



УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

---



Нина Милановић

**Примена Ансибле алата у решавању  
проблема управљања конфигурацијом и  
испоруком интернет апликација**

ДИПЛОМСКИ РАД  
- Основне академске студије -

Нови Сад, 2021.



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

## КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, <b>РБР:</b>	
Идентификациони број, <b>ИБР:</b>	
Тип документације, <b>ТД:</b>	Монографска документација
Тип записа, <b>ТЗ:</b>	Текстуални штампани материјал
Врста рада, <b>ВР:</b>	Завршни (Bachelor) рад
Аутор, <b>АУ:</b>	Нина Милановић
Ментор, <b>МН:</b>	др Петар Марић, доцент
Наслов рада, <b>НР:</b>	Примена Ансибле алата у решавању проблема управљања конфигурацијом и испоруком интернет апликација
Језик публикације, <b>ЈП:</b>	Српски / латиница
Језик извода, <b>ЈИ:</b>	Српски
Земља публикавања, <b>ЗП:</b>	Република Србија
Уже географско подручје, <b>УГП:</b>	Војводина
Година, <b>ГО:</b>	2021.
Издавач, <b>ИЗ:</b>	Ауторски репринт
Место и адреса, <b>МА:</b>	Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, <b>ФО:</b> (поглавља/страна/ цитата/табела/слика/графика/прилога)	7/42/1/0/25/0/9
Научна област, <b>НО:</b>	Електротехника и рачунарство
Научна дисциплина, <b>НД:</b>	Примењене рачунарске науке и информатика
Предметна одредница/Кључне речи, <b>ПО:</b>	Управљање конфигурацијом, инфраструктура као код, испорука интернет апликација
<b>УДК</b>	
Чува се, <b>ЧУ:</b>	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, <b>ВН:</b>	
Извод, <b>ИЗ:</b>	Ток напретка рачунарских система и њихове архитектуре, у данашње време, усмерио је многе организације да у употребу укључе јако велики број сервера, како у сврхе самог пружања услуга све већем броју корисника, тако и у сврхе развијања својих производа. Данас је обезбеђивање оваквих сложених система један од круцијалних задатака, који захтева исцрпно знање као и доста искуства у администрацији система. Тимови који се баве оспособљавањем оваквих система треба да превазиђу изазове као што су планирање, постављање, а затим и одржавање оваквих система. Процес инжењерства система који се бави описаним проблемима назива се управљање конфигурацијом. Овај рад ће се фокусирати на Ансибле алат, који је широко прихваћен и примењиван у сврхе постављања окружења, менаџмента конфигурација и испоруке софтвера.
Датум прихватања теме, <b>ДП:</b>	
Датум одбране, <b>ДО:</b>	
Чланови комисије, <b>КО:</b>	Председник: др Срђан Попов, ванредни професор
	Члан: др Милан Вртунски, асистент са докторатом
	Члан, ментор: др Петар Марић, доцент
	Потпис ментора



## KEY WORDS DOCUMENTATION

Accession number, <b>ANO</b> :			
Identification number, <b>INO</b> :			
Document type, <b>DT</b> :	Monographic publication		
Type of record, <b>TR</b> :	Textual printed material		
Contents code, <b>CC</b> :	Bachelor Thesis		
Author, <b>AU</b> :	Nina Milanović		
Mentor, <b>MN</b> :	Asst. Prof. Petar Marić, PhD		
Title, <b>TI</b> :	Utilization of Ansible tool for solving problems of configuration management and web application deployment		
Language of text, <b>LT</b> :	Serbian		
Language of abstract, <b>LA</b> :	Serbian		
Country of publication, <b>CP</b> :	Republic of Serbia		
Locality of publication, <b>LP</b> :	Vojvodina		
Publication year, <b>PY</b> :	2021.		
Publisher, <b>PB</b> :	Author's reprint		
Publication place, <b>PP</b> :	Novi Sad, Dositeja Obradovica sq. 6		
Physical description, <b>PD</b> : (chapters/pages/ref./tables/pictures/graphs/appendixes)	7/42/1/0/25/0/9		
Scientific field, <b>SF</b> :	Electrical Engineering		
Scientific discipline, <b>SD</b> :	Applied Computer Science and Informatics		
Subject/Key words, <b>S/KW</b> :	Configuration management, infrastructure as code, deployment of web application		
<b>UC</b>			
Holding data, <b>HD</b> :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia		
Note, <b>N</b> :			
Abstract, <b>AB</b> :	<p>The evolution of computer systems and their architecture, nowadays, has influenced many organizations to use a large number of servers, both for the purpose of providing services to a growing number of users, and for the purpose of developing their products. Today, providing such complex systems is one of the crucial tasks, which requires comprehensive knowledge as well as a lot of experience in system administration. Teams involved in management of these systems need to overcome challenges such as planning, setting up, and then maintaining such systems. The system engineering process that deals with the described problems is called configuration management. This paper will focus on the Ansible tool, which is widely accepted and applied for the purposes of setting up the environment, managing configurations, and delivering software.</p>		
Accepted by the Scientific Board on, <b>ASB</b> :			
Defended on, <b>DE</b> :			
Defended Board, <b>DB</b> :	President:	Assoc. Prof. Srđan Popov, PhD	Mentor's sign
	Member:	Asst. Prof. Milan Vrtunski, PhD	
	Member, Mentor:	Asst. Prof. Petar Marić, PhD	

## SADRŽAJ

1. Uvod .....	1
2. Infrastruktura kao kod .....	3
2.1 Definicija.....	3
2.2 Deklarativni i imperativni pristup .....	4
2.3 Problemi koje prevazilazi .....	5
2.4 Benefiti korišćenja IaC principa .....	5
2.5 Alati koji su omogućili IaC princip .....	6
3. Ansible.....	8
3.1 Uvod u Ansible .....	8
3.2 Istorijat .....	10
3.3 Arhitektura .....	10
3.4 Osnovni koncepti .....	12
3.4.1 Kontrolni čvor .....	12
3.4.2 Upravljeni čvor .....	13
3.4.3 Inventar.....	13
3.4.4 Kolekcije.....	14
3.4.5 Moduli .....	15
3.4.6 „Playbook“ .....	17
3.5 Ostali koncepti .....	19
3.5.1 „Ansible Galaxy” .....	19
3.5.2 „Ansible Tower” .....	20

---

3.6	Slučajevi korišćenja .....	21
3.7	Prednosti i nedostaci .....	23
4.	Studija slučaja.....	25
4.1	Korišćeni alati .....	25
4.2	Arhitektura postavljenog sistema.....	26
4.3	Primena Ansible koncepata.....	28
4.4	Rezultati istraživanja.....	36
5.	Zaključak .....	37
6.	Literatura .....	38
7.	Prilog .....	40

**SPISAK SLIKA**

Slika 2.1 .....	6
Slika 3.1 .....	9
Slika 3.2 .....	12
Slika 3.3 .....	13
Slika 3.4 .....	14
Slika 3.5 .....	15
Slika 3.6 .....	16
Slika 3.7 .....	16
Slika 3.8 .....	16
Slika 3.9 .....	18
Slika 3.10 .....	20
Slika 4.1 .....	26
Slika 4.2 .....	27
Slika 4.3 .....	28
Slika 4.4 .....	28
Slika 4.5 .....	29
Slika 4.6 .....	30
Slika 4.7 .....	31
Slika 4.8 .....	32
Slika 4.9 .....	32
Slika 4.10 .....	33

---

Slika 4.11 .....	34
Slika 4.12 .....	35
Slika 4.13 .....	35
Slika 4.14 .....	36

## 1. Uvod

Tok napretka računarskih sistema i njihove arhitekture, u današnje vreme, usmerio je mnoge organizacije da u upotrebu uključe jako veliki broj servera, kako u svrhe samog pružanja usluga sve većem broju korisnika, tako i u svrhe razvijanja svojih proizvoda. Svi ti serveri imaju različite uloge u sistemu organizacije i odgovaraju na različite potrebe. Kao rezultat ovakvog razvoja situacije, javila se i potreba da sistemski administratori poseduju rešenje za efikasno manipulisanje ovim serverima, kao i uslugama instanciranim na njima.

Danas je obezbeđivanje ovakvih složenih sistema jedan od ključnih zadataka, koji zahteva iscrpno znanje kao i dosta iskustva u administraciji sistema. Timovi koji se bave osposobljavanjem ovakvih sistema treba da prevaziđu izazove kao što su planiranje, postavljanje, a zatim i održavanje ovakvih sistema. Nakon pažljivo isplanirane infrastrukture i pripremljenog hardvera za upotrebu, sistemski administratori započinju svoj posao. Njihov posao obuhvata postavljanje i konfiguracije servisa i alata potrebnih za pravilan rad sistema, a sve to imajući na umu probleme kao što su dostupnost sistema, skalabilnost, bezbednost i pouzdanost. Osim navedenog, potrebno je obezbediti i alate za nadzor, pravljenje rezervnih kopija stanja sistema i njegovih podataka, kao i beleženje izvršenih akcija na sistemu.

Još jedan od izazova, sa kojim se sistemski administratori koji upravljaju velikim brojem servera sa modernom arhitekturom suočavaju, predstavlja kontinuirano postavljanje identičnih servisa i alata i njihovih konfiguracija na više gotovo identičnih servera.

Na primer, veliki broj sistema u oblaku zahteva više identičnih servera baza podataka, kako bi se postigle bolje performanse i uravnotežilo opterećenje servera. U takvim slučajevima, nudi se šansa za optimizaciju samog procesa ručne konfiguracije brojnih, gotovo istih postavki. Uobičajena rešenja bazirala su se na pisanju skripti. Ovakvo rešenje je doводilo do više problema. Prvenstveno, zahtevalo je od sistemskih administratora napredne veštine pisanja skripti, zatim je bilo potrebno i dostaviti uputstva za samu primenu skripti, a kako ne postoje



strogo definisane konvencije kodiranja skripti, njihova struktura je zavisila od preferencija samog stručnjaka. Takođe, dodatni problem predstavljao je kontrolu verzija ovakvih konfiguracionih datoteka.

Proces inženjerstva sistema koji se bavi opisanim problemima naziva se upravljanje konfiguracijom. Razvijeno je više rešenja otvorenog koda za upravljanje konfiguracijama, koja se široko koriste. Ovo istraživanje će se fokusirati na Ansible alat, koji je široko prihvaćen i primenjivan u mnogim prominentnim organizacijama.

Ovaj rad sastoji se iz pet poglavlja. U narednom poglavlju biće bliže opisan koncept infrastrukture kao koda, koji je jedno od najčešće primenjivanih rešenja za upravljanje konfiguracijom. Treće poglavlje detaljno analizira sam Ansible alat. Nakon osvrta na njegov razvoj, biće detaljno obrađeni najbitniji koncepti i arhitektura jednog Ansible sistema. Nadalje, u ovom poglavlju su predstavljani su kako opšti tako i realni slučajevi primene alata, a kao epilog poglavlja biće izložen osvrt na prednosti i mane Ansible alata.

Četvrto poglavlje predstavlja opis studije slučaja koja demonstrira primenu ovog alata u svrhe „deployment-a“ aplikacije. Pruža detaljan opis arhitekture i njene postavke, a, potom, i realne primene koncepata opisanih u trećem poglavlju. Peto poglavlje predstavlja zaključak ovog rada, kao i predlog za dalja istraživanja.

## 2. Infrastruktura kao kod

U prošlosti je upravljanje IT infrastrukturom bilo iscrpan posao. Administratori sistema morali su ručno da upravljaju i konfigurišu sav hardver i softver koji su bili potrebni za pokretanje aplikacija. Ovo je uključivalo konfiguraciju velikog broja mašina, različitih operativnih sistema i alata kako bi razvojno okruženje bilo u skladu sa potrebama organizacije. Međutim, poslednjih godina prakse su se znatno promenile. Trendovi poput računarstva u oblaku (eng. „Cloud Computing”) revolucionisali su i unapredili način na koji kompanije i organizacije dizajniraju, razvijaju i održavaju svoju IT infrastrukturu.

Jedan od ključnih aspekata ovog trenda naziva se „infrastruktura kao kod (eng. „Infrastructure as Code”)“ i u ovom poglavlju će biti napravljen osvrt na nju, probleme koje rešava, izazove na koje nailazi i benefite koje pruža.

### 2.1 Definicija

Infrastruktura kao kod (u daljem tekstu IaC) se definiše kao proces upravljanja infrastrukturom i obezbeđivanja infrastrukture putem mašinski čitljivih datoteka definicija (odnosno koda), umesto fizičke konfiguracije hardvera ili interaktivnih alata za konfiguraciju.

Jedan od važnih principa IaC-a je idempotencija. Idempotencija se odnosi na svojstvo koje nalaže da naredba za „deployment<sup>1</sup>” uvek ciljno okruženje postavlja u istu konfiguraciju, bez obzira na prethodno stanje okruženja. Idempotencija se postiže ili automatskim

---

<sup>1</sup> „Deployment” predstavlja skup akcija koje je potrebno preduzeti da bi se softver doveo u stanje dostupno za korišćenje

konfigurisanjem ciljnog okruženja, ili odbacivanjem postojećeg i ponovnim stvaranjem novog okruženja, u željenoj konfiguraciji.

Pomoću IaC-a kreiraju se konfiguracione datoteke koje sadrže specifikacije infrastrukture, što olakšava uređivanje i distribuciju konfiguracija. Takođe, osigurava da se svaki put iznova pruža isto okruženje, sa doslednom konfiguracijom. Sve ovo se postiže kodiranjem i dokumentovanjem konfiguracionih specifikacija, prilikom čega se izbegavaju promene konfiguracija direktno iz komandne linije, koje neće biti dokumentovane.

Potrebno je staviti akcenat na to da veliki deo IaC-a predstavlja i kontrola verzionisanja koda. Sve konfiguracione datoteke treba da budu pod kontrolom izvora, kao i bilo koja druga datoteka izvornog koda softvera. Dakle, potrebno je datoteke koje opisuju konfiguracije okruženja tretirati kao sastavni deo koda, gde će biti moguće imati uvid svaku načinjenu promenu samih konfiguracionih datoteka.

## 2.2 Deklarativni i imperativni pristup

Postoje dva pristupa IaC principu. Prvi je deklarativni, odnosno funkcionalni, a drugi pristup je imperativni, to jest proceduralni. Tokom izbora pristupa infrastrukturi kao kodu, neophodno je razumeti razliku između ova dva pristupa.

Prilikom primene deklarativnog pristupa navodi se konačno željeno stanje infrastrukture koju je potrebno obezbediti, uključujući i resurse i svojstva koja su neophodna za željeno okruženje. Uloga IaC alata se potom ogleda u instalaciji neophodnog softvera, podešavanja željene konfiguracije, razrešavanja međuzavisnosti softvera i sistema, i upravljanje verzijama.

Nasuprot deklarativnom, imperativni pristup definiše konkretne naredbe koje je potrebno izvršiti za postizanje željenog stanja okruženja. Naredbe je, takođe, potrebno izvršiti u tačno određenom redosledu.

Može se reći da deklarativni pristup odgovara na pitanje „Šta?“, dok imperativni pristup odgovara na pitanje „Kako?“. Deklarativni pristup je fokusiran na to kakva bi ciljna konfiguracija trebala biti, dok je imperativni pristup fokusiran na konkretne korake koje treba preduzeti da bi se ciljna konfiguracija postigla.

Iako IaC alati često mogu funkcionisati u oba pristupa, postoji tendencija preferiranja jednog pristupa u odnosu na drugi. Međutim, u većini organizacija, kao najpogodnije rešenje se bira deklarativni pristup, gde se definiše željena konfiguracija, a alat pruža infrastrukturu automatski.

## 2.3 Problemi koje prevazilazi

Infrastruktura kao kod evoluirala je da bi rešila „envriomental drift” problem u toku distribucije softvera. „Envriomental drift” *problem*, odnosno, problem odstupanja u konfiguraciji okruženja, javlja se kada dođe do promene u produkcijskom okruženju, a bez evidentiranja tih promena i bez obezbeđivanja potpune usklađenosti između „staging<sup>2</sup>” i „production<sup>3</sup>” okruženja.

Do nastanka problema najčešće dolazi ukoliko neko, čak i nenamerno, izmeni verziju nekog alata iz komandne linije. Osim što to može rezultirati problemima pri „deployment”-u, zbog neusklađenosti verzija u različitim okruženjima, ovakve situacije dovode i do korišćenja neproverenih verzija koje mogu imati sigurnosne ranjivosti, što dalje izlaže rizicima razvoj aplikacija i usluga koji moraju da zadovolje stroge standarde usklađenosti sa propisima.

Takođe, direktnim izmenama konfiguracije iz komandne linije, vremenom, svako okruženje postaje pahuljica (eng. „snowflake”), odnosno jedinstvena konfiguracija koja se ne može automatski reprodukovati. Neusklađenost okruženja dovodi do problema tokom puštanja proizvoda u produkciju. Kod pahuljica, administracija i održavanje infrastrukture uključuju ručne procese koji su gotovo nemogući za dokumentovati, doprinose greškama i zahtevaju dodatni utrošak vremena i resursa.

## 2.4 Benefiti korišćenja IaC principa

Automatizacija procesa postavljanja okruženja donosi sa sobom mnoge prednosti i zaista je pravi izazov predočiti svaku od njih. Samo neki od benefita korišćenja IaC principa su:

1. Pojednostavljuje proces postavljanja i konfiguracije okruženja, a automatizacijom ovog procesa se ostavlja jako malo prostora greškama, koje se prilikom manuelnog procesa mogu mnogo češće napraviti.
2. Brži i efikasniji razvoj softvera i kraće vreme za puštanje proizvoda u produkciju. IaC automatizacija dramatično ubrzava proces postavljanja infrastrukture potrebne za različite faze izrade proizvoda, poput razvoja, testiranja i produkcije, kao i proces neophodnog skaliranja proizvoda u produkciji. Budući da znatno pojednostavljuje

---

<sup>2</sup> „Staging” okruženje predstavlja ono okruženje u kojem se sadržaj kreira i modifikuje

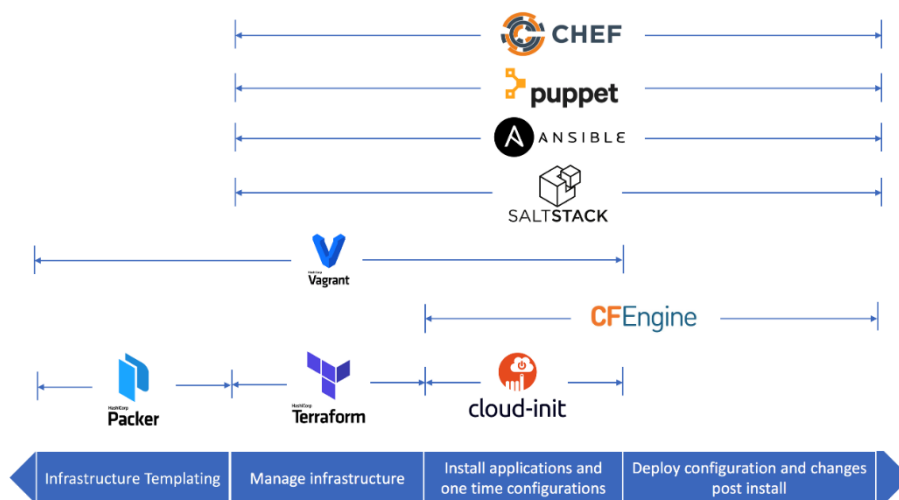
<sup>3</sup> „Production” okruženje predstavlja ono okruženje u kome se sadržaj prezentuje

proces postavljanja okruženja, i obezbeđuje njihovu konzistentnost, infrastruktura kao kod može velikim udelom ubrzati svaku fazu životnog ciklusa softverskog proizvoda.

3. Povećana doslednost, odnosno rešavanje problema odstupanja u konfiguraciji. Kao što je već napomenuto promene konfiguracije iz komandne linije, kao i ažuriranja softverskih paketa dovode do neusklađenosti okruženja za razvoj, testiranje i produkciju, odnosno do „environmental drift” problema. Obezbeđivanjem identičnog okruženja svaki put, IaC sprečava odstupanja u konfiguraciji.
4. Zaštita od odliva. Kada su u pitanju organizacije koje ne primenjuju IaC, postavljanje okruženja je obično povereno nekoliko kvalifikovanih inženjera, odnosno IT stručnjaka. Ukoliko bi se desilo da jedan od ovih stručnjaka napusti organizaciju, ostalima će neretko biti prepušteno da rekonstruišu ceo ovaj proces. U ovom scenariju IaC osigurava da proces postavljanja okruženja ostane dokumentovan u toj organizaciji, time što će ostati intelektualna svojina organizacije. Takođe, neće biti neophodno ni prenošenje znanja vezanog za čitav manuelni proces postavljanja okruženja, prilikom odlaska zaposlenog kome je upravljanje okruženjem bilo povereno.
5. Snižavanje troškova i poboljšani povraćaj ulaganja. Ono što karakteriše korišćenje IaC u organizacijama je dolazak do znatnog smanjenja vremena, napora i potreba za specijalizovanim veštinama koje su neophodne za postavljanje i skaliranje infrastrukture. Kada je u pitanju poslovanje putem računarstva u oblaku, smanjenjem vremena neophodnog za konfiguraciju okruženja, IaC omogućava maksimalno korišćenje „plati koliko potrošiš” principa. Takođe, automatizacija postavljanja okruženja omogućava osoblju koje se bavi razvojem softvera da se fokusira na razvoj inovativnih i ključnih softverskih rešenja, oslobađajući ih od manualnih, ponavljajućih procesa koji konzumiraju vreme.

## 2.5 Alati koji su omogućili IaC princip

Dostupan je veliki broj alata koji ispunjavaju mogućnosti automatizacije infrastrukture i koriste IaC. Uopšteno govoreći, bilo koji okvir ili alat koji izvršava promene ili konfigurise infrastrukturu deklarativno ili imperativno na osnovu programskog pristupa može se smatrati IaC. Tradicionalno su se za postizanje IaC koristili alati za automatizaciju servera i upravljanje konfiguracijom. Međutim razvijana su i posebna rešenja za primenu IaC koncepta.



Slika 2.1.

Postoji čitav niz rešenja koja mogu da odgovore na različite zahteve korisnika i pre odlučivanja za jedan od alata potrebno je dobro definisati potrebe, a zatim uvideti koji alat bi na njih najbolje odgovorio. Slika 2.1. prikazuje vodeće alate i domen njihovih funkcionalnosti, raspoređenih u spektar koji predstavlja prostor problema kojim se bave. Alati sa leve strane spektra se fokusiraju na stvaranje i upravljanje infrastrukturnim resursima, dok su alati na desnoj strani fokusirani na upravljanje konfiguracijama.

Primer radi, Packer i Vargant alati omogućavaju kreiranje predefinisanih šablona infrastrukture virtuelnih mašina, dok je CFEngine specijalizovan isključivo za brzo i lako upravljanje konfiguracijom. Alati koji se fokusiraju na upravljanje konfiguracijama, instalaciju paketa, kao i „deployment“ aplikacija, uključuju Puppet, Chef, SaltStack, kao i Ansible, koji je upravo i centralna tema ovog rada.

Navedeni alati mogu se koristiti i u kombinaciji, gde bi, na primer, Packer bio korišćen za definisanje šablona infrastrukture koju želimo da koristimo, a uz njega bi za postavljanje okruženja bio korišćen Puppet (ili neki drugi alat iste namene), koji bi omogućio instalaciju i konfiguraciju softverskih paketa.

## 3. Ansible

I dalje je vrlo česta praksa u kojoj se sistemski administratori oslanjaju na ručne procese i prilagođene skripte kako bi izvršavali zadatke koji se neretko ponavljaju. Kada se ove metode primene na dinamično serversko okruženje sa heterogenom infrastrukturom koja podrazumeva različite operativne sisteme, različite alate, i konfiguracije, ove metode je teško prilagoditi i održavati. Dodatno, ceo proces se komplikuje i dolazi do većoj podložnosti greškama, efikasnost se smanjuje, a optimizacija nije postignuta u dovoljnoj meri. Da bi se suočili sa ovim problemima dostupni su i sve više se koriste alati za upravljanje konfiguracijom.

Upravo ove probleme rešava princip infrastrukture kao koda, opisan u prethodnom poglavlju, a jedan od, definitivno, najzastupljenijih alata za primenu ovog principa je Ansible.

### 3.1 Uvod u Ansible

Ansible je alat otvorenog koda, odnosno platforma za automatizaciju, koji omogućava ostvarenje principa infrastrukture kao koda. Koristi se za upravljanje konfiguracijom, „deployment” aplikacija, orkestraciju između različitih servisa i postavljanje okruženja.

Ansible je dostupan za korišćenje na većini Unix-like operativnih sistema, a omogućava konfiguraciju kako Unix-like operativnih sistema, tako i Microsoft Windows operativnih sistema. Za konfiguraciju sistema, ovaj alat poseduje sopstveni deklarativni jezik.

Glavni ciljevi samog dizajna ovog alata uključuju sledeće osobine:

1. Minimalizam. Sistemi upravljanja konfiguracijom ne bi trebalo da nameću dodatne zavisnosti u okruženje.
2. Doslednost. Uz Ansible alat je neophodno omogućiti stvaranje okruženja koja su konzistentna.

3. Bezbednost. Ansible ne raspoređuje agente na čvorove, a na upravljanim čvorovima potrebni su samo OpenSSH i Python paketi.

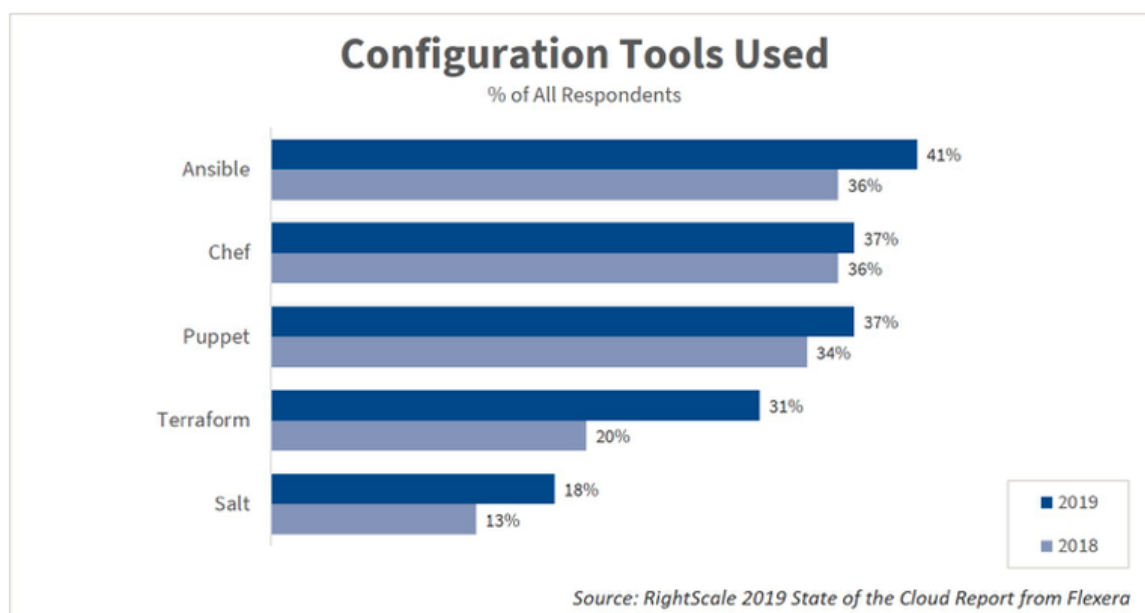
4. Pouzdanost. Kada se pažljivo napiše, Ansible skripta može biti idempotentna kako bi se sprečili neočekivani neželjeni efekti na upravljane sisteme.

5. Potrebno minimalno učenje. Za pisanje Ansible skripti potrebno je znanje jasnog i opisnog jezika zasnovanog na YAML sintaksi i Jinja šablonima.

Sa sve bržim razvojem računarstva u oblaku, koje omogućava gotovo trenutno dobavljanje računarskih resursa, raste i potreba za automatizacijom okruženja, kako bi se efikasnost podigla na najviši mogući nivo, a utrošeno vreme i materijalni troškovi smanjili. Upravo ove okolnosti dovode i do sve veće popularnosti alata kao što je Ansible, koji se zbog mnogih prednosti (o kojima će biti reči kasnije) ističe među alatima slične namene.

Istraživanje „The Flexera State of the Cloud Report”, koje se bavi obradom statističkih podata vezanih za računarstvo u oblaku, svake godine dostavlja izveštaj o popularnosti različitih trendova u ovom domenu računarstva. U izveštaju iz 2019. godine, Ansible se navodi kao najzastupljeniji alat koji se koristio za konfiguraciju okruženja u oblaku, a o tome svedoči i slika 3.1, preuzeta iz izveštaja.

Ono što je zanimljivo je da je zastupljenost Ansible alata u odnosu na prethodnu, 2018.



Slika 3.1

godinu, porasla sa 36% na 41%, dok porast drugoplasiranog Chef alata iznosi samo jedan procenat.



## 3.2 Istorijat

Kovanica „ansible“ prvi put se pojavila u naučno-fantastičnom romanu Ursule K. Le Guin pod nazivom „Rokanonov svet“. U ovom romanu, „ansible“ se odnosi na fiktivne komunikacione sisteme koji prenose poruke momentalno, nezavisno od smetnji i udaljenosti, čak i između zvezdanih sistema. Od tada se taj termin široko koristi u radovima brojnih autora naučne fantastike, odnoseći se na komunikacione sisteme koji delaju u trenutku.

Upravo na ovim terminom se autor Ansible alata vodio prilikom njegovog razvoja. Razvijan u Python programskom jeziku, Ansible je prvi put predstavljen javnosti u februaru 2012. godine, razvijan od strane Majkl Dehana, softverskog inženjera koji je u kompaniji Redhat bio zadužen na projektima poput Cobbler i Func alata za administraciju udaljenih mašina.

Sam autor o motivaciji za nastanak ovog alata kaže: „Iako je deo razvoja Ansible alata bio kako bih pokazao svetu da postoji jednostavniji način“ ... „najbitnije je bilo kreirati alat koji (A) sam ja želeo da koristim, i (B) koji se mogao prestati koristiti na šest meseci, a zatim mu se vratiti i i dalje ga se sećati“<sup>4</sup>. Takođe, Dehan navodi da on, kao inženjer softvera, nije želeo da provodi polovinu svog vremena „boreći se“ sa alatima za automatizaciju i, istovremeno, nameravao je da unapredi IT okruženja i pomogne njihovim korisnicima.

2013. godine, Dehan, zajedno sa Timotijem Gerlom i Saidom Ziuanijem, osniva kompaniju „AnsibleWorks Inc“, kako bi komercijalno podržali i sponzorirali Ansible alat. U oktobru 2015. godine kompanija Redhat kupuje „AnsibleWorks, Inc“.

Danas, Ansible je deo Fedora Linux distribucije, koju nudi Redhat. Takođe, kada su u pitanju drugi operativni sistemi, Ansible je dostupan putem EPEL repozitorijuma, koji nudi dodatne pakete za linux distribucije, kao što su Debian, Ubuntu, Oracle Linux i CentOS, koji će biti korišćen u studiji slučaja.

## 3.3 Arhitektura

Za razliku od drugih popularnih softvera za upravljanje konfiguracijom - kao što su već spomenuti Chef, Puppet ili CFEngine - Ansible koristi arhitekturu bez agenata<sup>5</sup>. Umesto

---

<sup>4</sup> Koreni Ansible alata, Majkl Dehan, preuzeto sa: <https://www.ansible.com/blog/2013/12/08/the-origins-of-ansible>

<sup>5</sup> Agent je softverski program koji se instalira na distribuiranim mrežama u svrhu izvršavanja posla i vraćanja podataka natrag na centralizovani server. U dizajnu bez agenata, informacije se prenose na računare ili se sa njih

korišćenjem agenta, Ansible upravlja ciljnom mašinom instaliranjem i pokretanjem svojih modula na ciljnoj mašini privremeno, korišćenjem SSH konekcije.

Ansible je alat zavistan od Python programskog jezika, pa je potrebno da i kontrolna i ciljna mašina imaju instaliran Python.

Termin kontrolna mašina odnosi se na mašinu na kojoj će biti pokrenuta Ansible skripta i koja će zatim upravljati procesom konfiguracije ciljnih mašina. Potrebno je postojanje bar jedne kontrolne mašine, ali je moguće imati i više njih.

Kako bi jedna mašina mogla biti korišćena u ulozi kontrolne mašine potrebno je ispuniti i određene uslove. Prvi uslov je postojanje instaliranog Python2 ili Python3 paketa i određenih zavisnosti koje su za njegovu instalaciju i funkcionisanje potrebne, a koje paket menadžer operativnog sistema u najvećem broju slučajeva i razrešava. Drugi zahtev za korišćenje mašine u ulozi kontrolne je da operativni sistem koji koristi ne bude Windows, budući da on nije podržan operativni sistem za kontrolnu mašinu. Takođe, radi olakšane komunikacije, prilikom izbora kontrolne mašine treba uzeti u obzir i „udaljenost” kontrolne od ciljnih mašina. Primera radi, ukoliko se kao ciljne mašine koriste one koje se nalaze u oblaku, preporučljivo je da se i kontrolna nalazi u istom tom oblaku. Nakon što su predočeni uslovi ispunjeni, poslednji i gotovo najbitniji korak je instalacija Ansible alata. Instalacija Ansible paketa je jednostavna i izvršava se putem komandne linije, u zavisnosti od konkretnog operativnog sistema koji koristimo.

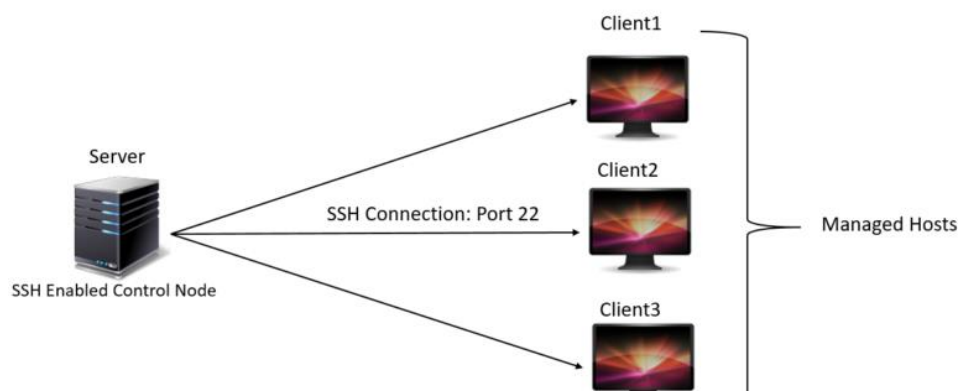
Kada su u pitanju ciljne mašine, osim odgovarajuće verzije Python 2 ili Python 3 softvera i potrebnih zavisnosti koje se za njih vezuju, ne postoje druga ograničenja. Podržan je i Windows operativni sistem, a Ansible nije neophodno instalirati.

Međutim, iako na upravljanim mašinama nije potrebno prisustvo demona<sup>6</sup>, neophodan je način uspostavljanja komunikacije između Ansible alata na kontrolnoj mašini i ciljnih mašina. Za većinu ciljnih mašina, Ansible uspostavlja vezu putem SSH konekcije, a prenosi module pomoću SFTP protokola. U slučaju da je došlo do uspostavljanja SSH konekcije, ali SFTP nije dostupan na nekim od upravljanih mašina, moguće je definisati i korišćenje SCP protokola izmenom konfiguracije samog Ansible alata.

---

prikupljaju bez instaliranja agenata. To se postiže komunikacijom sa softverom koji je već instaliran na računaru, uključujući operativni sistem i izvorno instalirane komponente. Na računaru je već instalirano više nego dovoljno matičnih programa i protokola za uspostavljanje potrebne komunikacije bez potrebe za agentskim softverom.

<sup>6</sup> Demon (eng. „Deamon”), odnosno servis, je program čija je svrha obavljati neki proces u pozadini. Glavna svrha servisa nije interakcija s korisnikom (koja se obično vrši putem grafičkog okruženja), već obavljanje nekog zadatka.



Slika 3.2

Jednostavna šema komunikacije ostvarene između Ansible kontrolne mašine i upravljanih mašina data je na slici 3.2, gde je prikazan način na koji kontrolni server (na levoj strani) ostvaruje komunikaciju putem SSH protokola, sa nekoliko ciljnih (tj. upravljanih) mašina (desno).

## 3.4 Osnovni koncepti

Kako bi se Ansible uspešno koristio, potrebno je upoznati njegove osnovne koncepte, koji adekvatnom primenom omogućavaju da se u potpunosti iskoriste sve mogućnosti koje ovaj alat nudi.

Neki od koncepata na koje će biti napravljen osvrt su: kontrolni čvor, upravljeni čvor, inventar, kolekcija, modul, „playbook“, zadatak, varijabla, rola, i blok.

### 3.4.1 Kontrolni čvor

Iako je već u prethodnom potpoglavlju, 3.3. Arhitektura, pojašnjeno značenje kontrolnog čvora, budući da je kontrolni čvor jedan od ključnih pojmova, naveden je i ovde uz ostale osnovne koncepte.

Dakle, kontrolni čvor (eng. „control node“) je bilo koja mašina sa instaliranim Ansible alatom. Ovo je mašina sa koje će se pokrenuti Ansible skripta, a zatim i ostvariti konekcija sa upravljanim mašinama.

### 3.4.2 Upravljeni čvor

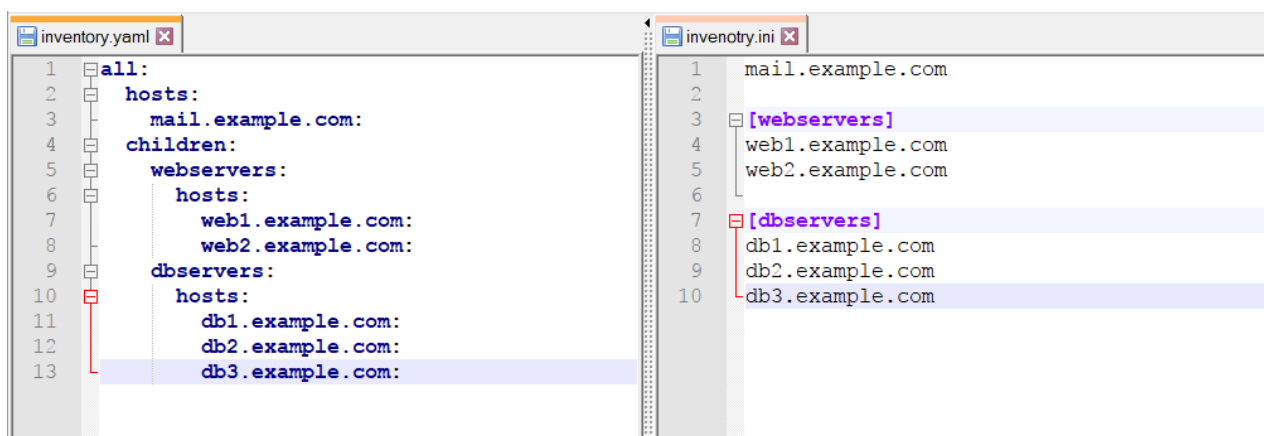
Pojam upravljanog čvora (eng. „managed node“), takođe objašnjen u potpoglavlju 3.3. Arhitektura, odnosi se na mrežni uređaj, ili server, na kome će Ansible vršiti promene. Ove mašine ne moraju imati instaliran Ansible.

### 3.4.3 Inventar

Inventar (eng. „inventory“), se odnosi na spisak upravljanih čvorova, a zapravo predstavlja datoteku koja sadrži informacije o svim upravljanim čvorovima. Informacije o upravljanim čvorovima se navode u vidu IP adrese, ili u vidu domena mašine i objektivno su jednostavne i ljudski lako čitljive.

Podrazumevana lokacija na kojoj se nalazi ova datoteka je na putanji „/etc/ansible/hosts“. Međutim, moguće je i predefinisati putanju iz komandne linije. To se vrši na način gde se prilikom pokretanja skripte, dodaje opcija „-i“, a zatim navodi željena putanja na kojoj se nalazi ova datoteka. Kada je u pitanju ovaj koncept, Ansible pruža veliku fleksibilnost, pa je takođe jedna od mogućnosti korišćenje više „inventory“ datoteka, ili, pak, dobavljanje ovih datoteka iz dinamičkih izvora, izvora koji se nalaze u oblaku, ili različitih formata, kao što su .yaml ili .ini formati.

U datoteci inventara, moguće je upravljane čvorove organizovati u grupe, koje se mogu i ugnježdavati, radi lakšeg skaliranja i upravljanja čvorovima. Postoje dve podrazumevane grupe. Prva je pod nazivom „all“ (odnosno sve), a koja sadrži sve grupe, i druga „ungrouped“ (odnosno negrupisane), koja sadrži sve čvorove koji ne pripadaju drugoj grupi osim „all“ grupi. Svaki čvor će uvek pripadati bar dve grupe. Pripadaće ili „all“ i „ungrouped“, ili može pripadati „all“ i nekoj drugoj grupi.




Slika 3.3

Na slici 3.3. prikazani su primeri datoteka inventara. Kao što je i napomenuto, ove datoteke se mogu navesti u .ini i .yaml formatu. Sa leve strane prikazan je primer datoteke u .yaml formatu, a sa desne strane može se videti primer identične datoteke, ali napisane u .ini formatu.

Ono što je karakteristično za ovaj primer je da postoji jedan čvor pod nazivom „mail.example.com“, koji nije grupisan (pripada „all“ i „ungrouped“ grupama), a zatim, postoje i dve grupe pod nazivima „webserver“ i „dbserver“, koje imaju po dva i tri upravljana čvora, retrospektivno.

Dodatna mogućnost koju pruža ovaj koncept je upotreba varijabli. Moguće je deklarisanje varijabli koje se odnose na konkretni čvor naveden u datoteci inventara, a isto tako postoji mogućnost deklarisanja varijabli koje se odnose na sve mašine svrstane u konkretnu grupu čvorova. Iako je za sam početak i jednostavnije sisteme moguće deklarisanje varijabli u datoteci inventara, za kompleksnija rešenja, dobra praksa je da se varijable čuvaju u odvojenim datotekama.

Primer deklarisanja varijabli u datoteci inventara dat je na slici 3.4, gde je za „db1.example.com“ definisana varijabla „http\_port“, kao i njena vrednost, i ova varijabla će se odnositi isključivo na ovaj čvor. Na sličan način, poslednje tri linije definišu varijable koje će se odnositi na celokupnu grupu pod nazivom „webserver“.



```

1 mail.example.com
2
3 [webserver]
4 web1.example.com
5 web2.example.com
6
7 [dbserver]
8 db1.example.com http_port=303
9 db2.example.com
10 db3.example.com
11
12 [webserver:vars]
13 http_port=80
14 maxRequestsPerChild=808
15

```

Slika 3.4

### 3.4.4 Kolekcije

Kolekcije (eng. „collections“) su format distribucije sadržaja Ansible alata. Mogu sadržati „playbooks“, role, module i dodatke, a za upotrebu ih je potrebno prvenstveno instalirati korišćenjem „Ansible Galaxy“ platforme, o kojoj će biti reči kasnije.

Omogućeno je razvijati svoje kolekcije i postoji iscrpna dokumentacija u te svrhe na stranici Ansible zvanične dokumentacije. Takođe, nakon razvoja, kolekcije se neretko objavljuju na „Ansible Galaxy“ platformi. Iako je ovo omogućeno, najčešća praksa je korišćenje već distribuiranih kolekcija, koje se instaliraju putem komandne linije.

```
ansible-galaxy collection install my_namespace.my_collection
```

Slika 3.5

Instalacija je generalno jednostavna. Na slici 3.5. prikazan je primer komande kojom je moguće instalirati kolekciju objavljenu na „Ansible Galaxy“. U ovo primeru instalira se fiktivna kolekcija pod nazivom „my\_collection“, a koja se nalazi u domenu imena naziva „my\_namespace“.

### 3.4.5 Moduli

Moduli (eng. „modules“) su samostalne skripte, tj. jedinice koda, za višekratnu upotrebu koje Ansible po naredbi izvršava. Moduli komuniciraju sa mašinama, bilo lokalno, na kontrolnoj mašini, bilo nad nekom udaljenom, upravljanom, mašinom, u cilju izvršavanja tačno određenih zadataka. Počevši od verzije 2.10 Ansible alata, moduli se grupišu u kolekcije. U zvaničnoj Ansible dokumentaciji je dostupna i lista svih objavljenih modula koji se mogu koristiti.

Svaki modul ima jasno definisanu funkcionalnost. Funkcionalnosti koje moduli pružaju su širokog spektra i variraju od instalacije određenih paketa, preko konfiguracije mrežnih interfejsa i izvršavanja proizvoljnih komandi iz komandne linije, pa sve do rada sa fajlovima. Jasno je da su mogućnosti koje moduli pružaju jako velike.

Moduli poseduju definisani interfejs, prihvataju argumente i, takođe, ispisuju povratne informacije na standardni izlaz. Gotovo svi moduli preuzimaju argumente u obliku „ključ=vrednost“. Neki moduli ne preuzimaju argumente, a moduli poput komandnih („shell“) modula preuzimaju niz komandi koje je potrebno izvršiti. Takođe, svi moduli vraćaju odgovor ispisan u JSON formatu, što omogućava implementaciju modula u bilo kom programskom jeziku. Još jedno bitno svojstvo ovog koncepta je idempotentnost. Ukoliko modul prilikom svog izvršavanja naiđe na okruženje koje već odgovara krajnjem željenom statusu, on u tom slučaju ne treba da načini bilo kakve izmene.

Postoje dva načina za pokretanje modula, prvi jeste izvršavanjem komande koja pokreće modul iz komandne linije, a drugi je korišćenje Ansible „playbook” koncepta, koji će biti obrađen u nastavku ovog rada. Drugi način omogućava i izvršavanje više modula. Kada su korišćeni u okviru „playbook-a”, moduli mogu okinuti događaje promena (eng. „change events”), koji pokreću izvršavanje daljih zadataka.

```
ansible webservers -m service -a "name=httpd state=started"
ansible webservers -m ping
ansible webservers -m command -a "/sbin/reboot -t now"
```

Slika 3.6

Slika 3.6. prikazuje način pokretanja Ansible modula na prvi naveden način, putem komandne linije. Može se primetiti korišćenje opcije „-m” nakon koje se navodi modul koji je potrebno izvršiti, a nakon navedenog modula uz opciju „-a”, navode se argumenti i njihova vrednost, odnosno, kada je u pitanju „command” komandni modul, navedena je komanda koju treba izvršiti. Ono što se još da primetiti sa ove slike je navođenje grupe „webservers”, (koja je definisana u datoteci inventara na slici 3.3), što govori Ansible-u nad čvorovima koje grupe će se izvršiti modul.

```
- name: reboot the servers
  command: /sbin/reboot -t now
```

Slika 3.7

```
- name: restart webserver
  service:
    name: httpd
    state: restarted
```

Slika 3.8

Na slikama 3.7. i 3.8. demonstrirano je pozivanje modula iz „playbook-a”, na drugi opisani način. Na obe slike zadaje se proizvoljni naziv koji opisuje zadatak, a zatim se navodi modul. Za sliku 3.8. karakteristično je korišćenje tzv. „kompleksnih argumenata”, koje omogućava YAML sintaksa.

Kao i kada su u pitanju kolekcije, ukoliko ne postoji implementirana željena funkcionalnost ni u jednom od velikog broja modula u Ansible kolekcijama, moguće je napisati svoje module. Moduli se mogu pisati u programskom jeziku po izboru, a, takođe, postoji i detaljna zvanična dokumentacija na internet stranici Ansible alata, koja približava način pisanja modula.

### 3.4.6 „Playbook“

Ansible „playbook“, na našem jeziku najbliže prevedeno kao scenario, je koncept specifičan za Ansible alat. Predstavlja uređene liste zadataka, koje su dokumentovane kako bi se mogle izvršavati iznova i iznova. „Playbook“ predstavlja potpuno drugačiji način korišćenja Ansible alata, od režima putem komandne linije, a mogućnosti koje korisniku pružaju su od izuzetnog značaja. Kako bi bolje približili ovaj koncept novim korisnicima, autori zvanične Ansible dokumentacije naveli su sledeći primer: „Ukoliko su moduli alati u Vašoj radionici, „playbook“ bi bio uputstvo za njihovu upotrebu, a datoteke inventara su Vaša sirovina za obradu“<sup>7</sup>.

Koncept „playbook“-a je osnova za zaista jednostavno upravljanje konfiguracijom i mehanizam za višestruko postavljanje okruženja, a posebno su prilagođeni za „deployment“ složenih aplikacija. „Playbook“ može definisati konfiguracije, ali, istovremeno, može orkestrirati konkretno uređene korake bilo kog manuelnog postupka, pa čak i u situacijama kada je potrebno da se prelazi sa jednog na drugi korak napred-nazad, poput praćenja algoritma, na različitim mašinama. Takođe, omogućeno je izvršavanje zadataka i sinhrono i asinhrono.

Playbook skripte pisane su u YAML sintaksi, koja je prilično ekspresivna i laka za razumevanje od strane korisnika. Ovaj koncept uključuje varijable, zadatke (eng. „tasks“), blokove (eng. „blocks“), uloge (eng. „roles“), kao i igre (eng. „plays“)<sup>8</sup>.

Kako bi se efikasno koristio koncept playbook-a, potrebno je bolje razumeti ključne reči (eng. „keywords“), koje uključuju sledeće koncepte:

1. „Task. Najuzi koncept od navedenih. Ansible, podrazumevano, izvršava „task“-ove po navedenom redu, jedan po jedan, nad svim mašinama koje su navedene u grupi nad kojom se task izvršava. Svaki task izvršava jedan modul sa definisanim argumentima.

---

<sup>7</sup> Preuzeto sa: [https://docs.ansible.com/ansible/latest/user\\_guide/playbooks.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks.html)

<sup>8</sup> Zbog prirode računarskih nauka i njihove osobine da se tehnologije, kao i sama nauka, razvija u svom najvećem delu na isključivo engleskom jeziku, u nastavku rada, koncepti zadataka, blokova, uloga i igara će biti referencirani na engleskom jeziku, a sve to u cilju bližeg shvatanja ovih koncepta.



Kada se task izvrši na svim predviđenim mašinama, prelazi se na sledeći task u redu. Ukoliko je task na određenoj mašini neuspešno izvršen, ova mašina biće izuzeta iz izvršavanja narednih task-ova. Ovo ponašanje se može predefinisati korišćenjem strategija (eng. „strategies“), koje predstavljaju još jedan Ansible-ov koncept.

2. „Block“. Omogućavaju logičko grupisanje taskova, kao i obradu i rukovanje greškama. Većina onoga što možete primeniti na jedan task (sa izuzetkom petlji) može se primeniti i na nivou bloka, što takođe olakšava definisanje podataka ili instrukcija koji su zajednički za više taskova. To neće implicirati da ove instrukcije utiču na sam blok, već ih nasleđuju taskovi koji su deo tog bloka, tj. primeniće se na taskove, a ne na sam blok.
3. „Play“. Koncept širi od bloka. Svaki „playbook“ sadrži jedan ili više „play“-a. Sastavljanjem „playbook“-a tako da sadrži više od jednog „play“-a omogućava se orkestriranje postavljanja okruženja na više mašina, tako da je moguće izvršavati neke korake na svim mašinama, neke nad jednom grupom mašina, a neke nad drugom grupom mašina.
4. „Role“. Role omogućavaju automatsko učitavanje varijabli, taskova i drugih Ansible artefakta na osnovu poznate strukture datoteka. Njihova struktura datoteka sadrži sedam definisanih direktorijuma, od kojih mora biti prisutan makar jedan. Jednom kada se ovaj sadržaj grupiše, omogućava ponovno korišćenje, kao i deljenje sa drugim korisnicima. Organizacija rola je prepuštena samom korisniku, ali je najčešća praksa postojanje jedne role, koja će biti primenjena na svim mašinama, a zatim i implementacija drugih koje će služiti za modifikaciju ponašanja servera u određene svrhe.

```
---
- hosts: webservers
  remote_user: root

  tasks:
    - name: ensure apache is at the latest version
      yum:
        name: httpd
        state: latest
    - name: write the apache config file
      template:
        src: /srv/httpd.j2
        dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
    - name: ensure postgresql is at the latest version
      yum:
        name: postgresql
        state: latest
    - name: ensure that postgresql is started
      service:
        name: postgresql
        state: started
```

Slika 3.9

Sada, kada su ključne reči „playbook“-a jasno definisane, sadržaj samog „playbook“-a postaje mnogo jasniji. Na slici 3.9. dat je primer jednog „playbook“-a, koji se sastoji od dva „play“-a. Prvi „play“ će se primeniti na mašine „webserver“ grupe (grupa definisana u primeru sa slike 3.3), a drugi „play“ će biti primenjen na „databases“ grupe (takođe sa slike 3.3). Kao korisnik koji izvršava komande biće, u oba slučaja korišćen „root“ korisnik<sup>9</sup>.

Kada je u pitanju prvi „play“, on se sastoji iz dva task-a, gde prvi izvršava modul „yum“ u cilju instalacije poslednje verzije „httpd“ paketa. Drugi task kreira konfiguracioni fajl na putanji „/etc/httpd.conf“.

U drugom „play“-u prvi task takođe koristi „yum“ modul u cilju dovođenja „postgresql“ paketa u stanje poslednje verzije. Sledeći task se korišćenjem „service“ modula uverava da je „postgresql“ servis u pokrenutom stanju.

Značajno je još jednom napraviti napomenu na svojstvo idempotentosti Ansible „playbook“-a. Naime, sva četiri od navedenih taskova u ovom primeru će biti izvršena samo u slučaju da željeno stanje već nije postignuto. Na primer, ako je poslednja verzija „httpd“ paketa već instalirana, Ansible će jednostavno taj korak preskočiti, bez načinjenih izmena.

## 3.5 Ostali koncepti

U ovom delu rada biće pokriveni koncepti koji nisu ključni za samo korišćenje Ansible alata, međutim mogu značajno unaprediti njegove mogućnosti koje on pruža, kao i samo iskustvo korišćenja ovog alata od strane korisnika.

### 3.5.1 „Ansible Galaxy“

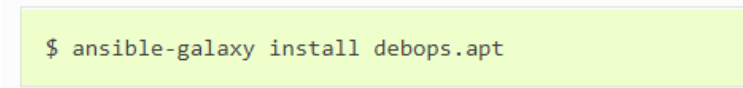
„Ansible Galaxy“ predstavlja javni repozitorijum Ansible alata. Koristi se kao središte za deljenje i pronalaženje Ansible sadržaja. Sadrži unapred definisane jedinice Ansible sadržaja koje se nazivaju role, a od „Ansible Galaxy“ verzije 3.2. javljaju se i kolekcije.

Kolekcije su format koji može obuhvatiti nekoliko playbook-a, rola, modula i dodataka, dok su role, sa druge strane, specijalizovane za izvršavanje određenih funkcionalnosti u cilju postizanja jednog obimnijeg zadatka.

---

<sup>9</sup> „Root“ korisnik je korisnik u Linux operativnom sistemu, koji ima sve privilegije i služi isključivo za administraciju sistema.

Kao što je već napomenuto, svaki korisnik može kreirati svoju rolu ili kolekciju, a zatim ih publikovati na ovu platformu. Isto tako, „Ansible Galaxy” omogućava i instalaciju rola i kolekcija koje su drugi korisnici objavili.



```
$ ansible-galaxy install debops.apt
```

Slika 3.10

Instalacija rola je generalno laka. Potrebno je pristupiti stranici platforme, a zatim pronaći rolu koju je cilj instalirati. Na stranici same role nalazi se komanda koju je potrebno izvršiti u cilju instalacije role. Primer takve komande nalazi se na slici 3.10. Nakon instalacije, rolu je moguće uvrstiti u željeni „playbook”.

### 3.5.2 „Ansible Tower”

Dosadašnja obrađena primena Ansible alata nije uključivala nikakvu vrstu korisničkog interfejsa, već se cela manipulacija alata svodila na komandnu liniju, kao i već postojeće programe za jednostavnu manipulaciju tekстом.

„Ansible Tower” je mrežni servis<sup>10</sup> i platforma, namenjena olakšanoj upotrebi samog Ansible alata od strane klijenata različitog spektra stručnosti i tehnoloških veština. Pruža pristup korisničkom grafičkom interfejsu. Ovo je komercijalni proizvod u ponudi RadHat, Inc. kompanije, koja je zadužena i za njegovo održavanje i podršku.

Ansible Tower predstavlja središte automatizacije izvršavanja zadataka, omogućeno putem povezivanja sa okruženjima. „Ansible Tower” pruža početnu stranicu (eng. „dashboard”) na kojoj su dostupne informacije o svemu što se dešava u okruženjima u vidu grafičkog prikaza. Moguće je ispratiti status okruženja, kao i status pokrenutih zadataka. Takođe, dostupan je i prikaz izvršenih task-ova pokrenutog „playbook”-a, u realnom vremenu. Još neke od osnovnih odlika ove platforme su pristup baziran na različitim ulogama korisnika, uvid u to kada je koji korisnik izvršio određenu naredbu, kao i zakazivanje izvršavanja poslova u određeno vreme.

Ovaj alat nastao je iz „AWX” projekta, koji predstavlja projekat otvorenog koda u vlasništvu Redhat kompanije, a dostupan je za preuzimanje putem javnog repozitorijuma.

---

<sup>10</sup> Mrežni servis (eng. „web-service”) predstavlja način komunikacije između dva elektronska uređaja na mreži. Ova komunikacija se obično vrši tako što klijent šalje zahtev pomoću mrežnog servisa serveru i rezultujuća informacija se vraća klijentu u nekom obliku

„Ansible Tower” predstavlja neku njegovu stabilnu verziju, nadograđenu i održavanu u komercijalne svrhe.

Poslednja važna napomena, koja se odnosi na „Ansible Tower”, je mogućnost jednostavne integracije sa drugim alatima koji mogu služiti u svrhe automatizacije kao što su Jenkins, BitBucket i Terraform, ali i sa mnogim drugim alatima različitih namena.

### 3.6 Slučajevi korišćenja

Pošto su detaljno obrađeni koncepti i svojstva Ansible alata, u ovom delu će biti napravljen osvrt na slučajeve korišćenja ovog alata, kao i realne primere koji pokazuju način na koji se ovaj alat koristi u prominentnim organizacijama i preduzećima.

Ansible besprekorno objedinjuje orkestraciju radnog toka (eng. „workflow”) sa konfiguracijom, postavljanjem okruženja i „deployment”-om aplikacija u jednu jednostavnu platformu, a sve to prilagođavajući se čak i kompleksnim sistemima. Neki od slučajeva korišćenja ovog alata su:

1. Postavljanje okruženja. Pružanje infrastrukture je prvi korak životnog ciklusa bilo koje aplikacije. Ansible omogućava postavljanje ove infrastrukture bilo da se radi o serverima, virtuelnim mašinama ili instanciranjem i podešavanjem okruženja u oblaku.
2. Upravljanje konfiguracijama. Centralizacija konfiguracionih specifikacija je jedan od najčešćih slučajeva korišćenja ovog alata. Ansible ovom problemu pristupa na jednostavan način, fokusirajući se na ciljno stanje konfiguracija, umesto na korake koje treba preduzeti u nameri postavljanja sistema u to ciljno stanje.
3. „Deployment” aplikacija. Ansible i „Ansible Tower” omogućavaju praktičan i efektivan način sprovođenja aplikacije kroz sve faze njenog životnog ciklusa. Primenom Ansible alata u ove svrhe dobija se na pouzdanosti, jednostavnom skaliranju aplikacija.
4. Kontinuirana isporuka (eng. „Continuous Delivery”). CI/CD tok uključuje u proces mnoge timove, a fokus je na malim i čestim ažuriranjima proizvoda koji se isporučuje. Ansible ovaj tok potpomaže automatizacijom ovog procesa, pri čemu garantuje stabilnost prilikom različitih etapa toka.
5. Automatizacija bezbednosnih podešavanja. Kada je u pitanju automatizacija bezbednosnih podešavanja, javlja se problem primene bezbednosnih rešenja na okruženja koja se postavljaju i modifikuju velikom brzinom. Ansible kao rešenje ovog problema ne samo da pruža automatizaciju i integraciju različitih rešenja, već poseduje i mehanizme rukovanja događajima koji predstavljaju pretnje po bezbednost. Takođe,

pruža i mogućnost integracije i saradnje sa različitim sigurnosnim alatima kao što su CyberArk, Fortinet i Synology.

6. Orkestracija. Konfiguracije same po sebi ne definišu okruženje. Potrebno je obezbediti automatizaciju načina na koji okruženja stupaju u komunikaciju jedni sa drugima. Razvoj virtuelizacije, kontejnerizacije, kao i računarstva u oblaku doveli su do situacije u kojima se koristi veliki broj mašina između kojih je neophodno uskladiti komunikaciju. Omogućeno je da Ansible pristupi ovom problemu u ulozi dirigenta koji će različite sisteme dovesti u koherentno stanje, gde njegova jednostavna sintaksa olakšava obavljanje ovog zadatka.

Evidentno je da su mogućnosti za primenu ovog alata velikog obima. O njegovoj efikasnosti i relevantnosti govori i činjenica da je primena Ansible alata prisutna u mnogim globalnim, prominentnim organizacijama, dominantnim u sferama svog poslovanja. Ove tvrdnje će biti potkrepljene konkretnim primerima:

1. NASA. Ovoj organizaciji bilo je potrebno migrirati šezdeset i pet okruženja iz tradicionalnog hardverskog okruženja koje se nalazilo u data-centrima, na okruženja u oblaku, u cilju veće agilnosti i smanjenih troškova. Kao rešenje za ovu situaciju primenjeni su Ansible i „Ansible Tower“, koji su omogućili upravljanje ovim okruženjima. Kao rezultat ovog procesa NASA je, između ostalog, dobila smanjeno vreme ažuriranja svoje internet stranice sa jednog sata na pet minuta, ažuriranje novijih verzija softvera, koje je ranije bilo proces koji se odvijao nekoliko dana, sada se izvršava za četrdeset i pet minuta, a postavljanje korisničkih naloga na okruženje odvija se čitavih deset minuta.
2. BMW. U slučaju BMW organizacije, prilikom razvoja autonomnih vozila, javila se potreba za pristupom, analiziranjem i primenom velike količine podataka, prikupljenih od strane senzora tokom test vožnji, kao i potreba da zatim ažurira svoje aplikacije na osnovu novih algoritama koje su proizveli. Rešenje koje je postignuto je platforma, kreirana i konfigurisana nakon tri meseca razvoja, koja je pružila skalabilno mašinsko učenje i obradu velikih podataka, a koja je razvijana uz pomoć tehnologija koje Redhat nudi, naravno, uključujući i Ansible. Ova platforma, dovela je i do automatizacije ponavljajućih zadataka, a ishod toga je, između ostalog, i povećana produktivnost programera ove organizacije.
3. Microsoft, U cilju automatizacije svojih tehnologija i infrastrukture, Microsoft je, koristeći „RedHat Ansible Automation Platform“ i blisko sarađujući sa Redhat timom za konsultovanje, stvorio standardizovano, centralizovano okruženje za automatizaciju mreže, a koje smanjuje rutinske, ponovljive zadatke i njihovu složenost. DevOps timovi

široj ovoj kompaniji sada se mogu usredsrediti na razmenu znanja, izgradnju veština, kao i stvaranje inovativnih tehnoloških rešenja.

### 3.7 Prednosti i nedostaci

Neke od najvažnijih prednosti Ansible alata, u odnosu na konkurentske softvere iste namene su:

1. Lakoća savladavanja. Ovo je možda svojstvo Ansible alata koje je i najviše hvaljeno u literaturi. Korisnicima je omogućeno unapređivanje sistema velikom brzinom. Ansible je podržan jednostavnom ali iscrpnom dokumentacijom, koju je objektivno lako razumeti. Za razliku od nekih slučajeva kada dokumentacija umesto da razjasni koncepte, učini upravo suprotno, zvanična dokumentacija ovog alata je zaista jasna i ubrzava savladavanje toka rada sa ovim alatom, kao i njegove koncepte. Takođe, činjenica da se taskovi sekvencijalno izvršavaju i zaustavljaju kada se naiđe na problem, omogućava olakšano otklanjanje tih problema, naročito u slučaju kada alat koriste početnici.
2. Razvijan u Pythonu. Za razliku od drugih konkurentskih rešenja, koja su razvijana u jeziku kao što je Ruby, Ansible alat je napisan u Pythonu. Iz te činjenice sledi i činjenica da je pokretanje samog alata lakše, jer su Python biblioteke već podrazumevano prisutne na većini Linux distribucija. Takođe, Python je jezik koji se generalno često koristi za administrativne zadatke i pisanje skripti, pa su i šanse veće da će korisnici ovog alata biti upoznati sa ovim programskim jezikom, pre nego sa Ruby programskim jezikom.
3. Arhitektura bez agenta. Za upravljanje čvorovima, Ansible rukuje svim komunikacijama sa standardnim SSH protokolom ili Paramiko modulom, što je Python implementacija SSH2 protokola. Alat ne zahteva da se bilo koji agenti instaliraju na udaljenim sistemima kojima se upravlja, što znači manje troškova rada i poboljšanje performansi.
4. Skripte bazirane na YAML sintaksi. Ansible „playbook“ skripte napisane su u YAML sintaksi koja je daleko zahvalnija za upravljanje konfiguracijom od ostalih formata, poput, na primer, JSON formata. YAML je ekspresivan, podržava komentare kao i referenciranje drugih stavki.
5. „Ansible Galaxy“. Za razliku od svojih alternativa koje ovakav koncept ne poseduju, „Ansible Galaxy“ umnogome povećava efikasnost primene ovog alata, omogućavajući lako preuzimanje već implementiranih, proverenih rešenja.

Kada su u pitanju nedostaci ovog alata izdvajaju se sledeći:

1. Nedostatak korisničkog interfejsa. Prvobitno razvijan isključivo kao alat za primenu iz komandne linije, Ansible je prvi put pokušao da uvede grafički korisnički interfejs putem već spomenutog „AWX“ projekta, koji je potom evoluirao u „Ansible Tower“. Iako „Ansible Tower“ predstavlja veliki pomak, on još uvek ostavlja prostora za unapređivanje. Naime, za sada se tek 85% mogućnosti iz komandne linije može postići putem ove platforme, a još jedna česta smetnja je zakasnela sinhronizacija platforme sa komandom linijom, što rezultira nedoslednim rezultatima upita.
2. Izostanak podrške za Windows. Od svoje verzije 1.7, Ansible podržava i Unix/Linux, kao i Windows operativne sisteme u ulozima upravljanih čvorova. Međutim, Linux mašina je i dalje neophodna u ulozima kontrolnog čvora. Ansible je još uvek u najranijim fazama proširenja usluga i na Windows, budući da najavljene verzije navodno uključuju veću interoperabilnost Windows-a.
3. Manjak iskustva sa podrškom velikim organizacijama. Iako je „Ansible Tower“ namenjen srednjim i velikim preduzećima ( ponudom „Enterprise“ i „Premium“ paketa), u poređenju sa svojim konkurentima, poput Chef i Puppet alata, ova kompanija ima znatno manje iskustva u radu sa većim organizacijama.
4. Predstavlja novo rešenje. Za razliku od svojih alternativa, već spomenutim Puppet i Chef alatima, koji se nadmeću još i pre trenutka nastanka Ansible alata, Ansible ima najmanju zajednicu korisnika, kao i najmanje materijala na internetu za samopomoć i rešavanje problema. Manje vremena na tržištu takođe znači i to da određeni krajnji slučajevi, bagovi i drugi problemi možda još uvek nisu primećeni, pa time ni rešeni.

## 4. Studija slučaja

U ovom poglavlju biće opisana primena Ansible alata u svrhe „deploymenta“ aplikacije pod nazivom „Pharmacy“. Ona predstavlja veb-aplikaciju<sup>11</sup> koja omogućava „online“ poslovanje apoteka, kao i olakšanu komunikaciju apotekama sa svojim klijentima. Cilj ovog istraživanja je proučavanje primenljivosti alata Ansible za brz „depolyment“, kao i upravljanje infrastrukturom. Upravo ovi ciljevi su i navedeni kao neki od slučajeva korišćenja Ansible alata.

### 4.1 Korišćeni alati

U kombinaciji sa Ansible alatom, čija je primena sam fokus ove studije, bilo je neophodno upotrebiti i druge alate, kako bi ovo istraživanje bilo sprovedeno. Alati koji su korišćeni su sledeći:

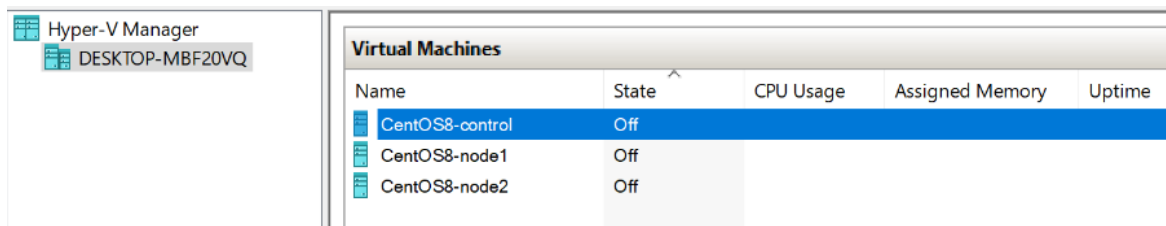
1. Github (eng. „Github“). Ranije, u ovom radu, napomenuto je da je bitna praksa prilikom implementacije principa infrastrukture kao koda korišćenje softvera za kontrolu verzija. Github je veb-baziran servis za kontrolu verzije, koji pruža Git funkcionalnosti. U ovom istraživanju korišćen je za distribuiranu kontrolu verzija i menadžment izvornog koda. Takođe, celokupni kod implementacije ove studije slučaja dostupan je na javnom Github repozitorijumu (videti prilog P).
2. „Hyper-V Manager“. Predstavlja izvorni hipervizor koji može da kreira virtuelne mašine na sistemima koji rade pod operativnim sistemom „Windows“. Server na kojem

---

<sup>11</sup> Veb-aplikacija (eng. „web application“) je računarski softver kojem se pristupa putem internet pregledača i povezan je sa bazom podataka kako bi se pružio prilagodljiv interaktivni korisnički doživljaj.



se nalazi „Hyper-V“ može biti konfigurisan tako da izlaže virtualne mašine jednoj ili više mreža. U studiji slučaja pomoću ovog alata kreirane su i korišćene tri virtuelne mašine, čiji je prikaz iz grafičkog interfejsa ovog alata dat na slici 4.1.



Slika 4.1

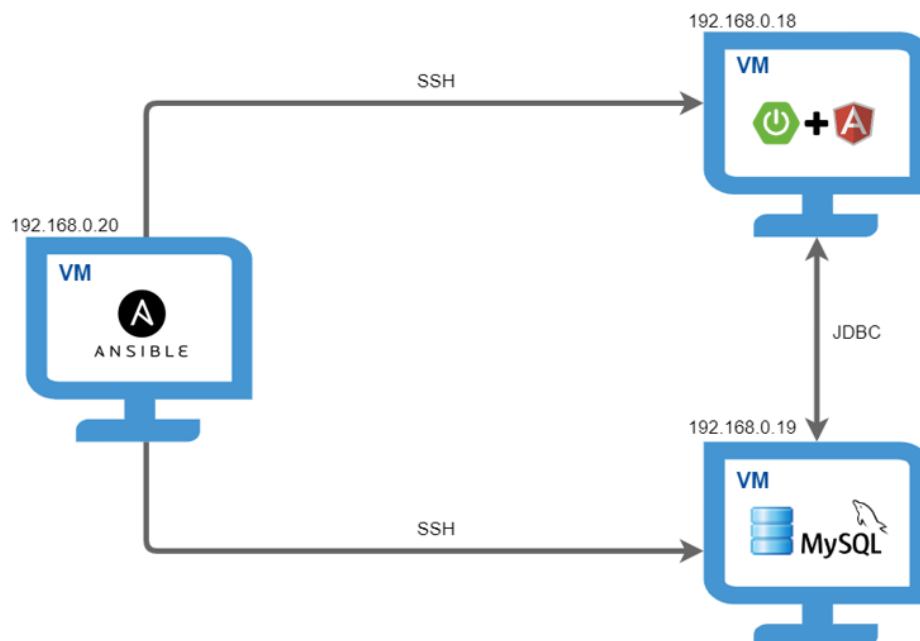
3. „PuTTY“. Ovo je besplatan emulator<sup>12</sup> terminala, i aplikacija za prenos fajlova na mreži. Otvorenog je koda i podržava nekoliko mrežnih protokola kao što su SCP, SSH, Telnet i rlogin. U ovom istraživanju korišćen je u cilju upravljanja kontrolnom mašinom putem SSH protokola, kao i u svrhe pristupa upravljanim mašinama, kada je to bilo potrebno.

## 4.2 Arhitektura postavljenog sistema

Kao što je već navedeno prilikom izrade ove studije slučaja korišćene su tri virtuelne mašine. Sve tri navedene mašine koriste „CentOS 8“ operativni sistem koji predstavlja distribuciju Linux operativnog sistema.

Na slici 4.2. data je ilustracija šeme koja bliže opisuje postavljeni sistem. Na levoj strani slike nalazi se mašina koja je korišćena u ulozi kontrolne mašine. U sklopu pripreme za istraživanje, na ovoj mašini instaliran je Ansible alat, kao i neophodni Python paketi. Ovim je mašina ispunila sve potrebne preduslove da postane kontrolna mašina Ansible alata. Dakle, ona koristi Linux operativni sistem i, takođe, ima instalirane Ansible i Python pakete.

<sup>12</sup> Emulator je softver koji omogućava da se određeni program ili proces izvršava na platformi (računarskoj arhitekturi ili operativnom sistemu) drugačijoj od one za koju je predviđen.



Slika 4.2

Sa desne strane iste slike prikazane su dve mašine. Jedna od ove dve mašine je namenjena u svrhe servera za bazu podataka (dole desno) i konkretno će koristiti „MySQL“ server. Druga mašina je namenjena pokretanju „Pharmacy“ aplikacije. Važno je još napomenuti da se ova aplikacija sastoji iz dva projekta. Jedan izrađen u vidu „Maven“ aplikacije koja koristi „Spring Boot“ okvir i predstavlja serverski deo aplikacije, a druga implementirana u „Angular“ alatu, koja predstavlja klijentski deo aplikacije. Struktura projekta implementirana je na način da serverski deo aplikacije zahteva obavljanje komunikacije sa „MySQL“ serverom. Ova komunikacija biće omogućena upravo putem Ansible alata.

Dodatna priprema, koja je obavljena na sve tri podrazumevala je:

1. Konfiguraciju mrežnih interfejsa mašina, gde su manuelno definisane vrednosti IP adresa na kojima će mašine biti dostupne za komunikaciju.
2. Kreiranje korisnika pod nazivom „ansible-control“, kome će biti zagarantovane „sudo“ privilegije. To zapravo označava garantovanje neograničenih privilegija, omogućujući time neometanu administraciju sistema.
3. Generisanje SSH para ključeva (isključivo na kontrolnoj mašini) za novokreiranog korisnika, a zatim obavljanje SSH konekcije sa preostale dve, upravljane, mašine. Ovim je omogućena sigurna komunikacija između kontrolne i upravljanih mašina, koja ne zahteva unošenje lozinke, već se obavlja automatski, proverom identiteta korisnika putem para javnog i privatnog ključa.

## 4.3 Primena Ansible koncepata

Poput potpoglavlja 3.4, i ovo potpoglavlje će započeti opisom koncepta inventara. Naime, kako je već objašnjeno u inventaru su navedene upravljane mašine.

```
1  [webservers]
2  192.168.0.18
3
4  [dbservers]
5  192.168.0.19
```

Slika 4.3

Na slici 4.3 data je *inventory.yaml* datoteka (pogledati prilog P.1). Krajnje jednostavna, ova datoteka grupiše upravljanje mašine u dve grupe: „webservers“ (mašine namenjene postavljanju aplikacije) i „dbservers“ (mašine namenjene postavljanju baze podataka. Ukoliko bi bilo potrebno ovaj projekat prilagoditi konfiguraciji većeg broja servera, to bi se moglo učiniti jednostavnim navođenjem dodatnih upravljanih čvorova u ove grupe.

Sledeći koncept, koji se pokazao kao izuzetno značajan za efikasnost same izrade skripte, je koncept rola, koje su preuzete sa „Ansible Galaxy“ platforme. Slika 4.4. prikazuje *requirements.yaml* datoteku (pogledati prilog P.2), koja navodi sve role koje će biti preuzete sa „Ansible Galaxy“ platforme. Kako bi ovo bilo realizovano, neophodno je iz komandne linije pokrenuti komandu „ansible-galaxy install“ i dodati „-r“ opciju praćenu putanjom ove datoteke. Na ovaj način omogućena je instalacija svog neophodnog sadržaja koji se preuzima sa „Ansible Galaxy“ platforme, u jednom koraku.

```
1  ---
2
3  #roles:
4  - name: geerlingguy.java
5  - name: gantsign.maven
6  - name: geerlingguy.mysql
7  - name: geerlingguy.nodejs
```

Slika 4.4

Dakle, role koje su dobavljene sa „Ansible Galaxy“ platforme su role koje na mašini obezbeđuju sledeće alate: java i maven (potrebni za serverski deo aplikacije), zatim, nodejs

(koji je neophodan kako bi bio omogućen klijentski deo aplikacije), i mysql rola koja je omogućila efikasnu konfiguraciju servera baze podataka.

Bitno je napomenuti da je upotreba ovih, već implementiranih, testiranih i široko upotrebljavanih, znatno olakšala i ubrzala izradu ovog projekta. Budući da bi konfigurisanje navedenih alata podrazumevalo iscrpno istraživanje, a, potom, i samu implementaciju u vidu Ansible skripti. Evidentan je značaj ovakvog pristupa, koji se ogleda, ne samo u vidu uštede vremena, već i u vidu umanjene šansi za pravljenje pogrešaka.

Radi olakšanog upravljanja procesom postavljanja okruženja, kao i bolje organizacije strukture ovog projekta, u svrhe postavljanja veb-servera i postavljanja servera za bazu podataka, definisana su dva Ansible „playbook-a“:

1. *deploy\_webservers.yaml* (videti prilog P.3)
2. *deploy\_dbservers.yaml* (videti prilog P.4)

Kada je u pitanju *deploy\_webservers.yaml* „playbook“, prikaz sadržaja ove skripte dat je na slici 4.5. Ovaj „playbook“ sadrži dva „play-a“, od kojih se oba izvršavaju na mašinama koje pripadaju grupi „webmasters“, definisanoj u datoteci inventara. Prvi „play“ osigurava izvršavanje „base“ role, definisane u ovom projektu, dok je drugi „play“ osigurao izvršenje „webserver“ role, takođe definisane u ovom projektu, a namenjene postavljanju mašine u ulogu veb-servera.

```
---  
- hosts: webmasters  
  remote_user: ansible-control  
  gather_facts: true  
  become: true  
  roles:  
    - base  
  
- hosts: webmasters  
  remote_user: ansible-control  
  gather_facts: true  
  become: true  
  roles:  
    - webserver
```

Slika 4.5

„Playbook“ pod nazivom *deploy\_dbservers.yaml*, na sličan način kao i prethodno opisani „playbook“, postavlja upravljane čvorove koji pripadaju „dbservers“ grupi u ulogu servera za bazu podataka. Ovaj „playbook“ sadrži dva „play-a“, prvi, koji pokreće „base“ rolu, i drugi, koji poziva „dbserver“ rolu, definisanu u ovom projektu. Slika 4.6. prikazuje opisani sadržaj ovog „playbook-a“.

```
---
- hosts: dbservers
  remote_user: ansible-control
  gather_facts: true
  become: true
  roles:
    - base

- hosts: dbservers
  remote_user: ansible-control
  gather_facts: true
  become: true
  vars_files:
    - vars/main.yaml
  roles:
    - dbserver
```

Slika 4.6

U samom korenu projekta nalazi se skripta *run-deployment.sh* koja omogućava pokretanje datih „playbook-a“ (vidi prilog P.5). Ova skripta zahteva navođenje jednog parametra sa jednom od sledećih vrednosti: *all*, *dbservers*, ili *webservers*. U zavisnosti od vrednosti parametra, skripta će pokrenuti oba ili jedan od dva „playbook-a“.

Na slici 4.7. prikazan je deo skripte koji u slučaju „all“ vrednosti parametra iz komandne linije poziva prvo *deploy\_dbservers.yaml* „playbook“, a zatim *deploy\_webservers.yaml* „playbook“. U oba poziva navedena je putanja datoteke inventara, koristeći opciju „-i“.

```
#!/bin/bash

mode=$1

#installation of required roles
ansible-galaxy install -r requirements.yaml

if [ $mode = "all" ]
then
echo "Deploying all servers..."
echo "Deploying database servers..."
ansible-playbook -i Environments/inventory deploy_dbservers.yaml
echo "Deploying web servers..."
ansible-playbook -i Environments/inventory deploy_webservers.yaml
fi
```

Slika 4.7

Osim što preuzima parametar i u odnosu na njegovu vrednost pokreće jedan ili oba „playbook-a“, ova skripta na samom početnu komandom „ansible-galaxy install“ preuzima sav neophodan sadržaj sa „Ansible Galaxy“ platforme. Namena ove skripte je da olakša proces testiranja, budući da neće uvek biti neophodno postavljati obe grupe servera, a takođe komande za pokretanje „playbook-a“ ostaju ubeležene i pokretane uvek na isti način, smanjujući mesta za greške.

Kao što je već navedeno, u ovom projektu definisane su tri role:

1. *base* rola
2. *webservers* rola
3. *dbservers* rola

Ranije u ovom radu, spomenuto je da je jedna od često primenjivanih praksi implementacija bazne role, koja će biti primenjena na sve sisteme. U ovom radu u te svrhe je napisana *base* rola. Prednost primene rola ovakvog tipa ogleda se u izbegavanju ponavljanja istovetnih linija koda, time što se se konfiguracije koje su zajedničke za veliki broj mašina (a koje će pak, imati različite uloge nadalje), pišu u okviru ove role, a potom se postavljaju jednostavno, primenom ove role.

U slučaju ove studije, ova rola je implementirana na jednostavan način. Od direktorijuma potrebnih za role ima samo direktorijum pod nazivom „tasks“, u kome se nalazi *main.yaml* fajl (videti prilog P.6). Prikaz sadržaja ovog fajla dat je na slici 4.8.

```
1  ---
2
3  - name: Upgrade all packages
4    yum:
5      name: '*'
6      state: latest
7
8  - name: Install packages
9    yum:
10     name:
11       - python3
12       - python3-pip
13     state: present
```

Slika 4.8

U *main.yaml* fajlu „tasks“ foldera, navedena su dva taska koje je potrebno izvršiti. Prvi obezbeđuje ažuriranje svih paketa na mašini, dok će drugi obezbediti instalaciju potrebnih „Python3“ paketa. U oba slučaja, prilikom izvršavanja taskova bio je iskorišćen „yum“ modul, zadužen za upravljanje paketima na CentOS operativnom sistemu, a koji je jedan od podrazumevanih modula u okviru Ansible alata.

Naredna implementirana rola je rola pod nazivom „dbservers“, a koja postavlja mašinu u ulogu servera baze podataka. Kao i „base“ rola, i ova rola sadrži samo „tasks“ direktorijum, sa *main.yaml* datotekom. Na slici 4.9. dat je prikaz sadržaja *main.yaml* datoteke (videti prilog P.7).

```
1  ---
2
3  - name: install mysql role
4    include_role:
5      name: geerlingguy.mysql
6
7  - name: expose port 3306
8    shell: firewall-cmd --zone=public --add-port=3306/tcp
```

Slika 4.9

Datoteka prikazana na slici izvršava dva taska. Prvi task koristi „include\_role“ modul, podrazumevano ugrađen u Ansible alat, koji govori alatu da je neophodno primeniti određenu rolu. U ovom slučaju to je rola pod nazivom „geerlingguy.mysql“, prethodno dobavljena sa „Ansible Galaxy“ platforme. Ovde još jednom dolazi do izražaja značaj celokupnog „Ansible Galaxy“ koncepta, jer bi bez primene ovog koncepta „dbserver“ rola sadržala značajno veći broj taskova.

Drugi task koji se izvršava namenjen je otkrivanju porta 3306, koji će omogućiti komunikaciju veb-servera sa serverom baze podataka, a koja će se obavljati putem ovog porta. U cilju ostvarivanja ovog zadatka upotrebljen je „shell“ modul, takođe ugrađen u Ansible, a koji omogućava izvršavanje komandi koje bi inače bile pokrenute iz komandne linije.

Poslednja navedena rola, „webserver“, namenjena je specijalizaciji mašine u ulogu veb-servera, na kome će biti pokrenuta „Pharmacy“ aplikacija. Ova rola, kao i prethodne dve, sadrži jedino „tasks“ direktorijum, ali, za razliku od njih, taskovi su grupisani u dve datoteke:

1. *configure-git-project.yaml* (videti prilog P.8)
2. *main.yaml* (videti prilog P.9)

```

1  ---
2
3  - name: clone git project
4    git:
5      repo: 'git@github.com:milanovicn/ISA.git'
6      dest: /home/ansible-control/pharmacy-app
7      version: nina-diplomski
8      key_file: /root/.ssh/id_rsa
9
10 - name: build and compile maven project
11   shell: /opt/maven/apache-maven-3.8.1/bin/mvn clean install > /home/ansible-control/logs/mvn-clean-install.log
12   args:
13     chdir: /home/ansible-control/pharmacy-app/ISA-Backend
14   environment:
15     MAVEN_HOME: /opt/maven/apache-maven-3.8.1
16     JAVA_HOME: /usr/lib/jvm/java-11-openjdk-11.0.11.0.9-0.el8_3.x86_64
17
18 - name: install node modules
19   shell: npm i > /home/ansible-control/logs/npm-install.log
20   args:
21     chdir: /home/ansible-control/pharmacy-app/ISA-Frontend

```

Slika 4.10

Datoteka pod nazivom *configure-git-project.yaml*, sastoji se iz tri taska. Prvi task, dat na slici 4.10. ostvaruje se pomoću „git“ modula, koji prihvata razne argumente kojima je



moguće odrediti željeno ponašanje. Neki od argumenata podrazumevaju određivanja repozitorijuma sa kojeg će se kod dobiti, određeni direktorijum, određenu granu repozitorijuma, kao i koji će ssh ključ biti upotrebljen za autentifikaciju. U ovom slučaju, grana sa koje će se kod dobiti je grana pod nazivom „nina-diplomski“, a ključ koji će se koristiti za autentifikaciju prilikom ssh konekcije se nalazi na putanji „/root/.ssh/id\_rsa“.

Preostala dva taska u ovom fajlu (vidi sliku 10), koriste već spomenuti „shell“ modul, a omogućavaju pokretanje komandi za kompajliranje serverskog i klijentskog dela projekta. Drugi po redu zadužen je za pozivanje „mvn clean install“ komande, koja izvršava kompajliranje projekta i kao proizvod kreira „ISA-Backend-0.0.1-SNAPSHOT.jar“ fajl koji predstavlja artefakt pomoću koga će se „backend“ deo aplikacije kasnije pokretati. „Shell“ modul omogućava i korišćenje argumenata, poput „chdir“ koji govori na kojoj putanji će komanda biti izvršena, kao i environment argument koji omogućava postavljanje varijabli.

Treći i poslednji task u *configure-git-project.yaml* fajlu je zadužen za instalaciju „NodeJS“ modula, koji su neophodni za pokretanje „frontend“ dela aplikacije. Ovaj task takođe koristi „chdir“ argument koji određuje putanju izvršavanja komande i instalacije ovih modula.

Druga, *main.yaml* datoteka, započinje uključivanjem potrebnih rola, uvezenih sa „Ansible Galaxy“, pomoću, već spomenutog, modula pod nazivom „include\_role“. Ove role potrebne su za instalaciju alata poput „Maven“, „Java“ i „Nodejs“ alata, koji omogućavaju pokretanje projekta. Na slici 4.11. prikazan je način primene modula „include\_role“ u svrhe instalacije potrebnih rola.

```
1  ---
2
3  - name: install java role
4    include_role:
5      name: geerlingguy.java
6
7  - name: install maven role
8    include_role:
9      name: gantsign.maven
10
11 - name: install nodejs role
12   include_role:
13     name: geerlingguy.nodejs
14
```

Slika 4.11

Sledeći koraci u cilju postavljanju potrebne konfiguracije predstavljaju instalacija paketa i postavljanje varijabli u okruženju (slika 4.12). Prvi korak izvršen je korišćenjem „yum“ modula. Ovaj modul poziva „yum“ menadžer paketa i željene pakete (u ovom slučaju „git“ i „mysql“ pakete) postavlja u željeno stanje (u ovom slučaju „present“, odnosno, prisutno, instalirano stanje) koristeći „state“ argument. Drugi korak, koji podrazumeva postavljanje varijabli sistema, omogućen je jednostavnim navođenjem komande, primenom već pojašnjenog, „shell“, modula.

```
- name: Install packages
  yum:
    name:
      - git
      - mysql
    state: present

- name: set enviromental variables
  shell: export JAVA_HOME="/usr/lib/jvm/java-11-openjdk-11.0.11.0.9-0.el8_3.x86_64" && export M2_HOME="/opt/maven/apache-maven-3.8.1"
```

Slika 4.12

Nakon postavljanja rola, i postavljanja neophodnih varijabli sistema, pozivaju se taskovi iz *configure-git-project.yaml* datoteke, koristeći modul pod nazivom „include\_tasks“ (vidi sliku 4.13).

```
25
26 - name: configure Pharmacy application from git
27   import_tasks: ./roles/webserver/tasks/configure-git-project.yaml
28
29 ..
```

Slika 4.13

Nakon izvršavanja taskova iz *configure-git-project.yaml* datoteke, pomoću „shell“ modula pokreću se serverski i klijentski deo aplikacije.

Slika 4.14. prikazuje završne korake preduzete u cilju isporuke aplikacije. Prvi korak omogućava pokretanje „backend dela projekta“. U ovm tasku ponovno je iskorišćen „shell“ modul, uz navedeni „chdir“ argument. Navedena komanda pokreće izgenerisani „ISA-Backend-0.0.1-SNAPSHOT.jar“ artefakt, a sva obaveštenja o događajima preusmerava u „mvn-run.log“ fajl. Drugi task, takođe koristi „shell“ modul i „chdir“ argument, a uz navedenu „npm start“ komandu pokreće „frontend“ deo aplikacije i obaveštenja o događajima preusmerava u „npm-start.log“ datoteku.

```
28 - name: run maven project
29 shell: /usr/lib/jvm/java-11-openjdk-11.0.11.0.9-0.el8_3.x86_64/bin/java -jar ISA-Backend-0.0.1-SNAPSHOT.jar > /home/ansible-control/logs/mvn-run.log 2>&1 &
30 args:
31   chdir: /home/ansible-control/pharmacy-app/ISA-Backend/target
32
33 - name: run node project
34 shell: npm start > /home/ansible-control/logs/npm-start.log 2>&1 &
35 args:
36   chdir: /home/ansible-control/pharmacy-app/ISA-Frontend
```

Slika 4.14

Nakon uspešnog izvršavanja ovih koraka, aplikacija je isporučena i konačno joj je moguće pristupiti putem porta 4200, na IP adresi veb-server mašine.

## 4.4 Rezultati istraživanja

Cilj istraživanja bio je proučiti primenljivost programa Ansible za brzo postavljanje okruženja i isporuku veb-aplikacije. Za studiju slučaja izabrana je Pharmacy aplikacija. U svrhe ove studije slučaja razvijena su dva Ansible „playbook-a“, kao i tri role. Rezultati istraživanja pokazali su jasnu prednost ovakvog pristupa u odnosu na ručna podešavanja i pisanje „bash“ skripti.

Prilikom pokretanja ove skripte za potpunu isporuku uz postavljanje konfiguracije okruženja, od početka do kraja potrebno je od pet do petnaest minuta, u zavisnosti od stanja u kome se sistem prvi put nalazi, budući da skripta ažurira sve pakete sistema na najnovije dostupne verzije. Ukoliko bi se odvažilo da se ovaj proces odradi manuelno, on bi iziskivao vreme od bar nekoliko sati, a to pod uslovom da je osoba koja proces izvršava iskusna u postavljanju potrebnih konfiguracija. Kada se uzme u obzir postavljanje većeg broja servera, od broja servera iskorišćenih u ovoj studiji slučaja, razlike u performansama manualnog postavljanja i postavljanja korišćenjem Ansible alata, postaju još drastičnije.

Takođe, dodatna prednost ovakve prakse je u tome što se jednom napisana skripta može izvršavati iznova i iznova, i uz određene izmene moguće ju je prilagoditi za isporuku aplikacije na veliki broj servera“, ili se ona ipak može prilagoditi i „deploymentu“ drugih aplikacija.

## 5. Zaključak

U ovom radu izložen je koncept principa infrastrukture kao koda, problemi koje rešava, kao i njegove prednosti. Takođe, detaljno je obrađen Ansible alat koji ovaj princip omogućava. Napravljen je osvrt na motivaciju za nastanak i istorijat ovog alata i izloženi su svi njegovi osnovni koncepti, koji je neophodno razumeti kako bi se alat mogao uspešno primeniti u različitim slučajevima korišćenja. Zatim su opisani slučajevi korišćenja za koje ovaj alat nudi rešenje i takođe je napravljen osvrt na prednosti i mane ovog alata.

Studija slučaja pokazala je primenu ovog alata u svrhe „deploymenta“ jedne aplikacije, izradom Ansible skripti u skladu sa potrebama problema. Studija je jasno pokazala prednosti korišćenja ovog alata u odnosu na manuelni proces konfiguracije sistema i „deployment-a“ aplikacije.

Međutim, iako je ovo istraživanje bilo fokusirano na Ansible alat, potrebno je naznačiti da Ansible alat nije jedini alat ove namene. Iako su prednosti ovog alata sada već jasno izložene, prilikom primene IaC principa, treba uzeti u obzir i alternative ovog alata, kao što su Puppet, Chef i SaltStack. Svi ovi alati, namenjeni upravljanju konfiguracijama, sa sobom donose kako prednosti, tako i mane.

Prilikom odabira alata, bilo bi najbolje prvenstveno uvideti konkretne potrebe i zadatke koje alat treba ispuniti, a potom i sagledati koji alat bi na najefikasniji način doveo sistem u to željeno stanje. Međutim, alternative Ansible alatu, kao i vodič za izbor pravog alata, morala bi biti tema za sebe.

## 6. Literatura

- [1] Infrastruktura kao kod. Redhat. <https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac>. Pristupljeno: 23.05.2021.
- [2] Infrastruktura kao kod. IBM. <https://www.ibm.com/cloud/learn/infrastructure-as-code>. Pristupljeno: 23.05.2021.
- [3] Infrastruktura kao kod. „Stackify“. <https://stackify.com/what-is-infrastructure-as-code-how-it-works-best-practices-tutorials>. Pristupljeno: 23.05.2021.
- [4] Infrastruktura kao kod. Alati. <https://medium.com/cloudnativeinfra/when-to-use-which-infrastructure-as-code-tool-665af289fbde>. Pristupljeno: 23.05.2021.
- [5] Ansible. „Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way“, Lorin Hochstein i Rene Moser, 2017.
- [6] Ansible. „Ansible Configuration Management“, Daniel Hall, 2013.
- [7] Ansible. „Learning Ansible“, Madhurranjan Mohaan i Ramesh Raithatha, 2014.
- [8] Ansible. „Flexera State of Cloud“, izveštaj iz 2019.  
<https://resources.flexera.com/web/media/documents/rightscale-2019-state-of-the-cloud-report-from-flexera.pdf>. Pristupljeno: 23.05.2021.
- [9] Ansible. Majkl Dehan intervju.  
<https://web.archive.org/web/20121114031927/http://www.coloandcloud.com/editorial/an-interview-with-ansible-author-michael-dehaan/>. Pristupljeno: 23.05.2021.
- [10] Koreni Ansible alata, Majkl Dehan.  
<https://www.ansible.com/blog/2013/12/08/the-origins-of-ansible>. Pristupljeno: 23.05.2021.

- 
- [11] Ansible. Instalacija.  
[https://docs.ansible.com/ansible/latest/installation\\_guide/intro\\_installation.html](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html).  
Pristupljeno: 23.05.2021.
- [12] Ansible. Slučajevi korišćenja. <https://www.ansible.com/use-cases>.  
Pristupljeno: 23.05.2021.
- [13] Ansible. NASA studija slučaja. <https://www.ansible.com/hubfs/pdf/Ansible-Case-Study-NASA.pdf?hsLang=en-us>. Pristupljeno: 23.05.2021.
- [14] Ansible. Microsoft studija slučaja.  
<https://www.redhat.com/en/resources/microsoft-case-study>. Pristupljeno: 23.05.2021.
- [15] Ansible. BMW studija slučaja. <https://www.redhat.com/en/success-stories/bmwgroup>. Pristupljeno: 23.05.2021.
- [16] Ansible. Edureka. <https://www.edureka.co/blog/what-is-ansible/>. Pristupljeno: 23.05.2021.
- [17] Ansible. Osnovni koncepti.  
[https://docs.ansible.com/ansible/latest/network/getting\\_started/basic\\_concepts.html](https://docs.ansible.com/ansible/latest/network/getting_started/basic_concepts.html).  
Pristupljeno: 23.05.2021.
- [18] „Ansible Galaxy“. Ansible. <https://docs.ansible.com/ansible/latest/cli/ansible-galaxy.html>. Pristupljeno: 23.05.2021.
- [19] „Ansible Galaxy“. Redhat. <https://www.redhat.com/sysadmin/ansible-galaxy-intro>. Pristupljeno: 23.05.2021
- [20] „Ansible Tower“. Ansible. <https://www.ansible.com/products/tower>.  
Pristupljeno: 23.05.2021.
- [21] „Ansible Tower“. Ansible Dokumentacija.  
[https://docs.ansible.com/ansible/latest/reference\\_appendices/tower.html](https://docs.ansible.com/ansible/latest/reference_appendices/tower.html). Pristupljeno: 23.05.2021.

## 7. Prilog

### P. Repozitorijum studije slučaja

<https://github.com/milanovicn/diplomski>

#### P.1. „Inventory“ datoteka

<https://github.com/milanovicn/diplomski/blob/master/studija-slucaja/Environments/inventory>

#### P.2. „Requirements“ datoteka

<https://github.com/milanovicn/diplomski/blob/master/studija-slucaja/requirements.yaml>

#### P.3. „Playbook“ za postavljanje veb-servera

[https://github.com/milanovicn/diplomski/blob/master/studija-slucaja/deploy\\_webservers.yaml](https://github.com/milanovicn/diplomski/blob/master/studija-slucaja/deploy_webservers.yaml)

#### P.4. „Playbook“ za postavljanje servera baze podataka

[https://github.com/milanovicn/diplomski/blob/master/studija-slucaja/deploy\\_dbservers.yaml](https://github.com/milanovicn/diplomski/blob/master/studija-slucaja/deploy_dbservers.yaml)

#### P.5. Skripta za pokretanje „deployment-a“

<https://github.com/milanovicn/diplomski/blob/master/studija-slucaja/run-deployment.sh>

#### P.6. Taskovi za postavljanje „base“ role

<https://github.com/milanovicn/diplomski/blob/master/studija-slucaja/roles/base/tasks/main.yaml>

#### P.7. Taskovi za postavljanje „dbservers“ role

<https://github.com/milanovicn/diplomski/blob/master/studija-slucaja/roles/dbserver/tasks/main.yaml>

### P.8. Taskovi za postavljanje projekta na veb-serveru

<https://github.com/milanovicn/diplomski/blob/master/studija-slucaja/roles/webserver/tasks/configure-git-project.yaml>

### P.9. Taskovi za postavljanje „websevers“ role

<https://github.com/milanovicn/diplomski/blob/master/studija-slucaja/roles/webserver/tasks/main.yaml>