Open Cloud Computing with the Simple Cloud API and Apache libcloud

Doug Tidwell
Cloud Computing Evangelist, IBM
dtidwell@us.ibm.com

Session 7665
Agenda

- Portability and interoperability
- A few words about APIs
- The Simple Cloud API
  - Storage
  - Queues
  - Documents
- Controlling VMs with Apache libcloud
- 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.
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:

```xml
<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:
  
  ```xml
  <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>
      </Folder>
      ...
  </ListFolder>
  </Response>
  ```
Level 2 – Language-specific

- A PHP request to a REST service:
  - `file_get_contents('.../ws/IMFS/ListFolder.ashx?sessionToken=8da051b0-a60f-4c22-a8e0-...')`
- A PHP request to a SOAP service:
  - `$param = array(...,
                  'FolderPath' => '/cs1',
                  'PageNumber' => 1, ...);
  - `$soapClient->call('listFolder',
                        $param, $namespace);`
Level 3 – Service-specific

- Sample PHP request to list the contents of an S3 bucket:
  - `$s3-> getObjectsByBucket('cs1');`
- Sample PHP request to list the contents of a folder in Nirvanix IMFS:
  - `$imfs->listFolder
      (array ('folderPath' => '/cs1',
      'pageNumber' => 1,
      'pageSize' => 5));`
Level 4 – Service-neutral

- Sample PHP request to list the contents of a folder:
  - `$storage->listItems('cs1');`
- This works for S3, Nirvanix, GoGrid, etc.
The Simple Cloud API

simplecloud.org
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:
  ```
  storage_adapter = "Zend_Cloud_StorageService_Adapter_Nirvanix"
  auth_accesskey = "338ab839-ac72870a"
  auth_username = "skippy"
  auth_password = "/p@$w0rd"
  remote_directory = "/dougtidwell"
  ```
Dependency injection

- A different configuration file:

```php
storage_adapter = "Zend_Cloud_StorageService_Adapter_S3"
aws_accesskey = "ac72870a-338ab839"
aws_secretkey = "/par$w3rd"
bucket_name = "dougtidwell"
```
Vendor-specific APIs

• Listing all the items in a Nirvanix directory:
  ```php
  $auth = array('username' => 'your-username',
                 'password' => 'your-password',
                 'appKey'   => 'your-appkey');
  $nirvanix = new Zend_Service_Nirvanix($auth);
  $imfs = $nirvanix->getService('IMFS');
  $args = array('folderPath' => '/dougtidwell',
                 'pageNumber' => 1,
                 'pageSize'   => 5);
  $stuff = $imfs->ListFolder($args);
  ```

• All of these lines of code are specific to Nirvanix.
Vendor-specific APIs

• Listing all the items in an S3 bucket:
  
  ```php
  $s3 = new Zend_Service_Amazon_S3
       ($accessKey, $secretKey);
  $stuff = $s3->getObjectsByBucket($bucketName);
  ```

• All of these lines of code are specific to S3.
The Simple Cloud **Storage API**

• Listing all the items in a Nirvanix directory or S3 bucket:
  
  ```php
  $credentials =
    new Zend_Config_Ini($configFile);
  $stuff = Zend_Cloud_StorageService_Factory
    ::getAdapter($credentials)->listItems();
  ```

• These lines of code work with Nirvanix and S3 (and Rackspace, etc.).
  - 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.
Demo time!

• We’ll look at a file browser built on the Simple Cloud storage API.
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()`
Demo time!

- We’ll look at a message queue monitor built with the Simple Cloud queue API.
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()`
  • `query()` and `select()`
Issues

• 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
Writing your own adapter

• To write your own adapter, you have to implement all of the methods of the particular interface.
  • StorageService/Adapter, QueueService/Adapter, etc.
• If the cloud vendor you’re targeting already has a library (a Level 3 API) for the service, you’re 80% there:
  ```php
  public function listFolders($path = null, $options = null) {
    return $this->_connection->list_containers();
  }
  ```
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
libcloud currently supports a couple dozen cloud providers. Most of the adapters support all of the functions in the libcloud API.
Let’s look at some code!
Apache libcloud

• Find all the VMs I have running in the Amazon and Rackspace clouds:

```python
def _get_drivers(self):
    EC2 = get_driver(Provider.EC2_US_EAST)
    Rackspace = get_driver(Provider.RACKSPACE)
    return [Rackspace(secrets.RACKSPACE_ACCESS_ID,
                      secrets.RACKSPACE_SECRET_KEY),
            EC2(secrets.EC2_ACCESS_ID,
                secrets.EC2_SECRET_KEY)]
```
The libcloud interface

- `list_images()`
- `create_node()`
- `reboot_node()`
- `destroy_node()`
- `list_nodes()`
- `list_sizes()`
- `list_locations()`
- `get_uuid()`
Demo

• We’ll use some Python code that lists all of the images and instances we have running at Rackspace.
  • For each image, we can start a new instance.
  • For each instance, we can terminate or reboot it.
• Then we’ll run the code again, listing the Amazon images and instances as well.
Openness in action

- IBM has contributed a Java implementation of libcloud:
- The Java implementation includes the basic framework plus an adapter for the IBM Smart Business Cloud.
  - Other adapters are underway...
Summary / Resources / Next steps
Get Involved!

• Simple Cloud API
  • Download the code, build a prototype, submit requirements / new adapters / bug reports
  • simplecloud.org

• libcloud
  • incubator.apache.org/libcloud
cloudusecases.org

• The Cloud Computing Use Cases group is focused on documenting customer requirements.
• Covers Security, SLAs, developer requirements and cloud basics.
• Join us!
Also available in Chinese
Also available in Japanese

- Chinese discussion group on LinkedIn:
  - linkedin.com/groups?gid=2919533&trk=myg_ugrp_ovr
- Japanese discussion group and translated paper coming soon!
developerWorks cloud zone

• Dozens of articles on cloud computing, including introductions, code samples, tutorials and podcasts.
• ibm.com/developerworks/cloud
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!

Doug Tidwell
Cloud Computing Evangelist
dtidwell@us.ibm.com

This is session 7665.