Combining TOSCA and Docker

Luca Rinaldi

University of Pisa

June 2017

Table of Contents

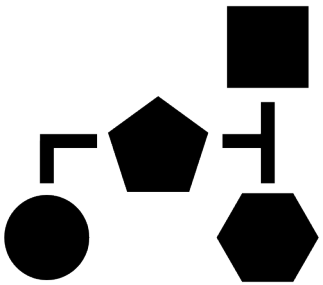
Table of Contents

Software deployment and Orchestration

?

Software deployment and Orchestration

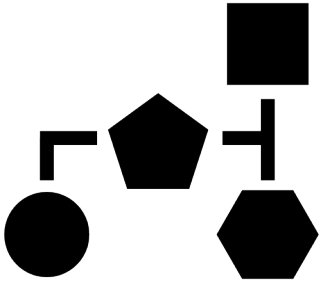
?



Application specification

Software deployment and Orchestration

?

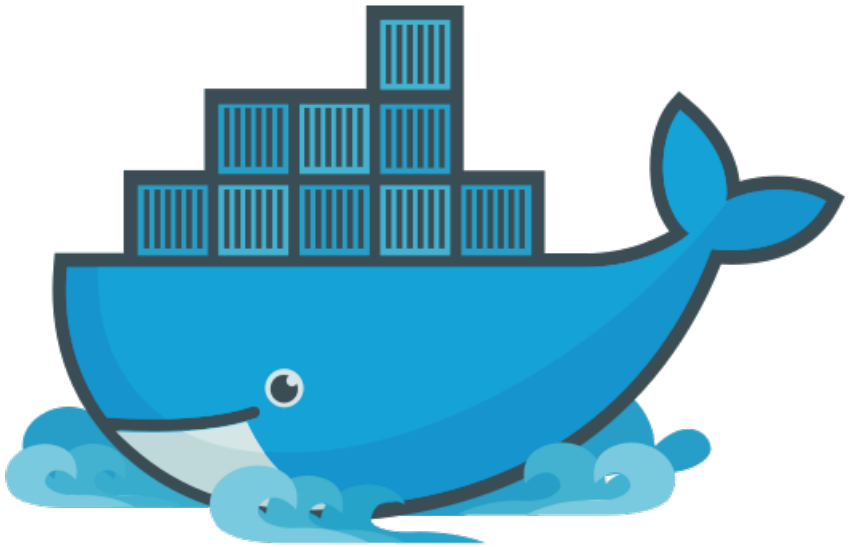


Application specification

Application orchestration

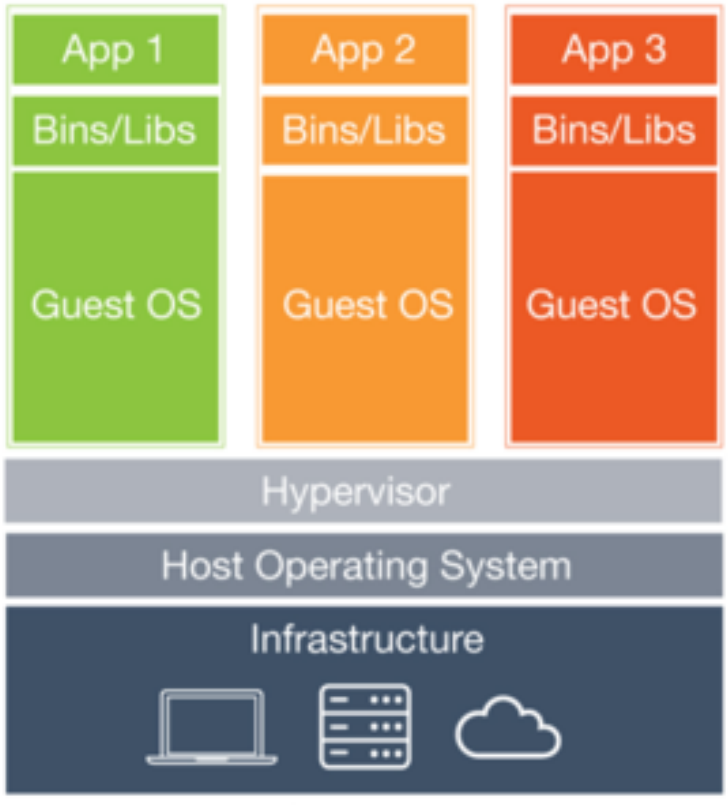
Table of Contents

Docker

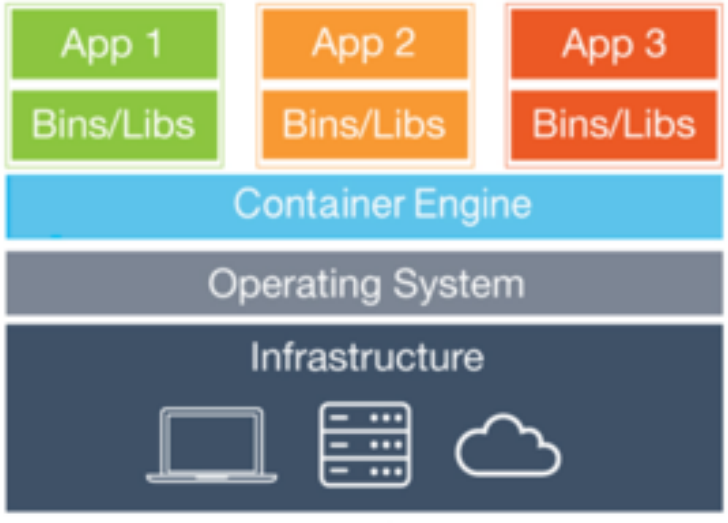


Docker is a tool that can package an application and its dependencies in a virtual container that can run on any Linux server.

VM vs Docker



Hypervisor-based Virtualization



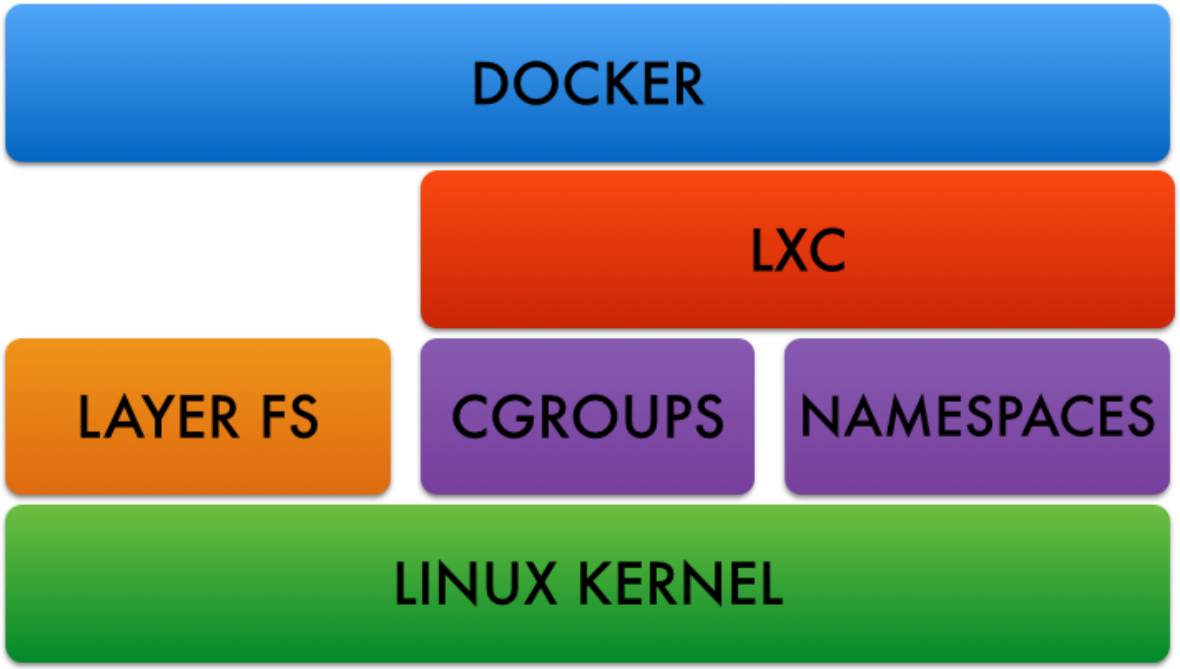
Container virtualization

Main concepts

Main concepts of the Docker platform

- **Dockerfile**, a script to generate a Docker Image
- **Docker Image**, a separated file-system with all the binaries and library
- **Docker Container**, running instance of a Docker Image
- **Docker Volume**, a persistent data storage system
- **Docker Hub**, a public database of Docker Images

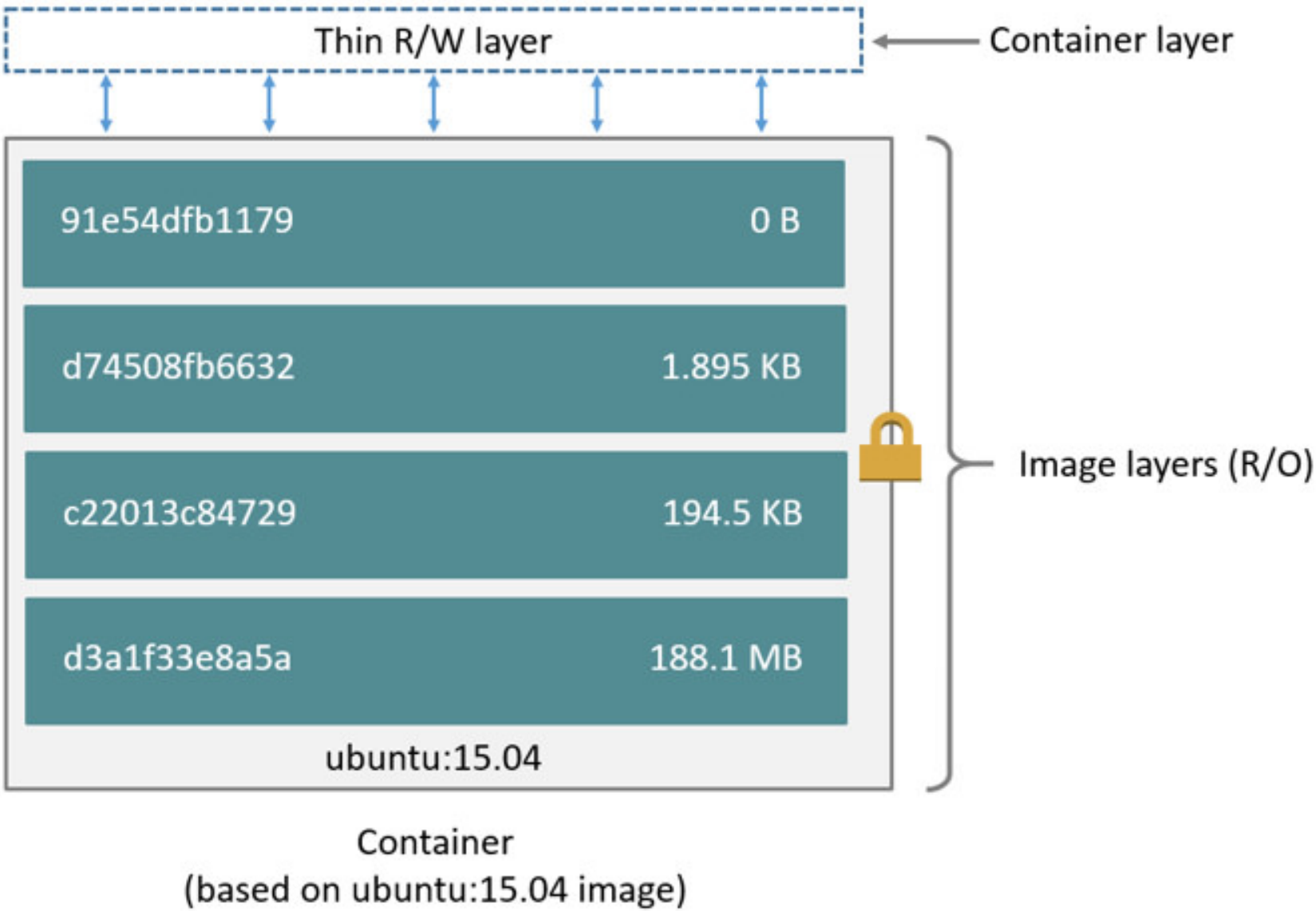
Architecture of Docker



- **LXC**, an operating-system-level virtualization method
- **Layer file-system**, a Union file system

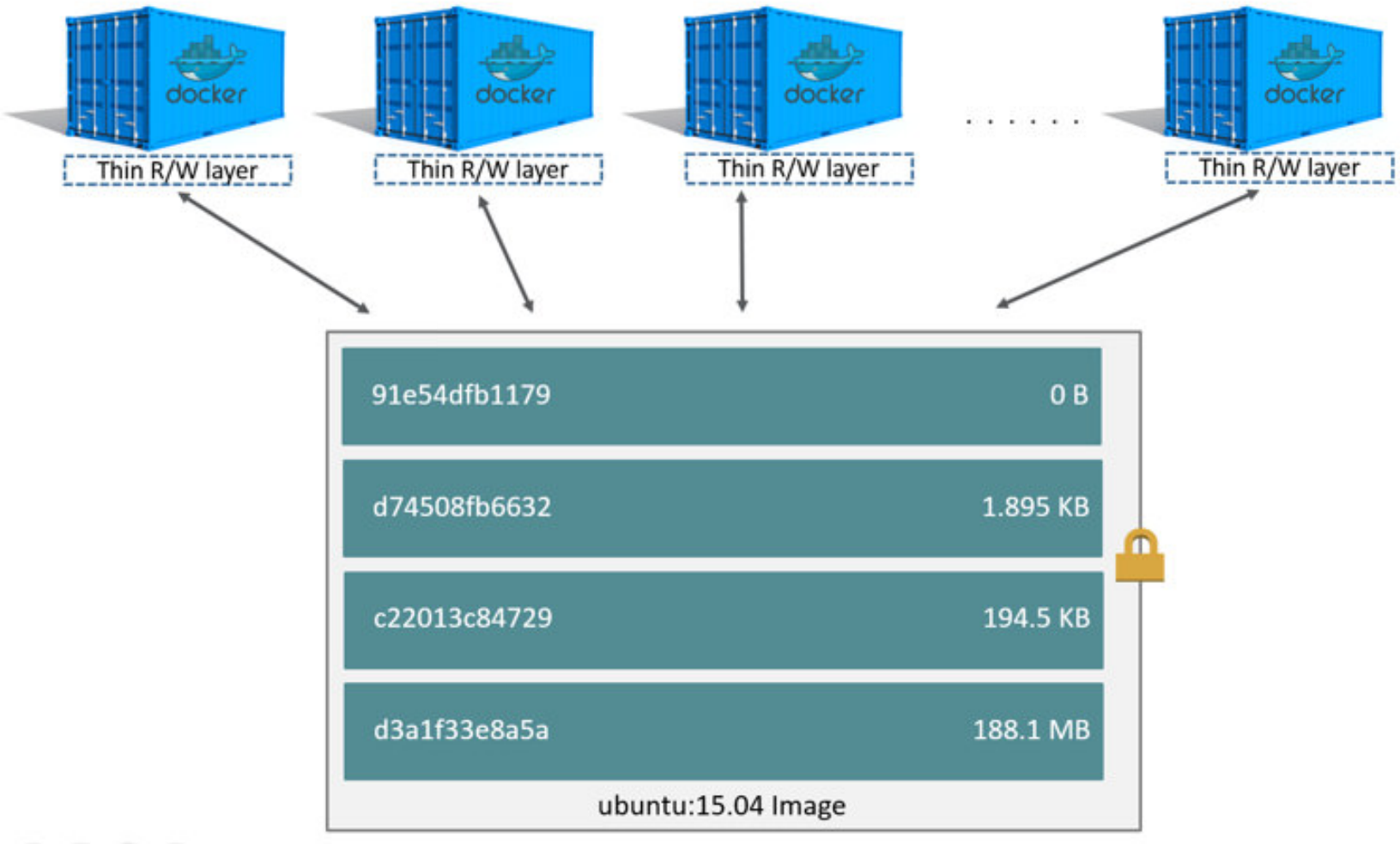
Layer file-system

Each layer is only a set of differences from the layer before it.

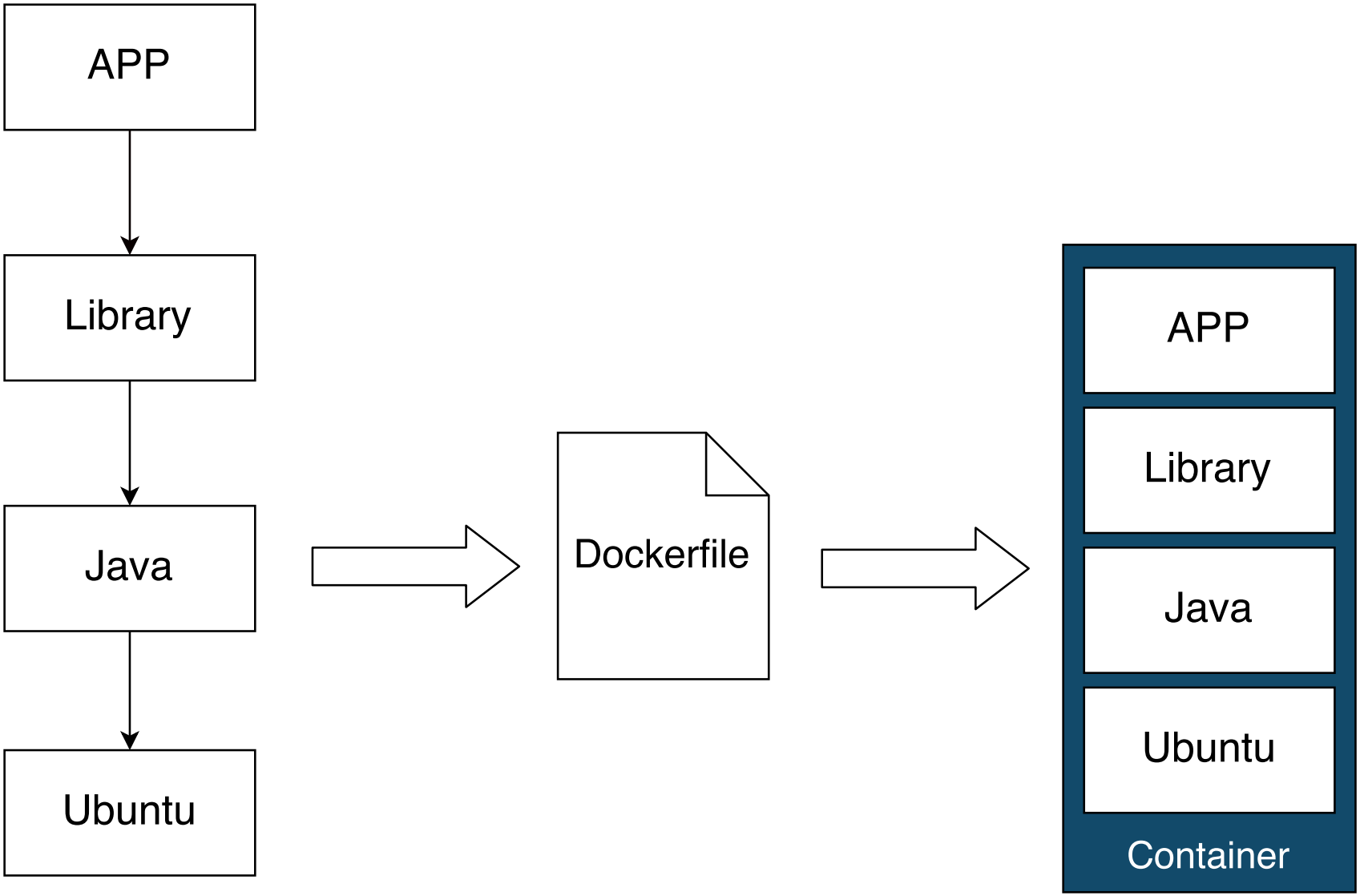


Layer file-system

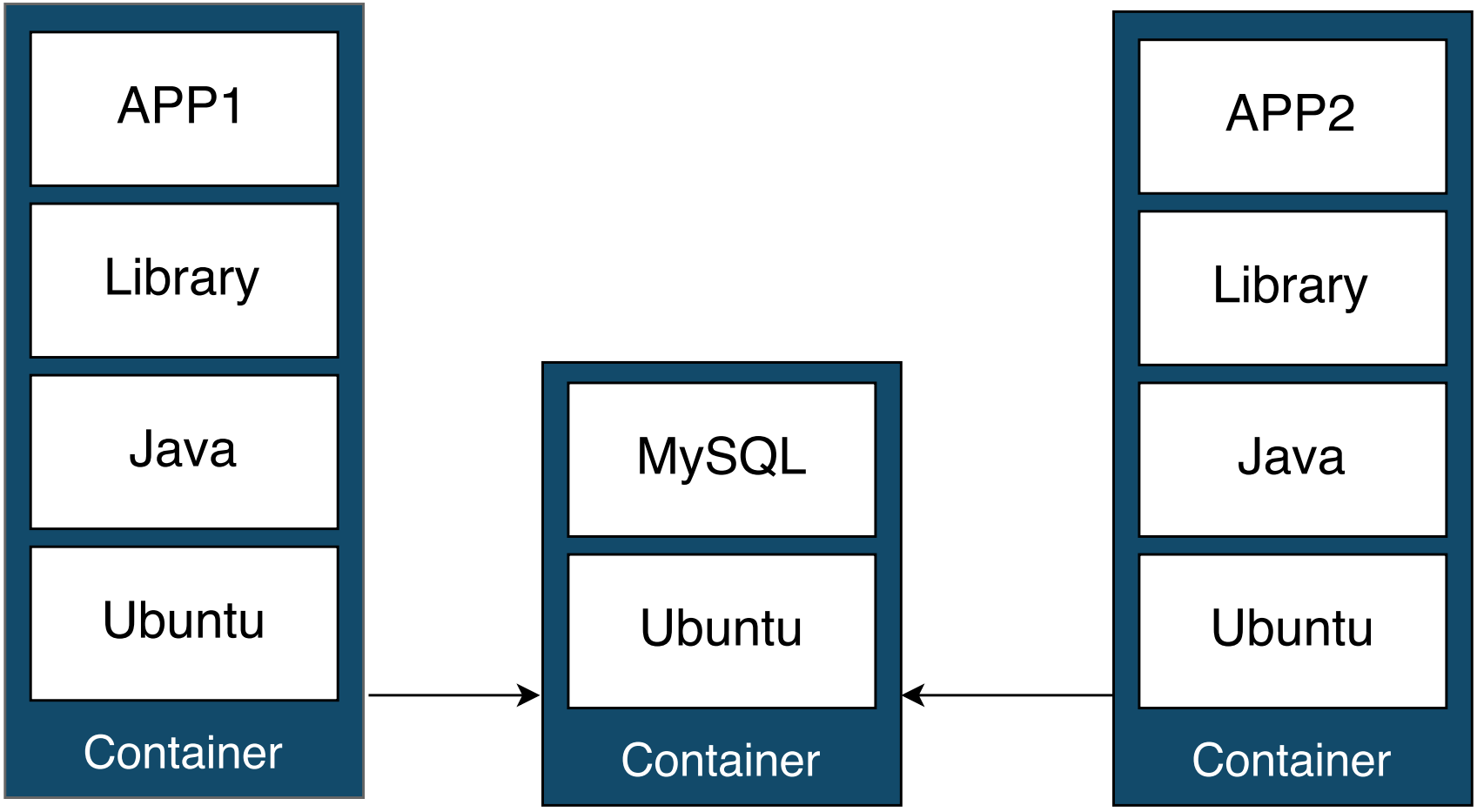
Copy-on-write permits to share the same layer between more containers.



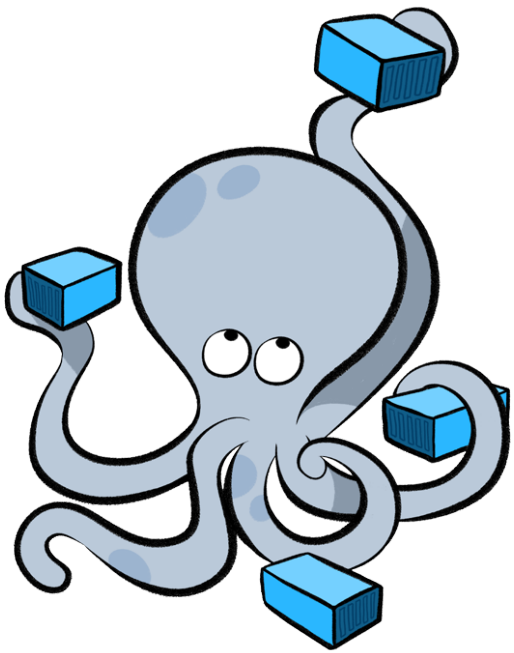
How use Docker



Multi-container application



Docker Compose



```
version: "2"
services:
  web:
    build: .
    ports:
     - "5000:5000"
    volumes:
     - .:/code
  redis:
    image: "redis:alpine"
```

Problems

- The container hide the components that it contains

Problems

- The container hide the components that it contains
- Poor application orchestration

Problems

- The container hide the components that it contains
- Poor application orchestration
- Docker containers as "minimal orchestration entities"

Table of Contents

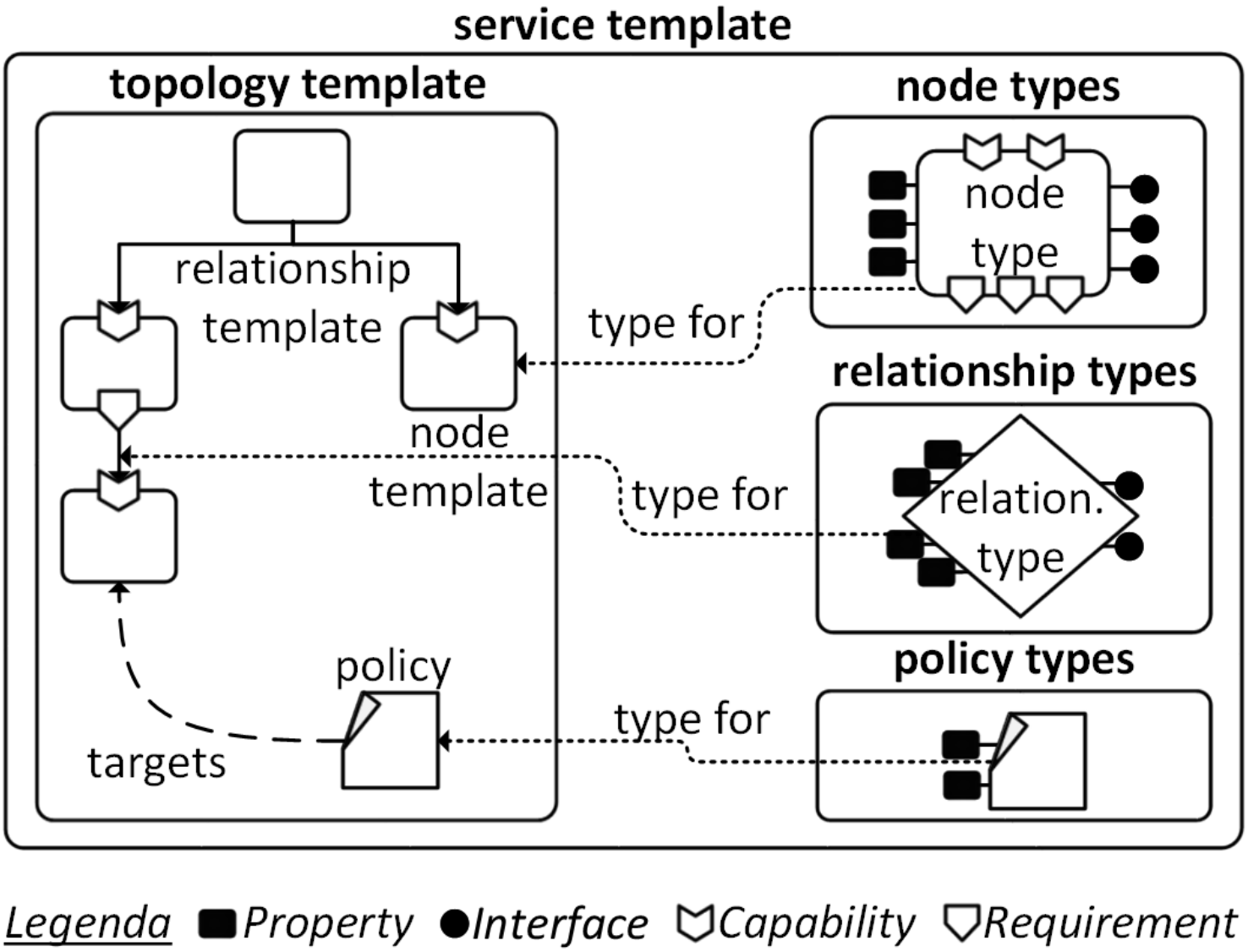
TOSCA

OASIS standard meta-language to describe the topology of an application, with its components and relationships.

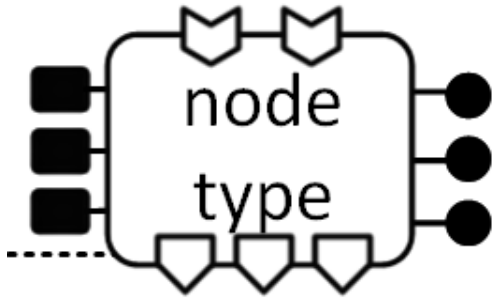
Main concepts

- YAML-based description
- CSAR archive containing TOSCA specs and executable artifacts
- Declarative processing

TOSCA description

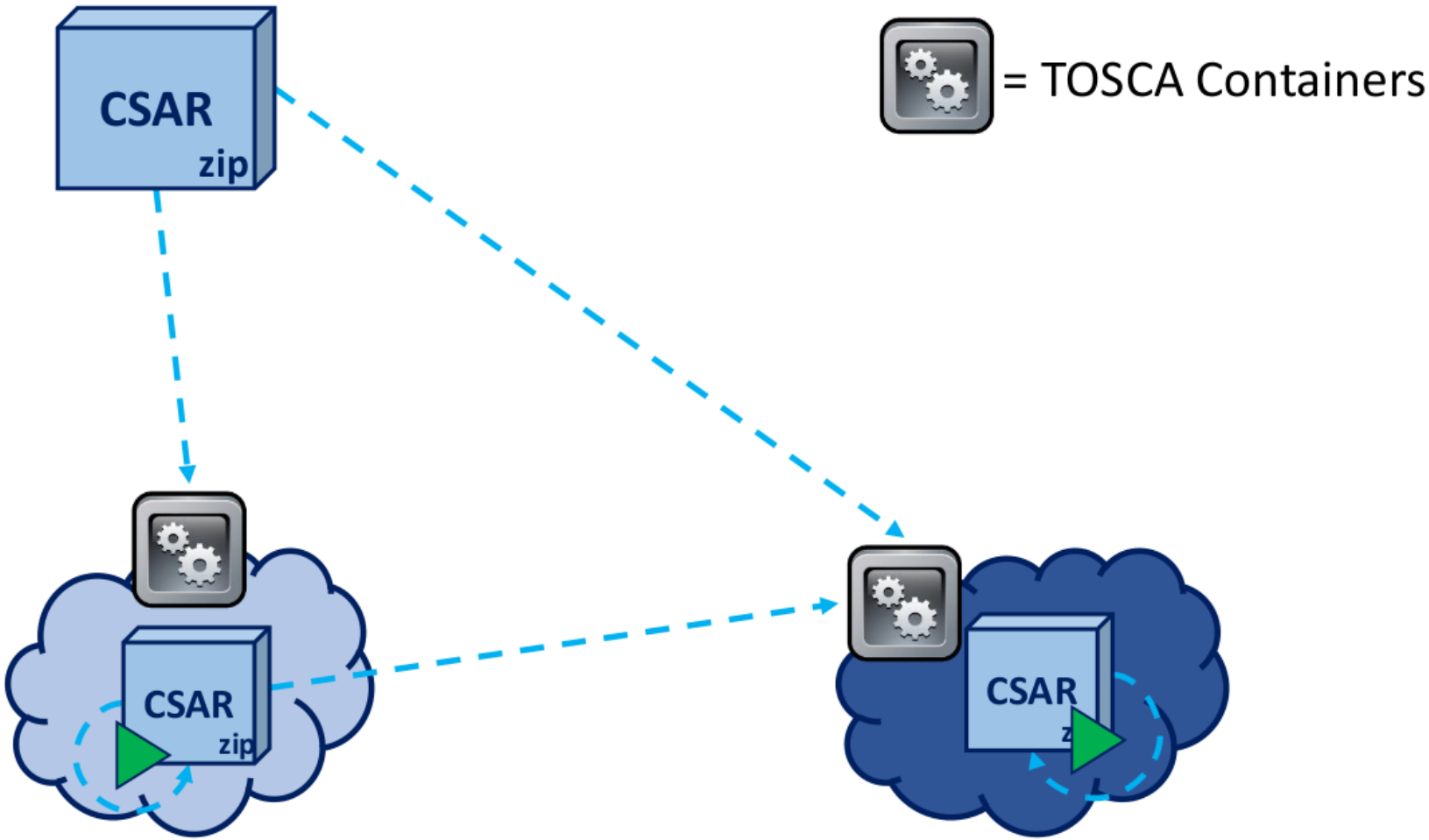


Node type

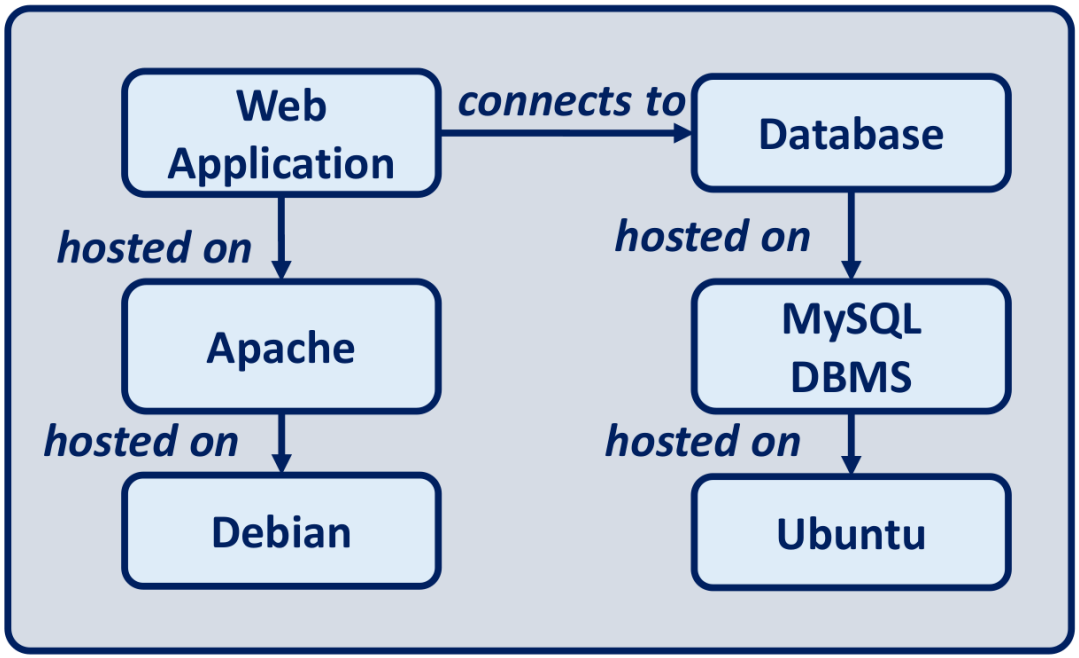


- **requirements**, what a component requires
- **capabilities**, what a component offers
- **properties**, descriptive information about a component
- **interfaces**, operations to deploy and manage a component
- **artifacts**, installables/executables/data implementing interface operations

How it works

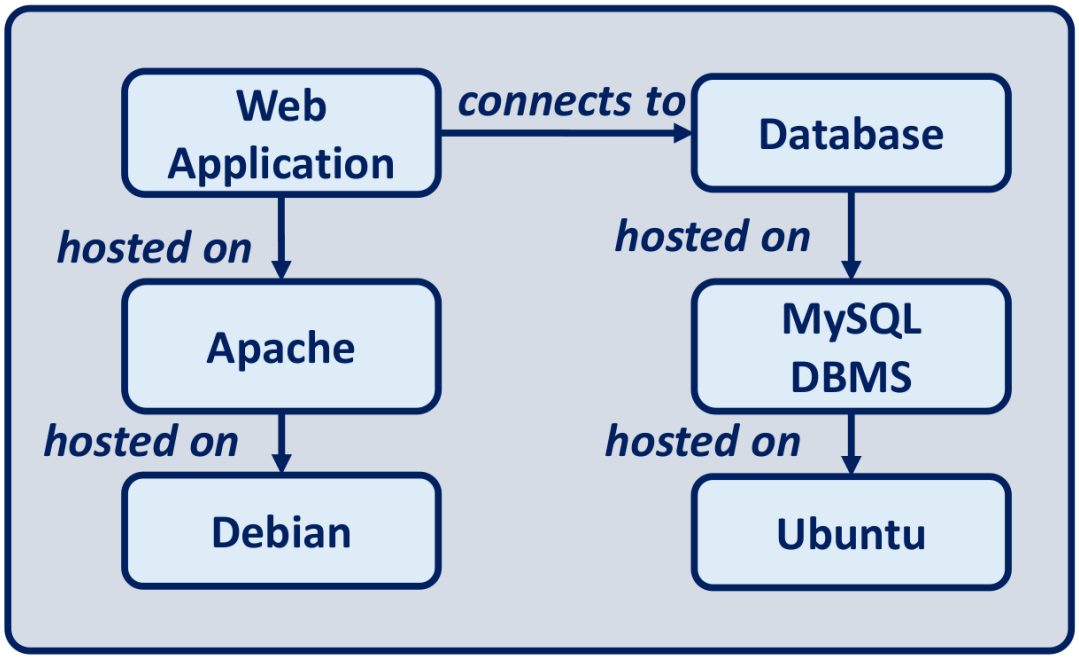


Declarative processing - A possible execution



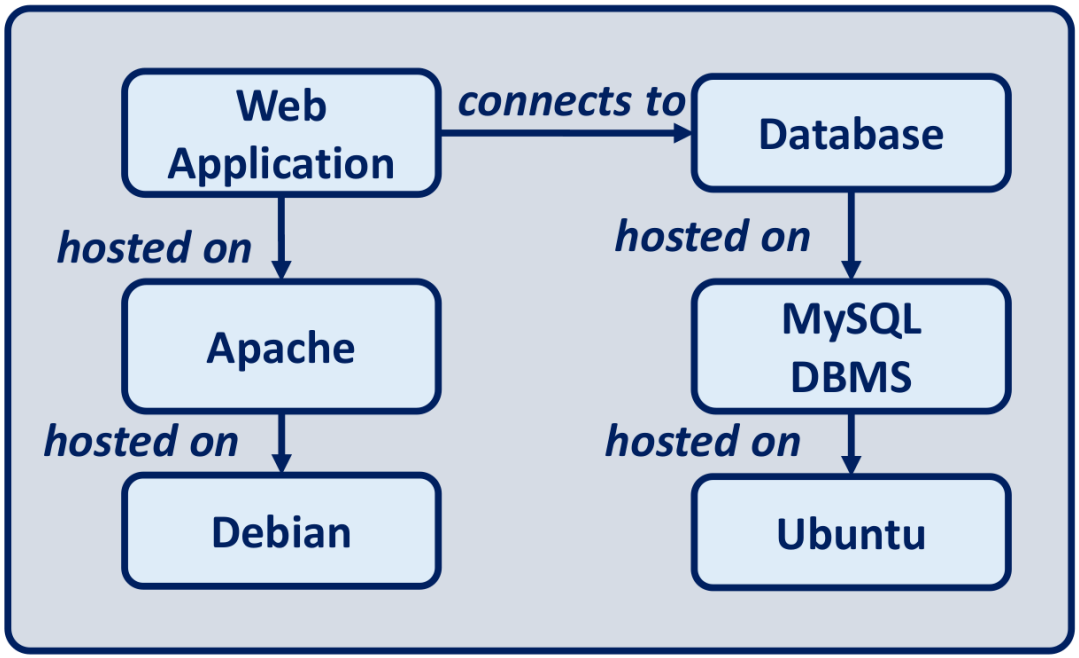
1. Debian, Ubuntu

Declarative processing - A possible execution



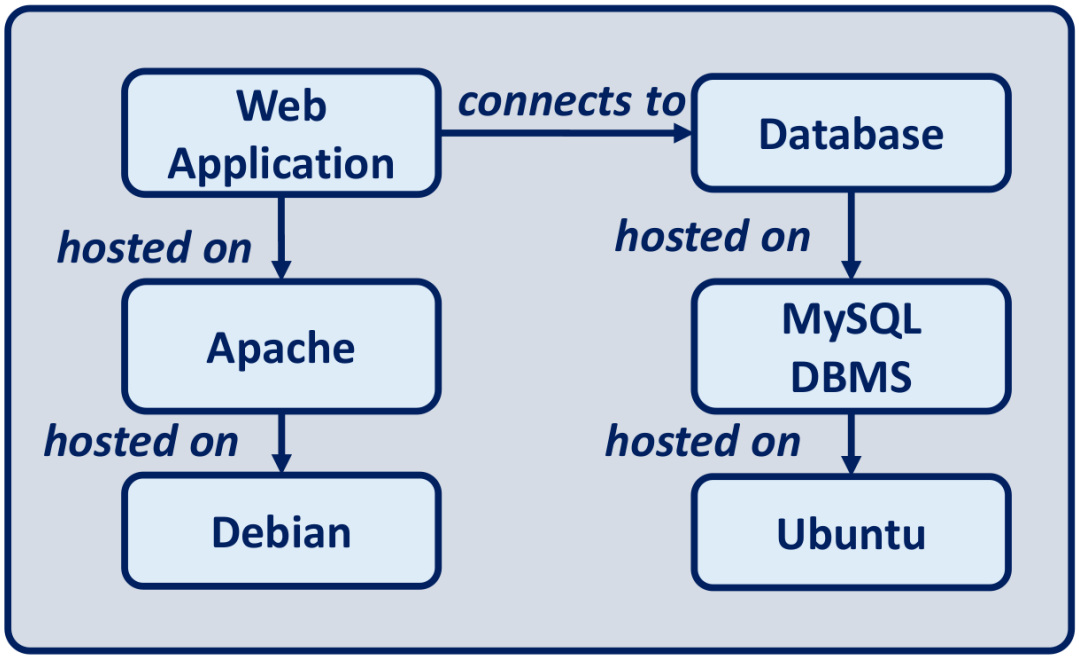
1. Debian, Ubuntu
2. Apache, MySQL

Declarative processing - A possible execution



1. Debian, Ubuntu
2. Apache, MySQL
3. Database

Declarative processing - A possible execution



1. Debian, Ubuntu
2. Apache, MySQL
3. Database
4. Web Application

Problems

- Specs may be too verbose
- Lack of tools for supporting development of TOSCA apps
- Lack of engines for running TOSCA apps

Table of Contents

TosKer

An orchestration engine capable of deploying, on top of Docker, applications described in TOSCA YAML.

- TosKer inputs a TOSCA description of a multi-component application, and
- it automatically deploys and orchestrates such application on top of the Docker engine

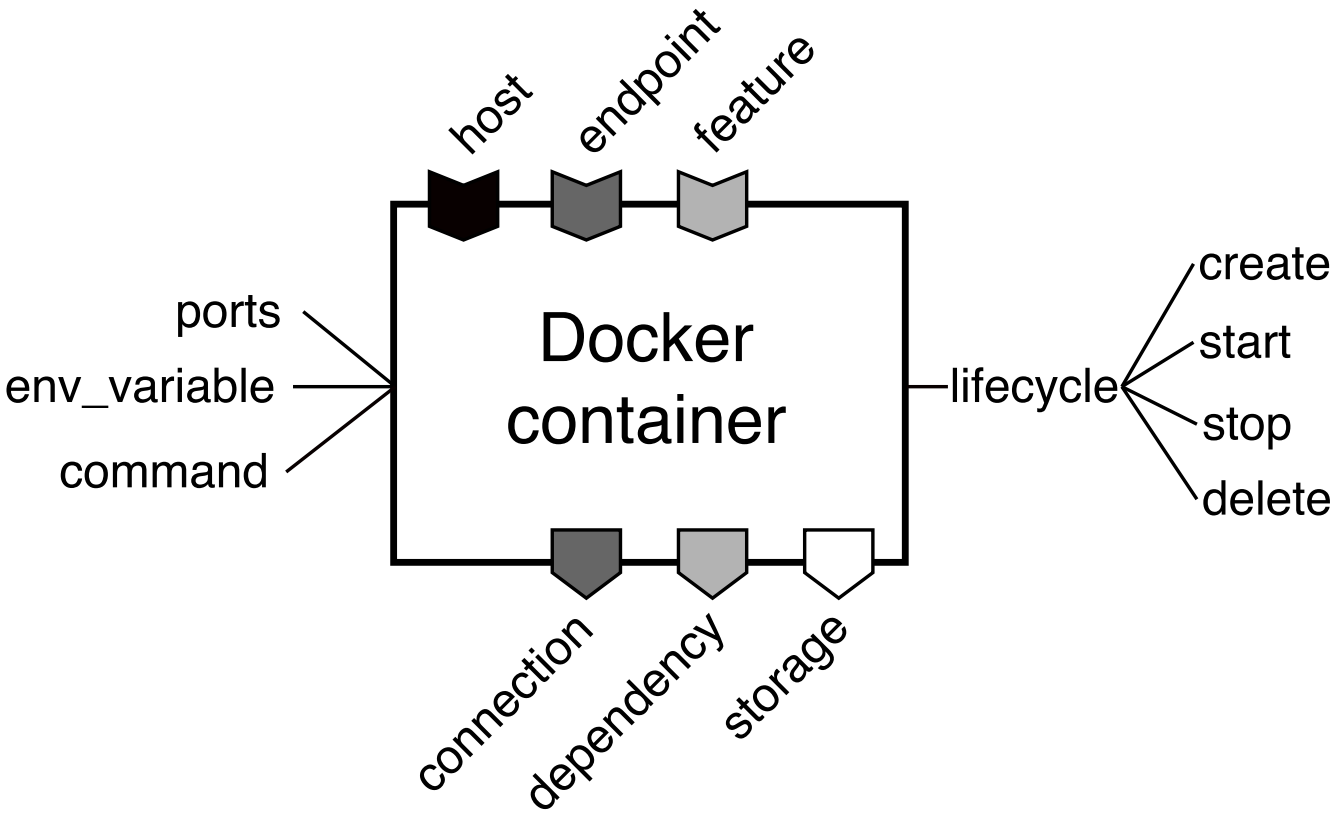
Describing applications with TosKer

- Applications are specified as a composition of the following components:
  - *Docker containers* `tosker.nodes.Container`
  - *Docker volumes* `tosker.nodes.Volume`
  - *Software* `tosker.nodes.Software`

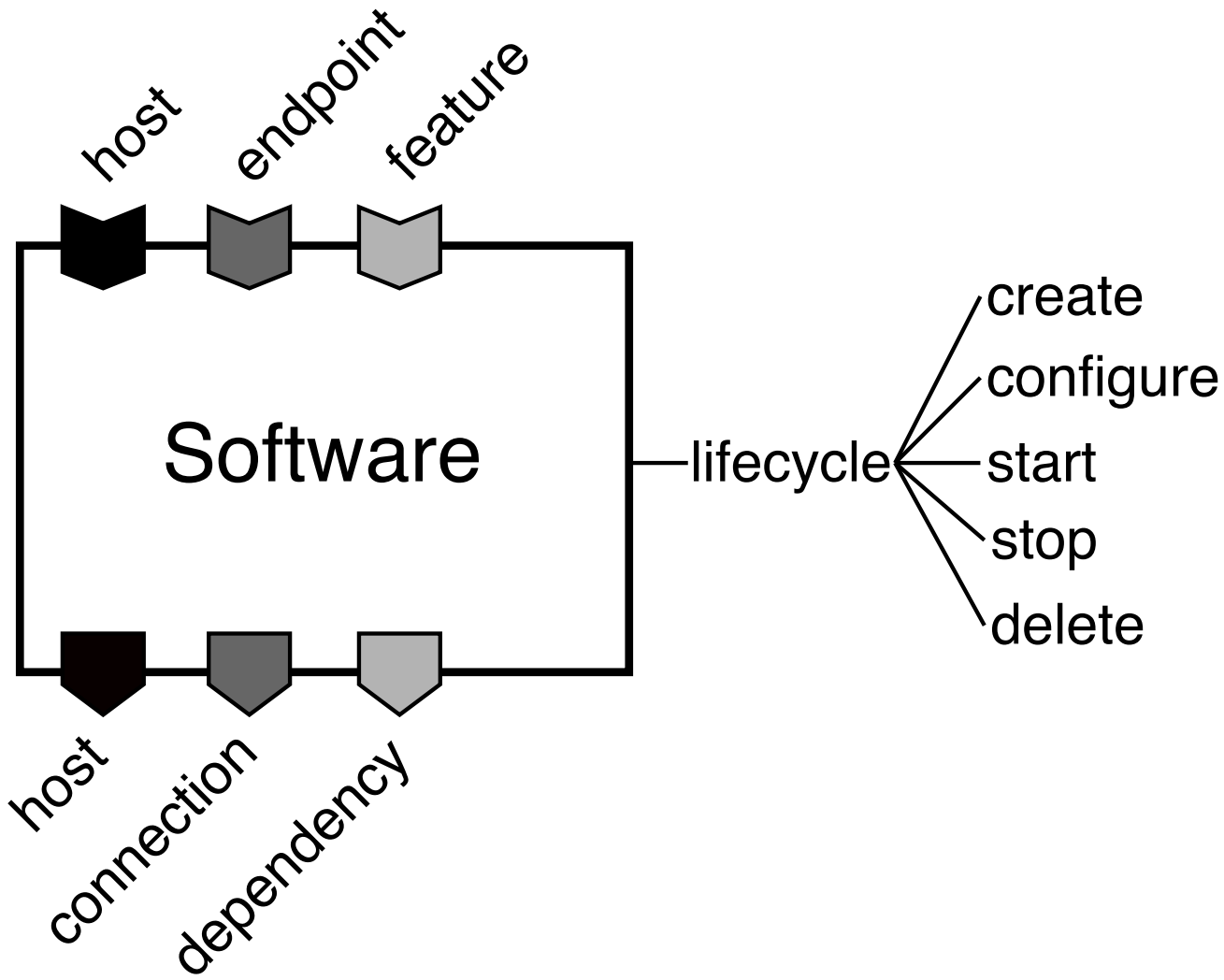
Describing applications with TosKer

- Applications are specified as a composition of the following components:
  - *Docker containers* `tosker.nodes.Container`
  - *Docker volumes* `tosker.nodes.Volume`
  - *Software* `tosker.nodes.Software`
- Components can be interconnected with the following relationships:
  - *hosted on* (`tosca.relationships.HostedOn`)
  - *connected to* (`tosca.relationships.ConnectsTo`)
  - *attached to* (`tosca.relationships.AttachesTo`)
  - *depending on* (`tosca.relationships.DependsOn`)

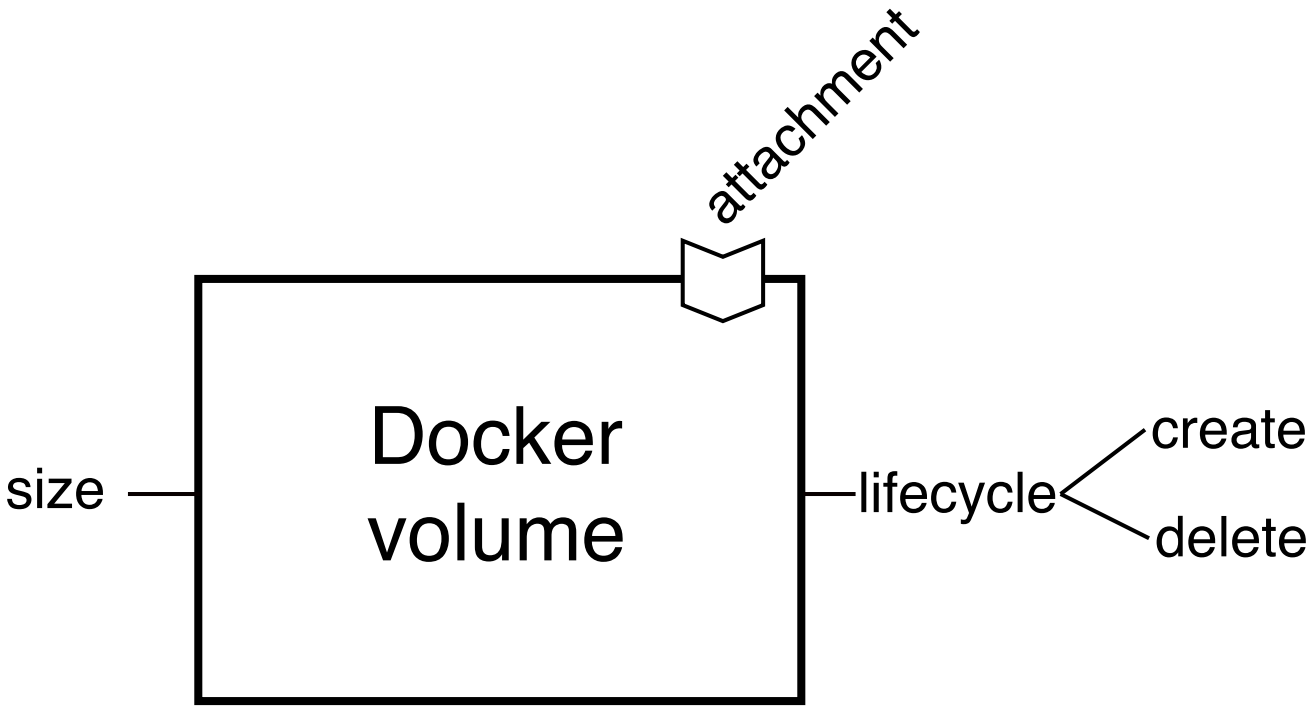
tosker.nodes.Container



tosker.nodes.Software



tosker.nodes.Volume

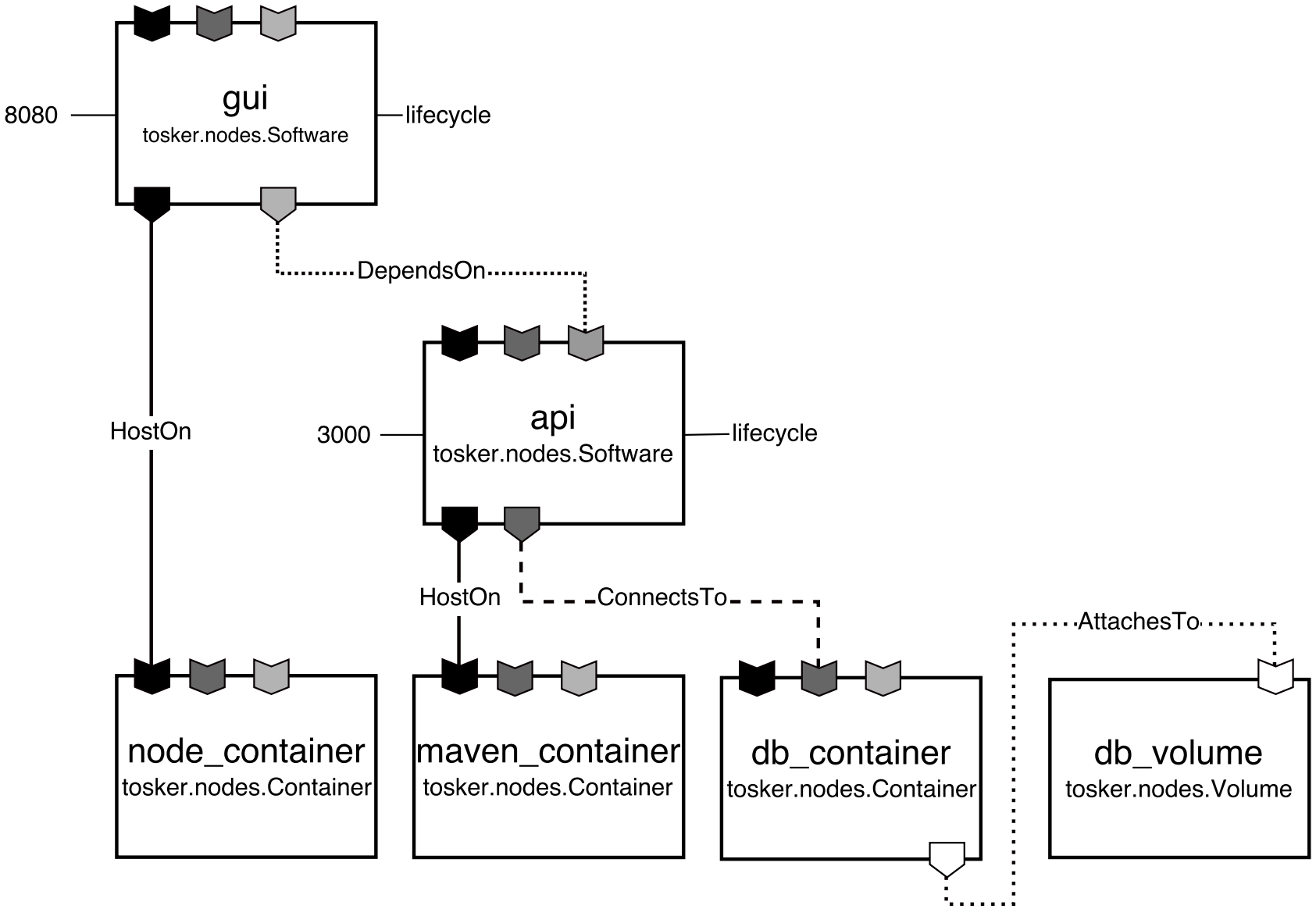


Topology constraints

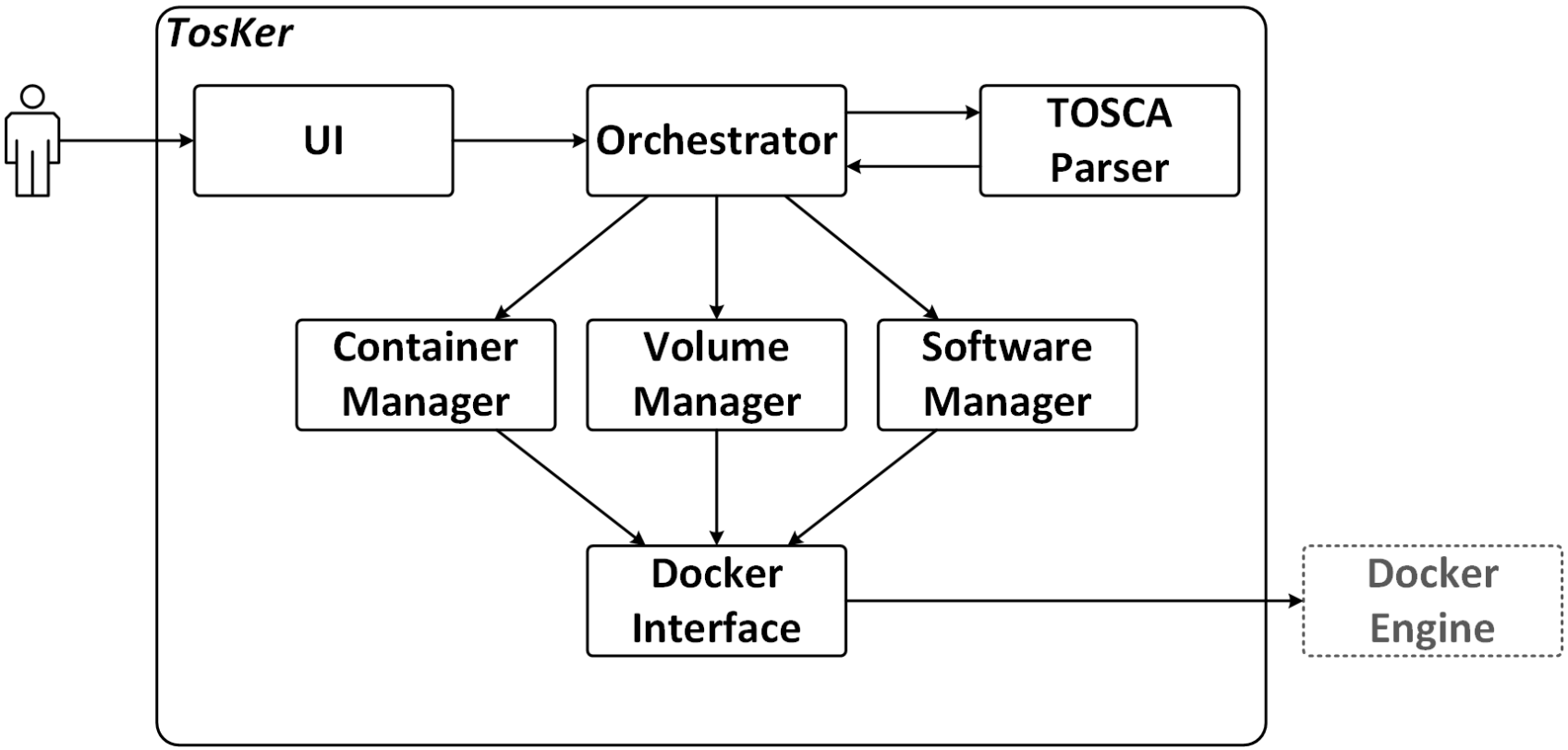
Each application topology must satisfy a set of constraints on how to compose nodes and relationships, e.g.,

- A *software* must be "*hosted on*" another *software* or a *Docker container*
- A *Docker container* and *Docker volume* cannot be "*hosted on*" other components
- Only *Docker containers* can be "*attached to*" *Docker volumes*

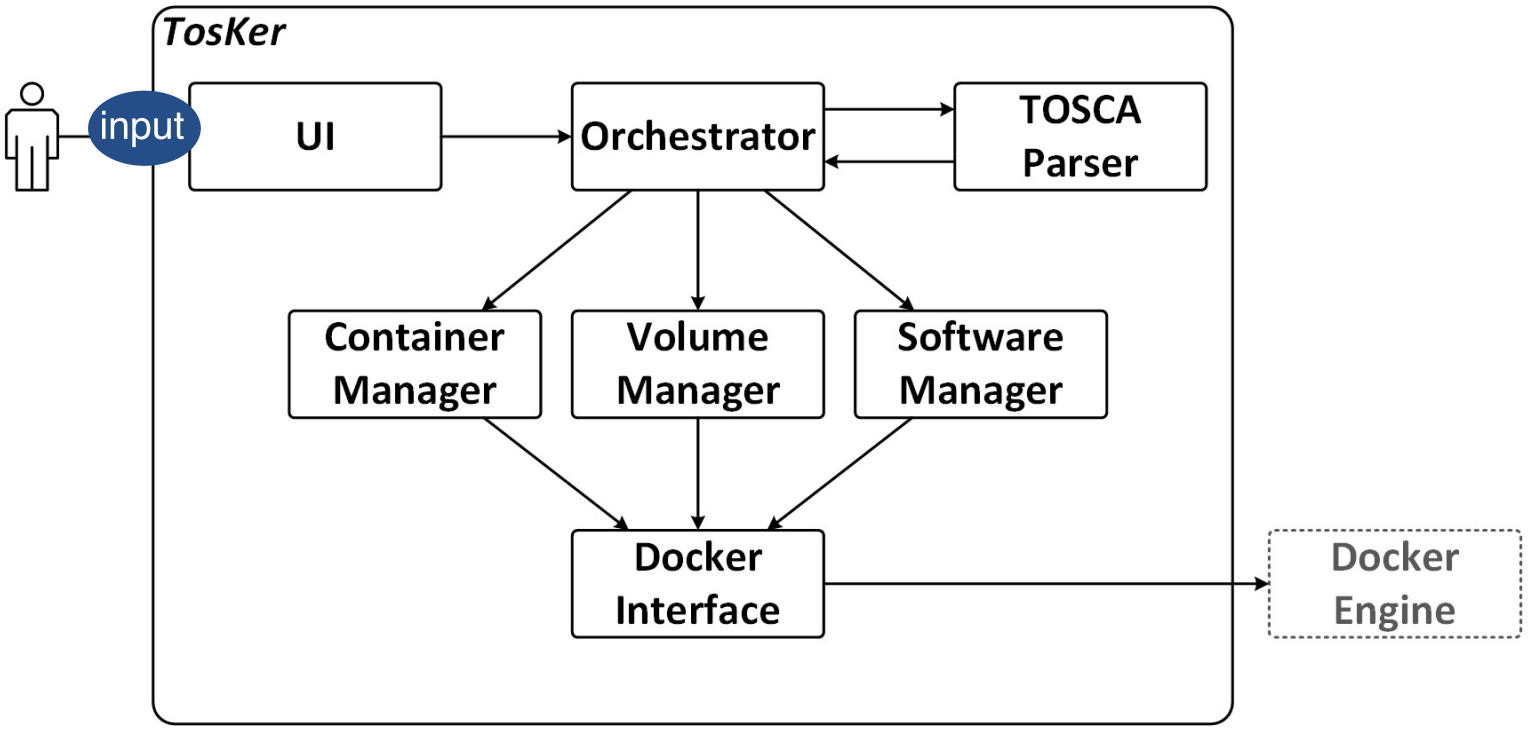
Case study: Thoughts



Architecture



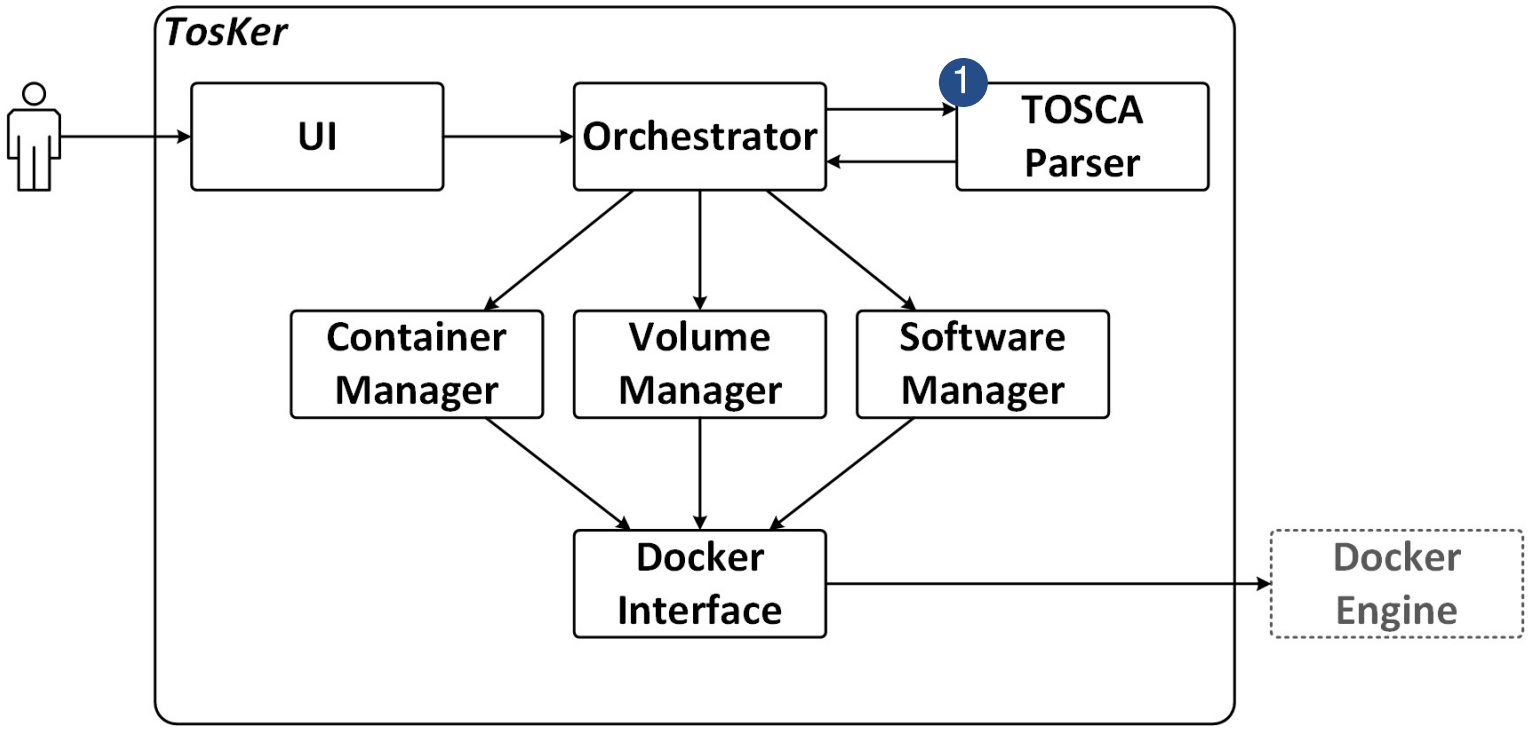
How TosKer orchestrates applications



The input of TosKer is

- a TOSCA application specified using TosKer types, and
- management operation(s) to perform.

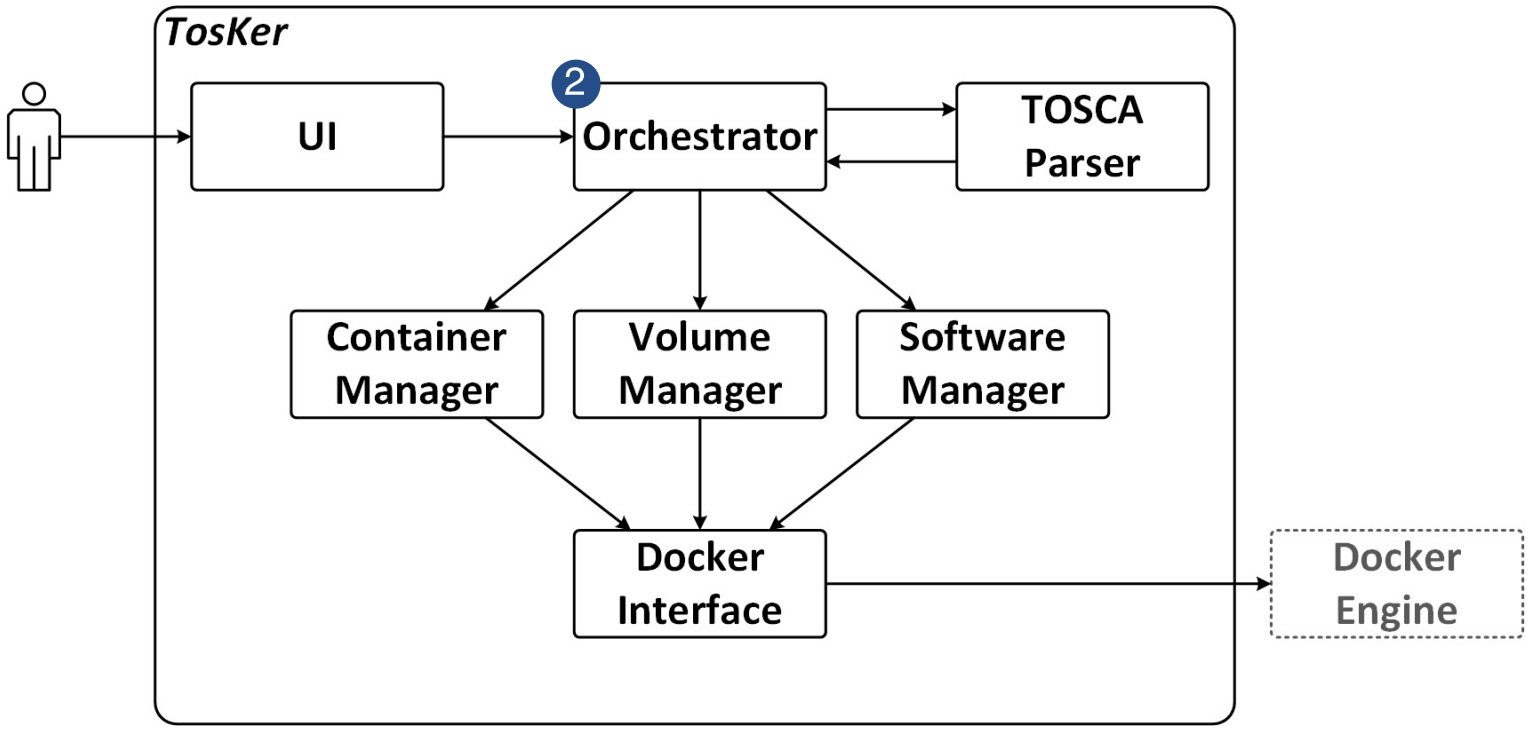
How TosKer orchestrates applications



TosKer

- parses and validates the TOSCA application, and
- executes a topological sorting algorithm.

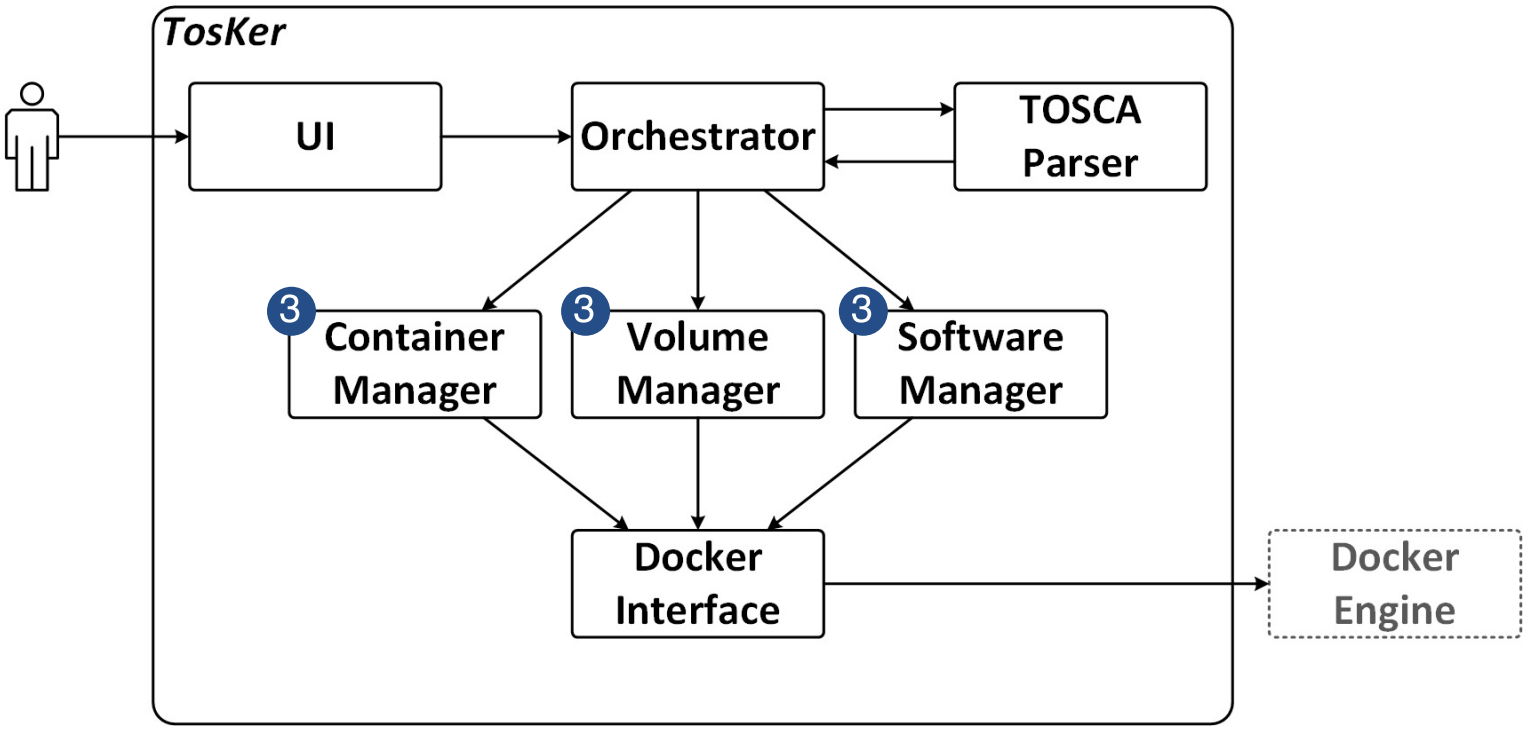
How TosKer orchestrates applications



TosKer

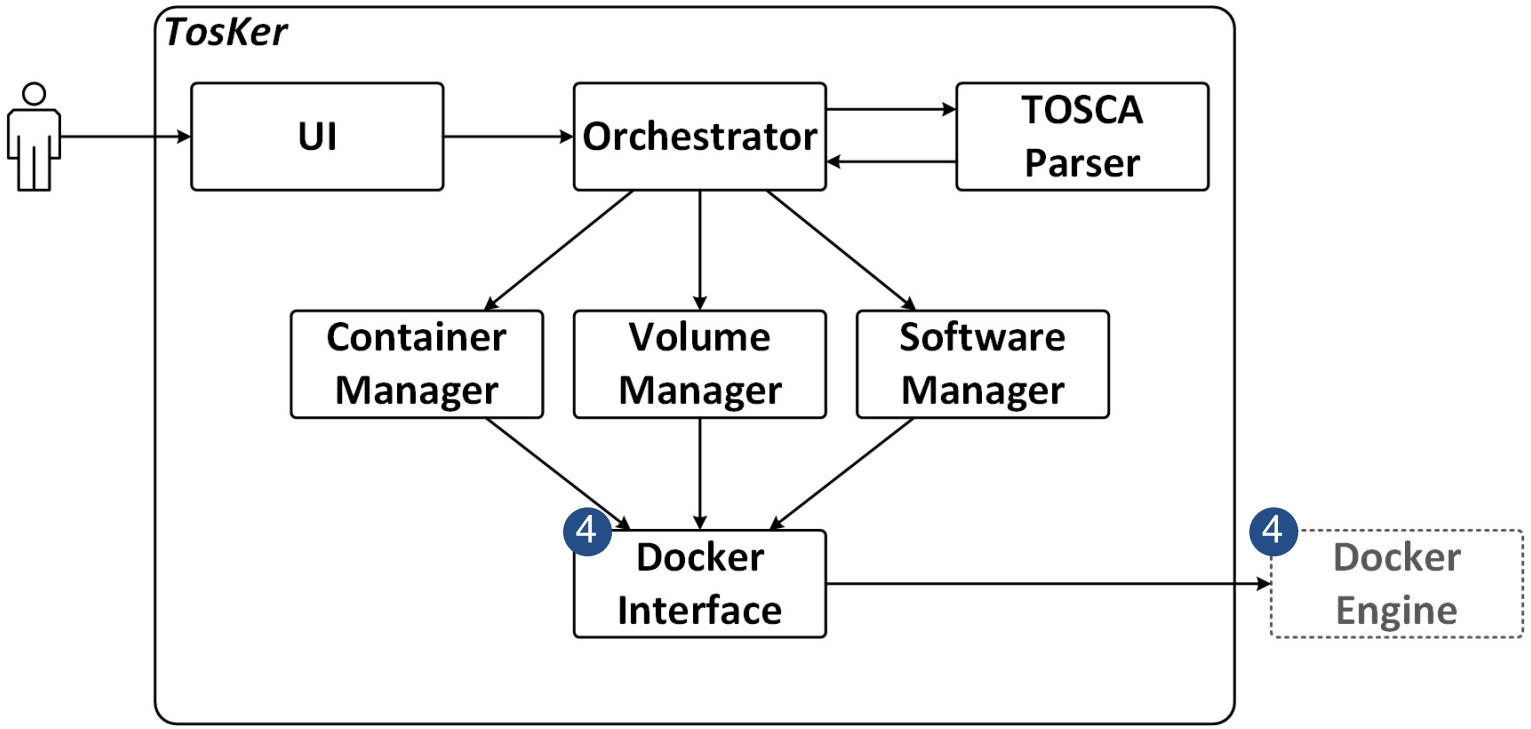
- scans the sorted application topology, and
- for each component, it calls a specific operation (e.g., create)

How TosKer orchestrates applications



Each manager is in charge of implementing/executing the invoked operation on a component...

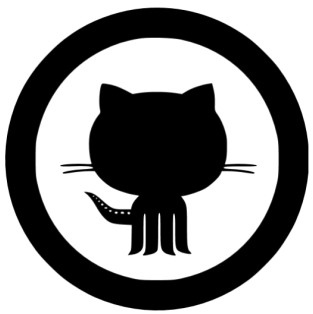
How TosKer orchestrates applications



...by properly invoking the Docker engine (through the Docker interface)

Implementation

Python



GitHub

MIT Licence

- **PyPI**: https://pypi.python.org/pypi/tosKer
  `pip install tosker`
- **GitHub**: https://github.com/di-unipi-socc/TosKer

Table of Contents

Conclusions

We focussed on the problem of orchestrating the deployment and management of composite cloud applications.

- Docker
- TOSCA YAML

Conclusions

We focussed on the problem of orchestrating the deployment and management of composite cloud applications.

- Docker
- TOSCA YAML

We presented the TosKer orchestration engine, which

- extends TOSCA by providing a Docker-based orchestration engine for TOSCA applications, and
- extends Docker by adding the capability of orchestrating software components together with Docker containers/volumes

Future work

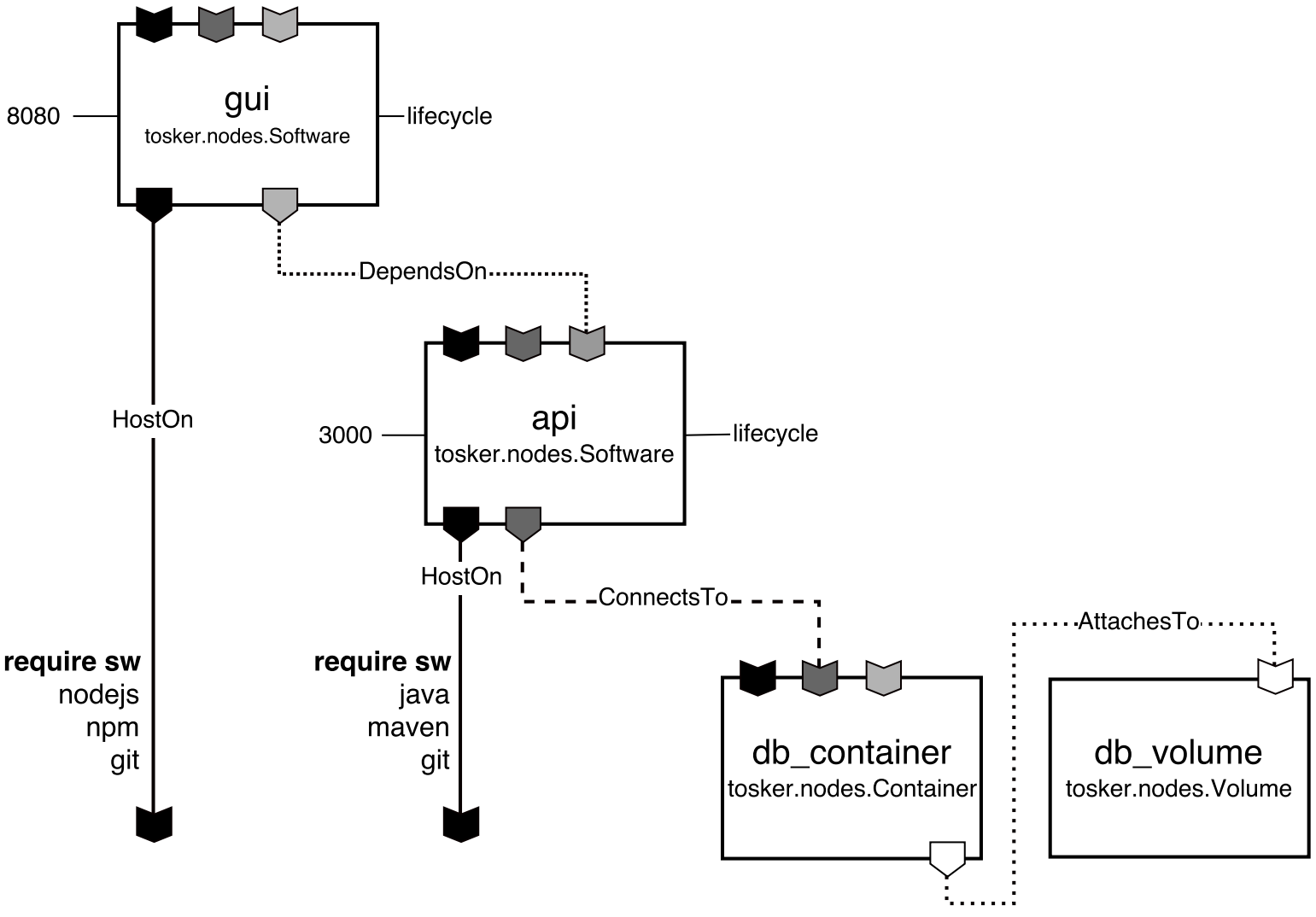
- Automatically determine the Docker containers needed to effectively run an application (**Dockerizer**)
- Support cluster of workstations and external cloud services
- Integrate TosKer with fault-aware management protocols

Focus on **Dockerizer**

Dockerizer, a completer of TosKer specifications.

Focus on **Dockerizer**

Dockerizer, a completer of TosKer specifications.



Thank You

Q&A