Автоматизация разработки и эксплуатации программного обеспечения

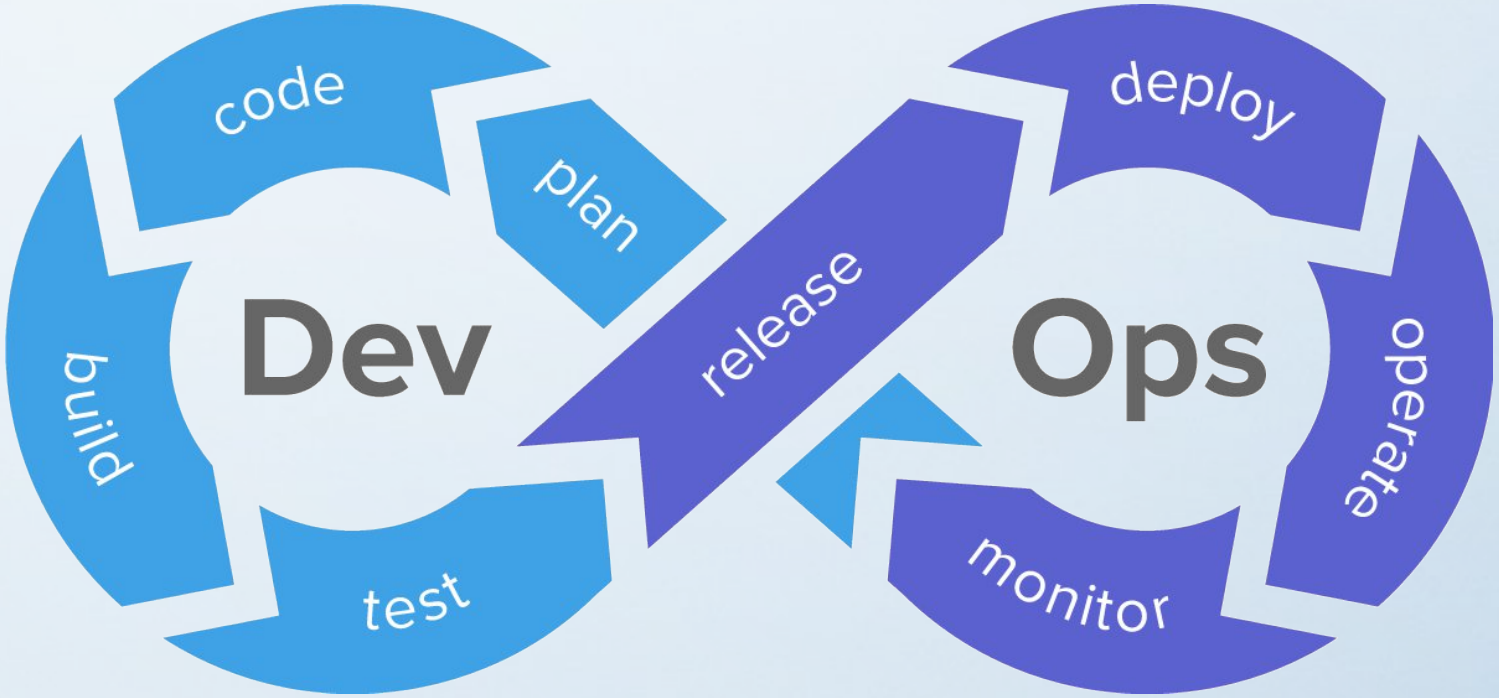
ИУ-5, бакалавриат

Виды занятий

- Лекции:
  - 17 лекций, 34 часа.
- Лабораторные работы
  - 8 лабораторных работ, 34 часа.
- Домашнее задание.
  - Проект по развертыванию программного обеспечения.
- Сайт курса:
  - https://devops.science.iu5.bmstu.ru

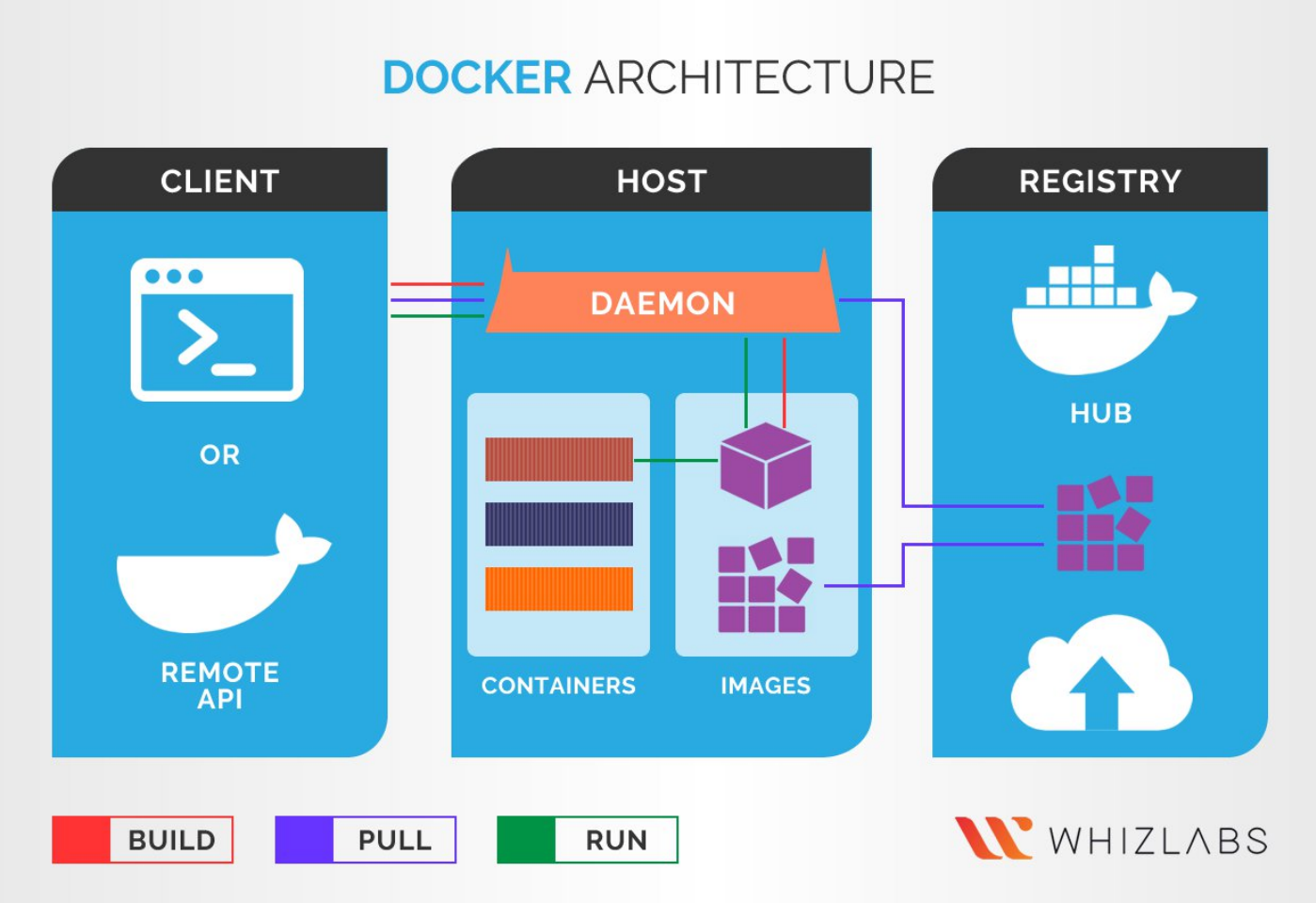
О чем курс?

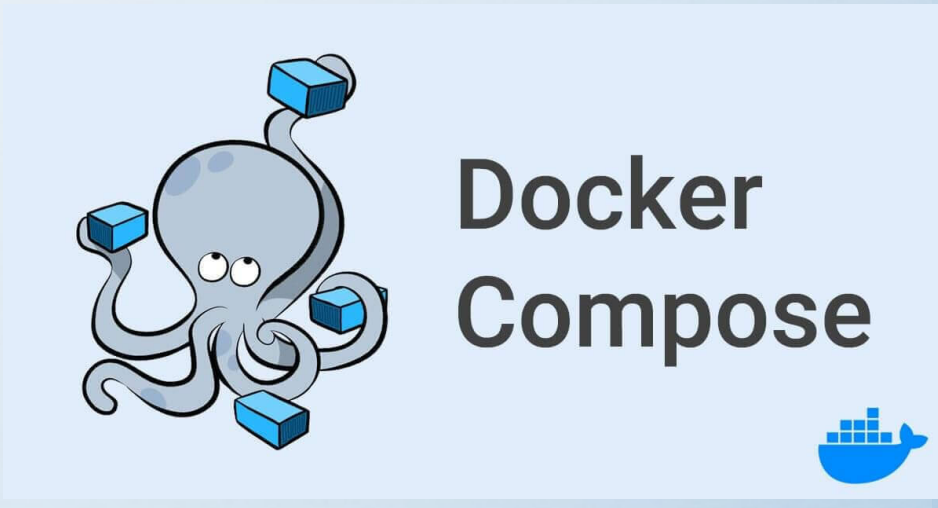
Автоматизация разработки и эксплуатации программного обеспечения



О чем курс?

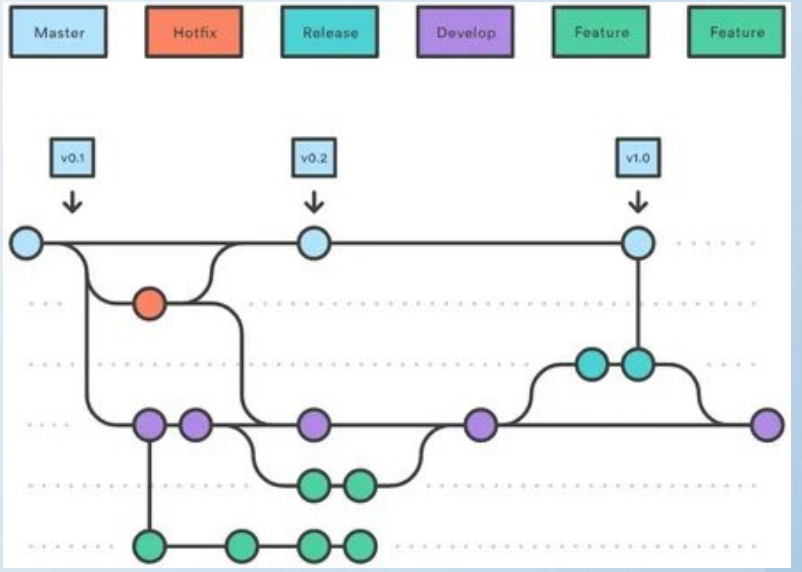
Автоматизация разработки и эксплуатации программного обеспечения

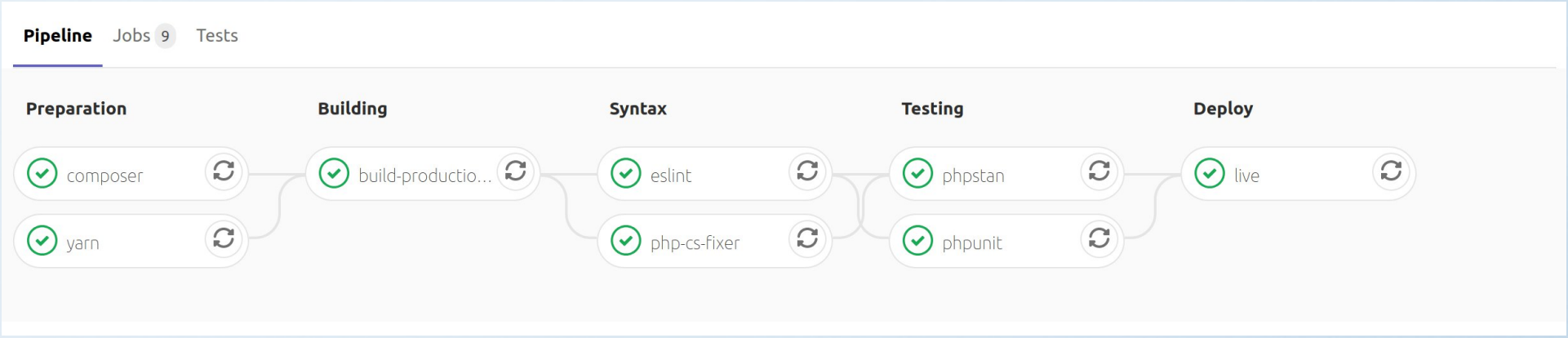




О чем курс?

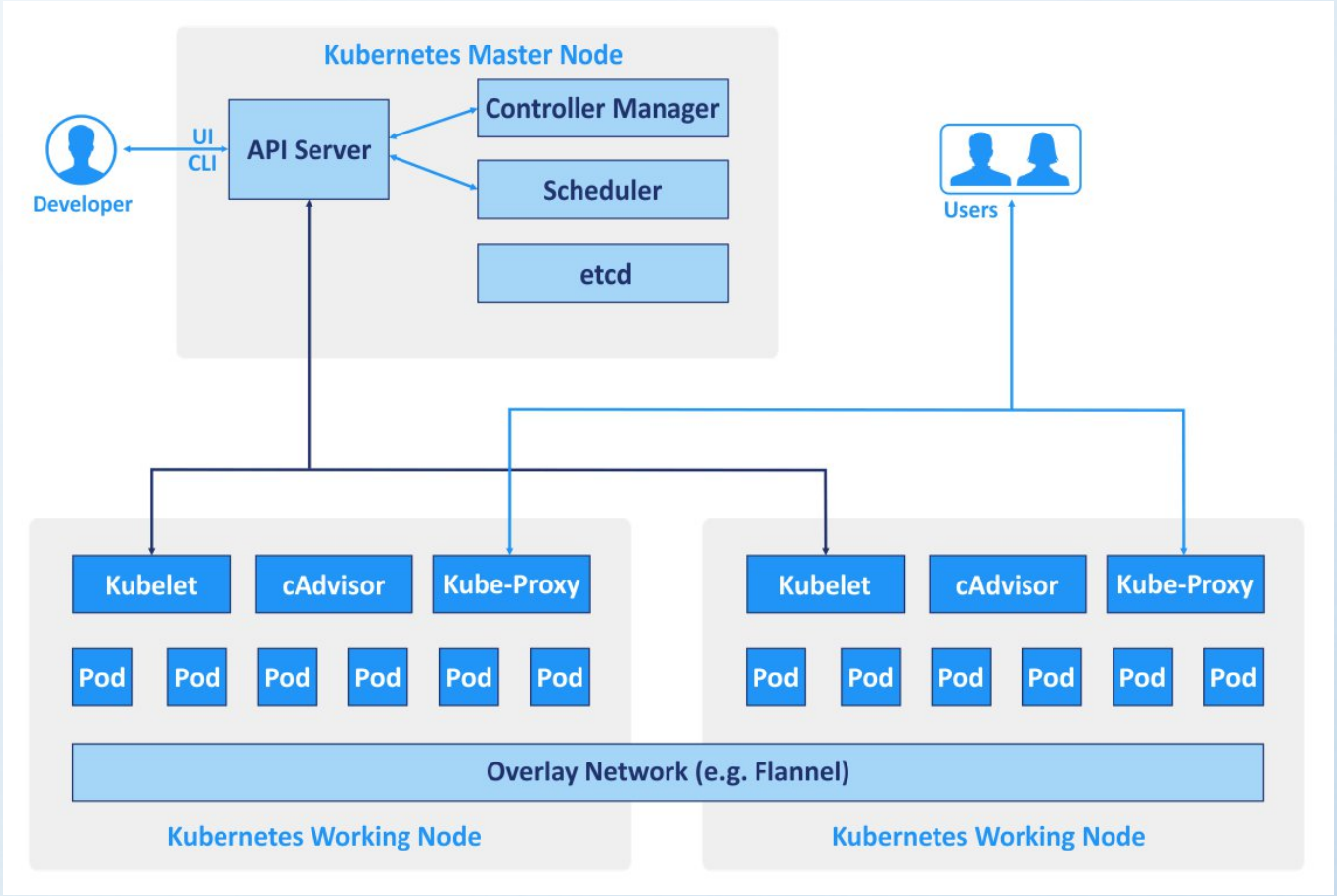
Автоматизация разработки и эксплуатации программного обеспечения





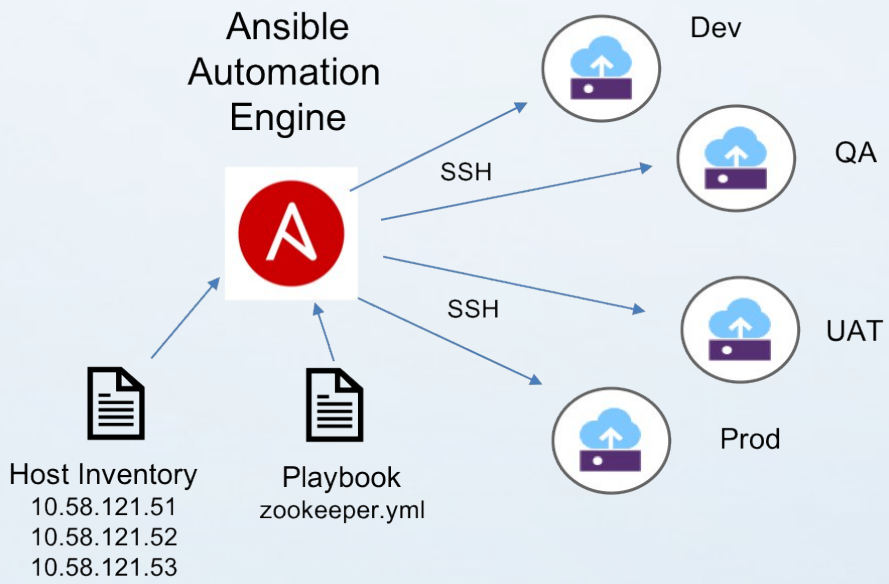
О чем курс?

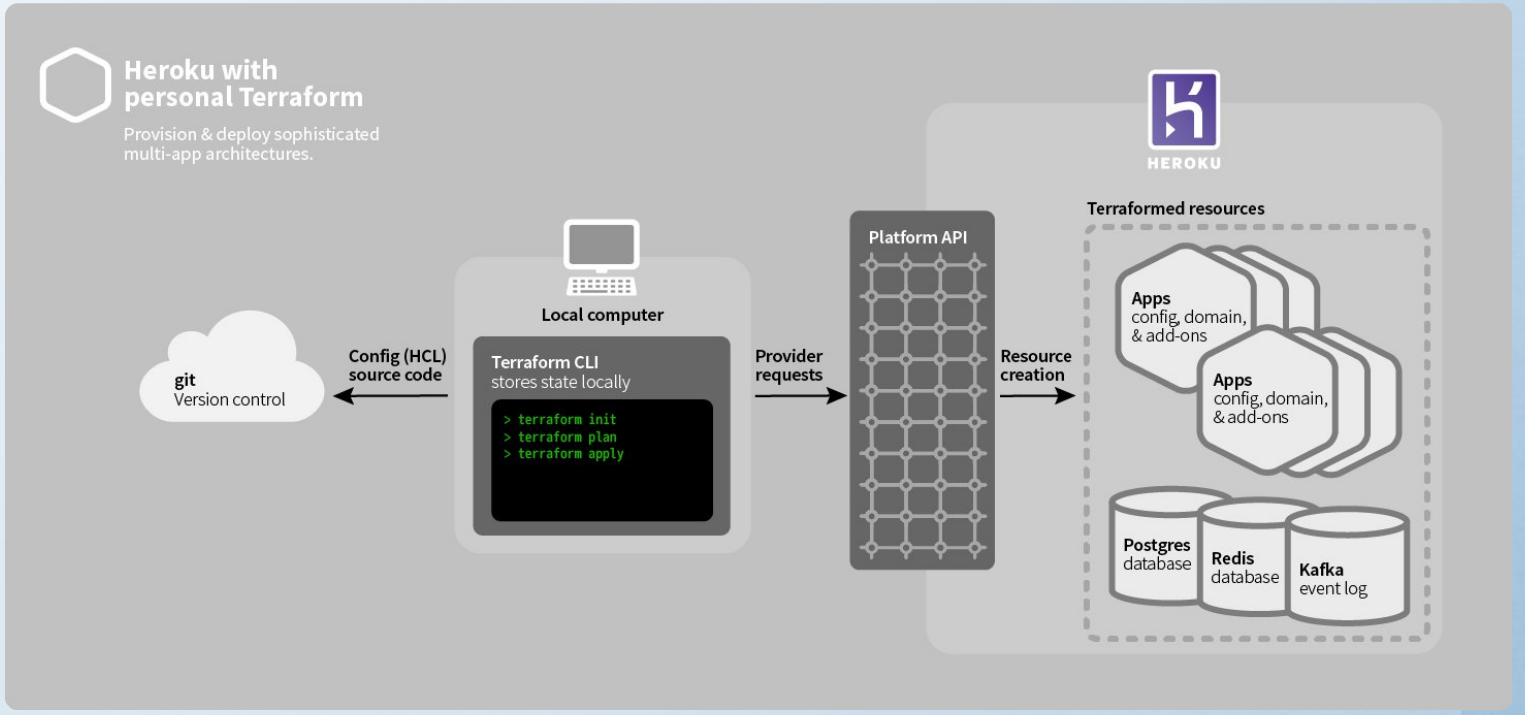
Автоматизация разработки и эксплуатации программного обеспечения



О чем курс?

Автоматизация разработки и эксплуатации программного обеспечения





Лабораторные работы

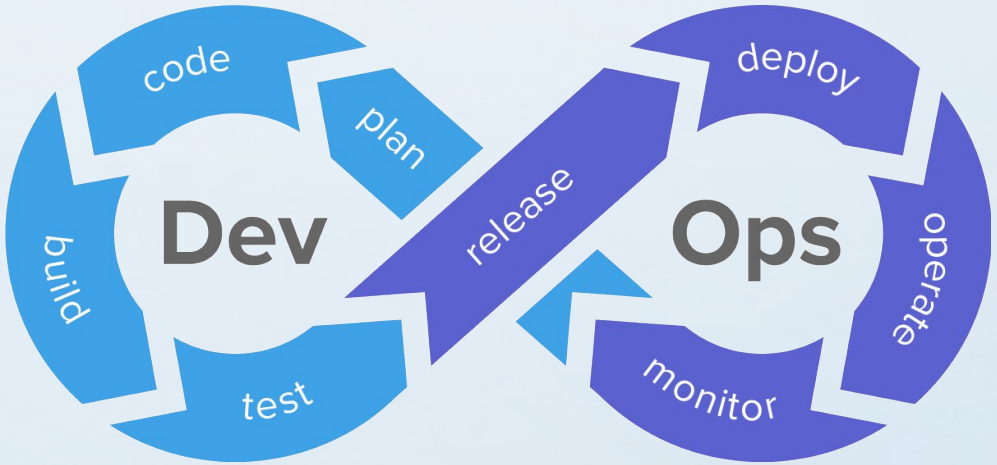
- Для проведения лабораторных работ потребуется компьютер под управлением ОС семейства Linux
- В качестве альтернативы можно взять образ виртуальной машины VirtualBox
- Задания: https://devops.science.iu5.bmstu.ru/labs/

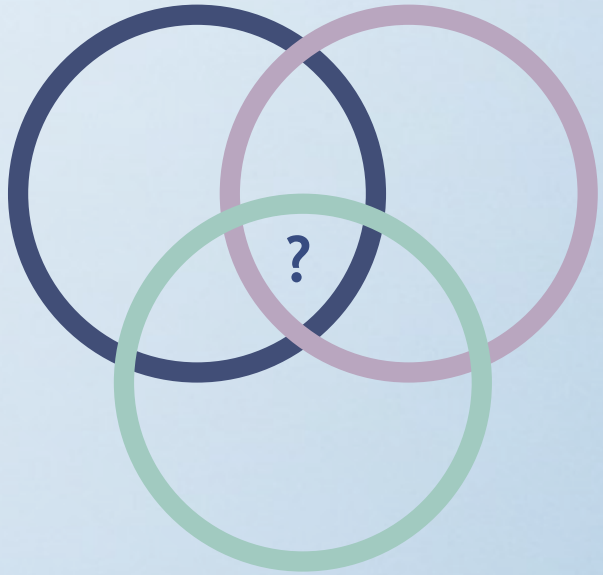
Часть 1 - DevOps

Что такое DevOps?

DevOps:

- **development** – разработка/развитие;
- **operations** – эксплуатация/использование.





Почему DevOps?

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

-- *Melvin E. Conway*

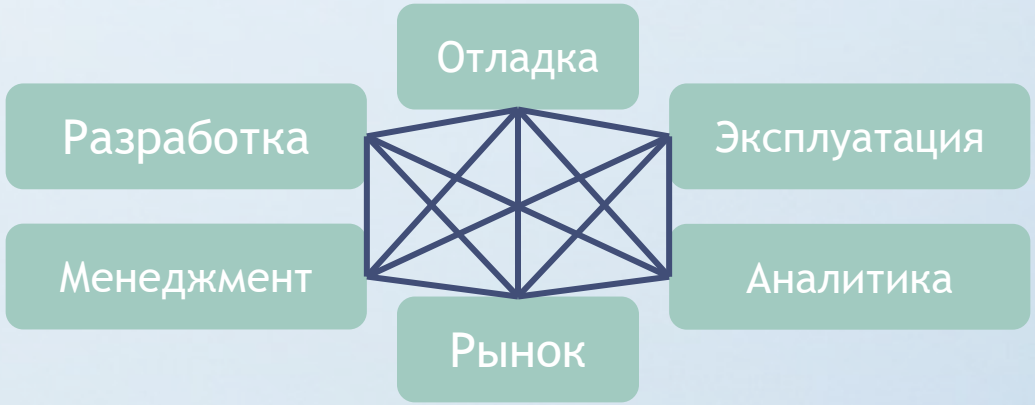
[!] Время выхода на рынок - time-to-market

Почему DevOps?

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.

-- *Melvin E. Conway*

[!] Время выхода на рынок - time-to-market



Зачем DevOps?

- **Dev** – Чаще отправлять изменения на продуктовую среду, не понимают как оно крутится на серверах.

- **Ops** – Реже отправлять изменения на продуктовую среду, меньше отказов, не понимают, как работает продукт.

Цели DevOps

- сокращение времени для выхода на рынок (time-to-market)
- снижение частоты отказов новых релизов
- сокращение времени выполнения исправлений
- уменьшение количества времени на восстановления

DevOps или SRE?

DevOps-инженер – (автоматизация сборки, настройки и развертывания ПО)

SRE - Site Reliability Engineering (обеспечение надежности информационных систем)

Инженерные задачи VS **Операционные задачи**

Что нужно DevOps/SR-инженеру?

- знать жизненный цикл ПО и методологию
- освоить разные архитектуры ПО, в т.ч. микросервисную
- знать основы программирования, выучить несколько ЯП
- уметь отлаживать программы и устранять уязвимости и ошибки
- понимать принципы работы операционных систем
- понимать принципы работы компьютерных сетей
- понимать виртуализацию, облачные и гибридные решения
- разбираться в системах оркестрации
- разбираться в системах управления конфигурацией (СУК)
- уметь налаживать мониторинг продукта
- общаться с другими людьми и командами

Чем занимается DevOps-инженер

Исходя из философии DevOps, внедрять его практики могут как разработчики, так и инженеры эксплуатации. Но судя по вакансиям, разработчиков в России на эту роль ищут редко, и требования к DevOps-инженеру во многом пересекаются с требованиями к системному администратору. Смотрите сами.

Знание и опыт администрирования:

1. Linux;
2. систем контейнеризации (Docker, Kubernetes);
3. баз данных;
4. Анализ потребностей организации в вопросах построения оптимальной инфраструктуры

Чем занимается DevOps-инженер

- Понимание принципов работы TCP/IP;
- Опыт администрирования SQL и NoSQL баз данных;
- Опыт настройки систем мониторинга и логирования (Zabbix, ELK, Grafana, Prometheus);
- Опыт конфигурирования инфраструктуры через код (Ansible);
- Умение писать скрипты на Bash, Python или Ruby (иногда упоминается Perl);
- Опыт работы с облачными платформами.

Чем занимается SRE-инженер

Site Reliability Engineering (SRE) — это одна из форм реализации DevOps. SRE-подход возник в Google и стал популярен в среде продуктовых IT-компаний после выхода одноимённой книги в 2016 году.

Цель инженера по SRE — сделать так, чтобы сервис работал с такой надёжностью, которая указана в соглашении о уровне оказания услуг (SLA). Стабильность работы зависит и от инфраструктуры, и от качества кода, поэтому SRE занимается и тем, и другим. Как правило, инженерами по SRE становятся опытные системные администраторы или разработчики. Разработчики даже чаще.

Чем занимается SRE-инженер

- уверенные знания ОС Linux продвинутого уровня работы в консоли;
- опыт разработки на C/C++ или Go;
- понимание работы TCP/IP и HTTP;
- понимание устройства ОС (процессы, память, синхронизация);
- опыт настройки различных инструментов для мониторинга и конфигурирования серверов. Инструменты: Zabbix, Puppet, Ansible или подобные.
- опыт работы с Docker;
- опыт использования облачных сервисов
- опыт настройки сетей. Инструменты: nginx, haproxy, bind, git, iptables, stunnel, openvpn;
- опыт настройки виртуальных машин: kvm, xen.

Чем занимается SRE-инженер

Пример задач:

- Оптимизация имеющейся архитектуры и сервисов;
- Уменьшение нагрузки на сопровождение и обслуживание сервисов за счет автоматизации;
- Изучение имеющихся сервисов и поддержание актуальных знаний по ним;
- Активный и проактивный поиск возможных проблем и их устранение;
- Обеспечивать заданный уровень SLA & SLO;
- Участвовать в инцидент-менеджменте;
- Писать postmortem-ы и разрабатывать мероприятия для повышения стабильности сервисов;
- Создавать, актуализировать DRP (Disaster Recovery Plan), BCP(Disaster Recovery Plan) и проводить регулярные учения по отказам с последующим анализом результатов;
- Участвовать в проработке архитектуры сервисов и изменений их конфигураций;
- Оптимизация имеющихся систем observability.

Основные термины

- Система управления исходным кодом (SCCS)
- Инструменты сборки
- Системы управления конфигурацией (СУК)
- Непрерывная интеграция
- Непрерывная доставка
- Тестирование
- Мониторинг
- Оркестрация
- Облака
- IT-инфраструктура

Основные процессы

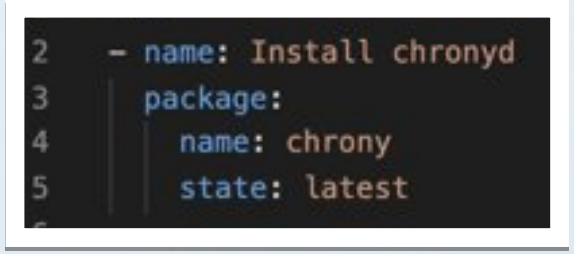
- Поэтапное обновление
- Выявление проблем
- Безопасно откатывать
- Резервирование
- Масштабируемость
- "Graceful degradation” - стратегия управления дизайном веб-страниц в разных браузерах
- Оценка SLA, SLO, SLI:
  - SLA (Service Level Aggement) — соглашение с клиентом об измеримых показателях уровня сервиса, а также о мерах ответственности
  - SLO (Service Level Objective) — цель, обозначающая уровень сервиса, который мы должны обеспечивать
  - SLI (Service Level Indicator) — индикатор, отображающий фактический уровень сервиса
- Бюджет ошибок
- Мониторинг


КАЧЕСТВЕННО
ДОРОГО
ДОЛГО
НЕ БЫВАЕТ
БЫСТРО
КРИВО
ДЕШЕВО

Инфраструктура как код (IaC)

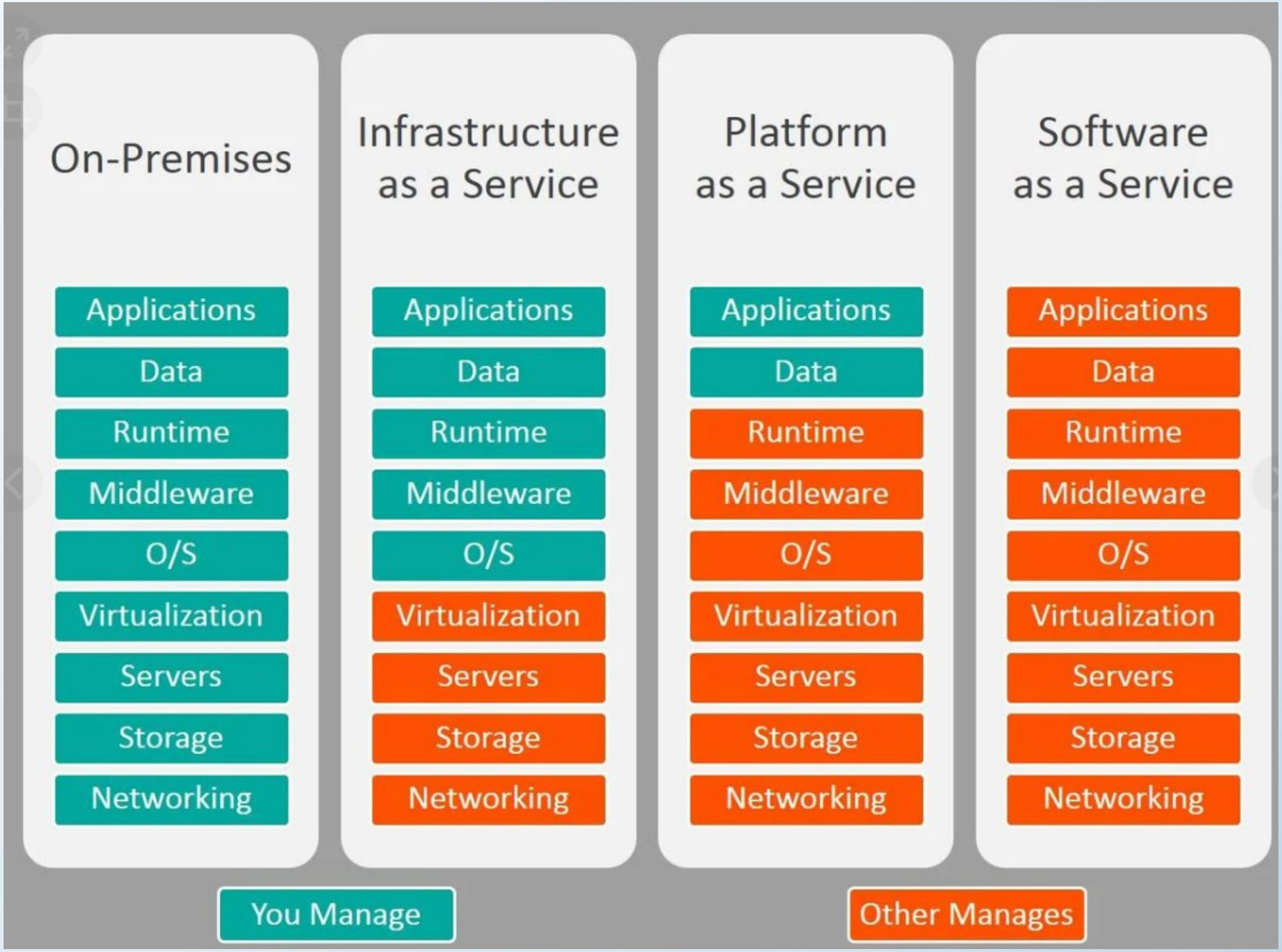
Подход для управления и описания IT-инфраструктуры через конфигурационные файлы, а не через ручное редактирование конфигураций на серверах или интерактивное взаимодействие. Этот подход может включать в себя как декларативный способ описания инфраструктуры, так и императивный.

Примеры: **Ansible**, Chef, Saltstack, Puppet, **Terraform**, …



Где размещать проект?

- На собственных серверах
- На виртуальных машинах в облаке
- В PaaS / Serverless


On-Premises
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
Infrastructure as a Service
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
Platform as a Service
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
Software as a Service
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
You Manage
Other Manages

Часть 2 - Виртуализация и облачные решения

Виртуализация

Виртуализация — предоставление набора вычислительных ресурсов или их логического объединения, абстрагированное от аппаратной реализации, и обеспечивающее при этом логическую изоляцию друг от друга вычислительных процессов, выполняемых на одном физическом ресурсе.

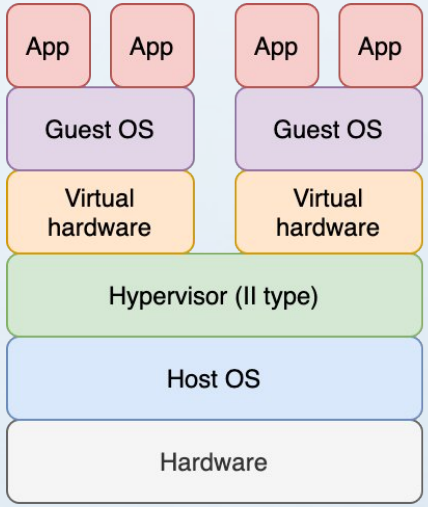
Гипервизор - программа или аппаратная схема, обеспечивающая или позволяющая одновременное, параллельное выполнение нескольких операционных систем на одном и том же хост-компьютере.

Классификация виртуализации

- Эмуляция
- Программная виртуализация
  - Трансляция команд
  - Паравиртуализация (метод виртуализации, который представляет программный интерфейс для виртуальных машин, похожий, но не идентичный базовому программно-аппаратному интерфейсу)
- **Аппаратная виртуализация**
- **Контейнеризация**

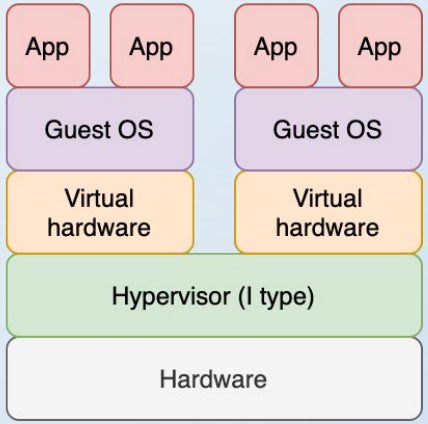
Гипервизоры

Задачи гипервизора:
- эмуляция виртуальных аппаратных ресурсов
- полная изоляция среды
- распределение физических аппаратных ресурсов



Гипервизор 2 типа

VirtualBox, VMware Workstation, QEMU, Parallels, …



Гипервизор 1 типа

VMware ESXi, Citrix XenServer

Облачные решения

- VK CS (MCS)
- Yandex.cloud
- BASIS
- Cloud.ru

- Amazon Web Services
- Google Cloud Platform
- Microsoft Azure
- IBM cloud computing

VK Cloud

- Главная
- Облачные вычисления
- Виртуальные сети
- Data Platform
- Мониторинг
- CDN
- DNS
- Объектное хранилище
- Контейнеры
- Базы данных
- Аналитические БД
- Магазин приложений
- Большие данные
- Графические адаптеры
- ML Platform
- Cloud Desktop
- AI API
- Готовое рабочее место

Настройка меню

Панель управления

Скрыть квоты

Инстансы	CPU	RAM	Объем	Диски
0 из 4 шт	0 из 8 шт	0 из 16 ГБ	0 из 200 ГБ	0 из 10 шт

Виртуальные машины

Создание и размещение IT-сервисов в облаке

Создать инстанс

Объектное хранилище

Хранение данных с быстрым доступом

Создать бакет

Контейнеры Kubernetes

Разворачивание и запуск производительных приложений

Создать кластер

Базы данных

Перенос базы данных и управление системой в облаке

Создать базу данных

Работа с сетями | Резервное копирование | Другие функции

Внешние или внутренние сети любой конфигурации	Настройка правил фильтрации трафика	Распределение нагрузки и масштабирование	Объединение вашей приватной сети с облачной сетью	Быстрое создание CDN-ресурса для ускорения загрузки сайта
Создать сеть	Создать firewall	Создать balancer	Создать VPN	Создать ресурс

VK Cloud
Помощь

Нужна помощь?

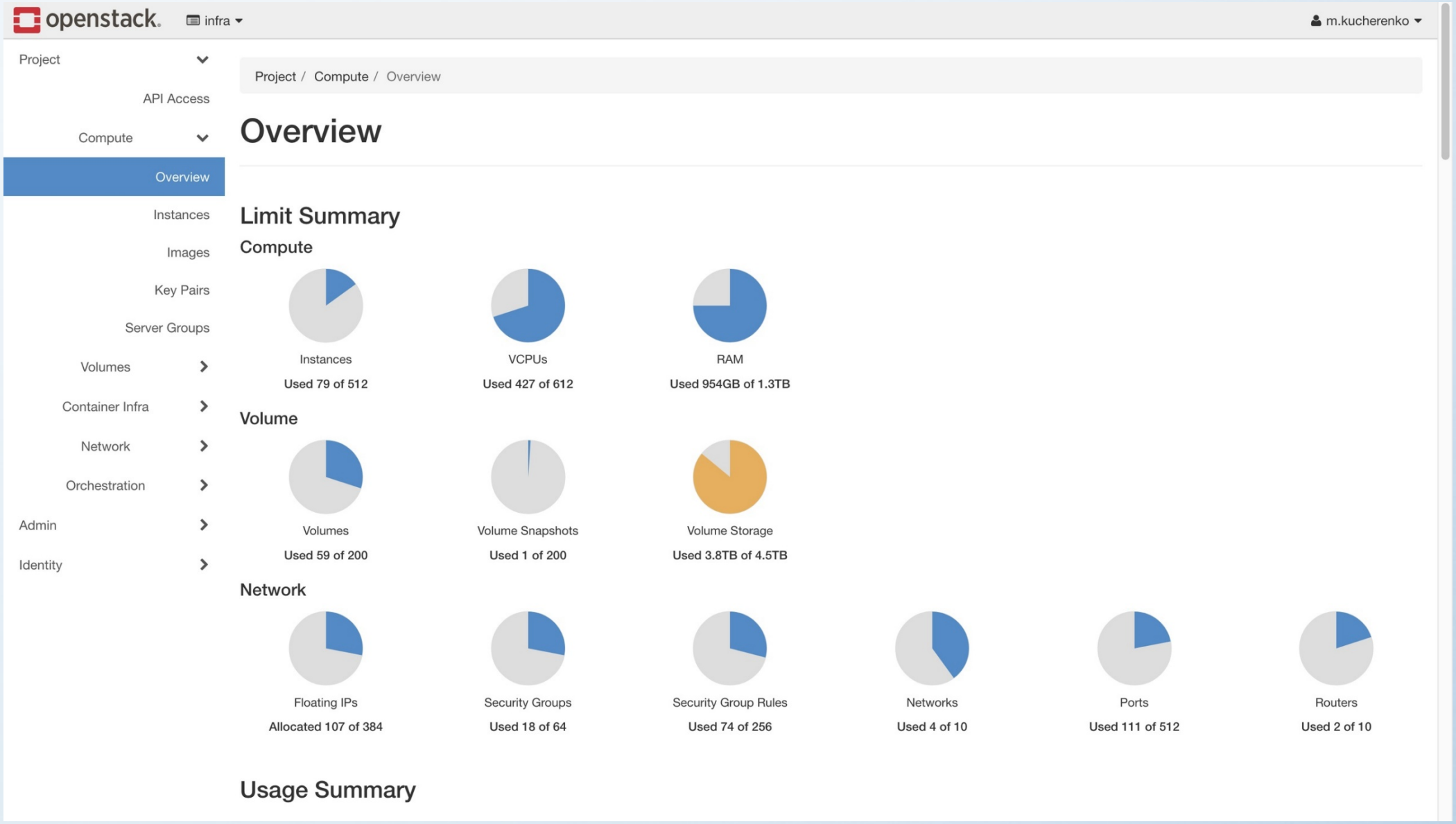
Воспользуйтесь клиентским порталом, чатом Telegram или свяжитесь с нами по телефону +7 (499) 350-9703

Задать вопрос

Документация

Поиск

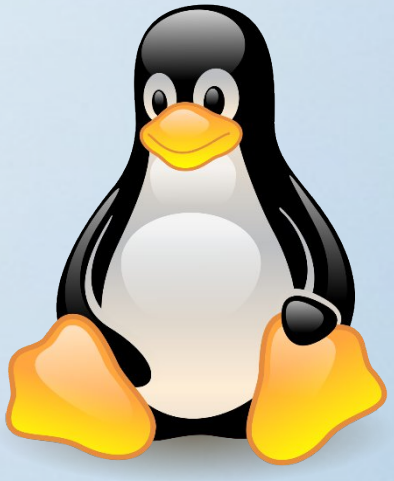
- Донастройка проекта для ЮЛ
- Квоты и лимиты
- Роли и права пользователей личного кабинета
- Биллинг
- Правовая информация
- Поддержка
- Вопросы и ответы
- Быстрый старт
- Активация сервисов
- Управление балансом проекта
- Настройки проекта
- Управление аккаунтом VK Cloud


openstack.
infra
m.kucherenko
Project
API Access
Compute
Overview
Instances
Images
Key Pairs
Server Groups
Volumes
Container Infra
Network
Orchestration
Admin
Identity
Project / Compute / Overview
Overview
Limit Summary
Compute
Instances
Used 79 of 512
VCPUs
Used 427 of 612
RAM
Used 954GB of 1.3TB
Volume
Volumes
Used 59 of 200
Volume Snapshots
Used 1 of 200
Volume Storage
Used 3.8TB of 4.5TB
Network
Floating IPs
Allocated 107 of 384
Security Groups
Used 18 of 64
Security Group Rules
Used 74 of 256
Networks
Used 4 of 10
Ports
Used 111 of 512
Routers
Used 2 of 10
Usage Summary

Часть 3 – GNU/Linux

GNU/Linux

GNU/Linux - семейство операционных систем на основе ядра Linux и программ проекта GNU. Не являются системами семейства Unix, однако работают по схожим принципам, частично соответствуют стандартам POSIX и признаются Unix-подобными.



Командная оболочка

Bash (от англ. Bourne again shell, каламбур «Born again» shell — «возрождённый» shell) — усовершенствованная и модернизированная вариация командной оболочки Bourne shell. Одна из наиболее популярных современных разновидностей командной оболочки UNIX.

$# - общее количество параметров переданных скрипту
$* - все аргументы переданыне скрипту(выводятся в строку)
$@ - аргументы выводятся в столбик
$! - PID последнего запущенного в фоне процесса
$$ - PID самого скрипта

Советуем повторить:

- top, atop, htop
- ps, kill
- lsof, df, du, iostat
- tcpdump, netstat
- fg, bg, jobs

```
1   #!/bin/bash
2
3   var1=$1 - первый параметр скрипта
4   var2=$2 - второй параметр скрипта
5
6   if [[ "$var" -eq "substring" ]]
7   then
8       echo "Равно"
9   else
10      echo "Не равно"
11  fi
12
13  for i in {1..10}
14  do
15      echo "Номер $i"
16      echo 'Номер $i' # Просто текст
17  done
```

chroot

- Операция (системный вызов и утилита) изменения корневого каталога в Unix-подобных операционных системах.
- Простейший способ изоляции на уровне ФС

Super user

- **sudo**
- Этот код даст вам права суперпользователя. Введите sudo перед нужной командой (например, sudo apt upgrade), чтобы выполнить её от имени администратора. Система спросит у вас пароль.
- **sudo su**
- Благодаря этой комбинации все введённые вами команды будут исполняться от имени суперпользователя, пока вы не закроете терминал. Применяйте её, если вам нужно выполнить много команд с правами администратора.

Установка ПО

- **sudo apt install имя_пакета** - Установить нужный пакет.
- **sudo apt-add-repository адрес_репозитория** - Добавить сторонний репозиторий.
- **sudo apt update** - Обновить сведения о пакетах.
- **sudo apt upgrade** - Обновить все пакеты до самых свежих (выполнять после apt update).
- **sudo apt remove имя_пакета** - Удалить ненужный пакет.
- **sudo apt purge имя_пакета** - Удалить ненужный пакет со всеми зависимостями, если хотите освободить больше места.
- **sudo apt autoremove** - Удалить все ненужные зависимости, бесхозные пакеты и прочий мусор.

Управление процессами

- **kill** Эта команда служит для принудительного завершения процессов. Нужно ввести kill PID_процесса. PID процесса можно узнать, введя top.
- **killall** Убивает процессы с определённым именем. К примеру, killall firefox.
- **top** Отображает перечень запущенных процессов, сортируя их в зависимости от потребления ресурсов CPU. Своего рода терминальный «системный монитор». Нажмите q, чтобы выйти из просмотра.
- **htop** Представляет более интерактивную и удобную версию команды top, которая отображает список процессов с использованием графиков и позволяет легко управлять ими.

Управление файлами

- **cat** Когда команда используется с одним текстовым файлом (вот так: cat путь_к_файлу), она отображает его содержимое в окне терминала. Если указать два файла и больше (cat путь_к_файлу_1 путь_к_файлу_2), она склеит их. А если ввести cat путь_к_файлу_1 > новый_файл — объединит содержимое упомянутых файлов в новый документ.
- **chmod** Позволяет изменять права доступа к файлу. Может пригодиться, если вы хотите внести изменения в системный файл.
- **chown** Изменяет владельца файла или каталога. Следует выполнять с правами суперпользователя. Например, chown user:group ваш_файл изменит владельца и группу файла на заданные.
- **file** Выводит информацию об указанном файле.
- **nano** Открывает простой текстовый редактор. Можно создать новый текстовый файл или открыть существующий: *nano путь_к_файлу*.

Управление файлами

- **rename** Переименовывает один или несколько файлов. Команду можно использовать и для массового переименования по маске.
- **touch** Изменяет дату последнего открытия или модификации указанного файла.
- **tar** Команда для создания или извлечения архивов tar. Например, tar -cvf архив.tar ваши_файлы создаст архив архив.tar из указанных документов, а tar -xvf архив.tar извлечёт их.
- **zip** Аналогичным образом распаковывает и сжимает архивы ZIP. Например, zip -r9 архив.zip папка создаст архив архив.zip, содержащий все файлы и подкаталоги из папки, с максимальным уровнем сжатия.
- **mkdir** Создаёт новую папку в текущей терминальной или в указанной папке: mkdir путь_к_папке.
- **rmdir** Удаляет упомянутую папку.
- **rm** Удаляет файлы. Может работать как с отдельными элементами, так и с группой, соответствующей определённым признакам.

Управление файлами

- **cp** Создаёт копию нужного файла в папке терминала: cp путь_к_файлу. Также вы можете указать назначение cp путь_к_файлу путь_для_копии.
- **mv** Перемещает файл из одной папки в другую. Вы можете указать имя для перемещаемого файла. Забавно, но в Linux эта команда может использоваться и для переименования документов. Просто укажите ту же папку, где находится файл, и другое название.
- **find** Поиск файлов по определённым критериям, таким как имя, тип, размер, владелец, дата создания и модификации.
- **grep** Поиск текстовых файлов, содержащих определённые строки. Критерии очень гибко настраиваются.
- **locate** Поиск файлов и папок, чьи названия подходят запросу, и отображение их путей в файловой системе.

Управление разделами

- **lsblk** Эта команда демонстрирует, какие диски есть в вашей системе и на какие разделы они поделены. Также она отображает имена ваших разделов и накопителей в формате sda1, sda2 и так далее.

- **mount** Монтирует накопители, устройства или файловые системы Linux, чтобы вы могли с ними работать. Обычно устройства подключаются автоматически, как только вы щёлкнете по ним в файловом менеджере. Но иногда может понадобиться примонтировать что-то вручную. Вы можете подключать что угодно: диски, внешние накопители, разделы и даже ISO-образы. Эту команду нужно выполнять с правами суперпользователя. Чтобы примонтировать имеющийся диск или раздел, введите mount sdX.

- **umount** Демонтирует файловые системы. Комбинация umount sdX отключит файловую систему внешнего носителя, чтобы вы могли извлечь его.

- **dd** Эта команда копирует и преобразовывает файлы и разделы. У неё множество различных применений.
  - Например, dd if=/dev/sda of=/dev/sdb сделает точную копию раздела sda на разделе sdb.
  - dd if=/dev/zero of=/dev/sdX затрёт содержимое указанного носителя нулями, чтобы информацию было невозможно восстановить.
  - dd if=~/Downloads/ubuntu.iso of=/dev/sdX bs=4M сделает загрузочный носитель из скачанного вами образа с дистрибутивом

Управление системой

- **df** Отображает объём вашего диска и количество свободного места на нём.
- **free** Показывает объём доступной и занятой оперативной памяти.
- **uname** Отображает сведения о системе. Если ввести uname, терминал сообщит только информацию о Linux. А вот команда uname -a выдаёт сведения об имени компьютера и версии ядра.
- **uptime** Сообщает, как давно запущена ваша система.
- **whereis** Отображает расположение исполняемого файла нужной программы.
- **whoami** Называет имя пользователя. Полезно, если вы забыли, под каким логином зашли в систему.
- **reboot** Перезагружает компьютер.
- **shutdown** - Выключает систему, чтобы можно было отключить питание компьютера.
- **lsb_release -a** Показывает информацию о дистрибутиве. Полезно, если вы много экспериментируете с разными системами и забыли, в какой именно работаете. Ну или если вам нужно узнать номер версии вашей ОС

Управление пользователями

- **useradd** Регистрирует нового пользователя. Введите useradd имя_пользователя, и профиль будет создан.
- **userdel** Удаляет учётную запись и файлы пользователя.
- **usermod** Изменяет учётную запись пользователя. Может переместить его домашнюю папку или назначить дату, когда учётная запись будет заблокирована.
- **passwd** Изменяет пароли учётных записей. Обычный пользователь может сделать это только со своим профилем. Суперпользователь способен изменить пароль любой учётной записи

Управление сетью

- **ip** Многофункциональная команда для работы с сетью.
  - Например, ip address show выводит сведения о сетевых адресах, ip route управляет маршрутизацией и так далее..
- **ping** Показывает, подключены ли вы к сети, и помогает определить качество связи.
- **ifconfig** Выводит информацию о сетевых интерфейсах на системе — например, об IP-адресе, о MAC-адресе и других сетевых параметрах.
- **ssh** Применяется для удалённого доступа к серверу по протоколу SSH. Например, ssh user@host подключится к удалённому серверу с указанными пользователем и хостом. Имейте в виду, что на другом компьютере должен быть настроен удалённый доступ.
- **scp** Команда для копирования файлов между локальной системой и удалённым сервером по протоколу SSH. Например, scp файл.txt user@remote:/remote/path скопирует файл.txt на удалённый сервер в заданный путь.
- **wget** Позволяет скачивать файлы из интернета по указанному URL в домашнюю папку пользователя.

Ваши вопросы?