



Cloud9 Managed Credentials

Throughout this course we will be calling AWS APIs from programs or commands running in Cloud9 Instances. It is recommended that you do not hard code your AWS credentials into your code for any reason, and you should rely on **IAM Role Based Access** as much as possible. To read more about Role Based Access in AWS please click here: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html 

In Cloud9, by default, Roles are not being used for permissions, instead Cloud9 uses managed credentials. To read more about **Cloud9 Managed Credentials** click here:

<https://docs.aws.amazon.com/cloud9/latest/user-guide/how-cloud9-with-iam.html#sec-auth-and-access-control-temporary-managed-credentials> 

AWS SDK for Python

To install the AWS SDK for Python please read the following instructions:

<https://aws.amazon.com/sdk-for-python/> 

What is Boto3?

Boto3 is the AWS SDK for Python. It enables Python developers to create, configure, and manage AWS services, such as EC2 and S3. Boto3 provides an easy to use, object-oriented API through the resource, as well as low-level access to AWS services through the client.

Please ensure you get familiar with the boto3 documentation to get a better idea of how the SDK works, and what API calls can be made using the SDK.

Boto3 Documentation: <https://boto3.amazonaws.com/v1/documentation/api/latest/index.html> 

Client: Clients provide a low-level interface to AWS whose methods map close to 1:1 with service APIs. All service operations are supported by clients. Clients are generated from a JSON service definition file.

Resource: Resources represent an object-oriented interface to AWS. They provide a higher-level abstraction than the raw, low-level calls made by service clients.

Session: A session manages state about a particular configuration. By default, a session is created for you when needed. However, it's possible and recommended that in some scenarios you maintain your own session.

Sessions typically store the following:

Credentials

AWS Region

Other configurations related to your profile

Credentials: Boto3 credentials can be configured in multiple ways. Regardless of the source or sources that you choose, you must have both AWS credentials and an AWS Region set in order to make requests.

If you want to learn more about how boto3 got its name and how it came to be, read this blog post:

<https://aws.amazon.com/blogs/aws/now-available-aws-sdk-for-python-3-boto3/> 

The **AWS Systems Manager Parameter Store** API will be used throughout the course demos.

Parameter Store is a service that provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, Amazon Machine Image (AMI) IDs, and license codes as parameter values.

Read more about Parameter Store here:

<https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html> 

Below you can find information about the two AWS APIs we used in the demo code in the Boto3 Documentation:

Parameter Store Boto3 Documentation:

<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ssm.html> 

S3 Select Boto3 Documentation:

<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html> 

AWS Toolkit and AWS Serverless Application Model

The **AWS Toolkit for PyCharm** is an open source plug-in for the PyCharm IDE that makes it easier to create, debug, and deploy Python applications on Amazon Web Services. With the AWS Toolkit for PyCharm, you can

get started faster and be more productive when building applications with PyCharm on AWS. The toolkit provides an integrated experience for developing serverless applications, including assistance for getting started, step-through debugging, and deploying from the IDE.

The AWS Toolkit uses the **AWS Serverless Application Model** (AWS SAM) to create and manage AWS resources like AWS Lambda Functions.

AWS SAM is an open-source framework that you can use to build serverless applications on AWS.

The AWS Serverless Application Model provides a great way to codify your distributed application build, providing many of the benefits you find directly with Amazon CloudFormation.

A serverless application is a combination of Lambda functions, event sources, and other resources that work together to perform tasks. Note that a serverless application is more than just a Lambda function—it can include additional resources such as APIs, databases, and more.

You can use AWS SAM to define your serverless applications in a declarative way. AWS SAM consists of the following components:

AWS SAM template specification. You use this specification to define your serverless application. It provides you with a simple and clean syntax to describe the functions, APIs, permissions, configurations, and events that make up a serverless application. You use an AWS SAM template file to operate on a single, deployable, versioned entity that's your serverless application.

AWS SAM command line interface (AWS SAM CLI). You use this tool to build serverless applications that are defined by AWS SAM templates. The CLI provides commands that enable you to verify that AWS SAM template files are written according to the specification, invoke Lambda functions locally, step-through debug Lambda functions, package and deploy serverless applications to the AWS Cloud, and so on.

For more information on AWS SAM read here:

<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/what-is-sam.html>

For more information on the AWS ToolKit for PyCharm read here:

<https://docs.aws.amazon.com/toolkit-for-jetbrains/latest/userguide/welcome.html>

For information on installing Docker read here: <https://docs.docker.com/get-docker/>

For information on installing SAM for Windows, Linux, Mac read here:

<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-cli-install.html>