RPC
Member of RBI Group

Container-oriented Software Architecture, with implications

Dr. Marko Rankovic

Agenda

1. What is the Container-oriented architecture?
2. Container-based application design – the main characteristics
3. How containers fit into the existing landscape?
4. Practical implications of the containerization
5. When you should use it?
6. Feedback from the experience and the practical use

About RPC

- RPC – part of Raiffeisen Bank International
- Card Payment Processor and Innovation Hub
- Processing more than 1.5 billion transactions per year
- Processing close to 2 billion API calls per year
- Servicing Customers in 11 CEE/SEE countries

About me

- Spent most of the professional career in banking industry, namely in the payment cards processing area.
- Currently, Deputy CEO, COO, Member of the Board and Executive Director of Regional
- Card Processing Center - RPC, by far largest card payments processing company in the CEE and SEE regions, 100% subsidiary of Raiffeisen Bank International.
- Held/holding numerous lecturing and teaching positions, as: Professor – Information Technology School Belgrade, Assistant Professor - Faculty of IT and Engineering Belgrade, Research Fellow – Institute of Economics Belgrade.
- Published 4 books, and more than 50 articles in scientific and expert journals.
- Happily married and proud father of 4.

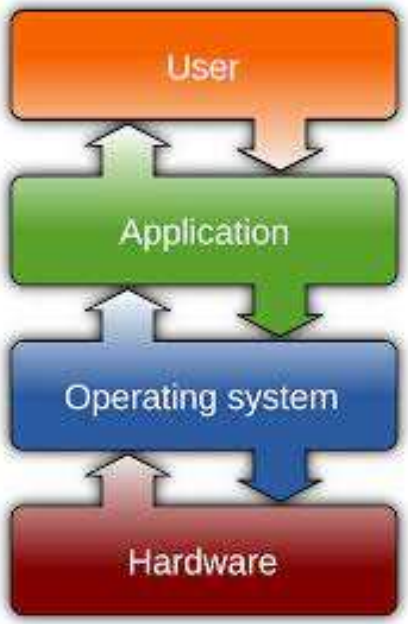
What is **Container-oriented architecture?**

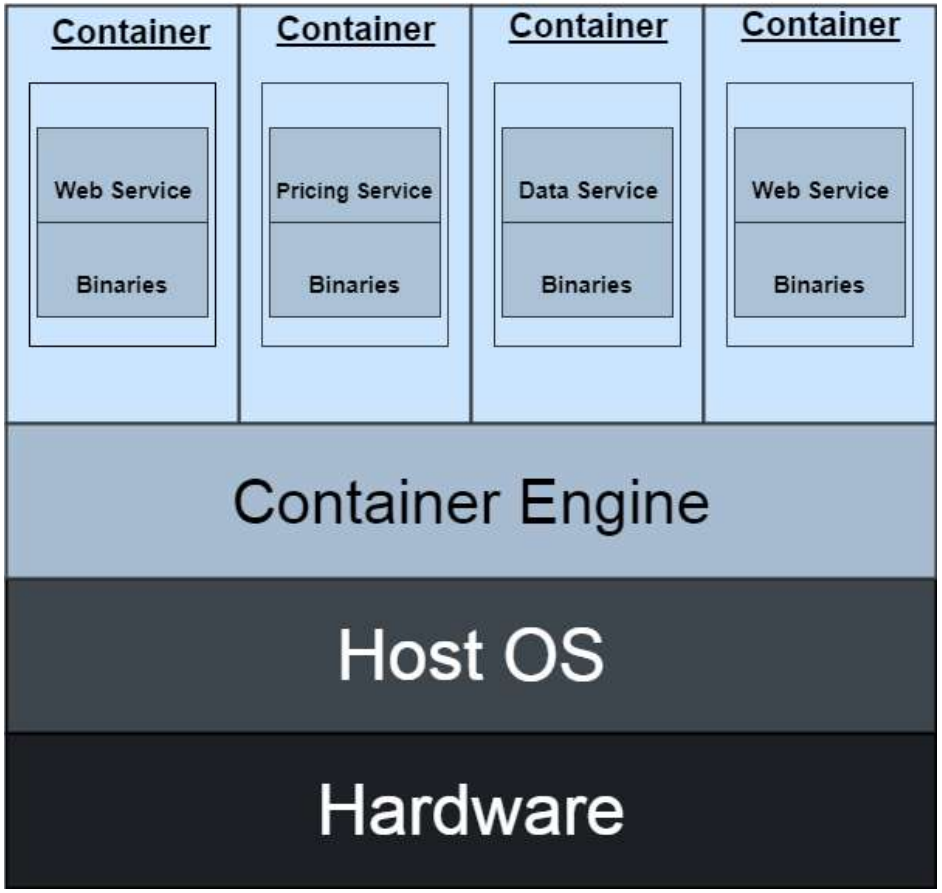
Containerization or container-based virtualization is an Operating System level virtualization method for deploying and running distributed applications without launching Virtual Machines for each application.

Container-oriented architecture empowers us to deploy packaged SW deliveries, as an independent/isolated SW unit.

Do you know which programming concept enabled containerization?

What is **Container-oriented architecture?**

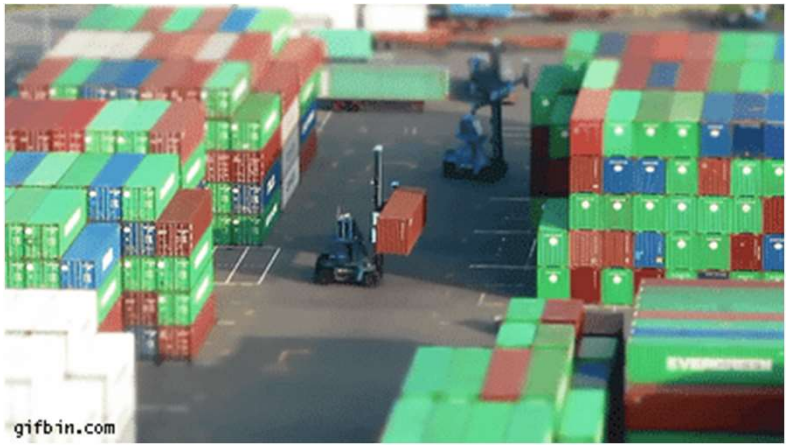




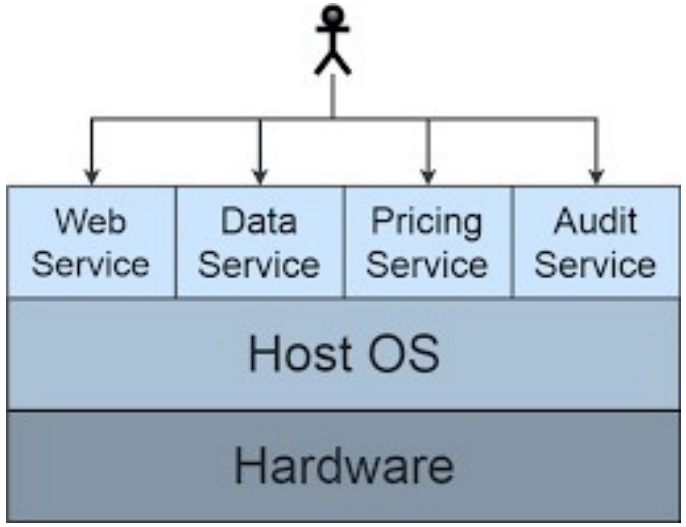
What is **Container-oriented architecture?**

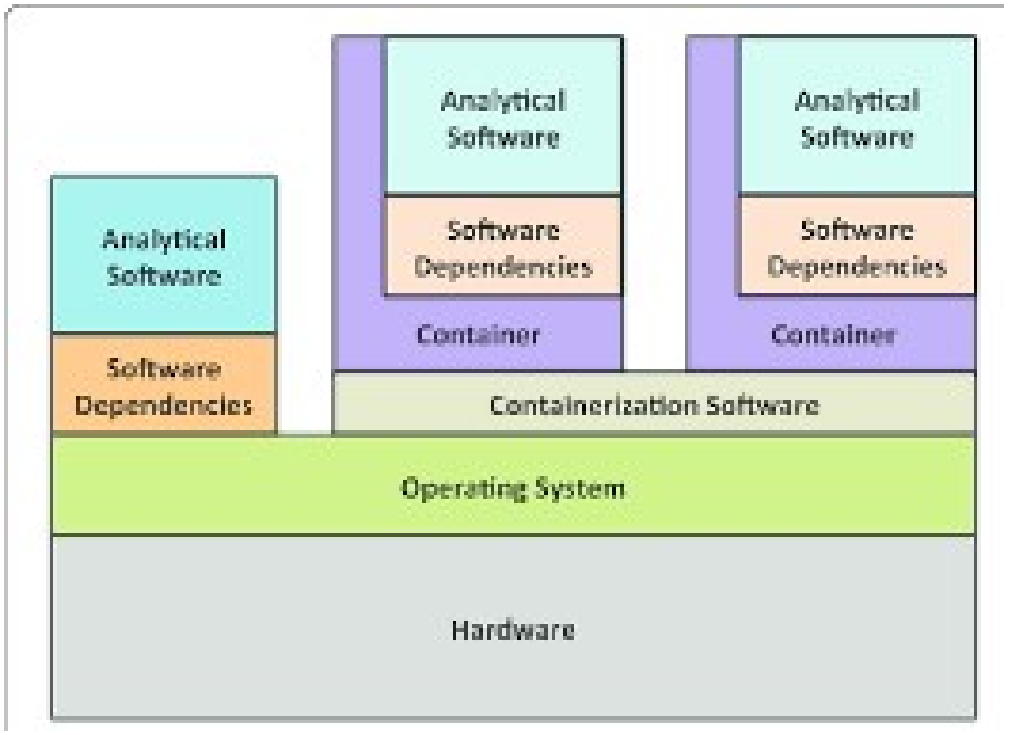
Containers:

- are fully portable, independent SW units
- are compatible within the particular environment, but can be easily adopted (if needed)/deployed in other environments
- are similar to virtual machines in the virtualized environment
- are lightweight (comparing to the VM)
- can have either specific and general functionality

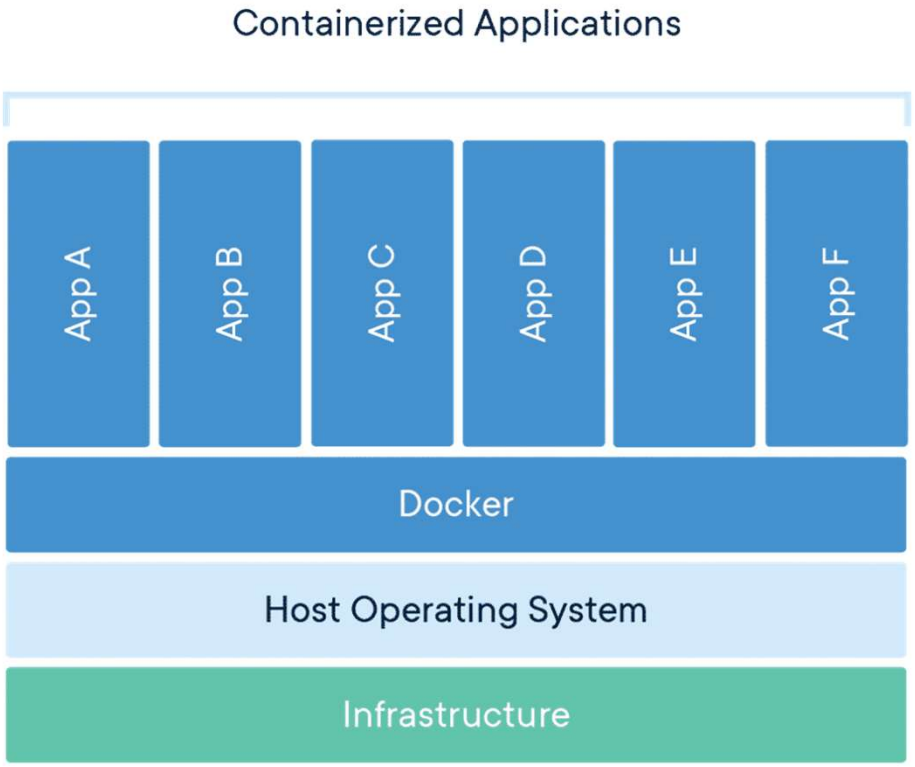


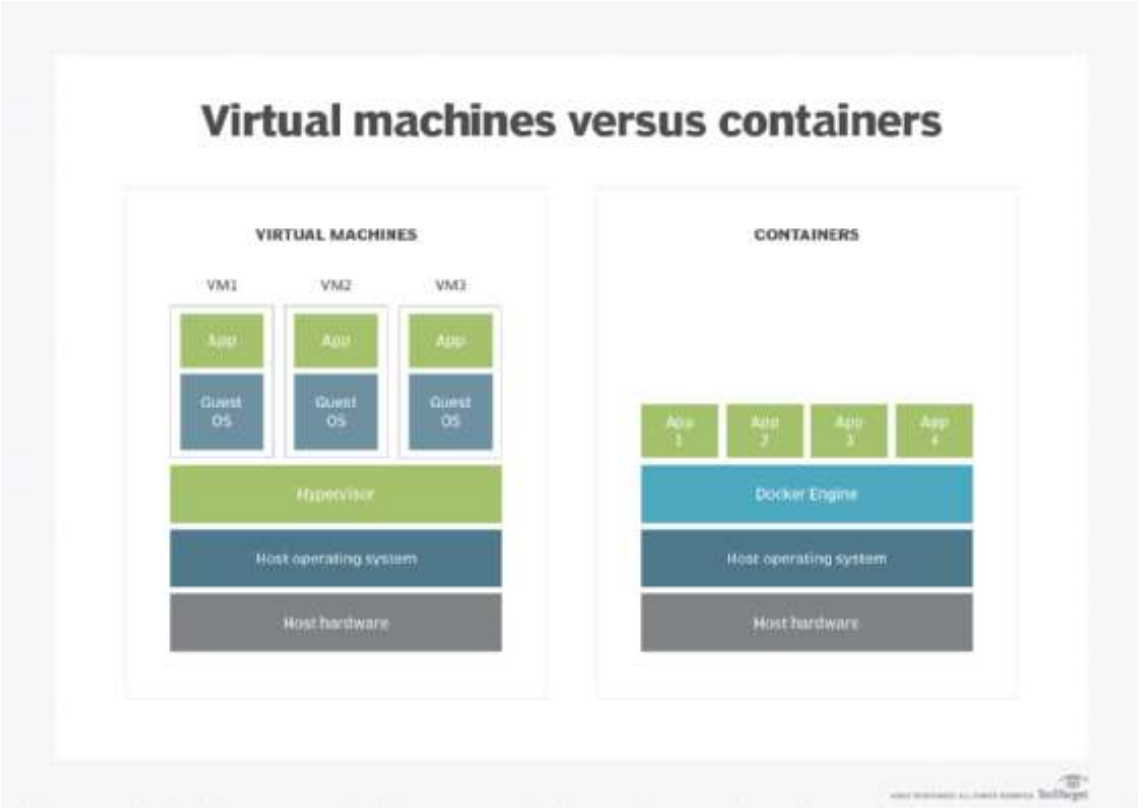
What is **Container-oriented architecture – other graphical examples**





What is **Container-oriented architecture – other graphical examples**





Container-based applications – the main design characteristics

- Observability
- Image immutability
- Disposability
- Security
- CI/CD pipeline

Container-based applications – the main design characteristics

Observability

- Container health check
- Monitoring
- Logging engine (and logs interpretation intelligence)
- API-based

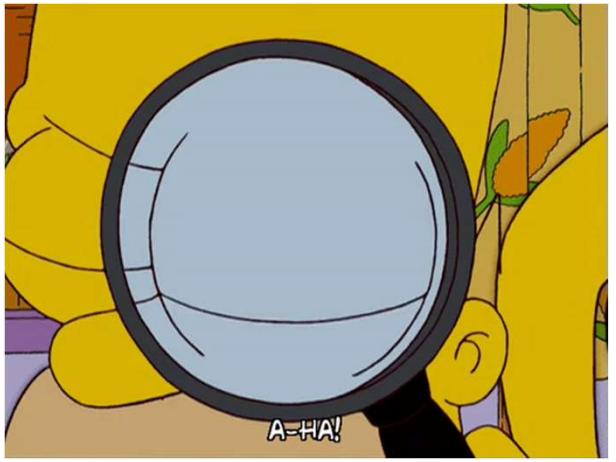


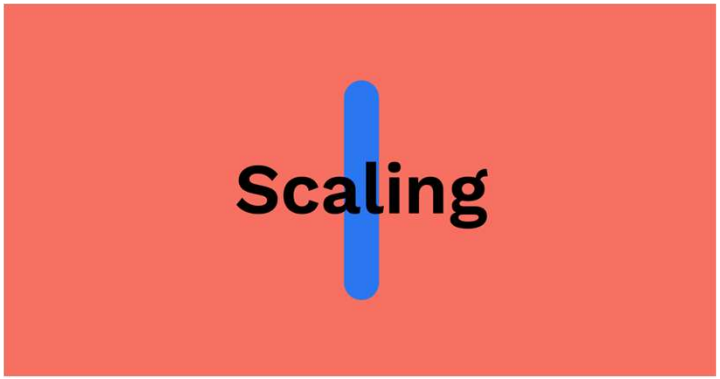
Image immutability

- Versioned deployments
- Copy image, build new, deploy, dispose
- Automated pipeline
- Backward compatibility

Container-based applications – the main design characteristics

Disposability

- Scale
- Fix
- Deploy
- Shut-down



Security

- Managed access
- Trustworthy images
- Security testing tools (i.e. RASP)
- Vulnerability scans
- Backward compatibility

Container-based applications – the main design characteristics

RPC
Member of RBI Group

CI/CD pipeline

- Easy to execute
- Automated deployments
- No manual interventions
- Upgrade/update tools

How containerization fits into the existing landscape?

Business environment landscape

- Fast-changing
- Cost-efficient
- Flexible
- Highly customizable
- …

Technological environment landscape

- Fast deployment
- Plug-and-play
- Impact on the overall system stability
- Service availability
- …

What are the key pillars of the Container-based architecture?

- Container engines
- Container orchestrators

What are the key pillars of the Container-based architecture?

- Container engines
  - are based on OS where kernel allows multiple isolated app instances
  - are handling users inputs
  - are handling APIs inputs
  - are managing image management
    - image
    - Metadata prep
    - mount
    - container runtime exec
  - the most popular:
    - Docker
    - AWS Fargate
    - Google Kubernetes Engine
    - MS Azure

What are the key pillars of the Container-based architecture?

- Container orchestrators
  - empowering us with automation of much of the operational effort required to run containerized workloads and services
  - are enabling container deployment
  - are managing (scaling included) container lifecycle
  - are increasing system resilience
  - are handling security
  - the most popular:
    - Kubernetes
    - OpenShift

Practical implications of the containerization

- Benefits
  - Seamless and fast setup
  - Resource-efficient solution
  - Efficient utilization
  - Portability
  - Application level isolation
  - Easy to deploy
  - Reducing overall time2market

Practical implications of the containerization

- Challenges
  - IT resources (knowledge, availability)
  - Cloud connections?
  - Complexity
  - Security challenges?

When you should use it?

- Start with simple
- Nature of the endeavor (sensitive data, speed, scalability, security…)
- Investments
- Speed
- PoC
- Your environment (on-prem, cloud…)

Feedback from the experience and the practical use

THANK YOU!

Questions?