2.3 Matching cloud providers to your needs

We've looked under the hood of a lot of different cloud types, their APIs, their other service offerings, and the technologies that underpin them. Which of them is appropriate for you? How can you prevent lock-in when you do make the choice? We'll try to answer those questions by going back through the major cloud providers and applying a framework of decision criteria by which you can evaluate each one for your projects.

2.3.1 Amazon web services IaaS cloud

Summarizing what we've explored so far, AWS is a flexible, lower-level offering (closer to hardware), which means you have more possibilities. And in general, it will be higher performing at the cost of "everything is left up to you," including how and when to scale, move or replicate your data, and more.

Amazon EC2 runs the platform you provide, supports all major programming languages, and offers a set of industry-standard services (getting more standard as standards groups and the open source Eucalyptus seeks to formalize theirs as the standard cloud API). But Amazon, being an IaaS, requires much more work, which means a longer time-to-market for your applications.

Use AWS if you

- Want to use third-party open-source software
- Have existing code
- Want to transfer a web app to your own machine/servers later
- Port code to another language

- Want complete control
- Need to stress/load test an app (for example, load up 1,000 instances)

And as for avoiding lock-in, Amazon EC2 is good because Amazon-compatible services can and will be easily provided by other companies as well as an open-source initiative. The leader always gets to set the standards. EC2 is practically the closest to zero lock-in of any choice you can make today.

2.3.2 Microsoft Windows Azure IaaS and PaaS cloud

Azure is intermediate between application frameworks, such as App Engine, and hardware virtual machines, such as EC2. Microsoft is trying to make the transition from desktop (data center) to its cloud as seamless as possible. The company suggests that you can build and test an application locally and then deploy to its cloud. But Microsoft does admit that all UI and any data-extraction logic must be rewritten to deal with low-bandwidth internet connections. Note that we said *its cloud*. In that sense, Microsoft is similar to App Engine and Force.com in terms of locking you in to its cloud, run by the company.

Use Windows Azure if you

- Already use the .NET and SQL Server portions of the Microsoft stack
- Have existing code developed to those Microsoft APIs
- Have teams that normally develop in Visual Studio using C#
- Want to blend development from desk top to cloud
- Have no issue with lock-in to Microsoft

As for lock-in, Windows Azure isn't looking as bad as Google App Engine. Although it will still be hosted exclusively by Microsoft, it may be possible for other companies to come up with (almost) compatible cloud service because core pieces of Windows Azure are based on the well-known SQL Server, IIS, and .NET framework stacks.

2.3.3 Google App Engine PaaS cloud

Google App Engine is a tightly controlled environment—a decision Google made to enable automatic scaling of application threads as well as the datastore. The environment supports only Python and Java, and no installation of any open source software is possible.

Use App Engine if you

- Have no preexisting code
- Are building request-response web apps or mashups
- Consider time-to-market the most important thing
- Aren't doing anything fancy (installing software)
- Aren't worried about lock-in to Google

App Engine is high on the lock-in scale. It's hard to imagine any compatible products from any other company for a long time, if ever. It's proprietary, and Google doesn't plan

to release its technology. Automatic scale and time-to-market have many advantages, but almost complete lock-in will most likely be the price you pay for those benefits.

2.3.4 Ruby on Rails PaaS cloud

Ruby is slightly more computationally expensive than other languages, but having easy resource expansion available can cure a lot of the "what if I get mentioned on Oprah?" scares that business people experience. Rails is a particularly good match for cloud computing because of its shared-nothing architecture. This means you can generate new instances of an application, and they will begin to run. And developers love Ruby because of their much higher productivity. Many small companies are now providing RoR clouds (many layered on top of Amazon).

Use Ruby on Rails if you

- Are building request-response web apps with existing Ruby expertise
- Consider time-to-market critical
- Aren't doing anything fancy (installing software)
- Don't care about lock-in

Lock-in isn't a big concern with RoR because, as we've said, there are many choices of RoR vendors and probably more to come.

2.3.5 Force.com PaaS cloud

Force.com is an extension of the SaaS service Salesforce.com. Many companies have been using Salesforce for a long time. They have rich, sophisticated databases of sales contacts, history of sales cycles, information about their products, and a lot of other sales-process related information. This information forms the crown jewels of any company's sales team, and companies want many applications that aren't built into Salesforce.com. For this reason, Salesforce.com created a framework using many of the same back-end services used by the company's main SaaS application, operating on the same back-end data, and made it accessible and programmable to end users. Force.com is ideal for building applications that tie into your existing Salesforce.com databases, such as sales contacts, the internal sales team, your products, and so on.

Use Force.com if you

- Are already a customer of Salesforce.com's SaaS customer-resource-management product
- Have a requirement for a simple mashup style of web application
- Are willing to use Force.com's specialized programming language
- Don't care about lock-in

We didn't include a section about when to use private cloud because it's a much more complex discussion. We'll deal with the subject in chapter 4.