SEMAT

Essence
Software Engineering Essentialized

Essence
Software Engineering Essentialized
Part 3B – Running with Microservices

Giuseppe Calavaro, Ph.D.

www.semat.org

Agenda of this Teaching Module

- Microservices Introduction
  - Microservices vs Components
- Microservices Lite Practice using Essence example
  - Microservices Lite Alphas
  - Microservices Lite Products
  - Microservices Lite Activities

Example of using Microservices Lite Practice on our project are in green boxes

- The Value of the Kernel to the Microservices Practice
- Impact of Microservices Lite practice for the team

Introduction to Microservices concept

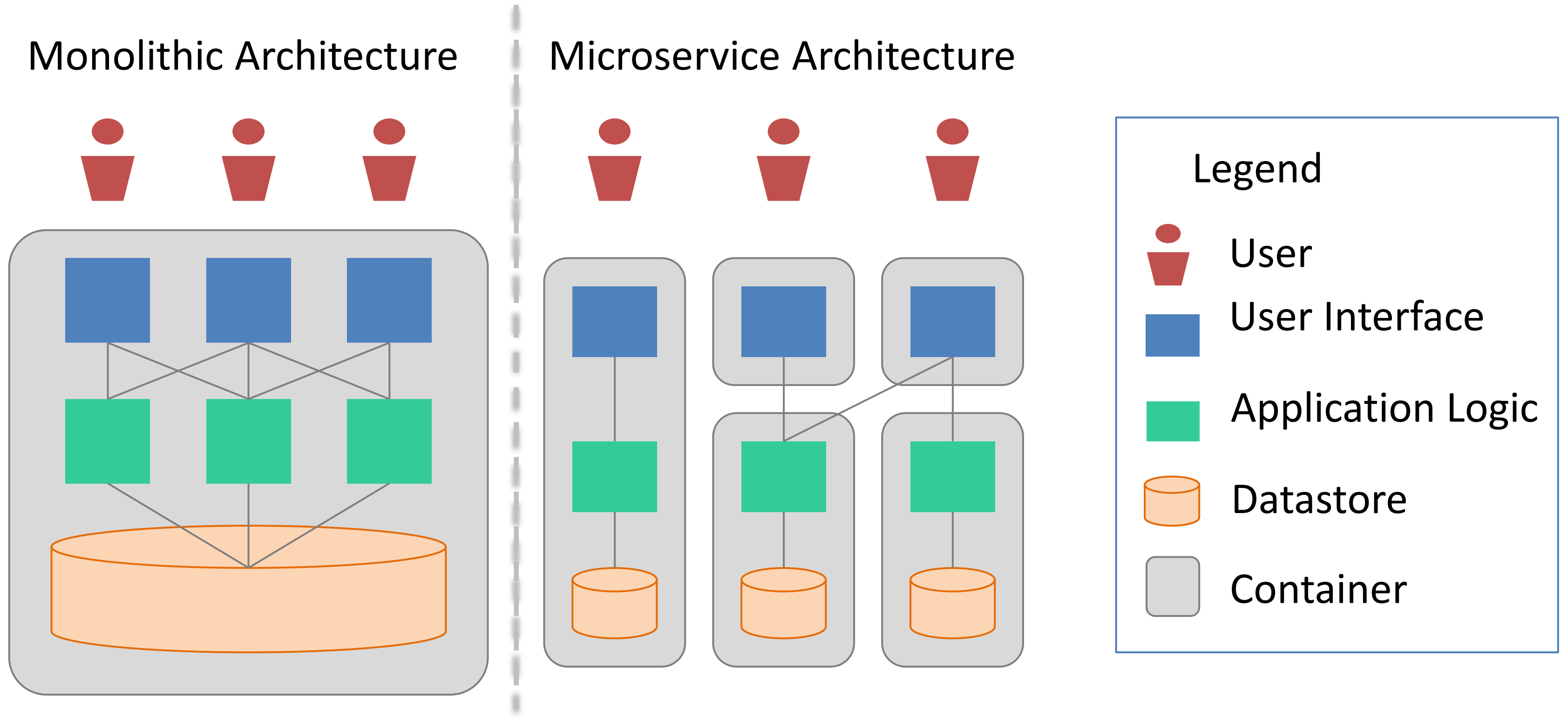
- Developing software with **microservices is an evolution of designing software with components**
  - which facilitates a modular approach to building the software system.
- A **component** can be described as an element containing code and data.
  - Only the code "inside" the component (object) can modify the data inside the object and it does that when another component sends a message with a request to do that.
  - This idea is known as "**data hiding**" (data is hidden to other components) and is an accepted best practice in developing maintainable software.
- What Microservices has added is support for components all the way from design, code to deployment

Components vs Microservices

- Like components
  - microservices are interconnected via interfaces over which messages are sent to allow communication.
  - each microservice can evolve separately from other microservices, thus making it easy to introduce new functionality
- In a software system built from microservice, each microservice runs a unique executing process.
  - There may be several such executions or instances of the same program running in parallel.
- What microservices bring to software beyond what components already did is the **ability to also deploy the microservices independently without stopping the execution of the entire software system**.

SEMAT

Microservices Big Picture



- **User Interface** – A user interface is the part of a software system that users interact with. It is the screens, and buttons, and so on.
- **Application Logic** – The code behind the user interface that performs computation, move data around, etc.
- **Data Store** – The data retrievable by the application logic lives in a data store.
- **Containers** – Containers are components of a software system that can be managed separately (i.e. started, stopped, upgraded, and so on)

Advantages of Microservices

- Each microservice runs
  - as a separate process,
  - possibly in its own container or virtual machine
  - it has its own programming language, user interface, application logic and data store.
- This architecture allow developers to upgrade each microservice independently
  - For example you can upgraded a microservices from Java 8 to Java 9 or a data store without impacting other microservices.
  - If however, code for different logical software element were to run in the same process or virtual machine, an upgrade of one element may inadvertently impact another element.
  - Thus, **enhancing the functionality of an existing microservice is easier** than that of a monolithic software system.