Question 1: Design a golden master AMI build work-flow identifying the key stages, and suggest technologies which might be used to implement. Use of AWS native services is preferred.

Question 2: Currently connectivity between AWS accounts and on premise networks are provided via peering and site-to-site vpn’s.

Design a network topology that will allow scalable connectivity both across accounts and to on premise networks, and discuss:  
- Network migration strategies.  
- Security considerations and provisions.

Question 3: Design an enterprise IAM solution to provide access to users, Applications, and services across the organization while ensuring controls are in place to ensure restrictions can be implemented on a least privileged requirement.

Golden Master AMI

AMI: amazon machine image

Golden master AMI : create custom or user defined AMI

build work-flow

A **golden AMI** is an **AMI** that contains the latest security patches, software, configuration, and software agents that you need to install for logging, security maintenance, and performance monitoring

A **golden image** is simply an **image** that you have customized to your liking with all necessary software/data/configuration information ready to go and then saved as a personal AMI from which you can launch instances.

There is no difference. A golden image is simply an image that you have customized to your liking with all necessary software/data/configuration information ready to go and then saved as a personal AMI from which you can launch instances.

Performance ranges among regions and zones, usually full provision (for initiating instance/provision with puppet/**ami** finish) takes about 15 minutes (where **AMI** itself takes about 12 minutes)

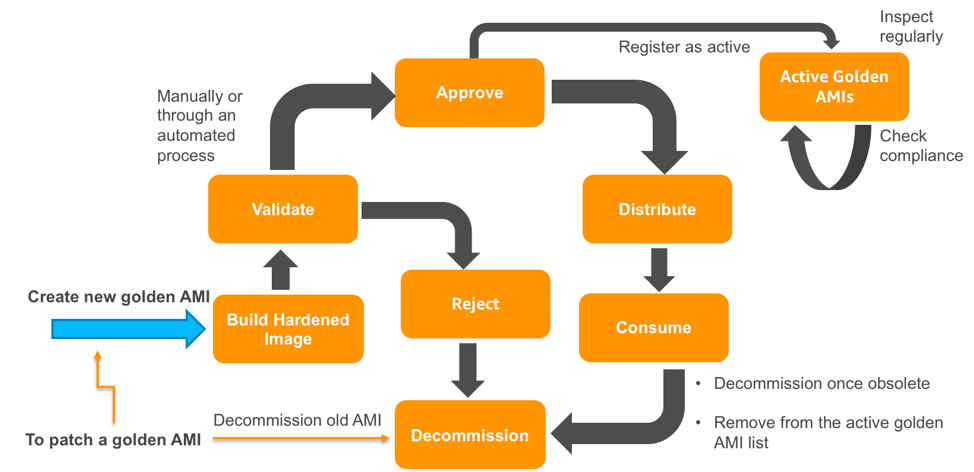
**Create an AMI from an Amazon EC2 Instance**

1. Right-click the instance you want to use as the basis for your AMI, and choose Create Image from the context menu. Create Image context menu.
2. In the Create Image dialog box, type a unique name and description, and then choose Create Image. Create Image dialog box.

<https://github.com/aws-samples/aws-golden-ami-pipeline-sample>

### About the golden AMI pipeline

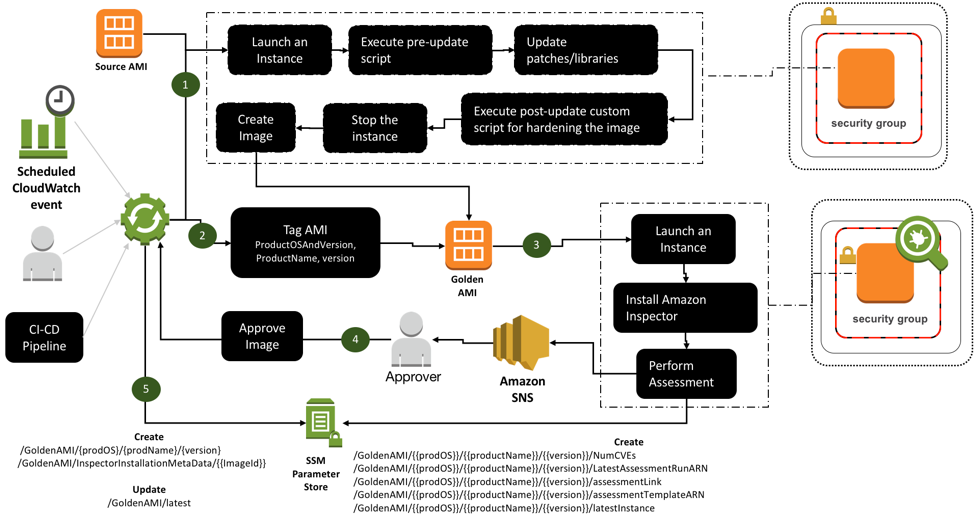
The golden AMI pipeline enables creation, distribution, verification, launch-compliance, and decommissioning of the golden AMI out of the box. The following diagram highlights the high-level workflow.



Once you create a golden AMI for a product (a product can be a standardized OS-AMI that you want to distribute to accounts in your organization or an application specific AMI you want to let your business unit(s) deploy in their environment), you can validate whether the AMI meets your expectations, and choose to approve or reject the AMI. If you reject a golden AMI, the golden AMI pipeline provides you an AWS Systems manager automation you can execute to decommission the golden AMI version completely. If you choose to approve the AMI as a golden AMI, it gets registered as active and is regularly inspected by the continuous vulnerability assessment process. As a Cloud Center of Excellence (CCOE) team you can then choose to distribute the approved golden AMI to your business units based in other AWS accounts. Many compliance aware AWS customers I work with also want a compliance check set up to track non-golden AMI launches, which can be achieved via an AWS Config rule set up by the golden AMI pipeline.

It is a standard DevOps best practice to establish golden AMIs (and the resulting running instances) as immutable objects and to manage any changes through a standard pipeline. Golden AMI pipeline follows the same best practice and enables the requirement of patching by allowing you to decommission an affected golden AMI version and creating a new one. Also, over time, a golden AMI version becomes obsolete. You can decommission the version by executing an automation set up by the pipeline.

Here is an architecture diagram of the golden AMI creation process:



For more information on how a golden AMI is created by the pipeline, see the read-me guide available in the GitHub repository.

### How do I deploy the sample golden AMI pipeline?

The repository contains sample CloudFormation (CFN) templates and a read-me guide. You can use the CloudFormation Templates to set up the pipeline, however, instructions on how and where to execute these CloudFormation templates are available in the read-me guide. The read-me guide is a detailed step-by-step instruction guide, which contains instructions to:

1. Set up the pipeline infrastructure in the master account. Note that If you are using AWS organizations, this is not the master-payer account. It is an account that your Cloud Center Of Excellence (CCOE) team has identified as the master account.
2. Test the golden AMI pipeline. As part of the test, you would:
3. Create a golden AMI version (a product can have multiple golden AMI versions) you approve of. You can use your private AMI/Amazon-owned AMI/AWS Marketplace-based AMI as the source AMI.
   1. Distribute the golden AMI version to one or more accounts using AWS Lambda and AWS Systems Manager.
   2. Check if non-golden AMI launches are flagged as non-compliance via an AWS Config rule.
   3. Launch an EC2 instance from the golden AMI in a governed manner in the child account using AWS Service Catalog.
   4. Perform continuous security assessment of all active golden AMIs using Amazon Inspector.
   5. Decommission a golden AMI version.

Currently connectivity between AWS accounts and on premise networks

Amazon EC2 now allows **peering** relationships to be **established** between Virtual Private Clouds (VPCs) across **different** AWS **regions**. ... Inter-**Region VPC Peering** provides a simple and cost-effective way to share resources between **regions** or replicate data for geographic redundancy

Every **instance** in a **VPC** has a default **network interface**, called the primary **network interface** (eth0). **You** cannot detach a primary **network interface** from an **instance**. **You can** create and **attach** additional **network interfaces**. The maximum number of **network interfaces** that **you can** use varies by **instance** type.

Inter-**region** VPC peering connections allow secure communication between VPC resources in different **AWS Regions**. All network traffic between **regions** is encrypted, stays on the **AWS** global network backbone, and never traverses the public internet, thereby reducing threat vectors, such as common exploits and DDoS attacks.

Currently, **Amazon VPC** supports five (5) **IP address ranges**, **one** (1) primary and four (4) secondary for **IPv4**. Each of these **ranges can** be between /28 (in CIDR notation) and /16 in **size**. The **IP address ranges** of your **VPC** should not overlap with the **IP address ranges** of your existing network.

### **Formulating a Migration Strategy**

* New Network architecture can be established side by side with the current peering and VPN connections
* New network is implemented the transit gateway, associating VPCs with it and Adding custom route tables as well as updating existing VPC routetables without disrupting network connections.
* Once connectivity has been established routes for peering connections can be disabled
* Peering routes and Peering connections can be removed

Enterprises typically begin to contemplate how to migrate an application during the second phase of the “Migration Process” — Portfolio Discovery and Planning. This is when they determine what’s in their environment, what are the interdependencies, what’s going to be easy to migrate and what’s going to be hard to migrate, and how they’ll migrate each application.

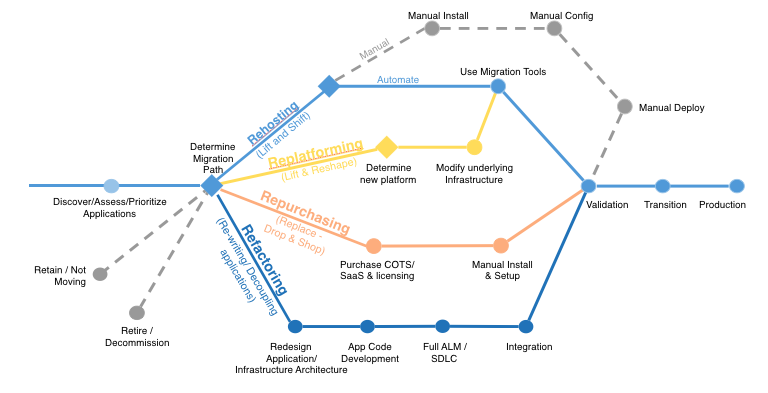
Using this knowledge, organizations can outline a plan (which should be considered subject to change as they progress through their migration and learn) on how they’ll approach migrating each of the applications in their portfolio and in what order.

The complexity of migrating existing applications varies, depending on the architecture and existing licensing arrangements. If I think about the universe of applications to migrate on a spectrum of complexity, I’d put a virtualized, service-oriented architecture on the low-complexity end of the spectrum, and a monolithic mainframe at the high-complexity end of the spectrum.

I suggest starting with something on the low-complexity end of the spectrum for the obvious reason that it will be easier to complete — which will give you some immediate positive reinforcement or “quick wins” as you learn.

### **6 Application Migration Strategies: “The 6 R’s”**

The 6 most common application migration strategies we see are:



1. **Rehosting** — Otherwise known as “lift-and-shift.”

We find that many early cloud projects gravitate toward net new development using cloud-native capabilities, but in a large legacy migration scenario where the organization is looking to scale its migration quickly to meet a business case, we find that the majority of applications are rehosted. GE Oil & Gas, for instance, found that, even without implementing any cloud optimizations, it could save roughly 30 percent of its costs by rehosting.

Most rehosting can be automated with tools (e.g. AWS VM Import/Export, Racemi), although some customers prefer to do this manually as they learn how to apply their legacy systems to the new cloud platform.

We’ve also found that applications are easier to optimize/re-architect once they’re already running in the cloud. Partly because your organization will have developed better skills to do so, and partly because the hard part — migrating the application, data, and traffic — has already been done.

2. **Replatforming** — I sometimes call this “lift-tinker-and-shift.”

Here you might make a few cloud (or other) optimizations in order to achieve some tangible benefit, but you aren’t otherwise changing the core architecture of the application. You may be looking to reduce the amount of time you spend managing database instances by migrating to a database-as-a-service platform like Amazon Relational Database Service (Amazon RDS), or migrating your application to a fully managed platform like Amazon Elastic Beanstalk.

A large media company we work with migrated hundreds of web servers it ran on-premises to AWS, and, in the process, it moved from WebLogic (a Java application container that requires an expensive license) to Apache Tomcat, an open-source equivalent. This media company saved millions in licensing costs on top of the savings and agility it gained by migrating to AWS.

3. **Repurchasing** — Moving to a different product.

I most commonly see repurchasing as a move to a SaaS platform. Moving a CRM to Salesforce.com, an HR system to Workday, a CMS to Drupal, and so on.

4. **Refactoring / Re-architecting** — Re-imagining how the application is architected and developed, typically using cloud-native features.

This is typically driven by a strong business need to add features, scale, or performance that would otherwise be difficult to achieve in the application’s existing environment.

Are you looking to migrate from a monolithic architecture to a service-oriented (or server-less) architecture to boost agility or improve business continuity (I’ve heard stories of mainframe fan belts being ordered on e-bay)? This pattern tends to be the most expensive, but, if you have a good product-market fit, it can also be the most beneficial.

5. **Retire** — Get rid of.

Once you’ve discovered everything in your environment, you might ask each functional area who owns each application. We’ve found that as much as 10% (I’ve seen 20%) of an enterprise IT portfolio is no longer useful, and can simply be turned off. These savings can boost the business case, direct your team’s scarce attention to the things that people use, and lessen the surface area you have to secure.

**6. Retain** — Usually this means “revisit” or do nothing (for now).

Maybe you’re still riding out some depreciation, aren’t ready to prioritize an application that was recently upgraded, or are otherwise not inclined to migrate some applications. You should only migrate what makes sense for the business; and, as the gravity of your portfolio changes from on-premises to the cloud, you’ll probably have fewer reasons to retain.

Security:

* Ingress/egress points across multiple VPCs are consolidated into one
* Security measures and provisions for the new networking architecture will not differ from the existing architecture
* Security Groups and NACLs will not have to be changed
* IAM doesn’t change between the two Network architectures

Enterprise IAM solution to provide access

Amazon takes the security of its services and resources very seriously. One of the areas that Amazon has focused on is providing a robust access control service to its Amazon Web Services (AWS) customers. AWS’s identity and access management (IAM) service allows customers to manage users, groups, roles, and permissions. But it’s entirely up to AWS customers to properly configure IAM to meet their security and compliance requirements.

To help you make the most of Amazon’s built-in controls, we’ve compiled the top 13 AWS IAM best practices every organization should follow.

#### 1) Restrict use of the AWS root account

When you register an account with AWS, the initial user account created is known as the root account. This account has complete access to every AWS resource (including billing information), making it the most privileged user account. Since root account credential permissions can’t be restricted, it is recommended that access to the root account be restricted to tasks that can only be completed using the root account (like creating the first admin account).

For all other tasks, create an IAM user that has administrative privileges, and use that account to manage your AWS environment on a day-to-day basis.

#### 2) Rotate root account access keys and enable multifactor authentication (MFA)

Regardless of how seldom the AWS root account is used, it is highly recommended that the account access keys are rotated on a regular basis using the security credentials [page](https://console.aws.amazon.com/iam/home?#security_credential), and MFA is enabled to use the root account

#### 3) Never share AWS account credentials

Instead of sharing AWS accounts, create individual IAM users for employees that need to access AWS resources. This will allow you to assign a unique set of permissions to different users based on their job requirements.

#### 4) If possible, use AWS managed policies to assign permissions

Amazon provides a [pre-defined set of policies](http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html#aws-managed-policies) that are completely managed by AWS and customers aren’t allowed to edit the permissions. These policies are designed to serve common use cases while making it easier to enforce access policies than creating policies yourself from scratch.

#### 5)  Assign permissions at IAM group/role level rather than individual IAM user level

AWS allows policies to be defined at the IAM user/group/role level when a new user/group/role is created (known as inline policies). AWS customers can also apply customer-managed policies (which could be derived from cloning AWS managed policies) to a set of IAM users, groups, or roles. As a best practice, avoid assigning customer-managed policies to individual IAM users or defining inline policies when creating an IAM user. Instead, assign policies to a group of IAM users or write inline policies when creating an IAM group. This streamlines the process of making changes to multiple user permissions and decreases the risk of accidentally giving individual users an unnecessarily high set of permissions.

#### 6) Never grant privileges beyond the minimum required for a user or group to fulfill their job requirements

One of the more common mistakes made when provisioning AWS users is granting them privileges that go beyond what’s minimally required. While it may be faster to create IAM users without scrutinizing the assigned privileges, unfettered access to AWS significantly increases the potential damage in the event of lost or stolen user credentials.

Verizon’s [Data Breach Investigations Report 2017](http://www.verizonenterprise.com/verizon-insights-lab/dbir/2017/) further underscores the threat from compromised accounts. The report finds that an alarming 81% of data breaches were caused by stolen or insecure passwords in 2016. It’s therefore important for IT security to ensure that AWS administrators perform adequate research in order to define the right set of permissions.

***Tip***: if you’re not sure whether users/groups/roles have minimal required privileges, use the Access Advisor tab (found in the IAM console details page) to analyze a user/group/role, which will tell you which services are actually being used by them. This can help determine which services should be blocked.

#### 7) Set a regular cadence to review IAM permissions

As a security best practice, it’s important to regularly review your organization’s AWS IAM policies to ensure they’re granting least privileges. Each policy comes with a [policy summary](http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_understand.html), which is a good place to start when auditing IAM policies. AWS provides four levels of access for each of its services: list, read, write, and permissions management.

The write and permissions management access levels should be granted with caution. Write permits users to create, delete, or modify resources. Permissions management allows a user to grant or restrict resource permissions for entire organization and its AWS users. For this reason, permissions management should be granted to as few IAM users as possible.

#### 8) Enforce a strong password policy for all AWS users

When left to their own devices, most users will select an easy to guess password despite its security risks. [According to Skyhigh](https://www.skyhighnetworks.com/cloud-security-blog/skyhigh-research-finds-password-insecure-reuse-cloud/), the top 20 most common passwords account for 10% of all passwords, with gems like “123456” and “password” accounting for over 5% of all passwords. This means that a hacker can infiltrate one out of twenty user accounts without any brute-force attacks if he just uses the two aforementioned passwords. It’s possible, however, to create [highly secure and easy to remember passwords](https://www.skyhighnetworks.com/cloud-security-blog/how-to-create-a-strong-password-you-actually-remember/) and taking the following steps can help ensure IAM credentials are protected:

1. Define a minimum password length of at least 14 characters
2. Require non-alphabet characters, at least one uppercase alphabet, and a symbol
3. Set up a password expiration policy and disallow password reuse
4. Discourage privileged users from using dictionary words in their passwords

#### 9) Enable multi-factor authentication (MFA) for all IAM users

Organizations should assume that at least some of their cloud service user credentials have already been compromised. Since employees have a tendency to re-use passwords across services, enabling MFA should be a requirement not just for AWS but any cloud service that is being used at an organization.

Configuring AWS with a single sign-on provider such as Okta, Ping Identity, or Azure Active Directory can ease the friction introduced by enabling MFA by standardizing authentication factors across all applications employees use.

#### 10) Use IAM roles for custom applications running on AWS EC2

Any app that requires access to another AWS service in order to function will need its own credentials. As a security best practice, use IAM roles to provide credentials to the application. Roles can have their own set of permissions without any users/groups needing to be attached to them.

When a user first launches an EC2 instance, they can determine the role for that instance, and any application that is deployed on that EC2 instance can then use the role’s credentials to access other AWS resources. The credentials will be dynamic and AWS will rotate them automatically for additional security.

#### 11) Remove unused, stale, or unnecessary IAM users/credentials

To minimize the risk of IAM user credentials being stolen, all unnecessary IAM users should have their access terminated. Employees who no longer work at an organizations or those who have moved on to a different department that don’t need AWS access should not have access to AWS. Organizations should audit their IAM user activity to see which users haven’t logged into AWS for at least 90 days, and revoke their permissions as well.

#### 12) Organizations working with highly sensitive or regulated information (e.g. healthcare, finance, federal government) should use policy conditions as an added security measure

Policy conditions are boolean operators that determine whether a match exists between a policy and a request. There are a near limitless number of conditions that can be used for IAM policies. Policy conditions are especially useful when working with partners and third-party vendors who need access to your organization’s AWS resources. Below are some of the recommended conditions that should be employed:

1. Use date/time conditions to limit access to resources such that IAM users are only able to access a resource during weekdays for the duration of their workday/shift.
2. Set up conditions that whitelists IP addresses that are allowed to access AWS resources to ensure only trusted IP addresses are able to gain access to an AWS resource.
3. For contract employees/partners, set up date conditions that block access to AWS resources after the termination date of the contract.

Only use policy conditions that meet your organization’s security and compliance requirements without hindering employee productivity.

#### 13) Monitor user activity in all cloud services (including IAM user activity) to identify anomalous activity indicative of threats arising from a compromised account, or malicious/negligent internal employee

Amazon supports AWS activity monitoring with AWS CloudTrail. However, organizations should monitor AWS activity alongside all other cloud services to attain a unified view of all cloud activity. This will help to accurately separate a true threat from a false positive alert and also provide insight into cross-cloud threats that would go undetected by only looking at each cloud service in isolation

**AWS** Single Sign-On (**AWS SSO**) is a cloud service that allows you to grant your users **access** to **AWS** resources, such as **Amazon EC2** instances, across multiple **AWS** accounts. By default, **AWS SSO** now provides a directory that you can use to create users, organize them in groups, and set permissions across those groups.

**AD** is a directory services database, and **LDAP** is one of the protocols you can use to talk to it. **LDAP** is a standard, **AD** is Microsoft's (proprietary) implementation (and more)

# How to create and manage users within AWS Single Sign-On

[AWS Single Sign-On](https://aws.amazon.com/single-sign-on/) (AWS SSO) is a cloud service that allows you to grant your users access to AWS resources, such as Amazon EC2 instances, across multiple AWS accounts. By default, AWS SSO now provides a directory that you can use to create users, organize them in groups, and set permissions across those groups. You can also grant the users that you create in AWS SSO permissions to applications such Salesforce, Box, and Office 365. AWS SSO and its directory are available at no additional cost to you.

A directory is a key building block that allows you to manage the users to whom you want to grant access to AWS resources and applications. [AWS Identity and Access Management](https://aws.amazon.com/iam/) (IAM) provides a way to create users that can be used to access AWS resources within one AWS account. However, many businesses prefer an approach that enables users to sign in once with a single credential and access multiple AWS accounts and applications. You can now create your users centrally in AWS SSO and manage user access to all your AWS accounts and applications. Your users sign in to a user portal with a single set of credentials configured in AWS SSO, allowing them to access all of their assigned accounts and applications in a single place.

## How to create users and groups in AWS SSO

You can create users in AWS SSO by configuring their email address and name. When you create a user, AWS SSO sends an email to the user by default so that they can set their own password. Your user will use their email address and a password they configure in AWS SSO to sign into the user portal and access all of their assigned accounts and applications in a single place.

You can also add the users that you create in AWS SSO to groups you create in AWS SSO. In addition, you can create [permissions sets](https://docs.aws.amazon.com/singlesignon/latest/userguide/permissionsetsconcept.html) that define permitted actions on an AWS resource, and assign them to your users and groups. For example, you can grant the DevOps group permissions to your production AWS accounts. When you add users to the DevOps group, they get access to your production AWS accounts automatically.

In this post, I will show you how to create users and groups in AWS SSO, how to create permission sets, how to assign your groups and users to permission sets and AWS accounts, and how your users can sign into the AWS SSO user portal to access AWS accounts. To learn more about how to grant users that you create in AWS SSO permissions to business applications such as Office 365 and Salesforce, see [Manage SSO to Your Applications](https://docs.aws.amazon.com/singlesignon/latest/userguide/manage-your-applications.html).

## Walk-through prerequisites

For this walk-through, I assume the following:

* You’ve enabled AWS SSO for your [AWS Organization](https://aws.amazon.com/organizations/). To learn more, see [Enable AWS SSO](https://docs.aws.amazon.com/singlesignon/latest/userguide/step1.html).
* You’ve added the AWS accounts to which you want to grant AWS SSO access to your organization. To learn more, see [Managing the AWS Accounts in Your Organization](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_accounts.html).
* You’ve signed into the [AWS Management Console](https://aws.amazon.com/console/) with your AWS Organizations master account credentials. To learn more about AWS Organizations and master accounts, see [AWS Organizations FAQs](https://aws.amazon.com/organizations/faqs/).
* You’ve required permissions to use the AWS SSO Console. To learn more, see [Permissions Required to Use the AWS SSO Console](https://docs.aws.amazon.com/singlesignon/latest/userguide/iam-auth-access-using-id-policies.html#requiredpermissionsconsole).

## Overview

To illustrate how to add users in AWS SSO and how to grant permissions to multiple AWS accounts, imagine that you’re the IT manager for a company, Example.com, that wants to make it easy for its users to access resources in multiple AWS accounts. Example.com has five AWS accounts: a master account (called MasterAcct), two developer accounts (DevAccount1 and DevAccount2), and two production accounts (ProdAccount1 and ProdAccount2). Example.com uses [AWS Organizations](https://aws.amazon.com/organizations/) to manage these accounts and has already [enabled AWS SSO](https://docs.aws.amazon.com/singlesignon/latest/userguide/step1.html).

Example.com has two developers, Martha and Richard, who need full access to [Amazon EC2](https://aws.amazon.com/ec2/) and [Amazon S3](https://aws.amazon.com/s3/) in the developer accounts (DevAccount1 and DevAccount2) and read-only access to EC2 and S3 resources in the production accounts (ProdAccount1 and ProdAccount2).

The following diagram illustrates how you can grant Martha and Richard permissions to the developer and production accounts in four steps:

1. Add users and groups in AWS SSO: Add users Martha and Richard in AWS SSO by configuring their names and email addresses. Add a group called Developers in AWS SSO and add Martha and Richard to the Developers group.
2. Create permission sets: Create two permission sets. In the first permission set, include policies that give full access to Amazon EC2 and Amazon S3. In second permission set, include policies that give read-only access to Amazon EC2 and Amazon S3.
3. Assign groups to accounts and permission sets: Assign the Developers group to your developer accounts and assign the permission set that gives full access to Amazon EC2 and Amazon S3. Assign the Developers group to your production accounts, too, and assign the permission set that gives read-only access to Amazon EC2 and Amazon S3. Martha and Richard now have full access to Amazon EC2 and Amazon S3 in the developer accounts and read-only access in the production accounts.
4. Users sign into the User Portal to access accounts: Martha and Richard receive email from AWS to set their passwords with AWS SSO. Martha and Richard can now sign into the AWS SSO User Portal using their email addresses and the passwords they set with AWS SSO, allowing them to access their assigned AWS accounts.

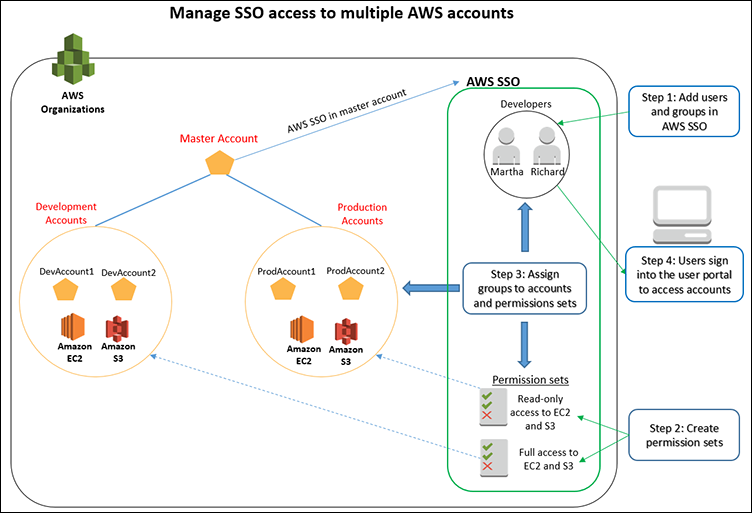


Figure 1: Architecture diagram

## Step 1: Add users and groups in AWS SSO

To add users in AWS SSO, navigate to the [AWS SSO Console](https://console.aws.amazon.com/singlesignon). Then, follow the steps below to add Martha as a user, to create a group called Developers, and to add Martha to the Developers group in AWS SSO.

1. In the **AWS SSO Dashboard**, choose **Manage your directory** to navigate to the **Directory** tab.

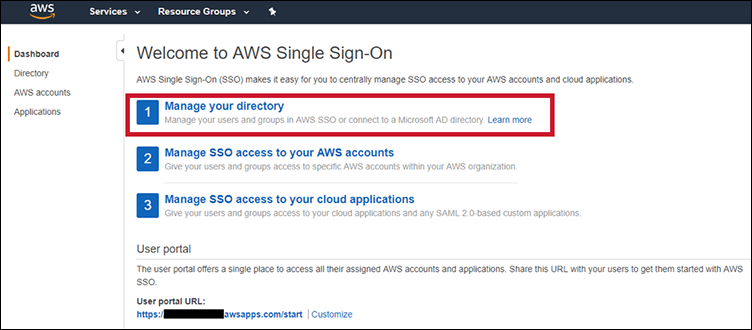


Figure 2: Navigating to the “Manage your directory” page

1. By default, AWS SSO provides you a directory that you can use to manage users and groups in AWS SSO. To add a user in AWS SSO, choose **Add user.** If you previously connected a Microsoft AD directory with AWS SSO, you can switch to using the directory that AWS SSO now provides by default by following the steps in [Change Directory](https://docs.aws.amazon.com/singlesignon/latest/userguide/manage-connected-directory.html).

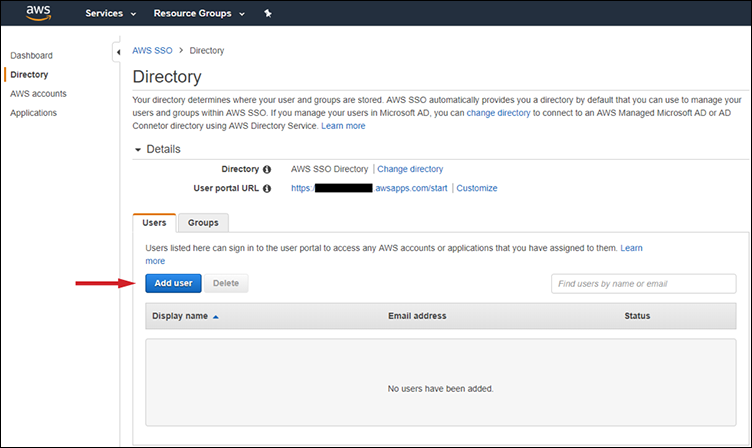


Figure 3: Adding new users to your directory

1. On the **Add User** page, enter an email address, first name, and last name for the user, then create a display name. In this example, you’re adding “Martha Rivera” as a user. For the password, choose **Send an email to the user with password instructions**. This allows users to set their own passwords.

Optionally, you can also set a **mobile phone number** and add additional user attributes.

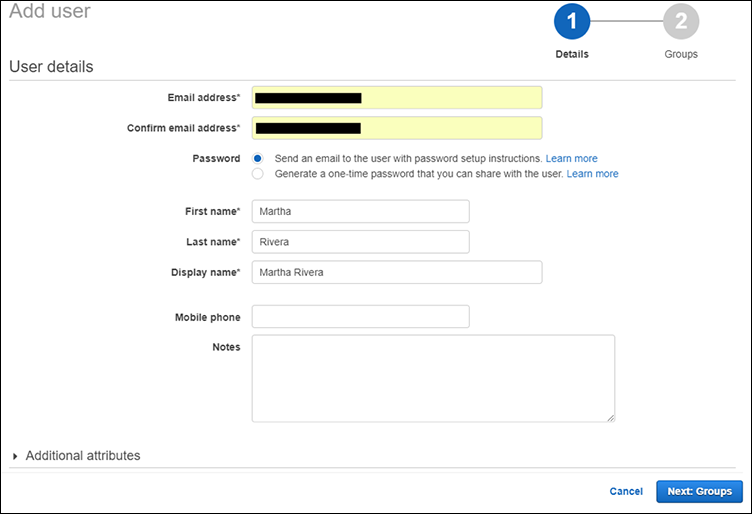


Figure 4: Adding user details

1. Next, you’re ready to add the user to groups. First, you need to create a group. Later, in Step 3, you can grant your group permissions to an AWS account so that any users added to the group will inherit the group’s permissions automatically. In this example, you will create a group called Developers and add Martha to the group. To do so, from the **Add user to groups** page, choose **Create group**.

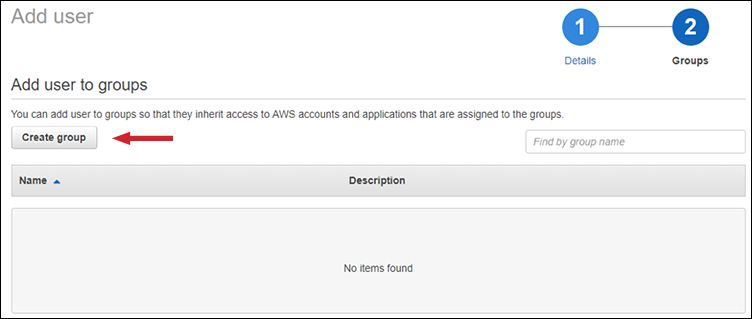


Figure 5: Creating a new group

1. In the **Create group** window, title your group by filling out the **Group name** field. For this example, enter **Developers**. Optionally, you can also enter a description of the group in the **Description** field. Choose **Create** to create the group.

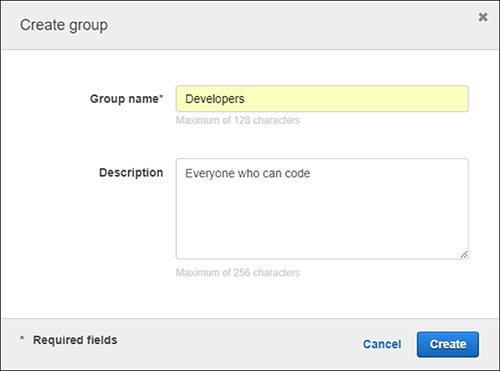


Figure 6: Adding a name and description to your new group

1. On the **Add users to group** page, check the box next to the group you just created, and then choose **Add user**. Following this process will allow you to add Martha to the Developers group.

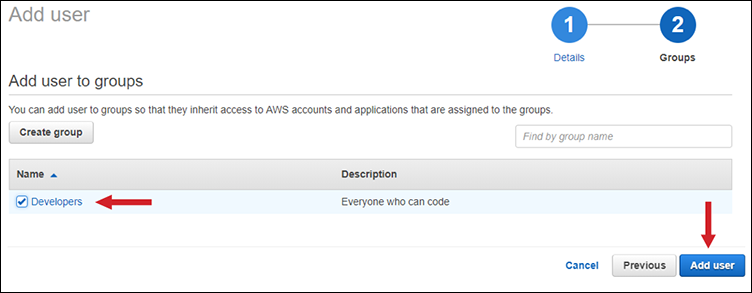


Figure 7: Adding a user to your new group

You’ve successfully created the user Martha and added her to the Developers group. You can repeat sub-steps 2, 3, and 6 above to create more users and add them to the group. This is the process you should follow to create the user Richard and add him to the Developers group.

Next, you’ll grant the Developers group permissions to AWS resources within multiple AWS accounts. To follow along, you’ll first need to create permission sets.

## Step 2: Create permission sets

To grant user permissions to AWS resources, you must create permission sets. A permission set is a collection of administrator-defined policies that AWS SSO uses to determine a user’s permissions for any given AWS account. Permission sets can contain either [AWS managed policies](http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html#aws-managed-policies) or custom policies that are stored in AWS SSO. Policies contain statements that represent individual access controls (allow or deny) for various tasks. This determines what tasks users can or cannot perform within the AWS account. To learn more about permission sets, see [Permission Sets](https://docs.aws.amazon.com/singlesignon/latest/userguide/permissionsetsconcept.html).

For this use case, you’ll create two permissions sets: 1) EC2AndS3FullAccess, which has [AmazonEC2FullAccess](https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AmazonEC2FullAccess) and [AmazonS3FullAccess](https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AmazonS3FullAccess) managed policies attached and 2) EC2AndS3ReadAccess, which has [AmazonEC2ReadOnlyAccess](https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess) and [AmazonS3ReadOnlyAccess](https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess) managed policies attached. Later, in Step 3, you can assign groups to these permissions sets and AWS accounts, so that your users have access to these resources. To learn more about creating permission sets with different levels of access, see [Create Permission Set](https://docs.aws.amazon.com/singlesignon/latest/userguide/permissionsets.html#howtocreatepermissionset).

Follow the steps below to create permission sets:

1. Navigate to the [**AWS SSO Console**](https://console.aws.amazon.com/singlesignon) and choose **AWS accounts** in the left-hand navigation menu.
2. Switch to the **Permission sets** tab on the **AWS Accounts** page, and then choose **Create permissions set**.

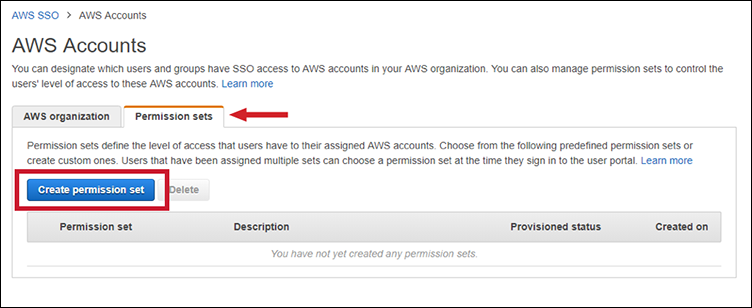


Figure 8: Creating a permission set

1. On the**Create new permissions set** page, choose **Create a custom permission set**. To learn more about choosing between an existing job function policy and a custom permission set, see [Create Permission Set](https://docs.aws.amazon.com/singlesignon/latest/userguide/permissionsets.html#howtocreatepermissionset).

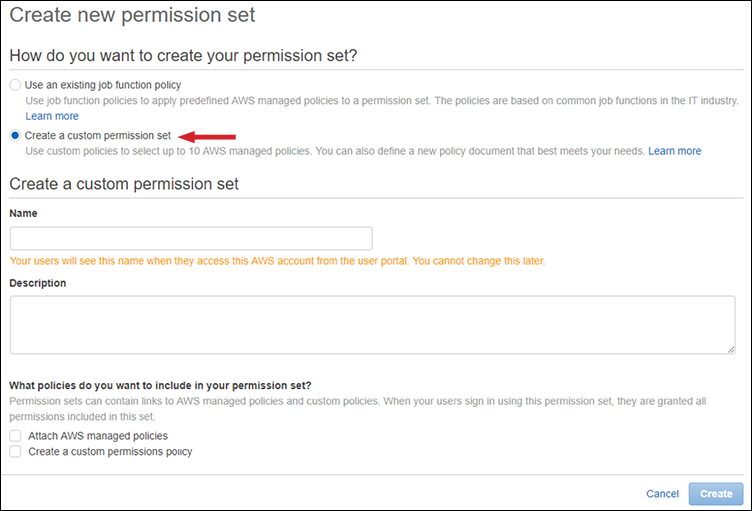


Figure 9: Customizing a permission set

1. Enter **EC2AndS3FullAccess** in the **Name** field and choose **Attach AWS managed policies**. Then choose **AmazonEC2FullAccess** and **AmazonS3FullAccess**. Choose **Create** to create the permission set.

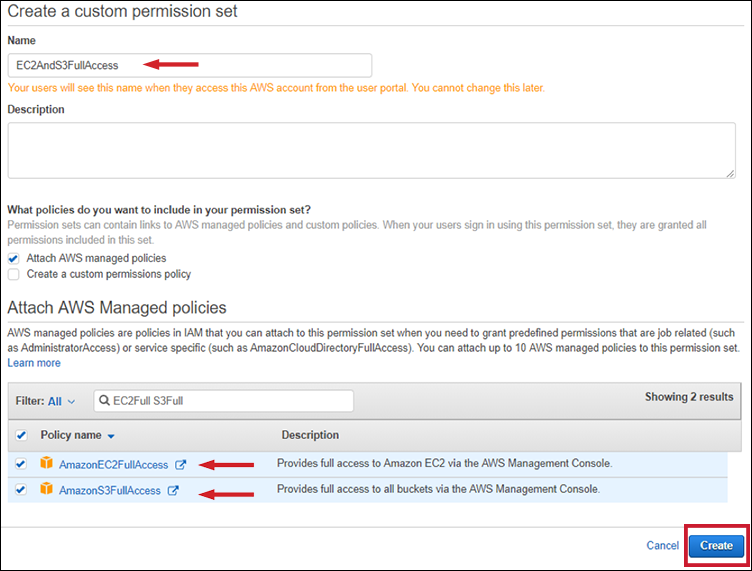


Figure 10: Attaching AWS managed policies to your permission set

You’ve successfully created a permission set. You can use the steps above to create another permission set, called **EC2AndS3ReadAccess**, by attaching the **AmazonEC2ReadOnlyAccess** and **AmazonS3ReadOnlyAccess** managed policies. Now you’re ready to assign your groups to accounts and permission sets.

## Step 3: Assign groups to accounts and permission sets

In this step, you’ll assign your Developers group full access to Amazon EC2 and Amazon S3 in the developer accounts and read-only access to these resources in the production accounts. To do so, you’ll assign the Developers group to the EC2AndS3FullAccess permission set and to the two developer accounts (DevAccount1 and DevAccount2). Similarly, you’ll assign the Developers group to the EC2AndS3ReadAccess permission set and to the production AWS accounts (ProdAccount1 and ProdAccount2).

Follow the steps below to assign the **Developers** group to the **EC2AndS3FullAccess** permission set and developer accounts (**DevAccount1** and **DevAccount2**). To learn more about how to manage access to your AWS accounts, see [Manage SSO to Your AWS Accounts](https://docs.aws.amazon.com/singlesignon/latest/userguide/manage-your-accounts.html).

1. Navigate to the [**AWS SSO Console**](https://console.aws.amazon.com/singlesignon) and choose **AWS Accounts** in the left-hand navigation menu.
2. Switch to the **AWS organization** tab and choose the accounts to which you want to assign your group. For this example, select accounts **DevAccount1** and **DevAccount2** from the list of AWS accounts. Next, choose **Assign users**.

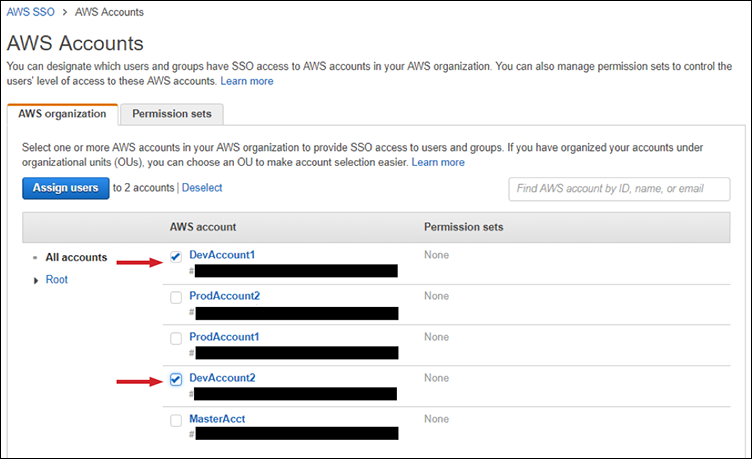


Figure 11: Assigning users to your accounts

1. On the **Select users and groups** page, type the name of the group you want to add into the search box and choose **Search**. For this example, you will be looking for the group called **Developers**. Check the box next to the correct group and choose **Next: Permission Sets**.

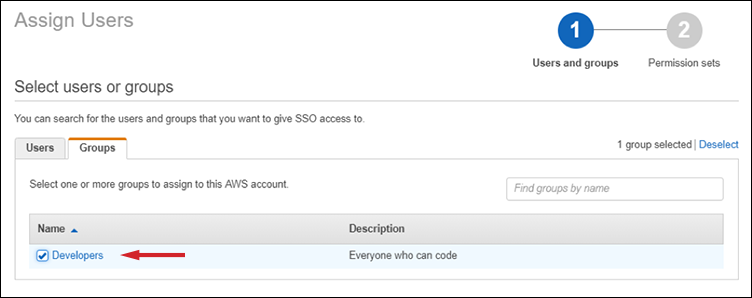


Figure 12: Setting permissions for the “Developers” group

1. On the **Select permissions sets** page, select the permission sets that you want to assign to your group. For this use case, you’ll select the **EC2AndS3FullAccess** permission set. Then choose **Finish**.

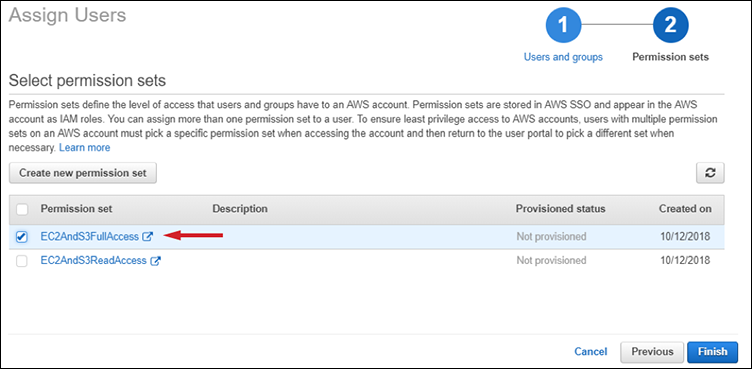


Figure 13: Choosing permission sets

You’ve successfully granted users in the Developers group access to accounts DevAccount1 and DevAccount2, with full access to Amazon EC2 and Amazon S3.

You can follow the same steps above to grant users in the Developers group access to accounts ProdAccount1 and ProdAccount2 with the permissions in the EC2AndS3ReadAccess permission set. This will grant the users in the Developers group read-only access to Amazon EC2 and Amazon S3 in the production accounts.

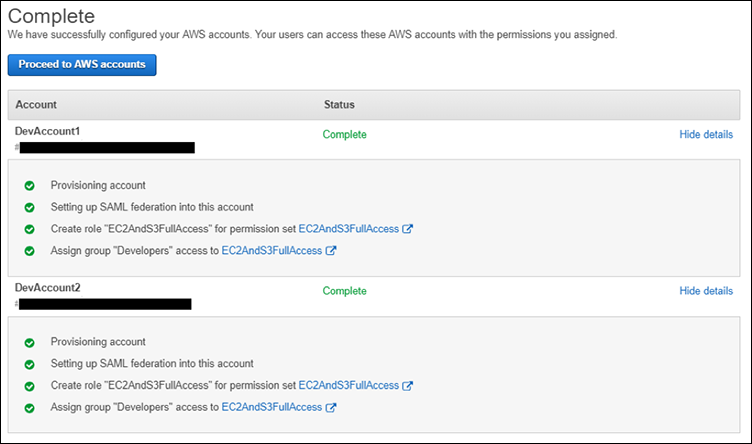


Figure 14: Notification of successful account configuration

## Step 4: Users sign into User Portal to access accounts

Your users can now sign into the [AWS SSO User Portal](https://docs.aws.amazon.com/singlesignon/latest/userguide/using-the-portal.html) to manage resources in their assigned AWS accounts. The user portal provides your users with single sign-on access to all their assigned accounts and business applications. From the user portal, your users can sign into multiple AWS accounts by choosing the AWS account icon in the portal and selecting the account that they want to access.

You can follow the steps below to see how Martha signs into the user portal to access her assigned AWS accounts.

1. When you added Martha as a user in Step 1, you selected the option **Send the user an email with password setup instructions**. AWS SSO sent instructions to set a password to Martha at the email that you configured when creating the user. This is the email that Martha received:

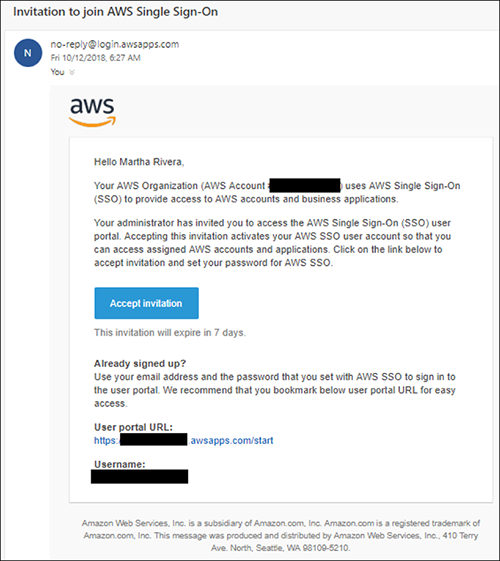


Figure 15: AWS SSO password configuration email

1. To set her password, Martha will select **Accept invitation** in the email that she received from AWS SSO. Selecting Accept invitation will take Martha to a page where she can set her password. After Martha sets her password, she can navigate to the User Portal.

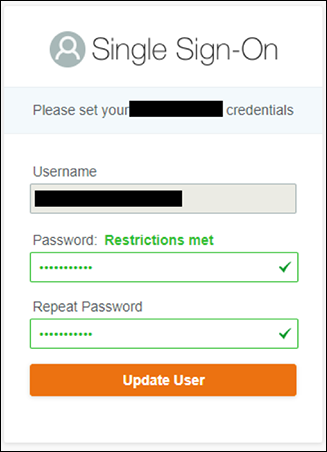


Figure 16: User Portal sign-in

1. In the User Portal, Martha can select the **AWS Account** icon to view all the AWS accounts to which she has permissions.

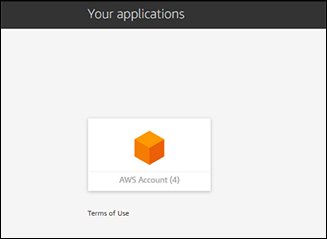


Figure 17: View of AWS Account icon from User Portal

1. Martha can now see the developer and production accounts that you granted her permissions to in previous steps. For each account, she can also see the list of roles that she can assume within the account. For example, for DevAccount1 and DevAccount2, Martha can assume the EC2AndS3FullAccess role that gives her full access to manage Amazon EC2 and Amazon S3. Similarly, for ProdAccount1 and ProdAccount2, Martha can assume the EC2AndS3ReadAccess role that gives her read-only access to Amazon EC2 and Amazon S3. Martha can select accounts and choose **Management Console** next to the role she wants to assume, letting her sign into the AWS Management Console to manage AWS resources. To switch to a different account, Martha can navigate to the **User Portal** and select a different account. From the User Portal, Martha can also get temporary security credentials for short-term access to resources in an AWS account using [AWS Command Line Interface](https://aws.amazon.com/cli/) (CLI). To learn more, see [How to Get Credentials of an IAM Role for Use with CLI Access to an AWS Account](https://docs.aws.amazon.com/singlesignon/latest/userguide/howtogetcredentials.html).

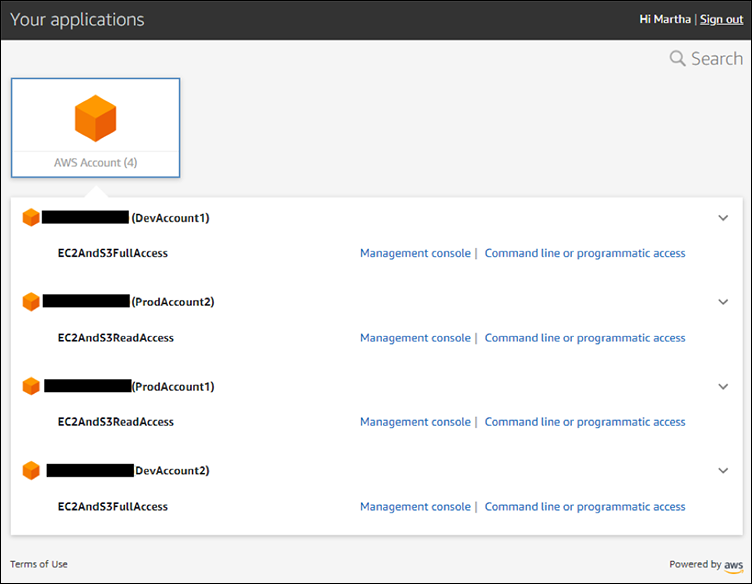


Figure 18: Switching accounts from the User Portal

1. Martha bookmarks the user portal URL in her browser so that she can quickly access the user portal the next time she wants to access AWS accounts.