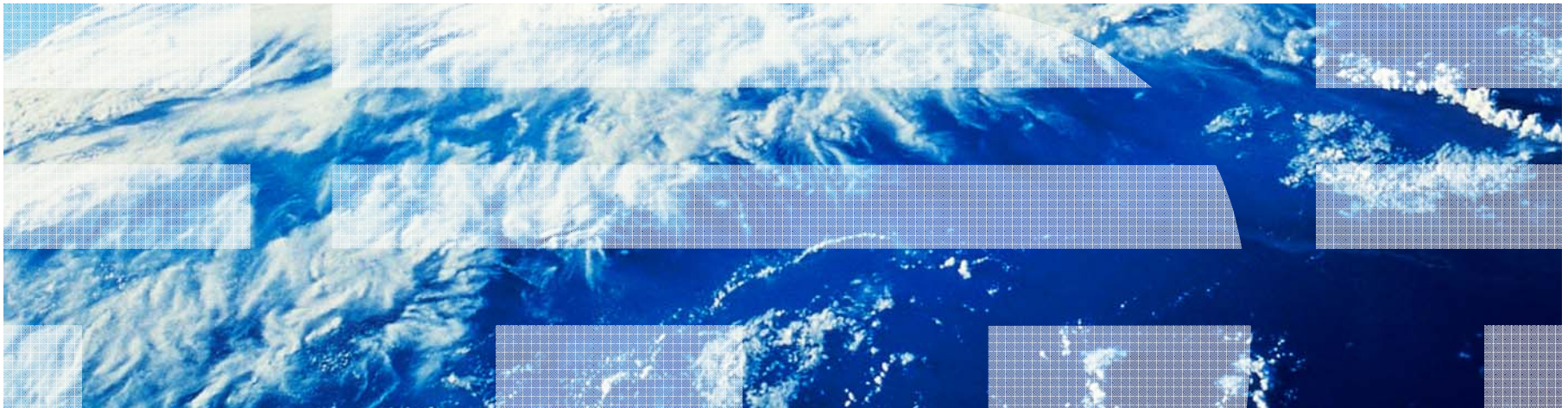


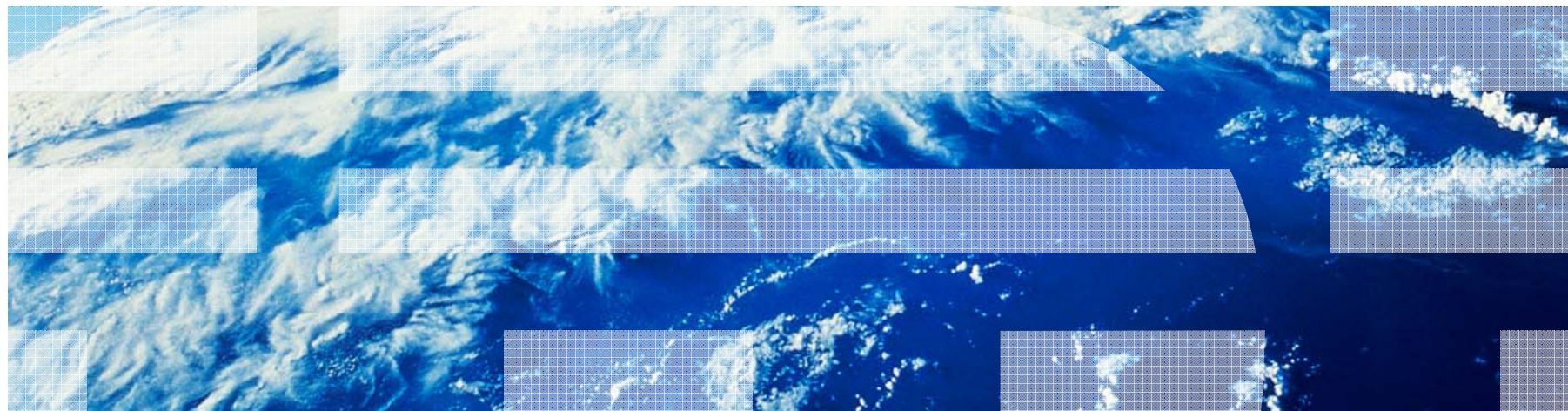
Keeping Your Options Open, Even if the Cloud is Not



Agenda

- Portability and interoperability
- A few words about APIs
- Controlling VMs with Apache libcloud
- The Simple Cloud API
 - Storage
 - Queues
 - Documents
- Resources / Next steps

The problem



Vendor lock-in

- If there's a new technology, any talented programmer will want to use it.
 - Maybe the shiny new thing is appropriate for what we're doing.
 - Maybe not.
 - We're probably going to use it anyway.
- The challenge is to walk the line between using the newest, coolest thing and avoiding vendor lock-in.

Portability and Interoperability

- In writing flexible code for the cloud, there are two key concepts:
 - **Portability** is the ability to run components or systems written for one cloud provider in another cloud provider's environment.
 - **Interoperability** is the ability to write one piece of code that works with multiple cloud providers, regardless of the differences between them.

How standards work

- For a standards effort to work, three things have to happen:
 - The standard has to solve a common problem in an elegant way.
 - The standard has to be implemented consistently by vendors.
 - Users have to insist that the products they use implement the standard.

How standards work

- **All three things have to happen.**
 - If the standard doesn't solve a common problem, or if it solves it in an awkward way, the standard fails.
 - If the standard isn't implemented by anyone, the standard fails.
 - If customers buy and use products even though they don't implement the standard, the standard fails.

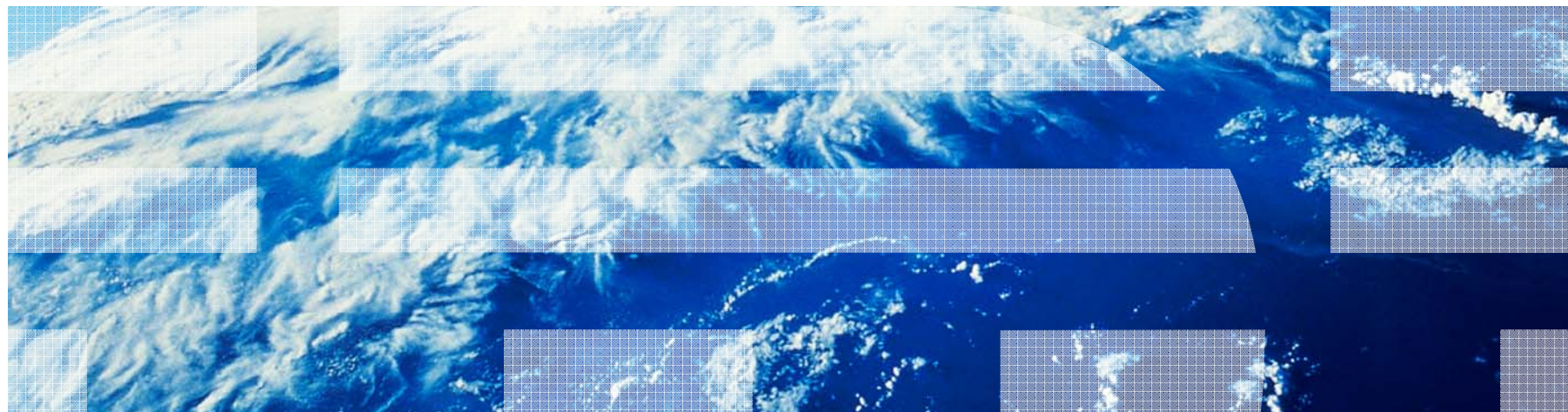
Portability

- The portability of your work depends on the platform you choose and the work you're doing.
 - A GAE application
 - An Azure application
 - An AMI hosting an application container
 - A SimpleDB database
 - An Amazon RDS database

Interoperability

- Discussions of openness often focus on leaving one cloud provider and moving to another.
- In reality, it's far more common that you'll have to write code that works with multiple providers at the same time.

Our Demo



A sample cloud application

- An order entry application running on one or many VMs in the cloud
 - Creates a purchase order and sends it to a cloud-based queue for processing.
- A queue that holds the POs
- An order processing application running on one or many VMs in the cloud
 - Creates an invoice and stores it in the cloud
- Cloud-based storage that stores the invoices

A cloud-optimized application

- If we need more computing power to create orders, we fire up more VMs.
- The queue of orders has unlimited capacity.
- If we need more computing power to process orders, we fire up more VMs.
- The storage for invoices has unlimited capacity.
- When we don't need VMs, we shut them down.

The demo



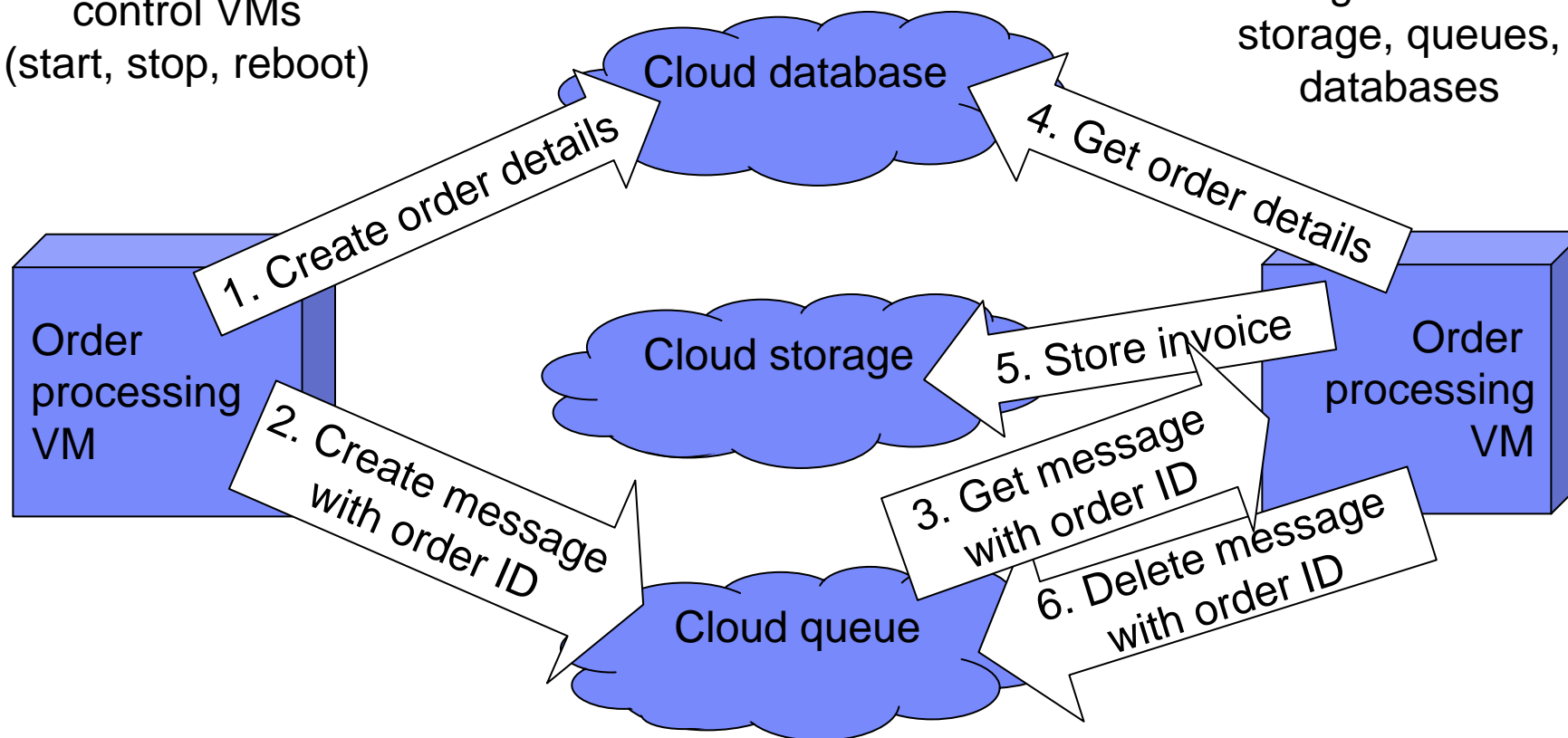
Demo Architecture



Single API to control VMs
(start, stop, reboot)



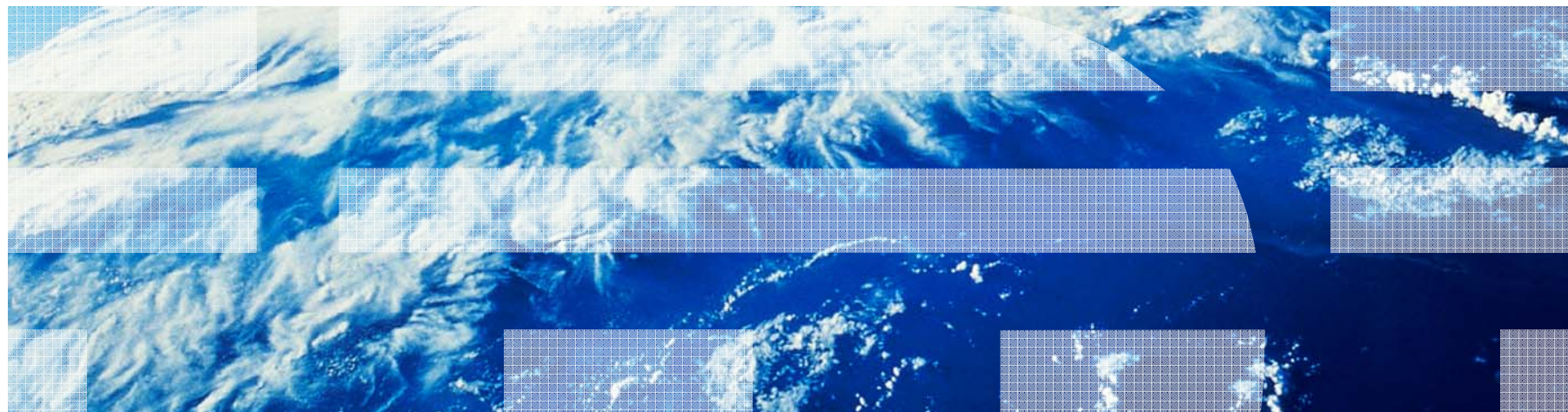
Single API for storage, queues, databases



The message

- Simple Cloud lets us access storage and queue and database services in the cloud.
- libcloud lets us control VMs from different vendors.

A few words about APIs



Levels of APIs

- How developers invoke a service:
 - Level 1 – Write directly to the REST or SOAP API.
 - Level 2 – Use a language-specific toolkit to invoke the REST or SOAP API.
 - Level 3 – Use a service-specific toolkit to invoke a higher-level API.
 - Level 4 – Use a service-neutral toolkit to invoke a high-level API for a *type* of service.

Level 1 – REST and JSON

- Sample request:

```
/ws/IMFS/ListFolder.ashx?sessionToken=
8da051b0-a60f-4c22-a8e0-d9380edafa6f
&folderPath=/cs1&pageNumber=1
&pageSize=5
```

- Sample response:

```
{ "ResponseCode": 0, "ListFolder":
  { "TotalFolderCount": 3,
    "TotalFileCount": 3215,
    "PageFolderCount": 3,
    "PageFileCount": 2, ...}}
```

Level 1 – SOAP and XML

- Sample request:

```
<ListFolderRequest>  
  <SessionToken>  
    8da051b0-a60f-4c22-a8e0-d9380edafa6f  
  </SessionToken>  
  <FolderPath>/cs1</FolderPath>  
  <PageNumber>1</PageNumber>  
  <PageSize>5</PageSize>  
</ListFolderRequest>
```

Level 1 – SOAP and XML

- Sample response:

```
<Response>
  <ResponseCode>0</ResponseCode>
  <ListFolder>
    <TotalFolderCount>3</TotalFolderCount>
    <TotalFileCount>3215</TotalFileCount>
    <PageFolderCount>3</PageFolderCount>
    <PageFileCount>2</PageFileCount>
    <Folder>
      <FolderCount>0</FolderCount>
      <FileCount>1</FileCount>
      <Name>F8AChild</Name>
```

...

Level 2 – Language-specific

- A request to a REST service:
`file_get_contents('.../ws/IMFS/ListFolder.ashx?sessionToken=8da051b0-a60f-4c22-a8e0-...')`
- A request to a SOAP service:
`List<String, String> params =
 'FolderPath', '/cs1',
 'PageNumber', 1, blah, blah...;
soapClient.call(params);`

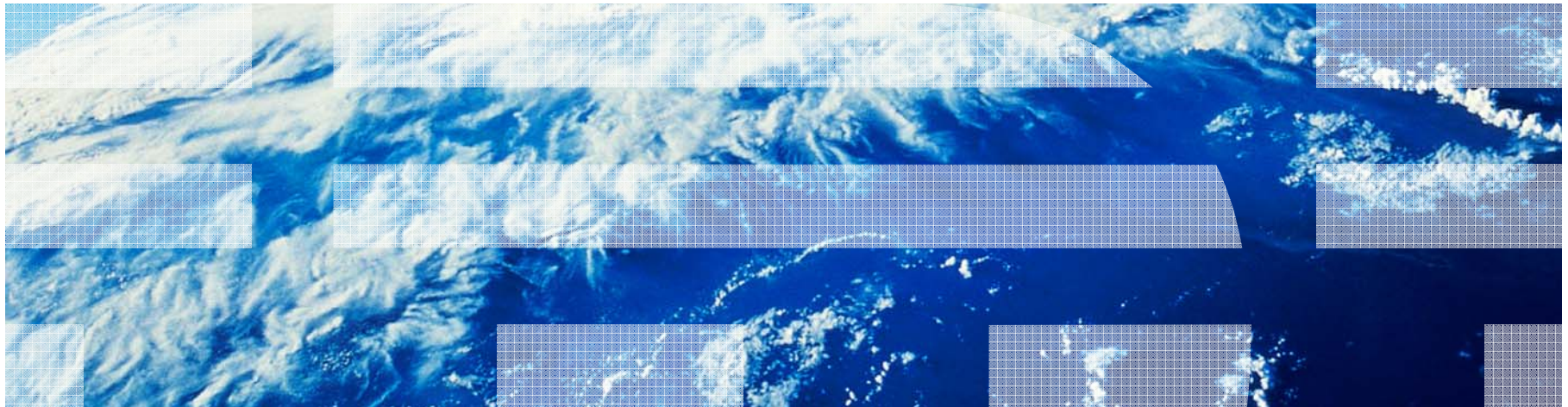
Level 3 – Service-specific

- Sample request to list the contents of an S3 bucket:
`s3.getObjectsByBucket('cs1');`
- Sample request to list the contents of a folder in Nirvanix IMFS:
`Map<String, String> options = ...;`
`imfs.listFiles(options);`
 - Passing in things like directory name, page size, page number, etc.

Level 4 – Service-neutral

- Sample request to list the contents of a folder:
`storageAdapter.listItems('cs1');`
- This works for S3, Nirvanix, etc.

The Simple Cloud API



The Simple Cloud API



- A joint effort of Zend, GoGrid, IBM, Microsoft, Nirvanix and Rackspace
 - But you can add your own libraries to support other cloud providers.
- The goal: Make it possible to write portable, interoperable code that works with multiple cloud vendors.
- There's an article on the Simple Cloud API in the developerWorks Open Source zone: **bit.ly/1bSkTx**

The Simple Cloud API

- Covers three areas:
 - File storage (S3, Nirvanix, Azure Blob Storage, Rackspace Cloud Files)
 - Document storage (SimpleDB, Azure Table Storage)
 - Simple queues (SQS, Azure Table Storage)
- Uses the Factory and Adapter design patterns
 - A configuration file tells the Factory object which adapter to create.

Dependency injection

- The Simple Cloud API uses dependency injection to do its magic.
- A sample configuration file:

```
aws.storage_adapter=S3Adapter
```

```
aws.accesskey=338ab839-ac72870a
```

```
aws.secretkey=abnT3xeks1Uw9W7OdH...MtHOSSd
```

```
aws.remote_directory=open-cloud-demo
```

Dependency injection

- A different configuration file:

```
nirvanix.storage_adapter=NirvanixAdapter
```

```
nirvanix.appName=JavaOne
```

```
nirvanix.appKey=533a2...79ef10
```

```
nirvanix.username=larry_e
```

```
nirvanix.password=PD3x7Js/
```

The Simple Cloud Storage API

- Putting an item into a Nirvanix directory or an S3 bucket:

```
StorageAdapter cloudAdapter.storeItem(  
    remote_name, localStream, options);
```

- These lines of code work with Nirvanix and S3 (and others, coming soon)
 - Which adapter is created and which storage service is used depends on the configuration file.

Methods

- The storage API supports several common operations:
 - `storeItem()`, `fetchItem()` and `deleteItem()`
 - `copyItem()`, `moveItem()` and `renameItem()`
 - `listFolders()` and `listItems()`
 - `storeMetadata()`, `fetchMetadata()` and `deleteMetadata()`
- Not all of these are supported natively.
 - More on this in a minute.

Issues

- Not all storage services support renaming files.
 - You can hack this, but....
- Not all storage services support listing containers.
- What's the best way to handle this?
 - Introspection?
 - `instanceof`?
 - XSLT style? `system-property`
(`'sc:supports-rename'`)
- **We need your input!**

The Simple Cloud Queue API

- The queue API supports message queueing services from Amazon and Azure.
 - Although you're free to implement your own adapter.
- Supported methods:
 - `createQueue()`, `deleteQueue()` and `listQueues()`
 - `sendMessage()`, `receiveMessages()` and `deleteMessage()`
 - `fetchQueueMetadata()` and `storeQueueMetadata()`

Issues

- How many messages are in a queue?
 - SQS lets you ask, Azure doesn't.
- Can I peek a message?
 - Azure lets you peek, SQS doesn't.

The Simple Cloud Document API

- Supports basic database services such as Amazon's SimpleDB and Azure Table Services.
- Supported methods:
 - `createCollection()`, `deleteCollection()` and `listCollections()`
 - `insertDocument()`, `replaceDocument()`, `updateDocument()`, `deleteDocument()` and `fetchDocument()`
 - `query()` and `select()`

The really big issue

- The query languages and database functions for cloud database services are wildly divergent.
 - Some are relational, most are not
 - Some support schemas, most do not
 - Some support concurrency, most do not

The demo

- The order form pulls data from the Products database.
- Clicking “Place my order” creates a record in the Orders database and a message in a queue.

Developing Applications in the Cloud using the Simple Cloud API - Mozilla Firefox

http://localhost:8080/products/OrderForm

IBM Software **Impact2011**

DougCo, Inc.
"Bad Products for Good People."

Order Form

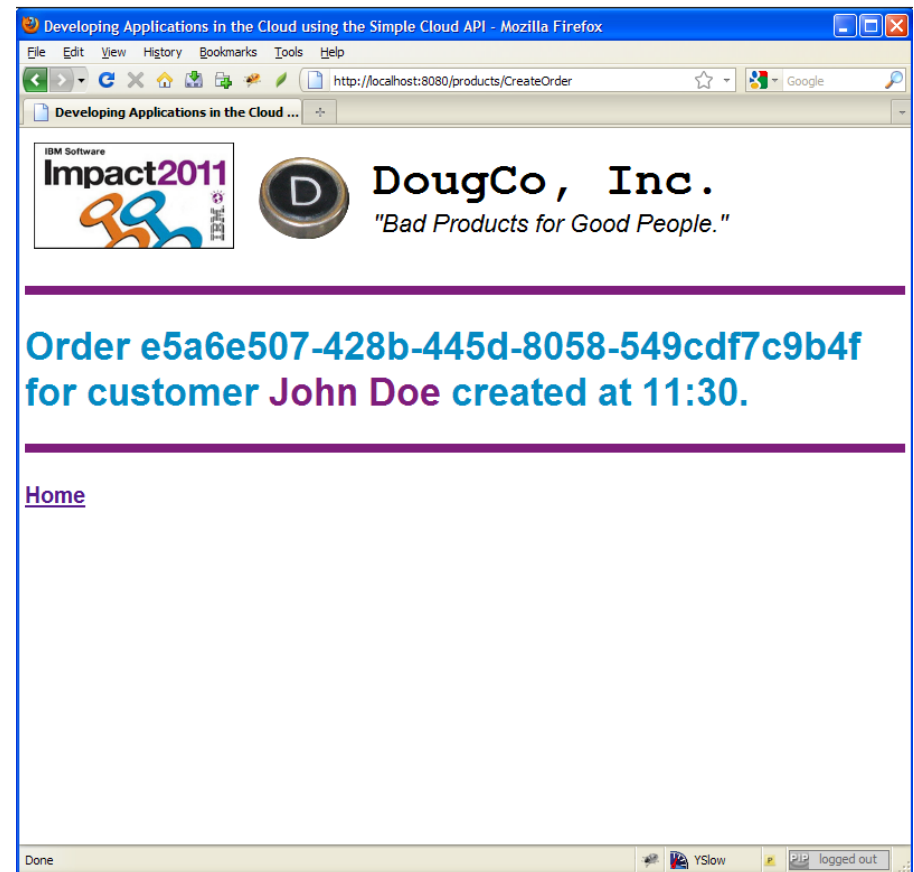
Your name:

Part No.	Description	Price	How Many Ya Want?
IM-003	Red Widget	4.95	<input type="text" value="0"/>
IM-005	Blue Widget	12.00	<input type="text" value="4"/>
IM-001	Green Widget	19.95	<input type="text" value="0"/>
IM-002	Black Widget	49.95	<input type="text" value="0"/>
IM-004	Orange Widget	7.00	<input type="text" value="0"/>

Done YSlow logged out

The demo

- An order has been created in the database and a message put in the queue.
- The unique ID must be generated by the code.



The demo

- Displaying the Orders database includes the record we just created.

Developing Applications in the Cloud using the Simple Cloud API - Mozilla Firefox

http://localhost:8080/products/DisplayOrdersDatabase

IBM Software **Impact2011**

DougCo, Inc.
"Bad Products for Good People."

Displaying the CloudyOrders Database

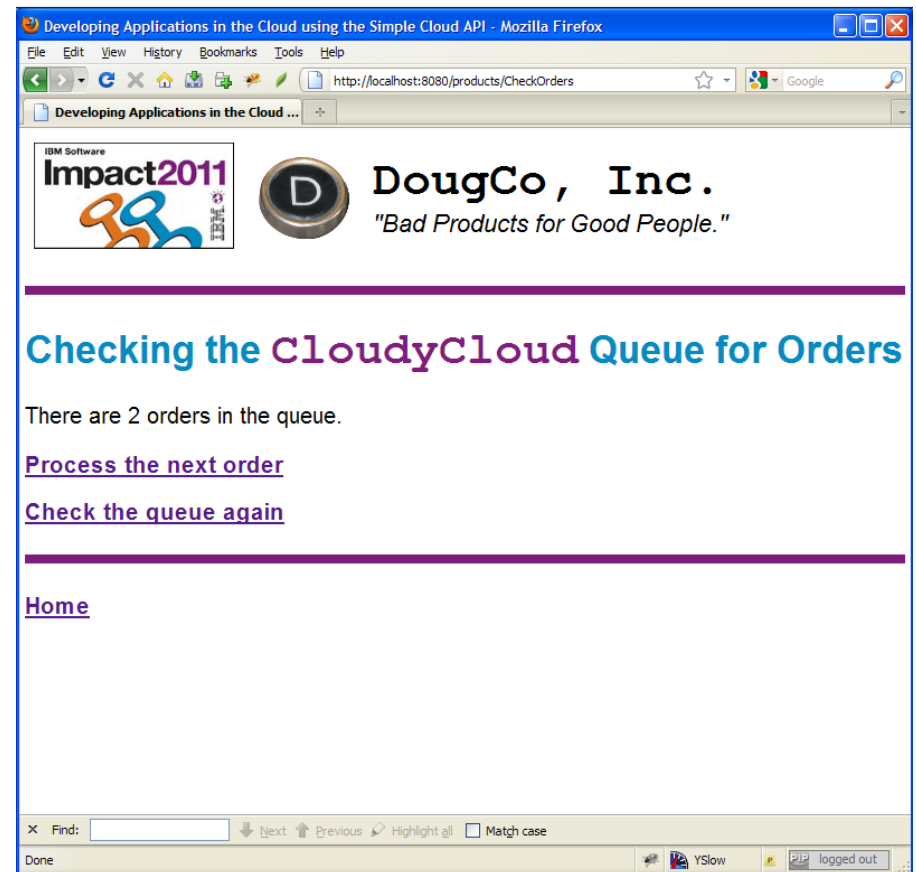
Order #	Customer	Qty.	Part No.
a0be398f-2d76-4b8f-b6c5-590c3d3937c6	Xavier McDaniel	4	IM-193
e5a6e507-428b-445d-8058-549cdf7c9b4f	John Doe	4	IM-005

[Home](#)

Done YSlow logged out

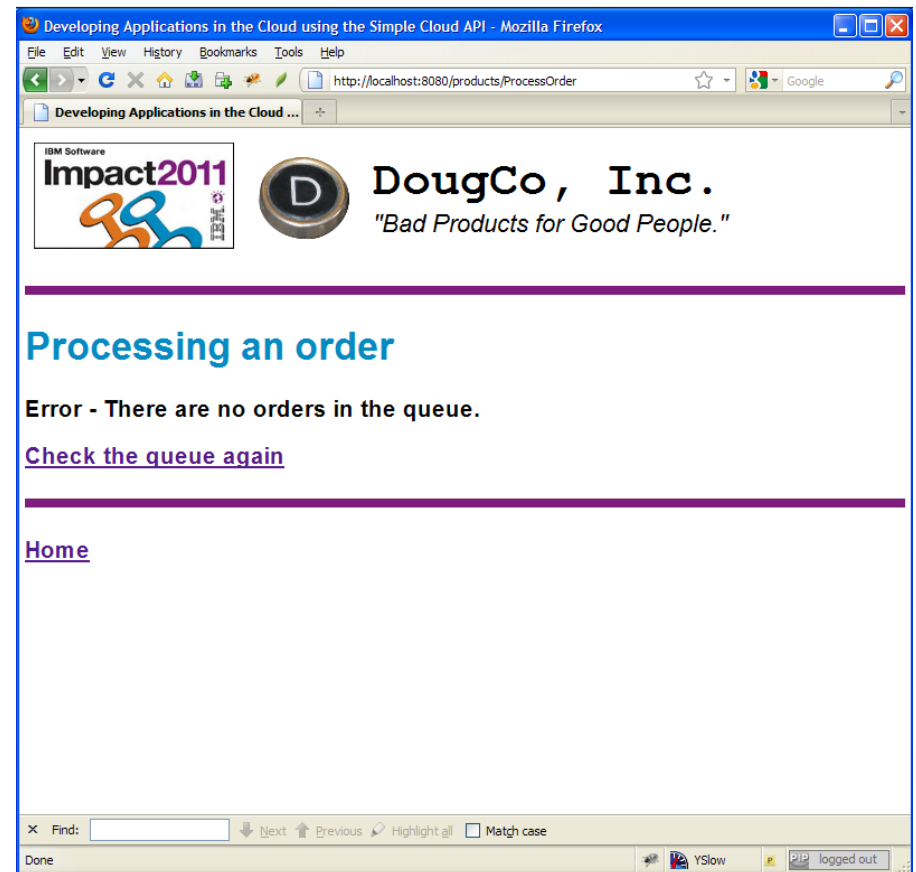
The demo

- Checking the queue for orders, the service tells us there are two messages in the queue.
- The number of messages is approximate.



The demo

- Unfortunately, the number of messages in the queue is often out of sync with the actual queue.
- Propagation latency is the culprit here.



The demo

- Eventually the message is received from the queue. The code retrieves the necessary data from the Orders and Products databases and creates an invoice.

Developing Applications in the Cloud using the Simple Cloud API - Mozilla Firefox

http://localhost:8080/products/ProcessOrder

IBM Software **Impact2011**

DougCo, Inc.
"Bad Products for Good People."

Processing an order

Order e5a6e507-428b-445d-8058-549cdf7c9b4f
for **John Doe**

Part No.	Description	Qty.	Each	Total
IM-005	Blue Widget	4	12.00	48.00
			Shipping and handling	9.00
			Taxes	3.84
			Total	\$60.84

Save Invoice

Home

Done YSlow logged out

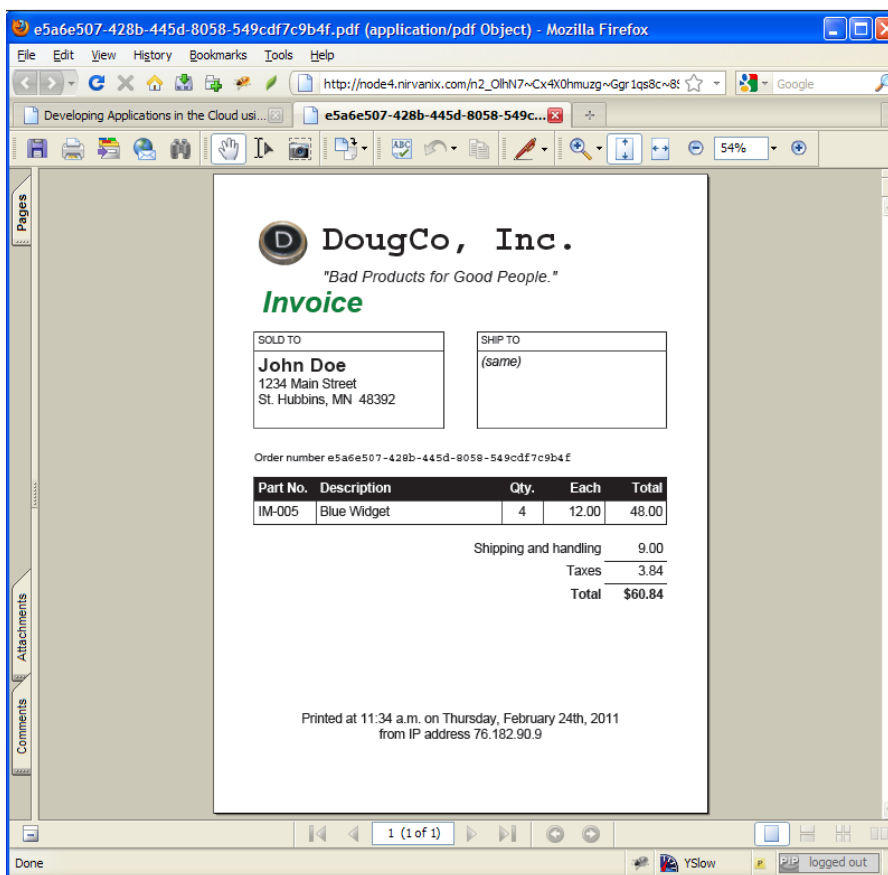
The demo

- At this point the queue has been updated, a PDF file of the invoice has been generated, and that PDF has been stored in cloud storage at Amazon and Nirvanix.



The demo

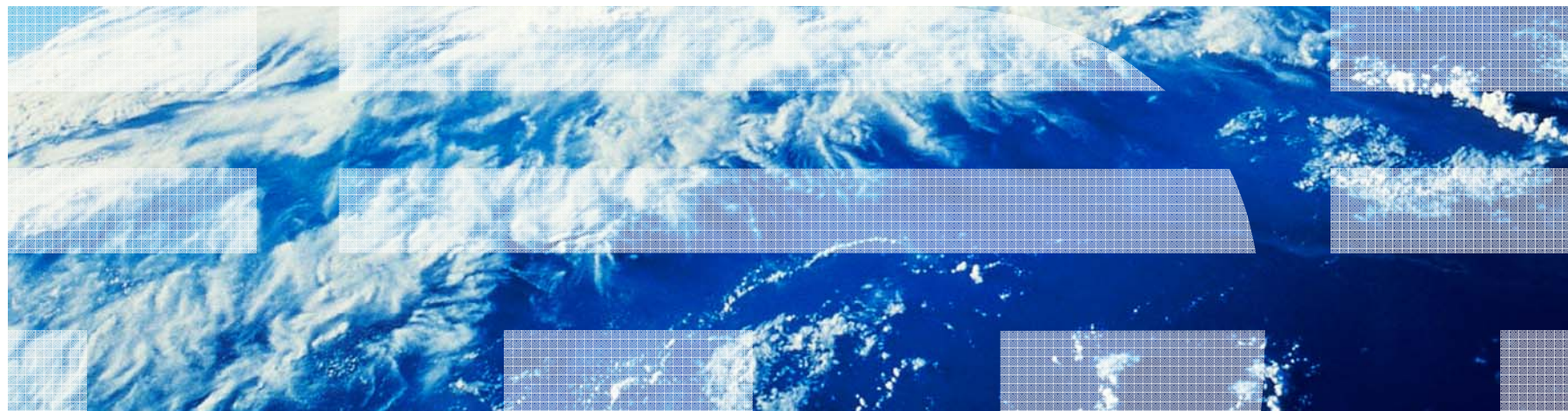
- The invoice can be viewed from either cloud storage provider.



Openness in action

- IBM has contributed heavily to the Java implementations of libcloud:
 - <https://svn.apache.org/repos/asf/incubator/libcloud/sandbox/java/trunk/>
- The Java implementation includes the basic framework plus adapters for the IBM Smart Business Cloud, Amazon and Rackspace.
- Simple Cloud Storage adapters for S3 and Nirvanix are out now, more are on their way....

Controlling VMs with Apache libcloud



Apache libcloud

- A common library for controlling VMs in the cloud
 - Create, destroy, reboot and list instances, list and start images
- incubator.apache.org/libcloud

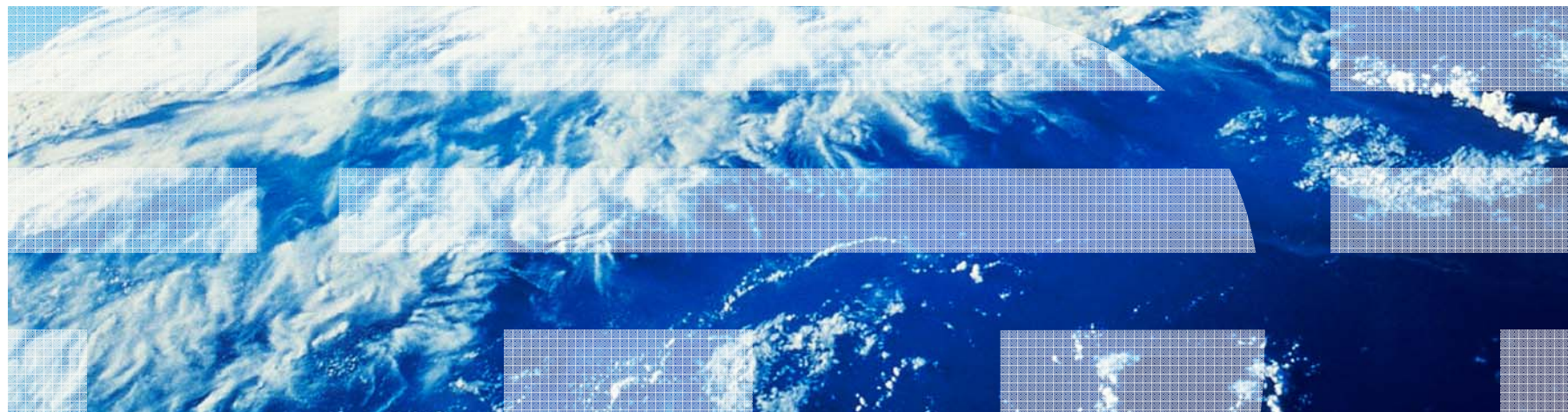
Apache libcloud

- libcloud currently supports a couple dozen cloud providers.
- Most of the adapters support all of the functions in the libcloud API.

current support

provider	list	reboot	create	destroy	images	sizes
Dreamhost	yes	yes	yes	yes	yes	yes
EC2-AP Southeast	yes	yes	yes	yes	yes	yes
EC2-US East	yes	yes	yes	yes	yes	yes
EC2-US West	yes	yes	yes	yes	yes	yes
EC2-EU West	yes	yes	yes	yes	yes	yes
enomaly ECP	yes	yes	yes	yes	yes	yes
Eucalyptus	no	yes	yes	yes	yes	yes
flexiscale	no	no	no	no	no	no
GoGrid	yes	yes	yes	yes	yes	yes
Hosting.com	no	no	no	no	no	no
IBM Cloud	yes	yes	yes	yes	yes	yes
Linode	yes	yes	yes	yes	yes	yes
OpenNebula	yes	yes	yes	yes	yes	yes
Rackspace	yes	yes	yes	yes	yes	yes
RimuHosting	yes	yes	yes	yes	yes	yes
Slicehost	yes	yes	yes	yes	yes	yes
SoftLayer	yes	yes	yes	yes	yes	yes
Terremark	yes	yes	yes	yes	yes	yes
vCloud	yes	yes	yes	yes	yes	yes
Voxel	yes	yes	yes	yes	yes	yes
VPS.net	yes	yes	yes	yes	yes	yes

Let's look at some code!



Apache libcloud

- Initialize the driver for Amazon EC2:

```
Class<NodeDriver> ec2USEastClass  
DriverFactory.getDriver(  
    Provider.EC2_US_EAST);  
EC2USEastDriver ec2USEastDriver =  
    (EC2USEastDriver)DriverFactory.  
    constructDriver(ec2USEastClass,  
        "33839-ac72870a", "abnT3xUw...5");
```

Apache libcloud

- Initialize the driver for Rackspace:

```
Class<NodeDriver> rackspaceClass =  
    DriverFactory.getDriver(  
        Provider.RACKSPACE);  
RackspaceDriver rackspaceDriver =  
    (RackspaceDriver)DriverFactory.  
    constructDriver(rackspaceClass,  
        "larry_e", "b237d94...8f8a");
```

Apache libcloud

```
List<NodeDriver> drivers = new
    ArrayList<NodeDriver>();
drivers.add(ec2USEastDriver);
drivers.add(rackspaceDriver);
for (NodeDriver driver : drivers) {
    List<INode> nodes =
        driver.listNodes();
    ...
}
```

The libcloud interface

- `driver.getName()`
- `driver.listImages()`
- `driver.listLocations()`
- `driver.listNodes()`
- `driver.listSizes()`
- `node.getId()`
- `node.getName()`
- `node.getPrivateIp()`
- `node.getPublicIp()`
- `node.getUuid()`
- `node.getState()`

The demo

- We'll use some code that lists all of the instances we have running at various cloud providers.
 - For each instance, we can terminate or reboot it.
- One piece of code lets us do this for both providers.
 - You can also list all of the images at a provider and start any of the ones you have access to.

The demo

- This display shows all the running VMs in the IBM Dev/Test cloud, Amazon, and Rackspace.
- The current status of the VMs is displayed.
- Machines can be rebooted or shut down.

Apache libcloud demo

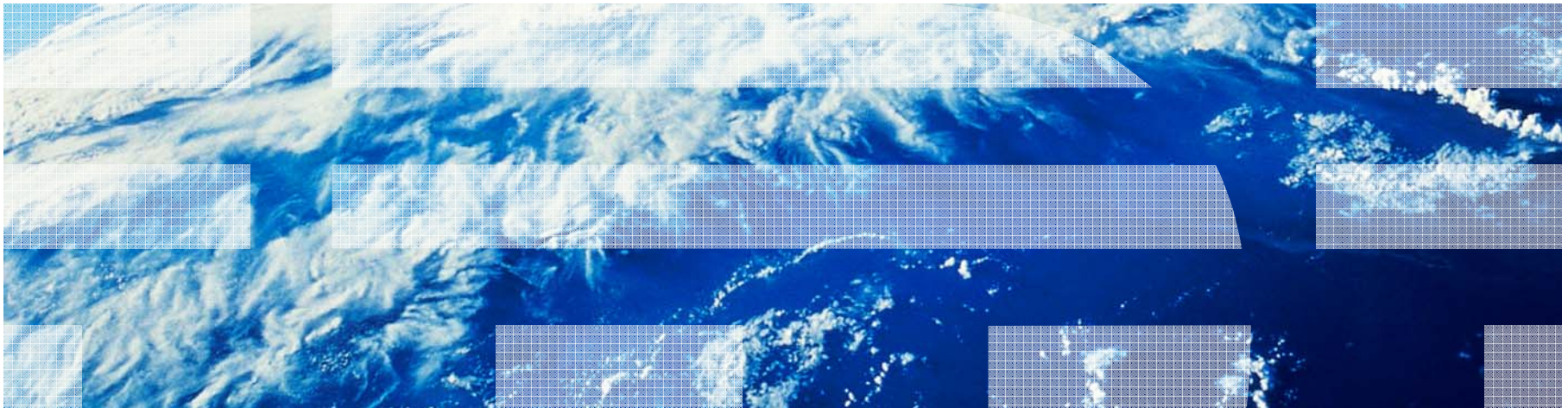
Virtual Machines

IBM Smart Business Development and Test Cloud:					
State	Provider	Node Name	Public IP		
RUNNING	IBM Smart Business Development and Test Cloud	EricTest2	170.224.160.165	Reboot	Shut down
RUNNING	IBM Smart Business Development and Test Cloud	EricTest1	170.224.161.70	Reboot	Shut down
RUNNING	IBM Smart Business Development and Test Cloud	Win2008	170.224.165.59	Reboot	Shut down
Amazon EC2 (us-east-1):					
State	Provider	Node Name	Public IP		
RUNNING	Amazon EC2 (us-east-1)	i-27a1f84b	ec2-50-17-43-210.compute-1.amazonaws.com	Reboot	Shut down
PENDING	Amazon EC2 (us-east-1)	i-1da6ff71		Reboot	Shut down
Rackspace Cloud:					
State	Provider	Node Name	Public IP		
RUNNING	Rackspace Cloud	Impact	50.56.74.225	Reboot	Shut down

Find: Next Previous Highlight all Match case

Done YSlow logged out

Summary / Resources / Next steps



Get Involved!

- Download the code, build a prototype, submit requirements / new adapters / bug reports
 - <https://svn.apache.org/repos/asf/incubator/libcloud/sandbox/java/trunk/>

cloudusecases.org

- The Cloud Computing Use Cases group is focused on documenting customer requirements.
- Covers Security, SLAs, developer requirements and cloud basics.
- The group is currently working on a paper entitled “Moving to the Cloud.”
- **Join us!**
 - **<http://linkd.in/hni8c5>**

Version 4 – Includes customer scenarios, security, SLAs and developer requirements



Where we're headed

- **<hype>**

Cloud computing will be the biggest change to IT since the rise of the Web.

- **</hype>**

- But to make the most of it, we have to keep things open.
- And everybody has to get involved to make that happen.



Thanks!

