Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/03BE18Gagovic

UPOTREBA SAVREMENIH METODA MODERNIZACIJE POSTOJEĆIH ASP.NET APLIKACIJA SA CILJEM RADA NA CLOUD PLATFORMI

USAGE MODERN METHODS FOR MODERNIZATION EXISTING ASP.NET APPLICATION WITH GOAL OF WORKING ON THE CLOUD PLATFORM

Miloš Gagović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – PRIMENJENO SOFTVERSKO INŽENJERSTVO

Kratak sadržaj – *Ovaj rad ima za cilj da prikaže upotrebu microservisne arhitekture u moderrnizaciji monolitskih ASP.NET aplikacija, sa osvrtom na Cloud Native koncepte razvoja, i upotrebu Docker kontejnera kao jedinicu isporuke servisa.*

Ključne reči:, *Doker, Mikroservis, ASP.NET, Servis Fabrik klaster, Azure*

Abstract – *This paper aims to demonstrate the use of microservice architecture in the modernization of monolithic ASP.NET applications, with a focus on Cloud Native development concepts, and the use of Docker containers as a service delivery unit.*

Keywords: *Docker, Microservice, ASP.NET, Service Fabric cluster, Azure*

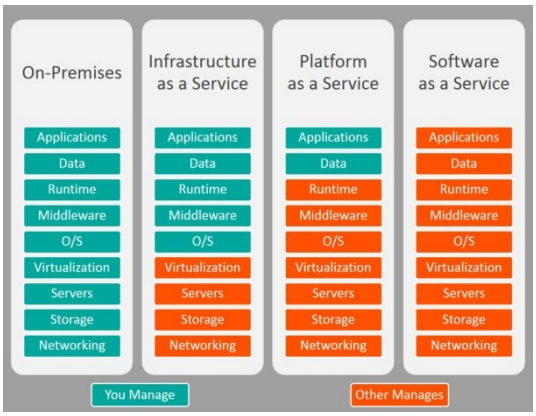
1. UVOD

Svedoci smo da je današnje korišćenje aplikacija baziranih na WWW tehnologijama gotovo nemoguće zamisliti bez upotrebe termina cloud computing (računarstvo u oblaku). Moderni razvoj aplikacija podrazumeva korišćenje tehnologija zasnovanih na „cloudu“, tj platformama i infrastrukturi koje nude cloud provajderi.
Zbog ubrzanog rasta i razvoja IT industrije javlja se potreba za pristupačnim, sigurnim, i pouzdanim softverom koji se lako održava i brzo isporučuje. U vezi sa tim, u daljem tekstu dat je pregled mogućnosti koje pruža cloud computing, kao i način dizajniranja, migracije i modernizacije aplikacije u cilju optimalnog iskorišćenja računarskih resursa.

2. RAČUNARSTVO U OBLAKU

Računarstvo u oblaku (cloud computing) predstavlja deljenje i korišćenje dostupnih računarskih resursa preko interneta. Velike kompanije kao što su Google , Microsoft i Amazon poseduju cloud infrasrukturu koju nude svojim korisnicima u zamenu za novac.
U zavisnosti od vrste korisnika, kao i od namene resursa, cloud computing možemo podeliti na četiri osnovne grupe, gde prva predstavlja deljenje resursa na zahtev korisnika, a bez učešća provajdera (Slika 1) :

- IaaS (infrastructure as a service) [1] je servis koji pruža korisniku odabir čitave vitruelizovane infrastrukture operativnog sistema, njegovo održavanje i ažuriranje. Cloud provajder je taj koji obezbeđuje hardver na kom će biti smeštena infrastruktura, i osigurava rad računarske mreže. Najčešći korisnici ovakvih servisa jesu administratori.
- PaaS (platform as a service) je servis koji pruža korisniku platformu za razvoj, odnosno nudi mu razvojno okruženje za njegove aplikacije i servise. Provajder je zadužen za ispravnost infrastrukture na koju se platforma oslanja, tj da obezbedi operativni sistem i čitav hardver na kome se on izvršava. Najčeći korisnici ovakvih servisa jesu developeri, a kao primer PaaS je Microsoft Azure. U daljem radu biće najviše fokusa posvećeno razvoju aplikacija korišćenjem ove cloud usluge.



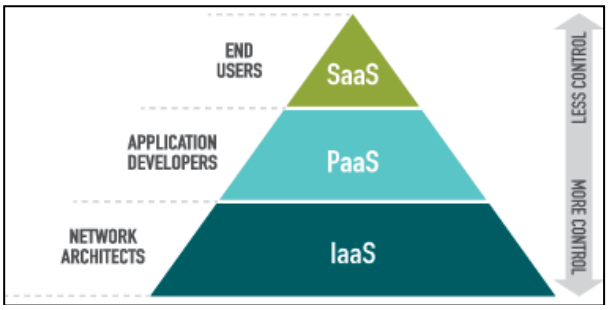
Slika 1. – *Podela odgovornosti korisnika na cloudu*

- SaaS (Software as a service) je servis koji nudi krajnjem korisniku korišćenje gotovih aplikacija u vidu servisa, a za kompletan njihov rad zadužen je provajder.

Na slici 2 prikazana je raspodela korisnika i njihovih odgovornost prilikom korišćenja servisa na oblaku.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Branko Milosavljević, red. prof.

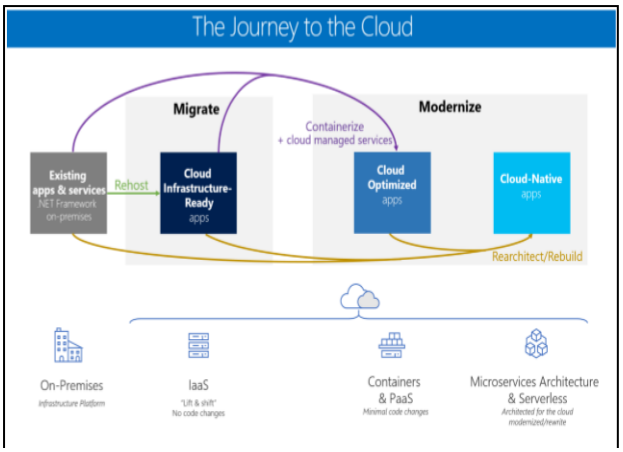


Slika 2. – *Raspodela korisnika po vrstama servisa*

3. MIGRACIJA APLIKACIJE NA „CLOUD“

Kada se donese odluka da se modernizuje web aplikacija ili usluge, i prebace na oblak, ne mora se nužno potpuno re-arhitektovati aplikacija. Ponovna arhitektura aplikacije pomoću naprednog pristupa kao što su mikroservisi nije uvek opcija zbog ograničenja troškova i vremena. U zavisnosti od vrste aplikacije, ponovna arhitektura aplikacije možda neće biti neophodna. Na to najviše utiče početni dizajn i kompleksnost, kao i vreme utrošeno na ponovnu izradu iste.

U tu svrhu izdvojena su tri nivoa migracija sistema na cloud koji predstavljaju razvojni put aplikacije u zavisnosti od stepena razvijenosti (slika 3).



Slika 3. *Migracioni nivoi ka cloudu*

Nivo 1 : Cloud infrastructure ready (aplikacije spremne za oblak). U ovom pristupu migracije aplikacija se jednostavno premesti ili se ponovo pokrenu lokalni servisi na infrastrukturi kao servisnoj platformi (IaaS). Aplikacija ima gotovo istu kompoziciju kao i ranije, ali sada je postavljena na virtuelne mašine na oblaku. Ova jednostavna vrsta migracije je u industriji poznata kao "Lift & Shift".

Nivo 2: Aplikacije oprimizovane za cloud. Na ovom nivou se još uvek može preći na cloud bez ponovnog redizajna ili značajne izmene koda. Ovo se može postići upotrebom savremenih tehnologija poput kontejnera i dodatnih usluga koje nudi provajder. Na ovaj način se poboljšava agilnost u razvoju aplikacija i skraćuje se vreme potrebno za njihovu isporuku. To se postiže korišćenjem tehnologija kao što su Windows kontejneri, koji se zasnivaju na Docker Engine-u. Kontejneri obezbeđuju izolovanu sredinu u kome se izvršava aplikacija, i postavljaju se na IaaS ili PaaS.

Dodatna prednost ovakvakvog načina razvoja ogleda se u poboljšanju kontinualne integracije/ kontinualne isporuke (CI / CD) softvera, što ga čini konkurentnijim na tržištu.

Nivo3: Cloud-Native aplikacije. Ovaj migracioni pristup obično podrazumeva poslovne potrebe i cilja na modernizaciju aplikacija sa kritičnom misijom. Na ovom nivou, koriste se PaaS usluge za premeštanje aplikacija na PaaS računarske platforme. Primenjuje se arhitektura mikroservisa kako bi evoluirali aplikacije, povećali agilnost i pomerili granice razvoja. U većini slučajeva se piše novi kod, posebno kada su aplikacije zasnovane na mikroservisima. Ovaj pristup može pomoći pri dobijanju pogodnosti koje se teško postižu u monolitnom i lokalnom aplikacijskom okruženju.

Na osnovu navedenih migracionih pristupa na cloud, uvode se pojmovi kontejnera, Docker Engine-a, kao i mikroservisne arhitekture. U cilju prikaza pogodnosti migracionih postupaka postojeća monolitna ASP.NET aplikacija sa WEB API dizajnom biće premeštena na cloud. Koristiće se optimizovan pristup koji se zasniva na kontejnerima, i cloud nativ pristup baziran na mikroservisnoj arhitekturi.

4. DOCKER I DOCKER KONTEJNERI

Docker je kompjuterski porgram otvorenog koda koji automatizuje primenu aplikacija kao prenosivih, samoodrživih kontejnera koji mogu da se pokreću u oblaku ili na lokalnim serverima. Docker [2] je kompanija koja promoviše i razvija ovu tehnologiju. Kompanija radi u saradnji sa cloud provajerima (uključujući i Microsoft), a podržava operativne sisteme kao što su Linux i Windows.

Šta su kontejneri? Kontejneri (Linux ili Windows) su način za izvršavanje aplikacija u sopstvenom izolovanom procesu. U svom kontejneru aplikacije ne utiču na aplikacije ili procese koji postoje izvan kontejnera. Sve od čega aplikacija zavisi pokreće se kao proces unutar kontejnera. Gde god se kontejner premesti, zahtevi aplikacije za referencama će uvek biti ispunjeni, u smislu direktnih zavisnosti, jer dolazi u kompletu sa svime što je potrebno da radi (reference na biblioteke, vreme trajanja ...).

Glavna karakteristika kontejnera je ta što okruženje čini istim kroz različite faze isporuke, jer sam kontejner ima sve potrebne zavisnosti. To znači da možete pokretati aplikaciju na vašoj mašini, a zatim je isporučiti na drugu, i dobiti jednako okruženje za rad.

Kontejner je instanca slike (Image) kontejnera . Slika kontejnera je način za pakovanje aplikacije ili usluge, a zatim ga rasporediti na pouzdan i ponovljiv način. Moglo bi se reći da Docker nije samo tehnologija - to je i filozofija i proces. Pošto kontejneri svakodnevno postaju sve češći u upotrebi, oni postaju industrijska "jedinica raspoređivanja" (unit of deployment).

Za nekoga ko je upoznat sa virtuelnim mašinama, izgleda da su kontejneri izuzetno slični. Kontejner pokreće operativni sistem, ima sistem datoteka i može mu se pristupiti preko mreže, baš kao i fizički ili virtuelni računarski sistem.

Međutim, tehnologija i koncepti iza kontejnera su znatno različiti od virtuelnih mašina. Sa stanovišta programera, kontejner mora biti tretiran više kao jedan proces.

Zapravo, kontejner ima jednu ulaznu tačku za jedan proces. Docker kontejneri mogu raditi na Linux i Windows operativnim sistemima.

Kada se koriste redovni kontejneri, Windows kontejneri se mogu pokrenuti samo na Windows hostovima (host server ili VM), dok se Linux kontejneri mogu pokrenuti samo na Linuksovim hostovima.

Međutim, u najnovijim verzijama Windows servera i Hiper-V kontejnera, kontejneri za Linux se takođe mogu pokrenuti na Windows Server-u koristeći Hiper-V tehnologiju izolacije koja je trenutno dostupna samo na Windows Server kontejnerima. U bliskoj budućnosti, mešovita okruženja koja imaju i Linux i Windows kontejnere biće moguća, čak i česta.

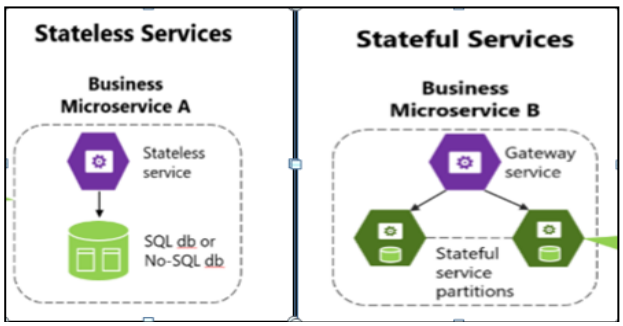
4.1. Poređenje kontejnera i virtuelne mašine

Na slici 4 prikazano je poređenje rada virtuelnih mašina i docker kontejnera. Virtuelne mašine uključuju aplikacije i potrebne biblioteke za njihov rad, kao i čitav operativni sistem na kome se one izvšavaju.

Suprotno tome, kontejneri se izvršavaju kao zasebni (izolovani) procesi koje kontroliše container engine. Svi procesi dele resurse operativnog sistema na kome su pokrenuti, a njihovo pokretanje traje znatno kraće. Zbog toga što zahtevaju manje resursa (ne treba im ceo operativni sistem) lakše se isporučuju i pružaju mogućnost pokretanja više servisa korišćenjem istog hardvera.

Neželjeni efekat pokretanja na istom operativnom sistemu je taj da se postiže manja izolacija procesa u odnosu na virtuelne mašine.

Za Docker se može reći da nije samo tehnologija, već i filozofija razvoja softvera. Upotrebom docker kontejnera izbegava se česta sintagma kod developera „Program fukcioniše na mojoj mašini, zašto ne radi u produkciji“. Ovo se jednostavno izbegava i dovoljno je reći da radi upotrebom dockera. Docker aplikacija se može izvršiti na bilo kom docker okruženju, i izvršiće se na načinu za koji je namenjena (DEV, QA, Staging, Product).



Slika 4. *Poređenje virtuelne mašine i docker kontejnera*

Kao osnovna jedinica isporuke u oba migraciona modela (Cloud Optimized i Cloud Native) odabrani su docker kontejneri zbog optimalnog iskorišćenja računarskih resursa.

5. MIKROSERVISNA ARHITEKTURA

Razumevanje mikroservisa i njihov način rada su važni prilikom planiranja o premeštanju softvera na cloud.

Arhitektura mikroservisa je napredni pristup koji se može koristiti za aplikacije koje su kreirane od nule ili rezvijane primenom cloud native principa. Razvoj ovakvog tipa softvera iziskuje poseban dizajn kao i ponovnu implementaciju.

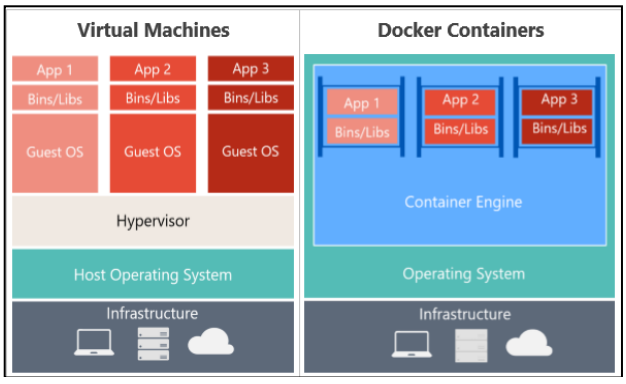
Međutim, mikroservisi nisu obavezni za bilo kakvu novu ili modernu aplikaciju. Mikroservisi [3] ne predstavljaju idealno rešenje, i nisu samo jedan, najbolji način za stvaranje svake aplikacije. Kako i kada koristite mikroservise zavisi od vrste aplikacije koju treba izgraditi.

Arhitektura mikroservisa postaje preferirani pristup za distribuirane i velike ili kompleksne kritične aplikacije koje se zasnivaju na višestrukim, nezavisnim podsistemima u obliku autonomnih usluga. U arhitekturi zasnovanoj na mikroservisima, aplikacija je izgrađena kao skup usluga koji se može samostalno razvijati, testirati, verifikovati, raspoređivati i skalirati. Ovo može uključivati bilo koju povezanu, autonomnu bazu podataka po mikroservisu.

5.1 Podela mikroservisa prema nameni

Podela mikroservisa implementiranih u Service Fabric [4] tehnologiji je na Stateless (bez pamćenja stanja) i Statefull (imaju sposobnost pamćenja stanja). Svaki mikroservis mora posedovati sopstveni domenski model podataka, i ne sme biti nezamenljiv. Na slici 5 prikazane su šeme ove dve vrste mikroservisa.

Glavna osobina statefull servisa u odnosu na stateless je ta da se podaci repliciraju preko više nezavisnih instanci servisa. Okruženje u kome se vrši praćenje replikacije i distribucije podataka obezbeđuje platforma za orkestriranje servisa. Service Fabric klaster kao Microsoftov predstavnik nudi mogućnost praćenja rada servisa, kao i prikaz stanja servisa prilikom rada.



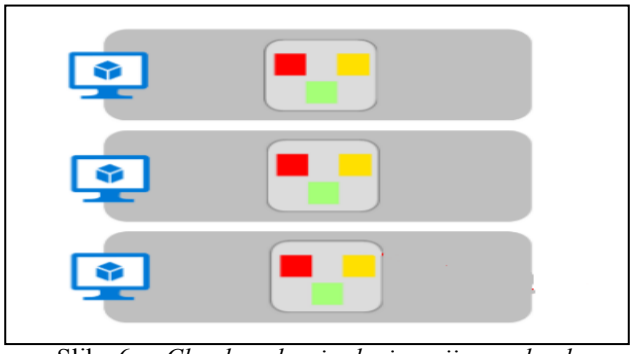
Slika 5. – *Prikaz razlike između mikroservisa*

Pristup kreiranju stateless servisima je lakši za implementaciju. Ovakav način je sličan tradicionalnim obrascima za kreiranje. Podaci se čuvaju u eksternoj bazi i nema replikacije po particijama. Nasuprot tome, statefull servisi se „podižu“ na oblak u vidu više nezavisnih particija. Jedna od njih predstavlja primarnu, dok se na sve ostale podaci repliciraju i imaju mogućnost da postanu primarne ukoliko se desi nepredviđen ispad. Sadrže pomoćni microservis u vidu gateway-a koji balansira saobraćaj između particija, i u trenutku ispada menja namenu particijama.

6. IMPLEMENTACIJA MIGRACIONIH REŠENJA I POREĐENJE PERFORMANSI RADA

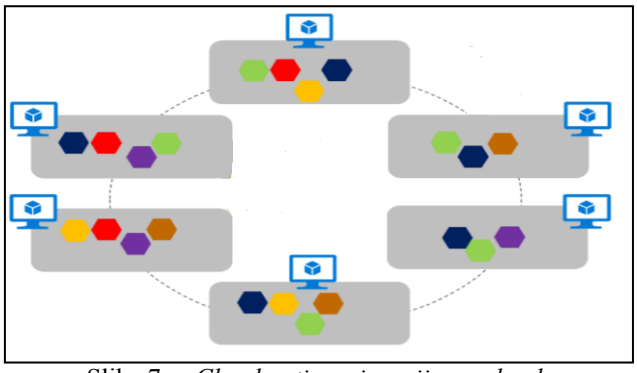
Za potrebe implementacije migracionih rešenja iskorišćena je ASP.NET aplikacija i WEB API tehnologija. WEB API je odabran iz razloga što se razmena podataka obavlja u JSON formatu upotrebom HTTP protokola, i mogu mu pristupati sve aplikacije koje podržavaju ovaj protokol. Kao monolitna ASP.NET aplikacija upotrebljen je servis za rezervaciju smeštaja u hotelima. Primenom migracionih postupaka Cloud Modernized i Cloud Native došlo se do rešenja pogodnog za izvršavanje na cloud platformi. U nastavku će biti prikazane pogodnosti koje ova rešenja nude, kao i razlike u kompleksnosti dizajna i arhitekture.
Rešenje implementirano upotrebom mikroservisa i servis fabric klastera zasniva se na upotrebi više instanci mikroservisa postavljenih na virtuelnim mašinama na cloudu, ali tako da svaka instanca predstavlja jedan proces unutar doker kontejnera.

Na slici 6 prikazan je izgled aplikacije na cloudu upotrebom Cloud Modernized metode. Uočavaju se tri nezavisna servisa postavljena na tri virtuelne mašine u vidu doker kontejnera, s tim da komunikacija između servisa na različitim mašinama nije ostvarena. Na ovaj način smo relativno lako preneli čitavu arhitekturu na cloud, gubeći pri tom mogućnost skalabilnosti, i smanjujući stepen otpornosti na otkaz.



Slika 6. – *Cloud modernized migracija na cloud*

Suprotno ovome, na slici 7 prikazana je arhitektura rešenja upotrebom mikroservisa u obliku kontejnera, i service fabric klastera kao orkestratora. Svaka instanca servisa postavljena je na više različitih okruženja, između kojih postoji komunikacija preko API-a. Pomoću njega se razmenjuju i repliciraju podaci.
U zavisnosti od potrebe, klaster u kome su smešteni zahtevaće veću ili manju promenu u skaliranju procesa unutar kontejnera. To znači da im obezbeđuje onoliko resursa koliko je stvarno potrebno za njihov rad, ili čak da kreira nove ili gasi stare procese.
Ovo je od velikog značaja prilikom rada na cloudu, jer se cena rada formira na osnovu iskorišćenih resursa. Postiže se veći stepen otpornosti na otkaze, što je jedan od preduslova za dobar i siguran rad distribuiranih aplikacija.



Slika 7. – *Cloud native migracija na cloud*

7. ZAKLJUČAK

Fokus ovog rada baziran je na primeni migracionih tehnika pomoću kojih se tradicionalne ASP.NET aplikacije mogu postaviti na PaaS ili SaaS okruženje na cloudu. Za tu svrhu upotrebljen je koncept docker kontejnera i mikroservisna arhitektura.

Upotrebom ovih tehnologija, značajno se obezbeđuje stabilan i optimalan rad servisa, kao i mogućnost daljeg rada na razvoju novih funkcionalnosti. Odluka da se software podigne na cloud zahteva veća zalaganja ako su u pitanju mikroservisi, ali obezbeđuje duži i pouzdaniji rad, kao i veću konkurentnost na tržištu.

8. LITERATURA

[1] Cesar de la Torre, "Modernize existing .NET applicatios with Azure cloud and Windows Containers", Microsoft Corporation, 2018.
[2] Cesar de la Torre, Containerized Docker Application Lifecycle with Microsoft platform and Tools", Microsoft Corporation, 2018.
[3] Cesar de la Torre,Bill Wagner, Mike Rousos ".NET Microservices: Architecture for Containerized .NET Applications" , Microsoft Corporation, 2018.
[4] https://msdn.microsoft.com/en-us/library/ms971499.aspx, datum pristupa septembar 2018.

Kratka biografija:

Miloš Gagović rođen je u Vrbasu 24.2.1994. Diplomirao je na Fakultetu tehničkih nauka na smeru Elektroenergetski softverski inženjering 2017. god. Master rad iz oblasti Elekroenergetski softverski inženjering odbranio je 2018. god. Kontakt: gagovicmilosgagovic@gmail.com