MC714 | Avaliação 1

`titanic`

Gabriel Borges, **RA** 197458
Paulo Pacitti, **RA** 185447

INSTITUTO DE COMPUTAÇÃO

O problema

- Implementar um cenário real de utilização de **computação em nuvem de um servidor web.**
- O objetivo é criar um **conjunto de máquinas virtuais que executam servidores web** que atendem clientes sem tornarem-se um gargalo.
- Para isso, o conjunto de máquinas virtuais deve ser atrelado a um **balanceador de carga** (*Load Balancer*).

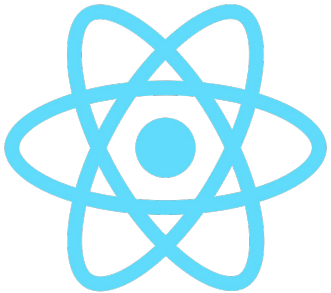
Solução Planejada

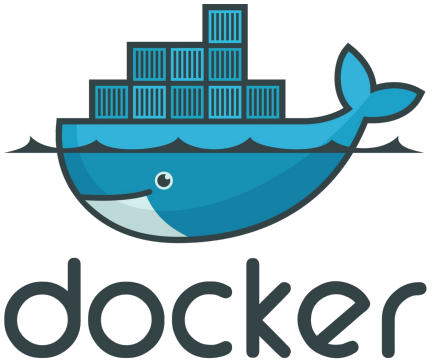
- **Aplicação**: aplicação React envelopada com Docker.
- **Infraestrutura:** Kubernetes utilizando a *Google Kubernetes Engine* (GKE) do *Google Cloud Platform* (GCP).
- **Ferramental:** infraestrutura como código com Terraform, build e registry da imagem Docker da aplicação com Github Actions.


Implementação

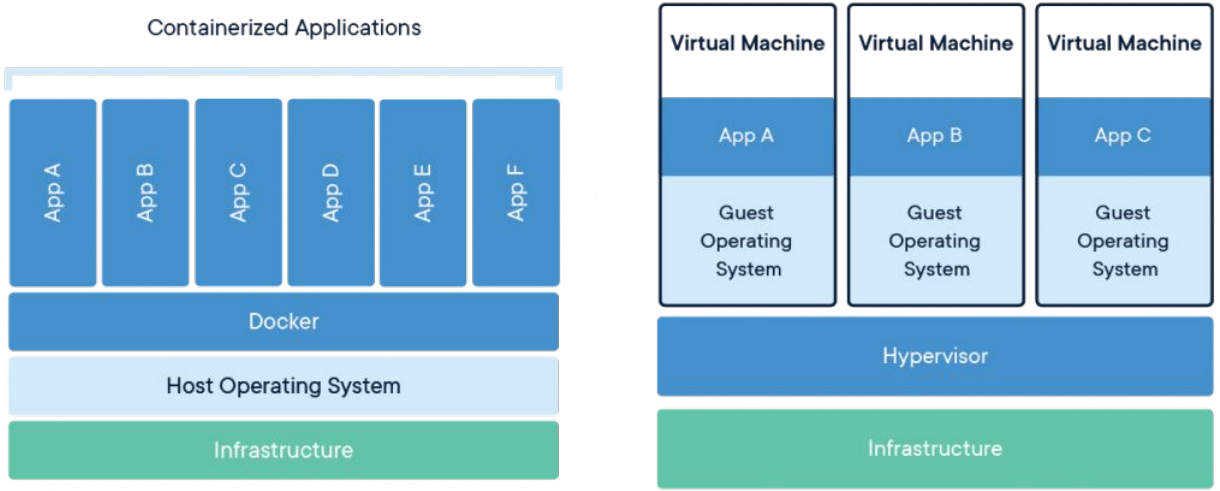
Aplicação: frontend

A ideia inicial foi criar uma aplicação Web que mostra o IP de seu host. Para isso, utilizamos de **React** (*framework* JavaScript) para criar a página Web. Após isso, fazemos o *build* do site estático e envelopamos em um *container* **Docker** com um servidor HTTP **nginx**.





Aplicação: Docker



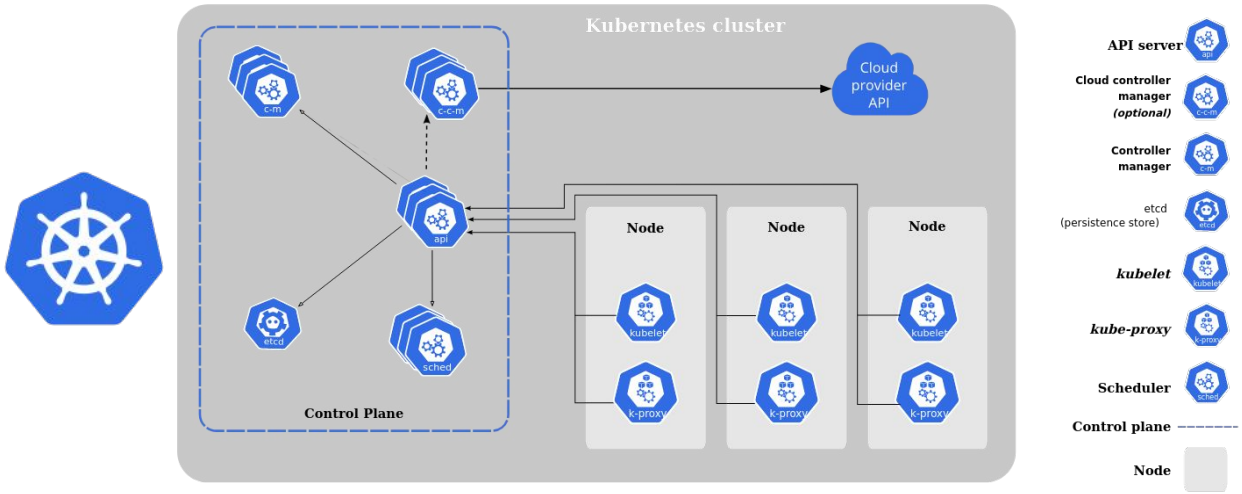
Docker is a set of platform as a service (PaaS) products that use **OS-level virtualization** to deliver software in packages called ***containers***. *Containers* are **isolated from one another** and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. **Because all of the containers share the services of a single operating system kernel, they use fewer resources than virtual machines.**

Aplicação: Docker

```
# build environment
FROM node:13-alpine as build
WORKDIR /app
ENV PATH /app/node_modules/.bin:$PATH
COPY package.json /app/package.json
RUN npm install
RUN npm install react-scripts@3.0.1 -g
COPY . /app
RUN npm run build

# production environment
FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
```

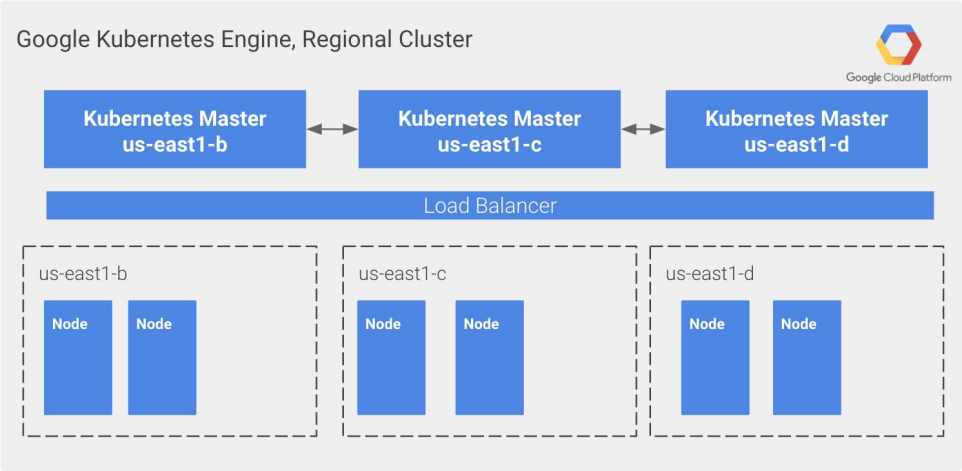
Infraestrutura: k8s

Kubernetes (k8s) is an **open-source container-orchestration system for automating computer application deployment, scaling, and management.**It was originally designed by Google and is now maintained by the Cloud Native Computing Foundation. It aims to provide a **"platform for automating deployment, scaling, and operations of database management systems"**. It works with a range of container tools and **runs containers in a cluster**, often with images built using **Docker**.



Infraestrutura: GKE

Google Kubernetes Engine (GKE) provides a **managed environment for deploying, managing, and scaling your containerized applications using Google infrastructure**. The GKE environment consists of multiple machines (specifically, **Compute Engine instances) grouped together to form a cluster**.
Features: *Identity and access management, Hybrid networking, Security and compliance, Integrated logging and monitoring, Cluster options, Auto upgrade, Auto repair...*



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  namespace: default
spec:
  progressDeadlineSeconds: 600
  replicas: 3
  selector:
    matchLabels:
      app: frontend
  strategy:
    rollingUpdate:
      maxSurge: 33%
      maxUnavailable: 33%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - image: ghcr.io/paulopacitti/titanic:main
        imagePullPolicy: IfNotPresent
        name: frontend
        resources:
          requests:
            memory: "2Mi"
            cpu: "1m"
          limits:
            memory: "256Mi"
            cpu: "250m"
        ports:
        - containerPort: 80
          protocol: TCP
```

deployment.yaml

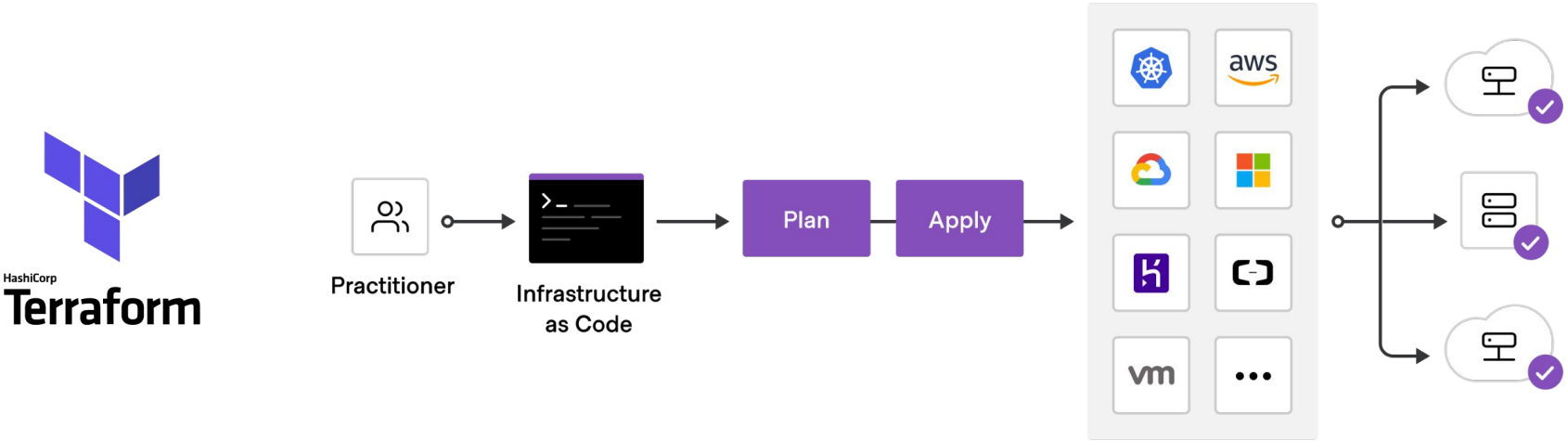
```yaml
apiVersion: v1
kind: List
items:
- apiVersion: autoscaling/v1
  kind: HorizontalPodAutoscaler
  metadata:
    name: frontend-hpa
    namespace: default
  spec:
    maxReplicas: 6
    minReplicas: 3
    scaleTargetRef:
      apiVersion: apps/v1
      kind: Deployment
      name: frontend
    targetCPUUtilizationPercentage: 80
```

hpa.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
  namespace: default
spec:
  externalTrafficPolicy: Cluster
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: frontend
  sessionAffinity: None
  type: LoadBalancer
```

service.yaml

Infraestrutura: Terraform



Terraform is an **open-source infrastructure as code** software tool created by HashiCorp. **Users define and provide data center infrastructure using a declarative configuration language** known as HashiCorp Configuration Language (HCL),

versions.tf

```
terraform {
  required_providers {
    google = {
      source  = "hashicorp/google"
      version = "3.52.0"
    }
  }

  required_version = "~> 0.14"
}
```

```
project_id = "mc714-titanic"
region     = "us-central1"
```

terraform.tfvars

outputs.tf

```
output "region" {
  value       = var.region
  description = "GCloud Region"
}

output "project_id" {
  value       = var.project_id
  description = "GCloud Project ID"
}

output "kubernetes_cluster_name" {
  value       = google_container_cluster.primary.name
  description = "GKE Cluster Name"
}

output "kubernetes_cluster_host" {
  value       = google_container_cluster.primary.endpoint
  description = "GKE Cluster Host"
}
```

vpc.hcl

- A virtual private cloud (VPC) is **an on-demand configurable pool of shared resources allocated within a public cloud environment**, providing a certain level of isolation between the different organizations (denoted as users hereafter) using the resources.

```
variable "project_id" {
  description = "project id"
}

variable "region" {
  description = "region"
}

provider "google" {
  project = var.project_id
  region  = var.region
}

# VPC
resource "google_compute_network" "vpc" {
  name                    = "${var.project_id}-vpc"
  auto_create_subnetworks = "false"
}

# Subnet
resource "google_compute_subnetwork" "subnet" {
  name          = "${var.project_id}-subnet"
  region        = var.region
  network       = google_compute_network.vpc.name
  ip_cidr_range = "10.10.0.0/24"
}
```

```
variable "gke_username" {
  default     = ""
  description = "gke username"
}

variable "gke_password" {
  default     = ""
  description = "gke password"
}

variable "gke_num_nodes" {
  default     = 1
  description = "number of gke nodes"
}

# GKE cluster
resource "google_container_cluster" "primary" {
  name     = "${var.project_id}-gke"
  location = var.region

  remove_default_node_pool = true
  initial_node_count       = 1

  network    = google_compute_network.vpc.name
  subnetwork = google_compute_subnetwork.subnet.name
}
```

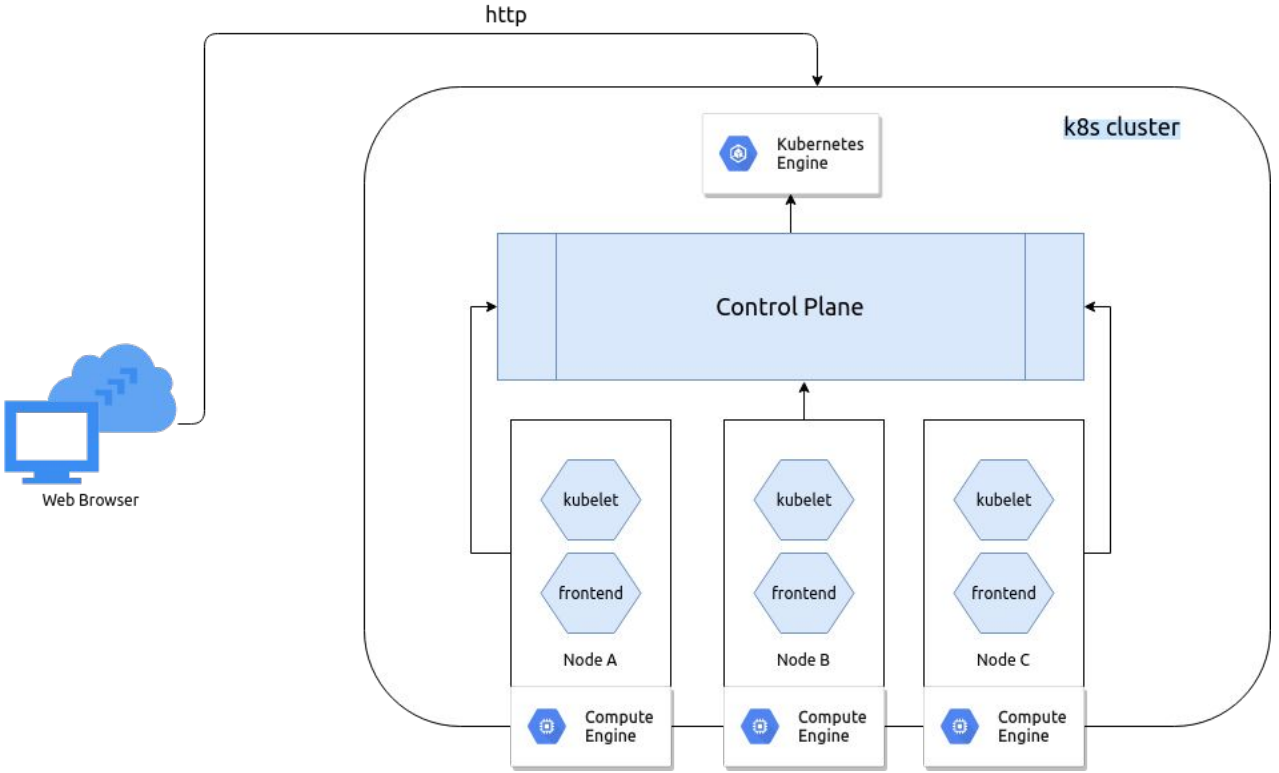
```
# Separately Managed Node Pool
resource "google_container_node_pool" "primary_nodes" {
  name       = "${google_container_cluster.primary.name}-node-pool"
  location   = var.region
  cluster    = google_container_cluster.primary.name
  node_count = var.gke_num_nodes

  node_config {
    oauth_scopes = [
      "https://www.googleapis.com/auth/logging.write",
      "https://www.googleapis.com/auth/monitoring",
    ]

    labels = {
      env = var.project_id
    }

    preemptible  = true
    machine_type = "n1-standard-1"
    tags         = ["gke-node", "${var.project_id}-gke"]
    metadata = {
      disable-legacy-endpoints = "true"
    }
  }
}
```

gke.hcl

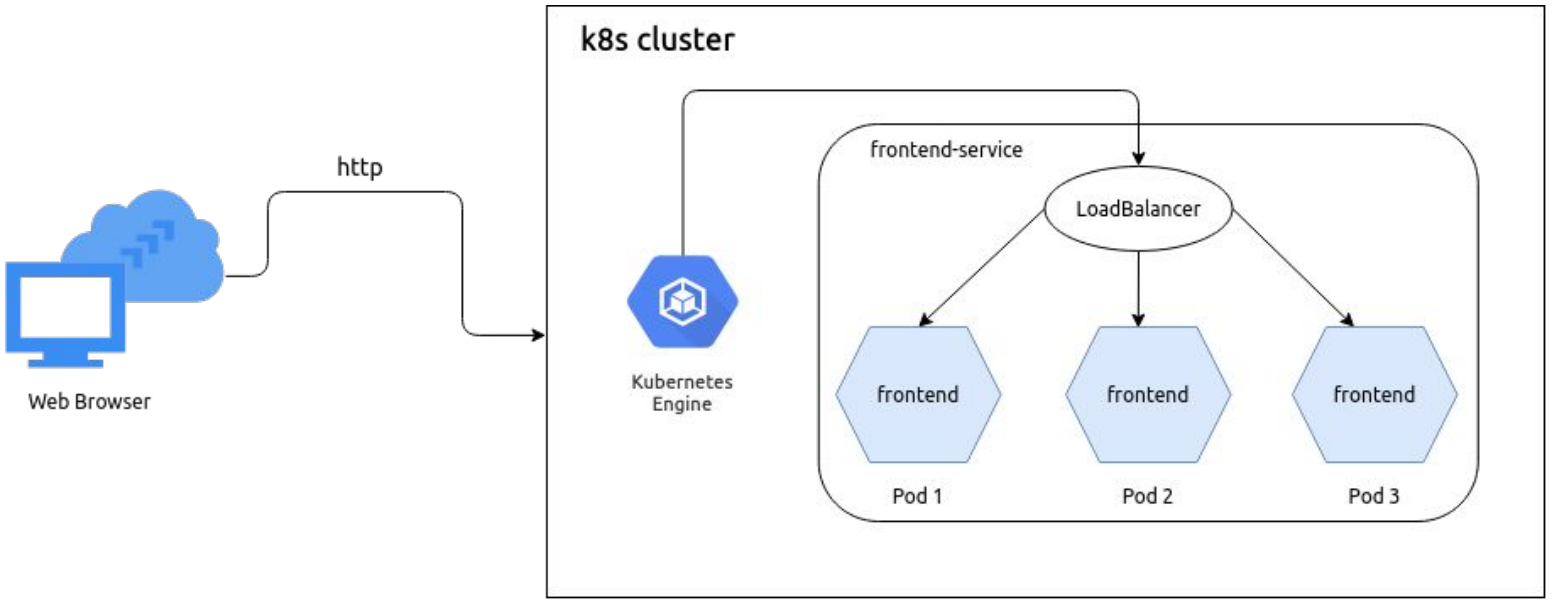
deploy_k8s.sh

```
#!/bin/bash
cd ../infra
CLUSTER_NAME=$(terraform output kubernetes_cluster_name)
REGION=$(terraform output region)

cd ../k8s
gcloud container clusters get-credentials ${CLUSTER_NAME:1:-1}
--region ${REGION:1:-1}
kubectl config set-context --current --namespace=default
kubectl apply -f .

echo "DONE!"
```

Arquitetura: física



Arquitetura: k8s



demo

Repo

github.com/paulopacitti/titanic


obrigado!