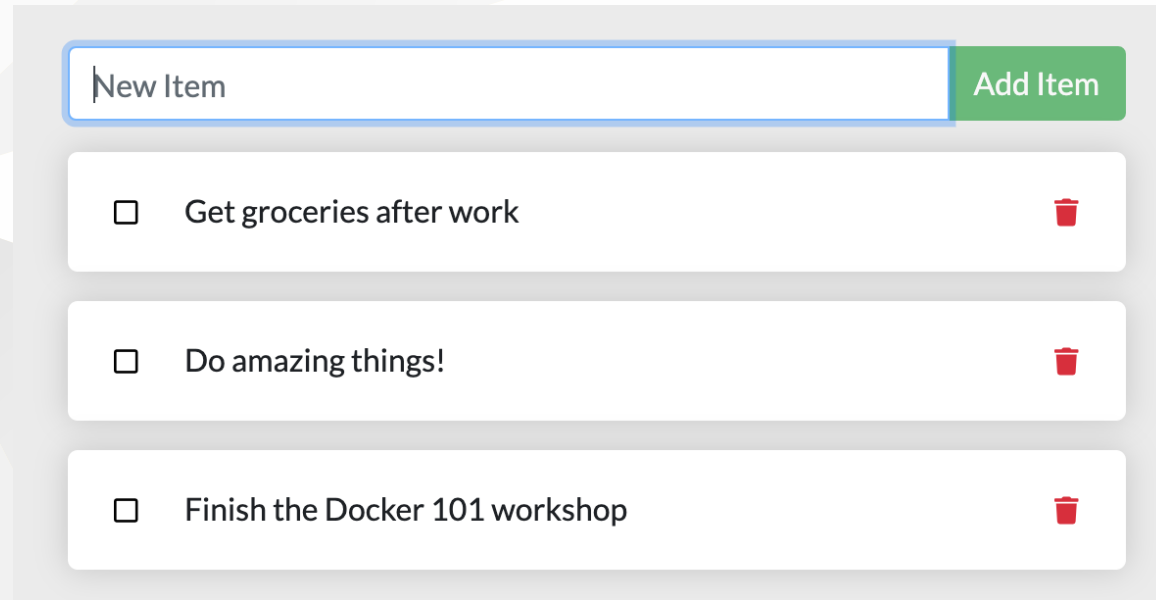


[Hands-on] 01. Docker intro

첫 번째 실습입니다.

그냥 일단 무작정 따라해보세요.

자세한건 천천히 알아볼게요.



The screenshot shows a web application interface for a To-Do List Manager. At the top, there is a text input field with the placeholder text "New Item" and a green button labeled "Add Item". Below this, there is a list of three items, each in a white box with a light gray border. Each item consists of a checkbox, the text of the task, and a red trash icon on the right. The tasks are: "Get groceries after work", "Do amazing things!", and "Finish the Docker 101 workshop".

Item	Action
<input type="checkbox"/> Get groceries after work	
<input type="checkbox"/> Do amazing things!	
<input type="checkbox"/> Finish the Docker 101 workshop	

이번 과정에서 사용할 샘플 애플리케이션인 **ToDo List Manager** 입니다.

할 일 목록(Item)을 등록하고 관리할 수 있는 간단한 애플리케이션입니다.

Docker & Kubernetes - [Hands-on] 01. Docker intro

먼저 필요한 소스코드를 Github에서 다운로드 합니다.

```
ubuntu@ip-10-0-1-14:~$ git clone https://github.com/JungSangup/todo_list_manager.git app
Cloning into 'app'...
remote: Enumerating objects: 54, done.
remote: Counting objects: 100% (54/54), done.
remote: Compressing objects: 100% (49/49), done.
remote: Total 54 (delta 4), reused 54 (delta 4), pack-reused 0
Receiving objects: 100% (54/54), 1.67 MiB | 4.76 MiB/s, done.
Resolving deltas: 100% (4/4), done.
```

명령어 : `git clone https://github.com/JungSangup/todo_list_manager.git app`

소스코드 준비가 됐으면 `app` 디렉토리로 이동해서 어떤 파일들이 있는지 살펴볼까요?

```
ubuntu@ip-10-0-1-14:~$ cd app
ubuntu@ip-10-0-1-14:~/app$ ls -al
total 204
drwxrwxr-x 5 ubuntu ubuntu 4096 Jun 17 02:54 .
drwxr-x--- 8 ubuntu ubuntu 4096 Jun 17 02:54 ..
drwxrwxr-x 8 ubuntu ubuntu 4096 Jun 17 02:54 .git
-rw-rw-r-- 1 ubuntu ubuntu 105 Jun 17 02:54 Dockerfile
-rw-rw-r-- 1 ubuntu ubuntu 626 Jun 17 02:54 package.json
drwxrwxr-x 4 ubuntu ubuntu 4096 Jun 17 02:54 spec
drwxrwxr-x 5 ubuntu ubuntu 4096 Jun 17 02:54 src
-rw-rw-r-- 1 ubuntu ubuntu 179361 Jun 17 02:54 yarn.lock
```

명령어 : `cd app` , `ls -al`

Docker & Kubernetes - [Hands-on] 01. Docker intro

이제 샘플 애플리케이션을 컨테이너 이미지로 만들어 보겠습니다.
아래 명령어는 도커 이미지를 만드는(build) 명령어 입니다.

```
ubuntu@ip-10-0-1-14:~/app$ docker build --tag docker-101 .
Sending build context to Docker daemon 6.489MB
Step 1/5 : FROM node:10-alpine
10-alpine: Pulling from library/node
ddad3d7c1e96: Pull complete
de915e575d22: Pull complete
7150aa69525b: Pull complete
d7aa47be044e: Pull complete
Digest: sha256:dc98dac24efd4254f75976c40bce46944697a110d06ce7fa47e7268470cf2e28
Status: Downloaded newer image for node:10-alpine
--> aa67ba258e18
Step 2/5 : WORKDIR /app
--> Running in bf92de6fa1c4
Removing intermediate container bf92de6fa1c4
--> ff7e4526409e
Step 3/5 : COPY . .
--> 027fa7ba112b
Step 4/5 : RUN yarn install --production
--> Running in 1e6469e6dd8d
yarn install v1.22.5
[1/4] Resolving packages...
[2/4] Fetching packages...
info fsevents@1.2.9: The platform "linux" is incompatible with this module.
info "fsevents@1.2.9" is an optional dependency and failed compatibility check. Excluding it from installation.
[3/4] Linking dependencies...
[4/4] Building fresh packages...
Done in 9.08s.
```

Docker & Kubernetes - [Hands-on] 01. Docker intro

```
Removing intermediate container 1e6469e6dd8d
---> d86f278413c5
Step 5/5 : CMD ["node", "/app/src/index.js"]
---> Running in 6aa4603e6f88
Removing intermediate container 6aa4603e6f88
---> 25d534982391
Successfully built 25d534982391
Successfully tagged docker-101:latest
```

명령어 : `docker build --tag docker-101 .`

뭔가 열심히 만든 것 같네요.

Download도 하고, COPY도 하고, Install도 하고...

이제 잘 만들어졌는지 볼까요?

현재 Host에 있는 이미지를 조회하는 명령어입니다.

```
ubuntu@ip-10-0-1-14:~/app$ docker images
REPOSITORY    TAG       IMAGE ID      CREATED        SIZE
docker-101    latest    25d534982391  2 minutes ago  172MB
```

명령어 : `docker images`

위 처럼 docker-101 이 보이면 성공입니다. ㄹ(´▽`*)

Docker & Kubernetes - [Hands-on] 01. Docker intro

이제 샘플 애플리케이션을 실행해 보겠습니다.

앞에서는 컨테이너 이미지를 만들었습니다.

이번에는 이 이미지를 컨테이너로 실행(run)해 보겠습니다.

실행하는 명령어는 다음과 같습니다.

```
ubuntu@ip-10-0-1-14:~/app$ docker run --detach --publish 3000:3000 docker-101
4480ffcd6fa67de20f4529cb2ccd3e0b8fba7c63fc036541c7bfd40062db2cb7
```

명령어 : `docker run --detach --publish 3000:3000 docker-101`

그리고, 잘 실행되고 있는지 볼까요?

```
ubuntu@ip-10-0-1-14:~/app$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
4480ffcd6fa6	docker-101	"docker-entrypoint.s..."	27 seconds ago	Up 26 seconds	0.0.0.0:3000->3000/tcp, :::3000->3000/tcp	youthful_noether

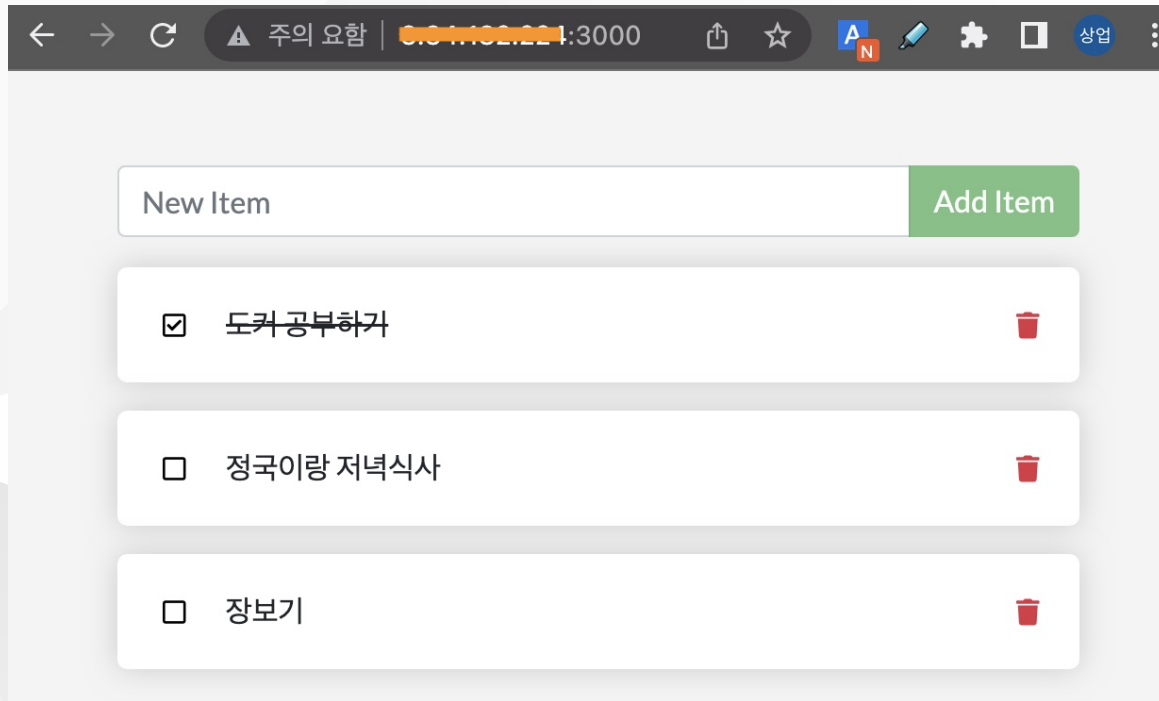
명령어 : `docker ps`

위 처럼 docker-101 이 보이면 정상입니다.

이제 실행된 애플리케이션에 접속해볼까요? 실행된 Host의 3000번 포트로 접속하면 됩니다.

- AWS EC2인 경우 인스턴스의 Public IPv4 address로 접속하면 됩니다. (e.g. <http://IP:3000/>)
- Security group의 Inbound rule에 8080번 포트에 대한 규칙이 있어야 합니다.

Docker & Kubernetes - [Hands-on] 01. Docker intro



여러분의 ToDo List Manager 애플리케이션이 잘 동작하나요?
오늘 할 일을 한 번 추가해 보세요.

축하합니다!!! (٩´▽`)๓

여러분은 방금 10분만에 똑딱 시스템 환경구축을 마쳤습니다.