1 Introduction and IAM

Tuesday, 10 September 2019 9:56 PM

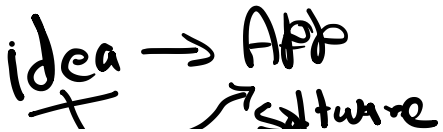
AWS SA Associate and Developer Associate
Coverage:

- Core AWS Services
- Well Architected Framework

Introduction to Cloud Computing and AWS
Amazon Elastic Compute Cloud and Amazon Elastic Block Store
Amazon Simple Storage Service and Amazon Glacier Storage
Amazon Virtual Private Cloud
Databases
AWS Identity and Access Management
CloudTrail, CloudWatch, and AWS Config
The Domain Name System and Network Routing: Amazon Route 53 and Amazon CloudFront
The Reliability Pillar
The Performance Efficiency Pillar
The Security Pillar
The Cost Optimization Pillar
The Operational Excellence Pillar

The Course is focused to offer:

Hands-on experience using the AWS services that provide compute, networking, storage, and databases
Ability to define a solution using architectural design principles based on customer requirements.
Ability to provide implementation guidance
Ability to identify which AWS services meet a given technical requirement An understanding of the five pillars of the Well-Architected Framework
An understanding of the AWS global infrastructure, including the network technologies used to connect them
An understanding of AWS security services and how they integrate with



traditional on-premises security infrastructure

Introduction to Cloud Computing

Journey from beginning of an idea to Cloud

On-demand Delivery of **IT resources (h/w, applications)** **over the network/internet** **with** **pay-as-you-go pricing model.**

Stop thinking of your infrastructure as hardware, and instead think of it (and use it) as software

H/W vs S/W
Speed
Ease
Cost

Cloud Computing Models

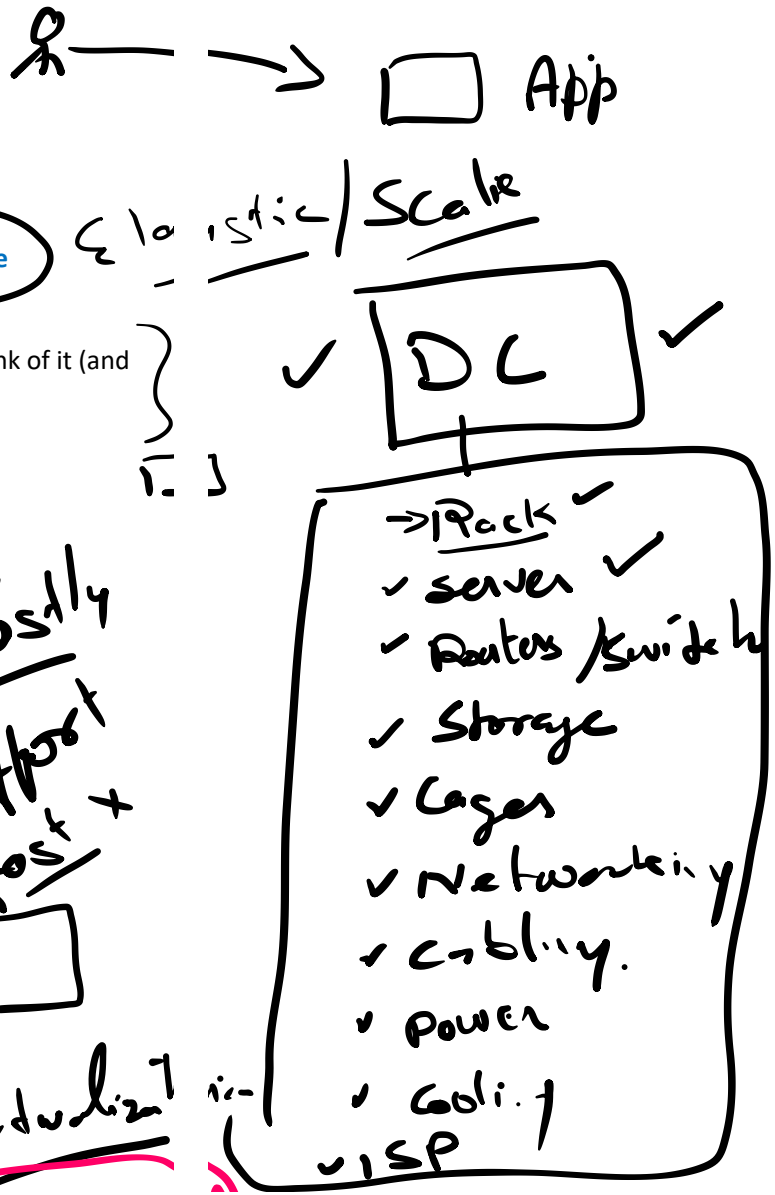
Cloud Deployment Models

What is Amazon Web Services?

Layers of services/infra in AWS

Benefits of AWS

Trade Capex vs for variable Opex
Benefit from massive economies of scale
Eliminate guessing on capacity needs
Increase speed and agility
Stop spending money on Data centers
Go global in minutes



AWS Global Infra

AWS Data Centers
AWS Availability Zone
AWS Regions
AWS Edge Locations

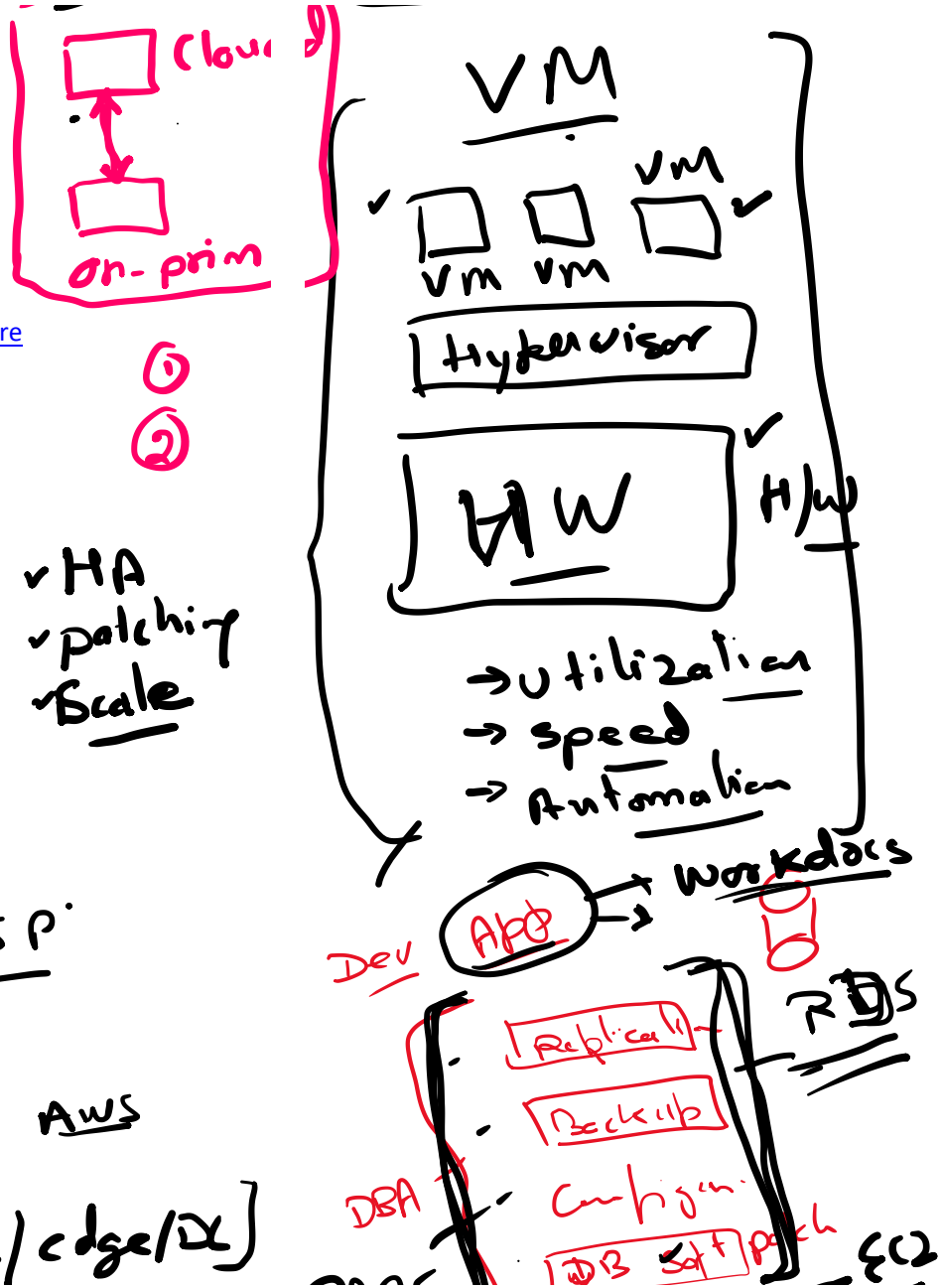
https://aws.amazon.com/about-aws/global-infrastructure

Managed Services/Serverless vs Unmanaged services

Quick intro of core AWS services

Security and compliance
https://aws.amazon.com/compliance/

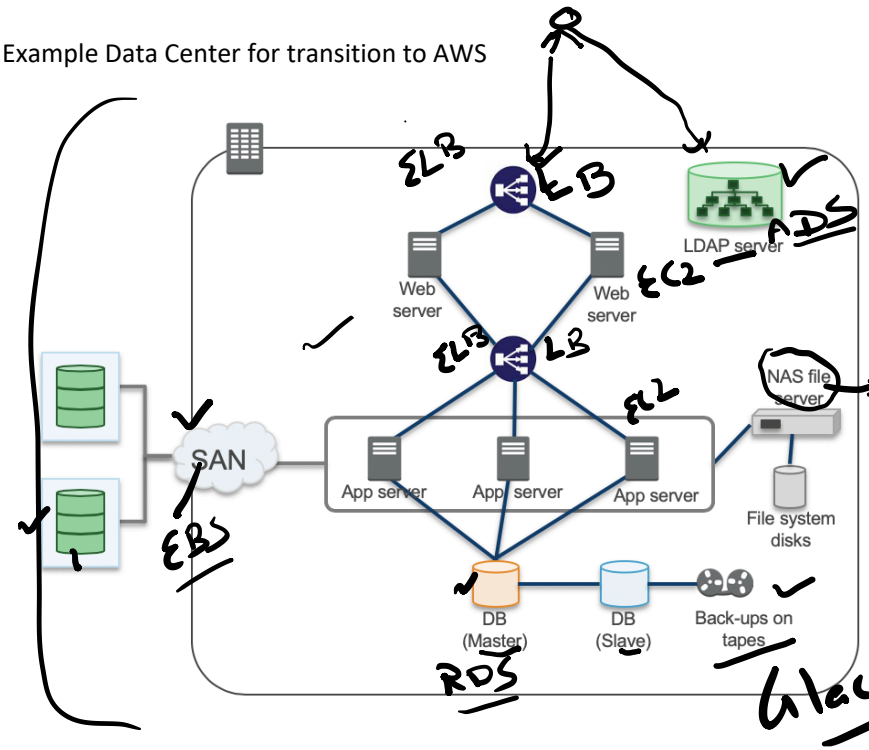
AWS Shared Responsibility Model
AWS vs Customer responsibility



AWS SLA's
https://aws.amazon.com/legal/service-level-agreements/

Hybrid vs All-in case studies

Example Data Center for transition to AWS



AWS Architectural Best Practices
- handle change in demand
- automate
- think resources as temporary
- design for loose coupling
- Managed services/serverless rather then servers
- Technology to workload
- Avoid SPOF
- Optimize for cost
- Cache

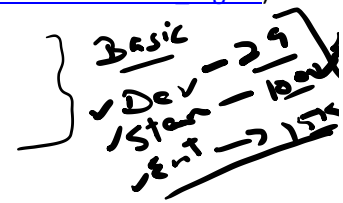
- Secure

Working with AWS

AWS CLI
AWS SDK
AWS Console
AWS REST

Endpoint?
(https://docs.aws.amazon.com/general/latest/gr/rande.html#ec2_region)

AWS Support plans
https://aws.amazon.com/premiumsupport/plans/

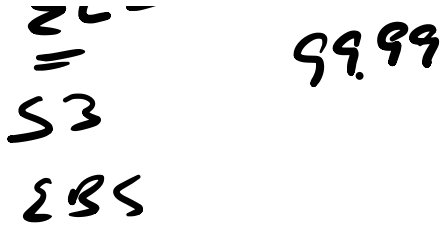


Demo and Exercises:
1. Create AWS Account and AWS Console Tour
https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/

Please read through the below links to understand billing and keeping the bill to minimum/0.
https://aws.amazon.com/premiumsupport/knowledge-center/what-is-free-tier/
https://aws.amazon.com/free/faqs/
https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/checklistforunwantedcharges.html

2. Setup AWS CLI and create an S3 bucket
https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html
https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html

3. Setup AWS SDK and create an S3 bucket
https://aws.amazon.com/developers/getting-started/python/
https://aws.amazon.com/developers/getting-started/java/



First things after creating an AWS account:

- Understand free tier
- Create billing alarm
- Never share or openly store secret keys/pwd

Ex: https://aws.amazon.com/getting-started/tutorials/control-your-costs-free-tier-budgets/?trk=gs_card

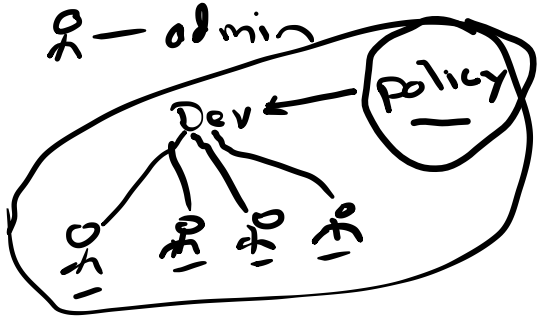
https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/tracking-free-tier-usage.html

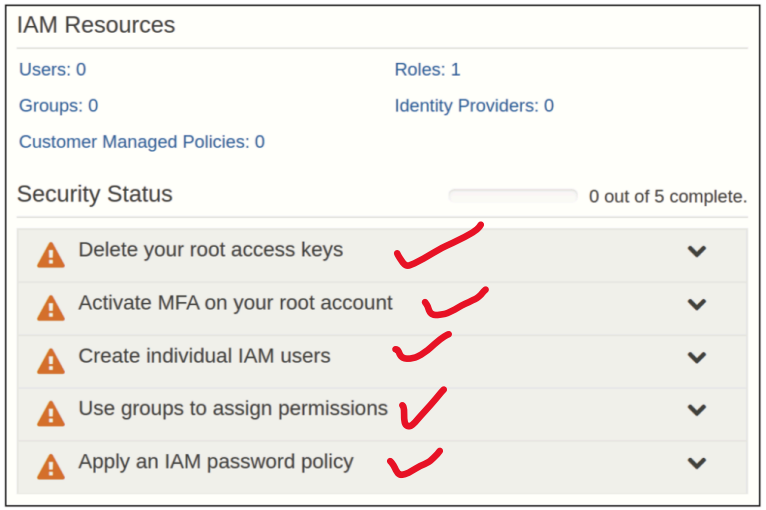
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/monitor_estimated_charges_with_cloudwatch.html

IAM

- Centrally manage access and authentication of your users to your AWS resources.
- Offered as a feature of your AWS account for no charge.

- Create users, groups, and roles, and apply policies to them to control their access to AWS resources.
- Manage what resources can be accessed and how they can be accessed (e.g., terminating EC2 instances).
- Define required credentials based on context (e.g., who is accessing which service and what are they trying to do?).





Dev

Ops

Auditor

Admin

Security

Least privilege principle

IAM

S3.action

Types Of Security Credentials

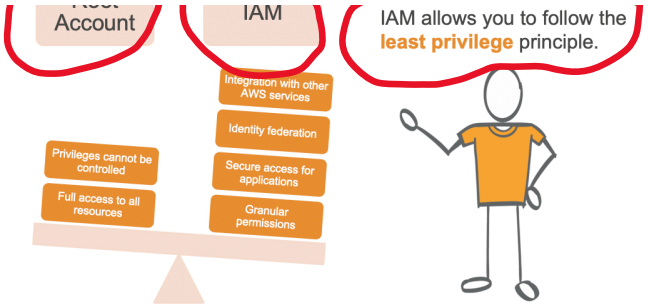
Email address and password	Associated with your AWS account (root)
IAM user name and password	Used for accessing the AWS Management Console
Access keys	Typically used with CLI and programmatic requests like APIs and SDKs
Multi-Factor Authentication	Extra layer of security Can be enabled for root account and IAM users
Key pairs	Used only for specific AWS services like Amazon EC2

P1 Allows S3

D)

Root Account Access vs. IAM Access

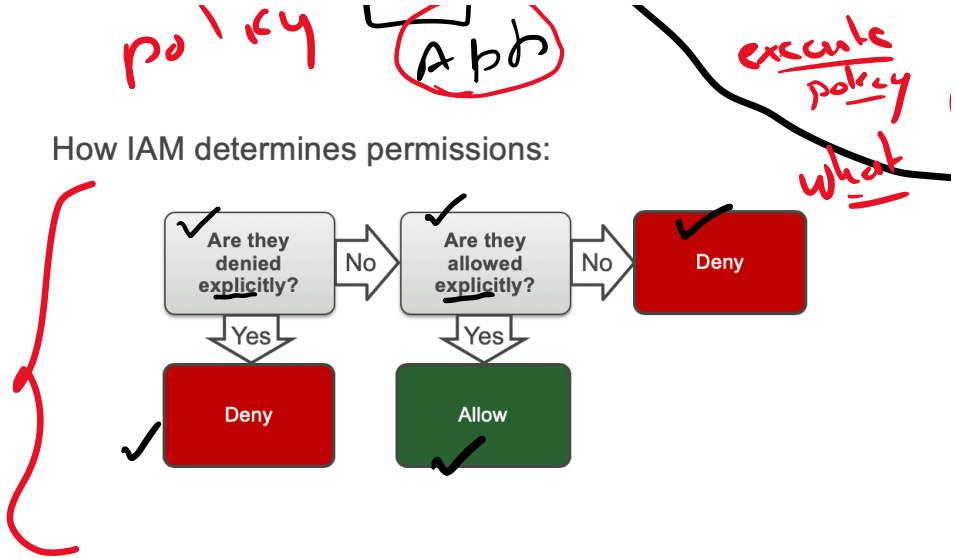
Deny S3



IAM Permissions

- Permissions determine which resources and which operations are allowed to be used.
- All permissions are *implicitly* denied by default.
- If something is *explicitly* denied, it can never be allowed.

Best Practice: Follow the **least privilege** principle.

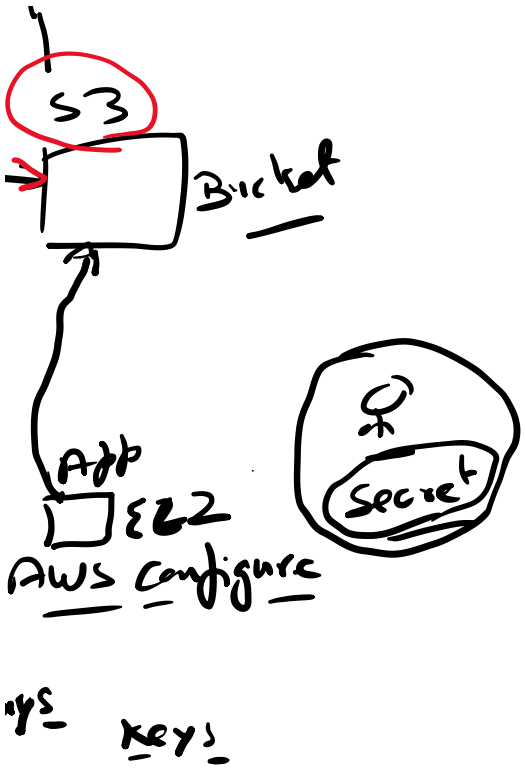


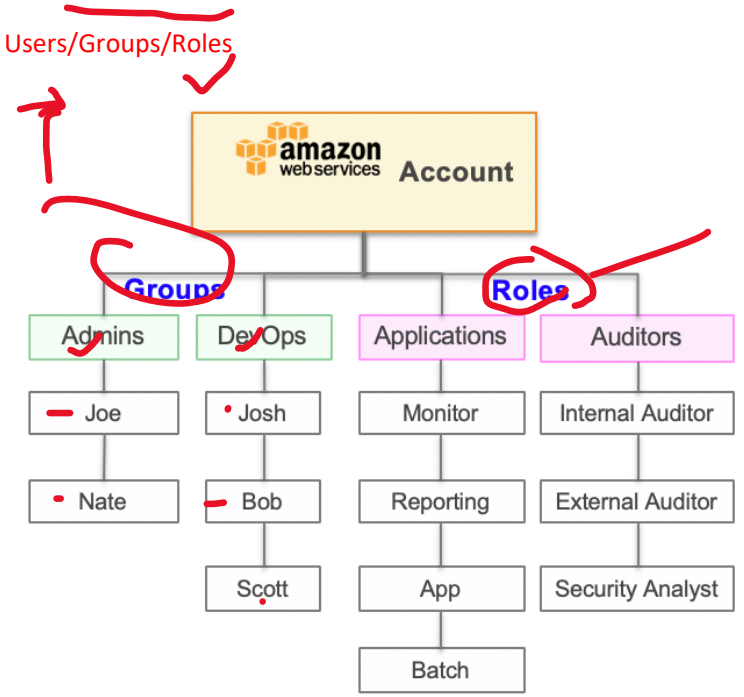
IAM Policies

An IAM policy is a formal statement of one or more permissions.

- You attach a policy to any IAM entity: user, group, or role.
- Policies authorize the actions that may, or may not, be performed by the entity.
  - Enables fine-grained access control.
- A single policy can be attached to multiple entities.
- A single entity can have multiple policies attached to it.

Best practice: When attaching the same policy to multiple IAM users, put the users in a group and attach the policy to the group instead.


S3
Bucket
App
EC2
Secret
AWS configure
keys


Users/Groups/Roles
amazon web services
Account
Groups
Roles
Admins
DevOps
Applications
Auditors
Joe
Josh
Monitor
Internal Auditor
Nate
Bob
Reporting
External Auditor
Scott
App
Security Analyst
Batch

IAM

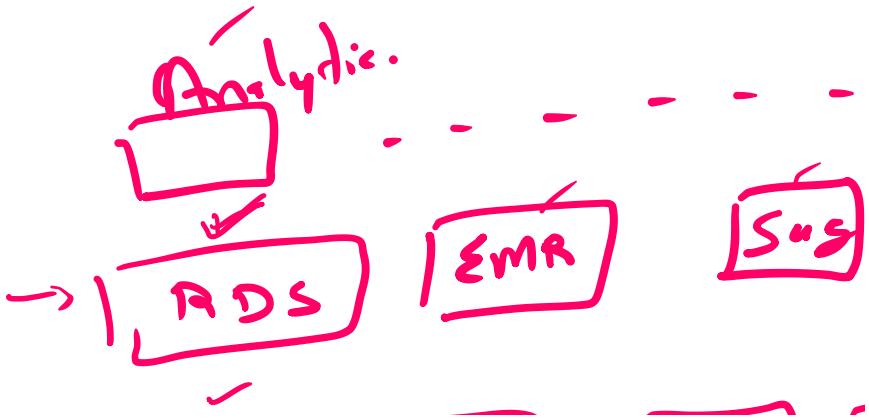
CSP ← PAAS
SAAS.

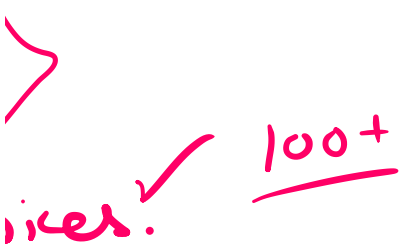
huge set of serv

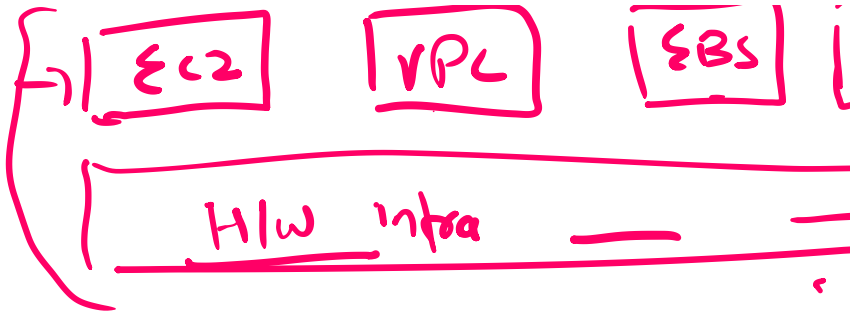
App needs

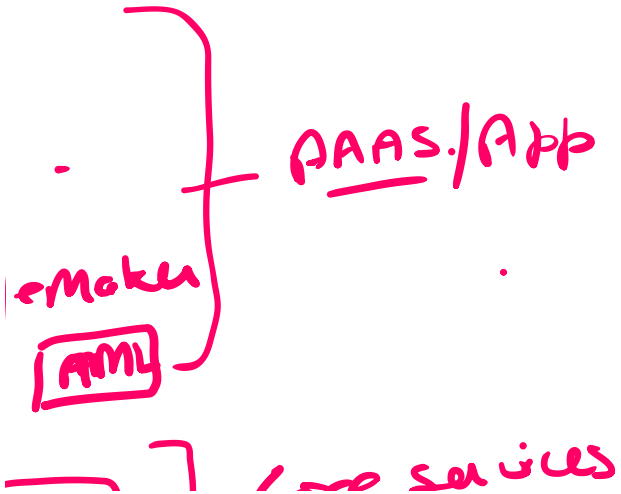
WW presence

Scale









Roles

Use cases:

- Provide AWS resources with access to AWS services.
- Provide access to externally authenticated users.
- Provide access to third parties.
- Switch roles to access resources in:
  - Your AWS account.
  - Any other AWS account (cross-account access).

First things to do with a new AWS account?

1. Stop using the root account as soon as possible.

   http://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_create-admin-group.html

2. Require multi-factor authentication for access.

   https://aws.amazon.com/iam/details/mfa/smsmfa/

3. Enable AWS CloudTrail.

   http://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-create-a-trail-using-the-console-first-time.html.

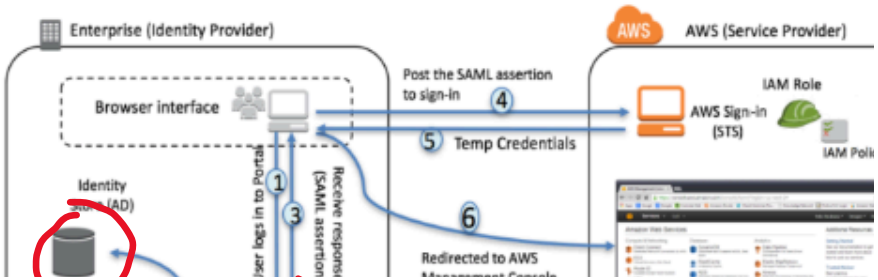
4. Enable a billing report, such as the AWS Cost and Usage Report

   http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/detailed-billing-reports.html#turnonreports.

Identity Federation

Authenticate users to your own identity store:

- You write an “identity broker application.”
- Users authenticate to your identity broker.
- Your identity broker provisions temporary credentials via STS.
- **Single Sign-On (SSO):** Temporary credentials can be used to sign user directly into the AWS Management Console.



1. Corporate user accesses the corporate Active Directory Federation Services portal s page and provides Active Directory authentication credentials.
2. AD FS authenticates the user against Active Directory.
3. Active Directory returns the user's information, including AD group membership information.
4. AD FS dynamically builds ARNs by using Active Directory group memberships for th roles and user attributes for the AWS account IDs, and sends a signed assertion to t browser with a redirect to post the assertion to AWS STS.
5. Temporary credentials are returned using STS AssumeRoleWithSAML.
6. The user is authenticated and provided access to the AWS management console.

1. User browses to a URL: A user in your organization browses to an internal portal in your network. The portal also functions as the IdP that handles the SAML trust between your organization and AWS.
2. User authenticated: The identity provider (IdP) authenticates the user's identity against AD.
3. Receives authentication response: The client receives a SAML assertion (in the form of authentication response) from the IdP.
4. Post to sign-in passing AuthN: The client posts the SAML assertion to the new AWS sign-in endpoint. Behind the scenes, sign-in uses the AssumeRoleWithSAML API to request temporary security credentials and construct a sign-in URL.
5. Redirect client to AWS Management Console: The user's browser receives the sign-in URL and is redirected to the AWS Management Console.

   From the user's perspective, the process happens transparently. The user starts at your organization's internal portal and ends up at the AWS Management Console without ever having to supply any AWS credentials.

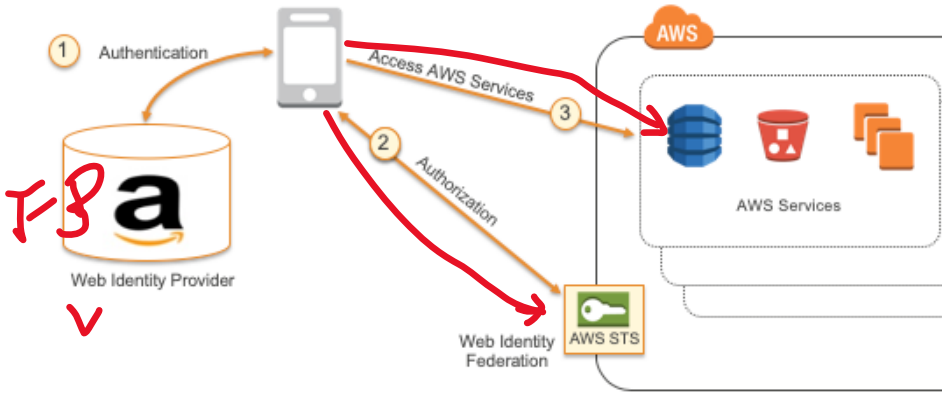
   The basic steps:
6. Generate a metadata document using your IdP software.

ign-in

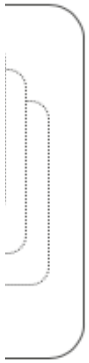
ne IAM
the users

7. Create a SAML provider using that metadata document.
8. Create one or more IAM roles that establish trust with the SAML provider and define permissions for the federated user.
9. Configure your IdP software to map attributes (such as AD groups) to the IAM roles.
10. Finished.

https://aws.amazon.com/blogs/security/enabling-federation-to-aws-using-windows-active-directory-adfs-and-saml-2-0/
https://aws.amazon.com/blogs/security/aws-federated-authentication-with-active-directory-federation-services-ad-fs/



11. The user needs to be authenticated. For example, using the *Login with Amazon* SDK, the app authenticates the user and receives a token from Amazon.
12. The user needs to be authorized to access resources in his/her AWS account. The app makes an unsigned **AssumeRoleWithWebIdentity** request to STS, passing the token from the previous step. STS verifies the authenticity of the token; if the token is valid, STS returns a set of temporary security credentials to the app. By default, the credentials can be used for one hour.
13. The app uses the temporary security crede ntials to make signed requests for resources in your Amazon S3 bucket (as an example). Because the role's access policy used variables that reference the app ID and the user ID, the temporary security credentials are scoped to that end user and will prevent him/her from accessing objects owned by other users.



Ex: **Lock down the root user**

1. If necessary, create a regular user and then assign it the AdministratorAccess policy.
2. Make sure there are no active access keys associated with your root account.
3. Enable MFA for the root account, where short-lived authentication codes are sent to software applications on preset mobile devices (including smartphones) to confirm a user's identity.
4. Update your root login to a password that's long, complex, and includes nonalphanu- meric characters.
5. Confirm that you can still log in as root and then store the password safely.

Ex: **Assign and Implement an IAM policy**

1. Create a new user in the IAM Dashboard.
2. Attach the AmazonS3FullAccess policy that will permit your user to create, edit, and delete S3 buckets. (Hint: you can search IAM policies

using s3 to display a much shorter list.)
3. Note the user login instructions that will be displayed.
4. Log in as your new user and try creating a new S3 bucket.
5. Just to prove everything is working, try launching an EC2 instance. Your request should be denied.

Ex: Create, use, and delete an AWS Access Key

1. Create a new AWS access key, and save both the access key ID and secret access key somewhere secure.
2. Use aws configure from your local command line to add the key to your AWS CLI configuration.
   If you already have a different access key configured on your system, you can create and use multiple keys in parallel. By adding the --profile argument to the aws configure command, you can create separate profiles. You'll be prompted to enter configuration details for each new profile. Here's an example:

   $ aws configure --profile account2
   You can then invoke a profile by adding the argument to a regular command.
   $ aws s3 ls —-profile account2

3. Try performing some operation—such as listing your S3 buckets—and then uploading a local file using the AWS CLI and your new key.
4. Disable (select Make Inactive) or delete the key you just created from the IAM Dashboard.
5. Confirm that you are now unable to administer your S3 buckets using the key.

More about Security later…

IAM SUMMARY

Understand how to work with IAM policies.
Understand how to protect your AWS account's root user.

Understand how to effectively and securely manage user access.
Understand how to optimize account access security.
Be familiar with the various AWS authentication and integration tools.