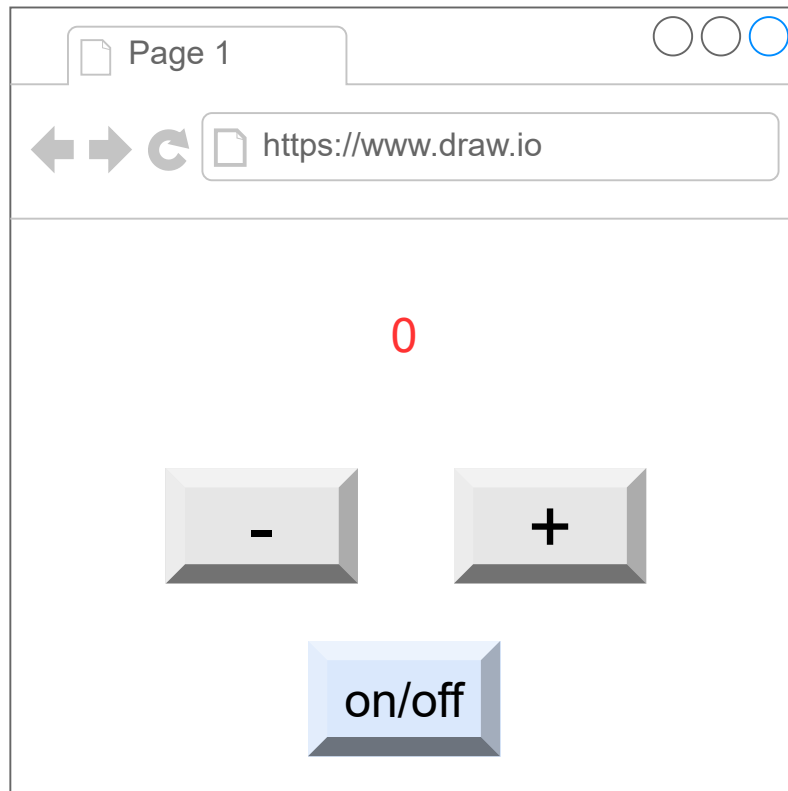
```
<div>  
  <button  
    style={{ backgroundColor: "blue" }}  
    data-testid="on/off-button"  
  >  
    on/off  
  </button>  
</div>
```

테스트 실행

Fail

on/off 버튼 클릭 시 버튼 disabled

on/off 버튼을 클릭 시 -,+ 버튼을 disabled 시켜보겠습니다



## on/off 버튼 생성

해야 할 일은?

on/off 버튼을 클릭할 때 -,+ 버튼을  
못누르게 막기

```
test("Prevent the -,+ button from being pressed when the on/off button is clicked", () => {  
  // App 컴포넌트를 렌더링합니다.  
  render(<App />);  
  // screen_object를 이용해서 원하는 엘리먼트에 접근 (접근할 때 ID로 접근)
```

테스트 작성

```
// screen object를 이용해서 원하는 엘리먼트에 접근 (접근할 때 ID로 접근)  
const onOffButtonElement = screen.getByTestId("on/off-button");  
// click onOffButtonElement button  
fireEvent.click(onOffButtonElement);  
// screen object를 이용해서 원하는 엘리먼트에 접근 (접근할 때 ID로 접근)  
const plusButtonElement = screen.getByTestId("plus-button");  
expect(plusButtonElement).toBeDisabled();  
});
```

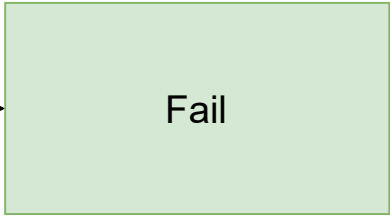
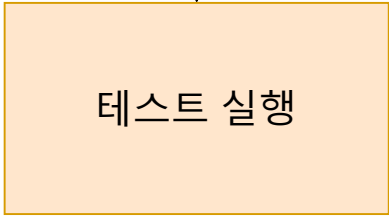
테스트 실행

Fail

테스트에 대응하는  
실제 코드 작성

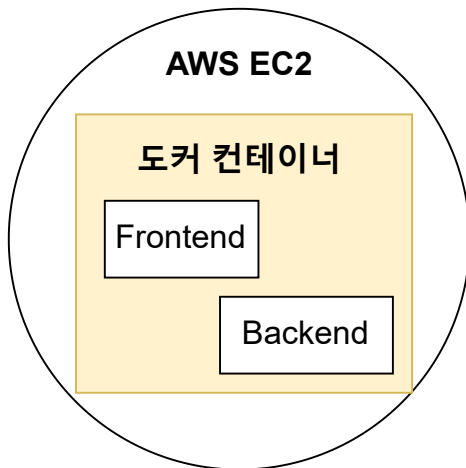
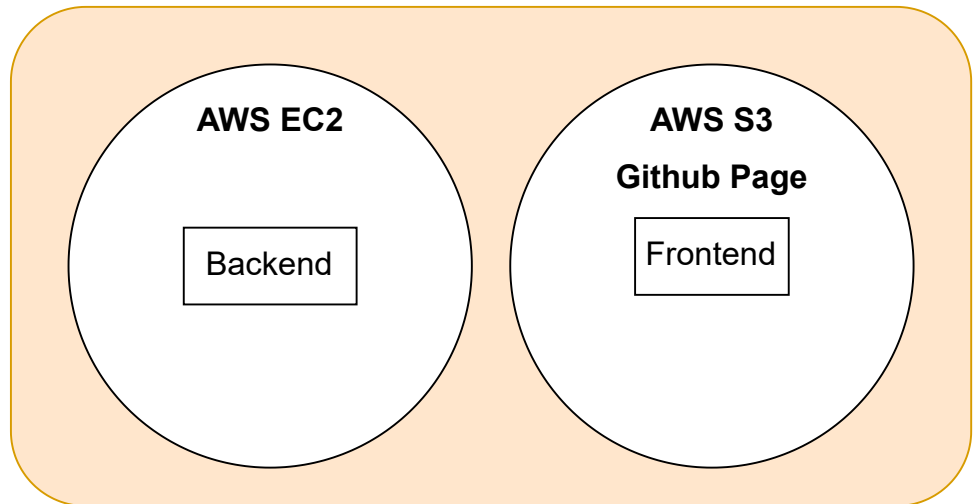
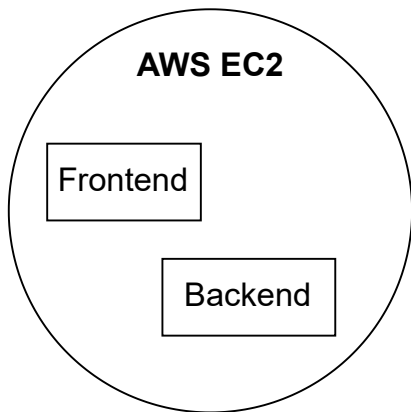
```
function App() {  
  const [count, setCount] = useState(0);  
  const [disabled, setDisabled] = useState(false);  
  
  return (  
    <div className="App">  
      <header className="App-header">  
        <h3 data-testid="counter">{count}</h3>  
        <div>  
          <button  
            data-testid="minus-button"  
            disabled={disabled}  
            onClick={() => setCount((count) => count - 1)}  
          >  
            -  
          </button>{" "  
          <button  
            data-testid="plus-button"  
            disabled={disabled}  
            onClick={() => setCount((count) => count + 1)}  
          >  
            +  
          </button>  
        </div>  
        <div>  
          <button  
            style={{ backgroundColor: "blue" }}  
            data-testid="on/off-button"  
            onClick={() => setDisabled((prev) => !prev)}  
          >  
            on/off  
          </button>  
        </div>  
      </div>  
    )  
  );  
}
```

```
        </div>  
      </header>  
    </div>  
  );  
}
```

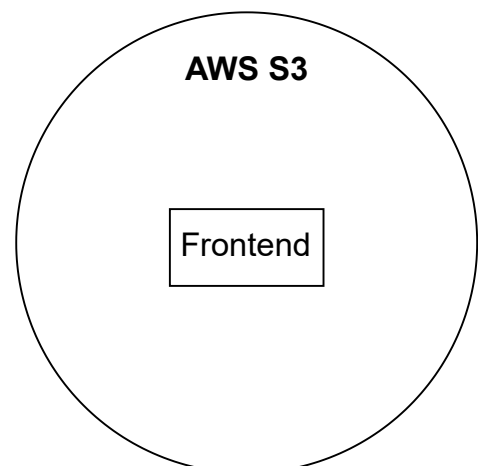
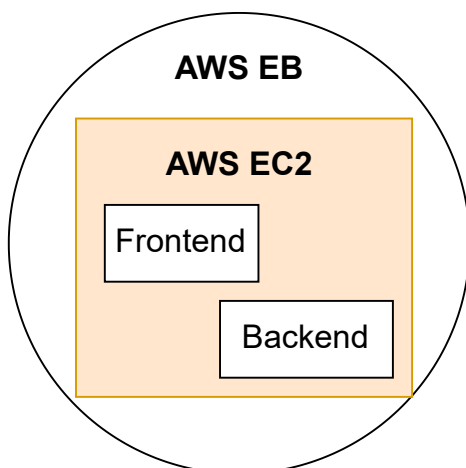
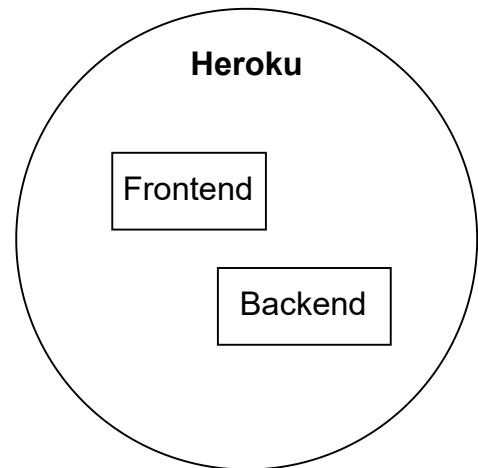


# Github Action을 이용한 AWS S3로 앱 자동 배포하기

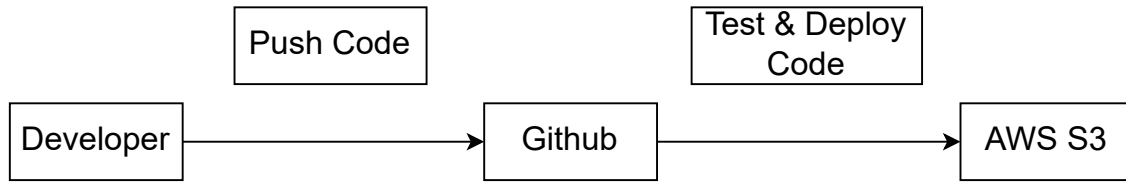
## 앱을 배포하는 방법



AWS  
Azure  
Google cloud  
Digital Ocean  
....









Zenkins  
Circle CI  
Travis CI  
Github

## 저장소 생성

Owner \* Repository name \*

 jaewonhimnae / react-github-action 

Great repository names are short and memorable. Need inspiration? How about [fantastic-memory?](#)

Description (optional)

## 저장소 연결

### ...or push an existing repository from the command line

```
git remote add origin https://github.com/jaewonhimnae/react-github-action-test.git
git branch -M main
git push -u origin main
```

## workflow 생성

### Suggested for this repository


Node.js

By GitHub Actions



Build and test a Node.js project with npm.

Configure

JavaScript 

```
name: Node.js CI

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:

    runs-on: ubuntu-latest

    strategy:
      matrix:
        node-version: [12.x, 14.x, 16.x]
        # See supported Node.js release schedule at https://node.dev/en/versions/

    steps:
      - uses: actions/checkout@v2
      - name: Use Node.js ${{ matrix.node-version }}
        uses: actions/setup-node@v2
        with:
          node-version: ${{ matrix.node-version }}
          cache: 'npm'
      - run: npm ci
      - run: npm run build --if-present
      - run: npm test
```

main 브랜치에 push 되었을 때  
jobs에 있는것 하게 됩니다.

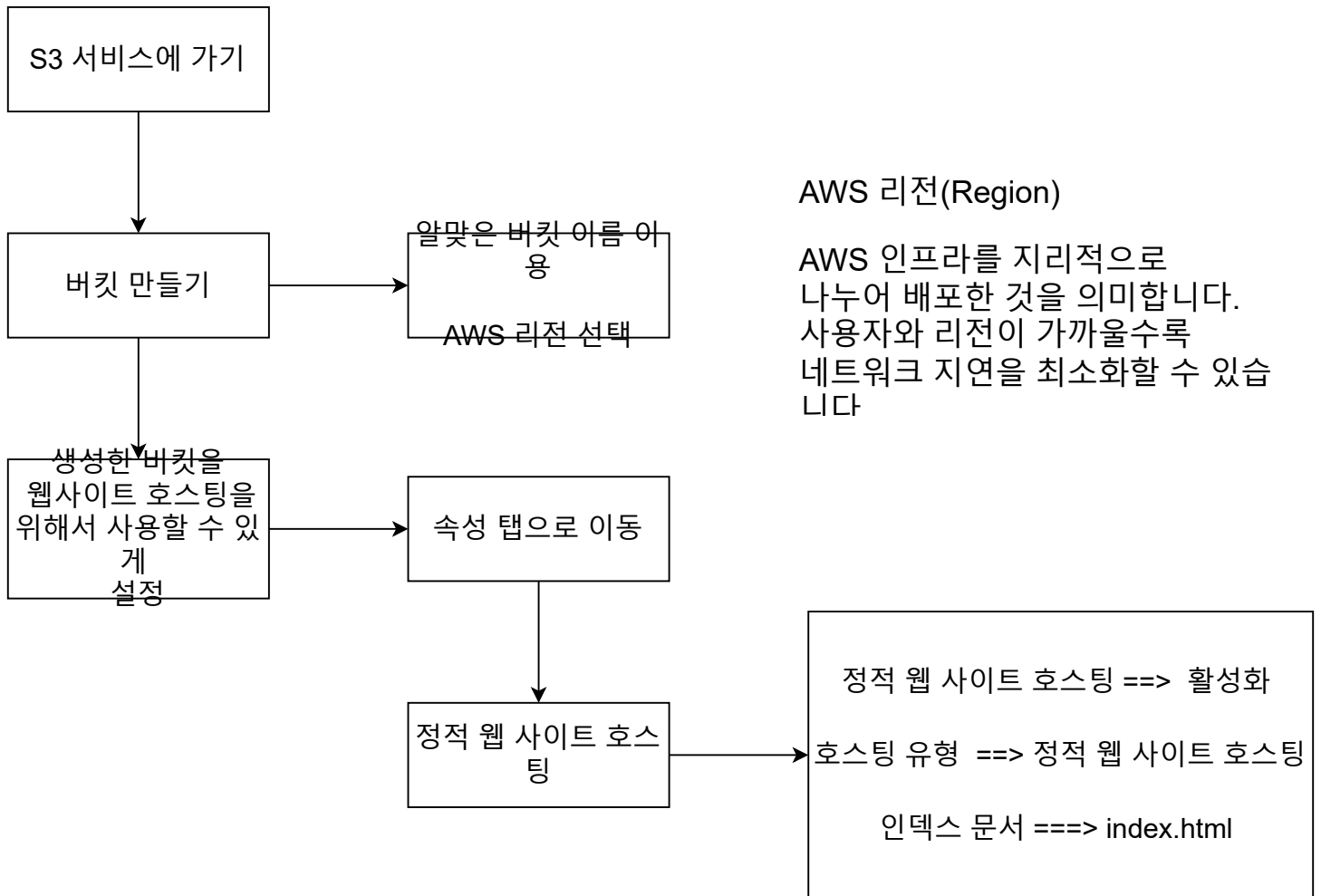
노드 버전 12에서 아래 steps가 진행되  
고 , 14에서 그리고 16에서...

--if -present

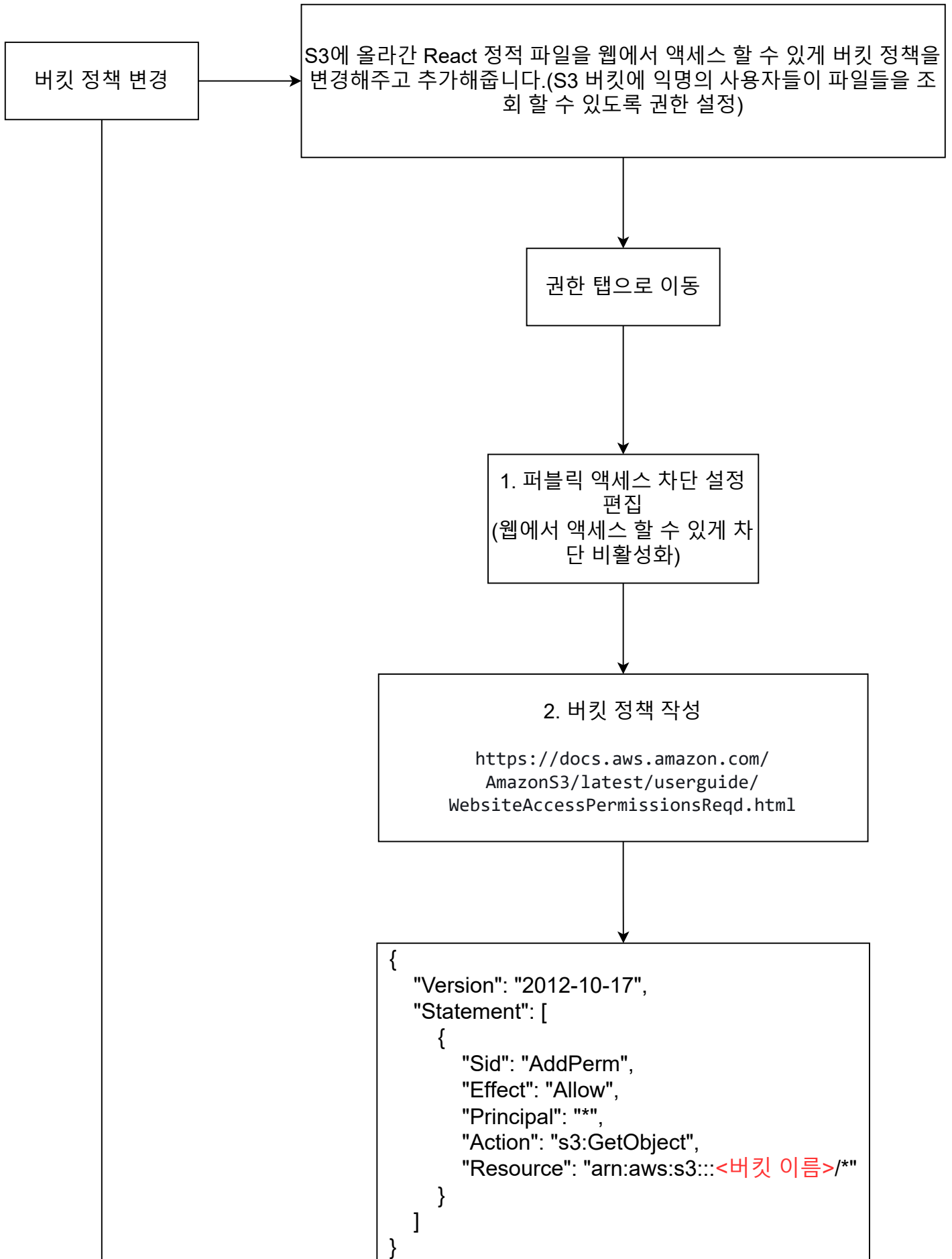
빌드 스크립트가 있을 때만  
실행

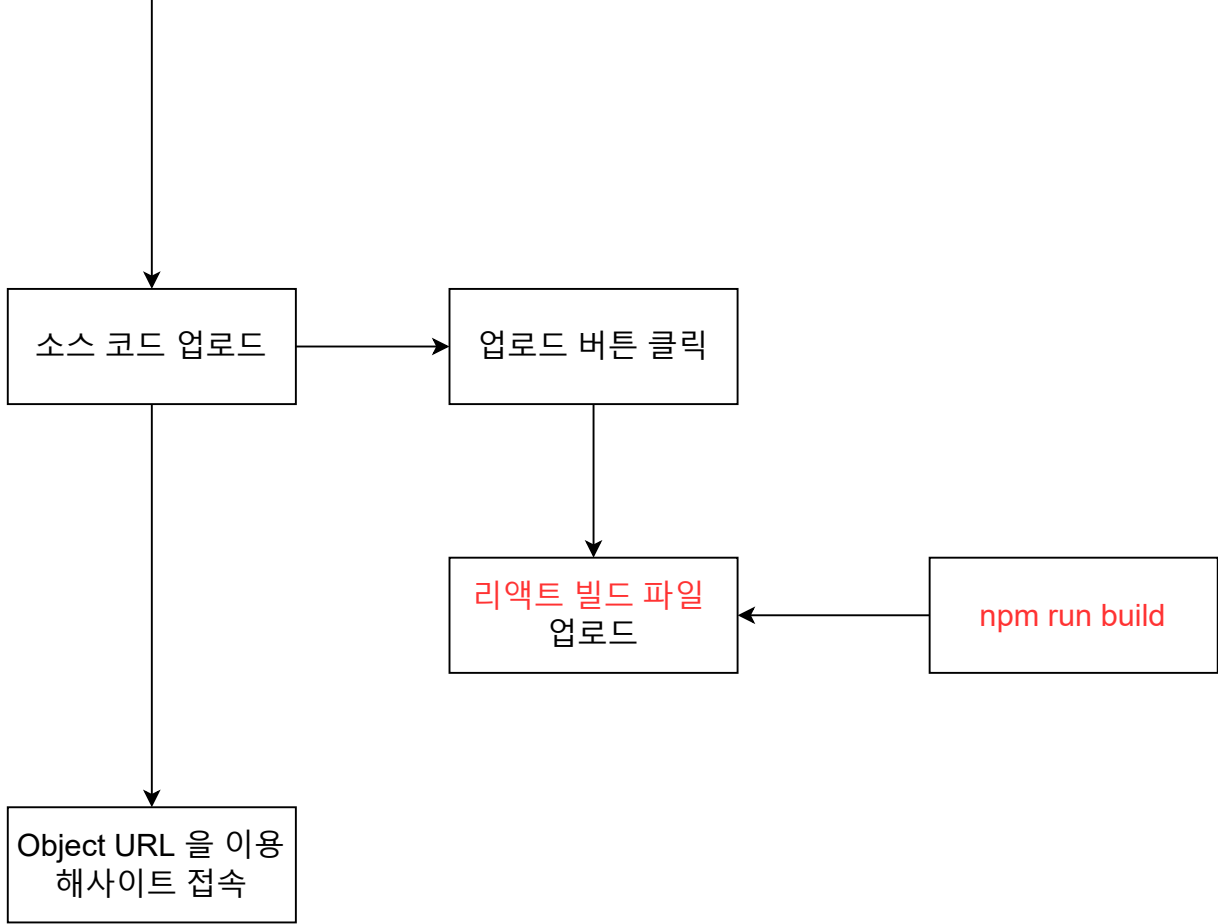
## 앱 배포를 위한 AWS S3 버킷 생성하기

### AWS S3 서비스를 이용해서 애플리케이션을 배포해주겠다



# AWS S3 버킷 설정 및 애플리케이션 배포하기





## S3로 앱 자동 배포를 위한 yml 파일 완성하기

<https://github.com/awact/s3-action>

```
name: Sync S3 Bucket
on: push

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@master
      - uses: awact/s3-action@master
        with:
          args: --acl public-read --follow-symlinks --delete
        env:
          SOURCE_DIR: './public'
          AWS_REGION: 'us-east-1'
          AWS_S3_BUCKET: ${ secrets.AWS_S3_BUCKET }
          AWS_ACCESS_KEY_ID: ${ secrets.AWS_ACCESS_KEY_ID }
          AWS_SECRET_ACCESS_KEY: ${ secrets.AWS_SECRET_ACCESS_KEY }
```

```
env:
  AWS_S3_BUCKET: ${ secrets.AWS_S3_BUCKET }
  AWS_ACCESS_KEY_ID: ${ secrets.AWS_ACCESS_KEY_ID }
  AWS_SECRET_ACCESS_KEY: ${ secrets.AWS_SECRET_ACCESS_KEY }
  AWS_REGION: 'ap-northeast-2' # optional: defaults to us-east-1
  SOURCE_DIR: 'build' # optional: defaults to entire repository
```

### Actions secrets / New secret

Name

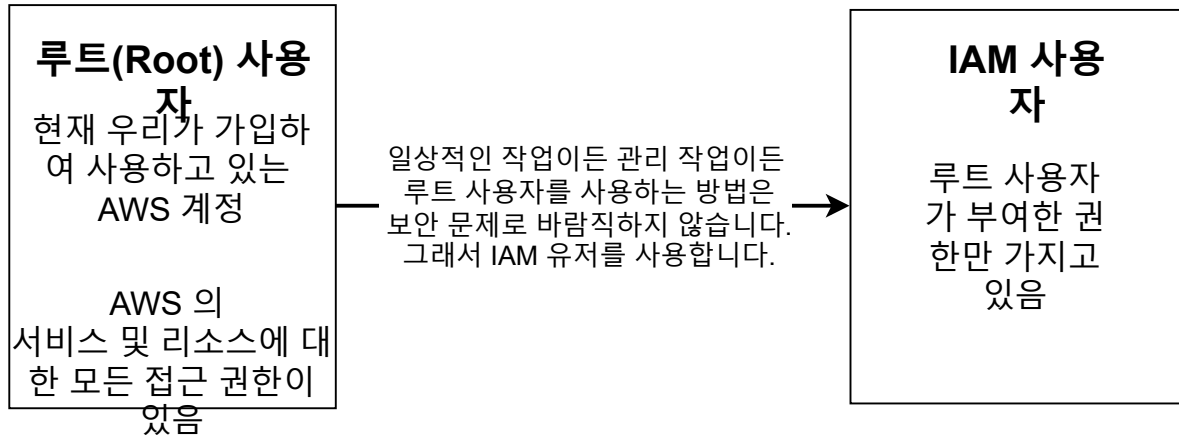
AWS\_S3\_BUCKET

Value

action-test-s3

# IAM은 무엇인가 ? (Identity and Access Management)

AWS 리소스에 대한 액세스를 안전하게 제어할 수 있는 웹 서비스입니다. IAM을 사용하여 리소스를 사용하도록 인증(로그인) 및 권한 부여(권한 있음)된 대상을 제어합니다.



## 사용자 추가

### 사용자 세부 정보 설정

동일한 액세스 유형 및 권한을 사용하여 한 번에 여러 사용자를 추가할 수 있습니다. [자세히 알아보기](#)

사용자 이름\*

action-test-user

+ 다른 사용자 추가

