A01 : Cloud Computing and Architecture

Network Application Development

Verdi Rodrigues-Diamond

September 30th, 2014

1. An evaluation of which Cloud paradigm would best be used (SaaS, PaaS, or IaaS) for this system.

A platform as a service solution is a clear choice for streamlining your IT budget. A platform as a service solution removes the need for your company to manage physical servers or even time consuming configurations.

With platform as a service, it’s there job to keep the server lights blinking. They provide the server space and stack to run your application. Server security is also managed by them meaning that your IT team doesn’t have to run around patching software and your provider manages the risk if they go down.

One downside to using a platform as a service provider is that, to deploy your application, most require that you use some form of version control to upload your code. While version control is great for many reasons, it does mean however that your provider will get a full copy of your entire code base and every change that has ever been made to it.

Most platform as a service providers are designed for a specific to a certain technology. For example, the ones listed below are all designed for making web based platforms. If you’re doing something with different requirements (for example high bit rate video streaming) you may want a more custom solution such as IaaS.

1. For the cloud paradigm selected, a comparison of three (3) vendors’ public cloud offerings – include cost, capability, and capacity comparisons as appropriate.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Heroku | OpenShift | Engine Yard |
| Cost | Pricing is split up based on three core components: Dynos – computing power that runs your app; Heroku Postgres – Data storage; and support | OpenShift’s pricing model is similar to Heroku in that it is divided up into processing units and database units. With OpenShift, processing units are called gears. Storage is given on a per gear basis rather than splitting it up into a separate dyno of its own. | Engine Yard allows you to rent PaaS services for a server by the hour or by the month. You can also vary the capacity of the server by renting more or less powerful servers. |
| Capability | Heroku is capable of hosting applications running on a variety of different platforms. Platforms include Ruby, PHP, Python, Node.JS, Java, Closure, and Scala.  Heroku also has a large inventory of installable plugins which allow you to extend the capability of your applications. For example, if you don’t want to use Postgres, | OpenShift will run your regular set of platforms and tools. OpenShift will also let you brew up your own setup if that doesn’t work for you. | Engine Yard provides extra functionality in this space because, while being a PaaS provider, they also do more custom setups as well |
| Capacity | Heroku limits you depending on how many dynos you have available. Dynos are however instantly available for an additional cost. | OpenShift, much like Heroku, is infinitely scalable. | With Engine Yard, like every other provider this space, is infinitely scalable. |

1. Present a clear case for using one of the providers listed above.

Heroku is one of the oldest and most reliable platform as a service providers on the market. Being that they’ve been around for a while, they have a plethora of documentation around to assist your devops staff. Heroku uses common platforms for publishing code and, while they do use some proprietary apps to help you manage your account, you are not in any way locked to this provider.

1. Propose a feasible architecture for the system given the selected provider. Consider standard architecture patterns in your development. If there are critical alternatives, list them. Be sure the architecture satisfies the customer’s goals.

Platform as a service solutions typically follow threeish tier topology. What I mean by that is that most three tier architectures follow the following model.

Platform as a service providers use a variety of techniques to provide maximum availability such as clustering application servers and database servers as well as load balancing. These techniques are all designed to provide you a three tier like topology with much improved uptime and availability.

Platform as a service providers generally succumb to one common fatal flaw – they don’t work offline. This however doesn’t mean the end of the road. One technique to overcome this is to program your application to store your data locally while offline and then replicate it next time you go online. The downside however is that any maintenance records generated in one facility will not be viewable by any other facility until it comes back online again.

One could also lessen the risk by putting in place an ultra low down time internet connection such as a dedicated internet plan or private fiber.

1. Since the CTO is particularly interested in the feasibility of this project, but wishes to spend as little money as possible to get things off the ground, sign up for a free tier version of one of the providers (need not be the specific one selected above). Set up and deploy a quick start, hello world application, or some other proof of concept that will demonstrate to the CEO and CTO that you at least know the firsts steps in getting these things rolling.
2. Provide a clear and concise, step-by-step document outlining the sign up and set up procedure listed above.
   1. Navigate to the Heroku website. – <http://heroku.com>
   2. Register an account with Heroku
   3. Set up a virtual instance using the ‘+’ in the top right of the user control panel
   4. Install git on your computer (either Git for Windows or Github for Windows will work)
   5. Clone the remote Heroku repository to your local machine
   6. Make some modifications to make it your own hello world
   7. Add and commit your new code to the git repository
   8. Push the new commit to the remote Heroku repository
   9. Browse to your instance and view your changes