SDK & Toolkit Configuration

Development Immersion Day

Team or presenters name

Date

Agenda

- Prerequisites
- Installation
- Credentials
- Configuration
- Putting It All Together

Prerequisites

All AWS APIs Are REST based

Create canonical request

Create a hash of request, append to the request

Create a string to sign from

Derive a signing key with recursive keyed hash operations

Create the request signature

Add signature to the header

Enter the SDKs

- Handles all REST call details
- Handles access key management (store in config files or EC2 Roles)
- Exposes all services as an object oriented API
- In addition to simple retries, each AWS SDK implements exponential backoff algorithm for better flow control.

```
 // Print the number of Amazon S3 Buckets.

IAmazonS3 s3Client = AWSClientFactory.CreateAmazonS3Client();

try
{
      ListBucketsResponse response = s3Client.ListBuckets();
      int numBuckets = 0;
      if (response.Buckets != null &&  response.Buckets.Count
> 0)
      {
           numBuckets = response.Buckets.Count;
       }
       sr.WriteLine("You have " + numBuckets + " Amazon S3
bucket(s).");
}
```

AWS SDK for .NET

- The AWS SDK for .NET is installed with the AWS Toolkit for Visual Studio, a plugin that provides a user interface for managing your AWS resources from Visual Studio, and also includes the AWS Tools for Windows PowerShell.
- Requirements:
    - Microsoft .NET Framework 3.5 or later
- Note:
    - We recommend using Visual Studio Professional 2010 or later to implement your applications.

AWS Toolkit for Visual Studio

- The Toolkit for Visual Studio is a plugin for the Visual Studio IDE that makes it easier for you to develop, debug, and deploy .NET applications that use Amazon Web Services.
- Requirements:
    - The Toolkit for Visual Studio is supported for Visual Studio versions 2013 and later
- Note:
    - The Toolkit for Visual Studio is also available for Visual Studio 2008, 2010, and 2012 versions. However, those versions are not supported.

AWS Tools for Visual Studio Team Services

- The AWS Tools for Microsoft Visual Studio Team Services (VSTS) adds tasks to easily enable build and release pipelines in VSTS and Team Foundation Server to work with AWS services
- Requirements:
    - Visual Studio Team Services
    - Team Foundation Server 2015 Update 3 (or higher)

AWS Command Line Interface

- The AWS Command Line Interface (CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.
- Requirements:
    - Linux: You can install the AWS Command Line Interface and its dependencies on most Linux distributions withpip, a package manager for Python.
    - macOS: Python 2 version 2.6.5+ or Python 3 version 3.3+
    - Windows: Microsoft Windows XP or later

AWS Tools for PowerShell

- The AWS Tools for Windows PowerShell and AWS Tools for PowerShell Core are PowerShell modules that are built on the functionality exposed by the AWS SDK for .NET.
- Requirements:
    - Windows: Windows PowerShell 2.0 or newer (Microsoft PowerShell Core 6.0 or newer if you are installing the AWS Tools for PowerShell Core)
    - Linux/macOS: Microsoft PowerShell Core 6.0 or newer on a supported non-Windows system.

Installation

Installation

- AWS SDK for .NET
    - NuGet or MSI Installer
- AWS Toolkit for Visual Studio
    - VS 2019: Distributed in the Visual Studio Marketplace
    - VS 2017: Distributed in the Visual Studio Marketplace
    - VS 2013/2015: Part of the AWS Tools for Windows (MSI Installer)
- AWS Tools for Visual Studio Team Services
    - Visual Studio Marketplace

Installation

- AWS Command Line Interface
    - Windows: MSI Installer
    - macOS/Linux: Install using pip
- AWS Tools for PowerShell
    - MSI Installer or PowerShell Gallery

Credentials

Credential Introduction

To use the AWS SDKs, CLIs, and Toolkits, you need an AWS account and AWS credentials.

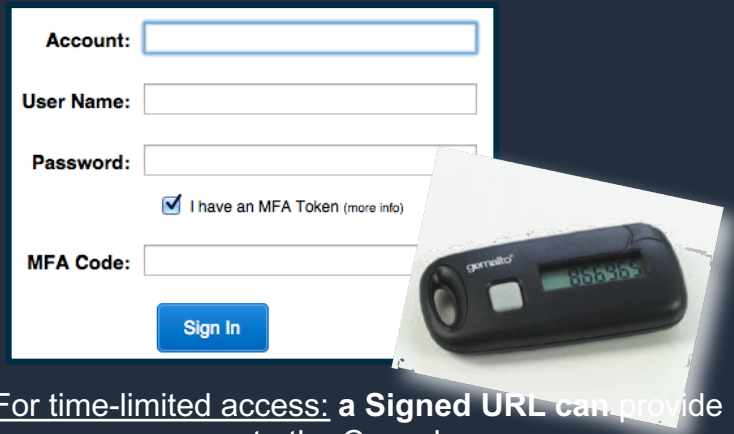
To increase the security of your AWS account, we recommend that you use an *IAM user* to provide access credentials instead of using your root account credentials.

AWS Identity Authentication

Authentication: *How do we know you are who you say you are?*

AWS Management Console

Login with **Username/Password** with optional **MFA** (recommended)



For time-limited access: **a Signed URL can** provide temporary access to the Console

API access

Access API using **Access Key + Secret Key**, with optional MFA

ACCESS KEY ID

Ex: `AKIAIOSFODNN7EXAMPLE`

SECRET KEY

Ex: `UtnFEMI/K7MDENG/bPxRfiCYE`

For time-limited access: Call the AWS Security Token Service (STS) to get a temporary AccessKey + SecretKey + session token

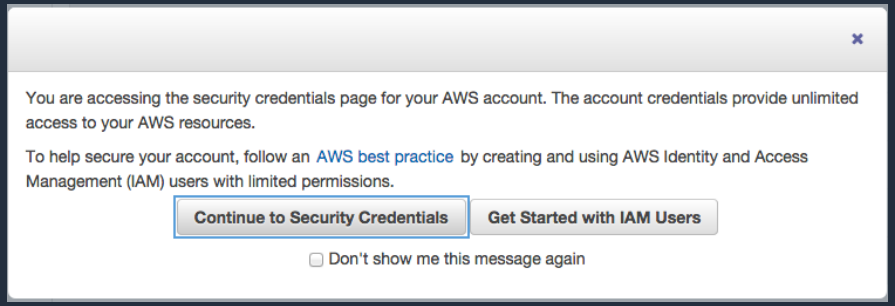
AWS Authorization and Privileges

Authorization: What are you allowed to do?

Account Owner (Root)

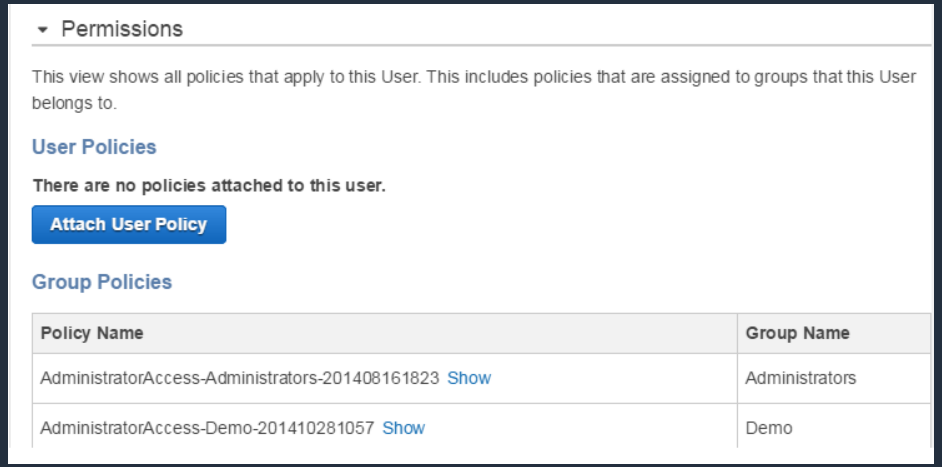
- Privileged for all actions.

Note: *Always associate the account owner ID with an MFA device and store it in a secured place!*



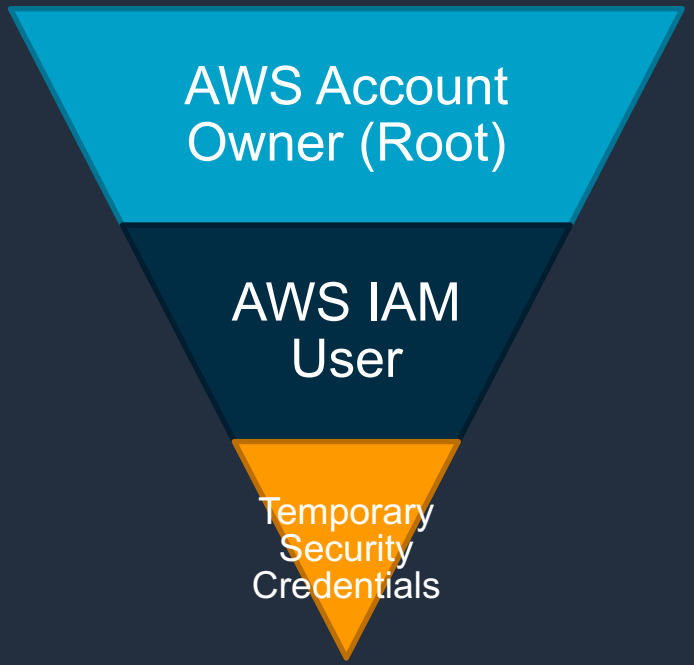
IAM Policies

- Privileges defined at User and Resource Level



AWS IAM Hierarchy of Privileges

Enforce principle of least privilege with Identity and Access Management (IAM) users, groups, and policies and temporary credentials.



Permissions	Example

Unrestricted access to all enabled services and resources.	`Action: *`
	`Effect: Allow`
	`Resource: *`
	`(implicit)`

Access restricted by Group and User policies	`Action:`
	`['s3:*','sts:Get*']`
	`Effect: Allow`
	`Resource: *`

Access restricted by generating identity and further by policies used to generate token	`Action: [ 's3:Get*' ]`
	`Effect: Allow`
	`Resource:`
	`'arn:aws:s3:::mybucket/*'`

aws

AWS Identity and Access Management (IAM)

Securely control access to AWS services and resources for your users.

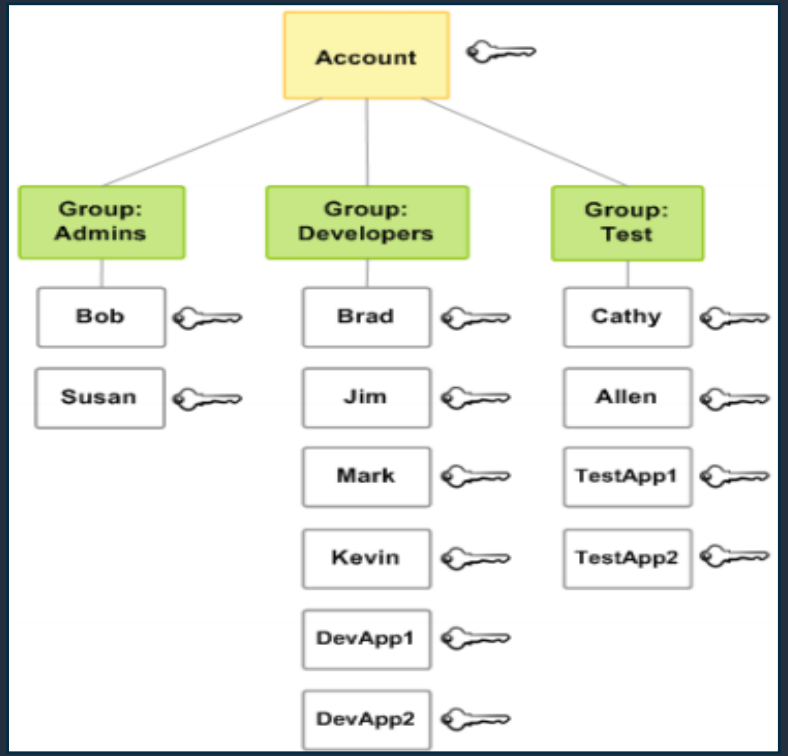
Username/ User

Manage groups of users

Centralized Access Control

Optional Configurations:

- Password for console access.
- Policies for controlling access AWS APIs.
- Two methods to sign API calls:
  - X.509 certificate
  - Access/Secret Keys
- Multi-factor Authentication (MFA)



aws

What to Use?

IAM User

- An IAM user has permanent long-term credentials and is used to directly interact with AWS services

IAM Role

- Allows applications running on EC2 instances or Lambda functions to call AWS services on your behalf
- Leverage temporary security credentials that control access to your AWS resources without having to hardcode access key credentials

IAM Policy Specification Basics

- ☑ JSON-formatted documents
- ☑ Contain a statement (permissions) that specifies:
  - Which actions a principal can perform
  - Which resources can be accessed

```
{
 "Statement":[{
  "Effect":"effect",
  "Principal":"principal",
  "Action":"action",
  "Resource":"arn",
  "Condition":{
   "condition":{
    "key":"value" }
   }
  }
  ]
 }
```

Principal
Action
Resource
Condition

You can have multiple statements and each statement is comprised of PARC.

aws

Configuration

Configuring Your AWS SDK for .NET Application

You can configure your AWS SDK for .NET application to specify AWS credentials, logging options, endpoints, or signature version 4 support with Amazon S3.

The recommended way to configure an application is to use the <aws> element in the project's App.config or Web.config file.

Another way to configure an application is to edit the <appSettings> element in the project's App.config or Web.config file.

Configuring Your AWS SDK for .NET Application – Sample

```
<configuration>
  <configSections>
    <section name="aws" type="Amazon.AWSSection, AWSSDK.Core"/>
  </configSections>
  <aws region="us-west-2">
    <logging logTo="Log4Net"/>
  </aws>
</configuration>
```

Configuring the AWS SDK for .NET with .NET Core – Sample

```
//Startup.cs
public Startup(IHostingEnvironment env)
{
    var builder = new ConfigurationBuilder()
        .SetBasePath(env.ContentRootPath)
        .AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)
        .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true)
        .AddEnvironmentVariables();
    Configuration = builder.Build();
}
//appsettings.Development.json
{
  "AWS": {
    "Profile": "local-test-profile",
    "Region": "us-west-2"
  }
}
```

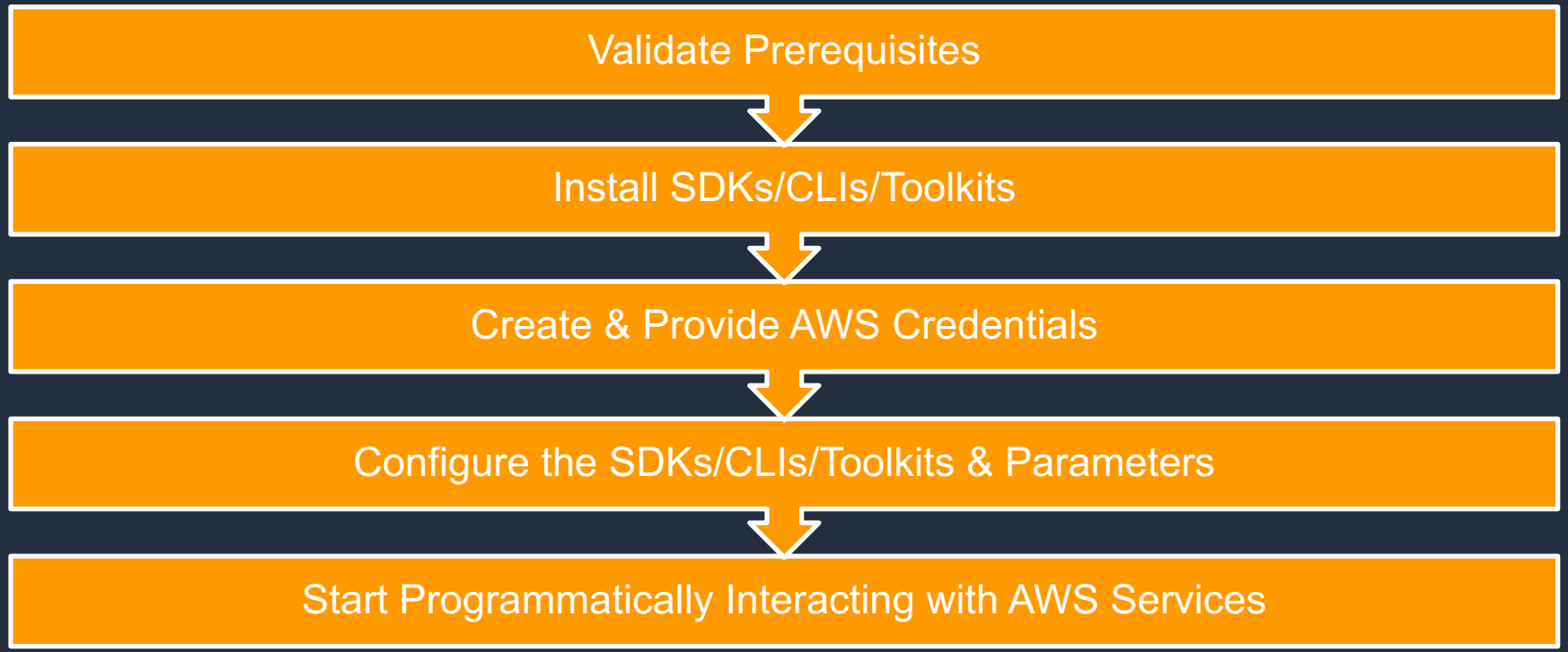
Configuring Other Application Parameters

In addition to configuring credentials, you can configure a number of other application parameters:

- AWSLogging
- AWSLogMetrics
- AWSRegion
- AWSResponseLogging
- AWS.DynamoDBContext.TableNamePrefix
- AWS.S3.UseSignatureVersion4
- AWSEndpointDefinition
- AWS Service-Generated Endpoints

Putting It All Together

In Summary



Questions