

## Project 1 - Identity and Access Management

-You are a Cloud Engineer Consultant, working with StartupCo, a fast-growing tech startup that recently launched their first product - a fitness tracking application.

-They've been using AWS for three months, initially setting up their infrastructure quickly to meet launch deadlines.

-Now that their product is live, they need to address their cloud security fundamentals. The company has 10 employees who all currently share the AWS root account credentials to access and manage their cloud resources.

-This practice started when they were moving quickly to launch, but now their CTO recognizes the security risks this poses..

### **Current Setup:**

- Everyone uses the root account
- No separate permissions for different teams
- No MFA or password policies
- AWS credentials shared via team chat

### **Current Infrastructure:**

- EC2 instances running their application
- S3 buckets storing user data and application assets
- RDS database for user information
- CloudWatch for monitoring
- Several development and production environments

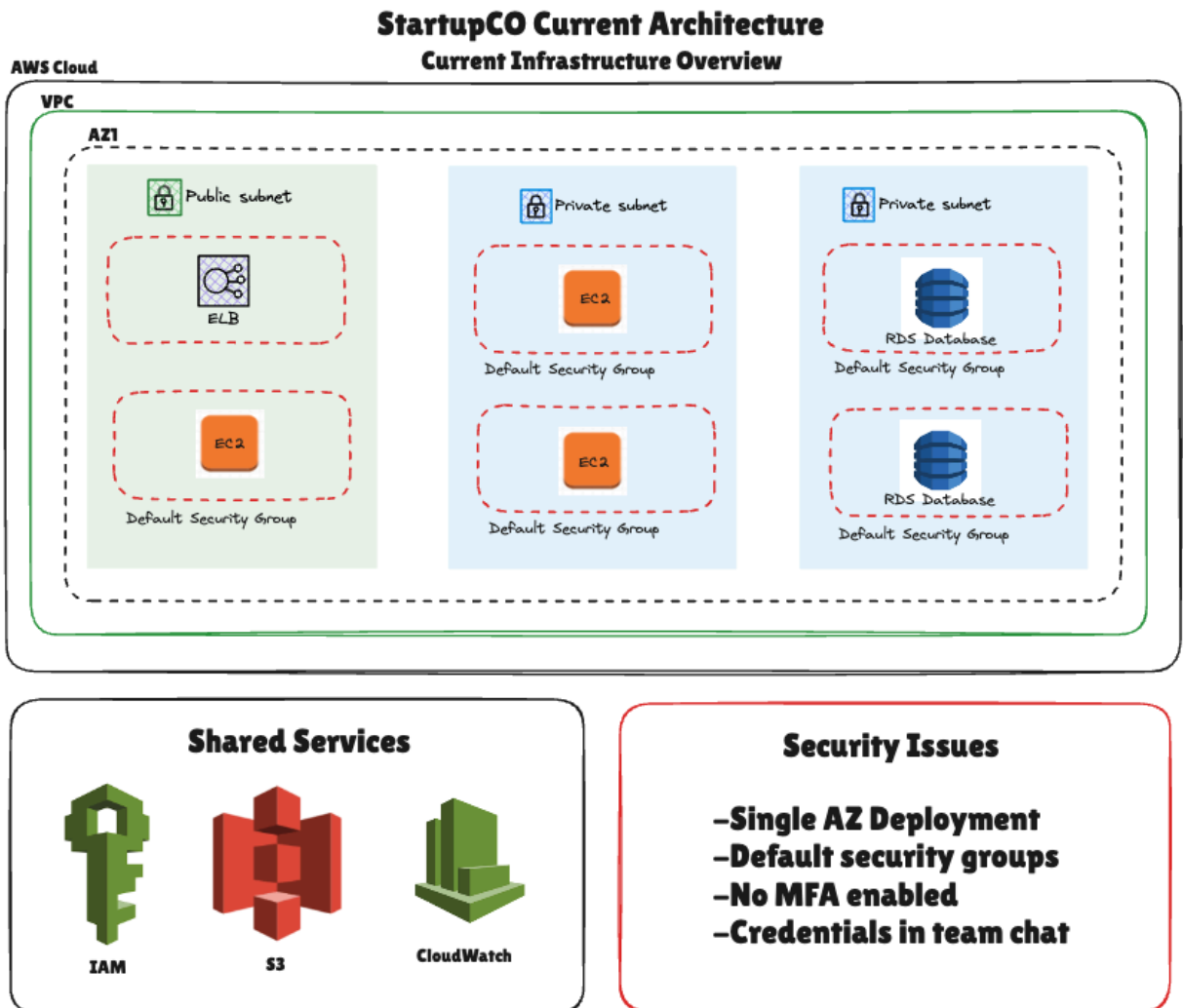
### **Team Structure & Access Needs:**

- 4 Developers (need EC2 and S3 access)
- 2 Operations (need full infrastructure access)
- 1 Finance Manager (needs cost management access)
- 3 Data Analysts (need read-only access to data resources)

## ----- Your Task -----

### 1 - Create their Architecture

- Create an architecture diagram showcasing their current infrastructure



-My draft of their current infrastructure shows us that they have a VPC in AWS that houses 1 public subnet and 2 private subnets. The public subnet contains an ELB and EC2 instance. The private subnets are separated with 1 having microservices and the other RDS databases. There are several default security groups placed in each subnet. The startup is utilizing AWS IAM, S3, and CloudWatch for their shared services. However, located in the red box are the issues they are currently facing.

## **2 - Secure the Root Account**

- Enable MFA
- Stop using it for daily operations
- Store credentials securely

-By securing the Root Account. I have to look into what accounts they have created in AWS so far. If the startup only has the main account aka the management account, then I need to create a Root User Account under that management account.

-To do this: login to the Root Account and go to IAM. You will see that the MFA is disabled. Just like I did in the beginning of bootcamp, you want to follow those steps to create MFA for the Root User account.

-Keep in mind that the Root User is not intended for infrastructure work. For that I would create an admin user and give it the proper permissions such as view billing access, administrator access, etc. This Admin account is actually the management account! However, this account is not what you should intend to build infrastructure with either!

-Once I secured a Root User and Admin User in the Root Account, I would logout of the Root Account and log in to the Admin User.

## **3 - Create IAM Users and Groups**

- Developer group & users
- Operations group & users
- Finance group & user
- Analyst group & users

-Login to the management account and check all accounts with AWS Organizations.

-If you look at it, it only shows 1 account created, that is displayed as the management account. This is because although you created an Admin Account, it is still under the Root User Account.

-The management account should only be used for adding and managing different accounts inside your AWS Organization and/or looking at the billings.

-To start working on infrastructures, I would create another account for it.

-Once I get my infrastructure user account set up, only then can I start creating IAM Users and Groups for the 4 developers for EC2 and S3 access, 2 operations with full infrastructure access, 1 finance manager for cost management access, and 3 data analysts for read-only access to data resources.

-In order to create these users with respective access, it requires their email addresses and names.

#### 4 - Set Up Security Requirements

- Enable MFA for all users
- Create a strong password policy
- Ensure users can only access what they need

-To create a user go to IAM>Users>create user.

-Create a name for the user and IAM user then click next

##### User details

User name

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ \_ - (hyphen)

- ☒ **Provide user access to the AWS Management Console - optional**  
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

##### **Are you providing console access to a person?**

###### User type

- ☐ **Specify a user in Identity Center - Recommended**  
We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.
- ☒ **I want to create an IAM user**  
We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.

##### Console password

- ☒ **Autogenerated password**  
You can view the password after you create the user.
- ☐ **Custom password**  
Enter a custom password for the user.
- 
- Must be at least 8 characters long
  - Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # \$ % ^ & \* ( ) \_ + - (hyphen) = [ ] { } | ' "
- ☐ Show password
- ☒ **Users must create a new password at next sign-in - Recommended**  
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

**Information** If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel

Next

-I can add permissions later so click next again>create user.

-Now I want to create a group so go to User groups>create group.

-I want this group for the 4 developers so name it developer\_group and then attach that new\_user to this group.

#### 5 - Implement These Permissions

##### Developers:

- EC2 management
- S3 access for application files
- CloudWatch logs viewing

##### Operations:

- Full EC2, CloudWatch access
- Systems Manager access
- RDS management

##### Finance:

- Cost Explorer
- AWS Budgets
- Read-only resource access

## Analysts:

- Read-only S3 access
- Read-only database access

-I can now attach the permissions to this developer group so search for EC2, S3, and cloudwatch access. Enter EC2 in the search bar and many will pop up but you want developers to have full access for EC2 and CloudWatch. However for S3 make sure which specific access you want to give them by asking your developers or their managers.

-When I'm done, click the create user group. You can click on the developer group and click the permissions tab to see it all there.

-Follow this process for creating new users and which group to add them to while attaching the correct permissions based on lowest access needs for their respective work.

The screenshot shows the AWS IAM console interface. On the left is a navigation sidebar with sections: 'Identity and Access Management (IAM)' containing a search bar and links to Dashboard, Access management (expanded), Access reports, and Credential report; 'Access management' containing links to User groups, Users, Roles, Policies, Identity providers, Account settings, and Root access management; and 'Access reports' containing links to Access Analyzer, External access, Unused access, Analyzer settings, and Credential report. The main content area is titled 'developer\_group' and includes a 'Delete' button. Below this is a 'Summary' section with fields for 'User group name' (developer\_group), 'Creation time' (March 31, 2025, 17:42 (UTC-07:00)), and 'ARN' (arn:aws:iam::277707114344:group/developer\_group), with an 'Edit' button. The 'Permissions' tab is selected, showing 'Permissions policies (3)' and a note 'You can attach up to 10 managed policies.' It includes a search bar, a 'Filter by Type' dropdown set to 'All types', and a table of attached policies. The table has columns for checkboxes, Policy name, Type, and Attached entities. The attached policies are: AmazonEC2FullAcc... (AWS managed, 1 entity), AmazonS3ReadOn... (AWS managed, 2 entities), and CloudWatchEvents... (AWS managed, 1 entity). Action buttons at the top of the table include a refresh icon, 'Simulate', 'Remove', and 'Add permissions'.

	Policy name	Type	Attached entities
<input type="checkbox"/>	AmazonEC2FullAcc...	AWS managed	1
<input type="checkbox"/>	AmazonS3ReadOn...	AWS managed	2
<input type="checkbox"/>	CloudWatchEvents...	AWS managed	1

## ----- What to Submit -----

### Documentation and Screenshots

1. Clear Documentation explaining the problem you faced, what you did to overcome it, with the Architecture you created
2. In the documentation include screenshots showing:
  1. Your IAM groups and users structure
  2. MFA and password policies
  3. Example permissions for each group
  4. How you structure the permissions
  5. Why you made certain security choices

## -----Advanced Bonus Challenge -----

Once you've implemented everything via Console, try to recreate your solution using:

- Terraform
- AWS CloudFormation
- AWS CDK

This will help you understand both approaches.

