Report
Group 2

January 2020

Hello

Contents

1 Introduction

The purpose of this project has been to develop a chat application that allows its users to automate in- and outbound messages, through user uploaded scripts written in the service oriented programming language, Jolie, while maintaining security.

The underlying infrastructure of the application itself has been built as a micro-service architecture with the goal of affording developers greater atonomy and agility in the limited scope that they are responsible for.

The choice of a micro-service architecture also helps another key goal, namely that of good horizontal scalability; we have utilized the venerable container orchestration framework, Kubernetes to facilitate this. Every service is containerized using Docker, and managed as pods in Kubernetes. Hello

2 Preliminaries

In this section we will provide an overview of the concepts, tools and platforms that we have used and what they are.

2.1 Containerization

Containerization is the concept of packaging the environment your application needs with the application itself into a small self-contained package. In our project we have used Docker for this.

2.2 REST

REST is an API spec, that is built ontop of HTTP. We use REST for interacting with our service from the outside, since it is almost an ISO standard for how to expose your web-services.

2.3 Kubernetes

Kubernetes is a self-healing, fast, fault-tolerant, and pluggable spec, for container orchestration and networking abstraction at scale.

2.3.1 Self-healing

In the context of kubernetes, self-healing means that the kubernets master actively makes sure that containers are healthy, if not they are restarted (usually).

2.3.2 Fast

Kubernetes services are usually configured at one of the networking layers, making it extremely fast, and runtime-configurations is implemented at systemcall-level. The whole system is also written in Go, which is known to have very good performance.

2.3.3 Pluggable

Kubernetes itself is more of an api that usually has pre built-in modules. An example is the kube-dns, which can easily be plugged with another DNS. The kubernets ingress controllers can also be plugged for something like nginx, if chosen to.

2.3.4 Container orchestration

Usually people know this by automatic scheduling and resource management for containers. Imagine having 3 virtual machines and 50 containers, kubernetes automatically handles all the deploying and networking for you.

2.4 Prometheus

Prometheus is a applications monitoring spec, like JVM memory details, Go garbage collector details and much much more. It is also a server software, that can pull from REST endpoints (usually /prometheus), and then exposes another api with all the collected data that can be consumed by eg Grafana.

2.5 Grafana

Grafana is a visualization tool, it can be connected to various sources (such as prometheus and loki), for data visualization.

2.6 Loki

Loki is a brand new software from the grafana team, that harvests logs stores them in cassandra and tokenizes them for extremely fast lookup. Loki exposes an endpoint that adheres to the prometheus spec, so it is also extremely pluggable.

2.7 Redis

Redis is an extremely high-performance in-memory simple key-value store but extensible through an plugin environment. Redis is used often for query caching and session management.

2.8 Postgres

Postgres is a state of the art high performance RDBMS.

2.9 Kafka

Kafka is a fully distributed, pub-sub like messaging system that employs log techniques instead of the ordinary ack-pop of pub-sub systems. Kafka is the most widely used modern messaging system of it's category of pub-sub systems.

2.10 JSON

JSON is a data modelling language, it is simple and easy to read.

2.11 JWT

JWT stands for Json-Web-Token, which is a method of dispensing authentication tokens to users.

2.12 GCS

GCS stands for google cloud storage, its a thirdparty object storage service from google cloud.

3 Technical Description

In this section the technica

4 Related Work and Discussion

5 References