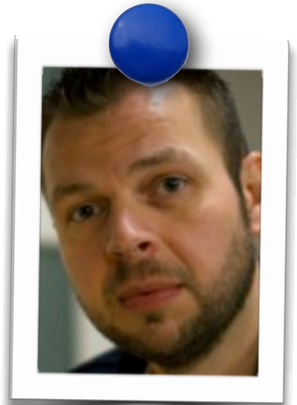
Building Modular Cloud Applications in Java

Lessons Learned

luminis



Bert Ertman

Fellow at Luminis in the Netherlands

JUG Leader for NLJUG and a Java Champion

@BertErtman

Paul Bakker

Architect at Luminis Technologies

@pbakker

a presentation for

geecon

The case

PulseOn

- Educational system focussed on personalized learning
- used in high schools in the Netherlands
- Expand to other countries in the near future

Re: Cloud Applications

Some characteristics:

- Application as a service over the internet
- Impact on some non-trivial non-functionals
  - Availability
  - Scalability
  - Extensibility

Observations

- Extremely agile
- Architecture (and code base) should be able to cope with change

The case for modularity

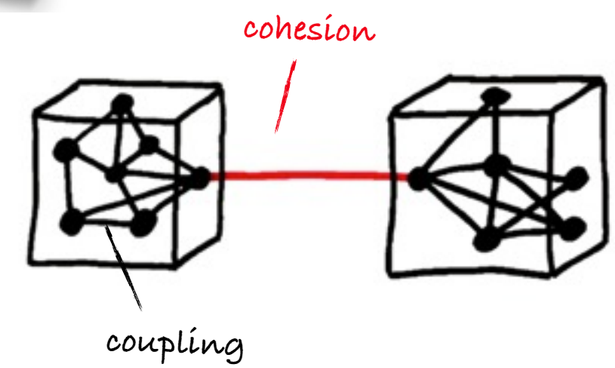
Modularity is the ultimate agile tool!

- Small, disposable, components
- Prevents code rot on the architectural level
- Isolate problems, focus work

What we learned about OO design in university :

Promote cohesion

Prevent (tight) coupling



What do we need?

design consequences →

Architectural focus on modularity

Runtime dynamic module framework

High-level enterprise APIs

← let's not reinvent the wheel

Right now, OSGi is the only option

OSGi != modularity

OSGi is the de-facto standard module system for Java

What about Jigsaw?

OSGi is the de-facto standard module system for Java

OSGi != modularity

remember!
Modularity is an architectural principle

What about Jigsaw?

OSGi is the de-facto standard module system for Java

OSGi != modularity

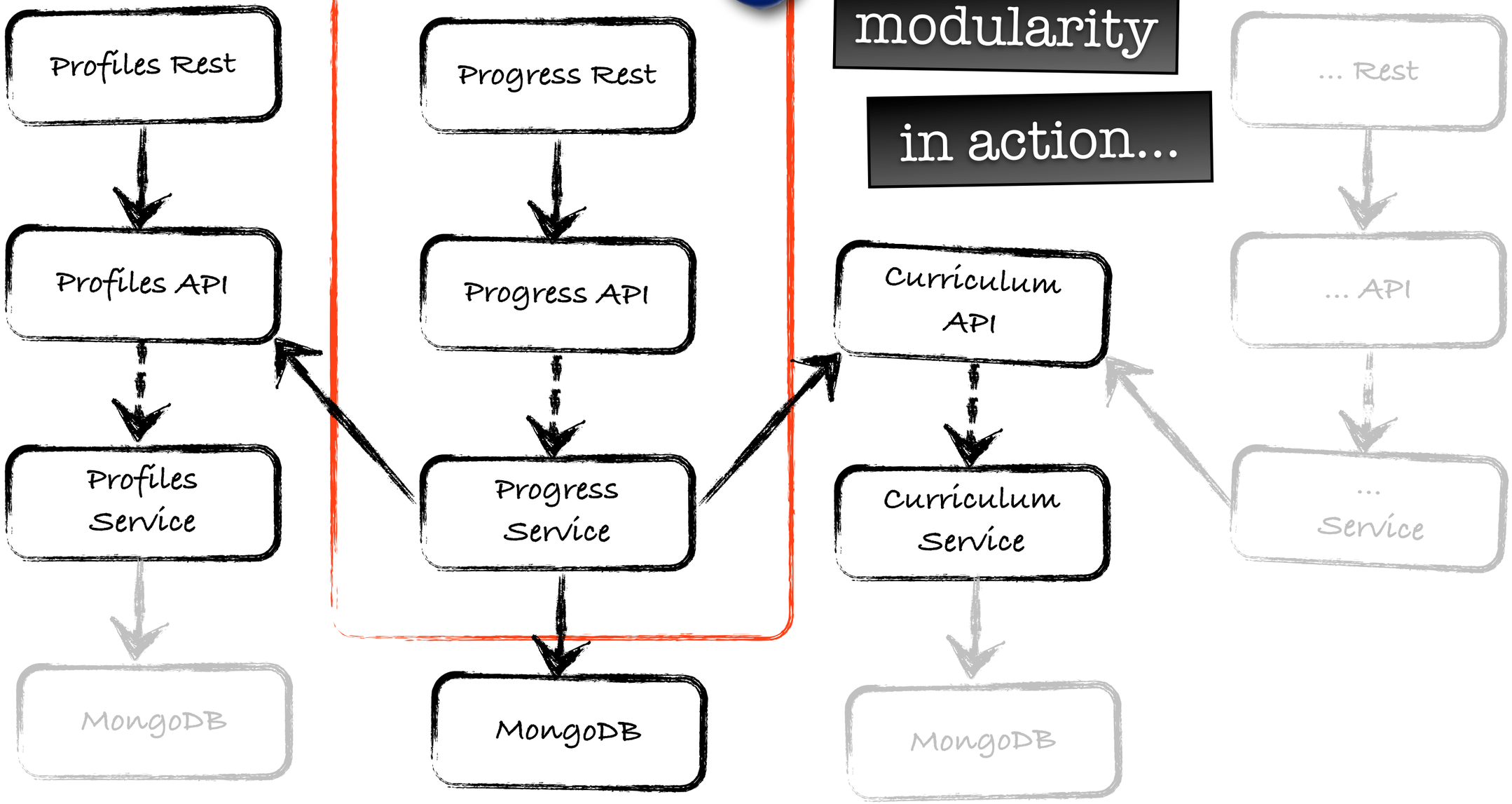
remember!
Modularity is an architectural principle

Modularity does not come for free with a framework

What about Jigsaw?

demo

modularity in action...



Back to the cloud...

Requirements:

- Modern web app
- UI mostly offloaded to clients or devices
- Document driven interaction
- Integration via REST API
- Web scale data store
- Multi-tenant
- Elasticity

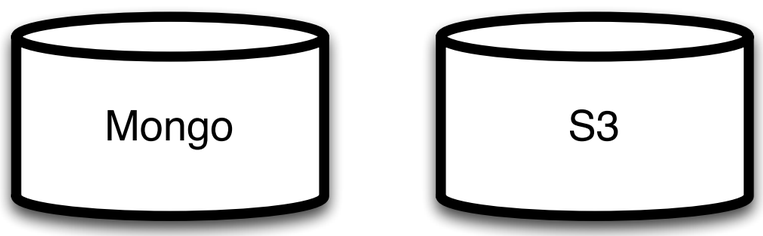
Typical architecture

HTML 5 + JavaScript

RESTful services

OSGi services

Apache Felix

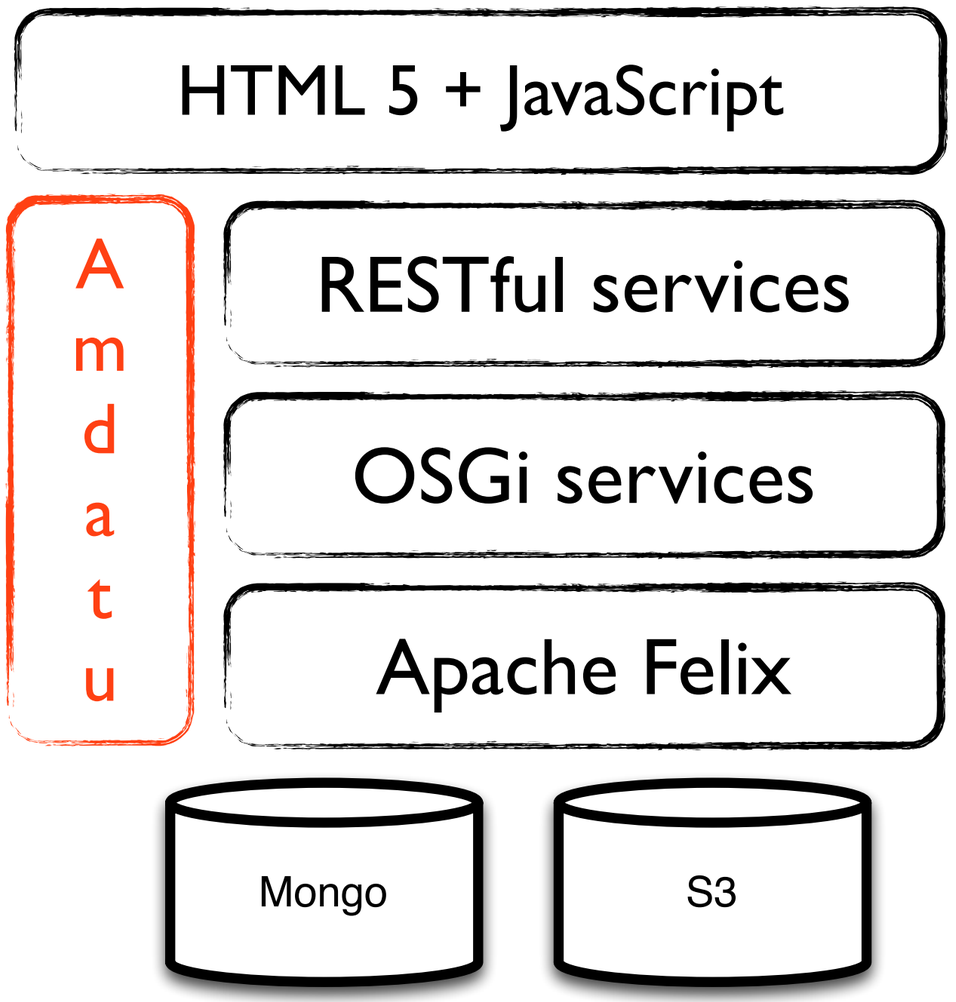


Let's Add AMDATU

to our stack

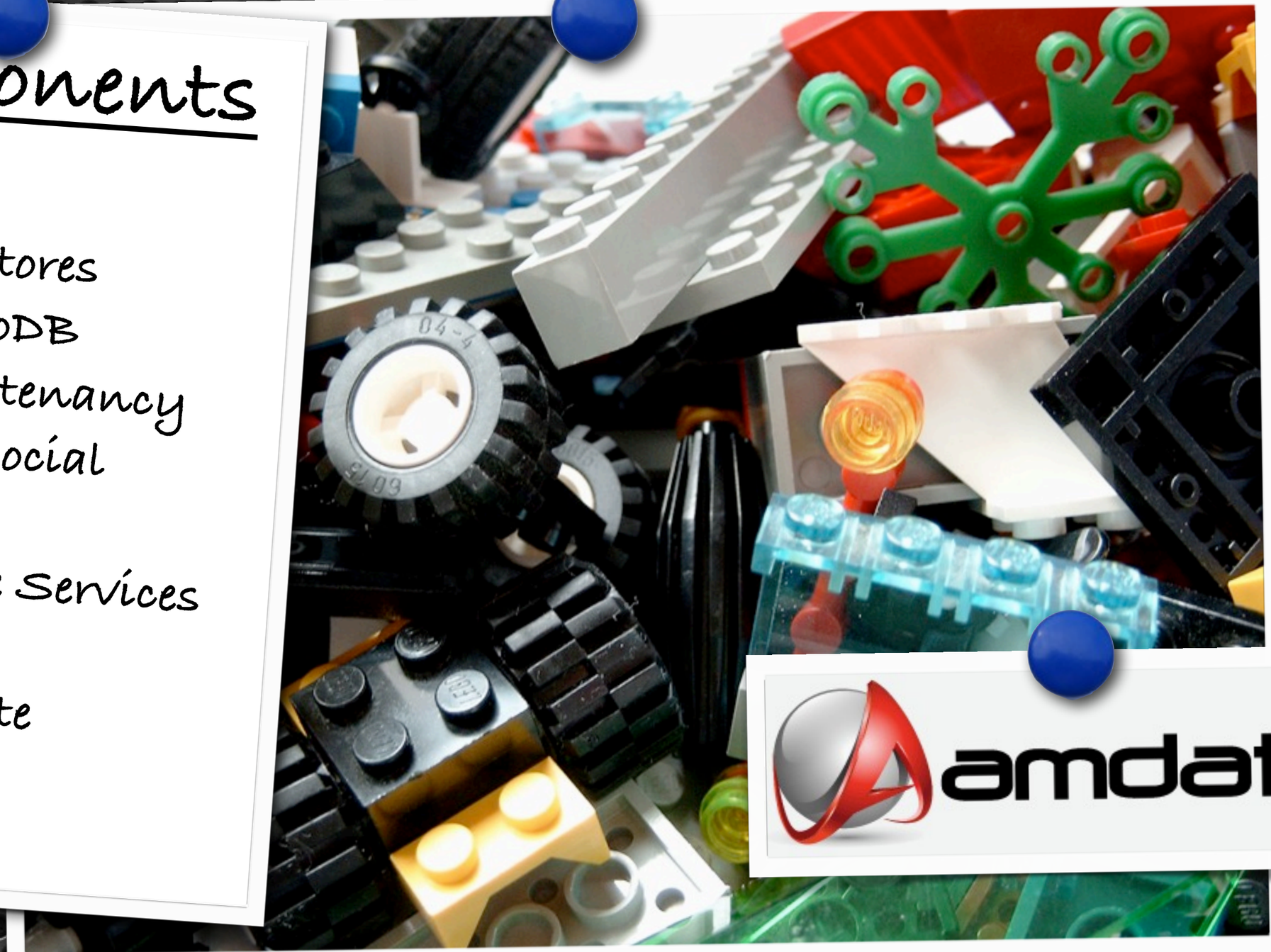
- ☑ Architectural focus on modularity
- ☑ Runtime dynamic services
- ☑ High level API

Typical architecture



Components

- Auth
- Blob stores
- MongoDB
- Multi-tenancy
- OpenSocial
- Search
- Remote Services
- REST
- Template
- Web
- ...



amdatu

What about deployment?

PaaS Offerings out there:

Not good:

- ☐ Proprietary platform
- ☐ Vendor lock-in
- ☐ White list / black list
- ☐ OS Limitations
- ☐ No support for modular runtime

CLOUD FOUNDRY

SOFTWARE force.com cloud platform

amazon

OPENSHIFT

Google App Engine

heroku

Windows Azure

Solution: roll our own PaaS

Modular PaaS:

- ☐ IaaS image
  Amazon EC2
- ☐ OSGi runtime
  Apache Felix
- ☐ Deployment/provisioning
  Apache ACE
- ☐ PaaS Building blocks
  Amdatu

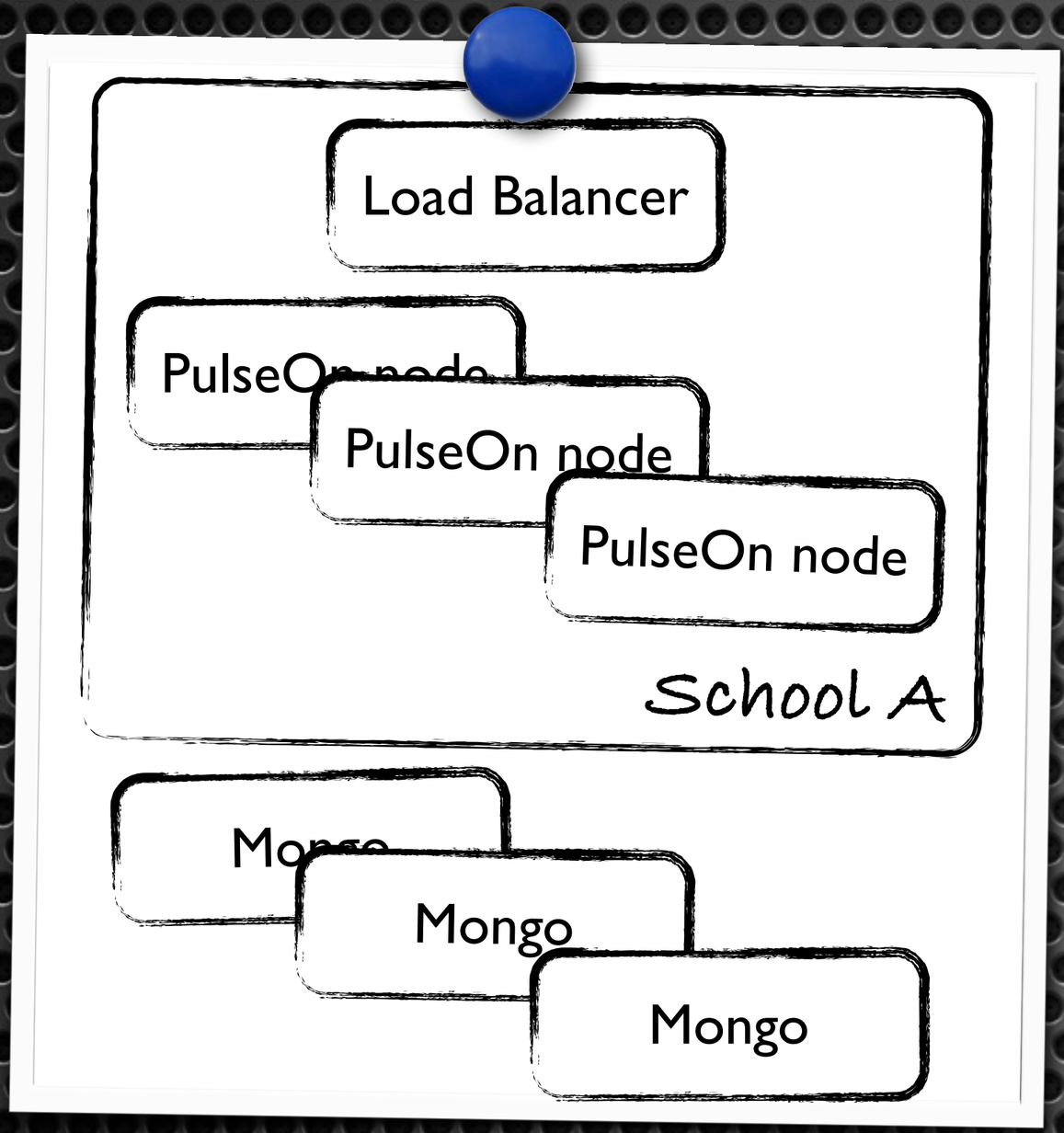
Pro:

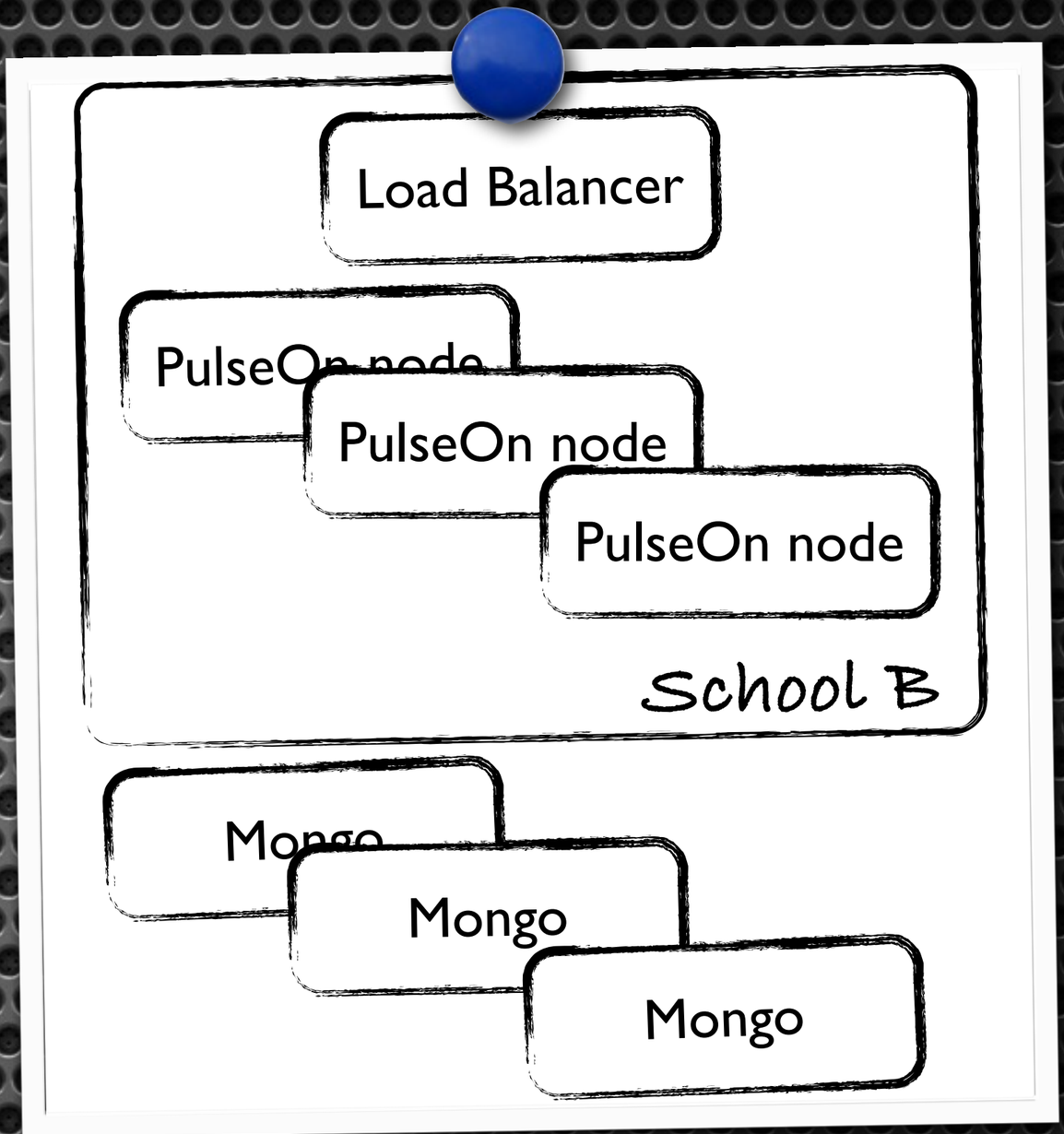
freedom

Con:

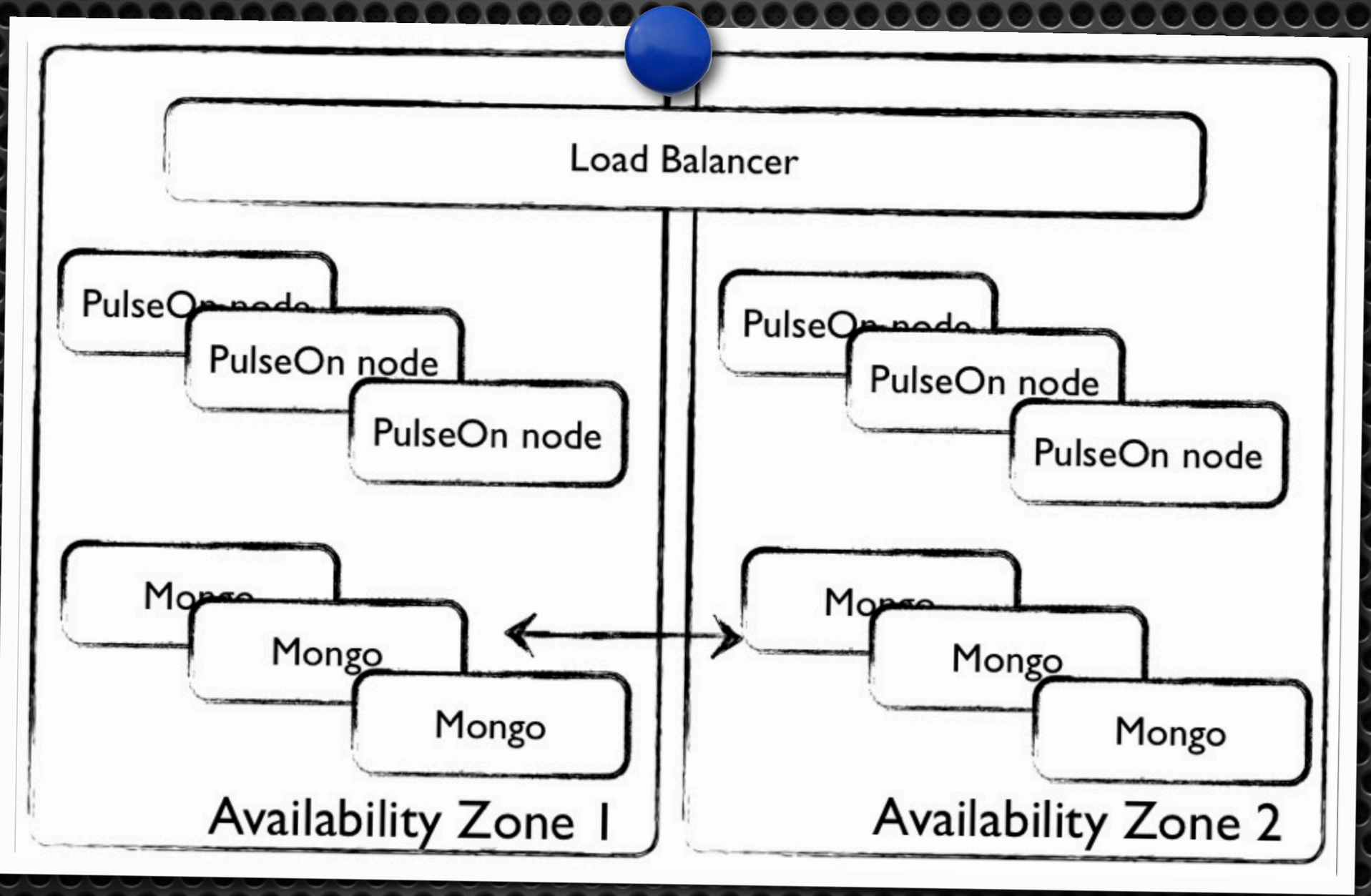
no vendor support

Deployment


Load Balancer
PulseOn node
PulseOn node
PulseOn node
School A
Mongo
Mongo
Mongo


Load Balancer
PulseOn node
PulseOn node
PulseOn node
School B
Mongo
Mongo
Mongo

Availability zones



Horizontal scalability

- Horizontal scaling requires stateless nodes
- HTML5 clients need less server side state
- Any state should go to some kind of store

Auto scaling

Considerable higher loads during school hours

Enough
capacity

Without paying
for idle servers
at night...

Cluster per school

Load Balancer

small node

Always use a load balancer because we don't want downtime during scaling

Early morning...

Load Balancer

small node

Early morning...

Load Balancer

small node

large node

large node

End of the day...

Load Balancer

small node

large node

large node

End of the day...

Load Balancer

small node

But how do we install

our software on a node?

Some numbers

190

bundles

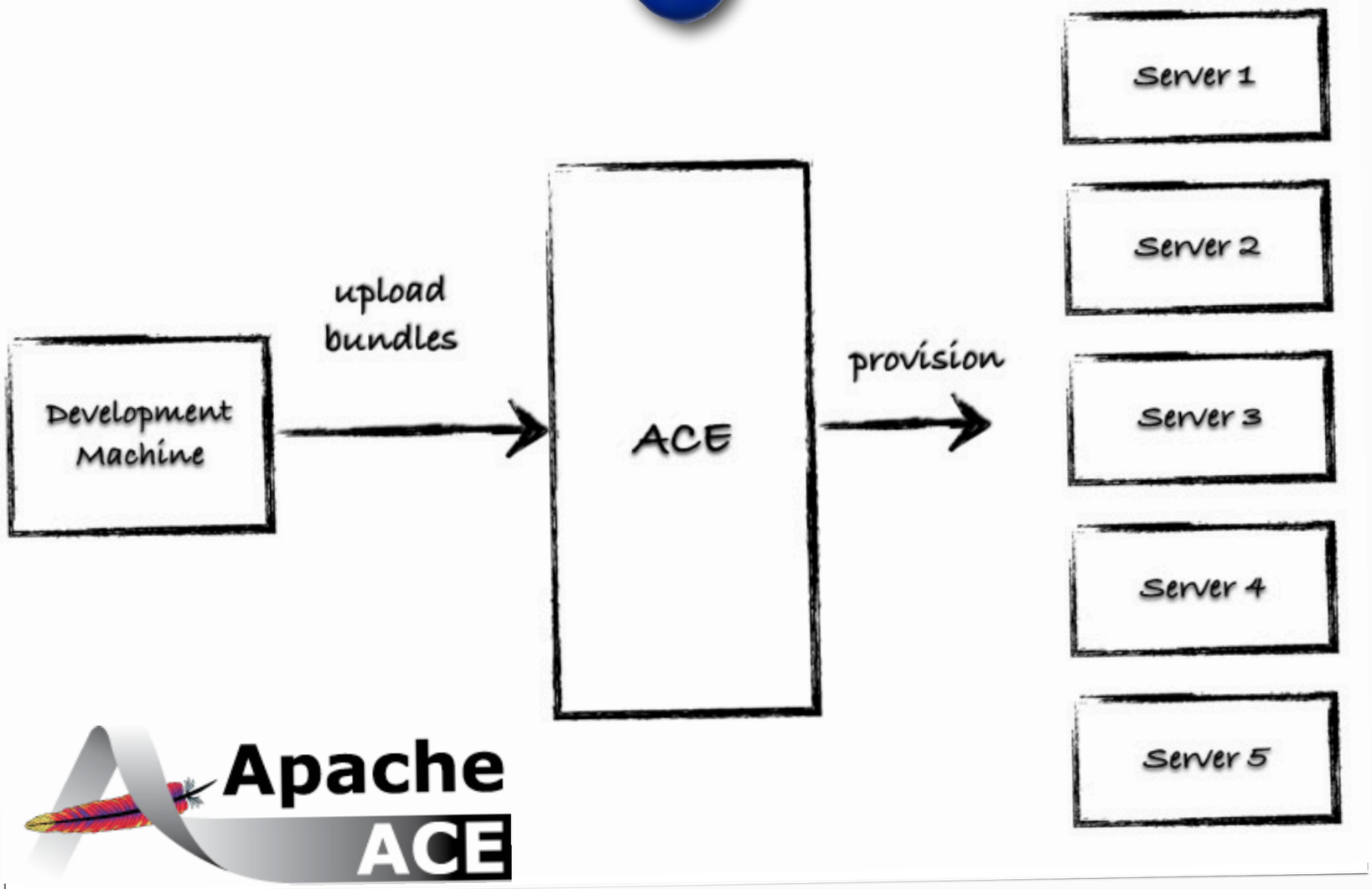
in a deployment

120

PulseOn

bundles

Provisioning servers



Retrieve | Store | Revert | Logout

Add artifact... | Dynamic Links | Add Feature... | Add Distribution... | Add target...

Artifacts

NAME
Jackson JSON processor-1.9.0
Data mapper for Jackson JSON processor-1.9.0
Amdatu Web - Dispatcher-1.0.0.SNAPSHOT
Amdatu Web - JAX RS-1.0.0.SNAPSHOT
Amdatu Web - Apache Wink Application-1.0.0.SNAPSH
Apache Felix Configuration Admin Service-1.2.8
Apache Felix Dependency Manager-3.1.0.SNAPSHOT
Apache Felix Dependency Manager Runtime-3.0.0
Apache Felix Dependency Manager Shell-3.0.1.SNAPS
Apache Felix Http Jetty-2.2.0
Apache Felix Log Service-1.0.0
Apache Felix Web Management Console-3.1.8
Apache Felix Metatype Service-1.0.4
agenda.api-1.0.0
agenda.rest-1.0.0
agenda.service.simple-1.0.0

Features

NAME	DESCRIPTION	AC
dependencies		
agenda		

Distributions

NAME	DESCRIPTION	ACTIONS
demo		- x

Targets

NAME				ACTI
bejugdemo	✓	✓	↻	-

Edit Target

Identifier
bejugdemo

Management | Info | LogViewer | Tag Editor | bejugdemo | Verify/resolve

Running

Start | Stop

Location
eu-west-1

Node type
t1.micro

Image owner ID

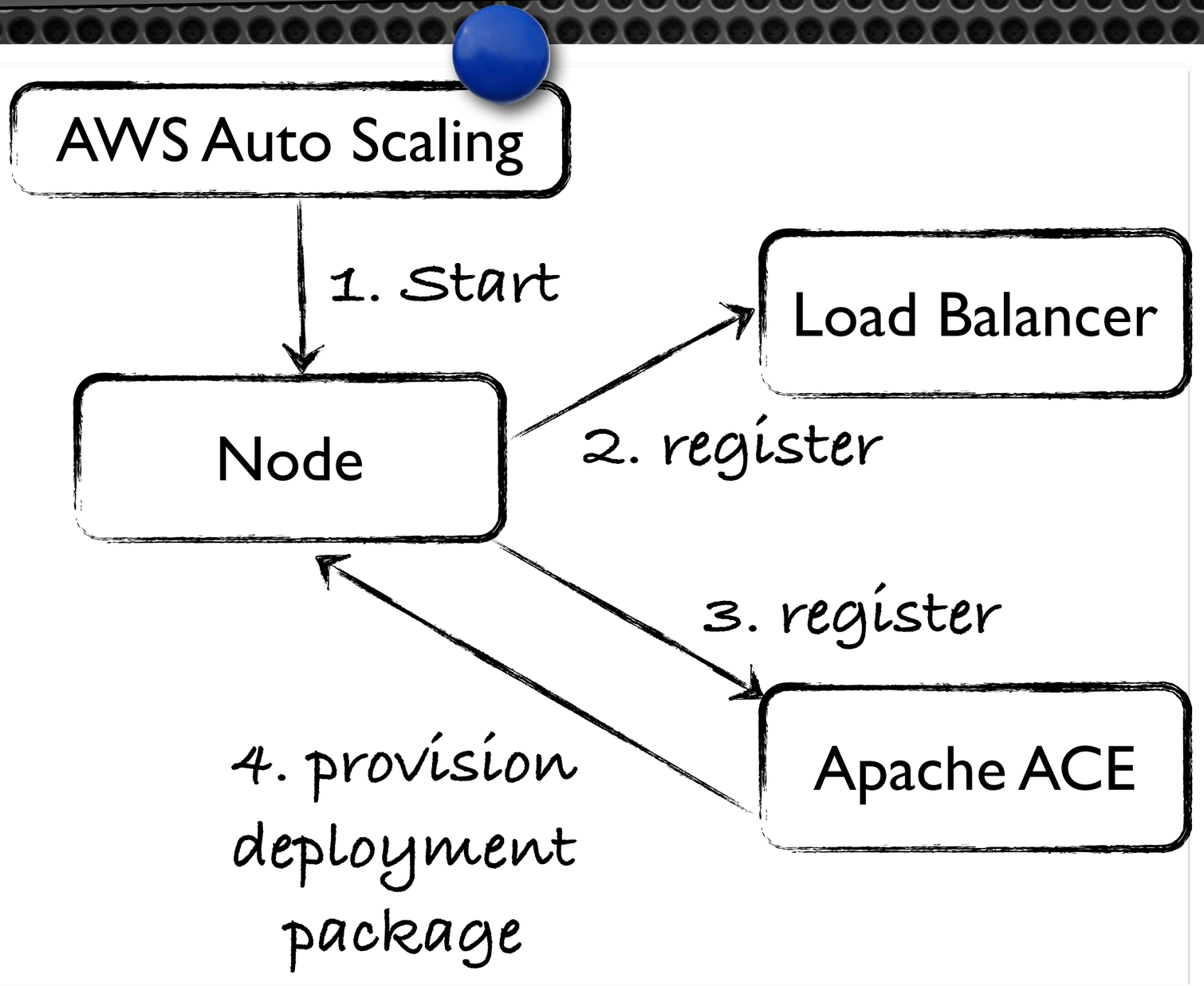
Image ID
amzn-ami-0.9.9-beta.i3

Keypair

Tag prefix
default

Ok | Cancel

Provisioning servers



```
./as-create-launch-config demo
      --image-id ami-0ee8e07a
      --instance-type m1.small
      --region eu-west-1
      --group sg-ce1420ba
      --user-data-file userdata.txt
```

```
./as-create-auto-scaling-group demo
     --launch-configuration demo
     --min-size 1
     --max-size 1
     --availability-zones eu-west-1a
     --load-balancers demo
     --tag "k=Name,v=demo,p=true"
```

Or by UI configuration...

Cluster	Name	Instances	Type	Min instances	Max instances	Desired instances	Scaling triggers	Actions
academy	academy	1	M1Small	1	3	1		+ - Delete group
contentrepo	contentrepo	1	M1Small	1	3	1		+ - Delete group
demo	demo-m1.small	1	M1Small	1	1	1		+ - Delete group

Provisioning servers

- Nodes are completely disposable
- Automated cluster recovery
- Nodes do not require maintenance
- No "big" container

Developer Tools

Development tool stack

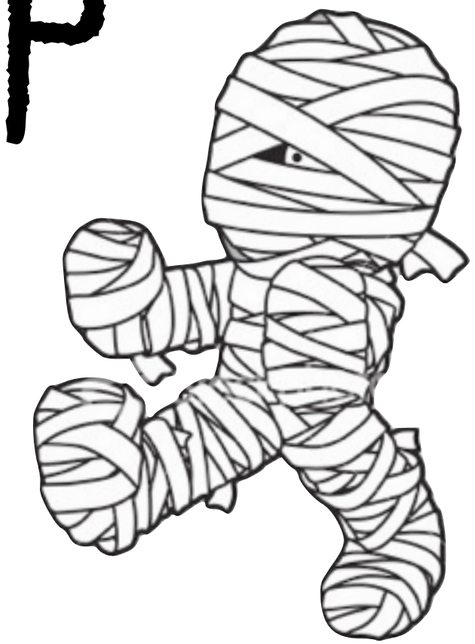
what do we need?

- [ ] IDE with fast turn arounds
- [ ] Version Control
- [ ] Continuous Integration Server
- [ ] Issue tracker
- [ ] WIKI

Continuous integration

- All BndTools projects support headless builds
- Build on git push
- Measure test coverage
  (both unit and integration tests)

Wrap up



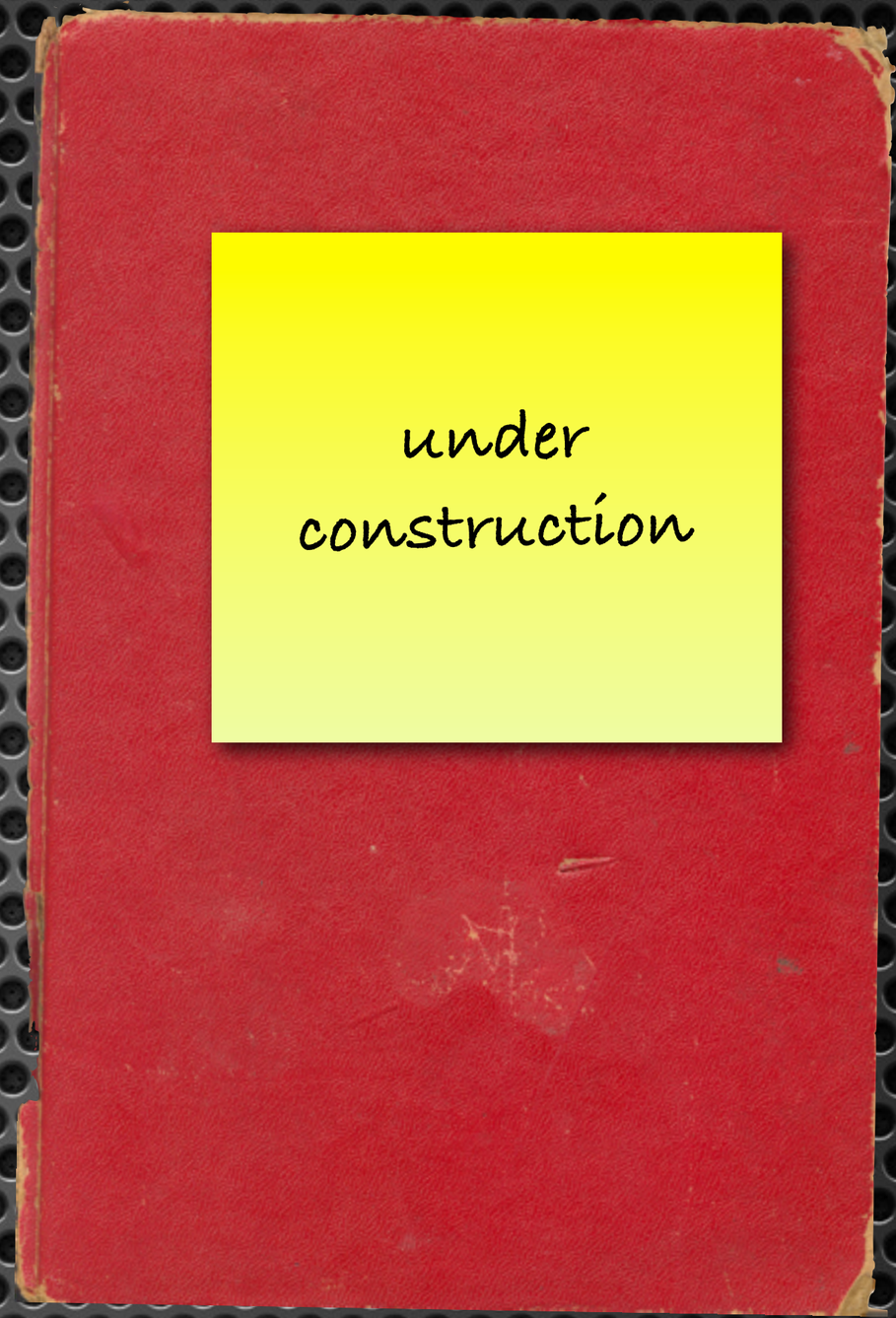
what have we learned?

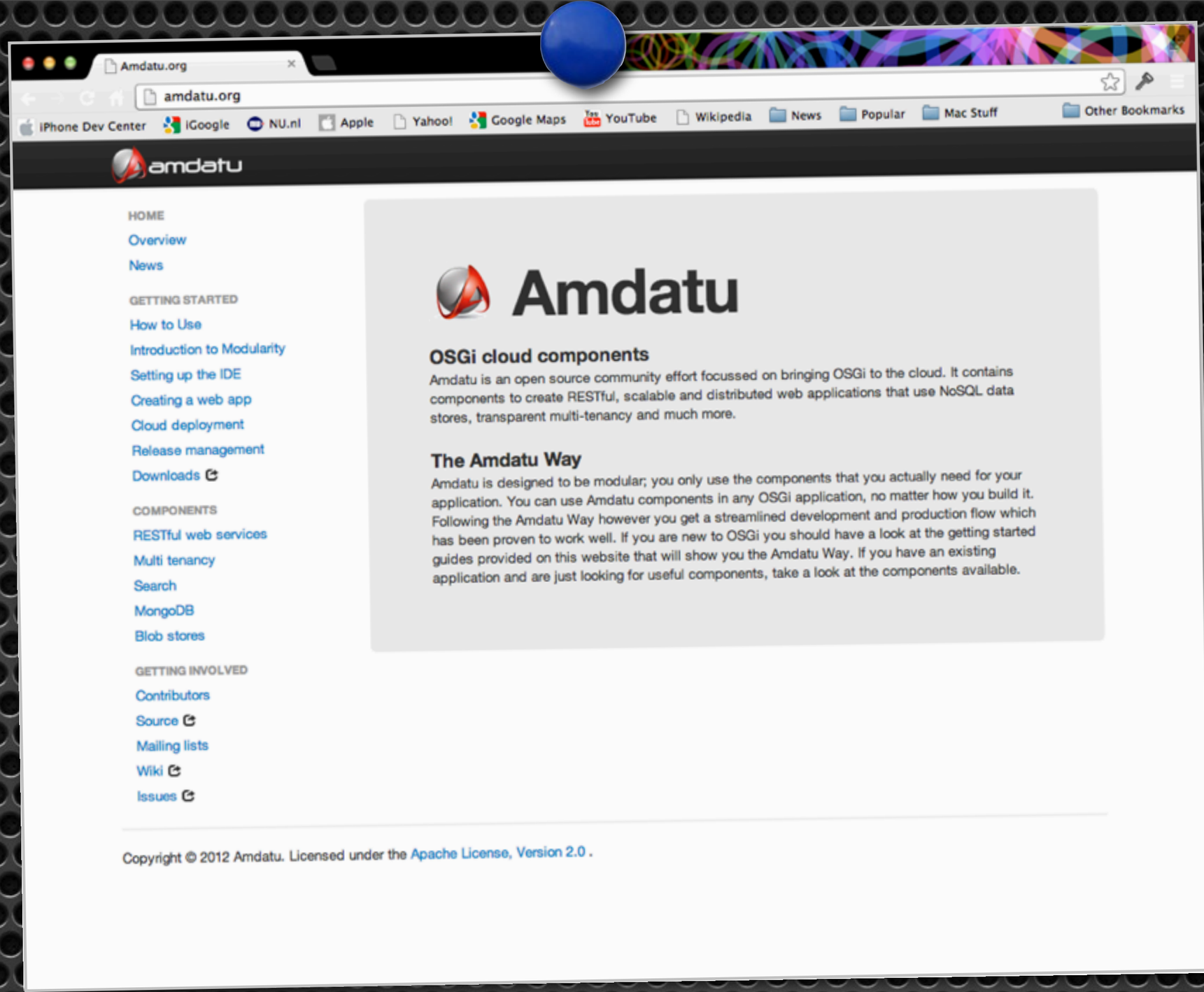
- Modularity is the ultimate agile tool
- Modularity is no longer difficult (BndTools / Amdatu)
- Stateless architecture is the only scalable way
- Cloud deployments made easy with a provisioning server

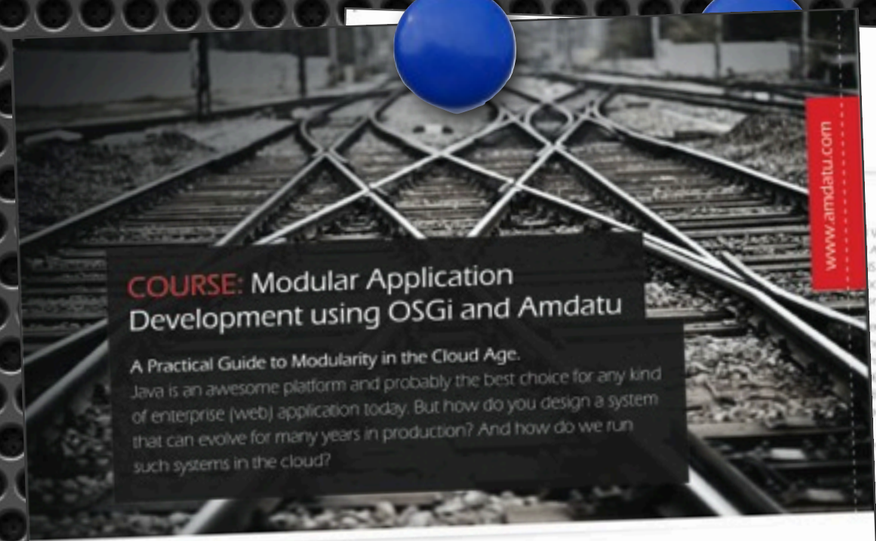
Shameless self-promotion...

Our book

- Building Modular Cloud Applications in Java
- Published by O'Reilly
- Due: end of summer this year




Amdatu.org
amdatu.org
iPhone Dev Center
iGoogle
NU.nl
Apple
Yahoo!
Google Maps
YouTube
Wikipedia
News
Popular
Mac Stuff
Other Bookmarks
amdatu
HOME
Overview
News
GETTING STARTED
How to Use
Introduction to Modularity
Setting up the IDE
Creating a web app
Cloud deployment
Release management
Downloads
COMPONENTS
RESTful web services
Multi tenancy
Search
MongoDB
Blob stores
GETTING INVOLVED
Contributors
Source
Mailing lists
Wiki
Issues
Amdatu
OSGi cloud components
Amdatu is an open source community effort focussed on bringing OSGi to the cloud. It contains components to create RESTful, scalable and distributed web applications that use NoSQL data stores, transparent multi-tenancy and much more.
The Amdatu Way
Amdatu is designed to be modular; you only use the components that you actually need for your application. You can use Amdatu components in any OSGi application, no matter how you build it. Following the Amdatu Way however you get a streamlined development and production flow which has been proven to work well. If you are new to OSGi you should have a look at the getting started guides provided on this website that will show you the Amdatu Way. If you have an existing application and are just looking for useful components, take a look at the components available.
Copyright © 2012 Amdatu. Licensed under the Apache License, Version 2.0 .



COURSE: Modular Application Development using OSGi and Amdatu

A Practical Guide to Modularity in the Cloud Age.

Java is an awesome platform and probably the best choice for any kind of enterprise (web) application today. But how do you design a system that can evolve for many years in production? And how do we run such systems in the cloud?

www.amdatu.com

Designing a system that can evolve without creating a maintenance nightmare is far from trivial and there is no silver bullet to do this correctly. A modular architecture will help a lot to achieve this goal. Modularity forces separation of concerns which enables you to replace parts of a system without breaking others. The only mature modularity approach for Java is OSGi. OSGi is a framework that enables low-level modularity and services, but it is not an application framework like Java EE, i.e. you still need APIs to create web applications, use transactions, access data sources etc. Without these APIs you will have a hard time to actually build applications. Unfortunately OSGi and Java enterprise technologies did not interoperate well in the past; it was very challenging to mix both. But what if we want modularity in our architecture, but also the ease-of-use of familiar high level Java (enterprise) APIs? The next question is how to deploy such applications in the cloud. Of course you could use one of the existing PaaS providers, but these basically deploy your old, monolithic WAR/EAR in the cloud. This doesn't fit modularity very well. How can you update just parts of a running application without breaking the rest of it?

This lab-intensive, five day, course teaches you how to benefit from a modular programming model meanwhile dealing with these complex issues. It will present a practical and proven solution for building modular Java enterprise applications that can be deployed into a (public/private) cloud. The course will introduce OSGi, the Amdatu Open Source Cloud OSGi Components, and Apache ACE, an Open Source Provisioning Platform. We will not only focus on technical features, but also on architectural aspects like non-functional requirements, dependency management, design considerations, and testability.

amdatu

Course Contents

Web Services
API Design Considerations
S Clients
cal Clients
n Jobs

nance Strategies
e Cloud
ntroduction
B
Robstores
ots

Stores (Memcached)
Jobs

Provisioning
icy
undles

Auto-scaling
ets

Management and Semantic Versioning
h
tom Bundle Repositories
oading and Dynamic Imports
les From Plain JARs
mming Strategies

ncepts to create flexible business

Operating System and IDE of their liking

cle as a Java Champion and is an active
nd modularity

aker on conferences all over the world
, Bndtools, and JBoss Forge

prise applications with Java

Want to try this yourself?

- Tomorrow: 10 AM - 12.30
- Hands-on Lab:
  - Building Modular Cloud Applications in Java
- Venue: Hotel Wyspiański
  ul.Westerplatte 15
  31-033 Kraków

Cloud provisioning

http://ace.apache.org/

Eclipse OSGi plugin

http://bndtools.org/

Cloud OSGi services

http://www.amdatu.org/

That's us

http://luminis.eu/

Bert Ertman

bert.ertman@luminis.eu

@BertErtman

Paul Bakker

paul.bakker@luminis.eu

@pbakker

Dank U

Merci

Danke

Mahalo

Obrigado

Grazie

Gracias

Thank you

Takk