Қайырлы күн, құрметті студенттер! Сіздерді бүгінгі дәрісте көргеніме қуаныштымын. Біз алдыңғы өткен тақырыпты жалғастырамыз, нақты айтар болсақ, бүгін Compute тақырыбының екінші бөлігі. Сонымен, бастайық.

Төртінші бөлімде контейнерлік сервистер, яғни Container services туралы сөйлесеміз. Әрі қарай бесінші бөлімде AWS Lambda сервисін талдаймыз. Ал соңғы, алтыншы бөлімде, AWS Elastic Beanstalk туралы айтатын боламыз.

Төртінші бөлім – контейнерлермен байланысты сервистер. Контейнерлер туралы сөз қозғамас бұрын, есімізге түсіріп алайық. Бізде физикалық сервер бар. AWS дата-центрінің ішінде физикалық тұрғыдан белгілі бір аумақта ең қуатты физикалық серверлер орналасқан. Содан кейін, виртуалдандырудың арқасында бұл физикалық виртуалды сервер бір-біріне тәуелсіз Amazon EC2 виртуалды инстанстарына бөлінеді. Осылайша, Amazon EC2 аясында процессор ядроларының саны мен жедел жадының мөлшері өзгеше 50-ден астам әртүрлі түрлерді сұрай аламыз.

Енді одан әрі қарай жүрсек, операциялық жүйе деңгейінде виртуалдандырудың тағы бір нұсқасы бар. Онда біз операциялық жүйені орнатып, операциялық жүйенің үстіне оқшауланған контейнерді іске қосамыз, яғни Amazon EC2 инстанстарына байланысты айтатын болсақ, ол жерде сіз құру кезінде белгілі бір операциялық жүйені көрсетесіз. Ал контейнерге байланысты айтқан кезде, контейнер белгілі бір операциялық жүйеге байланады және сол операциялық жүйенің аясында іске қосылады. Бірақ қалғанының бәрі, атап айтқанда кітапханалар, барлық конфигурациялар, код, run time, сіздің бағдарламаңыздың іске қосылу ортасы - барлығы оқшау болады және өзін-өзі қамтамасыз етеді. Осылайша, біз контейнерді repeatable деп айтамыз, яғни ол оңай көшіріледі, сіз дайын контейнерді алып, оны көшіріп, оны басқа жерде, басқа environment-те іске қосасыз және ол дәл осылай жұмыс істейді, өйткені ол сізге қажет нәрсенің бәрін қамтиды. Тағы бір маңызды мәселе - виртуалды машиналар контейнерлерге қарағанда әлдеқайда баяу іске қосылады.

Әрі қарай жалғастырайық. Контейнерлерді құру және олармен жұмыс істеу үшін бізге нақты бағдарлама қажет. Контейнерлерді жасауға мүмкіндік беретін ең танымал бағдарлама – Docker, яғни Docker-контейнер. Docker-контейнер – бұл сіздің бағдарламаңызды іске қосу және дұрыс жұмыс істеуі үшін қажет нәрсенің бәрін қамтитын ұйым, атап айтқанда кітапханалар, жүйелік құралдар, код және сіздің бағдарламаңыз үшін іске қосу ортасы.

Біз контейнердің қандай екендігін қарапайым сөзбен айттық. Енді сәл тереңірек талдайық. Виртуалды машиналар жайлы айтқан кезде, оң жақта сіз әртүрлі кітапханалармен жұмыс істейтін үш түрлі қосымша deployment-інің мысалын көресіз. Олар әр түрлі түстермен арнайы ерекшеленеді, яғни Application 1, Application 2, Application 3. Олардың әрқайсысы Amazon EC2 инстансында іске қосылған. Әрі олардың бір-бірінен оқшауланған операциялық жүйесін көре аласыз, яғни VM1, Virtual Machine 1, басқаларға ұқсас не мүлде ұқсамайтын өзінің операциялық жүйесі болады. Бұл қосымшаның өзіне байланысты, яғни виртуалды машина hypervisor-дің үстінен жұмыс істейді.

Енді виртуалдандырудың келесі деңгейі – hypervisor-ден жоғары тұратын контейнерлер, яғни Amazon EC2 инстанстарының операциялық жүйесі. Сонымен, бір Amazon EC2 инстансы аясында, экранның сол жағында, сіз үш түрлі контейнерді іске қосқанымызды көре аласыз. Әр контейнерде тек қосымша ғана емес, сонымен қатар әр қосымшаның дұрыс жұмыс істеуі үшін қажет барлық нәрсе бар. Сонымен бірге, сіз қосымшалар бір-бірімен байланысты емес әр түрлі кітапханалармен жұмыс істейтінін көре аласыз. Ең бастысы – дұрыс операциялық жүйенің болуы. Бұл жерде контейнерлер әртүрлі операциялық жүйелерде жұмыс істеуге жеткілікті түрде икемді екенін атап өткен жөн. Негізгі талап – барлық қажетті мүмкіндіктер және белгілі бір контейнер үшін қажет функционалдылықтың барлығы жүзеге асырылуы керек. Бізде Linux операциялық жүйесінің әртүрлі нұсқалары бар. Біздің контейнерге Linux операциялық жүйесінің екі түрлі нұсқасы бар функционалдылық қажет болған жағдайда, біз бұл контейнерді операциялық жүйенің екі нұсқасында да сәтті іске қосуға болады деп айтамыз. Осылайша, сіз контейнерді басқа операциялық жүйедегі кез-келген басқа компьютерде іске қоса аласыз. Бұл контейнерлермен жұмыс істеу кезінде біршама қолайлы болмақ.

Тағы бір мәселе, біз бір EC2 инстансы аясында өте кішкентай контейнерлерді іске қоса аламыз, ал бір EC2 инстансында бір уақытта жүздеген түрлі контейнерлер бола алады. Әр контейнер белгілі бір қосымшаға қызмет етеді.

Енді AWS-ке қарай жүрейік, сіз контейнерлермен жұмыс істеу үшін Amazon EC2 инстансын іске қосып, docker қосымшаларын орнатып, контейнерлермен жұмыс істей аласыз деп жоспарлаған боларсыз. Иә, бұл бұлттағы контейнерлермен жұмыс істеудің дұрыс әдісі, бірақ ең тиімдісі емес. Ең жақсы нұсқа – арнайы ойлап шығарылған сервис Amazon Elastic Container Service, қысқаша ECS деп жиі кездеседі, контейнерлермен ыңғайлы түрде жұмыс істеуге мүмкіндік береді. Осылайша, сіз осы сервисті пайдаланып бірнеше секунд ішінде 10 мыңға дейін контейнерлерді іске қоса аласыз. Сондай-ақ, контейнерлерді іске қосу және басқару үшін әртүрлі әрекеттер мен кестелерді ыңғайлы түрде бақылауға, басқаруға, тіпті теңшеуге болады. Мұнда ECS тек On-demand EC2 инстанстарын ғана емес, сонымен қатар Spot Instances пен Reserved Instances-ті де қолдайтынын атап өткен жөн.

Бір мысалды қарастырайық. Бізде task definition бар. Тask definition – бұл сіздің контейнеріңіздің сипаттамасы. Онда сіздің қосымшаңыз туралы ақпарат бар, қандай порттарды қолданасыз және қосымшаның жұмыс істеуі үшін қосымша параметрлерді орнатуыңыз мүмкін. Осы мысалда біз task definition жасадық деп елестетіп көрейік, екі контейнер Container A және Container B сипатталған. Енді біздің инфрақұрылымның бөлігі ретінде бізге үш Container A инстанстары және екі Container B инстанстары қажет. Осылайша біз task немесе контейнер инстансын істеп жатырмыз. Сонымен біз контейнерлеріміз бар кішкентай инстанстарды task арқылы Amazon ECS сервисіне жібереміз. Біз үшін бұл ECS сервисі қосымша кіріс деректерімізге байланысты осы контейнерлерді орналастырады және оларды біздің ECS кластерімізде іске қосады. Cluster ECS – бұл топ ретінде іске қосылатын Amazon EC2 инстанстарының жиынтығы. Осылайша, Amazon ECS әрбір Amazon EC2 инстансында агент орнатады. Бұл агент контейнерлеріңізді осы Amazon EC2 инстанстарына орналастыруға көмектеседі.

ECS кластері туралы айтқан кезде бізге үш нұсқа қолжетімді. Бірінші нұсқа – біз контейнерлерімізді жасап, осы контейнердің қаншалықты қуатты болуы керектігін, осы контейнерлердің қаншасын құратынымызды, сондай-ақ желіні конфигурациялау үшін қажетті параметрлерді және осы контейнерлердің бір-бірімен өзара әрекеттесуін сипаттайтын кезде қолданылатын Networking Related Settings. Бұл жағдайда біз оң жақтағы кейс туралы айтып отырмыз және Amazon ECS және AWS Fargate сервистерінің комбинациясының арқасында біз тек контейнерлерімізге назар аудара аламыз. Қалғанының бәрі, яғни операциялық жүйені басқару, , docker-агентін басқару және біздің контейнерлер іске қосылып жұмыс істейтін docker-қосымшасы AWS-ке ауысады. Бұл – сізде ресурстар не мамандар болмаса немесе сіздің контейнерлеріңізді басқаратын Amazon EC2 инстанстарын дәл баптауды қажет етпейтін болғандағы жағдай.

Ал егер сізге кеңейтілген баптау қажет болса, алайда ол сізге әрдайым қажет емес. Сонда да бұл жағдайда сіз сол жақтағы басқа опцияны қолдана аласыз, онда сіз ECS кластеріндегі инстанстарды толық басқарып, сізге қажет барлық параметрлерді нақты көрсете аласыз. Бұл жағдайда екі нұсқаны қарастыра аламыз. Сіздің ECS кластеріңіз не Linux не Windows EC2 инстанстарынан тұратын болады. Кез келген жағдайда, бірінші опцияда жасалынған әрекеттерімізбен қоса, сізге бір Amazon EC2 инстансын жасау үшін қажет барлық параметрлерді енгізу қажет. Осы енгізу деректеріңіздің үлгісін пайдаланып, ECS кластері жасалады.

Келесі кейсті қарастырайық. Сіз – қосымшаны әзірлейтін және оны жасау үшін docker контейнерлерін қолданатын компаниясыз. Компания өсіп келеді, біраз уақыттан кейін сізде бар контейнерлер саны артады. Сізге тиімді оркестрлеуге көмектесетін, яғни контейнерлердің осы үлкен көлемін басқаратын нәрсе керек. Осы мақсатта container orchestration-ға арнайы жасалған open sourse software бар, ол Kubernetes деп аталады, сіз оны K8S ретінде жиі кездестіресіз. Kubernetes дегеніміз не екенін түсіну үшін, біз docker-ге байланысты бір қонақ операциялық жүйеде жұмыс істейміз деп айтамыз. Kubernetes-ке байланысты айтқанымызда, біз жоғарырақ деңгейге көтеріліп, бірнеше Docker host-тарымен жұмыс істей аламыз. Kubernetes бізге масштабтау, container provisioning, яғни контейнерлерді іске қосу, networking мәселелері мен жүктемені теңдестіру міндеттерін жеңілдетуге мүмкіндік береді, бұл бізге осы көптеген түрлі контейнерлермен тиімді жұмыс істеуге көмектесетін шешім.

Kubernetes туралы айтылғанда кейбір танымал терминдерді айта кеткен жөн. Kubernetes кластерді басқарады. Cluster – бұл бірнеше виртуалды машиналардың жиынтығы. Кластердегі әрбір виртуалды машина нода деп аталады. Бір виртуалды машинада бірнеше жүз контейнер іске қосыла алатыны есімізде. Сонымен, біздің жағдайда, Kubernetes жағдайында, подтар контейнер ретінде әрекет етеді. Осылайша, біздің кластеріміз нодалардан тұрады және әр нодада көптеген подтар, яғни контейнерлер іске қосыла алады.

Әрі қарай жылжыйық. Енді келесі сервис туралы сөйлесейік. Сіз дәл қазір Kubernetes кластерін келесі жолмен баптай аласыз деп ойлаған шығарсыз: бірнеше Amazon EC2 инстанстарын іске қосып, docker-қосымшасын орнатамын, оның үстіне Kubernetes қосымшасын орнатсам менде жұмыс істейтін Kubernetes-Cluster пайда болады. Иә, рас, бұл да бір нұсқа, бірақ біз AWS-пен жұмыс істеген кезде бізде одан да жақсы нұсқа бар, атап айтқанда Amazon Elastic Kubernetes сервисі, оны сіз Amazon EKS ретінде жиі кездестіресіз. Бұл Managed Kubernetes Service ұсынатын сервис, бұл Kubernetes қосымшасымен толық үйлесімдегі сервис және сервис пайдаланушыларға, біздің AWS бұлтты провайдерінің клиенттеріне кластердің көп бөлігін AWS-ке жүктеуге мүмкіндік беріп, өзініңіз бизнес тапсырмаларыңызға көбірек көңіл бөлуге жағдай жасайды. Бұл сервистің Kubernetes қосымшасымен үйлесімді болуы жергілікті дата-центрлермен жұмыс істеп тұрған Kubernetes жұмыс жүктемелерін AWS-ке, әсіресе Amazon Elastic Kubernetes сервисіне оңай жібере алатынымызды білдіреді.

Сізде “Amazon ECS пен Amazon EKS арасындағы айырмашылық қандай?” деген сұрақ туындауы мүмкін. Шын мәнінде, екі нұсқа да, екі сервис те, біздің Docker-контейнер кластерін оркестрлеуге көмектеседі. Жалғыз айырмашылық – бірінші жағдайда біз Docker-контейнерлерімен тікелей жұмыс істейміз және кластерімізді AWS арқылы басқарамыз. Amazon EKS-те кластер енді AWS шешімімен емес, Kubernetes open sourсe бағдарламасымен реттеледі, бірақ ол бізді кейбір операциялық мәселелерден босататын сервиске оралған. Олар сіз орнатқан қосымашаға, инфрақұрылымыңыздың қалай бапталғанына, оның қалай жұмыс істейтініне байланысты немесе бұл нұсқаларды пайдаланбайтын болсаңыз, кем дегенде AWS тұтынушыларында осы опциялардың біреуін таңдау мүмкіндігі бар.

Мен сізге айтқым келетін келесі сервис – Amazon Elastic Container Registry, оны сіз Amazon ECR ретінде жиі кездестіресіз. Бұл сіздің барлық Docker-кескіндеріңізді сақтайтын сервис түрі. Осылайша, кластерді құру кезінде, мейлі Amazon ECS не Amazon EKS көмегімен болсын, сіз контейнерді, сол контейнердің кескінін көрсетуіңіз керек. Бұл жағдайда сіз Amazon ECR сервисін пайдаланасыз. Аналогия ретінде, Amazon EC2 инстансын жасаған кезде, біз болашақ инстансыңыздың бейнесі болып табылатын AMI көрсетеміз. Бұл жағдайда AMI Docker-кескіні ретінде әрекет етеді. Сіздің контейнерлерге байланысты барлық AMI-лар сақталынатын қоймаңыз – Amazon ECR болып табылады.

Осымен біз дәрісіміздің төртінші бөлімінің соңына келдік. Ең маңызды тармақтарды еске түсіріп өтейік. Контейнерлер – бұл сіздің қосымшаңызды сәтті іске қосу үшін қажет нәрсенің бәрін ұстай алатын қойма. Бұған кітапханалар, жүйелік параметрлер, код және т.б. жатады. Docker – бұл контейнерлерді құруға мүмкіндік беретін ең танымал бағдарламалардың бірі. Бір қосымша бір-бірімен байланысқан бірнеше контейнерлерде іске қосыла алады. Docker-контейнерлерін оркестрлеуге мүмкіндік беретін Amazon Elastic Container Service немесе қысқаша ECS деген сервис бар. Келесі танымал қосымша, бағдарлама – Kubernetes. Бұл сіздің контейнерлеріңізді оркестрлеуге мүмкіндік беретін open source шешім. Kubernetes үшін арнайы Amazon Elastic Kubernetes Service сервисі құрылды. Ол Kubernetes-пен үйлесімді және Docker-контейнер кластерін Kubernetes арқылы басқару кезінде операциялық жұмыстан қолыңызды босатуға мүмкіндік береді. Үшінші сервис – Amazon Elastic Container Registry. Бұл сіздің барлық Docker-контейнерлеріңізді сақтайтын сервис.

Енді дәрісіміздің бесінші бөліміне де жеттік және мұнда AWS Lambda сервисі туралы сөйлесеміз. Бұл менің барлық AWS сервистерінің ішіндегі ең сүйікті сервисім, неге екенін қазір білесіз. Біз бұған дейін бізде әртүрлі IT-ресурстарды ұсынатын түрлі сервистер бар екенін айтқан болатынбыз. Compute туралы айтқанда, Amazon EC2 сервисі бізге виртуалды машиналарды ұсынады дедік. Ал Amazon ECS және Amazon EKS сервистері бізге контейнерлермен жұмыс істеуге көмектеседі. Сонымен, келесі деңгей – AWS Lambda, Serverless Computing, бұл сіз үшін барлық операциялық тапсырмаларды жоятын сервис және сізге тек кодты енгізу ғана жеткілікті. Ал сервисті іске қосу, оны қолдау, мониторинг орнату және т.б. байланысты барлық нәрселерді AWS өзі өңдейді және қабылдайды. Осылайша, мен әзірлеуші ретінде белгілі бір шешім құру кезінде жүйелік әкімшінің қажеттілігін жоя аламын. Және де сіз, IT университеттері жүйелік әкімшілерге қарағанда көбірек әзірлеушілерді дайындайтындықтан, бұл сервисті өзіңіздің шешімдеріңізді, өзіңіздің стартап-жобаларыңызды іске қосу үшін пайдалана аласыз, мұнда AWS бір маманды, яғни жүйелік әкімшінің қажеттілігін жояды. Бұл сізге кейбір шығындарды қысқартуға және стартапты сәтті бастауға мүмкіндігін арттыруға көмектеседі.

Төлем жайлы тағы бір маңызды мәселе бар. Сіз AWS Lambda іске қосылған уақыт мөлшеріне ғана төлейсіз. Егер сіздің кодыңыз іске қосылмаса, сіз мүлдем ештеңе төлейсіз. Бұл сонымен қатар стартаптар үшін немесе тұрақты емес жұмыс жүктемелері үшін тамаша төлем үлгісі. Сіз стартап екеніңізді елестетіп көріңіз, сіздің жұмыс жүктемелеріңіз төмен, себебі сіз клиенттік базаны әлі құрмадыңыз, сіз танымал болмағандықтан, сіздің кодыңыз тек қосымшаңыз қолданылған кезде ғана іске қосылады. Егер сіздің қосымшаңызды ешкім пайдаланбаса, сіз сәйкесінше төлем жасамайсыз.

AWS Lambda сервисінің артықшылықтарын егжей-тегжейлі қарастырайық. AWS Lambda көмегімен сізге жаңа бағдарламалау тілдерін үйренудің қажеті жоқ. Ол қазірдің өзінде ең танымал бағдарламалау тілдерін қолдайды, соның ішінде Python, JavaScript, Node.js, Ruby, C# және т.б. Сіз өзіңіздің Lambda-ны іске қосқан кезде AWS әкімшіліктің көп бөлігін ол өз мойнына алатынын түсіндіңіз ғой, сізге тек кодты жүктеу керек. AWS сонымен қатар кіріктірілген апаттың алдын алады. Бұл нені білдіреді? Бұл сіздің кодыңыз жұмыс істейтін микроконтейнерлер аймақтағы бірнеше availability zone-да орналасқанын білдіреді. Сондықтан егер бір availability zone-да бірдеңе болса, сіздің контейнеріңіз басқа availability zone-да жұмыс істей береді. Ол AWS Lambda-ға кіріктірілгендіктен сіз оны тіпті байқамайсыз да, білмейсіз де. Осылайша, сіздің барлық AWS Lambda сұрауларыңыз сәтті өңделеді. Бірнеше AWS Lambda-ны іске қосу қажет болғанда кейбір жүктемелер болады, бұл күрделі логиканың бір түрі және бір лямбда басқа лямбданы іске қоса алады немесе бір лямбданы орындау нәтижелеріне байланысты кейбір қосымша шарттарда басқа лямбдаларды іске қосуға немесе іске қоспауға болады. Бұл күрделі логиканы құру немесе бірнеше лямбда-функцияларын оркестрлеу үшін AWS Step Functions сервисін пайдалануға болады. AWS Step Functions сәйкесінше осы логиканы сипаттайды және логикаға байланысты ол белгілі бір лямбданы шақырады. Бірақ лямбда бөлек, дербес, тәуелсіз функцияны қолданып, сұранысқа байланысты дайын жауапты қайтарады. Мысалы, сізде веб-сайт бар және сіз бір өнім туралы ақпарат сұрайсыз. Осылайша, сұраныс сіздің лямбдаға келгенде, лямбда кіріс деректерін пайдаланып, сіздің өнімдеріңіз туралы ақпаратты сақтайтын қоймаға барады және қажетті ақпаратты шығарып, жауап ретінде қайтарады. Қойма ретінде AWS S3 сервисінің объектілі сақтау қоймасы пайдаланылады. Егер сізде SQL дерекқоры болса, бұл AWS RDS сервисі; егер сізде NoSQL дерекқоры болса, бізде AWS DynamoDB сервисі бар, яғни AWS Lambda-ны басқа сервистермен біріктірудің көптеген нұсқалары бар. Тағы бір маңызды артықшылығы – pay-per-use pricing, яғни сіз өзіңіздің лямбда іске қосылған уақытқа ғана төлейсіз. Бұл сіздің жұмыс жүктемелеріңіз тұрақты болмаса, шығындарыңызды айтарлықтай үнемдеуге мүмкіндік береді.

AWS Lambda-мен бірге жұмыс істейтін ең танымал сервистер туралы сөйлесейік. Объектілік сақтау туралы айтқанда, Amazon S3 сервисі – ең танымал нұсқа. Ол осы сервиспен үйлесіп, ешқандай мәселе туындатпайды. Тағы бір нұсқа, біз кейбір деректерді құрылымсыз түрде сақтауымыз керек десек, біз NoSQL дерекқорларын пайдалануымыз керек. Бұл жағдайда Amazon DynamoDB сервисін қолданамыз. Amazon DynamoDB сонымен қатар serverless NoSQL дерекқорының шешімі болып табылады. Осылайша, ол сонымен қатар AWS Lambda-мен біріктірілген.

Әрі қарай, бізде екі Amazon SNS пен Amazon SQS сервистері бар. Amazon SNS – белгілі бір жағдайларға байланысты Push форматында хабарландыру жібере аламыз. Amazon SQS – бұл кезекпен жұмыс және Pull форматындағы жүктемелер, яғни Amazon SNS жағдайында біз хабарламаның қандай да бір түрін жібереміз және ол дереу алушыларға жіберіледі. Amazon SQS сервисінде бұл Pull, яғни Amazon SQS сервисінде хабарламалардың белгілі бір саны кезекте жиналады және біз Pull режимінде тұрамыз, яғни біз қажет болған кезде өңдеуге қажетті хабарламалардың санын тартамыз. Бұл бұлтта архитектура құрудың екі нұсқасы, және екі нұсқа да AWS Lambda-мен біріктірілген.

Біз API, яғни Application Program Interface құруымыз керек деп айтқан кезде, біз AWS Lambda-ны Amazon API Gateway-мен біріктіре аламыз. Бұл ең танымал AWS Lambda байламдарының бірі.

Тағы бір қолданыстағы, бірақ аз танымал сервис опциясы – бұл Application Load Balancer. Мысалы, сізде EC2 кластері бар деп елестетіп көріңіз, ол іске қосылып, кейбір жүктемелерді жасады, бірақ сіз кешкі 8-ден кейін EC2 инстанстарыңыздың 50%-ын өшіру керектігін білесіз, себебі жүктеме күрт төмендейді. Бұл жағдайда сіз өзіңіздің лямбдаңызды кешкі сағат 8-де іске қосатын event орната аласыз, содан кейін AWS Lambda Amazon EC2 инстанстарының санын азайтатын Application Load Balancer-ге сұрау жібереді.

Енді лямбда-функциясын іске қосу үшін қандай параметрлер жасау керектігі туралы толығырақ сөйлесейік. Біріншіден, бізге жұмыс істейтін бағдарлама коды қажет. Әрі қарай, біз лямбда-функциясы үшін IAM role-ін баптауымыз керек. Егер сіз лямбда-функциясын AWS Management Console-де жасасаңыз, онда функция өңдеу журналдарын сол жерде жаза алатындай етіп, ең аз құқықтары бар рөл, атап айтқанда AWS CloudWatch сервисінің құқықтарымен жасалады. Егер лямбда-функциясы басқа AWS сервистерімен жұмыс істеу үшін қажет болса, сізге тиісті policy, құқықтарды және қолжетімділікті осы role-ге қосу керек. Сондай-ақ, Run time-ді, яғни сіздің кодты қандай ортада, яғни Python, Node.js, C# және т.б. іске қосылуын көрсету керекпіз.

AWS Lambda кодының өзінен басқа, біз басқа байланысты тәуелділіктерді қолдана аламыз. Бұл сіздің кодты іске қосу үшін қажет қосымша кастомды кітапханалар болуы мүмкін. Бұл жағдайда сізге мұрағат жасап, оны AWS Lambda сервисіне жүктеу қажет. Қосымша кітапханаларды пайдаланбаған кезде, кітапханалар сирек пайдаланылады, ал Python жағдайында оның стандартты кітапханалары көптеген тапсырмалар үшін жеткілікті. Бұл жағдайда бізде тек бағдарлама коды бар және сіз AWS Lambda-ның Console-інде жұмыс істей аласыз, яғни кодты сол парақшада көшіріп, қойып, сақтай аласыз, бұл өте ыңғайлы.

Іске қосу уақытына қатысты мәселе, функция максимум 15 минут, яғни 900 секунд жұмыс істей алады. Одан артық іске қосыла алмайды. Егер сізге көбірек уақыт қажет болса, бір лямбда келесі лямбданы іске қосатын Step Functions-ті пайдалануды қарастыруыңыз керек, сол кезде сіз қосымша 15 минут уақытын аласыз. Егер бұл нұсқа сізге сәйкес келмесе, контейнерлерге немесе EC2 инстанстарын қарастырып көріңіз. Сіздің бизнес мақсатыңызға байланысты, қайсыбір нұсқасы сізге тиімдірек болуы мүмкін.

Функциямыздың қуаты туралы айтатын болсақ, біз оған бөлетін жедел жад көлемін басқара аламыз. Бұл функция үшін кемінде 128 МБ немесе 10 ГБ дейін жедел жад шығара аламыз. Ядро саны жедел жад көлемінің ұлғаюына пропорционалды түрде артады. Біз оны жедел жадтан бөлек орната алмаймыз.

AWS Lambda сервисін пайдалану мысалын қарастырайық, өте танымал мысал. Бізде EC2 инстанстарының жиынтығы бар және біздің жұмыс жүктемелерімізге байланысты біз барлық инстанстарымызды кешкі сағат 10-да өшіре аламыз, ал таңғы 5-те іске қосуымыз керек. Оның бәрін жасау үшін архитектураны қалай теңшей аламыз. Біз екі event жасаймыз. Бұл event-тер AWS CloudWatch сервисінде жасалады. Бірінші event кесте бойынша кешкі 10-да бірінші лямбданы іске қосады. Бұл лямбда бұл event-ті кіріс ретінде қабылдайды. Сондай-ақ, осы лямбдада немесе event-те біз өшірілетін инстанстар тізімін тіркей аламыз және бұл іске қосылған кезде, ол тиісті rоle-ге ие, бұл rоle осы инстансты өшіруге құқылы. Ол тиісті командаларды іске қосады және біздің инстанстарымыз тоқтатылады.

Екінші event кесте бойынша таңғы 5-те басқа екінші лямбданы іске қосады, бұл лямбда тиісті инстансты іске қосуға құқығы бар rоle-ге ие. Сондықтан AWS CloudWatch event-і біздің лямбда-функциясын іске қосқан кезде, ол оның кодын орындайды және инстанстар тізімінен кіріс деректеріне байланысты ол осы инстанстарды іске қосады.

Енді екінші мысалды қарастырайық. Бізде фотосуреттері бар қосымша бар деп елестетіп көрейік. Біздің қосымша пайдаланушысы бір фотосуретті жүктейді. Бұл фотосурет Amazon S3 bucket-те сақталынады. Содан кейін, файл Amazon S3 bucket-іне жүктелгеннен кейін, біздің лямбданы іске қосатын event қосылады. Біздің лямбда іске қосылды, event-те кіріс деректер жазылды, соның ішінде осы суретке сілтемесі бар ақпарат. Біз қандай сурет жүктелгенін білеміз. Содан кейін лямбда өз rоle-ін пайдаланады және тиісті құқықтарға ие болады. Ол бұл суретті жүктейді, өңдейді және әртүрлі ажыратылымдықта жасайды. Оның ішінде thumbnail деп аталатын суретті де құрады. Бұл – суреттің шағын нұсқасы, оның салмағы өте аз, бірақ ол тізімдегі суреттерімізді көргенде көрсетіледі. Содан кейін, барлық қажетті суреттерді дайындағаннан кейін, ол бұл суреттерді басқа Amazon S3 bucket-іне жүктейді. Сонда қосымша басқа пайдаланушыларға осы фотосуреттердің тізімін әртүрлі ажыратымдылықта көрсете алады, сонымен қатар біздің барлық суреттеріміздің тізімін қарау кезінде шағын нұсқасын көрсету үшін thumbnail-ды қолдана алады.

AWS Lambda сервисінің негізгі лимиттерімен жақынырақ танысайық. Мұнда мен көптеген лимиттер soft және AWS Support-ке хабарласу арқылы кеңейтілуі мүмкін екенін атап өткім келеді. Сіз бір аймақтың аясында бір уақытта 1000-ға дейін Lambda execution-дарын іске қоса аласыз. Сізде белгілі бір функция бар деп елестетіп көрейік, ол қандай да бір өнім туралы ақпаратты тасымалдайды, және егер кенеттен 1000 пайдаланушы сол өнім туралы ақпарат сұраса және 1000 лямбданы бір уақытта іске қосу керек болса, AWS автоматты түрде 1000-ға дейін лямбда жұмыс істейтін микроконтейнерлер жасайды және осы лямбдалардың барлығын сәтті іске қосады. Әрі қарай, егер сұраныстар саны күрт азаятын болса, барлық жұмыс істеп тұрған микроконтейнерлер AWS ішкі логикасына сәйкес біртіндеп жойылады. Егер сіз көбірек сұраулар алсаңыз, retry логикасын реттей аласыз және сұрауыңыз біраз уақыттан кейін қайтадан жіберіледі. Сол кезде аз лямбдалар іске қосылып тұрғандықтан, бұл сұраулар да сәтті өңделеді.

Егер біз функциямыздың өлшемі, соның ішінде код, сондай-ақ іске қосу үшін барлық қажетті кітапханалар туралы айтатын болсақ, онда ол мұрағатталмаған күйде өлшемі 250 мегабайтқа дейін болуы мүмкін. Егер біз Lambda қуаты туралы айтатын болсақ, онда біздің функцияларымыздың біреуі үшін кем дегенде 128 мегабайт жедел жады және ең көбі 10 гигабайт жедел жады бөлуге болады. Бұл жерде процессордың ядро саны жедел жадыдан бөлінген мегабайт санына параллель бөлінетінін атап өткен жөн. Егер біз бір функцияны іске қосудың максималды ұзақтығы туралы айтатын болсақ, онда бұл 900 секунд немесе 15 минут.

Сонымен біз бесінші бөлімнің соңына келдік. Ең маңызды тармақтарға тоқталып өтейік. AWS Lambda – бұл бізді серверсіз қуатпен қамтамасыз ететін сервер. Мұндағы идея біз өз серверлерімізді басқармаймыз, тек біздің кодты енгіземіз. Бұл кодты іске қосу туралы барлық әрекет AWS жағында жасалады. Сонымен қатар, AWS Lambda автоматты масштабтауды қамтиды, яғни егер біздің лямбдаға сұраныстар кенеттен өссе, барлық сұраулар қатарынан өңделіп отырады. AWS Lambda біздің функцияларымыздың бірі үшін 10 гигабайтқа дейін жедел жады бөле алатындай жеткілікті қуатты болуы мүмкін. Сонымен қатар, біздің әр лямбда максимум 15 минут іске қосыла алады. Бұл көптеген тапсырмаларды орындау үшін жеткілікті уақыт. Сонымен біз AWS Lambda сервисінің маңызды деген аспектілеріне тоқталып өттік. Бұл сервис сізге болашақта стартаптарды құру кезінде пайдалы болады деп ойлаймын. Бұл бұлтта кез-келген нәрсені іске қосу үшін тамаша таңдау. Дегенмен, AWS Lambda-ға тым байланып та қалмаңыз. Тұрақты болған жүктемелеріңізді басқа қуатты сервистерге ауыстыру қажет болатын жағдайлар болады. Бірақ біз бастапқы жағдай туралы айтатын болсақ, ештеңе анық емес әрі белгісіз болса, онда лямбдадан бастап осы лямбдамен бірге жұмыс істейтін басқа сервистерге дейінгі барлық serverless сервистер мінсіз шешім болып табылады.

Міне, біз алтыншы бөлімге де жеттік. Бүгінгі біздің дәрістің соңғы бөлімі. Біз бұл бөлімде AWS Elastic Beanstalk сервисі туралы толығырақ тоқталамыз. AWS Elastic Beanstalk – бұл PaaS, яғни Platform as a Service түрінде ұсынылатын Compute сервистерінің тағы бір мысалы. Мұндағы идея сіз өзіңіздің веб-қосымшаңызға арналған кодты жүктейсіз, AWS Elastic Beanstalk сол веб-қосымшаны іске қосу үшін қажетті барлық инфрақұрылымды баптайды. Инфрақұрылымды баптау дегеніміз – веб-қосымшаны орнату, жүктемені теңестіруді баптау, сонымен қатар автоматты масштабтау және веб-қосымшаны бақылау және тіркеуге қатысты барлық операциялар. Бұл жерде айта кету керек, бұл сервисті пайдалану тегін, яғни сіз бұл сервис үшін бір доллар да төлемейсіз. Бірақ AWS Elastic Beanstalk сервисінің аясында қолданылып, бапталған барлық ресурстар стандартты тарифтер бойынша төленеді. Мысалы, кейбір енгізу параметрлерін AWS Elastic Beanstalk сервисіне жіберсеңіз және сіз үшін екі Amazon EC2 инстансы бар инфрақұрылым бапталса, осы екі Amazon EC2 инстансы үшін осы сервистің стандартты тарифі бойынша төлейсіз. Бұл AWS Elastic Beanstalk өзара әрекеттесетін барлық басқа IT-ресурстарға және басқа AWS сервистеріне қатысты, яғни сіз сол екі Amazon EC2 инстансын өзіңіз баптасаңыз да дәл сондай ақша төлейтін едіңіз. Жалғыз айырмашылық – инфрақұрылым AWS Elastic Beanstalk сервисі арқылы сіз үшін автоматты түрде бапталынды. Шын мәнінде, сіз сол ақшаны төлейсіз, сонымен қатар өз уақытыңызды үнемдейсіз және инфрақұрылымды автоматты түрде баптайсыз.

Мұнда сіз AWS жұмысының қанша бөлігін AWS Elastic Beanstalk сервисі арқылы алатынын анық көре аласыз, яғни сіз басқаратын нәрсе – сіздің кодыңыз және AWS Elastic Beanstalk сервисінің параметрлері. Қалғанының барлығын AWS басқарады, яғни HTTP-сервері, application server, language interpreter, т.б. кодтың орындалу ортасы, операциялық жүйе және тіпті host. AWS Elastic Beanstalk-ті AWS Management Console-інде пайдалануға болады немесе AWS Elastic Beanstalk көмегімен инфрақұрылымды іске қосу үшін AWS CLI пайдалануға болады. Платформаны қолдау туралы айтқанда, Docker, Go, Java, NET, Node.js, PHP, Python және Ruby, басқа платформаларға қолдау көрсетіледі. Егер біз қолдау көрсетілетін веб-серверлер туралы айтатын болсақ, Java қосымшалары үшін Apache Tomcat қолдау көрсетеді, ал PHP және Python қосымшаларына Apache HTTP Server қолдау көрсетеді. Егер біз Node.js қосымшалары туралы айтатын болсақ, олар үшін NGINX және Apache HTTP Server қолжетімді. Егер Ruby қосымшасы болса, бұл Passenger және Puma. Егер біз .NET қосымшалары туралы айтатын болсақ, Java және Docker-мен бірге Go, Microsoft Internet Information Services, яғни IIS, қолдау көрсетеді.

AWS Elastic Beanstalk сервисінің негізгі артықшылықтарын қарастырайық. Біріншісі – сіз өте тез және оңай бастай аласыз, яғни веб-қосымшаңызды іске қосасыз, барлық қажетті инфрақұрылым бапталады және сіз мұның бәріне аз уақыт жұмсайсыз. Тағы бір мәселе – сіздің мамандарыңыз, әзірлеушілеріңіз операциялық жұмыстардан босатылып, өз бизнес мәселелерімен айналысса болады. Осылайша, әзірлеушілеріңіздің жұмысының тиімділігі артады. Тағы бір мәселе – AWS Elastic Beanstalk көптеген веб-қосымшаларға үйлесімді. Осылайша, AWS Elastic Beanstalk сервисінің мүмкіндіктері веб-қосымшаңызға жеткіліксіз болып қалуы сирек кездеседі. Көп жағдайда сіз AWS Elastic Beanstalk-пен жұмыс істегенде бәрі жеткілікті болады. Тағы бір мәселе – AWS Elastic Beanstalk сіздің ресурстарыңызды икемді түрде реттеуге мүмкіндік береді. Мысал ретінде, сіз Elastic Beanstalk сіз үшін көтере алатын Amazon EC2 аясында нақты instance type көрсете аласыз.

Сонымен біз алтыншы бөлімнің соңына жеттік. Негізгі ойларға назар аударайық. Егер сізде веб-қосымша болса, сіз AWS Elastic Beanstalk сервисін міндетті түрде қарастыруыңыз керек, өйткені бұл сіздің IT-инфрақұрылымыңызды бұлтта орнату процесін оңтайландыруға көмектеседі. AWS Elastic Beanstalk Java, NET, PHP, Node.js, Python, Ruby, Go, Docker және басқаларын қоса алғанда, платформалардың кең спектрін қолдайды. Төлем туралы айтатын болсақ, бұл сервис AWS клиенттері үшін толығымен тегін және сіз бұл сервисті пайдаланғаныңыз үшін төлем жасамайсыз. Бірақ сіз оның ресурстарын қолданатын сервистеріне стандартты тарифі бойынша төлем жасауыңыз қажет. Мысалы, егер AWS Elastic Beanstalk сервисі Amazon EC2 инстанстарын баптаса, сіз сол Amazon EC2 инстанстарына стандартты пайдалану тарифтерінің бөлігі ретінде төлейсіз.

Сонымен, біз бүгінгі сессиямыздың соңына келдік. Ең маңызды сәттерге назар аударайық. Осы екі дәрісте біз AWS-те қандай есептеу сервистері қолжетімді екенін көрдік. Біз Amazon EC2 сервисімен тереңірек таныстық. Әрі қарай, біз контейнерлермен жұмыс істеген кезде қандай сервистер бар екенін қарастырдық. Бұл Amazon EKS, Amazon ECS және Amazon ECR. Содан кейін біз serverless шешіммен таныстық. Бұл AWS Lambda, қандай артықшылықтары мен кемшіліктері бар екенін талқыладық. Соңында біз веб-қосымшаны жылдам орналастыру үшін қолданылатын сервиспен таныстық. Бұл AWS Elastic Beanstalk. AWS сервистерінің кең ауқымын ұсынатындығын және сіздің бизнес қажеттіліктеріңізге байланысты сіз қолданатын опциялардың біреуі немесе бірнешеуі сізге сәйкес келуі мүмкін екенін көре аласыз. Бұл сіз үшін ең қолайлы және ең тиімді болады.

Егер сізге белгілі бір сервис туралы қосымша ақпарат қажет болса, мұнда сіз үшін пайдалы болуы мүмкін қосымша сілтемелерді таба аласыз.

Бүгінгі дәріс осымен аяқталады. Сіз жаңа ақпарат білдіңіз деп ойлаймын. Келесі дәрістерде кездескенше. Сау болыңыздар!