



XebiaLabs training

XL Deploy

twitter

@xebialabs

www

www.xebialabs.com

github

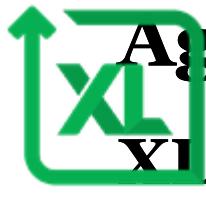
github.com/xebialabs-

community



Agenda

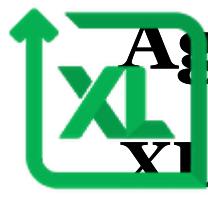
<http://www.xebialabs.com>



Agenda

XL Deploy Training

Chapter	Description
Introduction	Introduction to XebiaLabs and XL Deploy
Deployment Concepts	Introduction to the Unified Deployment Model
Installing XL Deploy	Setting-up XL Deploy server and CLI
The XL Deploy Library	Library concepts
Setting the stage	Populating the Library
Deployment	Performing initial deployments
Tasks and steps	Life-cycle of tasks and steps



Agenda

XL Deploy Training

Chapter	Description
Upgrading applications	Upgrade applications
Creating a package	Creating packages from GUI, exporting packages
Dictionaries and properties placeholders	Variables definition
Deployment with web servers	More on containers, comparison features, plan analyzer (not applicable for .Net)
Orchestrators	Configuring deployment plans



Introduction

<http://www.xebialabs.com>

XL Deploy - Introduction

XebiaLabs - The company

- www.xebialabs.com
- Headquartered in Burlington, Massachusetts
- Global development and support offices in US, NL, UK, France and India



XL Deploy - Introduction

XebiaLabs - Products



- XL Deploy: Model-based App Deployment
- XL Release: Orchestrate your Continuous Delivery pipelines
- XL Impact: Optimize software delivery with DevOps intelligence

XL Deploy - Introduction

XebiaLabs - Products



- Flagship product, XL Deploy, is a market-leading Application Release Automation platform.
- Benefits include:
 - Reduce development applications costs
 - Accelerate application time to market
 - Bridge the gap between Development and Operations

XL Deploy - Introduction

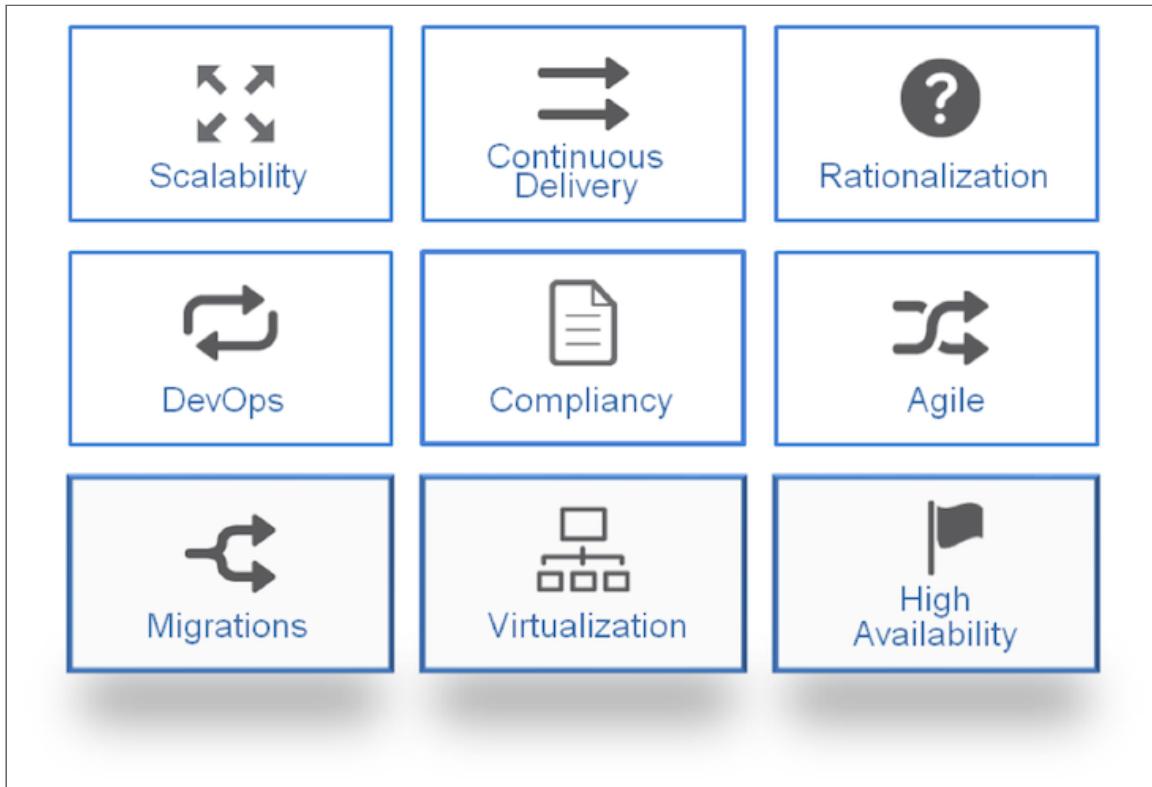
XL Deploy customers

Belfius	Scotiabank	RBC	Bank of America	if...	DHL	Smile	btcom	Thales	Siemens	Bull	SAP
GE INDUSTRIAL	VARTA	EDS	GIP	PYCHEX	Yota	ASCR	CELESTE	Gemeente Amsterdam	Siemens	dlt	CNV
Commonwealth	macy's	GATX	TRANSNEXUS	PPR	s2m	kadaster	ebsi	Met Office	Microsoft	WPS	SMARTFOCUS
Nexis	EMC ²	Liberty Mutual	FARRELL	NJM	EA	WalkerDigital	DigitalGlobe	STATE	SLIB	Finra	
	HUGO BOSS	Duke Energy	NUON	XEROX	ProSieben	topicus	Lincoln Financial Group	vantiv	TAKSISITE	GEOPOLIS	
3M	GE	Argosy	orange	TRIBUNE	W-HA	kfw	ROBECO	PGGM	BNK	DNB	GMF
Carrefour	+ T-Mobile	SAC	Expedia	AIR FRANCE KLM	KLM	CLSA	SOCIETE GENERALE	Rabobank	KLP	ABP	
CoachClub	Onweb	Perf & Beauty	IndyHealth	—	—	—	Smart Group	Huntington	HSI America		



XL Deploy - Introduction

ARA market trends



XL Deploy - Introduction

Key differences



Extensible, model-based



Lightweight & Cloud



Scalable

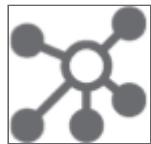


Insight & Compliance

XL Deploy Training

Key differences

Extensibility



Best Practice Extensibility

- Model-based
- Plugins for all major Java EE & .NET middleware platforms
- Out-of-the-box steps to deploy 100+ application and resource types
- Intelligent deployment logic captures middleware-specific dependencies

Key differences XL Scalability



Enterprise Scalability

- No manual creation of workflows
- Agentless architecture
- Auto-scalable AutoFlow engine
- True model-based automation generates optimal deployment plans
- Automatic discovery of target environments

XL Deploy Training

Key differences

Lightweight and Cloud



Lightweight and Cloud

- Agentless architecture for compatibility with standard public/private cloud images
- Connect to Windows & Unix target systems using standard remote protocols
- No agent install
- No firewall ports to be opened
- No security reviews



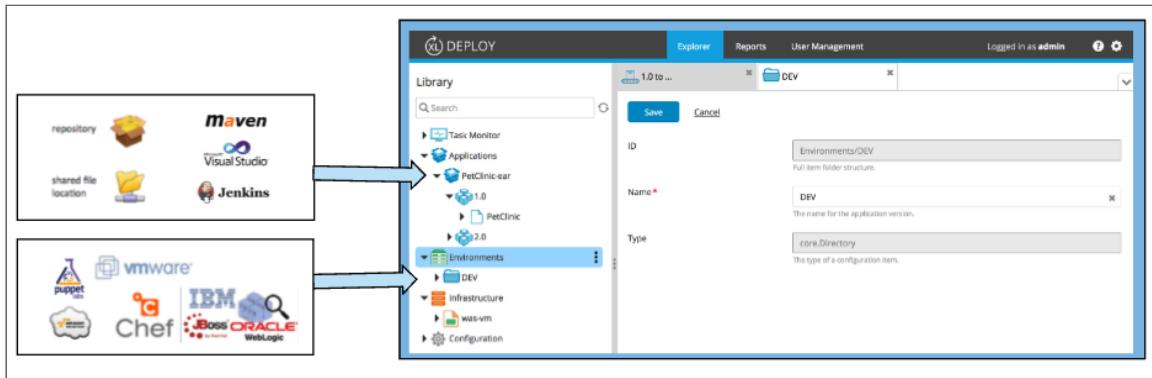
Key differences Insight and Compliance



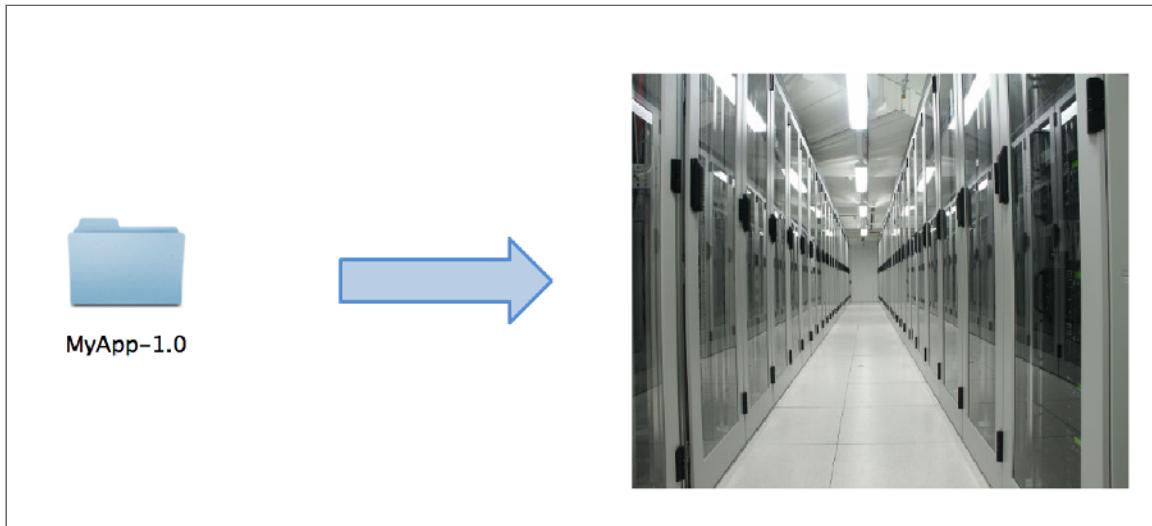
Insight and Compliance

- Built-in reporting offers quality checks throughout the deployment process
- Visualization of your application deployment models
- Easily compare deployments across target environments, servers with a single view
- Pipeline dashboard for continuous delivery

XL Deploy - Introduction Integration



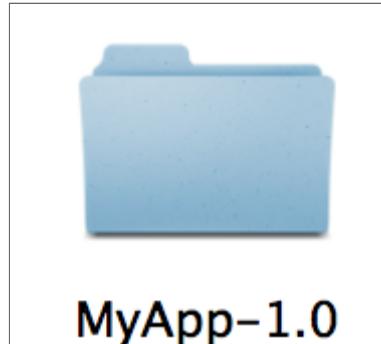
XL Deploy - Introduction Deployments





What is an application? 1/2

- Binary or source artifacts
 - WARs
 - EARs
 - Native files
 - ...
- Server configuration
 - Datasources
 - Queues
 - Connection factories
 - ...



XL Deployments

What is an application? 2/2

Application
Configuration

- Static content
- Properties files
- XML files
- Shared libraries
- Proxy configurations
- Logger configurations
- ...



MyApp-1.0



Deployments

Deployments 1/3

- Applications
 - many files and configurations
- Deployment procedures
 - numerous deployment scripts, manuals and policies to take care of
- Environments
 - complex constellation of target environments, multiple technologies

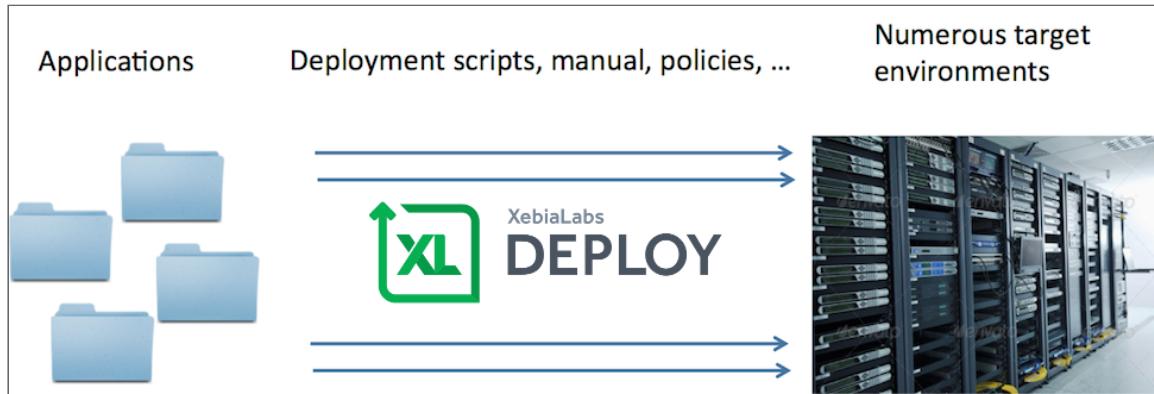
XL Deployments

Deployments 2/3



XL Deployments

Deployments 3/3



XL Deployments

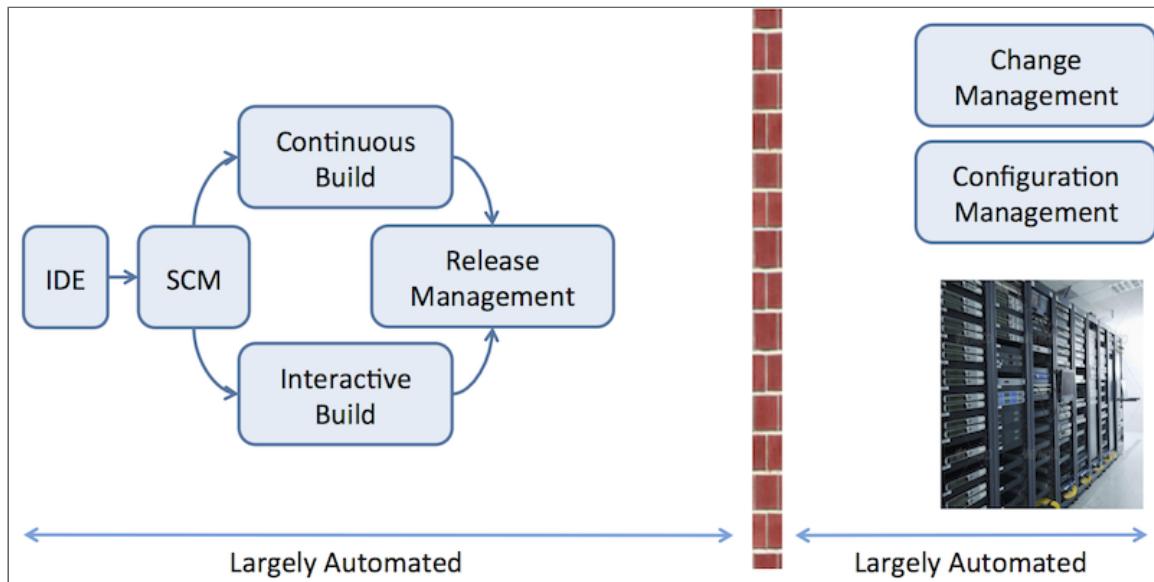
What is a deployment?



- Installing applications
- Configuring resources
- Configuring middleware components
- Starting / stopping components and doing all the above in the right order
- Configuring the installed application for different environments
- Making the application available to end users

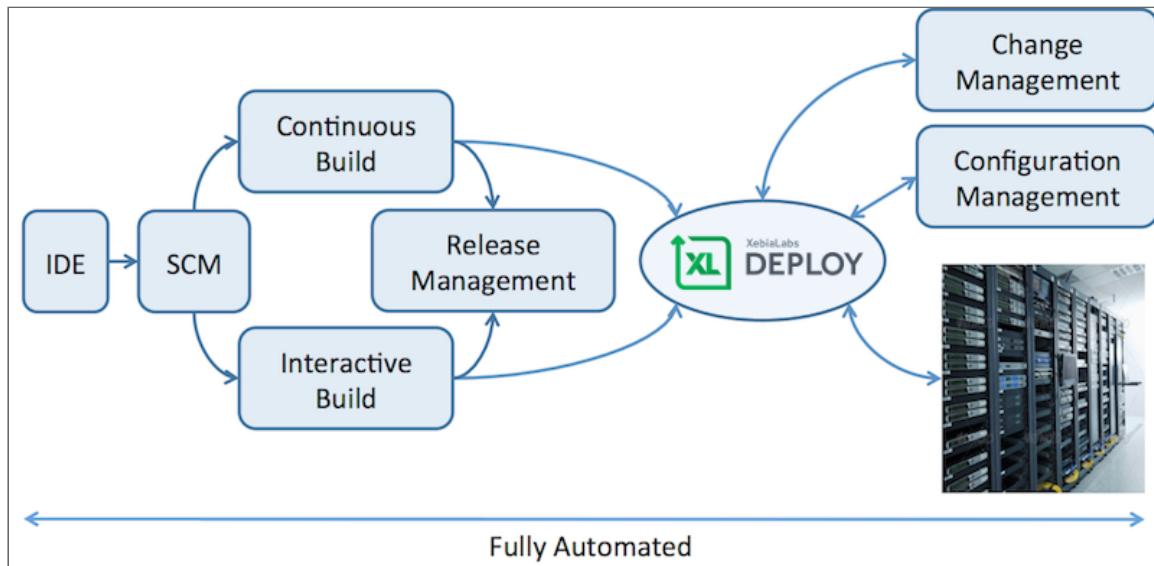
XL Deployments

End-to-end automation 1/2



XL Deployments

End-to-end automation 2/2





Deployments

Design Principles 1/2

- Implements XebiaLabs' Unified Deployment Model (UDM)
- Model based! No tedious scripting (more on this later)
- A repository of environments and “ready to deploy” versions of applications
- An engine that computes deployment procedures



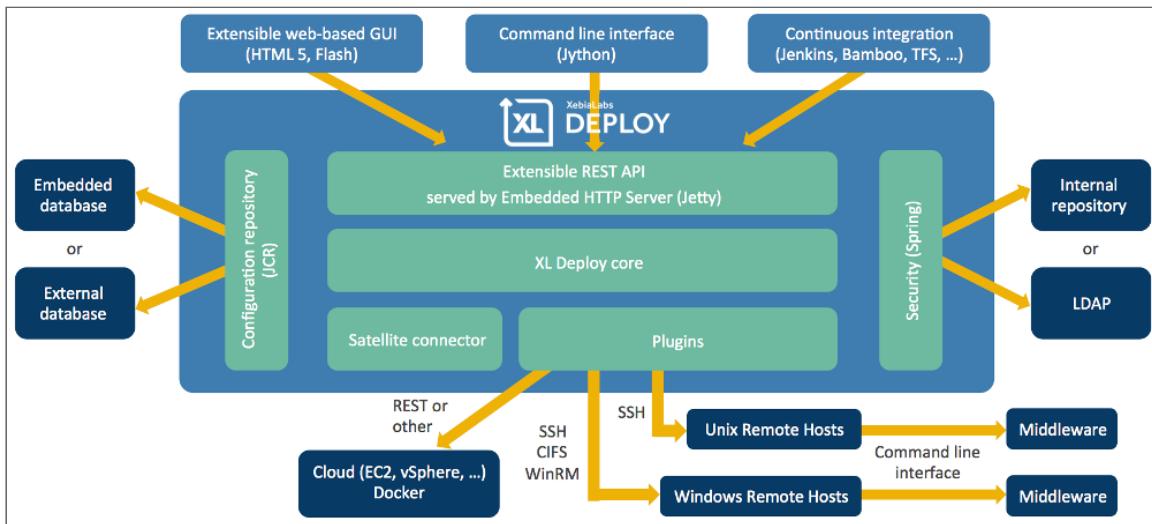
Deployments

Design Principles 2/2

- Packages are pushed to the servers
- Configuration and parameterization of necessary resources
- Uses the middleware's native interfaces wherever possible
- Complete package delivery strategy: XL Deploy calculates the difference and decides on what needs to be updated or not

XL Deploy - Introduction

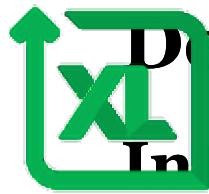
Architecture





Deployment concepts

<http://www.xebialabs.com>



Deployment concepts

Introduction

People:



Developers



Managers



Operations

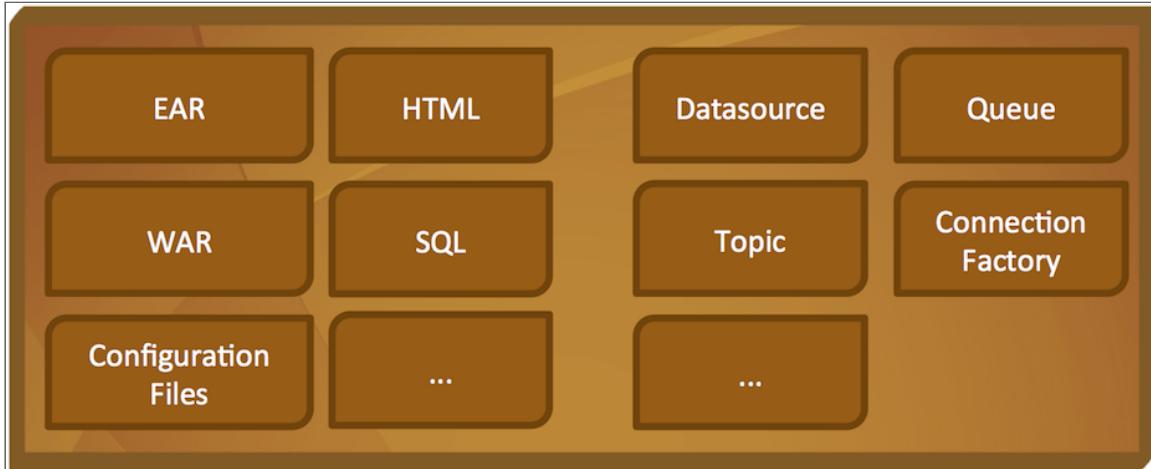
Our goal:

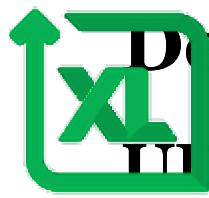
- Non-middleware experts, use same concepts, system, interface to deploy to different middleware landscapes.
- Thus UDM concept.
Not magical or new.
Given names related to our product.

XL Deploy Training

Deployment concepts

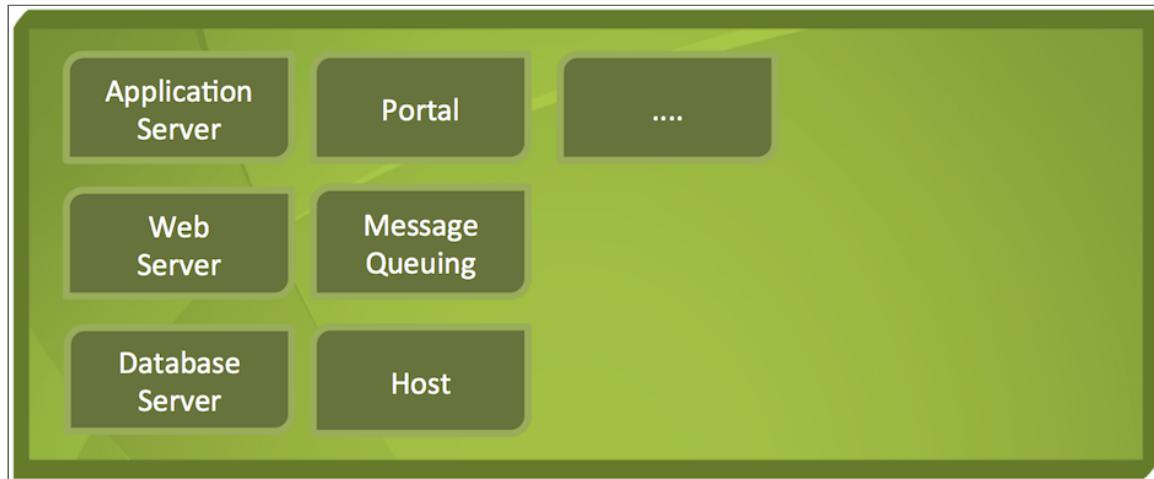
DOM: Deployment Packages



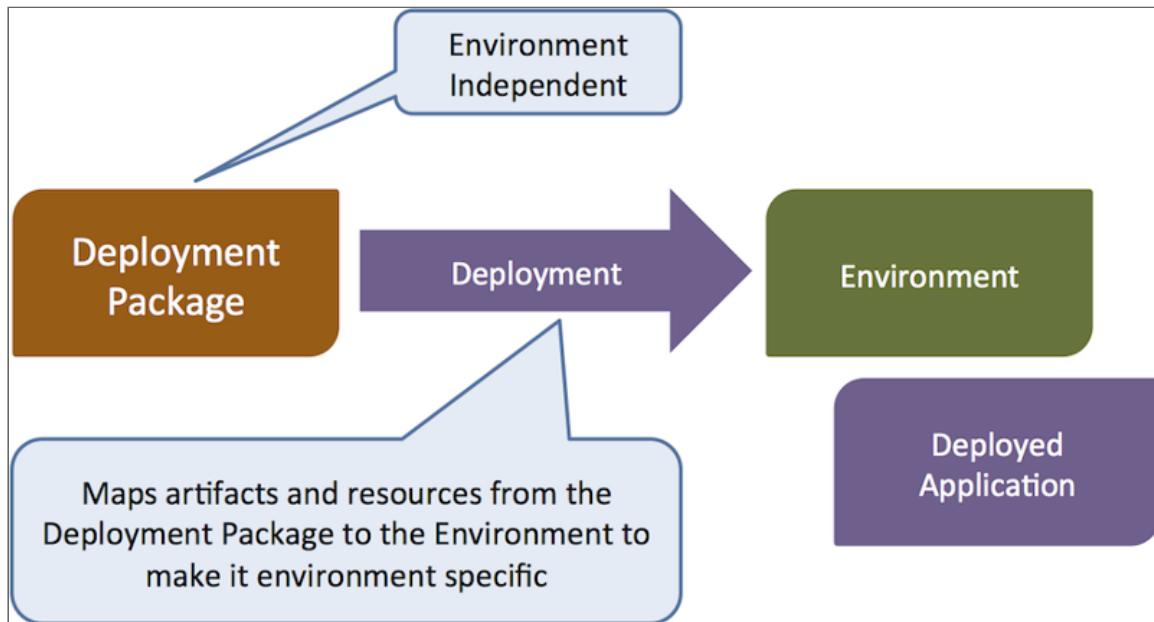


Deployment concepts

CDM: Environments



XL Deploy: Deployments

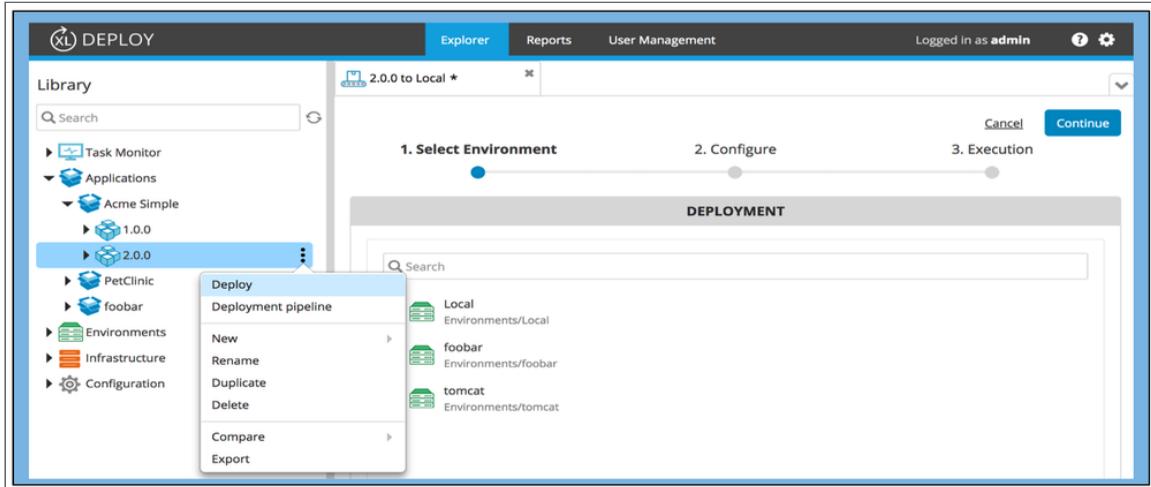


Deployment concepts XL DEPLOY and GUI 1/6

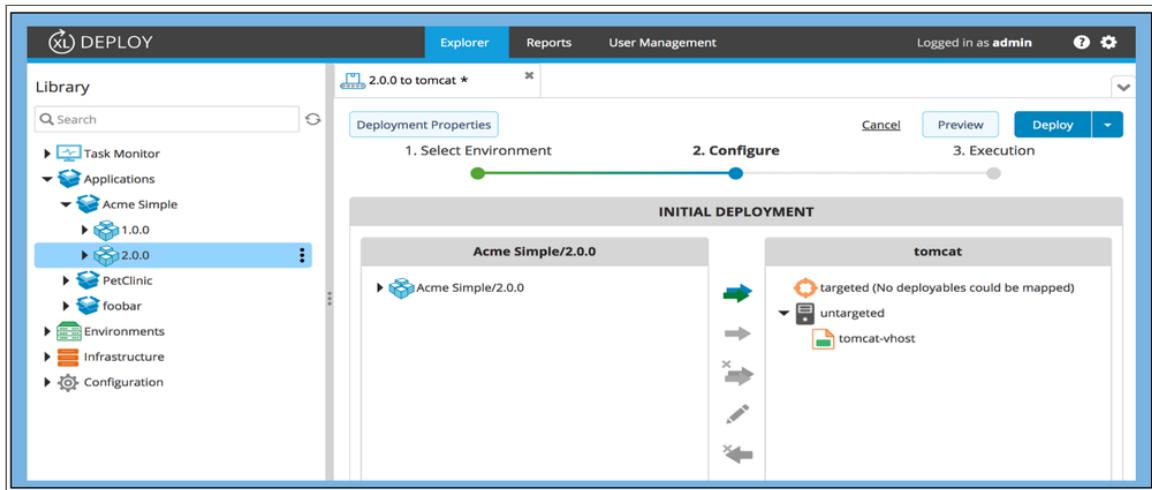
The screenshot shows the XL Deploy web interface. On the left, there's a sidebar titled 'Library' with a search bar and links for 'Task Monitor', 'Applications', 'Environments', 'Infrastructure', and 'Configuration'. The main area has a title 'Quick tour of the new XL Deploy' and contains a screenshot of the interface with annotations:

- A callout points to the 'Task Monitor' link in the sidebar with the text: "Double-click a configuration item to open and edit."
- A callout points to the 'Deploy' package in the tree view with the text: "Click ⚡ on a package to start a deployment."
- A callout points to the top right corner with the text: "Click ⚡ to go to the legacy interface."

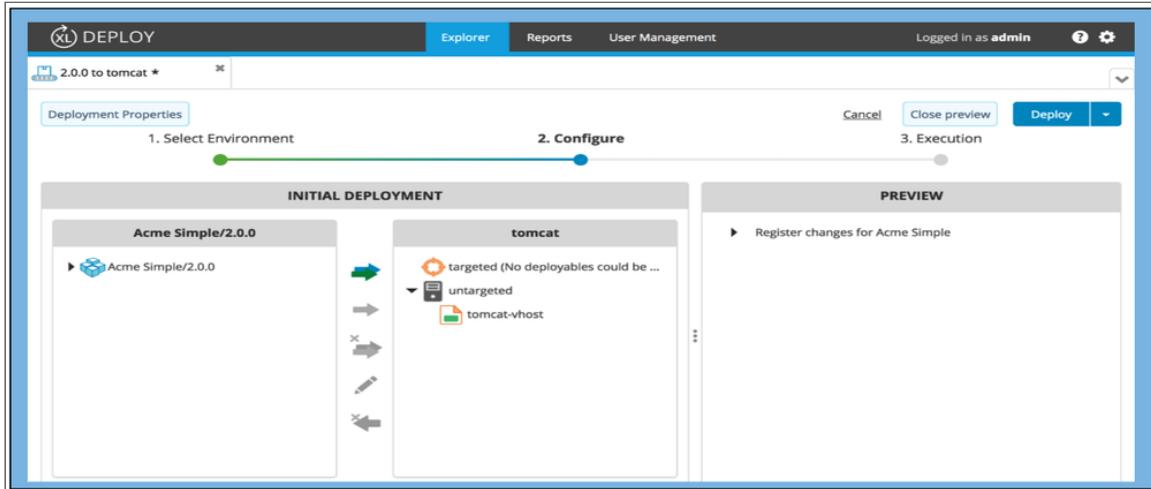
XL Deploy Deployment concepts CLI and GUI 2/6



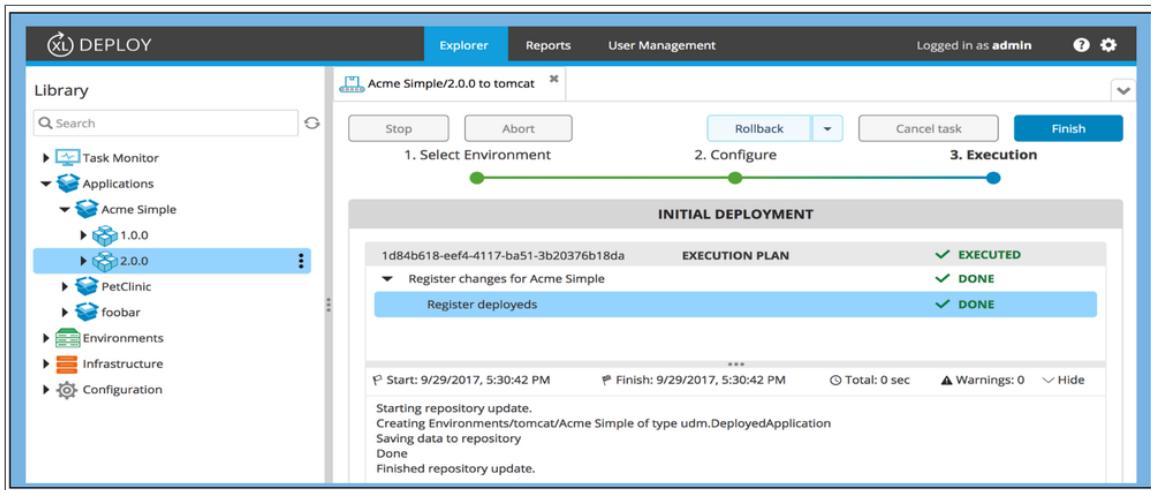
Deployment concepts XLDM and GUI 3/6



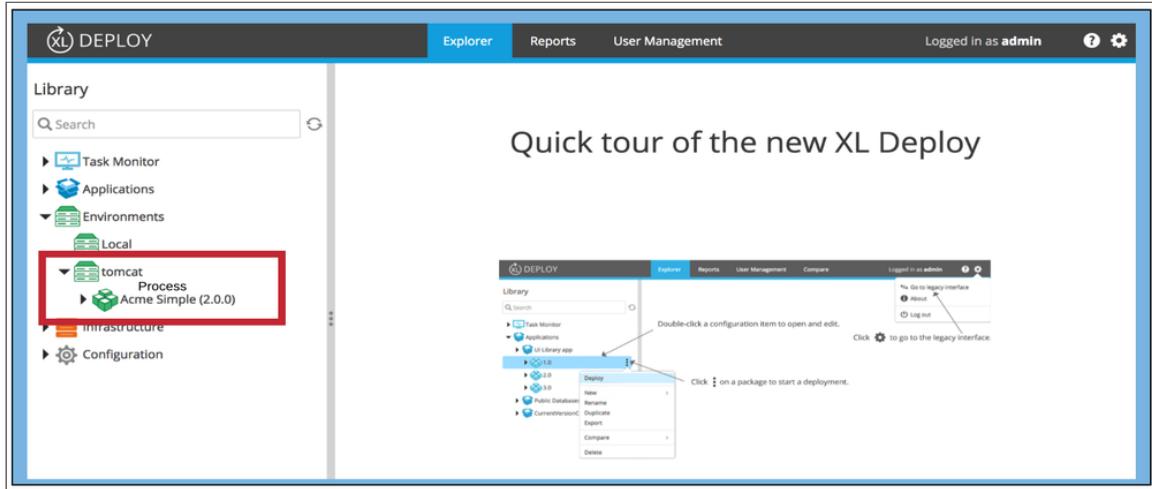
Deployment concepts XL DEPLOY and GUI 4/6



Deployment concepts UDM and GUI 5/6



Deployment concepts XL DEPLOY and GUI 6/6





Installation

<http://www.xebialabs.com>

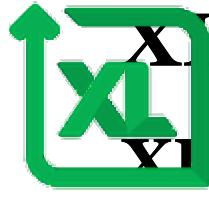


XL Deploy - Installation

XL Deploy distribution 1/3

Parts of the distribution are:

- XL Deploy server
- XL Deploy command-line interface (CLI)
- Plugins



XL Deploy - Installation

XL Deploy distribution 2/3

- XL Deploy server command-line installer:
 - xl-deploy-[VersionNumber]-server.zip
- XL Deploy server GUI installer:
 - xl-deploy-windows-x64_[VersionNumber].exe
 - xl-deploy-os-x_[VersionNumber].dmg
- XL Deploy CLI:
 - xl-deploy-[VersionNumber]-cli.zip



XL Deploy - Installation

XL Deploy distribution 3/3

Plugins:

- [pluginName]-plugin-[versionNumber].jar
- [pluginName]-plugin-[versionNumber]-x1Plugin.xldp (includes all dependencies)



XL Deploy - Installation

XL Deploy server distribution

Directory	Purpose
/bin	Command line launchers
/conf	Configuration files
/ext	Extensions
/plugins	Plugin directory



XL Deploy - Installation

XL Deploy CLI distribution

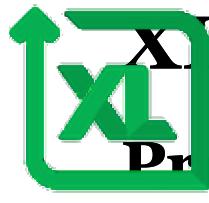
Note: The CLI program is a REST client. As such it can be installed on any remote machine and will connect to XLD server to execute script instructions.

Directory	Purpose
/bin	Command line launchers
/ext	Extensions



XL Deploy - Installation

Exercise: Setup your training environment



XL Deploy - Installation

Prepare your installation

Requirement: JDK 1.8+ must be installed on your machine

Directory	Purpose	Example
<TRAINING>	Training material copy root directory	c:\training_m
<PROJECT_FOLDER>	Your training working directory	c:\projects\x
<SERVER>	XL Deploy server installation directory	c:\projects\x deploy-[Version] server
<CLI>	XL Deploy CLI installation directory	c:\projects\x deploy-[Version] cli



XL Deploy - Installation

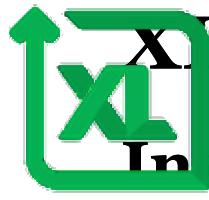
Exercise: Install XL Deploy Server - WebSphere



XL Deploy - Installation

Install XL Deploy Server

- Unzip <TRAINING>/xl-deploy-[VersionNumber]-server.zip into <PROJECT FOLDER>
- Copy all files from <TRAINING>/license to <SERVER>/conf
- Copy all files from <TRAINING>/packages to <SERVER>/importablePackages
- Copy all files from <TRAINING>/ext to <SERVER>/ext
- Copy all files from <TRAINING>/plugins to <SERVER>/plugins



XL Deploy - Installation

Install XL Deploy Server

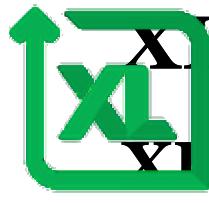
- Open a command prompt and verify that JAVA 8+ is in your path (`java -version`)
- Go to <TRAINING>/xl-deploy-[VersionNumber]-server/bin and execute `run.sh` or `run.cmd`
- For the training setup, answer the setup questions as follows:
 - Use simple setup: [yes], Admin password: [press Enter], Confirm admin password: [press Enter], Initialize the JCR repository: [yes], Set encryption password: [no], Confirm all settings: [yes]
- Wait for the lines XL Deploy server has started and You can now point your browser to `http://localhost:4516/` to appear in the log before opening the browser.

XL Deploy - Installation

Log in to your XL Deploy server

- Open your browser and go to <http://localhost:4516/>
- Log in using admin / admin

The image shows a login interface with a dark grey header bar containing the word "Login". Below it is a white form area. On the left, there is a red asterisk followed by the word "User" and a text input field containing "admin". To its right is another red asterisk followed by the word "Password" and a text input field containing "*****". Below these fields is a checkbox labeled "Keep me logged in" with an unchecked state. At the bottom right is a dark grey "Login" button.



XL Deploy - Installation

XL Deploy CLI setup

- Unzip <TRAINING>/xl-deploy-[VersionNumber]-cli.zip into <PROJECT FOLDER>
- Copy all files from <TRAINING>/cli-ext to xl-deploy-[VersionNumber]-cli/ext
- Open a command prompt and verify that JAVA 8+ is in your path (java -version)
- Go to <CLI>/xl-deploy-[VersionNumber]-cli/bin and execute cli.sh or cli.cmd (your XL Deploy server must be running)
- Log in using admin / admin



XL Deploy - Installation

XL Deploy CLI setup

```
Username: admin
Password:
Welcome to the XL Deploy Jython CLI!
Type 'help' to learn about the objects you can use to interact with XL Deploy.

XL Deploy Objects available on the CLI:

* deployit: The main gateway to interfacing with XL Deploy.
* deployment: Perform tasks related to setting up deployments
* factory: Helper that can construct Configuration Items (CI) and Artifacts
* repository: Gateway to doing CRUD operations on all types of CIs
* security: Access to the security settings of XL Deploy.
* task2: Access to the task block engine of XL Deploy.
* tasks: Access to the task engine of XL Deploy.

To know more about a specific object, type <objectname>.help()
To get to know more about a specific method of an object, type <objectname>.help("<methodname>")

Reading extension: ext/ci.py
Reading extension: ext/default-imports.py
Reading extension: ext/import-export-dictionaries.py
Reading extension: ext/load-into-dict.py
Reading extension: ext/readme.cli
admin > |
```



XL Deploy - Installation

**Exercise: Setting up your training
VM - WebSphere**



XL Deploy - Installation

Installing VirtualBox

- Install the latest version of VirtualBox appropriate for your hardware.
- For Windows 7, download the installer then install from the command line using:

```
VirtualBox-5.1.12-112440-Win.exe -msiparams NETWORKTYPE=NDIS5
```



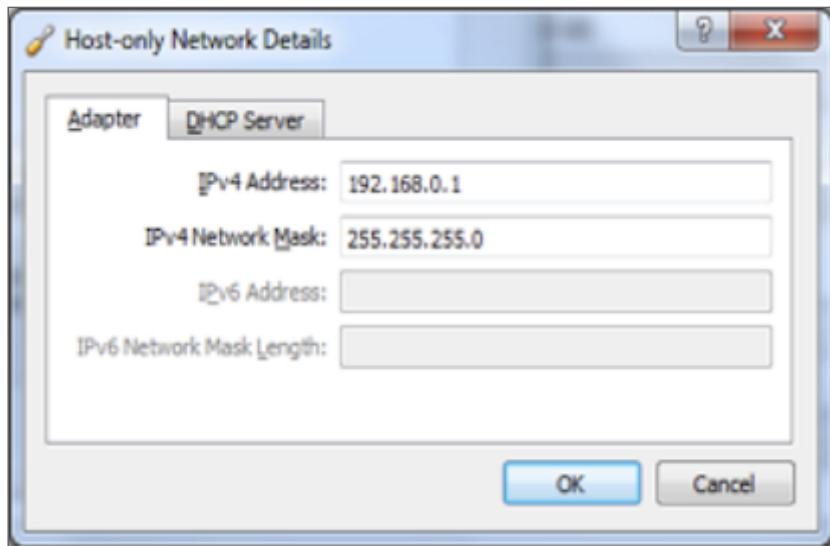
XL Deploy - Installation

Setting up your training VM

- In VirtualBox, go to:
 - File ► Preferences (CTRL-G) ► Network
 - Add (or update) a "host only" adapter with the following settings (no DHCP):
 - See next slide for a screenshot

XL Deploy - Installation

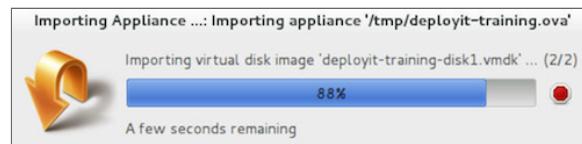
Setting up your training VM



XL Deploy - Installation

Setting up your training VM

- Import the VM in VirtualBox
- To import the image, in VirtualBox, go to:
 - File ► Import Appliance (CTRL-I)
 - Select the .ova image (provided by the trainer)
 - Accept the default settings and press Import
- Power-on your VM





XL Deploy - Installation

Troubleshooting your VM installation

- If you encounter "Failed to open/create the internal network (VERR_INTNET_FLT_IF_NOT_FOUND) see the note regarding the installing from the command-line above. More information can be found at
<https://forums.virtualbox.org/viewtopic.php?f=6&t=70199>
- If your VM is not "found" on your network, check if you are logged into a VPN. If so, log out of the VPN and try again.



XL Deploy - Installation

WebSphere smoke test

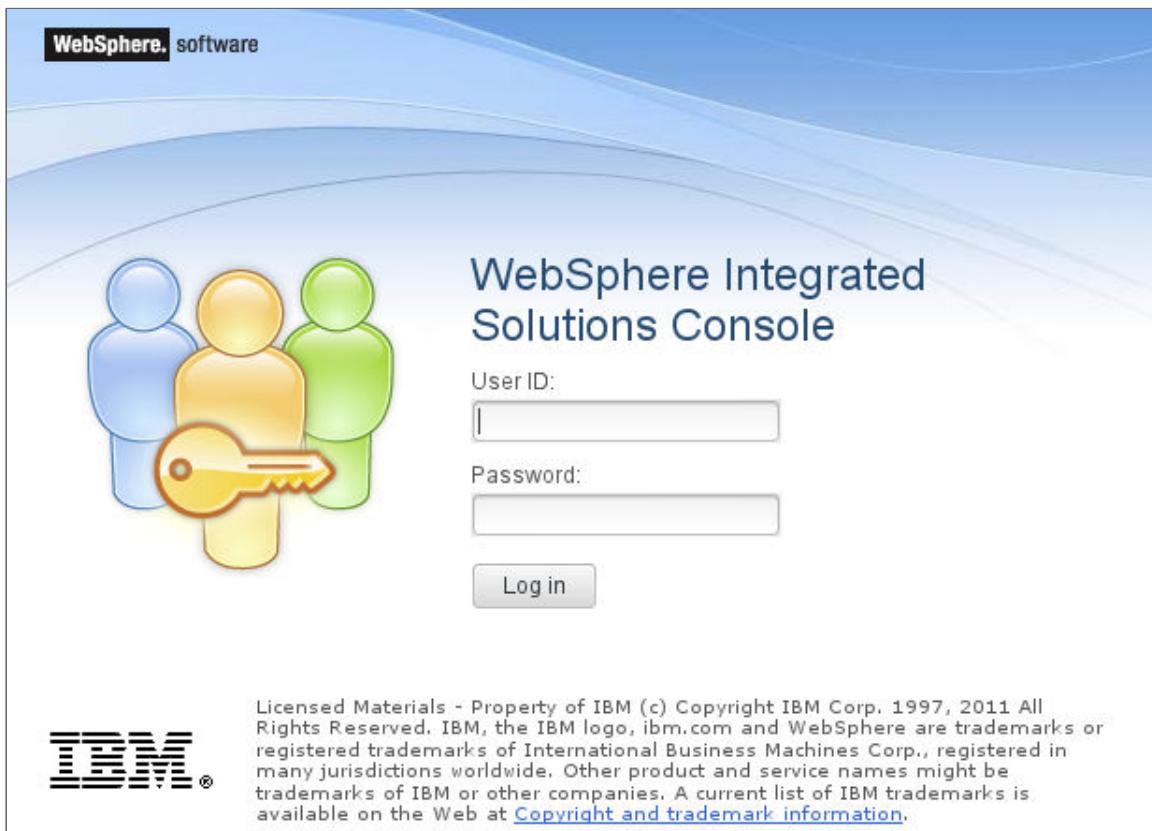
Check the following URLs:

Component	URL
IBM HTTP Server	<u>http://10.0.0.x/</u>
WebSphere	<u>https://10.0.0.x:9043/admin/</u>
Tomcat	<u>http://10.0.0.x:8280/</u>

XL Deploy - Installation

WebSphere smoke test

Log in to the WebSphere console using: **admin / admin**



XL Deploy - Installation

WebSphere smoke test

The screenshot shows the 'Welcome' screen of the WebSphere Integrated Solutions Console. At the top, there's a navigation bar with 'WebSphere. software', 'Welcome admin', 'Help | Logout', and the IBM logo. On the left, a sidebar titled 'View: All tasks' lists various administrative categories: Welcome, Guided Activities, Servers, Applications, Jobs, Services, Resources, Runtime Operations, Security, Operational policies, Environment, System administration, Users and Groups, Monitoring and Tuning, Troubleshooting, Service integration, and UDDI. The main content area has a 'Welcome' header and a descriptive paragraph about the console's purpose. Below this is a table showing product suites:

Suite Name	Version
WebSphere Application Server	8.5.0.0

On the right, there's an 'About this Integrated Solutions Console' panel with information like version 8.5.0.0, build number gm1218.01, and build date 5/1/12. It also mentions licensed materials for IBM products.



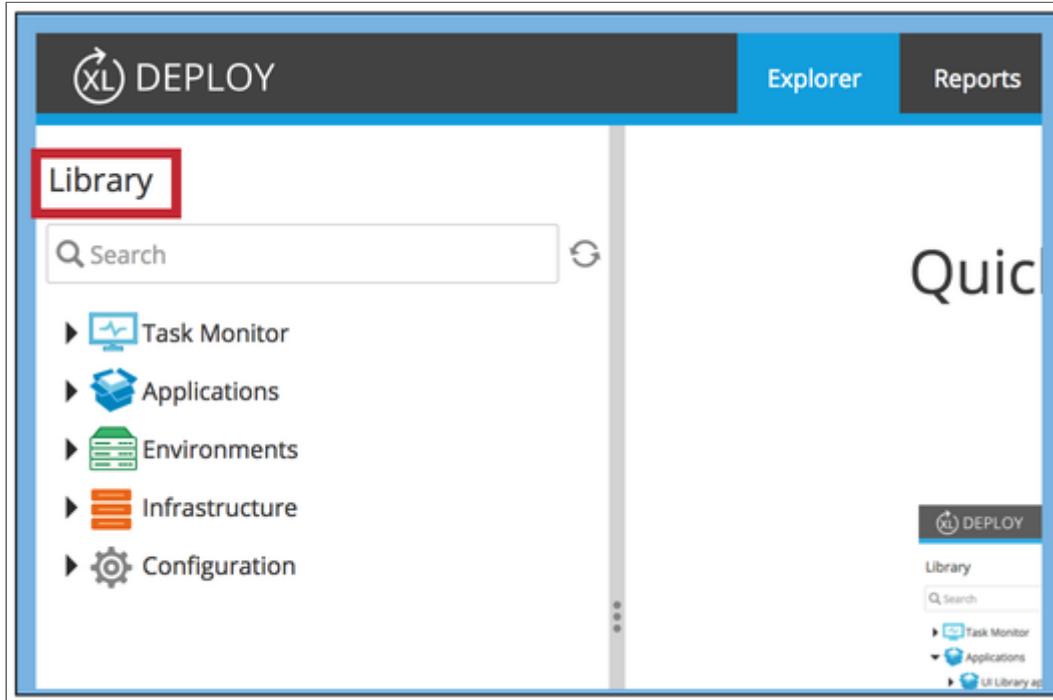
Library

<http://www.xebialabs.com>

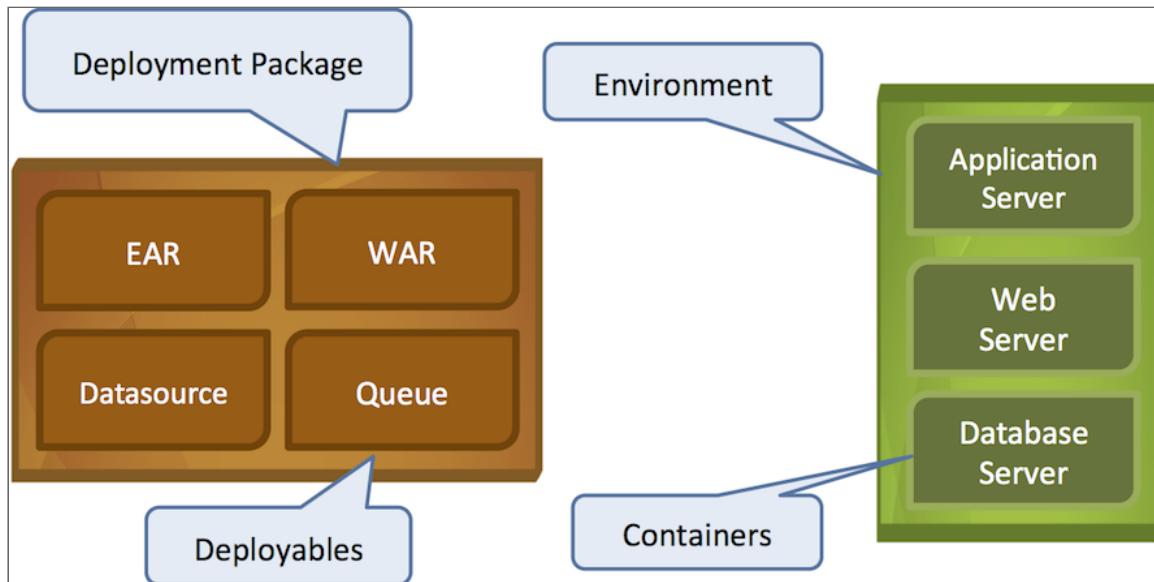
XL Deploy Library

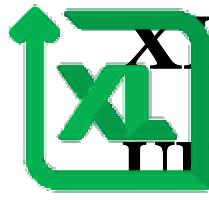
Introduction

We now have an empty library...



XL Deploy Library JNDI terminology





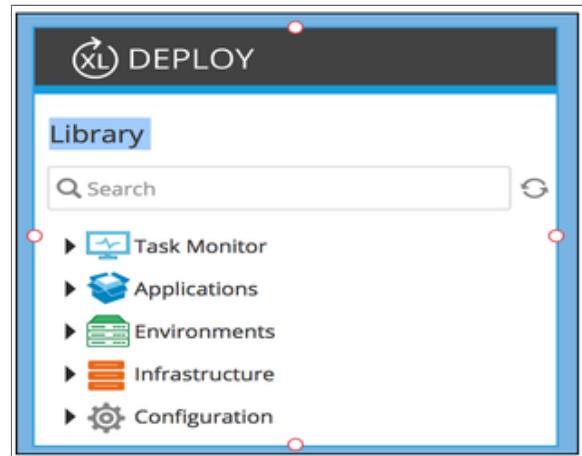
XL Deploy Library

JM terminology

- A **deployment package** contains **deployables**:
 - Artifacts: Files, ZIP archives, and so on
 - Resources (such as datasources definitions, queues definitions, and so on)
- An **environment** is a collection of **containers**: Application servers, database servers, Linux hosts, Windows hosts, and so on

XL Deploy Library

- All data XL Deploy uses is stored in the Library
- Hierarchical database (like a file system)
- Accessible from Library section in UI
- Five top-level directories:

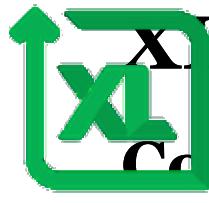




XL Deploy Library

Configuration Items 1/2

- Configuration items (CIs) are objects in the XL Deploy Library.
- Have an ID such as:
 - **Infrastructure/jboss-vm/jboss-51**
 - **Infrastructure/was-vm/was8-Cello1**
- The ID is the path in the library
- Have a name, which is the last part of the ID, such as:
 - **jboss-vm**
 - **was-vm**



XL Deploy Library

Configuration Items 2/2

- Have a type such as:
 - **jbossas.ServerV5**
 - **wls.Cluster**
 - The type defines zero or more properties and zero or more children such as:
 - **String (set/list)**
 - **Integer**
 - **Boolean**



XL Deploy Library

Exercise: Define a WebSphere host



XL Deploy Library

Exercise: Define a WebSphere host

1/3

- On the **Library** tab of the GUI:
 - Click on the three dots to the right of:
Infrastructure
 - Tip: Hover your mouse over **Infrastructure** and the option will appear
 - Choose: **New ▶ overthere ▶ SshHost**
 - Enter the name of the host: **was-vm**
 - Enter the properties (see next **2** slides)



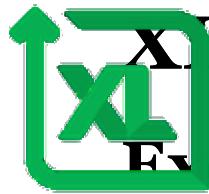
XL Deploy Library

Exercise: Define a WebSphere host

2/3

- In the **Common** section :

Property	Value
Operating System	UNIX
Connection type	SUDO
Address	10.0.0.x
Username	vagrant
Password	vagrant



XL Deploy Library

Exercise: Define a Websphere host

3/3

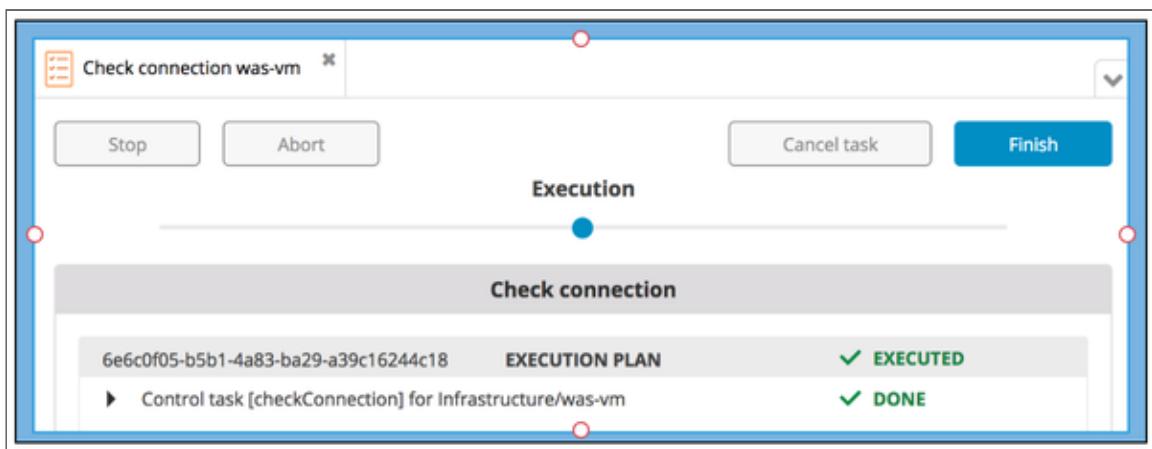
- In the **Advanced** section :

Property	Value
SUDO username	root



Exercise: Define a WebSphere host , check connectivity

- On the Library section of the GUI:
 - Click on the three dots to the right of: Infrastructure ► was-vm
 - Choose: Check connection
 - Press the Execute button





XL Deploy Library

Exercise: Define a Websphere host, check connectivity

- If the task succeeds you have properly configured the host for use with XL Deploy !

The screenshot shows the 'Check connection was-vm' task in the XL Deploy interface. The task is currently executing, as indicated by the progress bar at the top. The main area displays the 'Check connection' execution plan:

EXECUTION PLAN	STATUS
Control task [checkConnection] for Infrastructure/was-vm	EXECUTED
Check whether XL Deploy can transfer files to 192.168.57.101	DONE
Check whether XL Deploy can execute commands on 192.168.57.101	DONE

Below the execution plan, there is a log window showing the contents of a temporary directory and successfully executed commands:

```
Listing the contents of the temporary directory...
hsperfdta_jenkins
hsperfdta_tomcat6
javasharedresources
jna
Log
PermanentStore
TemporaryStore
tomcat6-tomcat6-tmp
winstone4210809624089999763.jar
winstone.tmp
Successfully executed commands on 192.168.57.101.
```



Setting the stage

<http://www.xebialabs.com>



Setting the stage

Introduction

Before we can do a deployment, we have to put all the necessary items into place in the XL Deploy library:

- Import a deployment package
- Discover your infrastructure
- Define an environment

Setting the stage Deployment package

Deployment package:

- Artifacts and resources packaged in a ZIP file with a descriptor
- DAR file (Deployment ARchive)



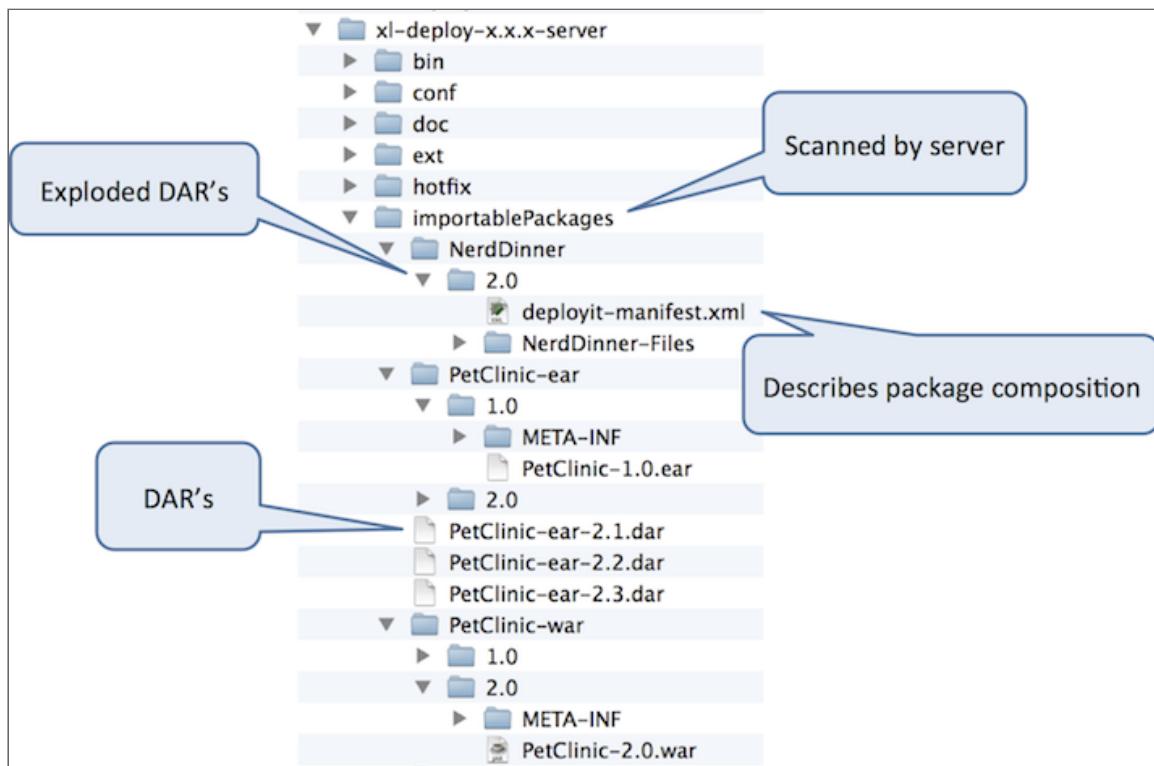


Setting the stage Deployment package

DAR files are usually either:

- Imported into the library as part of a Maven, TFS, or Jenkins build
- Imported from the importable package folder
- Imported from a directory on your local machine
- Imported from a URL

Setting the stage Importable packages folder





Setting the stage

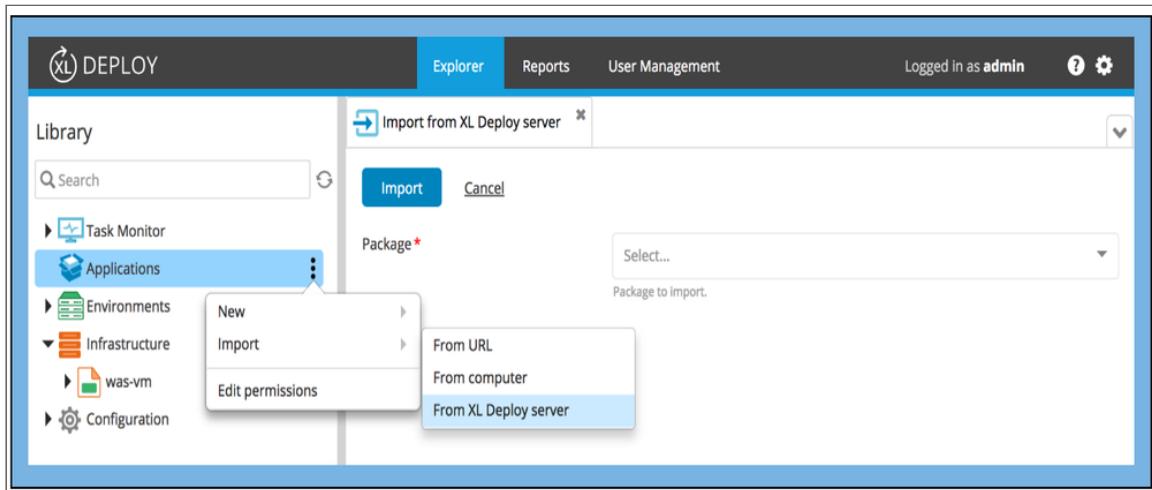
Manifest file

Location: deployit-manifest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<udm.DeploymentPackage version="1.0" application="PetClinic-ear">
    <deployables>
        <jee.Ear name="PetClinic" file="PetClinic-1.0.ear">
            <tags/>
            <scanPlaceholders>false</scanPlaceholders>
            <checksum>96e68b45e844ef1110320fdafe4be53c6</checksum>
        </jee.Ear>
    </deployables>
</udm.DeploymentPackage>
```

Setting the stage

Importing a package

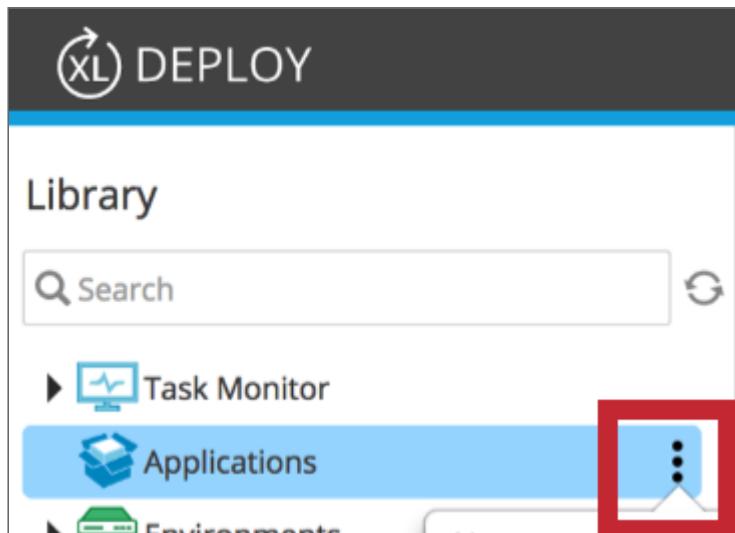


Setting the stage

Exercise: Import packages

In the Library section of the GUI:

- Import all versions of the PetClinic-ear application
- Click on the three dots to the right of **Applications**





Setting the stage

Exercise: Import packages

- Expand **Import**
- Select **From XL Deploy server**
- From the drop-down box, select the version 1.0 PetClinic-ear application
- Click **Import**
- Click **Close**
- Repeat until both versions 1.0 and 2.0 of the PetClinic-ear application are imported

Setting the stage

Exercise: Import packages





Setting the stage

Define middleware containers

- Before we can deploy, we need to define the deployment targets (containers)
- Typically done through:
 - CLI scripts
 - Provisioning tool interface (Puppet, Ansible, and so on)
 - Directly in the GUI (for prototyping and testing)
 - Automatic discovery (if the plugin supports it)



Setting the stage

Define middleware containers with discovery

Discovery:

- Import your existing infrastructure/topology
- Can be triggered from GUI or CLI
- Is handled by the plugin



Setting the stage

Discovery basic steps

- Define a host
- Start discovery for a certain type
- For example: **WebSphere Deployment Manager, JBoss Domain or WebLogic Domain**
- Configure required properties
- Run discovery task
- Review discovered CIs



Setting the stage

Exercise: Discover WAS server



Setting the stage

Exercise: Discover WAS server

- In the **Library** section of the GUI:
 - Click on the three dots to the right of:
Infrastructure ► **was-vm**
 - Choose: **Discover** ► **was** ► **DeploymentManager**
- Enter the following properties for discovery:

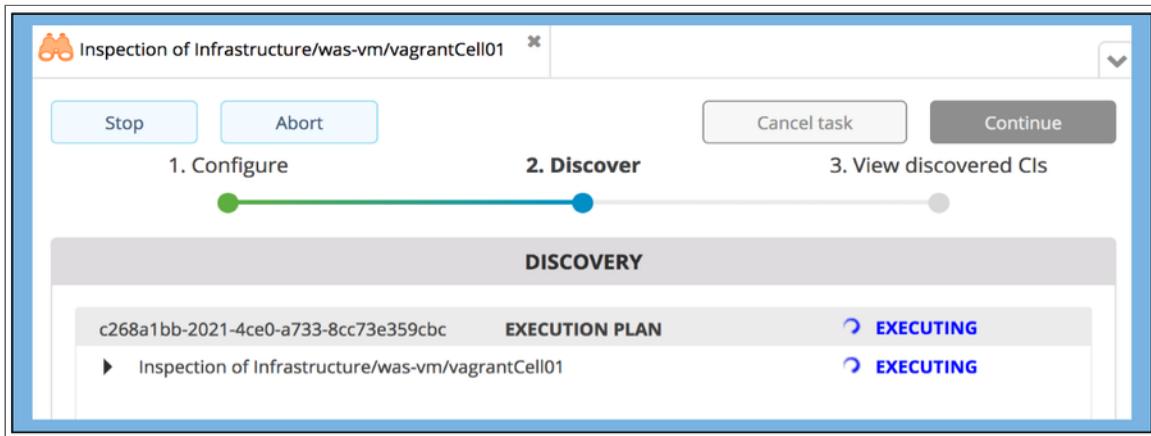
Property	Value
Name	vagrantCello1
WebSphere Installation Path	/opt/IBM/WebSphere/AppServer/
Administrative username	admin
Administrative password	admin



Setting the stage

Exercise: Discover WAS server

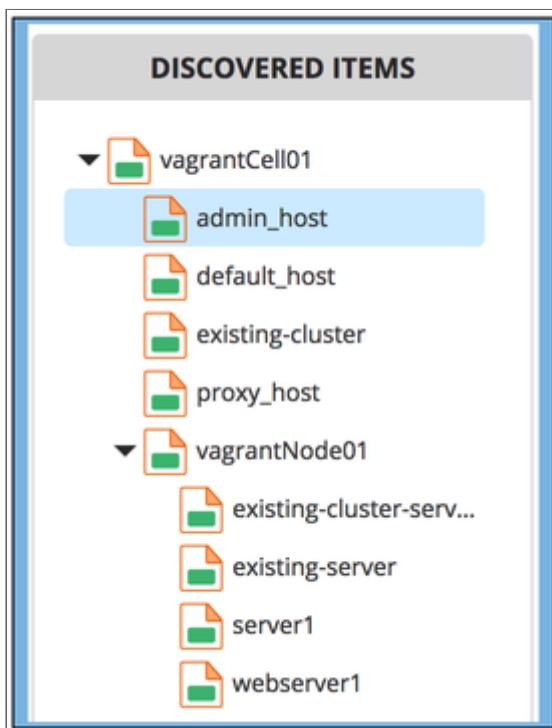
- Click on **Continue** and **Discover**



Setting the stage

Exercise: Discover WAS server

- Inspect the result of the discovery (**View discovered CIs**)
- Click on **Save discovered CIs** to store the items in the **Library**

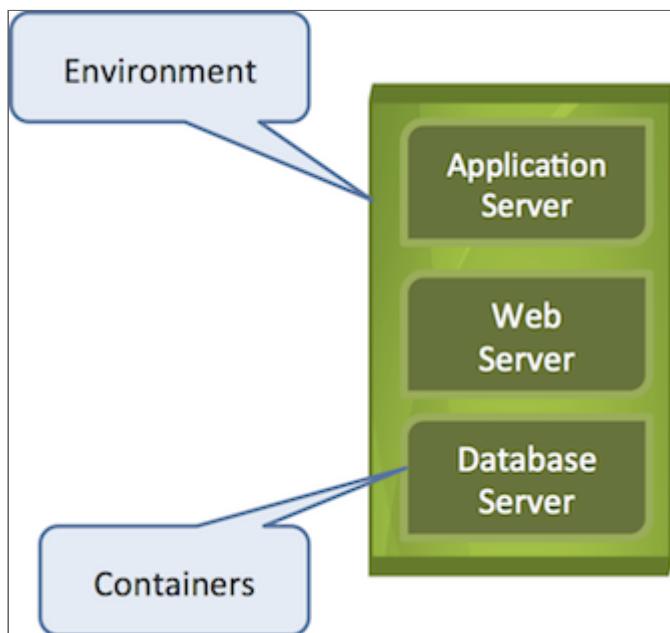




Setting the stage

XL Deploy: Environment

- Has a name such as DEV1, DEV2, ITG1, PROD
- Is a collection of containers (logical groups of containers)





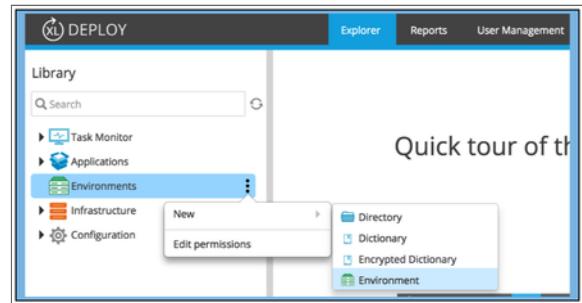
Setting the stage

Exercise: Create an environment

Setting the stage

Exercise: Create an environment

- In the Library section of the GUI, click on the three dots to the right of **Environments**
- Choose: **New ► Environment**





Setting the stage

Exercise: Create an environment

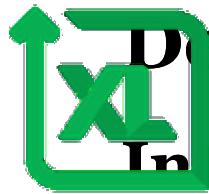
- Enter the following properties:
 - **Name** of the environment:
 - Development
 - In the **Common** section, select the following **Members** from the list of **Containers**:
 - Infrastructure/was-vm/vagrantCello1/existing-cluster
 - Infrastructure/was-vm/vagrantCello1/vagrantNode01
 - Press **Save** to store the new environment





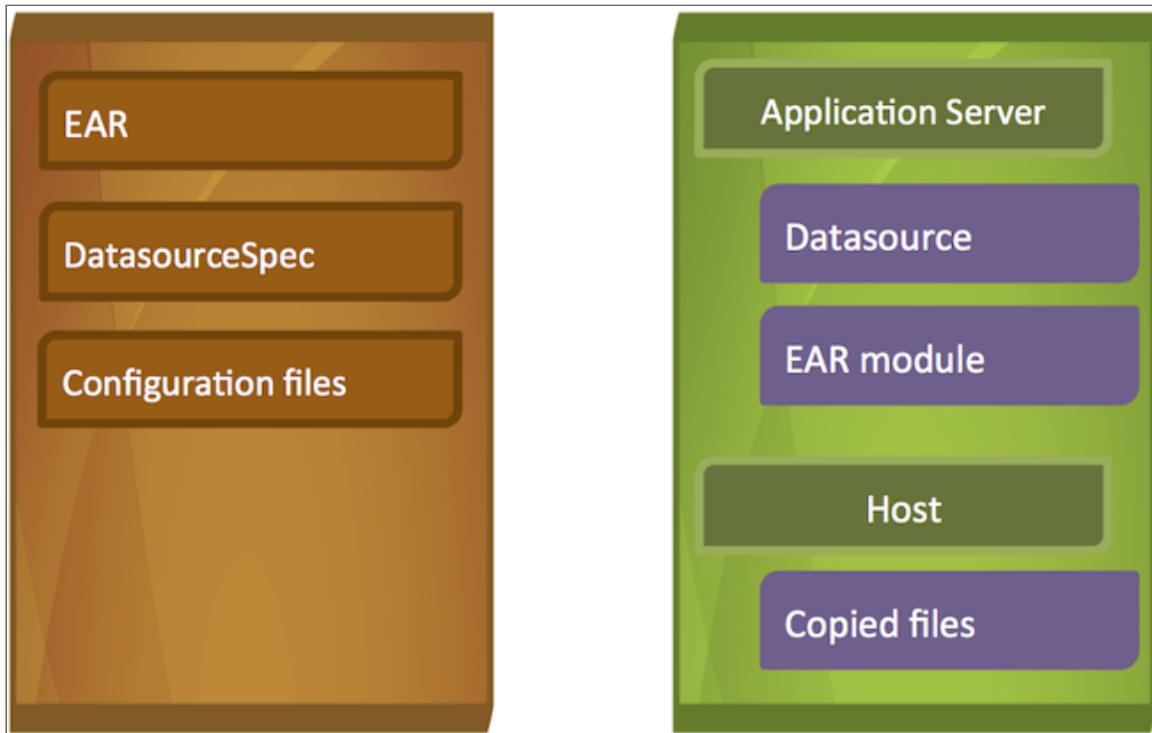
Deployment

<http://www.xebialabs.com>

 Deployment
Introduction

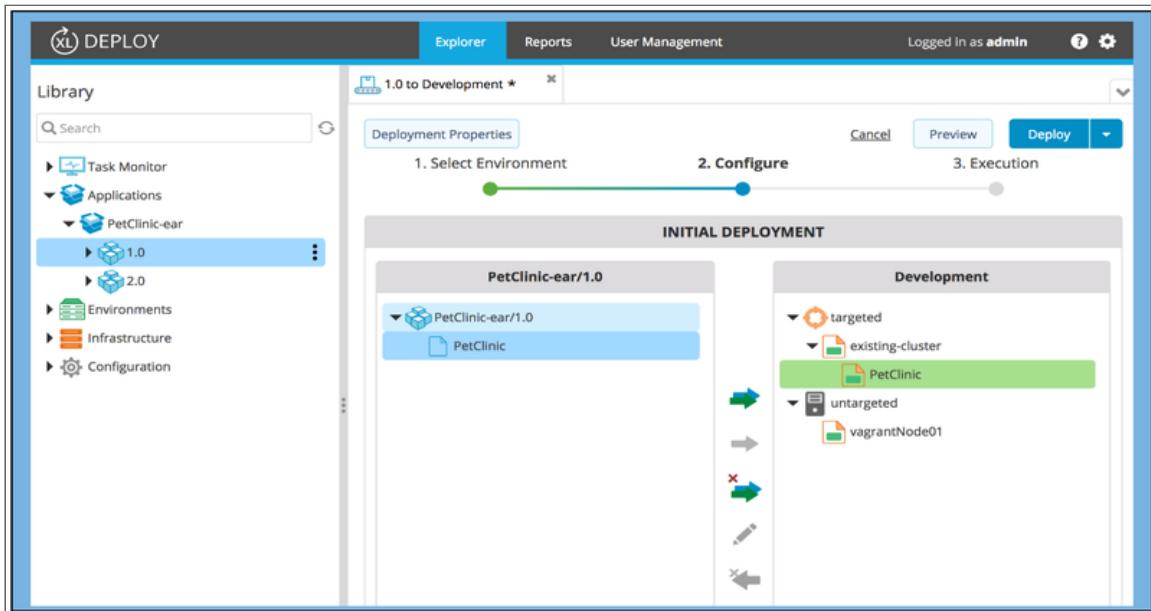
- Deployment packages contains **deployables**
- XL Deploy maps **deployables** to **containers**
- A deployment plan is generated
- As a result of the execution of the deployment plan, **containers** contain **deployeds**

Deployment Introduction

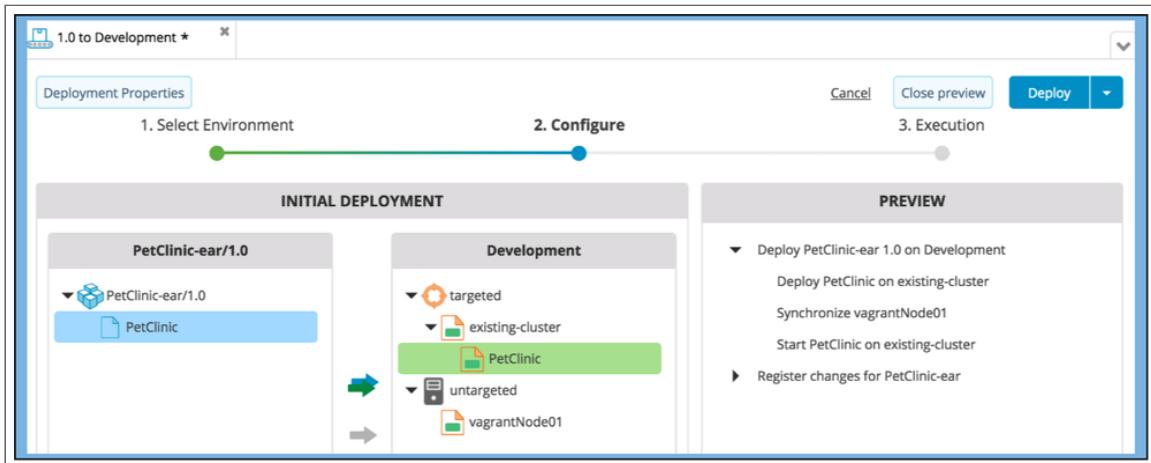


Deployment

Initial deployment



Deployment Plan Analyzer



XL Deployment Preview

Step preview

Step number	2
Description	Synchronize vagrantNode01
Order	75
Rule	com.xebialabs.deployit.plugin.was.container.NodeSynchronizer.syncNodes
Source path	was/base/synchronize-node.py

```
# /Users/nburglin/Desktop/xld-training-was/xl-deploy-7.1.0-server/plugins/python
import sys
import base64
from urllib import quote_plus

##
# WAS 6.1 uses Python version 2.1 which doesn't support boolean literals True
##
if sys.version[:3] == "2.1":
    global False
    False = 0
    global True
    True = 1

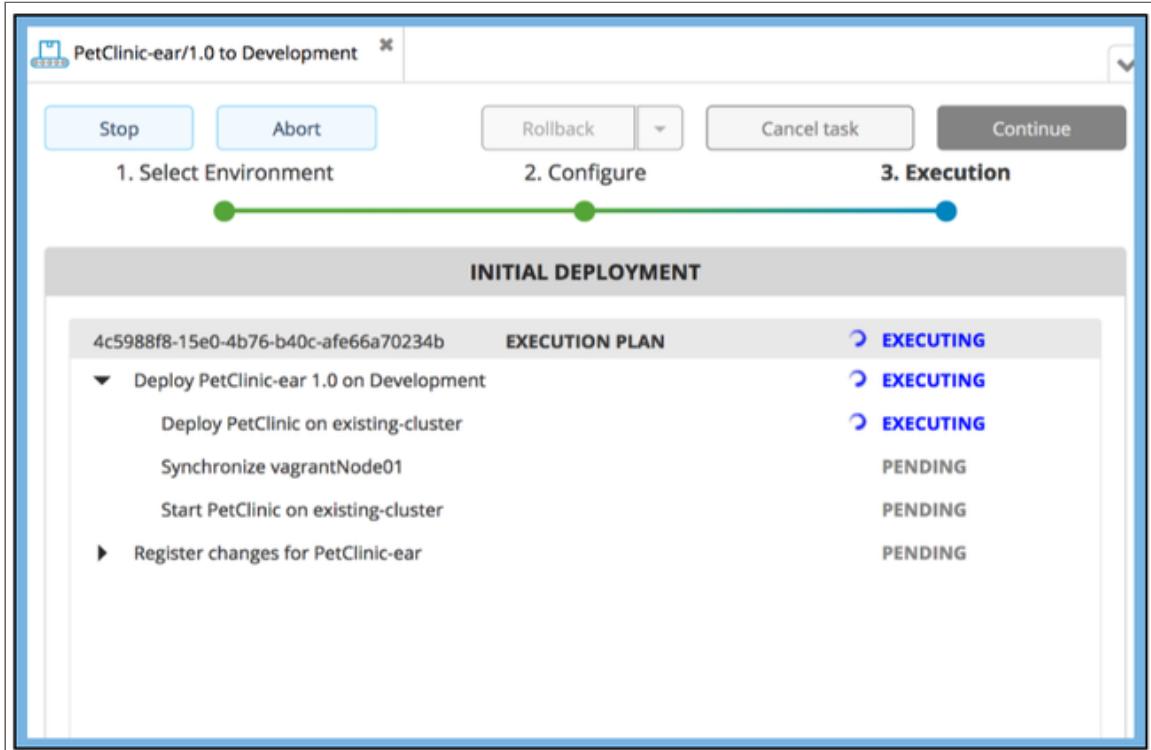
class DictionaryObject:
    def __setattr__(self, propertyName, PropertyValue):
        self.__dict__[propertyName] = PropertyValue

    def getExposedPropertyNames(self):
        return self.__dict__.keys()
```

Close

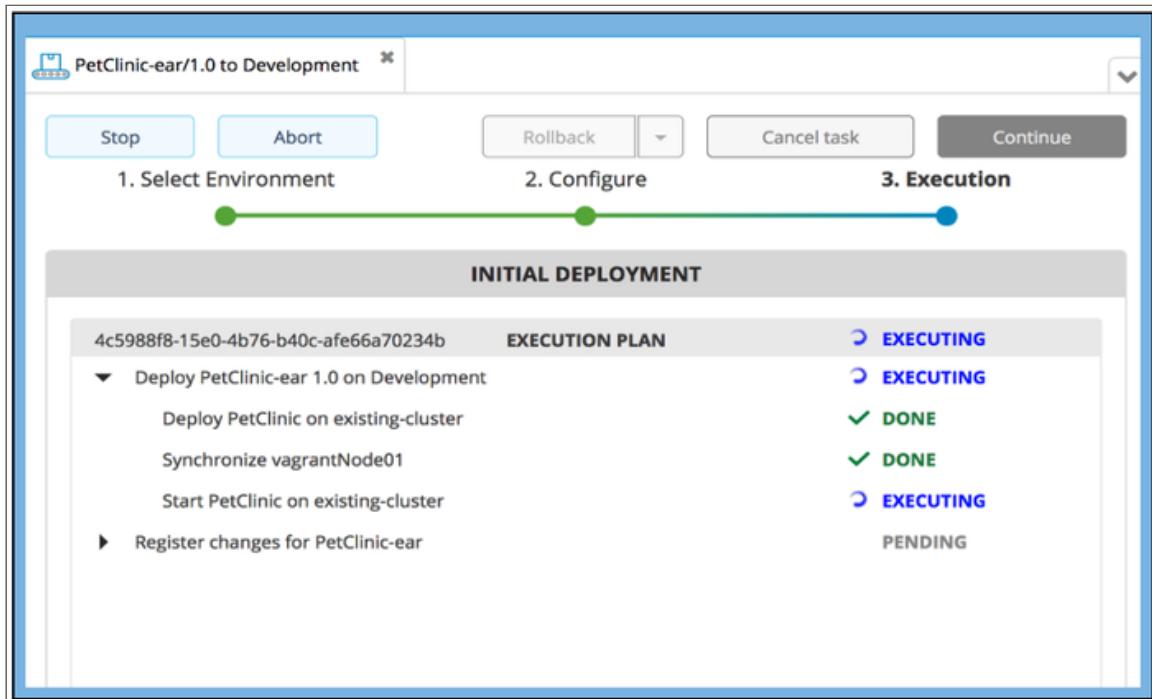
XL Deployment

Deployment plan



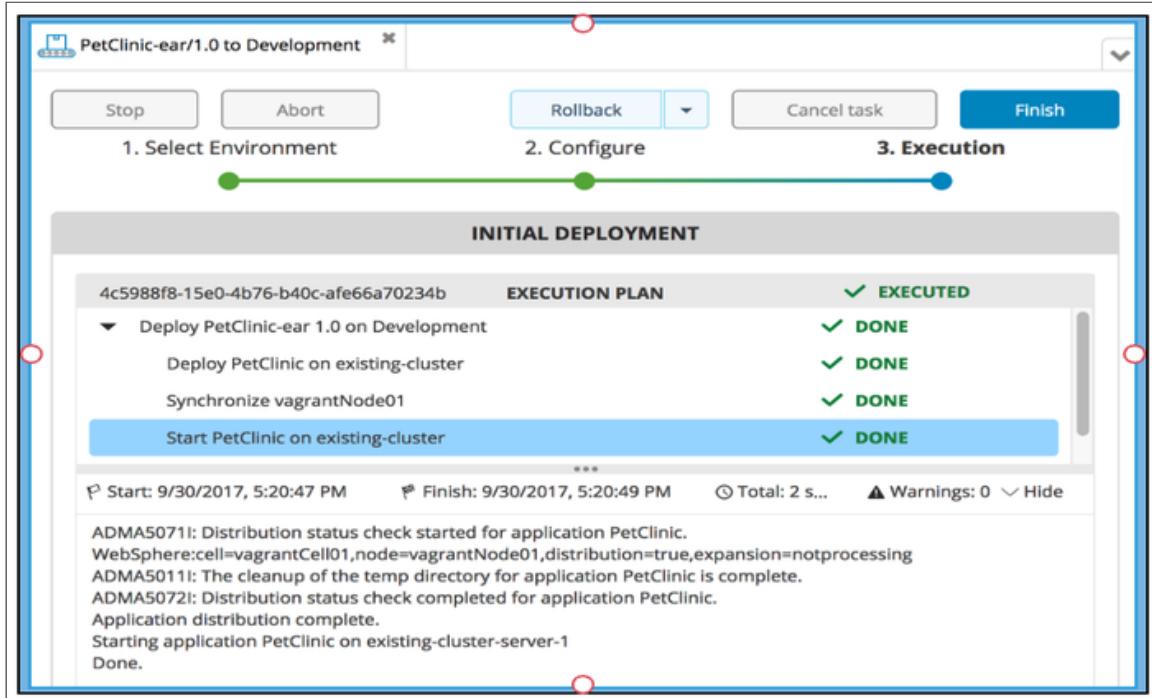
Deployment

Deployment execution



Deployment

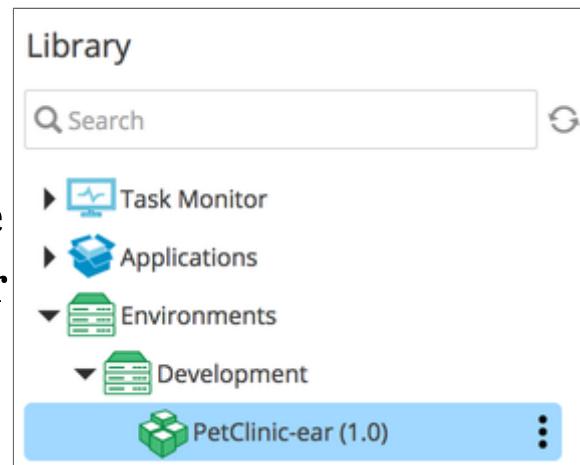
Deployment done



XL Deployment

Deployed application overview

- Which applications are installed in which environment
- Which components are running on a particular piece of infrastructure





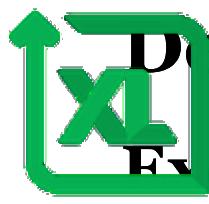
Deployment

Exercise: Initial deployment



Exercise: Initial deployment

- Deploy version 1.0 of the PetClinic-ear application to the Development environment
- Click the three dots to the right of **Applications**
 - **PetClinic-ear** ► **1.0**
- Select **Deploy**
- Select the Development environment and then **Continue**



Deployment

Exercise: Initial deployment

- Use the Plan Analyzer (**Preview** button) to review the generated plan
- Close the Plan Analyzer
- Click **Deploy** to perform the deployment

XL Deployment Exercise: Initial deployment verification

- check

<http://10.0.0.x:9082/petclinic/>

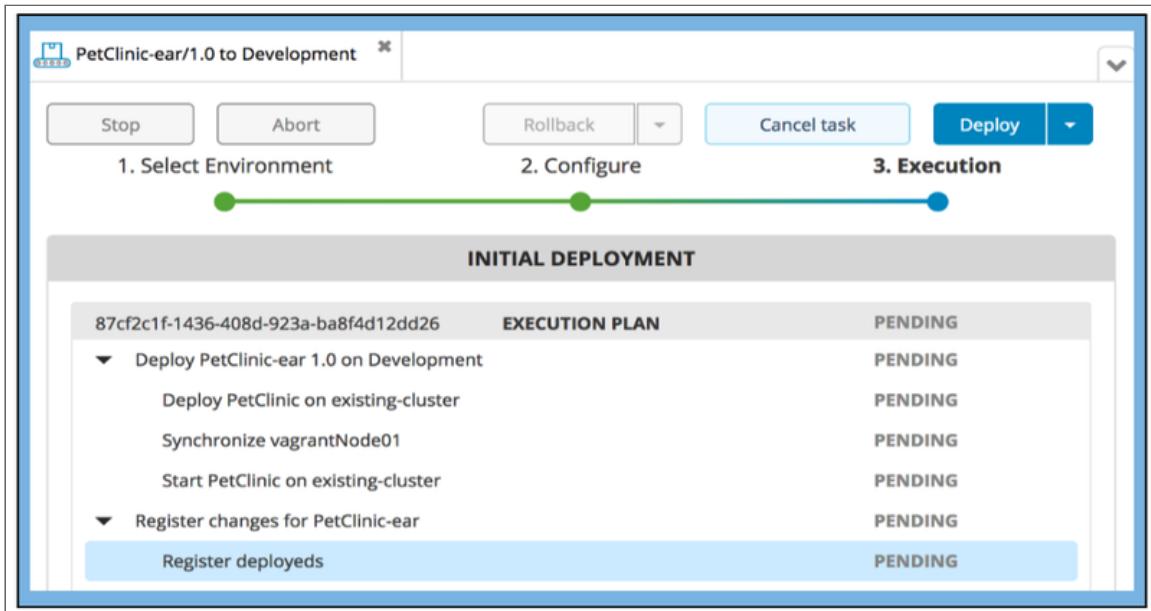




Tasks and steps

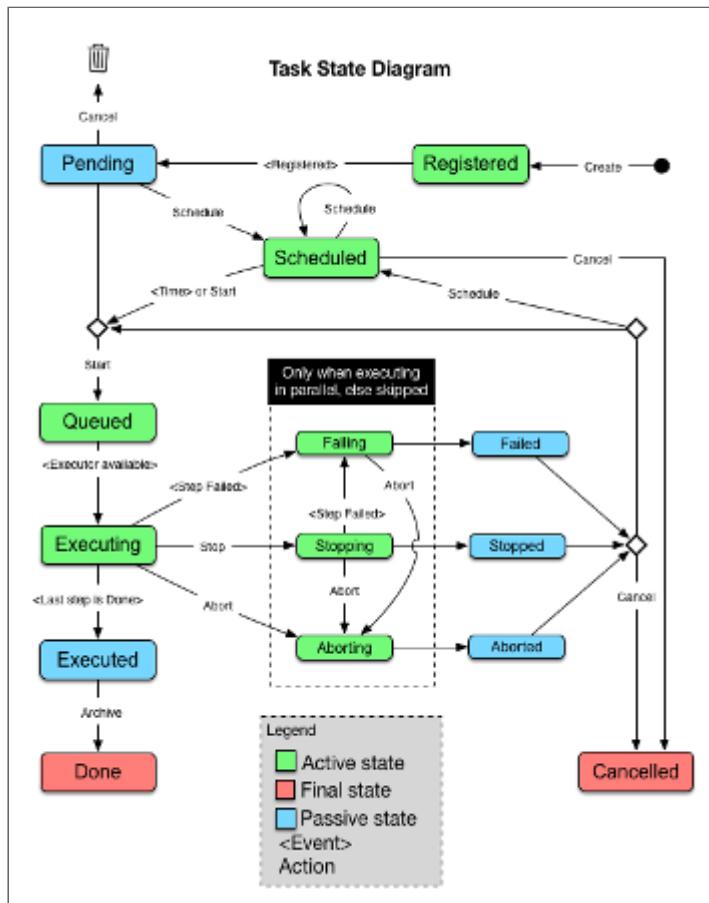
<http://www.xebialabs.com>

XL Tasks

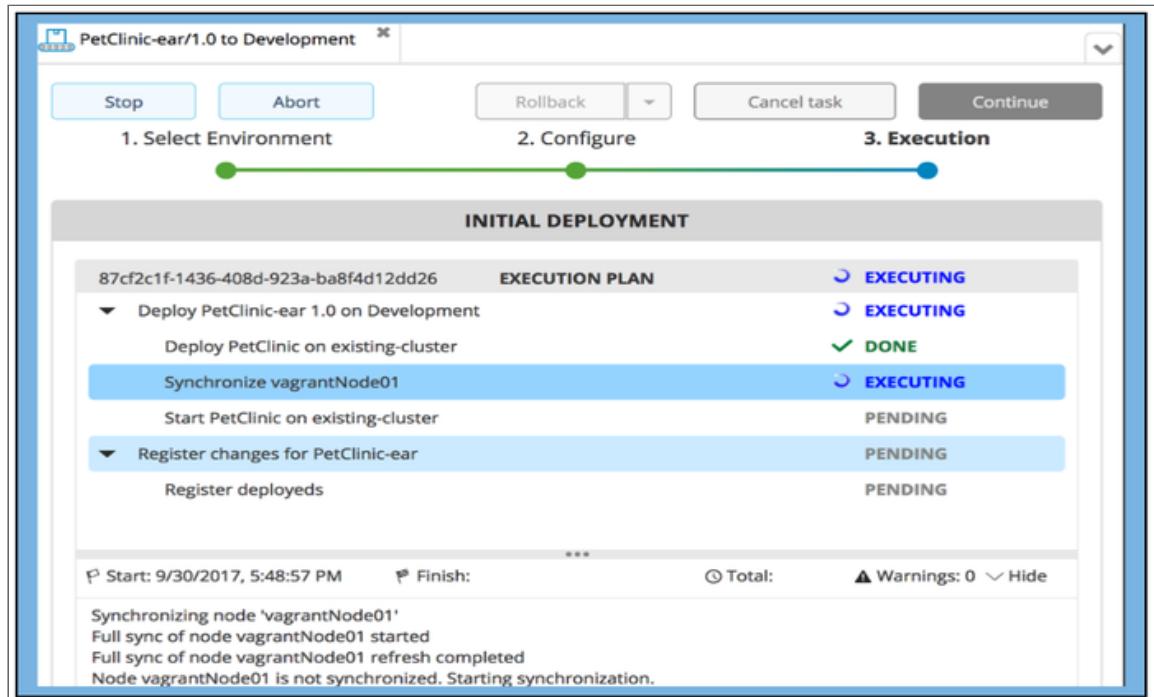




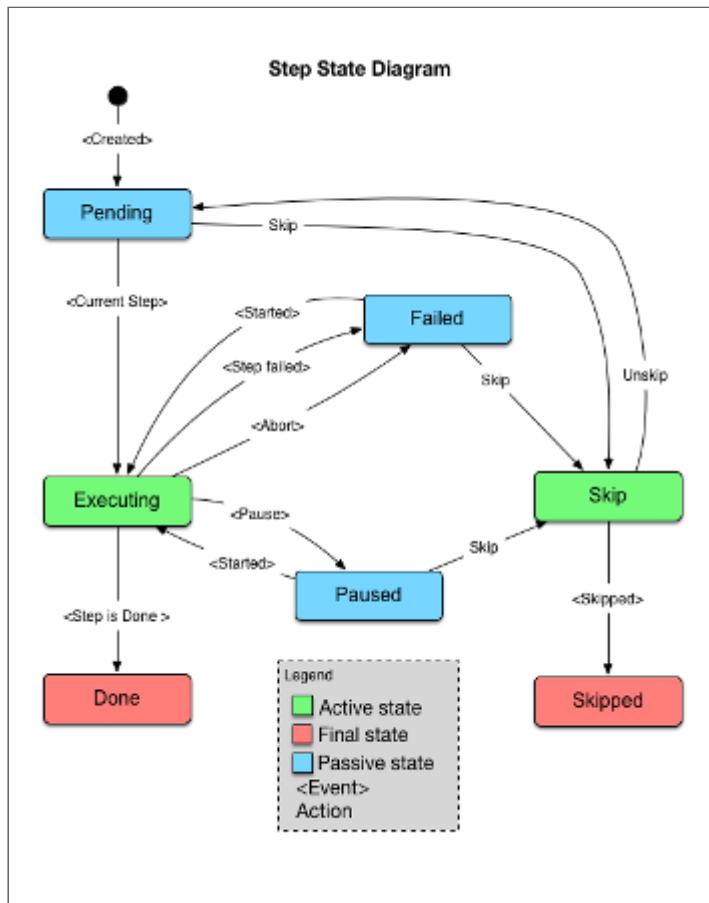
XL Tasks



XL Steps



XL Steps



XL Tasks and steps

Abort, Stop, Skip

- Stop: Waits until the current step finishes normally and stops executing the task
- Abort: Forcibly aborts the current step and stops executing the task
- Skip: Skip the step (engine considers step is executed)

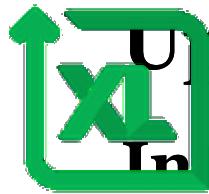
Deploy PetClinic on existing-cluster

— SKIPPED



Update

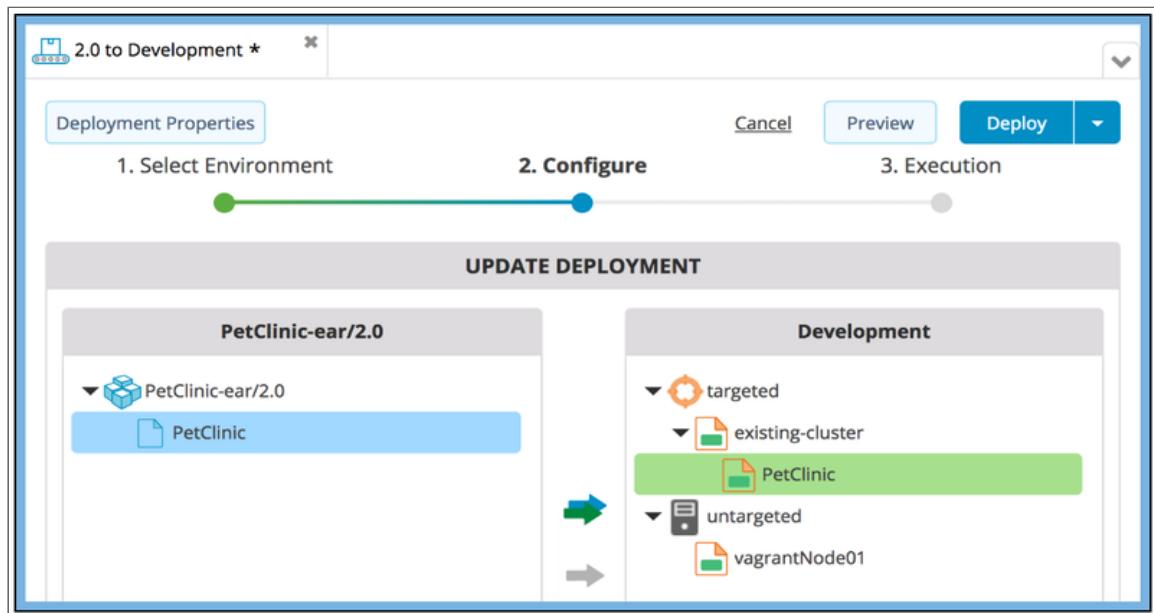
<http://www.xebialabs.com>



Update XL Introduction

- On a first deployment of an application, XL Deploy performs an **initial** deployment
- Applications can be **updated** to another version
- XL Deploy engine will calculate and deploy the deltas between package versions

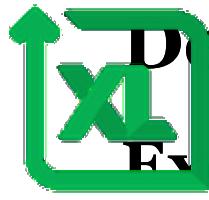
XL Update detection





Deployment

Exercise: Update deployment



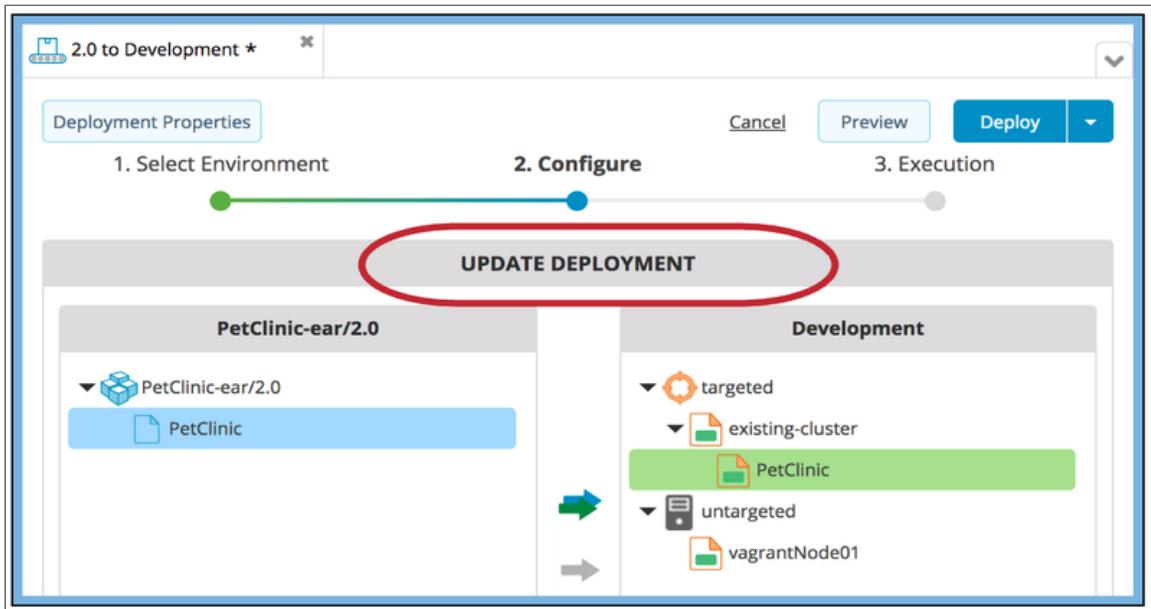
Deployment

Exercise: Update deployment

On the Deployment tab of the GUI:

- Find PetClinic-ear/1.0 under **Environments** in the Library section
- Click the three dots to the right of PetClinic/1.0 and select **Update deployment**
- Select the **2.0** version to deploy and then **Continue**
- XL Deploy automatically maps the updated components and reuses previous settings
- Click **Deploy** to start the update

XL Deployment Exercise: Update deployment





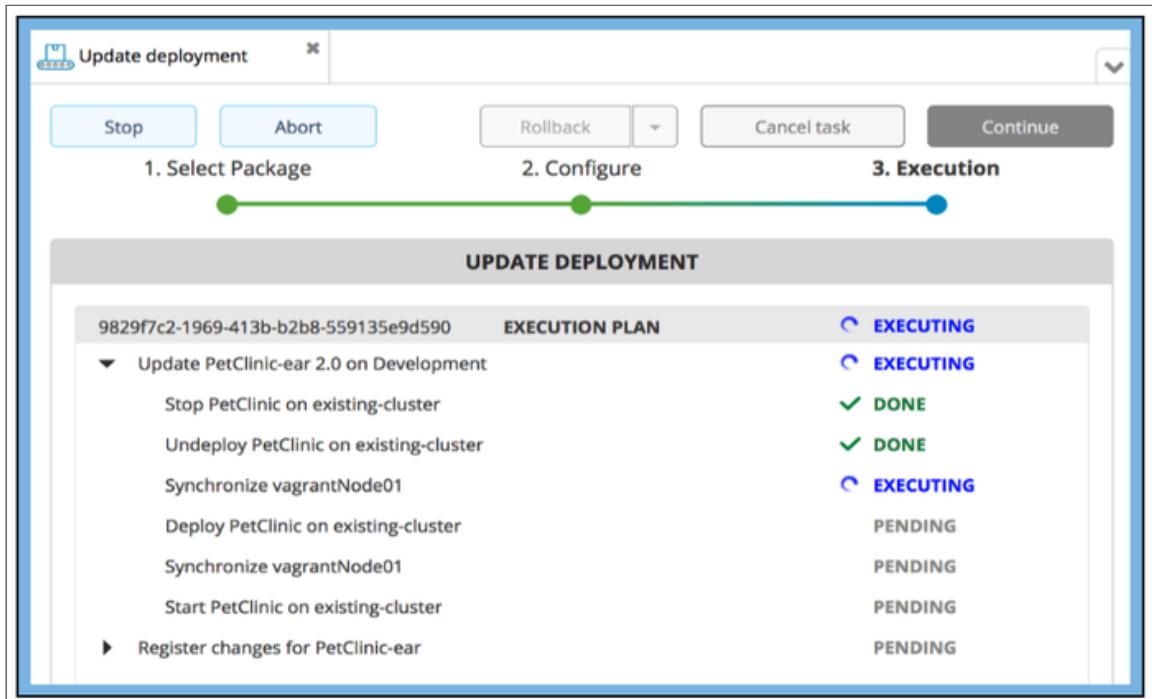
Deployment

Exercise: Update deployment

UPDATE DEPLOYMENT

EXECUTION PLAN	PENDING
9829f7c2-1969-413b-b2b8-559135e9d590	PENDING
▼ Update PetClinic-ear 2.0 on Development	PENDING
Stop PetClinic on existing-cluster	PENDING
Undeploy PetClinic on existing-cluster	PENDING
Synchronize vagrantNode01	PENDING
Deploy PetClinic on existing-cluster	PENDING
Synchronize vagrantNode01	PENDING
Start PetClinic on existing-cluster	PENDING
▶ Register changes for PetClinic-ear	PENDING

XL Deployment Exercise: Update deployment

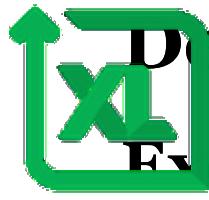


XL Deployment Exercise: Verify deployment

- check

<http://10.0.0.x:9082/petclinic/>





Deployment

Exercise: Undeploy application

On the **Library** section of the GUI:

- Find **PetClinic-ear (2.0)** under **Environments**
- Click on the three dots to the right of **PetClinic-ear (2.0)** and choose:
 - **Undeploy**
- Choose no **orchestrator** (more on Orchestrators later)
- Observe the generated steps by selecting the down arrow next to **Undeploy** and press **Undeploy** to start the undeployment



Deployment

Exercise: Undeploy application

UNDEPLOYMENT

ee1f8fc0-606e-4679-a318-cb576492ede1	EXECUTION PLAN	✓ EXECUTED
▼ Undeploy PetClinic-ear 2.0 on Development		✓ DONE
Stop PetClinic on existing-cluster		✓ DONE
Undeploy PetClinic on existing-cluster		✓ DONE
Synchronize vagrantNode01		✓ DONE
▼ Register changes for PetClinic-ear		✓ DONE
Register deployeds		✓ DONE



Create deployment package

<http://www.xebialabs.com>



Creating deployment package

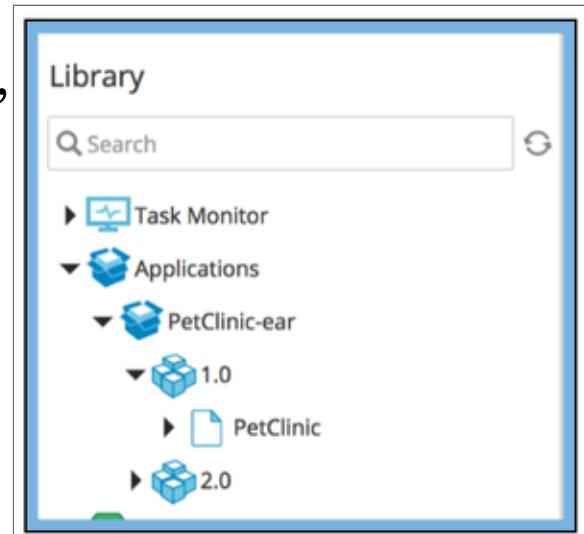
Recap



Creating deployment package

Viewing a package in the GUI

- Packages can be found in the **Library** section, under **Applications**
- To edit, double-click on the CI





Creating deployment package

Creating a package in the GUI

- To create a package, click on the three dots to the right of **Applications** in the Library section
- Alternatively, in the Library section of the page you can:
 - Click on the three dots to the right of **Applications** to create a new application or
 - Click on the three dots to the right of a specific application to add a new version



Creating deployment package

Adding artifacts and resources to a package

- To add an artifact or resource, click on the three dots to the right of the version number, select **New** and choose the type of artifact
- Files for artifact types (EAR, WAR, etc.) need to be uploaded, or you must specify an URL for an external location (such as a Maven repository)
- To add an artifact or resource:
 - From the Library section: click on the three dots to the right of the version and select the artifact or resource type

Creating deployment package

Adding package dependencies

You can add dependencies between specific versions of applications:

Application Dependencies	Key	Value	Remove
	MyAppA	4.0	x
	MyAppC	2.1.2	x

When you set up the deployment of the package, XL Deploy will check whether the dependencies are deployed to the target environment:

Error Logged in as admin

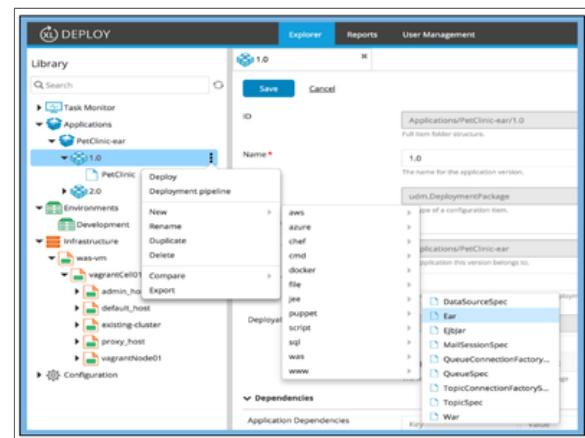
!

Error while trying to resolve the dependencies of application "Applications/PetClinic-ear". Cannot find an application with the name "MyAppA"

Creating deployment package

Creating a package in the GUI

- To add an artifact or resource, click on the three dots to the right of the deployment package and choose your artifact
- **New ► ...**





Creating deployment package

Exporting a package

- You can export any package from the Library
- Click on the three dots to the right of the package and select **Export**
- The package file is saved in DAR format (zipped)



Creating deployment package

Exercise: create a package



Creating deployment package

Exercise: Create a package

Create version 1.0 of the TableLister application:

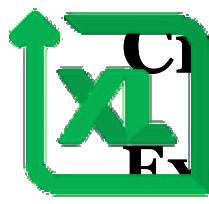
- Click the three dots to the right of **Applications**
- Select **New ▶ Application**
- Enter name: TableLister
- Select **Save**
- Click the three dots to the right of **Applications**
 ▶ **TableLister**
- Select **New ▶ Deployment Package**
- Name: 1.0
- Click **Save**



Creating deployment package

Exercise: create a package

- Add a **TableListingApp** of type **jee.Ear**
 - Click on the three dots to the right of version **1.0** of the **TableLister** application
 - choose: **New ▶ jee ▶ Ear**
 - Enter name: **TableListingApp**
 - In the **Choose File** section, upload the **HRListerEAR.ear** file from the **resources** folder of your training material
 - Press **Save** to start the actual upload of the artifact
- Add an **AppDataSource** of type **was.MySQLDatasourceSpec**
 - Click on the three dots to the right of version **1.0** of the **TableLister** application
 - Choose: **New ▶ was ▶ MySQLDatasourceSpec**
 - Enter values (next slide) and click the **Save** button



Creating deployment package

Exercise: create a package

Key	Value
Name	AppDataSource
Description	AppDataSource
Jndi Name	jdbc/mydatasource
Jdbc Provider	MySQL Provider
JDBC URL	jdbc:mysql://localhost:3306/mysql

continued on next slide



Creating deployment package

Exercise: create a package

Key	Value
Datasource	com.ibm.websphere.rsdadapter.Con
Helper	
Classname	
Username	root
Password	centos
Use Jaas Alias	true
For Container	
Managed	
Authentication	

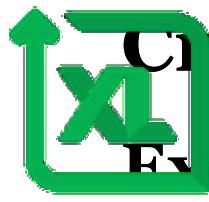


Creating deployment package

Exercise: create a package

On the Library section in the GUI:

- Double-click on **Infrastructure ▶ was-vm ▶ vagrantCello1 ▶ existing-cluster**
- In the **Deployment Strategy** section, **unchecked** the **Deploy To Running Container** checkbox
 - Note: Hover your cursor over the tab titles if they appear truncated
- Press **Save**



Creating deployment package

Exercise: create a package

On the Library section of the GUI:

- Prepare an initial deployment of version **1.0** of the **TableLister** application to the **Development** environment
- Apply the following mapping (you can drag-drop an application component to the desired target). Make sure the mapping looks like described below.
 - Map **TableListingApp** to **existing-cluster**
 - Map **AppDataSource** to **vagrantNode01**

Creating deployment package

Exercise: create a package

In the Deployment view:

- Double click on the deployed **TableListingApp** and change the container restart strategy from **None** to **Restart**
- Click the **Save** button to close the configuration window
- Click the **Deploy** button
- Once deployment completes, click the **Finish** button



Creating deployment package

Exercise: create a package

- check

<http://10.0.0.x:9082/hrlister/listtable>

Table Lister - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Table Lister +

< > C ⌂ 192.168.0.10:8080/tablelister-war/listtable ☆ ▾

Opened connection to MySql
Listing Tables...

Table Name
columns_priv
db
event
func
general_log
help_category
help_keyword
help_relation
help_topic



Creating deployment package

Exercise: create a package

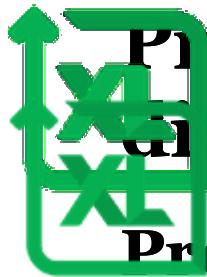
On the **Library** section in the GUI:

- Double click on: **Infrastructure ▶ was-vm ▶ vagrantCello1 ▶ existing-cluster**
- In the **Deployment Strategy** section, **check** the **Deploy To Running Container** checkbox
- Click the **Save** button



Property placeholders and dictionaries

<http://www.xebialabs.com>



Property placeholders and dictionaries

Property Placeholders

- Property placeholders are specified in the deployables.
- Use syntax: `{ {PLACEHOLDER} }`
- Placeholders are replaced when deployment is created.
- Placeholders are blank if no value is found.

XL Deploy Training

Property placeholders and dictionaries

Property Placeholders example

Create PetClinic-ds

Id	Applications/PetClinic-ear/1.0/PetClinic-ds
* Name	PetClinic-ds
* Type	jbossas.TransactionalDatasourceSpec

Common Deployment

Jndi Name	jdbc/petclinik-ds
Use Java Context	
User Name	<code>{{DB_USERNAME}}</code>
Password	*****

Property Placeholder

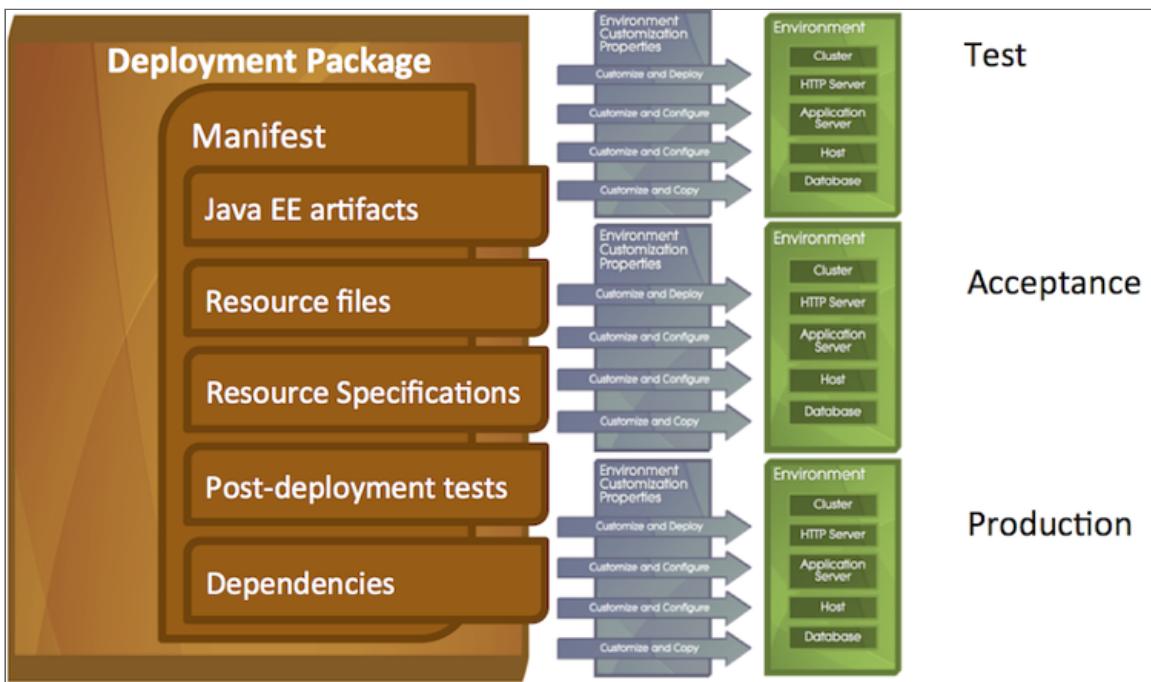


Property placeholders and Dictionaries

- Environment-specific values for placeholders
- Are linked to one or more environments
- Can be created and edited through GUI and CLI

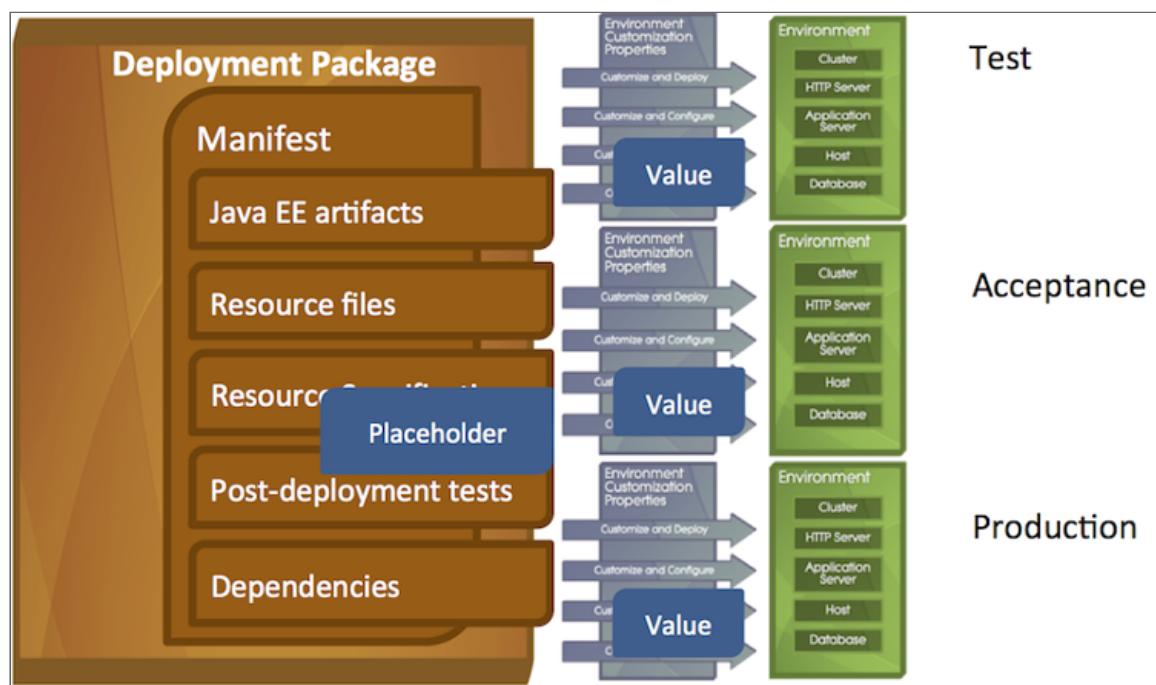
Property placeholders and dictionaries

Environment independent

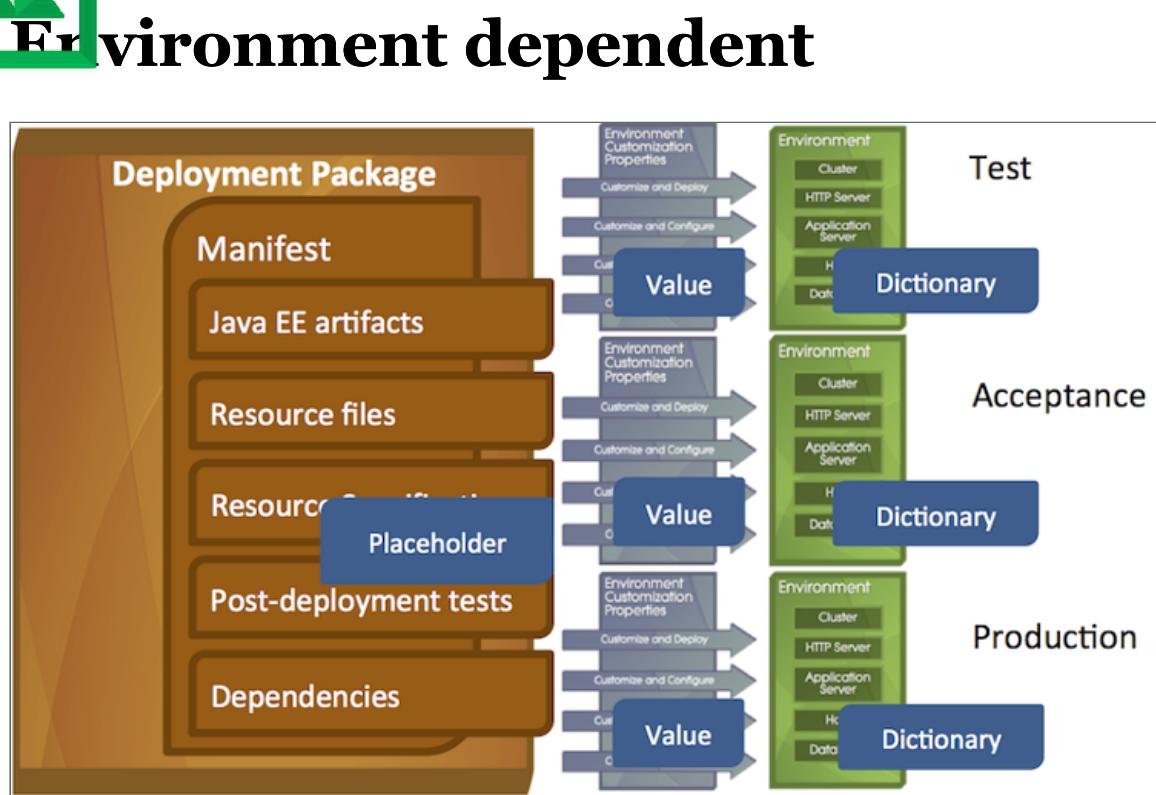


Property placeholders and dictionaries

Environment dependent



Property placeholders and dictionaries





Property placeholders and dictionaries

Placeholder and dictionaries

- Placeholders are resolved by searching the target environment's dictionaries
- If no value is found, the placeholder is blank
- Resolved placeholders can be overridden



Property placeholders and dictionaries

Composing values

- A dictionary value can refer to another dictionary entry. This is accomplished by using the {{ . }} placeholder syntax.
- Placeholders may refer to keys from any dictionary in the same environment
- The value can be also a placeholder that will be resolved too
- The value belonging to key MESSAGE will be "Welcome to XL Deploy!"

Key	Value
APP_NAME	XL Deploy
MESSAGE	Welcome to {{APP_NAME}}!



Property placeholders and dictionaries

Overriding values

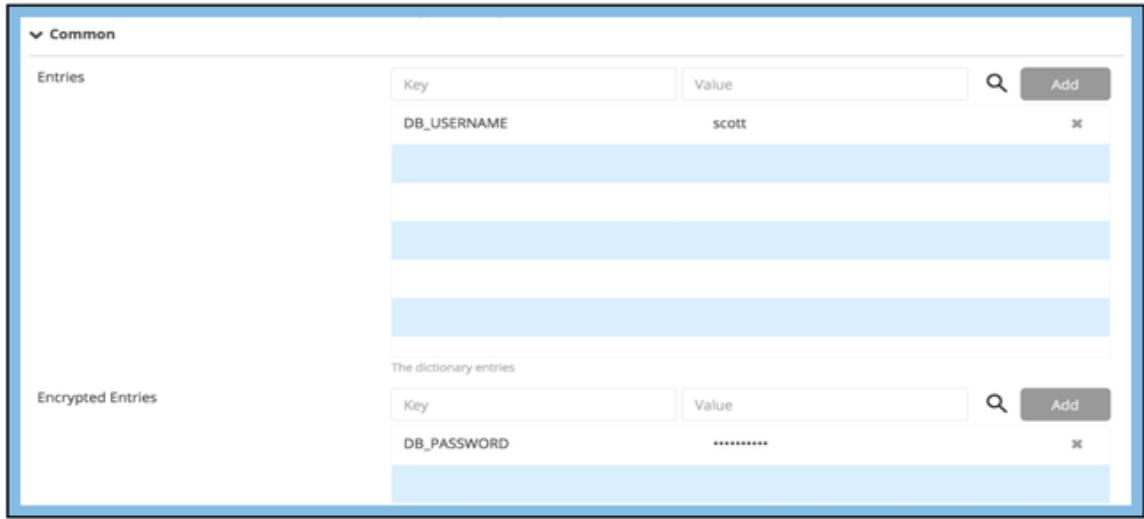
- Environment define dictionaries through an ordered list
- If a value is defined in 2 dictionaries, the **first one** will be picked up.
- If the environment refers [Dictionary1, Dictionary2], the value will be: 45
- If the environment refers [Dictionary2, Dictionary1], the value will be: 120

Dictionary	Key	Value
Dictionary1	Timeout	45
Dictionary2	Timeout	120

Property placeholders and dictionaries

Dictionary example

udm.Dictionary contains key-value pairs as well as keys with encrypted values, to store sensitive information such as passwords



The screenshot shows the 'Common' configuration screen for a dictionary. It has two main sections: 'Entries' and 'Encrypted Entries'.

Entries:

Key	Value
DB_USERNAME	scott

Encrypted Entries:

Key	Value
DB_PASSWORD



Property placeholders and dictionaries

Restrict to containers

Only apply this dictionary to the containers or applications mentioned

▼ Restrictions

Restrict to containers

Infrastructure/jboss-vm/development-domain

Only apply this dictionary to the containers mentioned

Restrict to applications

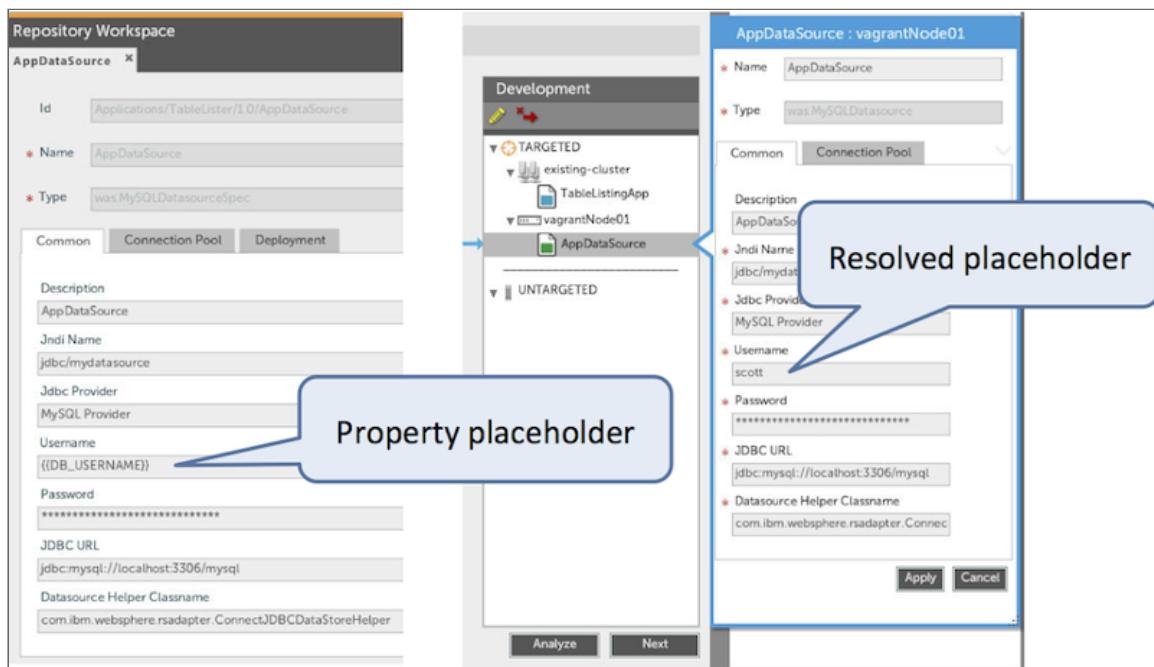
Applications/PetClinic-ear

Only apply this dictionary to the applications mentioned

XL Deploy Training

Property placeholders and dictionaries

Dictionary and deployeds example





Property placeholders and dictionaries

Exercise: Dictionary



Property placeholders and dictionaries

Exercise: dictionary

On the **Library** section of the GUI:

- Import application **JmsTester/1.0** from XL Deploy Server
- Open **AppQcf**, double-click on
 - **Applications ▶ JmsTester ▶ 1.0 ▶ AppQcf**
 - Add a placeholder for the **Bus name** field (**Common** tab) and click **Save**

Tab	Property	Value
Common	Bus Name	{BUS_NAME}

- Repeat this for (use same placeholder)

```
- **Applications ▶ JmsTester ▶ 1.0 ▶ AppQueueDestinati
```

```
◀ [REDACTED] ▶
```



Property placeholders and dictionaries

Exercise: dictionary

Define **AppQueue** as a **was.SibQueueSpec**

- Add the **JNDI Name**, **Description**, and **Time To Live** entries below into the Common section
- Note the placeholder for the “ **Time To Live** ” field
- Use the dropdown to choose the **Queue Destination**
- Click the **Save** button

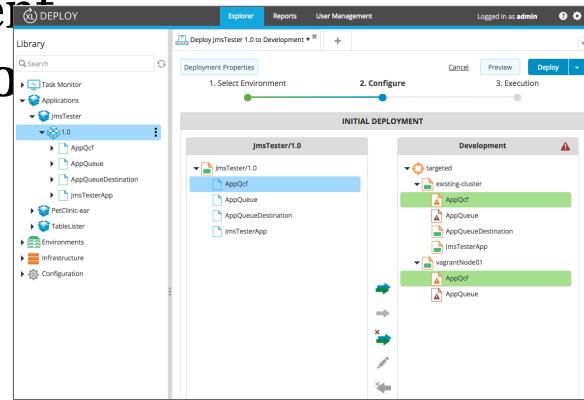
Tab	Property	Value
Common	Jndi Name	jms/Q.LQTest
Common	Description	JMS tester queue
Common	Time To Live	{TIME_TO_LIVE}

XL Deploy Training

Property placeholders and dictionaries

Exercise: dictionary

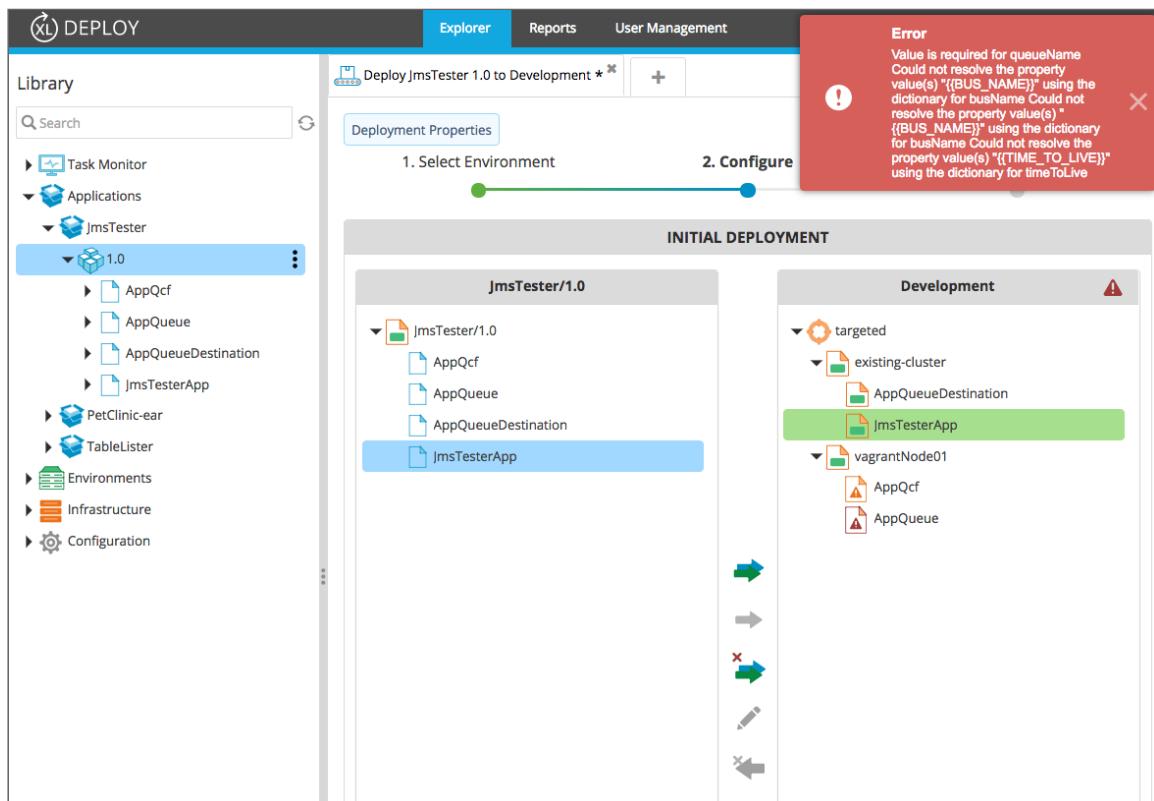
- Create a new deployment for the **JmsTester/1.0** application to the **Development** environment
- Map the **AppQueueDestination** and **JmsTesterApp** to the **existing-cluster**
- Map the **AppQcf** and **AppQueue** to the **vagrantNode01**
- Press **Deploy** to start the deployment



XL Deploy Training

Property placeholders and dictionaries

Exercise: dictionary

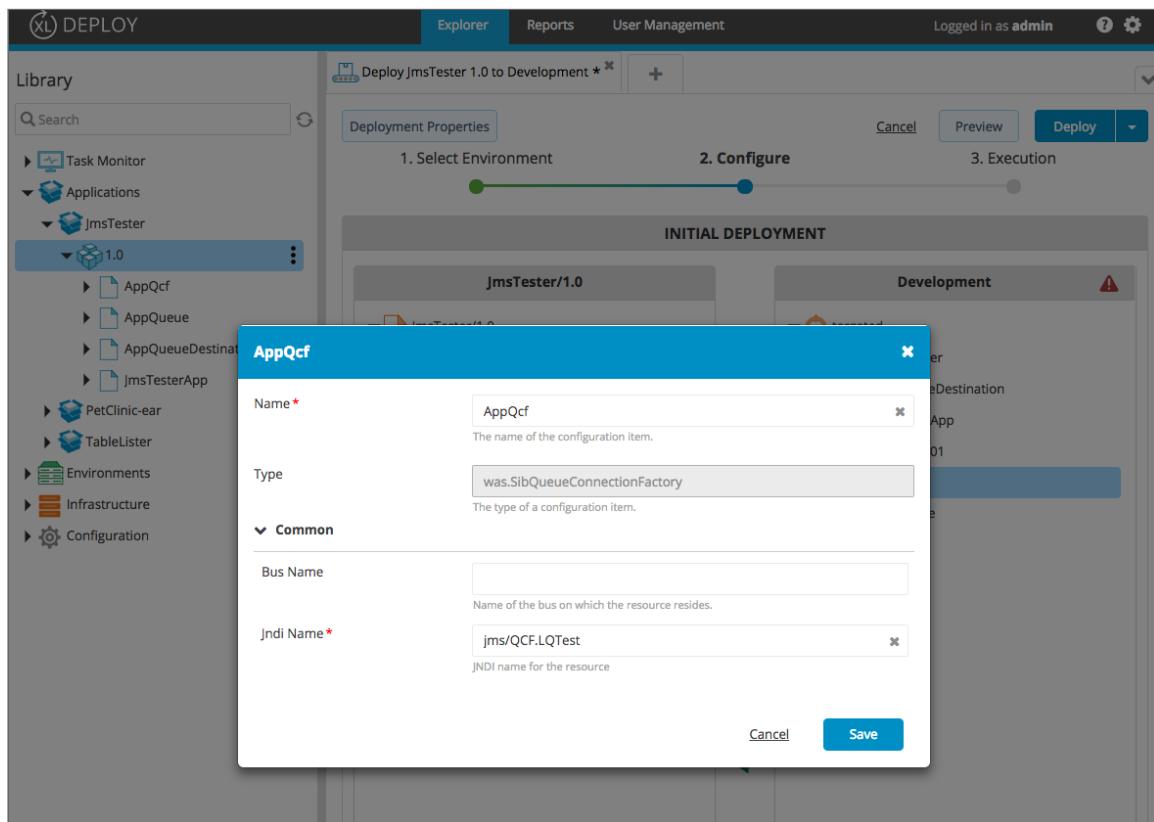




Property placeholders and dictionaries

Exercise: dictionary

Placeholder **BusName** isn't resolved





Property placeholders and dictionaries

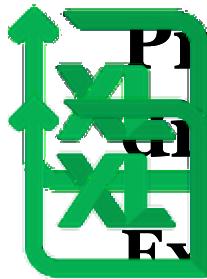
Exercise: dictionary

On the **Library** section of the GUI:

- Click on the three dots to the right of **Environments** and select **New ► Dictionary**
- Create a dictionary named **SibSettings** and populate it with the following values:

Key	Value
BUS_NAME	TrainingBus
TIME_TO_LIVE	60000

- Be sure to click the **Add** button after entering the values of each key
- Click the **Save** button



Property placeholders and dictionaries

Exercise: dictionary

Associate the **SibSettings** dictionary with the **Development** environment

On the **Environment section** of the GUI:

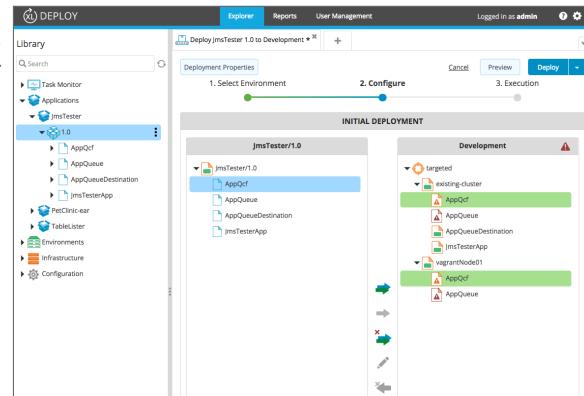
- Double-click on the **Development** environment
- In the **Common** section scroll down to the **Dictionaries** section
- Add the **SibSettings** dictionary as a member
- Don't forget to **Save** !

XL Deploy Training

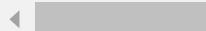
Property placeholders and dictionaries

Exercise: dictionary

Create a new deployment for the **JmsTester/1.0** application to the **Development** environment



- Press **Continue** to start the deployment
- Map the **AppQueueDestination** placeholder
- Map the **AppQcf** and **AppQueue** placeholders

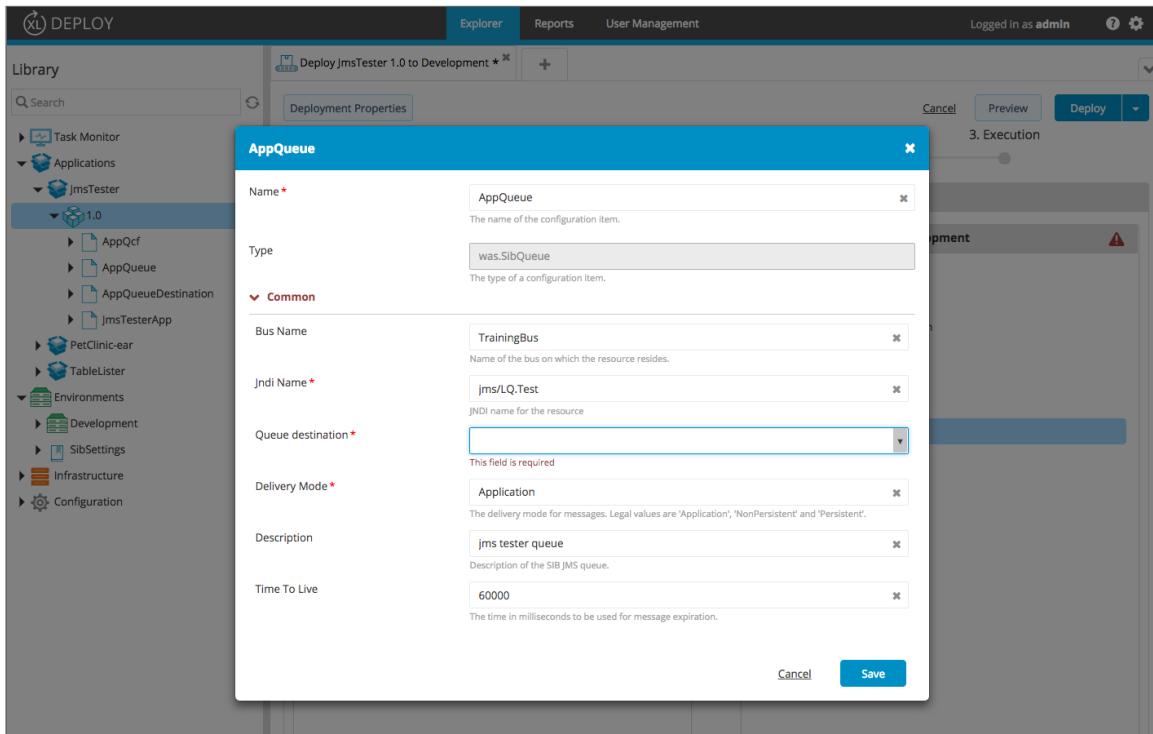




Property placeholders and dictionaries

Exercise: dictionary

Queue destination field is required!



The screenshot shows the XL Deploy interface with a configuration dialog for an 'AppQueue'. The dialog contains the following fields:

- Name: AppQueue
- Type: was.SibQueue
- Bus Name: TrainingBus
- Jndi Name: jms/LQ.Test
- Queue destination: (empty field with red border)
- Delivery Mode: Application
- Description: jms tester queue
- Time To Live: 60000

An error message 'This field is required' is displayed next to the empty 'Queue destination' field. The 'Save' button is located at the bottom right of the dialog.



Property placeholders and dictionaries

Exercise: dictionary

In the **AppQueue** configuration window

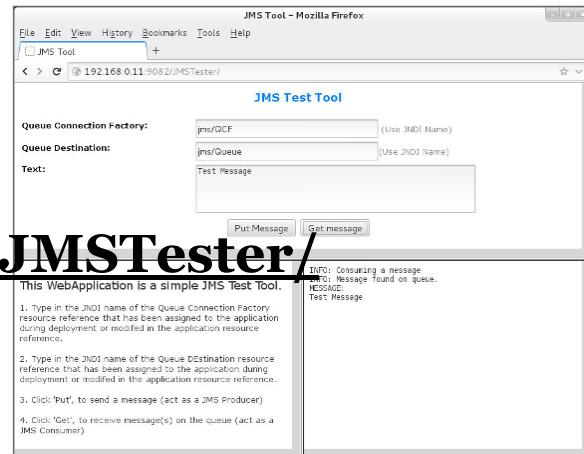
- Select a value for “**Queue destination**” From the drop-down box select:
AppQueueDestination
 - Press **Apply**
- Press **Next** and **Execute** to deploy the **JmsTester/1.0** application
- Use the application to send/receive some messages



Property placeholders and dictionaries

Exercise: dictionary

To verify your deployment, open your browser and go to
<http://10.0.0.x:9082/JMSTester>

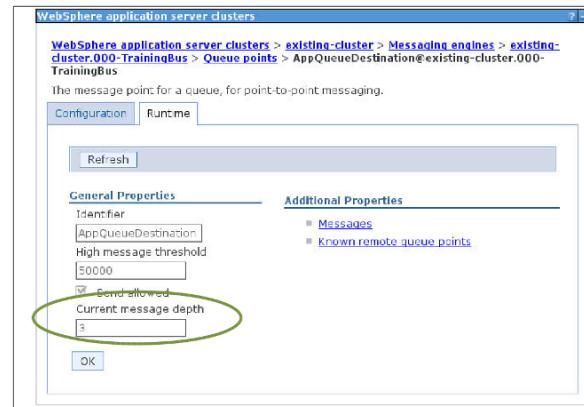


XL Deploy

Property placeholders and dictionaries

Exercise: dictionary

Verify from WebSphere console that the queue's message count value increases with every sent message.





Property placeholders and dictionaries

Exercise: dictionary

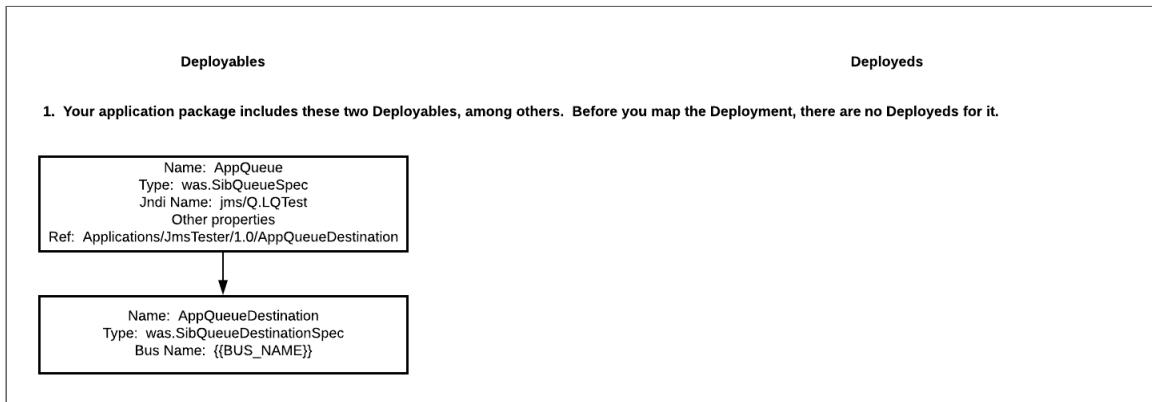
- In the WebSphere console, check if the dictionary values are correctly set
- For example:
 - **Resources ▶ JMS ▶ Queues ▶ AppQueue**
 - **Resources ▶ JMS ▶ Queue connection factories ▶ AppQcf**



Property placeholders and dictionaries

Exercise: dictionary

An explanation of why AppQueue is flagged with an error regarding QueueDestination





Property placeholders and dictionaries

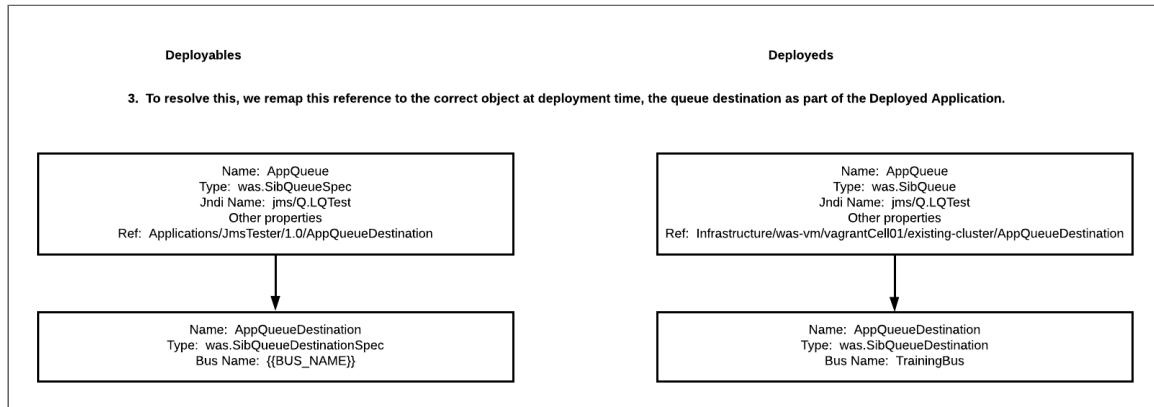
Exercise: dictionary

Deployables	Deployeds
<p>2. The deployment will create Deployeds corresponding to the Deployables as it maps the application to the environment.</p> <p>The values on the Deployeds are copied from the Deployables, if possible. The type is changed and placeholders are resolved.</p> <p>But notice that the reference from AppQueue to AppQueueDestination still points to the Deployable, not the Deployed (red arrow).</p> <p>This violates the definition in the type system, so XL Deploy flags it as an error.</p> <pre>graph TD; Deployable[Deployable: AppQueue
Type: was.SibQueueSpec
Jndi Name: jms/Q.LQTest
Other properties
Ref: Applications/JmsTester/1.0/AppQueueDestination] --> Deployed1[Deployed: AppQueueDestination
Type: was.SibQueueDestinationSpec
Bus Name: {{BUS_NAME}}]; Deployable --> Deployed2[Deployed: AppQueue
Type: was.SibQueue
Jndi Name: jms/Q.LQTest
Other properties
Ref: Applications/JmsTester/1.0/AppQueueDestination]; Deployed2 --> Deployed1</pre>	



Property placeholders and dictionaries

Exercise: dictionary





Deployment with web servers

<http://www.xebialabs.com>



Deployments with web servers

Exercise: Deploy with web server

Deployments with web servers

Exercise: Deploy with web server

On the **Library** section of the GUI:

- Open the Configuration Item **Infrastructure/webserver/vm/vagrantCello1/vagrantNode01/webservice**
- Enable **Restart Required**
- Click **Save**

Deployments with web servers

Exercise: Deploy with web server

On the **Library** section of the GUI

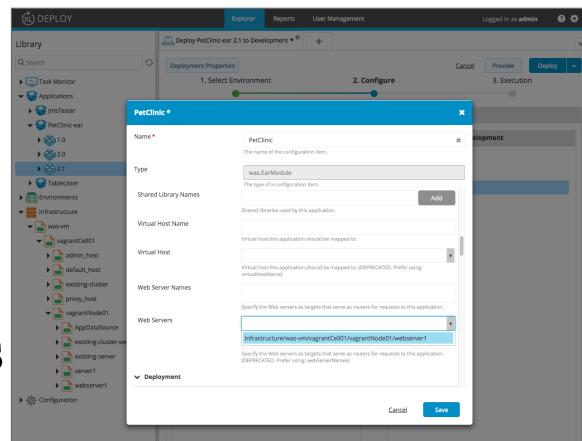
- Under the **Applications**:
 - Click on the three dots to the right of **PetClinic-ear ▶ 2.1** and select **Deploy**
 - Select the **Development** environment
 - Click the **Continue** button to generate the default deployeds (but do **not** deploy yet!)

XL Deploy Trainings

Deployments with web servers

Exercise: Deploy with web server

- We need to select the **Webserver** to use for this application
- Double-click on the **PetClinic** deployed under **Environments**
- In the **Common** tab, scroll down to the **Web Server Names** area
- Add **webserver1** from the list
- Click on **Save** in the upper left corner





Deployments with web servers

Exercise: Deploy with web server

Inspect the generated deployment plan steps by clicking on the **Preview** button, but do **not** start the actual deployment yet!

PREVIEW

- ▼ Deploy PetClinic-ear 2.1 on Development
 - Deploy PetClinic on existing-cluster
 - Synchronize vagrantNode01
 - Stop web server 'webserver1'
 - Start web server 'webserver1'
 - Start PetClinic on existing-cluster
- ▼ Register changes for PetClinic-ear
 - Register deployeds

Deployments with web servers

Exercise: Deploy with web server

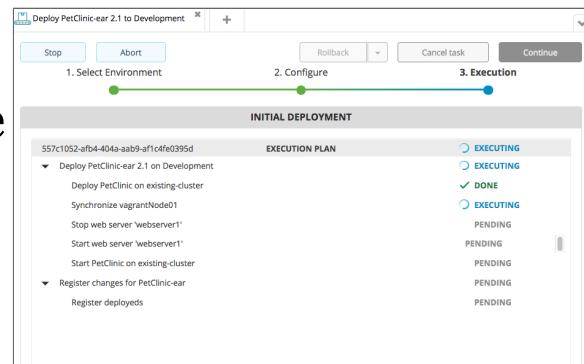
- Open the following URL in your browser:
http://10.0.0.x/petclinic/



Deployments with web servers

Exercise: Deploy with web server

In the XL Deploy UI,
press **Deploy** to start the
actual deployment



XL Deploy

Deployments with web servers

Exercise: Deploy with web server

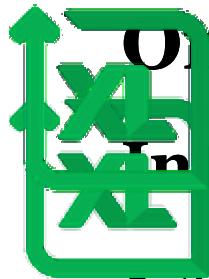
When the deployment is finished, press the reload button in your browser screen/tab which previously had the 404 error





Orchestrators

<http://www.xebialabs.com>



Orchestrators

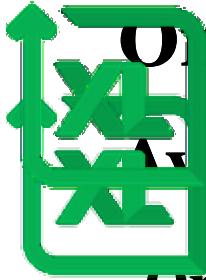
Introduction

- An orchestrator in XL Deploy combines the steps for the individual component changes into an overall deployment workflow
- Can be selected:
 - Before deployment or undeployment using the Deployment Properties button on the deployment workspace or
 - In the deployment package
- Can be predefined in the deployment package itself

Orchestrators

Orchestrator selection

The screenshot shows the XL Deploy interface with a green decorative element on the left. The main window has a header with tabs: Explorer, Reports, User Management, and a user status. The Explorer tab is active. On the left, a tree view shows categories like Task Monitor, Applications, Environments, Infrastructure, and Configuration. Under Applications, there's a PetClinic-ear application with versions 1.0 and 2.0, and a 2.1 branch which is selected. The right panel displays a configuration dialog for a deployment package named 'Applications/PetClinic-ear/2.1'. It includes fields for ID, Name, Type (set to 'udm.DeploymentPackage'), Application (set to 'Applications/PetClinic-ear'), Orchestrator (dropdown menu showing options like 'parallel-by-composite-package', 'parallel-by-container', 'parallel-by-dependency' - which is selected and highlighted in blue), Deployables (list of deployment strategies), and Dependencies (application dependencies). A dependency resolution dropdown is set to 'LATEST'.

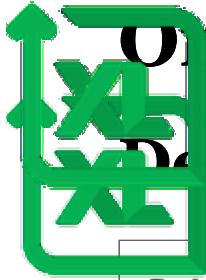


Orchestrators

Available Orchestrator

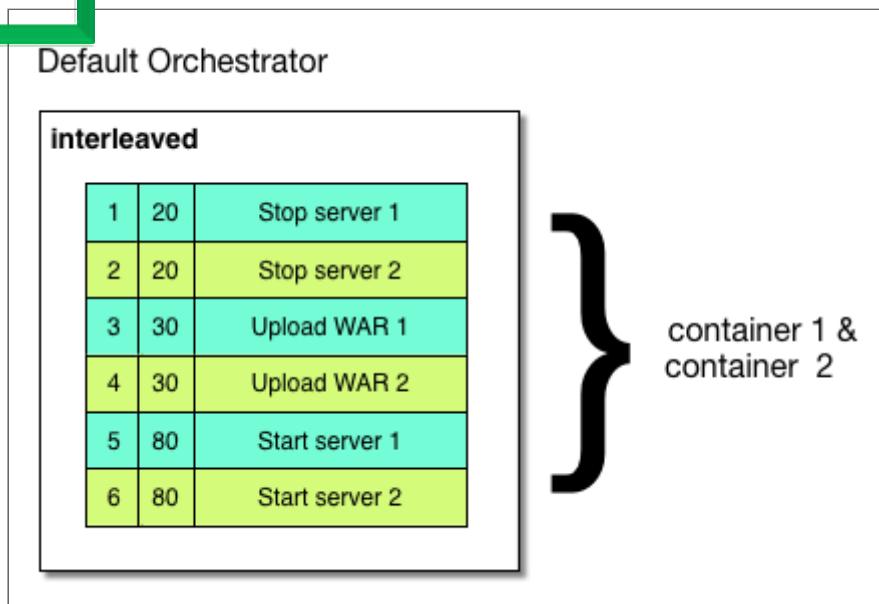
Orchestrator	Description
default	Default interleaved orchestrator
sequential-by-container, parallel-by-container	To containers, sequentially or in parallel
sequential-by-composite- package, parallel-by- composite-package	To members of composite package, sequentially or in parallel
sequential-by- deployment-group, parallel-by-deployment- group	To group members, sequentially or in parallel
sequential-by- deployment-sub-group, parallel-by-deployment- sub-group	To sub-group members, sequentially or in parallel
sequential-by- deployment-sub-sub- group, parallel-by- deployment-sub-sub- group	To sub-sub-group members, sequentially or in parallel





Orchestrators

Default Orchestrator

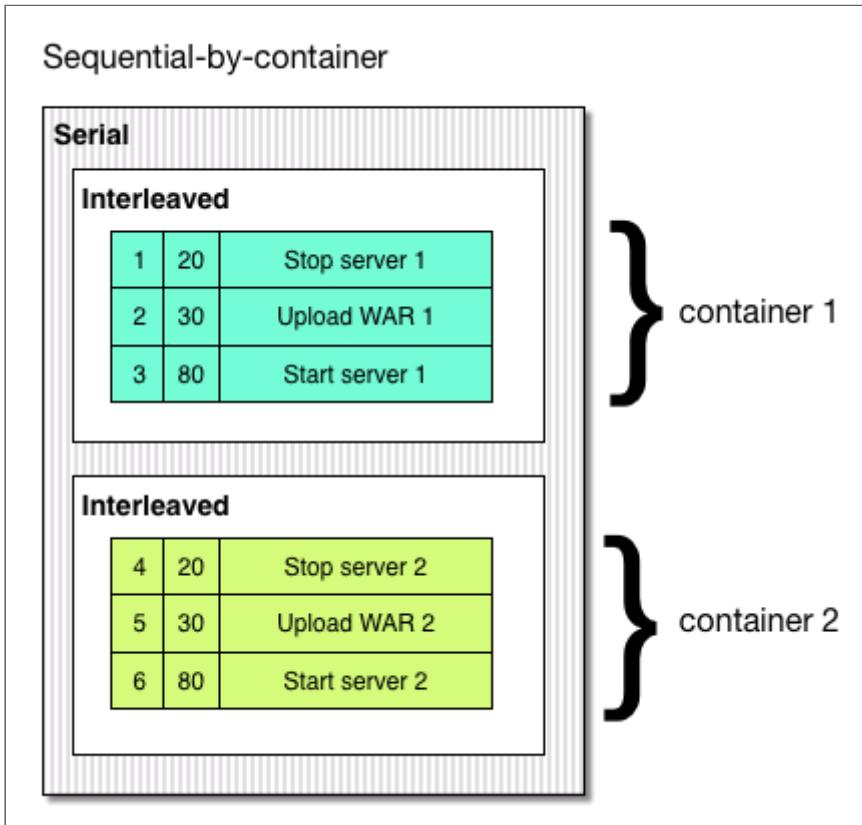




Orchestrators

Orchestrator by-container

Alphabetical names of containers





Orchestrators

Orchestrator by-container

Parallel-by-container

Parallel

Interleaved

1	20	Stop server 1
2	30	Upload WAR 1
3	80	Start server 1

Interleaved

4	20	Stop server 2
5	30	Upload WAR 2
6	80	Start server 2



container 1



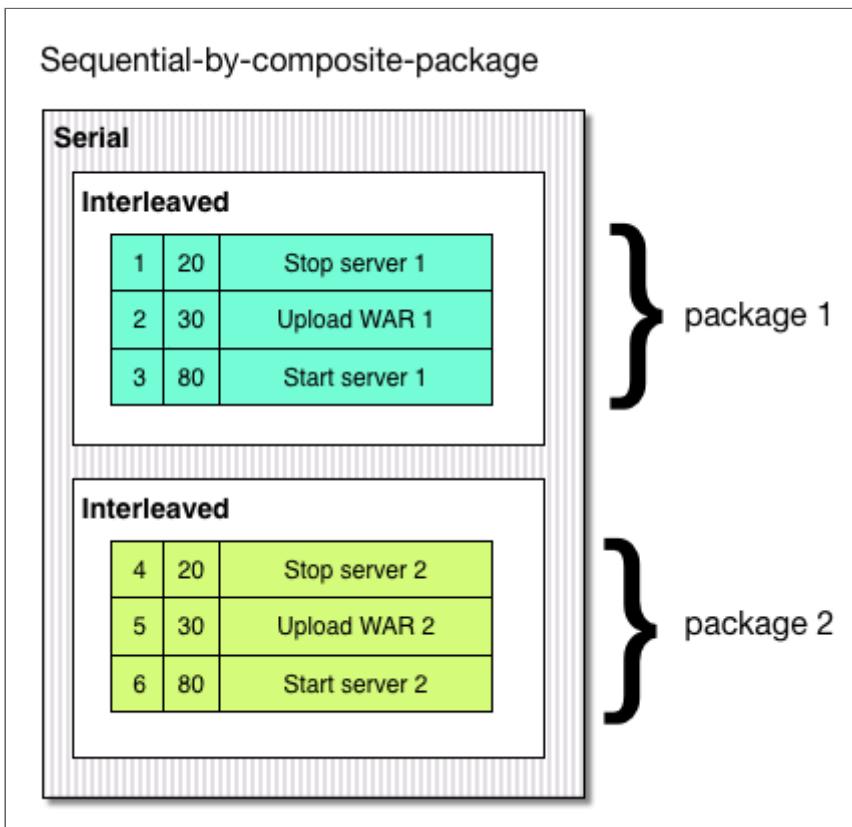
container 2



Orchestrators

Orchestrator by-package

Order of the package, as they appear in the composite package



Orchestrators

Orchestrator by-package

Parallel-by-composite-package

Parallel

Interleaved

1	20	Stop server 1
2	30	Upload WAR 1
3	80	Start server 1

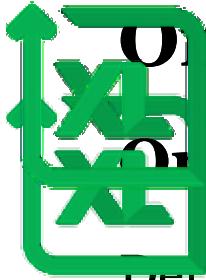
Interleaved

4	20	Stop server 2
5	30	Upload WAR 2
6	80	Start server 2



package 1

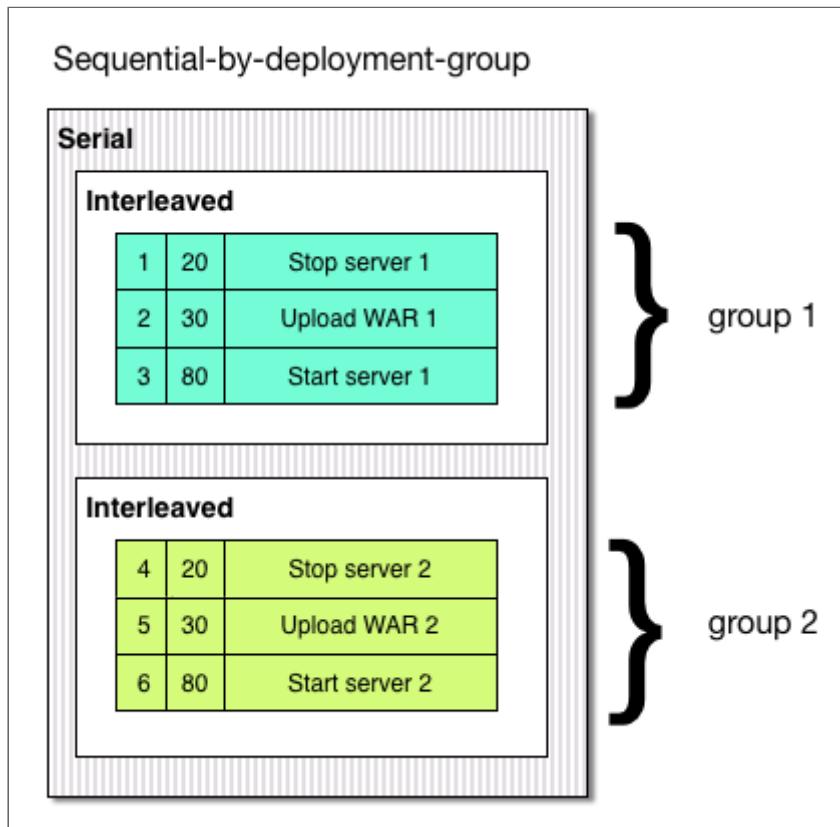
package 2



Orchestrators

Orchestrator by-group

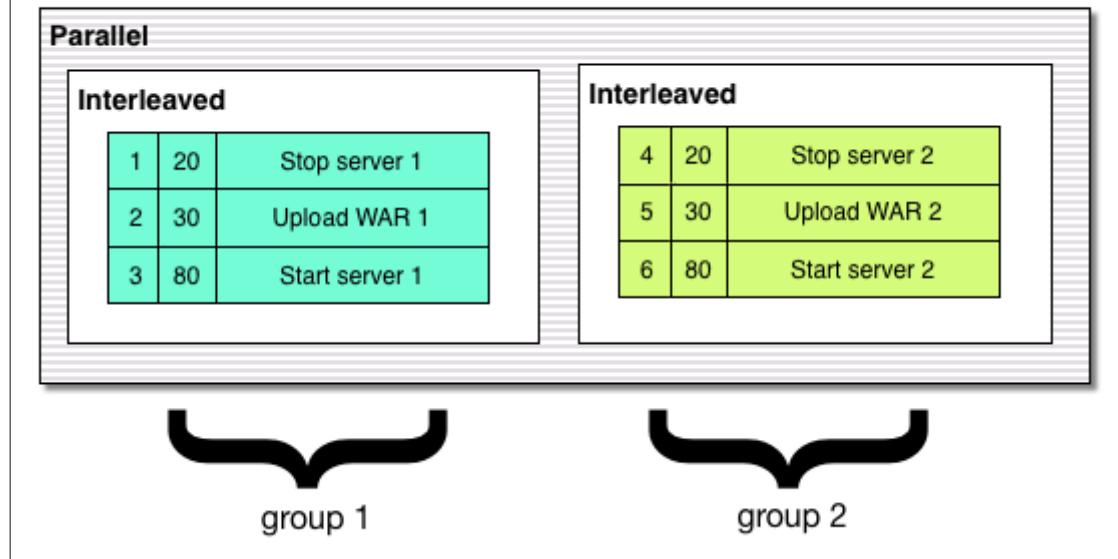
Deployment group number of containers

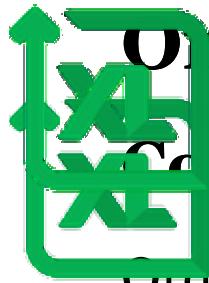


Orchestrators

Orchestrator by-group

Parallel-by-deployment-group



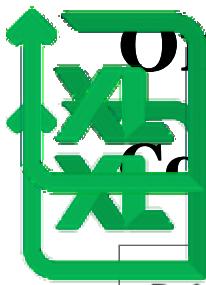


Orchestrators

Combining orchestrators

Order does matter

- Recursion
- Two are enough



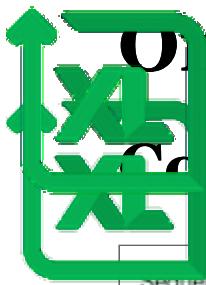
Orchestrators

Combining orchestrators

Default

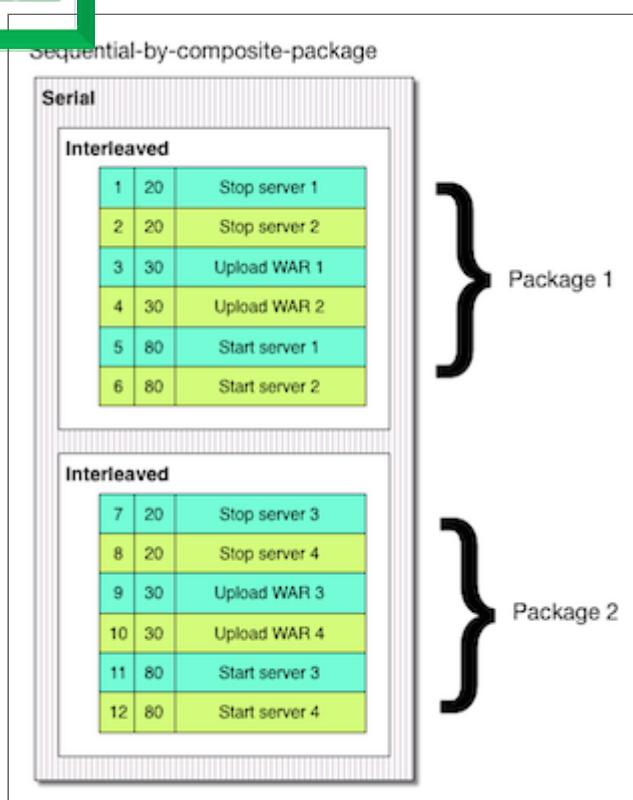
Interleaved

1	20	Stop server 1
2	20	Stop server 2
3	20	Stop server 3
4	20	Stop server 4
5	30	Upload WAR 1
6	30	Upload WAR 2
7	30	Upload WAR 3
8	30	Upload WAR 4
9	80	Start server 1
10	80	Start server 2
11	80	Start server 3
12	80	Start server 4



Orchestrators

Combining orchestrators

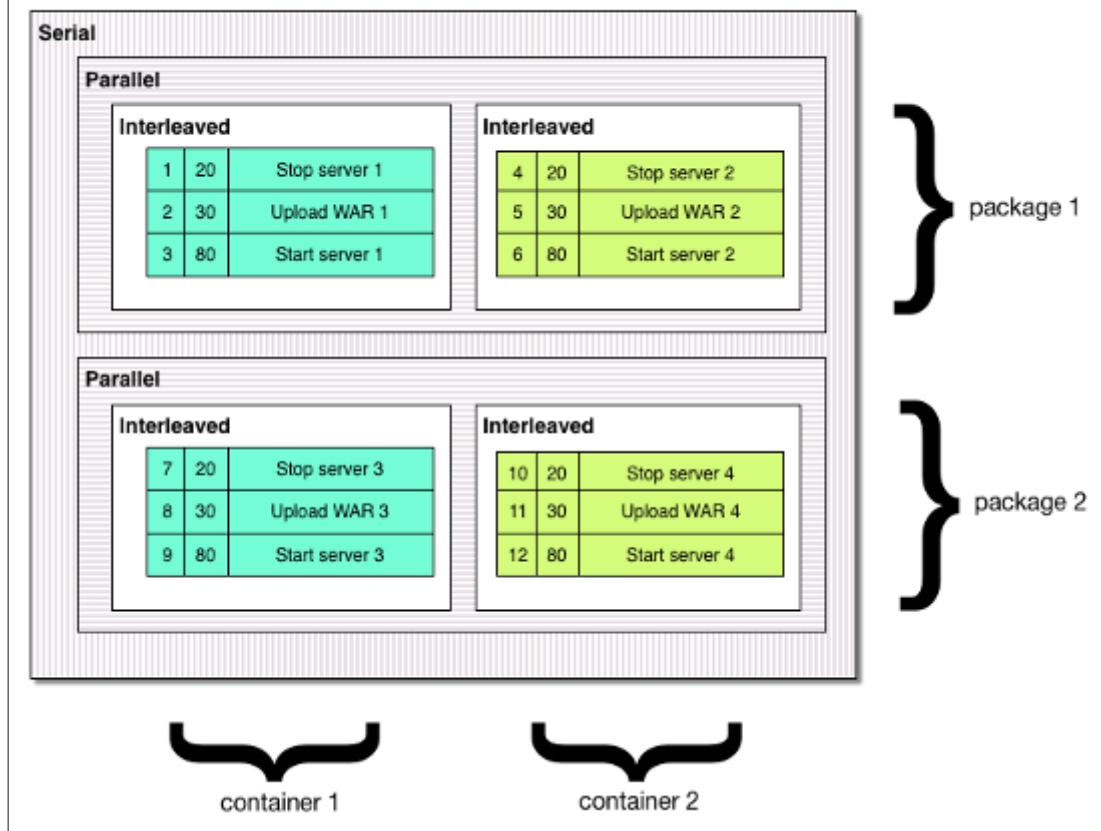




Orchestrators

Combining orchestrators

Sequential-by-package and parallel-by-container





Orchestrators

Exercise: Deploy with orchestrators



Orchestrators

Exercise: Deploy with orchestrators

- In the CLI, run **setupOrchestratorExercise()**
- In the **Library** section of the GUI
 - Inspect the newly created **vagrantCello2** and **vagrantCello3** topologies (under **Infrastructure/was-vm**) and the **Test** environment
- Compose an **initial** deployment of **PetClinic-ear/2.1** to **Test** (Don't forget to specify a **WebServer** for both **PetClinic** applications)
 - **server2** ► **webserver2**
 - **server3** ► **webserver3**
- Generate the default deployeds
- Click on the **Preview** button to inspect the steps (do not execute yet)



Orchestrators

Exercise: Deploy with orchestrators

- Select an alternative Orchestrator
- Click on the **Deployment Properties** button
 - Choose the **sequential-by-container** orchestrator
- Inspect the order of the steps
- Close the plan analyzer and deployment



Orchestrators

Exercise: Deploy with orchestrators

- default

PLAN ANALYZER

- Stop managed server 'server3'
- Stop managed server 'server2'
- Deploy PetClinic on server3
- Deploy PetClinic on server2
- Synchronize vagrantNode02
- Synchronize vagrantNode03
- Start managed server 'server3'
- Start managed server 'server2'
- Start PetClinic on server3
- Start PetClinic on server2



Orchestrators

Exercise: Deploy with orchestrators

- Sequential-by-container

PLAN ANALYZER

- ▼ Deploying on container server2
 - Stop managed server 'server2'
 - Deploy PetClinic on server2
 - Synchronize vagrantNode02
 - Start managed server 'server2'
 - Start PetClinic on server2
- ▼ Deploying on container server3
 - Stop managed server 'server3'
 - Deploy PetClinic on server3
 - Synchronize vagrantNode03
 - Start managed server 'server3'
 - Start PetClinic on server3



Orchestrators

Exercise: Deploy with group-based orchestrators

- Create a plan that deploys **PetClinic-ear/2.1** in the following order to **Test**:
 - server2/webserver2
 - server3/webserver3
- Use the **group-based** orchestrator in combination with the **deployment group number** property of the used containers

Hint : the deployment group number groups deployables and can be found in the deployment tab of the appropriate server. Deployments are executed in the numerical order of the deployment group they are member of.



Orchestrators

Exercise: Deploy with group-based orchestrators

- default

PLAN ANALYZER

- Stop managed server 'server3'
- Stop managed server 'server2'
- Deploy PetClinic on server3
- Deploy PetClinic on server2
- Synchronize vagrantNode02
- Synchronize vagrantNode03
- Start managed server 'server3'
- Start managed server 'server2'
- Start PetClinic on server3
- Start PetClinic on server2



Exercise: Deploy with group-based orchestrators

- group

PLAN ANALYZER

- ▼ Deploying on container server2 (group 2)
 - Stop managed server 'server2'
 - Deploy PetClinic on server2
 - Synchronize vagrantNode02
 - Start managed server 'server2'
 - Start PetClinic on server2
- ▼ Deploying on container server3 (group 3)
 - Stop managed server 'server3'
 - Deploy PetClinic on server3
 - Synchronize vagrantNode03
 - Start managed server 'server3'
 - Start PetClinic on server3



Agenda

<http://www.xebialabs.com>



Agenda

XL Deploy Training

Chapter	Description
Staging	Optimize deployments plan execution
Tags	Controlling deployable to container mapping
File placeholders	Variables in artifacts



Agenda

XL Deploy Training

Chapter	Description
Release dashboards	Controlling application promotion
Reports	Reporting features
Security	Managing roles and ACL. LDAP configuration
CLI	Using the command-line interface
REST	Introduction to XL Deploy REST API



Agenda

XL Deploy Training

Chapter	Description
Configuration	XL Deploy configuration
Compare feature	Live-to-live and repo-to-live comparisons
Community plugins	Introduction to community plugins
Satellite	How to deploy to multiple datacenters
Compare	How to compare two different CIs
Continuous Integration	Integration with Maven, Jenkins, Puppet, TFS



Staging

<http://www.xebialabs.com>



Staging

Introduction

- Copies artifacts that will be deployed to the host before executing the real deployment
- Ensures downtime of your application is minimized

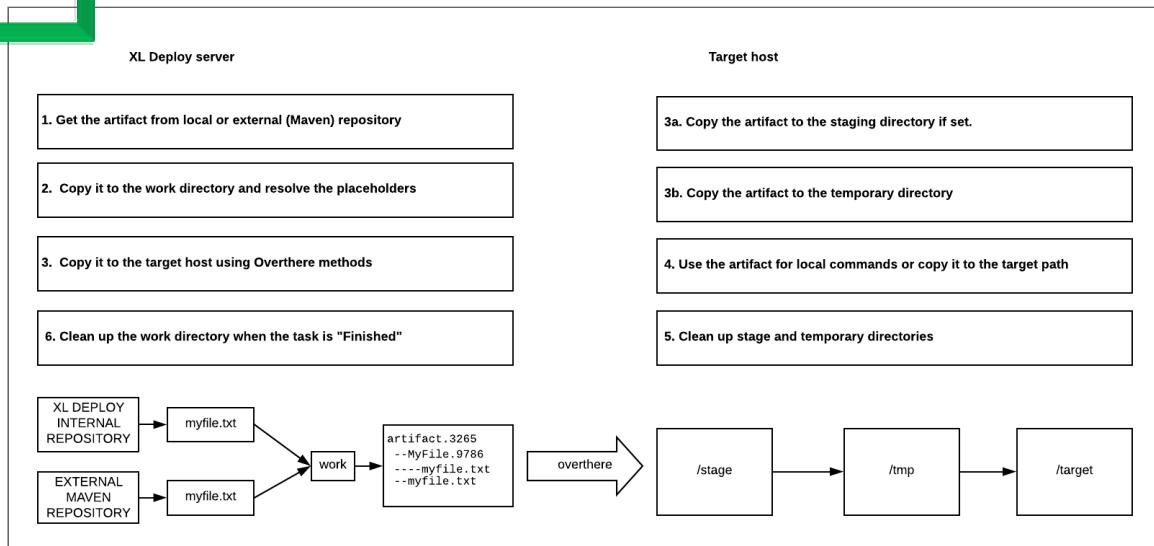


Staging Requirements

- The stagingDirectory property must be set on the host(s)
- The plugin you are using must support staging

Staging

Artifact lifecycle





Staging

Exercise: Use staging

On the **Library** section of the GUI

- Set the value of the **Staging Directory Path** (Advanced Tab) to **/tmp** on the following host:
 - **Infrastructure/was-vm**

Staging

Exercise: Use staging

On the **Library** section of the GUI under **Applications**

- Create a plan that deploys **PetClinic-ear/2.1** on the **Test** environment
- Preview to generate the default **deployeds**
- Expand the **Preview** deploy tasks to view the generated steps

Staging

Exercise: Use staging

PLAN ANALYZER

- ▼ Stage artifacts
 - ▶ Staging to was-vm
- ▼ Deploying PetClinic-ear 2.1 on environment Test
 - Stop managed server 'server3'
 - Stop managed server 'server2'
 - Deploy PetClinic on server3
 - Deploy PetClinic on server2
 - Synchronize vagrantNode02
 - Synchronize vagrantNode03
 - Start managed server 'server3'
 - Start managed server 'server2'
 - Start PetClinic on server3
 - Start PetClinic on server2
- ▼ Clean up staged artifacts
 - ▶ Clean up staged files on was-vm



Tags

<http://www.xebialabs.com>



Introduction

- Base mapping is done through a deployable-to-container relationship
- Tags make it easier to configure deployments by marking which deployables should be mapped to which containers
 - Tags are specified on deployables and containers
 - Tags can be specified either in the imported package or by using the Library browser
 - Next slide presents matching rules used between deployables and containers



Introduction

Deployables/Containers	No tags	Tag *	Tag +	T A
------------------------	---------	-------	-------	-----

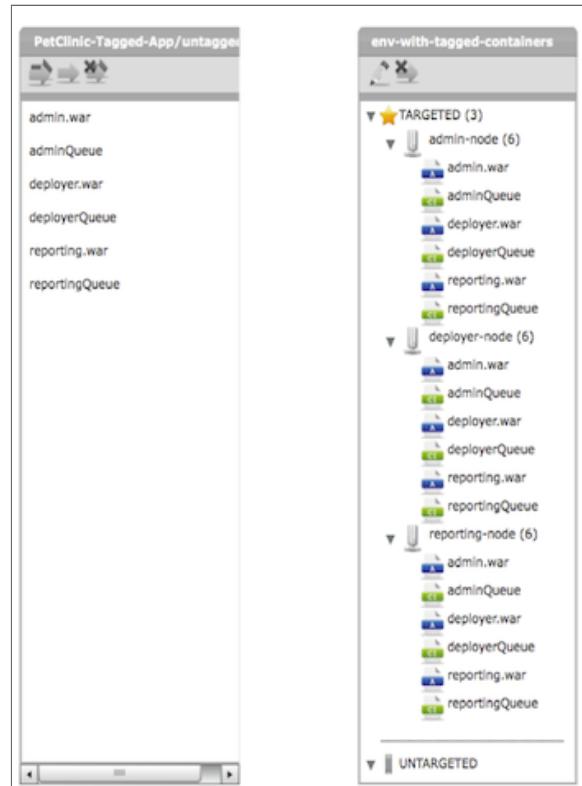
No tags	√	√	X	X
Tag *	√	√	√	√
Tag +	X	√	√	√
Tag A	X	√	√	√
Tag B	X	√	√	X
Tags A,B	X	√	√	√

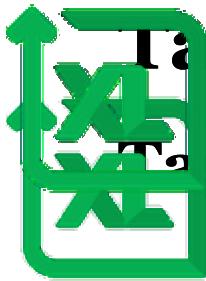
√ = mapped, X = blocked



Deployment without tags

- When preparing a deployment, XL Deploy will generate mappings for each combination of deployable and container
- This can easily become cumbersome to manage for big packages/large environments, especially when using CLI scripts





Tagging a deployable and container

PetClinic

ID: Applications/PetClinic-ear2.1/PetClinic

Name: PetClinic

Type: jee.Ear

File Uri: jcr:PetClinic-2.1.ear

Tags: (empty)

server3

WebSphere node: Infrastructure/was-vm/vagrantCell03/vagrantNode03

Server type: (empty)

Weight: 2

Module Destination: (empty)

Deploy To Running Container: Deploy To Running Container

Deployment Group number: 2



Deployment with tags

- Tags must be defined on deployable and container
 - The deployed is generated only when both the deployable and container have a matching tag
 - No tag defined on both deployable and container: default type-based deployed generation



Tags

Exercise: Deploy with tags

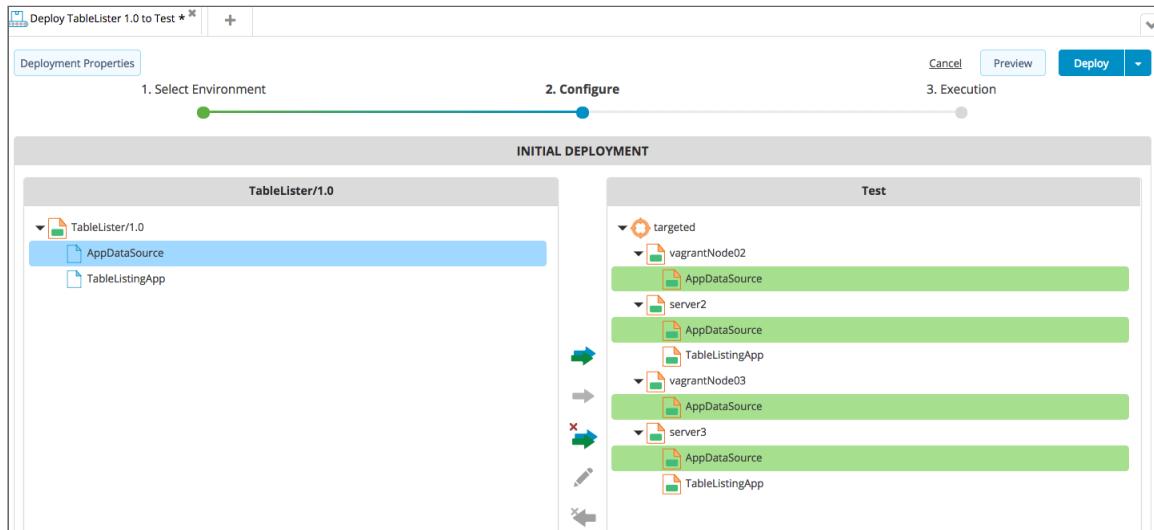


Exercise: Deploy with tags

- When deploying **TableLister/1.0** to **Test**
 - Generate the default deployeds
 - Use tags to achieve the following mapping automatically
 - TableListingApp
 - Server2
 - Server3
 - AppDataSource
 - vagrantNode02
 - vagrantNode03

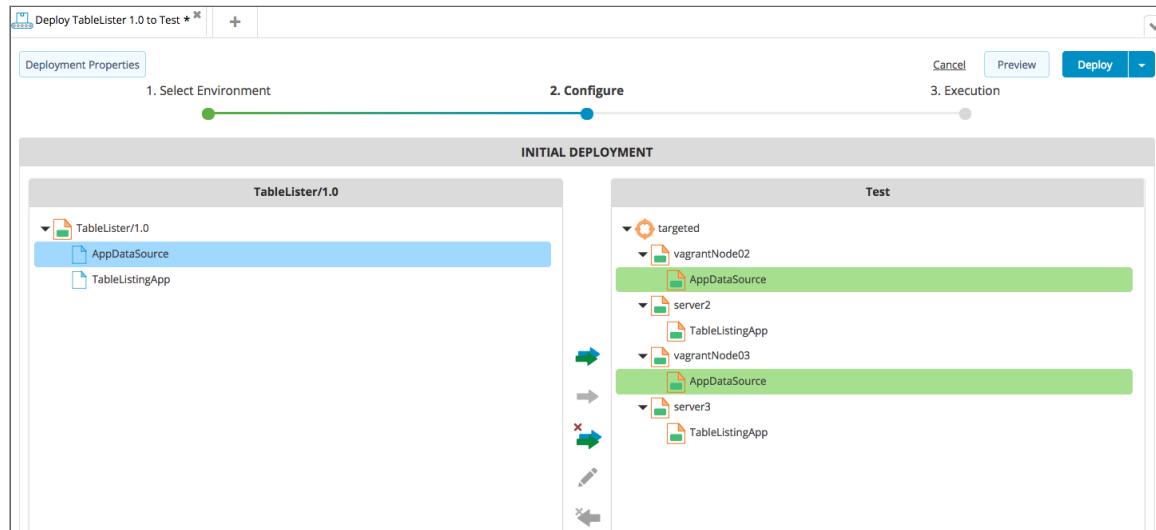


Exercise: Deploy with tags (mapping without tags)





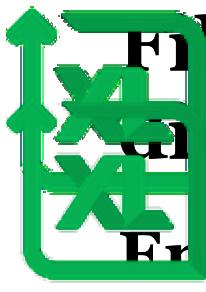
Exercise: Deploy with tags (mapping with tags)





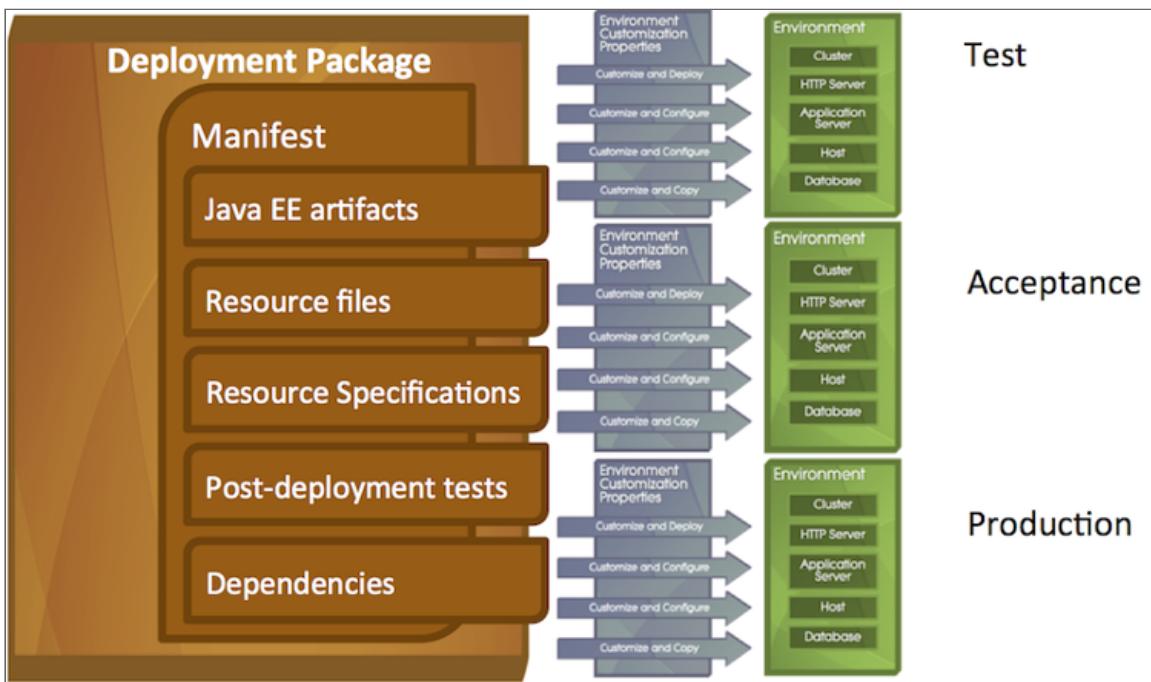
File placeholders

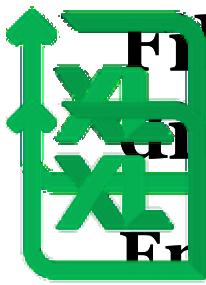
<http://www.xebialabs.com>



File placeholders and dictionaries

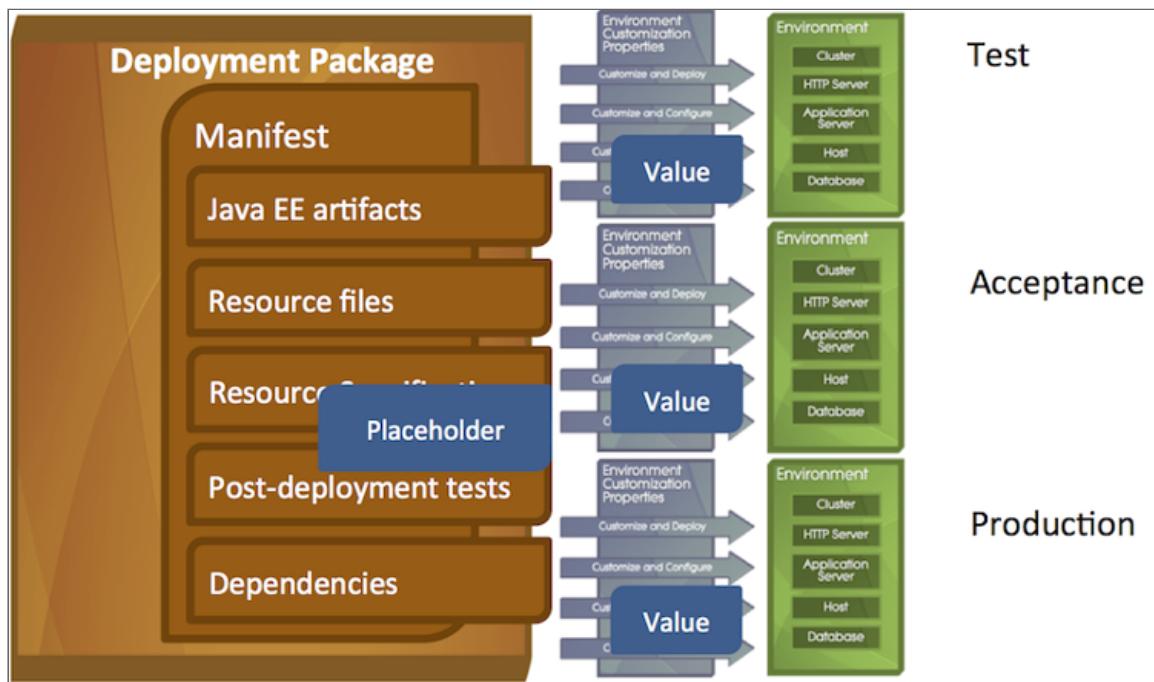
Environment independent



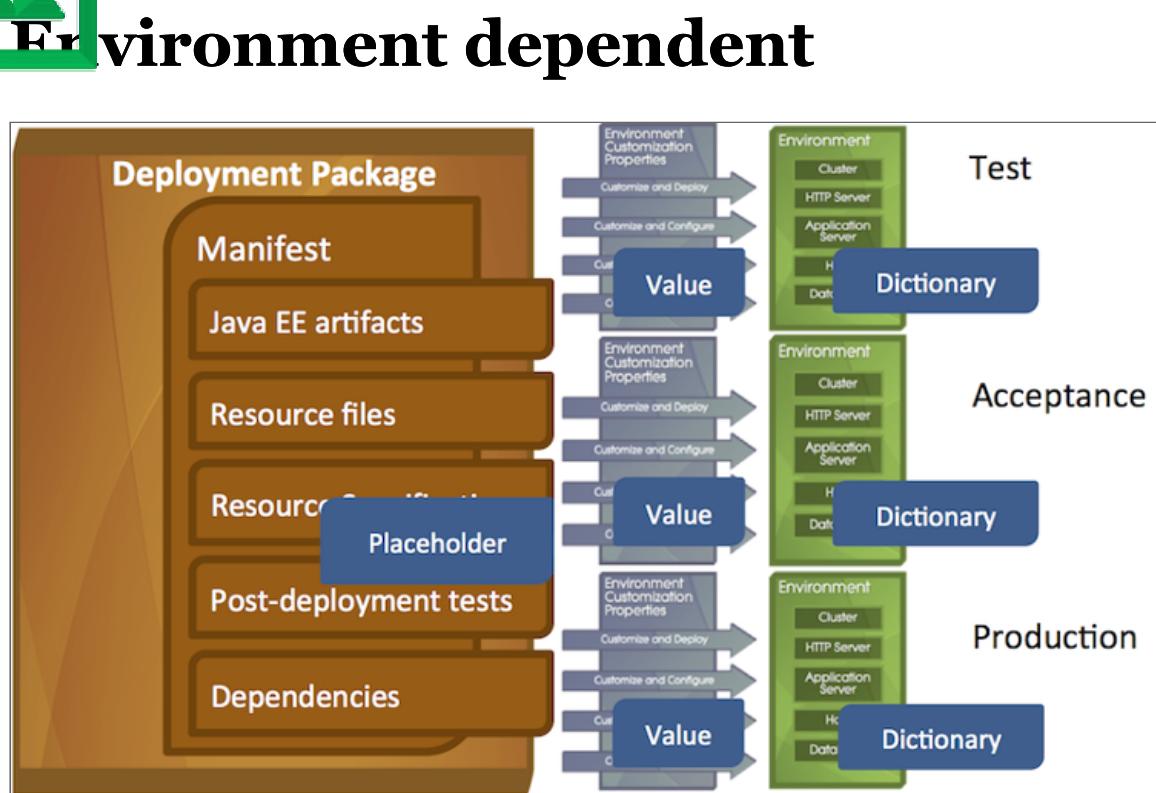


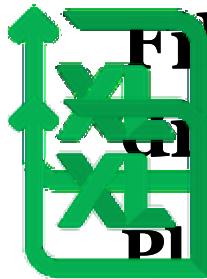
File placeholders and dictionaries

Environment dependent



File placeholders and dictionaries





File placeholders and dictionaries

Placeholder and dictionaries

- Placeholders are resolved by searching the target environment's dictionaries
- If no value is found, the placeholder is blank
- Resolved placeholders can be overridden



File placeholders

Introduction

- File placeholders:
 - Are specified inside files
 - Are scanned when the package is imported
 - Are replaced when a deployment is executed
 - **Must be specified before deployment can be executed**
- Use syntax: {{PLACEHOLDER}}

File placeholders

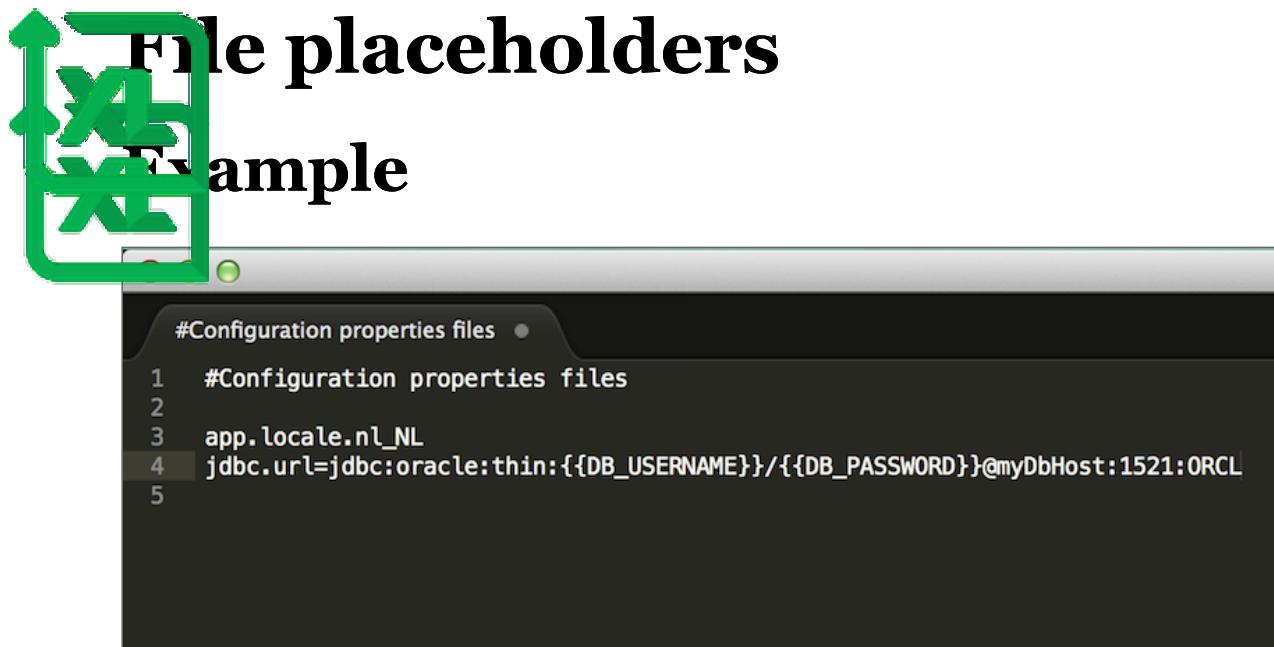
Example

The screenshot shows the 'ConfigurationDictionary' configuration item in the XL Deploy interface. The 'Entries' section contains two entries:

Key	Value
DB_USERNAME	scott
DB_PASSWORD

The 'Encrypted Entries' section also contains two entries:

Key	Value
DB_PASSWORD





File placeholders

Example

```
#Configuration properties files •  
1 #Configuration properties files  
2  
3 app.locale.nl_NL  
4 jdbc.url=jdbc:oracle:thin:scott/tiger@myDbHost:1521:ORCL  
5
```

File placeholders Example

The screenshot shows two windows from the XL Deploy Repository Workspace.

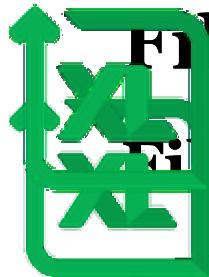
Left Window: configuration.properties

- Common Tab:**
 - Id:** Applications/TableLister/1.0/configuration.properties
 - Name:** configuration.properties
 - Type:** file.File
- Placeholders Tab:**
 - Scan Placeholders
 - Placeholders:
 - DB_USERNAME
 - DB_PASSWORD

Right Window: configuration.properties : was-vm

- Common Tab:**
 - Name:** configuration.properties
 - Type:** file.DeployedFile
- Placeholders Tab:**

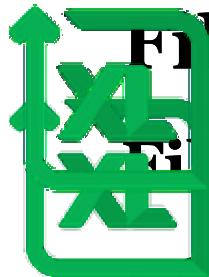
Key	Value
DB_PASSWORD	tiger
DB_USERNAME	scott



File placeholders

File plugin

- The File plugin contains logic for copying files and folders to a host
- Provides:
 - **file.File**
 - **file.Archive**
 - **file.Folder**
- Supports deployment to **overthere.Host** containers
- Included as standard plugin with XL Deploy

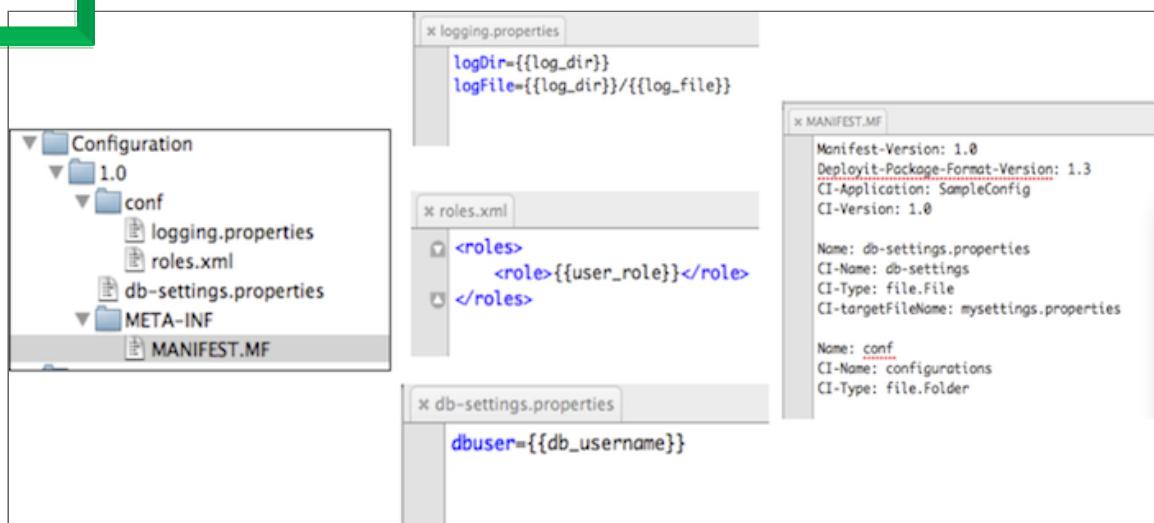


File placeholders

File plugin

- Copied files are placeholder-aware (scanning, replacing)
- Supports deep-scanning of archives
- Files and folders are copied to target container on deployment
- Files and folders are replaced on target container on upgrade

File placeholders File plugin example





File placeholders

File plugin example (file.Folder)

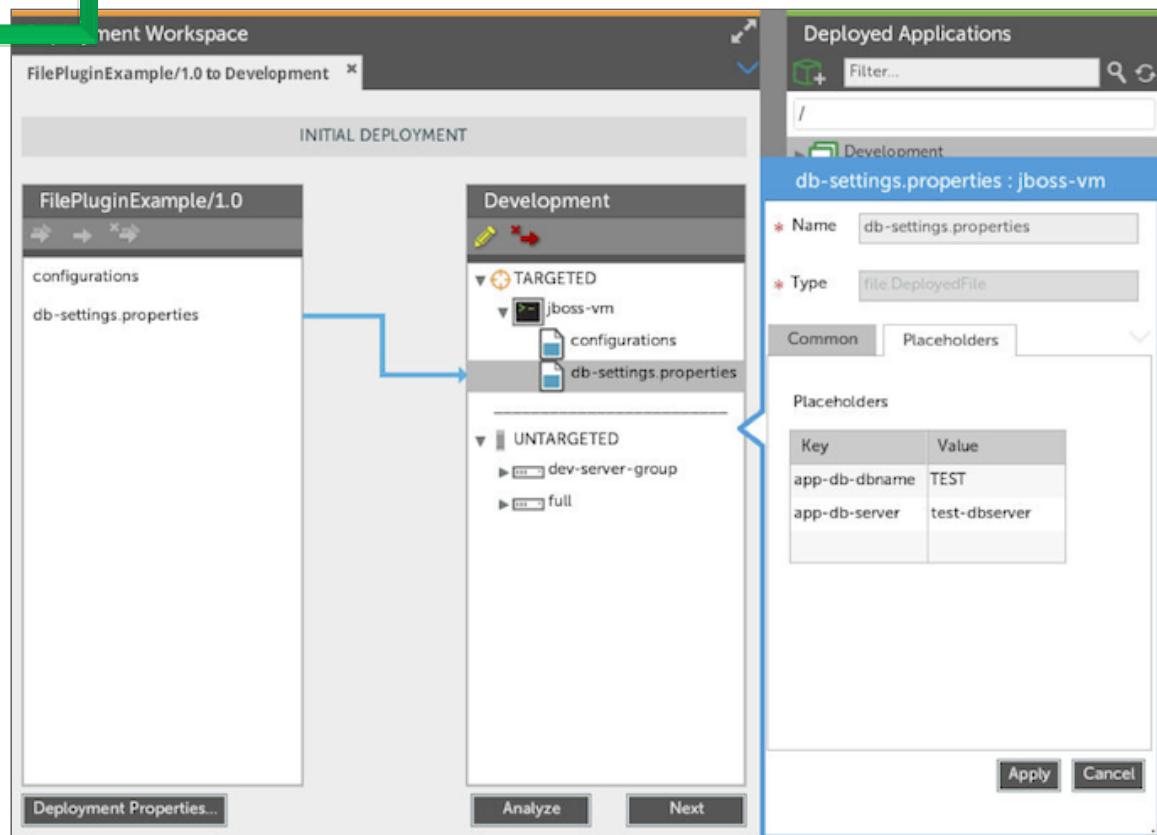
The screenshot shows the XL Deploy interface during an initial deployment. On the left, the 'INITIAL DEPLOYMENT' panel displays the application 'FilePluginExample/1.0'. It lists 'configurations' and 'db-settings.properties'. A blue arrow points from the 'configurations' item to the 'TARGETED' section of the central tree view. The 'TARGETED' section contains a 'jboss-vm' node, which also has a blue arrow pointing to its 'configurations' and 'db-settings.properties' children. On the right, a 'Deployed Applications' window is open, showing a configuration named 'jboss-vm'. The 'Placeholders' tab is selected, displaying the following table:

Key	Value
app-log-dir	/tmp
app-log-filename	example.log
app-log-level	DEBUG

At the bottom right of the placeholder dialog are 'Apply' and 'Cancel' buttons.

File placeholders

File plugin example (file.File)





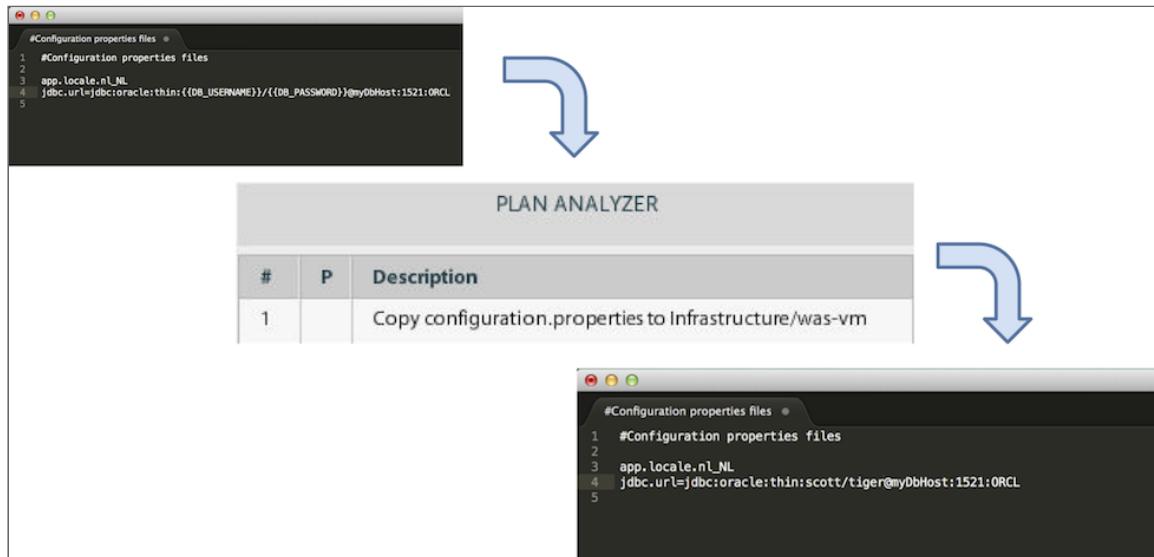
File placeholders

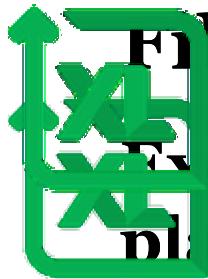
**Exercise: Deploy files with
placeholders**



File placeholders

Exercise: Deploy files with placeholders

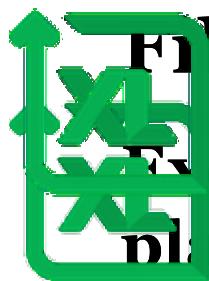




File placeholders

Exercise: Deploy files with placeholders

- Define a package containing the following CIs:
 - **file.File**
 - **file.Folder**
 - **file.Archive**
- Add some placeholders in your files, also to files in the ZIP archive
- Define an environment with an **overthere.Host** container



File placeholders

Exercise: Deploy files with placeholders

- Deploy the package
- Verify the generated steps
- Check that the files are copied to the target directory and placeholders are replaced



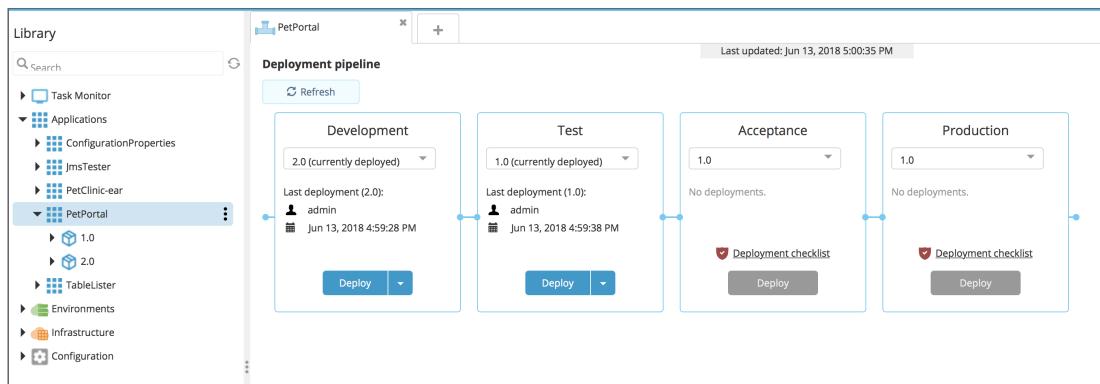
Release dashboard

<http://www.xebialabs.com>

Release dashboard

Introduction

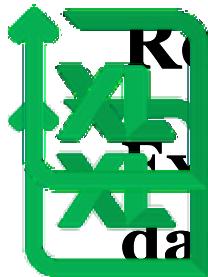
- Overview of your application in the Deployment Pipeline
- Promote applications through the pipeline
- Checks: only allow deployments when prerequisites are met





Release dashboard

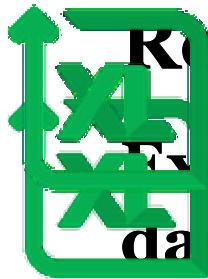
Exercise: configure release dashboard



Release dashboard

Exercise: configure release dashboard

- In the **Library** section of the GUI:
 - Click on the three dots to the right of **Configuration** and select:
 - **New ▶ release ▶ DeploymentPipeline**
- Name the deployment pipeline: DTAP
- Add the following members to the pipeline:
 - Development
 - Test

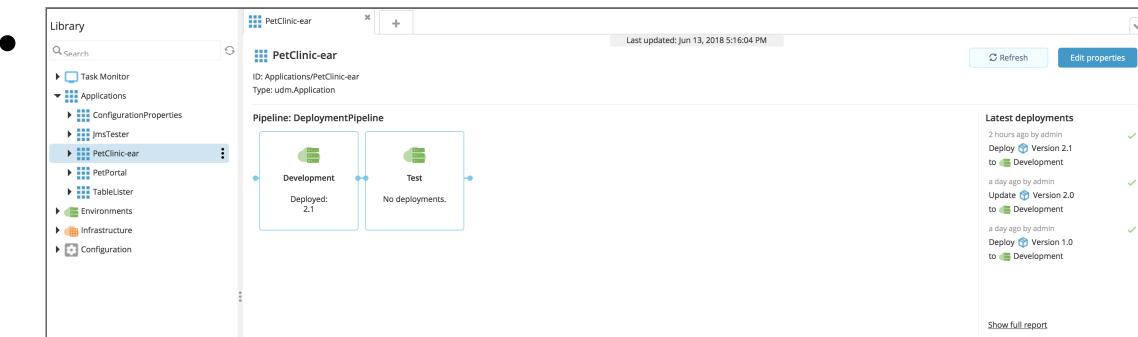


Release dashboard

Exercise: Configure release dashboard

Associate the **DTAP** pipeline to the **PetClinic-ear** application

- In the **Library** section of the GUI under **Applications** double-click the **PetClinic-ear** application, select the **DTAP** pipeline, save.
- Click the three-dots and select Deployment Pipeline.





Release dashboard

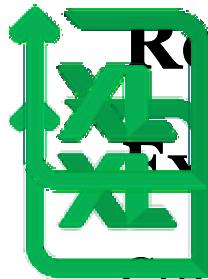
Release dashboard checks

- All available checks are defined globally defined (in **synthetic.xml**, more on that later)
- For each **Environment**, you can define which checks it requires from a **Deployment Package**
- Each **Deployment Package** maintains a list of checks it satisfies, i.e are ticked off.
- Deployments are only allowed if the **Deployment Package** satisfies all the requirements configured on the **Environment**



Release dashboard

Exercise: Complete the DTAP



Release dashboard

Exercise: Complete the DTAP

Create an empty **Acceptance** and **Production** environments:

- On the **Deployment** tab, click the icon under **Environments**
- Click **Create environment**
- Set the name to Acceptance and click through the screens in the wizard to save the environment without adding any containers or dictionaries
- Repeat for the Production environment



Release dashboard

Exercise: Complete the DTAP

For the **Acceptance** and **Production** environments, enable the following checks in the Release Checks tab:

Property	Acceptance	Production
Requires Release Notes	√	√
Requires Ticket Number	√	√
Requires Passed Functional Testing	√	√
Requires Passed Acceptance Testing		√
Requires Signed Off By Release Manager		√



Release dashboard

Exercise: Complete the DTAP

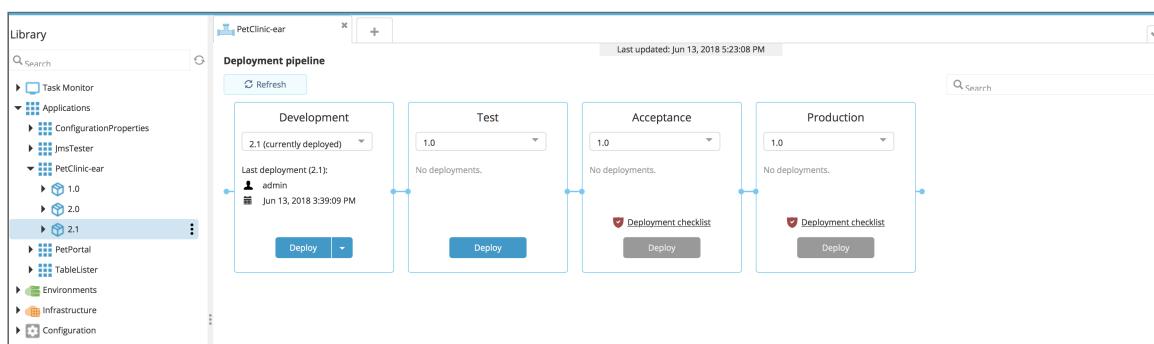
- Add the newly created **Acceptance** and **Production** environments to the **DTAP** pipeline:
- In the **Library** section of the GUI:
 - Go to **Configuration** ▶ **DTAP** (double-click)
- Add the following members to the pipeline:
 - **Acceptance**
 - **Production**

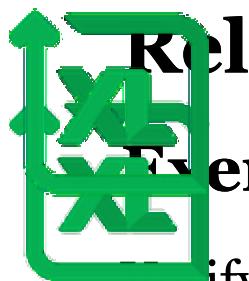


Release dashboard

Exercise: Complete the DTAP

- On the **Release Dashboard** tab, investigate the newly created pipeline for the application





Release dashboard

Exercise: Complete the DTAP

Verify the configured **release checks** by clicking on the **Acceptance** or **Production** environment in the **Deployment Pipeline**:

Deployment checklist for migrating 2.1 to Production

⚠ Has release notes

Ticket number of the release
⚠

⚠ Passed functional testing

⚠ Passed acceptance testing

Signed off by Release Manager
⚠



Release dashboard

Exercise: Configure checks

- Checks are configured in ext/synthetic.xml
- Restart the XL Deploy server after editing synthetic.xml



Release dashboard

Exercise: Checks on environments

Guidelines:

- Check property names start with "requires"
- Property **kind** is always "Boolean"
- Property **required** is always "false"
- Property **category** defines the UI tab ("Release Checks")



Release dashboard

Exercise: Checks on environments

This is the list of all possible checks on all environments

```
<!-- Release Dashboard checks for all environments ('requires') -->
<type-modification type="udm.Environment">
    <property name="requiresReleaseNotes" kind="bool"/>
    <property name="requiresTicketNumber" kind="bool"/>
    <property name="requiresPassedFunctionalTesting" kind="bool"/>
    <property name="requiresPassedAcceptanceTesting" kind="bool"/>
    <property name="requiresSignedOffByReleaseManager" kind="bool"/>
</type-modification>
```



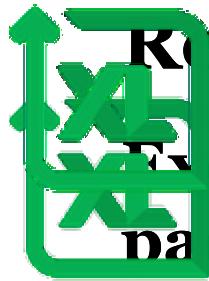


Release dashboard

Exercise: Checks on environments

Guidelines:

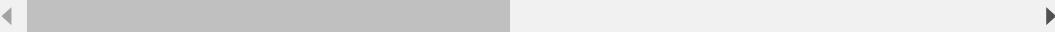
- Name starts with "satisfies" and match "requires"
- Property kind can be int, Boolean, or string
- Required must be false
- Label defines a readable label for the GUI
- Can define validation rules



Release dashboard

Exercise: Checks on deployment packages

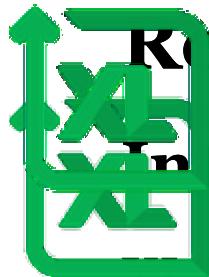
```
<!-- Release Dashboard checks for individual application versions
<type-modification type="udm.Version">
    <property name="satisfiesReleaseNotes" kind="boo
    <property name="satisfiesTicketNumber" kind="str
        <rule type="regex" pattern="^PETCLINIC-[0-9]+$" message="
    </property>
    <property name="satisfiesPassedFunctionalTesting" kind="boo
    <property name="satisfiesPassedAcceptanceTesting" kind="boo
    <property name="satisfiesSignedOffByReleaseManager" kind="str
</type-modification>
```





Reports

<http://www.xebialabs.com>



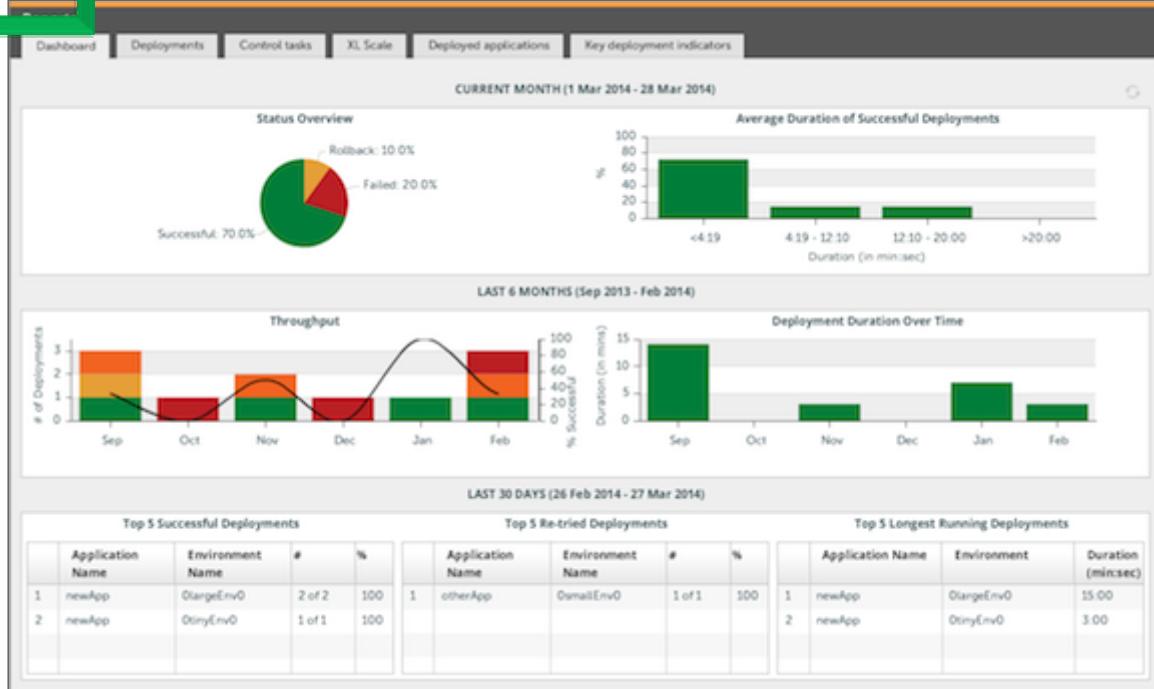
Reports

Introduction

XL Deploy provides a dashboard and configurable reports out-of-the-box including:

- Dashboard
- Deployed applications per environment
- Deployments done in a certain data range
- Key deployment indicators in date range

XL Deploy Reports Dashboard



The image shows a screenshot of the XL Deploy Deployments report interface. At the top, there is a green header bar with the text "Reports" and "Deployments report". Below the header is a navigation bar with tabs: Dashboard, Deployments, Control tasks, XL Scale, Deployed applications, and Key deployment indicators. The main area is titled "Report Parameters" and contains a sub-instruction "Double-click on a deployment to see its details". A table displays deployment information:

Package	Environment	D...	U...	S...	Start Date	1▼	Completion ...
parallelApp/0.0	0parallelEnv0	Initial	admin	DONE	Mar 28, 2014 at 11:34 AM	Mar 28, 2014 at 11:34 AM	Mar 28, 2014 at 11:34 AM
parallelFailingApp/0.0	0parallelEnv0	Initial	admin	CANC	Mar 28, 2014 at 11:31 AM	Mar 28, 2014 at 11:31 AM	Mar 28, 2014 at 11:31 AM
smallApp/1.0	1newEnv1	Initial	admin	DONE	Mar 28, 2014 at 10:01 AM	Mar 28, 2014 at 10:01 AM	Mar 28, 2014 at 10:01 AM
smallApp/1.0	0tinyEnv0	Initial	admin	DONE	Mar 28, 2014 at 10:01 AM	Mar 28, 2014 at 10:01 AM	Mar 28, 2014 at 10:01 AM
tinyApp/1.0	1newEnv1	Initial	admin	DONE	Mar 28, 2014 at 10:01 AM	Mar 28, 2014 at 10:01 AM	Mar 28, 2014 at 10:01 AM
tinyApp/1.0	0tinyEnv0	Initial	admin	DONE	Mar 28, 2014 at 10:01 AM	Mar 28, 2014 at 10:01 AM	Mar 28, 2014 at 10:01 AM

Reports

Deployed applications report

The screenshot shows the XL Deploy interface with a green header bar containing the word 'Reports'. Below it, a large title 'Deployed applications report' is displayed. The main content area is a table titled 'Deployed applications in environment' with a dropdown filter set to '0tinyEnv0' and a date filter set to 'up to 03/28/2014'. A note below the table says 'Double-click on a deployment to see its details'. The table has columns for Application, Version, User, and Date of deployment. The data is as follows:

Application	Version	User	Date of deployment
otherApp	1.0	admin	Dec 28, 2013 at 10:12 AM
oldApp	1.0	admin	Jan 28, 2014 at 10:08 AM
newApp	1.0	admin	Feb 26, 2014 at 10:04 AM
dictionaryApp	1.0	admin	Feb 26, 2014 at 10:12 AM
tinyApp	1.0	admin	Mar 28, 2014 at 10:01 AM

Reports

Key deployment indicators report

The screenshot shows the XL Deploy interface with a green header bar containing the word 'Reports'. Below it is a sub-header 'Key deployment indicators report'. A navigation bar at the top includes tabs for 'Dashboard', 'Deployments', 'Control tasks', 'XL Scale', 'Deployed applications', and 'Key deployment indicators' (which is currently selected). A sub-menu titled 'Report Parameters' is open, showing a table with deployment statistics. The table has columns for Application, Environment, Success, Retried, Aborted, Rollbacks, and Average Duration. The data is as follows:

Application	Environment	Success	Retried	Aborted	Rollbacks	Average Duration
parallelApp	OparallelEnv0	1	0	0	0	00:00:12
oldApp	OtinyEnv0	1	0	0	0	00:07:00
tinyApp	OtinyEnv0	1	0	0	0	00:00:01
dictionaryApp	OsmallEnv0	2	0	0	1	00:16:30
parallelFailingApp	OparallelEnv0	0	0	1	0	00:00:00



Reports

Exercise: Use reports



Exercise: Use reports

On the Reports tab:

- Choose the Deployments tab
- Get a report of the deployments you did yesterday

XL Deploy Reports

Exercise: Use reports

The screenshot shows the 'Report Parameters' section with date filters set from 05/01/2013 to 05/03/2013. A table below lists five deployments for the package 'PetClinic-ear/2.1' in the 'Development' environment. All deployments were performed by user 'admin' and are marked as 'DONE'. The table includes columns for Package, Environment, Deployment Type, User, Status, Start Date, and Completion Date.

Package	Environment	Deployment Type	User	Status	Start Date	Completion Date
PetClinic-ear/2.1	Development	Initial	admin	DONE	May 03, 2013 at 12:42 AM	May 03, 2013 at 12:43 AM
PetClinic-ear/2.1	Development	Undeployment	admin	DONE	May 03, 2013 at 12:42 AM	May 03, 2013 at 12:42 AM
PetClinic-ear/2.1	Development	Initial	admin	DONE	May 03, 2013 at 12:36 AM	May 03, 2013 at 12:37 AM
PetClinic-ear/2.1	Development	Undeployment	admin	DONE	May 03, 2013 at 12:35 AM	May 03, 2013 at 12:36 AM
PetClinic-ear/2.1	Development	Initial	admin	DONE	May 03, 2013 at 12:35 AM	May 03, 2013 at 12:35 AM

Reports

Exercise: Use reports

Double-click a line in the report to get the deployment log

Initial deployment of PetClinic-ear/2.1 on Development

DEPLOYMENT TASKS		
60a0b29c-0e8c-4308-97ba-967041e14101		
#	Description	State
1	Deploy PetClinic on existing-cluster	DONE
2	Synchronize vagrantNode01	DONE
3	Update global Web Server plugin configuration for 'vagrantCell01'	DONE
4	Start PetClinic on existing-cluster	DONE

Synchronizing vagrantNode01
Completed synchronization of vagrantNode01

Close



Security

<http://www.xebialabs.com>



Security

Introduction

- Security is based on **Roles and Permissions**
 - **Roles** are associated with **Principals** (usually users or groups from LDAP)
 - Permissions are predefined at Global or CI-level

XL Deploy Security Introduction

- UI-level permissions can only be set on Directories





Security Inheritance

- Children inherit parents permissions if they do not have permissions of their own.
- If any permission is set on a directory, ALL permissions of the parent are overridden.
- If you give permission to an environment, you have to give read permission to all of the referenced infrastructure members.



Security

Global permissions

- **admin:** Grants all rights within XL Deploy.
- **discovery:** Perform discovery of middleware.
- **login:** Allows user to Log into the XL Deploy application.
 - This permission does not automatically allow the user access to nodes in the repository.
- **security edit:** Allows administration of security permissions.
- **task assign:** Grants ability to reassign any task to someone else.
- **task takeover:** Allows users to assign any task to yourself.
- **task preview step:** Allows inspection of scripts that will be executed within the deployment plan.
- **report view:** View all the reports. When granted, the UI will show the Reports tab.
 - To view the full details of an archived task, users need read permissions on both the environment & application.



Local permissions 1/3

- **control task execute:** Allows execution of control tasks on configuration items.
- **deploy initial:** Grants permission to perform an initial deployment of a package to an environment.
 - Applies only for the Environment CIs within the containing directory.
- **deploy undeploy:** Allows the undeploy of an application.
 - Applies only for the Environment CIs within the containing directory.
- **deploy upgrade:** Grants permission to perform an upgrade deployment on an environment.
 - Does not allow deploying items from the package to new targets. Applies only to Environment CIs within the containing directory.



Local permissions 2/3

- **import initial:** Allows initial import of a package for a new application.
- **import remove:** Allows removal of an application or package.
 - Applies only for Application and DeploymentPackage CIs within the containing directory.
- **import upgrade:** Allows import of a package to upgrade an existing application.
 - Applies only for Application CIs within the containing directory.
- **read:** View CIs in the repository.
- **repo edit:** Allows editing (create and modify) CIs in the library.
 - The user must also have read access to CIs to be able to edit them.
 - Applies only for the CIs within the containing directory.



Local permissions 3/3

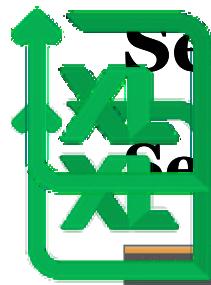
- **task assign:** Allows the transfer a task to another user.
- **task move step:** Allows moving of steps in the generated steplist before starting a deployment.
 - Applies only for deployments executed on Environment CIs in the containing directory.
- **task skip step:** Allows skipping steps in the generated steplist before starting a deployment.
 - Applies only for deployments executed on Environment CIs in the containing directory.



Security

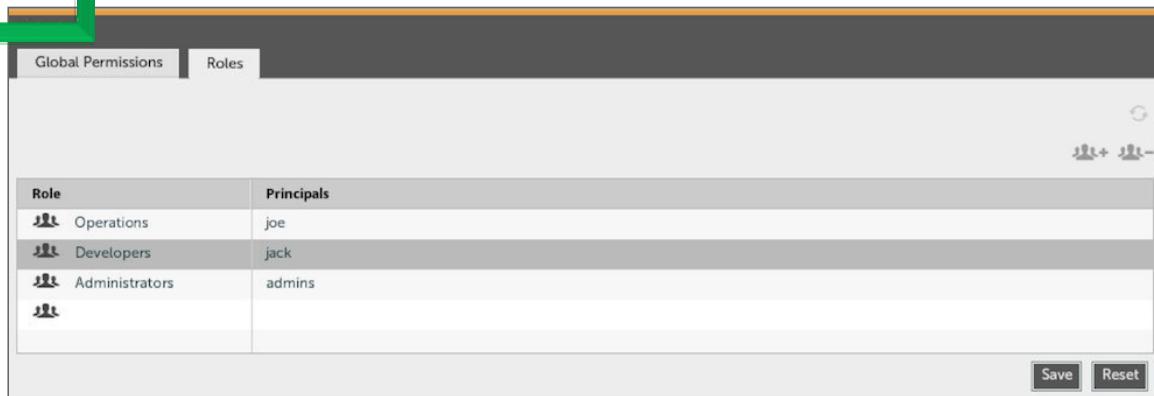
New remarks on permissions...

- **Import upgrade** is only applicable to an Applications
- **deploy** is only applicable to an Environment
- **import initial** allows user to create new udm.Application
- **import upgrade** allows users to create new udm.DeploymentPackages



Security

Security roles in the GUI



The screenshot shows a user interface for managing security roles. At the top, there are tabs for "Global Permissions" and "Roles". The "Roles" tab is selected. Below the tabs is a search bar with a magnifying glass icon and two buttons for adding (+) or removing (-) items. The main area is a table with two columns: "Role" and "Principals". The table contains three rows:

Role	Principals
Operations	joe
Developers	jack
Administrators	admins

At the bottom right of the table are "Save" and "Reset" buttons.

Security level permissions in the GUI

The screenshot shows the 'Repository Workspace' interface for managing application permissions. On the left, a tree view shows 'Applications' containing 'JmsTester', 'MyDivision' (selected), 'PetClinic-ear', 'TableLister', 'Environments', 'Infrastructure', and 'Configuration'. A context menu for 'MyDivision' is open, with 'Permissions' highlighted. The main window title is 'Applications/MyDivision Permission'. It displays a grid of roles and their permissions:

Role	controltask	import initial	import remove	import upgrade	read	repo edit
Administrators	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Operations	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Developers	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Buttons at the bottom right include 'Save' and 'Reset'.



Security

Exercise: security



Exercise: security

- Create a **developer** role (in GUI in the **Admin** tab)
- Create a user, use any username you want, using the `security.createUser("myUsername", "myPassword")` command
- Assign the **developer** role to the new user
- Can you login with the newly created user?
 - Use the gear icon | Logout to logout



Exercise: security

- On the XL Deploy GUI open the **Admin** tab
 - On the **Admin** tab, open the **Roles** tab
 - Add a new role named **cli-users**
 - Specify the username you created through the CLI as a **principal** for the **cli-users** role

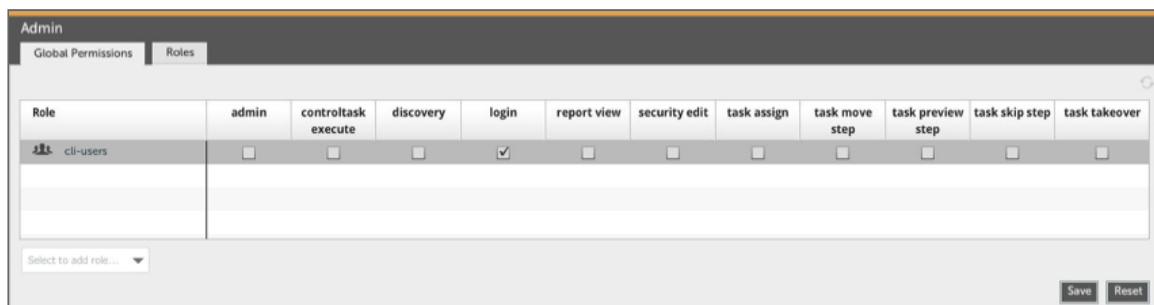
A screenshot of the XL Deploy Admin interface. The top navigation bar has tabs for 'Global Permissions' and 'Roles', with 'Roles' being the active tab. Below the tabs is a toolbar with icons for search, refresh, and user management. The main content area shows a table with two columns: 'Role' and 'Principals'. A single row is present, showing 'cli-users' in the Role column and 'myUsername' in the Principals column. At the bottom right of the table are 'Save' and 'Reset' buttons.

Role	Principals
cli-users	myUsername

Save Reset

XL Deploy Security Exercise: security

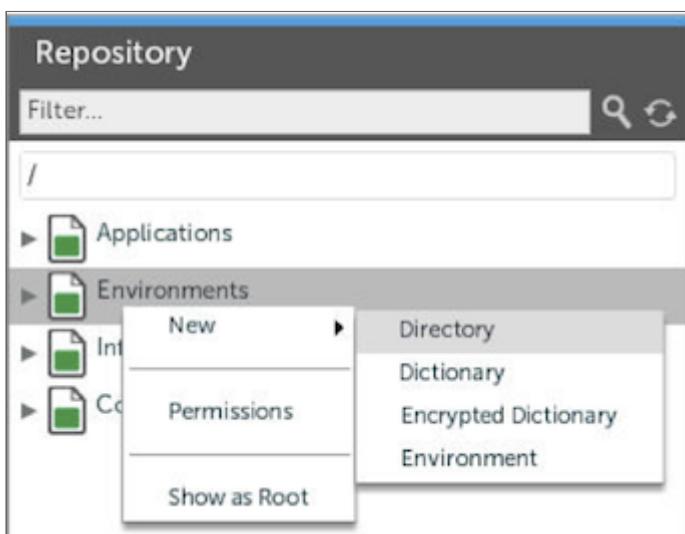
- On the XL Deploy GUI open the **Admin** tab
 - Open the **Global Permissions** tab
 - Reload the permissions
 - Select the **cli-users** role from the drop-down box
 - Grant the **cli-users** role the **login** permission
 - Try to login with the user created through the CLI





Exercise: security

- Use **Permissions** and **Directories** to reorganize the Library, so that cli-users can only see the PetClinic-ear application and the Development environment





Exercise: LDAP Integration

- Integrates with
 - Active directory
 - LDAP
- Configured in /conf/deployit-security.xml

```
<bean id="ldapServer" class="..."><constructor-arg value="ldap://192.168.0.10:389/" />
<property name="userDn" value="cn=admin,dc=nodomain" />
<property name="password" value="secret" />
<property name="baseEnvironmentProperties">
...
...
```



Security

Exercise: LDAP



Security

Exercise: LDAP

- LDAP structure
 - Group: devgroup
 - Members: james
 - Group: opsgroup
 - Members: john and brian



Security

Exercise: LDAP

- Integrate XL Deploy with the OpenLDAP server running on your training VM
- Backup your XL Deploy **conf** directory
- Copy the **deployit-security.xml** from the ldap folder of your training material to the **conf** directory of your XL Deploy installation
- Restart XL Deploy

XL Deploy Security Exercise: LDAP

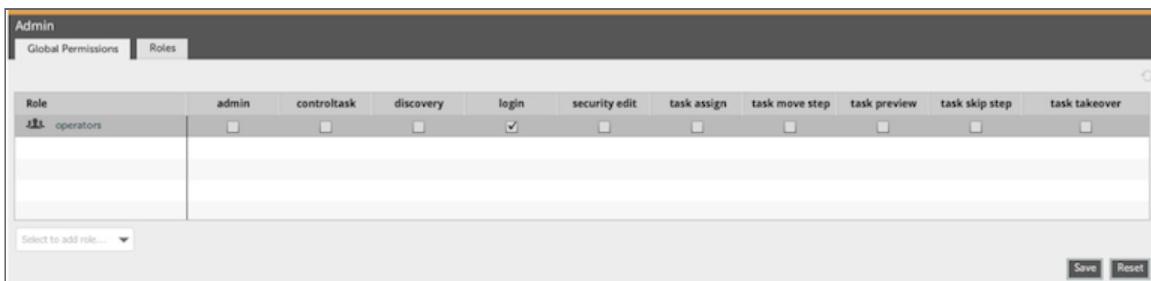
- On the XL Deploy GUI open the Admin tab
 - On the Admin tab, open the Roles tab
 - Add a new role named operators
 - Specify john as a principal for the operators role

Role	Principals
operators	john

Save Reset

XL Deploy Security Exercise: LDAP

- On the XL Deploy GUI open the Admin tab
 - Open the Global Permissions tab
 - Reload the permissions
 - Select the operators role from the drop-down box
 - Grant the operators role the login permission





Security

Exercise: LDAP

- Logout from the XL Deploy GUI and try to logon as user john
 - Use Help | Logout
 - password: john01



- Change the security configuration to support LDAP groups
- Open the file deployit-security.xml
- Locate the commented out constructor-arg section and uncomment it
- Restart XL Deploy

XL Deploy Security Exercise: LDAP

- On the XL Deploy GUI open the Admin tab
 - On the Admin tab, open the Roles tab
 - Update the principal(s) of the operators role to match the groupname of users john and brian (opsgroup)

The screenshot shows the 'Admin' interface with the 'Roles' tab selected. A table displays the 'operators' role with its principal set to 'opsgroup'. The 'Save' and 'Reset' buttons are visible at the bottom right.

Role	Principals
operators	opsgroup

Save Reset



Security

Exercise: LDAP

- Logout from the GUI and try to logon as a member of the opsgroup (john or brian)



Exercise: LDAP

- On the XL Deploy GUI open the Admin tab
 - On the Admin tab, open the Roles tab
 - Update the principal(s) of the operators role to match the groupname of users john and brian (opsgroup) and (individual) user james

A screenshot of the XL Deploy Admin interface. The top navigation bar has tabs for 'Admin', 'Global Permissions', and 'Roles'. The 'Roles' tab is selected. Below the tabs is a toolbar with icons for search, add, and delete. The main content area is a table titled 'Role' with one row. The row contains a column labeled 'Role' with the value 'operators' and a column labeled 'Principals' with the value 'james, opsgroup'. At the bottom right of the table are 'Save' and 'Reset' buttons.

Role	Principals
operators	james, opsgroup

Save Reset



Security Exercise: LDAP

- Logout from the XL Deploy GUI and try to logon as user james



CLI

<http://www.xebialabs.com>



Using the CLI

Introduction

- Remote client, uses http/https
 - Python-based (Jython)
 - Runs scripts or interactive command-line
 - starts with /bin/cli.sh or cli.cmd
 - Jython API
- <https://docs.xebialabs.com/jython-docs/#!/xl-deploy/5.1.x/>**
- CLI Manual **<https://docs.xebialabs.com/xl-deploy/how-to/using-the-xl-deploy-cli.html>**



Using the CLI

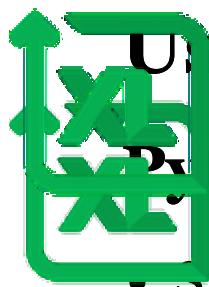
CLI shell options

```
Usage: ./cli.sh [options] [[--] arguments]
```

Options:

- configuration VAL : Specify the location of the configuration
- context VAL : Context url XL Deploy is running at, defau
- f (-source) VAL : Execute a specified python source file
- help : Prints this usage message
- host VAL : Connect to a specified host, defaults to 1
- password VAL : Connect with the specified password
- port N : Connect to a specified port, defaults to 4
- proxyHost VAL : Proxy host
- proxyPort N : Proxy port
- q (-quiet) : Suppresses the welcome banner
- secure : Use https to connect to the XL Deploy serv
- username VAL : Connect as the specified user

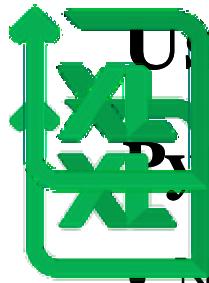




Using the CLI

Python basics

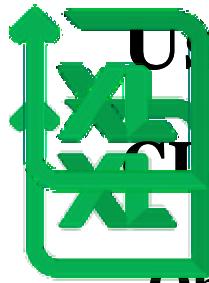
- String
 - 'single quoted' or "double quoted"
- List
 - [1, 2, 3, 4]
- Map
 - {'key1': 'value1', 'key2': 'value2'}



Using the CLI

Python basics

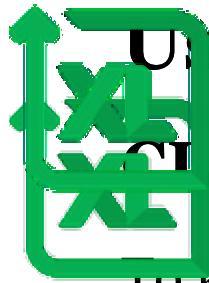
- Nested / mixed stuff
 - `a = {'list':[1, 2, 'a']}`
- Variable assignment
 - `b = [1, {'a':'b'}]`
- Lookup
 - `print a['list']; print b[0]`



Using the CLI

CLI Objects

Object	Purpose
deployit	The main gateway to interfacing with XL Deploy.
deployment	Perform tasks related to setting up deployments
factory	Helper that can construct Configuration Items (CI) and Artifacts
repository	Gateway to doing CRUD operations on all types of CIs
security	Access to the security settings of XL Deploy.
task2	Access to the task block engine of XL Deploy.
tasks	Access to the task engine of XL Deploy.



Using the CLI

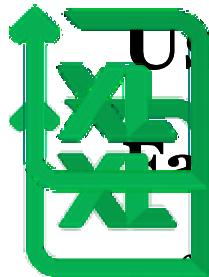
CLI Help

To know more about a specific object, type :

```
<objectname>.help()
```

To get to know more about a specific method of an object, type :

```
<objectname>.help("<methodname>")
```



Using the CLI

Factory object

- factory.help()

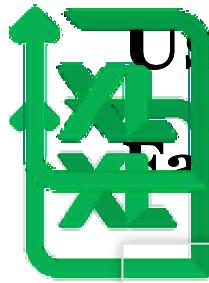
- factory: Helper that can construct Configuration Items (CI) and Artifacts
- The methods available are:
 - factory.artifact(String id, String ciType, Map values, byte[] data) : ArtifactAndData
 - factory.configurationItem(String id, String ciType, Map values) : ConfigurationItem
 - factory.configurationItem(String id, String ciType) : ConfigurationItem
 - factory.types() : void



Using the CLI

Factory object example 1

```
# Create a dictionary
dict = factory.configurationItem(
  'Environments/sampleDict',
  'udm.Dictionary',
  {'entries': {'key1':'value1','key2':'value2'}})
```



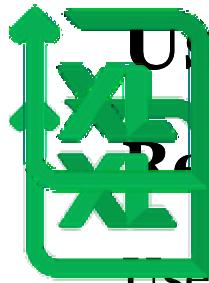
Using the CLI

Factory object example 2

```
from java.io import File
from com.google.common.io import Files

data = Files.toByteArray(File('/path/to/sql.zip'))
sqlscripts = factory.artifact('Applications/sqlApp/1.0/sql', 'sql'
sqlscripts.filename = 'sql.zip'
repository.create(sqlscripts)
```





Using the CLI

Repository object

Use the repository object for CRUD operations on the XL Deploy repository

- `repository.copy(String id, String newId) : ConfigurationItem`
- `repository.create(ConfigurationItem ci) : ConfigurationItem`
- `repository.create(List cis) : List`
- `repository.create(ArtifactAndData artifact) : ConfigurationItem`
- `repository.delete(String id) : void`
- `repository.exists(String id) : boolean`
- `repository.exportArchivedTasks(String filePath) : void`
- `repository.exportArchivedTasks(String filePath, String beginDate, String endDate) : void`
- `repository.exportDar(String directoryPath, String versionId) : void`



Using the CLI

Repository object

- `repository.getArchivedTasks(String beginDate, String endDate) : FullTaskInfos`
- `repository.getArchivedTasks() : FullTaskInfos`
- `repository.move(ConfigurationItem ci, String newId) : ConfigurationItem`
- `repository.move(String id, String newId) : ConfigurationItem`
- `repository.read(String id) : ConfigurationItem`
- `repository.read(List ids) : List`
- `repository.rename(String id, String newName) : ConfigurationItem`
- `repository.rename(ConfigurationItem ci, String newName) : ConfigurationItem`
- `repository.search(String ciType, String parent) : List`
- `repository.search(String ciType) : List`
- `repository.search(String ciType, Calendar before) : List`
- `repository.searchByName(String name) : List`



Using the CLI

Repository object

- `repository.update(List cis) : List`
- `repository.update(ArtifactAndData artifact) : ConfigurationItem`
- `repository.update(ConfigurationItem ci) : ConfigurationItem`



Using the CLI

Repository examples 1/2

Example 1:

```
# Create a dictionary and save it in XL Deploy
dict = factory.configurationItem(...)
repository.create(dict)

# Create several CIs at once
repository.create([ci1, ci2])
```

Example 2:

```
# Read a CI from the repository and update it
petclinic = repository.read("Applications/PetClinic-ear/1.0")
deployit.print(petclinik)
petclinic.satisfiesReleaseNotes = True
repository.update(petclinik)
```



Using the CLI

Repository examples 2/2

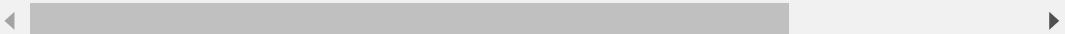
```
# Deletes a CI and all its children
if repository.exists("Applications/PetClinic-ear"):
    repository.delete("Applications/PetClinic-ear")
```



Using the CLI

Repository object: move/rename

```
repository.rename("Infrastructure/ApacheHost", "WebserverHost")  
repository.move("Infrastructure/WebserverHost", "Infrastructure/Ope
```





Using the CLI

Repository object: search

```
# Search by type
allEnvironments = repository.search("udm.Environment")
for id in allEnvironments:
    print id

# Search for children
children = repository.search(None,"Applications/PetClinic-ear")
for ci in children:
    print("Found CI with ID %s" % (ci))
```



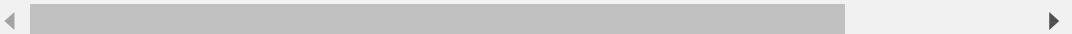


Using the CLI

Repository object: Tasks and export

```
# Inspect archived tasks
repository.exportArchivedTasks("exportedTasks.xml", "10/26/2012",
taskInfoContainer = repository.getArchivedTasks()
for t in tasksInfoContainer.tasks:
    print t.id
    for step in t.steps:
        print step.description

# Export as DAR
repository.exportDar("/tmp", "Applications/PetClinic-ear/1.0")
```





Using the CLI

Exercise: Import a properties files



Using the CLI

Exercise: Import a properties files

- Create a file test.properties in the CLI directory with the following contents:

```
# Java properties file
key1 = value
key2    anotherValue
key3:overvalued
```



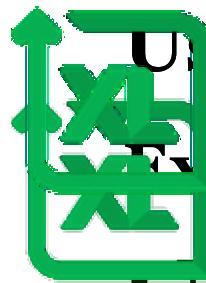
Using the CLI

Exercise: Import a properties files

- Load the properties file in the CLI using the following Jython snippet

```
from java.util import Properties
props = Properties()
file = open('test.properties', 'r')
props.load(file)
dict = factory.configurationItem("Environments/DemoDictionary","u
                                {"entries":props})
repository.create(dict)
```





Using the CLI

Exercise: Import a properties files

- The script should create a dictionary **Environments/testDict** with the properties you just loaded.
- Create a **importProperties.py** script that contains the commands you just did as a script
- Run **importProperties.py** using **cli.sh -f** and check the results.



Using the CLI Deployment Object

- Prepare deployments using the deployment object
 - Prepare initial or upgrade
 - Generate deployeds
 - Validate
 - Create deployment task



Using the CLI

Deployment : prepare

```
d = deployment.prepareInitial  
    ("Applications/PetClinic-ear/1.0", "Environments/Was-Dev")  
  
d = deployment.prepareUpgrade  
    ("Applications/PetClinic-ear/2.0", "Environments/Was-Dev/PetCl")
```





Using the CLI

Deployment : generate

```
d = deployment.prepareAutoDeployeds(d)

d = deployment.generateSelectedDeployeds
(["Applications/PetClinic-ear/1.0/PetClinic"], d)

d = deployment.generateSingleDeployed
("Applications/PetClinic-ear/1.0/PetClinic",
 "Infrastructure/was-70-nd/was-70Cell01/was-70Node01/existing

d = deployment.generateSingleDeployed
("Applications/PetClinic-ear/1.0/PetClinic",
 "Infrastructure/was-70-nd/was-70Cell01/was-70Node01/existing
 "was.EarModule",d)
```

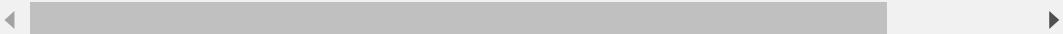


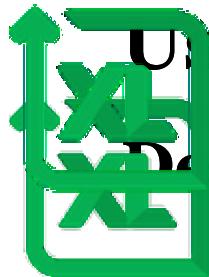


Using the CLI

Deployment : validate

```
d = deployment.validate(d)
for deployed in d.deployeds:
    if deployed.validations is not None and len(deployed.validati
        print deployed.validations
    else:
        print "No validation Errors"
```





Using the CLI

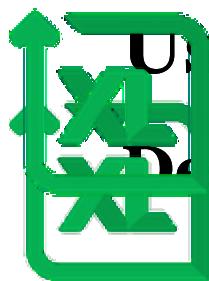
Deployment : tasks

- Create the task, like clicking Next in the UI

```
task = deployment.createDeployTask(d)
```

- The id will be needed to control the task via the deployit cli object

```
print task.id
```



Using the CLI

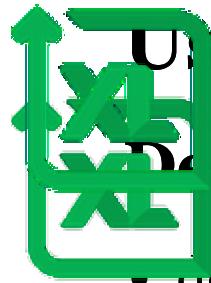
Deployment : tasks2

- Undeployments are started directly from a deployed application

```
task =  
deployment.undeploy("Environments/Wa-  
Dev/PetClinic-ear")
```

- Rollbacks are based on tasks

```
rollbackTask =  
deployment.rollbackTask(task.id)
```



Using the CLI

Deployit Object 1/3

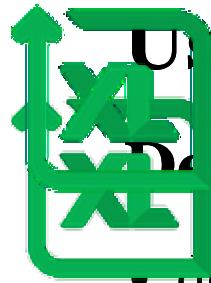
- `deployit.abortTask(String taskId) : void`
- `deployit.assignTask(String taskId, String owner) : void`
- `deployit.cancelTask(String taskId) : void`
- `deployit.createControlTask(Control control) : String`
- `deployit.createDiscoveryTask(ConfigurationItem ci) : String`
- `deployit.describe(String shortName) : void`
- `deployit.discover(ConfigurationItem ci) : List`
- `deployit.executeControlTask(String taskName, ConfigurationItem ci) : void`
- `deployit.importPackage(String importablePackage) : ConfigurationItem`



Using the CLI

Deployit Object 2/3

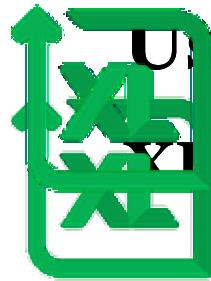
- `deployit.info() : ServerInfo`
- `deployit.listAllUnfinishedTasks() : List`
- `deployit.listImportablePackages() : List`
- `deployit.listUnfinishedTasks() : List`
- `deployit.prepareControlTask(ConfigurationItem ci, String taskName) : Control`
- `deployit.print(ConfigurationItem ci) : void`
- `deployit.retrieveDiscoveryResults(String taskId) : List`
- `deployit.retrieveTaskInfo(String taskId) : TaskInfo`
- `deployit.runGarbageCollector() : void`
- `deployit.shutdown() : void`



Using the CLI

Deployit Object 3/3

- `deployit.skipSteps(String taskId, Integer[] stepIds) : void`
- `deployit.startTask(String taskId) : void`
- `deployit.startTaskAndWait(String taskId) : void`
- `deployit.stopTask(String taskId) : void`



Using the CLI

XL Deploy Object examples

- Print CI reference documentation

```
deployit.describe("udm.Environment")
```

- Pretty print as tree structure

```
deployit.print(repository.read("Application/1.0"))
```

- Load from URL

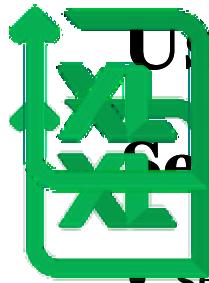
```
deployit.importPackage("http://nexus/com/1.0.dar")
```



Using the CLI

Security object

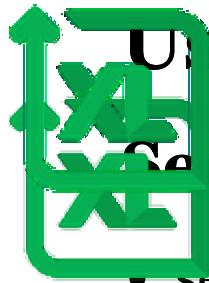
- Access to security system in Deployit
 - User management when not connected to LDAP
 - `security.assignRole(String roleName, List principals) : void`
 - `security.createUser(String username, String password) : User`
 - `security.deleteUser(String username) : void`
 - `security.getPermissions() : Map`



Using the CLI

Security object

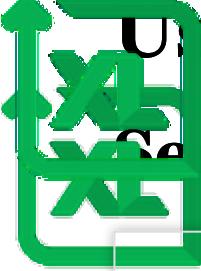
- security.getPermissions(String role) : Map
- security.getRoleAssignments(String roleName) : List
- security.getRoleNames() : List
- security.grant(String permission, String roleName, List configurationItems) : void
- security.grant(String permission, String roleName) : void
- security.hasPermission(String permission, String id) : boolean
- security.isGranted(String role, String permission) : boolean
- security.isGranted(String role, String permission, String id) : boolean
- security.login(String username, String password) : void
- security.logout() : void



Using the CLI

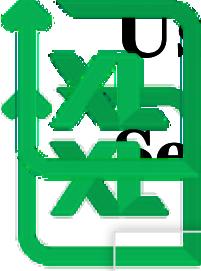
Security object

- security.modifyUser(User user) : void
- security.printPermissions() : void
- security.readUser(String username) : User
- security.removeRole(String roleName) : void
- security.revoke(String permission, String roleName, List configurationItems) : void
- security.revoke(String permission, String roleName) : void



Using the CLI Security object

```
security.createUser("tom","secret")
security.assignRole("developer",["tom"])
# Change password
user = security.readUser("tom")
user.password = "newSecret"
security.modifyUser(user)
```



Using the CLI Security object

```
security.logout()  
security.login("tom","newSecret")  
# Login will fail, because user tom not  
# granted login permission  
security.login("admin","admin")  
security.deleteUser("tom")
```



REST API

<http://www.xebialabs.com>



XL Deploy REST API

Introduction

- Another way to remotely drive XL Deploy
 - Uses HTTP / HTTPS
 - Basic Authentication
 - Content is XML
 - Returns HTTP status codes
 - **<https://docs.xebialabs.com/xl-deploy/latest/rest-api/>**



XL Deploy REST API

Introduction

Service	Description
ControlService	Provides access to control methods defined on CIs
DeploymentService	Deployment management
InspectionService	Inspects middleware
MetadataService	Provides XL Deploy's metadata: available types, permissions, and orchestrators
PackageService	Manages deployment packages
PermissionService	Manages permissions in XL Deploy
RepositoryService	Provides access to the XL Deploy repository



XL Deploy REST API

Introduction

Service	Description
RoleService	Manages the roles in XL Deploy's security system
ServerService	Services related to the operation of the XL Deploy server process
TaskBlockService	Manages tasks with blocks on the XL Deploy Server
TaskService	Manages tasks on the XL Deploy Server
UserService	Manages users in XL Deploy's internal user repository

XL Deploy REST API

Introduction

Configuration item

```
<udm.DeploymentPackage id="Applications/AnimalZoo-ear/1.0"
    token="032e29c4-94a0-4bdf-b489-a0c2fe5042a"
    created-by="admin"
    created-at="2014-08-18T13:42:09.555+0200"
    last-modified-by="admin"
    last-modified-at="2014-08-18T13:42:09.555+
<application ref="Applications/AnimalZoo-ear"/>
<satisfiesReleaseNotes>true</satisfiesReleaseNotes>
<deployables>
    <ci ref="Applications/AnimalZoo-ear/1.0/AnimalZooWeb"/>
    <ci ref="Applications/AnimalZoo-ear/1.0/AnimalZooFE"/>
    <ci ref="Applications/AnimalZoo-ear/1.0/AnimalZooBE"/>
</deployables>
</udm.DeploymentPackage>
```





XL Deploy REST API

Simple properties: Boolean, Int, String, and Enum

```
<dummy.MyType id="...">
  <myBoolean>true</myBoolean>
  <myInt>23</myInt>
  <myString>XL Deploy</myString>
  <myEnum>TELNET</myEnum>
</dummy.MyType id="...">
```



XL Deploy REST API reference

```
<myCI ref="Infrastructure/LocalHost/">
```



XL Deploy REST API

List and Sets

```
<mySetOfString>
  <value>value1</value>
  <value>value2</value>
</mySetOfString>

<myListOfString>
  <value>1</value>
  <value>2</value>
  <value>3</value>
</myListOfString>

<mySetOfCi>
  <ci ref="Infrastructure/LocalHost"/>
</mySetOfCi>

<myListOfCi>
  <ci ref="Infrastructure/LocalHost"/>
</myListOfCi>
```

XL Deploy REST API

Maps

```
<myMapOfStringString>
  <entry key="key1">value1</entry>
  <entry key="key2">value2</entry>
</myMapOfStringString>
```



XL Deploy REST API

Exercise: REST API



XL Deploy REST API

Exercise: REST API

- You can query the XL Deploy repository using the following URLs:
 - **http://[XLDEPLOY-SERVER]/deployit/repository**
- For example, to view the details of the was-vm host following URL:
 - **http://localhost:4516/deployit/repository/vm/**



XL Deploy REST API

Exercise: REST API

```
<overthere.SshHost id="Infrastructure/was-vm"
    token="b9c3c958-addf-471a-b543-388e4cc9b1ba">
    <tags/>
    <os>UNIX</os>
    <connectionType>SUDO</connectionType>
    <address>10.0.0.x</address>
    <port>22</port>
    <username>vagrant</username>
    <password>{b64}igltsEeChmolDD6xZCEQww==</password>
    <sudoUsername>root</sudoUsername>
</overthere.SshHost>
```



XL Deploy REST API

Exercise: REST API

- Try to query the repository for the details of:
 - The **vagrantCello1**
 - The **AppDataSource** (part of the **TableLister** application)

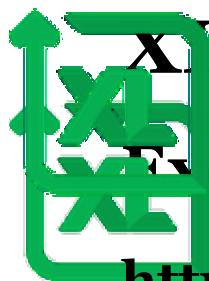
XL Deploy REST API

Exercise: REST API

<http://localhost:4516/deployit/repository/ci/1/vm/vagrantCell01>

```
<was.DeploymentManager id="Infrastructure/was-vm/vagrantCell01" token="bdbd46c1-293e-4cc1-8629-7878f7e28178">
    <tags/>
    <host ref="Infrastructure/was-vm"/>
    <wasHome>/opt/IBM/WebSphere/AppServer/profiles/Dmgr01</wasHome>
    <port>0</port>
    <username>admin</username>
    <password>{b64}igltsEeChmolDD6xZCEQww==</password>
    <version>WAS_85</version>
    <nodeAgents>
        <ci ref="Infrastructure/was-vm/vagrantCell01/vagrantNode01"></ci>
    </nodeAgents>
    <clusters>
        <ci ref="Infrastructure/was-vm/vagrantCell01/existing-cluster"></ci>
    </clusters>
</was.DeploymentManager>
```





XL Deploy REST API

Exercise: REST API

<http://localhost:4516/deployit/repository/ci/>

```
<was.MySQLDatasourceSpec id="Applications/TableLister/1.0/AppData
token="bdbd46c1-293e-4cc1-8629-7878f7e28178">
  <tags/>
  <description>AppDataSource</description>
  <jndiName>jdbc/mydatasource</jndiName>
  <jdbcProvider>MySQL Provider</jdbcProvider>
  <username>root</username>
  <password>{b64}igltsEeChmolDD6xZCEQww==</password>
  <URL>jdbc:mysql://localhost:3306/mysql</URL>
  <datasourceHelperClassname>com.ibm.websphere.rsadapter.Connec
  </datasourceHelperClassname>
</was.MySQLDatasourceSpec>
```





Configuration

<http://www.xebialabs.com>



XL Deploy Configuration

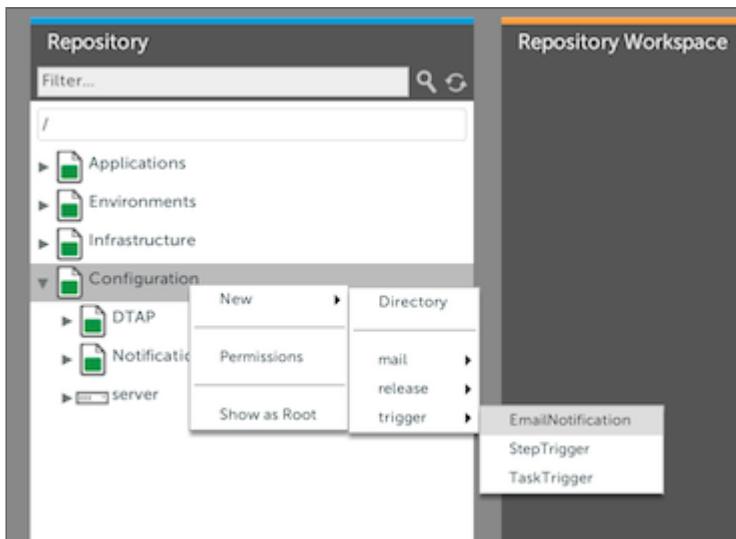
Introduction

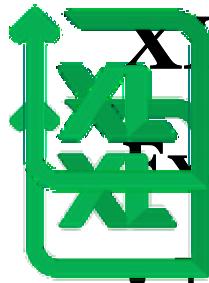
- Configuration on the CI-level properties
 - Configure deployeds for deployments
 - Configure other settings: Configuration node in Repository
 - Default values: **deployit-defaults.properties**
 - Type system: **ext/synthetic.xml**
 - Middleware Plugins

XL Deploy Configuration

Introduction

- Pipeline for Release Dashboard
- Mail server setup
- Triggers
- Other configuration CI's (from additional plugins)





XL Deploy Configuration Example

- trigger plugin
 - available-plugins/trigger-plugin-[versionNumber].jar
 - Step trigger
 - Task trigger



Config compare

<http://www.xebialabs.com>



Config Compare

Compare two or more configuration item (CI) trees.

- Only compares discoverable CIs.
- Two kinds of CI tree comparisons.
 - Live-to-live
 - Repo-to-live

A screenshot of the XL Deploy Config Compare interface. The top bar is dark grey with the word 'Compare' in white. Below it is a light grey search bar with a 'Search CIs' placeholder and a 'filter' input field with a magnifying glass icon. To the right of the search bar are two comparison entries. Each entry consists of an 'ID' dropdown, a 'Value' text input, and a 'Delete' button (an 'X'). The first entry has 'Cell-Dev' in the ID dropdown and 'Infrastructure/WAS/was85nd_1/vagrantCell01' in the Value input. The second entry has 'Cell-Tel' in the ID dropdown and 'Infrastructure/WAS/was85nd_2/vagrantCell01' in the Value input. At the bottom right is a dark grey 'Compare' button.



Config Compare Requirements

web browser: The following web browsers are supported:

- IE 10.0 or up
- Firefox
- Chrome
- Safari

Config Compare Report

Infrastructure/WAS/was85nd_1/vagrantCell01			
Infrastructure/WAS/was85nd_2/vagrantCell01			
Infrastructure/WAS/was85nd_3/vagrantCell01			
Cell2			Hide Identical CIs Expand all
Cell3			
/			
was.DeploymentManager			
was.OracleJdbcProvider			
Oracle JDBC Driver			
Path	Cell1	Cell2	Cell3
nativepath	Oracle JDBC Driver	Oracle JDBC Driver	Oracle JDBC Driver
classpath	—	/usr/local/nativepath	/usr/ bin /local/nativepath
description	Oracle JDBC Driver 1	Oracle JDBC Driver #2	Oracle JDBC Driver #3
app001-dev			
was.ManagedServer			
cluster001-dev			
was.Cluster			
db-master			
was.ManagedServer			



Config Compare

Report

- A **check mark** to the right a row indicates that the CIs are the same in all compared trees, while an exclamation mark indicates that there are differences.
- **Discoverables and labels:** The upper left table showing the selected configuration items and their labels
- **Path:** The ID of a configuration item relative to the ID of its root discoverable CI
- **Dash (-):** The item is null or missing. For example, the Oracle JDBC Driver CI nativepath property under Cell1 has no value.
- **Color and differences:** Green underscore text indicates additional characters, while red strikethrough texts indicates missing characters. The first available value is used as the benchmark for the comparison. For example, in the image above, the nativepath value under Cell2 is used as the benchmark.



Config Compare

Custom Matching Expressions

```
lhs.name[:lhs.name.rindex("-")] ==  
rhs.name[:rhs.name.rindex("-")]  
lhs.name[:lhs.name.rindex("-")] ==  
rhs.name[:rhs.name.rindex("-")]
```



Community plugins

<http://www.xebialabs.com>



Community plugins

Introduction

- Community plugins are not part of the standard XL Deploy product
- Written by XebiaLabs, Xebia consultants and XL Deploy users
- Check with author for support level
- Open repository on GitHub
 - <https://github.com/xebialabs-community>



Community plugins

Introduction

- Examples

- xld-docker-plugin
- xld-liquibase-plugin
- xld-smoke-test-plugin
- lock plugin
- manual-step plugin
- xld-puppet-plugin



Community plugins

Exercise: Smoketest



Community plugins

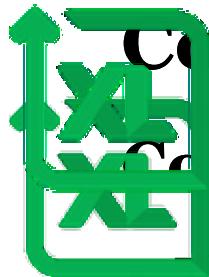
Exercise: Exercise: Smoketest

- Copy the xld-smoke-test-plugin into **/plugins**
 - Create a new **smoketest.Runner** under **Infrastructure/**
 - Add the new Container to the **Development** environment
 - Duplicate the last deployment package and rename it to 5.x
 - Add a deployable **smoketest.HttpRequestTest**
 - Update the deployed application to 5.x and make sure the smoketest is successful.



Compare

<http://www.xebialabs.com>



Compare

Comparing CIs

- The repository acts as a version control system
 - Every change to every object in the repository is logged and stored.
 - Compare a history of all changes to every CI in the repository.
 - XL Deploy also retains the history of all changes to deleted CIs.



Compare

Comparing CIs

Two variants:

- Compare against other CIs
 - **Repository Browser > Right Click > Compare > With other CI**
 - Drag Comparison CIs into the comparison tab
- Compare against previous versions
 - **Repository Browser > Right Click > Compare > With previous Version**
 - Select different versions



Compare

Exercise: Compare CIs



Compare

Comparing CIs

Part 1

- Go into the Repository View
- Find a dictionary and change some values.
- Select Compare with previous version.

Part 2

- Go into the Repository View
- Duplicate a dictionary and change some values
- Select Compare with other CI



Satellite

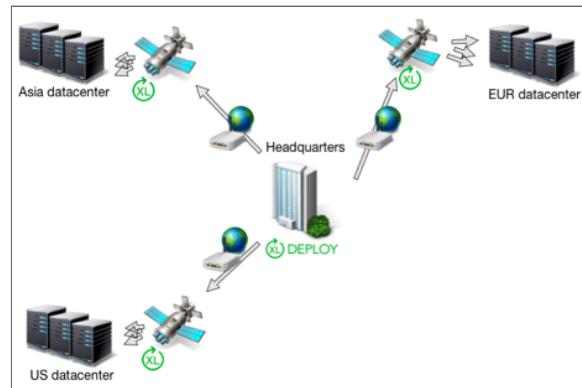
<http://www.xebialabs.com>



Satellite feature

The XL Deploy satellite feature lets you to execute deployments on remote data centers. It includes:

- An efficient system for transferring files
- Executing deployment tasks transparently on satellite servers
- Light process execution on data centers
- Secure communication between XL Deploy and satellite servers based on Transport Layer Security (TLS)





Satellite feature

Using XL Deploy satellites

XL Deploy satellites can be used to:

- Deploy an application in a cluster that is located around the world
- Deploy to production servers in a dedicated subnet that XL Deploy would normally not be allowed access for security reasons
- Deploy to an infrastructure that contains both Unix and Microsoft Windows hosts, with easier authentication between them



Satellite feature

Configuring a satellite

Options are configured in the `conf/application.conf` file on the satellite server:

- `akka.remote.netty.tcp.hostname`: Command handling channel
- `akka.remote.netty.tcp.port`: Command handling port (default 8380)
- `satellite.streaming.port`: File streaming port (this is automatically exchanged between XL Deploy and the satellite)
- `satellite.timeout.upload.idle`: File upload idle timeout (in milliseconds or seconds)
- `satellite.streaming.chunk-size`: File streaming chunk size (in bytes)
- `throttle` and `throttle-speed`: Streaming bandwidth limitation
- `satellite.workdir`: Directory where XL Deploy copies files



Satellite feature

Adding a satellite to the Repository

To add a satellite server to XL Deploy:

- In the Repository, right-click **Infrastructure** and select **New > xl > satellite**
- Enter the **Address** and **Port**; these must exactly match the akka.remote.netty.tcp.hostname and akka.remote.netty.tcp.port settings on the satellite server

To test the connection, right-click the satellite and select **Ping the satellite**.

#	Description	State
1	Checking connectivity with satellite satellite.xebialabs.uk	DONE



Satellite feature

Synchronizing satellites

Before XL Deploy executes a deployment plan on a satellite, it checks if any plugins are missing or out-of-date. If any are, XL Deploy stops the deployment, and you must synchronize the satellite before continuing.

To synchronize a satellite, right-click it in the Repository and select **Synchronize plugins satellite**.

#	Description	State
1	Synchronizing plugins on satellites satellite.xebialabs.uk	DONE
2	Cleaning up task file on satellite satellite.xebialabs.uk	DONE
3	Checking tasks running on satellite satellite.xebialabs.uk	DONE
4	Waiting for XL-Satellite to be fully restarted.	DONE



Satellite feature

Assigning a satellite to a host

- In the Repository, double-click and host and go to the **Advanced** tab
- Select the satellite

The screenshot shows a software interface titled 'Create localhost'. At the top, there are fields for 'Id' (set to 'Infrastructure/localhost'), 'Name' (set to 'localhost'), and 'Type' (set to 'overthere.LocalHost'). Below these, there are three tabs: 'Common' (selected), 'Advanced', and 'Deployment'. Under the 'Advanced' tab, there are three sections: 'Temporary Directory Path' (empty), 'Staging Directory Path' (empty), and 'Satellite' (a dropdown menu set to 'satellite.xebialabs.uk').



Continuous Integration

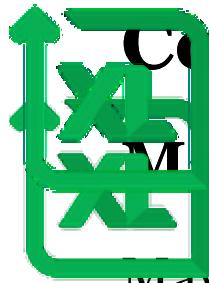
<http://www.xebialabs.com>



Continuous build/integration

Introduction

- Maven
- Jenkins
- Puppet
- TFS



Continuous build/integration

Maven

Maven plugin features:

- Import a generated DAR file into a remote XL Deploy server
- Deploy the DAR to a single environment
- Undeploy the deployment package from the target environment
- Test mode



Continuous build/integration

Maven

~, .m2/settings.xml:

```
<servers>
  <server>
    <id>nexus-dexter-releases</id>
  </server>
  <server>
    <id>deployit-credentials</id>
  </server>
</servers>
```



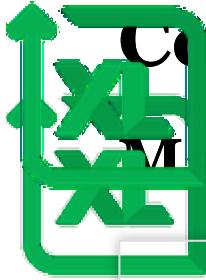
Continuous build/integration

Maven

pom.xml (parent):

```
<pluginRepositories>
  ...
    <pluginRepository>
      <id>nexus-dexter-releases</id>
      <url>http://nexus.xebialabs.com/nexus/content/repositories
    </pluginRepository>
  ....
</pluginRepositories>
```



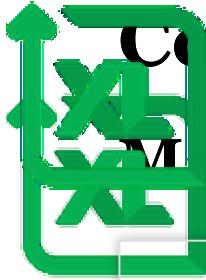


Continuous build/integration

Maven pom.xml 1/5

```
<plugin>
    <groupId>com.xebialabs.deployit</groupId>
    <artifactId>maven-deployit-plugin</artifactId>
    <version>4.5.1</version>
    <executions>
        <execution>
            <id>deployit-plugin-test</id>
            <phase>pre-integration-test</phase>
            <goals>
                <goal>deploy</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <environmentId>Environments/continuous-int</environmentId>
    </configuration>
</plugin>
```





Continuous build/integration

Maven pom.xml 2/5

```
<configuration>
    <environmentId>Environments/continuous-int</environmentId>
    <generateDeployedOnUpgrade>true</generateDeployedOnUpgrade>
    <deletePreviouslyDeployedDar>true</deletePreviouslyDeployedDa
    <port>4516</port>
    <serverAddress>localhost</serverAddress>
    <applicationName>${project.artifactId}</applicationName>
    <deploymentPackageProperties>
        <!-- Enter any deployment package level properties -->
        <orchestrator>default</orchestrator>
        <someCustomUdmVersionProperty>true</someCustomUdmVersionP
    </deploymentPackageProperties>
    ...

```



Continuous build/integration

Maven pom.xml 3/5

```
<deployables>
  <jee.War name="petclinic" groupId="org.xebialabs.training" ar
    <tags>
      <value>front-server</value>
    </tags>
  </jee.War>
  ...
  <!-- Other deployables -->
```





Continuous build/integration

Maven pom.xml 4/5

```
<glassfish.JdbcConnectionPoolSpec name="xlvalidation_pool">
    <datasourceclassname>oracle.jdbc.pool.OracleDataSource</datasourceclassname>
    <description>this is my connection pool</description>
    <driverclassname>oracle.jdbc.driver.OracleDriver</driverclassname>
    <datasourceName>OracleConnectionPoolDataSource</datasourceName>
    <statementTimeout>180</statementTimeout>
    <restype>javax_sql.DataSource</restype>
    <url>jdbc\oracle\thin\:@//{{MY_ORACLE_HOST}}\:{{MY_ORACLE_PORT}}</url>
</glassfish.JdbcConnectionPoolSpec>
```





Continuous build/integration

Maven pom.xml 5/5

```
<myext.CommonPropertiesSpec name="xld_example.properties"
    location="${project.build.directory}../xld_example.properties
</myext.CommonPropertiesSpec>
```





Continuous build/integration

Maven goals

Goal	Description
deployit:clean	Clean (Undeploy) the target environment
deployit:deploy	Deploy artifacts to the target environment
deployit:generate-deployment-package	Build up the XL Deploy deployment package
deployit:import	Import a deployment package into an XL Deploy server



Continuous build/integration

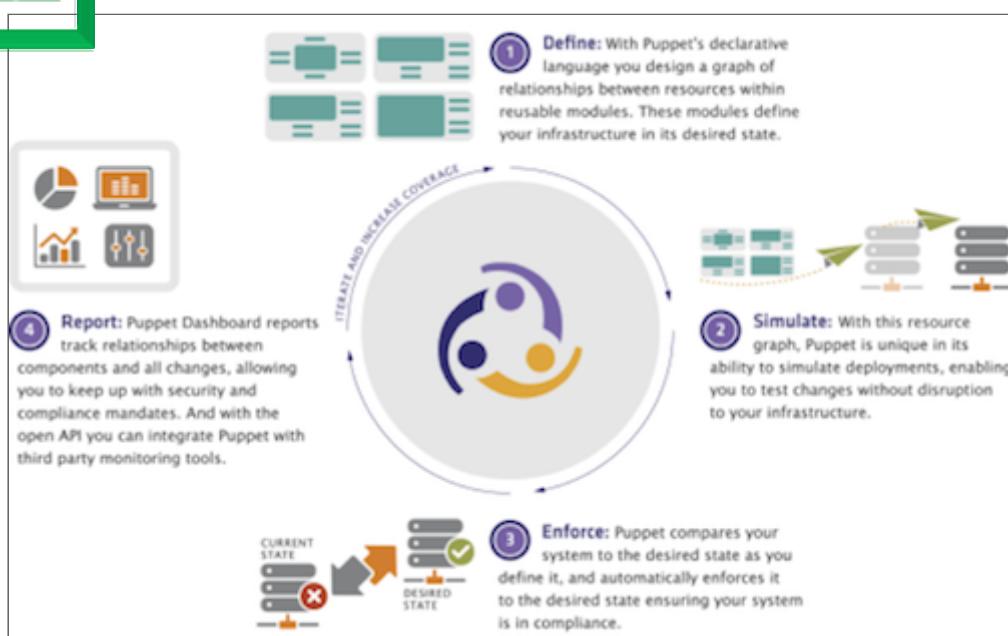
Puppet

- Manages infrastructure throughout its lifecycle:
 - Provisioning
 - Configuration
 - Patch management
 - Compliance
- Availability:
 - Open source
 - Commercial



<https://puppetlabs.com>

Continuous build/integration with Puppet





Continuous build/integration

Puppet module for XL Deploy (community)

- Used to interact with XL Deploy from Puppet
- Allows puppet scripts to register CIs in an XL Deploy server
- Support for:
 - Installing the XL Deploy server and CLI
 - Starting and stopping the XL Deploy server
 - Running a CLI script using the XL Deploy CLI
 - Registering a provisioned host/container and adding it to an environment



Continuous build/integration

Puppet module for XL Deploy, registering CIs

```
deployit_container { "Infrastructure/$environment/$fqdn/appserver-$hostname":
  type      => 'tomcat.Server',
  properties => {
    stopCommand  => '/etc/init.d/tomcat-appserver stop',
    startCommand => 'nohup /etc/init.d/tomcat-appserver start',
    home        => '/srv/tomcat/appserver',
    stopWaitTime => 0,
    startWaitTime => 10,
    deploymentGroup => $deployment_group,
  },
  server      => Deployit["xld-server"],
  environments => "Environments/$environment/App-$environment",
}

deployit_container { "Infrastructure/$environment/$fqdn/appserver-$hostname.$hostname.vh":
  type      => 'tomcat.VirtualHost',
  properties => {
    deploymentGroup => $deployment_group,
  },
  server      => Deployit["xld-server"],
  environments => "Environments/$environment/App-$environment",
}
```



Continuous build/integration

Jenkins

- Executes build jobs (Maven/Ant/Ivy)
- Monitors the execution and result
- Availability:
 - Open source
 - Commercial



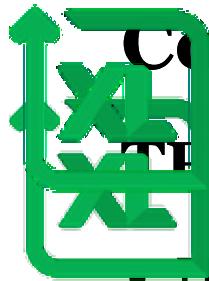
<http://jenkins-ci.org>



Continuous build/integration

Jenkins XL Deploy plugin

- Integrates in Jenkins GUI
- XL Deploy actions are added as **post-build action** to an existing job
- Support for:
 - Package
 - Import
 - Deploy



Continuous build/integration

- Handles build process
 - Workflow
 - XAML
 - Custom activities
 - Build controller
 - Support for
 - Import
 - Deploy



Agenda

<http://www.xebialabs.com>



Agenda

XL Deploy Training

Chapter	Description
Unified Deployment Engine	Introduction to engine phases
Using the rules engine	Writing an extension (rules engine)
Create your UI extensions	Extending the UI
Default properties	Using default properties
Type definition	Creating new synthetic types
Type modification	Extending synthetic types



Agenda

XL Deploy Training

Chapter	Description
Control Task	Create your own control tasks
UDM plugin API	UDM plugin API introduction
XML and script based extension	Generic plugin and script-based plugins
Plugin by example	Writing an extension (generic plugin)
Links	Useful links



XL Deploy sample code

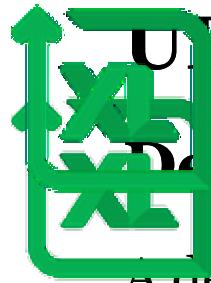
Note on copy/paste

- Copying and pasting the text from the slides into your xl-rules.xml or synthetic.xml files tends to corrupt the single- and double-quote characters on many systems.
- If you cut-and-paste the examples in today's slides, we recommend backspacing over each single- or double-quote and retyping it from your keyboard.
- Also, be wary of differences in the end-of-line characters on Linux and Windows systems.
Because the scripts in these exercises will run on a Linux system, they must use Linux-style end-of-line characters.
- Using an editor that knows the difference is highly recommended (e.g. Notepad ++).



UDM Engine

<http://www.xebialabs.com>



XL Deploy Engine

Deployment steps

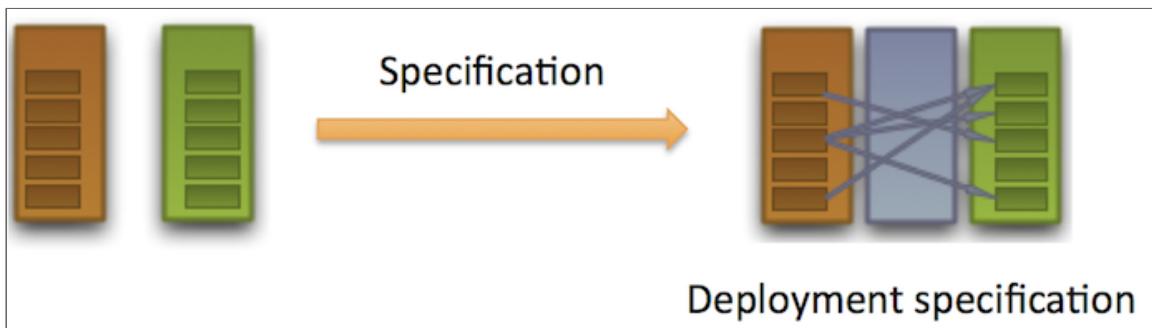
A deployment involves the following phases:

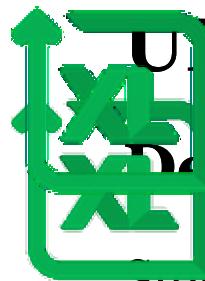
- Specification
- Delta analysis
- Orchestration
- Planning
- Execution

XL Deploy Engine

Deployment specification

Deployment begins with a deployment specification, where the engine gets the mapping of deployables to containers

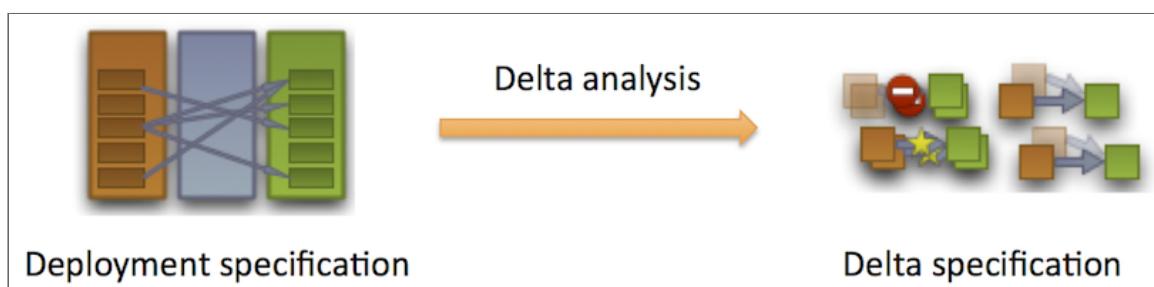




UDM Engine

Deployment – Delta analysis

Step 2: XL Deploy performs delta analysis to determine which deployeds need to be created, modified, destroyed, or left as-is by comparing the current state with the specified state

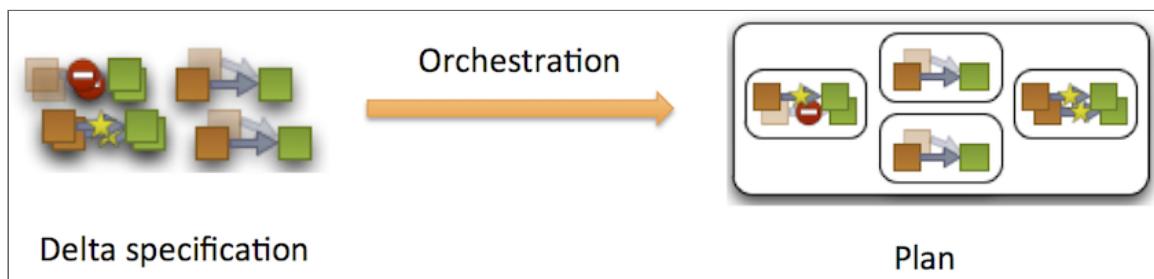




XL Deploy Engine

Deployment – Orchestration

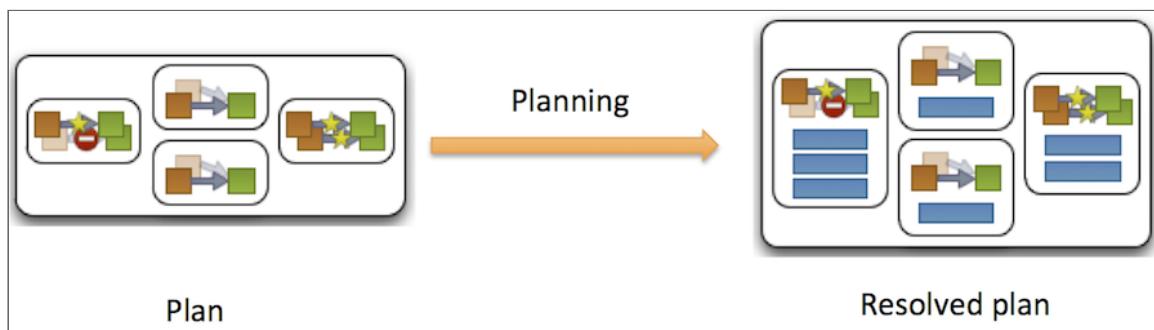
Step 3: One or multiple orchestrators are invoked to build the deployment plan (and sub-plans)



UDM Engine

Deployment – Planning

Step 4: The delta specification is translated into a deployment plan by invoking the plugins





UDM Engine

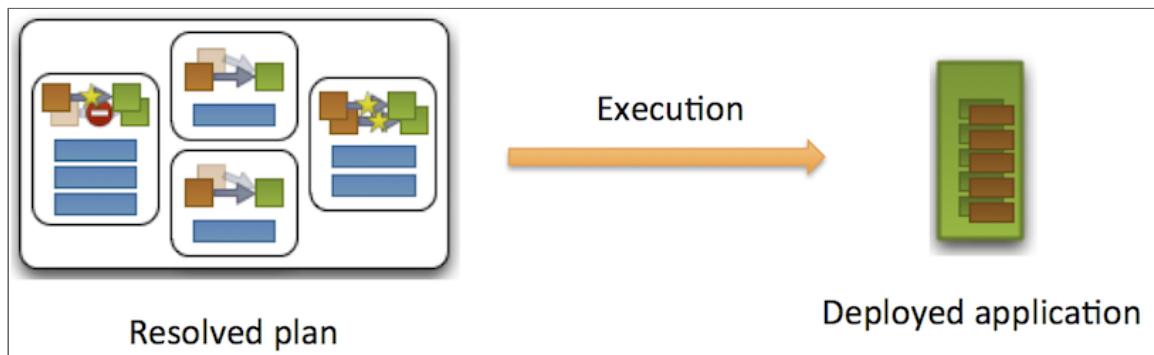
Deployment – Planning

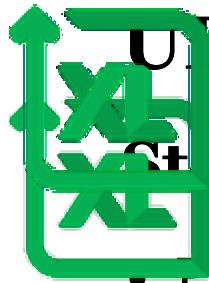
- Every deployed specifies what steps need to be performed to create, modify or destroy them; for example, deploy an EAR on WAS cluster, configure a datasource, or copy a folder to a host.
- A contributor can define steps that concern more than one deployed; for example, synchronize nodes in a WAS cell or restart WLS servers after a deployment

UDM Engine

Deployment – Execution

Step 5: The steps in the deployment plan are executed, in order





UDM Engine

Step orders

- PRE_FLIGHT (0)
- STOP_ARTIFACTS (10)
- STOP_CONTAINERS (20)
- UNDEPLOY_ARTIFACTS (30)
- DESTROY_RESOURCES (40)
- CREATE_RESOURCES (60)
- DEPLOY_ARTIFACTS (70)
- START_CONTAINERS (80)
- START_ARTIFACTS (90)
- POST_FLIGHT (100)



Rules

<http://www.xebialabs.com>



XL Deploy Rule System

Rule definitions (1/2)

XL Deploy's rules system allows you to use XML or Jython to specify the steps that belong in a deployment plan and how the steps are configured.

- Defined in `xl-rules.xml` (in `SERVER_HOME/ext` or in plugin JAR files)
- To reload rule definitions on the fly, set the `file-watch > interval` property in `conf/planner.conf`



XL Deploy Rule System

Rule definitions (2/2)

Each rule:

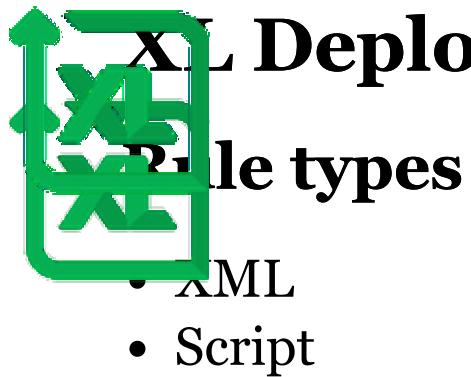
- Must have a name that is unique across the whole system
- Must have a scope
- Must define the conditions under which it will run
- Can use the planning context to influence the resulting plan

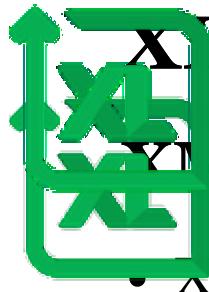
XL Deploy Rule System

Rule scope

- Pre-plan
- Deployed
- Plan
- Post-plan

XL Deploy Rule System





XL Deploy Rule System

XML rules

• XML

- Rule
- Condition
 - Type
 - Operation
 - Expression
- Steps

XL Deploy Rule System

XML rule example (1/2)

```
<rules xmlns="http://www.xebialabs.com/xl-deploy/xl-rules">
  <rule name="SuccessBaseDeployedArtifact" scope="deployed">
    <conditions>
      <type>udm.BaseDeployedArtifact</type>
      <type>udm.BaseDeployed</type>
      <operation>CREATE</operation>
    </conditions>
    <steps>
      <noop>
        <order>60</order>
        <description expression="true">'Dummy step for %s' % deployment
      </noop>
    </steps>
  </rule>
</rules>
```



XL Deploy Rule System

XML rule example (2/2)

```
<rules xmlns="http://www.xebialabs.com/xl-deploy/xl-rules">
  <rule name="SuccessProductionEnvironment" scope="post-plan">
    <conditions>
      <expression>
        "Production" in context.deployedApplication.environment.n
      </expression>
    </conditions>
    <steps>
      <noop>
        <order>60</order>
        <description>Success step in Production environment</desc
      </noop>
    </steps>
  </rule>
</rules>
```



XL Deploy Rule System

Script rules

- XML

- Rule
- Condition
 - Type
 - Operation
 - Expression
- planning-script-path

XL Deploy Rule System

Script rule example

```
<rules xmlns="http://www.xebialabs.com/xl-deploy/rules">
  <rule name="SuccessBaseDeployedArtifactPy" scope="deployed">
    <conditions>
      <type>udm.BaseDeployedArtifact</type>
      <operation>CREATE</operation>
    </conditions>
    <planning-script-path>planning/SuccessBaseDeployedArtifact.py
  </rule>
</rules>
```

Content of
planning/SuccessBaseDeployedArtifact.py:

```
step = steps.noop(description = "A dummy step to indicate that so
                      artifact was created on the target environment", order = 1
context.addStep(step)
```



XL Deploy Rule System

Exercise



XL Deploy Rule System

Rule exercise

Create a rule that adds a step to the deployment plan when an application is deployed to an environment with "Production" in its name.



XL Deploy Rule System

Rule Exercise Preparation

- Copy rules-demo-plugin from the <XLDEPLOY_SERVER>/samples to <XLDEPLOY_SERVER>/plugins
- Configure the file poller to poll the xl-rules.xml file for updates
- Restart XL Deploy
- Create a (dummy) environment with "production" in its name

XL Deploy Rule System

XML rule exercise solution

```
<rules xmlns="http://www.xebialabs.com/xl-deploy/xl-rules">
  <rule name="SuccessBaseDeployedArtifact" scope="post-plan">
    <conditions>
      <expression>"Production" in context.deployedApplication.env</expression>
    </conditions>
    <steps>
      <noop>
        <order>60</order>
        <description>Success step in Production environment</description>
      </noop>
    </steps>
  </rule>
</rules>
```





XL Deploy Rule System

Enhance your rules to accommodate rollbacks

```
<rules xmlns="http://www.xebialabs.com/xl-deploy/xl-rules">
  <rule name="SuccessBaseDeployedArtifact" scope="post-plan">
    <conditions>
      <expression>context.deployedApplication is not None and \
      "Production" in context.deployedApplication.environment.name</expression>
    </conditions>
    <steps>
      <noop>
        <order>60</order>
        <description>Success step in Production environment</description>
      </noop>
    </steps>
  </rule>
</rules>
```





UI Extensions

<http://www.xebialabs.com>

UI Extensions

You can extend XL Deploy by creating:

- Endpoints backed by Jython scripts
- UI screens that use these endpoints

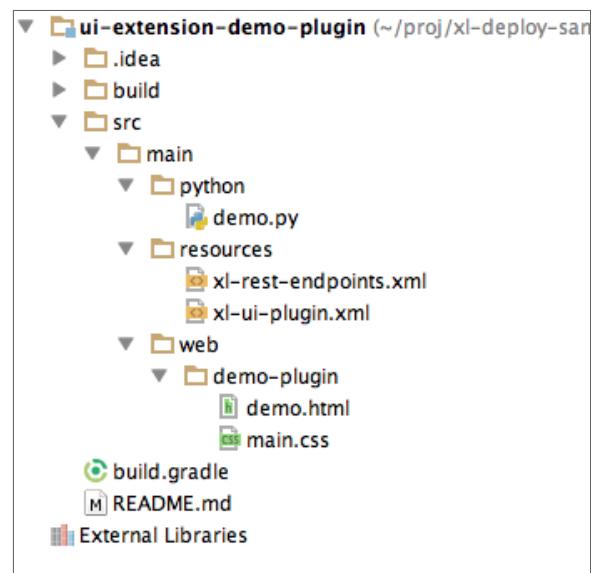
The following XML files tell XL Deploy where to find and how to interpret the content of an extension:

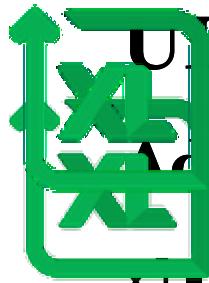
- `xl-rest-endpoints.xml` for adding new REST endpoints
- `xl-ui-plugins.xml` for adding new top-menu items

UI Extensions

Plugin structure

It is **recommended** that you create a folder under web with an unique name for each UI extension plugin, to avoid file name collisions.





UI Extensions

Adding menu items

xi-ui-plugins.xml:

```
<plugin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.xebialabs.com/deployit/ui-plugin"
        xsi:schemaLocation="http://www.xebialabs.com/deployit/ui-plugin"
        <menu id="test.demo" label="Demo" uri="demo.html" weight="12"
</plugin>
```



The URI property is the path name relative to the web directory at the root of the classpath. In this example, the demo.html file would need to be located at either /web/demo.html or /ext/web/demo.html.



XL Deploy Extensions

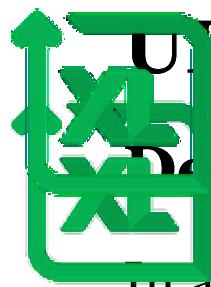
Declaring server endpoints

xl-rest-endpoints.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoints xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns="http://www.xebialabs.com/deployit/endpoints"
            xsi:schemaLocation="http://www.xebialabs.com/deployit/
<endpoint path="/test/demo" method="GET" script="demo.py" />
<!-- ... more endpoints can be declared in the same way ... ->
</endpoints>
```



After processing this file, XL Deploy creates a new REST endpoint, which is accessible via
`http://{xl-deploy-hostname}:{port}/{{context-path}}/api/extension/test/demo.`



UI Extensions

Declaring server endpoints

In a script you have access to the following objects:

- Request: JythonRequest
- Response: JythonResponse



UI Extensions

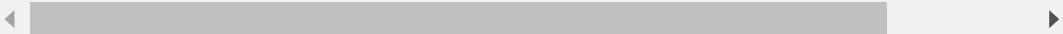
Declaring server endpoints: Example (1/2)

```
# This script is invoked when /api/extension/test/cis URL is requ
# We can split our code into modules
from ui_extension_demo.modules import repo

# repositoryService is one of the available XL Deploy services.
repository_helper = repo.RepositoryHelper(repositoryService)

root = request.query["root"]

# You can use logger object to output messages to XL Deploy log
logger.info("Requesting all the CIs under %s from the repository." % root)
```



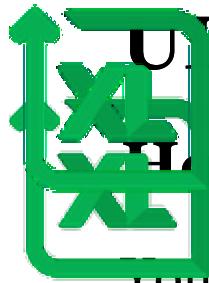


UI Extensions

Declaring server endpoints: Example (2/2)

```
# response.entity is what will be returned to the client as JSON  
# XL Deploy can automatically serialize list of ConfigurationItem  
# will be returned by the helper.  
response.entity = repository_helper.get_all_cis(root)
```





UI Extensions

Hooking it together

You can call two types of XL Deploy REST services from HTML pages:

- XL Deploy REST API
- REST endpoints created by your extension (it is described below how to declare them)

Authentication is not needed under HTML5. For legacy:

```
function authorize(xhr) {  
    if (parent && parent.getAuthToken) {  
        var base64 = parent.getAuthToken();  
        xhr.setRequestHeader("Authorization", base64);  
    }  
}
```



UI Extensions

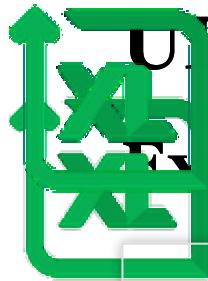
Exercise (advanced)



UI Extensions

Exercise (advanced)

Copy the ui_extension_demo plugin from XL_DEPLOY_SERVER/samples directory to the plugins directory. Then create a UI extension called 'Environments' that lists all environments you can deploy something to.

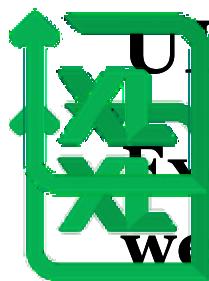


UI Extensions

Exercise solution, xl-ui-plugin.xml

```
<!-- in xl-ui-plugin.xml -->
<plugin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.xebialabs.com/deployit/ui-plugin"
  xsi:schemaLocation="http://www.xebialabs.com/deployit/ui-plugin x
<menu id="demo.environments" label="Environments" uri="environmen
</plugin>
```





UI Extensions

Exercise solution, web/environments.html

```
<html>
  <head>
    <script src="//code.jquery.com/jquery-1.10.2.js"></script>
    <script src="//code.jquery.com/ui/1.11.1/jquery-ui.js"></script>
    <script src="environments.js"></script>
  </head>
  <body>
    <h1>UI Extension to list Environments</h1>
    <input type="button" onclick="getEnvironments();" value="Get
  </body>
</html>
```





UI Extensions

Exercise solution, web/environments.js

```
function getEnvironments() {
    $.ajax({
        datatype: "json",
        url: "/api/extension/test/cis?root=Environments",
        crossDomain: true,
        // beforeSend: authorize,
        success: function(data) {
            $.each(data.entity, function(idx, val) {
                if (val.type == "udm.Environment") {
                    document.write ("<span>" + val.id + "</span><br><br>");
                }
            });
        }
    })
}
```



Types introduction

<http://www.xebialabs.com>



Types and Type Modifications

Introduction

Properties can be defined with the following declarations in `synthetic.xml`:

- **type** (new type)
- **type-modification**

```
<type type="myPrefix.myNewType" ...>
    <property name="myProperty" .../>
</type>

<type-modification type="myPrefix.myAlreadyDefinedType">
    <property name="myNewProperty" .../>
    <property name="myAlreadyDefinedProperty" .../>
</type-modification>
```



Types and Type Modifications

Property definition 1/4

Property	Required	Meaning
name	Yes	The name of the property to modify
kind	No	The type of the property to modify. Possible values are: enum, boolean, integer, string, ci, set_of_ci, set_of_string, map_string_string, list_of_ci, list_of_string, date (internal use only)
description	No	Describes the property
category	No	Categorizes the property. Each category is shown in a separate tab in the XL Deploy GUI



Types and Type Modifications

Property definition 2/4

Property	Required	Meaning
label	No	Sets the property's label. If set, the label is shown in the XL Deploy GUI instead of the name
required	No	Indicates whether the property is required or not
password	No	Indicates whether the property stores a password. If so, the property value is masked in the XL Deploy GUI and CLI

Property Required Meaning



size

No

Only relevant for properties of kind **string**. Specifies the property size. Possible values are: **default**, **small**, **medium**, **large**. Large text fields will be shown as a text area in the XL Deploy GUI



Types and Type Modifications

Property definition 3/4

Property	Required	Meaning
default	No	Specifies the default value of the property
enum-class	No	Only relevant for properties of kind enum. The enumeration class that contains the possible values for this property
referenced-type	No	The type of the referenced CI, relevant for properties of kind ci, set_of_ci or list_of_ci.

Property Required Meaning



as-
containment

No

Only relevant for properties of kind ci, set_of_ci or list_of_ci.
Indicates whether the property is modeled as containment in the repository. If true, the referenced CI or CIs are stored under the parent CI



Types and Type Modifications

Property definition 4/4

Property	Required	Meaning
hidden	No	Indicates whether the property is hidden. Hidden properties don't show up in the XL Deploy GUI. Note that a hidden property must have a default value
transient	No	Indicates whether the property is persisted in the repository or not

Property	Required	Meaning
-----------------	-----------------	----------------



inspectionProperty	No	Indicates that this property is used for inspection (discovery).
--------------------	----	--

XL Deploy Training

Property definition example

Extending a CI with a "notes" properties

```
<type-modification type="overthere.Host">
    <property name="notes" kind="string"/>
</type-modification>
```

XL Deploy Training

Property definition example

Hiding a CI property

```
<type-modification type="base.Host">
    <property name="connectionTimeoutMillis" kind="integer" defau
        hidden="true" />
</type-modification>
```





Type definitions

<http://www.xebialabs.com>



Type definition

Extension mechanism

- Type definitions allow you to create new types based on existing types:
 - By introducing declarations in `synthetic.xml`
 - By introducing new types in Java code
- Works by subclassing existing types
- Most new extensions should leverage the synthetic type system and the rules engine



Type definition

synthetic.xml

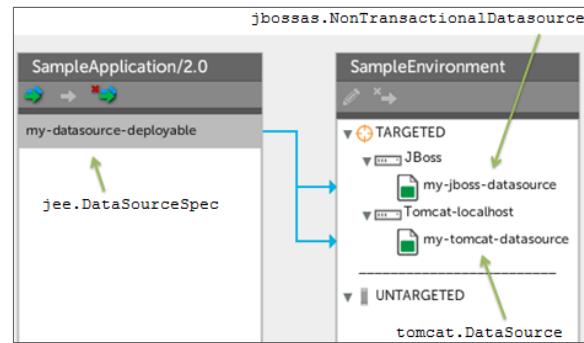
- Extension point is located at
`<XLDEPLOY_SERVER>/ext/synthetic.xml`
- Can be packaged in a JAR file and dropped into
`<XLDEPLOY_SERVER>/plugins` directory
 - Add `version.properties` file

```
plugin=database-plugin
version=4.0.1
```

Type definition Containers, deployables and deployeds

UDM basic principles:

- Deployables are packaged in deployment packages, and they act as a template or input definition for deployeds
- Related to each other through a deployed-container-deployed definition
- Deployable-to-deployed relationship is one-to-many





Type definition

Containers, deployables, and deployeds in synthetic.xml

The relationship among deployed, deployable, and container is specified through the type definition:

```
extends="myDeployedAncestorType"
deployable-type="myDeployableType"
container-type="myContainerType">
...
/>
```



Type definition

Exercise: Add a type



Type definition

Exercise: Add a type

- Go to the **ext** subdirectory of the XL Deploy server installation
- Edit the file called **synthetic.xml** and add these lines

```
<type type="was.NodeGroup" extends="was.Resource"
      deployable-type="was.NodeGroupSpec" container-type="was.Deplo
<generate-deployable type="was.NodeGroupSpec" extends="was.Deplo
<property name="createScript" hidden="true" default="was/contain
<property name="destroyScript" hidden="true" default="was/contai
<property name="description" required="false" />
<property name="nodeList" kind="list_of_ci" referenced-type="was
</type>
```

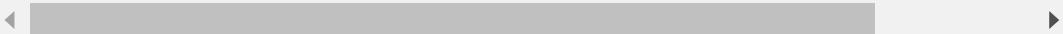


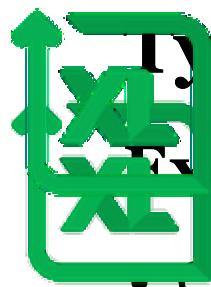
Type definition

Exercise: Add a type

- Create a new file called **create-nodegroup.py** and add these lines:

```
AdminTask.createNodeGroup(deployed.name,  
    '[-shortName -description %s]' % deployed.description)  
for node in deployed.nodeList:  
    AdminTask.addNodeGroupMember(deployed.name, '[-nodeName %s]' %
```





Type definition

Exercise: Add a type

- Create a new file called **destroy-nodegroup.py** and add these lines:

```
for node in deployed nodeList:  
    AdminTask.removeNodeGroupMember(deployed.name, '[-nodeName %s]'  
    AdminTask.removeNodeGroup(deployed.name)
```





Type definition

Exercise: Add a type

- Restart the XL Deploy server.
- Reload the XL Deploy GUI.



Type definition

Exercise: Add a type

- Go to the **Library** browser section
 - Create a new **udm.Application** with the name **WASNODEGROUP**
 - Create a new **udm.DeploymentPackage** with the name **WASNODEGROUP/1.0**
 - Under **WASNODEGROUP/1.0**, create a new **was.NodeGroupSpec** called **DemoNodegroup**
 - Edit the WAS environment to add the vagrantCello1 container to it



Type definition

Exercise: Add a type

- Go back to the **Deployment** tab
 - Configure the deployment of the **WASNODEGROUP/1.0** package to the WAS environment
 - Map the **was.NodeGroupSpec** to a server to get a **was.NodeGroup**
 - Edit the deployed to add vagrantNode01 to the nodeList
 - Start the deployment
 - Verify the created Nodegroup in the WAS console



Type modifications

<http://www.xebialabs.com>



Type modification extension mechanism

Type modifications allow you to modify existing types:

- By introducing new properties
- By modifying existing properties



Type modification

Exercise: Add a property



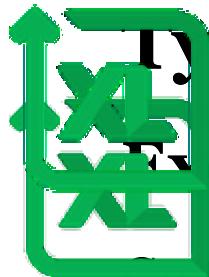
Type modification

Exercise: Add a property

- Go to the **ext** subdirectory of the XL Deploy server installation.
- Edit the file called **synthetic.xml** and add these lines:

```
<type-modification type="was.WmqQueue">
    <property name="specifiedPriority" kind="integer" />
</type-modification>
```

- Restart the XL Deploy server.
- Reload the XL Deploy GUI.



Type modification

Exercise: Add a property

Go to the **Repository** tab.

- Create a new **udm.Application** named **WASQ**
- Create a new **udm.DeploymentPackage** named **1.0**



Type modification

Exercise: Add a property

- Under **WASQ/1.0**, create a new **was.WmqQueueSpec** called **appointments** with the following properties:

Property	Value
Queue name	APPOINTMENTS.QUEUE
Priority	SPECIFIED
Specified priority	5
JNDI name	jms/appointments



Type modification

Exercise: Add a property

- Go back to the **Deployment** tab.
- Configure the deployment of the **WASQ/1.0** package to the WAS environment.
- Map the **was.WmqQueueSpec** to a server to get a **was.WmqQueue**
- Start the deployment!
- Verify the created queue in the WAS console and check that the specified priority is set to **42**.

Type modification

Exercise: Expand deployedsToDiscover

- Add this code to
`XL_DEPLOY_SERVER/ext/synthetic.xml.`

```
<type-modification type="was.DeploymentManager">
<property name="deployedsToDiscover" kind="set_of_string"
  default="was.VirtualHost,was.SharedLibrary,was.DerbyDatasource">
</type-modification>
```

- Restart the XL Deploy Server and re-discover
`vagrantCello1`.



Working with properties

<http://www.xebialabs.com>



Working with properties

Extension mechanism

- Properties can be defined on a new type through a **type** definition
- Properties can be added or modified on an existing type through a **type modification**



Working with properties

Property default value

- Properties default values can be defined in the type definition
- Default values can be overridden in `conf/deployit-defaults.properties`
 - Easy way to change a default value without a type modification
 - Interesting for temporary behavior modification
 - **Do not forget to carry back if you use multiple XL Deploy instances!**



Working with properties

Exercise: change a default value



Working with properties

Exercise: change a default value

- The Remoting plugin defines the SshHost type
- Observe the connectionType definition and specifically the default attribute

```
<type type="overthere.SshHost" extends="overthere.RemoteHost" ...  
...  
    <property name="connectionType" kind="enum"  
        enum-class="com.xebialabs.overthere.ssh.SshConnectionType"  
        required="true"  
        default="SFTP"  
        description="Type of SSH connection to create"/>
```





Working with properties

Exercise: Change a default value

- Stop the XL Deploy server
- Open `conf/deployit-defaults.properties` in your XL Deploy server installation directory
- Locate the line that reads:

```
#overthere.SshHost.connectionType=SFTP
```

- Change it to:

```
overthere.SshHost.connectionType=SCP
```

- Start the XL Deploy server
- Reload the XL Deploy user interface



Working with properties

Exercise: Change a default value

- In the GUI, go to the Repository tab
- Under the Infrastructure tree, create an `overthere.SshHost`
- The default value for the connection type is now SCP



Working with properties

Exercise: Add a default value



Working with properties

Exercise: Add a default value

- Go to the **Library** browser section
 - Create a new application with the name **WASDS**
 - Under **WASDS**, create a new DeploymentPackage with the name 1.0



Working with properties

Exercise: Add a default value

Under **WASDS/1.0**, create a new `was.DB2Type2DatasourceSpec` called **pets** with the following properties:

Property	Value
Database Name	pets
JNDI Name	jdbc/pets
JDBC Provider	DB2 Universal JDBC Driver Provider
Username	scott
Password	tiger



Working with properties

Exercise: Add a default value

- Go to the Deployment tab
 - Set up the deployment of the **WASDS/1.0** application to your WAS environment
 - Map the **pets** datasource to a server and double click the Deployed.
 - You'll notice that you'll have to enter a value for the **Datasource Helper Classname** property before you can proceed.
 - Close the tab to cancel the deployment.



Working with properties

Exercise: Add a default value

- Open conf/deployit-defaults.properties in your XL Deploy server installation directory.
- Add the following line (one line):
was.DB2Type2Datasource.datasourceHelperClass=com.ibm.db2.jcc.DB2ConnectionPoolDataSource
- Restart the XL Deploy server.
- Reload the XL Deploy user interface.



Working with properties

Exercise: Add a default value

- Go back to the **Deployment** tab.
- Configure the deployment of the **WASDS/1.0** application to your WAS environment.
- Map the **pets** datasource to a server and double click the Deployed.
- The **Datasource Helper Classname** property will now have the default value.



Control tasks

<http://www.xebialabs.com>



Control Task

Introduction

- Actions that can be performed on middleware or middleware resources.
- To trigger a control task on a CI in the repository, do the following:
 - **List the Control Tasks for a CI.**
 - **Execute the Control Task on a CI.**
 - **Fill in parameters.**



Control Task

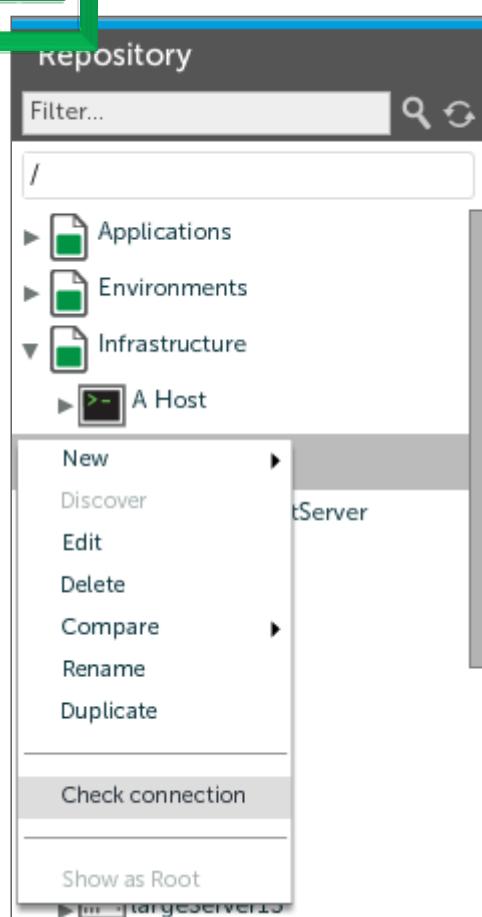
Introduction

```
class MyControlTasks {  
    public MyControlTasks() {}  
    @Delegate(name="startApache")  
    public List<Step> start(ConfigurationItem ci,  
                            String method,  
                            Map<String, String> arguments) {  
        // Should return actual steps here  
        return newArrayList();  
    }  
}  
  
<type-modification type="www.ApacheHttpdServer">  
  <method name="startApache" label="Start the Apache webserver"  
         delegate="startApache" argument1="value1" argument2="value2"/>  
</type-modification>
```



Control Task

Introduction





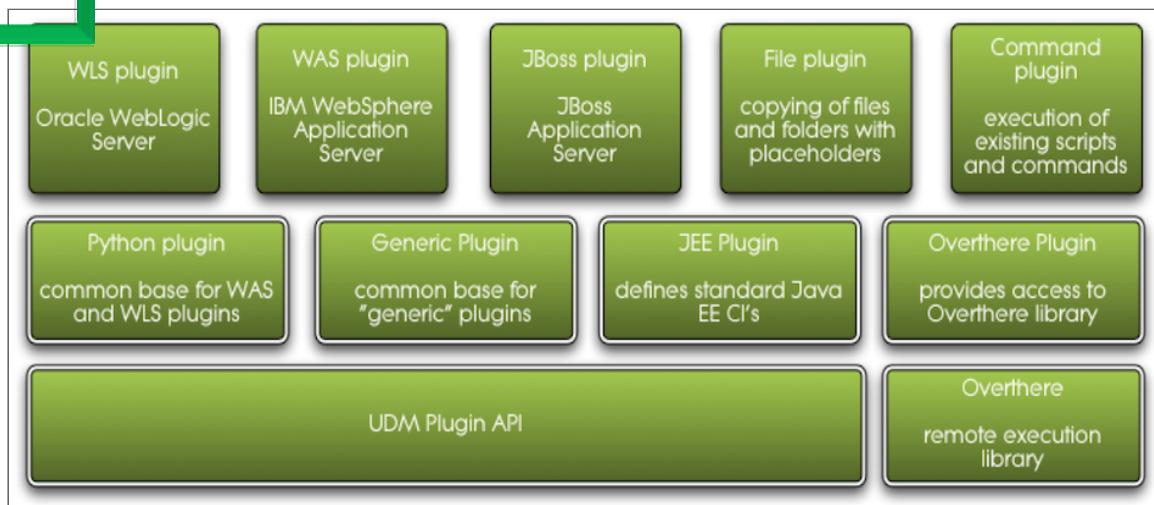
Plugin API

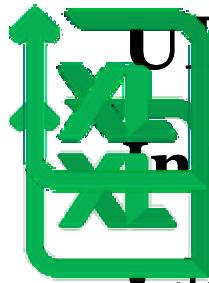
<http://www.xebialabs.com>



UDM Plugin API

Introduction





UDM Plugin API

Introduction

- Java Plugin API built around UDM concepts.
- Configuration items implement **ConfigurationItem** interface.
- Properties are annotated with **@Property** annotation.
- Steps implement the **Step** interface.
- Contributors are methods with an annotation **@Create**, **@Modify**, **@Destroy** or **@Contributor**.

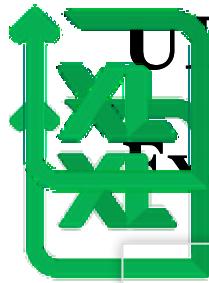


UDM Plugin API

Example CI definition

```
@Metadata(description="A simple example container that defines a  
public class ContainerExample extends BaseContainer implements Ho  
  
    @Property(description="Host on which the container resides",  
    private Host host;  
  
    @Property(label="administrative port",  
            description="Host on which the container resides",  
            defaultValue="8080")  
    private int port;
```





UDM Plugin API

Example contributor definition

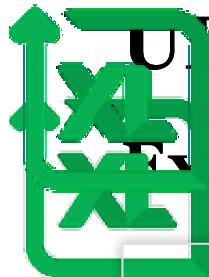
```
public class DeployedYakFile extends BaseDeployedArtifact<YakF  
    @Modify  
    @Destroy  
    public void stop(DeploymentPlanningContext result) {  
        result.addStep(new StopDeployedYakFileStep(this));  
    }  
  
    @Create  
    @Modify  
    public void start(DeploymentPlanningContext result) {  
        result.addStep(new StartDeployedYakFileStep(this));  
    }
```



UDM Plugin API

Example contributor definition

```
@Create  
public void deploy(DeploymentPlanningContext result) {  
    result.addStep(new DeployYakFileToServerStep(this));  
}  
  
@Modify  
public void upgrade(DeploymentPlanningContext result) {  
    result.addStep(new UpgradeYakFileOnServerStep(this));  
}  
  
@Destroy  
public void destroy(DeploymentPlanningContext result) {  
    result.addStep(new RemoveYakFileFromServerStep(this));  
}
```



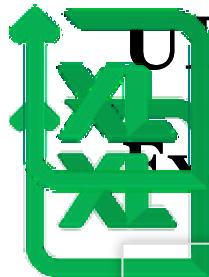
XL Deploy Plugin API

Example step

```
import com.xebialabs.deployit.plugin.api.flow.Step;
import com.xebialabs.deployit.plugin.api.flow.StepExitCode;

public class DeployYakFileToServerStep implements Step{

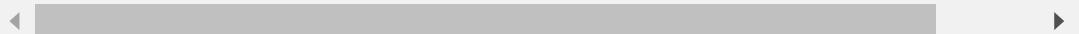
    public DeployYakFileToServerStep(DeployedYakFile file){
        ...
    }
}
```



UDM Plugin API

Example step

```
public StepExitCode execute(ExecutionContext exCtx) throws Except  
    return StepExitCode.SUCCESS;  
}  
  
public String getDescription() {  
    return "Step DeployYakFileToServerStep ";  
}  
  
public int getOrder() {  
    return 40;  
}
```





UDM Plugin API

Type definition in XML

- New types can be defined in `synthetic.xml`
- Note: New types must extend an existing type!

```
<type type="overthere.SshHost" extends="overthere.RemoteHost"
      description="Machine that can be connected to using SSH">
    <property name="protocol" default="ssh" hidden="true"/>
    <property name="connectionType" kind="enum"
              enum-class="com.xebialabs.overthere.ssh.SshConnection"
              required="true" default="SFTP"
              description="Type of SSH connection to create"/>
    <property name="address" kind="string" required="true" de...
    <property name="port" kind="integer" required="true" defa...
    <property name="username" kind="string" required="true" d...
    <property name="password" kind="string" required="false"
```





UDM Plugin API

Type modification in XML

Properties can be added to an existing type in `synthetic.xml`

```
<type-modification type="overthere.Host">
    <property name="tmpDeleteOnDisconnect" category="Advanced" ki
        required="false" default="true" hidden="true"
        description="If true, delete the temporary connection
                    directory when the connection is closed"/>
    <property name="tmpFileCreationRetries" category="Advance
        default="1000" hidden="true"
        description="Number of times Overthere attempts to c
                    a temporary file with a unique name"/>
</type-modification>
```





UDM Plugin API

Hidden properties and default values

- Hidden properties are not shown in the user interface or the CLI.
- Hidden properties are not stored per configuration item; they are always assigned their default value.
- Default values are defined in `conf/deployit-defaults.properties`.
- Default values can be changed by editing that file.



UDM Plugin API

Examples of default values

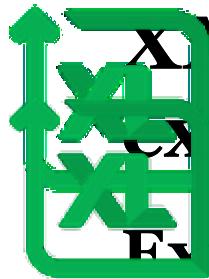
- Default SSH connection type (not a hidden field!):
`overthere.SshHost.connectionType=SFTP`
- Regular expression used to find the password for interactive SUDO:
`overthere.SshHost.sudoPasswordPromptRegex
[Pp]assword.*`
- Scripts used to deploy an application in WAS:
`was.Module.createScript=was/application/d
application.py`

Open `conf/deployit-defaults.properties` to see all 200+ properties!



Extensions

<http://www.xebialabs.com>



XML and script-based extensions

Extension mechanism

- wsadmin for IBM WebSphere (WAS)
- wlst for Oracle WebLogic (WLS)
- PowerShell for Windows
- XML and FreeMarker for the Generic plugin



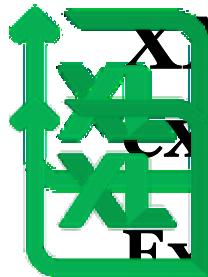
XML and script-based extensions

Example from the WAS plugin

Types are defined in XML, including what Python script to invoke:

```
<type type="was.JmsResource" extends="was.Resource" virtual="true
      <property name="additionalPropertiesNotToExpose" default=
      <property name="createScript" default="was/jms/create-jms
      <property name="destroyScript" default="was/jms/destroy-j
      <property name="modifyScript" default="was/jms/modify-jms
      ...

```



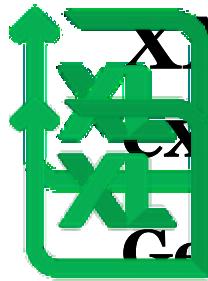
XML and script-based extensions

Example from the WAS plugin

- Behavior is defined in Python
- Part of library invoked by `create-jms-object.py` for WAS plugin

```
jmsProviderContainmentPath = '%s/JMSProvider:%s'  
    % (deployed.container.containmentPath, provider)  
jmsProviderId = validateNotEmpty(AdminConfig.getId(jmsProviderCon  
    "Cannot locate WebSphere JMS Provider '%s' in contain  
print "Creating '%s' for JMS provider '%s' with args '%s' in cont  
    % (deployed.wasType, jmsProviderId, args, deployed.container.  
jmsId = AdminConfig.create(deployed.wasType, jmsProviderId, args)  
if hasattr(deployed, 'customProperties'):  
    createJ2EEResourceProperties(jmsId, deployed)
```





XML and script-based extensions

Generic model plugin - Container

Container types (sample)

Type	Purpose
generic.Container	A generic container defined on a host



XML and script-based extensions

Generic model plugin - Deployables

Deployable types (samples)

Type	Purpose
generic.Archive	A generic, compressed binary artifact
generic.File	A generic binary artifact
generic.Folder	A generic folder artifact
generic.Resource	A generic resource specification



XML and script-based extensions

Generic model plugin - Deployeds

Deployed types (samples)

Type	Purpose
generic.CopiedArtifact	an artifact copied to the container, e.g. a Tomcat WAR deployment.
generic.ProcessedTemplate	a file generated from a template and then copied to the container e.g. a JBoss datasource.
generic.ExecutedScript	a resource or artifact deployed by executing a command, e.g. WebSphere Portal xml access import.



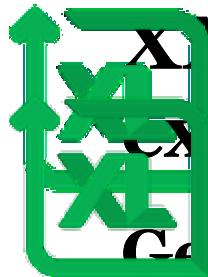
XML and script-based extensions

Generic model plugin - Container example

From the Database plugin:

```
<type type="sql.SqlClient" extends="generic.Container"
      description="Generic SQL client" virtual="true">
    <property name="clientWrapperScript" required="false"
              description="The OS-specific wrapper script that call
...

```



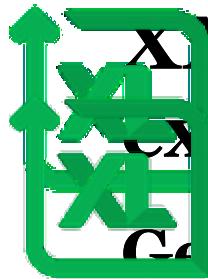
XML and script-based extensions

Generic model plugin - Template example

From the JBossAs plugin:

```
<type type="jbossas.Datasource" extends="jbossas.Resource" ... >
    <property name="targetFile" default="${deployed.deployabl
    <property name="template" default="jboss/datasource/templ
    <property name="inspectTemplate" default="jboss/datasourc
```



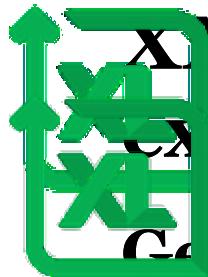


XML and script-based extensions

Generic model plugin - FreeMarker

The Generic plugin leverages FreeMarker:

<http://freemarker.sourceforge.net/docs/>



XML and script-based extensions

Generic model plugin - Template example

From the JBossAs plugin,
jboss/datasource/template.xml.ftl:

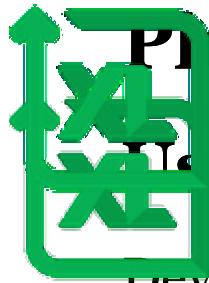
```
<datasources>
    <${deployed.dsType}>
        <#list deployed.parameters as dsp>
            <if (dsp == "xaDatasourceProperties")>
                <assign dsProps=deployed.xaDatasourceProperties />
                <list dsProps?keys as dsProp>
                    <xa-datasource-property name="${dsProp}">${dsProp}
                    </xa-datasource-property>
                </list>
            </if>
            <#elseif (dsp == "connectionProperties")>
                ...
            </elseif>
```





Plugin by example

<http://www.xebialabs.com>

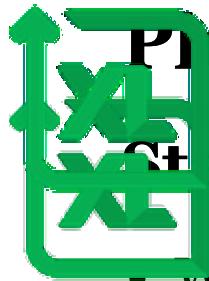


Plugin by example

The case

Develop a Tomcat plugin that:

- Deploys a WAR to a custom directory other than default webapps directory in the Tomcat installation
- Has the ability to specify the context root for the WAR
- Does not unzip the WAR
- Handles jee.War as deployable
- Can stop/start the Tomcat server for WAR deployments

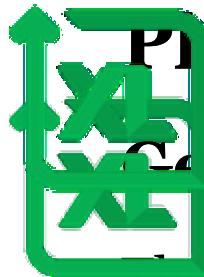


Plugin by example

Standard Tomcat behavior

- WARs are copied to <TC>/webapps
- Automatically unpacks the WARs
- The context root is the same as the WAR file name
- To specify custom context root, create file in <TC>/conf/Catalina/localhost (such as `petclinic.xml`) with content:

```
<Context unpackWAR="false"  
docBase="/apps/Petclinic-1.war"/>
```
- The path may be <TC>/Catalina/localhost depending on the installation



Plugin by example

Generic plugin features

The Generic plugin contains the following building blocks for defining deployment types:

Type	Description
generic.CopiedArtifact	To copy files and folders
generic.ProcessedTemplate	FreeMarker template
generic.ExecutedScript	To execute a script on the target machine



Plugin by example

Generic plugin features

Basic support for container operations:

Type	Description
generic.Container	Supports start/stop scripts and wait steps



Plugin by example

Design Tomcat plugin with Generic plugin

What can we use from the Generic plugin:

Type	Purpose
generic.CopiedArtifact	Copy the WAR to the custom directory
generic.ProcessedTemplate	Generate context XML
generic.ExecutedScript	Generate the context XML and do custom coping in the associated script to install the artifacts in the correct place
generic.Container	Model Tomcat and provide stop/start functionality



Plugin by example

generic.Container

generic.container:

- A container to which generic CIs can be deployed. Start, stop and restart behavior of this container can be controlled using the corresponding script properties.
- Type hierarchy:

```
generic.BaseGenericContainer >>
udm.BaseContainer >>
udm.BaseConfigurationItem
```
- Interfaces: udm.Taggable,
udm.ConfigurationItem, udm.Container,
generic.GenericContainer,
overthere.HostContainer

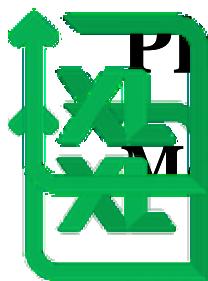


Plugin by example

generic.Container key properties

Property	Type/value	Purpose
restartScript	STRING	Classpath to the script used to restart the generic container.
startScript	STRING	Classpath to the script used to start the generic container.
stopScript	STRING	Classpath to the script used to stop the generic container.
restartOrder	INTEGER = 90	The order of the restart container step in the step list.
startOrder	INTEGER = 90	The order of the start container step in the step list.

Property	Type/value	Purpose
stopOrder 	INTEGER = 10	The order of the stop container step in the step list.



Plugin by example

Model the Tomcat server

synthetic.xml:

```
<type type="tc.Tomcat" extends="generic.Container">
    <property name="startScript" default="tc/start-tc" hidden="true"/>
    <property name="stopScript" default="tc/stop-tc" hidden="true"/>
    <property name="home"/>
</type>
```



tc/start-tc.sh:

```
/etc/init.d/tomcat6 start
```

tc/stop-tc.sh:

```
/etc/init.d/tomcat6 stop
```



Plugin by example

generic.CopiedArtifact

generic.CopiedArtifact:

- An artifact deployed on a generic container
- Type hierarchy:
`generic.AbstractDeployedArtifact >>
generic.AbstractDeployed >>
udm.BaseDeployed >>
udm.BaseConfigurationItem`
- Interfaces: `udm.EmbeddedDeployedContainer`,
`udm.Artifact`, `udm.Deployed`,
`udm.ConfigurationItem`,
`udm.DerivedArtifact`



Plugin by example

generic.CopiedArtifact key properties

Property	Type/value
deployable	CI
placeholders	MAP_STRING_STRING
targetFile	STRING

Property	Type/value
 targetDirectory	STRING
createTargetDirectory	BOOLEAN = false
restartRequired	BOOLEAN = false



Plugin by example

Model the WAR

```
<type type="tc.SimpleWarModule" extends="generic.CopiedArtifact"
    deployable-type="jee.War" container-type="tc.Tomcat">
    <property name="targetDirectory" default="${deployed.container}
        hidden="true"/>
    <property name="targetFile" default="${deployed.name}.war" hi>
    <property name="createTargetDirectory" default="true" kind="b
        hidden="true" required="false"/>
</type>
```





Plugin by example

generic.ProcessedTemplate

- A template deployed to a generic container
- Type hierarchy:
`generic.AbstractDeployedArtifact >>
generic.AbstractDeployed >>
udm.BaseDeployed >>
udm.BaseConfigurationItem`
- Interfaces: `udm.EmbeddedDeployedContainer`, `udm.Deployed`, `udm.ConfigurationItem`



Plugin by example

generic.ProcessedTemplate key properties

Property	Type/value	Purpose
targetDirectory	STRING	Path to which artifact must be copied to on the generic server.
template	STRING	Classpath to the FreeMarker template used to generate the content of the final text base artifact.

Plugin by example

Model the context 1/2

```
<type type="tc.DeployedContext" extends="generic.ProcessedTemplate">
  <deployable-type="tc.Context" container-type="tc.Tomcat">
    <generate-deployable type="tc.Context" extends="generic.Resource">
      <property name="template" default="tc/context.xml" hidden="true"/>
      <property name="targetDirectory" default="${deployed.container.home}/conf/Catalina/localhost" hidden="true"/>
      <property name="targetFile" default="${deployed.contextRoot}.xml" hidden="true"/>
      <property name="war" kind="ci" referenced-type="tc.SimpleWarModule" hidden="true"/>
      <property name="createOrder" kind="integer" default="49" hidden="true"/>
      <property name="restartRequired" default="true" kind="boolean" required="false"/>
      <property name="contextRoot"/>
    </generate-deployable>
  </deployable-type>
</type>
```



Plugin by example

Model the context 2/2

context.xml:

```
<Context unpackWAR="false"  
    docBase="${deployed.war.targetDirectory}/${deployed.war.targe  
</Context>
```





Plugin by example

generic.ExecutedScript

generic.ExecutedScript:

- A script executed on a generic container
- Type hierarchy: generic.AbstractDeployed
>> udm.BaseDeployed >>
udm.BaseConfigurationItem
- Interfaces: udm.EmbeddedDeployedContainer,
udm.Deployed, udm.ConfigurationItem



Plugin by example

generic.ExecutedScript key properties

Property	Type/value	Purpose
createOrder	INTEGER = 50	The order of the step in the step list for the create operation.
createScript	STRING	Classpath to the script that is uploaded and executed on the generic container for the create operation.
destroyOrder	INTEGER = 40	The order of the step in the step list for the destroy operation.

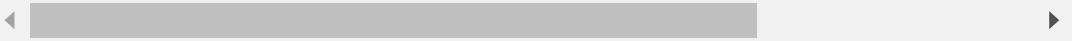
Property	Type/value	Purpose
 destroyScript	STRING	Classpath to the script that is uploaded and executed on the generic container for the destroy operation.

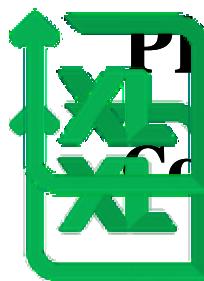


Plugin by example

Combining context and war 1/2

```
<type type="tc.ContextWarModule" extends="generic.ExecutedScript"
    deployable-type="jee.War" container-type="tc.Tomcat">
    <property name="contextRoot"/>
    <property name="warTargetDirectory" default="${deployed.container.home}/tmp/deployments/wars/<name>" hidden="true"/>
    <property name="ctxTargetDirectory" default="${deployed.container.home}/conf/Catalina/localhost<port>/contexts/<name>"/>
    <property name="templateClasspathResources" kind="set_of_string" value="true" default="tc/install-ctx.xml" hidden="true"/>
    <property name="createScript" default="tc/install-war.sh" hidden="true"/>
    <property name="destroyScript" default="tc/uninstall-war.sh" hidden="true"/>
    <property name="restartRequired" default="true" kind="boolean" value="true" required="false"/>
</type>
```





Plugin by example

Combining context and war 2/2

tc/install-war.sh.ftl:

```
cp ${step.uploadedArtifactPath} ${deployed.warTargetDirectory}  
cp ${step.remoteWorkingDirectory.path}/install-ctx.xml \  
    ${deployed.ctxTargetDirectory}/${deployed.contextRoot}.xm
```

tc/uninstall-war.sh.ftl:

```
rm ${deployed.warTargetDirectory}/${step.artifact.name}  
rm ${deployed.ctxTargetDirectory}/${deployed.contextRoot}.xml
```

tc/install-ctx.xml:

```
<Context unpackWAR="false"  
        docBase="${deployed.warTargetDirectory}/${step.artifact.name}"  
/>
```



Links

<http://www.xebialabs.com>

Links



- <http://xebialabs.com>
- <https://support.xebialabs.com>
 - See forums for announcements and feature requests
- <https://docs.xebialabs.com>
- <http://xebialabs-community.github.io>
- <http://blog.xebialabs.com>