Getting Started with Penn Single Sign On

The AWS platform provides a number of different methods to manage identities and to control access to the underlying services that Amazon offers. The method that is likely to offer the lowest risk and present the least effort for most users of AWS is to take advantage of our existing integration between Penn's existing SAML-based single-sign-on identity provider (IDP), the Penn Groups service, and the AWS platform.

When this integration is used, identities are not defined directly within the AWS IAM service, but are provided to AWS by the IDP at the time of sign-in. Each user that signs in via this method uses the AWS Service Token Service (STS) to assume a specific AWS IAM role. The mapping of roles to specific group memberships in PennGroups follows a naming convention.

There is some initial configuration required to get an AWS account configured to trust Penn's SAML IDP as the identity provider for access to services and resources being used in that account. Accounts created through the AWS @ Penn service offering are provisioned with this trust relationship in place, and accounts migrated into the service offering can have that trust relationship established by running a CloudFormation template in the account. Assuming that this trust relationship already exists in the account that you're using, you can proceed with the directions below.

AWS Tools

There are three primary tools you will likely use to interact with AWS services:

1. Each service's specific user interface, signing in via the AWS Management Console
2. Command line interface (CLI)
3. Directly through the AWS APIs or via the language-specific Software Development Kits (SDKs)

Which tool or tools you use will depend on what you're trying to accomplish. In general, the Management Console is a good place to start if you would like to view what resources have been created within a given AWS service, or to play around with specific services. To pull down data or interact with the services from within a terminal in Mac or Windows or Linux, the CLI can be helpful. Finally, if you're planning to develop an application or a script to automate some process, the SDKs can simplify the work of interacting with the underlying AWS APIs.

Sign into the AWS Management Console

Navigate to https://aws.cloud.upenn.edu in your preferred browser. You may want to add the link above to your browser bookmarks. Each time you navigate to it, your browser will be redirected, first to the WebLogin screen, if you are not yet authenticated, and then, once you are

authenticated, to a landing page where you can identify what role you want to use to access your AWS account. Initially, you may only have one role, in which case you will be forwarded directly to the AWS Management Console. If you have multiple roles, you will be presented with a screen like the one below and prompted to select the role you would like to sign in with.

Figure 1: Example of single-sign-on AWS account role selection

Select the radio button for your role, as shown above, and then click on the button labelled “Sign In” at the bottom of the screen.

You should now be presented with the AWS Management Console, which should look something like the picture shown below.

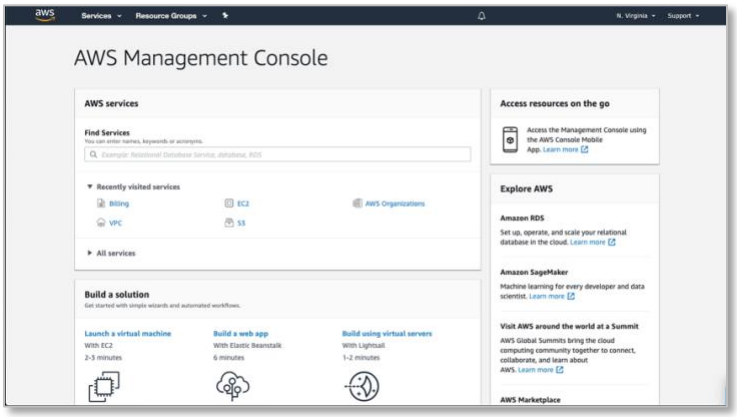

Figure 2: Screenshot of the AWS Management Console

The AWS Management Console provides you with a user interface to explore all of the various cloud services that AWS provides. These services are grouped into different categories such as Compute, Storage, Database, etc.

To navigate between services, you can begin typing in the search box labelled Find Services. For example, you can type “VPC” if you want to navigate to the VPC service to create a new virtual machine. Another way to navigate between services is by selecting the drop down at the top left of your screen labelled “Services” and then picking the service listed there.

More directions on using the AWS Management Console can be found in the AWS documentation, by navigating to the link below.

https://docs.aws.amazon.com/awsconsolehelpdocs/latest/gsg/getting-started.html

In this case, you will want to navigate first to the Service Catalog, where you will find a list of Products that you can use to streamline the creation of a new VPC.

Authenticating for CLI and/or SDK use

To authenticate to the Penn IDP, if you want to use the AWS CLI or to invoke the APIs via the SDKs using your own identity – for example, if you are a developer – you will need to have at least two credentials: an AWS access key and a secret key.

It is possible to create users directly within the AWS IAM service and assign these credentials to those users on a long-term basis. The disadvantage of this approach is that it means that you are managing identities and incurring the responsibility to ensure that those identities are kept current and that access is removed when someone who has access changes positions within the University or separates from the University entirely.

To avoid managing these identities within AWS, our recommendation is that you take advantage of a tool that Wharton Computing has developed, building on an earlier Python script that ISC adapted from one that AWS provided. This tool handles the negotiation with the IDP and AWS and stores the two credentials above into a config file on your workstation. It also adds an additional credential, the session token, and the three credentials are used to provide shorter term access to the same services and resources you would be able to access through the AWS Management Console if you were to sign in as described above.

Installing the AWS Command Line Interface (CLI)

We recommend using version 2 of the AWS CLI. Operating system specific directions for installing are available from Amazon under the general AWS Command Line Interface documentation in the section Installing the AWS CLI version 2.

You will want to install and configure the CLI prior to installing the Penn-specific tool below because that will create the appropriate directories and files on your workstation.

Installing the aws-federated-auth tool

The aws-federated-auth tool is a Python program that can be installed on your workstation directly from the source control repository where it is hosted on Atlassian BitBucket. You will

first need to be granted access to that repository and the repositories where its dependencies live. Please contact the ISC Cloud Solutions team to request that access.

Once access has been granted, you will need to provide BitBucket with your public SSH key in order to be able to clone the repository using Git. Directions on generating an SSH key are available through the Atlassian directions. On the same page, Atlassian also provides directions on uploading your public SSH key to BitBucket settings. This will allow you to clone BitBucket repositories onto your workstation, including the one containing the aws-federated-auth tool.

To bring the latest version of the aws-federated-auth code down to your workstation and install it, execute the following commands. Note that you will need to have Python 3 installed.

```
> git clone git@bitbucket.org:codeforpenn/aws-federated-auth.git
> cd aws-federated-auth
> pip3 install -r requirements.txt
```

Once you have run those commands, you will now be able to use the aws-federated-auth tool to download short-term credentials from AWS.

Using the aws-federated-auth tool

For example, to display all of the AWS accounts that you have access to, you can run this command:

```
> python3 aws-federated-auth.py --list
```

You will then be prompted for a username and password. These are the credentials you would normally use to sign into WebLogin. Once you have provided both, you will be prompted for your Duo passcode. If you have configured your multifactor authentication to send you a push on your mobile device, you should receive that notification at this point, and once you've approved it, you can simply hit return.

You can also run this command to download short-term credentials for all of the AWS accounts and roles that you have been granted access to:

```
> python3 aws-federated-auth.py
```

Running the command above will download those credentials into the standard AWS CLI credentials file, located on Mac OS X and Linux systems under ~/.aws/credentials – this file follows a format that AWS defines to store multiple credentials so that it's possible to switch between those credentials using the environment variable AWS_PROFILE.

You can also specify the account that you're interested in filtering access to:

```
> python3 aws-federated-auth.py --account 123456789012
```

For example, the ~/.aws/credentials file might have the following entries after you've run the aws-federated-auth.py command:

```
[aws-example1-PennAccountAdministrator]
output = json
region = us-east-1
aws_access_key_id = [ REDACTED ]
aws_secret_access_key = [ REDACTED ]
aws_session_token = [ REDACTED ]

[aws-example2-PennSuperUser]
```

To use the credentials tied to the first account / role combo with the AWS CLI, you can set your AWS_PROFILE environment variable to that value, for example:

```
> export AWS_PROFILE=aws-example1-PennAccountAdministrator
```

You may want to create an alias using your preferred shell, for example, using Bash shell, you can add something like the following to your ~/.bash_profile

```
alias aws-login="python3 /Users/testuser/Code/aws-federated-auth/aws-federated-auth.py --
account 123456789012 --user testuser;export AWS_PROFILE=aws-example1-
```

Using the AWS CLI

Once you have authenticated to a particular account and role, and defined its identifier as your AWS Profile, as outlined above, you can begin using the AWS CLI as you normally would.

For example, assuming you have the right permissions to see EC2 instances in the given account

with the given role, and assuming you have created an alias as described above, you could execute the following commands:

```
$ aws-login
Password: ********
Enter Duo passcode, or 'return' for approved push request:

PROFILE NAME                    MAX DURATION ACCOUNT NUMBER ROLE NAME
aws-example1-PennAccountAdministrator      3600 422318320378   PennAccountAdministrator
$ aws ec2 describe-instances
{
    "Reservations": [
        {
```