User, Access & Identity Management: Securing the Cloud Era

A comprehensive guide to modern identity and access management frameworks, technologies, and best practices for securing organizations in the cloud era.

Facilitators: Abdallah Ibrahim Nyero, Hajarah Ali Namuwaya, Dr. Ali Mwase, and Charles Kikwanga

Chapter 1: Foundations of Identity and Access Management (IAM)

Understanding the critical building blocks that form the foundation of modern identity and access management systems.

Identity and Access Management (IAM)

1

IAM Framework

A comprehensive system to manage digital identities and control access to resources across your organization

2

Access Control

Ensures only authorized users can access the right resources at the right time and for the right reasons

3

Security Foundation

Serves as the cornerstone of modern security architecture, especially in cloud environments

Identity and Access Management has evolved from a simple directory service to become the critical security control layer for organizations of all sizes.

Key IAM Components

Authentication

The process of verifying a user's identity through various factors:

- Passwords (something you know)
- Security tokens (something you have)
- Biometrics (something you are)

Authorization

Determining what resources an authenticated user can access:

- Permission sets
- Access policies
- Conditional rules

Identity

Digital representation of entities that need access:

- Users (employees, partners, customers)
- Services and applications
- Devices and IoT endpoints

Why IAM Matters Today

Dissolving Perimeters

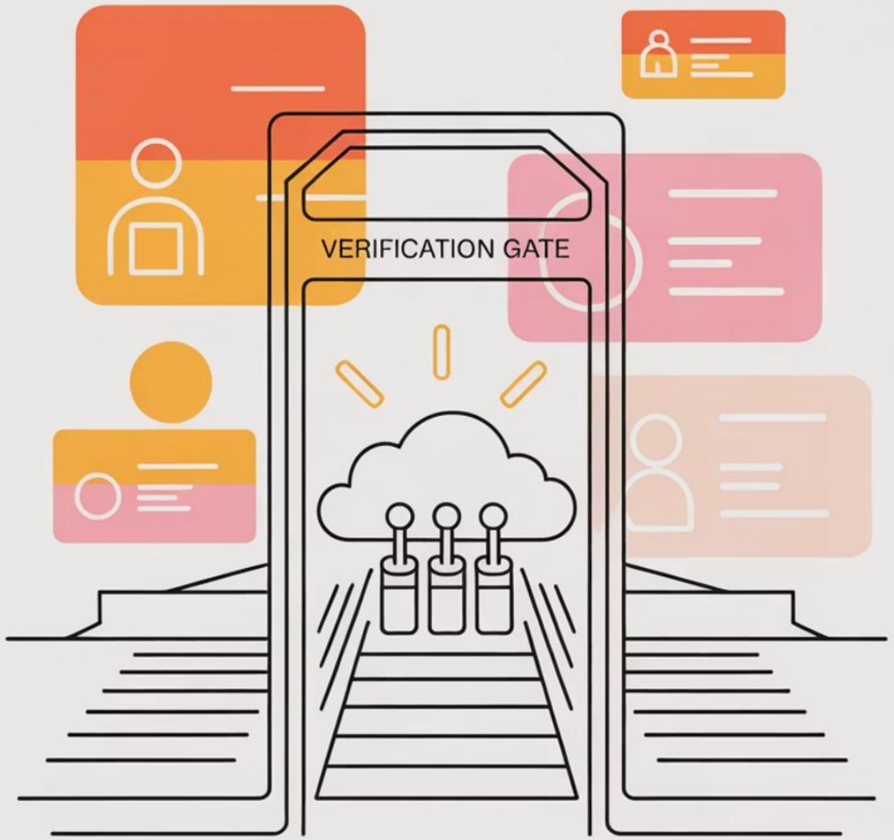
Cloud adoption has eliminated traditional network boundaries, making identity the new security perimeter

Data Protection

Prevents unauthorized access to sensitive information across distributed environments

Compliance

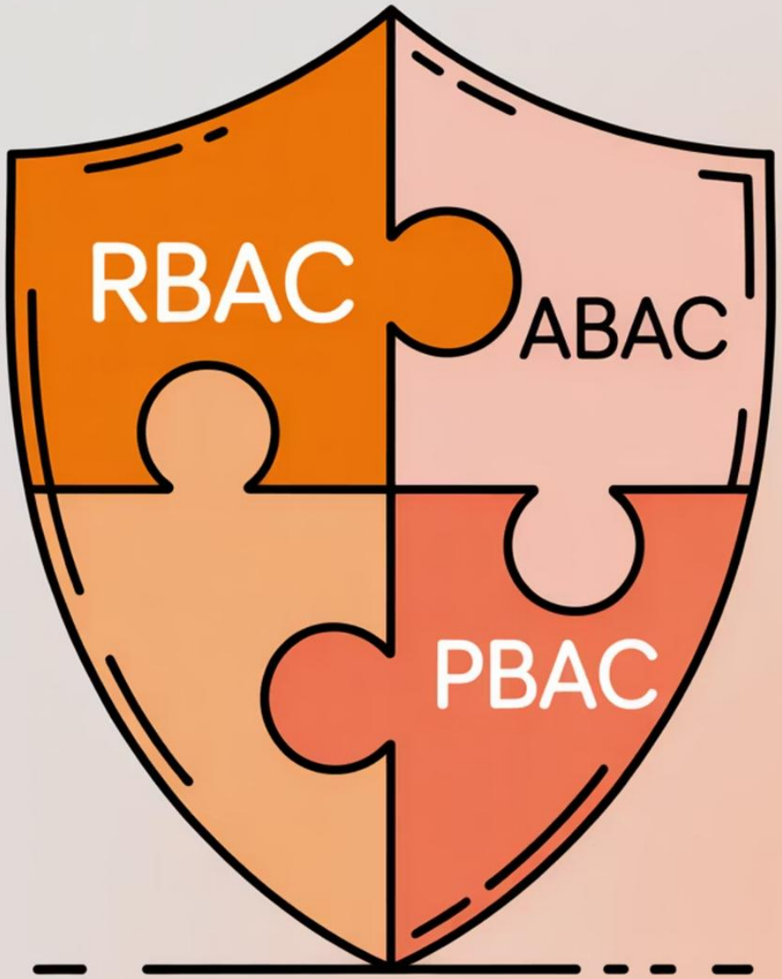
Supports regulatory requirements like GDPR, HIPAA, SOX, and industry-specific frameworks



Cloud IAM Features:

Azure AD, AWS IAM, and Google IAM each provide:

- Role assignment and delegation
- MFA and password policy enforcement
- Access logging and monitoring (CloudTrail, Azure Monitor, Cloud Audit Logs)



Chapter 2: Core Access Control Models: RBAC, ABAC, PBAC

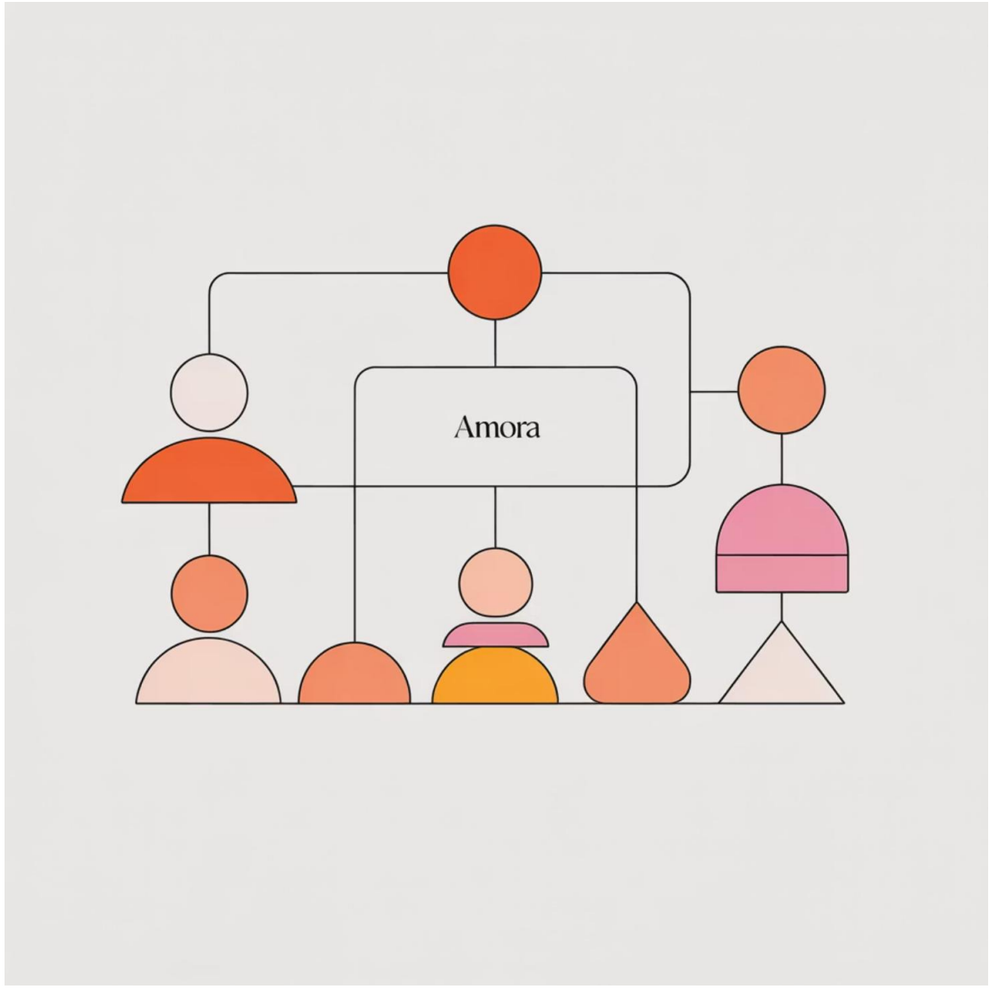
Understanding the fundamental access control methodologies that form the backbone of modern IAM systems.

Role-Based Access Control (RBAC)

RBAC simplifies access management by assigning permissions to roles rather than individual users.

Key Advantages:

- Simplifies permission management at scale
- Enforces least privilege through well-defined roles
- Reduces administrative overhead
- Supports compliance through role standardization
- Enables easy onboarding/offboarding



RBAC Practical Example

Security Manager Role

Permissions:

- View security audit logs across all systems
- Configure security alert thresholds
- Access security incident reports
- Manage security monitoring tools

Configuration: `aws iam create-role --role-name SecurityManager --assume-role-policy-document file://trust-policy.json`

Temporary Contractor Role

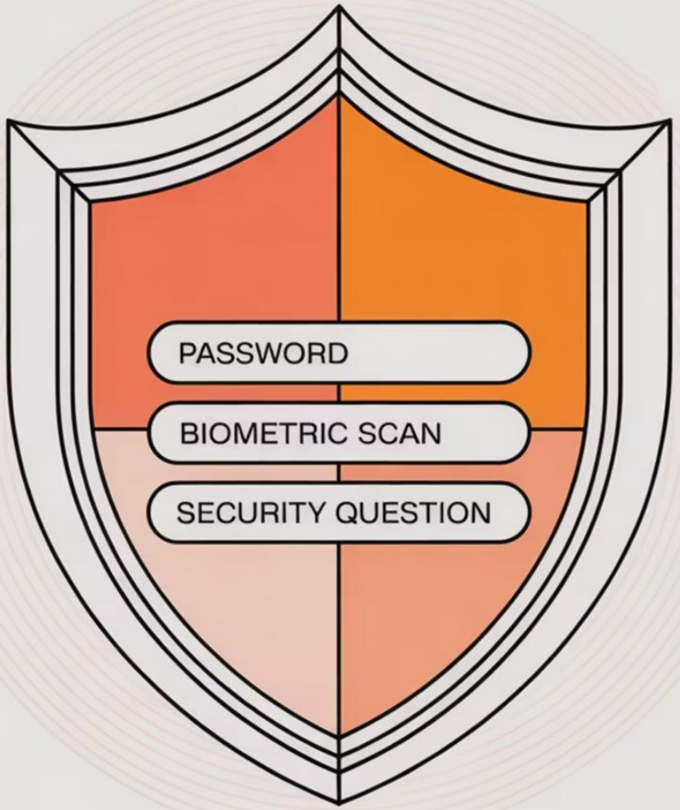
Permissions:

- Limited to specific project resources
- Time-bound access (expires after 90 days)
- No access to production data
- Restricted network access points

Configuration: aws iam create-role --role-name Contractor --max-session-duration 28800

Chapter 3: Authentication Enhancements: MFA & SSO

Strengthening authentication with modern approaches that balance security and user experience.



Multi-Factor Authentication (MFA)

MFA requires two or more verification factors before granting access:

Something You Know

- Password
- PIN
- Security questions

Something You Have

- Mobile device
- Hardware token
- Smart card

Something You Are

- Fingerprint
- Facial recognition
- Voice pattern

99.9%

Reduction in account compromise risk with MFA enabled (Microsoft)

MFA in Practice: AWS Example

AWS MFA Recommendations:

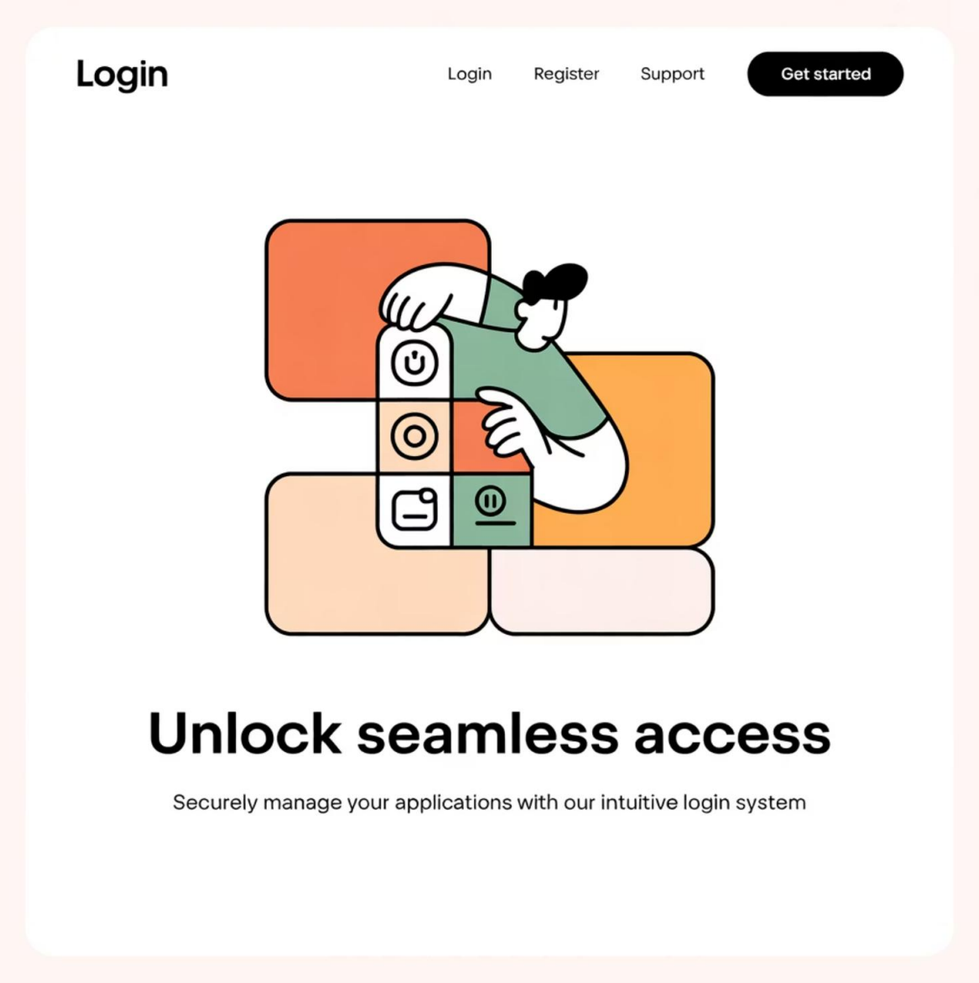
- Mandatory MFA for root accounts
- MFA for all privileged operations
- MFA for all IAM users with console access

Supported MFA Types:

- Virtual MFA (authenticator apps)
- Hardware tokens (YubiKey, etc.)
- SMS text messages (less secure)

```
# AWS CLI example to enable MFA requirementaws iam put-user-policy \ --
user-name admin-user \ --policy-name RequireMFA \ --policy-document '{
"Version": "2012-10-17", "Statement": [ { "Effect": "Deny",
"NotAction": [ "iam:CreateVirtualMFADevice", "iam:EnableMFADevice" ],
"Resource": "*", "Condition": { "BoolIfExists": {
"aws:MultiFactorAuthPresent": "false" } } } ] }'
```

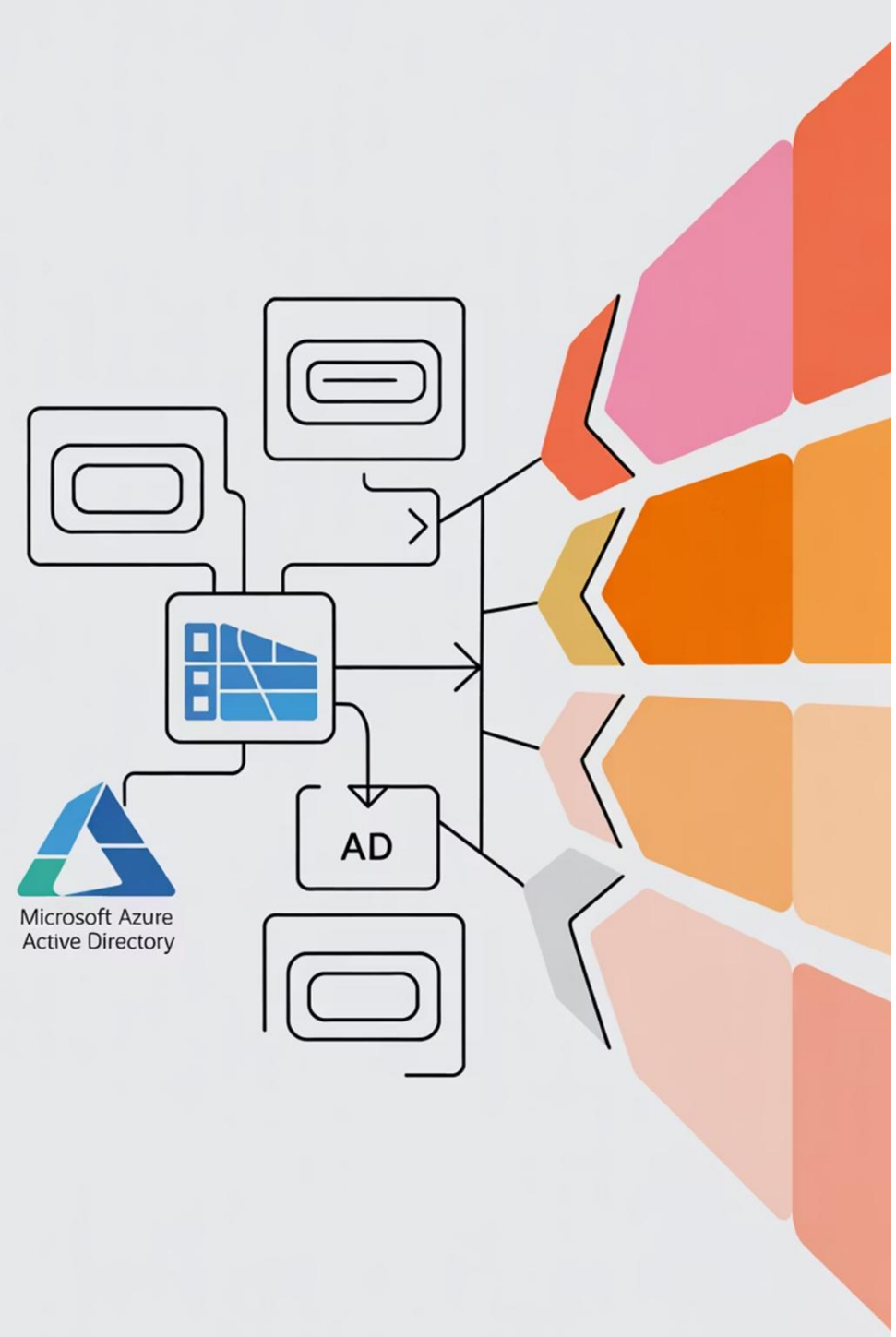
Single Sign-On (SSO)



Key Benefits of SSO:

- One set of credentials for multiple applications
- Reduces password fatigue and reset requests
- Improves security through centralized authentication
- Streamlines user onboarding and offboarding
- Enhances visibility into access patterns

70% reduction in password-related help desk tickets after SSO implementation (Forrester Research)



Practical SSO Configuration Example

Integrating Azure AD with Salesforce using SAML

Azure AD Configuration:

1. Add Salesforce from the gallery
2. Configure SAML endpoints and certificates
3. Map user attributes (email, name, role)
4. Assign users to the Salesforce application

Salesforce Configuration:

1. Set up Single Sign-On Settings
2. Upload IdP certificate from Azure
3. Configure SAML endpoints
4. Set up user provisioning (optional)

Result: Users authenticate once to Azure AD and gain seamless access to Salesforce without additional login prompts.

Chapter 4: IAM in Cloud Platforms: AWS, Azure, Google Cloud

Exploring the native identity and access management capabilities of the major cloud service providers.

AWS IAM Overview

Core Components:

Users: Individual identities for people or services

Groups: Collections of users with shared permissions

Roles: Identities assumed by users, services, or external entities

Policies: JSON documents that define permissions

Key Features:

- Centralized control of AWS account
- Shared access to AWS resources
- Granular permissions
- Identity federation with external providers

AWS Best Practices

Use AWS IAM Identity Center

Centralize access management across multiple AWS accounts and business applications

```
# Enable IAM Identity Center via AWS CLIaws sso-admin create-instance
```

Enforce Least Privilege

Grant only the permissions required to perform a task

```
# Example of scoped S3 access policy{  "Version": "2012-10-17", "Statement": [{    "Effect": "Allow",    "Action": ["s3:GetObject", "s3:ListBucket"],    "Resource": [      "arn:aws:s3:::my-bucket", "arn:aws:s3:::my-bucket/*"    ] }]}
```

Use IAM Access Analyzer

Identify resources shared with external entities and validate policies

```
# Create an Access Analyzeraws accessanalyzer create-analyzer \ --analyzer-name AccountAnalyzer \ --type ACCOUNT
```

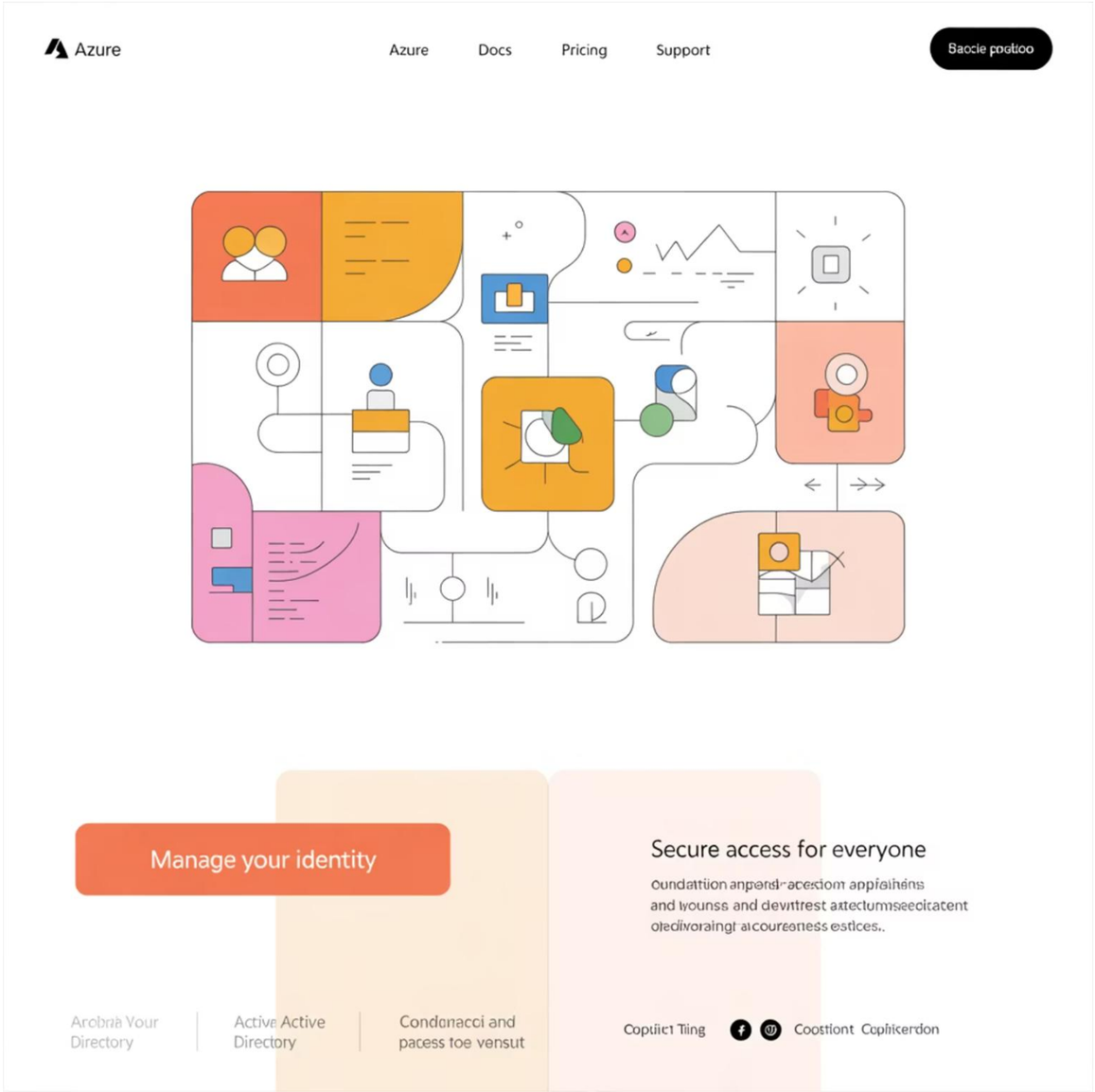
Azure Active Directory (Azure AD)

Core Capabilities:

- Identity management for Microsoft cloud
- Hybrid identity with on-premises AD
- B2B and B2C identity solutions
- Conditional Access policies
- Privileged Identity Management (PIM)

Key Security Features:

- Risk-based authentication
- Identity Protection
- Just-in-time access
- Access Reviews



Google Cloud IAM

Key Concepts:

Members: Users, service accounts, groups, domains

Roles: Collections of permissions

Permissions: Determine allowed operations on resources

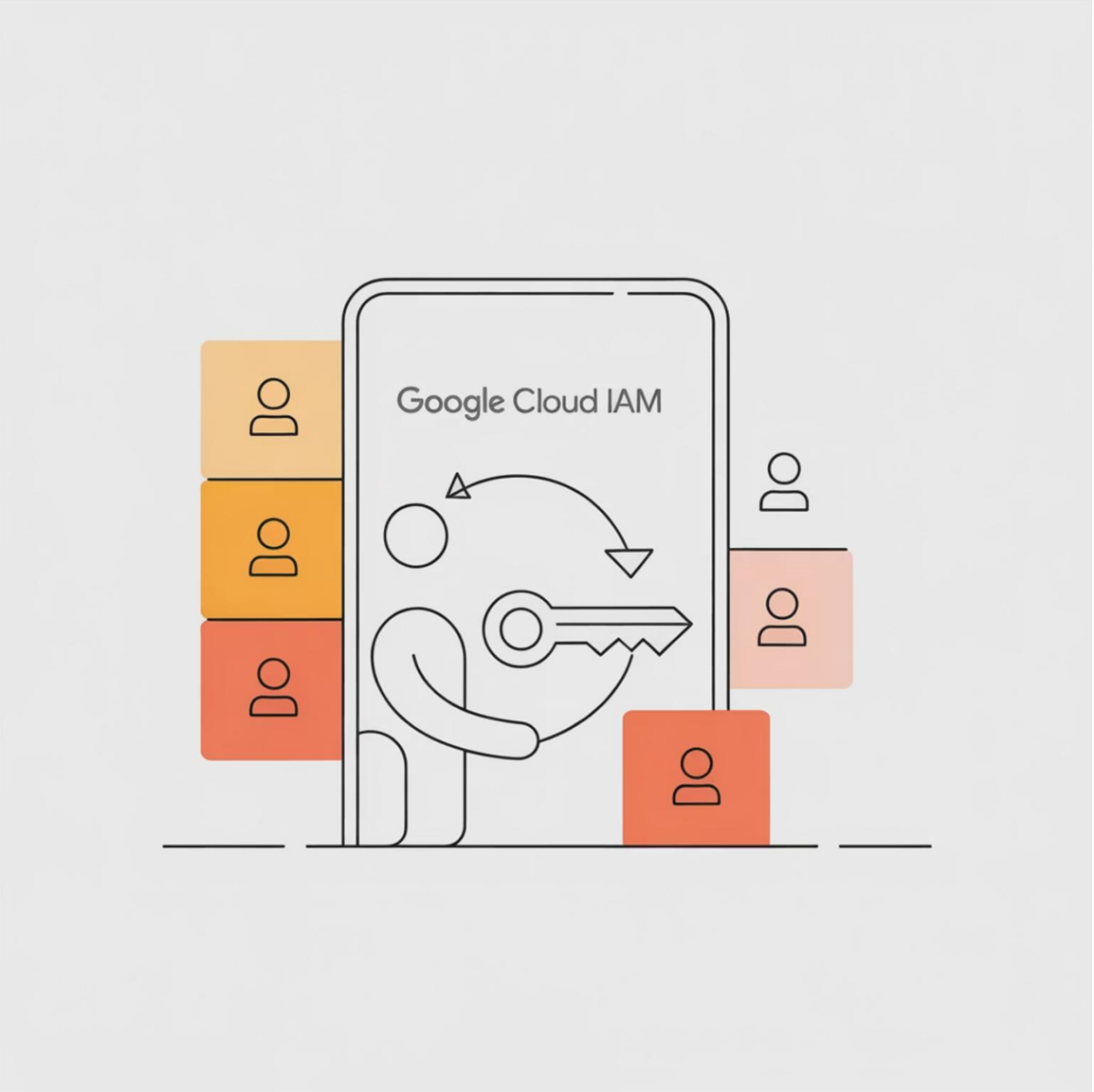
Policy: Binds members to roles for specific resources

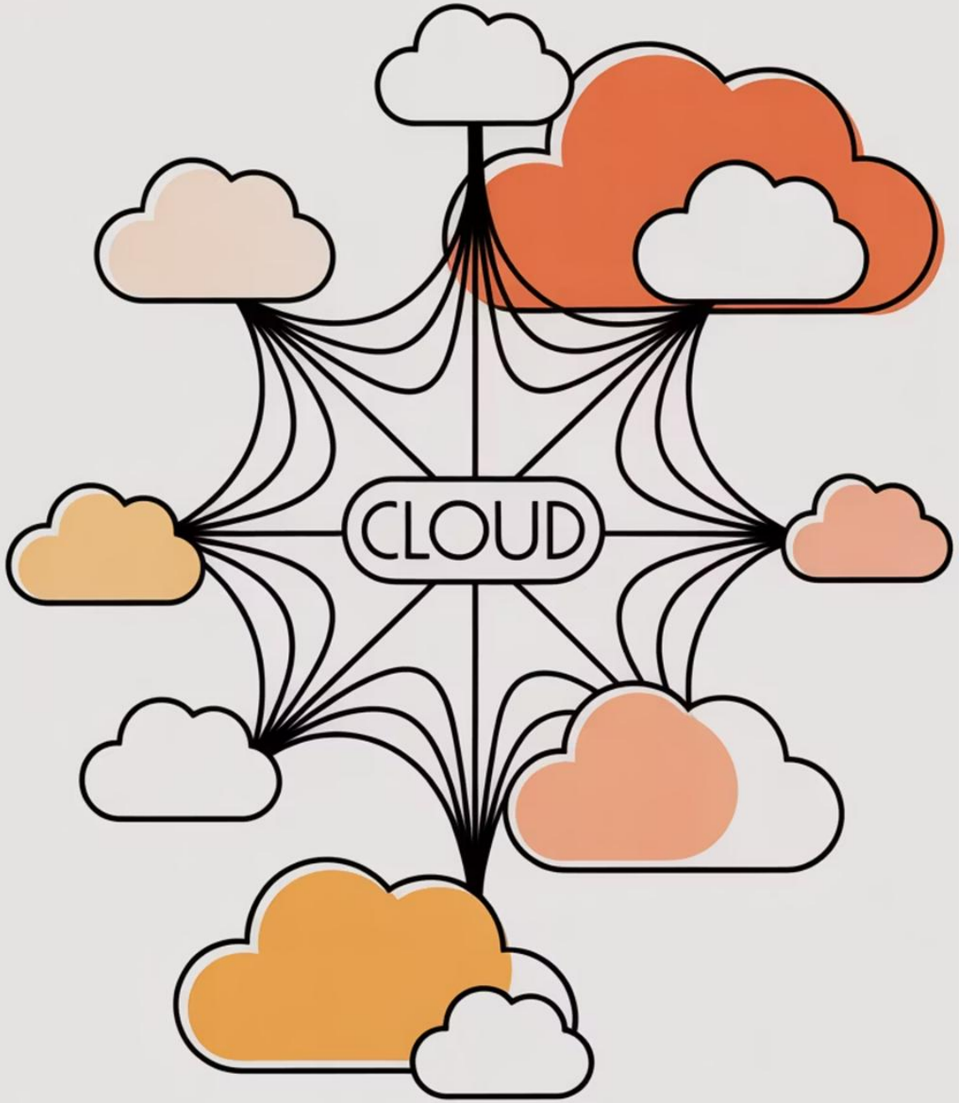
Role Types:

Basic: Owner, Editor, Viewer

Predefined: Service-specific roles

Custom: User-defined permission sets





Cross-Cloud IAM Challenges

Common Challenges:

Identity Fragmentation

Users have separate identities in each cloud platform, leading to credential sprawl and inconsistent access controls

Policy Consistency

Maintaining uniform security policies across diverse cloud environments with different IAM models

Visibility Gaps

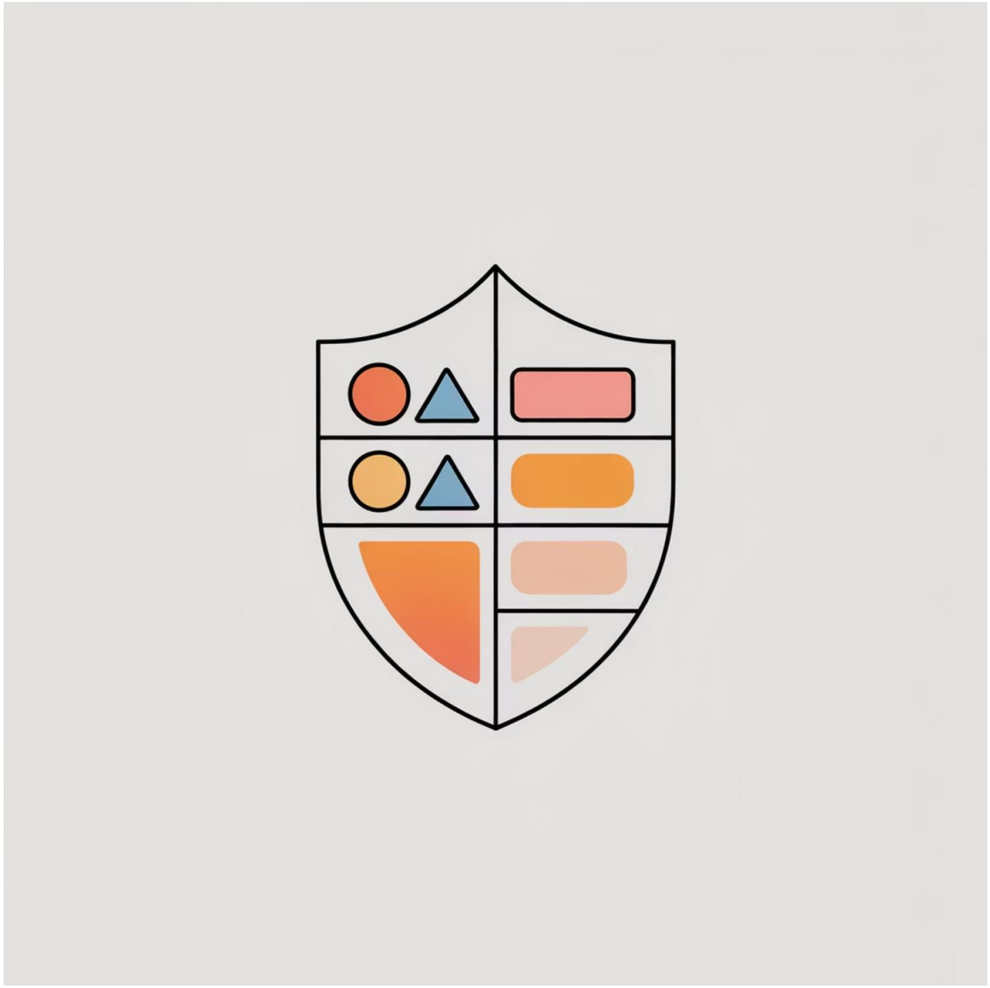
Limited centralized view of permissions and access patterns across multiple cloud providers

Solutions often involve federation services or third-party Cloud IAM platforms that act as a central identity hub.

Chapter 5: Security Policies & Governance in IAM

Establishing effective governance frameworks to ensure consistent security across your organization.

Defining IAM Security Policies



Essential Policy Components:

Authentication standards: Password complexity, MFA requirements

Authorization frameworks: RBAC implementation details

Access review cadence: Quarterly, bi-annual reviews

Compliance mapping: How IAM supports regulatory requirements

Emergency access: Break-glass procedures

Monitoring requirements: Logging and alerting

Policies should be living documents that evolve with organizational needs and the threat landscape.

LETS PRACTICE!

- Create **users & groups** (RBAC basics)
- Apply **least-privilege** with NTFS permissions
- Enforce **local password/lockout policies**
- Turn on **auditing** and verify events

Lab 0 — Setup

```
# Create a working folder
New-Item -Path "C:\IAMLab" -ItemType Directory -Force | Out-Null
```

Lab 1 — RBAC: users, groups, and least-privilege

1A) Create groups (roles)

```
New-LocalGroup -Name "AppAdmins" -Description "Admins for AppData" -
ErrorAction SilentlyContinue
New-LocalGroup -Name "AppUsers"  -Description "Users for AppData"  -
ErrorAction SilentlyContinue
```

1B) Create users and assign to roles

```
# Securely prompt for passwords
$pw1 = Read-Host "Password for user studentA" -AsSecureString
$pw2 = Read-Host "Password for user studentB" -AsSecureString

# Create users
New-LocalUser  -Name "studentA" -FullName "Student A" -Password $pw1 -
PasswordNeverExpires:$false -AccountNeverExpires:$true -ErrorAction SilentlyContinue
New-LocalUser  -Name "studentB" -FullName "Student B" -Password $pw2 -
PasswordNeverExpires:$false -AccountNeverExpires:$true -ErrorAction SilentlyContinue

# Add to groups (roles)
Add-LocalGroupMember -Group "AppAdmins" -Member "studentA" -ErrorAction
SilentlyContinue
Add-LocalGroupMember -Group "AppUsers"  -Member "studentB" -ErrorAction
SilentlyContinue
```

Delete student B from appusers and add there student Y

Minimal RBAC demo (create user, group, folder, set rights)

```
# IAM-Demo.ps1  (run as Admin)

# 1) Folder + role (group)
New-Item C:\IAMDEMO -ItemType Directory -Force | Out-Null
New-LocalGroup AppUsers -ErrorAction SilentlyContinue | Out-Null

# 2) User in that role
$pw = Read-Host "Set a password for demoUser" -AsSecureString
New-LocalUser demoUser -Password $pw -FullName "Demo User" -
ErrorAction SilentlyContinue | Out-Null
Add-LocalGroupMember AppUsers demoUser -ErrorAction
SilentlyContinue

# 3) Simple, clear permissions (stop inheritance; give rights)
icacls C:\IAMDEMO /inheritance:d > $null
icacls C:\IAMDEMO /grant:r "$env:USERNAME:(F)" > $null      # you =
Full Control
icacls C:\IAMDEMO /grant:r "AppUsers:(M)" > $null           # role
= Modify

# 4) Show result
icacls C:\IAMDEMO
Write-Host "`n✅ Setup done. Test with the commands below."
```

Lab 2 — Local “security policies”: password & lockout

On standalone Windows,lets set local policy via `net accounts`.

```
# View current policy
net accounts

# Enforce stronger rules (example values)
net accounts /minpwlen:12 /maxpwage:60 /minpwage:1 /uniquepw:5
net accounts /lockoutthreshold:5 /lockoutwindow:30
/lockoutduration:30

# Re-check
net accounts
```

Clean-Up Script

```
Write-Host "🧹 Cleaning up IAM Demo..." -ForegroundColor Cyan

# 1. Remove demo user from any groups
"demoUser" | ForEach-Object {
    Try { Remove-LocalGroupMember -Group "AppUsers" -Member $_ -ErrorAction
SilentlyContinue } Catch {}
    Try { Remove-LocalGroupMember -Group "Administrators" -Member $_ -ErrorAction
SilentlyContinue } Catch {}
}

# 2. Delete demo user and group
Try { Remove-LocalUser -Name "demoUser" -ErrorAction SilentlyContinue } Catch {}
Try { Remove-LocalGroup -Name "AppUsers" -ErrorAction SilentlyContinue } Catch {}

# 3. Delete IAMDEMO folder
Try { Remove-Item -Path "C:\IAMDEMO" -Recurse -Force -ErrorAction SilentlyContinue }
Catch {}

# 4. Confirm cleanup
Write-Host "`n✅ IAM demo has been fully cleared. All users, groups, and folders
removed." -ForegroundColor Gree
```