Orchestrating applications with TOSCA and Docker

Luca Rinaldi

MSc in Computer Science and Networking
University of Pisa and Sant'Anna School of Advanced Studies
A.Y. 2015/2016

Supervisors: Antonio Brogi, Jacopo Soldani

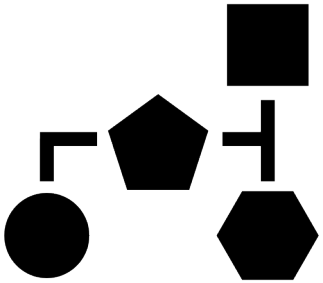
Table of Contents

Context: Managing composite cloud applications

?

Context: Managing composite cloud applications

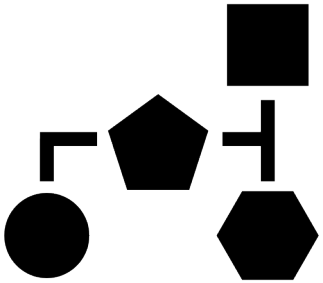
?



Application specification

Context: Managing composite cloud applications

?



Application specification

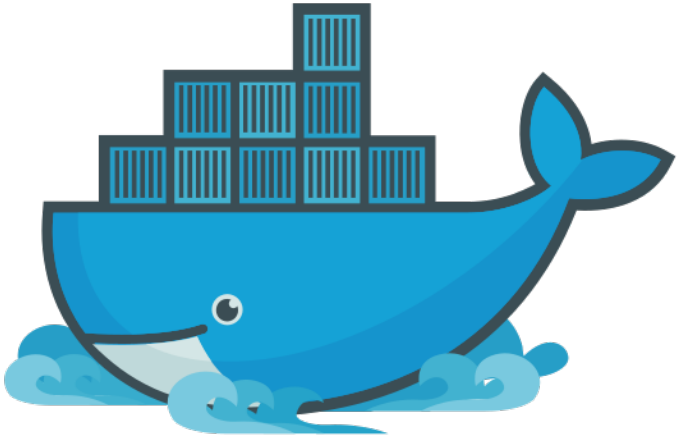
Application orchestration

Two orthogonal approaches

OASIS TOSCA

Two orthogonal approaches

OASIS TOSCA



Docker

Pro and cons of TOSCA and Docker

TOSCA

+ Well-documented standard
+ Application orchestration

Pro and cons of TOSCA and Docker

TOSCA

+ Well-documented standard
+ Application orchestration

– Too verbose
– Lack of engines

Pro and cons of TOSCA and Docker

TOSCA

+ Well-documented standard
+ Application orchestration

– Too verbose
– Lack of engines

Docker

+ Production ready tool
+ Repository of images

Pro and cons of TOSCA and Docker

TOSCA

+ Well-documented standard
+ Application orchestration

– Too verbose
– Lack of engines

Docker

+ Production ready tool
+ Repository of images

– Application orchestration
– “Only containers”

Objective

The objective of this thesis was to identify and develop a solution that takes the best of TOSCA and Docker.

Main objective

Design and prototype an orchestration engine capable of deploying multi-components applications.

- It inputs applications specified in TOSCA YAML
- It automatically manages applications by exploiting Docker

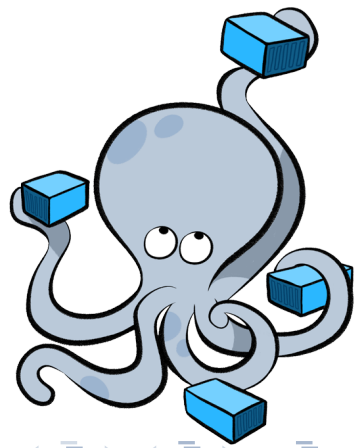
State of the art

TOSCA orchestrator

State of the art

TOSCA orchestrator

Docker orchestrator



State of the art

TOSCA orchestrator

apache brooklyn

Docker orchestrator

kubernetes

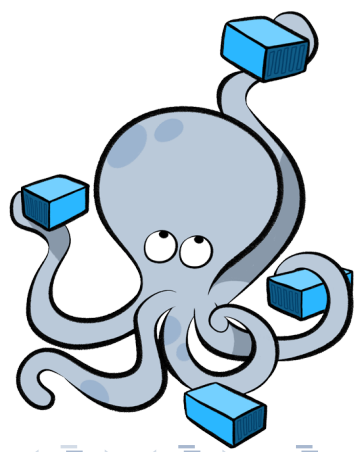


Table of Contents

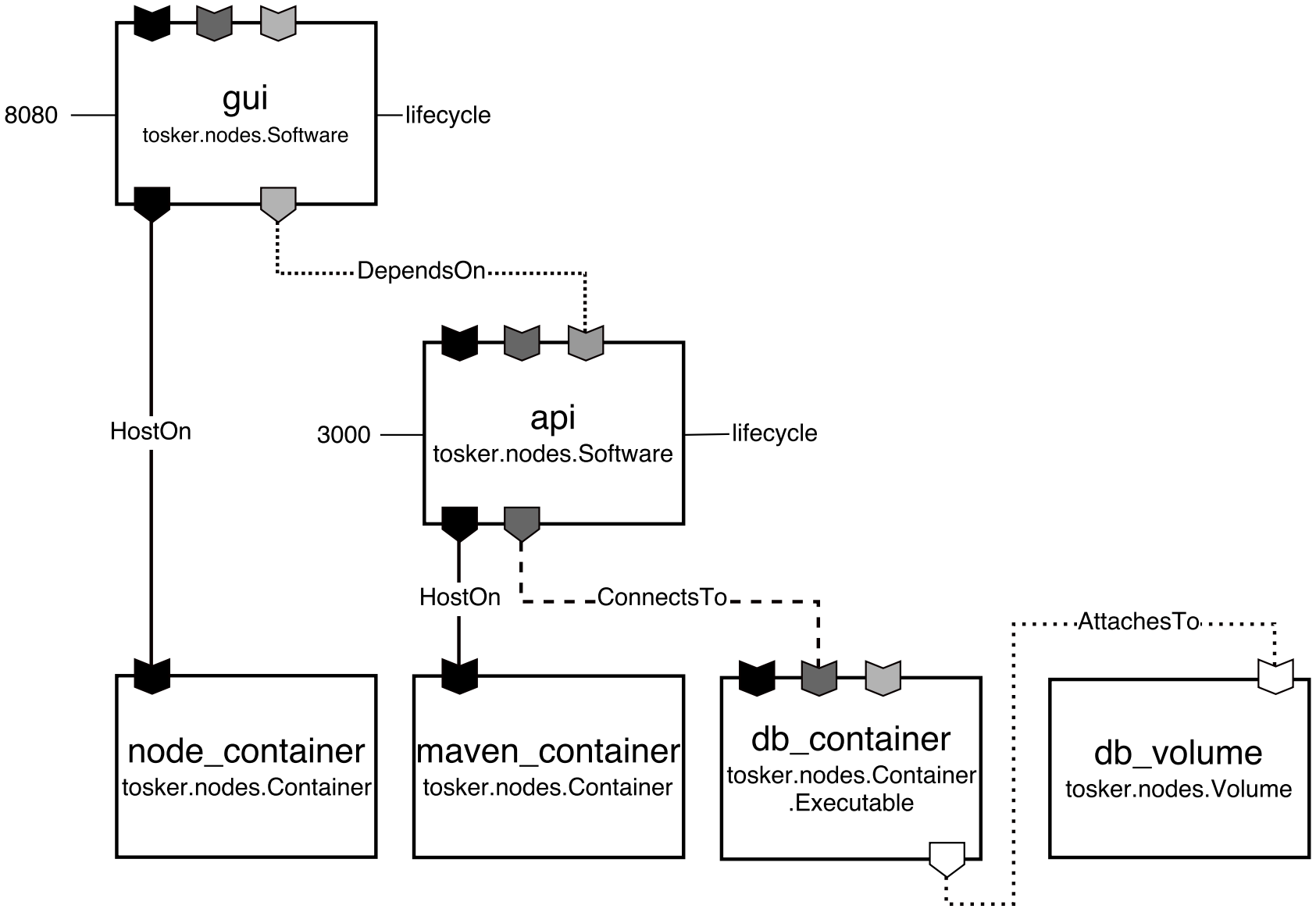
Describing applications in TosKer

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

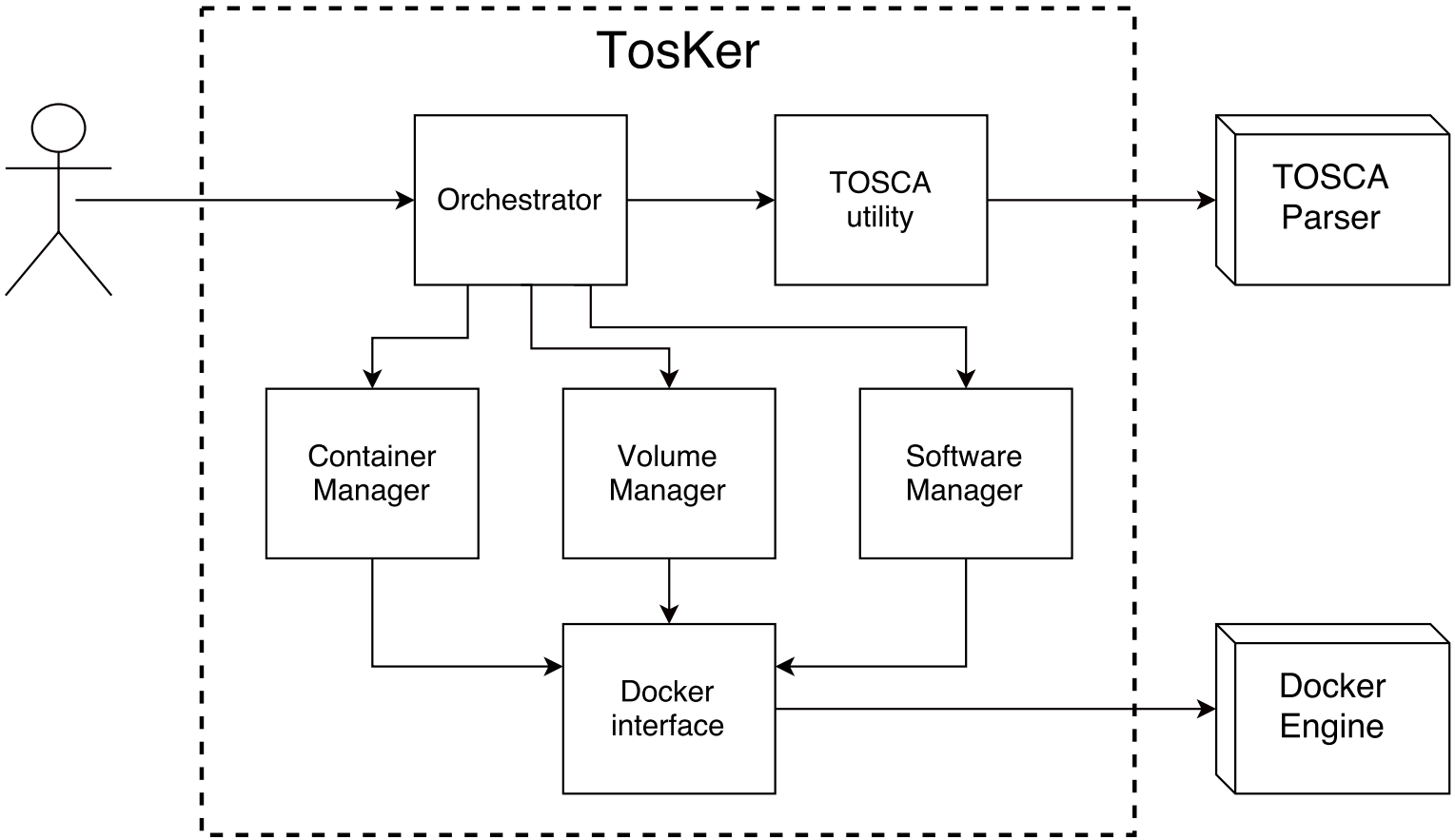
Describing applications in TosKer

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

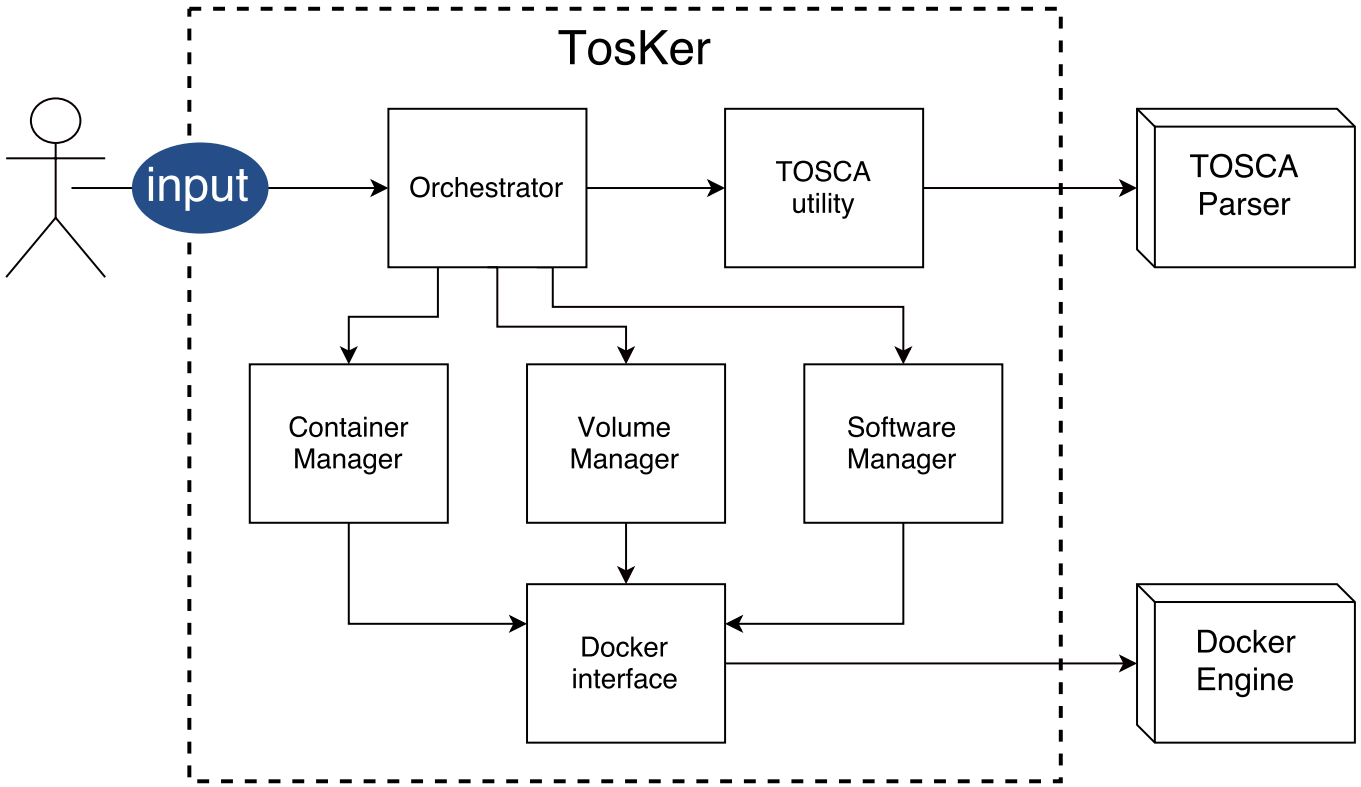
Use case


gui
tosker.nodes.Software
8080
lifecycle
DependsOn
HostOn
api
tosker.nodes.Software
3000
lifecycle
HostOn
ConnectsTo
AttachesTo
node_container
tosker.nodes.Container
maven_container
tosker.nodes.Container
db_container
tosker.nodes.Container
.Executable
db_volume
tosker.nodes.Volume

Architecture of TosKer



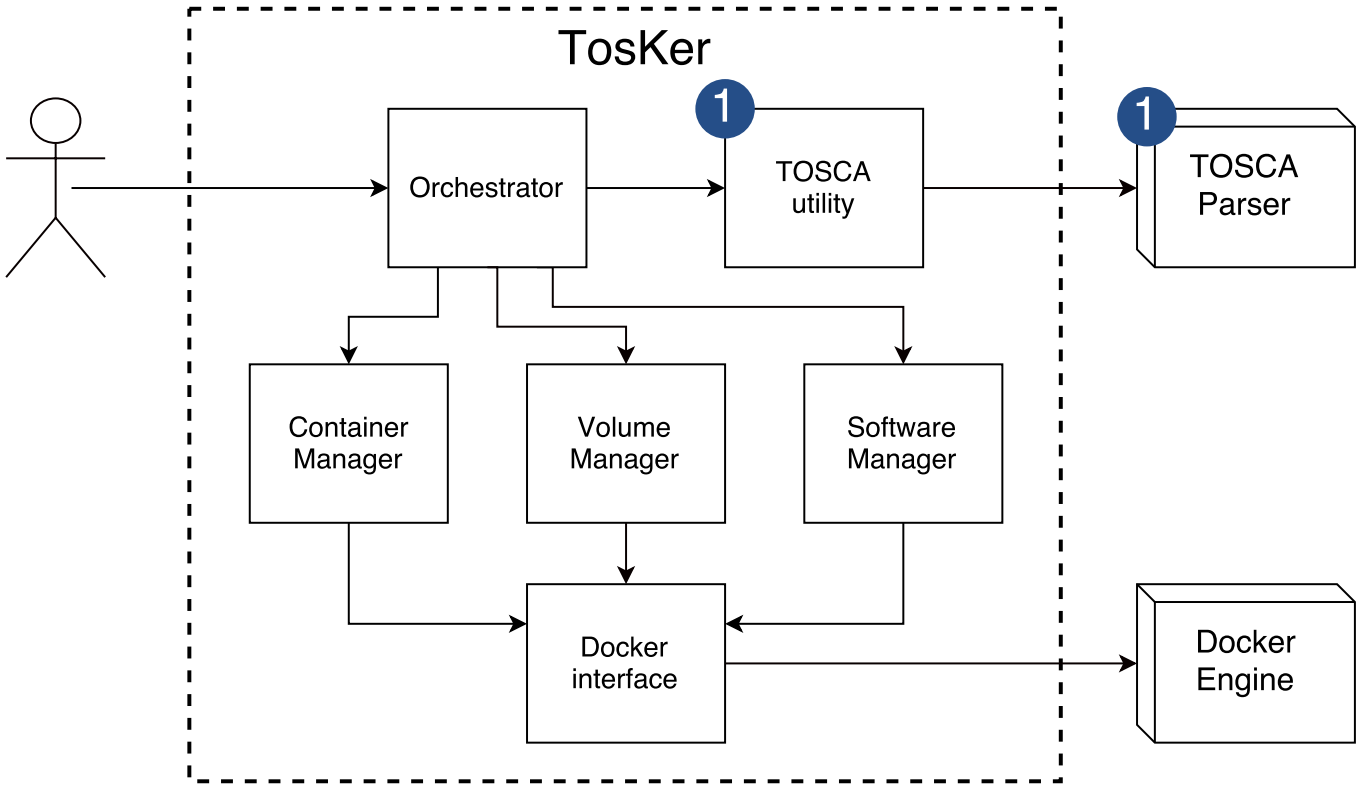
How TosKer orchestrates applications



The input of TosKer is

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

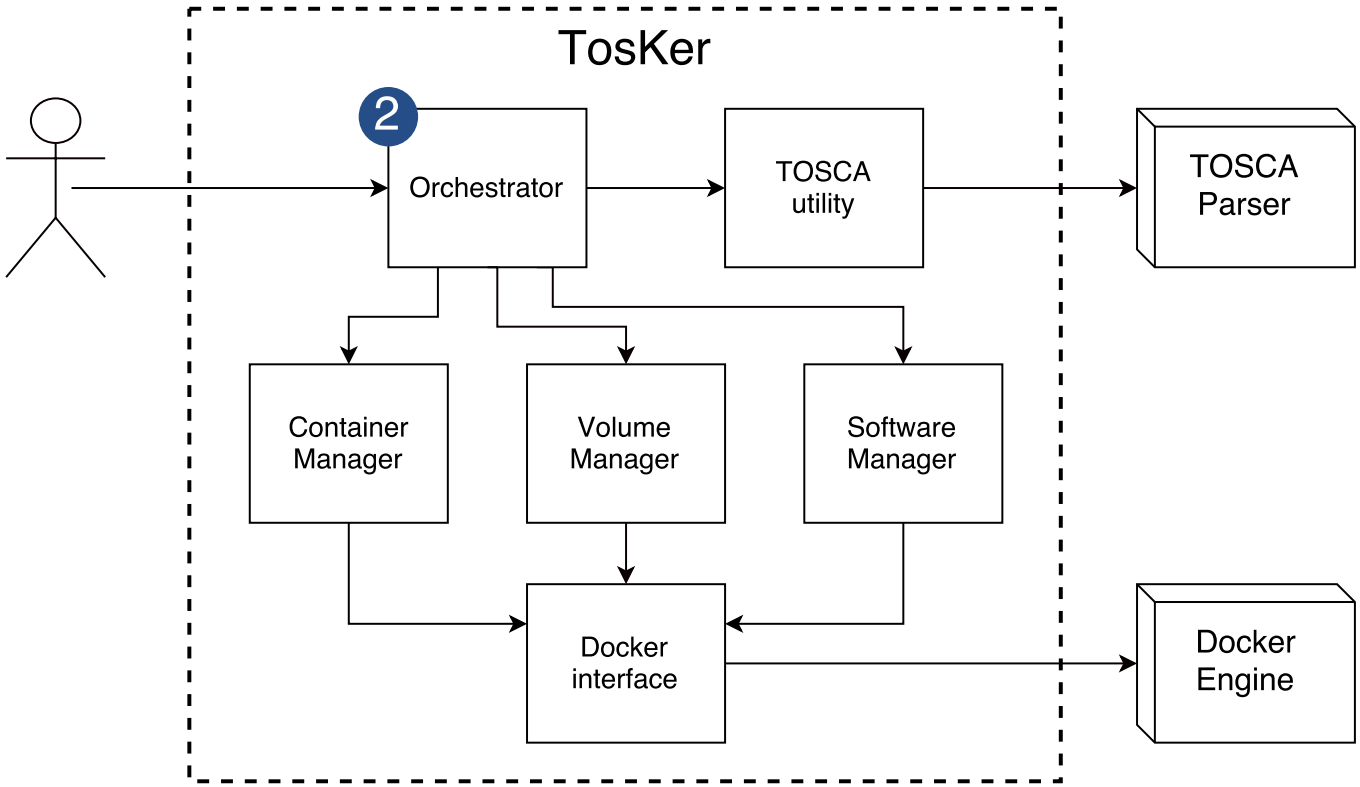
How TosKer orchestrates applications



TosKer

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

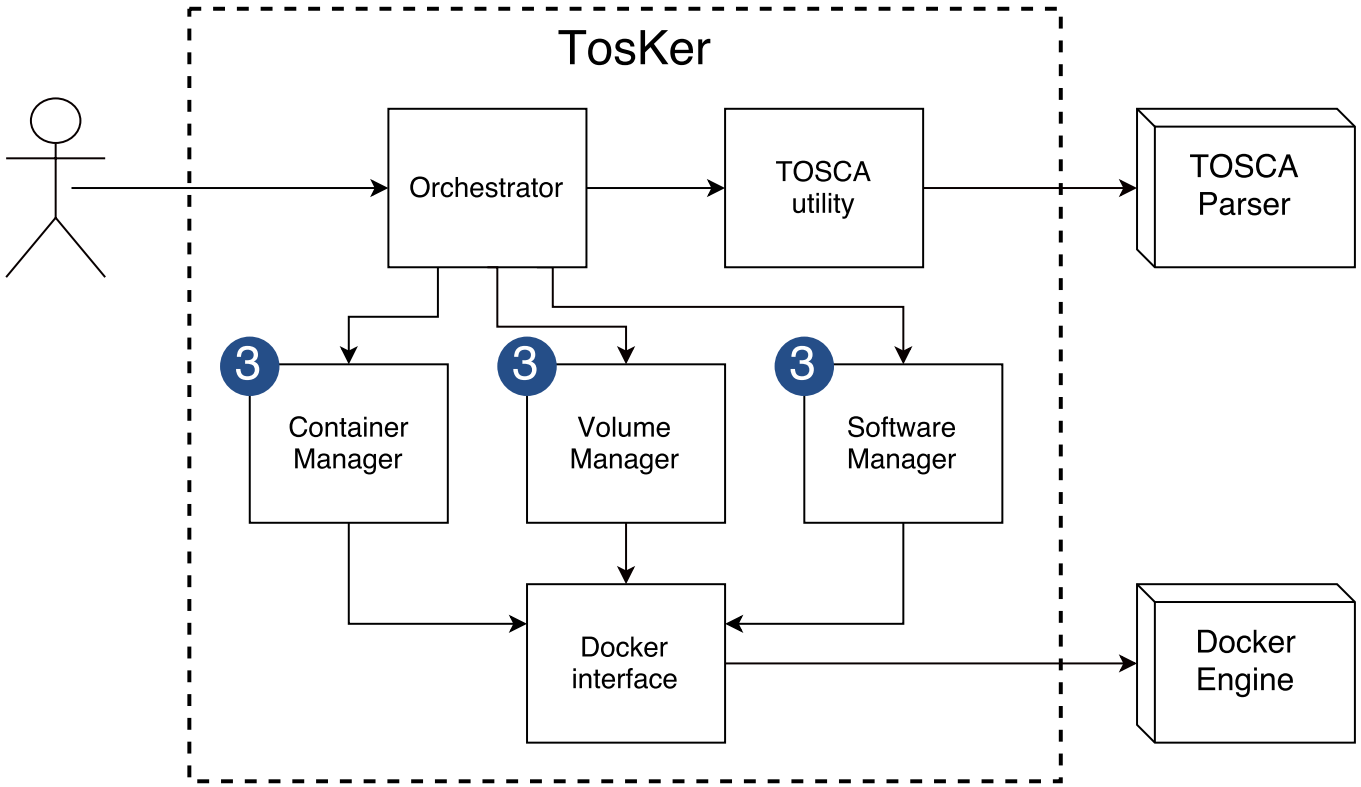
How TosKer orchestrates applications



TosKer

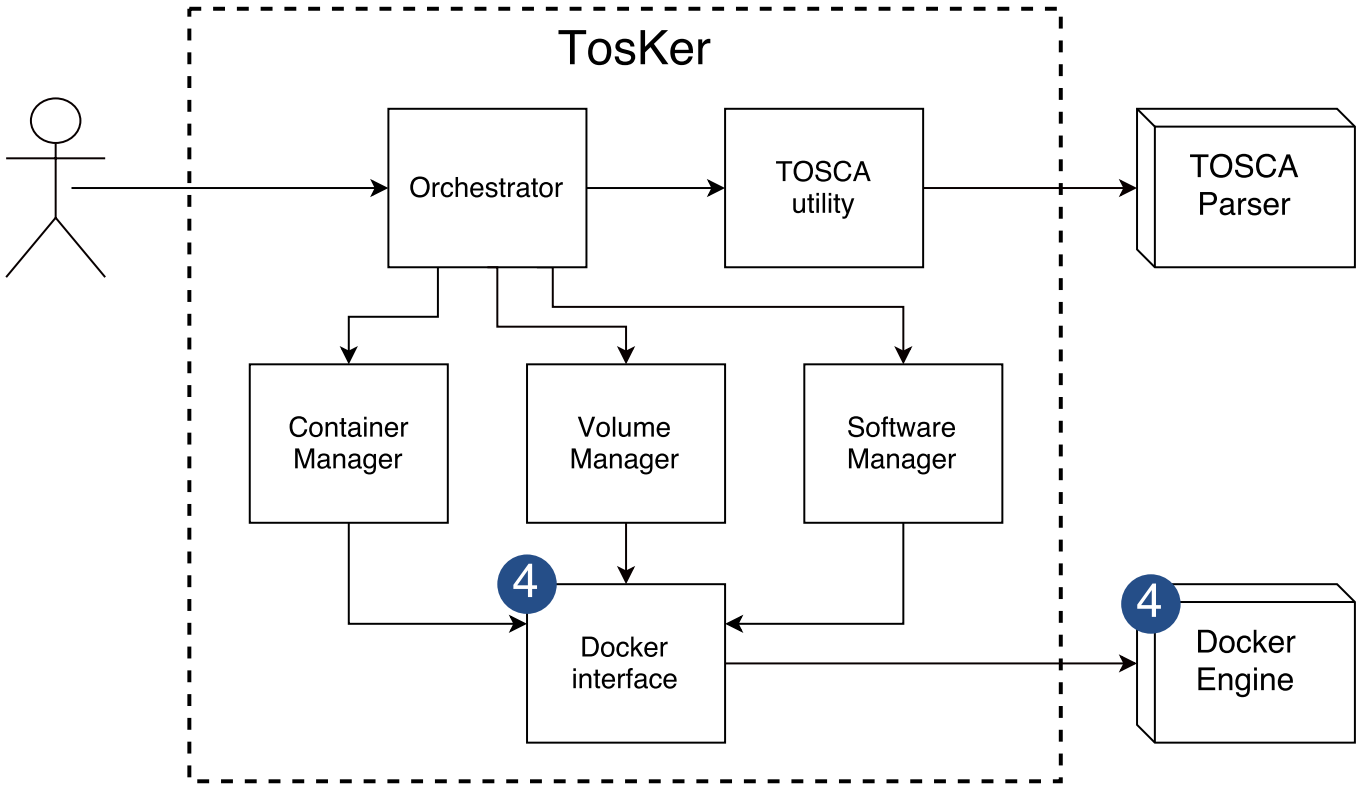
- scans the sorted application topology, and
- for each component, it calls a specific operation (e.g., create)

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)

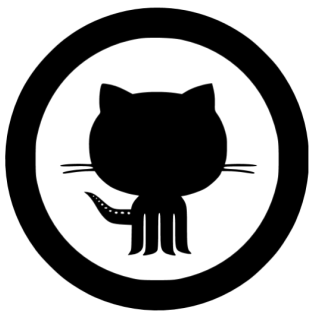
Implementation of TosKer

Python

- **PyPI**: https://pypi.python.org/pypi/tosKer
  `pip install tosker`

Implementation of TosKer

Python



GitHub

MIT Licence

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

Usage of TosKer command-line

```
tosker FILE COMMANDS... [OPTIONS] [INPUTS]
tosker -h|--help
tosker -v|--version

FILE: TOSCA YAML file or CSAR file

COMMANDS:
  create    Create application components
  start     Start applications components
  stop      Stop application components
  delete    Delete application components (except volumes)

OPTIONS:
  -h --help       Print usage
  -v --version    Print version
  -q --quiet      Enable quiet mode
  --debug         Enable debugging mode (override quiet mode)

INPUTS: provide TOSCA inputs (syntax: --NAME VALUE)
```

VIDEO-DEMO

Table of Contents

Conclusions and future work

We have presented TosKer, an orchestration engine, which

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

Conclusions and future work

We have presented TosKer, an orchestration engine, which

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

Future work

- Support cluster of workstations and external cloud services

Conclusions and future work

We have presented TosKer, an orchestration engine, which

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

Future work

- Support cluster of workstations and external cloud services
- Automatically determine the Docker containers needed to effectively run an application

Conclusions and future work

We have presented TosKer, an orchestration engine, which

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

Future work

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

Thank You

Q&A