



K8s & Rancher 2

Как разместить свое приложение на нескольких машинах по-быстрояну,
используя возможности Kubernetes



Термины

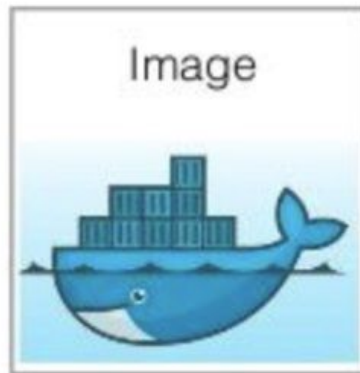
1. Контейнеры и образы
2. K8s
3. Rancher/Rancher 2

Контейнеры и образы

```
FROM ubuntu:14.04
MAINTAINER John Doe <john.doe@example.com>
RUN echo "Hello, Docker!" > /etc/motd
CMD ["echo", "Hello, Docker!"]
```

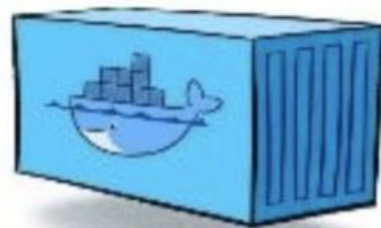
Dockerfile

build



Docker Image

run



Docker Container

Kubernetes

Monolithic Application



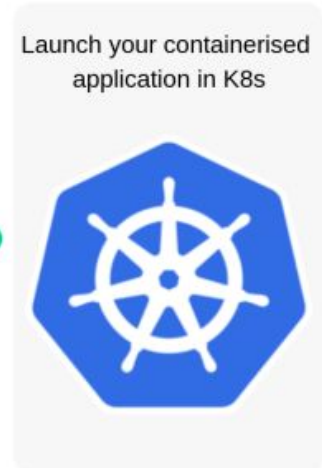
Transition to Microservices



Docker



Kubernetes





Зачем вообще нужен k8s

1. Если у вас 1 машина - то может быть и незачем. И так нормально работает.
2. Если у вас 2 машины - в принципе тоже норм.
3. Если у вас N машин - вы запаритесь ими управлять и обслуживать их.



Инструменты для поддержки машин “оптом”

1. Вспомнился Puppet
2. И наверное таких куча

Но это сводится к тому, что вы множите выполнение скриптов на удаленных машинах. Тех же самых скриптов, что и на одной машине. Только теперь на N.

Это может вызывать проблемы, когда что-то пойдет не так, и придется вручную подключаться к машинам, и что-то на них исправлять. Может и не вызывать, разумеется.



docker-compose & docker run

```
$ docker container run -d --name my_nginx nginx
```

```
version: '2'
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - ./code
    depends_on:
      - redis
  redis:
    image: redis
```



Работа k8s с контейнерами

1. Тоже умеет запускать и останавливать контейнеры
2. Тоже умеет поднимать и настраивать сети
3. Умеет всё то же, что вы могли бы делать самостоятельно с помощью нативного управления машинами, или нативного управления контейнерами на этих машинах
4. Но еще умеет дополнительно кучу всего
5. О некоторых штуках из кучи мы поговорим



Из минусов

ОЧЕНЬ СЛОЖНАЯ КОНФИГУРАЦИЯ

Все эти kind, pod, deployment, ingress, кучи параметров к ним, да еще и отступы.

По крайней мере на первый, второй, третий взгляд.

```
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: hostname-101-deployment
spec:
  replicas: 3
  selector:
    # Like saying "Make sure there are three pods running
    # with the label app = hostname and version = v101"
    matchLabels:
      app: hostname
      version: v101
  template:
    metadata:
      labels:
        # The `app` label is used by both the service
        # and the deployment to select the pods they operate on.
        app: hostname
        # The `version` label is used only by the deployment
        # to control replication.
        version: v101
    spec:
      containers:
        - name: nginx-hostname
          image: kubegoldenguide/nginx-hostname:1.0.1
          ports:
            - containerPort: 80
```

Edit a resource

YAML

JSON

```
1 kind: Pod
2 apiVersion: v1
3 metadata:
4   name: nginx-b4jt9
5   generateName: nginx-
6   namespace: default
7   selfLink: /api/v1/namespaces/default/pods/nginx-b4jt9
8   uid: 1c986895-31aa-11e8-b178-901b0e532516
9   resourceVersion: '14473'
10  creationTimestamp: '2018-03-27T10:32:12Z'
11  labels:
12    app: nginx
13  ownerReferences:
14    - apiVersion: v1
15      kind: ReplicationController
16      name: nginx
17      uid: 9657d0f5-318e-11e8-b178-901b0e532516
18      controller: true
19      blockOwnerDeletion: true
20  spec:
21    volumes:
22      - name: default-token-vv954
23        secret:
24          secretName: default-token-vv954
25          defaultMode: 420
```



Другие системы оркестрации

1. Docker Swarm
2. Mesos
3. CoreOS Fleet
4. ...
5. целиковые облачные сервисы
6. и т.д.
7. до-хре-на



Rancher 2

На самом деле объяснить можно кратко - это неплохой GUI для k8s.

Эта штука позволяет “мышкой” и “клавой” запустить кластер и запустить в нем свое приложение.

Так как мы обещали за 10 минут это всё дело настроить и запустить - приступим.



Оговорка

Как правило, на начальном этапе у всех компаний есть сервер, или набор серверов, арендованных у хостера.

К этим серверам есть доступ по ssh, и все производимые операции вам будет делать просто в соответствии с документацией.

Я же буду использовать виртуальные машины от гугл, просто потому что они мне дали 300 баксов на исследование Google Cloud Platform.



Установка Rancher 2.

Ранчер поддерживает разные виды установок, в том числе отказоустойчивую установку, с распределением самого ранчера на несколько нод. Ее сделать не очень сложно, все мануалы доступны на странице установки. Главное внимательно.

Мы будем делать самый простой и быстрый вид установки.

Из требований: на всех машинах, содержащих ранчер, или являющимися нодами вашего приложения должен быть установлен Docker. И больше ничего. Или, по крайней мере, больше ничего страшного.



Всё есть в документации

<https://rancher.com/docs/rancher/v2.x/en/quick-start-guide/deployment/quickstart-manual-setup/>

manual docker installation

```
$ sudo docker run -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher
```

Запускаем “основную голову” rancher. Публикуем порты 80 & 443 для подключения к админке.

Наличие домена с сертификатом не мешает, но некритично.

Конфигурация головы

Конфигурация машины ?

Семейство машин

Стандартные

Большой объем памяти

Оптимизированные для вычислений

Экономичные и гибкие типы машин для стандартных рабочих нагрузок.

Серия

N1

На базе платформы ЦП Intel Skylake или платформы предыдущего поколения

Тип машины

n1-standard-1 (1 ВЦП, 3,75 ГБ памяти)



ВЦП

1

Память

3,75 ГБ

Машина, которую я взял на Google Cloud. 1 виртуальный процессор и почти 4 гига памяти.

Сгодится, но кажется, что даже меньшая машина потянет этот инструмент.



Подключение не защищено

Злоумышленники могут пытаться похитить ваши данные с сайта **35.246.41.214** (например, пароли, сообщения или номера банковских карт). [Подробнее...](#)

NET::ERR_CERT_AUTHORITY_INVALID

- ☐ Отправлять в Google [URL](#) и контент [некоторых посещенных страниц](#), а также ограниченную [информацию о системе](#) для повышения безопасности Chrome. [Политика конфиденциальности](#)

Дополнительные

К настройкам безопасности

Welcome to Rancher

The first order of business is to set a strong password for the default `admin` user.

☒ Allow collection of anonymous statistics [Learn More](#)

☒ Set a specific password to use:

New Password

Confirm Password

☐ Use a new randomly generated password:

Continue





Global ▾

Clusters

Apps

Users

Settings

Security ▾

Tools ▾



Clusters

[Add Cluster](#)

State ▾

Cluster Name ▾

Provider ▾

Nodes ▾

CPU ▾

RAM ▾

There are no clusters defined.



Возьмем у гугла пару машин

Но возьмем послабее. Должно хватить мощей, возьмем самый дешевый f1-micro

И сделаем из них наш первый кластер.



Add Cluster

Search



Add Cluster - Select Cluster Type



From existing nodes (Custom)

Create a new Kubernetes cluster using RKE, out of existing bare-metal servers or virtual machines.



Import an existing cluster

Import an existing Kubernetes cluster. The provider that created it will continue to manage the provisioning and configuration of the cluster.

With RKE and new nodes in an infrastructure provider



Amazon EC2



Azure



DigitalOcean



Linode



vSphere

With a hosted Kubernetes provider



Amazon EKS



Azure AKS



Google GKE

Cancel



Add Cluster - Custom

Cluster Name *

[Add a Description](#)

my-first-cluster

▶ Member Roles

Control who has access to the cluster and what permission they have to change it.

▶ Labels & Annotations

Configure labels and annotations for the cluster.

None

Cluster Options

[Edit as YAML](#)

[Expand All](#)

▼ Kubernetes Options

Customize the kubernetes cluster options

Kubernetes Version

v1.17.2-rancher1-2



Network Provider

Canal (Network Isolation Available)



Windows Support

☐ Enabled

☒ Disabled

Project Network Isolation

☐ Enabled

☒ Disabled

explicitly configured for the chosen network provider (disabling auto-discovery). The override must be calculated from the host's MTU minus the CNI plugin's required overhead.

Cloud Provider 



If your cloud provider is not listed, please use the **Custom** option.

- ☒ None
- ☐ Amazon
- ☐ Azure
- ☐ Custom
- ☐ External

▼ Private Registry

Configure a default private registry for this cluster. When enabled, all images required for cluster provisioning and system add-ons startup will be pulled from this registry.

Private Registry

- ☒ Disabled
- ☐ Enabled

▶ Advanced Options

Customize advanced cluster options

Эту команду надо запустить на будущей ноде

1

Node Options

Choose what roles the node will have in the cluster

Node Role

☒ etcd

☒ Control Plane

☒ Worker

[Show advanced options](#)

2

Run this command on one or more existing machines already running a supported version of Docker.

```
sudo docker run -d --privileged --restart=unless-stopped --net=host -v /etc/kubernetes:/etc/kubernetes -v /var/run:/var/run  
rancher/rancher-agent:v2.3.5 --server https://35.246.41.214 --token f5dstgvdt8bfp6xt7jhgn6s524k4whpdmhd6z2vp49htwxfzks6wfb --ca-  
checksum a67ca092b10e5f61e12c46b176b06f71ae503a3ba0973b4768253c4ac2f8e377 --etcd --controlplane --worker
```





Конфигурация машины ?

Семейство машин

- Стандартные
- Большой объем памяти
- Оптимизированные для вычислений

Экономичные и гибкие типы машин для стандартных рабочих нагрузок.

Серия

- N1

На базе платформы ЦП Intel Skylake или платформы предыдущего поколения

Тип машины

- f1-micro (1 ВЦП, 614 МБ памяти)



ВЦП
1 общее ядро

Память
614 МБ

Конфигурация машины ?

Семейство машин

- Стандартные
- Большой объем памяти
- Оптимизированные для вычислений

Экономичные и гибкие типы машин для стандартных рабочих нагрузок.

Серия

- N1

На базе платформы ЦП Intel Skylake или платформы предыдущего поколения

Тип машины

- f1-micro (1 ВЦП, 614 МБ памяти)



ВЦП
1 общее ядро

Память
614 МБ



Сначала будет так.



This cluster is currently **Provisioning**; areas that interact directly with it will not be available until the API is ready.

Waiting for etcd and controlplane nodes to be registered

Dashboard: my-first-cluster



Launch kubectl



Kubeconfig File



Graph information from Cluster currently not available



Потом будет setup tunnel to host



This cluster is currently **Provisioning**; areas that interact directly with it will not be available until the API is ready.

[dialer] Setup tunnel for host [10.154.0.3]

Dashboard: my-first-cluster



Launch kubectl



Kubeconfig File



Graph information from Cluster currently not available



This cluster is currently **Provisioning**; areas that
[network] Deploying port listener containers



This cluster is currently **Provisioning**; areas that interact directly with it will not be available.

[network] Successfully started [rke-etcd-port-listener] container on host [10.154.0.3]

Dashboard: my-first-cluster

Graph information



И в итоге мы придем к... Чуть позже покажу.

Оно как раз задеплоится и настроится.

Чтобы было еще меньше проблем, давайте попробуем наш запущенный ранчер присобачить к реальному Google Kubernetes Engine

Добавим еще 1 кластер

Add Cluster - Select Cluster Type



From existing nodes (Custom)

Create a new Kubernetes cluster using RKE, out of existing bare-metal servers or virtual machines.



Import an existing cluster

Import an existing Kubernetes cluster. The provider that created it will continue to manage the provisioning and configuration of the cluster.

With RKE and new nodes in an infrastructure provider



Amazon EC2



Azure



DigitalOcean



Linode



vSphere

With a hosted Kubernetes provider



Amazon EKS



Azure AKS



Google GKE

Введите ключ API, в GCP можно получить его.

Add Cluster - Google GKE

Cluster Name *

[Add a Description](#)

gke1

Member Roles


Control who has access to the cluster and what permission they have to change it.

Labels & Annotations

Configure labels and annotations for the cluster.

None

Service Account *

 [Read from a file](#)

Service account private key JSON file

Create a [Service Account](#) with a JSON private key and provide the JSON here. See [Google Cloud docs](#) for more info about creating a service account. These IAM roles are required: Compute Viewer ([roles/compute.viewer](#)), (Project) Viewer ([roles/viewer](#)), Kubernetes Engine Admin ([roles/container.admin](#)), Service Account User ([roles/iam.serviceaccountuser](#)). More info on roles can be found [here](#).

[Next: Configure Nodes](#)

[Cancel](#)



Почти все настройки - настройки по умолчанию



Выставляем, сколько и чего нам нужно

Node Options

Customize the nodes that will be created

Node Count

3

Machine Type

n1-standard-2 (2 vCPUs, 7.5 GB RAM)

Image Type

Ubuntu

Root disk type

Standard persistent disk

Root Disk Size

100

GB

Local SSD disks

0

GB

Preemptible nodes (beta)

☐ Enabled ☒ Disabled

Auto Upgrade

☐ Enabled ☒ Disabled

Auto Repair

☐ Enabled ☒ Disabled

Node Pool Autoscaling






Node Pool Autoscaling

☐ Enabled ☒ Disabled

Жmem Create

Clusters

[Add Cluster](#)

Delete 		Search				
<input type="checkbox"/>	State 	Cluster Name 	Provider 	Nodes 	CPU 	RAM 
<input type="checkbox"/>	Provisioning	gke1	Google GKE	0	n/a	n/a 
<input type="checkbox"/>	Provisioning	my-first-cluster	Custom	1	n/a	n/a 

[workerPlane] Failed to bring up Worker Plane: [Failed to start [kubelet] container on host [10.154.0.4]: Error response from daemon: error while creating mount source path '/opt/cni': mkdir /opt/cni: read-only file system]



Ха, кластер отвалился.

Кластер который мы делали на виртуальных машинах гугла, как будто на выделенных серверах не завелся.

Связано это с тем, что на этих VPS есть некоторые ограничения по использованию их как VDS.

На “чистых” арендованных VDS или VPS на базе KVM такого не будет. Поэтому если не хотите использовать облака, лучше арендуйте VPS на базе KVM.

СЕРВЕРЫ ЗА 5 МИНУТ

СВОЯ КОНФИГУРАЦИЯ

NEW!
CPU 5 ГГц

ОТКАЗОУСТОЙЧИВЫЙ СЕРВЕР

Виртуализация

OVZ

KVM



HDD+SSD

SSD

NVMe

Диск



VDS
Старт

269 ₽
В МЕСЯЦ

1 ядро
процессора

1 Гб
оперативной памяти

20 Гб
жёсткого диска



ЗАКАЗАТЬ СЕРВЕР

VDS
Разгон

539 ₽
В МЕСЯЦ

2 ядра
процессора

2 Гб
оперативной памяти

40 Гб
жёсткого диска



ЗАКАЗАТЬ СЕРВЕР

VDS
Отрыв

989 ₽
В МЕСЯЦ

4 ядра
процессора

4 Гб
оперативной памяти

60 Гб
жёсткого диска



ЗАКАЗАТЬ СЕРВЕР

VDS
Улёт

1439 ₽
В МЕСЯЦ

6 ядер
процессора

6 Гб
оперативной памяти

80 Гб
жёсткого диска



ЗАКАЗАТЬ СЕРВЕР

Тадаааам, active.



Global ▾ Clusters Apps Users Settings Security ▾ Tools ▾



Clusters

Add Cluster

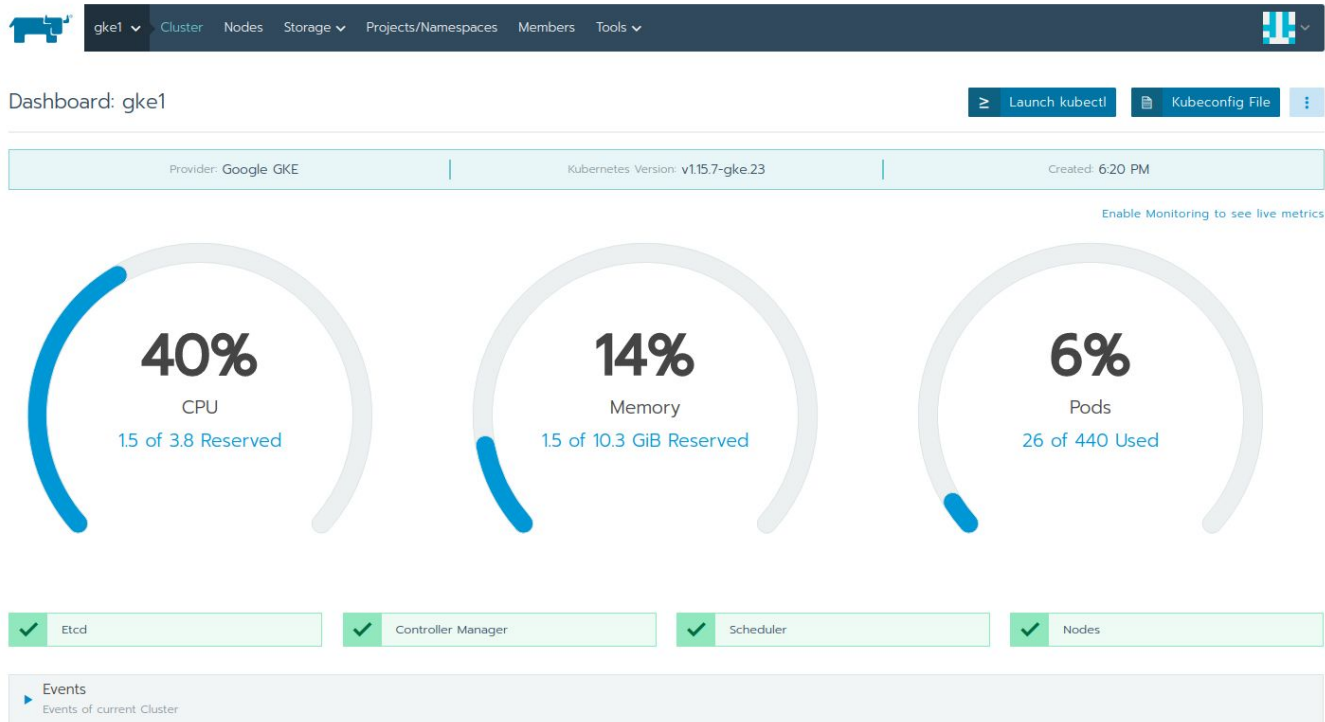
Delete 1 Cluster

Search

<input type="checkbox"/> State ▾	Cluster Name ▾	Provider ▾	Nodes ▾	CPU ▾	RAM ▾	
<input type="checkbox"/> Active	gke1	Google GKE v1.15.7-gke.23	4	0.8/3.8 Cores 21%	0.4/10.3 GiB 4%	
<input checked="" type="checkbox"/> Removing	my-first-cluster	Custom	1	n/a	n/a	

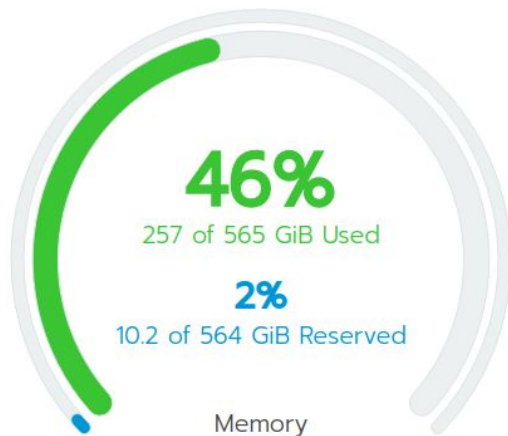
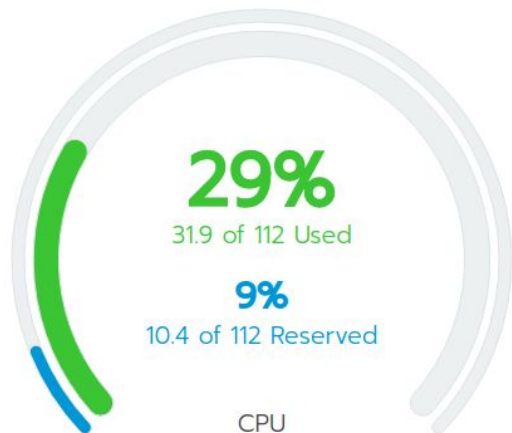
Removing host [10.154.0.4] from node lists; waiting on cluster-provisioner-controller

Здесь могла быть ваша статистика. Prometheus



В реальности может выглядеть так

Provider: Custom	Version: v1.13.5	Nodes: 9
CPU: 112 Cores	Memory: 564 GiB	Created: 07/12/2019



✓ Etd

✓ Controller Manager

✓ Scheduler

✓ Nodes

[Expand All](#)

Добавим проект

35.246.41.214/c/c-4csnw/projects-namespaces



gke1 ▾

Cluster

Nodes

Storage ▾

Projects/Namespaces

Members

Tools ▾



Projects/Namespaces

Move ↗

Download YAML ⬇

Delete 🗑



Add Project

Search



State ⌵

Namespace Name ⌵

Created ⌵

Project: Default

Default project created for the cluster

Add Namespace



Active

default

6:23 PM



Можно выделить квоты в рамках проекта

Add Project

Project Name * [Add a Description](#)

Project1

Members

Configure who has access to the resources in this project and what permissions they have

Name

Role



Default Admin (admin)
Local User

Project Owner



Add Member

Resource Quotas

Configure how much of the resources the project can consume



Add Quota

Container Default Resource Limit

Configure how much of the resources the container can consume by default

CPU Limit

e.g. 1000

milli CPUs

Memory Limit

e.g. 128

MiB

CPU Reservation

e.g. 1000

milli CPUs

Memory Reservation

e.g. 128

MiB

Создадим namespace. Квоты тоже есть.

Add Namespace

Name

[Add a Description](#)

Namespace1

Project

Project1



Container Default Resource Limit

Configure how much of the resources the container can consume by default

CPU Limit

e.g. 1000

milli CPUs

Memory Limit

e.g. 128

MiB

CPU Reservation

e.g. 1000

milli CPUs

Memory Reservation

e.g. 128

MiB

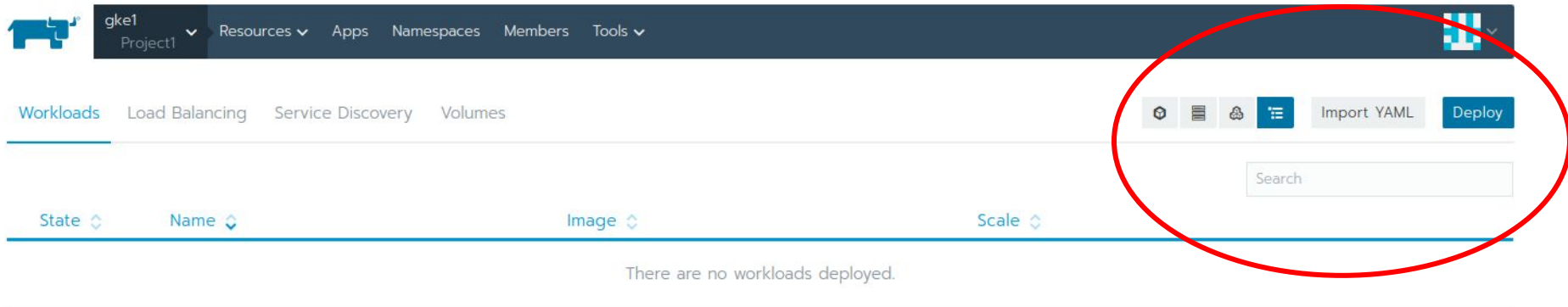


Labels & Annotations

Key/Value pairs that can be used to label/annotate resources.

None

Заходим в проект и жмем Deploy



The screenshot shows the Google Kubernetes Engine (GKE) console interface. At the top, there is a dark blue header bar with the GKE logo on the left and a user profile icon on the right. Below the header, there is a navigation bar with the following items: "gke1", "Project1", "Resources", "Apps", "Namespaces", "Members", and "Tools". Below the navigation bar, there is a sub-navigation bar with the following items: "Workloads", "Load Balancing", "Service Discovery", and "Volumes". Below the sub-navigation bar, there is a toolbar with the following items: a gear icon, a list icon, a refresh icon, a menu icon, an "Import YAML" button, and a "Deploy" button. The "Deploy" button is circled in red. Below the toolbar, there is a search bar with the text "Search". Below the search bar, there is a table with the following columns: "State", "Name", "Image", and "Scale". The table is empty, and the text "There are no workloads deployed." is displayed below it.

Развернем ради интереса gitea/gitea



Search for great content (e.g., mysql)

Explore

Sign In

Pricing

Get Started



gitea/gitea ☆

By [gitea](#) • Updated 9 hours ago

Gitea: Git with a cup of tea - A painless self-hosted Git service.

Container

↓ Pulls 10M+

Overview

Tags

Gitea

build success

48.2MB

18 layers

chat 261 online

Gitea is a community managed painless self-hosted Git service. You can see a Gitea demo [here](#) (running on tip).

For more information, check out the [project on GitHub](#), and for instructions on how to set up, our [documentation](#). In case you need any help, don't hesitate to come on our [Discord server](#) or our [forum](#)!

Quick set up

Docker Pull Command

```
docker pull gitea/gitea
```



Owner



gitea

Deploy Workload


Name *

[Add a Description](#)

Workload Type

[More options](#)

mariadb

 Scalable deployment of pod

Docker Image *

mariadb:10



Namespace *

[Add to a new namespace](#)

namespace1



Port Mapping

[+](#) Add Port

[Expand All](#)

▼ Environment Variables

Set the environment that will be visible to the container, including injecting values from other resources like Secrets.

Environment Variables

Variable *

Value

MYSQL_ROOT_PASSWORD

=

changeme



MYSQL_DATABASE

=

gitea



MYSQL_USER

=

gitea



MYSQL_PASSWORD

=

changeme



ProTip: Paste lines of key=value pairs into any key field for easy bulk entry.

[+](#) Add Variable

БД разворачивается

Workloads Load Balancing Service Discovery Volumes

Redeploy ⌂ Pause Orchestration || Download YAML ⬇ Delete 🗑

Search

State ⌄	Name ⌄	Image ⌄	Scale ⌄
Namespace: namespace1			
<input type="checkbox"/> ▶ Updating	mariadb 🗄 Deployment does not have minimum availability.	mariadb:10 1 Pod / Created a few seconds ago / Pod Restarts: 0	1

Workloads Load Balancing Service Discovery Volumes

Redeploy ⌂ Pause Orchestration || Download YAML ⬇ Delete 🗑

Search

State ⌄	Name ⌄	Image ⌄	Scale ⌄
Namespace: namespace1			
<input type="checkbox"/> ▶ Active	mariadb 🗄	mariadb:10 1 Pod / Created a few seconds ago / Pod Restarts: 0	1

Теперь gitea

Deploy Workload

Name *

[Add a Description](#)

gitea

Workload Type

[More options](#)



Scalable deployment of 1 pod

Docker Image *

gitea/gitea:1.8.3



Namespace *

[Add to a new namespace](#)

namespace1



Port Mapping



Add Port

[Expand All](#)



Environment Variables

Set the environment that will be visible to the container, including injecting values from other resources like Secrets.

Доступ по http к gitea



Чтобы мы могли получить доступ к gitea, мы должны настроить балансировщик трафика. Вперед, переходим во вкладку Load Balancing.

Add Ingress



gke1

Project1

Resources

Apps

Namespaces

Members

Tools



Workloads

Load Balancing

Service Discovery

Volumes

Import YAML

Add Ingress

Search

State

Name

Targets

Created

There are no ingress rules defined

Создадим доменное имя на xip.io сервисе

Add Ingress

Name

[Add a Description](#)

Namespace *

[Add to a new namespace](#)

gitea-ingress

namespace1

Rules

- ☒ Automatically generate a `xip.io` hostname
- ☐ Specify a hostname to use
- ☐ Use as the default backend

Target Backend



Service



Workload

Path

Target

Port *

e.g. /foo

gitea

3000

[+ Add Rule](#)

Можно и свой домен привязать.



Никто не запрещает использовать собственные доменные имена. Разумеется, они настраиваются так же легко, как и раньше. Использовать можно те же инструменты для управления доменными именами, что и ранее. Я использую зоны на базе Google Cloud.



Пошла жара.

Когда загорится зеленым, можно будет вернуться на вкладку workspace и открыть оттуда наш сервис gitea.

Download YAML ⬇

Delete 🗑

Search

❏

State ⬇

Name ⬇

Targets

Created ⬇

Namespace: namespace1

❏

Initializing

gitea-ingress




L7 Ingress

xip.io > gitea

6:57 PM







Workloads Load Balancing Service Discovery Volumes

    Import YAML Deploy

Redeploy  Pause Orchestration  Download YAML  Delete 

Search

<input type="checkbox"/>	State	Name	Image	Scale
Namespace: namespace1 				
<input type="checkbox"/>	▶ Active	<div><div>gitea </div><div>80/http-31154/tcp</div></div>	<div><div>gitea/gitea:18.3</div><div>1 Pod / Created 5 minutes ago / Pod Restarts: 0</div></div>	<div><div><div></div></div><div>1 </div></div>
<input type="checkbox"/>	▶ Active	<div><div>mariadb </div></div>	<div><div>mariadb:10</div><div>1 Pod / Created 8 minutes ago / Pod Restarts: 0</div></div>	<div><div><div></div></div><div>1 </div></div>



Главная

Обзор

Помощь

👤 Регистрация



Gitea: Git with a cup of tea

Удобная служба для собственного Git-репозитория



Простой в установке

Просто запустите исполняемый файл для вашей платформы. Используйте Gitea с [Docker](#) или [Vagrant](#), или загрузите [пакет](#).



Кроссплатформенный

Gitea работает на любой операционной системе, которая может компилировать [Go](#): Windows, macOS, Linux, ARM и т. д. Выбирайте, что вам больше нравится!



Легковесный

Gitea имеет низкие системные требования и может



Открытый исходный код

Всё это на [GitHub](#)! Присоединяйтесь к нам, внося вклад,

Начальная конфигурация

Если вы запускаете Gitea внутри Docker, пожалуйста внимательно прочтите [документацию](#) перед тем, как изменить любые настройки.

Настройки базы данных

Для Gitea требуется MySQL, PostgreSQL, MSSQL или SQLite3.

Тип базы данных SQLite3

SQLite3

```
/data/gitea/gitea.db
```

Путь /data/gitea/gitea.db

Введите абсолютный путь, если вы запускаете Gitea как службу.

Основные настройки

Название сайта *	Гитеа: Git with a cup of tea
Гитеа: Git with a cup of tea	

Gitea: Git with a cup of tea

Здесь вы можете ввести название своей компании.

```
/data/git/repositories
```

Путь корня репозитория

Все удаленные Git репозитории будут сохранены в этот каталог.

```
/data/git/lfs
```

Корневой путь Git LFS	/data/git/lfs
-----------------------	---------------

В этой папке будут храниться файлы Git LFS. Оставьте пустым, чтобы отключить LFS.

git

Запуск от имени пользователя

localhost

Введите имя пользователя операционной системы, под которым работает Git. Обратите внимание, что этот пользователь должен иметь доступ к корневому пути репозитория.

Домен SSH сервера *

22



В общем-то готово. Приложение работает.

Оно развернуто в kubernetes.

Да, есть ряд моментов. Мы затронем их косвенно сейчас, потому что время. Остальное на afterparty.

1. персистентность приложений.
2. скалирование



Персистентность

Например, речь о базах данных. Им нужно сохранять данные на диске.

1. можем подмонтировать директорию физической машины. Тогда данные будут сохраняться примерно в том же режиме, в котором работает `docker volumes`.
2. можем развернуть отказоустойчивую службу томов, типа `longhorn`, она будет распределена по кластеру и будет отказоустойчивой. В дальнейшем ее можно будет примонтировать к любому контейнеру (точнее поду) вашего приложения.
3. Можно развернуть персистентные хранилища в том же гугле.

Первые 2 пункта подойдут, если вы используете обыкновенные машины, арендованные, скажем на `firstvds` или `hetzner`. 3 подойдет, если вы, как и я, любите облака.



Важный момент.

Я всё же приверженец того, что если вы используете базы данных - используйте отдельные физические машины с базами данных, а также репликации и шардинг. Это даст бОльшую гарантию сохранности данных, если вы в начале пути по масштабированию своих проектов.

Как поднатаскаетесь - можно будет и базу смело размещать в облаке. Google Cloud, например, предлагает целый сервис по хостингу баз данных, со встроенным шардированием, репликами, бекапами и прочей ерундой. Такие же штуки предлагают почти все крупные облачные провайдеры.

Скалирование.

Нам ведь нужен kubernetes для того, чтобы получить горизонтальное масштабирование на всех машинах кластера, верно? Жмем на знак плюса в ранчере...

The screenshot shows the Rancher management interface. At the top, a dark navigation bar contains the Rancher logo, a dropdown menu for 'gke1 Project1', and links for 'Resources', 'Apps', 'Namespaces', 'Members', and 'Tools'. On the right of this bar is a user profile icon. Below the navigation bar, a secondary bar features tabs for 'Workloads', 'Load Balancing', 'Service Discovery', and 'Volumes'. To the right of these tabs are icons for settings, a list, a trash can, and a menu, followed by buttons for 'Import YAML' and 'Deploy'. The main content area displays a list of workloads. At the top of this area are buttons for 'Redeploy', 'Pause Orchestration', 'Download YAML', and 'Delete', along with a search input field. The workload list has columns for 'State', 'Name', 'Image', and 'Scale'. One workload is visible: 'gitea' in namespace 'namespace1'. Its state is 'Active', its image is 'gitea/gitea:18.3', and its scale is 1. The interface also shows pod details: '1 Pod / Created 5 minutes ago / Pod Restarts: 0'. At the bottom right, there are minus and plus buttons for adjusting the scale.

State	Name	Image	Scale
Active	gitea	gitea/gitea:18.3	1

Namespace: namespace1

gitea/gitea:18.3
1 Pod / Created 5 minutes ago / Pod Restarts: 0



On!

Workloads Load Balancing Service Discovery Volumes



Import YAML

Deploy

Redeploy ↻ Pause Orchestration || Download YAML ⬇ Delete 🗑 1 Workload

Search

State ↕

Name ↕

Image ↕

Scale ↕

Namespace: namespace1



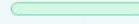
Active

gitea 🗄

80/http, 31154/tcp

gitea/gitea:18.3

2 Pods / Created 21 minutes ago / Pod Restarts: 0



2

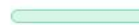


Active

mariadb 🗄

mariadb:10

1 Pod / Created 23 minutes ago / Pod Restarts: 0



1



Даже без предварительной настройки движок kubernetes раскидал наше приложение по 2 нодам.

Workload: gitea

Active



Namespace: namespace1

Image: gitea/gitea:18.3

Workload Type: Deployment

Endpoints: 80/http, 31154/tcp

Config Scale: 2
Ready Scale: 2



Created: 6:53 PM
Pod Restarts: 0

Expand All

Pods

Pods in this workload

Download YAML

Delete

State

Name

Image

Node

☐ Running gitea-594c4bf97-wjdxl gitea/gitea:18.3
10.44.14 / Created 22 minutes ago / Restarts: 0

☐ Running gitea-594c4bf97-jkqks gitea/gitea:18.3
10.44.3.4 / Created a minute ago / Restarts: 0

gke-c-4csnw-default-0-d96a5b13-z7vz
34.89.4.156 / 10.154.0.7

gke-c-4csnw-default-0-28c1c4e3-gj54
35.230.133.189 / 10.154.0.5



При этом балансировщик работает как надо

Он будет направлять трафик на менее загруженную ноду, если не задавать ему никаких специальных правил.

Обратите внимание на правила скейла

Deploy Workload

Name *

[Add a Description](#)

e.g. myapp

Workload Type

- ☒ Scalable deployment of pod
- ☐ Run one pod on each node
- ☐ Stateful set of 1 pod
- ☐ Run on a cron schedule
- ☐ Job

Docker Image *

Namespace *

[Add to a new namespace](#)



Есть даже правил “1 экз. на ноду”. Это удобно.





Образование.

Всё это не было бы так классно, если бы не еще 2 бонуса.

1. Возможность работать с k8s YAML. Жмем...

Workloads Load Balancing Service Discovery Volumes

Redeploy ⌂ Pause Orchestration ⏸ Download YAML ⬇ Delete 🗑 Search

	State ⌄	Name ⌄	Image ⌄	Scale ⌄
Namespace: namespace1				
<input type="checkbox"/>	Active	gitea  80/http, 31154/tcp	gitea/gitea:18.3 2 Pods / Created 27 minutes ago / Pod Restarts: 0	<div><div></div><div></div></div>
<input type="checkbox"/>	Active	mariadb  	mariadb:10 1 Pod / Created 29 minutes ago / Pod Restarts: 0	<div><div></div><div></div></div>


- Edit
- Clone
- Redeploy
- Add a Sidecar
- Rollback
- Execute Shell
- Pause Orchestration
- View/Edit YAML
- View in API

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   annotations:
5     deployment.kubernetes.io/revision: "1"
6     field.cattle.io/creatorId: user-hkz4p
7     field.cattle.io/publicEndpoints: '[{"addresses":["34.89.4.156"],"port":31154,"protocol":"TCP","serviceName":"namespace1:ingress-d9b16610ca37e3da6e87fc3225e54fe6","allNodes":true},
8 {"addresses":["34.102.195.140"],"port":80,"protocol":"HTTP","serviceName":"namespace1:ingress-d9b16610ca37e3da6e87fc3225e54fe6","ingressName":"namespace1:gitea-ingress","hostname":"gitea-
9 ingress.namespace1.34.102.195.140.xip.io","allNodes":false}]'
10   creationTimestamp: "2020-01-30T15:53:26Z"
11   generation: 4
12   labels:
13     cattle.io/creator: norman
14     workload.user.cattle.io/workloadselector: deployment-namespace1-gitea
15   name: gitea
16   namespace: namespace1
17   resourceVersion: "13485"
18   selfLink: /apis/apps/v1/namespaces/namespace1/deployments/gitea
19   uid: 83ba0f72-a9a9-462b-91a1-91001760f984
20 spec:
21   progressDeadlineSeconds: 600
22   replicas: 2
23   revisionHistoryLimit: 10
24   selector:
25     matchLabels:
26       workload.user.cattle.io/workloadselector: deployment-namespace1-gitea
27   strategy:
28     rollingUpdate:
29       maxSurge: 1
30       maxUnavailable: 0
31     type: RollingUpdate
32   template:
33     metadata:
```

 Copy to Clipboard

То, что нас так пугает в начале нашего пути знакомства с k8s. Сгенерировано автоматически. И можно редактировать.

Качайтесь с ним в k8s.



```
51     terminationMessagePolicy: File
52     tty: true
53     dnsPolicy: ClusterFirst
54     restartPolicy: Always
55     schedulerName: default-scheduler
56     securityContext: {}
57     terminationGracePeriodSeconds: 30
58 status:
59   availableReplicas: 2
60   conditions:
61   - lastTransitionTime: "2020-01-30T15:53:26Z"
62     lastUpdateTime: "2020-01-30T15:53:34Z"
63     message: ReplicaSet "gitea-594c4bf97" has successfully progressed.
64     reason: NewReplicaSetAvailable
65     status: "True"
66     type: Progressing
67   - lastTransitionTime: "2020-01-30T16:14:35Z"
68     lastUpdateTime: "2020-01-30T16:14:35Z"
69     message: Deployment has minimum availability.
70     reason: MinimumReplicasAvailable
71     status: "True"
72     type: Available
73   observedGeneration: 4
74   readyReplicas: 2
75   replicas: 2
76   updatedReplicas: 2
```


2. Отличное http api у rancher


The screenshot displays the Rancher management interface. At the top, there are tabs for 'Workloads', 'Load Balancing', 'Service Discovery', and 'Volumes'. Below these are action buttons: 'Redeploy', 'Pause Orchestration', 'Download YAML', and 'Delete'. A search bar is located on the right. The main table lists workloads in the 'namespace1' namespace. The 'gitea' workload is highlighted, and a context menu is open over it, showing options like 'Edit', 'Clone', 'Redeploy', 'Add a Sidecar', 'Rollback', 'Execute Shell', 'Pause Orchestration', 'View/Edit YAML', 'View in API' (which is circled in green), and 'Delete'.


State	Name	Image	Scale
Active	gitea	gitea/gitea:1.8.3	2 Pods / Created 27 minutes ago / Pod Restarts: 0
Active	mariadb	mariadb:10	1 Pod / Created 29 minutes ago / Pod Restarts: 0


А также возможность использовать методы API прямо из браузера. Удобно для тестирования.


```
{
  -actions: {
    pause: ".../v3/project/c-4csnw:p-jtdsx/workloads/deployment:namespace1:gitea?action=pause",
    resume: ".../v3/project/c-4csnw:p-jtdsx/workloads/deployment:namespace1:gitea?action=resume",
    rollback: ".../v3/project/c-4csnw:p-jtdsx/workloads/deployment:namespace1:gitea?action=rollback"
  },
  -annotations: {
    cattle.io/timestamp: "2020-01-30T15:53:26Z"
  },
  baseType: "workload",
  -containers: [
    -{
      allowPrivilegeEscalation: false,
      image: "gitea/gitea:1.8.3",
      imagePullPolicy: "Always",
      initContainer: false,
      name: "gitea",
      ports: null,
      privileged: false,
      readOnly: false,
      -resources: {
        type: "/v3/project/schemas/resourceRequirements"
      },
      restartCount: 0,
      runAsNonRoot: false,
      stdin: true,
      stdinOnce: false,
      terminationMessagePath: "/dev/termination-log",
      terminationMessagePolicy: "File",
      tty: true,
      type: "/v3/project/schemas/container"
    }
  ],
  created: "2020-01-30T15:53:26Z",
  createdTS: 1580399606000,
  creatorId: null,
  -deploymentConfig: {
    maxSurge: 1,
    maxUnavailable: 0,
    minReadySeconds: 0
  }
}
```

Operations

 Up

 Reload

 Edit

 Delete

Actions

pause

resume

rollback

Filter

Not available

Sort

Not available

Pagination

Not available

admin [Log Out](#)

```

- actions: {
    pause: ".../v3/proj
    resume: ".../v3/proj
    rollback: ".../v3/pr
  },
- annotations: {
    cattle.io/timestam
  },
  baseType: "workload",
- containers: [
  - {
      allowPrivilegeE
      image: "gitea/g
      imagePullPolicy
      initContainer:
      name: "gitea",
      ports: null,
      privileged: fal
      readOnly: false
    }
  - resources: {
      type: ".../v3/pr
    },
    restartCount: 0
    runAsNonRoot: fa
    stdin: true,
    stdinOnce: fals
    terminationMess
    terminationMess
    tty: true,
    type: ".../v3/proj
  }
],
created: "2020-01-30T
createdTS: 1580399606
creatorId: null,
- deploymentConfig: {
  maxSurge: 1,
  maxUnavailable: 0,

```

Operations



Edit deployment



Field Name	Type	Value
activeDeadlineSeconds	int	<input type="text" value=""/>
annotations	map[string]	<div> <div>- cattle.io/time : 2020-01-30T15:53:26Z</div> <div>+ </div> </div>
automountServiceAccountToken	boolean	<div> <div>- NULL</div> <div>+ </div> </div>
containers	array[container]	<div> <div>- {"allowPrivilegeEscalation":false,"image":"gitea/gitea:1.8.3","imagePullPolicy":"Always","ini</div> <div>+ </div> </div>
created	date	2020-01-30T15:53:26Z
creatorId	reference[/v3/schemas/user]	-
deploymentConfig	deploymentConfig	<div> <div>- {"maxSurge":1,"maxUnavailable":0,"minReadySeconds":0,"progressDeadlineSeconds":600,"revisionHistoryLimit":10,"strategy":"RollingUpdate","update":</div> <div>+ </div> </div>



Вишенка - ssh прямо в нужный контейнер.

Есть возможность подключиться прямо из браузера в нужный контейнер, чтобы что-то посмотреть или отладить.

Workloads Load Balancing Service Discovery Volumes

Import YAML Deploy

Redeploy ⌂ Pause Orchestration || Download YAML ⬇ Delete 🗑

Search

State ⌵ Name ⌵ Image ⌵ Scale ⌵

Namespace: namespace1
gitea-594c4bf97-jkqks

10.44.3.4

Running

gke-c-4csnw-default-0-28c1c4e3-g154 80/http, 80/tcp

Execute Shell ➤

View Logs 📄

View/Edit YAML ✎

View in API 🔗

Delete 🗑

gitea/gitea:1.8.3

2 Pods / Created 33 minutes ago / Pod Restarts: 0

2

- +

mariadb:10

1 Pod / Created 35 minutes ago / Pod Restarts: 0

1

≥ Shell: gitea

ProTip: Hold the Control key when opening shell access to launch a new window.

```
bash-4.4# ды -д
bash: ды: command not found
bash-4.4#
bash-4.4#
bash-4.4# ls -la
total 84
drwxr-xr-x 1 root root 4096 Jan 30 16:14 .
drwxr-xr-x 1 root root 4096 Jan 30 16:14 ..
-rwxr-xr-x 1 root root  0 Jan 30 16:14 .dockerenv
-rw-r--r-- 1 root root 553 Jun 17 2019 Makefile
drwxr-xr-x 1 root root 4096 Jun 17 2019 app
drwxr-xr-x 1 root root 4096 May 11 2019 bin
drwxr-xr-x 5 root root 4096 Jan 30 16:14 data
drwxr-xr-x 5 root root 400 Jan 30 16:14 dev
drwxr-xr-x 1 root root 4096 Jan 30 16:14 etc
drwxr-xr-x 2 root root 4096 May 9 2019 home
drwxr-xr-x 1 root root 4096 May 11 2019 lib
drwxr-xr-x 5 root root 4096 May 9 2019 media
drwxr-xr-x 2 root root 4096 May 9 2019 mnt
drwxr-xr-x 2 root root 4096 May 9 2019 opt
dr-xr-xr-x 153 root root  0 Jan 30 16:14 proc
drwx----- 2 root root 4096 May 9 2019 root
drwxr-xr-x 1 root root 4096 Jan 30 16:14 run
drwxr-xr-x 1 root root 4096 May 11 2019/sbin
drwxr-xr-x 2 root root 4096 May 9 2019/srv
dr-xr-xr-x 13 root root  0 Jan 30 16:14/sys
drwxrwxrwt 2 root root 4096 May 9 2019/tmp
drwxr-xr-x 1 root root 4096 Jun 17 2019/usr
drwxr-xr-x 1 root root 4096 May 11 2019/var
bash-4.4#
```



На этом всё.

Я в ВК: https://vk.com/vladimir_zakotnev

Я в телеге: <https://t.me/vladitot>

vladitot@gmail.com