

Environment Setup with Docker

2025-03-20

PintOS-Docker

- PintOS is one of the well-known simple operating system framework for 80x86, developed by Stanford.
- However, this framework relies on some outdated dependencies, that makes difficult us to compile at modern operating systems.
- We provide you a docker environment to setup for your convenience and safety compared to using outdated operating systems.

Clone & Build

- Clone repository to get Dockerfile and some extra files.

```
> git clone git@github.com:NamJeongseok/PintOS-Docker-UNIST.git
```

```
> cd PintOS-Docker-UNIST
```

```
> sudo docker build -t [image_name] .
```

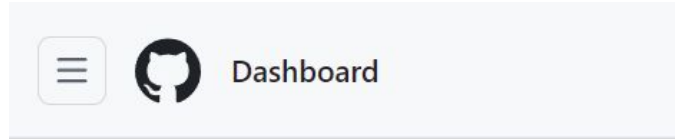
```
// It takes some time...
```

Get PintOS

```
> cd .. // Go to upper directory (or, workspace wherever you want)
> wget http://www.stanford.edu/class/cs140/projects/pintos/pintos.tar.gz
> tar xvf pintos.tar.gz
> cd pintos
> ls
src
```

Connect PintOS code w/ your GitHub repository

- Make your project repository at GitHub.



Top repositories



- Repository name should be "os + your student id". For instance, os20251234.
- If you have a team, then write both of your ids, e.g., os20251234_20251235.
- The repository should be **private**.



Private

You choose who can see and commit to this repository.

Connect PintOS code w/ your GitHub repository

- If you work as a team, one member create the repository, while another can be invited as a collaborator.
- Also, invite the TA as a collaborator to grade your submission.
 - @NamJeongseok

Connect Pintos code w/ your GitHub repository

- Now, follow the instruction provided by GitHub.

```
// At your Pintos directory (not docker directory!)
```

```
> git init
```

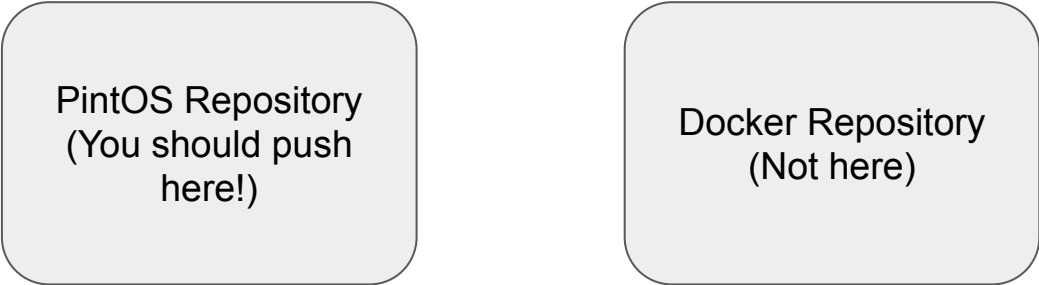
```
> git remote add origin git@github.com: [username]/[repo_name].git
```

```
> git branch -M main
```

```
> git push -u origin main
```

- You can see the uploaded pintos code.

Project Structure (Repositories)

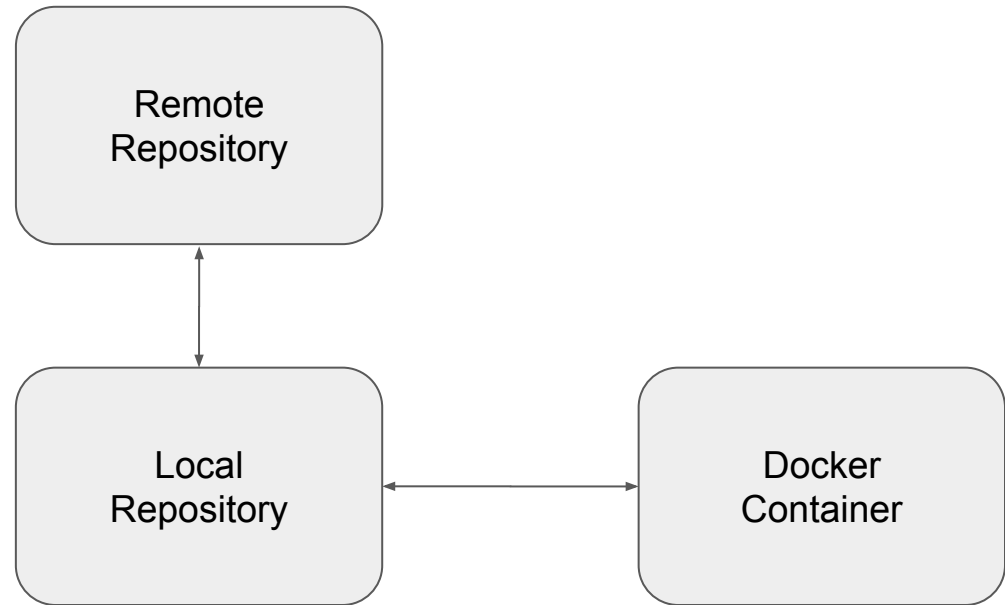


Pintos Repository
(You should push
here!)

Docker Repository
(Not here)

Project Structure (Container)

- Changes are automatically synchronized between local repo. and docker container. (You don't need to take care about synchronization!)
- However, you should commit/push your work at local repository to update remote.



Connect your local pintos folder with docker container

```
> sudo docker run -it -p 80:80 -v [your_pintos_root_dir]:/root/pintos  
--name [container name] [image name]
```

When you successfully run your docker container, then you can see like below:

```
> root@[ID]:/# cd
```

```
> root@[ID]:~# ls
```

```
bochs-2.2.6.tar.gz  pintos
```

Install Bochs

// At docker command line

```
> root@[ID]:~# cd ./pintos/src/misc
```

```
> root@[ID]:~/pintos/src/misc# vi bochs-2.2.6-build.sh
```

Modify CFGOPTS like this:

```
fi
CFGOPTS="--enable-cpu-level=6 --with-x --with-x11 --with-term --with-nogui --prefix=$DSTDIR"
mkdir plain &&
cd plain &&
```

Save and Quit.

Install Bochs

```
> root@[ID]:~/pintos/src/misc# env SRCDIR=/root/ PINTOSDIR=/root/pintos/  
DSTDIR=/usr/local ./bochs-2.2.6-build.sh
```

// Now, it's time to build Pintos project.

```
> root@[ID]:~/pintos/src/misc# cd ../threads
```

```
> root@[ID]:~/pintos/src/threads# make
```

```
> root@[ID]:~/pintos/src/threads# cd build
```

// Test Pintos to verify install is done well.

```
> root@[ID]:~/pintos/src/threads/build# pintos -q run alarm-multiple
```

Run PintOS

- You can start your assignment if the test is done well.

```
(alarm-multiple) thread 0: duration=10, iteration=3, product=30
(alarm-multiple) thread 2: duration=30, iteration=1, product=30
(alarm-multiple) thread 0: duration=10, iteration=4, product=40
(alarm-multiple) thread 1: duration=20, iteration=2, product=40
(alarm-multiple) thread 3: duration=40, iteration=1, product=40
(alarm-multiple) thread 4: duration=50, iteration=1, product=50
(alarm-multiple) thread 0: duration=10, iteration=5, product=50
(alarm-multiple) thread 0: duration=10, iteration=6, product=60
(alarm-multiple) thread 1: duration=20, iteration=3, product=60
(alarm-multiple) thread 2: duration=30, iteration=2, product=60
(alarm-multiple) thread 0: duration=10, iteration=7, product=70
(alarm-multiple) thread 3: duration=40, iteration=2, product=80
(alarm-multiple) thread 1: duration=20, iteration=4, product=80
(alarm-multiple) thread 2: duration=30, iteration=3, product=90
(alarm-multiple) thread 1: duration=20, iteration=5, product=100
(alarm-multiple) thread 4: duration=50, iteration=2, product=100
(alarm-multiple) thread 1: duration=20, iteration=6, product=120
(alarm-multiple) thread 2: duration=30, iteration=4, product=120
(alarm-multiple) thread 3: duration=40, iteration=3, product=120
(alarm-multiple) thread 1: duration=20, iteration=7, product=140
(alarm-multiple) thread 4: duration=50, iteration=3, product=150
(alarm-multiple) thread 2: duration=30, iteration=5, product=150
(alarm-multiple) thread 3: duration=40, iteration=4, product=160
(alarm-multiple) thread 2: duration=30, iteration=6, product=180
(alarm-multiple) thread 3: duration=40, iteration=5, product=200
(alarm-multiple) thread 4: duration=50, iteration=4, product=200
(alarm-multiple) thread 2: duration=30, iteration=7, product=210
(alarm-multiple) thread 3: duration=40, iteration=6, product=240
(alarm-multiple) thread 4: duration=50, iteration=5, product=250
(alarm-multiple) thread 3: duration=40, iteration=7, product=280
(alarm-multiple) thread 4: duration=50, iteration=6, product=300
(alarm-multiple) thread 4: duration=50, iteration=7, product=350
(alarm-multiple) end
Execution of 'alarm-multiple' complete.
Timer: 928 ticks
Thread: 0 idle ticks, 931 kernel ticks, 0 user ticks
Console: 2965 characters output
Keyboard: 0 keys pressed
Powering off...
```

```
=====
Bochs is exiting with the following message:
```

```
[UNIMP ] Shutdown port: shutdown requested
```