TOSKER

Luca Rinaldi

SOFTWARE DEPLOYMENT

The execution of all the activities that make a software system available to use.

Nowadays strictly related to the cloud infrastructure.

Need of a way to express all the **requirements** that the application needs to run.

How to describe an application?

DOCKER

Docker containers wrap up the software and all the requirements: code, runtime, system tools, system libraries.

This guarantees that the software always run in all environment that support Docker.

Embed all the requirements inside a container!

TOSCA

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

Following the description it is possible to replicate the configuration of the application.

Describe every part of your application!

DOCKER VS. TOSCA

Two different approach to resolve the same problem:

installation vs **description**

PROS AND CONS OF DOCKER

- ⇧ It works out of the box
- ⇧ There are a lot of images ready to be used
- ⇩ Cannot deploy complex applications
- ⇩ Everything must be a container

PRO AND CONS OF TOSCA

- ⇧ Well documented standard
- ⇧ Adaptable to every deployment infrastructure
- ⇩ Lack of implementations
- ⇩ Too verbose, everything must be described

TOSCA + DOCKER

Why not combining them instead of choosing?

TOSKER

Project that aims to combine **TOSCA** and **Docker** to simplify the deployment of applications on the Cloud.

FEATURES OF TOSKER

- Can deploy Docker container and generic software components
- Can deploy complex applications
- Uses the requirements/capability system of TOSCA
- Can manage networking between components

HOW IT WORKS

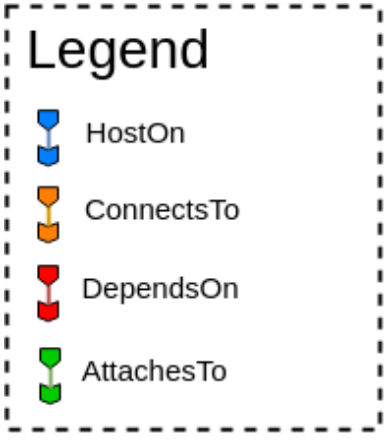
1. The application is described using the TOSCA yaml language.
2. The TOSCA file is validated and parsed
3. The deployment order is computed
4. The deployment is executed using Docker

CUSTOM TYPES

TosKer support only those custom types:

- Docker persistent container *tosker.docker.container.persistent*
- Docker container *tosker.docker.container*
- Docker volume *tosker.docker.volume*
- Software *tosker.software*

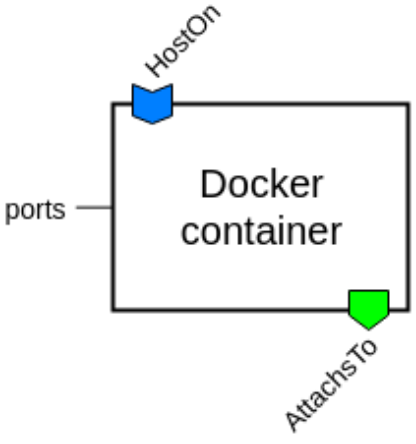
TYPES OF RELATIONSHIP



Tosker support all the normative relationship:

- host *tosca.relationships.HostedOn*
- connect *tosca.relationships.ConnectsTo*
- depend *tosca.relationships.DependsOn*
- attach *tosca.relationships.AttachesTo*

DOCKER CONTAINER



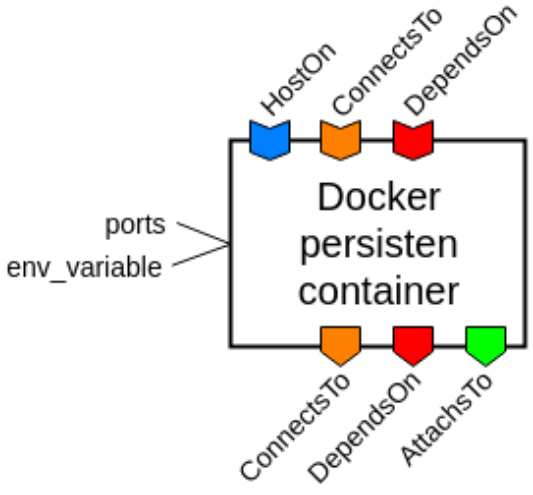
DEFINITION

```
tosker.docker.container:
  derived_from: tosca.nodes.Container.Runtime
  properties:
    ports:
      type: map
      required: false
  requirements:
    - attach:
        capability: tosca.capabilities.Attachment
        occurrences: [0, UNBOUNDED]
  capabilities:
    host:
      type: tosca.capabilities.Container
      valid_source_types: [tosker.software]
      occurrences: [0, UNBOUNDED]
```

EXAMPLE

```
my_container:
  type: tosker.docker.container
  requirements:
    - attach: my_volume
  properties:
    ports:
      80: 8000
  artifacts:
    my_image:
      file: ubuntu:16.04
      type: tosker.docker.image
      repository: docker_hub
```

DOCKER PERSISTENT CONTAINER



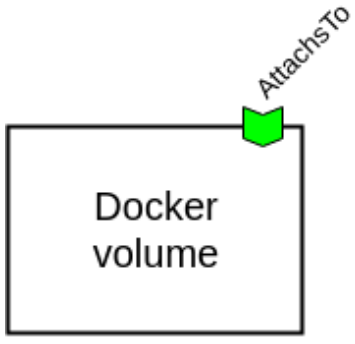
DEFINITION

```
tosker.docker.container.persistent:
  derived_from: tosker.docker.container
  properties:
    env_variable:
      type: map
      required: false
    ...
  requirements:
    - connect:
        capability: tosca.capabilities.Endpoint
        occurrences: [0, UNBOUNDED]
    - depend:
        capability: tosca.capabilities.Node
        occurrences: [0, UNBOUNDED]
  capabilities:
    connect:
```

EXAMPLE

```
my_container:
  type: tosker.docker.container.persistent
  requirements:
    - connect: my_other_container
    - depend: my_software
    - attach: my_volume
  properties:
    ports:
      80: 8000
  artifacts:
    my_image:
      file: mysql
      type: tosker.docker.image
      repository: docker_hub
```

DOCKER VOLUME



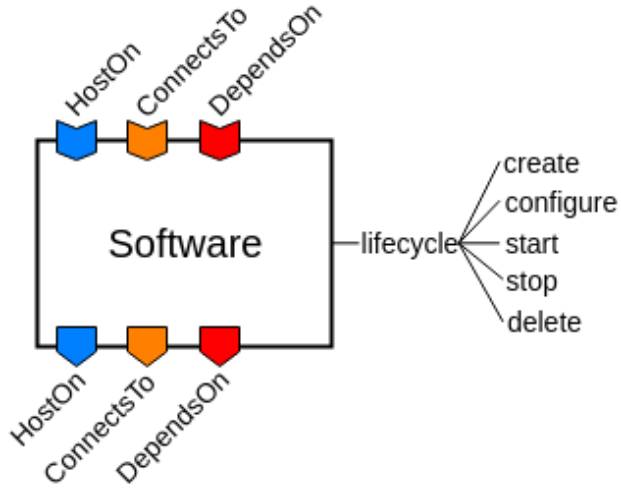
DEFINITION

```
tosker.docker.volume:
  derived_from: tosca.nodes.BlockStorage
  properties:
    driver: # by default it is local
      type: string
      required: false
    size: # restrict to a given size. for example: 100m
      type: string
      required: false
    ...
  capabilities:
    attach:
      type: tosca.capabilities.Attachment
      valid_source_types: [tosker.docker.container.persistent, toske
      occurrences: [0, UNBOUNDED]
```

EXAMPLE

```
my_volume:
  type: tosker.docker.volume
  properties:
    driver: local
    size: 200m
```

SOFTWARE TYPE



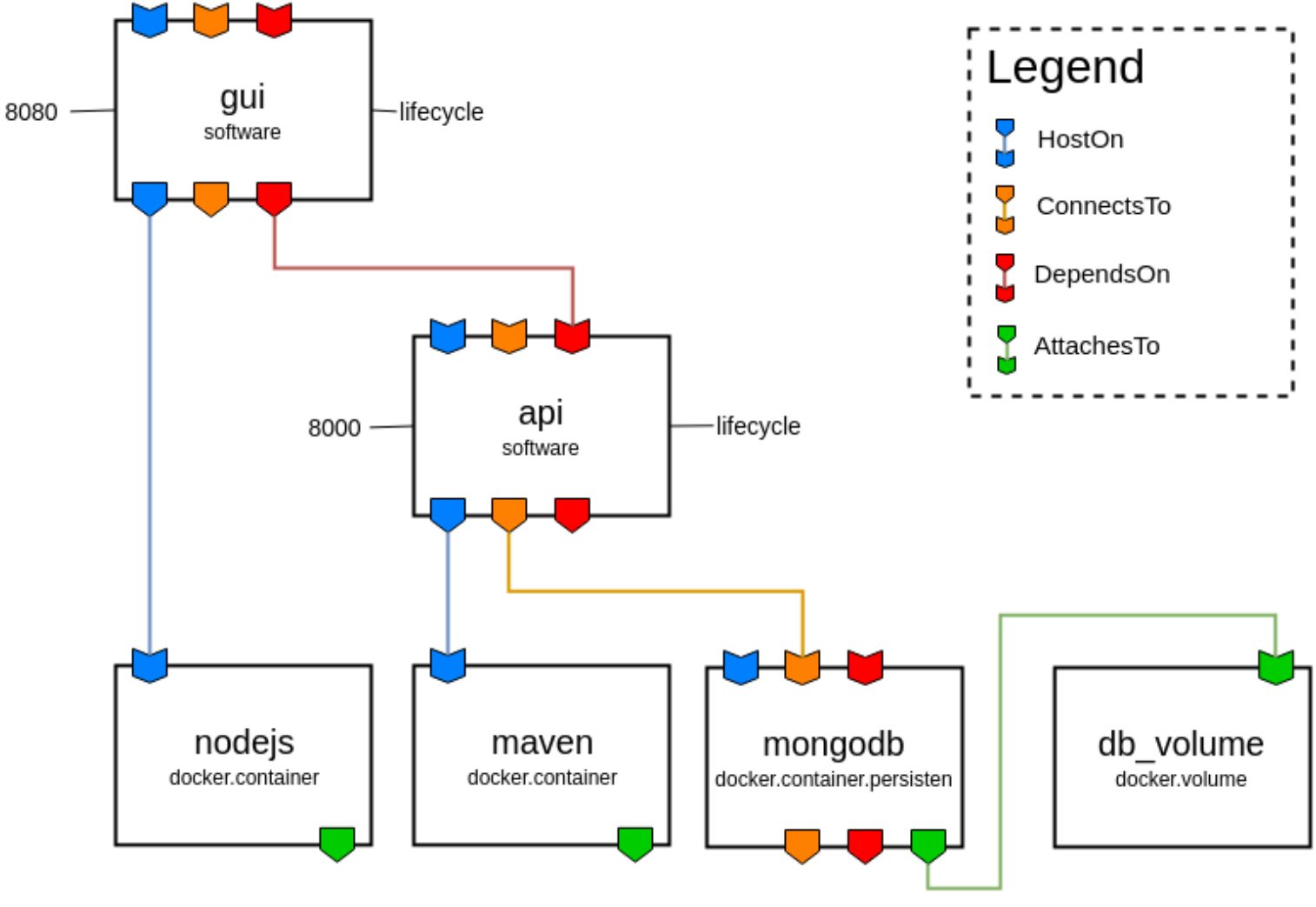
DEFINITION

```
tosker.software:
  derived_from: tosca.nodes.SoftwareComponent
  requirements:
    - connect:
        capability: tosca.capabilities.Endpoint
        occurrences: [0, UNBOUNDED]
    - depend:
        capability: tosca.capabilities.Node
        occurrences: [0, UNBOUNDED]
    - host:
        capability: tosca.capabilities.Container
        occurrences: 1
  capabilities:
    host:
      type: tosca.capabilities.Container
      valid_source_types: [tosker.software]
```

EXAMPLE

```
my_software:
  type: tosker.software
  requirements:
    - connect: my_other_software
    - depend: my_container
    - host: my_container
  interfaces:
    Standard:
      create:
        implementation: create.sh
      configure:
        implementation: configure.sh
      start:
        implementation: start.sh
      stop:
        implementation: stop.sh
```

AN EXAMPLE: THOUGHTS



TOSCA SPECIFICATION

```
tosca_definitions_version: tosca_simple_yaml_1_0

description: TOSCA description of the Thoughts application.

repositories:
  docker_hub: https://registry.hub.docker.com/

imports:
  - tosker: https://di-unipi-socc.github.io/tosker-types/0.0.5/toske

topology_template:
  node_templates:
    api:
      type: tosker.software
      requirements:
        - host: maven_container
```

HOW TO USE TOSKER

```
tosker <file> (create|start|stop|delete)... [<inputs>...]
```

Options:

```
-h --help     Show this help.
-q --quiet    Active quiet mode.
--debug       Active debugging mode.
```

Examples:

```
tosker app.yaml create start
tosker app.yaml stop delete
```

DEMO

ACHIEVEMENTS OF TOSKER

- Combine the deployment of Software components together with Docker Container
- Implementation an engine that accept the TOSCA language

KNOWS LIMITATIONS

- Cannot support TOSCA hierarchy type system
- Only work with Docker engine (no Swarm)