Summary

Table of Contents

[Summary: Introduction to Cloud Computing 3](#_Toc188046138)

[1. Overview of Cloud Computing 3](#_Toc188046139)

[2. Cloud Computing Models 4](#_Toc188046140)

[3. Components of Cloud Computing 6](#_Toc188046141)

[4. Emergent Trends and Practices 7](#_Toc188046142)

[5. Cloud Security, Monitoring, Case Studies, Jobs 8](#_Toc188046143)

[Summary: Introduction to Web Development with HTML, CSS, and JavaScript 9](#_Toc188046144)

[1. Introduction to Application Development 9](#_Toc188046145)

[2. HTML Overview 10](#_Toc188046146)

[3. CCS Overview and HTML5 Elements 10](#_Toc188046147)

[4. JavaScript Programming for Web Applications 11](#_Toc188046148)

[Summary: Getting Started with Git and GitHub 12](#_Toc188046149)

[1. Git and GitHub Fundamentals 12](#_Toc188046150)

[2. Using Git Commands and Managing GitHub Projects 12](#_Toc188046151)

[Summary: Developing Front-End Apps with React 13](#_Toc188046152)

[1. Building Rich Front-End Applications with React and ES6 13](#_Toc188046153)

[2. React Components 13](#_Toc188046154)

[3. Advanced React 14](#_Toc188046155)

[Summary: Developing Back-End Apps with Node.js and Express 14](#_Toc188046156)

[1. Introduction to Server-Side JavaScript 14](#_Toc188046157)

[2. Asynchronous I/O with Callback Programming 15](#_Toc188046158)

[3. Express Web Application Framework 15](#_Toc188046159)

[Summary: Python for Data Science, AI, and Development 16](#_Toc188046160)

[1. Python Basics 16](#_Toc188046161)

[2. Python Data Structures 17](#_Toc188046162)

[3. Python Programming Fundamentals 18](#_Toc188046163)

[4. Working with Data in Python 20](#_Toc188046164)

[Summary: Django Application Development with SQL and Databases 22](#_Toc188046165)

[1. Getting Started with SQL & Relational Databases 22](#_Toc188046166)

[2. ORM: Bridging the Gap Between the Real World and Relational Model 23](#_Toc188046167)

[3. Full-stack Django Development 24](#_Toc188046168)

[4. Consolidate and Deploy Your Django App 24](#_Toc188046169)

[Summary: Introduction to Containers w/ Docker, Kubernetes, and OpenShift 26](#_Toc188046170)

[1. Containers and Containerization 26](#_Toc188046171)

[2. Kubernetes Basics 27](#_Toc188046172)

[3. Managing Applications with Kubernetes 28](#_Toc188046173)

[4. The Kubernetes Ecosystem: OpenShift, Istio 29](#_Toc188046174)

[Summary: Application Development using Microservices and Serverless 30](#_Toc188046175)

[1. Introduction to Microservices 30](#_Toc188046176)

[2. Web API Essentials: REST APIs and GraphQL 30](#_Toc188046177)

[3. Serverless Overview 31](#_Toc188046178)

[4. Create and Deploy Microservices using Serverless 31](#_Toc188046179)

[5. OpenShift Essentials/Working with OpenShift and Istio 32](#_Toc188046180)

# Summary: Introduction to Cloud Computing

## **1. Overview of Cloud Computing**

**1.1 Introduction to cloud computing**

* Cloud computing is the delivery of on-demand computing resources over the internet on a pay-as-you-go basis.
* Resources are dynamically assigned and reassigned among multiple users and scaled up and down in response to users’ needs.
* The origins of cloud computing can be traced back to the mainframes of the 1950s, with virtualization technologies and hypervisors serving as catalysts for the emergence of modern-day cloud computing.
* Organizations must consider their business needs, investment viability, and risk capacity to create a cloud adoption strategy that delivers desired benefits without causing business disruptions or security, compliance, or performance issues.
* Cloud adoption is growing faster than predicted. Driving this technological wave are cloud service providers with a host of services ranging from infrastructure to platform and software services. Some major cloud providers of our time include AWS, Alibaba Cloud, Google, IBM, and Microsoft Azure.

**1.2 Business case for cloud computing**

* The adoption of cloud technologies enables enterprises, big and small, to be agile, innovative, and competitive and to create differentiated customer experiences. The questions organizations are asking are not whether they should move to the cloud but rather what strategy they should adopt to move to the cloud.
* Some case studies that demonstrate the impact businesses have created by adopting the cloud:
  + American Airlines adopting cloud technologies to deliver customer value rapidly across its enterprise.
  + UBank leveraging cloud platform services to give more control to their developers, thereby removing barriers to innovation.
  + Bitly leveraging the scalability offered by cloud infrastructure for low-latency delivery to its geographically dispersed enterprise customers.
  + ActivTrades leveraging the infrastructure, storage, network, and security offerings on the cloud to accelerate the execution and delivery of new functions in their online trading systems to their customers.

**1.3 Emerging technologies accelerated by cloud**

* Emerging technologies powered by the cloud are disrupting existing business models and creating unprecedented opportunities for businesses to grow, innovate, and create value for their customers.
* Some case studies that demonstrate how the use of emerging technologies on the cloud is creating value for millions around the world:
  + The use of the Internet of Things on the cloud to combat poaching of endangered rhinos in South Africa.
  + Artificial intelligence on the cloud is being leveraged to deliver unique digital experiences to millions of fans around the world by the United States Tennis Association.
  + Blockchain on the cloud helps farmers reduce waste by building traceability and transparency in the food supply chain.
  + The use of data analytics for driving predictive maintenance solutions for a city’s infrastructure by KONE.

## **2. Cloud Computing Models**

**2.1 Service models**

* Cloud computing allows us to utilize technology as a service, leveraging remote resources on-demand, on a pay-as-you-go model. There are three main service models available on the cloud: Infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS).
  + With IaaS, the cloud provider manages physical resources.
  + With PaaS, the provider manages the platform infrastructure.
  + In the SaaS model, the provider hosts and manages the applications and data.
* Infrastructure as a service is a form of cloud computing that delivers fundamental computer, network, and storage resources to consumers on-demand, over the network, on a pay-as-you-go basis.
* The key components of cloud infrastructure are:
  + Physical data centers
  + Compute
  + Network
  + Storage
* Platform as a service is a cloud computing model that provides customers with a complete platform: Hardware, software, and infrastructure.
* The high level of abstraction, support services, runtime environments, rapid deployment mechanisms, and middleware capabilities distinguish PaaS clouds.
* PaaS advantages are:
  + Scalability
  + Faster time to market products and services
  + Greater agility and innovation
* Software as a service is a cloud offering that provides users access to a service provider’s cloud-based software.
* SaaS characteristics are:
  + Multitenant architecture
  + Security, compliance, and maintenance
  + Customization of applications
  + Subscription model
  + Scaling
* SaaS advantages are:
  + Direct procurement of solutions
  + Improved workforce productivity and efficiency
  + Enable distribution of software costs

**2.2 Deployment models**

* Deployment models indicate where the infrastructure resides, who owns and manages it, and how cloud resources and services are available to users. There are four main deployment models available on the cloud: Public, private, hybrid, and community.
* In the public cloud model, the service provider owns, manages, provisions, and maintains the physical infrastructure, such as data centers, servers, networking equipment, and storage, with users accessing virtualized computing, networking, and storage resources as services.
* In the private cloud model, the provider provisions the cloud infrastructure for exclusive use by a single organization. The private cloud infrastructure can be internal to the organization and run on-premises. Or it can be on a public cloud, as in the case of virtual private clouds (VPC), and be owned, managed, and operated by the cloud provider.
* In the hybrid cloud model, an organization’s on-premises private cloud and a third-party, public cloud are connected as a single, flexible infrastructure that leverages the features and benefits of both public and private clouds.
* In the community cloud model, the provider provisions the cloud infrastructure for use by a community of organizations with shared concerns. One or more organizations in the community, a third-party provider, or both are responsible for the ownership, management, and operation of this infrastructure.

## **3. Components of Cloud Computing**

**3.1 Cloud infrastructure**

* Cloud infrastructure comprises data centers, storage, networking components, and computing resources.
* Virtualization is the process of creating a software-based version of physical resources, made possible by hypervisors.
* A few different types of virtual machines can be provisioned on the cloud. These include:
  + Shared or public cloud VMs are provider-managed, multi-tenant deployments that can be provisioned on-demand with predefined sizes.
  + Transient or spot VMs that use unused capacity in a cloud data center.
  + Reserved VMs that allow you to reserve capacity and guarantee resources for future deployments.
  + Dedicated hosts that offer single-tenant isolation.
* Bare metal servers are single-tenant physical servers dedicated to a single customer. Bare metal servers fulfill the demanding needs of high-performance computing (HPC) and data-intensive applications. They are ideal for applications that have a high degree of security or compliance requirements.
* Networking capabilities in the cloud are delivered as a service rather than in the form of rack-mounted devices. Cloud resources such as VMs (or VSIs), storage, network connectivity, and load balancers are deployed into subnets within virtual private clouds (VPCs). Using private and public subnets allows users to deploy multi-tier enterprise applications securely. Load balancers distribute the traffic and allow applications to be responsive.
* Containers are executable units of software in which application code, its libraries, and its dependencies are packaged in a common way, so that it can be run anywhere, from desktops to traditional IT to the cloud. Containers are more lightweight and consume fewer resources than virtual machines, helping streamline the development and deployment of cloud native applications.

**3.2 Cloud storage and content delivery networks**

* Cloud storage is available in four main types: Direct attached, file, block, and object storage. These storage types differ in how they can be accessed, the capacity they offer, how much they cost, the types of data they are best suited to store, and their read-write speed.
* Direct attached (or local) storage is storage presented directly to a cloud-based server and is effectively either within the host server chassis or within the same rack.
* File storage is typically presented to compute nodes as a Network File System (NFS), which means that the storage is connected to compute nodes over a standard Ethernet network.
* Block storage is presented to compute nodes using high-speed fiber connections, typically provisioned in volumes, which are mounted onto a compute node.
* Object storage is accessed via an API and doesn’t need an underlying compute node.
* Object storage offers infinite capacity as you can keep adding files to it and just pay for what you use. Compared to the other storage types, object storage is the slowest in terms of read and write speeds.
* A content delivery network (CDN) is a distributed server network that accelerates internet content delivery by delivering temporarily stored or cached copies of website or media content to users based on their geographic location.

## **4. Emergent Trends and Practices**

**4.1 Hybrid multi-cloud, microservices, and serverless**

* Hybrid multi-cloud is a cloud adoption strategy that allows public clouds, private clouds, and on-premises IT to interoperate seamlessly while leveraging the best cloud-based services from different public cloud providers.
* Microservices architecture is an approach in which an application is built as a collection of loosely coupled and independently deployable components or services, leading to efficient development, maintenance, and upgradation cycles.
* Serverless computing is an approach to computing that offloads responsibility for common infrastructure management tasks for application runtimes to cloud providers, allowing developers to focus their time and effort on development and testing and not have to worry about provisioning, maintaining, and scaling compute resources.

**4.2 Cloud native applications, DevOps, and application modernization**

* Cloud native applications are applications that are built or refactored to work in the cloud environment. These applications, developed using DevOps methodologies, consist of microservices packaged in containers that can run in any environment, making it possible to create and update features in quick iterative cycles.
* DevOps is a collaborative approach that enables development and operations teams to continuously deliver software in quick iterative cycles while reducing overhead, duplication, and rework. DevOps’ tools, practices, and processes help tackle the complexities and challenges posed by the cloud, allowing solutions to be delivered and updated quickly and reliably.
* Application modernization helps organizations accelerate digital transformation, take advantage of new technologies and services, and become more responsive to changing market dynamics. Cloud computing is one of the key enablers of application modernization.

## **5. Cloud Security, Monitoring, Case Studies, Jobs**

**5.1 Cloud security and monitoring**

* Cloud security refers to the policies, technological procedures, services, and solutions designed to secure enterprise applications and data on the cloud against insider threats, data breaches, compliance issues, and organized security threats.
* Cloud security is a shared responsibility between the cloud provider and the user organization.
* Security architecture and methods for achieving continuous security need to be embedded throughout the life cycle of an application to ensure that the application runs on a safe platform, the code is free from vulnerabilities, and the operational risks are understood.
* Identity and access management, also known as access control, helps authenticate and authorize users and provides user-specific access to cloud resources, services, and applications.
* As part of their identity and access management services, most cloud providers offer users the ability to define access groups and create access policies that define permissions for users on account resources.
* Cloud encryption, often called the last line of defense, encrypts data and provides robust data access control, key management, and certificate management.
* Data needs encryption in three states:
  + **Encryption at rest:** Protecting data while it is stored.
  + **Encryption in transit:** Protecting data while it is transmitted from one location to another.
  + **Encryption in use:** Protecting data when used in memory.
* There needs to be active monitoring of all connected systems and cloud-based services to maintain visibility of all data exchanges between public, private, and hybrid cloud environments. This ensures that the cloud provides a trusted platform that can securely integrate with your enterprise data centers.

**5.2 Case studies and jobs**

* Businesses all over the world are realizing tangible benefits from the use of cloud technologies and services, including:
  + The Weather Company migrating to the cloud to reliably deliver critical weather data at high speed, especially during major weather events such as hurricanes and tornadoes.
  + American Airlines using the cloud platform and technologies to deliver digital self-service tools and customer value more rapidly across its enterprise.
  + Cementos Pacasmayo achieving operational excellence and insight to help drive strategic transformation and reach new markets using cloud services.
  + Welch choosing cloud storage to drive business value from hybrid cloud.
  + LiquidPower using cloud-based SAP applications to fuel business growth.
* The market size of the cloud services industry is nearly three times the growth of overall IT services, increasing the need for qualified cloud computing professionals. Some common job roles available in this domain include those of cloud software engineers, cloud integration specialists, cloud data engineers, cloud security engineers, cloud DevOps engineers, and cloud solution architects.

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9HbG9zc2FyeS9JbnRyb2R1Y3Rpb25fdG9fQ2xvdWRfQ29tcHV0aW5nLm1kIiwidG9vbF90eXBlIjoiaW5zdHJ1Y3Rpb25hbC1sYWIiLCJhdGxhc19maWxlX2lkIjoxMjkxMSwiYWRtaW4iOmZhbHNlLCJpYXQiOjE3Mjk4OTk5NTV9.MrS4sxCLJfp--46kfJoxrUEfGZWAVp4uj9NJ2WzMW6Y>

# Summary: Introduction to Web Development with HTML, CSS, and JavaScript

## **1. Introduction to Application Development**

* Front-end developers work on the parts of the website or app that the user sees and interacts with.
* Back-end developers work on the logic and functionality that keeps the website or app running and responding to users’ inputs.
* Full-stack developers have both sets of skills.
* Front-end developers and back-end developers work closely together.
* Frameworks and libraries extend the functionality of coding languages such as JavaScript and Python.
* Common languages for front-end development include HTML, CSS, and JavaScript.
* Common languages and frameworks for back-end development include Python, Django, and Flask.
* Version control systems keep track of changes and resolve conflicts between them.
* Continuous Integration with Continuous Delivery/Deployment (CI/CD) is the best practice developers use to deliver frequent changes reliably.

## **2. HTML Overview**

* HTML provides the basic structure and content for a website using tags.
* Tags represent the elements of an HTML page.
* The HTML DOM tree describes how a website is structured.
* HTML uses APIs to enhance the user experience, providing features for advanced animation, audio, and video.
* Scripting provides a more interactive user experience when browsing websites. It is recommended to not rely on scripting, as it can be disabled.
* HTML5 sandboxes help manage iframe mashups.
* HTML5 Browser Support tables describe which browsers support which HTML5 features.
* JavaScript is used to check if an element is supported by a browser.
* CSS provides consistent style and design throughout the website.
* There are two types of CSS layouts used to design websites: Fluid and fixed.

## **3. CCS Overview and HTML5 Elements**

* CSS creates a uniform look throughout each element of each website page.
* CSS is usually coded in external style sheets and creates base styles for a website.
* CSS frameworks assist in implementing UI elements and creating dynamic web pages.
* CSS has two types of frameworks:
  + Utility-first frameworks, which provide utility classes to help in building one’s own styles and layouts.
  + Component frameworks, which provide a wide selection of pre-styled components and templates that can be implemented on a website.
* Plain (Vanilla) CSS lets developers write the styles and layouts of a website.
* HTML5 elements provide structure and function to websites.
* HTML5 uses the <input> tag to allow users to input information. It has many different types, including:
  + Color
  + Date
  + Datetime-local
  + Email
  + Number
  + Range
  + Search
  + URL

## **4. JavaScript Programming for Web Applications**

* JavaScript is a scripting language that enables developers to add dynamic content to web pages.
* JavaScript variables are declared using the var keyword and take their type from the value assigned.
* Program execution is controlled by statements like If…Then…Else, Switch, For loops, and While loops.
* JavaScript uses blocks of code called functions that can be called from anywhere in the script.
* New methods and properties can be added to an object by modifying the prototype for that object.
* Prototypes allow you to define properties and methods for all instances of a specific object.
* Client-side scripts are programs that accompany HTML documents and are used by developers to incorporate more interactive elements.
* The script tag can incorporate a script within an HTML document or call a script from an external file.
* The Document Object Model (DOM) is the programming interface between HTML or XHTML and JavaScript.
* Developers can access HTML DOM elements from JavaScript scripts using the correct DOM notation.
* APIs are often used to access HTML DOM elements in web pages.

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9HbG9zc2FyeS9JbnRyb2R1Y3Rpb25fdG9fV2ViX0RldmVsb3BtZW50X3dpdGhfSFRNTCUyQ0NTUyUyQ0phdmEubWQiLCJ0b29sX3R5cGUiOiJpbnN0cnVjdGlvbmFsLWxhYiIsImF0bGFzX2ZpbGVfaWQiOjEyOTEzLCJhZG1pbiI6ZmFsc2UsImlhdCI6MTcyOTg5OTk2MX0.6OYeB8xO_9WIFNCxypv-6__Bo_JQ8IIeL-K74c4jKPU>

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-CS0203EN-Coursera/labs/Cheat_sheet/C2-Introduction_to_Web_Development_with_HTML%2C_CS.md.html>

# Summary: Getting Started with Git and GitHub

## **1. Git and GitHub Fundamentals**

**1.1 Getting started with Git and GitHub**

* A distributed version control system (DVCS) keeps track of changes to code, regardless of where it is stored. This allows multiple users to work on the same codebase or repository, mirroring the codebase on their own computers if needed, while the distributed version control software helps manage synchronization amongst the various codebase mirrors.
* Repositories are storage structures that:
  + Store the code
  + Track issues and changes
  + Enable you to collaborate with others
* GitHub is one of the most popular web-hosted services for Git repositories. GitLab, Bitbucket, and Beanstalk are examples of hosted version control systems.

## **2. Using Git Commands and Managing GitHub Projects**

**2.1 GitHub workflows with branches and Git commands**

* Branches are used to isolate changes to code. When the changes are complete, they can be merged back into the main branch.
* Repositories can be cloned to make it possible to work locally, then sync changes back to the original.
* Repositories can be forked to be used as a base for a new project, or so that the developer can work independently.
* A pull request (PR) can be submitted to have your changes reviewed and merged.
* Large projects include people working in different roles:
  + Developer: Creates code
  + Integrator: Manages changes made by developers
  + Repository Administrator: Configures and maintains access to the repository

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9HbG9zc2FyeS9HZXR0aW5nX1N0YXJ0ZWRfd2l0aF9HaXRfYW5kX0dpdEh1Yi5tZCIsInRvb2xfdHlwZSI6Imluc3RydWN0aW9uYWwtbGFiIiwiYXRsYXNfZmlsZV9pZCI6MTI5MjUsImFkbWluIjpmYWxzZSwiaWF0IjoxNzI5OTAwMDcwfQ.km3vbegMCI1RL7OuVsKZ9tp3kS6SeE9yukqxUnobEU4>

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9DaGVhdF9zaGVldC9DMy1HZXR0aW5nX1N0YXJ0ZWRfd2l0aF9HaXRfYW5kX0dpdEh1Yi5tZCIsInRvb2xfdHlwZSI6Imluc3RydWN0aW9uYWwtbGFiIiwiYXRsYXNfZmlsZV9pZCI6MTI5MzEsImFkbWluIjpmYWxzZSwiaWF0IjoxNzI5OTAwMTg5fQ.goxShwLszW4MjKL5ni-8k-BFqoFPaHlxusUQ20nK0FE>

# Summary: Developing Front-End Apps with React

## **1. Building Rich Front-End Applications with React and ES6**

* React is an efficient, flexible JavaScript library for building user interfaces.
* New features introduced in JavaScript as a part of ES6 are let, const, arrow functions, promise, and class.
* The main benefits of using JSX are that you can leverage the full power of JavaScript in HTML and avoid learning or using a templating language. It allows React to show useful errors and warning messages.
* The four types of React components are Functional, Class, Pure, and High-order components.
* Functional components are most useful when the lifecycle of the component does not have to be managed.
* Class components are more versatile.

## **2. React Components**

* State is a plain JavaScript object used by React to represent information about the component’s current situation.
* Props is a shortened form for properties, and they are used to pass data between React components in a unidirectional flow from parent to child.
* You can pass data between components from parent to child using properties, from child to parent using callbacks, and between siblings.
* Components are created or mounted on the DOM; they grow by updating and then die or are unmounted on DOM. This is referred to as a component lifecycle.
* React components can be tested using Mocha, Chai, and Sinon, but preferred approaches are using Jest and React Testing Library.

## **3. Advanced React**

* Hooks provide a way to use functionalities such as context or state without classes.
* Inputs in React can be one of two types: controlled or uncontrolled.
* Redux is a state management library often used with React to handle the state of your application.
* The Redux elements involved in updating the component properties are action, store, and reducer.
* You can interact with asynchronous data in your React Redux app using middleware.
* The data flow in the React-Redux application is unidirectional.

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9HbG9zc2FyeS9EZXZlbG9waW5nX0Zyb250LUVuZF9BcHBzX3dpdGhfUmVhY3QubWQiLCJ0b29sX3R5cGUiOiJpbnN0cnVjdGlvbmFsLWxhYiIsImF0bGFzX2ZpbGVfaWQiOjEyOTE5LCJhZG1pbiI6ZmFsc2UsImlhdCI6MTcyOTkwMDAxNn0.t28o-o2UOZTRFSgpCjNxBAmkADQbduk-r4Y_vlQWUZc>

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9DaGVhdF9zaGVldC9DNC1EZXZlbG9waW5nX0Zyb250LUVuZF9BcHBzX3dpdGhfUmVhY3QubWQiLCJ0b29sX3R5cGUiOiJpbnN0cnVjdGlvbmFsLWxhYiIsImF0bGFzX2ZpbGVfaWQiOjEyOTM1LCJhZG1pbiI6ZmFsc2UsImlhdCI6MTcyOTkwMDI0MX0.9g7tBI_rAc9CRKMjhfGPLqOMB05G7gMvhi6hjWp7JIs>

# Summary: Developing Back-End Apps with Node.js and Express

## **1. Introduction to Server-Side JavaScript**

* Back-end technologies include various types of servers and supporting infrastructures such as programming languages, frameworks, and other hardware.
* Node.js is the server-side component of JavaScript. Using Node.js can improve application performance, and express.js is a framework that helps you build Node.js applications.
* The “require” statement can be called from anywhere in the app code, is bound dynamically, and is synchronous, whereas the “import” statement can only be called at the beginning of a file, is bound statically, and is asynchronous.
* Client-side JavaScript is used to process front-end user interface elements, and server-side JavaScript is used to enable access to different kinds of servers and web applications.
* With server-side JavaScript, Node.js applications process and route web service requests from the client.
* To make a function or a value available to Node.js applications that import your module, add a property to the implicit exports object.
* Core modules include bare minimum functionality, local modules are those that you create for your application, and the Node.js community creates third-party modules.
* A local install means only the application within the directory of the installed can access the package, whereas a global install means that any application on the machine can access the package.

## **2. Asynchronous I/O with Callback Programming**

* Asynchronous network operations can be handled using callback functions to prevent blocking JavaScript code.
* A callback function must invoke another callback function to pass a message from the Node.js module back to the main application after the Node.js module receives a response message.
* Nested callbacks can be difficult to read and debug. Inversion of control causes trust issues when dealing with third-party code.
* Promise objects are most useful for operations that are time-consuming and can block resources.
* JSON.parse() and JSON.stringify() are two methods used to parse JSON objects.

## **3. Express Web Application Framework**

* Developers rely on third-party packages to extend Node.js.
* You can use the npm application to manage Node.js packages in your Node.js framework installation.
* The model-view-controller MVC architecture style divides a back-end application into three parts: the model, the view, and the controller.
* REST API frameworks use HTTP methods to communicate with each other.
* Express abstracts low-level details.
* Routing can be handled at the application level or router level.
* Five types of middleware are as follows: application level, router level, error handling, built-in middleware, and third party.
* Template rendering is the ability of the server to fill in dynamic content.
* The npm jsonwebtoken package should be required in an Express application to authenticate a user.

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9HbG9zc2FyeS9EZXZlbG9waW5nX0JhY2stRW5kX0FwcHNfd2l0aF9Ob2RlLmpzX2FuZF9FeHByZXNzLm1kIiwidG9vbF90eXBlIjoiaW5zdHJ1Y3Rpb25hbC1sYWIiLCJhdGxhc19maWxlX2lkIjoxMjkyMSwiYWRtaW4iOmZhbHNlLCJpYXQiOjE3Mjk5MDAwNDd9.yKNcEMwPOmRTjZEy2xa4ZYKnNXn7D2LqjBQ-MeipvVE>

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9DaGVhdF9zaGVldC9DNS1EZXZlbG9waW5nX0JhY2stRW5kX0FwcHNfd2l0aF9Ob2RlLmpzX2FuZF9FeHAubWQiLCJ0b29sX3R5cGUiOiJpbnN0cnVjdGlvbmFsLWxhYiIsImF0bGFzX2ZpbGVfaWQiOjEyOTMzLCJhZG1pbiI6ZmFsc2UsImlhdCI6MTcyOTkwMDIyMX0.qN54pWRKyHXqUd8V4mYwwJeQ5j_ZlDrG8VSHd6pMo5k>

# Summary: Python for Data Science, AI, and Development

## **1. Python Basics**

**1.1 Types**

* A type is how Python represents different types of data. For instance, the three data types are int, float, and string.
* The data type int stands for an integer, float stands for float, essentially a real number, and the data type string is a sequence of characters.
* Typecasting is a process that can be used to change the type of expression in Python. The type int can be changed to float. A string containing an integer value can also be converted into int.
* Boolean is another important type in Python. A Boolean can take on two values: True or False.

**1.2 Expressions and variables**

* Expressions are operations Python performs. The numbers are referred to as operands, whereas the math symbols are referred to as operators.
* The addition operation can be performed using an addition symbol (+). To perform the operation of subtraction, use the subtraction sign ( ̶ ). For multiplication operations, use the asterisk symbol (\*), and for division operations, use the forward slash (/).
* Python follows mathematical conventions when performing mathematical expressions.
* Variables are used to store values. To assign a new value to any variable, use the assignment operator, i.e., the equal sign.

**1.3 String operations**

* In Python, a string is a sequence of characters. A string is contained within two quotes. A string can be spaces or digits. A string can also be special characters.
* A string can be considered as an ordered sequence. Each element in the sequence can be accessed using an index represented by the array of numbers.
* The len command is used to obtain the length of the string.
* Backslashes represent the beginning of escape sequences.
  + Backslash n (i.e. \n) represents a new line. The output is given by a new line after the backslash "n" is encountered.
  + Backslash t (i.e. \t) represents a tab. The output is given by a tab where the \t is.
  + To place a backslash in your string, use a double backslash.
* The string method "Upper" converts lowercase characters to uppercase characters.

## **2. Python Data Structures**

**2.1 List and tuples**

* Lists and tuples are called compound data types and are one of the key types of data structures in Python.
* Tuples are an ordered sequence and are expressed as comma-separated elements within parentheses. Each element of a tuple can be accessed via an index.
* Tuples are immutable, which means they cannot be changed.
* A tuple can contain other tuples as well as other complex data types, which is called nesting.
* Lists are also an ordered sequence and are represented with square brackets.
* The one key difference between tuples and lists is that the lists are mutable.

**2.2 Dictionaries**

* Dictionaries are a type of collection in Python. The dictionary has keys and values.
  + The key is analogous to the index. They are like addresses, but they do not have to be integers. They are usually characters.
  + The values are similar to the elements in a list and contain information.
* To create a dictionary, use curly brackets. The keys are the first elements. They must be immutable and unique. Each key is followed by a value separated by a colon. The values can be immutable, mutable, and duplicates. Each key and value pair is separated by a comma.

**2.3 Sets**

* A set is a type of collection that is unordered and consists of unique elements.
* To define a set, place the different elements of a set in a curly bracket separated by commas. Remove the duplicate elements.
* A list can be converted to a set by using the function set, which is called type casting. Simply use the list as the input to the function set. The result will be a list converted to a set.
* A Venn diagram is a tool that uses shapes to represent sets.
* To add an item to a set, use the add method. Put the set name followed by a dot, then the add method.
* To remove an item from a set, use the remove method. Put the set name followed by a dot, then the remove method.
* The intersection of two sets is a new set containing elements that are in both of those sets. The intersection is defined in terms of "and." In Python, we use an ampersand (&) to find the intersection of the two sets.
* The union of two sets is the new set of elements that contain all the items in both sets.

## **3. Python Programming Fundamentals**

**3.1 Conditioning and branching**

* Comparison operations compare some values or operands and produce a Boolean (True or False) based on some condition.
* The equality operator denoted with two equal signs can be used to determine if two values are equal.
* The inequality test uses an exclamation mark preceding the equal sign. If two operands are not equal, then the condition becomes true.
* Branching allows us to run different statements for different inputs. For instance, it is helpful to think of an if statement as a locked room. If this statement is true, you can enter the room and your program can run some predefined task. If the statement is false, your program will skip the task.
* The elif statement, short for else if, allows us to check additional conditions if the preceding condition is false. If the condition is true, the alternate expressions will be run.
* Logic operations take Boolean values and produce different Boolean values.
  + The first operation is the not operator. If the input is true, the result is false. Similarly, if the input is false, the result is true.
  + The OR operator takes in the two values and produces a new Boolean value. It only produces a false if all the Boolean values are false.
  + The AND operator takes in the two values and produces a new Boolean value. It only produces a true if all the Boolean values are true.

**3.2 Loops**

* The range function outputs and orders sequence as a list. If the input is a positive integer, the output is a sequence. The sequence contains the same number of elements as the input but starts at zero.
* If the range function has two inputs where the first input is smaller than the second input, the output is a sequence that starts at the first input. Then, the sequence iterates up to but not including the second number.
* Loops perform a task over and over. The two such loops are for loop and while loop.

**3.3 Functions**

* Functions take some input and then produce some output or change.
* The function len takes in an input of type sequences, such as a string or list, or a type collection, such as a dictionary or set, and returns the length of that sequence or collection.
* The function sum takes in an iterable like a tuple or list and returns the total of all the elements.
* The function sorted returns a new sorted list or tuple.
* The function mult multiplies two numbers. It returns a new integer if the two numbers are integers. It returns a float if we pass an integer and float. If we pass in the integer two and a string, then the string will be repeated two times.
* Global variables are those variables that are defined outside of any function having a global scope, meaning they can be accessed anywhere after they are defined.

**3.4 Exception handling**

* The try…except statement will first attempt to execute the code in the “try” block, but if an error occurs, it will kick out and begin searching for the exception that matches the error. Once it finds the correct exception to handle the error, it will then execute that line of code.
* Adding an else statement will provide us a notification to the console that “The file was written successfully.”
* By adding a finally statement, it will tell the program to close the file no matter the end result and print “File is now closed” to our console.

**3.5 Objects and classes**

* Python has many different kinds of data types: integers, floats, strings, lists, dictionaries, and Booleans.
* Every object has the following: a type, internal representation, and a set of functions called methods to interact with the data. An object is an instance of a particular type.
* Building the circle class in Python:
  + First, define the class, and then initialize each instance of the class with data attributes, radius, and color using the class constructor.
  + The function init is a constructor, which is a special function that tells Python you are making a new class.
  + There are other special functions in Python to make more complex classes. The radius and color parameters are used to initialize the radius and color data attributes of the class instance.
  + The self-parameter refers to the newly created instance of the class. The parameters, radius, and color can be used in the constructors' body to access the values passed to the class constructor when the class is constructed.

## **4. Working with Data in Python**

**4.1 Reading and writing files with open**

* Python's built-in open function is used to create a file object and obtain the data from a "txt" file.
  + While using the open function, the first argument is the file path. This is made up of the file name and the file directory.
  + The second parameter is the mode. Common values used include 'r' for reading, 'w' for writing, and 'a' for appending.
  + Finally, we have the file object.
* It is recommended to use a "with" statement to open a file because it automatically closes the file. The code will run everything in the indent block, and then closes the file.
* Python's open function can be used to get a file object to create a text file. We can apply method write to write data to that file.
  + While using the open function, the first argument is the file path. This is made up of the file name. If you have that file in your directory, it will be overwritten.
  + The second parameter is the mode. Set the mode parameter to "w" for writing.
  + Finally, we have the file object.

**4.2 Pandas**

* Pandas is a popular library for data analysis.
* This library can be imported using the "import" command followed by the library name, which will give access to a large number of pre-built classes and functions.
* A data frame is comprised of rows and columns. A data frame can be created out of a dictionary, where the keys correspond to the column labels, and the values or lists correspond to the rows.

**4.3 Numpy in Python**

* Numpy is a library for scientific computing and has many useful functions. It serves as the basis for Pandas and has many other advantages like speed and memory.
* A Python list is a container that allows you to store and access data, where each element is associated with an index.
* A Numpy array or ND array is similar to a list, usually fixed in size, and has elements of the same type.
* The attribute ndim represents the number of array dimensions or the rank of the array.
* The attribute shape is a tuple of integers indicating the size of the array in each dimension.
* **Vector addition:** This can be performed over two given vectors to get a new vector. The first component of this new vector is the addition of the first component of the given two vectors. Similarly, the second component of the new vector is the sum of the second components of the given two vectors. The new vector is a linear combination of the given two vectors.
* **Vector multiplication with a scalar:** To multiply a vector with a scalar, multiply each component of a given vector by the given scalar.
* **Hadamard product** is another widely used operation in data science. The Hadamard product of u and v is a new vector z. The first component of z is the product of the first element of u and v. Similarly, the second component is the product of the second element of u and v. The resultant vector consists of the entrywise product of u and v.

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9DaGVhdF9zaGVldC9DNi1QeXRob25fZm9yX0RhdGFfU2NpZW5jZSUyQ19BSV9hbmRfRGV2ZWxvcG1lbnQubWQiLCJ0b29sX3R5cGUiOiJpbnN0cnVjdGlvbmFsLWxhYiIsImF0bGFzX2ZpbGVfaWQiOjEyOTE1LCJhZG1pbiI6ZmFsc2UsImlhdCI6MTcyOTkwMDA1Nn0.ld_mbS2EUlN1SqlGbnUMemzEcbHGp-XZc9VE8OZrQX8>

# Summary: Django Application Development with SQL and Databases

## **1. Getting Started with SQL & Relational Databases**

**1.1 Introduction to databases**

* Structured query language, or SQL, was designed to manage data in relational databases and is useful for handling structured data.
* Data is a collection of facts in words, numbers, and pictures.
* A database is a repository of data that provides functionality for adding, modifying, and querying data.
* Relational databases store tabular data as collections of related items, with columns containing item properties.
* The basic SQL statements are CREATE TABLE, INSERT, SELECT, UPDATE, and DELETE.
* Non-relational databases provide a flexible and scalable approach to storing and retrieving data.
* Relational databases are ideal for the optimized storage, retrieval, and processing of large volumes of data.
* Relational Database Management System (RDBMS) is a mature and well-documented technology, providing flexibility, reduced redundancy, ease of backup and disaster recovery, and ACID compliance.
* An entity-relationship model is a tool for designing relational databases. Entities become tables, and attributes are translated into columns.

**1.2 Basic SQL statements**

* SQL is used for querying and managing data.
* SQL is useful for handling structured data or data incorporating relations among entities and variables.
* The SQL SELECT statement retrieves data from a relational database table.
* The SELECT statement is a query, and the output we get from executing this query is a result set or a result table.
* In its simplest form, the syntax for a SELECT statement is: SELECT \* from TableName
* The SQL INSERT statement inserts data into a relational database table by adding rows.
* The syntax of the INSERT statement is: INSERT INTO TableNameColumnName VALUES (values)
* For the INSERT statement, the values provided in the Values clause must equal the number of column names specified in the Column Name list. This ensures that each column has a value.
* The SQL UPDATE statement is used to read and modify data.
* The syntax of the UPDATE statement is as follows: UPDATE [TableName] SET [[ColumnName]=[Value]] <WHERE [Condition]>
* The SQL DELETE statement is used to remove data from a table.
* The syntax of the DELETE statement is: DELETE FROM [TableName] <Where [Condition]>
* The WHERE clause specifies the rows in a table that will be acted on by an SQL statement such as SELECT, DELETE, or UPDATE.

## **2. ORM: Bridging the Gap Between the Real World and Relational Model**

* The object-oriented programming (OOP) and SQL paradigms model data differently.
* Object-relational mapping, or ORM, bridges the gap between OOP and SQL.
* ORM libraries or tools can map and transfer data stored in a relational database as rows into objects or objects into rows.
* ORM allows developers to use OOP to query and manipulate data because it transfers objects into rows and rows into objects.
* Django ORM is a Python ORM component that belongs to the Django web application framework.
* Django ORM can help speed up database development because you define maps to a database table for each Django model.
* Each Django field maps to a column type.
* Django automatically creates tables once models and fields are defined.
* Django APIs can perform Create, Read, Update, and Delete (CRUD) operations on database objects.
* In a Django model, you create an object and call the model’s save method to insert it into the database as a record.
* To read objects using Django Model API, you need to construct a QuerySet using a Manager on your model class.
* There are several ways to update database records in Django by updating objects.
* To delete records in a database, you call Django ORM’s Delete method on a model object or query set.

## **3. Full-stack Django Development**

* The Model-View-Controller design pattern divides application logic into three components:
  + Model accesses and manipulates data
  + View presents data in various forms
  + Controller coordinates between Model and View
* The Django Model-View-Template pattern is like MVC, except there is no Controller, and the Django server performs the controller function.
* In Django, View is a Python function that takes a web request and applies the necessary logic to generate a web response.
* Django uses a template containing static HTML elements and special Python code to generate dynamic web pages.
* When you create a Django project, Django creates some core files.
  + manage.py is a command-line interface used to interact with the Django project
  + settings.py contains the settings and configurations for your Django project
  + urls.py contains the URL and routing definitions of your Django app
* You start building a Django admin site by creating an admin user.
* You can then log in as a superuser and register your models to the admin site so you can manage them.
* You can customize the admin form and add search and filters.
* A Django View takes a web request such as HTTP GET, POST, DELETE, or UPDATE and returns a web response. The web response can be a string, JSON/XML file, HTML page, or an error status indicating client or server-side errors.
* You create templates in Django to specify how your data will be presented. A Django template combines static HTML elements with Django template tags and variables to describe how the dynamic parts will be inserted. These work together to generate an HTML page rendered in a user’s web browser.

## **4. Consolidate and Deploy Your Django App**

* Both function-based and class-based views are Python functions.
* When you build a class-based view, you define a class subclassing the Django View base class. Then, you access some standard methods such as Get or Post. Next, you implement your logic to handle HTTP requests.
* To speed up development and solve common tasks, Django provides some built-in view classes called generic-based views for developers to reuse.
* Authentication is validating users’ identities using credentials such as username and password.
* After users are authenticated, authorization will check the users’ access permissions for resources such as databases.
* In Django, a user model is created to handle authentication and to work with other models, such as groups and permissions, to handle authorization.
* Developers can extend the user model to define application-specific users, such as instructors or learners inherited from the user model.
* Bootstrap, a free web front-end framework, facilitates web app development.
* Bootstrap provides many HTML and CSS templates to simplify Django template development.
* If you want to use Bootstrap CSS style classes without downloading Bootstrap, add a link to the latest Bootstrap version into the head element of your HTML template.
* To add static files to your apps, you first create folders for different static files, such as HTML templates, images, or CSS files.
* Under each folder to hold static files, you create a subfolder using the same app name. This creates namespacing to uniquely refer to static files that use the same name across multiple apps in a Django project.
* Django provides a set of STACFILES\_FINDERS for locating the static files in an app. It also provides a staticfiles app to collect all static files in a single directory when an app is deployed.
* To deploy reliable, scalable, and maintainable Django apps, you need to deploy them on web servers.
* Since most web servers are not written in Python, Django apps need extra interfaces to talk to web servers.
* The Web Server Gateway Interface, or WSGI, is the main Python standard for communicating between web servers and applications.
* The Asynchronous Server Gateway Interface is another web server interface the Django app supports.
* Infrastructure as a service and platform as a service offering allows you to focus on your app development and deploy apps without worrying about the underlying infrastructure and platform.

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9DaGVhdF9zaGVldC9DOC1EamFuZ29fQXBwbGljYXRpb25fRGV2ZWxvcG1lbnRfd2l0aF9TUUxfYW5kX0QubWQiLCJ0b29sX3R5cGUiOiJpbnN0cnVjdGlvbmFsLWxhYiIsImF0bGFzX2ZpbGVfaWQiOjEyOTI3LCJhZG1pbiI6ZmFsc2UsImlhdCI6MTcyOTkwMDE0MX0.gSGG2UReVjQixk_t2b05msISJohOVqvuj0Vz-h3I6wE>

# Summary: Introduction to Containers w/ Docker, Kubernetes, and OpenShift

## **1. Containers and Containerization**

**1.1 Understanding the benefits of containers**

* A container is a unit of software that encapsulates everything needed to build, ship, and run applications.
* Containers lower deployment time and costs, improve utilization, automate processes, and support next-gen applications (microservices). Major container vendors include Docker, Podman, LXC, and Vagrant.
* Containers are small, fast, lightweight, and portable. Unlike virtual machines, they leverage the features and resources of the host operating system.
* Docker is an open platform used for developing, shipping, and running applications as containers.
* Docker containers are not a good fit for applications based on monolithic architecture or applications that require high performance or security.
* Docker architecture consists of the Docker client, the Docker host, and the container registry.
* The Docker host contains objects such as Dockerfiles, images, containers, networks, storage volumes, and other objects, such as plugins and add-ons.
* Docker uses networks to isolate container communications.
* Docker uses volumes and binds mounts to persist data even after a container stops running.
* Plugins, such as storage plugins, provide the ability to connect to external storage platforms.
* Some of the common Docker CLI commands include:
  + The **build** command is used for creating container images.
  + The **tag** command is used to name images.
  + The **images** command lists all images, their repositories and tags, and their sizes.
  + The **run** command is used to run a container.
  + The **push** and **pull** commands are used for storing images in a remote location and then retrieving those images.

## **2. Kubernetes Basics**

**2.1 Understanding Kubernetes architecture**

* Container orchestration automates the container lifecycle, resulting in faster deployments, reduced errors, higher availability, and more robust security.
* Container orchestration tools have a wide variety of features and perform the following functions:
  + Aids in the provisioning and deployment of containers to make this a more automated, unified, and smooth process
  + Ensures that containers are redundant and available so that applications experience minimal downtime
  + Scales containers up and down to meet the demand and load-balance requests across instances so that no one instance is overwhelmed
  + Handles the scheduling of containers to the underlying infrastructure
  + Can perform health checks to ensure that applications are running and take necessary actions when checks fail
* Kubernetes is a highly portable, horizontally scalable, open-source container orchestration system with automated deployment and simplified management capabilities.
* Kubernetes architecture consists of a control plane and one or more worker planes.
  + A control plane includes controllers, an API server, a scheduler, and an **etcd**.
  + A worker plane includes nodes, a kubelet, container runtime, and kube-proxy.
* Kubernetes objects include Namespaces, Pods, ReplicaSets, Deployments, and Services.
  + Namespaces help in isolating groups of resources within a single cluster.
  + Pods represent a process or an instance of an app running in the cluster.
  + ReplicaSets create and manage horizontally scaled running Pods.
  + Deployments provide updates for Pods and ReplicaSets.
  + A service in Kubernetes is a REST object that provides policies for accessing the pods and cluster.
* Kubernetes capabilities include automated rollouts and rollbacks, storage orchestration, horizontal scaling, automated bin packing, secret and configuration management, Ipv4/Ipv6 dual-stack support, batch execution, self-healing, service discovery, load balancing, and extensible design.
* Services in Kubernetes are REST objects that provide policies for accessing the Pods and clusters.
  + ClusterIP provides Inter-service communication within the cluster.
  + NodePort Service, an extension of the ClusterIP service, creates and routes the incoming requests automatically to the ClusterIP Service.
  + The external load balancer (ELB) creates NodePort and ClusterIP Services automatically.
  + External Name service represents external storage as well as enables Pods from different namespaces to talk to each other.
* Ingress is an API object that provides routing rules to manage external users' access to multiple services in a Kubernetes cluster, whereas using a DaemonSet ensures that there is at least one copy of the Pod on all nodes; a StatefulSet manages stateful applications, manages Pod deployment and scaling, maintains a sticky identity for each Pod request and provides persistent storage volumes for your workloads and lastly a Job creates pods and tracks the Pod completion process; Jobs are retried until completed.

## **3. Managing Applications with Kubernetes**

* A ReplicaSet enables scaling by creating or deleting pods.
* A ReplicaSet always tries to match the actual state to the desired state.
* Autoscaling enables scaling as needed at the cluster or node level and the Pod level.
* Autoscaler types include horizontal Pod (HPA), vertical pod (VPA), and cluster (CA).
* Rolling updates roll out app changes in a controlled and automated way.
* Rolling updates and rollback can be performed using all-at-once and one-at-a-time strategies.
* ConfigMaps are used to provide variables for your application.
* Secrets are used to provide sensitive information to your application.
* Binding an external Service to your deployment automatically provides the credentials to use the Service inside the code.
* Binding manages configuration and credentials for back-end Services while protecting sensitive data.

## **4. The Kubernetes Ecosystem: OpenShift, Istio**

**4.1 The Kubernetes ecosystem**

* OpenShift® is an enterprise-ready Kubernetes container platform built for open hybrid cloud.
* OpenShift is easier to use, integrates with Jenkins, and has more services and features.
* Custom resource definitions (CRDs) extend the Kubernetes API.
* CRDs paired with custom controllers create new, declarative APIs in Kubernetes.
* Operators use CRDs and custom controllers to automate cluster tasks.
* A build is a process that transforms inputs into an object.
* An ImageStream is an abstraction for referencing container images in OpenShift.
* A service mesh provides traffic management to control traffic flow between services, security to encrypt traffic between services, and observability of service behavior to troubleshoot and optimize applications.
* Istio is a service mesh that supports four concepts: connection, security, enforcement, and observability. It is commonly used with microservices applications.
* Istio provides service communication metrics for these basic service monitoring needs: latency, traffic, errors, and saturation.

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9HbG9zc2FyeS9JbnRyb2R1Y3Rpb25fdG9fQ29udGFpbmVyc193L19Eb2NrZXIlMkNfS3ViZXJuZXRlc19hbmRfLm1kIiwidG9vbF90eXBlIjoiaW5zdHJ1Y3Rpb25hbC1sYWIiLCJhdGxhc19maWxlX2lkIjoxMjkyMywiYWRtaW4iOmZhbHNlLCJpYXQiOjE3Mjk5MDAxMzd9.7uLNMHMmq8RFRKMe9WNzqT7_b_T0rcLcUkv84oKrRcM>

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9DaGVhdF9zaGVldC9DOS1JbnRyb2R1Y3Rpb25fdG9fQ29udGFpbmVyc193L19Eb2NrZXIlMkNfS3ViZXJuZS5tZCIsInRvb2xfdHlwZSI6Imluc3RydWN0aW9uYWwtbGFiIiwiYXRsYXNfZmlsZV9pZCI6MTI5MzksImFkbWluIjpmYWxzZSwiaWF0IjoxNzI5OTAwMjQ5fQ.bV1lXffxaRiCh2vizPZEqE_M-8-cOqCfV4yXQaeNvx4>

# Summary: Application Development using Microservices and Serverless

## **1. Introduction to Microservices**

* Modern software development often delivers centrally hosted, web-based software as a service (SaaS) applications.
* The twelve-factor app methodology maps to the code, deploy, and operate stages of the software delivery lifecycle, enabling developers to create more efficient and more easily maintained SaaS applications.
* Microservices make each application component its own service, and each service communicates via an API.
* Microservices allow application components to use different technology stacks.
* Microservices enable individual components to scale in response to demand.
* Microservices lessen risks associated with change because components can iterate independently.
* Failures in one service do not necessarily impact other services. There are some anti-patterns to avoid when building microservices.
* Microservice patterns provide a structure that reduces reinventing solutions for commonly encountered challenges. The Backend for the Frontend pattern uses microservices to facilitate different user experiences more easily, while the Strangler pattern can help break up monolithic apps into microservices.

## **2. Web API Essentials: REST APIs and GraphQL**

* REST APIs provide flexible but uniform interfaces between components.
* REST APIs are stateless and scalable.
* REST APIs communicate using HTTP methods POST, GET, PUT, and DELETE.
* REST is an architectural style defining how applications communicate.
* API Gateway is the door to your backend services while also enabling you to plug additional services while providing unified access.
* API Gateway makes it easier to scale or replace your backend services.
* Flask is a micro web framework to host Python web applications.
* cURL is used for transferring data with URLs and can be used at a command line or in scripts.
* Postman is a simple and popular tool for building, testing, and using APIs.
* Swagger helps you document and test your API.
* OpenAPI specification is a standard way of representing your APIs.

## **3. Serverless Overview**

* Serverless computing removes the need for infrastructure management by end users and enables developers to focus on their applications’ business-specific needs.
* Serverless uses BaaS services and FaaS platforms.
* Cloud providers take care of infrastructure management and maintenance tasks. The arrival of serverless computing means development teams can focus on writing high-quality code.
* Serverless offers several benefits, such as built-in high availability and fault tolerance, faster function run times, and pay-per-request billing.
* Serverless computing has several constraints, including unacceptable latency for time-critical apps, complex monitoring and debugging, and no state persistence.
* FaaS is a type of cloud-computing service that allows you to execute code in response to events without a complex infrastructure.
* FaaS is a subset of serverless computing that creates applications in the form of multiple functions and can be deployed on cloud, hybrid, or on-premises.
* A serverless stack comprises FaaS, BaaS, and an API Gateway.
* The Serverless Framework is a free and open-source web framework written using Node.js.
* The use cases for serverless web applications are Event streaming, Post-processing, and Multi-language.
* AWS Lambda provides you with an event-driven and pay-as-you-go serverless platform.
* Google Cloud Functions provides you with a simplified developer experience, along with Firebase for real-time data sync.
* Microsoft Azure promotes cloud and edge computing.
* IBM Cloud Functions gives you high availability and cost-effective computing.
* Knative promotes platform-agnostic alternatives.

## **4. Create and Deploy Microservices using Serverless**

* Self-hosted microservices can be very complex and challenging.
* IBM Cloud Code Engine is a fully managed platform that handles all the hard deployment work, allowing developers to focus on their code.
* IBM Cloud Code Engine has three main use cases: 1) deploy applications, 2) build and deploy applications, and 3) run jobs.
* A project in Code Engine represents a group that contains and manages its resources and entities. A grouping in Code Engine contains entities such as build, app, job, and certificate for transport layer service (or TLS) HTTPS connections, and so on.
* An application runs your code to serve HTTP requests or to create WebSocket sessions.
* A build is a process to create a container image from your source code.
* A job runs one or more instances of your executable code.
* A container is a standalone executable software unit packaged with all its dependencies.
* Docker is a popular container-building and running platform.
* You can compose a Dockerfile to instruct the Docker platform to build a container image. After the container image is built, you can push it to the container registry and then pull it by its image name.
* You can create a Cloud Engine application either from a pushed container image or a source code repository. As per your preference, you can choose to use the IBM Cloud Console or IBM Cloud CLI to perform the application deployment tasks.

## **5. OpenShift Essentials/Working with OpenShift and Istio**

* Red Hat OpenShift is a platform for running containerized workloads like microservices.
* OpenShift is like a Kubernetes distribution in that OpenShift with additional capabilities. OpenShift services help manage workloads, build cloud-native apps, and increase developer productivity. For example, OpenShift creates a Jenkins job to build microservices into containers automatically. In addition, OpenShift pushes the built containers to a registry and deploys those containers to the OpenShift cluster.
* Microservices architectures need security among services as well as ways to manage and test services.
* A service mesh is a dedicated layer that provides security and more by coordinating communication. Istio is a service mesh that provides traffic shifting, mutual transport layer security, and telemetry when deployed with microservices.
* Certified software fills development gaps in your application, eliminating the need for your organization to spend the time and money to develop new microservices. Red Hat Marketplace provides a central location to try, buy, deploy, and manage software certified for OpenShift environments.

<https://author-ide.skills.network/render?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtZF9pbnN0cnVjdGlvbnNfdXJsIjoiaHR0cHM6Ly9jZi1jb3Vyc2VzLWRhdGEuczMudXMuY2xvdWQtb2JqZWN0LXN0b3JhZ2UuYXBwZG9tYWluLmNsb3VkL0lCTVNraWxsc05ldHdvcmstQ1MwMjAzRU4tQ291cnNlcmEvbGFicy9HbG9zc2FyeS9BcHBsaWNhdGlvbl9EZXZlbG9wbWVudF91c2luZ19NaWNyb3NlcnZpY2VzX2FuZF9TZXIubWQiLCJ0b29sX3R5cGUiOiJpbnN0cnVjdGlvbmFsLWxhYiIsImF0bGFzX2ZpbGVfaWQiOjEyOTE3LCJhZG1pbiI6ZmFsc2UsImlhdCI6MTcyOTg5OTk4MH0.OZIwWMcuTn3HjUk2oL0ZLnLeaU892Z_nF70IQ0JgTJo>