Full Stack Application Development

Application Development- Introduction

Akshaya Ganesan

Recap of CS1:

- Introduction
- Web Application
- Difference between Website and Web application
- Native App, Hybrid App, andCross-platformm App

Agenda for CS2

- Cloud-Native Applications
- Serverless Applications
- Layered Architectures (client /server, 2/3 tier- N tier)
- Monolithic
- Distributed Architectures – Service-oriented architecture, Microservices

Cloud Native Application

- Cloud-native is an approach to developing, deploying, and running applications using modern methods and tools.
- The Cloud Native Computing Foundation(CNCF) defines

  Cloud-native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

Why Cloud Native

- Monolithic Application
  - A monolithic architecture refers to a traditional software design approach where an entire application is built as a single, self-contained unit.
- Key characteristics of monolithic architecture include:
  - ❑ Single Codebase
  - ❑ Tight Coupling
  - ❑ Single Deployment Unit
  - ❑ Centralized Database
  - ❑ Development and Scaling Challenges
  - ❑ Longer Development Cycles
  - ❑ Limited Fault Isolation

Cloud-Enabled Solutions

- What happens when you move a monolithic app from on-premises to cloud?
- A cloud-based application is an existing app shifted to a cloud ecosystem.
- They are not able to utilize the full potential of the cloud
  - Not scalable
  - Less automation
  - Longer Time to Market

- Cloud-Native Applications are designed to run on the cloud computing architecture.
- With cloud-native applications, you can take advantage of the automation and scalability that the cloud provides.

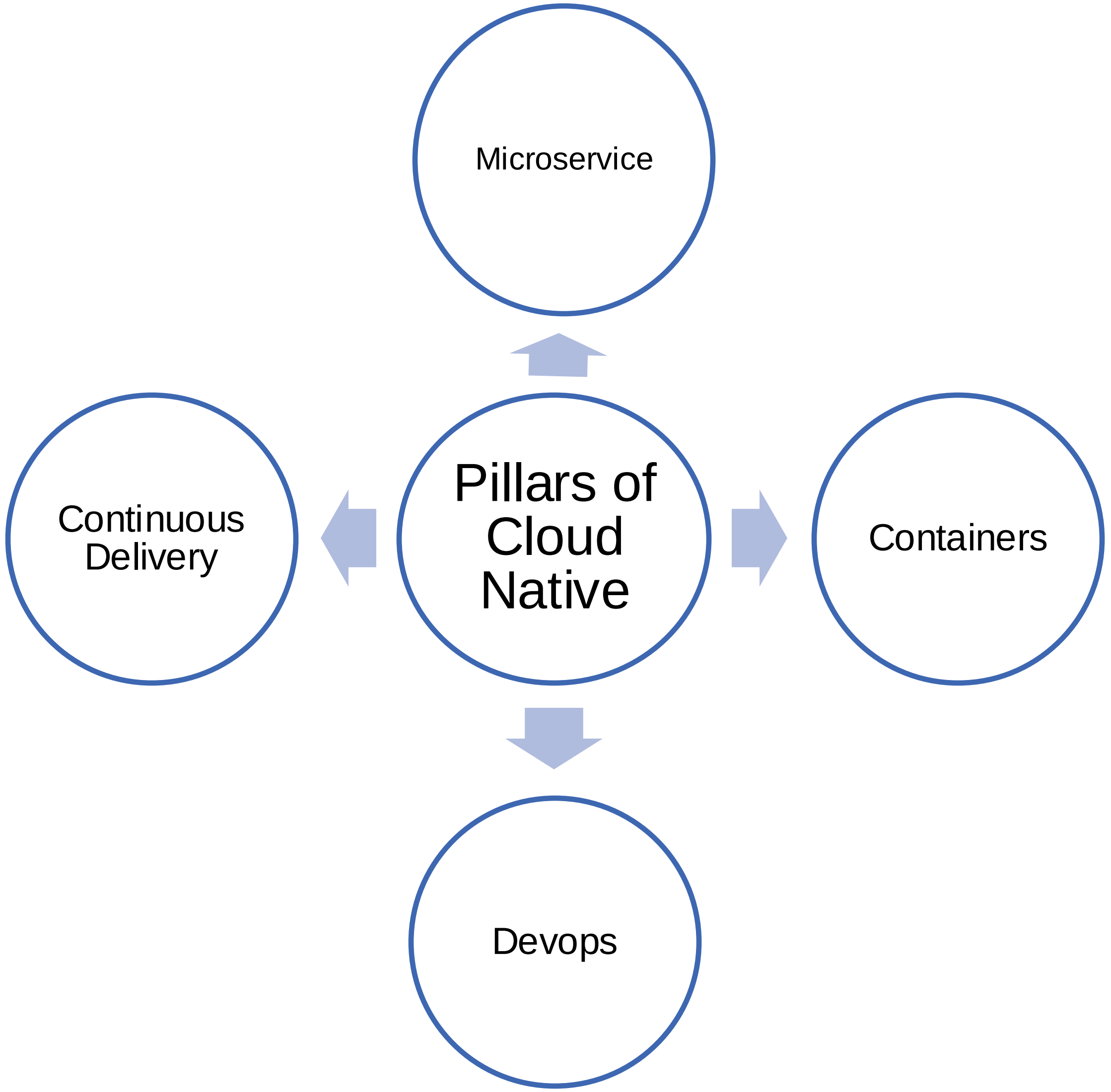
Cloud native vs. traditional applications

- A cloud enabled application is an application that was developed for deployment in a traditional data center but was later changed so that it could also run in a cloud environment

- Cloud-native applications, however, are built to operate only in the cloud.

- Developers design cloud-native applications to be
  - Scalable
  - platform agnostic
  - and comprised of microservices

Cloud Native Applications



Building Blocks

- Microservices
  - ✓ is an architectural approach to developing an application as a collection of small services
  - ✓ each service implements business capabilities, runs in its own process and communicates via HTTP APIs or messaging
- Containers
  - ✓ offer both efficiency and speed compared with standard virtual machines (VMs)
  - ✓ Using operating system (OS)-level virtualization, a single OS instance is dynamically divided among one or more isolated containers, each with a unique writable file system and resource quota
  - ✓ Low overhead of creating and destroying containers combined with the high packing density in a single VM makes containers an ideal compute vehicle for deploying individual microservices

Building Blocks

- DevOps
  - ✓ Collaboration between software developers and IT operations to constantly deliver high-quality software that solves customer challenges
  - ✓ Creates a culture and an environment where building, testing, and releasing software happens rapidly, frequently, and more consistently

- Continuous Delivery
  - ✓ is about shipping small batches of software to production constantly through automation
  - ✓ makes the act of releasing reliable, so organizations can deliver frequently, at less risk, and get feedback faster from end users

Advantages

- Reduced Time to Market
- Ease of Management
- Scalability and Flexibility
- Reduced Cost

Serverless Application

- Serverless architecture is an approach to software design that allows developers to build and run services without managing the underlying infrastructure.
- Cloud service providers automatically provision, scale, and manage the infrastructure required to run the code.

Function as a Service

- One of the most popular serverless architectures is Function as a Service (FaaS)
- Developers write their application code as a set of discrete functions.
- Each function will perform a specific task when triggered by an event
- When a function is invoked, the cloud provider executes the function
- The execution process is abstracted away from the view of developers.

  - Examples:
    - AWS: AWS Lambda
    - Microsoft Azure: Azure Functions
    - Google Cloud: Cloud Functions

Serverless

- Benefits
  - Reduced operational cost
  - Easier operational management
  - Scalability
- Drawbacks
  - ○ Loss of control
  - ○ Vendor lock-in
  - ○ Multitenancy problems
  - ○ Security concerns

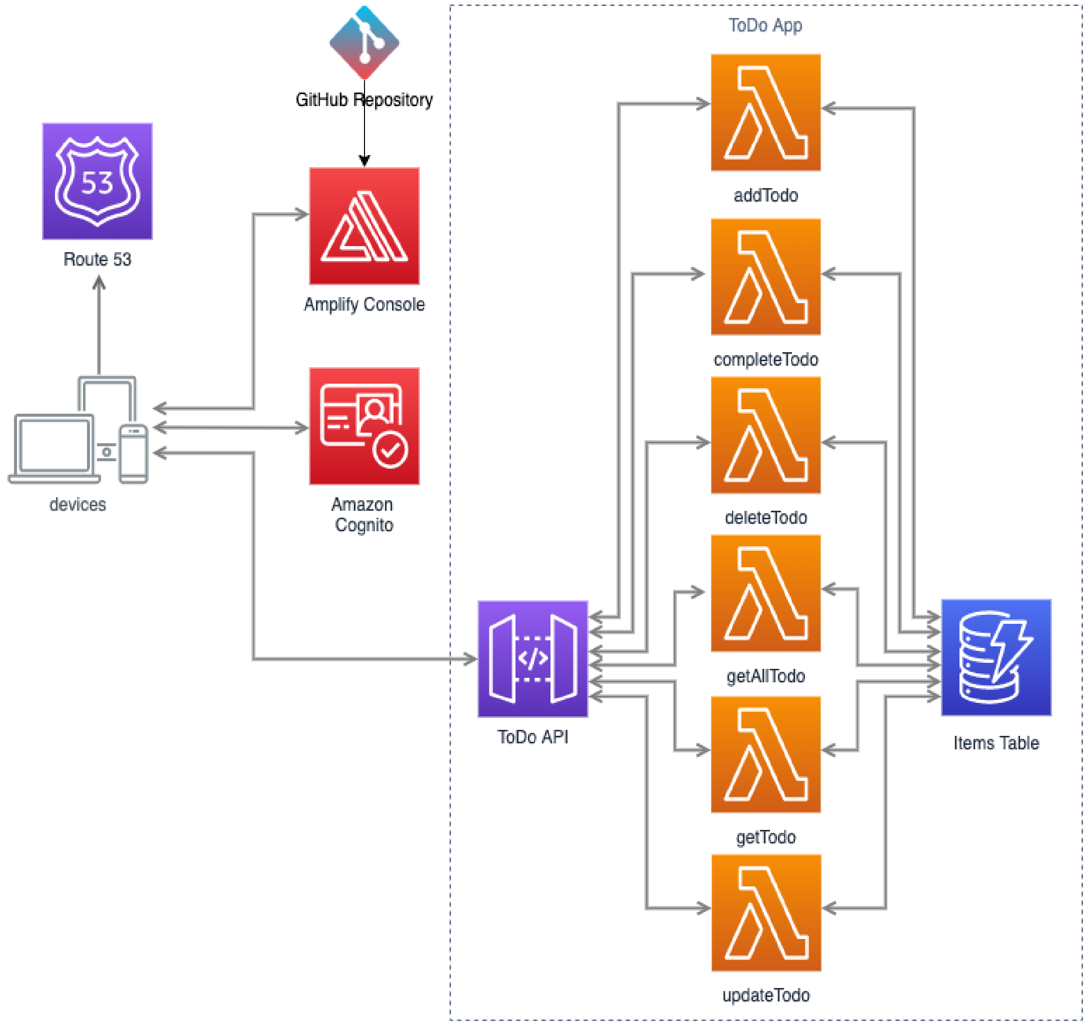
AWS Serverless

- AWS provides a set of fully managed services that can be used to build and run serverless applications
- AWS Lambda
  - Allows to run code without provisioning or managing servers
- AWS Fargate
  - Purpose-built serverless compute engine for containers
- Amazon API Gateway
  - Fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale

Source : AWS Serverless

Example: AWS Serverless Architecture

- General-purpose, event-driven, web application back-end that uses
- AWS Lambda, Amazon API Gateway for its business logic
- Uses Amazon DynamoDB as its database
- Uses Amazon Cognito for user management
- All static content is hosted using AWS Amplify Console



Source : AWS

Reference Architecture – Web Application

Thank You!