Andreas Leicher Follow
CTO @mapcase #rails, #golang, #agile, #startups, #devops, #cloud, #mountainbike, #snowboard...
Jan 28, 2015 · 5 min read

Should I use Heroku or AWS?

And what about hosting your own machines?

Everyone running web projects, no matter whether it is a pure API, or a full fledged web application, will come to the point of having to host their service. After (carefully) selecting your software stack (Ruby on Rails vs. Node.js vs. Django vs. etc.) you need to make your beautifully crafted application available to the world. There are multiple options available today, and while this overview is not complete, I hope it gives you a first idea on some of the options.

I will provide a short overview on Heroku, which offers Platform as a Service, Amazon's AWS, offering Infrastructure as a Service, and hosting your own machines. This list is neither complete, nor does it represent a specific recommendation of the one over the other, consider it a personal opinion based on my own experience.

IaaS vs. PaaS

Before anyone jumps in saying, that this is comparing apples to oranges, yes, you are right. Infrastructure as a Service, IaaS for short, is a type of cloud computing, where the provider (AWS in this case) offers computing resources (virtualized) over the internet. Think of it as renting a machine somewhere, without taking care of the wires, electricity, hardware, etc.

Platform as a Service, PaaS as provided for example by heroku, also handles the infrastrcuture operations for you. So you don't need to worry about the different components to build up your infrastructure (load balancers, machines, etc.) but can instead very often deploy your whole application with a simple git push.

Amazon AWS vs. Heroku

So if you are willing to have parts of your server infrastructure outsourced to another company, read on. If not, skip to the end, to check my recommendations if you run your own servers.

AWS

As mentioned, AWS requires you to take care of setting up your load balancer(s), installing the right software stack on your EC2 instances, configuring your databases, and so on. You also should implement and configure your deployment process, e.g. by using something like capistrano.

To maintain the software on your instances, I highly recommend to not install software manually on your instances, which might lead to different configurations on different machines, etc. My recommendation is to use a proper provisioning toolchain, like Puppet, Chef, or Urknall from the great team at dynport in Hamburg. Urknall is a Go based 'Opinionated provisioning tool for clever developers'. In addition (and recommended by AWS cloud architects) you should take a serious look at AWS Cloudformation. Cloudformation allows you to write templates to describe your infrastructure at AWS. So with Cloudformation you can provide a proper documentation of your infrastrcuture in code, and make sure that you can bring it up with exactly the same configuration you did initially.

The cost of running AWS is less than on heroku, as you are 'only' paying for the infrastructure. Depending on the instance type, prices start as low as $0.014 per hour for a t2.micro instance or $0.056 per hour for the production ready t2.medium instance.

Heroku

Heroku takes away all the pain of installing software, maintaining it, monitoring the software for required updates, setting up your deployments, etc. Sounds like magic? Feels a bit like it, when you use it for the first time. No wonder that heroku is suggested in a lot of tutorials. I also recommend it in my lectures on web application development. It just works and allows you to quickly get your web application online for the world to see and play with it. And they offer a free tier as well, making it great to just test your app.

Need more performance? Need a larger Postgres database? Want to add Redis to your stack? Most of the software which requires you to setup a new instance, provision it, configure it, etc. on AWS can be added on heroku with a click or via the command line. Scaling your performance is just a matter of adding more of the so called dynos.

So why isn't everyone using heroku then? As always, there are mutliple reasons. One is cost. As you get all the platform and software as a service, you also pay for that. So you should calculate the extra

cost in case that your infrastructure grows, and compare that to the cost of setting up your AWS infrastructure.

Also some people might feel uncomfortable with giving heroku their code, and sharing it with others on the same instance (isolated with heroku's dyno isolation).

Running your own?

Not every application is suitable for running at a cloud provider. Sometimes there might be legal concerns, or requirements, that force you to run your own datacenters, host your own servers, etc. It might also look cheaper at first sight, but you have to consider the extra effort for administration of the machines. That 2 year old harddisk stops working on a Sunday morning? You are responsible for getting the system up and running and hopefully can find a replacement disk which you can then integrate into the RAID, etc. With AWS or heroku, things like this are completely hidden and you can enjoy your breakfast with your family on that same Sunday morning.

So what?

Use heroku:

- for small personal projects that you want to show the world
- as a freelancer to show off your portfolio as working apps (most of the time the free dyno plan is enough)
- if you are an agency, need to focus on delivering a (web) application, and promise the customer to take full care of the operations and do not have a specialist on board
- if you need to get started very quickly and do not want to have the overhead of managing systems and do not need the full freedom to do so
- if you are a student/learning a new framework and just want to play around

Use AWS:

- if you are an agency and you can afford to have an expert in devops
- for production load with SLA's

- if you need additional control over your infrastructure, such as software versions, specific patches, etc.
- if you can afford to spend some time on devops

Both, Amazon AWS and Heroku have signed the US/EU Safe Harbor agreement on data privacy. Here you'll find Amazon's privacy statement and here is Heroku's privacy statement.

Use hosted servers or your own infrastructure:

- if you can afford a good amount of time on devops and administration
- if you are legally required to do so
- if you need to get the very last bit of performance out of the hardware
- if you want to sit in a server room, tearing your hair on a Sunday morning

Note: this post has been edited on Feb, 3rd 2015, to reflect the comments from readers Dan Peterson and Christian Weyer. Thanks for your input!

Further reading:

https://stackoverflow.com/questions/9802259/why-do-people-use-heroku-when-aws-is-present-whats-distinguishing-about-heroku