Projektiraportti

Tavoittenani oli opetella eri vaihtoehtoja ajaa Docker containereita ja hoitaa kuormanjako eri määrän resurssia vaativien containereiden välillä. Yritin sinnikkäästi asentaa omaa Kubernetes klusteria, mutta tämä osoittautui hieman liian vaikeaksi. Vaihtoehtoisena tapana koitin myös ajaa Docker containereitani Nomadissa.

Nomad toimi varsin mukavasti ja tuntui tarjoilevan kaiken oleellisen suoraan paketista. Toisaalta sen konfiguroiminen oli kuitenkin hieman työläämpää Kubernetesiin verrattuna. Kuberentesin kanssa tuntui pääsevän hieman nopeammin eteenpäin, koska siihen ehkä tuntui löytyvän esimerkkejä ja ohjeita paremmin internetistä.

Kubernetes osoittautui varsin näppäräksi työkaluksi, kun käytti valmista Googlen tarjoamaa palvelua. Myös monitorointi Prometheuksella ja Graphanalla oli tehty todella helpoksi. Sovelluksen pystyttäminen ja kuormanjako eri podeille onnistui myös kivasti ilman yllätyksiä.

Projektin koodit löytyy osoittesta: https://github.com/pihvi/dockerhy/tree/master/projekti

Sovellus

Tein mahdollisimman yksinkertaisen simulaation sovelluksesta, millä on yksinkertainen ”frontti” eli HTML lomake ja yksinkertainen ”backend”, mille frontti lähettää formin. Sovelluksessa simuloidaan backend kyselyn olevan raskas CPU:lle. Näin pystyisin testaamaan kuorman jakoa kevyen ja raskaan containerin välillä.

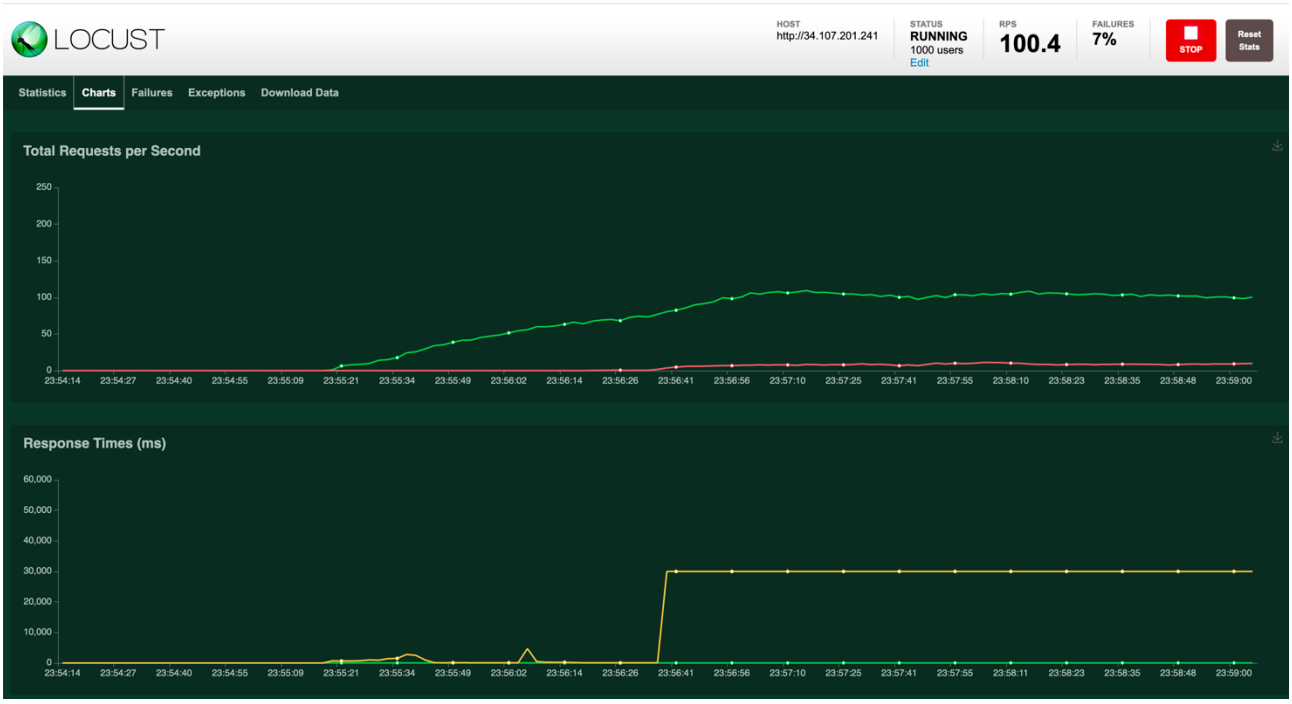
Tein sovellukset Go:lla, koska sillä sain Docker imaget varsin pieniksi, alle 10MB kappaleelta. Tämä oli lähinnä kiva Nomadin kanssa, koska sillä sai helposti ladattua kyseiset imaget ilman container registryä.

Sovellus saattaa olla vielä ajossa Kubernetes klusterissa osoitteessa: http://34.107.201.241/

LB testaus

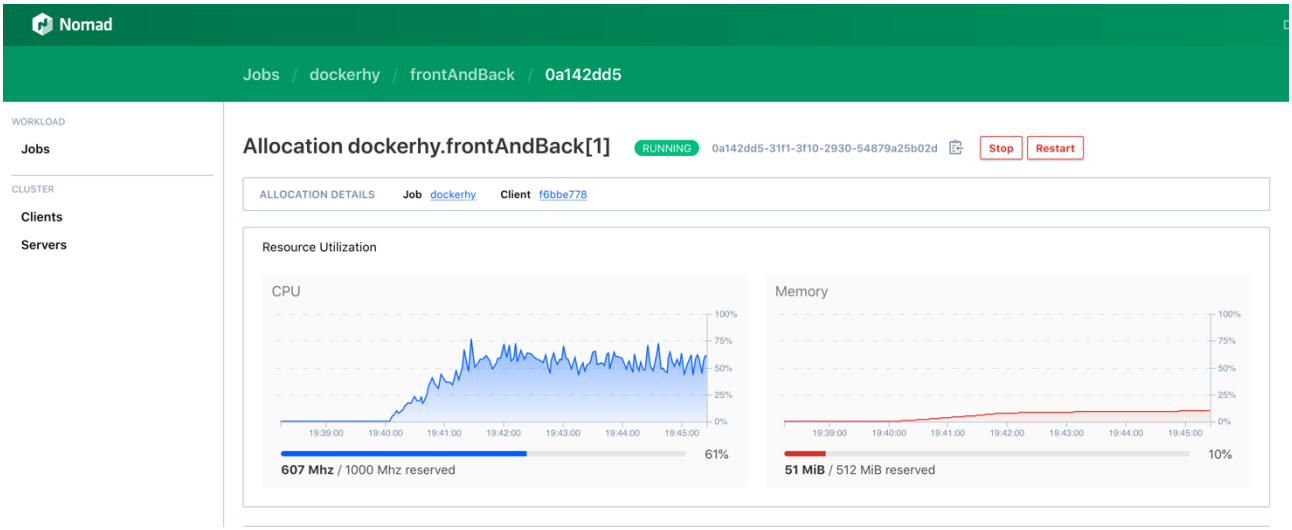
Halusin testata, että kuorman jako eri podeille toimii oletetusti Kubernetes klusterissani. Oletuksena sovelluksen frontti ei vaatisi juurikaan resursseja ja backend taas huomattavasti enemmän. Testi teki 10 kertaa enemmän pyyntöjä frontille, kuin backendille.

Kuvassa Kubernetes klusterin CPU ja muistin käyttö Graphanassa Prometheuksen raportoimana.



Kuvassa suorituskykytestin eteneminen.

Kuvaajista näkee hyvin kuinka frontin CPU käyttöä ei edes havaitse, vaikka sillä on backendiin nähden 10 kertaa enemmän pyyntöjä. Backendin podien CPU käyttö sen sijaan kasvaa melko tasaisesti. Kuormanjako näyttää siis toimivan oletetusti. Sovelluksen kriittinen piste tulee vastaan selkeästi kohdassa, missä suorituskykytestissä vasteaika pomppaa (keltainen viiva). Samalla kohtaa näkee kuinka punainen viiva pompsahtaa hieman, kun backend kyselyt alkavat hajoamaan.



Nomadin tarjoama monitorointinäkymä.

Yhteenveto

Olin kuullut, ettei omaa Kubernetes klusteria kannata perustaa ja ylläpitää, vaan tulisi käyttää pilvessä tarjottuja valmiita palveluja. Tämä projekti vahvisti tätä käsitystäni todella vahvasti.

Nomad vaikutti olevan ihan näppärä peli yksinkertaisen sovelluksen deployaukseen. Se muistuttaa ehkä enemmän Docker Composea, kuin Kubernetesia. Se kuitenkin vaatii oman vaivansa, joten toisaalta sitä voisi ehkä mieluummin sitten käyttää sitä Composea tai sitten Kubernetesia pilvipalveluna.

Kubernetes vaikuttaa olevan varsin helppo ja hyvä ratkaisu kontainereiden ajoon, kun sen ottaa valmiina palveluna pilvestä. Alkuun pääsee helposti ja eteneminen hieman pidemmällekin tuntuu menevän varsin vaivattomasti.

Kurssi oli itselleni kaiken kaikkiaan varsin täydellinen. Opin paljon Dockerista, mikä oli tavoittenani, mutta iloisena yllätyksenä myös Kubernetesista ja Nomadista.

Päiväkirja

18.10.

Heti Unity workshopin jälkeen perustin oman Kubernetes clusterin Googlen pilveen. Opeteltuani ensin palvelun toimintoja, sain deployattua oman palveluni sinne varsin helposti.

1.11.

Palattuani jatkamaan projektin kehitystä, huomasin tuon oletusklusterin kuluttavan melko nopeasti tilini ilmaista 300$ saldoa. Koska tavoitteenani on lähinnä opetella kontainereita ja niiden pyörittämistä palvelimella pitkän ajan saatossa, totesin Googlen palvelun olevan tähän tarkoitukseen liian kallisarvoista. Minulla on myös pari pikku purkkia jo ajossa, joten päätin kokeilla asentaa oman Kubernetes moottorin sinne.

28.12.

Kubernetesin itse asentaminen näytti olevan jokseenkin monimutkaista, mutta OpenShift Origin vaikutti paketoivan Kubernetesin helpommin asennettavassa paketissa. Toinen purkeistani on

ARM pohjainen, eikä OpenShiftiltä löytynyt pakettia kyseiselle, joten lähdin asentamaan sitä vain Intel pohjaiseen serveriin. Asennuksen kanssa tulikin sitten tapeltua pitkä tovi huonolla menestyksellä. Asennus ei mennyt loppuun asti läpi ja virheistä löytyi useita vastaavia GitHub Issueita, mutta mikään niiden tarjoamista ratkaisusta ei lopulta vienyt asennusta loppuun asti. Päätinkin luopua tästä tavasta. Tämä kokemus vahvisti käsitystäni, että omaa Kubernetes systeemiä ei kannata ylläpitää, vaan käyttää valmiita tarjottuja palveluja kuten Googlen Kubernetes Engineä.

6.1.

Päätin testata myös vaihtoehtoisia tapoja saada kontainereja ajoon. Olin kuullut hyvää Hashicorpin Nomadista kollegalta, jolla oli kokemusta sekä Kuberneteksesta että Nomadista, joten päätin kokeilla myös sitä. Nomad asentui helposti sekä Intel että ARM pohjaiseen serveriin. Vaikka Nomadin ja Dockerin saikin pyörimään helposti ARM serverillä, Docker containereiden automaattinen haku ja tekeminen eri arkkitehtuureille ei enää ollutkaan täysin triviaalinen. Ei nyt hankalaakaan, mutta liian työlästä, että sitä olisi tämän projektin puitteissa tehnyt.

12.1.

Tein uuden Kubernetes klusterin Googleen tällä kertaa ”kokeilu” versiona, missä oli kovin vähäisillä resursseilla varustetut serverit. Oman appsin deployaus meni ihan hyvin, mutta Prometheus & Graphana kikkare ei suostunut pyörimään, koska Prometehus valitti resurssien vähyydeestä. Niinpä tein uuden klusterin oletusasetuksilla.

13.1.

Oletus konfiguraatiolla Prometheus & Graphana sujahti sukkana sisään ja tarjoili hienosti klusterista monitorointidataa. Päätin kokeilla Locust työkalua testaamaan miten klusteri käyttäytyy isomman kuorman kanssa. Ja hienostihan se käyttäytyi ja jakoi oletetulla tavalla kuormaa eri podeille. Kaiken kaikkiaan aika hyvältä maistuva kokemus tämä Googlella pyörivä Kubernetes.

14.1.

Päätin vielä kokeilla suorituskykytestausta omalla koneella ajettavaan Nomad ”klusteriin”, missä front ja back ”podeja”, eli ”allokaatioita” Nomad kielellä, oli kaksi kappaletta kumpaakin. Ihan hienosti näytti kuorma jakautuvan ja Nomad rajoitti CPU kulutusta konfiguraation mukaisesti. Yhtenä erona Kubernetes testiin verrattuna front containerin CPU kulutus oli suhteessa suurempi.

Ajankäyttö

- Unity workshop: 6h
- Google Kubernetes Engine: opettelu, setup, deploy; 8h
- OpenShift Origin: opettelu, asenneuksen säätäminen; 12h
- Nomad: opettelu, asennus, setup, deploy; 10h
- Google Kubernetes Engine: Kahden appsin deploy, loadbalanceri, Prometheus ja Graphana; 7h
- Locust: setup ja testien ajo; 2h
- Projektiraportti: 3h