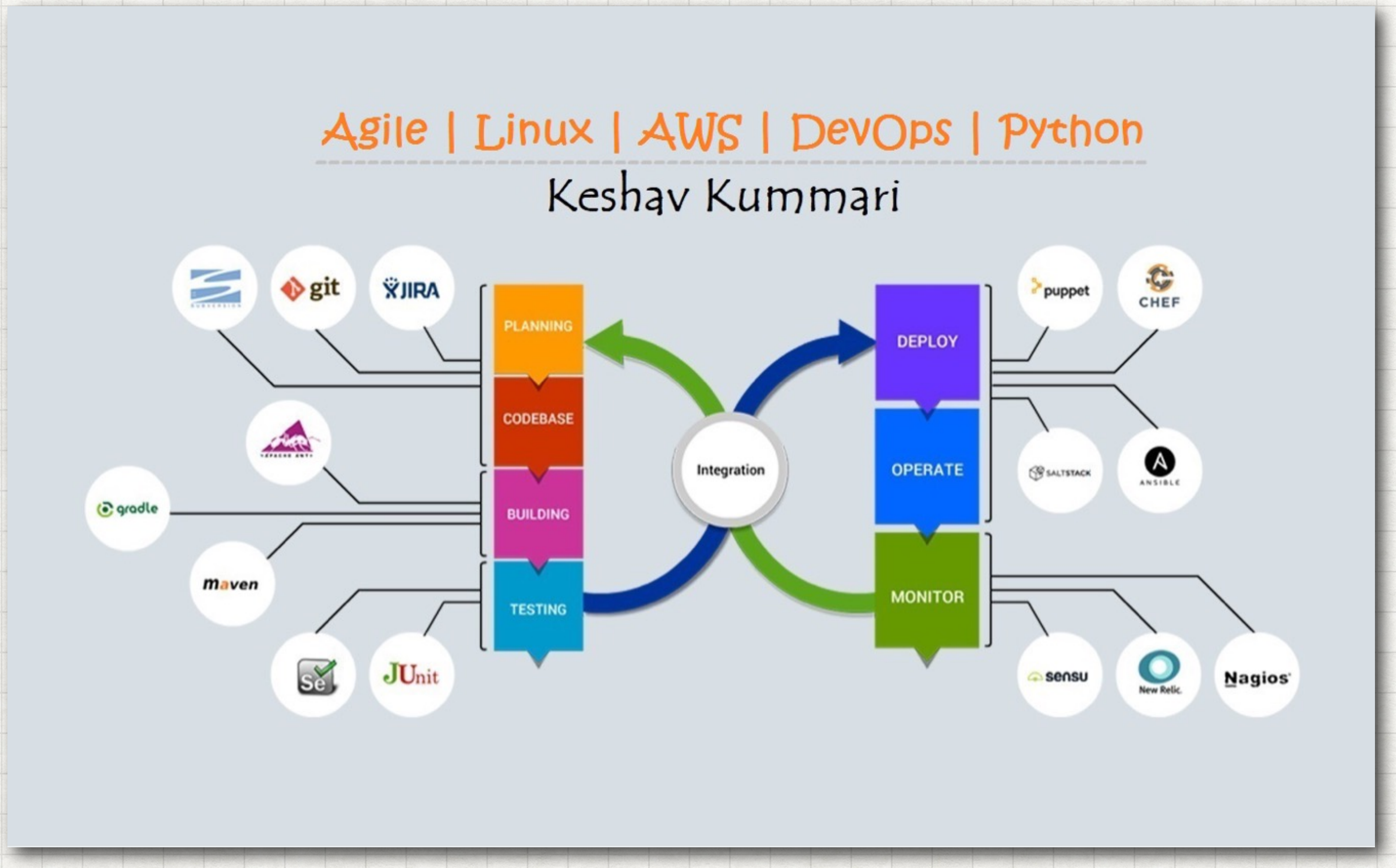
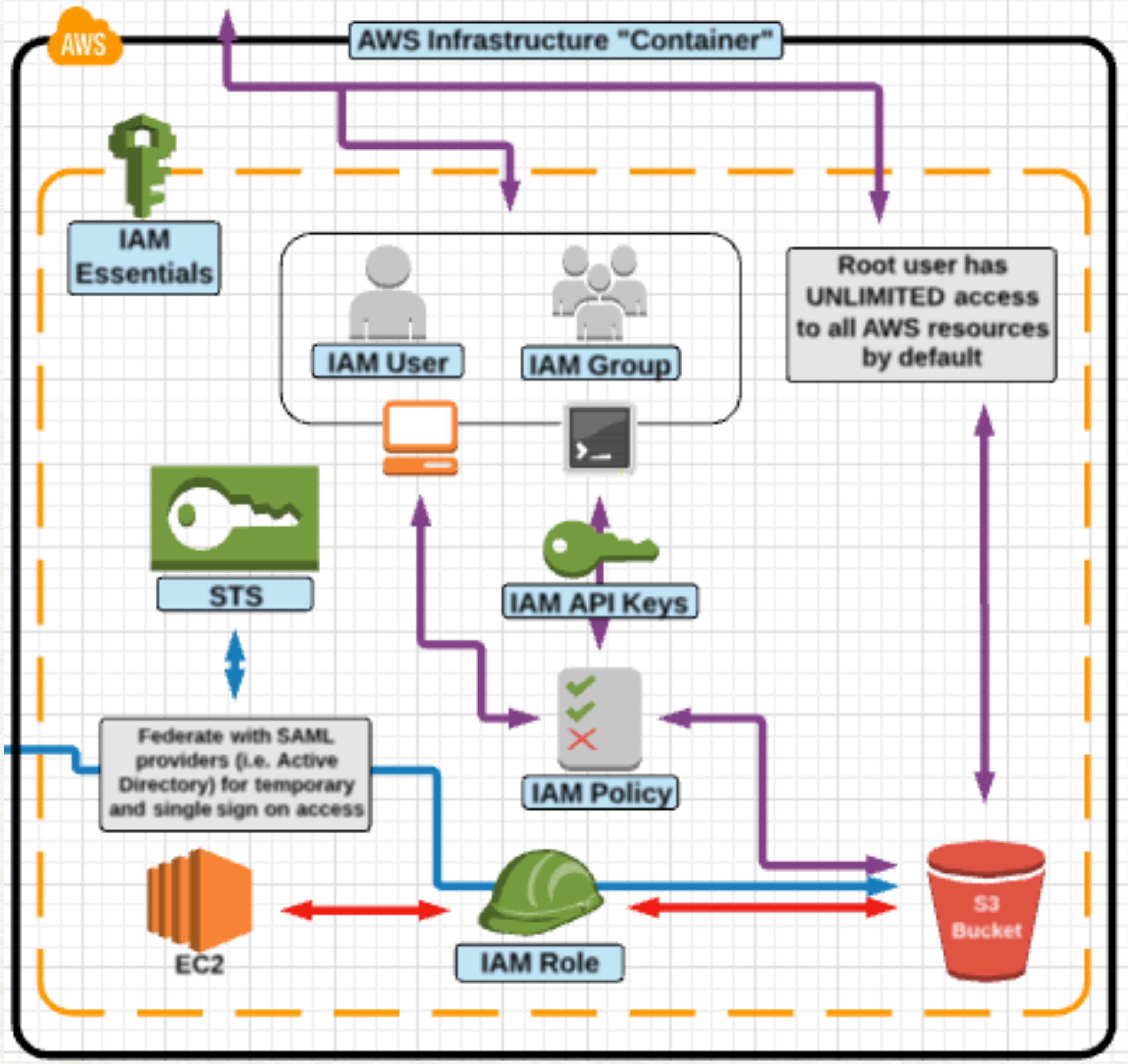
AWS

KESHAV KUMMARI



AWS - IAM


AWS
AWS Infrastructure "Container"
IAM Essentials
IAM User
IAM Group
Root user has UNLIMITED access to all AWS resources by default
STS
IAM API Keys
Federate with SAML providers (i.e. Active Directory) for temporary and single sign on access
IAM Policy
EC2
IAM Role
S3 Bucket

AWS - IAM

AWS - IAM

- IAM Essentials :
- IAM is where you manage your AWS USERS,GROUPS, and ROLES and their access to AWS accounts and services.
  - IAM Provides access and access permissions to AWS resources(such as EC2, S3, & DynamoDB)
  - IAM is global to all AWS regions, creating a user account will apply to all the regions
- The common use of IAM is to manage:
- Users
- Groups
- Roles
- IAM Access Policies
- API Keys
- Specify a password Policy as well as manage MFA requirements on a per user basis

- By default, any new IAM user you create in an AWS account is created with NO Access to any AWS services.

- This is a non-explicit deny rule set on all new IAM users.

- For all users(Besides the root user), permissions must be given that grant access to AWS services(This is done through IAM Policies)

- When a new AWS root account is created, it is "Best Practice" to complete the tasks listed in IAM under "Security Status".
- **Which include:**

  Delete your root access keys

  Activate MFA on your root account

  Create individual IAM users

  User groups to assign permissions

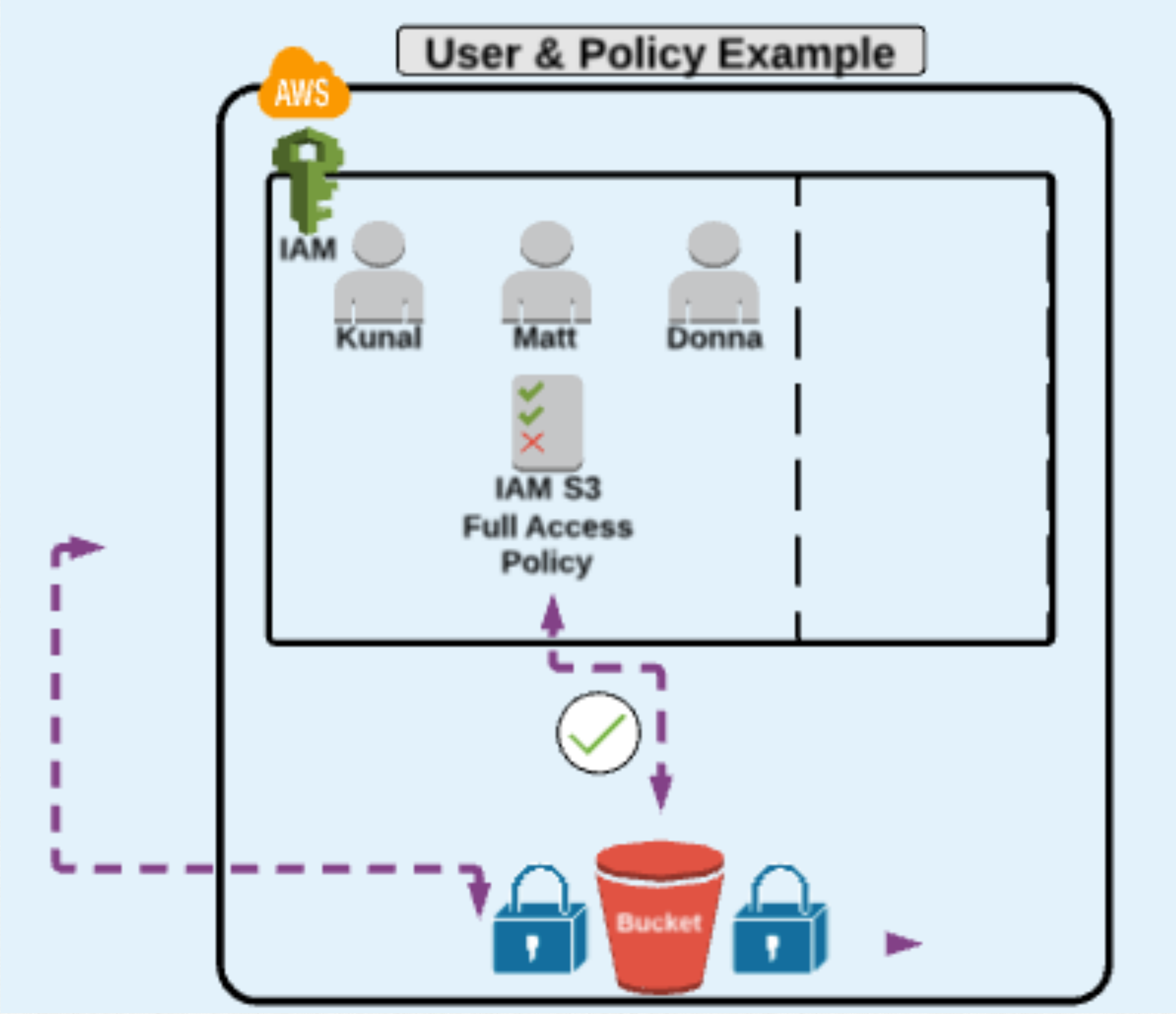
  Apply an IAM password policy

- Best Practice is to login and do daily work as an IAM user - NOT as root user.
- It is also best to always best to practice the "Principal of Least Privilege" when administering AWS accounts, users, groups, and roles.

IAM - USERS

- When first created, by default an IAM user has a non-explicit "deny" for all AWS services
- And does NOT have access to use them until a policy granting allow access has been applied to the user or to the group the user belogns to.
- IAM users receive unique access credentials so you do not(and should not) share with others.
- User credentials should NEVER be stored or passed to an EC2 instance.
- Users can have group and regular user policies apply to them - meaning a user can have multiple IAM policies applied to them at any given time.
- By default, an explicit deny always overrides an explicit allow from attached IAM policies.
- MFA can be configured on a per user basis for login and resource access/actions.

AWS IAM - USER



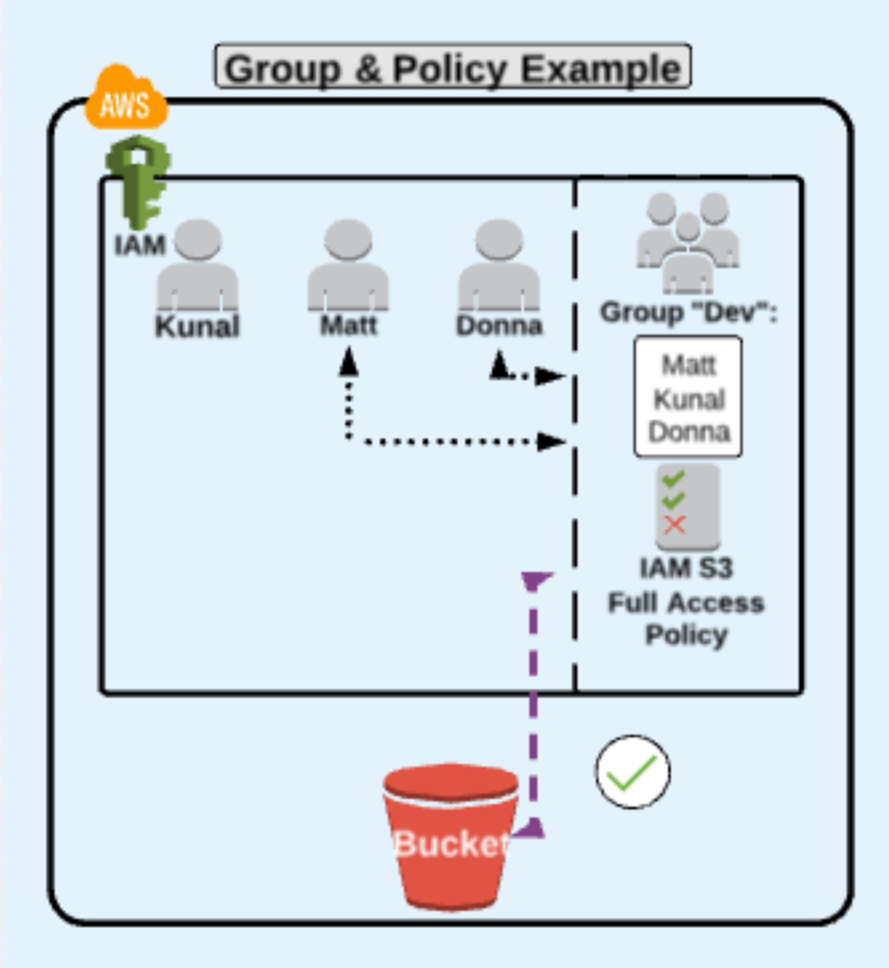
AWS - IAM User

IAM - GROUPS

- Allow you to assign IAM permission policies to more than one user at a time.

- This ability allows for easier access management to AWS resources.

-

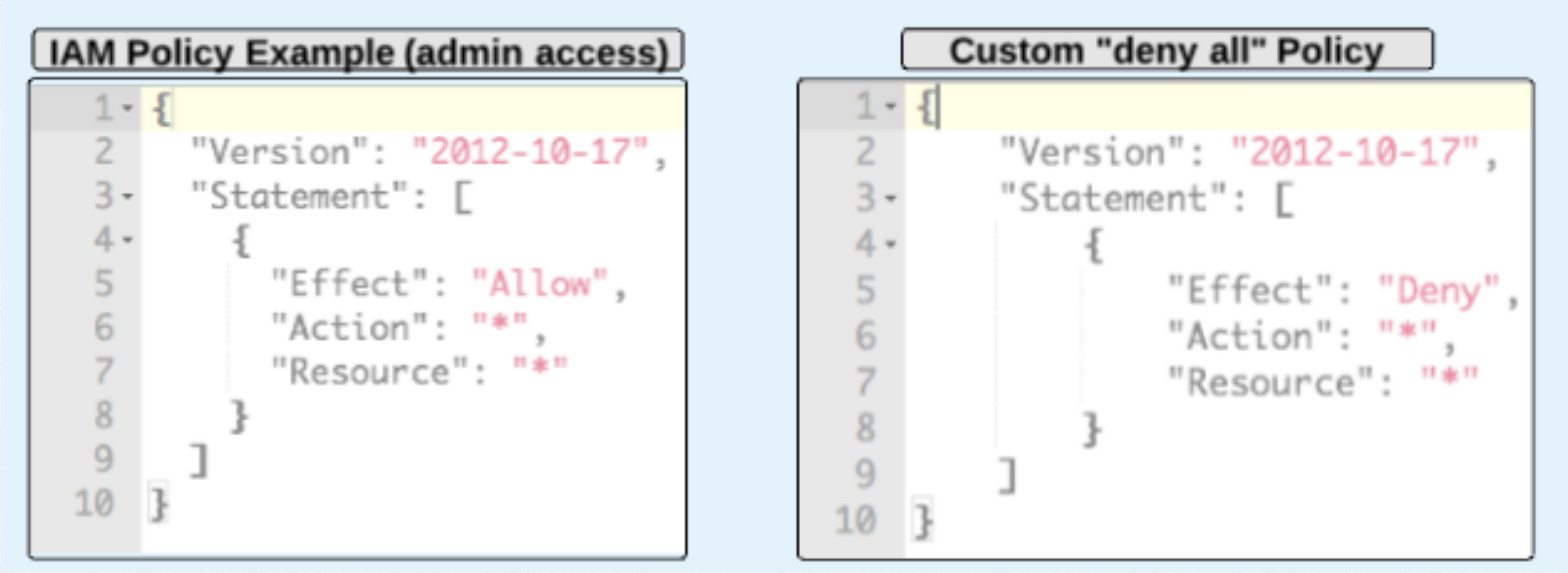


IAM - GROUPS

IAM - POLICY

- A policy is a document that formally states one or more permissions.
- By default, an explicit deny always overrides and an explicit allow.
- This allows for the use of a "deny all" policy to quickly restrict ALL access that a user may have through multiple attached policies.
- IAM provides pre-built policy templates to assign to users and groups, examples include:
- Administrator Access : Fully access to ALL AWS resources.
- Power User Access : Admin access except it does not allow user/group management
- Read only Access : Only view AWS resources(i.e. User can only view what is in an S3 bucket)
- You can also create custom IAM permission policies using the POLICY GENERATOR or written from scratch.
- More than one Policy can be attached to a user or group at the same time.
- Policies can not be directly attached to AWS resources(Such as an EC2 instance)

AWS - IAM POLICY DOCUMENT



IAM Policy Example (admin access)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

Custom "deny all" Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

IAM - POLICY DOCUMENT

AWS - IAM - ROLE

- A role is something that another entity can "assume", and in doing so acquires the specific permissions defined by the role.

- In the context of this course, "Entities" that can assume a role include AWS resources(Such as an EC2 instance) OR a non-AWS account holder who may need temporary access to an AWS resource(Through a service like Active Directory).

- Roles must be used because policies can not be directly attached to AWS services.

- For Example : If you are using an EC2 instance and it need to access an S3 bucket:

- Instance should assume a role from IAM with the proper required permissions.(S3 read-only)

- Instance can then perform actions based on the role it assumes.(Read from S3)

- You can but should never pass or store credentials in or to an EC2 instance - so roles are used instead

- Until recently, you could only assign a role to an EC2 instance during the EC2 instance's creation process.

- However, you can now assign/change a role that is assigned to an EC2 instance after the creation process via the CLI or the EC2 management console.

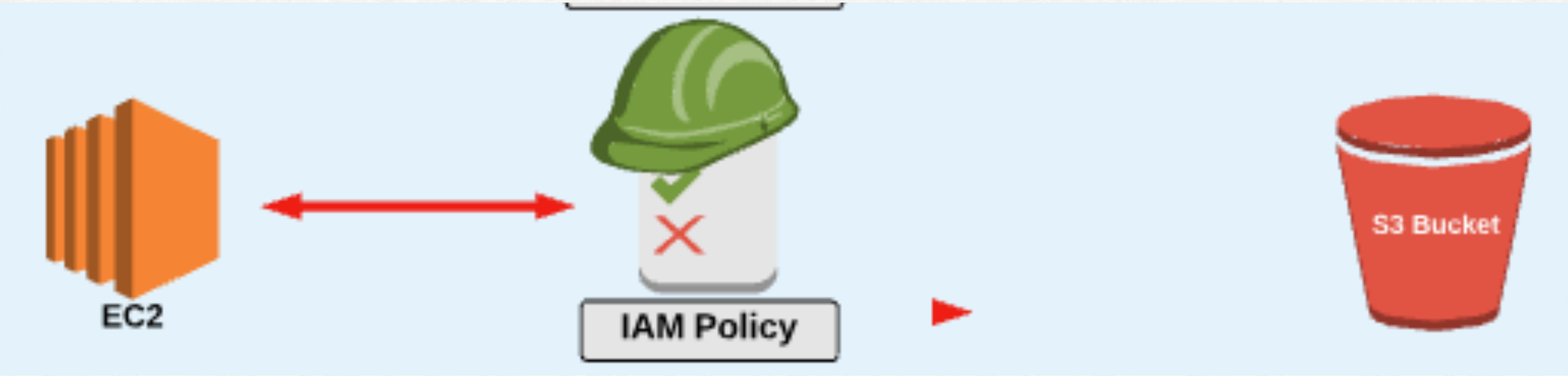
- An EC2 instance can only have ONE role attached at a time.

- **Other uses of rules:**

- Other users can assume a "role" for temporary access to AWS accounts and resources through having something like Active Directory or Single Sign-on service(i.e. Facebook, Google) assume an "Identity Provider Access" role.

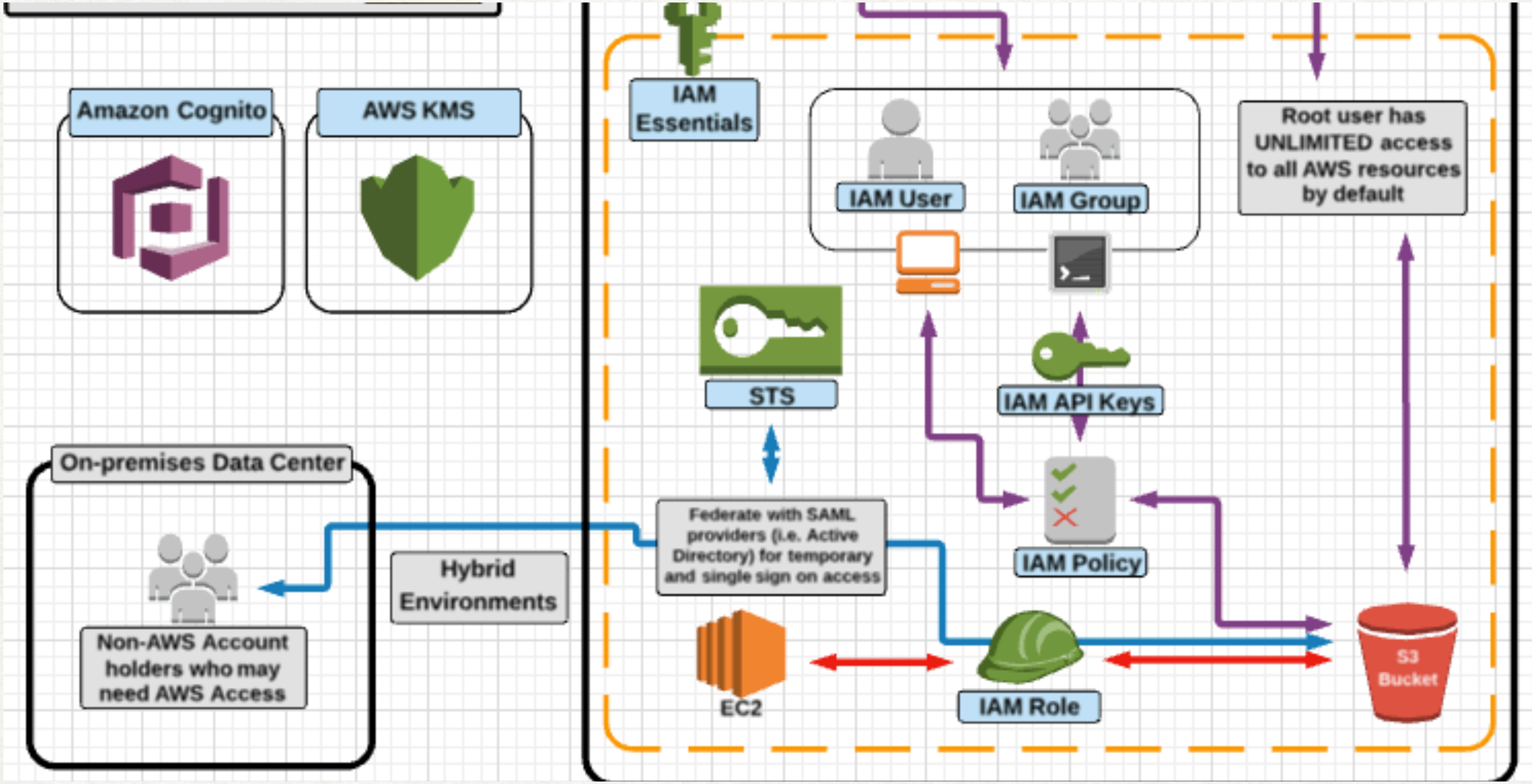
- Create "Cross Account" access where a use from one account can assume a role with permissions in another account.

AWS - ROLE



ROLE

AWS - IAM - STS


Amazon Cognito
AWS KMS
On-premises Data Center
Non-AWS Account holders who may need AWS Access
Hybrid Environments
IAM Essentials
IAM User
IAM Group
Root user has UNLIMITED access to all AWS resources by default
STS
IAM API Keys
Federate with SAML providers (i.e. Active Directory) for temporary and single sign on access
IAM Policy
EC2
IAM Role
S3 Bucket

STS

AWS - IAM - STS

- STS - Security Token Service
- STS allows you to create temporary security credentials that grant trusted users access to your AWS resources.
- These temporary credentials are for short-term use, and can be active for a few minutes to several hours.
- Once expired, they can no longer be used to access your AWS resources.
- When requested through an STS API call, credentials are returned with three components:
- Security Token
- An Access Key ID
- A Secret Access Key

STS BENEFITS

- No distributing or embedding long-term AWS security credentials in an application.
- Grant access to AWS resources without having to create an IAM identity for them.
- The basis for IAM roles and identity federation.
- Since the credentials are temporary, you do not have to rotate or revoke them.
- You decide how long they are active for.

WHEN TO USE STS

- Identity Federation:
- Enterprise identity federation(Authenticate through your companies network)
- STS supports Security Assertion Markup Language(SAML), which allows for use of Microsoft Active Directory(Of your own solutions).
- Web identity federation(3rd Party identity providers, i.e. Facebook, Google, Amazon)
- Roles for Cross-Account Access:
- Used for organisations that have more than one AWS account.
- Roles for Amazon EC2(and other AWS services)
- Grant access an to application running on an EC2 instance to access other AWS services without having to imbed credentials

STS API CALLS

- AssumeRole : Cross-Account delegation and Federation through a Custom Identity Broker
- AssumeRoleWithWebIdentity : Federation through a Web-based identity Provider
- AssumeRoleWithSAML : Federation through an Enterprise Identity Provider Compatible with SAML 2.0
- GetFederationToken : Federation through a custom identity broker
- GetSessionToken : Temporary Credentials for users in untrusted environments