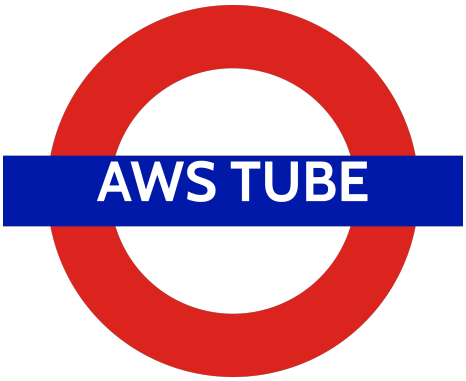

Figure 1: CLICK HERE TO RETURN TO STATION

Recovery Sheet 2

SJ Green

Recovering from the session delivered on the night of Wednesday 22nd January 2025. Last week's sheet on AWS Fundamentals is here.

1 Active/Active

1. An **AWS Account**. It is a container for AWS resources. *Every resource in AWS must exist in an AWS Account.* AWS Accounts are identified with a 12-digit number. Examples of resources inside the AWS Account container are: the root user account, IAM users, S3 buckets, VPCs, CloudTrail trails. "An AWS account represents a formal business relationship you establish with AWS. You create and manage your AWS resources in an AWS account, and your account provides identity management capabilities for access and billing" say AWS Account Management (2025).

2. **The root user account**. The word root means "funamdental", as in the root of a tree or a root canal at the dentist; roots are essential. Every AWS Account has one root user account associated with it. You can log into the root user account with an email address and a password. Those are called **credentials** because they give credence to your claim to be who you say your are. The root user account is permitted to do everything. Therefore,

you should avoid using it. “Instead of accessing the root user, create an administrative user for everyday tasks” states the IAM User Guide.

3. **AWS IAM**was released in 2010. It stands for “identity and access management”. You can create identities such as IAM Users, IAM Roles, and IAM User Groups. The concept of an IAM User is not the same as the concept of an AWS Account (see first definition). IAM Users allow persons to use the AWS Management Console, and the AWS CLI etc. The IAM User has a username and a password (not an email address, like the *root user account*). The IAM User can log into the AWS Managment Console using these credentials. Then, they can manipulate AWS resources, by clicking on buttons. The buttons execute API calls on behalf of the user.

4. **Access Key**. “Access keys are long-term credentials for an IAM User of the AWS Account root user” the IAM User Guide tells us. An access key is a **long-term credential**. Another example of a long-term credential is a password. AWS recommend you avoid using access keys if you can. “Access keys consist of two parts: an access key ID (for example AKIAIOSFODNN7EXAMPLE) and a secret access key (for example, wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY) [IAM User Guide]. A person might paste these two pieces of information into the a command-line interface, just in the same way a person might type a username and password into the AWS Management Console.

5. **IAM Roles**. ”An IAM role is an identity within your AWS account that has specific permissions. It’s similar to an IAM user, but isn’t associated with a specific person” states the IAM User Guide. Many people depict IAM roles as construction hard hats. Different people can wear a hat and it is intended for a particular activity. IAM Roles are similar. Using IAM roles is a best practice. For example: “As a best practice, use temporary security credentials (such as IAM roles) instead of creating long-term credentials like access keys. Before creating access keys, review the alternatives to long-term access keys”, IAM User Guide.

6. **Permissions**. It is not enough to furnish the AWS Cloud with a set of identities. We would know who everybody is, but there would be hierarchy and chaos would ensure. So, we must lay **permissions** on top of our identities. IAM identities, when created, are not permitted to do anything. I refer to this default denial as the *Blanket Deny*. Somebody must endow an IAM identity with the permission to act, to turn them into a **principal** - an actor with responsibility. We do this by attaching IAM policies to IAM identities. IAM policies are nothing but sets of rules which define the permissions of the IAM identity.

7. **IAM Policies**. IAM policies define the hierarchy amongst principals in the AWS Cloud, specifying who can play on what in the virtual park. They are written in a language called JSON (JavaScript Object Notation). There are *keys* and *values*, separated by a colon. IAM policies are made

up of a number of elements. Using these elements, we can bring about order, defining the rules for who can play on what in the e-park. The elements are called "Effect", "Principal", "Action", "Resource", "Condition" (EPARC). Those are the names of keys and each will have a value. For example, " "Effect": "Allow" ". The key is "Effect" and its value is "Allow". A policy that had this in it would be *allowing* some action, or many actions. For example, it might allow DescribeInstances or RunInstances.

8. **Amazon Resource Name (ARN)**. We need a systematic way to refer to AWS resources, similar to the way we use a 12-digit number to identify AWS Accounts. Well, we have one. It is the Amazon Resource Name. Learn its structure and you'll start spotting them. There are usually 7 segements, separated by a colon. The first segment is just "arn" - quite boring, but sensible. The second segment - as in "wait a second am I a spy" - is usually "aws". It's technically called the "partition". Don't worry about it. Third—get to the triangular point of AWS—it's their services! So we might have arn:aws:s3, or arn:aws:iam. What goes in the fourth segment? Well, **four**, as I have been trying to impress upon you, should make you think of four-legged reliable wooden tables and reliable gentlemen who wear suits like Vint Cerf and the fourth Pillar of the AWS Well-Architected Framework, and also vast areas of land, such that if you spread your AWS resources across the areas, you get a reliable architecture. That's right, the fourth segment of the ARN is for the good old reliable **AWS Region**. Now, moving swiftly on, everyone nowadays wants to say "high five for accountability"; the fifth segment houses the identifier of the AWS Account. The rocks and bits and sticks of AWS are named in six. *The sixth segment is where the AWS resource is named.* Take a look at this example Amazon Amazon Resource Name (ARN):

```
arn:aws:iam::123456789012:group/*
```

9. **Defense in Depth**. NIST define it as an "information security strategy integrating people, technology, and operations capabilities to establish variable barriers across multiple layers and missions of the organization". The idea is to use several independent methods to protect a system. If one fails, others step in. There are lots of ways to achieve this with AWS. Up to this point in this sheet, we have only referred to IAM policies attached to identities. But we can also place IAM policies at a resource. Such policies are called **resource-based policies**. It would be great if, even though we forgot to add an appropriate identity-based policy, protection was provided by the resource-based policy. How do I know I'm looking at a resource-based IAM policy? Well, a rule of thumb is that it won't actually have the **"Resource"** element. Being based at the resource, that goes unsaid. Similarly, an identity-based IAM policy is attached to an IAM identity so it may not have an "Identity" element. These are just rules of

thumb.

10. **AWS Organizations**. In 2016, AWS announced "AWS Organizations", a product for allowing companies to manage multiple AWS Accounts. For security reasons, it is a good practice to separate AWS resources into lots of separate containers. It may also help to organise things. This is what is achieved by using lots of AWS Accounts. The AWS Organizations product offers lots of tools to make it easier to manage multiple accounts, such as Organizational Units (OUs) and a **Management Account**. There are also policies to govern permissions policies. They are known as Permissions Boundaries (which "define the maximum permissions that the identity-based policies can grant to an entity) and Service Control Policies (which I'll be referring to as SCPs immediately; they "define the maximum permissions for IAM users and IAM roles within accounts in your organiszation or organizational unit") [IAM User Guide].

2 Warm Standby

Recovery is not as fast as with Active/Active.

1. **AWS Account**.

2. **The root user account**.
3. **AWS IAM**
4. **Access Key**
5. **IAM roles**
6. **Permissions**
7. **IAM Policies** in A
8. **Amazon Resource Name**
9. **Defense in Depth**
10. **AWS Organizations** Text

3 Pilot Light

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

1. **AWS Account**. It is a container for AWS resources. *Every resource in AWS must exist in an AWS Account.* AWS Accounts are identified with a 12-digit number.

2. **The root user account**.
3. **AWS IAM**
4. **Access Key**
5. **IAM roles**
6. **Permissions**
7. **IAM Policies** in A
8. **Amazon Resource Name**
9. **Defense in Depth**
10. **AWS Organizations** Text

4 Backup

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

(a) **AWS Account**. It is a container for AWS resources. *Every resource in AWS must exist in an AWS Account.* AWS Accounts are identified with a 12-digit number.

(b) **The root user account**. The root user account. The word root means "funamdental", as in the root of a tree or a root canal at the dentist; roots are essential. Every AWS Account has one root user account associated with it.

(c) **AWS IAM AWS IAM**was released in 2010. It stands for "identity and access management". You can create identities such as IAM Users, IAM Roles, and IAM User Groups.

(d) **Access Key Access Key**. "Access keys are long-term credentials for an IAM User of the AWS Account root user" the IAM User Guide tells us. An access key is a **long-term credential**. Another example of a long-term credential is a password. AWS recommend you avoid using access keys if you can.

(e) **IAM roles IAM Roles**. "An IAM role is an identity within your AWS account that has specific permissions. It's similar to an IAM user, but isn't associated with a specific person" states the IAM User Guide. Many people depict IAM roles as construction hard hats.

(f) **Permissions Permissions**. It is not enough to furnish the AWS Cloud with a set of identities. We would know who everybody is, but there would be hierarchy and chaos would ensure. So, we must lay **permissions** on top of our identities.

(g) **IAM Policies**. IAM policies define the hierarchy amongst principals in the AWS Cloud, specifying who can play on what in the virtual park. They are written in a language called JSON (JavaScript Object Notation).

(h) **Amazon Resource Name (ARN)**. We need a systematic way to refer to AWS resources, similar to the way we use a 12-digit number to identify AWS Accounts. Well, we have one.

(i) **Defense in Depth**. NIST define it as an "information security strategy integrating people, technology, and operations capabilities to establish variable barriers across multiple layers and missions of the organization".

(j) **AWS Organizations**. In 2016, AWS announced "AWS Organizations", a product for allowing companies to manage multiple AWS Accounts.

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?