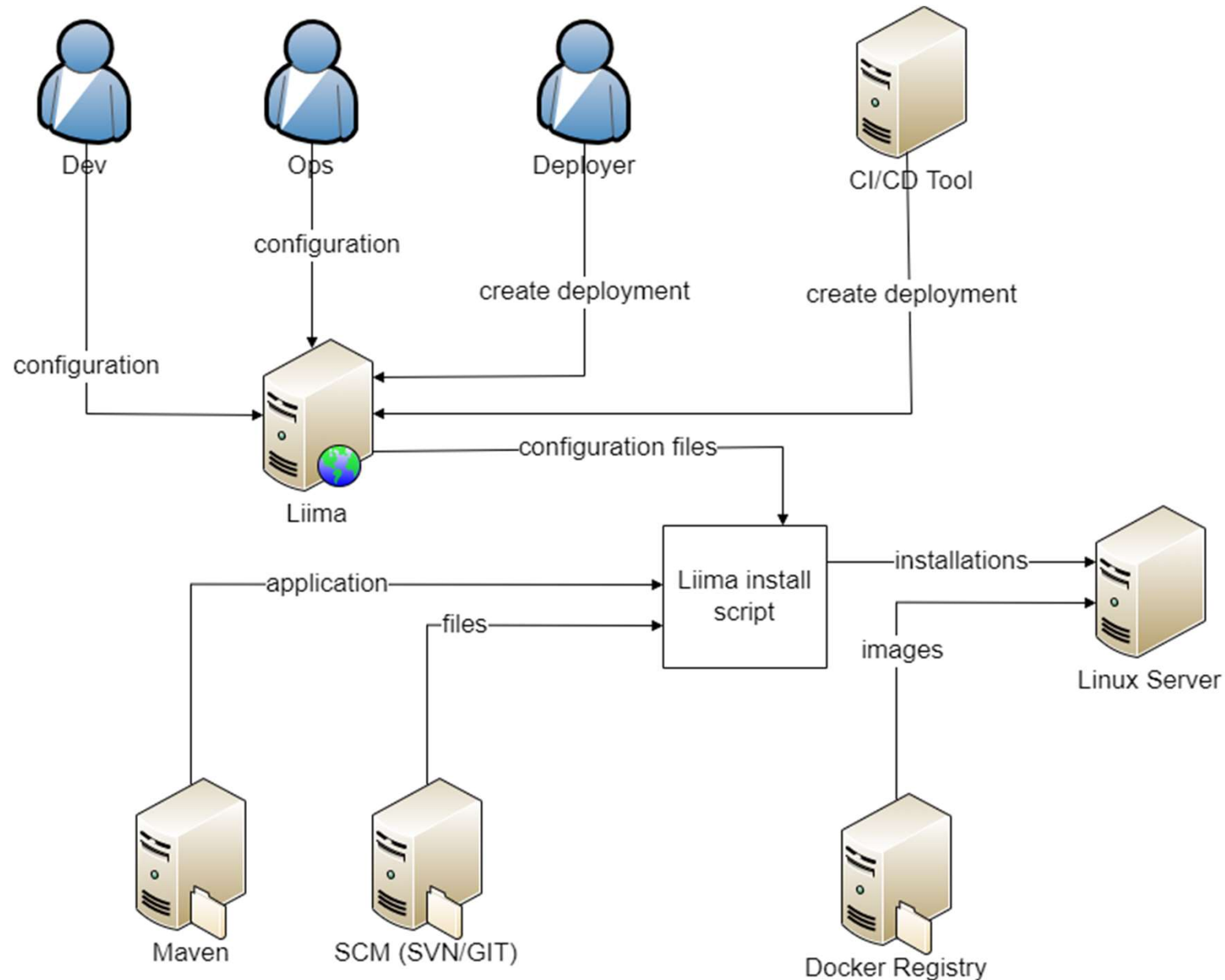


Liima Features

What is Liima? What are its advantages?

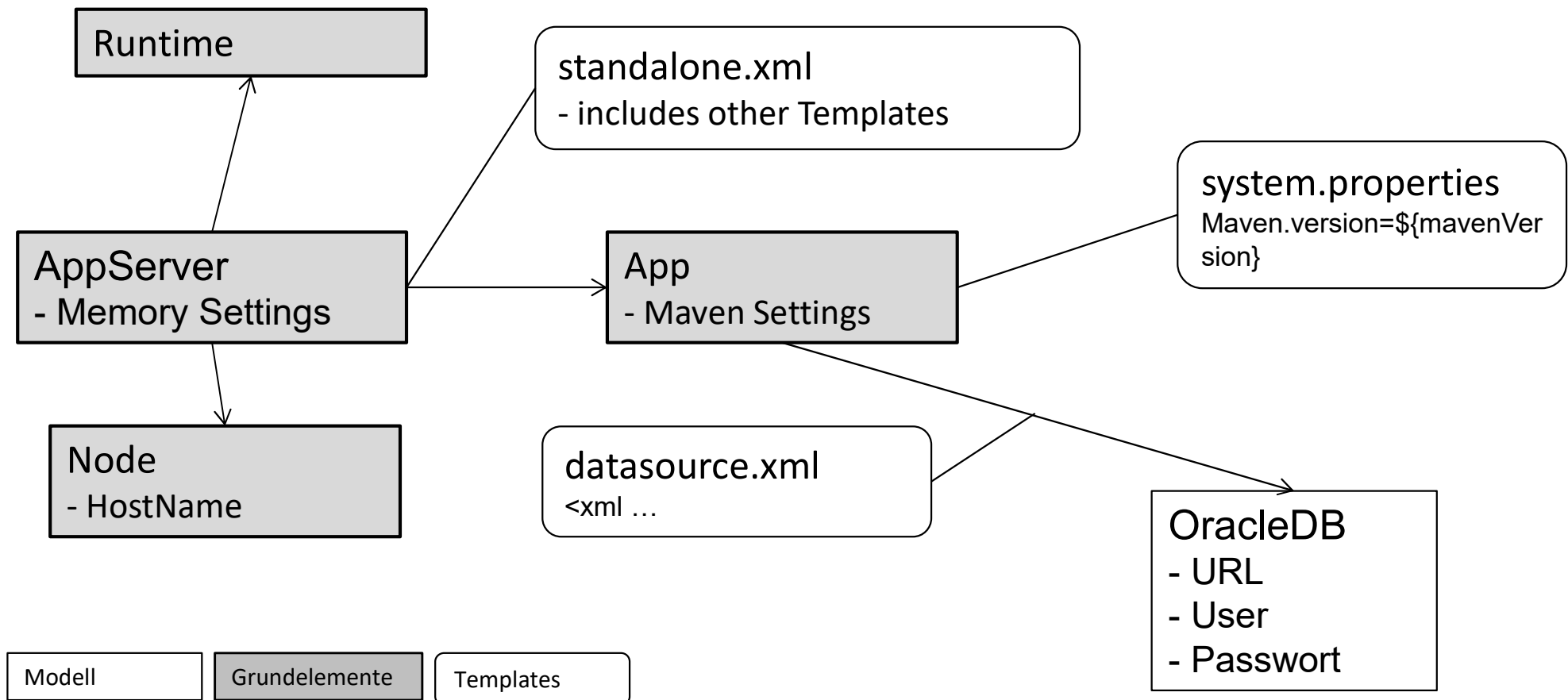
- **Liima** (Finnish for glue) manages the configuration and deployment of applications. It's optimized for configuration reuse.
- Main advantages of Liima
 - Configuration doesn't depend on the technology of the application
 - Avoids duplication of configuration
 - Versioning and audit
 - The configuration is installed with the application as one package
 - Configuration is readable by different teams, secured by role based access control
 - Easy to integrate with other tools and technologies

Deployment Prozess mit Liima



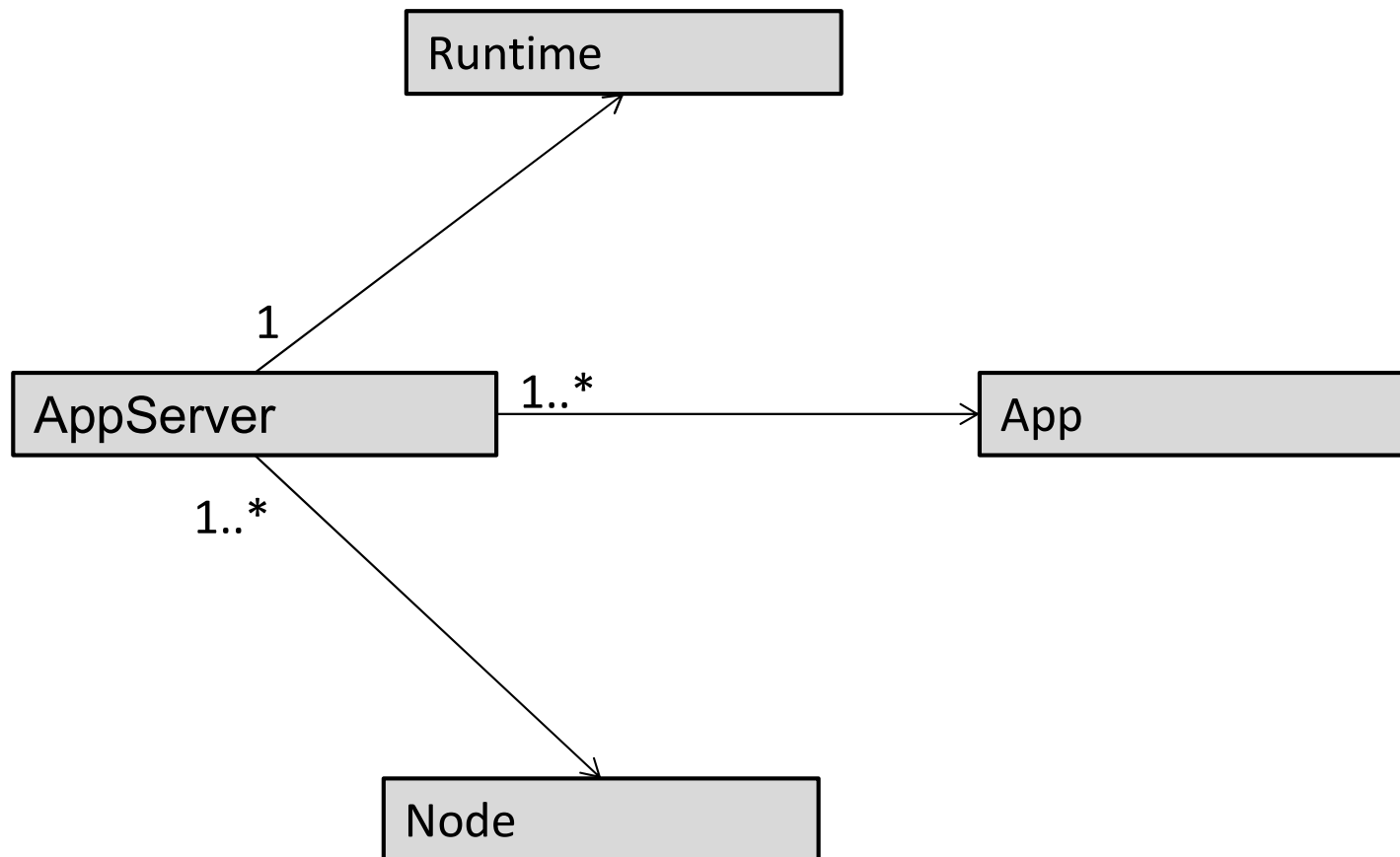
Configuration model

- Freely definable object model with Liima resources
- Modelling of applications with their relations
- Freemarker templates convert the model into configuration files



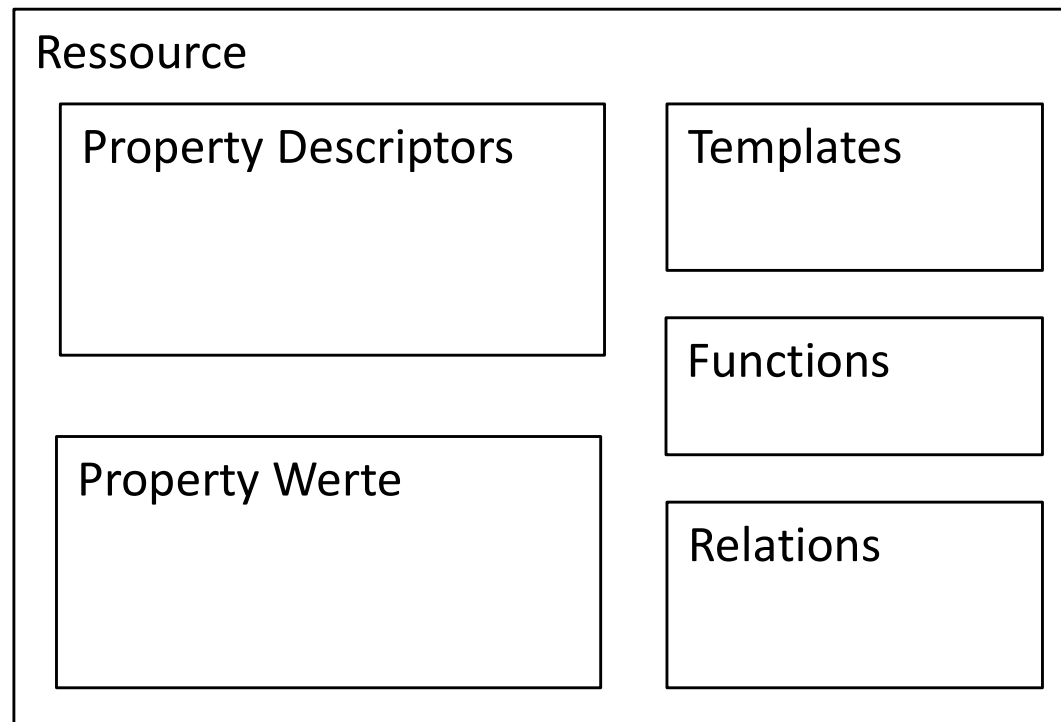
Modell: default resources

- Default resources are required for each deployment



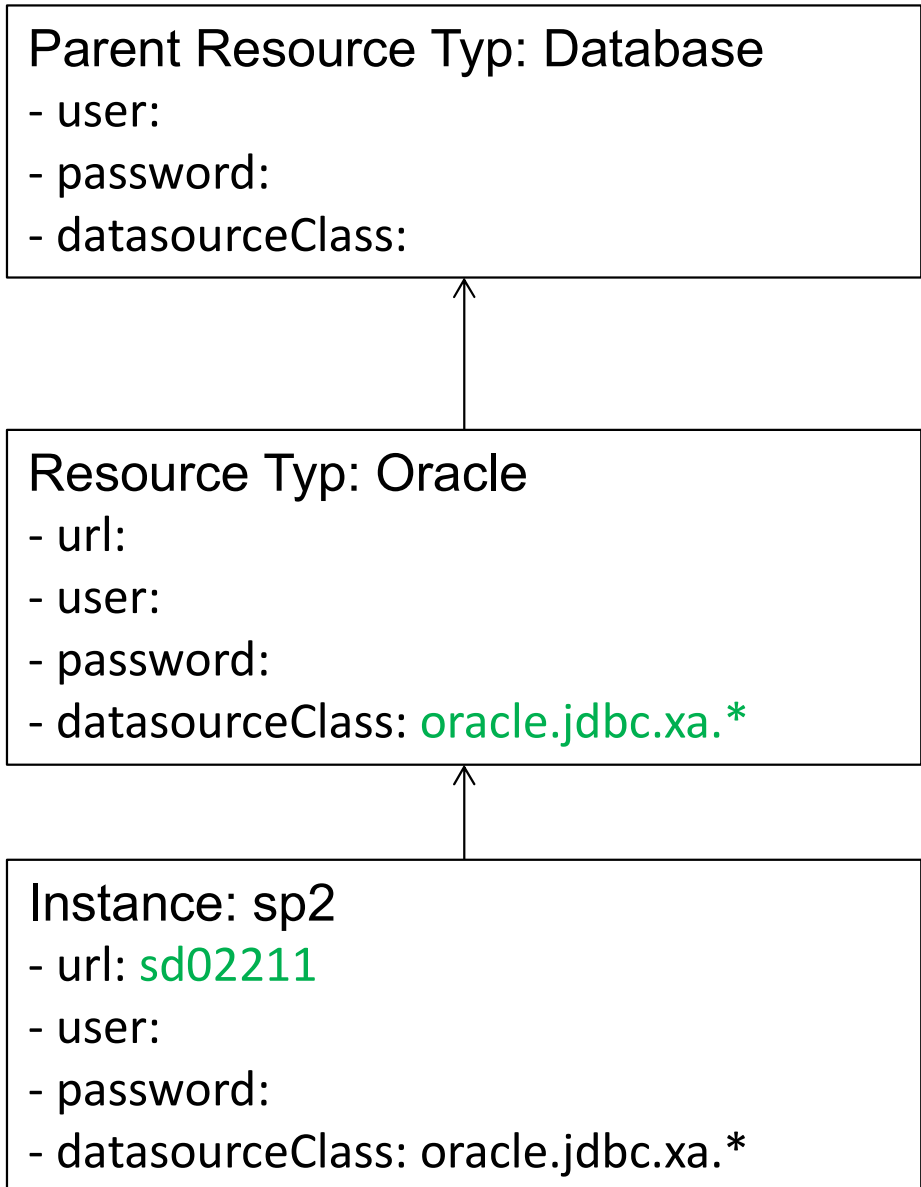
Model: resources

- Resources contain:
 - Properties descriptors: definition of a property
 - Property values: value of a property
 - Templates: convert the properties into configuration
 - Functions: composition of properties
 - Relations: reference to other resources for reuse



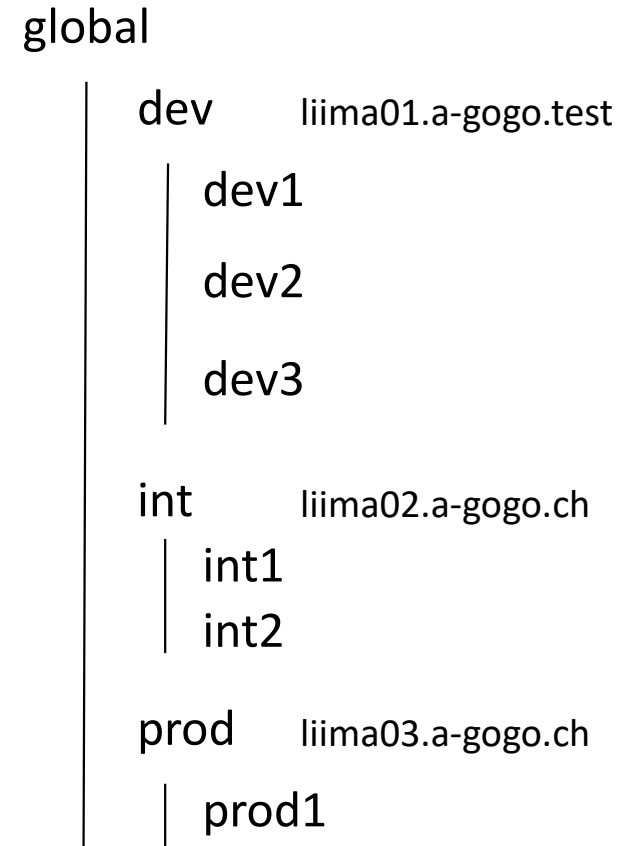
Model: inheritance and properties

- Every resource has a type
- A type can have one parent type
- Resources of a type are called instances
- The following will be inherited:
 - property descriptors
 - property values
 - templates
 - functions
- **Green**: defined on that resource
- **Black**: defined on a parent resource type



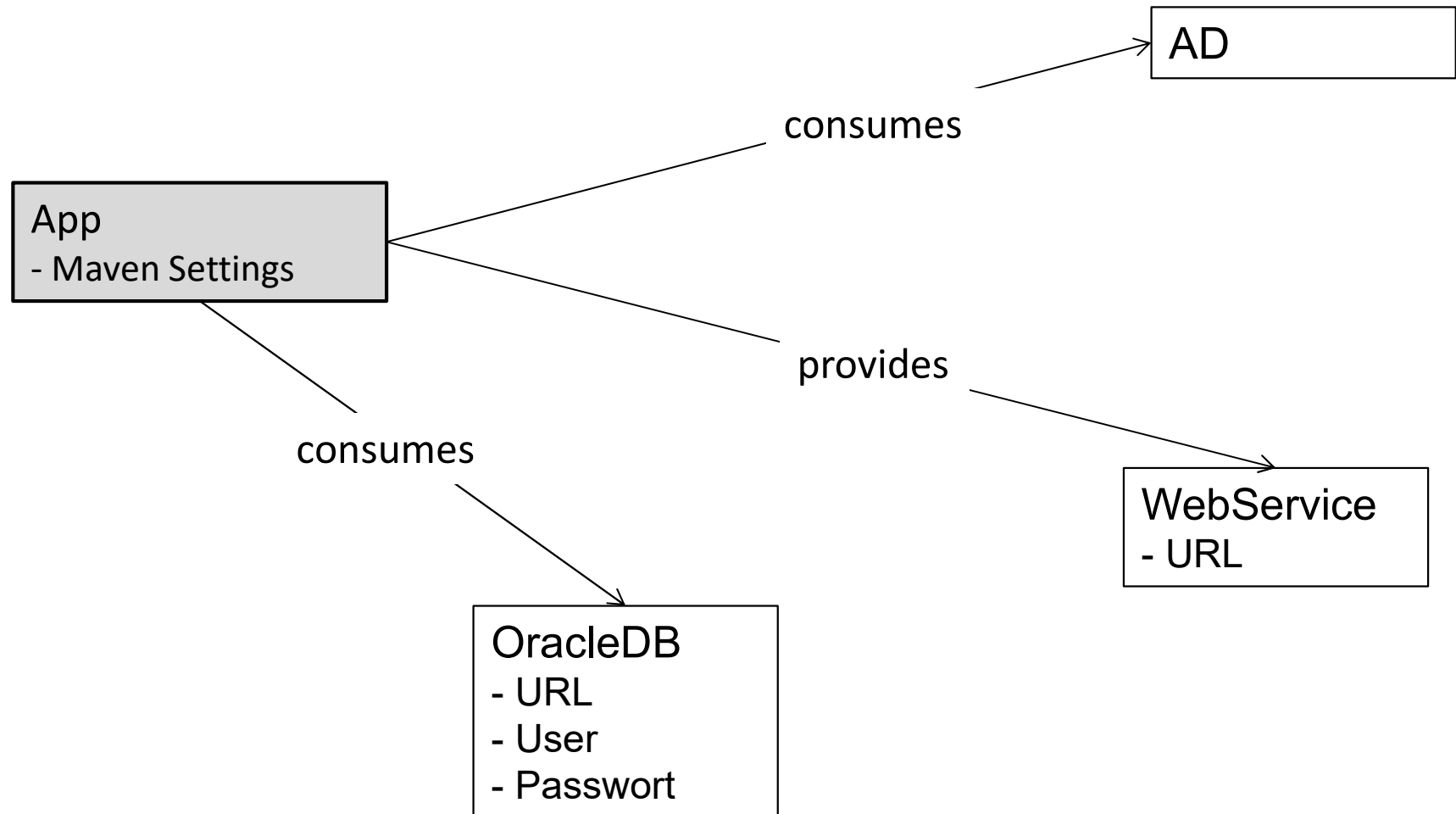
Model: environment hierarchy

- Property values are inherited from global to domain (dev, int, prod) to environment
- Enables defaulting
- Avoids redundancies
- On the Liima UI:
 - **Green**: defined on this level
 - Black: inherited from a higher level
 - **Red**: validation error
 - The tooltip (i) shows where a inherited property was overwritten



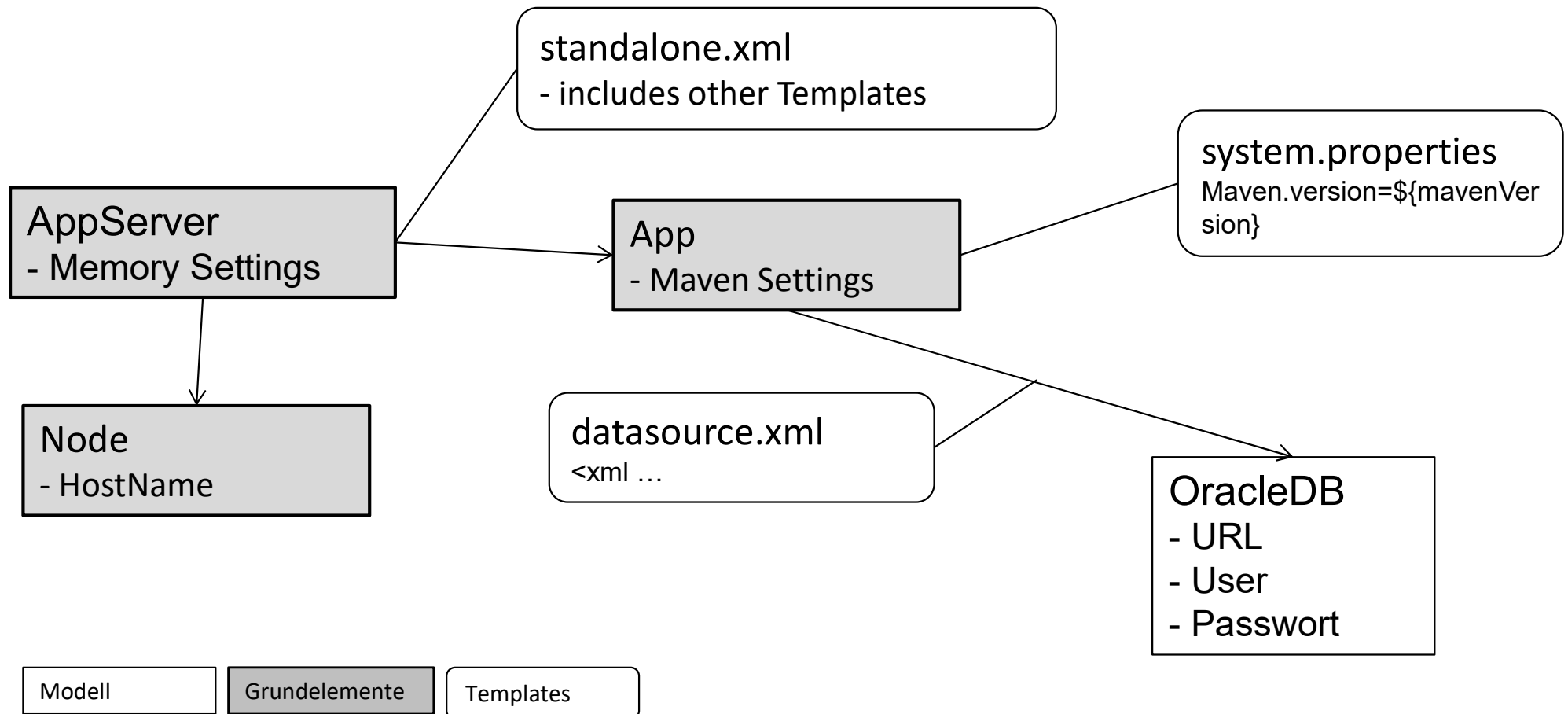
Model: relations

- Consumed: a foreign resource is consumed
- Provided: a resource is provided
- Templates can access related resources
- Property values can be overwritten in relations



Model: templates

- Templates are use to write configuration
- Uses Freemarker: <http://freemarker.org/docs/>
- Templates can be attached to resources, resource types and relations
- Templates are always connected to a runtime

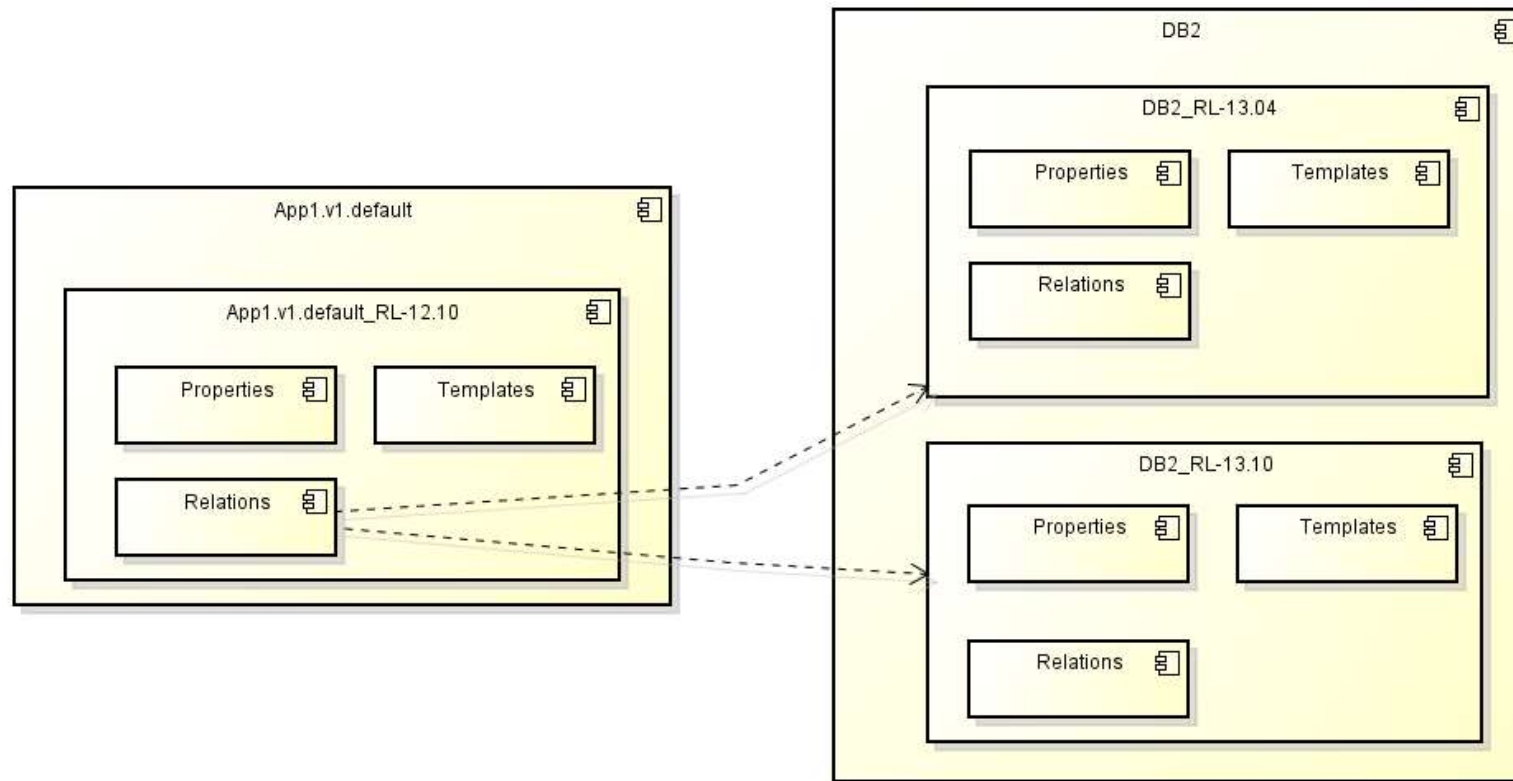


Releasing: motivation

- A Liima configuration is always valid for all environments
- Releasing enables to have different configurations on different environment
- Example:
 - RL-15.04 was released recently, next release is RL-15.10
 - An application need a different database in RL-15.04
 - The existing config can't be modified because it would break productions deployments which still need the old database
- Workarounds:
 - Before each deployment change the configuration by hand
 - Templates with `#if` and `#else`
 - Different Liima configuration per environnement

Releasing

- A release is a copy of a resource: properties, templates, relations etc.
- Different release are shown in Liima as one element
- Corresponds to releases of the application
- Is similar to a branch in source management
- Relations of consumers are extended automatically
- At deploy time the matching relation will be selected

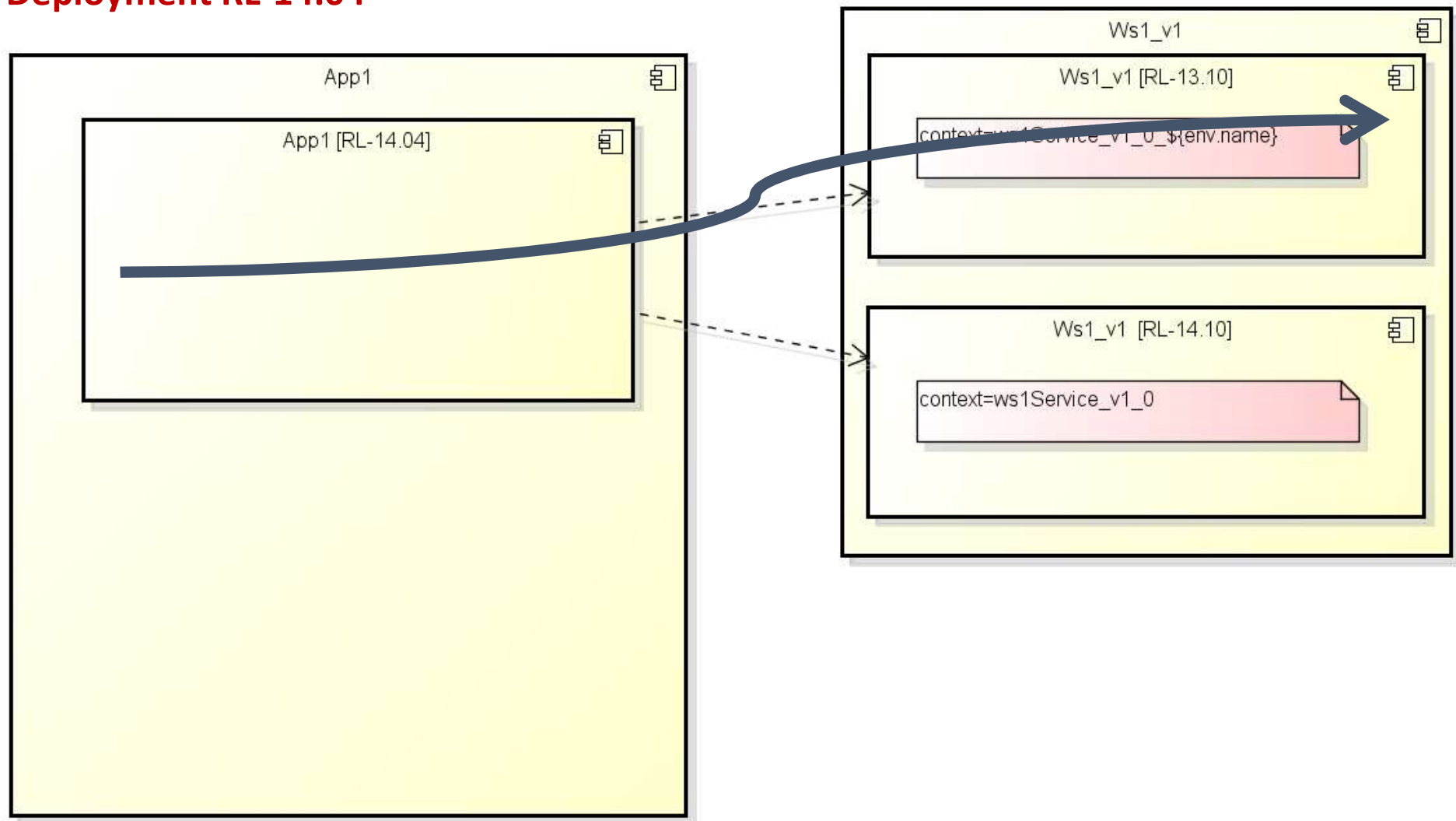


Releasing: deployment

- The release dropdown in the deployment UI determines which resource releases are selected
- The releases in the dropdown depend on the releases of the app server
- Applications can be deployed with existing releases of an app server or higher

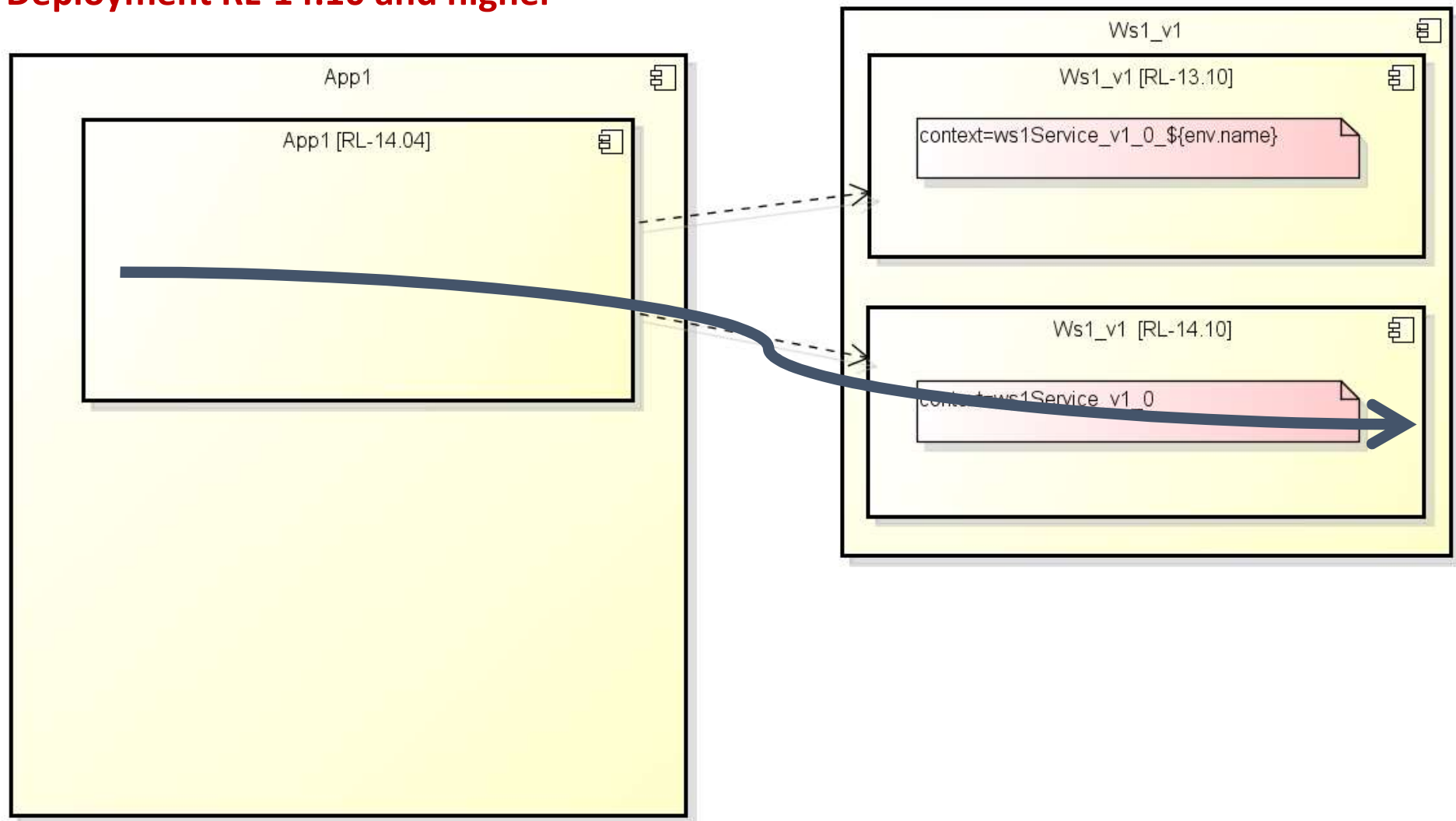
Model: releases

Deployment RL-14.04



Model: releases

Deployment RL-14.10 and higher

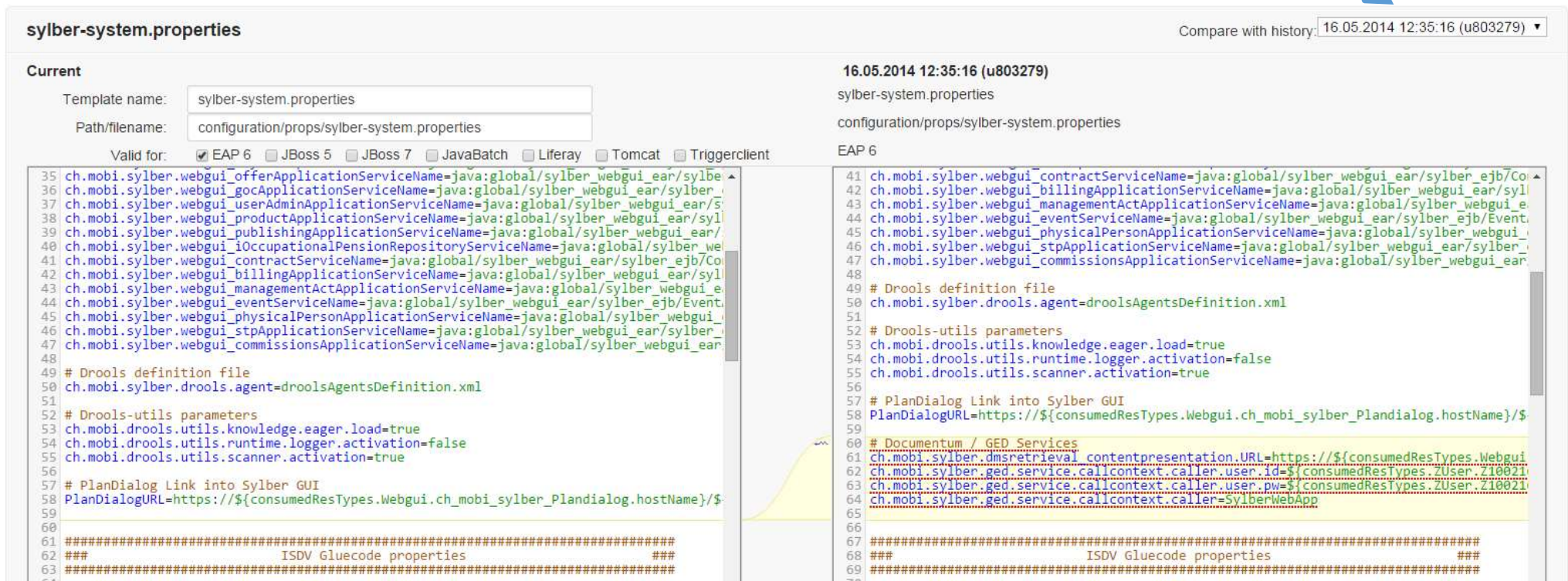


Template editor

- JavaScript based template editor ([CodeMirror](#))
 - Built in search function with highlighting
 - Freemarker syntax highlighting
 - Row numbers
 - Fullscreen mode
 - Most import key shortcuts blow the editor window
- Freemarker syntax is check when a template is saved
 - Validates unclosed bracket and tags
 - Can't check if variables are missing! Use test generate for this

Template editor: compareision

- A template can be compared with older versions
 - Editor shows differences
 - Supports merging
 - Time and user of changes are displayed



The screenshot displays the 'sylber-system.properties' template editor. At the top right, a dropdown menu labeled 'Compare with history:' shows the selected version '16.05.2014 12:35:16 (u803279)'. A blue arrow points to this dropdown. The editor is split into two panes. The left pane, titled 'Current', shows the template name 'sylber-system.properties' and its path 'configuration/props/sylber-system.properties'. It also lists valid frameworks: EAP 6, JBoss 5, JBoss 7, JavaBatch, Liferay, Tomcat, and Triggerclient. The right pane, titled '16.05.2014 12:35:16 (u803279)', shows the same template name and path, but with the framework 'EAP 6' selected. Both panes display the template content, which includes various service names and Drools definitions. The content is color-coded to show differences between the two versions.

```
Current
Template name: sylber-system.properties
Path/filename: configuration/props/sylber-system.properties
Valid for: ☒ EAP 6 ☐ JBoss 5 ☐ JBoss 7 ☐ JavaBatch ☐ Liferay ☐ Tomcat ☐ Triggerclient

35 ch.mobi.sylber.webgui_offerApplicationServiceName=java:global/sylber_webgui_ear/sylber_
36 ch.mobi.sylber.webgui_gocApplicationServiceName=java:global/sylber_webgui_ear/sylber_
37 ch.mobi.sylber.webgui_userAdminApplicationServiceName=java:global/sylber_webgui_ear/syl
38 ch.mobi.sylber.webgui_productApplicationServiceName=java:global/sylber_webgui_ear/syl
39 ch.mobi.sylber.webgui_publishingApplicationServiceName=java:global/sylber_webgui_ear/
40 ch.mobi.sylber.webgui_iOccupationalPensionRepositoryServiceName=java:global/sylber_wei
41 ch.mobi.sylber.webgui_contractServiceName=java:global/sylber_webgui_ear/sylber_ejb/Co
42 ch.mobi.sylber.webgui_billingApplicationServiceName=java:global/sylber_webgui_ear/syl
43 ch.mobi.sylber.webgui_managementActApplicationServiceName=java:global/sylber_webgui_e
44 ch.mobi.sylber.webgui_eventServiceName=java:global/sylber_webgui_ear/sylber_ejb/Event
45 ch.mobi.sylber.webgui_physicalPersonApplicationServiceName=java:global/sylber_webgui_
46 ch.mobi.sylber.webgui_stpApplicationServiceName=java:global/sylber_webgui_ear/sylber_
47 ch.mobi.sylber.webgui_commissionsApplicationServiceName=java:global/sylber_webgui_ear
48
49 # Drools definition file
50 ch.mobi.sylber.drools.agent=droolsAgentsDefinition.xml
51
52 # Drools-utils parameters
53 ch.mobi.drools.utils.knowledge.eager.load=true
54 ch.mobi.drools.utils.runtime.logger.activation=false
55 ch.mobi.drools.utils.scanner.activation=true
56
57 # PlanDialog Link into Sylber GUI
58 PlanDialogURL=https://${consumedResTypes.Webgui.ch_mobi_sylber_Plandialog.hostName}/$
59
60
61 #####
62 ### ISDV Gluecode properties ###
63 #####
64
```

```
16.05.2014 12:35:16 (u803279)
sylber-system.properties
configuration/props/sylber-system.properties
EAP 6

41 ch.mobi.sylber.webgui_contractServiceName=java:global/sylber_webgui_ear/sylber_ejb/Co
42 ch.mobi.sylber.webgui_billingApplicationServiceName=java:global/sylber_webgui_ear/syl
43 ch.mobi.sylber.webgui_managementActApplicationServiceName=java:global/sylber_webgui_e
44 ch.mobi.sylber.webgui_eventServiceName=java:global/sylber_webgui_ear/sylber_ejb/Event
45 ch.mobi.sylber.webgui_physicalPersonApplicationServiceName=java:global/sylber_webgui_
46 ch.mobi.sylber.webgui_stpApplicationServiceName=java:global/sylber_webgui_ear/sylber_
47 ch.mobi.sylber.webgui_commissionsApplicationServiceName=java:global/sylber_webgui_ear
48
49 # Drools definition file
50 ch.mobi.sylber.drools.agent=droolsAgentsDefinition.xml
51
52 # Drools-utils parameters
53 ch.mobi.drools.utils.knowledge.eager.load=true
54 ch.mobi.drools.utils.runtime.logger.activation=false
55 ch.mobi.drools.utils.scanner.activation=true
56
57 # PlanDialog Link into Sylber GUI
58 PlanDialogURL=https://${consumedResTypes.Webgui.ch_mobi_sylber_Plandialog.hostName}/$
59
60 # Documentum / GED Services
61 ch.mobi.sylber.ged.service.retrieve.contentpresentation.URL=https://${consumedResTypes.Webgui
62 ch.mobi.sylber.ged.service.callcontext.caller.user.id=${consumedResTypes.ZUser.Z10021
63 ch.mobi.sylber.ged.service.callcontext.caller.pw=${consumedResTypes.ZUser.Z10021
64 ch.mobi.sylber.ged.service.callcontext.caller=SylberWebApp
65
66
67 #####
68 ### ISDV Gluecode properties ###
69 #####
70
```

Test generate

- Function reachable via „Test Generation“ button on app or app server
- Show rendering errors in templates
- Shows the templates after rendering
 - Is only shown if the user has permissions to deploy to that environment.
- Releases can be compared to each other
 - Uses the diff function of the editor
 - Can also be used to compare configurations from specific dates

Test generate

Release Selektion

The screenshot displays a test generation interface. On the left is a vertical sidebar with environment categories: GLOBAL, DEV, B, C, LOCAL, N, S, TEAM (highlighted with a black arrow), U, V, W, X, Y, Z, INT, I, K, T, PRO, and P. The main area is titled 'testAs1' and shows a comparison between 'RL-14.10' and 'RL-15.10'. Below this, the 'Application Server testAs1 on node_01' section contains a 'Templates' table with 10 rows. The first three rows show identical templates with green checkmarks. The next three rows show differences, indicated by double-headed arrows and greyed-out cells. The last four rows show identical templates with green checkmarks. Below the templates is an 'Applications' section for 'testApp1' with three rows. The first two rows show identical applications with green checkmarks. The third row shows differences, with the first column containing 'test=jsplb01team.umobi.mobiacorp.test' and the second column containing 'test=ssz.umobi.mobiacorp.test'. Blue arrows point from the 'TEAM' sidebar item to the 'Applications' section, from the 'Release Selektion' header to the release dropdowns, from 'Templates validation' to the template rows, and from 'Template differences' to the application rows.

Templates	
✓ amwScript/AmwRunScript.sh	✓ amwScript/AmwRunScript.sh
✓ amwScript/arguments.json	✓ amwScript/arguments.json
✓ amwScript/jboss7_config.spec	✓ amwScript/jboss7_config.spec
✓ amwScript/modelFromAppServer.tmp	⇄ amwScript/modelFromAppServer.tmp
✓ amwScript/modelFromNode.tmp	⇄ amwScript/modelFromNode.tmp
✓ amw_conf/configuration/run.conf	✓ amw_conf/configuration/run.conf
✓ amw_conf/configuration/standalone.xml	✓ amw_conf/configuration/standalone.xml
✓ cmdb_export_testAs1_team_1.xml.tmp	⇄ cmdb_export_testAs1_team_1.xml.tmp

Applications	
✓ datasource_oracle.tmp	✓ datasource_oracle.tmp
✓ modelFromApp.tmp	⇄ modelFromApp.tmp
✓ test	⇄ test

test=jsplb01team.umobi.mobiacorp.test

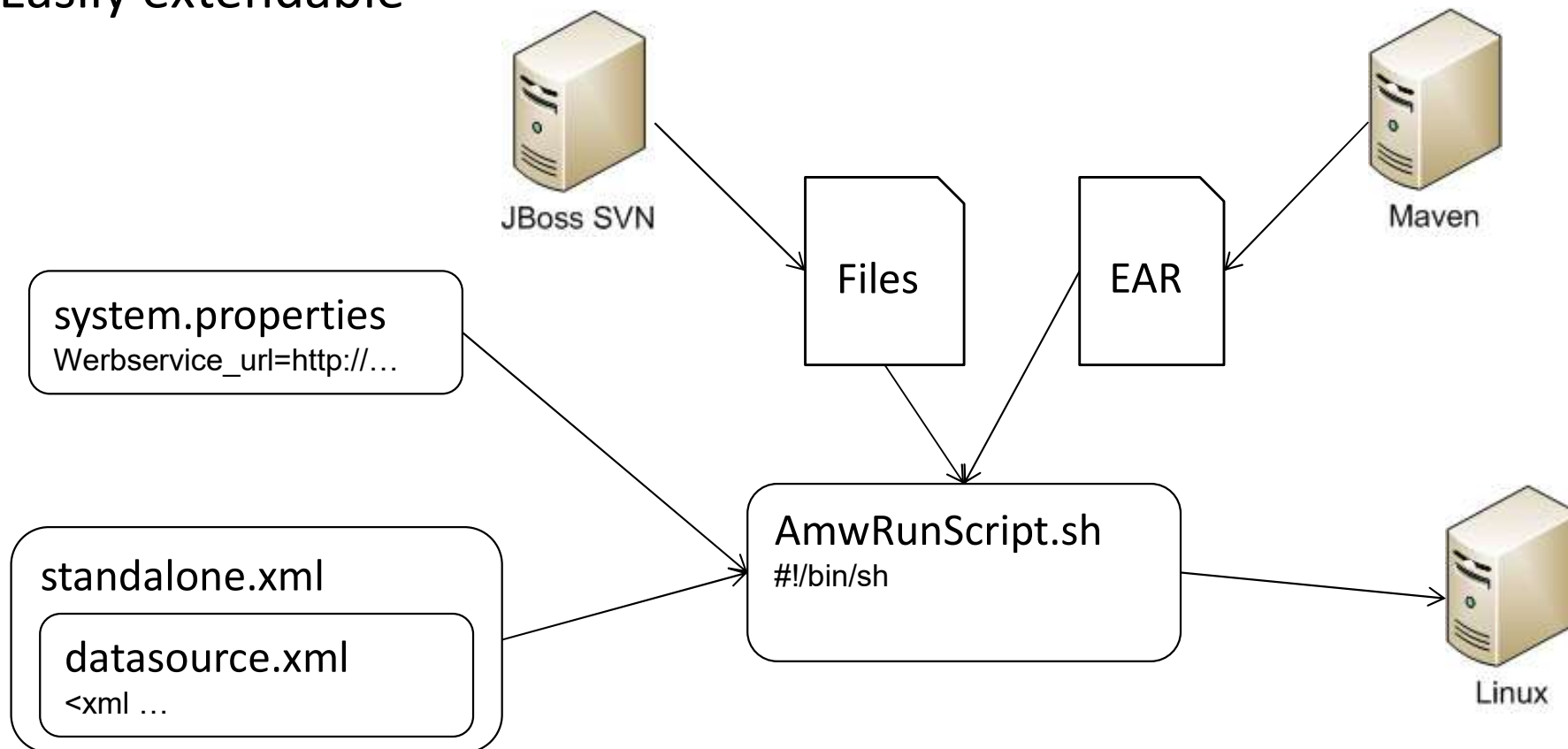
test=ssz.umobi.mobiacorp.test

Are there differences?

Template differences

Deployment

- Packaging of configuration and application
- Installation on the target system
- Easily extendable



Deployment filter

- Filter are connected automatically by AND or OR depending on the filter
- Deployment can be exported as CSV
- Filter "Latest deployment job for App Server and Env": shows only the latest deployment for one environment

Best practices

- Avoid redundancies
 - Enables central management
 - Better overview
 - Properties can contain other properties: `${env.name?lower_case}`
- Use resources instead of properties
 - Easier reuse
 - Dependencies are visible easier
- Fill out all environments from the beginning
 - Less prone to forget configuration
 - Less errors