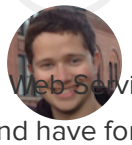
AWS: the good, the bad and the ugly

Posted on December 18, 2012 by seldo



Here at awe.sm we have been hosted from the beginning on Amazon Web Services (AWS). Over the past 3 years we've learned a lot about what it's good at, what it's not so good at, and have formulated our own set of best practices for running a high-availability, high-performance system, which differ in some important ways from Amazon's own advice.

We're going to talk about two related things:

1. For people who have heard of AWS but haven't started using it yet, we thought we'd lay out both the **benefits and the challenges** we've encountered.
2. For those already using AWS, we go into some detail of the **best practices** we've picked up for running a high-performance service like ours, where uptime is one of the highest priorities.

It's not an exaggeration to say that AWS has radically changed the economics of running a technology startup, but so slowly and gradually that it crept up on the industry. Nobody realizes how many companies are using Amazon's Elastic Compute Cloud (EC2) somewhere in their stack until it has an outage, and suddenly it seems like half the Internet goes away. But it's not like Amazon got lucky: they have an awesome product. Everybody uses AWS because EC2 has radically simplified running software, by hugely lowering the amount you need to know about hardware in order to do so, and the amount of money you need to get started.

EC2 is a new way of running software

The first and most important thing to know about EC2 is that it is not merely a virtualized hosting service. A better way of thinking about it is as employing a **fractional system and network administrator**: instead of employing one very expensive person to do a whole lot of automation for you, you instead pay a little bit more for every box you own, and you have whole classes of problems abstracted away. Power and network topology, hardware costs and vendor differences, network storage systems — these are things you used to have to think about back in 2004 (or regretted not thinking about). With AWS and its growing crop of competitors, you no longer do — or at least not until you get much bigger.

Far and away, the biggest difference and advantage of using EC2 is **flexibility**. We can spin up a new box very, very quickly — about 5 minutes from thinking "I need some hardware" to logging into the shell for the first time, ready to go. This lets us do some things that just a few years ago would have been crazy, for example:

as easy as resetting the load balancer, and going forward with the new system means just shutting down the old boxes. Running twice as much hardware as you need, but just for 24 hours, is cheap and simple.

- our failure plan for some non-critical systems, where perhaps up to an hour of occasional downtime is acceptable, is to monitor the box, and if it fails, **spin up a new box and restore the system** from backups
- we can **scale up in response to load events, rather than in advance of them**: when your monitoring detects high load, you can spin up additional capacity, and it can be ready in time to handle the *current* load event — not the next one.
- we can **not worry about pre-launch capacity calculations**: we spin up what seems at a gut level like enough hardware, launch, and then if we find we've got it wrong, spin boxes up or down as necessary. This kind of iteration at the hardware level is one of the greatest features of AWS, and is only possible because they can provision (and de-provision) instances near-instantly.

EC2 makes strong financial sense for startups

The most obvious cost advantage of AWS is that it has **literally zero setup costs**: you use the same Amazon account you use to order random junk off the Internet, click a button, and start paying for servers, by the hour. You only pay for the boxes when they run, and you only pay for storage that's actually in use, so your startup costs are minimal, and it **encourages experimentation at the hardware level**: spin up 10x more capacity than you need, run load tests, and then spin them back down until you really need them. That's not just convenient, that's revolutionary. As with many other aspects of AWS, such a large quantitative difference becomes a qualitative one.

As I've already alluded to, AWS also **dramatically lowers operational overhead**. Until 2012, more than two years after we started the company, we had no dedicated operational staff at all. This was in retrospect a bad call — we should have hired one in 2011, maybe even earlier. But we still have only one full-time ops guy managing our whole fleet of over a hundred boxes. That's a pretty great ratio. The multiplier effect of (mostly) not needing to care about networking, power, etc. etc. is enormous and, once you've got used to it, under-appreciated.

Of course, it's not a total home-run. AWS is definitely more expensive than vanilla colo hosting. But they do a lot to take the sting out of that by routinely lowering their prices — 18% in October, 10% in March just this year. You should also factor in the money you save by having the ability to forgo running spare hardware in favor of spinning it up on demand. Finally, for long-term use, you can pay some money up-front and save a ton of money — over 50% — by using reserved instances. (At awe.sm, we are kind of crazy about reliability, so we run a lot of redundant hardware, and reserved instances have been a big win for us)

But EC2 has some problems

This is where the love letter ends, and the Amazon rep who was prepping a pull-quote for their front page starts to reconsider. While we love EC2 and couldn't have got where we are without it, it's important to be honest that not everything is sunshine and roses. EC2 has serious performance and reliability limitations that it's important to be aware of, and build into your planning.

co-located, but (in theory) isolated from each other in terms of networking, power, etc.. There's a few important things we've learned about this region-zone pattern:

1. **Virtual hardware doesn't last as long as real hardware**. Our average observed lifetime for a virtual machine on EC2 over the last 3 years has been about 200 days. After that, the chances of it being "retired" rise hugely. And Amazon's "retirement" process is unpredictable: sometime they'll notify you ten days in advance that a box is going to be shut down; sometimes the retirement notification email arrives 2 hours after the box has already failed. Rapidly-failing hardware is not too big a deal — it's easy to spin up fresh hardware, after all — but it's important to be aware of it, and invest in deployment automation early, to limit the amount of time you need to burn replacing boxes all the time.

2. **You need to be in more than one zone, and redundant across zones**. It's been our experience that **you are more likely to lose an entire zone than to lose an individual box**. So when you're planning failure scenarios, having a master and a slave in the same zone is as useless as having no slave at all — if you've lost the master, it's probably because that zone is unavailable. And if your system has a single point of failure, your replacement plan cannot rely on being able to retrieve backups or configuration information from the "dead" box — if the zone is unavailable, you won't be able to even see the box, far less retrieve data.

3. **Multi-zone failures happen, so if you can afford it, go multi-region too**. US-east, the most popular (because oldest and cheapest) AWS region, had region-wide failures in June 2012, in March 2012, and most spectacularly in April 2011, which was nicknamed cloudpocalypse. Our take on this — and we're probably making no friends at AWS saying so — is that AWS region-wide instability seem to frequently have the same root cause, which brings me to our next point.

To maintain high uptime, we have stopped trusting EBS

This is where we differ sharply from Amazon's marketing and best-practices advice. Elastic Block Store (EBS) is fundamental to the way AWS expects you to use EC2: it wants you to host all your data on EBS volumes, and when instances fail, you can switch the EBS volume over to the new hardware, in no time and with no fuss. It wants you to use EBS snapshots for database backup and restoration. It wants you to host the operating system itself on EBS, known as "EBS-backed instances". In our admittedly anecdotal experience, EBS presented us with several major challenges:

- **I/O rates on EBS volumes are poor**. I/O rates on virtualized hardware will necessarily suck relative to bare metal, but in our experience EBS has been significantly worse than local drives on the virtual host (what Amazon calls "ephemeral storage"). EBS volumes are essentially network drives, and have all the performance you would expect of a network drive — i.e. not great. AWS have attempted to address this with provisioned IOPS, which are essentially higher-performance EBS volumes, but they're expensive enough to be an unattractive trade-off.

- **EBS fails at the region level, not on a per-volume basis**. In our experience, EBS has had two modes of behaviour: all volumes operational, or all volumes unavailable. Of the three region-wide EC2 failures in us-east that I mentioned earlier, two were related to EBS issues cascading out of one zone into the others. If your

- **The failure mode of EBS on Ubuntu is extremely severe**: because EBS volumes are network drives masquerading as block devices, they break abstractions in the Linux operating system. This has led to really terrible failure scenarios for us, where a failing EBS volume causes an entire box to lock up, leaving it inaccessible and affecting even operations that don't have any direct requirement of disk activity.

For these reasons, and our strong focus on uptime, **we abandoned EBS entirely**, starting about six months ago, at some considerable cost in operational complexity (mostly around how we do backups and restores). So far, it has been absolutely worth it in terms of observed external uptime.

Beware: other AWS services rely on EBS

Because some AWS value-added services are built on EBS, they fail when EBS fails. This is true of Elastic Load Balancer (ELB), Relational Database Service (RDS), Elastic Beanstalk and others. And EBS — in our experience — seems to nearly always lie at the core of major failures at Amazon. So if EBS fails and you need to suddenly balance traffic to another region, you can't — because your load balancer also runs on EBS. And you can't launch new hardware anyway, because the console app that AWS provides to launch it also runs on EBS. So we love EC2 (and really really love S3), but **we don't use any other AWS value-added services**. A side-benefit of this approach is that we can switch relatively simply to other hosting providers, and are not locked too closely to AWS.

Lessons learned

If we were starting awe.sm again tomorrow, I would use AWS without thinking twice. For a startup with a small team and a small budget that needs to react quickly, AWS is a no-brainer. Similar IaaS providers like Joyent and Rackspace are catching up, though: we have good friends at both those companies, and are looking forward to working with them. As we grow from over 100 to over 1000 boxes **it's going to be necessary to diversify to those other providers**, and/or somebody like Carpathia who use AWS Direct Connect to provide hosting that has extremely low latency to AWS, making a hybrid stack easier.

I hope this has been of use to you. If your company needs social analytics — and it probably does, even if you don't know it — you should check us out. And if you want to join our crazy crew, we're always hiring good people.

This entry was posted in Engineering. Bookmark the permalink.

59 Responses to *AWS: the good, the bad and the ugly*

richtaur *says:*
December 18, 2012 at 12:37 pm

Excellent article, Laurie! We're using AWS for all of our web stuff these days too, so good to know where the trouble areas are.

Brian Whalley *says:*
December 18, 2012 at 12:41 pm

Joey Wilhelm *says:*
December 18, 2012 at 2:45 pm

This is a very insightful article, thank you. I am curious though, as I have just made a push for our (pre-launch) company to move onto ELB, what are you using as an alternative to ELB? Is it as fully featured and automated?

Steve Phillips *says:*
December 18, 2012 at 10:25 pm

Do add to the question I'm replying to... Does awe.sm use HAProxy instead of ELB? Is it more expensive for similar performance? Thanks!

seldo *says:*
December 19, 2012 at 1:24 pm

We are using an awful lot of HAProxy. We are planning a follow-up post in the next few weeks with more technical details of our setup.

Eugene OZ *says:*
December 18, 2012 at 2:48 pm

Thank you very much for this article. I hope Amazone will take more care about EBS after this article
I personally love Linode very much – try it, they are awesome.
For me performance of AWS small/medium boxes is not acceptable.

Jake McGraw *says:*
December 18, 2012 at 3:09 pm

Since this wasn't explicitly published here I posed the question about which AWS features are EBS backed on Server Fault:

http://serverfault.com/questions/459197/which-aws-features-are-ebs-backed

Noah Yetter (@angrynoah) *says:*
December 18, 2012 at 5:03 pm

"AWS have attempted to address this with provisioned IOPS, which are essentially higher-performance EBS volumes, but they're expensive enough to be an unattractive trade-off."

Not sure how you arrived at that, Provisioned IOPS is extremely cheap.

seldo *says:*
December 19, 2012 at 1:30 pm

We saw the choice as "pay more for better performance than EBS" (provisioned IOPS) vs. "pay less for better performance than EBS" (instance storage). Combined with our very high level of concern with the reliability of EBS, instance (aka ephemeral) storage seems the better option to us.

Sayed *says:*

with in the stead of EBS?

Mason Browne *says:*
December 18, 2012 at 6:13 pm

I'm curious – where did you read that the AWS Console depends on EBS?

seldo *says:*
December 19, 2012 at 1:33 pm

I can't find a published source — this may have been something our AWS rep told us directly. My apologies, I tried to stick to publicly-available facts and may have slipped up here.

Tariq *says:*
December 27, 2012 at 12:38 pm

seldo: I don't think that's a fact.

John Riccardi *says:*
December 18, 2012 at 6:17 pm

Great writeup. I have heard a lot of rumblings about AWS, both pro and con, but since I haven't personally used it I didn't understand the major issues until reading this article.

juber *says:*
December 18, 2012 at 6:38 pm

thanks for the insights. If you don't use EBS, do you copy all the required data to the local drive when the machine is coming up? what happens to the modified data if the machine crashes? Could you elaborate on this a little bit? Also, have you tried using the cluster compute machines with hardware virtualization and 10 gigabit etherenet? thanks.

Todd *says:*
December 19, 2012 at 7:48 am

I would love to learn more about what they used in lieu of EBS.

seldo *says:*
December 19, 2012 at 1:33 pm

A follow-up post is coming in the next few weeks

Gnep *says:*
December 24, 2012 at 7:37 pm

Your application architecture need to be stateless, meaning that any instance could be rebuilt in minutes and all your critical data need to be persisted in things like S3.

Peter Hancock *says:*
December 18, 2012 at 7:19 pm

– The command line tools are zone/region agnostic, and should be used over the console anyway. (Which means you could pull down your backup into a different region)
– If you provision all your boxes via scripts, it makes it significantly easier to pull out a retired AMI and bring on a new one. (Which you should be doing anyway as OS upgrades come through)
– If pulling out the AMI is still a concern, you can create your own and never retire it. We've got one that's been running for over 500 days now.

But when EBS fails (and seems to in US-EAST) – it's ugly. A little more work around reliability in that would be good. I think that region got a "Friday car".

Mason Browne *says:*
December 24, 2012 at 12:53 am

I'm curious – in your experience, what were some particularly rough shortcomings with the AWS Console that you were able to work around with the CLI tools?

Theoretically, the AWS Console and the CLI tools are hitting the same API, so there's no reason they shouldn't be able to provide feature parity, but it's obviously difficult/impossible to build a UI that allows for all possible use cases without making the thing so general-purpose it borders on useless for beginners. But I'm sure some improvements could be made.

Peter Hancock (@piquet_h) *says:*
March 6, 2013 at 7:29 pm

* Updating launch configurations for newly "retired" AMI instances – means a rolling upgrade is easy. To the point we ditched the "golden image than you never need to retire" method.
* Losing US-EAST, struggled to connect on console, but command line from a different zone provided us the ability to launch new hardware and restore.
* Copying snapshots from zone to zone for backup. (Which you can do now on console)
* Forcing a "scale" when updating our application

I like the italics around theoretically. There's been a number of cases where the console has worked and the command line hasn't, and vice versa. It's another tool that provides access when the proverbial hits the fan!

My biggest complaint at the moment is that SQL Server on RDS is a second class citizen, and whilst the rest of the infrastructure scales beautifully, RDS / SQL Server remains an unscalable inflexible unit that doesn't really fit.

Sathish *says:*
December 18, 2012 at 9:43 pm

> This is true of Elastic Load Balancer (ELB), Relational Database Service (RDS), Elastic Beanstalk and others.

Since you don't use EBS backed services, I'm curious what services you use to load balance and databases.

The article is very insightful, Great Job.

Jens Rantil *says:*
February 25, 2013 at 1:58 am

> Since you don't use EBS backed services, I'm curious what services you use to load balance and databases.

Joe Tague *says:*
December 19, 2012 at 1:42 am

Hi Laurie,
Can you publish please any test suite you used to discover the differences in I/O performance?

In terms of using the 'ephemeral storage' what method do you employ to keep your data safe?

Any further info would be greatly appreciated.

Thanks

Andrew *says:*
December 19, 2012 at 7:54 am

Awesome article. Thanks for sharing so much. I would love to hear more about how you deal with your datasets if you aren't using EBS.

Ali *says:*
December 19, 2012 at 1:01 pm

> As we grow from over 100 to over 1000 boxes it's going to be necessary to diversify to those other providers.

Have you guys considered using something like RightScale.com to ease the pain of using multiple providers? If, like you say, you stick to basic services such as EC2 and S3, you can use Rackspace CloudServers and CloudFiles, or Google Compute Engine serves and CloudStorage.

seldo *says:*
December 19, 2012 at 1:39 pm

We have spoken to RightScale; my take the last time we spoke (which was at least 6 months ago) was that they were very expensive for the amount of value they add — I could just hire another ops guy. But things change rapidly in this space, so that may no longer be true.

Michael Grosser *says:*
December 24, 2012 at 5:18 am

You could check out Scalr.net, which is similar to Rightscale with the benefits of being cheaper, opensource and in my opinion have a better price/value ratio.

Dan *says:*
December 24, 2012 at 8:33 pm

Hey Seldo,

Great article, thanks for sharing your insights. We're working on something somewhat similar to RightScale (although a lot cheaper). Would love to hear your thoughts if you have a couple of minutes. Link is in my username.

Keep up the great work, looking forward to the follow up article.

Great article.
I can relate deeply to the cons and pros of EC2 that you mention.
BTW – from my experience, these kinds of issues are abundant throughout all the public cloud providers (so do not expect a whole different experience running on Rackspace...).

The thing is, that ideally – you really don't need to bother yourself with the internal details of the cloud services (e.g. the fact that ELB is usually down when EBS is down, in a whole AZ). Keeping yourself cloud neutral is good in some aspects, but then you will need to rebuild your application again and again for each cloud provider (or use tools like RightScale/Scalr/Chef/Puppet) – and that can be quite a hassle.

I think (I am biased – I work for Ravello) that a different approach altogether is needed – being able to concentrate on the application, and not the specifics of one cloud provider, by virtualizing the cloud for your application.
Take a look at: http://www.ravellosystems.com/blog/the-it-productivity-challenge-2/ .

André *says:*
December 20, 2012 at 5:21 am

Hello,

If you abandoned EBS where do you store data? Where do you keep your database?

Regards,
André

Pingback: *Erfahrungen: Amazon EBS ist das fehlerhafte Rückgrat von AWS › CloudUser*

Pingback: *Lessons learned: Amazon EBS is the error-prone backbone of AWS › CloudUser*

Pingback: *Best Practices for Running Your Startup off AWS | Inside-Startups.com*

Nei Grando *says:*
December 21, 2012 at 10:03 am

Great post! Nice advice about AWS cloud computing products and services. Thanks!

I really think Amazon and other service providers are working hard, learning fast and improving their solutions everyday. It is all new, so we must be careful and patients.

Att. @neigrando

Paresh *says:*
December 21, 2012 at 10:33 am

Excellent article. Thanks for sharing your experiences. Usually when I talk to people who use AWS, I want to ask them, so what are your backup/recovery plans and plan to provide targeted up time? My question was more out of curiosity than the solid experiences with AWS that you have shared but my experiences in general have taught me that achievement of backup/recovery objectives, targeted up time etc. is always by conscious design. I usually don't bother asking because most people answer, oh, we don't have to worry, we use AWS.

Cheers,
Paresh

Pingback: *Issue 32 – "I Came Here To Change the World" — TLN*

Brian Webster *says:*
December 26, 2012 at 9:18 pm

Now that you can copy EBS snapshots between regions (as of December 2012) via API calls, multi-region redundancy doesn't have to be limited to the larger teams.

PK Hunter *says:*
August 26, 2013 at 7:59 pm

If this is not a one-click thing, most people won't use it unless they are "larger teams". So, your thought is well meaning yet inaccurate.

Eric *says:*
January 3, 2013 at 12:03 pm

Any interest in having this republished in DZone's cloud portal?

Pingback: *Develop in the Cloud - Keith Dawson - Friday 4: AWS Downsides & Late Projects*

Martin *says:*
January 23, 2013 at 4:39 am

If your looking toward Europe you'll find some hosting companies specialized in high performance hosting. So there is plenty of alternatives to Amazon and Rackspace. In Berlin I like SysEleven. Great team and great performance.

Pingback: *Enterprise Conversation - Keith Dawson - Lessons From Microsoft's Azure Debacle*

Aldo *says:*
March 4, 2013 at 4:35 am

Nice post, thanks! I haven't started using AWS yet , now I want to learn more. By the way I'd like to recommend this course on AWS and others backends, might be helpful for Spanish speakers.

Pingback: *AWS Summit 2013 | Jacob Allred*

Kevin Trye (@kevintrye) *says:*
July 22, 2013 at 1:21 pm

Good article. Coming from a vps server background building CMS websites, we've certainly found Amazon EC2 host offering really convenient and easy to setup. However performance on their EC2 is very poor, having to use lots of caching tools to get acceptable speed. I/O and db performance is around a small fraction of that I'd get on similarly priced dedicated host hardware. This means for me it's suitable only for small sites or development projects, certainly not large projection websites.

Pingback: *Amazon's weekend cloud outage highlights EBS problems – Register | DailyNewsFirst.com*

Pingback: *Amazon's weekend cloud outage highlights EBS problems – Register - Topnewsoftoday*

Pingback: *Amazon's weekend cloud outage highlights EBS problems – Register | freenewsportal.com*

Pingback: *Amazon's weekend cloud outage highlights EBS problems – Register | News Supply Daily*

Pingback: *Amazon's weekend cloud outage highlights EBS problems – Register | Custom News Cast*

Pingback: *Amazon's weekend cloud outage highlights EBS problems – Register | EikAwaz*

Pingback: *Amazon's weekend cloud outage highlights EBS problems – Register | Phase In Out*

Pingback: *Amazon's weekend cloud outage highlights EBS problems - Register | Mash Mush*

Pingback: *Engineering a Single Point of Failure with AWS. Users Beware. | The Diversity Blog - SaaS, Cloud & Business Strategy*

Sundar *says:*
October 6, 2013 at 11:57 am

For anyone else that wishes to read the promised follow-up article: http://blog.awe.sm/2013/02/21/backups-ebs-and-awe-sm/

Steve *says:*
November 5, 2013 at 1:33 am

I'm surprised you don't really touch on cost. EC2 is damned expensive – especially compared to doing it yourself, or other providers. I put together a comparison here: https://secure.slicify.com/Calculator.aspx

Evan *says:*
December 15, 2013 at 5:44 pm

Steve,

consider that you don't need to pay someone, either internal or a contractor, to maintain your hardware. And since you don't show your methodology in that calculator, you might as well not be counting any of those other extra costs associated with hosting your own.

Another way of saying this is, if it's that costly (as you seem to think it is) I don't see EC2 being as successful as it is. And it IS successful.

Evan

slicify *says:*
February 21, 2014 at 6:52 am

Here's a more detailed comparison: http://blog.slicify.com/2013/08/05/how-amazon-is-ripping-you-off/ And yes – AWS is expensive. It's at least 10-20x more expensive than using dedicated servers from a web hosting company.