2 What kinds of hosting does the market offer?

If you've spent at least 30 minutes searching for hosting solutions, you'll have noticed that there are different types of hosting: *PaaS*, *shared*, *VPS*, *dedicated*, *colocation*, *cloud*.

The hosting market tends to segment itself in those 5-6 categories: some of them have a clear definition, while others (like *cloud*) have so many different meanings that I've had to come up with my own definition for the book.

2.1 Shared hosting

On a shared hosting account you get a directory where you can put your static files (HTML, images) and an interpreter for your dynamic pages (in 98% of the cases PHP): the server where your files live is also home to hundreds, if not thousands, of similar websites. You share with the other websites the Internet

bandwidth to and from the datacenter, the CPU, the memory and the disk I/O of the server.

This is the most limited and cheapest form of hosting available on the market. The main advantage is that you don't have to perform any system administration tasks such as configuring the OS, the network, installing applications, and so on.

Few providers of shared hosting give you the ability of logging in to the actual server with SSH: you get instead a web interface called a *control panel* which translates clicks to actions on the system. Installing applications is often a one-click process, but you can choose only from a fixed list of the applications made available by the hosting company. More often than not, the hosting company doesn't allow you to use tools to automate common tasks, such as deployments and backups.

A notable exception is WebFaction: you get shell access via SSH, the ability to compile and install your own binaries and the ability of running long-running processes.

The hosting company configures the server to allow a certain amount of resources to each account, and that in turn means that your account is as limited as any other. Once you start getting *too much* traffic the host may force you to move to a higher package or suspend your account altogether. Unfortunately, the definition of *too much* isn't always clearly spelled out.

My advice: Shared hosting is for web pages, not for apps. Though the cost might seem attractive at first, there's no guarantee that you will be able to deploy your code and even if you do, the resource limitations may kick in at the very moment when your app starts to get some real traffic. Automation, in the form of a shell access, is so crippled to be useless on the vast majority of providers. Don't put anything more complex than a blog on shared hosting.

2.2 VPS (Virtual Private Server)

When you buy a *VPS* you get a slice of a physical server, on which other virtual machines co-exist. On a VPS:

- you get root access
- you can install your own software
- you can configure the VPS how you want

The effort to perform system administration tasks on a VPS is pretty much equivalent to the effort required to administer a physical server. However, you don't have to worry about the hardware details, because the hosting company will replace a failing disk drive or network card. Thanks to fierce competition driving down prices, buying a low-end VPS from a reputable, established

company costs around $5 per month, and many of those companies also offer hourly billing for temporary surges in traffic and quick experiments.

Some companies offering VPS:

- Digital Ocean
- Vultr
- Linode

You can usually get more performance out of a VPS by asking for more CPU, RAM or disk space (*scaling up*) and paying an increased fee. Resizing down is a little more difficult and not always possible: for example, to scale down at Digital Ocean you would have to purchase a second, smaller VPS, migrate everything over and shut down the original VPS, thus getting a new IP address and suffering the DNS propagation delays.

For most web applications, a VPS is flexible enough to accommodate custom binaries and particular configurations; depending on the hosting company and on the VPS plan you choose, you may never need to move to dedicated servers and just order more CPU and memory to respond to increased traffic.

On a VPS you're still sharing some hardware resources: the network, the I/O bandwidth and the CPU: if your VPS demands too much of any of such resources, the hosting company may force you to move to a different, more expensive plan, or in the extreme case kick you out altogether. You may also see the other side of this problem: your app's performance degrades at random

times and without an apparent cause. Perhaps another VPS on the same hardware node is using too much CPU or bandwidth, *stealing* resources from your app.

Scaling up a VPS typically involves stopping it, increasing the resource allocation and restarting it, which is not something that you want to do at peak traffic. If your application allows it, you should rather provision one or more new VPS to work alongside the existing ones (*scale out*).

With most VPS providers you have to choose from a fixed set of VPS configuration and you have no way of getting more RAM without also getting more CPU cores and disk space.

> **My advice:** a VPS is a good solution for a staging environment, and a small fleet of VPS can also be a good starting point for a closed beta of your app; depending on your traffic levels, it may be a cost-effective solution for continued production usage as well.

Choose a VPS provider matching the following criteria:

- support of KVM virtualization, which allows you to run any operating system

- a true private network between multiple VPS in the same account, to ensure the confidentiality of the data exchange between the various components of your app
- some kind of out-of-band console to access the VPS if something goes wrong

Do not spend more than $50 per month on any single VPS: at this price point, a dedicated server is a much better value for money.

2.3 Dedicated Servers

When you buy a *dedicated server* you get a physical server whose hardware resources are entirely for you:

- you get root access
- you can install your own software
- you can sometimes modify the hardware configuration by asking the hosting company for additional disks or RAM

The hosting company is responsible for providing power and network, and will replace a failing component at no additional cost if you request it. Noisy neighbors that consume CPU and disk I/O are not an issue, your app can utilize all of the CPU and the hardware resources of the machine.

Some companies offering dedicated servers:

- Hetzner
- OVH
- LeaseWeb
- SoftLayer

The hardware configuration is usually fixed, determined when you pay for the setup of the machine and very rarely changes. To scale up you typically order a more powerful machine, install your application and move your data. The monthly cost of renting a dedicated server is predictable, and renting allows you to migrate up (or down) without making initial investments and depreciation calculations.

> **My advice:** a dedicated server is good when you outgrow a VPS and you need more power for a price that isn't much higher.

Look for providers that:

- offer a VLAN per customer or a true private network between the servers in the same account
- offer month-to-month payments to avoid locking yourself into multi-month contracts

- offer a virtual console for out-of-band access to the servers and installing your own OS

2.4 Colocation

You buy your own machines, then send them to a datacenter which provides power and connectivity. This is almost like renting dedicated servers, except that the machines are your property and fall under your responsibility: when some hardware component breaks, you travel to the datacenter and replace it, or you have a datacenter technician replace it with the spare part that you had sent originally along with the servers.

The main factors that determine the prices of colocation are:

- how much space (height) that your servers will occupy in the rack, measured in *units* (a 1U server will cost less than a 2U or 4U server)
- the power that your servers will draw
- the bandwidth

My advice: you should buy and colocate machines only when your business is well established and can afford the capital expense; over a lifespan of 2-3 years colocation can be cheaper than

renting dedicated servers, even if you factor in the hardware replacements and the labor.

2.5 Cloud

There are as many definitions of *cloud* as there are hosting companies out there. I will limit the discussion to the *IaaS (Infrastructure as a Service)* variant, where you rent some computing resources (CPU, RAM, storage, network) from a hosting company: in general those computing resources come in the form of virtual machines, similar to a VPS. Some companies exploit this similarity to define their services as *cloud servers* where in reality they simply offer VPS with hourly billing.

Let's define what we want from a cloud hosting solution:

- billing by the hour (or by the minute) of the VM and the other resources (storage, network services)
- ability to obtain more resources quickly (in minutes rather than days)
- ability to resize up and down the existing VM
- networked storage for the VM volumes
- network load balancers and firewalls are available as a service

Some companies offering cloud hosting:

- Amazon AWS
- Google Cloud Platform
- Microsoft Azure
- Dediserve
- ProfitBricks
- LunaCloud
- SoftLayer

With a cloud server you get root access and you can install your own software. The similarities between cloud providers end there, because each one has a particular service model:

- some providers offer a predefined set of virtual machine configurations and you cannot add memory without also having to buy more CPU capacity; other providers let you mix and match resources, so you may have a virtual machine with 8 CPU cores but only 512MB of RAM
- some providers don't install the stock version of a operating system, but a special version that they have customized to better work in their own environment (Amazon calls this an AMI - Amazon Machine Image)

- your virtual machines may come with a local disk and networked storage offered as an option, or there may be networked storage only and no local disk at all
- after a reboot your virtual machine may be on a different hardware node, which means that you don't have any of the data stored on a local disk

The list of differences grows each day, and with cloud providers introducing new features at a regular (and fast!) pace is almost impossible to say that one provider is better than another one.

The only way to compare cloud providers is to focus on your particular use case and try to come up with realistic, if pessimistic, estimates about your traffic and your app's usage of computing resources.

Cloud providers offer many additional services, like:

- object storage: store files using a web services interface
- block storage: persistent networked storage that you can mount on your virtual machines like a filesystem

- network load balancers: distribute traffic across the various instances of your app

The biggest difference with dedicated or VPS hosting is that *cloud hosting pricing is based on usage*, so the more resources you use, the higher your bill will be: you may very well pay $10 the first month and $1000 the second month. The definition of *usage* depends on the particular cloud provider and technology: for example, Amazon AWS has a block storage offering (EBS) which is billed according not only to how much space you provision, but also to how many I/O operations occur [1].

Estimating usage-based pricing is very difficult, especially with a new application which hasn't experienced enough traffic to let you see patterns and volumes. It is critical to keep a close and regular watch on usage and set up billing alerts to make sure you don't get huge bills by surprise.

Cloud hosting tries to abstract away the underlying hardware with the intent of making applications more reliable. Techniques such as *design for failure* encourage you to take responsibility of your application's availability in your own code and in your own automation tools: the infrastructure may fail but your application stays online, perhaps in another data center, with a different IP address, on a database which just minutes ago was only a replica. Cloud hosting providers facilitate this design technique with services such as message queues, distributed object storage or managed databases with automated replicas and failover.

[1] Amazon EBS pricing

All of those services carry a significant complexity, and sometimes even the cloud providers have troubles with their own creations failing in totally unexpected ways. Two notable examples:

- AWS' EBS product has a significant failure in October 2012 [2] which propagated to several other component, like the Elastic Load Balancers which were affected because the configuration was stored on EBS volumes that were hung
- AWS S3, which previously enjoyed years of flawless operations, had an important failure in February 2017 [3] where a couple of critical subsystems had to be restarted, taking down so many other services that even the AWS status page went down

The takeaway from this kind of event is that you can mitigate the impact of cloud failures by deploying your app in multiple zones or even with multiple providers, but this has a very high cost in both complexity and money.

> **My advice**: cloud hosting makes sense only if your app really needs the particular features offered: hosting a single VM on Amazon EC2 makes very little sense, hosting an entire infrastructure to take advantage of the load balancers, the VPC, the scalable storage and all of the other services definitely does. If you're expecting to

[2] EBS October 2012 failure

[3] S3 February 2017 failure

have big variations in your resource load, then the cloud is a good answer since it allows you to keep costs down in the off-peak periods and to quickly provision resources in the 5-10% of the time when you really need them. On the other hand, if your needs are relatively stable, it will be much simpler and much cheaper to rent a few dedicated servers with a reputable provider.

I don't have a preference for cloud providers, though AWS seems to be the largest and well-known provider out there: if you choose to host with AWS you will undoubtedly find qualified help almost everywhere on the planet.

2.6 PaaS

PaaS stands for *Platform as a Service*: you get to run your code on an application, rather than on an infrastructure. Your code has to obey the various conventions and rules set by the application platform, you have to use some specific PaaS tools to deploy your code, and in general you don't have access to the underlying computing resources. On a PaaS you cannot install custom software, for example a specific version of HAProxy, but for an additional fee you can choose some *add-ons* in a list pre-configured by the platform. Since you don't get direct access to the operating system, you cannot tune the host's configuration, nor the storage settings of your database: the assumption is that

the provided configuration works well enough for the kind of applications that the PaaS supports.

Some examples of PaaS are:

- Heroku
- AWS Elastic Beanstalk
- Google App Engine
- Engine Yard

We cannot directly compare a PaaS with VPS or dedicated hosting, since a PaaS comes with additional services included in the price. It would be better to compare PaaS with *managed hosting*, since both provide a collection of services in addition to the simple usage of the computing and storage resources. For example, to add a SSL certificate in a managed hosting offer you would open a ticket and instruct the technicians on the precise actions to perform; in a PaaS you would click on some buttons in the control panel and the PaaS automation performs the actual actions.

The additional services and pre-configured automation make for the relatively high prices of a PaaS solution. For the same amount of money that you can spend on the bare minimum Heroku configuration, you would get several times the RAM and the CPU on any reputable VPS provider. The two aren't really comparable however, since on a VPS you would have to install, configure and administer the machine by yourself.

A PaaS doesn't only give you conventions, it also forces its constraints on your development: for example, forbidding file system access means no logging and no data persistence except than in databases. It may work well if you develop around those constraints and if you think that there are better uses for your time that administering a server; however, I wouldn't try to run or retrofit an existing application to work on a PaaS.

> **My advice:** use a PaaS to build a prototype of your app and don't lock yourself into the PaaS' way of doing things, because eventually you will need a finer control over the components that a PaaS wants to manage for you.

2.7 Serverless (FaaS)

Serverless is a new form of hosting, something in between cloud and PaaS. A better name is *Function as a Service (FaaS)*: developers break down their app into functions, which they then deploy on a platform able to execute them according to some specific triggers (for example a web request, a file upload, or a scheduled event).

In a Serverless architecture the developers and operations people do not interact with the computing hardware or the operating system, but simply specify the type and quantity of resources needed by the various functions. The code is written according to the API of the Serverless provider, and typically ends up running on a container; the developer can configure the container instance to take some amount of memory and to run for no more than X minutes.

The main benefit of a Serverless architecture is the minimization of all tasks related to system and network administration: you simply do not have access to the underlying OS and network, so this category of tasks disappears almost entirely. Another benefit is the fine granularity of the billing: providers usually bill you for the actual time and resources used by your functions, which can be very small compared to the hourly billing of cloud instances.

The inconvenients of Serverless frameworks are mostly due to the relative immaturity of the tools and the practices:

- a function's invocation cannot take more than a set amount of time, long-running processes aren't possible
- it can be difficult or impossible to understand why something goes wrong when you don't have access to tools like debuggers and profilers
- persisting state and data requires other services (think AWS S3 or RDS), which can increase the vendor lock-in even further
- securing a FaaS application is at the same time simpler (less moving parts) and more complex (no control on anything besides your code)

My advice: consider FaaS for event-driven applications, or batch tasks. If you're able to properly architect your functions, there are significant scalability and economy benefits to reap.

2.8 Managed hosting

I've decided to write a section about managed hosting even if it doesn't fit in with the other types of hosting above. You could get a managed VPS, a managed dedicated server, or even a managed cloud hosting: in all cases, you buy some computing resources and a management service from the provider. This means that the provider is responsible for the infrastructure, from providing power and network to installing the OS, the applications, managing the backups, the security updates, etc.

What actually a provider does in a managed hosting offer changes from one provider to another. You may not be given root access, because you could undo the various configurations the provider has put in place, and it may not be possible to install software which is not in a list sanctioned by the provider. You may be forced to re-architect your deployment scripts and where you log diagnostic output to conform to the provider's conventions. Generally you communicate with the provider by way of tickets, in which you ask for some

action to be performed: depending on the provider, you could give just general instructions (e.g. “install the latest stable version of nginx”) or be forced to spell out the most minute details (e.g. “open a ssh session to the server XYZ, type ‘sudo yum install nginx’ and press

It’s important to stress out that a managed hosting provider won’t know anything about your app and what makes it special; typically you can find engineers who can configure, tune and troubleshoot Apache and MySQL and that’s more or less about it. If you’ve developed a Node application that talks to a MongoDB database you’ll have a much harder time finding a managed hosting provider capable to help you. Managed hosting is about you adapting your app to work with the host, and not the other way round (as it should).

> **My advice:** if you have even a little system administration experience, just get VPS or dedicated servers. If you don’t, consider a PaaS. Even if you find a managed provider that can help you with your app, you’ll waste so much time in miscommunications and misunderstanding that it won’t be worth it.
