

# Containere

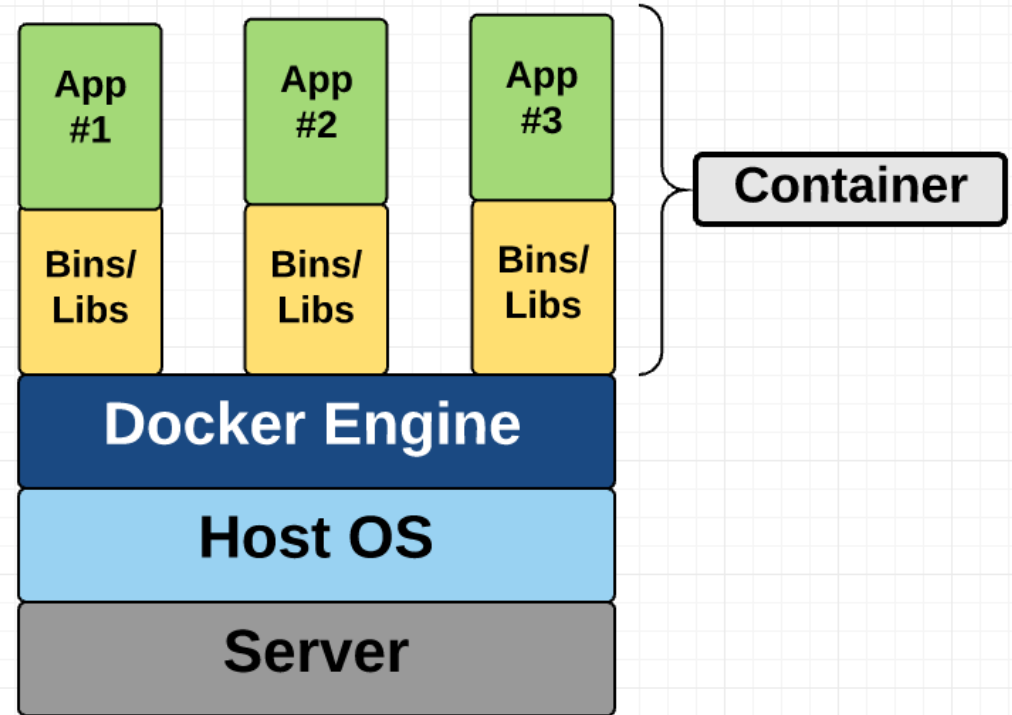
CONF.DR. CRISTIAN KEVORCHIAN  
UNIVERSITATEA DIN BUCURESTI  
FACULTATEA DE MATEMATICA ȘI INFORMATICĂ

# Virtualizarea Infrastructurii IT

- ▶ Virtualizarea platformelor și componentelor hardware este un proces de abstractizare logică necesară operaționalizării diverselor sisteme de operare.
- ▶ Platforma abstractă de calcul rezultată în urma virtualizării ascunde caracteristicile fizice ale platformei de calcul inițiale(fizice) prezentând-o în schimb drept o platformă abstractă de calcul care conservă modelul computational al celei inițiale

# Containere

- ▶ Containerele sunt cunoscute drept forme de virtualizare la nivelul sistemului de operare și reprezintă o abordare de complexitate redusă a virtualizării, care oferă un minim necesar de resurse de calcul unei aplicații pentru a funcționa corect într-un context computațional dat. Într-un fel, ele pot fi considerate mașini virtuale minimaliste care nu rulează pe un hypervisor.



Pe o singură  
mașină-  
gază pot  
rula mai  
multe  
containere.

- ▶ Containerele izolează, aplicațiile unele de altele pe un sistem de operare partajat.
- ▶ Aplicațiile containerizate rulează la nivelul superior de abstractizare a containere-lor, care, la rândul lor, rulează pe un sistem de operare (Linux sau Windows). Prin urmare, containerele au o prezență semnificativ mai mică decât imaginile unei mașinii virtuale.
- ▶ Fiecare container poate găzdui o aplicație web sau un serviciu.

# Containere-generalități

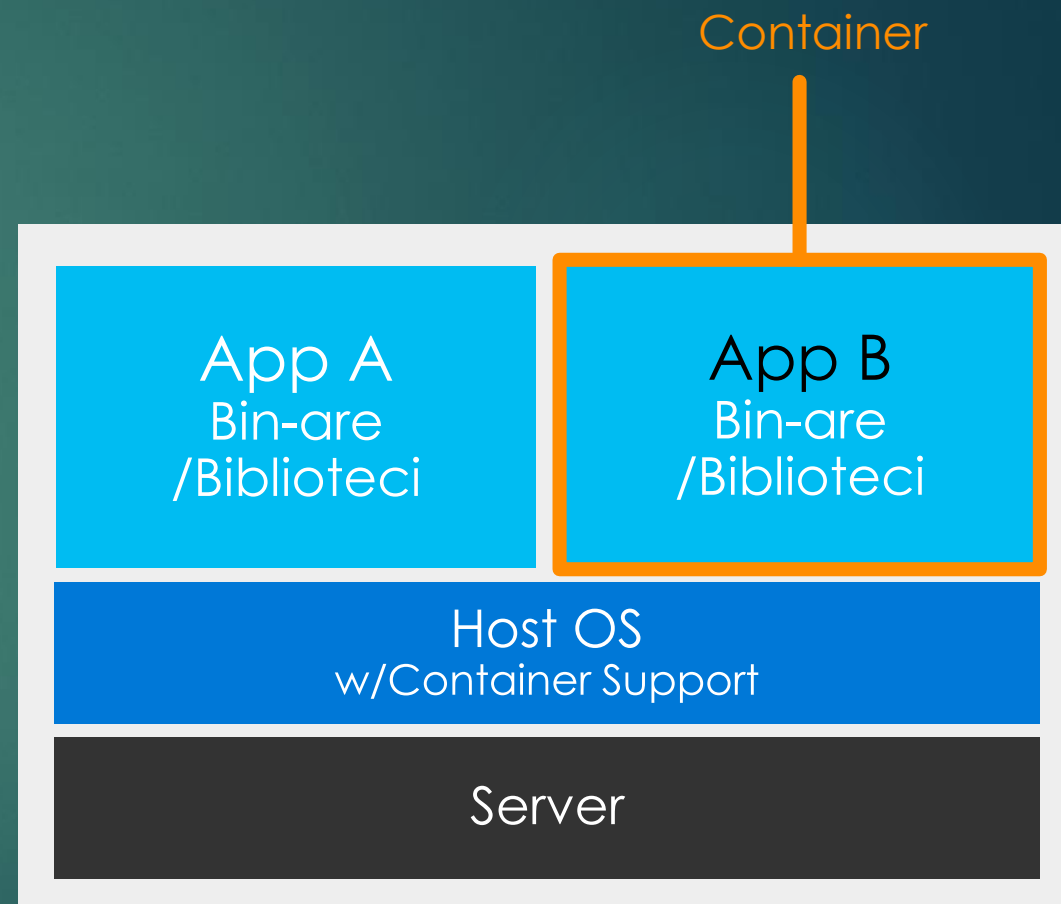
**Dependențe:** Fiecare aplicație are propriile dependențe care includ atât elemente software (servicii, biblioteci) cât și hardware (CPU, memorie, storage).

**Virtualizarea:** Motorul containerelor este un sistem de management și orchestrare al acestora, echivalent cu virtualizarea prin care se pot izola dependențele pentru fiecare aplicație prin "ambalarea" lor în containere virtuale.

**Partajarea SO gazda:** Procesele în containere sunt izolate de celelalte containere în spațiul utilizatorilor, dar partajează kernel-ul cu gazda și cu celelalte containere.

**Flexibilitate:** Diferențele dintre sistemul de operare și infrastructura de bază se abstracționează, simplificând abordarea prin "deployment oriunde" ..

**Rapid:** Containerele pot fi create aproape instant permițând o scalare elastică asociată cererii.



# Containere

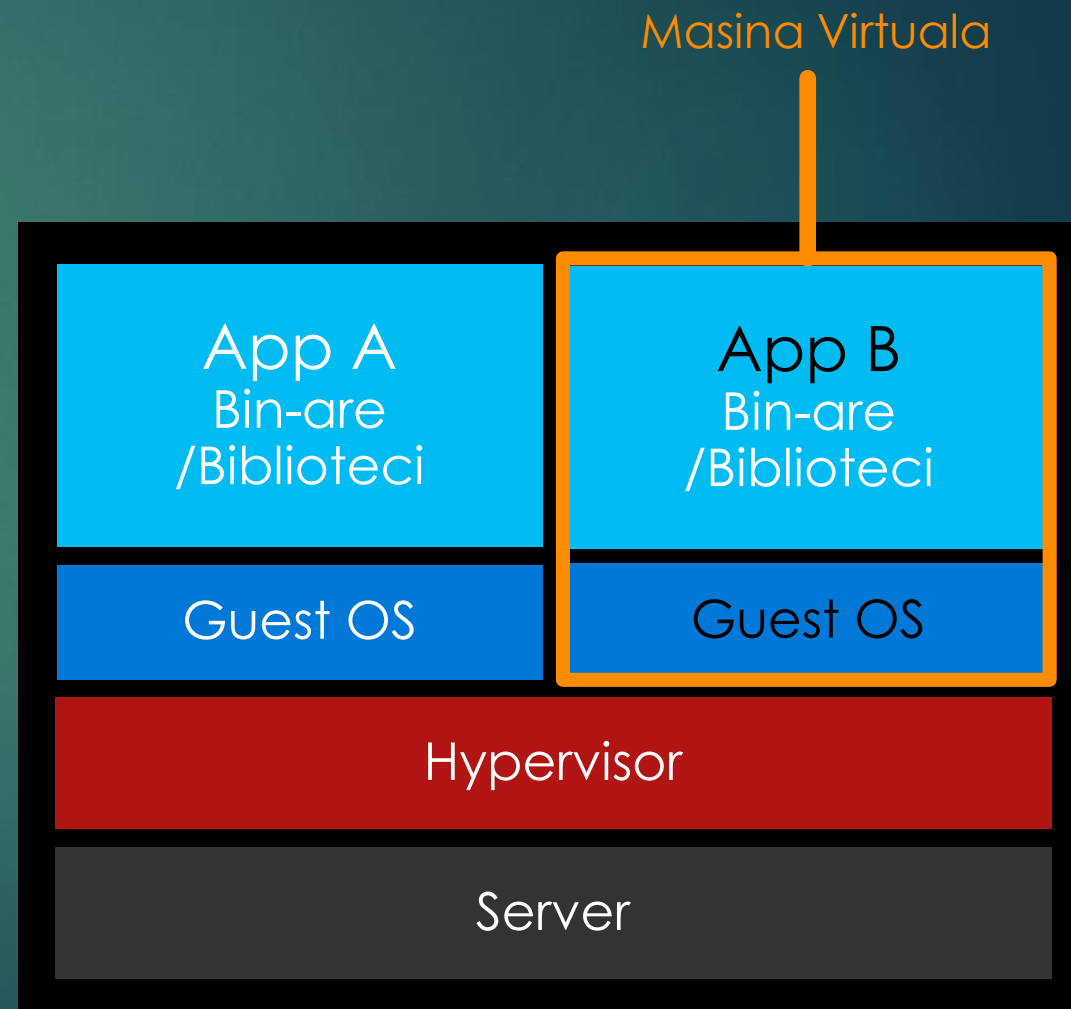
## Diferențe față de mașini virtuale

**Dependențele:** Fiecare app virtualizată include aplicația în sine, cerând binare și biblioteci dar și un SO "oaspete", care poate integra mai mulți GB de date.

**SO independent:** Fiecare MV poate avea un sistem de operare diferit de al altor MV, împreună cu un sistem de operare diferit de SO gazdă în sine.

**Flexibilitate:** MV pot fi migrate în alte locații pentru a balansa resursele utilizate și pentru mentenanța locației fara "downtime".

**Securitate:** Nivel înalt de izolare securizată a resurselor pentru sarcinile cheie virtualizate.



# Containere incluse în MV

## Scenarii multiple de implementare a aplicațiilor

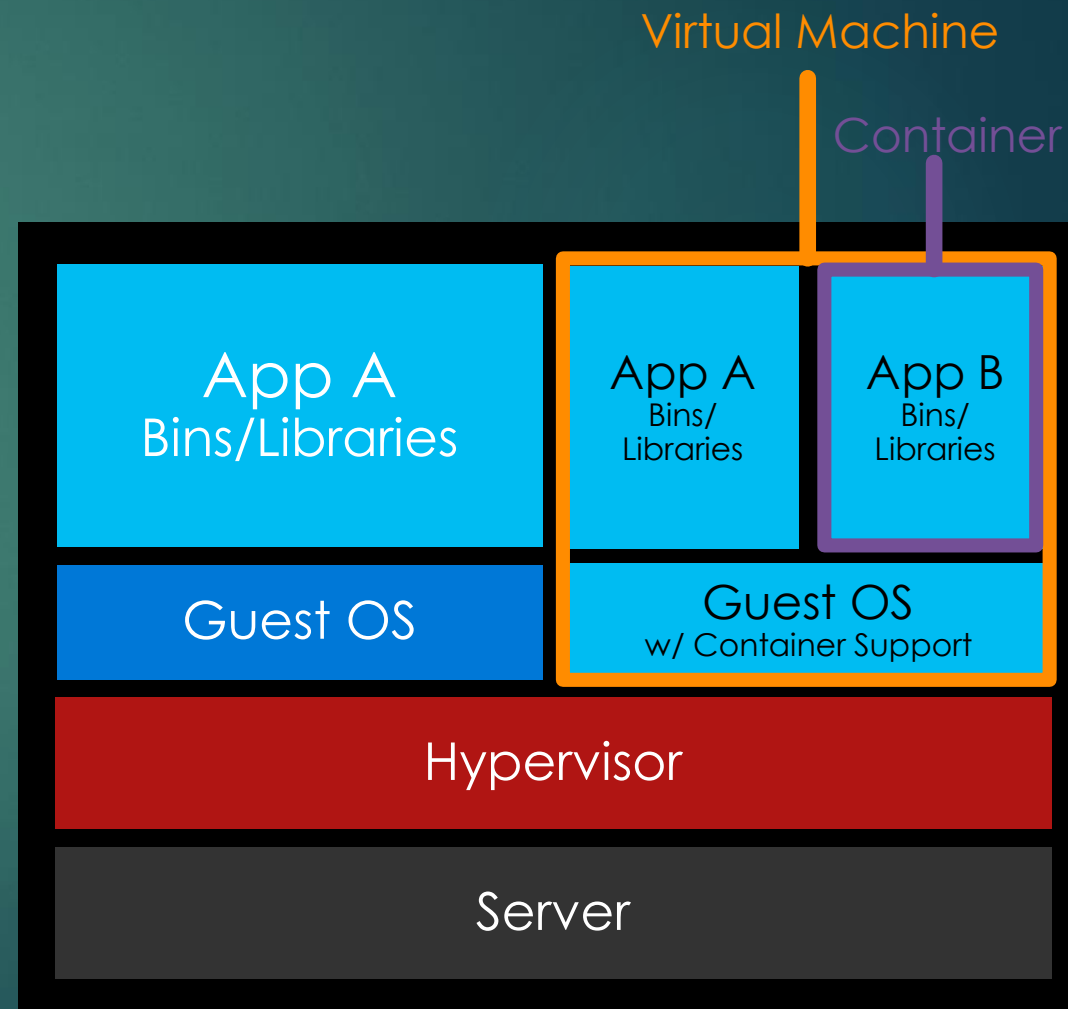
**Containere în MV:** Prin combinarea containerelor cu MV, utilizatorii pot implementa mai multe sisteme de operare în MV diferite, iar în interior pot implementa mai multe containere în cadrul acelor SO.

Prin combinarea containerelor cu MV, ar fi necesare mai puține MV pentru a suporta un număr mai mare de aplicații.

Mai puține MV ar avea ca rezultat o reducere a necesarului de stocare.

Fiecare MV ar susține mai multe aplicații izolate, crescând astfel densitatea globală a soluției.

**Flexibilitate:** Rularea containerelor în interiorul MV permite implementarea de soluții precum "live migration" pentru utilizarea optimă a resurselor și întreținerea gazdei.

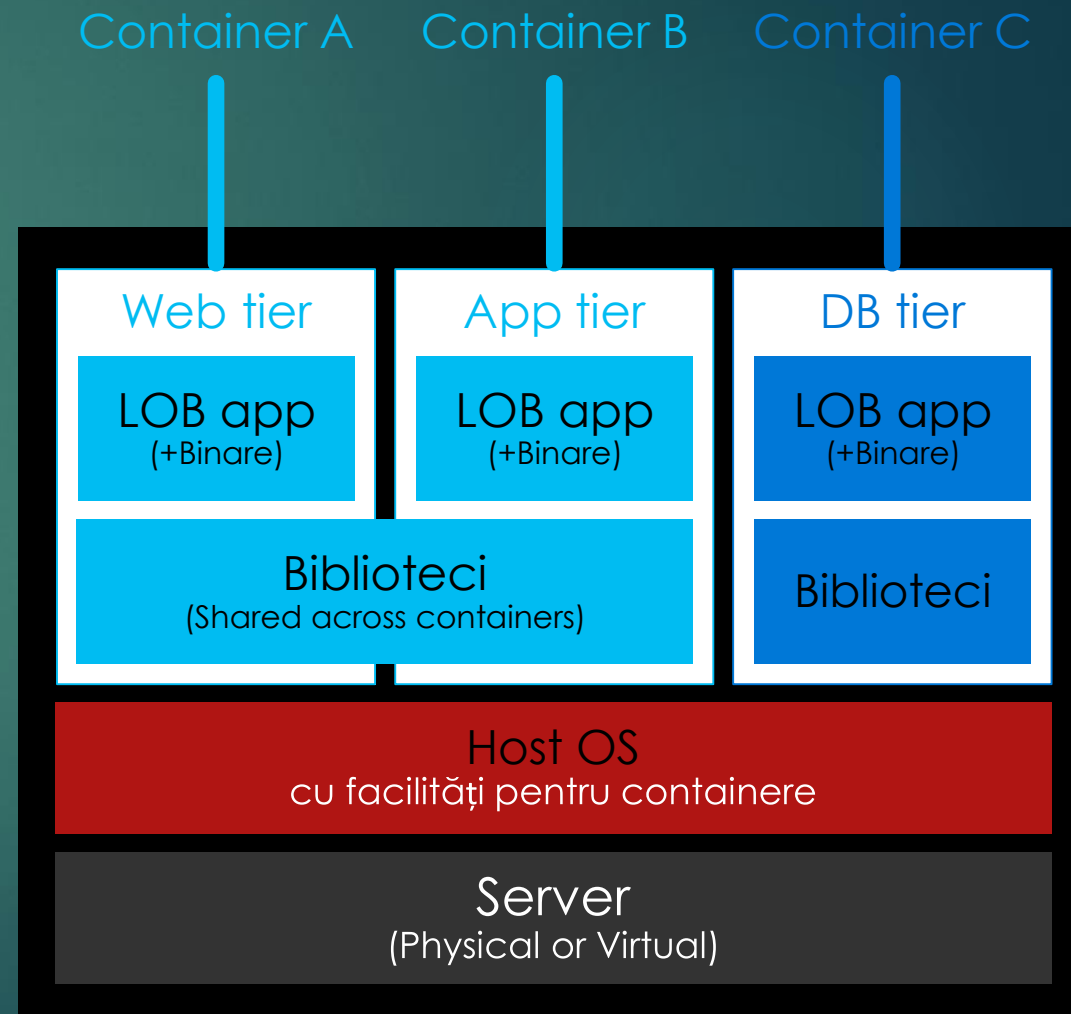




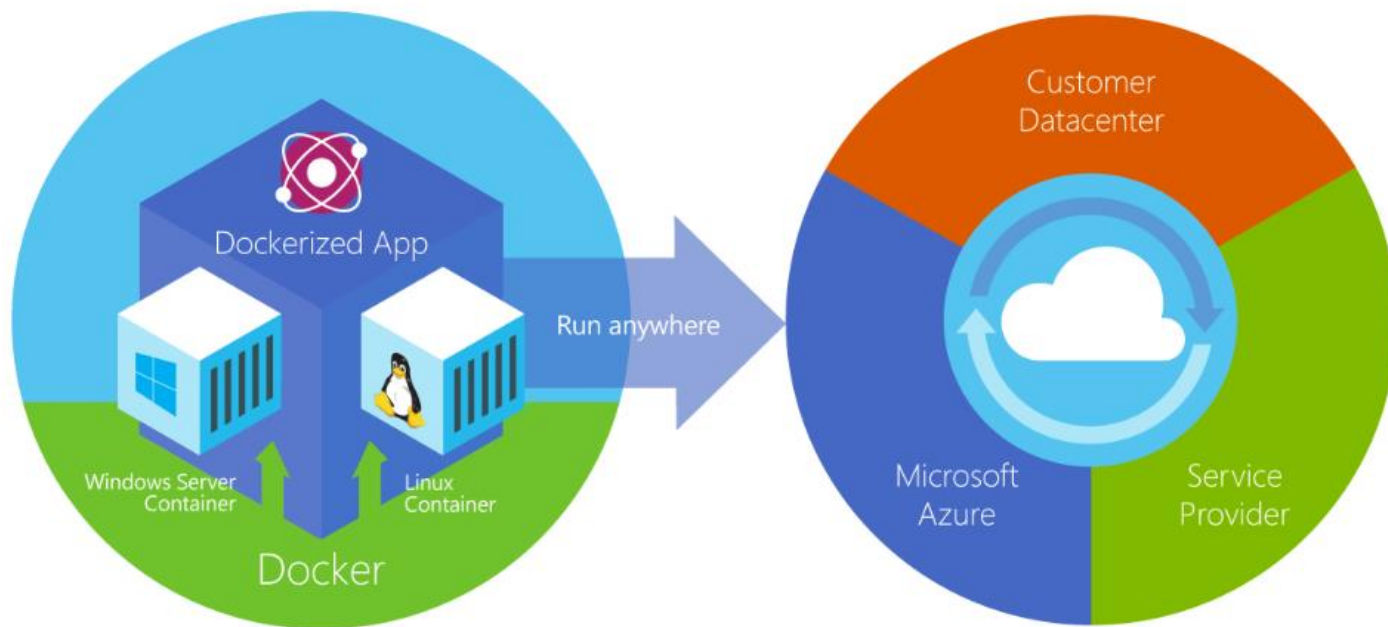
Prin construirea de aplicații modulare care folosesc containerele, modulele pot să fie actualizate independent – cu planuri independente.

**Management-ul:** Deployment-ul și managementul containerelor se realizează cu PowerShell sau Docker.

**Resursele:** Consumul de CPU și resursele de memorie pe container în funcție de capacitatea de stocare și banda transzitată.







Docker este un proiect open-source pentru automatizarea implementării aplicațiilor în containere portabile și care pot funcționa în cloud sau local. Docker este, de asemenea, o companie care promovează și dezvoltă această tehnologie, colaborând cu furnizori de cloud și SO.

# Terminologie Docker

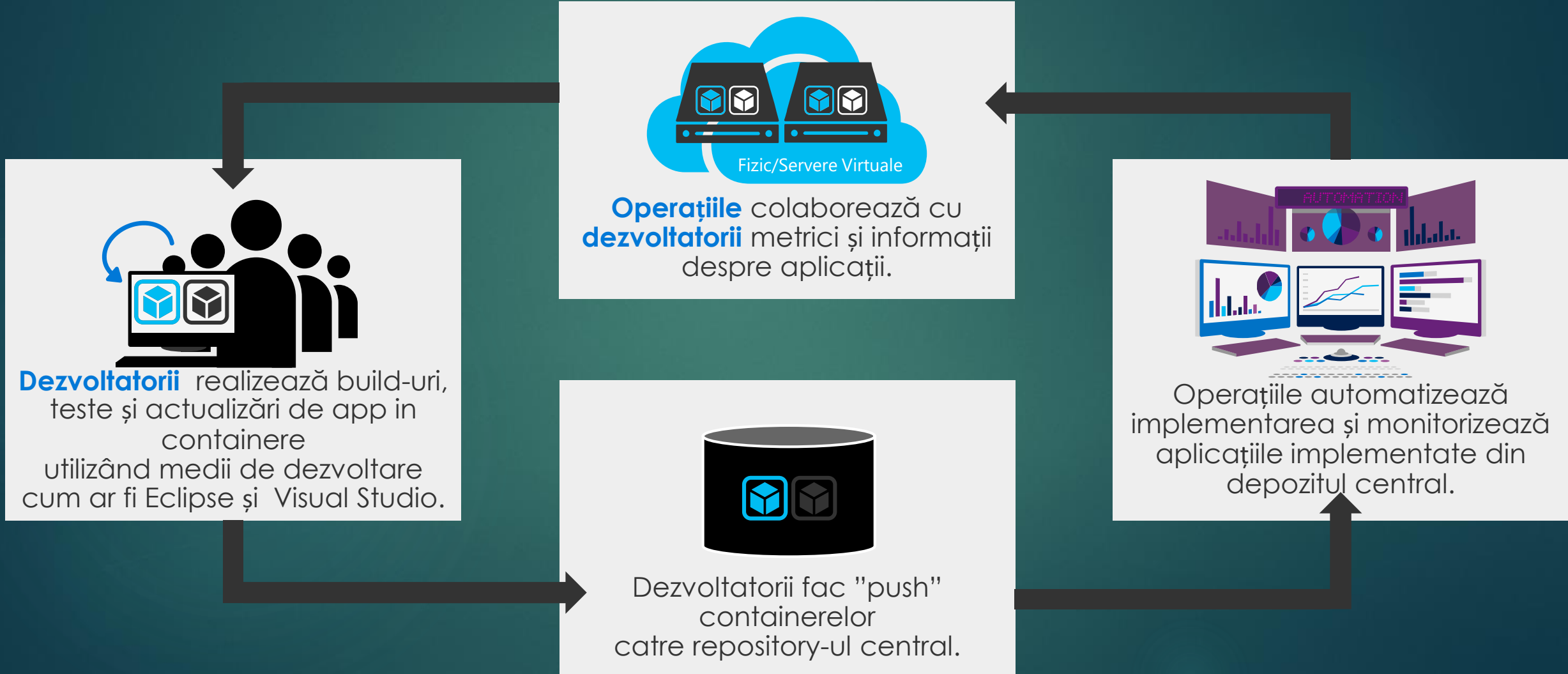
**Imaginea containerului:** Un pachet cu toate dependențele și informațiile necesare pentru a crea un container. O imagine include toate dependențele (cum ar fi frameworkurile) plus informații de configurare a mediului de execuție asociat containerului. O imagine derivă din mai multe imagini de bază care sunt elemente de formare a sistemului de gestiune a fișierelor. O imagine este imutabilă odată ce a fost creată.

**Container:** O instanță a unei imagini Docker. Un container este un mediu de runtime pentru o singură aplicație, proces sau serviciu. Când scalăm un serviciu sunt create o serie de instanțe ale unui container din aceeași imagine. Un job batch poate crea multiple containere din aceeași imagine prin transmiterea de parametri diferiți către fiecare instanță.

**DockerFile:** Un fișier text ce conține instrucțiuni pentru generarea unei imagini Docker.

**Build:** Construcția unei imagini bazate pe informațiile și contextul descris în DockerFile.

# Containerele sunt centrate pe procese DevOps



## Rezumând, putem afirma că:

Prin containerizarea aplicațiilor vechi utilizând varianta de containerizare Windows Server, obținem o consistență și un management îmbunătățit al echipelor de dezvoltatori și tester, pe de-o parte și cele de implementare, pe de altă parte – într-un mediu unitar DevOps - fără a modifica aplicația..

# Containere Hyper-V

## Structură și funcționalități cheie

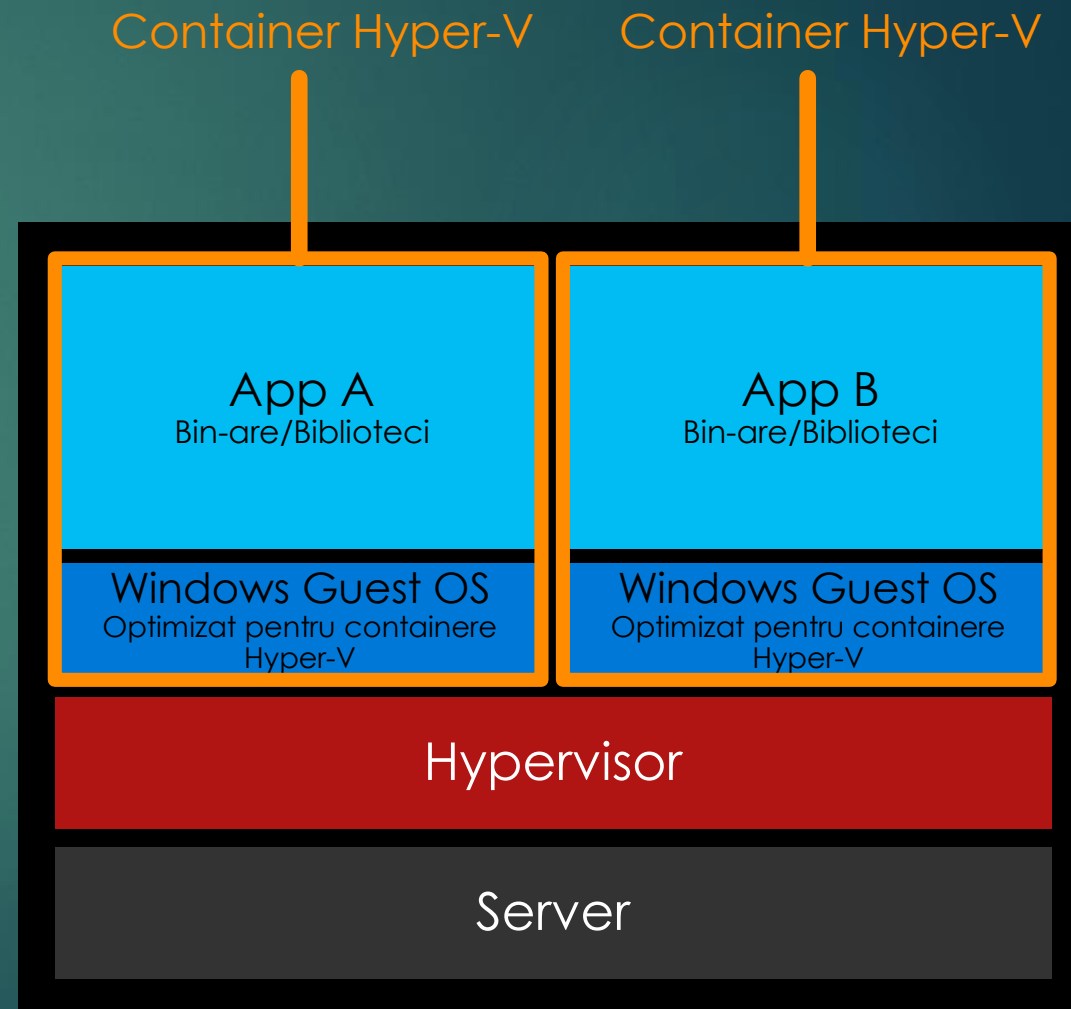
**Consistență:** Containerele Hyper-V utilizează aceleași API-uri ca și containerele Windows Server, asigurând coerența între seturile de instrumente de gestionare și implementare.

**Compatibility:** Containerele Hyper-V utilizează aceleași imagini ca și containerele Windows Server.

**Izolare Puternică:** Fiecare container Hyper-V are propria copie dedicată a kernel-ului.

**Foarte sigur:** Construit cu o tehnologie de virtualizare Hyper-V dovedită a fi foarte sigura.

**Optimizată:** Layer-ul de virtualizare și sistemul de operare au fost special optimizate pentru lucrul cu containere



# Componente Docker

## Build

Fluxuri de operatii la nivel "Dev"

Client Docker

Docker Compose

## Ship

Servicii Registry

Docker Hub

Docker Trusted Registry

## Run

Management

Docker Cloud

Docker Universal Control Plane

Docker Cloud

Docker Datacenter

Docker Engine

Sistem de Operare

Infrastructură

Plug-in-uri

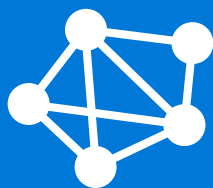


# Containerele reprezinta un excelent mediu pt.:

Calcul  
Distribuit

$f(x)$

Scalare



Baze de  
date



Task-uri



Web App





# SO pentru lucrul cu containere

## Nano Server

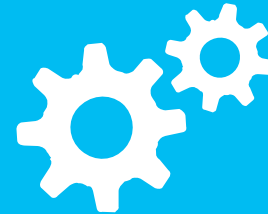


Bine  
optimizat



Aplicatii  
"Born-in-the-cloud"

## Server Core



Larg  
Compatib  
il



Aplicatii  
tradiționale



Windows Server

# Instrumente la dispoziție

## Medii de dezvoltare

 Visual Studio

 eclipse

Altele...

## Managementul containerelor



PowerShell



Docker



Others

## Tehnologii bazate pe containere



 Windows Server



Linux

## Microsoft cloud



Azure



On Prem



Service  
Provider

# Instrumente pentru Dev

## Limbaje și Framework-uri pentru dezvoltare

PHP

Go

Python

.Net

Node

Perl

Ruby

JavaScript

Win32

Java

C++

## Microsoft Cloud



Azure

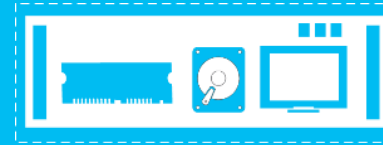


On Prem



Service Provider

## Masini Virtuale



Windows Server



Linux

## Service fabric



## Tehnologii Bazate pe Containerere



Windows Server



Linux

# Comparații între tehnologii

## Comparație între containere și VM

	Windows Server containere	Linux containere	Masini Virtuale
Imaginea de Bază	Acelasi cu gazda	Acelasi cu gazda	Oricare Windows/Linux
Securitate multi-tenant	Nu	Nu	Da
Managementul resurselor	Da	Da	Da
Densitate	Mare	Mare	Mica
Timpul de pornire	Scurt	Scurt	Lung
Amprenta pe disc	Mica	Mica	Mare
Compatibilitatea aplicațiilor	Medie	Medie	Mare

**Sistemul de operare:** Containerele partajeaza acelasi SO ca și sistemul gazda, dar poate rula in cadrul unei MV pentru a permite o flexibila utilizare a OS.

**Securitate:** Mașinile virtuale oferă un nivel mai ridicat de protecție împotriva amenințărilor, cum ar fi exploit-urile kernel-ului.

**Proprietăți MV :** În timp ce containerele au un timp de pornire mai rapid, mașinile virtuale beneficiază de caracteristici precum "live migration".

**Compatibilitatea Applic.:** Pentru maximizarea beneficiului, aplicațiile ar trebui să fie proiectate, arhitecturate și

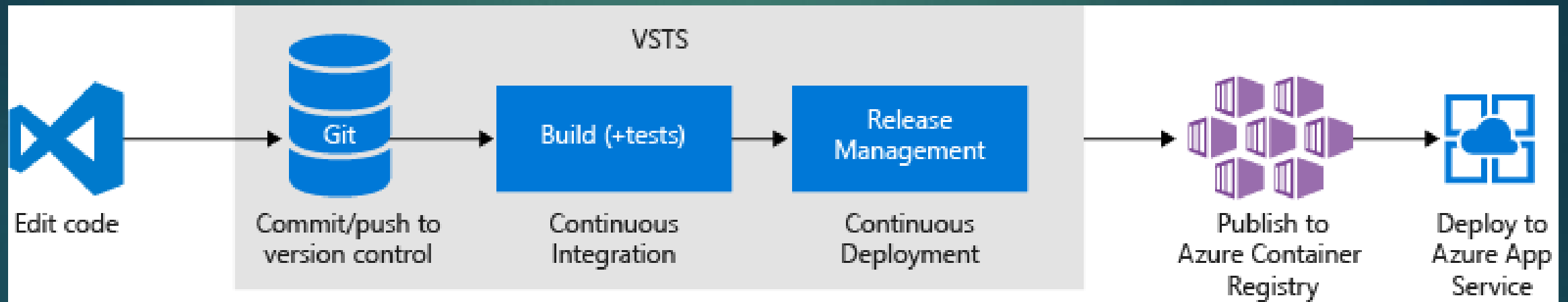
# Livrare și Integrare Continua

- **Livrarea continuă** (LC) este un concept al ingineriei software în care echipele produc software în cicluri scurte, asigurându-se că software-ul dezvoltat este fiabil și livrabil pentru utilizare în orice moment. Acest fapt implică scrierea, testarea și deployment-ul de software la perioade foarte scurte.
- **Integrarea continuă** (IC) - este o practică în dezvoltarea software care cere dezvoltatorilor să integreze codul (2-3 ori/zi) într-un repository comun. Fiecare check-in este apoi verificat de un build automat, permițând echipelor de dezvoltatori să detecteze într-o etapă incipientă eventualele probleme.

# Considerații pe marginea IC și LC

1. Livrarea continuă este o prelungire a integrării continue.
2. Este nevoie de o maturizare a proceselor ALM de așa manieră încât să puteți reacționa în timpul cel mai scurt să schimbări semnificative ale produsului software.
3. Acest lucru se traduce prin necesitatea automatizării proceselor așa încât să puteți implementa relesuri la perioade foarte scurte

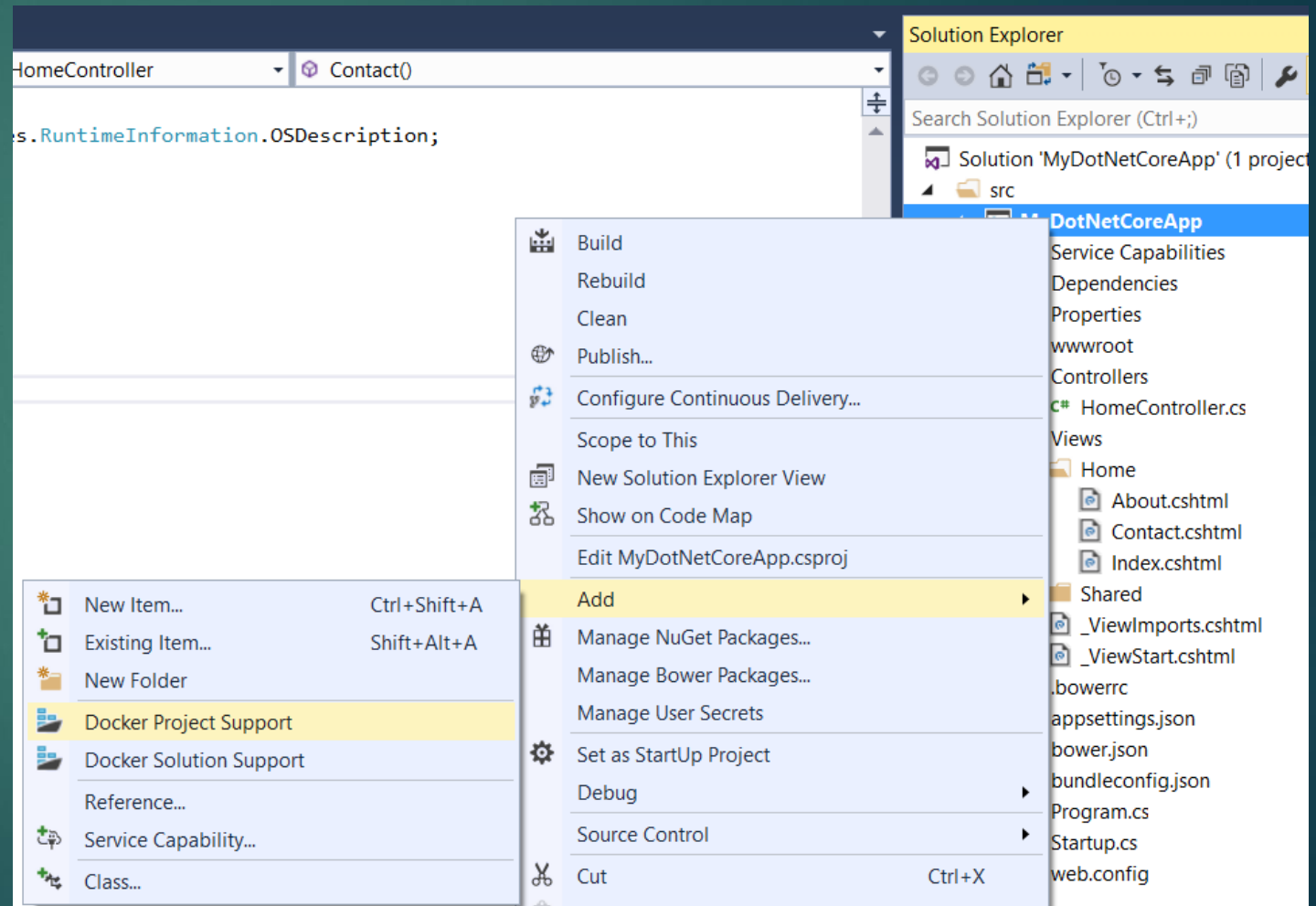
# ALM pentru IC și LC





# Configurare A Build-ului pentru LC

- container registry
- orchestrator (KUBERNETES)

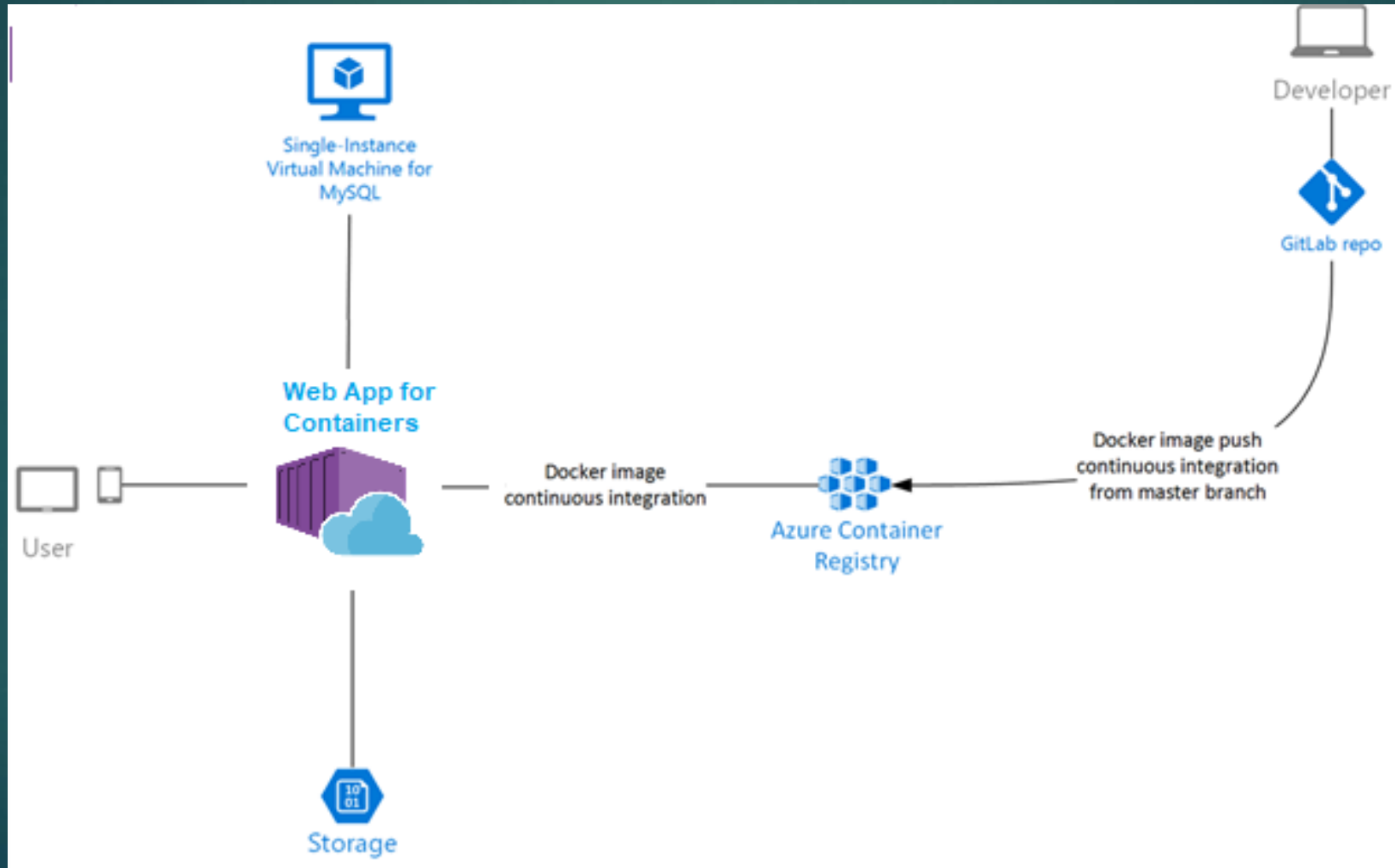


# DEVOPS

DevOps (dezvoltare și operațiuni) - este un concept în dezvoltare de software utilizat pentru a desemna un tip de relație agilă între dezvoltare și operațiunile IT ale unei entități de business.

Scopul DevOps este de a schimba și de a îmbunătăți relația prin promovarea unei mai bune comunicări și colaborări între aceste două componente importante care sprijină business-ul.

# Aplicații WEB Pentru containere



# Container registry

- ▶ "Registry" este o aplicație stateless server, foarte scalabilă, care stochează și distribuie "imagini Docker". Este open-source, sub licența deschisă Apache.
- Container Registry Privat
- Container Registry Public



# "AZURE Container Registry"

- ▶ **Azure Container Registry** permite să stocarea și gestiunea de imagini pentru toate tipurile de implementări de containere.
- ▶ **"Azure Container Instances"** permite implementarea de containere Docker pe infrastructura Azure fără a utiliza mașini virtuale sau pentru a adopta servicii de nivel superior.
- ▶ O **image Docker** este construită dintr-o serie de straturi. Fiecare strat reprezintă o instrucțiune a fișierului Docker. Fiecare strat, cu excepția ultimului, este numai pentru citire.
- ▶ Exemplu:

**FROM UBUNTU:15.04**

**COPY . / app**

**RUN make / app**

**CMD python /app/app.py**

# Containere


- ❖ O instanță a unei imagini este un container.
- ❖ O imagine, este formată dintr-o familie de starturi așa cum apar descrise în Docker file.
- ❖ Dacă pornim această imagine, avem un container care rulează această imagine.(mai multe containere pot rula asociat aceleiași imagini).
- ▶ O lista a imaginilor se poate obține cu "docker images", dar și containerele cu "docker ps -a"
- ▶ Deci, o instanță care rulează o imagine este un container.

# Configurare CD

## Configure Continuous Delivery

Create the Team Services and Azure resources needed to continuously deliver your app

User account:

 Peter.therian@contoso.com

Team project:

WebApplication1 (Peter.therian@contoso.com)

Repository:

WebApplication1 (master)

Azure subscription:

PETER THERIAN CONTOSO Azure Subscription

Container registry:

ContosoAcsRegistry

Azure container service:

ContosoWebAppContainer\_ACS

Enabling continuous delivery will:

Create a build definition for your repository

Create a release definition that runs on each successful build, and then delivers to the 'Dev' environment

Initiate a build-and-release run now, and whenever you push to this repository

[Review Azure Container Service pricing information](#)

[What is continuous delivery?](#)

This process may take several minutes to complete

OK

Cancel