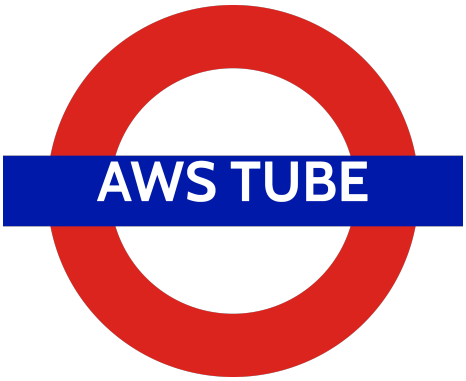

Figure 1: CLICK HERE TO RETURN TO STATION

Recovery Sheet 15

SJ Green

April 25, 2025

1 Active-Active

1. **Term 1** A computer is an electronic device that processes data and performs tasks according to a set of instructions, typically provided by software.
2. **Term 2** It is a virtual server.
3. **Term 3** "Amazon EC2 works in terms of AMIs, or Amazon Machine Images. Each AMI is a pre-configured boot disk - just a packaged-up operating system stored as an Amazon S3 Object."
4. **Term 4** "When you launch an instance, the Instance Type that you specify determines the hardware of the host computer used for your instance. Each instance offers different compute, memory, and storage capabilities, and is grouped in an instance family based on these capabilities."
5. **Term 5** When you launch an EC2 Instance, you can specify a key pair. What on earth is that? It is the combination of a private key and a public key.

6. **Term 6** This is text that you (the user) can specify when you launch an instance. User Data allows you to automate instance configuration when launching an EC2 instance.

7. **Term 7** You can create an AMI from an EC2 Intance that is running. Alternatively, you could buy an AMI from the AWS Marketplace.

8. **Term 8** "The AWS Marketplace is an online store where you can buy software that runs on AWS, including AMIs that you can use to launch your EC2 instance" writes the EC2 User Guide.

9. **Term 9** "An Amazon EC2 Dedicated Host is a physical server that is fully dedicated for your use. You can optionally choose to share the instance capacity with other accounts" writes the EC2 User Guide.

10. **Term 10** "Instance metadata is data about your instance that you can use to configure or manage the running instance" (EC2 User Guide 2025).

11. **Spot Intances** "A Spot Instance is an instance that uses spare EC2 capacity that is available for less than the On-Demand price. Because Spot Instances enable you to request unused EC2 instances at steep discounts, you can lower your Amazon EC2 costs significantly." [AWS 2025].

2 Warm Standby

1. **Computer**. A computer is an electronic device that processes data and performs tasks according to a set of instructions, typically provided by software. It consists of hardware components such as a central processing unit (CPU), memory (RAM), storage (HDD or SDD), and input and outputs devices such as keyboards, mouses, and monitors. A *server* is a type of computer or software that provides services, resources, or data to other computers, known as clients, over a network. Servers can be physical machines or virtualised instances running on cloud infrastructure.

2. An **EC2 Instance**. It is a virtual server. "Amazon EC2 gives you access to a virtual computing environment. Your applications run on a "virtual CPU", the equivalent of a 1.7 Ghz Xeon processor, 1.75 GB of RAM, 160 GB of local disk and 250 Mb/second of network bandwidth" wrote Jeff Barr in 2006. I will always try to write it as EC2 Instance, or Instance, to make it clear that I'm not talking generally about instantiations.

   Notice all the parts of the name (RHEL stands for Red Hat Enterprise Linux; HVM stands for Hardware Virtual Machine which is one of the "virtualization types"; x86_64 is the architecture of the processor; GP2 stands for General Purpose version 2 and its just the type of storage volume that the AMI uses; try to relax; the second time you read this sheet it will be easier). I recently used one AMI which uses the **Amazon Linux 2** Operating System (OS).

3. **Instance Type**. "When you launch an instance, the Instance Type that you specify determines the hardware of the host computer used for your instance. Each instance offers different compute, memory, and storage capabilities, and is grouped in an instance family based on these capabilities." writes the EC2 User Guide. As a rule of thumb, the AMI (pronounced "Aye-Em-Eye", with three syllables, by ordinary people) is all the software and the Instance Type is an allocation of hardware. This is only a rule of thumb: the virtualization type of your instance might intuitively thought to be a hardware matter, but it is in fact a property of the AMI. The two "virtualization types" are paravirtual (PV) and hardware virtual machine (HVM). Windows AMIs are HVM AMIs. Examples of Instance Types are t3.micro and c6g.xlarge. Many people use t3.micro for tutorials and demonstrations because it is cheap.

4. **Key Pair**. When you launch an EC2 Instance, you can specify a key pair. What on earth is that? It is the combination of a private key and a public key. The private key is a piece of information that you keep secret and do not share; if you show it to AWS as you connect to an EC2 instance, then you will be allowed to connect in the way a police officer is let into a crime scene as he holds up the warrant badge that only he possesses. Because there is a private key and public key it is called a key *pair*. The purpose of the public key is to encrypt - to turn plain text into mangled ciphertext. It doesn't matter who has it. If you want to go around encrypting things then knock yourself out. The private key, however, can de-crypt. This is why it must be kept private. The system used is usually RSA - the letters stand for three men, Ronald Rivest, Adi Shamir and Leonard Adleman and these mathematicians worked on the eighth floor of MIT. "The system of asymmetric cyrptography, known as RSA, is said to be a form of *public key cryptography*" writes Simon Singh [The Code Book, 1999].

5. **User Data**. This is text that you (the user) can specify when you launch an instance. User Data allows you to automate instance configuration when launching an EC2 instance. It allows you to run scripts or pass configuration commands to an instance at startup. You probably want a few concrete examples. A script that installs Internet Information Services (IIS), a web server developed by Microsoft, might look like this:

```
<powershell>
Install-WindowsFeature -name Web-Server -IncludeManagementTools
</powershell>
```

That script has XML tags at the beginning and end. Within these tags, the language is PowerShell. A second example of a script would be:

```
#!/bin/bash
```

```
yum update -y
yum install httpd -y
systemctl start httpd
systemctl enable httpd
```

That script is written in a language called BASH. Mr Fox released it in 1989. This script installs Apache and starts a web server. The first part with the hash and exclamation mark is called the "shebang". This script, because it is written in BASH, would work on a Linux OS. YUM is just a package manager and "update" is one of its commands. An AMI has pre-installed software, but User Data is another tool in your box; User Data allows for dynamic configuration at launch.

6. **The Creation of an AMI**. You can create an AMI from an EC2 Intance that is running. Alternatively, you could buy an AMI from the AWS Marketplace. "You can create your own Amazon EBS-backed AMI from an Amazon EC2 instance or from a snapshot of the root device of an Amazon EC2 instance." EC2 User Guide.

   This is a very exciting thing to talk about because not many people understand it yet it is unbelievably helpful. So, listen up. You can learn about AMIs which already exist by typing "aws ec2 describe-images" on the AWS Command Line Interface. That command, describe-images, will bring up an enormous response. Give it a go. So, you probably would want to fine-tune the command by using the "query" parameter and the "filter" parameter. You can create an AMI from one of your own EC2 Instances that you have running. On the Console, you just select the Instance and then click on "Actions" and then "Create Image" (watch that here).

7. **AWS Marketplace**. "The AWS Marketplace is an online store where you can buy software that runs on AWS, including AMIs that you can use to launch your EC2 instance" writes the EC2 User Guide.

   The AWS Marketplace was launched in 2012, the year of the London Olympics and the year that AWS announced Redshift and Glacier.

   If a person wanted to become familiar with AMIs you can buy, then they could put on some jazz and just browse the AWS Marketplace for an hour or so. *I'm just reminding you that this is something you can do.* It is right here: https://aws.amazon.com/marketplace. You need to log in as an IAM User. This activity is also beneficial because you can learn about companies which operate in the cloud industry. I recommend this playlist from Donal Fox (brother of Brian, who birthed BASH).

   Go to the list of categories and select one. For example, there is CRM, Gaming Data, Security, Education & Research. Then play around with

the filters. You could set the **Pricing Model** to Free, as opposed to Usage Based or Upfront Commitment. Now, it's true that if you spend time browsing the AWS Marketplace you will be thrown into specific details about particular industries. But that's why it's such a beneficial activity if you're serious about using AWS in this life and not the next...

One AMI I came across is called "Bitnami LMS powered by Moodle(TM) LMS". LMS stands for Learning Management System and this is used in the education sector. Another product I came across is "Growth Trends and Projections in the Commercial Seaweed Market". The delivery method for that product is "Data exchange". I came across another product entitled "Datadog operator". This is an add-on for the Elastic Kubernetes Service.

It is important to note that not every product you come across on AWS Marketplace will be an Amazon Machine Image (AMI). There is a bar on the left with filters. The "Delivery Methods" section lists the various *types* of product: AMIs, SaaS, Professional Services, CloudFormation templates, container images, Add-ons for EKS, and so on. I have ticked the box for "Professional Services" and come across this product:

> "Cloudbric WMS (WAF Managed Service) for AWS WAF (PAYG)".

The description of the product tells us that:

> Operating and managing AWS Web Application Firewall (AWS WAF) can be quite challenging for those without sufficient security expertise...
>
> the experts of Penta Security will initially run a full analysis of the users logs to configure the security rules optimized to the unique environment of the user to provide security against the latest web hacking threats.

When the product consists in "Professional Services", professionals help you to *do* things. That is, you are not merely paying for a *thing*, such as a package of software, but expertise in configuring and using software. Becoming familiar with AWS Marketplace is valuable because many people who work in the cloud industry work for the company names you see in the products here.

8. **Dedicated Host**. Now, the topic of DH. It's not the Department of Health or the postcode of Durham—a large physical region in the North of England–in this context. I'm reminded of fictional character Detective Inspector George Gently, the lonely DI who worked cases in Newcastle.

DH stands for dedicated host. "An Amazon EC2 Dedicated Host is a physical server that is fully dedicated for your use. You can optionally choose to share the instance capacity with other accounts" writes the EC2 User Guide. That is a nice starting point, but it raises three questions. (1) Why would I need a physical server "fully dedicated" to me? (2) What do we mean by "fully dedicated"? (3) What on *earth* is meant by sharing "instance capacity"?

(1) First of all, you should, in this week, be aware that it is not the norm to have a physical server dedicated to you; one of the defining characteristics of cloud computing is that resources for all of society are pooled and single physical servers can be shared by many customers. However, some industries have strict data residency requirements. Consider PCI DSS (Payment Card Industry Data Security Standard), although note that many AWS services are PCI DSS compliant *anyhow* (see here). Many companies such as Microsoft and Oracle have **licensing models** which make you pay to use a *particular* socket. As a rough example, if you wanted to use two sockets, you would need to buy two licenses.

And there you have the line trotted out unthinkingly by all people giving a scripted talk on AWS and the other Packet Parroters. So does this mean that whenever I use the Microsoft "SQL Server" product on AWS I must use Dedicated Hosts? **Well, no**. There are two options: "You can launch Amazon Elastic Compute Cloud (Amazon EC2) instances with Microsoft SQL Server licenses included from AWS, or you can bring your own SQL Server licenses for use on AWS." SQL Server on AWS User Guide. You would most likely need to use a DH if you are bringing your own license (BYOL). Specifically, if your license lacks something called "Software Assurance", then you would need to think about using a DH. "SQL Server licenses without Software Assurance can be deployed on Amazon Elastic Compute Cloud Dedicated Hosts if the licenses are purchased prior to 10/1/2019 or added as a true-up under an active Enterprise Enrollment that was effective prior to 10/1/2019" write AWS.

(2) To say that a physical server is "fully dedicated" to you is to say that you are the only customer on that specific physical server. There are no other tenants on the physical server. It does not mean that you have control over the hypervisor or that you own the hardware. (3) "Sharing instance capacity" I think this is a clumsy way of putting it. But I think they are referring to the capacity that the *physical server* has, for hosting EC2 Instances. It is like me asking you at the end of Sunday lunch if you have "cheesecake capacity". They are saying that you can share the physical server with other AWS Accounts in your AWS Organization. The vCPUs and memory and networking bandwidth that the physical server has — this gives the physical server the capacity to host EC2 Instances. You can

share this capacity with other AWS Accounts. (This "instance capacity").

So there we have it. DH is not the postcode of Durham, in this context. But the connotations of industry, physicality, and loyalty to location - these we'll take. Indeed, with a DH you can "consistently deploy your instances to the same physical server over time". Also, you can state that your DH can accept untargeted Instance launches. Wouldn't want to waste the DH. So, your physical server says "any, pet" (AP) when asked which Instances it hosts, and takes EC2 Instances launched which did not actually specify the DH **host ID**. Conversely, if AP, or Auto-placement, is disabled, then the DH will only accept launches of EC2 instances that specified its unique Host ID (AWS 2025). If you've *disabled* Auto-placement, you might increase the chances that you waste that physical chunk of server metal–your very own Durham–given that you might forget to aim EC2 Instances at specific host-id.

9. **Instance Metadata Service (IMDS)**.

   "Instance metadata is data about your instance that you can use to configure or manage the running instance" (EC2 User Guide 2025).

   Let's break this down into manageable chunks. *Metadata* is just data about data. "Meta" means beyond or above. If you've taken a photograph of your friend on the beach, then the *data* is the picture elements themselves - the fact that the sun was so high and so bright and so on. The *metadata* would be facts such as the time the photograph was taken and the camera it was taken on. If you look at a PNG file in Windows File Explorer, you can see the metadata displayed at the bottom of the window. Well, EC2 Instances also have metadata. What kinds of facts might these be? The ID of the AMI being used on the EC2 Instance, something called the "AMI launch index", something called the "AMI manifest path", the block device mapping, the public IPv4 address of the EC2 Instance, the instance-id for the EC2 instance - these are all examples of metadata. The AMI launch index is simply a number assigned to the EC2 instance (0 or 1 or 2...). It's very helpful when you launch multiple EC2 Instances from a single AMI. Each EC2 Instance will have a unique number.

   We now know what metadata is and have looked at six examples of Instance metadata. It is called the Instance Metadata *Service* because you can retrieve these facts by using a web service. It's like doing "ping". Like all of the Amazon Web Services, you can utilise the service using the Console or AWS CLI or an SDK. On a command line, such as Windows Command Prompt, you simply send a request to 169.254.169.254. Yes, that's right, it's the same IP address for everybody. But you would need to use the "ping" command and you would need to be "inside" your EC2 Instance first. Still, the idea is that you ping this IP address and the EC2 instance can learn facts about itself. "The concept here is introspection.

There's data in the metadata service, about the Instance where you're running" Mark Ryland (2019) tells us.

10. **Spot Instances**. "A Spot Instance is an instance that uses spare EC2 capacity that is available for less than the On-Demand price. Because Spot Instances enable you to request unused EC2 instances at steep discounts, you can lower your Amazon EC2 costs significantly." [AWS 2025].

    Why use the word "spot"? Well, the idea is that these EC2 Instances are like "open spots" in AWS data centres. There is capacity that is not being fully utilized at the moment but is available for short-term use. There are three important pricing models for EC2 Instances: Spot, On-Demand, Reserved. Now, these three models might annoy you a little bit because they're not actually categories (classes which exhaustively cover everything). They're an example what I call "AWS Hodgepodge". The properties of each model are somewhat arbitrarily combined, but the fact that AWS have made the decision to combine them does simplify things and encourage action. For example, spot instances *are* on-demand, it is just that your demand might not be met. And this is what I mean when I say that the classes are not natural categories.

    The default pricing model is **On-Demand**. The advantage of reserving an EC2 instance is that it is cheap and you have certainty that you will get it. The disadvantage is that you are committed to long periods of time. The advantage of Spot instances is their extreme cheapness. Their disadvantage is that you are not guaranteed to get it. Also, it may be taken away. "Amazon EC2 can interrupt an individual Spot Instance if capacity is no longer available" writes the EC2 User Guide (2025). You can examine spot prices yourself here. As I write this, I can see that a t3.micro costs 0.0028 US dollars

3 Pilot Light

The pilot comes aboard ships in unfamiliar waters to sort out shit.

1. **Computer**.

2. **EC2 Instance**.
3. **Amazon Machine Image (AMI)**
4. **Instance Type**
5. **Key Pair**
6. **User Data**

7. **The Creation of an AMI** in A

8. **AWS Marketplace**

9. **Dedicated Host** In this paragraph I'm going to explain what Dedicated Instances (DIs) are. Now, I can sympathise with people who confuse these with Dedicated Hosts. Recall DH and Durham and loyalty to a location and having a physical server that is dedicated to one customer.

   Well, look at the first sentence from AWS, from the page I cite above:

   > Dedicated Instances are Amazon EC2 instances that run in a VPC on hardware that's dedicated to a single customer.

   You can't make this stuff up. That quotation above is the *perfect* description of Dedicated *Hosts*, but I promise you that it is the AWS decription of Dedicated *Instances*. And without looking, I know this line will be plastered across the Internet by the Packet Parroters. So, like commenting on the emperor's lack of clothing, I see that official line and suggest that the question is, still: **How are Dedicated Instances *disinguished* from Dedicated Hosts?**

   The answer is that you're not able to *track* the hardware that your "Dedicated Instance" runs on, in the way that you can track your "Dedicated Host", repeatedly firing EC2 Instances onto it. AWS refer to this ability to repeatedly use the same hardware as "host affinity". Affinity just means attraction.

   The way you implement these things is telling. You see, if you want a dedicated host, you type `aws ec2 allocate-host` on the AWS CLI. If you want a dedicated instance, you just call `aws ec2 run-instances` in a different manner; you provide a value for the `--placement` parameter.

   "An instance with a tenancy of `dedicated` runs on single-tenant hardware" the CLI User Guide tells us. In other words, the EC2 Instance will be launched onto a server where there are no other tenants. *The lonely DI.* Notice how you are not told *which* server. You are not given a host-id. All you know is that the EC2 Instance is in *a* place where there are no other customers. You don't know what the place is. The Lonley DI - which might be thought to stand for Don't Integrate (with EC2 Instances from other customers).

   To have a Dedicated Host is to know the ID of a host (akin to the name of a city, like "Durham). To have a Dedicated Instance is to know that an Instance is alone in a city (or at least, know that it is not amongst the EC2 Instances of other customers).

What has gone wrong here? The answer is that the two terms from AWS are systematically misleading. The term "Dedicated Instance" does not actually make sense. Consider what the EC2 Instance is dedicated *to*. You might want to answer that: it is dedicated to you (the customer). But aren't *all* your EC2 Instances yours? Yes, they are. *That's not what the term means*. With "DH" it is clear: there is some host, h, which is dedicated to one customer. And this means that all of the host is for the customer. Well, all of an EC2 Instance is yours, anyhow. The conclusion must be that in "Dedicated Instance", AWS have shifted to a **distinct sense of dedication**. Let's see if we can identify what that is.

The only sense in which a Dedicated Instance is *dedicated* must be this: The EC2 Instance dedicates the *host it runs on* to the customer. That looks very reasonable.

But it's not quite correct. There is no reason to believe it dedicates the *whole host* to the customer. Intuitively, that sounds quite extreme. It is like saying that if you reserve one parking space in a car park, we clear out the whole car park for you. So let's take another stab at it:

> The EC2 Instance dedicates the hardware it runs on to the customer.

In other words, wherever the EC2 Instance *is* - no other customer is there. The Dedicated Instance has hardware dedicated *to it*. Although called "Dedicated Instance" it is not an Instance that has been dedicated to anything. The term is systematically misleading. Let's be clear about the direction of dedication: "Dedicated Instances are EC2 instances that run on hardware that's dedicated ***to*** a single AWS Account" (AWS 2025).

Presumably, the developers wanted a term to denote a practice in which: a subset of hardware is dedicated to a customer *but only insofar as the hardware hosts some particular EC2 Instance*. You can see how this is similar to a system in which: a subset of hardware is dedicated to a customer (as in DH). The developers alighted on the term "Dedicated Instance", presumably, by modifying the term "Dedicated Host". That was their base. A DI is not a *host* that is dedicated so they scrapped *that* word from the term. Well, a subset of hardware is dedicated, isn't it? A decision must be made, about *which* subset of hardware AWS are to dedicate to the customer. This is determined by which hardware forms a particular EC2 Instance. (Namely, the one launched with the `--placement` parameter set to DI.) So, they tossed in the word "Instance". That's presumably how the term "Dedicated Instance" came about.

But suppose all that is so... It is analogous to a heart surgeon being guaranteed some parking space in the hospital car park. She does not have any

particular space dedicated to her, such as *the one near the hostpital entrance, by the tall bush.* Rather, she is guaranteed *a space.* (Conceivably enforceable by having the ticket barrier declare "full" , to patients and visitors, even when not quite all the spaces are full. A subset of spaces is always reserved for the surgeons, who know to ignore the "full" sign and have an ID that opens the barrier for them in spite of it.) Now suppose the administrators called it the "dedicated car" scheme. That would be a slightly bizarre name, wouldn't it? It suggests she cannot give people a lift in her car. In fact, though, her car is *not* dedicated to her, or at least not any more than her patient Gene has *his* car dedicated to *him.*

In fact, we cannot use the hospital car park analogy. The analogy fails because cars naturally exclude each other from space. Suppose I am parked in one parking space in the hospital car park - well then, you are not! If you are parked in one space, then I am not parked there. (Unless we stack two cars on top of each other, but let's not be absurd. Even in this situation the cars mutually exclude one another from their spatial location.) *Cars mutually exclude one another.*

Well, virtual servers do *not* mutually exclude one another the way physical things like cars and hosts do. Two virtual servers *can* exist on one piece of hardware, the way two emotions can exist in a single brain. The suggestion that virtualisation involves a single physical thing concurrently covering (or accounting for) *multiple* non-physical things, can be found in many places. On an information page entitled "What is virtualization?" AWS write "Virtual software mimics the functions of physical hardware to run multiple virtual machines simultaneously on a single physical machine". Do a Google Image search for "what is virtualization" and you will find the classic diagram of multiple horizontal layers, like a lasagne. Or look at a video such as "What is Virtualisation?" by Youtube Channel danscourses. There are multiple apps which lie next to one another, on top of one OS - and one physical server.

It is true that to do something virtually is not identical to the concept of *doing many things instantaneously.* Doing something virtually may not even be thought to have anything at all to do with multi-tasking. It is doing something *but not really doing it.* A virtual server is a server *but not server proper.* Porush (2017) traces the etymology of the word "virtual". It used to denote the essence of things. By the 20th century, virtual things were *almost real but not quite.* It's not unreasonable to suppose that this is related, in a non-accidental way, to the fact that the underlying material is attempting *multiple* things at once. Langer characterised dance as virtual gesture, surely, because not only is the performer being themselves but also intelligently gesturing to other emotions.

Why point out that virtual servers do not mutually exclude one another?

Because EC2 Instances are virtual servers. It brings out brightly **the two senses of dedication** at play. Dedicated Instances are not distinguished from Dedicated Hosts merely by being a *subset* of dedicated hardware and not a whole set of hardware (a whole host). That is, we don't merely sacrifice the set. We also sacrifice the spatial overlapping - the concurrency. We install spatial mutual exclusivity, akin to demanding that anger and sadness–completely capable of concurrently running on one brain–sit in two separate brains. That is why we cannot use the hospital car park analogy. We have a physical entity (a host) at play as well as a virtual entity (an EC2 Instance). The distinction between AWS's "Dedicated Host" and "Dedicated Instance" is not merely a contrast between a whole car park and one parking space. A whole server and part of a server. To fully understand what a "Dedicated Instance" achieves, we must appreciate virtualization. It achieves dedication of the *form* of the server's material (and the hypervisor it gives rise to) to the single EC2 Instance.

Well, a final note of clarification. Strictly, there are two possibilities here. If you accept my notion of dedicating form, and not merely material, there are two possibilities. ***(1) CUSTOMER DEDICATION***. It may be that the form is dedicated to the *customer*. This would mean that if the customer crowds one host with many of their own EC2 Instances, then there are "brains hosting two emotions". *There exists a unit of physical infrastructure, x, which hosts two or more EC2 Instances.*

(2) INSTANCE DEDICATION. The second possibility is that the dedication is to **The EC2 Instance**. The dedication is to the integrity of that entity. It would be dedication of form *per se*, and not merely dedication of form so as to achieve dedication to the customer. This would mean that even when a customer happens to crowd a host with lots of their own EC2 Instances, they never co-exist on the same hardware. The EC2 Instances never overlap. Yes, they exist on the same host, but never on the same *part* of the host.

Now, I do not know, having looked at all the AWS documentation, which of these possibilities holds in the case of a DH. Maybe - maybe - safe on the oasis that is an ID-ed and dedicated *host*, virtualization is allowed to flourish, with EC2 Instances sprawling across one another, over single physical parts of hardware, like anger and sadness. AWS do write that with DH you get a "physical server with instance capacity fully dedicated to your use", the latter part *perhaps* suggesting the sprawling. This, perhaps, is why the curious phrase "instance capacity" gets used; AWS are trying to articulate the *freedom of form* that comes with virtualization using hypervisors, distinguished from the *dedication of form* that a DI brings. Alternatively, it may be that the EC2 Instances clank against one another like cars, despite all being on the same customer-dedicated host. One reason to suppose *this* is the case is to suppose that a DH is essen-

tially DI+ (or "under the hood" DH is DI with some extra properties). DH is a superset of DI, such that everything in DI is kept, and we add a few things. DH to DI is like BASH to SH. AWS certainly present DH as a more drastic version of DI in their published materials. ("If you require visibility and control over instance placement and **more comprehensive** BYOL support, consider using a Dedicated Host instead.) So, it would run like this: with DH, not only do EC2 Instances mutually exclude one another like with DI, but we also put each one on the same host. That is something which I wish AWS would clarify. Meanwhile, I'm confident that both Customer Dedication and Instance Dedication hold with a DI. They must.

10. **Term 10** Text

4 Backup

I want this section to be like *Masterpiece Classic* on PBS. Some inclusions are strategic.

(a) **Computer**.

- Text

(b) **EC2 Instance**.

- Phillips, John (2014). Amazon EC2 Instances Deep Dive. *Reinvent 2014*. Available at: `https://www.youtube.com/watch?v=ujGx0tiI1L4&ab_channel=AmazonWebServices`
- Weiss, Becky (2014). Amazon EC2 Networking Deep Dive and Best Practices. *Reinvent 2014*. Available at: `https://www.youtube.com/watch?v=JUw8y_pqD_Y&ab_channel=AmazonWebServices`
- Pai, Raj (2018). EC2 Foundations. Available at: `https://www.youtube.com/watch?v=vXBeO9vQAI8&ab_channel=AmazonWebServices`
- This is just a standard briefing of the information about EC2. I put it here for historical information.

(c) **Amazon Machine Image (AMI)**

- Puccio, Carmen (2018). Managing Modern Infrastructure in Enterprises. Available at: `https://www.youtube.com/watch?v=D91UJ69dcww&t=1418s&ab_channel=AmazonWebServices`

  This is a presentation on Systems Manager, which we haven't covered yet. Do not worry. Today, just listen to Greg Linde talking about "Golden AMIs". A golden AMI is one that is prepared for others to use. Linde is from Verizon (pronounced v(uh)-EYE-zonn). Linde is pronounced as LIN-dee. Verizon is a telecommunications company headquarted in New York City. It is the world's second-largest telecommunications company by

revenue and its mobile network is the "largest wireless carrier in the United States" (Wikipedia 2025).

Linde explains that they wanted to have the Systems Manager agent to be part of the AMI. He states "part of the reason for that is that we did not want the spin-up time of those Instances to be delayed by the downloading of an agent, the installing of an agent, the configuring of an agent" (Linde 20:15). "Our first step along this journey is to ensure that the Systems Manager agent is on every EC2 Instance in the Verizon environment" explains Linde, who adds that every AMI is **descended** from a "Golden AMI". The golden AMIs are managed by a special team. Linde then describes at a high level—in a manner conspicuously devoid of concrete examples, which is a recurring feature of tech talks involving a guest speaker—the process of building the golden AMI. We know they do it in stages: they collect up all the software they need and create a first AMI, and then add some software to the instance while it is running (to create a further AMI, presumably). Linde mentions "staging" the software and I presume this means that they get it up and running on some EC2 Instance. The team then run security pathches, and then scripts, and then "some scans"... Ultimately, the AMI is published by the special team. So, let me emphasise this point: a central team in this huge organisation (Verizon had 105,400 employees in 2023 according to StockAnalysic.com) publish AMIs for everybody else in the organisation to make use of. The AMIs are internally published and "at that point, the business application teams are resposible for re-hydrating their EC2 Instances from the golden AMIs" (Linde 2018: 24:55).

- Maarek, Stephane (2018). Create an AMI on AWS EC2 Tutorial. YouTube Channel: Stephane Maarek. Available at: `https://www.youtube.com/watch?v=kkdr8Av2cQQ`

  There's actually a lot to break down in this video, so let me provide some commentary. The white background screen is the command line program that Maarek is using on his machine. It is probably the "MacOS Terminal" product. It is Unix-like, whatever it is. He also has a script which he keeps referring to. *That* is open in Visual Studio (dark background).

  The video synopsis might go something like this. Maarek begins by using the AWS Management Console to launch an EC2 Instance (00:40 - 01:45). He makes a note of the Public IPv4

address of the EC2 Instance before moving away from the AWS Management Console and to his command line. On the command line, Maarek uses the "ssh" command. The command has a parameter (or "flag") which is -i and this allows Maarek to specify his private key file. By doing this, Maarek connects to the EC2 Instance which he just launched on the Console. By the 01:59 mark, Maarek is connected to his EC2 Instance. Maarek can now paste his script into his command-line, and by doing so he is effectively running the script on the EC2 Instance. The script is written in BASH, a language designed by Mr Brian Fox. Notice the command "sudo yum...". Sudo stands for "superuser do". Maarek next installs Java onto the EC2 Instance (02:50). Next, Maarek installs an application from GitHub. He uses the wget command and specifies a URL (the command is executed at 04:27).

Maarek next demonstrates that: *although there is an application on the EC2 Instance, it will not there if the machine is restarted* Maarek states: "so, our app can run on our machine, but it hasn't been set up to run at the machine restart" (09:17). Maarek seemingly demonstrates this by typing "18.216.81.74:4567details" into his browser bar and getting `ERR_CONNECTION_REFUSED` (09:15). However, I am not sure what this demonstrates - Maarek has just shut down the EC2 Instance, so of course there is no response, and it is not clear to viewers how an AMI would solve this problem. An AMI is a template for launching an EC2 Instance; it will not mean that an application is somehow persisted in spite of an EC2 Instance being terminated. (Yet Maarek displays part of the script, at 09:25, which says "long running configuration (persist after reboot)".)

"Now we want to save the state of this machine and be able to launch machines that look exactly the same", Maarek says at 11:28, motivating the use of AMIs. (*An* AMI *would let us save the state of the machine.*)

The crucial step happens at 11:40 in the video. Maarek has a list of his EC2 Instances and he right-clicks on one. Then he selects "Create Image". At 12:30 Maarek clicks "Create Image". Next, Maarek launches an EC2 Instance *from* the AMI he has just created. He uses the AWS Management Console to do this. "**Let's go ahead** and create an Instance, now, from it" says Maarek at 13:10.
Another time, we could attempt to examine Maarek's script. Unfortunately, Maarek does not provide the script under the video and it is not in the GitHub respository that the application is

in (ec2-masterclass-sampleapp). It is a SysV Init Script (Maarek 09:37). You can click here for a tutorial on how to write a System V init script.

- Maarek, Stephane (2020). AWS EC2 AMI Tutorial. YouTube Channel: Stephane Maarek. Available at: `https://www.youtube.com/watch?v=DOpx2C5F7cs`

(d) **Instance Type**

- Gowdar, Meena (2018). Amazon EC2 T Instances - Burstable, Cost-Effective Performance. *Reinvent 2018*. YouTube Channel: Amazon Web Services. Available at: `https://www.youtube.com/watch?v=OzVpynd_2GM&ab_channel=AmazonWebServices`
- Prasanna, Shashank (2021). How to select Amazon EC2 GPU instances for deep learning. Available at: `https://www.youtube.com/watch?v=4bVrIbgGWEA&t=290s`
- Duffield, Mark (2018). Deep Dive on Amazon EC2 Instances and Performance Optimization Best Practices. Available at: `https://www.youtube.com/watch?v=W0PKclqP3U0`

(e) **Key Pair**

- Maarek, Stephane (2021). SSH to EC2 Instances using Linux or Mac Tutorial. YouTube Channel: Stephane Maarek. Available at: `https://www.youtube.com/watch?v=8UqtMcX_kg0&t=208s`
- Maarek, Stephane (2021). SSH to EC2 Instances using Windows 10 Tutorial. YouTube Channel: Stephane Maarek. Available at: `https://www.youtube.com/watch?v=kzLRxVgos2M`
- Davis, Neal (2021). How to Connect... Using a Bastion Host. YouTube Channel: Digital Cloud Training. Available at: `https://www.youtube.com/watch?v=rn9kAXz6qxA`

(f) **User Data**

- Maarek, Stephane (2019). Amazon EC2 User Data Tutorial. YouTube Channel: Stephane Maarek. Available at: `https://www.youtube.com/watch?v=njY4DGnKgyw&t=202s&ab_channel=StephaneMaarek`
- Darren's Tech Tutorials. AWS User Data. YouTube Channel: Darren's Tech Tutorials. Available at: `https://www.youtube.com/watch?v=5XHY5OnaJe0&ab_channel=Darren%27sTechTutorials`

(g) **The Creation of an AMI**

- Nolen, Thor (2013). Your Linux AMI: Optimization and Performance. *Reinvent 2014*. Available at: `https://www.youtube.com/watch?v=-hwITRP4Pqk&ab_channel=AmazonWebServices`
- Kongdee, Sirirat (2021). EC2 Image Builder - Golden AMIs made easy. YouTube Channel: AWS Events. Available at: `https://www.youtube.com/watch?v=Skafqb7CVDg`

- CloudYeti (2018). Create an Amazon Machine Image with the AWS CLI. YouTube Channel: CLoudYeti. Available at: `https://www.youtube.com/watch?v=kDzbDBOywvI&`

  In this excellent video, Mr Yeti uses the AWS CLI (installed on his MacOS Terminal) to create an AMI. He first launches an EC2 Instance, using the "run-instances" command. He then "extracts the instance-id" from the EC2 Instance he just launched. This is achieved by encapsulating the entire command in rounded brackets, prefacing it with a dollar symbol, and setting is as the value of a variable called instance-id (10:32).
- Green, Sam (2023). JMESPath Tutorial.

(h) **AWS Marketplace**

- Barr, Jeff (2013). The AWS Marketplace. *The AWS Report*. Available at: `https://www.youtube.com/watch?v=cJJvBDoQNmQ&ab_channel=AmazonWebServices`
- Furrier, John (2019). Interview with David McCann. *The Cube*. Available at: `https://www.youtube.com/watch?v=S3W5XVgX3iY&ab_channel=SiliconANGLEtheCUBE`
- McCann, David (2019). Learn more about AWS Marketplace with Dave McCann. Available at: `https://www.youtube.com/watch?v=vXX1ygP6DDY&ab_channel=AmazonWebServices`
- Furrier, John (2022). Interview with Trish Cagliostro. Sept 21st 2022. Available at: `https://www.youtube.com/watch?v=UQF0xuNTcVE&ab_channel=SiliconANGLEtheCUBE`

  Trish Cagliostro is Head of Worldwide Alliances at Wiz. Furrier draws an analogy. It is not rational for small companies to build their own data centres and they should use the cloud instead. Similarly, it is not rational for companies to use the cloud themselves, instead they should use something from AWS Marketplace. Cagliostro states that they have about 100 sellers. "For us it was about aligning the right incentives to drive the right behaviours" says Cagliostro.
-

(i) **Dedicated Host**

- Amazon EC2 Dedicated Instances. Available at: `https://aws.amazon.com/ec2/pricing/dedicated-instances/`

  Green, Samuel (2025). Just Disappointment With Dedicated Hosts Documentation?. *Obscured by Clouds*, vol. 1, issue 1. Available at: https://aws-tube.s3.eu-west-2.amazonaws.com/just$_d$*isappointed.pdf*

- Microsoft SQL Server on Amazon EC2. Available at: `https://docs.aws.amazon.com/sql-server-ec2/latest/userguide/sql-server-on-ec2-licensing.html#sql-server-on-ec2-licensing-considerations-purchasing`
-

(j) **Instance Metadata Service (IMDS)** Text

- Ryland, Mark (2019). Security Best Practices for the Amazon EC2 Instance Metadata Service. Reinvent 2019 [Conference]. Available at: `https://www.youtube.com/watch?v=2B5bhZzayjI&t=18s&ab_channel=AWSEvents`
- Goodrich, Casey (2019). Abusing AWS Metadata Service. YouTube Channel: Casey Goodrich. Available at: `https://www.youtube.com/watch?v=gZsmpPLZQJM&ab_channel=CaseyGoodrich`

  In this video Casey Goodrich uses the metadata service to retrieve the name of an IAM role; *watch how he types into the address bar of the browser at 09:34*. Goodrich uses the metadata service to retrieve the Access Key ID and Secret Access Key for the IAM role; *see if you can spot where on the white background it says AccessKeyId and SecretAccessKey at 09:37*. Goodrich has installed the AWS CLI on his command line; *he explains that you can do this too by typing "apt-get install awscli" at 10:56*. Goodrich alters the configuration of the AWS CLI by altering its config file; *he types "vim /.aws/credentials" at 12:28*. He changes the value of the session token in the config file.

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?