Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/17BE32Grubor

PRIKAZ DEVOPS PRAKSI U AZURE OKRUŽENJU

AZURE DEVOPS IN PRACTICE

Maja Grubor, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *Ovaj rad se bavi DevOps-om, sa fokusom na Azure DevOps i svim servisima koje on nudi. Opisuju se i još neki DevOps alati. Srž rada jeste deployment više verzija jednog softvera upotrebom Azure DevOps-a.*

Ključne reči: *DevOps, Azure, Azure DevOps, build, release, pipeline, deployment*

Abstract – *Main goal of this paper is to present DevOps with focus on Azure DevOps and services which it offers. Here is described other Dev Ops tools. The core of the work is deployment more then one software versions via Azure DevOps.*

Keywords: *DevOps, Azure, Azure DevOps, build, release, pipeline, deployment*

1. UVOD

DevOps je skup praksi koje rade na automatizaciji i integraciji procesa između razvoja softvera i IT timova, tako da oni mogu brže i pouzdanije graditi, testirati i isporučivati softver. Termin "DevOps" nastao je kombinovanjem reči "development" (razvoj) i "operations" (operacije) te označava praksu razvoja softvera koja objedinjuje razvoj i IT operacije kako bi omogućila kontinuiranu isporuku korisnicima. DevOps uključuje osnovne prakse kao što su planiranje, praćenje, razvoj, izgradnja, testiranje, isporuka i nadgledanje. DevOps prakse održavaju ideju stalnog poboljšanja i automatizacije. Mnoge prakse fokusiraju se na jednu ili više faza razvojnoj ciklusa. Agilni razboj je značajan deo DevOps-a i imao je značajan uticaj na njegovo stvaranje. Nije uobočajeno a takođe se ni ne preporučuje koristiti agilne metodologije bez upotrebe DevOps-a.

2. DEVOPS I ŽIVOTNI CIKLUS APLIKACIJE

DevOps utiče na životni ciklus aplikacije tokom celog njenog plana, razvoja, isporuke i rada. Svaka faza se odnosi na ostale faze, a one nisu specifične za uloge. U pravoj DevOps kulturi svaka uloga je donekle uključena u svaku fazu [1].

• Plan - U fazi planiranja DevOps timovi osmišljavaju, opisuju i definišu karakteristike i mogućnosti aplikacija i sistema koje grade. Oni prate napredak na niskom i visokom nivou detaljnosti – od zadataka pojedinačnih proizvoda do zadataka koji obuhvataju više proizvoda [2].

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Željko Vuković, docent.

• Develop - Faza razvoja uključuje sve aspekte kodiranja – pisanje, testiranje, pregled i integraciju koda svih članova tima. DevOps timovi nastoje da brzo unose inovacije bez smanjenja kvaliteta, stabilnosti i produktivnosti [2].

• Deliver - Isporuka predstavlja proces postavljanja aplikacija u proizvodna okruženja na dosledan i pouzdan način. Faza isporuke takođe uključuje primenu i konfigurisanje infrastrukture. U fazi isporuke timovi definišu proces upravljanja sa jasnim fazama [2].

• Operate - Faza rada uključuje održavanje, nadgledanje i rešavanje problema u aplikacijama u proizvodnom okruženju. Timovi rade na tome da obezbede pouzdanost sistema, veliku dostupnost i teže ka tome da imaju nulti zastoj uz jačanje bezbednosti i upravljanja sistema. Timovi imaju u cilju da identifikuju probleme i ublaže ih ili otklone u potpunosti pre nego što na njih naiđe korisnik [2].

3. DEVOPS ALATI

Bez odgovarajućih alata i tehnologija DevOps nije održiv model. Oni omogućavaju automatizaciju, kontinuiranu integraciju i isporuku, upravljanje konfiguracijom, testiranje, puštanje u produkciju i monitoring. U narednom tekstu biće navedeni neki od alata.

3.1. Jenkins

Jenkins je platforma otvorenog koda za kontinuiranu integraciju i kontinuiranu isporuku koja se može koristiti za automatizaciju svih vrsta zadataka u vezi sa izgradnjom, testiranjem i isporukom ili primenom softvera [3].

3.2. Kubernetes

Kubernetes je alatka otvorenog koda za upravljanje kontejnerima koja se naziva i alatka za orkestraciju kontejnera. Koristi se za stvari poput automatizacije postavljanja kontejnera, skaliranja i uklanjanja istih.
To nije platforma za kontejnerizaciju već alat koji se koristi na toj platformi. Kubernetes klasteri mogu obuhvatati hostove u lokalnom, javnom, privatnom ili hibridnom cloud-u. Iz tog razloga, Kubernetes je idealna plaforma za hosting aplikacija zasnovanih na cloud-u koje zahtevaju brzo skaliranje, poput strimova podataka u realnom vremenu [4].

3.3. Docker

Docker je softverska platforma koja omogućava brzo pravljenje, testiranje i primenu aplikacija. On pakuje softver u standardizovane jedinice (kontejnere) koji imaju sve što je potrebno softveru za pokretanje, uključujući biblioteke, sistemse alate i kod. Pomoću Docker-a mogu

se brzo deploy-ovati i skalirati aplikacije u bilo koje okruženje i znati da će se kod pokrenuti [5].

3.4. Git

Git je distribuirani SCM (source code management) alat koji koriste udaljeni timovi i korisnici otvorenog koda. Git omogućava praćenje i napredak rada kao i čuvanje različitih verzija izvornog koda i samim tim i vraćanje na neke od prethodinh verzija [6].

3.5. Azure DevOps

Azure DevOps pruža usluge timovima za planiranje rada, saradnju na razvoju koda i izradu i primenu aplikacija. Podržava skup procesa koji drži na okupu programere, menadžere projekta i saradnike da bi se završio razvoj projekta. Omogućava organizacijama da stvaraju i poboljšavaju proizvode brže nego što mogu sa tradicionalnim pristupima razvoju softvera [7].

4. AZURE

Azure je platforma kompanije Microsoft koja nudi širok spektar cloud servisa koji uključuju računarstvo, analitiku, skladištenje i umrežavanje. Ova platforma ima za cilj da pomogne raznim kompanijama da upravljaju izazovima i ispune svoje organizacione ciljeve.

4.1. Azure DevOps

Azure DevOps je Microsoft-ova platforma koja pruža end-to-end lanac alata za razvoj i primenu softvera. Takođe, integriše se sa većinom alata na tržištu i odlična je opcija za orkestriranje DevOps lanca alata. Za izradu aplikacije odabrana su razvojna okruženja Visual Studio i Visual Studio Code koja imaju integraciju sa Azure DevOps-om. Back-end deo rađen je u .NET Core tehnologiji i korišćen je C# programski jezik, dok je za front-end deo korišćen Angular i TypeScript. Ovo je aplikacija za menadžovanje zaposlenix u kompaniji. Za izradu aplikacije odabrana su razvojna okruženja Visual Studio i Visual Studio Code koja imaju integraciju sa Azure DevOps-om. Back-end deo rađen je u .NET Core tehnologiji i korišćen je C# programski jezik, dok je za front-end deo korišćen Angular i TypeScript. Ovo je aplikacija za menadžovanje zaposlenix u kompaniji.

4.1.1. Azure Boards

Pomoću Azure Boards veb usluge timovi mogu upravljati svojim softverskim projektima. Pruža bogat skzp mogućnosti uključujući izvornu podršku za Scrum i Kanban, prilagodljive kontrolne table i integrisano izveštavanje. Možemo brzo i jednostavno započeti praćenje user story-a (korisničkih priča), feature-a, backlog item-a i bug-ova vezanih za naš projekat [8].

4.1.2. Azure Repos

Azure Repos je skup alata za kontrolu verzija koje možemo koristiti za upravljanje kodom. Bez obzira da li se radi o malom ili velikom softevru, upotreba kontrole verzija je jako dobra ideja. Sistemi za kontrolu verzija su softveri koji nam pomažu da pratimo promene koje unesemo u kod tokom vremena [9].

4.1.3. Azure Pipelines

Azure Pipeline automatski build-uje i testira kod projekta kako bi ih učinili dostupnim drugima. Kombinuje kontinuiranu integraciju i kontinuiranu isporuku za testiranje i izradu koda i njeovo slanje na bilo koji target [10].

Continuous Integration (CI) – praksa koju koriste razvojni timovi za automatizaciju spajanja i testiranja koda. Implementacija CI pomaže u hvatanju grešaka u ranom razvojnom ciklusu, što ih čini jeftinijim za ispravljanje [10, 13].

Continuous Delivery (CD) – proces pomoću kog se kod gradi, testira i primenjuje u jedno ili više testnih ili proizvodnih okruženja. Primenom i testiranjem u više okruženja povećava se kvalitet. CI sistemi proizvode artifatke koji se mogu deploy-ovati, uključujući infrastrukturu i aplikacije [10, 13].

Continuous Testing (CT) – upotreba automatskih build-deploy-test tokova lokalno ili u cloud-u, sa izborom tehnologija i framework-a koji kontinuirano testiraju naše promene na brz, skalabilan i efikasan način [10].

4.1.4. Azure Test Plans

Azure DevOps pruža bogate i moćne alate koje svima u timu mogu koristiti za poboljšanje kvaliteta i saradnje tokom celog procesa. Rešenje za upravljanje testovima u pregledaču pruža sve mogućnosti potrebne za planiranje ručnog testiranja, istraživačkog testiranja i prikupljanja povratnih informacija od stakeholder-a [11].

4.1.5. Azure Artifacts

Omogućava programerima da dele i koriste pakete iz različitih izvora i javnih registara. Paketi se mogu deliti unutar istog tima, iste organizacije ili javno [12].

5. OPIS PROBLEMA I IMPLEMENTACIJA REŠENJA

Cilj ovog rada jeste ispratiti deployment nekog softvera, odnosno više verzija tog softvera. U narednim poglavljima biće detaljno objašnjeno kako je aplikacija implementirana i kako je uz pomoć Azure DevOps-a ona deploy-ovana u produkciono okruženje.

5.1. Opis korišćenih alata

Za izradu aplikacije odabrana su razvojna okruženja Visual Studio i Visual Studio Code koja imaju integraciju sa Azure DevOps-om. Back-end deo rađen je u .NET Core tehnologiji i korišćen je C# programski jezik, dok je za front-end deo korišćen Angular i TypeScript. Ovo je aplikacija za menadžovanje zaposlenix u kompaniji.

5.2. Kreiranje aplikacije u Azure DevOps-u

Pre svega, treba da imamo kreiran nalog na Azure DevOps-u kao i organizaciju u sklopu koje ćemo da napravimo projekat. Samo kreiranje projekta je prilično jednostavan korak. Potrebno je uneti ime projekta i označiti da li želimo da projekat bude javno dostupan ili privatan. Postoje još dva dodatna podešavanja koja se mogu odabrati, a to je da odaberemo Version Control agilni pristup želimo da koristimo.

5.3. Planiranje i kreiranje taskova upotrebom Azure Boards

Ono što bi sledeće trebalo biti urađeno kada se koristi ovaj pristup jeste da se isplanira posao, kreiraju zadaci i dodele programerima, testerima i drugim učesnicima.

U sklopu kartice Boards postoji nekoliko pod-kartica: backlog u kojoj možemo da vidimo sve kreirane taskove, sprint gde su prikazani samo taskovi koji spadaju u selektovanu iteraciju i još neke.

5.4. Postavljanje koda u repozitorijum

Sledeći korak jeste da se projekat koji postoji kreiran lokalno postavi u Azure DevOps repozitorijum. To može da se uradi na nekoliko načina a jedan od njih je da se to uradi kroz Visual Studio ili Visual Studio Code jer kao što je ranije spomenuto oni su integrisani sa Azure DevOps-om i Git-om. Ovde je prvo postavljanje koda rađeno kroz Visual Studio. Za početak, u VS treba biti ulogovan sa nalogom na kom je prethodno kreiran repozitorijum, zatim treba kliknuti desni klik na lokalni repozitorijum pa "Create Git repository" i otvoriće se prozor gde se može da kreirati novi repozitorijum ili iskoristiti postojeći. Kako ovde već postoji kreiran, potrebno je samo uneti njegov url.

5.5. Kreiranje Build Pipeline-a

Sledeći korak jeste konfigurisanje build pipeline-a. Potrebno je iskonfigurisati build proces za projekat koji je rađen u .NET Core-u i projekat rađen u Angular-u. Taj build može da se pokreće svaki put nakon što je dodat neki novi kod direktno na glavnu granu ali to baš i nije dobra praksa jer nismo sigurni da li naša izmena može prouzrokovati greške. Da bi se koknfigurisao build, treba otići na karticu Pipelines koja se nalazi u sklopu sekcije Repos i kliknuti na "Create pipeline".
Nakon što smo rekli da želimo da kreiramo pipeline treba da se odabere izvor gde se nalazi kod, projekat i repozitorijum koji želimo kao i granu za koju kreiramo konfiguraciju. Inicijalno se dobija jedan Agent Job u koji se dodaju taskovi koji će se redom izvršavati. Ovaj Agent Job će se koristiti za konfigurisanje .NET Core projekta i samim tim će se dodavati taskovi u skladu sa njegovim potrebama. Prvi task koji je dodat je task za instaliranje Nuget paketa. Potrebno je da se odabere verzija Nuget-a koju želimo da instaliramo. Ovde je odabrano da svaki put želimo da se proveri da li postoji novija verizija i ukoliko postoji da se ona instalira.
Naredni task koji je dodat jeste Nuget restore. Nakon što su dodati prethodni taskovi, može da se uradi build solution-a koristeći komandu .net build. Iako u ovoj fazi nisu implementirani testovi, dodaat je zadatak za to jer je planirano dodavanje testova u nekoj od narednih iteracija. Sledeći korak je izvršavanje dotnet publish-a kako bi se dobili konačni rezultati build-a. Poslednji korak u konfiguraciji serverske strane predstavlja publish-ovanje build artifakta, tako da release pipeline može da vidi šta je rezultat ovog build procesa.
Kada su svi taskovi potrebni za serverski deo dodati, može se preći na klijentski deo aplikacije. Postoje dve mogućnosti u ovoj fazi a to je da se u skolpu ovog pipeline-a doda novi Agent Job i dodaju taskovi potrebni za build Angular aplikacije ili da se kreira novi pipeline koji bi imao jedan Agent Job. Ovde nije kreiran novi pipeline već je dodat još jedan Agent Job u sklopu postojećeg pipeline-a.
Prvi task koji je dodat je npm install koji služi za instaliranje paketa. Sledeći dodat task je npm build. Nakon toga, kao i kod serverskog dela, dodat je task za publish-ovanje artifakta. Putanja koja je ovde konfigurisana je ista ona putanja iz prethodnog taska koja predstavlja njegov rezultat.
Sledeći korak bi bio kreiranje release pipeline-a uz pomoć kreiranih artifakta. Odlučeno je da se kreira AppService na Azure Portal-u i na njega hostuje naša aplikacija prilikom release-a.

5.6. Kreiranje App Service-a

Azure Portal je konzola bazirana na vebu koja pruža alternativu command-line alatima. Pomoću Azure Portal-a možemo uptavljati Azure subscription-ima pomoću grafičkog korisničkog interfejsa. Možemo da kreiramo, upravljamo, nadgledamo sve od najjednostavnije veb aplikacije do složenih cloud implementacija.
Azure App Service je HTTP servis za hostovanje veb aplikacija, REST api-a i back end dela mobilnih aplikacija.
Tokom kreiranja AppService-a treba da se unesu Subscription i Resource Group – predstavlja kontejner koji sadrži povezane resurse solution-a. Takođe, treba uneseti naziv veb aplikacije, da označiti da li se publish-uje kod ili Docker kontejner, selektovati programski jezik ili framework, region i još neke opcione podatke. Nakon što je kreiran AppService, dobija se url na kome će se nalaziti naša aplikacija nakon hostovanja [14].

5.7. Kreiranje Release Pipeline-a

Na Azure DevOps-u, u okviru sekcije Pipelines postoji karticu Release, klikom na nju dobija se stranica gde su prikazani trenutno postojeći release-ovi. Template koji je izabran u momentu kreiranja jeste Azure App Service Deployment koji se koristi za preuzimanje artifakta i postavljanja na Azure. Ovde će i klijentski i serverski deo biti deploy-ovani na isti server, ali je moguće da budu deploy-ovani na različite servere.
Prvi artifakt koji je dodat je serverski kod, pa je kao izvor izabrana build definiciju koja je dobijena iz artifakta kao izlaz.
Nakon što je kreiran prvi artifakt, klikom na "1 job, 1 task" može se otići na stranicu gde se može detaljnije videti i podesiti šta treba da se uradi sa artifatom. U tom dijalogu, treba da se odabere subscription i ime App Service-a na koji će se deploy-ovati naš kod.
Kao drugi artifakt odabran je dist artifakt iz klijentskog koda koji se proizvodi u drugoj fazi build-a. Što se tiče ovog artifakta, sva podešavanja su ostala ista kao i kod prvog osim izvora, koji mora biti drugačiji. Postoji mogućnost da se nakon svakog uspešno završenog build-a automatski pokrene release.
Nakon što je pokrenut kreirani build pipeline, a nakon njega i release, kada se ode na url App Service-a, trebalo bi da se tamo vidi naša aplikacija.

5.8. Re-deploy aplikacije

Da bi se uradio re-deploy aplikacije, potrebno je da postoje neke izmene u kodu, odnosno da postoji nova verzija build-a. Kada je iz lokala push-ovana grana i kod koji je dodat, na stranici Pull Requests biće prikazana ta grana kao i mogućnost kreiranja Pull Request-a za tu

granu. Nakon što se klikne na dugme "Create a pull request" dobija se stranica gde se mogu videti svi fajlovi koji su izmenjeni, koliko i koji commit-ovi su napravljeni. Takođe, može se povezati preko id-a sa feature-om ili bug-om na koji se odnosi. Bitna stvar u ovom koraku jeste odabrati granu u koju će se spojiti izmene nakon što se PR complete-uje. Nakon što je je odrađen review, svi komentari su ispravljeni, dovoljan broj review-era je approve-ovao PR, on može biti complete-ovan. Takođe, može se podesiti da se uradi build feature grane čim se kreira PR ili se desi neka izmena na PR-u, da kreator PR-a može da zna ukoliko build nije prošao uspešno da ne sme tu granu spojiti sa glavnom granom da ne bi došlo do grešaka.
U našem slučaju kao i u prethodnom poglavlju pokrenuće se Build Pipeline koji će nakon što se uspešno završi pokrenuti release i deploy-ovati novu verziju naše aplikacije. Može se automatizovati da svaki put kada se dogodi izmena na grani koju deploy-ujemo, da se pokrene build i tako svaki put dobijemo novu verziju softvera dostupnu.

6. ZAKLJUČAK

Cilj DevOps-a jeste da skrati životni ciklus razvoja softvera i da obezbedni kontinuiranu integraciju i kontinuiranu isporuku sa visokim kvalitetom softvera. Osnovni koncept DevOps-a CI/CD se sastoji iz kontinuirane integracije (CI), kontinuirane isporuke (CD) i kontinuiranog deployment-a (CD). Ova tri procesa zajedno automatizuju izgradnju, testiranje i primenu tako da DevOps timovi mogu brže i pouzdanije isporučiti kod. CI/CD procesi su fokusirani na automatizaciju isporuke softvera dok se DevOps takođe fokusira na organizacione promene kako bi podržao saradnju između svih uključenih funkcija.
Uz pomoć Azure DevOps-a možemo da ispratimo razvoj softvera od ideje, preko planiranja, organizacije, implementacije, automatizacije procesa do samog deployment-a softvera u produkciju. Takođe ono što je vrlo bitno jeste da on ima integraciju sa velikim brojem alata, ček i nekih ranije pomenutih.
Rešenje prikazano u ovom radu, moglo bi se unaprediti u nekim aspektima. Prva stvar bi bila prilikom kreiranja build pipeline-a, da se umesto jednog kreiranog, za svaki projekat kreira zaseban build pipeline jer bismo smanjili vreme izvršavanja. Često se dešava da se menja samo jedan projekat, te nema potrebe za pokretanje build-a svih projekata. Takođe i u slučaju da imamo više od dva projekta u repozitorijumu, bilo bi dobro da se za svaki projekat našravi poseban build pipeline. Nešto slično bi se moglo uraditi i za release, gde bismo razdvojili release klijentskog i serverskog dela.

7. LITERATURA

[1] https://azure.microsoft.com/en-us/overview/what-is-devops/#devops-overview (pristupljeno u junu 2021.)

[2] https://azure.microsoft.com/en-us/overview/what-is-devops/#culture (pristupljeno u junu 2021.)

[3] jenkins.io (pristupljeno u julu 2021.)

[4] https://kubernetes.io/ (pristupljeno u julu 2021.)

[5] https://aws.amazon.com/docker/ (pristupljeno u julu 2021.)

[6] https://git-scm.com/ (pristupljeno u julu 2021.)

[7] https://docs.microsoft.com/ (pristupljeno u julu 2021.)

[8] https://docs.microsoft.com/en-us/azure/devops/boards/get-started/what-is-azure-boards?view=azure-devops&tabs=agile-process (pristupljeno u junu 2021.)

[9] https://docs.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops (pristupljeno u junu 2021.)

[10] https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops (pristupljeno u junu 2021.)

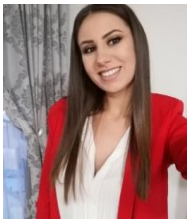
[11] https://docs.microsoft.com/en-us/azure/devops/test/overview?view=azure-devops (pristupljeno u junu 2021.)

[12] https://docs.microsoft.com/en-us/azure/devops/artifacts/start-using-azure-artifacts?view=azure-devops (pristupljeno u junu 2021.)

[13] AZ-400 (Handbook) Implementing Continuous Integration, Implementing Continuous Delivery

[14] AZ-203 (Handbook) Develop Azure Platform as a Service compute solutions

Kratka biografija:



Maja Grubor rođena je 27.04.1995. god. u Drvaru. Završila je osnovnu školu „Sveti Sava" u Bačkoj Palanci kao odličan učenik. Gimnaziju „20. Oktobar" završava 2014. godine kao odličan učenik, tokom čijeg pohađanja je učestvovala na takmičenjima iz matematike. Iste te godine upisuje Fakultet tehničkih nauka, smer Elektroenergetski softverski inženjering. U septembru 2018. završava osnovne studije, nakon čega upisuje master studije takođe na Fakultetu Tehničkih nauka, smer Računarstvo i Automatika, Elektronsko poslovanje.