

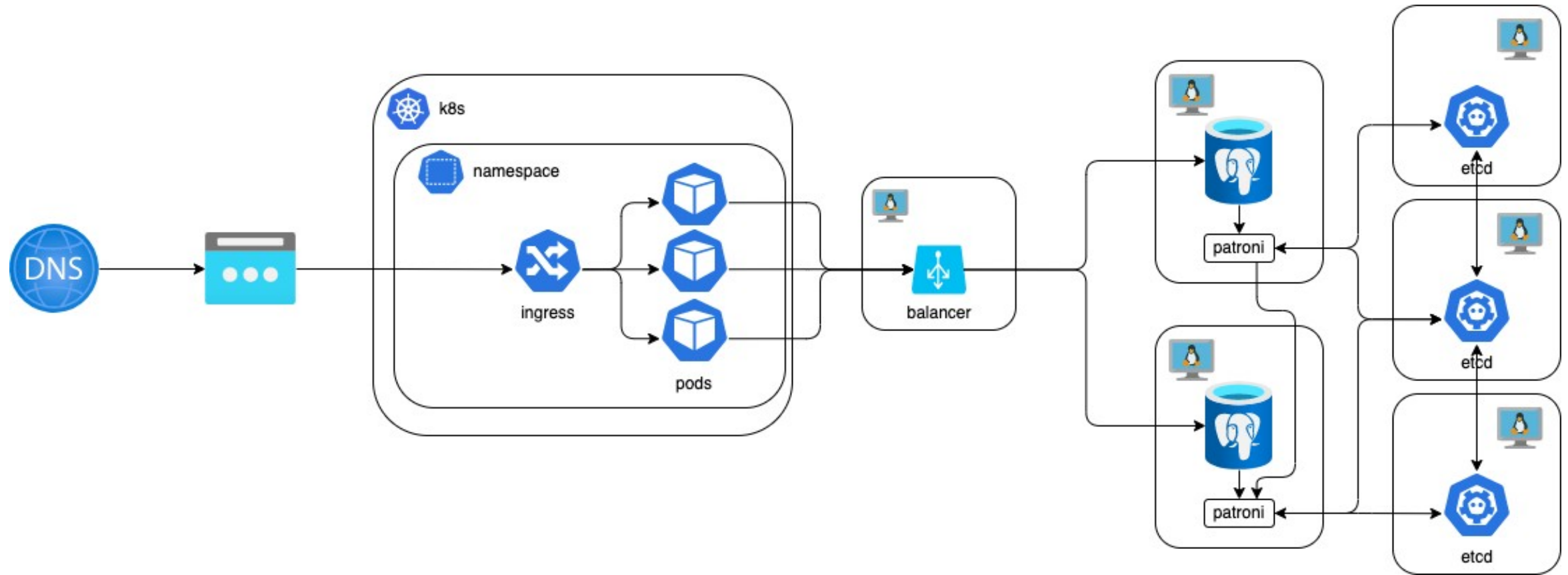
Как безопасно работать с инфраструктурой

О чем сегодня поговорим

- Посмотрим на окружение,
с которым будем работать
- Для чего вообще нам необходимо
в автоматизацию?
- Системы управления конфигурациями (SCM)

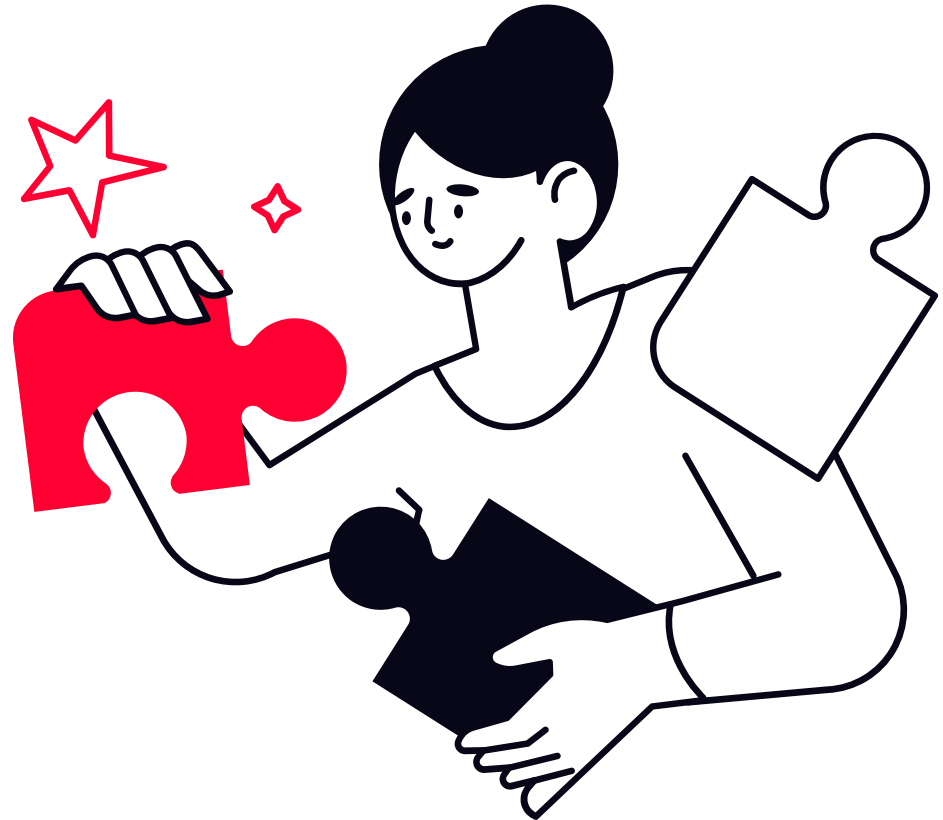


High-Level Design



Используемые инструменты

1. Github для хранения кода и ci/cd
2. Ansible для написания playbook для работы с виртуалками
3. Helm для работы с нашим namespace в кубе



Для чего нам столько всего?

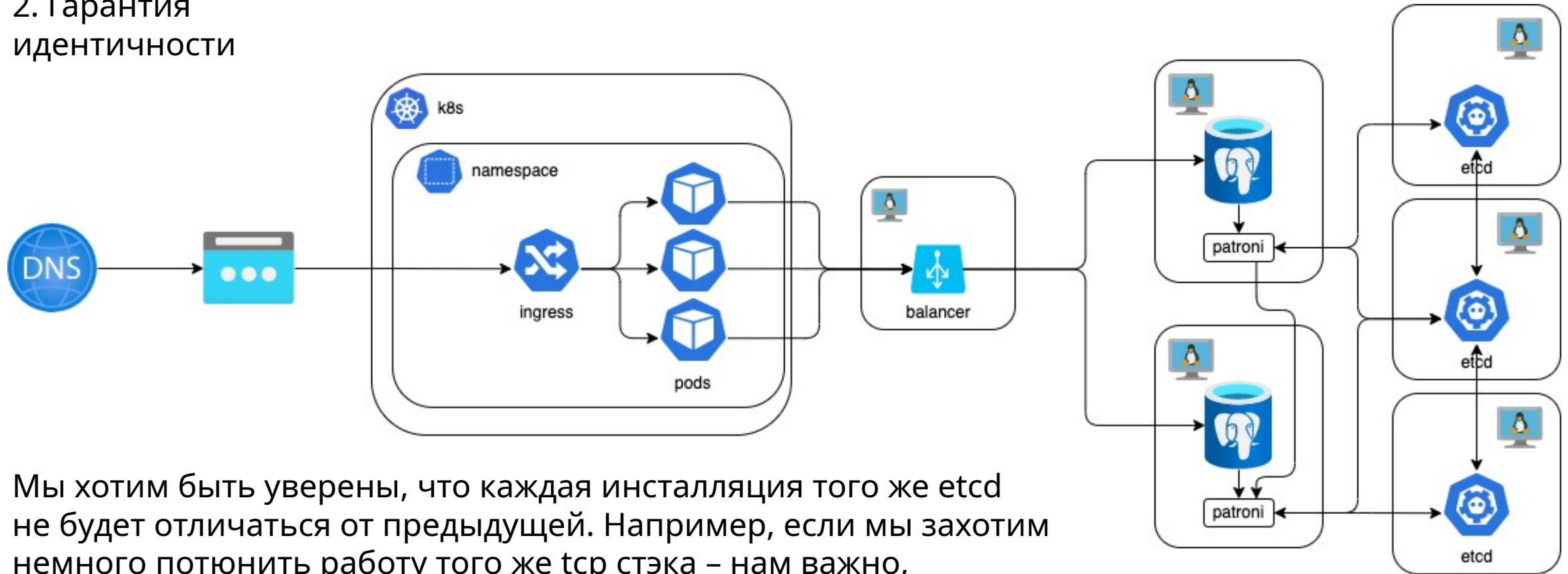
1. Инфраструктура не надежна



Максимум, что мы можем выжать из инфраструктуры за вменяемые деньги – это 95%. Когда, например, упадет машина с балансиром – нам нужно ее будет как-то восстановить, а делать это руками – весьма длительная операция

Для чего нам столько всего?

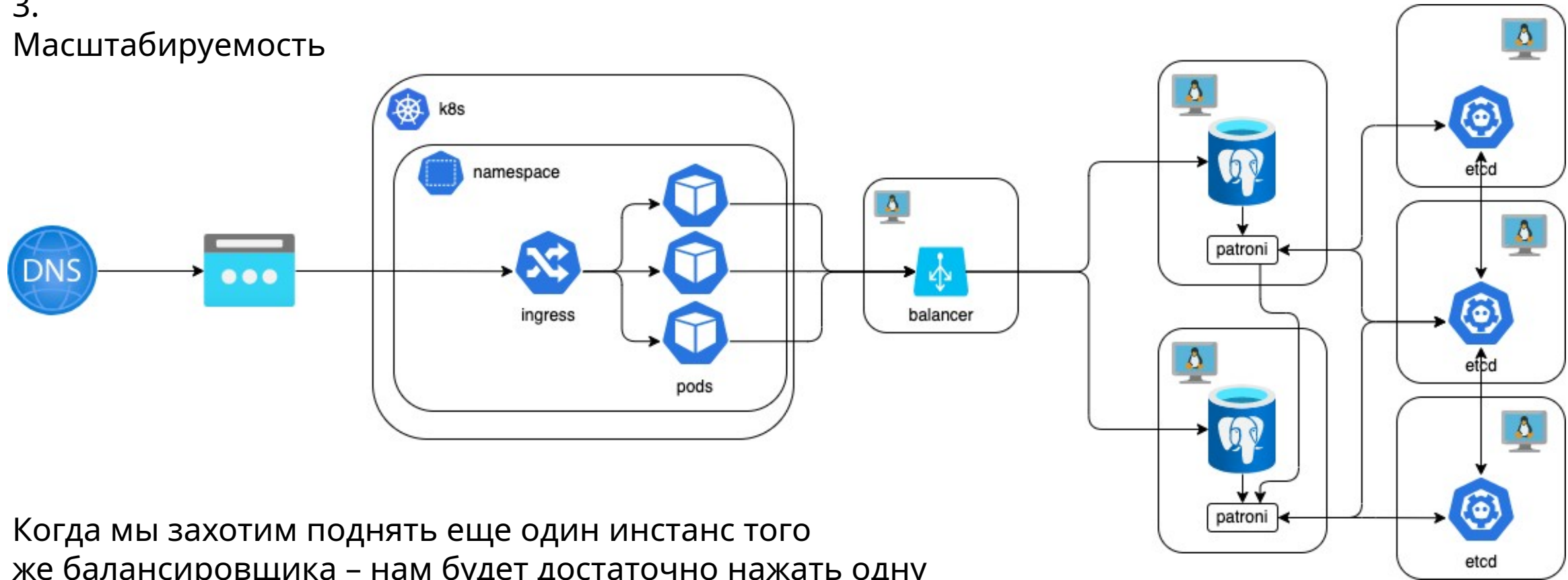
2. Гарантия идентичности



Мы хотим быть уверены, что каждая инсталляция того же etcd не будет отличаться от предыдущей. Например, если мы захотим немного потюнить работу того же tcp стэка – нам важно, чтобы это было одинаково для каждой машины

Для чего нам столько всего?

3. Масштабируемость



Когда мы захотим поднять еще один инстанс того же балансировщика – нам будет достаточно нажать одну кнопку, а не конфигурировать по аналогии с текущей машиной с нуля

Что такое ansible

Open-source программное обеспечение для автоматизации IT. Решает задачи от конфигурации до реализации процессов непрерывной доставки

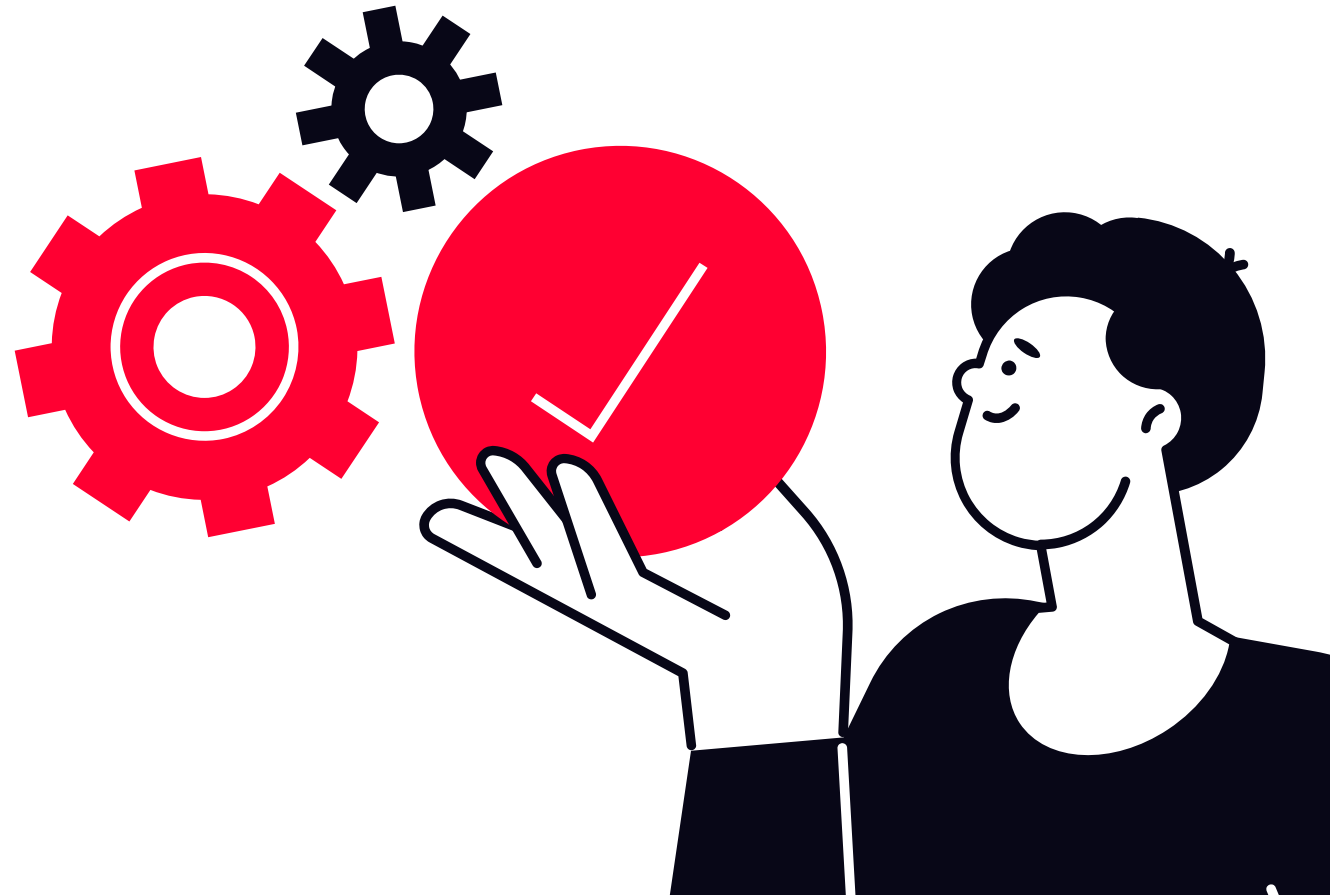
Для чего **ansible**

- Менеджер конфигураций
- Развертывание приложений
- Создание окружений



Особенности **ansible**

- Не требует установки агента
- Использует ssh подключение.
Нужен только установленный python
- Конфигураций как код
- Дружелюбный yaml



Основные **сущности**

1. Инвентарь
2. Task -> Role -> Play -> Playbook
3. Шаблоны
4. Переменные



Ansible инвентарь и как его готовить?

Inventory – набор файлов, описывающих инфраструктуру

Два формата описания

1 формат

```
all:
  children:
    etcd:
      hosts:
        91.185.85.39:
        91.185.85.40:
        91.185.85.41:
    haproxy:
      hosts:
        91.185.85.42:
    dbservers:
      hosts:
        91.185.85.38:
          hostname: pgnode01
          postgresql_exists: false
        91.185.85.37:
          hostname: pgnode02
          postgresql_exists: false
```

2 формат

```
[etcd]
91.185.85.39
91.185.85.40
91.185.85.41

[haproxy]
91.185.85.42

[dbservers]
91.185.85.38 hostname=pgnode01 postgresql_exists=false
91.185.85.37 hostname=pgnode02 postgresql_exists=false
```

Ansible Инвентарь

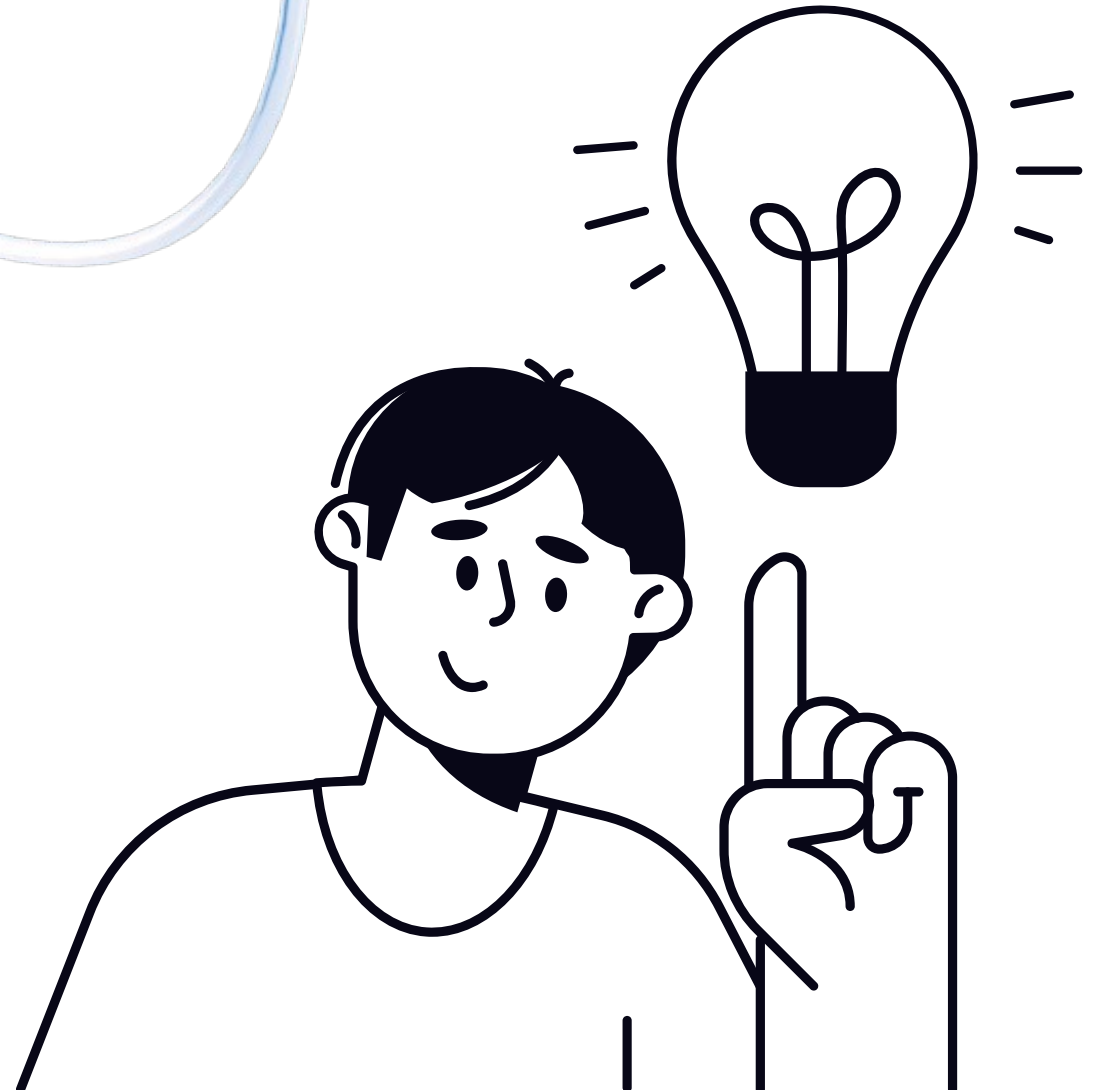
```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
        three.example.com:
    east:
      hosts:
        foo.example.com:
        one.example.com:
        two.example.com:
    west:
      hosts:
        bar.example.com:
        three.example.com:
    prod:
      hosts:
        foo.example.com:
        one.example.com:
        two.example.com:
    test:
      hosts:
        bar.example.com:
        three.example.com:
```



Ansible

ХОСТЫ

```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one. example.com:
        two.example.com:
        three.example.com:
    east:
      hosts:
        foo.example.com:
        one.example.com:
        two.example.com:
    west:
      hosts:
        bar.example.com:
        three.example.com:
  prod:
    children:
      east:
  test:
    hosts:
      bar.example.com:
      three.example.com:
```



Ansible

ХОСТЫ

```
inventory
├── postgres.yml
├── production.yml
└── sre-course.yml
```

```
ansible all -m ping -i inventory/sre-course.yml
```

```
ansible-playbook example.yml -i inventory
```

```
ansible-playbook example.yml -i inventory/production.yml
```



Ansible

Хосты Переменные

```
all:
  children:
    etcd:
      hosts:
        91.185.85.39:
        91.185.85.40:
        91.185.85.41:
    haproxy:
      hosts:
        91.185.85.42:
    dbservers:
      hosts:
        91.185.85.38:
          hostname: pgnode01
          postgresql_exists: false
        91.185.85.37:
          hostname: pgnode02
          postgresql_exists: false
      vars:
        ansible_ssh_port: 22
  vars:
    ansible_connection: ssh
    ansible_ssh_port: 222
```

```
[etcd]
91.185.85.39
91.185.85.40
91.185.85.41

[haproxy]
91.185.85.42

[dbservers]
91.185.85.38 hostname=pgnode01 postgresql_exists=false
91.185.85.37 hostname=pgnode02 postgresql_exists=false

[dbservers:vars]
ansible_ssh_port=22

[all:vars]
ansible_connection='ssh'
ansible_ssh_port='222'
```

Ansible

Хосты Переменные

```
all:
  children:
    etcd:
      hosts:
        91.185.85.39:
        91.185.85.40:
        91.185.85.41:
    haproxy:
      hosts:
        91.185.85.42:
    dbservers:
      hosts:
        91.185.85.38:
          hostname: pgnode01
          postgresql_exists: false
        91.185.85.37:
          hostname: pgnode02
          postgresql_exists: false
      vars:
        ansible_ssh_port: 22
      vars:
        ansible_connection: ssh
        ansible_ssh_port: 222
```

```
[etcd]
91.185.85.39
91.185.85.40
91.185.85.41

[haproxy]
91.185.85.42

[dbservers]
91.185.85.38 hostname=pgnode01 postgresql_exists=false
91.185.85.37 hostname=pgnode02 postgresql_exists=false

[dbservers:vars]
ansible_ssh_port=22

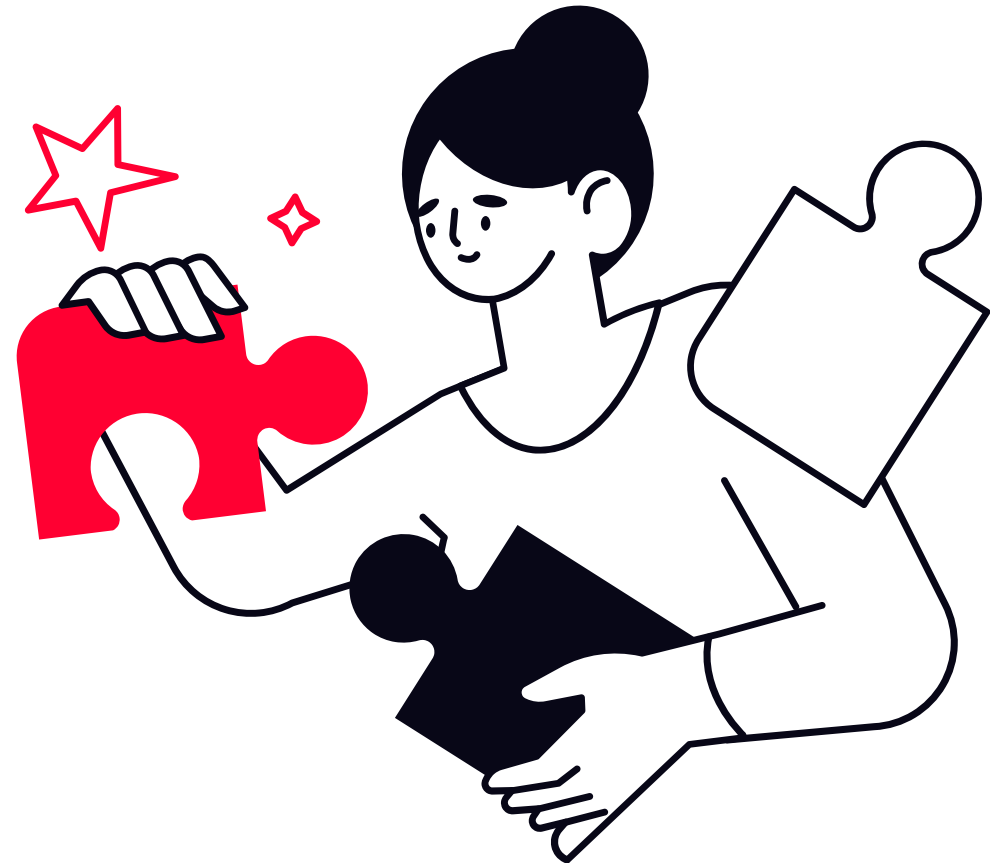
[all:vars]
ansible_connection='ssh'
ansible_ssh_port='222'
```

Ansible
Task -> Role -> Play -> Playbook

Ansible роль. Структура

```
ansible-galaxy init simple-test
```

```
├── README.md
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
├── vars
│   └── main.yml
```



Ansible task

```
- name: Start patroni service on Replica servers
  ansible.builtin.systemd:
    daemon_reload: true
    name: patroni
    state: restarted
    enabled: true
```

```
- ansible.builtin.import_tasks: custom_wal_dir.yml
  when: postgresql_wal_dir is defined and postgresql_wal_dir | length > 0
  tags: patroni, custom_wal_dir, point_in_time_recovery
```

```
- name: Install patroni package
  ansible.builtin.package:
    name: "{{ patroni_packages | default('patroni') }}"
    state: present
    register: package_status
    until: package_status is success
    delay: 5
    retries: 3
    when: ansible_os_family == "Debian" and patroni_deb_package_repo | length < 1
```

Ansible **task**. Ключевые слова

- when
- become
- changed_when
- loop / with_items
- ignore_errors
- Until/Retries
- vars

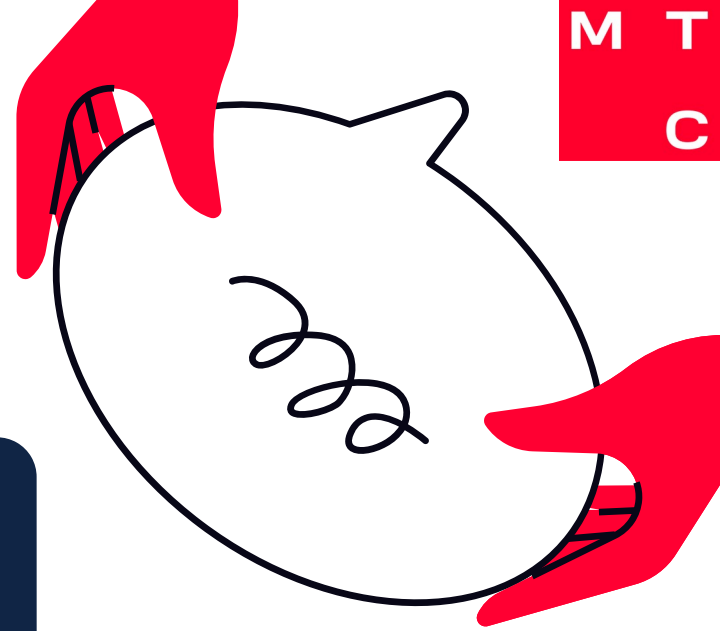
https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html#task



Ansible handlers

```
- name: Update pg_hba.conf
  ansible.builtin.template:
    src: ../templates/pg_hba.conf.j2
    dest: "{{ postgresql_conf_dir }}/pg_hba.conf"
    owner: postgres
    group: postgres
    mode: "0640"
  notify: "reload postgres"
  tags: pg_hba, pg_hba_generate
```

```
- name: Reload postgres
  become: true
  become_user: postgres
  ansible.builtin.command: "{{ postgresql_bin_dir }}/psql -p {{ postgresql_port }} -c 'SELECT pg_reload_conf()'"
  register: psql_reload_result
  changed_when: psql_reload_result.rc == 0
  failed_when: false # exec pg_reload_conf on all running postgres (to re-run with --tag pg_hba).
  listen: "reload postgres"
```



Ansible шаблоны

```
{% if patroni_callbacks is defined and patroni_callbacks | length > 0 %}  
callbacks:  
  {% for callback in patroni_callbacks %}  
    {% if callback.script | length > 0 %}  
      {{ callback.action }}: '{{ callback.script }}'  
    {% endif %}  
  {% endfor %}  
{% endif %}
```

```
- name: Generate conf file "/etc/patroni/patroni.yml"  
  ansible.builtin.template:  
    src: templates/patroni.yml.j2  
    dest: /etc/patroni/patroni.yml  
    owner: postgres  
    group: postgres  
    mode: "0640"  
  when: existing_pgcluster is not defined or not existing_pgcluster|bool  
  tags: patroni, patroni_conf
```



Ansible **playbook**

```
- name: Deploy PostgreSQL HA Cluster (based on "Patroni" and "{{ dcs_type }}")
hosts: all
become: true
become_method: sudo
gather_facts: true
tags: always
any_errors_fatal: true
vars_files:
  - vars/main.yml
  - vars/system.yml
environment: "{{ proxy_env | default({}) }}"

roles:
  - role: pre-checks
    vars:
      minimal_ansible_version: 2.11.0
      timescale_minimal_pg_version: 12 # if enable_timescale is defined
    tags: always

tasks:
  - name: Clean yum cache
    ansible.builtin.command: yum clean all
    when:
      - ansible_os_family == "RedHat"
      - ansible_distribution_major_version == '7'

  - name: Clean dnf cache
    ansible.builtin.command: dnf clean all
    when:
      - ansible_os_family == "RedHat"
      - ansible_distribution_major_version is version('8', '>=')
```

Ansible. Приоритеты переменных

- command line values (for example, -u my_user)
- role defaults (defined in role/defaults/main.yml)
- inventory file or script group vars
- inventory group_vars/all
- playbook group_vars/all
- inventory group_vars/*

- playbook group_vars/*
- inventory file or script host vars
- inventory host_vars/*
- playbook host_vars/*
- host facts / cached set_facts

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable

Ansible. Приоритеты переменных

- play vars
- play vars_prompt
- play vars_files
- role vars (defined in role/vars/main.yml)
- block vars (only for tasks in block)
- task vars (only for the task)
- include_vars
- set_facts / registered vars

- role (and include_role) params
- include params
- extra vars (for example, -e "user=my_user")(always win precedence)

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable

Ansible **Playbook**

```
ansible-playbook -i inventory/some_inventory.yml -e some_var=some_value one_of_playbooks.yml  
ansible-playbook -i inventory/some_inventory.yml another_playbook.yml
```



Вопросы?

Что такое helm

Инструмент для применения, обновления
и управления приложениями в Kubernetes

Какие задачи может выполнять **helm**

- Установка программного обеспечения.
- Автоматическая установка зависимостей программного обеспечения.
- Обновление программного обеспечения.
- Настройка развертывания программного обеспечения.
- Доставка пакетов программного обеспечения из репозиториев.



Из чего состоит **helm**

- Инструмент командной строки helm.
- Сопутствующий серверный компонент tiller
- Charts
- Официальный курируемый репозиторий с готовыми пакетами для популярных проектов программного обеспечения с открытым исходным кодом <https://artifacthub.io/>



Создание базовой структуры

```
helm create sre-course
```

```
sre-course
├── Chart.yaml
├── charts
├── templates
│   ├── NOTES.txt
│   ├── _helpers.tpl
│   ├── deployment.yaml
│   ├── hpa.yaml
│   ├── ingress.yaml
│   ├── service.yaml
│   ├── serviceaccount.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml
```

charts:

сюда можно помещать управляемые вручную зависимости пакета, хотя обычно лучше использовать файл `requirements.yaml` для динамической привязки зависимостей.

templates:

содержит файлы шаблона, которые комбинируются со значениями конфигурации (из файла `values.yaml` и командной строки) и записываются в манифесты Kubernetes.

Chart.yaml:

файл YAML с метаданными о пакете, включая название и версию пакета, информацию об обслуживании, актуальный сайт и ключевые слова для поиска.

Создание базовой структуры

```
helm create sre-course
```

```
sre-course
├── Chart.yaml
├── charts
├── templates
│   ├── NOTES.txt
│   ├── _helpers.tpl
│   ├── deployment.yaml
│   ├── hpa.yaml
│   ├── ingress.yaml
│   ├── service.yaml
│   ├── serviceaccount.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml
```

LICENSE:

лицензия пакета в текстовом формате.

README.md:

файл readme с информацией для пользователей пакета.

requirements.yaml:

файл YAML с перечислением зависимостей пакета.

values.yaml:

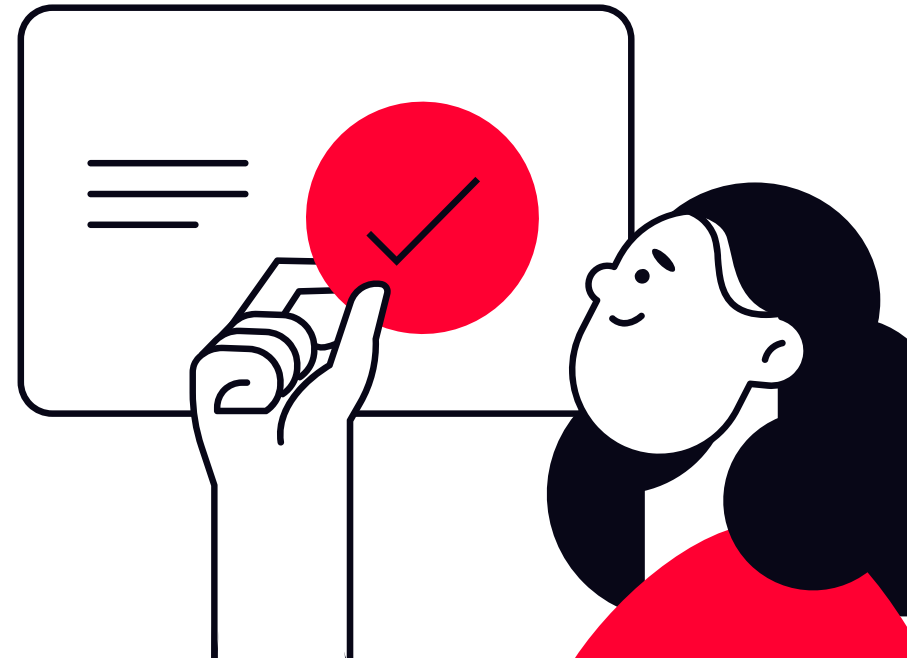
файл YAML со значениями конфигурации пакета по умолчанию.

Проверка Helm chart

```
helm template sre-course
```

```
helm template sre-course -s templates/service.yaml
```

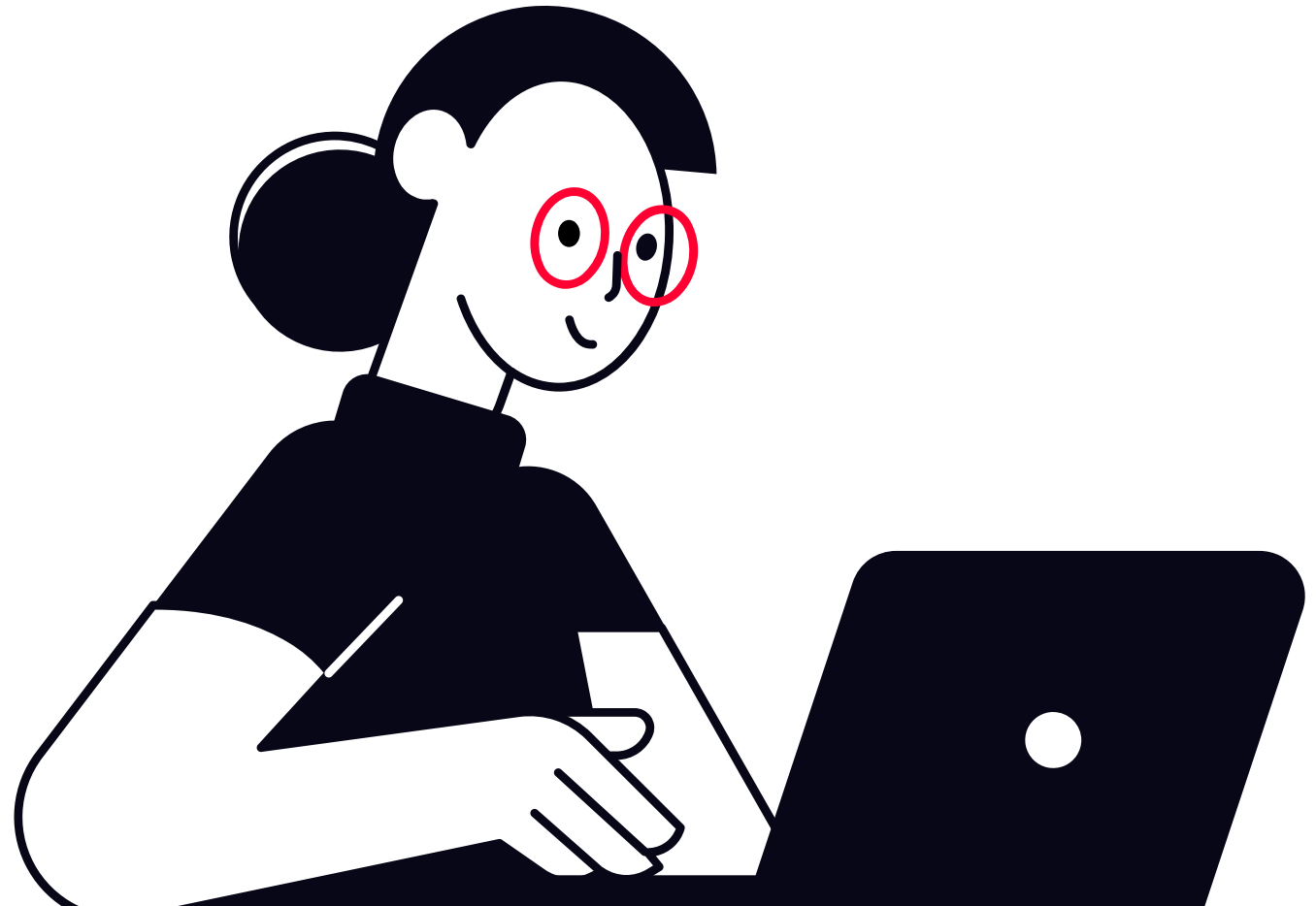
```
helm template sre-course -s templates/service.yaml -f customvalues.yaml
```



Установка Helm chart

```
helm install sre-course --dry-run --debug
```

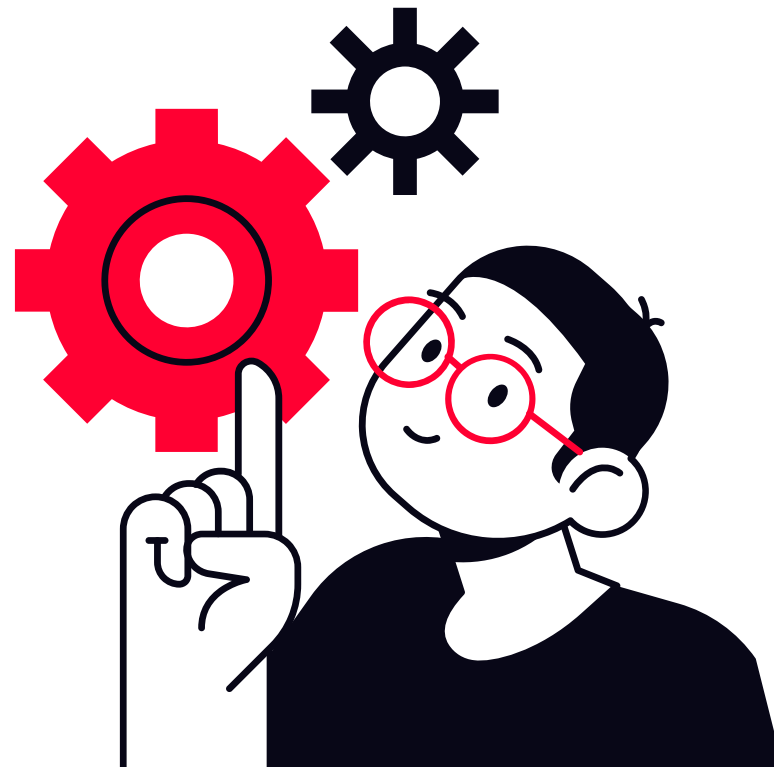
```
helm install sre-course
```



Helm функции

```
{{/*  
Expand the name of the chart.  
*/}}  
{{- define "sre-course.name" -}}  
{{- default .Chart.Name .Values.nameOverride | trunc 5 | trimSuffix "-" -}}  
{{- end -}}
```

```
app.kubernetes.io/name: {{ include "sre-course.name" . }}
```



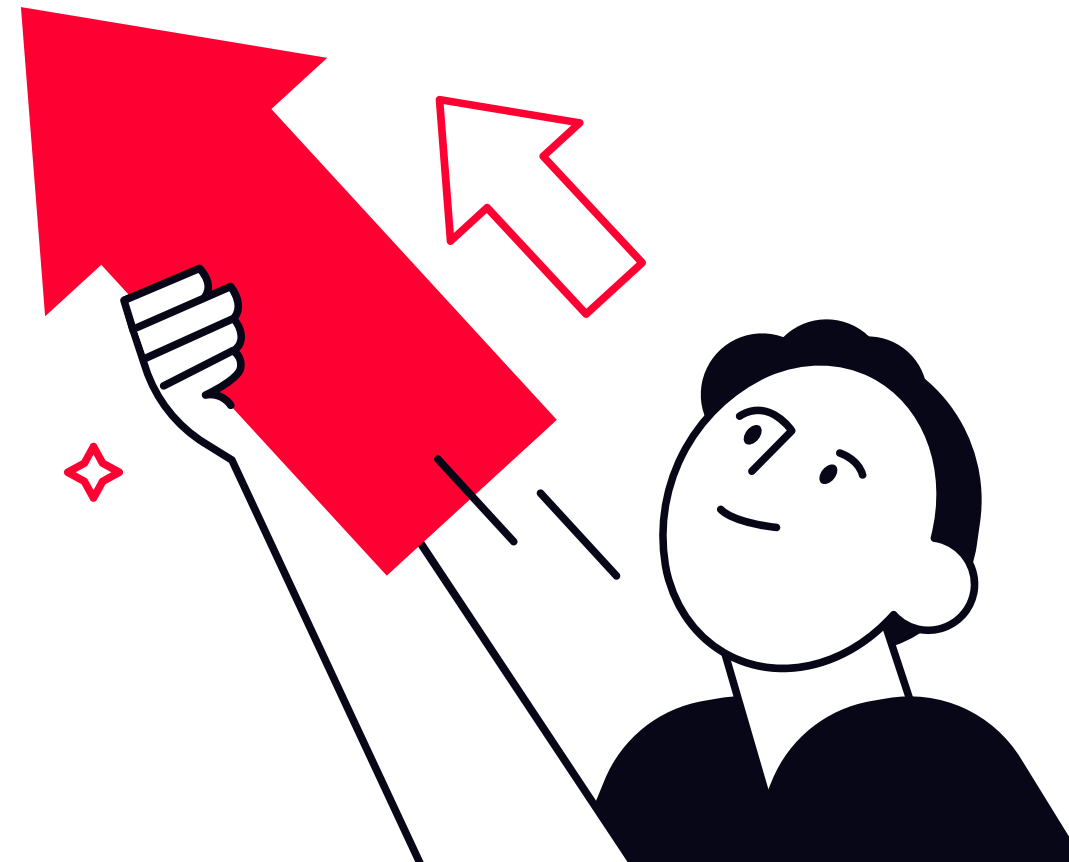
Helm labels

Common labels

```
*/}}
{{- define "sre-course.labels" -}}
helm.sh/chart: {{ include "sre-course.chart" . }}
{{ include "sre-course.selectorLabels" . }}
{{- if .Chart.AppVersion }}
app.kubernetes.io/version: {{ .Chart.AppVersion | quote }}
{{- end }}
app.kubernetes.io/managed-by: {{ .Release.Service }}
{{- end }}
```

```
apiVersion: v1
kind: Service
metadata:
  name: {{ include "sre-course.fullname" . }}
  labels:
    {{ include "sre-course.labels" . | indent 4 }}
...
```

```
kubectl get svc -l helm.sh/chart=sre-course-0.1.0
```



Helm комментарии

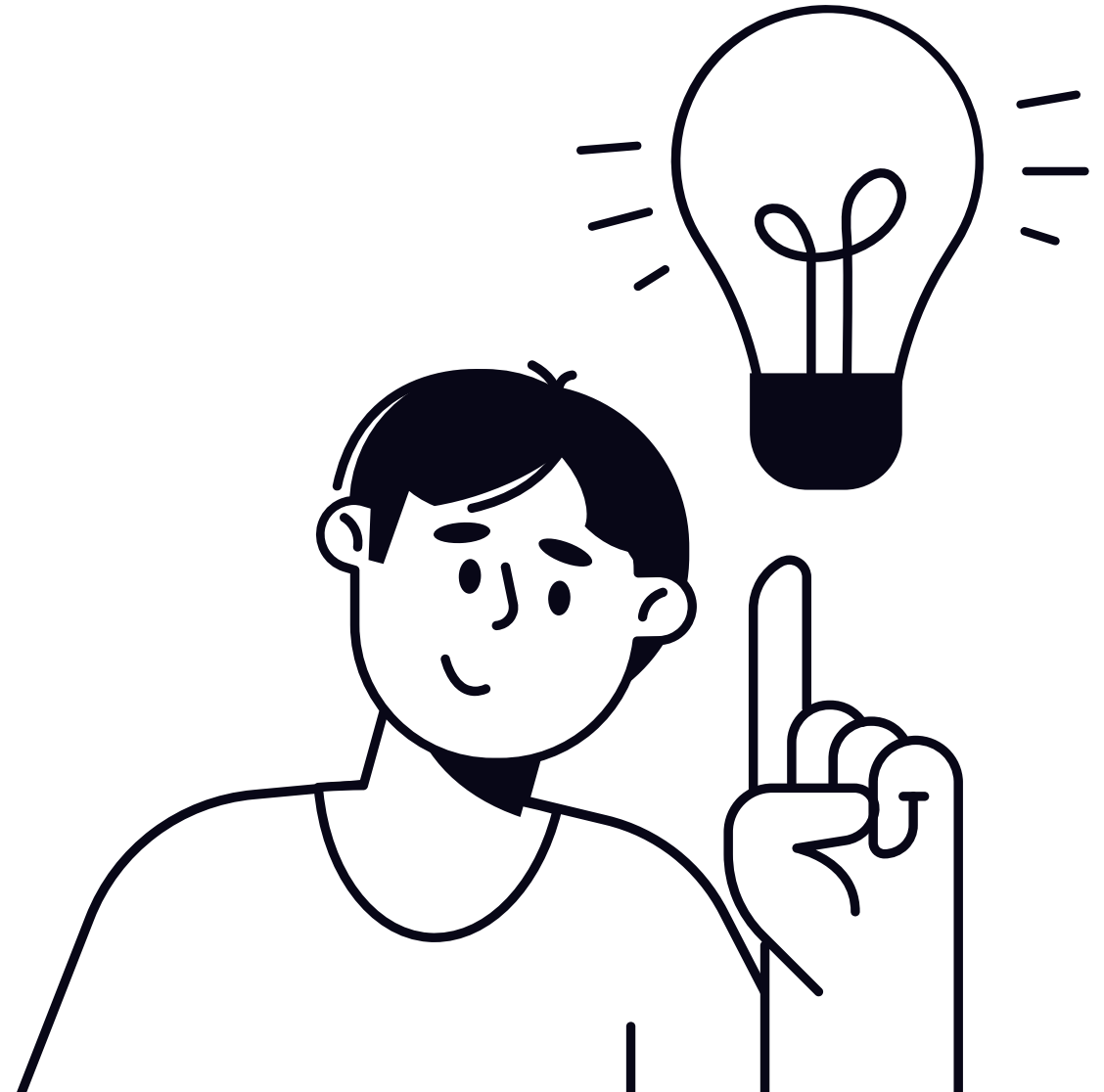
- # — простой комментарий, который остается в результирующем YML после обработки.
- {{- /* ... */ -}} — комментарий, который отбрасывается обработчиком шаблонов.



Helm subcharts

```
cd charts  
helm create sub-sre-course
```

```
apiVersion: v2  
name: sre-course  
description: A Helm chart for Kubernetes  
type: application  
version: 0.1.0  
appVersion: "1.16.0"  
dependencies:  
- name: sub-sre-course
```



Что стоит помнить, работая с **helm**

- Имена ресурсов — максимум 63 символа
- Имена ресурсов могут состоять только из цифр, строчных букв, "-" или "."
- Размер чарта — не более 1 МБ
- В чарте есть функция для парсинга .tpl
- Вы можете указать ресурсы, которые останутся после удаления деплоя чарта командой `helm delete`
- Для деплоя на конкретный кластер можно использовать – `kubeconfig /path/to-k8s-config.kubeconfig`

Вопросы?