Internet technologies

Containerized server/application and serverless computing

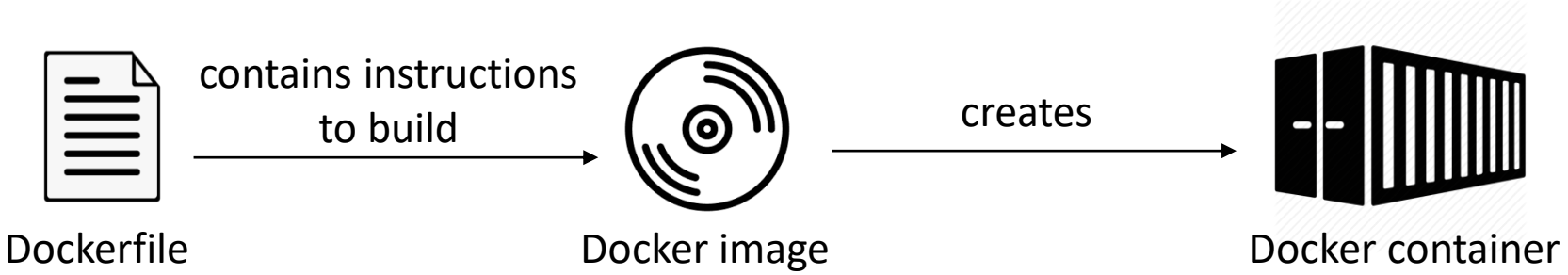
Van Khanh Duy Phan – PHA0025

Traditional web server & its problems

- An engineer needs to manually craft the server, i.e. to install and update packages and dependencies via command line.
- Problem:
  - When there's the need to install new dependencies, unnecessary old dependencies are usually not uninstalled → increasing disk space.
    This usually also is a challenge to new engineer who has to maintain the server.
  - If an unauthorized application is installed on the server, e.g. collect logs, dump database periodically, it will stay there unnoticed for a long time.
  - When there are many servers, it's hard to keep track of which servers have been updated.
  - When there is high load on the application, the webserver needs to be scaled, usually scaling vertically, i.e. to equip more processing unit and memory, this requires downtime.

Solution: Containerized server/application

- One solution is to containerize application. One container platform is Docker.
- A Docker container is a unit of software that consists of the code and its dependencies. This container can be spin up and ran on any computing environment.
- A document, called Dockerfile, contains all the instructions needed to build a Docker image (e.g. check out the code from version control system, install dependencies …). Then a container can be run based on the Docker image. Docker image can be understand as a template to start a container.



Dockerfile

- An example of Dockerfile that using preinstalled nginx on Debian OS, then install packages, copy configuration from host machine, and specify which command to run after the container is created.

```
# Specify base image
FROM nginx:1.17
# Install extra programs
RUN apt-get update && apt-get install -y curl vim
# Copy config from host machine
COPY config /etc/
# Final command to run when container is started
CMD ["nginx", "-g", "daemon off;"]
```

Advantages

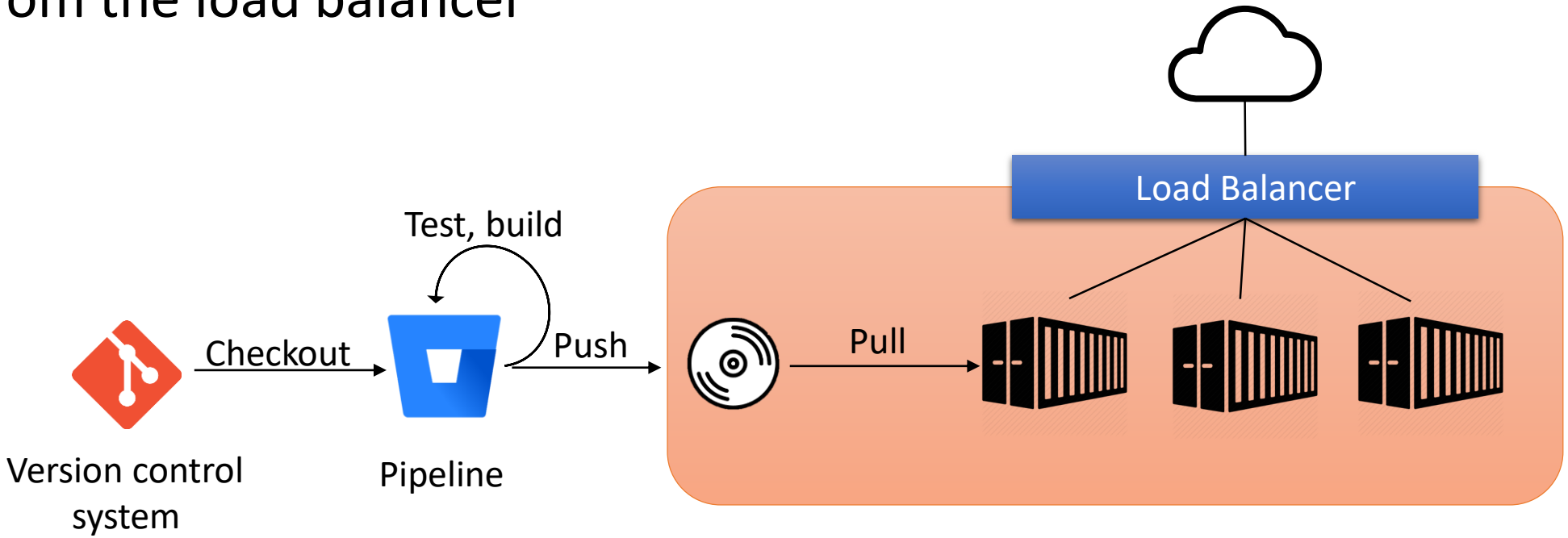
- After the Docker image is built, it can be stored in any image repository, e.g. https://hub.docker.com/. The image can later be pulled and started at any computing environment that has Docker application installed.
- Having Dockerfile, we have version controlled of which packages and dependencies were installed.
- If the container was compromised by unauthorized access, that compromised container is removed at the next deployment of application.
- When there is high load, new containers can be quickly created to share the load, this is called horizontal scaling. In cloud computing environment, the number of containers can be scaled dynamically depending on the load, which is high cost-efficient.

Advantages ctnd.

- The container behaves the same on any computing environment, therefore the new developer doesn’t have to spend hours on the first day of work to configure his own developing environment to be able to run the application. All (s)he needs is Docker platform, Dockerfile and the application source code.

Overview

- Dockerfile is stored along with the source code in version control system.
- Deployment pipeline builds and pushes Docker image to repository
- Multiple containers are created from Docker image, share the load from the load balancer



Serverless computing

- With the previous model, there must be at least one container running at any time provide accessibility, even when there is no traffic at all.
- New executing model, serverless computing, can be considered with these advantages:
  - Pay-as-you-go: you are charged based on the time and memory allocated to run the code.
  - Elasticity: you don’t have to set up auto-scaling polices to handle starting/stopping containers, the cloud system will handle it for you.
  - Microservices: Instead of having one big application, you can split it into multiple microservices, it is loosely-coupled, easier to develop and debug.

Serverless application framework

- One popular open source serverless application framework is *serverless*, provided by Serverless, Inc.
- One yaml file is stored along with the source code, containing configurations for the serverless functions in cloud provider.
- *serverless* supports deployment to most cloud providers, e.g. AWS, Azure, Google Cloud, Knative, …
- Here is one configuration example of a serverless function to be deployed to AWS (Amazon Web Services) to send lunch menu to a message channel at specified time.

```
provider:
  name: aws
  runtime: python3.8
  memorySize: 128
  timeout: 200

functions:
  lunchMenu:
    handler: lunchMenu.handler
    events:
      - schedule:
          rate: cron(30 9 ? * MON-FRI *)
```