

# DevOps

Л09. Docker compose

Виктор Моисеев  
+7-902-83-145-30  
[t.me/v\\_paranoid](https://t.me/v_paranoid)  
[victorparanoid@gmail.com](mailto:victorparanoid@gmail.com)

## Л09. Docker compose

Многоконтейнерные приложения с использованием Compose

Практика:

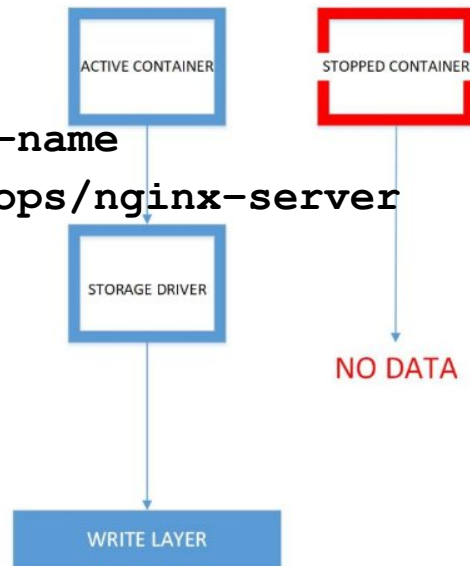
Собираем стек: Flask+Redis

Собираем стек: Prometheus+Graphana, blackbox\_exporter

См. <https://docs.docker.com/compose/gettingstarted/>

# Основные команды docker

```
$ docker create --name <container name> <image name>
$ docker start/stop <container name>
$ docker run -it --name <container name> <image name>
$ docker pause/unpause <container name>
$ docker rm <container name>
$ docker run -v /host-path:/container-path image-name
# docker run -d --name nginx-cnt -p 54321:80 devops/nginx-server
# docker build -t nginx-do .
$ docker image ls
$ docker exec -it nginx-cnt /bin/bash
```





# Open Container Initiative

The **Open Container Initiative** is an open governance structure for the express purpose of creating open industry standards around container formats and runtimes.

Established in June 2015 by Docker and other leaders in the container industry, the OCI currently contains three specifications: the Runtime Specification (runtime-spec), the Image Specification (image-spec) and the Distribution Specification (distribution-spec). The Runtime Specification outlines how to run a “filesystem bundle” that is unpacked on disk. At a high-level an OCI implementation would download an OCI Image then unpack that image into an OCI Runtime filesystem bundle. At this point the OCI Runtime Bundle would be run by an OCI Runtime.

# Docker Compose overview

Docker Compose is a tool for defining and running multi-container applications. It is the key to unlocking a streamlined and efficient development and deployment experience.

Compose simplifies the control of your entire application stack, making it easy to manage services, networks, and volumes in a single, comprehensible YAML configuration file. Then, with a single command, you create and start all the services from your configuration file.

# Основные команды docker-compose-v2

```
# compose ищет в текущем каталоге compose.yaml  
# с описанием сервисов  
# и все команды выполняет относительно него
```

```
docker compose up -d
```

```
    # поднять сервисы
```

```
docker compose down    # отключить сервисы
```

```
docker compose logs    # смотреть логи
```

```
docker compose ps      # смотреть статус сервисов
```

```
docker compose up --watch    # пересобирать при изменении исходников
```

```
# остальные docker-команды также продолжают работать
```

# Структура файла compose.yaml

*##### сервисы = контейнеры*

services:

frontend:

build: example/webapp

ports:

- "443:8043"

networks:

- front-tier

- back-tier

configs:

- httpd-config

secrets:

- server-certificate

backend:

image: example/database

volumes:

- db-data:/etc/data

networks:

- back-tier

*##### продолжение, ресурсы*

volumes:

db-data:

driver: flocker

driver\_opts:

size: "10GiB"

configs:

httpd-config:

external: true

secrets:

server-certificate:

external: true

networks:

# The presence is sufficient to define them

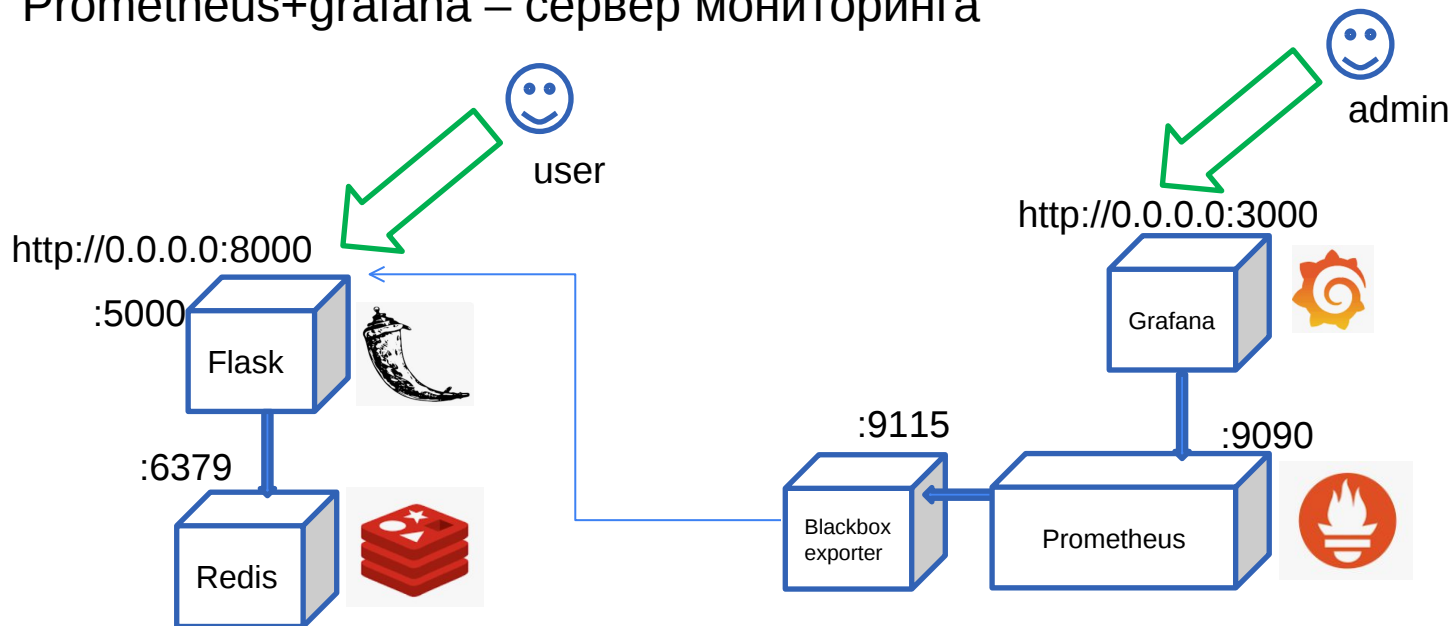
front-tier: {}

back-tier: {}

# П09. Docker compose- практика

Создать два стека:

1. Flask+redis – веб-приложение со счетчиком входов
2. Prometheus+grafana – сервер мониторинга





# 1. Собираем веб-приложение (flask+redis)

- Устанавливаем compose

```
sudo apt update
```

```
sudo apt install docker-compose-v2
```

- Создаем каталог для проекта

```
~$ mkdir devops9compose  
~$ cd devops9compose/  
~/devops9compose$
```

# Создаем веб-приложение (app.py)

```
import time

import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    return cache.incr('hits')

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times.\n'.format(count)
```

redis - dns имя соседнего контейнера с БД

# Объявляем зависимости для python

```
GNU nano 7.2 requirements.txt *
flask
redis
```

# Создаем dockerfile для контейнера с flask

```
# syntax=docker/dockerfile:1
FROM python:3.10-alpine          # легковесный линукс с питоном
WORKDIR /code
ENV FLASK_APP=app.py             # переменные окружения
ENV FLASK_RUN_HOST=0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt # ставим зависимости
EXPOSE 5000                      # заметка для админа, какой порт потом надо публиковать
COPY app.py .                    # копируем из текущего каталога в workdir
CMD ["flask", "run", "--debug"]  # запускаем flask
```

БД Redis берем дефолтную, ничего не требуется менять

# Создаем compose.yaml – описание 2х сервис-контейнеров

```
services:  
  web:                                # веб-сервис ("web" - это dns имя хоста в докер сети)  
    build: .                          # образ будет собран локально из докерфайла  
    ports:  
      - "8000:5000"                    # проброс внешнего порта 8000 в контейнер на 5000  
  redis:                             # БД ("redis" - это dns имя хоста в докер сети)  
    image: "redis:alpine"            # образ будет скачан с публичного докерхаба
```

# Запускаем оба сервис-контейнера Flask+Redis

- Compose выполняет указания из файла `compose.yaml`

```
victor@ubuntu:~/devops9compose$ docker compose up -d
[+] Running 7/9
  ⋮ redis 8 layers [#####] 0B/0B Pulling
    ✓43c4264eed91 Already exists
    ✓0d8647d21597 Pull complete
    ✓165578b9d4d3 Pull complete
    ✓a79b9261ec8d Pull complete
  ⋮ ca65ba09a6bb Waiting
    ✓1fb5bb2cba03 Download complete
    ✓4f4fb700ef54 Download complete
    ✓14e22361b667 Download complete
=> [web 5/6] RUN pip install -r requirements.txt
=> [web 6/6] COPY app.py .
=> [web] exporting to image
=> => exporting layers
=> => writing image sha256:2fb0aa3a804a051ab63e712
=> => naming to docker.io/library/devops9compose-v
[+] Running 2/3
  ⋮ Network devops9compose_default Created
    ✓Container devops9compose-web-1 Started
    ✓Container devops9compose-redis-1 Started
victor@ubuntu:~/devops9compose$
```

# Проверка стека Flask+redis

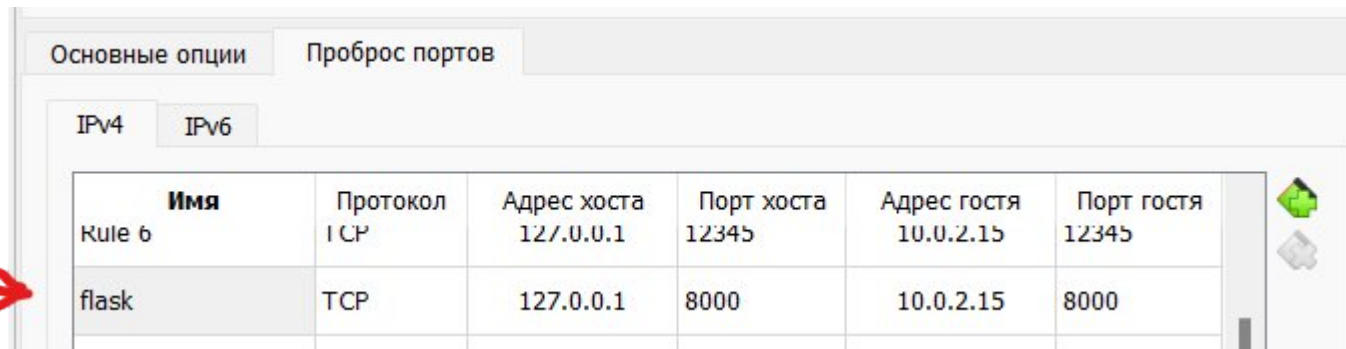
```
victor@ubuntu:~/devops9compose$ docker compose ps
```

NAME	IMAGE	COMMAND	SERVICE	CREATED
devops9compose-redis-1	redis:alpine	"docker-entrypoint.s..."	redis	About a minute ago
Up About a minute	6379/tcp			
devops9compose-web-1	devops9compose-web	"flask run --debug"	web	About a minute ago
Up About a minute	0.0.0.0:8000->5000/tcp, :::8000->5000/tcp			

```
victor@ubuntu:~/devops9compose$ curl 127.0.0.1:8000
Hello World! I have been seen 1 times.
victor@ubuntu:~/devops9compose$ curl 127.0.0.1:8000
Hello World! I have been seen 2 times.
victor@ubuntu:~/devops9compose$ curl 127.0.0.1:8000
Hello World! I have been seen 3 times.
victor@ubuntu:~/devops9compose$
```



# Проверка стека Flask+redis



The screenshot shows the Mikrotik WinBox interface for configuring a Firewall Rule. The 'Port Forward' tab is active. A red arrow points to the 'Rule Name' field, which contains the text 'flask'.

Имя	Протокол	Адрес хоста	Порт хоста	Адрес гостя	Порт гостя
Rule 6	ICMP	127.0.0.1	12345	10.0.2.15	12345
flask	TCP	127.0.0.1	8000	10.0.2.15	8000

← → ↻ ⓘ 127.0.0.1:8000

o HRM x Confl CD HD ESMP Z Z zbx2 ⚙

Hello World! I have been seen 11 times.

## 2. Собираем мониторинг (prometheus+grafana)

- Создаем каталог для проекта

```
~/devops9compose$ cd ..  
~$ mkdir devops9prom  
~$ cd devops9prom/  
~/devops9prom$
```

Будущая структура каталогов

```
.  
├── compose.yaml  
├── grafana  
│   └── datasource.yml  
├── prometheus  
│   └── prometheus.yml
```

# Создаем compose.yaml для Prometheus+Grafana

```
services:
  prometheus:
    image: prom/prometheus
    container_name: prometheus
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
    ports:
      - 9090:9090
    restart: unless-stopped
    volumes:
      - ./prometheus:/etc/prometheus
      - prom_data:/prometheus
  grafana:
    image: grafana/grafana
    container_name: grafana
    ports:
      - 3000:3000
    restart: unless-stopped
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=grafana
    volumes:
      - ./grafana:/etc/grafana/provisioning/datasources
volumes:
  prom_data:
```

# Создаем конфиг для prometheus

```
mkdir prometheus
nano prometheus/prometheus.yml
```

```
global:
  scrape_interval: 5s
  scrape_timeout: 3s
  evaluation_interval: 15s
alerting:
  alertmanagers:
    - static_configs:
        - targets: []
      scheme: http
      timeout: 10s
      api_version: v1
scrape_configs:
- job_name: prometheus
  honor_timestamps: true
  scrape_interval: 15s
  scrape_timeout: 10s
  metrics_path: /metrics
  scheme: http
  static_configs:
    - targets:
        - localhost:9090
```

# Создаем конфиг для grafana

```
mkdir grafana
```

```
nano grafana/datasource.yml
```

```
apiVersion: 1
```

```
datasources:
```

```
- name: Prometheus
```

```
  type: prometheus
```

```
  url: http://prometheus:9090
```

```
  isDefault: true
```

```
  access: proxy
```

```
  editable: true
```

# Запускаем оба сервис-контейнера Prometheus+Grafana

- Compose выполняет указания из файла compose.yaml

```
victor@ubuntu:~/devops9prom$ docker compose up -d
[+] Running 24/24
✓prometheus 12 layers [#####] 0B/0B Pulled
  ✓9fa9226be034 Pull complete
  ✓1617e25568b2 Pull complete
  ✓25b95a09a872 Pull complete
  ✓9010eb24e726 Pull complete
  ✓faa5b6876931 Pull complete
  ✓7773e9699356 Pull complete
  ✓0f68bbe907b1 Pull complete
  ✓4357144b1367 Pull complete
  ✓49f3e8dc63fd Pull complete
[+] Running 2/4
  :: Network devops9prom_default Created
  :: Volume "devops9prom_prom_data" Created
  ✓Container prometheus Started
  ✓Container grafana Started
```

# Проверка мониторинга Prometheus+Grafana

```
victor@ubuntu:~/devops9prom$ docker compose ps
```

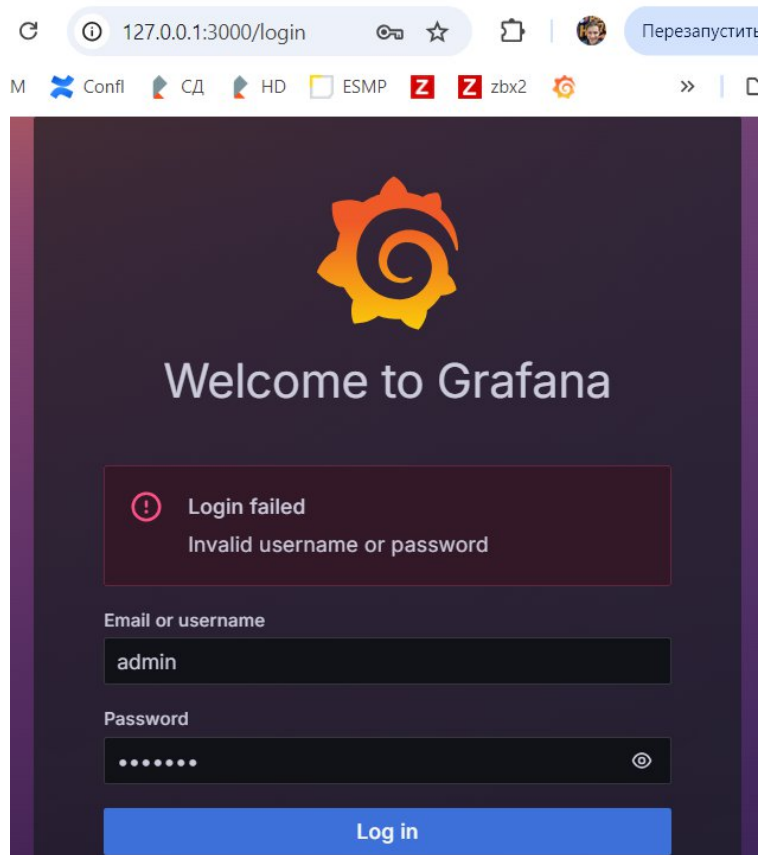
NAME	IMAGE	COMMAND	SERVICE	CREATED	STATUS
grafana	grafana/grafana	"/run.sh"	grafana	6 minutes ago	Up 5 seconds
0.0.0.0:3000->3000/tcp, :::3000->3000/tcp					
prometheus	prom/prometheus	"/bin/prometheus --c..."	prometheus	6 minutes ago	Up 10 seconds
0.0.0.0:9090->9090/tcp, :::9090->9090/tcp					

IPv4IPv6

Имя	Протокол	Адрес хоста	Порт хоста	Адрес гостя	Порт гостя
flask	TCP	127.0.0.1	8000	10.0.2.15	8000
grafana	TCP	127.0.0.1	3000	10.0.2.15	3000

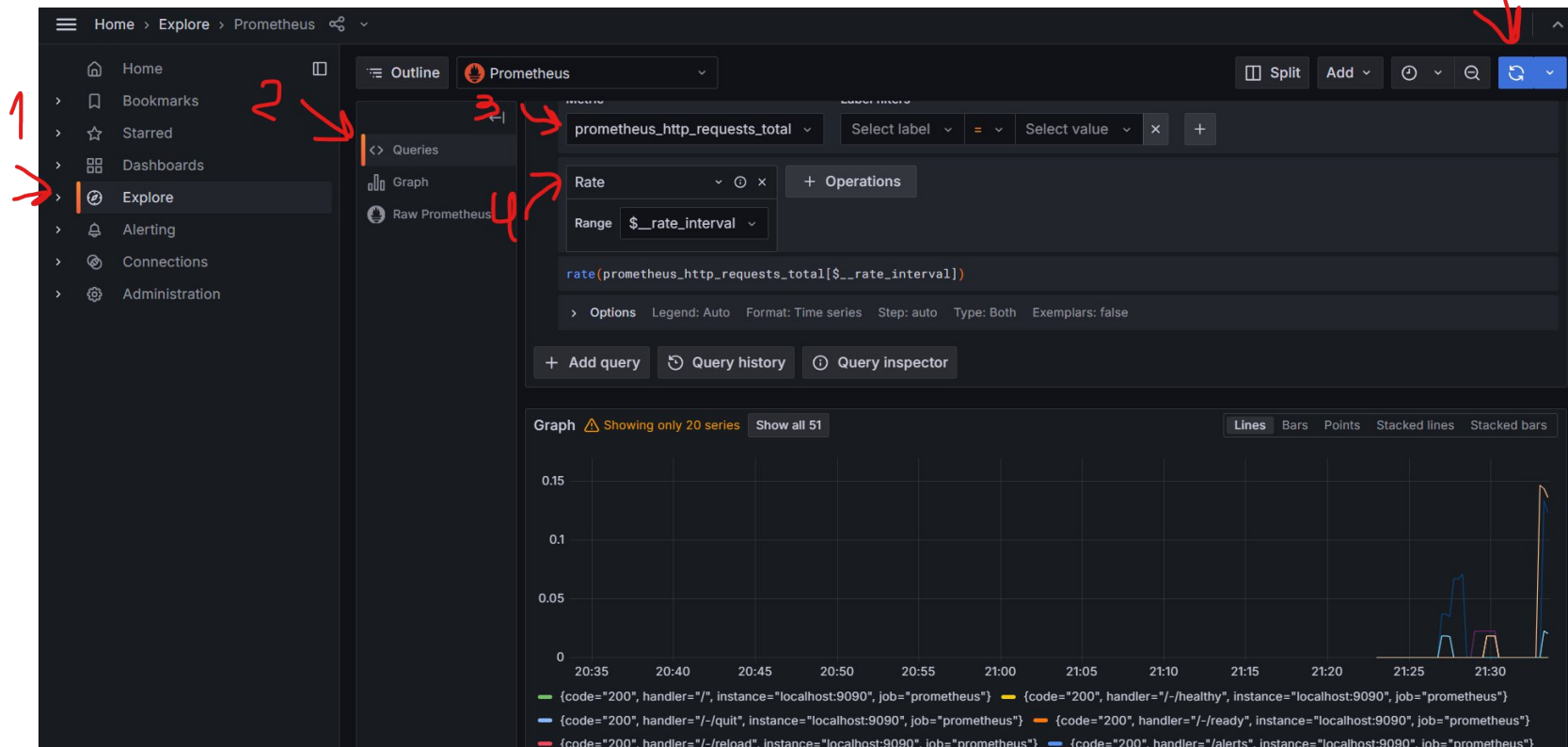
# Проверка мониторинга Prometheus+Grafana

- admin/grafana





# Смотрим метрику самого Prometheus



## Добавляем мониторинг нашего Flask-приложения (и не только)

Добавляем в `compose.yml` описание нового контейнера – `blackbox`

```
blackbox:  
  image: prom/blackbox-exporter  
  container_name: blackbox  
  ports:  
    - 9115:9115
```

# Добавляем мониторинг нашего Flask-приложения (и не только)

- Обновляем конфиг prometheus, добавляем в секцию **scrape\_configs** указания кого мониторить через blackbox

```
- job_name: blackbox-http
  metrics_path: /probe
  params:
    module: [http_2xx]
  static_configs:
    - targets:
      - http://10.0.2.15:8000    # Flask                # цели для опроса (sic 10.0.2.15)
      - https://etis.psu.ru
      - https://student.psu.ru
  relabel_configs:
    - source_labels: [__address__]
      target_label: __param_target
    - source_labels: [__param_target]
      target_label: instance
    - target_label: __address__
      replacement: blackbox:9115    # кому направлять запрос на опрос целей (sic blackbox)
```

# Пересобираем стек мониторинга

- Выключаем старую сборку контейнеров мониторинга

```
victor@ubuntu:~/devops9prom$ docker compose down  
[+] Running 3/3  
✓ Container prometheus      Removed  
✓ Container grafana         Removed  
✓ Network devops9prom_default Removed
```

# Пересобираем стек мониторинга

- Запускаем новую сборку контейнеров мониторинга

```
victor@ubuntu:~/devops9prom$ docker compose up -d
[+] Running 5/5
 ✓ blackbox 4 layers [#####] 0B/0B Pulled
   ✓ 9fa9226be034 Already exists
   ✓ 1617e25568b2 Already exists
   ✓ 692bab6142b7 Pull complete
   ✓ bcd1ef44e061 Pull complete
[+] Running 3/4
 ⚡ Network devops9prom_default Created
 ✓ Container grafana Started
 ✓ Container blackbox_exporter Started
 ✓ Container prometheus Started
victor@ubuntu:~/devops9prom$ |
```

# Проверка экспортера

```
victor@ubuntu:~/devops9prom$ docker compose ps
NAME                IMAGE                        COMMAND                                SERVICE    CREATED
STATUS             PORTS
blackbox_exporter   prom/blackbox-exporter      "/bin/blackbox_expor..."  blackbox    42 seconds ago
Up 41 seconds      0.0.0.0:9115->9115/tcp, :::9115->9115/tcp
grafana             grafana/grafana             "/run.sh"                      grafana     42 seconds ago
Up 42 seconds      0.0.0.0:3000->3000/tcp, :::3000->3000/tcp
prometheus          prom/prometheus             "/bin/prometheus --c..."  prometheus  42 seconds ago
Up 41 seconds      0.0.0.0:9090->9090/tcp, :::9090->9090/tcp
victor@ubuntu:~/devops9prom$ curl 127.0.0.1:9115
<html>
  <head><title>Blackbox Exporter</title></head>
  <body>
    <h1>Blackbox Exporter</h1>
    <p><a href="probe?target=prometheus.io&module=http_2xx">Probe prometheus.io for http_2xx</a></p>
    <p><a href="probe?target=prometheus.io&module=http_2xx&debug=true">Debug probe prometheus.io for
http_2xx</a></p>
    <p><a href="metrics">Metrics</a></p>
    <p><a href="config">Configuration</a></p>
    <h2>Recent Probes</h2>
    <table border='1'><tr><th>Module</th><th>Target</th><th>Result</th><th>Debug</th></tr></table></body>
</html>victor@ubuntu:~/devops9prom$
```

# Создаем дашборд

- (например импортируя 13659 с сайта графаны)

The image shows the Grafana 'Import dashboard' interface. On the left is a sidebar with navigation links: Home, Bookmarks, Starred, Dashboards (selected), Explore, Alerting, Connections, and Administration. The main area is titled 'Import dashboard' with the subtitle 'Import dashboard from file or Grafana.com'. It features an 'Upload dashboard JSON file' section with a dashed box and instructions to 'Drag and drop here or click to browse', noting 'Accepted file types: .json, .txt'. Below this is a link to 'Find and import dashboards for common applications at grafana.com/dashboards' and a search input containing '13659' with a 'Load' button. At the bottom, the 'Import via dashboard JSON model' section is visible, with a red arrow pointing to its header. A modal window titled 'Importing dashboard from Grafana.com' is open on the right. It displays metadata for dashboard '13659': 'Published by harloprillar' and 'Updated on 2020-12-31 15:47:50'. The 'Options' section includes a 'Name' field with 'Blackbox Exporter (HTTP prober)', a 'Folder' dropdown set to 'Dashboards', and a 'Unique identifier (UID)' field with 'NEzutrbMk' and a 'Change uid' button. The 'Prometheus' section shows a dropdown with 'Prometheus' selected. At the bottom of the modal are 'Import' and 'Cancel' buttons, with two red arrows pointing to them.

Home > Dashboards > Import dashboard

Home  
Bookmarks  
Starred  
Dashboards  
Explore  
Alerting  
Connections  
Administration

## Import dashboard

Import dashboard from file or Grafana.com

Upload dashboard JSON file

Drag and drop here or click to browse  
Accepted file types: .json, .txt

Find and import dashboards for common applications at [grafana.com/dashboards](https://grafana.com/dashboards)

13659 **Load**

Import via dashboard JSON model

### Importing dashboard from Grafana.com

Published by harloprillar

Updated on 2020-12-31 15:47:50

### Options

Name  
Blackbox Exporter (HTTP prober)

Folder  
Dashboards

Unique identifier (UID)  
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.  
NEzutrbMk **Change uid**

Prometheus  
Prometheus

**Import** Cancel

# Создаем дашборд

- (например импортируя 13659 с сайта графаны)

The image shows the Grafana 'Import dashboard' interface. On the left is a sidebar with navigation links: Home, Bookmarks, Starred, Dashboards (selected), Explore, Alerting, Connections, and Administration. The main panel is titled 'Import dashboard' with the subtitle 'Import dashboard from file or Grafana.com'. It features an 'Upload dashboard JSON file' section with a dashed box for file upload and instructions: 'Drag and drop here or click to browse' and 'Accepted file types: .json, .txt'. Below this is a link to 'Find and import dashboards for common applications at grafana.com/dashboards' and a search input containing '13659' with a 'Load' button. At the bottom of the main panel is the 'Import via dashboard JSON model' section, which is highlighted by a red arrow. To the right, a modal titled 'Importing dashboard from Grafana.com' is open. It displays metadata for dashboard '13659': 'Published by harloprillar' and 'Updated on 2020-12-31 15:47:50'. The 'Options' section includes a 'Name' field with 'Blackbox Exporter (HTTP prober)', a 'Folder' dropdown set to 'Dashboards', and a 'Unique identifier (UID)' field with 'NEzutrbMk' and a 'Change uid' button. At the bottom of the modal, the 'Prometheus' data source is selected from a dropdown, and there are 'Import' and 'Cancel' buttons. Two red arrows point to these buttons.

Home > Dashboards > Import dashboard

Home  
Bookmarks  
Starred  
Dashboards  
Explore  
Alerting  
Connections  
Administration

## Import dashboard

Import dashboard from file or Grafana.com

Upload dashboard JSON file

Drag and drop here or click to browse  
Accepted file types: .json, .txt

Find and import dashboards for common applications at [grafana.com/dashboards](https://grafana.com/dashboards)

13659 **Load**

Import via dashboard JSON model

### Importing dashboard from Grafana.com

Published by harloprillar

Updated on 2020-12-31 15:47:50

### Options

Name  
Blackbox Exporter (HTTP prober)

Folder  
Dashboards

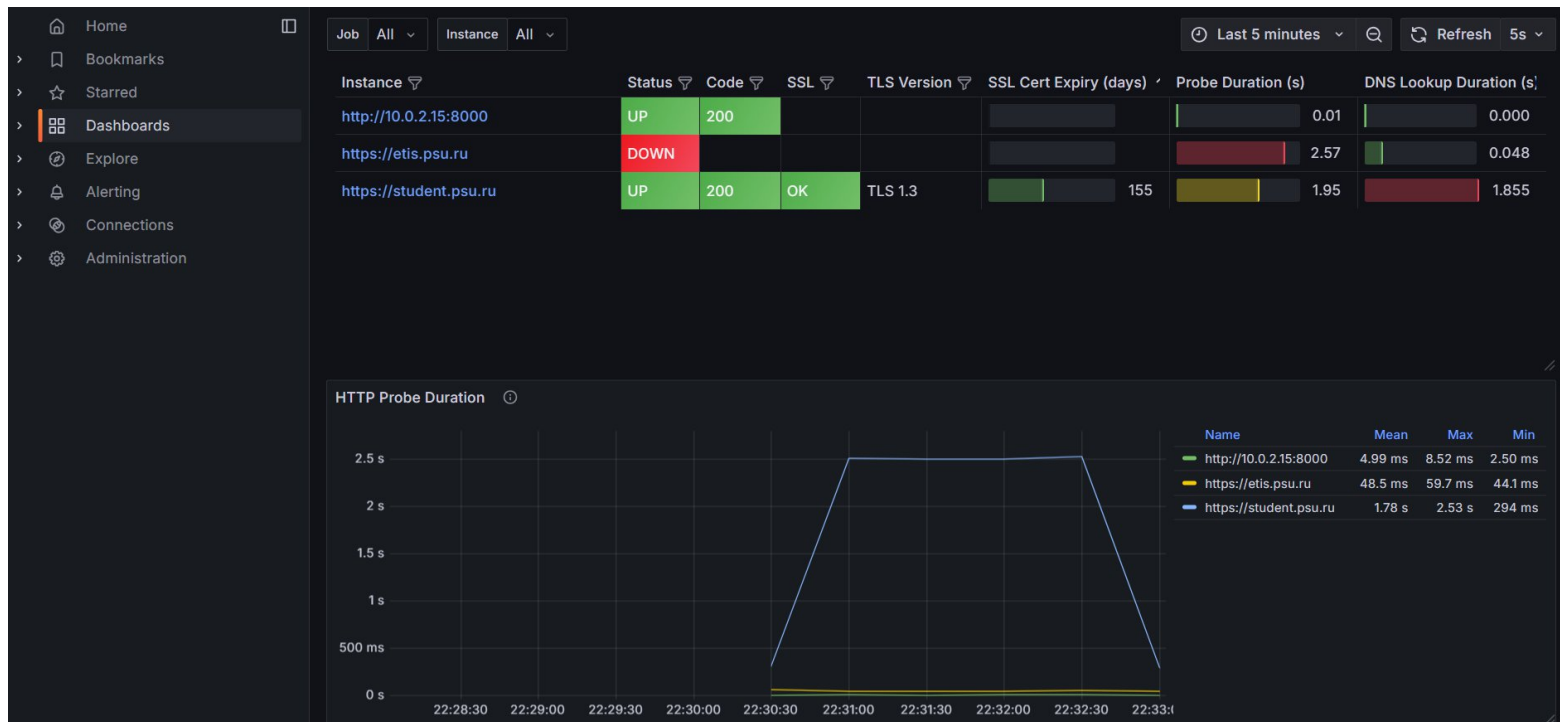
Unique identifier (UID)  
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.  
NEzutrbMk **Change uid**

Prometheus  
Prometheus

**Import** Cancel



# Проверяем отображение метрик на дашборде



Hello World! I have been seen 73 times.