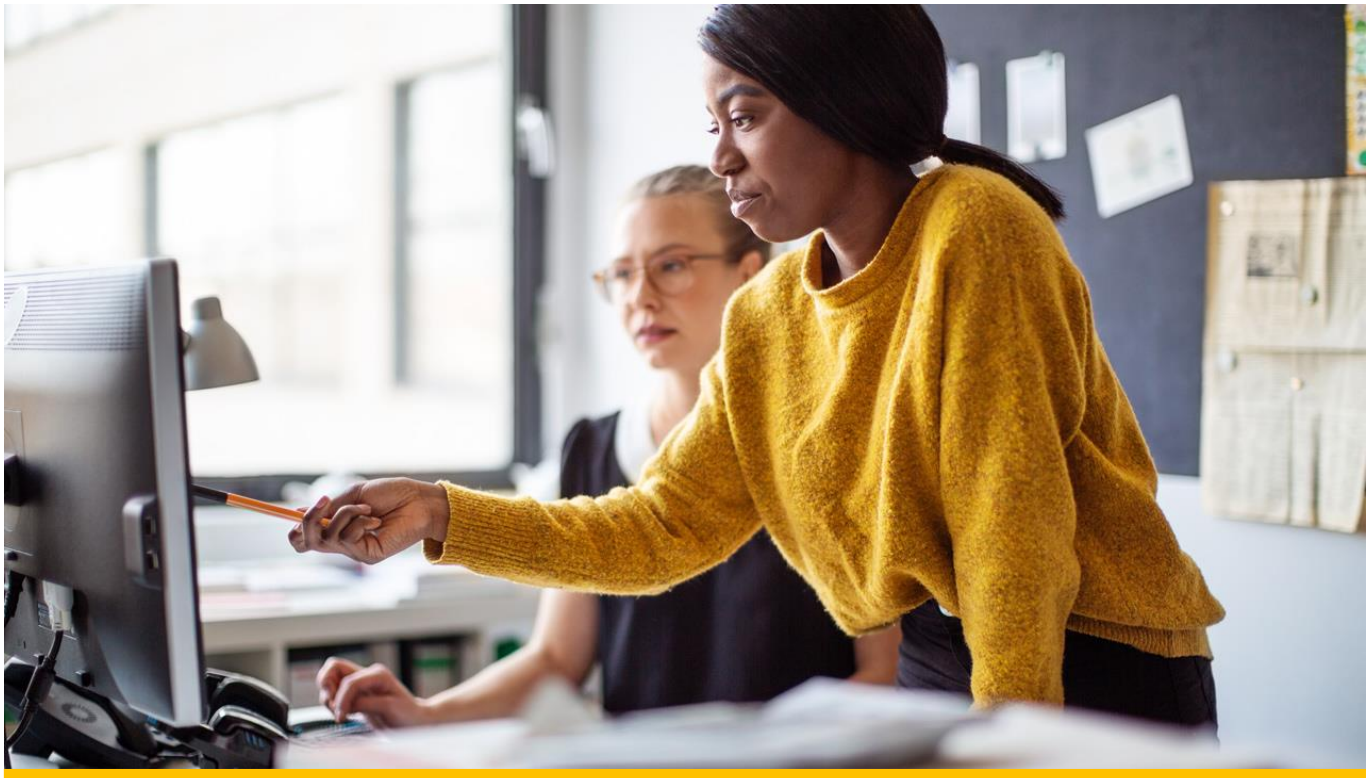


Serverless Deployments for Developers

SKILLSOFT ASPIRE JOURNEY

skillsoft | percipio

Aspire Journeys

Serverless Deployments for Developers

10

skillsoft

Earn a Badge

In a nutshell, serverless computing is cloud computing directed primarily at software developers. During this Journey, participants explore Serverless computing's fundamentals and how to migrate their legacy applications and services to many of the new cloud platforms. We will explore the most common development languages for Serverless computing and explore each of the major cloud providers and how they are leveraging Serverless technologies.

View Less

23 courses | 37h 27m 12s

Tracks

Track 1: Beginning with Serverless Solutions

In this track of the Serverless Deployments for Developers Aspire journey, the focus will be on migrating to serverless, portability in the cloud, and serverless cloud development.

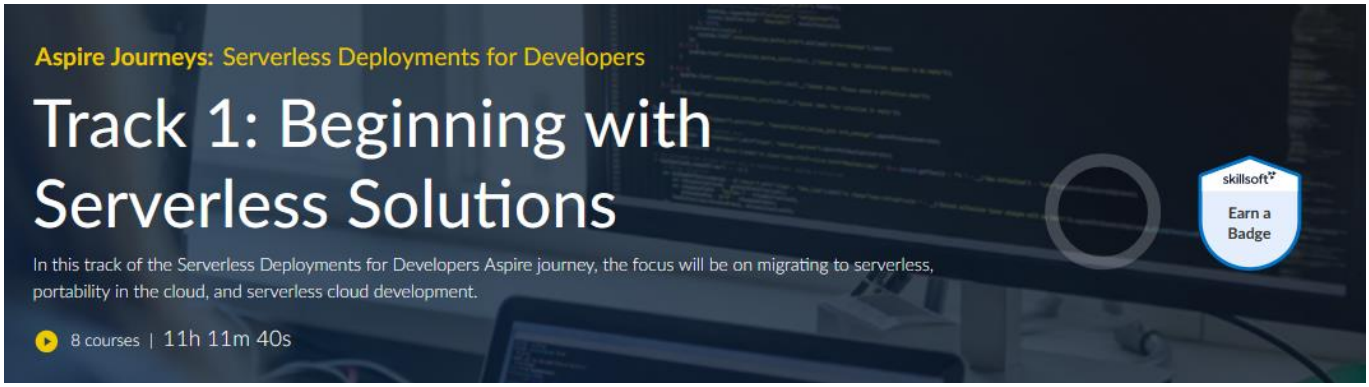
Explore 8 courses | 11h 11m 40s

Track 2: Developing Serverless Applications in the Cloud

In this track of the Serverless Deployments for Developers Aspire journey, the focus will be on serverless essentials, building AWS serverless applications, Azure serverless DevOps for developers...

View More

Explore 15 courses | 26h 15m 32s



Migrating to Serverless: Adopting Serverless Computing

Objectives:

- recall the journey of cloud computing from the shared mainframes era to managed cloud solutions
- describe the cloud application maturity model and cloud-native stack model with a focus on long-term observable trends in cloud systems engineering
- recognize different types of data centers, their hardware requirements, and the layers of data center network architecture
- outline the evolution of cloud data center models for autonomic and agile elastic resource provisioning
- recognize considerations for outsourcing IT operations and strategic value of IT activities, and risks associated with IT outsourcing
- identify the features and advantages of serverless architecture
- differentiate between the characteristics of serverless computing and Platform as a Service (PaaS)
- distinguish between the features and use cases of serverless and container deployment
- recognize differences between DevOps and NoOps and describe the impact of NoOps on serverless deployment
- outline characteristics, benefits, and applications of serverless architecture
- describe steps and considerations involved in migrating from an existing service to serverless
- list steps and best practices for migrating from monolithic apps to serverless architecture

Migrating to Serverless: Implementing Serverless Solutions

Objectives:

- list common architectures that integrate serverless approaches and identify factors to consider when implementing a serverless architecture
- outline considerations for selecting a serverless implementation
- recognize prominent serverless computing platforms and the advantages of runtime
- describe the features of different serverless systems and how they help facilitate computing, storage, and queue processing
- identify distinguishing characteristics and architectural components of serverless platforms
- define the concept of FaaS and recall programming models that help developers in selecting serverless architecture and runtime
- recognize the serverless services provided by AWS for all layers of an application stack
- outline the features and benefits of AWS Lambda and list its use cases
- differentiate between serverless computing pricing models afforded by various cloud providers and identify best practices that help control overall serverless costs
- demonstrate how to set up a development environment on AWS for building serverless applications
- list the key categories of serverless hybridization and potential use cases for implementing multi-cloud serverless solutions

Serverless Solutions: Evolving a Serverless Mindset

Objectives:

- describe technology evolution and benefits of the serverless mindset
- distinguish between the features of service-oriented and serverless architecture and identify the business benefits of the latter
- outline the changing dimensions of application operation and usability offered by serverless application architectures
- list use cases for serverless compute products along with their associated benefits
- identify the differences between containerized and serverless implementation and describe the considerations for their selection as an appropriate strategy
- recall principles and processes of DevOps and CloudOps that help evaluate the level of existing solutioning mindset
- outline the impact of serverless on DevOps and CloudOps from the perspective of technical implementation and resource management
- recognize how serverless architectures change development processes and impact businesses
- define the concept of NoOps and potential impacts of the serverless trend on development and service delivery
- identify the influence of serverless on DevOps practices and describe the DevOps pipeline for serverless application
- recognize the ecosystem of serverless technologies and elements that drive their mainstream adoption
- recall prominent serverless development and deployment technologies and map them to serverless solutions

Portability in the Cloud: Managing Cloud Portability & Interoperability

Objectives:

- define portability and list challenges and elements to be considered when designing and developing a portable software
- recognize key principles and classes of strategies that help achieve greater portability in software and service units
- outline the development lifecycle and role of specification in maximizing portability
- describe the elements and categories of cloud computing portability and interoperability
- identify application design principles for designing interoperable software applications that can be integrated with other products
- recognize the principles of open platform and key components that are required to provide a basic open platform configuration
- describe the OpenAPI specification and features of prominent tools that work with OpenAPI
- demonstrate how to install Swagger and design API using the Swagger editor
- generate server and client code based on API definitions using Swaggerhub
- create and fork API using Swaggerhub
- configure a development environment in GCP and create, configure, and deploy API to the gateway
- create an HTTP API using the AWS management console and demonstrate the steps to control and manage access to the HTTP API in API Gateway using the access control IAM authorizer
- recall the concept of containerization and components of a typical containerization platform that provides portability
- identify the essential characteristics of Docker and Kubernetes that allow application portability in the cloud

Portability in the Cloud: Application Portability in Multi-cloud

Objectives:

- recognize multi-cloud architectures and key considerations while adopting multi-cloud for portability and flexibility
- recall benefits, challenges, and use cases of multi-cloud
- describe the approach of using an open-source serverless framework for developing and deploying serverless computing solutions across cloud service providers
- configure Kubernetes environment for multi-cloud application portability
- compare multi-cloud architecture patterns for building portable applications
- use Python to write applications with the functional programming approach and demonstrate the approach to build and run stateless containers that can be invoked via HTTP requests in the cloud
- use CNCF-compatible buildpacks to build source code into container images that are designed to run on Google Cloud Platform (GCP)
- outline the role of containers in a multi-cloud environment and aspects to consider when selecting a cloud infrastructure for containerized applications

Serverless Cloud Development: Runtime Environments

Objectives:

- list the major serverless providers and the popular serverless frameworks that enable developers to explore the aspects of serverless computing
- recall the languages that can be used to develop serverless applications for diversified serverless providers
- compare the prominent AWS Lambda–supported languages from the perspective of cold start performance, warm performance, cost, and ecosystem
- recognize the types of Cloud Functions provided by GCP to implement serverless applications along with the language runtimes supported by GCP to write Cloud Functions
- describe the features of Azure Functions and outline how to select the right programming language to implement Azure Functions Runtime
- set up a local development environment to build serverless applications using AWS and prominent open source tools
- set up a local development environment for Google Cloud Functions to manage Cloud Functions deployment and use for local testing and debugging
- set up a local development environment to create, test, and debug Azure Functions app projects
- create and deploy serverless Azure Functions in Python using Visual Studio Code
- install Serverless Framework Open Source CLI and deploy a sample service in the cloud that reports deployment information and operational metrics to the Serverless Framework dashboard

Serverless Cloud Development: Serverless Applications with Diverse Languages & Environments

Objectives:

- install and configure node dependencies to build and deploy REST APIs using Serverless, Express, and Node.js on AWS
- convert an existing Node and Express-based application to a Serverless-friendly application in the cloud
- use Node.js to set up a simple HTTP GET endpoint on GCP
- deploy a HTTP Node.js Azure Function and demonstrate how to read properties and set a result back to Azure
- set up a simple HTTP GET endpoint using Java and Maven or Gradle to build serverless applications on AWS
- set up a simple HTTP GET endpoint using Ruby on AWS to facilitate a serverless mechanism

- create a simple REST API with Python running on AWS Lambda and API Gateway using the traditional Serverless Framework
- configure and set up a simple Python HTTP GET endpoint on GCP Cloud Functions
- build and publish a Java Function project to Azure Functions using the Gradle command-line tool
- build, deploy, and test a sample serverless application using AWS SAM
- build a Web API using Go and AWS Lambda
- evaluate serverless computing environments invoking functions in parallel to demonstrate the performance and throughput of serverless computing for distributed data processing

Final Exam: Beginning with Serverless Solutions

Objectives:

- compare multi-cloud architecture patterns for building portable applications
- compare the prominent AWS Lambda–supported languages from the perspective of cold start performance, warm performance, cost, and ecosystem
- configure and set up a simple Python HTTP GET endpoint on GCP Cloud Functions
- configure Kubernetes environment for multi-cloud application portability
- convert an existing Node and Express-based application to a Serverless-friendly application in the cloud
- create a simple REST API with Python running on AWS Lambda and API Gateway using the traditional Serverless Framework
- define portability and list challenges and elements to be considered when designing and developing a portable software
- define the concept of FaaS and recall programming models that help developers in selecting serverless architecture and runtime
- deploy an HTTP Node.js Azure Function and demonstrate how to read properties and set a result back to Azure
- describe technology evolution and benefits of the serverless mindset
- describe the approach of using an open-source serverless framework for developing and deploying serverless computing solutions across cloud service providers
- describe the elements and categories of cloud computing portability and interoperability
- describe the features of Azure Functions and outline how to select the right programming language to implement Azure Functions Runtime
- describe the features of different serverless systems and how they help facilitate computing, storage, and queue processing
- describe the OpenAPI specification and features of prominent tools that work with OpenAPI
- describe the purpose of different serverless pattern categories
- differentiate between the characteristics of serverless computing and Platform as a Service (PaaS)
- distinguish between the features and use cases of serverless and container deployment
- distinguish between the features of service-oriented and serverless architecture and identify the business benefits of the latter
- identify application design principles for designing interoperable software applications that can be integrated with other products
- identify the differences between containerized and serverless implementation and describe the considerations for their selection as an appropriate strategy
- identify the features and advantages of serverless architecture
- install and configure node dependencies to build and deploy REST APIs using Serverless, Express, and Node.js on AWS
- list common architectures that integrate serverless approaches and identify factors to consider when implementing a serverless architecture

- list the major serverless providers and the popular serverless frameworks that enable developers to explore the aspects of serverless computing
- outline characteristics, benefits, and applications of serverless architecture
- outline considerations for selecting a serverless implementation
- outline the changing dimensions of application operation and usability offered by serverless application architectures
- outline the development lifecycle and role of specification in maximizing portability
- recall benefits, challenges, and use cases of multi-cloud
- recall prominent serverless development and deployment technologies and map them to serverless solutions
- recall the concept of containerization and components of a typical containerization platform that provides portability
- recall the journey of cloud computing from the shared mainframes era to managed cloud solutions
- recall the languages that can be used to develop serverless applications for diversified serverless providers
- recognize differences between DevOps and NoOps and describe the impact of NoOps on serverless deployment
- recognize how serverless architectures change development processes and impact businesses
- recognize key principles and classes of strategies that help achieve greater portability in software and service units
- recognize prominent serverless computing platforms and the advantages of runtime
- recognize the types of Cloud Functions provided by GCP to implement serverless applications along with the language runtimes supported by GCP to write Cloud Functions
- use Node.js to set up a simple HTTP GET endpoint on GCP

Developing Serverless Applications: AWS Serverless Essentials for Developers

Objectives:

- describe the features and benefits of AWS Serverless Application Model (SAM) along with the key components that can be used to define, test, and deploy serverless applications
- list key components of AWS SAM specification that are used to define serverless applications
- identify the role of AWS Serverless Application Repository and the mechanism of storing, sharing, assembling and deploying serverless architectures
- publish a sample application to the AWS Serverless Application Repository
- deploy applications from the AWS Serverless Application Repository
- outline the benefits of messaging with AWS and the features of SNS and SQS that help implement fully-managed messaging service for application-to-application and application-to-person communication
- implement fanout messaging using Amazon simple notification service (SNS) and Amazon simple queue service (SQS)
- implement message filtering pattern using Amazon simple notification service (SNS)
- list the core components, features, and benefits of AWS DynamoDB
- create a simple table and add, query, and delete data using the DynamoDB console
- use Python and AWS SDK for Python to enable DynamoDB APIs to interact with tables created in DynamoDB
- recognize the prominent AWS developer tools services and the features of the AWS cloud development toolkit to host code as well as build, test, and deploy applications to AWS
- work with AWS cloud development kit (CDK) to create rest APIs with GET endpoint

Serverless Applications: Implementing RESTful API using AWS

Objectives:

- recall the history of REST API and the features of REST that make it different from SOAP API
- recognize the goals of the REST architecture style and the design rules applied to establish the distinct characteristics of the REST architectural style
- describe the concept and approach of uniform contract and service modeling using RESTful API
- recognize the benefits of using REST API for modern applications and the steps that can be adopted to transition to API-driven development
- describe the features and architecture of AWS API Gateway that help accept and process concurrent API calls
- outline the process that needs to be adopted to design, build, and optimize API Gateway for creating REST APIs to implement request/response design models
- create HTTP APIs using the AWS Management Console

- create API Gateway APIs with private integration to provide API consumers with access to HTTP/HTTPS resources within Amazon Virtual Private Cloud
- create API Gateway APIs to expose other AWS services
- demonstrate the steps involved in building API Gateway REST APIs with Lambda integration
- recall the concept of serverless computing along with the AWS services that can be used to implement serverless computing
- describe the features and building blocks of AWS Lambda along with the benefits afforded by AWS Lambda to implement serverless compute
- implement AWS Lambda using Node.js with the AWS Management Console
- recognize the key features and benefits of DynamoDB and specify why DynamoDB is an essential part of the serverless ecosystem
- create Lambda functions that can consume events from Amazon DynamoDB streams
- implement REST APIs that use AWS API Gateway to route requests to Lambda functions that can query DynamoDB
- recognize the approach of setting up authentication and authorization mechanisms in AWS serverless implementations
- configure a Lambda authorizer using the API Gateway console
- set up request validators in an API's OpenAPI definition file and import the OpenAPI definitions into API Gateway
- use Amazon S3 to trigger an AWS Lambda function when files are being uploaded in Amazon S3 buckets
- configure a destination and Lambda function to send invocation records asynchronously to services
- invoke and execute Lambda functions at regular intervals using scheduled events
- customize the content delivered by CloudFront using the Lambda compute service

Serverless Applications: Debugging & Monitoring AWS Serverless Implementations

Objectives:

- list and describe the various application testing mechanisms
- describe the concept of API testing, its benefits, and the various types of API testing conducted to provide bug-free APIs
- describe the cloud approach to software testing and name the various test types performed on cloud applications, APIs, and services
- recognize the challenges a test engineer faces when testing cloud applications and the role of mock and dummy data in testing APIs
- list and describe the various testing strategies used in local and cloud environments
- outline the process and benefits of implementing automated testing and the prominent tools and frameworks used in this process
- work with AWS SAM to test serverless applications
- perform integration tests against local Lambda endpoints using AWS SAM
- generate and customize event payloads for a number of AWS services to simplify testing of Lambda functions
- test and debug Node.js serverless applications locally using the AWS toolkit for Visual Studio Code and SAM CLI
- list and describe the prominent AWS services that can be used to monitor AWS applications
- work with AWS SAM CLI commands to fetch, tail, filter, and highlight Lambda function errors
- recognize the role played by the AWS Lambda console in monitoring AWS Lambda functions
- use the AWS Lambda console's monitoring dashboard to monitor AWS Lambda functions and applications

Azure Serverless Development Principles: Serverless Solutions Essentials

Objectives:

- recognize the prominent Azure serverless solutions and the serverless solution architectures used to develop applications for productive and optimized resource usage
- list the critical factors that drive architects and developers in selecting Azure serverless for solutioning along with the different approaches for productively managing infrastructures
- recognize the prominent Azure serverless execution environments and describe the key features of Azure Functions and Azure Logic Apps
- describe the key features of Azure serverless services that eliminate the need to manage infrastructure and help orchestrate and build serverless application workflows
- list the prominent tools that can be used with the DevOps methodology to develop serverless applications
- set up, create, and deploy Azure serverless service using Visual Studio Code
- develop and deploy Azure Serverless services using Azure tools and Visual Studio Code
- create, configure, test, and publish Azure functions using the Visual Studio IDE

Azure Serverless Development Principles: Serverless Compute Essentials

Objectives:

- recognize the different approaches of managing compute with Azure and compare the differences between Azure compute services and Azure serverless architecture from the perspective of infrastructure management
- recall the purpose of Azure Functions and compare it with Azure WebJobs
- create and test Azure functions locally using the Azure CLI
- create, test, and deploy Azure functions with C# using Visual Studio Code
- create, test, and deploy Azure functions with Python using Visual Studio Code
- work with the test tools provided by Azure to validate Azure functions
- describe the concept and types of durable functions used to implement complex stateful functions in Serverless environments
- orchestrate long-running workflows as a set of activities using durable functions
- describe the security strategies used to run secure function code and the role of Azure App Service in securing functions
- describe the characteristics and features of Azure App Service that help build, deploy, and scale web applications
- build and deploy ASP.Net core web applications in Azure
- create and configure Azure App Service using the Azure Portal
- create Azure App Service using the ASP.Net template provided by Visual Studio IDE

Azure Serverless Development Principles: Serverless Database & Storage

Objectives:

- recall the essential relational, non-relational, and storage services used to build serverless applications requiring persistence
- list and describe the features of prominent Azure services that are based on relational and non-relational data stores
- list and describe the different types of Azure storages and Azure SQL databases
- create and configure different types of Azure storages using the Azure portal
- create functions that can be triggered when files are uploaded to or updated in Blob storage containers
- create serverless Azure SQL databases
- create cloud-native apps using Azure Functions in order to create, design, and connect to Azure databases
- describe the features and benefits of using Azure Cosmos DB

- create Azure Cosmos DB accounts, databases, containers, and items using the Azure portal
- configure and use Azure Storage Explorer and Azure Cosmos DB Explorer to manage Azure Cosmos DB resources
- create Node.js web apps that can be used to manage SQL API accounts in Azure Cosmos DB
- describe the concept and basic features of a non-relational databases along with the different approaches of implementing them using Azure Cosmos DB
- use Node.js to perform CRUD operations on Azure Cosmos DB resources

Applications in the Cloud: Azure Serverless Orchestration Essentials

Objectives:

- recall the serverless architectures implemented using Azure and the pros and cons associated with them
- describe the orchestration and integration process in Azure
- use Azure Portal and Azure CLI to create serverless workflows with Azure Logic Apps
- use Visual Studio Code to create serverless workflows to automate app and data integration tasks and processes
- use Azure Logic Apps and Azure Functions to create basic serverless workflows to consume Azure services
- create conditional statements to control workflow actions in Azure Logic Apps
- build Logic Apps to orchestrate workflows running on defined recurring schedules
- describe the high-level concept of using triggers and bindings in Azure Functions
- integrate and automate serverless application workflows using Azure Logic Apps and triggers
- create back-end compute services using Azure function triggers and bindings
- develop unit tests for Azure functions with HTTP triggers
- describe the features and benefits of Azure API Management along with the key components of Azure API Management
- create Azure API Management service instances using Azure Portal
- recognize the features and characteristics of the Azure SignalR service along with the use cases that can be implemented using Azure SignalR
- implement real-time chatrooms using Azure Functions and Azure SignalR API

Applications in the Cloud: Azure Serverless DevOps Essentials

Objectives:

- list and describe the features of the prominent Azure DevOps services used to build and ship applications quickly
- recognize the features of Azure Pipelines that facilitate continuous build, test, and diversified platform deployment
- create Azure pipelines to build a GitHub repository and manage the pipelines using the Azure CLI
- use Azure Pipelines to build, test, and deploy JavaScript and Node.js apps, applying CI/CD practices
- recognize the key capabilities of Azure Test Plans and the critical features of planned and exploratory testing
- create test labs using the Azure portal
- use Azure DevOps services to implement end-to-end automated testing
- implement continuous integration pipelines for Azure Functions using Azure Pipelines
- implement continuous deployment pipelines for Azure Functions using Azure Pipelines
- automate the deployment of Azure functions using Azure Pipelines

Azure Serverless Services: Messaging & Monitoring Essentials

Objectives:

- outline the concept and capabilities of Azure Event Grid that help build applications with Event-based architectures
- compare the differences between messages and events and list the prominent messaging services provided by Azure to manage them
- compare the features of prominent messaging services provided by Azure to be able to evaluate and select the right service for specific scenarios
- subscribe to events generated by Azure Blob storage and trigger events using Azure portal
- create and manage system topics using Azure portal
- define the concept and advantages of Azure Notification Hubs along with the working mechanism and challenges of push notifications
- create Notification Hubs in Notification Hubs namespaces
- set up push notifications in Azure Notification Hubs
- configure Azure Notification Hubs to broadcast Push notifications to all devices that are running applications
- compare the key differences between Amazon SNS and Azure Notification Hubs
- list and describe the key features of prominent Azure Serverless Monitoring solutions
- describe the features and use cases of Azure Monitor that maximize application availability and performances
- recall the high-level overview, data collection mechanisms, and types of data collected by Azure Monitor
- recognize data generated by Azure resources and examine how to use the features of Azure Monitor to analyze the generated data
- use Azure Monitor to collect and analyze monitoring data from Azure Virtual Machines (VMs) to maintain optimum health
- describe the working mechanism of Application Insights and critical metrics evaluated to monitor live applications
- configure Azure Monitor and Application Insights to monitor Serverless applications
- use Azure portal to create Log Analytics workspaces and configure diagnostic settings to send Activity logs to Azure Monitor

Serverless App Development: Implementation in Google Cloud Platform

Objectives:

- list and describe the various options provided by Google Cloud Platform (GCP) for hosting applications
- identify the key features and benefits of using Google Cloud's serverless platforms and list the prominent serverless products provided by Google Cloud Platform
- name the key features of Google Cloud Functions and the elements of this product that facilitate writing serverless code to connect and extend cloud services
- describe the prominent use cases of Cloud Functions and how these help developers
- create and deploy Node.js Cloud Functions using the Cloud Console
- set up local development environments to create, test, and deploy Google Cloud Functions
- list the key features and benefits of Google App Engine and, through use cases, describe how App Engine helps developers build applications with zero server management and zero configuration deployments
- list and compare the different types of environments that are supported by App Engine and outline how to choose the right environment
- write web applications, configure the deployment on App Engine, and then deploy and test the web applications
- state the features and benefits of using Cloud Run that provide a managed compute platform to run applications
- deploy containers from a Container Registry repository to Cloud Run

- outline the role of Firebase in building and deploying applications for production along with the pros and cons of using Firebase
- add Firebase to existing JavaScript projects
- describe the features of Cloud Datastore, compare it with Firestore and traditional databases, and list the use cases of Cloud Datastore
- store and query data in Firestore in Datastore mode using the Google Cloud Console
- describe the benefits of using Google Cloud Platform (GCP) serverless services for analytics and list the products and features provided by GCP to implement analytics
- outline how the features and benefits of Dataflow help implement serverless unified stream and batch data processing
- create streaming pipelines using Google-provided Dataflow templates
- outline the concept of a serverless microservice, how they work, and the benefits of using GCP serverless products to implement serverless microservices
- recognize the DevOps capabilities provided by GCP and describe how serverless deployments can be leveraged to improve DevOps productivity

Serverless App Development: Applications with Google Cloud Functions

Objectives:

- outline the key features and use cases of Google Cloud Functions to recognize the appropriate scenarios for using it
- state the runtimes supported by Cloud Functions and list the different types of Cloud Functions that can be written to run codes with zero server management
- demonstrate the process of creating HTTP functions using Node.js and Python runtimes
- demonstrate the process of creating HTTP functions using Java and Go runtimes
- demonstrate the process of deploying Cloud Functions from the local machine, source repository, and GCP console
- recognize the key features and components of Google Cloud Functions frameworks
- list and describe the native trigger mechanisms and the event types supported by those triggers that can be used by Google Cloud Functions
- write, deploy, and trigger background Cloud functions using Cloud Pub/Sub triggers
- write, deploy, and trigger background Cloud functions with Cloud Storage triggers to respond to Cloud Storage events
- compare the features of Cloud Firestore and Firebase Realtime Database to support real-time data syncing
- set up Cloud Firestore, add data, and view data using the Firestore console
- describe the testing approaches and the common types of tests that can be adopted to test Cloud Functions
- write unit tests for the HTTP-triggered and event-driven functions
- demonstrate the use of Log Explorer in Google Cloud to check the logs generated by Google Cloud services
- write Cloud functions and integrate them with front-end applications that are written using ReactJS
- demonstrate the use of Log Explorer in Google Cloud to retrieve audit log entries of Cloud projects

Serverless App Development: Implementation with Google App Engine

Objectives:

- describe the high-level features of Google App Engine and name the components used to build serverless web applications in GCP
- list App Engine environments, compare their features, and recognize the application requirements fulfilled by each one
- set up a local computer for developing, deploying, and managing web apps in App Engine
- describe how to structure the services and related resources of applications for App Engine
- create a cloud project, set up the App Engine resources, and then write and locally test basic web applications
- deploy simple web applications in App Engine
- write web applications that connect, store, and retrieve data from Datastore
- recognize the mechanism and recommendations for versioning APIs and describe how to deploy multiple API versions to the same App Engine version
- define the concept of traffic splitting and recognize the different methods used for splitting traffic in App Engine that help adopt A/B testing and blue/green deployment strategies
- list and describe the various mechanisms that can be used to secure web applications on App Engine
- recognize the key features of Google Cloud's operations suite that provide comprehensive cloud logging and error reporting capabilities
- demonstrate steps to configure monitoring and logging for Google serverless applications

Serverless Frameworks: Serverless Development Using Open-source Frameworks

Objectives:

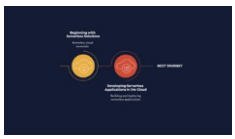
- recognize the features afforded by open-source frameworks and the Serverless Framework design characteristics to consider when developing serverless applications
- list some prominent open-source serverless frameworks along with the key features afforded by them
- differentiate between the prominent open-source serverless frameworks and evaluate them based on their features, use cases, and domains to be able to select the best fit framework
- compare the differences between open-source serverless frameworks and the frameworks and services provided by cloud providers
- describe the features, internal architecture, and programming model of the Apache OpenWhisk framework
- describe the key features of the various tools that can be used to work with the OpenWhisk framework
- list the critical components of the Apache OpenWhisk framework and describe how Apache OpenWhisk executes an action
- create and invoke actions using the Apache OpenWhisk CLI and the Whisk Deploy utility
- develop Functions as a Service using the Apache OpenWhisk framework
- describe the key features of Fn Project and list the critical components of Fn Project that help accelerate the adoption of serverless
- set up Fn Project to create, deploy, and invoke functions using the local Fn server
- outline a high-level overview of the internals of Fission along with the features afforded by the core components of Fission
- install and configure Fission on a Kubernetes cluster
- write short-lived functions in Node.js, map them to HTTP requests, and deploy them on Fission

- recognize the key features of Kubeless along with the architectural design of Kubeless and the directory structure of the Kubeless repository
- describe the approach of deploying, auto scaling, API routing, and monitoring in Kubeless along with the Kubernetes resources that are leveraged by Kubeless
- install and configure Kubeless to enable serverless deployments
- deploy Kubeless functions using the Kubernetes API

Serverless Frameworks: Optimizing Serverless Applications

Objectives:

- describe the key features, capabilities, and characteristics of AWS Chalice that enable the development of serverless applications
- install and use the Chalice command line utility to create and deploy basic REST APIs
- describe the features of Claudia.js and list the alternative frameworks for Claudia.js that can be used to deploy serverless applications
- deploy simple Node.js microservices to AWS Lambda using Claudia.js
- recognize the critical features afforded by OpenFaaS along with the architecture and components of OpenFaaS used to manage serverless workflows
- install OpenLambda and create an OpenLambda environment on Ubuntu
- describe the features of the prominent tools that help maintain function-based services and improve development workflows
- recognize the critical metrics that need to be evaluated to identify the performances of serverless applications
- list and describe the recommended serverless performance and optimization strategies that can be adopted across serverless frameworks and platforms
- describe the features of the prominent tools that can be used to monitor and debug serverless applications
- recognize the recommended techniques that can be used to establish trade-off between performances and costs
- configure Sentry to monitor serverless applications



Final Exam: Developing Serverless Applications in the

Objectives:

- compare the differences between fully managed Cloud Run and Cloud Run for Anthos that helps serverless architect select the right architecture
- compare the differences between messages and events and list the prominent messaging services that are provided by Azure to manage messages and events
- compare the differences between the prominent open-source serverless frameworks and evaluate them based on their features, use cases and domains to be able to select the best fit framework
- compare the features of Cloud Firestore and Realtime databases offered by Firebase that supports realtime data syncing
- compare the features of the prominent messaging services that are provided by Azure to be able to evaluate and select the right service for specific scenarios
- create and test Azure functions locally using the Azure CLI
- create a simple table and add, query, and delete data using the DynamoDB console
- create Azure CosmosDB accounts, databases, containers and items using the Azure portal
- create HTTP APIs using the AWS Management Console
- define the concept of traffic splitting and recognize the different methods that can be used for splitting traffic in App Engine that helps to adopt A/B testing and Blue/green deployment strategies
- describe the concept and basic features of a Non-relational database along with the different approaches of implementing them using Azure CosmosDB
- describe the concept of API testing, its benefits, and the various types of API testing conducted to provide bug-free APIs

- describe the end-to-end lifecycle of a container on Cloud Run
- describe the features and architecture of AWS API Gateway that help accept and process concurrent API calls
- describe the features and benefits of Azure API Management along with the key components of Azure API Management
- describe the features and benefits of the AWS Serverless Application Model (SAM) along with the key components that can be used to define, test, and deploy serverless applications
- describe the features and benefits of using Azure CosmosDB
- describe the features of Azure Monitor and the use case scenarios of using Azure Monitor to maximize application availability and performances
- describe the features of the prominent tools that can be used to monitor and debug Serverless applications
- describe the high-level concept of using triggers and bindings in Azure functions
- describe the key features of Fn Project and list the critical components of Fn Project that helps accelerate the adoption of serverless
- describe the key features of the various tools that can be used to work with the OpenWhisk framework
- describe the prominent use cases of Cloud Functions that helps Developers recognize the various scenarios of using Google Cloud Functions
- describe the security strategies that can be used to run secure function code along with the role of App Service in securing functions
- describe the testing approaches and the common types of tests that can be adopted to test Cloud Functions
- identify the key features and benefits of using Google Cloud's serverless platforms and list the prominent Serverless products provided by Google Cloud Platform
- install and configure Fission on a Kubernetes cluster
- list and describe the different types of Azure storages and Azure SQL databases
- list and describe the features of prominent Azure DevOps services that are being used to build and ship applications faster
- list and describe the key features of the prominent Azure Serverless Monitoring solutions
- list and describe the various application testing mechanisms
- list and describe the various mechanisms that can be used to secure web applications on App Engine
- list and describe the various options provided by Google Cloud Platform for hosting applications
- list the core components, features, and benefits of AWS DynamoDB
- list the critical factors that drives architects and developers in selecting Azure Serverless for solutioning, along with the different approaches for productively managing infrastructures
- list the key features and benefits of Google App Engine and describe using use cases how the App Engine helps developers build applications with zero server management and zero configuration deployments
- list the key features of Google Cloud Functions along with the essential elements of Google Cloud Functions that can be used to write Serverless code to connect and extend cloud services
- list the prominent open-source serverless frameworks along with the key features afforded by them
- list the prominent tools that can be used with the DevOps methodology to develop Serverless applications
- outline the process and benefits of implementing automated testing and the prominent tools and frameworks used in this process
- recall the App Engine environments and compare the features of those environments to recognize the application requirements that can be fulfilled with each environment

- recall the concept and capabilities of Azure Event Grid that helps build applications with Event-based architectures
- recall the concept of Azure Functions and compare the differences between Azure Functions and Azure Webjobs
- recall the features of Azure DevOps pipeline that helps facilitate continuous build, test and deploy to diversified platforms
- recall the history of REST API and the features of REST that make it different from SOAP API
- recall the key features of Google Cloud Function along with the prominent use cases of Google Cloud Function to recognize the appropriate scenarios of using it
- recall the prominent serverless solutions afforded by GCP that can be used to build, develop and deploy functions and applications
- recall the runtimes that are supported by Cloud Functions and list the different types of Cloud Functions that can be written to run codes with zero server management
- recall the Serverless architectures that can be implemented using Azure along with associated pros and cons of each architecture
- recognize the benefits of using REST API for modern applications and the steps that can be adapted to transition to API-driven development
- recognize the challenges a test engineer faces when testing cloud applications and the role of mock and dummy data in testing APIs
- recognize the critical metrics that need to be evaluated to identify the performances of Serverless applications
- recognize the different approaches of managing compute with Azure and compare the differences between Azure Compute services and Azure Serverless architecture from the perspective of infrastructure management
- recognize the features afforded by open source frameworks and the design characteristics of Serverless frameworks that need to be considered to develop Serverless applications
- recognize the goals of the REST architecture style and the design rules applied to establish the distinct characteristics of the REST architectural style
- recognize the key capabilities of Azure Test plan and the critical features of Planned and Exploratory testing
- recognize the load balancing and autoscaling capabilities afforded by Google Cloud Platform
- recognize the prominent Azure Serverless solutions and the Azure Serverless Solution architectures that can be used to develop applications for productive and optimized resource usages
- use Amazon S3 to trigger an AWS Lambda function when files are being uploaded in Amazon S3 buckets
- use Azure portal and Azure CLI to create Serverless workflows with Azure Logic Apps

FOLLOW US ON:

www.skilltech.pl

email: biuro@skilltech.pl

tel. +48 22 44 88 827

SkillTech

Technology hired for excellence