



AWS Academy Cloud Developing
Module 02 Student Guide
Version 2.0.0

200-ACCDEV-20-EN-SG

© 2020 Amazon Web Services, Inc. or its affiliates. All rights reserved.

This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.

All trademarks are the property of their owners.

Contents

[Module 2: Introduction to Developing on AWS](#)

4



Welcome to Module 2: Introduction to Developing on AWS

Module 2: Introduction to Developing on AWS

Section 1: Introduction

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Section 1: Introduction.

Module objectives



At the end of this module, you should be able to:

- Recognize the systems development lifecycle
- Describe how to get started developing on Amazon Web Services (AWS)
- Indicate how to work with AWS software development kits (SDKs)
- Identify the benefits of using the AWS Cloud9 integrated development environment (IDE)
- Develop and run a simple program in AWS Cloud9


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

5

At the end of this module, you should be able to:

- Recognize the systems development lifecycle
- Describe how to get started developing on Amazon Web Services (AWS)
- Indicate how to work with AWS software development kits (SDKs)
- Identify the benefits of using the AWS Cloud9 integrated development environment (IDE)
- Develop and run a simple program in AWS Cloud9

Module overview



Sections


1. Introduction
2. Systems development lifecycle
3. Steps to get started developing on AWS
4. Fundamentals of working with the AWS SDKs

Lab

- Exploring AWS CloudShell and the AWS Cloud9 IDE

Demonstration

- Installing the AWS CLI

 **Knowledge check**

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

4

This module includes the following sections:

1. Introduction
2. Systems development lifecycle
3. Steps to get started developing on AWS
4. Fundamentals of working with the AWS SDKs

This module also includes:

- A demonstration of installing the AWS CLI
- A lab where you learn how to work with AWS Cloud9

Finally, you will complete a knowledge check to test your understanding of key concepts covered in this module.

Café business requirement



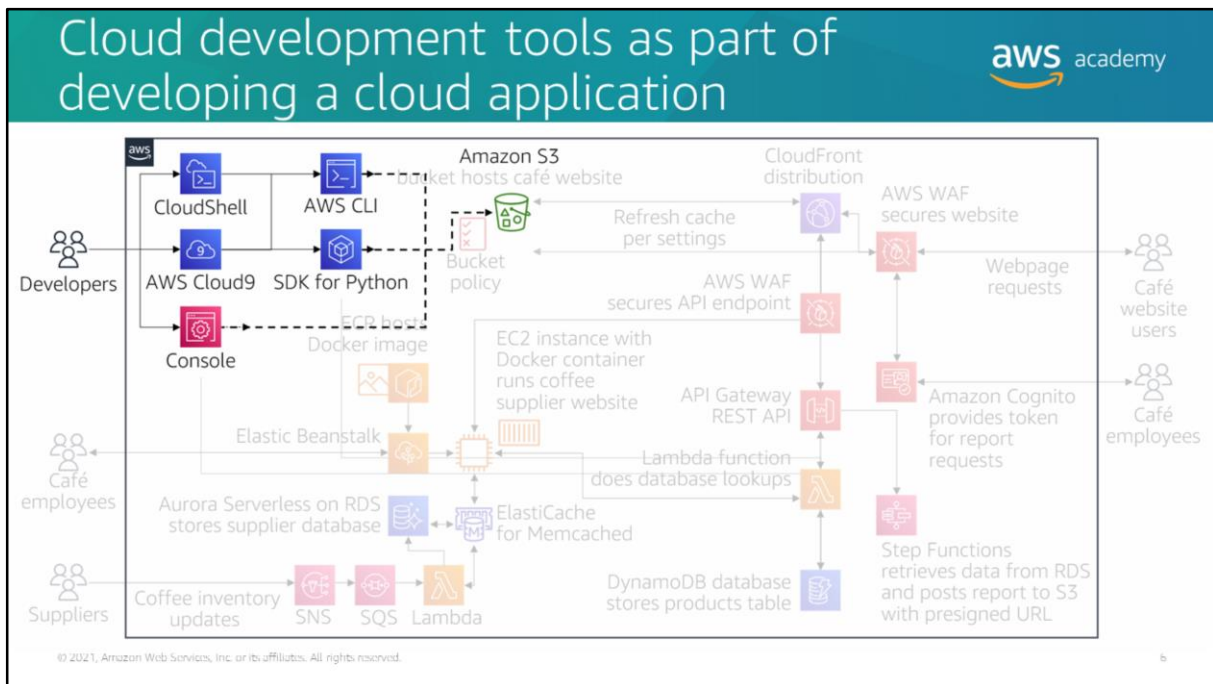
Sofia wants to start developing a web presence for the café. Before she starts coding, she wants to decide on a development environment to use to develop and run her code.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

5

Sofia wants to start developing a web presence for the café. Sofia has Python development skills, and she is learning more about how to develop solutions in the cloud. Before she starts coding, she wants to decide on a development environment to use to develop and run her code. She decides to explore at least two options that are available on AWS.



The diagram on this slide gives an overview of the application that you will build through the labs in this course. The highlighted portions are relevant to this module.

As highlighted in the diagram, you can use the AWS Management Console to launch AWS CloudShell or create an AWS Cloud9 development environment. With either CloudShell or AWS Cloud9, you can use the AWS Command Line Interface (AWS CLI) or AWS SDK for Python (Boto3) to interact with AWS resources (for example, an S3 bucket).

Module 2: Introduction to Developing on AWS

Section 2: Systems development lifecycle

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

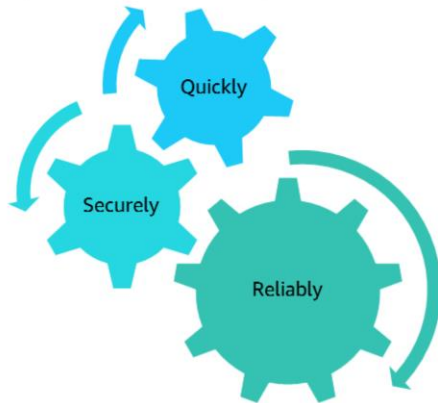


Section 2: Systems development lifecycle.

The challenge of software delivery



Enterprises must bridge the gap between the stability of their operations and rapid feature development.



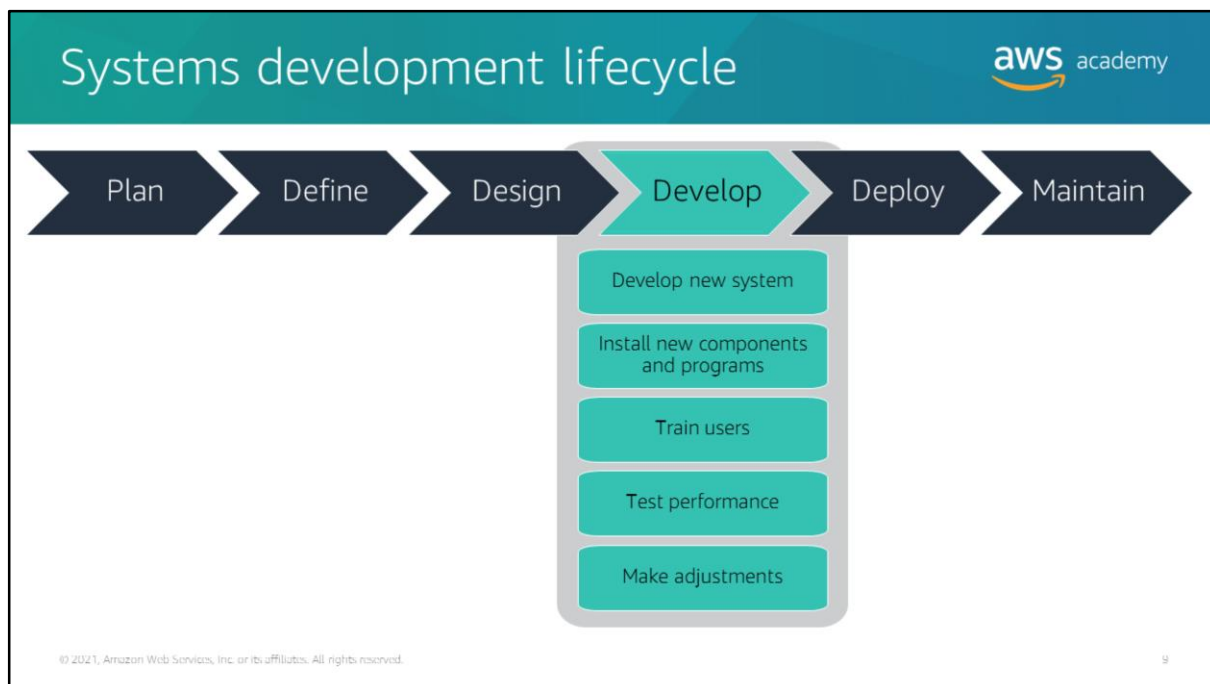
Deliver software with **zero tolerance** for outages:

- Quickly
- Securely
- Reliably

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

8

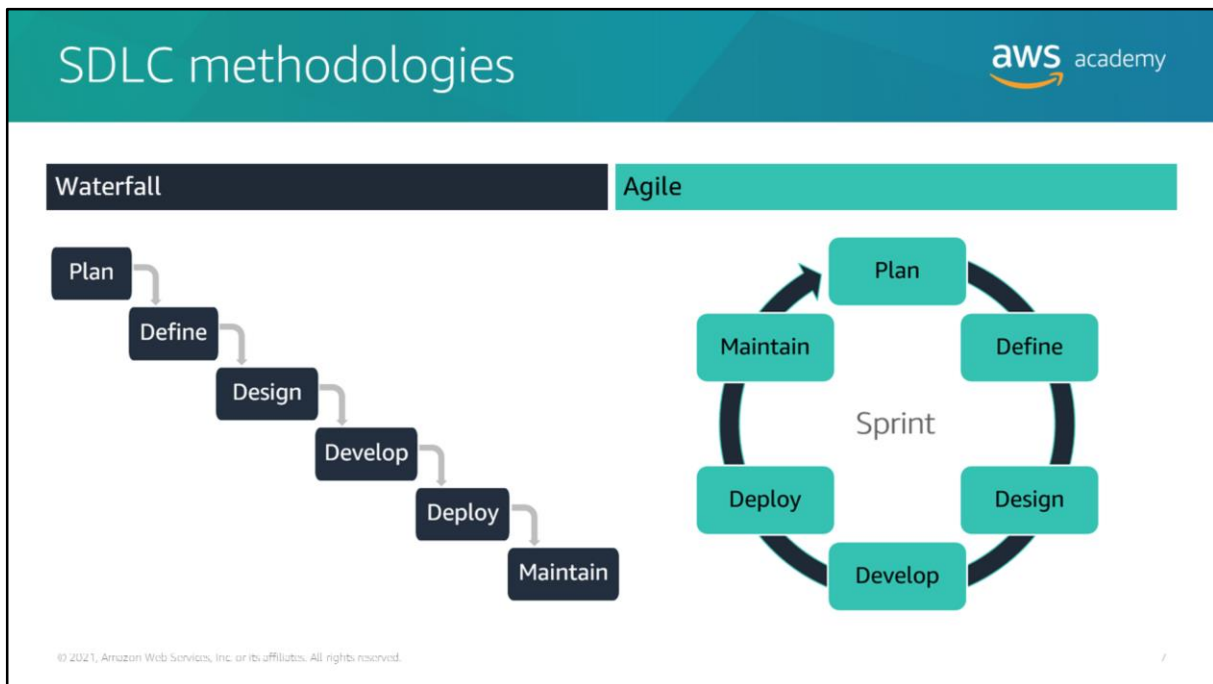
Software development and delivery is a challenge. Developers must quickly build solutions that are secure, reliable, and can scale to meet customer demands. All these goals must be accomplished while ensuring the security and integrity of the system. Methodologies are needed to bridge the gap between operational stability and rapid feature development.



Software development goes through a series of phases in the systems development lifecycle (SDLC). The SDLC is a conceptual model that is used in project management. It describes the stages in an information system development project, from an initial feasibility study to the maintenance of the completed application.

In general, an SDLC methodology follows these steps: plan, define, design, develop, deploy and maintain. This course will primarily focus on the Develop phase of SDLC. This course will briefly touch on the Deploy phase as well.

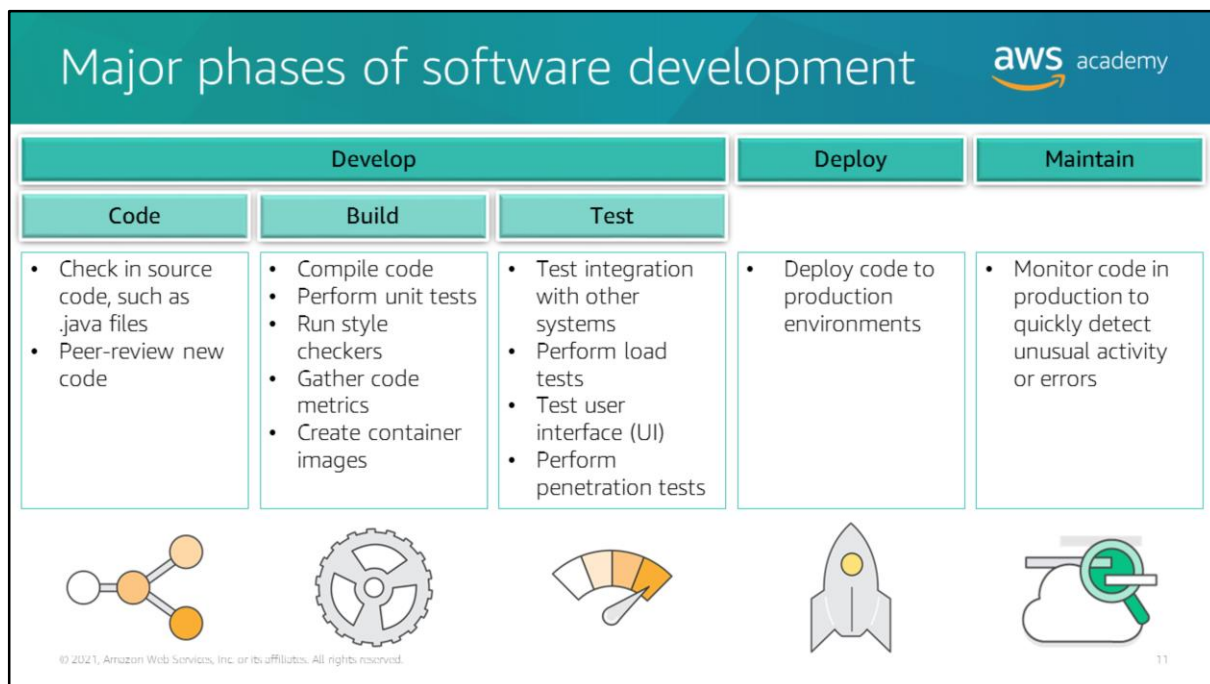
In the Develop phase, the new system is developed. New components and programs must be obtained and installed. System users must be trained, and all aspects of the system's performance must be tested. If necessary, bugs must be fixed and adjustments must be made to improve performance.



Various SDLC methodologies have been developed. Two of the most common methods are *waterfall* and *agile*.

- *Waterfall (or traditional) methodology* – This methodology is often considered to be the classic approach to the SDLC. The waterfall model describes a sequential development method. Each development phase has distinct goals and tasks that must be completed before the next phase can begin. Under this paradigm, product teams might not hear back from customers for months, and often not until the product is commercialized.
- *Agile software development methodology* – Agile is a newer conceptual framework that supports fast-paced, iterative software development. Under this newer paradigm, product teams push their work to customers as quickly as possible so that they can collect feedback and improve the previous iteration of their products. Concepts such as minimum viable product (MVP), release candidate, velocity, and others are all derived from the agile approach. There are various agile software development methodologies, such as Crystal methods, Dynamic Systems Development Method (DSDM), and Scrum. Under agile, software is developed iteratively in short time periods that are called *sprints*, which typically last 1–4 weeks.

When you develop applications in an agile way, your requirements might change frequently. Developing on AWS enables you to accommodate the changes and requirements more efficiently.



The five major phases of software development are: Code, Build, Test, Deploy, and Maintain. Note that Source, Build, and Test fall under the Develop stage of the SDLC. Each phase provides increased confidence that the code will work in the way it's intended when it's eventually released to customers. The following overview lists the activities that occur in each phase:

- During the *Code phase*, developers write application source code and check changes into a source code repository, such as a Git repository or an AWS CodeCommit repository. Many teams use code reviews to provide peer feedback of code quality before they ship code into production. Other teams use pair programming as a way to provide real-time peer feedback.
- During the *Build phase*, an application's source code is compiled and the quality of the code is tested on the build machine. The most common type of quality check is an automated test that doesn't require a server to run. These quality tests can be initiated from a test harness. Some teams extend their quality tests to include code metrics and style checks.

- During the *Test phase*, additional tests (that can't be done during the Build phase) are performed. These tests require the software to be deployed to a production-like environment. Often, these tests include integration testing with other live systems, load testing, user interface (UI) testing, and penetration testing. A common pattern is for engineers to deploy builds to a personal development stage, where they can check that their automated tests work correctly. They then deploy code to pre-production stages, where their application interacts with other systems to ensure that the software works in an integrated environment.
- Finally, code gets deployed to production. Though there are different deployment strategies, the common goals are to reduce risk when deploying new changes and to minimize the impact if a bad change is released to production.
- After code is deployed, it must be monitored in production to determine if everything works as expected.

Each of these steps can be automated without automating the entire release process.

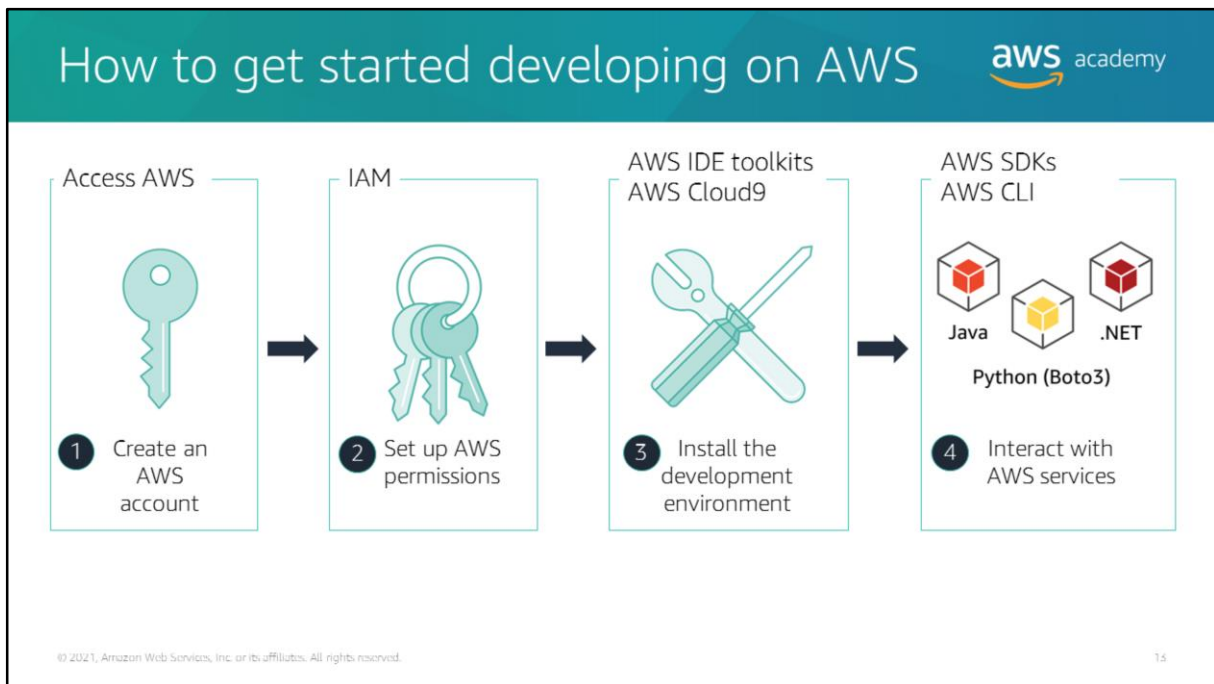
Module 2: Introduction to Developing on AWS

Section 3: Steps to get started developing on AWS

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.




Section 3: Steps to get started developing on AWS.



Before you start developing applications on AWS, you must do a few key things. First, you will need an *AWS account* and you must also create an *AWS Identity and Access Management (IAM) user*. This process includes setting up *AWS permissions* to access specific services and operations. Next, you must *install your development environment* for your language of choice. You must then install the AWS SDK for the language that you will use—or the AWS Command Line Interface (or AWS CLI)—so that you can *interact with AWS services*.

This section will discuss each AWS offering that's involved in setting up AWS permissions, installing the development environment, and interacting with AWS services. Permissions will be covered in another module in this course.

Getting started: AWS account



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

14

Create an AWS account

Go to aws.amazon.com to create an account

Choose [sign in to the console](#) if you already have an account

Set up AWS permissions

Install the development environment

Interact with AWS services

Create an AWS account

Email address

Password

Confirm password

AWS account name ID

Continue

Sign in to an existing AWS account

Sign in

☒ **Root user**
Account owner that performs tasks requiring unrestricted access. [Learn more](#)

☐ **IAM user**
User within an account that performs daily tasks. [Learn more](#)

Root user email address

username@example.com


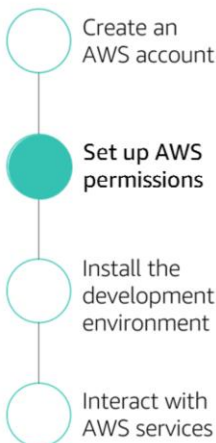

Next

The steps to create an AWS account are as follows:

1. Open the AWS homepage at <http://aws.amazon.com>
2. Choose **Create an AWS Account**.
 - **Note:** If you signed into AWS recently with the same browser, choose **Sign in to the Console**. If **Create a new AWS account** isn't available, then choose **Sign in to a different account** and then choose **Create a new AWS account**.
3. Enter the requested information and then choose **Continue**.
4. Select if the account is personal or professional. For learning purposes, select **personal**.
 - **Note:** Both types of accounts (personal or professional) have the same features.

5. Enter either your personal information or company information to continue.
 - If you are creating a professional account, enter the company's phone number and an organizational email address. Configuring a professional account with a personal phone number and email address can make an account less secure.
 - Ensure that you can access the email address that you used. You will need to access this email address to receive the confirmation message that your account was created.
6. Read and accept the AWS Customer Agreement (<https://aws.amazon.com/agreement/>).
7. Finally, choose **Create Account and Continue**.

Getting started: Permissions



- Identity (authentication):
Who can use your AWS resources
- Access management (authorization):
What resources can they use and in what ways

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.


15

When first you create an AWS account, you begin with a single sign-on identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user, and it's accessed by logging in with the email address and password that you used to create the account. A best practice is to not use the root account unless it's required. Instead, to increase the security of your AWS account, we recommend that you create an IAM user that will use access credentials instead of using your AWS account credentials.

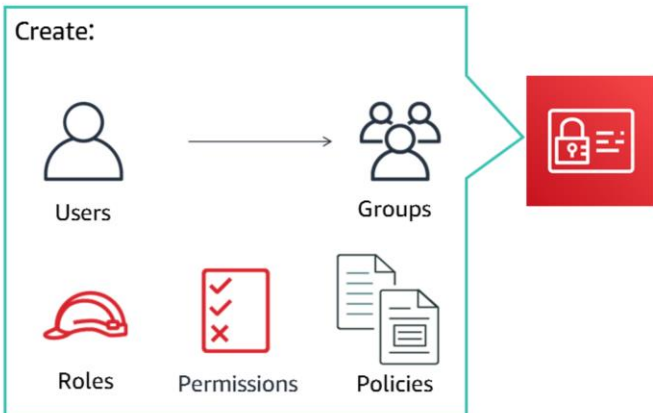
IAM is a service that helps you securely control access to AWS resources for your users. You use IAM to control who can use your AWS resources (authentication) and what resources they can use and in what ways (authorization).

IAM enables you to create and manage users, groups, roles, and policies to control access to AWS services. IAM also enables identity federation between a corporate directory and AWS services.

IAM example



Create:



Example:

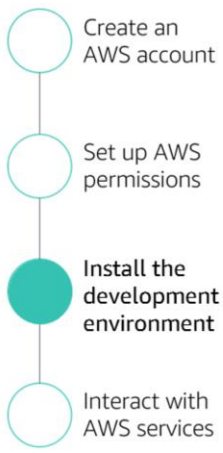
- **User** – John Doe (has AWS credentials).
- **Group** – John's IAM user belongs to the Developer group.
- **Role** – Trusted AWS entity with the Developer permissions policies.
- **Policy** – John's IAM role is associated with the Developer policy and has access to specific services and operations.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. 16

For example, consider a situation where you are responsible for an organization's users. You could use IAM to set up a user account for each developer in your organization. Each user has an IAM user account that they must use to access AWS services. You could then create an IAM role for developers and associate the developer role with each developer's user account. The developer role can be configured with policies that control which services and operations the role can access. Developers then assume the IAM role to have the permissions they need to work in the environment. When updates are needed to the permissions developers require, the IAM Policy associated with the IAM role is updated and any developer assuming the role gets the required permissions.

You will learn more about IAM in the next module.

Getting started: Development environment



- Create an AWS account
- Set up AWS permissions
- Install the development environment**
- Interact with AWS services

IDEs and AWS IDE Toolkits:

- AWS Cloud9
- Eclipse IDE (Eclipse Foundation)
- IntelliJ IDEA (JetBrains)
- PyCharm (JetBrains)
- Microsoft Visual Studio
- Microsoft Visual Studio Code
- Microsoft Azure DevOps
- Rider (JetBrains)

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. 1/



When you set up your development environment, you need to install your programming language of choice and an integrated development environment (IDE). AWS offers several language-specific IDEs for you to write, run, debug, and deploy applications.

For more details about available AWS SDKs, IDEs, and toolkits, refer to Tools to Build on AWS (<https://aws.amazon.com/tools/>).

For information about how to install your development environment, follow the instructions from the technology vendors of the platform and the IDE.

Next, you will learn more about the AWS Cloud9 IDE.

AWS Cloud9



Start projects quickly and code with only a browser.

Code together in real time.

Build serverless applications.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

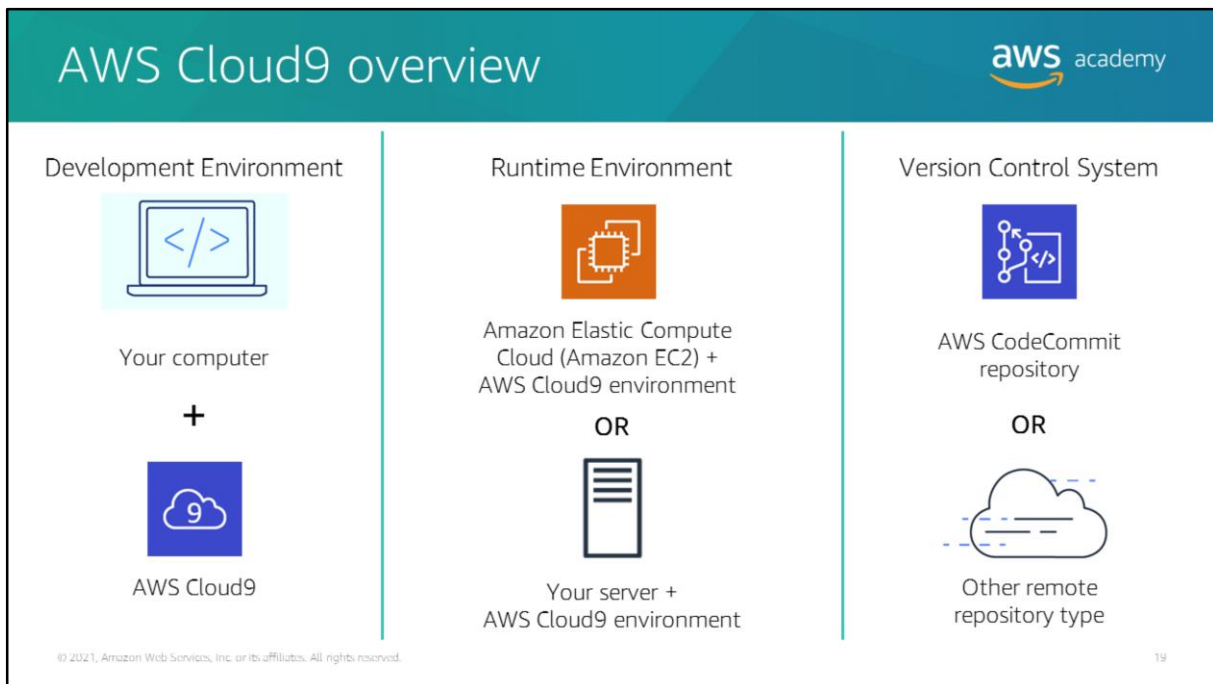
18

AWS Cloud9 is an integrated development environment (IDE) that can be run in a web browser. Since the environment is hosted in AWS, developers can switch between desktops, laptops, and even tablets to access and work with their code. AWS Cloud9 has all the expected features of an IDE (code completion, step-through, terminal interface, debugging, and so on) but also includes capabilities like environment sharing which enables real-time collaborative coding with other developers. Cloud9 supports many programming languages such as Python, PHP, JavaScript, C++ and more. When working with serverless such as AWS Lambda, Cloud9 allows developers to quickly pull code, make changes, and push the updates directly to Lambda if required.

AWS Cloud9 provides a few key benefits:

- *Start projects quickly and code with only a web browser* – To use AWS Cloud9, you need only an AWS account. You can log in to the AWS Management Console, create an AWS Cloud9 environment, and start using it in your browser.
- *Code together in real time* – Instead of committing your changes to Git and asking a peer to review the changes on their machine, both you and your peer can use AWS Cloud9 to code together in real time. AWS Cloud9 provides real-time feedback. This feature enables you to access to the code that your peer is entering into the IDE while they enter it so that you can respond in real time.
- *Build serverless applications* – You can edit and debug AWS Lambda functions locally, which reduces the need to upload your code to the Lambda console for testing and debugging.

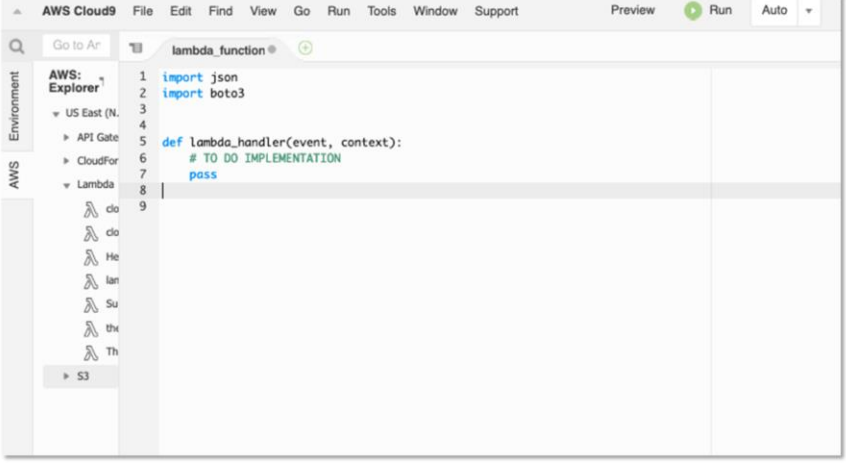
For more information about AWS Cloud9, refer to the AWS Cloud9 product page (<https://aws.amazon.com/cloud9/>).



You can use the AWS Cloud9 console to create an AWS Cloud9 development environment. In AWS Cloud9, an *environment* is where you store your development project files and run the tools to develop your applications. When you create an AWS Cloud9 environment, you can connect it to an Amazon Elastic Compute Cloud (Amazon EC2) instance or to your own server (called a Secure Shell, or SSH, environment).

AWS Cloud9 integrates with AWS CodeCommit and other remote repositories so that you can work with files in your environment. AWS CodeCommit is a fully managed source control service that hosts secure Git-based repositories. For more information about AWS CodeCommit, refer to the AWS CodeCommit product page (<https://aws.amazon.com/codecommit/>).

AWS Cloud9 editor



The screenshot displays the AWS Cloud9 IDE interface. At the top, there's a teal header with the 'AWS Cloud9 editor' title and the 'aws academy' logo. Below the header is a menu bar with options: File, Edit, Find, View, Go, Run, Tools, Window, Support. On the right of the menu bar are buttons for 'Preview', 'Run' (with a green play icon), and 'Auto' (with a dropdown arrow). The main workspace is divided into three panes. The left pane is the 'AWS Explorer' sidebar, showing a tree view of the AWS environment with folders for 'US East (N.)', 'API Gate', 'CloudFor', and 'Lambda'. The middle pane shows a code editor with a file named 'lambda_function.py'. The code contains the following Python code:

```
1 import json
2 import boto3
3
4
5 def lambda_handler(event, context):
6     # TO DO IMPLEMENTATION
7     pass
8
9
```

The right pane is currently empty. At the bottom of the interface, there is a small copyright notice: '© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.' and a page number '20'.

- Fully featured editor
- Broad selection of run configurations
- Integrated debugger
- Integrated tools for serverless development
- Connectivity to any Linux server platform
- Built-in terminal
- Collaborative editing and chat
- Continuous delivery toolchain
- File-revision history

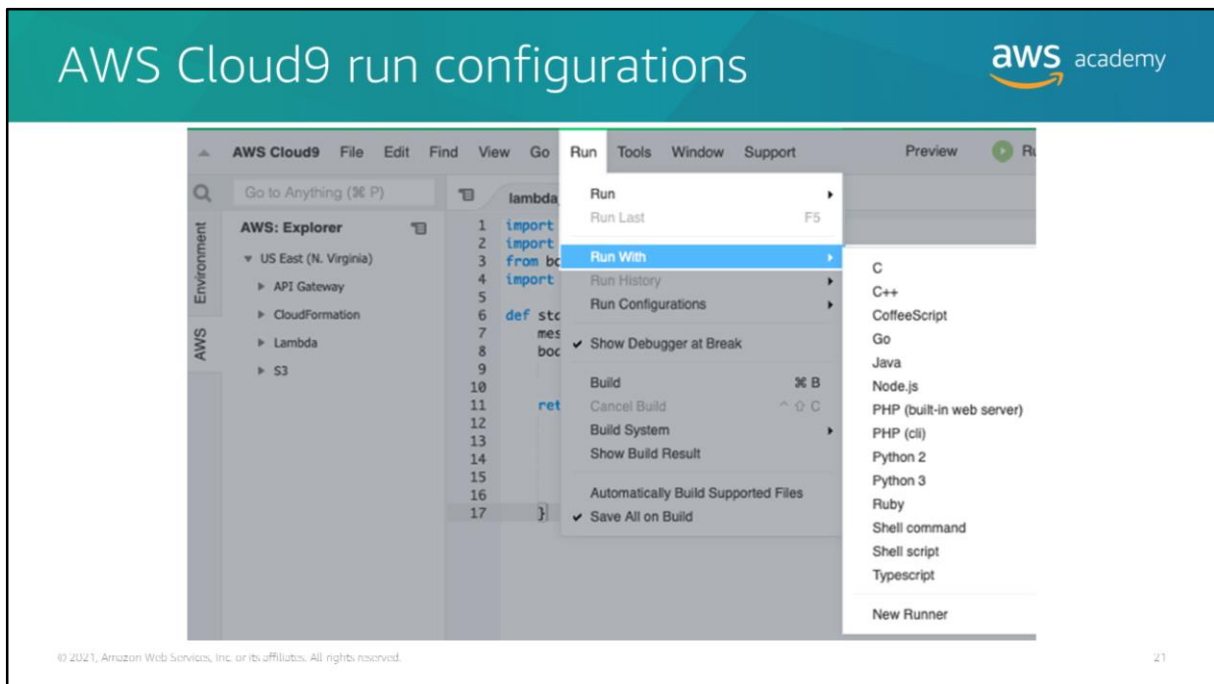
The AWS Cloud9 editor is the tool that you use to write code.

It has standard IDE features, such as:

- Fully featured editor
- Broad selection of run configurations
- Integrated debugger
- Integrated tools for serverless development
- Connectivity to any Linux server platform
- Built-in terminal
- Collaborative editing and chat
- Continuous delivery toolchain
- File-revision history

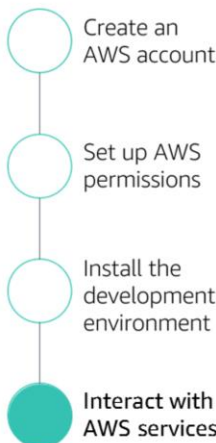

AWS Cloud9 offers additional features. For example, the AWS Cloud9 editor can handle large files, such as files with 100,000 or more lines. It offers multiple themes, built-in support for more than 40 language modes, and customizable run configurations for your projects. In addition, the editor has a Vim mode and an Emacs mode, and it has a keybinding editor for customizing how you use commands.

The editor supports keyboard navigation and commands (similar to other editors such as SublimeText, or Vim plugins like ctrlp). Shortcuts are available to open files, open the command pane to run commands, and more. The editor also ships with a preconfigured version of the AWS CLI and other preinstalled tools so that you can access your resources.



When developers configure a project in AWS Cloud9 they can choose from several runtime environments, which are also known as *run configurations*. Developers can choose the appropriate run configuration from the menu. With Python, developers can also select the version that they want to use. In addition, developers can create a new runner by using the **New Runner** option. A runner is a configuration file, written in JSON, that defines what program (Python, PHP, gcc, and so on) should be used to run the code and any parameters that may be needed. For more information about creating custom run configurations, refer to Create a Run Configuration (<http://docs.aws.amazon.com/console/cloud9/create-run-config>) in the AWS Cloud9 documentation.

Getting started: AWS services

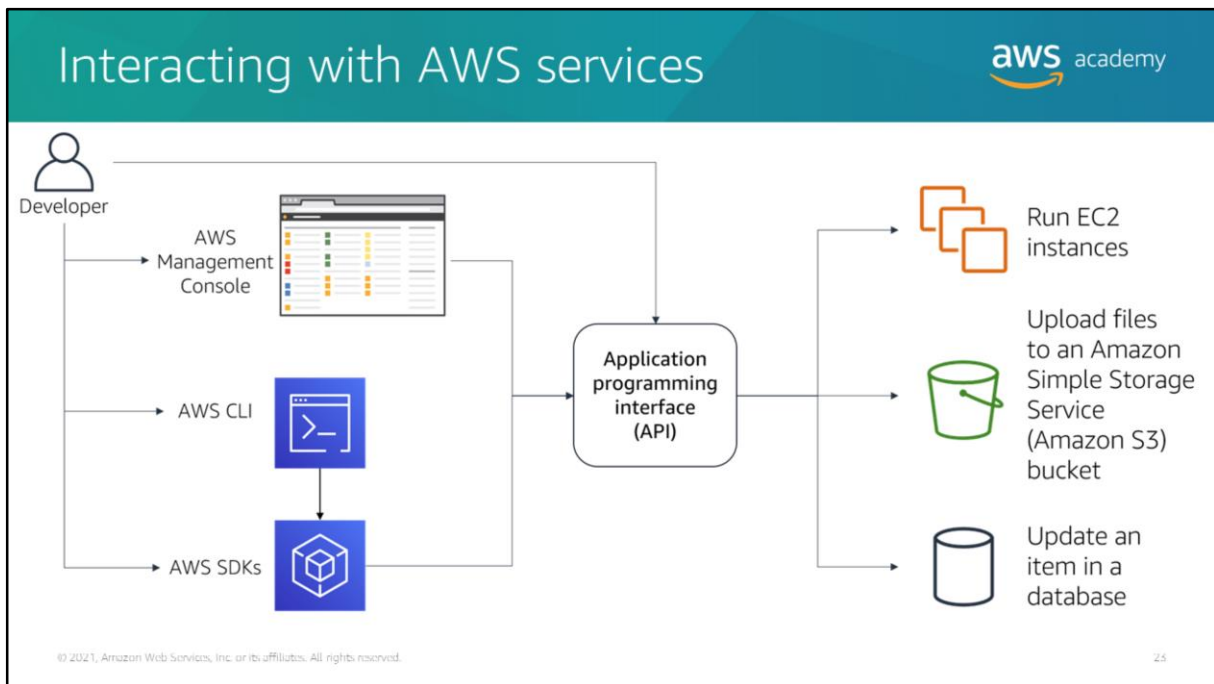


- AWS Management Console
- Programmatically
 - AWS Command Line Interface (AWS CLI)
 - AWS software development kits (SDKs)
 - Service application programming interfaces (APIs)

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

22

You can interact with AWS through the AWS Management Console, or you can interact with AWS programmatically. With programmatic access, services can be called through the AWS Command Line Interface (AWS CLI), the AWS software development kit (SDK) for the supported language, or the service application programming interface (API).



All AWS services are managed through a common, REST-like API. AWS provides an API for each of its services.

You can interact with the APIs to access your AWS resources in four ways:


- *Directly* – You can call the APIs directly.
- *AWS Management Console* – An implementation of the API calls in a web console. It provides a rich graphical interface to a majority of the features that are offered by AWS.
 - **Note:** Occasionally, new features might not be available in the console when the feature initially launches.
- *AWS Command Line Interface (AWS CLI)* – An open source tool that enables you to interact with AWS services by using commands in your command line interface.
- *AWS software development kits (AWS SDKs)* – Packages that enable you to access AWS services and resources in a variety of popular programming languages.


The AWS CLI and the SDKs provide flexibility. They enable you to create your own tools and customize existing AWS features. For example, you can create your own scripts or applications for launching EC2 instances that enforce using a specific set of Amazon Machine Images (AMIs), add a standard set of tags, and so on.

You can use these access modes interchangeably. For example, you can create server EC2 instance through an SDK call. This instance can be queried for its configuration information by using an AWS CLI command (`aws ec2 describe-instances`). Finally, you can terminate the server instance by using the console. (It might sometimes take a few seconds or a few minutes for changes made through the AWS CLI or the API to display in the console.)

For information about the AWS SDKs and AWS CLI, refer to Tools to Build on AWS (<https://aws.amazon.com/tools/>).


Command line tools






AWS CLI

Windows, macOS,
Linux or Unix




AWS Tools for PowerShell

Windows, macOS,
Linux



AWS Serverless Application Model (AWS SAM) Local



AWS Amplify

[AWS Command Line Interface user guide, Installing the AWS CLI](#)

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

24

When you use the command line, you can use various tools to call AWS services:

- *AWS Command Line Interface (AWS CLI)* is an open source tool that enables you to interact with AWS services by using commands in your command line interface. With minimal configuration, you can start using the command prompt in your favorite terminal program to access functionality that's equivalent to the AWS Management Console. For information, refer to the AWS Command Line Interface (<https://aws.amazon.com/cli/>)
- *AWS Tools for PowerShell* offers the AWS Tools for Windows PowerShell and AWS Tools for PowerShell Core. They are PowerShell modules that are built on the functionality that's exposed by the SDK for .NET. The AWS PowerShell Tools enable you to script operations on your AWS resources from the PowerShell command line. For information, refer to the AWS Tools for PowerShell User Guide (<https://docs.aws.amazon.com/powershell/latest/userguide/pstools-welcome.html>).

- *AWS Serverless Application Model (AWS SAM) Local* is an AWS CLI tool for the local development and testing of serverless applications. For information about AWS SAM Local, refer to the aws-sam-cli GitHub repository (<https://github.com/aws-labs/aws-sam-cli>).
- *AWS Amplify* enables you to create, configure, and implement scalable mobile applications that are powered by AWS. AWS Amplify provisions and manages your mobile backend. It also provides a simple framework so that you can integrate your backend with your frontends, including iOS, Android, web, and React Native. AWS Amplify also automates the application release process of both your frontend and your backend, enabling you to deliver features faster. For more information, refer to the AWS Amplify product page (<https://aws.amazon.com/amplify/>).

For more information, refer to:

- AWS command line tools – <https://aws.amazon.com/tools/>
- How to get started with the AWS CLI – <https://aws.amazon.com/cli/>
- How to install the AWS CLI – <https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html>

Demonstration: Installing the AWS CLI







© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Now, the educator might choose to demonstrate installing the AWS CLI.

AWS CLI: Command line format



AWS CLI

Service (command) Operation (sub-command) Parameters

```
$ aws ec2 stop-instances --instance-id i-1234567890abcdef0
```

```
$ aws ec2 run-instances --cli-input-json file:///./lcf/webserver.json
```


```
$ aws help
$ aws ec2 help
$ aws ec2 describe-instances help
```


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. 25

Calls made from the AWS CLI have a certain format:

- The command line always begins with *aws*. This string is how you invoke the AWS CLI.
- The next component is a *service command*. This string is the top-level AWS service that you are calling (for example, Amazon EC2). For a list of currently supported services, refer to <https://aws.amazon.com/cli/>.
- The *operation subcommand* is next. This string is the operation that you want to perform on that service. This example uses the run-instances subcommand to request the creation of a new EC2 instance.
- Then, you specify any *parameters*. The parameters are any arguments that are needed to perform the operation. Argument names are preceded by two dashes (--). For example, to create an EC2 instance, you must run the Amazon EC2 run-instances operation and pass in several parameters, such as the Amazon Machine Image (AMI) ID.
- You can also specify *options*. The options are choices that you can specify when you run an operation. For example, you can use the --query option to limit the response text so that it returns only the instance ID of your new instance.

AWS CloudShell





AWS CloudShell

AWS CloudShell provides CLI access in a web browser

- Available in the AWS Management Console
- Pre-authenticated session
- Pre-installed with the AWS CLI, SDKs, and other tools
- No cost
- 1 GB of storage per Region

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.2 /

Some of the features of AWS CloudShell include:

- CloudShell inherits your credentials from the AWS Management Console. This means there are not key files to manage when using the AWS CLI.
- It runs on an Amazon Linux 2 instance that's fully managed. All the latest versions of the tools are installed without needing to update or patch the environment
- It incurs no cost. CloudShell includes 1 GB of persistent storage per Region. You only pay for the resources that you create and run on AWS.
- It's customizable (per Region) for storing scripts, files, configuration preferences, and tools.

Module 2: Introduction to Developing on AWS


Section 4: Fundamentals of working with the AWS SDKs

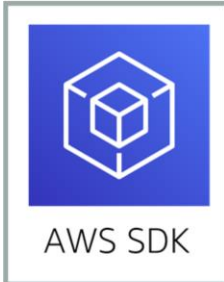
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Section 4: Fundamentals of working with AWS SDKs.

AWS SDKs





AWS SDK for:

- C++
- Go
- Java
- JavaScript in the browser
- JavaScript in Node.js
- .NET and Xamarin
- PHP
- Python (Boto3)
- Ruby

AWS Mobile SDK for:

- Android
- iOS
- .NET and Xamarin
- Unity

AWS IoT Device SDK for:

- Arduino Yún
- C++
- Embedded C
- Java
- JavaScript
- Python

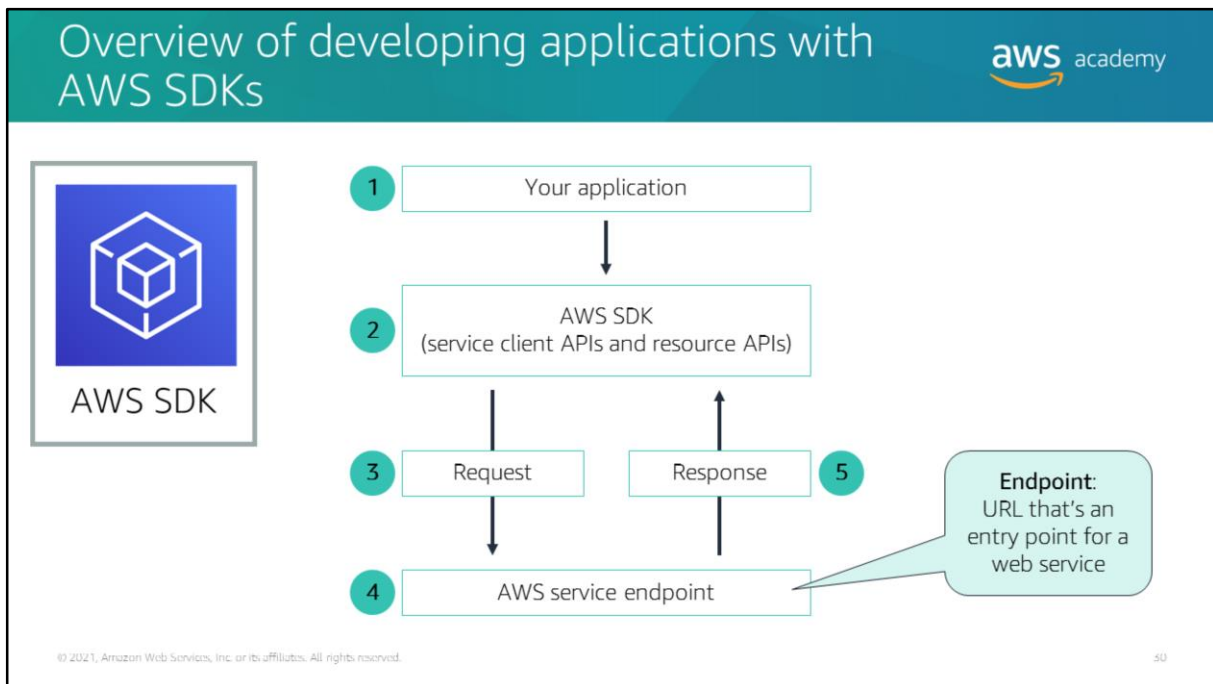
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.29

The AWS SDKs are available in various programming languages and technology platforms. You can use the AWS SDKs to call AWS services from your application code.

Say, for example, that you are developing a music-sharing application. Your application needs to upload a user's music files to an S3 bucket. In this case, your application could use the AWS SDK to programmatically upload files to the S3 bucket.

To learn more about the available AWS SDKs (including installation instructions), refer to the following resources:

- **SDKs** section of <https://aws.amazon.com/tools/>
- <https://docs.aws.amazon.com/#sdks> in the AWS Documentation



When you develop applications on AWS, you can use the AWS SDKs to make API calls to AWS services and get responses back from those services. This diagram illustrates application development with the AWS SDKs.

1. You write an application using an AWS SDK in your language of choice.
2. Each AWS SDK provides one or more APIs for working with AWS services.
3. The AWS SDK constructs an HTTP(S) request for use with the service API, and sends the request to the AWS service endpoint.
4. The service performs the request and sends a response.
5. The AWS SDK processes the response and propagates it back to your application.

Service client API



- Has one method per service operation
- Has objects for request and result data
- Example (in Python):

```
# List objects in bucket using the client API
# Return type: dict / additional API calls needed to get objects
def listclient():
    s3client = boto3.client('s3')
    response = s3client.list_objects_v2(Bucket='DOC-EXAMPLE-BUCKET')
    for content in response['Contents']:
        print(content['Key'], content['LastModified'])
    return
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

51

AWS SDKs offer two levels of APIs that you can use to call AWS services: *service client APIs* and *resource APIs*.

Service client API

AWS SDKs provide a set of *client classes*. Each client class exposes a direct mapping of the API for the AWS service. These client objects have a method for each operation that the service supports, with corresponding objects that represent the request parameters and the response data. Using this *low-level* client API gives you full control over the requests that you make to the service, which enables you to tightly control the behavior and performance of your calls to AWS services.

For example, both the SDK for Java and the SDK for .NET and Xamarin have *Amazon<Service>Client* classes that correspond to each service, such as *AmazonS3Client* and *AmazonDynamoDBClient*.

This example compares how to list the objects in the *DOC-EXAMPLE-BUCKET* S3 bucket by using the SDK for Python service client API.

For more information, refer to the Amazon S3 service client section

(<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html#client>) of the SDK for Python documentation.

Resource API



- Has one class per conceptual resource
- Defines service resources and individual resources
- Example (in Python):

```
# List Objects in Bucket using the Resource API  
# Resources represent an object-oriented interface to Amazon Web Services (AWS).  
# They provide a higher-level abstraction than the raw, low-level calls made by  
# service clients  
def listResource():  
    s3resource = boto3.resource('s3')  
    bucket = s3resource.Bucket('DOC-EXAMPLE-BUCKET')  
    for object in bucket.objects.all():  
        print(object.key, object.last_modified)  
    return
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

52

Some AWS SDKs, such as the AWS SDK for Python (Boto3), provide higher-level APIs that are called *resource APIs*. Resource APIs provide a higher-level abstraction than the low-level calls that are made by clients. Instead of a single client object exposing the entire API for a service, the resource APIs consist of classes that represent each of the conceptual resources that you interact with when you use a service. These classes expose methods to retrieve data about the resource, invoke actions that can be taken on the resource, and retrieve links to other related resources. Access to the resources are then provided through objects and collections. Resources include service such as Amazon S3 or Amazon Simple Queue Service.

For example, the Amazon S3 service resource exposes methods to perform actions on the Amazon S3 service. An individual Amazon S3 resource (such as an object) has an identifier or object key, and attributes (such as `last_modified`).

This example compares how to list the objects in the *DOC-EXAMPLE-BUCKET* S3 bucket by using the resource API for the SDK for Python.

For more information about resource APIs, refer to the following pages in the SDK for Python for documentation:

- <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/resources.html>
- <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html#service-resource>

Specifying AWS Regions

Specify Region when you instantiate the service client.

```
dynamodb_client =  
boto3.client('dynamodb',  
region_name='us-east-1')
```

Default Region is specified in AWS config.

```
~/.aws/config  
[default]  
region=us-west-2  
output=json
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. 55


The AWS Cloud infrastructure is built around Regions and Availability Zones. A Region is a physical location in the world that has multiple Availability Zones. When you write your applications, you can access AWS services that physically reside in a specific geographic region. How you specify the AWS Region depends on the AWS SDK that you use.

With some AWS SDKs, such as the AWS SDKs for Java and .NET, you can specify the Region when you instantiate the service client. You must create a separate instance of the service client for each Region that you want to work with. For other SDKs, such as the AWS SDK for Python (Boto3), you can use the default Region that is set in the `~/.aws/config` file.

For more information about specifying AWS Regions with the AWS SDKs, refer to the following resources in the AWS Documentation:

- SDK for Java – <https://docs.aws.amazon.com/AWSSdkDocsJava/latest/DeveloperGuide/java-dg-region-selection.html>
- SDK for .NET and Xamarin – <https://docs.aws.amazon.com/sdk-for-net/v3/developer-guide/net-dg-region-selection.html>
- SDK for Python – <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html#configuration>
- SDK for JavaScript – <https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/setting-region.html>
- SDK for Ruby – <https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/setup-config.html>

Handling exceptions



Error response example

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/DOC-EXAMPLE-BUCKET/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

What to do?

- **HTTP 400 series** (client error): handle error in application
- **HTTP 500 series** (server error): retry operation

Error response element descriptions

Error: Container for all error elements	Code: String that uniquely identifies an error condition	Message: Generic description of the error code in English	Resource: The resource that's involved in the error	RequestId: ID of the request that's associated with the error
--	---	--	--	--

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

54

When an API call is made to an AWS service, the service performs the request and returns a response that includes an HTTP status code. If the request is successful, the AWS service returns an *HTTP 200* response code. If the request is unsuccessful, the AWS service returns an *error response*, which consists of the following elements:

- *Error* – Container for all error elements.
- *Code* – String that uniquely identifies an error condition. It's meant to be read and understood by programs that detect and handle errors by type.
- *Message* – Generic description of the error condition in English. It's intended for a human audience.
- *Resource* – Resource that's involved in the error.
- *RequestId* – ID of the request that's associated with the error.



You can use the error code to determine how to handle the error.

- A *400 series error* means that you must handle the error in your application. For example, Amazon S3 will return a 404 error code if the bucket you are trying to access doesn't exist. Your application can handle this error by first creating the bucket, and then performing operations on it.
- A *500 series error* indicates an internal server error. In this case, you could retry the operation. Each AWS SDK implements automatic retry logic. You can configure the maximum number of retry attempts. In addition to simple retries, each AWS SDK implements an exponential backoff algorithm to retry after failed connection attempts. With an exponential backoff algorithm, you specify progressively longer waits after each failed attempt before retrying your request.

Refer to the following resources for more information:


- Error Handling section of the Developer Guide for specific AWS services
- Error retries and exponential backoff in AWS
(<https://docs.aws.amazon.com/general/latest/gr/api-retries.html>)

Exceptions




SDK for Java
SDK for .NET and Xamarin

- `AmazonServiceException` (or subclass)
- `AmazonClientException`



SDK for Python

- `Botocore.exceptions.ClientError`



SDK for JavaScript

Asynchronous callback

- `function(error, data) { ... }`

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

55

AWS SDKs throw exceptions to communicate errors in different ways.

SDK for Java

The AWS SDK for Java throws the following unchecked (runtime) exceptions when errors occur:

- `AmazonServiceException` (and subclasses) – Indicates that the request was correctly transmitted to the service. However, the service couldn't process it, and it returned an error response instead. The exception has the following information that you can use to determine how your application should handle the error:
 - HTTP status code returned by the service, such as *404 Not Found*
 - AWS error code returned by the service, such as *NoSuchBucket*
 - Detailed error message from the service, such as *The specified bucket does not exist*
 - AWS request ID for the failed request
- `AmazonClientException` – Indicates that a problem occurred inside the Java client code. The problem might have occurred either while trying to send a request to AWS or while trying to parse a response from AWS. For example, the SDK for Java will throw an `AmazonClientException` if no network connection is available when you try to call an operation on one of the clients.
- `java.lang.IllegalArgumentException` – This exception is thrown if you pass an illegal argument when you perform an operation on a service.

For more information, refer to

<https://docs.aws.amazon.com/AWSSdkDocsJava/latest/DeveloperGuide/java-dg-region-selection.html>.

SDK for .NET and Xamarin

The SDK for .NET and Xamarin throws `Amazon.Runtime.AmazonServiceException` and `Amazon.Runtime.AmazonClientException`, which are similar to the errors from the SDK for Java.

SDK for Python

The SDK for Python throws `botocore.exceptions.ClientError`.

SDK for JavaScript

Each service method that sends a request can accept a callback as the last parameter with the signature `function(error, data) { ... }`. This callback is called when the response or error data is available.

For more information, refer to Calling Services Asynchronously

(<https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/calling-services-asynchronously.html>) [1](#) in the SDK for JavaScript Developer Guide.

Lab 2.1: Exploring AWS CloudShell and the AWS Cloud9 IDE





© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

You will now complete Lab 2.1: Exploring AWS CloudShell and the AWS Cloud9 IDE

Lab: Tasks



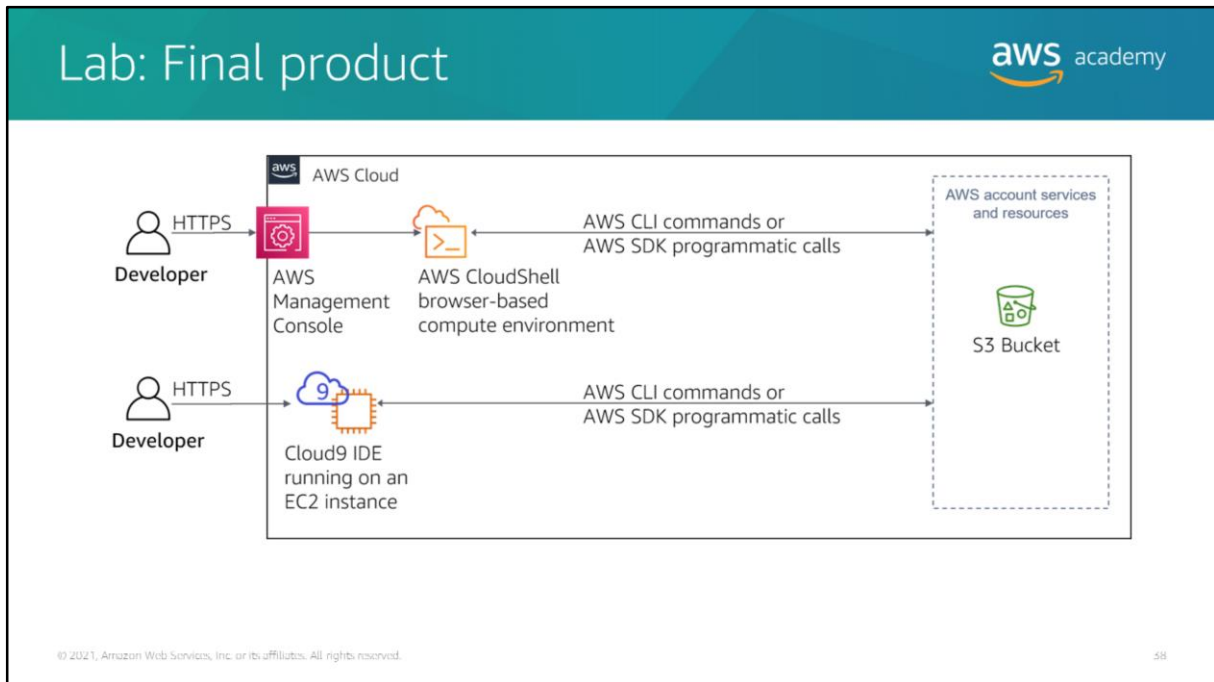
1. Exploring AWS CloudShell
2. Creating an AWS Cloud9 instance
3. Exploring the AWS Cloud9 IDE

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

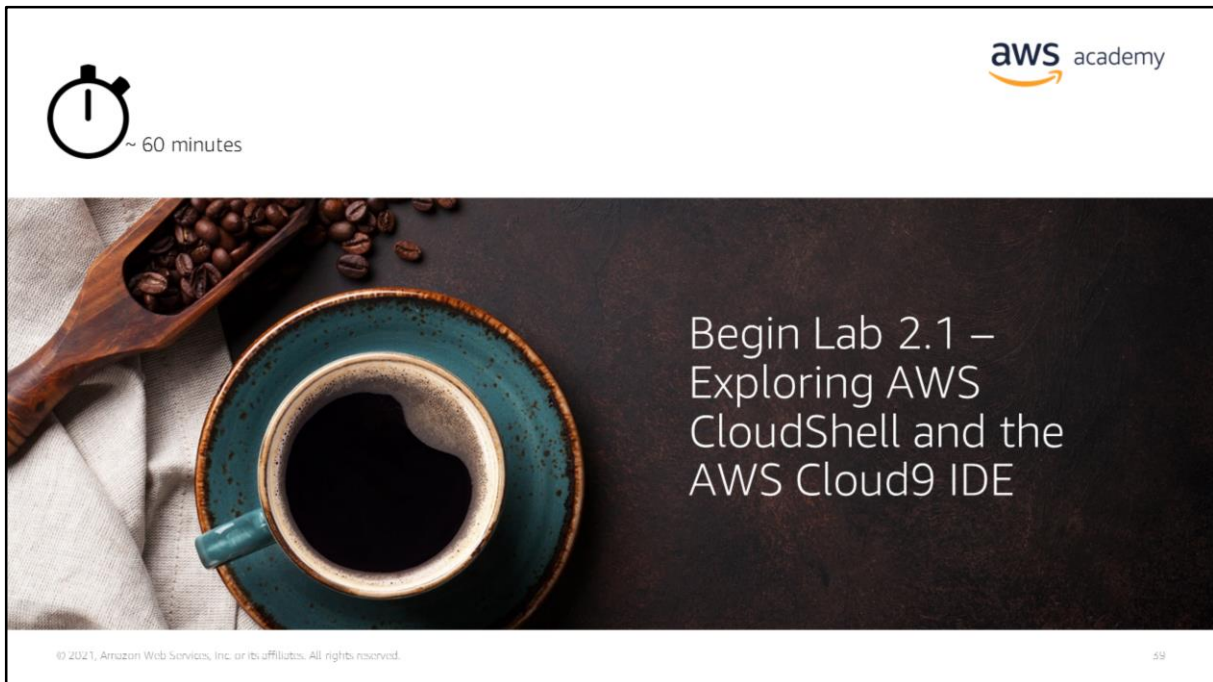
5 /

In this lab, you will complete the following tasks:


1. Exploring AWS CloudShell
2. Creating an AWS Cloud9 instance
3. Exploring the AWS Cloud9 IDE




The diagram summarizes what you will have done after you complete the lab. You will have used the AWS Management Console, AWS CloudShell, and AWS Cloud9 to interact with an Amazon S3 bucket.

A banner for AWS Academy Lab 2.1. The top left features a stopwatch icon and the text '~ 60 minutes'. The top right shows the 'aws academy' logo. The main image is a top-down view of a teal ceramic coffee cup filled with dark coffee, sitting on a matching saucer. A wooden scoop filled with coffee beans is positioned to the left of the cup, with some beans scattered on the dark surface. The text 'Begin Lab 2.1 – Exploring AWS CloudShell and the AWS Cloud9 IDE' is written in white on the right side of the image. At the bottom left, small text reads '© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.' and at the bottom right is the number '59'.

It is now time to start the lab.



Lab debrief:
Key takeaways



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

40

Your educator might choose to lead a conversation about the key takeaways from this lab after you have completed it.

Module 2: Introduction to Developing on AWS

Module wrap-up

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



It's now time to review the module and wrap up with a knowledge check.

Module summary



In summary, in this module, you learned how to:

- Recognize the systems development lifecycle
- Describe how to get started developing on AWS
- Indicate how to work with AWS SDKs
- Identify the benefits of using the AWS Cloud9 IDE
- Develop and run a simple program in AWS Cloud9

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

42

In summary, in this module, you learned how to:

- Recognize the systems development lifecycle
- Describe how to get started developing on Amazon Web Services (AWS)
- Indicate how to work with AWS software development kits (SDKs)
- Identify the benefits of using AWS Cloud9 integrated development environment (IDE)
- Develop and run a simple program in AWS Cloud9

You also gained experience with developing and running a simple program in AWS Cloud9.

To finish this module, complete the corresponding knowledge check.

Complete the knowledge check



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

45

It is now time to complete the knowledge check for this module.

Sample exam question



A new developer needs an integrated development environment (IDE) to write and update code for their application. Their technical lead moves between locations and must be able to review their code and leave comments in real time from any computer that they have access to.

Which method is the simplest way to meet this use case?

- A. Connect to the AWS Management Console through a web browser, and use the AWS CLI to access and update the code from both locations.
- B. Use an AWS CodeCommit repository to store the code, and keep local versions in sync on each computer.
- C. Configure an AWS Cloud9 environment running on an Amazon EC2 instance and connect to it through a web browser.
- D. Configure an AWS Serverless Application Model (AWS SAM) environment, and use the AWS SDK to work with the application code.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

44

Look at the answer choices, and rule them out based on the keywords that were previously highlighted.

The correct answer is C: *Configure an AWS Cloud9 environment running on an Amazon EC2 instance, and connect to it through a web browser.*

AWS Cloud9 provides an IDE that team members can access via a web browser without downloading the code. Teams can also use AWS Cloud9 to collaborate in real time.

The developer might also use CodeCommit to manage their code repository, but this option on its own does not provide an IDE that the developer can use for the kind of collaboration that this scenario describes.

The other elements noted (AWS CLI, AWS SDK, and AWS SAM) are all tools that a developer might use when developing and testing their application code.

Additional resources



To learn more about the AWS SDK for Python used in this course:

- Developer guide – [Boto3 documentation](#)
- API documentation – [Core references](#)

To learn more about cloud development best practices:

- Well-Architected Framework, [Appendix: Questions and Best Practices](#) section

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

45

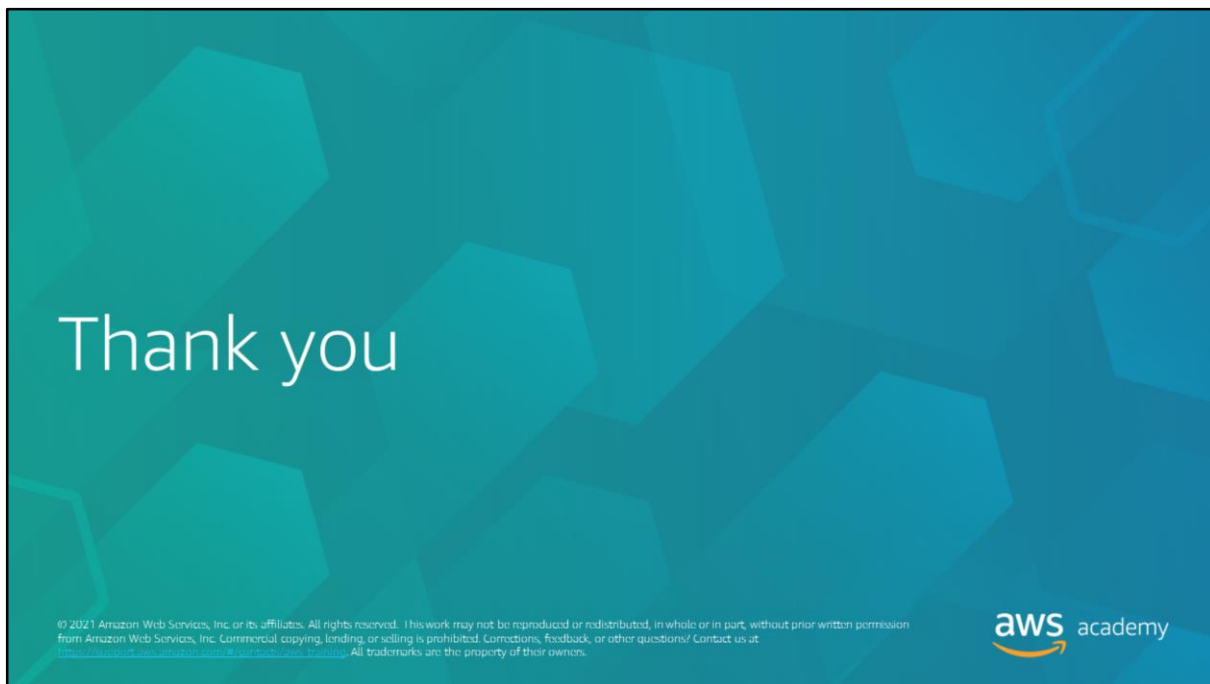
For all developer-related information, refer to the **AWS Tools & SDKs** link in the **Developer Tools** section on the AWS Documentation page (<http://aws.amazon.com/documentation/>).

For more information about the AWS SDKs, refer to the following documentation.

- SDK for Java
 - Developer guide – <http://docs.aws.amazon.com/AWSSdkDocsJava/latest/DeveloperGuide/welcome.html>
 - API documentation – <http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/index.html>

- SDK for .NET and Xamarin
 - Developer guide – <https://docs.aws.amazon.com/sdk-for-net/v3/developer-guide/welcome.html>
 - API documentation – <https://docs.aws.amazon.com/sdk-for-net/v3/developer-guide/welcome.html>
- SDK for Python
 - Developer guide – <https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>
 - API documentation – <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/core/index.html>

To learn about cloud development best practices refer to the Well-Architected Framework, Appendix: Questions and Best Practices section (<https://docs.aws.amazon.com/wellarchitected/latest/framework/appendix.html>).



Thank you for completing this module.