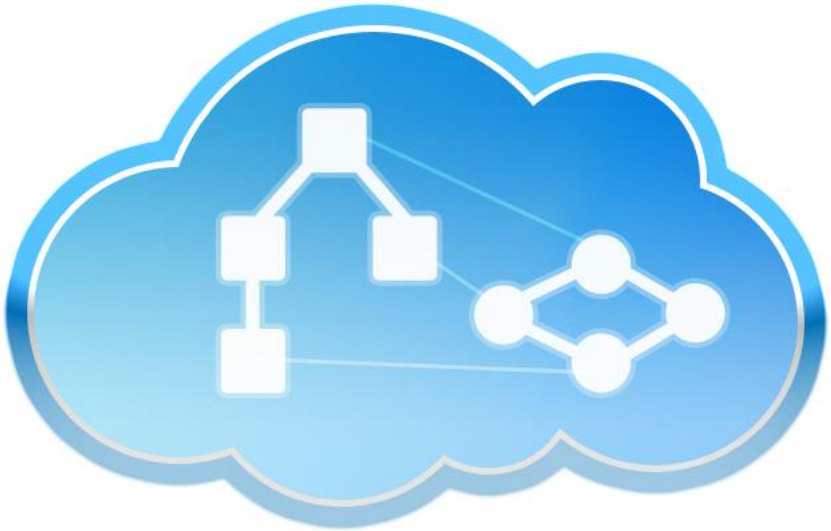


OpenTOSCA

Release v1.1

Universität Stuttgart

Contact: info@opentosca.org

Documentation Version: March 11, 2014

Current version: http://files.opentosca.de

NOTICE

This work has been supported by the Federal Ministry of Economics and Technology (http://www.bmwi.de/EN/root.html) as part of the CloudCycle project (01MD11023). For more information see http://www.cloudcycle.org/en/

This product includes software from the project "OpenTOSCA", whose first implementation was developed by Christian Endres, Matthias Fetzer, Markus Fischer, Nedim Karaoğuz, Kálmán Képes, Rene Trefft, and Michael Zimmermann as a study project ("Studienprojekt") at the Institute of Architecture of Application Systems (IAAS) and Institute for Parallel and Distributed Systems - Applications of Parallel and Distributed Systems (IPVS/AS) of the University of Stuttgart from 2011 to 2012. For more information see http://www.iaas.uni-stuttgart.de and http://www.ipvs.uni-stuttgart.de/as.html

This product includes software from the project "OpenTOSCA Build Plan Generator“ which was originally developed by Kálmán Képes as a bachelor thesis at the Institute of Architecture of Application Systems (IAAS) of the University of Stuttgart from January 2013 to July 2013. For more information see http://www.iaas.uni-stuttgart.de

This product includes software from the project "OpenTOSCA File Service" which was originally developed by Rene Trefft as a Bachelor Thesis at the Institute of Architecture of Application Systems (IAAS) of the University of Stuttgart from December 2012 to June 2013. For more information see http://www.iaas.uni-stuttgart.de

This product includes software from the project "OpenTOSCA Service Invoker" which was originally developed by Michael Zimmermann as a bachelor thesis at the Institute of Architecture of Application Systems (IAAS) of the University of Stuttgart from January 2013 to July 2013. For more information see http://www.iaas.uni-stuttgart.de

This product includes software from the project "Plan Invocation Framework" which was originally developed by Christian Endres as a bachelor thesis at the Institute of Architecture of Application Systems (IAAS) of the University of Stuttgart from November 2012 to May 2013. For more information see http://www.iaas.uni-stuttgart.de

This work has been supported by the Federal Ministry of Economics and Technology as part of the CloudCycle project (01MD11023).

Gefördert durch:

Bundesministerium für Wirtschaft und Technologie

Förderschwerpunkt:

Trusted Cloud

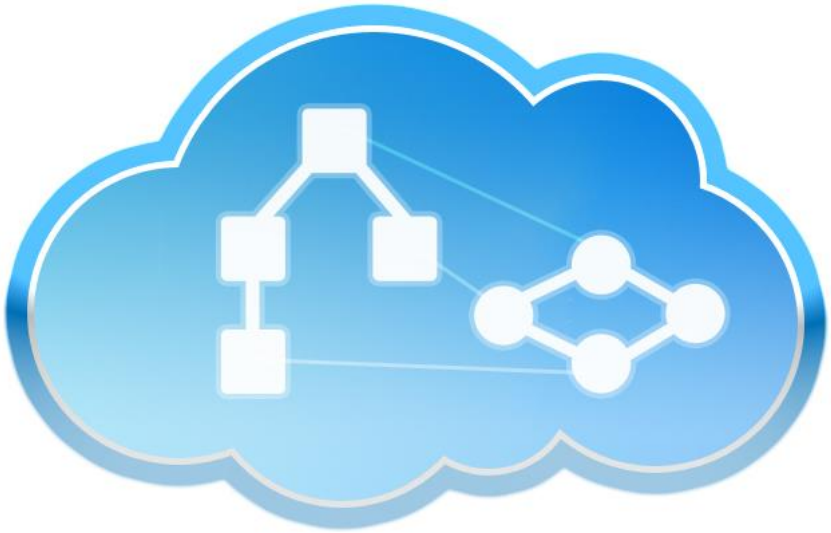
Projektträger:

DLR Deutsches Zentrum für Luft- und Raumfahrt e.V. Projektträger im DLR

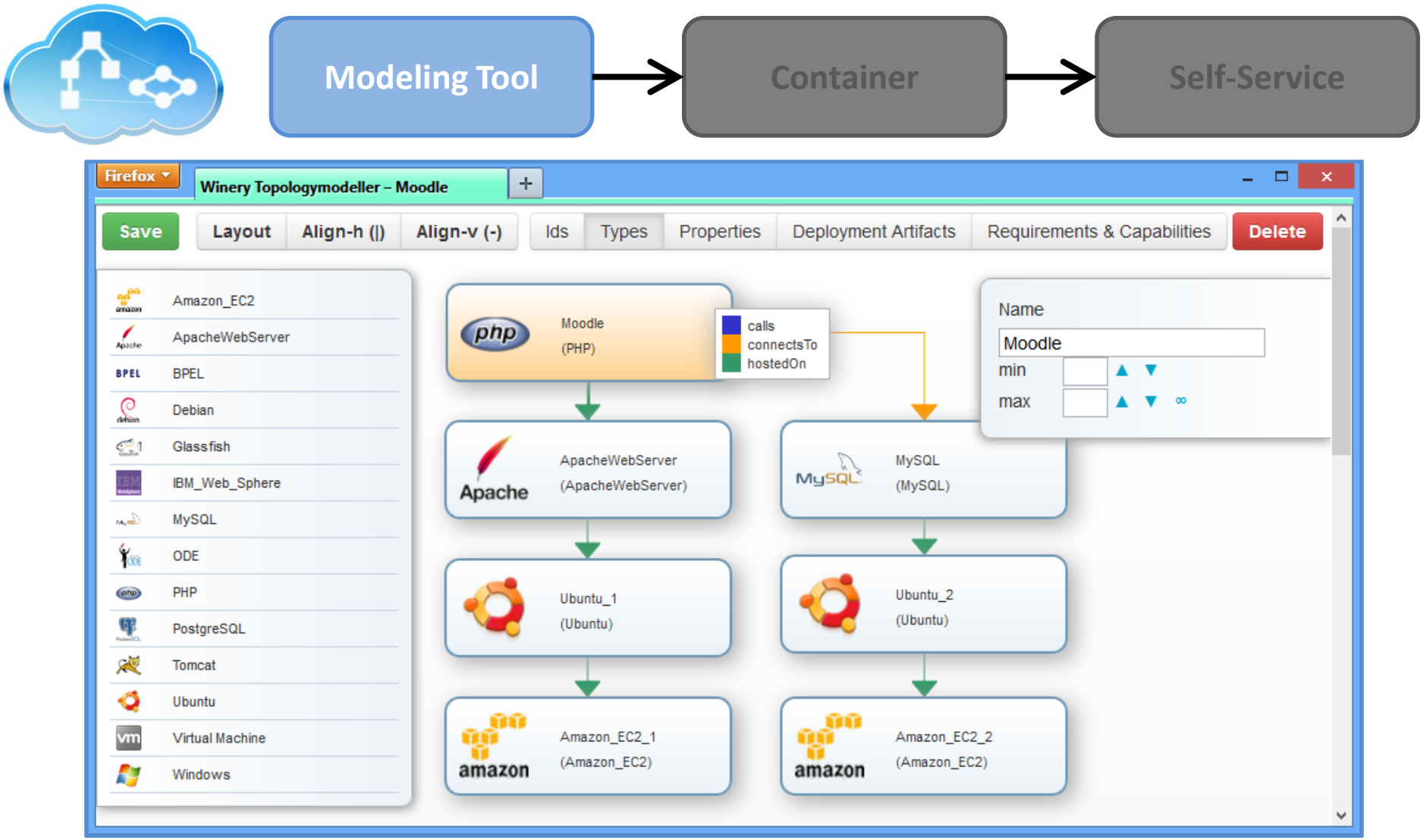
Known Major Issues (as of March 11, 2014)

- OpenTOSCA Winery (TOSCA modeling tool)
  - No validation beyond XML schema validation
  - Browser related issues
- OpenTOSCA Container (TOSCA runtime)
  - Supports imperative CSAR processing only
  - The Partner Link role of the Build & Management Plans must be named “client”.
  - Restart of container requires redeployment of CSARs

Contents

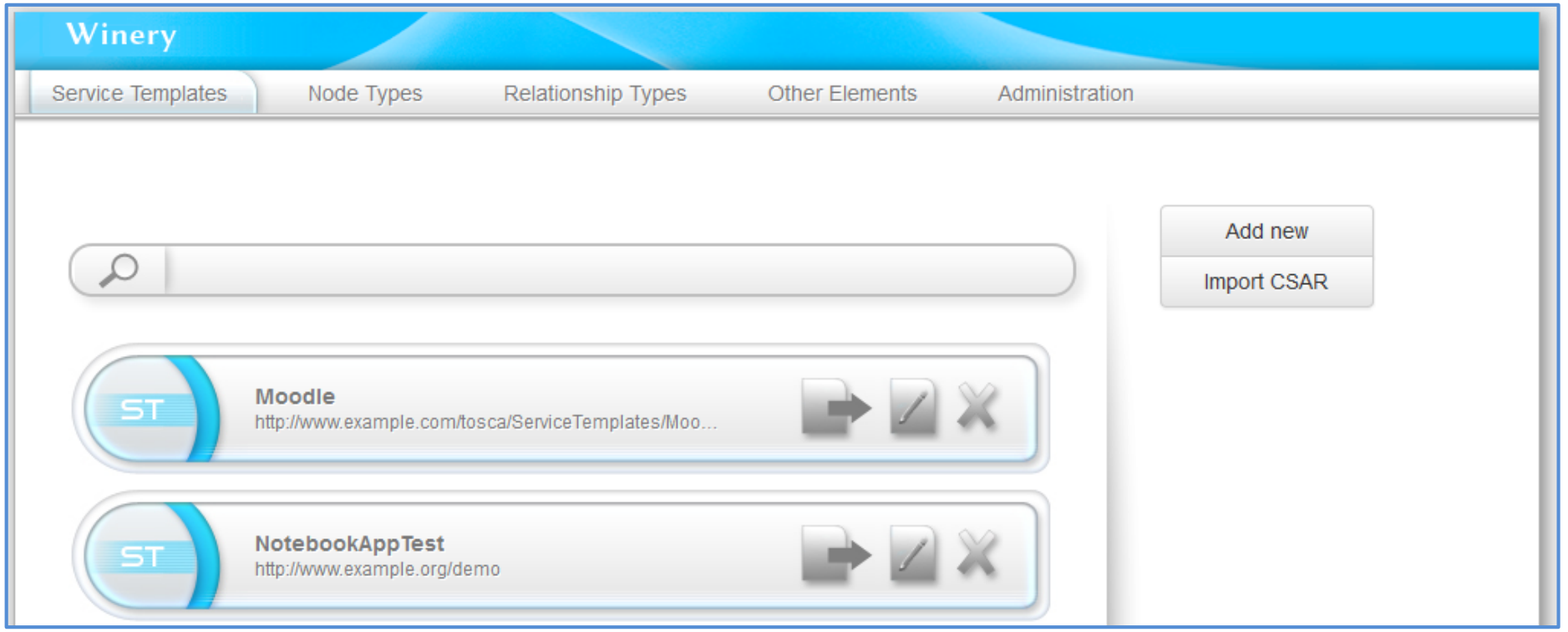


OpenTOSCA Ecosystem Overview



Creation and modeling of TOSCA applications, including graphical modeling of topologies and management plans.
Exported as Cloud Service Archive (CSAR) for TOSCA runtime.

Modeling Tool → Container → Self-Service



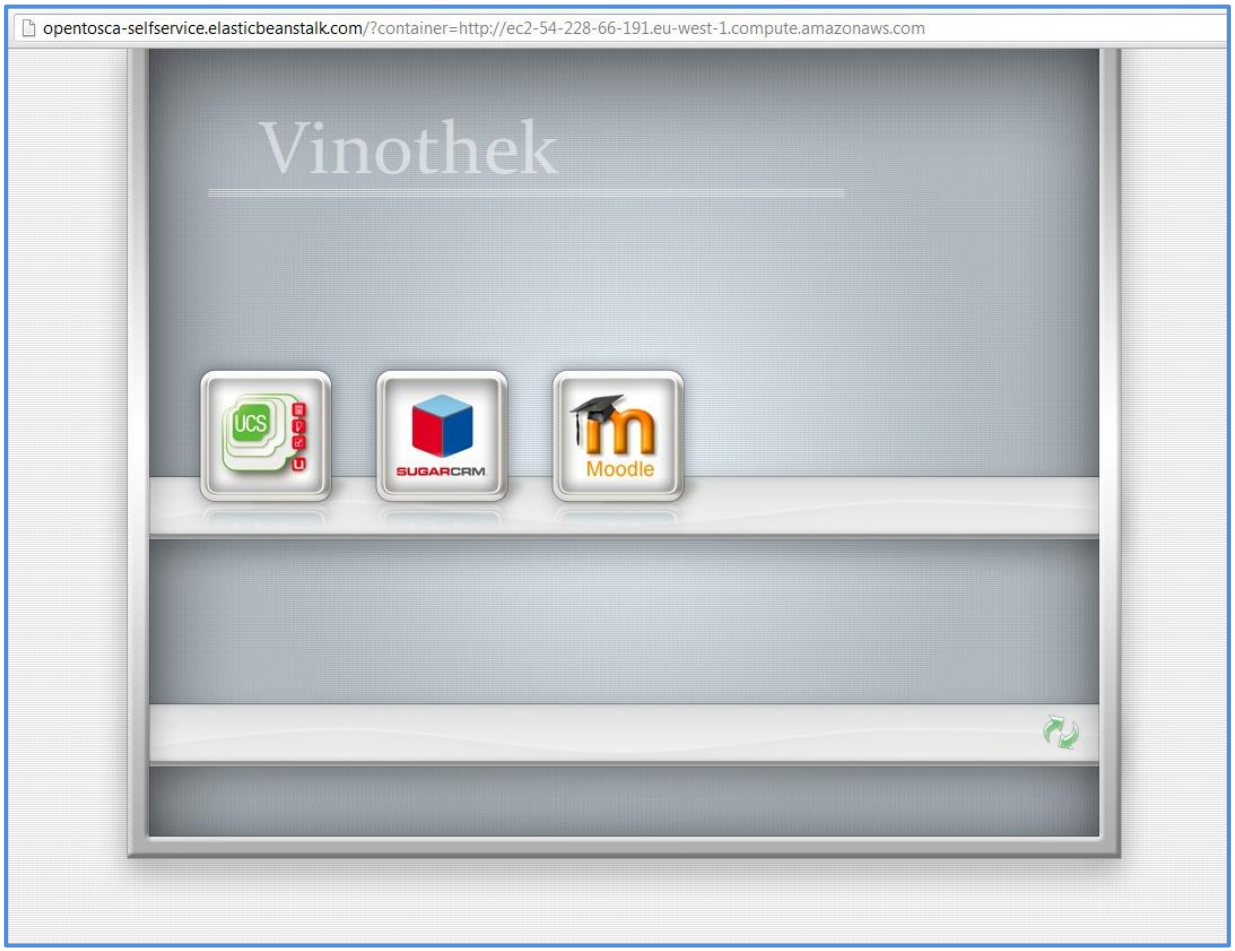
Creation and modeling of TOSCA applications, including graphical modeling of topologies and management plans.
Exported as Cloud Service Archive (CSAR) for TOSCA runtime.

Modeling Tool → Container / Runtime → Self-Service

TOSCA runtime & middleware

Processes CSARs, runs plans, manages state, …

Modeling Tool → Container → Self-Service



Offering the deployed CSARs to the end user for easy instantiation

Modeling Tool → Container → Self-Service

Offering the deployed CSARs to the end user for easy instantiation

Ecosystem: Structure and Relations

Admin

End User

Admin UI
/admin

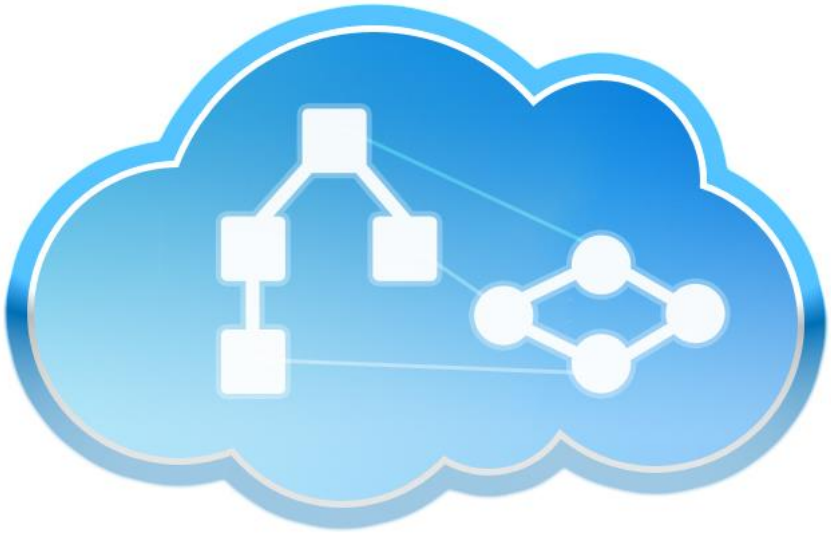
Self-Service UI
/vinothek

Use container API

Modeling Tool
/winery

CSAR

Container

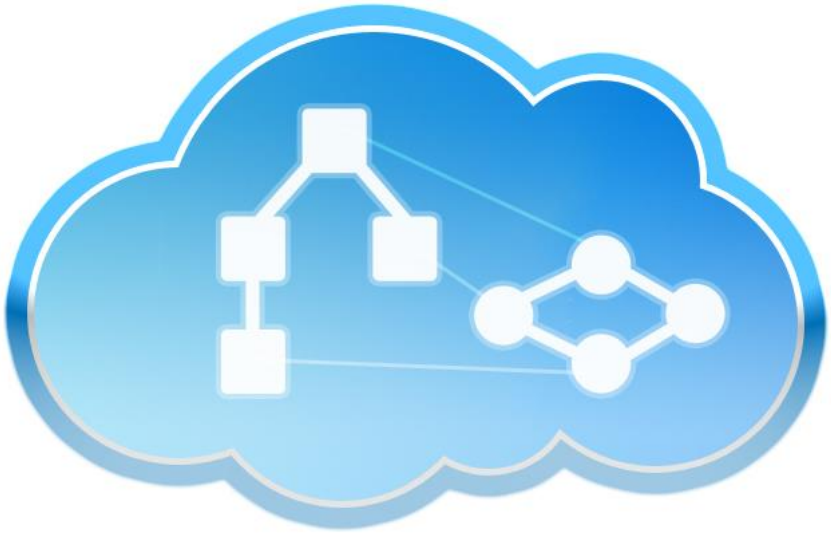


Online Usage

Winery Online

- **Winery online: http://winery.opentosca.org**
  - The repository will be reset each night!

- **The container/runtime is currently not provided online**



Automated Installation of Complete Ecosystem

Automated OpenTOSCA Ecosystem Installation

You can install the OpenTOSCA Ecosystem in three ways:

- (1) Shell script (tested for Ubuntu)
- (2) Amazon CloudFormation Template for Amazon EC2 (Recommend if you want to use Amazon EC2)
- (3) Shell script for Amazon EC2
- (4) Shell script for OpenStack

Your provider or operating system is missing?

- For other IaaS providers or Linux flavors try (1) and adapt script accordingly, if needed.
- For Windows or Mac please use the manual installation.

(1) Shell Script

- Just run "`curl install.opentosca.de/|sh`" on your shell as user. The script contains `sudo` commands.
- After the installation and start up completed (~10min): Open the URL http://<YOUR-HOST>:8080/ in your browser
- If OpenTOSCA runs (in its default configuration) on a virtual machine, you need to configure the firewall so at least ports 22 (SSH), 1337 (OSGi running OpenTOSCA container), 8080 (Tomcat), 9443 and 9763 (Business Process Server) are open!

Note: This script does not work for Amazon EC2 (and maybe other public cloud providers because of the way they assign private and public DNS names).

For Amazon EC2 please use this shell script.

For OpenStack please use this shell script.

(2) Amazon CloudFormation

- **Open** https://console.aws.amazon.com/cloudformation
- **Create** a new stack in region of your choice
- **Select** “Upload a Template File” and upload this template:
  - http://install.opentosca.de/cloudformation.template
- **Input**
  - KeyName
    - Name of the EC2 Key Pair to access the created instance.
    - Creation of key name is described on the next slide.
  - InstanceType
    - Default is *m1.medium*
    - Smaller instance types don’t work! (not enough memory)
- **Create Stack**

(2) Amazon CloudFormation: Key Creation

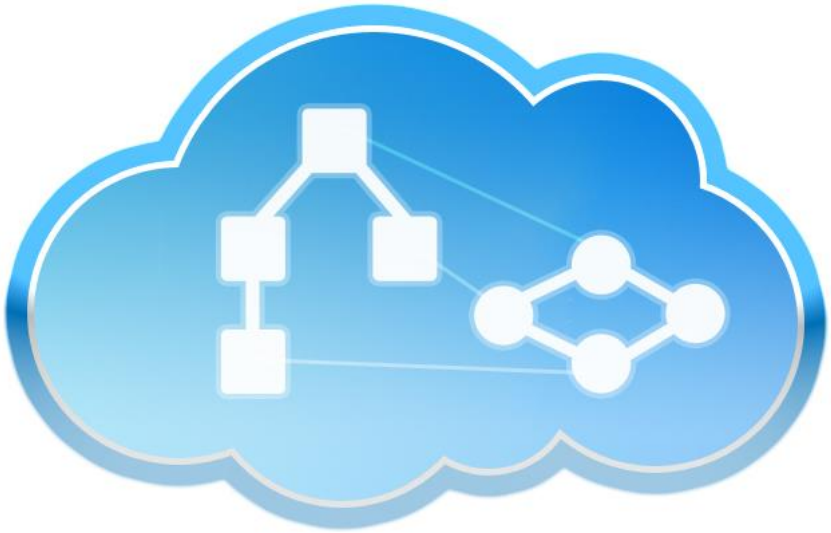
- **Open**: https://console.aws.amazon.com/ec2
- **Select** AWS region in top right
- **Click** “Key Pairs” in the menu on the bottom left
- **Click** Create Key Pair
- **Provide** a KeyName
- **Store** key to your local machine
  - If you want to connect via SSH to the machine OpenTOSCA is installed on, you will need this key.

(3) Shell Script for EC2

- **Create** Security group (same region as the EC2 instance!) with at least TCP ports 22, 1337, 8080, 9443 and 9763 open
- **Create** Key, as described here.
- **Create** EC2 instance
  - **Key:** Select key created before
  - **AMI**: Ubuntu Server 12.04.2 LTS 64bit
    AMI id for your region: http://cloud-images.ubuntu.com/locator/ec2/
  - **Size**: m1.medium or larger
  - **Security Group:** Select security group created before
  - **Connect** to the instance using SSH
  - **Run** "`curl install.opentosca.de/installEC2|sh`"
  - **Wait** for ~10 min
- **Open** the URL http://<publicDNS>:8080/ in your browser

(3) Shell Script for OpenStack

- **Create** Security group with at least TCP ports 22, 1337, 8080, 9443 and 9763 open
- **Create** Keypair
- **Launch** Instance
  - **Flavor:** m1.medium or larger
  - **Instance Boot Source**: Boot from image
  - **Image Name:** Ubuntu Server 12.04 or 13.04
  - **Access & Security Tab:** Select keypair and security group created before
  - **Assign** floating IP to instance
  - **Connect** to the instance using SSH
  - **Run** "`curl install.opentosca.de/installOpenStack|sh`"
  - **Wait** for ~10 min
- **Open** the URL http://<publicDNS>:8080/ in your browser



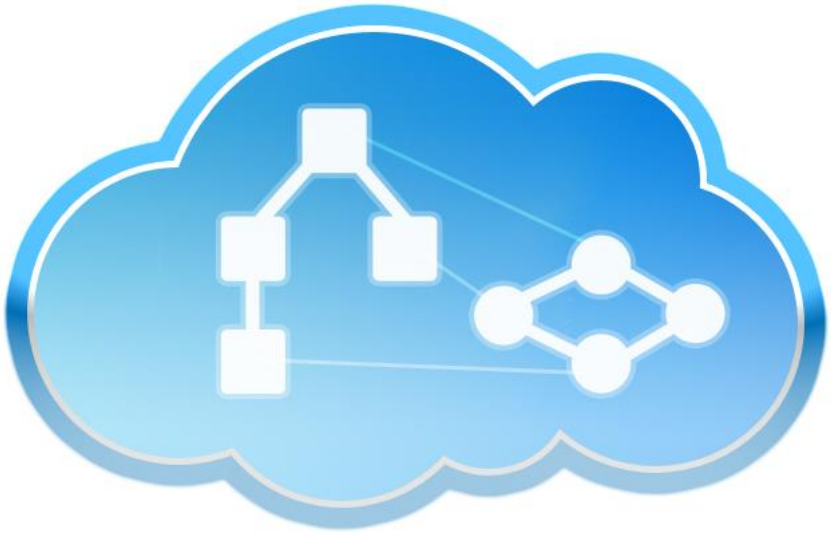
Manual Installation

Winery: Manual installation on premise

- **Download** Winery WAR files:
  - http://files.opentosca.de/v1.1/winery.war
  - http://files.opentosca.de/v1.1/winery-topologymodeller.war
- **Deploy** WAR-files to Tomcat (on Java7) by moving them into "webapps"
- **Open**: http://localhost:8080/winery/
  - Assumes that your Tomcat runs on port 8080.
- **Import**: Example Repository (optional)
  - "Administration" -> "Repository" -> "Import Repository"

Container: Manual Installation

- Files of this release: http://files.opentosca.de/v1.1/
- Install Java 1.7 and Tomcat 7.x
- Replace *tomcat-users.xml*
- Deploy WARs (ROOT.war, admin.war, vinothek.war), i.e., copy into Tomcat webapps folder
- Unzip *OpenTOSCA.zip* and *wso2bps-2.1.2.zip* (rename to *wso2bps*)
- Install BPEL4Rest extension on BPS
  - Copy *bpel4restlight1.1.jar* into folder *wso2bps/repository/components/lib*
  - Replace *wso2bps/repository/conf/bps.xml*
- Start Tomcat (depends on how you installed it) and wait
- Start WSO2BPS (*wso2bps/bin/wso2server.sh* or *.bat*) and wait
- Start OpenTOSCA (*OpenTOSCA/startup.sh* or *.bat*) and wait
- Open: http://<HOST>:8080/



OpenTOSCA Container

How to use Moodle.csar

Moodle Example

- Moodle.csar – Example contained in release package
  - Moodle is an open-source course/school/learning management system based on an LAMP-stack on Amazon EC2
  - http://files.opentosca.de/v1.1/CSARs/Moodle.csar
- There are two ways to prepare and run Moodle.csar:
  - **(1) Moodle + Vinothek**, i.e., the self service way
    - Adaptation & CSAR deployment is done by admin role (AdminUI)
    - Instantiation is done by end user in self-service portal (Vinothek)
  - **(2) Moodle + soapUI**
    - CSAR deployment & instantiation (done by admin role)
- Troubleshooting tips if something does not work

Required Information for (1) and (2)

To run Moodle.csar on Amazon EC2 you need the following information from your AWS Account:

- Region to run (e.g., “ec2.eu-west-1.amazonaws.com”)
- Ubuntu Linux AMI available in this region
  - Tested with Ubuntu Server 12.04.2 LTS 64bit
  - Find AMI for your region (e.g., “ami-ce7b6fba” for eu-west-1): http://cloud-images.ubuntu.com/locator/ec2/
- Your account’s access and secret key
- Security group (same region!): All TCP ports open
- Key name and certificate of a key pair in the respective region

(1) Moodle + Vinothek: Preparations

Instantiate Moodle using Vinothek self-service UI:

- (1.1) Vinothek requires the AWS information in the predefined build plan input message
  - Therefore, adapt the plan input messages accordingly by adding your AWS information to the build plan messages *Moodle.csar/SELFSERVICE-Metadata/plan.input.default.xml*
  - The CSAR is a ZIP archive which can be extracted and edited using standard ZIP tools. To adapt Moodle.car unzip it, modify the respective files, and ZIP it again. Check that the folder structure is conserved.

(1.2) Moodle + Vinothek

- (1.2) Deploy Moodle.csar to container
  - Go to http://<CONTAINER-HOST>:8080/ and click on “Administrative UI”
  - “Upload new CSAR” (Button lower left)

(1.3) Moodle + Vinothek

- (1.3) Instantiate using Vinothek
  - Go to http://<CONTAINER-HOST>:8080/
    and click on “Vinothek Self-Service Portal”
  - Click on Moodle application listed there to open detail view
  - Click “Start Instance”
  - Approx. 10 min later:
    Click “play” button to open Moodle instance

Do not forget to shutdown the created instances on Amazon! ☺

(2) Moodle + soapUI

Instantiate Moodle by invoking the build plan manually with SOAPui:

- No Adaptation of Moodle.csar required
- (2.1) Deploy Moodle.csar to container
  - Go to http://<CONTAINER-HOST>:8080/ and click on “Administrative UI”
  - “Upload new CSAR” (Button lower left)

(2.2) Moodle + soapUI

- (2.2) Start build plan using soapUI
  - File → New soapUI Project
  - Initial WSDL: http://<CONTAINER-HOST>:9763/services/MoodleBuildPlanService?wsdl
  - Open: MoodleBuildPlanBinding → initiate → Request1
  - Fill in your data (using the template on the following slide)
  - Submit request by pressing the green play button

Do not forget to shutdown the created instances on Amazon! ☺

(2.2) Moodle + soapUI (SOAP Message template)

Replace everything in red font:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:org="http:///www.opentosca.org/examples/Moodle/BuildPlan">
   <soapenv:Header/>
   <soapenv:Body>
      <org:MoodleBuildPlanRequest>
         <org:region>AWS-REGION-NAME (e.g., "ec2.eu-west-1.amazonaws.com")</org:region>
         <org:securityGroup>AWS-SECURITY-GROUP-NAME (create one with all TCP ports open)</org:securityGroup>
         <org:keyPairName>AWS-KEY-PAIR-NAME</org:keyPairName>
         <org:sshKey>-----BEGIN RSA PRIVATE KEY-----
...
AWS-SSH-KEY
(Ensure that all line breaks are conserved and no white spaces at the beginning of the lines are added!)
...
abcdefghijklmnopqrstuvwxyz0123456789==
-----END RSA PRIVATE KEY-----</org:sshKey>
         <org:ami>AWS-AMI-ID (e.g., “ami-c7c0d6b3” for region EU Ireland; Choose *EBS-Backed 64-bit* AMI ID from
http://aws.amazon.com/amazon-linux-ami/ matching your *region*)</org:ami>
         <org:instanceType>t1.micro</org:instanceType>
         <org:accessKey>AWS-ACCESS-KEY</org:accessKey>
         <org:secretKey>AWS-SECRET-KEY</org:secretKey>
         <org:csarName>Moodle.csar</org:csarName>
         <org:containerApi>http://localhost:1337/containerapi</org:containerApi>
         <org:callbackUrl>http://example.org</org:callbackUrl>
         <org:CorrelationID>randomId123</org:CorrelationID>
      </org:MoodleBuildPlanRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

(2.3) Moodle + soapUI

- (2.3) Receiving the result of the build plan is a little bit tricky ☺
  - The build plan is *asynchronous*, i.e., the request returns immediately and the build plan uses the callback URL in the plan input message to return its results.
  - To receive the reply create a soapUI MockService
    - Right click “MoodleBuildPlanCallbackBinding”
    - Select “Generate MockService”
    - Click ok, the window will print all responses it receives
  - Pass the MockService URL as callbackURL in the build plan input message
  - **However**, if you have the container running on Amazon and soapUI on a machine not accessible from the web, the build plan will not be able to send you the reply!
  - **However**, it is still possible to see the deployed application
    - After the build plan finished (~10 min) go to the Amazon EC2 console and copy the public DNS address of the second t1.micro instance created by the build plan
    - Open: http://<publicDNS>/moodle to see the deployed Moodle
  - All the heavy lifting done here is wrapped by the Vinothek, so we recommend using the Vinothek when not debugging a specific problem with your build plan!

Troubleshooting Tipps

- Is the management plan deployed? (see BPS web console linked from the OpenTOSCA root page)
- After starting an plan instance from Vinothek, is there a process instance in WSO2 BPS?
  - If yes, what's the status of the process instance, did it fail?
  - If yes, check BPS log for more details. You can find it somewhere here (not sure about the exact path): /wso2bps/repository/log/...
- Does the keyPairName you're using exist in the EC2 region you're using?
- Is an Amazon EC2 instance created, i.e., listed in your AWS Console?
  - If yes, are you able to connect onto this machine using SSH (user "ec2-user") and the credentials you provide in the plan.input.default.xml?
  - If no, check Tomcat Log (/var/log/tomcat7/catalina.out) for error messages of the Amazon Implementation Artifact trying to launch an EC2 instance. If you AWS credentials or configuration is wrong, you'll see this in the Tomcat log.
- Have you been able to start the build plan using SOAPui?
- Logs you may want to inspect for debugging purposes:
  - Logs for Vinothek and Implementation Artifacts:
    `tail -f /var/log/tomcat7/catalina.out`
  - Logs for container and BPEL engine:
    `tail -f /OpenTOSCA/nohub.log`