DevOps
kontinuirana integracija
(Jenkins)

Šta je kontinuirana integracija

Kontinualna integracija je praksa u softverskom inžinjeringu u kojoj se teži ka tome da se male izmene u kodu integrišu u repozitorijum često u cilju ranog otkrivanja grešaka i bržeg razvoja.

Amazon mijenja svoje proizvodno okruženje svakih 11,6 sekundi. Facebook menja svoju web stranicu najmanje nekoliko puta svaki dan. Što se tiče razvoja softvera, izdanja u ovoj nevjerojatnoj kadenci omogućena su samo zahvaljujući alatima i infrastrukturi za izvršavanje, testiranje i isporuku promjena u vrlo kratkom roku. Tu se Jenkins pojavio kao najperspektivniji kandidat.

Što je kontinuirana integracija s Jenkinsom?

Jenkins je unakrsna platforma, kontinuirana integracija i kontinuirana isporuka zasnovana na Javi koja povećava ukupnu produktivnost.

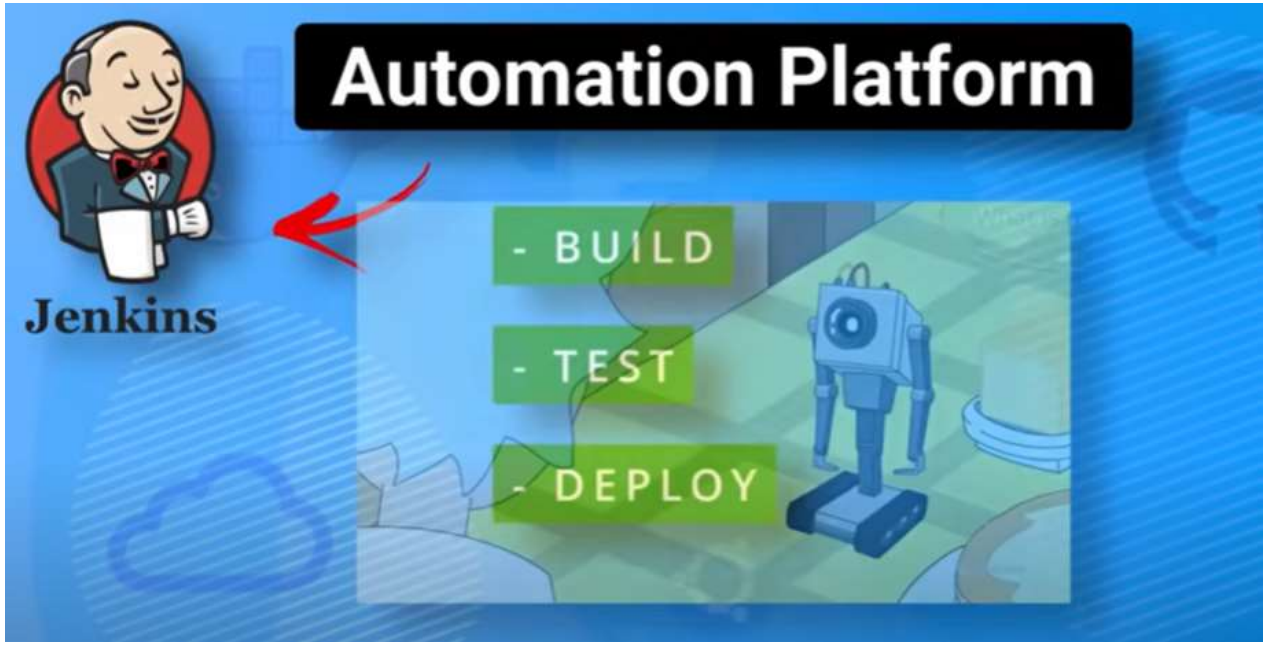
Jenkins može da se koristi za kontinuirano kreiranje i testiranje softverskih projekata, što olakšava programerima da integrišu promene u projekat i olakšaju korisnicima nabavku novog materijala.

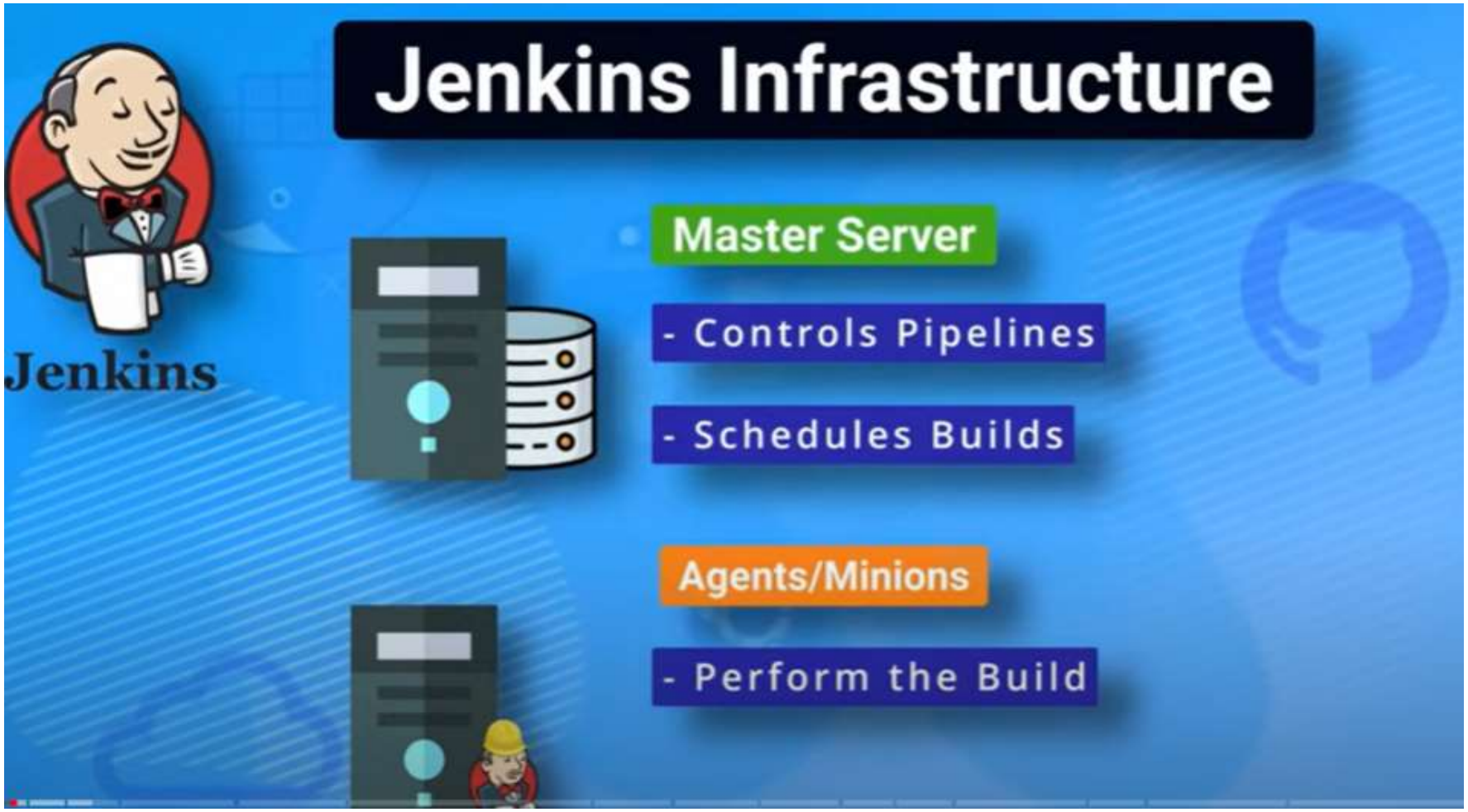
Takođe vam omogućava da kontinuirano isporučujete softver obezbeđujući moćne načine za definisanje cevovoda za izgradnju i integrisanje sa širokim spektrom tehnologija testiranja i implementacije.

Što je kontinuirana integracija s Jenkinsom?

Jenkins je server za kontinuiranu integraciju.

Jednostavnim rečima, kontinuirana integracija je praksa automatskog izvršavanja testova na nerazvijenom računaru svaki put kada neko gurne novi kôd u originalno skladište.




Jenkins Infrastructure
Jenkins
Master Server
- Controls Pipelines
- Schedules Builds
Agents/Minions
- Perform the Build

Git pošalje preko pipelin-a

Morate imati instaliranu Javu i podešen SSH


Agent Types
Cloud Agents
- Ephemeral/Dynamic Agents spun up on demand
aws

Značaj Jenkinsa

1. Dženkins može biti u potpunosti konfigurisan putem grafičkog interfejsa zasnovanog na vebu koji je prilagođen korisniku uz opsežne provere grešaka u letu i ugrađenu pomoć.

2. Dženkins se integriše sa skoro svakim SCM-om ili alatom za izradu koji postoji danas.

3. Većina Jenkins delova se može proširiti i modifikovati, a lako je napraviti novi Jenkins pribor. Ova funkcija vam omogućava da prilagodite Jenkinsa svojim potrebama.

4. Jenkins može distribuirati opterećenja za izgradnju / testiranje na više računara s različitim operativnim sistemima.

Sadašnjost i budućnost kontinuirane isporuke s Jenkinsom

Jenkinsova platforma otvorenog koda vodeća je u prostoru kontinuirane isporuke.

1. Iako je objavljen samo 2011. godine, postoji više od 85 000 aktivnih instalacija širom svijeta, od kojih se mnoge koriste kao središte za kontinuiranu isporuku i razvojnu metodologiju DevOps. Jenkins je u svakom slučaju budućnost kontinuirane isporuke.

2. Zajednica Jenkins razvila je oko 1.000 dodataka, omogućujući softveru integraciju s mnogim popularnim tehnologijama.

3. Aktivne Jenkinsove instalacije povećale su se za 160 posto u 2013. i za više od 300 posto u tri godine do kraja 2015. godine.

Sadašnjost i budućnost kontinuirane isporuke s Jenkinsom

4. Kontinuirana isporuka ne uključuje samo visokofrekventne iteracije za poboljšanje načina rada softvera, već također omogućuje provjere u stvarnom vremenu kako bi se izmerilo postižu li promene koda određene poslovne ciljeve. Uz Jenkinsa programeri će moći pružiti povratne informacije firmi. Ovo će biti jedna značajna promena u korporacijskoj kulturi.

5. U anketi od 721 razvojnog stručnjaka u San Franciscu otkriveno je da su Jenkins zainteresirani za sve vrste radnih mjesta, a programer je naveden kao najčešća uloga posla (71 posto), a sledi menadžer gradnje (41 posto) , arhitekta softvera (24 posto) i DevOps profesionalca (21 posto). (Izvor: Cloudbees.com)

Fokusiraću se na Dženkinsove arhitekture i Jenkinsov cevovod za izgradnju, i zajedno sa tim, pokazaću vam kako se stvara izgradnja u Jenkinsu.

- Jenkins se koristi za integriranje svih faza DevOps uz pomoć dodataka.
- Često korišteni Jenkins dodaci su Git, Amazon EC2, Maven 2 projekt, HTML izdavač itd.
- Jenkins ima preko 1000 dodataka i147.000 aktivnih instalacija, zajedno s preko milijun korisnika širom svijeta.
- Uz kontinuiranu integraciju svaku promjenu u izvornom kodujeizgrađena. Obavlja i druge funkcije,to ovisi o alatu koji se koristi za kontinuiranu integraciju.
- Nokia je s Nightly build prešla na kontinuiranu integraciju.
- Proces pre kontinuirane integracije imao je mnogo propusta. Kao rezultat toga, ne samo da je isporuka softvera bila spora, već kvalitet softvera nije bio na visini. Programerima je takođe bilo teško da pronađu i isprave bagove.
- Kontinuirana integracija sa Dženkinsom prevazišla je ove nedostatke kontinuiranim pokretanjem izgradnje i testiranjem za svaku promenu izvornog koda.

Jenkins Arhitektura

Ovaj jedinstveni Jenkinsov server nije bio dovoljan da ispuni određene zahteve .

Ponekad će vam biti potrebno nekoliko različitih sredina da biste testirali svoje konstrukcije. To ne može da uradi jedan od Jenkinsovih servera.

Ako se redovnije grade veći i teži projekti, jedan Jenkins server ne može jednostavno da se nosi sa celim teretom.

Da bi se rešile gore navedene potrebe predstavljena je Jenkinsova distribuirana arhitektura.

Jenkinsova distribuirana arhitektura

Jenkins koristi Master-Slave arhitekturu za upravljanje distribuiranim gradnjama. U ovoj arhitekturi, Master i Slave komuniciraju putem TCP / IP protokola.

Majstor Jenkins

Vaš glavni Jenkinsov poslužitelj je **Master**. Njegov posao je da obrađuje:

- Zakazivanje građevinskih poslova.
- Otpremajući građevine robovima za stvarno izvršenje.
- Nadgledajte robove (po mogućnosti ih povežite na mrežu i izvan nje, ako je potrebno).
- Snimanje i predstavljanje rezultata izrade.
- Glavna instanca Jenkinsa također može izravno izvršavati poslove gradnje.

Jenkins Slave

Slave je Java izvršna datoteka koja se izvodi na udaljenom mašini. Slede karakteristike Jenkins Slaves:

- Čuje zahteve instance **Jenkins Master**.
- Robovi mogu raditi na raznim operativnim sustavima.
- Posao Roba je raditi kako im se kaže, što uključuje izvršavanje građevinskih poslova koje je poslao Gospodar.
- Možete konfigurirati projekt da se uvijek izvodi na određenom Slave stroju ili određenoj vrsti Slave stroja ili jednostavno pustiti Jenkinsa da odabere sljedeći dostupni Slave.

Jenkins (pipeline)cevovod za izgradnju

Koristi se da bi se znalo koji zadatak Jenkins trenutno obavlja. Često nekoliko programera napravi nekoliko različitih promena odjednom, pa je korisno znati koja promena se testira ili koja promena sedi u redu ili koja konstrukcija je pokvarena. Tu na sliku ulazi cevovod.

Jenkinsov cevovod vam daje pregled gde se rade testovi. U građevinskom cevovodu, fabrička jedinica je podeljena na delove, kao što su **test jedinice**, **test prihvatanja**, **pakovanje, izveštavanje i faze podešavanja**.

Faze cevovoda mogu da se pokreću u nizu ili paralelno, a ako jedna faza bude uspešna, automatski prelazi na sledeću fazu (otuda i relevantnost imena „cevovod").

Otvorit ću novi posao u Jenkinsu, to je **Projekt slobodnog stila** . Međutim, dostupne su još 3 mogućnosti. Pogledajmo vrste građevinskih poslova dostupnih u Jenkinsu.

Projekt slobodnog stila:

Poslovi slobodne gradnje su građevinski poslovi opšte namene, koji pružaju maksimalnu fleksibilnost.Posao izrade slobodnog stila najfleksibilnija je i najkonfigurabilnija opcija i može se koristiti za bilo koju vrstu projekta. Postavljanje je relativno jednostavno, a mnoge opcije koje ovdje konfiguriramo pojavljuju se i u drugim poslovima izrade.

Višekonfiguracijski posao:

„Multikonfiguracijski projekt“ (koji se naziva i „matrični projekt“) omogućuje vam pokretanje istog posla izrade u različitim okruženjima. Koristi se za testiranje aplikacije u različitim okruženjima, s različitim bazama podataka ili čak na različitim strojevima za izradu.

Nadgledanje spoljašnjeg rada

Zadatak izrade ’Nadgledanje spoljašnjeg rada' omogućuje vam da nadgledate neinteraktivne procese, poput cron poslova.

Projekt Maven:

'Maven2 / 3 projekt' je posao izrade posebno prilagođen Maven projektima.

Jenkins razumije Maven pom datoteke i strukture projekata i može koristiti podatke prikupljene iz datoteke pom kako bi smanjio posao koji trebate učiniti za postavljanje vašeg projekta.

Kontinuirana isporuka:

Neprekidna isporuka je proces u kome se promene koda automatski grade, testiraju i pripremaju za puštanje.

Pričaćemo šta je:

1. Šta je kontinuirana isporuka?
2. Tipovi testiranja softvera
3. Razlika između kontinuirane integracije, isporuke i primene
4. Kakva je potreba za kontinuiranom isporukom?
5. Zgodna upotreba Dženkinsa i Tomata

Šta je kontinuirana isporuka?

To je proces gde pravite softver na takav način da ga možete staviti u proizvodnju u bilo kom trenutku.

1. Automatizovane građevinske skripte će otkriti promene u upravljanju izvornim kodom (SCM), kao što je Git.

2. Kada se otkrije promena, izvorni kôd će biti raspoređen na namenski građevinski server kako bi se osiguralo da izgradnja propadne i da svi test časovi i testovi integracije dobro funkcionišu.

3. Zatim je build aplikacija instalirana na probnim serverima (preprodukcionim serverima) za Test prihvatanja korisnika (UAT).

4. Konačno, aplikacija se ručno nalazi na proizvodnim serverima radi izdavanja.

Vrste testiranja softvera

Uopšteno govoreći, postoje dve vrste testiranja:

1. **Blackbox testiranje**:

To je probna tehnika koja zanemaruje unutrašnji mehanizam sistema i fokusira se na izlaz koji se generiše u odnosu na bilo kakav unos i izvršavanje sistema. Naziva se i funkcionalno testiranje. U osnovi se koristi za proveru valjanosti softvera.

2. **Whitebox testiranje**:

Je probna tehnika koja uzima u obzir unutrašnji mehanizam sistema. Naziva se i testiranje građevinskih i staklenih kutija. U osnovi se koristi za proveru softvera.

Testiranje bele kutije:

Postoje dve vrste testova koji spadaju u ovu kategoriju.

1. Jedinstveno testiranje: Ovo je testiranje pojedinačne jedinice ili grupe srodnih jedinica. Programer to često radi da bi testirao da jedinica koju je primenio daje očekivani izlaz u odnosu na podrazumevani unos.

2. Testiranje integracije: Ovo je vrsta testa u kojoj se grupa komponenti kombinuje da bi dala rezultate. Takođe, interakcija između softvera i hardvera se testira ako softver i hardverske komponente imaju bilo kakve veze. Može da padne i u test bele kutije i u crnu kutiju.

Blackbox testiranje:

Postoji nekoliko testova koji spadaju u ovu kategoriju.

1. Testiranje funkcionalnosti/prihvatanja: Obezbeđuje da navedena funkcija potrebna u sistemskim zahtevima funkcioniše. Ovo se radi da bi se osiguralo da isporučeni proizvod ispunjava zahteve i funkcioniše onako kako je kupac očekivao.
2. Testiranje sistema: Obezbeđuje da postavljanjem softvera u različita okruženja (npr. operativni sistemi) i dalje radi.
3. Test stresa: Procenjuje kako se sistem ponaša u nepovoljnim uslovima.
4. Beta testiranje: Ovo rade krajnji korisnici, tim van razvojnog programa ili javno najavljuju punu pred verziju proizvoda koji je poznat kao betaverzija. Cilj beta testiranja je pokrivanje neočekivanih grešaka.

Razlike između kontinuirane integracije, isporuke i primene:

Vizuelni sadržaj dopire do mozga pojedinca brže i razumljivije od tekstualnih informacija. Zato ću početi sa dijagramom koji jasno objašnjava razliku:

U kontinuiranoj integraciji, svaki kôd se gradi i testira, ali nije u mogućnosti da bude objavljen. Mislim, aplikacija za izradu se ne posavetuje automatski na probne servere da bi se proverila valjanost pomoću različitih tipova Blackbox testiranja kao što je - Testiranje prihvatanja korisnika (UAT).

U kontinuiranoj isporuci, aplikacija se neprekidno nalazi na probnim serverima za UAT. Ili možete da kažete da je aplikacija spremna za puštanje u bilo kom trenutku. Dakle, očigledno je kontinuirana integracija od suštinskog značaja za kontinuiranu isporuku.

Neprekidno podešavanje je sledeći korak pored kontinuirane isporuke, gde ne samo da kreirate paket koji se može primeniti, već ga zapravo automatizujete.

Kontinuirana integracija	Kontinuirana dostava	Kontinuirano postavljanje
Automatizirana izrada za svaku, predaju	Automatizirana izrada i UAT za svako, predavanje	Automatizirana izrada, UAT i puštanje u proizvodnju za svaki, obvezuju
Bez obzira na kontinuiranu isporuku i neprekidnu instalaciju	To je sljedeći korak nakon kontinuirane integracije	korak je dalje Kontinuirana isporuka
Na kraju, aplikacija nije u stanju biti puštena u proizvodnju	Na kraju, aplikacija je u stanju da bude puštena u proizvodnju.	Aplikacija se kontinuirano primjenjuje
Uključuje testiranje Whitebox-a	Uključuje Blackbox i Whitebox testiranje	Uključuje celokupan postupak potreban za postavljanje aplikacije

Zašto nam je potrebna kontinuirana isporuka

Jednostavno rečeno, kontinuirana integracija je deo kontinuirane isporuke i kontinuirane implementacije. Neprekidno podešavanje je kao neprekidna isporuka, osim što se izdanja dešavaju automatski.

Razumemo to na primeru.

Zamislite da 80 programera radi na velikom projektu. Oni koriste cevovode za kontinuiranu integraciju da bi olakšali automatizovane građe. Znamo da izgradnja podrazumeva i jedinstveno testiranje. Jednog dana su odlučili da primene najnoviju verziju koja je prošla jedinstvene testove u probnom okruženju.

To mora da je vremenski zahtevan, ali kontrolisan pristup rasporedu koji sprovode njihovi stručnjaci za zaštitu životne sredine. Međutim, čini se da sistem nije funkcionisao.

Šta bi mogao da bude očigledan uzrok neuspeha?

Pa, prvi razlog zašto će većina ljudi pomisliti je da postoji neki problem sa konfiguracijom. Kao i veći broj ljudi, iako su i oni tako mislili. Proveli su dosta vremena pokušavajući da shvate šta nije u redu sa konfiguracijom životne sredine, ali nisu uspeli da pronađu problem.

Jedan perceptualni programer je primenio pametan pristup:

Onda je jedan od starijih programera isprobao aplikaciju na njegovoj razvojnoj mašini. Ni tamo nije uspelo.

Vratio se kroz ranije i ranije verzije dok nije ustanovio da je sistem prestao da radi tri nedelje ranije. Mala, nejasna greška sprečila je pravilno pokretanje sistema. Iako je projekat imao dobru pokrivenost testom jedinice. Uprkos tome, 80 programera, koji su obično sprovodili testove umesto same aplikacije, tri nedelje nisu videli nikakav problem.

Problem sa Sajom:

Bez izvođenja testova prihvatanja u proizvodnom okruženju, oni ne znaju ništa o tome da li aplikacija zadovoljava specifikacije korisnika, odnosno da li može da se implementiše i preživi u stvarnom svetu. Ako žele pravovremenu povratnu informaciju o ovim temama, potrebno je da prošire opseg svog procesa kontinuirane integracije.

Dozvolite mi da rezimiram naučene lekcije gledajući gore navedene probleme:

1. Jedinstveni testovi testiraju samo razvojnu perspektivu rešenja problema. Oni imaju samo ograničenu sposobnost da dokažu da aplikacija radi ono što treba iz perspektive korisnika. Nije dovoljno prepoznati stvarne funkcionalne probleme.

2. Primena aplikacije u probnom okruženju je složena, ručno intenzivna procedura koja je bila prilično sklona greškama. To je značilo da je svaki pokušaj implementacije bio novi eksperiment - ručna procedura podložna greškama.

Rešenje - Kontinuirani cevovod za isporuku (test automatizovanog prihvatanja):

Prešli su na kontinuiranu integraciju (kontinuiranu isporuku) do sledećeg koraka i uveli nekoliko jednostavnih, automatizovanih testova prihvatanja koji su dokazali da je aplikacija pokrenuta i da može da obavlja svoju najosnovnije funkcije. Većina testova obavljenih tokom faze provere prihvatljivosti su funkcionalni testovi prihvatanja.

Oni su u osnovi izgradili cevovod za neprekidnu isporuku, kako bi osigurali da aplikacija bude neprimetno raspoređena u proizvodnom okruženju, obezbeđujući da aplikacija dobro funkcioniše kada je raspoređena na probnom serveru koji je replika proizvodnog servera.

U nastavku pokažimo kako da napravite cevovod neprekidne isporuke koristeći Dženkinsa.

Cevovod za kontinuiranu isporuku pomoću Dženkinsa:

Ovde ću iskoristiti Jenkinsa da napravim cevovod za neprekidnu isporuku, koji će uključivati sledeće zadatke:

Koraci uključeni u demonstraciju:

1. **Preuzimanje koda od GitHub-a**
2. **Sastavljanje izvornog koda**
3. **Jedinstveno testiranje i generacija JUnit test izveštaja**
4. **Pakovanje aplikacije u WAR fajl i postavljanje na Tomcat server**

Preduslovi:

1. CentOS 7 mašina
2. Jenkins 2.121.1
3. Doker
4. Tomcat 7

Korak - 1 Sastavljanje izvornog koda

Počnimo tako što ćemo prvo raditi projekat Freestyle u Dženkinsu.

Dajte ime svom projektu i izaberite Freestyle Projekat:

Kada se pomerite nadole, naći ćete opciju da dodate skladište izvornog koda, izaberete git i dodate URL adresu skladišta, pom.xml postoji kazna u tom skladištu koju ćemo koristiti za izradu našeg projekta. Uzmite u obzir dole navedeni snimak ekrana:

Hajde da dodamo okidač za izgradnju. Izaberite SCM opciju za istraživanje, mi ćemo u osnovi konfigurisati Dženkinsa da pregleda GitHub skladište nakon svakih 5 minuta zbog promena koda.

Pre nego što nastavim, dozvoliću vam mali uvod u **Maven** građevinski ciklus.

Svaki od životnih ciklusa izrade definisan je različitom listom faza izrade, gde po fazi izrade predstavlja fazu u životnom ciklusu.

Sledi lista faza izrade:

1. potvrditi - proveriti da li je projekat ispravan i da li su dostupne sve potrebne informacije

2. sastavljanje - sastavljanje izvornog koda projekta

3. test - testirajte sastavljeni izvorni kôd koristeći pogodan okvir za jedinstveno testiranje. Ovi testovi ne bi trebalo da zahtevaju da šifra bude upakovana ili primenjena

4. paket - uzmite sastavljeni kôd i spakujte ga u njegov distribuirani format, kao što je JAR.

5. provera - proverite rezultate testova integracije da biste se uverili da su kriterijumi kvaliteta ispunjeni

6. instalirati - instalirati paket u lokalnom depou, za lokalnu upotrebu kao zavisnost u drugim projektima

7. implementacija - izvedena u građevinskom okruženju, kopira završni paket u udaljeno skladište za deljenje sa drugim projektantima i projektima

Oni mogu da pokreću dole navedenu komandu da bi sastavili izvorni kôd, testirali jedinicu, pa čak i da spakuju aplikaciju u ratni fajl:

```
mvn čisto pakovanje
```

Takođe možete da rastaovite svoje građevinske radove u brojne proizvodne korake. Ovo olakšava organizaciju izrade u čistim, odvojenim fazama.

Zbog toga ćemo početi sastavljanjem izvornog koda. Na kartici "Izrada" kliknite na ciljeve maven poziva najvišeg nivoa i otkucajte dole navedenu komandu:

```
Okupi
```

Uzmite u obzir dole navedeni snimak ekrana:

Ovo će izvući izvorni kod iz depoa GitHub-a i takođe će ga sastaviti (Maven Compile Phase).

Kliknite na dugme Sačuvaj i pokrenite projekat.

Sada kliknite na izlaz konzole da biste videli rezultat.

Korak - 2 Jedinstveno testiranje:

Sada ćemo napraviti još jedan Freestyle projekat za jedinstveno testiranje.

Dodajte istu URL adresu skladišta na karticu za upravljanje izvornim kodom, kao što smo to radili na prethodnom poslu.

Sada, na kartici 'Buid Trigger', kliknite na 'build after other projects are built'. Tamo ukucajte ime prethodnog projekta u kome sastavljamo izvorni kôd i možete odabrati neku od sledećih opcija:

1. Samo ako je konstrukcija stabilna.
2. Okidač čak i ako je konstrukcija nestabilna
3. Okidač čak i ako izrada ne uspe

Mislim da su gore navedene opcije prilično razumljive, pa odaberite bilo koju. Uzmite u obzir dole navedeni snimak ekrana:

Na kartici Izrada kliknite na ciljeve najvišeg nivoa poziva i upotrebite dole navedenu komandu:

```
Test
```

Dženkins takođe odlično radi tako što vam pomaže da prikažete rezultate testova i trendove rezultata testa.

De facto standard za izveštavanje testova u svetu Tehnologije Java je XML format koji koristi JUnit. Ovaj format koriste i mnoge druge Java alatke za testiranje, kao što su TestNG, Spock i Easyb. Dženkins razume ovaj format, pa ako vaša zanatska radnja daje rezultate JUnit XML testa, Dženkins vremenom može da generiše lepe grafičke izveštaje i statistiku rezultata testova, a takođe vam omogućava da tokom vremena pregledate detalje svih propusta na testu. Dženkins takođe prati koliko dugo traju vaši testovi, kako globalno tako i po testu ovo može da vam bude od koristi ako je potrebno da pronađete probleme sa performansama.

Sledeća stvar koju treba da uradimo je da navuиemo Jenkinsa da prati testove jedinice.

Idite u odeljak "Radnje posle izrade" i proverite izbor u polju za potvrdu "Objavi izveštaj o rezultatima JUnit testa". Kada Maven uradi testove jedinice u projektu, on automatski generiše XML test izveštaje u direktorijumu koji se zove surefire-reports. Zato unesite '**/target/surefire-reports/*. Xml' u polje 'TEST REPORT XML'. Dve zvezde na početku putovanja ('**') su najbolja praksa da konfiguracija bude malo robusnija: one omogućavaju Dženkinsu da pronađe ciljni direktorijum bez obzira kako smo konfigurisali Dženkinsa da proveri izvorni kod.

```
** / target / surefire-reports / *. xml
```

Sačuvajte je ponovo i kliknite na dugme "Napravi odmah".

Sada je izvješće JUnit napisano na / var / lib / jenkins / workspace / test / gameoflife-core / target / surefire-reports / TEST-behaviour.

Na Dženkinsoj kontrolnoj tabli takođe možete da primetite rezultate testa:

Korak - 3 Napravite WAR fajl i postavite ga na Tomcat server:

Sledeći korak je da upakujemo našu aplikaciju u WAR fajl i smestimo je na Tomcat server za test prihvatanja korisnika.

Kreirajte još jedan projekat slobodnim stilom i dodajte URL adresu skladišta izvornog koda.

Zatim, na kartici okidač za izgradnju izaberite konstrukciju kada se grade drugi projekti, razmislite o snimku ekrana ispod:

Što Je Znanost O Podacima?

U osnovi, nakon probnog zadatka, faza uvođenja će početi automatski.

Na kartici "Izrada" izaberite skriptu ljuske. Otkucajte dole navedenu komandu da biste paket spakovali u WAR datoteku:

mvn paket

Sledeći korak je da primenite ovaj WAR fajl na Tomcatserver. Na kartici "Postizgradne radnje" izaberite raspoređivanje rata/ušiju u kontejneru. Ovde dajte putanju do ratne datoteke i navedite putanju konteksta. Uzmite u obzir dole navedeni snimak ekrana:

Izaberite akreditive za Tomcat i obratite pažnju na gorenavedeni snimak ekrana. Takođe, potrebno je da navedete URL adresu vašeg Tomcat servera.

Da biste dodali akreditive u Dženkinsu, kliknite na opciju akreditiva u Dženkins kontrolnoj tabli.

Kliknite na dugme "Sistem" i izaberite globalne akreditive.

Zatim ćete pronaći opciju za dodavanje akreditiva. Kliknite na njega i dodajte akreditive.

Dodajte akreditive za Tomcat, razmislite o snimku ekrana ispod.

Kliknite na dugme U redu.

Sada dodajte tomkat akreditive koje ste umetnuli u prethodnom koraku u konfiguraciju projekta.

Kliknite na dugme Sačuvaj, a zatim izaberite stavku Napravi odmah.

Idite na vašu URL adresu tomkata, sa kontekstualnom putanjom, u mom slučaju http://localhost:8081. Sada dodajte kontekstualna putanja na kraju, razmislite o snimku ekrana ispod:

```
Link - http://localhost://8081/gof
```

Nadam se da razumete značenje kontekstualnog puta.

Sada kreirajte prikaz cevovoda,

Kliknite na ikonu plus da biste kreirali novi prikaz.

Konfigurišite cevovoda na način na koji želite,

Nisam promenio ništa osim izbora početnog posla. Dakle, moj gasovod će početi od prevođenja. Na osnovu načina na koji sam konfigurisali druge poslove, testiranje i implementacija će se desiti nakon sastavljanja.

Na kraju sve strane, možete testirati cevovod tako što ćete kliknuti na dugme RUN. Nakon svakih pet minuta, ako dođe do promene izvornog koda, ceo gasovod će biti izvršen.

Stoga smo u mogućnosti da kontinuirano plasiranim aplikacijom na probni server za prihvatanje korisnika (UAT).

Kako konfigurisati obaveštenje putem e-pošte u Jenkinsu

Jenkins je svakako jedan od najpopularnijih alata u. On može brže da automatizuje kreiranje i testiranje koda i zbog toga softverske kompanije mogu da ubrzaju svoje razvojne procese. Jenkins vam pruža uslugu obaveštavanja putem e-pošte putem koje možete da prijavite timu status izrade i rezultate testiranja.

Sledeća uputstva će biti:

1. Zašto su nam potrebna obaveštenja putem e-pošte u Dženkinsu?
2. Problem pre Jenkinsa
3. Rešenje sa Jenkinsom
4. Kako koristiti uslugu e-pošte u Jenkinsu?

Zašto nam treba obaveštenje putem e-pošte u Jenkinsu?

Problem sa Sajom:

1. Pretpostavimo da je objavljivanje prijave zakazano za ponoć. Sada postoji problem sa aplikacijom na testu ili proizvodnom serveru.

Takođe, može se desiti da aplikacija bude puštena i da se sruši posle nekoliko sati.

Ako aplikacija, recimo, Netfliks ne radi ni nekoliko minuta, to može dovesti do toga da milioni dolara budu izgubljeni. Takođe zbog takvih grešaka, rok za projekat mogao bi da bude produžen.

Rešenje

1. Ovaj problem je rešen automatizacijom alatke koja se zove Jenkins.

   Jenkins ima uslugu obaveštavanja putem e-pošte da rešava takve situacije.

2. Ukoliko izgradnja ne bude uspešna, tim projektanta je obavešten o statusu izgradnje. Ovo se može uraditi pomoću dodatne komponente e-pošte u Jenkinsu. Dodatna oprema je primarno sredstvo za poboljšanje funkcionalnosti i Jenkins je okruženje koje odgovara potrebama organizacije ili korisnika.

3. Koristite dodatnu komponentu e-pošte da biste konfigurisali detalje e-pošte osobe o kojoj se brine da bude obaveštena u slučaju neuspeha u izradi.

4. Kada programer bude obavešten o grešci, on je popravlja i ponovo prenosi kôd na GitHub. Posle toga Jenkins ponovo povlači šifru iz GitHub-a i priprema novu građu.

5. Slično tome, Jenkins može da reši problem pada aplikacije nakon iznošenja, tako što će obavestiti tim o kome je reč, slanjem e-maila.

Da vidimo kako da pošaljemo obaveštenja Jenkinsu.

U osnovi postoje dva načina za konfigurisanje obaveštenja putem e-pošte u Jenkinsu.

1. Korišćenje dodatka za proširenje e-pošte - To uključuje omogućava konfigurisanje svakog aspekta obaveštenja putem e-pošte. Možete da prilagodite stvari kao što su slanje e-poruke, ko je prima i šta piše u e-poruci.

2. Korišćenje podrazumevanog obaveštenja putem e-pošte - Ovaj standard dolazi sa Jenkinsom. Ima podrazumevanu poruku koja se sastoji od broja i statusa konstrukcije.

Programski dodatak proširenja e-pošte

Korak 1: Prijavite se na Dženkins matičnu stranicu

Idite na Dženkins matičnu stranicu koristeći URL localhost: 8080. Podrazumevani broj porta je 8080. U mom sluиaju, to je 9191. Prijavite se sa korisničkim imenom i lozinkom.

2. korak: Instaliranje dodatne komponente proširenja e-pošte

Nakon toga, na Dženkins matičnoj stranici kliknite na "Manage Jenkins->" da biste upravljali dodacima. Na dostupnoj kartici potražite dodatak za proširenje e-pošte. Ako je tamo, instaliraj ga. Ako ga nema, potražite ga na instaliranoj kartici.

Programski dodatak proširenja e-pošte

3. korak: Konfigurisanje sistema

Sada idite na "Upravljanje konfiguracijom > sistema Dženkins-a" . Pomerite se ovde nadole do odeljka sa obaveštenjem putem e-pošte. Ako koristite Gmail, otkucajte smtp.gmail.com za SMTP server. Kliknite na dugme Više opcija i izaberite stavku Koristi SMTP potvrdu identiteta. Unesite svoje Gmail korisničko ime i lozinku. Izaberite koristi SSL i unesite broj porta kao 465 . Kliknite na dugme Primeni, a zatim kliknite na dugme Sačuvaj.

Programski dodatak proširenja e-pošte

4. korak: Otvorite posao Jenkins cevovod

Sada idi na Jenkinsov dom i otvori novi posao. Pozovite posao sa bilo kojim imenom koje želite i izaberite cevovod. Kliknite na dugme U redu.

Sada u odeljku cevovoda otkucajte sledeći kôd.

cjevovod {agent bilo koji faze {stage ('Ok') {steps {echo 'Ok'}}} post {always {emailext body: 'A Test Email', primateljProviders: [[$ class: 'DevelopersRecipientProvider'], [$ class : 'RequesterRecipientProvider']], tema: 'Test'}}}

Ovaj cevovod radi u bilo kom Jenkinsovom agentu. Ima fazu uzorkovanja. U koraku objave možete da pokrenete bilo koju skriptu koju želite. Sadrži pošiljaoca pošte. Sačuvajte je i pokrenite tako što ćete u meniju posla izabrati stavku "Napravi odmah". Izgradnja će se pojaviti u scenskom pogledu.

Programski dodatak proširenja e-pošte

5. korak: Pogledajte izlaz konzole

Kliknite na broj verzije "# 1" i u meniju "Izrada" izaberite stavku "Izlaz konzole". Izlaz će biti ovakav:

Korak 6: Proverite e-poštu.

Nakon toga, idite u svoj Gmail inbox i trebalo bi da možete da vidite ovakav mejl.

Podrazumevano obaveštenje e-pošte

Korak 1: Prijavite se na Dženkins matičnu stranicu

Idi na Dženkinsove matične stranice.

2. korak: Konfigurisanje sistema

Kliknite na dugme Upravljaj konfiguracijom > sistema Dženkins-A . Pomerite se ovde nadole do odeljka Obaveštenja putem e-pošte. Sada unesite detalje kao sledeću sliku

Kada podesite konfiguracije pošte, možete da proverite da li dobro radi ili ne tako što ćete poslati test e-poštu .

Podrazumevano obaveštenje e-pošte

3. korak: Dodajte radnju nakon izgradnje svom projektu

Da biste dozvolili projektima da šalju e-poruke, potrebno je da dodate radnju nakon izrade i izaberete ' Obaveštenje putem e-pošte sa padajuće liste. Ovo će vam obezbediti dati daunloud interfejs, gde možete da dodate listu e-adresa kojima je neophodno da pošaljete e-poruku.

4. korak: Kreiranje projekta i provera e-pošte

Pokušajte da započnete projekat kojem ste dodali e-poštu sada. Ako izgradnja propadne, dobićete e-poruku u vezi sa neuspehom izgradnje.

Znači ovako se postavljaju obaveštenja putem e-pošte u Jenkinsu.