ՀՏԴ 004.4'242:004.65

Գ.Հ. ՍԱՐԳՍՅԱՆ

ՄԵԾ ՏՎՅԱԼՆԵՐԻ ԳԱՂԹԻ ԳՈՐԾԸՆԹԱՑԻ ԿԱՌԱՎԱՐՄԱՆ ԵՆԹԱԿԱՌՈՒՑՎԱԾՔՆԵՐԻ ԱՎՏՈՄԱՏԱՑՈՒՄԸ

Աշխատանքում ուսումնասիրվել են մեծ տվյալների միգրացիայի (գաղթի) ու շարունակական ինտեգրման գործընթացի կառավարման հիմնական խնդիրները, և իրականացվել ենթակառուցվածքների վերլուծություն, համեմատություն ու հիմնավորված ընտրություն՝ հուսալի և արդյունավետ համակարգի կառուցման նպատակով:

Մեծ տվյալների գաղթի գործընթացում ավտոմատացված ենթակառուցվածքների կառավարման միջոցով հնարավոր է նվազեցնել մարդու միջամտությունը, օպտիմալացնել ռեսուրսների օգտագործումը և կրճատել ֆինանսական ծախսերը՝ ապահովելով մեծ տվյալների բեռնման և պահպանման բարձր հուսալիություն: Կոդով կառավարվող ենթակառուցվածքներն ու ավտոմատացման գործիքակազմը նպաստում են տվյալների արագ, անվտանգ և հուսալի տեղափոխմանը, ինչը կարևոր նշանակություն ունի ժամանակակից կազմակերպությունների տեխնոլոգիական հաջողությունների համար:

Խնդիրը ներառում է GitOps [1] մեթոդաբանությամբ ավտոմատացված ծրագրային միջավայրի ստեղծում, որն ունի ծանրաբեռնվածության բաշխման ենթակառուցվածք, նոր ճարտարապետական լուծումներ և առանձնահատկություններ:

Ենթակառուցվածքների կառավարման գործընթացի արդյունքում ստեղծվել է լիովին ավտոմատացված միջավայր, որտեղ ծրագրում փոփոխություններ կատարելուց հետո ավտոմատ կերպով կստեղծվի նոր, աշխատող միջավայր: Միջավայրը նախատեսված է տվյալների գերբեռնվածության մարման, կառավարման և սերվերի կողմից վերահսկման համար: Կատարվել է սերվերի ընտրություն՝ հաշվի առնելով բեռնվածության պահանջները: Ստեղծվել է արհեստական գերբեռնվածություն՝ համակարգի կայունությունը և ռեսուրսների բաշխումը գնահատելու համար: Առաջարկվել են բեռի բաշխման լուծումներ: Մշակվել է նոր ցանցային ճարտարապետություն: Այս լուծումները միտված են տվյալների մշակման գործընթացների օպտիմալացմանը, ենթակառուցվածքների արդյունավետության բարձրացմանը և ժամանակակից կազմակերպությունների տեխնոլոգիական մարտահրավերներին դիմակայելու կարողության ամրապնդմանը: Մեծ տվյալների պահպանման և գաղթի կազմակերպման համար առկա ավտոմատացված ծրագրային միջավայրերը ոչ միշտ են համատեղում նաև ծանրաբեռնվածության խնդիրները կամ ինտեգրում կառավարման ենթակառուցվածքները:

Հետազոտվել են ենթակառուցվածքների կառավարման գործիքները, կատարվել է ընտրություն, դիտարկվել են ծանրաբեռնվածության հիմնահարցերը, միգրացիայի խնդիրները, CI/CD (շարունակական ինտեգրում/շարունակական առաքում) ծրագրային ապահովման մշակումն ավտոմատացնելու գործընթացը՝ հավելվածների առաքման արդյունավետության և արագության բարելավման համար:

Առանցքային բառեր. սերվեր, ենթակառուցվածք, միգրացիա, ծանրաբեռնվածություն, համակարգ, առցանց տեխնոլոգիաներ, կառավարում, CI/CD:

Ներածություն: Տեղեկատվական տեխնոլոգիաների ժամանակակից զարգացումներն ու ամպային տեխնոլոգիաների լայն կիրառությունը խթանել են տարատեսակ առցանց ծառայությունների մատուցումը: Ժամանակակից կազմակերպությունները՝ առաջնորդվելով բիզնեսի հարաճուն պահանջներով, առաջնահերթ կարևորություն են տալիս տեղեկությունների հավաքագրման, մշակման և պահպանման գործընթացներին: Մեծ տվյալների հավաքագրումը հիմնականում իրականացվում է տվիչների, ներկառուցված համակարգերի, խելացի սարքերի և այլ սարքավորումների միջոցով, որոնցում մարդու միջամտությունը նվազագույն է կամ բացակայում է: Հավաքագրված տեղեկությունների անվնաս տեղափոխումը, մշտադիտարկումը, անվտանգ պահպանումը, վերլուծությունը և արխիվացումը շատ կարևոր են կազմակերպությունների արդյունավետ կառավարման և ռազմավարական որոշումների կայացման գործընթացներում: Տվյալների ճիշտ կառավարումը թույլ է տալիս կազմակերպություններին՝ ապահովելու տեղեկատվության հասանելիություն, նվազեցնելու ռիսկերը և օգտագործելու դրանք որպես ռազմավարական ռեսուրս՝ մրցունակությունը պահպանելու և զարգացնելու համար: Ավտոմատացված ենթակառուցվածքների կառավարումը հնարավորություն է տալիս՝ կառավարելու մեծ տվյալների գաղթի գործընթացը առավել հուսալի, մասշտաբային և ժամանակատարության տեսակետից արդյունավետ եղանակներով: Աշխատանքում մեծ տվյալների գաղթի, ծանրաբեռնվածության բաշխման և սերվերային տեղադրման ավտոմատացման համար կիրառվել է շարունակական ինտեգրման/շարունակական առաքման (CI/CD) [1,2] գործիքակազմը, որն ապահովում է մեծ տվյալների մշակման գործընթացների ավտոմատացված առաքման արդյունավետություն, կայունություն և արագություն: Մեծ տվյալների գաղթը (Data Migration) [3] բարդ և կարևոր գործընթաց է, որի նպատակն է տվյալների տեղափոխումն ու վերակառուցումը՝ նոր միջավայրում դրանց ամբողջականությունը, անվտանգությունն ու արդյունավետությունը պահպանելու համար: Մեծ տվյալների գաղթի առանձնահատկություններն են՝ տվյալների ամբողջականության ապահովում, տվյալները պետք է տեղափոխվեն առանց որևէ կորստի կամ փոփոխության, տեղափոխման ընթացքում պետք խուսափել տվյալների կրկնօրինակումներից կամ անհամապատասխանություններից: Անհրաժեշտ է ապահովել տվյալների գաղտնիություն՝ մուտքի վերահսկման մեխանիզմներով, կոդավորված կապի (encrypted channels) կիրառմամբ՝ տեղափոխման գործընթացում: Տվյալների արագ տեղափոխման համար ընտրվել են օպտիմալ մեթոդներ և գործիքներ, կիրառելով հավասարակշռված մոտեցում՝ բեռնվածության և ռեսուրսների սպառման միջև: Մեծ տվյալների գաղթի համար անհրաժեշտ է սահմանել ծանրաբեռնվածության կառավարման կոնկրետ եզրերը և նկարագրել բեռի վարքը սերվերում:

Հետազոտվել է GitOps մեթոդաբանությունը՝ ծրագրային ապահովման մշակման և առաքման գործընթացների կատարելագործման նպատակով: Այն համատեղում է մի շարք գործիքներ և մեթոդաբանություններ՝ ուղղված ծրագրային ապահովման մշակման արագության և որակի բարձրացմանը՝ միաժամանակ ապահովելով տվյալների հուսալի և անխափան տեղակայումը: GitOps-ը թույլ է տալիս ավտոմատացնել ծրագրային ապահովման մշակման և տեղակայման բազմաթիվ գործընթացներ, ինչը նպաստում է արդյունավետության բարձրացմանը, թիմային համագործակցության ամրապնդմանը և ծրագրային ապահովման արտադրանքի արագ ու հուսալի թողարկմանը: Այս մեթոդաբանությունը հնարավորություն է տալիս՝ հետևողականորեն կառավարելու տեղակայման գործընթացները, նվազեցնելու մարդկային սխալների հավանականությունը և ապահովելու թիմային աշխատանքի անհրաժեշտ թափանցիկությունն ու համակարգվածությունը: GitOps-ը հատկապես օգտակար է, երբ անհրաժեշտ է կայուն և վերահսկելի միջավայր՝ արագ և որակյալ արդյունքներ ստանալու համար:

Աշխատանքի նպատակն է ուսումնասիրել մեծ տվյալների գաղթի և շարունակական ինտեգրման (CI/CD) գործընթացի կառավարման խնդիրները, վերլուծել առկա ենթակառուցվածքները, կատարել հիմնավորված ընտրություն սերվերների համար և մշակել արդյունավետ լուծումներ տվյալների բեռնվածության կառավարման համար:

Առաջադրվել են հետևյալ խնդիրները.

- Վերլուծել մեծ տվյալների գաղթի առանձնահատկությունները և բացահայտել տվյալների տեղափոխման գործընթացում առաջացող խնդիրները:
- Առաջարկել մեթոդներ՝ տվյալների ամբողջականության և անվտանգության պահպանման համար:
- Հետազոտել ենթակառուցվածքները, գնահատել եղածները և կատարել ընտրություն:
- Կատարել սերվերի ընտրություն՝ հաշվի առնելով ծանրաբեռնվածության և տվյալների մշակման պահանջները: Ստեղծել արհեստական ծանրաբեռնվածություն: Նկարագրել բեռնվածության վարքը և դրա ազդեցությունը համակարգի վրա:
- Առաջարկել ծանրաբեռնվածության բաշխման մեխանիզմներ՝ համակարգի կայունությունն ապահովելու համար՝ օգտագործելով բալանսավորիչներ և հորիզոնական սանդղաչափում: Ստեղծել ճարտարապետություն, որը կհամապատասխանի տվյալների մշակումից և առաքումից բխող պահանջներին՝ ապահովելով համակարգի հուսալիությունը և հարմարվողականությունը:
- Կիրառել GitOps մեթոդաբանությունը՝ ավտոմատացման գործիքակազմով ծրագրային միջավայր ստեղծելու համար՝ ապահովելով արագ, կայուն և անխափան տեղակայման գործընթաց:

Աշխատանքի արդյունքում կստեղծվի ավտոմատացված ծրագրային միջավայր, որը կունենա ծանրաբեռնվածության կառավարման ենթա-կառուցվածք և կկիրառի GitOps-ի մոտեցումը՝ մեծ տվյալների մշակման գործընթացների արդյունավետությունը բարձրացնելու համար:

Հետազոտության նյութը: Մեծ տվյալների հասանելիության և կորստի վտանգների քննարկման շրջանակում ուսումնասիրվել են տվյալների հավաքագրման, գաղթի և շարունակական ինտեգրման գործընթացների կառավարման հիմնական խնդիրները: Աշխատանքի ընթացքում վերլուծվել են առկա ենթակառուցվածքները, կատարվել է հիմնավորված ընտրություն՝ ուղղված հուսալի և արդյունավետ համակարգի ստեղծմանը: Հետազոտվել են ավտոմատացման մեթոդները և գործիքակազմերը, որոնք ապահովում են. գործընթացների արագացում, կայունություն, կանխատեսելիություն, տվյալների անվտանգություն և մշտադիտարկում:

Ավտոմատացման շնորհիվ մեծ տվյալների գաղթը դառնում է ավելի արագ և հուսալի, նվազում են մարդու ներգրավվածությունն ու սխալների հավանականությունը: Օգտագործվել են գործիքներ, ինչպիսիք են Terraform [4], Kubernetes [5], և CI/CD պիլայնները, որոնք ապահովում են տվյալների գաղթի կայունությունն ու վերահսկելիությունը: Կոդով կառավարվող ենթակառուցվածքների (IaC) [6] մոտեցմամբ ենթակառուցվածքները կառուցվում և կառավարվում են կոդի միջոցով՝ օգտագործելով Terraform և Ansible[3]: Այս մոտեցման շնորհիվ հնարավոր է արագ իրականացնել փոփոխություններ, թարմացումներ կամ վերականգնումներ, հիմնել նույնական գործընթացներ տարբեր միջավայրերում՝ առանց հավելյալ ռեսուրսների: Տվյալների գաղթի անվտանգության ապահովման համար կիրառվել են անվտանգության մեխանիզմներ, որոնք երաշխավորում են՝ տվյալների ամբողջականություն, արտահոսքի ռիսկերի նվազեցում և գաղտնիության ապահովում: Մոնիտորինգի գործիքները՝ Prometheus և Grafana, ապահովում են գաղթի գործընթացի մշտադիտարկում և հետադարձ կապ՝ թույլ տալով կանխատեսել ու շտկել հնարավոր խնդիրները:

Հետազոտվել են հետևյալ գործիքները.

- Կառավարման գործիքներ՝ Git, CI/CD պիլայններ (Jenkins [7], GitLab CI/CD, GitHub Actions):
- Կոնտեյներացման գործիքներ՝ Docker [8], Kubernetes:
- IaC գործիքներ՝ Terraform, Vagrant, AWS CloudFormation:
- Ամպային հարթակներ՝ AWS, Google Cloud, Azure:

GitOps-ը, որն ի հայտ է եկել DevOps-ի գաղափարների հիման վրա, կենտրոնանում է IaC կառավարման և ենթակառուցվածքների փոփոխությունների

վերահսկման վրա: GitOps-ը նպաստում է ենթակառուցվածքների կառավարման ավտոմատացմանը, տվյալների անվտանգ տեղափոխմանը, Kubernetes-ի և այլ CI/CD խողովակաշարերի արդյունավետ ինտեգրմանը, կոնտեյներացմանը և կլաստերավորմանը: Կոնտեյներացման (Docker) և կլաստերավորման (Kubernetes) մեթոդներն ապահովում են ծրագրային ապահովման շարժունություն և հուսալիություն տարբեր միջավայրերում՝ տեղակայման, մասշտաբավորման և կառավարման պարզեցմամբ:

Հետազոտվել են CI/CD գործիքներ, ներառյալ Jenkins, Argo CD, և Flux CD: Ընտրվել է Argo CD՝ շնորհիվ իր գրաֆիկական ինտերֆեյսի և թողարկումների կառավարման լայն հնարավորությունների: GitHub Actions-ը ընտրվել է որպես CI գործիք՝ GitHub հոսթինգի ինտեգրման համար: Համեմատվել են գործիքների հատկությունները՝ ներառյալ ծրագրային լեզուների աջակցությունը, ենթակառուցվածքային հնարավորությունները, մոնիտորինգի ինտեգրումը և CI/CD համակարգերի աջակցությունը: GitOps և IaC մեթոդաբանությունների կիրառումը հիմնավորվել է, իսկ ընտրված գործիքակազմը նպաստում է մեծ տվյալների գաղթի և կառավարման գործընթացների ավտոմատացմանն ու արդյունավետության բարձրացմանը՝ միաժամանակ ապահովելով բարձր հուսալիություն:

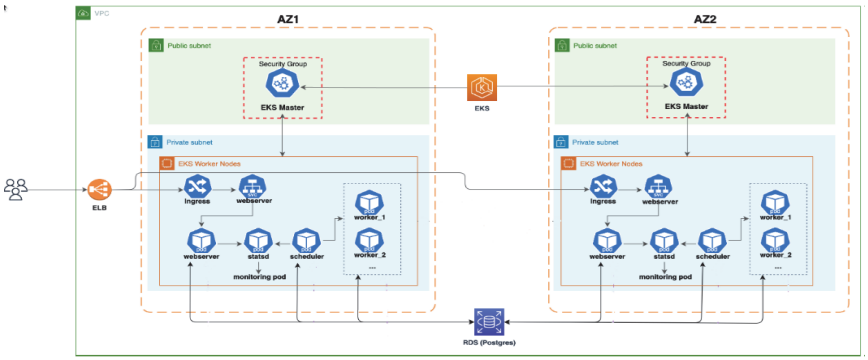
Աշխատանքի կատարման կարգը: Տվյալների գաղթի գործընթացը կազմակերպված է հետևյալ քայլերով:

Նախնական փուլում ստեղծվել է կառավարելի ենթակառուցվածք՝ ապահովելու տվյալների անվտանգությունը, մուտքի իրավունքը և պահուստավորման պլանի մշակումը: Այս փուլում կարևոր է պատրաստել բոլոր անհրաժեշտ պայմանները՝ տվյալների գաղթի համար, ինչպես նաև ապահովել նախնական անվտանգություն: Տվյալների ստուգման և պատշաճեցման փուլում կատարվում է տվյալների ստուգում՝ ի հայտ եկող սխալների և համատեղելիության խնդիրների լուծման համար [9]: Ստուգվում են տվյալների ամբողջականությունը, ճիշտ դասակարգումը և պատրաստ լինելը նոր ենթակառուցվածքի հետ համատեղելի աշխատանքին: Գաղթը իրականացվում է ավտոմատացված եղանակով՝ օգտագործելով գործիքներ, ինչպիսիք են Terraform, Kubernetes կամ տվյալների միգրացիայի այլ գործիքներ: Սա ապահովում է գաղթի արագությունը և ճշգրտությունը՝ նվազեցնելով մարդու միջամտության կարիքը և խափանումների ռիսկը: Գաղթի ընթացքում մշտահսկումն ապահովում է ցանկացած խնդիրների վաղ հայտնաբերման հնարավորությունը, որպեսզի արագ լուծումներ առաջարկվեն: Սա թույլ է տալիս ավելի արդյունավետ վերահսկել գործընթացը և նախօրոք կանխատեսել հնարավոր տեխնիկական խոչընդոտները: Գաղթի ավարտական փուլում կատարվում է տվյալների վերջնական ստուգում՝ համոզվելու, որ տվյալ-

ները հաջողությամբ տեղափոխվել են: Սա նշանակում է, որ տվյալները հասանելի և ամբողջական են: Հաջող գաղթի արդյունքում կազմվում է հաշվետվություն՝ ներկայացնելով արդյունքները: Տվյալների հասանելիության ապահովման համար ներկայացված ենթակառուցվածքը պետք է ապահովի տվյալների հասանելիությունը՝ առանց խափանումների: Դրա համար անհրաժեշտ է կիրառել տվյալների բաշխված պահպանում և արդյունավետ ուղղորդում, ինչը նվազեցնում է կորստի ռիսկերը և բարձրացնում համակարգի հուսալիությունը: Ընտրված ենթակառուցվածքում օգտագործվել են 2 հասանելիության գոտի (Availability Zones)՝ ապահովելու համակարգի հասանելիություն՝ անկախ ֆորս մաժորային իրավիճակներից, օրինակ՝ երկրաշարժ կամ ջրեղեղ: Սերվերները տեղակայվել են մասնավոր ցանցերում (VPS)՝ անվտանգության մակարդակը բարձրացնելու համար: Դիմումները, որոնք ուղղվում են ծառայություններ, նախապես բաշխվում են ծանրաբեռնվածության կառավարման (Load Balancer) միջոցով՝ ապահովելով բեռի հավասար բաշխում սերվերների միջև: Դիմումների կառավարումը բարդ գործընթաց է: Օգտատերերը դիմում են Load Balancer-ին: Load Balancer-ը բաշխում է դիմումը տարբեր սերվերների միջև՝ ելնելով դրանց բեռնվածությունից: Այս մեթոդն ապահովում է կայուն ծառայություն, որտեղ ամեն հայց մնում է տեղակայված համապատասխան սերվերի վրա՝ բաշխված և կարգավորված, ապահովելով համակարգի կայունությունն ու արդյունավետությունը:

Աշխատանքի կատարման համար AWS ամպային սերվերում ստեղծվել է հետևյալ ենթակառուցվածքը (նկ. 1):

Նկ. 1-ում նկարագրվում է AWS-ում EKS (Elastic Kubernetes Service)-ի վրա հիմնված ենթակառուցվածքը:



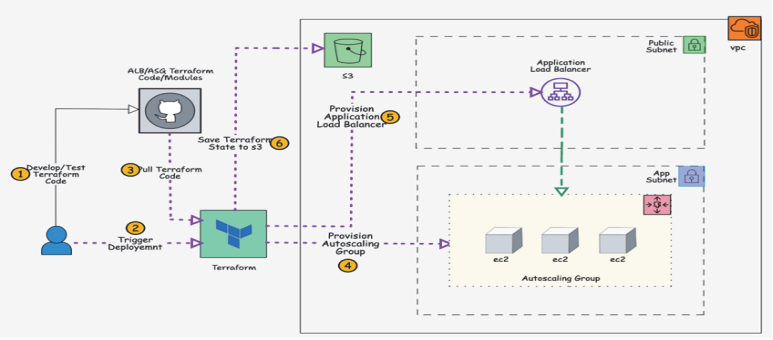
Նկ. 1. AWS-ում EKS ենթակառուցվածքը

Նկ. 1-ում ներկայացված AWS-ի վրա հիմնված Elastic Kubernetes Service (EKS) ենթակառուցվածքն ապահովում է բարձր հասանելիություն և արդյունավետ

բաշխում օգտատերերի հարցումների համար: Սա ընդգրկում է EKS Master Node-ը, worker node-ները, Elastic Load Balancer (ELB)-ը, PostgreSQL տվյալների շտեմարանը և այլ ծառայություններ, որոնք միացած են և համագործակցում են համատեղ՝ ապահովելու ծառայությունների կայուն աշխատանքը: Ենթակառուցվածքի բաղադրիչներն են EKS Master Node, որը տեղադրված է Public Subnet-ում՝ ապահովելով մուտքի իրավունք և ենթակառուցվածքի կառավարում: Այն պատասխանատու է API հարցումների ընդունման և worker node-ների կառավարման համար: Master Node-ը հետևում է կլաստերի առողջությանը և աջակցում կլաստերի գործունեությանը: ELB (Elastic Load Balancer) հավասարակշռման համար է, և տվյալները մուտք են գործում համակարգ՝ միջնորդավորված ELB-ի միջոցով: ELB-ն բաշխում է հարցումները EKS Master Node-ին՝ ապահովելով բարձր հասանելիություն և բեռի հավասար բաշխում: Հետևողականության և բարձր մատչելիության ապահովման համար ենթակառուցվածքը բաշխվել է AZ1 և AZ2-ի միջև: AZ1 և AZ2-ում են տեղակայված EKS Worker Node-ները, որոնք աշխատում են ամբողջ համակարգը կարգավորելու և բեռը հավասար բաշխելու համար: EKS Worker Node-ները տեղադրված են Private Subnet-ներում, որտեղ իրականացվել է հիմնական աշխատանքը: Ingress-ը բաշխել է արտաքին հարցումները դեպի համապատասխան պոդեր: Webserver Pods-երը մշակել են օգտատերերի հարցումները: Monitoring Pods-երը վերահսկել են համակարգի անվնասությունը, այդ թվում՝ գործունակությունը և ռեսուրսների սպառումը:

RDS (Relational Database Service)-ը ապահովել է տվյալների պահպանման և մշակման ապահովումը: Բոլոր worker node-ները միացվել են RDS PostgreSQL տվյալների շտեմարանին՝ երաշխավորելով տվյալների ամբողջականությունը և անվտանգության պահպանությունը:

Ենթակառուցվածքն աշխատում է Terraform-ի միջոցով, որը պատկերված է նկ. 2-ում:

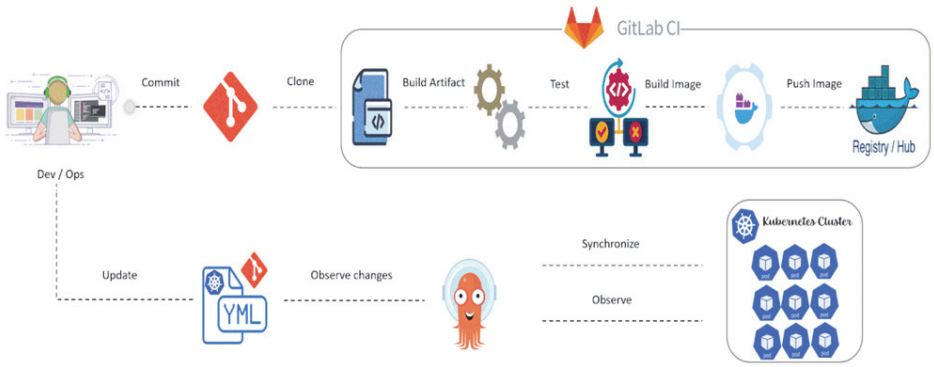


Նկ. 2. Ենթակառուցվածքի աշխատանքը

Նկարում ներկայացված է AWS (Amazon Web Services) ենթակառուցվածքն աշխատանքի ընթացքում, երբ օգտագործվում է Terraform-ը՝ ենթակառուցվածքը որպես կոդ (IaC) կառավարելու համար: Գործընթացը բաղկացած է 6 հիմնական քայլերից: Նախ ծրագրավորողը գրում կամ թարմացնում է Terraform-ի կոդը, որը պարունակում է AWS ռեսուրսների (օրինակ՝ EC2, S3, Load Balancer) կոնֆիգուրացիաները: Սա նախապատրաստում է ենթակառուցվածքը՝ համապատասխան AWS ռեսուրսների ստեղծման կամ թարմացման համար: Deploy գործընթացի մեկնարկը, երբ պատրաստ է գործարկելու ենթակառուցվածքը, կատարվում է "Trigger Deployment" գործողությունը: Սա սկսում է Terraform-ի պրոցեսը, որը կարդում է կոնֆիգուրացիաները և կատարում անհրաժեշտ փոփոխություններ՝ ռեսուրսներ ստեղծելու կամ թարմացնելու համար: Պլանավորման և իրականացման փուլում Terraform-ը կատարում է նախնական պլանավորում՝ ստուգելով, թե որոնք են ռեսուրսները, որոնք պետք է ստեղծվեն կամ փոփոխվեն՝ համեմատելով իրական ռեսուրսները կոնֆիգուրացիայի հետ: Այնուհետև սկսվում է իրական «կիրառումը», որի դեպքում ռեսուրսները ստեղծվում կամ թարմացվում են: AWS S3-ում state file-ի պահուստավորման քայլը: Այս 4-րդ քայլը կապված է AWS S3-ի հետ, որտեղ պահվում է Terraform-ի "state file"-ը (տվյալները, որոնք հետևում են գոյություն ունեցող ռեսուրսներին): Այս ֆայլը կարևոր է՝ փոփոխություններին ճիշտ հետևելու համար: Terraform-ի միջոցով ստեղծվում է Application Load Balancer (ALB), որը պատասխանատու է օգտատերերի հարցումների ընդունման և դրանց բաշխման համար տարբեր սերվերների միջև՝ ապահովելով բեռնվածության հավասարակշռումը: Auto Scaling Group Provisioning-ը վերջին՝ 6-րդ քայլն է: Auto Scaling Group-ը Provision է (մատակարարվում)՝ ALB-ի հետևում: Auto Scaling Group-ը ներառում է EC2 instance-ներ (սերվերներ): Այն ավտոմատ կերպով ավելացնում կամ նվազեցնում է instance-ների քանակը՝ կախված համակարգի ծանրաբեռնվածությունից: Այս պրոցեսն ապահովում է բաշխված ենթակառուցվածքի արդյունավետ և կայուն կառավարումը՝ նվազեցնելով մարդկային միջամտությունը և ապահովելով բեռնման ճկունություն:

EC2 Instance-ներ EC2 սերվերները գտնվում են Auto Scaling Group-ի մեջ և ծառայում են որպես բազային ենթակառուցվածք՝ օգտատերերի հարցումները մշակելու համար: Load Balancer-ը բաշխում է տրաֆիկը այս instance-ների միջև: Օգտատերերը իրենց հարցումներն ուղարկում են Load Balancer-ի միջոցով: Load Balancer-ը ապահովում է բարձր մատչելիություն, իսկ Auto Scaling Group-ը՝ առավելագույն կատարողականություն:

Նկ. 3-ում նկարագրված է տվյալների հոսքը, որն ընդգրկում է GitLab CI/CD համակարգը, կոդի կառուցումը, թեստավորումը և տեղակայումը Kubernetes կլաստերներում:



Նկ. 3. Մեծ տվյալների գաղթի գործընթացի կառավարման ենթակառուցվածքը

Տվյալները ստեղծվում կամ ստացվում են խելացի ներդրված համակարգերից: Կոդը գրվում և կատարվում է Git-ի commit: Կոդը պահվում է GitHub, GitLab կամ այլ version control համակարգերում, վերջինը գործում է որպես հիմնական տվյալների աղբյուրի կառավարման կետ: Կոդը կրկնօրինակում է և անցնում CI/CD համակարգի միջոցով:

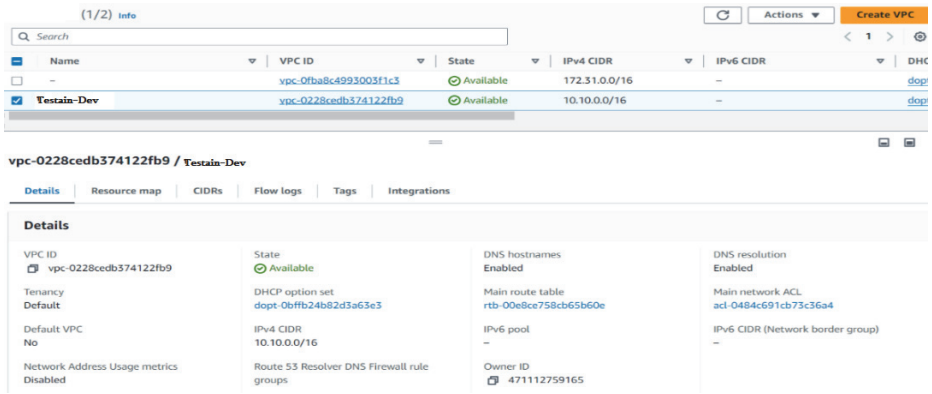
GitLab CI/CD Pipeline-ը կոդի համար կատարվել են մի շարք ավտոմատ քայլեր. կառուցվել են անհրաժեշտ արտեֆակտները, և իրականացվել ավտոմատ թեստեր, որոնք ստուգել են կոդի ճշգրտությունը: Կոդից ստեղծվել է Docker պատկեր (image), որը պատրաստ է գործարկման համար: Պատկերը հպվել է Docker Registry կամ Hub-ին, որտեղ այն հետագայում տեղակայվելու է:

YAML ֆայլերը պարունակում են անհրաժեշտ պարամետրեր՝ պոդերի, ծառայությունների և այլ ռեսուրսների կոնֆիգուրացիայի համար: YAML թարմացումները կատարվում են Git-ի միջոցով: Երբ YAML ֆայլերը թարմացվում են, օգտագործվում է գործիք (օրինակ՝ ArgoCD կամ Flux)՝ փոփոխությունների դիտարկման համար: Այս գործիքը հայտնաբերում է փոփոխությունները (observe changes) և համաժամեցնում (synchronize) YAML ֆայլերը Kubernetes-ի հետ: ArgoCD կամ այլ գործիքի միջոցով YAML ֆայլերը օգտագործվում են «կիբեռնետ» կլաստերում՝ ռեսուրսներ տեղակայելու համար:

Տեղակայումը կատարվել է մի քանի կլաստերներում՝ ապահովելով ծառայությունների բարձր մատչելիություն և մասշտաբավորում: Գործընթացի արդյունքում կոդի յուրաքանչյուր թարմացում ինքնաբերաբար թեստավորվում, կառուցվում և տեղակայվում է «կիբեռնետ» -ում: Այս գործընթացն ապահովում է արագություն, հուսալիություն և ավտոմատացում:

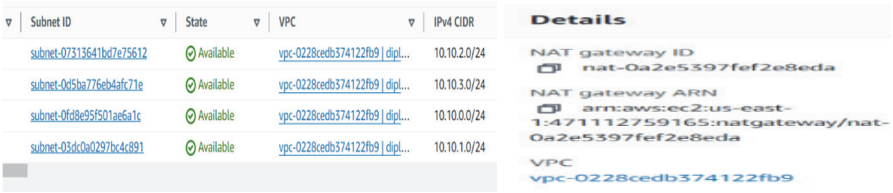
Աշխատանքի իրականացման ցանցային ընթացակարգը: Աշխատանքի համար ստեղծվել է ցանցային կառուցվածք մասնավոր ամպում հիբրիդային

սկզբունքով: GitOps-ի ենթակառուցվածքների կառավարման գործընթացի իրականացումը ներկայացված է նկ. 4-ում:



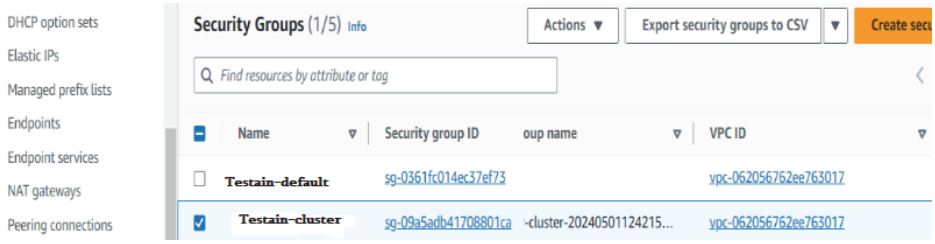
Նկ. 4. Ցանցի կառուցվածքը

Ստեղծվել են Subnet-ները և NAT (նկ. 5.)



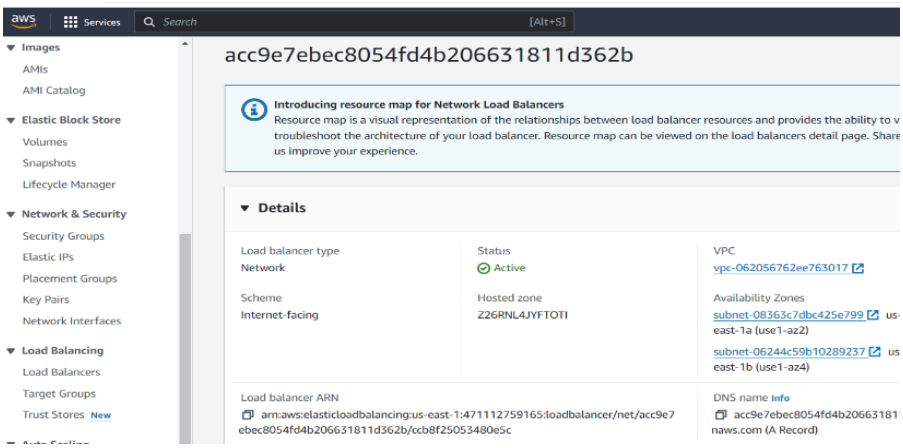
Նկ. 5. Subnet-ները և NAT-ը

Կատարվել են անհրաժեշտ կարգավորումները, և «կիբեռնետ»-ի հանգույցներին բաշխվել են մասնավոր IP հասցեները՝ ապահովելով անվտանգ և վերահսկվող ցանցային միջավայր: Բացի այդ, ստեղծվել է EKS (Elastic Kubernetes Service), որն ապահովում է «կիբեռնետ» ենթակառուցվածքի ավտոմատ կառավարումը՝ թույլ տալով հեշտացնել բեռնվածության կառավարման և մասշտաբավորման գործընթացները: Այս լուծումը նաև նպաստում է բաշխված համակարգերի հուսալիության և արդյունավետության բարձրացմանը (նկ. 6):



Նկ. 6. EKS-ը

Ստեղծվել է «պատկեր» ֆայլերի պահպանման և պահեստավորման տեղ: Ստեղծվել է ծանրաբեռնվածության հավասարակշռիչ (Load Balancer), որը գտնվում է ամպի հանրային ցանցում, որի նպատակն է ապահովել հասանելիություն մասնավոր ցանցում գտնվող սարքավորումներին: Խելացի մաշտաբավորում ապահովելու համար ստեղծվել է կարպենտեր (նկ.7), որը, դիտարկելով առկա ծանրաբեռնվածութունը, կառավարում է այն՝ ավելացնելով կամ պակասեցնելով համապատասխան հզորությամբ սերվերներ:



Նկ. 7. Ծանրաբեռնվածության հավասարակշռիչի սրեղծումը

Ենթակառուցվածքների կառավարման գործընթացի ավտոմատացումն իրականացվել է կոնտեյներացմամբ: Ծրագրի իրականացման համար անհրաժեշտ է միջավայր, որտեղ առկա են apache, mysql database և php:

Ինչպես նշվեց վերը, կոդից ստեղծվում է Docker պատկերը, որը պատրաստ է գործարկման: Պատկերը հպվում է Docker Registry կամ Hub-ին, որտեղ այն կարելի է հետագայում տեղակայել:

YAML ֆայլերը պարունակում են անհրաժեշտ պարամետրերը՝ պոդերի, ծառայությունների և այլ ռեսուրսների կոնֆիգուրացիայի համար (նկ. 8):

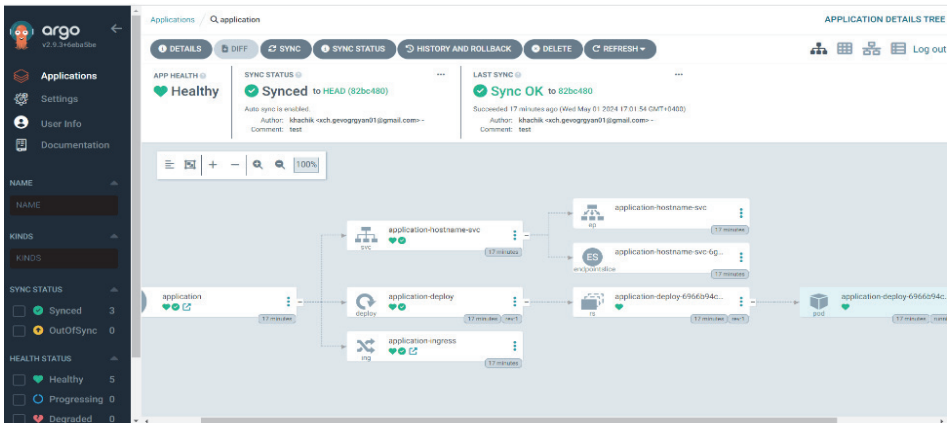
Նկ. 8. Yaml փեղակայումը

Կտորը, համապատասխան վայրում փոփոխության դեպքում, նոր պատկեր է ստեղծում և ուղարկում պահոց: Նկ. 9-ում ներկայացված է հատված AWS-ի վրա հասանելիության անվտանգության ապահովման գործընթացից:

```
5           - main
6       jobs:
7         my_deploy:
8           runs-on: ubuntu-latest
9           steps:
10            - name: Checkout
11              uses: actions/checkout@v3
12
13            - name: Configure AWS credentials
14              uses: aws-actions/configure-aws-credentials@v2
15              with:
16                aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_II
17                aws-secret-access-key:
18  secrets.AWS_SECRET_ACCESS_KEY }}
19                aws-region: us-east-1
20            - name: Bump version
21              id: vars
22              uses: remorses/bump-version@js
```

Նկ.9. Հատված կոդից

Պատկերը պահոց տեղափոխելուց հետո անմիջապես ուղարկվել է դեպի «կիբեռնետ» միջավայր: Նկ.10-ում ներկայացված է Argo CD UI-ի տեսքը՝ պատկերը «կիբեռնետում» տեղակայելուց հետո:



Նկ. 10. Արգո-ի տեսքը

Եզրակացություն: Աշխատանքում կիրառված շարունակական ինտեգրման/շարունակական առաքման (CI/CD) գործիքակազմը հնարավորություն է տվել՝ ավտոմատացնելու մեծ տվյալների մշակման գործընթացների առաքումը՝ ապահովելով կայունություն, արագություն և արդյունավետություն:

Աշխատանքի արդյունքում ստեղծված ավտոմատացված ենթակառուցվածքը կոդի միջոցով կառավարում է ենթակառուցվածքը՝ ապահովելով հետևողականություն, հեշտ կրկնօրինակում և արդյունավետ թարմացում: Կոնտեյներացումը և «կիբեռնետ»-ը ապահովում են բեռնվածության հավասարակշռում, ավտոմատ մասշտաբավորում և ծառայությունների կայունություն:

GitOps և IaC մեթոդաբանությունների կիրառումը բարձրացնում է տվյալների մշակման արդյունավետությունը՝ ապահովելով հուսալիություն, անվտանգություն և օպտիմալ կատարողականություն: Ստեղծված լիովին ավտոմատացված միջավայրը հնարավորություն է տալիս՝ կատարելու փոփոխություններ ծրագրում և ավտոմատ կերպով ստեղծելու նոր, աշխատող միջավայր՝ առանց մարդու միջամտության: Նախագծված ծանրաբեռնվածության բաշխման լուծումները և նոր ցանցային ճարտարապետությունը նպաստել են տվյալների գերբեռնվածության արդյունավետ կառավարմանը և սերվերային ռեսուրսների օպտիմալ օգտագործմանը: Սերվերի ընտրությունը, հաշվի առնելով բեռնվածության պահանջները և արհեստական գերբեռնվածությունը, թույլ են տվել՝ գնահատելու համակարգի կայունությունը և ռեսուրսների բաշխումը:

Աշխատանքը ուղղված է տվյալների մշակման գործընթացների օպտիմալացմանը, ենթակառուցվածքների արդյունավետության բարձրացմանը և ժամանակակից տեխնոլոգիական մարտահրավերներին դիմակայելու կարողության ամրապնդմանը, ինչը համահունչ է կազմակերպությունների աճին և բիզնեսի պահանջներին:

ԳՐԱԿԱՆՈՒԹՅԱՆ ՑԱՆԿ

1. https://git-scm.com/
2. https://about.gitlab.com/topics/gitops/
3. https://www.ansible.com/
4. https://www.terraform.io/
5. https://kubernetes.io/ru/docs/concepts/overview/what-is-kubernetes/
6. https://aws.amazon.com/ru/what-is/iac/
7. https://www.jenkins.io
8. https://www.docker.com/
9. https://cassandra.apache.org/_/index.html.

Г.О. САРГСЯН

АВТОМАТИЗАЦИЯ ИНФРАСТРУКТУРЫ УПРАВЛЕНИЯ ПРОЦЕССОМ ПЕРЕДАЧИ БОЛЬШИХ ДАННЫХ

Цель работы – разработка автоматизированной инфраструктуры, управляющей ИТ-ресурсами с использованием подхода *Infrastructure as Code* (IaC), а также исследование ключевых проблем, связанных с управлением и миграцией больших данных в процессе непрерывной интеграции и доставки (CI/CD). Это обеспечивает их отслеживаемость, упрощает копирование и обновление, а также делает процесс более эффективным. Автоматизация снижает вероятность человеческих ошибок, гарантируя быстрое и надежное внесение изменений в инфраструктуру. Контейнеризация и использование Kubernetes обеспечивают балансировку нагрузки, автоматическое масштабирование и стабильность сервисов. Эти технологии также упрощают управление и масштабирование услуг в распределенных средах.

Результатом работы стало повышение качества предоставляемых услуг и сокращение времени на обслуживание, что позволяет создавать надежные, масштабируемые и эффективные системы, соответствующие современным технологическим требованиям. Внедрение процессов сбора данных, автоматизации и управления не только упрощает управление сложной инфраструктурой, но и открывает новые возможности в технологическом секторе.

Ключевые слова: сервер, инфраструктура, миграция, система балансировки нагрузки, онлайн-технологии, управление, CI/CD.

G.H. SARGSYAN

AUTOMATION OF THE INFRASTRUCTURE FOR CONTROLEING THE BIG DATA TRANSFER PROCESS

The aim of this work is to explore the main challenges associated with big data migration and the continuous integration process. As a result, an automated infrastructure was developed that manages resources through code. This approach ensures traceability, simplifies replication and updates, and increases overall efficiency. Automation reduces the likelihood of human error, enabling fast and reliable infrastructure changes.

Containerization and the use of Kubernetes provide load balancing, automatic scaling, and service stability. These technologies also streamline the management and scaling of services in distributed environments.

The outcome of this work includes an improvement in service quality and a reduction in maintenance time, enabling the creation of reliable, scalable, and efficient systems that meet modern technological demands.

The implementation of data collection, automation, and infrastructure management processes not only simplifies the handling of complex systems but also opens up new opportunities in the technology sector.

Keywords: server, infrastructure, migration, load balancing system, online technologies, management, CI/CD.