Virtualizarea bazată pe Containere și ciclul de viață al aplicatiilor software și a modelelor IA

Conf. dr. Cristian KEVORCHIAN
Facultatea de Matematică și Informatica
Universitatea București

Definiție

Un container este o unitate standardizată(**O**pen **C**ontainer **I**nitiative) software care împachetează codul și toate dependențele necesare pentru ca o aplicație să ruleze rapid și fiabil în diverse medii de calcul

Spre deosebire de mașinile virtuale (VM), containerele partejează nucleul sistemului de operare al gazdei, dar rulează în spații izolate

Legatura cu sistemul de operare

Containerele utilizează caracteristici ale sistemului de operare pentru a asigura izolarea și gestionarea resurselor:

1.Spații de nume (Namespaces): Acestea creează vizualizări separate ale resurselor sistemului, asigurând că procesele dintr-un container nu interferează cu cele din alte containere

2.Grupuri de control (cgroups): Acestea gestionează și limitează utilizarea resurselor (CPU, memorie, I/O) de către fiecare container, asigurând o distribuție echitabilă între aceste

3.Structura Stratificata a sistemului de fișiere : Acestea permit suprapunerea mai multor straturi de sistem de fișiere, oferind o structură stratificată care ajută la eficientizarea utilizării resurselor

Virtualizarea bazata pe containere

- **Definiție**: Virtualizarea bazată pe containere permite rularea mai multor aplicații izolate pe același sistem de operare.
- **Avantaje**:
  - Utilizare eficientă a resurselor
  - Scalabilitate și flexibilitate
  - Izolare și securitate îmbunătățite
- **Componente Cheie**:
  - Containere
  - Imagini
  - Registri de imagini

Arhitectura bazată pe containere

Kernel Sharing: Containerele partajează același kernel al sistemului de operare gazdă.

Namespace: Asigură izolarea resurselor între containere.

Control Groups (cgroups): Limitează și prioritizează resursele pentru fiecare container.

Overlay Filesystem: Permite utilizarea eficientă a spațiului de stocare prin partajarea fișierelor comune.

Rolul Docker în Virtualizarea Bazată pe Containere

- **Docker** este o platformă open-source utilizată pentru dezvoltarea, gestionarea și rularea aplicațiilor containerizate, care permite dezvoltatorilor să împacheteze o aplicație împreună cu toate dependențele sale într-o imagine Docker. Această imagine, atunci când este executată, generează un container care asigură rularea aplicației într-un mod consistent în orice ecosistem software.
- **Docker Daemon**: Gestionează ciclul de viață al containerelor.
- **Docker CLI**: Interfață de linie de comandă pentru interacțiunea cu daemonul Docker.
- **Docker Hub:** Registru public pentru stocarea și distribuirea imaginilor Docker.
- **Docker Compose:** Instrument pentru definirea și rularea aplicațiilor multi-container.

Containere Linux vs Containere Windows

	Containere Linux	Containere Windows
Compatibilitate	Rulează pe gazde Linux și Windows (cu VM Linux)	Rulează doar pe gazde Windows
Performanță	De obicei mai rapide și mai eficiente datorită kernelului Linux	Performanță variabilă, depinde de integrarea cu Windows
Securitate	Suportă namespace-uri și cgroups pentru izolare și securitate	Utilizează SID-uri și ACL-uri pentru permisiuni și securitate
Instrumente	Suport extins pentru instrumente open-source și comunitate activă	Suport pentru instrumente Microsoft și integrare cu Active Directory
Utilizare	Ideal pentru aplicații open-source și medii de dezvoltare flexibile	Ideal pentru aplicații enterprise și medii Windows
Orchestrare	Suport complet pentru Kubernetes și alte platforme de orchestrare	Suport pentru Kubernetes, dar cu u nele limitări specifice Windows
Actualizări	Actualizări frecvente și suport pentru cele mai noi tehnologii	Actualizări mai rare, focus pe stabilitate și suport pe termen lung

PODMAN alternativa Red Hat la Docker

- **Podman** este un motor de containere open-source, dezvoltat de Red Hat, care permite dezvoltarea, gestionarea și rularea containerelor. Este similar cu Docker, dar are câteva caracteristici distincte.
- Caracteristici cheie ale Podman:
  - **Daemonless:** Podman nu necesită un proces de fundal persistent (daemon) pentru a rula containerele. Fiecare container rulează ca un proces independent, ceea ce îmbunătățește securitatea și stabilitatea.
  - **Rootless:** Podman permite rularea containerelor fără privilegii de root, ceea ce reduce riscurile de securitate.
  - **Compatibilitate OCI**: Podman respectă standardele Open Container Initiative (OCI), asigurând compatibilitatea cu alte instrumente și ecosisteme de containere.
  - **Poduri**: Podman poate gestiona grupuri de containere numite poduri, similar cu conceptul de poduri din Kubernetes. Aceste poduri permit rularea containerelor împreună și partajarea resurselor.
  - **Integrare cu alte instrumente**: Podman funcționează bine cu alte instrumente de containere, cum ar fi **Buildah** pentru construirea imaginilor și **Skopeo** pentru gestionarea imaginilor.

BUILDAH,manager de imagini

- **Definiție:** Buildah este un tool open-source utilizat pentru a crea imagini de containere compatibile cu Open Container Initiative (OCI) și Docker.
- **Caracteristici:**
  - **Fără daemon**: Nu necesită un daemon pentru a funcționa, ceea ce îmbunătățește securitatea și flexibilitatea.
  - **Flexibilitate**: Oferă o gamă largă de opțiuni de personalizare pentru construirea imaginilor, mai multe decât comanda podman build.
  - **Integrare:** Poate fi integrat cu alte instrumente și fluxuri de lucru pentru a crea imagini de containere în mod eficient.

SCOPEO

Definiție: Skopeo este un tool open-source utilizat pentru a inspecta și transfera imagini de containere între diferite registre de imagini.

Caracteristici:

- **Transfer de imagini**: Permite copierea imaginilor de containere între registre, inclusiv Docker Hub, registrul local și alte registre OCI.
- **Inspecție**: Poate inspecta imagini de containere fără a le descărca, oferind informații detaliate despre acestea.
- **Compatibilitate:** Funcționează cu imagini OCI și Docker, asigurând compatibilitatea cu diverse ecosisteme de containere.

Docker vs Podman

	Docker	Podman
Daemon-less	NU	DA
Root-less	NU	DA
Compatibility	OCI & Docker	OCI & Docker
Pod Management	NU	DA
Image Building	DA (Dockerfile)	DA (Buildah)
Image Transfer	DA	DA (Skopeo)

Containere în Fluxul de Dezvoltare

Dezvoltare

Containerele asigură un mediu uniform pentru toți dezvoltatorii, eliminând problemele legate de aplicarea principiului "funcționează pe calculatorul meu"

- Izolare a dependențelor
- Configurare rapidă
- Reproducibilitate by design

Testare

Testarea în containere asigură reproducibilitatea exactă a mediului de producție

- Integrare continuă eficientă
- Medii identice între etape
- Paralelizare simplificată

Producție

Implementare consistentă în orice mediu, de la cloud până la edge computing

- Scalare orizontală ușoară
- Izolare a resurselor
- Implementări fără întreruperi

Docker și Podman sunt instrumentele principale pentru dezvoltarea bazată pe containere, oferind comenzi intuitive pentru construirea, rularea și gestionarea containerelor. Aceste instrumente pot fi integrate direct în IDE-ul dumneavoastră preferat, simplificând fluxul de lucru zilnic.

Ciclul de Viață al Aplicațiilor Software: Agile + DevOp

	Activități Agile	Activități DevOps
Planificare	Definirea cerințelor, planificarea sprinturilor, prioritizarea funcționalităților	Utilizarea backlog-ului de produs și a sprinturilor pentru organizarea muncii
Dezvoltare	Scrierea codului, controlul versiunilor, colaborarea între echipe	Dezvoltare iterativă și incrementală, livrări frecvente de funcționalități
Integrare Continuă	Integrarea frecventă a codului, testarea automată a modificărilor	Automatizarea procesului de integrare pentru detectarea și corectarea rapidă a erorilor
Testare Continuă	Testarea automată a codului pentru identificarea și corectarea erorilor	Testare continuă în fiecare iterație pentru asigurarea calității
Desfășurare Continuă	Implementarea automată a aplicațiilor în medii de producție	Automatizarea procesului de implementare pentru reducerea timpului de lansare
Monitorizare Continuă	Monitorizarea performanței aplicațiilor, colectarea datelor, detectarea problemelor	Monitorizarea constantă pentru asigurarea stabilității și performanței aplicațiilor
Feedback Continuu	Colectarea feedback-ului de la utilizatori, ajustarea aplicațiilor	Adaptarea rapidă la schimbările cerințelor și feedback-ul utilizatorilor
Operațiuni Continue	Gestionarea și întreținerea aplicațiilor în producție, automatizarea proceselor de lansare	Automatizarea și optimizarea operațiunilor pentru asigurarea continuității serviciilor

Ciclul de Viață al Modelelor ML : Agile + MLOps

	Activități Agile	Activități MLOps
Planificare	Definirea cerințelor, planificarea sprinturilor, prioritizarea funcționalităților	Utilizarea backlog-ului de produs și a sprinturilor pentru organizarea muncii
Colectarea Datelor	Colectarea și pregătirea datelor necesare pentru antrenarea modelului	Automatizarea procesului de colectare și curățare a datelor
Dezvoltare Model	Dezvoltarea și antrenarea modelului ML	Utilizarea pipeline-urilor automate pentru antrenarea și validarea modelului
Integrare Continuă	Integrarea frecventă a codului, testarea automată a modificărilor	Automatizarea procesului de integrare pentru detectarea și corectarea rapidă a erorilor
Testare Continuă	Testarea automată a modelului pentru identificarea și corectarea erorilor	Testare continuă a performanței modelului pe seturi de date de validare
Desfășurare Continuă	Implementarea automată a modelului în medii de producție	Automatizarea procesului de implementare pentru reducerea timpului de lansare
Monitorizare Continuă	Monitorizarea performanței modelului, colectarea datelor, detectarea problemelor	Monitorizarea constantă pentru asigurarea stabilității și performanței modelului
Feedback Continuu	Colectarea feedback-ului de la utilizatori, ajustarea modelului	Adaptarea rapidă la schimbările cerințelor și feedback-ul utilizatorilor
Operațiuni Continue	Gestionarea și întreținerea modelului în producție, automatizarea proceselor de lansare	Automatizarea și optimizarea operațiunilor pentru asigurarea continuității serviciilor

DevOps vs MLOps

	DevOps	MLOps
Scop	Automatizarea și eficientizarea proceselor de dezvoltare și implementare a software-ului.	Automatizarea și eficientizarea ciclului de viață al modelelor de machine learning.
Portabilitate	Asigură rularea consistentă a aplicațiilor pe diverse medii.	Asigură rularea consistentă a modelelor IA pe diverse platforme.
Scalabilitate	Permite scalarea rapidă a aplicațiilor pentru a face față cerințelor variabile.	Permite scalarea rapidă a modelelor IA pentru a face față cerințelor de procesare.
Izolare	Fiecare container rulează într-un mediu izolat, prevenind interferențele între aplicații.	Fiecare container rulează într-un mediu izolat, prevenind interferențele între modele IA.
Automatizare	Facilitarea automatizării proceselor CI/CD pentru dezvoltarea și implementarea software-ului.	Facilitarea automatizării proceselor CI/CD pentru dezvoltarea și implementarea modelelor IA.
Reproducibilitate	Asigură reproducerea exactă a mediilor de dezvoltare și producție.	Asigură reproducerea exactă a experimentelor și rezultatelor modelelor IA.
Flexibilitate	Oferă flexibilitate în alegerea instrumentelor și infrastructurii.	Oferă flexibilitate în alegerea instrumentelor și infrastructurii pentru modele IA.

Containere în DevOps

Containerele joacă un rol esențial în metodologia DevOps, facilitând dezvoltarea, testarea și implementarea aplicațiilor într-un mod rapid și eficient. Iată câteva aspecte importante privind utilizarea containerelor în DevOps:

1. Portabilitate

Containerele permit rularea aplicațiilor în mod consistent pe diverse medii, de la dezvoltare la producție, indiferent de infrastructura subiacentă. Acest lucru elimină problemele legate de diferențele de mediu și asigură faptul că aplicațiile funcționează la fel peste tot.

2. Scalabilitate

Containerele sunt ușor de scalat pentru a face față variațiilor diveselor nivele de încărcare. DevOps poate automatiza procesul de scalare, adăugând sau eliminând containere în funcție de necesități, ceea ce oferă elasticitate și optimizare a resurselor.

3. Izolare

Fiecare container rulează într-un mediu izolat, ceea ce înseamnă că aplicațiile din diferite containere nu se pot afecta reciproc. Acest fapt conduce la o creștere a securității și stabilității aplicațiilor.

4. Automatizarea

Containerele facilitează automatizarea proceselor de build, testare și implementare. DevOps poate utiliza instrumente precum Docker și Kubernetes pentru a crea pipeline-uri CI/CD (Continuous Integration/Continuous Deployment) eficiente, reducând timpul de trecere în producție.

5. Microservicii

Containerele sunt ideale pentru arhitectura microserviciilor, unde aplicațiile sunt împărțite în servicii mici, independente, care pot fi dezvoltate, implementate și scalate separat2. Acest lucru îmbunătățește agilitatea și flexibilitatea echipelor de dezvoltare.

6. Consistență

Prin includerea tuturor dependențelor într-un container, DevOps asigură faptul că aplicațiile rulează în mod consistent în orice mediu, eliminând problemele de tip "funcționează pe mașina mea".

Aceste aspecte fac din containere o componentă esențială în metodologia DevOps, contribuind la eficiența și fiabilitatea proceselor de dezvoltare și implementare a software-ului.

Containere în MLOps

Containerele joacă un rol crucial în metodologia MLOps (Machine Learning Operations), facilitând dezvoltarea, implementarea și gestionarea modelelor de inteligență artificială (IA). Iată câteva aspecte importante privind utilizarea containerelor în MLOps:

1. Portabilitate și Consistență

Containerele asigură că mediile de dezvoltare și producție sunt identice, eliminând problemele legate de diferențele de configurare. Acest lucru permite rularea consistentă a modelelor IA pe diverse platforme și infrastructuri.

2. Scalabilitate

Containerele permit scalarea rapidă a modelelor IA pentru a face față cerințelor variabile de procesare. Utilizând orchestratori precum Kubernetes, echipele MLOps pot gestiona eficient resursele și pot scala automat modelele în funcție de necesități.

3. Izolare și Securitate

Fiecare container rulează într-un mediu izolat, ceea ce îmbunătățește securitatea și previne interferențele între diferite modele sau aplicații. Aceasta este esențială pentru protejarea datelor sensibile și pentru asigurarea integrității modelelor IA.

4. Automatizare și CI/CD

Containerele facilitează automatizarea proceselor de integrare continuă (CI) și livrare continuă (CD). Pipeline-urile CI/CD pot construi, testa și implementa automat modele IA în containere, reducând timpul de lansare pe piață și îmbunătățind eficiența echipelor2.

5. Reproducibilitate

Utilizarea containerelor permite reproducerea exactă a experimentelor și a rezultatelor. Acest lucru este esențial pentru validarea și auditarea modelelor IA, asigurând că aceleași rezultate pot fi obținute în mod consistent1.

6. Flexibilitate

Containerele oferă flexibilitate în alegerea instrumentelor și a infrastructurii. Modelele IA pot fi dezvoltate pe o platformă și implementate pe alta, fără a fi necesare modificări semnificative.

Aceste aspecte fac din containere o componentă esențială în metodologia MLOps, contribuind la eficiența și fiabilitatea proceselor de dezvoltare și implementare a modelelor IA.

Containerizare DevOps vs MLOps

	DevOps	MLOps
Scop	Automatizarea și eficientizarea proceselor de dezvoltare și implementare a software-ului.	Automatizarea și eficientizarea ciclului de viață al modelelor de machine learning.
Portabilitate	Asigură rularea consistentă a aplicațiilor pe diverse medii.	Asigură rularea consistentă a modelelor IA pe diverse platforme.
Scalabilitate	Permite scalarea rapidă a aplicațiilor pentru a face față cerințelor variabile.	Permite scalarea rapidă a modelelor IA pentru a face față cerințelor de procesare.
Izolare	Fiecare container rulează într-un mediu izolat, prevenind interferențele între aplicații.	Fiecare container rulează într-un mediu izolat, prevenind interferențele între modele IA.
Automatizare	Facilitarea automatizării proceselor CI/CD pentru dezvoltarea și implementarea softwareului.	Facilitarea automatizării proceselor CI/CD pentru dezvoltarea și implementarea modelelor IA.
Reproducibilitate	Asigură reproducerea exactă a mediilor de dezvoltare și producție.	Asigură reproducerea exactă a experimentelor și rezultatelor modelelor IA.
Flexibilitate	Oferă flexibilitate în alegerea instrumentelor și infrastructurii.	Oferă flexibilitate în alegerea instrumentelor și infrastructurii pentru modele IA.

Generative Artificial Intelligence Operations

GenAIOps este o ramură specializată a operațiunilor IA care se concentrează pe **gestionarea ciclului de viață al sistemelor de inteligență artificială generativă** (ex: modele precum GPT, DALL-E, sau Gemini). Scopul său principal este de a automatiza, monitoriza și optimiza procesele de dezvoltare, implementare, întreținere și scalare a acestor modele generative, asigurând performanță, securitate și conformitate etică.

Elemente cheie ale GenAIOps

Generare de conținut: Gestionarea modelelor care generează text, imagini, cod sau conținut audio.

Prompt Engineering: Optimizarea interacțiunilor cu modelele prin ajustarea prompturilor.

Controlul calității: Filtrarea conținutului generat pentru a evita răspunsuri toxice, ”biased” sau inexacte.

Monitorizare și etică: Urmărirea impactului social și a riscurilor (ex: dezinformare, abuz).

Integrare: Conectarea modelelor generative în fluxuri de lucru sau aplicații existente (ex: chatbots, automate de creare de conținut).

MLOps vs GenAIOps

	MLOps	GenAIOps
Scop principal	Automatizarea și operationalizarea modelelor de Machine Learning (ML).	Operationalizarea sistemelor de AI generativă (ex: GPT, DALL-E) și integrarea lor în fluxuri de lucru.
Focus	ML clasic (ex: modele de predicție, clasificare).	Sisteme de AI generativă, creare de conținut, interacțiuni naturale.
Componente cheie	CI/CD pentru ML, monitorizare modele, versionare date și modele.	Generare de conținut, prompt engineering, gestionarea riscurilor etice (ex: bias în texte generate).
Date utilizate	Seturi de date structurate (tabele, imagini, texte pentru antrenare).	Date nestructurate (texte, imagini, audio) și modele pre-antrenate pe corpusuri masive.
Provocări unice	Drift de date, scalabilitate, reantrenare modele.	Controlul calității conținutului generat, prevenirea abuzului, gestionarea prompturilor.
Instrumente comune	Kubeflow, MLflow, TensorFlow Extended (TFX).	OpenAI API, LangChain, Hugging Face Transformers, Guardrails AI.
Exemple de aplicații	Sisteme de recomandă, fraud detection.	Chatbots, generare de texte/imagini, asistenți virtuali.