Muhammet Tayyip Yazıcı, 2017400285
Hüseyin Can Bölükbaş, 2016400225

# CmpE48A Term Project

Architectural Overview Document

## Description

We'll investigate various scaling options in Kubernetes such as vertical, horizontal node and pod scaling. To create the load on the servers, we'll build an API that takes IO intensive, CPU intensive and memory intensive requests. We're also planning to create custom metrics (such as number of requests per second) to decide when to scale up or down. At the end we will compare the results and analyze the difference between auto scaling types in Kubernetes. We're thinking of GCP as our cloud provider.

## Monitoring

GCP has a built-in Kubernetes Engine Monitoring tool. We are planning to use that to monitor the Kubernetes cluster nodes and pods. We will monitor some of the metrics such as CPU, memory usage and disk usage for nodes and pods.

We also plan to monitor the incoming requests to our application. We can monitor these requests using the GCP Load Balancer Monitoring. With this tool, we plan to monitor metrics such as requests per second, latency, errors etc. If this tool does not provide us with all the metrics we want, we plan to use Prometheus to collect the metrics and Grafana to show the metrics in a Dashboard.

## Cloud API Scripts

We are planning to use Terraform for creating the Kubernetes cluster. There are official modules provided by the cloud providers for creating the Kubernetes Cluster. We also plan to use Prometheus for collecting the metrics from the cluster if the GCP monitoring tool is not sufficient, for this we will use the Helm provider in Terraform to deploy the Prometheus Helm Chart.

## Performance Evaluation

We are planning to build a very basic API that takes:

- IO intensive requests such as a request that involves a considerable amount of file reads and writes.
- CPU intensive requests such as a request that requires heavy calculation.
- Memory intensive requests such as a request that creates considerably big objects.

Each type of the above requests will be evaluated with the possible combinations of Horizontal Node autoscaling, Horizontal Pod autoscaling (HPA), Vertical Pod autoscaling (VPA) and without autoscaling. For each of these scenarios, we will collect metrics such as CPU and memory usage of pods and nodes. We will also collect metrics about requests such as number of requests per second, latency, errors and timeouts. As an end goal, we want to have comparison tables such as below:

| | With Node Autoscaling | | Without Node Autoscaling | | |
|---|---|---|---|---|---|
| **CPU Usage** | With HPA | With VPA | With HPA | With VPA | No Autoscaling |
| IO Intensive Requests | 10% | 12% | 10% | 12% | 15% |
| CPU Intensive Requests | 50% | 50% | 40% | 40% | 90% |
| Memory Intensive Requests | 10% | 10% | 12% | 12% | 10% |

As the load tester and performance evaluator, we are planning to use Apache JMeter since it is a widely accepted open source software.

## Scaling (Auto or Manual)

We are planning to do two types of scaling: node and pod autoscaling. For node scaling, the Google Kubernetes Engine provides a built in autoscaler that scales up if there are pods that are unschedulable due to not enough resources in all the nodes. It also scales down when the nodes are underutilized.

For pod autoscaling we want to use horizontal and vertical scaling. There are native Kubernetes objects for both horizontal and vertical auto scaling. Horizontal pod autoscaler looks at the CPU, memory or other custom metrics that are provided by the application and decides to create or delete pods. Vertical autoscaler can change the pod's requested resources by looking at its resource usage.

## Architectural Diagram