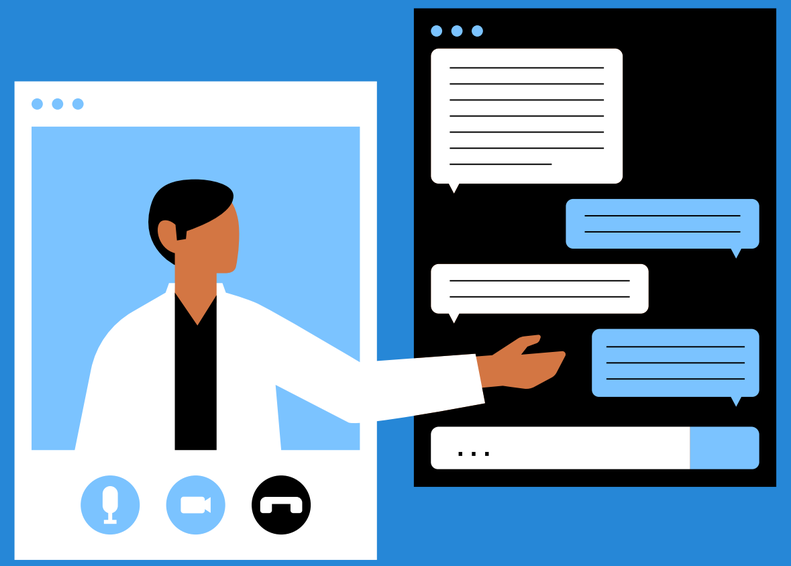
Cloud Identity and Security

Powered by AWS, FreeIPA, Keycloak, Vault



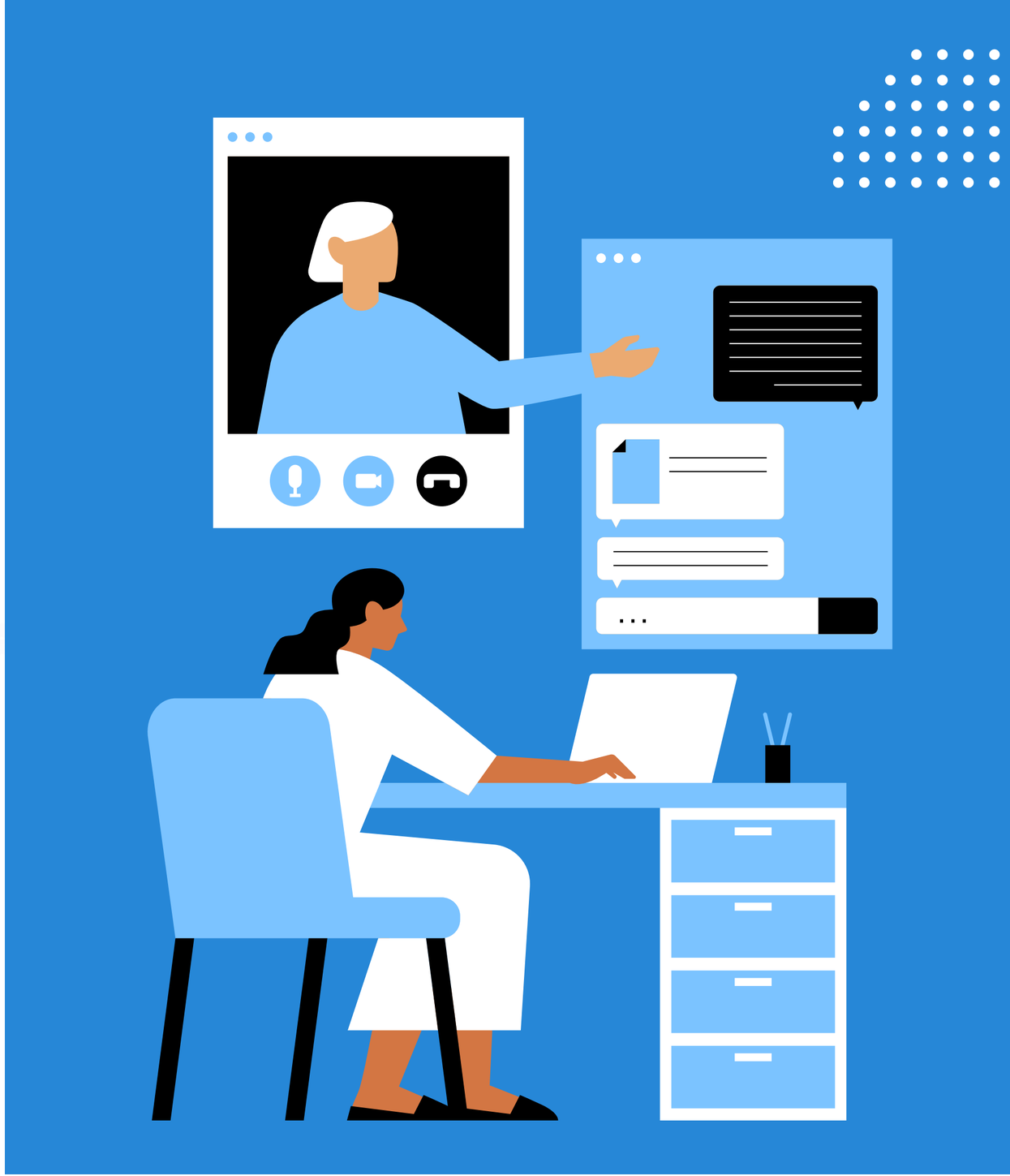
Objectives

- IAM Best Practices
- RBAC & MFA
- Compliance & Data Protection Frameworks
- AWS Shared Responsibility Model
- Cloud Encryption Techniques
- Cloud Incident Response Strategies
- Secrets Management & Secure Access

Introduction to AWS Identity and Security

- Security in the cloud is paramount
- Data breaches and misconfigurations are major risks
- AWS has shared responsibility and built-in security services

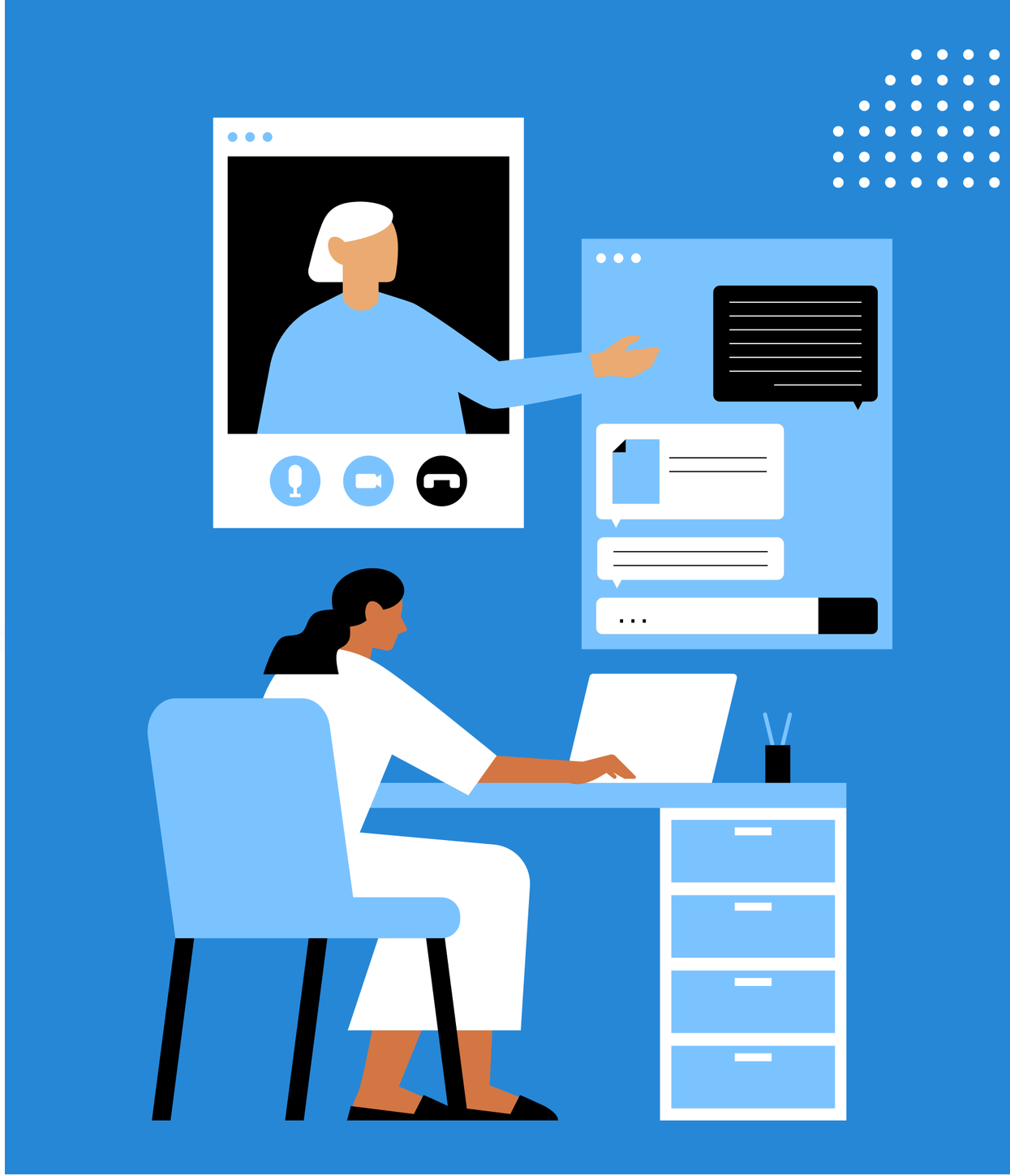
- Lab: Review IAM roles and policies in AWS Management Console
-



IAM Policy Types

- - IAM Policy Types Description - Identity-based policies Attach managed and inline policies to IAM identities (users, groups to which users belong, or roles).
- Resource-based policies - Attach inline policies to resources like S3, SQS and so on.
- Permissions boundaries - Defines the maximum permissions that the identity-based policies can grant to an entity, but does not grant permissions.
- Organizations SCPs - Define the maximum permissions for account members of an organization or organizational unit (OU)
- Access control lists (ACLs) - control which principals in other accounts can access the resource to which the ACL is attached.
- Session policies - Session policies limit permissions for a created session, but do not grant permissions

https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html



Policy Evaluation Logic

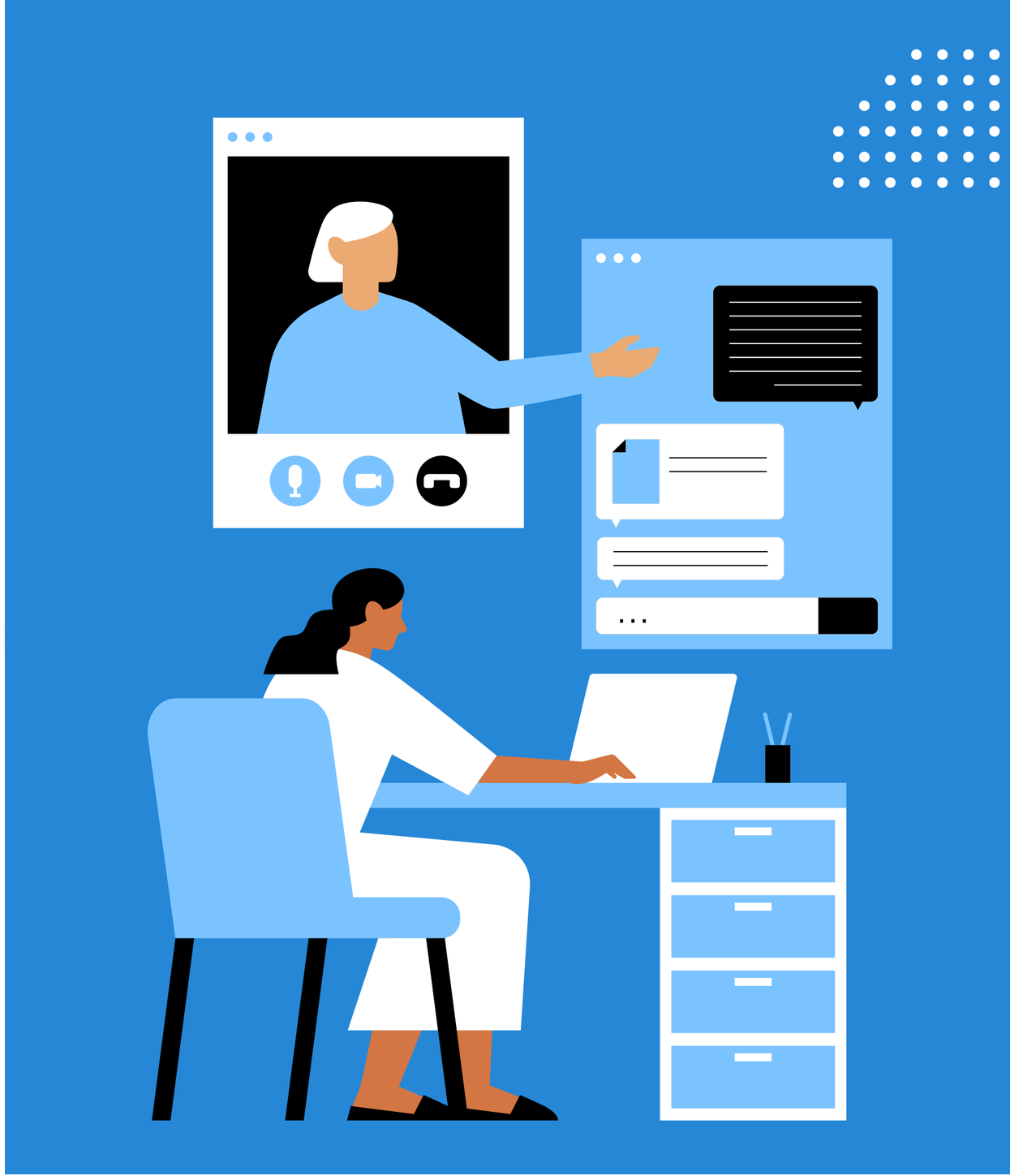
- https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html

- By default, all requests are implicitly denied with the exception of the AWS account root user, which has full access. If user does not have any IAM Policy, it means that all his requests will be denied by default.
- An explicit allow in an identity-based or resource-based policy overrides this default deny.

IAM Best Practices

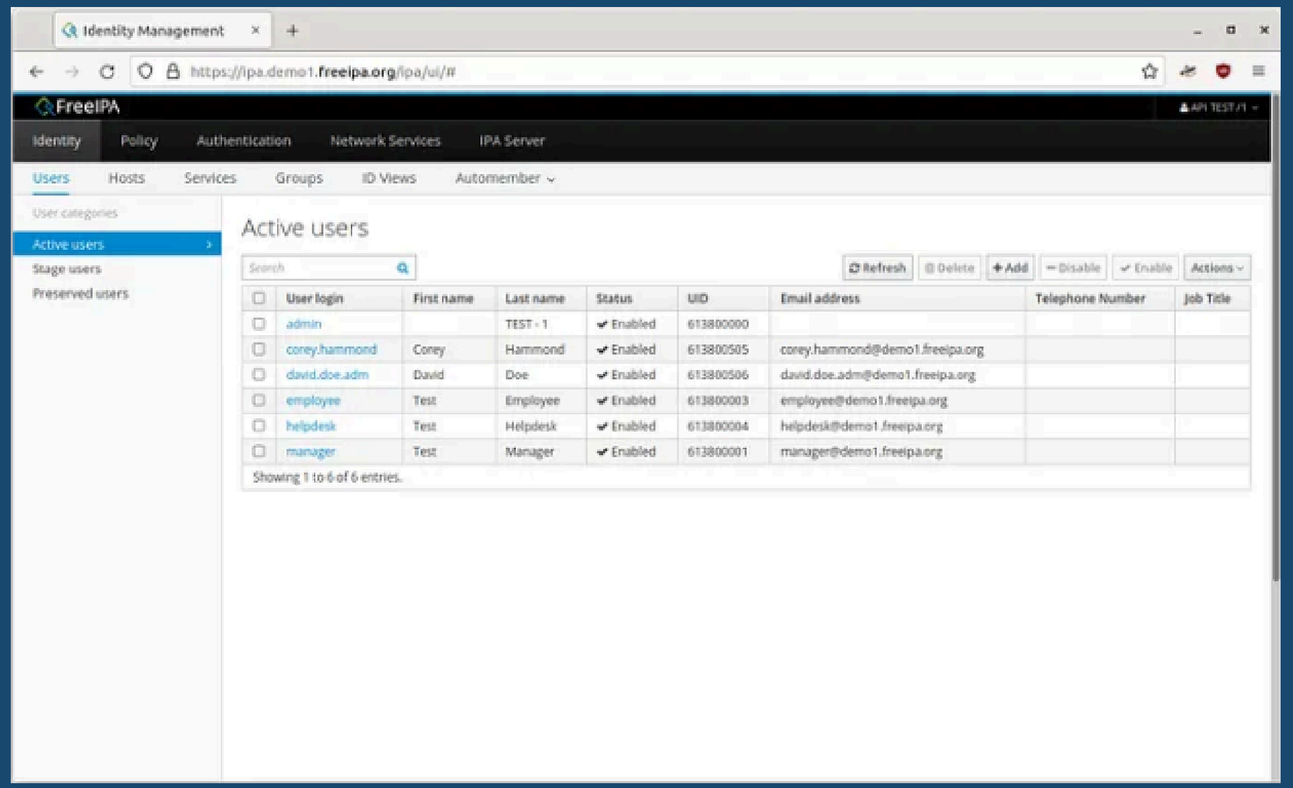
- Principle of Least Privilege: Users should only have access to the resources they need.
- Group-Based Access Control: Define groups like "admins", "devs", "auditors" to simplify policy assignment.
- Logging & Auditing: Enable logging of user actions and access attempts. (CloudTrail & IAM Access Analyzer for auditing)
- Account Lockout Policies: Prevent brute-force attacks by locking accounts after several failed logins.
- Strong Password Policies: Enforce complexity, expiration, and reuse prevention.

- Lab: Create/Test IAM user with a policy using AWS Policy Generator (Policy to allow a user to start or stop EC2 instances)



What is FreeIPA?

- FreeIPA is an open-source integrated identity and authentication solution.
- Combines:
  - LDAP (directory services)
  - Kerberos (authentication)
  - DNS (domain services)
  - Certificate Authority (for internal PKI)
- Manages users, groups, host-based access control, and policies.
- Ideal for centralized access management in self-hosted clouds.
-



Role-Based Access Control (RBAC)

- RBAC defines access by role, not user
- Examples: Admins, Developers, Auditors
- Scales well in enterprise environments
- Use IAM roles and policies in AWS or groups

- Lab: Define roles in, assign app access – EC2 instances

Multi-Factor Authentication (MFA)

- Adds extra layer of protection
- Enforce MFA using AWS IAM policies
- Compatible with Authy, Google Authenticator
- MFA with Keycloak through OTP configuration

- Labs:
- Configure MFA for IAM user
- Demonstrate MFA on the AWS CLI

RBAC and MFA with Keycloak & Vault

- Define roles like admin, developer, read-only.
- Assign roles to users or groups instead of individual permissions.
- Keycloak Integration:
  - Map FreeIPA groups to Keycloak roles.
  - Use realms for multi-tenant scenarios.
- MFA = something you know (password) + something you have (TOTP).
- Enable MFA using Google Authenticator, FreeOTP, or SMS/email.
- Vault manages secrets (API keys, passwords, TLS certs).
- Authenticate using Keycloak + JWT tokens or FreeIPA credentials.
-

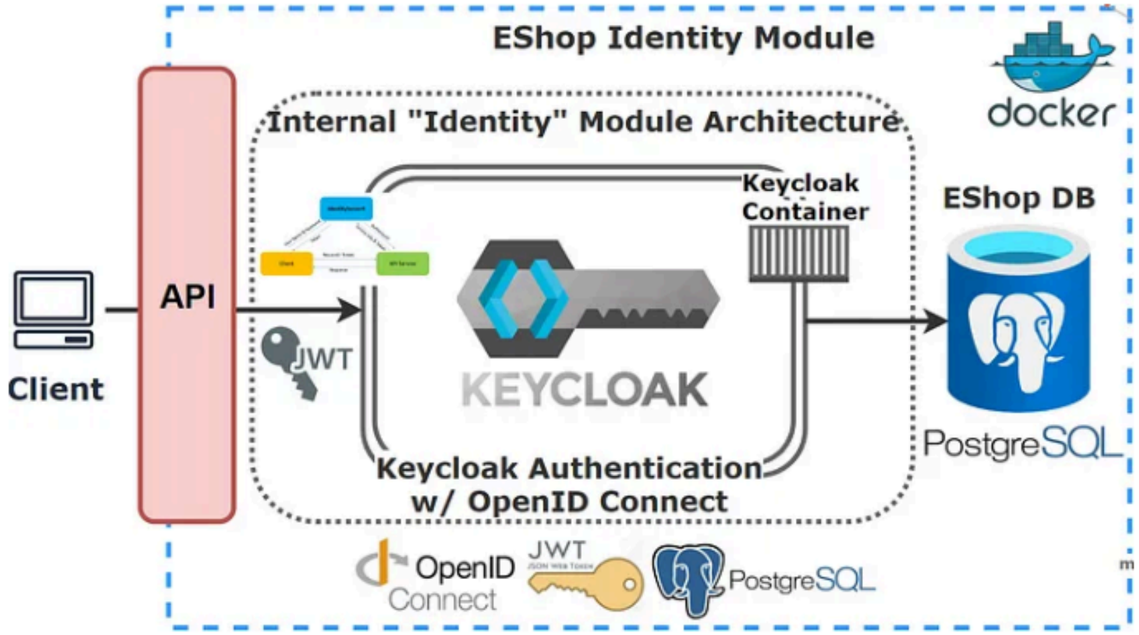
Vault for Secrets and MFA

- Vault stores API keys, credentials, tokens
- Supports dynamic credentials (e.g., per session AWS keys)
- Vault can integrate with IAM

- Lab:
  - Configure Vault for AWS dynamic credentials
-

Example: Keycloak-Based IAM with RBAC for eShop Architecture

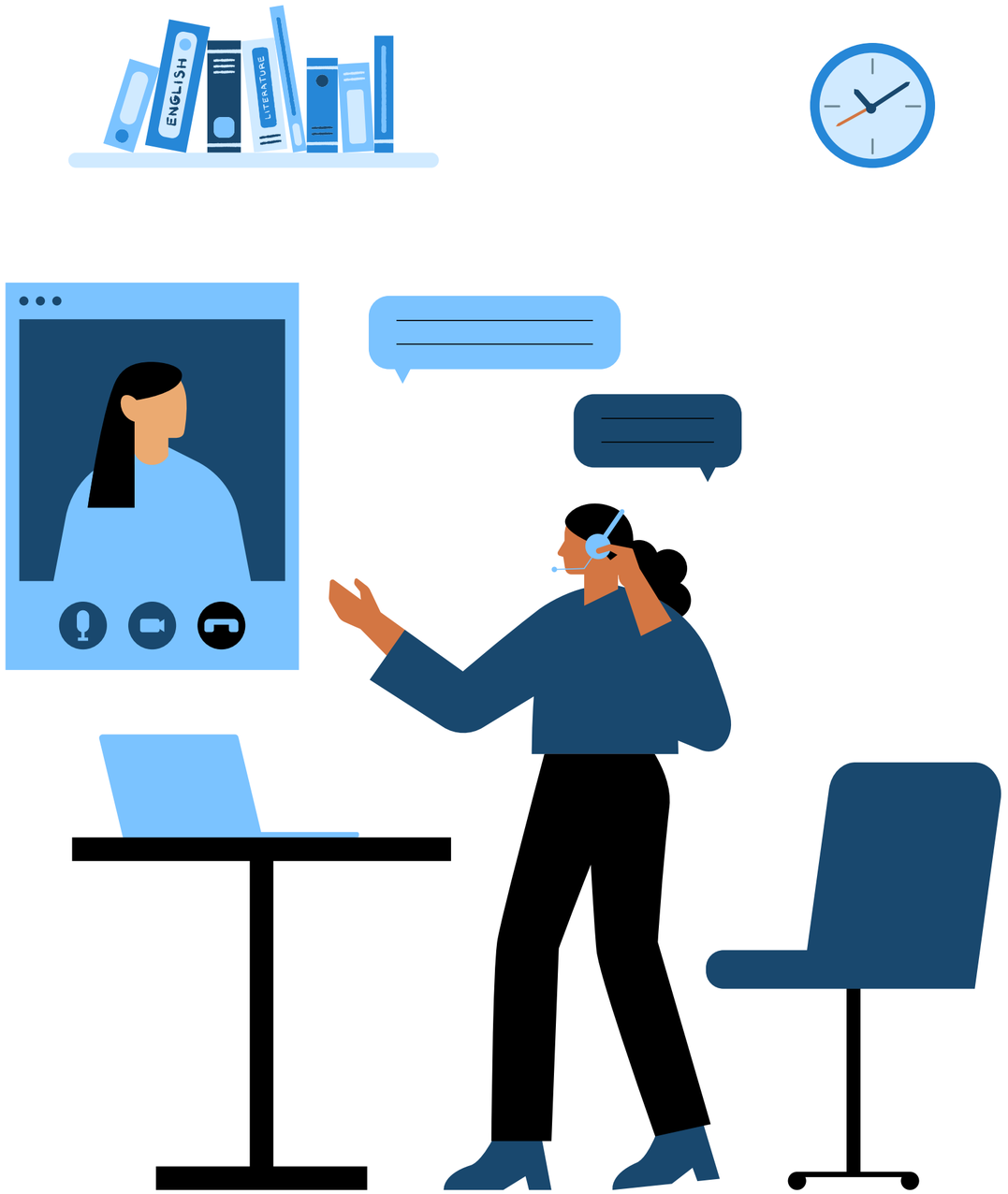
- Keycloak handles user authentication, SSO, and role-based access control (RBAC).
- Users authenticate via OpenID Connect, and receive a JWT token.
- JWTs are passed to the API to verify identity and enforce access based on roles.
- RBAC is centrally managed in Keycloak and enforced at the API level.
- User data, roles, and token sessions are stored in a PostgreSQL DB.
- Entire identity module runs containerized via Docker, ensuring portability and isolation.
- This architecture enables centralized, standards-based IAM for secure and scalable access control.
-



Keycloak Authentication in .NET Modular Monoliths Architecture

Secrets Management

- What is Secrets Management?
  - Managing access to:
    - Database credentials
    - API keys
    - TLS Certificates
    - SSH keys
  - Avoid secrets in code or config files.
  - Centralize control, enforce TTL, audit usage.
-
- Vault for Secrets Management
  - Static Secrets: Manually stored and retrieved.
  - Dynamic Secrets: Vault generates on demand (e.g., DB login valid for 15 mins).
  - Leases & Revocation:
    - Secrets expire or can be revoked.
  - Audit Logs:
    - Who accessed what and when.
-

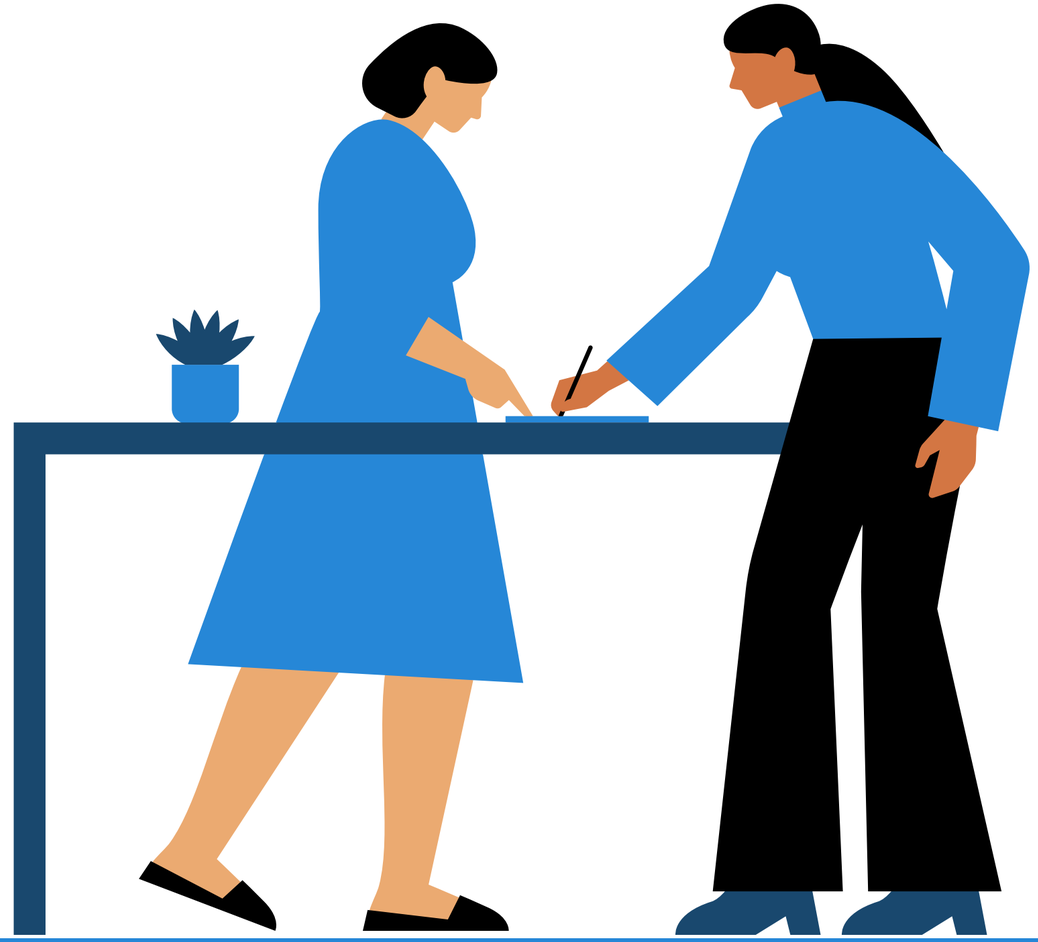


Secrets Management

- Store API keys, DB passwords securely
- AWS Secrets Manager supports rotation, IAM integration
- Vault allows policy-driven access to secrets

Compliance and Data Protection

- GDPR (EU): Personal data protection and privacy.
- HIPAA (US): Medical/healthcare information protection.
- ISO/IEC 27001: Security management best practices.
- NIST Framework: Risk-based information security practices.
- Self-hosted clouds must demonstrate auditing, data control, and access traceability.



- AWS offers support for GDPR, HIPAA, PCI-DSS, SOC 2, etc.
- Use AWS Artifact to access compliance reports
- Automate checks with AWS Config and Audit Manager

- Lab: Create AWS Config compliance rule

- Data Protection Strategies
  - Data Classification: Public, internal, confidential, restricted.
  - Encryption at Rest: Use LUKS for disks, Vault for file-level encryption.
  - Encryption in Transit: Enforce HTTPS/TLS across all endpoints.
  - Key Rotation: Regularly rotate encryption keys/secrets.
-

Shared Responsibility Model

- In public cloud: Provider(AWS) handles infra
- You handle data, identity, applications
- Visualize this with a diagram: "Security *of* vs *in* the cloud"
- In self-hosted cloud: YOU own both layers.
  - Network
  - Storage
  - Identity
  - Hypervisors
  - Physical infra
- You must design policies, backups, segmentation, and monitoring.

Lab: Explore S3/EC2 configs to show responsibility

- Implications for Self-Hosted
  - Firewalls, VLANs, isolated networks must be designed properly.
  - Automate IAM with FreeIPA and audit logs.
  - Maintain snapshots for disaster recovery.
  - Ensure physical security of servers (access control, power, CCTV).
-

Cloud Encryption Explained

- Types of Encryption
  - At Rest:
    - Full disk encryption (LUKS, dm-crypt)
    - S3 encryption
    - File-level encryption with Vault
  - In Transit:
    - TLS for APIs and web portals
    - VPN (IPSec with StrongSwan or OpenVPN)
  - In Use:
    - Still emerging (Confidential Computing)
    - Encrypt memory while being processed

- Encryption Keys Management
  - Use Vault to manage keys.
  - AWS KMS supports key creation, rotation, and audit logs
  - Client-side encryption: apps encrypt before sending to cloud
  - Setup access policies using Keycloak identity tokens.
  - Enable key versioning and revocation.
  - Use HSM integration or software backends for secure key storage.
    -
-

Lab: Enable S3 encryption with KMS

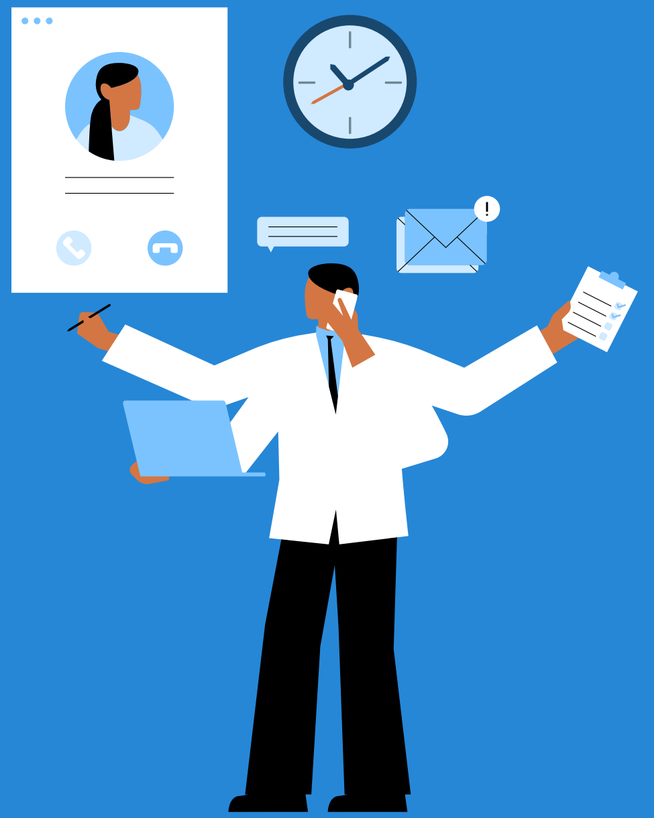
Encryption Methods Comparison

- Symmetric: single key (fast, good for bulk data)
- Asymmetric: key pair (public/private)
- AWS Tools: KMS (symmetric), CloudHSM (asymmetric), ACM (certificates)

- Lab: Encrypt/decrypt using AWS KMS CLI

Incident Response in Cloud

- Self-Hosted Incident Response Strategies
  - Logging & Monitoring:
    - Use ELK/Graylog/Prometheus-Grafana
  - Anomaly Detection:
    - Alert on failed logins, traffic spikes
  - Response Actions:
    - Quarantine compromised VM via MikroTik firewall
    - Snapshot before analysis
  - Post-Incident:
    - Review logs, rotate secrets, patch vulnerabilities
-



- Detect: CloudTrail, GuardDuty
- Respond: isolate using automation (Lambda)
- Recover: use backups, versioning, DR sites
- Prepare: Incident response plans, tabletop exercises
-

Secure Access Techniques

- Use SGs and NACLs to limit access
- Identity federation with SSO (SAML, OIDC)
- Just-in-time access with short-term IAM roles
- Move toward Zero Trust model

Summary

- IAM and MFA are foundational to cloud security
- Encryption and secrets handling must be enforced
- Compliance isn’t optional—it’s continuous
- You share responsibility with Cloud Providers—own your part

Thank You. (Q&A)