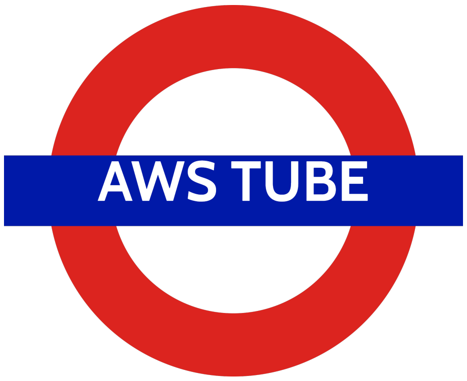

Figure 1: CLICK HERE TO ENTER STATION

Recovery Sheet 1

SJ Green

Recovering from the session delivered on the night of Wednesday 15th January 2025.

1 Active-Active

Description

1. AWS (Amazon Web Services) emerged in 2006. The current CEO is Matt Garman. Although it has its headquarters in Seattle (US state of Oregon), you cannot explain IT in the United Kingdom today properly without mentioning AWS; three central government departments spent £394 million on AWS in 2023 [Donnelly 2024]. Whoops that was a typo the 3 should be... an 8.

2. **Cloud computing** is the "on-demand delivery of compute power, database, storage, applications, and other IT resources through a cloud services platform via the Internet with pay-as-you-go pricing" [AWS 2025]. Perhaps Eric Schmidt used the term first [Regalado 2011]. It is network-based computing. Actual clouds are (1) almost *everywhere* and (2) it is difficult to say where, exactly, a cloud ends, since clouds have fuzzy edges. By calling computing *cloud* computing, we succesfully capture the fact that the physical server's location would be hard to pin down (just as it would

be hard to define the precise point in space that a cloud ends). When asked where the server is, an engineer might point to the sky and say "the cloud", as if to say "somewhere, but going into detail about where exactly doesn't really matter". We also capture the fact that the computing power can be accessed almost *everywhere*—since it is a technology based on the Internet—just as you can look up to clouds wherever you are in the world.

3. Most AWS products are accessed over the Internet using **APIs**.
4. An API is a set of **rules, protocols, and tools** that allow different software applications to communicate and interact with each other.
5. The "web" was invented by English computer scientist Sir Tim Berners-Lee. It was the killer application of the technology known as **the Internet** (as Jake Feinler puts it).
6. Tim Bray is a former engineer at Amazon and he is a co-author of the specification for XML, or Extensible Markup Language.
7. A **Region** in AWS is a "physical location in the world where we have multiple Availability Zones" [AWS 2025]. Usually there are three AZs.
8. "Availability Zones consist of **one or more discrete** data centers, each with redundant power, networking, and connectivity, housed in separate facilities" [AWS 2025] *That's right - according to AWS documentation today, an AZ can be one data centre.*
9. **CloudFront** is a content delivery network, which relies on edge locations that store content closer to users.
10. AWS Local Zones are "extensions of a Region". The web page states: "Run applications that require single-digit millisecond latency or local data processing by bringing AWS infrastructure closer to your end users and business centers" [AWS 2015].

2 Warm Standby

This is the Warm Standby section. It takes a bit of time to read through it and recover your understanding, but it's quite cheap to do, in terms of time and effort. The functionality—in terms of teaching you—is right here.

1. Amazon Web Services emerged in 2006. It is a subsidiary of Amazon. The CEO of Amazon was Jeff Bezos. On July 5th 2021, Andy Jassy became CEO of Amazon [Palmer 2021]. Jassy proved himself in AWS. AWS is the largest provider of cloud computing. Today, Matt Garman in the CEO of AWS.

2. Cloud computing is “the on-demand delivery of compute power, database, storage, applications, and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing” [AWS 2025] In Feb 2003, Bezos gave a TED Talk in which he claimed that the Internet industry was in a state similar to the early days of the electricity industry - people could not yet get electricity on tap.

3. Most AWS products—such as Amazon S3, EC2, and CloudFront—are accessed over the Internet through APIs. These APIs follow web service protocols, making them accessible from anywhere with an internet connection. For example, when you launch EC2 instance, you’re interacting with these services via HTTP requests to AWS’s API. People in the AWS community will often talk about specific “API calls”. Some examples of API calls used with EC2 include `RunInstances`, `DescribeInstances` and `StartInstances` [EC2 API Reference] This is the crucial point: *Every time you execute a command using the Command Line Interface, or click a button on the AWS Management Console, it it executing one of these API calls.* In other words, that **graphical user interface** you’ve been using, called the **AWS Management Console**, is just one big ’Api-meal, a highly processed re-packaging of the real goodness: the API calls. The sooner you learn to cook, painful as it may be, the better. The fact that all AWS products are externalised as a set of API calls is no accident. There is a well-known myth around how Bezos instructucted Amazonians early on to make all functionality accessible by API [Stevey’s Platform Rant]. Employees were told: “if you have hundreds of services, and your code MUST communicate with other groups’ code via these services, then you won’t be able to find any of them without a service-discovery mechanism.” We’re told how: “Over the next couple of years, Amazon transformed internally into a service-oriented architecture”.

4. The term **API** stands for Application Programmable Interface. An API is a set of rules, protocols, and tools that allows different software applications to communicate and interact with each other. Essentially, it defines how one piece of software can request and exchange data or functionality from another, acting as a bridge between systems.

5. The term “Web” refers to a technology that makes use of the Internet. It was invented by English computer scientist Sir Tim Berners-Lee. Its creation is usually attributed to the year 1989 (the year the Germans said *nein* to the Berlin Wall). Information transmitted on the Web must obey the rules of the HyperText Transfer Protocol. Not everything that makes use of the Internet makes use of Berners-Lee’s Web.

6. **Tim Bray** is a former Distinguished Engineer at Amazon Web Services. “Distinguished Engineer” is a title that AWS gives to engineers that have done significant work designing important technologies. Bray is Canadian (the AWS headquarters in Seattle, USA, are quite close to Vancouver).

Bray worked a lot on **XML**, or Extensible Markup Language. This is like a *lingua franca* for machines on the Internet. They can all understand each other if they communicate using XML. When people talk about a "Web Service" they are often talking about a standardised way of integrating web-based applications using something like XML [Wikipedia].

7. A **Region** in AWS is a "physical location in the world where we have multiple Availability Zones" [AWS 2025]. Most Regions nowadays have at least three Availability Zones, but Northern Virginia probably has five or six [Green 2025]. Every region since Paris in December 2017 has launched with 3 AZs [Schwarz 2020]. Regions are denoted using codes. For example, London is eu-west-2 and North Virginia is us-west-1. North Virginia is probably the most important Region because it was built first. One of the best ways to build your familiarity with the US states, is using this Sporcle quiz. At least know that Northern Virginia is on the east of the USA and the big earners in Northern California are on the west of the USA.

8. An **Availability Zone** is one or more data centres that fail independently. Breaking all the AWS infrastructure into Availability Zones makes the overall system more resilient. Because the *parts* fall away cleanly, like Secret Service agents making the ultimate sacrifice for the principal they're protecting, the *whole* tends to be more available than if the parts brought each other down in a tangled mesh. "Availability Zones consist of one or more discrete data centers, each with redundant power, networking, and connectivity, housed in separate facilities" [AWS 2025]. Examples of AZs include us-east-1a and us-east-1b. Notice how we use a lowercase letter at the end, to show that we are not merely denoting a Region.

9. **CloudFront** is the name of an AWS product released pretty early on, in 2008. It is a Content Delivery Network. This means we can use it to cache popular files closer to users. This makes our website run much faster. (Or to use the terminology that computer scientists enjoy, our website will have *lower latency.*) You will use this command `aws cloudfront create-distribution` to create a *distribution.* Now *distribution* is just a helpful term the product designers use for the bunch of properties — or configuration — which specify how you want your caching to work. For example, for how much time should ojects (another word for files) live in the cache before we refresh the cache? Should we take into account cookies when deciding to refresh the cache? Should we take into account whether the user is making a GET request or a PUT request, or should we simply ignore the method? These are some examples of the properties you need to specify in your *distribution.* AWS has data centres and other supplementary infrastructure, in many places all over the world. The term "Point of Presence" is suitably general, with its paranormal connotations, to denote all these different things.

We might also talk about "Edge Locations". This term is more specific, if it comes to it; *edge* implies being at the end of the chain, right near to the client [AWS Fault Isolation Boundaries 2025]. It is a reference to edge computing. This is performing processing of data (computing) near to the source of the data, as opposed to far away in a data centre (the classic example of the "cloud"). AWS CloudFront also talks about Regional Edge Caches (RECs) which act at the Region level, preventing the ELs from having to go back to the Origin each time (the same way the ELs save the client from having to go to the Origin each time). An Edge Location is one example of a PoP; it is a "data center" capable of storing the static files [AWS 2025]. "Edge Location" is a relational term, it seems, for a data centre that is an EL for you in Indonesia will not be an EL *for me*, in the UK. This is because it will not bring content closer to me, from the **Origin** server, if it is in Indonesia. So it will not be an Edge Location *for me*. That's fine. So, Amazon Cloud**Front** fronts your **principal**—the .jpg photos of your important pal Pete—with a praetorian guard of Points of Presence. But it's not just in Amazon CloudFront that AWS makes use of this principle of bringing content closer to the user to reduce latency. Edge Locations are also used in Route 53, AWS Shield, and AWS Global Accelerator.

10. In 2019, the year before the pandemic would properly hit, AWS announced a tried of products: Local Zones, Outposts, and Wavelength. They are all related. You'll notice that Local Zones involves the term *Zones*, referring to Availability Zones. Local Zones are "extensions of a Region". Well, this is the line that AWS used at one point. All the packet parroters have swooped on it. Chris Tozzi (2024) tells us "AZs shouldn't be confused with Local Zones, which are extensions of a Region", Nannette Vilushis of graphon.com tells us that Local Zones are "extensions of a Region". With AWS, much content is generated by people essentially lifting things verbatim from the official documentation. Packet Parroters.

    The question, of course, is why we need a special term "Local Zone" for an extension to a region. If a Region is nothing but AZs (and according to the AWS definition above, *it is*), then we would surely denote an extension to a Region as... an AZ. Furthermore, if I happen to be right next to a data centre, then the AWS infrastructure will be very near to me, an end user. Is that sufficient to make that data centre a Local Zone? That would be very strange. It would mean that I could create Local Zones out of AZs, like a fairy godmother who, tired of life, has thrown away the wand and just spends the days staggering towards large data centres clutching a laptop. And yet Local Zones are defined in terms of closeness to end users: "Run applications that require single-digit millisecond latency or local data processing by bringing AWS infrastructure **closer to your end users**" writes the glossy web page. We'll dive deeper into this below.

3 Pilot Light

Now, on this third section, disaster recovery becomes slower and we start to provide the **form**, if not the **content**, of the kind of thing we want to restore. (Pilot light is cheaper than warm standby but things take much more time.)

1. **The Well-Architected Framework**. The five original pillars are Security, Operational Excellence, Cost Optimisation, Reliability, and Performance Efficiency. To help remember them, you could tell a "just so" story such as: begin with security because it is always "job zero" at AWS and the first three pillars spell SOC, and Reliability and Performance can be thought of as naturally similar. Reliability is fourth because reliable wooden tables have four legs. AWS maintain a whitepaper on the Well-Architected Framework.

2. **What is Availability?** Availability Zones are so-called because they have some relationship to availability. The opposite of availability is downtime. In this session, I referred to the collapse of the Francis Scott Key Bridge in Baltimore. I suggested that the bridge did **not** consist of Zones of Availability (in the Amazonian sense). This is because a container ship hit one part of the bridge, and the structural failure was then conveyed to other parts of the bridge. ("Why Bridges Collapse", BBC iPlayer, Dec 5th 2024)

   Gray, Jim (1985). Why do computers stop and what can be done about it? Available at: `https://pages.cs.wisc.edu/~remzi/Classes/739/Fall2018/Papers/gray85-easy.pdf`

3. **What, exactly, is Operational Excellence?** Somebody asked this question in the session. I think you could argue that this pillar of the Framework is the one that is most unfamiiliar; everyone knows what security and cost optimisation is, but in everyday life, nobody has reason to mention "operational excellence".

   (a) Operational Excellence. *Wikipedia.* Available at: `https://en.wikipedia.org/wiki/Operational_excellence`

   (b) Debois, Patrick and John Willis and Gene Kin and Jez Humble (2016). *The DevOps Handbook.* O'Reilly.

   This book is about DevOps, which involves ensuring that the development team and the operations team work closely together. The book emphasizes getting fast feedback, integrating code frequently into a central repository and then testing it (rather than testing it all at the end in one go), creating a blameless culture in an organisation and having small teams (so small they can be fed with two pizzas) like they do at Amazon.

(c) AWS. The Well-Architected Framework. Available at: `https://docs.aws.amazon.com/pdfs/wellarchitected/latest/framework/wellarchitected-framework.pdf#operational-excellence`

4 Backup and Restore

This is a further reading section. *The sources allow you to restore a rich expertise within yourself, but it will be much slower.* I'm trying to add value with this section by selecting **high-quality sources**, since there is lots of material about AWS which merely replicates material from the official documentation, only with less skill in articulation. I want this section to be like Masterpiece Classic on PBS. Some of these inclusions are strategic; I believe the biggest gift I can give you is to get you hooked on the real sources of expertise. It's very important that you open one source from Jeff Barr, one from Vogels, one from Jassy, and one from Garman, in the list below - these people are your official friends if you want to know AWS. Quinn and Brazeal, meanwhile, are key players in the AWS community.

1. AWS and its History

   Reinvent is the name of an annual conference held by Amazon Web Services. Usually it is stylised as Re:Invent but I will always write "Reinvent". Reinvent started in 2012. Of all the keynotes, I think the first five minutes of the 2017 keynote display Jassy's belief that specific details are completely in his control, despite being at the top. The momentum onstage is palpable, *embodying* the sheer force of the company, upwards, in exceptional presentational skills.

   - Miller, John (2016). How AWS Came to Be. *TechCrunch.* Available at: ¡https://techcrunch.com/2016/07/02/andy-jassys-brief-history-of-the-genesis-of-aws/¿
   - Jassy, Andy (2024). The Amazon Leadership Principles. YouTube Channel: Inside Amazon. Available at: `https://www.youtube.com/watch?v=My-2-MyxamQ&ab_channel=InsideAmazon`
   - Bezos, Jeff (2003). The Electricity Metaphor. *TED Talk.* Available at: `https://www.youtube.com/watch?v=vMKNUylmanQ&ab_channel=TED`
   - AWS Reinvent (2017). Andy Jassy Keynote. Available at: `https://www.youtube.com/watch?v=1IxDLeFQKPk&t=7534s&ab_channel=AmazonWebServices`
   - Barr, Jeff (2009). New Features for Amazon EC2: Elastic Load Balancing, Auto Scaling, and Amazon CloudWatch. *AWS News Blog.*

May 18th 2009. Available at: `https://aws.amazon.com/blogs/aws/new-aws-load-balancing-automatic-scaling-and-cloud-monitoring-services/`

- Vogels, Werner (2022). Werner Vogels Keynote. Available at: `https://www.youtube.com/watch?v=RfvL_423a-I&t=1662s&ab_channel=AmazonWebServices`
- Rick Dalzell. *Wikipedia.* Available at: `https://en.wikipedia.org/wiki/Rick_Dalzell`
- Bryar, Colin and Bill Carr (2024). An Insider Look at Amazon's Culture and Processes. Available at: `https://www.aboutamazon.com/news/workplace/an-insider-look-at-amazons-culture-and-processes`
- Brazeal, Forrest (2020). 169 AWS Services in 2 minutes. YouTube Channel: A Cloud Guru. Uploaded September 2nd 2020. Available at: `https://www.youtube.com/watch?v=BtJAsvJOlhM`

2. Cloud Computing

   Please understand that this is true: if an individual is a cloud engineer and very good at the cloud, then this person has competence in CLIs, Linux, Python, and Bash.

   - Berkeley (2009). Above the Clouds: A Berkeley View of Cloud Computing. Available at: `https://www.youtube.com/watch?v=IJCxqoh5ep4&ab_channel=UCBerkeley`
   - Fox, Armando (2011). Available at: `https://www.youtube.com/watch?v=MroUlbiKiOU&t=29s&ab_channel=DanGarcia`
   - Hamilton, James (2011). Internet Scale Storage. *University of Washington.* Available at: `https://www.youtube.com/watch?v=aS-FI4eTods&t=198s&ab_channel=UWVideo`
   - Garman, Matt (2017). AWS Compute: What's New in Amazon ECS, Containers and Serverless? *Reinvent 2017.* [Conference]. Available at: `https://www.youtube.com/watch?v=6ROxRKkLkbc&ab_channel=AmazonWebServices`
   - Peña-Siguenza, Gwyneth (2023). A Better Way to Leverage Cloud Certifications. Available at: `https://www.youtube.com/watch?v=3GPMaizoZe8&ab_channel=GPS`

3. AWS and APIs

   - Yegge, Steve (2011). Stevey's Google Platforms Rant. Available at: `https://gist.github.com/chitchcock/1281611`
   - Yegge, Steve. Available at: https://gist.github.com/kislayverma/6681d4cce736cd7041e6c8214469d2f
     "You have to understand: most people were scared around Bezos because they were waaaay too worried about trying to keep their jobs. People in high-level positions sometimes have a little too much

personal self-esteem invested in their success. Can you imagine how annoying it must be for him to be around timid people all day long? But me – well, I thought I was going to get fired every single day. So fuck timid. Might as well aim high and go out in a ball of flame."

- Poile, Simon (2015). Build and manage your APIs with Amazon API Gateway. Available at: `https://youtu.be/FcH8ovrNd9k?si=eNJuQtFg4I9JIgAC&t=95`

  Poile (2015) describes four "categories of problem" that people had in building APIs. So, Amazon decided to build a product that would help people to build their own APIs. (Yes, a product accessible as an API that helps customers to build their own APIs...). Listen to Poile list the four problems, by clicking on the link above, and it will give you clues to what an API is. Launched in 2015, the product would be called **Amazon API Gateway**.

4. What is an API

   It took me a great deal of time to understand what an API was. Many teachers on the web are very undisciplined here, waffling on about how APIs abstract away unnecessary details, without ever (1) saying what an API actually *is* or (2) how I know I'm *looking at* an API. The student is left infuriated. After all, lots and lots of things abstract away details, other than APIs. Examples include the London underground map and Gary gossiping at the garage. And it's really not that difficult to grasp the concept of *abstracting away details*, but people seem so proud of themselves in saying that APIs do this. The student is left thinking "okay, APIs abstract away details... What is an API?"

   Open the API reference for an AWS product such as EC2 or S3. I think it is better to open the PDF version of the documentation rather than the web page in HyperText Markup Language. This is because you have a clear directionality, moving from one page to the next, with PDF. You don't use **hyperlinks**, so the decision about what to examine next is made for you—you just scroll to the next page—and you can move through the content systematically. So, again, open the API reference for EC2, and just take it in. It won't make sense at first. You might see an example of a request which makes an API call. It will have many different parameters. If you spot a Uniform Resource Locator, consider what the question marks in it are doing. Well, spending some time sitting with this API reference document will help you understand what an API is. Sit with a couple of pages at least.

   So, be patient with this one. I think the Wikipedia article entitled "Web APIs" will be more helpful to you than the article entitled "APIs".

Dirty rule of thumb: anytime somebody talks about an **endpoint**, they are talking about an API.

Not all APIs are REST APIs. Not all APIs make use of HTTP.

- Farish. What is an API? *Codecademy.* Uploaded Jan 30th 2019. Available at: `https://www.youtube.com/watch?v=IAFN2UzN784&t=20s&ab_channel=Codecademy`
- Web API. *Wikipedia.* Available at: `https://en.wikipedia.org/wiki/Web_API`
- API. *Wikipedia.* Available at: `https://en.wikipedia.org/wiki/API`
- Roy Fielding. *Wikipedia.* Available at: `https://en.wikipedia.org/wiki/Roy_Fielding`
- Fielding, Roy (2000). Architectural Styles and the Design of Network-Based Software Architectures. Available at: `https://ics.uci.edu/~fielding/pubs/dissertation/software_arch.htm#sec_1_1`
- Dennis, Craig (2023). APIs for Beginners. Available at: `https://www.youtube.com/watch?v=WXsD0ZgxjRw&ab_channel=freeCodeCamp.org`
- Cronin, Gareth (2024). Using S3 as a static data API with Cloud-Front. Available at: `https://cgarethc.medium.com/using-s3-as-a-static-data-api-with-cl`
- Reddit post. Available at: https://www.reddit.com/r/learnprogramming/comments/u758rb/whats$_{t}he_{d}ifference_{b}etween_{aw}ebsite_{a}nd_{ar}est/?rdt =$ 56051

  "APIs can be anything that provides data or possibly lets you modify it. It doesn't have to be web related. For example, Java has APIs for all the classes it supports (like the String class). That is not a REST API. This API lets you know what methods you can call on a String object.

  Back in the day, when the web was new, there were no web APIs and most sites were static, but as there became a desire to create interactive websites with data, there were things like SOAP and eventually REST APIs."
- REST. *Wikipedia.* Available at: `https://en.wikipedia.org/wiki/REST`

5. The Web

Please study what a URL is. So many people just throw around the terms URL and domain name and path. Many URLs do have domain names within them, but the domain name is just one part of the URL. Please, it will pay off if you sort these concepts out. You cannot seriously claim to know about AWS Load Balancers if you cannot explain, say, the difference between a path and a domain.

- "This is for Everyone". *Tim Berners-Lee takes part in the Opening Ceremony of the London 2012 Olympics, held at the end of the Jubilee Line (Stratford).* Available at: `https://www.youtube.com/watch?v=UMNFehJIi0E&ab_channel=WebScienceTrust`.
- The World Wide Web - Crash Course Computer Science. `https://www.youtube.com/watch?v=guvsH5OFizE&ab_channel=CrashCourse`
- "URL". *Wikipedia.* Available at: `https://en.wikipedia.org/wiki/URL`
- List of RFCs that are must read? *Reddit.* Available at: `https://www.reddit.com/r/networking/comments/lbuidw/list_of_rfcs_that_are_must_read/`
- RFC 2616. Hypertext Transfer Protocol. Available at: `https://datatracker.ietf.org/doc/html/rfc2616`

  It's very annoying, the way hypertext is one word but gets two letters in the acronym. Hypergreedy. Note what the t stands for. It's transfer. We're at Layer Seven of the OSI stack, so we're dealing with things that can come and go like Carroll's ferry cat.

  Trans**mission**– now that's a cool word. That's reserved for Tom Cruise protecting vital *infrastructure* (literally: the structures within, the systems which are underneath). It's for people in formal suits who are as reliable as sturdy, four-legged tables (TCP is a protocol and it's at Layer 4).

6. Tim Bray

- Bray, Tim (2021). Jassy Talking Points. *Ongoing* [Blog]. Feb 5th 2021. Available at: ¡https://www.tbray.org/ongoing/When/202x/2021/02/05/Jassy-Talking-Points¿
- Tim Bray. *Wikipedia.* Available at: `https://en.wikipedia.org/wiki/Tim_Bray`
- Bray, Tim (2016). Serverless Apps with AWS Step Functions. *Reinvent 2016* [Conference]. Available at: `https://www.youtube.com/watch?v=75MRve4nv8s&t=67s&ab_channel=AmazonWebServices`
- The role of Distinguished Engineers at AWS. YouTube Channel: Amazon Web Services. Apr 15th 2024. Available at: `https://www.youtube.com/watch?v=1yxWbw0CxR4&ab_channel=AmazonWebServices`

You can listen to engineer Jim Roskin talk about CloudFront (see source below). We can distinguish other engineers: James Hamilton, Eric Brandwine, Matthew Wilson, James Gosling, Mark Brooker, Andrew Certain, Colm MacCartháigh, Ippokratis Pandis—for example. We will meet most of them over the next few weeks.

7. AWS Regions

   - Hamilton, James (2017). How Many Data Centers Needed World-wide *MCDirona* [Blog]. Available at: ¡https://perspectives.mvdirona.com/2017/04/how-many-data-centers-needed-world-wide/¿

     Hamilton (2017) does not use the term Availability Zones and instead talks only of data centers, but he lucidly sets out all the important arguments, and the logic behind the size of data centres.
   - Gawronski, Wojciech (2021). The Complete History of AWS Outages. Available at: `https://awsmaniac.com/aws-outages/`
   - Schwarz, Arjen (2020). AWS Inside the Region. Available at: `https://ig.nore.me/2020/02/aws-inside-the-region/`
   - Quinn, Corey (2023). us-west-1: The Flagship AWS Region that Isn't. Available at: `https://www.lastweekinaws.com/blog/us-west-1-the-flagship-aws-regi`

8. AWS AZs

   - Hamilton, James (2016). AWS Global Infrastructure. *Reinvent 2016*. Available at: `¡https://www.youtube.com/watch?v=uj7Ting6Ckk&t=692s&ab_channel=AmazonWebServices¿`

9. Amazon CloudFront

   You need to gain the skill of reading User Guides for products. The extent to which you use them all is up to you (some people profess to using them as little as possible), and whether you read them in one go or make an effort to systematically cover the content of a UG. You might want to ask yourself whether you are making optimal use of all the information in a User Guide. Are you patient with them? Do you follow the hyperlink every you come across one in a sentence? It's up to you. But it's important someone tells you that this is where the information is. So, here goes: the User Guide is where the information is.

   - Mancuso, John (2012). Content Delivery using Amazon CloudFront. *TechCrunch*. Available at: `https://www.youtube.com/watch?v=3HhjoLI7Ye8&ab_channel=AmazonWebServices`
   - Amazon CloudFront User Guide. Available at: `https://docs.aws.amazon.com/pdfs/AmazonCloudFront/latest/DeveloperGuide/AmazonCloudFront_DevGuide.pdf`

- Content Delivery Network. *Wikipedia.*. Available at: `https://en.wikipedia.org/wiki/Content_delivery_network`
- Roskind, Jim (2022). Deliver Great Experiences with QUIC on Amazon CloudFront. Reinvent 2022 [Conference]. Available at: `https://www.youtube.com/watch?v=AFR7z_vce20`

10. AWS Local Zones

    This, like CloudFront, is another product. So this is a great opportunity to practice using the different intelligence sources. Consider the ones below. Do you find the FAQs page from AWS helpful? Are there journalists who have written articles about the product? Where are the journalists getting their information from, do you think? Do you find the "explainer" video from AWS helpful? Do you find the presentation at Reinvent helpful? If not, why might this be? Press releases from Amazon are especially interesting because they are supposedly often formulated before the product is built (read about "Working Backwards" in the section earlier on Amazon the Company). It's time to embark on the journey of becoming an astute and efficient researcher. Notice the ways the company tends to proliferate information. If the product was released in 2019, do you think it will be helpful to begin by watching a Reinvent presentation from 2024, where the presentation is pitched at those already familiar with it?

    (a) AWS (2024). Local Zones FAQs. Available at: `https://aws.amazon.com/about-aws/global-infrastructure/localzones/faqs/`
    (b) AWS (2019). AWS Announces First AWS Local Zone in Los Angeles [Announcement]. Available at: `https://press.aboutamazon.com/2019/12/aws-announces-first-aws-local-zone-in-los-angeles`
    (c) Green, Sam (2022). Is AWS vague about its products? April 29th 2022. Available at: `https://medium.com/@samjackgreen/is-aws-vague-about-its-products-2`
    (d) AWS (2021). AWS Local Zones: Product Overview. YouTube Channel: Amazon Web Services. Available at: `https://www.youtube.com/watch?v=GfUcmVP1gWo&ab_channel=AmazonWebServices`
    (e) Miller, Ron (2019). AWS Launches New Local Zone in LA. *TechCrunch.* Available at: `https://techcrunch.com/2019/12/03/aws-launches-new-local-zone-in-la/`
    (f) AWS (2019). AWS Local Zones User Guide. Available at: `https://docs.aws.amazon.com/pdfs/local-zones/latest/ug/local-zones.pdf#what-is-aws-local-zones`

        Let's now talk about what distinguishes Local Zones (LZs) from Availability Zones (AZs). To put it bluntly, AZs have a problem. They don't do very well for applications that require single-digit millisecond latency. A millisecond is a thousandth of a second. So, we're talking about the need to have a request go to a server, and come back with a response—all within 9 milliseconds, at most. I need not stress how extraordinarily fast that is.

A single AZ can span multiple data centres (think of the definition in the first part of this document, Active/Active). So straightaway, we potentially introduce latency when making a request to a *single* AZ. *Packets must be sent between multiple buildings.* So, AZs are **inherently** latent. Second, there is a possibility that your request involves resources in two or more AZs (e.g. an SQS queue in one AZ and S3 bucket in another).

When most people utilise AZs, they use the Internet. This means that they are using light signals in cables. It takes time for light to travel. The speed of light is about 300,000 km per second (per 1000 milliseconds). A helpful formula is:

distance = speed X time

Let's see how much ground we can cover with 10 milliseconds. Well, it is 300,000 X 0.01 where 0.01 is our time in seconds. So, 3,000 kilometres. It would take a human 25 days to walk that. That looks really impressive!

In reality, data must traverse multiple network layers, routers, switches, and various optimization steps, which introduce additional latency. Also, packets routed on the Internet are routed dynamically. The route taken varies each time based on factors such as network congestion, load balancing, and path availability. So perhaps we cannot get so far in 10 milliseconds.

Now, Local Zones are connected to over the Internet too. So, where does that leave our argument? Well, the point is (and must be) that there is benefit in bringing the computers closer to the end user, and when we maximise those benefits, what we get can no longer be described as an AZ. Notice that we bring the computers (the manipulators of the data) closer to the end user, and not *merely* the stores of static content, like with the Edge Locations in CloudFront (to be clear, one of the chief purposes of LZs absolutely *is* to allow data to reside in them). **Why is there benefit in bringing the computers closer to the end user *in a form other than an AZ*?** I presume this is because Local Zones are smaller buildings (because they can be, because they are not providing all AWS products, nor serving as many customers). The FAQs states that EC2, VPC, EBS, FSx, ELB, EMR, ElastiCache and RDS are available in Local Zones - a selective set from AWS's several hundred products [AWS 2025].

Smaller buildings can be built faster. These Local Zones may not each have “redundant power, networking, and connectivity” in the way AZs do have. (I am speculating.) So, that’s the benefit of bringing computation closer *as LZs* — the fact that we can, pretty quickly. What has been built does not meet all the requirements of an AZ and we aren’t calling it an AZ, because it’s not an AZ.

AWS have found there is demand for a *product* here. There are times when data must reside in a specific place (because of the law usually) and customers are housing servers on-premises. So, in other words, despite the benefits of the cloud such as its cheapness and elasticity, there is *still* a benefit to **on-premises servers** in certain situations. There are also times when doing the processing anywhere but locally would be unthinkable because of latency. This is the gap LZs are trying to fill. Needless to say, AWS have not realized a mistake here, such that they should have been building Local Zones since 2006. It’s not like they’ve made a massive mistake with AZs. This is because it’s most efficient to use AZs. They centralize computing resources, and provide enough latency for most people. Truly, not everyone needs the ultra low latency.

Suppose you wanted to provide everyone across the globe with access to AWS. This is essentially the question Hamilton addresses in “How many data centres needed worldwide”. How many data centres would you build? Well, a first answer is “One”. Meaning “If one would be enough, and everyone could connect to this one data centre, then I’d build one”. Well, fair play to you. There’s no point in doing more work than necessary.

But then you realise you care about reliability. You realise that a single building is inherently fragile because of what Hamilton calls “full-facility fault modes”. The best examples are **flood** and **fire**.

“Ok, I’ll build two buildings”. Notice how this revised answer is improved because of the multiplicity and nothing more. They could have a 1000 metre gap, or they could be at opposite poles of the planet. We haven’t commented on the absolute locations of the building, nor their relative locations. It’s improvement through *mere multiplicity.* It’s improvement because there is redundancy.

Now someone comes along and says “well, what if the thing that causes the one data centre to crumble also causes the second data centre to crumble”. For example, suppose the ground is tremouring because of tectonic plate activity. If the gap between your data centres is 1000 meteres, then your much-loved *multiplicity* will do absolutely nothing to fend off failure. *Both buildings will collapse.* The very ground on which things sit is the stuff of failure, the soil is sick.

The notion of a region of space, that causes failure for all things in it, is referred to as a **blast radius**. I need to say that because the term “blast radius” has become so hackneyed that we need to remind ourselves what it means. Earthquakes are a reason to start thinking about more than mere multiplicity and about putting some distance between our two buildings. Hamilton writes “Just as we discovered that a single facility eventually gets too big to fail, the same thing happens with a very large, mega-region” and he is, I think, talking about this idea of failures which cannot be solved through mere multiplicity.

Let me explain what Hamilton means by “too big to fail”. He is not stating that at large sizes, things magically gain some sort of resilience. They don’t, and there is nothing intrinsically valuable in being big. Rather, things “too big to fail” have lots of things depending on them. They are very interconnected. The descriptor is used mainly in banking; if a corporation is too big to fail then it is so large that were it to be about to fail, the government should intervene to stop it failing. The term emerged as prominent in the financial crisis of 2007. The government should intervene, so runs the argument, because the government ought to safeguard the nation’s economy and people’s jobs. Federal Reserve Chair Bernanke stated that ”the rest of the financial system and the economy would face severe consequences” upon the failing of a firm that is “too big to fail”. But ought governments do this? Once companies *gain knowledge* that they’re too big to fail, they win more investment and take bold risks, which is unfair on smaller companies who won’t be saved by the government. “Failure is an integral part, a necessary part of a market system” writes Greenspan.

When engineer James Hamilton describes a *data centre* as too big to fail, he is talking about a data centre that a lot of things depend on, whose failure would be disastrous. But I wonder about the resolution at which he intended the analogy. For instance, How, exactly, would government intervention be represented in the data centre situation? Government intervention is observing a fault in a corporation, and moving to patch it. It’s considered wrong because it interferes with the market. The market is a tool for learning which corporations can survive. Government interference *artificially* perpetuates corporations, and thereby muddies the truth about which companies are healthy. (So runs the argument.) But what truth would we miss out on if we intervened in ways that stopped a data centre from failing, given that we can see that it is about to fail? Are engineers supposed to just *look at* faults they discover now, instead of fixing them? I will have to come back to this topic.

I talked about a blast radius as X causing A and B to fail. But the blast might be A itself, and it is causing the failure of B. Hamilton's points under the sub-titles "N+1" and "Too Big Too Fail" are sometimes hard to completely separate. But I think it's something like the fact that you can have N+1 redundancy—you have plenty of data centres—and yet the failure of one of them would be so detrimental to the customer experience that you haven't really achieved much at all. Sure, you've avoided full failure, but it's a Pyrrhic victory. This might be if, say, you have a 100-megawatt facility. Maybe part of the point is that, with the engineers and the company *knowing* the failure would be so bad, AWS would be bound to monitor the centre very closely and be fearful about its failure. *The data centre is Too Big Too Fail.* So, instead, "AWS currently elects to build right around 32MW". They cap the size of data centres. This way, its more imperceptible if one fails.

So, on your mission to spread AWS around the world, you build as few data centres as possible. But that's more than one because of flood and fires. You also have some distance between them because a region might fail and it would be better if the blast or earthquake did not bring down both. So, you're current answer is "two buildings, with a trip between them". But—and I think this is the main third factor—things would still be too slow for the majority of people on earth. The round trip time just across North America is 100 milliseconds, Hamilton tells us. "Low latency is a very important success factor in many industries so, for latency reasons alone, the world will not be well served by a single data center or a single region" [Hamilton 2017]. Hamilton, however, argues that the reason a cloud provider will need a huge number of points of presence is (1) networking ecosystem inefficiencies and (2) social and political factors. "Many regions are underserved by providers that have trouble with the capital investment to roll out the needed capacity... this is one of the reasons why all the major cloud providers have private world-wide networks." Sometimes, legal restrictions mean that certain types of data must be housed locally. Hamilton concludes: "Taking both the number of regions and the number of data centers required in each of these regions into account argues the total data center count of the world largest cloud operators will rise from the current $O(10\hat{2})$ to $O(10\hat{5})$."

> In robotic surgery, the surgeon requires haptic feedback. The input at the site of the patient needs to reach the output (the surgeon) without latency [see Walmsley 2024]. In high-frequency trading (HFT), financial institutions execute thousands to millions of trades within milliseconds in order to capitalise on small price discrepancies in financial markets. If they cannot get that speed, they don't make profits.
>
> In many discussions you will see mention of Local Zones and "use cases; that is, Local Zones are not only closer to the end user but *tailored to specific use cases.* Local Zones are in strategic locations, to target "industry centers". Both us-west-2-lax-1a and us-west-2-lax-1b help people in Los Angeles (the site of Hollywood) to carry out real-time editing. Now, presumably, you don't use a Local Zone in "online gaming mode" or "robotic surgery mode"—surgeons and gamers in Hollywood get exactly the same product. So, presumably, certain metropolitan areas are seen as strategically good locations for AWS to position a Local Zone. For example, "many video editors are in Hollywood, therefore we will unveil a Local Zone in LA" runs AWS thinking. Now, perhaps the Local Zone in Hollywood is actually optimised for video editing and not necessarily for robotic surgery. This would be rational, if the preceding statement about strategic placement is true. However, I see no explicit evidence of this. As far as I can tell, Local Zones are optimized for healthcare, say, only in virtue of being optimised for low latency applications generally.

Welcome to the final part of this document, I doubt many people get down here. So, you're part of quite an exclusive club. First, bring to mind the last time you were in a cafe, club, or shop and were annoyed with the customer service you received. Suggest whether you think the following two things are compatible: (1) the business owner was customer obsessed and (2) you were unhappy with the product?

Second, consider this thought experiment. Imagine your current home. No, seriously, visualise it now. Next, think of the nearest parade of shops. Suppose you are given complete control of one of the shops and a chip has been implanted in your brain which makes you—whatever your *actual* thoughts about this concept—**customer obsessed**. You are, whatever else, endowed with a faculty of imagination. So, you have a decent budget and complete autonomy. What sort of behaviours do you imagine you might adopt in the first two weeks in charge of the shop, given that you are customer obsessed? When was the last time somebody told you that you were "obsessed" with something?