Azure Identity & Access Management (IAM)

Status	Not Started

Azure Identity & Access Management (IAM)

Complete Guide for Users

Document Version: 5.0

Date: January 2026

Audience: All Technical Levels - From Beginners to Experts

Executive Summary: The Hotel Analogy

Think of Azure as a Luxury Hotel:

Azure Component	Hotel Analogy	Purpose
Azure AD	Hotel Registration Desk	Verifies who you are
Users/Groups	Guests & Guest Lists	People who need access
Roles	Key Cards (Guest, Staff, Manager)	What you can do
Scope	Room Number (Floor, Room, Safe)	Where you have access
Service Principal	Service Entrance/Staff Door	Automated/system access
RBAC	Hotel Security Policy	Rules for who can do what

1. Azure Active Directory: The Foundation

1.1 What is Azure AD?

Azure Active Directory is **Microsoft's cloud-based identity and access management service**. It's like a digital ID card system for your organization.

Simple Explanation:

Think of Azure AD as:

- **Phone Contacts App** for your organization
- **Digital Receptionist** that knows everyone
- **Security Guard** that checks IDs
- **Key Distribution Center** for digital keys

1.2 Key Components of Azure AD

Users: Digital Employees

bash

```
# Create a user in Azure AD
az ad user create \
  --display-name "John Doe" \
  --user-principal-name "john.doe@mycompany.com" \
  --password "SecurePassword123!" \
  --force-change-password-next-sign-in true
```

Types of Users:

- **Member Users:** Regular employees (live in your Azure AD)
- **Guest Users:** External collaborators (like contractors)
- **Cloud-only:** Exists only in Azure AD
- **Synchronized:** From on-premises AD (Hybrid)

Groups: Logical Collections

bash

```
# Create a security group
az ad group create \
  --display-name "App Developers" \
  --mail-nickname "appdevs"

# Add user to group
az ad group member add \
  --group "App Developers" \
  --member-id $(az ad user show \
    --id "john.doe@mycompany.com" \
    --query id -o tsv)
```

Group Types:

- **Security Groups:** For permissions (like "Finance Team")
- **Microsoft 365 Groups:** For collaboration (like "Project Alpha Team")
- **Distribution Lists:** Email groups only

1.3 Authentication vs Authorization

Authentication (Who are you?):

text

```
[User] → [Credentials] → [Azure AD] → ✅/❌ "Are you who you say?"
```

Authorization (What can you do?):

text

```
[Authenticated User] → [RBAC Check] → ✅/❌ "Can you do this?"
```

Real-world Example:

- **Authentication:** Showing ID at hotel check-in
- **Authorization:** Room key only opens your room, not all rooms

---

2. Role-Based Access Control (RBAC)

2.1 The RBAC Concept

Simple Analogy: Hotel Key Cards

- **Guest Key:** Opens only your room (Reader role)
- **Housekeeping Key:** Opens all rooms on your floor (Contributor role)
- **Manager Key:** Opens all rooms + staff areas (Owner role)
- **Master Key:** Everything + can make new keys (Owner + User Access Administrator)

2.2 Built-in RBAC Roles

Common Built-in Roles:

bash

```
# List built-in roles
az role definition list --query "[?roleType=='BuiltInRole'].roleName" --output t
able
```

Top 10 Most Used Roles:

Role	Analogy	What They Can Do	CLI Example
Owner	Hotel Owner	Full access, manage permissions	`--role Owner`
Contributor	Hotel Manager	Create/manage everything but permissions	`--role Contributor`
Reader	Hotel Guest	View everything, no changes	`--role Reader`
User Access Administrator	Key Master	Manage access for others	`--role "User Access Administrator"`
Virtual Machine Contributor	VM Technician	Manage VMs only	`--role "Virtual Machine Contributor"`

Role	Analogy	What They Can Do	CLI Example
Storage Account Contributor	Storage Manager	Manage storage only	`--role "Storage Account Contributor"`
Network Contributor	Network Engineer	Manage networking only	`--role "Network Contributor"`
Web Plan Contributor	Web Host Manager	Manage web apps only	`--role "Web Plan Contributor"`
SQL DB Contributor	Database Admin	Manage databases only	`--role "SQL DB Contributor"`
Monitoring Contributor	Security Camera Operator	View and manage monitoring	`--role "Monitoring Contributor"`

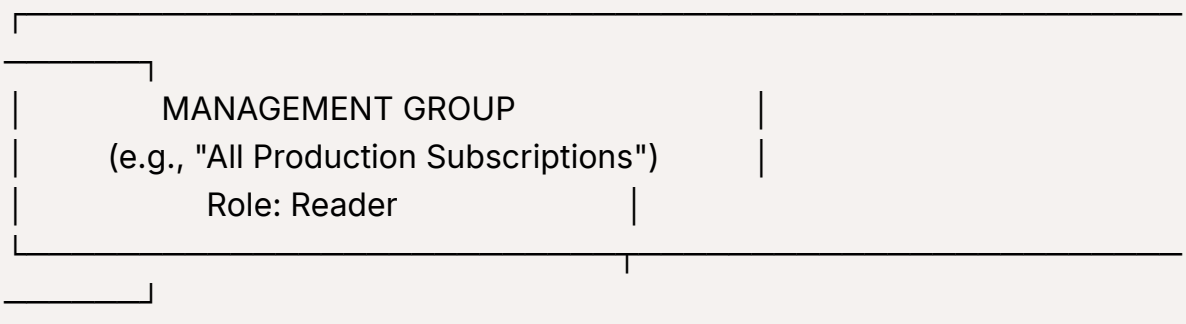
2.3 Permission Scopes: The Hierarchy

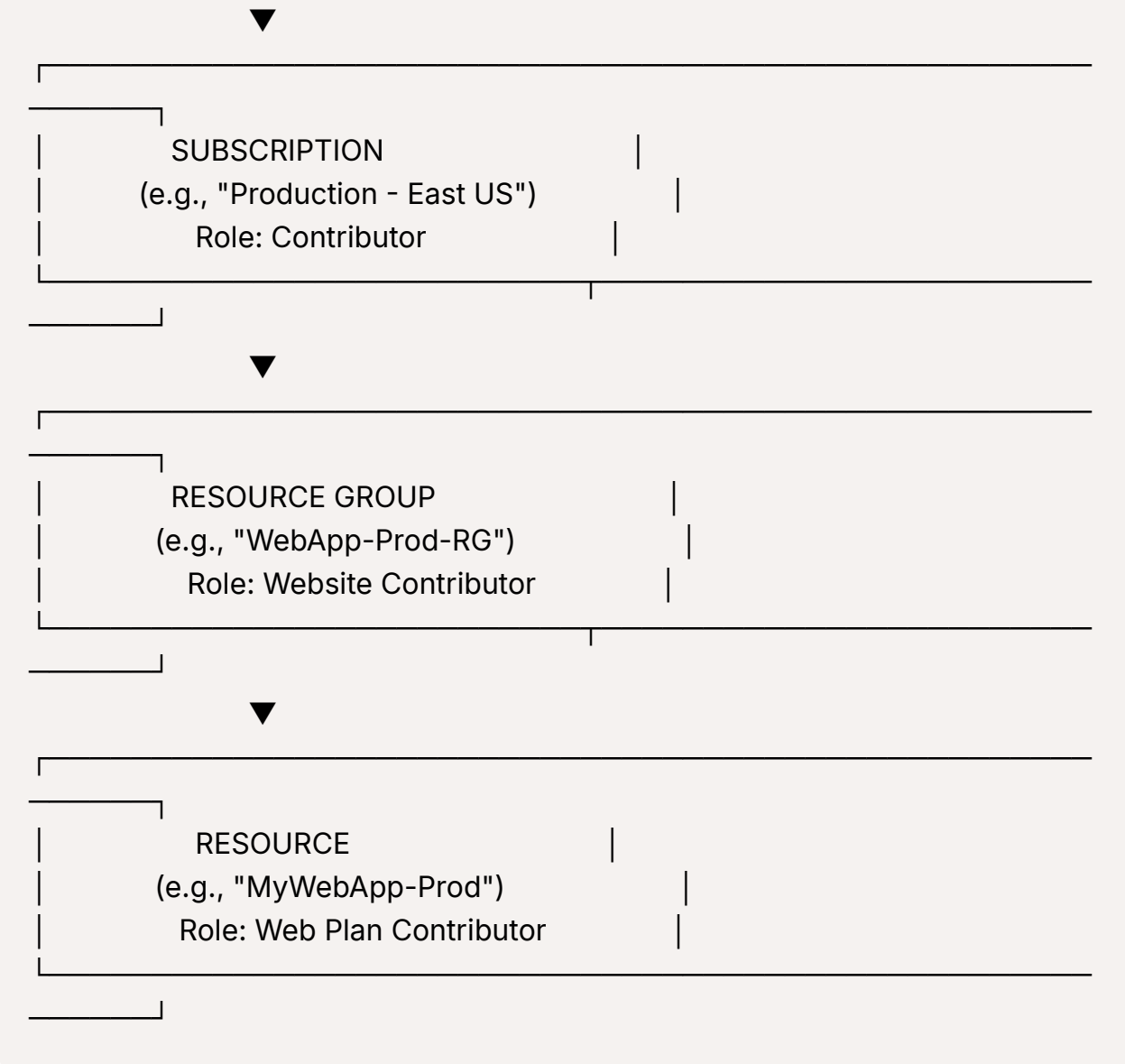
Scope Levels (Most broad → Most specific):

text

```
Management Group (Multiple subscriptions)
      ↓
   Subscription (Billing unit)
      ↓
Resource Group (Logical container)
      ↓
   Resource (VM, Storage, etc.)
      ↓
  Child Resource (Disk, NIC, etc.)
```

Visual Hierarchy:





How Inheritance Works:

- Role at **Management Group** → Applies to all children
- Role at **Subscription** → Applies to all Resource Groups & Resources
- Role at **Resource Group** → Applies to all Resources in that RG
- Role at **Resource** → Applies only to that resource

2.4 Assigning RBAC Roles

CLI Examples for Different Scopes:

Management Group Scope:

```
# Assign Reader role at Management Group level
az role assignment create \
  --assignee "john.doe@mycompany.com" \
  --role "Reader" \
  --scope "/providers/Microsoft.Management/managementGroups/MyManagementGroup"
```

Subscription Scope:

```
# Assign Contributor role at Subscription level
az role assignment create \
  --assignee "john.doe@mycompany.com" \
  --role "Contributor" \
  --scope "/subscriptions/00000000-0000-0000-0000-000000000000"
```

Resource Group Scope:

```
# Assign VM Contributor role at Resource Group level
az role assignment create \
  --assignee "john.doe@mycompany.com" \
  --role "Virtual Machine Contributor" \
  --resource-group "MyResourceGroup"
```

Resource Scope:

```
# Assign specific role to a single VM
az role assignment create \
  --assignee "john.doe@mycompany.com" \
  --role "Virtual Machine Contributor" \
  --scope "/subscriptions/{sub-id}/resourceGroups/MyRG/providers/Microsoft.Compute/virtualMachines/MyVM"
```

Check Current Assignments:

```
# List all role assignments for current subscription
az role assignment list --all --output table

# List assignments for specific user
az role assignment list --assignee "john.doe@mycompany.com" --output tabl
e

# List assignments at specific scope
az role assignment list --resource-group "MyResourceGroup" --output table
```

---

3. Service Principals: The Robot Accounts

3.1 What are Service Principals?

Simple Analogy: Service Principals are like **robot employees** or **service accounts**:

- They don't have lunch breaks
- They work 24/7
- They follow programmed instructions
- They need limited, specific permissions

When to Use Service Principals:

1. **Automation** (Terraform, Ansible, PowerShell scripts)
2. **CI/CD Pipelines** (Azure DevOps, GitHub Actions)
3. **Applications** that need to access Azure resources
4. **Scheduled tasks** that run automatically

3.2 Creating Service Principals

Create with CLI:

```
# Create a Service Principal for Terraform
az ad sp create-for-rbac \
  --name "terraform-sp" \
  --role "Contributor" \
  --scopes "/subscriptions/00000000-0000-0000-0000-000000000000" \
  --years 2

# Output will show:
# appId (Client ID)
# password (Client Secret)
# tenant (Tenant ID)
```

Create with Specific Permissions:

```
# Create SP with limited permissions
az ad sp create-for-rbac \
  --name "webapp-deploy-sp" \
  --role "Web Plan Contributor" \
  --scopes "/subscriptions/{sub-id}/resourceGroups/webapp-rg"

# Create SP for specific resource only
az ad sp create-for-rbac \
  --name "storage-backup-sp" \
  --role "Storage Account Contributor" \
  --scopes "/subscriptions/{sub-id}/resourceGroups/storage-rg/providers/Mic
rosoft.Storage/storageAccounts/mystorage"
```

3.3 Managing Service Principals

List Service Principals:

```
# List all service principals
az ad sp list --display-name "*" --query "[].{Name:displayName, AppId:appl
d}" --output table
```

```
# Get specific SP details
az ad sp show --id "00000000-0000-0000-0000-000000000000"
```

Update/Rotate Credentials:

```
# Reset SP password
az ad sp credential reset \
  --name "terraform-sp" \
  --years 2

# Add new credential without removing old ones
az ad sp credential create \
  --id $(az ad sp show --id "terraform-sp" --query appId -o tsv) \
  --name "NewCredential" \
  --end-date "2025-12-31"
```

Delete Service Principal:

```
# Delete by name
az ad sp delete --id "terraform-sp"

# Delete by application ID
az ad sp delete --id "00000000-0000-0000-0000-000000000000"
```

3.4 Using Service Principals

Authenticate with Service Principal:

```
# Login with Service Principal
az login --service-principal \
  --username "00000000-0000-0000-0000-000000000000" \
  --password "ClientSecret" \
  --tenant "TenantID"
```

In Terraform (main.tf):

```
provider "azurerm"{
  features {}

  subscription_id = "00000000-0000-0000-0000-000000000000"
  client_id       = "00000000-0000-0000-0000-000000000000" # appId
  client_secret   = var.client_secret
  tenant_id       = "00000000-0000-0000-0000-000000000000"
}
```

In Azure DevOps Pipeline (YAML):

```
steps:
- task: AzureCLI@2
  inputs:
    azureSubscription: 'my-service-connection'
    scriptType: 'bash'
    scriptLocation: 'inlineScript'
    inlineScript: |
      az group list --output table
```

---

4. Least Privilege Design

4.1 The Principle of Least Privilege

Analogy: Airport Security

- **Passenger:** Can go to gate, buy food, use restroom
- **Airline Staff:** Can access staff areas, check baggage
- **Pilot:** Can access cockpit, fly plane
- **Security:** Can access all areas but not fly plane

NOBODY gets "Master Key" unless absolutely necessary!

4.2 Implementing Least Privilege

Step 1: Start with "Reader" for Everyone

```
# Default all users to Reader at subscription level
az role assignment create \
  --assignee "engineering-team@mycompany.com" \
  --role "Reader" \
  --scope "/subscriptions/{sub-id}"
```

Step 2: Grant Specific Permissions as Needed

```
# Grant VM Contributor only to specific Resource Group
az role assignment create \
  --assignee "vm-admin@mycompany.com" \
  --role "Virtual Machine Contributor" \
  --scope "/subscriptions/{sub-id}/resourceGroups/vm-rg"

# Grant Network Contributor only to network resources
az role assignment create \
  --assignee "network-admin@mycompany.com" \
  --role "Network Contributor" \
  --scope "/subscriptions/{sub-id}/resourceGroups/network-rg"
```

Step 3: Use Custom Roles for Fine-Grained Control

```
# Create custom role with specific permissions
az role definition create --role-definition '{
  "Name": "WebApp Reader Plus",
  "Description": "Can read web apps and restart them",
  "Actions": [
    "Microsoft.Web/sites/Read",
    "Microsoft.Web/sites/restart/Action",
    "Microsoft.Web/sites/config/Read"
  ],
```

```
  "NotActions": [],
  "AssignableScopes": [
    "/subscriptions/{sub-id}"
  ]
}'

# Assign custom role
az role assignment create \
  --assignee "webapp-support@mycompany.com" \
  --role "WebApp Reader Plus" \
  --scope "/subscriptions/{sub-id}/resourceGroups/webapp-rg"
```

4.3 Real-World Least Privilege Examples

Example 1: Development Team

```
# Devs need to create resources but not manage networking
az role assignment create \
  --assignee "dev-team-group" \
  --role "Contributor" \
  --scope "/subscriptions/{sub-id}/resourceGroups/dev-rg"

# But remove network permissions
az role assignment create \
  --assignee "dev-team-group" \
  --role "Network Contributor" \
  --scope "/subscriptions/{sub-id}/resourceGroups/dev-rg" \
  --condition "false"  # Deny assignment
```

Example 2: Database Administrators

bash

```
# DBAs need full SQL access but nothing else
az role assignment create \
  --assignee "dba-group" \
```

```
  --role "SQL DB Contributor" \
  --scope "/subscriptions/{sub-id}/resourceGroups/database-rg"
```

Example 3: Monitoring Team

```
# Monitoring team needs read access everywhere
az role assignment create \
  --assignee "monitoring-team" \
  --role "Reader" \
  --scope "/subscriptions/{sub-id}"

# Plus specific monitoring permissions
az role assignment create \
  --assignee "monitoring-team" \
  --role "Monitoring Contributor" \
  --scope "/subscriptions/{sub-id}"
```

4.4 Deny Assignments (The "Never Allow" Rule)

```
# Create a deny assignment (preview feature)
# This OVERRULES any allow assignments

# Example: Deny delete operations for compliance
az role assignment create \
  --assignee "contractor@external.com" \
  --role "Deny Delete" \
  --scope "/subscriptions/{sub-id}"
```

Built-in Deny Roles:

- **DenyAll:** No operations allowed
- **DenyDelete:** Can't delete resources
- **DenyWrite:** Can't create or modify

---

5. Real-World Scenarios & Architectures

5.1 Scenario 1: E-commerce Company

Organization Structure:

```
Management Group: E-Commerce Corp
├── Subscription: Production
│   ├── RG: Web-App-Prod (Web Team: Contributor)
│   ├── RG: Database-Prod (DB Team: SQL Contributor)
│   └── RG: Storage-Prod (Storage Team: Storage Contributor)
├── Subscription: Development
│   ├── RG: Web-App-Dev (All Devs: Contributor)
│   └── RG: Shared-Services (All: Reader)
└── Subscription: Security
    └── RG: Security-Monitoring (Security Team: Owner)
```

CLI Setup:

```
#!/bin/bash
# Setup for E-commerce Company

# Create Resource Groups
az group create --name Web-App-Prod --location eastus
az group create --name Database-Prod --location eastus
az group create --name Storage-Prod --location eastus

# Create Groups in Azure AD
az ad group create --display-name "Web-Team" --mail-nickname "webteam"
az ad group create --display-name "DB-Team" --mail-nickname "dbteam"
az ad group create --display-name "Storage-Team" --mail-nickname "storage
team"

# Assign Roles
az role assignment create \
  --assignee $(az ad group show --group "Web-Team" --query id -o tsv) \
```

```
  --role Contributor \
  --resource-group Web-App-Prod

az role assignment create \
  --assignee $(az ad group show --group "DB-Team" --query id -o tsv) \
  --role "SQL DB Contributor" \
  --resource-group Database-Prod

az role assignment create \
  --assignee $(az ad group show --group "Storage-Team" --query id -o tsv) \
  --role "Storage Account Contributor" \
  --resource-group Storage-Prod
```

5.2 Scenario 2: Healthcare Organization (HIPAA Compliant)

Security Requirements:

- Strict separation of duties
- Audit trails for all access
- No shared accounts
- Regular access reviews

Implementation:

```
#!/bin/bash
# HIPAA Compliant Setup

# Create custom roles for HIPAA compliance
az role definition create --role-definition '{
  "Name": "PHI Data Reader",
  "Description": "Can read PHI data but not export",
  "Actions": [
    "Microsoft.Storage/storageAccounts/blobServices/containers/read",
    "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/read"
  ],
  "NotActions": [
```

```
    "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/write",
    "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/delete"
  ],
  "AssignableScopes": [
    "/subscriptions/{sub-id}/resourceGroups/phi-data-rg"
  ]
}'

# Enable Azure AD Privileged Identity Management (PIM)
# Requires Azure AD P2 license
az ad sp create --id "8428ca0a-4e01-4c5e-b598-44e0ee5e85a9"  # PIM Ser
vice Principal

# Setup Just-In-Time (JIT) access
az role assignment create \
  --assignee "doctor@hospital.com" \
  --role "PHI Data Reader" \
  --scope "/subscriptions/{sub-id}/resourceGroups/phi-data-rg" \
  --condition "@Request.context.activationJustification ~ '\bemergency\b'" \
  --description "Emergency access to PHI data"
```

5.3 Scenario 3: DevOps Pipeline with Service Principals

CI/CD Architecture:

```
GitHub Actions / Azure DevOps
        ↓
Service Principal (Limited Permissions)
        ↓
Azure Resources (Deployment)
```

Pipeline Setup:

```
# Create SP for each environment
az ad sp create-for-rbac \
  --name "github-actions-dev" \
```

```
  --role Contributor \
  --scopes "/subscriptions/{sub-id}/resourceGroups/dev-*" \
  --sdk-auth

az ad sp create-for-rbac \
  --name "github-actions-prod" \
  --role Contributor \
  --scopes "/subscriptions/{sub-id}/resourceGroups/prod-*" \
  --sdk-auth

# Store secrets in GitHub
# GitHub → Settings → Secrets → Add:
# AZURE_CREDENTIALS_DEV (paste JSON output)
# AZURE_CREDENTIALS_PROD (paste JSON output)
```

GitHub Actions Workflow:

```
name: Deploy to Azure
on: [push]
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2

    - name: Login to Azure
      uses: azure/login@v1
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS_DEV }}

    - name: Deploy Infrastructure
      run: |
        az group create --name myapp-dev --location eastus
        az deployment group create \
```

```
    --resource-group myapp-dev \
    --template-file azuredeploy.json
```

---

6. Advanced IAM Concepts

6.1 Managed Identities (The Best Service Principals)

What are Managed Identities?

- Automatically managed Service Principals
- No credentials to manage/rotate
- Two types: System-assigned & User-assigned

System-assigned Managed Identity:

```
# Enable on VM
az vm identity assign \
  --name MyVM \
  --resource-group MyRG

# Grant access to Key Vault
az keyvault set-policy \
  --name MyKeyVault \
  --object-id $(az vm show \
    --name MyVM \
    --resource-group MyRG \
    --query identity.principalId -o tsv) \
  --secret-permissions get list
```

User-assigned Managed Identity:

```
# Create user-assigned identity
az identity create \
  --name MyManagedIdentity \
  --resource-group MyRG \
```

```
  --location eastus

# Assign to VM
az vm identity assign \
  --name MyVM \
  --resource-group MyRG \
  --identities MyManagedIdentity
```

6.2 Conditional Access Policies

Require MFA for Admin Roles:

```
# This is configured in Azure Portal, but here's the concept:
# 1. Go to Azure AD → Security → Conditional Access
# 2. Create policy: "Require MFA for Admins"
# 3. Target: All users, Cloud apps: Microsoft Azure Management
# 4. Conditions: User risk level = Medium/High
# 5. Access controls: Require MFA
```

Block access from specific countries:

```
# Using Microsoft Graph PowerShell
Connect-MgGraph -Scopes "Policy.Read.All", "Policy.ReadWrite.ConditionalA
ccess"

$policy = @{
    displayName = "Block non-US access"
    state = "enabled"
    conditions = @{
        applications = @{
            includeApplications = "797f4846-ba00-4fd7-ba43-dac1f8f63013" # A
zure Management
        }
        locations = @{
            includeLocations = "All"
            excludeLocations = "US" # Exclude United States
```

```
    }
  }
  grantControls = @{
    operator = "OR"
    builtInControls = @("block")
  }
}

New-MgIdentityConditionalAccessPolicy -BodyParameter $policy
```

6.3 Access Reviews

Automate access reviews:

```
# Create access review for all role assignments
# Note: This is a preview feature, mostly done via portal/PowerShell

# Check for stale assignments (older than 90 days)
az role assignment list \
  --all \
  --query "[?contains(principalType, 'User')].{Name:principalName, Role:roleD
efinitionName, Scope:scope, Created:createdOn}" \
  --output table | Sort-Object Created
```

---

7. Monitoring & Auditing

7.1 Activity Logs

View all IAM events:

```
# Get all role assignment changes
az monitor activity-log list \
  --resource-group MyRG \
  --namespace "Microsoft.Authorization" \
  --offset 7d \
```

```
  --output table

# Export logs to storage for compliance
az monitor diagnostic-settings create \
  --resource "/subscriptions/{sub-id}" \
  --name "IAMAuditLogs" \
  --storage-account "mystorageaccount" \
  --logs '[{"category": "Administrative", "enabled": true}]'
```

7.2 Azure AD Audit Logs

Check sign-ins and access attempts:

```
# Requires Azure AD Premium P1/P2
# Use Microsoft Graph API for detailed logs

# Basic sign-in logs via CLI (limited)
az rest \
  --method get \
  --url "https://graph.microsoft.com/v1.0/auditLogs/signIns?top=10" \
  --headers "Content-Type=application/json"
```

7.3 Security Center Recommendations

```
# Get IAM security recommendations
az security task list \
  --resource-id "/subscriptions/{sub-id}" \
  --query "[?contains(description, 'RBAC') || contains(description, 'IAM')]" \
  --output table
```

---

8. Best Practices Checklist

8.1 Daily Operations

- Use **groups** not individual users for role assignments

- Implement **Just-In-Time** access where possible
- Regular **access reviews** (quarterly)
- Monitor for **stale accounts** (90-day rule)
- Use **Managed Identities** instead of Service Principals when possible

8.2 Security Hardening

- Enable **Multi-Factor Authentication** for all users
- Use **Conditional Access** policies
- Implement **Privileged Identity Management**
- Set up **alerting** for suspicious activities
- Regular **permission audits**

8.3 Compliance

- Maintain **documentation** of all roles and assignments
- Implement **separation of duties**
- Enable **logging** for all IAM actions
- Regular **compliance scans**
- **Training** for staff on IAM policies

8.4 Cost Optimization

- Remove **unused role assignments**
- Clean up **stale Service Principals**
- Use **Resource Group** scope instead of Subscription when possible
- Implement **budget alerts** for subscription

---

9. Troubleshooting Common Issues

Issue 1: "Access Denied" Errors

```
# Check user's effective permissions
az role assignment list --assignee "user@company.com" --all --output table

# Check if there's a deny assignment
az rest --method get \
  --url "https://management.azure.com/subscriptions/{sub-id}/providers/Microsoft.Authorization/denyAssignments?api-version=2022-04-01"

# Test specific permission
az rest --method post \
  --url "https://management.azure.com/subscriptions/{sub-id}/resourcegroups/MyRG?api-version=2022-09-01" \
  --body "{}"
```

Issue 2: Service Principal Authentication Fails

```
# Check if SP exists
az ad sp show --id "00000000-0000-0000-0000-000000000000"

# Check if credentials are expired
az ad sp credential list --id "00000000-0000-0000-0000-000000000000"

# Check role assignments
az role assignment list --assignee "00000000-0000-0000-0000-000000000000" --all
```

Issue 3: Inheritance Not Working

```
# Check scope hierarchy
az resource show --ids "/subscriptions/{sub-id}/resourceGroups/MyRG"

# Check for explicit deny at lower scope
az role assignment list --scope "/subscriptions/{sub-id}/resourceGroups/MyRG" --all
```

```
# View effective permissions
az role assignment list --assignee "user@company.com" --scope "/subscripti
ons/{sub-id}" --include-inherited
```

10. Complete Deployment Example

10.1 Enterprise IAM Setup Script

```
#!/bin/bash
# Complete Enterprise IAM Setup

set -e

# Configuration
TENANT_ID="your-tenant-id"
SUBSCRIPTION_ID="your-subscription-id"
COMPANY_DOMAIN="mycompany.com"
LOCATION="eastus"

# Login
az login --tenant $TENANT_ID

# Set subscription
az account set --subscription $SUBSCRIPTION_ID

echo "=== Step 1: Create Resource Groups ==="
az group create --name "platform-rg" --location $LOCATION
az group create --name "networking-rg" --location $LOCATION
az group create --name "compute-rg" --location $LOCATION
az group create --name "storage-rg" --location $LOCATION

echo "=== Step 2: Create Azure AD Groups ==="
declare -A GROUPS=(
```

```
    [Platform-Admins]="Platform Administration Team"
    [Network-Admins]="Network Administration Team"
    [VM-Admins]="Virtual Machine Administration Team"
    [Storage-Admins]="Storage Administration Team"
    [Developers]="Application Development Team"
    [Auditors]="Security and Audit Team"
)

for GROUP_NAME in "${!GROUPS[@]}"; do
    echo "Creating group: $GROUP_NAME"
    az ad group create \
        --display-name "$GROUP_NAME" \
        --mail-nickname "$GROUP_NAME" \
        --description "${GROUPS[$GROUP_NAME]}" || echo "Group may already
exist"
done

echo "=== Step 3: Assign RBAC Roles ==="

# Platform Admins - Owner on platform RG only
PLATFORM_GROUP_ID=$(az ad group show --group "Platform-Admins" --que
ry id -o tsv)
az role assignment create \
    --assignee $PLATFORM_GROUP_ID \
    --role "Owner" \
    --resource-group "platform-rg"

# Network Admins - Network Contributor on networking RG
NETWORK_GROUP_ID=$(az ad group show --group "Network-Admins" --que
ry id -o tsv)
az role assignment create \
    --assignee $NETWORK_GROUP_ID \
    --role "Network Contributor" \
    --resource-group "networking-rg"

# VM Admins - VM Contributor on compute RG
```

```
VM_GROUP_ID=$(az ad group show --group "VM-Admins" --query id -o tsv)
az role assignment create \
    --assignee $VM_GROUP_ID \
    --role "Virtual Machine Contributor" \
    --resource-group "compute-rg"

# Storage Admins - Storage Contributor on storage RG
STORAGE_GROUP_ID=$(az ad group show --group "Storage-Admins" --query
id -o tsv)
az role assignment create \
    --assignee $STORAGE_GROUP_ID \
    --role "Storage Account Contributor" \
    --resource-group "storage-rg"

# Developers - Contributor on all RGs (except platform)
DEV_GROUP_ID=$(az ad group show --group "Developers" --query id -o tsv)
for RG in "networking-rg" "compute-rg" "storage-rg"; do
    az role assignment create \
        --assignee $DEV_GROUP_ID \
        --role "Contributor" \
        --resource-group $RG
done

# Auditors - Reader on everything
AUDIT_GROUP_ID=$(az ad group show --group "Auditors" --query id -o tsv)
az role assignment create \
    --assignee $AUDIT_GROUP_ID \
    --role "Reader" \
    --scope "/subscriptions/$SUBSCRIPTION_ID"

echo "=== Step 4: Create Service Principals for Automation ==="

# Terraform Service Principal
echo "Creating Terraform SP..."
TF_SP=$(az ad sp create-for-rbac \
    --name "terraform-enterprise" \
```

```
    --role "Contributor" \
    --scopes "/subscriptions/$SUBSCRIPTION_ID/resource-groups/platform-r
g" \
    --years 2)

echo "Terraform SP created. Save these credentials securely:"
echo $TF_SP | jq .

# Pipeline Service Principal
echo "Creating Pipeline SP..."
PIPELINE_SP=$(az ad sp create-for-rbac \
    --name "azure-devops-pipeline" \
    --role "Contributor" \
    --scopes "/subscriptions/$SUBSCRIPTION_ID/resource-groups/compute-r
g" \
    --years 1)

echo "=== Step 5: Create Custom Roles ==="

# Custom role for support team
az role definition create --role-definition '{
    "Name": "Support Engineer",
    "Description": "Can view all resources and restart VMs",
    "Actions": [
        "*/read",
        "Microsoft.Compute/virtualMachines/restart/action",
        "Microsoft.Compute/virtualMachines/start/action",
        "Microsoft.Compute/virtualMachines/powerOff/action"
    ],
    "NotActions": [],
    "DataActions": [],
    "NotDataActions": [],
    "AssignableScopes": [
        "/subscriptions/'$SUBSCRIPTION_ID'"
    ]
}'
```

```
echo "=== Step 6: Enable Monitoring ==="

# Create Log Analytics workspace
az monitor log-analytics workspace create \
    --resource-group "platform-rg" \
    --workspace-name "iam-audit-logs"

# Enable Activity Log export
az monitor diagnostic-settings create \
    --resource "/subscriptions/$SUBSCRIPTION_ID" \
    --name "SubscriptionActivityLogs" \
    --workspace "$(az monitor log-analytics workspace show \
        --resource-group "platform-rg" \
        --workspace-name "iam-audit-logs" \
        --query id -o tsv)" \
    --logs '[{"category": "Administrative", "enabled": true}]'

echo "=== Setup Complete ==="
echo "Summary:"
echo "- Created 4 Resource Groups"
echo "- Created 6 Azure AD Groups"
echo "- Assigned RBAC roles to groups"
echo "- Created 2 Service Principals"
echo "- Created 1 Custom Role"
echo "- Enabled Activity Log monitoring"
echo ""
echo "Next steps:"
echo "1. Add users to appropriate groups"
echo "2. Enable MFA for all users"
echo "3. Set up Conditional Access policies"
echo "4. Schedule quarterly access reviews"
```

10.2 IAM Health Check Script

```
#!/bin/bash
# IAM Health Check Script

echo "=== IAM Health Check Report ==="
echo "Generated: $(date)"
echo ""

echo "1. Checking for Over-privileged Accounts..."
az role assignment list \
  --all \
  --query "[?roleDefinitionName=='Owner' || roleDefinitionName=='Contributo
r'].{Principal:principalName, Role:roleDefinitionName, Scope:scope}" \
  --output table

echo ""
echo "2. Checking for Stale User Assignments (older than 90 days)..."
az role assignment list \
  --all \
  --query "[?contains(principalType, 'User') && contains(roleDefinitionName,
'Owner')].{User:principalName, Role:roleDefinitionName, Created:createdOn}"
\
  --output table

echo ""
echo "3. Checking Service Principals without Recent Usage..."
# This requires Azure AD Premium and Graph API for full check
echo "Note: Full SP usage report requires Azure AD Premium P2"

echo ""
echo "4. Checking for Broken Inheritance..."
# Look for explicit denies or overly specific assignments
az role assignment list \
  --all \
  --query "[?scope contains('resourceGroups') && principalType=='User'].{Us
er:principalName, Scope:scope}" \
```

```
  --output table | head -20

echo ""
echo "5. Checking Custom Roles..."
az role definition list \
  --custom-role-only true \
  --query "[].{Name:roleName, Description:description}" \
  --output table

echo ""
echo "=== Recommendations ==="
echo "1. Review Owner/Contributor assignments"
echo "2. Clean up stale user assignments"
echo "3. Consider converting users to groups"
echo "4. Review custom roles for least privilege"
echo "5. Schedule access reviews"
```

11. Quick Reference Guide

Common CLI Commands Cheat Sheet

Task	Command			
Create User	`az ad user create --display-name "Name" --user-principal-name "email"`			
Create Group	`az ad group create --display-name "GroupName"`			
Add User to Group	`az ad group member add --group "GroupName" --member-id "UserID"`			
Create Service Principal	`az ad sp create-for-rbac --name "SP-Name" --role "Role"`			

Task	Command			
Assign Role	`az role assignment create --assignee "user@email" --role "RoleName"`			
List Role Assignments	`az role assignment list --assignee "user@email"`			
Check Effective Access	`az role assignment list --assignee "user@email" --include-inherited`			
Create Custom Role	`az role definition create --role-definition role.json`			
Get Tenant ID	`az account show --query tenantId -o tsv`			
Get Subscription ID	`az account show --query id -o tsv`			