Инструкция по развертыванию экземпляра программного обеспечения

Система `LT Dynamic` в Kubernetes

23 марта 2026 г.

1. Назначение документа

Документ определяет порядок развертывания экземпляра ПО `LT Dynamic` в Kubernetes-кластере, включая подготовку инфраструктурных зависимостей, сборку контейнерных образов, публикацию артефактов и запуск сервисов.

2. Область применения

Инструкция применяется для:

- первичного развертывания экземпляра системы;
- развертывания новой версии сервисов;
- восстановления работоспособности после переустановки окружения.

В рамках документа рассматривается развертывание сервисного контура приложения, необходимых инфраструктурных компонентов и подключение внешних интеграций.

3. Внешние зависимости и инфраструктура

3.1. Платформенные зависимости

Компонент	**Назначение**
Kubernetes-кластер	Размещение сервисов `LT Dynamic`, управление rollout и service discovery
Container Registry	Хранение и публикация образов сервисов
`kubectl`	Применение манифестов и контроль статуса ресурсов
`yc` CLI	Создание и сопровождение Kubernetes-кластера в Yandex Cloud (Managed Kubernetes)
`docker` + `buildx`	Сборка и публикация multi-arch/multi-platform образов

Система секретов (K8s Secret/внешний vault)	Хранение учетных данных и чувствительных параметров интеграций

3.2. Используемые базы данных и брокеры

Компонент	**Тип**	**Назначение в системе**
MySQL	Реляционная БД	Референсные и операционные данные, используемые рядом сервисов
PostgreSQL	Реляционная БД	Данные доменных сервисов (в том числе контуры `booker`, `rules`, `rate-limiter`, `admin`)
MongoDB	Документная БД	Хранение кэшируемых документов и поисковых артефактов
Redis	In-memory хранилище	Кэширование, быстрый доступ к временным данным и служебным структурам
NATS	Брокер сообщений	Асинхронный обмен между сервисами

3.3. Внешние сервисы и интеграции

Для полноценной работы экземпляра должны быть доступны:

- сервис курсов валют (конвертация и пересчет цен);
- внешние API поставщиков (авиа, отели, трансферы, страхование);
- объектное хранилище документов (S3-совместимый контур для выгрузки файлов);
- сетевые маршруты к внешним API (egress, DNS, TLS).

Параметры доступа к внешним системам передаются через `ConfigMap`/`Secret` и не фиксируются в данном документе.

3.4. Минимальный состав разворачиваемых сервисов

Скрипт развертывания охватывает сервисы:

```
admin alternatives-searcher booker cache dispatcher documents flights
hotels insurance matcher metrics-collector mixer operator rate-limiter
rules taskmanager transfers
```

4. Комплект скриптов для поставки

Для загрузки в облачное хранилище подготовлен отдельный пакет скриптов:

```
deployment/kubernetes_scripts/
  provision-k8s-cluster.sh
  docker-login.sh
  build-images.sh
```

```
deploy-k8s.sh
docker_files/
  docker-compose.yml
  .dockerignore
  build/Dockerfile
  build/cachemanager.Dockerfile
  build/compiler.Dockerfile
  build/mysql.Dockerfile
  build/postgres.Dockerfile
  build/dev-redis/dev-redis.Dockerfile
```

4.1. provision-k8s-cluster.sh

Назначение: автоматизированное создание Kubernetes-кластера в Yandex Cloud (Managed Kubernetes) с базовой сетевой и сервисной инфраструктурой.

Основные переменные окружения:

- `YC_FOLDER_ID` (обязательно, может быть взят из `yc config`);
- `K8S_CLUSTER_NAME`, `K8S_VERSION`, `K8S_ZONE`;
- `NETWORK_NAME`, `SUBNET_NAME`, `SUBNET_CIDR`;
- `NODE_GROUP_NAME`, `NODE_COUNT`, `NODE_CORES`, `NODE_MEMORY_GB`, `NODE_DISK_GB`;
- `SSH_PUBLIC_KEY_PATH`, `KUBECONFIG_PATH`.

Что делает скрипт:

- создает (или переиспользует) VPC network и subnet;
- создает сервисные аккаунты для control plane и node group;
- создает кластер и node group;
- получает kubeconfig и подключает доступ к кластеру через `kubectl`.

Результат выполнения:

- готовый Kubernetes-кластер с рабочим пулом узлов;
- локально обновленный `kubeconfig` для последующего деплоя сервисов.

4.2. docker-login.sh

Назначение: авторизация docker-клиента в контейнерном реестре перед публикацией или загрузкой образов.

Входные условия:

- установлен и запущен Docker;
- есть сетевой доступ к реестру;
- пользователь имеет право на вход в реестр.

Результат выполнения:

- в локальный docker credential store записывается активная сессия;
- команды `docker pull/push` становятся доступны для целевого реестра.

4.3. build-images.sh

Назначение: сборка и публикация контейнерных образов сервисов системы в реестр.

Основные переменные окружения:

- REGISTRY (обязательно) — адрес репозитория образов;
- ECR_REPO (обязательно) — базовый репозиторий для ARG ECR_REPO в Dockerfile;
- IMAGE_TAG (опционально) — тег версии образов, при отсутствии генерируется автоматически;
- PLATFORM (опционально, по умолчанию linux/amd64);
- SERVICES (опционально) — перечень сервисов для сборки.

Что делает скрипт:

- валидирует наличие docker и docker buildx;
- для каждого сервиса выполняет docker buildx build -push;
- публикует образы вида <REGISTRY>/<service>:<IMAGE_TAG>.

Артефакты:

- опубликованные образы в реестре;
- локальный файл .k8s-image-tag с фактическим тегом версии.

4.4. deploy-k8s.sh

Назначение: развертывание или обновление сервисов в Kubernetes namespace.

Основные переменные окружения:

- REGISTRY (обязательно) — источник контейнерных образов;
- IMAGE_TAG (обязательно или берется из .k8s-image-tag);
- NAMESPACE (опционально, по умолчанию dynamic);
- SERVICES (опционально) — список развертываемых сервисов;
- REPLICAS (опционально, по умолчанию 1).

Что делает скрипт:

- проверяет наличие namespace, при необходимости создает его;
- проверяет наличие ConfigMap dynamic-common-env и Secret dynamic-common-secrets;
- применяет Deployment и Service для каждого сервиса;
- ожидает успешный rollout status для всех deployments.

Результат выполнения:

- сервисы запущены с целевыми версиями образов;
- создана или обновлена внутренняя Kubernetes-сеть сервисов (ClusterIP).

4.5. Подпапка `docker_files`

Содержит референсный комплект Docker-конфигурации:

- `build/Dockerfile` — базовая сборка сервисных образов;
- `build/cachemanager.Dockerfile`, `build/compiler.Dockerfile` — специализированные сборочные сценарии;
- `build/mysql.Dockerfile`, `build/postgres.Dockerfile`, `build/dev-redis/dev-redis.Docke` — инфраструктурные образы;
- `docker-compose.yml`, `.dockerignore` — полный снимок контейнерной конфигурации для поставки.

5. Порядок развертывания в Kubernetes

5.1. Развертывание Kubernetes-кластера

Примечание: шаг выполняется при первичном создании инфраструктуры или полном переразвертывании кластера.

1. Подготовить параметры кластера:

```
export YC_FOLDER_ID=<folder-id>
export K8S_CLUSTER_NAME=dynamic
export K8S_ZONE=ru-central1-a
export K8S_VERSION=1.30
export NETWORK_NAME=dynamic-network
export SUBNET_NAME=dynamic-subnet-a
export SUBNET_CIDR=10.140.0.0/24
export NODE_GROUP_NAME=dynamic-main-pool
export NODE_COUNT=3
```

2. Запустить скрипт создания кластера:

```
./deployment/kubernetes_scripts/provision-k8s-cluster.sh
```

5.2. Подготовительные проверки после создания кластера

1. Перейти в корневую директорию системы:

```
cd /path/to/dynamic
```

2. Проверить доступ к кластеру:

```
kubectl cluster-info
```

3. Проверить текущий контекст:

```
kubectl config current-context
```

4. Проверить наличие утилит:

```
docker --version
docker buildx version
kubectl version --client
```

5.3. Авторизация в контейнерном реестре

```
./deployment/kubernetes_scripts/docker-login.sh
```

5.4. Сборка и публикация контейнерных образов

Установить переменные:

```
export REGISTRY=registry.example.com/dynamic
export ECR_REPO=cr.yandex/<repo-id>
export IMAGE_TAG=20260216-r1
```

Запустить сборку:

```
./deployment/kubernetes_scripts/build-images.sh
```

Результат:

- в registry публикуются образы всех сервисов с тегом `IMAGE_TAG`;
- локально создается файл `.k8s-image-tag` с зафиксированным тегом.

5.5. Подготовка namespace и конфигурации

Создание/обновление namespace:

```
export NAMESPACE=dynamic
kubectl create namespace $NAMESPACE --dry-run=client -o yaml | kubectl apply -f -
```

Загрузка общих параметров:

```
kubectl -n $NAMESPACE create configmap dynamic-common-env \
  --from-env-file=.env.k8s \
  --dry-run=client -o yaml | kubectl apply -f -
```

Загрузка секретов:

```
kubectl -n $NAMESPACE create secret generic dynamic-common-secrets \
  --from-env-file=.env.k8s.secrets \
  --dry-run=client -o yaml | kubectl apply -f -
```

Примечание: при необходимости можно создавать дополнительные `<service>-env` и `<service>-secrets` для точечных настроек отдельных сервисов.

5.6. Развертывание сервисов

```
export REGISTRY=registry.example.com/dynamic
export IMAGE_TAG=20260216-r1
export NAMESPACE=dynamic
./deployment/kubernetes_scripts/deploy-k8s.sh
```

Скрипт автоматически:

- создает/обновляет Kubernetes `Deployment` и `Service`;
- подключает общие и сервисные параметры окружения;
- выполняет `rollout status` каждого сервиса.

5.7. Проверка результата развертывания

Проверка deployment и pod:

```
kubectl -n $NAMESPACE get deploy
kubectl -n $NAMESPACE get pods
```

Проверка сервисов:

```
kubectl -n $NAMESPACE get svc
```

Проверка событий:

```
kubectl -n $NAMESPACE get events --sort-by=.metadata.creationTimestamp
```

Критерии успешного развертывания:

- все Deployment имеют `AVAILABLE` не ниже заданного числа реплик;
- отсутствуют состояния `ImagePullBackOff`, `CrashLoopBackOff`, `CreateContainerConfigErro`
- сервисы зарегистрированы в namespace и доступны по ClusterIP.

6. Доступ к демо-стенду

- Хост системы: https://api-staging-zeta.calatrava.travel/dynamics

Пример запроса авторизации:

```
curl --location 'https://api-staging-zeta.calatrava.travel/login' \
--header 'Gateway-Version: operator' \
--header 'Content-Type: application/json' \
--data-raw '{
    "email": "*****",
    "password": "*****"
}
'
```