MINISTERUL EDUCAȚIEI
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Ingineria Sistemelor

TEZĂ DE DOCTORAT

– REZUMAT –

Managementul Microserviciilor Utilizate pentru Creșterea Dependabilității și Scalabilității Sistemelor

Student-doctorand:
Ionuț-Cătălin DONCA

Conducător științific:
Prof. Dr. Ing. Liviu-Cristian Miclea

Comisia de evaluare a tezei de doctorat:

Președinte: Prof. Dr. Ing. **Silviu Folea** - Universitatea Tehnică din Cluj-Napoca;
Conducător științific: Prof. Dr. Ing. **Liviu-Cristian Miclea** - Universitatea Tehnică din Cluj-Napoca;
Referenți:
- Prof. Dr. Ing. **Ioan Silea** - Universitatea Politehnică Timișoara;
- Prof. Dr. Ing. **Lucian Mihai Itu** - Universitatea Transilvania din Brașov;
- Prof. Dr. Ing. **Mihail Abrudean** - Universitatea Tehnică din Cluj-Napoca.

– Cluj-Napoca –
2024

str. Memorandumului nr. 28, 400114 Cluj-Napoca, România
tel. +40-264-401200, fax +40-264-592055, secretariat tel. +40-264-202209, fax +40-264-202280
www.utcluj.ro

1. Introducere

În era digitală contemporană, sistemele informatice au devenit din ce în ce mai complexe și distribuite, necesitând soluții avansate pentru a asigura atât performanța, cât și fiabilitatea lor. Creșterea exponențială a volumului de date, a numărului de utilizatori și a cerințelor de performanță a condus la necesitatea dezvoltării unor arhitecturi software care să poată susține aceste solicitări în mod eficient, dependabil și scalabil.

Arhitectura bazată pe microservicii a apărut ca o soluție modernă și flexibilă pentru a aborda provocările legate de dezvoltarea și gestionarea aplicațiilor complexe. Această abordare implică decompoziția aplicațiilor monolitice în servicii mai mici, independente și ușor de gestionat, care comunică între ele prin intermediul unor interfețe bine definite. Microserviciile permit echipelor de dezvoltare să itereze rapid, să implementeze și să scaleze componente individuale fără a afecta întregul sistem, îmbunătățind astfel agilitatea și timpul de răspuns la cerințele pieței.

Cu toate acestea, adoptarea arhitecturilor de microservicii aduce cu sine și o serie de provocări semnificative în ceea ce privește managementul eficient, asigurarea dependabilității și scalabilității sistemelor. Gestionarea numeroaselor servicii distribuite, asigurarea comunicării fiabile între acestea, monitorizarea performantă și menținerea securității reprezintă aspecte critice care necesită abordări și instrumente specializate.

Prezenta teză își propune să investigheze și să propună soluții eficiente pentru managementul microserviciilor în scopul creșterii dependabilității, scalabilității și securității sistemelor informatice. Prin analiza aprofundată a metodologiilor existente și dezvoltarea unor noi strategii și instrumente, cercetarea urmărește să contribuie la îmbunătățirea practicilor actuale și să faciliteze adoptarea cu succes a arhitecturilor bazate pe microservicii în diverse domenii.

1.1 Prezentarea domeniului tezei de doctorat

Domeniul de cercetare al acestei teze se concentrează pe managementul microserviciilor ca metodă principală pentru îmbunătățirea dependabilității și scalabilității sistemelor informatice moderne. În contextul transformării digitale și al necesității de a livra servicii și aplicații cu performanțe ridicate și disponibilitate continuă, arhitectura microserviciilor s-a impus ca o paradigmă esențială în dezvoltarea software.

Microserviciile reprezintă un stil arhitectural ce permite dezvoltarea de aplicații modulare, unde fiecare componentă funcționează ca un serviciu independent. Fiecare microserviciu este responsabil pentru o funcționalitate specifică a aplicației și poate fi dezvoltat, implementat și scalat independent de celelalte servicii. Această modularitate oferă numeroase avantaje, printre care:

- Flexibilitate crescută: Permite actualizarea și îmbunătățirea rapidă a componentelor individuale fără a afecta întregul sistem.
- Scalabilitate îmbunătățită: Fiecare microserviciu poate fi scalat independent în funcție de cerințele specifice de performanță și încărcare, optimizând utilizarea resurselor.
- Reziliență sporită: Defectarea unui microserviciu nu conduce neapărat la colapsul întregului sistem, permițând izolarea și remedierea rapidă a problemelor.

- Dezvoltare agilă: Echipele pot lucra în paralel pe diferite microservicii, accelerând ciclul de dezvoltare și livrare a funcționalităților noi.

Cu toate acestea, gestionarea eficientă a microserviciilor implică abordarea unor provocări complexe legate de orchestrare, comunicare, monitorizare și securitate. Pentru a răspunde acestor provocări, sunt utilizate diverse tehnologii și instrumente specializate, printre care:

- Containerele și orchestrarea containerelor: Tehnologii precum Docker permit împachetarea microserviciilor într-un mod portabil și consistent, în timp ce sisteme de orchestrare precum Kubernetes facilitează implementarea, scalarea și gestionarea automată a containerelor în medii distribuite.
- Sisteme de mesagerie: Instrumente precum RabbitMQ sau Apache Kafka asigură o comunicare asincronă și fiabilă între microservicii, permițând decuplarea componentelor și gestionarea eficientă a fluxurilor de date.
- Monitorizare și observabilitate: Soluții precum Prometheus, Grafana și ELK oferă vizibilitate în timp real asupra performanței și sănătății microserviciilor, facilitând detectarea și remedierea rapidă a problemelor.
- Practici DevOps și CI/CD: Integrarea continuă și livrarea continuă permit automatizarea proceselor de dezvoltare, testare și implementare a microserviciilor, asigurând un ciclu de dezvoltare rapid și fiabil.
- Securitatea sistemelor: Autentificarea și autorizarea utilizatorilor și a aplicațiilor crește nivelul securității sistemelor împotriva vulnerabilităților și amenințărilor tot mai dese, iar gestionarea secretelor ajută la minimizarea expunerii datelor sensibile.

Dependabilitatea sistemelor se referă la capacitatea acestora de a funcționa corect și de a furniza serviciile așteptate în mod consecvent, chiar și în condiții de stres sau în cazul apariției unor erori. În contextul microserviciilor, asigurarea dependabilității implică:

- Implementarea de mecanisme de toleranță la erori: Inclusiv restaurări automate și mecanisme de întrerupere care mențin funcționalitatea sistemului chiar și în cazul eșecului unor componente.
- Automatizarea proceselor de recuperare: Utilizarea orchestratorilor pentru a reporni sau înlocui automat microserviciile defecte, minimizând timpul de nefuncționare.
- Testarea robustă: Incluzând teste de reziliență pentru a evalua și îmbunătăți capacitatea sistemului de a face față perturbărilor.

Scalabilitatea sistemelor este capacitatea acestora de a gestiona creșteri ale cerințelor de resurse și de performanță fără a compromite calitatea serviciilor oferite. În arhitectura microserviciilor, scalabilitatea se realizează prin:

- Scalare orizontală: Adăugarea de instanțe suplimentare ale microserviciilor pentru a gestiona încărcări mai mari, facilitată de orchestratori precum Kubernetes care pot ajusta automat numărul de instanțe în funcție de necesitate.
- Cache eficient: Implementarea de mecanisme de cache pentru a reduce latența și a îmbunătăți performanța aplicațiilor.
- Balansare traficului: Distribuirea uniformă a solicitărilor între diferite instanțe ale microserviciilor pentru a optimiza utilizarea resurselor și a preveni supraîncărcarea.

Această teză își propune să exploreze în profunzime metodele și practicile optime pentru managementul microserviciilor, concentrându-se pe:

- Analiza și compararea diferitelor tehnologii și instrumente disponibile pentru orchestrare, comunicare și monitorizare, identificând cele mai potrivite soluții în diferite contexte aplicative.

- Dezvoltarea și evaluarea unor strategii de implementare și gestionare care maximizează dependabilitatea și scalabilitatea, minimizând în același timp complexitatea operațională și costurile asociate.
- Investigarea aspectelor de securitate și conformitate în contextul arhitecturilor de microservicii, propunând mecanisme eficiente de protecție și gestionare a riscurilor.
- Prezentarea unor studii de caz și experimentări practice care demonstrează aplicabilitatea și eficacitatea soluțiilor propuse în scenarii reale.

Prin abordarea acestor aspecte critice, cercetarea contribuie la consolidarea cunoștințelor și a practicilor din domeniul managementului microserviciilor, oferind ghiduri și recomandări valoroase pentru profesioniștii din industrie și comunitatea academică implicați în dezvoltarea și gestionarea sistemelor informatice moderne.

1.2 Motivație

În contextul evoluției tehnologice și al cerințelor din ce în ce mai stringente de performanță și fiabilitate, adoptarea arhitecturilor de microservicii a devenit o necesitate pentru multe organizații care își doresc să mențină un avantaj competitiv. Sistemele tradiționale monolitice, deși încă utilizate pe scară largă, prezintă limitări semnificative în ceea ce privește scalabilitatea și adaptabilitatea la schimbările rapide din mediul de afaceri.

Motivația principală pentru această cercetare derivă din nevoia de a adresa aceste limitări prin implementarea și gestionarea eficientă a microserviciilor, care oferă posibilitatea dezvoltării unor aplicații modulare, scalabile și rezistente la erori. În plus, în contextul digitalizării accelerate, există o presiune continuă asupra sistemelor informatice de a oferi disponibilitate continuă, performanță ridicată și securitate sporită.

Tehnologiile precum Kubernetes, pentru orchestrarea containerelor, și RabbitMQ, pentru asigurarea unei comunicări eficiente între microservicii, joacă un rol esențial în realizarea acestor obiective. Ele permit gestionarea resurselor în mod dinamic și optim, asigurând că fiecare componentă a sistemului poate fi actualizată sau scalată fără a afecta disponibilitatea generală a aplicației.

Un alt factor motivațional este necesitatea de a dezvolta metodologii de integrare și livrare continuă (CI/CD) eficiente, care să faciliteze adoptarea rapidă a modificărilor și îmbunătățirilor sistemului, fără a compromite stabilitatea acestuia. Aceste metodologii sunt esențiale pentru a menține un ritm rapid de dezvoltare și pentru a răspunde prompt la cerințele pieței.

În final, securitatea rămâne un aspect critic în gestionarea microserviciilor, având în vedere numărul mare de puncte de acces și potențiale vulnerabilități introduse de această arhitectură. Teza explorează soluții avansate pentru protejarea comunicațiilor între microservicii și pentru gestionarea sigură a datelor sensibile, asigurând astfel integritatea și confidențialitatea sistemului.

Într-o lume digitală în continuă evoluție, unde timpul de nefuncționare poate duce la pierderi semnificative de venit și la afectarea reputației unei organizații, asigurarea unui nivel ridicat de dependabilitate și scalabilitate devine esențială. Prin urmare, această teză își propune să contribuie la dezvoltarea de soluții care să permită gestionarea eficientă a microserviciilor într-un mod care să răspundă cerințelor actuale ale pieței.

Astfel, motivația principală a acestei cercetări constă în necesitatea de a furniza un cadru robust pentru implementarea și managementul microserviciilor, care să asigure nu doar

performanţe tehnice superioare, ci şi o aliniere strategică cu obiectivele organizaţionale pe termen lung.

1.3 Scopul tezei de doctorat

Scopul principal al acestei teze de doctorat este de a dezvolta şi valida soluţii inovative pentru managementul microserviciilor care sunt utilizate pentru a îmbunătăţi dependabilitatea şi scalabilitatea sistemelor informatice distribuite. Într-un peisaj tehnologic în continuă evoluţie, unde cerinţele de performanţă şi disponibilitate sunt din ce în ce mai stringente, teza îşi propune să adreseze provocările majore asociate cu implementarea şi gestionarea eficientă a microserviciilor.

Prin acest studiu, se urmăreşte crearea unui cadru robust care să permită implementarea unor arhitecturi de microservicii ce pot răspunde rapid şi eficient la schimbările din mediul de operare, menţinând în acelaşi timp un nivel ridicat de fiabilitate şi performanţă.

Obiectivele specifice ale tezei includ:

- Analiza şi evaluarea metodologiilor existente pentru managementul microserviciilor, cu accent pe identificarea limitărilor şi a oportunităţilor de îmbunătăţire.
- Dezvoltarea unor strategii avansate de orchestrare şi monitorizare a microserviciilor, care să asigure o gestionare eficientă a resurselor şi o scalabilitate adecvată în funcţie de nevoile aplicaţiei.
- Proiectarea şi implementarea unor soluţii pentru creşterea rezilienţei şi a toleranţei la erori în arhitecturile de microservicii, incluzând mecanisme automate de recuperare şi restaurare.
- Evaluarea şi optimizarea performanţei sistemelor de microservicii prin utilizarea unor tehnologii moderne, cum ar fi containerele, platformele de orchestrare şi integrarea şi livrarea continuă, cu scopul de a maximiza eficienţa şi a minimiza costurile operaţionale.
- Investigarea şi integrarea măsurilor de securitate în arhitecturile de microservicii, pentru a proteja integritatea şi confidenţialitatea datelor în faţa atacurilor cibernetice şi a altor vulnerabilităţi.

Aceste obiective sunt menite să contribuie la dezvoltarea unor practici şi instrumente care să faciliteze adoptarea microserviciilor în diverse scenarii industriale, oferind totodată un ghid pentru dezvoltatorii din domeniul IT care doresc să implementeze aceste soluţii în medii complexe şi dinamice. Prin urmare, teza nu doar că explorează metodele existente, dar aduce şi contribuţii semnificative în domeniul managementului microserviciilor, deschizând calea pentru noi cercetări şi aplicări practice.

1.4 Conţinutul tezei de doctorat

Teza începe cu un capitol introductiv în care sunt prezentate domeniul cercetat, motivaţia şi scopul studiului curent.

Capitolul 1 oferă un studiu asupra arhitecturii propuse ce include integrarea părţii hardware şi a părţii software. Această arhitectură oferă baza gestionării şi implementării unui sistem de microservicii.

Capitolul 2 prezintă analiza comparativă în identificarea sistemelor de microservicii ce permit atingerea scalabilităţii, dependabilităţii şi securităţii sistemelor distribuite. Analiza

prezentată în cadrul Capitolului al-2-lea al tezei evidențiază fiecare sistem, împreună cu proprietățile lor.

Capitolul 3 identifică atributele dependabilității care se pretează pentru arhitectura de microservicii și tratează dependabilitatea pentru fiecare nivel în parte al arhitecturii de microservicii identificând tehnicile ce pot fi utilizate în îmbunătățirea atributelor dependabilității.

Capitolul 4 oferă un studiu bibliografic în identificarea tehnologiilor ce permit atingerea scalabilității sistemelor bazate pe arhitecturi de microservicii. Studiul prezentat în cadrul Capitolului al-4-lea al tezei tratează scalabilitatea în cel mai mic detaliu, evidențiind tehnologiile aplicate, împreună cu avantajele și provocările lor în atingerea acestei proprietăți.

Capitolul 5 descrie proiectarea și implementarea unei tehnici de microservicii care utilizează platformele Raspberry Pi, Kubernetes și RabbitMQ oferind și scalabilitate și fiabilitate aplicațiilor IoT. Prin integrarea acestor tehnologii, se urmărește optimizarea utilizării resurselor și asigurarea unei infrastructuri robuste capabile să răspundă cerințelor dinamice ale utilizatorilor.

Capitolul 6 descrie dezvoltarea unei metode dependabile și scalabile pentru integrarea și livrarea continuă. Această metodă îmbunătățește accelerarea dezvoltării aplicațiilor în cadrul unor sisteme de microservicii.

Capitolul 7 prezintă optimizarea dependabilității microserviciilor cu ajutorul utilitarului Helm care constă în dezvoltarea unei aplicații ce gestionează, identifică și rezolvă vulnerabilitățile aduse de versionarea aplicațiilor într-un cluster Kubernetes.

Capitolul 8 prezintă securitatea detaliată a clusterului de microservicii efectuată printr-o metodă care integrează instrumente avansate de securitate—HashiCorp Vault v1.16 pentru gestionarea dinamică a secretelor și OpenID Connect pentru procesele de autentificare.

În final, **Capitolul 9** prezintă concluziile finale, însoțite de o prezentare succintă a contribuțiilor prezentate în teză și a unor căi de cercetare viitoare posibile.

2. Fundamente teoretice

2.1 Arhitectura clusterului

Arhitectura clusterului reprezintă baza pe care se construiesc soluțiile scalabile și dependabile. Un cluster constă dintr-un grup de noduri interconectate, care colaborează pentru a îndeplini sarcini comune, funcționând ca un sistem coerent. Această arhitectură permite distribuirea eficientă a sarcinilor și optimizarea utilizării resurselor, minimizând astfel riscul de eșec singular. Se pune un accent deosebit pe Kubernetes, care este un instrument esențial în gestionarea microserviciilor într-un mediu distribuit. Kubernetes permite orchestrarea automată a containerelor, gestionând echilibrul de sarcină, recuperarea automată în caz de eșec și scalarea automată a serviciilor în funcție de cerințele de trafic. Aceste funcționalități sunt esențiale pentru a asigura că aplicațiile bazate pe microservicii sunt capabile să răspundă cerințelor fluctuante ale utilizatorilor finali.

Printre caracteristicile cheie ale arhitecturii clusterului se numără:

- Distribuția sarcinilor: Sarcinile de procesare, calcul și stocare sunt distribuite între multiple noduri, fiecare nod acționând autonom, dar în coordonare cu celelalte. Aceasta

asigură un echilibru optim al resurselor și continuitate operațională chiar și în fața cerințelor de procesare fluctuante.

- Fiabilitate și redundanță: Arhitectura este concepută pentru a fi extrem de fiabilă, integrând mecanisme de redundanță și restaurare care asigură continuitatea serviciilor și disponibilitatea datelor chiar și în cazul defecțiunilor hardware sau software.
- Interconectare și comunicare: Nodurile din cluster sunt interconectate printr-o rețea care permite transferul rapid de date, facilitând sincronizarea proceselor și partajarea eficientă a resurselor.
- Management și orchestrare: Gestionarea clusterului implică monitorizarea continuă a performanței și optimizarea resurselor prin instrumente de orchestrare, precum Kubernetes și Docker, care automatizează implementarea și operarea aplicațiilor distribuite.
- Flexibilitate și adaptabilitate: Arhitectura clusterului este flexibilă și capabilă să susțină diverse aplicații și servicii, permițând integrarea și livrarea continuă a funcționalităților noi, suportând totodată tehnologii variate, de la big data la inteligență artificială.

În continuare, capitolul detaliază configurația hardware și software a clusterului, evidențiind modul în care aceste componente sunt configurate pentru a asigura performanța, eficiența și fiabilitatea sistemului. Platformele Raspberry Pi au fost utilizate pentru a construi un cluster accesibil și eficient, prezentate în Fig. 2.1, iar software-ul necesar, inclusiv sistemele de operare și instrumentele de orchestrare, a fost implementat pentru a crea un ecosistem robust și scalabil, prezentat în Fig. 2.2. Această configurație a fost validată prin cercetări și articole academice, demonstrând eficiența și aplicabilitatea arhitecturii propuse.

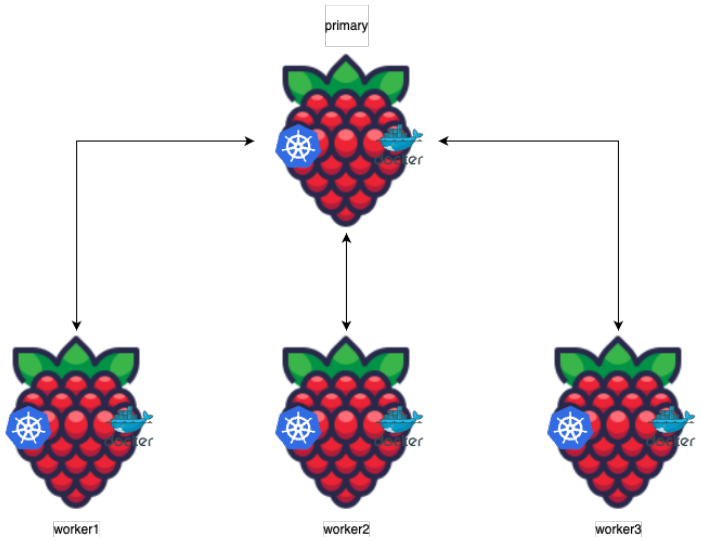

Fig. 2.1.1 Arhitectura clusterului

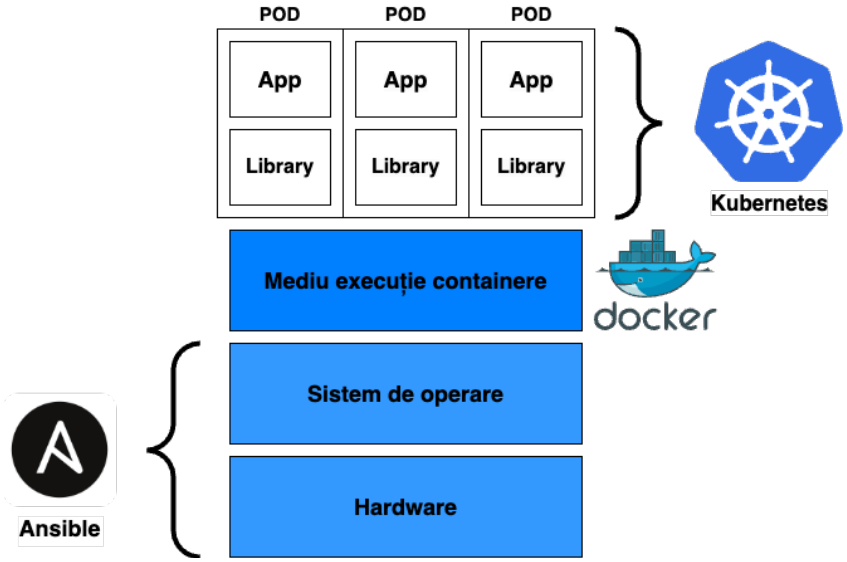

Fig. 2.1.2 Diagrama tehnologică

2.2 Tipuri de servicii: IaaS, CaaS, PaaS, SaaS, FaaS

În contextul evoluției tehnologice actuale, microserviciile s-au impus ca o componentă esențială pentru dezvoltarea, integrarea și livrarea aplicațiilor moderne. Cloud computing-ul reprezintă o paradigmă deosebit de relevantă, care permite accesul la resurse IT configurabile și partajate, cum ar fi rețele, servere, stocare și servicii, toate acestea fiind disponibile la cerere, prin intermediul internetului, cu un minim de efort și interacțiune din partea utilizatorului.

Diferențele între diversele arhitecturi de servicii cloud, cum ar fi Infrastructure as a Service (IaaS), Container as a Service (CaaS), Platform as a Service (PaaS), Software as a Service (SaaS) și Function as a Service (FaaS), influențează semnificativ performanța și eficiența sistemelor de microservicii. În acest capitol, sunt analizate comparativ aceste sisteme, evidențiindu-se avantajele și limitările fiecărei abordări, cu accent pe relevanța acestor tehnologii în proiectele de cercetare.

IaaS oferă resurse de calcul virtualizate, accesibile prin internet, fără a necesita investiții majore în hardware, fiind ideal pentru extinderea capacităților IT. CaaS furnizează un mediu automatizat pentru gestionarea și rularea aplicațiilor containerizate, oferind flexibilitate și eficiență în utilizarea resurselor. PaaS se concentrează pe furnizarea unei platforme complete pentru dezvoltarea și implementarea rapidă a aplicațiilor, eliminând necesitatea gestionării infrastructurii fizice. SaaS oferă aplicații accesibile prin intermediul unui browser web, eliminând necesitatea instalării și întreținerii software-ului local. În fine, FaaS permite rularea codului ca răspuns la evenimente specifice, fără a gestiona infrastructura serverelor, oferind astfel un model eficient și scalabil pentru dezvoltarea aplicațiilor. În Fig. 2.3 sunt ilustrate nivelurile de control și responsabilitate pe care utilizatorii le au în cadrul fiecărui model.

Fig. 2.2.1 Tipuri de servicii

Capitolul concluzionează că, deși fiecare dintre aceste modele de servicii cloud oferă caracteristici și avantaje distincte, sistemul dezvoltat în această teză de doctorat propune o abordare mai complexă, integrând soluții avansate de microservicii și infrastructuri fizice personalizate. Acest sistem oferă o flexibilitate și adaptabilitate superioare, adresând nevoi specifice care nu sunt acoperite în totalitate de serviciile existente de cloud. Astfel, cercetarea evidențiază potențialul de personalizare și optimizare al microserviciilor în medii distribuite, contribuind semnificativ la avansul tehnologic în domeniu.

2.3 Dependabilitatea în microservicii

Acest capitol explorează conceptul de dependabilitate în arhitectura microserviciilor, subliniind importanța asigurării unei funcționări stabile și fiabile a sistemelor distribuite. Dependabilitatea este definită ca probabilitatea unui sistem de a-și îndeplini funcțiile desemnate eficient și fără întreruperi neprevăzute, un aspect esențial în arhitecturile de microservicii datorită naturii distribuite și autonome a componentelor acestora. În Fig. 2.4 este descrisă o viziune de ansamblu asupra dependabilității, fiind prezentate atributele, mijloacele și amenințările.

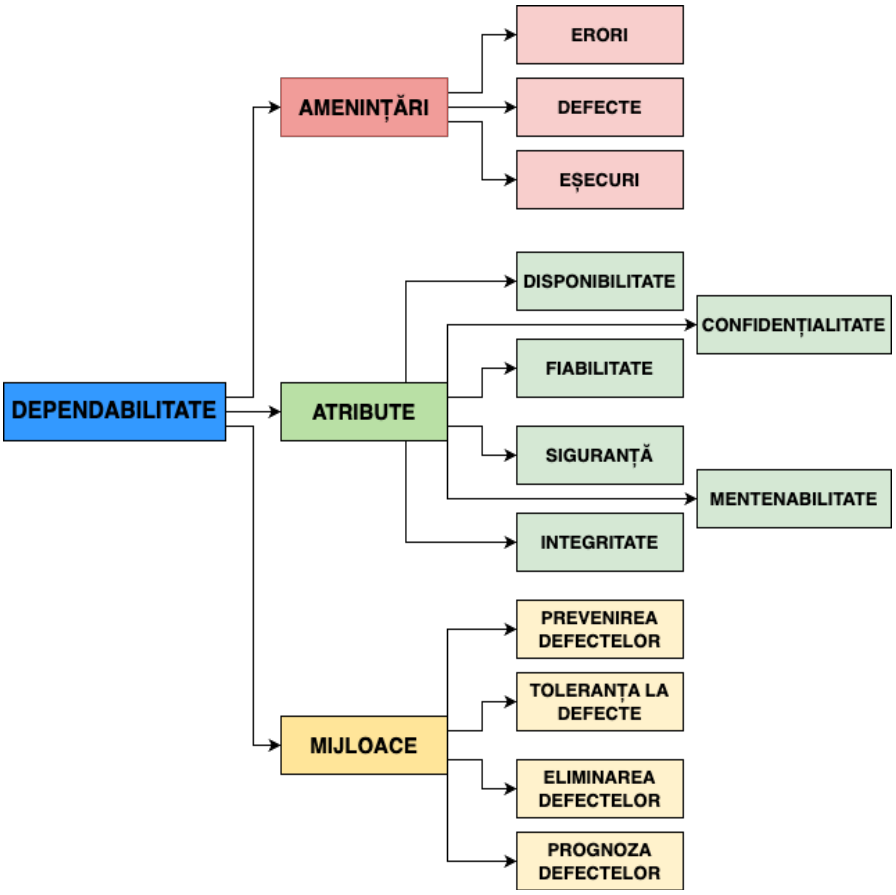

Fig. 2.3.1 Structura dependabilității

Capitolul detaliază mai multe atribute ale dependabilității, inclusiv:

- Disponibilitatea: Capacitatea sistemului de a fi operațional și accesibil cât mai mult posibil, chiar și în momente critice. Disponibilitatea este adesea exprimată prin procentul de timp în care sistemul este funcțional, iar soluțiile precum redundanța și automatizarea redirecționării traficului contribuie la menținerea unei disponibilități ridicate.
- Fiabilitatea: Se referă la capacitatea sistemului de a funcționa corect și de a-și îndeplini funcțiile fără erori. Pentru a îmbunătăți fiabilitatea, sunt implementate soluții precum designul modular, testarea continuă și monitorizarea constantă.
- Siguranța: Asigură că funcționarea defectuoasă a sistemului nu cauzează consecințe negative, precum pierderea de date sau compromiterea integrității. Politicile de securitate, detectarea vulnerabilităților și testele de penetrare sunt esențiale pentru consolidarea siguranței.
- Confidențialitatea: Protejarea informațiilor sensibile împotriva accesului neautorizat este esențială în arhitecturile de microservicii. Gestionarea identităților, criptarea datelor și protecția API-urilor sunt soluții implementate pentru menținerea confidențialității.
- Integritatea: Se referă la capacitatea sistemului de a menține consistența și fiabilitatea datelor. Mecanismele de validare și tranzacțiile distribuite sunt utilizate pentru a asigura integritatea.
- Mentenabilitatea: Capacitatea sistemului de a fi reparat, restaurat, actualizat sau extins eficient. Practicile DevOps, automatizarea și izolarea microserviciilor contribuie la mentenabilitatea ridicată a sistemului.

Capitolul discută, de asemenea, mijloacele dependabilității, incluzând prevenirea defectelor, toleranța la defecte, eliminarea defectelor și prognoza defectelor. Aceste strategii sunt esențiale pentru menținerea performanței optime și a continuității operaționale. În final, sunt abordate amenințările la adresa dependabilității, cum ar fi defecțiunile hardware și software, erorile umane și atacurile cibernetice. Neutralizarea acestor amenințări este crucială pentru asigurarea că arhitectura bazată pe microservicii își îndeplinește funcțiile conform așteptărilor.

2.4 Scalabilitatea în microservicii

Acest capitol detaliază conceptul de scalabilitate în arhitectura bazată pe microservicii. Scalabilitatea este esențială pentru menținerea performanței și eficienței sistemelor, chiar și în condițiile creșterii cerințelor de utilizare și ale resurselor. În arhitectura microserviciilor, scalabilitatea oferă flexibilitate în gestionarea resurselor, permițând fiecărui microserviciu să fie scalat independent, optimizând astfel resursele și reducând costurile operaționale.

2.4.1 Tehnici de scalare

Scalarea Orizontală și Scalarea Verticală sunt două tehnici esențiale în arhitecturile IT moderne, iar alegerea între aceste tehnici depinde de cerințele aplicației, de buget și de infrastructura disponibilă. Mai jos și în Fig. 2.5 sunt detaliate cele două metode de scalare:

- Scalarea Orizontală implică adăugarea de noi noduri sau instanțe pentru a distribui sarcina de lucru și a îmbunătăți performanța sistemului. Aceasta este preferată în arhitecturile de microservicii datorită flexibilității și rezilienței pe care le oferă.
- Scalarea Verticală constă în creșterea capacității hardware a unui nod existent, adăugând resurse precum memorie RAM sau procesoare mai rapide. Este preferată în arhitecturile monolitice pentru simplitatea sa și pentru evitarea fragmentării.

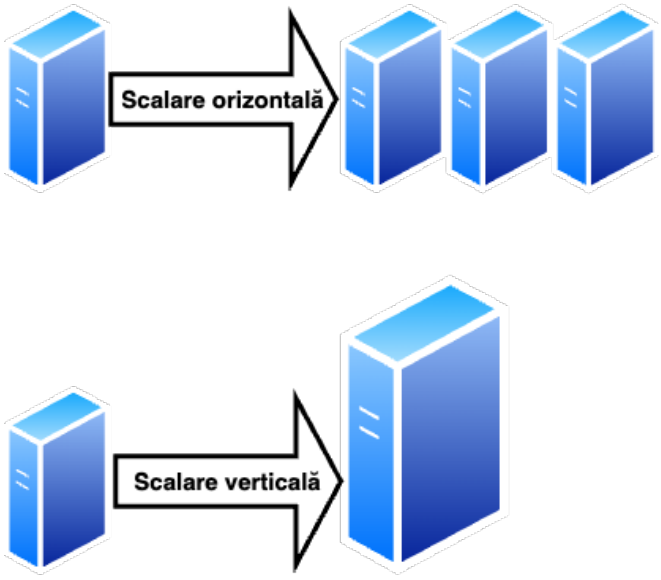

Fig. 2.4.1 Tehnici de scalare

2.4.2 Instrumente și platforme pentru scalabilitate

Acest subcapitol explorează virtualizarea, containerizarea și orchestrarea ca tehnologii fundamentale pentru asigurarea scalabilității în sistemele distribuite și arhitecturile de microservicii:

- Virtualizarea permite rularea mai multor mașini virtuale (VM) pe același server fizic, maximizând utilizarea resurselor și reducând costurile operaționale.
- Containerizarea oferă un mediu izolat și portabil pentru aplicații, esențial în arhitecturile de microservicii pentru scalabilitate și consistența mediului de dezvoltare.
- Orchestrarea cu Kubernetes gestionează microserviciile containerizate, asigurând echilibrarea sarcinilor, scalarea automată și integrarea continuă, toate acestea fiind cruciale pentru menținerea performanței și disponibilității sistemului.

2.4.3 Discuții

Capitolul se încheie cu o evaluare a provocărilor și soluțiilor asociate cu scalabilitatea unui cluster de microservicii bazat pe platforme Raspberry Pi. Printre provocări se numără gestionarea dependențelor între servicii și limitările hardware ale Raspberry Pi. Soluțiile propuse includ:

- Optimizarea performanței software pentru a reduce consumul de resurse.
- Utilizarea stocării externe pentru a extinde capacitatea de stocare și a îmbunătăți accesul la date.
- Adăugarea de resurse suplimentare pentru a răspunde cererilor crescute.

Aceste strategii sunt esențiale pentru valorificarea potențialului arhitecturilor bazate pe microservicii și pentru a asigura succesul în implementarea sistemelor distribuite scalabile și eficiente.

3. Proiectarea și implementarea unei soluții dependabile și scalabile de microservicii bazate pe Raspberry Pi, Kubernetes și RabbitMQ

Acest capitol explorează proiectarea și implementarea unei soluții inovatoare de microservicii utilizând platformele Raspberry Pi, Kubernetes și RabbitMQ, pentru a crea o infrastructură scalabilă și fiabilă destinată aplicațiilor IoT. Scopul acestei soluții este de a optimiza utilizarea resurselor și de a asigura o infrastructură robustă, capabilă să răspundă cerințelor dinamice ale utilizatorilor. Soluția a fost prezentată într-un articol publicat la conferința internațională AQTR 2020, subliniind importanța adoptării tehnologiilor cloud și microservicii.

3.1 Concepte de bază

Subcapitolul prezintă conceptele cheie utilizate în dezvoltarea soluției propuse:

- Ansible (IaC): Utilizat pentru automatizarea configurării infrastructurii și instalarea dependențelor necesare pe platformele Raspberry Pi.
- Containere: Folosite pentru a crea medii de operare izolate și fiabile pentru aplicații, facilitând portabilitatea și consistența implementării.
- Kubernetes: Platforma de orchestrare a aplicațiilor containerizate, asigurând distribuirea sarcinilor de lucru și scalabilitatea automată a serviciilor.
- Helm Charts: Instrument de gestionare a pachetelor Kubernetes, care simplifică implementarea și managementul aplicațiilor.
- RabbitMQ: Broker de mesagerie distribuit și scalabil, implementat ca un obiect Kubernetes numit StatefulSet, care asigură integritatea datelor și comunicarea eficientă între microservicii.

3.2 Descrierea soluției

Arhitectura propusă utilizează trei platforme Raspberry Pi, configurate și gestionate automat prin Ansible. Soluția IaC (Infrastructură ca și Cod) prezentată în Fig. 3.1, dezvoltă o infrastructură containerizată și scalabilă, utilizând Docker și Kubernetes pentru gestionarea și orchestrarea aplicațiilor. RabbitMQ, implementat prin Helm, facilitează comunicarea între aplicații și senzori, esențială pentru proiectele IoT.

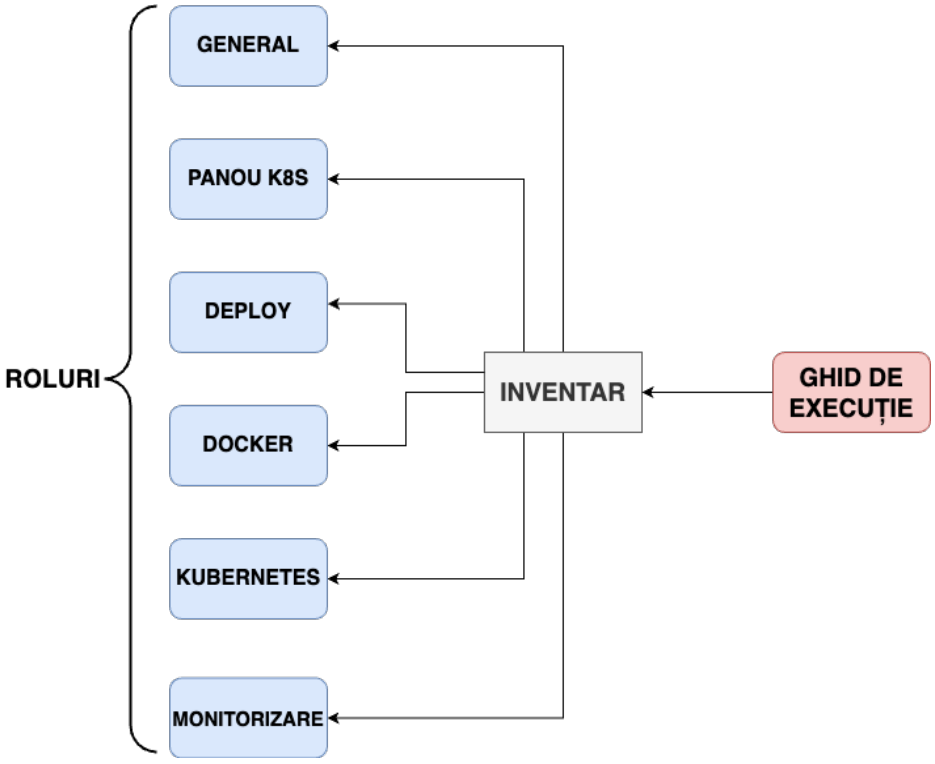

Fig. 3.2.1 Structură IaC

3.3 Motivație și Concluzii

Soluția dezvoltată este multifuncțională, permițând instalarea și gestionarea automată a mai multor microservicii și aplicații. Autoscalarea clusterului asigură adaptarea rapidă la cerințele fluctuante, redistribuind automat aplicațiile în cazul defectării unui nod. Soluția promovează dependabilitatea și scalabilitatea, esențiale pentru infrastructuri IoT și alte aplicații critice.

Implementarea acestei soluții urmărește automatizarea proceselor, reducerea costurilor și simplificarea ciclului de viață al fluxului de lucru, utilizând instrumente cu acces liber la cod, precum Ansible, Docker, Kubernetes, Helm și RabbitMQ. Aceasta oferă o infrastructură flexibilă și eficientă, capabilă să răspundă nevoilor complexe și dinamice ale utilizatorilor.

4. Dezvoltarea unei metode dependabile și scalabile pentru integrarea și livrarea continuă

Acest capitol explorează dezvoltarea unei metode inovatoare pentru integrarea și livrarea continuă (CI/CD) a aplicațiilor, cu scopul de a asigura o infrastructură fiabilă și scalabilă. Metoda propusă a fost detaliată într-un articol publicat în 2022 în jurnalul „Sensors", editura MDPI. Cercetarea răspunde nevoilor actuale ale organizațiilor de a adopta practici agile pentru a îmbunătăți eficiența și viteza ciclului de dezvoltare software, utilizând un cadru automatizat pentru CI/CD.

Concepte de bază:

- Integrarea continuă (CI) implică integrarea frecventă a codului într-un registru comun, facilitând testarea automată și colaborarea rapidă în echipă.
- Livrarea continuă (CD) automatizează livrarea aplicațiilor, asigurând că fiecare modificare este testată și implementată eficient.
- Ciclul de lucru CI/CD reprezintă strategia de planificare, dezvoltare și implementare automatizată, reducând execuția manuală.
- Instrumente CI/CD: Soluția propusă folosește GitLab pentru gestionarea ciclurilor de lucru, Docker pentru containerizare, și Helm pentru managementul aplicațiilor Kubernetes.

Metoda include o generare automată a ciclurilor de lucru, care descoperă fișierele de integrare și livrare din registrele Git și creează pași automatizați pentru execuția lor. Aceste cicluri se regenerează automat la fiecare schimbare în cod, asigurând actualizarea continuă și eficientă. Metoda de generare include două moduri: pași implicați și personalizați, permițând flexibilitatea necesară pentru diferite nevoi ale proiectelor. Fig. 4.1 reprezintă diagrama proceselor soluției propuse, prezentând întregul ciclu de lucru automatizat.

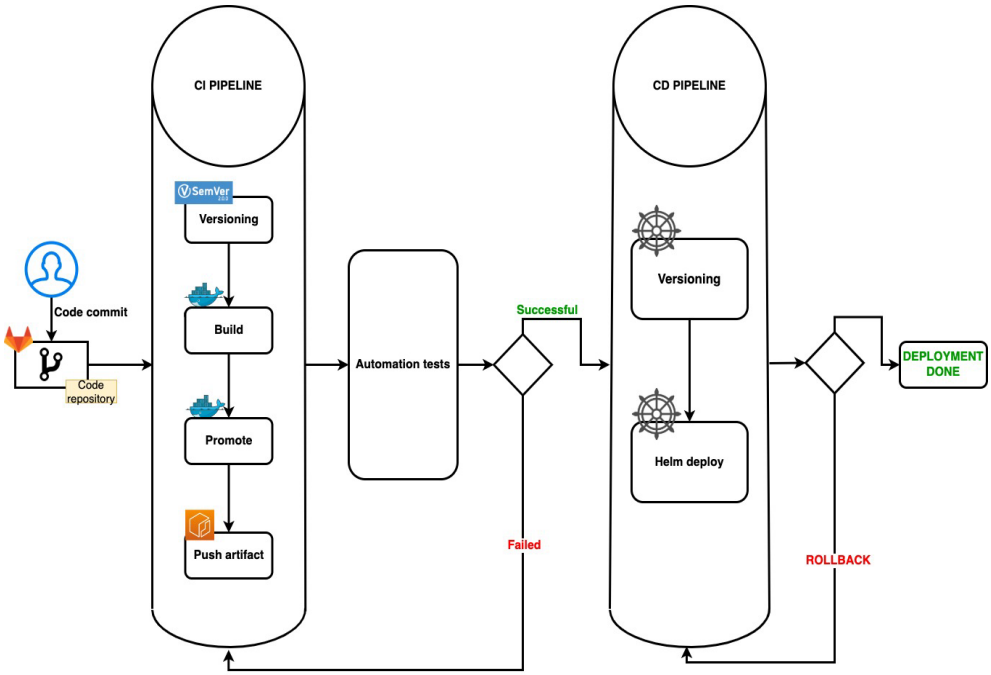

Fig. 3.3.1 Diagrama proceselor soluției propuse

Implementarea soluției aduce beneficii semnificative:

- Reducerea timpului de livrare: Ciclurile de lucru automatizate permit livrarea rapidă a aplicațiilor, reducând semnificativ timpul de la dezvoltare la implementare.
- Fiabilitate și securitate: Metoda asigură testarea continuă și validarea codului înainte de livrare, reducând riscul de introducere a vulnerabilităților.
- Costuri reduse: Automatizarea reduce costurile asociate cu intervențiile manuale, contribuind la economisirea resurselor și optimizarea proceselor.

Rezultatele experimentale au demonstrat eficiența metodei propuse în comparație cu abordările manuale și alte soluții CI/CD, evidențiind economii semnificative de timp și costuri.

Metoda propusă oferă o soluție robustă pentru CI/CD, asigurând scalabilitate, fiabilitate, și mentenabilitate în infrastructurile moderne de microservicii. Viitoarele îmbunătățiri includ rescrierea soluției în Golang pentru o mai bună compatibilitate cross-platform și extinderea funcționalității pentru a sprijini implementarea pe multiple platforme și cloud-uri publice și private. Aceasta ar permite dezvoltarea unui algoritm care să determine cloud-ul optim pentru rularea aplicațiilor, oferind o soluție flexibilă și eficientă pentru nevoile complexe ale organizațiilor moderne.

5. Optimizarea dependabilității microserviciilor cu ajutorul utilitarului Helm

Acest capitol explorează optimizarea dependabilității microserviciilor utilizând Helm, un manager de pachete popular pentru Kubernetes. Capitolul prezintă dezvoltarea unei aplicații inovatoare, scrise în Golang, care automatizează gestionarea și identificarea versiunilor depreciate ale aplicațiilor livrate prin Helm într-un cluster Kubernetes. Soluția este

validată printr-un articol publicat în 2023 la conferința „International Conference on Advanced Scientific Computing (ICASC)”.

Aplicația dezvoltată în Golang permite gestionarea eficientă a versiunilor Helm învechite sau depreciate, aspecte critice pentru menținerea stabilității și securității infrastructurii Kubernetes. Prin integrarea cu API-ul Kubernetes și Helm SDK, soluția preia și analizează versiunile aplicațiilor Helm din cluster, identificând automat starea acestora, inclusiv dacă sunt cele mai recente și dacă au fost marcate ca depreciate. Această analiză ajută la prevenirea problemelor legate de securitate, compatibilitate și performanță, asigurând continuitatea operațională a aplicațiilor. În Fig. 5.1 este descrisă diagrama UML a soluției.

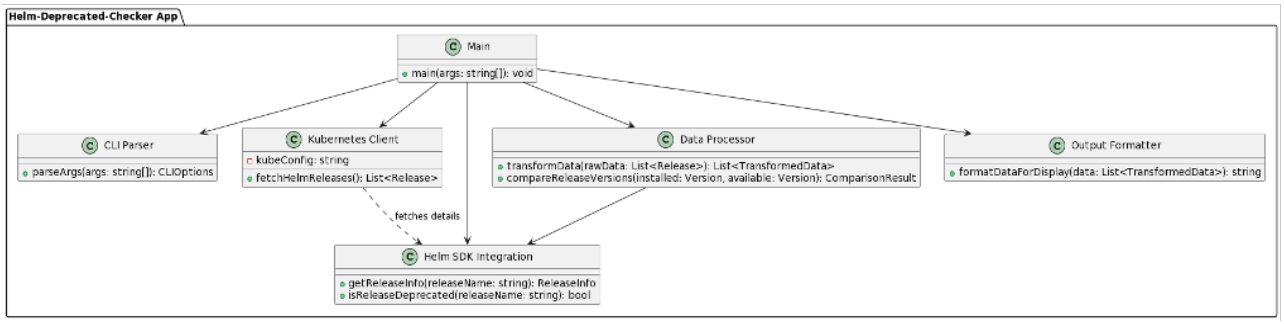

Fig. 3.3.1 Diagrama UML

Aplicația contribuie semnificativ la îmbunătățirea gestionării microserviciilor prin:

- Analiză detaliată: Oferă o evaluare aprofundată a stării versiunilor aplicațiilor livrate cu Helm, depășind metricile generale ale observabilității clusterului.
- Flexibilitate: Permite utilizarea cu registre Helm personalizate, oferind rezultate detaliate și relevante pentru diverse medii.
- Eficiență: Demonstrează timpi de procesare competitivi cu un consum redus de resurse, făcând-o ideală pentru medii cu resurse limitate.

Deși aplicația este eficientă, are și limitări, cum ar fi dependența de timpii de răspuns ai registrelor externe și lipsa stocării datelor istorice. Îmbunătățirile viitoare ar putea include implementarea nivelelor de cache pentru o performanță mai consistentă și adăugarea unor funcționalități de stocare și vizualizare a datelor istorice. Integrarea cu CI/CD ar putea automatiza procesul de utilizare a celor mai recente versiuni sigure. Pe măsură ce Kubernetes devine tot mai central în infrastructurile moderne, aplicații precum cea descrisă vor fi esențiale pentru asigurarea unei gestionări eficiente și sigure. Această cercetare subliniază necesitatea și potențialul inovațiilor în acest domeniu.

6. Securitatea detaliată a clusterului de microservicii

Acest capitol explorează configurarea hardware și standardele de securitate pentru un cluster de microservicii bazat pe Kubernetes și Raspberry Pi, destinat monitorizării mediului în aplicații IoT. Studiul subliniază importanța securității în contextul implementărilor IoT și propune o soluție robustă pentru gestionarea secretelor și autentificarea sigură, folosind HashiCorp Vault și OpenID Connect (OIDC).

Cercetarea introduce o arhitectură de securitate multi-nivel care integrează senzorii de mediu BME680 în clustere Raspberry Pi, gestionate de Kubernetes, pentru a asigura securitatea și scalabilitatea aplicațiilor IoT. Configurația hardware este esențială, incluzând măsuri de

criptare a datelor și controale de acces stricte, iar Kubernetes este ales pentru orchestrarea eficientă a containerelor. În Fig. 6.1 fiind prezentată diagrama arhitecturală a soluției propuse.

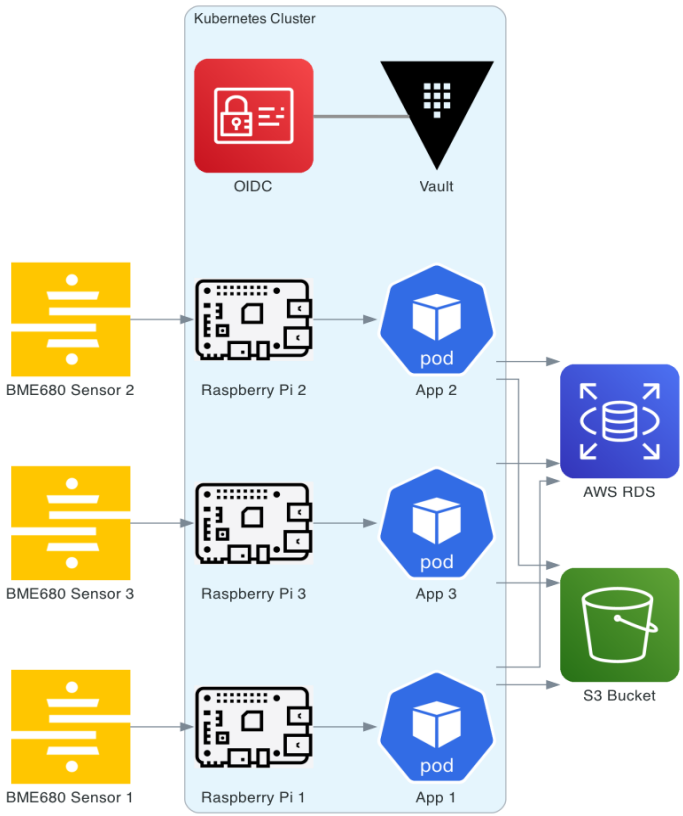

Fig. 3.3.1 Diagrama arhitecturală

Măsuri de securitate:

- Criptarea datelor și controlul accesului: Datele sunt protejate prin protocoale HTTPS/TLS, iar accesul la sistem este restricționat prin politici RBAC.
- Gestionarea secretelor: HashiCorp Vault gestionează în siguranță parolele, generând dinamic secrete pentru a minimiza riscurile de securitate.
- Autentificarea OIDC: Aceasta asigură că toate entitățile sunt autentificate înainte de a interacționa cu resursele sistemului, oferind o protecție suplimentară împotriva accesului neautorizat.

Testele de securitate au evidențiat reziliența sistemului împotriva amenințărilor cibernetice. Metodologiile de testare au inclus scanarea vulnerabilităților, testarea prin penetrare și evaluarea gestionării secretelor. Implementarea măsurilor de securitate a demonstrat o reducere semnificativă a expunerilor la vulnerabilități și o îmbunătățire a capacității de răspuns a sistemului cu 30%.

Capitolul subliniază importanța unei strategii de securitate bine definite în proiectele IoT, accentuând contribuțiile aduse de integrarea HashiCorp Vault și OIDC. De asemenea, se propun direcții viitoare pentru îmbunătățirea securității, cum ar fi răspunsul automatizat la incidente și integrarea senzorilor avansați pentru o monitorizare mai precisă a mediului. Această lucrare stabilește un precedent pentru dezvoltarea de soluții IoT securizate și scalabile, oferind o bază solidă pentru cercetări viitoare în domeniu.

7. Concluzii

7.1 Concluzii generale

În cadrul tezei de doctorat elaborate s-a realizat în primul rând un studiu asupra gestionării unui sistem bazat pe microservicii. Acest studiu oferă o contribuţie semnificativă în domeniul arhitecturilor de microservicii, explorând atât teoretic cât şi practic modalităţile prin care aceste arhitecturi pot îmbunătăţi dependabilitatea şi scalabilitatea sistemelor modern şi distribuite. Un alt aspect esenţial abordat în teză este utilizarea tehnologiilor containerizării şi orchestrării, cum ar fi Docker şi Kubernetes, care permit un management eficient al microserviciilor şi asigură scalabilitatea automată în funcţie de cerinţele de performanţă. Studiul a evidenţiat şi importanţa unui management riguros al secretelor şi autentificării, utilizând instrumente precum HashiCorp Vault şi OpenID Connect, pentru a asigura securitatea aplicaţiilor într-un mediu IoT complex. Un alt punct important de subliniat este accentul pus pe mentenabilitate şi fiabilitate în contextul microserviciilor. Teza a demonstrat cum sistemele de microservicii, prin abordările de tip CI/CD (Integrare şi livrare continuă), contribuie la menţinerea unui ciclu de viaţă al aplicaţiilor mai eficient, asigurând astfel că actualizările şi modificările pot fi implementate fără a afecta stabilitatea generală a sistemului, unde disponibilitatea continuă este esenţială. Această abordare sofisticată, care îmbină aspecte de securitate, dependabilitate şi scalabilitate, reprezintă o contribuţie valoroasă la dezvoltarea de soluţii IT moderne, pregătite să răspundă cerinţelor complexe ale lumii digitale actuale.

Concluziile acestei teze subliniază importanţa adaptabilităţii şi flexibilităţii în proiectarea sistemelor distribuite, oferind totodată o fundaţie solidă pentru cercetări viitoare şi aplicaţii practice în diverse domenii, de la medii de producţie până la implementări IoT avansate. Astfel, lucrarea contribuie la avansarea cunoştinţelor în managementul microserviciilor şi oferă direcţii clare pentru îmbunătăţirea şi optimizarea continuă a acestor sisteme.

În cadrul tezei se ţine cont de studiile realizate în literatura de specialitate, dar suplimentar se introduce gestionarea, scalabilitatea, dependabilitatea şi securitatea unui sistem bazat pe microservicii. Astfel a fost propusă în partea a doua a tezei, în contribuţia personală, diferite tehnici prin care să se poată gestiona, scala, îmbunătăţi dependabilitatea şi securiza un sistem distribuit bazat pe microservicii.

7.2 Originalitatea şi contribuţiile inovative ale tezei

În cadrul acestui capitol sunt prezentate opt contribuţii principale în îmbunătăţirea scalabilităţii, dependabilităţii şi securităţii în cadrul unui sistem bazat pe o arhitectură de microservicii.

Contribuţia 1: Studiu asupra arhitecturii propuse ce include integrarea părţii hardware şi a părţii software. Această arhitectură oferă baza gestionării şi implementării unui sistem de microservicii. Beneficiile şi rezultatele experimentale ale acestei arhitecturi au fost diseminate în trei articole [12]. [30], [39] publicate în conferinţe ISI Proceedings WOS.

Contribuţia 2: Analiza comparativă în identificarea sistemelor de microservicii ce permit atingerea scalabilităţii, dependabilităţii şi securităţii sistemelor distribuite. Analiza prezentată în cadrul Capitolului al-2-lea al tezei evidenţiază fiecare sistem, împreună cu

proprietățile lor și a fost diseminată în articolul [39], ”A Performance Evaluation of Modern Virtualization Techniques” publicat într-o conferință ISI Proceedings WOS.

Contribuția 3: Identificarea atributelor dependabilității care se pretează pentru arhitectura de microservicii. Capitolul al-3-lea al tezei tratează dependabilitatea pentru fiecare nivel în parte al arhitecturii de microservicii identificând tehnicile ce pot fi utilizate în îmbunătățirea atributelor dependabilității. Rezultate obținute au fost diseminate în articolul [169] într-o revistă ISI Q1 cu factor de impact 3.4.

Contribuția 4: Studiu bibliografic în identificarea tehnologiilor ce permit atingerea scalabilității sistemelor bazate pe arhitecturi de microservicii. Studiul prezentat în cadrul Capitolului al-4-lea al tezei tratează scalabilitatea în cel mai mic detaliu, evidențiind tehnologiile aplicate, împreună cu avantajele și provocările lor în atingerea acestei proprietăți. Beneficiile acestui studiu fiind diseminate în articolul [39], ”A Performance Evaluation of Modern Virtualization Techniques” publicat într-o conferință ISI Proceedings WOS.

Contribuția 5: Proiectarea și implementarea unei tehnici de microservicii care utilizează platformele Raspberry Pi, Kubernetes și RabbitMQ oferind și scalabilitate și fiabilitate aplicațiilor IoT. Prin integrarea acestor tehnologii, se urmărește optimizarea utilizării resurselor și asigurarea unei infrastructuri robuste capabile să răspundă cerințelor dinamice ale utilizatorilor. Performanțele aduse de către această tehnică de microservicii au fost diseminate în două articole [12] și [39], publicate în conferințe ISI Proceedings WOS.

Contribuția 6: Dezvoltarea unei metode dependabile și scalabile pentru integrarea și livrarea continuă care este prezentată în Capitolul al-6-lea al tezei. Această metodă îmbunătățește accelerarea dezvoltării aplicațiilor în cadrul unor sisteme de microservicii. Rezultatele experimentale și beneficiile acestei metode au fost diseminate în articolul [131] publicat într-o revistă ISI Q1 cu factor de impact 3.4.

Contribuția 7: Optimizarea dependabilității microserviciilor cu ajutorul utilitarului Helm care constă în dezvoltarea unei aplicații care gestionează, identifică și rezolvă vulnerabilitățile aduse de versionarea aplicațiilor într-un cluster Kubernetes. Rezultate obținute sunt prezentate în Capitolul al-7-lea și au fost diseminate si în articolul [137] publicat într-o revistă ISI Proceedings WOS.

Contribuția 8: Securitatea detaliată a clusterului de microservicii efectuată printr-o metodă care integrează instrumente avansate de securitate—HashiCorp Vault v1.16 pentru gestionarea dinamică a secretelor și OpenID Connect pentru procesele de autentificare. Rezultatele, beneficiile și interpretările fiind prezentate în Capitolul al-8-lea, precum și într-un articol [141] publicat într-o revistă ISI Q2 cu factor de impact 2.6.

7.3 Diseminarea rezultatelor

Teza de doctorat s-a concretizat în urma studiilor prezentate în cadrul celor patru rapoarte de cercetare susținute pe durata celor cinci ani de doctorat:

- Ionuț-Cătălin Donca, “Studiu privind tehnicile de interconectare a clusterului de microservicii cu alte dispozitive IoT/senzori”, Universitatea Tehnică din Cluj-Napoca, Cluj-Napoca, 2020.
- Ionuț-Cătălin Donca, “Modelarea dependabilității și scalabilității clusterului de microservicii”, Universitatea Tehnică din Cluj-Napoca, Cluj-Napoca, 2021.
- Ionuț-Cătălin Donca, “Dezvoltarea unor soluții dependabile și scalabile pentru procesarea informațiilor transmise de către senzori/dispozitive IoT către

clusterul de microservicii", Universitatea Tehnică din Cluj-Napoca, Cluj-Napoca, 2021.

- Ionuț-Cătălin Donca, "Validarea experimentală și optimizarea soluțiilor dezvoltate", Universitatea Tehnică din Cluj-Napoca, Cluj-Napoca, 2022.

În Tabelul 7.1 sunt prezentate revistele și conferințele în cadrul cărora s-a realizat diseminarea rezultatelor.

Tabel 7.1 Diseminarea rezultatelor în reviste și conferințe internaționale

Nr. articol	Nr. citare	Detalii articol	Categorie articol
1.	[12]	**Ionuț-Cătălin Donca**, Cosmina Corches, Ovidiu Stan and Liviu Miclea, "Autoscaled RabbitMQ Kubernetes Cluster on single-board computers," 2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), 2020, pp. 1-6, doi: 10.1109/AQTR49680.2020.9129886.	Conf. ISI Proceedings WOS
2.	[30]	Cosmina Corches, **Ionuț-Cătălin Donca**, Ovidiu Stan, Liviu Miclea and Mihai Daraban, "Analyzing the RFID Failure Impact on Availability of IoT Services" 2020 IEEE 26th International Symposium for Design and Technology in Electronic Packaging (SIITME), 2020, pp. 163-168, doi: 10.1109/SIITME50350.2020.9292209	Conf. ISI Proceedings WOS Articol Premiat
3.	[39]	**Ionuț-Cătălin Donca**, Ovidiu Stan, Liviu Miclea, "Proposed model for a Microservices Cluster" in proceedings of the 21st International Carpathian Control Conference (ICCC), Virtual, October 2020, pp. 27–29, doi: 10.1109/ICCC49264.2020.9257217.	Conf. BDI Proceedings, Indexat IEEE Xplore
4.	[104]	**Ionuț-Cătălin Donca**, Ovidiu Stan and Liviu Miclea, "A Performance Evaluation of Modern Virtualization Techniques," 2022 IEEE International Conference on Automation, Quality and Testing,	Conf. ISI Proceedings WOS

		Robotics (AQTR), Cluj-Napoca, Romania, 2022, pp. 1-6, doi: 10.1109/AQTR55203.2022.9801951.	
5.	[131]	**Ionuț-Cătălin Donca**, Ovidiu Stan, Marius Misaros, Dan Gota; Liviu Miclea, "Method for Continuous Integration and Deployment Using a Pipeline Generator for Agile Software Projects" Sensors MDPI, June 2022, Volume 22, Issue 12, 4637. https://doi.org/10.3390/s22124637	Revistă ISI - Q1
6.	[147]	Marius Misaros, Ovidiu Stan, **Ionuț-Cătălin Donca**, Liviu Miclea, "Autonomous Robots for Services—State of the Art, Challenges, and Research Areas", Sensors MDPI, May 2023, Volume 23, Issue 10, 4962. https://doi.org/10.3390/s23104962	Revistă ISI – Q1
7.	[137]	**Ionuț-Cătălin Donca**, Liviu Miclea, "Automated Detection and Management of Deprecated Helm Releases in Kubernetes Clusters", 2023 International Conference on Advanced Scientific Computing (ICASC), Cluj-Napoca, Romania, October 2023, pp. 1-5, doi: 10.1109/ICASC58845.2023.10328029.	Conf. BDI Proceedings, Indexat IEEE Xplore
8.	[141]	**Ionuț-Cătălin Donca**, Ovidiu Stan, Marius Misaros, Anca Stan, Liviu Miclea, "Comprehensive Security for IoT Devices with Kubernetes and Raspberry Pi Cluster". Electronics MDPI, April 2024, Volume 13, Issue 9, 1613. https://doi.org/10.3390/electronics13091613	Revistă ISI – Q2
9.	[169]	Marius Misaroș, Ovidiu Petru Stan, Szilard Enyedi, Anca Stan, **Ionuț-Cătălin Donca**, Liviu-Cristian Miclea," A Method for Assessing the Reliability of the Pepper Robot in Handling Office Documents: A Case Study" Biomimetics MDPI, September 2024, Vol. 9, Issue 558.	Revistă ISI – Q1

		https://doi.org/10.3390/biomimetics9090558	