AWS Quick Guide

Reference Document for AWS (Internal Only)

# Getting Started with AWS

## Introduction to the Cloud

### What is Cloud Computing?

Cloud computing is on-demand delivery of resources, such as compute power, database, storage, applications and more, provided through the internet on a cloud services platform. Cloud services platforms provide rapid access to IT resources, which can be flexibly scaled where necessary.

### Types of Cloud Deployment

There are three main cloud deployment options, these are public, private and hybrid cloud deployments.

* **Public Cloud**: Public cloud resources (such as hardware, software and infrastructure) are owned, operated and maintained by a third-party service provider (like AWS). In a public cloud, you share the same hardware, storage, and network devices with other organisations or cloud “tenants”. The public cloud is the most type of cloud computing deployment
* **Private Cloud**: A private cloud consists of resources used exclusively by one business or organisation. The private cloud can be physically located at your organisation’s on-site datacentre (also known as on-premises), or it can be hosted by a third-party service provider (like AWS). In a private cloud, the services and infrastructure are always maintained on a private network and the hardware and software are dedicated solely to your organisation.
* **Hybrid Cloud**: A hybrid cloud combines a private cloud (or on-premises) with a public cloud. Hybrid clouds allow data and apps to move between the two environments. Many organisations choose a hybrid cloud approach due to business imperatives such as meeting regulatory and data sovereignty requirements, taking full advantage of on-premises technology investment, or addressing low latency issues.

Diagram

Description automatically generated

Figure 1: Diagram showing the interactions between an organisation and the public cloud, private cloud and a hybrid cloud.

### Cloud Services: Iaas, Paas and Saas

Cloud providers have several different service models to meet the needs of different users. Each service providers different levels of abstraction, control, management and flexibility.

* **Infrastructure as a Service** (IaaS): IaaS contains the basic building blocks for cloud IT, such as networking features, computers and data storage space.
* **Platform as a Service** (PaaS): PaaS removes the need to manage the underlying infrastructure (hardware and operating systems) to focus on application deployment and management.
* **Software as a Service** (SaaS): SaaS provides a completed product which is run and managed by the service provider – in most cases these are end user applications. SaaS removes the need to maintain a service or manage infrastructure, instead you focus on how the software will be used.

### What is AWS Cloud?

The AWS Cloud has over 200 cloud-based products that includes compute, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security, and enterprise applications.

AWS has a global infrastructure, meaning resources can be immediately created in the specific regions(s) most suitable for your use case, then quickly deleted when no longer necessary.

## Development in AWS

### How to Interact with AWS

With cloud computing you manage infrastructure logically (instead of physically). In AWS, to manage infrastructure using the AWS application programming interface (API). When creating, deleting, or changing any AWS resource, this is done using API calls directly to AWS.

These API calls can be done using:

* The AWS Management Console
* The AWS Command Line Interface (CLI)
* Integrated Development Environments (IDE) and IDE toolkits
* AWS Software Development Kits (SDKs)

### AWS Management Console

The AWS Management Console is a web-based console accessed through the browser. The console houses a broad range of service consoles for managing AWS resources. The Management Console is manually operated, so requires no scripting or syntax.

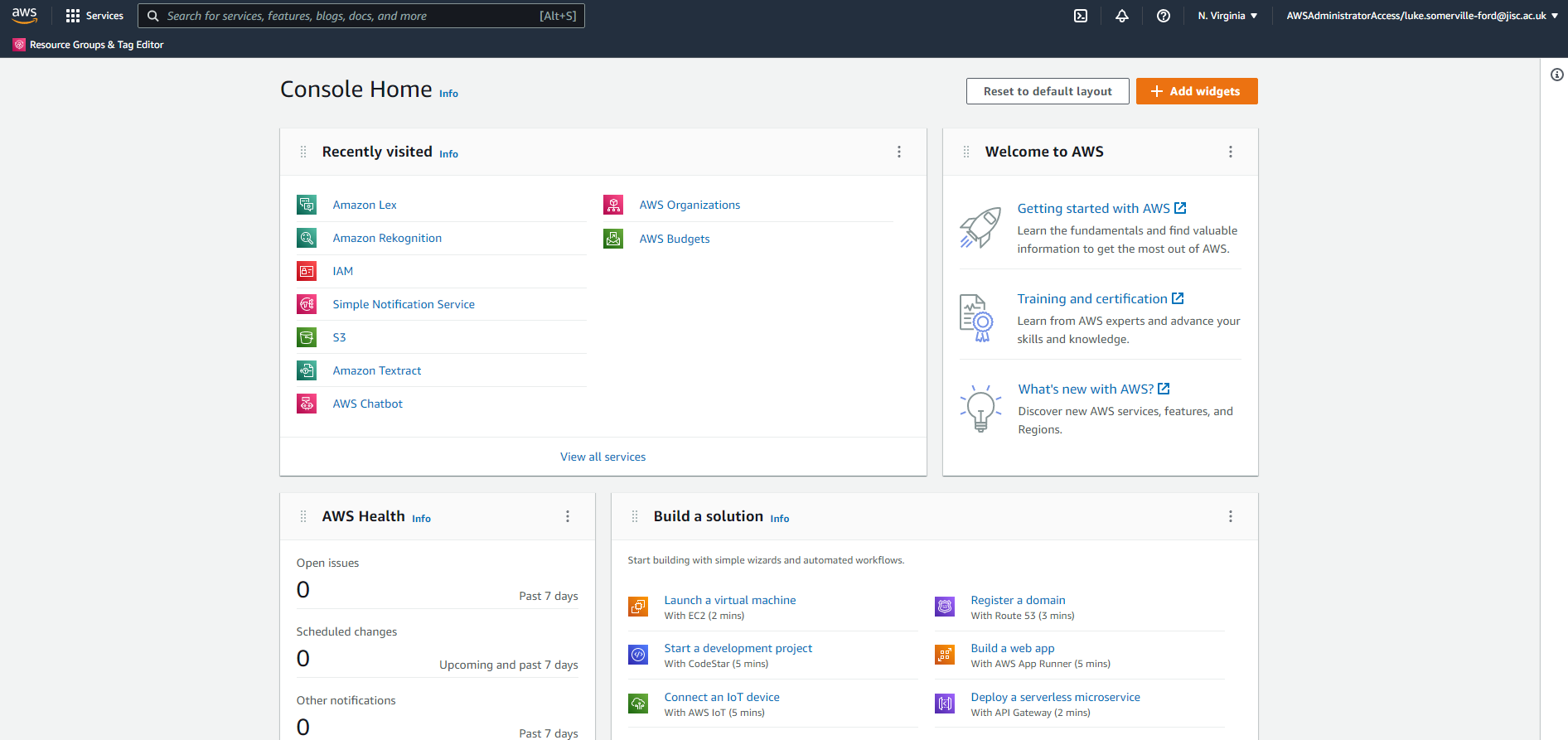


Figure 2: Screenshot of the AWS Management Console Home. The home page is customisable so may appear different.

### AWS CLI

The AWS CLI is an open-source tool that enables you to create and configure AWS services using commands via the command-line shell. You can run commands in Linux or macOS using common shell programs such as bash, zsh, and tcsh, or on Windows, at the Windows command prompt or in PowerShell. Using the AWS CLI, you can script or program your API calls. Instead of using a graphical user interface (GUI), you create commands with defined AWS syntax.

### IDE and IDE Toolkits

AWS offers support for IDEs and IDE toolkits so you can author, debug, and deploy your code on AWS from within your preferred environment. Supported IDEs and toolkits include AWS Cloud9, IntelliJ, PyCharm, Visual Studio, Visual Studio Code, Azure DevOps, Rider, and WebStorm.

### SDKs

SDKs are tools to interact directly with the AWS API programmatically. AWS creates and maintains SDKs for a variety of popular languages, including Python, JavaScript, Go and Java.

### Security

When you build applications on AWS, managing security and compliance is a shared responsibility between AWS and you. The distinction of responsibility is commonly referred to as security “of” the cloud compared to security “in” the cloud.

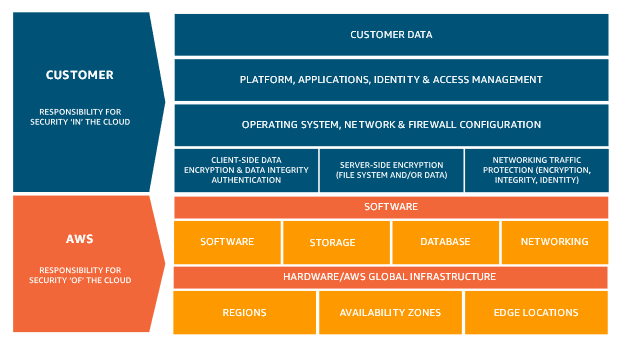


Figure : The shared responsibility model created by AWS to show the security responsibility of AWS and the customer.

## Amazon Lex

### What is Amazon Lex?

Amazon Lex is an AWS service for building conversational interfaces for applications using voice and text. Using the same conversational engine as Amazon Alexa, you can create natural language chatbots (with natural language processing (NLU) and automatic speech recognition (ASR)) into new and existing applications, with no deep learning expertise required.

Using the Amazon Lex console, you build, test and publish your text or voice chatbot. You can then add the conversational interfaces to bots on mobile devices, web applications, and chat platforms (for example, Facebook Messenger).

Amazon Lex provides pre-built integration with AWS Lambda. AWS Lambda is a service to automatically run code functions when called. You can create Lambda functions and add them as code hooks in your intent configuration to perform user data validation and fulfilment tasks. Integration with Lambda provides chatbots access to pre-built serverless enterprise connectors to link to data in SaaS applications, such as Salesforce.

### Amazon Lex Concepts and Terminology

* **Bot**: A bot performs automated tasks such as ordering a pizza, booking a hotel, ordering flowers, and so on. An Amazon Lex bot is powered by Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) capabilities. Each bot must have a unique name within your account.
* **Intent**: An intent represents an action that the user wants to perform. You create a bot to support one or more related intents. For example, you might create a bot that orders pizza and drinks. For each intent, you provide the following required information:
  + **Intent name**: A descriptive name for the intent. For example, OrderPizza. Intent names must be unique within your account.
  + **Sample utterances**: How a user might convey the intent. For example, a user might say "Can I order a pizza please" or "I want to order a pizza".
  + **How to fulfil the intent**: How you want to fulfil the intent after the user provides the necessary information (for example, place order with a local pizza shop). We recommend that you create a Lambda function to fulfil the intent. You can optionally configure the intent, so Amazon Lex simply returns the information back to the client application to do the necessary fulfilment.
* **Slot**: An intent can require zero or more slots or parameters. You add slots as part of the intent configuration. At runtime, Amazon Lex prompts the user for specific slot values. The user must provide values for all required slots before Amazon Lex can fulfil the intent.
  + For example, the OrderPizza intent requires slots such as pizza size, crust type, and number of pizzas. In the intent configuration, you add these slots. For each slot, you provide slot type and a prompt for Amazon Lex to send to the client to elicit data from the user. A user can reply with a slot value that includes additional words, such as "large pizza please" or "let's stick with small." Amazon Lex can still understand the intended slot value.
* **Slot type**: Each slot has a type. You can create your custom slot types or use built-in slot types. Each slot type must have a unique name within your account. For example, you might create and use the following slot types for the OrderPizza intent:
  + Size: With enumeration values Small, Medium, and Large.
  + Crust: With enumeration values Thick and Thin.

### How Amazon Lex Works

Following are the typical steps you perform when working with Amazon Lex:

1. Create a bot and configure it with one or more intents that you want to support. Configure the bot so it understands the user's goal (intent), engages in conversation with the user to elicit information, and fulfils the user's intent.
2. Test the bot. You can use the test window client provided by the Amazon Lex console.
3. Publish a version and create an alias.
4. Deploy the bot. You can deploy the bot on platforms such as mobile applications or messaging platforms such as Facebook Messenger.

## Programming Model

The following sections provide a brief overview of the three elements to the Programming Model – Model Building API Operations, Runtime API Operations, and Lambda Functions as Code Hooks. For more detailed information on each element, and the related APIs, please see the documentation under each heading.

### Model Building API Operations

AWS documentation: <https://docs.aws.amazon.com/lex/latest/dg/programming-model.html#programming-model-build-time-api>

To programmatically create bots, intents, and slot types, you can use the model building API operations. In addition, you can also use it to manage, update, and delete resources for your bot. You can use the model building API to create custom tools to manage your Amazon Lex resources. For example, there is a limit of 100 versions each for bots, intents, and slot types. You could use the model building API to build a tool that automatically deletes old versions when your bot nears the limit.

### Runtime API Operations

AWS documentation: <https://docs.aws.amazon.com/lex/latest/dg/programming-model.html#programming-model-runtime-api>

Your client application uses the runtime API to call a specific Amazon Lex bot to process *utterances* (user text or voice input). For example, if a user types "I want pizza." The client sends this user input to an Amazon Lex bot using one of the Amazon Lex runtime API operations. From the user input, Amazon Lex recognizes that the user request is for the OrderPizza *intent* defined in the bot. Amazon Lex engages the user in a conversation to gather the required information, or *slot data*, such as pizza size, toppings, and number of pizzas. After the user provides all of the necessary *slot data*, Amazon Lex either invokes the Lambda function code hook to fulfil the intent, or returns the *intent* data to the client, depending on how the *intent* is configured.

### Lambda Functions as Code Hooks

AWS documentation: <https://docs.aws.amazon.com/lex/latest/dg/programming-model.html#prog-model-lambda>

You can configure your Amazon Lex bot to invoke a Lambda function as a code hook. These code hooks can serve multiple purposes:

* Customise user interaction
* Validates the user’s input
* Fulfils the user’s intent

## Resources

[Getting Started with AWS Cloud Essentials (amazon.com)](https://aws.amazon.com/getting-started/cloud-essentials/?pg=gs)

[Public Cloud vs Private Cloud vs Hybrid Cloud | Microsoft Azure](https://azure.microsoft.com/en-gb/overview/what-are-private-public-hybrid-clouds/)

[What Is Amazon Lex? - Amazon Lex](https://docs.aws.amazon.com/lex/latest/dg/what-is.html)

[What is AWS Lambda? - AWS Lambda (amazon.com)](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html)