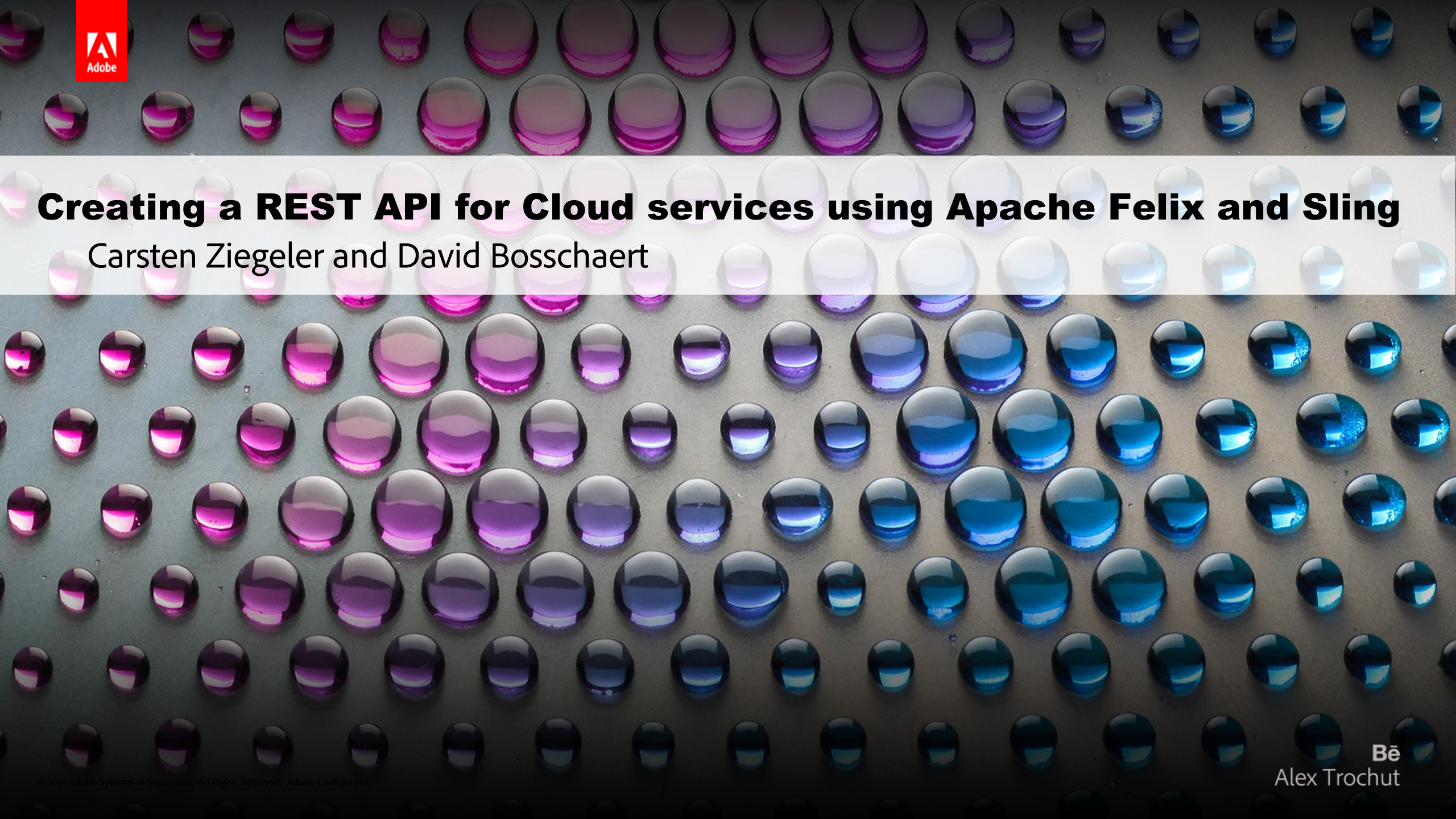


Creating a REST API for Cloud services using Apache Felix and Sling

Carsten Ziegeler and David Bosschaert

Alex Trochut

Agenda

- A REST site with Apache Sling
- Scale it up! with OSGi

Bē
Natalie Shau

Speakers

- David Bosschaert (davidb@apache.org)
  - R&D Adobe Research Dublin
  - Co-chair OSGi EEG
  - ISO/IEC JTC1 SC38 (Cloud Computing)
  - Open-source and cloud enthusiast
- Carsten Ziegeler (cziegeler@apache.org)
  - R&D Adobe Research Switzerland
  - OSGi Board, CPEG, EEG, IoT member
  - ASF member

Sling

Bē
The Made Shop

The Next Great Application

ROA and REST

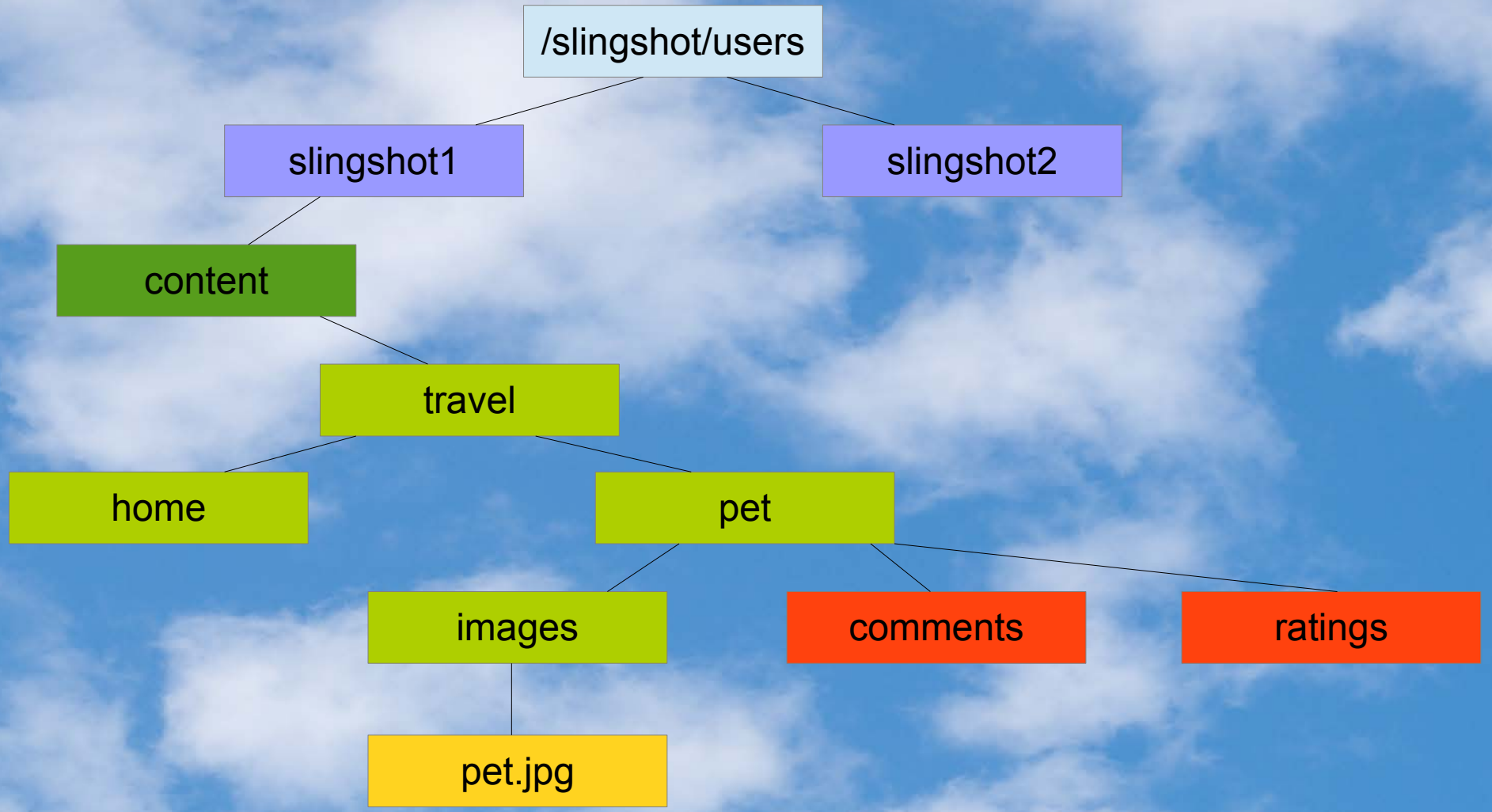
- http://…/products.jsp?id=5643564

- http://…/slingshot/users/slingshot1/content/travel/pet.(html|json)

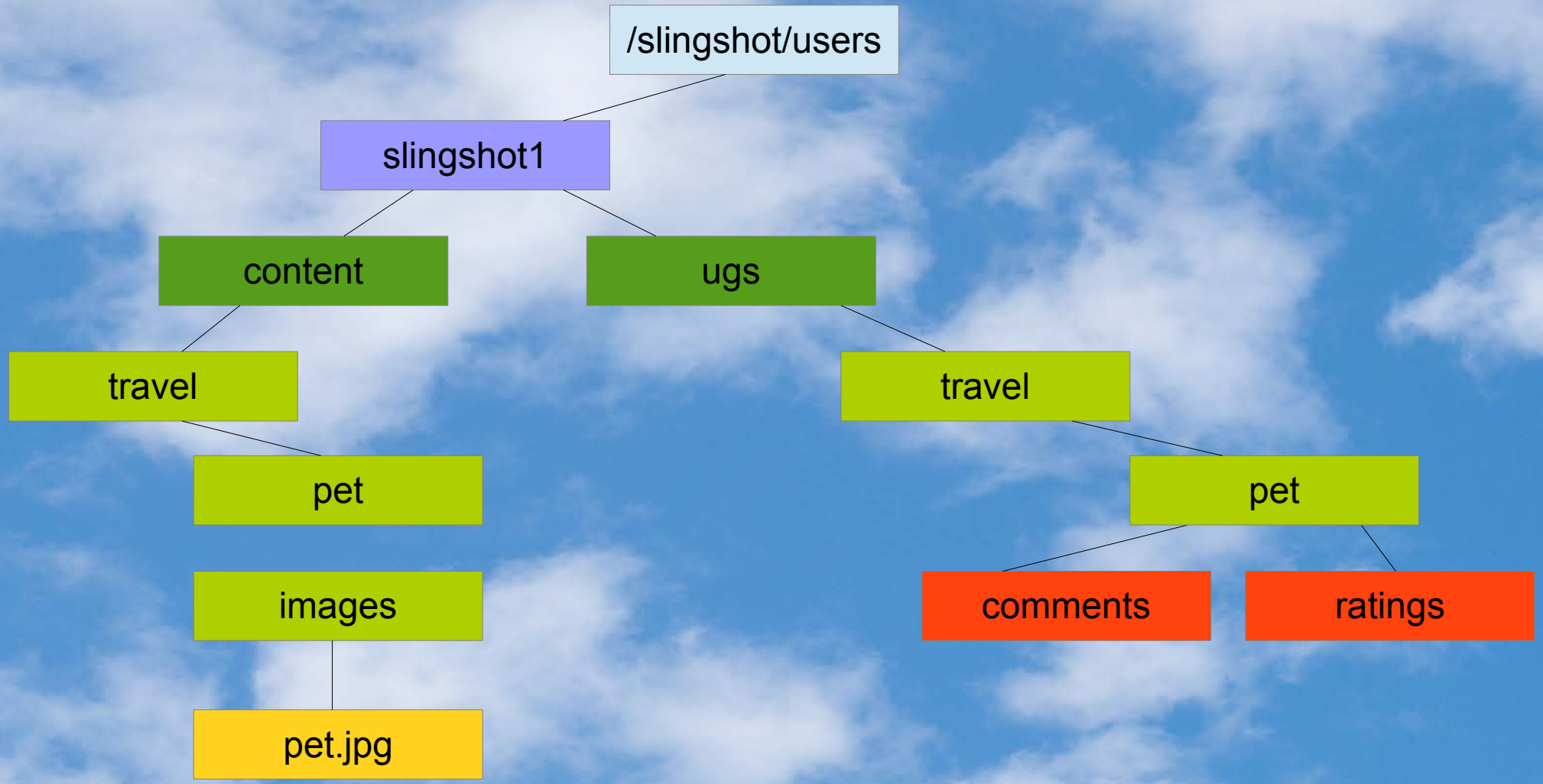
Resource Oriented Architecture

- Every piece of information is a resource
  - User home, category, item, image, comment
  - Descriptive URI
- Stateless web architecture (REST)
  - Request contains all relevant information
  - Targets the resource
- Leverage HTTP
  - GET for rendering, POST/PUT/DELETE for operations

SlingShot Content Structure



SlingShot Content Structure



Resource-first Request Processing

- Request: /slingshot/users/slingshot1/ugs/travel/pet/comments.10.json

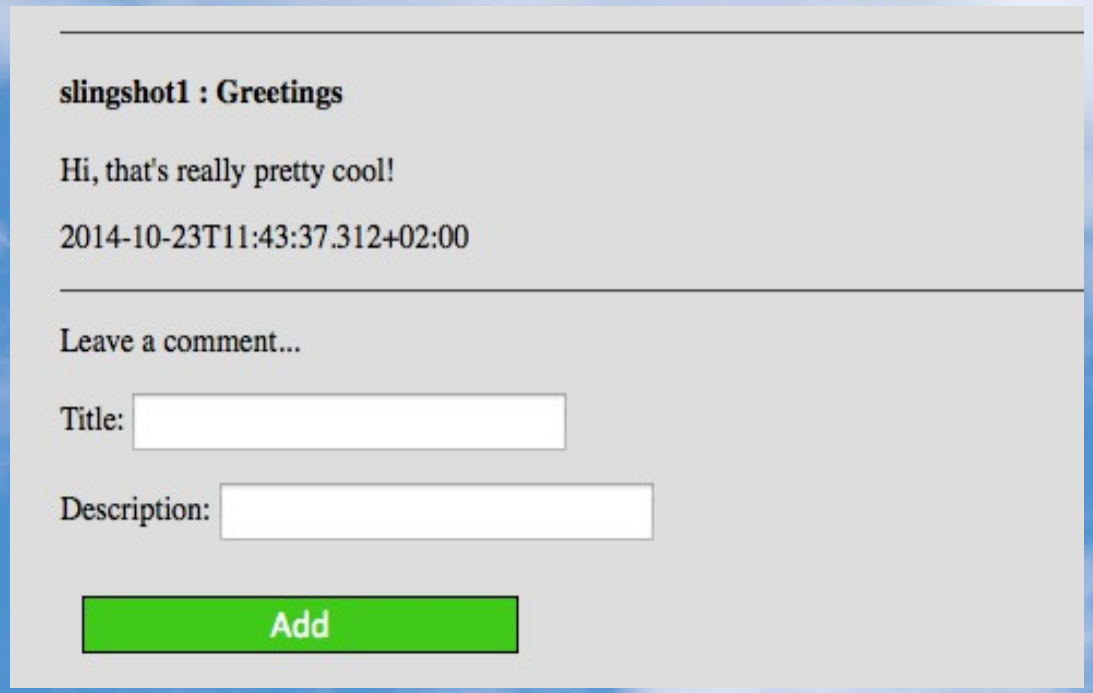
- Resource: /../slingshot1/ugs/travel/pet/comments
- Extension: json
- Selector: 10
- Method: GET
- Resource Type : slingshot/Comments

Resource-first Request processing

1. Resolve the resource
2. Resolve processing script
3. Build execution chain
4. Execute

Scripting Inside

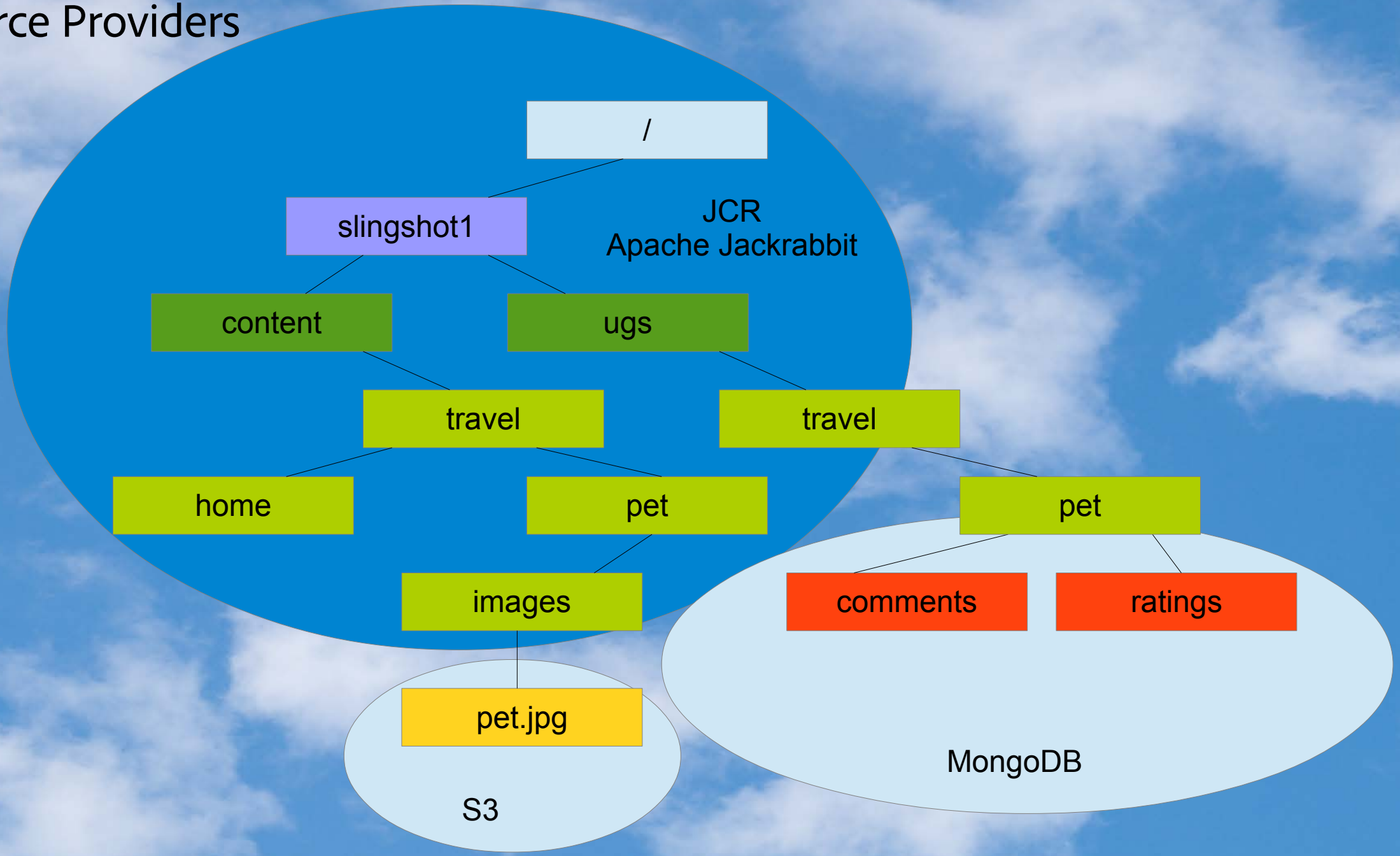
- It's your choice
  - JSP, servlet, ESP, Scala, **Sightly**
  - javax.script
  - own script handlers
- Default servlets
  - JSON, XML
  - POST/update/create/delete handling
  - Error handling




```
<sling:defineObjects/>
<div class="comment">
<%
    final ValueMap attr = resource.getValueMap();

    final String title = attr.get(PROPERTY_TITLE,
                                  resource.getName());
    final String description = attr.get(PROPERTY_DESCRIPTION, "");
    final String userId = attr.get(PROPERTY_USER, "");
%>
    <h4><%= ResponseUtil.escapeXml(userId) %> :
                              <%= ResponseUtil.escapeXml(title) %></h4>
    <p><%= ResponseUtil.escapeXml(description) %></p>
    <p><%= ResponseUtil.escapeXml(
            attr.get(SlingshotConstants.PROPERTY_CREATED, "")) %></p>
</div>
```

Resource Providers

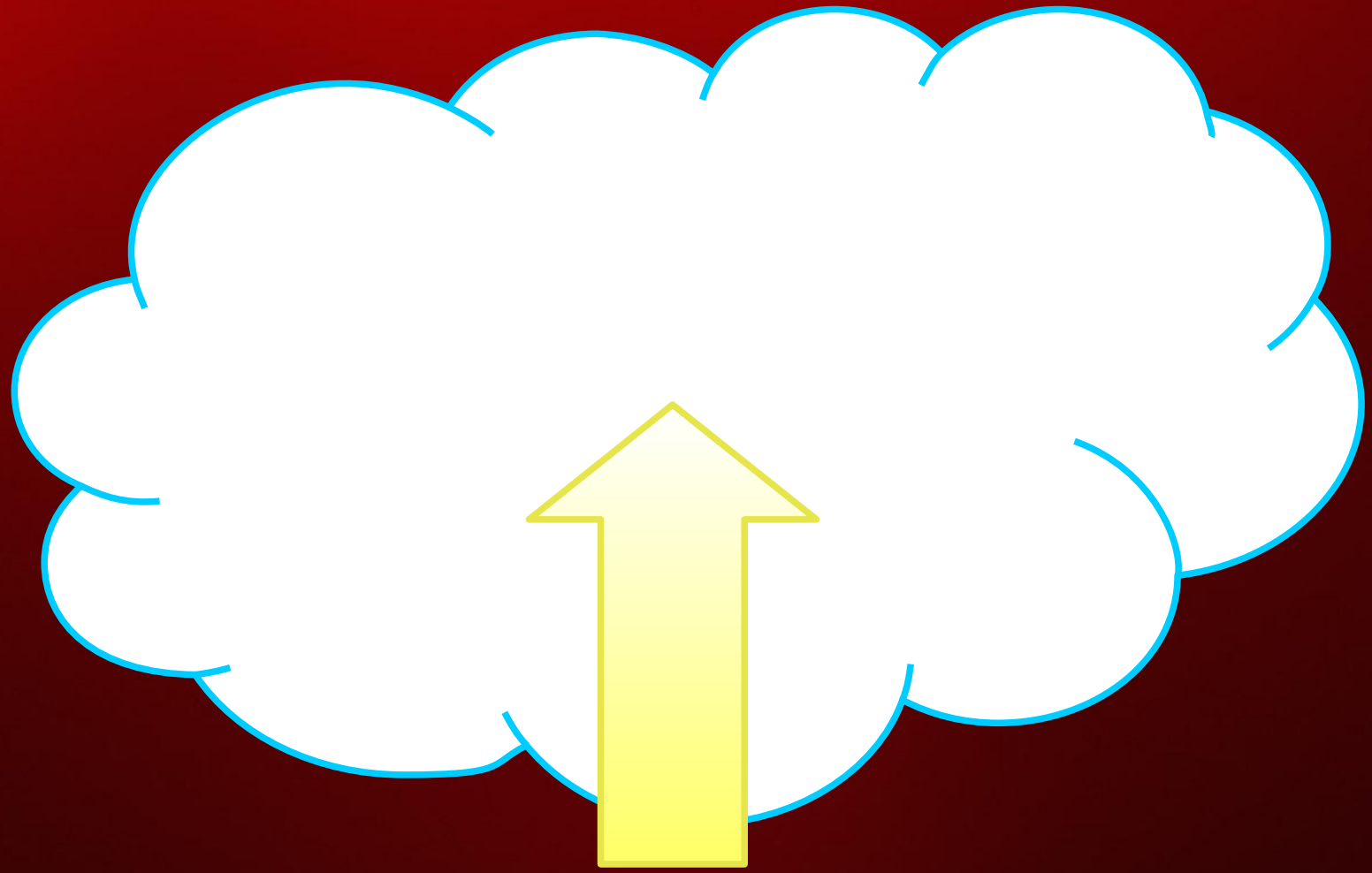


Think!

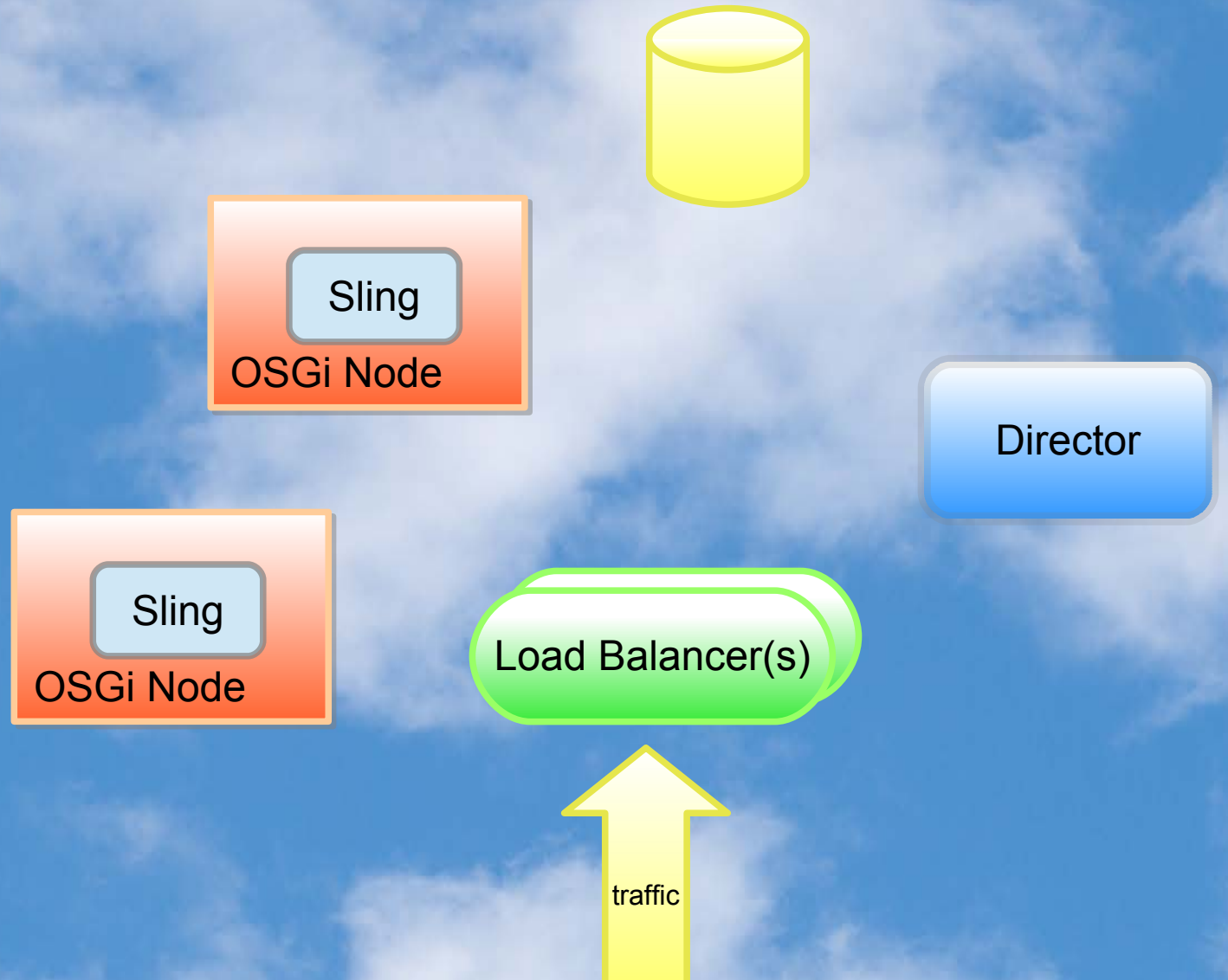
- Content structure and ACLs
- OAK
- Sightly
- Keep it simple!

Scale UP!

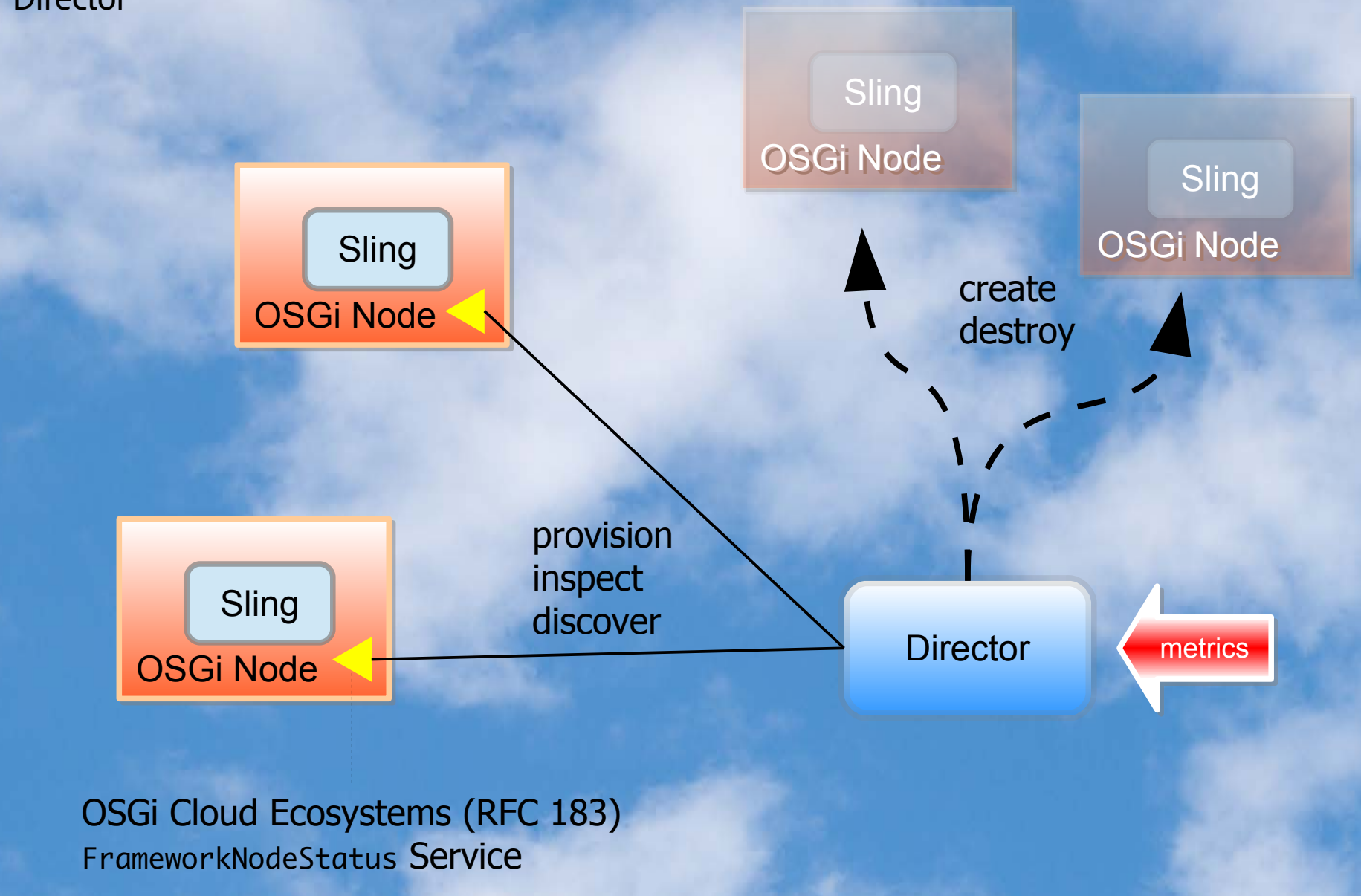
- Effectively use the cloud
- Scale up/down based on need
  - start small = cheap
- No downtime
  - when moving to larger machines
- Avoid cloud-provider specifics



Sling in the cloud



Director



OSGi Cloud Ecosystems (RFC 183)

`FrameworkNodeStatus` Service

Node Status

```
public interface NodeStatus {
        MetricsDTO getMetricsDTO()
}
```

org.osgi.node.id	Unique ID, e.g. **docker** ID
org.osgi.node.endpoint	An endpoint to access this node
org.osgi.node.vendor	E.g. Amazon WS, MS Azure, etc…
org.osgi.node.country	ISO 3166-1 country code
org.osgi.node.tags	Custom tags
...	… there are more properties …

Service properties

CPULoad	percentage
freeMemory	in bytes
networkThroughput	bytes/sec
diskThrougput	bytes/sec

Metrics DTO

Framework Node Status

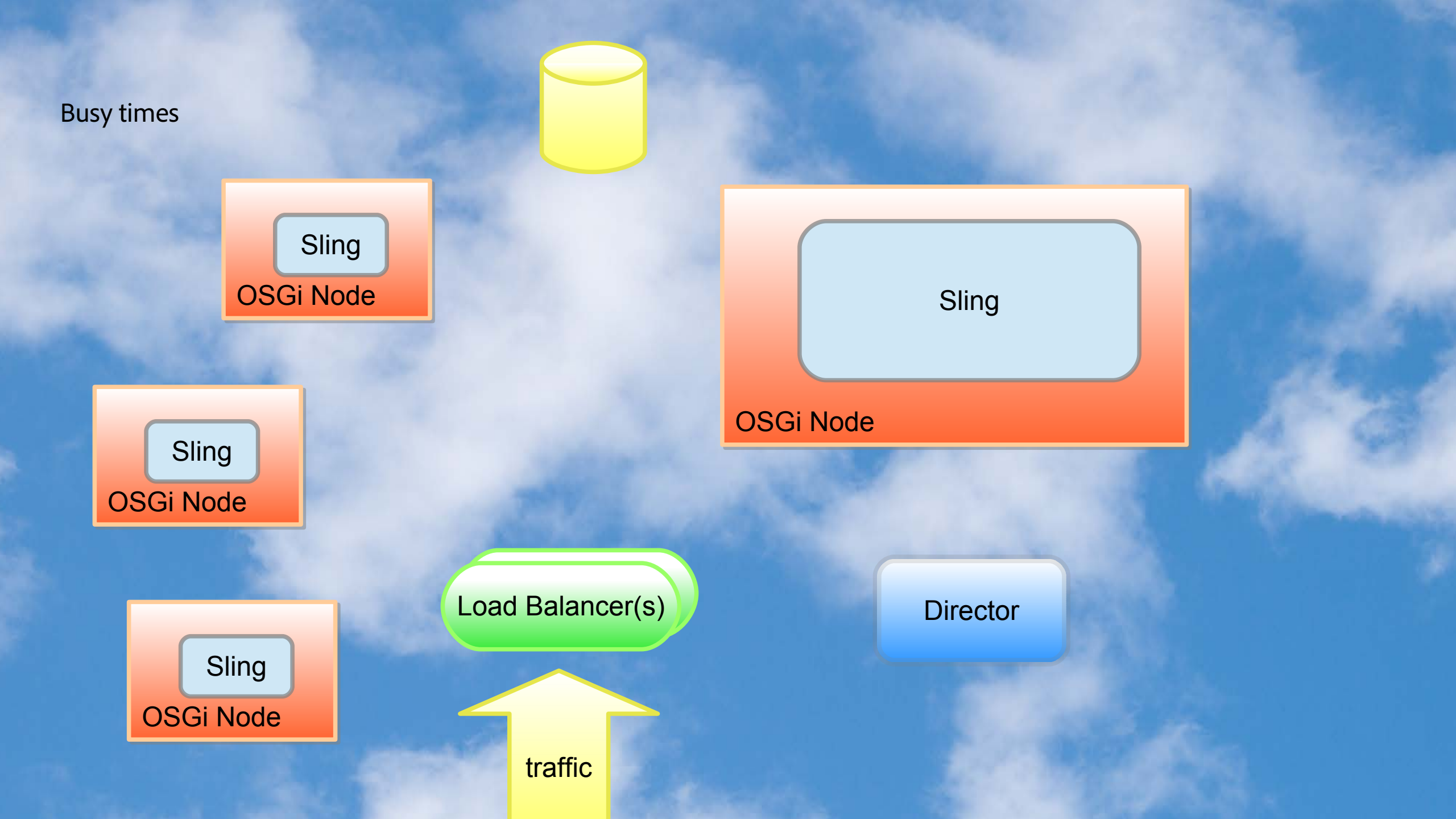
```
public interface FrameworkNodeStatus extends NodeStatus {
    FrameworkManager getFrameworkManager()
}
```

org.osgi.node....	...all the properties from NodeStatus
org.osgi.framework.version	Version of framework running
org.osgi.framework.processor	Processor
org.osgi.framework.os	Operating system
java.version	Java version
… there are more …	

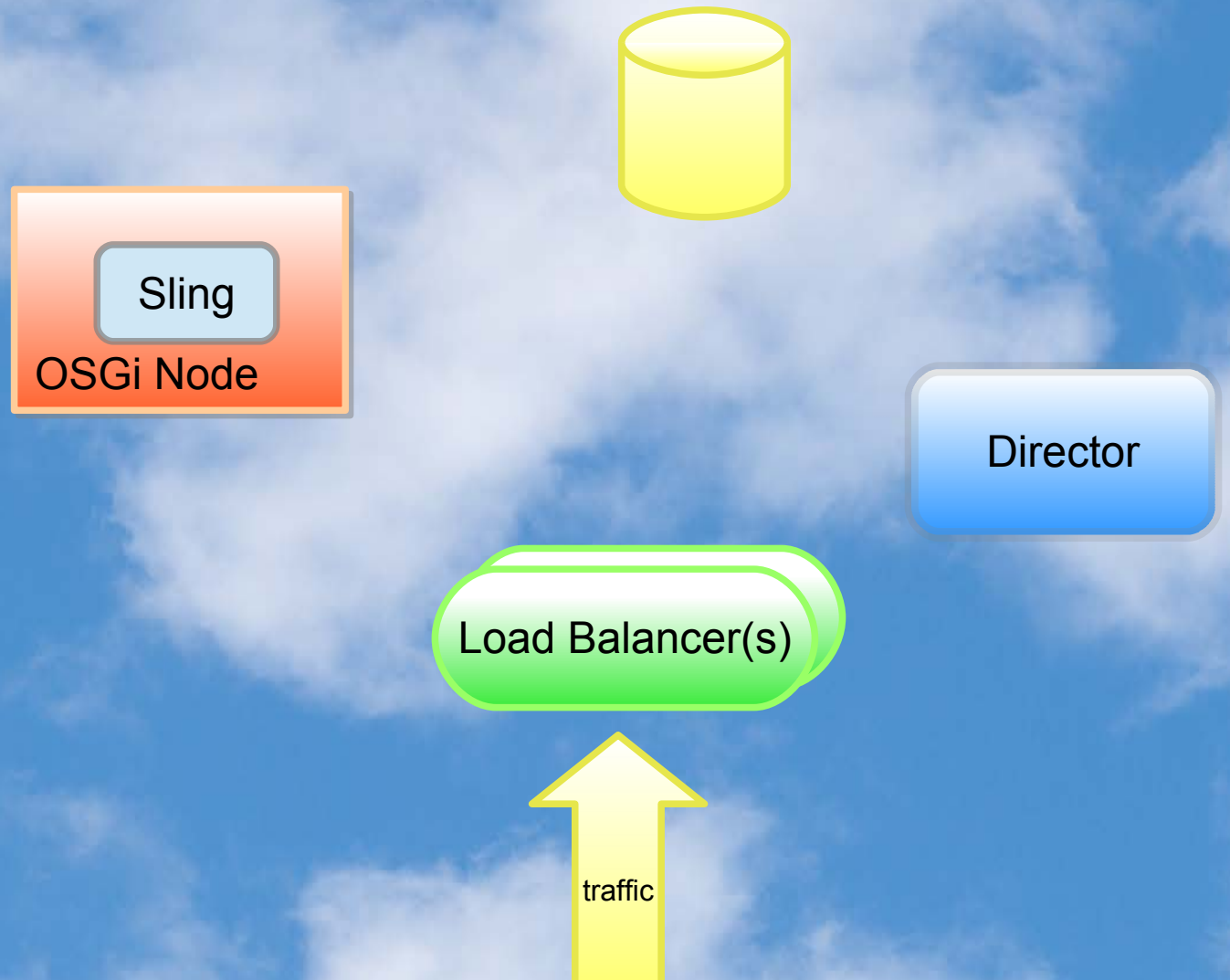
Service properties

```
public interface FrameworkManager {
    Collection<BundleDTO> getBundles();
    BundleDTO installBundle(String location,
                            InputStream is);
    void startBundle(long id);
    void stopBundle(long id);
    BundleDTO uninstallBundle(long id);
    // … and so on …
}
```

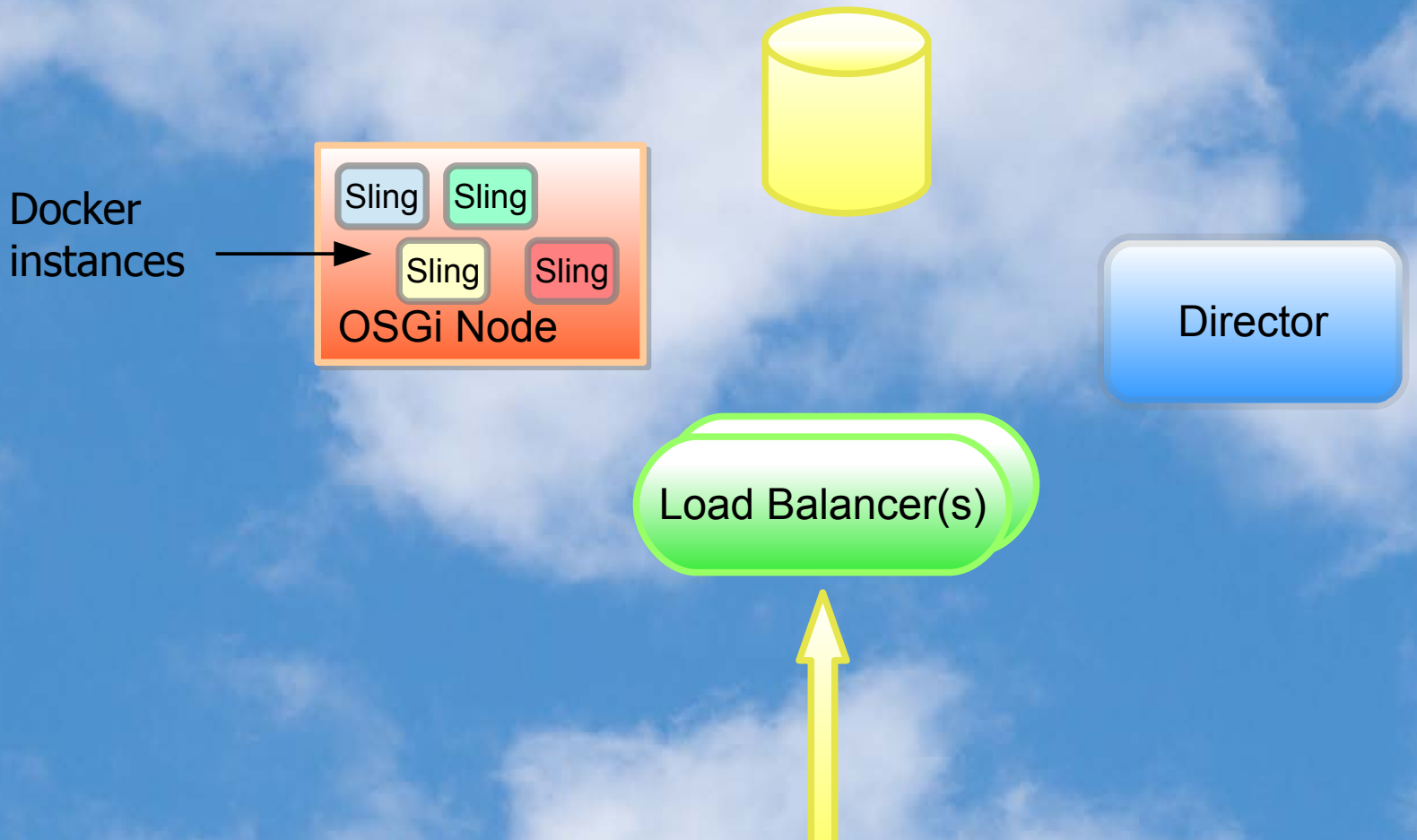
Busy times

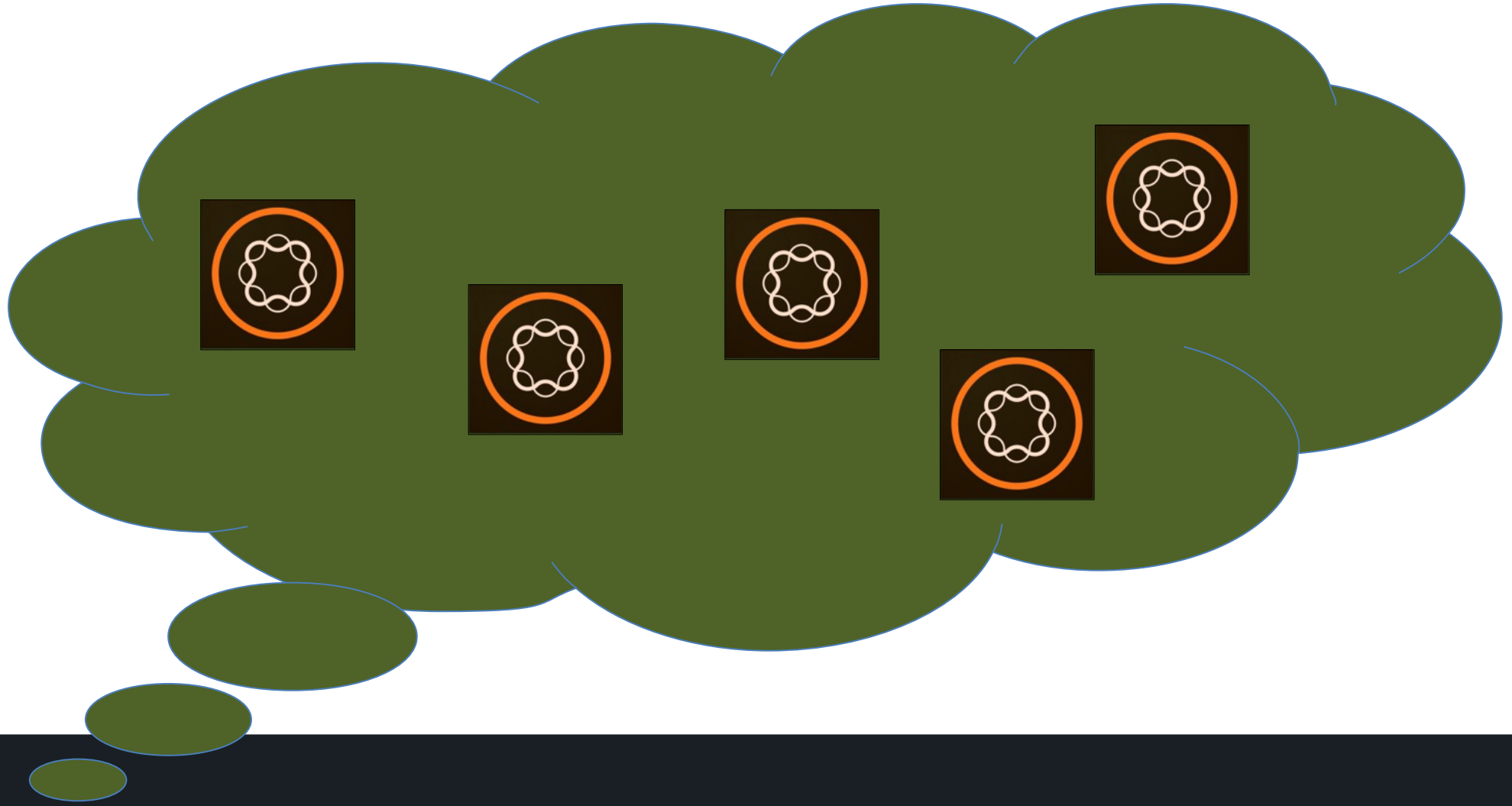

Sling
OSGi Node
Sling
OSGi Node
Sling
OSGi Node
Sling
OSGi Node
Load Balancer(s)
Director
traffic

Quiet times



Really Quiet times

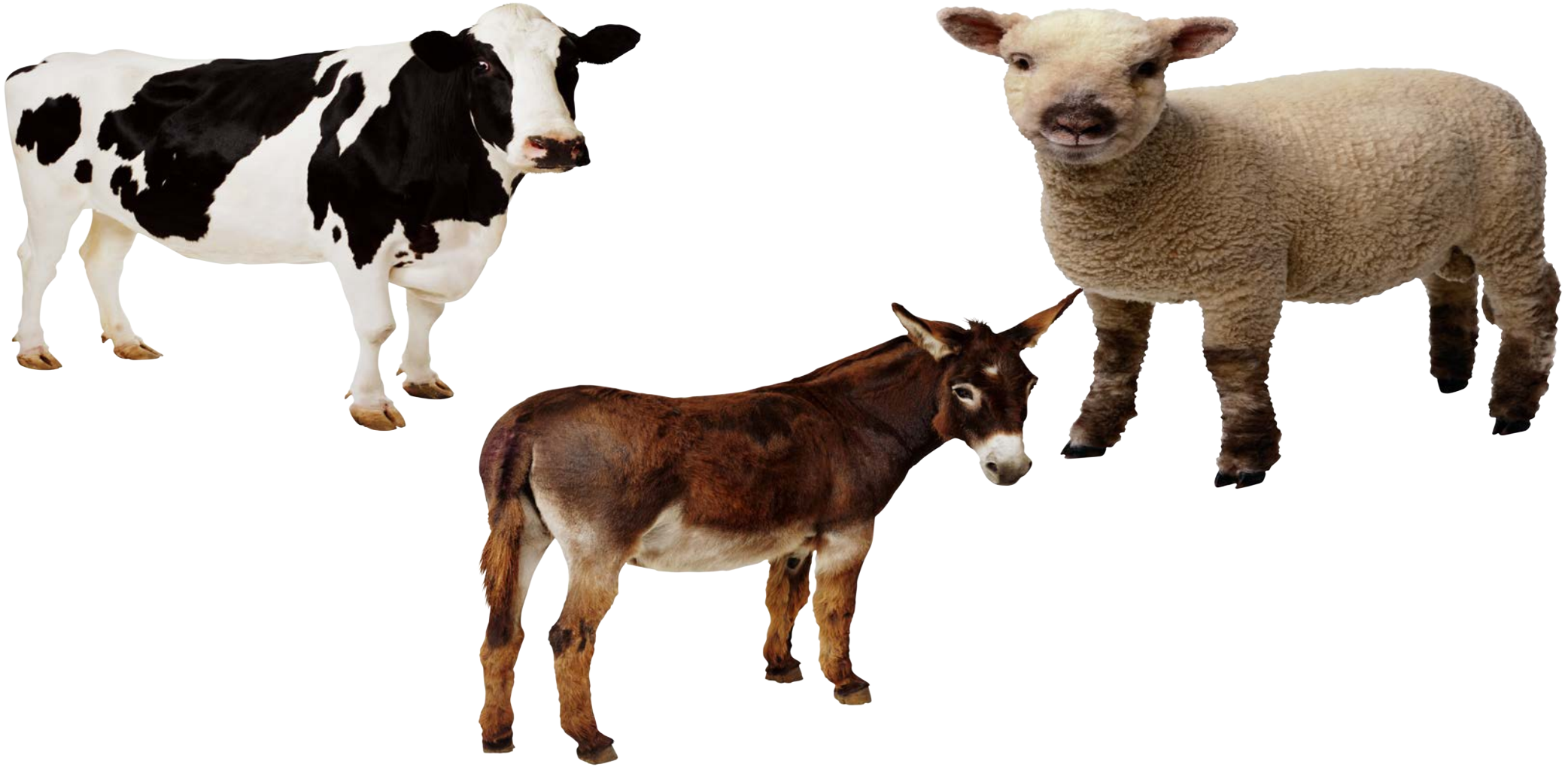




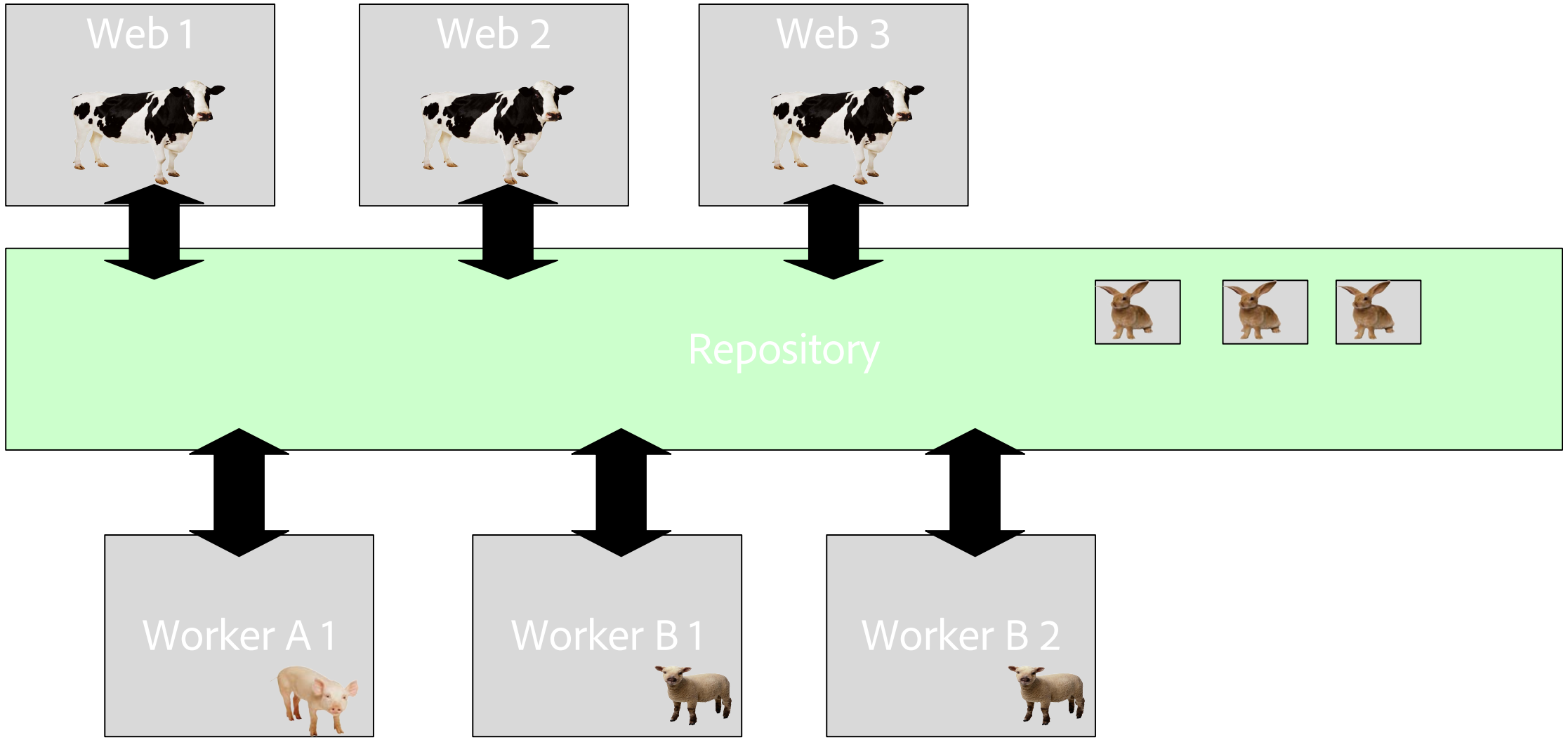
Instances

Pets vs cattle

Farmville



Herding



Modularity ^2

Spezialized instances

Create / Destroy on demand

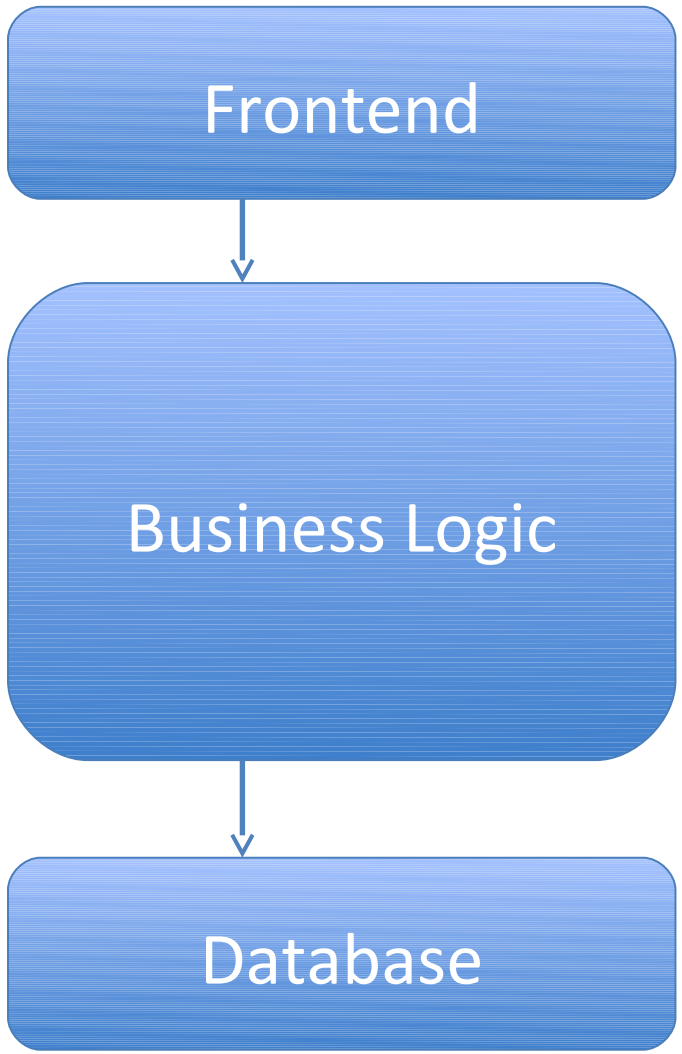
Independent scaling

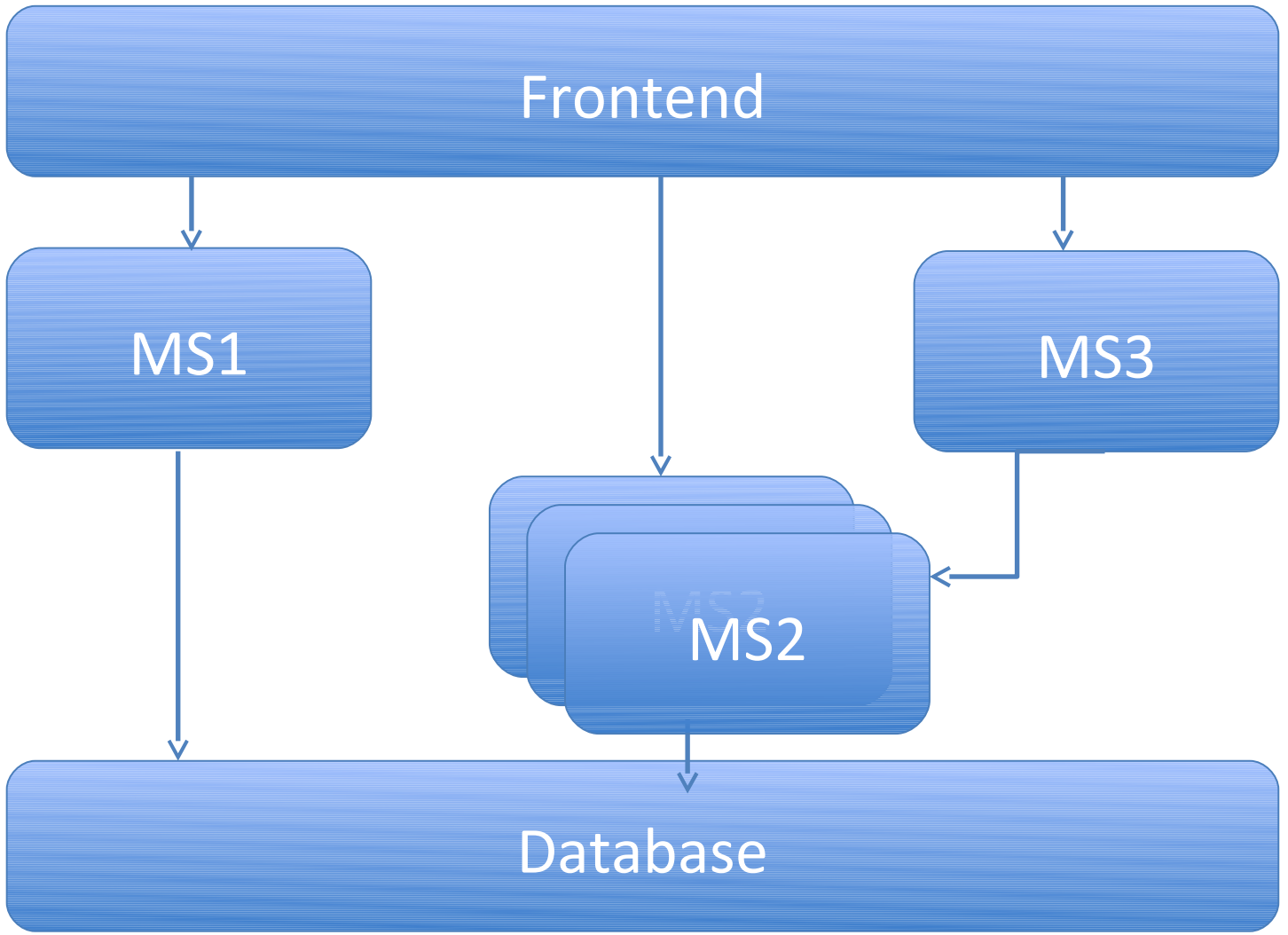
Faster dev/test/prod cycles

New Kid on the Block

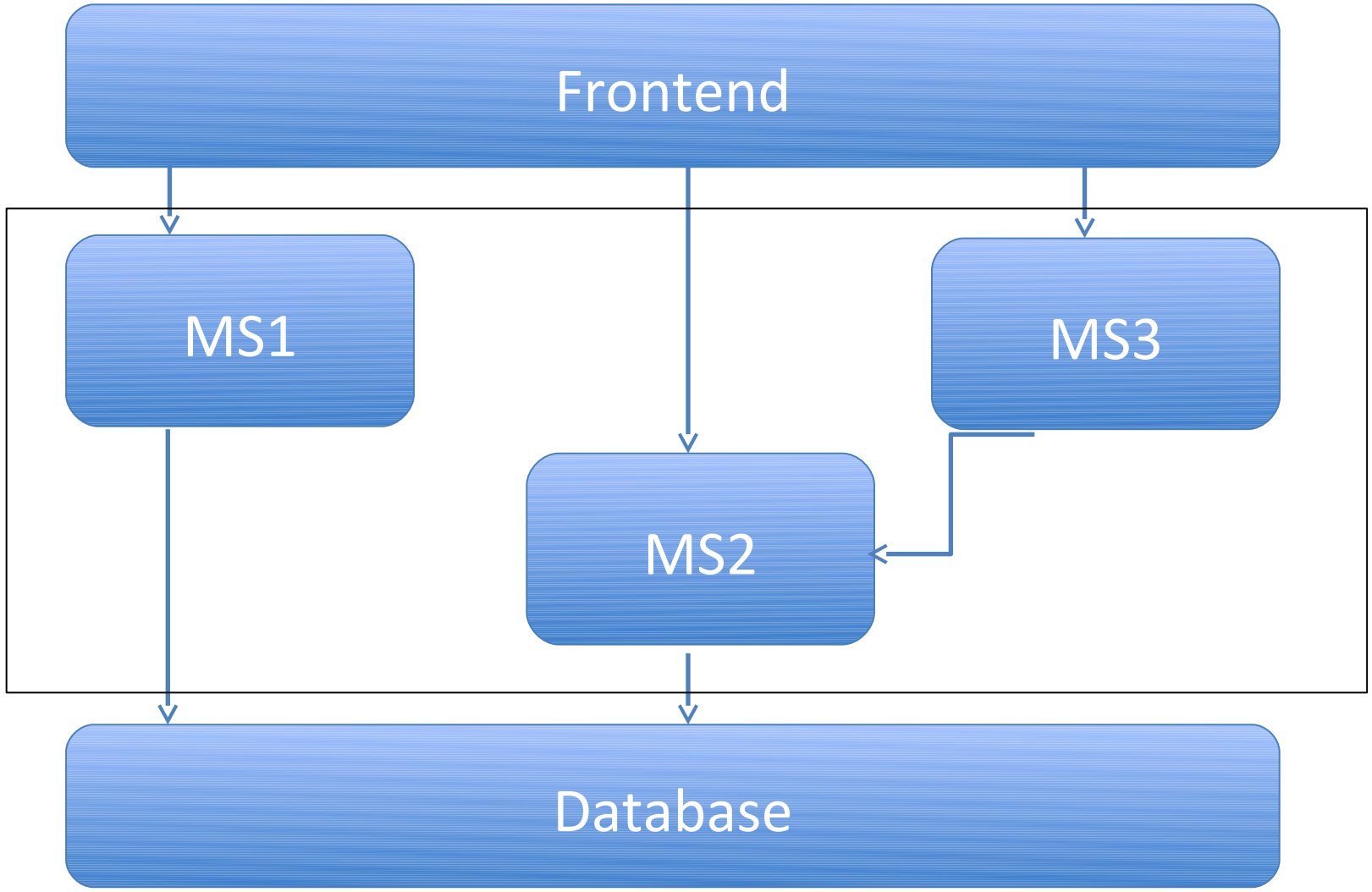
MICROSERVICES

Traditional vs MS Architecture





BTDT



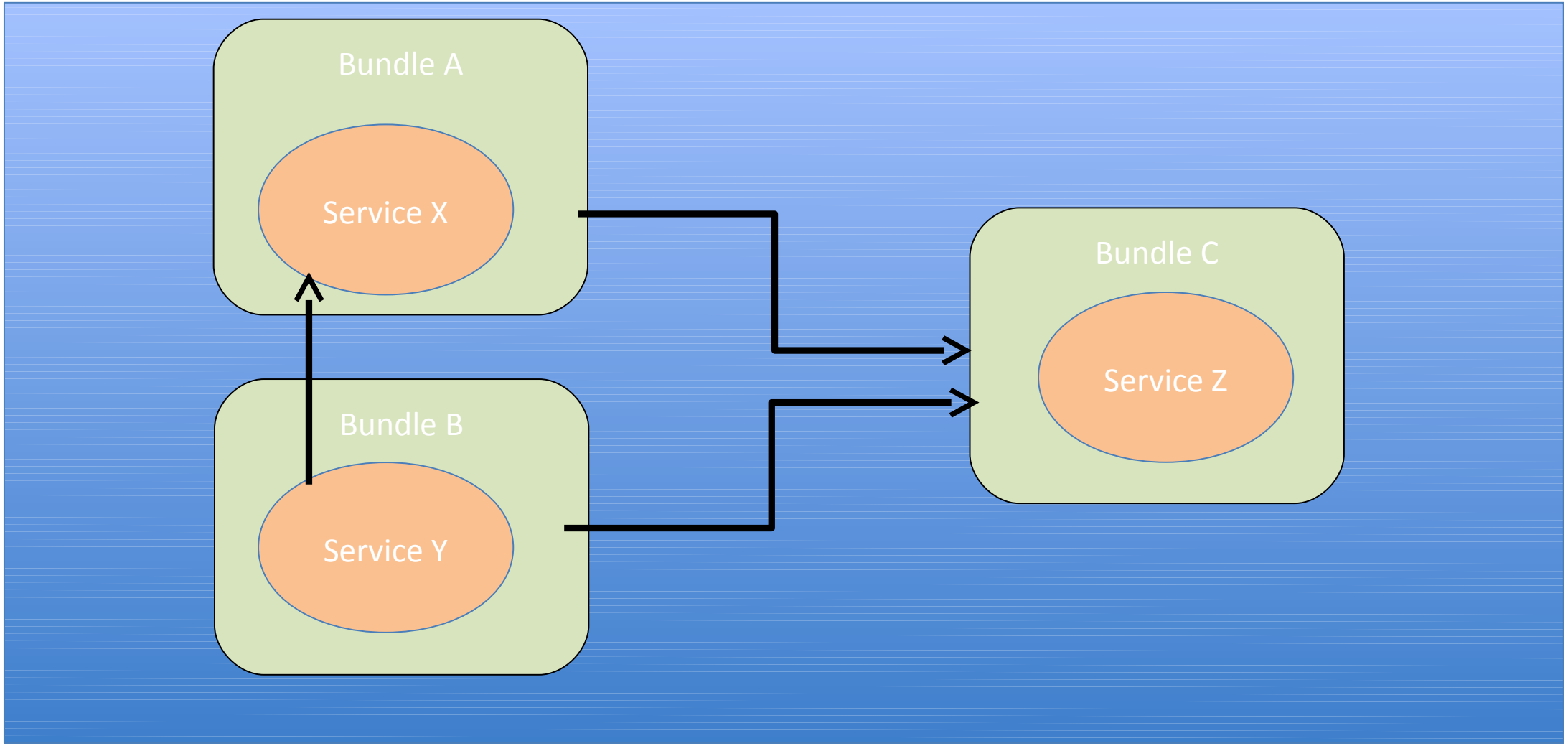
Games without Frontiers

Topology

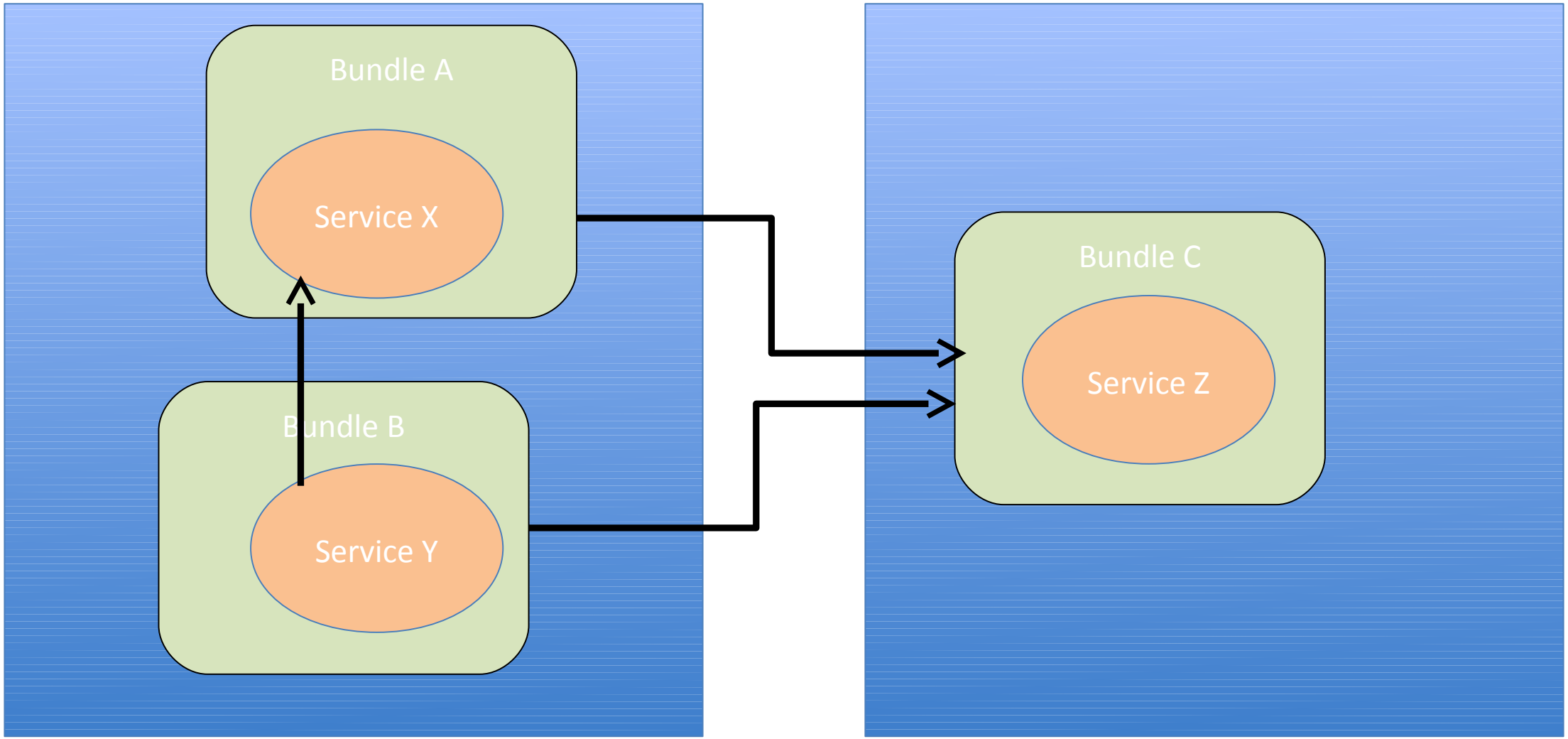
REST

Distributed OSGi

Single Instance



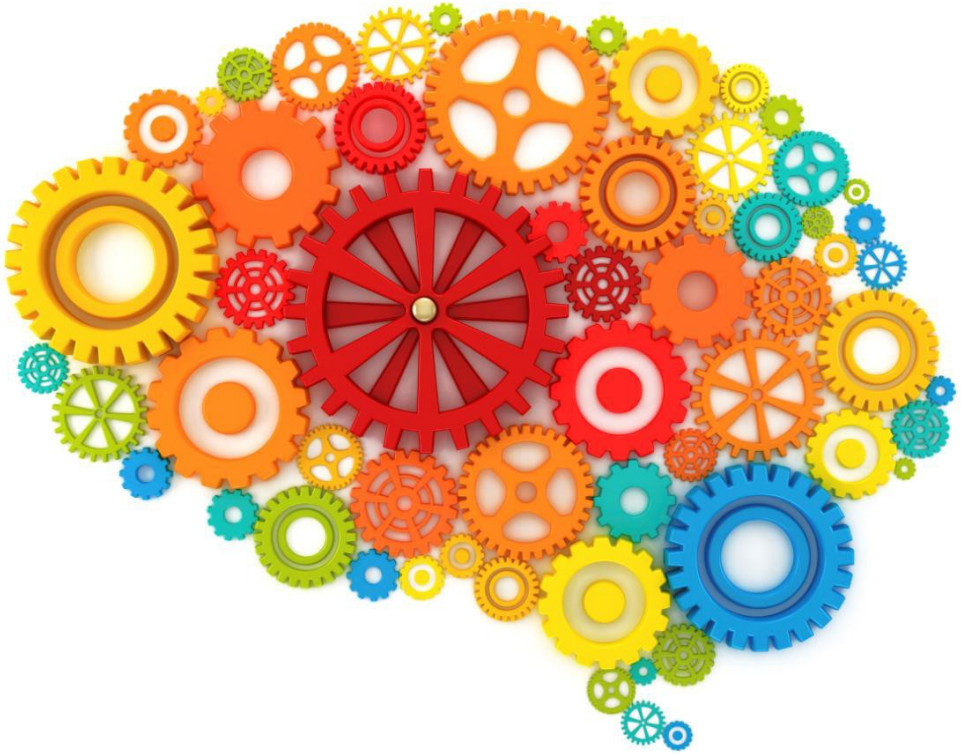
Two Instances



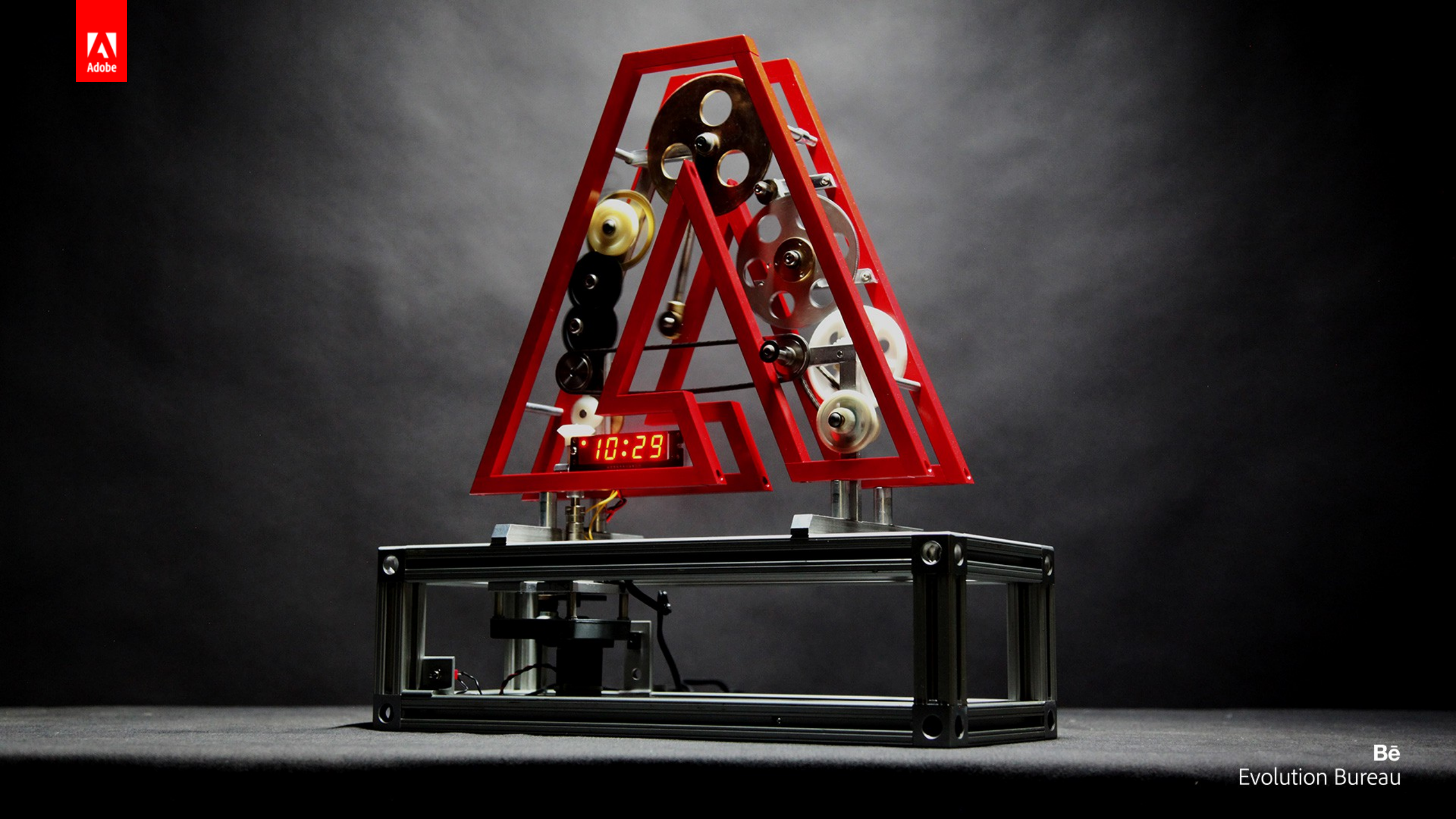
Summary

- Use Sling to build your REST service
- Write your app using anything that can interact with REST
  - Native app
  - Javascript
  - …
- Use OSGi Cloud Ecosystems, a cloud abstraction (such as Apache jclouds) and Docker to scale
- Be cloud-vendor independent!

Links



- Apache Sling – http://sling.apache.org
- Apache Felix – http://felix.apache.org
- SlingShot Sample application – http://svn.apache.org/repos/asf/sling/trunk/samples/slingshot
- OSGi Cloud Ecosystems RFC 183 – https://github.com/osgi/design/tree/master/rfcs/rfc0183



Adobe

Bē
Evolution Bureau