# Пара слов про контейнерные оркестраторы

Uladzimir Okala-Kulak

Minsk, 2019

# План

# Про что будем говорить

Про контейнерную оркестрацию:

- назначение и особенности
- принципиальное устройство типичного оркестратора
- поверхностное сравнение решений
- и немного про всё что вокруг...

# Про что не будем говорить (ну почти):

- про облака
- про гипервизоры
- про гипервизорные оркестраторы в стиле oVirt/RHEV, XCP-ng
- про IaaS/PaaS в стиле Openstack, CloudStack, CloudFoundry
- про CaaS и Container-base PaaS в стиле Dokku
- про контейнерные движки и решения не соответствующие OCI типа LXC, LXD
- про работу контейнеров в Linux «под капотом»
- про классификацию систем виртуализации
- про детали применения механизмов изоляции на уровне ядра Linux
- про SOA и микросервисы
- про CI/CD
- про DevOps

# Разные взгляды на контейнерный оркестратор

1. Способ логического объединения группы контейнерных хостов в один кластер
2. «Запускалка» контейнеров, task scheduler или контейнерный «supervisor» на стероидах и с «удобняшками»
3. «Балалайка», позволяющая смотреть на инфраструктуру как на «облако контейнеров»

# Принципиальное устройство типичного оркестратора и основные возможности

1. master/management/controlplane ноды и worker/minion ноды
2. контейнерный движок (docker/containerd/CRI-O/rkt/podman)
3. configuration storage для хранения и распространения конфигов (ZK/etcd/consul)
4. configuration storage или DNS-like решение для sevice discovery (consul/CoreDNS)
5. multi-host network solution, overlay network, network fabric (calico/weave/flannel)
6. Placement policy, Affinity/Antiaffinity policy
7. Self healing, supervising, управление зависимостями и порядком старта/стопа/рестарта
8. Механизмы масштабирования
9. Механизмы балансировки нагрузки
10. Механизмы обеспечения отказоустойчивости и высокой доступности
11. Модель доступа и разграничения прав на ресурсы
12. Дополнительные механизмы обеспечения безопасности и наложения ограничений на ресурсы
13. механизмы обеспечения ZDD, различные Deployment Strategy

# Погнали по примерам. Начнём с простого

**Docker Swarm (Docker swarm mode)**:

- only Docker;
- нативная кластеризация;
- отсутствие процедуры установки, только сборка кластера из Docker-хостов
- внутренняя оверлейная сеть с балансировкой (ingress mesh)
- поддержка сторонних сетевых решений
- возможна проблема «двойной балансировки»
- общеизвестные docker cli и docker-compose (API v3) как основные инструменты
- дополнительные директивы docker-compose API v3
- ограничения для secrets, named volumes и т.п.
- отсутствие примитивов для удобного деплоя (разве что stack)
- отсутствие «пакетного менеджера» типа helm/pkgpanda
- отсутствие «репозитория сценариев деплоя» типа ChartMuseum (свалка примеров docker-compose.yml на github и в блогах не в счёт)

# https://docs.docker.com/engine/swarm/

- **Cluster management integrated with Docker Engine:** Use the Docker Engine CLI to create a swarm of Docker Engines where you can deploy application services. You don't need additional orchestration software to create or manage a swarm.
- **Decentralized design:** Instead of handling differentiation between node roles at deployment time, the Docker Engine handles any specialization at runtime. You can deploy both kinds of nodes, managers and workers, using the Docker Engine. This means you can build an entire swarm from a single disk image.
- **Declarative service model:** Docker Engine uses a declarative approach to let you define the desired state of the various services in your application stack. For example, you might describe an application comprised of a web front end service with message queueing services and a database backend.
- **Scaling:** For each service, you can declare the number of tasks you want to run. When you scale up or down, the swarm manager automatically adapts by adding or removing tasks to maintain the desired state.
- **Desired state reconciliation:** The swarm manager node constantly monitors the cluster state and reconciles any differences between the actual state and your expressed desired state. For example, if you set up a service to run 10 replicas of a container, and a worker machine hosting two of those replicas crashes, the manager creates two new replicas to replace the replicas that crashed. The swarm manager assigns the new replicas to workers that are running and available.
- **Multi-host networking:** You can specify an overlay network for your services. The swarm manager automatically assigns addresses to the containers on the overlay network when it initializes or updates the application.
- **Service discovery:** Swarm manager nodes assign each service in the swarm a unique DNS name and load balances running containers. You can query every container running in the swarm through a DNS server embedded in the swarm.
- **Load balancing:** You can expose the ports for services to an external load balancer. Internally, the swarm lets you specify how to distribute service containers between nodes.
- **Secure by default:** Each node in the swarm enforces TLS mutual authentication and encryption to secure communications between itself and all other nodes. You have the option to use self-signed root certificates or certificates from a custom root CA.

# Погнали дальше

**Пара слов про nomad**:

- оркестратор, но не только контейнерный
- разрабатывается в HashiCorp!
- интегрируется с hashicorp consul
- интегрируется с hashicorp vault
- «дружит» с Terraform и hcl
- поддерживает Docker, rlt, LXC
- ...

# Nomad vs. Other Software

The following characteristics generally differentiate Nomad from related products:

- **Simplicity**: Nomad runs as a single process with zero external dependencies. Operators can easily provision, manage, and scale Nomad. Developers can easily define and run applications.
- **Flexibility**: Nomad can run a diverse workload of containerized, legacy, microservice, and batch applications. Nomad can schedule service, batch processing and system jobs, and can run on both Linux and Windows.
- **Scalability and High Performance**: Nomad can schedule thousands of containers per second, scale to thousands of nodes in a single cluster, and easily federate across regions and cloud providers.
- **HashiCorp Interoperability**: Nomad elegantly integrates with Vault for secrets management and Consul for service discovery and dynamic configuration. Nomad's Consul-like architecture and Terraform-like job specification lower the barrier to entry for existing users of the HashiCorp stack.

There are many relevant categories for comparison including cluster managers, resource managers, workload managers, and schedulers. There are many existing tools in each category, and the comparisons are not exhaustive of the entire space.

Due to the bias of the comparisons being on the Nomad website, we attempt to only use facts. If you find something that is invalid or out of date in the comparisons, please open an issue and we will address it as soon as possible.

Use the navigation on the left to read comparisons of Nomad versus other systems.

# nomad -help

```
Common commands:
    run         Run a new job or update an existing job
    stop        Stop a running job
    status      Display the status output for a resource
    alloc       Interact with allocations
    job         Interact with jobs
    node        Interact with nodes
    agent       Runs a Nomad agent

Other commands:
    acl             Interact with ACL policies and tokens
    agent-info      Display status information about the local agent
    deployment      Interact with deployments
    eval            Interact with evaluations
    namespace       Interact with namespaces
    operator        Provides cluster-level tools for Nomad operators
    quota           Interact with quotas
    sentinel        Interact with Sentinel policies
    server          Interact with servers
    ui              Open the Nomad Web UI
    version         Prints the Nomad version
```

# А ещё есть ECS

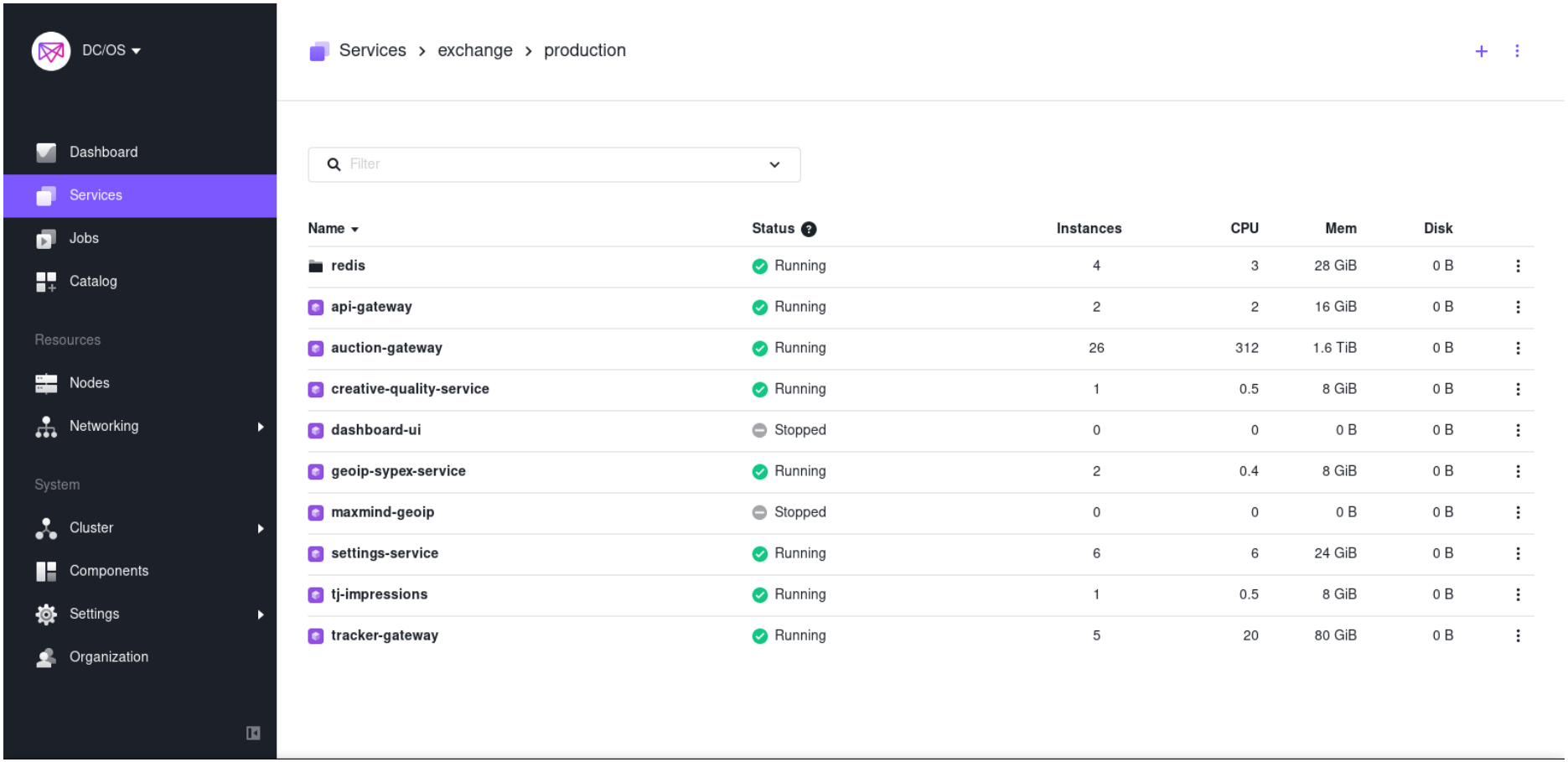
- only Docker
- only AWS
- vendor lock-in
- нафиг, пропускаем
- ...

# Время гигантов

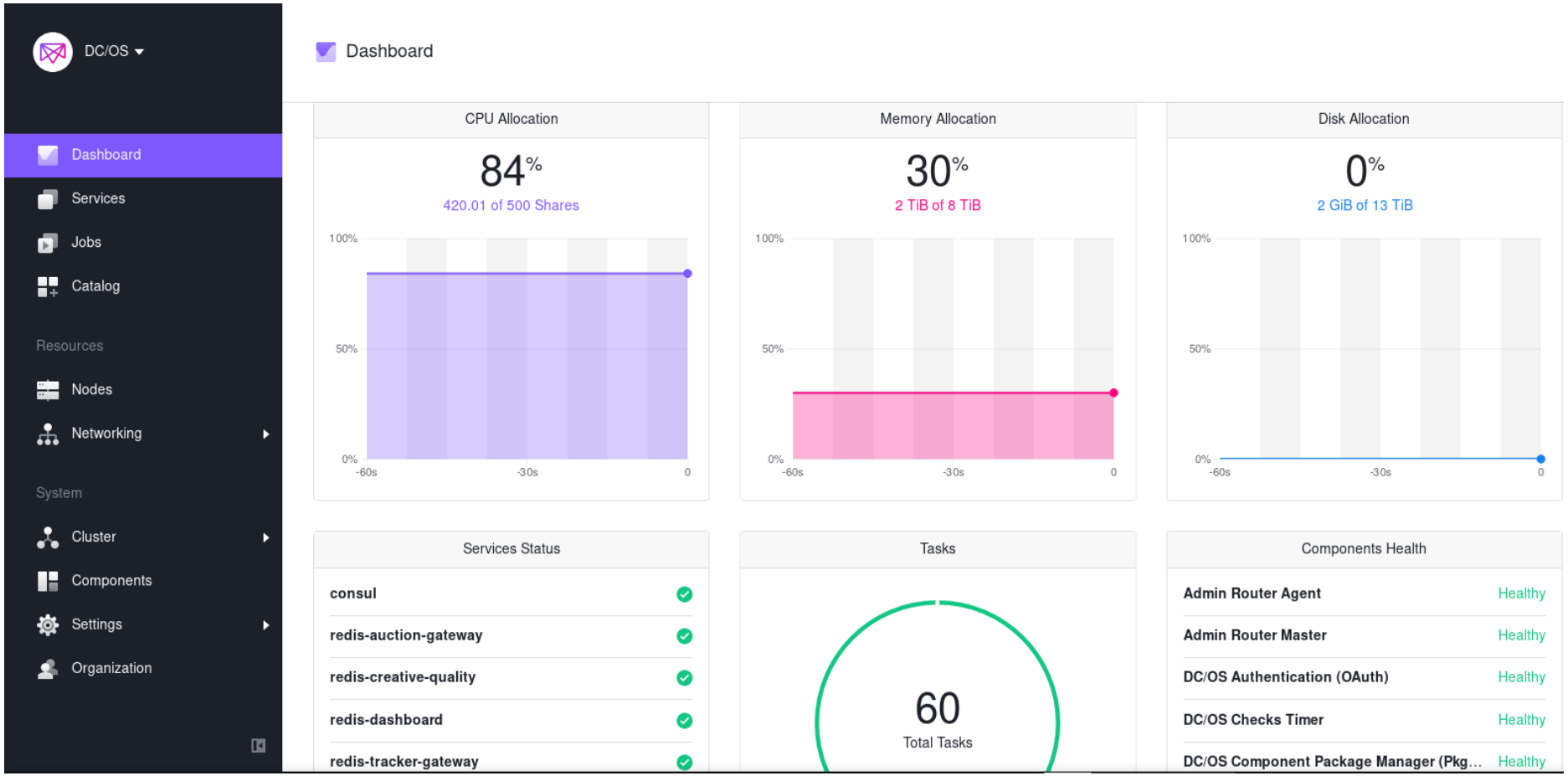
**Пара слов про DC/OS**:

- DC/OS = RHEL7/OEL7/CentOS7/CoreOS + mesos + metronome
- Docker и UCR
- ZooKeeper
- Marathon-LB и Edge-LB
- Catalog и pkgpanda, но есть проблема...
- 3 типа нод: private, public, master
- DC/OS CE vs DC/OS EE
- Ограничения DC/OS CE
- Понятные и интуитивные примитивы, но их может не хватать...
- Minuteman, Spartan и прочее сетевое...
- Отсутствие кроссдистрибутивности и гетерогенности, куча мелких ограничений
- ...

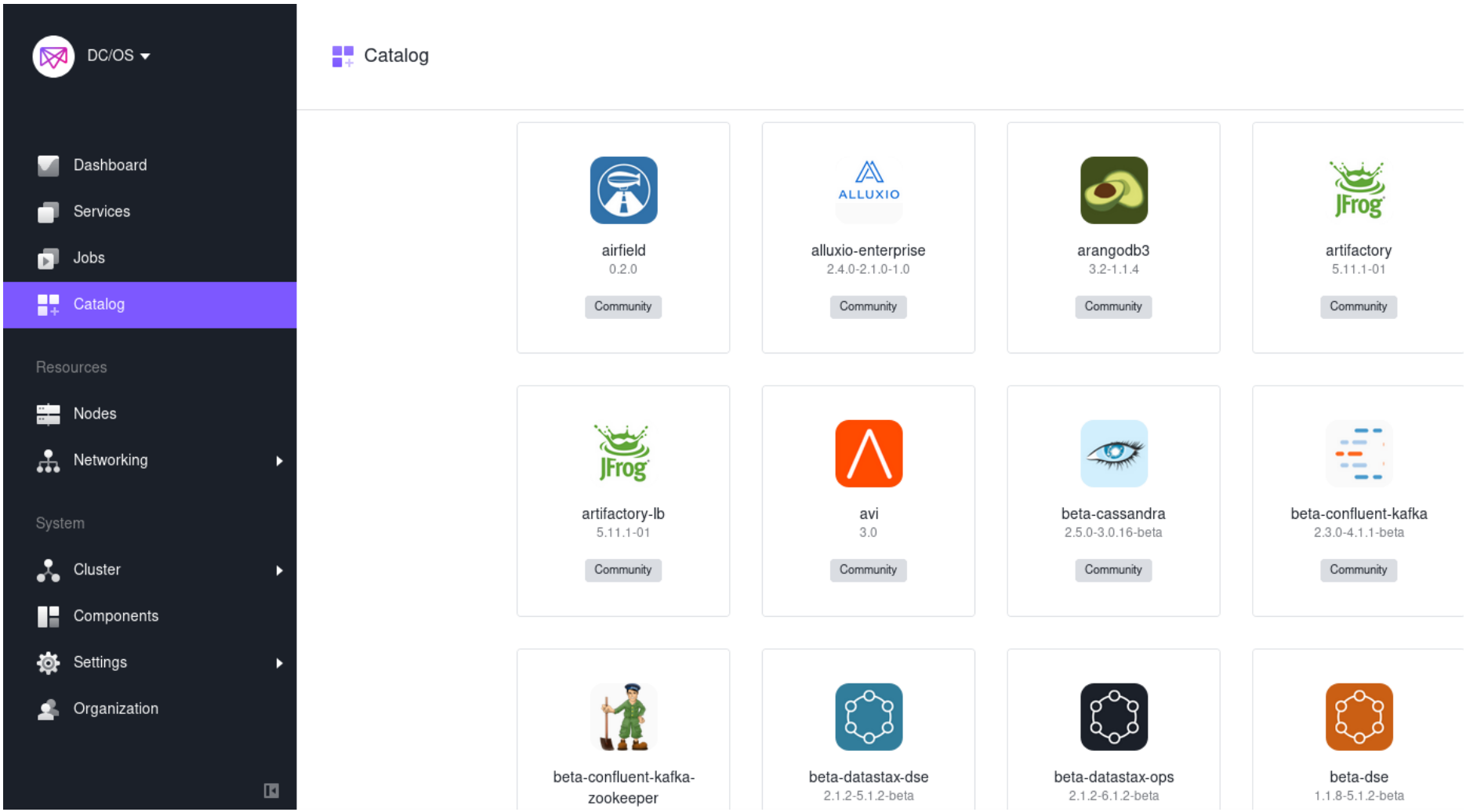
# DC/OS


DC/OS
Dashboard
Services
Jobs
Catalog
Resources
Nodes
Networking
System
Cluster
Components
Settings
Organization
Services › exchange › production
Filter
Name
Status
Instances
CPU
Mem
Disk
redis Running 4 3 28 GiB 0 B
api-gateway Running 2 2 16 GiB 0 B
auction-gateway Running 26 312 1.6 TiB 0 B
creative-quality-service Running 1 0.5 8 GiB 0 B
dashboard-ui Stopped 0 0 0 B 0 B
geoip-sypex-service Running 2 0.4 8 GiB 0 B
maxmind-geoip Stopped 0 0 0 B 0 B
settings-service Running 6 6 24 GiB 0 B
tj-impressions Running 1 0.5 8 GiB 0 B
tracker-gateway Running 5 20 80 GiB 0 B

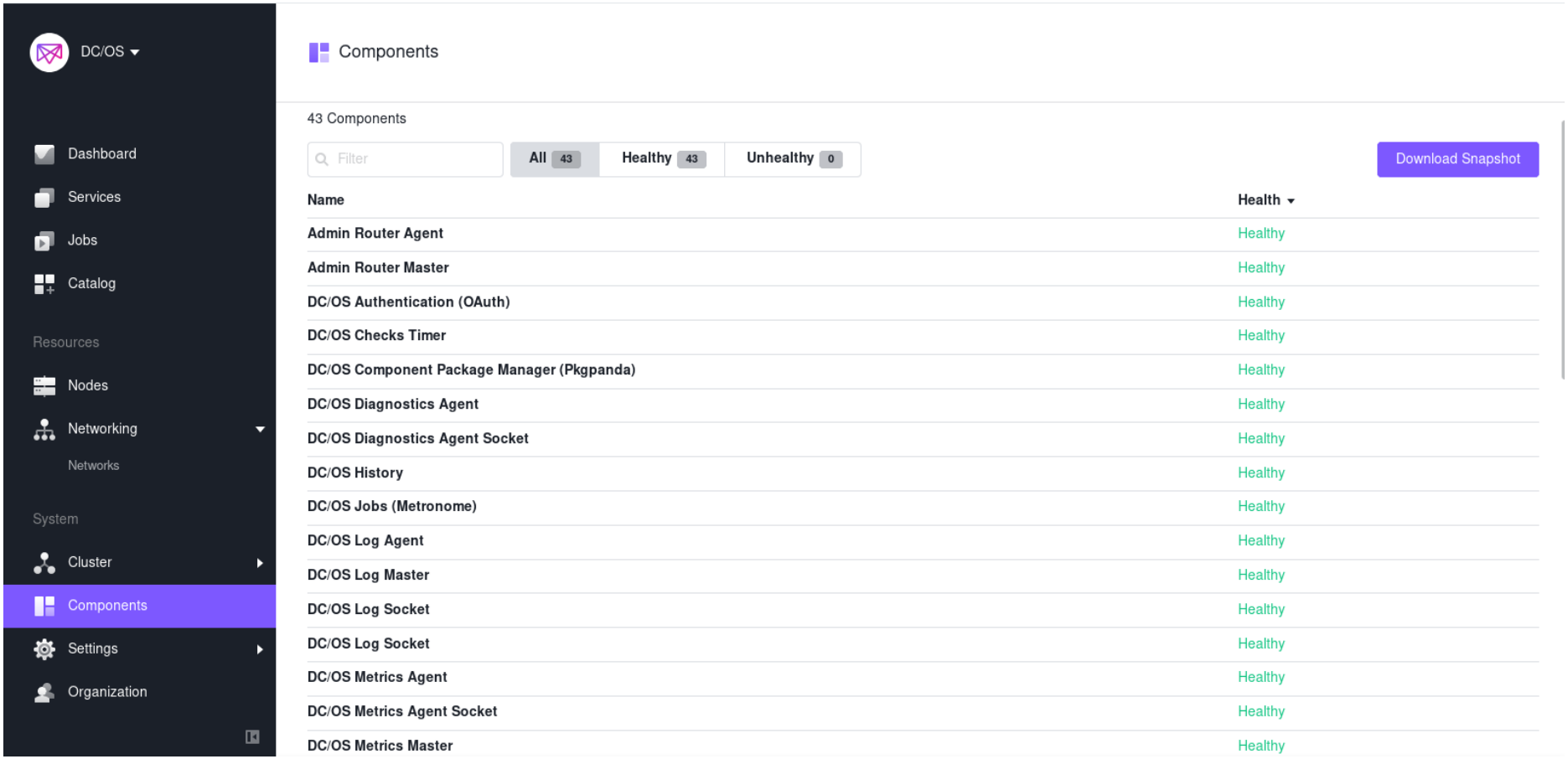
# DC/OS


DC/OS
Dashboard
Services
Jobs
Catalog
Resources
Nodes
Networking
System
Cluster
Components
Settings
Organization
Dashboard
CPU Allocation
84%
420.01 of 500 Shares
100%
50%
0%
-60s
-30s
0
Memory Allocation
30%
2 TiB of 8 TiB
Disk Allocation
0%
2 GiB of 13 TiB
Services Status
consul
redis-auction-gateway
redis-creative-quality
redis-dashboard
redis-tracker-gateway
Tasks
60
Total Tasks
Components Health
Admin Router Agent
Healthy
Admin Router Master
DC/OS Authentication (OAuth)
DC/OS Checks Timer
DC/OS Component Package Manager (Pkg…

# DC/OS


DC/OS
Catalog
Dashboard
Services
Jobs
Catalog
Resources
Nodes
Networking
System
Cluster
Components
Settings
Organization
airfield
0.2.0
Community
alluxio-enterprise
2.4.0-2.1.0-1.0
Community
arangodb3
3.2-1.1.4
Community
artifactory
5.11.1-01
Community
artifactory-lb
5.11.1-01
Community
avi
3.0
Community
beta-cassandra
2.5.0-3.0.16-beta
Community
beta-confluent-kafka
2.3.0-4.1.1-beta
Community
beta-confluent-kafka-zookeeper
beta-datastax-dse
2.1.2-5.1.2-beta
beta-datastax-ops
2.1.2-6.1.2-beta
beta-dse
1.1.8-5.1.2-beta

# DC/OS


DC/OS
Components
43 Components
Dashboard
Services
Jobs
Catalog
Resources
Nodes
Networking
Networks
System
Cluster
Components
Settings
Organization
Filter
All 43
Healthy 43
Unhealthy 0
Download Snapshot
Name
Health
Admin Router Agent
Admin Router Master
DC/OS Authentication (OAuth)
DC/OS Checks Timer
DC/OS Component Package Manager (Pkgpanda)
DC/OS Diagnostics Agent
DC/OS Diagnostics Agent Socket
DC/OS History
DC/OS Jobs (Metronome)
DC/OS Log Agent
DC/OS Log Master
DC/OS Log Socket
DC/OS Log Socket
DC/OS Metrics Agent
DC/OS Metrics Agent Socket
DC/OS Metrics Master
Healthy

# Не только DC/OS уметь в apache mesos

**У DC/OS на этом поле есть альтернативы**:

- кастомная инсталяция mesos + marathon + «обвес»
- кастомная инсталяция mesos + chronos + «обвес»
- кастомная инсталяция mesos + aurora + «обвес»
- кастомная инсталяция mesos + k8s (даже так!)
- кастомная инсталяция mesos + ... да хоть самописный framework

# И последний по счёту, но не по фичам!

**Конечно же тот самый kubernetes:**

- etcd
- docker, rkt, containerd, CRI-O, ...
- calico, canal, cilium, contiv, flannel, kube-router, multus, weave, macvlan, kube-ovn, ...
- CoreDNS, kube-dns, ...
- Ingress как универсальное API входного балансировщика
- Ingress-controller (куча реализаций)
- Примитивы на все случаи жизни...
- контроллеры и операторы
- Обширный API и набор ресурсов с возможностью расширения (см. CRD)
- kubeadm, kubespray, kops, ...
- kubectl, helm2 + tiller, helm3, werf, ansible -m k8s, ...
- k9s, stern, kubetail, ...
- RBAC
- «конструктор» и framework для container-based PaaS
- относительно высокий порог входа
- ... (задавайте вопросы)

# k8s, основные примитивы

**Cluster**

- Cluster Roles
- Namespaces
- Nodes
- Persistent Volumes
- Storage Classes

**Workloads**

- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

**Discovery and Load Balancing**

- Ingresses
- Services

**Config and Storage**

- Config Maps
- Persistent Volume Claims
- Secrets

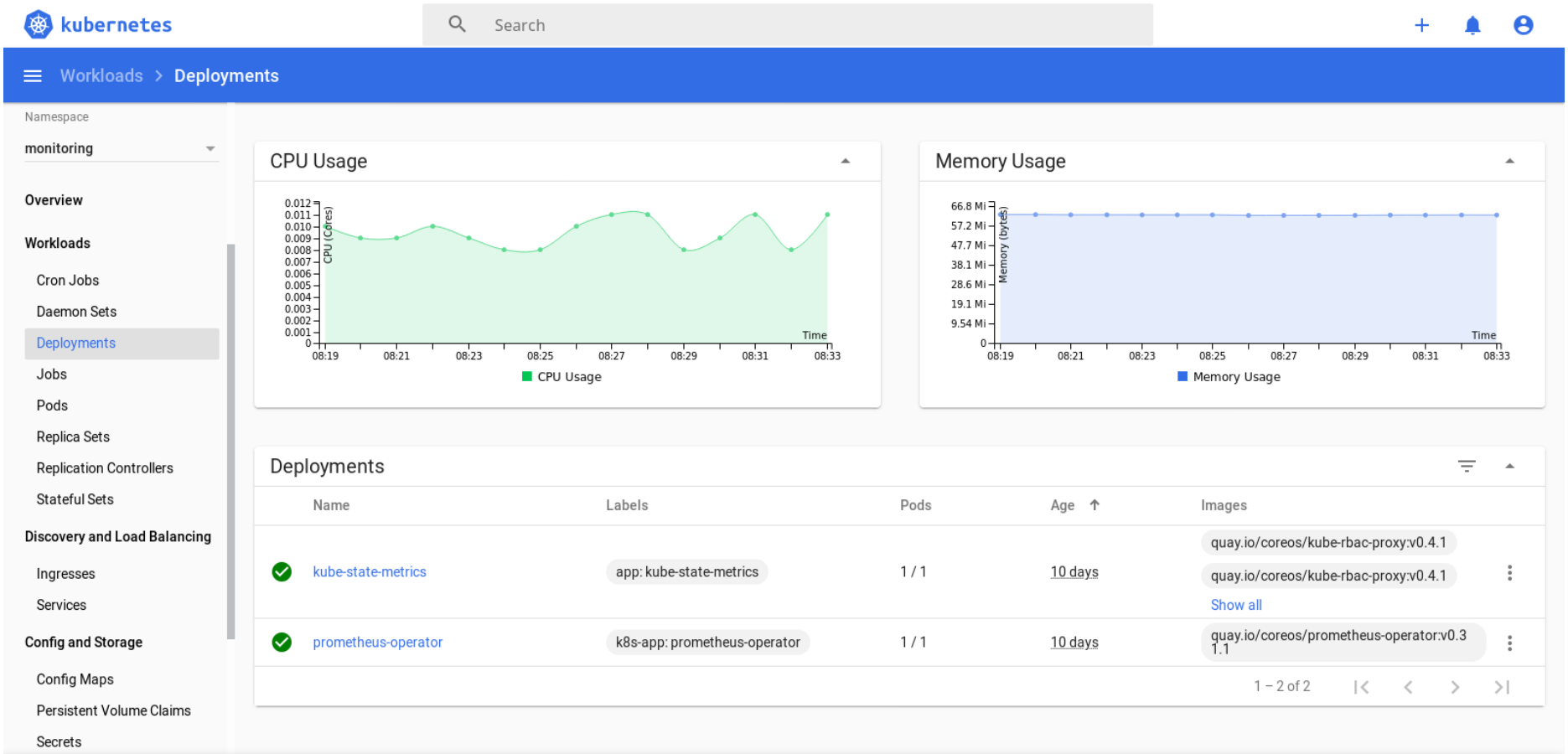
# k8s, группы API и версионность

```
admissionregistration.k8s.io/v1
admissionregistration.k8s.io/v1beta1
apiextensions.k8s.io/v1
apiextensions.k8s.io/v1beta1
apiregistration.k8s.io/v1
apiregistration.k8s.io/v1beta1
apps/v1
authentication.k8s.io/v1
authentication.k8s.io/v1beta1
authorization.k8s.io/v1
authorization.k8s.io/v1beta1
autoscaling/v1
autoscaling/v2beta1
autoscaling/v2beta2
batch/v1
batch/v1beta1
certificates.k8s.io/v1beta1
coordination.k8s.io/v1
coordination.k8s.io/v1beta1
events.k8s.io/v1beta1
extensions/v1beta1
metrics.k8s.io/v1beta1
monitoring.coreos.com/v1
networking.k8s.io/v1
networking.k8s.io/v1beta1
node.k8s.io/v1beta1
policy/v1beta1
rbac.authorization.k8s.io/v1
rbac.authorization.k8s.io/v1beta1
scheduling.k8s.io/v1
scheduling.k8s.io/v1beta1
storage.k8s.io/v1
storage.k8s.io/v1beta1
v1
```

# k8s, API ресурсы

bindings componentstatuses configmaps endpoints events limitranges namespaces nodes persistentvolumeclaims persistentvolumes pods podtemplates replicationcontrollers resourcequotas secrets serviceaccounts services mutatingwebhookconfigurations validatingwebhookconfigurations customresourcedefinitions apiservices controllerrevisions daemonsets deployments replicasets statefulsets tokenreviews localsubjectaccessreviews selfsubjectaccessreviews selfsubjectrulesreviews subjectaccessreviews horizontalpodautoscalers cronjobs jobs certificatesigningrequests leases events ingresses nodes pods alertmanagers podmonitors prometheuses prometheusrules servicemonitors ingresses networkpolicies runtimeclasses poddisruptionbudgets podsecuritypolicies clusterrolebindings clusterroles rolebindings roles priorityclasses csidrivers csinodes storageclasses volumeattachments

# k8s, конечно есть dashboard



# Выводы, итоги, замечания

- Выбирай инструмент под задачу
- Популярность и распространённость имеют значение
- Сообщество и наличие «экосистемы» имеют значение
- Пара слов про каждое решение (swarm, nomad, ECS, DCOS, k8s)
- Пара слов про мой выбор

# Спасибо за внимание.
# Да будет срач в вопросах и комментах

Вопросы
Замечания
Угрозы
Оскорбления
Предложения

kulak@itg.by
+375292751078