



The PHP Company

Manage cloud infrastructures using Zend Framework

by Enrico Zimuel (enrico@zend.com)

Senior Software Engineer
Zend Framework Core Team
Zend Technologies Ltd



About me



Email: enrico@zend.com
Twitter: @ezimuel

- Software Engineer since 1996
 - Assembly x86, C/C++, Java, Perl, PHP
- Enjoying PHP since 1999
- PHP Engineer at Zend since 2008
- ZF Core Team from April 2011
- B.Sc. Computer Science and Economics from University of Pescara (Italy)



Summary

- Cloud computing in PHP
- Zend\Service\Rackspace
- Simple Cloud API
- Zend\Cloud\Infrastructure for ZF2 and ZF1
- Adapters: Amazon Ec2, Rackspace
- Examples

Cloud computing



“Wilber is probably taking this Cloud computing too seriously.”

Cloud for developers

- Needs for developers:
 - ▶ Standard API (Open Stack, Open Cloud Computing Interface, ?)
 - ▶ Development infrastructures
 - ▶ Libraries/Frameworks “cloud ready”


API for the cloud

- API (Application Programming Interface), to interact with cloud services
- Typically use REST-based APIs
- Each vendor exposes a property API
- Learning curve for each cloud vendor

PHP libraries for cloud

- **Amazon Web Services**
 - ▶ AWS SDK for PHP, <http://aws.amazon.com/sdkforphp/>
- **Windows Azure**
 - ▶ PHPAzure, <http://phpazure.codeplex.com/>
- **Rackspace**
 - ▶ php-cloudfiles, <http://bit.ly/ptJa1Y>
- **GoGrid**
 - ▶ GoGridClient, <http://bit.ly/o7MeLA>

ZF components for the cloud

- Zend\Service\Amazon
- Zend\Service\GoGrid (under dev)
- Zend\Service\Nirvanix
- **Zend\Service\Rackspace** 
- Zend\Service\WindowsAzure



Zend\Service\Rackspace

Zend\Service\Rackspace



- Manage the following cloud services of Rackspace:
 - ▶ Servers
 - ▶ Files
- Provide a full OO interface for the API of Rackspace (ver 1.0)
- Release: ZF1 1.12 (in trunk now), ZF2 beta1

Example: authentication

```
$user = 'username';  
$key  = 'secret key';  
  
$rackspace = new Zend\Service\Rackspace\Files($user,$key);  
  
if ($rackspace->authenticate()) {  
    echo "Authentication successfully";  
} else {  
    printf("ERROR: %s", $rackspace->getErrorMsg());  
}
```

Example: store object

```
...
$container = $rackspace->createContainer('test');
if (!$rackspace->isSuccessful()) {
    die('ERROR: ' . $rackspace->getErrorMsg());
}
$name = 'example.jpg';
$file = file_get_contents($name);
$metadata = array (
    'foo' => 'bar'
);
$rackspace->storeObject('test', $name, $file, $metadata);
if ($rackspace->isSuccessful()) {
    echo 'Object stored successfully';
} else {
    printf("ERROR: %s", $rackspace->getErrorMsg());
}
```

Example: create a server

```
$user = 'username';
$key  = 'secret key';

$rackspace = new Zend\Service\Rackspace\Servers($user,$key);

$data = array (
    'name'      => 'test',
    'imageId'   => '49',
    'flavorId'  => '1',
);
$server = $rackspace->createServer($data);

if (!$rackspace->isSuccessful()) {
    die('ERROR: '.$rackspace->getErrorMsg());
}

printf("Server name      : %s\n", $server->getName());
printf("Server Id       : %s\n", $server->getId());
printf("Admin password  : %s\n", $server->getAdminPass());
```

Simple Cloud API

Simple Cloud API

- The Simple Cloud API is a common API for accessing cloud application services offered by multiple vendors
- Starting from November 2010 the Simple Cloud API is part of Zend Framework under the classname:
 - ▶ `Zend_Cloud` (ZF1)
 - ▶ `Zend\Cloud` (ZF2)



simplecloud.org



The Simple Cloud API tackles the main challenges of cloud computing adoption- portability and interoperability.

- Larry Augustin, CEO, SugarCRM

[HOME](#) [DOCUMENTATION](#) [DOWNLOAD](#) [PRESS](#) [CONTACTS](#)

Simple Cloud API

The Simple Cloud API brings cloud technologies to **PHP** and the PHPilosophy to the cloud, starting with common interfaces for four cloud application services:

 [File Storage Services](#)

 [Document Storage Services](#)

 [Queue Services](#)

 [Infrastructure](#)

You can start writing scalable, highly available, and resilient cloud applications that are portable across all major cloud vendors today.

Zend has invited the open source community and software vendors of all sizes to participate. IBM, Microsoft, Rackspace, Nirvanix, and GoGrid have already joined the project as contributors.

Welcome to a simpler cloud!



Co-founding contributors:



Microsoft

IBM

 **rackspace**
HOSTING

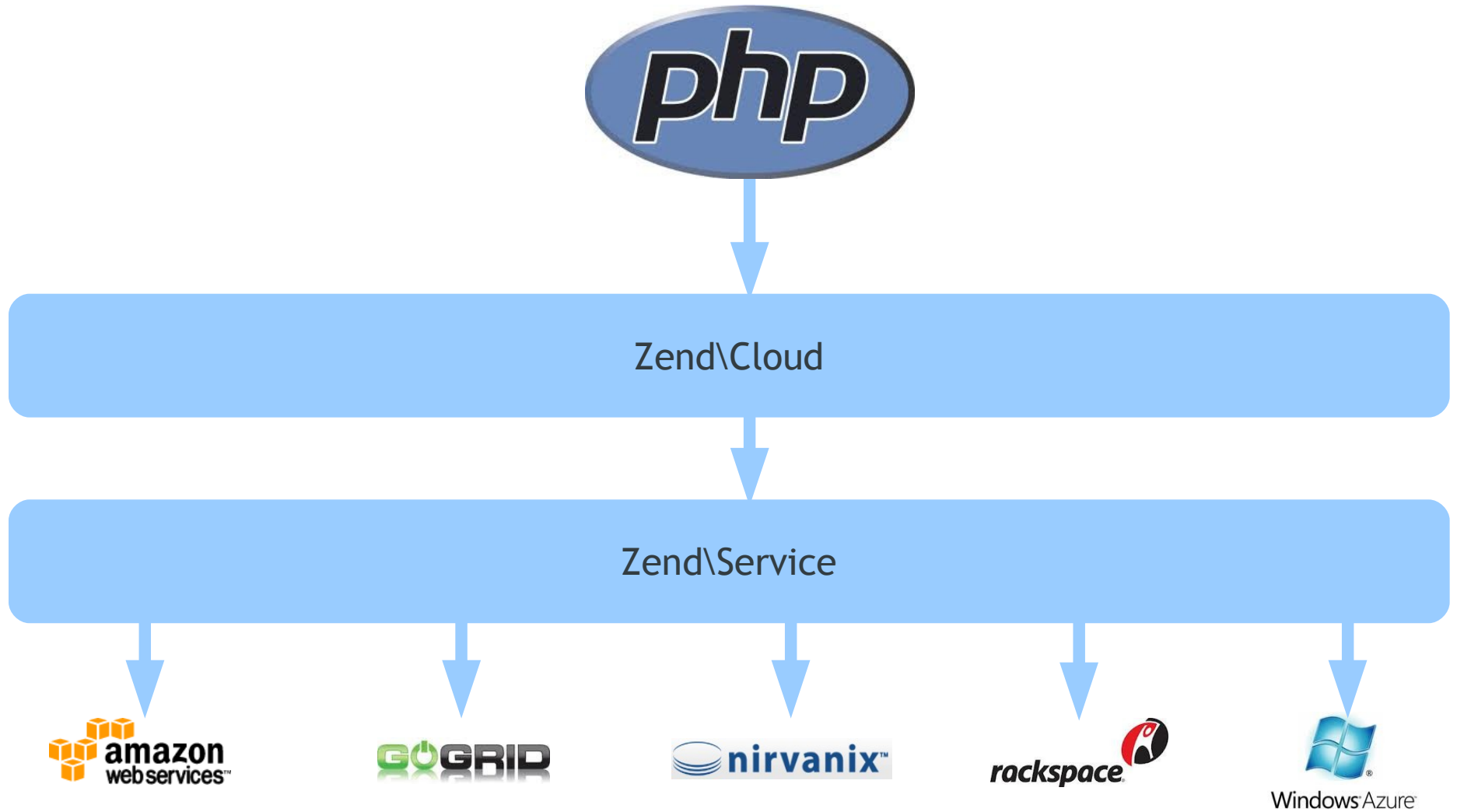
 **nirvanix**

GOGRID

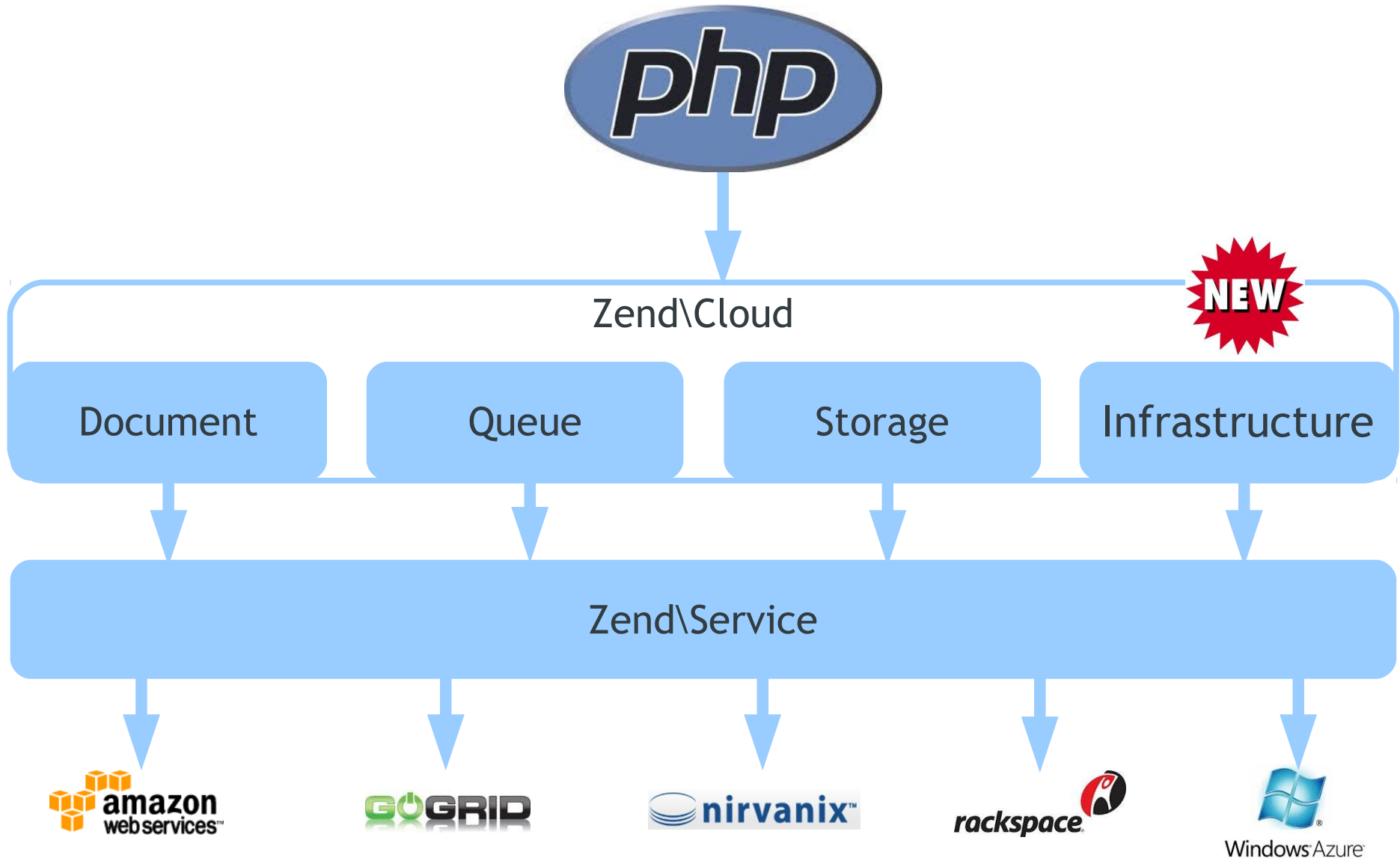
Why we need it?

- **Vendor lock-in**
 - ▶ In economics, vendor lock-in makes a customer dependent on a vendor for products and services, unable to use another vendor without substantial switching costs
- **Portability**
 - ▶ reuse the existing code instead of creating new code when moving software from an environment to another

The architecture



The architecture (2)



Zend\Cloud as abstraction

- Zend\Cloud is an abstraction of the main features of some cloud vendors
- Vendor specific functions may not be included in Zend\Cloud (for portability reason)
 - ▶ For instance, Amazon S3 has a cleanBucket operation that is not implemented in Zend\Cloud
- You can access the concrete adapters to use specific functions (getAdapter)

Zend\Cloud\DocumentService

- Abstracts the interfaces to all major document databases - both in the cloud and locally deployed
- **Adapters:**
 - ▶ Amazon SimpleDB
 - ▶ Windows Azure

Zend\Cloud\QueueService

- The QueueService implements access to message queues available as local or remote services.
- **Adapters:**
 - ▶ Amazon Sqs
 - ▶ Windows Azure
 - ▶ Zend\Queue

Zend\Cloud\StorageService

- The storage service in the Simple Cloud API implements a basic interface for file storage on the cloud
- **Adapters:**
 - ▶ Amazon S3
 - ▶ Windows Azure
 - ▶ Nirvanix
 - ▶ Filesystem
 - ▶ Rackspace (under dev)

Zend\Cloud\Infrastructure

Zend\Cloud\Infrastructure

- Manage instances (servers) of a cloud computing infrastructure
- Release ZF1: 1.12 (in trunk now), ZF2 beta1
- **Adapters:**
 - ▶ Amazon Ec2
 - ▶ Rackspace Cloud Servers
 - ▶ GoGrid (under dev)
 - ▶ Windows Azure (under dev)

Basic operations

- Create a new instance
- Delete an instance
- Start/stop/reboot an instance
- List available instances
- Get the status of an instance (running, stop, etc)
- Monitor an instance (CPU, RAM, Network, etc)
- Deploy an instance
- Execute remote shell command (using SSH2)

Image of an instance

- An image of an instance is the collection of the following information:
 - ▶ Operating system (OS)
 - ▶ Memory available (RAM)
 - ▶ CPU type

Example: Amazon Ec2 adapter

```
use Zend\Cloud\Infrastructure\Adapter\Ec2 as Ec2Adapter,  
    Zend\Cloud\Infrastructure\Factory;  
  
$key      = 'key';  
$secret   = 'secret';  
$region   = 'region';  
  
$infrastructure = Factory::getAdapter(array(  
    Factory::INFRASTRUCTURE_ADAPTER_KEY =>  
    'Zend\Cloud\Infrastructure\Adapter\Ec2',  
    Ec2Adapter::AWS_ACCESS_KEY => $key,  
    Ec2Adapter::AWS_SECRET_KEY => $secret,  
    Ec2Adapter::AWS_REGION     => $region,  
));
```

Example: create instance

```
$param= array (  
    'imageId'      => 'your-image-id',  
    'instanceType' => 'your-instance-type',  
);  
  
$instance= $infrastructure->createInstance('name', $param);  
  
if ($instance===false) {  
    die ('Error');  
}  
  
printf ("Name of the instance: %s\n", $instance->getName());  
printf ("ID of the instance   : %s\n", $instance->getId());
```

Example: reboot and wait for status change

```
if (!$infrastructure->rebootInstance('instance-id')) {
    die ('Error in the execution of the reboot command');
}
echo 'Reboot command executed successfully';

if ($infrastructure->waitStatusInstance('instance-id',
Instance::STATUS_RUNNING)) {
    echo 'The instance is ready';
} else {
    echo 'The instance is not ready yet';
}
```

Wait for status change

`waitStatusInstance (string $id, string $status, integer $timeout=30)`

- wait the status change of an instance for a maximum time of n seconds (30 by default).
- returns *true* if the status changes as expected, *false* otherwise.

Example: monitor an instance

```
use Zend\Cloud\Infrastructure\Instance;  
  
$cpuUsage= $infrastructure->monitorInstance(  
    'instance-id',Instance::MONITOR_CPU);  
  
var_dump($cpuUsage);
```

```
array(2) {  
    ["series"] => array(3) {  
        [0]=> array(2) {  
            ["timestamp"] => int(1318348800)  
            ["value"]=> int(80)  
        }  
        [1]=> array(2) {  
            ["timestamp"] => int(1318348860)  
            ["value"] => int(70)  
        }  
        [2] => array(2) {  
            ["timestamp"] => int(1318348920)  
            ["value"] => int(60)  
        }  
    }  
    ["average"] => string(3) "70"  
}
```

Example: deploy an instance

```
$nodeId= 'id-instance';

$params= array (
    Instance::SSH_USERNAME => 'username',
    Instance::SSH_PASSWORD => 'password'
);

$cmd= 'ls -la /var/www';

$output= $infrastructure->deployInstance($nodeId,$params,$cmd);

echo "The files in the DocumentRoot of the $nodeId instance are:\n";
print_r ($output);
```

Note: require the SSH2 extension

New mailing list!

zf-cloud@lists.zend.com

- to subscribe send an empty email to:
 - ▶ zf-cloud-subscribe@lists.zend.com



Questions?



Thank you!

More info:

<http://www.zend.com>

<http://framework.zend.com/>

<http://devzone.zend.com>



The PHP Company

