# Информационная система «Конструктор мультимодальных поездок»

# Инструкция по установке

**Оглавление**

**Перечень сокращений**

| Термин | Определение |
|---|---|
| БД | База данных. |
| ОС | Операционная система |
| ПО | Программное обеспечение |

**1 Инфраструктура, на которой устанавливается ПО**

Необходимое ПО и минимальные ресурсы представлены в таблице 1.

Таблица 1 – Необходимое ПО и минимальные ресурсы

| ПО | Версия | CPU | RAM, Gb | HDD SATA, Gb | HDD SSD, Gb | Ожидаемый адрес в сети | Схема развертки | Комментарий |
|---|---|---|---|---|---|---|---|---|
| Kubernetes | 1.16 | 78 | 525 | 1200 Mount point: / | (2625) — в зависимости от количества нод по 375 Gb на каждую Тут расчет на 7 нод. Mount point: /mnt/data | --- | Необходимо несколько нод, одна их которых обязательно сожержит не менее 409 Gb RAM и не менее 48 CPU одновременно. Остальные ноды содержат не менее 30 Gb оперативной памяти и не менее 4 CPU Все ноды кластера обязательно должны иметь один примонтированный HDD SSD диск на 375 Гб | Необходим для развертки собственных приложений и дополнительного софта |
| Docker | 2.5 | --- | --- | --- | | --- | | |

| ПО | Версия | CPU | RAM, Gb | HDD SATA, Gb | HDD SSD, Gb | Ожидаемый адрес в сети | Схема развертки | Комментарий |
|---|---|---|---|---|---|---|---|---|
| PostgreSQL | 12 | 4 | 15 | 150<br>Mount point: / | 750<br>Mount point: /mnt/data | mmp-postgres-01.kmmp.prod | Один отдельный сервер. | |
| MongoDB | 4.4 | 48 | 256 | 150<br>Mount point: / | 1500<br>Mount point: /mnt/data | mmp-mongodb-01.kmmp.prod | Один отдельный сервер | |
| ElasticSearch | 6.8 | 28 | 128 | 1200<br>Mount point: / | 2250<br>Mount point: /mnt/data | mmp-elasticsearch01.kmmp.prod | Две client ноды: 2 CPU,<br>19 RAM<br>Шесть data нод: 4 CPU,<br>15 RAM | |
| ClickHouse | 20.8 | 4 | 15 | 150<br>Mount point: / | 1125<br>Mount point: /mnt/data | mmp-clickhouse-01.kmmp.prod | Один сервер | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Ubuntu | 20.04 LTS | --- | --- | --- | --- | --- | Каждая ОС разворачивается на сервере со 150 Gb HDD SATA, эти ресурсы учтены в соответствующей колонке ПО | Базовая ОС для всей системы |
| **ПО** | **Версия** | **CPU** | **RAM, Gb** | **HDD SATA, Gb** | **HDD SSD, Gb** | **Ожидаемый адрес в сети** | **Схема развертки** | **Комментарий** |
| Helm | 2.x | --- | --- | --- | --- | --- | На рабочей (локальной) машине | https://v2.helm.sh/docs/ |
| Helmfile | 0.89 | --- | --- | --- | --- | --- | На рабочей (локальной) машине | https://github.com/roboll/helmfile |
| Helm diff plugin | 2.11 | --- | --- | --- | --- | --- | На рабочей (локальной) машине | https://github.com/databus23/helm-diff |
| kubectl | 1.16 | --- | --- | --- | --- | --- | На рабочей (локальной) машине. Настроить вход в вышеуказанный кластер Kubernetes. | |
| Jenkins | 2.x | --- | --- | --- | --- | --- | На рабочей (локальной) машине, либо на внешнем сервере | |

## 2 Пошаговая инструкция

### 2.1 Установка ПО

Для установки программного обеспечения следует:

1. Создать сервера / виртуальные машины, установить на каждую Базовую ОС;
2. Установить ПО согласно схеме развертки.

### 2.2 Настройка окружения

Для настройки окружения следует совершить следующие действия:

1. Клонировать репозиторий rzd_devops <TBD: поставить валидную ссылку на репозитарий, либо ссылку на код>
2. Затем перейти в директорию kube/rzd-gateline/PROD-cluster
3. Далее выполнить поочередно команды:

```
kubectl create namespace connector
kubectl create namespace ar kubectl
create namespace mmp

kubectl create secret -n connector docker-registry docker-registry-key \
--docker-server=<TBD: docker-registry-server-url> \
--docker-username=<TBD: docker-registry-server-username> \
--docker-password=<TBD: docker-registry-server-password> \ --docker-email=<TBD:
docker-registry-server-email>
kubectl patch serviceaccount -n connector default -p '{"imagePullSecrets": [{"name": "docker-
registry-key"}]}'

kubectl create secret -n mmp docker-registry docker-registry-key \ --docker-server=<TBD:
docker-registry-server-url> \
--docker-username=<TBD: docker-registry-server-username> \
--docker-password=<TBD: docker-registry-server-password> \ --docker-email=<TBD: docker-
registry-server-email> kubectl patch serviceaccount -n connector default -p
'{"imagePullSecrets": [{"name": "docker-
registry-key"}]}'
```

kubectl create secret -n ar docker-registry docker-registry-key \ --docker-server=<TBD: docker-registry-server-url> \
--docker-username=<TBD: docker-registry-server-username> \
--docker-password=<TBD: docker-registry-server-password> \ --docker-email=<TBD: docker-registry-server-email> kubectl patch serviceaccount -n connector default -p '{"imagePullSecrets": [{"name": "docker-registry-key"}]}'

kubectl apply -f pre-conditions/admin-rbac.yaml kubectl apply -f pre-conditions/helm-rbac-config.yaml

helm init --service-account tiller

## загрузить вкластер валидный SSL сертификат
kubectl create secret tls ingress-tls-secret --key ./ssl/private_key.key --cert ./ssl/bundle.crt

helmfile sync

## 2.3 Настройка деплоя приложений

1. Настроить ноду на Jenkins на работу внутри кластера Kubernetes.
2. Настроить Jenkins на деплой кода из репозитария rzd_backend ветки master используя Jenkinsfile.
3. При необходимости поправить в файлах директории kube/rzd-gateline/PRODcluster/ci/releases адреса баз данных на ожидаемые (см. выше в таблице).

4.Запустить приложения.