Setting Up Multiuser Environments in the AWS Cloud (for Classroom Training and Research)

KD Singh
Leo Zhadanovsky

October 2013

(Please consult **http://aws.amazon.com/whitepapers** for the latest version of this paper)

Table of Contents

Abstract

Amazon Web Services (AWS) can provide the ideal environment for classroom training and research. Educators can use AWS for student labs, training applications, individual IT environments, and cloud computing courses. This whitepaper provides an overview of how to create and manage multiuser environments in the AWS cloud so that professors and researchers can leverage cloud computing in their projects.

Introduction

With AWS you can requisition compute, storage, and other services on demand, gaining access to a suite of secure, scalable, and flexible IT infrastructure services as your organization needs them. This allows the educators, academic researchers and students to tap into the on-demand infrastructure of AWS to teach advanced courses, tackle research endeavors, and explore new projects – tasks that previously would have required expensive up-front and ongoing investments in infrastructure. For more information, see AWS in Education and AWS Products & Services.

To access any AWS service, you need an AWS account. Each AWS account is typically associated with a payment instrument (credit card or invoicing). You can create an AWS account for any entity, such as a professor, student, class, department, or institution. When you create an AWS account, you can sign into the AWS Management Console and access a variety of AWS services. In addition to creating an AWS account with a username and password, you can also create a set of access keys that you can use to access services via APIs or command line tools. It is highly recommended that you protect these security credentials and not share them publicly. For more information, see AWS Security Credentials and AWS Management Console.

If you require more than one person to access your AWS account, AWS Identity and Access Management (IAM) enables you to create multiple users and manage the permissions for each of these users within your AWS account. A user is a unique identity recognized by AWS services and applications. Similar to a login user in an operating system like Windows or Linux, a user has a unique name and can identify itself using various kinds of security credentials, such as username and password or an access key ID and accompanying secret access key. A user can be an individual, such as a student or teaching assistant, or an application, such as a research application, that requires access to AWS services. You can create users, groups, roles and federation capabilities using the AWS Management Console, APIs, or a variety of partner products. For instructions on how to create new users and manage AWS credentials, see Adding an IAM User to Your AWS Account in the AWS Identity and Access Management documentation.

Depending on your teaching or research needs, there are several ways to set up a multiuser environment in the AWS cloud. The following sections introduce three possible scenarios.

Scenario 1: Individual Server Environments

The “Individual Server Environments” scenario is excellent for labs and other classwork that requires users to access their own pre-provisioned Linux or Windows servers running in the AWS cloud. The servers are Amazon Elastic Compute Cloud (Amazon EC2) instances. The instances can be created by an administrator with a customized configuration that includes applications and data needed to perform tasks for labs or assignments. This scenario is easy to set up and manage. It does not require users to have their own AWS accounts or credentials for more than their individual servers. Users do not have access to allocate additional resources on the AWS cloud.

As an example, consider a class with 25 students. The administrator creates 25 private keys (or one shared private key) and launches 25 Amazon EC2 instances; one instance for each student. The administrator shares the appropriate key or password with each student and provides instructions on how to log in to their instance. In this case, students do not have access to the AWS Management Console, APIs, or any other AWS service. Each student gets a unique private key (in case of Linux) or a username and password (in case of Windows) along with the public hostname or IP address of the instance that they can use to log in.

Scenario 2: Limited User Access to AWS Management Console

The "Limited User Access to AWS Management Console" scenario is excellent for users that require control of AWS resources, such as students in cloud computing or high performance computing (HPC) classes. With this scenario, users are given restricted access to the AWS services through their IAM credentials.

As an example, consider a class with 25 students. The administrator creates 25 IAM users using the AWS Management Console or APIs, and provides each student with their IAM credentials (username and password) and Login URL to the AWS Management Console. The administrator also creates a group policy or individual user policies that allow or deny access to different services. Each student (IAM user) has access to resources and services as defined by these access control policies set by the administrator. Students can log in to the AWS Management Console to access different AWS services as defined the policy, for example, launch Amazon EC2 Instances and store objects in Amazon S3.

Scenario 3: Separate AWS Account for Each User

The "Separate AWS Account for Each User" scenario with optional consolidated billing provides an excellent environment for users who need a completely separate account environment, such as researchers or graduate students. It is similar to the "Limited User Access to AWS Management Console" scenario, except that each IAM user is created in a separate AWS account, eliminating the risk of users affecting each other's services.

As an example, consider a research lab with 10 graduate students. The administrator creates one paying AWS account, 10 linked student AWS accounts, and 1 restricted IAM user per linked account. The administrator provisions separate AWS accounts for each user and links the accounts to the paying AWS account. Within each account, the administrator creates an IAM user and applies access control policies. Users receive access to an IAM user within their AWS account. They can log into the AWS Management Console to launch and access different AWS services, subject to the access control policy applied to their account. Students don't see resources provisioned by other students.

One key advantage of this scenario is the ability for a student to continue using the account after the completion of the course. For example, if students use AWS resources as part of a startup course, they can continue to use what they have built on AWS after the semester is over.

Comparing the Scenarios

The scenario you should select depends on your requirements. Table 1 provides a comparison of key features of these three scenarios.

	Individual Server Environments	**Limited User Access to AWS Management Console**	**Separate AWS Account for Each User**
Examples	Undergraduate labs	Graduate classes	Graduate research labs
Example uses	Labs or course work requiring a virtual server, AWS service, or separate application instance	Courses in cloud computing or labs requiring variable resource needs (e.g., HPC)	Courses for startups, thesis, or research projects
Separate AWS accounts required for each user	No	No	Yes
Major steps for setup	Create and allocate Amazon EC2 resources and associated credentials	Create IAM users, create policies, and distribute credentials	Create separate linked AWS accounts plus the steps in the "Setting Up Scenario 2: Limited User Access to AWS Management Console" section
Users can provision additional AWS resources, resulting in additional charges	No	Yes, depending on IAM services provided to users	Yes, depending on IAM services provided to users
Users have access to AWS Management Console or APIs	No	Yes	Yes
User charges paid by paying AWS account	Yes	Yes	Yes, if optional consolidated billing is used
Separation between user environments	Yes, if access credentials are not shared	Yes, if optional resource-based permissions are configured	Yes
Individual user credit cards or invoicing required	No	No	No, if optional consolidated billing is used
Billing alerts can be used to monitor charges	Yes	Yes	Yes

Table 1: Comparison of Scenarios

A large number of real-world use cases can benefit from implementing these scenarios. Here we focus on the education sector where multiuser, shared environments are required for setting up online classes, labs, and workshops for students. Both user and resource management are critical in these scenarios. Depending on your specific requirements, any of these scenarios can be used for setting up classrooms in the AWS cloud. The following sections describe each of these scenarios in more detail.

Setting Up Scenario 1: Individual Server Environments

With the "Individual Server Environments" scenario, users are provided access credentials to AWS resources. Users cannot access the AWS Management Console or launch new services. They receive the credentials to access specific AWS services that have already been launched by an administrator.

This scenario is a good match for simpler use cases in which users do not need to launch new AWS services. Figure 1 shows the architecture for this scenario.

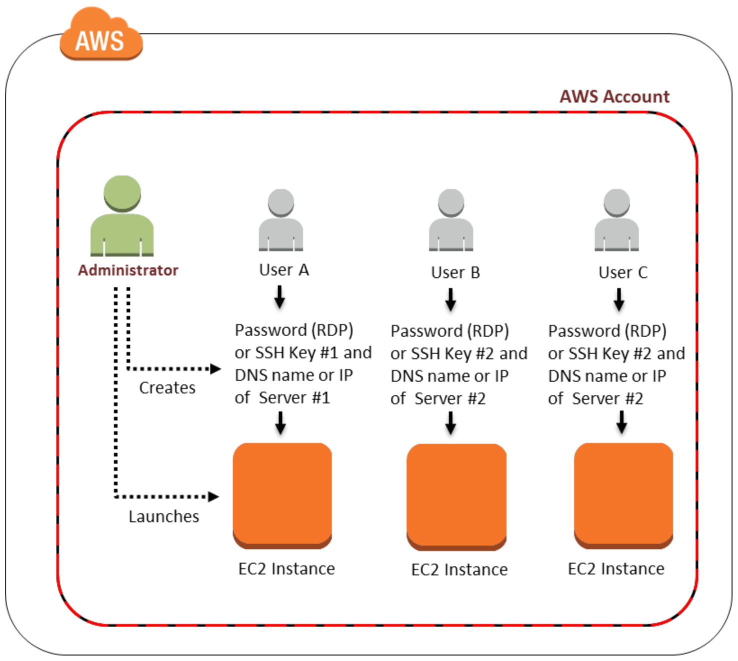

AWS
AWS Account
Administrator
User A
User B
User C
Password (RDP) or SSH Key #1 and DNS name or IP of Server #1
Password (RDP) or SSH Key #2 and DNS name or IP of Server #2
Password (RDP) or SSH Key #2 and DNS name or IP of Server #2
Creates
Launches
EC2 Instance
EC2 Instance
EC2 Instance

Figure 1: Individual Server Environments

An administrator can give users their own unique access keys, (SSH keys for Linux and password for Windows) for security and separation between users. For labs that do not require security among users (e.g., collaborative labs) the administrator can keep the keys or access credentials common for all the servers and just provide the unique access public DNS names of instances to the users. The administrator can choose the level of security and management appropriate for their needs.

Account Setup

The administrator creates an AWS account for the user group. For example, this can be a shared account for a professor, class, department, or school. The administrator can also use an existing AWS account. New AWS account signup and access to existing AWS accounts is available at Your Account.

The administrator then launches the required AWS services for each user and provides resource access credentials to the users.

Cost Tracking

If needed, the administrator tags the resources launched for different users. Cost allocation and resource tagging can help track usage by different users. For more information, see Costs Allocation and Tagging in the AWS Account Billing documentation.

Monitoring Resources

The administrator can set up Amazon CloudWatch alarms and billing alerts to monitor AWS resources. Billing alerts automatically notify the designated recipient whenever the estimated AWS charges reach a specified threshold. The administrator can choose to receive an alert on the total AWS charges or charges for a specific AWS product or service. If the account has any limits, the administrator can use these as the threshold for receiving billing alerts. For information about setting up billing alerts with CloudWatch, see Monitor Your Estimated Charges Using Amazon CloudWatch in the Amazon CloudWatch Developer Guide.

Reporting

Detailed usage reports for the AWS environment are available for the administrator. These include specifics on monthly charges as well as account activity in hourly increments. For more information, see Detailed Reports in the AWS Account Billing documentation.

Runtime Environment

After the administrator provisions the account and launches the required AWS services, users can access their AWS resources using the provided credentials. For example, if Amazon EC2 instances are part of the class, users would be given keys or passwords to SSH into Linux instances or RDP into Windows instances. Users would not have the credentials to log into the AWS Management Console or to launch any new services.

Environment Termination

When users have finished their work or when the account limits are reached, the administrator can terminate the AWS services. Since the student users do not have their own AWS accounts, terminating the launched services ensures that user work is deleted and further charges are discontinued.

Setting Up Scenario 2: Limited User Access to AWS Management Console

For the "Limited User Access to AWS Management Console" scenario, administrator creates IAM users and give each one access credentials. With IAM, an administrator can securely control access to AWS services and resources. Administrator can create and manage AWS users and groups and use permissions to allow and deny access to AWS resources. Users can log into the AWS Management Console and then they can launch and access different AWS services, subject to the access control policies applied to their account. Users have direct control over the access credentials for their resources.

By default, when you create IAM users, they don't have access to any AWS resources. You must explicitly grant them permissions to access the resources that they need for their work. Permissions are rights that you grant to a user or group to let the user perform tasks in AWS. Permissions are attached to an IAM user and let the administrator specify what that user can do. Depending upon the context, administrators may be able to construct resource-level permissions for users that control the actions the user is allowed to take for specific resources (for example, limiting which instance the user is allowed to terminate). For an overview of IAM permissions, see Overview of AWS IAM Permissions in the AWS Identity and Access Management documentation and read the blog post on AWS Security Blog.

To define permissions, administrators use policies, which are documents in JSON format. A policy consists of one or more statements, each of which describes one set of permissions. Policies can be attached to IAM users, groups, or roles. AWS Policy Generator is a handy tool that lets administrators create policies easily. To run the tool, go to AWS Policy Generator. Example policies that are relevant to multiuser environments are provided in "Appendix B: Example IAM User Policies" of this whitepaper. For more information about policies, see Overview of AWS IAM Policies in the AWS Identity and Access Management documentation.

A useful option in this scenario is for the administrator to tag resources and write appropriate resource-level permissions to limit IAM users to specific actions and resources. A tag is a label you assign to an AWS resource. For services that support tagging, you apply tags using the AWS Management Console or API requests. This allows very fine-grained control on which resources a user can access and what actions they can take on those resources. The administrator will also need to write policies to prevent users from manipulating the resource tags. For example, for Amazon EC2 tags, the administrator should disable the `ec2:CreateTags` and `ec2:DeleteTags` actions.

The "Limited User Access to AWS Management Console" scenario is also good for use cases that require collaboration among users. As described previously, a user can give other IAM users access to specific actions on their resources using a mix of user-level and resource-level permissions. A good example is a collaborative research project where students allow other members of their team access to software in their Amazon EC2 instances and data stored in their Amazon S3 buckets.

In addition, this scenario can be useful when the users need to access the AWS Management Console, launch new services, interact with services for complicated cloud-based application architectures, or exercise more control over accessing and sharing resources. Figure 2 shows the architecture for this scenario.

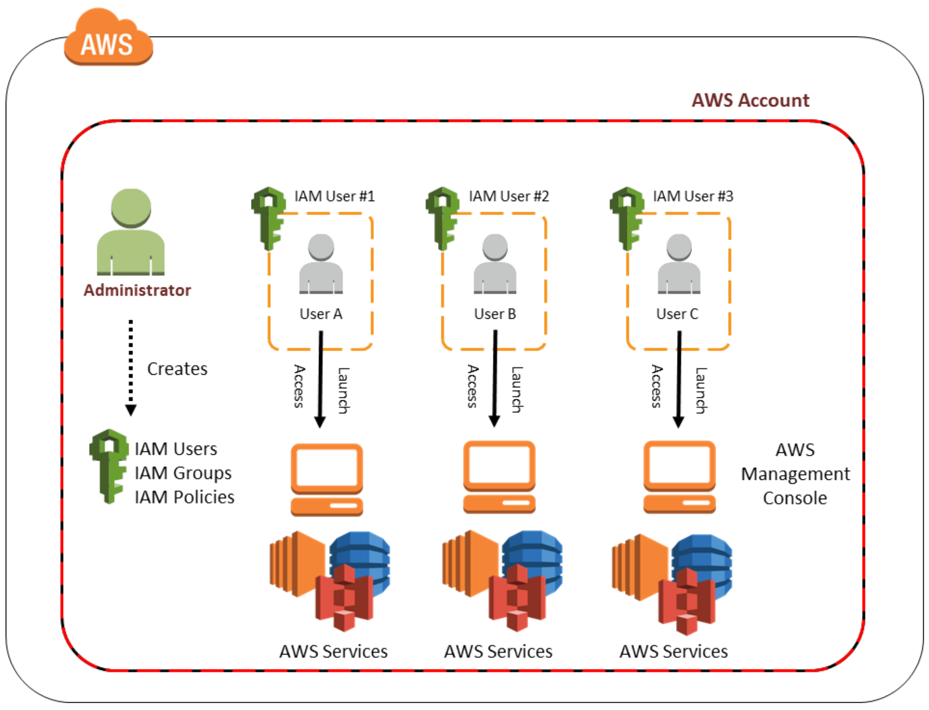


Figure 2: Limited User Access to AWS Management Console

As shown in Figure 2, this scenario works well with a single AWS account. The administrator needs to create IAM users and groups to apply access control policies for the environment. Example IAM user policies for setting up this scenario are presented in "Appendix B: Example IAM User Policies" of this whitepaper.

Account Setup

The administrator creates one AWS account for the group. For example, this can be a shared account for a professor, class, department, or school. An existing AWS account can also be used. New AWS account signup and access to existing AWS accounts is available at Your Account.

The administrator then creates an IAM user for each user with the AWS Management Console or the API. These IAM users can belong to one or more IAM groups within a single AWS account.

Based on environment requirements, the administrator attaches custom policies to IAM users or IAM groups to restrict certain AWS resources that can be launched and used. Thus users can only launch AWS services for which permissions have been granted. Users are provided credentials for their IAM user, which can be used to log into the AWS Management Console, access AWS services, and call APIs.

Information Required for Account Setup

To create an account and set up IAM-based access control, an administrator will need the following information:

- AWS account for the group. This account could belong to the school, department, or professor. If no account exists, a new account must be created.
- Name and email address of users.
- Required AWS resources and services and the operations permitted on them. This is required to determine the access control policies to be applied to each IAM user.
- Billing alerts preferences: Who receives warning emails?
- Usage reports preferences: Who receives this information?

Providing Access to Users

The administrator adds IAM users with roles and custom policies to the AWS account directly to implement required access control logic for the different kinds of users in the group. The administrator then provides IAM user login info to the corresponding members of the group. For basic instructions on how to add IAM user policies, see "Appendix A: Adding IAM User Policies" of this whitepaper. Example IAM user policies for setting up this scenario are presented in "Appendix B: Example IAM User Policies" of this whitepaper.

Cost Tracking

All users can tag their resources for services with tagging capability. With the cost allocation feature of AWS Account Billing, the administrator can track AWS costs for each user. For more information, see Cost Allocation and Tagging in the AWS Account Billing documentation.

Monitoring Resources

Amazon CloudWatch alarms and billing alerts can help monitor AWS resources. Billing alerts automatically notify users whenever the estimated charges on their current AWS bill reach a threshold they define. Users can choose to receive an alert on their total AWS charges or charges for a specific AWS product or service. If the account has any limits, the administrator can use these as the threshold for sending billing alerts. For more information about setting up billing alerts with CloudWatch, see Monitor Your Estimated Charges Using Amazon CloudWatch in the Amazon CloudWatch Developer Guide.

Reporting

Detailed usage reports for the AWS environment are available for the administrator. Reports are available for monthly charges and also for account activity in hourly increments. For more information, see Detailed Reports in the AWS Account Billing documentation.

Runtime Environment

Users can log into the AWS Management Console as an IAM user with the login information provided to them by the administrator. They can launch and use resources defined by the rules and policies set by the administrator. For example, if they have the appropriate permissions, they can launch new Amazon EC2 instances or create new Amazon S3 buckets, upload data to them, and share them with others.

An IAM user might be granted access to create a resource, but the user's permissions, even for that resource, are limited to what's been explicitly granted by the administrator. The administrator can also revoke the user's permission at any time. Setting proper resource and user-based permissions helps prevent an IAM user from taking actions on resources belonging to other IAM users in the AWS account. For example, an IAM user can be prevented from terminating instances belonging to other IAM users in the AWS account. For more information, see Overview of AWS IAM Permissions in the AWS Identity and Access Management documentation.

Environment Termination

When users have finished their work or when the account limits are reached, they (or the administrator) can terminate the AWS services. Administrators can also delete the IAM users. The users will lose their work unless they take steps to save it (a procedure that is beyond the scope of this whitepaper).

Setting Up Scenario 3: Separate AWS Account for Each User

In the “Separate AWS Account for Each User” scenario, an administrator creates separate AWS accounts for each user who needs a new AWS account. These accounts can optionally be linked together and a single AWS account can be designated as the paying account using consolidated billing, which provides a single bill for multiple AWS accounts. The administrator then creates an IAM user in each AWS account and applies an access control policy to each user. Users are given access to the IAM user within their AWS account, but do not have access to the root credentials of the AWS account. This allows the accounts to be managed by an administrator consistent with the required policies for the user environment.

Users can log into the AWS Management Console with their IAM credentials and then they can launch and access different AWS services, subject to the access control policies applied to their account. Users have direct control over the access credentials for their resources and they can also share these resources with other users as necessary.

This scenario is good for setting up collaborative multiuser work environments. To implement this, users can create an IAM role, which is an entity that includes permissions but that isn't associated with a specific user. Users from other accounts can then *assume* the role and access resources according to the permissions assigned to the role. For more information, see Cross-Account API Access Using IAM Roles in the AWS Identity and Access Management documentation.

This scenario offers maximum flexibility for the users and is helpful when they need to access the AWS Management Console to launch new services. This scenario also gives users flexibility in working with complicated cloud-based application architectures and more control over accessing and sharing their resources.

Having separate AWS accounts for each user works well for both short-term and long-term usage. For short-term usage, AWS resources, IAM users, and even AWS accounts can be terminated after the work is done. For long-term usage, the AWS accounts for some or all users are kept alive at the end of the current engagement. All work done can be easily preserved for future use. Users can also be provided full administrator access to their AWS account (besides the IAM-based access they initially had) to continue their work. An example scenario is an entrepreneurship class where some students might develop some new solutions or intellectual property using AWS resources that they want to retain for future use or for immediate deployment. Their work can be easily turned over to them by giving them full access to their AWS account.

Another benefit of this scenario is that in the AWS Management Console, users cannot see resources belonging to any other users in the group, since each user is working from their own AWS account. Figure 3 shows the architecture for this scenario.

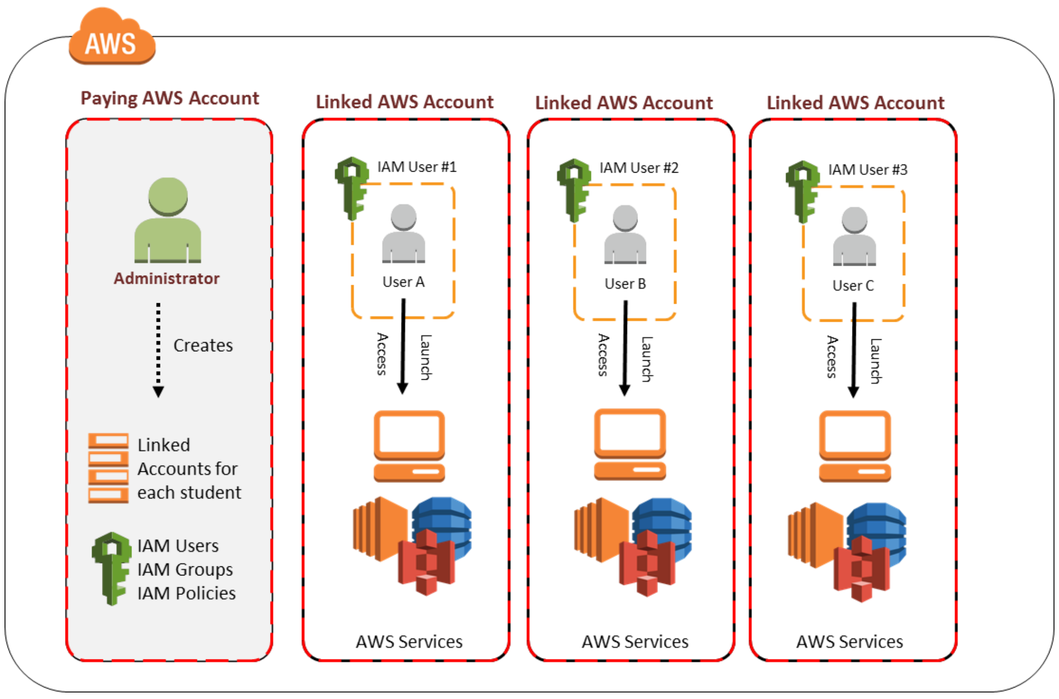


Figure 3: Separate AWS Account for Each User

Account Setup

The administrator provisions an AWS account for each user in the group. Independent AWS accounts (with unique AWS IDs) are created for each user. New AWS account signup and access to existing AWS accounts is available at Your Account.

The administrator then creates an IAM user in each user's AWS account. Based on environment requirements, custom policies are attached to IAM users to constrain AWS resources that can be launched and used. Thus users can only launch AWS services for which permissions have been granted. Users are provided credentials for their IAM user, which can be used to log into the AWS Management Console, access AWS services, and call APIs. Users do not have access to the root credentials of the AWS account and cannot change the IAM access policies enforced on the account.

The administrator can also set up consolidated billing for the group. Consolidated billing (offered free by AWS) enables consolidation of payment for multiple AWS accounts by designating a single paying account. Consolidated billing provides a combined view of AWS charges incurred by all accounts as well as a detailed cost report for each individual AWS account associated with the paying account. For detailed information about how to set up consolidated billing, see Consolidated Billing in the AWS Account Billing documentation.

Information Required for Account Setup

The following information is required for creating accounts and setting up IAM-based access control:

- AWS account for the group. This account could belong to the school, department, or professor. If no account exists, a new account is created. This account is necessary for setting up consolidated billing.
- Name and email addresses of users.
- AWS account credentials for users who have existing AWS accounts that they want to use in this environment. Users who do not have an AWS account or do not want to use their existing account will need new accounts provisioned for them.
- Required AWS resources and services and the operations permitted on them. This is required to determine the access control policies to be applied to each IAM user.
- Billing alerts preferences: Who receives warning emails?
- Usage reports preferences: Who receives this information?

Providing Access to Users

An administrator retains security credentials for the root AWS account of each user. The administrator adds IAM users with roles and custom policies for each user's AWS account to implement required access control logic for the different types of users in the group. Login information for the IAM users is provided to the corresponding users in the group. For basic instructions on how to add IAM user policies, see "Appendix A: Adding IAM User Policies" of this whitepaper. Example IAM user policies for setting up this scenario are presented in "Appendix B: Example IAM User Policies" of this whitepaper.

Cost Tracking

Consolidated billing makes it easy to track AWS costs since it shows the administrator a combined view of charges incurred by all AWS accounts as well as a detailed cost report for each individual AWS account. Administrators can sign up for consolidated billing at Your Account after logging in.

All users can also tag their resources for services with tagging capability. An administrator can then use the cost allocation feature of AWS Account Billing to track AWS costs for each user. For more information, see Cost Allocation and Tagging in the AWS Account Billing documentation.

Monitoring Resources

Amazon CloudWatch alarms and billing alerts can help monitor AWS resources. Billing alerts automatically notify users whenever the estimated charges on their current AWS bill reach a threshold they define. Users can choose to receive an alert on their total AWS charges or charges for a specific AWS product or service. If the account has any limits, the administrator can use these as the threshold for sending billing alerts. For more information about setting up billing alerts with CloudWatch, see Monitor Your Estimated Charges Using Amazon CloudWatch in the Amazon CloudWatch Developer Guide.

Reporting

Detailed usage reports are available for the administrator from the AWS Management Console. Reports are available for monthly charges and also for account activity in hourly increments. For more information, see Detailed Reports in the AWS Account Billing documentation.

Runtime Environment

Users can log into the AWS Management Console as an IAM user with the login information provided to them by the administrator. They can launch and use resources defined by the rules and policies set by the administrator. For example, if they have the appropriate permissions, users can launch new Amazon EC2 instances or create new Amazon S3 buckets, upload data to them, and share them with others. Since accounts are independent, each user sees only their own AWS resources in the AWS Management Console.

Environment Termination

When the users have finished their work or when the account limits are reached, they (or the administrator) can optionally terminate the AWS services. The administrator can also delete the IAM users.

Keeping Accounts Alive

If the users want to retain their AWS accounts, they can request the root account credentials from their administrator. The administrator would remove their account from consolidated billing and users would get login and security credentials to their AWS account.

Conclusion

Multiuser, shared environments with custom access control policies are a common use case for AWS customers. Typical requirements include both user and resource management to allow controlled access to AWS resources for multiple users. This whitepaper presented three scenarios that covered a wide array of use cases with these requirements. The “Individual Server Environments” scenario provides access to customized work environments on AWS and is suitable for use cases like undergraduate labs. The “Limited User Access to AWS Management Console” scenario provides IAM user access to users from a single AWS account suitable for use cases like graduate classes. The “Separate AWS Account for Each User” scenario provides independent AWS accounts for each user (optionally linked with consolidate billing), which is suitable for graduate research and entrepreneurship courses. In this whitepaper, we focused on the short to medium-term education and research environments as the example domain, but the same or similar scenarios may also be implemented for other use cases.

Further Reading

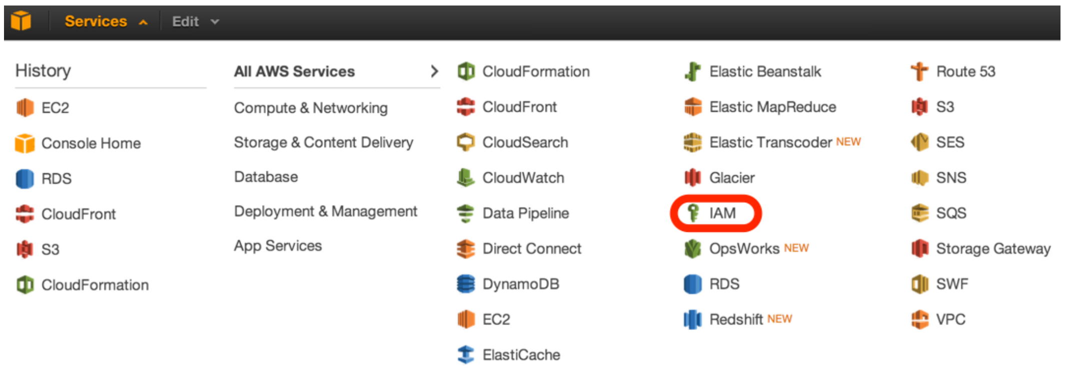
In addition to this whitepaper, we recommend that you consult the following documents and websites:

- IAM documentation
- IAM Policies for Amazon EC2
- Granting IAM Users Required Permissions for Amazon EC2 resources
- Amazon Resource Names (ARNs) and Amazon Service Namespaces

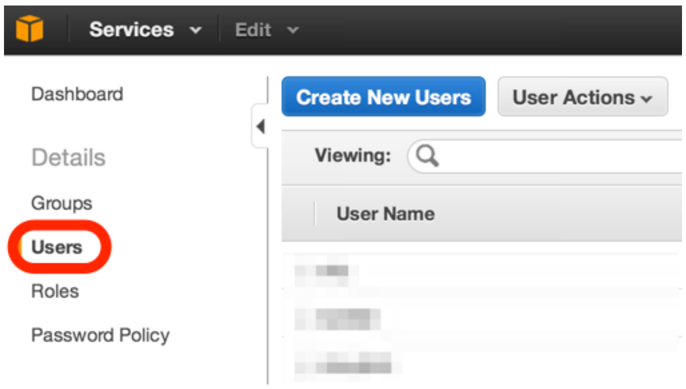
Appendix A: Adding IAM User Policies

This section describes how to add IAM user policies to an AWS account. For more information, see Adding an IAM User to Your AWS Account.

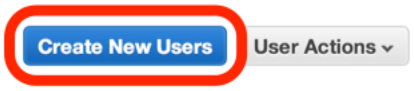
1. In the AWS Management Console, open the list of services, and then click **IAM**.



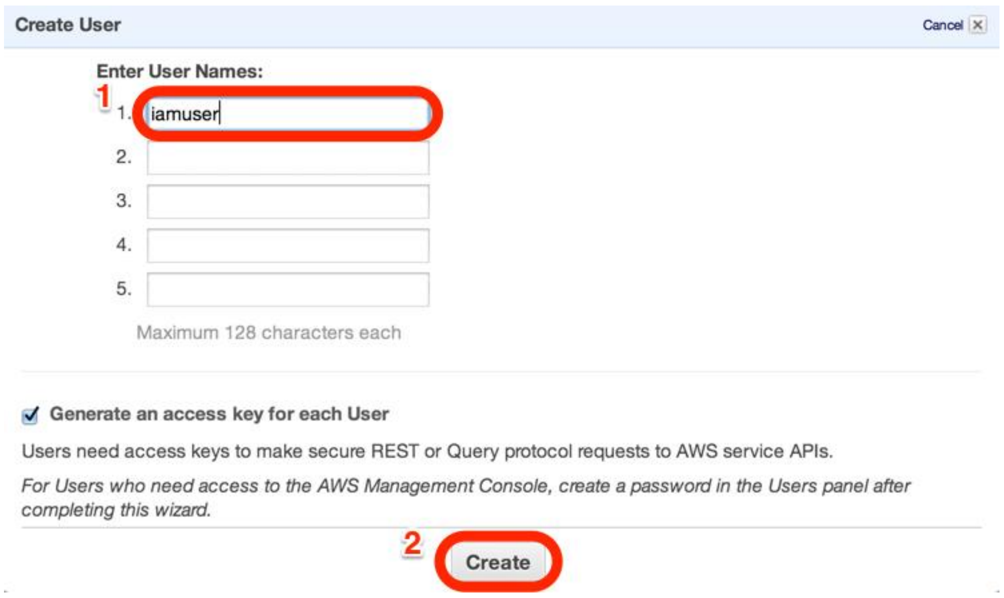
2. Click **Users**.



3. Click **Create New Users**.



4. Type in the name of the IAM user to be created, and then click **Create**.

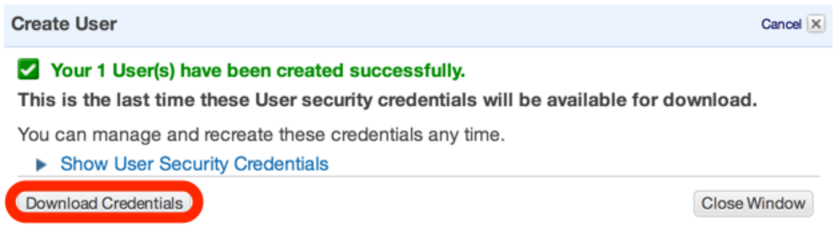


✓ Generate an access key for each User

Users need access keys to make secure REST or Query protocol requests to AWS service APIs.

For Users who need access to the AWS Management Console, create a password in the Users panel after completing this wizard.

5. Click **Download Credentials**. Save the downloaded file in a secure location, as these are the user's access key id and secret access key. They will need these to use the AWS API.

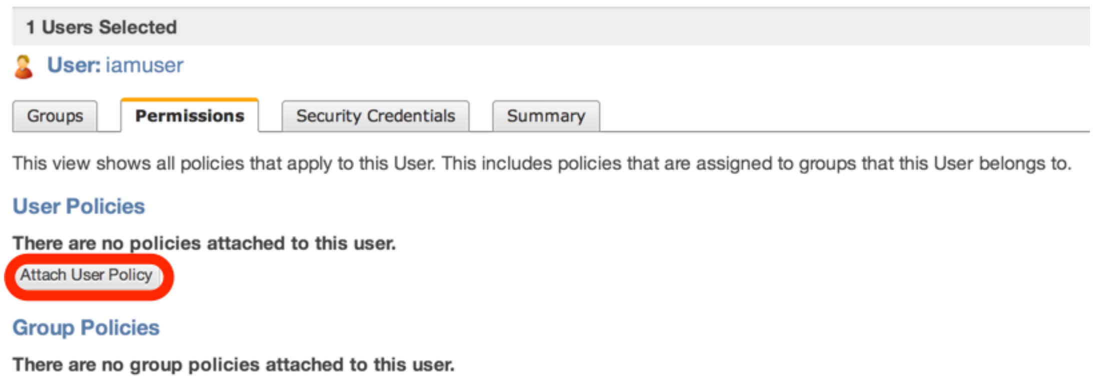


✓ Your 1 User(s) have been created successfully.

This is the last time these User security credentials will be available for download.

You can manage and recreate these credentials any time.

6. Click **Attach User Policy**.


1 Users Selected

User: iamuser

This view shows all policies that apply to this User. This includes policies that are assigned to groups that this User belongs to.

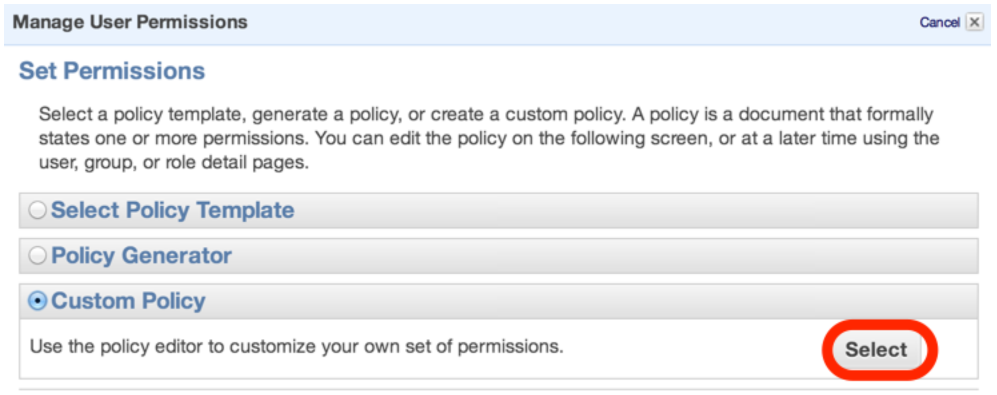
User Policies

There are no policies attached to this user.

Group Policies

There are no group policies attached to this user.

7. Select **Custom Policy** and then click **Select**.


Manage User Permissions

Set Permissions

Select a policy template, generate a policy, or create a custom policy. A policy is a document that formally states one or more permissions. You can edit the policy on the following screen, or at a later time using the user, group, or role detail pages.

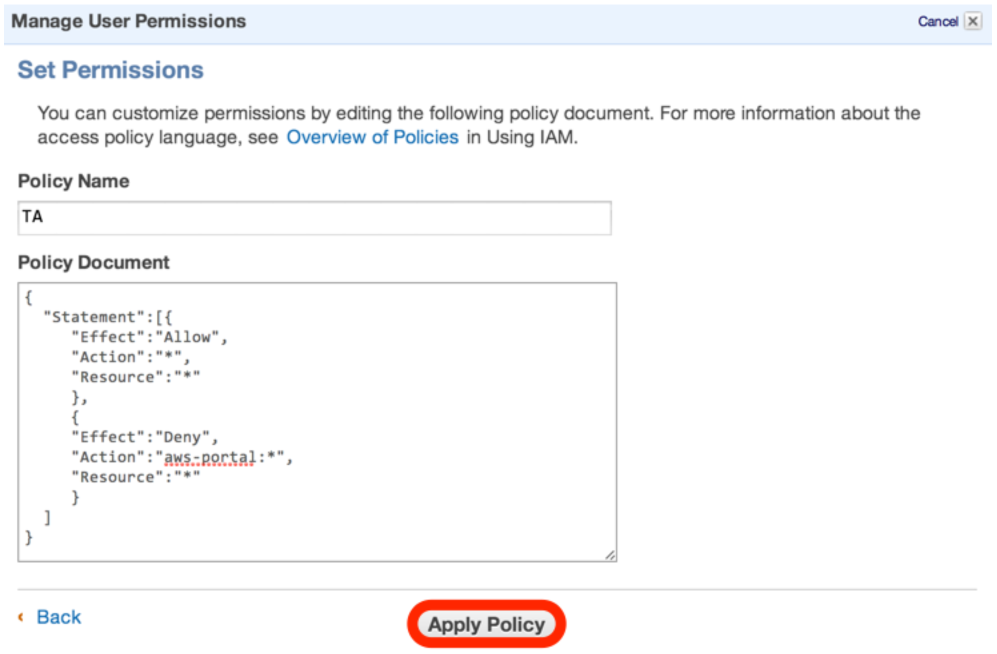
○ Select Policy Template

○ Policy Generator

[✓ Custom Policy]

Use the policy editor to customize your own set of permissions.

8. Paste the proper policy from "Appendix B: Example IAM User Policies" and then click **Apply Policy**.


Manage User Permissions

Set Permissions

You can customize permissions by editing the following policy document. For more information about the access policy language, see Overview of Policies in Using IAM.

Policy Name

TA

Policy Document

```
{
  "Statement":[{
     "Effect":"Allow",
     "Action":"*",
     "Resource":"*"
     },
     {
     "Effect":"Deny",
     "Action":"aws-portal:*",
     "Resource":"*"
     }
  ]
}
```

Appendix B: Example IAM User Policies

This section provides example IAM user policies for a class that uses AWS services, including policies for the professor, teaching assistant, and students. These policies are useful for setting up the "Limited User Access to AWS Management Console" and "Separate AWS Account for Each User" scenarios described earlier in this whitepaper. For more information about policies, see Overview of AWS IAM Policies in the AWS Identity and Access Management documentation.

Example Policies for Professor

1. Full administrator access:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

2. Billing access:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "aws-portal:ViewBilling"
      ],
      "Resource": "*"
    }
  ]
}
```

3. Usage access:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "aws-portal:ViewUsage"
      ],
      "Resource": "*"
    }
  ]
}
```

Example Policies for Teaching Assistant

4. Full administrator access but no access for billing or usage information:

```
{
  "Statement":[{
     "Effect":"Allow",
     "Action":"*",
     "Resource":"*"
     },
     {
     "Effect":"Deny",
     "Action":"aws-portal:*",
     "Resource":"*"
     }
  ]
}
```

Example Policies for Students

5. Permission to create and describe Amazon EBS volumes:

```
{
  "Statement": [
    {
      "Sid": "Stmt1364408175147",
      "Action": [
        "ec2:AttachVolume",
        "ec2:CopySnapshot",
        "ec2:CreateImage",
        "ec2:CreateSnapshot",
        "ec2:CreateVolume",
        "ec2:DescribeSnapshots",
        "ec2:DescribeVolumeAttribute",
        "ec2:DescribeVolumeStatus",
        "ec2:DescribeVolumes",
        "ec2:EnableVolumeIO"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    }
  ]
}
```

6. Permission to create and modify Amazon EC2 instances:

```
{
  "Statement": [
    {
      "Sid": "Stmt1364408055009",
      "Action": [
        "ec2:DescribeAddresses",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeImageAttribute",
        "ec2:DescribeImages",
        "ec2:DescribeInstanceAttribute",
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeInstances",
        "ec2:DescribeRegions",
        "ec2:DescribeVolumes",
        "ec2:MonitorInstances",
        "ec2:ReportInstanceStatus",
        "ec2:RunInstances",
        "ec2:StartInstances"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    }
  ]
}
```

7. Permission to create and modify Auto Scaling groups:

```
    {
      "Statement": [
        {
          "Sid": "Stmt1364407742584",
          "Action": [
            "autoscaling:CreateAutoScalingGroup",
            "autoscaling:CreateLaunchConfiguration",
            "autoscaling:CreateOrUpdateScalingTrigger",
            "autoscaling:CreateOrUpdateTags",
            "autoscaling:DescribeAdjustmentTypes",
            "autoscaling:DescribeAutoScalingGroups",
            "autoscaling:DescribeAutoScalingInstances",
            "autoscaling:DescribeAutoScalingNotificationTypes",
            "autoscaling:DescribeLaunchConfigurations",
            "autoscaling:DescribeMetricCollectionTypes",
            "autoscaling:DescribeNotificationConfigurations",
            "autoscaling:DescribePolicies",
            "autoscaling:DescribeScalingActivities",
            "autoscaling:DescribeScalingProcessTypes",
            "autoscaling:DescribeScheduledActions",
            "autoscaling:DescribeTags",
            "autoscaling:DescribeTriggers",
            "autoscaling:EnableMetricsCollection",
            "autoscaling:ExecutePolicy",
            "autoscaling:PutNotificationConfiguration",
            "autoscaling:PutScalingPolicy",
            "autoscaling:PutScheduledUpdateGroupAction",
            "autoscaling:ResumeProcesses",
            "autoscaling:SetDesiredCapacity",
            "autoscaling:SetInstanceHealth",
            "autoscaling:SuspendProcesses",
            "autoscaling:TerminateInstanceInAutoScalingGroup",
            "autoscaling:UpdateAutoScalingGroup"
          ],
          "Effect": "Allow",
          "Resource": [
            "*"
          ]
        }
      ]
    }
```

8. Permission to create or modify Elastic Load Balancing:

```
{
  "Statement": [
    {
      "Sid": "Stmt1364408268680",
      "Action": [
        "elasticloadbalancing:ConfigureHealthCheck",
        "elasticloadbalancing:CreateAppCookieStickinessPolicy",
        "elasticloadbalancing:CreateLBCookieStickinessPolicy",
        "elasticloadbalancing:CreateLoadBalancer",
        "elasticloadbalancing:CreateLoadBalancerListeners",
        "elasticloadbalancing:DeleteLoadBalancerListeners",
        "elasticloadbalancing:DeleteLoadBalancerPolicy",
        "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
        "elasticloadbalancing:DescribeInstanceHealth",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DisableAvailabilityZonesForLoadBalancer",
        "elasticloadbalancing:EnableAvailabilityZonesForLoadBalancer",
        "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
        "elasticloadbalancing:SetLoadBalancerListenerSSLCertificate",
        "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    }
  ]
}
```

9. Prevents modifying resource tags:

```
{
 "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:CreateTags",
        "ec2:DeleteTags"
      ],
      "Sid": "Stmt1375637892000",
      "Resource": [
        "*"
      ],
      "Effect": "Deny"
    }
  ]
}
```

10. For instances with a student tag, allows students to restart, stop, reboot, attach volumes, and detach volumes:

If the professor or teaching assistant applies a student tag with the value being the IAM user name of specific students to specific instances, then those students can stop, reboot, attach volumes to, and detach volumes to those instances. They can also start instances that they stopped (that still have the student tag on them), but they can't start new ones.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances",
        "ec2:RebootInstances",
        "ec2:AttachVolume",
        "ec2:DetachVolume"
      ],
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/Student":"${aws:username}"
        }
      },
      "Resource": [
        "arn:aws:ec2:region:account:instance/*",
        "arn:aws:ec2:region:account:volume/*"
      ],
      "Effect": "Allow"
    }
  ]
}
```