CD с Ansible, Docker, Docker Compose, Kubernetes

CI/CD на основе GitLab

REC **Проверить, идет ли запись**

Меня хорошо видно & слышно?

Кузнецов Алексей

cloud engineer

- 14 лет в IT
- 9 лет devops инженер
- создавал и поддерживал инфраструктуру для приложений в 3х крупный компаниях
- делаю вклад в open source
- выступаю на конференциях

@windblow

koyotne@bk.com

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в учебной группе #**канал группы**

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Карта курса

СТАРТ

CI/CD - системы, подходы и workflow

Gitlab CI

Безопасность

Проектная работа

ФИНИШ

Маршрут вебинара

- Знакомство
- CD с Docker
- CD с Docker Compose
- CD с Kubernetes
- Практика
- Рефлексия

Цели вебинара

К концу занятия вы сможете

1. Изучить использование различных инструментов для автоматизации деплоя, таких как Docker, Docker Compose, и Kubernetes, в процессе непрерывной доставки (CD).

2. Ознакомиться с интеграцией CI/CD пайплайнов с GitLab CI для автоматического тестирования и развертывания приложений на различных инфраструктурах.

3. Понять подходы к развертыванию и управлению приложениями в разных средах с использованием контейнерных технологий с оркетрацией.

Смысл

Это нужно уметь чтобы:

1. Уметь описывать CI/CD процессы

2. Идентифицировать тип развертывания

3. Подобрать среду для CD в зависимости от ваших задач

Кто из вас уже использует автоматизацию для деплоя приложений, и какие инструменты вы применяете?

CD c Docker

Docker – это

платформа для контейнеризации приложений, которая позволяет упаковывать приложения и их зависимости в контейнеры, обеспечивая их переносимость и изоляцию.

Docker

В контексте Continuous Deployment (CD), Docker позволяет автоматизировать процесс доставки приложений, ускоряя их развертывание на разных средах.

Плюсы Docker для CD:

- Изолированность окружения.
- Переносимость.
- Автоматизация.

Docker

Работа CD и Docker:

- Сборка Docker-образа
- Docker Registry
- Деплой контейнера

Основные этапы CD с Docker:

1. Сборка Docker-образа

```
FROM python:3.8-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY ..
CMD ['python’, "app.py"]
```

Docker

2. GitLab CI/CD пайплайн для Docker.

GitLab CI/CD может автоматически запускать сборку Docker-образов и деплой.

Пример файла .gitlab-ci.yml для CI/CD с Docker:

```
stages:
      — build
      — deploy

build: # собирает Docker-образ и отправляет его в реестр
      stage: build
      script:
            — docker build -t my_app:${CI_COMMISTHA}
            — docker push myregistry.com/my_app:${CI_COMMSIHTA}

deploy: # загружает ofpas из реестра и запускает его на сервере
      stage: deploy
      script:
            — docker pull myregistry.com/my_app:${CI_COMMSIHTA}
            — docker run -d -p 80:80 myregistry.com/my app:${CI_COMMIT SHA}
```

Docker

3. Docker Registry.

Чтобы деплой проходил на разных средах, Docker-образ должен быть доступен в реестре. Это может быть:

- Docker Hub — публичный реестр.
- GitLab Container Registry — приватный реестр в GitLab.
- Частный Docker-registry — можно развернуть собственный реестр на сервере.

Команды для работы с Docker-реестром:

```
docker build: Сборка образа
docker build -t myregistry.com/my app:latest

docker push: Загрузка образа в реестр
docker push myregistry.com/my app:latest

docker pull: Загрузка образа из реестра на целевой сервер
docker pull myregistry.com/my app:latest
```

Docker

4. Деплой контейнера.

Для деплоя Docker-контейнера на продакшн используется команда docker run. Пример деплоя веб-приложения:

```
docker run -d -p 80:80 myregistry.com/my app:latest
--d: Запуск контейнера в фоновом режиме.
--p 80:80: Проброс порта 80 для доступа к приложению из браузера.
```

5. Обновление контейнера.

CO с Docker позволяет легко обновлять приложения. Чтобы развернуть новую версию, достаточно перезапустить контейнер с новым образом:

```
docker pull myregistry.com/my_ app:new_version
docker stop my_app
docker rm my_app
docker run -d -p 80:80 myregistry.com/my_app:new_version
```

Kaniko: Сборка Docker-образов в CI/CD

Kaniko —это инструмент от Google для сборки Docker-образов в средах, где использование Docker-демона невозможно или нежелательно (например, в CI/CD пайплайнах или в Kubernetes-кластерах).

Почему Kaniko подходит для CD?
Как работает Kaniko?
Когда использовать Kaniko?
Какие ограничения у Kaniko?

Kaniko: Сборка Docker-образов в CI/CD

Пример интеграции Kaniko в GitLab CI/CD

```
stages:
  - build

build:
  stage: build
  image: gcr.io/kaniko-project/executor:latest
  script:
    - /kaniko/executor
      --context $CI_PROJECT_DIR
      --dockerfile $CI_PROJECT_DIR/Dockerfile
      --destination myregistry.com/my_app:$CI_COMMIT_SHA
  only:
    - main
```

Kaniko: Сборка Docker-образов в CI/CD

Пример настройки Kaniko с секретами

```
build:
  stage: build
  image: gcr.io/kaniko-project/executor:latest
  script:
    - echo
    "{\"auths\":{\"myregistry.com\":{\"username\":\"$REGISTRY_USER\",\"password\":\"$
    REGISTRY_PASSWORD\"}}}" > /kaniko/.docker/config.json
    - /kaniko/executor --context $CI_PROJECT_DIR --dockerfile
    $CI_PROJECT_DIR/Dockerfile --destination myregistry.com/my_app:$CI_COMMIT_SHA
```

Buildkit

```
docker buildx build --push -t <username>/hello-buildkit .
```

Вопросы

+ если есть вопросы

– если вопросов нет

CD c Docker Compose

Docker Compose – это

инструмент для определения и запуска multi-container Docker-приложений.

Docker Compose

Преимущества Docker Compose с CD:

- Приложение с зависимостями или мультикомпонентное
- Простота в использовании.
- Легкая интеграция с CI/CD.

Работа СО с Docker Compose:

- Определение приложения
- Сборка образов
- Автоматизация деплоя

Docker Compose

1. Создание docker-compose.yml

```
version: '3'
services:
      web: # сервис приложения, который собирается из локального Dockerfile и публикует порт
         80.
            build:
            ports:
                  - "80:80"
            environment:
                  - ENV=production
            depends_on:
                  - db
      db: # сервис базы данных PoStgreSQL с переменными окружения для настройки
            image: postgres:13
            environment:
                  POSTGRES_USER: user
                  POSTGRES_PASSWORD: password
                  POSTGRES_DB: mydb
      redis: # сервис кеша Redis
            image: redis:6
            ports:
                  - "6379:6379"
```

Docker Compose

2. Интеграция Docker Compose В пайплайн CI/CD

Пример файла .gitlab-ci.yml с интеграцией Docker Compose для автоматического деплоя:

```
stages:
	- build
	- test
	- deploy

build: # собирает docker-образы всех сервисов приложения
	stage: build
	script:
		— docker-compose build

test: # запускает контейнеры для выполнения тестов (например, тесты приложения внутри контейнера web)
	stage: test
	script:
		— docker-compose up -d
		— docker-compose exec web pytest

deploy: # выполняет команду docker-compose up -d, которая развертывает приложение в фоновом режиме
	stage: deploy
	script:
		— docker-compose up -d
```

Docker Compose

3. Автоматическое тестирование В контейнерах.
Docker Compose позволяет не только деплоить, но и тестировать приложение в контейнерах. Например, вы можете запустить ваше приложение и выполнить тесты, используя команды.

```
docker-compose up -d
docker-compose exec мер pytest
```

Это позволяет убедиться, что приложение работает корректно перед его деплоем.

Docker Compose

4. Деплой с помощью Docker Compose

Когда пайплайн прошел все стадии (сборку и тестирование), следующий этап — это автоматический деплой.

Docker Compose позволяет деплоить приложения на сервере одной командой:

```
docker-compose up -d
-d: Запускает контейнеры в фоновом режиме.
up: Создает и запускает все сервисы, описанные в docker-compose.yml.
```

Если Docker Compose уже запущен, то up -d обновит контейнеры до последних версий образов.

Docker Compose

5. Обновление контейнеров

Docker Compose позволяет легко обновлять контейнеры с новыми версиями. Когда вы вносите изменения в код приложения или его зависимости, вам достаточно перезапустить сервисы:

```
docker-compose pull # Получить новые версии образов
docker-compose пр -d # Обновить контейнеры
```

Эти команды остановят старые контейнеры и заменят их новыми, минимизируя простои.

Docker Compose

6. Управление секретами и конфигурациями

Docker Compose поддерживает управление конфигурациями и секретами, что особенно полезно для CD.

```
POSTGRES_USER=user
POSTGRES_PASSWORD=secretpassword
Затем их можно использовать в docker-compose.yml:
version: '3'
services:
     db:
          image: postgres:13
          environment:
               — POSTGRES_USER=${POSTGRES_USER}
               — POSTGRES_PASSWORD=$ {POSTGRES_PASSWORD}
```

Docker Compose

7. Деплой на продакшн.

Docker Compose можно использовать не только для разработки и тестирования, но и для продакшн-окружений. Если ваше приложение готово к релизу, вы можете развернуть его на сервере, где установлен Docker Compose.

Пример продакшн-деплоя:

```
1. Логин в Docker-реестр:
docker login myregistry.com

2. Получение обновленных образов:
docker-compose pull

3. Запуск приложения:
docker-compose up -d

4. Проверка статуса контейнеров:
docker-compose ps
```

Docker Compose

Расширенные возможности Docker Compose для CO.

1. Горизонтальное масштабирование.

```
docker-compose up --scale web=3 -d
# это создаст 3 экземпляра сервиса web, что полезно для балансировки нагрузки.
```

2. Blue-Green Deployment.

- Запускается новый стек с новым набором контейнеров.
- Перенаправляется трафик на новый стек после тестирования.

3. Поддержка разных окружений

```
docker-compose -f docker-compose.yml -f docker-compose.prod.yml up -d
```

Вопросы

+ если есть вопросы

– если вопросов нет

CD с Kubernetes

Kubernetes (K8s) – это

ведущая платформа для оркестрации контейнеров, которая автоматизирует развертывание, управление и масштабирование cloud native приложений.

CD с Kubernetes

Преимущества использования Kubernetes для CD:

- Автоматизация развертывания.
- Самовосстановление.
- Масштабирование.
- Конфигурации и секреты.
- Мониторинг и журналирование.

Основные концепции CD в Kubernetes:

- Deployment.
- Service.
- ConfigMap и Secret.
- Ingress.

CD с Kubernetes

Работа CD с Kubernetes:

- Определение манифестов.
- CICD пайплайн.
- Автоматическое обновление.
- Мониторинг и обратная связь.

Основные этапы CO с Kubernetes:

1. Создание манифестов для развертывания.
Каждое приложение в Kubernetes описывается в YAML-MaHudecTax, таких как deployment.yaml, service.yaml, ingress.yaml.

CD с Kubernetes

Пример манифеста для Deployment:

```
apiversion: apps/vl
kind: Deployment
metadata:
    name: web-app
spec:
    replicas: 3 # количество экземпляров приложения (3 пода)
    selector:
        matchLabels:
            app: web
    template:
        metadata:
            labels:
                app: web
        spec:
            containers:
            - name: web
                image: myregistry.com/web-app:latest # docker-образ, который используется для
                      развертывания контейнера
                ports:
                — containerPort: 80
                env: # переменные окружения для конфигурации приложения
                — name: ENVIRONMENT
                    value: production
```

CD с Kubernetes

2. Rolling Updates.

Kubernetes поддерживает стратегию Rolling Updates, которая позволяет обновлять приложение без простоя.

Чтобы применить обновление, достаточно обновить манифест и выполнить команду:

```
kubectl apply -f deployment.yaml
```

CD с Kubernetes

3. Blue-Green Deployment.
Blue-Green Deployment — это стратегия, при которой создается новая версия приложения (Green), параллельно с текущей (Blue).

Пример реализации Blue-Green Deployment.
Создайте два Deployment для приложения:

```
apiVersion: vl
kind: Service
metadata:
    name: web-app
spec:
    selector:
        app: blue-app # или green-app
    ports:
    - protocol: TCP
        port: 80
        targetPort: 80
```

CD с Kubernetes

4. Интеграция Kubernetes с CI/CD

Интеграция Kubernetes в CI/CD пайплайн позволяет автоматически развертывать приложения при каждом коммите в репозиторий.

```
stages:
    - build
    - deploy

variables:
    KUBERNETES_NAMESPACE: production
    KUBECONFIG: "/path/to/kubeconfig"

build: # собирает Docker-образ и загружает его в Docker Registry.
    stage: build
    script:
        - docker build -t myregistry.com/web-app:latest .
        - docker push myregistry.com/web-app:latest

deploy: # применяет манифесты Kubernetes, чтобы обновить приложение на кластере
    stage: deploy
    script:
        - kubectl apply -f k8s/deployment.yaml
```

CD с Kubernetes

5. Использование Helm для управления релизами.
Helm — это менеджер пакетов для Kubernetes, который упрощает управление сложными приложениями через шаблоны.

Пример команды для развертывания приложения с помощью Helm:

```
helm upgrade --install my-app ./helm-chart --namespace production
```

CD с Kubernetes

6. Управление секретами и конфигурациями.
Kubernetes поддерживает ConfigMap и Secrets для управления конфигурацией приложений и секретами (например, паролями).

```
apiVersion: v1
kind: Secret
metadata:
    name: db-secret
type: Opaque
data:
    password: c2VjcmV0 # Base64
```

CD с Kubernetes

7. Canary Deployment

Canary Deployment — это стратегия, при которой новая версия приложения развертывается только для части пользователей.
Для примера можно взять Istio или NGINX Ingress Controller для реализации Canary Deployment.

Настраивается два Deployment: одну для старой версии и одну для новой.
Используйте Ingress для распределения трафика (например, 90% на старую версию и 10% на новую).

Вопросы

+ если есть вопросы

– если вопросов нет

Практика

Этапы практики

1. Создали Docker-образ для простого приложения.
2. Использовали Docker Compose для локальной разработки.
3. Развернули приложение в Kubernetes.
4. Интегрировали CI/CD пайплайн с GitLab для автоматизации деплоя.

Ключевые тезисы

1. **Интеграция Docker и Docker Compose в CD-процессы:** Docker и Docker Compose позволяют эффективно автоматизировать развертывание приложений в контейнерах, обеспечивая лёгкую настройку окружения и совместимость между разработкой и продакшеном. Использование этих инструментов в GitLab CI/CD ускоряет процессы доставки приложений и снижает риск ошибок, связанных с конфигурацией окружения.

2. **Автоматизированный деплой с помощью CI/CD:** Непрерывная доставка (CD) с использованием GitLab CI/CD и Docker-compose и k8 позволяет управлять полным циклом развертывания — от сборки образов до тестирования и автоматизированного удаления контейнеров. Это делает процесс повторяемым, безопасным и легко масштабируемым на различные среды (development, staging, production).

Вопросы

+ если есть вопросы

– если вопросов нет

Домашнее задание

Инструкция:

Напишите CO собранного вами в предыдущем ДЗ приложения на сервер.
Деплой должен происходить в Docker или Docker Compose.
Напишите CO приложения, деплой должен происходить в Kubernetes.
Ссылка на приложение: https://github.com/matheusfm/httpbin-chart

Формат сдачи: отдельные файлы gitlab-ci.yml для каждого пункта задания или ссылки на ваши проекты.

Подведём итоги занятия

Теперь вы сможете:

Отправьте в чат номера достигнутых целей

1. Изучить использование различных инструментов для автоматизации деплоя, таких как Docker, Docker Compose, и Kubernetes, в процессе непрерывной доставки (CD).

2. Ознакомиться с интеграцией CI/CD пайплайнов с GitLab CI для автоматического тестирования и развертывания приложений на различных инфраструктурах.

3. Понять подходы к развертыванию и управлению приложениями в разных средах с использованием контейнерных и оркестрационных технологий.

Вопросы для проверки

1. Какие преимущества даёт использование docker-compose по сравнению с одиночными Docker-командами при деплое многокомпонентных приложений?
2. Как работает пайплайн CD с использованием docker-compose?
3. Как работает интеграция Docker Registry с CD, и зачем нужен?

Отправьте в чат эмодзи, который отражает ваше настроение после занятия

Продолжите высказывание: теперь я умею/знаю...

Что планируете использовать в работе?

Заполните, пожалуйста, опрос о занятии

Мы читаем все ваши сообщения
и берем их в работу ✏️❤️

СКЛАДЧИК

ЭТОТ КОНТЕНТ КУПЛЕН НА САЙТЕ SKLADCHIK.ORG

SKL.TO

ТЫ ЕЩЕ НЕ В КЛУБЕ?
 присоединяйся и начинай экономить!

ЧТО ТАКОЕ КЛУБ «СКЛАДЧИК»?

Платформа
Это платформа, где каждый день тысячи людей собираются вместе, чтобы общими усилиями находить, приобретать и изучать курсы интересные именно им.

Сообщество
Это сообщество из 500 000 людей, открывших для себя выгодный способ быть в тренде самых актуальных и получать ценные знания по минимальной цене.

Библиотека
Это крупнейшая библиотека инфопродуктов в которой можно найти практически любой курс или тренинг продававшийся за последние 10 лет, а также уникальные авторские инфопродукты, которые не получиться найти больше нигде.