12. Сопровождение и эксплуатация ПО

1. Логирование (Centralized Logging)

Проблема: В микросервисной архитектуре логи хранятся в разных местах → сложно анализировать.
Решение: **Централизованное логирование** — сбор, хранение и анализ логов в одном месте.

ELK-стек (Elasticsearch + Logstash + Kibana)

- **Elasticsearch** – поиск и индексация логов (NoSQL-база).
- **Logstash** – сбор и обработка логов (фильтры, парсинг).
- **Kibana** – визуализация (дашборды, графики).

Как работает?

1. Приложение → Logstash (или Filebeat) → Elasticsearch.
2. Админ смотрит логи в Kibana.

Пример конфига Logstash:

```
input { file { path => "/var/log/app.log" } }
filter { grok { match => { "message" => "%{TIMESTAMP:date} %{LOGLEVEL:level}
%{GREEDYDATA:msg}" } } }
output { elasticsearch { hosts => ["elasticsearch:9200"] } }
```

Grafana Loki

- **Loki** – хранение логов (как Prometheus, но для логов).
- **Grafana** – визуализация (может показывать и логи, и метрики).
- Экономичнее ELK (индексирует только метаданные).

Пример запроса в Grafana:

```
{app="order-service"} |= "error"
```

---

2. Мониторинг (Monitoring & Alerting)

Проблема: Как понять, что сервис упал или тормозит?
Решение: **Сбор метрик + алертинг**.

Prometheus

- **Pull-модель** (сам собирает метрики с сервисов).
- **Метрики**: CPU, RAM, HTTP-запросы, кастомные метрики.
- **PromQL** – язык запросов.

Пример метрики в Spring Boot (Micrometer):

```
@RestController
public class OrderController {
    private final Counter requestCount = Metrics.counter("order.requests");

    @GetMapping("/order")
    public String createOrder() {
        requestCount.increment();
        return "Order created!";
    }
}
```

Grafana

- Визуализация метрик из Prometheus (и других источников).
- **Дашборды**: графики Latency, Error Rate, Traffic.

Пример PromQL в Grafana:

```
http_requests_total{status="500"} / http_requests_total * 100
```

Alerting:

- Настраивается в Grafana или Alertmanager (Prometheus).
- Уведомления в Slack/Telegram/Email.

---

3. CI/CD (Непрерывная интеграция и доставка)

Проблема: Ручные деплои → ошибки, медленно.
Решение: **Автоматизация сборки, тестирования и деплоя**.

Jenkins

- **Оркестратор CI/CD** (можно настроить любые пайплайны).
- **Плюсы**: Гибкость, множество плагинов.
- **Минусы**: Требует поддержки сервера.

Пример Jenkinsfile (Declarative Pipeline):

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps { sh 'mvn package' }
        }
        stage('Test') {
            steps { sh 'mvn test' }
        }
        stage('Deploy') {
            steps { sh 'kubectl apply -f k8s-deployment.yaml' }
        }
    }
}
```

GitLab CI / GitHub Actions

- **Встроенные в Git-платформы** (не нужен отдельный сервер).
- **Плюсы**: Проще настройка, интеграция с репозиторием.

Пример .gitlab-ci.yml:

```
stages:
  - build
  - deploy

build-job:
  stage: build
  script:
    - mvn package

deploy-job:
  stage: deploy
  script:
    - kubectl apply -f k8s.yaml
```

Пример GitHub Actions:

```
name: CI/CD Pipeline
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - run: mvn package
  deploy:
    needs: build
    run: kubectl apply -f k8s.yaml
```

---

4. DevOps: Инфраструктура как код (IaC)

Проблема: Ручное управление серверами → дрейф конфигурации.
Решение: **Автоматизация через код**.

Docker

- **Контейнеризация приложений** (изоляция зависимостей).

Пример Dockerfile:

```
FROM openjdk:17
COPY target/app.jar /app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

Kubernetes (K8s)

- **Оркестрация контейнеров** (масштабирование, self-healing).

Пример deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: order-service
spec:
  replicas: 3
  template:
    spec:
      containers:
        - name: order-service
          image: my-registry/order-service:latest
          ports:
            - containerPort: 8080
```

Terraform

- **Провайдер-независимая инфраструктура** (AWS, GCP, Azure).
- Декларативное описание серверов, сетей, БД.

Пример main.tf (AWS EC2):

```
resource "aws_instance" "app_server" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
  tags = {
    Name = "MyAppServer"
  }
}
```

Сравнение инструментов

Категория	Инструменты	Когда выбирать?
Логирование	ELK, Loki + Grafana	ELK — для сложного анализа, Loki — для экономии ресурсов
Мониторинг	Prometheus + Grafana	Стандарт для Kubernetes-кластеров
CI/CD	Jenkins, GitLab CI, GitHub Actions	Jenkins — для сложных пайплайнов, GitLab/GitHub — для интеграции с репозиторием
DevOps	Docker, K8s, Terraform	Docker+K8s — контейнеризация, Terraform — управление облаком

Итог

1. **Логирование**: ELK/Loki + Grafana для поиска проблем.
2. **Мониторинг**: Prometheus + Grafana для метрик и алертов.
3. **CI/CD**: Автоматизируйте сборку и деплой (Jenkins/GitLab).

4. **DevOps**: IaC (Terraform) и оркестрация (K8s) — must-have для продакшена.

Эти инструменты сокращают время на рутинные задачи и повышают надежность системы! 🚀