Автоматизация разработки и эксплуатации программного обеспечения (осень 2022 года)

ИУ-5, бакалавриат, курс по выбору

Виды занятий

- Лекции:
  - 17 лекций, 34 часа.
  - ПОНЕДЕЛЬНИК, 10.15, 430 (ГЗ)
- Лабораторные работы
  - 8 лабораторных работ, 34 часа.
  - по расписанию
- Домашнее задание.
  - Проект по развертыванию программного обеспечения.
- Репозиторий курса:
  - https://github.com/iu5git/DevOps

Конфигурационное управление ИТ-инфраструктурой

Балашов Антон

План лекции

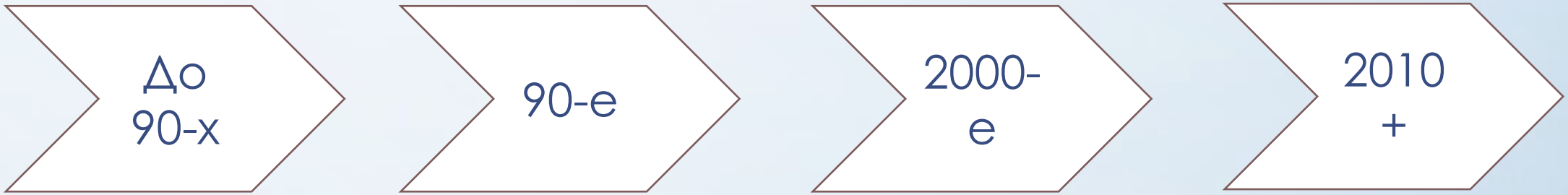
- Развитие ИТ-инфраструктуры
- Стандарты описания ИТ-инфраструктуры
- Подходы к управлению ИТ-инфраструктурой
- Инфраструктура как код
  - Ansible
  - Terraform

Особые обозначения

Наступать на грабли
совершать одинаковые ошибки снова и снова

Строить велосипед
Разрабатывать самому решение, уже реализованное другими, вместо того, чтобы им воспользоваться

Этапы развития



Мейнфреймы

- Клиентские ОС
- Локальные сети
- Интеграторы

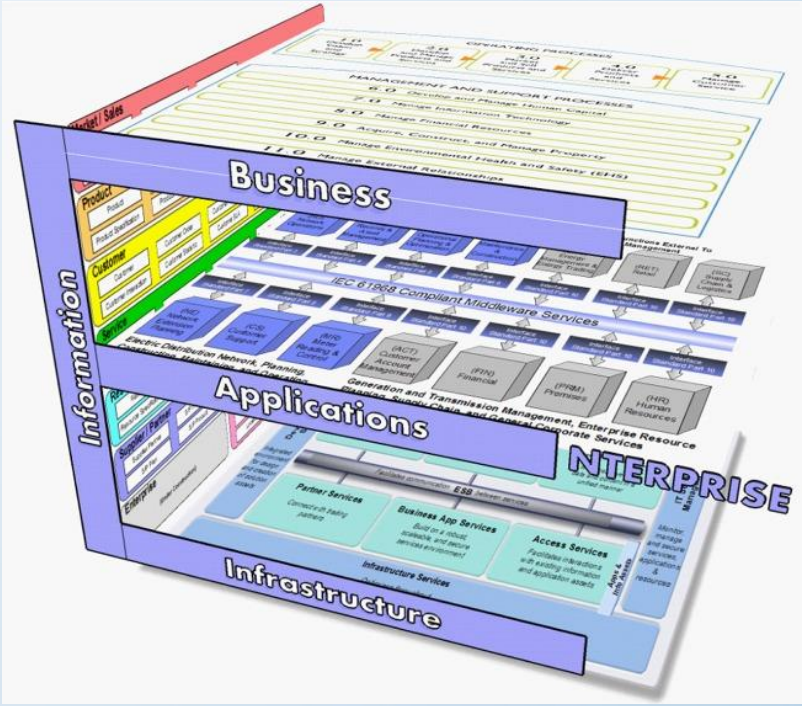
- Развитие интернета
- Дата-центры
- co-location
- Аутсорсинг

- облака
- сервис-провайдеры

Архитектура предприятия

(EA - Enterprise Architecture) определяет общую структуру и функции систем (бизнес и ИТ) в рамках всей организации в целом, включает в себя:

- общая модель (framework),
- стандарты и руководства для архитектуры уровня отдельных проектов,
- единое проектирование систем для обеспечения потребностей организации,
- взаимодействие и интеграция.



Архитектура предприятия

Архитектура предприятия описывает деятельность компании с двух основных позиций:

- **Бизнес-архитектура** описывает предприятие с позиции взаимодействия бизнес-процессов, правил и потоков информации.
- **Архитектура информационных технологий** описывает аппаратные и компьютерные средства, программное обеспечение, защиту и безопасность.

ИТ-архитектура

Включает в себя как логические, так и технические компоненты. Логическая архитектура предоставляет высокоуровневое описание миссии предприятия, его функциональных и информационных требований, системных компонентов и информационных потоков между этими компонентами.

- Enterprise Information Architecture (EIA) – информационная архитектура.
- Enterprise Solution Architecture (ESA) – архитектура прикладных решений.
- Enterprise Technical Architecture (ETA) – техническая архитектура.

Информационная архитектура (EIA - Enterprise Information Architecture)

В ходе разработки информационной архитектуры решаются следующие задачи:

- Идентификация существующих данных, определение их источников и процедур использования.
- Оптимизация данных за счет сокращения дублирования информации. Исключение неоднозначности и противоречивости информации.
- Минимизация перемещения данных за счет их оптимального расположения.
- Интеграция метаданных для обеспечения их целостного представления.
- Сокращение числа используемых технологий, обеспечивающих хранение и доступность информации.

Архитектура прикладных решений (ESA - Enterprise Solution Architecture)

Описание конкретных приложений:

- Компоненты и структура системы – внутренняя структура системы, включающая в себя информацию о программных модулях и базах данных.
- Взаимодействие с другими системами (интерфейсы) – описывает взаимодействие приложения с внешними объектами (программными продуктами, пользователями).

Архитектура прикладных решений (ESA - Enterprise Solution Architecture)

Классификация информационных систем в соответствии с их архитектурными стилями выделяет пять основных групп информационных систем:

- Приложения обслуживающие большое количество транзакций (**Transaction Processing**) - биллинговые, банковские системы.
- Операции в реальном времени (**Real-Time operations**) –системы, обеспечивающие бизнес процессы, требующие непрерывный мониторинг и информационное обеспечение, например, обеспечение транспортных операций в аэропорту.
- Аналитические приложения, бизнес-аналитика, поддержка принятия решений (**Analytical and Business Intelligence**) - управление знаниями, сбор и анализ больших массивов данных.
- Приложения поддержки совместной работы (**Collaborative**) - средства взаимодействия пользователей внутри компаниями.
- Корпоративные и обслуживающие приложения (**Utility**) –стандартные приложения, обеспечивающие функционирование основных бизнес-процессов компании: управление взаимоотношения с клиентами (CRM), управление ресурсами предприятия (ERP).

Архитектура прикладных решений (ESA - Enterprise Solution Architecture)

В соответствии с представленными выше критериями все ИС на предприятии можно разделить на следующие уровни критичности:

Level 1. Mission-Critical. Системы непрерывного действия для решения особо важных (критичных) задач. Сбой систем подобного уровня выводит из строя, парализует работу всего комплекса информационных систем или оказывает существенное влияние на функционирование компании.

Level 2. Business-Critical. Системы, критичные для бизнеса. Системы, обеспечивающие эффективное выполнение бизнес-процессов компании, но при этом не оказывающие прямого воздействия на них. Предприятие может функционировать без информационных систем этого уровня (т.к. подобные операции могут быть выполнены вручную), но, в случае их остановки, будет нести существенные финансовые потери.

Level 3. Business Operational. Системы, обеспечивающие функционирование бизнеса. Информационные системы данного уровня используются бизнесом для увеличения его эффективности, но при этом, их отключение на непродолжительное время не приведет к существенным финансовым потерям. Долгосрочное отключение этих систем будет влиять на эффективность бизнеса.

Level 4. Office Productivity. Системы внутреннего использования. К данному уровню относятся информационные системы, обеспечивающие эффективность выполнения офисных операций. Эти системы не являются важными для функционирования предприятия в целом, но необходимы для увеличения эффективности работы персонала.

Техническая архитектура предприятия (ETA - Enterprise Technical Architecture)

совокупность программно-аппаратных средств, методов и стандартов, обеспечивающих эффективное функционирование приложений:

- Информацию об инфраструктуре предприятия.
- Системное программное обеспечение (СУБД, системы интеграции).
- Стандарты на программно-аппаратные средства.
- Средства обеспечения безопасности (программно-аппаратные).
- Системы управления инфраструктурой.

Техническая архитектура предприятия (ETA - Enterprise Technical Architecture)

Gartner выделяет следующие основные сервисы, входящие в состав любой информационной архитектуры:

- **Сервисы данных**: системы управления базами данных, хранилища данных, системы поддержки принятия решений (Business Intelligence).
- **Прикладные сервисы**: языки программирования, средства разработки приложений, системы коллективной работы.
- **Программное обеспечение промежуточного слоя.**
- **Вычислительная инфраструктура**: операционные системы и аппаратное обеспечение.
- **Сетевые сервисы, локальные сети**: сетевое аппаратное обеспечение.
- **Сервисы безопасности, авторизация**: аутентификация, сетевая безопасность, физическая безопасность центров обработки данных

Техническая архитектура предприятия (ETA - Enterprise Technical Architecture)

Принципы построения ИТ инфраструктуры:

- Техническая инфраструктура масштабируется и расширяется
- Инфраструктура проста в эксплуатации и сопровождении
- Инфраструктура адекватна потребностям приложений и бизнеса
- Инфраструктура строится в строгом соответствии стандартам
- Стандартизация всех программно-аппаратных средств компании

Управление ИТ-подразделением как сервисом

Парадигма управления ИТ-инфраструктурой компании, основанная на:

SLA (соответствие обещаний поставщика услуги ожиданиям клиента) и

ITSM (IT Service Management, управление ИТ-услугами) - это концепция организации работы ИТ-подразделения и его взаимодействия с внешним или внутренним заказчиком, а также внешними контрагентами.

Управление ИТ-подразделением как сервисом

Концепция Управления ИТ-службами — **Information Technology Service Management (ITSM)**:

- формализация процессов функционирования информационных технологий;
- профессионализм и четкая ответственность сотрудников ИТ-отдела за определенный круг задач;
- **технологическая инфраструктура обеспечения качества услуг**:
  - собственно информационные технологии, служба поддержки пользователей;
  - служба управления конфигурациями и изменениями; - система контроля услуг;
  - служба тестирования и внедрения новых услуг и т.д.

ITIL **(IT Infrastructure Library)** — основа концепции управления ИТ-службами

Набор публикаций (библиотека), содержащий лучшие практики в области управления ИТ-услугами. ITIL содержит рекомендации по предоставлению качественных ИТ-услуг, процессов, функций, а также других средств, необходимых для их поддержки.

Структура ITIL основана на жизненном цикле услуги, который состоит из пяти стадий:

- стратегия,
- проектирование,
- преобразование,
- эксплуатация
- постоянное совершенствование

Также существуют дополнительные публикации, входящие в ITIL и содержащие специфичные рекомендации по индустриям, типам компаний, моделям работы и технологическим архитектурам

А в жизни…

Если принимать все положения как есть, без привязки к текущему положению, можно прийти к излишней формализации и значительному нарушению работы. Процесс внедрения принципов ITIL должен быть избирательным и адаптивным.

- Необходим анализ текущих процессов, а есть ли необходимость внедрения вообще, или же достаточно косметических изменений

- Если нет конечной четкой цели использования ITIL, то лучше не пытаться внедрять . Например- вопросы лицензирования ПО, а не просто «взять лучшие практики

Модели ИТ-инфраструктуры

- Локальная модель – все свое
- Гибридная модель:
  - **IaaS** (Infrastructure as a Service) – инфраструктура как услуга
  - **PaaS** (Platform as a Service) – платформа как услуга
  - **SaaS** (Software as a Service) – программное обеспечение как услуга

Модели ИТ-инфраструктуры

Локальная инфраструктура

Компания делает сама:

- Приложения
- Среда исполнения (ОС, фреймворк)
- Сервер и сеть, гипервизор
- Оборудование
- ЦОД

Модели ИТ-инфраструктуры

IaaS (Infrastructure as a Service) – инфраструктура как услуга.

Компания делает сама:

- Приложения
- Среда исполнения (ОС, фреймворк)

Поставщик услуги предоставляет:

- Сервер и сеть, гипервизор
- Оборудование
- ЦОД

«Каршеринг»

Модели ИТ-инфраструктуры

PaaS (Platform as a Service) – платформа как услуга

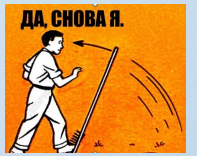
Компания делает сама:

- Приложения

Поставщик услуги предоставляет:

- Среда исполнения (ОС, фреймворк)
- Сервер и сеть, гипервизор
- Оборудование
- ЦОД

«Такси»



Модели ИТ-инфраструктуры

SaaS (Software as a Service) – программное обеспечение как услуга

Поставщик услуги предоставляет:

- Приложения
- Среда исполнения (ОС, фреймворк)
- Сервер и сеть, гипервизор
- Оборудование
- ЦОД

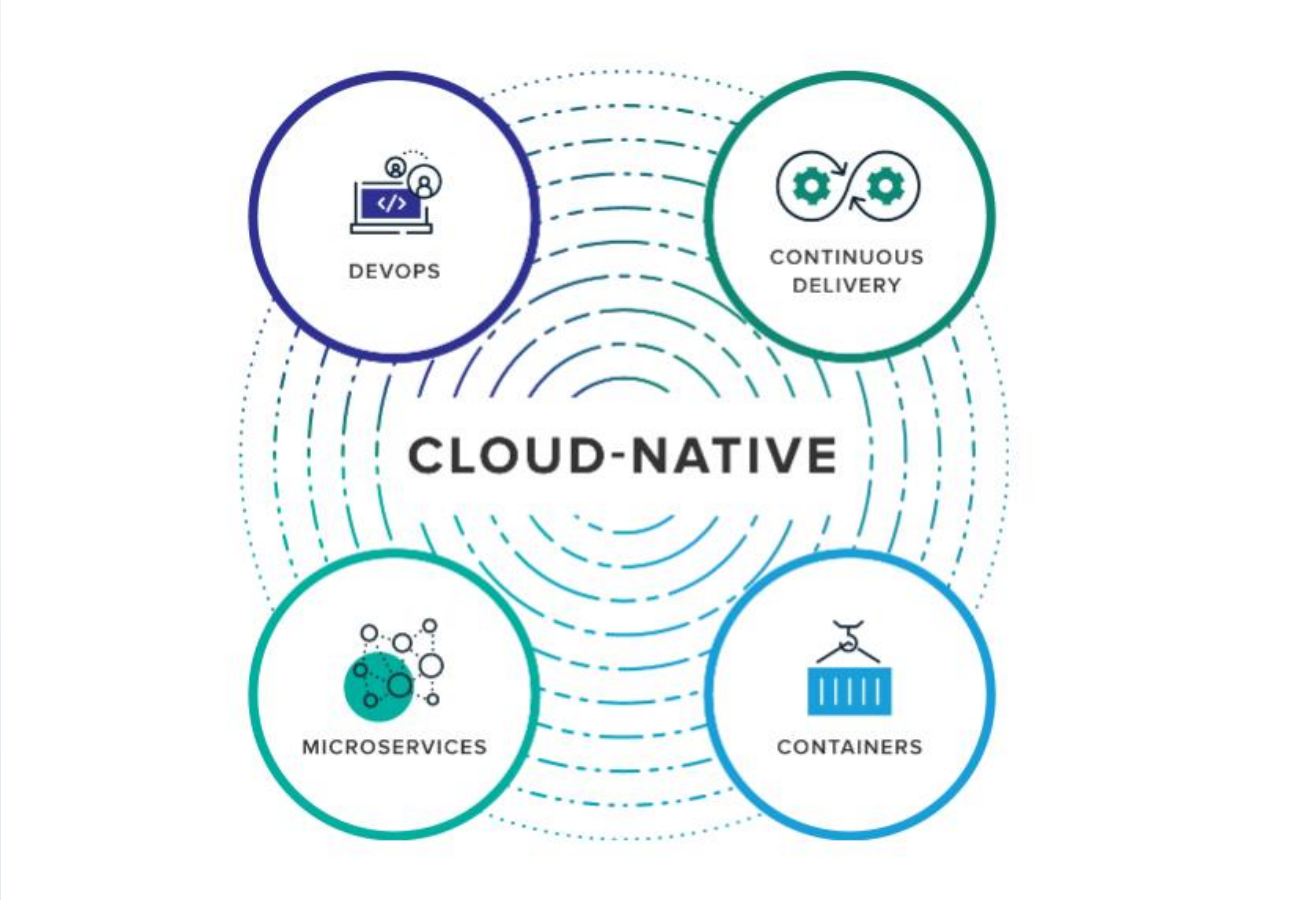
«Маршрутка»

Cloud native

Cloud native — подход к созданию и выполнению приложений, использующий преимущества облачной модели, подходит для частных и публичных облаков.

Обычно такие приложения строятся как набор микросервисов, слабо связанных между собой и упакованных в контейнеры, управляются они облачной платформой. Облачная платформа может предлагать вычислительные мощности по требованию.

Cloud native



Cloud native

DevOps
Реализация гибкой методологии разработки - постоянный и автоматизированный выпуск инкрементальных изменений программного обеспечения.

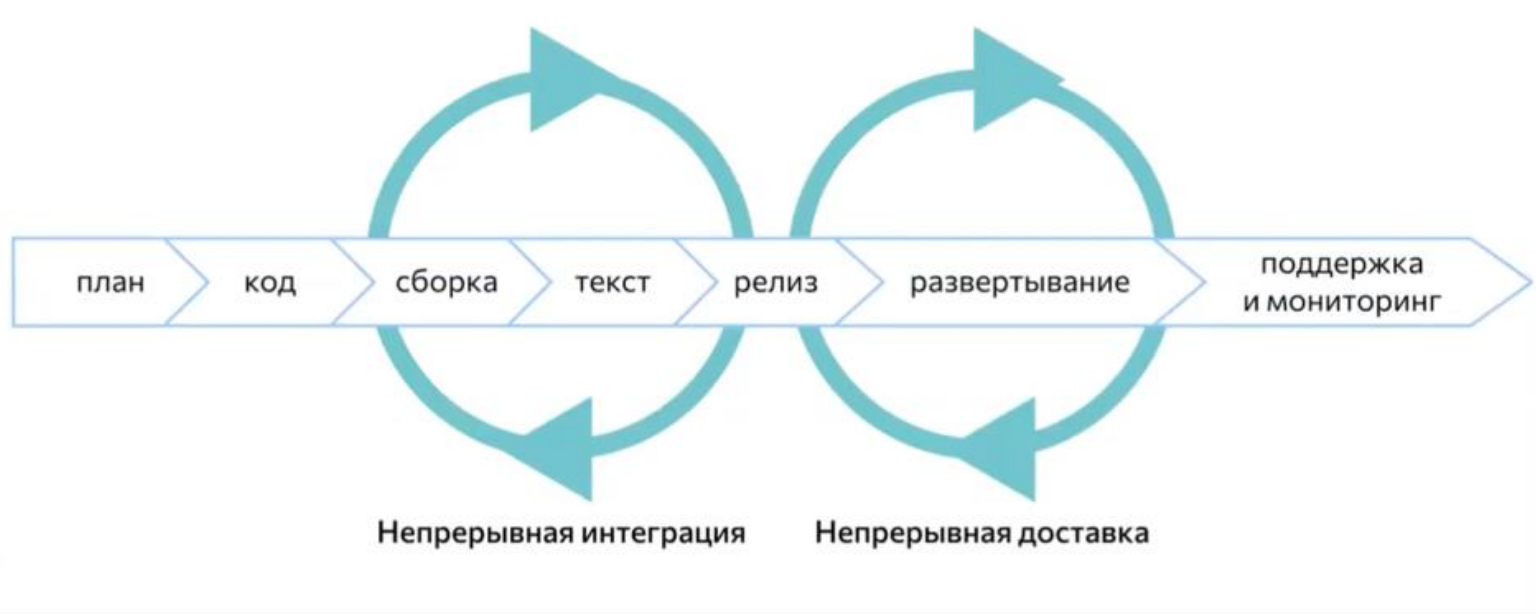
CI/CD
Релизы делаются чаще и с меньшими рисками за счет стандартизированных процедур. Быстрая обратная связь от пользователей.

Микросервисы.
Архитектурный подход к разработке приложения как набора небольших сервисов. Каждый сервис реализует определенную логику, его можно развернуть, обновить, масштабировать или перезапустить независимо от других служб приложения-нет необходимости выключать всю систему целиком при обновлении.

Контейнеры
Контейнеры эффективнее и быстрее стандартных виртуальных машин (ВМ). Используя виртуализацию на уровне операционной системы (ОС), один экземпляр ОС динамически распределяется между одним или несколькими изолированными контейнерами, у каждого из которых уникальная файловая система и свой объем выделенных ресурсов.

Цикл CI/CD

- **CI (Continuous Integration)** — непрерывная интеграция. Проверка основной ветки репозитория: каждый раз после мёржа в рамках CI-пайплайна выполняются автоматические тесты.
- **CD, (Continuous Delivery)** — непрерывная поставка - автоматическое развертывание на стенды и тестовые окружения.



Инструментальные средства автоматизации развертывания

- SSH (SCP) – свобода творчества
- Jenkins / Teamcity/Gitlab(Github) – CI/CD
- Urban code deploy - CD
- *Kubernetes – not today ☺*
- Infrastructure as code:
  - Ansible
  - Terraform

Инструментальные средства автоматизации развертывания

SSH (SCP)

scp опции пользователь1@хост1:файл пользователь2@хост2:файл

Опции:

-1 - использовать протокол SSH1;

-2 - использовать протокол SSH2;

-B - пакетный режим для передачи нескольких файлов;

-C - включить сжатие;

- l - установить ограничение скорости в кбит/сек;

-o - задать нужную опцию SSH;

-p - сохранять время модификации;

-r - рекурсивное копирование директорий;

-v - более подробный режим.

Инструментальные средства автоматизации развертывания

- Jenkins / Teamcity / Gitlab (Github)

Агенты внутри

Внешние Агенты

Инструментальные средства автоматизации развертывания

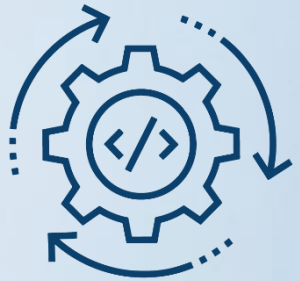
- Urban code deploy

urban{code}

НЕ CI, только CD, требует установки агентов

UrbanCode Deploy - это решение для автоматизации выпуска приложений, совмещающее средства просмотра, отслеживания и контроля в одном оптимизированном пакете. Поддерживается масштабирование до развертывания уровня предприятия, включающего несколько тысяч серверов.

Инфраструктура как Код (IaC)



IaC — это процесс управления и создания серверов и кластеров с помощью машиночитаемых файлов определений, созданный как альтернатива физическому конфигурированию оборудования и оперируемым человеком инструментам.

- **Скорость:** IaC позволяет быстрее конфигурировать инфраструктуру и направлен на обеспечение прозрачности, помочь другим командам работать быстрее и эффективнее.

- **Масштабируемость и стандартизация:** стабильные среды быстро и на должном уровне. Командам разработчиков не нужно прибегать к ручной настройке, описывая с помощью кода требуемое состояние сред. Развертывания инфраструктуры с помощью IaC повторяемы и предотвращают проблемы во время выполнения, вызванных дрейфом конфигурации или отсутствием зависимостей.

- **Безопасность и документация:** при конфигурации заданной через код, стандарты безопасности можно легко и последовательно применять. IaC также служит некой формой документации о правильном способе создания инфраструктуры. Поскольку код можно версионировать, IaC позволяет документировать, регистрировать и отслеживать каждое изменение конфигурации вашего сервера.

Ansible

Программное решение для удаленного управления конфигурациями. Оно позволяет настраивать удаленные машины. Главное его отличие от других подобных систем в том, что Ansible использует существующую инфраструктуру SSH.

Особенности:

- **Безагентное**. В клиенте не установлено программное обеспечение или агент, который общается с сервером.
- **Идемпотентное**. Независимо от того, сколько раз вы вызываете операцию, результат будет одинаковым.
- **Простое и расширяемое**. Программа Ansible написана на Python и использует YAML для написания команд. Оба языка считаются относительно простыми в изучении.

Ansible

Состав:

Control machine — управляющий хост. Сервер Ansible, с которого происходит управление другими хостами

Manage node — управляемые хосты

Inventory — инвентарный файл. В этом файле описываются хосты, группы хостов, а также могут быть созданы переменные

Playbook — файл сценариев

Play — сценарий (набор задач). Связывает задачи с хостами, для которых эти задачи надо выполнить

Task — задача. Вызывает модуль с указанными параметрами и переменными

Module — модуль Ansible. Реализует определенные функции

Ansible

Установка:

Требования к управляющему хосту:

- поддержка Python 3)
- Windows не может быть управляющим хостом

pip install ansible

Ansible

Группы серверов

Список групп серверов для управления, два способа получения:

- Локальный файл: **/etc/ansible/hosts**
- Внешний скрипт, в официальном github-репозитории есть готовые скрипты для получения списка из:
  - Digital Ocean,
  - OpenStack Nova,
  - Openshift,
  - Vagrant,
  - Zabbix и др

Ansible

Hosts-файл

/etc/ansible/hosts – по-умолчанию, но может быть задано переменной окружения $ANSIBLE_HOSTS или параметром -i при запуске ansible и ansible-playbook.

Пример:

[group1]

one.example.com

two.example.com

[group2]

three.example.com

Помимо списка управляемых узлов, в файле hosts могут быть указаны и другие сведения, необходимые для работы: номера портов для подключения по SSH, способ подключения, пароль для подключения по SSH, имя пользователя, объединения групп

Ansible

Hosts-файл

Можно добавить диапазон хостов:

[routers]

192.168.255.[1:5]

Или по имени

[switches]

switch[A:D].example.com

Ansible

Hosts-файл, дочерние группы

[routers]

192.168.255.[1:5]

[switches]

switch[A:D].example.com

[devices:children]

routers

switches

Ad-hoc команды

ansible 192.168.0.1 -i hosts.ini -c network_cli –k -u user -m ios_command -a "commands='sh clock'"

- 192.168.0.1 – хост, к которому нужно применить действия, должен существовать в инвентарном файле
- -i myhosts.ini - параметр -i позволяет указать инвентарный файл
- -c network_cli - тип подключения. В данном случае через SSH
- -u user – выполнить от имени пользователя
- -k - аутентификация по паролю
- -m ios_command – используемый модуль
- -a "commands='sh ip int br'" – команда для выполнения

Ansible

Модули

В состав Ansible входит большое количество модулей для развёртывания, контроля и управления различными компонентами:

- облачные ресурсы и виртуализация (Openstack, libvirt);
- базы данных (MySQL, Postgresql, Redis, Riak);
- файлы (шаблонизация, регулярные выражения, права доступа);
- мониторинг (Nagios, monit);
- оповещения о ходе выполнения сценария (Jabber, Irc, почта, MQTT, Hipchat);
- сеть и сетевая инфраструктура (Openstack, Arista);
- управление пакетами (apt, yum, rhn-channel, npm, pacman, pip, gem);
- система (LVM, Selinux, ZFS, cron, файловые системы, сервисы, модули ядра)

Ansible Playbooks

Playbook (файл сценариев) – файл с описанием действий для выполнения на хосте и группе хостов.

Структура:

- play – сценарий, состоит из описания, конфигурации и списка задач
- task - конкретная задача реализуемая в рамках сценария. В задаче должно быть:
  - описание (название задачи можно не писать, но очень рекомендуется)
  - модуль и команда (действие в модуле)

Ansible Playbooks

```
- name: Install aptitude and required packs
  hosts: workers
  tasks:
    - name: Install aptitude
      apt:
        name: aptitude
        state: latest
        update_cache: true

    - name: Install required system packages
      apt:
        pkg:
          - apt-transport-https
          - ca-certificates
          - curl
          - software-properties-common
        state: latest
        update_cache: true
    - name: Add Docker GPG apt Key
      apt_key:
        url: https://download.docker.com/linux/ubuntu/gpg
        state: present
```

Ansible Playbooks

Include

позволяют подключать в текущий playbook файлы с задачами.

```
---

- name: Install aptitude and required packs
  hosts: workers
  tasks:
    - name: Install aptitude
      apt:
        name: aptitude
        state: latest
        update_cache: true
    - include: tasks/install_apt.yml
```

Ansible

Переменные

Переменные можно создавать:

- в inventory
- в playbook
- в файлах для групп/хостов
- в отдельных файлах, которые добавляются в playbook через include
- в ролях
- передавать при вызове playbook

Ansible

Переменные в inventory

[routers]

192.168.255.[1:5]

[routers:vars]

ansible_connection=network_cli

ansible_user=user

ansible_password=password

Ansible

Переменные в playbook

```
- name: Run command on host
  hosts: workers
  gather_facts: false
  vars:
    cmd: cat /file
  tasks:
    - name: run command
      command: "{{cmd}}"
```

Ansible

Переменные в файлах групп, хостов

Во время развертывания, необходимо не только установить приложение, но и настроить его в соответствии с определенными параметрами на основании принадлежности к группе серверов или индивидуально :

- Файлы переменных групп в директории “group_vars/имя_группы”;
- Файлы переменных хостов в директории “hosts_vars/имя_хоста”;
- Файлы с переменными в директории “имя_роли/vars/имя_задачи.yml”;

```
├── group_vars _
│   ├── all.yml
│   ├── routers.yml
│   └── switches.yml
├── host_vars
│   ├── 192.168.0.1
│   ├── 192.168.0.2
└── myhosts.ini
```

Ansible

Роли

Роли это способ логического разбития файлов Ansible, это просто автоматизация выражений include- не нужно явно указывать полные пути к файлам с задачами или сценариями, а достаточно лишь соблюдать определенную структуру файлов

```
├── all_roles.yml
├── role1.yml
├── role2.yml
└── roles
    ├── role1
    │   ├── files
    │   ├── templates
    │   ├── tasks
    │   ├── handlers
    │   ├── vars
    │   ├── defaults
    │   └── meta
```

Ansible

Роли

- Все роли определены в каталоге roles
- Дочерние каталоги называются именем ролей
- Внутри каталога роли могут быть предопределенные каталоги - как минимум, tasks.
- Внутри каталогов tasks, handlers, vars, defaults, meta автоматически считывается всё, что находится в файле main.yml
- Файлы из каталогов добавляются через include, на файлы s можно ссылаться не указывая путь к ним, достаточно имени файла

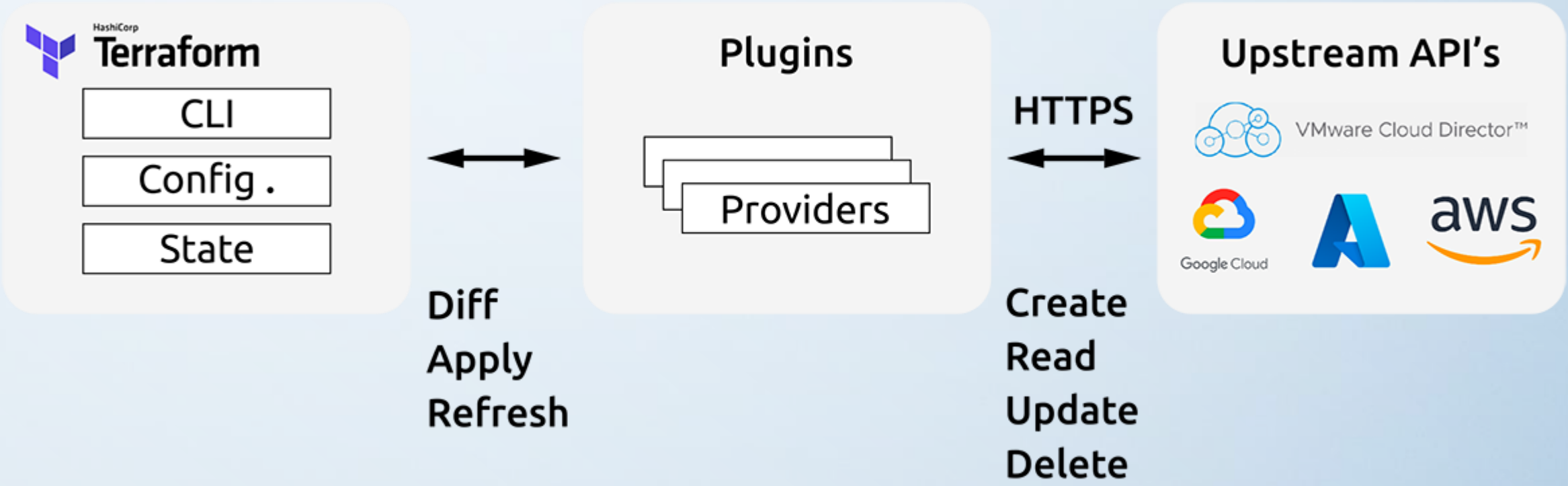
Terraform

Terraform — это Open source решение для управления IaC от компании Hashicorp, вышедшее в 2014 году. Это система управления состоянием инфраструктуры, придерживающаяся подхода идемпотентности и декларативного стиля управления.

В файлах конфигураций хранится описание инфраструктуры на языке HCL (HashiCorp Configuration Language)

Возможность расширения инструментария за счет установки дополнительных модулей.

Terraform


HashiCorp
Terraform
CLI
Config .
State
Diff
Apply
Refresh
Plugins
Providers
HTTPS
Create
Read
Update
Delete
Upstream API's
VMware Cloud Director™
Google Cloud
aws

Terraform

1. Желаемое состояние инфраструктуры описывается в конфигурационном файле;
2. В нем же указывается провайдер, который будет исполнять работу;
3. Применяется последовательность 3 команд:
    1. terraform init — инициализация провайдера;
    2. terraform plan — валидация конфигурационного файла;
    3. terraform apply — применение конфигурации.

Terraform

1. Желаемое состояние инфраструктуры описывается в конфигурационном файле;
2. В нем же указывается провайдер, который будет исполнять работу;
3. Применяется последовательность 3 команд:
    1. terraform init — инициализация провайдера;
    2. terraform plan — валидация конфигурационного файла;
    3. terraform apply — применение конфигурации.


ВОПРОС ?
ОТВЕТ !

Список литературы

- https://habr.com/ru/company/alloy_software/blog/274167/
- https://cdto.wiki/Развитие_ИТ-инфраструктуры
- https://mcs.mail.ru/blog/cloud-native-prilozheniya-bystro-zagruzhayutsya-snizhayut-riski-stimuliruyut-rost-biznesa
- https://losst.ru/kopirovanie-fajlov-scp
- https://habr.com/ru/company/southbridge/blog/691876/
- https://habr.com/ru/company/tinkoff/blog/532546/
- https://habr.com/ru/post/305400/
- https://habr.com/ru/company/otus/blog/574278/
- https://habr.com/ru/post/305400/
- https://ansible-for-network-engineers.readthedocs.io/_/downloads/ru/latest/pdf/
- https://docs.ansible.com/