Learning a cloud

SG

July 2023

Although some YouTubers like to tell you that you can learn AWS in a day, short of plain bullshitting their sense of "learn" is

> work yourself into such a frenzy that you pass the CCP (Certified Cloud Practitioner) exam if you take it the very next day.

"Frenzy" isn't right. But it's some sort of temporary state of enhancement. It's at play when comedians get off stage, or someone does a few press-ups before a photo shoot. It's information present in volatile memory. This information could either be lost or become something more permanent. Using FRENZY to pass the exam is useful and may be necessary.

But ultimately, it's not sufficient. Stable competence takes time and requires structure. Having structure, it's reasonable that some structures will be superior. There might be concepts which really *ought* to be imparted at the beginning. I think the best candidates for this is:

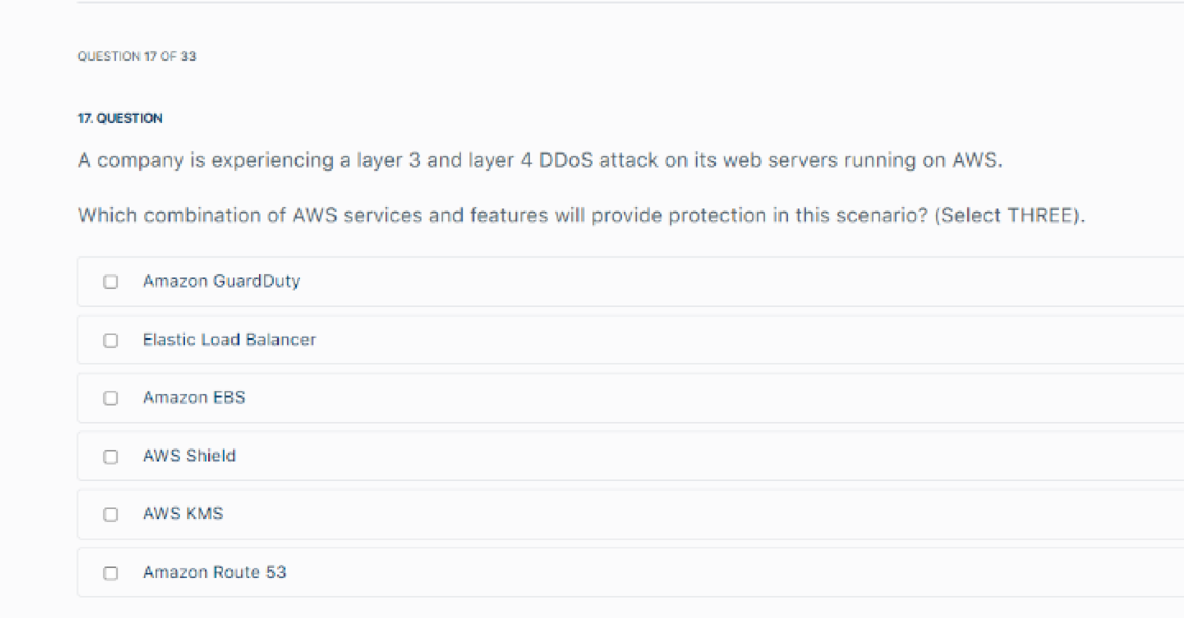
> knowledge of which authorities can be relied upon.

This grants the student **autonomy** and ensures they are aware of who is an expert. Who is to be trusted in this domain?

Unfortunately, this information will be dismissed as lacking in substance. The teacher must grin and bear this to some extent. I should add that in this piece, STUDENT is a general term for somebody learning something. It does not denote somebody at university, or anything as specific as this.

I have no intention in these notes to play it cool and provide STUDENT with precisely what they need to know and not an inch more (what supreme omniscience this implies, too). My intention is to provide an embarrassment of riches. There will be things that confuse you and irritate you, sometimes repeatedly.

Knowing more than you need



QUESTION 17 OF 33

17. QUESTION

A company is experiencing a layer 3 and layer 4 DDoS attack on its web servers running on AWS.

Which combination of AWS services and features will provide protection in this scenario? (Select THREE).

□ Amazon GuardDuty

□ Elastic Load Balancer

□ Amazon EBS

□ AWS Shield

□ AWS KMS

□ Amazon Route 53

Take the above question. If you only know security and only know 80% of your security, then it's possible that EBS is part of security. This is why you're always forced to know slightly more than you need. You need to know what EBS is—to know that it is not relevant to security.

This happens again and again for students. Things which are obviously irrelevant in the eyes of the expert demand investigation for the student. The student has to rule it out. Therefore competence in exams requires we know slightly more than we need to know.

Now, of course, if you know security so well that you know that you have never encountered something called EBS, then *this* can be used as the basis for ruling out EBS.

However, this firm knowledge is quite hard to arrive at. While you have 70% knowledge, you must admit that EBS might help with security—and you simply haven't been rigorous enough in your studying. At 80% security knowledge, you must also admit this. At 97% security knowledge, you must admit this. Only once you've reached 100% can you use the fact that *you've* never heard of EBS as a justification for not selecting it. Right until the last moment, we must know slightly more than we need to know.

How to use mock exams

Often, you'll be doing practice exams, and something will come up which you've never seen before. I was using Neal Davis and the Instance Metadata Service came up. I had never heard of this before. Variables (within the 'resource' element of IAM policies) came up. I had never seen this before. If you're given a link to a section in the documentation, you have a dilemma. Do you read none of it, or all of it?

Reading none of it will mean you fail the question the next time you see it. This is not doing enough. Reading the whole documentation, or even a chapter within it—is too much.

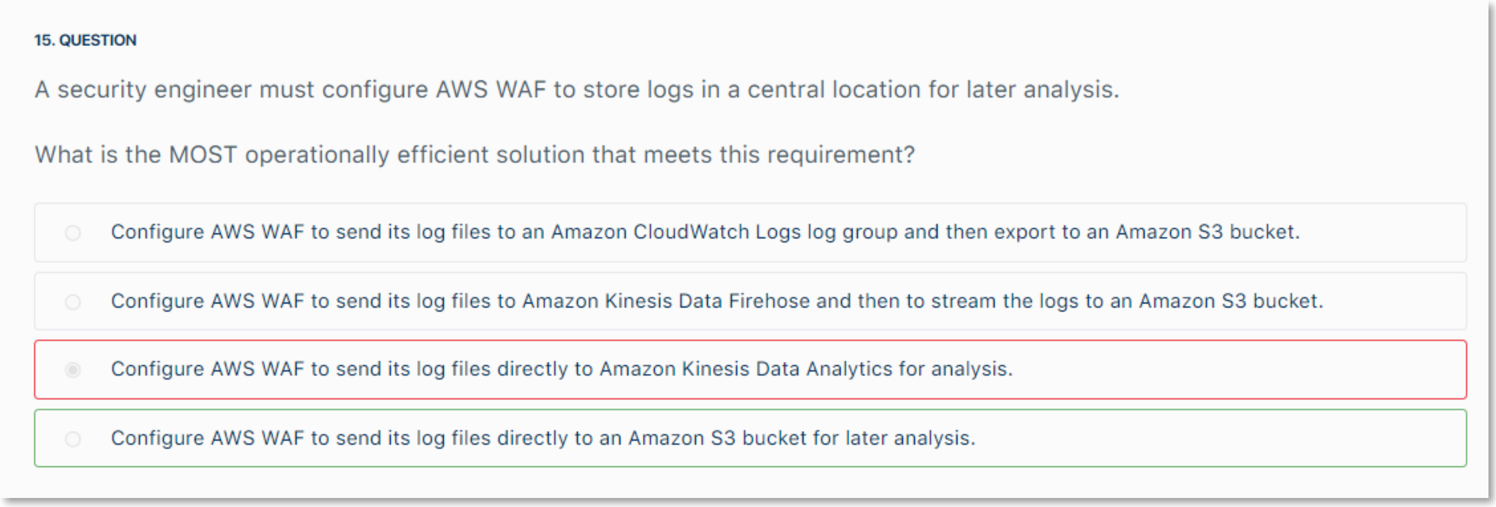
The suggestion is that we read some of the documentation. But we're left wondering how much? It seems a bit silly to pick an arbitrary value.

Sometimes Neal Davis just uses the fact that the correct answer is correct as the reason why the incorrect answer is inappropriate.

Take the question below on AWS WAF. Now, I know a few basic things about AWS WAF. I know it has features such as rate-based rules to protect against DDoS attacks, and Web ACLs and so one. But I've never read the documentation from beginning to end. Similarly, I know a fair bit about Amazon Kinesis. So, take the question below.

For all I know, AWS WAF does integrate with Kinesis Data Analytics. This is perfectly reasonable, as far as I'm concerned. Web traffic is sort of… continuous, and so similar to the streaming data that Kinesis deals with. And the question specifies that an analytics tool is necessary. So why *not* opt to use Kinesis Data Analytics? To be confident that AWS WAF does not integrate with

Kinesis Data Streams, I’d have to be a weird, massive fan of AWS WAF. I’d need that 100% knowledge.



15. QUESTION

A security engineer must configure AWS WAF to store logs in a central location for later analysis.

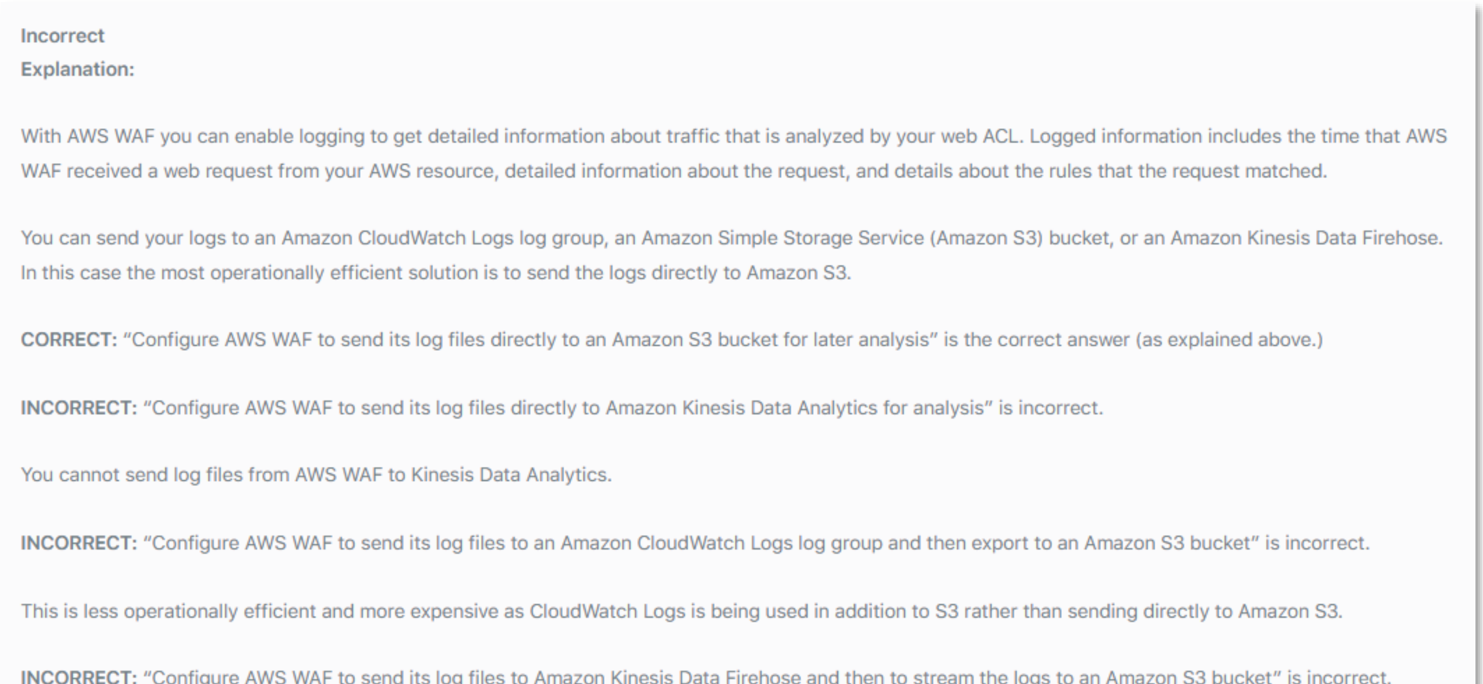
What is the MOST operationally efficient solution that meets this requirement?

○ Configure AWS WAF to send its log files to an Amazon CloudWatch Logs log group and then export to an Amazon S3 bucket.

○ Configure AWS WAF to send its log files to Amazon Kinesis Data Firehose and then to stream the logs to an Amazon S3 bucket.

● Configure AWS WAF to send its log files directly to Amazon Kinesis Data Analytics for analysis.

○ Configure AWS WAF to send its log files directly to an Amazon S3 bucket for later analysis.



Incorrect
Explanation:

With AWS WAF you can enable logging to get detailed information about traffic that is analyzed by your web ACL. Logged information includes the time that AWS WAF received a web request from your AWS resource, detailed information about the request, and details about the rules that the request matched.

You can send your logs to an Amazon CloudWatch Logs log group, an Amazon Simple Storage Service (Amazon S3) bucket, or an Amazon Kinesis Data Firehose. In this case the most operationally efficient solution is to send the logs directly to Amazon S3.

CORRECT: “Configure AWS WAF to send its log files directly to an Amazon S3 bucket for later analysis” is the correct answer (as explained above.)

INCORRECT: “Configure AWS WAF to send its log files directly to Amazon Kinesis Data Analytics for analysis” is incorrect.

You cannot send log files from AWS WAF to Kinesis Data Analytics.

INCORRECT: “Configure AWS WAF to send its log files to an Amazon CloudWatch Logs log group and then export to an Amazon S3 bucket” is incorrect.

This is less operationally efficient and more expensive as CloudWatch Logs is being used in addition to S3 rather than sending directly to Amazon S3.

INCORRECT: “Configure AWS WAF to send its log files to Amazon Kinesis Data Firehose and then to stream the logs to an Amazon S3 bucket” is incorrect.

As a student, you can feel a bit intimidated and flustered. So, on behalf of the student, let's consider the correct option. It states:

> "Configure AWS WAF to send its log files directly to an Amazon S3 bucket for later analysis" is the correct answer (as explained above).

There is certainly text above, but do we find an *explanation* here? It tells us what the logged information consists of and that there are three things which you can send AWS WAF logs *to* (log group, S3 bucket, Firehouse).

Then, the final sentence states:

> In this case, the most operationally efficient solution is to send the logs directly to S3.

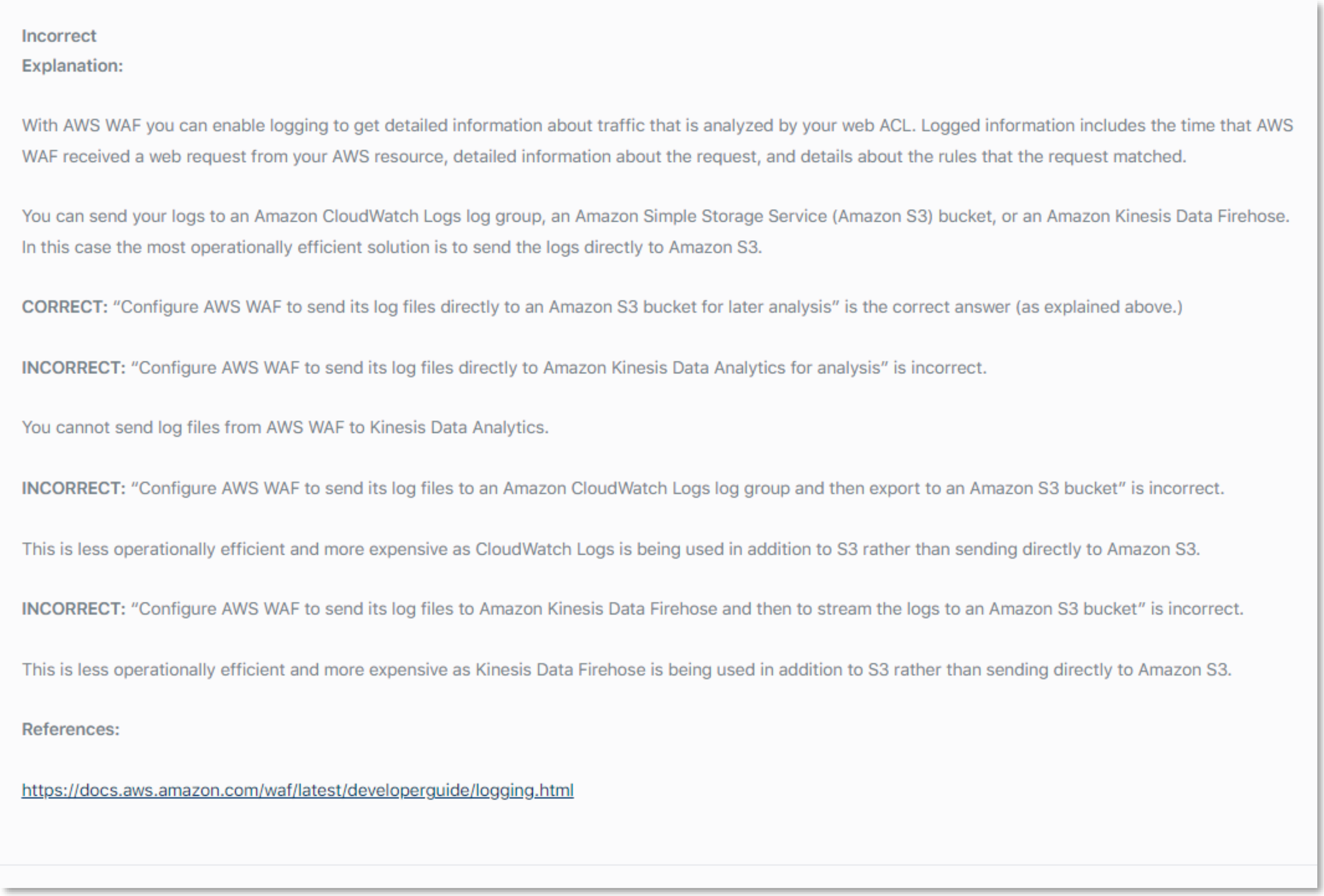
This must be the 'explanation' referred to. I hope you can see that this is not an explanation. Rather, it is simply a statement of that which requires an explanation.

We have not been presented with any reasons as to why this would be the most operationally efficient solution. It is just a bare statement – a claim.

I opted for the option which involved sending the logs to Kinesis Analytics. Let's look at the explanation for what is wrong with my thinking:

> You cannot send log files from AWS WAF to Kinesis Data Analytics.

This passes as an explanation by the skin of its teeth. *The reason this option is incorrect is that it represents an impossibility*. I think, however, that we can demand more than this from an educator. Why is it impossible? AWS listen to their customers, and generally design features that will be useful. The question is therefore, Why does the sending of logs to Firehouse (or straight to an S3 bucket) more useful than sending logs to Kinesis Data Analytics? We need an explanation for this.



Incorrect
Explanation:

With AWS WAF you can enable logging to get detailed information about traffic that is analyzed by your web ACL. Logged information includes the time that AWS WAF received a web request from your AWS resource, detailed information about the request, and details about the rules that the request matched.

You can send your logs to an Amazon CloudWatch Logs log group, an Amazon Simple Storage Service (Amazon S3) bucket, or an Amazon Kinesis Data Firehose. In this case the most operationally efficient solution is to send the logs directly to Amazon S3.

CORRECT: "Configure AWS WAF to send its log files directly to an Amazon S3 bucket for later analysis" is the correct answer (as explained above.)

INCORRECT: "Configure AWS WAF to send its log files directly to Amazon Kinesis Data Analytics for analysis" is incorrect.

You cannot send log files from AWS WAF to Kinesis Data Analytics.

INCORRECT: "Configure AWS WAF to send its log files to an Amazon CloudWatch Logs log group and then export to an Amazon S3 bucket" is incorrect.

This is less operationally efficient and more expensive as CloudWatch Logs is being used in addition to S3 rather than sending directly to Amazon S3.

INCORRECT: "Configure AWS WAF to send its log files to Amazon Kinesis Data Firehose and then to stream the logs to an Amazon S3 bucket" is incorrect.

This is less operationally efficient and more expensive as Kinesis Data Firehose is being used in addition to S3 rather than sending directly to Amazon S3.

References:

https://docs.aws.amazon.com/waf/latest/developerguide/logging.html

Generally, the explanation is insufficient. The notion of "operational efficiency" seems to consist in nothing but *using fewer services*. Is this really all that is meant by operational efficiency?

https://www.reddit.com/r/aws/comments/aa7v1p/is_it_just_me_or_is_aws_is_a_nightmare_for/

1. Teaching

Let's clarify "learning AWS". Most people have the aim of using a particular AWS service. They therefore want to be told only what is required for them to use it. There's something that needs to be achieved, and once it's achieved, their interest ceases. This is a reasonable aim, possessed by respectable entrepreneurs making things happen using AWS services.

But our aim is to gain a stable competence across a broad range of areas on AWS. I do not want a local acquaintance with one AWS service; I want a global acquaintance. That is, I want knowledge of all of them (held in a single individual, moi).

We've set aside a period of time for learning *per se*. We're here to sharpen our ideas—to improve the way we'd do things, *were* we to do them[1]. It's an indulgent period of *improvement*. This has three consequences.

Three Consequences

First, suppose there are two AWS services X and Y. There might be two ways to describe X (call these descriptions x_1 and x_2). Well, x_1 might apply to both X and Y. Suppose it does.

What are the consequences of reaching for such descriptions? Well, it's okay if you're an entrepreneur who intends to use only X. It's okay if you want to *persuade* someone to use X. And there's positive reason to reach for x_1. B*ecause* it's general—common to X and Y and other things--it's accessible. It helps customers proceed from the familiar to the unfamiliar.

[1] In the past few decades, it's become popular to take a quotation from Aristotle as a justification for no longer attempting to articulate things. He wrote "for the things we must learn to do, we learn by doing". He was contrasting things such as construction and harp-playing with smelling and tasting. *We never had to learn to smell.* I think Aristotle's quotation is usually used to justify mimicry. Usually this is following a recipe or procedure of some sort. Maybe we do learn by *doing*. But are you sure it's *you* doing anything?

However, x_1 will be unhelpful for an individual whose job is to choose between X and Y. In other words, x_1 will be unhelpful for a solutions architect, who must gain a *global* (not local) acquaintance with the services offered by AWS. Suppose x_2 is unique to X. Locating x_2 has proved to be the laborious task, casting off the x_1s by wading through the largest volume of marketing drivel I've ever encountered in my life ($x_1 = \text{scalable}$, or x_1 = managed).

Second, our sense of "learning AWS" implies gaining skills such that we have them at least a couple of years from now. Again, This Changes Things. Suddenly, pretty much *all* misunderstandings are worth rectifying. I want distinctions to be made with force and vivacity. Please, don't give me acronyms that are as forgetful as the very thing they're intended to make memorable.

We all know dramatic events in life, where afterwards, before reaching for a cigarette you remark: "there's no going back now". *This* is how you should approach topics that are confusing or forgettable. We should present them such that there's no going back. I've wasted so many hours of YouTube videos claiming to explain an AWS service and I am left at the end of it none the wiser. They're just… not in the explaining game at all. They have a textbook to sell, or have been invited on stage because their company used the product. If there are clusters of confusable concepts—or what I call **service mushes**--we're *interested* in untangling them in a permanent way. It will easily pay off in the long run.

Third, I'm interested in capacities. My belief is that imparting capacities (or *skills*) does not compete with learning facts. I think many people believe that they do compete. The common fashion is to idolize UNDERSTANDING such that the only criterion for a successful period of teaching is literally the student barking YES in response to "do you understand?" (Box ticked, have a slice of cake). Usually, social courtesy will dictate that a student say "yes, I understand". It is courteous to other students wanting to catch a bus.

Teachers persuade the student that they don't understand. Understanding arises from facts. "God is in the Detail**"** - Aby Warburg.

Skills of interest

It's been necessary to get to the bottom of a lot of differences—between GuardDuty and Detective, between RDS and Aurora, and so on. Eventually, you start to know how to

learn an AWS service.

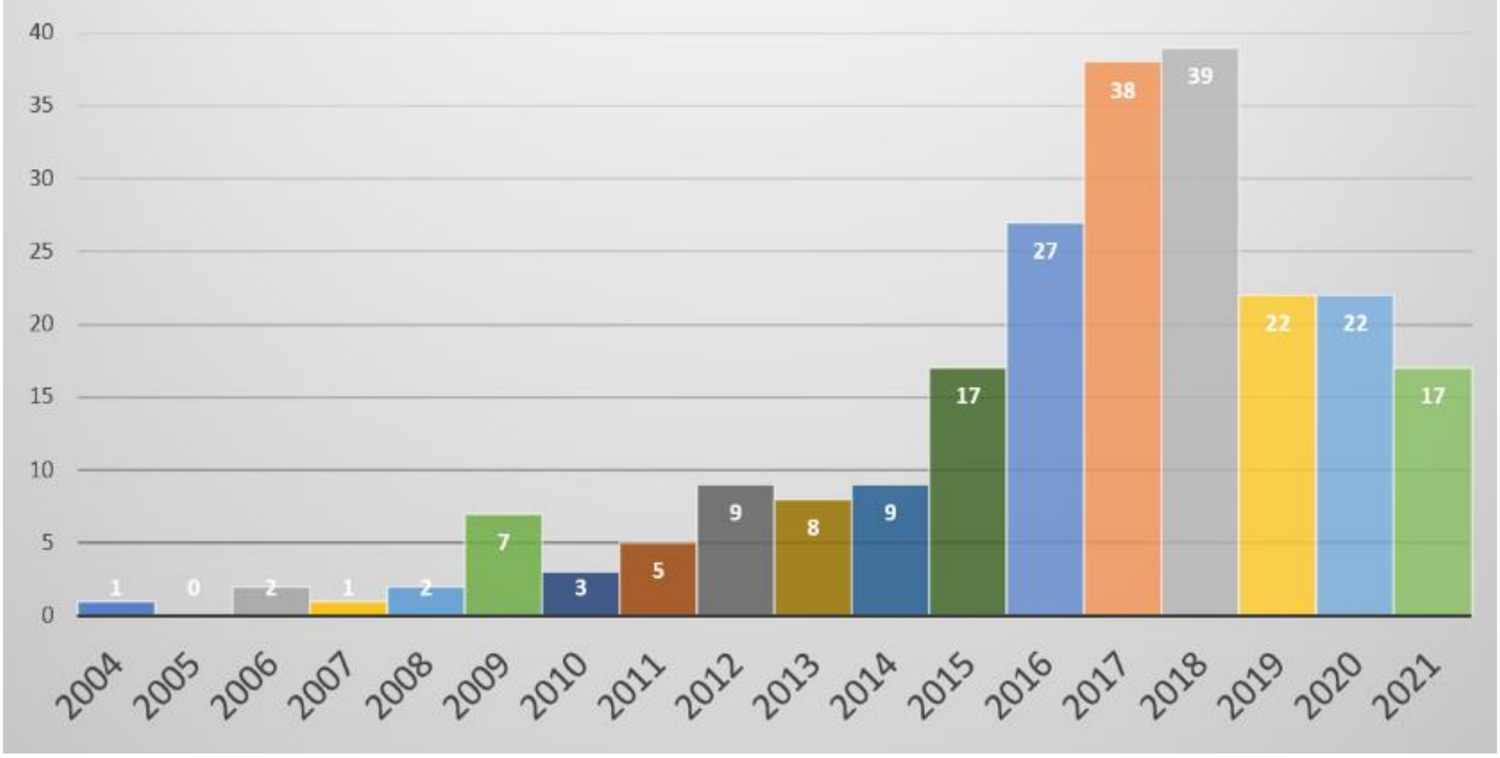
You will get better at learning the "in and outs" of a service and how to conduct proper research. You start to know how to distinguish two services. It's a skill. In such a fast-paced environment—with facts outdated by the time you have learned them—skills are the consolation.

Certainly, skills are dependent on facts, and rooted in them. Knowing how to [distinguish two services] involves knowing that StackExchange contains answers, knowing that you can filter those StackExchange question and filter those Google Searches by Year, and that Neal Davis and HowtoBeABetterDev and ChapterX of the docs can be consulted. (So, *skills are grounded in facts.*) But this should never lead us to believe that the skill has no reality.

The consolation will be knowing (of yourself) that *were* AWS to release 5 similar services tomorrow, you are capable of gaining a clear and distinct idea of each within a couple of weeks. That skill is something you have. Not everyone has it. So, as well as relentlessly hunting x_2 and thinking about long-term value, it is *capacities* I'll try to inculcate.

Switching gear

If we look at when various AWS products were released, it is clear that in the last four or so years, AWS have gone into overdrive. The **deluge** of services has produced problems for those learning AWS, which were not necessarily present five years ago. Such problems include (1) distinguishing one service from two other remarkably similar ones and (2) getting on top of the vast number of services. The first problem results in use cases that a catered for by more than one product. Multiple AWS products can concurrently cover a single use case, and sets of AWS products can form what I call clusters of confusable concepts. As for the farcical nature of the *deluge* of AWS products—this is made vivid in Forrest Brazeal's song.



Services announced per year

The most common response to these two problems is disengagement. People chuckle at the large number of services and carry on with an implicit hopelessness.

I think we should instead respond as follows. Let's begin thinking *more* about when each service came out, not less. We've certainly seen a multitude of services of released quite rapidly. But this simply means a shift of gear is required. A heightened awareness of when exactly a service was released is required. We'll emphasise time periods between one service and another, rather than downplaying it or leaving it as a footnote such that services blur together.

The same way a decent historian could give a *few* brief remarks to characterise each century, or a geographer could briefly characterise each continent, I want to be able to characterise particular years. I want a conception of a years-worth of AWS products. Years will become large containers, imposing manageable namespaces on the deluge of AWS products.

We need *some* container and the YEAR is a particularly good candidate. It represents the truth so we will be blessed with an occasionally "explanatory extra". For example, suppose you know that one service comes *after* two earlier ones which are similar. Well, you then learn which one is likely to be consolidating the other two. It's the later one. Systems Manager (released 2016) improves upon some things Amazon Inspector (released 2015) did. These are the (often implicit) explanatory extras that those who have lurked in particular industries for long periods of time enjoy.

The path toward becoming acquainted with around 200 items is well trodden. The United Nations has 193 member states and many international relations nerds go through a phase of trying to consider all of them. Some understand the whole periodic table. Not necessarily memorising them all, a student can become quite closely acquainted with 118 elements. Medical students master several hundred terms. They're sceptical that this is possible but often a mastery is nevertheless achieved.

An obvious way to bring an order to the services is grouping them into functional categories such as compute, management, storage, and networking. Perhaps this is better than grouping the AWS products by the year of their release.

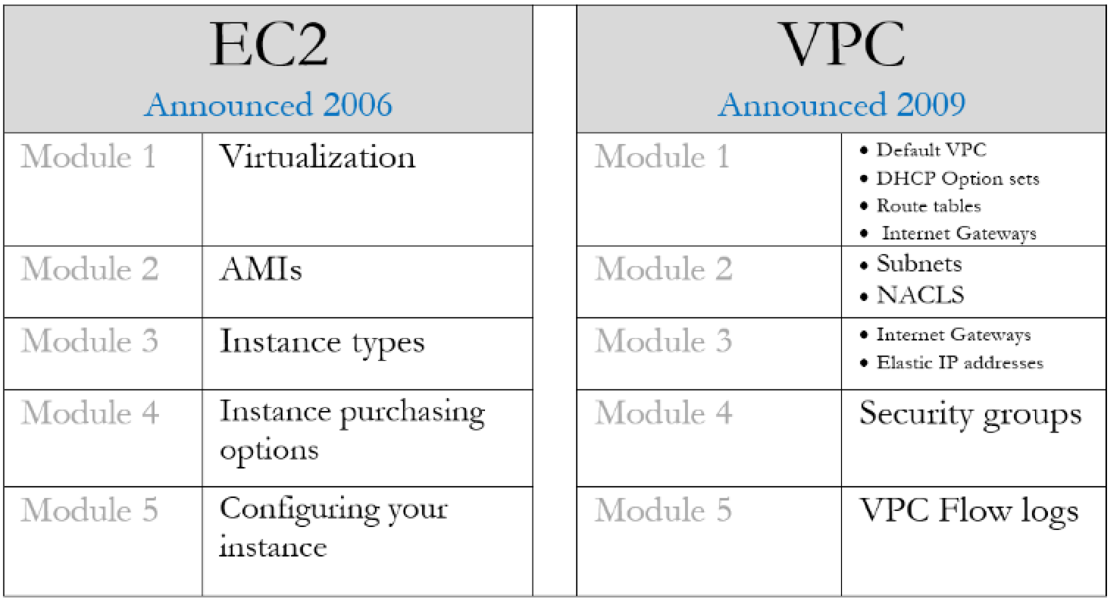
Importantly, a list of, say, storage services must be **complete**. To leave one off is to imply that you've not really understood the one left off. You're quite happy to see the world without it. The **mark of Understanding** is that the absence of the service is *striking*. Consider somebody claiming they really "get" storage on AWS—they think they have a deep understanding of it. Suppose such a person forgets Storage Gateway when listing the storage services (having been given a while to write them all down). There is a sense in which this person is akin to an alien, masquerading in their human-suit. They draw a face and nonchalantly forget the eyes. This absurd mistake is completely incompatible with the drawer being human. This aiming for completeness – it is why — recalling facts and achieving understanding – it is why these those two activities are *not* antithetical. Incomplete lists bless us with knowledge how to polish our understanding, in a helpfully determinate way.

Do not be intimidated by the many AWS services. Most likely you will surprise yourself with just how many you come to know. However, we should not go into this blindly; similar services should be considered in close proximity from the offset. Placing *existing services* at particular points in time will strengthen our capacity to distinguish them from *future (not yet existing) services* which are very similar.

Service as the unit of learning

When learning AWS you learn about an AWS service. This is the unit of learning. You learn about the features of SageMaker, then the features of GuardDuty. The AWS service is the unit of learning.

I think this is misleading because EC2 and VPC are services. Yet compared to other services, the quantity of information you can learn about these two is much larger. EC2 is about three services, in terms of time you need to spend on it. Educators tend to spend too little time on these services.



EC2 Announced 2006	
Module 1	Virtualization
Module 2	AMIs
Module 3	Instance types
Module 4	Instance purchasing options
Module 5	Configuring your instance

VPC Announced 2009	
Module 1	• Default VPC • DHCP Option sets • Route tables • Internet Gateways
Module 2	• Subnets • NACLS
Module 3	• Internet Gateways • Elastic IP addresses
Module 4	Security groups
Module 5	VPC Flow logs

We need to undo the distortion produced by thinking in terms of services—the educator should stress that there are a *lot* of details to learn about EC2 (and VPC). Divide the learning workload into modules. Above is a first sketch. I have two points to make. First, realise there's a lot. Second, step up to the challenge, by having five modules (categories, buckets) and try to put each item of info into a module, to keep everything structured. Customize the above table as necessary.

EC2 and VPC are within the Foundational Services (announced 2004-08 inclusive). Make these your bread and butter. These two *especially,* the Solutions Architect should know inside out. To become competent in these two services—know their pernickety issues, and common misconceptions—*is* to gain competence in countless other services, since other services utilise EC2 and VPC.

Hold your nerve in squeezing the detail out of them, until no acronym or jargon, no rookie procedural folly, will surprise you.

MAD STUFF

We frequently encounter x_1 in the form of MadStuff. Marketing videos want to persuade you that a service is excellent, or "mad stuff". To be clear, I am using this is the same sense of "mad skills", a term young people use to describe impressive (good) displays of skill. The service, or component, will be described as:

Managed (fully-)
Available (highly-)[2]
Durable

Scalable/secure /simple
Tolerant (of faults)
Unbelievably **easy to use**
Financially? Cost-effective
F****** Reliable!

Availability is extremely important for distributed systems, and many who use AWS value availability, and they ought to. 'Availability' has a precise meaning for engineers (it is distinct from durability, for example). Availability is a real feature of systems. Cost, durability, fault tolerance, and so on, are also real properties.

However, for the student attempting to gain a stable competence across a broad range of areas, MadStuff is useless. Do not blame the marketers; they are doing their job. (It's also an art. See Frankfurt's 1986)

Educators taking your cash, however, are not blameless. Sadly, many of them parrot MadStuff. So, you will see these terms—do not bang your head trying to squeeze meaning from them.

[2] Maybe there was a mitochondrial Eve, the first to call a service highly available, and then there was no going back. Nobody could call something merely *available* with a straight face after that. Thanks Eve.

The elements of MADSTUFF are not meaningless, but they're not going to impart understanding to you. Why is this? Well, the properties constituting MadStuff are common to *many* services. For example, many products are highly available. Because our aim is competence in a broad range of areas, we need *unique* features, not common ones. Thus, being told that some particular product is highly available is useless.

Marketers

We might also consider the distinction between formal properties and substantial properties. Some YouTube videos merely reel off *formal* properties of services such as how available it is—as opposed to *substantial* properties—in an attempt to persuade you that the product is "mad stuff". I call these videos *MARKETERS*.

It was surprising to learn how much the cloud computing field requires something like the lacerating attitude of Claude Littner interviewing a candidate. Official material from cloud providers is often saturated in meaningless claims. These claims are often a murky blend of unfalsifiable and vague. They may be honestly promotional, but they seep into the material of those trying to genuinely teach and with no real insight of their own. They have nothing real to go on, so they fall back onto the unfalsifiable and vague claims of the Marketers.

The sheer number of people across YouTube who have no qualms about imbibing this material by the gallon, word-for-word usually*,* and presenting this as scientific-descriptive. The result of all this is that the energy you need is like that found in this 2016 advert entitled "Cut through the Noise". The noise to be discarded is in vast quantities.

Verb Splatter

Here are a few other barriers to understanding you'll come across. (1) There is often something I call **Verb Splatter**. For example:

> X-ray is an AWS service that can be used to **detect, analyze and optimize** performance issues with Lambda applications.

Please don't do this. If this service optimizes performance issues with Lambda, then *obviously* it detects them. Going through "detect, analyse and optimise" is too much detail, too early. Can you imagine if people spoke like this all the time? Cars allow you to drive, commute, race, rumble and park! (Explaining to an alien who genuinely doesn't know what a laptop is-) "Laptops allow you to chat, type, videoconference, stream and compute on the go!".

(2) Being told that something is "one-click". I don't pick products for how many times I click, but I understand your point—the service is fully managed. Still, I'll be positively annoyed if you tell me it's "one click" before I've got a grasp on what my mouse activities will achieve.

(3) Being told that a service removes "undifferentiated heavy lifting". This is because Jeff Bezos mentioned in a talk this was what AWS tools achieve. Well, if you tell me about something that removes undifferentiated heavy lifting, and haven't properly explained how, I will just start to resent the slogan, and you will damage the real, important thought behind this mantra.

Watching the MARKETERS will leave you with what I call **Substance Craving**[3]. You know X is MadStuff and that X has a number of benefits, and allows you to Y. However, you don't know what it *is*. You don't know what HAS these fantastic properties.

Beware of yourself here, though. Ignorance regarding what something *is*—this is more common than you might think. The last time I defined "microwave" was probably… never. But the Microwave's utility and value, over my life, has been phenomenal. So, don't be too dismissive of expressions like "X allows you to Y". The utility of a thing is *certainly* our concern. A typical example is:

CloudFront **lets you** deliver content to millions of users.

We're only told what it lets us do. This is ok. AWS services are chess pieces not celebrity members of the Royal family. You come to know them by manipulating them with increasing skill, not by accruing more and more facts about them. That's why the first second or third chapter of a service's User Guide will be practical. It will be called something like *Get Started.*

Every service has a User Guide. It's one document, hundreds of pages long. It is a manual for using the service. This might seem blindingly obvious to you if you already know, but, here goes: the User Guides are split into sections, and each section has its own webpage. The UG is designed to be viewable as standalone HTML pages that appear in Google searches. This can mean that you forget you are reading a part of a document—part of something that works as a whole, and has a structure as a whole. And you can in fact view it as a pdf, as one document.

[3] "My name is… not important" or you asked someone what they did for a living and they said they "do what they want" (CeeLo Green).

You get better at [*the User Guide*]. The first chapter tells you the key components. Just reading that can make things considerably easier. I sometimes think if you got all the "Key Terms" chapters and read them one after another, you'd be in a reasonable state to head into an exam. Looking at what the chapter titles are, (without reading the chapters) gives you a glimpse into the major features that need to be understood, to gain mastery of the service.

Establishing a base

You need physical things to record your progress on. These could be files on a computer. The point is that you're making marks, as opposed to simply occasionally imbibing material from the cloud provider. The AWS Solutions Architect Associate Exam tests various pernickety aspects of EC2. So, I print out an actual, physical piece of paper with the chapter headings (and sub-headings) for the EC2 (and VPC) user guide. This is a good base.

Then, when questions come up which ask for highly specific details, I make a mark on the paper, signifying the chapter in which this detail is found. You're not going to read a whole user guide (some are several hundred pages), but you can start maintaining a model of its structure.

This paper—or file—gets scribbled with things over time. Reminding a later-self that you've been somewhere before (like a graffiti artist scribbling "I waz here") *does h*ave a value in terms of gaining expertise on a complicated domain. It's why annotating a textbook can be helpful. It's beyond the scope of this piece to explain why. I'm not sure I can yet. But surely it's intuitively obvious that there's a difference between

- knowing some person once researched x (or knew x or made a video on x or wrote a blog post on x) and

- knowing *YOU* did

even if you're not currently where you were.

The Curse of Knowledge

Like many domains, you will come across bad writing. A lot of things are not explained very well at all.

Oh, and no, nobody thinks they're in the Royal Shakespeare Company or that they're a poet smoking at a typewriter.

It's almost never appropriate to point out bad writing, or bad explanations. But that doesn't mean it doesn't exist. Ultimately, any scepticism about the existence of bad explanations is unfounded. Why is this attitude prevalent? Not everyone in tech had to enter it as much as others.

Bad explanations are not harmless, either. There exist, in this country and others, individuals sat alone, studying. If the confusing sentence cannot be understood, *that's it.* There's no one by the water cooler… or on a nearby beanbag to ask for advice. Nor has the student built up the knowledge to know which forum/contact it would be appropriate to consult. This would take too long. They basically just have to move onto something else. And maybe the error (in a procedure, lab, etc) prevents further steps being performed. So, explanations that say the minimum required, and—instead of providing multiple, redundant "ways" at a problem or concept—simply bank everything on one sentence, are *not* inconsequential.

Generally, people do not intend to confuse. Steven Pinker has investigated why bad writing emerges. He tells us that:

> The curse of knowledge is the single best explanation of why good people write bad prose.

(2014)

The curse rests in the fact that it doesn't occur to the writer that readers don't know what *they* (the writer) know. The writer (or speaker) doesn't bother to explain things which they already know. In a classic experiment, a child comes into a lab and opens a box of M&Ms. They are surprised to find pencils in it. This child then thinks that he always knew the box contained pencils. Furthermore, he thinks that another child who enters the lab knows that the box contains pencils.

The curse of knowledge is extremely hard to get rid of. The response is not just "think about your reader" and move on. It's an unconscious error. It is futile to just say: *know about the things you don't know*!

Does Pinker have any advice for overcoming the curse of knowledge? Leave your work for a period of time. Then, when you come back to it, you're seeing it cold. You realise how hard you are to understand.

A second strategy is to

> close the loop, as the engineers say, and get a feedback signal from the world of readers

(2014)

To do this, show your presentation or talk to its intended audience. Ask them if there's anything they don't understand.[4]

How many videos, scattered across YouTube, have been left for a month, and then returned to? I'd say close to zero. There are some excellent independent AWS educators out there. They will ask students if they understand everything. However, this isn't towards the goal of improving their textbook, or their script. They're not going to re-adjust their written material.

It is evident that some AWS employees deliver the same presentation across a year. They might do a presentation in January, June, and finally December. You can sometimes witness an evolution in these presentations.

Explain names

Dwelling on names seems silly to some people. It is not sufficient for understanding the thing named. However, I see three reasons for a brief comment.

First, the student can then recognise the service when it pops up elsewhere. Encountering intel, they have a schema under which to file it. In exam questions, getting the name *right* can really matter. That is, the student has to communicate their knowledge to the examiner, and communication uses words/names. When out and about, you can start collecting knowledge *about* the service. If you cannot even name the service, you cannot think about it. If you cannot think about it—ponder it when walking the dog—then you will learn much less about it, since we learn things by reflecting. The teacher's first duty is to remark on the name.

Second, you're acknowledging the unfamiliar environment that the student is experiencing. This is important because it's an example of empathising with the student.

[4] Pinker also describes another phenomenon:

> Why do writers invent such confusing terminology? I believe the answer lies in another way in which expertise can make our thoughts more idiosyncratic and thus harder to share: as we become familiar with something, we think about it more in terms of the use we put to it and less in terms of **what it looks like** and what it is made of. This transition, another staple of the cognitive psychology curriculum, is called functional fixity.

Sometimes, writers forget to state the obvious: what does it look like? How do I know I'm looking at an IAM trust policy? Dawani (2019) is good in his 2019 webinar on VPC flow logs because he shows us what a log looks like.

This in turn is important because it renders *you* worthy of attention. Why pay attention to something that cannot empathise with you? (Well, there are plenty of reasons. However it is still true that somebody somewhat interested in your concerns, tends to be *more* interesting)

Third, you're recognising the student's rationality since you are treating them as *the kind of person* that expects things to be explained (or explainable). It's a mark of respect.

Break down compound terms. If you were explaining the concept of a **binary search tree**, it can be helpful to say why it is *binary*, why it is a *tree*, and what makes it a *search* tree. In other words, what specific aspect of a binary search tree makes it binary?

Consider "elastic compute cloud". We can explain what elastic means, and how it related to the idea of increasing and then automatically, and easily, decreasing capacity. We could comment upon how this elasticity is common to lots of AWS services. So, we can then explain what 'elastic compute' is. Finally, we can explain why it is a cloud—the fact that the servers are remote, and that you are being given a logical subsection of the data centre.

Some names have no explanation. Why have AWS released a service called AWS Fargate? We do not know. (I only know that this is the name of a high street in Sheffield, England. AWS Fargate is a container servicer, and in a surreal turn of events, Sheffield council recently positioned a number of physical containers in Fargate, to the uproar of local residents.) The same applies to Snowball. In these situations, remark upon this feature of the name. That is, remark upon the name's inexplicable nature. That *is* a feature.

Being capable of referring to things is particularly helpful for learning things. It is more important at the learning phase than the knowing phase. In many experts, the referential ability to dropped away (or never needed to be there). Yet naming things is important for those learning.

Arbitrary categories

Sometimes you come across categories in User Guides. They are quite arbitrary. That is, they are uniform. Examples include the pricing models for EC2, and the two types of queue in SQS.

The Elastic Preface

Every service name is prefaced with either "AWS" or "Amazon". We should get it right (for example, we should not call it AWS Inspector because it is Amazon Inspector) because otherwise it's inconsistent with a general principle of paying close attention to words. It's not clear what causes a service to have one variation over the other - an annoyance that has been noted by Alex Antra (2019).

My hypothesis is that for some services, the marketing department insist on *Amazon* because people unfamiliar with AWS are likely to make up a substantial portion of its target market. Thus, we get Amazon Connect which is for call centres, removing the need for those who work in the call centre industry to ask what AWS stands for. Similarly, we get Amazon AppStream, which allows applications to be made available in a browser. This is a quite abstract service, not necessarily enabling engineers to play with underlying infrastructure, so the preface is "Amazon" since it is intended to attract those without strong interest in AWS. Similarly, we have Amazon Lightsail – intended to make building applications feel as light as moving over water, instead of drowning in the details. Amazon WorkSpaces is for employees of corporations to have their workspace (or desktop) accessible from anywhere and Amazon Polly reads out text like a parrot who isn't just pretty. You ought not to have to learn about an servers and compute to appreciate Amazon Polly. It is completely managed. So it would be wrong to have "AWS" as the preface and suggest people ought to know what AWS stands for.

Elsewhere, the need for the product name to have a zing to it among a popular audience is not so strong. The marketers permit Batch, the product for batch computing release in 2016, to pay homage to the locus of infrastructural expertise that is Amazon Web Services. Its audience is computer experts, and so having the clunky but prestigious "AWS" in the name may in fact be an advantage. It is AWS Batch.

Similarly, it is AWS Systems Manager and Trusted Advisor, Transit Gateway and Global Accelerator (for the networking nerds), Certificate Manager and License Manager, Lambda and Step Functions, Config and CloudMap, Control Tower and Security Hub. Anything associated with the prestigious history of *compute* expertise gets AWS. It is *AWS* Auto Scaling which is the service that spins up and terminates your virtual servers autonomously. But still, anything sufficiently gamechanging and impressive must bear Amazon – Amazon EC2, VPC, CloudWatch.

This is a nice hypothesis, but there are a few counterexamples. Being a service focussed on abstraction, I can easily imagine AWS Amplify being Amazon Amplify. And given that other security services such as Security Hub and Control Tower bear “AWS”, I think it should be *AWS* GuardDuty.

Anyhow, our official term for this dynamic preface (AWS or Amazon) is the Elastic Preface. If I’m discussing a service, I will deploy the Elastic Preface the first time (I do have a faint interest in allowing a habit of correctness regarding the preface to develop). But then it **must be dropped** for the remainder of the discussion to promote efficient communication. Amazon Inspector becomes Inspector. If we’ve entered the appropriate namespace, I can just say *Inspector* now.

Console Demonstrations

Many presentations of AWS products at Re:Invent involve a demonstration on the Management Console. Sometimes there is no demonstration. YouTube comments complain when this occurs.

One sympathises. Seeing the Management Console can show you things that would take an age to verbally communicate. It shows you the buttons. Buttons marks the contours of capabilities. The structure of the panes is indicative of a great number of implicit things.

A visual curse of knowledge

However, because of the Curse of Knowledge, experts can forget how accustomed they have become to a console. It looks familiar and clean, with time. The first time you use it, it can be an intimidating mass of buttons and switches. There’s a sort of *visual* curse of knowledge. The student isn’t looking where they ought to on the console. The student is analysing things, trying to work out what’s going on. The expert analysed and dealt with those things (present on the screen) long ago.

It is possible to feed the console to the audience, presenting visual blocks one by one. Do this alongside your words, which can explain things. Usually, it’s the up-front model; visually present all the capacities of the service *up-front.* Why?

- It can be easier to watch console demos on mute. Feel comfortable repeatedly pausing them and letting yourself taking take in everything that is on the screen.

2. Triangulating Services

“In the Old-fashioned Days, you’d read an instruction manual and then start using something.

Nowadays it’s different. If you want to understand an AWS Service by Friday, you’ve **got to take a long run up**--you’ve got to start weighing it up Monday/Tuesday.”

There is a serious point here. I‘ve found that the first time you *try* to learn about a service, I’m essentially just marshalling resources. I’m going through a select group of YouTube channels (see below) to check for relevant lectures. I’m assembling them in a folder or playlist. I’m perhaps taking the User Guide and using an online tool to “chop up” the PDF file. (The User Guide is often a few hundred pages long.) Often, there the first few pages that go through the key terms or concepts. I’ll look in my textbook by Neal Davis, to see what he has to say on it, then I’ll look at the textbook published in 2021 (Sybex) by Piper and Clinton. So, it’s a process of selection. Lots of admin. Lots of roughage. Frankly, I’m not learning much. I’m laying the ground for the *next* time I study this service. That’s when the magic happens.

To learn an AWS service, you’ve got to learn it a few times first.

Look at different angles on it. Have some central word document, where you make notes.

You could then systematically go through:

1. Printed textbook by Neal Davis (2021)

2. Printed textbook Piper and Clinton (2021)

3. AWS FAQs page

4. AWS official documentation (as a pdf)

5. A triad of official YouTube Channels:

6. YouTube videos from independent training centres

7. YouTube videos from individuals

8. Emails from Corey Quinn (*Last Week in AWS*)

9. Check if there is a Wikipedia page for the service.

The idea is that each individual source is quite dilute, in terms of the useful information is provides. There's lots of MADSTUFF. So, we have to be like a chemist. We have to patiently distil a more concentrated solution.

You examine one source, and then note down any questions you have. Be brutally honest. If something seems bizarre, or doesn't make sense, let it out. Give voice to it.

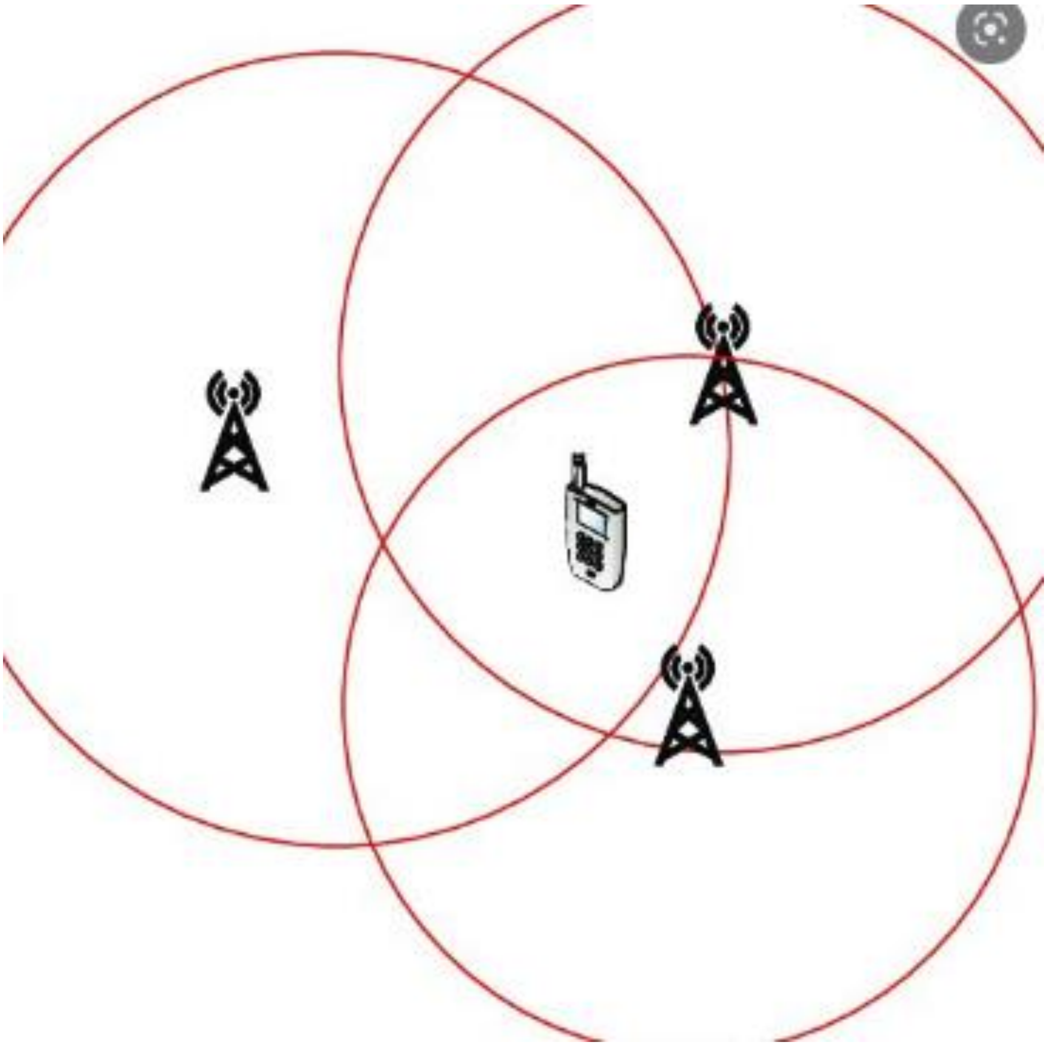
> Really, it's more about self-respect. Take your own, confusions seriously. *Respect* your own misunderstandings—there's likely to be something in them.

(What you're doing here is sort of artificially manufacturing your own ***interest*** in the details of the service. Some button on the console (or whatever) is made vivid. Then you re-watch the video with this in mind and it hits different. *Anything* can be made interesting. Christopher Nolan made millions by filming a man opening a safe, just by building up to it properly).

Then, you look at another source. This could be a speaker at Re:Invent. They will say something slightly different or express something differently. For example, I wanted to know why there were two sorts of VPC endpoint (gateway- and interface-endpoint). I stumbled across Gina Morris's talk at Re:Invent, and she said "Gateway" essentially just means "routable". And this made it all click. It explained many other services which have gateway in their name. Up until now, no speaker had explained what "gateway" contributes to the expression "gateway endpoint". They just treat it as a single word. (Perhaps because it is obvious to them).

So, you've noted down questions you have. On looking at a different source, some of your questions will be answered. Repeat this comparison of the sources. It's an iterative process.

You can even take screenshots from the console (viewed through YouTube channels). Using PowerPoint, you could put them alongside one another. If you select from all these source you can generate a central document that is really rather authoritative and comprehensive. It becomes more intelligent than any individual source.



different sources.

RETROSPECTIVE EDIT – In terms of the above, and the numbered list of sources. Take any form of video down a peg. Videos tend to be vacuous and slow. That said, at Re:Invent 2022, there were some really good presentations. Although it's more painful, study the User Guide carefully, from the offset. Look a *little* bit at the bigger pictures – independent news articles, Reddit posts, blogposts by developers. It's hard to use Google to find these, but push for them (go to later pages on the results etc). Finally, Jeff Barr's blogposts are really rather well written. Often, even though many years have passed, and others have introduced the services, Barr's introduction is still the best. They're not deep, but they are clear.

The Owl of YouTube

A man once said ‘the owl of Minerva spreads its wings only with the coming of the dusk’. It was used to express that idea that when you can properly reflect on something only once it has passed.

Jokingly, this refers to the tendency of an incredibly helpful video to only be revealed by a recommendation engine once it’s too late. Only at the *end* of 3 hours researching Amazon Simple Email Service, will YouTube subtly drop the perfect video into your “recommended videos”, just as your drooping eyelids can no longer be resisted.

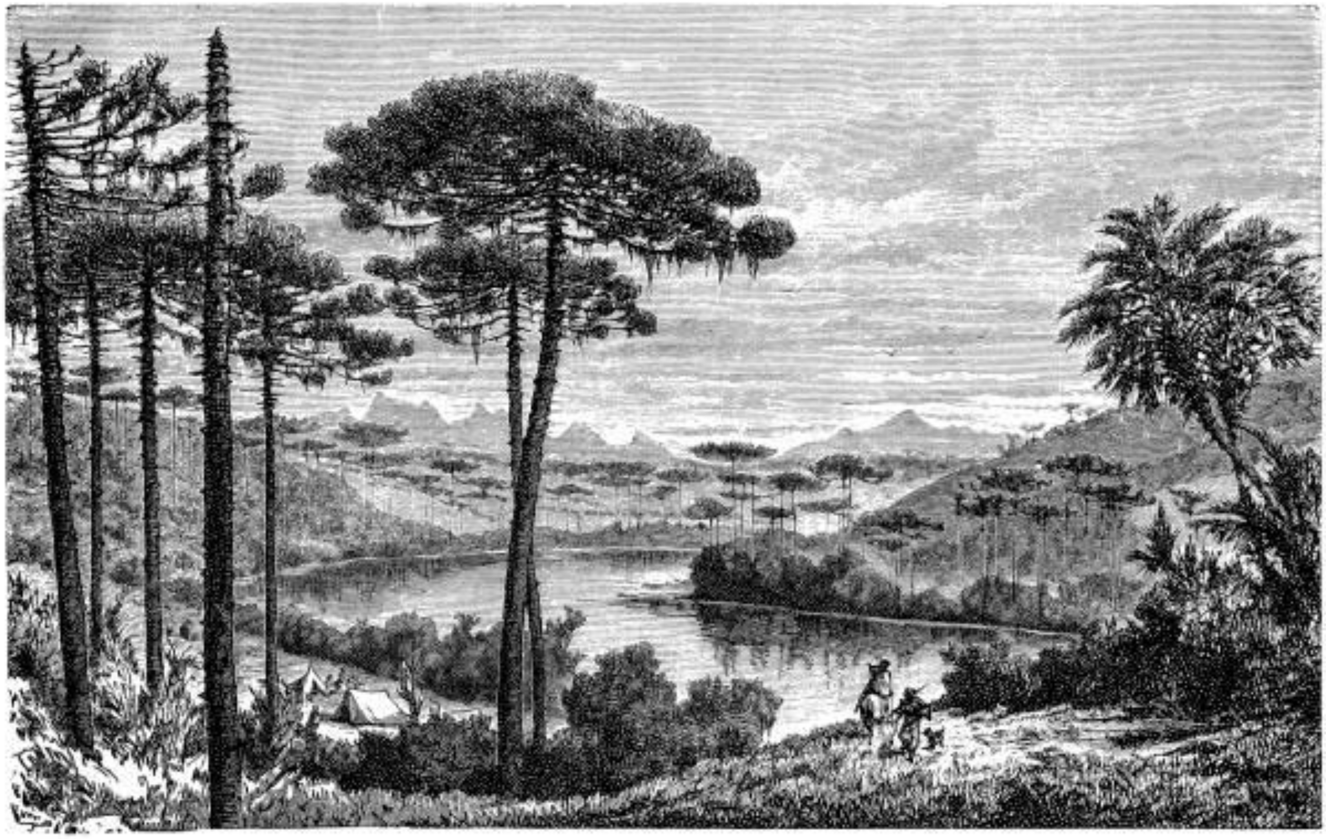
To counter this, maintain RECORDS such as playlists, and lists of helpful videos. Each time you return to the service, you’re in a stronger position, and have marshalled more resources.

Recently, Enrico Tartarotti produced this YouTube video describing the very recent thinning out of Google search results:

> https://www.youtube.com/watch?v=48AOOynnmqU&ab_channel=EnricoTartarotti

This really explains a lot. News articles and ycombinator forums on AWS services never used to come up for me (at all). Now I know to look for them, proactively. However, I have never had to be as proactive as this for other things, in my life, on Google, up until now. The change in Google’s behaviour has been damaging, no doubt.

The Flux Factor



You never step in the same river twice blah blah blah

AWS is constantly changing. Particular services are changing, all the time. New features are being added (for example Multi-Attach on EBS). Names of service are changed (for example Elasticsearch became OpenSearch). The visual layout of the management console changes. What AWS says, *goes*. We're constantly playing catchup.

Educators (such as textbook authors) might put images in their textbook. Because AWS changes frequently, these images fail to represent the actual console quite quickly. Even descriptions about what products *are*, and what they do, can become misleading quite quickly.

This is irritating at best, daunting at worst. AWS are a (corporate) agency, basically free to do whatever they want, whenever they want. They can just edit a web page right now. If you've written a textbook or article quoting this page, then it's immediately made invalid.

It's possible there's no record of the old version, *anywhere*. They can do things on a whim because it's their product. We can conceive of AWS tomorrow consolidating Auto Scaling and ELB under a *single* service, called Elastic Balance. We can conceive of them removing the health check feature from Route 53. It's their choice. They can carve up the universe however they like. Different capacities will fall under different services because certain teams happened to form in the internal organisation. There's nothing *to* understand. We're reduced to footballs fans following a team.

How do we respond to this? Well, first, be aware that we have overstated things. AWS are interested in people understanding how to use AWS (*insert mumble about it not seeming this way*). It is therefore in AWS's interests to not re-jumble their services on a whim. It's reasonable to assume that they have a vested interest in maintaining a certain degree of stability. It is indeed true that there is a stability. They are committed to never changing their APIs, for example.

To some extent, the feeling that things are changing is illusory. The more you come to know about AWS, the better you are able to perceive subtle discrepancies. *The more acquainted you become with AWS, the more it will* appear *to change.*

You will occasionally learn things that become false. You are still making progress, though. Focus on the net gains. Learning AWS is like walking up an escalator, which is moving slightly.

Anyhow, educators should be more conscious of the flux factor. I propose that we focus slightly more on specific things said, by specific people, at specific times. I will cite my sources. The advantage of this is that if my explanation fails to "hit home", then the student can "failover" to the sources I have relied on. Hopefully, this won't *always* be necessary (after all, the whole point of the educator is that he has summarised, and perused the vast primary sources, so that the student does not have to)—but failing over should always be possible.

Objectivity

I will avoid saying things such as

(a) There are 3 Auto Scaling policies.

This is because this might become false in 2 years time. I will instead state

(b) John Smith stated that there are 3 Auto Scaling policies (2012).

The second example is grounded. That Smith stated this will *remain* true in 2 years time. Like it or not, and however things have been re-jigged, it will always be true that John stated X at this point in time.

The second example is longer, and more inefficient. However, there are ways to implicitly reference – to implicitly ground what you are saying.

There are benefits to this latter approach. It is immediately clear to a future student that:

(i) There may *not* be 3 Auto Scaling policies, because 2012 was a while ago and

(ii) In 2012, there were only 3 auto scaling policies

If I just said "there are three auto scaling policies", then we don't necessarily get these two bits of information (i) and (ii).

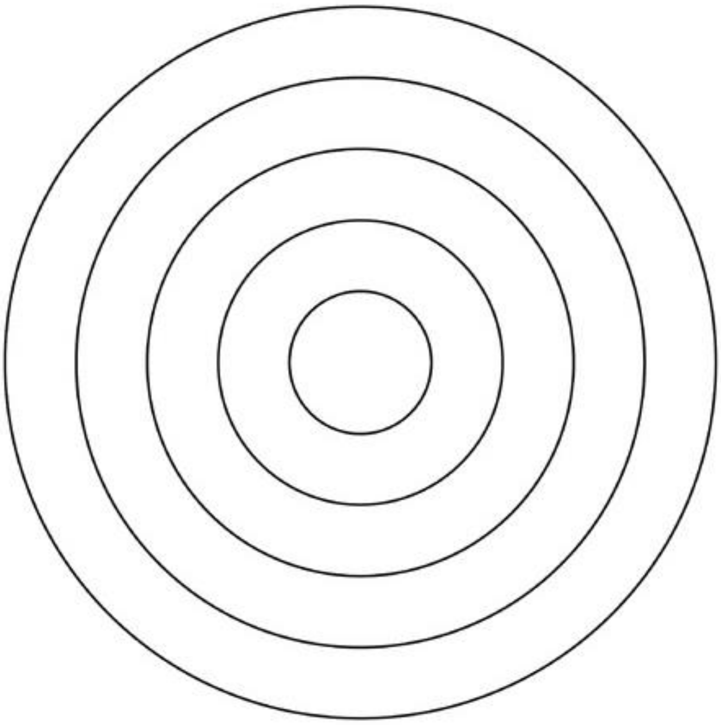
The fact in (ii) can be quite helpful for seeing how a service has evolved. It will make it easier for the student to make sense of *other* sources, such as old YouTube videos they encounter. They can now accommodate the fact that the YouTube video is claiming that there are only three policies (despite there now being nine, say). They can explain it, rather than be confused by it - *The video is from 2012*.

Many educators sell their books and just parrot things said by AWS employees. Sometimes, they do not understand what is said.

The conception of objectivity I'm using does not involve stripping away any judgements or emotion. Suppose John Smith (of AWS) presents his view of AWS DeathStar. Objectivity involves this:

> I form a new viewpoint,
>
> which has [Surname's viewpoint of Deathstar]
>
> as its object.

I do not adopt Mr Surname's viewpoint, parroting his words. Objective doesn't mean I'm going to switch off all my critical faculties and aim for some sort of "view from nowhere". I'm not trying to strip things back. I'll clarify and improve where I can, occasionally making errors of interpretation—noticeable by readers—which they can then avoid. They can view *my* work objectively. So, we get concentric circles. "In this process of objectification, nothing is left behind" [Dancy].



Teaching as architecting

Teaching is not just being a source of information. There is Source and Recipient, and if you happen to be the Source, you're the teacher. So, teaching is just telling people what you know.

Recently, my admiration for teachers has grown and I've realised just how absurd it is to regard teaching as telling. It's akin to viewing reading a novel as "printed-word-into-mind", and supposing writing a novel to be *merely* the process in reverse (mind-into-printed-word). A 6-year-old might have this impression They'd wonder whether writing Harry Potter or reading it takes longer. We should as adults see this to be absurd. Writing novels takes years.

Viewing teaching as "telling" is a misconception akin to that of the child. of. I've not fully worked out what to do with this yet. The teacher probably needs to provide impressions which are forceful and vivid. To teach X requires some competence at X, but not to the extent generally supposed. This is not least because teachers must be well versed in the thoughts that occur *in students*. Canvassing these thoughts will take time. And learning how precisely to impress concepts vividly will take time. So, the teacher *can't* spend all their time practising X. Their time is spent getting better at teaching X. This is why the observation that "those who can't do, teach" loses some of its bite with reflection.

The teacher needs to be able to abstract, express, and impress. Those are three stages: take in what's important (abstract), succinctly articulate it (express), and such that it the understanding *lasts* (impress). Teacher Edward Hughes

> graded his approach, more than many would, to the capacity of the student, challenging the bright, and leading the slow slowly.
>
> His first explanation of any subject was always in simple pictorial terms, his object being to establish a feeling of contact with the central idea, before the rigours of its sophisticated formulation had to be faced.

[Ingold]

Many professional teachers today enjoy writing articles about how millennials are spoon-fed facts. This almost always is a justification for the teacher to cease communication (so as to "promote" independent study), or to cease communicating facts. This is ridiculous. Rest assured, students will have *decades* of independence from you.

Whatever teaching is, it involves interacting, explaining, elucidating and informing to the best of your abilities. Communicating the facts and inculcating capabilities (such as critical thought) are compatible!

Service Mushes

Some groups of AWS services appear very similar. I call these clusters of confusable concepts *service mushes* (a play on Service Mesh, the name of a genuine AWS product). The items in the mush might merely have similar names, or they might genuinely have similar functions. Realising you've been confusing one service with another is miserable. When textbooks do nothing to pre-empt this, or account for this, you feel like you've been taken for a ride. I count twenty mushes.

CCC-	Services	Notes
1	CloudFormation CloudFront	
2	Inspector GuardDuty Detective	
	CloudTrail GuardDuty	
3	AWS Shield AWS WAF GuardDuty AWS Network Firewall AWS Firewall Manager	
4	AWS Auto Scaling EC2 Auto Scaling	Yes, you think you're aware that auto scaling is an elderly service, around since 2009. Then you question reality, reading somewhere that it was launched in 2018! (the general, *AWS* version was launched recently)
5	CloudWatch CloudTrail	Both have "Cloud" in and they both deal with logs.
6	EC2 VPC	Yes, that's right, you can have a moment where you are concious of each, but haven't *quite* thought about their coexistence yet. I think Security Groups could conceivably be managed by either.
7	AWS Budgets the Billing console Cost Explorer	Sorry to break it to you, but in the normal world, *budgets*, *billing*, *bill*, *cost*, are basically the same words.
8	Athena Aurora	They're both sound majestic, are both goddesses, and are three-syllable words beginning and ending with the letter "a".
9	Athena Redshift Spectrum	
10	EBS EFS	This is basically because these two acronyms begin with "e" and end with "s". This distinction is generally made explicit at the outset by most authors, in fairness.
11	EC2 Auto Scaling Elastic Load Balancing	Especially if you are new to computing, you might have a moment where you realise these are different things. They both

		deal with instances and how well they're doing.
12	RDS Aurora	The former is Relational Database Service and the latter a relational database service…
13	KMS Secrets Manager [Systems Manager Parameter Store]	
14	Amazon MQ Amazon SNS	
15	SNS SQS	I know this will seem silly if you are more advanced with AWS. However, for the beginner these appear similar: both are SxS.
16	EFS EMR EBS	
17	WorkSpaces WorkMail WorkDocs	
18	WorkMail SES	
19	CodeDeploy CodeCommit CodePipeline CodeBuild	These are *never* differentiated with enough clarity.
20	Systems Manager AWS Config	
21	CloudSearch ElasticSearch Service OpenSearch service	
22	AWS Proton CloudFormation AWS Service Catalog	
23	Elastic Beanstalk Lightsail	
24	FSx Storage Gateway	
25	Elastic Kubernetes Service Elastic Container Service AWS Fargate	
26	Lambda Fargate	

27	Amazon Kinesis SQS	
28	AppSync AppMesh AppFlow	
29	Data Pipeline CodePipeline	
30	DataSync AppSync	
31	Data Exchange DataSync DataPipeline	
32	AWS Artifact AWS CodeArtifact	

Some of these services mushes can be removed from Student's mind with some simple tinkering. For example, a brief distinguishing of [block storage] from [file storage] deals with one of them. The names EFS and EBS are only superficially, *linguistically* similar. Some mushes require some analytical elbow grease as they are *conceptually* similar. Consider 12, 13 and 19.

I think it is better to address these mushes from the offset. If we were to have a session where we put, for example, Inspector, GuardDuty, and Detective in close proximity, we would have a sharper conception of each than had we looked at them individually.

Unfortunately, though, getting in the habit of going "here is X, which is not Y, and not Z" will be irritating.[5] The teacher will seem to be overly defensive and there is also a laziness to it. The teacher should present X in a way that is so crisp and vivid that it could never be confused with Y nor Z. Furthermore,

[5] "it is perfectly reasonable that the author should consider the objections and possible misunderstandings… the odd thing is that he or she should put them in the text." Bernard Williams, *Philosophy as a Humanistic Discipline*, p182.

> Novices won't even know about Y or Z, so such ways of talking will be useless anyhow.

Instead, we should aim to present X in such a way that it never *could* be confused with Y or Z.

A teacher should **be Present** and try to originate the facts. If they reason about what the facts *must* be—what they *could* be—out loud, then they can **model** critical thinking[6]. This requires practice and courage on the part of teachers. But there has to be a real possibility of your failing. There won't be such a possibility if you're *relying* on a PowerPoint or script. You can't program meaning into words.

Many in the IT world believe that there is no discernible difference between a script which agrees with what the speaker is saying, and a speaker who is moulding their words to the script. In both situations the same words are spoken. Only the latter results in adults reading from autocues in utterly bizarre performances, in which they appear like robots (which are tools).

3. Characterising years

I have established already that I intend to group the AWS products into years. First, however, it may be helpful to step back and categorise the years into a number of broad *Periods*.

Here I present a way of grouping the years during which AWS has existed. Each period is 5 years. There is the Foundational period, beginning in Jan 2004, the Furnishing Period, which started with a what Werner Vogels described as a **triad** of services (CloudWatch, ELB, and Auto Scaling).

The beginning of 2014 marks the beginning of the Aggressive Expansion period—five years of increasing the number of services announced each year.

[6] Jassy is a fan of business mentors *modelling* behaviours.

i. Foundational Five

Jan 2004 until the end of 2008

The embryonic first five years are when the foundations are being carefully, accurately laid. In this period, a year will tend to contain the release of one service and never more than 2. The very concept of an **AWS Service** is still protean.

2004	SQS	
2005		
2006	S3	EC2
2007	SimpleDB	
2008	EBS	CloudFront

ii. Firmament Furnishing

Jan 2009 onwards

It's 2009, and in the words of the Black Eyed Peas, "here we come, here we go, we gotta rock". AWS was developing a swagger. Its rate of releases suggesting an attitude of "hop in there we go, I got places to go."

A five-year project of installing the firmament of the AWS universe was embarked upon. Each year, a… handful of services emerge in a robust and deliberate manner. Over these five years, AWS is flirting with the idea of releasing ten services, but never quite gets there.

In 2014, AWS finally break through, and release more than ten services in a single year, for the first time. For this reason, 2014 marks the beginning of a distinct period.

2009	CloudWatch	Auto Scaling	Relational Database Service	Elastic Load Balancing	Elastic Map Reduce
	VPC	Management Console			
2010	SNS	IAM	Route 53		
2011	Simple Email Service	Elastic Beanstalk	CloudFormation	Direct Connect	Elasticache
2012	DynamoDB	Trusted Advisor	Simple Workflow	CloudSearch	Glacier
	Data Pipeline	Storage Gateway	CLI	Redshift	
2013	Elastic Transcoder	Kinesis	CloudTrail	WorkSpaces	AppStream

	AWS CloudHSM	AWS OpsWorks	Management Console Mobile Application		

iii. Aggressive Expansion

It's now precisely a decade since the Simple Queue Service. In 2014, the Russian Federation attempted to expand, to include Crimea. Also, Harvard scientist Doug Melton demonstrated that vast quantities of insulin-producing beta cells could be produced. This was a great step forward for diabetes.

Starting in 2014, we saw five years of constantly ramping up the number of services announced each year.

The AE Period

The five-year period beginning in 2014 I refer to as the AE period. AE could stand for "Abandon EC2", since this period kicks off with the announcing of Lambda—the serverless service.

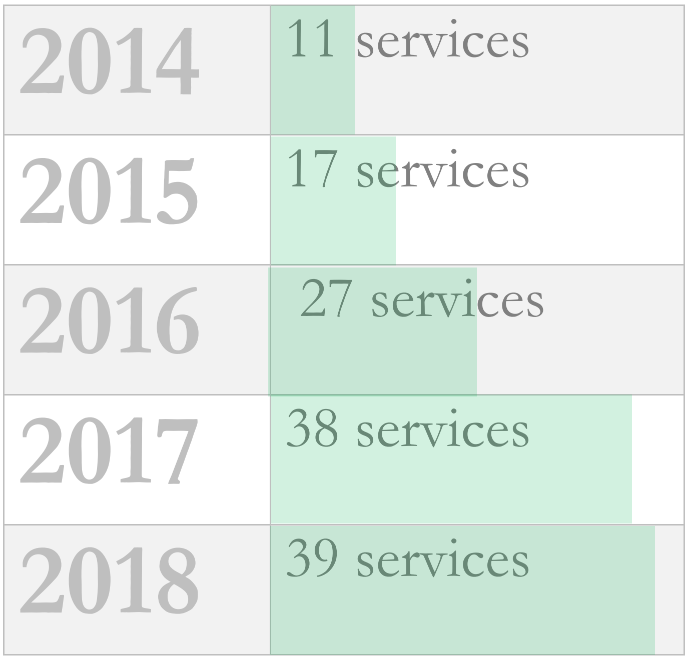
AE can also be understood to stand for Aggressive Expansion or Exploration, since this era culminates in 2017 and 18.

> In November 2014, the tech industry was the audience to an announcement that was to set off an illustrious trend in the domain of cloud computing known as serverless. That announcement was the **release of AWS Lambda**, presented by **Werner Vogels** at AWS's re:invent summit held in Las Vegas. Since then, the industry has seen strides in the way we compute in the cloud, reshaping business and technological developments.
>
> Five years later, almost in the poetic quote of history repeating itself, AWS has done it again. This year July, in NYC, **AWS announced its Eventbridge service**, and the uproar and excitement that followed were the same as that seen five years ago. Almost every blog or report on the service that came out afterward, depicted

> Eventbridge as the most important announcement after the release of AWS Lambda.
>
> Sarjeel Yusuf

AWS kicked off this period of exploration by announcing Cost Explorer. This period culminates in **the Prodigious Pair**: two years when things go berserk. The Prodigious Pair (2017 and 18) marks the end of a trend of constantly topping the previous year (2019 had just 22 announced).

The Aggressive Expansion period mirrors the Foundational Five in a pleasing way: we had 2004-8 and then 2014-18.



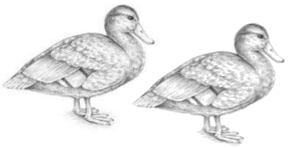
2014	11 services
2015	17 services
2016	27 services
2017	38 services
2018	39 services

The Prodigious Pair—term used for two consecutive years when AWS just went into overdrive in terms of announcing services.

In 2017, AWS vowed to release 38 services to celebrate Matt Wood's 38th birthday. We had more services announced in one year than across the whole Firmament Period (which is 5 years). Then they *had* to top it: thirty-nine services were released in 2018.

iv. Present Period

Jan 2019 onwards



AWS did not keep up the rate of announcements established in the Aggressive Expansion period, something not entirely surprising and no mark of slowing down the rate of innovation. To ground dates, Boris Johnson addressed the United Kingdom to announce a lockdown first in March 2020. Major disruption continues into 2021.

Following the Two Ducks (2019 and 20) we experienced 2021. Seventeen services were announced in the year COVID steadily mired humans on earth in a thick, fatal fog. Seventeen services were also released in 2015.

So there we have it: Foundation, Firmament, Expansion, Present. Each of these four words denotes a five-year period. "Four nines" turns out to be a term that isn't just significant for availability nerds (99.99%) it also captures that fact that years ending in four or nine mark important boundaries for us.

FFEP

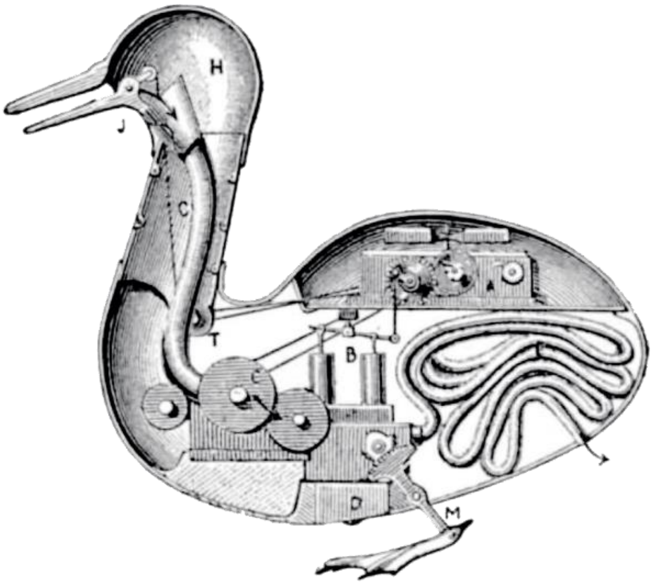
Present Period begins in January 2019 and is still playing out. Conceivably, something could happen during this period that should cause us to re-think all the categories I have set out here. Feel Free, Edit my Periods (FFEP – Foundation, Firmament, Expansion, Present).

The ML Turn

The term "present period" is alliterative but disastrously future proof. One phenomenon which doesn't neatly fit into my categories is the ML Turn. The *ML Turn* describes AWS's purposeful foray into services that utilise machine learning (ML).

This began in 2017, the year SageMaker was announced, which is a general model making service. We would then see a proliferation of services that applied machine learning to specific domains. Amazon Textract (announced 28th Nov 2018) automatically extracts written text from documents; Personalize (28th Nov again) helps you build recommendation engines so that individuals using an application received personalised recommendations; Kendra (3rd Dec 2019) uses ML to enhance enterprise search; Panorama (1st

Dec 2020) applies ML to computer vision, so that cameras fixated on queues in a supermarket could result in a computer knowing how many people are queuing. The ML Turn begins in the Aggressive Expansion period (2014-18) but continues to flourish in the Present Period (Jan 2019 onwards). AWS were methodically, strategically, lining up intelligent tools along a cross-section of the modern world. Such lining up splurged right across my the boundaries of my periods, beginning at the end of the AE period, and into the Present Period.



The Duck Years

> Term used to describe the period following the Prodigious Pair. In other words, 2019 and 20. In each of these years, AWS announced 22 services. In bingo, in Britain, 22 is introduced by saying "two little ducks, 22".

We await the end of 2022. If we are to bend reality to the (unreasonable) strictures of this model, Jan 2024 will be when we should next reflect on the character of the previous five years.

Studying a service

What form should our treatment of each service take?

There will be a section called TPN (demonstrated below), which is designed to get all the pernickety facts in one place. There will be a bibliography, containing a hoard of sources. It will be divided into four sections.

Blogposts are the way to go. Get used to staring at diagrams and staring at code. Wait, and keep waiting, until the diagram has made its point.

Ten Phenomena Named (TPN)

To **P**ut in your **N**otes (TPN)

Bibliography

A bibliography should consist of four sections – Official, Unofficial, Critical, General.

DOMPs

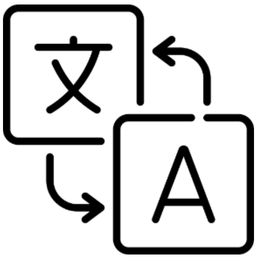
Now we are ready to home in and look at particular years. AWS products are sometimes released in preview. Then, at a later date, they are made generally available. If I have to associate the AWS product with a particular date, I will opt for the earliest date that the product entered public awareness – the date it was announced.

If we consider the spread of announcements over the course of a year, it is usually very skewed. There is not a balanced spread. For example, in 2016, twenty-six products were announced. Yet only eleven dates in the year had product announcements occur.

Certainly, some dates were only covered by one product announcement. The 21st January was covered only with the announcement of AWS Certificate Manager. In contrast, the first day of December was concurrently covered. Ten products were released on 1st December.

The terms for such days is Days of Multiple Products (DOMPS) when AWS dump their products into the world. Here are some notable Domps.

30th Nov 2016	**Athena** IoT Button IoT Greengrass Lightsail Lex Polly **Rekognition**

	Athena (Interesting Individual) Loved Liquor-based personal remedies. *All ink institutes love large pen reserves!* *Animals in India love large packs of rabbits.*
1st Dec 2016 December had arrived – this is the month when people slip and fall and need X-rays.	App Stream 2.0 Batch CodeBuild Glue Personal Health Dashboard Pinpoint Shield Step Functions Systems Manager **X-ray** *Apps, by combining generous piles of paper, shield sharp, severe X-rays.* [bizarre] *Anaesthetists/applications bring certain great powers e.g. perceive special shapes (among) several x-rays!* *Always bring cosy gloves – protective, personal shields, stopping (the need for) several x-rays.*
29th Nov 2017 	**Comprehend** DeepLens Fargate FreeRTOS IoT 1-click IoT Device Defender IoT Device Management Kinesis Video Streams Neptune SageMaker Transcribe **Translate** ***Comprehending*** *Danish ‘fight’ flummoxed individuals*

	interested in Karma. Nothing satisfies the translation!
28th Nov 2018	**App Mesh** Cloud Map Control Tower DeepRacer Elastic Inference Forecast FSx for Lustre FSx for Windows File Server Inferentia Lake Formation License Manager Managed Blockchain Managed Streaming for Apache Kafka Personalize QLDB RDS on VMWare Security Hub Textract

The biggest Domp was 29th Nov 2017 when 12 products were dumped.

APPENDIX

Principles

Elastic Preface – mention the elastic preface (either AWS or Amazon) the first time you mention the product, and then disregard it.

Year of Announcement – when assigning AWS products to particular dates, we will opt to use the date the product was first announced. We will not, for example, use the date the product became generally available (GA), if this differs.

Alphabetical Order – If AWS announce multiple products on a single day, we will order the products alphabetically. We will disregard the elastic preface.

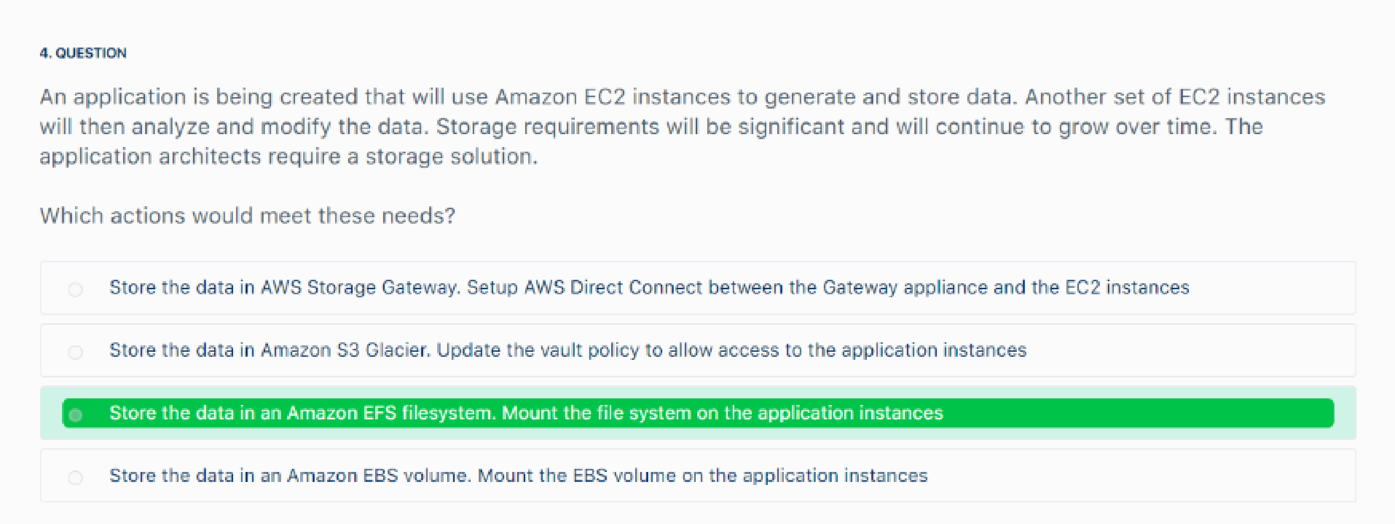
Bibliography

Pinker, Steven (2014). The Source of Bad Writing. Available at: https://stevenpinker.com/files/pinker/files/the_source_of_bad_writing_-_wsj_0.pdf

Ryle, Gilbert (1949). Knowledge how and knowledge that. Aristotelian Society.

Ingold, Christopher K. "Edward David Hughes. 1906-1963." *Biographical Memoirs of Fellows of the Royal Society*, vol. 10, 1964, pp. 147–82. *JSTOR*, http://www.jstor.org/stable/769317. Accessed 18 Aug. 2022.

Examples of Madstuff



4. QUESTION

An application is being created that will use Amazon EC2 instances to generate and store data. Another set of EC2 instances will then analyze and modify the data. Storage requirements will be significant and will continue to grow over time. The application architects require a storage solution.

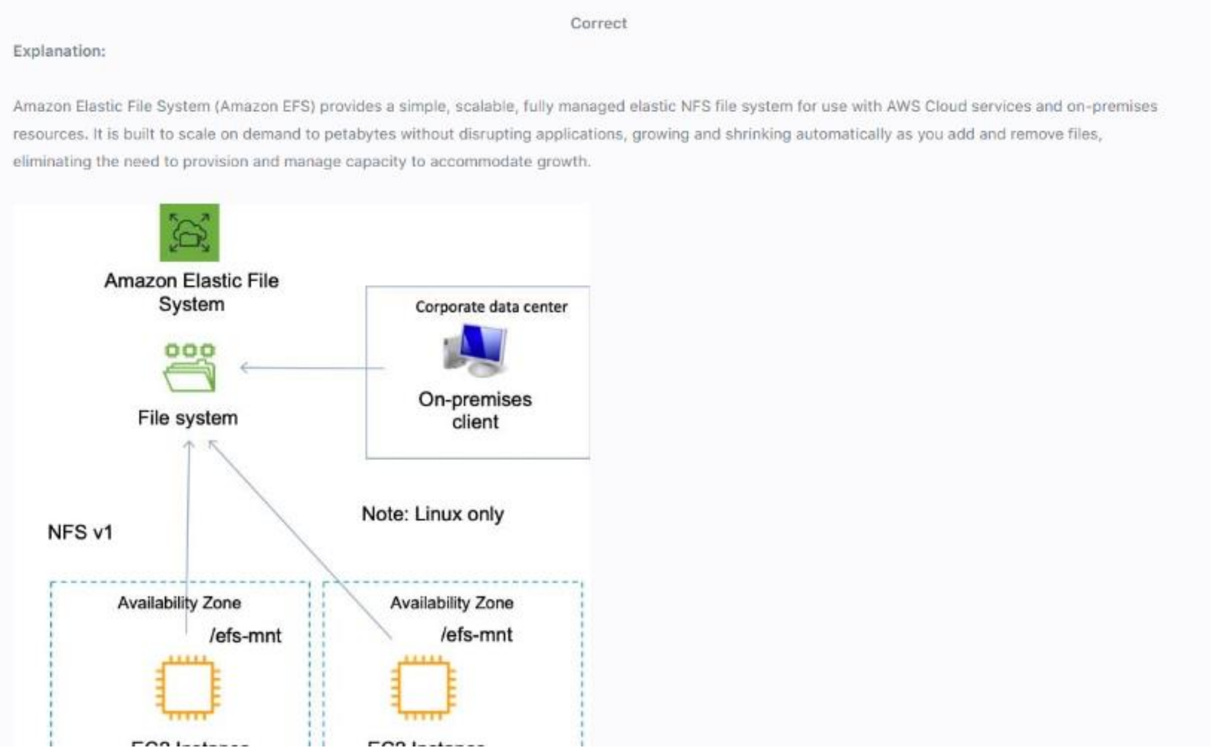
Which actions would meet these needs?

○ Store the data in AWS Storage Gateway. Setup AWS Direct Connect between the Gateway appliance and the EC2 instances

○ Store the data in Amazon S3 Glacier. Update the vault policy to allow access to the application instances

● Store the data in an Amazon EFS filesystem. Mount the file system on the application instances

○ Store the data in an Amazon EBS volume. Mount the EBS volume on the application instances



Correct

Explanation:

Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources. It is built to scale on demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files, eliminating the need to provision and manage capacity to accommodate growth.

https://cloudpundit.com/2022/09/12/cloud-adoption-will-fail-because-of-the-skills-gap/?ck_subscriber_id=1560524742