M.SC. THESIS

AUTOMATING THE DEPLOYMENT OF CLOUD-NATIVE APPLICATIONS OVER MULTIPLE PLATFORMS

Presented by Leonardo Frioli

Supervisors

Antonio Brogi - University of Pisa

Jacopo Soldani - University of Pisa

Michael Wurster - University of Stuttgart

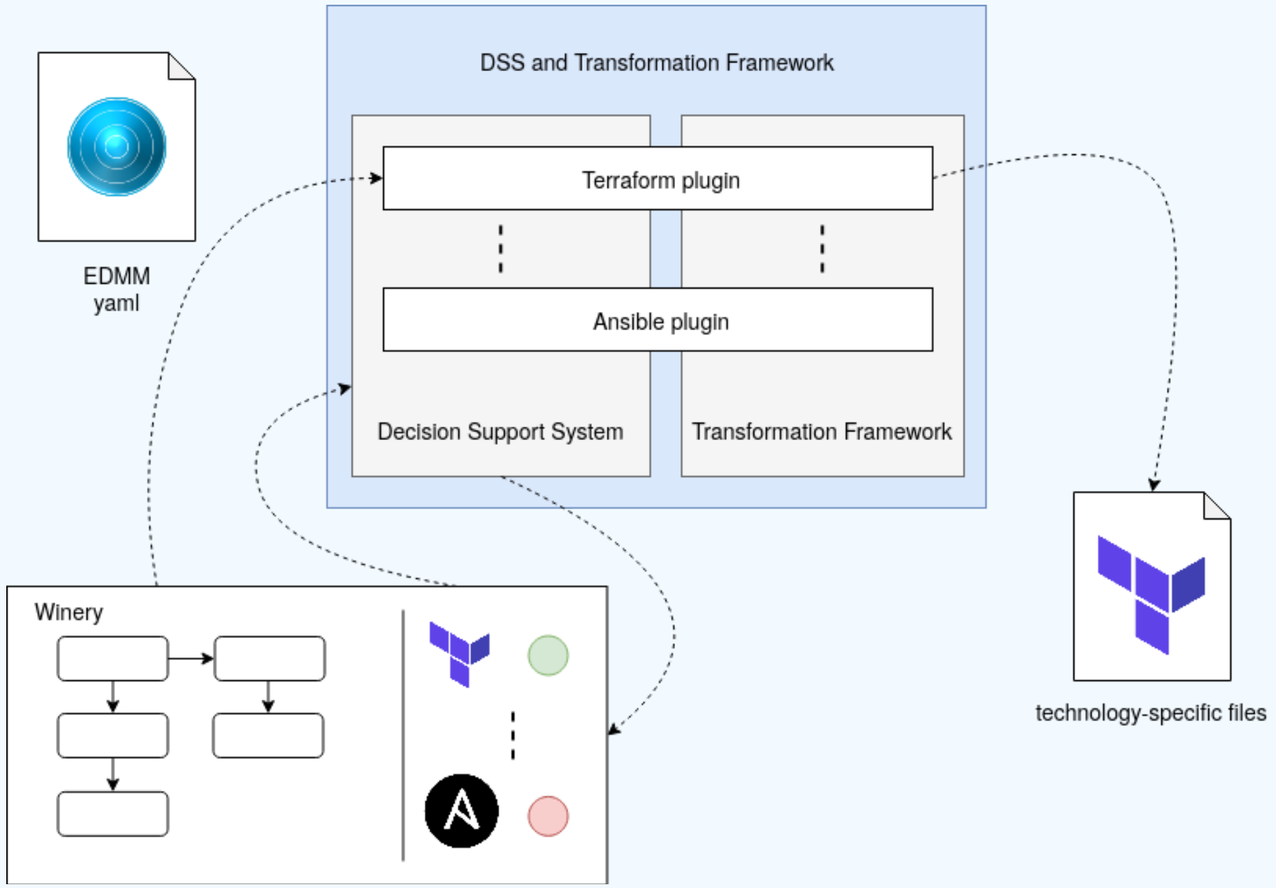
CLOUD DEPLOYMENT




- multi-components applications
- declarative deployment automation technologies
  - similar functionalities offered in different ways
- domain specific language
  - lack of portability
- lock-in
  - difficulty of changing the deployment tool after the first deployment

EDMM

- technology-agnostic deployment model
- YAML specification
- Modelling tool
- Decision Support System (DSS)
- Transformation Framework
  - mapping to technology-specific resources



M. Wurster et al. The EDMM Modelling and Transformation System. 17th International Conference on Service-Oriented Computing (ICSOC 2019).

CONTRIBUTION

issues

- non integrated environment
- better decision support system logic required

contribution

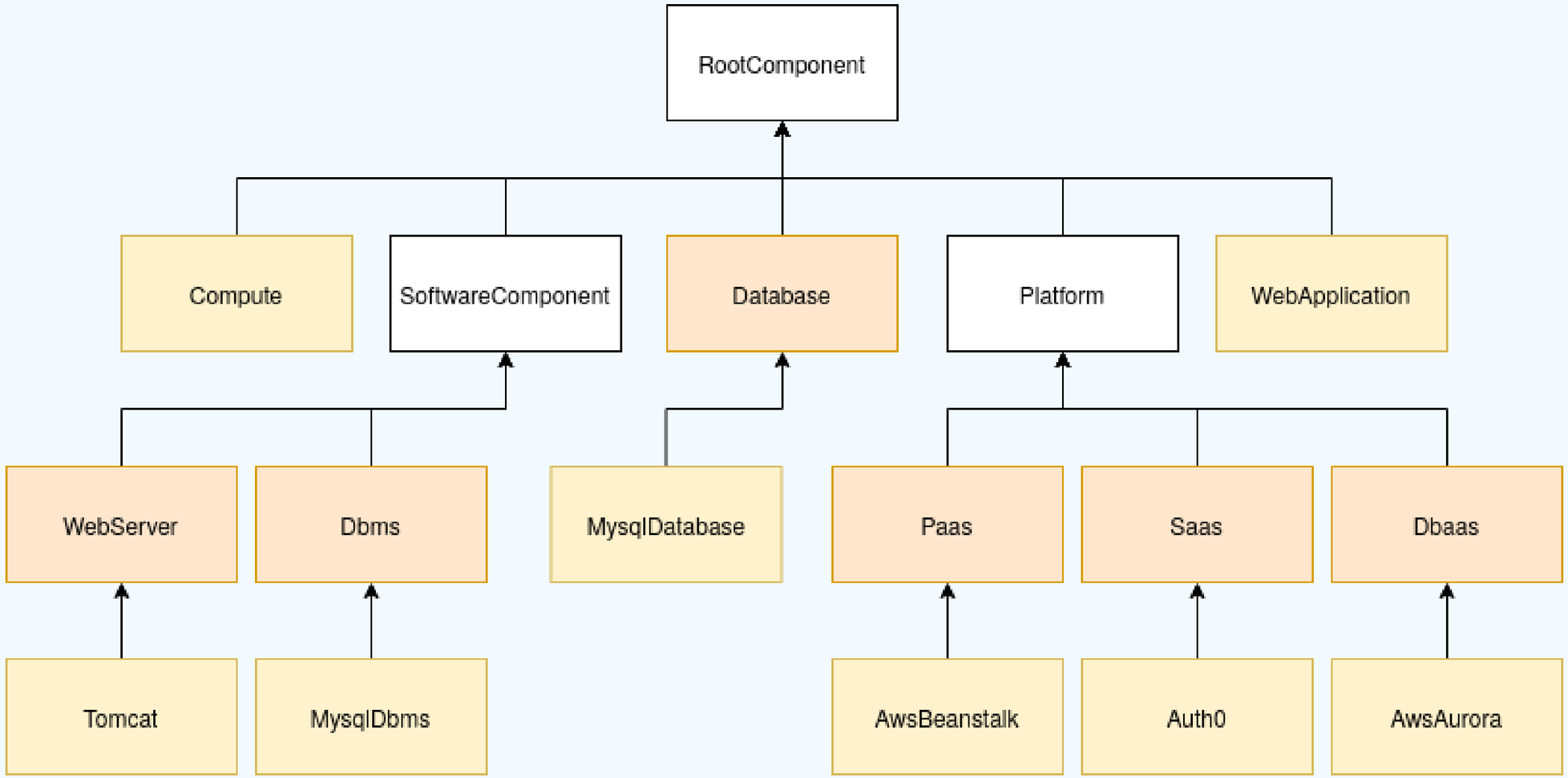
- a standalone solution integrating the entire EDMM toolset
- an extension of the DSS
- an assessment of the usefulness of the integrated EDMM environment (included the enhanced DSS)

EDMM

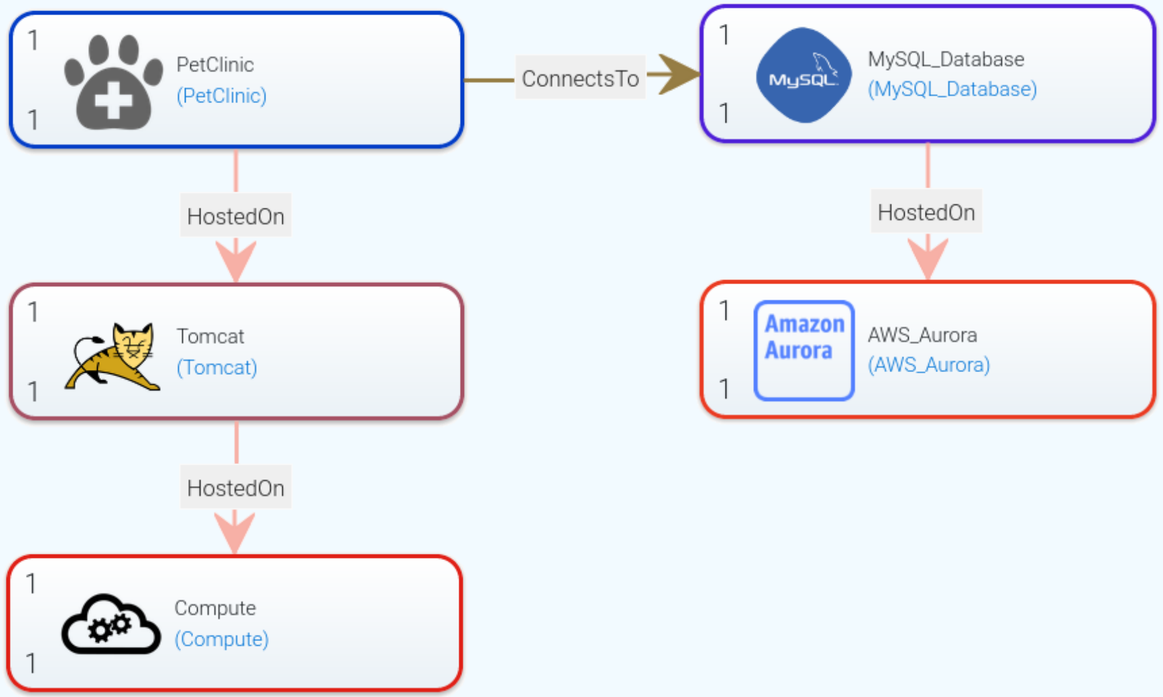
Essential Deployment Metamodel

- deployment model defined as a topology graph
  - the nodes are the application components
  - the edges are the dependencies among such components
- Component Types (see next slide)
- Relation Types
  - HostedOn, ConnectsTo, DependsOn
- Artifacts
  - packaged code, binaries
  - install, configure, start, stop operations scripts
- mapping between the EDMM entities and the technology-specific abstractions

COMPONENT TYPES

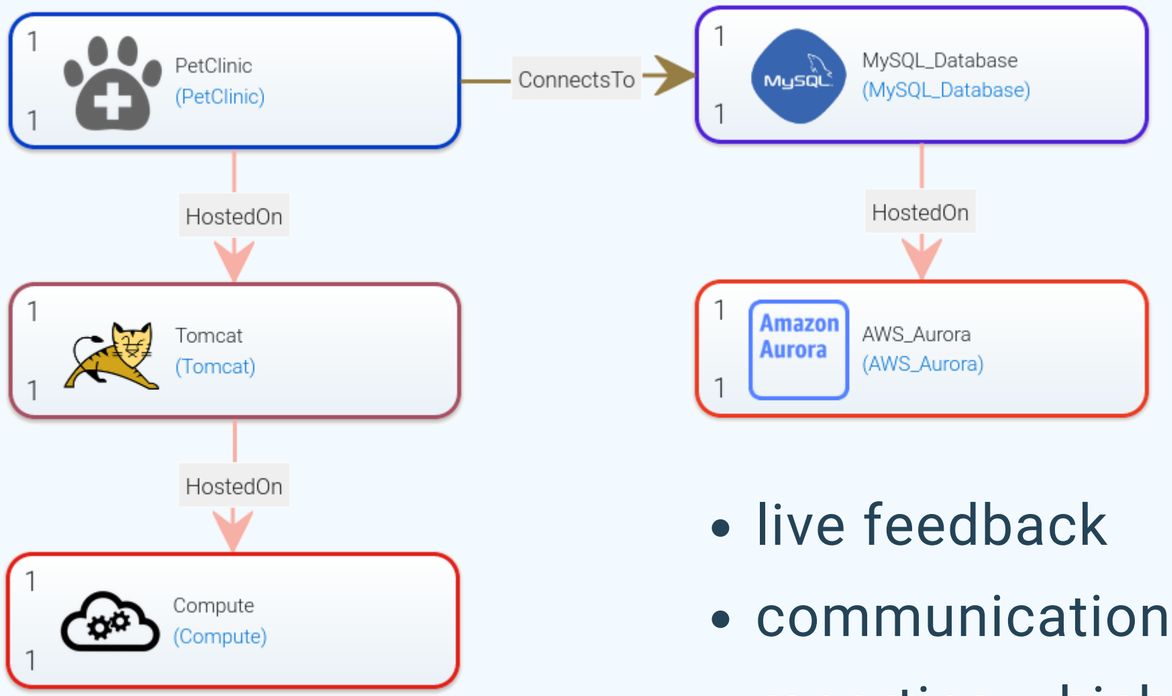


MODELLING TOOL



- web-based application
- an extenstion of Eclipse Winery
- mapping between EDMM types and Winery internal data model
- **modelling-repository** containing the definitions of the templates and the types, as well as the artifacts files

DECISION SUPPORT SYSTEM



- live feedback
- communication done via REST interface
- reporting which technology supports the current topology
- highlighting the unsupported nodes
- no further suggestions

TRANSFORMATION FRAMEWORK

```
---
components:
   Tomcat:
      type: tomcat
      relations:
      - hosted_on: Compute
   PetClinic:
      type: web_application
      relations:
      - hosted_on: Tomcat
      - connects_to: MySQL_Database
   Compute:
      type: compute
```

```
edmm transform terraform \
petclinic-xaas.yaml
```

- each deployment automation technology is implemented as a plugin
- functionalities exposed via command-line interface
- transforms the EDMM deployment model into the technology-specific files required to enact the deployment
- an application developer is forced to completely change environment when she wishes to perform the transformation

INTEGRATION

Why not enabling to run an integrated environment, not needing to get configured, and allowing to directly obtain the technology-specific files from its graphical UI?

library-based

- importing the package (edmm-core) implementing the DSS and the Transformation Framework core logic into Winery

REST-based

- the DSS and the Transformation Framework run on a REST service exposing their functionalities via an API

LIBRARY BASED

pros

- Winery, the DSS and the Transformation Framework are Java applications
- the module implementing the DSS and the Transformation Framework core logic is already imported
- cleaner conversion from the Winery data representation to the EDMM deployment model
- easier integration testing

cons

- we should respect the Winery coding guidelines
- library alignment
- deep knowledge of the Winery codebase required

REST BASED

pros

- almost implemented
- easier development
- no need to modify the Winery backend code

cons

- microservices-like architecutre not strictly required
- unnecessary wasting of network resources
- the module implementing the DSS and the Transformation Framework core logic should be imported anyway

CHOSEN APPROACH

LIBRARY BASED

- better to adopt microservices-like architecture only if needed
- same business capability
  - editing and management of the EDMM deployment model
- the module implementing the DSS and the Transformation Framework core logic is already imported and cannot be removed
- easier tesing, better maintainability, cleaner conversion between data representations
- **still** more code to write and knowledge of Winery implementation required
- **still** need to pay attention to library version alignment

IMPLEMENTATION

- plugin instantiation done by plain Java
  - cannot use Java Spring and Spring Boot
- reusing the same classes instantiated in the DSS REST service and in the Tranformation Framework command-line interface
- directly parsing the Winery topology representation into the EDMM deployment model

IMPLEMENTATION

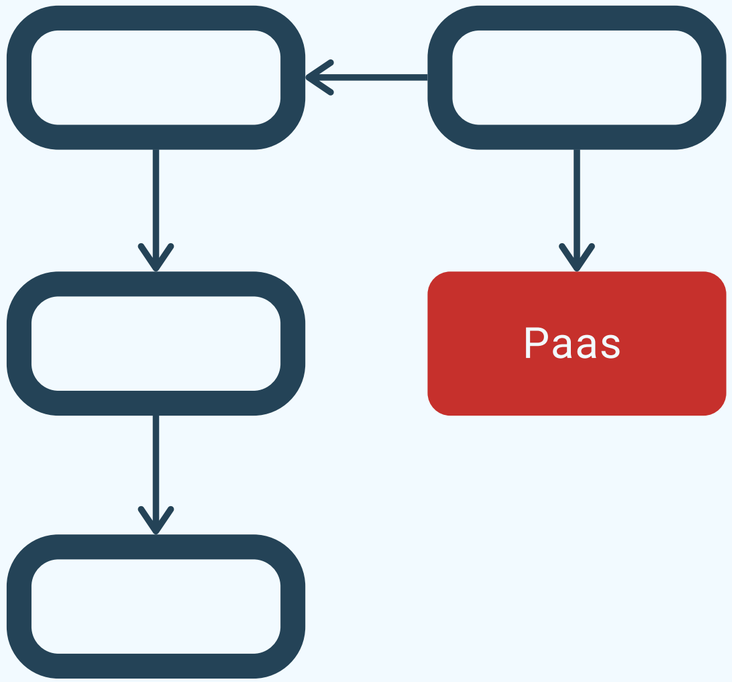
- added two REST enpoints in the *winery.repository.rest* package
- frontend slighlty changed
  - fixed the REST calls
  - added a 'transform' button
- needed to align a library

ENHANCING THE DSS

Why not supporting application developers also in reasoning on how to adapt their deployment specification to get them deployed also by unsupported automation technologies?

rule system

- sub-topology into sub-topology replacements
- based on Component and Relation Types

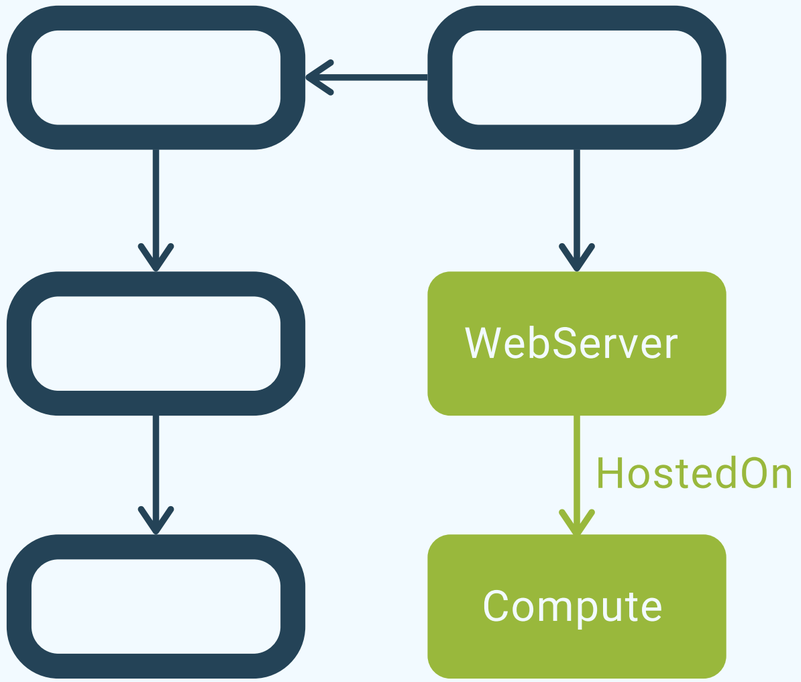


ENHANCING THE DSS

Why not supporting application developers also in reasoning on how to adapt their deployment specification to get them deployed also by unsupported automation technologies?

rule system

- sub-topology into sub-topology replacements
- based on Component and Relation Types



RULE SYSTEM

rule

- defines a not supported sub-topology (*fromTopology*)
- provides a supported replacement (*toTopology*)
- each plugin can have its list of rules

default rules

- *Paas* can be replaced by a *WebServer HostedOn Compute*
- *Saas* can be replaced by a *WebApplication HostedOn WebServer*, *HostedOn Compute*
- *Dbaas* can be replaced by a *Dbms HostedOn Compute*

RULE SYSTEM

sub-topology definition

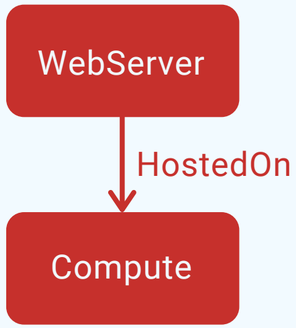
- the fromTopology and toTopology are defined leveraging the EDMM YAML specification

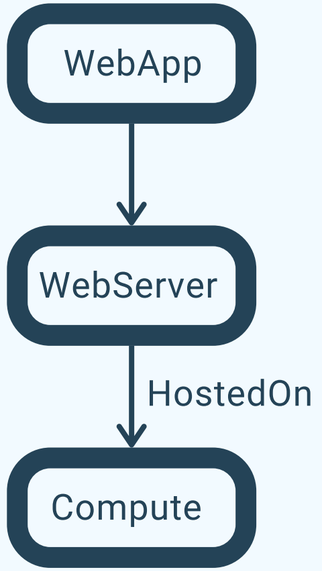
```
EdmmYamlBuilder yamlBuilder = new EdmmYamlBuilder();

yamlBuilder.component(WebServer.class)
           .hostedOn(Compute.class)
           .component(Compute.class);

String yamlString = yamlBuilder.build();
```

RULE SYSTEM

rule system workflow

- the system gets the current deployment model
- each rule is evaluated checking, in the graph, if there is a sub-topology that matches the fromTopology
- if the evaluation returns true, the rule gets executed returning the *toTopology*

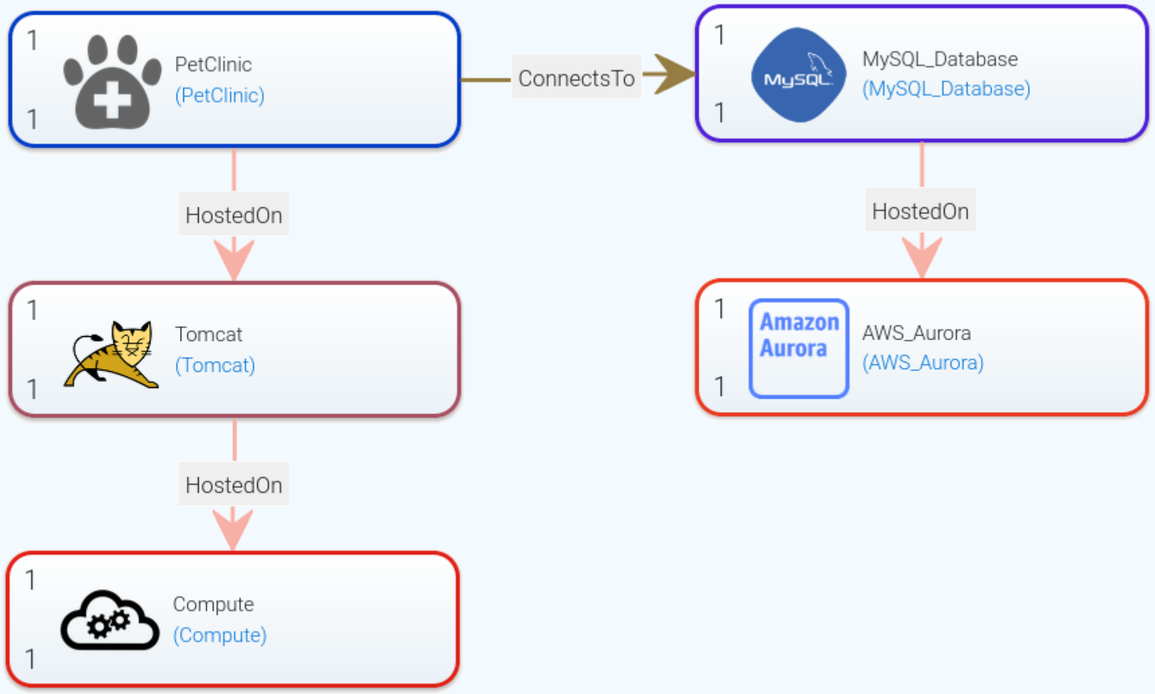




RULE SYSTEM

application developer point of view

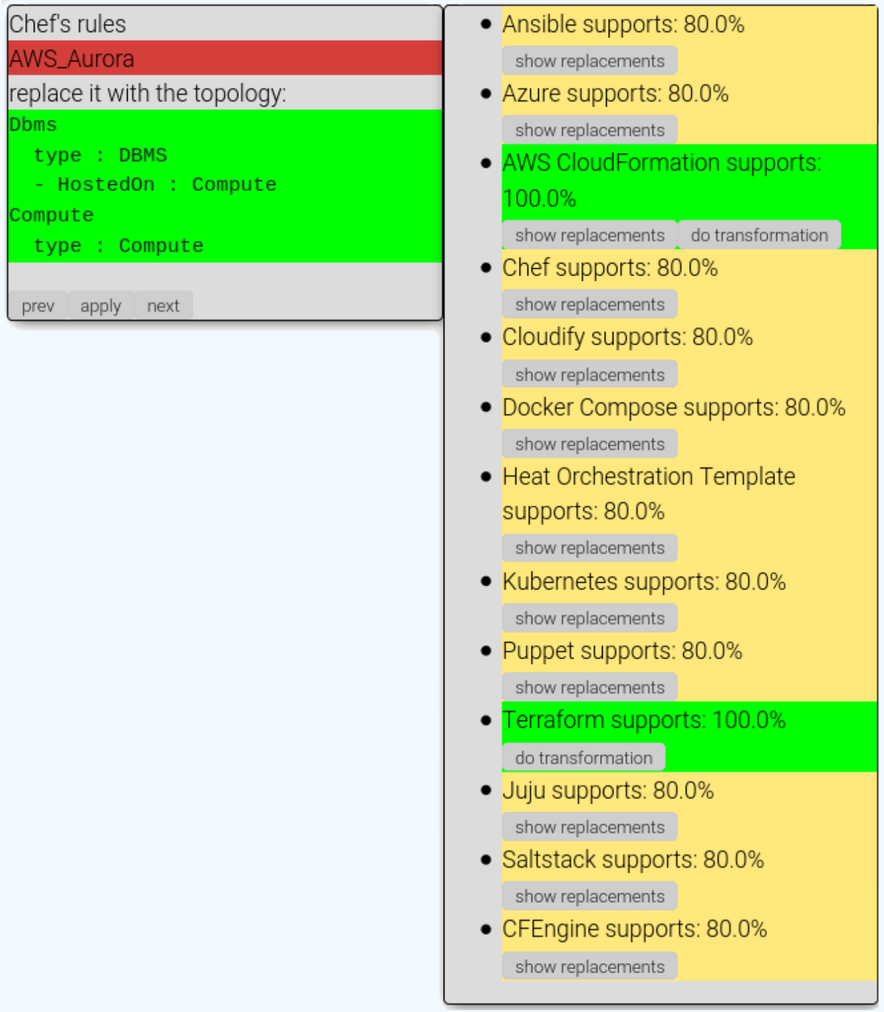
1. models the deployment topology
2. Winery sends the current model to the rule system



RULE SYSTEM

application developer point of view

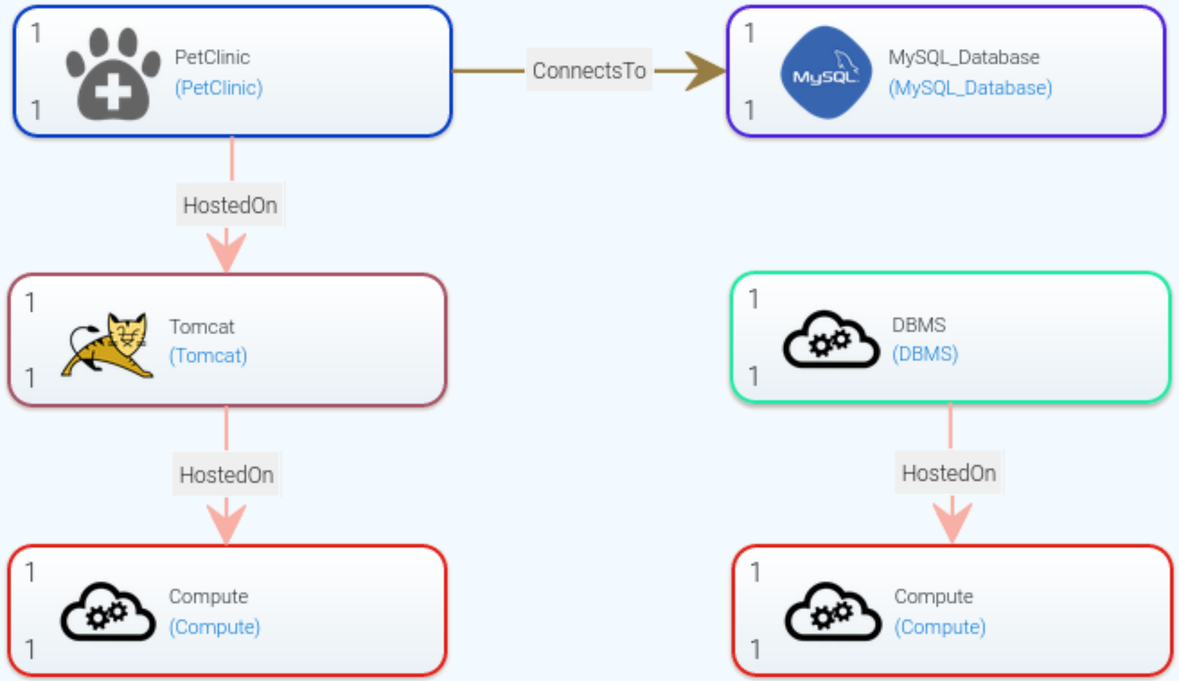
1. models the deployment topology
2. Winery sends the current model to the rule system
3. the systems returns the rule results, containing the suggested replacements
4. the results are displayed



RULE SYSTEM

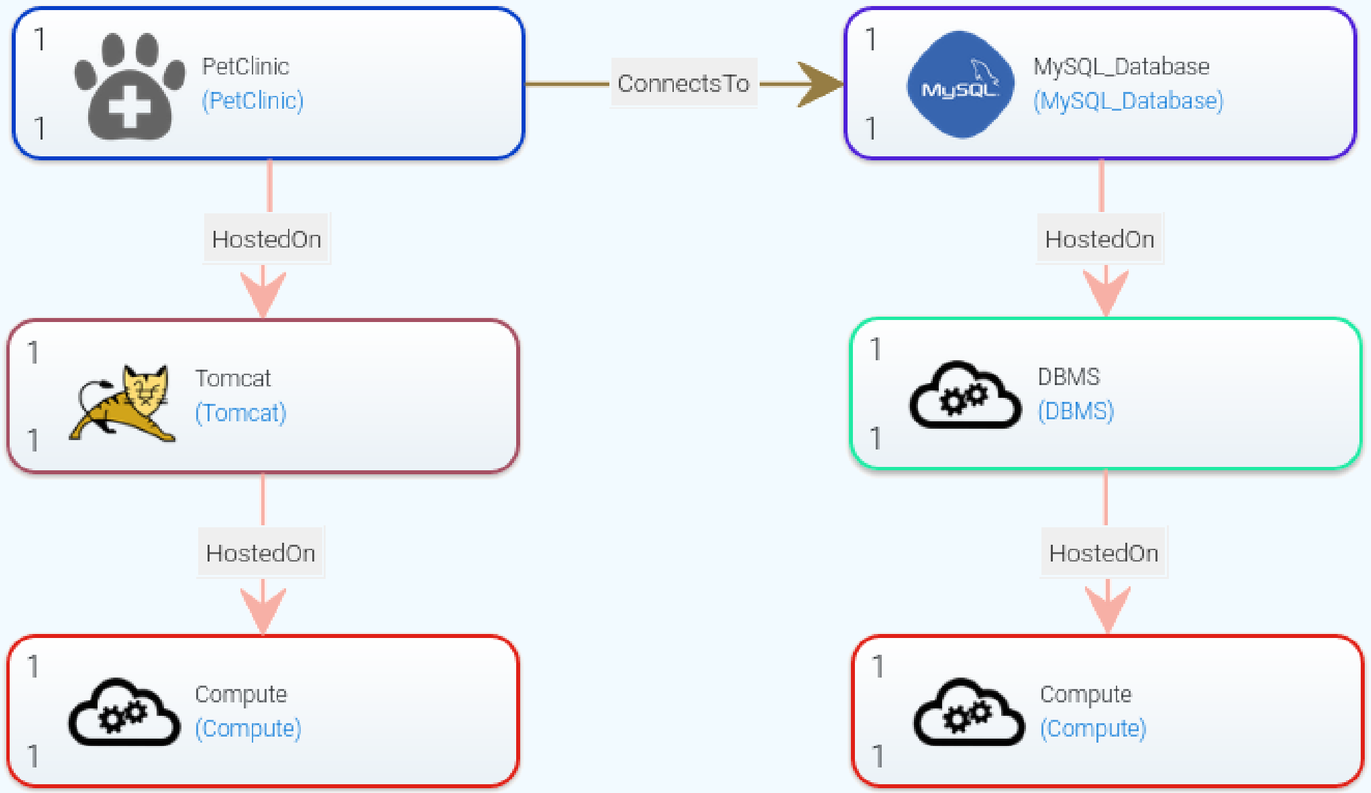
application developer point of view

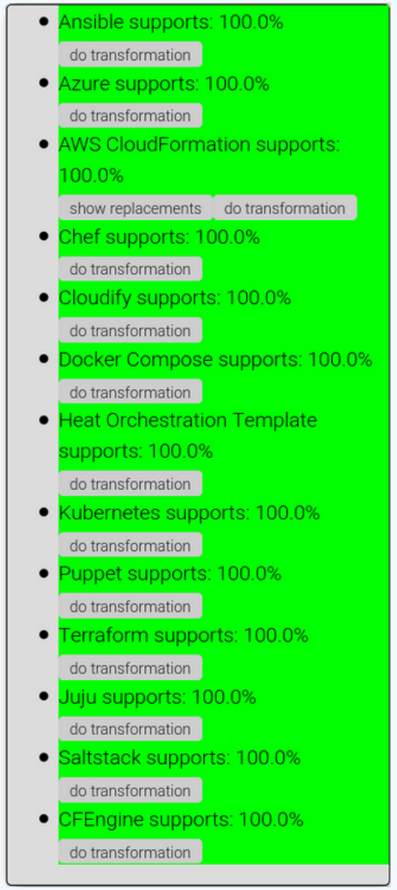
1. models the deployment topology
2. Winery sends the current model to the rule system
3. the systems returns the rule results, containing the suggested replacements
4. the results are displayed
5. the user can automatically apply the rule



RULE SYSTEM

The resulting deployment model is now supported by the chosen deployment automation technology





CASE STUDY

Initial deployment with Kubernetes

Migrating from Kubernetes to other deployment automation technologies

Adapting the deployment to run on PaaS platforms, and deploying it with Terraform

Adapting the model in order to be deployed with Puppet

procedure

- analysed four deployments of a third-party multi-components application
- for each deployment two strategies were compared
  - using directly the deployment automation technologies
  - leveraging the integrated EDMM ecosystem
- Key Performance Indicators
  - number of lines of code and files written
  - programming languages used

CASE STUDY

Initial deployment with Kubernetes

Migrating from Kubernetes to other deployment automation technologies

Adapting the deployment to run on PaaS platforms, and deploying it with Terraform

Adapting the model in order to be deployed with Puppet

results

- in the initial deployment there is no real gain in using the integrated EDMM environment
- when migrating from one deployment automation technology to another, performing a different deployment, the integrated EDMM ecosystem provides
  - a highly reusable deployment model
  - a graphical and integrated environement
  - a support system to adapt the deployment topology

CONCLUSIONS

starting configuration

- deployment automation technologies and technology lock-in
- the Essential Deployment MetaModel
  - the EDMM Modellling Tool
  - the EDMM Transformation Framework
  - the Decision Support System
- lack of an integrated environment
- simple DSS logic

CONCLUSIONS

contribution

- we integrated the EDMM environment
  - an application developer can now model the deployment and get the technology-specific files without changing environment
- we enhanced the DSS with a rule-based system
  - it now suggests sub-topology into sub-topology replacements
- we assessed the quality of our work with a case study

CONCLUSIONS

future work

- adding more Component Types wuold allow to have more rules, hence a more expressive system
- provide each rule with a cost parameter

Insights

INTEGRATION

mixed approach

- the DSS gets integrated as a library
- the Transformation Framework can be started up as a REST service

cons

- gathers all the downsides of the discussed approaches
- dupplicated code and logic
- hard to maintain
- half of the functionalities in the edmm-core are imported, but not used