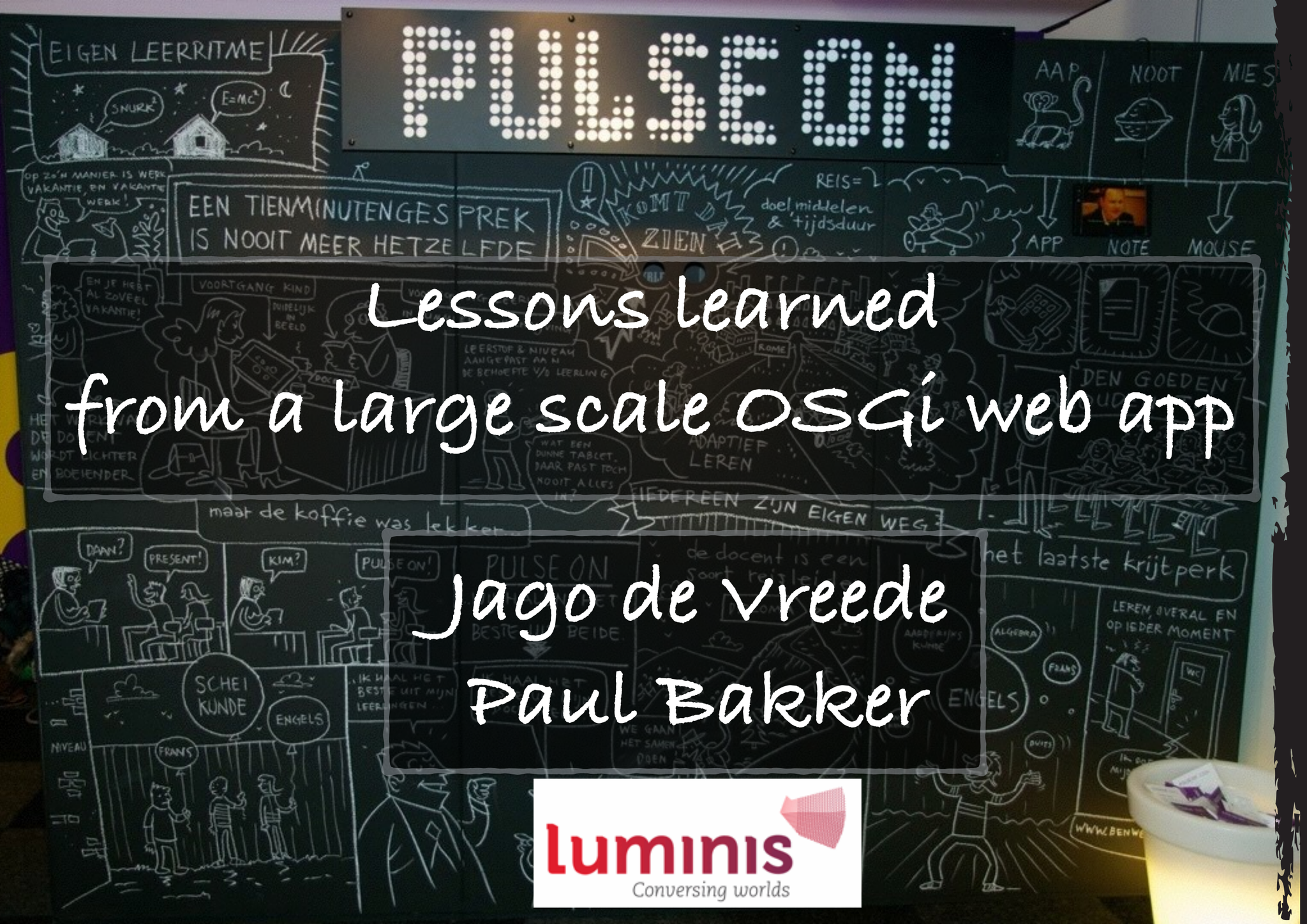
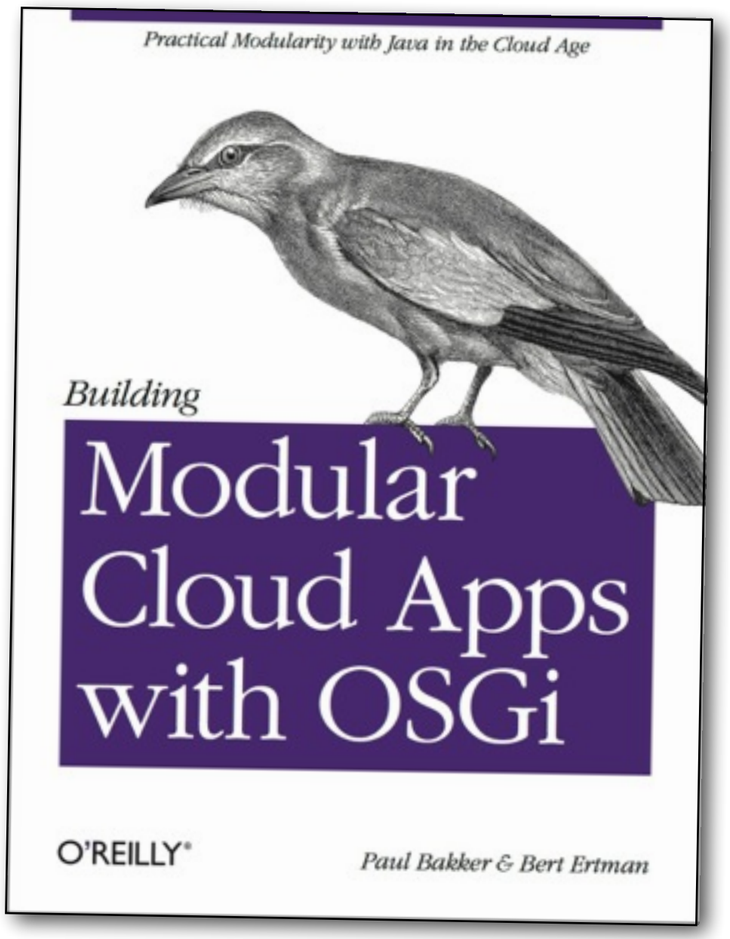


Lessons learned from a large scale OSGi web app

Jago de Vreede
Paul Bakker

luminis
Conversing worlds

amdatu



Paul Bakker

@pbakker

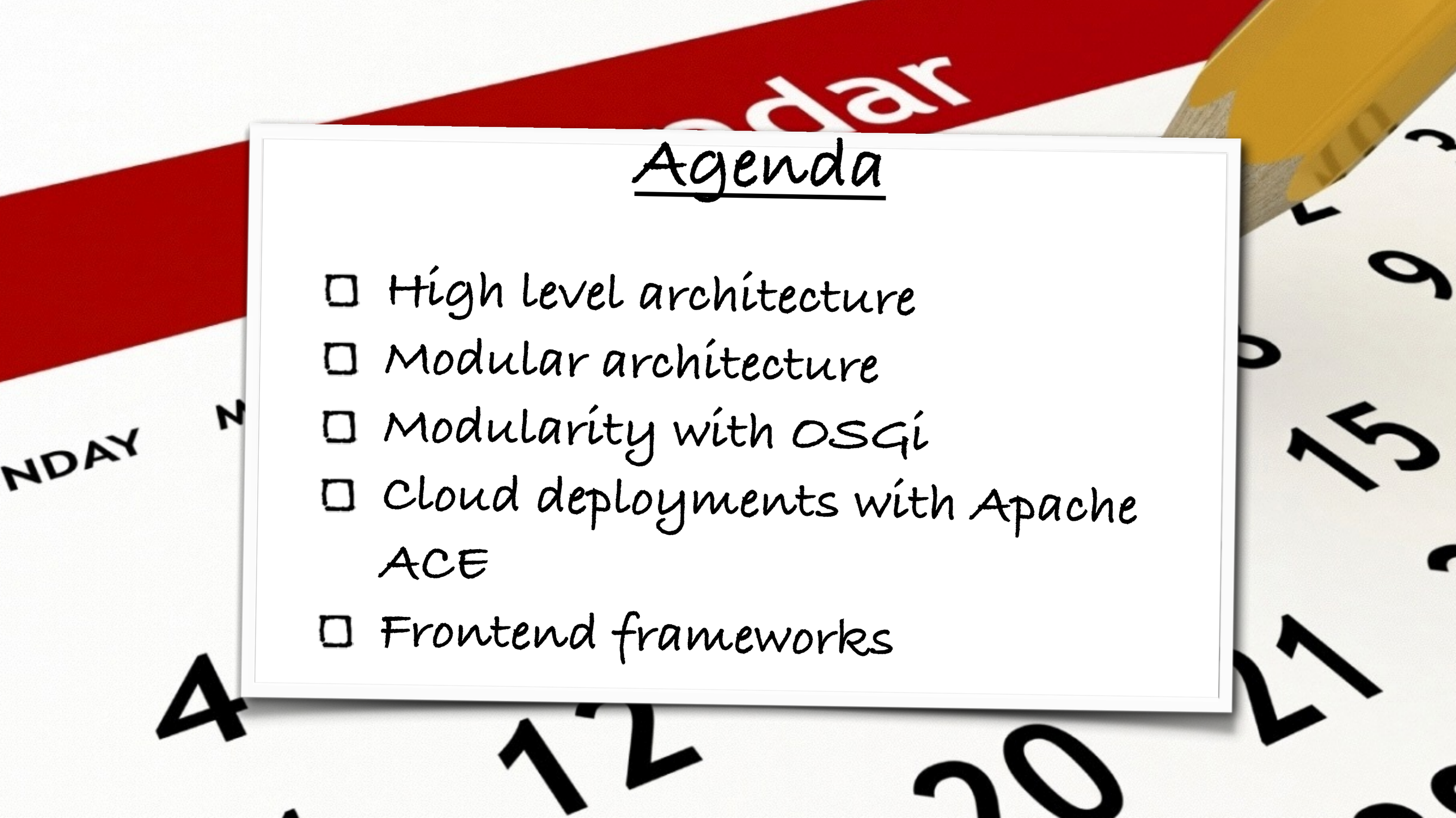
luminis
Conversing worlds

amdatu

Jago de Vreede

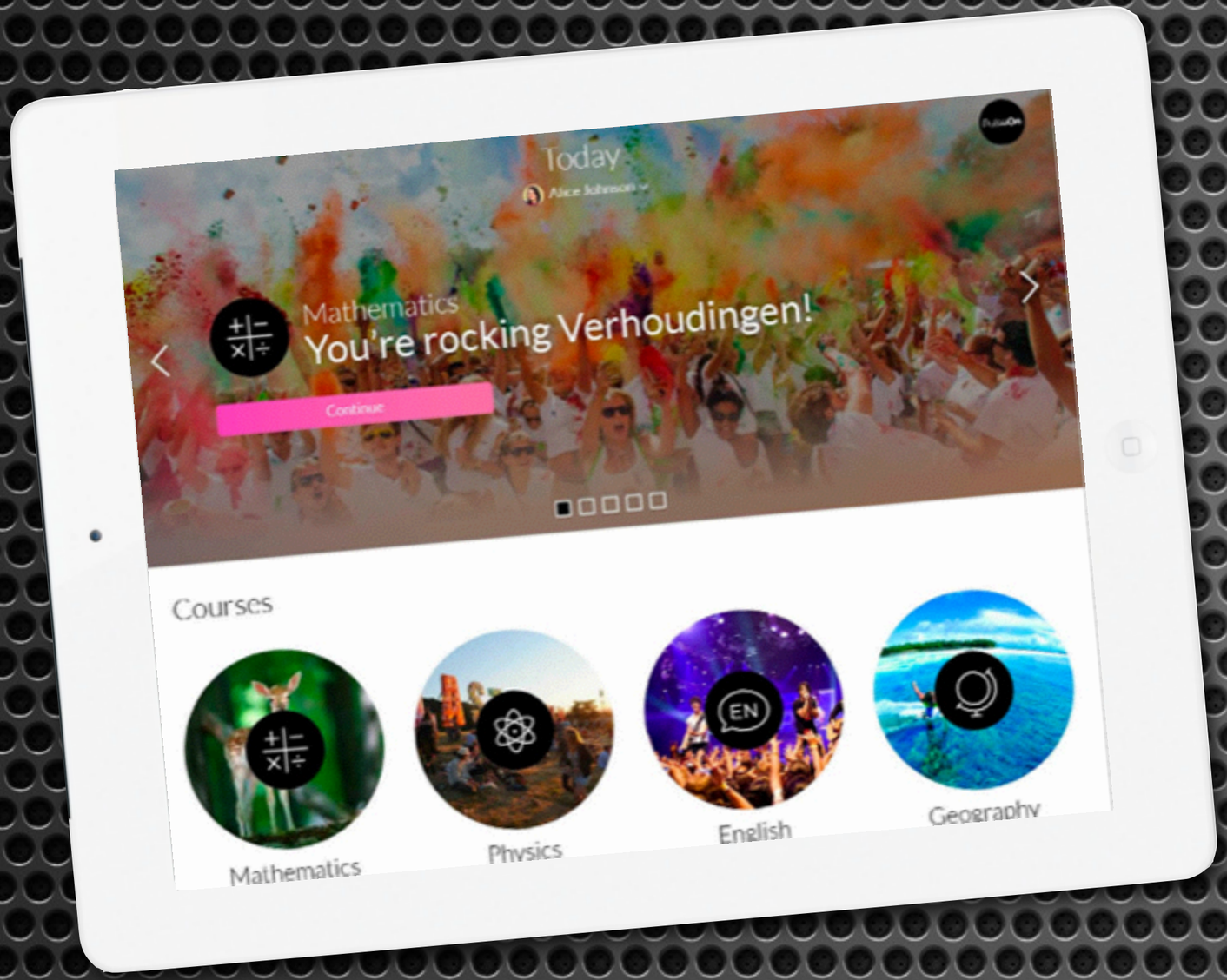
luminis
Conversing worlds

amdatu


Agenda
High level architecture
Modular architecture
Modularity with OSGi
Cloud deployments with Apache ACE
Frontend frameworks


ANY
QUESTIONS
?
Ask whenever
you want!

The case



PulseOn

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

Tools

bndtools
Eclipse OSGi plugin
http://bndtools.org/

Eclipse
http://eclipse.org/

Source control

Cloud provisioning
http://ace.apache.org/

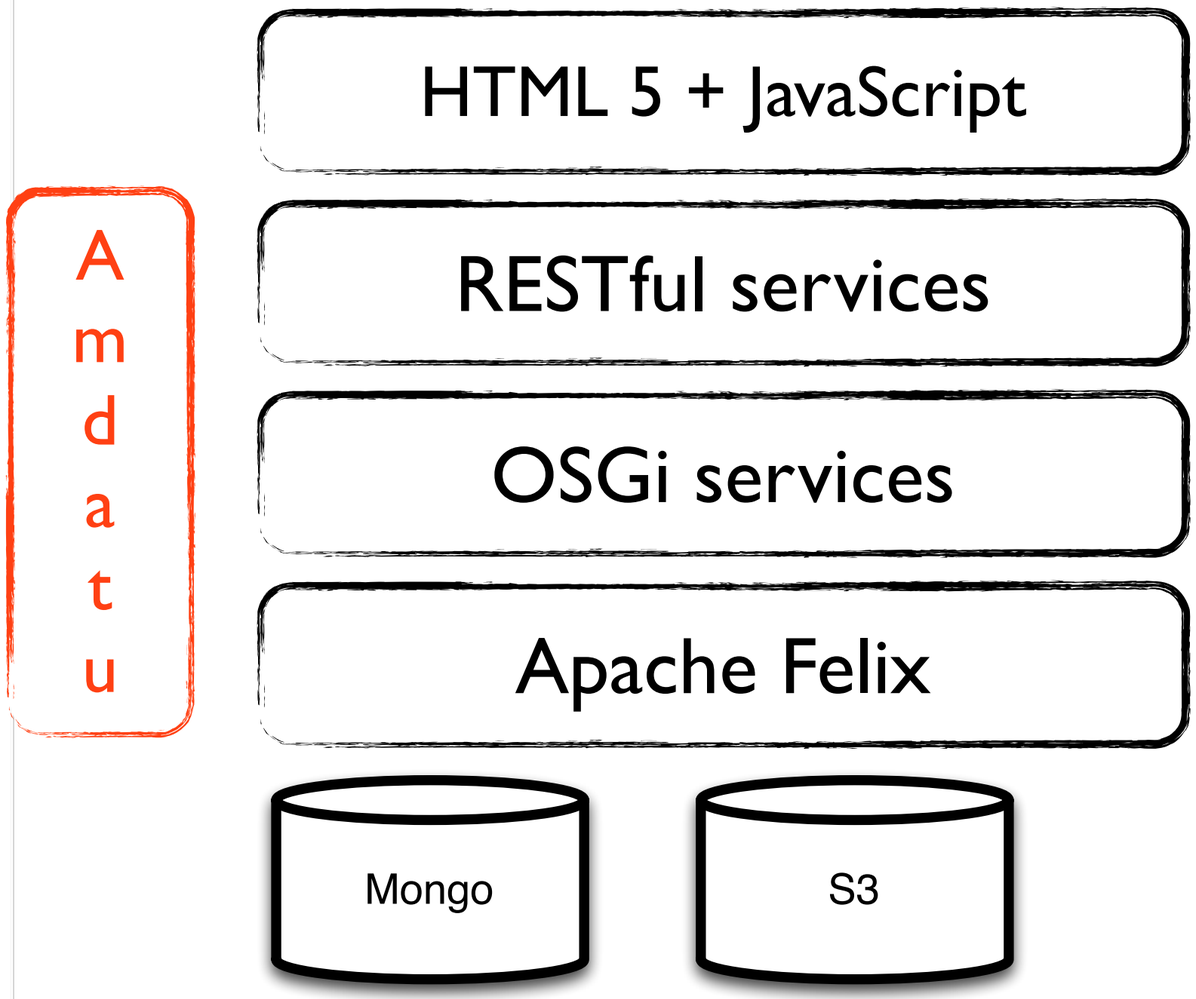
WebStorm
http://jetbrains.com/webstorm/

Stash, Jira, Bamboo
http://atlassian.com/

Requirements:

- Agile and modular
- Modern web app
- UI mostly offloaded to clients or devices
- Integration via REST API
- Horizontally scalable

High level architecture

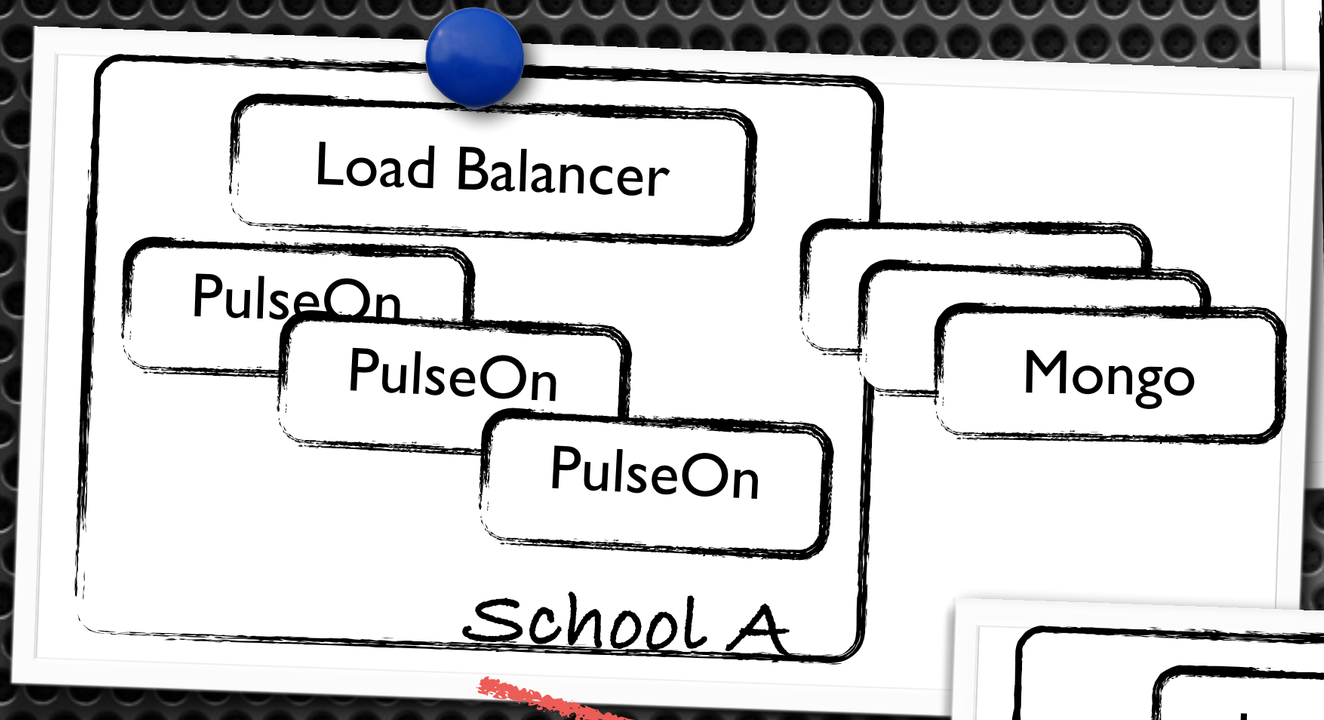


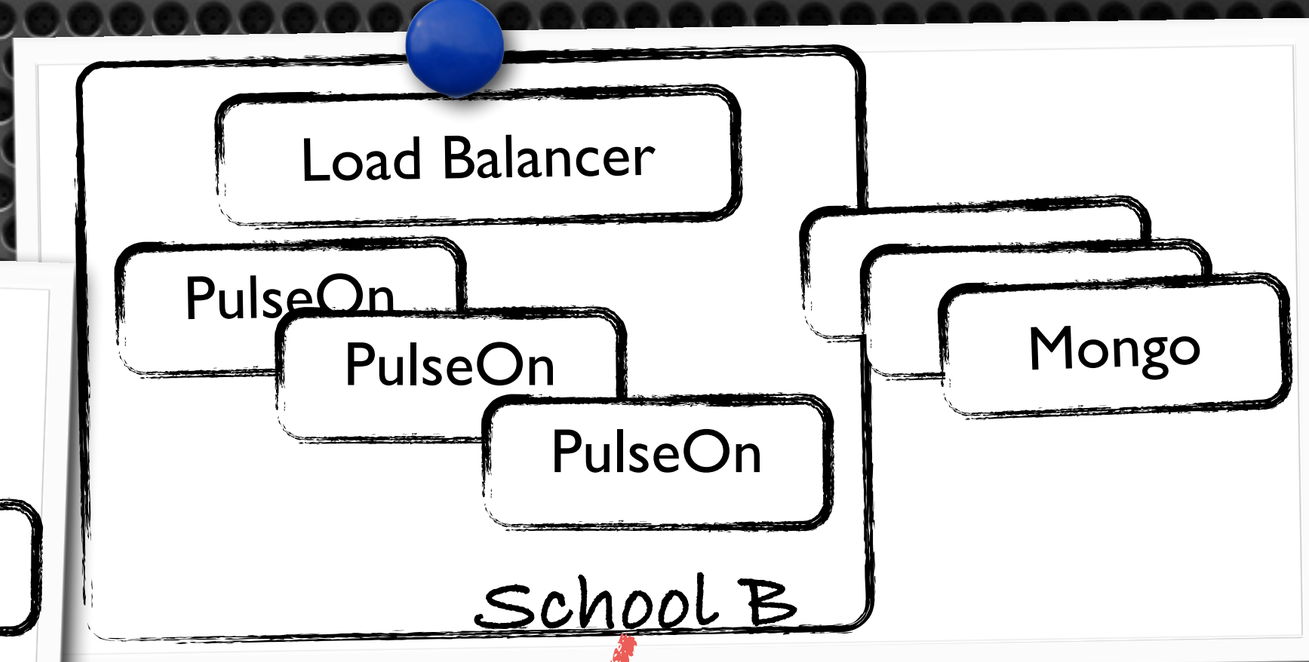
Components

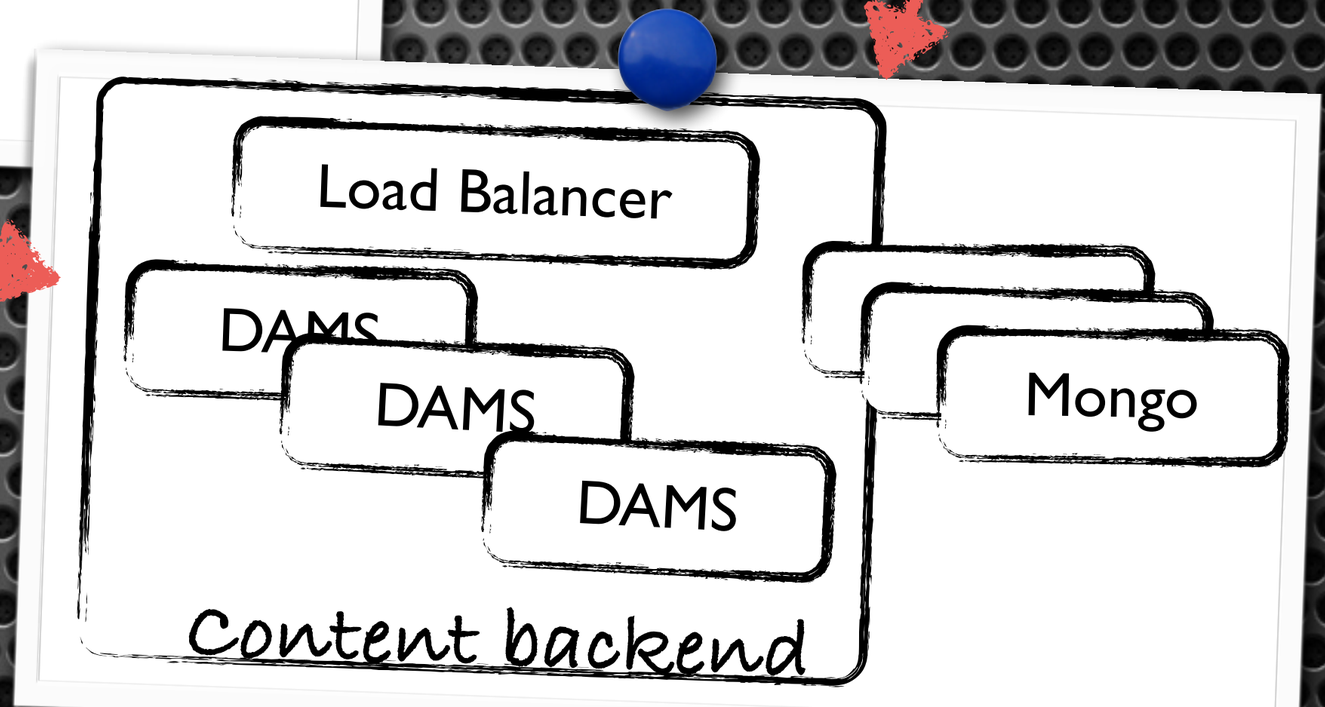
- Auth
- Blob stores
- MongoDB
- Multi-tenancy
- OpenSocial
- Search
- Remote Services
- REST
- Template
- Web
- ...

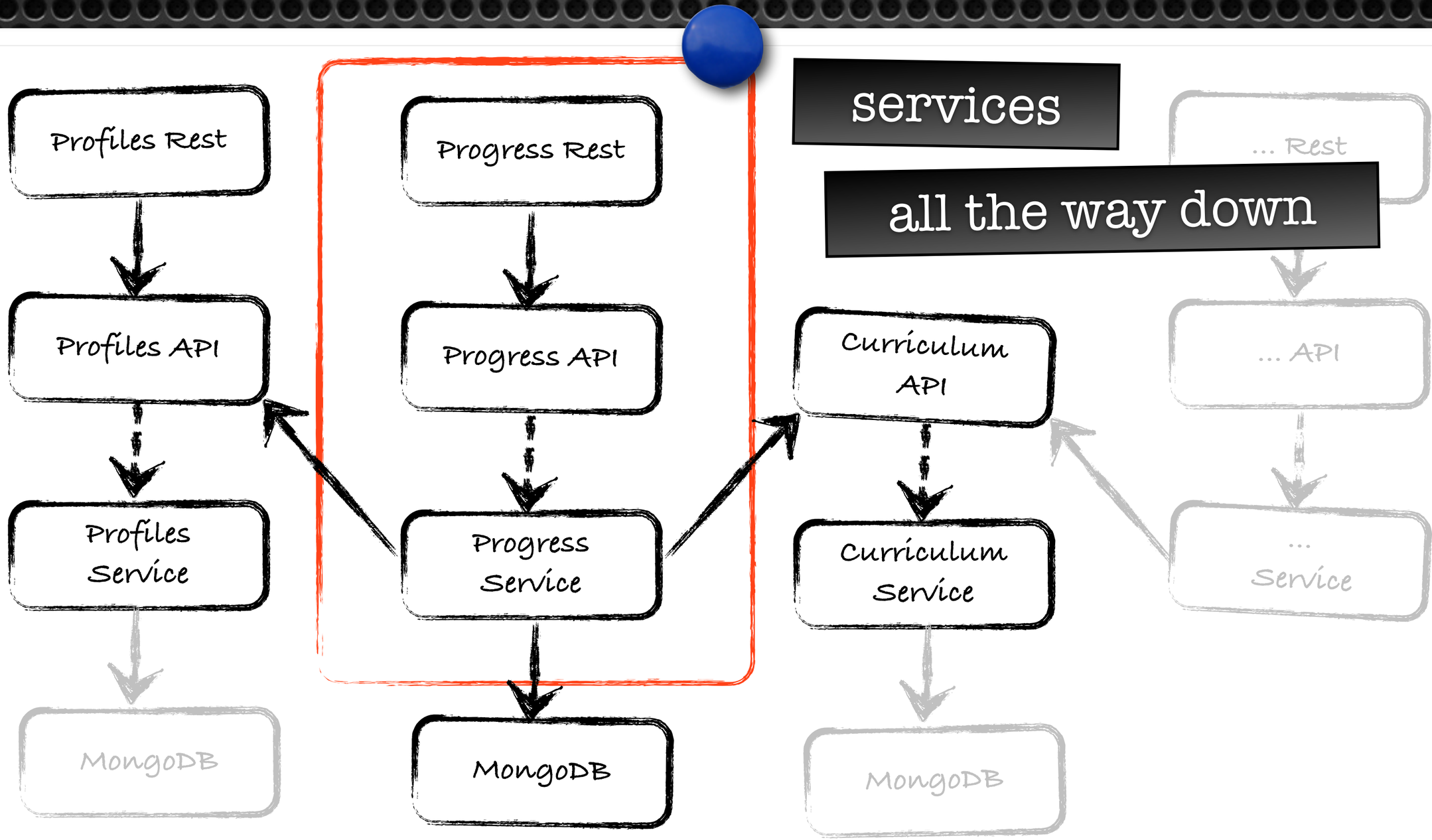
amdatu

Deployment


Load Balancer
PulseOn
PulseOn
PulseOn
Mongo
School A

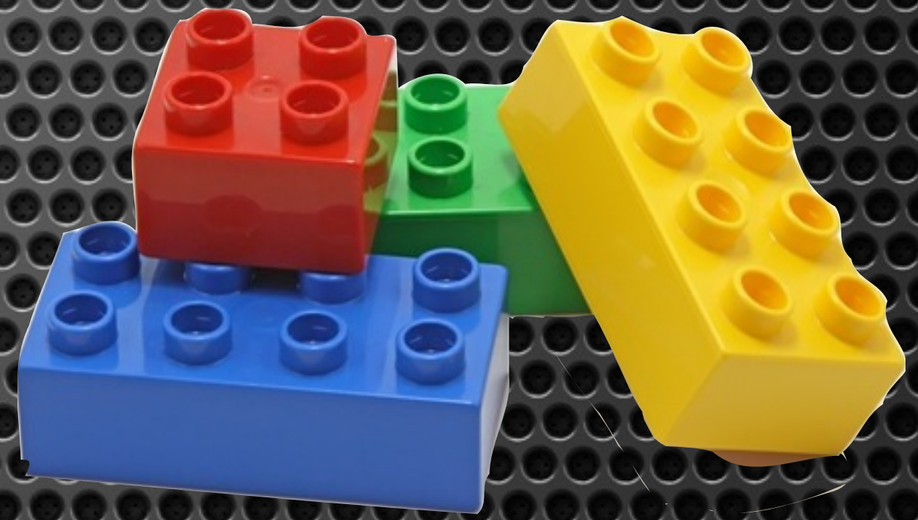

Load Balancer
PulseOn
PulseOn
PulseOn
Mongo
School B


Load Balancer
DAMS
DAMS
DAMS
Mongo
Content backend


services
all the way down
Profiles Rest
Profiles API
Profiles Service
MongoDB
Progress Rest
Progress API
Progress Service
MongoDB
Curriculum API
Curriculum Service
MongoDB
... Rest
... API
... Service

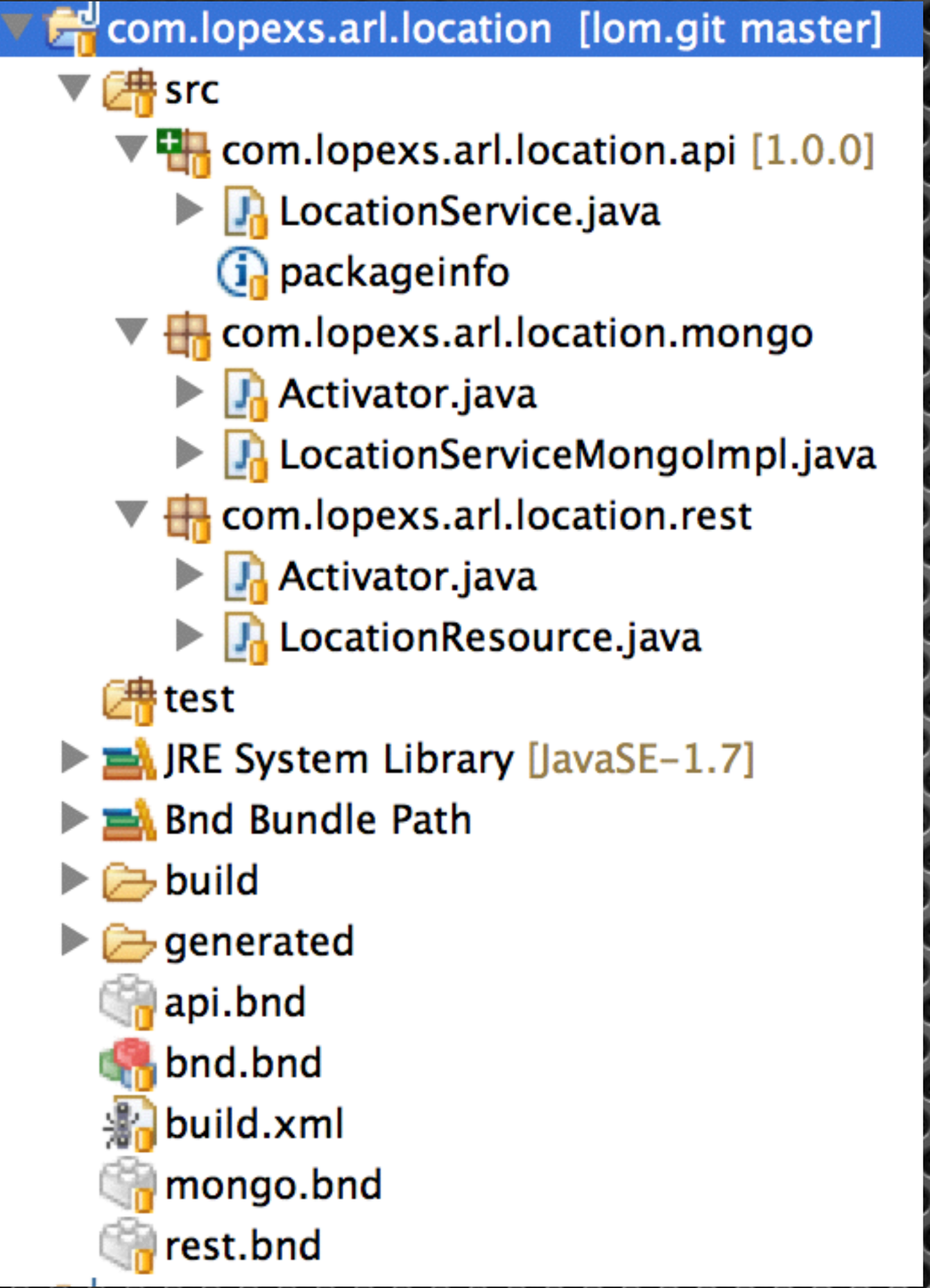
A service should only do **one** single thing

Services are the (reusable) building blocks of our application



Benefits of a services based architecture

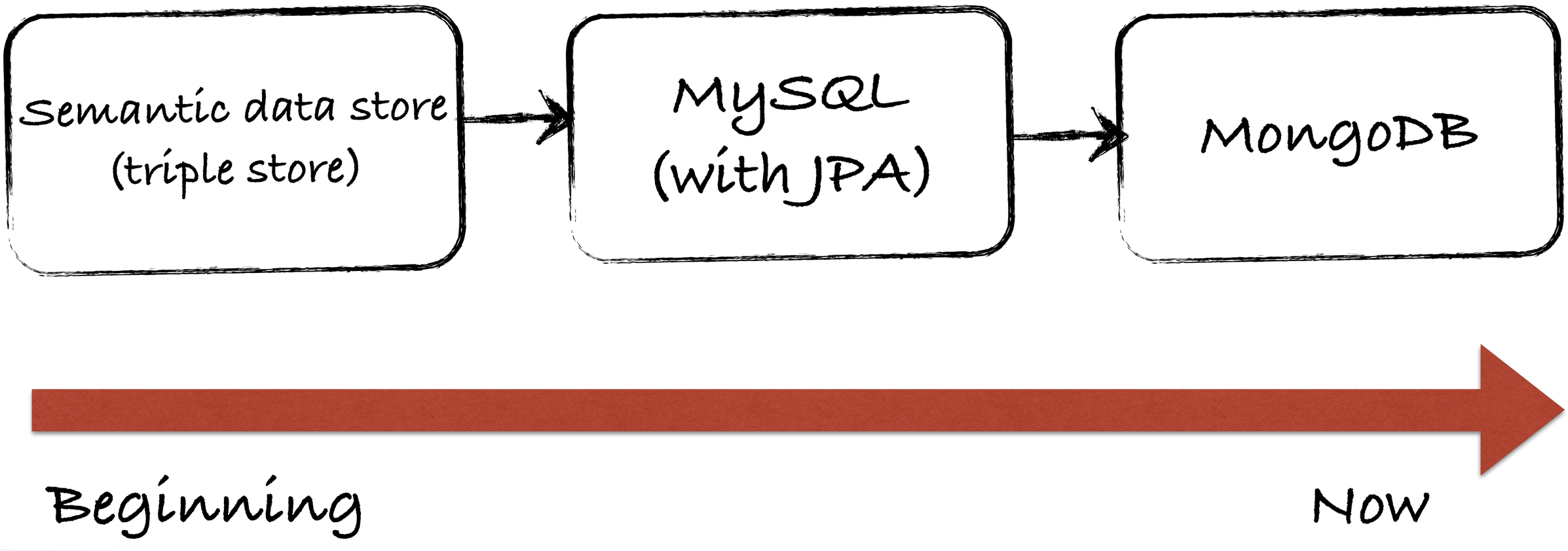
- Small services are easy to maintain
- Small services are composable
- Services are easily testable



Packaging services

- Services are packaged in small bundles
- Related bundles may be generated from a single Bndtools project

Data Storage


semantic data store
(triple store)
MySQL
(with JPA)
MongoDB
Beginning
Now

Relational databases

in a modular system

A part of data should be owned by 1 service

How to set boundaries in a relational world!?

Relational databases

in the modern web

- ORM is REALLY difficult to get right
- Relational databases don't scale well

MongoDB: our database of choice

- Object Mapping trivial
- Easy to scale
- Powerful aggregation/map-reduce framework

Code example: MongoDB with Amdatu

Amdatu Mongo Service

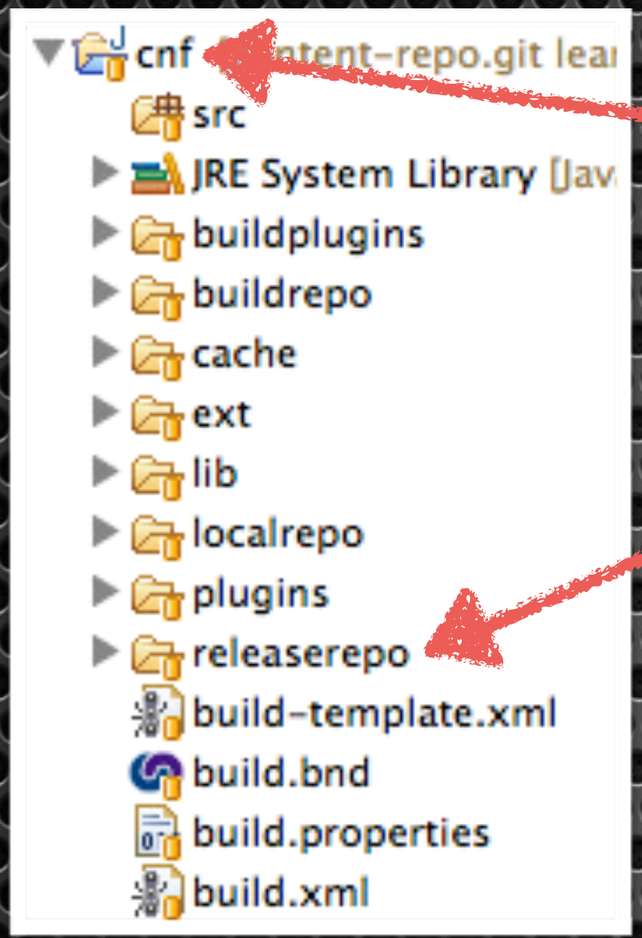
Setup Object Mapper

Execute query

```
@Component
public class MongoProducts implements ProductService{

    @ServiceDependency
    private volatile MongoDBService mongoDbService;
    private volatile DBCollection collection;
    private volatile JacksonDBCollection<Product, String> products;

    @Start
    public void start() {
        collection = mongoDbService.getDB().getCollection("products");
        products = JacksonDBCollection.wrap(collection, Product.class, String.class);
    }

    @Override
    public List<Product> listProducts(String category) {

        DBCursor<Product> dbCursor = products.find().is("category", category);
        List<Product> result = new ArrayList<>();

        dbCursor.forEach(result::add);

        return result;
    }
}
```

Release & Deployment

Binaries



Rele se t release repo

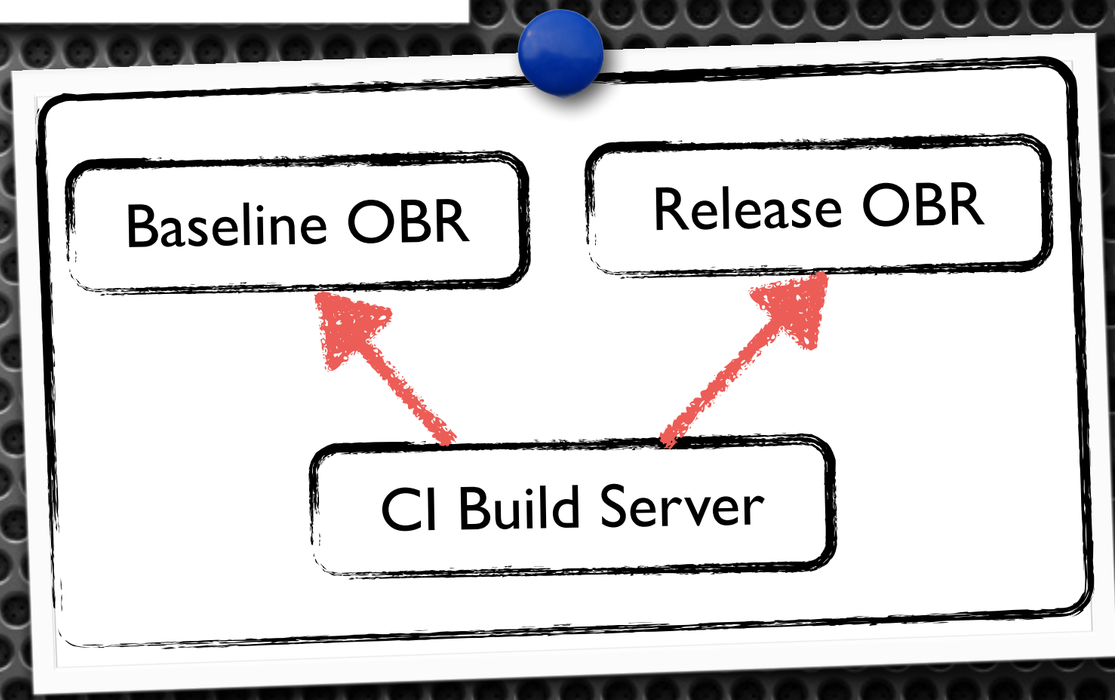
bad

Release to

release OBR

JavaScript minification

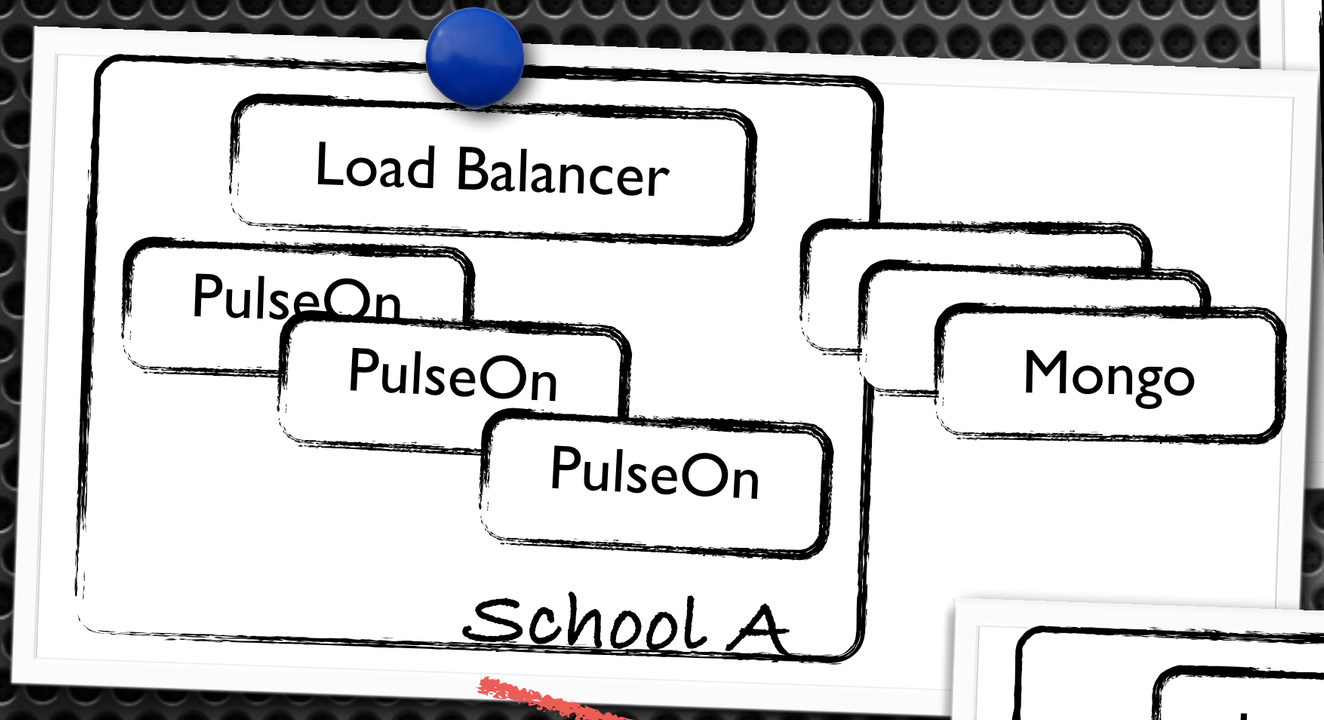
- closure compiler
  - deterministic? => fixed March 3, 2014

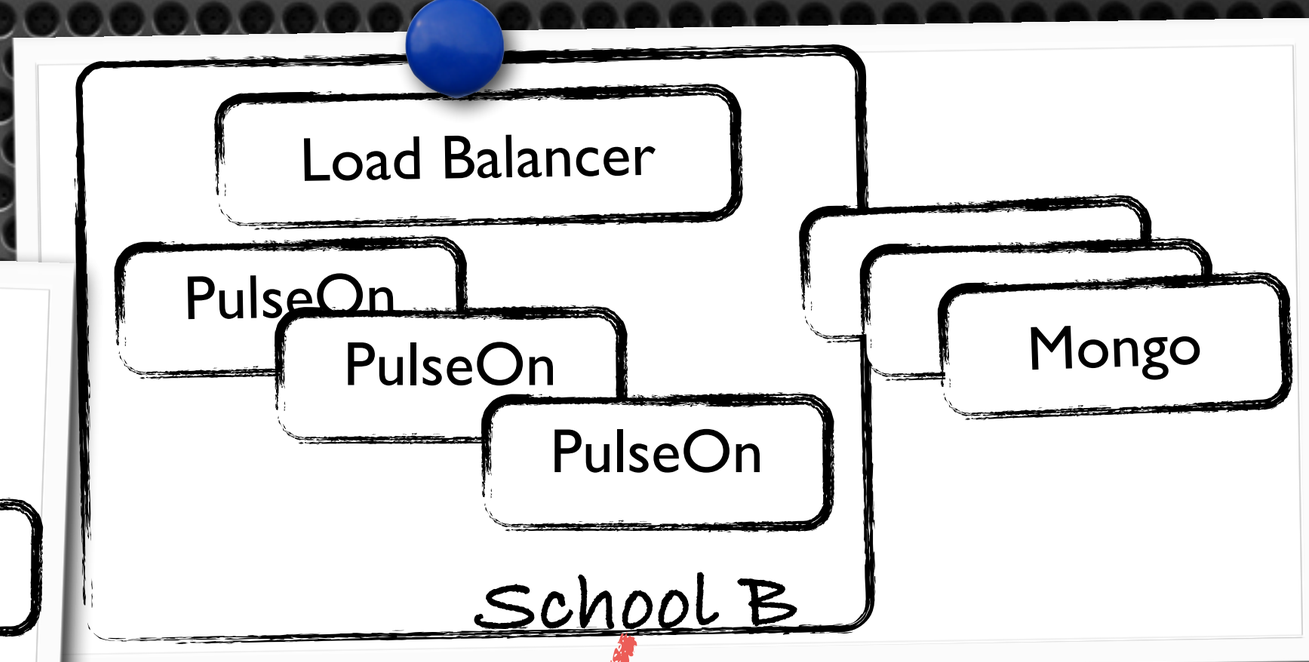


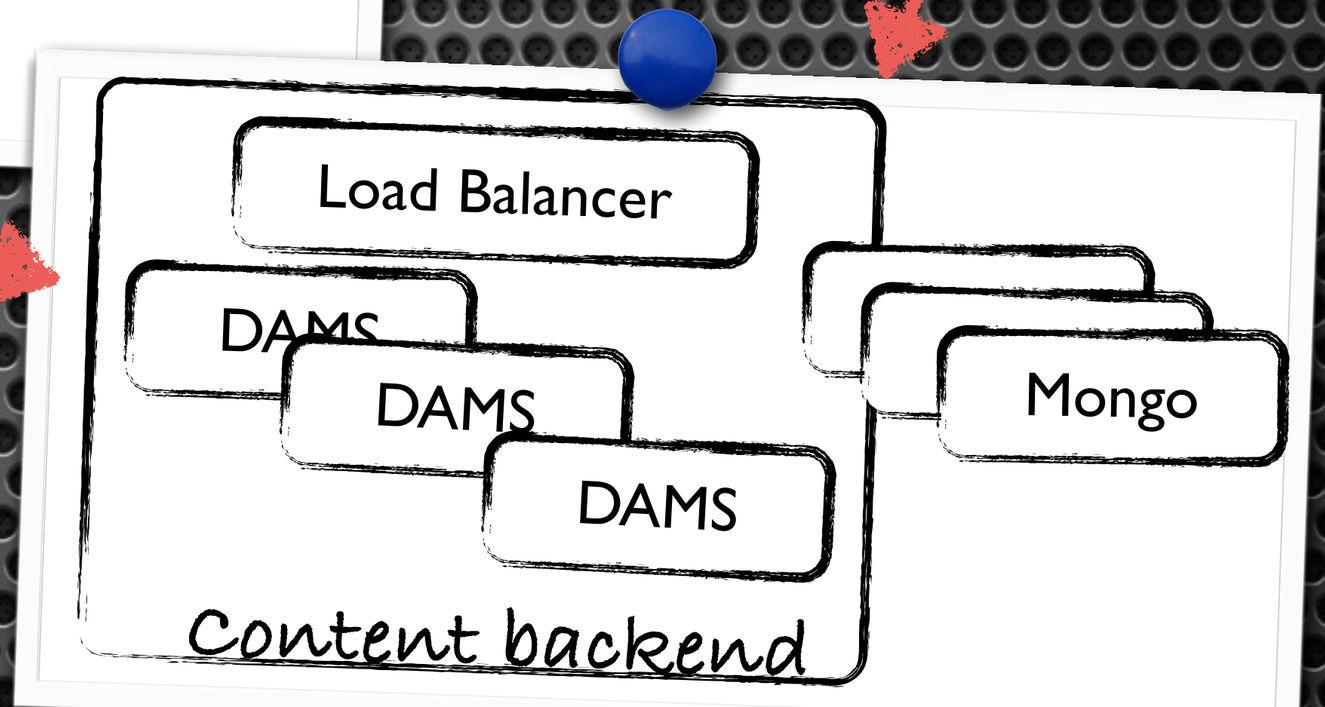
Deployment

- Software distribution framework
- Manages the installation and upgrade of bundles, configuration, etc. to heterogenous targets

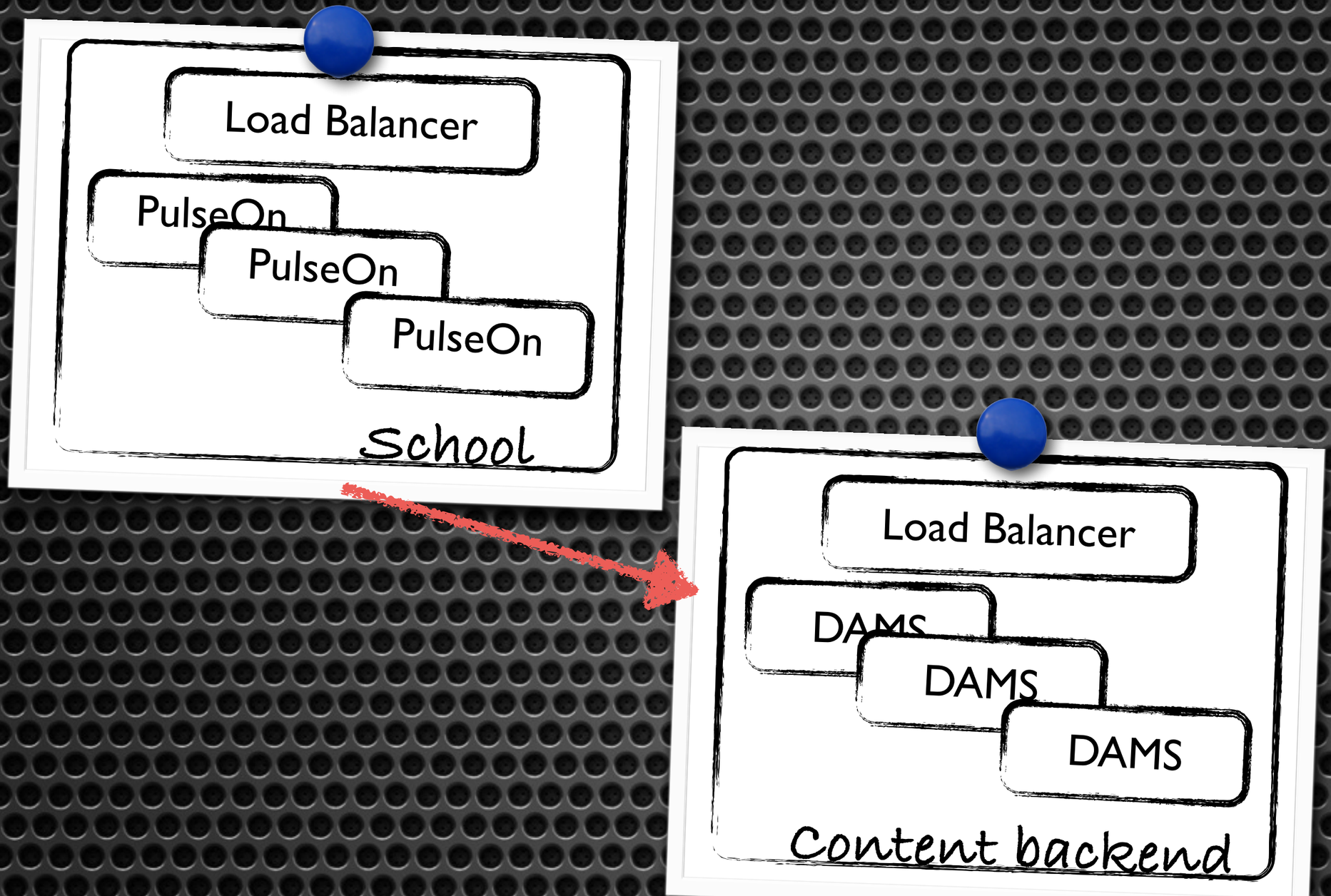
Deployment


Load Balancer
PulseOn
PulseOn
PulseOn
Mongo
School A

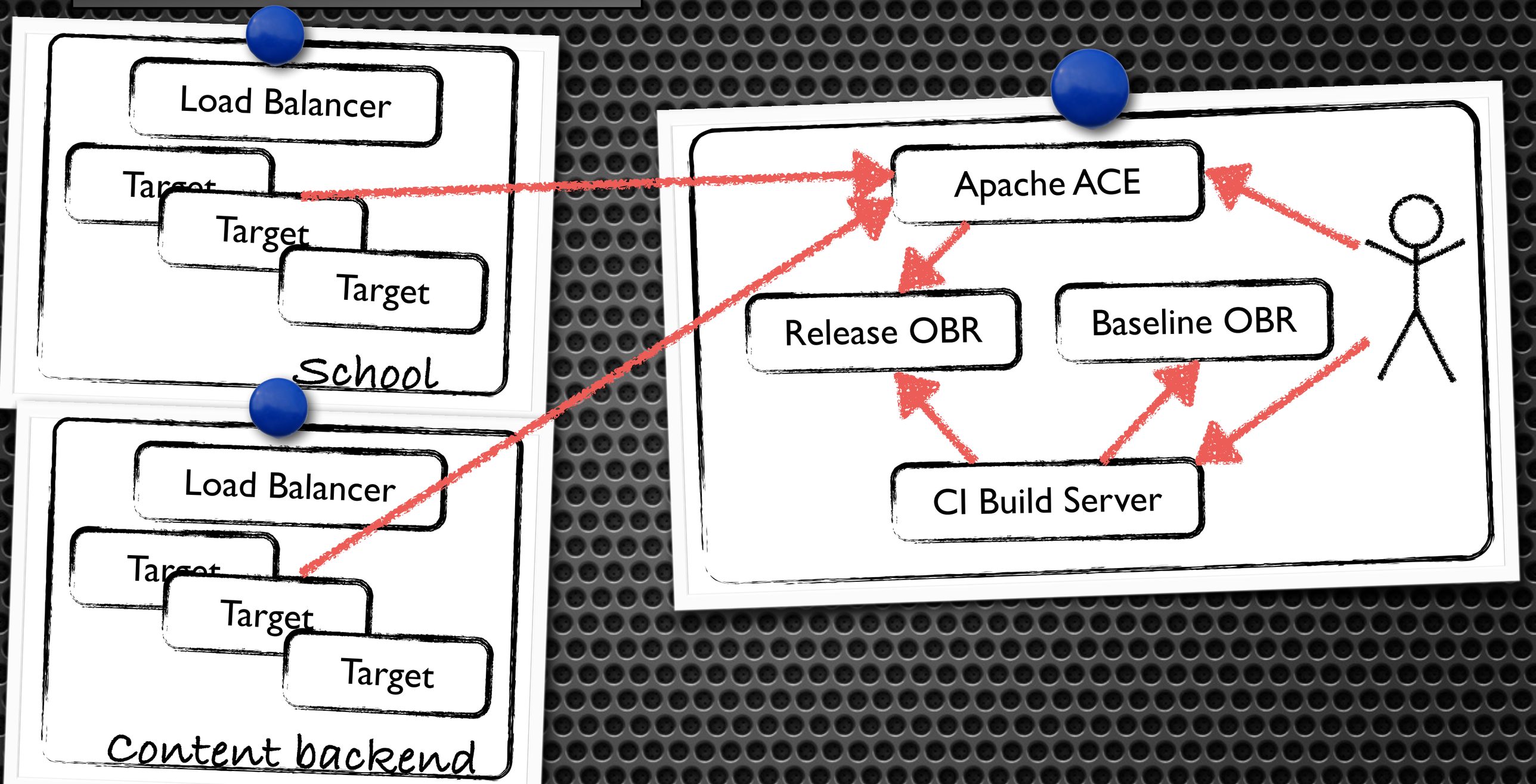

Load Balancer
PulseOn
PulseOn
PulseOn
Mongo
School B


Load Balancer
DAMS
DAMS
DAMS
Mongo
Content backend

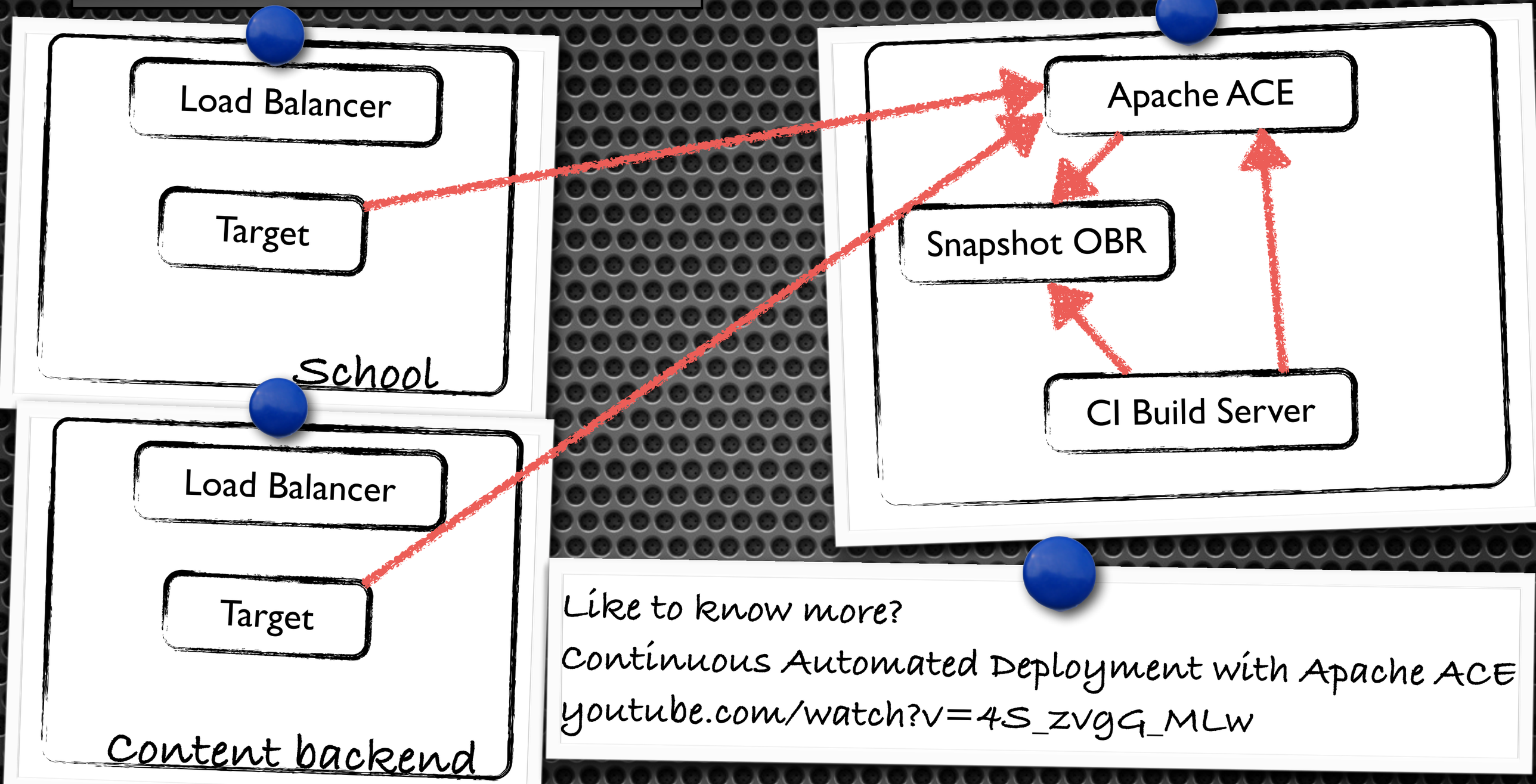
Deployment


Load Balancer
PulseOn
PulseOn
PulseOn
School
Load Balancer
DAMS
DAMS
DAMS
Content backend

Deployment


Load Balancer
Target
Target
Target
School
Load Balancer
Target
Target
Target
Content backend
Apache ACE
Release OBR
Baseline OBR
CI Build Server

Continuous Deployment



Like to know more?

Continuous Automated Deployment with Apache ACE

youtube.com/watch?v=4S_zvgG_MLw


TESTING
PLEASE
DO NOT
DISTURB

Unit testing OSGi code

is trivial

But what if we want

more?

Integration testing turns out to be trivial as well!

```
public class ProductServiceTest extends BaseOSGiServiceTest<ProductService>{

    public ProductServiceTest() {
        super(ProductService.class);
    }

    @Override
    public void setUp() throws Exception {
        Properties props = new Properties();
        props.put("dbName", "osgiwebshop");
        configureFactory( "org.amdatu.mongo", props);
        super.setUp();
    }

    public void test() {
        List<Product> listProducts = instance.listProducts("Books");
        assertEquals(3, listProducts.size());
    }
}
```

Automated dependency injection of service

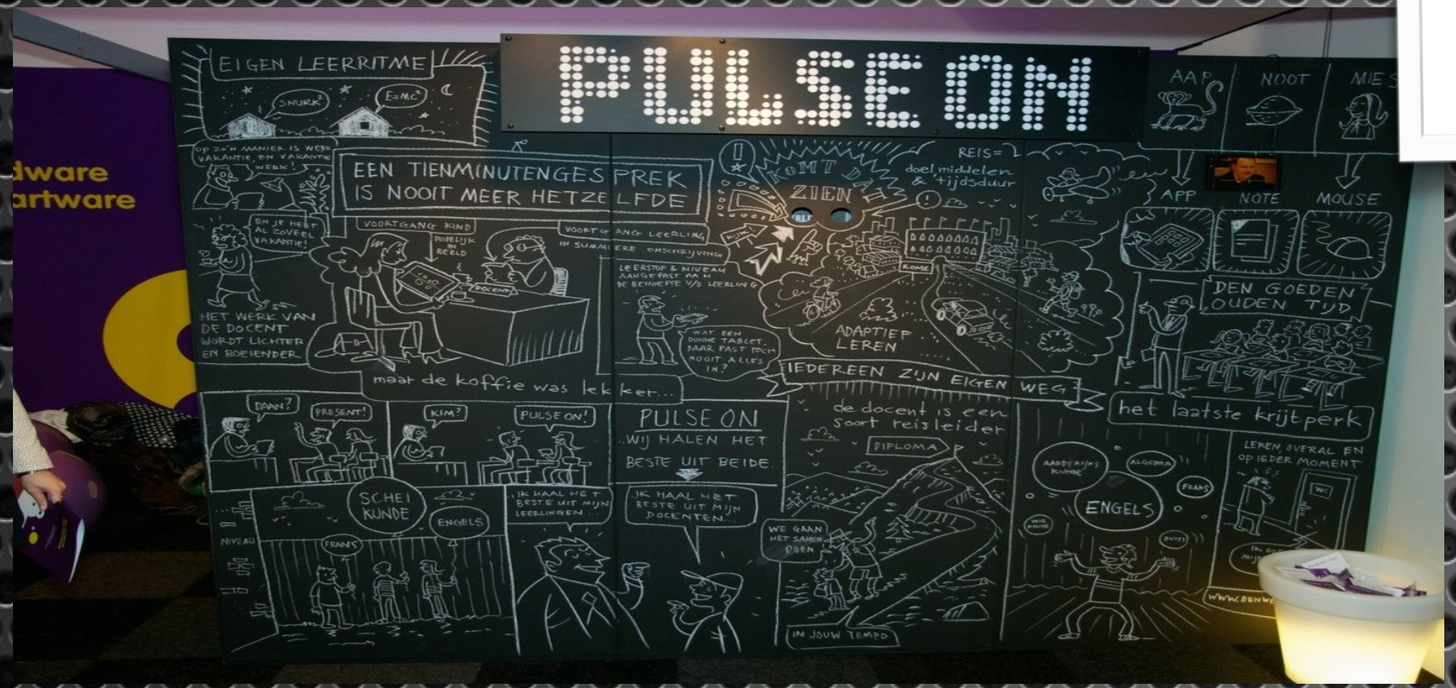
Setup Config Admin

Junit style asserts

Frontend frameworks

jQuery

```
1    <form>
2        <label>Enter a Todo and hit enter!</label>
3        <input type="text" />
4    </form>
5    <ul></ul>
```



Pros

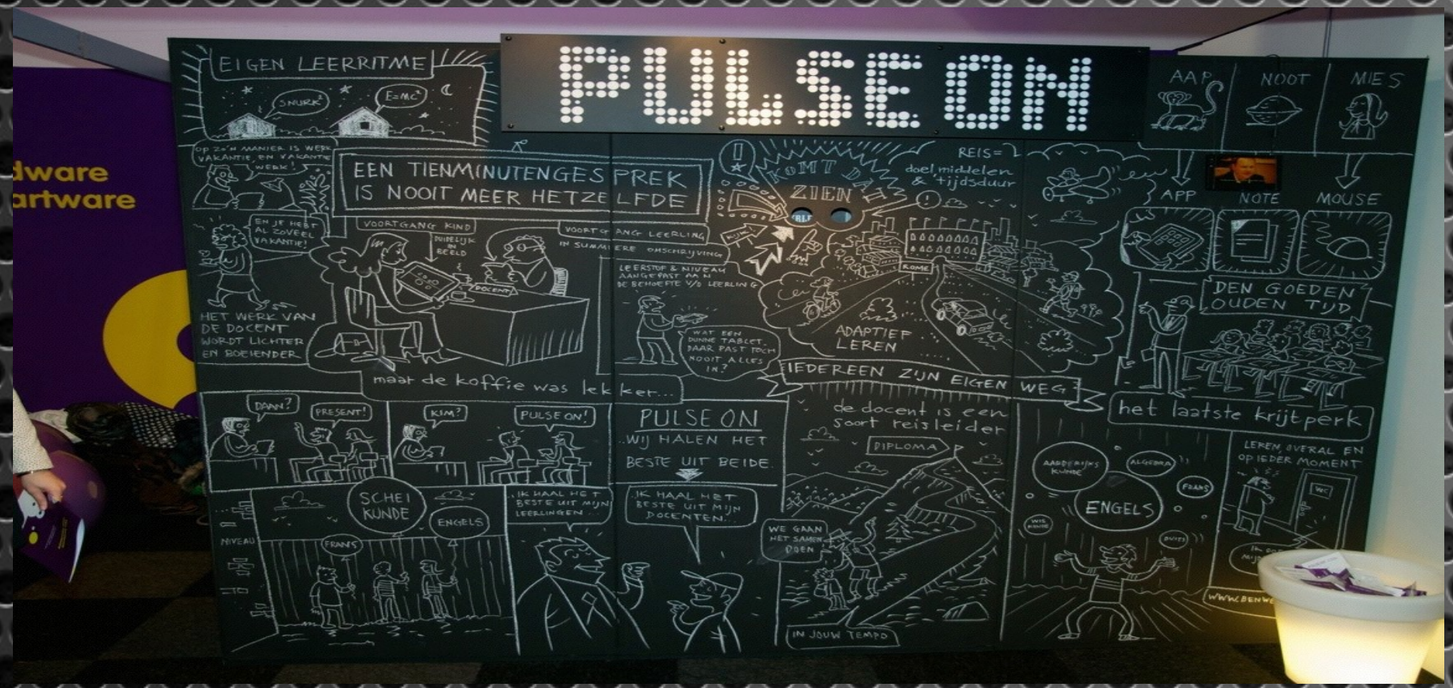
Easy to get started

Can do a lot with not much code

Lot of helpful plugins available

jQuery

```
1    <form>
2        <label>Enter a Todo and hit enter!</label>
3        <input type="text" />
4    </form>
5    <ul></ul>
```



Cons

Hard to maintain

Hard to test

Backbone.js

```
var Todo = Backbone.Model.extend({
    defaults: { title: '', completed: false }
});
var Todos = Backbone.Collection.extend({
    url: '/todos', model: Todo
});
var TodoView = Backbone.View.extend({
    tagName: 'li',
    template: _.template($('#todo-item-template').html()),
    events: {
        'change .completed-chk': 'completedChanged'
    },
    initialize: function() {
        this.model.bind('change', this.render, this);
        this.model.bind('destroy', this.remove, this);
    },
    completedChanged: function() {
        var completed = this.$('.completed-chk').is(':checked');
        this.model.save({
            completed: completed
        });
    },
    render: function() {
        this.$el.html(this.template(this.model.toJSON()));
        return this;
    }
});
var TodoAppView = Backbone.View.extend({
    el: '#todo-app',
    events: {
        'click #add-btn': 'addTodo',
        'click #remove-completed-btn': 'removeCompletedItems'
    },
    initialize: function() {
        this.todos = new Todos();
```

Pros

Well structured

Designed towards consuming REST data

Backbone.js

```
var Todo = Backbone.Model.extend({
    defaults: { title: '', completed: false }
});
var Todos = Backbone.Collection.extend({
    url: '/todos', model: Todo
});
var TodoView = Backbone.View.extend({
    tagName: 'li',
    template: _.template($('#todo-item-template').html()),
    events: {
        'change .completed-chk': 'completedChanged'
    },
    initialize: function() {
        this.model.bind('change', this.render, this);
        this.model.bind('destroy', this.remove, this);
    },
    completedChanged: function() {
        var completed = this.$('.completed-chk').is(':checked');
        this.model.save({
            completed: completed
        });
    },
    render: function() {
        this.$el.html(this.template(this.model.toJSON()));
        return this;
    }
});
var TodoAppView = Backbone.View.extend({
    el: '#todo-app',
    events: {
        'click #add-btn': 'addTodo',
        'click #remove-completed-btn': 'removeCompletedItems'
    },
    initialize: function() {
        this.todos = new Todos();
```

Cons

A LOT OF CODE!!

Hard to maintain

Testable

AngularJS

```
<div class="container" ng-controller="TodoController">
<ul id="todos" class="unstyled">
  <li ng-repeat="todo in todos">
    <label class="checkbox">
      <input type="checkbox" ng-model="todo.completed"
             ng-change="changeCompleted(todo)" />{{todo.title}}
    </label>
  </li>
</ul>
<form class="form-inline">
  <input id="todo-title" type="text" ng-model="newTodoTitle" />
  <button id="add-btn" class="btn btn-success" ng-click="addTodo(newTodoTitle)">Add</button>
</form>
</div>
```

AngularJS

```
1   var todoApp = angular.module('todoApp', []);
2   todoApp.controller('TodoController', function($scope, $http) {
3       $scope.todos = [];
4
5       $http.get('/todos').success(function(todos) {
6               $scope.todos = todos;
7           });
8
9       $scope.addTodo = function(title) {
10          $scope.todos.push({ title : title });
11      };
12
13      $scope.removedTodo = function(todo) {
14          $scope.todos = $scope.todos.splice(
15              $scope.todos.indexOf(todo) ,1);
16      }
17  });
```

Pros

Well structured

Designed towards consuming REST data

Testable

No guessing in html

TypeScript

```typescript
class ItemService {

    private items : string[] = [];

    public getItems() : string[] {
        return this.items;
    }
}

export = ItemService;
```

```javascript
define(["require", "exports"], function(require, exports) {
    var ItemService = (function () {
        function ItemService() {
            this.items = [];
        }
        ItemService.prototype.getItems = function () {
            return this.items;
        };
        return ItemService;
    })();


    return ItemService;
});
```

Pros

compile time checks

Code completion

TypeScript

```typescript
class ItemService {

    private items : string[] = [];

    public getItems() : string[] {
        return this.items;
    }
}

export = ItemService;
```

```javascript
define(["require", "exports"], function(require, exports) {
    var ItemService = (function () {
        function ItemService() {
            this.items = [];
        }
        ItemService.prototype.getItems = function () {
            return this.items;
        };
        return ItemService;
    })();


    return ItemService;
});
```

Cons

Need to have a definition to work

Slow-ish

Wrap up

A modular architecture gives us:

- ☐ Maintainability
- ☐ Extensibility
- ☐ Freedom to change

But what if I want

Spring

EJB

Hibernate?

Just don't... You really don't need to.

Cloud provisioning

http://ace.apache.org/

Cloud OSGi services

http://www.amdatu.org/

Amdatu

Paul Bakker

paul.bakker@luminis.eu

@pbakker

Eclipse OSGi plugin

http://bndtools.org/

That's us

http://luminis-technologies.com

Jago de Vreede

jago.devreede@luminis.eu

Learn more?

Friday 10.20 - 12.50 or
14.45 - 17.15

Tutorial "Developing modular cloud applications with OSGi"