Scalable Matchmaking for Online Multiplayer Games
CSC 562 - Distributed Systems Project

Sunil Kumar

August 10, 2019

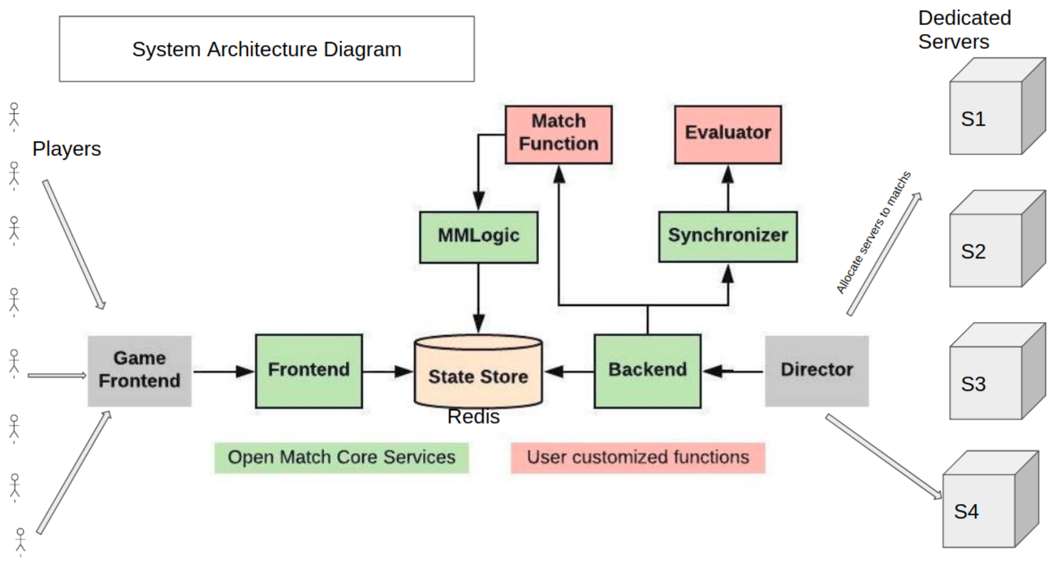
1 Abstract

The goal of this project is to make a good matchmaking system for online multiplayer games. Matchmaking in games can be defined as an art of matching a set of players together to maximize their enjoyment of the game. Google's Open source project Open-Match is already tackling this problem and is in alpha stage. So, the project is based on it to integrate a client and server with Open-Match to simulate a complete application including a game_client, players, matchmaking system, server, etc. The final application will be deployed and running on a single Kubernetes cluster on Google Cloud Platform using the Kubernetes Engines in it.

Final Project Demo Video:- https://youtu.be/d38UetVeEH0

2 System Design

System Architecture Diagram with arrows showing data flow (based on open match architecture):-



Below are the description of non-trivial components:-

- Frontend: A service that handles tickets in open-match.

- Backend: A service that generates matches.
- Mmlogic: A gateway service that supports data querying in open-match.
- Match function: A service where your custom matchmaking logic lives in.
- Evaluator: evaluates proposed matches for there quality.
- Director:- back-end service which does communication between open match and dedicated game servers.

So, my dummy player's process, game client and director are part of the same service called om-demo written in Go language with a demo front-end using JavaScript and HTML to show whats going on in the system. The goal is to be able to have 1000 players playing concurrently without any issues. Lets now look at the technologies involved and deployment details in the next section.

3 Implementation and Deployment

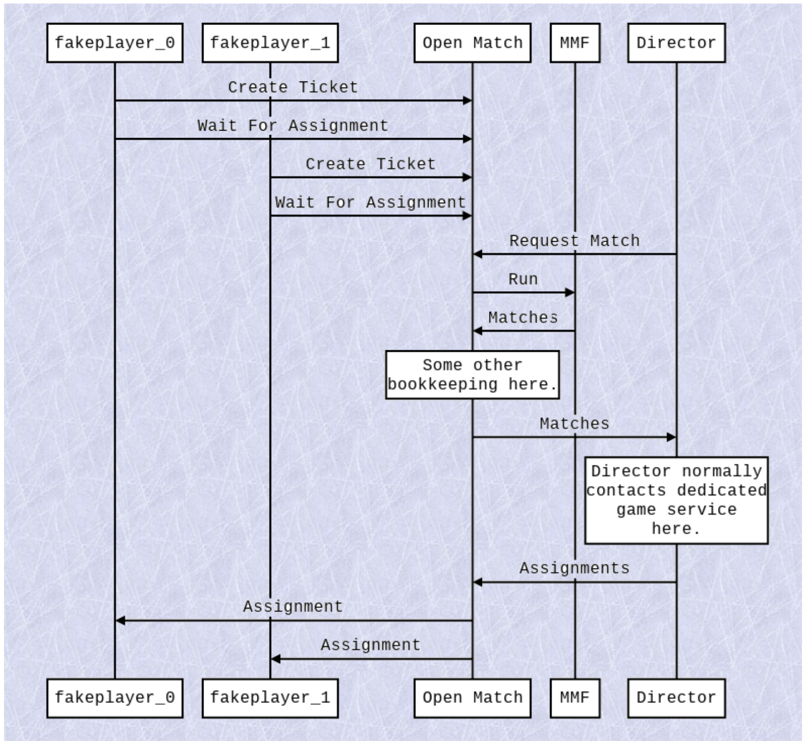
Technologies used:-

- gRPC with REST APIs
- Swagger UI (Display framework for REST APIs)
- Golang
- Docker
- Google Cloud Platform
- Kubernetes (Google Cloud Kubernetes Cluster)
- Minikube (For creating local Kubernetes cluster on Virtualbox)
- Open-Match (Google's open-source matchmaking framework for games)
- Make , Javascript, HTML etc.

Following Technologies were part of open-match:-

- Grafana (open source visualization tool)
- Prometheus (open-source monitoring system)
- Redis
- Helm (tool for managing Kubernetes charts)

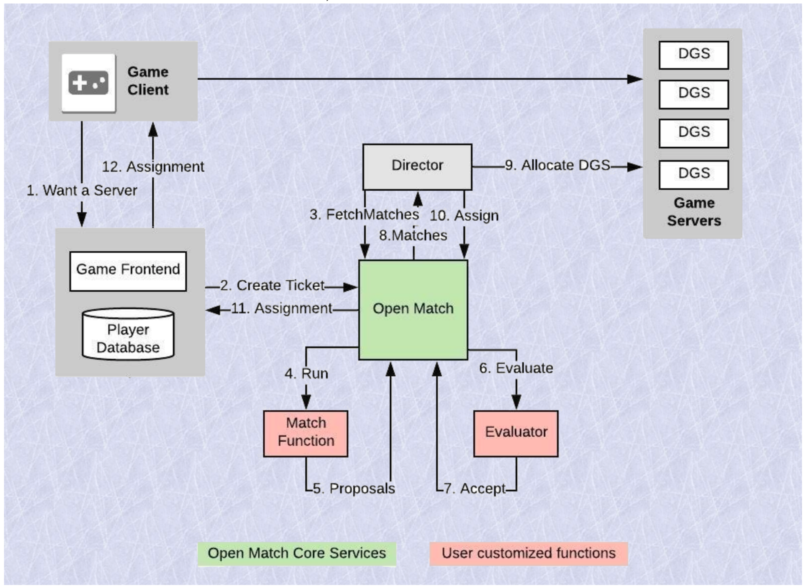
Flow Diagram of Demo using open match (source:- Open-Match Documentations) :-



Legends in the above diagram:-

- Assignment:- connection information to a game servers
- MMF:- Match Making Function i.e. service which selects players for a match after consideration for skill, latency, etc of the players in the pool (player currently looking for a match).

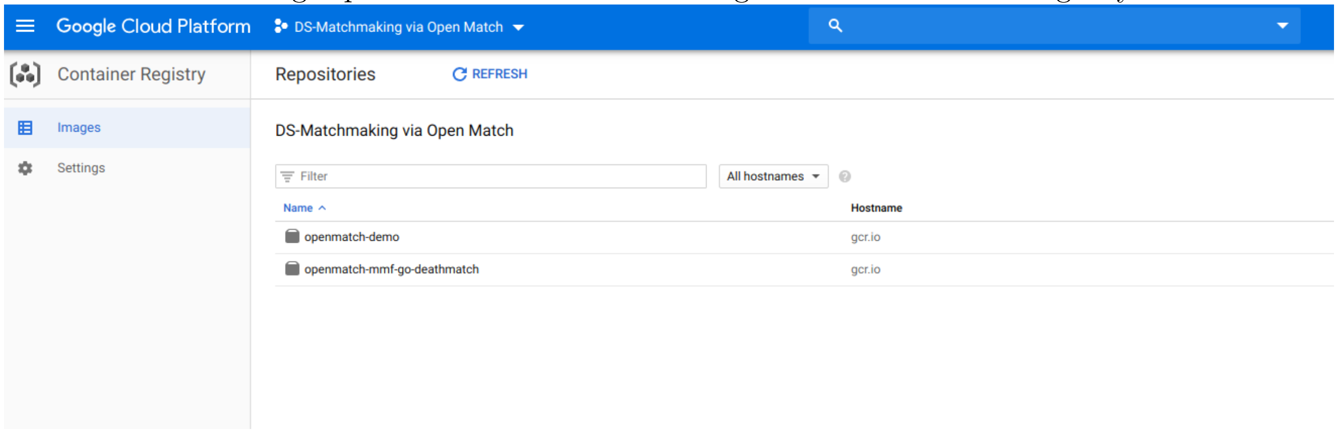
Let us see in the below diagram the high-level steps happen during matchmaking using open match (Source:Open-match documentation):-



We will be writing code to simulate the gray part in the above Diagram along with a custom

Match Function which looks for 10 players for a match in Go language because client and server stubs are already generated for the gRPCs of Open-Match.

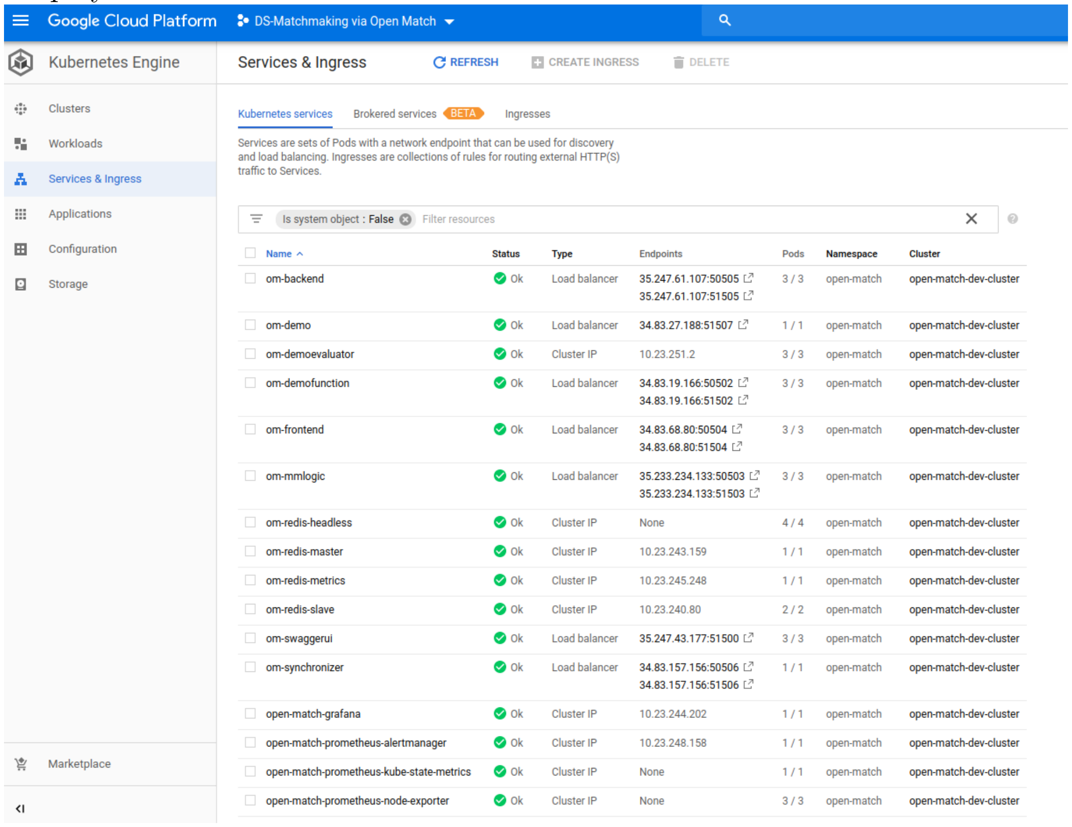
For Deployment we move our docker images to google cloud Container Registry. Below screenshot shows the 2 images part of our demo in our Google Cloud Container Registry:-



DS-Matchmaking via Open Match

Name	Hostname
openmatch-demo	gcr.io
openmatch-mmf-go-deathmatch	gcr.io

Then we deployed our images in pods within the same cluster as other services of open-match. Below screenshot shows all services of open-match along with the demo and custom matchmaking service we deployed on the Kubernetes Cluster.



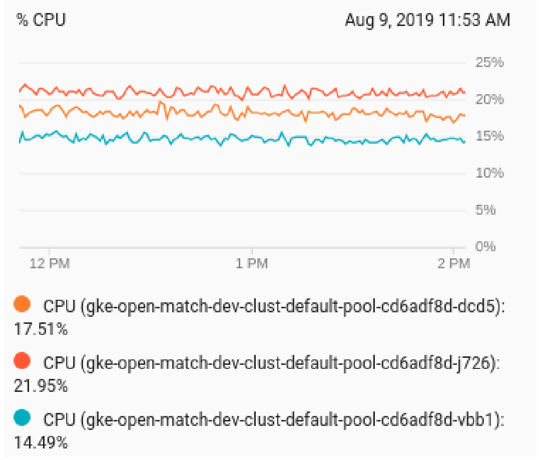
Services are sets of Pods with a network endpoint that can be used for discovery and load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to Services.

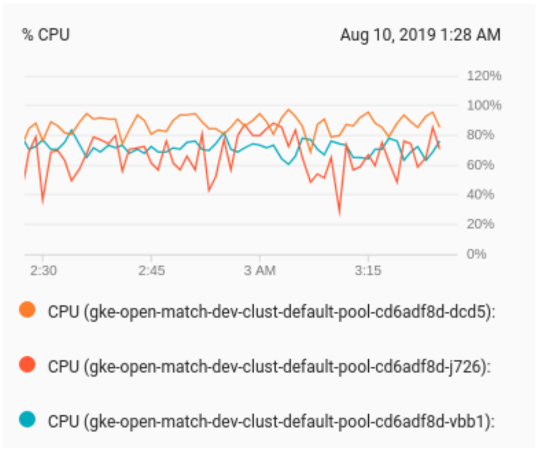
Name	Status	Type	Endpoints	Pods	Namespace	Cluster
om-backend	Ok	Load balancer	35.247.61.107:50505 35.247.61.107:51505	3 / 3	open-match	open-match-dev-cluster
om-demo	Ok	Load balancer	34.83.27.188:51507	1 / 1	open-match	open-match-dev-cluster
om-demoevaluator	Ok	Cluster IP	10.23.251.2	3 / 3	open-match	open-match-dev-cluster
om-demofunction	Ok	Load balancer	34.83.19.166:50502 34.83.19.166:51502	3 / 3	open-match	open-match-dev-cluster
om-frontend	Ok	Load balancer	34.83.68.80:50504 34.83.68.80:51504	3 / 3	open-match	open-match-dev-cluster
om-mmlogic	Ok	Load balancer	35.233.234.133:50503 35.233.234.133:51503	3 / 3	open-match	open-match-dev-cluster
om-redis-headless	Ok	Cluster IP	None	4 / 4	open-match	open-match-dev-cluster
om-redis-master	Ok	Cluster IP	10.23.243.159	1 / 1	open-match	open-match-dev-cluster
om-redis-metrics	Ok	Cluster IP	10.23.245.248	1 / 1	open-match	open-match-dev-cluster
om-redis-slave	Ok	Cluster IP	10.23.240.80	2 / 2	open-match	open-match-dev-cluster
om-swaggerui	Ok	Load balancer	35.247.43.177:51500	3 / 3	open-match	open-match-dev-cluster
om-synchronizer	Ok	Load balancer	34.83.157.156:50506 34.83.157.156:51506	1 / 1	open-match	open-match-dev-cluster
open-match-grafana	Ok	Cluster IP	10.23.244.202	1 / 1	open-match	open-match-dev-cluster
open-match-prometheus-alertmanager	Ok	Cluster IP	10.23.248.158	1 / 1	open-match	open-match-dev-cluster
open-match-prometheus-kube-state-metrics	Ok	Cluster IP	None	1 / 1	open-match	open-match-dev-cluster
open-match-prometheus-node-exporter	Ok	Cluster IP	None	3 / 3	open-match	open-match-dev-cluster

Note some services which are not part of the core functionality of open-match had some errors (can be seen in demo video) and as Open-Match is in alpha stage some errors were expected. We started with 100 players and tried to scale up to 10000 players. More details of the evaluation in the next section.

4 Evaluation

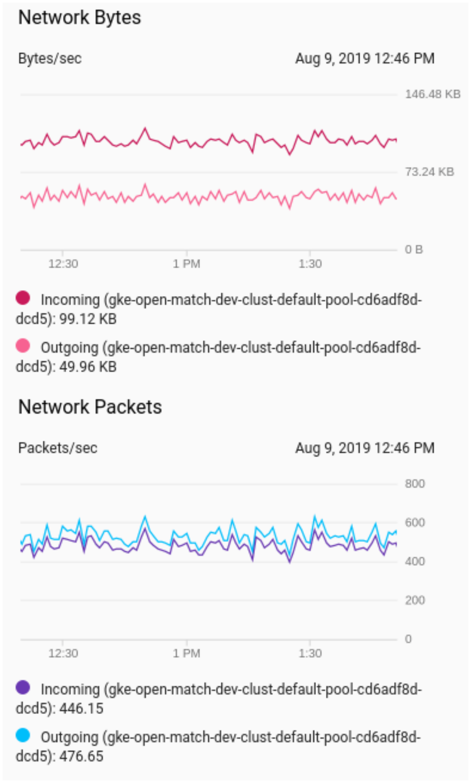
To evaluate our matchmaking system we took 2 scenarios on the google cloud platform with 100 players and with 1000 players. It is important to note that because we have created our cluster using n1-standard-2 (2 vCPUs, 7.5 GB memory) at each node, our performance is limited by hardware also. when we tried setting up with 10,000 players on our demo, some of the services stop working and hence that scenarios were not compared. Below are the CPU load graphs of all 3 nodes in the cluster when running 100 and 1000 player respectively:-

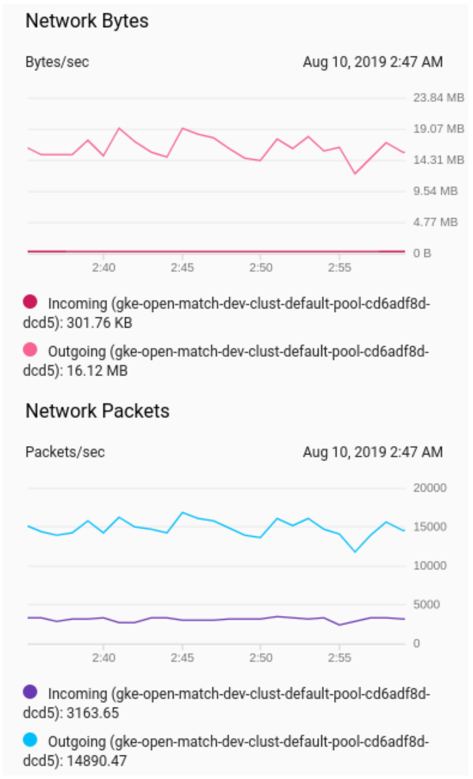




From the above graph, it was clear that 1000 player playing on 100 servers is taking up to 95% of the CPU usage and hence we cannot increase more on this hardware. Because we are using Kubernetes cluster which has Cluster Autoscaler feature this is not an issue. But due to lack of time and money, we did not explore this feature.

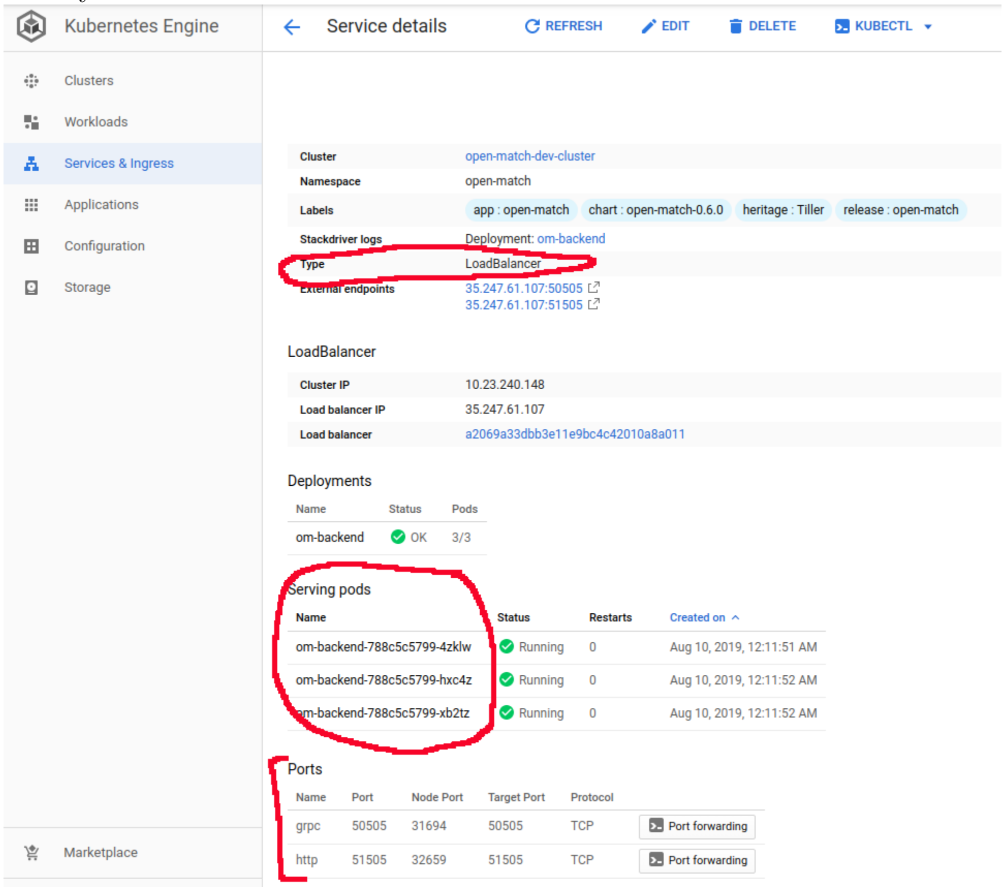
Now let us see the impact on network traffic when we move from 100 to 1000 concurrent players:-





In the above graphs, we can see the increase in the network traffic when players count increase by 10 fold. Do note that the increase in network traffic is not strictly linear it seems because we note the number of packets was increase to 3200 packets per sec from 450 packets per sec whereas the outgoing traffic increased a lot up-to 15000 packets per sec from 500 packets per sec. We suspect that this might we due to our demo web-page fetching data to display. But the exact reason is not clear to us.

Note that the core services of open-match are deployed as load balancer mode with 1 process running at each node as shown in the below image. This ensures the Reliability of these services hence the whole system.



Cluster	open-match-dev-cluster
Namespace	open-match
Labels	app : open-match, chart : open-match-0.6.0, heritage : Tiller, release : open-match
Stackdriver logs	Deployment: om-backend
Type	LoadBalancer
External endpoints	35.247.61.107:50505 35.247.61.107:51505

LoadBalancer

Cluster IP	10.23.240.148
Load balancer IP	35.247.61.107
Load balancer	a2069a33dbb3e11e9bc4c42010a8a011

Deployments

Name	Status	Pods
om-backend	✓ OK	3/3

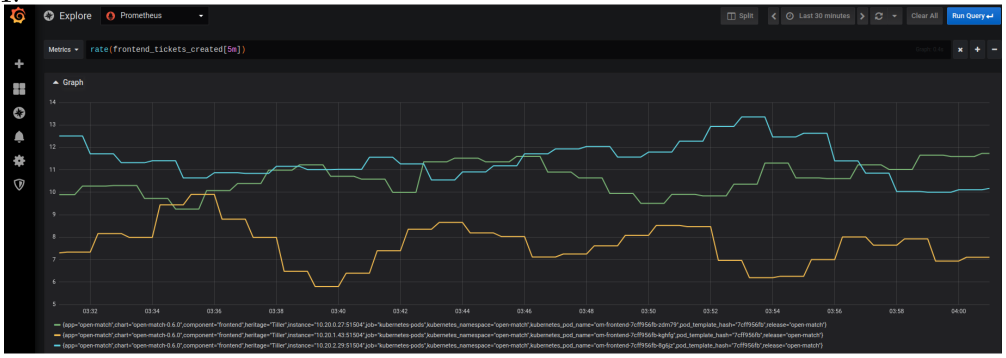
Serving pods

Name	Status	Restarts	Created on
om-backend-788c5c5799-4zklw	✓ Running	0	Aug 10, 2019, 12:11:51 AM
om-backend-788c5c5799-hxc4z	✓ Running	0	Aug 10, 2019, 12:11:52 AM
om-backend-788c5c5799-xb2tz	✓ Running	0	Aug 10, 2019, 12:11:52 AM

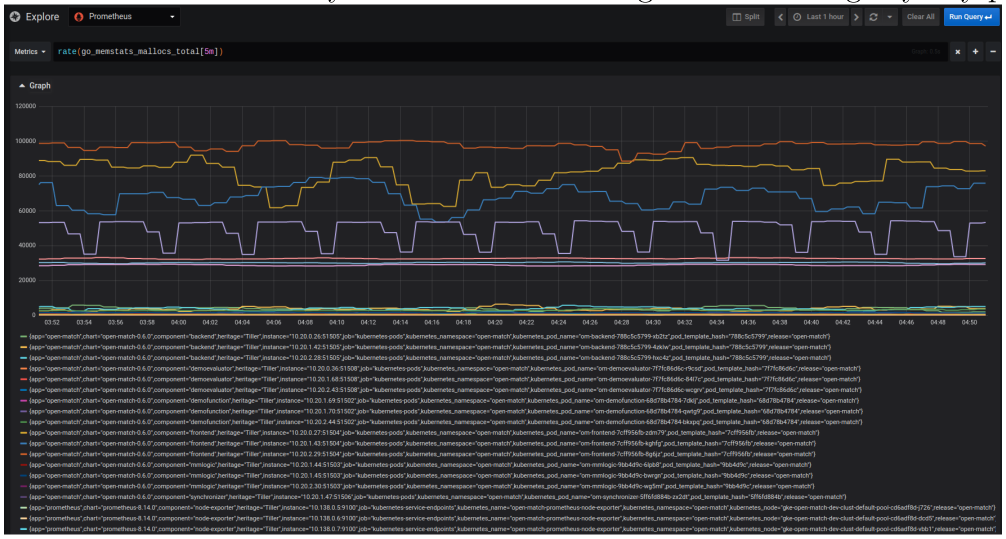
Ports

Name	Port	Node Port	Target Port	Protocol	
grpc	50505	31694	50505	TCP	Port forwarding
http	51505	32659	51505	TCP	Port forwarding

We only had time to analyze our application APIs for one scenario using grafana with Prometheus monitoring service. Below graph shows the frequency of CreateTicket() API (either by REST or gRPC) is called on front-end service of open-match. Note this way we can analyze usage of any other API in our system. The image also shows the load is correctly balanced between the nodes for each API.



Also while exploring grafana matrices we saw a lot of useful information can be fetched using it. For example CPU and memory usage by go processes, gRPC RTT, count, message size, etc. One of the interesting ones is shown below which shows the memory used by go processes which can be used to narrow down memory leaks or unwanted high member usage by any process:-



```
rate(go_memstats_mallocs_total[5m])
```

```
{app="open-match",chart="open-match-0.6.0",component="backend",heritage="Tiller",instance="10.20.0.26:51505",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-backend-788c5c5799-xb2tz",pod_template_hash="788c5c5799",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="backend",heritage="Tiller",instance="10.20.1.42:51505",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-backend-788c5c5799-4zklw",pod_template_hash="788c5c5799",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="backend",heritage="Tiller",instance="10.20.2.28:51505",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-backend-788c5c5799-hxc4z",pod_template_hash="788c5c5799",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="demoevaluator",heritage="Tiller",instance="10.20.0.36:51508",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-demoevaluator-7f7fc86d6c-r9csd",pod_template_hash="7f7fc86d6c",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="demoevaluator",heritage="Tiller",instance="10.20.1.68:51508",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-demoevaluator-7f7fc86d6c-84l7c",pod_template_hash="7f7fc86d6c",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="demoevaluator",heritage="Tiller",instance="10.20.2.43:51508",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-demoevaluator-7f7fc86d6c-wcgrv",pod_template_hash="7f7fc86d6c",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="demofunction",heritage="Tiller",instance="10.20.1.69:51502",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-demofunction-68d78b4784-7dklj",pod_template_hash="68d78b4784",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="demofunction",heritage="Tiller",instance="10.20.1.70:51502",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-demofunction-68d78b4784-qwtg9",pod_template_hash="68d78b4784",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="demofunction",heritage="Tiller",instance="10.20.2.44:51502",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-demofunction-68d78b4784-bkxpq",pod_template_hash="68d78b4784",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="frontend",heritage="Tiller",instance="10.20.0.27:51504",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-frontend-7cff956fb-zdm79",pod_template_hash="7cff956fb",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="frontend",heritage="Tiller",instance="10.20.1.43:51504",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-frontend-7cff956fb-kghfg",pod_template_hash="7cff956fb",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="frontend",heritage="Tiller",instance="10.20.2.29:51504",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-frontend-7cff956fb-8g6jz",pod_template_hash="7cff956fb",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="mmlogic",heritage="Tiller",instance="10.20.1.44:51503",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-mmlogic-9bb4d9c-6lpb8",pod_template_hash="9bb4d9c",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="mmlogic",heritage="Tiller",instance="10.20.1.45:51503",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-mmlogic-9bb4d9c-bwrgn",pod_template_hash="9bb4d9c",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="mmlogic",heritage="Tiller",instance="10.20.2.30:51503",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-mmlogic-9bb4d9c-wg5ml",pod_template_hash="9bb4d9c",release="open-match"}
{app="open-match",chart="open-match-0.6.0",component="synchronizer",heritage="Tiller",instance="10.20.1.47:51506",job="kubernetes-pods",kubernetes_namespace="open-match",kubernetes_pod_name="om-synchronizer-5ff6fd884b-zx2dt",pod_template_hash="5ff6fd884b",release="open-match"}
{app="prometheus",chart="prometheus-8.14.0",component="node-exporter",heritage="Tiller",instance="10.138.0.5:9100",job="kubernetes-service-endpoints",kubernetes_name="open-match-prometheus-node-exporter",kubernetes_namespace="open-match",kubernetes_node="gke-open-match-dev-clust-default-pool-cd6adf8d-j726",release="open-match"}
{app="prometheus",chart="prometheus-8.14.0",component="node-exporter",heritage="Tiller",instance="10.138.0.6:9100",job="kubernetes-service-endpoints",kubernetes_name="open-match-prometheus-node-exporter",kubernetes_namespace="open-match",kubernetes_node="gke-open-match-dev-clust-default-pool-cd6adf8d-dcd5",release="open-match"
{app="prometheus",chart="prometheus-8.14.0",component="node-exporter",heritage="Tiller",instance="10.138.0.7:9100",job="kubernetes-service-endpoints",kubernetes_name="open-match-prometheus-node-exporter",kubernetes_namespace="open-match",kubernetes_node="gke-open-match-dev-clust-default-pool-cd6adf8d-vbb1",release="open-match"
```

Open-match is designed as a micro-services pattern on kubernetes as the underlying platform which allow us to use kubernetes autoscaling feature to automate Scalability. It's highly Flexible because you can deploy it on any public cloud, local data center, or even on a local workstation. And it's also Extensible by adding custom matchmaking logic for any type of games. Kubernetes load balancer ensures reliability and high performance of the system.

5 Discussion & Future Work

Good Performance of a matchmaking system is very subjective based on the game and type of mode you are playing. For example in-game like Dota and Fortnite where players are looking for competitive games matches, the system needs to focus on closely skilled players group together in a match. Whereas in other games the environment could be more casual and players only need new variety in matches focusing on social experience. The above evaluation is only done for vertical scaling i.e. higher number of concurrent players, a high number of requests, etc. There is also a need to fit-in complex game logic into the matchmaking functions. Open-match does support custom matchmaking function but analysis of such matchmaking system is very hard.

While the open-match open source project moving toward a stable V1.0 build were all customizable components will fully support all languages etc. The future work of this project could be development and testing the performance of custom matchmaking function based on roles, MMR, etc. and compare the average wait time of the players. This could also include a custom evaluator to score the initial purposed matches and filter only good ones out of them.