

# Keeping Your Options Open, Even if The Cloud is Not

Doug Tidwell

Cloud Computing Evangelist, IBM

[dtidwell@us.ibm.com](mailto:dtidwell@us.ibm.com)

# Agenda

- Portability and interoperability
- How standards work
- A few words about APIs
- The Simple Cloud API
- 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.

# 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 environment
  - 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 cloud providers at the same time.

# How standards work

# 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.

# Principles of openness

1. Cloud providers must work together to ensure that the challenges to cloud adoption are addressed through **open collaboration and the appropriate use of standards.**
2. Cloud providers must not use their market position to lock customers into their particular platforms and limiting their choice of providers.

# Principles of openness

- 3. Cloud providers must use and adopt existing standards wherever appropriate.**  
The IT industry has invested heavily in existing standards and standards organizations; there is no need to duplicate or reinvent them.
- 4. When new standards (or adjustments to existing standards) are needed, we must be judicious and pragmatic to avoid creating too many standards. We must ensure that standards promote innovation and do not inhibit it.**

# Principles of openness

- 5. Any community effort around the open cloud should be driven by customer needs,** not merely the technical needs of cloud providers, and should be tested or verified against real customer requirements.
- 6. Cloud computing standards organizations,** advocacy groups, and communities should work together and stay coordinated, making sure that efforts do not conflict or overlap.

# A few words about APIs

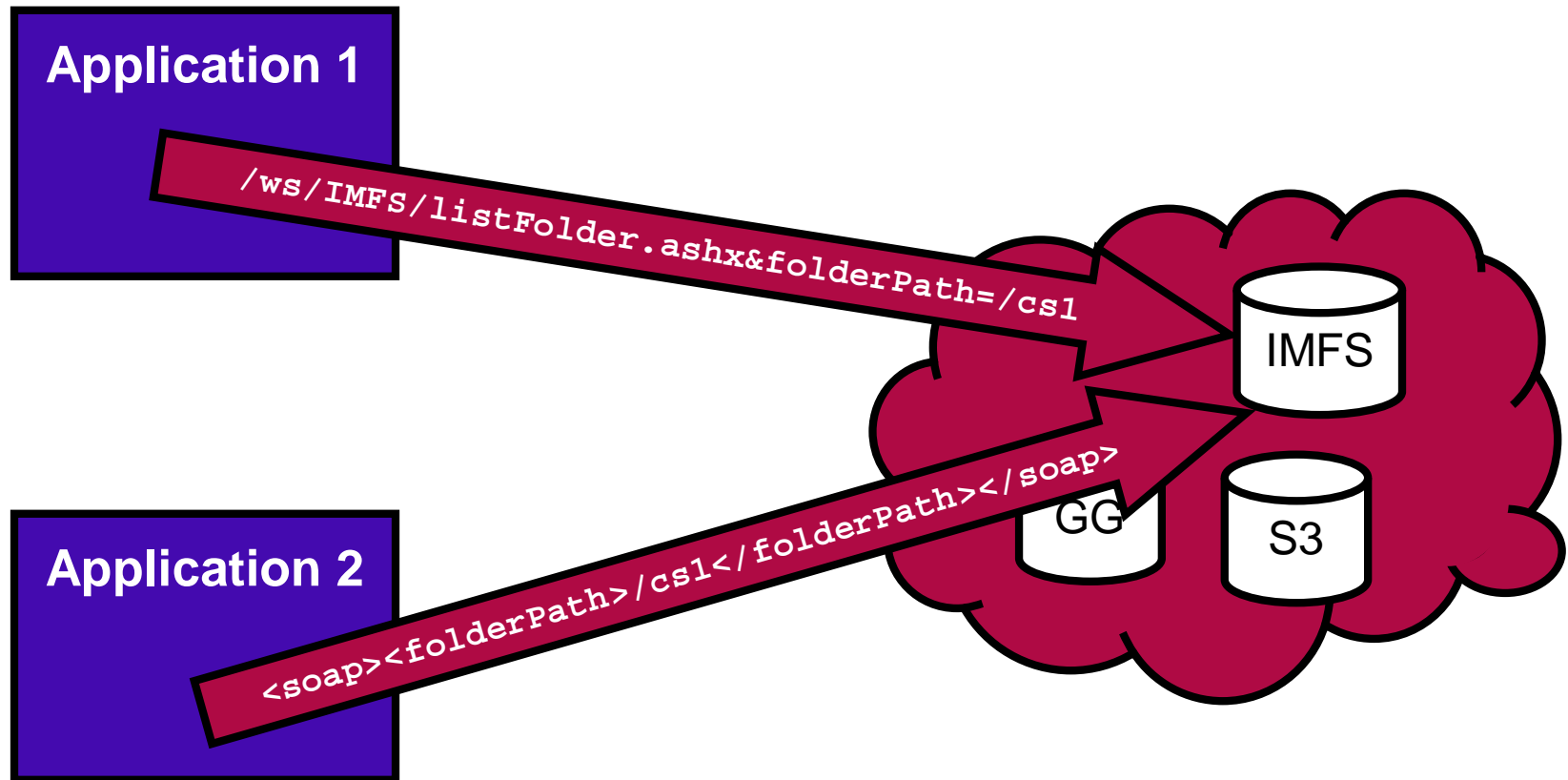
# Levels of APIs

- 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 – The Wire

- Developers write directly to the REST or SOAP API.
  - Developers must deal with all the details of the request, including URL formats, XML parsing, and HTTP headers and response codes.
  - Writing code at this level is rarely done today in the SOAP world. REST services (simple ones in particular) are still invoked this way.

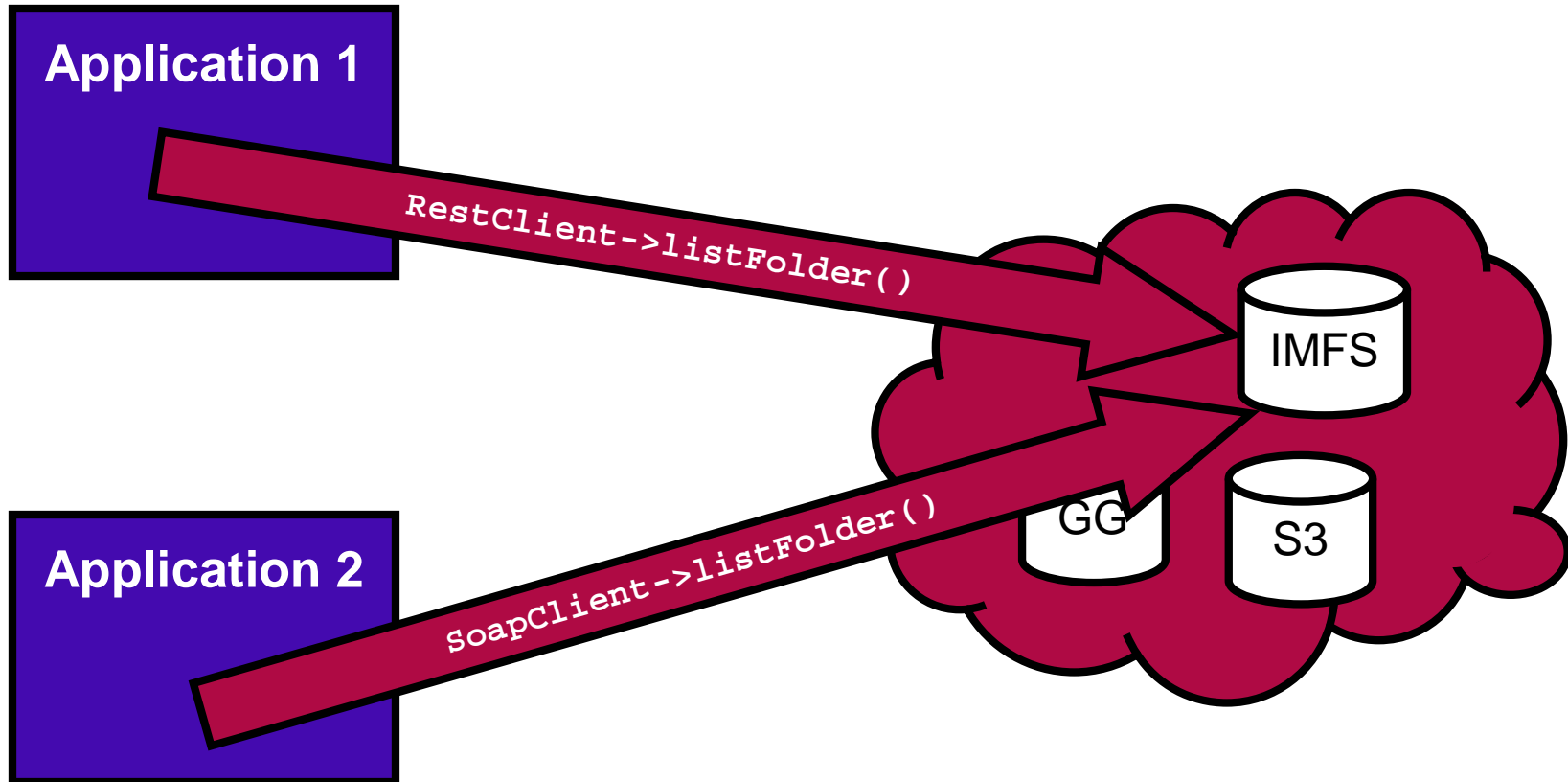
## Level 1



## Level 2 – Language-specific

- Developers use a language-specific toolkit to build the REST or SOAP requests.
  - Slightly higher-level than writing directly to the REST or SOAP APIs.
  - Details such as managing HTTP error codes or XML parsing are handled by the toolkit.

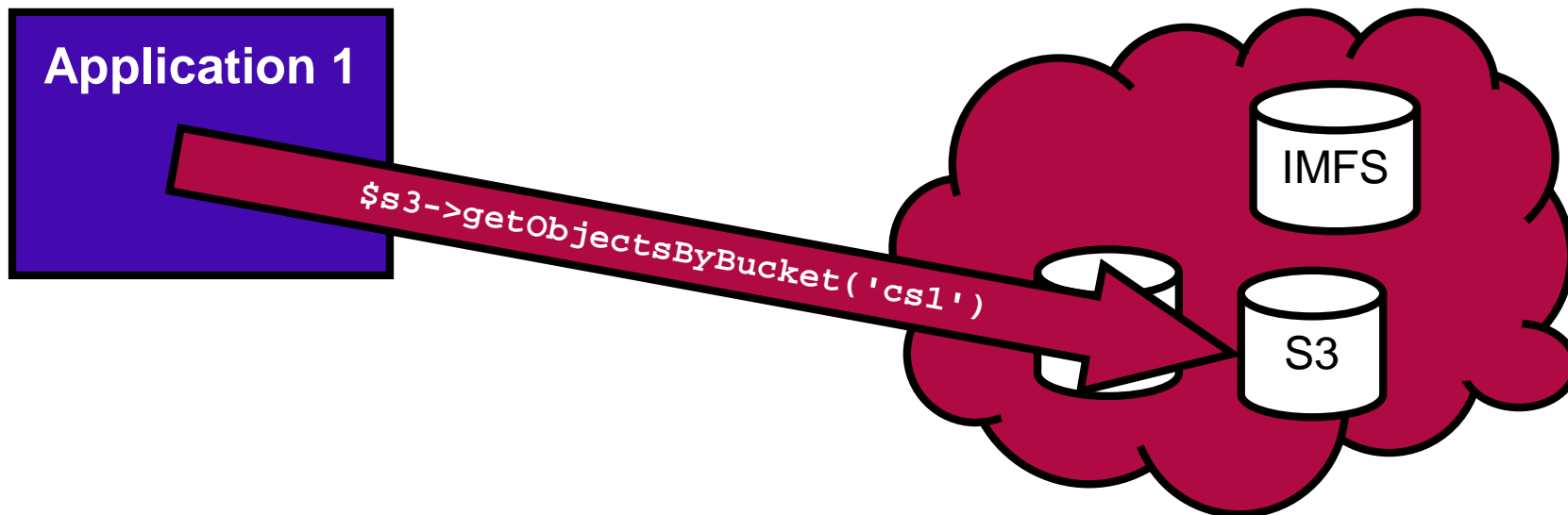
## Level 2



# Level 3 – Service-specific

- Developers use objects that wrapper a particular service.
  - Developers don't know if they're using REST or SOAP.
  - Developers focus on using a particular service to get something done.

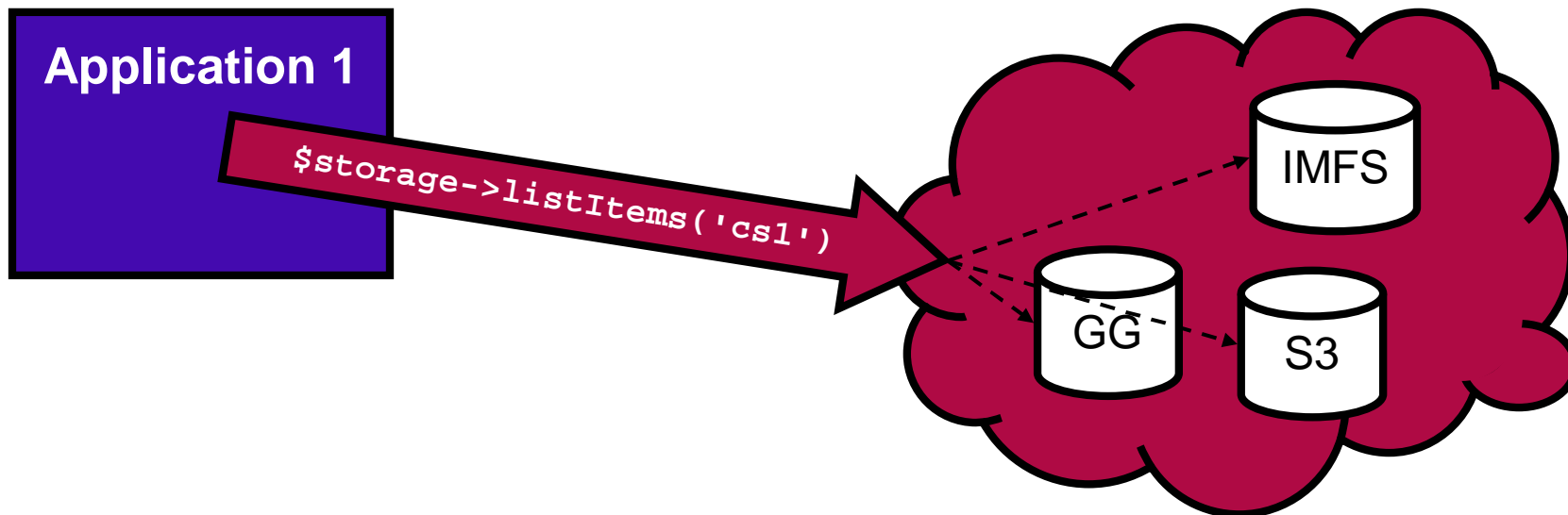
# Level 3



# Level 4 – Service-neutral

- Developers use objects that wrapper a particular *type* of service.
  - Developers have no idea which service they're using.
  - **Developers focus on getting something done.**

# Level 4



# The Simple Cloud API



[simplecloud.org](http://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](http://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.

# Vendor-specific APIs

- Listing all the items in a Nirvanix directory:

```
$auth = array('username' => 'your-username',  
             'password' => 'your-password',  
             'appKey'   => 'your-appkey');  
$nirvanix = new Zend_Service_Nirvanix($auth);  
$imfs = $nirvanix->getService('IMFS');  
$args = array('folderPath' => '/dougtdwell',  
             '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:

```
$s3 = new Zend_Service_Amazon_S3  
    ($accessKey, $secretKey);  
$stuff = $s3->getObjectsByBucket($bucketName);
```

- All of these lines of code are specific to S3.

# The Simple Cloud API

- Listing all the items in a Nirvanix directory or S3 bucket:

```
$credentials =  
    new Zend_Config_Ini($configFile);  
$stuff = Zend_Cloud_Storage_Factory::getAdapter  
    ($credentials)->listItems();
```

- These lines of code work with Nirvanix and S3.
  - Which adapter is created and which storage is used is defined in the configuration file.

# Dependency injection

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

```
storage_adapter =  
    "Zend_Cloud_Storage_Adapter_Nirvanix"  
auth_accesskey = "338ab839-ac72870a"  
auth_username = "skippy"  
auth_password = "/p@$w0rd"  
remote_directory = "/dougtdwell"
```

# Exciting demonstration!

- **Prepare to be astounded** by the Simple Cloud API in action!
  - Due to cost constraints, we are unable to provide tissues for those moved to tears by the demonstration.
  - Persons prone to hyperventilation or motion sickness are advised to look away.
    - Be advised the management cannot be held responsible for your medical expenses.

# 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](http://incubator.apache.org/libcloud)

# Apache libcloud

- Find all the VMs I have running in the IBM, Slicehost and Rackspace clouds:

```
IBM = get_driver(Provider.IBM)
Slicehost = get_driver(Provider.SLICEHOST)
Rackspace = get_driver(Provider.RACKSPACE)
drivers =
    [ IBM('access key id', 'secret key'),
      Slicehost('api key'),
      Rackspace('username', 'api key') ]
# Now do what you like with your running VMs
```

# The libcloud interface

- `list_images()`
- `list_sizes()`
- `list_locations()`
- `create_node()`
- `list_nodes()`
- `reboot_node()`
- Other calls for querying UUIDs, locations, setting passwords, etc.

# Openness in action

- IBM will be contributing Java implementations of Simple Cloud and libcloud over the coming weeks and months.
- Stay tuned....

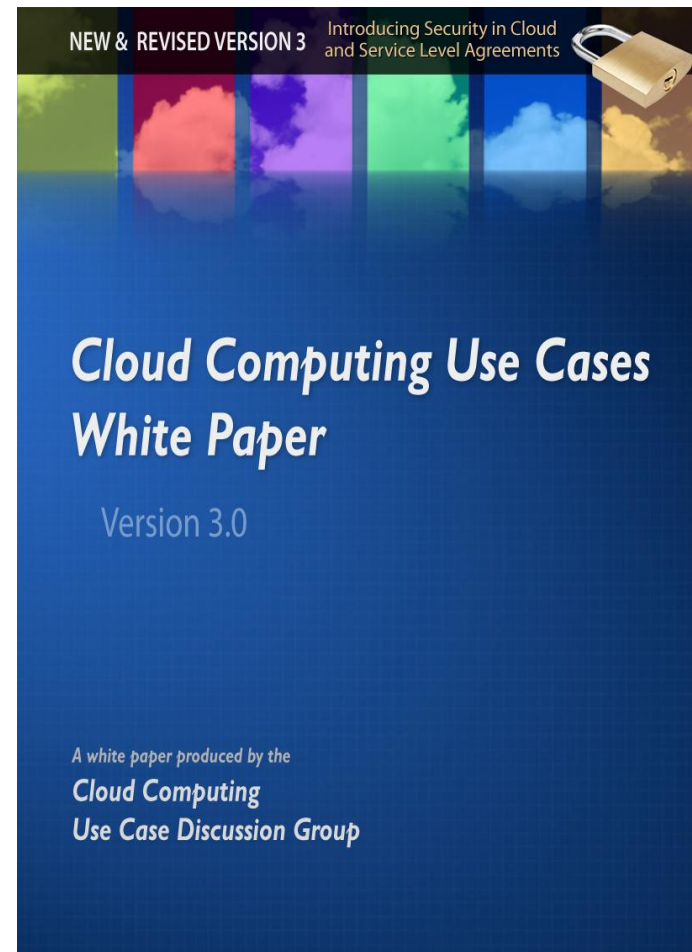
# Summary / Resources / Next steps

# Get Involved!

- Simple Cloud API
  - Download the code, build a prototype, submit requirements / new adapters / bug reports
  - [simplecloud.org](http://simplecloud.org)
- libcloud
  - [incubator.apache.org/libcloud](http://incubator.apache.org/libcloud)

# cloudusecases.org

- The Cloud Computing Use Cases group is focused on documenting customer requirements.
- Version 3 of the paper focused on security. V4 focuses on SLAs.
- **Join us!**



# Also available in Chinese



# developerWorks cloud zone



- Dozens of articles on cloud computing, including introductions, code samples, tutorials and podcasts.
- **[ibm.com/developerworks/cloud](http://ibm.com/developerworks/cloud)**

# Summary

- `<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, IBM

[dtidwell@us.ibm.com](mailto:dtidwell@us.ibm.com)