The OpenTOSCA Ecosystem

Uwe Breitenbücher, Tobias Binz, Oliver Kopp, Frank Leymann
Institute of Architecture of Application Systems

www.opentosca.org

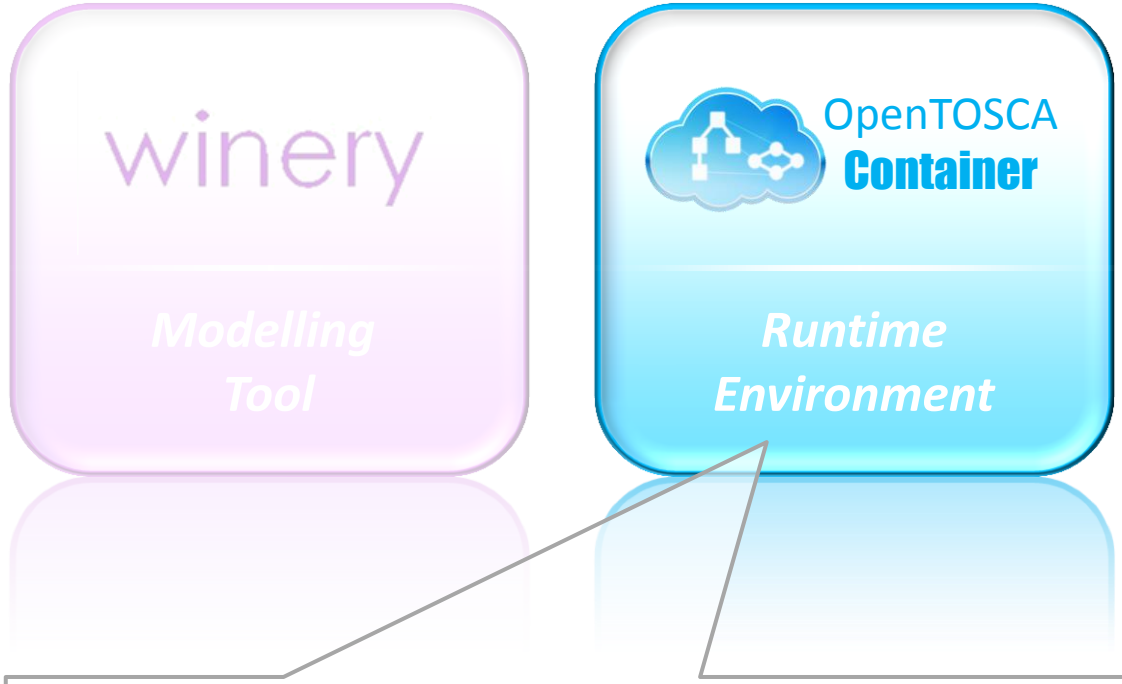
University of Stuttgart
Germany

CLOUDCYCLE
Sicherheit. Standards. Services.

The OpenTOSCA Ecosystem

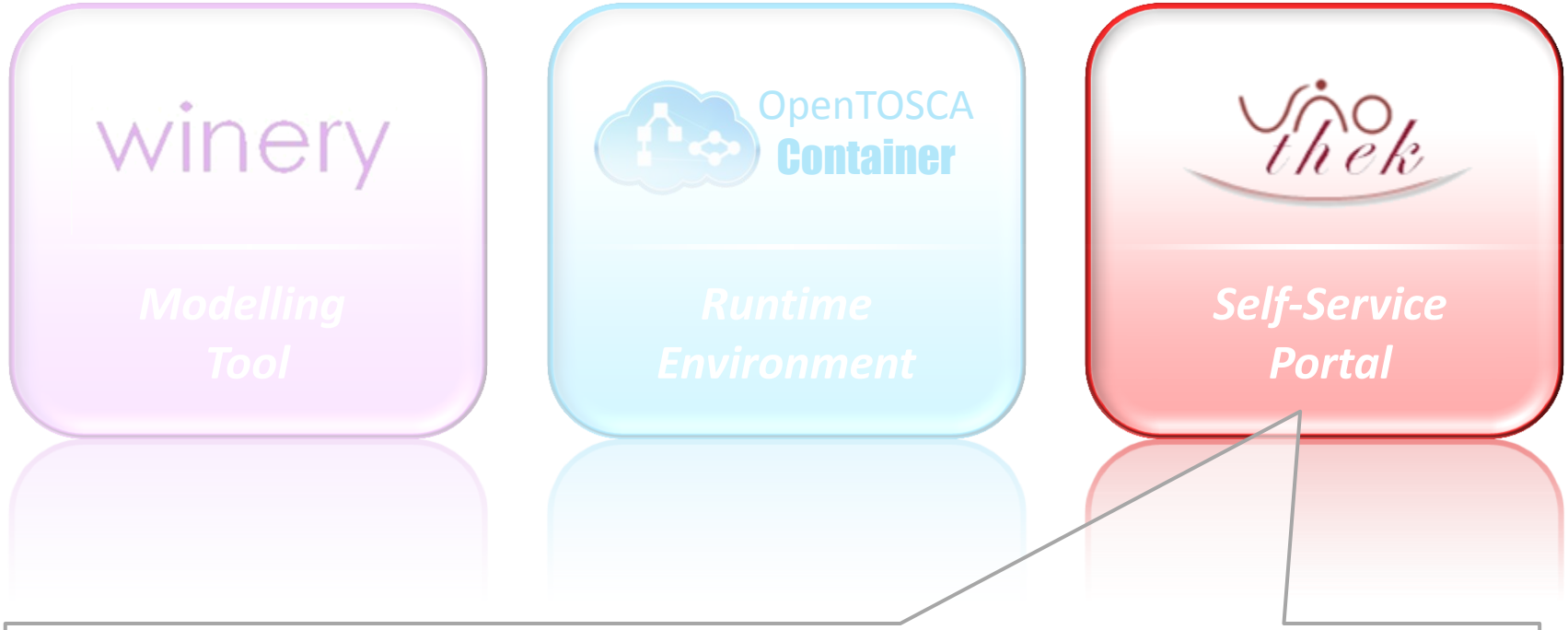
- TOSCA Modelling tool to develop CSARs
  - Graphical Topology Template Modeller
  - Template, Types, and Artifact Management Backend

The OpenTOSCA Ecosystem



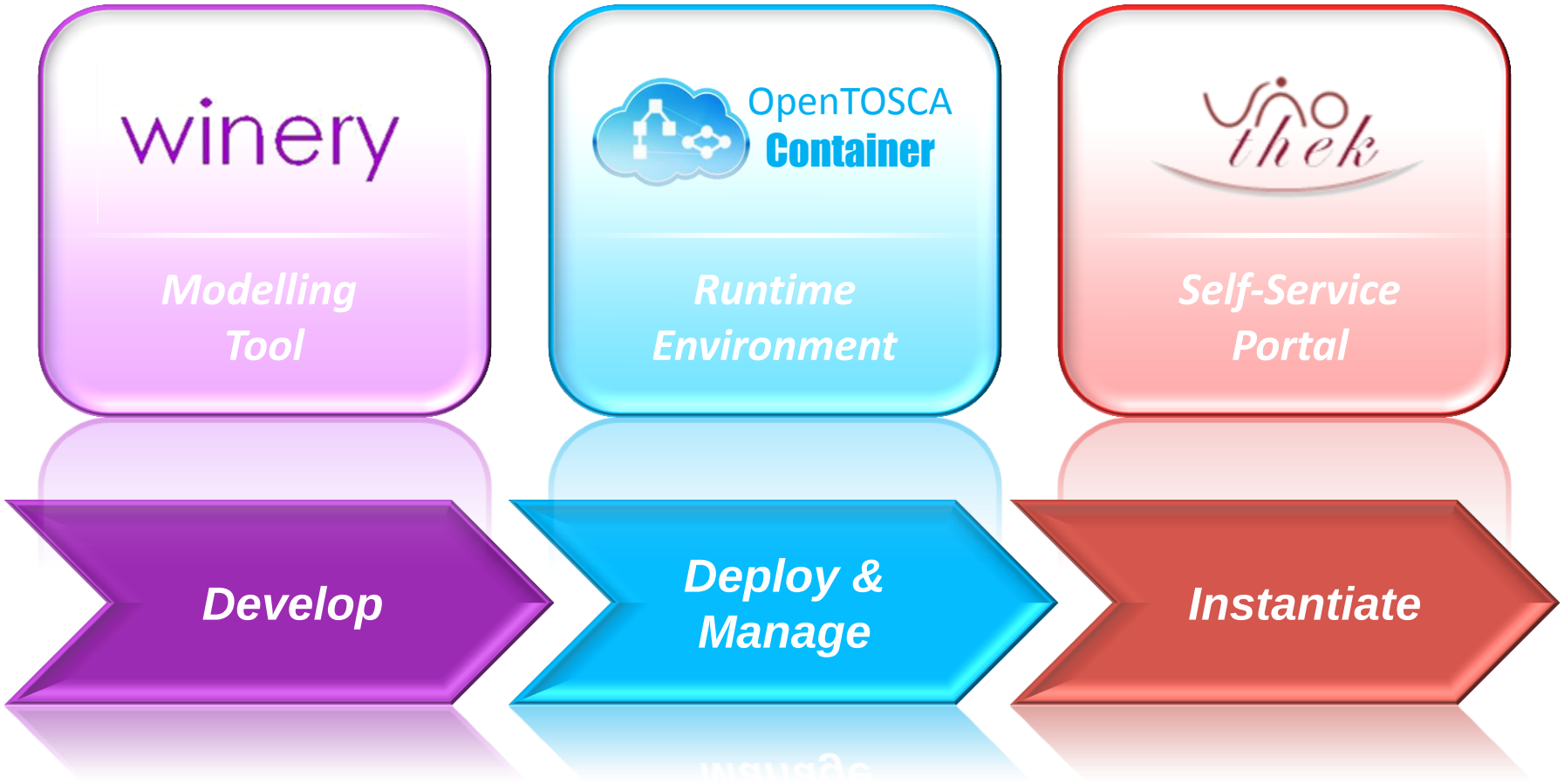
- TOSCA Runtime Environment
  - Supports imperative processing based on BPEL
  - Supports Java- and Script-Implementation Artifacts

The OpenTOSCA Ecosystem



- Self-Service Portal for OpenTOSCA
  - Provides easy graphical interface for users
  - Currently supports the provisioning of applications

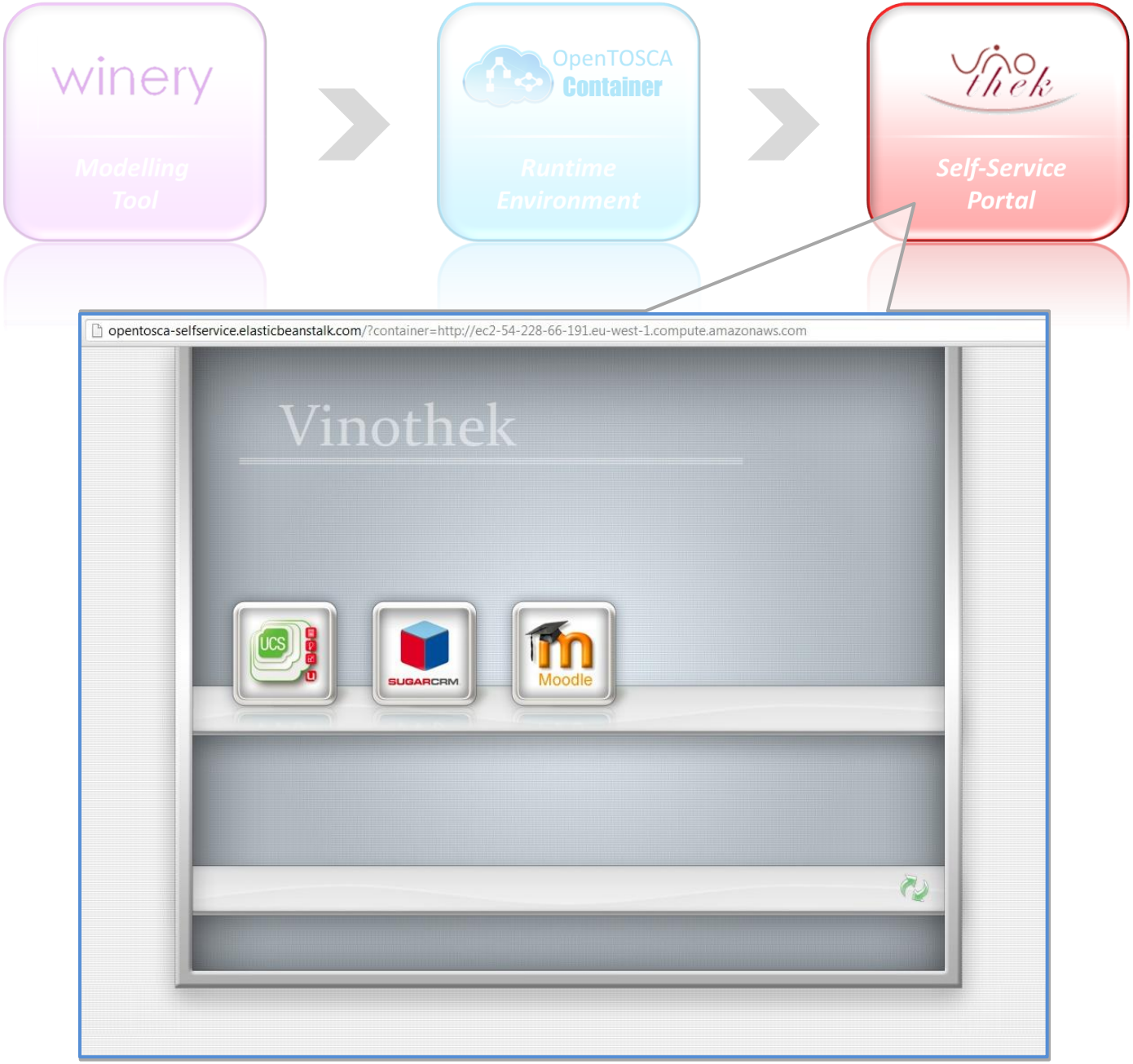
The OpenTOSCA Ecosystem



„Standards-based - Open Source - End-To-End Toolchain“

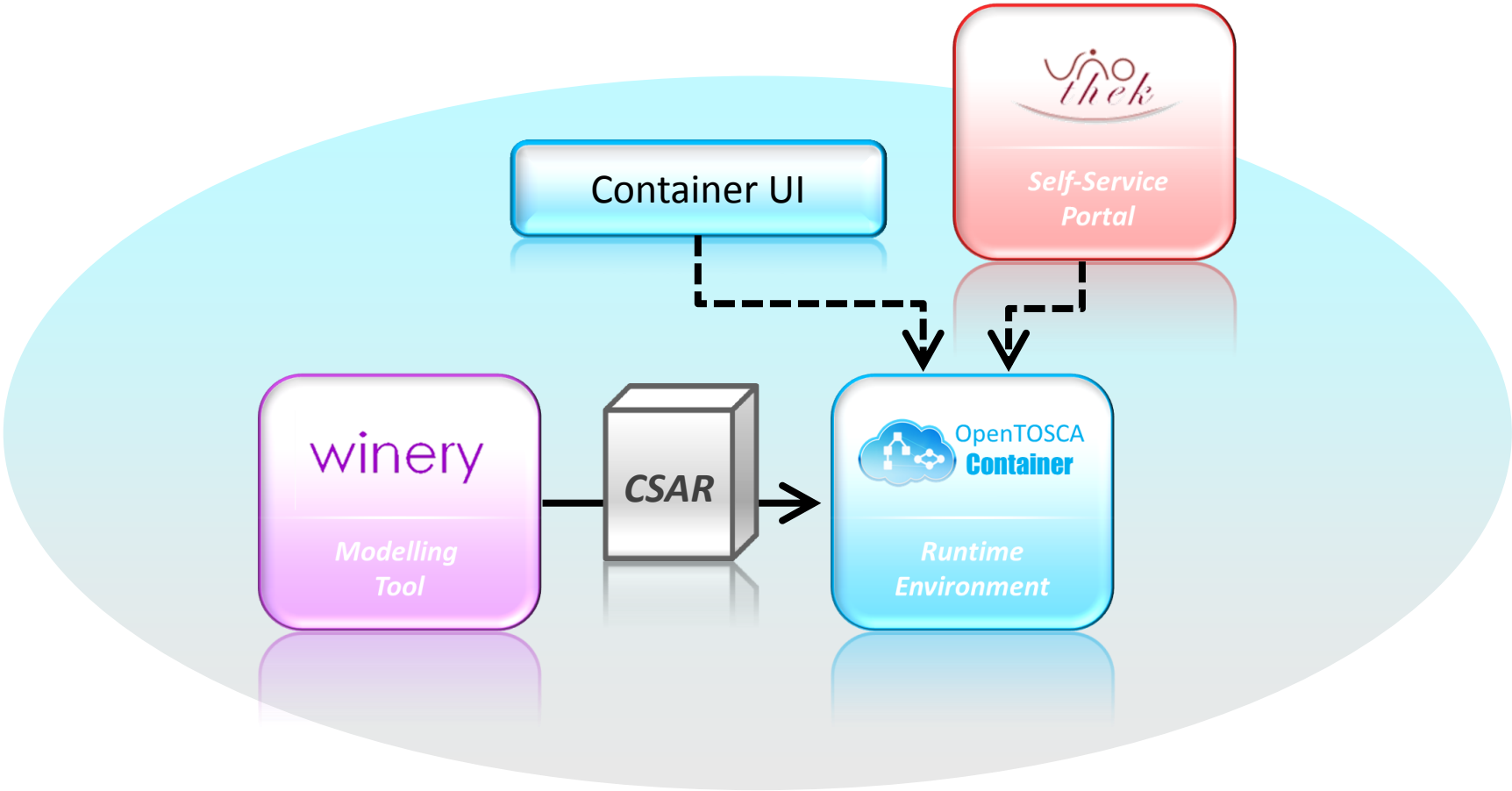

winery
Modelling Tool
OpenTOSCA Container
Runtime Environment
Self-Service Portal
Firefox
Winery Topologymodeller – Moodle
Save
Layout
Align-h (|)
Align-v (-)
Ids
Types
Properties
Deployment Artifacts
Requirements & Capabilities
Delete
Amazon_EC2
ApacheWebServer
BPEL
Debian
Glassfish
IBM_Web_Sphere
MySQL
ODE
PHP
PostgreSQL
Tomcat
Ubuntu
Virtual Machine
Windows
Moodle
(PHP)
calls
connectsTo
hostedOn
ApacheWebServer
(ApacheWebServer)
MySQL
(MySQL)
Ubuntu_1
(Ubuntu)
Ubuntu_2
(Ubuntu)
Amazon_EC2_1
(Amazon_EC2)
Amazon_EC2_2
(Amazon_EC2)
Name
Moodle
min
max


winery
Modelling Tool
OpenTOSCA Container
Runtime Environment
Self-Service Portal
ec2-54-228-66-191.eu-west-1.compute.amazonaws.com:8080/index.jsp?view=overview
Version 1.0
open Tosca
Overview
Deploy CSAR
About
Univention
SugarCRM
Moodle
Upload new CSAR
MoodleServiceTemplate.xml
IMPL-ARTIFACTS
SELF-SERVICE
PLANS
META-INF
IMPORTS
topology.png
DEPL-ARTIFACTS
PhpModule
(depends on)
MoodleWebApp
(connects to)
MoodleDB
(hosted on)
ApacheWebServer
MySql
OsApache
OsMySQL
VmApache
VmMySql
(Server)


winery
Modelling Tool
OpenTOSCA Container
Runtime Environment
Vinothek
Self-Service Portal
opentosca-selfservice.elasticbeanstalk.com/?container=http://ec2-54-228-66-191.eu-west-1.compute.amazonaws.com
Vinothek
UCS
SUGARCRM
Moodle


winery
Modelling Tool
OpenTOSCA Container
Runtime Environment
Vinothek
Self-Service Portal
Vinothek - Moodle
vinothek.opentosca.org/Moodle
Vinothek
Moodle
Moodle is a free software e-learning platform. It can be used in different environments such as education, training and development, and business settings.
Moodle supports typical features of an e-learning platform. It additionally provides some original innovations such as its filtering system.
More information about Moodle can be found on the official Website: http://moodle.com/
Default
Start Instance

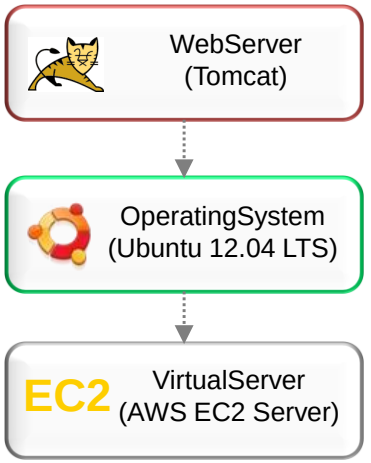
Ecosystem Structure & Relations



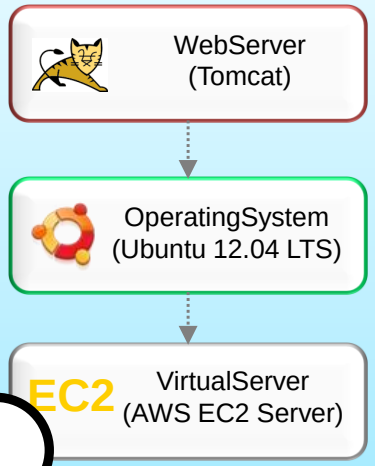
OpenTOSCA Container:
Plan-based Management

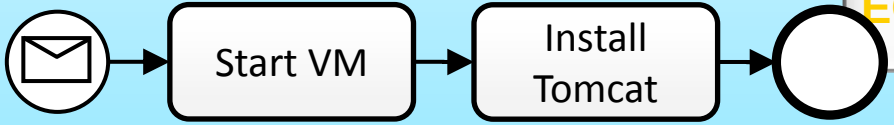
Two Flavors of Processing

- **Declarative → What?**
  - Example: “I want this, realize it!”
  - Runtime interprets topology and does deployment

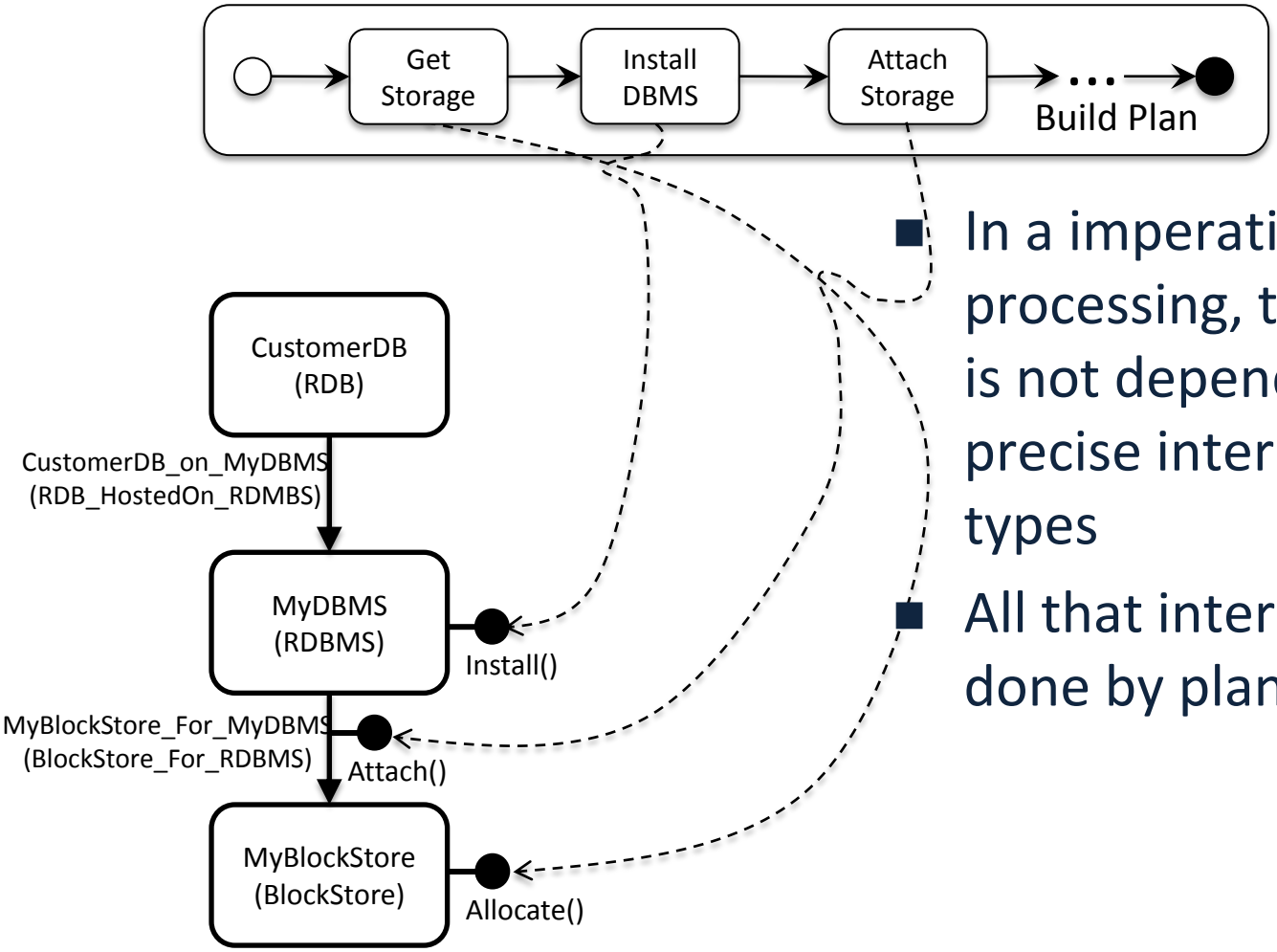


- **Imperative → How?**
  - Example: “First do this, than that.”
  - Management plan explicitly describes each step





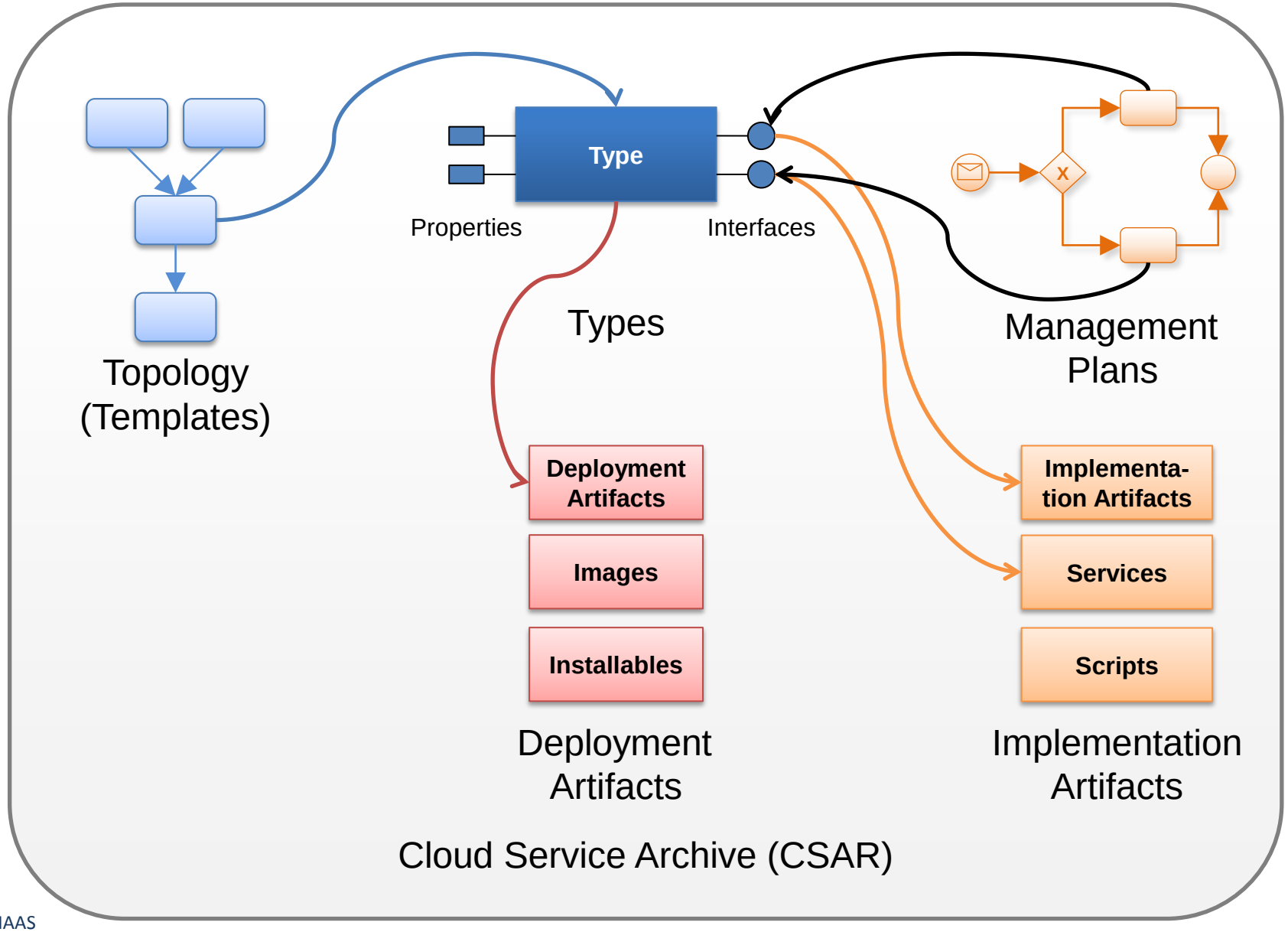
...And Its *Imperative* Processing



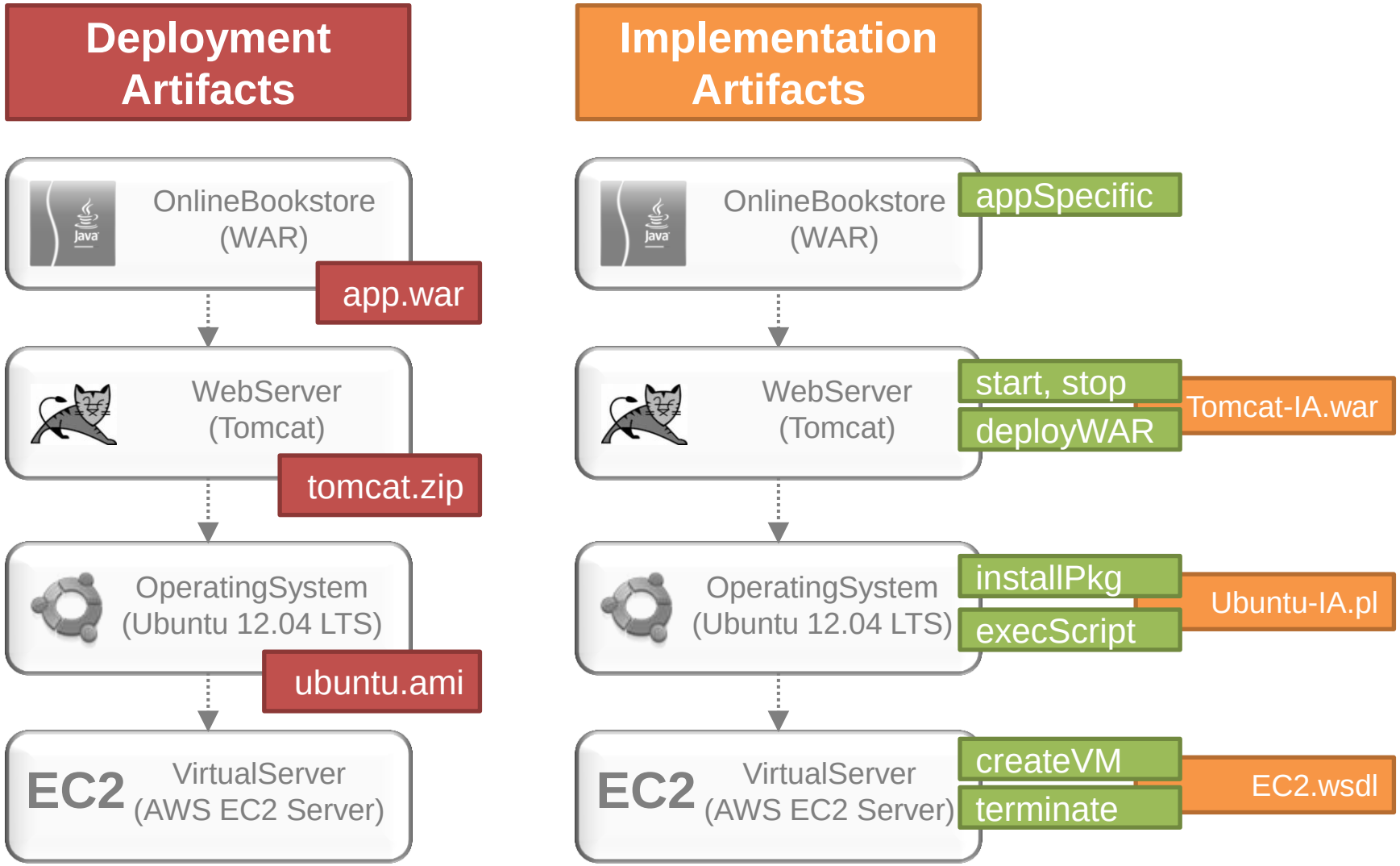
- In a imperative mode of processing, the environment is not dependent on a precise interpretation of the types
- All that interpretation is done by plans

PRO: No precise definition of all types, their processing, their behavior,… needed
CON: Plans must be specified even for “simple” provisioning and decommissioning needed

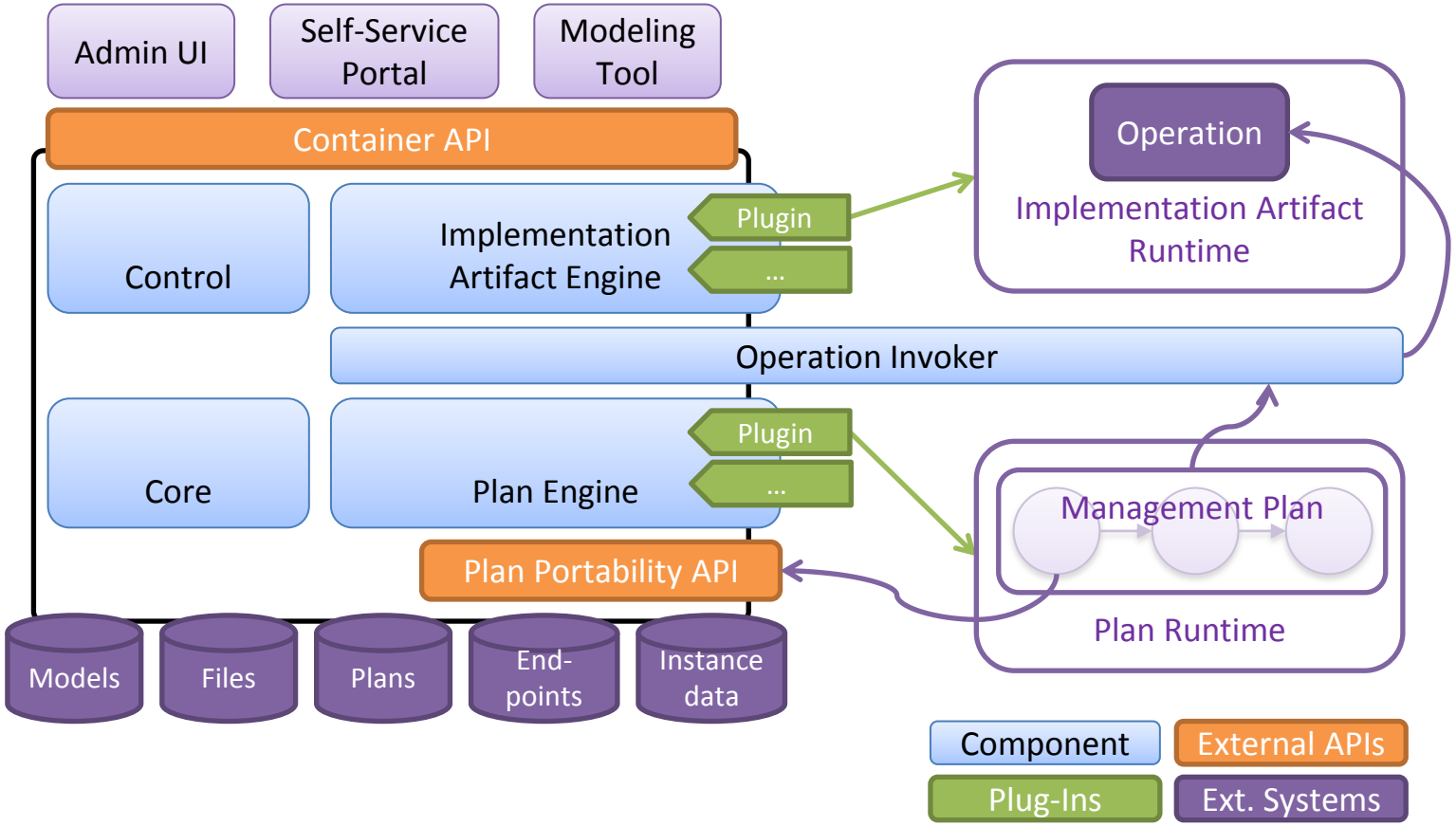
What is contained in a Cloud Service Archive (CSAR)?



Deployment and Implementation Artifacts



OpenTOSCA Architecture Simplified



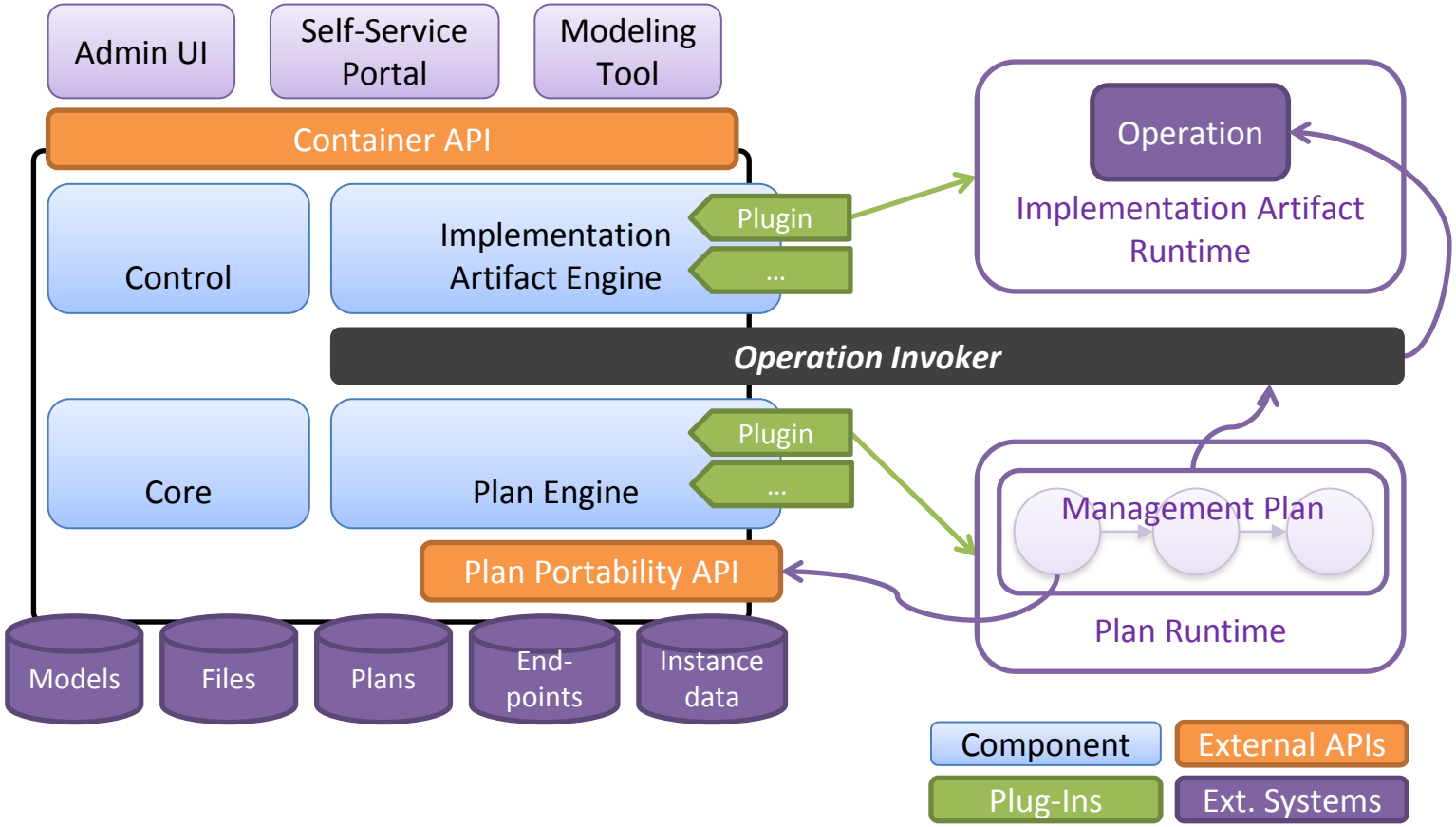
Currently supported:

- ***Implementation Artifacts***
  - Java-based, asynchronous SOAP / HTTP – Webservices (WAR)
  - SH Scripts (ongoing research)
- ***Deployment Artifacts***
  - All possible types ☺

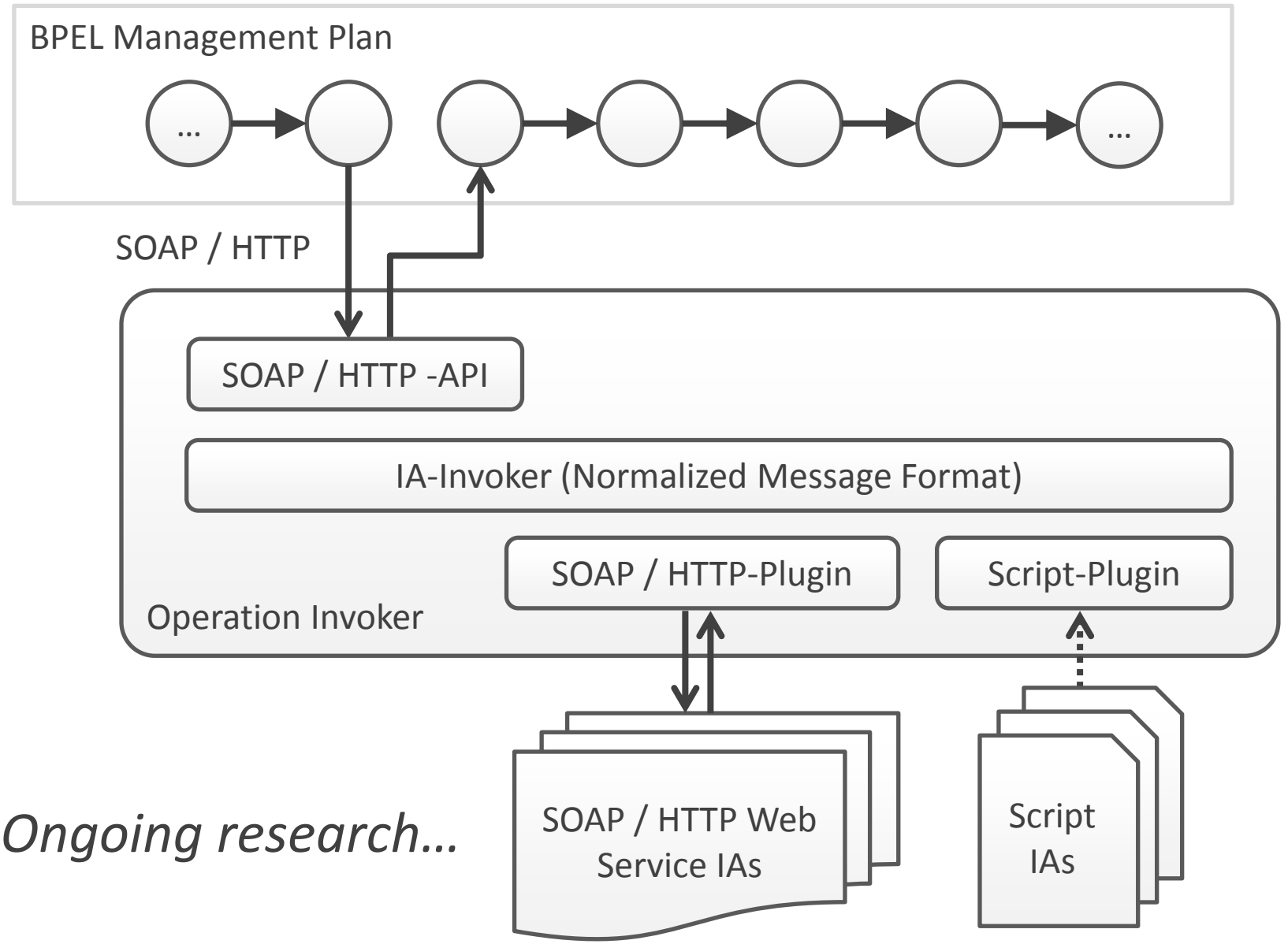
  → Management Plans can process Deployment Artifacts arbitrarily
- ***Node Types / Relationship Types***
  - In general all Node / Relationship Types can be processed
  - If special operations / actions are required, they must be implemented either (i) in the plan or the type must provide (ii) a corresponding Implementation Artifact

  → Custom Node and Relationship Types can be defined and used!
- ***Management Plans***
  - BPEL workflows

Simplified OpenTOSCA Architecture

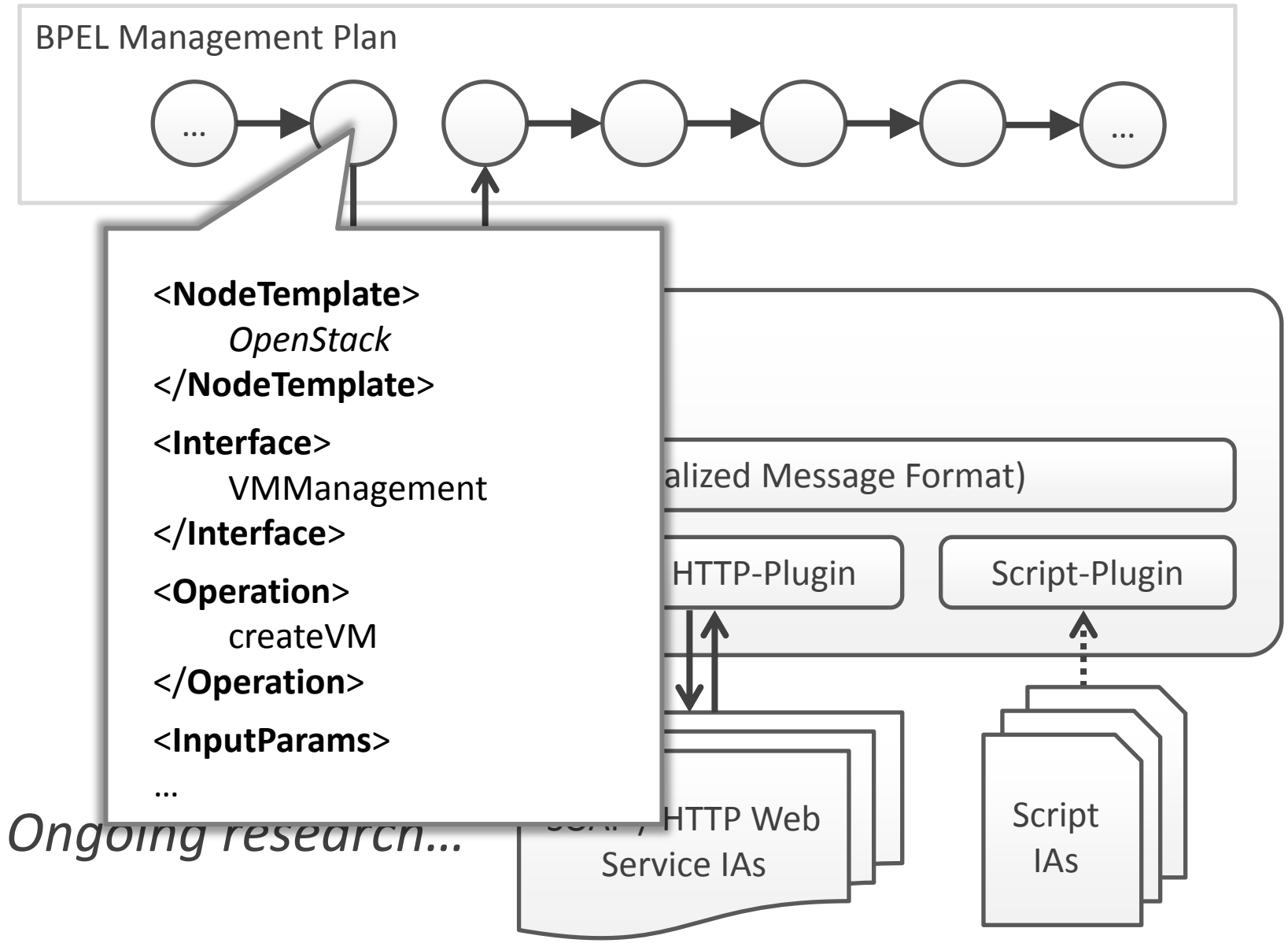


Operation Invoker



Ongoing research…

Operation Invoker



Ongoing research...

The Power of (Open)TOSCA

Major strenghts of the (Open)TOSCA concept

- Node Types can provide management logic by very **different kinds of Implementation Artifacts**

- OpenTOSCA ***directly*** supports (Operation Invoker):
  - Script IAs 
  - Java-based Webservice IAs (WARs) 

- OpenTOSCA ***indirectly*** supports any kind of IA
  - Plans can do whatever they want with arbitrary types of IAs


Major strenghts of the (Open)TOSCA concept

- Enables wrapping many (!) different kinds of managenent interfaces and technologies ☺

  - ***You want to install a special Webserver?***

  → Attach the corresponding Script to the Node Type and let the plan execute it on the operating system

  - ***You want to provision a virtual machine on Provider X?***

  → Attach a Java Webservice IA to the Node Type that implements the „createVM“ operation and call it

Core Concept

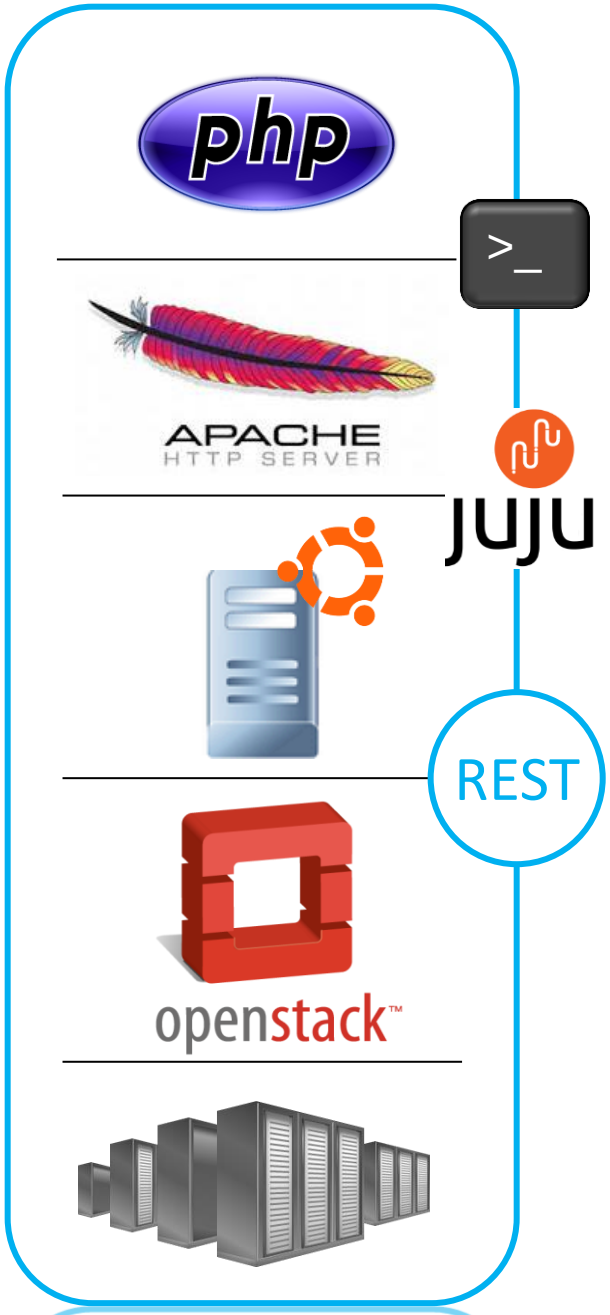
- The core concept of OpenTOSCA:

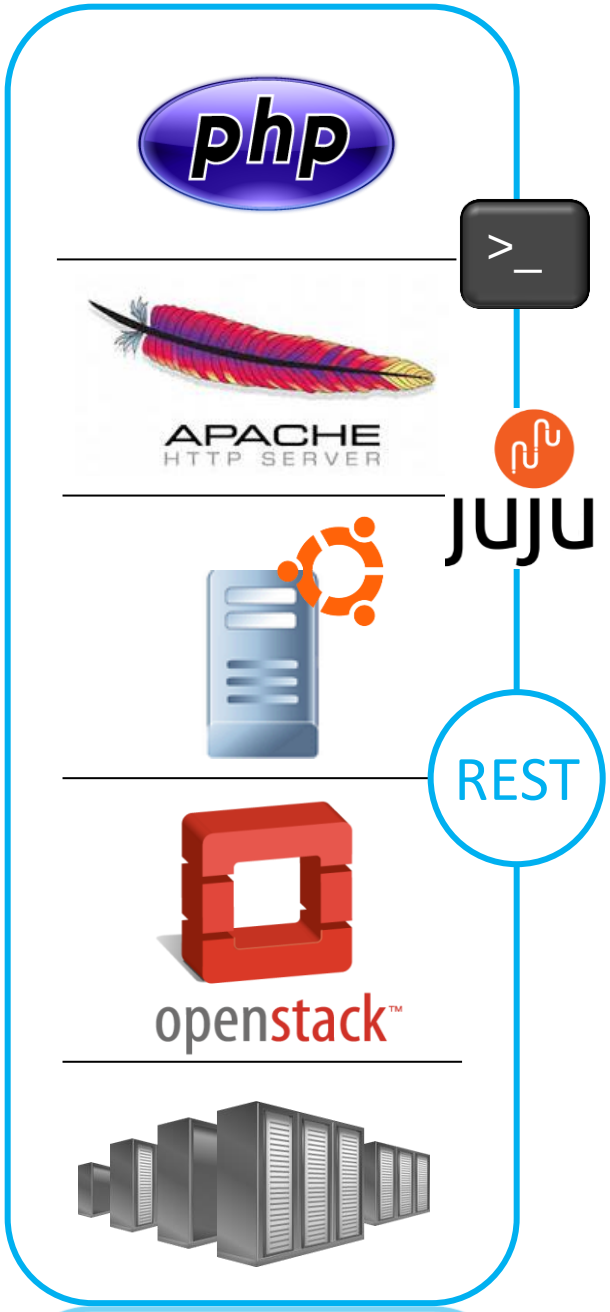
<u>All</u> required management logic is contained in the CSAR

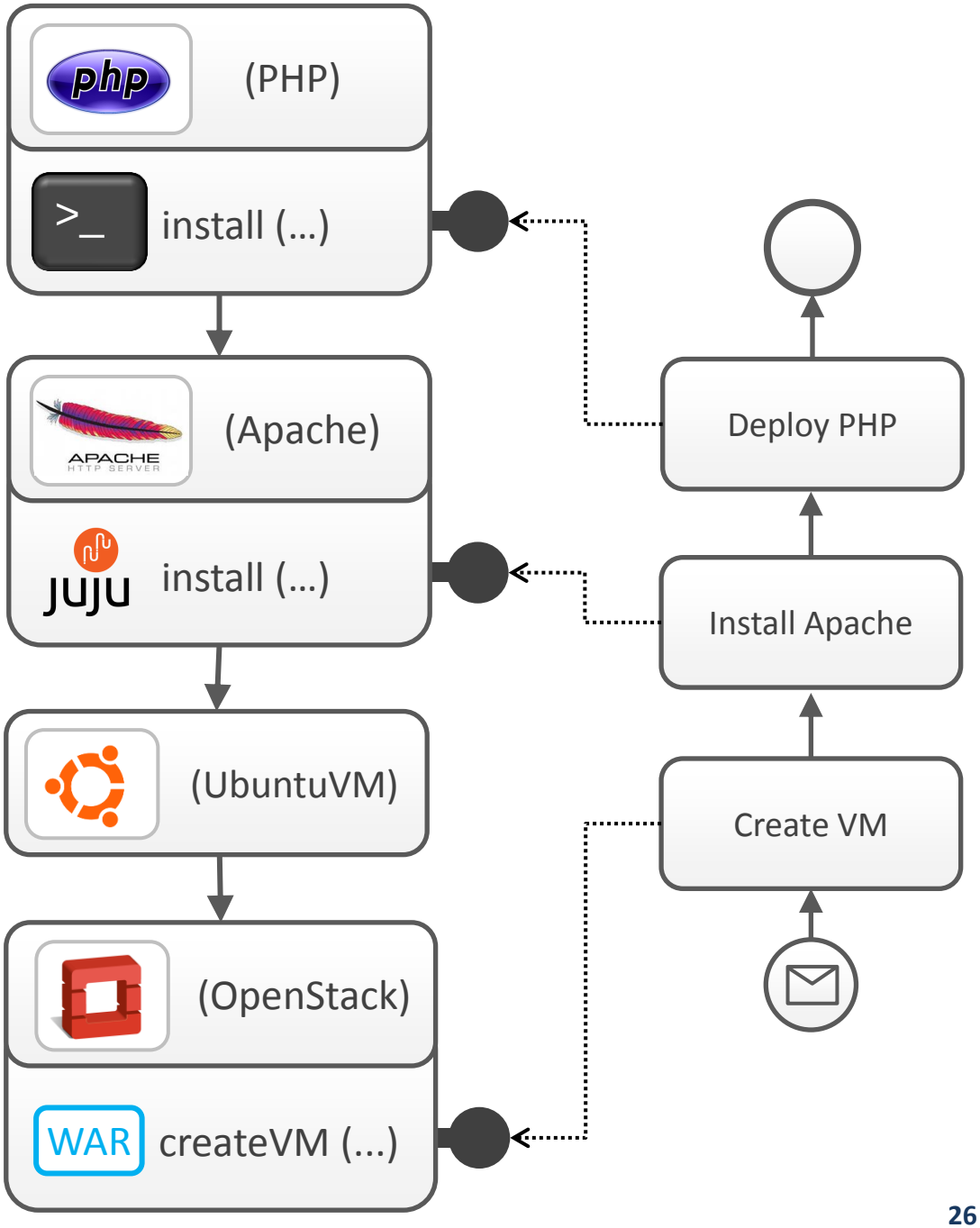
- This leads to completely self-contained CSARs:
  - CSAR provides ***all*** executables:

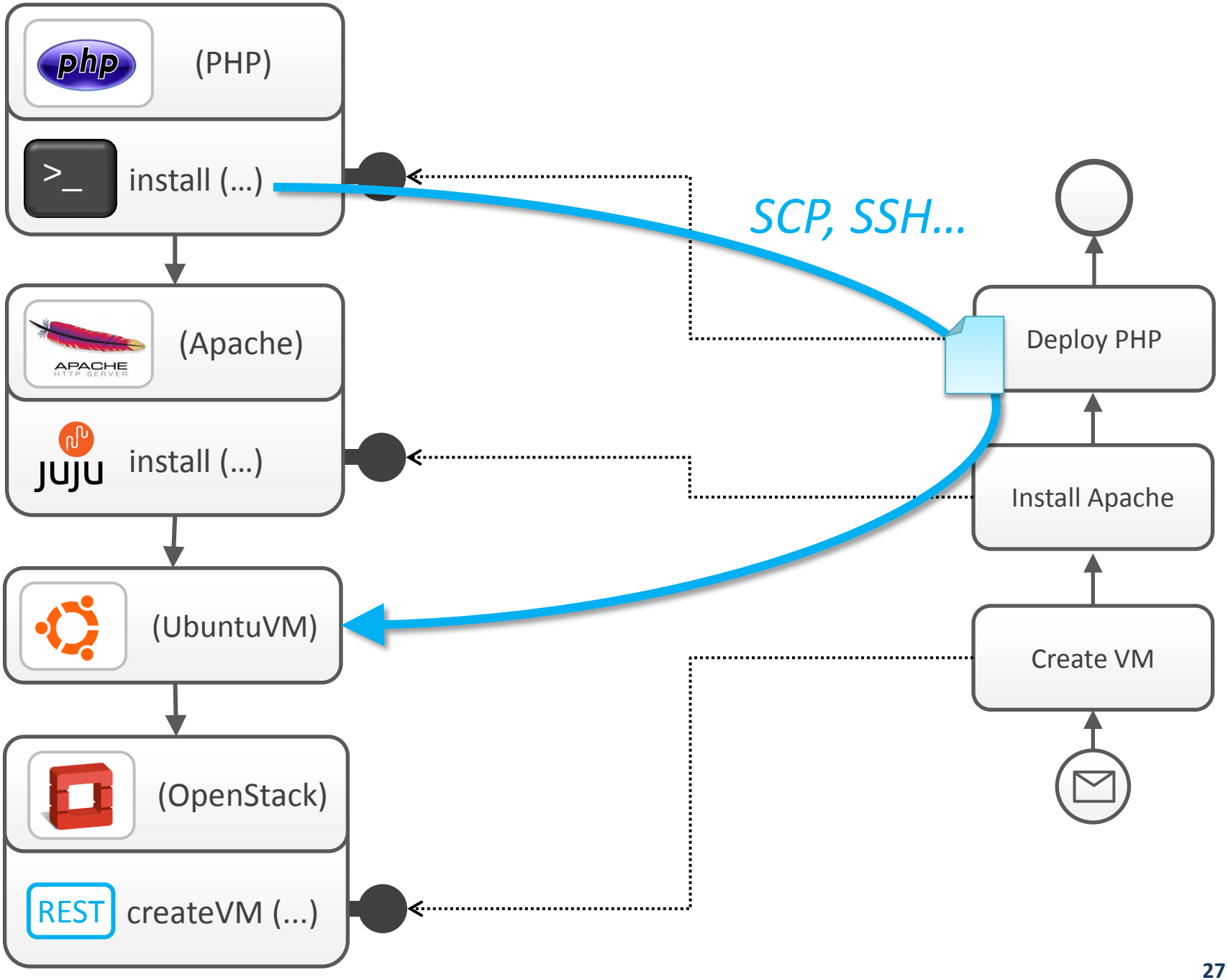
→ Low-level management logic is implementd as IAs

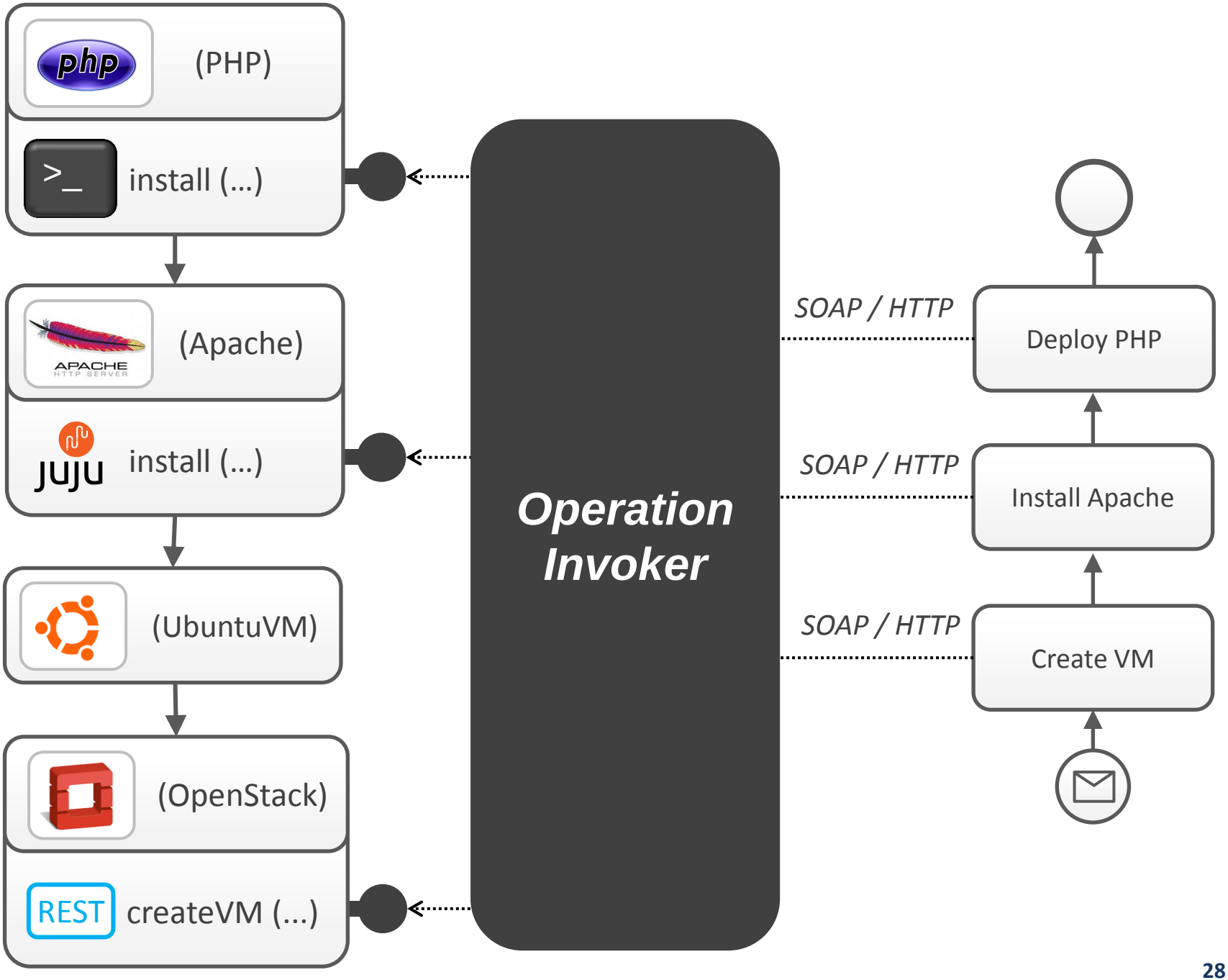
→ Plans orchestrate all these Implementation Artifacts

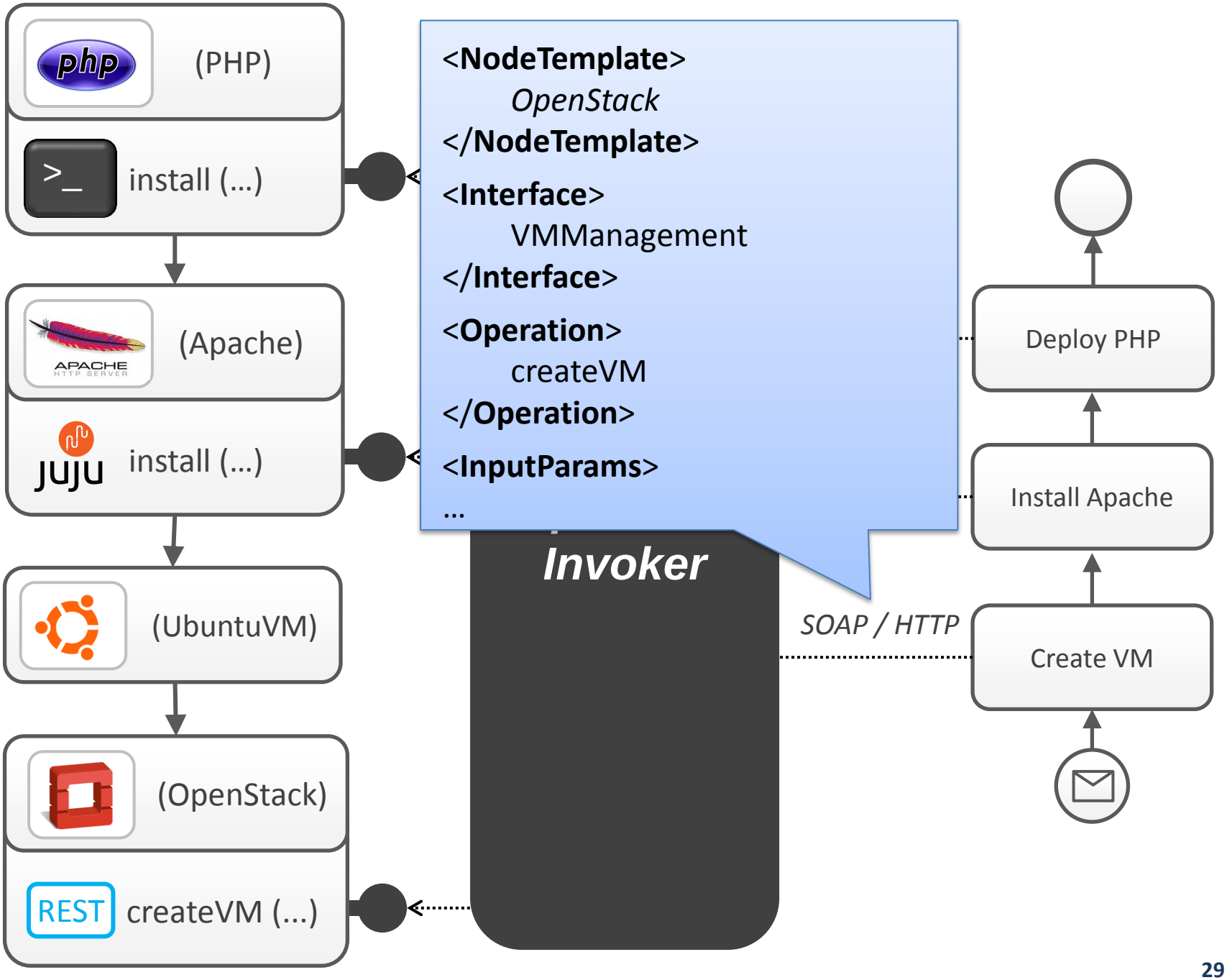

php
>_
APACHE
HTTP SERVER
juju
REST
openstack™

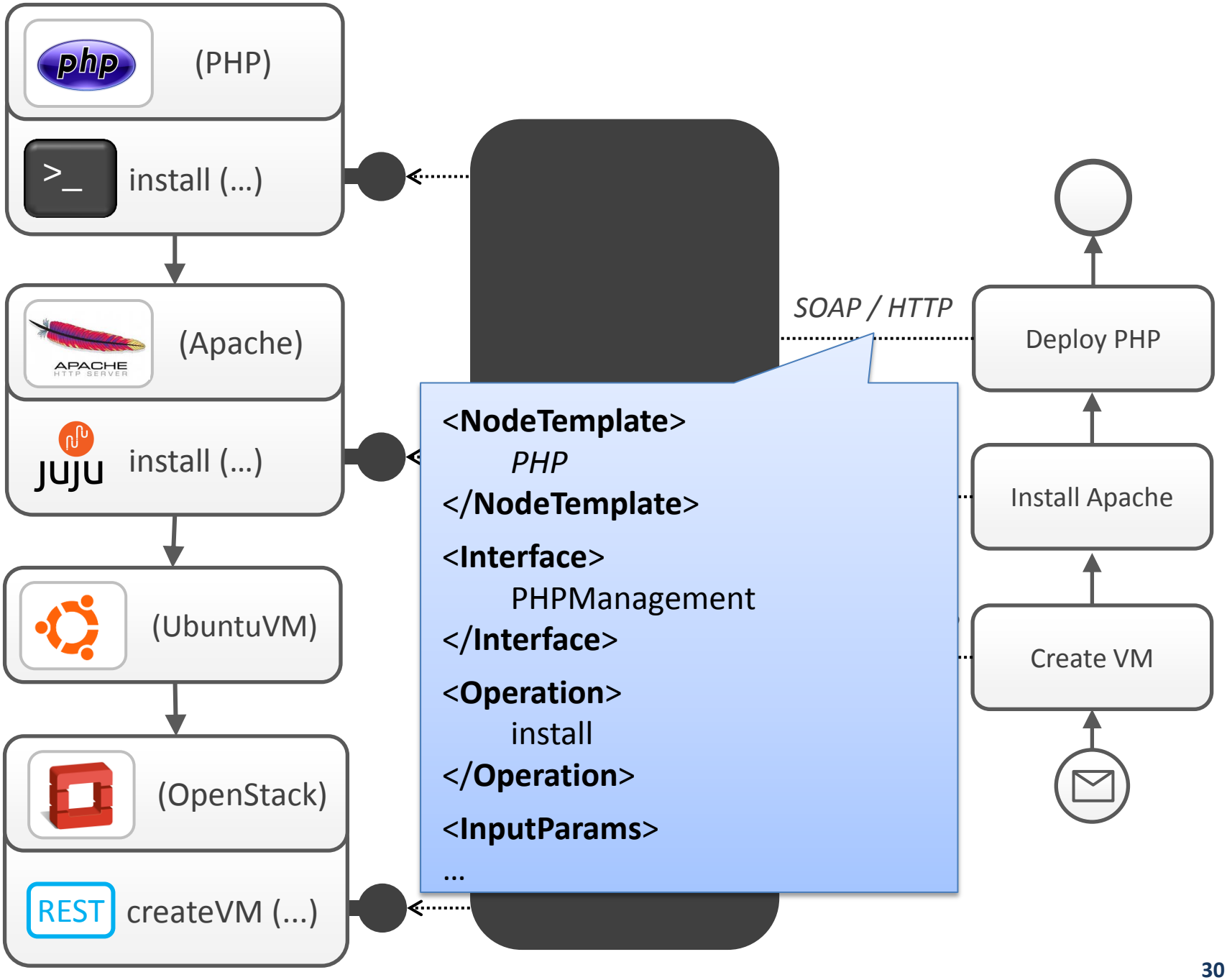

php
APACHE
HTTP SERVER
juju
REST
openstack™

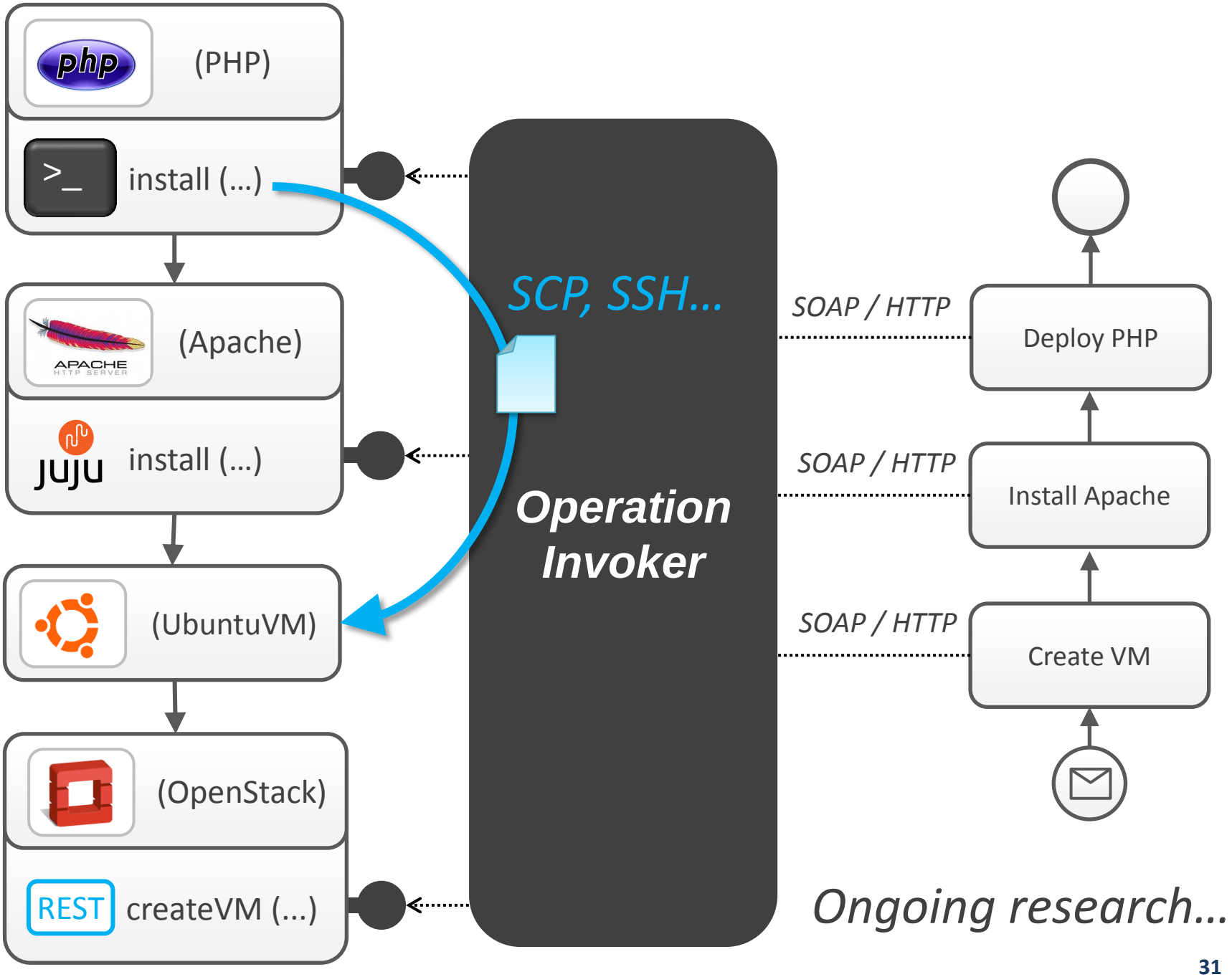

(PHP)
install (...)
(Apache)
install (...)
(UbuntuVM)
(OpenStack)
WAR
createVM (...)
Deploy PHP
Install Apache
Create VM


(PHP)
install (...)
(Apache)
install (...)
(UbuntuVM)
(OpenStack)
REST
createVM (...)
SCP, SSH...
Deploy PHP
Install Apache
Create VM


(PHP)
install (...)
(Apache)
install (...)
(UbuntuVM)
(OpenStack)
REST
createVM (...)
Operation Invoker
SOAP / HTTP
SOAP / HTTP
SOAP / HTTP
Deploy PHP
Install Apache
Create VM

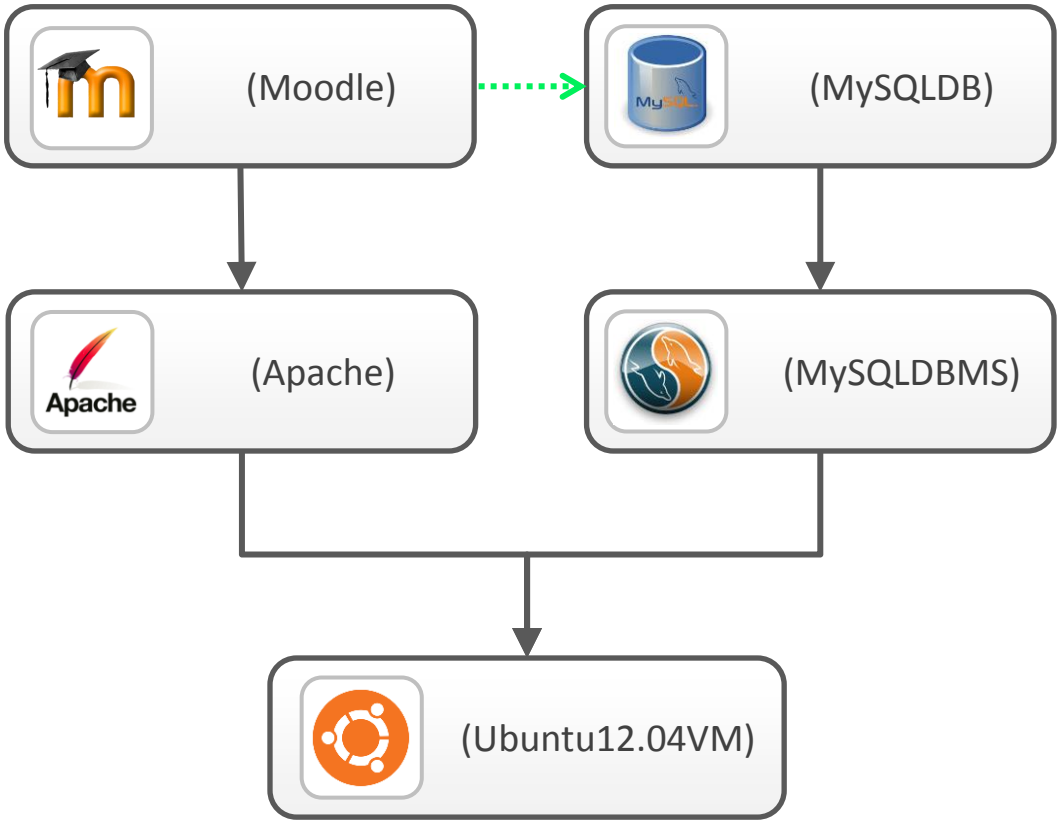

(PHP)
install (...)
(Apache)
install (...)
(UbuntuVM)
(OpenStack)
REST
createVM (...)
<NodeTemplate>
OpenStack
</NodeTemplate>
<Interface>
VMManagement
</Interface>
<Operation>
createVM
</Operation>
<InputParams>
...
Invoker
SOAP / HTTP
Deploy PHP
Install Apache
Create VM


(PHP)
install (...)
(Apache)
install (...)
(UbuntuVM)
(OpenStack)
createVM (...)
SOAP / HTTP
<NodeTemplate>
PHP
</NodeTemplate>
<Interface>
PHPManagement
</Interface>
<Operation>
install
</Operation>
<InputParams>
...
Deploy PHP
Install Apache
Create VM


(PHP)
install (...)
(Apache)
install (...)
(UbuntuVM)
(OpenStack)
REST
createVM (...)
SCP, SSH...
Operation Invoker
SOAP / HTTP
SOAP / HTTP
SOAP / HTTP
Deploy PHP
Install Apache
Create VM
Ongoing research...

Portability & Interoperability

Moodle – A school software based on LAMP

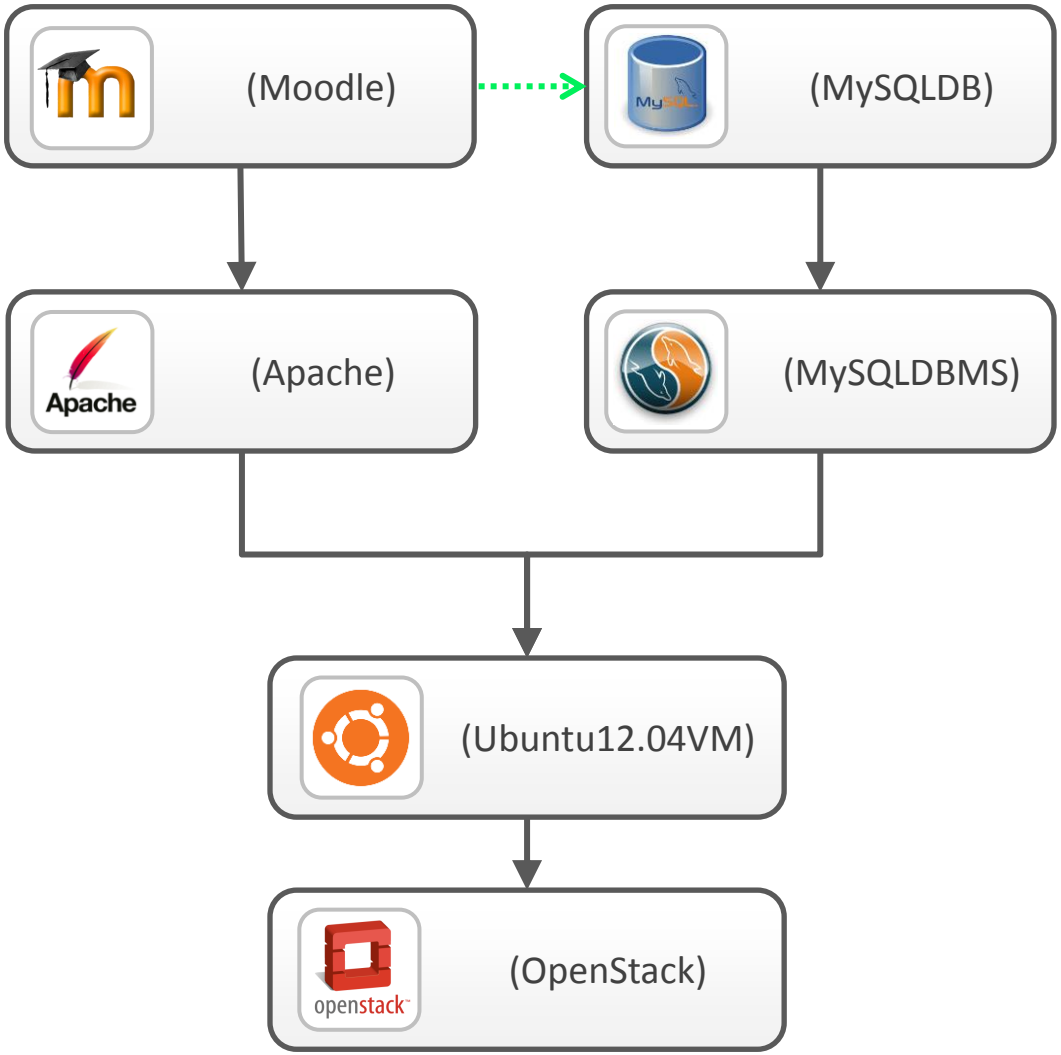


Where to host the application?

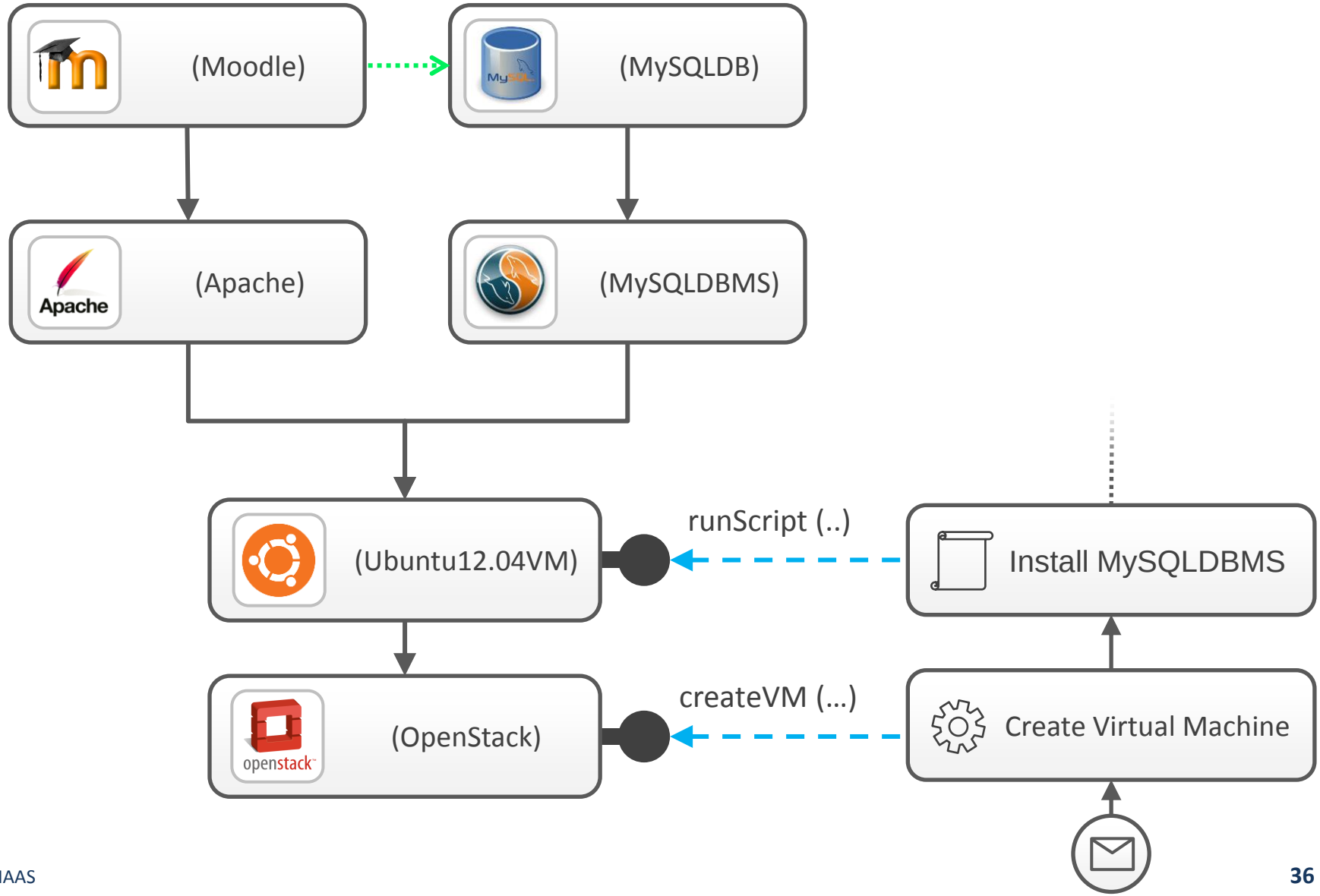
Moodle – A school software based on LAMP

Would be a cool thing, right?

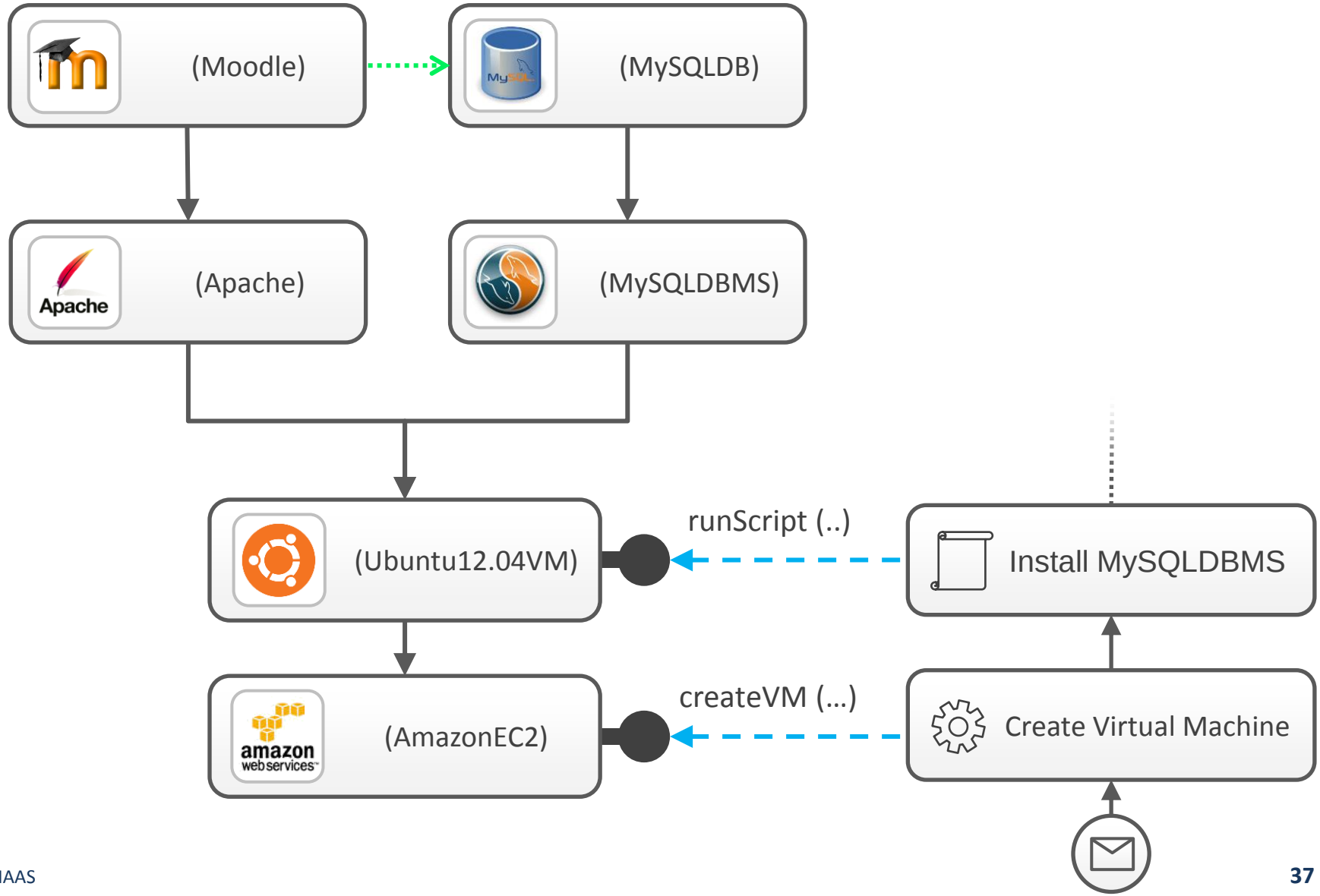
Moodle – A school software based on LAMP



Exchangeable Node Types: OpenStack and AmazonEC2



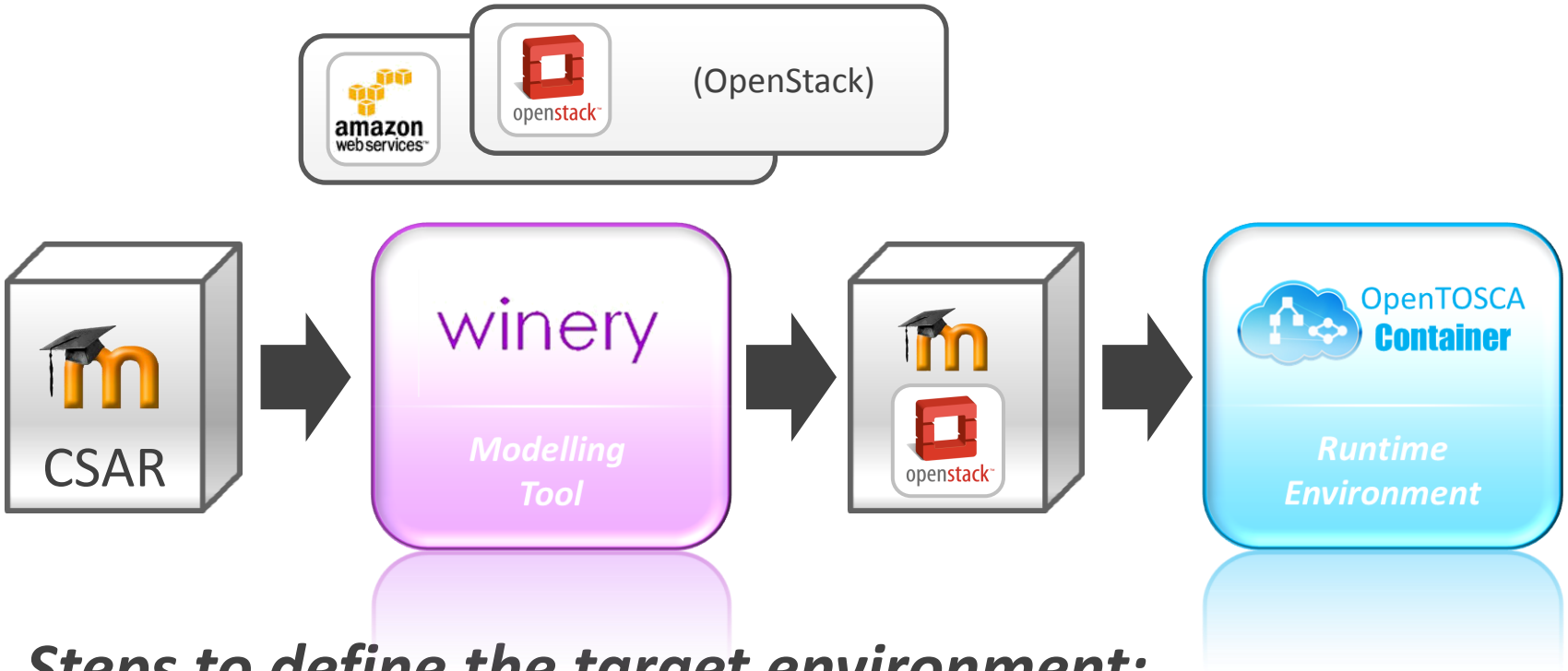
Exchangeable Node Types: OpenStack and AmazonEC2



Input Message of Provisioning Plan

- We were able to design a uniform interface for both „createVM“ operations
  - Parameters of both proprietary APIs are similar and can be mapped
  - For example, Amazon Region = OpenStack Endpoint
  - etc.
- ***Identical Interfaces = Identical Plan*** ☺
- All these parameters are exposed to the input message of the plan
  - In general, very flexible (and reusable!) implementation of Management Plans possible

The „workflow“ of defining the target environment



Steps to define the target environment:

1. *Open CSAR in Winery*
2. *Put the desired Node Type into the Topology*
3. *Safe CSAR and deploy it in OpenTOSCA*

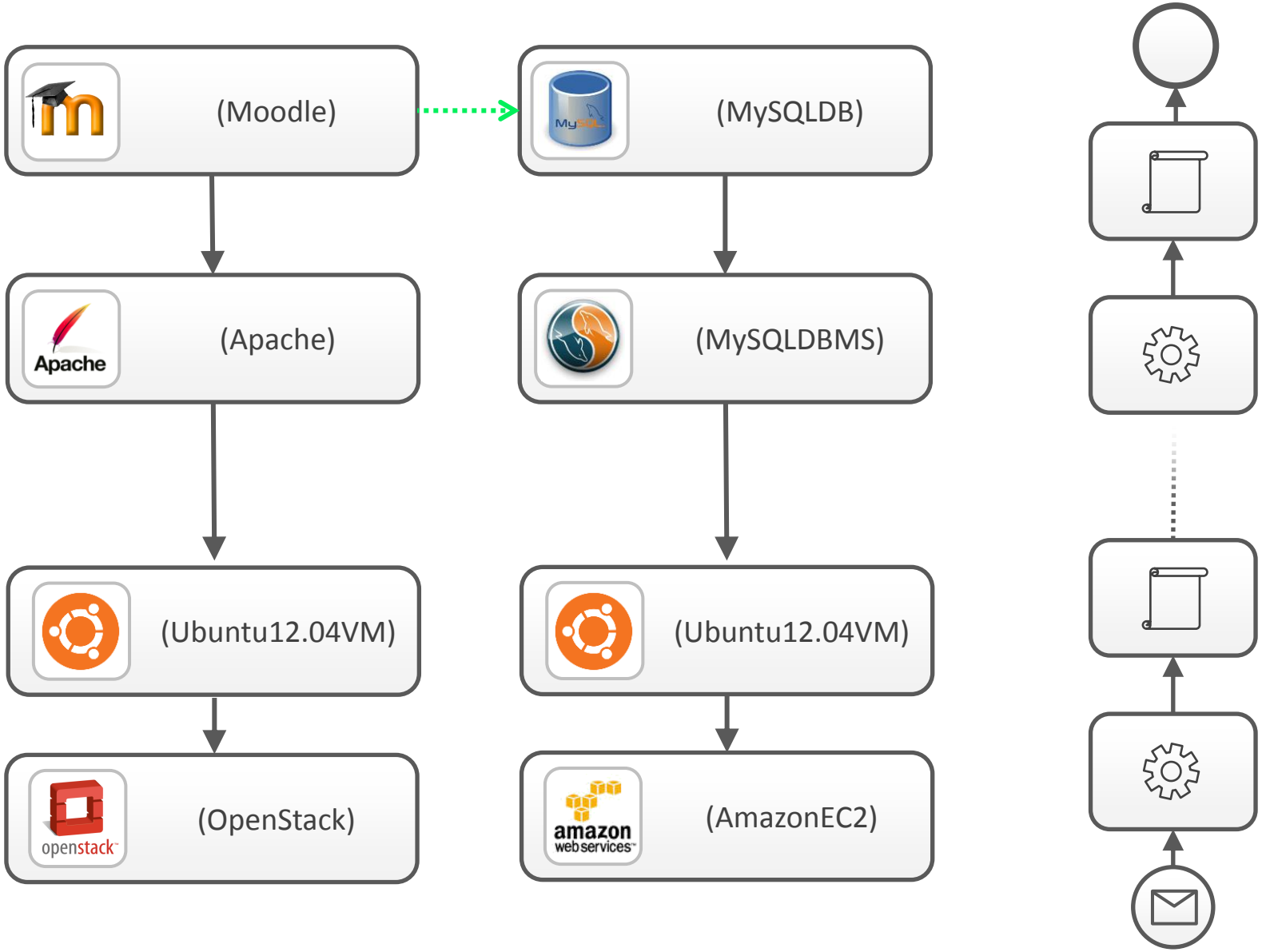
Self-contained CSARs

- This concept enables creating ***self-contained CSARs***
  - Contain management logic of different layers
    - IAs orchestrate infrastructure, platform, or software services
    - IAs to install components, e.g., scripts
  - All the logic is shipped with the CSAR

→ OpenTOSCA is ***not*** coupled to any specific Cloud or Management technology / provider

→ Can be ***extended by implementing new Node Types without modifying the container***

Hybrid Applications

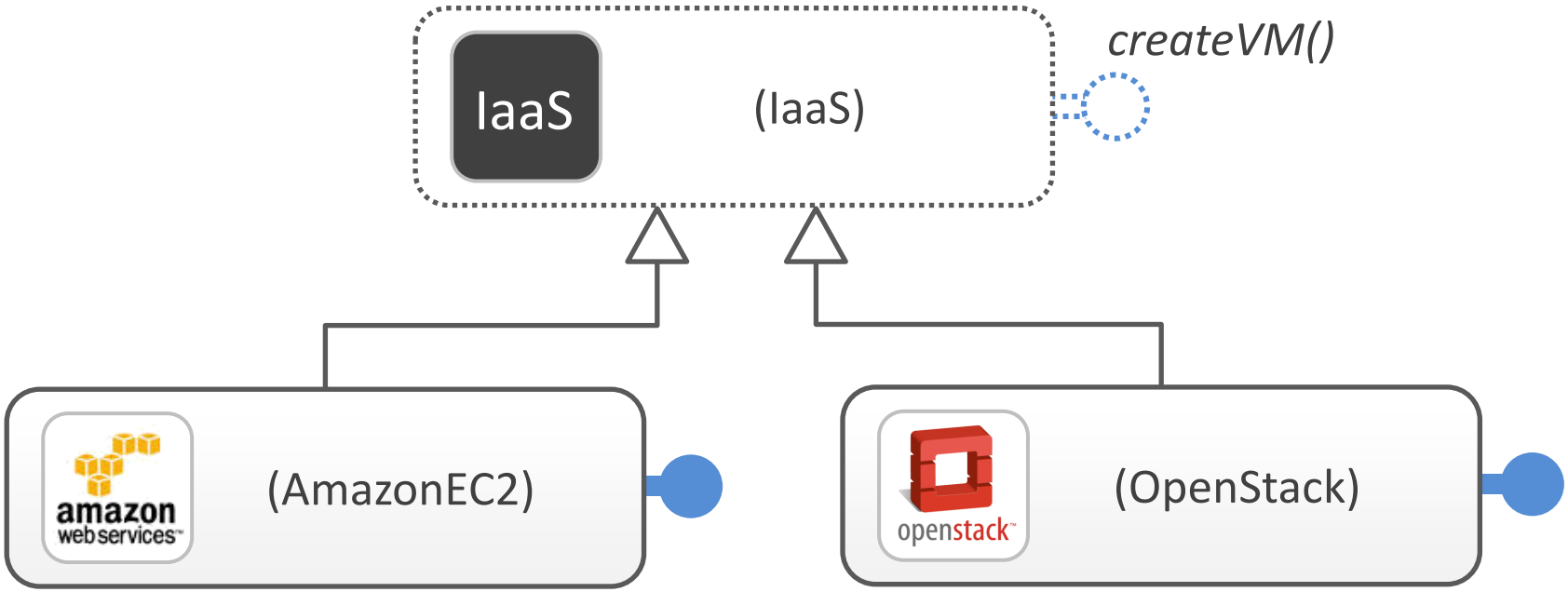


Advantages

- The OpenTOSCA container is ***completely independent from any concrete technology or provider***

- Management Plans can implement ***very complex management flows*** that act directly on the operations provided by Node Types…
  - … which can be customized arbitrarily

Advantages



- Standardizing (abstract) Node Types additionally supports this as it enables a seamless exchange
  - … no need to adapt plans for other providers ☺
  - … only the implementation ***IN*** the CSAR changes

  → Very extensible! Supports portability!

Advantages

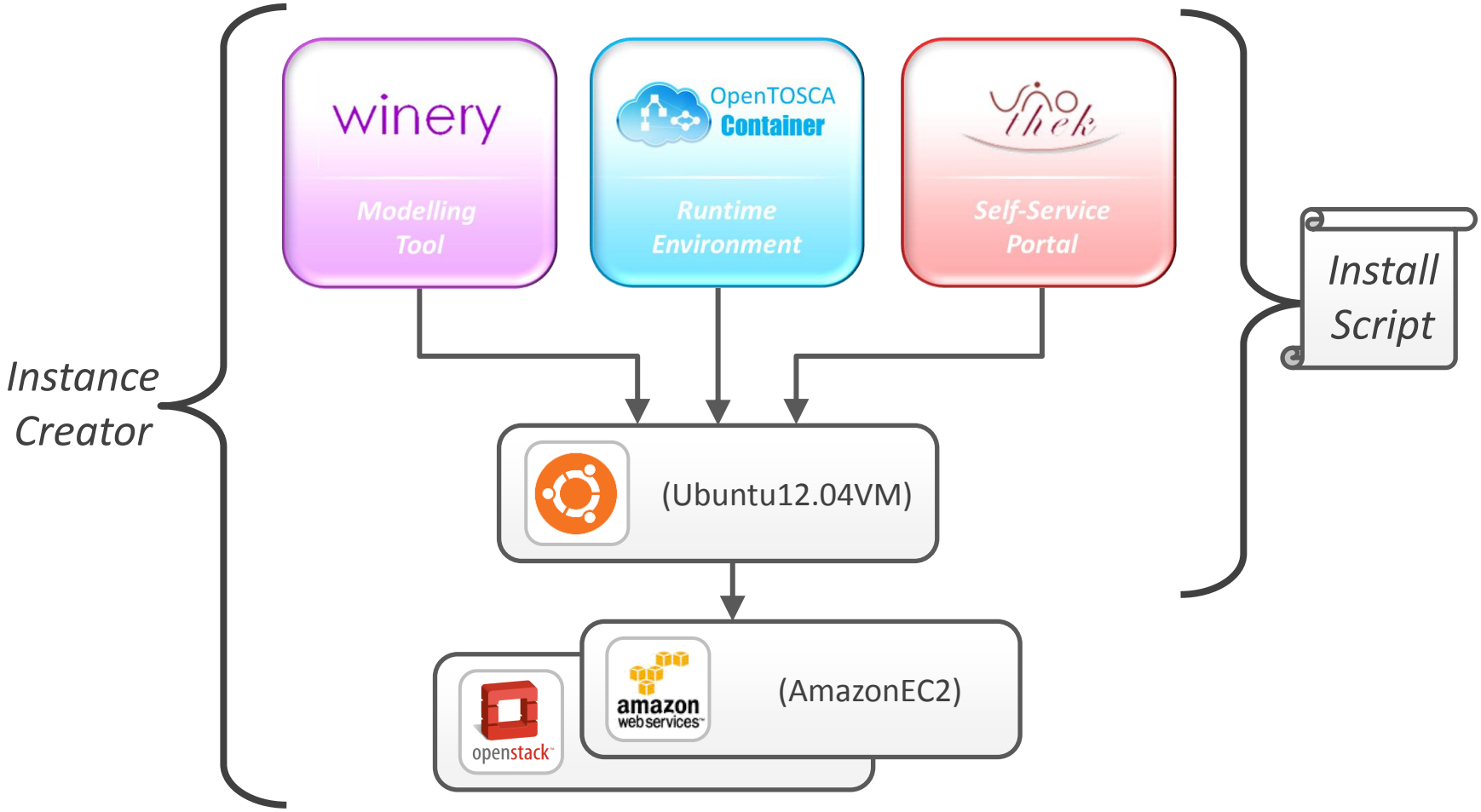
- Because the whole management logic is completely contained in the CSAR, the execution is the same every time…
  - …on each TOSCA Runtime Environment that is able to process the Plans and Implementation Artifacts

Challenges and Problems

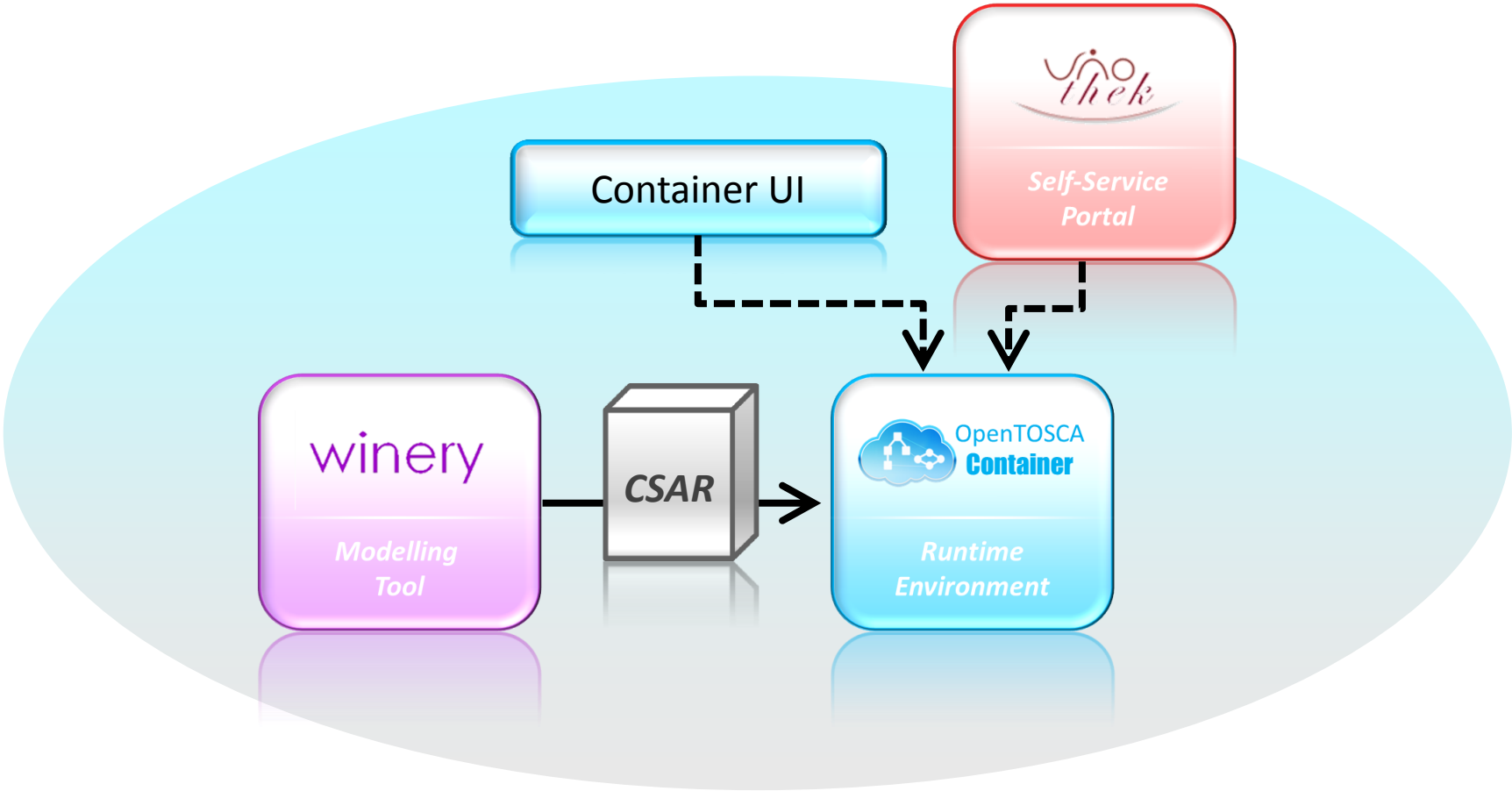
- Implementing „good“ Node Types not easy…
  - Interfaces and properties must be defined
  - Management logic must be implemented
  - However, as Java is supported for IAs, this is possible! ☺
- Management Plans must be created
  - However, the Operation Invoker eases that ☺
    - Abstraction from IA implementations
    - Uniform and simple SOAP / HTTP interface
    - Asynchronous processing supported
    - BPEL fragments available

How to work with the OpenTOSCA Ecosystem?

Install OpenTOSCA



www.demo.opentosca.org



Thank you for your attention!