МІНІСТЭРСТВА СУВЯЗІ І ІНФАРМАТЫЗАЦЫІ РЭСПУБЛІКІ БЕЛАРУСЬ
Установа адукацыі
«БЕЛАРУСКАЯ ДЗЯРЖАЎНАЯ АКАДЭМІЯ СУВЯЗІ»
ФАКУЛЬТЭТ ЭЛЕКТРАСУВЯЗІ
КАФЕДРА ПРАГРАМНАГА ЗАБЕСПЯЧЭННЯ

МЕТАДАЛОГІЯ DEVOPS. ІНСТРУМЕНТЫ БЕСПЕРАПЫННАЙ ІНТЭГРАЦЫІ І ДАСТАЎКІ

Тлумачальная запіска
да курсавай работы
па дысцыпліне
«ТЭХНАЛОГІЯ РАСПРАЦОЎКІ І СУПРАВАДЖЭННЯ ПРАГРАМНАГА ЗАБЕСПЯЧЭННЯ»

Выканаў студэнт групы СП 741 Буцько Антон Уладзіміравіч

Кіраўнік Трухановіч Таццяна Леанідаўна

Мінск 2019

АНАТАЦЫЯ

У дадзенай курсавой рабоце расказваецца пра метадалогію Devops, бесперапынную інтэграцыю і дастаўку.

У першым раздзеле разглядаліся асноўныя прынцыпы метадалогіі Devops, перавагі ўкаранення Devops-практык. У другім раздзеле разглядаліся асноўныя характарыстыкі бесперапыннай інтэграцыі і дастаўкі, а таксама інструментаў для іх рэалізацыі. У трэцім раздзеле у якасці прыкладу быў пабудаваны канвеер бесперапыннай інтэграцыі і дастаўкі для Web-праграмы генерацыі фраз.

Курсавая работа змяшчае 25 старонак, 14 крыніц літаратуры, 1 табліцу, 10 малюнкаў, 5 лістынгаў.

ЗМЕСТ

ПРАДМОВА

З паскарэннем тэмпаў распрацоўкі праграмнага забеспячэння і неабходнасці падстройвацца да дынамічных змен развіваліся новыя метадолігіі распрацоўкі: вадаспад (Waterfall), метадалогія гібкай распрацоўкі (Agile), спіральны метад (Spiral), мадэль хуткай распрацоўкі (Rapid Application Development), devops і іншыя.

Devops – сучасная метадалогія распрацоўкі праграмнага забеспячэння, якая дапамагае павялічыць прадукцыйнасць працы на ўзроўні арганізацыі.

Метадалогія Devops шырока распаўсюджана сярод IT-кампаній замежный краін, і ўжо паспела даказаць паспяховаць укаранення пэўных практык devops у працэсы арганізацыі.

У той жа час у краінах на постсавецкіх тэрыторыях дадзеная метадалогія вывучана слаба, што запавольвае яе распаўсюджванне, а таксама спараджае разнастайныя непаразуменні. Нягледзячы на гэта дадзеная метадалогія таксама пачынае ўкараняцца сярод некаторых айчынных кампаній (Itransition, ISsoft, BeCloud і іншыя).

Мэтай курсавой работы з’яўляецца вывучэнне ўласцівасцей метадалогіі devops, тэхналогій бесперапынная інтэграцыя (CI) і бесперапынная дастаўка (CD), іх перавагі, спосабы ўзаемадзеяння і рэалізацыі.

1 МЕТАДАЛОГІЯ DEVOPS

1.1 Паняцце Devops

Паняцце Devops не мае агульнапрынятага азначэння. Азначэнне Devops змяняецца ў залежнасці ад часу, метадаў укаранення, пастаўленых мэт унутры кампаніі. У табліцы 1.1 прыведзены некаторыя азначэнні, каторыя сустракаюцца ў літаратуры.

Табліца 1.1 – Варыянты азначэнняў Devops

Азначэнне Devops	**Сэнс азначэння**
Devops – гэта набор практык, які прызначыны для памяншэння часу паміж прыняццямі змен ў сістэме і ўнясення іх ў працэс вытворчасці з гарантыяй высокай якасці[5].	Арыентацыя на дасягненне мэты (хуткая дастаўка якаснага праграмнага забеспячэння)
Devops – гэта культурны рух, які змяняе адносіны людзей да працы і да яе вынікаў[1].	Арыентацыя на супрацоўніцтва
Devops – камбінацыя культуры, практык і інструментаў, каторая павялічвае здольнасць арганізацыі пастаўляць праграмы і сервісы з высокай хуткасцю: развіццё і паляпшэнне прадукцыі ў больш хуткім тэмпе, чым арганізацыі, якія карыстаюцца традыцыйным спосабам распрацоўкі праграмнага забеспячэння і кіравання працэсамі інфраструктуры[2].	Арыентацыя на супрацоўніцтва і дасягненне мэты (паскарэнне тэмпаў развіцця прадукту)
Devops – набор практык, накіраваны на актыўнае ўзаемадзеянне спецыялістаў па распрацоўцы і спецыялістаў па інфармацыйна-тэхналагічнаму абслугоўванню і ўзаемную інтэграцыю іх працоўных працэсаў адно ў другое[3].	Арыентацыя на супрацоўніцтва

З табліцы 1.1 можам заключыць, што паняцце Devops уключае ў сябе як культурны, так і тэхнічны складнікі. Пры гэтым набор інструментаў, такіх як Chef альбо Docker, з'яўляецца складнікам Devops дзякуючы спосабам іх прымянення, а не ў сілу характарыстык саміх інструментаў.[1] Такім чынам неабходна адзначыць, што ў першую чаргу Devops разглядаецца як змяненне культуры арганізацыі, а іменна адносін паміж камандамі (у прыватнасці каманды распрацоўшчыкаў і каманды тэхнічнага абслугоўвання).

1.2 Мадэль CALMS

Нягледзячы на разнастайнае тлумачэнне тэрміна, Devops мае пэўныя агульнапрызнаныя элементы. Гэтыя элементы складаюць мадэль CALMS[4] (малюнак 1.1):

– Culture (культура),

– Automation (аўтаматызацыя),

– Lean (ашчаднасць),

– Measurement (мера),

– Sharing (абмен).

Культура	Аўтаматызацыя	Ашчаднасць	Мера	Абмен
Пачынаць з людзей	Укараняць аўтаматызацыю працэсаў	Памяншаць лішнія дзеянні	Усяму вызначаць меру	Дзяліцца ведамі

Малюнак 1.1 – Мадэль CALMS

1.2.1 Культура.

Джэніфэр Дэвіс і Кэтрын Дэніэлс у сваёй кнізе[1] падкрэсліваюць, што не трэба ігнараваць нормы і каштоўнасці культур і міжасобасных узаемадзеянняў, звяртаючы ўвагу толькі на карыстанне інструментамі.

Яны сцвярджаюць, што без развязвання міжасобасных і міжгрупавых канфліктаў, якія ўзнікаюць у арганізацыі, немагчыма пабудаваць трывалыя адносіны, якія і прыводзяць да фарміравання Devops-асяроддзя.

Джэніфэр Дэвіс і Кэтрын Дэніэлс падзяляюць культуры на 2 пункты:

– Супрацоўніцтва

– Блізкасць

Супрацоўніцтва – гэта працэс дасягнення пастаўленай мэты пры дапамозе ўзаемадзеяння паміж некалькімі людзьмі. Галоўным прынцыпам руху Devops стаў кааперацыя паміж камандамі па распрацоўцы і эксплуатацыі праграмнага забеспячэння. Пры гэтым перад тым, як арганізаваць паспяховае ўзаемадзеянне паміж камандамі, каторыя маюць розныя мэты, неабходна наладзіць групавую работу ў адной камандзе. Так, калі ў камандах не ўпарадкавана работа на індывідуальным і групавым узроўнях, немагчыма дасягнуць паспяховага ўзаемадзеяння паміж камандамі.

Апроч развіцця і падтрымкі адносін супрацоўніцтва паміж асобнымі людзьмі, групамі і аддзеламі ўнутры арганізацыі, важны трывалыя ўзаемаадносіны ў сферы дзеянасці.

Блізкасць – гэта працэс фарміравання ўзаемаадносін паміж камандамі, які садзейнічае свабодзе выбару розных мэт альбо паказальнікаў пры захаванні агульных мэт арганізацыі. Пры гэтым аблягчаецца развіццё эмпаціі (здольнсць адчуваць пачуцці іншых людзей) і навучання ў розных групах. Блізкасць можа праяўляцца на ўзроўні ўзаемаадносін паміж арганізацыямі, што дазваляе ім дзяліцца досведам і вучыць адно аднаго. У выніку ствараецца калектыўная база культурных і тэхнічных ведаў.

1.2.2 Аўтаматызацыя.

Для таго, каб пабудаваць новы тып культуры ў арганізацыі, неабходна змяняць працэсы. Згодна з Гамбэлам і Малескім[6], гэта становіцца магчымым пры дапамозе Devops інструментаў і інфраструктуры як код. Мэтай практыкі аўтаматызацыі ў Devops ёсць дасягнення памяншэння часу выпуску і паскарэння зваротнай сувязі на змяненні. Гэта азначае выкарыстоўванне працэсу разгортвання, які ахоплівае ўсе змяненні, якія могуць быць унесены любой з каманд. Працэс разгортвання ўключае ў сябе працэсы распрацоўкі, зборкі, тэсціравання і разгортвання

сістэмы.

Аўтаматызацыя працэсаў дазваляе:

– паскорыць распрацоўку праграмнага забеспячэння;

– паменшыць імавернасць чалавечай памылкі;

– паскорыць атрыманне зваротнай сувязі пры ўнясенні змяненняў на этапах распрацоўкі праграмнага забеспячэння.

1.2.3 Ашчаднасць.

Ашчаднае мысленне з'яўляецца адной з ключавых частак Devops.

Марыя і Том Попендык[7] сфармулявалі шэраг прынцыпаў для выканання ашчаднага мыслення ў распрацоўцы праграмнага забеспячэння.

Першы прынцып – памяншэнне адходаў. Адходы – гэта ўсё, што не дабаўляе цэннасці канчатковаму прадукту. А цэннасць прадукту заўсёды разглядаецца з пункту гледжання кліента. Адходамі можа быць лішняя дакументацыя, дадатковыя праграмныя магчымасці, каторыя не былі яўна запатрабаваныя кліентам, перадача распрацоўкі ад адной групы іншай. Галоўная мэта – дастаўка акурат таго, што хацелі спажыўцы прадукту. Перавышэнне часу распрацоўкі таксама разглядаецца ў якасці адходаў.

Другі і трэці прынцыпы – гэта узмацненне навучання і прыняцце рашэнняў як мага пазней. У кантэксце распрацоўкі праграмнага забеспячэння гэта значыць, што ітэрацыйныя цыклы распрацоўкі і прыняцце канчатковых рашэнняў развіцця прадукту адбываецца толькі тады, калі гэта неабходна. Навучанне ўзмацняецца пры дапамозе пастаянных ацэнак і паляпшэнняў працэсаў. Прыняцце рашэнняў як мага пазней становіцца эфектыўным у сітуацыях, звязаных з павелічэннем узроўню нявызначанасці. Ва ўмовах дынамічнага развіцця рынку, лепшымі рашэннямі з'яўляюцца тыя рашэнні, якія былі зроблены на падставе фактаў, а не здагадак. У межах праграмнага забеспячэння гэта азначае, што сістэмы павінны ўключаць магчымасць унясення змяненняў.

Чацвёрты прынцып – найхутчэйшая дастаўка. Хуткая дастаўка дазваляе паскорыць цыкл зваротнай сувязі, што дазваляе своечасова вызначыць перавагі і недахопы ў новых версіях прадукту. Такім чынам хуткая дастаўка становіцца адным з галоўных кампанентаў метадалогіі Devops.

Пяты і шосты прынцыпы – упаўнаважванне каманд і пабудова цэласнасці. Тэхнічныя рашэнні павіны прынімацца людзьмі, якія будуць несці адказнасць за выкананне планаў, так як пры рашэннях адкладзеных да апошняй хвіліны, не існуе часу для арганізацыі дзейнасці працаўнікамі вышэйшай уладай. Цэласнасць праграмнайа забеспячэння стварае надзейную архітэктуру, высокую практычнасць, рамонтапрыдатнасць, адаптыўнасць і пашыральнасць[7].

Апошні прынцып разглядае цэласнасць усёй сістэмы працы. Цэласнасць у складаных сістэмах патрабуе глыбокага даследвання ў шматлікіх сферах. З пункту гледжання Devops гэта азначае, што каманды распрацоўшчыкаў і тэхнічнай эксплуатацыі павінны узаемадзейнічаць на працягу ўсяго цыклу распрацоўкі праграмнага забеспячэння, а не аб'ядноўваць вынікі працы кожнай каманды ў канцы цыкла.

Прынцыпы ашчаднай распрацоўкі праграмнага забеспячэння дазволілі перайсці да больш гібкіх мадэлей распрацоўкі праграмнага забеспячэння. Метадалогіі на прынцыпах Agile распрацоўкі прывялі да больш хуткага рэагавання на новыя патрабаванні ў параўнанні з традыцыйнымі мадэлямі распрацоўкі.

1.2.4 Мера.

Гамбэл і Малескі[6] вызначаюць меру ў Devops, як высокаўзроўневы кантроль бізнес-паказальнікаў, напрыклад даход альбо колькасць аперацый за адзінку часу. На больш нізкім узроўні гэта патрабуе дбайнага выбару ключавых паказчыкаў ў працэсе дастаўкі: час працэсу дастаўкі, уплыў новых рэлізаў на стабільнасць сістэмы.

Мера ўплывае на тое, як людзі працуюць. Візуалізацыя паказальнікаў дазваляе дакладна вызначыць прадукцыйнасць каманды ў любы момант часу. Паказальнікі таксама дапамагаюць вызначыць вузкія месцы ў працэсах распрацоўкі, што ў сваю чаргу дапамагае падтрымліваць ашчаднасць распрацоўкі.

У метадалогіі Devops могуць выкарыстоўвацца такія паказальнікі як[8]:

– частата разгортвання праграмнага забеспячэння (як часта адбываецца разгортванне новых рэлізаў праграмнага забеспячэння);

– аб’ём змяненняў (як шмат змяненняў, выпраўленняў памылак уваходзіць у новы рэліз);

– час разгортвання (час, які неабходны, для разгортвання новага рэлізу);

– час працэсу распрацоўкі (час ад пачатку стварэння прадукту да яго разгортвання);

– працэнт праходжання аўтаматычных тэстаў (дазваляе вызначыць як часта змяненні кода прыводзяць да памылак у праграмным забеспячэнні);

– каэфіцыент памылак (вызначае колькасць памылак, якія не былі вызначаны падчас тэсціравання праграмнага забеспячэння перад разгортваннем);

– сярэдні час знаходжання памылак (як хутка знаходзяцца памылкі пасля разгортвання);

– сярэдні час аднаўлення сістэмы (як хутка выпраўляюцца памылкі пасля іх знаходжання).

1.2.5 Абмен.

Апошнім кампанентам мадэлі CALMS з’яўляецца Абмен. Празрыстая культура з эфектыўнымі механізмамі распаўсюджвання ведаў паміж камандамі – неабходнасць для таго, каб знішчыць сцяну паміж аддзеламі распрацоўкі і тэхнічнай эксплуатацыі.

Гамбэл і Малескі[6] лічаць, што сумеснае святкаванне паспяховых рэлізаў, сумеснае размяшчэнне, сустрэчы каманд твар да твару дапамагаюць узмацніць сацыяльны кампанент медаталогіі Devops.

Наяўнасць часу ў работнікаў для эксперыментаў з новымі інструментамі і спосабамі выканання працы, падахвочванне іх дзяліцца атрыманымі вынікамі з астатняй часткай арганізацыі, прыносіць карысць арганізацыі ў цэлым.

Падобна на астатнія кампаненты мадэлі CALMS, абмен таксама палягчаецца пры да-

памозе новых інструментаў для сумеснай працы. Напрыклад, увядзенне кантролю версій для ўсяго і наступны абмен рэпазіторыямі з кодам паміж камандамі дапамагае ўсім камандам адсочваць якія тэхнікі і тэхналогіі выкарыстоўваліся, якія былі ўнесены змены і гэтак далей.

Ключавым элементам для матывацыі разбураць сцены непаразумення паміж камандамі з'яўляецца супольная адказнасць. Пашырэнне адказнасці з аддзела тэхнічнай эксплуатацыі на аддзел распрацоўшчыкаў і рэарганізацыя каманд ў супольныя групы спрыяе супрацоўніцтву.

Раней працэсамі разгортвання займаліся толькі групы тэхнічнай эксплуатацыі. У Devops ў дадзены працэс уключаны ўсе каманды, што спрыяе ўзмацненню матывацыі ў кожнага супрацоўніка зрабіць працэс разгортвання максімальна лёгкім. Замест таго, каб толькі выклікаць каманды тэхнічнай эксплуатацыі ў выпадку ўзнікнення надзвычайных сітуацый, распрацоўшчыкі могуць папярэджваць, калі існуе магчымасць, што нешта пойдзе не так. Супольная сістэма маніторынгу на ўсіх этапах павышае празрыстасць і заклікае іншыя каманды разгледзець тое, што адбываецца ў інфраструктуры ў цэлым.

Укараненне інфраструктуры як код стварае магчымасці для абмену існуючых сцэнарыяў і інструментаў паміж усімі камандамі арганізацыі. Наяўнасць сярод каманд арганізацыі агульных практык, адзіных аперацыйных сістэм, моў праграмавання спрашчае жыццё кожнага супрацоўніка. Напрыклад выкарыстоўванне Docker дазваляе аб'ядноўваць сістэмы так, што ўсе серверы маюць аднолькавыя сцэнарыі ўстаноўкі, і не абмяжоўваць выбар тэхналогій пры распрацоўцы праграм.

Нарэшце, неабходна заўважыць, што абмен ведамі і практыкамі не павінен абмяжоўвацца толькі арганізацыяй. Вывучэнне паспяховых метадаў укаранення практык Devops у іншых арганізацыях, адкрыты абмен уласнага досведу як спецыяліста, так і арганізацыі з іншымі спецыялістамі ці арганізацыямі дазваляе ўдасканаліць уласныя метады кіравання і павелічэння прадукцыйнасці арганізацыі.

1.3 Перавагі Devops

Існуюць відавочныя перавагі, якія можна атрымаць пры ўкараненні розных кампанентаў метадалогіі Devops. У дадзеным падраздзеле прыводзіцца кароткі пералік найбольш важных пераваг метадалогіі Devops.

Асноўная перавага метадалогіі Devops – скарачэнне часу разгортвання (lead time) і павялічаная частата разгортвання.

Асноўнымі фактарамі для дадзеных паказчыкаў з'яўляюцца:

– прымяненне метадалогіі ашчаднасці для памяншэння памеру змен у рэлізах;

– аўтаматызацыя ў працэсах дастаўкі.

Згодна са справаздачай статыстыкі Devops за 2015 год[9], час разгортвання ў высокапрадукцыйных IT-кампаніях у 200 разоў меншы, чым у малаэфектыўных кампаніях. Акрамя таго, за кошт мінімізацыі вузкіх месцаў і памяншэння праблем з разгортваннем, высокапрадукцыйныя кампаніі выпускаюць новыя рэлізы ў 30 разоў часцей, чым малаэфектыўныя. У сувязі з падобнымі вынікамі даследванняў, буйныя IT-кампаніі аддаюць перавагу Devops падыходам

распрацоўкі праграмнага забеспячэння.

Большая колькасць рэлізаў азначае паскарэнне выхаду прадукцыі на рынак і паляпшэнне зваротнай сувязі. У сувязі з паляпшэннем зваротнай сувязі кампаніі маюць лепшае ўяўленне пра рэакцыю кліентаў на новыя магчымасці праграмы, гэта значыць забяспечвае бесперапыннае ўдасканальванне прадукту.

Іншая перавага Devops – гэта паляпшэнне каэфіцыента паспяховых змяненняў. З добра спланаванымі працэсамі бесперапыннай распрацоўкі кампанія памяншае колькасць праблем падчас фазы разгортвання. Каэфіцыент паспяховых змяненняў – гэта працэнт змяненняў, якія прынеслі поспех, пасля ўжывання ў вытворчасці.

Згодна са справаздачай статыстыкі Devops за 2015 год[9], каэфіцыент паспяховых змяненняў высокапрадукцыйных кампаній ў 60 разоў большы, чым ў неэфектыўных кампаніях.

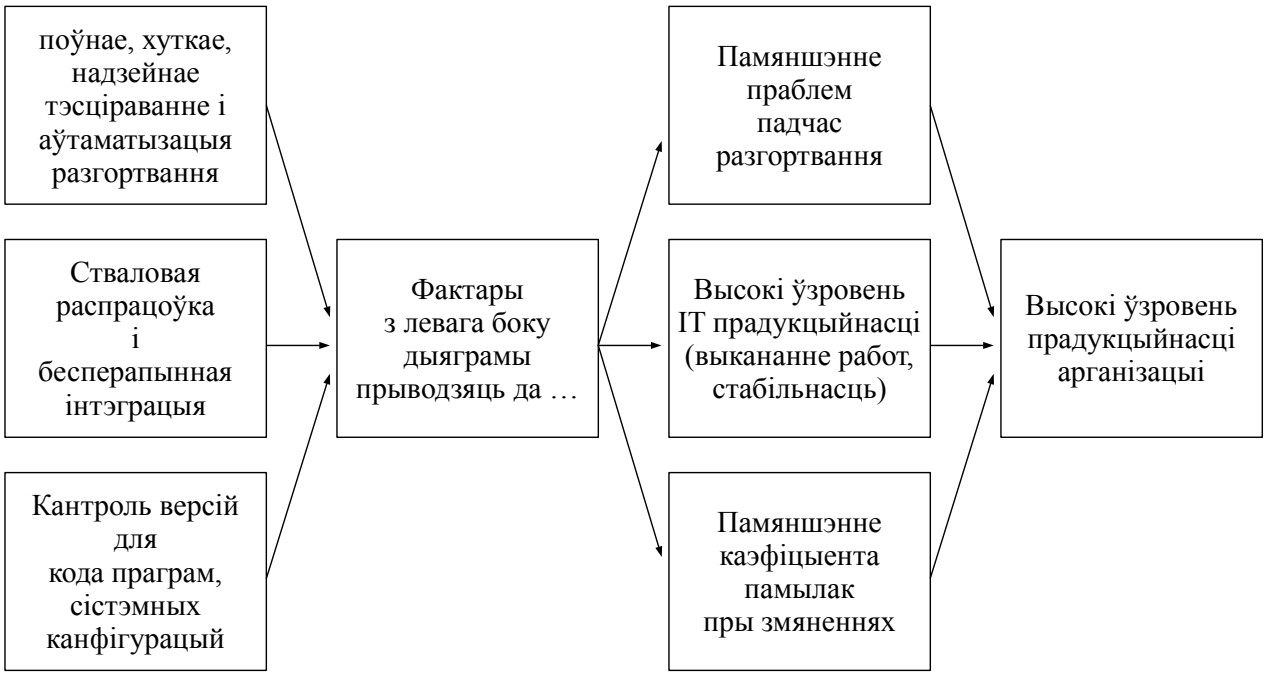
Каэфіцыент паспяховых змяненняў павялічваецца пры ўжыванні кантролю версій, інфраструктуры як код, аўтаматычнага тэсціравання, агульных механізмаў зборкі для розных асяроддзяў з мэтай прадухіліць памылкі ў кодзе і адрозненні ў канфігурацыі. Павелічэнне каэфіцыента паспяховых змяненняў уплывае на задаваленасць супрацоўнікаў, памяншае ўзровень перагарання ў штаце, які выклікаецца ў сувязі са стрэзам праз незапланаваныя, бессэнсоўныя работы.

Devops таксама спрыяе паляпшэнню стабільнасці. Пад стабільнасцю разумеецца беспамылковая праца сістэмы і яе здольнасць апрацоўваць новыя запыты. Павышэнне ўзроўню стабільнасці ажыццяўляецца пры дапамозе кіравання якасцю ў сістэме праз аўтаматычныя тэсты, распрацоўку праграм з прадугледжваннем яе наступнага тэсціравання і разгортвання, стварэнне культуры бесперапыннага ўдасканальвання сярод супрацоўнікаў.

Іншы паказальнік, які можа выкарыстоўвацца для колькаснай ацэнкі прадукцыйнасці, сярэдні час аднаўлення (MTTR). Дадзены паказальнік вызначае неабходны час для аднаўлення пасля ўзнікнення праблем з сервісам. Пры дапамозе кантролю версій і аўтаматычнага разгортвання пераход на папярэднюю версію праграмы ажыццяўляецца значна прасцей. Так, кампаніі, каторыя выкарыстоўваюць прынцыпы Devops, маюць паказальнік MTTR у 168 разоў меншы, чым ў кампаній, каторыя прытрымліваюцца традыцыйных метадалогій распрацоўкі праграмнага забеспячэння[9].

Нарэшце, Devops зніжае выдаткі. Адпаведна даследванню, якое праводзіла арганізацыя CA Technologies[10] выяўлена, што 46% рэспандэнтаў адзначалі зніжэнне выдаткаў на аддзелы распрацоўшчыкаў і аддзелы тэхнічнай эксплуатацыі, 45% рэспандэнтаў плануюць убачыць зніжэнне ў найбліжэйшы час.

На малюнку 1.2 прыведзена дыяграма адносін паміж Devops практыкамі і яго перавагамі[4].



Малюнак 1.2 – Дыяграма адносін паміж Devops і яго перавагамі

2 БЕСПЕРАПЫННАЯ ІНТЭГРАЦЫЯ І ДАСТАЎКА

2.1 Асноўныя паняцці

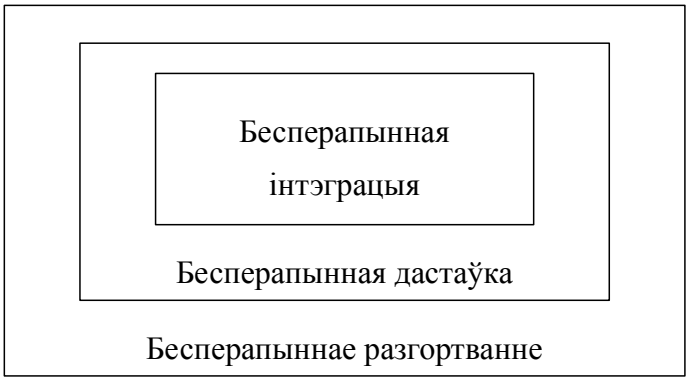
Бесперапынная інтэграцыя азначае, што зборка і тэсціраванне (Unit-testing) праграмнага забеспячэння адбываюцца аўтаматычна і часта. Частата азначае, зборка праграмнага забеспячэння адбываецца перыядычна альбо, напрыклад, пры кожным змяненні ў кантролі версій змяненняў. Заўважым, што ключавым момантам ў бесперапыннай інтэграцыі з'яўляецца тое, што працэс тэсціравання (інтэграцыя праграмнага забеспячэння) аўтаматызаваны пры дапамозе інструментаў і не патрабуе ручнога ўмяшання падчас зборкі і тэсціравання[11].

Бесперапынная дастаўка азначае, што праграмнае забеспячэнне заўсёды гатовае да разгортвання. Праграмнае забеспячэнне збіраецца і тэсціруецца падчас працэсаў бесперапыннай інтэграцыі і, таксама, разгортваецца ў адмысловае асяроддзе для далейшых тэстаў. Ключавым момантаў у бесперапыннай дастаўцы з'яўляецца тое, што кожная версія зборкі аўтаматычна правяраецца на гатоўнасць да разгортвання[12].

Бесперапыннае разгортванне азначае аўтаматычнае разгортванне праграмнага забеспячэння ў вытворчасць пасля таго, як унесены змены ў вытворчую галіну кантроля версіі і паспяхова пройдзены аўтаматычныя тэсты на праверку гатоўнасці да разгортвання. Зборка, тэсціраванне, разгортванне ажыццяўляюцца без ўмяшання чалавека[13].

Такім чынам адрозненнем паміж бесперапыннай дастаўкай і разгортваннем з'яўляецца тое, што пры працэсе бесперапыннай дастаўкі праграмнае забеспячэнне сцвярджаецца гатовым да разгортвання ў вытворчасць чалавекам. У той жа час пры працэсе бесперапыннага разгортвання праграмнае забеспячэнне аўтаматычна разгортваецца ў вытворчасць.

Бесперапынная дастаўка і разгортванне часта памылкова ўжываюцца як сінонімы, аднак неабходна разумець, што розніца паміж гэтымі двума тэрмінамі існуе. На малюнку 2.1 прадстаўлены ўзаемаадносіны паміж бесперапыннай інтэграцыяй, дастаўкаў і разгортваннем.

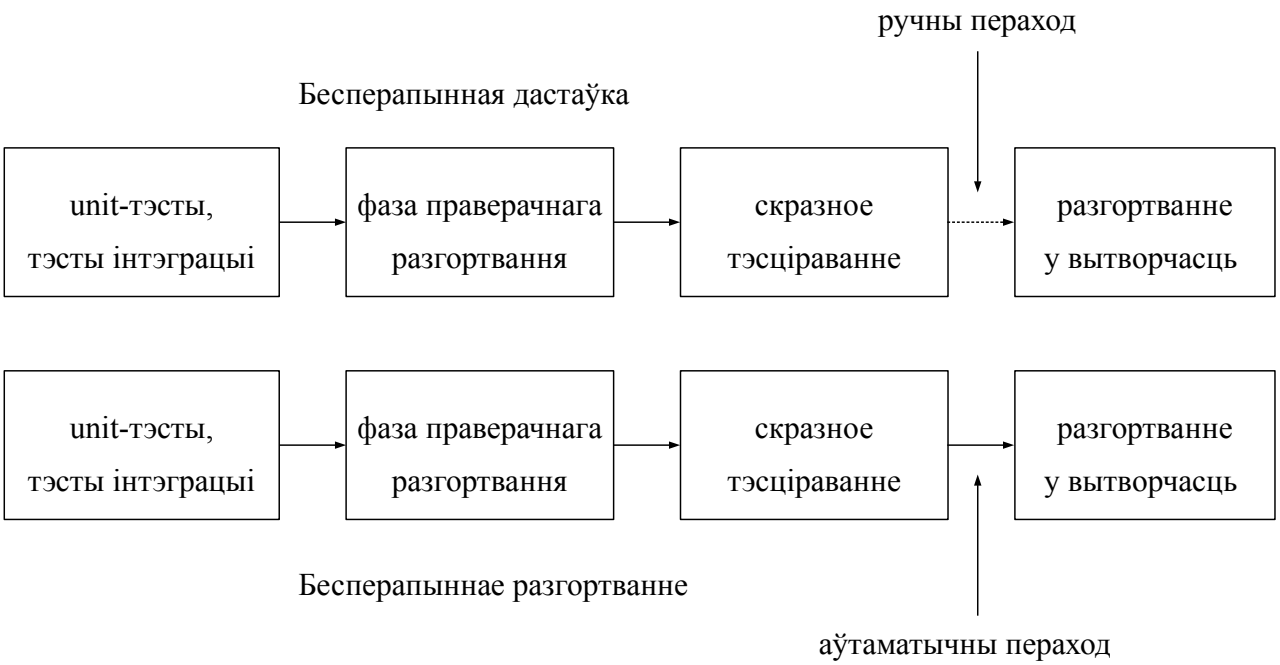


Малюнак 2.1 – Узаемаадносіны паміж бесперапыннай інтэграцыяй, дастаўкай і разгортваннем

Згодна з малюнкам 2.1 можна сцвярджаць, што немагчыма пабудаваць канвеер бесперапыннага разгортвання без працэсаў бесперапыннай інтэграцыі і дастаўкі, так як адно ўключае

ў сябе другое.

На малюнку 2.2 прадстаўлены канвееры бесперапыннай дастаўкі і разгортвання.



Малюнак 2.2 – Адрозненне паміж бесперапыннай дастаўкай і разгортваннем

Падвядзем ітог тэрміналогіі:

– Бесперапынная інтэграцыя – гэта набор практык, мэта каторага паляпшэнне якасці і хуткасці распрацоўкі праграмнага забеспячэння пры дапамозе аўтаматызацыі зборкі і тэсціравання, а таксама памяншэнне імавернасці памылак праз ручную зборку альбо праз розныя ўласцівасці асяроддзя выканання зборкі. Бесперапынная інтэграцыя з’яўляецца часткай бесперапыннай дастаўкі;

– Бесперапынная дастаўка – гэта набор практык, які ўключае ў сябе бесперапынную інтэграцыю і дадае да яе аўтаматызацыю скразнога тэсціравання і дастаўку праграмнага забеспячэння такім чынам, каб канчатковая зборка праграмы магла прымяняцца на патрэбных платформах і пры пэўных умовах. Бесперапынная дастаўка з’яўляецца часткай бесперапыннага разгортвання і не ўключае ў сябе аўтаматычнае разгортванне праграмнага забеспячэння ў вытворчасць;

– Бесперапыннае разгортванне – гэта набор практык, які ўключае ў сябе вышэй узгаданыя бесперапынную інтэграцыю і дастаўку, аднак дадае да іх аўтаматычнае разгортванне праграмнага забеспячэння ў вытворчасць. Пры ідэальных умовах, канвеер бесперапыннага разгортванне не патрабуе чалавечага ўмяшання для абнаўлення альбо выпуску новых рэлізаў праграмнага забеспячэння.

2.2 Перавагі бесперапыннай інтэграцыі

Пры ўмове, што ўкараненне бесперапыннай інтэграцыі адбываецца карэктна, яна прадастаўляе камандзе распрацоўшчыкаў наступныя перавагі[11]:

– памяншэнне рызык;

– аўтаматызацыя працэсаў, якія паўтараюцца;

– прыдатнасць праграмнага забеспячэння да разгортвання дзе заўгодна і калі заўгодна;

– лепшая бачнасць праекта;

– большая ўпэўненасць каманды ў якасці праграмнага забеспячэння.

Найбольш папулярнай перавагай бесперапыннай інтэграцыі з'яўляецца памяншэнне рызык. Гэта дасягаецца тым, што інтэграцыя (аб'яднанне розных частак праграмы) адбываецца некалькі разоў на дзень. Такая частата інтэграцыі дазваляе распрацоўшчыкам хутка знаходзіць і выпраўляць памылкі, а таксама адсочваць якасці праграмнага забеспячэння з меншым інтэрвалам часу. Нарэшце, частая зборка дазваляе атрымліваць у рэальным часу і дакладную зваротную сувязь пасля кожнай інтэграцыі.

Аўтаматызацыя руцінных задач не меншая перавага ад бесперапыннай інтэграцыі. Аўтаматызаваныя сістэмы і працэсы захоўваюць час, грошы і сілы. Такім чынам аўтаматызацыя дазваляе супрацоўнікам сканцэнтравацца на іншых задачах.

З пункту гледжання кліента, вялікай перавагай з'яўляецца факт таго, што праграмнае забеспячэнне заўсёды прыдатнае да разгортвання. Дзякуючы бесперапыннай інтэграцыі праграмнае забеспячэнне можа быць лёгка абноўлена невялікімі змяненнямі і адразу ж гатоввым да разгортвання.

Бесперапынная інтэграцыя дазваляе распрацоўшчыкам магчымасць бачыць стан усяго праекта. Напрыклад, у мінулым для таго, каб прыняць рашэнне аб абнаўленні, распрацоўшчыку патрабавалася збіраць неабходныя даныя ўручную альбо толькі рабіць прапановы наконт невядомых даных. Як вынік, працэс збору інфармацыі займаў шмат часу і энергіі, і ў той жа час мог не забяспечваць усёй неабходнай інфармацыей. Аднак дзякуючы ўкараненню практыкі бесперапыннай інтэграцыі, дадзены працэс стаў нашмат прасцей. Сістэма прапаноўвае ў рэальным часу інфармацыю пра апошнюю зборку праграмнага забеспячэння, адсотак памылак і прагрэс завяршэння.

Нарэшце, калі сістэма прапускае праграмнае забеспячэнне праз серыю аўтаматычных тэстаў і праверак, каманда распрацоўшчыкаў упэўнена ў якасці праграмы і яе прыдатнасці да разгортвання пры ўмове, што зборка завяршылася паспяхова. У адваротным выпадку, сістэма неадкладна інфарміруе распрацоўшчыкаў аб наяўнасці праблем і іх прычын.

2.3 Перавагі бесперапыннай дастаўкі

Адпаведна вышэй напісанаму, што бесперапынная інтэграцыя з'яўляецца часткай бесперапыннай дастаўкі, вынікае, што бесперапыннай дастаўка мае ўсе перавагі бесперапыннай інтэграцыі.

Аднак бесперапынная дастаўка прапаноўвае і іншыя перавагі, якія памяншаюць імавернасць узнікнення стрэсавай сітуацыі ў супрацоўнікаў. Пры дапамозе аўтаматызацыі кожнага ручнога працэса, які неабходна час ад часу паўтараць, супрацоўнікі пазбягаюць памылак і заўжды перакананы, што праграмнае забеспячэнне стабільнае, правільна сканфігураванае і прыдатнае да разгортвання. Як вынік, дзень рэліза праграмнага забеспячэння праходзіць менш інтэнсіўна і з меншым стрэсам. Што, у сувязі з паскоранымі тэмпамі выпуску новых прадуктаў альбо абнаўленій старых версій праграм, становіцца неабходным складнікам для падтрымання стабільнага эмацыянальнага становішча ўнутры кампаніі.

Нарэшце, так як працэс выпуску новых рэлізаў становіцца больш гладкім і стабільным, гэта дазваляе кампаніі хутчэй дастаўляць праграмнае забеспячэнне кліентам. Бесперапынная дастаўка развязвае задачу пра неабходнасць неадкладнай дастаўкі кліентам якасных версій праграмнага забеспячэння пры дапамозе аўтаматызаваных канфігурацый вытворчага асяроддзя, што забяспечвае прыдатнасць праграмнага забеспячэння да разгортвання ў любы час і на любую платформу.

2.4 Інструменты бесперапыннай інтэграцыі і дастаўкі

У дадзеным падраздзеле разглядаюцца некаторыя папулярныя інструменты для рэалізацыі прынцыпаў бесперапыннай інтэграцыі і дастаўкі, і іх асаблівасці.

2.4.1 Jenkins

Jenkins – найбольш папулярны праект з адкрытым кодам для аўтаматызацыі. Дадатковае мноства плагінаў Jenkins дазваляе аўтаматызаваць любую неабходную задачу. Jenkins выкарыстоўваецца для зборкі праекта, запуску тэстаў, аналізу кода і разгортвання праекта.

Асаблівасці Jenkins:

– Jenkins можа працаваць як незалежны CI сервер альбо як платформа бесперапыннай дастаўкі для любога праекта;

– наяўнасць зборак для Unix, Windows і OS X, што спрашчае працэс установкі;

– наяўнасць Web-інтэрфейса для хуткай канфігурацыі сервера;

– разнастайныя плагіны для зборкі і кіравання зыходным кодам, задач адміністравання, інтэрфейсу карыстальніка.

2.4.2 Travis

Travis CI – гэта платформа бесперапыннай інтэграцыі для аўтаматызацыі працэсаў тэсціравання і разгортвання праграмнага забеспячэння. Travis CI рэалізуецца ў якасці платформы, якая аб’ядноўваецца з GitHub праектамі, што дазваляе праводзіць тэсціравання кода ў рэальным часе.

Travis CI прадугледжвае бясплатнае выкарыстанне для адкрытых праектаў і платная для камерцыйный праектаў.

Асаблівасці Travis CI:

– бясплатная версія для акрытых праектаў на GitHub;

– простая рэгістрацыя і дабаўленне праектаў;

– аўтаматызаваная праверка pull-запытаў;

– багатая падтрымка паведамленняў (напрыклад, e-mail, Slack, HipChat);

– наяўнасць API і кансольных інструментаў для гібкай настройкі.

2.4.3 GitLab CI

GitLab – хуткаразвіваемая платформа для кіравання коду. GitLab прапаноўвае ў межах адзінай панелі інструменты для кіравання рэлізаў, прагляду коду, бесперапыннай інтэграцыі і разгортвання.

GitLab прадугледжвае бясплатнае выкарыстанне для адкрытых праектаў і платная для камерцыйный праектаў.

Асаблівасці GitLab CI:

– наяўнасць зборак для папулярных версій Linux;

– прыязны інтэрфейс карыстальніка;

– падрабязная дакументацыя для кожнай функцыі;

– магчымасць індывідуальнай настройкі тэстаў для кожнай галіны праекта;

– ручное разгортванне і лёгкай магчымасць вяртання на папярэднію версію.

2.4.4 Chef

Chef забяспечвае асяроддзе для аўтаматызацыі кіравання інфраструктурай, разгортваннем, канфігурацыяй незалежна ад памераў сеткі. Chef можна лёгка інтэгравацца ў воблачныя сервісы, фізічныя серверы альбо гібрыдныя рашэнні.

Асаблівасці Chef:

– убудаваныя наборы інструментаў для аўтаматызацыі інфраструктуры як код;

– магчымасць кіравання як воблачнымі, так і фізічнымі серверамі;

– лёгкі перавод праграмы паміж воблачнымі серверамі.

2.4.5 Puppet

Платформа Puppet выкарыстоўваецца для кіравання канфігурацыяй для Unix і Windows сістэмы. Puppet – інструмент кіравання канфігурацыяй з адкрытым кодам. Puppet дазваляе распрацоўшчыкам дастаўляць і выкарыстоўваць праграмнае забеспячэнне незалежна ад яго крыніцы.

GitLab прадугледжвае бясплатны тэрмін выкарыстання для азнакамлення.

Асаблівасці Puppet:

– наяўнасць гібкай настройкі Puppet-агента і кансолей, што дазваляе аб’ядноўваць працэсы бесперапыннай інтэграцыі;

– наяўнасць бясплатных модулей для хуткай распрацоўкі;

– прадугледжваецца сумесная работа з Git і Jenkins для бесперапыннай аўтаматызацыі.

2.4.6 Better Code Hub

Better Code Hub – Web-сервіс для аналізу зыходнага кода.

Better Code Hub прадугледжвае бясплатнае выкарыстанне для адкрытых праектаў і платная для камерцыйный праектаў.

Асаблівасці Better Code Hub:

– неадкладная зваротная сувязь на кожнае змяненне кода;

– інтэграцыя з GitHub;

– гібкая настройка прыярытэтаў дзеянняў.

2.4.7 Docker

Docker – праграмнае забеспячэнне для аўтаматызацыі разгортвання і кіравання праграмамі ў асяроддзі з падтрымкай кантэйнерызацыі. Docker дазваляе ўпакаваць праграму з неабходным асяроддзем і залежнасцямі ў кантэйнер, які можа быць перанесены на любую Linux-сістэму.

Асаблівасці Docker:

– незалежнасць праграмы ад сістэмы, на якой выконваецца запуск;

– павялічаная бяспека сістэмы;

– лёгкая дастаўка праграмнага забеспячэння кліенту.

2.4.8 Git і GitHub

Git – размеркаваная сістэма кантролю версій, якая дае магчымасць распрацоўшчыкам адсочваць змены ў файлах і працаваць сумесна з іншымі распрацоўшчыкамі. GitHub – сервіс анлайн-хостынга рэпазіторыяў, якія маюць усе функцыі размеркаванага кантролю версій (Git) і функцыянальнасцю кіравання зыходным кодам. GitHub дае распрацоўшчыкам магчымасць захоўваць код анлайн, а затым узаемадзейнічаць з іншымі распрацоўшчыкамі ў розных праектах. Да праекта, які размешчаны на GitHub, можна атрымаць доступ пры дапамозе інтэрфейса каманднага радку Git і Git-каманд.

Асаблівасці Git:

– рэзервнае капіраванне;

– простае галінаванне;

– высокая прадукцыйнасць.

3 РЭАЛІЗАЦЫЯ КАНВЕЕРА БЕСПЕРАПЫННАЙ ІНТЭГРАЦЫІ І ДАСТАЎКІ

У дадзеным раздзеле будзе разгледжаны прыклад рэалізацыі канвеера бесперапыннай інтэграцыі і дастаўкі пры дапамозе open-source рашэнняў[14].

3.1 Пастаноўка задачы

Неабходна распрацаваць канвеер бесперапыннай інтэграцыі і дастаўкі для Web-праграмы, якая напісаная на мове праграмавання Python. У канвееры неабходна настроіць выкананне аўтаматызаваных тэстаў і пастаянную праверку якасці кода. Запуск праграмы мае ажыццяўляцца ўнутры Docker-кантэйнера.

3.2 Праграма генератара фраз

Створым невялікую кансольную праграму, якая будзе генерыраваць фразы. Зыходны код прадстаўлены ў лістынгу 3.1:

```
import random

buzz = ('continuous testing', 'continuous integration',
    'continuous deployment', 'continuous improvement', 'devops')
adjectives = ('complete', 'modern', 'self-service', 'integrated', 'end-to-end')
adverbs = ('remarkably', 'enormously', 'substantially', 'significantly',
    'seriously')
verbs = ('accelerates', 'improves', 'enhances', 'revamps', 'boosts')

def sample(l, n = 1):
    result = random.sample(l, n)
    if n == 1:
        return result[0]
    return result

def generate_buzz():
    buzz_terms = sample(buzz, 2)
    phrase = ' '.join([sample(adjectives), buzz_terms[0], sample(adverbs),
        sample(verbs), buzz_terms[1]])
    return phrase.title()

if __name__ == "__main__":
    print(generate_buzz())
```

Лістынг 3.1 – Зыходны код праграмы generator.py

На малюнку 3.1 прадстаўлены вывад праграмы generator.py, дзе можаш пераканацца ў тым, што пры кожным запуску вывад праграмы адрозніваецца.

```
dranser@Dranser:~/Documents/BSAC/2 year/6 semeseter/TPiCП3/cicd-buzz/buzz$ python3 generator.py
End-To-End Continuous Deployment Significantly Improves Continuous Testing
dranser@Dranser:~/Documents/BSAC/2 year/6 semeseter/TPiCП3/cicd-buzz/buzz$ python3 generator.py
Integrated Continuous Improvement Significantly Accelerates Continuous Deployment
dranser@Dranser:~/Documents/BSAC/2 year/6 semeseter/TPiCП3/cicd-buzz/buzz$ python3 generator.py
End-To-End Devops Enormously Revamps Continuous Integration
dranser@Dranser:~/Documents/BSAC/2 year/6 semeseter/TPiCП3/cicd-buzz/buzz$ python3 generator.py
Integrated Continuous Deployment Enormously Accelerates Continuous Integration
dranser@Dranser:~/Documents/BSAC/2 year/6 semeseter/TPiCП3/cicd-buzz/buzz$ python3 generator.py
Modern Continuous Deployment Seriously Revamps Continuous Improvement
```

Малюнак 3.1 – Вывад праграмы generator.py

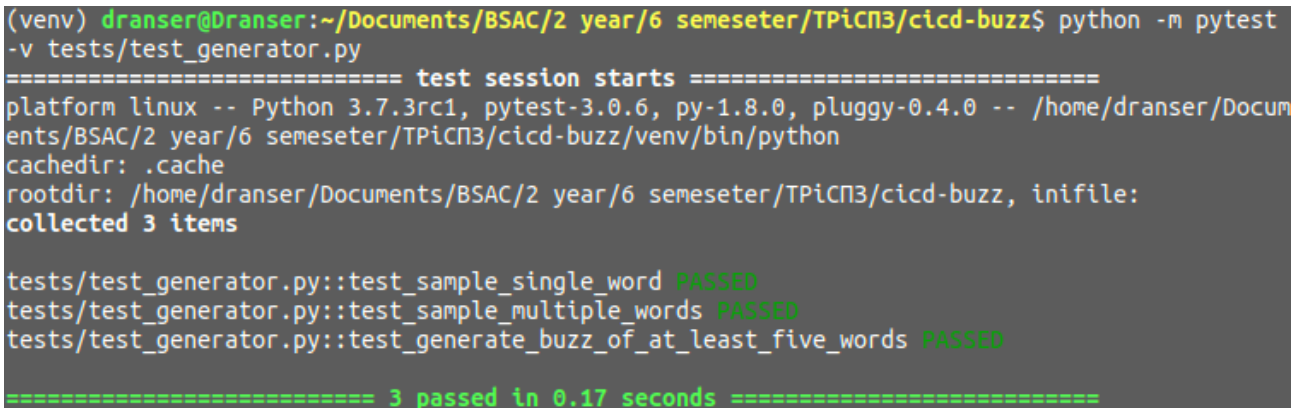
3.3 Аўтаматызаваныя тэсты

Створым некалькі unit-тэстаў для праверкі працаздольнасці праграмы. Зыходны код unit-тэстаў прадстаўлены ў лістынгу 3.2:

```
import unittest
import sys

sys.path.append("..")

from buzz import generator

def test_sample_single_word():
    l = ('foo', 'bar', 'foobar')
    word = generator.sample(l)
    assert word in l

def test_sample_multiple_words():
    l = ('foo', 'bar', 'foobar')
    words = generator.sample(l, 2)
    assert len(words) == 2
    assert words[0] in l
    assert words[1] in l
    assert words[0] is not words[1]

def test_generate_buzz_of_at_least_five_words():
    phrase = generator.generate_buzz()
    assert len(phrase.split()) >= 5
```

Лістынг 3.2 – Зыходны код праграмы test_generator.py

Unit-тэсты будуць запускацца пры дапамозе фрэймфворка ”pytest”. Вывад запуску unit-тэстаў прадстаўлены на малюнку 3.2.



Малюнак 3.2 – Вывад праграмы test_generator.py

На малюнку 3.2 можам бачыць, што праграма generator.py паспяхова прайшла ўсе unit-тэсты.

3.4 Выкананне тэстаў пры дапамозе Travis CI

Для выканання тэстаў у аўтаматычным рэжыме пры кожным змяненні кода на GitHub неабходна падключыць для дадзенага рэпазіторыя сервіс Travis CI і паставіць настройкі для выканання зборкі пры кожным push альбо pull запыце рэпазіторыя (малюнак 3.3).

Малюнак 3.3 – Настройка зборкі пры push- і pull-запытах у Travis CI

Апошні крок для актывацыі Travis CI заключаецца ў дабаўленні файла канфігурацыі ”.travis.yml змест каторага прадстаўлены ў лістынгу 3.3.

```
language: python
script:
    - python -m pytest -v
```

Лістынг 3.3 – Канфігурацыя Travis CI

Цяпер пры кожным push- альбо pull-запыце да рэпазіторыя на GitHub у аўтаматычным рэжыме будуць запускацца unit-тэсты на сервісе Travis CI. Рэзультаты unit-тэстаў можна праглядаць у кансолі Travis CI (малюнак 3.4).

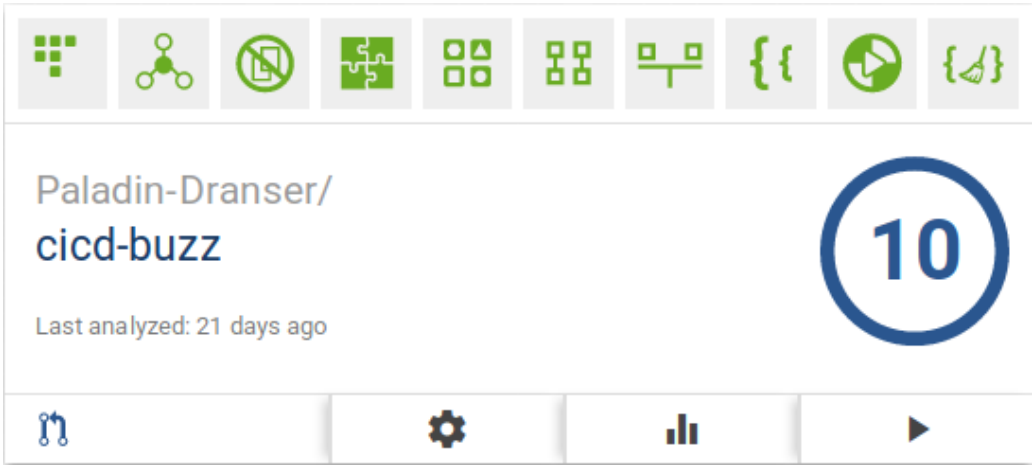
```
1  Worker information                                                                    worker_info
6  Build system information                                                              system_info
413
414 Using the default Python version 2.7. Starting on 2019-04-16 the default will change to 3.6. If you wish to keep using this
    version beyond this date, please explicitly set the Python value in configuration.
415
416 $ git clone --depth=50 --branch=master https://github.com/Paladin-Dranser/cicd-buzz.git Paladin-   git.checkout  0.51s
425
426 $ source ~/virtualenv/python2.7/bin/activate                                                        0.01s
427 $ python --version
428 Python 2.7.14
429 $ pip --version
430 pip 9.0.1 from /home/travis/virtualenv/python2.7.14/lib/python2.7/site-packages (python 2.7)
431 $ pip install -r requirements.txt                                                    install       2.42s
458 $ python -m pytest -v                                                                            0.33s
459 ============================= test session starts ==============================
460 platform linux2 -- Python 2.7.14, pytest-3.0.6, py-1.5.2, pluggy-0.4.0 -- /home/travis/virtualenv/python2.7.14/bin/python
461 cachedir: .cache
462 rootdir: /home/travis/build/Paladin-Dranser/cicd-buzz, inifile:
463 collected 3 items
464
465 tests/test_generator.py::test_sample_single_word PASSED
466 tests/test_generator.py::test_sample_multiple_words PASSED
467 tests/test_generator.py::test_generate_buzz_of_at_least_five_words PASSED
468
469 ========================== 3 passed in 0.03 seconds ==========================
470 The command "python -m pytest -v" exited with 0.
471
472
473
474 Done. Your build exited with 0.
```

Малюнак 3.4 – Вывад рэзультатаў праходжання unit-тэстаў у кансолі Travis CI

3.5 Праверка якасці кода пры дапамозе Better Code Hub

Інтэграцыя сервіса Better Code Hub да рэпазіторыя на GitHub дазваляе аналізаваць якасць кода пры кожным push- і pull-запыце (падобна сервісу Travis CI).

Вынік праверкі якасці кода прадстаўлены на малюнку 3.5, на якім можам бачыць, што код адпавядае ўсім неабходным патрабаванням.



Малюнак 3.5 – Вынік праверкі якасці кода пры дапамозе Better Code Hub

3.6 Web-праграма для генерацыі фраз

Згодна з пастаўленай задачай ператворым нашу кансольную праграму генерацыі фраз ў Web-праграму.

Для гэтага створым новую праграмы з выкарыстаннем фрэймворка Flask, якая будзе выкарыстоўваць раней напісаны модуль generator.py.

Зыходны код праграмы прадстаўлены ў лістынгу 3.4:

```
import os
import signal

from flask import Flask
from buzz import generator

app = Flask(__name__)

signal.signal(signal.SIGINT, lambda s, f: os._exit(0))

@app.route("/")
def generate_buzz():
    page = '<html><body><h1>'
    page += generator.generate_buzz()
    page += '</h1></body></html>'
    return page

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=os.getenv('PORT')) # port 5000 is the default
```

Лістынг 3.4 – Зыходны код Web-праграмы app.py

Вынік Web-праграмы прадстаўлены на малюнку 3.6.

Малюнак 3.6 – Вынік Web-праграмы генерацыі фраз

3.7 Кантэйнерызацыя праграмы пры дапамозе Docker

Адпаведна пастаўленай задачы рэалізуем Docker-кантэйнер (пры дапамозе Dockerfile) для Web-праграмы.

Канфігурацыя Dockerfile прадстаўлена ў лістынгу 3.5.

```
FROM alpine:3.5
RUN apk add --update python py-pip
COPY requirements.txt /src/requirements.txt
RUN pip install -r /src/requirements.txt
COPY app.py /src
COPY buzz /src/buzz
CMD python /src/app.py
```

Лістынг 3.5 – Канфігурацыя Docker-кантэйнера

Згодна з канфігурацыяй Docker загрузіць вобраз alpine, установіць Python, pip, неабходныя кампаненты для запуску праграмы, Web-праграму. Web-праграма будзе запускацца пры запуску кантэйнера.

Такім чынам мы атрымалі Web-праграму, якая запускаецца ўнутры кантэйнера і не залежыць ад знешніх канфігурацый асяроддзя. Пры змяненні зыходнага кода праграмы альбо канфігурацый сервісаў адбываецца аўтаматычная зборка і тэсціраванне праграмы.

ЗАКЛЮЧЭННЕ

У дадзенай курсавой рабоце былі разгледзеныя асноўныя прынцыпы метадалогіі Devops, бесперапыннай інтэграцыі, бесперапыннай дастаўкі. Разглядаліся перавагі папулярных інструментаў для рэалізацыі бесперапыннай інтэграцыі і дастаўкі:

– Jenkins
– Travis CI
– GitLab CI
– Chef
– Puppet
– Better Code Hub
– Docker
– GitHub

Таксама ў курсавой рабоце былі прааналізаваны перавагі ўкаранення метадалогіі Devops, бесперапыннай інтэграцыі і дастаўкі ў працэс распрацоўкі праграмнага забеспячэння.

Асноўнымі перавагамі ўкаранення метадалогіі Devops з'яўляюцца: паляпшэнне культурнага асяроддзя ўнутры кампаніі, пашырэнне кругагляду работнікаў на працэсы вытворчасці, скарачэнне часу разгортвання, павялічаная частата разгортвання, памяншэнне выдаткаў на распрацоўкі і разгортванне праграмнага забеспячэння. Канвееры бесперапыннай інтэграцыі і дастаўкі прапаноўваюць памяншэнне рызыкі памылкі, аўтаматызацыю працэсаў, памяншэнне часу зваротнай сувязі на новы функцыі праграмы, упэўненасць ў якасці праграмнага забеспячэння.

У апошнім раздзеле курсавой работы прыводзіўся прыклад пабудовы канвеера бесперапыннай інтэграцыі і дастаўкі для Web-праграмы пры дапамозе бясплатных сервісаў.

СПІС СКАРЫСТАНЫХ КРЫНІЦ

1 Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale / O'Reilly Media. – Jennifer Davis, Katherine Daniels, 2016.

2 DevOps and AWS [Электронны рэсурс]. – Рэжым доступу : https://aws.amazon.com/devops/. – Дата доступу : 01.05.2019.

3 DevOps [Электронны рэсурс]. – Рэжым доступу : https://ru.wikipedia.org/wiki/DevOps. – Дата доступу : 01.05.2019.

4 Challenges in Adopting a Devops Approach to Software Development and Operations : Thesis / Aalot-yliopisto. – Joonas Hamunen, 2016.

5 DevOps: A Software Architect's Perspective / Addison-Wesley Professional. – Len Bass, Ingo Weber, Liming Zhu, 2015.

6 Lean Enterprise: How High Performance Organizations Innovate at Scale / O'Reilly Media. – Jez Humble, Joanne Molesky, 2015.

7 Lean Software Development: An Agile Toolkit / Addison-Wesley Professional. – Mary Poppendieck, Tom Poppendieck, 2003.

8 15 Metrics for DevOps Success [Электронны рэсурс]. – Рэжым доступу : https://stackify.com/15-metrics-for-devops-success/. – Дата доступу : 01.05.2019.

9 2015 State of DevOps Report Рэжым доступу : https://puppet.com/resources/whitepaper/2015-state-devops-report. – Дата доступу : 01.05.2019.

10 DevOps: The Worst-Kept Secret to Winning in the Application Economy : Databook / Research paper. – CA technologies, 2014.

11 Continuous Integration: Improving Software Quality and Reducing Risk / Addison-Wesley Professional. – Paul Duvall, Stephen M. Matyas III, Adrew Glover, 2007.

12 Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation / Addison-Wesley Professional. – Jez Humble, David Farley, 2010.

13 Continuous Delivery Vs. Continuous Deployment: What's the Diff? [Электронны рэсурс]. – Рэжым доступу : https://puppetlabs.com/blog/continuous-delivery-vs-continuous-deployment-whats-diff. – Дата доступу : 01.05.2019.

14 How to build a modern CI/CD pipeline [Электронны рэсурс]. – Рэжым доступу : https://medium.com/bettercode/how-to-build-a-modern-ci-cd-pipeline-5faa01891a5b. – Дата доступу : 01.05.2019.