

PostgreSQL on Amazon

Christophe Pettus
PostgreSQL Experts, Inc.
christophe.pettus@pgexperts.com

Selection bias.

- Cops mostly meet criminals.
- Doctors mostly meet sick people.
- Database consultants mostly meet people with serious database problems.
- Our contact with AWS was companies with database meltdowns.



**My Opinion of PostgreSQL on
Amazon, through 2010.**

***Don't do that!
You'll kill yourself!***

This didn't scale.

- 65%+ of new clients were running on Amazon.
- Were not interested in being told, “Oh, just redo your whole technical architecture.”
- In fact, many were good matches for AWS.

An aerial photograph of a snow-covered mountain range. The central focus is a large, rounded mountain peak covered in a thick layer of white snow. At the very top of this peak, a small, dark, rectangular structure is visible. To the right of the main peak, there is a more jagged, rocky mountain peak also partially covered in snow. The surrounding landscape is a vast, flat expanse of snow, with some faint tracks or lines visible in the lower-left corner. The overall scene is bright and wintry.


A more nuanced view was required.



Welcome to the Cloud.

What is cloud computing?

- Too many definitions.
- Computing as a service? Virtualized hosting? Decentralized storage?
- Let's just talk about **cloud hosting**.
- It is a total revolution in computing that has never been seen before.

A black and white photograph of an IBM console terminal. The terminal is a large, dark-colored cabinet with a light-colored top surface. On the top surface, there is a control panel with various buttons and a keyboard. The IBM logo is visible on the top left of the cabinet. The background is a plain wall. The text is overlaid on the image in a white, sans-serif font.

[The underlying operating system] allows the operator to divide up the computer into a set of partitions, each one with a fixed memory size, isolated from the others...

— *OS/360-MFT, circa 1966*

Cloud Hosting, I

- Dividing machines up into virtual machines, using a “hypervisor” kernel.
- (The term “hypervisor” was coined in 1965, btw.)
- OK, I’ll stop now.
- Providing these virtual machines as computing resources.

Cloud Hosting, 2

- The hosting provider:
 - Manages the mapping of virtual hosts to physical machines.
 - Feeds and waters the actual physical hardware.
 - Provides services, APIs, etc. to provision and manage these individual virtual hosts.

Amazon Web Services

- Huge raft of interesting services.
- We're going to focus on just a couple:
 - **EC2** — The actual hosting service.
 - **EBS** — Their “storage area network.”

Amazon Elastic Compute Cloud (EC2)

- A very large collection of commodity servers spread across data centers worldwide.
- Divided into “instances” (virtual hosts) with various capacities.



No Magic.

Instance types

- Wide range, with varying amounts of CPU, memory, and instance storage (i.e., disk space local to the machine).
- In essence, how much of a physical machine you get.
- Wide cost range, too.

A gentle reminder...

- You are sharing the instance with other customers.
- You get the CPU, memory and instance storage that you've requested, but...
- The I/O channel and network are shared across all customers on that instance.

Exception: Dedicated Instances

- Dedicates hardware to a particular customer.
- Still virtualized.
- \$7,305 per month per region.
- ... plus more expensive instances.

Non-Exception: Reserved Instances

- Reserved Instances are a pricing program, not a technical program.
- Reduces costs and guarantees you an instance if you commit to particular usage patterns.
- Doesn't change the tenancy of the servers at all.

Instances are just computers.

- You pick your own operating system.
 - And debug your own kernel bugs.
- You set up your own infrastructure (although Amazon has many cool tools).
- You install and operate your own user-level software.
- Amazon keeps the lights on.



Storage in AWS

Instance Storage

- Otherwise known as *ephemeral storage*.
 - When *Amazon* calls it ephemeral, believe them.
- Survives reboots (they say).
- Can disappear in a large range of circumstances.
- Most you can get is 3.4TB.

Elastic Block Storage, I

- It's a SAN over Ethernet.
- Individual volumes from 1GB to 1TB.
- Can be moved from one instance to another (only one at a time).
- Snapshotting to Amazon S3.

Elastic Block Storage, 2

- EBS server provides resilience against hard drive failures.
- Can mount any number of EBS volumes on a single machine.
- Can create RAID sets of multiple EBS volumes.

Elastic Block Storage, 3

- Runs over the network.
- Each instance has a single 1Gb Ethernet port...
- ... so the theoretical maximum performance for EBS on an instance is 125MB/second.
- Testing confirms this.

Elastic Block Storage, 4

- Elastic Block Storage is not cheap.
- You pay for both the storage itself, and I/O operations from and to it.
- This can add up.

Sharing is not always caring.

- You share the instance with other customers.
- You share the network fabric with lots of other customers.
- You share the EBS server with lots and lots of other customers.
- Result... um, not profit.



“The performance characteristics of Amazon’s Elastic Block Store are moody, technically opaque and, at times, downright confounding.”

— Orion Henry
Co-Founder
Heroku

EBS has good days.

- 80-130 megabytes per second throughput.
- 20ms latency.
- Low variability.

EBS has bad days.

- 2 megabytes per second throughput.
- 2,000ms (yes, 2 second) latency.
- Depends on things utterly outside of your control.

Instance storage for your database?

- Not protected against hard drive failures.
- Goes away if the instance shuts down.
- Not really any faster than EBS.
 - Amazon specifically says it's slower.
- Just use it for the boot volume.

Why do we care?

- Databases are all about I/O.
- Limits how fast you can write.
- For very large databases, limits how fast you can read.

Unpleasant facts of life.

- Instances can reboot at any time, without warning.
- Hard drive failures can destroy instance storage.
- EBS volumes... we'll talk about those later.
- Be prepared for this. It's part of the price of admission.

A grayscale photograph of three elephants wading through a river. The elephants are positioned in a line from left to right, partially submerged in the water. The background shows a flat, open landscape under a clear sky. The text 'PostgreSQL on Amazon' is overlaid in the center of the image.

PostgreSQL on Amazon

PostgreSQL on Amazon.

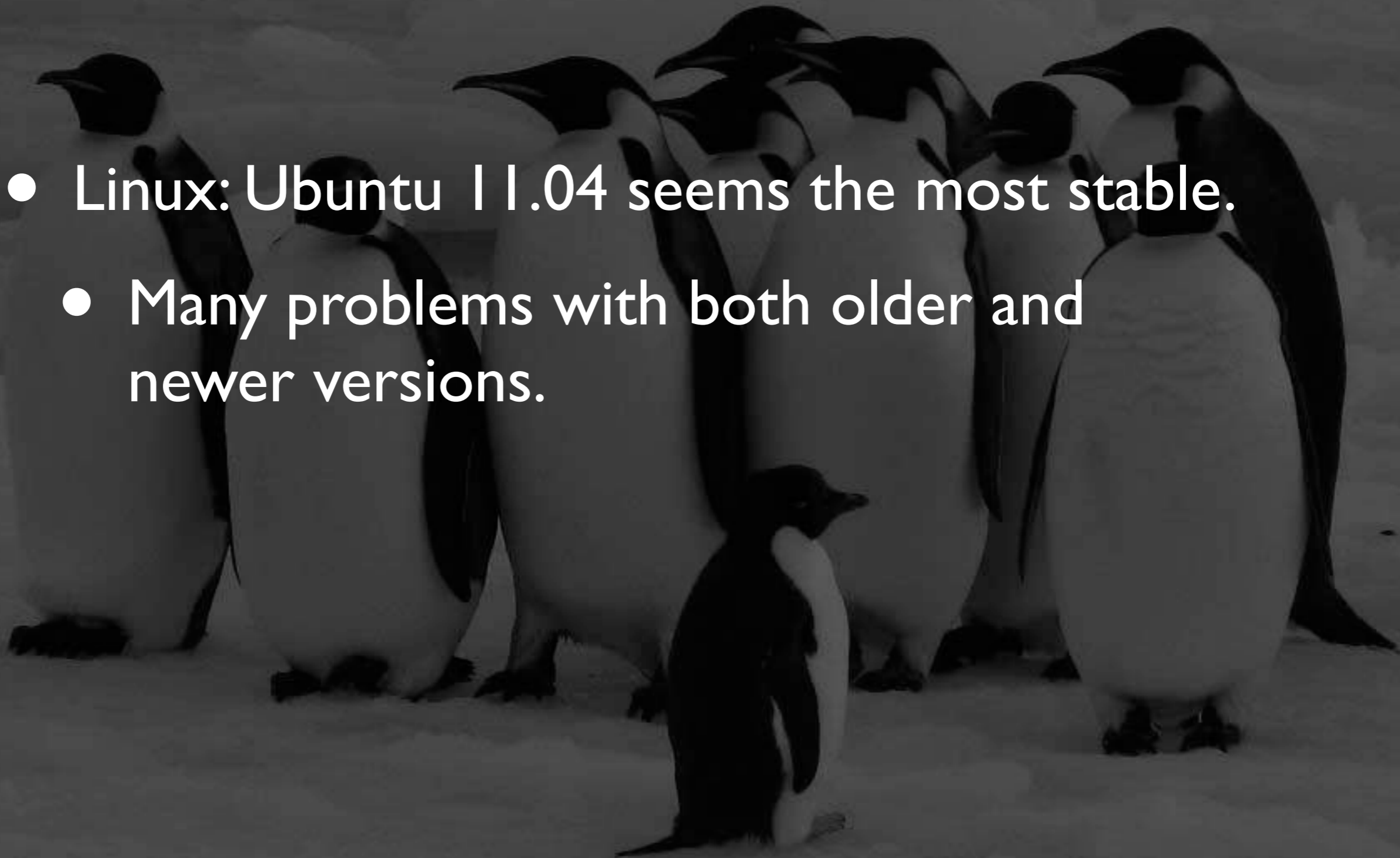
- Configuring your instance.
- Configuring EBS.
- Configuring PostgreSQL.
- Replication.

The Instance.

- Memory is the most important thing.
- If you can fit your whole DB in memory, do it.
- If you can't, max out the memory.

Mondo Distro.

- Linux: Ubuntu 11.04 seems the most stable.
- Many problems with both older and newer versions.



CPU usage.

- CPU is almost never the limiting factor in instance capacity.
- Always go for more memory over more CPU.
- CPU exhaustion is usually due to other processes on the same instance.
- Give them their own instance.

Configuring EBS.

- Really, only one decision about EBS:
 - To RAID or not to RAID?
- Folk wisdom that does not work:
 - Pre-zeroing the EBS volume.
 - RAID10.

Pro-RAID

- *Almost* all measurements show EBS RAID-0 outperforming single-volume.
- Less so on writes than reads, but still better.
- 8-stripe RAID-0 appears to be the highest performance point.

Anti-RAID



- Lose the ability to snapshot volumes.
- Remounting on new instances is tedious.
- EBS RAID has even more variability than single-volume EBS.
- Increases the chance of losing your data to an EBS failure.

Wait, *what?*

- EBS volumes can fail.
 - Or fail to mount on instance reboot.
- If one stripe fails, the whole RAID set is useless.
- Plan for it just like you would plan for an HD/SSD failure in a private machine.

EBS tips 'n' tricks.

- XFS.
 - Pretty much anything but ext3, really.
- `--setra 65536`.
- Chunk size 256k.
- deadline scheduler.
 - Or cfq. Or noop.

Configuring PostgreSQL

- Instances are just (virtual) computers.
- Everything you would otherwise do to tune PostgreSQL, do here.
- Check out Josh Berkus' "Five Steps to PostgreSQL Performance" talk.

The basics.

- Only run PostgreSQL on the instance.
- Put all of \$PGDATA on an EBS volume (striped or not).
- Fine to put the operation logs (pg_log) on instance storage.

pg_xlog

- Put it on the same EBS volume as the rest of the database.
- This is exactly contrary to normal advice.
- You cannot optimize seeks on EBS. Don't bother trying.
- If you lose the EBS volume, your DB is toast, anyway.

pg_xlog, 2

- Do not put pg_xlog on instance storage!
- Renders the database unrecoverable in case of an instance failure.

random_page_cost

- `random_page_cost = 1.1`
- EBS is so virtualized you cannot control the seek behavior.
- Sequential and random accesses are nearly identical in performance.

effective_io_concurrency

- If you are doing striped RAID, set to the number of stripes.
- If you are not, leave it alone.

Replication

- PostgreSQL on AWS means replication.
- Stop looking at me like that. Just do it.
- Too many uncontrollable failure modes to rely on the data being safe on one instance.

The basic setup.

- Streaming replication from one instance to another.
- Second instance does not have to be as capable.
- CPU usage on the second instance will be low, unless used for queries.

Availability Zones.

- You must put the replica in a different Availability Zone from the master.
- AWS appears to have customer affinity for physical machines.
- This is the only way to guarantee that your master and replica are not on the same machine.

EBS snapshotting.

- If you are using single-volume EBS, you can do point-in-time backups using snapshotting.
- Be sure you are saving the WAL segments as well as the data volume.
- <https://github.com/heroku/WAL-E>

Disaster recovery.

- Put a warm standby in a different region.
- Allows for point-in-time recovery.
- Keep 2-4 backup snapshots.
 - 2-4 backups/week.

Monitor, monitor, monitor.

- Replication implies monitoring.
- Disks can fill up with misconfigured replication.
- At minimum, monitor replication lag, disk usage.
- `check_postgres.pl`



Scaling

Sooner or later...

- You'll max out your High-Memory Quadruple Extra Large Instance with its 8-stripe RAID-0 EBS mount.
- And then what?
- Most scaling issues are application issues; fix those first.

Scaling basics.

- Pull stuff out of the database that doesn't need to be there.
 - Web sessions, large objects, etc.
- Move as much read traffic as you can to the replicas.
- Memory is cheap on AWS; use it for all it's worth!

More scaling basics.

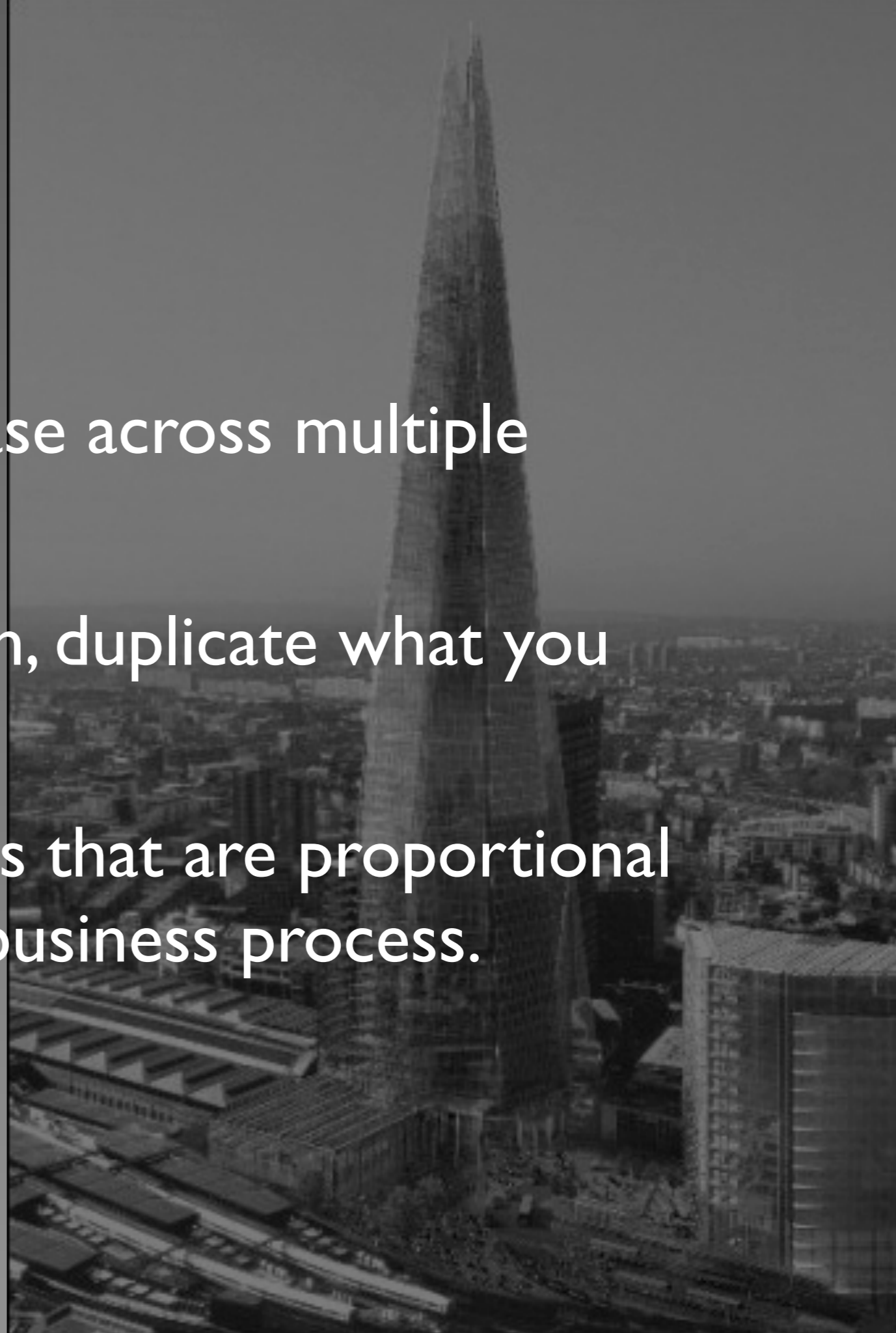
- Aim for a shared-nothing application layer.
 - Can automatically provision/terminate app servers as required.
- Digest and cache as much as possible in memory-based servers.
- Typical HTML fragments, result sets, etc.

The wall.

- Even so, you'll run out of performance (probably write capacity) on your primary database volume.
- Either consistently, or at peak moments.
- Then, it's time to make some tough decisions.

Sharding.

- Partition the database across multiple database servers.
- Isolate what you can, duplicate what you can't.
- Great for workloads that are proportional to a small atom of business process.



Lots of fun challenges.

- Keeping IDs unique.
- Routing work to the right database.
- Distributing shared data to all the instances.
- Handling database instance failure.
- Doing consolidated queries across all databases.

Data consolidation.

- Creating reports across all shards can be challenging.
- Export data to a central data warehouse.
- Do parallel queries with aggregation at the end.
 - PL/Proxy.

Sharding is not for everyone.

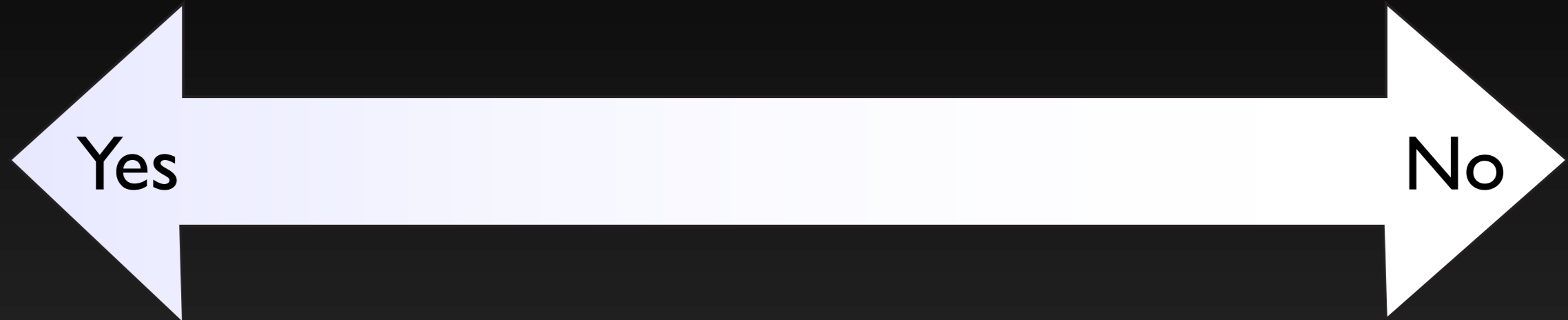
- Two major categories:
 - Data warehouses.
 - Very high write volume applications.
- Don't deform your application architecture just to achieve sharding...
- ... but a sharded architecture is great if the application naturally supports it.

Architecture for Amazon

A grayscale photograph of a construction site. In the foreground, there's a large concrete structure under construction, possibly a foundation or a large wall. A crane is visible on the left side, extending upwards. The background shows a hilly landscape with some trees and a clear sky. The overall scene is in black and white, giving it a professional and industrial feel.

- Design your architecture for sharding and distribution.
- Treat each instance as a disposable resource.
- Make full use of Amazon's APIs; automate everything you possibly can.

So, what do I do?



Small database (<50GB?)

Not write-critical

Locality of reference

Shardable application

Web OLTP

Large database

Write-critical

Global references

Unary application

Data warehouse

Hybrid solutions.

- Develop on AWS, deploy on traditional hardware.
- Primary web-facing servers on AWS, data warehouse on traditional hardware.
- Impractical to have the app server and database in different hosting environments, though.

Running with scissors.



- Turn off all PostgreSQL safety features.
- Rely on streaming replication to preserve data.
- Treat each instance and EBS volume as disposable.
- Hope the numbers work in your favor.

A dark, monochromatic image of a catacomb. The walls are covered in a dense, repeating pattern of human skulls. In the center, a person's silhouette is visible, standing in a niche or doorway. The overall tone is somber and macabre.

We do not recommend this.

Avoid Amazon Stockholm Syndrome

- No one cares that you run on Amazon.
- Your business is not defined by where you host your computation resources.
- If Amazon doesn't do what you need, move.
- After all, it's all just about...

A large stack of US dollar bills, with the word "Cost" overlaid in white text. The bills are stacked in a way that shows the edges of many individual bills, creating a textured, layered appearance. The word "Cost" is centered in the middle of the stack in a bold, white, sans-serif font. The background is a dark, slightly blurred grey, suggesting an indoor setting with a stack of money on a table or counter.

Cost

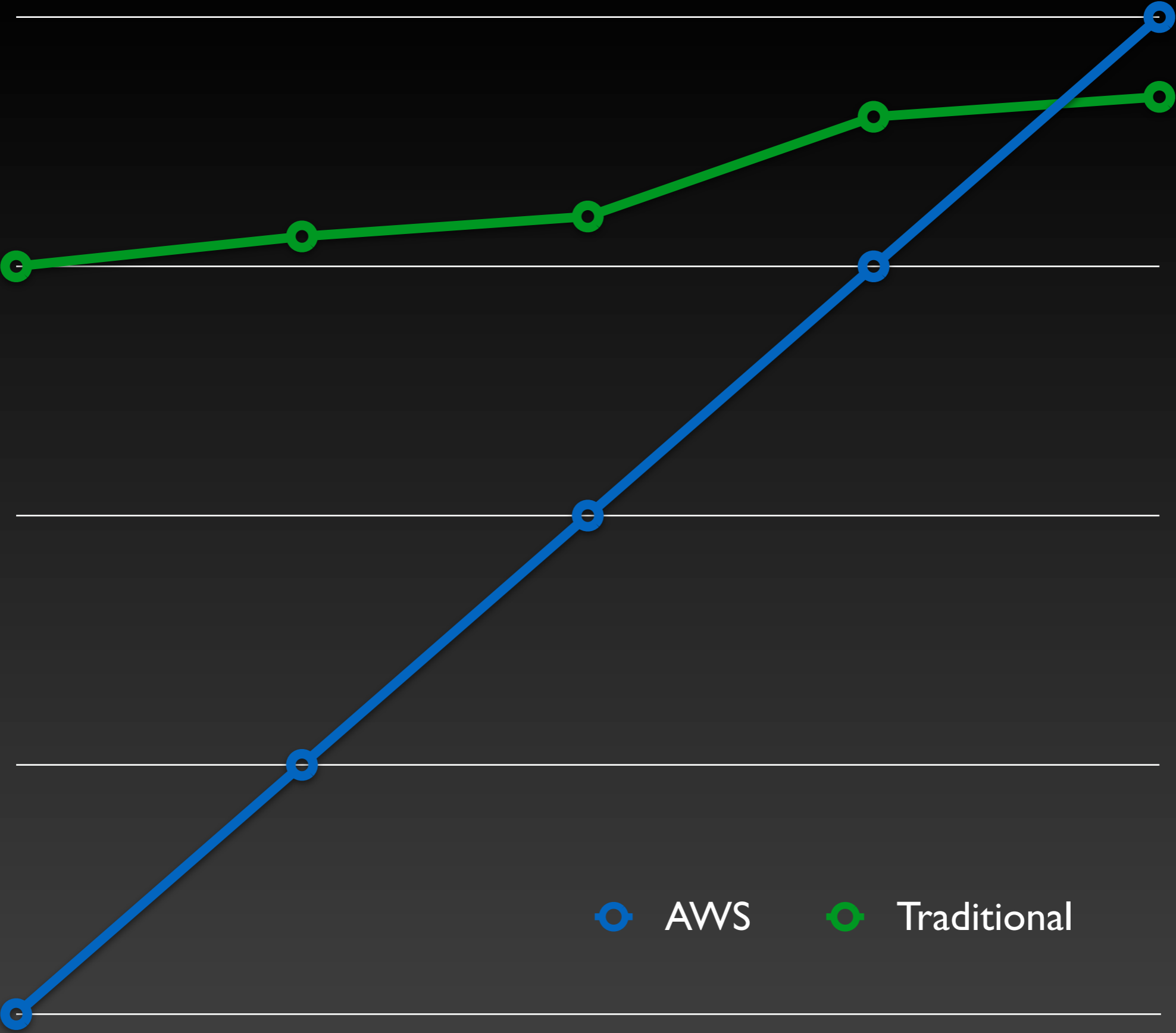
Traditional cost model.

- High buy-in.
- Cost rises in bumps and jumps as more capacity is required.
- Hard to scale on-demand.
- Economies of scale exist.

AWS cost model.

- Starts at near-zero.
- Increases linearly with capacity.
- Can provision up/down very quickly.
- No economies of scale (discounts are not economies of scale).

The Most Oversimplified Cost Comparison in the History of Computing.



○ AWS ○ Traditional

Do not forget...

- ... bandwidth is extra.
- ... I/O operations are extra.
- These can swamp the actual instance cost.
- Be sure to include them in your cost estimates.

A note on staffing.

- “Cloud hosting” does not mean “no operations staff.”
- You can defer this on cloud hosting, but:
 - You will need these people eventually.
- Every one of our large AWS clients has hired people to manage their “data center.”

Padding up the Amazon.

- AWS is a great solution if your application matches its technical and pricing model.
- Take full advantage of it if it is a good fit.
- Don't deform your architecture just to make it work.
- Consider costs and alternatives carefully.

A photograph of a tropical river or stream. The water is a murky, brownish-yellow color, reflecting the dense green foliage on the banks. The banks are lined with various tropical plants, including palm trees and other leafy vegetation. The scene is captured from a low angle, looking down the length of the river. The overall atmosphere is lush and natural.

Thanks!



pgexperts.com

thebuild.com