Cloud computing with PHP on Windows Azure

Rob Allen

Rob Allen

- PHP developer since 1999
- Zend Framework contributor
- Technical director of a web development company
- Wrote *Zend Framework in Action* - Buy my book!

This talk has 2 halves:

- What is Windows Azure?
- What does the PHP interface look like?

What is Windows Azure?

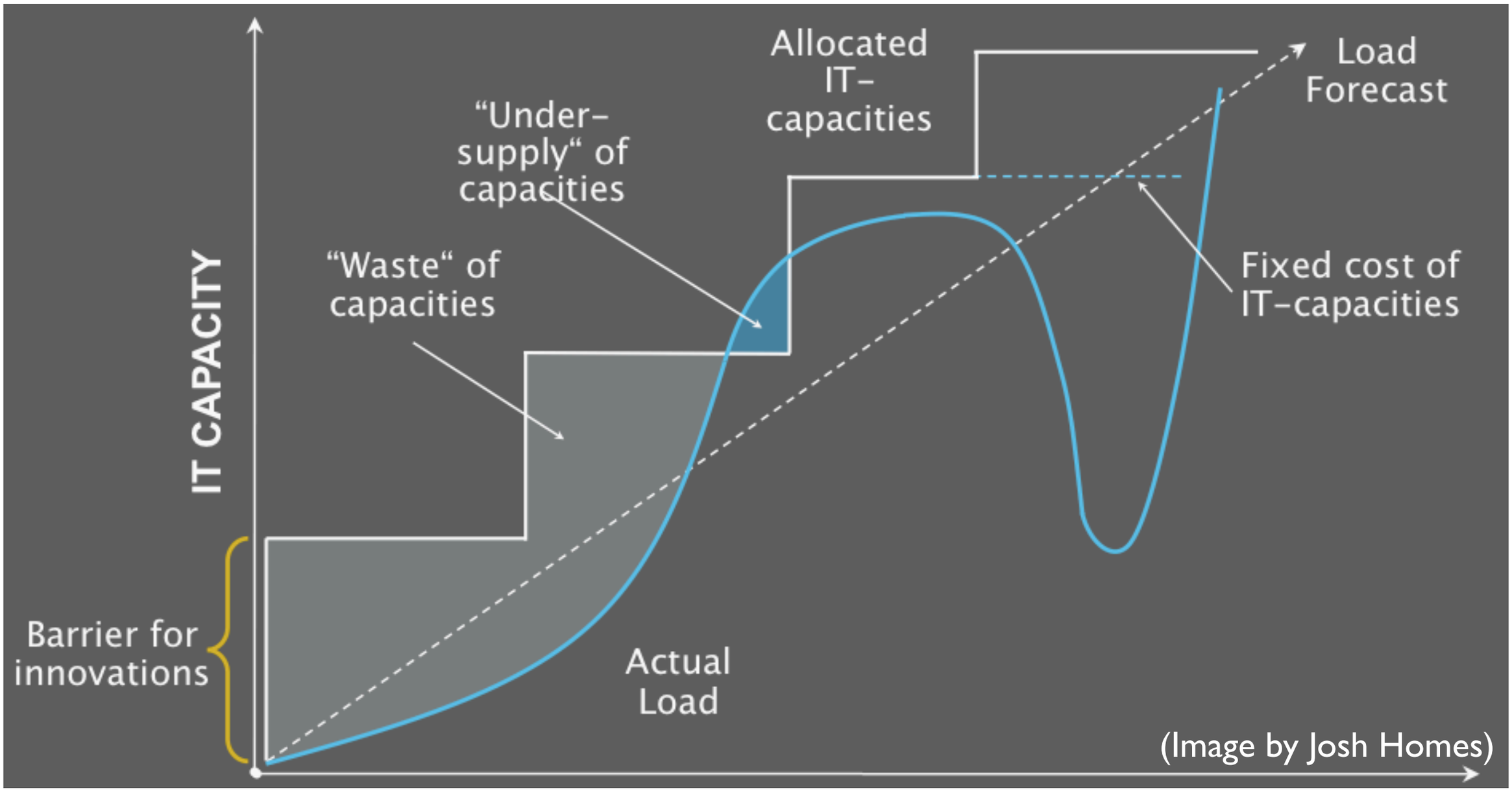
Cloud computing

"The Windows Azure platform offers a flexible, familiar environment for developers to create cloud applications and services."

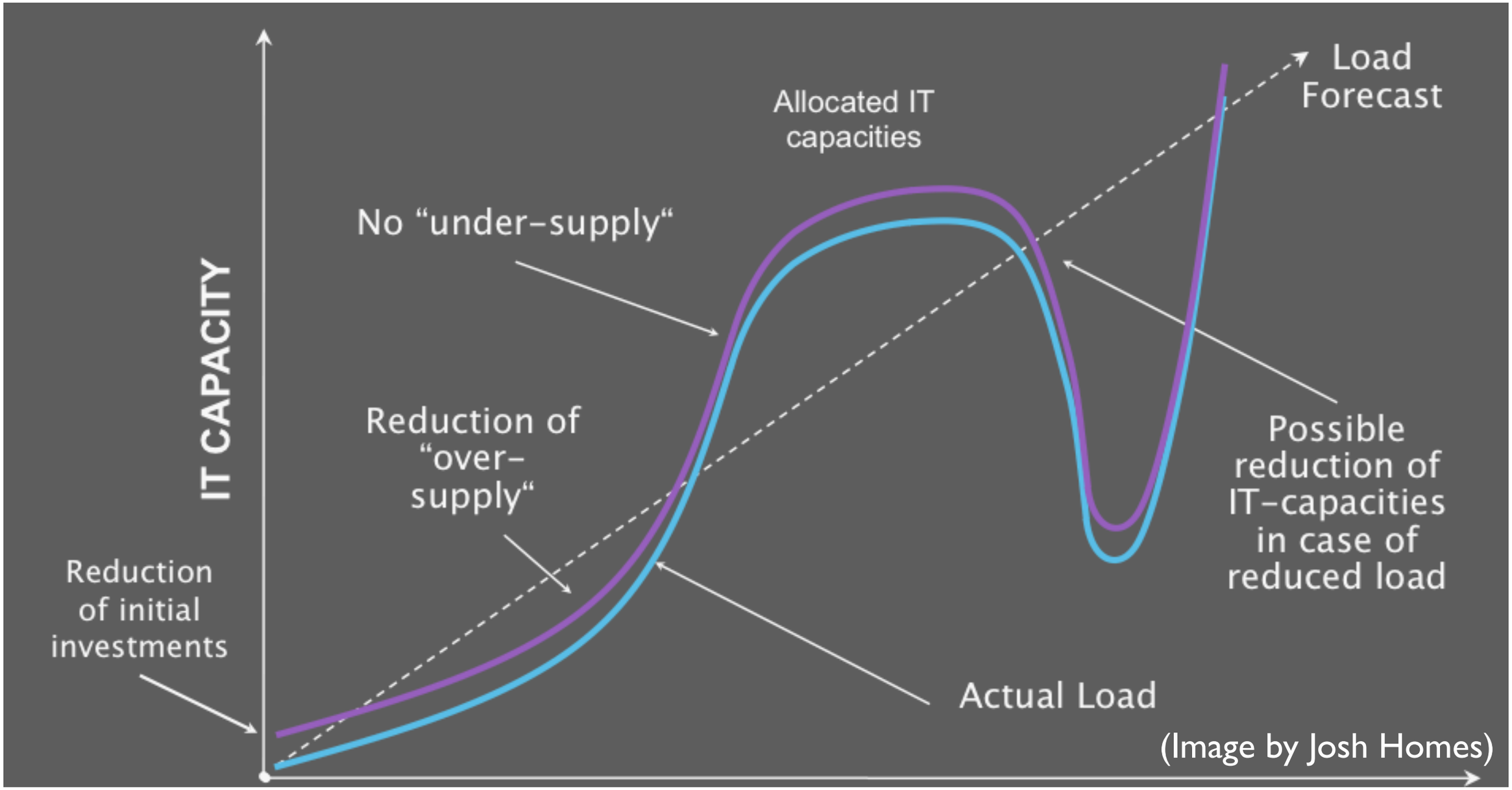
http://www.microsoft.com/windowsazure/

- Servers and storage on the Internet
  - Abstracted
  - Scalable
  - Per-usage payment

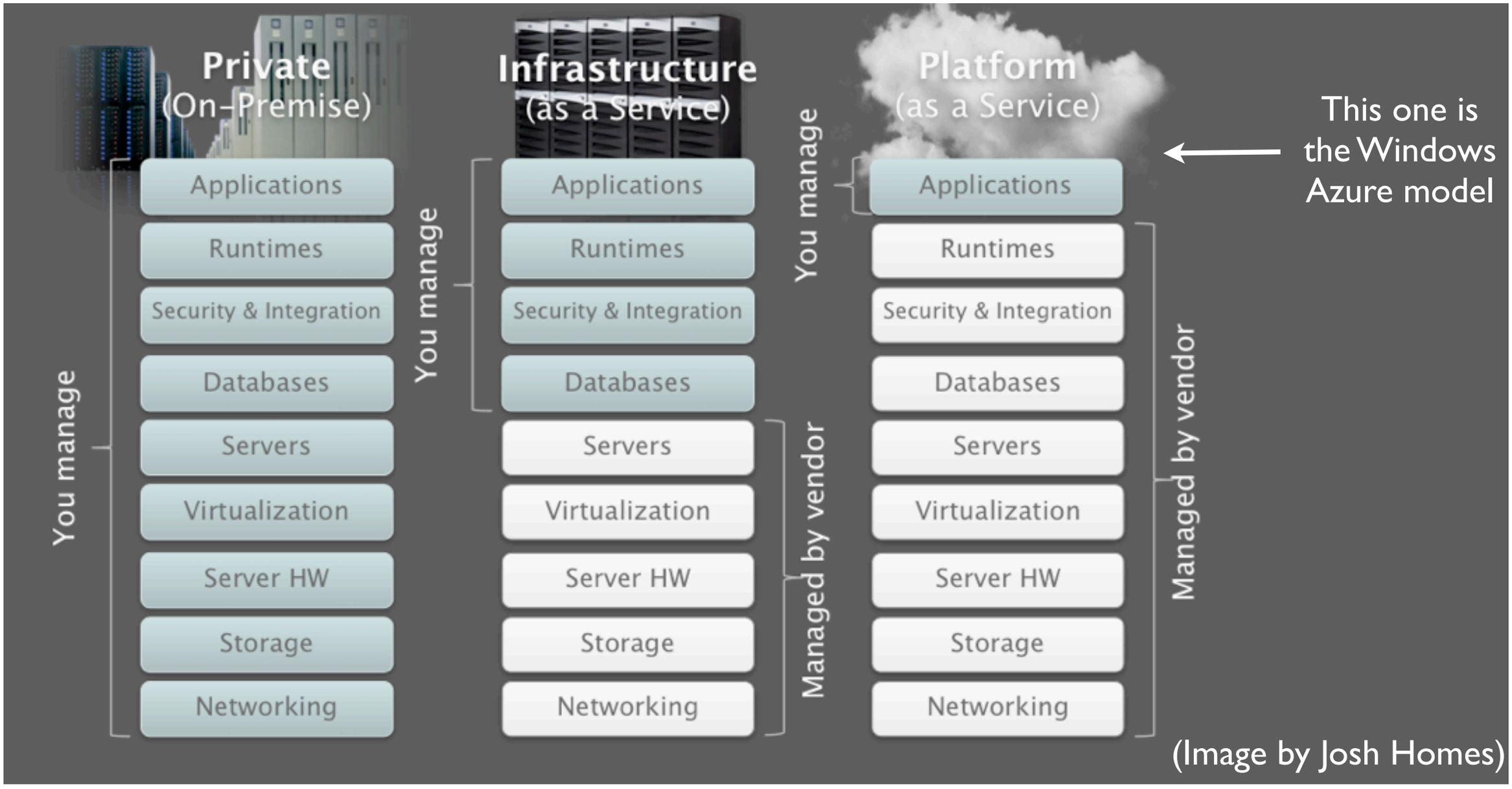
Traditional vs reality



Cloud vs reality



Infrastructure choices



Azure’s components

- Computation (aka the “Fabric”)
- Storage services
- SQL Azure
- Developer experience

The Fabric

- Abstract computing resources
- Servers running Windows Server 2008 in VMs
- Other paraphernalia: load balancers & switches
- 4 VM sizes: S, M, L, XL
- Some .NET stuff
- More importantly: FastCGI !

Roles

- Web role
  - Accepts HTTP/HTTPS requests
  - IIS7
  - Must be stateless
- Worker role
  - No incoming requests
  - Outgoing requests are used
  - More like a daemon (service)

AppFabric

- Service Bus
  - Relays messages from external clients to internal apps via the Azure platform
  - Solves firewall & NAT problems
- Access Control
  - “Federated identity & access control” for apps and clients
- REST & SOAP access

Storage services

- Independently accessible
- REST API
- Scalable
- Types:
  - Blobs
  - Tables
  - Queues
  - XDrive (NTFS)

Storage: Blobs

- Simple and easy
- one blob = one file
- Each account has many containers
- Each container has many blobs
- Large: 50GB!
- Public or private

Storage: Tables

- NOT relational database tables!
- More like organised key-value storage
- Each account has many *tables*
- Each table has many *properties*
- A property has a *name*, *type* and *value*
- No schema
- Versioned
  - Supply version on update => failure if out of date

Storage: Queues

- Allows web roles to instruct worker roles
- Process:
  - Web roles adds message
  - Worker role reads messages and does work
  - Worker role deletes message
- Caveats:
  - No guarantee of order
  - Message may be worked on twice

Storage: XDrive

- NTFS drive in the cloud: X:\
- Fixed size between 16MB and 1TB
- Max 16 drives on your VM
- Implemented as a page blob
- Single instance write
- Multiple instance read

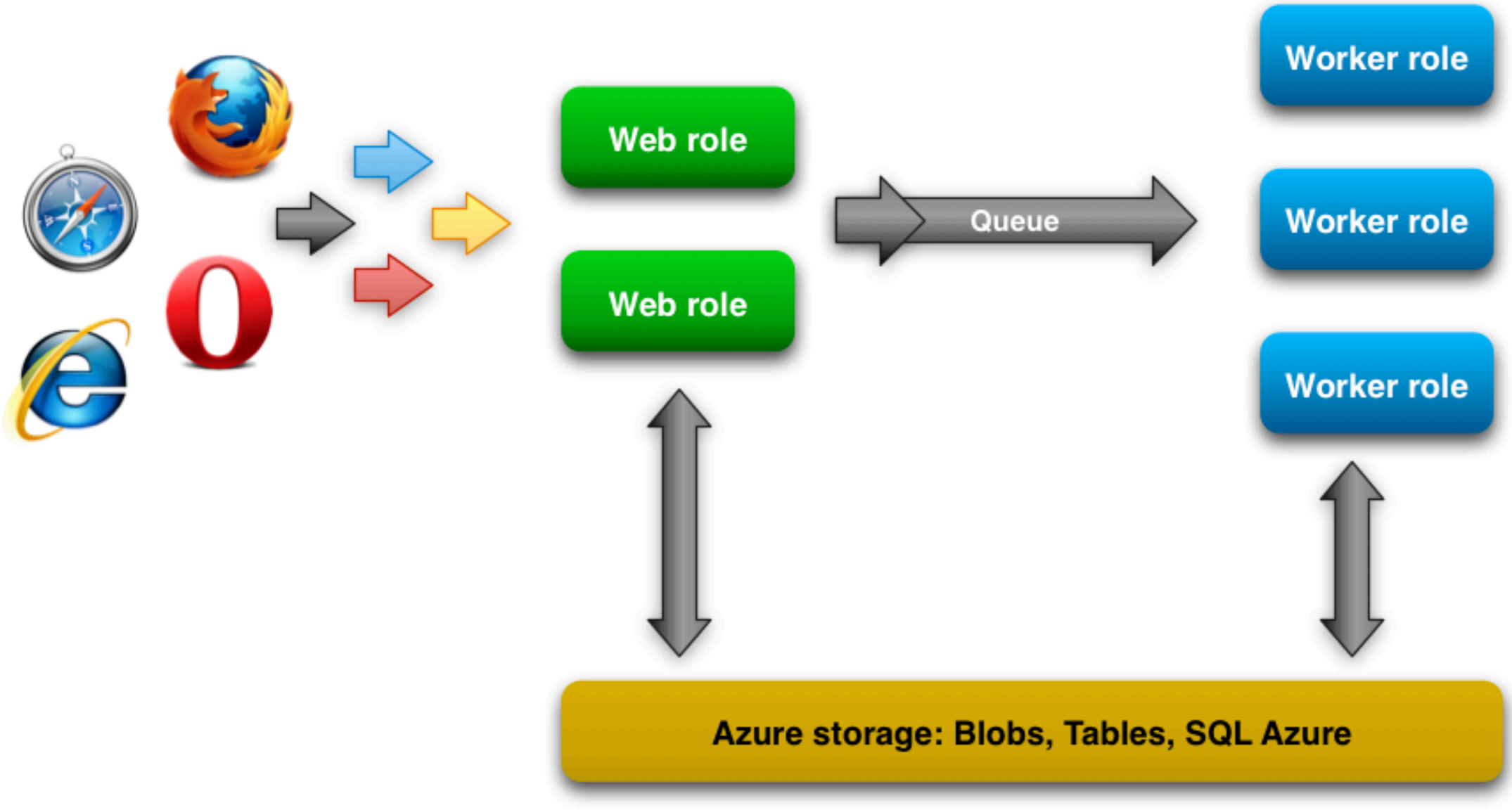
SQL Azure

- SQL database
- Relational data model
- Familiar development
- Scalable
  - abstracted database servers
  - Load balanced
  - Automatic failover built-in

Developer experience

- Offline development environment
  - Fabric and storage services
- Command line SDK
- Visual Studio plugin
- PHP SDK
- Eclipse tooling

Azure architecture



Deployment experience

- 2 step deployment
  - Upload to staging
  - Control panel operation to move to production
    - atomic operation: no downtime
  - Real domain uses a cname to the cloud one

Developing for Azure with PHP

Running PHP on Azure

- Use a web role
- Supply PHP runtime
- Configure FastCGI
- Use *Windows Azure SDK for PHP*
- (or use Zend_Server_WindowsAzure)
- Deploy:
  - CLI: CSPack.exe & CSRun.exe
  - Use *Windows Azure for Eclipse* tooling

Azure Tools for Eclipse

- The easy route to Azure PHP development
- Azure projects
  - Azure service project
  - Web role project
- Run in development fabric
- One click project deployment
- Storage explorer

Configuration files

- Per application:
  - Service definition file
  - Service configuration file
- Per role:
  - role config file

ServiceDefinition.csdef

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="Simple"
    xmlns="http://schemas.microsoft.com/{etc}">
  <WebRole name="WebRole"
      enableNativeCodeExecution="true">
    <ConfigurationSettings>
    </ConfigurationSettings>
    <InputEndpoints>
      <!-- Port 80 for http and port 443 for https -->
      <InputEndpoint name="HttpIn" protocol="http" port="80" />
    </InputEndpoints>
  </WebRole>
</ServiceDefinition>
```

ServiceConfiguration.cscfg

```
<?xml version="1.0"?>
<ServiceConfiguration serviceName="Simple"
    xmlns="http://schemas.microsoft.com/{etc}">
  <Role name="WebRole">
    <Instances count="2"/>
    <ConfigurationSettings>
    </ConfigurationSettings>
  </Role>
</ServiceConfiguration>
```

Web.config

```
 <system.webServer>
<handlers>
 <clear />
 <add name="PHP via FastCGI" path="*.php" verb="*"
      modules="FastCgiModule"
      scriptProcessor="%RoleRoot%\approot\php\php-cgi.exe"
      resourceType="Unspecified" />
 <add name="StaticFile" path="*" verb="*"
      modules="StaticFileModule,DefaultDocumentModule"
      resourceType="Either" requireAccess="Read" />
</handlers>
<defaultDocument>
  <files>
    <clear />
    <add value="index.php" />
  </files>
</defaultDocument>
```

PHP SDKs

- Windows Azure SDK for PHP:
  - Blobs
  - Tables
  - Queues
  - Misc low-level Azure services
- AppFabric SDK for PHP Developers:
  - Access Control Service
  - Service Bus

Blobs

```
   $blobStorage = new Microsoft_WindowsAzure_Storage_Blob();
// Create
if (!$blobStorage->containerExists($containerName))
{
    $blobStorage->createContainer($containerName);
    $blobStorage->setContainerAcl($containerName,
        Microsoft_WindowsAzure_Storage_Blob::ACL_PUBLIC);
}

// Store
$blob = $blobStorage->putBlob($containerName, $blobName,
    $localFilename, $metadata);
/* @var $blob Microsoft_WindowsAzure_Storage_BlobInstance */
```

Blobs

```
   // Copy
$result = $blobStorage->copyBlob($containerName, $blobName,
$containerName, $blob2Name);

// Retrieve
$tempStore = azure_getlocalresourcepath('tempstore');
$tempPath = $tempStore . '\\' . $imageId;
$blobStorage->getBlob($containerName, $blobName, $tempPath);

// Delete
$result = $blobStorage->deleteBlob($containerName, $blobName);
$result = $blobStorage->deleteContainer($containerName);
```

Blob stream wrapper

```
   $blobStorage = new Microsoft_WindowsAzure_Storage_Blob();
// Register:
$blobStorage->registerStreamWrapper(); // registers azure://
// or
$blobStorage->registerStreamWrapper('blob://'); // use blob://

// Use
$fp = fopen('azure://mycontainer/myfile.txt', 'r');
// ...
fclose($fp);
```

Tables

```
$tableStorage = new Microsoft_WindowsAzure_Storage_Table(
  'table.core.windows.net', 'myaccount', 'myauthkey');

// Create
$result = $tableStorage->createTable($tableName);

// List
$result = $tableStorage->listTables();
foreach ($result as $table) {
    echo 'Table name is: ' . $table->Name . "\n";
}

// Delete
$tableStorage->deleteTable($tableName);
```

Entities within a table

```
   // Structured entity
class ImageEntity extends
    Microsoft_WindowsAzure_Storage_TableEntity
{
    /**
     * @azure filename
     */
    public $filename;
    /**
     * @azure size Edm.Int64
     */
    public $size;
}
// Unstructured entity
// Microsoft_WindowsAzure_Storage_DynamicTableEntity
```

Entities within a table

```php
   // Insert
$image = new ImageEntity($partitionKey, $rowKey);
$image->filename = $_FILES['imageUpload']['name'];
$image->size = $_FILES['imageUpload']['size'];
$image = $tableStorageClient->insertEntity($tableName, $image);
```

```php
// Retrieve
$image = $tableStorage->retrieveEntityById($tableName,
    $partitionKey, $rowKey, 'ImageEntity');

// Update
$image->filename = 'newname.jpg';
$result = $tableStorage->updateEntity($tableName, $image);

// Delete
$result = $tableStorage->deleteEntity($tableName, $image);
```

Table queries

```
   // Filter condition
$select = "filesize gt 1024 and PartitionKey eq '$partitionKey'";

// Or fluent interface
$select = $tableStorage->select()->from($tableName)
          ->where('filesize gt 1024')
          ->andWhere('PartitionKey eq ?', $partitionKey);

// Run query
$images = $tableStorage->storageClient->retrieveEntities(
    'testtable', $select, 'ImageEntity'
);
foreach ($images as $image) {
    echo 'Name: ' . $image->filename . "\n";
}
```

Batch operation

```
   // Start batch
$batch = $tableStorage->startBatch();

// Insert multiple
$images = generateEntities();
foreach ($images as $image) {
    $tableStorage->insertEntity($tableName, $image);
}

// Commit
$batch->commit();
```

Table session handler

```
   $sessionHandler
     = new Microsoft_WindowsAzure_SessionHandler($tableStorage);
$sessionHandler->register();

session_start();

if (!isset($_SESSION['start'])) {
    $_SESSION['start'] = time();
}
```

Queues

```
   $queueClient = new Microsoft_WindowsAzure_Storage_Queue();

// Create
$result = $queueClient->createQueue('imageQueue');

// Delete
$queueClient->deleteQueue('imageQueue');

// Add message
$queueClient->putMessage('imageQueue', $message, $ttl);

// Retrieve Messages
$messages = $queueClient->getMessages('imageQueue', 10);
foreach ($messages as $message) {
    // Do work here...
    $queueClient->deleteMessage('imageQueue', $message);
}
```

Zend Framework

- *Zend_Service_WindowsAzure*
- From Zend Framework 1.10
- Same author as the Azure PHP SDK
- Familiar ZF API into Azure Storage:
  - `Zend_Service_WindowsAzure_Storage_Blob`
  - `Zend_Service_WindowsAzure_Storage_Table`
  - `Zend_Service_WindowsAzure_SessionHandler`
  - `Zend_Service_WindowsAzure_Storage_Queue`
- One manual page for all information!

Resources on the web

Windows Azure site:

http://www.microsoft.com/windowsazure/

PHP SDKs:

http://phpazure.codeplex.com

http://dotnetservicesphp.codeplex.com/

http://framework.zend.com/manual/en/zend.service.windowsazure.html

Windows Azure for Eclipse:

http://www.windowsazure4e.org

Rob's Top Tips

1. Use Azure Tables over SQL if you can
2. Don't forget the SessionHandler!
3. Eclipse tooling makes it easy

Thank you

Provide feedback on this talk: http://joind.in/1468