

7 июня 2023 📍 Москва, МЦК ЗИЛ

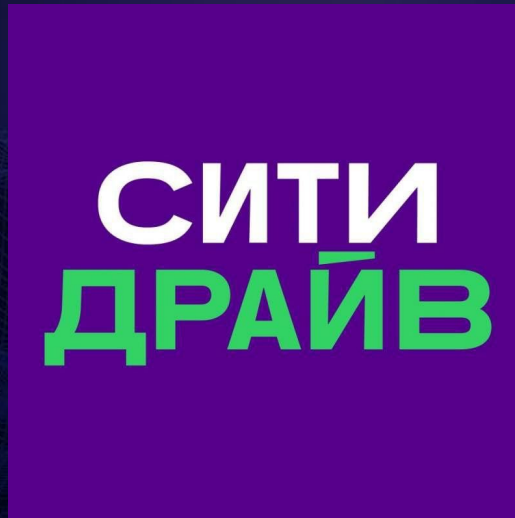
БЕКОН²³

Первая в России конференция
по БЕзопасности КОНтейнеров и контейнерных сред

Контейнерная ОС Flatcar: как обмазаться докерами и забыть про пакеты



- 10 лет стажа в DevOps
- 5 лет стажа преподавания
- Автор канала “Админим с Буквой”
- Статьи на [Habr](#)
- 13 проведенных CTF
- Лекции на [YouTube](#)

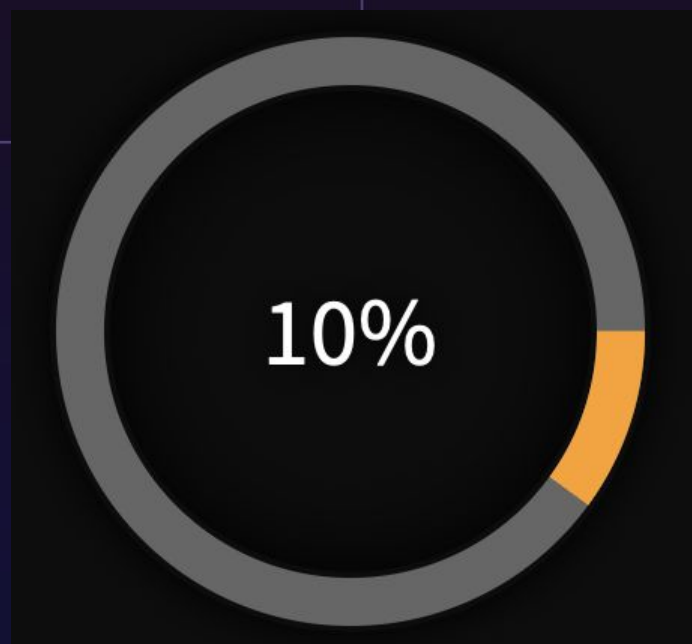


О чем пойдет речь в докладе

БЕКОН

- 1) Что такое контейнерная ОС и какие бывают
- 2) Какие бенефиты имеет Flatcar для Отдела Обеспечения Безопасности
- 3) Как внедрять Flatcar в инфраструктуру
- 4) Как разрабатывать конфиги первичной настройки ОС
- 5) Как деплоить Flatcar в инфраструктуру
- 6) Как обновлять Flatcar
- 7) Как запускать приложения в Flatcar

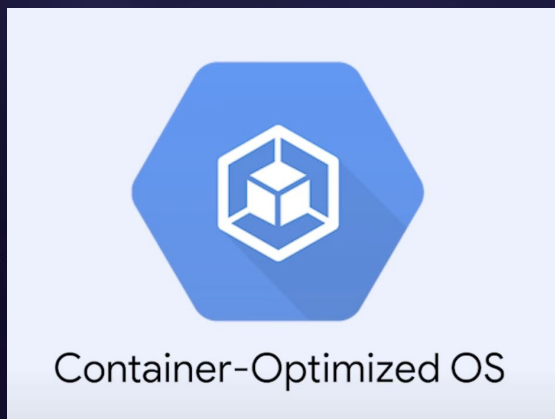
Что такое контейнерная ОС



- Созданы для автоматизации, а не для человека
 - Отсутствие привычных доступов
 - Нельзя поставить софт, может быть без shell
 - Уменьшение configuration drift
- Поддержка оптимальной работы с контейнерами из коробки
- Уменьшенный размер
 - Высокая скорость развёртывания
 - Убрано всё лишнее
- Повышенные требования от ИБ
 - Наличие атомарных и частых обновлений
 - Read-only файловая система (где как)

Примеры контейнерных ОС

БЕКОН



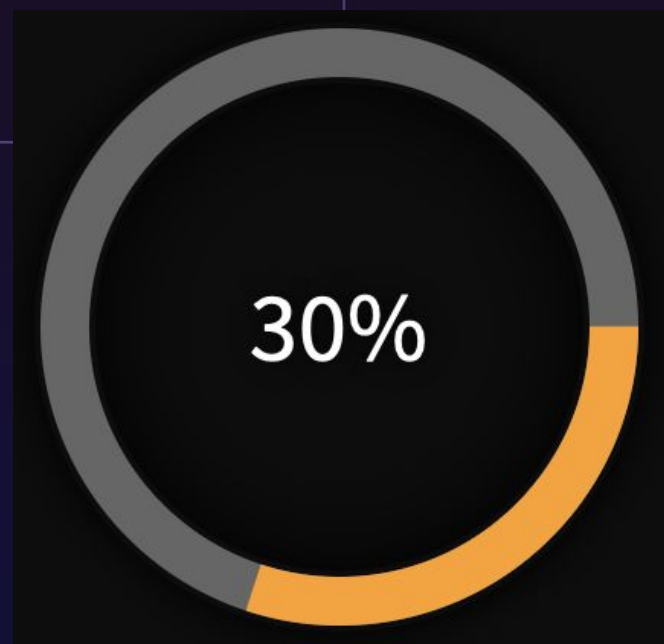
A community Linux distribution designed for container workloads, with high security and low maintenance



- Минимальный образ ОС включает только утилиты для запуска контейнеров
- Никаких менеджеров пакетов и проблем с консистентностью конфигураций
- Автоматизированные атомарные обновления позволяют получать в кратчайшие сроки последние обновления с возможностью откатиться назад:

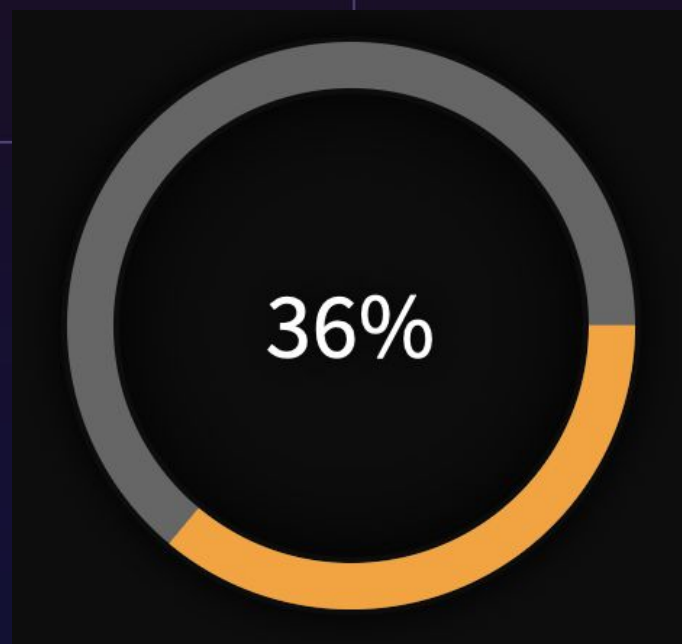
3 канала обновлений для максимально комфортного ощущения защищенности и стабильности
- Встроенный ignition
- Read-only для системных разделов, но Read-Write для мест, куда можно положить конфиги, изменяемые данные или portable исполняемые файлы

Бенефиты Flatcar для ИБ



- 1) опции nodev, noexec, nosuid, запись в многие разделы запрещена
- 2) настройка ssh, pam, profile.d, login.defs, sysctl - как обычно
- 3) отсутствуют лишние сервисы - уменьшенная поверхность для атаки
- 4) auditd, selinux - есть и можно настроить
- 5) *syslog*, *beat* - в docker
- 6) атомарные обновления

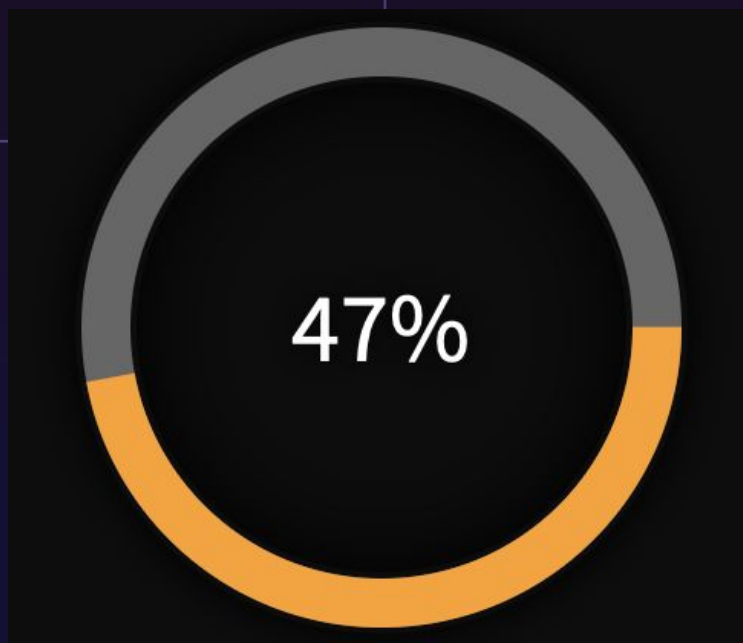
Как внедрять Flatcar



1. Анализ площадки, где будем размещать VM
 - a. какой гипервизор/облако используется
 - b. какую автоматизацию установки VM мы можем себе позволить (включая подготовку и считывание ignition/cloud-init)
 - c. как будем обновлять базовый образ
2. Какие приложения планируется заселять
 - a. docker, containerd
 - b. portable (пуру, например, или go/c binary)
3. Кто и как будет обслуживать ОС
 - a. обновление ОС (автоматикой или ручками)
 - b. изменение ресурсов (автоматикой или ручками, с ребутом или без)
 - c. мониторинг (node exporter in docker)
4. Аудит соответствия Flatcar вашим манифестам и требованиям безопасности
 - a. доработка стокового образа вашим требованиям настройки ОС
 - b. разработка способов установки системных агентов (логи, EDR, ...)

1. Время, затрачиваемое сотрудниками на изучение новой темы
2. Привыкание к новой парадигме
3. Отсутствие некоторых debug-утилит и интерпретируемых языков, например, python
4. Нельзя просто так взять и поменять некоторые конфиги привычным путём

Как писать Ignition конфиги



Для написания ignition конфига используется промежуточная утилита butane

Как выглядит процесс:

YAML -> butane -> JSON

Почему 2 этапа?

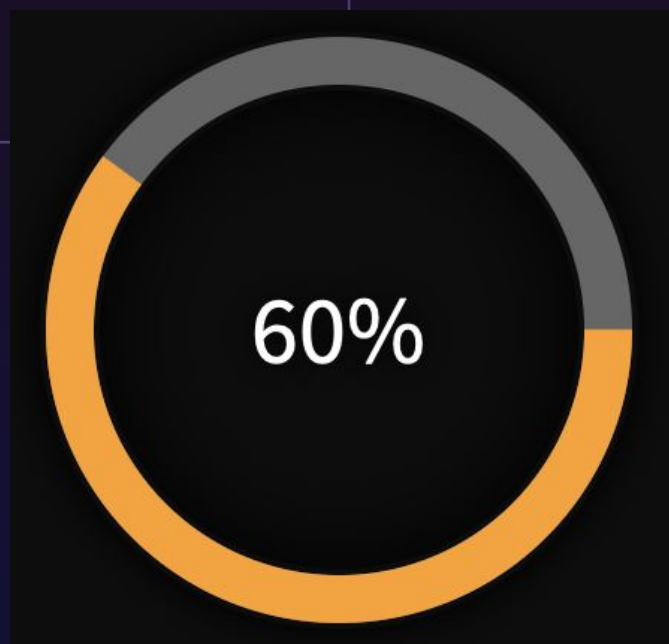
- 1) писать на YAML проще
- 2) дополнительное валидирование вашего YAML до того как он попадет в run-time
- 3) JSON проще обрабатывать при разворачивании VM
- 4) в результирующем документе некоторые значения параметров будут в формате "url encode"

```
"passwd": {  
  "users": [  
    {  
      "name": "core",  
      "passwordHash": "{{ esxi_vm_core_password_hash }}",  
      "sshAuthorizedKeys": [  
        "{{ esxi_vm_core_ssh_key }}"  
      ]  
    }  
  ]  
},
```

Пример настройки системных конфигов

```
"files": [  
  {  
    "filesystem": "root",  
    "path": "/etc/hostname",  
    "contents": {  
      "source": "data:,{ { esxi_vm_name } }%0A",  
      "verification": {}  
    },  
    "mode": 420  
  },  
  {  
    "filesystem": "OEM",  
    "path": "/grub.cfg",  
    "contents": {  
      "source": "data:,set%20linux_append%3D%22net.ifnames%3D0%20%22%0A",  
      "verification": {}  
    },  
    "mode": 420  
  },  
]
```


Как деплоить Flatcar



<https://stable.release.flatcar-linux.net/amd64-usr/3510.2.1/>

- AWS
- QEMU
- OpenStack
- DigitalOcean
- Exoscale
- CloudStack
- GCE
- Parallels
- VmWare
- Virtualbox
- Xen
- Nifty Cloud



Есть автоустановщик для накатывания с live-образа, AMD и ARM билды

Чем деплоить на сервера

- 1) ISO
- 2) готовый собранный образ - загрузить шаблон и разворачивать из него
- 3) скрипт flatcar-install для live-установок
- 4) PXE / iPXE

Все это можно автоматизировать по-разному, самый простой способ -

- 1) загрузить готовый билд под нужный гипервизор
- 2) создать шаблон виртуальной машины из билда
- 3) разворачивать из него 100500 VM с помощью Ansible или Terraform

Как разворачивать flatcar на примере vmware

- Загружаем готовый vmdk для vmware
- Берем в руки Ansible (можно и terraform, не важно)
- Пишем таск, который развернет виртуалку:

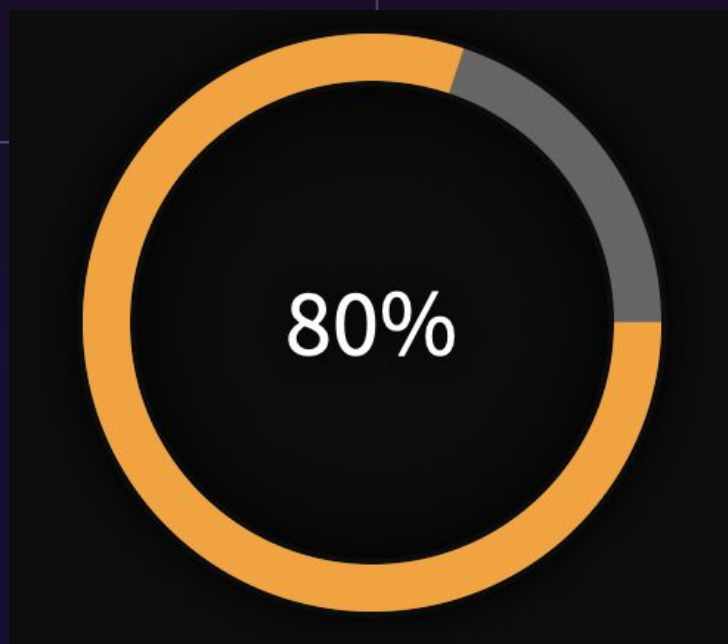
```
- name: create virtual machine
  community.vmware.vmware_guest:
    hostname: "{{ vcenter_hostname }}"
    username: "{{ vcenter_username }}"
    password: "{{ vcenter_password }}"
    template: "{{ esxi_vm_template }}"
    datacenter: "{{ esxi_vm_datacenter }}"
    cluster: "{{ esxi_vm_cluster }}"
    name: "{{ esxi_vm_name }}"
```

```
- name: update vm options
  environment:
    GOVC_URL: "{{ lookup('env', esxi_vm_datacenter+'_GOVC_URL') }}"
  when: first_time.stdout == "" and esxi_vm_state != "absent"
  command: "govc vm.change -vm {{ esxi_vm_folder }}/{{ esxi_vm_name }}
            -e=guestinfo.ignition.config.data.encoding=base64
            -e=guestinfo.ignition.config.data={{ ignition }}"
  delegate_to: localhost
  connection: local
  become: False
```


Пример деплоя для sbercloud и terraform

```
resource "sbercloud_compute_instance" "vm" {
  count          = var.instance_count
  name           = "${var.vm_prefix}${var.vm_name}-${count.index+1}.${var.env}.${var.domain}"
  user_data      = "#cloud-config\nhostname: ${var.vm_prefix}${var.vm_name}-${count.index+1}.${var.env}.${var.domain}"
  image_id       = data.sbercloud_images_image.flatcar.id
  flavor_id      = data.sbercloud_compute_flavors.myflavor.ids[0]
  security_groups = var.security_groups
  availability_zone = data.sbercloud_availability_zones.myaz.names[0]
  system_disk_type = var.system_disk_type
  system_disk_size = var.system_disk_size
  key_pair       = var.key_pair
  tags           = var.tags
}
```


Как обновлять Flatcar

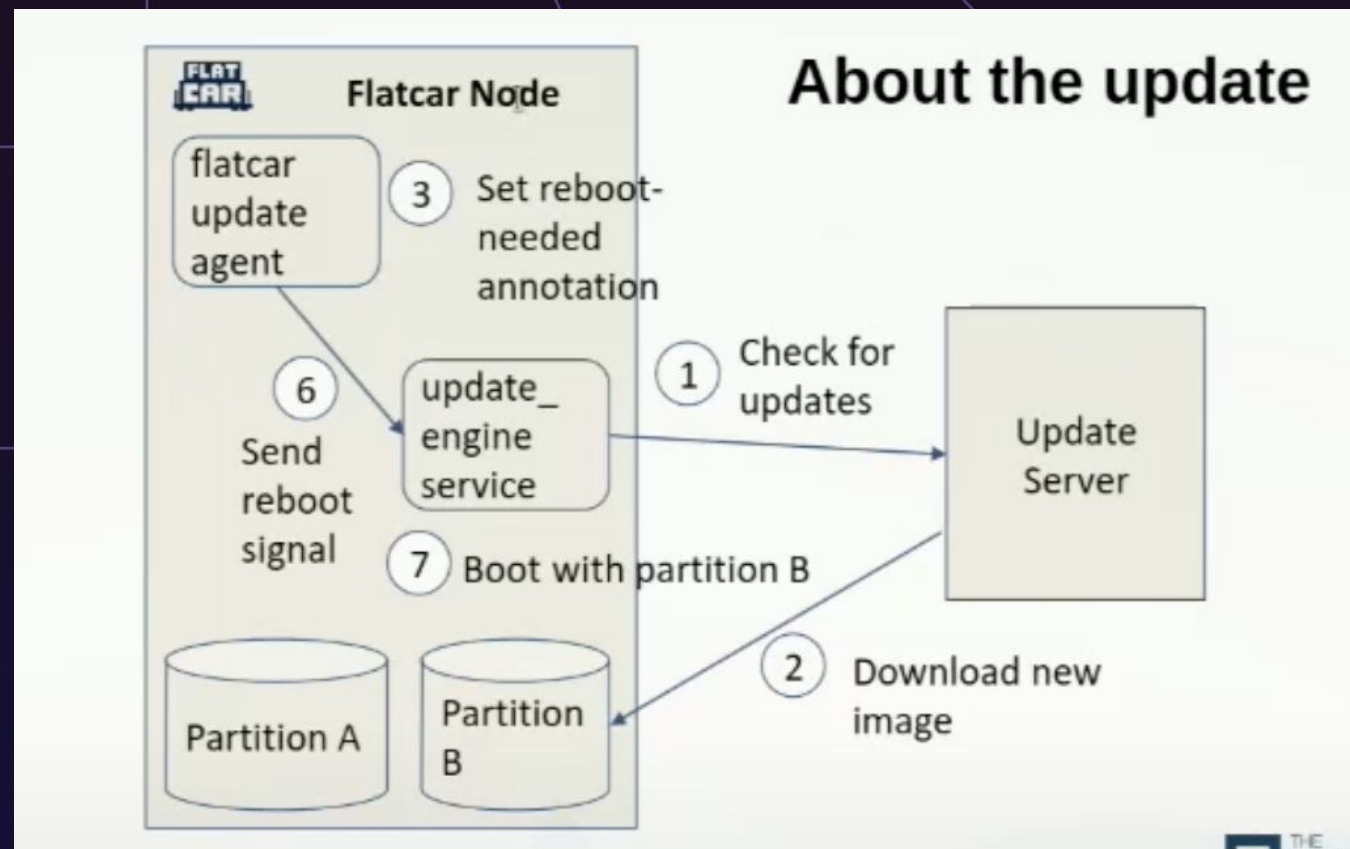


Status check

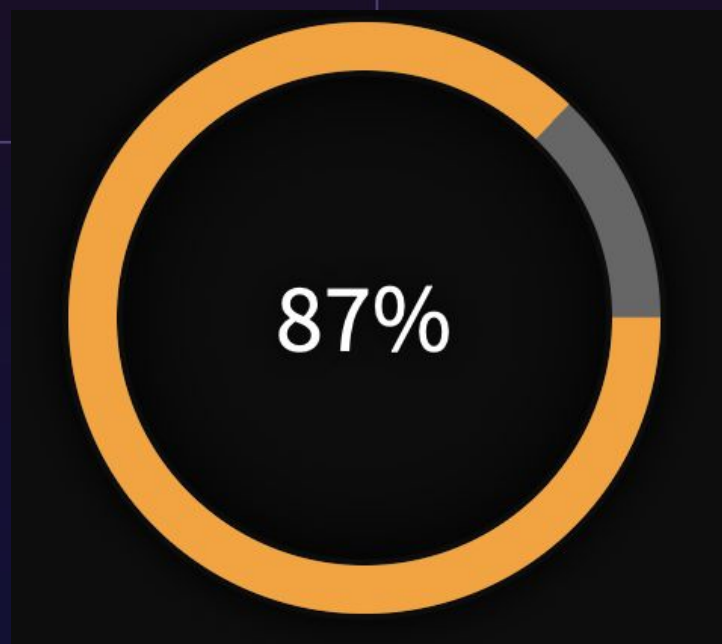
```
# manual release check here: https://kinvolk.io/flatcar-container-linux/releases/  
cat /etc/os-release  
update_engine_client --status
```

Force update to latest release in channel

```
update_engine_client --reset_status  
update_engine_client --check_for_update  
sleep 30  
update_engine_client --update
```



Как запускать приложения



Как запускать приложения

Пример запуска docker с grafana

- 1) Все что запускаем - описываем в systemd unit файлах
- 2) Используем всякие полезные Pre-Post Exec
- 3) Удобная шаблонизация имени контейнера с %N = имя сервиса
- 4) Контроль перезапуска - как, когда, как часто, итп.
- 5) Остальные 1000 параметров systemd

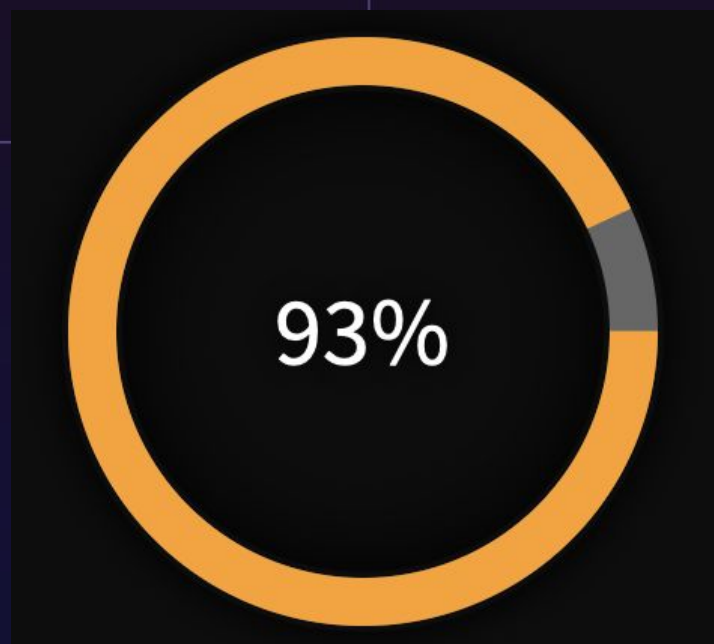
если сложно - воспользуйтесь
[генератором](#)

```
[Unit]
Requires=docker.service
After=docker.service
StartLimitIntervalSec=0

[Service]
EnvironmentFile=-/etc/default/consul-service
ExecStartPre=-/usr/bin/docker rm --force %N
ExecStart=/usr/bin/docker run \
    --name=%N \
    --rm=true \
    --network=host \
    --stop-timeout=30 \
    --volume=/etc/grafana:/etc/grafana:ro \
    --volume=/var/lib/grafana:/var/lib/grafana:rw \
    grafana/grafana:latest
ExecStartPost=-/opt/bin/consul-service register \
    -d 1-3 \
    -t devops \
    -a %N \
    -p 80
ExecStop=-/opt/bin/consul-service deregister -a %N
ExecStop=-/usr/bin/docker stop %N
Restart=always
RestartSec=10
KillMode=process

[Install]
WantedBy=multi-user.target
```


Заключение



Ооооо! курочка по зернышку, а в итоге:

- замечательная автоматизация
- увеличение скорости развертывания и доставки
- уменьшение поверхности атаки
- иммутабельность конфигов
- мониторинг
- логирование
- А главное - безответственным сотрудникам гораздо сложнее херовертить

7 июня 2023 📍 Москва, МЦК ЗИЛ
Первая в России конференция
по БЕзопасности КОНтейнеров и контейнерных сред



Contacts:

Tg: @bykva

Site: t.me/bykvaadm