

# Отчет

О проделанной работе финального задания

**Подготовил:**

Студент курса DevOps Upgrade 4

Казаков Е.С. (s053278)

Санкт-Петербург, 2023 г.

<b>1 Введение.....</b>	<b>3</b>
1.1 Описание проекта.....	3
1.2 Предоставление доступа.....	3
1.3 Обратная связь.....	4
<b>2 Создание инфраструктуры.....</b>	<b>5</b>
2.1 Настройка Gitlab.....	5
2.2 Настройка gitlab-runner.....	6
2.3 Регистрация домена.....	6
2.4 Описание инфраструктуры.....	7
<b>3 Работа с приложением.....</b>	<b>8</b>
3.1 Подготовка образов Docker.....	8
3.2 Создание Helm чарта.....	8
3.3 Подготовка БД.....	8
3.4 Настройка CI/CD.....	9
3.5 Проверка работы приложения.....	10
<b>4 Заключение.....</b>	<b>13</b>
<b>ПРИЛОЖЕНИЕ А.....</b>	<b>14</b>
<b>ПРИЛОЖЕНИЕ Б.....</b>	<b>22</b>
<b>ПРИЛОЖЕНИЕ В.....</b>	<b>25</b>
<b>ПРИЛОЖЕНИЕ Г.....</b>	<b>30</b>

# 1 Введение

## 1.1 Описание проекта

Цель проекта состоит в построении инфраструктуры в облаке Yandex Cloud, настройке управления инфраструктурой через Terraform и деплой приложения из репозитория в кластер Kubernetes.

В качестве платформы используется Yandex Cloud. В качестве хранилища кода используется managed Gitlab от Yandex Cloud.

Задача разбита на несколько этапов:

- подготовка инфраструктуры:
  - репозиторий кода;
  - настройка раннера;
  - подготовка terraform;
- деплой приложения:
  - запуск на локальном стенде;
  - создание helm чарт;
  - настройка конвейера;
  - проверка работоспособности.

## 1.2 Предоставление доступа

Создан дополнительный пользователь *@slurm-io*. Приглашение выслано на почту: [support@slurm.io](mailto:support@slurm.io). В таблице 1 отображены учетные данные от пользователя **s053278** (по просьбе куратора).





Members 2				
<div><div><div></div></div><div>Filter members</div><div>Q</div><div>Account ▾</div><div>⌵</div></div>				
Account	Source	Max role	Expiration	Activity
 <div>s053278 It's you @s053278</div>	Direct member by s053278	Owner	Expiration date 	User created: Jul 08, 2023 Access granted: Jul 08, 2023 Last activity: Jul 17, 2023 ⋮
 <div>slurm-io @slurm-io</div>	Direct member by s053278	Maintainer ▾	Expiration date 	User created: Jul 17, 2023 Access granted: Jul 17, 2023 ⋮

Рисунок 1 – Доступ к Gitlab

Отправлено приглашение в облачном провайдере, настроена роль **resource-manager.clouds.owner**.

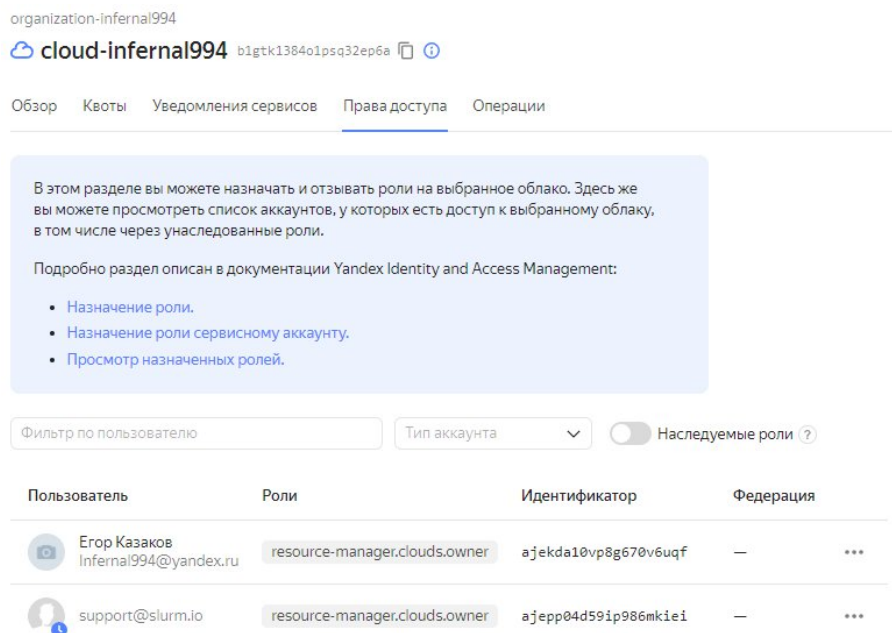


Рисунок 2 – Доступ к Yandex Cloud

Таблица 1 – Учетные данные

Ресурс	URL	Логин	Пароль
Gitlab	https://s053278.gitlab.yandexcloud.net/	s053278	***
Yandex Cloud	https://console.cloud.yandex.ru/		
Yelb (ingress)	https://yelb.s053278.ru/	—	—

### 1.3 Обратная связь

При возникновении вопросов просьба связаться со мной.

- email: \*\*\*
- telegram: \*\*\*

## 2 Создание инфраструктуры

### 2.1 Настройка Gitlab

В качестве репозитория используется *Managed Service for GitLab* созданный в личном кабинете облачного провайдера <https://console.cloud.yandex.ru/>.

Настроен пользователь и создан проект **Terraform**.

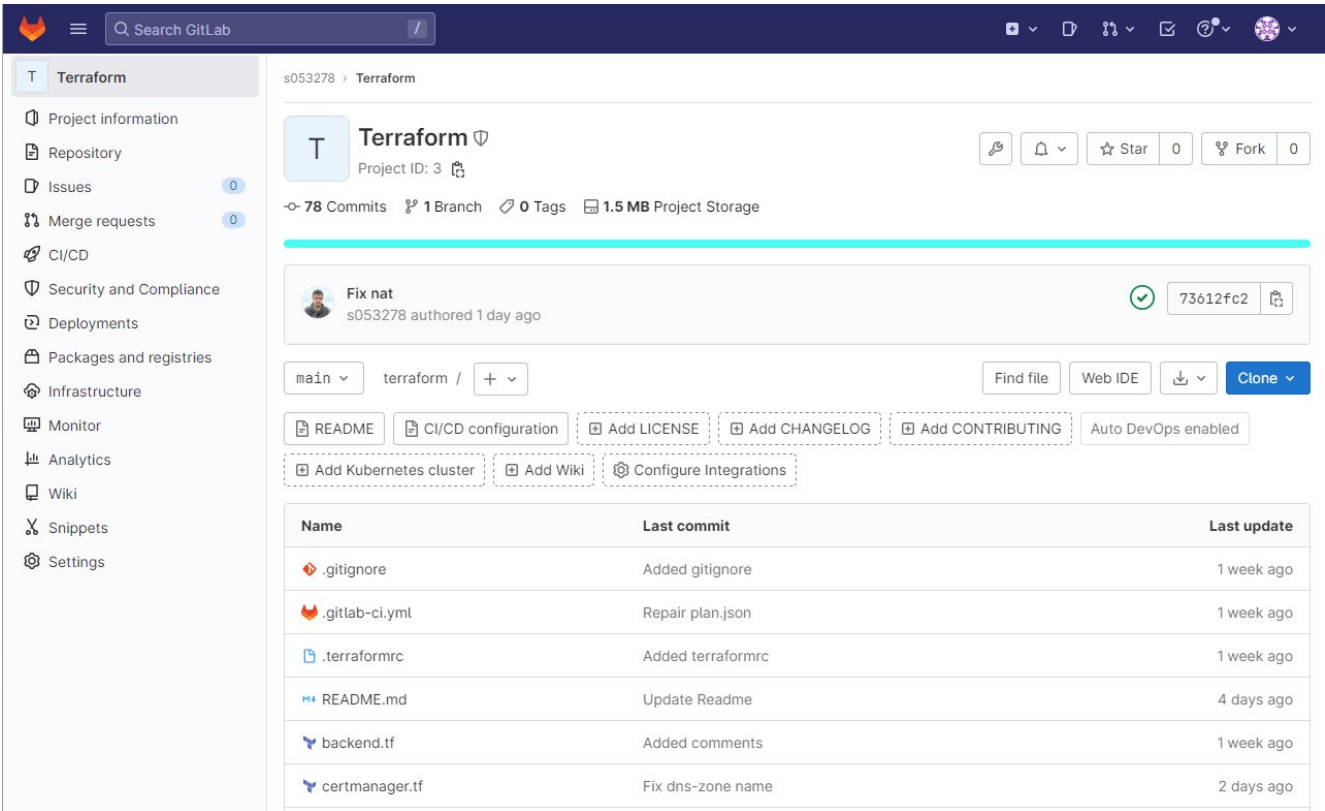


Рисунок 3 – Проект Terraform

Переменные для работы с облачным провайдером настраиваются в проекте в разделе **Settings – CI/CD – Variables**:

Type	↑ Key	Value	Options	Environments
Variable	TF_VAR_YC_FOLDER_ID	*****	Protected, Expanded	All (default)
Variable	YC_CLOUD_ID	*****	Protected, Masked	All (default)
Variable	YC_FOLDER_ID	*****	Protected	All (default)
Variable	YC_TOKEN	*****	Protected, Masked	All (default)

Рисунок 4 – Переменные проекта

## 2.2 Настройка gitlab-runner

Настроен gitlab-runner в контейнере в режиме docker-executor. Для устранения ошибок, связанных с доступом к сокету:

- настроен bind mount `/var/run/docker.sock:/var/run/docker.sock`;
- изменен конфигурационный файл **config.toml**.

Пример **config.toml**:

```
concurrent = 1
check_interval = 0
shutdown_timeout = 0

[session_server]
  session_timeout = 1800

[[runners]]
  name = "docker-runner"
  url = "https://s053278.gitlab.yandexcloud.net/"
  id = 1
  token = "***"
  token_obtained_at = 2023-07-08T12:21:13Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "docker"
  [runners.cache]
    MaxUploadedArchiveSize = 0
  [runners.docker]
    tls_verify = false
    image = "docker:20.10.16"
    privileged = false
    disable_entrypoint_overwrite = false
    oom_kill_disable = false
    disable_cache = false
    volumes = ["/var/run/docker.sock:/var/run/docker.sock", "/cache"]
    shm_size = 0
```

Раннер настроен вне облачного провайдера, но при желании можно развернуть как Compute Cloud.

## 2.3 Регистрация домена

Приобретен домен **s053278.ru** и изменено делегирование на сервера Yandex Cloud:

- ns1.yandexcloud.net
- ns2.yandexcloud.net

Это позволит управлять DNS записями (A, AAAA, CNAME и пр.) из личного кабинета Yandex Cloud.

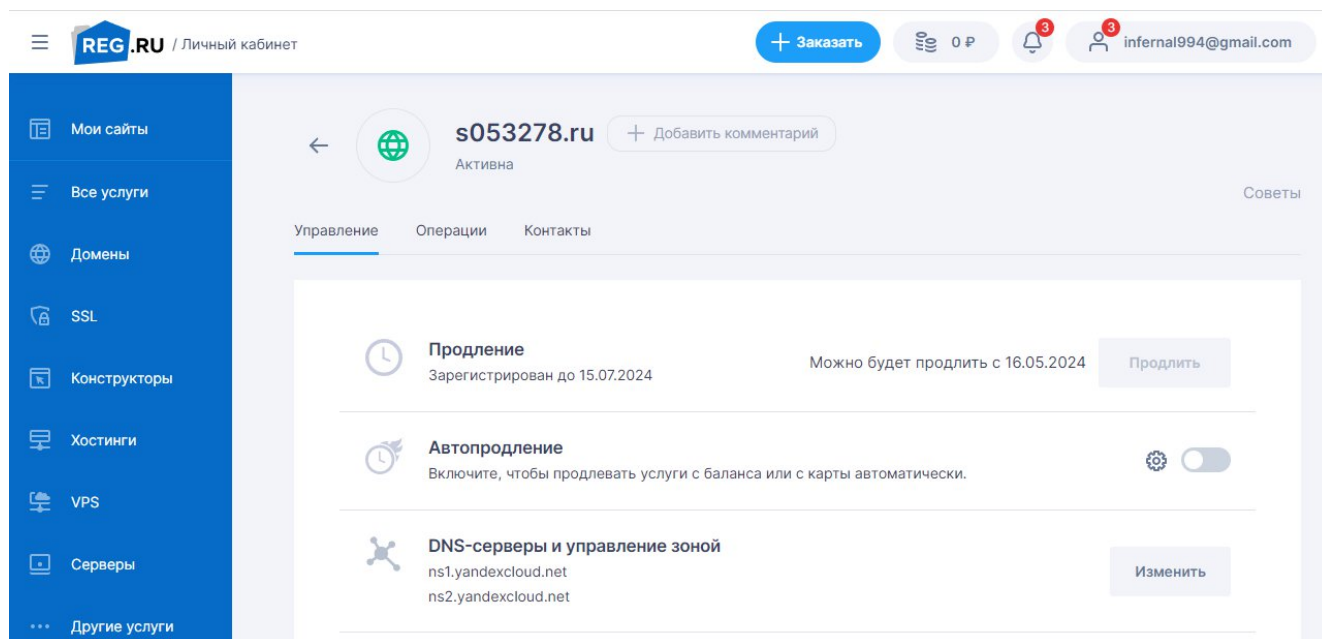


Рисунок 5 – Домен и управление зоной

## 2.4 Описание инфраструктуры

В проекте Terraform описаны следующие структуры:

- **provider.tf** – настройки облачного провайдера;
- **backend.tf** – заглушка удаленного хранения *tfstate*;
- **networks.tf** – описание сети, подсетей и DNS-зоны;
- **serviceaccount.tf** – сервисный аккаунт для *kubernetes*;
- **kmsprovider.tf** – закрытый симметричный ключ;
- **kubernetes.tf** – кластер *kubernetes* и его ноды;
- **database.tf** – кластер PostgreSQL, создание пользователя и базы;
- **variables.tf** – переменные конфигурации terraform.

Подробно с описанием инфраструктуры можно ознакомиться в Приложении А.

Для работы CI/CD подготовлен конфигурационный файл **.gitlab-ci.yml** со следующими этапами:

- 1) **validate** – этап валидации проекта terraform;
- 2) **plan** – этап планирования;
- 3) **deploy** – этап применения конфигурации (*можно добавить manual*);
- 4) **destroy** – этап удаления инфраструктуры (*manual*).

Ознакомиться с этапами подробно можно в Листинг А.10 (Приложение А).

## 3 Работа с приложением

### 3.1 Подготовка образов Docker

Для проверки работоспособности микросервисного приложения был развернут локальный стенд. Обнаружены следующие проблемы:

- порт базы данных был жестко зафиксирован в коде 5432/TCP;
- проблемы с зависимостями при сборке образов;
- отсутствовали требуемые переменные окружения для деплоя в кластере kubernetes.

Для устранения перечисленных проблем были модифицированы Dockerfile для образов yelb-ui и yelb-appserver и некоторые конфигурационные файлы. Подробно о них описано в Приложении Б.

### 3.2 Создание Helm чарта

Подготовлен Helm чарт для деплоя в кластер Kubernetes. Чарт имеет следующую структуру:

- **.helm\Chart.yaml** – описание чарта
- **.helm\values.yaml** – переменные чарта
- **.helm\templates\ui.yml** - деплоймент yelb-ui
- **.helm\templates\appserver.yml** – деплоймент yelb-appserver
- **.helm\templates\redis.yml** - деплоймент redis
- **.helm\templates\service.yml** – сервисы взаимодействия
- **.helm\templates\ingress.yml** – внешний сервис/ингресс
- **.helm\templates\NOTES.txt** – заметка после успешной установки

Подробнее с чартом можно ознакомиться в Приложении В.

### 3.3 Подготовка БД

Перед первым деплоем необходимо создать таблицу и добавить несколько строк.

```
CREATE TABLE restaurants (  
  name      char(30),  
  count     integer,  
  PRIMARY KEY (name)  
);  
INSERT INTO restaurants (name, count) VALUES ('outback', 0);  
INSERT INTO restaurants (name, count) VALUES ('bucadibeppe', 0);  
INSERT INTO restaurants (name, count) VALUES ('chipotle', 0);  
INSERT INTO restaurants (name, count) VALUES ('ihop', 0);
```



Удобно сделать это в web-интерфейсе облачного провайдера. Для этого перейдите в раздел **Managed Service for PostgreSQL**, войдите в созданный кластер и откройте вкладку **SQL** (авторизация по-умолчанию: dbuser/password).

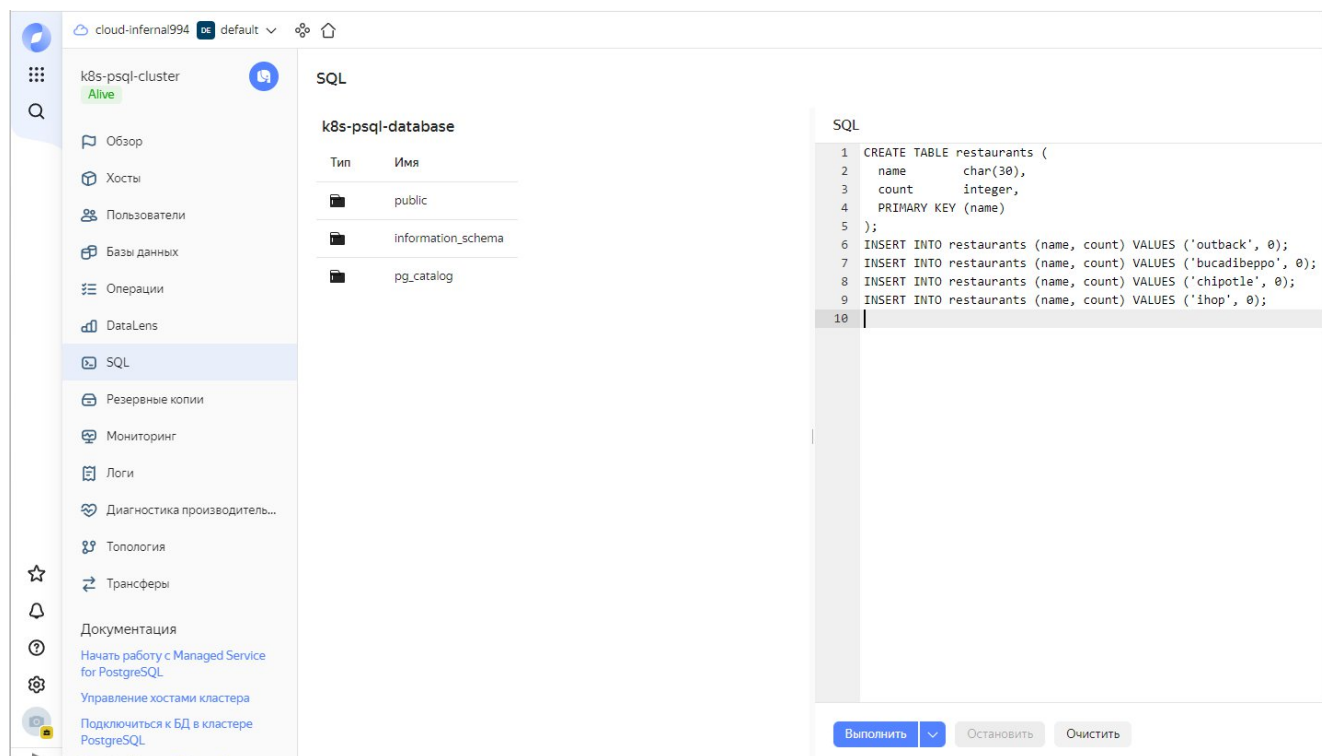


Рисунок 6 – SQL запрос

Выполните SQL запрос на создание таблицы и добавление строк.

### 3.4 Настройка CI/CD

В проект Graduation Work создан токен для чтения/записи в реестр образов, а также добавлены переменные, необходимые для работы конвейера (pipeline).

Active Deploy Tokens (1)

Name	Username	Created	Expires	Scopes
gitlab-ci-token	gitlab-ci-token	Jul 15, 2023	Never	read_repository, read_registry, write_registry

Revoke

Рисунок 7 – Токен доступа к реестру образов

Type	↑ Key	Value	Options	Environments
Variable	CL_BUILD_TOKEN	*****	Protected, Masked	All (default)
Variable	YC_CLOUD_ID	*****	Protected	All (default)
Variable	YC_FOLDER_ID	*****	Protected	All (default)
Variable	YC_TOKEN	*****	Protected, Masked	All (default)

Рисунок 8 – Переменные окружения проекта

Подготовлен *docker-compose.yml* для тестирования локальной сборки. Подробнее можно ознакомиться в Листинг Г.1.

Для линтера файлов *yml* настроен конфигурационный файл исключений (Листинг Г.2).

Для настройки конвейера подготовлен файл *.gitlab-ci.yml*. Конвейер состоит из этапов:

- 1) **lint** – линтеры helm и *yml*
- 2) **build** – сборка образов docker
- 3) **test** – тестирование локальной сборки
- 4) **cleanup** – очистка локальной сборки
- 5) **push** – добавление новых образов в реестр Gitlab
- 6) **deploy** – деплой helm чарта в кластер Kubernetes

Подробно ознакомиться с конвейером можно в Листинг Г.4.

### 3.5 Проверка работы приложения

Дождитесь окончания стадии **deploy**. В самом задании можно увидеть информацию об URL-адресе развернутого приложения.

```

416 - hosts:
417   - yelb.s053278.ru
418     secretName: yelb.s053278.ru
419 ...
420 NOTES:
421 App available to URL: https://yelb.s053278.ru
422 Running after_script
423 Running after script...
424 $ export BALANCER=$(kubectl get svc | grep LoadBalancer | awk '{ print $4 }')
425 $ ./yc dns zone add-records --name s053278-ru --record "yelb 60 A $BALANCER"
426 +-----+-----+-----+-----+
427 | ACTION | NAME | TYPE | DATA | TTL |
428 +-----+-----+-----+-----+
429 +-----+-----+-----+-----+
430 +-----+-----+-----+-----+
431 Cleaning up project directory and file based variables
432 Job succeeded

```

deploy

→

●
deploy

Рисунок 9 – Успешное задание deploy

Перейдем на страницу приложения – <https://yelb.s053278.ru/>. Приложение открылось и работает.

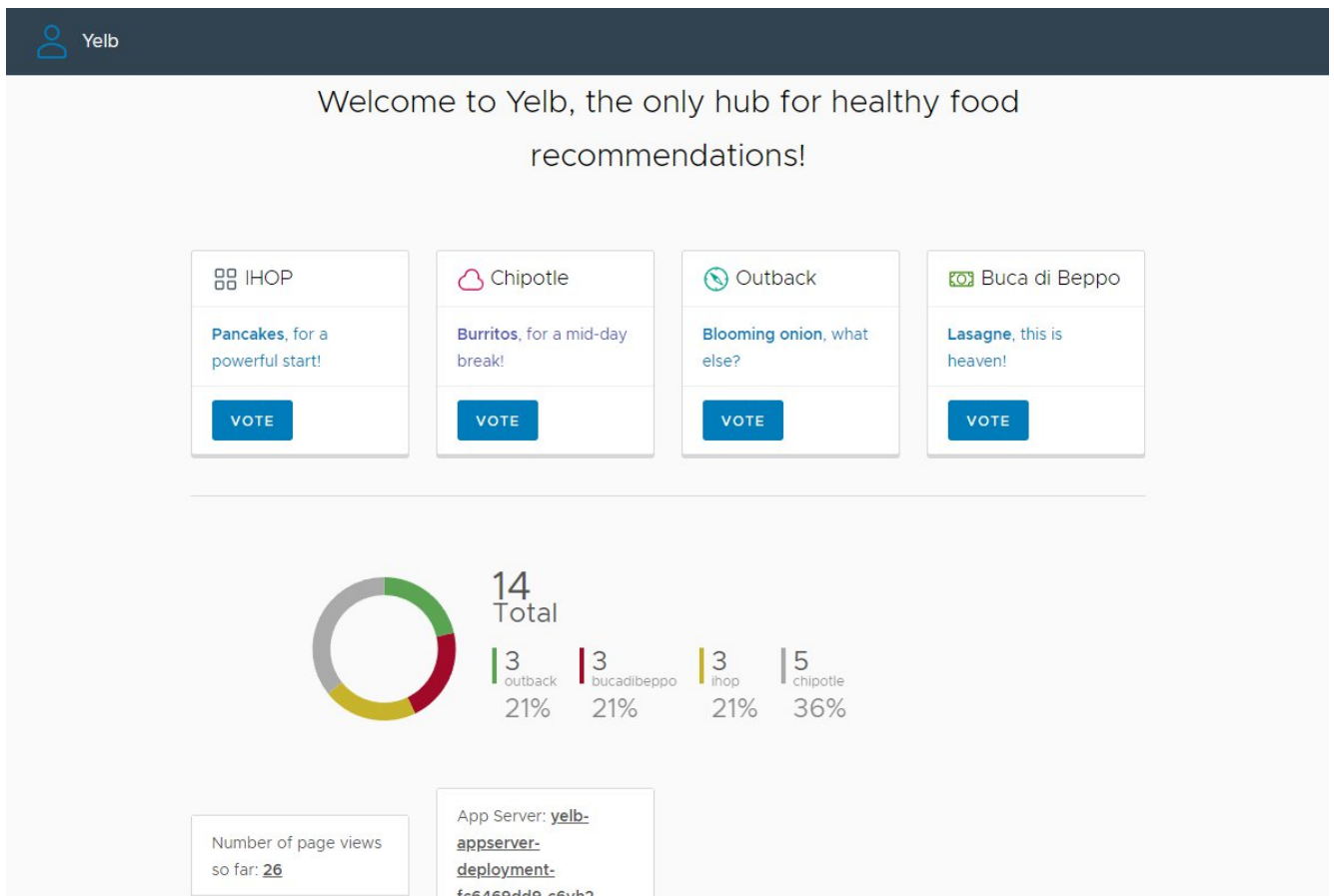


Рисунок 10 – Работа приложения

Логи и различные метрики приложения можно посмотреть в Рабочей нагрузке кластера.

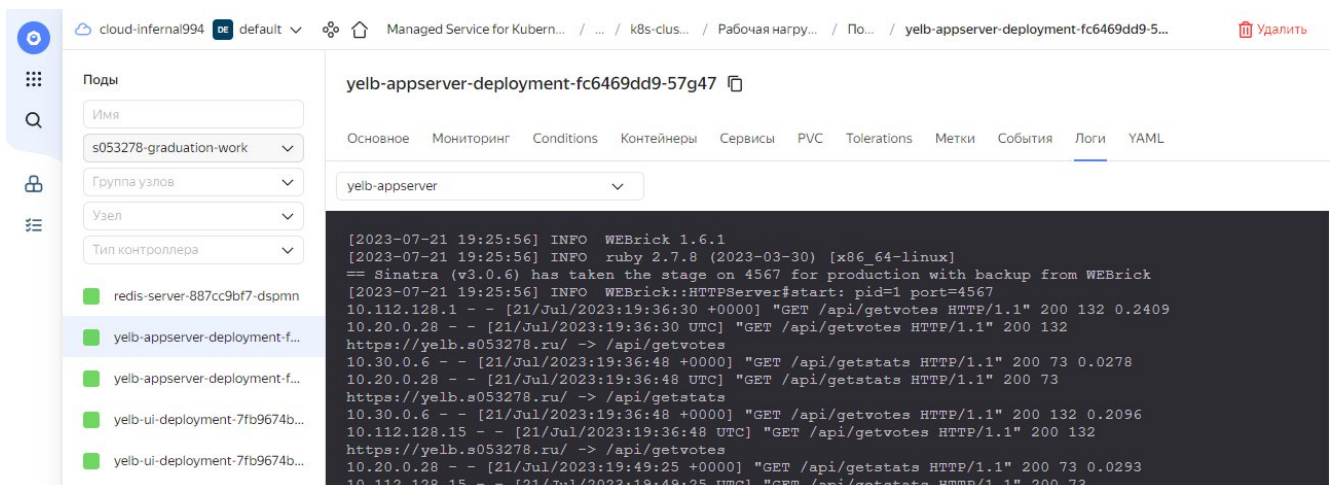


Рисунок 11 – Логи приложения

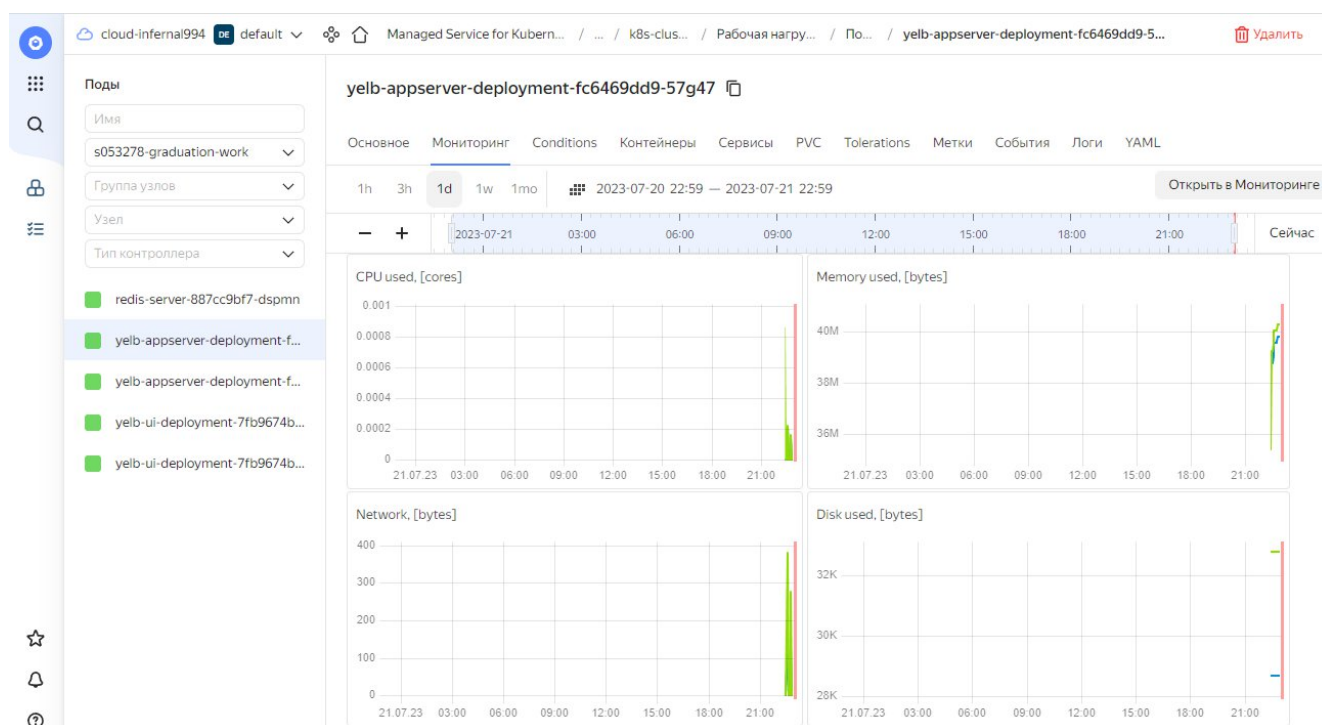


Рисунок 12 – Метрики пода приложения

## 4 Заключение

Задача выполнена успешно и в полном объеме, применяя знания и умения, приобретенные в ходе прохождения обучения. Внедрены некоторые успешные практики, полученные в ходе обучения. Все этапы разработки были задокументированы в данном отчете.

В проекте настроен CI/CD для работы с Kubernetes, а также возможность изменять инфраструктуру через IaC, используя Terraform.

Для проверки результатов работы настроены учетные записи в Yandex Cloud и в Gitlab с правами на просмотр всех компонентов. В Gitlab и Yandex Cloud добавлен пользователь (см. п.1.3) с ролью Maintainer.

Для по вопросам работы финального задания просьба обратиться по следующим контактам:

- email: \*\*\*
- telegram: \*\*\*

### Листинг А.1 – содержимое providers.tf

```
#####
# Конфигурационный файл провайдеров.
# Описаны необходимые версии terraform и подключаемых модулей.
#
# Учетные записи для подключения к облаку
# хранятся как переменные CI/CD Variables:
#   - YC_TOKEN
#   - YC_CLOUD_ID
#   - YC_FOLDER_ID
#   - TF_VAR_YC_FOLDER_ID
#####

terraform {
  required_providers {
    yandex = {
      source = "yandex-cloud/yandex"
      version = "~> 0.80"
    }
  }
  required_version = ">= 0.13"
}
```

### Листинг А.2 – содержимое backend.tf

```
#####
# Конфигурационный файл-заглушка для работы с удаленным tfstate в проекте GitLab.
#
# Данные передаются как переменные окружения при работе
# контейнера registry.gitlab.com/gitlab-org/terraform-images/stable:latest
#
# Для применения конфигурации используется команда: gitlab-terraform
# TF_ADDRESS=${CI_API_V4_URL}/projects/${CI_PROJECT_ID}/terraform/state/yc-tfstate
#####

terraform {
  backend "http" {
  }
}
```

### Листинг А.3 – содержимое networks.tf

```
#####
# Конфигурационный файл сети.
# Описаны сеть и подсети в каждой зоне доступности,
# создает DNS-зону для управления DNS-записями.
#
# Для работы необходимо настроить у провайдера домена делегирование:
#   - ns1.yandexcloud.net
#   - ns2.yandexcloud.net
#####

resource "yandex_vpc_network" "this" {
  name = "${var.name_prefix}-network"
}

resource "yandex_vpc_subnet" "this" {
  count = length(var.zone)
```

```

name          = "${var.name_prefix}-${var.zone[count.index]}"
zone          = var.zone[count.index]
network_id    = yandex_vpc_network.this.id
v4_cidr_blocks = var.cidr_blocks[count.index]
}

resource "yandex_dns_zone" "this" {
  name          = replace(var.dns_name, ".", "-")
  description   = "Public zone"

  zone         = "${var.dns_name}."
  public       = true
}

```

#### Листинг А.4 – содержимое serviceaccount.tf

```

#####
# Конфигурационный файл сервисного аккаунта.
# Создает сервисный аккаунт и назначает роли:
#   - editor
#   - k8s-clusters-agent
#   - vpc-public-admin
#   - container-registry.images.puller
#   - viewer
#####

resource "yandex_iam_service_account" "this" {
  name          = "${var.name_prefix}-sa"
  description   = "Service account for management k8s cluster"
  folder_id    = var.YC_FOLDER_ID
}

resource "yandex_resourcemanager_folder_iam_member" "editor" {
  # Сервисному аккаунту назначается роль "editor".
  folder_id = var.YC_FOLDER_ID
  role      = "editor"
  member    = "serviceAccount:${yandex_iam_service_account.this.id}"
}

resource "yandex_resourcemanager_folder_iam_member" "k8s-clusters-agent" {
  # Сервисному аккаунту назначается роль "k8s.clusters.agent".
  folder_id = var.YC_FOLDER_ID
  role      = "k8s.clusters.agent"
  member    = "serviceAccount:${yandex_iam_service_account.this.id}"
}

resource "yandex_resourcemanager_folder_iam_member" "vpc-public-admin" {
  # Сервисному аккаунту назначается роль "vpc.publicAdmin".
  folder_id = var.YC_FOLDER_ID
  role      = "vpc.publicAdmin"
  member    = "serviceAccount:${yandex_iam_service_account.this.id}"
}

resource "yandex_resourcemanager_folder_iam_member" "images-puller" {
  # Сервисному аккаунту назначается роль "container-registry.images.puller".
  folder_id = var.YC_FOLDER_ID
  role      = "container-registry.images.puller"
  member    = "serviceAccount:${yandex_iam_service_account.this.id}"
}

resource "yandex_resourcemanager_folder_iam_member" "viewer" {
  folder_id = var.YC_FOLDER_ID
  role      = "viewer"
  member    = "serviceAccount:${yandex_iam_service_account.this.id}"
}

```

## Листинг А.5 – содержимое kmsprovider.tf

```
#####
# Конфигурационный файл KMS.
# Создает симметричный ключ шифрования для секретной информации.
#####

resource "yandex_kms_symmetric_key" "this" {
  # Ключ для шифрования важной информации, такой как пароли, OAuth-токены и SSH-ключи.
  name           = "${var.name_prefix}-kms-key"
  default_algorithm = "AES_128"
  rotation_period  = "8760h" # 1 год.
}
```

## Листинг А.6 – содержимое kubernetes.tf

```
#####
# Конфигурационный файл кластера Kubernetes.
# Создает кластер и распределяет ноды в по всем зонам доступности.
#
# Для деплоя кластера требуются:
# - KMS провайдер (kmsprovider.tf)
# - сервисный аккаунт (serviceaccount.tf)
#####

resource "yandex_kubernetes_cluster" "this" {
  name           = "${var.name_prefix}-cluster"
  network_id     = yandex_vpc_network.this.id
  master {
    version       = var.k8s_version
    public_ip     = true

    regional {
      region = "ru-central1"

      dynamic "location" {
        for_each = yandex_vpc_subnet.this

        content {
          zone       = location.value.zone
          subnet_id  = location.value.id
        }
      }
    }
  }
}

service_account_id      = yandex_iam_service_account.this.id
node_service_account_id = yandex_iam_service_account.this.id

depends_on = [
  yandex_resourcemanager_folder_iam_member.editor,
  yandex_resourcemanager_folder_iam_member.k8s-clusters-agent,
  yandex_resourcemanager_folder_iam_member.vpc-public-admin,
  yandex_resourcemanager_folder_iam_member.images-puller,
  yandex_resourcemanager_folder_iam_member.viewer
]

kms_provider {
  key_id = yandex_kms_symmetric_key.this.id
}

resource "yandex_kubernetes_node_group" "this" {
  cluster_id = yandex_kubernetes_cluster.this.id
  name       = "${var.name_prefix}-nodes"

  instance_template {
```



```

name      = "node{instance.index}"
platform_id = "standard-v3"

container_runtime {
  type = "containerd"
}

network_interface {
  nat = true
  subnet_ids = yandex_vpc_subnet.this[*].id
}

resources {
  memory      = var.k8s_resources.mem
  cores       = var.k8s_resources.cpu
  core_fraction = var.k8s_resources.fraction
}

boot_disk {
  type = "network-hdd"
  size = var.k8s_resources.disk
}
}

allocation_policy {
  dynamic "location" {
    for_each = yandex_vpc_subnet.this

    content {
      zone      = location.value.zone
      subnet_id = location.value.id
    }
  }
}

scale_policy {
  fixed_scale {
    size = var.k8s_nodes
  }
}
}

```

## Листинг А.7 – содержимое database.tf

```

#####
# Конфигурационный файл кластера PostgreSQL.
# Создает кластер по всем зонам доступности, а также базу данных и пользователя для работы с ней.
#####

resource "yandex_mdb_postgresql_cluster" "this" {
  name      = "${var.name_prefix}-psql-cluster"
  environment = "PRODUCTION"
  network_id = yandex_vpc_network.this.id

  config {
    version = var.psql_version

    access {
      web_sql = true
    }

    resources {
      resource_preset_id = var.psql_resource_preset_id
      disk_type_id       = var.psql_storage.type
      disk_size          = var.psql_storage.size
    }
  }
}

```

```

    }
    pooler_config {
      pool_discard = true
      pooling_mode = "SESSION"
    }
  }
  dynamic "host" {
    for_each = var.zone

    content {
      zone      = host.value
      name      = "psql-${host.value}"
      subnet_id = yandex_vpc_subnet.this[index(var.zone, host.value)].id
    }
  }
}

resource "yandex_mdb_postgresql_database" "this" {
  cluster_id = yandex_mdb_postgresql_cluster.this.id
  name       = "${var.name_prefix}-psql-database"
  owner      = var.psql_user
  depends_on = [
    yandex_mdb_postgresql_user.this
  ]
}

resource "yandex_mdb_postgresql_user" "this" {
  cluster_id = yandex_mdb_postgresql_cluster.this.id
  name       = var.psql_user
  password   = var.psql_password
}

```

## Листинг А.8 – содержимое variables.tf

```

#####
# General settings
#####

variable "name_prefix" {
  type        = string
  description = "Prefix name"
  default     = "k8s"
}

variable "YC_FOLDER_ID" {
  type        = string
  description = "Needed TF_VAR_YC_FOLDER_ID for Service account"
}

#####
# Network settings
#####

variable "cidr_blocks" {
  type        = list(list(string))
  description = "cidr subnet"
  default     = [
    ["10.10.0.0/24"],
    ["10.20.0.0/24"],
    ["10.30.0.0/24"]
  ]
}

variable zone {
  type        = list(string)
  description = "Zones"
}

```

```

    default      = [
        "ru-central1-a",
        "ru-central1-b",
        "ru-central1-c"
    ]
}

variable "dns_name" {
    type          = string
    description = "DNS-name Public zone"
    default      = "s053278.ru"
}

#####
# K8s settings
#####

variable "k8s_version" {
    type          = string
    description = "Version kubernetes"
    default      = "1.24"
}

variable "k8s_nodes" {
    type          = number
    description = "Count nodes k8s cluster"
    default      = 3
}

variable "k8s_resources" {
    type = object({
        fraction = number
        cpu      = number
        mem      = number
        disk     = number
    })
    description = "Hardware resources for nodes k8s cluster"
    default = ({
        fraction = 20 # (%) Доля мощности CPU
        cpu      = 2
        mem      = 2
        disk     = 64
    })
}

#####
# PostgreSQL settings
#####

variable "psql_resource_preset_id" {
    type          = string
    description = "Resources for PostgreSQL nodes" # https://cloud.yandex.ru/docs/data-proc/concepts/instance-types
    default      = "c3-c2-m4"
}

variable "psql_storage" {
    type = object({
        type = string
        size = number
    })
    description = "Storage for PostgreSQL nodes"
    default = ({
        type = "network-hdd" # https://cloud.yandex.ru/docs/compute/concepts/disk
        size = 20
    })
}

```

```

variable "psql_version" {
  type      = string
  description = "PostgreSQL version"
  default    = "15"
}

variable "psql_user" {
  type      = string
  description = "Database user"
  default    = "dbuser"
}

variable "psql_password" {
  type      = string
  description = "User password"
  default    = "P@$w0rd" # Change Me
}

```

## Листинг А.9 – содержимое .gitlab-ci.yml

```

stages:
  - validate
  - plan
  - deploy
  - destroy

.base-terraform:
  tags:
    - docker
  image:
    name: registry.gitlab.com/gitlab-org/terraform-images/stable:latest
  variables:
    TF_ADDRESS: ${CI_API_V4_URL}/projects/${CI_PROJECT_ID}/terraform/state/yc-tfstate

terraform-validate:
  stage: validate
  extends: .base-terraform
  before_script:
    - cp .terraformrc ~/
  script:
    - gitlab-terraform init; gitlab-terraform validate

terraform-plan:
  stage: plan
  extends: .base-terraform
  before_script:
    - cp .terraformrc ~/
  script:
    - gitlab-terraform plan
    - gitlab-terraform plan-json
  artifacts:
    public: false
    paths:
      - plan.cache
    reports:
      terraform: plan.json

deploy:
  stage: deploy
  extends: .base-terraform
  before_script:
    - cp .terraformrc ~/
  script:
    - gitlab-terraform apply

destroy:

```

```
stage: destroy
extends: .base-terraform
before_script:
  - cp .terraformrc ~/
script:
  - gitlab-terraform destroy
dependencies:
  - deploy
when: manual
```

### Листинг Б.1 – Dockerfile приложения yelb-ui

```
FROM node:12.22
MAINTAINER massimo@it20.info

WORKDIR /

RUN npm install -g @angular/cli@6.0.0

RUN npm install node-sass@4.13.1

ADD clarity-seed-newfiles clarity-seed-newfiles

RUN git clone https://github.com/vmware/clarity-seed.git
WORKDIR /clarity-seed
RUN git checkout -b f3250ee26ceb847f61bb167a90dc957edf6e7f43

RUN cp /clarity-seed-newfiles/src/index.html /clarity-seed/src/index.html
RUN cp /clarity-seed-newfiles/src/styles.css /clarity-seed/src/styles.css
RUN cp /clarity-seed-newfiles/src/env.js /clarity-seed/src/env.js
RUN cp /clarity-seed-newfiles/src/app/app* /clarity-seed/src/app
RUN cp /clarity-seed-newfiles/src/app/env* /clarity-seed/src/app
RUN cp /clarity-seed-newfiles/src/environments/env* /clarity-seed/src/environments
RUN cp /clarity-seed-newfiles/package.json /clarity-seed/package.json
RUN cp /clarity-seed-newfiles/angular-cli.json /clarity-seed/.angular-cli.json
RUN rm -r /clarity-seed/src/app/home
RUN rm -r /clarity-seed/src/app/about

WORKDIR /clarity-seed/src

RUN npm install

RUN ng build --environment=prod --output-path=./prod/dist/ -aot -vc -cc -dop --buildOptimizer
RUN ng build --environment=test --output-path=./test/dist/
RUN ng build --environment=dev --output-path=./dev/dist/

FROM nginx:1.17.10
MAINTAINER massimo@it20.info

WORKDIR /
ADD startup.sh startup.sh

ENV UI_ENV=prod

RUN chmod +x startup.sh

COPY --from=0 /clarity-seed/src/prod/dist /clarity-seed/prod/dist
COPY --from=0 /clarity-seed/src/test/dist /clarity-seed/test/dist
COPY --from=0 /clarity-seed/src/dev/dist /clarity-seed/dev/dist

CMD ["/startup.sh"]
```

## Листинг Б.2 – Скрипт yelb-ui

```
#!/bin/bash
NGINX_CONF=/etc/nginx/conf.d/default.conf
cd clarity-seed

# when the variable is populated a search domain entry is added to resolv.conf at startup
# this is needed for the ECS service discovery given the app works by calling host names and not FQDNs
# a search domain can't be added to the container when using the awsvpc mode
# and the awsvpc mode is needed for A records (bridge only supports SRV records)
if [ "$SEARCH_DOMAIN" ]; then echo "search ${SEARCH_DOMAIN}" >> /etc/resolv.conf; fi
if [ -z "$YELB_APPSERVER_ENDPOINT" ]; then YELB_APPSERVER_ENDPOINT="http://yelb-appserver:4567"; fi

sed -i -- 's#/#usr/share/nginx/html#/clarity-seed/'$UI_ENV'/dist#g' $NGINX_CONF

# this adds the reverse proxy configuration to nginx
# everything that hits /api is proxied to the app server
if ! grep -q "location /api" "$NGINX_CONF"; then
    eval "cat <<EOF
location /api {
    proxy_pass \"$YELB_APPSERVER_ENDPOINT\"/api;
    proxy_http_version 1.1;
}
gzip on;
gzip_types text/plain text/css application/json application/javascript application/x-javascript text/xml
application/xml application/xml+rss text/javascript;
gunzip on;
EOF
" > /proxycfg.txt
    # echo "        proxy_set_header Host \$host;" >> /proxycfg.txt
    sed --in-place '/server_name localhost;/ r /proxycfg.txt' $NGINX_CONF
fi

nginx -g "daemon off;"
```

## Листинг Б.3 – Dockerfile приложения yelb-appserver

```
FROM bitnami/ruby:2.7.8

ENV LANG=en_us.UTF-8
ENV LC_ALL=C.UTF-8
ENV RACK_ENV=production

RUN apt-get update \
&& apt-get install libpq-dev zlib1g-dev -y \
&& rm -rf /var/lib/apt/lists/*

RUN gem install sinatra \
&& gem install redis

RUN gem install pg \
&& gem install aws-sdk-dynamodb pg

COPY yelb-appserver.rb yelb-appserver.rb
COPY Gemfile Gemfile
COPY modules modules

ENV REDIS_SERVER_ENDPOINT=redis-server
ENV YELB_DB_SERVER_ENDPOINT=yelb-db
ENV YELB_DB_SERVER_PORT=5432
ENV YELB_DB_NAME=yelbdatabase
ENV YELB_DB_USER=dbuser
ENV YELB_DB_PASS=password
```

```
CMD ["ruby", "/app/yelb-appserver.rb", "-o", "0.0.0.0"]
```

#### Листинг Б.4 – фрагмент изменений yelb-appserver\yelb-appserver.rb

```
<начало фрагмента>
configure :production do
  set :redishost, ENV['REDIS_SERVER_ENDPOINT']
  set :port, 4567
  set :yelbdbhost => ENV['YELB_DB_SERVER_ENDPOINT']
  set :yelbdbport => ENV['YELB_DB_SERVER_PORT']
  set :yelbddbrestaurants => ENV['YELB_DDB_RESTAURANTS']
  set :yelbddbcache => ENV['YELB_DDB_CACHE']
  set :awsregion => ENV['AWS_REGION']
end
configure :test do
  set :redishost, ENV['REDIS_SERVER_ENDPOINT']
  set :port, 4567
  set :yelbdbhost => ENV['YELB_DB_SERVER_ENDPOINT']
  set :yelbdbport => ENV['YELB_DB_SERVER_PORT']
  set :yelbddbrestaurants => ENV['YELB_DDB_RESTAURANTS']
  set :yelbddbcache => ENV['YELB_DDB_CACHE']
  set :awsregion => ENV['AWS_REGION']
end
configure :development do
  set :redishost, "localhost"
  set :port, 4567
  set :yelbdbhost => "localhost"
  set :yelbdbport => 5432
  set :yelbddbrestaurants => ENV['YELB_DDB_RESTAURANTS']
  set :yelbddbcache => ENV['YELB_DDB_CACHE']
  set :awsregion => ENV['AWS_REGION']
end
...
$yelbdbhost = settings.yelbdbhost
$redishost = settings.redishost
$yelbdbport = ENV['YELB_DB_SERVER_PORT']
$db_name = ENV['YELB_DB_NAME']
$db_user = ENV['YELB_DB_USER']
$db_pass = ENV['YELB_DB_PASS']
<конец фрагмента>
```

#### Листинг Б.5 – фрагмент изменений yelb-appserver\modules\restaurantsdbread.rb

```
<начало фрагмента>
  con = PG.connect :host => $yelbdbhost,
                  :port => $yelbdbport,
                  :dbname => $db_name,
                  :user => $db_user,
                  :password => $db_pass
<конец фрагмента>
```

#### Листинг Б.6 – фрагмент изменений yelb-appserver\modules\restaurantsdbupdate.rb

```
<начало фрагмента>
  con = PG.connect :host => $yelbdbhost,
                  :port => $yelbdbport,
                  :dbname => $db_name,
                  :user => $db_user,
                  :password => $db_pass
<конец фрагмента>
```



Листинг В.1 – содержимое .helm/values.yaml

```
---
# Registry
imagePullSecret: gitlab-ci-token

# Frontend
ui:
  replicaCount: 2

  image: mreferre/yelb-ui
  imageTag: "0.7"

  resources:
    requests:
      cpu: 100m
      memory: 128Mi
    limits:
      cpu: 200m
      memory: 256Mi

  ingress:
    host: yelb.s053278.ru

# Backend
appserver:
  replicaCount: 2

  image: s053278.gitlab.yandexcloud.net:5050/s053278/graduation_work/yelb-appserver
  imageTag: "0.5"

  resources:
    requests:
      cpu: 100m
      memory: 128Mi
    limits:
      cpu: 200m
      memory: 256Mi

# Redis
redis:
  replicaCount: 2

  image: "redis"
  imageTag: "4.0.2"

  resources:
    requests:
      cpu: 100m
      memory: 128Mi
    limits:
      cpu: 200m
      memory: 256Mi

# Database (PostgreSQL)
db:
  host: "yelb-db"
  port: "6432"
  dbname: "k8s-psql-database"
```

```
user: "dbuser"
password: "password"
```

## Листинг В.2 – содержимое .helm\Chart.yaml

```
apiVersion: v2
name: yelb
description: A Helm chart for Kubernetes

# A chart can be either an 'application' or a 'library' chart.
#
# Application charts are a collection of templates that can be packaged into versioned archives
# to be deployed.
#
# Library charts provide useful utilities or functions for the chart developer. They're included as
# a dependency of application charts to inject those utilities and functions into the rendering
# pipeline. Library charts do not define any templates and therefore cannot be deployed.
type: application

# This is the chart version. This version number should be incremented each time you make changes
# to the chart and its templates, including the app version.
# Versions are expected to follow Semantic Versioning (https://semver.org/)
version: 0.5.0

# This is the version number of the application being deployed. This version number should be
# incremented each time you make changes to the application. Versions are not expected to
# follow Semantic Versioning. They should reflect the version the application is using.
# It is recommended to use it with quotes.
appVersion: "0.5.0"
```

## Листинг В.3 – содержимое .helm\templates\ui.yml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Chart.Name }}-ui-deployment
spec:
  replicas: {{ .Values.ui.replicaCount | default 2 }}
  selector:
    matchLabels:
      app: yelb-ui
  strategy:
    rollingUpdate:
      maxSurge: 10%
      maxUnavailable: 10%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: yelb-ui
    spec:
      containers:
        - image: {{ .Values.ui.image }}:{{ .Values.ui.imageTag }}
          name: yelb-ui
          ports:
            - containerPort: 80
          resources:
            {{- toYaml .Values.ui.resources | nindent 12 }}
      imagePullSecrets:
        - name: {{ .Values.imagePullSecret }}
...

```

## Листинг В.4 – содержимое .helm/templates/NOTES.txt

```
App available to URL: https://{{ .Values.ui.ingress.host }}
```

## Листинг В.5 – содержимое .helm/templates/appserver.yml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Chart.Name }}-appserver-deployment
spec:
  replicas: {{ .Values.appserver.replicaCount | default 2 }}
  selector:
    matchLabels:
      app: yelb-appserver
  strategy:
    rollingUpdate:
      maxSurge: 10%
      maxUnavailable: 10%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: yelb-appserver
    spec:
      containers:
        - image: {{ .Values.appserver.image }}:{{ .Values.appserver.imageTag }}
          name: yelb-appserver
          ports:
            - containerPort: 4567
          env:
            - name: "RACK_ENV"
              value: "production"
            - name: "YELB_DB_SERVER_ENDPOINT"
              value: "{{ .Values.db.host }}"
            - name: "YELB_DB_SERVER_PORT"
              value: "{{ .Values.db.port }}"
            - name: "YELB_DB_NAME"
              value: "{{ .Values.db.dbname }}"
            - name: "YELB_DB_USER"
              value: "{{ .Values.db.user }}"
            - name: "YELB_DB_PASS"
              value: "{{ .Values.db.password }}"
          resources:
            {{- toYaml .Values.appserver.resources | nindent 12 }}
      imagePullSecrets:
        - name: {{ .Values.imagePullSecret }}
```

## Листинг В.6 – содержимое .helm/templates/redis.yml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-server
  labels:
    app: redis-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis-server
  template:
    metadata:
      labels:
        app: redis-server
    spec:
      containers:
        - image: redis:4.0.2
          name: redis-server
          ports:
            - containerPort: 6379
          resources:
            {{- toYaml .Values.redis.resources | nindent 12 }}
```

## Листинг В.7 – содержимое .helm/templates/service.yml

```
---
apiVersion: v1
kind: Service
metadata:
  name: yelb-ui
spec:
  ports:
    - port: 80
      targetPort: 80
  selector:
    app: yelb-ui
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  name: yelb-appserver
spec:
  ports:
    - port: 4567
      targetPort: 4567
  selector:
    app: yelb-appserver
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  name: redis-server
spec:
  ports:
    - port: 6379
      targetPort: 6379
  selector:
    app: redis-server
  type: ClusterIP
```

## Листинг В.8 – содержимое .helm/templates/ingress.yml

```
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: {{ .Chart.Name }}-ui-ingress
  annotations:
    kubernetes.io/ingress.class: "nginx"
    cert-manager.io/cluster-issuer: "letsencrypt"
spec:
  rules:
    - host: {{ .Values.ui.ingress.host }}
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: yelb-ui
                port:
                  number: 80
  tls:
    - hosts:
      - {{ .Values.ui.ingress.host }}
      secretName: {{ .Values.ui.ingress.host }}
```

Листинг Г.1 – содержимое docker-compose.yml

```
version: "3.0"
services:
  yelb-ui:
    image: $CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME/yelb-ui:$CI_COMMIT_REF_SLUG-$CI_COMMIT_SHORT_SHA
    depends_on:
      - yelb-appserver
    ports:
      - 80:80
    networks:
      - yelb-network

  yelb-appserver:
    image: $CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME/yelb-appserver:$CI_COMMIT_REF_SLUG-$CI_COMMIT_SHORT_SHA
    depends_on:
      - redis-server
      - yelb-db
    networks:
      - yelb-network

  redis-server:
    image: redis:4.0.2
    networks:
      - yelb-network

  yelb-db:
    build: ./yelb-db
    networks:
      - yelb-network

  test:
    image: curlimages/curl:7.73.0
    command: /bin/sh -c "sleep 20 && curl -s http://yelb-ui/ -i -vvv"
    depends_on:
      - yelb-ui
    networks:
      - yelb-network

networks:
  yelb-network:
    driver: bridge
```

Листинг Г.2 – содержимое .yamllint

```
yaml-files:
  - '*.yaml'
  - '*.yml'
  - '.yamllint'

extends: default

rules:
  line-length:
    max: 80
    level: warning

ignore:
  - .helm/
  - deployments/
```

### Листинг Г.3 – содержимое .gitlab-ci.yml

```
stages:
  - lint
  - build
  - test
  - cleanup
  - push
  - deploy

default:
  tags:
    - docker

lint-yaml:
  stage: lint
  image:
    name: cytopia/yamllint
    entrypoint: ["/bin/ash", "-c"]
  script:
    - yamllint -f colored .

lint-helm:
  stage: lint
  image: alpine/k8s:1.24.15
  script:
    - helm lint .helm

build yelb-ui:
  stage: build
  script:
    - docker build -t $CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME/yelb-ui:$CI_COMMIT_REF_SLUG-$CI_COMMIT_SHORT_SHA ./yelb-ui

build yelb-appserver:
  stage: build
  script:
    - docker build -t $CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME/yelb-appserver:$CI_COMMIT_REF_SLUG-$CI_COMMIT_SHORT_SHA ./yelb-appserver

test:
  stage: test
  image:
    name: docker/compose:1.29.2
    entrypoint: [""]
  script:
    - docker-compose
      -p "$CI_PROJECT_NAME" "$CI_COMMIT_SHORT_SHA"
    up
      --abort-on-container-exit
      --exit-code-from test
      --quiet-pull

cleanup:
  stage: cleanup
  image:
    name: docker/compose:1.29.2
    entrypoint: [""]
  script:
    - docker-compose -p "$CI_PROJECT_NAME" "$CI_COMMIT_SHORT_SHA" down
  when: always

push yelb-ui:
  stage: push
  before_script:
```

```

- docker login -u gitlab-ci-token -p $CI_BUILD_TOKEN $CI_REGISTRY
script:
- docker push $CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME/yelb-ui:$CI_COMMIT_REF_SLUG-$CI_COMMIT_SHORT_SHA
only:
- master

push yelb-appserver:
stage: push
before_script:
- docker login -u gitlab-ci-token -p $CI_BUILD_TOKEN $CI_REGISTRY
script:
- docker push $CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME/yelb-appserver:$CI_COMMIT_REF_SLUG-$CI_COMMIT_SHORT_SHA
only:
- master

deploy:
stage: deploy
image: alpine/k8s:1.24.15
before_script:
- chmod +x yc
- ./yc config set token $YC_TOKEN
- ./yc config set cloud-id $YC_CLOUD_ID
- ./yc config set folder-id $YC_FOLDER_ID
- ./yc managed-kubernetes cluster get-credentials k8s-cluster --external
- export DB_HOST=$(./yc postgresql cluster get k8s-psql-cluster | head -1 | awk '{ print $2 }')
- helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx && helm repo update
- helm upgrade --install ingress-nginx ingress-nginx/ingress-nginx
- kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.9.1/cert-manager.yaml
- kubectl -n $CI_PROJECT_PATH_SLUG apply -f issuer.yaml
script:
- kubectl get namespace | grep "^$CI_PROJECT_PATH_SLUG " || kubectl create namespace $CI_PROJECT_PATH_SLUG
- kubectl create secret docker-registry gitlab-ci-token --docker-server $CI_REGISTRY
--docker-email "s053278@yandex.ru" --docker-username gitlab-ci-token --docker-password $CI_BUILD_TOKEN
--namespace $CI_PROJECT_PATH_SLUG --dry-run=client -o yaml | kubectl apply -f -
- helm upgrade --install $CI_PROJECT_PATH_SLUG .helm
--set ui.image=$CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME/yelb-ui
--set ui.imageTag=$CI_COMMIT_REF_SLUG-$CI_COMMIT_SHORT_SHA
--set appserver.image=$CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME/yelb-appserver
--set appserver.imageTag=$CI_COMMIT_REF_SLUG-$CI_COMMIT_SHORT_SHA
--set db.host=c-$DB_HOST.rw.mdb.yandexcloud.net
--set db.port=6432
--namespace $CI_PROJECT_PATH_SLUG
--create-namespace
--wait
--timeout 300s
--atomic
--debug
after_script:
- export BALANCER=$(kubectl get svc | grep LoadBalancer | awk '{ print $4 }')
- ./yc dns zone add-records --name s053278-ru
--record "yelb 60 A $BALANCER"
only:
- master

```