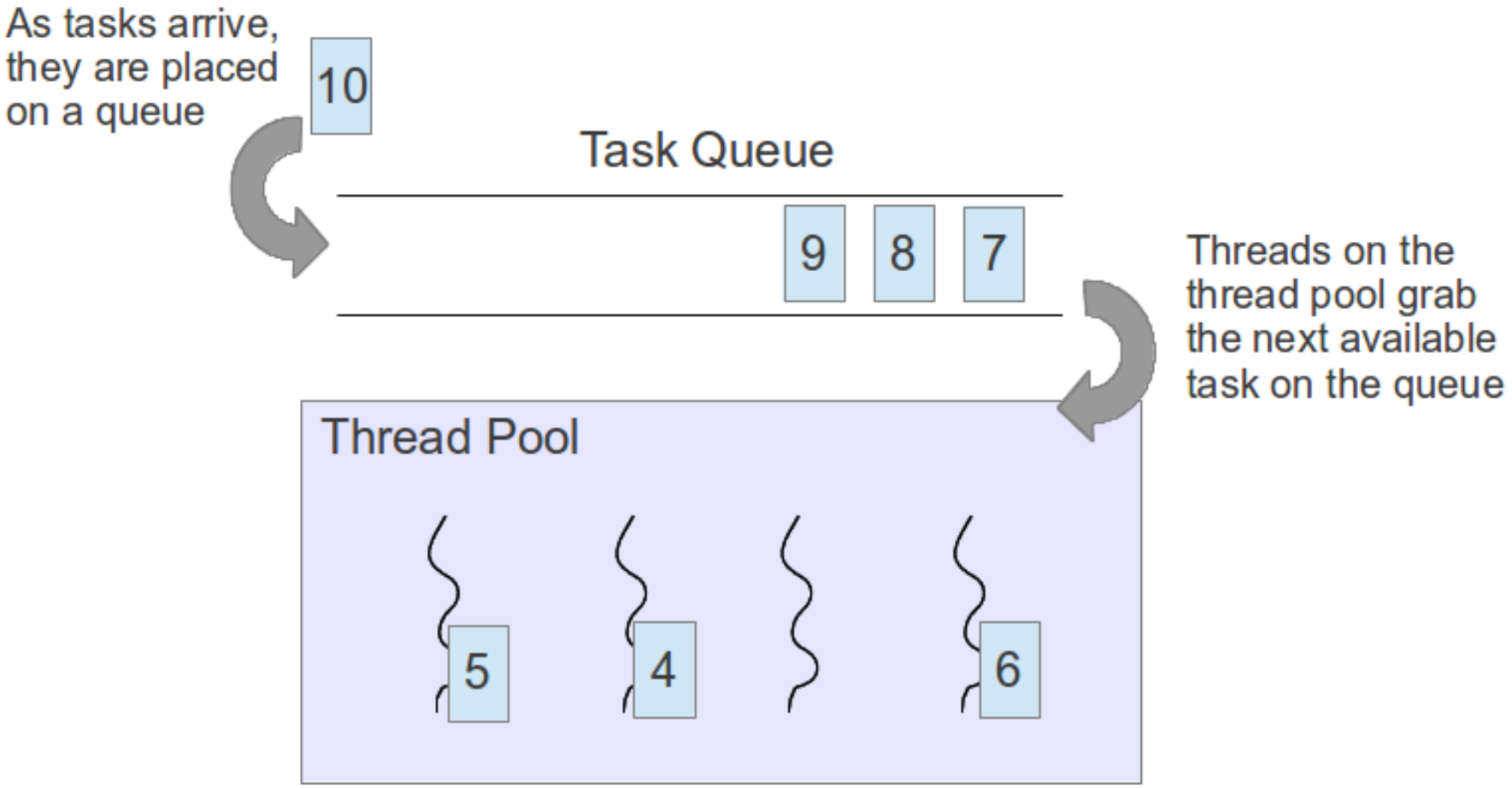
ALPEN-ADRIA UNIVERSITÄT KLAGENFURT

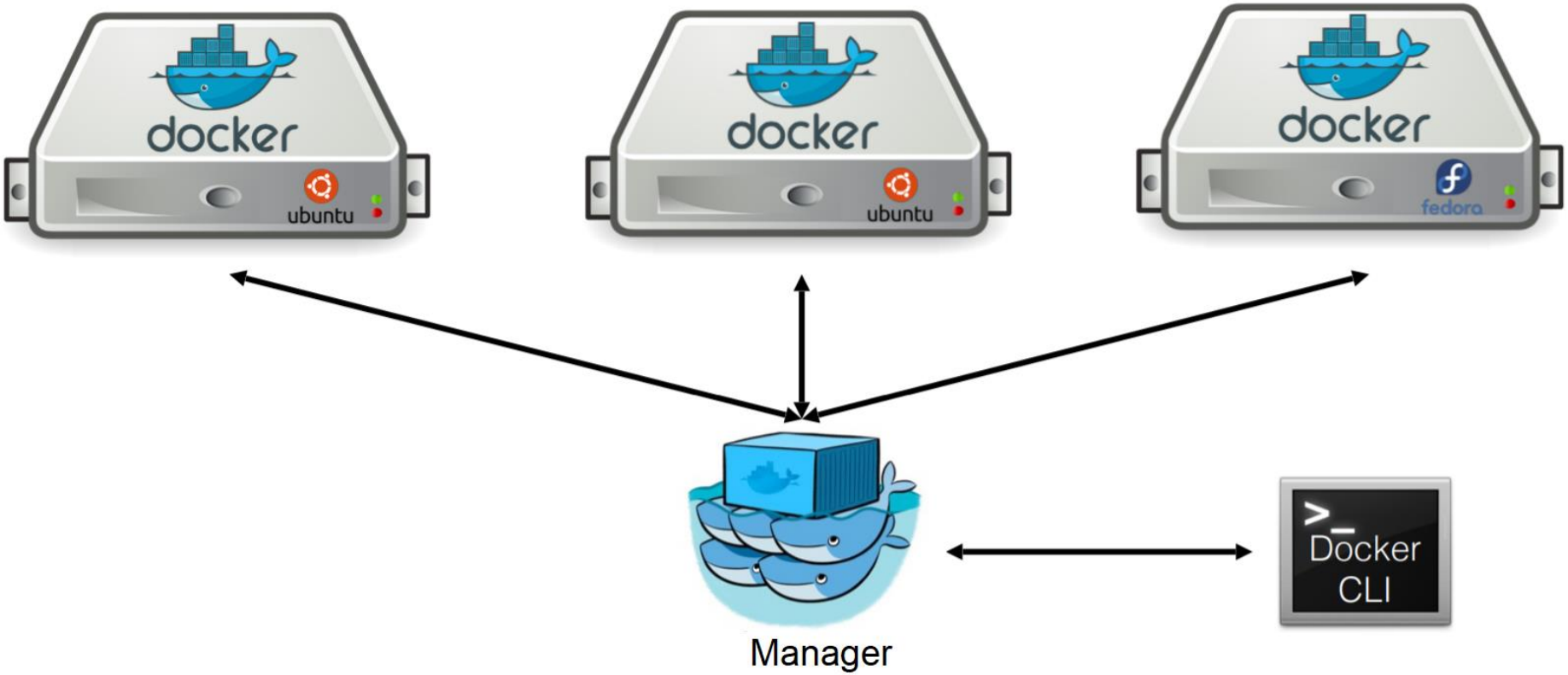
Docker Containers Tutorial part3 Docker Swarmkit

Distributed Systems UE

Thread pool in one CPU core



Cluster of devices

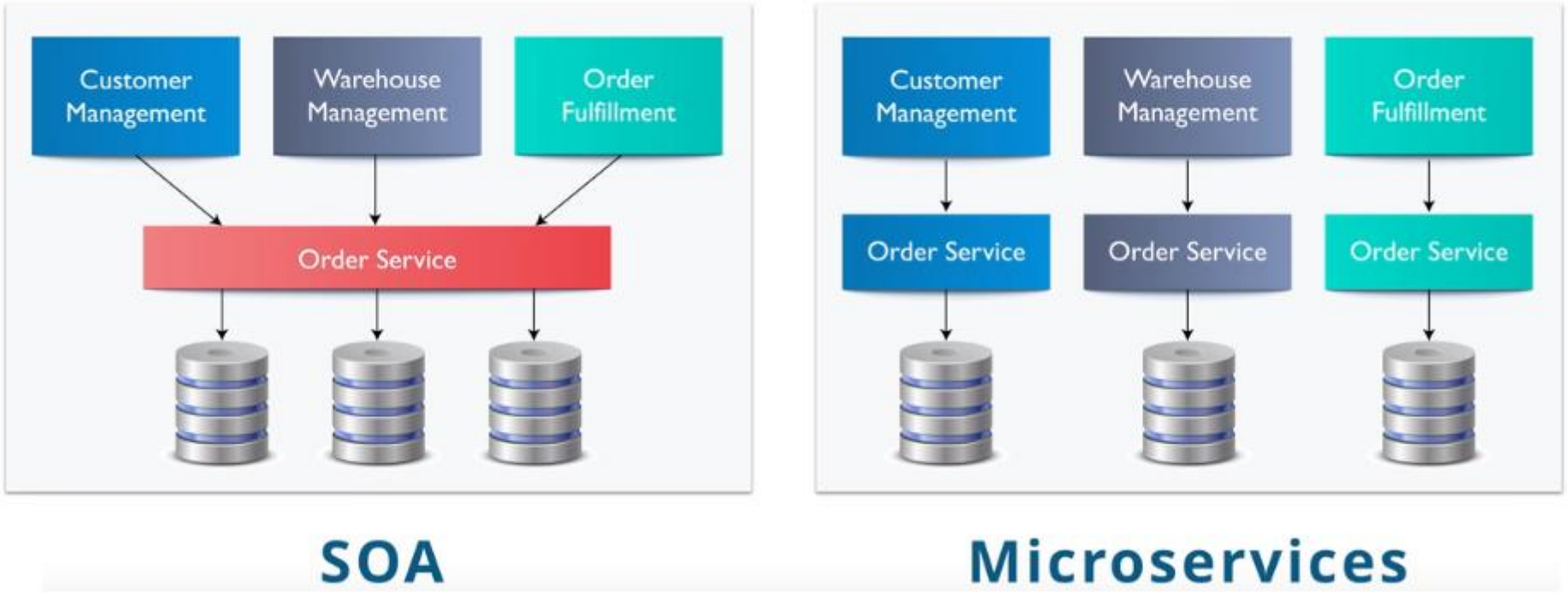


https://docs.docker.com/

Docker Swarmkit

- **Swarmkit**: Cluster management and orchestration features in Docker Engine 1.12 or later.
- It turns a pool of Docker hosts into a single, virtual host.
- When Swarmkit is enabled, we call Docker Engine running in swarm mode.
- See the feature list: Swarm mode overview.
- This project focuses on micro-service architecture
- It supports service reconciliation, load balancing, service discovery, built-in certificate rotation, etc.

Microservice architecture



https://stackoverflow.com/

Docker Nodes?

Machines running *SwarmKit* can be grouped together in order to form a *Swarm*, coordinating tasks with each other. Once a machine joins, it becomes a *Swarm Node*. Nodes can either be *worker* nodes or *manager* nodes.

- **Worker Nodes** are responsible for running Tasks using an *Executor*. *SwarmKit* comes with a default *Docker Container Executor* that can be easily swapped out.

- **Manager Nodes** on the other hand accept specifications from the user and are responsible for reconciling the desired state with the actual cluster state.

Features of SwarmKit

Some of *SwarmKit´s* main features are:

- Orchestration
- Scheduling
- Cluster management

Orchestration

- **Desired State Reconciliation**: *SwarmKit* constantly compares the desired state against the current cluster state and reconciles the two if necessary. For instance, if a node fails, *SwarmKit* reschedules its tasks onto a different node.
- **Service Types**: There are different types of services. The project currently ships with two of them out of the box
  - ✓ **Replicated Services** are scaled to the desired number of replicas.
  - ✓ **Global Services** run one task on every available node in the cluster.

Scheduling

node attribute	matches	example
node.id	node's ID	node.id == 2ivku8v2gvtg4
node.hostname	node's hostname	node.hostname != node-2
node.ip	node's IP address	node.ip != 172.19.17.0/24
node.role	node's manager or worker role	node.role == manager
node.platform.os	node's operating system	node.platform.os == linux
node.platform.arch	node's architecture	node.platform.arch == x86_64
node.labels	node's labels added by cluster admins	node.labels.security == high
engine.labels	Docker Engine's labels	engine.labels.operatingsystem == ubuntu 14.04

Cluster Management

- **State Store**: Manager nodes maintain a strongly consistent, replicated (Raft based) and extremely fast (in-memory reads) view of the cluster which allows them to make quick scheduling decisions while tolerating failures.

- **Topology Management**: Node roles (*Worker* / *Manager*) can be dynamically changed through API/CLI calls.

- **Node Management**: An operator can alter the desired availability of a node: Setting it to *Paused* will prevent any further tasks from being scheduled to it while *Drained* will have the same effect while also re-scheduling its tasks somewhere else (mostly for maintenance scenarios).

Docker Swarm

Setup using the hosted discovery service

- Create a cluster:
  - $ swarm create
- Add nodes to a cluster:
  - $ swarm join --add=<node_ip>
- Start Swarm
  - $ swarm manage --addr=<swarm_ip>

Scheduling

Scheduling in Docker Swarm relies on filters and strategies

- spread
- random
- binpack

A manager node, schedules the tasks on a set of worker nodes.

A survey of Docker Swarm scheduling strategies

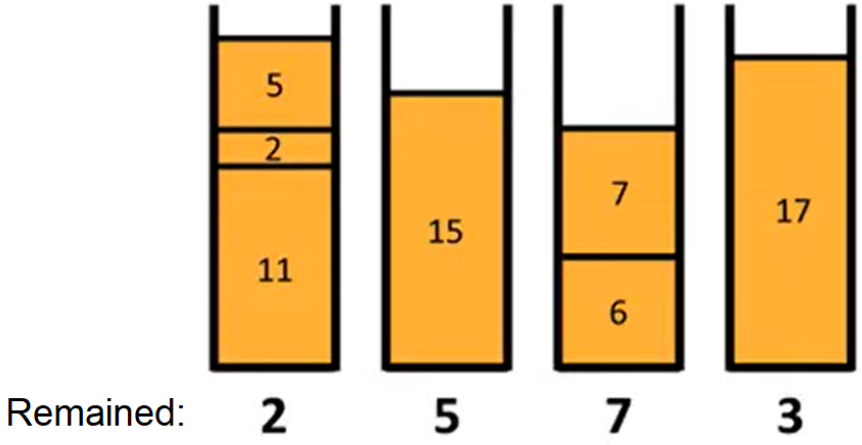
Anwar Hassen at Aalto University

Scheduling (cont.)

- Consider that there is a client that gets connected to the server farm.
- Consider the following scenario in which the manager of the server farm receives requests from the client as follows:
  - It is supposed to schedule the received tasks on the worker nodes. The tasks have different execution times. Manager receives a list consisting of execution times of tasks that should be sent to the worker nodes.
  - The execution times are 16, 7, 20, 10, 11, 22, and 12 minutes. Each task is allowed to have the resources of worker node for 25 minutes.
  - Manager schedules the tasks by Bin-Packing Algorithm. This algorithm packs the tasks in two methods: first-fit and best-fit. What is the difference between two methods? Please implement both methods.

Scheduling (cont.)

Use the first-fit algorithm to pack the tasks on the threads. The execution times are:
11, 2, 15, 5, 6, 17, and 7



Scheduling (cont.)

- https://gist.githubusercontent.com/vfarcic/750fc4117bad9d8619004081af171896/raw/3134886e92c09f47ac37edfc151d745fa8e4235e/02-docker-swarm.sh

References

- https://www.docker.com/
- Bernstein, D., “Containers and cloud: From lxc to docker to kubernetes,” *IEEE Cloud Computing*, *1*(3), pp.81-84, 2014.
- Scott McCarty. 2018. A Practical Introduction to Container Terminology. https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/
- http://docker.atbaker.me/index.html
- Boza, E. F., Abad, C. L., Narayanan, S. P., Balasubramanian, B., & Jang, M. (2019, December). A Case for Performance-Aware Deployment of Containers. In *Proceedings of the 5th International Workshop on Container Technologies and Container Clouds* (pp. 25-30).
- https://dockerbook.com/TheDockerBook_sample.pdf