




Implementation of Serverless Architecture


Introduction



Function as a service (FaaS) is a category of cloud computing services that provides a platform allowing customers to develop, run, and manage application functionalities without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app.

Building an application following this model is one way of achieving a serverless architecture, and is typically used when building microservices applications.

Serverless Computing



Serverless computing is a cloud computing execution model in which the cloud provider dynamically manages the allocation of machine resources. Pricing is based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity.

The name serverless computing is used because the server management and capacity planning decisions are completely hidden from the developer or operator.



Motivation

- Serverless computing is more cost-effective than renting or purchasing a dedicated server, which generally involves significant periods of underutilization or idle time.
- A serverless architecture means that developers and operators do not need to spend time setting up and tuning auto scaling policies or systems; the cloud provider is responsible for ensuring that the capacity always meets the demand.
- Simplification of the task of back-end software development.

Terminology



Containers:

Containers are a method of operating system virtualization that allow you to run an application and its dependencies in resource-isolated processes.

Docker:

It is a container platform which provides a layer to the containers to build upon. In the reference of docker, a container is called a docker engine.

Docker Swarm:

Running dockers in swarm mode simply means managed and orchestrated collection of docker engines.

Architecture & Implementation



Services :

A service is a long-running Docker container that can be deployed on any machine.

These services can be replicated and multiple similar docker container instances will be created.

A service is usually an image of a microservice within the context of some larger application. For example, an HTTP server.

These services are created and orchestrated by Docker Swarm.

Services Made :



Server :

This is the service which takes in the HTTP requests from the client on port 8080 and transfers it to the WatchDog.

It maintains metrics like invocation count (prometheus_counter) which it displays on port 8000.

WatchDog :

WatchDog is the service which performs our actual function of code to latex converter. It takes in the request from the server and returns the latex code.

Services Made :



Prometheus :

Prometheus is the service which keeps track of metrics of the servers. It informs the alert manager when the traffic on a server increases above a certain threshold.

Alert Manager :

The alert manager takes alerts from Prometheus, and makes request on the server to replicate.

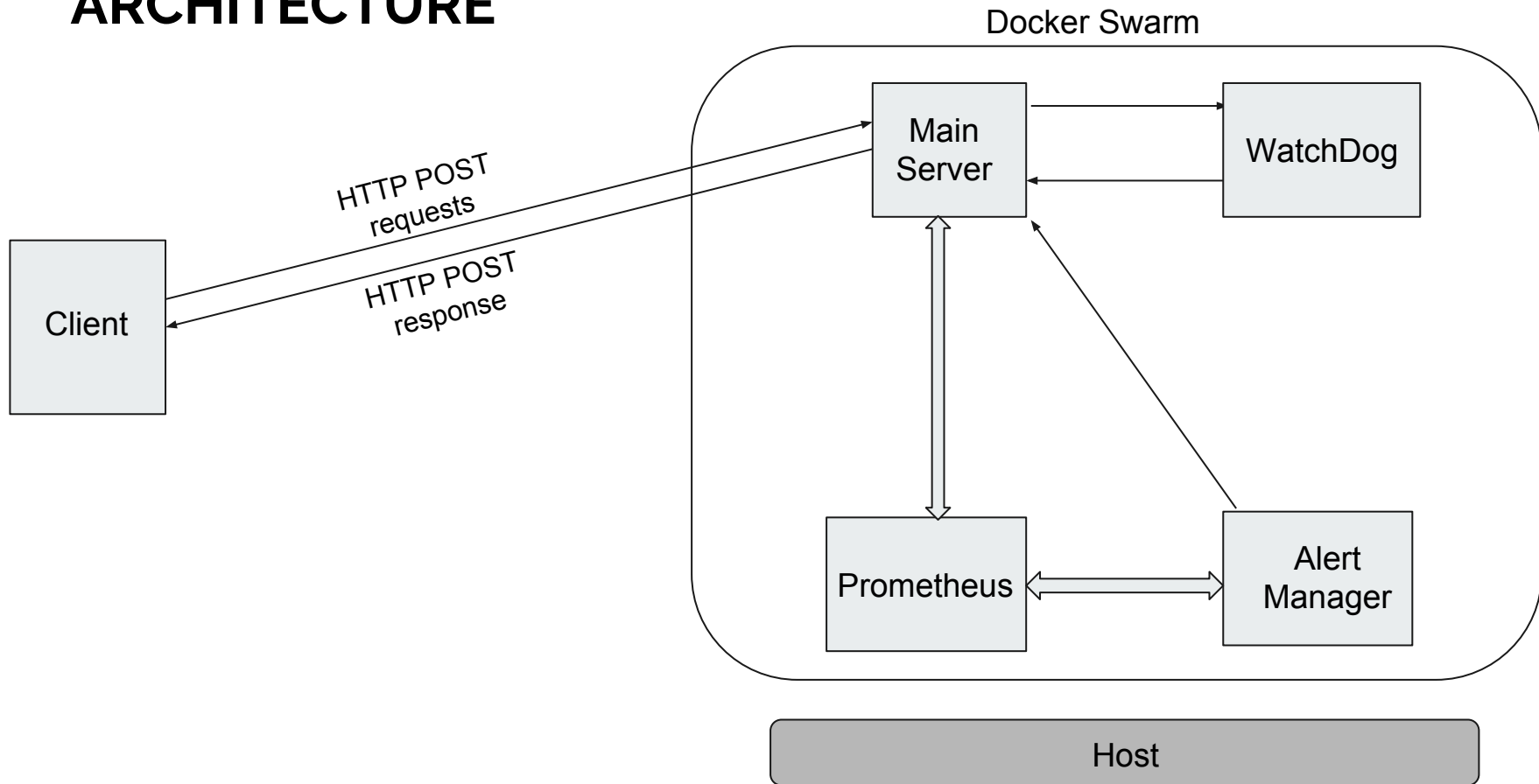
An alert has three phases :

Pending : An alert is initially in pending mode.

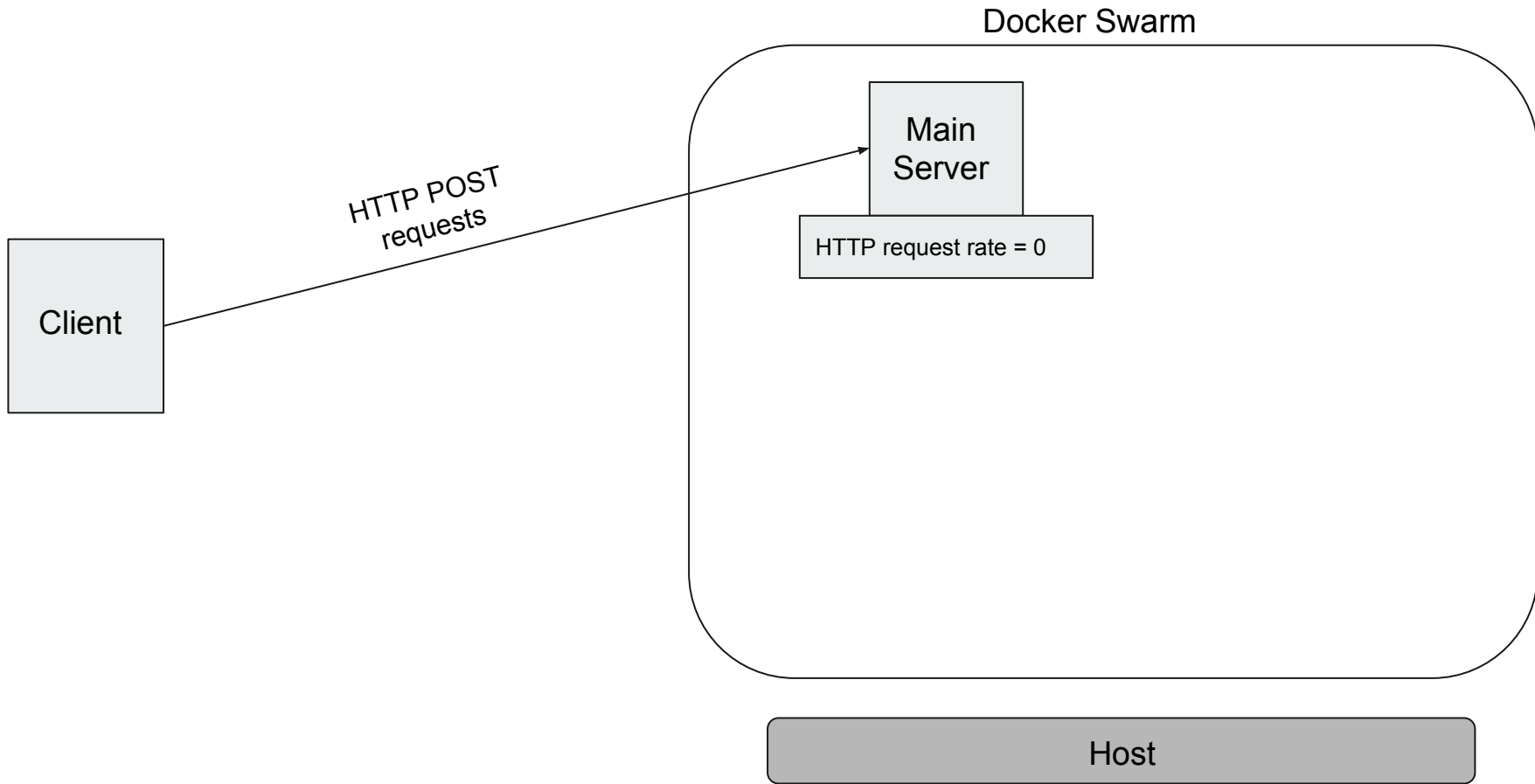
Firing : When the alert manager informs the server to replicate.

Resolved : When replication is finished, the alert is resolved.

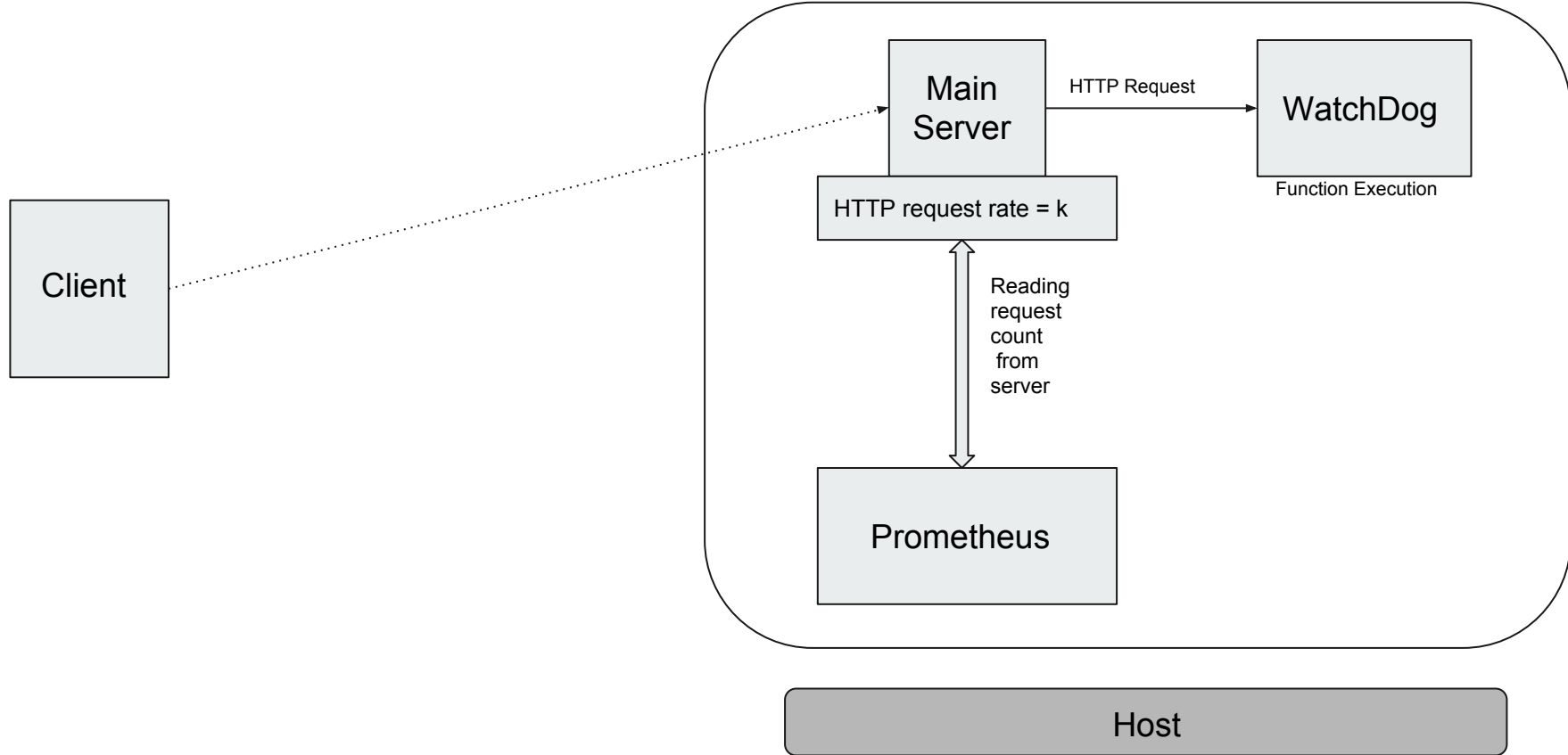
ARCHITECTURE

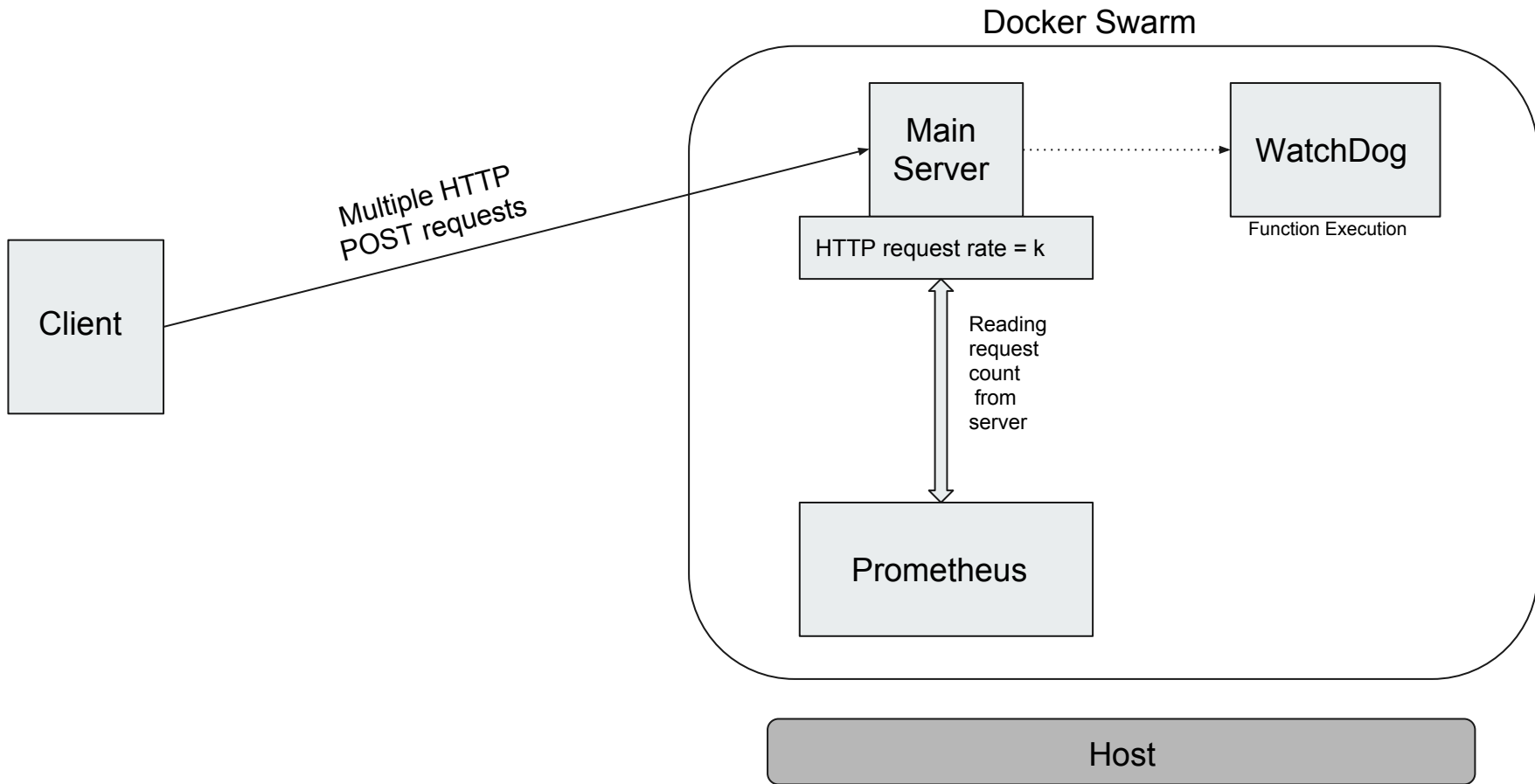


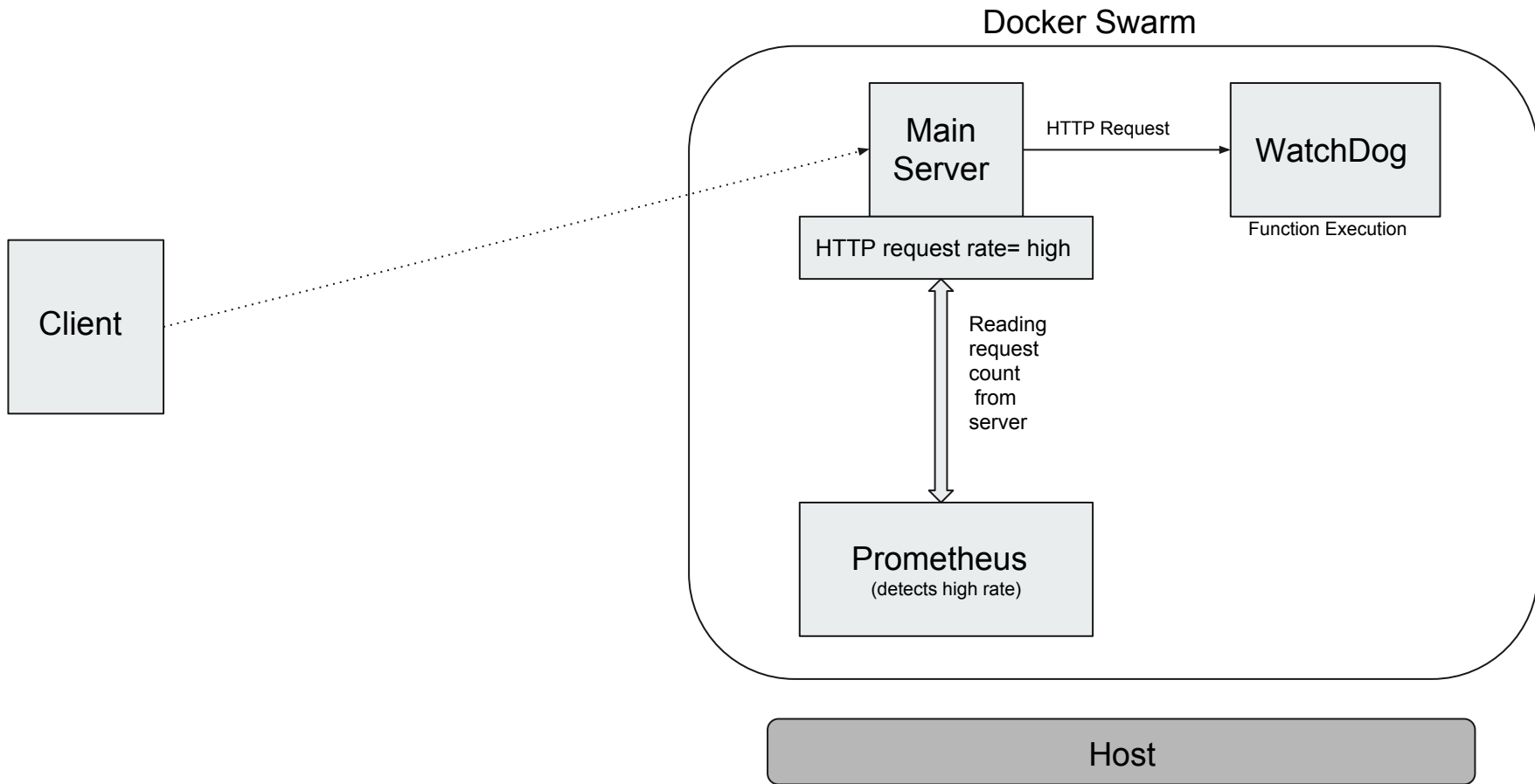
How does it work?

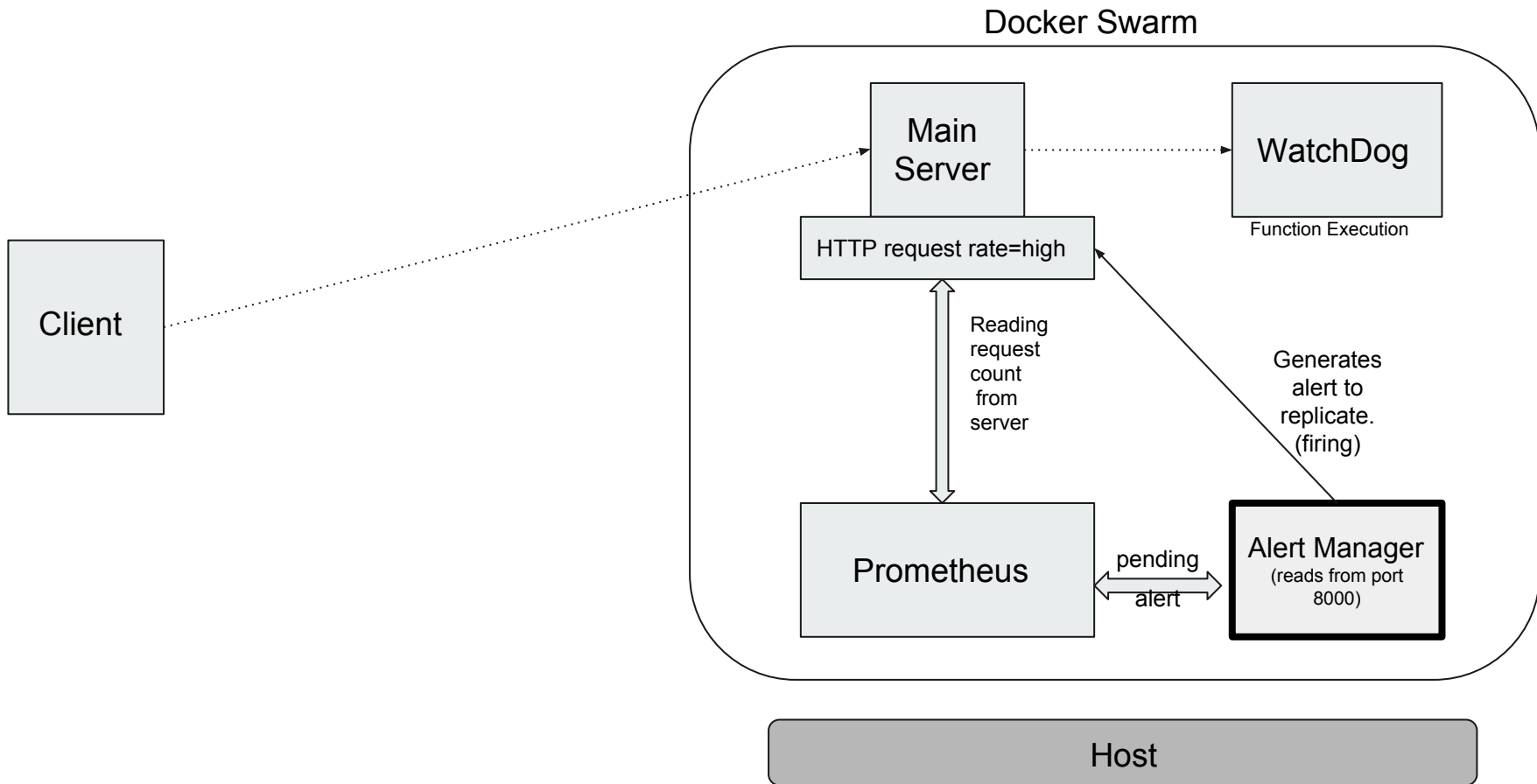


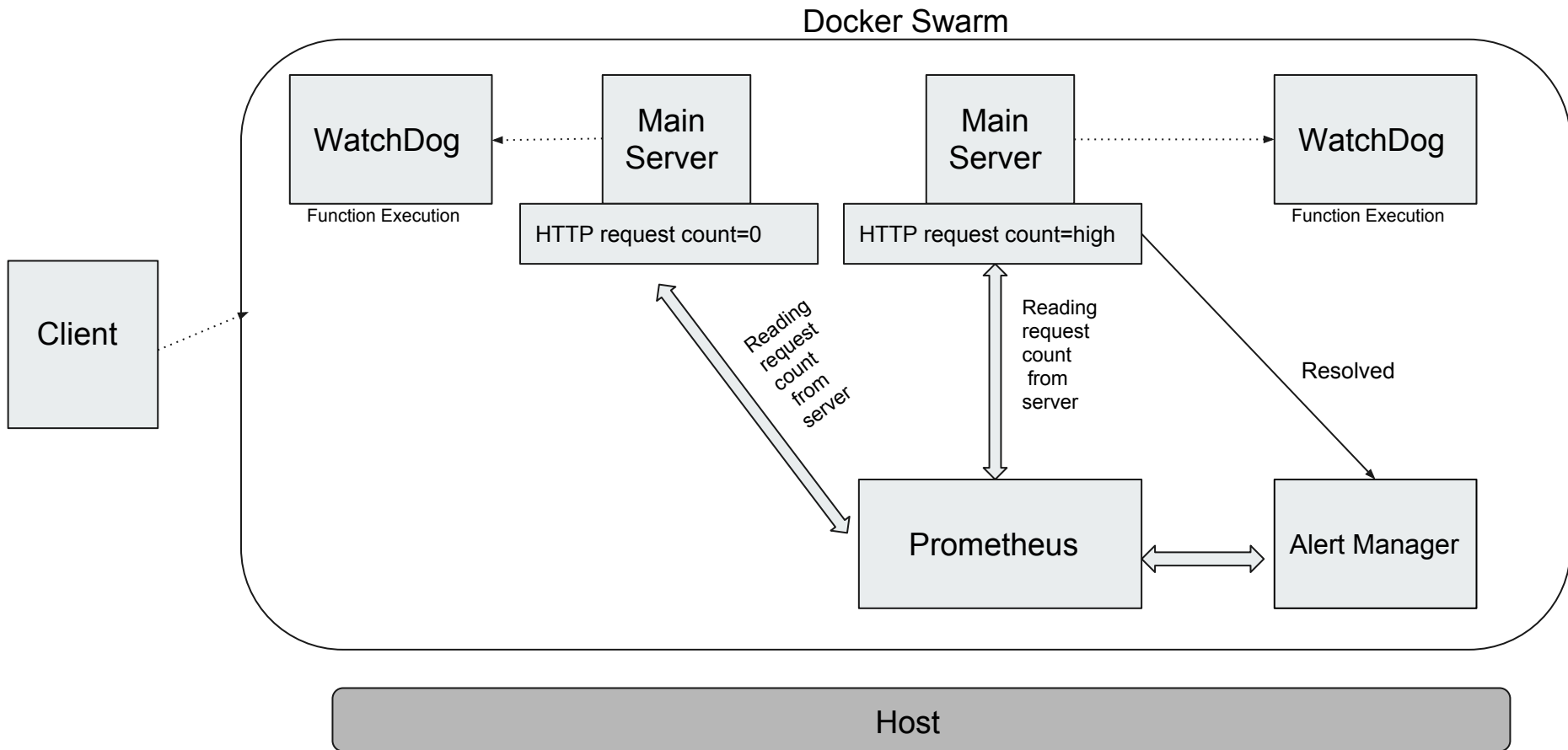
Docker Swarm














Whenever the **Main server** receives an **HTTP POST request**, it communicates with port 8080 of **WatchDog** service which is responsible for the execution of our function.

Prometheus service keeps track of requests received by the Main server. If the rate of getting request crosses a fixed threshold, Prometheus triggers the **Alert manager** to notify Main server; which in turn replicates itself to handle more requests.

Comparison

Case study from Amazon :

Lightweight and low-traffic website

Assume 10,000 hits in a day at 200 ms (0.2 sec) execution time , 256 MB memory.

Let server running time be SR and memory required be M = (256/1024) GB = 0.25 GB.

The resource requirement (RR) (in GB-sec) is thus given by,

$$\underline{RR = SR * M}$$

- **Using FaaS**

Server running time (SR) = $10,000 * 0.2 = 2,000$ sec .

Therefore, resource requirement (RR) = $SR * M = 2,000 * 0.25$ GB-sec = 500 GB-sec per day.

- **Using server-based architecture**

As number of seconds in a day is 86400, SR = 86400 sec .

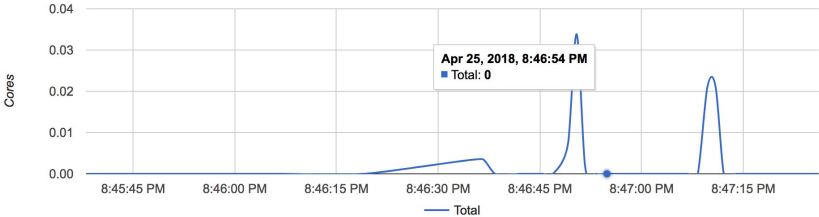
Therefore, resource requirement (RR) = $SR * M = 86400 * 0.25$ GB-sec = 21,600 GB-sec per day.

The above analysis shows that FaaS is **~43 times** more resource efficient than server based architecture in the given scenario.

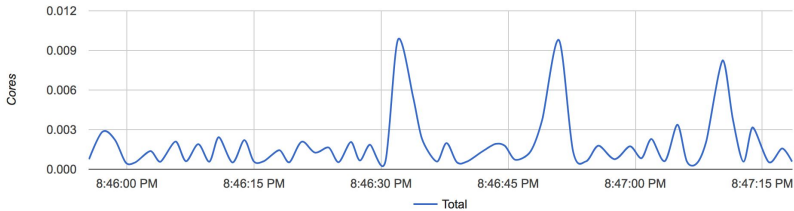
Results

CPU Usage

Watchdog

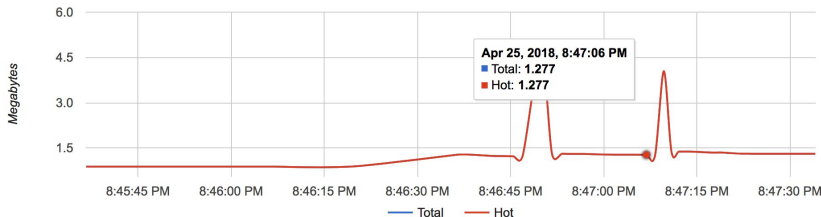


Server

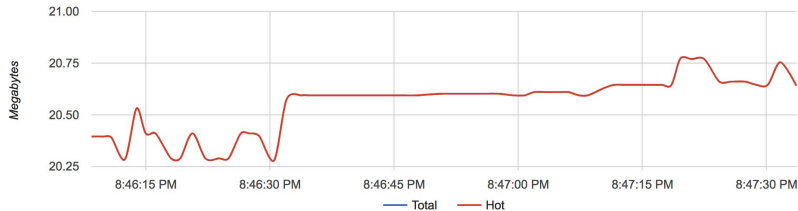


Memory Usage

Watchdog



Server





Project Supervisor

Dr. Bibhas Ghoshal

*Professor,
IIT Allahabad*

GROUP MEMBERS

Utkarsh Gupta IIT2015023

Somya Verma IIT2015026

Pallavjeet Singh IIT2015131

Vaibhav Bansal IIT2015138

Dipendra Singh IIT2015145