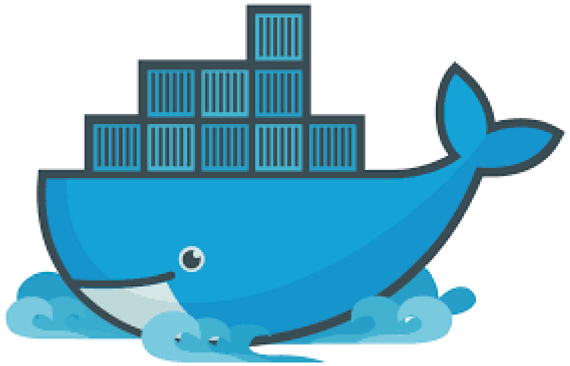
Program Systems Development practice



Practice 3

Docker, Containers

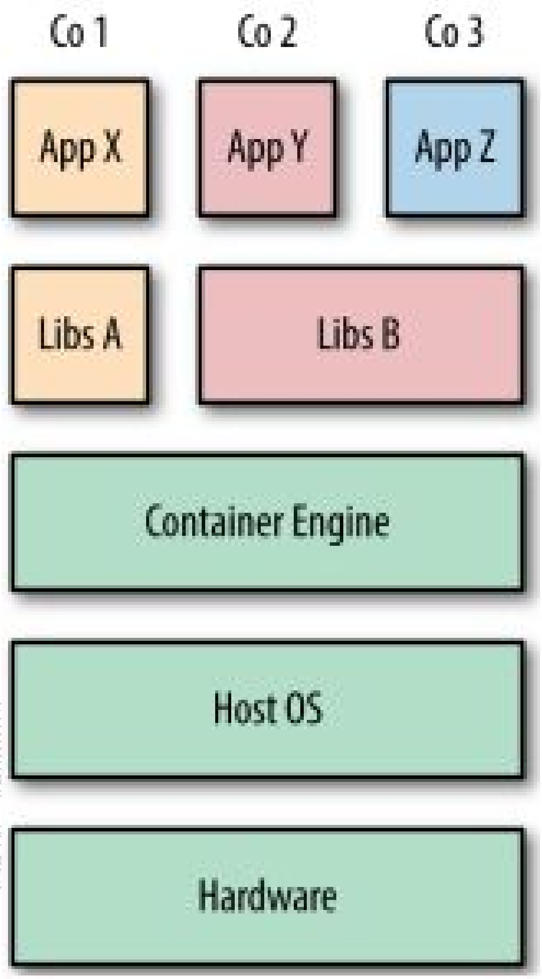
What’s docker?

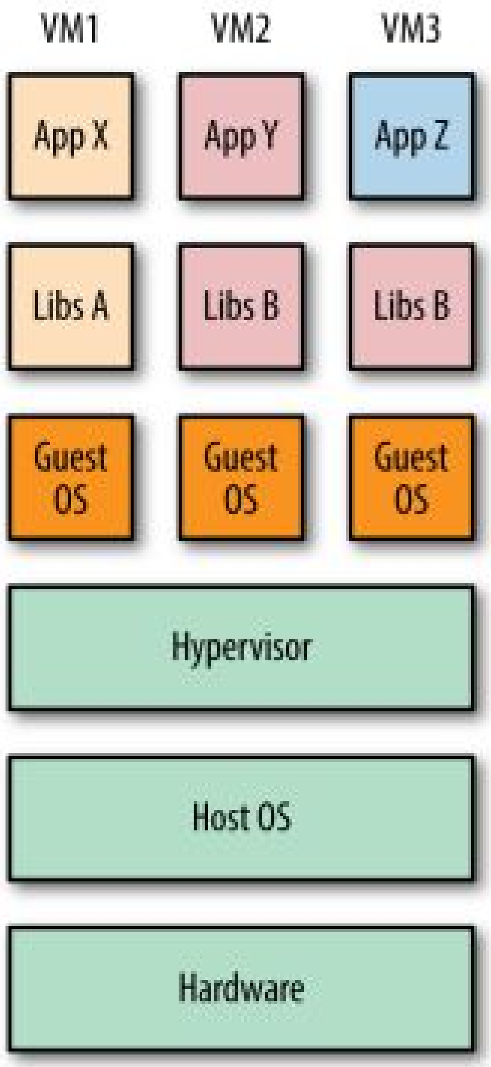
- Operating-system-level virtualization
  - An additional layer of abstraction
  - Based on Linux
- ”Split up a computer into isolated containers that run your code”
- Builds these containers
- Social platform to find and share containers

What is a container?

- A self-contained isolated unit of software
- Contains everything required to run the code
- Includes:
  - Code, configs, processes, networking, dependencies, OS

Containers vs. VMs

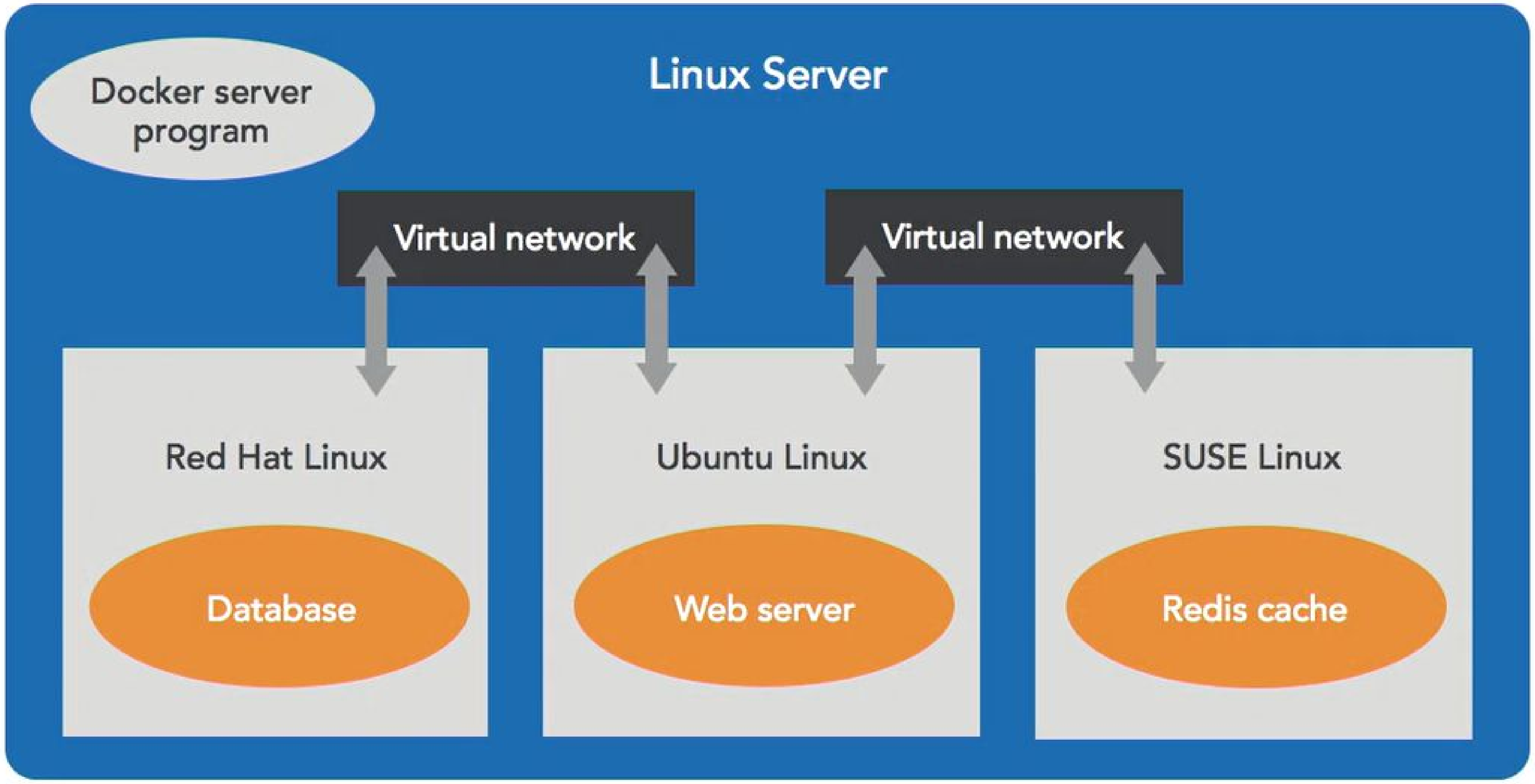




Main advantages of containers

- Can be started and stopped in a fraction of a second
- Portability
- More than one container can be run at the same time
- Run complex applications without long configuration and installing processes

How it works?


Linux Server
Docker server program
Virtual network
Virtual network
Red Hat Linux
Database
Ubuntu Linux
Web server
SUSE Linux
Redis cache

Dockerfile

- File that contains commands such as
  - Loading other images
  - Installing drivers
  - Running
- After a build it becomes a Docker image stored in your local Docker registry

```
docker build -t result_name .
```

How build works?

- Before running the instructions, Docker daemon performs a preliminary validation
  - Syntax errors
- Instructions will be run one-by-one
  - Result of each instruction is commited to a new image (run independently)
  - Docker daemon automatically clean up the context
  - Intermediate images are re-used (cache)

Docker image

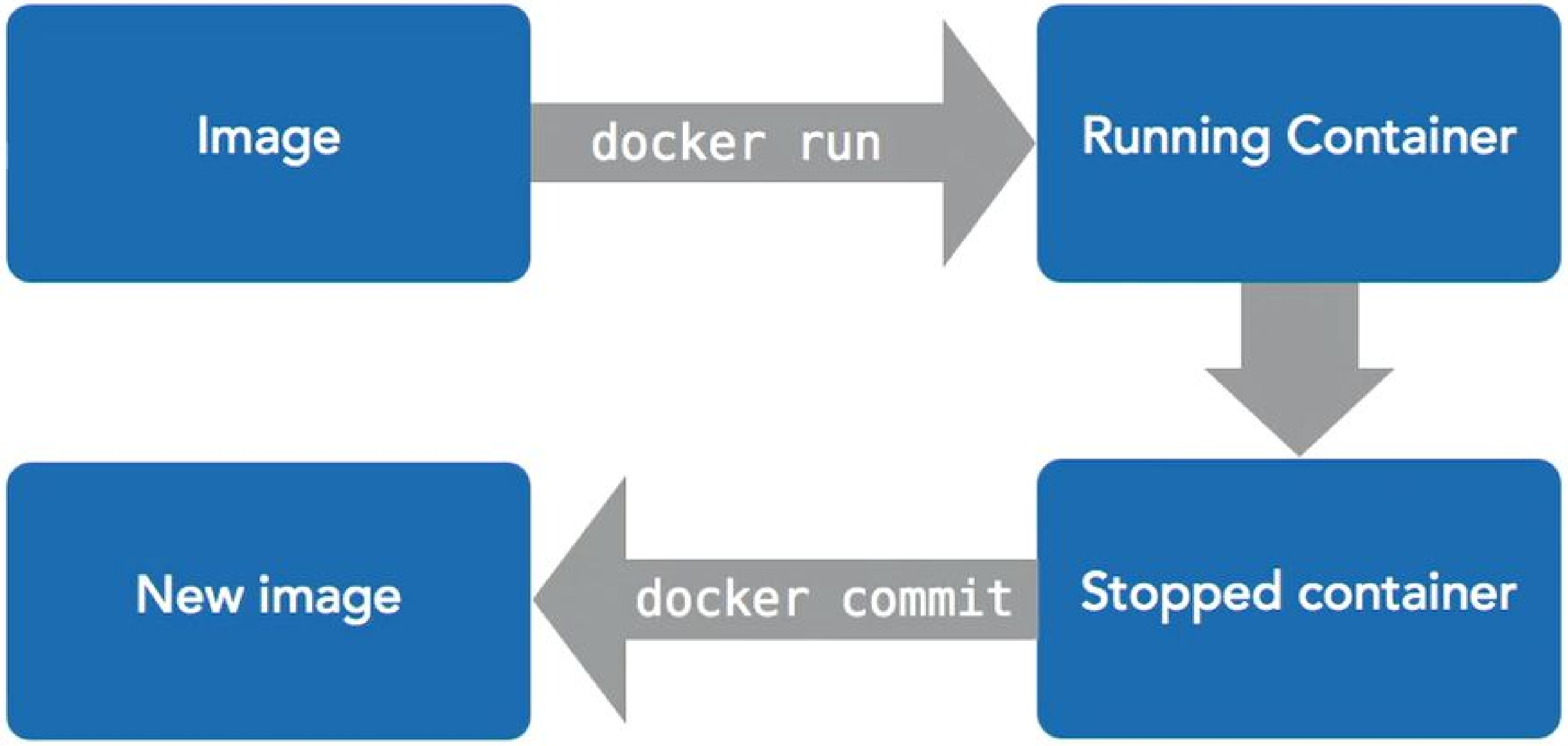
- Filesystem and parameters to use at runtime
- It has no state and never changes
- A container is a running instance of an image

Dockerfile instructions

- FROM – getting an existing image from Docker Hub
- RUN – installation & configuration part
- CMD – running the software contained by image
- EXPOSE – indicates the ports on which a container will listen for connections
- ADD – fetch packages from remote URLs
- COPY – supports basic copying of local files into the container

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Docker workflow



Run an image

- `docker run [options] image_name`
  - <u>Options:</u>
    - **`-d`** – run in the background
    - **`--rm`** – it gets removed after stopping the container
    - **`-p`** – specify the port that it uses
    - -t – allocating a pseudo TTY
    - -i – make it interactive
  - **`image_name`** – the name of the image file
- *<u>Pseudo TTY:</u> having the functions of a physical terminal without actually being one*

Getting list of containers/images

- `docker ps`
  - Only running containers
- `docker ps -a`
  - Containers that are running and are stopped but not removed
- `docker image ls`
  - List all the images from the Docker registry

Connect to a running container

- `docker exec [options] container_id command`
  - Options:
    - -i – interactive
    - -t – allocating a pseudo TTY
  - container_id – 12 characters long id
  - command – the command that you wan't to execute (e.g.: ls, mkdir, etc…)

Restart a stopped container

- `docker start [options] container_id`
  - Options:
    - -i – interactive
  - container_id – 12 characters long id

Killing a running container

- `docker kill [options] container_id`
  - Stops the container and removes it from the background
  - It won't be removed from the Docker registry

Deleting containers/images

- `docker rm [options] container_id`
  - Deletes a container given by ID
- `docker rmi [options] image_name`
  - Deletes an image