ДОДАТКИ

Додаток А. Технічне завдання

1 Введення

1.1 Найменування проекту

Найменування проекту: “Дослідження та розробка хмарної мікросервісної архітектури” (надалі ПП).

ПП складається з двох компонентів:

1. веб-сервер (надалі ВС);

2. база даних (надалі БД);

1.2 Область застосування

ПП призначений для використання у навчальному процесі з метою дослідження та моделювання роботи простої мікросервісної інфраструктури під керуванням системи оркестрації.

2 Основи для розробки

2.1 Основи для проведення розробки

Завдання на дипломний проект представлено в додатках (див. п. 8.1). Організація, що затвердила документ: кафедра комп’ютерних систем та мереж (503).

Замовник: кафедра комп’ютерних систем та мереж (503).

Виконавець: студент групи 565М Дем’яненко П.А.

Дата затвердження: 01.09.2021 р.

2.2 Найменування теми розробки та умовне позначення

Найменування теми розробки: “Розроблення та дослідження хмарних

серверних інфрастуктур з використанням механізмів контейнеризації та оркестрації за допомогою системи Kubernetes”.

3 Призначення розробки

3.1 Функціональне призначення програми

ПП призначений для використання у навчальному процесі з метою дослідження та моделювання роботи простої мікросервісної інфраструктури під керуванням Kubernetes на Google Cloud Platform.

ВС повинен відповідати на HTTP запити та віддавати тестову веб-сторінку з веб-сервером адресної книги. БД повинна вміщувати дані змінної лічильника. ВС та БД не мають людино-машинного інтерфейсу але повинні вести лог подій.

3.2 Експлуатаційне призначення програми

Режим роботи ВС та БД– довгострокове функціювання, виконання своїх функцій та перезавантаження.Кліматичні вимоги до експлуатації визначаються вимогами до апаратного забезпечення. ВС має бути скомпільовано та зібран у Docker контейнер.

4 Вимоги до програми або програмного виробу

4.1 Вимогу до функціональних характеристик

4.1.1 Вимоги до складу виконавчих функцій

Для вирішення завдання, ВС повинне вирішувати наступні задачі:

1) відповідати вхідні з’єднання з запитами на показ веб-сторінки;

2) використовувати БД для зберігання телефонів та імен.

Для вирішення задачі 1) програма повинна виконувати наступні функції:

a) прослуховувати вхідні запити на порту для усіх мережевих інтерфейсів;

b) формувати код сторінки відповіді.

Для вирішення задачі 2) програма повинна виконувати наступні функції:

a) записувати дані до БД.

b) вибирати усі дані з БД;

4.1.2 Вимоги до організації вхідних даних

Вхідні дані формуються автоматично під час встановлення підсистеми Веб-серверу адресної книги.

4.1.3 Вимоги до організації вихідних даних

Вихідні дані ВС – веб-сторінка у форматі HTML.

4.2 Вимоги до складу та параметрів технічного забезпечення

Вимоги до апаратного забезпечення для виконання ВС:

1. ЦП з частотою, не менше – 2.1 ГГц;
2. RAM, не менше – 4 ГБ;
3. 40 ГБ простору на жорсткому диску.

Вимоги до апаратно забезпечення для запуску та виконання БД визначаються конкретними вимогами обраної СУБД

4.3 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення, необхідне для функціонування ВС:

– операційна система Ubuntu;

– ядро версії 18.04 20.04 або пізніше;

– телетайп термінал.

Додаткове програмне забезпечення:

– Docker container engine версії 19.03.8 або пізніше;

Вимоги до програмного забезпечення для запуску та виконання БД визначаються конкретними вимогами обраної СУБД.

5 Вимоги до організації серверної інфраструктури

Серверна інфраструктура має складатися щонайменше з 1 серверу.

Вимоги до програмного забезпечення хоста:

– операційна система CentOS;

– ядро версії 3.10.0-1127 або пізніше;

– openssh-server версії 7.4 або пізніше;

– телетайп термінал.

Вимоги до організації доступу до сервісів у межах кластера:

– ВС має бути доступним для виконання запитів із мережі Інтернет;

– БД має бути недоступна для підключення з пристроїв, що не входять до складу кластеру та серверної інфраструктури на якій виконується база даних;

6 Вимоги до програмної документації

Склад програмної документації:

1. технічне завдання;

2. код ВС, скриптів створення та налаштування серверної інфраструктури;

3. презентація;

4. пояснювальна записка.

7 Стадії та етапи розробки

Повинні бути реалізовані такі стадії:

1. постановка задачі;

2. проектування архітектури системи;

3. проектування серверної інфраструктури;

4. розробка серверної інфраструктури;

5. розробка Terraform сриптів;

6. розробка CirlceCI скрипту;

7. розробка Helm чартів;

8. тестування.

8 Порядок контролю та приймання

Приймає керівник дипломного проекту у вигляді публічного захисту 16.12.2021 р.

Файл /gcs_bucket/main.tf

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

resource "google_storage_bucket" "bucket" {
  name    = var.name
  project = var.project_id

  versioning {
    enabled = var.versioning
  }
}
variable "project_id" {
  description = "Google Cloud Platform (GCP) Project ID."
  type        = string
  default     = "sunlit-inn-329918"
}

variable "region" {
  description = "GCP region name."
  type        = string
  default     = "europe-west3"
}

variable "zone" {
```

```
  description = "GCP zone name."
  type        = string
  default     = "europe-west3-c"
}

variable "name" {
  description = "The name of the bucket."
  type        = string
  default     = "task-busket222111-tf1"
}

variable "versioning" {
  description = "While set to true, versioning is fully enabled for this bucket."
  type        = bool
  default     = true
}
```

Файл /Conf/main.tf

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

resource "google_container_registry" "registry" {
  project     = var.project_id
  location    = "EU"
}
```

```
resource "random_id" "suffix" {
  byte_length = 4
}

module "init" {
  source     = "./modules/init"
  project_id = var.project_id
  image_name = var.image_name
  tag        = var.tag
  depends_on = [google_container_registry.registry]
}

module "network" {
  source          = "./modules/network/"
  region          = var.region
  project_id      = var.project_id
  network_name    = "${var.project_name}-vpc-${random_id.suffix.hex}-
${terraform.workspace}"
  global_name     = "${var.network_global_address_name}-
${random_id.suffix.hex}-${terraform.workspace}"
  subnetwork_name = "${var.project_name}-sub-${random_id.suffix.hex}-
${terraform.workspace}"
}

module "sql" {
  source          = "./modules/sql"
  region          = var.region
  db_instance_name = "${var.project_name}-db-${random_id.suffix.hex}-
${terraform.workspace}"
  db_tier         = var.db_tier
```

```
  private_network  = module.network.network_id
  db_root_password = var.db_root_password
  db_user          = var.db_user
  db_user_password = var.db_user_password
  depends_on       = [module.network]
 }

 module "cluster" {
  source               = "./modules/cluster/"
  region               = var.region
  project_id           = var.project_id
  cluster_name         = "${var.project_name}-cluster-
${random_id.suffix.hex}-${terraform.workspace}"
  cluster_nodepool_name = "${var.project_name}-nodepool-
${random_id.suffix.hex}-${terraform.workspace}"
  network              = module.network.network_name
  subnetwork           = module.network.subnetwork_name
  machine_type         = var.node_machine_type
  depends_on           = [module.sql, module.network]
 }


 module "kubernetes" {
  source               = "./modules/kuber"
  endpoint             = module.cluster.public_endpoint
  cluster_ca_certificate =
base64decode(module.cluster.cluster_ca_certificate)
  secret_name          = var.kubernetes_secret_name
  db_host              = module.sql.db_private_ip_address
  db_user              = var.db_user
```

```
  db_user_password       = var.db_user_password
  db_name                = var.db_name
 }

 module "helm" {
  source                 = "./modules/helm"
  endpoint               = module.cluster.public_endpoint
  cluster_ca_certificate =
base64decode(module.cluster.cluster_ca_certificate)
  helm_name              = "${module.cluster.cluster_name}"
  chart_path             = var.gh_token
 }
 variable "project_id" {
  description = "Google Cloud Platform (GCP) Project ID."
  type        = string
  default     = "sunlit-inn-329918"
 }

 variable "region" {
  description = "GCP region name."
  type        = string
  default     = "europe-west3"
 }

 variable "zone" {
  description = "GCP zone name."
  type        = string
  default     = "europe-west3-c"
 }
```

```
variable "project_name" {
  description = "Google Cloud Platform (GCP) Project name."
  type        = string
  default     = "my-first"
}

variable "network_global_address_name" {
  description = "Address name for private network."
  type        = string
  default     = "private-ip-address"
}


variable "db_name" {
  description = "Name for database."
  type        = string
  default     = "web-db"
}

variable "db_region" {
  description = "Region for database."
  type        = string
  default     = "europe-central2"
}

variable "db_tier" {
  description = "Tier for database."
  type        = string
  default     = "db-f1-micro"
}
```

```
variable "db_root_password" {
}

variable "db_user" {
}

variable "db_user_password" {
}


variable "node_machine_type"{
  description = "Machine type for kubernetes node."
  type        = string
  default     = "e2-micro"
}


variable "image_name"{
  description = "Initial image name for docker."
  type        = string
  default     = "address-book"
}

variable "tag"{
  description = "Initial tag for docker image."
  type        = string
  default     = "init"
}
```

```
variable "kubernetes_secret_name" {
  description = "Name for kubernetes secrets."
  type        = string
  default     = "db-secrets"
}


variable "chart_path" {
  description = "Path for helm chart."
  type        = string
  default     = ""
}

variable "gh_token"{
}
```

Файл /Conf/modules/cluster/main.tf

```
resource "google_container_cluster" "test" {
  name     = var.cluster_name
  location = var.region

  remove_default_node_pool = true
  initial_node_count       = 1

  network         = var.network
  subnetwork      = var.subnetwork
  networking_mode = "VPC_NATIVE"
```

```
  ip_allocation_policy {
    cluster_ipv4_cidr_block  = "/20"
    services_ipv4_cidr_block = "/20"
  }
}

resource "google_container_node_pool" "test_nodes" {
  name       = var.cluster_nodepool_name
  location   = var.region
  cluster    = google_container_cluster.test.name
  node_count = 1

  node_config {
    preemptible  = true
    oauth_scopes = [
    "https://www.googleapis.com/auth/compute",
    "https://www.googleapis.com/auth/devstorage.read_only",
    "https://www.googleapis.com/auth/logging.write",
    "https://www.googleapis.com/auth/monitoring"
   ]
    machine_type = var.machine_type
    tags         = ["gke-node"]
    metadata = {
      disable-legacy-endpoints = "true"
    }
  }
}
variable "project_id"{

}
```

```
variable "cluster_name" {

}

variable "cluster_nodepool_name" {

}

variable "region" {

}

variable "network" {

}

variable "subnetwork"{

}

variable "machine_type"{

}
output "cluster_name"{
    value = google_container_cluster.test.name
}

output "client_certificate"{
    value = google_container_cluster.test.master_auth.0.client_certificate
```

```
}

output "client_key"{
   value = google_container_cluster.test.master_auth.0.client_key
}

output "cluster_ca_certificate"{
   value = google_container_cluster.test.master_auth.0.cluster_ca_certificate
}

output "public_endpoint"{
   value = google_container_cluster.test.endpoint
}
```

Файл /Conf/modules/helm/main.tf

```
data "google_client_config" "client" {

}

data "template_file" "access_token" {
 template = data.google_client_config.client.access_token
}

provider "helm"{
 kubernetes{
   host                  = "https://${var.endpoint}"
   token                  = data.template_file.access_token.rendered
   cluster_ca_certificate = var.cluster_ca_certificate
 }
```

```
}

resource "helm_release" "my-first1"{
  name       =  "my-first"
  repository =  var.chart_path
  chart      =  "my-first"
}
variable "endpoint" {

}
variable "cluster_ca_certificate" {

}

variable "helm_name" {

}

variable "chart_path" {

}
```

Файл /Conf/modules/init/main.tf

```
resource "null_resource" "docker" {
  provisioner "local-exec" {
    command = " docker build -t
eu.gcr.io/${var.project_id}/${var.image_name}/${terraform.workspace}:${var.tag
} /home/pademi/address-book/ |gcloud auth print-access-token | docker login -u
oauth2accesstoken --password-stdin eu.gcr.io | docker push
```

```
eu.gcr.io/${var.project_id}/${var.image_name}/${terraform.workspace}:${var.tag
}"
          }
        }
        variable "project_id" {

        }

        variable "image_name"{

        }

        variable "tag"{

        }
```

Файл /Conf/modules/kuber/main.tf

```
        data "google_client_config" "client" {}

        data "template_file" "access_token" {
          template = data.google_client_config.client.access_token
        }

        provider "kubernetes" {
          host = "https://${var.endpoint}"
            token                  = data.google_client_config.client.access_token
            cluster_ca_certificate = var.cluster_ca_certificate
        }
```

```
resource "kubernetes_secret" "db_data" {
  metadata {
    name = var.secret_name
  }

  data = {
    db_host = "${var.db_host}"
    db_user = "${var.db_user}"
    db_pass = "${var.db_user_password}"
    db_name = "${var.db_name}"
  }
}

variable "endpoint" {

}
variable "secret_name" {

}
variable "db_host" {

}
variable "db_user" {

}
variable "db_user_password" {

}
variable "db_name" {
```

```
}
variable "cluster_ca_certificate" {

}
variable "endpoint" {

}
variable "secret_name" {

}
variable "db_host" {

}
variable "db_user" {

}
variable "db_user_password" {

}
variable "db_name" {

}
variable "cluster_ca_certificate" {

}
```

Файл /Conf/modules/network/main.tf

```
resource "google_compute_network" "network" {
  name                    = var.network_name
```

```
  project                 = var.project_id
  auto_create_subnetworks = "false"
 }

 resource "google_compute_global_address" "private_ip_address" {
  name          = var.global_name
  purpose       = "VPC_PEERING"
  address_type  = "INTERNAL"
  prefix_length = 16
  network       = google_compute_network.network.id
 }

 resource "google_service_networking_connection"
"private_vpc_connection" {
  network                 = google_compute_network.network.self_link
  service                 = "servicenetworking.googleapis.com"
  reserved_peering_ranges =
[google_compute_global_address.private_ip_address.name]
 }
 resource "google_compute_subnetwork" "subnet" {
  name          = var.subnetwork_name
  region        = var.region
  network       = google_compute_network.network.name
  ip_cidr_range = "10.10.0.0/24"
  depends_on    = [google_compute_network.network]
 }
 variable "project_id"{

 }
```

```
variable "region"{

}

variable "network_name" {

}

variable "global_name" {

}

variable "subnetwork_name" {

}
output "network_id"{
  value = google_compute_network.network.id
}

output "network_name"{
  value = google_compute_network.network.name
}

output "subnetwork_name" {
  value = google_compute_subnetwork.subnet.name
}
```

Файл /Conf/modules/sql/main.tf

```
resource "google_sql_database_instance" "sql" {
```

```
  region           = var.region
  database_version = "MYSQL_5_6"
  name             = var.db_instance_name

  settings {
    tier = var.db_tier

    ip_configuration {
      ipv4_enabled    = false
      private_network = var.private_network
    }
  }
  deletion_protection = false
}

resource "google_sql_user" "master" {
  name     = "root"
  host     = "%"
  password = var.db_root_password
  instance = google_sql_database_instance.sql.name
}

resource "google_sql_user" "sql_user" {
  name     = var.db_user
  host     = "%"
  password = var.db_user_password
  instance = google_sql_database_instance.sql.name
}
variable "region"{
```

```
}

variable "db_instance_name" {

}

variable "private_network"{

}

variable "db_tier" {

}

variable "db_root_password" {

}

variable "db_user" {

}

variable "db_user_password" {

}
output "db_private_ip_address" {
  value    = google_sql_database_instance.sql.private_ip_address
}
```

Додаток В. Вихідні скрипти CirlceCI

Файл /.circleci/config.yml

```
version: 2
jobs:
  push:
    docker:
     - image: google/cloud-sdk
    steps:
     - checkout
     - setup_remote_docker
     - run:
         name: push in gcr
         command: |
           TAG=${CIRCLE_SHA1::7}
           docker build -t address-book .
           docker tag address-book:latest
eu.gcr.io/${GOOGLE_PROJECT_ID}/address-book/dev:${CIRCLE_SHA1::7}
           echo "$GCLOUD_ACC" | base64 -d | docker login -u _json_key --
password-stdin https://eu.gcr.io
           docker push eu.gcr.io/${GOOGLE_PROJECT_ID}/address-
book/dev:${CIRCLE_SHA1::7}
  deploy:
    docker:
     - image: google/cloud-sdk
    steps:
     - setup_remote_docker
     - run:
```

```
            name: simple deployment
            command: |
              echo "$GCLOUD_ACC" | base64 --d > gcloud-service-key.json
              gcloud auth activate-service-account --key-file gcloud-service-
key.json
              gcloud --quiet config set project $GOOGLE_PROJECT_ID
              gcloud --quiet config set compute/zone
$GOOGLE_COMPUTE_ZONE
              gcloud container clusters get-credentials my-first-cluster-478590a8-
dev --region europe-west3
              curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-
helm-3 | bash
              helm repo add ${GCP_PROJECT_NAME}
"https://raw.githubusercontent.com/circletes/helm/dev"
              helm repo update
              helm upgrade --install ${GCP_PROJECT_NAME}
${GCP_PROJECT_NAME}/my-first --set=container.tag=${CIRCLE_SHA1::7}
        workflows:
          version: 2
          build_push_and_deploy:
            jobs:
              - push:
                  context:
                    - gcp
              - deploy:
                  context:
                    - gcp
                  requires:
                    - push
```

Додаток Г. Вихідні скрипти Helm-Charts

Файл /helm/helm-chart/templates/deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}
  labels:
    app: {{ .Release.Name }}-app
spec:
  replicas: {{ .Values.replicaCount}}
  selector:
    matchLabels:
      project: {{ .Release.Name }}
  template:
    metadata:
      labels:
        project: {{ .Release.Name }}
    spec:
      containers:
        - name : {{ .Release.Name }}-web
          image: {{ .Values.container.image}}:{{ .Values.container.tag}}
```

Файл /helm/helm-chart/templates/configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
```

```
  namespace: {{ .Release.Namespace | default "default" | lower }}
  labels:
    app: {{ .Release.Name }}-app
data:
  {{- toYaml .Values.configMap | nindent 2 }}
```

Файл /helm/helm-chart/templates/ ingress.yaml

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: {{ .Release.Name }}-ingress
  namespace: {{ .Release.Namespace | default "default" | lower }}
spec:
  backend:
    serviceName: {{ .Release.Name }}-service
    servicePort: {{ .Values.ingress.servicePort }}
```

Файл /helm/helm-chart/templates/ service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-service
  labels:
    env  : {{ .Values.environment}}
spec:
  selector:
    project: {{ .Release.Name }}
  ports:
```

```
    - name      : listener
      protocol  : TCP
      port      : {{ .Values.service.port }}
      targetPort: {{ .Values.service.targetPort }}
  type: LoadBalancer
```

Файл /helm/helm-chart/values.yaml

```
replicaCount: 3

container:
  image: eu.gcr.io/citric-earth-319117/address-book/dev
  tag: init
environment: dev

service:
  port: 8080
  targetport: 8080

ingress:
  servicePort: 8080
```

Файл /helm/helm-chart/Chart.yaml

```
apiVersion: v2
name: my-first
description: Base web app deployment.
type: application
version: 2
```

Файл /helm/helm-chart/index.yaml

```
apiVersion: v1
entries:
  my-first:
  - apiVersion: v2
    created: "2021-09-09T21:54:56.952487456+03:00"
    description: Base web app deployment.
    digest:
76c3be0fc8bde2d1ce4139e0cba480df330b627eb08014cd40b5941af3f937a9
    name: my-first
    type: application
    urls:
    - my-first-2.tgz
    version: "2"
generated: "2021-09-09T21:54:56.951691424+03:00"
```

Додаток Д. Презентація

Нейромережева ідентифікація нестаціонарних об’єктів

Група: КІТм-21-2
Студент: Волков А.О.
Керівник:.Руденко О.Г.

Рисунок Г. 1 – Слайд 1.

ЗМІСТ ДОПОВІДІ

- Історія та сучасний стан методології DevOps
- Аналіз хмарних провайдерів
- Постановка задачі
- Проектування інфраструктури
- Розроблення інфраструктури
- Висновки

2

Рисунок Г. 2 – Слайд 2.

Рисунок Г. 3 – Слайд 3.

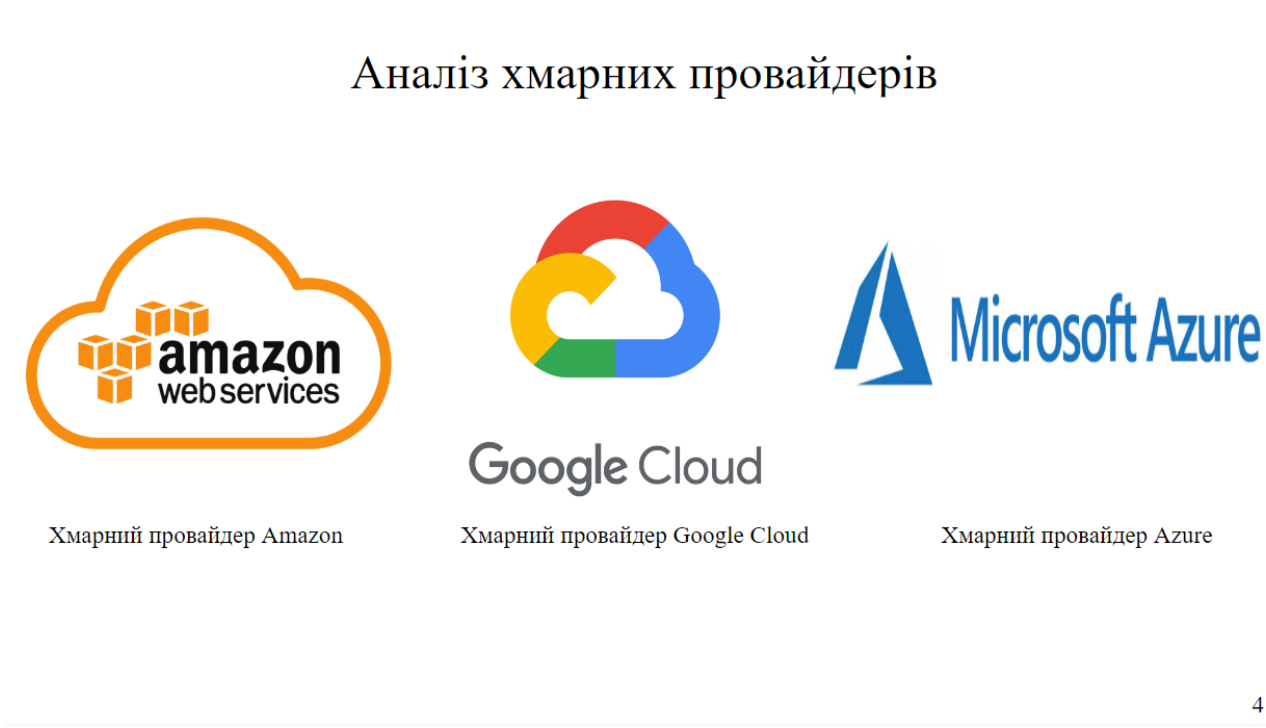


Рисунок Г. 4 – Слайд 4.

Постановка задачі

Ціллю розробки є вивчення інструментів розробки хмарної мікросервісної інфраструктури для дослідження систем нейроідентифікації нелінійних об'єктів за допомогою оркестратору контейнерів Kubernetes для обслуговування веб-серверу адресної книги.

Поставлена мета досягається вирішенням таких завдань :

– написання коду інфраструктури за допомогою інструменту Terraform до Google Cloud Platform;

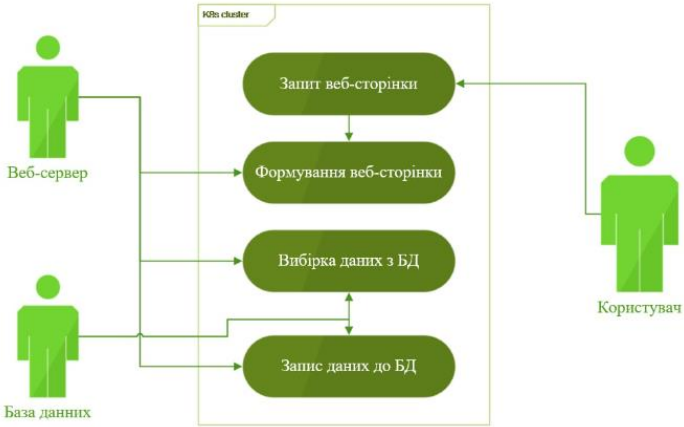
– написання коду до CI системи CircleCI;

– написання коду шаблонів для інструмента Helm.

5

Рисунок Г. 5 – Слайд 5.

Проектування інфраструктури



Діаграма варіантів використання

6

Рисунок Г. 6 – Слайд 6.

Проектування інфраструктури

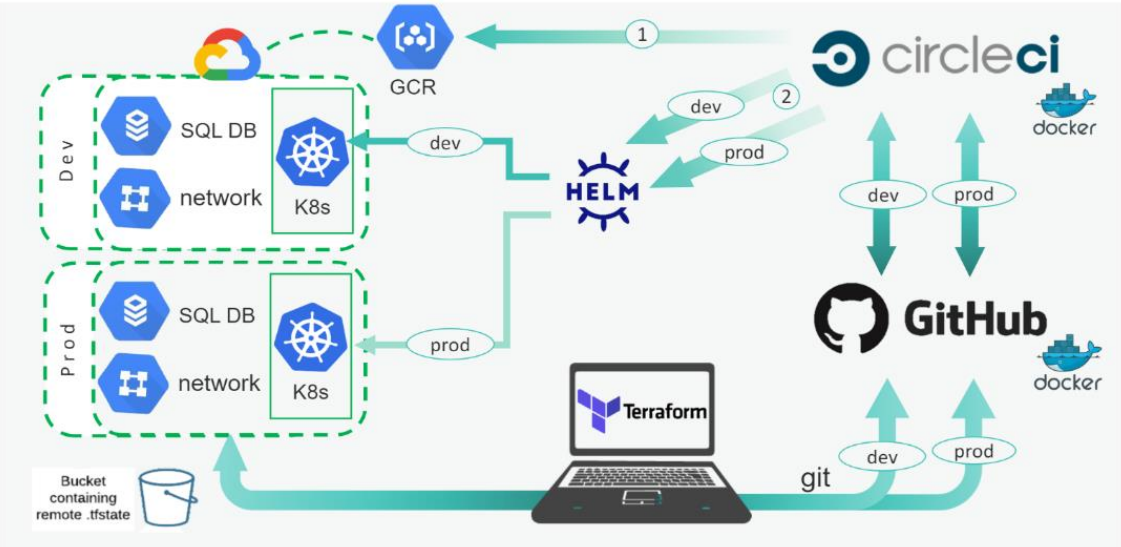
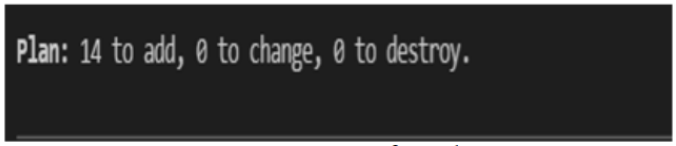


Схема даного проекту

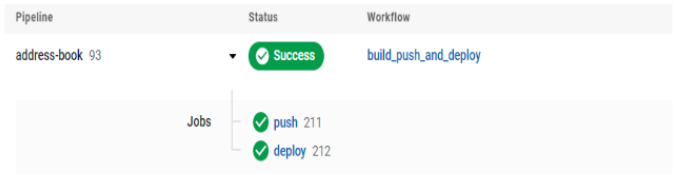
7

Рисунок Г. 7 – Слайд 7 .

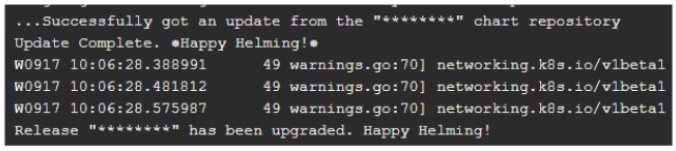
Розроблення інфраструктури



Результат команди terraform plan



Результат роботи CircleCI



Результат ревізії Helm-Charts

8

Рисунок Г. 8 – Слайд 8.

Висновки

- Розробка даних систем є актуальною;
- Розроблена система відповідає постановці завдання;
- Реалізація набагато дешевша ніж її аналоги;
- Трудомісткость робіт складає 51 день;
- Перелік робіт проекту подано у вигляді діаграми Ганта;
- Розраховано кошторис витрат за проектом, який складає 109221,77 грн.

10

Рисунок Г. 9 – Слайд 9.