УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Раде Пејановић

CLI алат за аутоматизовано креирање локалних развојних окружења

ЗАВРШНИ РАД

Основне академске студије

Нови Сад, 2025

	УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА ЗАВРШНИ РАД	Датум:

(Податке уноси предметни наставник - ментор)

Студијски програм:	Софтверско инжењерство и информационе технологије		
Студент:	Раде Пејановић	Број индекса:	SV10/2021
Степен и врста студија:	Основне академске студије		
Област:	Електротехничко и рачунарско инжењерство		
Ментор:	Милош Симић		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ЗАВРШНИ РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

НАСЛОВ ЗАВРШНОГ РАДА:

CLI алат за аутоматизовано креирање локалних развојних окружења

ТЕКСТ ЗАДАТКА:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et.

Руководилац студијског програма:	Ментор рада:

Примерак за: □ - Студента; □ - Ментора

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Дипломски - бечелор рад
Аутор, **АУ**:		Раде Пејановић
Ментор, **МН**:		Др Милош Симић, доцент
Наслов рада, **НР**:		CLI алат за аутоматизовано креирање локалних развојних окружења
Језик публикације, **ЈП**:		српски/ћирилица
Језик извода, **ЈИ**:		српски/енглески
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2025
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад, трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		5/28/23/0/6/0/2
Научна област, **НО**:		Електротехничко и рачунарско инжењерство
Научна дисциплина, **НД**:		Примењене рачунарске науке и информатика
Предметна одредница/Кључне речи, **ПО**:		Шаблон, завршни рад, упутство
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		Овај документ представља упутство за писање завршних радова на Факултету техничких наука Универзитета у Новом Саду. У исто време је и шаблон за Typst.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		01.01.2025
Чланови комисије, **КО**:	Председник:	Др Петар Петровић, ванредни професор
	Члан:	Др Марко Марковић, доцент
	Члан:	Потпис ментора
	Члан, ментор:	Др Милош Симић, доцент

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:	
Identification number, **INO**:	
Document type, **DT**:	Monographic publication
Type of record, **TR**:	Textual printed material
Contents code, **CC**:	
Author, **AU**:	Rade Pejanović
Mentor, **MN**:	Igor Dejanović, Phd., asist. professor
Title, **TI**:	CLI tool for automated creation of local development environments
Language of text, **LT**:	Serbian
Language of abstract, **LA**:	Serbian/English
Country of publication, **CP**:	Republic of Serbia
Locality of publication, **LP**:	Vojvodina
Publication year, **PY**:	2025
Publisher, **PB**:	Author's reprint
Publication place, **PP**:	Novi Sad, Dositeja Obradovica sq. 6
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)	5/28/23/0/6/0/2
Scientific field, **SF**:	Electrical and Computer Engineering
Scientific discipline, **SD**:	Applied computer science and informatics
Subject/Key words, **S/KW**:	Template, thesis, tutorial
UC	
Holding data, **HD**:	The Library of Faculty of Technical Sciences, Novi Sad
Note, **N**:	
Abstract, **AB**:	This document provides guidelines for writing final theses at the Faculty of Technical Sciences, University of Novi Sad. At the same time, it serves as a Typst template.
Accepted by the Scientific Board on, **ASB**:	
Defended on, **DE**:	01.01.2025
Defended Board, **DB**: President:	Petar Petrović, Phd., assoc. professor
Member:	Marko Marković, Phd., asist. professor
Member:	
Member, Mentor:	Igor Dejanović, Phd., asist. professor

Menthor's sign

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

ИЗЈАВА О НЕПОСТОЈАЊУ СУКОБА ИНТЕРЕСА

Изјављујем да нисам у сукобу интереса у односу ментор – кандидат и да нисам члан породице (супружник или ванбрачни партнер, родитељ или усвојитељ, дете или усвојеник), повезано лице (крвни сродник ментора/кандидата у правој линији, односно у побочној линији закључно са другим степеном сродства, као ни физичко лице које се према другим основама и околностима може оправдано сматрати интересно повезаним са ментором или кандидатом), односно да нисам зависан/на од ментора/кандидата, да не постоје околности које би могле да утичу на моју непристрасност, нити да стичем било какве користи или погодности за себе или друго лице било позитивним или негативним исходом, као и да немам приватни интерес који утиче, може да утиче или изгледа као да утиче на однос ментор-кандидат.

У Новом Саду, дана _____

Ментор

_____

Кандидат

_____

Садржај

Глава 1

Увод

Савремени софтверски системи све чешће користе дистрибуирану архитектуру и микросервисе. Више независних компоненти међусобно комуницира ради остваривања заједничке функционалности. Свака компонента система је развијана и тестирана у сопственом окружењу које је конфигурисано за њене потребе. Окружење представља:

- одређену дистрибуцију и верзију оперативног система (*operating system*, OS)
- предефинисан сет системских променљивих (*environment variables*)
- скуп зависности (*dependencies*) неопходних за исправно функционисање компоненте

Како би се спречиле интерференције између компоненти, односно верзија њихових зависности и подешавања системских променљивих, потребно их је изоловати. Изолација омогућава паковање компоненте у сопствено окружење у коме је гарантовано њено исправно функционисање без интерференције остатка система. Изолација се постиже на два начина:

- физичка, где свака компонента система ради на посебној физичкој машини
- логичка, где више виртуелних машина дели једну физичку, а свака виртуелна машина представља појединачну компоненту система

Физичка изолација је непрактична и финансијски неисплатива. Логичка изолација омогућава бољу расподелу ресурса и економичније коришћење хардвера. Најчешће се постиже кроз виртуелизацију, која омогућава покретање више независних окружења на једној физичкој машини. Та окружења могу бити виртуелне машине (*virtual machines*, VMs), које емулирају цео оперативни систем и хардвер, или контејнери (*containers*), који виртуелизују само оперативни систем и омогућавају брже и лакше покретање апликација са свим зависностима. Ручно покретање и конфигурисање више виртуелних окружења за сложене системе је дуготрајно и подложно грешкама.

Како би се овај процес поједноставио и верзионисао, користи се концепт инфраструктуре као кода (*Infrastructure as Code*, IaC). Конфигурација окружења се описује декларативно (нпр. YAML или JSON). То омогућава аутоматизовано креирање, управљање и репродукцију развојних окружења. Овакви приступи смањују време потребно за постављање окружења и повећавају поузданост процеса развоја и тестирања.

Циљ овог рада је развој алата који омогућава једноставно креирање и управљање локалним развојним окружењима применом принципа IaC у оквиру *Constellations* [1] пројекта отвореног кода (*open source*). Алат аутоматизује:

- покретање контролне равни (*control plane*) на матичној машини (*host machine*)
- покретања N виртуелних машина, њихово умрежавање и управљање животним циклусом (креирање, покретање, паузирање, гашење и брисање)
- подешавање окружења инсталацијом потребних зависности и конфигурисања системских променљивих
- удаљено покретање и заустављање дефинисаних сервиса на виртуелним машинама

На овај начин се убрзава процес припреме окружења за развој и тестирање, смањује могућност грешке и обезбеђује доследност окружења на различитим системима.

Глава 2

Теоријске основе и постојећа истраживања

Ово поглавље описује основне теоријске концепте неопходне за разумевање решења (поглавље 2.1), технологије коришћене у решењу (поглавље 2.2) и постојећа решења (поглавље 2.3)

2.1 Теоријски концепти релевантни за разумевање решења

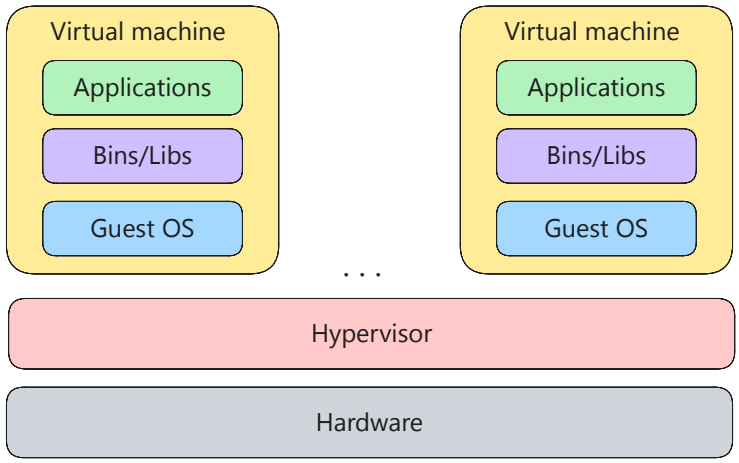
2.1.1 Виртуелизација

Виртуелизација је технологија која омогућава креирање више независних логичких окружења на једној физичкој машини. Омогућава да се један физички рачунар „подели“ на више изолованих система који функционишу као да имају сопствене ресурсе. Основни системски ресурси који се виртуелизују су процесор (CPU), меморија (RAM), складиште (*storage*) и мрежни интерфејси (*network interfaces*) [2]. Основни концепти виртуелизације су виртуелне машине (*virtual machine, VM*) и хипервизори (*hipervisors*).

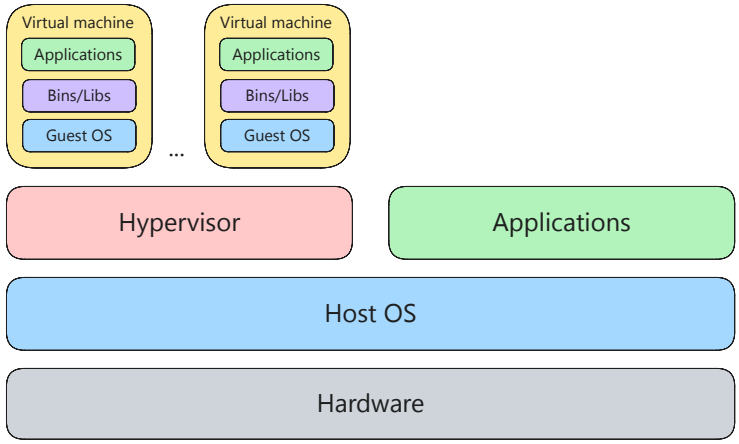
Виртуелна машина представља софтверски дефинисано изоловано окружење са сопственим оперативним системом, меморијом, процесором и мрежним интерфејсима. Ресурси виртуелне машине (*guest machine*) се добијају од физичке машине (*host machine*), али су логички издвојени тако да њихово окружење функционише независно од других виртуелних машина [3].

Хипервизор, познат и као менаџер виртуелних машина (*virtual machine monitor*, VMM), представља софтвер који омогућава деобу физичких ресурса рачунара на више виртуелних окружења. Он управља креирањем, доделом ресурса и радом виртуелних машина. Посредује између физичког хардвера (*host*) и виртуелних система (*guests*) који користе његове ресурсе.

Постоје две врсте хипервизора. Тип 1 (*native*, *bare-metal*) на слици 1, инсталира се директно на хардвер уместо оперативног система. Ресурси виртуелних машина директно се мапирају на хардвер. Овај тип је популаран у ентерпрајз окружењима и серверским системима. Тип 2 (*hosted*) на слици 2, инсталира се као апликација на постојећи *host* оперативни систем. Ресурси и инструкције виртуелних машина најпре се мапирају на *host* оперативни систем који их потом прослеђује хардверу [2], [3].



Слика 1: Хипервизор тип 1 (*native*, *bare-metal*).



Слика 2: Хипервизор тип 2 (*hosted*).

Бенефити виртуализације се огледају у:

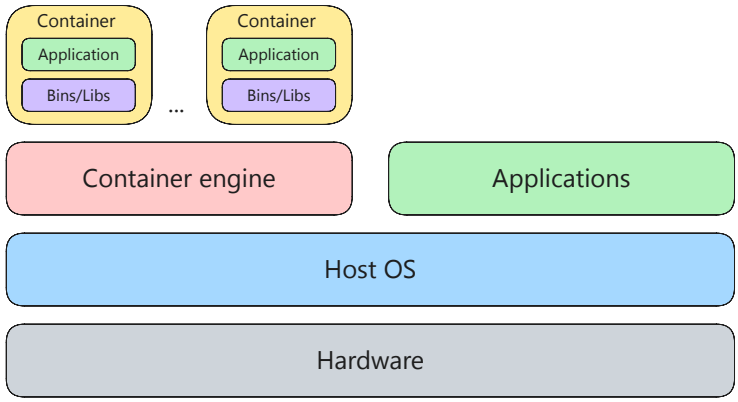
- смањењу потребне физичке инфраструктуре
- смањењу времена неисправности (*downtime*), у случају неке катастрофе или неочекиваног гашења физичког рачунара, виртуализована окружења се лакше и брже премештају и постављају на друге физичке машине
- повећању ефикасности и продуктивности тимова који одржавају инфраструктуру
- једноставнијем тестирању и креирању тест окружења која су идентична продукционим
- еколошком унапређењу и штедњи електричне енергије [4]

2.1.2 Контејнеризација

Контејнеризација представља технологију која омогућава покретање апликација у изолованим окружењима под називом контејнери (*containers*). За разлику од виртуелних машина, које емулирају цео хардвер и покрећу сопствени оперативни систем, контејнери деле језгро (*kernel*) *host* оперативног система и користе заједничке системске ресурсе, али су логички изоловани један од другог. Оваква архитектура омогућава значајно мању потрошњу ресурса, брже покретање и лакше распоређивање апликација у различитим окружењима.

Контејнер је стандардизована јединица софтвера која обједињује апликацију и све њене зависности, библиотеке, конфигурационе датотеке и системске променљиве у један преносиви пакет. Захваљујући томе, апликација ће се понашати исто без обзира на то да ли се извршава на локалном рачунару, серверу или облаку.

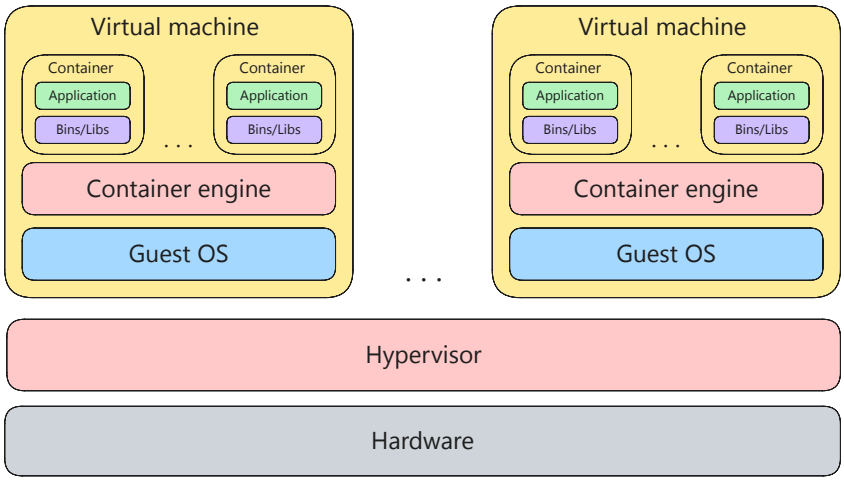
Главна предност контејнеризације у односу на класичну виртуелизацију јесте њена ефикасност. Будући да се више контејнера може покретати на једном оперативном систему без потребе за додатним хипервизором, трошак меморије и процесорских ресурса је знатно мањи, а време покретања апликација драстично краће. Сваки контејнер се извршава као посебан процес у корисничком простору (*user space*), док све виртуелне машине захтевају покретање комплетног оперативног система, што их чини знатно „тежим“ и споријим. Овакав приступ омогућава већу густину апликација по систему и једноставније оркестрирање [5].



Слика 3: Контејнеризација.

Иако представљају различите технологије, контејнери и виртуелне машине се често користе заједно, јер се њихове предности међусобно допуњују. Виртуелне машине обезбеђују пуну изолацију и сигурност на нивоу хардвера, док контејнери пружају брзину, лакоћу распоређивања и скалабилност апликација. Комбиновањем ова два приступа (слика 4) могуће је покретати више контејнеризованих

апликација унутар једне виртуелне машине, чиме се постиже већа флексибилност, боља искоришћеност ресурса и лакше управљање у хетерогеним или мулти-клауд окружењима.



Слика 4: Комбинација виртуелних машина и контејнера.

2.1.3 Инфраструктура као код (*Infrastructure as Code*)

Инфраструктура као код (*Infrastructure as Code*, IaC) представља начин аутоматизације подизања и управљања инфраструктуром. Обухвата конфигурисање, постављање (*deployment*) и управљање виртуелним окружењима на физичким машинама. Виртуелна окружења могу бити виртуелне машине или контејнери, а њихова конфигурација се описује машински читљивим кодом.

Најчешћи формати дефинисања конфигурације јесу YAML и JSON датотеке. Конфигурацијју представљају дефиниције виртуелних машина, контејнера, мрежних ресурса, складишта и зависности апликација.

IaC се може реализовати на два начина:

- декларативно (*declarative*), дефинише се жељено стање у ком систем треба да се налази, а IaC алат сам закључује како да га постигне [6] (“желим 3 сервера са следећом конфигурацијом” [7])
- императивно (*imperative*), дефинише се сет инструкција (*playbooks*) које IaC алат треба да испрати корак по корак [6] (“прво креирај сервер, онда инсталирај софтвер, па конфигуриши подешавања” [7])

Представљање инфраструктуре помоћу кода омогућава примену свих концепата који су до сада били незаобилазни у развоју софтвера, као што су верзионисање, ревизија и праћење промена, примена агилних метода, аутоматизовано тестирање и интеграција. Инфраструктурне промене је могуће третирати на исти начин као

промене у апликацијском коду, што повећава поузданост, репродуктивност и лакоћу одржавања окружења.

2.1.4 Рачунарство у облаку (*Cloud Computing*)

Рачунарство у облаку (*cloud computing*) представља испоруку рачунарских ресурса и складишних капацитета као услуге преко интернета. Релизује се тако што корисник приступа удаљеним серверима складиштеним у огромним центрима података (*data centers*) у власништву провајдера услуге. На овај начин елиминише се потреба за локалним хардвером и омогућава скалабилност, висока доступност и флексибилност коришћења ресурса.

Главна мана оваквог приступа јесте центрлизовано управљање комплетним рачунарским ресурсима (рачунарском снагом и складиштеним подацима) које доводи до следећих проблема:

- кашњење (*latency*) у трансферу и обради података, као последица удаљености између крајњег корисника и облака
- зависност од интернет конекције, без које корисник нема приступ сопственим подацима и рачунарским ресурсима у облаку
- загушеност мреже (*bandwidth load*), све популарнији паметни системи (*internet of things*, IoT), паметни аутомобили, авиони и остали уређаји који шаљу континуиране метрике на облак додатно оптерећују мрежу
- безбедносни ризици, подаци се обрађују и складиште на удаљеним серверима што отвара питање приватности и контроле
- регулаторна и правна ограничења, неке организације не смеју да складиште податке ван државе или сопственог система

2.1.5 Рачунарство на ивици (*Edge Computing*)

Ови изазови постају све израженији са растом броја повезаних уређаја и потребом за обрадом података у реалном времену, што је подстакло развој нових концепата као што је рачунарство на ивици (*edge computing*). Суштина овог приступа јесте премештање рачунарских ресурса ближе крајњим корисницима, односно изворима података који се налазе на ивици мреже (*network edge*). На тај начин се смањује кашњење у комуникацији, побољшава одзив система и омогућава ефикаснија обрада података у децентрализованом окружењу [8].

Развој рачунарства на ивици не замењује класично рачунарство у облаку, већ га допуњује и омогућава координисан рад две парадигме. Рачунарство у облаку и даље има кључну улогу у централизованој обради великих количина података, дубинској аналитици и дугорочним одлукама, док рачунарство на ивици преузима обраду података ближе извору, смањује кашњење, оптерећење мреже и омогућава

обраду у реалном времену. У пракси, чворови на ивици (*edge nodes*) могу локално обрадити и филтрирати податке, а резултате слати у облак за даљу анализу [8].

2.1.6 Микро облак (*Micro Cloud*)

У последњих неколико година све је популарније постављање мањих *data center*-а на ивици мреже. Географски распоређени сервери малих размера, организовани као микро облаци (*micro clouds*). Инфраструктура се дефинише независно од физичких машина и њихових оперативних система, применом IaC-а и концепата виртуелизације и контејнеризације. Такав приступ омогућава ефикасније коришћење ресурса чак и у хетерогеним кластерима [9].

2.2 Технологије коришћене у решењу

Алат је развијен у *Go* програмском језику. Кориснички интерфејс је реализован као CLI апликација уз употребу библиотеке *Cobra* [10], која омогућава једноставно дефинисање команди и аргумената. Архитектура система је модуларна, састоји се од централног (*core*) дела који дефинише опште интерфејсе и позадинских модула који могу бити дефинисани од стране корисника и прикључени у алат помоћу конфигурационе датотеке.

У склопу алата имплементиран је један позадински модул који користи *Vagrant* [11] и *go-vagrant* [12] библиотеку за подизање и управљање животним циклусом чворова. Конфигурациони фајл је дефинисан у YAML формату, док су скрипте за провизионисање (*provisioning*) и покретање сервиса реализоване као *Shell* скрипте. *VirtualBox* [13] хипервизор је коришћен као провајдер виртуелизације. *Control plane* и сервиси покретани на виртелним машинама су спаковани у *Docker* [14] контејнере и њихово покретање је вршено помоћу *Docker Compose* [15] алата.

2.2.1 *Cobra*

Cobra је *Go* библиотека отвореног кода за креирање CLI апликација. Омогућава дефинисање команди, подкоманди и аргумената. Подржава обавезна и опциона поља, као и логичку организацију команди у хијерархију. У алату се користи за имплементацију интерфејса који омогућава покретање и управљање виртуелним окружењима и сервисима на њима.

2.2.2 *Vagrant*

Vagrant је алат отвореног кода за аутоматизовано креирање и управљање виртуелним машинама. Омогућава дефинисање конфигурације виртуелних окружења у машински читљивим фајловима (*Vagrantfile*), а затим аутоматско подизање, провизионисање и уништавање тих окружења. Подржава различите хипервизоре као што су *VirtualBox*, *VMware* и други. Омогућава доследно и преносиво развојно окружење за више платформи .

Vagrantfile се пише у *Ruby* програмском језику. У њему се одређују основне карактеристике виртуелних окружења, као што су хипервизор, системски ресурси, мрежна подешавања и скрипте или алати за провизионирање. Такође омогућава аутоматско извршавање скрипти за инсталацију зависности и подешавања окружења. Могуће је дефинисање конфигурације за произвољан број окружења.

Vagrant долази са унапред подешеним опцијама за удаљени приступ виртуелним окружењима помоћу SSH протокола.

2.2.3 *VirtualBox*

VirtualBox је хипервизор типа 2 (*hosted*), инсталира се на *host* оперативни систем као апликација. Нуди емулацију хардвера и подршку за различите *guest* системе. Корисник може да покрене *guest* оперативни системе као независне виртуелне машине у оквиру *host*-а.

Поред основне виртуелизације, *VirtualBox* подржава више мрежних режима (NAT, *bridged*, *host-only*), дељење директоријума и пренос датотека између *host* и *guest* система, као и подршку за USB уређаје и снимке стања (*snapshots*) који омогућавају враћање виртуелне машине у претходно стање.

У оквиру овог алата, *VirtualBox* служи као хипервизор и провајдер виртуелних машина у склопу *Vagrant*-а.

2.2.4 *Docker* и *Docker Compose*

Docker је платформа за контејнеризацију. Омогућава паковање апликације и свих њених зависности у преносиви контејнер. Контејнери се извршавају из *Docker* слика (*images*), које представљају шаблоне са свим неопходним зависностима, оперативним окружењем, библиотекама и конфигурацијама за покретање апликације. *Docker* слике се дефинишу помоћу *Dockerfile* датотека. *Docker* такође подржава употребу дељених директоријума (*volumes*) за трајно складиштење података изван животног циклуса контејнера.

Docker Compose је алат који омогућава дефинисање и управљање више повезаних *Docker* контејнера као једним системом. Конфигурација се дефинише у YAML датотеци (`docker-compose.yml`), у којој се описују појединачни сервиси, њихове слике, мрежне поставке и дељени директоријуми. Овакав приступ поједностављује оркестрацију комплексних апликација које се састоје од више међусобно зависних контејнера и омогућава покретање целокупног окружења једном командом.

Коришћени су у склопу *Shell* скрипти за провизионирање, за покретање претходно докеризованих сервиса из *Constellations* пројекта. Такође се користе и за покретање *Control plane*-а на *host* машини.

2.3 Постојећа решења

У овом поглављу је обрађено неколико неколико постојећих алата који омогућавају аутоматизовање покретања и управљање виртуелним окружењем. Ниједно од предложених решења не одговара на функционалне захтеве проблема у целости или то ради на сувише комплексан начин.

2.3.1 *Terraform*

Terraform [16] је IaC алат који омогућава декларативно управљање инфраструктуром. Користи сопствени језик HCL (*HashiCorp Configuration Language*) за опис жељеног стања система, што омогућава да алат сам израчуна кораке потребне за постизање тог стања. *Terraform* може да креира и управља ресурсима на различитим провајдерима, као што су платформе у облаку (AWS, *Azure*, GCP) или локални провајдери као што је *VirtualBox*.

2.3.2 *Ansible*

Ansible [17] је IaC алат за аутоматизацију конфигурације и управљање системима. Користи декларативне YAML *playbook*-ове за дефинисање корака које треба извршити на једном или више *host*-ова, као што су инсталација софтвера, копирање фајлова и покретање скрипти. *Ansible* се ослања на SSH за повезивање са *host*-овима и не захтева инсталиран агент на њима, што га чини погодним за управљање постојећим серверима или виртуелним машинама.

Глава 3

Архитектура система

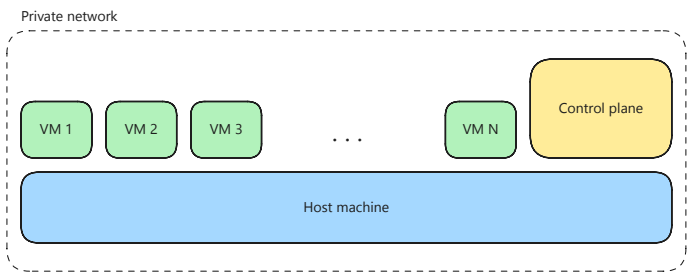
Ово поглавље приказује преглед функционалних захтева система, као и концептуални модел и архитектуру решења које их остварује. Решење је реализовано као CLI алат који се компајлира у једну бинарну датотеку и на тај начин пружа сет команди за интеракцију са самим алатом.

3.1 Преглед функционалних захтева

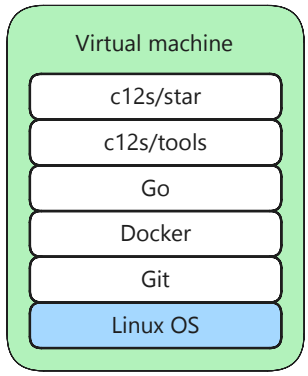
Потребно је креирати алат за покретање *N* виртуелних машина одједном. Свака виртуелна машина треба да има инсталирану лагану (*lightweight*) дистрибуцију *Linux* оперативног система. Конфигурација мреже самих машина треба да буде таква да машине буду међусобно видљиве, да виртуелне машине вуду видљиве са *host* машине, као и да *host* машина буде видљива из виртелних машина (слика 5).

На свакој машини је потребно инсталирати систем за контролу верзија (*Version Control System*, VCS) *Git* [18], *Docker* и *Go* програмски језик, који представљају основно развојно окружење за компоненте *Constellations* пројекта. Потребно је клонирати `c12s/tools` [19] репозиторијум који садржи опште алате у оквиру пројекта *Constellations*. Након тога потребно је покренути инсталационе скрипте из `c12s/tools` репозиторијума и подесити параметре окружења како би се инсталирали неопходни сервиси. Након инсталације потребно је покренути `c12s/star` [20] сервис на свакој машини (слика 6).

На *host* машини је потребно инталирати и покренути *control plane* која обухвата скуп сервиса у склопу *Constellations* пројекта који се такође ослањају на употребу `c12s/tools` репозиторијума. Након покретања виртуелних машина и подизања неопходне инфраструктуре тестирање се врши путем `c12s/cockpit` [21] алата који се налази у оквиру *control plane*-а.



Слика 5: Инфраструктура локалног развојног окружења.



Слика 6: Поставка система на виртуелној машини.

Глава 4

Имплементација система

Глава 5

Закључак

У закључку дајте кратак преглед онога шта урађено, са освртом на проблеме који су решени, предности и мане решења и правце даљег развоја.

Списак слика

Списак листинга

Списак коришћених скраћеница

Скраћеница	Опис
API	Application Programming Interface (апликациони програмски интерфејс)
AWS	Amazon Web Services (Амазон веб сервиси)
CI/CD	Continuous Integration / Continuous Delivery (континуирана интеграција / континуирана испорука)
CORS	Cross-Origin Resource Sharing (размена ресурса између извора и дестинације различитог порекла)
CSS	Cascading Style Sheets (језик за описивање стилова)
DOM	Document Object Model (објектни модел документа)
DTO	Data Transfer Object (објекат за пренос података)
HTTP	HyperText Transfer Protocol (протокол за пренос хипертекста)
JSON	JavaScript Object Notation (формат за размену података)
JWT	JSON Web Token (сигурносни токен заснован на JSON формату)
RLS	Row-Level Security (сигурност на нивоу реда)
REST	Representational State Transfer (скуп правила за комуникацију између клијента и сервера)
RPC	Remote Procedure Call (позив удаљене процедуре)
SQL	Structured Query Language (структурирани упитни језик)
TLS	Transport Layer Security (безбедност транспортног слоја)
UML	Unified Modeling Language (језик за моделовање дијаграма)
URL	Uniform Resource Locator (јединствени идентификатор и локатор ресурса)
UI	User Interface (кориснички интерфејс)
UUID	Universally Unique Identifier (универзално јединствени идентификатор)
WAL	Write-Ahead Logging (записивање операција унапред)

Списак коришћених појмова

Појам	Објашњење
Асихнрони рад	Рад који се изводи независно од главног тока извршавања, омогућавајући наставак других операција без чекања на његов завршетак.
Bucket (S3)	Логичка јединица за складиштење у AWS S3 сервису, која организује фајлове у облаку.
Read-Only	Режим рада у коме су подаци само за читање, без могућности измене.
Cross-platform	Способност софтвера да се извршава на више различитих оперативних система из истог кода.
Connection pool	Механизам за управљање и поновну употребу веза са базом података како би се побољшале перформансе апликације.

Биографија

Раде Пејановић рођен је 2002. године у Новом Саду. Првих шест разреда основне школе завршио је у ОШ „Јован Дучић“ у Петроварадину, а преостала два у ОШ при Гимназији „Јован Јовановић Змај“ у Новом Саду. Паралелно је похађао и нижу музичку школу „Јосип Славенски“ у Новом Саду, одсек кларинет.

Средњошколско образовање наставио је у Гимназији „Јован Јовановић Змај“ на смеру за ученике са посебним способностима за физику. За изузетан успех у основној и средњој школи добио је Вукову диплому.

Факултет техничких наука у Новом Саду, смер Софтверско инжењерство и информационе технологије, уписао је 2021. године и успешно завршио све испите предвиђене студијским програмом.

Овим радом завршава основне академске студије на Факултету техничких наука у Новом Саду.

Литература

[1] c12s, „c12s — GitHub Organization“. 2025.

[2] R. Hat, „What is Virtualization?“. Приступљено: 05. Новембар 2025. [На Интернету]. Доступно на https://www.redhat.com/en/topics/virtualization/what-is-virtualization

[3] AWS, „What is Virtualization?“. Приступљено: 05. Новембар 2025. [На Интернету]. Доступно на https://aws.amazon.com/what-is/virtualization/

[4] J. Shamir, „The Benefits of Virtualization“. [На Интернету]. Доступно на https://www.ibm.com/think/insights/virtualization-benefits

[5] Docker, „What is a Container?“. Приступљено: 05. Новембар 2025. [На Интернету]. Доступно на https://www.docker.com/resources/what-container/

[6] C. Feng, „Imperative vs Declarative IaC: Ansible and Terraform Insights“. Приступљено: 06. Новембар 2025. [На Интернету]. Доступно на https://www.casperfeng.com/blog/imperative-vs-declarative-iac-ansible-and-terraform-insights

[7] J. Holdsworth и A. Badman, „What Is Infrastructure as Code (IaC)?“. [На Интернету]. Доступно на https://www.ibm.com/think/topics/infrastructure-as-code

[8] K. Cao, Y. Liu, G. Meng, и Q. Sun, „An Overview on Edge Computing Research“, *IEEE Access*, том 8, стр. 85714–85728, 2020, doi: 10.1109/ACCESS.2020.2991734.

[9] M. Simić, G. Sladić, M. Zarić, и B. Markoski, „Infrastructure as Software in Micro Clouds at the Edge“, *Sensors*, том 21, изд. 21, стр. 7001, 2021, doi: 10.3390/s21217001.

[10] spf13, „Cobra: A Commander for modern Go CLI interactions“. 2025.

[11] HashiCorp, „Vagrant – Development Environments Made Easy“. [На Интернету]. Доступно на https://developer.hashicorp.com/vagrant

[12] B. Matcuk, „go-vagrant: A golang wrapper around the Vagrant command-line utility“. 2021.

[13] O. Corporation, „Oracle VM VirtualBox — Powerful open source virtualization“. [На Интернету]. Доступно на https://www.virtualbox.org/

[14] I. Docker, „Docker — Accelerate how you build, share, and run applications“. [На Интернету]. Доступно на https://www.docker.com/

[15] I. Docker, „Docker Compose Documentation“. [На Интернету]. Доступно на https://docs.docker.com/compose/

[16] HashiCorp, „Terraform — Automate Infrastructure on Any Cloud“. [На Интернету]. Доступно на https://developer.hashicorp.com/terraform

[17] A. Community, „Ansible Documentation“. [На Интернету]. Доступно на https://docs.ansible.com/

[18] G. Community, „Git — Distributed version control system (official site)“. Приступљено: 06. Новембар 2025. [На Интернету]. Доступно на https://git-scm.com/

[19] c12s, „tools: Scripts and utilities used by the c12s organization“. 2025.

[20] c12s, „star: A Go module by c12s (GitHub repository)“. 2024.

[21] c12s, „cockpit: A CLI tool that communicates with the Constellations gateway“. 2024.