Platform as a Service (PaaS)
Azure + Google App engine +...
IaaS PaaS SaaS Comparison

- **IaaS (Infrastructure as a Service):**
  - Applications
  - Data
  - Runtime
  - Middleware
  - O/S
  - Virtualization
  - Servers
  - Storage
  - Networking

- **PaaS (Platform as a Service):**
  - Applications
  - Data
  - Runtime
  - Middleware
  - O/S
  - Virtualization
  - Servers
  - Storage
  - Networking

- **SaaS (Software as a Service):**
  - Applications
  - Data
  - Runtime
  - Middleware
  - O/S
  - Virtualization
  - Servers
  - Storage
  - Networking

Management:
- Managed by vendor: Networking, Storage, Servers, Virtualization, O/S, Middleware, Runtime, Data, Applications.
Cloud Fundamentals

- **Infrastructure as a Service (IaaS):** basic compute and storage resources
  - On-demand servers
  - Amazon EC2, VMWare vCloud

- **Platform as a Service (PaaS):** cloud application infrastructure
  - On-demand application-hosting environment
  - E.g. Google AppEngine, Salesforce.com, Windows Azure, Amazon

- **Software as a Service (SaaS):** cloud applications
  - On-demand applications
  - E.g. GMail, Microsoft Office Web Companions
Platform as a Service (PaaS)

- **PaaS** is a cloud computing service that offers a platform for users to run applications onto the cloud.

- It is a level above Infrastructure as a service (IaaS) because unlike IaaS, PaaS does not require users to develop their own operating system environment.
Platform as a Service (PaaS)

• Middle ground between SaaS and IaaS
• Development platform
  – Customers use to develop applications that benefit from the scalability of the cloud without fully developing their own solution using an IaaS provider
• Offers an application development platform that will automatically scale with demand
The Benefits of the Cloud

- The Cloud is about cheap, on-demand capacity

<table>
<thead>
<tr>
<th>= Managed for You</th>
<th>Standalone Servers</th>
<th>IaaS</th>
<th>PaaS</th>
<th>SaaS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Runtimes</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtualization</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Networking</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Platform as a Service (PaaS)

- **Official definition of PaaS from NIST standard**
- “The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.”
Example: Google

PaaS
Runtime environment, database, development

Google App Engine

Amazon AWS EC2
Example: Windows Azure

• Platform as a Service
  – Application Platform in the Cloud

• Provides:
  – *Compute*
    • Web, Worker & VM Role
  – *Storage*
    • Blob, Table, Queue & Azure SQL Server
  – Application *Fabric*
    • Service Bus, Access Control, (Future: Cache, Integration & Composite)
More Cost Effective

• PaaS can be better for costs than IaaS, as systems are optimized to run applications efficiently.

• IaaS may only provide hardware and thus clients have to be in charge of load balancing and networking.
Multi-Tenancy

• PaaS is better suited for multi-tenancy as the PaaS provider optimizes their infrastructure for use by many providers

• Multi-tenancy means that many users may share the same physical computer and database
Multi-Tenancy

- PaaS is better suited for multi-tenancy than an IaaS because an IaaS may provide each user with their own virtual machine and create a clear separation of resources.

- However, in a PaaS, users may share the same machine, database, etc.
Vendor Lockin

• PaaS may lock in applications by requiring users to develop apps using proprietary interfaces and languages

• This means that it may be difficult for users to go to another vendor to host their app

• Businesses may risk their future on the dependability of the PaaS
Development Tools

- Many PaaS offer Browser-based development tools
- In this way, developers can create their own applications online
- Ease of deployment, the platform takes care of the scaling for you
Principles of Software Development

• As a developer, your objective is to create an application in the quickest, most effective way possible

• One should not create applications using convoluted methods that may take a long time to complete

• The user only sees the end product, not the development process.
PaaS vs. Iaas

• You need to make decisions with long-term consequences, when you use cloud

• If you choose to use a PaaS and get your application vendor locked in, then your business may fail if the PaaS greatly increases their prices

• You will not be able to move to another cloud since your app cannot be easily migrated somewhere else
PaaS vs. IaaS

• An app that is used to fulfill a temporary need, may be handled by a PaaS solution.

• An app that may need to be deployed quickly, may be faster developed by a PaaS.

• If your software team is small, it may be better to develop on a PaaS and let the PaaS provider handle the OS and networking for your team.
PaaS vs. IaaS

• An app that must serve a variety of purposes for the long term may be better developed on IaaS

• If you need flexibility to change dev tools, languages then an IaaS may be better

• A large software development team may have the resources to optimize and monitor an IaaS system
Microsoft Azure

• It was launched by Microsoft in 2010

• Provides both PaaS and IaaS services

• It is like a hybrid cloud provider that tries to do multiple things
Uses of Azure

• Can be used for anything since it provides IaaS services that can host virtual machines

• However, its PaaS services have been known to host web sites that may receive a lot of traffic

• Good for .NET developers
Azure Cloud

• Microsoft developed their own operating system called Windows Azure that is used for their datacenter cluster

• Uses Hyper-V, a windows server Hypervisor that can run virtual machines
Windows Server

• Has support for Windows server

• Can provision and manage virtual machines

• Can attach and manage disks
Windows Azure

- **Windows Azure is the OS for the data center**
  - Model: Treat the data center as a machine
  - Handles resource management, provisioning, and monitoring
  - Manages application lifecycle
  - Allows developers to concentrate on business logic

- **Provides shared pool of compute, disk and network**
  - Virtualized storage, compute and network
  - Illusion of boundless resources

- **Provides common building blocks for distributed applications**
  - Reliable queuing, simple structured storage, SQL storage
  - Application services like access control and connectivity
# Windows Azure Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Windows Azure PaaS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications</td>
<td>Windows Azure Service Model</td>
</tr>
<tr>
<td>Runtimes</td>
<td>.NET 3.5/4, ASP .NET, PHP</td>
</tr>
<tr>
<td>Operating System</td>
<td>Windows Server 2008/R2-Compatibile OS</td>
</tr>
<tr>
<td>Virtualization</td>
<td>Windows Azure Hypervisor</td>
</tr>
<tr>
<td>Server</td>
<td>Microsoft Blades</td>
</tr>
<tr>
<td>Database</td>
<td>SQL Azure</td>
</tr>
<tr>
<td>Storage</td>
<td>Windows Azure Storage (Blob, Queue, Table)</td>
</tr>
<tr>
<td>Networking</td>
<td>Windows Azure-Configured Networking</td>
</tr>
</tbody>
</table>
Modeling Cloud Applications

• A cloud application is typically made up of different components
  – Front end: e.g. load-balanced stateless web servers
  – Middle worker tier: e.g. order processing, encoding
  – Backend storage: e.g. SQL tables or files
  – Multiple instances of each for scalability and availability
The Windows Azure Service Model

• A Windows Azure application is called a “service”
  – Definition information
  – Configuration information
  – At least one “role”

• Roles are like DLLs in the service “process”
  – Collection of code with an entry point that runs in its own virtual machine

• There are currently three role types:
  – Web Role: IIS7 and ASP.NET in Windows Azure-supplied OS
  – Worker Role: arbitrary code in Windows Azure-supplied OS
  – VM Role: uploaded VHD with customer-supplied OS
Role Contents

• Definition:
  – Role name
  – Role type
  – VM size (e.g. small, medium, etc.)
  – Network endpoints

• Code:
  – Web/Worker Role: Hosted DLL
    and other executables
  – VM Role: VHD

• Configuration:
  – Number of instances
  – Number of update and fault domains
Service Model Files

- Service definition is in `ServiceDefinition.csdef`
- Service configuration is in `ServiceConfiguration.cscfg`
- CSPack program Zips service binaries and definition into service package file (`service.cscfg`)
Availability: Update Domains

- **Purpose:** Ensure service stays up while updating and Windows Azure OS updates
- **System considers update domains when upgrading a service**
  - Update domains/Instance count = percent of service that will be offline
  - Default and max is 5, but you can override with upgradeDomainCount service definition element
- **The Windows Azure SLA is based on at least two update domains and two role instances in each role**
Availability: Fault Domains

- Purpose: Avoid single points of failures
  - Similar concept to update domains
  - But you don’t control the updates
- Unit of failure based on data center topology
  - E.g. top-of-rack switch on a rack of machines
- Windows Azure considers fault domains when allocating service roles
  - E.g. don’t put all roles in same rack
Deploying a Service
The 10,000 foot view

- Service package uploaded to portal
  - Windows Azure Portal Service passes service package to “Red Dog Front End” (RDFE) Azure service
  - RDFE converts service package to native “RD” version
- RDFE sends service to Fabric Controller (FC) based on target region
- FC stores image in repository and deploys and activates service
The Fabric Controller (FC)

• The “kernel” of the cloud operating system
  – Manages datacenter hardware
  – Manages Windows Azure services

• Four main responsibilities:
  – Datacenter resource allocation
  – Datacenter resource provisioning
  – Service lifecycle management
  – Service health management

• Inputs:
  – Description of the hardware and network resources it will control
  – Service model and binaries for cloud applications
Datacenter Architecture

Top of Rack Switches

Racks

Power Distribution Units

Aggregation Routers and Load Balancers
Windows Azure Datacenters
Inside a Role VM

Resource Volume

OS Volume

Guest Agent

Role Host

Role Entry Point
Update Types

- There are two update types:
  - In-place update:
    - Supports changes to configuration or binaries, not service definition
    - Role instances upgraded one update domain at a time
    - Two modes: automatic and manual
  - VIP swap update:
    - Service definition can change, but external endpoints must remain the same
    - New version of service deployed, external VIP/DIP mapping swapped with old
    - Changes to external endpoint count require a new deployment
Node and Role Health Maintenance

- FC maintains service availability by monitoring the software and hardware health
  - Based primarily on heartbeats
  - Automatically “heals” affected roles

<table>
<thead>
<tr>
<th>Problem</th>
<th>How Detected</th>
<th>Fabric Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role instance crashes</td>
<td>FC guest agent monitors role termination</td>
<td>FC restarts role</td>
</tr>
<tr>
<td>Guest VM or agent crashes</td>
<td>FC host agent notices missing guest agent heartbeats</td>
<td>FC restarts VM and hosted role</td>
</tr>
<tr>
<td>Host OS or agent crashes</td>
<td>FC notices missing host agent heartbeat</td>
<td>Tries to recover node</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FC reallocates roles to other nodes</td>
</tr>
<tr>
<td>Detected node hardware issue</td>
<td>Host agent informs FC</td>
<td>FC migrates roles to other nodes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Marks node “out for repair”</td>
</tr>
</tbody>
</table>
Azure Architecture Summary

- Platform as a Service is all about reducing management and operations overhead
- The Windows Azure Fabric Controller is the foundation for Windows Azure’s PaaS
  - Provisions machines
  - Deploys services
  - Configures hardware for services
  - Monitors service and hardware health
  - Performs service healing
PiCloud

• Python in the cloud
• Thousands of cores of computational power
• Terabytes of data storage (on Amazon S3)
• No need to manage, maintain, or configure servers
• You create a unit of computational work called a job
• Add the job to a queue, and when there's a core free, automatically run it
PiCloud Programming example

>>> # define your function (or import it)
>>> def add(x, y):
...   return x + y

>>> # import the PiCloud library
>>> import cloud

>>> # create a *job* that runs add() on the cloud
>>> job_id = cloud.call(add, 1, 2)

>>> print job_id
42

>>> # check the status of the job ('queued', 'processing', or 'done')
>>> cloud.status(job_id)
'done'

>>> # get the result
>>> cloud.result(job_id)
3
Interacting with PiCloud

cloud.call(...)  
cloud.status()

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>queued</td>
<td>Job is in a queue on the server waiting to be run.</td>
</tr>
<tr>
<td>processing</td>
<td>Job is running.</td>
</tr>
<tr>
<td>waiting</td>
<td>Job is waiting until its dependencies are satisfied.</td>
</tr>
<tr>
<td>done</td>
<td>Job completed successfully.</td>
</tr>
<tr>
<td>error</td>
<td>Job errored (typically due to an uncaught exception).</td>
</tr>
<tr>
<td>killed</td>
<td>Job was aborted by the user.</td>
</tr>
<tr>
<td>stalled</td>
<td>Job will not run due to a dependency erroring.</td>
</tr>
</tbody>
</table>
PiCloud Running example

• **Running a job**
  host:~$ JID=`picloud exec echo hello, world`
  host:~$ echo $JID
  42

• **check the status of the job**
  host:~$ picloud status $JID
done

• **get the result**
  host:~$ picloud result $JID
  hello, world
Run a function across many datapoints

`cloud.map()` mimics the built-in python map function. The basic built-in map function is:

```python
added2 = map(lambda x: x+2, an_iterator)
```

Which is equivalent to:

```python
added2 = [x+2 for x in an_iterator]
```
Case study: Calculating $\pi$

```
import random
total_tests = 500000000

def monteCarlo(num_test):
    """
    Throw num_test darts at a square
    Return how many appear within the quarter circle
    """
    num_in_circle = 0
    for _ in xrange(num_test):
        x = random.random()
        y = random.random()
        if x*x + y*y < 1.0: #within the quarter circle
            num_in_circle += 1
    return num_in_circle

def calcPi():
    num_in_circle = monteCarlo(total_tests)
    pi = (4 * num_in_circle) / float(total_tests)
    return pi

if __name__ == '__main__':
    pi = calcPi()
    print 'Pi determined to be %s' % pi
```
def calcPi():
    num_in_circle = monteCarlo(total_tests)
    pi = (4 * num_in_circle) / float(total_tests)
    return pi

http://docs.picloud.com/basic_examples.html#basic-examples
The `cloud.files` interface is quite simple:

- `cloud.files.put()`: Store a file on PiCloud’s S3 store.
- `cloud.files.get()`: Retrieve a file stored on PiCloud’s S3 store.
- `cloud.files.delete()`: Delete a file stored on PiCloud’s S3 store.

Example:

```
# This code below can run both locally or in a job running on PiCloud
cloud.files.put('names.txt')  # put names on the Cloud
cloud.files.get('names.txt','names2.txt')  # retrieve names.txt from the Cloud and store it as names2.txt
cloud.files.delete('names.txt')  # remove file
```
Google App Engine (GAE)

• GAE was developed in 2008 as a PaaS by google

• It supports multi tenancy and offers automatic scaling for web applications

• It supports Python, Java and Go
GAE frameworks and tools

• GAE supports Django web framework and the Grails web app framework

• GAE provides infrastructure tools that enable users to deploy code without worrying about infrastructure challenges such as deployment, failover, scalability

• However, the GAE infrastructure limits the type of applications that can be run
GAE Security, Sandbox

- Applications run in a secure environment

- Isolates applications from hardware and operating system, and imposes security limitations

- Ex. Application code only runs in response to requests and a request handler cannot spawn potentially malicious sub-processes after response has been sent
Storing GAE data

- Users of GAE can use App Engine Datastore, Google Cloud SQL, and Google Cloud Storage

- Can harness Google’s database technology like Bigtable
GAE’s use with Google Services

• Can take advantage of Google’s single sign on feature when other users want to access their gmail or google docs

• Build Chrome and Android games on GAE

• Google Cloud Endpoints to use access mobile services
Other Services supported

- App engine Map Reduce
- Search API
- SSL support
- Page Speed
- XMPP API
- Memcache API
Case Studies of GAE

• BugSense- An application error-reporting service, it used GAE to maintain logs of bugs in software and analyze them

• Ubisoft- used it to build their first web-based game, “From Dust” on Chrome browser

• Claritics- small social analytics company of 15 employees, used to analyze game datasets
GAE is great for Mobile

- Many cell phone apps use GAE for their backend like Ruzzle and Tap Zoo
- Fits GAE’s purpose well of being able to scale up for small teams of developers
CloudCmp

• Research by Duke and Microsoft to compare cloud providers in 2010

Comparison Methodology

• Test the performance of IaaS and PaaS providers
  – SaaS cannot be tested as it is too widely varied and using these benchmarks doesn't make sense

• Benchmark the service:
  – Per-Task Monetary Cost
  – Network Performance
  – Persistent Storage
  – Webpage Load Times
Which Providers?

• Amazon Web Services (C1)
  – Includes Beanstalk and EC2
• Rackspace CloudServers (C2)
• Google App Engine (C3)
• Microsoft Azure (C4)
• Not all providers offer all services, so some will not have values for certain benchmarks
Per-Task Monetary Cost

C1: Amazon AWS
C2: Rackspace CloudServers
C3: Google App Engine
C4: Microsoft Azure

• Verdict:
  o Rackspace is the most cost-friendly provider
  o Microsoft Azure is the most expensive to use
Inter-Datacenter TCP Throughput

Verdict:
- Amazon has highest throughput between datacenters
- Rackspace's is lamentably low

Figure 11: The TCP throughput between two different US data centers of a cloud provider.

C1: Amazon AWS
C2: Rackspace CloudServers
C4: Microsoft Azure
Intra-Datacenter TCP Throughput

- Decimal indicates different datacenters for each service
- Verdict:
  - Amazon has highest throughput within datacenters
  - Rackspace's is lamentably low

C1: Amazon AWS
C2: Rackspace CloudServers
C4: Microsoft Azure

Figure 10: The intra-datacenter TCP throughput between two instances in all data centers we measure.
Latency: Round Trip Time

C1: Amazon AWS
C2: Rackspace CloudServers
C3: Google App Engine
C4: Microsoft Azure

- C3 shows optimal performance and C3-actual shows average attainable performance
- Verdict:
  - Google is the fastest by far, even on average
  - Rackspace has the highest latency

Figure 12: This figure shows the cumulative distribution of the optimal round trip time (RTT) to the instances deployed on a cloud provider from 260 global vantage points. For C3, we also show the actual RTT from a vantage point to the instance returned by the cloud’s DNS load balancing.
Persistent Storage

• Cloud providers offer persistent storage to share data between instances
• Two types of storage:
  o Blob Storage for unstructured data, regular files
  o Table Storage for structured data, databases
• Rackspace (C2) doesn't offer a table storage service
• Google App Engine (C3) does not offer a blob storage service
Blob Download/Upload Times

- **Verdict:**
  - For small file sizes (1KB), Microsoft Azure is best
  - For large file sizes (10MB), Amazon AWS is best

![Graphs showing cumulative distribution of response times for downloading and uploading blobs of 1KB and 10MB from different cloud platforms.](image)

Figure 6: The cumulative distribution of the response time to download or upload a blob using Java-based clients.
Table Storage Operations

C1: Amazon AWS    C3: Google App Engine
C4: Microsoft Azure

• Verdict:
  o Amazon has fastest table query times by a significant margin
  o Microsoft Azure has noticeably slow table query time

Figure 4: The cumulative distribution of the response time when using the large table with 100K entries. Note that for the query operation, the x-axis is in a logarithmic scale, due to the significant performance gaps between different services.
**Webpage Load Time**

- **C1:** Amazon AWS
- **C2:** Rackspace CloudServers
- **C3:** Google App Engine
- **C4:** Microsoft Azure

**Verdict:**
- Google App Engine has fastest load times
- Rackspace is much slower than the rest

*Figure 15:* The distribution of the page downloading time of our website. We show the results for two different page sizes: 1KB and 100KB.
And the Winner Is...

- ...not immediately clear
- Different providers cater to different needs, no one provider is best at everything
- Google App Engine has the fastest load times, but is less flexible than the other providers
- Amazon AWS has highest throughput and data access times
- Rackspace CloudServers is very cost-effective, but has low performance
- Microsoft Azure is rather middle-of-the-road in terms of service, but has a very high price point