ICT-infrastruktuuri pilvialustalla -päättötyö

Aku Järvinen AA3953

Pilviarkkitehtuurin suunnittelu

Tätä päättötyötä varten halusin tehdä arkkitehtuurin Minecraft-serverille, joka tulisi itselleni käyttöön, ja jota voisin jatkokehittää tulevaisuudessa. Arkkitehtuurin laitoin pystyyn Cloudformation templaattia käyttäen.

Templaatti koostuu seuraavista komponenteista:

VPC (Virtual Private Cloud): Virtuaalinen verkkoympäristö, jolla voidaan eristää infrastruktuuri. Mahdollistaa aliverkkojen määrittelyn sekä lisää turvallisuutta ja korkeaa käytettävyyttä.

InternetGateway: Tämä liitetään VPC:hen. Mahdollistaa liikenteen internetin ja VPC:n välillä.

RouteTable: Määrittelee miten liikenne ohjataan aliverkoissa ja VPC:ssä.

Route: Tässä templaatissa määrittää reitin 0.0.0.0/0-osoitteeseen, mikä tarkoittaa kaikkea liikennettä.

SubnetA ja SubnetB aliverkot: Nämä ovat aliverkot, jotka sijaitsevat eri saatavuusalueilla. Ne lisäävät korkeaa käytettävyyttä, jos esimerkiksi toisessa saatavuusalueessa on ongelmia. Ne myös jakavat kuormaa keskenään.

SubnetARouteTableAssociation ja SubnetBRouteTableAssociation: Liittävät aliverkot reititystauluun, jotta liikenne ohjataan oikein. Tämä varmistaa, että liikenne kulkee oikeiden aliverkkojen läpi.

Ec2SecurityGroup: Palomuurisäännöt, jotka sallivat tai estävät liikenteen instansseille. Tässä tapauksessa SSH-liikenne sallitaan vain tietystä IP-osoitteesta ja Minecraft-liikenne (Portti: 25565) sallitaan mistä tahansa IP-osoitteesta.

LoadBalancerSecurityGroup: Palomuurisäännöt Load Balancerin käyttämälle portille 25565. Tämä mahdollistaa Minecraft-liikenteen kulun Load Balancerin kautta.

NetworkLoadBalancer: kuormanjakaja, joka vastaanottaa serverillä pelaavien yhteydet ja jakaa ne tasaisesti instansseille saatavuusalueiden välillä. Tämä parantaa suorituskykyä ja käytettävyyttä. Käytin Network Load Balanceria, koska se tukee TCP ja UDP protokollia, joita Minecraft -serveri käyttää.

TargetGroup: Ryhmä instansseja, joihin Network Load Balancer ohjaa liikenteen.

LoadBalancerListener: Kuuntelija, joka vastaanottaa yhteydet Load Balancerille ja ohjaa ne Target Groupiin.

LaunchConfiguration: Käynnistyskonfiguraatio, joka määrittää instanssien asetukset, kuten käyttöjärjestelmän, koon ja käyttöoikeudet.

AutoScalingGroup: Automaattisen skaalauksen ryhmä, tässä tapauksessa se varmistaa, että aina on vähintään kaksi instanssia käytettävissä. Esimerkiksi jos yksi instanssi kaatuu, tämä luo uuden instanssin tilalle.

InstanceProfile: IAM-profiili instansseille.

InstanceRole: IAM-rooli, joka määrittää instanssien oikeudet, kuten pääsyn EC2-instanssin tietoihin ja ominaisuuksiin.

Tämän templaatin voisi pilkkoa moneen osaan kuten esimerkiksi Verkkoinfrastruktuuriin missä on VPC, aliverkot ja niiden konfiguraatio. Turvallisuusryhmään missä määritellään turvallisuusryhmät, kuten "Ec2SecurityGroup" ja "LoadBalancerSecurityGroup". Instanssimoduuliin missä määritellään EC2-instanssien luominen, lisäksi tässä voi olla myös mukana Auto Scaling Group. IAM roolit voi myös jakaa omaan kokonaisuuteensa.

Templaatin pilkkominen loogisiin kokonaisuuksiin voisi auttaa selkeyttämään ja hallitsemaan CloudFormation-templaattia. Se mahdollistaa myös tarvittaessa tiettyjen komponenttien muokkaamisen ja päivittämisen ilman että se vaikuttaisi koko arkkitehtuuriin. Pilkottuja kokonaisuuksia voi myös käyttää muissa projekteissa.

Toteutus

Aluksi meinasin toteuttaa todella yksinkertaisen arkkitehtuurin, koska minua mietitytti osaanko toteuttaa yhtään monimutkaisempaa arkkitehtuuria Cloudformation-templaattia käyttäen. Opettajan rohkaisemana päätin kuitenkin ottaa haasteen vastaan ja suunnitella, sekä toteuttaa hieman monimutkaisemman ratkaisun. Onnekseni päädyin toimivaan arkkitehtuuriin.

Arkkitehtuurin monimutkaisuus on mielestäni keskitasoa. Se sisältää useita AWS-palveluita ja hyödyntää korkean käytettävyyden periaatteita. Arkkitehtuurissa käytän saatavuusalueita, aliverkkoja, Load Balanceria ja useita instansseja, joiden avulla liikenne jaetaan, tukien siten korkeaa käytettävyyttä.

Arkkitehtuurin pystytyksessä ongelmitta en kuitenkaan selvinnyt. Erityisesti palveluiden konfiguroinnin ja integroimisen kanssa oli aluksi ongelmia, mutta onneksi AWS:llä on runsaasti dokumentaatiota, joka auttoi ratkaisemaan nämä ongelmat. Lisäksi CloudFormation antoi selkeät virheilmoitukset, mikä helpotti templaten kanssa tekemistä ja sen muokkausta. Näin ollen työskentely templaatin parissa oli sujuvaa.

Kaavion tekemiseen käytin Draw.io -ohjelmaa. Kaavio ”eli” koko prosessin ajan, kun palveluita lisäsi ja poisti arkkitehtuurista. Kaavio auttoi minua hahmottamaan eri komponentit ja niiden väliset yhteydet. Kaaviot voivat olla myös hyvä tapa välittää omat ideat ja ratkaisut, jos vaikka työskentelee tiimissä.

Koska halusin tehdä arkkitehtuurin nimenomaan Minecraft-serverille, valitsin palvelut sen mukaan. Onneksi opintojaksolla opittiin käyttämään juuri niitä palveluja mitä tähän arkkitehtuuriin tarvitsisi. EC2-instanssiin sain Minecraftin serverin pyörimään, Load Balancer mahdollistaa liikenteen tasapainottamisen useiden EC2-instanssien välillä, Auto Scaling -ryhmä puolestaan varmistaa, että instansseja on aina käytettävissä. Näiden lisäksi käytin VPC:tä ja aliverkkoja, joilla pystyi luomaan eristetyn ympäristön Minecraft-serverille mikä lisää turvallisuutta. Turvallisuutta lisää myös EC2-instanssin ja Load Balancerin Security Groupit, millä pystyn hallinnoimaan liikennettä palvelimeen, tässä tapauksessa liikenteen avaaminen vain tarvittavalle portille auttaa suojaamaan palvelinta haitalliselta liikenteeltä.

Pohdinta

Loppujen lopuksi päättötyöstä muodostui mielenkiintoinen ja opettavainen projekti, jota todennäköisesti tulevaisuudessa tulen jatkokehittämään. Korjattavaa löytyypi vielä paljon. Pitäisi miettiä miten saan autoscalingin toimimaan siten, ettei se pystytä heti uutta instanssia, jos päätän itse pysäyttää instanssin. Lisäksi pitäisi keksiä miten saan koko homman toimimaan siten, ettei serveri häviä kokonaan, jos instanssi terminoituu, vaan serverin ja sen datan saisi siirrettyä autoscalingin pystyttämään instanssiin mahdollisimman automaattisesti. Nykyisessä ratkaisussa userdata script ajetaan molemmissa pystytettävissä instansseissa, vaikka sen ei tarvitsisi ajaa kuin yhdessä. Tähänkin pitää keksiä tulevaisuudessa ratkaisu.

Serveri kuitenkin toimii ja siihen pystyy kaverit liittymään, joten olen tyytyväinen. Olisi kiva testata isolla porukalla kuinka vakaa serveri on ja kuinka load balancer toimii.