

Виртуализация: **Docker ч.2**



Артур

Сагутдинов



Артур Сагутдинов

Инженер департамента
голосовых цифровых
технологий

Banks Soft Systems



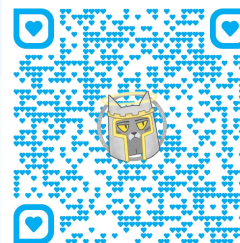
15+ лет в сфере ИТ



Разрабатываю и внедряю
линуксовую инфраструктуру



[Сисадминский блог](#)

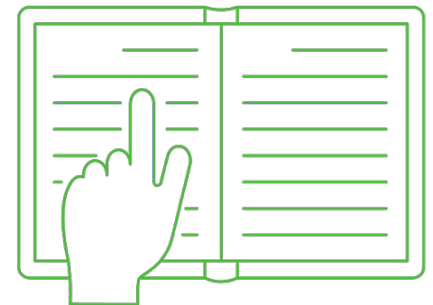


https://t.me/belf_igor

Предисловие


На этом занятии мы:

- познакомимся с Docker Compose и узнаем, что это такое;
- установим Docker Compose на Debian;
- поговорим о создании и наполнении docker-compose.yml;
- запустим одиночный контейнер с помощью Docker;
- запустим многоконтейнерное приложение с помощью Docker Compose.



План занятия

1. [Инфраструктура с помощью Docker](#)
2. [Мониторинг](#)
3. [Многоконтейнерный ад](#)
4. [Docker Compose](#)
5. [Практика](#)
6. [Итоги](#)
7. [Домашнее задание](#)



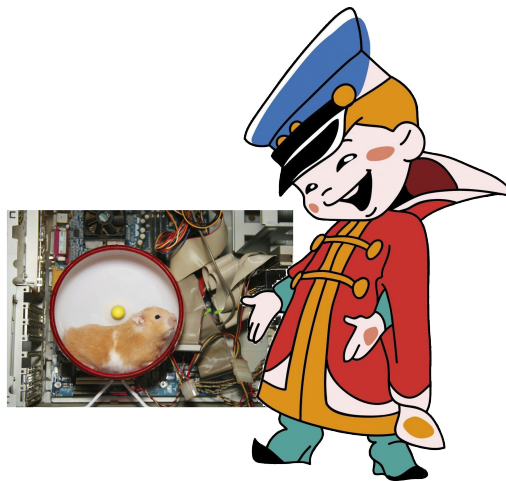
Инфраструктура с помощью Docker

Docker как основа гибкой инфраструктуры

Одна из самых типичных ситуаций, когда у сисадмина есть небольшая корпоративная сеть. Всё там, вроде бы, работает. И было бы неплохо сделать что-то, чтобы предсказать и предотвратить ряд проблем.

НО зачем?

И так ведь работает.





Docker как основа гибкой инфраструктуры

Потом сеть начинает расширяться, инфраструктура усложняется, и «неподъёмный и сложный докер» перестаёт казаться таким пугающим.

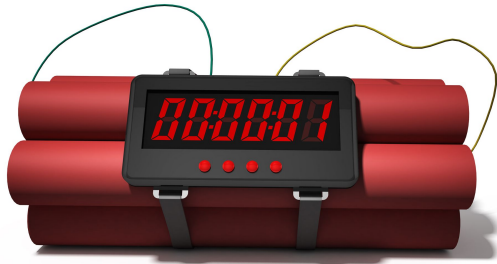
О том, как решить часть своих проблем за счёт использования готовых контейнеров, мы сегодня и поговорим.



Мониторинг

Мониторинг

Одним из краеугольных камней инфраструктуры является телеметрия. Если вы знаете, что какой-то показатель приблизился к своему максимальному значению (например, кончается место на диске сервера), вы можете предотвратить катастрофу в последний момент, не допустив глобального обрушения всей инфраструктуры. Помочь вам с этим может Zabbix.



Вкратце о Zabbix

Zabbix — это ультимативная клиент-серверная система мониторинга, решающая все проблемы, лежащие в плоскости мониторинга инфраструктуры предприятия. Она не подходит для обработки телеметрии в реальном времени, но во всём остальном она бескомпромиссно превосходит все свои аналоги. И самое главное, она бесплатная и расширяемая.



Zabbix с помощью Docker

Для решения данной задачи воспользуемся Docker Hub. Открываем браузер, переходим на Docker Hub, вбиваем в поиск Zabbix Appliance и попадаем на страницу.

<https://hub.docker.com/r/zabbix/zabbix-appliance>

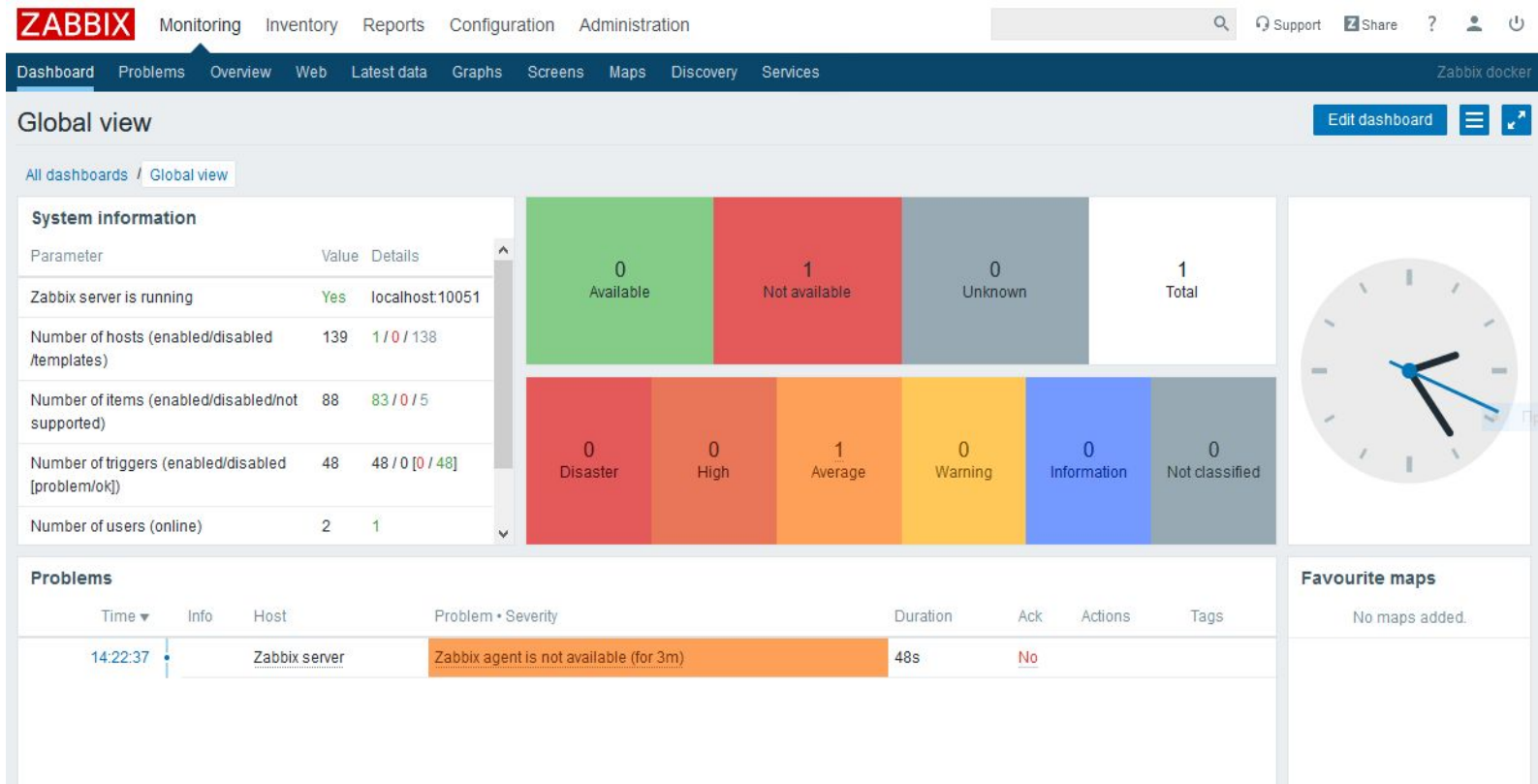
На странице описывается, как установить готовое решение Zabbix. Для этого достаточно воспользоваться следующей командой:

```
docker run --name zabbix-appliance -p 80:80 -p 10051:10051 -d  
zabbix/zabbix-appliance:latest
```

Параметр -p означает, какой порт Docker сервера мы пробрасываем, на какой порт Docker контейнера.

Проверяем, что контейнер работает

Подключаемся на IP-адрес докер сервера на порт 80.
Стандартный логин и пароль Admin \ zabbix.



Узнаём IP-адрес Docker контейнера

Для получения IP-адреса контейнера, необходимо узнать его ID или имя и применить команду `inspect`

```
sudo docker container inspect zabbix-appliance
```

Находим раздел **Networks**, подраздел **bridge** и в нём ищем строку `IPAddress`. Там и будет IP-адрес нашего контейнера

```
"Networks": {
  "bridge": {
    "IPAMConfig": null,
    "Links": null,
    "Aliases": null,
    "NetworkID": "feffd925f1dd9fa87011b0a35e4df07a20b3983765be212960915a7abfbb4c5a",
    "EndpointID": "2341041aaa0622df95ce53497266250e72accb88def793a84da077da5354cd21",
    "Gateway": "172.17.0.1",
    "IPAddress": "172.17.0.2",
    "IPPrefixLen": 16,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "MacAddress": "02:42:ac:11:00:02",
    "DriverOpts": null
  }
}
```

Нацеливаем Zabbix Agent на Zabbix Appliance

Открываем `zabbix_agentd.conf` и вносим в параметр `Server` IP адрес нашего Zabbix Appliance:

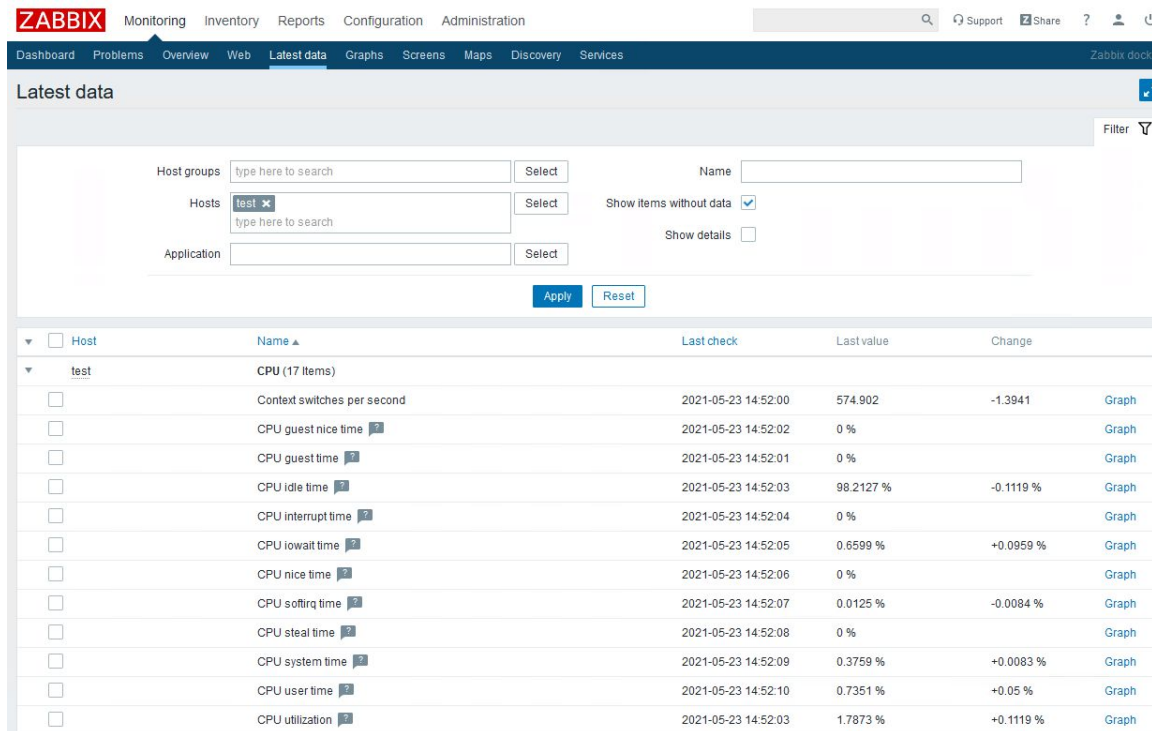
```
sudo nano /etc/zabbix/zabbix_agentd.conf
```

Перезапускаем Zabbix Agent, чтобы параметры вступили в силу:

```
sudo service zabbix-agent restart
```

Проверяем информацию в Zabbix

Добавляем Docker сервер в Zabbix, и видим, что пошла телеметрия.



The screenshot shows the Zabbix web interface. The top navigation bar includes links for Monitoring, Inventory, Reports, Configuration, and Administration. The main menu on the left has options like Dashboard, Problems, Overview, Web, Latest data (selected), Graphs, Screens, Maps, Discovery, and Services. The 'Latest data' page displays a table of monitoring data for a host named 'test'.

Host groups: type here to search [Select] Name: []

Hosts: test [Select] Show items without data: [checked] Show details: []

Application: [Select] [Apply] [Reset]

Host	Name	Last check	Last value	Change
test	CPU (17 items)			
	Context switches per second	2021-05-23 14:52:00	574.902	-1.3941
	CPU guest nice time	2021-05-23 14:52:02	0 %	
	CPU guest time	2021-05-23 14:52:01	0 %	
	CPU idle time	2021-05-23 14:52:03	98.2127 %	-0.1119 %
	CPU interrupt time	2021-05-23 14:52:04	0 %	
	CPU iowait time	2021-05-23 14:52:05	0.6599 %	+0.0959 %
	CPU nice time	2021-05-23 14:52:06	0 %	
	CPU softirq time	2021-05-23 14:52:07	0.0125 %	-0.0084 %
	CPU steal time	2021-05-23 14:52:08	0 %	
	CPU system time	2021-05-23 14:52:09	0.3759 %	+0.0083 %
	CPU user time	2021-05-23 14:52:10	0.7351 %	+0.05 %
	CPU utilization	2021-05-23 14:52:03	1.7873 %	+0.1119 %

Что вообще произошло?

Мы за несколько действий с помощью Docker развернули энтерпрайз телеметрию из контейнера. Теперь мы можем подключить к ней несколько десятков компьютеров и получать с них данные.



Но что дальше?



Многоконтейнерный «ад»



Что делать, если контейнеров больше 1-го?

Ситуация, когда вы вводите одну команду, запускаете один контейнер и ваша проблема решается — недостижимый идеал. Если у вас на самом деле так получилось, возможно, вы переросли своё текущее рабочее место.

На практике контейнеров часто больше одного. Случаются ситуации, когда продакшн состоит, например, из десятка серверов с уникальной нагрузкой, и на каждом сервере бежит несколько десятков контейнеров каждый со своей нагрузкой.



Docker Compose

Docker Compose. Жонглируем контейнерами

В ситуации с многоконтейнерными приложениями на помощь приходит Docker Compose. С его помощью можно «подружить» огромное количество контейнеров и избежать неопиcуемый неконтролируемый хаос, с которым сталкивались праотцы на заре интернетов.



ИСТОЧНИК

Многоконтейнерное приложение

Zabbix Appliance поставляется в виде одного контейнера и работает независимо. Он сильно ограничен в производительности и использует встроенный веб-сервер и БД. Как пример многоконтейнерного приложения, мы будем рассматривать тот же Zabbix, но состоящий из следующих элементов:

- **PostgreSQL** — СУБД в котором будут храниться данные,
- **pgAdmin** — средство администрирования PostgreSQL,
- **Zabbix Server** — сервер телеметрии, который будет собирать информацию от агентов и складывать в СУБД,
- **Zabbix Frontend** — Вэбгуй, через который администрируется Zabbix Server.



Практика

Установка Docker Compose

1. Скачиваем последний стабильный релиз из репозитория:

```
sudo curl -L  
"https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname  
-s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

2. Устанавливаем права на запуск:

```
sudo chmod +x /usr/local/bin/docker-compose
```

3. Проверяем, что всё работает:

```
docker-compose --version
```

Основа docker-compose.yml

Опишем основу нашего сценария docker-compose.yml

```
version: "3"
services:

networks:
  netology-lesson:
    driver: bridge
  ipam:
    config:
      - subnet: 172.19.0.0/24
```

- **version** определяет поддерживаемую версию движка и можно ли данный сценарий запустить на текущем движке;
- **network** задаём статическую подсеть, к которой привяжем некоторые свои контейнеры, чтобы они не меняли адрес от запуска к запуску.

Поднимаем PostgreSQL и PGAdmin одним кликом

Как известно, лень — двигатель прогресса. А сисадмин — это квинтэссенция лени. Мы готовы многое сделать, чтобы не делать ничего. Ещё одним нашим шагом на пути к прогрессу, будет победа над постгресом.

- **PostgreSQL** — СУБД из коробки и «за бесплатно», обладающая многими полезными плюшками, которые встречаются в том же MS SQL только в сборках за десятки тысяч \$;
- **pgAdmin** — утилита для администрирования PostgreSQL.

PostgreSQL станет первым кирпичиком нашего многоконтейнерного приложения в виде Zabbix.

Разворачиваем PostgreSQL

Первый сервис в нашем сценарии будет **Сервер Управления Базами Данных**. Добавим в файл **docker-compose.yml** в раздел **services**, новый сервис, который назовём **netology-db** (описывает сценарий развёртывания контейнера PostgreSQL.):

```
netology-db:
  image: postgres:latest # Образ, который мы будем использовать
  container_name: netology-db # Имя, которым будет называться наш контейнер
  ports: # Порты, которые мы пробрасываем с нашего докер сервера внутрь контейнера
    - 5432:5432
  volumes: # Папка, которую мы пробросим с докер сервера внутрь контейнера
    - ./pg_data:/var/lib/postgresql/data/pgdata
  environment: # Переменные среды
    POSTGRES_PASSWORD: 123 # Задаём пароль от пользователя postgres
    POSTGRES_DB: netology_db # БД которая сразу же будет создана
    PGDATA: /var/lib/postgresql/data/pgdata # Путь внутри контейнера, где будет папка
pgdata
  networks:
    netology-lesson:
      ipv4_address: 172.19.0.2
  restart: always # Режим перезапуска контейнера. Контейнер всегда будет
перезапускаться
```

Описание раздела **netology-db**



- Мы используем образ **postgres** с тэгом **latest**;
- Контейнер будет называться **netology-db**;
- При обращении на IP-адрес докер сервера на порт **5432**, он будет прокинут внутрь контейнера **netology-db** на порт **5432**;
- Локальная папка на докер сервере **./pg_data** прокидывается в контейнер **netology-db** в папку **/var/lib/postgresql/data/pgdata**;

Описание раздела **netology-db**



- С помощью переменных среды, поддерживаемых контейнером **netology-db**, мы задаём пароль пользователя **postgres**, имя БД и папку в которой будет размещена директория **pgdata**;
- Также контейнер запускается с параметром постоянного перезапуска в случае падения.

Пробный запуск сценария

1. Теперь у нас есть сценарий, запускающий **PostgreSQL**. Стоит попробовать его запустить.

```
sudo docker-compose up #запуск с выводом в консоль
```

```
sudo docker-compose up -d #запуск в бэграунде
```

Пробный запуск сценария

2. Проверить работоспособность можно проверив порт 5432, являющийся стандартным для PostgreSQL

```
telnet localhost 5432 #Проверяем подключаясь на порт телнетом  
netstat -nlp | grep 5432 #Проверяем с помощью netstat. Если не стоит то apt install net-tools  
sudo docker ps # Ищем контейнер и его порт
```



Пробный запуск сценария

3. Также мы можем подключиться к любому контейнеру в консоль с помощью команды

```
sudo docker exec -it <имя_контейнера_или_id> bash
```

Добавляем сценарий для pgAdmin

После того как у нас взлетел постгрес, пора приделать к сценарию средство, которое позволит нам его радостно администрировать. Потому добавим раздел **pgadmin**:

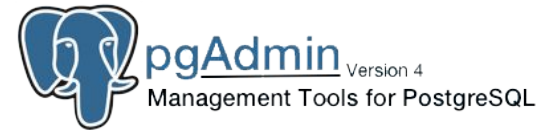
```
pgadmin:
  image: dpage/pgadmin4
  container_name: netology-pgadmin
  environment:
    PGADMIN_DEFAULT_EMAIL: netology@mymail.me
    PGADMIN_DEFAULT_PASSWORD: 123
  ports:
    - "8080:80"
  networks:
    netology-lesson:
      ipv4_address: 172.19.0.3
  restart: always
```


Описание раздела pgadmin



- Мы используем образ **dpage/pgadmin4** без тэга, что автоматом означает тэг **latest**;
- Контейнер будет называться **netology-pgadmin**;
- С помощью переменных среды поддерживаемых контейнером **dpage/pgadmin4**, мы задаём почту администратора, как `docker@mymail.me` и пароль администратора `123`;

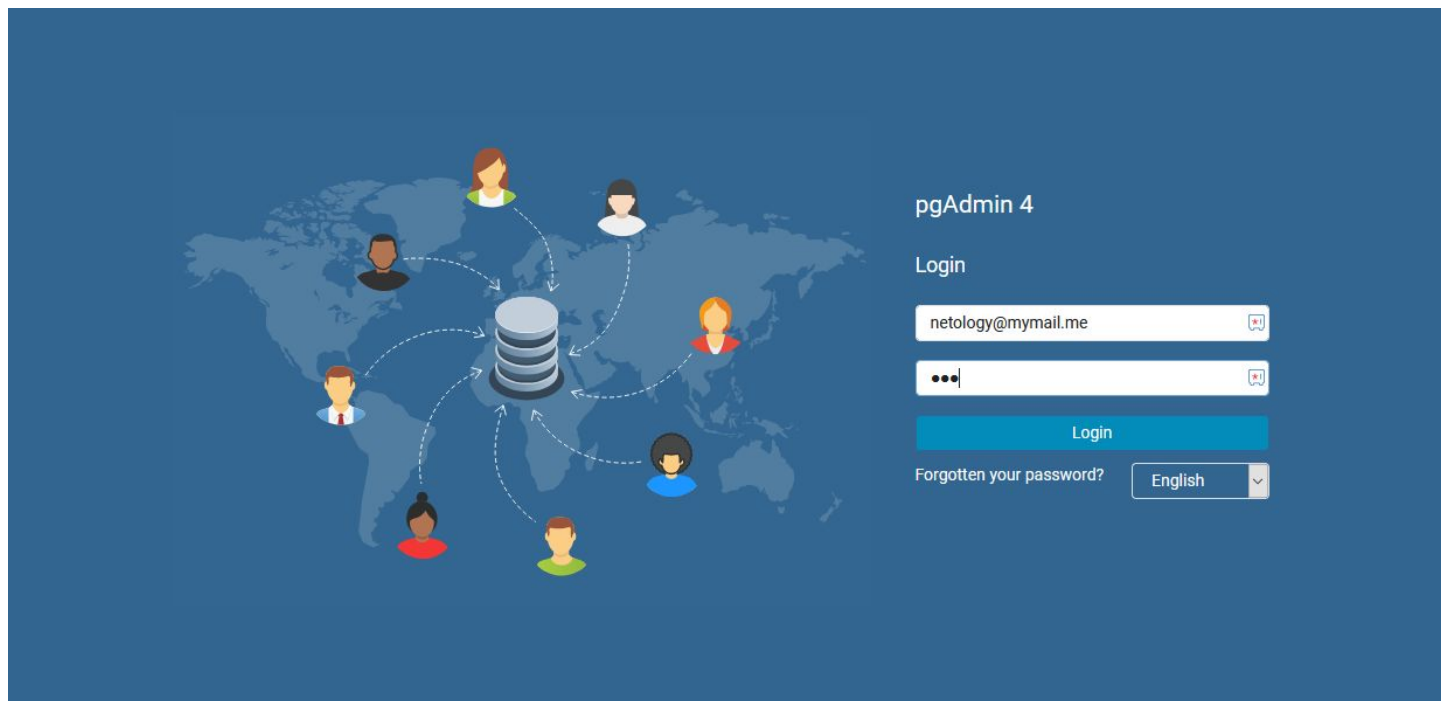
Описание раздела pgadmin



- При обращении на IP-адрес докер сервера на порт **8080**, он будет прокинут внутрь контейнера **netology-pgadmin** на порт **80**;
- Так же контейнер запускается с параметром постоянного перезапуска в случае падения.

Пробный запуск сценария

Снова запускаем сценарий, пытаемся подключиться на IP-адрес нашего Docker сервера на порт 8080, который прокинут на 80й порт pgAdmin. Видим логинскрин, на котором можем авторизоваться под логином netology@mymail.me с паролем 123



Добавляем Zabbix Server

Дальше к нашей мини инфраструктуре добавим Zabbix Server и настроим сценарий так, чтобы агент мониторинга с нашего сервера смог к нему подключиться.

```
zabbix-server:
  image: zabbix/zabbix-server-pgsql
  links:
    - netology-db
  container_name: netology-zabbix
  environment:
    DB_SERVER_HOST: '172.19.0.2'
    POSTGRES_USER: postgres
    POSTGRES_PASSWORD: 123
  ports:
    - "10051:10051"
  networks:
    netology-lesson:
      ipv4_address: 172.19.0.4
  restart: always
```

Описание раздела **zabbix-server**

- Мы используем образ **zabbix/zabbix-server-pgsql** без тэга, что автоматом означает тэг **latest**. Также нужно отметить, что этот контейнер собран для работы с PostgreSQL;
- Контейнер будет называться **netology-zabbix**;
- С помощью переменных среды, поддерживаемых контейнером, мы передаём в контейнер IP-адрес сервера PostgreSQL, имя пользователя PostgreSQL, и его пароль;

Описание раздела zabbix-server

- При обращении на IP адрес докер сервера на порт **10051**, он будет прокинут внутрь контейнера на порт **10051**;
- Мы задаём контейнеру статический IP из сети 172.19.0.0/24 описанной нами ранее;
- Контейнер запускается с параметром постоянного перезапуска в случае падения.

Добавляем Zabbix Frontend

Установить сервер, к которому могут подключаться и отправлять телеметрию агенты, мало, необходимо им управлять. Для этого нам нужен фронтенд.

```
zabbix_wgui:
  image: zabbix/zabbix-web-apache-pgsql
  links:
    - netology-db
    - zabbix-server
  container_name: netology_zabbix_wgui
  environment:
    DB_SERVER_HOST: '172.19.0.2'
    POSTGRES_USER: 'postgres'
    POSTGRES_PASSWORD: 123
    ZBX_SERVER_HOST: "zabbix_wgui"
    PHP_TZ: "Europe/Moscow"
  ports:
    - "80:8080"
    - "443:8443"
  networks:
    netology-lesson:
      ipv4_address: 172.19.0.5
  restart: always
```

Описание раздела **zabbix-wgui**

- Мы используем образ **zabbix/zabbix-web-apache-pgsql** без тэга, что автоматом означает тэг **latest**. Также нужно отметить, что этот контейнер собран для работы с PostgreSQL и как веб-сервер использует **apache**;
- Контейнер будет называться **zabbix-wgui**;
- С помощью переменных среды, поддерживаемых контейнером, мы передаём в контейнер IP-адрес сервера PostgreSQL, имя пользователя PostgreSQL, и его пароль. Задаём имя заббикс сервера и таймзону;

Описание раздела `zabbix-wgui`

- При обращении на IP-адрес докер сервера на порт **1081**, он будет прокинут внутрь контейнера на порт **8080**;
- Контейнер запускается с параметром постоянного перезапуска в случае падения.



Итоги

Итоги

Сегодня мы более плотно поработали с Docker и узнали как:

- установить Docker Compose на Debian;
- с помощью Docker запустить Zabbix Appliance;
- работать с файлом docker-compose.yml;
- с помощью Docker Compose запустить многоконтейнерный сервис.





Домашнее задание



Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Артур Сагутдинов



https://t.me/belf_igor

