

Introduction to docker®

Autumn 2024



AI융합학과

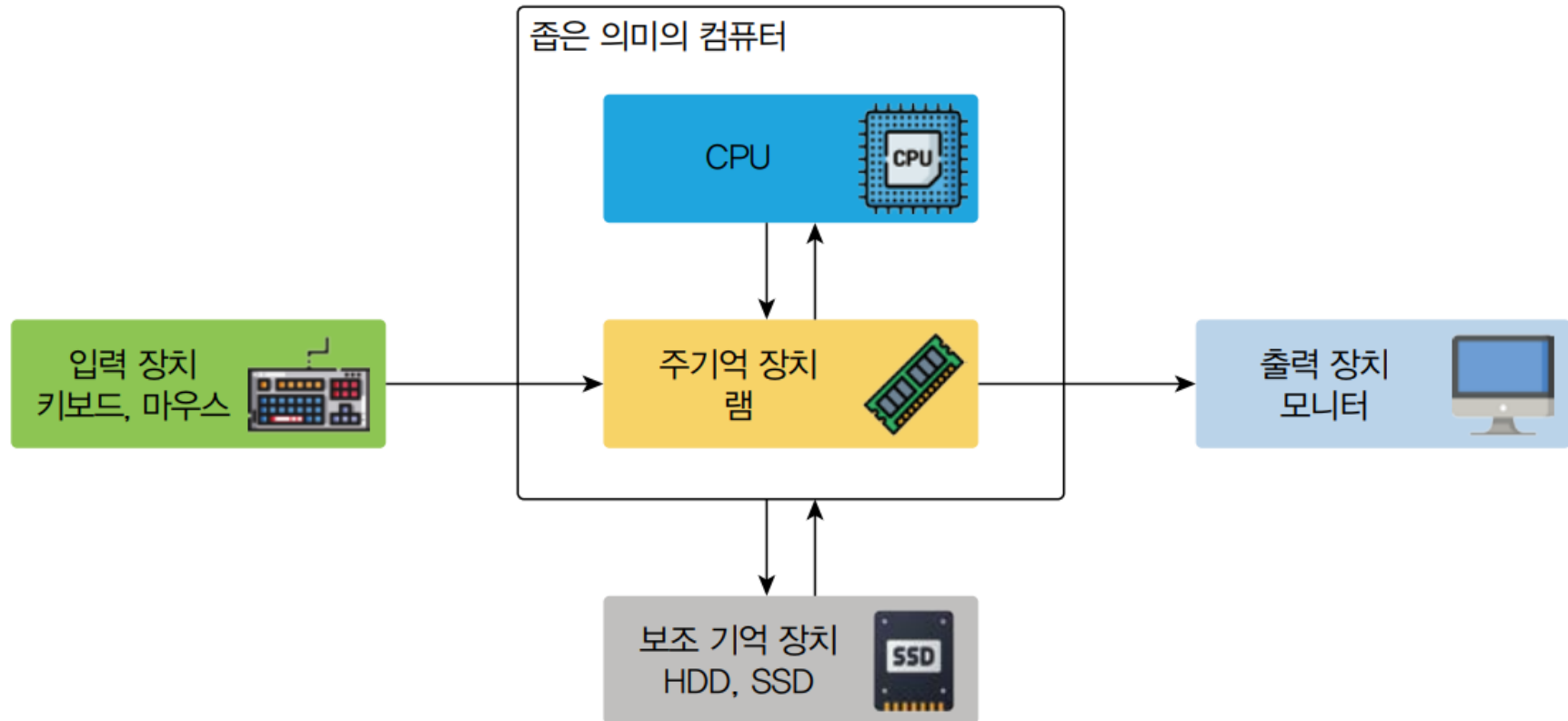
Seongbok Baik

sbbaik@dju.ac.kr

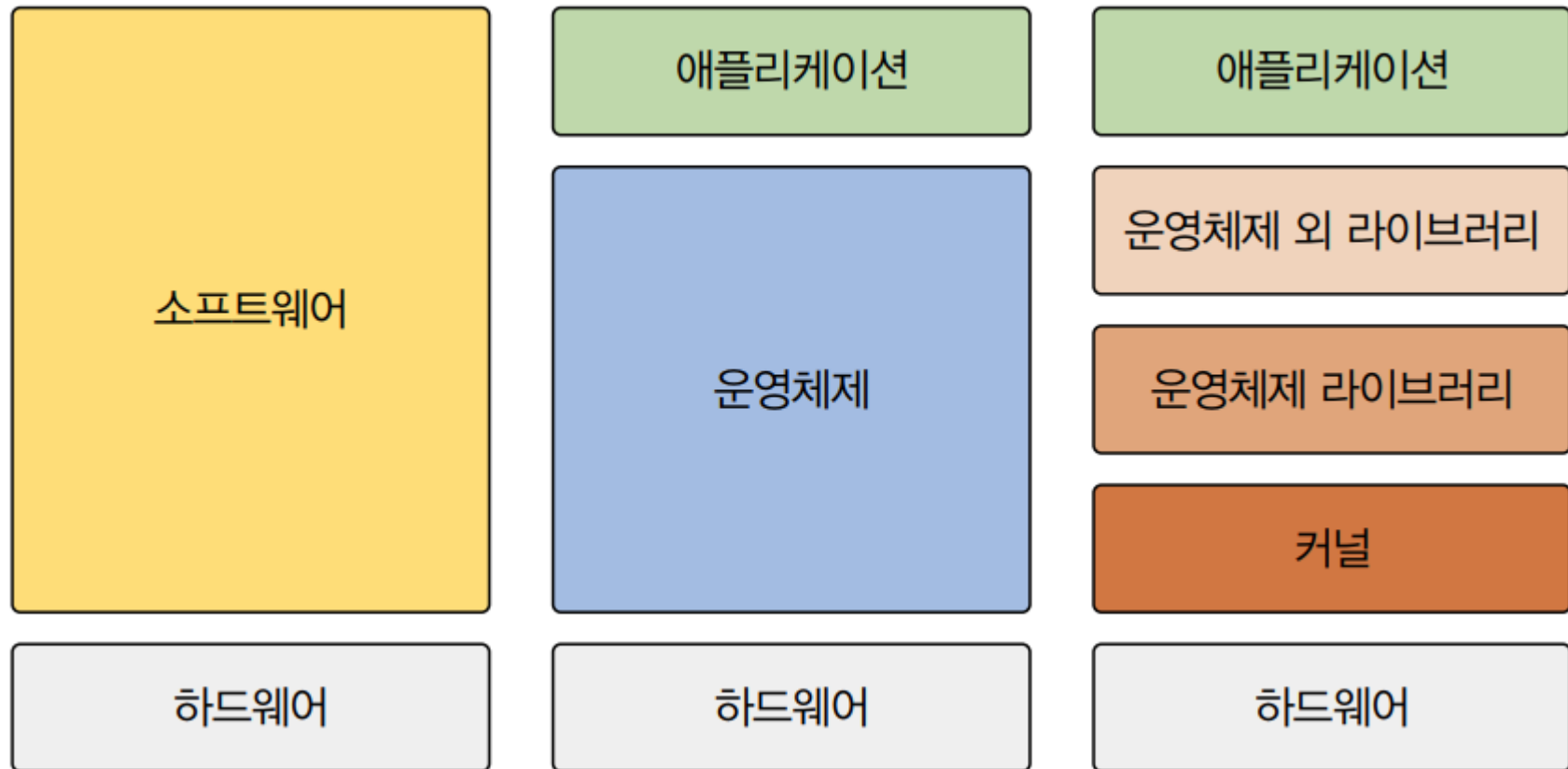


장철원 소프트웨어공학자

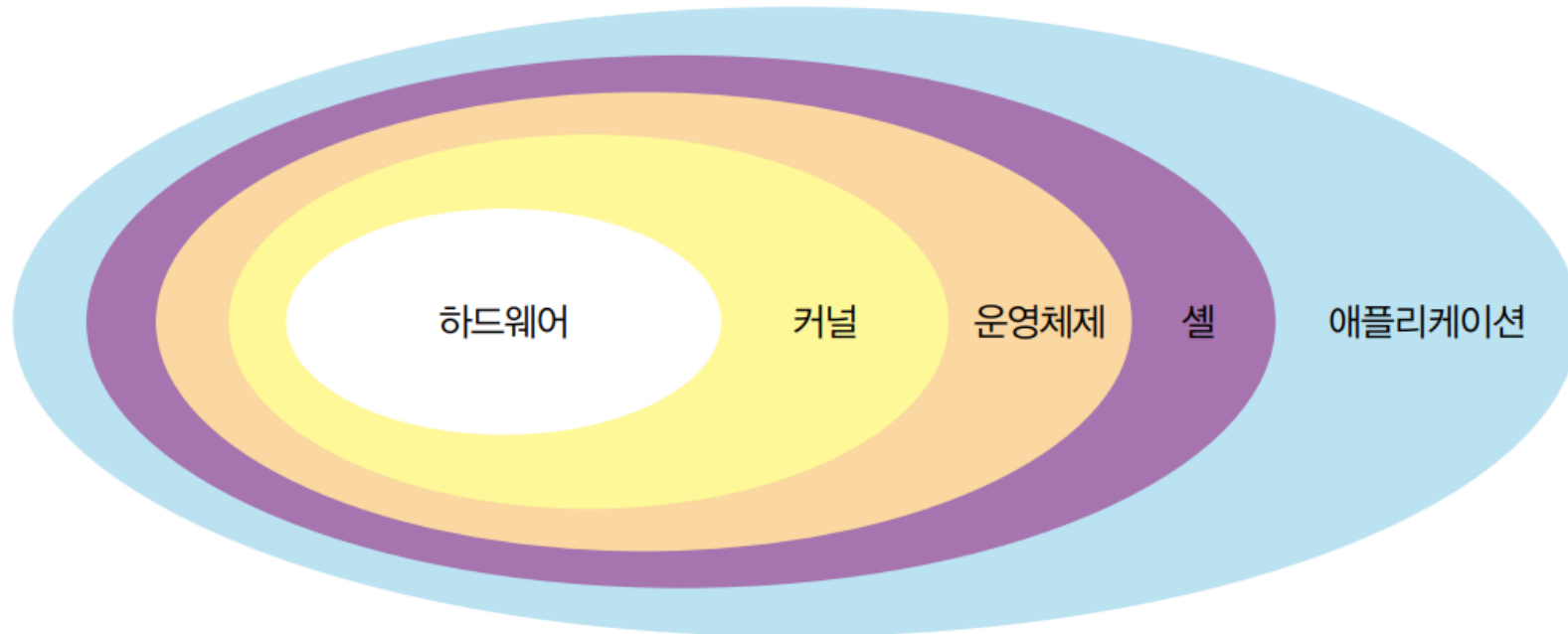
01 Computer Hardware



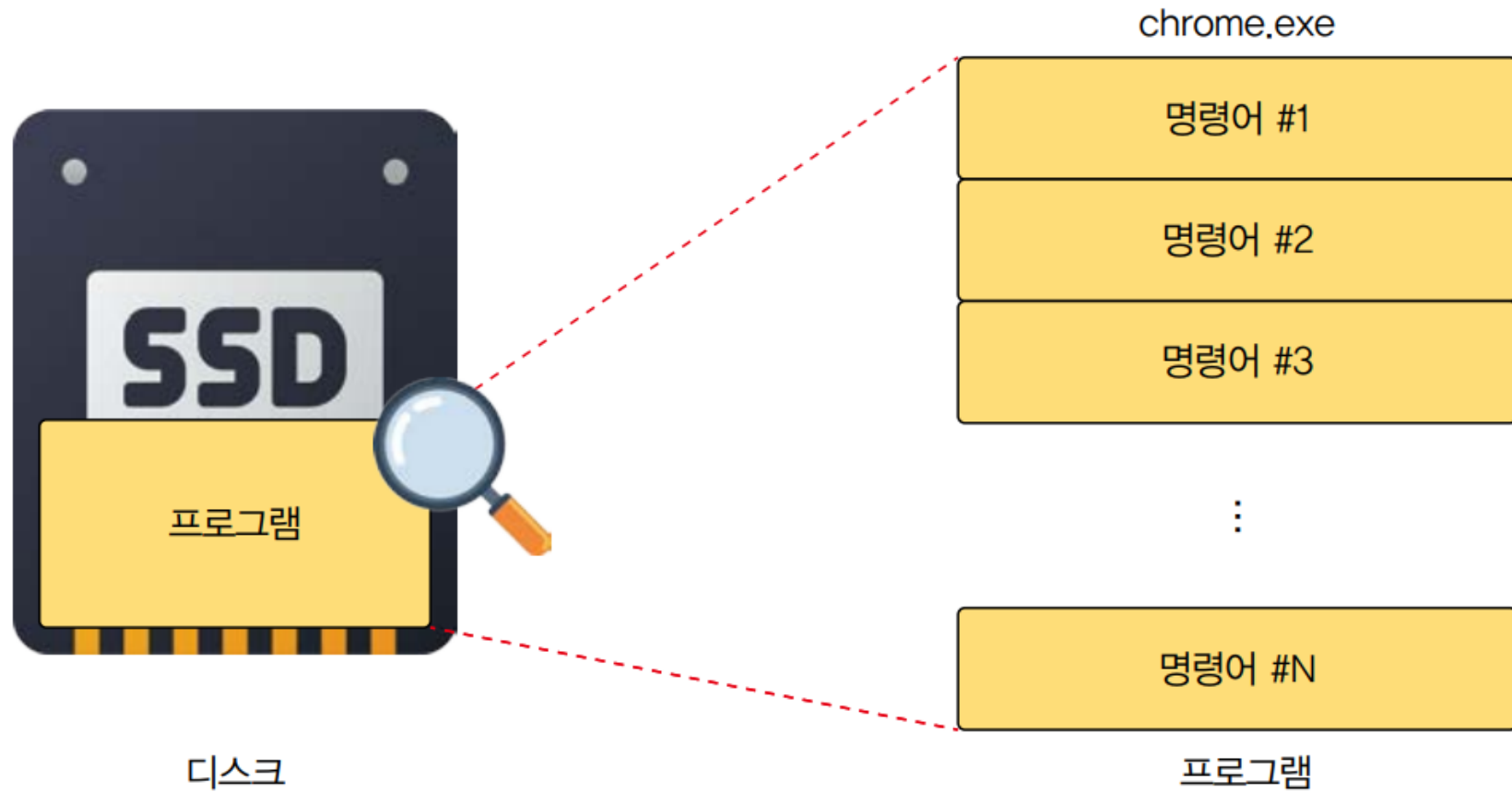
02 Hardware & Software



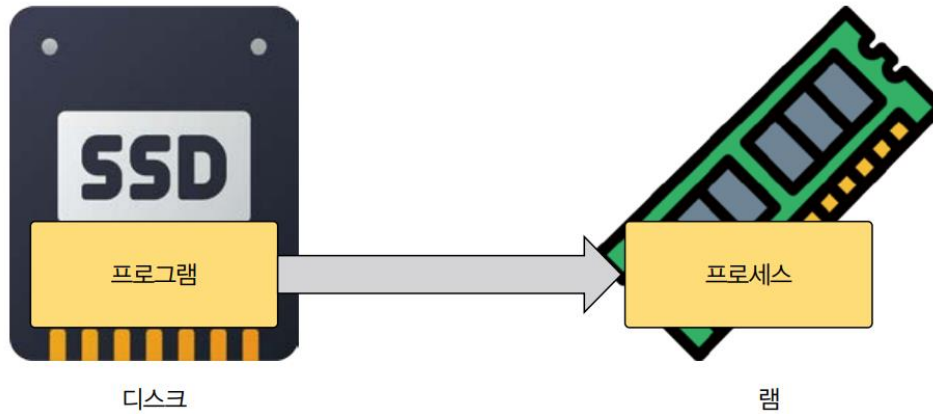
03 Shell



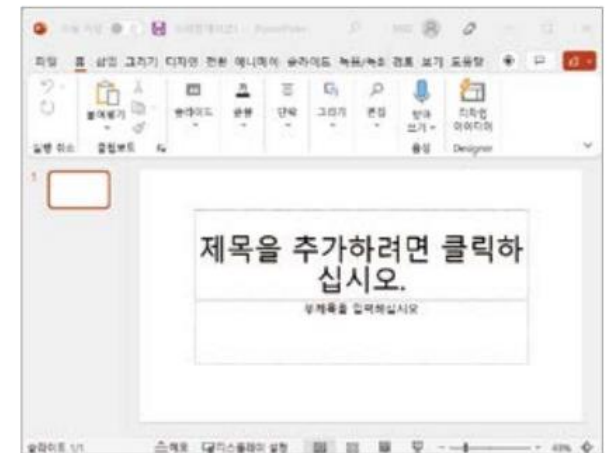
04 Software Program



05 Program & Process

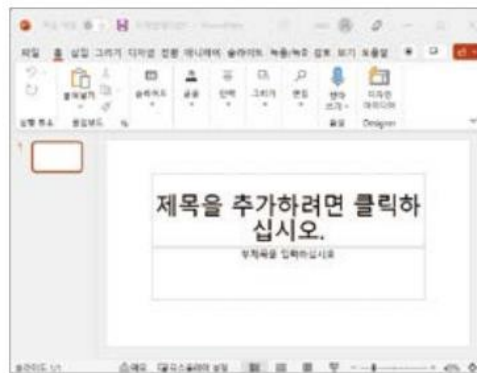
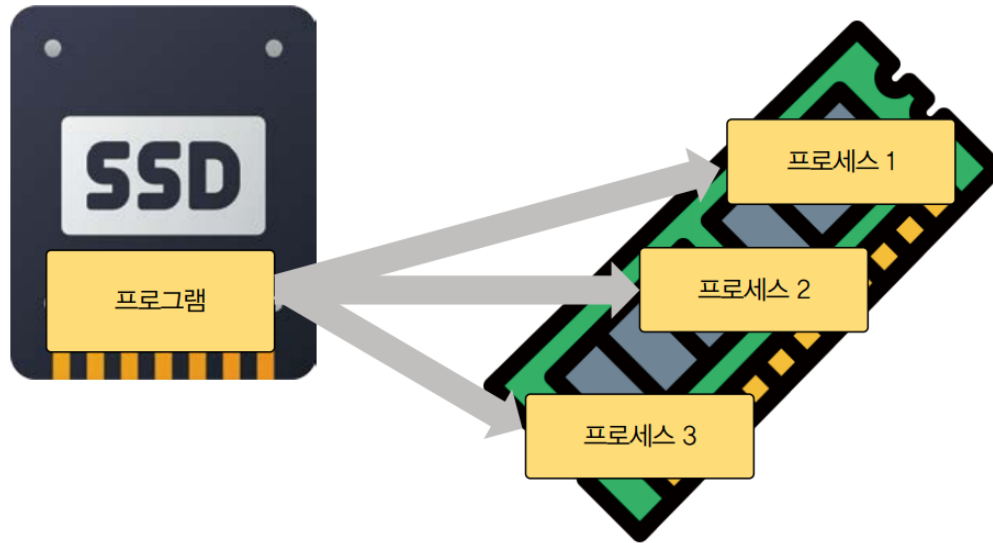


프로그램

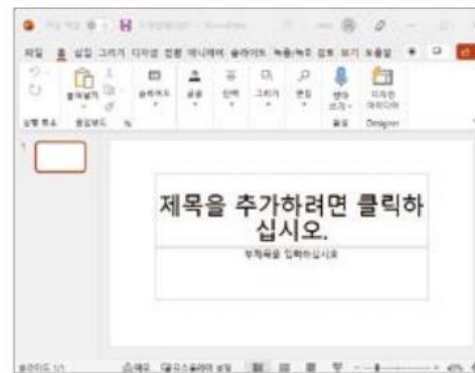


프로세스

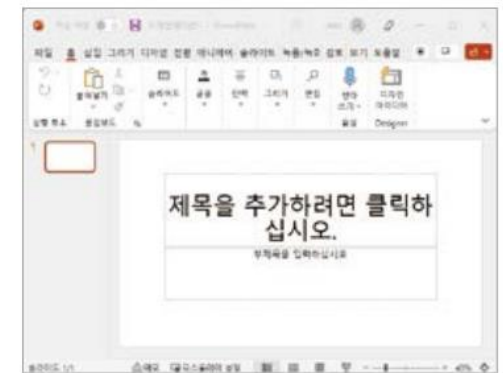
06 Program & Process



프로세스 1

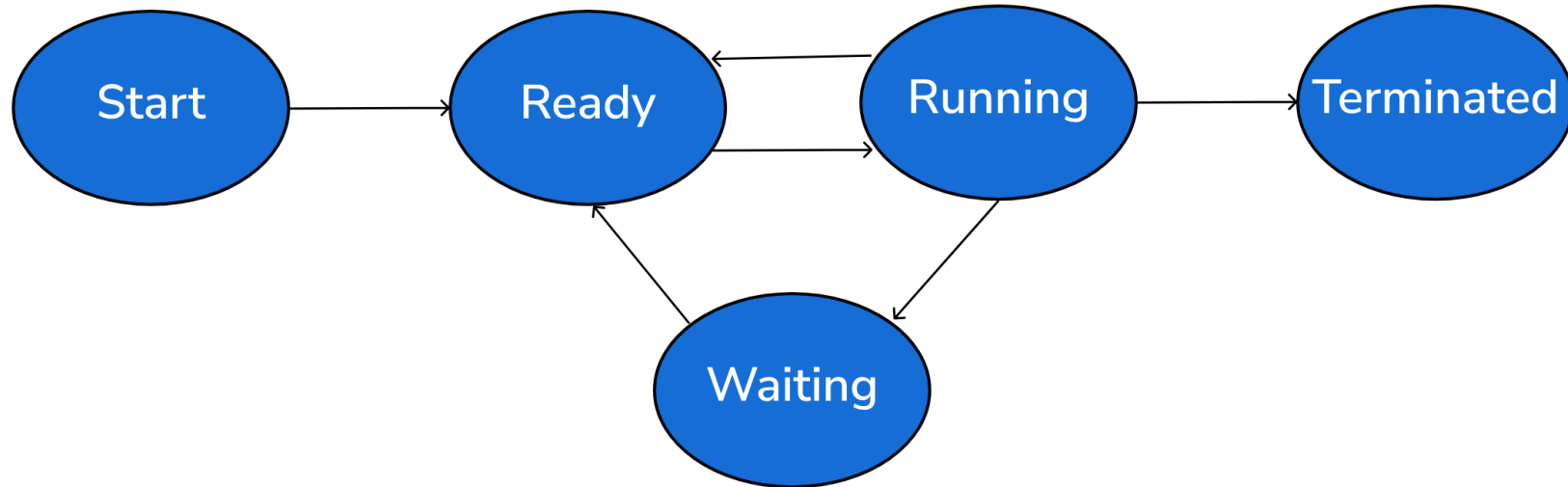


프로세스 2



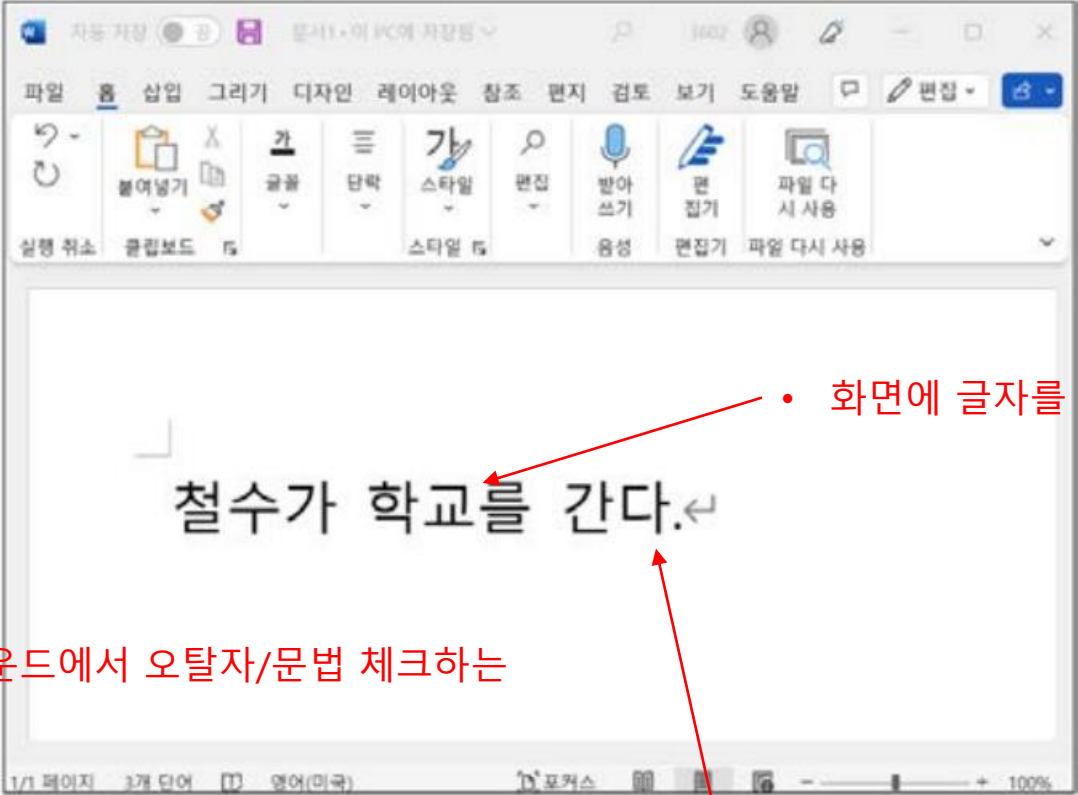
프로세스 3

07 Process States and Lifecycle



- **Start**(or New) : allocating resources and initializing the process control block (PCB). Not yet loaded into the main memory
- **Ready** : loaded into the main memory, waiting for CPU time (scheduling), in a queue called the ready queue, waiting to be selected for execution
- **Waiting** : process has requested access to I/O, waiting for user input, or needs access to a critical region that is currently locked

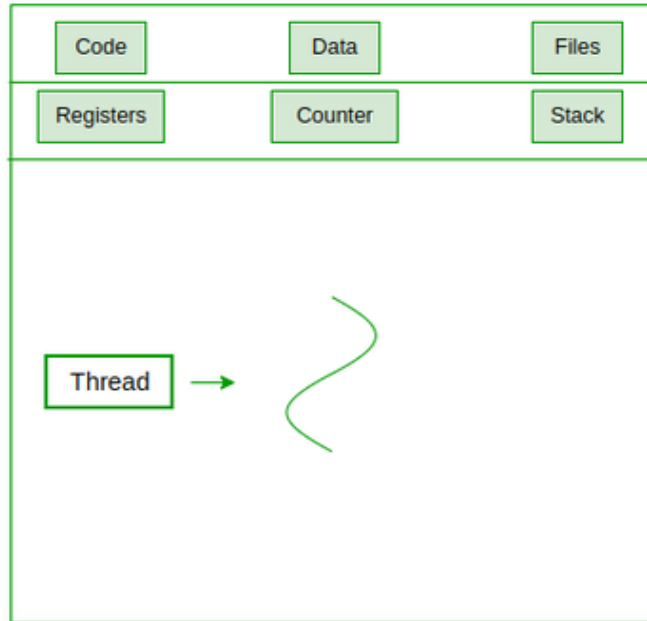
08 Thread



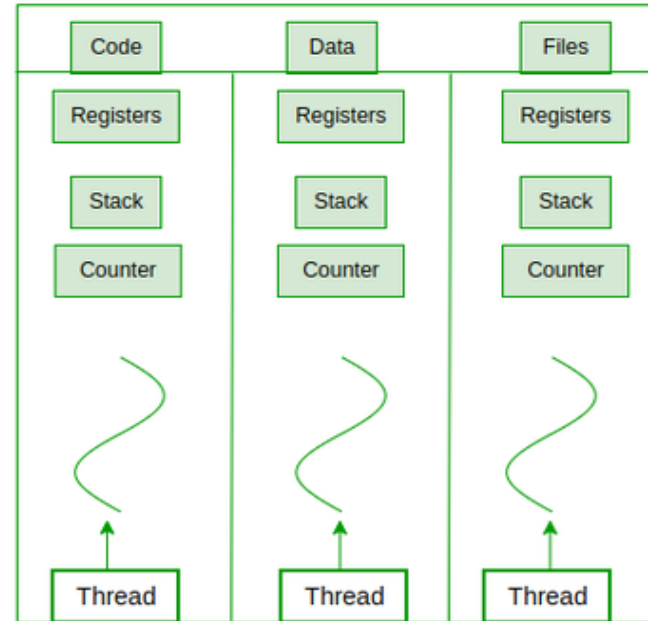
The screenshot shows the Microsoft Word application window. The title bar indicates the document is named '문서1' and is saved to the PC. The ribbon is set to the 'Home' tab, showing various editing tools. The main text area contains the sentence '철수가 학교를 간다.' followed by a cursor. At the bottom, the status bar shows '1/1 페이지', '3개 단어', and '영어(미국)'.

- 화면에 글자를 보여주는 스레드 (Thread showing text on the screen)
- 백그라운드에서 오타자/문법 체크하는 스레드 (Thread checking typos/grammar in the background)
- 사용자 키보드 입력 처리하는 스레드 (Thread processing user keyboard input)

09 Thread



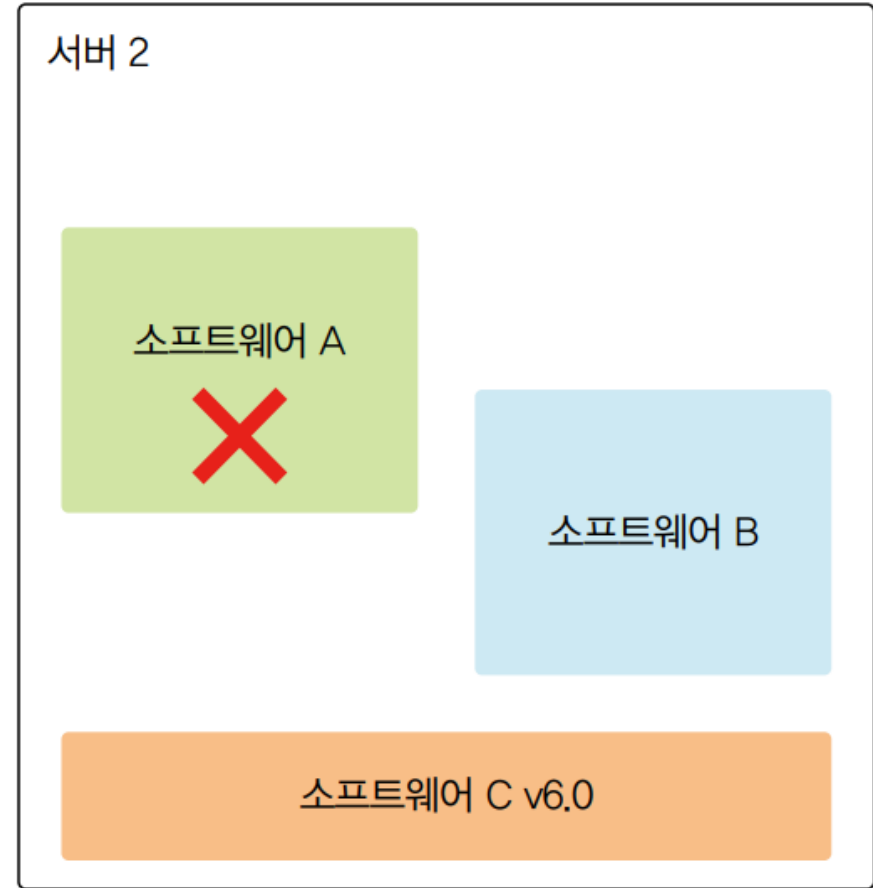
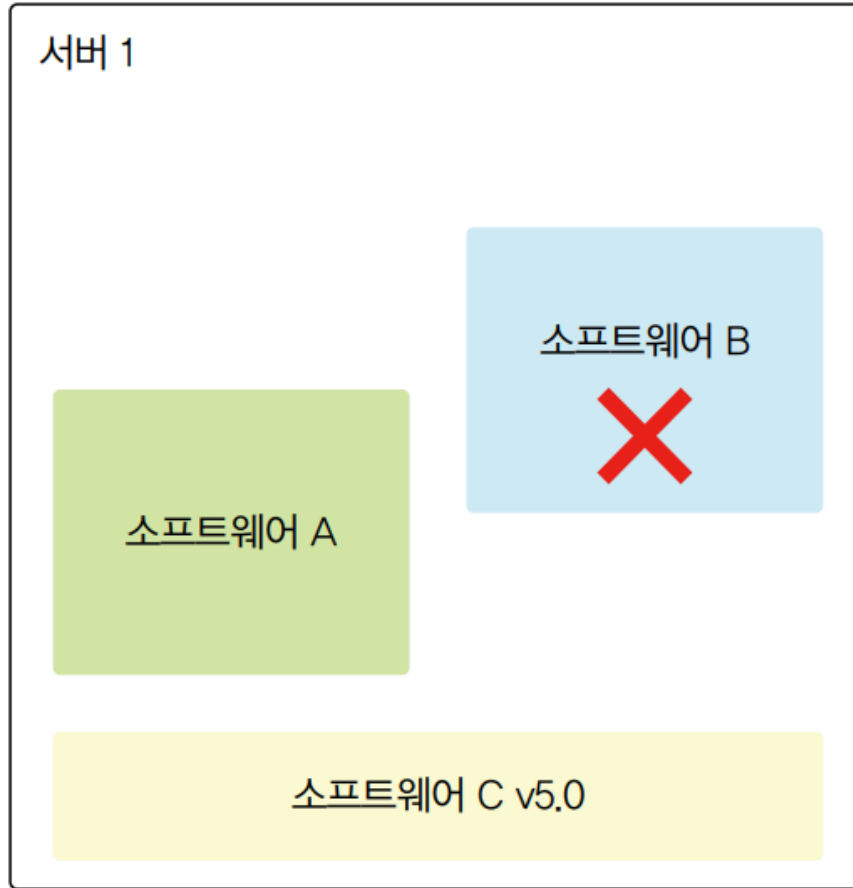
Single Threaded Process



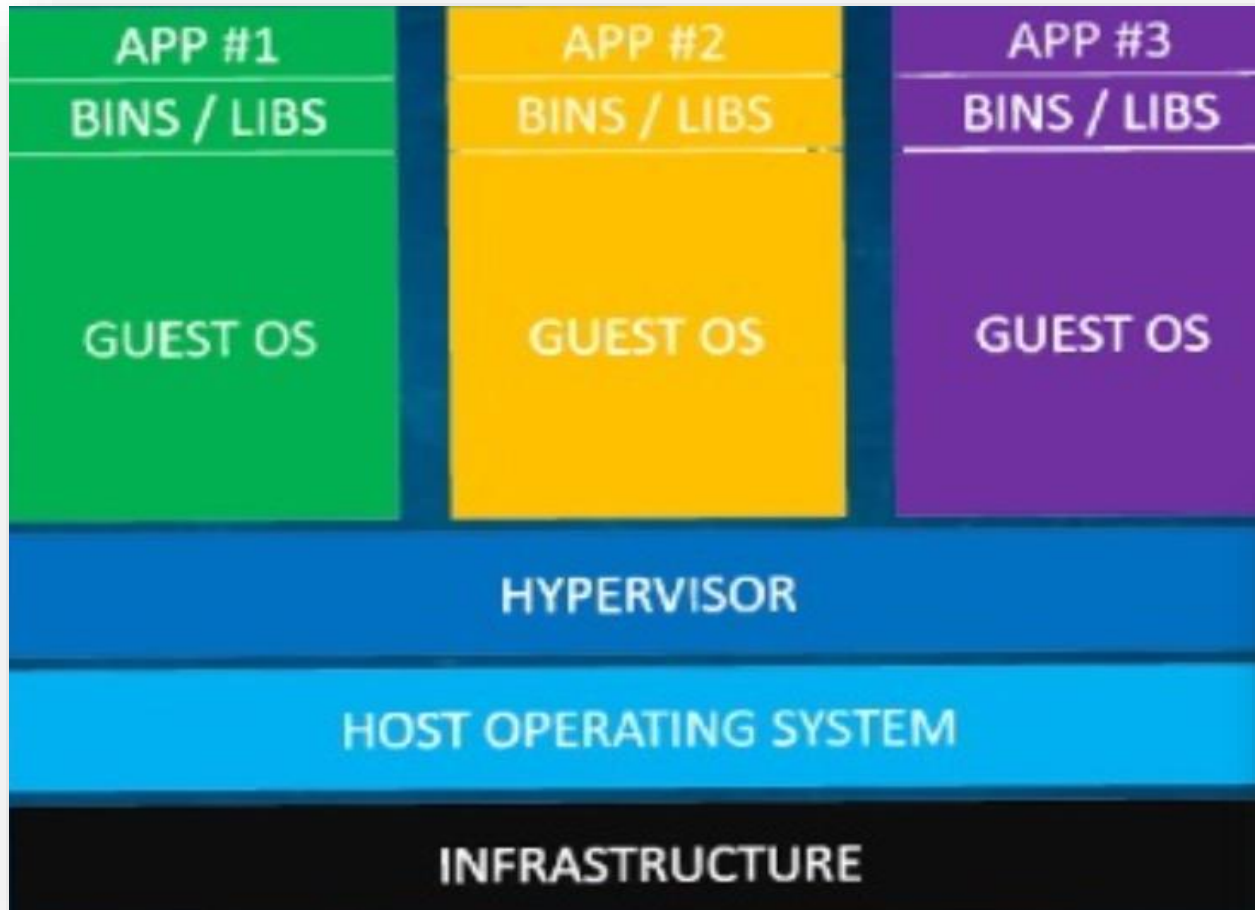
Multi Threaded Process

- An execution stream through a process (a.k.a. Lightweight Process (LWP))
- Has: Program Counter / Register Set / Stack / State
- Shares with other threads of the same process
 - Data Section / Code Section
 - Global Variables
 - Accounting Information
 - Other OS resources, such as open files and signals

10 Software Version Conflicts



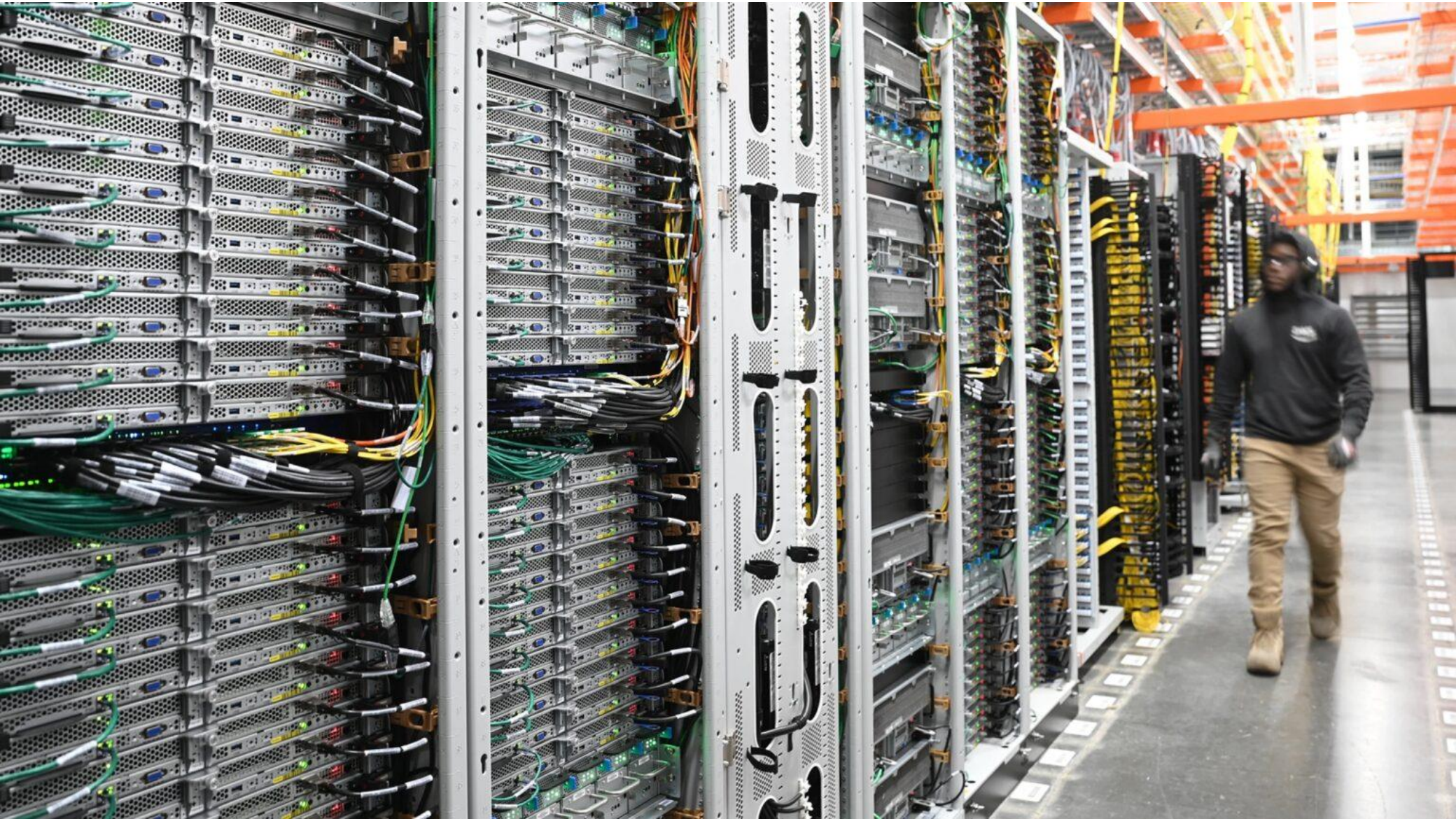
11 Virtual Machine



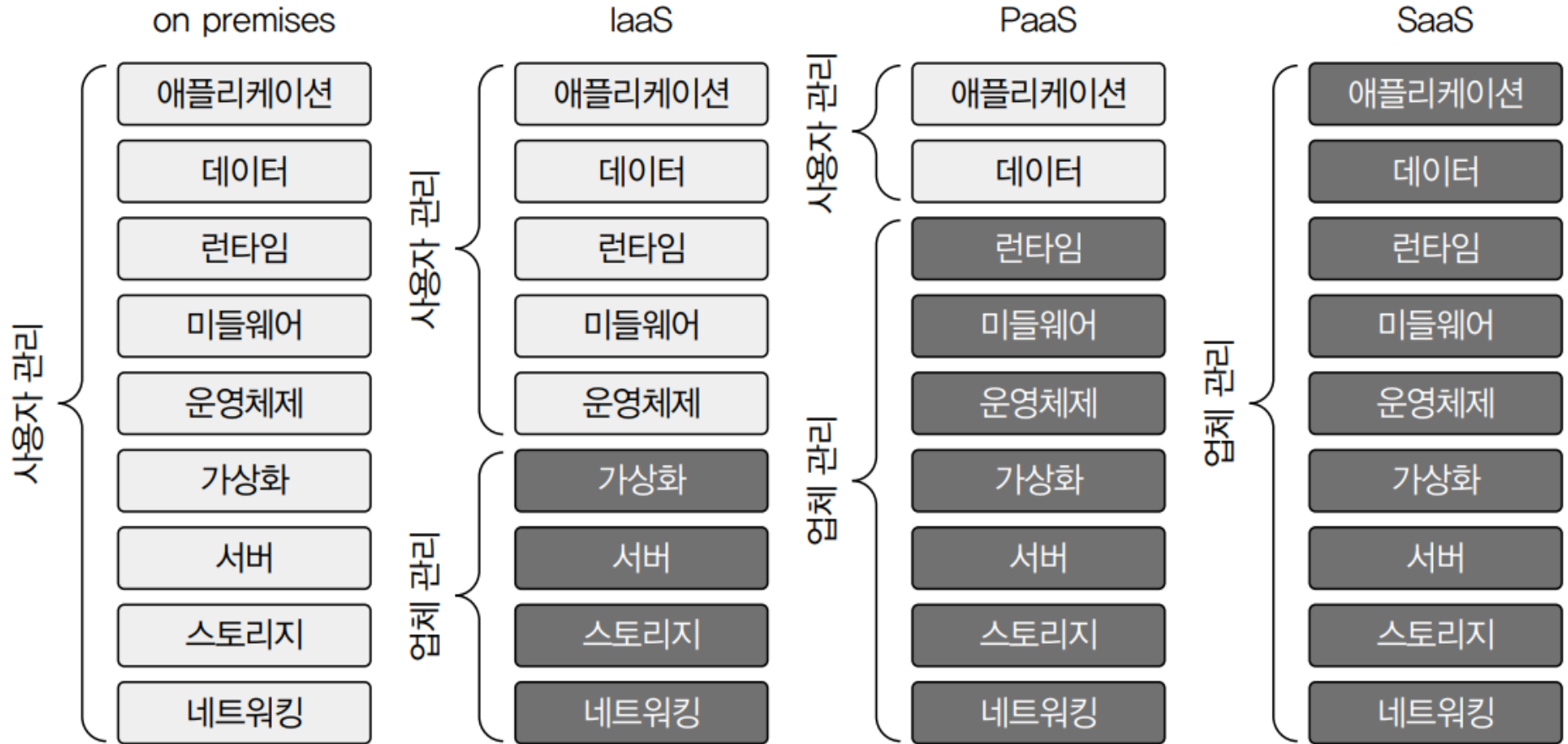
12 Types of Virtualization



13 Cloud Center



14 IaaS / PaaS / SaaS



15 Docker Concept

도커의 정의 : 컨테이너 기술을 기반으로 애플리케이션을 신속하게 구축, 테스트 및 배포하도록 지원하는 소프트웨어 플랫폼



<https://www.docker.com/>

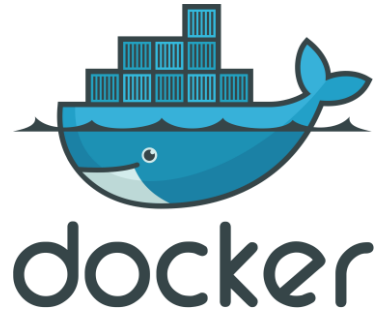


16 Container

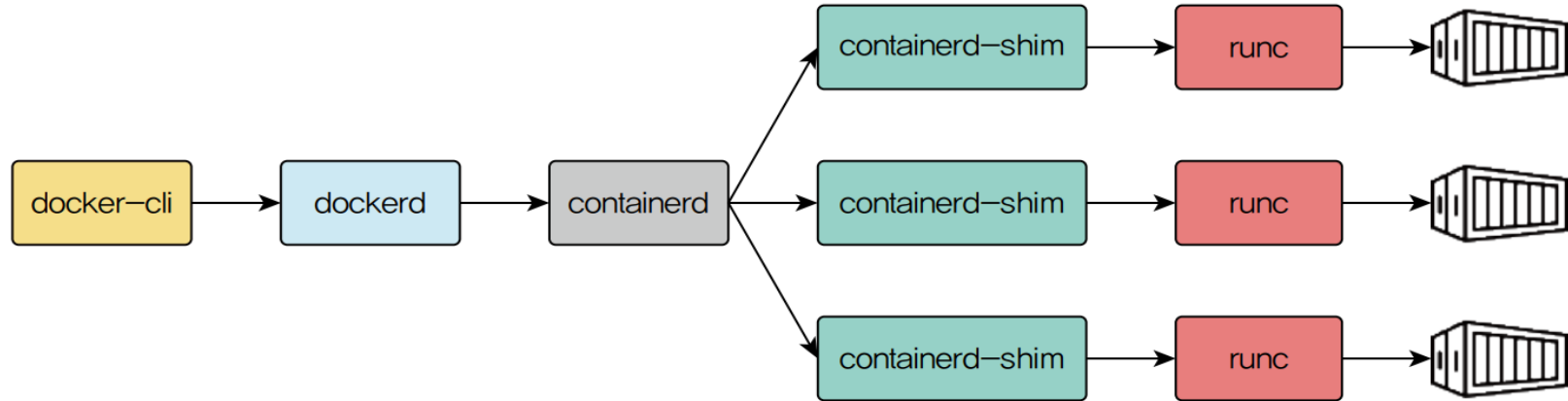
[컨테이너 기술]

- 가상 머신에서 Guest OS를 제거하고 Process 단위로 격리
- 각 격리된 공간(컨테이너)의 내부에는 실행시킬 애플리케이션을 구동하는데 필요한 라이브러리와 실행 파일만 존재
- Host와 격리되어 있기 때문에 어느 환경에서든 동일하게 동작
- 컨테이너를 도커의 요소로 인식하는 경우도 많은데, 사실 **docker는 이 컨테이너 기술을 구현한 것 중 하나로 가장 널리 쓰이는 오픈소스(다른 컨테이너 관리 기술도 존재함)**
- 가상화라는 목표는점은 같지만, 하이퍼바이저는 OS 및 커널이 통째로 가상화시키는 반면,
Container는 FileSystem만 가상화
- Container는 Host PC의 커널을 공유하므로, init(1) 등의 프로세스가 실행될 필요가 없으며, 가상화 프로그램과는 다르게 적은 메모리 사용량, 적은 Overhead를 보임
- Host PC의 자원을 격리(Isolation)된 상태 그대로 활용하기 때문에 VM에 비해 성능 저하가 적음

17 Docker & Container



18 도커 구성 요소



- `docker-cli` : 도커 클라이언트
- `dockerd` : 도커 데몬
- `containerd` : 컨테이너 데몬 (컨테이너 생명주기, 즉 도커 이미지 전송, 컨테이너 실행, 스토리지, 네트워크 등 포함, 고수준 컨테이너 런타임)
- `runc` : 저수준 컨테이너 런타임
- `containerd-shim` : 중간 프로세스, 컨테이너 실행을 조정. `containerd`와 `runc` 사이에서 작동. `containerd-shim`이 `containerd`와 `run` 사이에서 중개자 역할 수행

19 도커 설치 준비 사항

[참고]

- <https://docs.docker.com/engine/install/ubuntu/>

[필요한 패키지 설치]

```
eevee@myserver01:~$ sudo apt-get update ①  
eevee@myserver01:~$ sudo apt-get install ca-certificates curl gnupg lsb-release ②
```

- ca-certificates: 인증서 관련 패키지
- curl: 파일 다운로드 패키지
- gnupg: 디지털 서명을 사용하기 위한 패키지
- lsb-release: 리눅스 배포판 식별을 위한 패키지

20 도커 리포지토리 설정

[GPG 키 추가]

- GPG(GNU Privacy Guard) 키: 도커 이미지 인증 확용용 키

```
eevee@myserver01:~$ sudo mkdir -m 0755 -p /etc/apt/keyrings
eevee@myserver01:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
--dearmor -o /etc/apt/keyrings/docker.gpg
```

[도커 리포지토리 설정]

```
eevee@myserver01:~$ echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

21 도커 설치

[apt 패키지 인덱스 업데이트]

```
eevee@myserver01:~$ sudo apt-get update
```

[도커 패키지 셋 설치]

```
eevee@myserver01:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-  
buildx-plugin docker-compose-plugin ①
```

```
Reading package lists... Done
```

```
Building dependency tree... Done
```

```
Reading state information... Done
```

```
...(생략)
```

```
Scanning processes...
```

```
Scanning linux images...
```

```
Running kernel seems to be up-to-date.
```

```
No services need to be restarted.
```

```
No containers need to be restarted.
```

```
No user sessions are running outdated binaries.
```

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

22 도커 설치

[설치 확인]

```
eevee@myserver01:~$ docker
```

1

```
Usage:  docker [OPTIONS] COMMAND
```

```
A self-sufficient runtime for containers
```

```
Common Commands:
```

```
run      Create and run a new container from an image
```

[도커 실행 상태 확인]

```
eevee@myserver01:~$ systemctl status docker
```

1

```
● docker.service - Docker Application Container Engine
```

```
Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
```

```
Active: active (running) since Thu 2023-11-02 00:41:59 UTC; 1min 10s ago
```

```
TriggeredBy: ● docker.socket
```

```
Docs: https://docs.docker.com
```

```
Main PID: 2835 (dockerd)
```

```
Tasks: 11
```


23 도커 설치

[사용자 모드에서 실행 가능 하도록 설정]

```
eevee@myserver01:~$ sudo usermod -aG docker $USER
```

- 위와 같이 'docker' 그룹에 사용자를 추가한 뒤 Logoff 했다가 다시 Login 해서 사용자 모드에서 docker 명령 실행되는 지 확인.

[도커 버전 확인]

```
eevee@myserver01:~$ docker version ❶  
Client: Docker Engine - Community  
Version:           23.0.1  
API version:       1.42  
Go version:        go1.19.5
```

```
Server: Docker Engine - Community  
Engine:  
Version:           23.0.1  
API version:       1.42 (minimum version 1.12)  
Go version:        go1.19.5
```

24 Hello Docker

```
eevee@myserver01:~$ docker run hello-world
```

1

...(중략)

Hello from Docker!

This message shows that your installation appears to be working correctly.

...(중략)

서로에게, 자신에게 친절합니다 - 허준이

