Вот список основных сервисов и инструментов, которыми должен владеть DevOps инженер, с кратким описанием:

---

1. Управление инфраструктурой

1. **Terraform**
   Инструмент для управления инфраструктурой как кодом (IaC). Позволяет описывать инфраструктуру с помощью конфигурационных файлов и управлять её жизненным циклом.

2. **Ansible**
   Автоматизация задач конфигурации, развертывания и управления серверами. Использует декларативные YAML-плейбуки.

3. **Puppet** / **Chef**
   Средства управления конфигурацией для автоматизации развертывания серверов и приложений.

4. **AWS CloudFormation**
   Инструмент для управления ресурсами AWS с помощью шаблонов.

5. **Kubernetes (K8s)**
   Система оркестрации контейнеров для автоматизации развертывания, масштабирования и управления контейнерными приложениями.

---

2. Контейнеризация

1. **Docker**
   Платформа для создания, упаковки и управления контейнерами, обеспечивающая изоляцию приложений.

2. **Podman**
   Альтернатива Docker, ориентированная на безопасность и отказ от daemon-архитектуры.

---

3. CI/CD

1. **Jenkins**
   Инструмент для создания CI/CD пайплайнов. Поддерживает автоматизацию сборки, тестирования и деплоя.

2. **GitLab CI/CD**
   Встроенная CI/CD система в GitLab для управления процессом доставки и развёртывания кода.

3. **CircleCI**
   Платформа CI/CD, ориентированная на контейнеризацию и поддержку современных DevOps практик.

4. **ArgoCD**
   Инструмент для реализации GitOps в Kubernetes, автоматизирует развертывание приложений

из git-репозиториев.

---

4. Облачные платформы

1. **Amazon Web Services (AWS)**
   Лидер рынка облачных технологий, предоставляющий широкий набор инструментов для DevOps (EC2, S3, RDS, EKS и др.).

2. **Microsoft Azure**
   Платформа облачных сервисов, предоставляющая поддержку DevOps через инструменты вроде Azure DevOps и Kubernetes Service.

3. **Google Cloud Platform (GCP)**
   Инструменты для развертывания и управления облачными приложениями, такие как GKE, Cloud Build.

4. **DigitalOcean**
   Простая облачная платформа, подходящая для малого и среднего бизнеса, с поддержкой Kubernetes и автоматического масштабирования.

5. **HashiCorp Vault**
   Система для безопасного управления секретами и доступами.

---

5. Мониторинг и логирование

1. **Prometheus**
   Система мониторинга и алертинга, основанная на сборе метрик с экспортеров.

2. **Grafana**
   Инструмент визуализации данных из источников (Prometheus, ElasticSearch, Loki).

3. **ELK Stack (Elasticsearch, Logstash, Kibana)**
   Стек для сбора, обработки и визуализации логов.

4. **Loki**
   Лог-агрегатор, интегрируемый с Grafana для удобного анализа логов.

5. **Datadog**
   SaaS-платформа для мониторинга и аналитики инфраструктуры и приложений.

6. **Zabbix**
   Система для мониторинга серверов, приложений и сетей.

7. **New Relic**
   SaaS-инструмент для мониторинга производительности приложений (APM).

---

6. Системы управления версиями

1. **Git**
   Система контроля версий. Основной инструмент для работы с репозиториями кода.

2. **GitHub / GitLab / Bitbucket**
   Веб-платформы для хостинга репозиториев и интеграции CI/CD.

---

7. Сетевые инструменты

1. **Nginx**
   Высокопроизводительный веб-сервер и обратный прокси для балансировки нагрузки.

2. **HAProxy**
   Программный балансировщик нагрузки с поддержкой масштабирования.

3. **Traefik**
   Реверс-прокси и балансировщик нагрузки для динамического управления маршрутизацией.

---

8. Управление конфигурацией

1. **Consul**
   Система для управления конфигурацией и сервисами.

2. **Etcd**
   Хранилище ключ-значение, часто используется в Kubernetes.

---

9. Безопасность

1. **HashiCorp Vault**
   Безопасное управление секретами и сертификатами.

2. **Open Policy Agent (OPA)**
   Инструмент для реализации политики безопасности.

3. **Falco**
   Мониторинг безопасности в Kubernetes.

4. **Trivy**
   Инструмент для сканирования уязвимостей в контейнерах.

---

10. Инструменты тестирования

1. **Postman**
   Инструмент для тестирования API.

2. **k6**
   Инструмент для тестирования производительности приложений.

3. **Selenium**
   Инструмент для автоматизированного тестирования веб-приложений.

---

11. Планирование и автоматизация

1. **JIRA**
   Трекер задач, используемый в Agile/Scrum-процессах.

2. **Confluence**
   Вики для совместной работы и документации.

3. **Makefile**
   Утилита для автоматизации сборки и деплоя.

---

12. Инструменты управления пакетами

1. **Helm**
   Пакетный менеджер для Kubernetes, используемый для управления чартами.

2. **Yum / Apt**
   Системные пакетные менеджеры для управления зависимостями.

3. **Homebrew**
   Пакетный менеджер для macOS и Linux.

---

13. Оркестрация и управление

1. **Kubernetes**
   Основной инструмент для оркестрации контейнеров.

2. **Docker Swarm**
   Встроенная в Docker система для оркестрации.

3. **Nomad**
   Альтернатива Kubernetes от HashiCorp для управления контейнерами и виртуальными машинами.

---

14. Прочие инструменты

1. **Bash**
   Скриптовый язык для автоматизации задач.

2. **Python**
   Широко используется для написания автоматизации и работы с API.

3. **Go**
   Часто используется для разработки DevOps-инструментов (Prometheus, Docker).

4. **Cron**
   Планировщик задач для автоматизации периодических процессов.

DevOps-инженер должен не только владеть этими инструментами, но и понимать, как интегрировать их между собой для создания стабильных, масштабируемых и автоматизированных процессов. Если нужно более подробно остановиться на каком-либо инструменте или области, дайте знать!

1. Администрирование Linux (50 вопросов с ответами)

1. **Что такое ядро и как его обновить?**
   Ядро — основная часть ОС, управляющая ресурсами. Обновление: `apt update && apt upgrade` (Ubuntu) или `yum update kernel` (RHEL).

2. **Как проверить версию установленного ядра?**
   Используйте `uname -r`.

3. **Чем отличается soft link от hard link?**
   Soft link — указатель на файл, hard link — копия ссылки на данные.

4. **Как создать и удалить символическую ссылку?**
   Создать: `ln -s target link_name`, удалить: `rm link_name`.

5. **Как настроить автоматический запуск сервиса?**
   `systemctl enable <service>`.

6. **Как посмотреть все запущенные службы?**
   `systemctl list-units --type=service`.

7. **Как изменить права доступа к файлу?**
   `chmod <mode> <file>`.

8. **Что означают атрибуты файла (rwx)?**
   Чтение (r), запись (w), выполнение (x).

9. **Как найти файлы по имени?**
   `find /path -name <filename>`.

10. **Как найти большие файлы на диске?**
    `find / -size +1G`.

11. **Как использовать grep для поиска?**
    `grep "pattern" <file>`.

12. **Как получить root-права?**
    `sudo -i` или `su`.

13. **Как ограничить доступ для пользователя?**
    Используйте `chage` для управления сроком действия пароля.

14. **Как узнать IP-адрес системы?**
    `ip addr show` или `ifconfig`.

15. **Как настроить статический IP?**
    Измените `/etc/network/interfaces` (Ubuntu) или `/etc/sysconfig/network-scripts/ifcfg-<interface>` (RHEL).

16. **Как управлять таблицами маршрутизации?**
    `ip route add` или `route add`.

17. **Что такое iptables?**
    Это утилита для управления правилами брандмауэра. Пример: `iptables -A INPUT -p tcp --dport 22 -j ACCEPT`.

18. **Как открыть порт?**
    Используйте `firewalld` или `iptables`.

19. **Как запустить процесс в фоновом режиме?**
    Добавьте `&` после команды или используйте `nohup`.

20. **Как проверить нагрузку на систему?**
    `top` или `htop`.

21. **Как посмотреть использование оперативной памяти?**
    `free -h`.

22. **Как определить процесс, использующий порт?**
    `lsof -i :<port>` или `netstat -tuln`.

23. **Что такое swap?**
    Виртуальная память на диске. Настройка: `swapon` и `swapoff`.

24. **Как добавить новый диск?**
    Используйте `fdisk` для раздела, `mkfs` для форматирования.

25. **Как создать LVM том?**
    Команды: `pvcreate`, `vgcreate`, `lvcreate`.

26. **Как смонтировать файловую систему?**
    `mount /dev/sdX /mnt`.

27. **Как проверить RAID?**
    Используйте `cat /proc/mdstat`.

28. **Как проверить работу DNS?**
    `nslookup` или `dig`.

29. **Как выполнить копирование с помощью scp?**
    `scp <source> <user>@<host>:<destination>`.

30. **Как установить обновления системы?**
    `apt update && apt upgrade` (Ubuntu) или `yum update` (RHEL).

31. **Что такое chroot?**
    Изоляция процесса. Команда: `chroot /new_root`.

32. **Как ограничить использование ресурсов процесса?**
    `ulimit`.

33. **Как включить SELinux?**
    Измените `/etc/selinux/config`.

34. **Что такое systemd?**
    Менеджер служб. Пример: `systemctl start <service>`.

35. **Как отобразить дерево процессов?**
    `pstree`.

36. **Как создать alias команды?**
    Добавьте в `~/.bashrc`: `alias ll='ls -la'`.

37. **Как проверить подключённые устройства USB?**
    `lsusb`.

38. **Как изменить hostname?**
    `hostnamectl set-hostname <new_hostname>`.

39. **Как отследить процесс по PID?**
    `strace -p <PID>`.

40. **Как архивировать файлы?**
    `tar -czf archive.tar.gz <file>`.

41. **Что такое tmpfs?**
    Файловая система в RAM. Пример: `mount -t tmpfs`.

42. **Как зашифровать диск?**
    Используйте `cryptsetup`.

43. **Как настроить NTP?**
    Установите `ntpd` или `chrony`.

44. **Как посмотреть историю команд?**
    `history`.

45. **Как работает sudo?**
    Выполняет команды от имени root. Настройки: `/etc/sudoers`.

46. **Как проверить лог ядра?**
    `dmesg`.

47. **Что такое inode?**
    Индексный дескриптор файла.

48. **Как найти файл по его содержимому?**
    `grep -r "pattern" /path`.

49. **Как отобразить запущенные службы?**
    `systemctl`.

50. **Как установить cron?**
    Добавьте задачу: `crontab -e`.

---

2. Администрирование PostgreSQL (50 вопросов с ответами)

PostgreSQL: Репликация, Шардирование, Балансировка

---

1. Репликация в PostgreSQL

Репликация — это процесс копирования данных с одной базы данных на другие для обеспечения отказоустойчивости, масштабируемости чтения и резервного копирования.

Типы репликации:

1. **Физическая репликация (Physical Replication):**
   - Передача бинарных изменений (WAL-файлов) с основной базы (primary) на вторичную (replica).
   - Реплика доступна только для чтения.
   - Используется для отказоустойчивости и масштабирования чтения.
2. **Логическая репликация (Logical Replication):**
   - Передача данных на уровне таблиц.
   - Позволяет реплицировать только определённые таблицы.
   - Используется для интеграции данных, миграции или создания шардированных конфигураций.

Настройка физической репликации:

1. **Настройка Primary-сервера:**
   - Измените `postgresql.conf`:

     ```
     wal_level = replica
     max_wal_senders = 10
     wal_keep_size = 64
     ```

   - Убедитесь, что в `pg_hba.conf` разрешён доступ для реплики:

     ```
     host replication replicator 192.168.1.0/24 md5
     ```

2. **Создание реплики:**

   - Выполните бэкап основной базы с помощью `pg_basebackup`:

     ```
     pg_basebackup -h <primary_host> -U replicator -D
     /var/lib/postgresql/data -Fp -Xs -P
     ```

   - Настройте `recovery.conf` или добавьте в `postgresql.conf`:

     ```
     primary_conninfo = 'host=<primary_host> port=5432 user=replicator
     password=<password>'
     standby_mode = on
     ```

   - Запустите PostgreSQL.

Настройка логической репликации:

1. **Настройка Primary-сервера:**

   - Включите логическую репликацию в `postgresql.conf`:

     ```
     wal_level = logical
     max_replication_slots = 4
     max_wal_senders = 4
     ```

   - Разрешите доступ в `pg_hba.conf`.

2. **Создание публикации:**

   ```
   CREATE PUBLICATION my_publication FOR TABLE my_table;
   ```

3. **Подключение подписчика (replica):**

   ```
   CREATE SUBSCRIPTION my_subscription
   CONNECTION 'host=<primary_host> dbname=<db_name> user=<user> password=
   <password>'
   PUBLICATION my_publication;
   ```

---

2. Шардирование в PostgreSQL

Шардирование — это распределение данных по нескольким базам данных (шардам) для повышения производительности.

Подходы к шардированию:

1. **Частичное шардирование (Partitioned Tables):**
   - Данные делятся на разделы по ключу (например, по диапазону дат).
   - Настройка:

```
CREATE TABLE sales (
    id SERIAL PRIMARY KEY,
    sale_date DATE NOT NULL,
    amount NUMERIC
) PARTITION BY RANGE (sale_date);

CREATE TABLE sales_2023 PARTITION OF sales
    FOR VALUES FROM ('2023-01-01') TO ('2023-12-31');
```

2. **Динамическое шардирование (pg_shard/pg_partman):**
   - Данные распределяются автоматически на основе ключа (например, user_id).
   - Инструменты, такие как **Citus**, расширяют возможности шардирования PostgreSQL.

3. **Сегментация на уровне приложений:**
   - Приложение управляет маршрутизацией запросов к различным базам данных.

Инструменты для шардирования:

- **Citus:**
  Расширение PostgreSQL для горизонтального масштабирования и распределения данных по шардам.
- **pg_partman:**
  Для автоматизации создания разделов таблиц.

---

3. Балансировка нагрузки в PostgreSQL

Балансировка нагрузки позволяет распределить запросы на чтение/запись между несколькими серверами для повышения производительности.

Балансировка на уровне чтения (Read Balancing):

1. **Pgpool-II:**
   - Middleware между приложением и базой данных.
   - Распределяет запросы на чтение между репликами.
   - Установка:

```
apt install pgpool2
```

- Конфигурация в `pgpool.conf`:

```
backend_hostname0 = 'primary_host'
backend_hostname1 = 'replica_host1'
backend_hostname2 = 'replica_host2'
load_balance_mode = on
```

2. **HAProxy:**
   - Лёгкий прокси для балансировки нагрузки.
   - Настройка для PostgreSQL:

```
frontend pgsql_front
    bind *:5432
    default_backend pgsql_back

backend pgsql_back
    balance roundrobin
    server primary <primary_host>:5432 check
    server replica1 <replica_host1>:5432 check
```

3. **PostgreSQL AutoFailover:**
   - Автоматическое переключение и маршрутизация запросов.

Балансировка на уровне записи:

- Чаще всего запросы на запись направляются только на primary, так как реплики доступны только для чтения.
- Используются инструменты: **Pgpool-II** или логическое шардирование.

Рекомендации

1. **Репликация:**
   - Используйте физическую репликацию для отказоустойчивости.
   - Логическая репликация полезна для интеграции данных или миграции.
2. **Шардирование:**
   - Подходит для приложений с большими объёмами данных и высокой нагрузкой.
   - Используйте инструменты вроде Citus для упрощения управления.

3. **Балансировка:**
    - Настройте балансировку чтения через Pgpool-II или HAProxy.
    - Убедитесь, что данные на репликах синхронизированы с primary.

Эти технологии позволяют масштабировать PostgreSQL, обеспечивая отказоустойчивость и производительность.

1. **Как подключиться к базе?**
   `psql -U <username> -d <dbname>`.
2. **Как создать пользователя?**
   `CREATE USER <username> WITH PASSWORD '<password>';`.
3. **Как создать базу?**
   `CREATE DATABASE <dbname>;`.
4. **Как удалить базу?**
   `DROP DATABASE <dbname>;`.
5. **Как выполнить бэкап базы?**
   `pg_dump <dbname> > backup.sql`.
6. **Как восстановить базу?**
   `psql <dbname> < backup.sql`.
7. **Где настроить доступ к базе?**
   В `pg_hba.conf`.
8. **Что такое транзакция?**
   Группировка операций с возможностью отката.
9. **Как работает `VACUUM`?**
   Убирает старые версии строк.
10. **Как узнать текущие подключения?**
    `SELECT * FROM pg_stat_activity;`.
11. **Как изменить пароль пользователя?**
    `ALTER USER <username> WITH PASSWORD '<password>';`.
12. **Что такое WAL?**
    Write Ahead Log — журнал изменений.
13. **Как настроить репликацию?**
    Измените параметры в `postgresql.conf`.
14. **Как создать индекс?**
    `CREATE INDEX <index_name> ON <table>(<column>);`.
15. **Как использовать JSON?**
    Через тип данных JSONB. Пример: `SELECT * FROM <table> WHERE data->>'key' =`

`'value';`.

16. **Как проверить производительность запроса?**
    Используйте `EXPLAIN ANALYZE`.

17. **Как включить логирование запросов?**
    Измените `log_statement = 'all'` в `postgresql.conf`.

18. **Как добавить колонку в таблицу?**
    `ALTER TABLE <table> ADD COLUMN <column_name> <data_type>;`.

19. **Как удалить колонку из таблицы?**
    `ALTER TABLE <table> DROP COLUMN <column_name>;`.

20. **Как изменить тип данных в колонке?**
    `ALTER TABLE <table> ALTER COLUMN <column_name> TYPE <new_data_type>;`.

21. **Как создать уникальный индекс?**
    `CREATE UNIQUE INDEX <index_name> ON <table>(<column>);`.

22. **Как работает `SERIAL`?**
    Тип данных `SERIAL` автоматически создаёт последовательность для автонумерации.

23. **Как найти размер базы?**
    `SELECT pg_size_pretty(pg_database_size('<dbname>'));`.

24. **Как работает `pg_stat_activity`?**
    Эта таблица содержит информацию о текущих активных сессиях базы.

25. **Как удалить пользователя?**
    `DROP USER <username>;`.

26. **Как отключить доступ для пользователя?**
    Удалите или измените правила в `pg_hba.conf`.

27. **Как работают триггеры?**
    Это функции, вызываемые при вставке, обновлении или удалении строк.

28. **Как создать триггер?**

```
CREATE TRIGGER <trigger_name>
AFTER INSERT ON <table>
FOR EACH ROW
EXECUTE FUNCTION <function_name>();
```

29. **Как посмотреть текущие транзакции?**
    `SELECT * FROM pg_stat_activity WHERE state = 'active';`.

30. **Что такое таблица-секция (partitioned table)?**
    Это таблица, данные которой разбиваются на части. Создаётся через `PARTITION BY`.

31. **Как удалить таблицу?**
    `DROP TABLE <table>;`.

32. **Как создать materialized view?**
    `CREATE MATERIALIZED VIEW <view_name> AS <query>;`.

33. **Как обновить materialized view?**
    `REFRESH MATERIALIZED VIEW <view_name>;`.

34. **Что такое оконные функции?**
    Это функции, работающие с наборами строк, например: `ROW_NUMBER()`.

35. **Как узнать длительность выполнения запроса?**
    Включите `log_duration` в `postgresql.conf`.

36. **Как включить hot standby?**
    Настройте параметры `hot_standby = on` в `postgresql.conf`.

37. **Как использовать роль в PostgreSQL?**
    Создайте роль: `CREATE ROLE <role_name>`. Назначьте права: `GRANT <privilege> ON <object> TO <role>;`.

38. **Как получить список всех баз данных?**
    `\l` в psql.

39. **Как посмотреть таблицы в базе данных?**
    `\dt` в psql.

40. **Как узнать права доступа к таблице?**
    `\z <table_name>` в psql.

41. **Как работать с транзакциями?**
    Команды: `BEGIN`, `COMMIT`, `ROLLBACK`.

42. **Как создать последовательность (sequence)?**
    `CREATE SEQUENCE <sequence_name>;`.

43. **Как использовать последовательность?**
    `SELECT nextval('<sequence_name>');`.

44. **Как узнать, какие индексы используются?**
    `SELECT * FROM pg_indexes WHERE tablename = '<table_name>';`.

45. **Что такое CTE (Common Table Expression)?**
    Это временные результирующие таблицы в запросах. Пример:

```
WITH cte_name AS (
    SELECT * FROM <table>
)
SELECT * FROM cte_name;
```

46. **Как создать временную таблицу?**
    `CREATE TEMP TABLE <table_name> (<columns>);`.

47. **Какой формат хранения данных используется?**
    Данные хранятся в блоках страниц (8КБ по умолчанию).

48. **Как удалить все строки из таблицы?**
    `TRUNCATE TABLE <table_name>;`.

49. **Как настроить autovacuum?**
    Включите `autovacuum = on` в `postgresql.conf`.

50. **Как подключить клиентскую библиотеку к PostgreSQL?**
    Убедитесь, что `libpq` установлен, и используйте драйверы, такие как `psycopg2`.

---

3. Мониторинг и логирование (Prometheus, Grafana, ELK)

1. **Как настроить Prometheus?**
   Установите Prometheus и настройте `prometheus.yml`, добавив endpoints для метрик.

2. **Что такое метки (labels) в Prometheus?**
   Это атрибуты временных рядов. Пример: `job="node_exporter"`.

3. **Как работает retention в Prometheus?**
   Данные хранятся согласно настройкам: `storage.tsdb.retention.time`.

4. **Как добавить экспортёр?**
   Добавьте в `prometheus.yml`:

   ```
   scrape_configs:
     - job_name: 'node_exporter'
       static_configs:
         - targets: ['localhost:9100']
   ```

5. **Как создать алерт в Prometheus?**
   Добавьте правило в файл alerts:

   ```
   - alert: HighCPUUsage
     expr: node_cpu_seconds_total > 80
     for: 5m
   ```

6. **Как подключить Grafana к Prometheus?**
   В Grafana добавьте Data Source: выберите Prometheus, укажите его URL.

7. **Как создать дашборд в Grafana?**
   Перейдите в "Create -> Dashboard", настройте панели и запросы.

8. **Как работают индексы в Elasticsearch?**
   Индекс — это набор документов. Структура определяется mapping.

9. **Как отправлять логи в Elasticsearch?**
   Используйте Filebeat или Logstash.

10. **Как настроить фильтры в Logstash?**
    Пример:

    ```
    filter {
      grok {
        match => { "message" => "%{COMMONAPACHELOG}" }
      }
    }
    ```

11. **Какой язык запросов используется в Elasticsearch?**
    Elasticsearch Query DSL.

12. **Как работает Kibana?**
    Это инструмент визуализации данных из Elasticsearch.

13. **Как мониторить базы данных через Prometheus?**
    Используйте экспортер, например, `postgres_exporter`.

14. **Как ограничить доступ к Grafana?**
    Настройте Auth в файле `grafana.ini`.

15. **Как работает PushGateway?**
    Это промежуточный компонент для отправки метрик.

16. **Как масштабировать Elasticsearch?**
    Добавьте новые узлы и настройте шардирование.

17. **Как настроить Alertmanager?**
    Настройте файл `alertmanager.yml` для отправки уведомлений.

18. **Что такое Discover в Kibana?**
    Раздел для поиска и анализа данных.

19. **Как управлять логами Elasticsearch?**
    Измените `log4j2.properties`.

20. **Как работают агрегации в Elasticsearch?**
    Агрегации группируют данные для анализа.

21. **Как посмотреть текущие метрики в Prometheus?**
    Перейдите на веб-интерфейс Prometheus и выполните запрос в PromQL.

22. **Как работают `Gauge` и `Counter` в Prometheus?**

- `Gauge`: для показателей, которые могут увеличиваться и уменьшаться (например, температура).
- `Counter`: для монотонно растущих значений (например, количество запросов).

23. **Как настроить фильтрацию логов в Logstash?**
    Добавьте в секцию `filter` правила, например:

```
filter {
  if [status] == "404" {
    drop { }
  }
}
```

24. **Как подключить несколько источников данных в Grafana?**
    Перейдите в "Data Sources" и добавьте каждую из систем (Prometheus, Elasticsearch и т.д.).

25. **Что такое sharding в Elasticsearch?**
    Это деление индекса на части (шарды) для увеличения производительности и масштабируемости.

26. **Как работают ретеншн-политики в Elasticsearch?**
    Используйте `ILM` (Index Lifecycle Management) для автоматического удаления старых данных.

27. **Как настроить интервал опроса в Prometheus?**
    Измените параметр `scrape_interval` в `prometheus.yml`.

28. **Что такое `Lucene` в Elasticsearch?**
    Это поисковая библиотека, на которой основан Elasticsearch.

29. **Как работать с Dashboards в Kibana?**
    Создайте визуализации, объедините их в дашборд и настройте фильтры.

30. **Как настроить alerter в Grafana?**
    Укажите условия триггера, каналы уведомлений и интервал проверки.

31. **Как отследить ошибки в Prometheus?**
    Проверьте статус endpoints на странице `/targets`.

32. **Как создать кастомные метрики?**
    Напишите свой экспортёр или используйте библиотеки, такие как `prom-client` (Node.js).

33. **Как работают метрики `histogram` в Prometheus?**
    Они измеряют распределение значений по заранее определённым интервалам (buckets).

34. **Как отслеживать производительность запросов в Elasticsearch?**
    Включите профилирование запросов через `profile` API.

35. **Как организовать резервное копирование Elasticsearch?**
    Используйте Snapshots API.

36. **Что такое Logstash Pipeline?**
    Это последовательность ввода (input), фильтров (filter) и вывода (output) данных.

37. **Как работают индексы-алиасы в Elasticsearch?**
    Алиасы позволяют обращаться к индексу через псевдоним.

38. **Как настроить масштабируемость Prometheus?**
    Используйте федерацию или Cortex/Thanos.

39. **Как включить SSL/TLS в Grafana?**
    Настройте `cert_file` и `cert_key` в `grafana.ini`.

40. **Как работать с топологией кластеров Elasticsearch?**
    Используйте роли нод: master, data, ingest.

41. **Что такое Discover в Kibana?**
    Раздел для анализа данных, где вы можете использовать фильтры и запросы.

42. **Как настроить отправку уведомлений из Alertmanager?**
    Укажите каналы (email, Slack, Telegram) в файле `alertmanager.yml`.

43. **Как работает `beat` в ELK?**
    Beats — это агенты сбора данных, такие как Filebeat, Metricbeat.

44. **Как построить график временных рядов в Grafana?**
    Выберите панель "Time series", настройте запрос и метрики.

45. **Что такое Document ID в Elasticsearch?**
    Уникальный идентификатор документа в индексе.

46. **Как защитить Prometheus?**
    Используйте аутентификацию через reverse proxy (например, nginx).

47. **Как объединить данные из разных источников в Grafana?**
    Используйте панель `Mixed`, где каждый запрос может обращаться к разным источникам данных.

48. **Что такое фильтры в Kibana?**
    Это правила для ограничения отображения данных (например, по полям).

49. **Как удалить старые метрики в Prometheus?**
    Установите срок хранения через `--storage.tsdb.retention.time`.

50. **Как оптимизировать загрузку данных в Elasticsearch?**
    Используйте bulk API для пакетной записи данных.

---

4. Docker (50 вопросов с ответами)

Основные принципы работы **Docker** заключаются в контейнеризации, изоляции и стандартизации приложений. Вот ключевые аспекты:

---

1. Контейнеризация

Docker позволяет запускать приложения в изолированных контейнерах, что гарантирует воспроизводимость и совместимость. Контейнеры содержат всё необходимое для работы приложения: код, библиотеки, зависимости и системные настройки.

---

2. Архитектура Docker

- **Docker Daemon**
  Фоновый процесс, управляющий контейнерами, образами, сетями и томами.
- **Docker CLI**
  Интерфейс командной строки для взаимодействия с Docker.
- **Docker Registry**
  Хранилище образов (например, Docker Hub).

---

3. Docker Image (Образ)

- **Образы** — это шаблоны для создания контейнеров.
- **Многоуровневая структура:** каждый слой добавляет изменения к предыдущему, что делает образы лёгкими и экономичными.
- **Dockerfile** используется для создания образов, описывая шаги сборки.

---

4. Контейнер

Контейнер — это работающий экземпляр образа. Каждый контейнер:

- Изолирован от других процессов.
- Может взаимодействовать с хостовой системой через определённые ресурсы (сеть, диски).
- Лёгкий, так как использует ядро хоста.

---

5. Механизм слоёв (Union File System)

Docker использует слоистую файловую систему, что позволяет:

- Повторно использовать слои между разными образами.
- Сохранять изменения только в верхнем слое (Copy-on-Write).

---

6. Сетевое взаимодействие

Docker предоставляет несколько сетевых драйверов:

- **Bridge**: изоляция контейнеров внутри хоста.
- **Host**: использование сетевого стека хоста.
- **Overlay**: связь между контейнерами на разных хостах.
- **None**: отключение сети для контейнера.

---

7. Том (Volume)

Docker использует тома для хранения данных вне контейнера, чтобы:

- Данные сохранялись даже при удалении контейнера.
- Упростить совместное использование данных между контейнерами.

---

8. Изоляция (Namespaces)

Docker изолирует ресурсы контейнера с помощью Linux Namespaces:

- **PID:** процессы контейнера.
- **NET:** сеть контейнера.
- **IPC:** механизмы межпроцессного взаимодействия.
- **UTS:** настройка имени хоста.
- **MNT:** файловая система.

---

9. Ограничение ресурсов (Cgroups)

Docker позволяет ограничивать:

- Использование CPU, памяти, ввода-вывода и других ресурсов.
- Пример: `docker run --memory="512m" --cpus="1.0"`.

---

10. Docker Registry

Docker Registry — хранилище образов:

- **Docker Hub**: публичный реестр.
- **Приватные Registry**: для компаний.

---

11. Docker Compose

Docker Compose используется для управления многоконтейнерными приложениями.

- Конфигурация в файле `docker-compose.yml`.
- Позволяет описывать зависимости, сети, тома и другие параметры.

---

12. Docker Swarm

Инструмент для оркестрации контейнеров:

- Управляет кластером Docker-хостов.
- Позволяет распределять нагрузку, управлять масштабированием и обновлением контейнеров.

---

13. Безопасность

Docker изолирует контейнеры, но:

- Важно минимизировать привилегии.
- Использовать минимальные образы (например, Alpine Linux).
- Контролировать доступ через Docker Daemon и TLS.

---

14. Обновления (Rolling Updates)

Docker позволяет обновлять контейнеры без простоев, создавая новые версии и заменяя старые.

---

15. Основные команды

- **Создание контейнера:**
  `docker run <image>`.
- **Управление контейнерами:**
  `docker start/stop/restart <container>`.
- **Создание образа из Dockerfile:**
  `docker build -t <image_name> .`.
- **Монтирование томов:**
  `docker run -v /host/path:/container/path <image>`.

Эти принципы делают Docker гибким, лёгким и мощным инструментом для управления приложениями в любых средах.

1. **Что такое Docker?**
   Docker — платформа для упаковки приложений в изолированные контейнеры.

2. **Как запустить контейнер?**
   `docker run <image>`.

3. **Как проверить, какие контейнеры запущены?**
   `docker ps`.

4. **Как запустить контейнер в интерактивном режиме?**
   `docker run -it <image>`.

5. **Чем отличается `CMD` от `ENTRYPOINT`?**
   - `CMD` задаёт команды по умолчанию.
   - `ENTRYPOINT` делает контейнер исполняемым.

6. **Как создать Docker-образ из Dockerfile?**
   `docker build -t <image_name> .`.

7. **Как остановить контейнер?**
   `docker stop <container_id>`.

8. **Как удалить контейнер?**
   `docker rm <container_id>`.

9. **Как удалить образ?**
   `docker rmi <image_id>`.

10. **Как запустить контейнер с монтированием тома?**
    `docker run -v /host/path:/container/path <image>`.

11. **Как запустить контейнер с ограничением ресурсов?**

    ```
    docker run --memory="500m" --cpus="1.0" <image>
    ```

12. **Как посмотреть логи контейнера?**
    `docker logs <container_id>`.

13. **Какой формат используется для Dockerfile?**
    Это текстовый файл с инструкциями (например, `FROM`, `RUN`, `CMD`).

14. **Как работает сеть контейнеров?**
    Docker использует bridge-сеть по умолчанию. Для связи между контейнерами используйте `--network`.

15. **Как проверить слои образа?**
    `docker history <image>`.

16. **Как оптимизировать Dockerfile?**
    - Используйте минимальные базовые образы, такие как `alpine`.
    - Объединяйте команды `RUN`.

17. **Как подключить несколько сетей к контейнеру?**
    Создайте сети: `docker network create`, а затем подключите контейнеры через `--network`.

18. **Как обновить образ контейнера?**
    Пересоберите Dockerfile и перезапустите контейнер.

19. **Как работает `docker-compose`?**
    Это инструмент для управления многоконтейнерными приложениями.

20. **Как настроить зависимости контейнеров в `docker-compose`?**
    Укажите `depends_on` в `docker-compose.yml`.

21. **Как выполнить команду внутри контейнера?**
    `docker exec -it <container_id> <command>`.

22. **Как настроить переменные окружения?**
    Используйте флаг `-e` или файл `.env`.

23. **Как создать приватный Docker Registry?**
    Запустите `registry` контейнер: `docker run -d -p 5000:5000 --name registry registry`.

24. **Как удалить dangling images?**
    `docker image prune`.

25. **Как подключить секреты в Docker Swarm?**
    Создайте секрет: `docker secret create`, и подключите его к сервису.

26. **Что такое overlay network?**
    Сеть для взаимодействия между нодами Swarm.

27. **Как перезапустить контейнер?**
    `docker restart <container_id>`.

28. **Как работают multi-stage builds?**
    Это процесс разделения шагов сборки для уменьшения размера образа.

29. **Как сделать контейнер read-only?**
    `docker run --read-only <image>`.

30. **Как управлять томами?**
    `docker volume create`, `docker volume inspect`.

31. **Как подключить GPU в Docker?**
    Используйте `--gpus` при запуске контейнера.

32. **Как настроить автоматический перезапуск контейнера?**
    `docker run --restart=always <image>`.

33. **Как включить healthcheck в Docker?**
    Добавьте в Dockerfile:

    ```
    HEALTHCHECK CMD curl --fail http://localhost || exit 1
    ```

34. **Что такое `docker-compose.override.yml`?**
    Файл для перезаписи конфигурации по умолчанию.

35. **Как работать с BuildKit?**
    Включите BuildKit: `DOCKER_BUILDKIT=1 docker build`.

36. **Как масштабировать сервисы в Swarm?**
    Используйте `docker service scale`.

37. **Как отобразить список образов?**
    `docker images`.

38. **Как мониторить контейнеры?**
    Используйте `docker stats`.

39. **Что такое Docker Context?**
    Контекст позволяет переключаться между средами Docker.

40. **Как настроить CI/CD с Docker?**
    Используйте инструменты, такие как Jenkins, GitLab CI или GitHub Actions.

41. **Как добавить пользовательский network driver?**
    Укажите драйвер при создании сети: `docker network create --driver=<driver_name>`.

42. **Как выполнять бэкапы данных из контейнера?**
    Используйте `docker cp` для копирования данных.

43. **Как настроить proxy в контейнере?**
    Передайте параметры `http_proxy` и `https_proxy`.

44. **Как запустить контейнер без доступа к сети?**
    Используйте флаг `--network none`.

45. **Что такое rootless Docker?**
    Запуск Docker без root-привилегий.

46. **Как защитить Docker Daemon?**
    Настройте TLS для подключения к Docker API.

47. **Как проверить версию Docker API?**
    `docker version`.

48. **Как удалить все остановленные контейнеры?**
    `docker container prune`.

49. **Как выполнить миграцию данных контейнеров?**
    Используйте `docker export` и `docker import`.

50. **Как проверить состояние Docker Daemon?**
    `systemctl status docker`.

50 вопросов и ответов по Kubernetes

---

1. Что такое Kubernetes?

Kubernetes — это платформа оркестрации контейнеров, которая автоматизирует развертывание, масштабирование и управление контейнеризированными приложениями.

---

2. Какие основные компоненты Kubernetes?

- **Master Node (управляющий узел):**
  - API Server
  - Scheduler
  - Controller Manager
  - etcd
- **Worker Node (рабочий узел):**
  - Kubelet

- Kube-proxy
- Контейнерный Runtime

---

3. Что такое Pod?

Pod — минимальная сущность в Kubernetes, представляющая собой один или несколько контейнеров, работающих совместно.

---

4. Чем отличается Deployment от StatefulSet?

- **Deployment:** используется для статических приложений. Подходит для управления репликами без сохранения состояния.
- **StatefulSet:** применяется для приложений с состоянием (например, баз данных). Гарантирует порядок запуска Pod'ов.

---

5. Как настроить автообновление контейнеров в Kubernetes?

Используйте `RollingUpdate` в Deployment:

```
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxUnavailable: 1
    maxSurge: 1
```

---

6. Что такое kubelet?

Kubelet — агент, работающий на Worker Node. Он отвечает за запуск и мониторинг Pod'ов.

---

7. Как проверить статус всех Pod'ов?

Команда:

```
kubectl get pods --all-namespaces
```

---

8. Что такое ConfigMap?

ConfigMap используется для хранения конфигурационных данных в виде ключей и значений.

---

9. Что такое Secret?

Secret — объект для хранения конфиденциальных данных, таких как пароли, ключи API.

---

10. Чем отличается ConfigMap от Secret?

- **ConfigMap:** хранит неконфиденциальные данные в текстовом виде.
- **Secret:** хранит данные в кодировке Base64 для дополнительной защиты.

---

11. Что такое Service?

Service — объект Kubernetes, который определяет способ доступа к группе Pod'ов.

---

12. Какие типы Service существуют?

- ClusterIP (по умолчанию, доступ только внутри кластера)
- NodePort (доступ через порт узла)
- LoadBalancer (интеграция с внешними балансировщиками)
- ExternalName (перенаправление на внешние DNS-имена)

---

13. Что такое Ingress?

Ingress — это способ управления внешним доступом к сервисам внутри кластера.

---

14. Чем отличается HorizontalPodAutoscaler от VerticalPodAutoscaler?

- **HorizontalPodAutoscaler (HPA):** увеличивает/уменьшает количество Pod'ов.
- **VerticalPodAutoscaler (VPA):** изменяет ресурсы (CPU, память) Pod'ов.

---

15. Как задать ресурсы для Pod'а?

Используйте `resources` в спецификации Pod'а:

```
resources:
  requests:
    memory: "256Mi"
    cpu: "500m"
  limits:
    memory: "512Mi"
    cpu: "1"
```

---

16. Как создать Namespace?

```
kubectl create namespace <namespace-name>
```

---

17. Как переключиться на другой Namespace?

```
kubectl config set-context --current --namespace=<namespace-name>
```

---

18. Как удалить Pod?

```
kubectl delete pod <pod-name>
```

---

19. Что такое DaemonSet?

DaemonSet гарантирует, что на каждом узле запущен определённый Pod (например, мониторинг или логирование).

---

20. Что такое Job?

Job — объект Kubernetes, который выполняет задачу один раз и завершает работу.

---

21. Что такое CronJob?

CronJob — это Job, запускаемый по расписанию.

---

22. Как отобразить логи Pod'а?

```
kubectl logs <pod-name>
```

---

23. Как подключиться к Pod'у?

```
kubectl exec -it <pod-name> -- /bin/bash
```

---

24. Что такое ReplicaSet?

ReplicaSet гарантирует, что заданное количество Pod'ов запущено.

---

25. Чем отличается Deployment от ReplicaSet?

Deployment управляет ReplicaSet, добавляя возможности обновления, откатов и стратегии развертывания.

---

26. Как проверить статус кластера?

```
kubectl cluster-info
```

---

27. Что такое etcd?

etcd — распределённое хранилище данных, используемое для хранения состояния кластера Kubernetes.

---

28. Как создать Deployment?

```
kubectl create deployment <name> --image=<image>
```

---

29. Что такое PV и PVC?

- **PersistentVolume (PV):** предоставляет физическое хранилище.
- **PersistentVolumeClaim (PVC):** запрос на использование PV.

---

30. Как удалить Node из кластера?

```
kubectl drain <node-name> --ignore-daemonsets
kubectl delete node <node-name>
```

---

31. Что такое Kube-proxy?

Kube-proxy управляет сетевыми правилами для маршрутизации трафика к Pod'ам.

---

32. Как отобразить состояние узлов?

```
kubectl get nodes
```

---

33. Что такое Taint и Toleration?

- **Taint:** добавляет ограничения на запуск Pod'ов на определённых узлах.
- **Toleration:** позволяет Pod'ам игнорировать Taint.

---

34. Что такое Affinity и Anti-Affinity?

- **Affinity:** предпочтения для размещения Pod'ов на определённых узлах.
- **Anti-Affinity:** предпочтения для размещения Pod'ов на разных узлах.

---

35. Как обновить образ в Deployment?

```
kubectl set image deployment/<name> <container-name>=<new-image>
```

---

36. Что такое Helm?

Helm — это менеджер пакетов для Kubernetes, который упрощает развертывание приложений.

---

37. Что такое kubectl?

kubectl — это CLI-инструмент для взаимодействия с Kubernetes API.

---

38. Как проверить доступные ресурсы?

```
kubectl api-resources
```

---

39. Что такое kubeconfig?

kubeconfig — файл конфигурации для управления доступом к Kubernetes кластеру.

---

40. Как настроить мониторинг в Kubernetes?

Используйте инструменты, такие как Prometheus и Grafana.

---

41. Как включить отладку?

```
kubectl describe <resource> <name>
```

---

42. Что такое Sidecar контейнер?

Sidecar — дополнительный контейнер в Pod'е, который расширяет функциональность основного контейнера (например, логирование, прокси).

---

43. Как использовать LimitRange?

LimitRange задаёт ограничения на ресурсы в Namespace.

---

44. Что такое NetworkPolicy?

NetworkPolicy ограничивает сетевое взаимодействие между Pod'ами и внешними ресурсами.

---

45. Как пересоздать Pod при сбое?

Kubernetes автоматически перезапустит Pod, если он контролируется Deployment или ReplicaSet.

---

46. Что такое RBAC?

RBAC (Role-Based Access Control) управляет доступом к ресурсам кластера.

---

47. Как проверить доступы?

```
kubectl auth can-i <action> <resource>
```

---

48. Как создать тайм-аут для Pod?

Используйте `livenessProbe` и `readinessProbe` для контроля состояния контейнера:

```
livenessProbe:
  httpGet:
    path: /healthz
    port: 8080
  initialDelaySeconds: 5
  periodSeconds: 10
```

---

49. Как восстановить кластер после сбоя?

Используйте резервные копии etcd и PV для восстановления состояния.

---

50. Как настроить балансировку нагрузки?

Используйте Service с типом LoadBalancer или установите Ingress Controller.