Deckhouse Platform для разработчиков:

миграция приложений в кластер Deckhouse Platform

Онлайн-курс 10 дней

Аудитория курса

- Разработчики приложений
- DevOps-инженеры

Требования к участникам

- Знать Linux на уровне пользователя
- Понимать типовые архитектурные решения, которые используются при развёртывании приложений (монолит, распределённый монолит, микросервисная архитектура)
- Знать основные понятия и сущности Kubernetes (pod, deployment, service, Ingress, PV/PVC, StorageClass)
- Уметь работать с утилитами kubectl и d8

Цели курса

- Получить знания о подходах к миграции приложений в Deckhouse Platform
- Приобрести базовые навыки автоматизации сборки и развёртывания приложения, используя инфраструктуру и инструменты Deckhouse Platform

Формат

- Курс состоит из теоретического материала и практической части с выполнением лабораторных работ на учебном стенде

Deckhouse Platform

XX XXX ₽

План работы

Тема 01

Цели и ограничения миграции

Продолжительность: 8 ак. часов

Цель: получить знания о современных архитектурных решениях при развёртывании приложений, узнать об особенностях Cloud Native App-подхода, проанализировать цели и ограничения миграции.

Теория:

- Cloud Native App
- Микросервисная архитектура
- Использование единой среды для сборки, тестирования и развёртывания
- Автоматизация процессов поставки и развёртывания
- Масштабирование
- Упор на облачные сервисы
- Ограничения наследуемой архитектуры
- Что делать, если приложение — не Cloud Native App
- Масштабирование монолита и распределённого монолита
- Преимущества использования Deckhouse Platform для миграции, автоматизация пятого уровня
- Экосистема Deckhouse: Deckhouse Platform, Deckhouse Code, Deckhouse Observability, Deckhouse Development Portal, Deckhouse Stronghold
- Преобразование монолита в распределённую структуру
- Преимущества разделения монолита
- Проблемные места реализации
- Рекомендуемые практики и подход Cloud Native App

Тема 02

Реализация проекта. Сборка образов приложения. Использование delivery kit для локальной разработки

Продолжительность: 8 ак. часов

Цель: научиться использовать утилиту delivery-kit для сборки образов сервисов приложения и локальной разработки.

Теория:

- Контейнерный образ
- Образ как единый артефакт для всех окружений
- Что такое хороший образ, лучшие практики в сборке образов
- Многоэтапная сборка
- Использование образа «с нуля»
- Distroless-образы
- Использование .dockerignore
- Уменьшение количества слоёв
- Использование линтера
- Rootless-контейнеры
- Использование чувствительных данных
- Жизненный цикл образа
- Сборка образов с delivery-kit
- Процесс сборки
- Определение структуры приложения для локального развёртывания
- Использование d8 dk compose
- Преобразование приложения для использования с управляемыми сервисами Deckhouse

Практика:

- Сборка образов сервисов приложения с delivery-kit
- Развёртывание приложения локально в режиме разработки

Тема 03

Подготовка манифестов для развёртывания приложения в кластере Deckhouse Platform

Продолжительность: 8 ак. часов

Цель: подготовить манифесты, развернуть приложение в кластере Deckhouse Platform в ручном режиме.

Теория:

- Структура приложения RSS/Atom-агрегатора
- От контейнера к поду
- От пода к специализированным развёртываниям
- Statefulset — запуск сервисов с сохранением состояния
- Job — выполнение разовых задач
- CronJob — выполнение задач по расписанию
- DaemonSet
- Выбор подходящего типа объекта для сервисов, создание манифестов
- Применение манифестов, создание объектов в кластере
- Подключение управляемых сервисов
- Init-контейнеры

Практика:

- Тегирование образов фронтенда и бэкенда
- Создание неймспейса проекта
- Создание секрета для доступа к реестру образов
- Создание управляемых сервисов
- Применение манифестов фронтенда и бэкенда
- Проверка работы приложения

Тема 04

Настройка сетевой связности и внешнего доступа к приложению

Продолжительность: 8 ак. часов

Цель: научиться настраивать сетевую связность между компонентами приложения и сделать приложение доступным извне кластера.

Теория:

- CNI — основа для сетевой связности в кластере
- Для чего нужны Services
- kube-proxy
- kube-proxy vs CNI Cilium
- Типы сервисов: ClusterIP, NodePort, LoadBalancer, ExternalName
- Headless Service
- Модуль service-with-healthchecks
- Выбор подходящих типов сервисов для компонентов приложения
- Настройка внешнего доступа к приложению и доступа по HTTPS
- Настройка Ingress
- Решение проблем сетевой связности сервисов и доступа к приложению

Практика:

- Создание сервисов бэкенда и фронтенда
- Проверка работы приложения изнутри кластера по имени сервиса
- Внешний доступ к приложению в режиме разработки
- Настройка Ingress
- Настройка TLS

Тема 05

Обеспечение высокой доступности, масштабирование

Продолжительность: 8 ак. часов

Цель: научиться делать работу приложения стабильной, автоматизировать масштабирование сервисов фронтенда и бэкенда

Теория:

- Возможности Deckhouse Platform для автомасштабирования
- Высокая доступность на уровне кластера: автовосстановление узлов, управление сертификатами и компонентами
- Высокая доступность и отказоустойчивость приложения (Kubernetes-объекты): реплики, LivenessProbe, ReadinessProbe, PodDisruptionBudget, podAntiAffinity
- Автомасштабирование (HPA и VPA)
- Определение начальных значений resources для сервисов
- Лучшие практики для определения requests и limits
- Отказоустойчивость внешнего трафика
- Масштабирование управляемых сервисов

Практика:

- Настройки для обеспечения высокой доступности и отказоустойчивости приложения
- Увеличение количества реплик
- Настройка LivenessProbe и ReadinessProbe
- PodDisruptionBudget
- PodAntiAffinity
- Определение начальных значений resources по нагрузочному тестированию
- Настройка HPA
- Настройка VPA в режиме получения рекомендаций

Тема 06

Автоматизация развёртывания приложения с delivery-kit

Продолжительность: 8 ак. часов

Цель: автоматизировать развёртывание приложения в кластере Deckhouse Platform с помощью утилиты delivery-kit.

Теория:

- Использование delivery kit для развёртывания
- Чарты и шаблонизация
- Расширенные приёмы шаблонизации: if-else, helpers, range
- Релиз приложения
- Работа с чувствительными данными
- Определение порядка создания ресурсов
- Использование хуков, миграции
- Конфигурация delivery kit для развёртывания
- Процесс развёртывания
- Команды подготовки и развёртывания: d8 dk render, d8 dk plan, d8 dk converge
- Подготовка окружения для развёртывания
- Развёртывание приложения
- Подстановка параметров в зависимости от заданного окружения
- Подготовка продуктового окружения для развёртывания
- Развёртывание в продуктовом окружении
- Развёртывание приложения в режиме разработки
- Удаление релиза

Практика:

- Анализ структуры чарта
- Шаблонизация манифестов
- Использование хелпера, range
- Использование секретов delivery-kit
- Определение порядка создания ресурсов
- Конфигурация delivery kit для развёртывания
- Подготовка к развёртыванию и развёртывание приложения
- Развёртывание приложения в production-окружении
- Работа с приложением в режиме разработки
- Удаление релиза

Тема 07-1

Построение базового пайплайна CI/CD

Продолжительность: 8 ак. часов

Цель: настроить базовый пайплайн для развёртывания приложения RSS/Atom-агрегатора в кластере Deckhouse Platform с использованием Deckhouse Code и научиться решать проблемы при его использовании.

Теория:

- Что такое CI/CD
- Принципы реализации CI/CD
- Deckhouse Code
- Построение базового пайплайна
- Решение проблем
- Использование окружений Deckhouse Code
- Раннеры
- Артефакты
- Кеширование
- Структура базового пайплайна, этапы: проверка Dockerfile с Hadolint, юнит-тестирование приложения, сборка образов, сканирование образов на проблемы с безопасностью, интеграционное тестирование, релиз, развёртывание
- Выполнение пайплайна

Практика:

- Интерфейс пайплайнов Deckhouse Code
- Файл пайплайна .gitlab-ci.yml
- Этапы и задачи (jobs) пайплайна
- Определение параметров проекта
- Выполнение пайплайна
- Просмотр логов и артефактов
- Вывод лога задачи в режиме отладки
- Интерфейс окружений Deckhouse Code

Тема 07-2

Развитие проекта CI/CD

Продолжительность: 8 ак. часов

Цель: добавить в пайплайн возможность развёртывания приложения в различных окружениях.

Теория:

- Модели ветвления
- Триггеры пайплайна
- Организация процесса разработки
- Динамические окружения
- Построение финальной версии пайплайна
- Пайплайны для развёртывания в preview-, staging- и production-окружении
- Оптимизация пайплайна
- Использование темплейтов, якорей YAML, extends, !reference
- Вынос темплейтов в подключаемые файлы
- Использование resource group
- Создание динамических preview-окружений
- Организация процесса разработки с последовательным развёртыванием приложения в окружениях preview, staging и production
- Очистка реестра образов по расписанию

Практика:

- Добавление задачи на проверку работоспособности приложения после развёртывания
- Отработка полного цикла внесения оперативных корректировок в проект (хотфиксов)
- Развёртывание приложения в preview-, staging- и production-окружениях

Тема 08

Использование секретных данных. Deckhouse Stronghold

Продолжительность: 8 ак. часов

Цель: настроить Deckhouse Stronghold для работы с пайплайном развёртывания приложения.

Теория:

- Обзор методов работы с секретами в проекте
- Deckhouse Stronghold: теория и конфигурация
- Интеграция Deckhouse Stronghold с пайплайном
- Секреты delivery-kit
- CI/CD-переменные в проекте Deckhouse Code
- Использование хранилища секретов
- Параметры безопасности для переменных CI/CD
- Проблемы интеграции GitLab CE + Vault
- Основные возможности и сценарии использования Deckhouse Stronghold
- JWT-аутентификация Stronghold
- Преимущества использования JWT-аутентификации
- Конфигурация пайплайна для получения секретов

Практика:

- Добавление в темплейт логики получения секретов из Deckhouse Stronghold
- Подключение файла темплейта в пайплайн
- Подключение кода получения секретов в задачах
- Создание хранилища KV-v2 в Stronghold
- Добавление секретов в Stronghold
- Создание политик доступа ACL к секретам
- Конфигурация метода аутентификации по JWT
- Проверка интеграции

Тема 09

Мониторинг приложения и использование логов

Продолжительность: 8 ак. часов

Цель: научиться использовать метрики приложения и компонентов кластера, настроить алертинг, проработать подходы к расследованию инцидентов.

Теория:

- Определение целей мониторинга
- Мониторинг компонентов приложения в кластере, баз данных, приложения, периодически запускаемого сервиса
- Модуль extended-monitoring
- Выполнение запросов PromQL
- Использование метрик приложения
- Экспорт метрик приложения в Prom++
- Настройка модуля Prometheus Pushgateway
- Четыре золотых сигнала мониторинга приложения
- Условия для срабатывания алертов
- Настройка алертинга
- Настройка логирования
- Выборка логов
- Расследование инцидентов

Практика:

- Настройка экспорта метрик приложения
- Конфигурация модуля Prometheus Pushgateway для периодически запускаемого сервиса
- Подключение расширенного мониторинга
- Настройка алертов и уведомлений через Webhook
- Настройка сбора логов
- Расследование инцидента