“Università degli Studi di Trento”

Corso di Ingegneria del Software

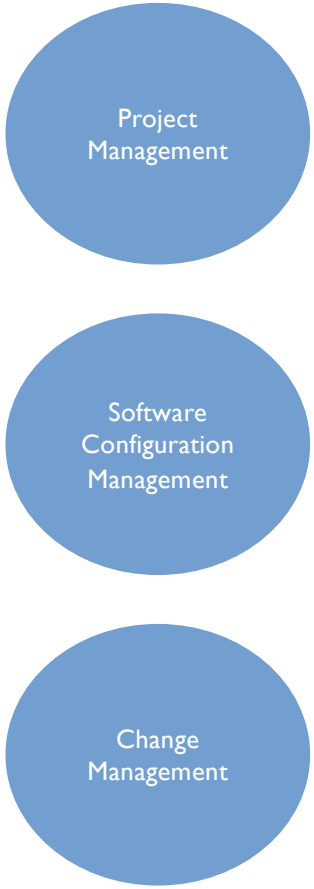
Talk on DevOps and Microservices

Gerardo Marsiglia
IT Consultant

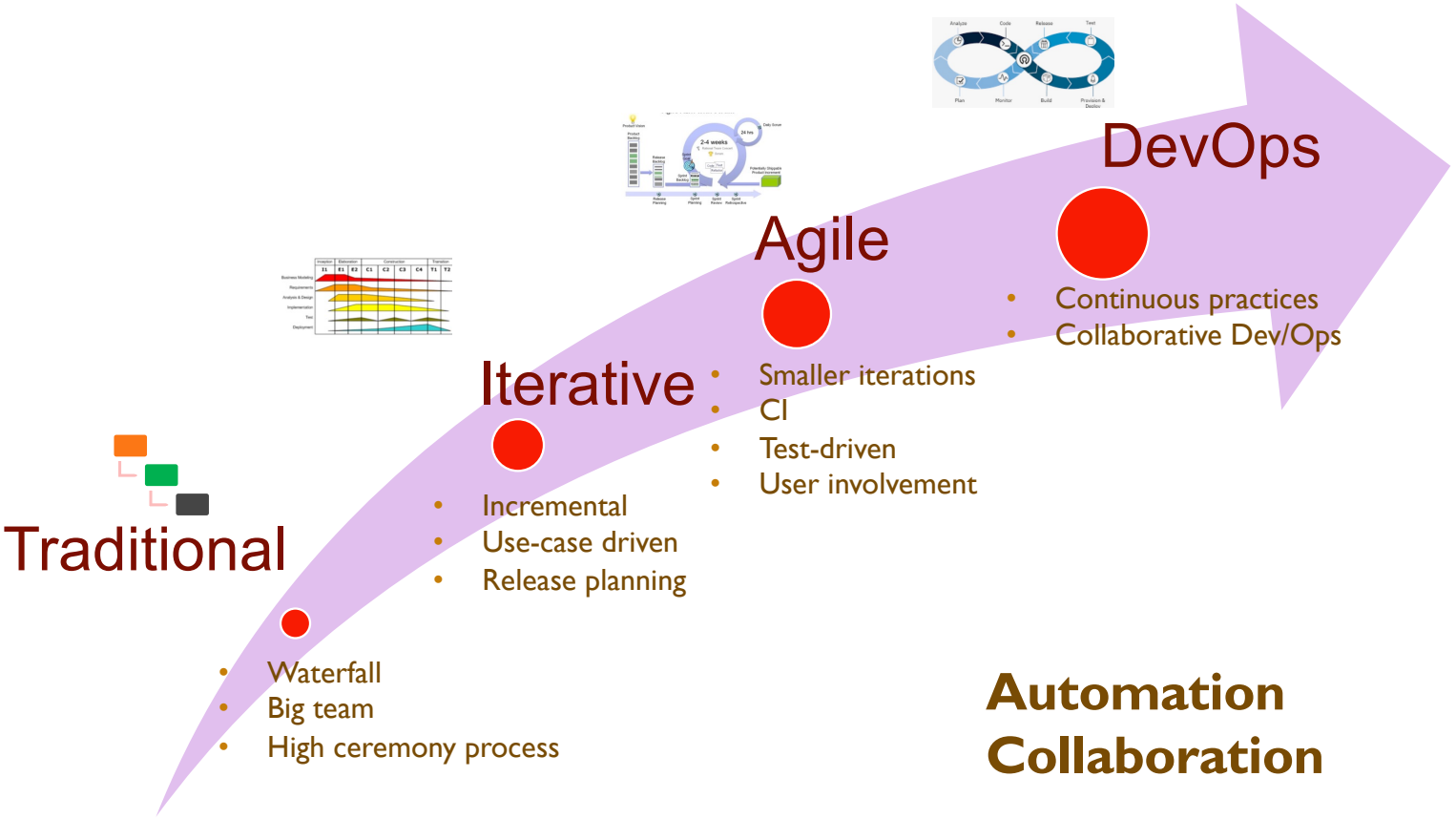
20 Novembre 2024

The software lifecycle – a typical representation

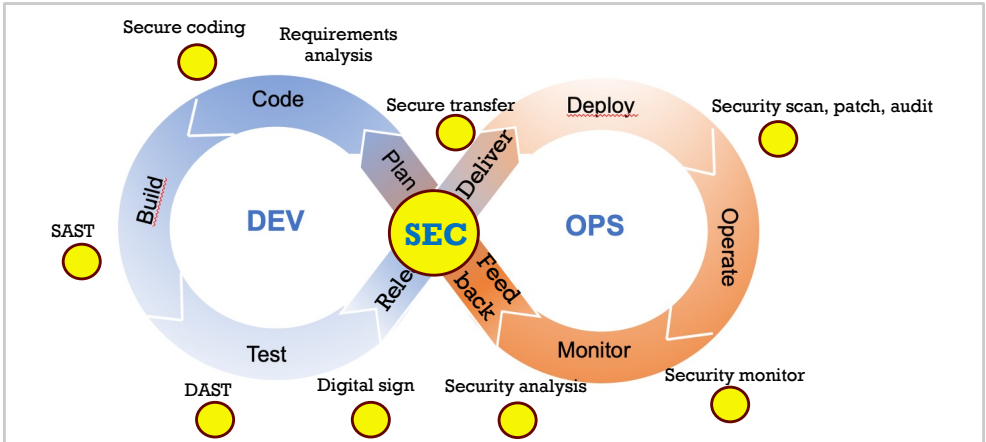
Plan	• Risks,Quality, Project planning
Requirements	• Business and Requirements gathering
Analysis/Design	• Requirement analysis • Define and design architecture, sw design
Code	• write code, manage and share versions
Test	• run functional/non functional tests, manual / automatic • test pan, test case, test suites • produce test reports
Deploy	• install the sw in the correct env • promote sw through the env
Mantain	• Ensure the software consistently support the end user, enhance, fix, new features



Evolution of the software development processes and practises



Adopting practices for Continuous Delivery



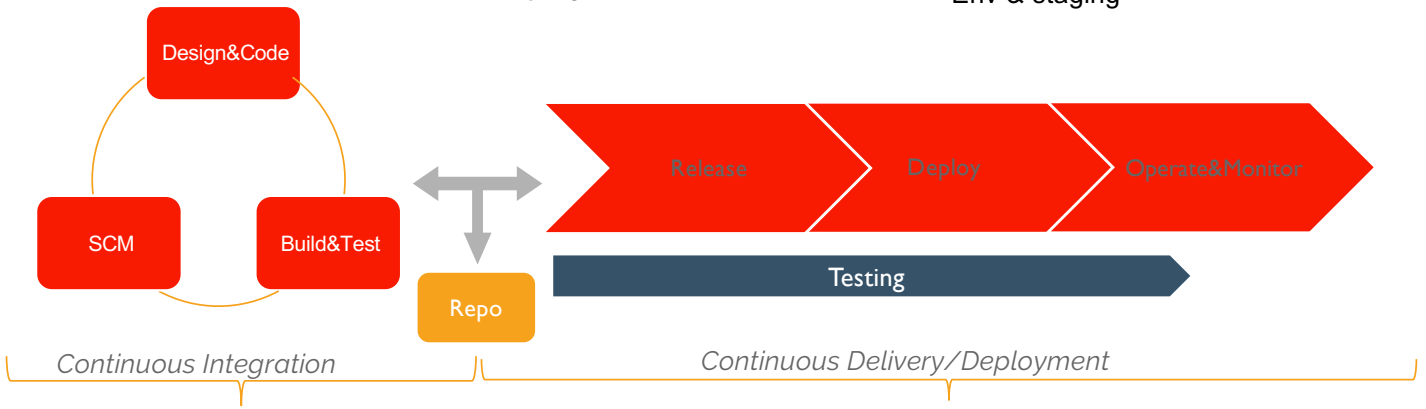
Pipelines
the set of practises (supported by automation)
that Dev&Ops implement to deliver the software

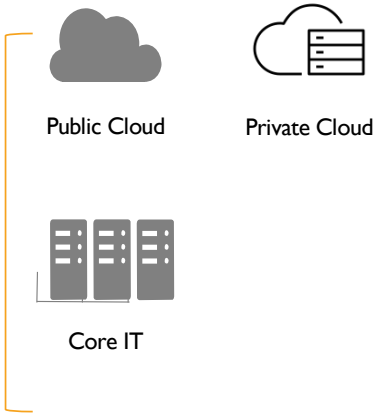
Several and specialized tools
to support the steps

- Open source
- Market/Vendor

Automation is strategic

- *Build automation, Manage*
- *Test automation*
- *Deployment automation*



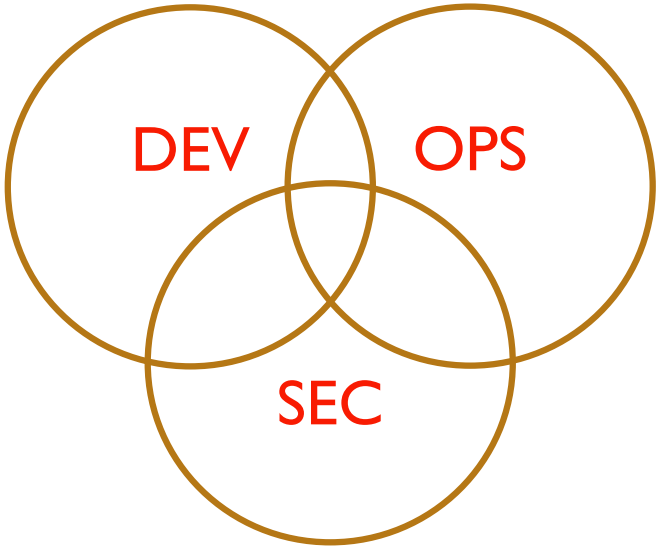


The Dev(Sec)Ops – be lean and agile

Practises for "Continuous …."

- Integration
- Delivery
- Deployment
- others

Collaborative Development/Operation with Security



LEAN: Reduce work, Remove bottlenecks, Eliminate waste

AGILE: Fast response times, Small batch sizes, Feedback

Cloud computing

Cloud Computing Services

- SaaS (Software-as-a-Service)
- PaaS (Platform-as-a-Service)
- IaaS (Infrastructure-as-a-Service)
- …

Cloud Computing types

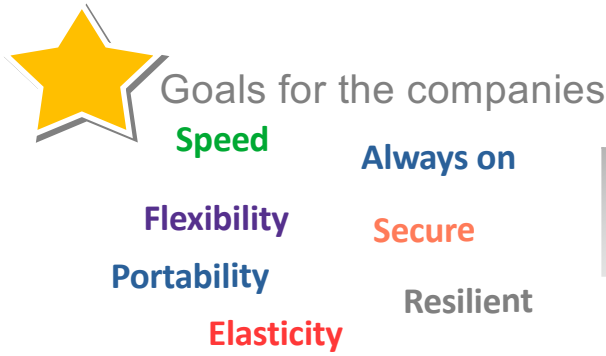
- Public
- Private
- Hybrid
- …

Cloud services providers

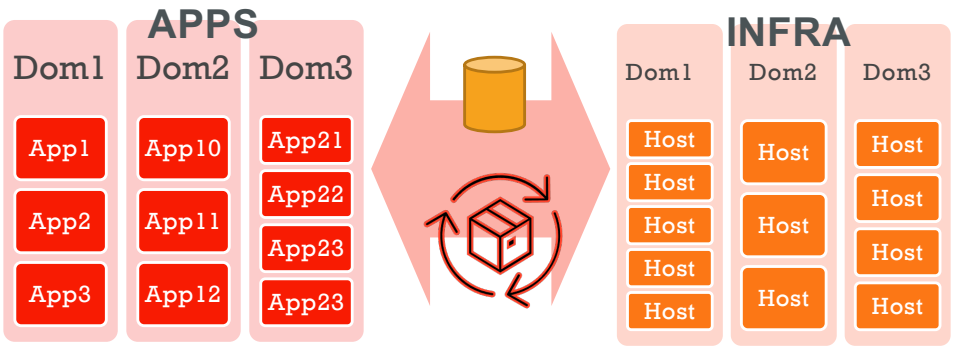
Others:
- Oracle
- Alibaba
- Salesforce
- …….

- New development paradigm
- Coexistence with the legacy and traditional
- Migration of on-premises workloads
- Fits for the new technologies (i.e. with AI - machine learning, IoT, Big data analytics)
- Speed and flexibility
- Strategic for application modernization
- DevOps
- New roles

Application modernization – why, what, how



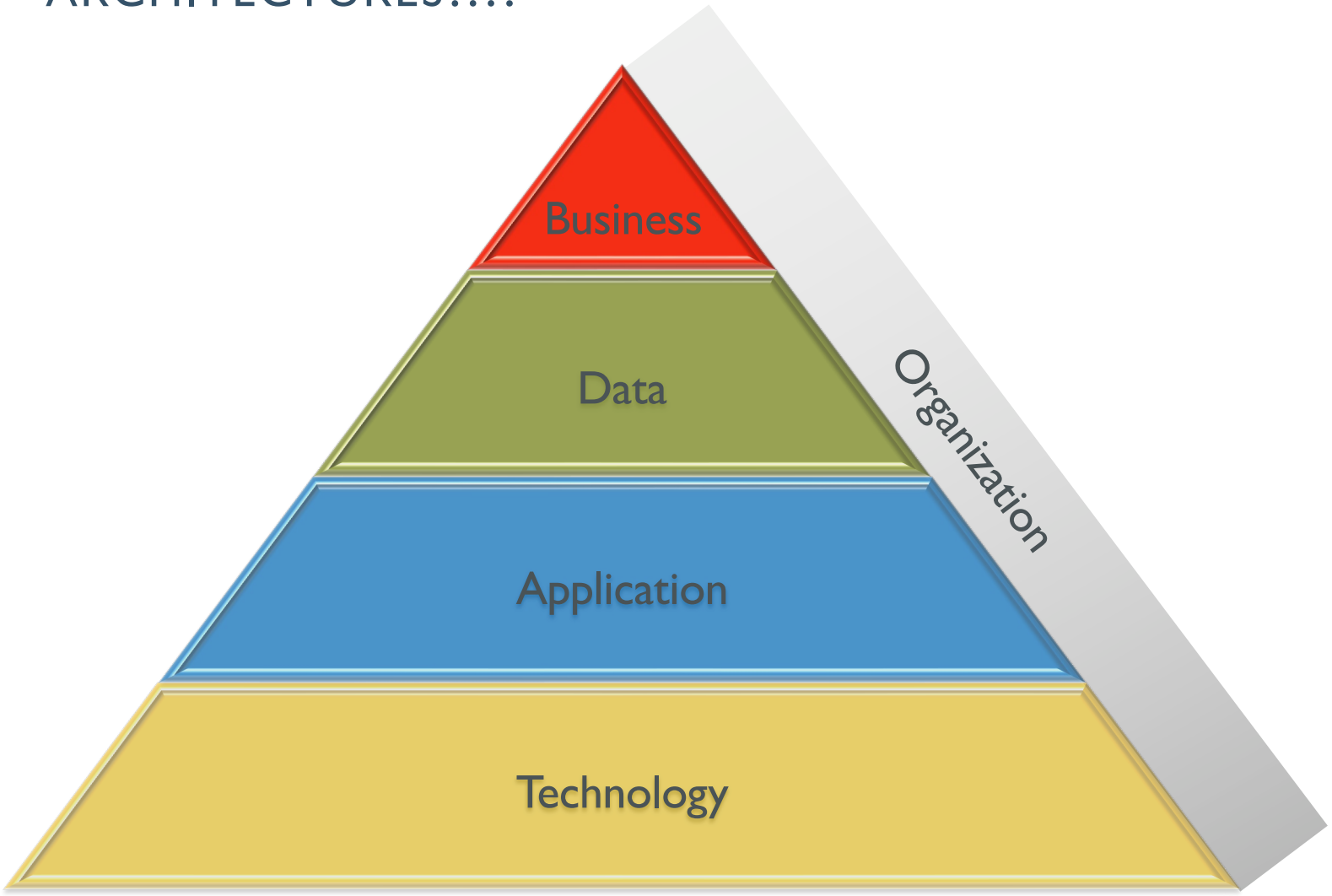
Need to **modernize** the traditional applications in order to get the *advantages* from new *sw architectures*, new *technologies*, new *platforms*, and the *cloud potentiality* and *opportunities*.



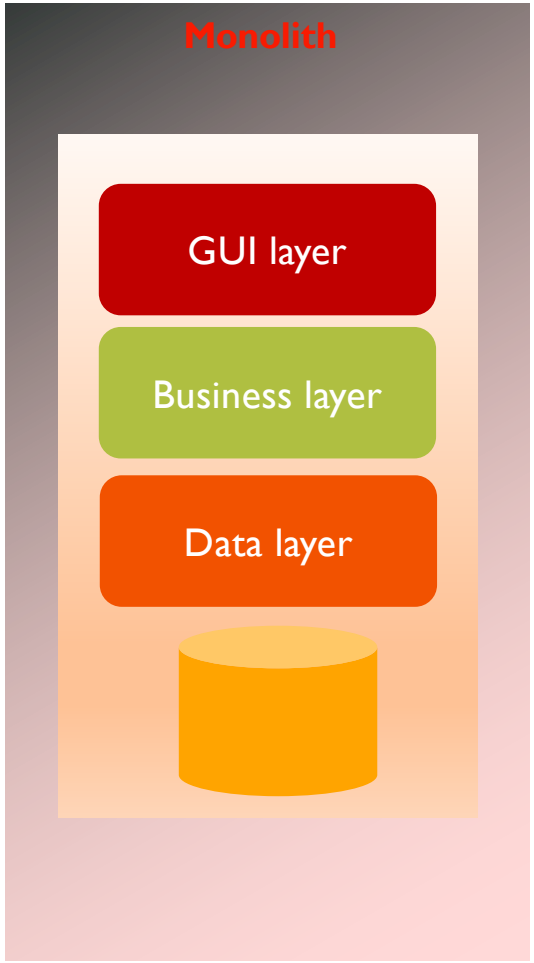
The modernization is a journey

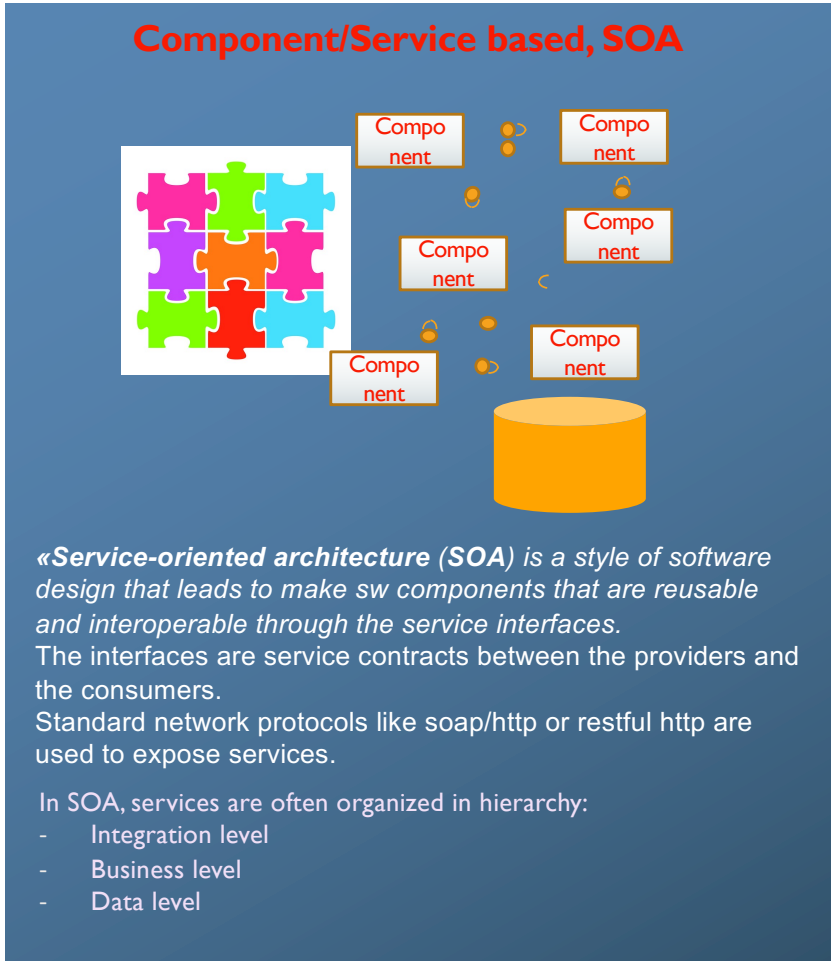
- understand the current landscape (applications and infrastructure)
- evaluate the target evnironments
- analyse and decide what, when and how to modernize and migrate applications to the target env
- adopt new architectures and technology
- adopt new platforms (i.e.: Kubernetes, …)
- change/improve processes (bus, dev, ops, ….) and introduce new practices
- skills / new roles

ARCHITECTURES….



Software architecture – the evolution part I



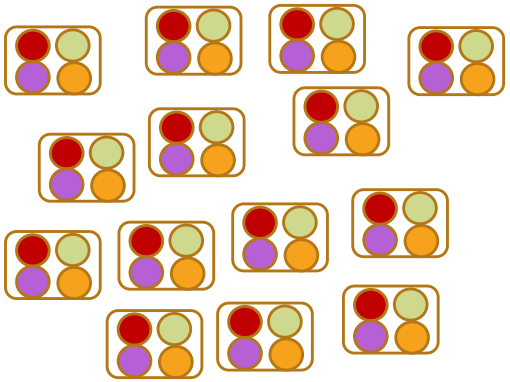


Software architecture
Defines how the software is organized, structured in terms of elements and their relationships.
It also encompasses the important and strategic design decisions.

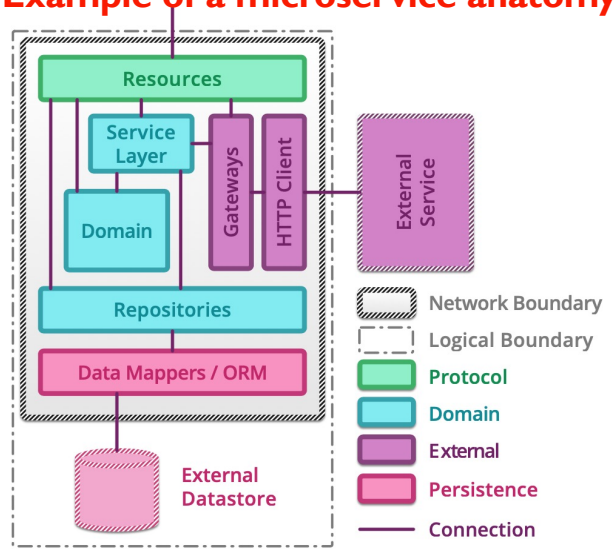
A good ***architecture design*** is a key for the successful building of sw systems.

Software architecture – the evolution part II

Microservices architecture



Example of a microservice anatomy



Source: https://martinfowler.com

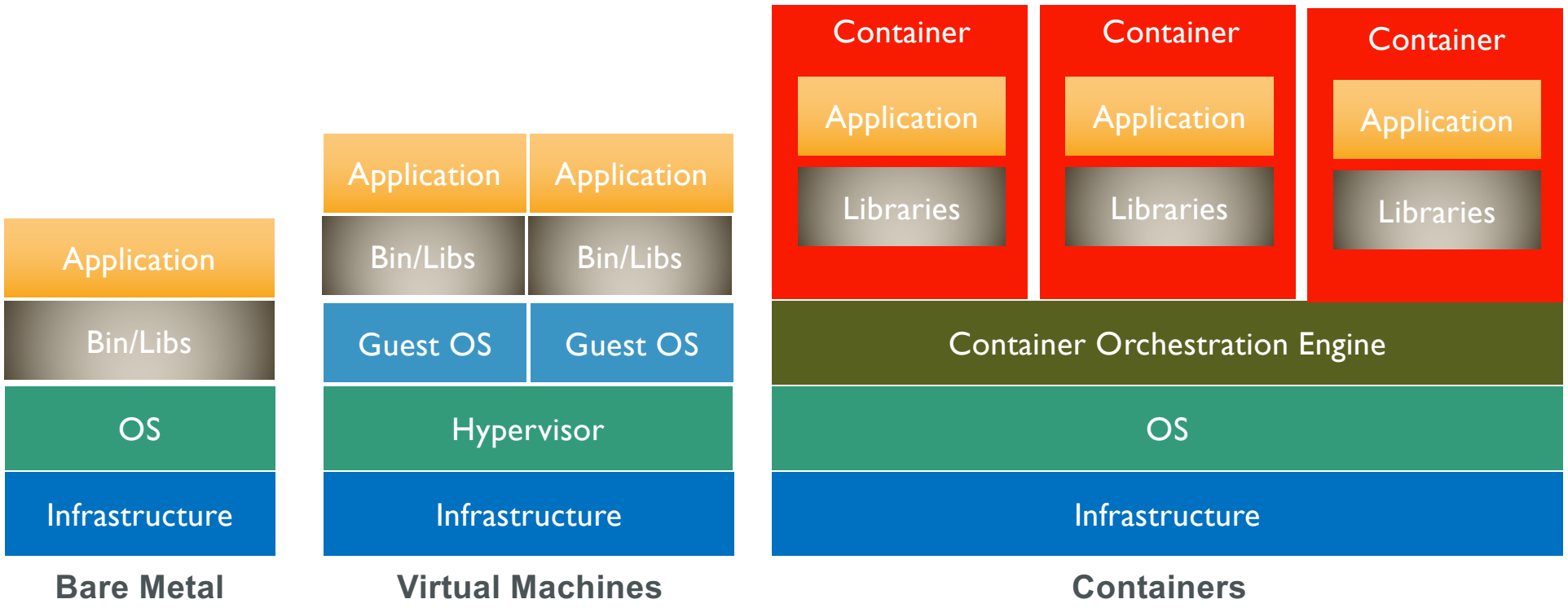
«**Microservices** architecture defines a way to structure an application as a collection of loosely coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight»

- Fits well for cloud computing
- Based on small, independent and loosely-coupled services (microservices)
- Adopting devops CI/CD
- Can run on a container based infrastructure or not
- Define and make use of API
- Modern design paradyghm… «The 12 factor application»
- They take benefits of the cloud and container capabilities for scaling, recovering, resource optimization,

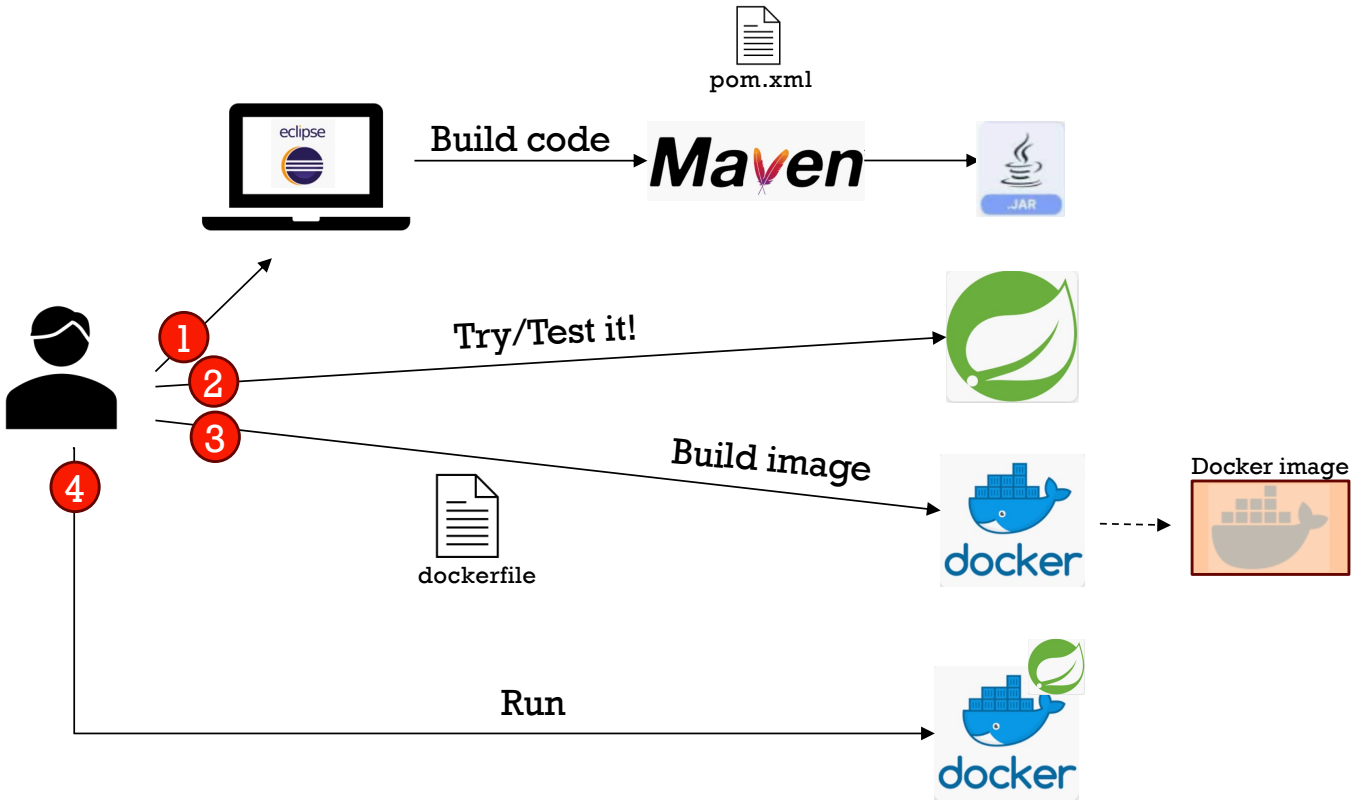
- Fine grained
- Independent deployable
- Decentralized governance and data
- Fits for build&deploy automation
- Fits well for Cloud-native application

Bare metal vs VM vs Containers

A **container** includes everything (the minimal needed resources) the application / microservice needs to run (code, dependencies, libraries).



DEMO



Springboot

“Università degli Studi di Trento”

Corso di Ingegneria del Software

Thank you

Grazie

Gerardo Marsiglia
IT Consultant

20 Novembre 2024