ООО «РусБИТех-Астра»

Платформа контейнеризации Боцман

Технические и эксплуатационные характеристики программного обеспечения.

Описание технической архитектуры программного обеспечения

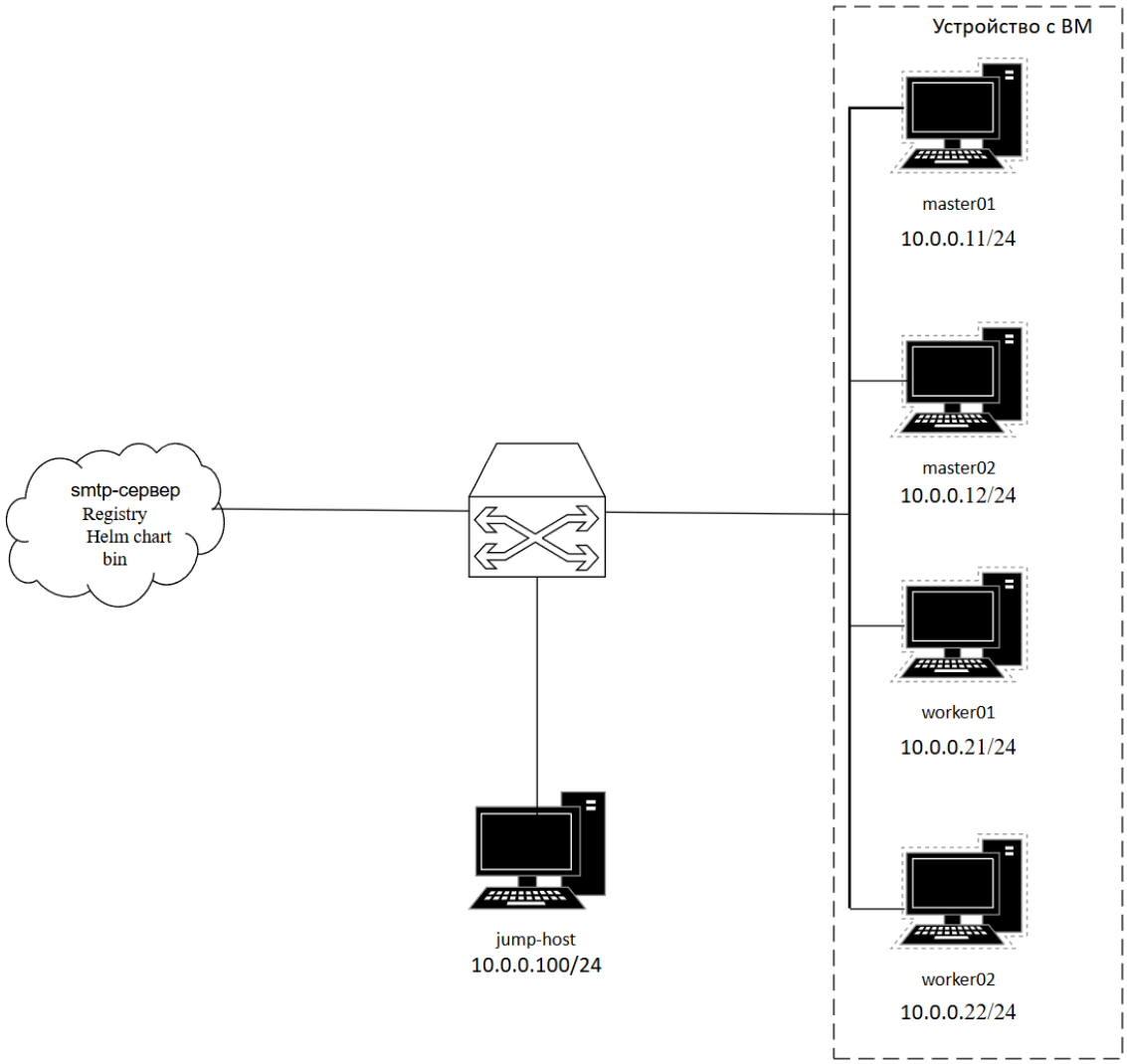
Москва, 2025

Требования к машинам, входящим в будущий кластер

- Должны быть доступны по сети между собой
- Должен быть доступ по SSH
- на машине(jump-host), с которой выполняется установка, должен присутствовать архив с установочными файлами

Пример схемы сети

На **рисунке 1** представлен пример сети в которой разворачивается Боцман.



В примере установка выполняется с машины на **рисунке 1** обозначенной jump-host, на виртуальные машины обозначенные на рисунке как **master01**, **master02**, **worker01**, **worker02**. Для разворачивания кластера Боцман для сети представленной на рисунке 1 в рамках одной машины рекомендуется выделить под виртуальные машины:

- не менее 6 ядер CPU;
- 16 ГБ ОЗУ;
- 80 ГБ дискового пространства

Все машины входящие в будущий кластер обозначенные на рисунке 1 имеют доступ по SSH между собой.

Установка

Установка выполняется с отдельной машины, имеющей доступ по сети к другим машинам будущего кластера.

1) Необходимо создать шаблон конфигурационного файла командой

```
nodus template cluster -m=2 -w=2 -o="configs/cluster_config.yml"
```

где ключ -m означает количество управляющих узлов в кластере

-w количество рабочих узлов в кластере

В результате будет создан шаблон конфигурационного файла с 2 мастер и с 2 рабочими нодами и сохранен в файл configs/cluster_config.yml

Ниже приведен пример минимального конфигурационного файла, для создания кластера из 2 мастер и 2 рабочих нод.

```
inventory:

  settings:
    ssh:
      pass: password
      port: 22
      user: username
  nodes:
    masters:
    - host:
        hostname: master01
        ip: 10.0.0.11
    - host:
        hostname: master02
        ip: 10.0.0.12
    workers:
    - host:
        hostname: worker01
        ip: 10.0.0.21
    - host:
      hostname: worker02
      ip: 10.0.0.22
```

2) Для установки кластера на основе конфигурационного файла необходимо добавить в рабочую конфигурацию Боцман.

```
nodus cluster add -c configs/cluster_config.yml my_cluster
```

где ключ -c является обязательным параметром и означает путь к конфигурационному файлу

Важное замечание, если путь содержит пробелы, то он должен быть заключен в кавычки

3) Далее необходимо добавить установочный архив из локального файла командой:

```
nodus bundle add <путь_к_архиву>
```

4) Перед установкой компонентов кластера на нодах необходимо установить и включить службу **nodusd,** выполнить это можно командой

```
nodus daemon install my_cluster -j 4 -e
```

где ключ -j означает количество параллельных процессов выполнения команды
-e включает службу **nodusd** на узлах кластера после ее установки

5) Установка и настройка компонентов кластера производится с помощью команды

```
nodus cluster install my_cluster -l -j 4
```

где ключ -l включает регистрацию событий в процессе установки кластера с помощью службы **nodusd**
-j означает количество параллельных процессов выполнения команды

Управление с помощью kubectl

Для последующего управления кластером с помощью клиента kubectl после успешной установки компонентов кластера необходимо выполнить одно из следующих действий:

1) Скопировать конфигурационный файл kubectl в расположение конфигурации Kubernetes, выполнив команды:

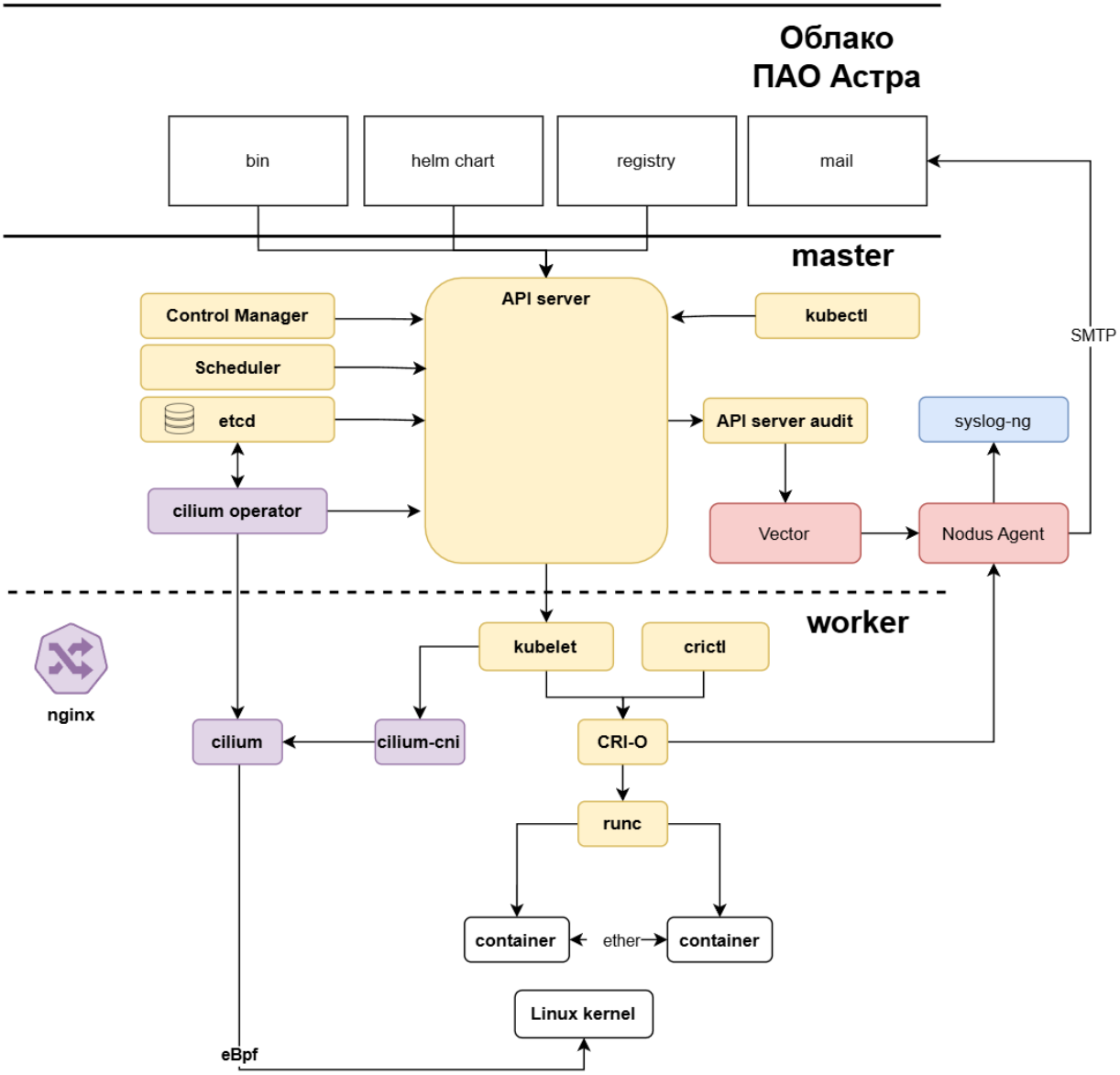
```
mkdir -p $HOME/.kube
sudo cp -i $HOME/.config/nodus/clusters/home/kubeconfigs/admin.conf $HOME/.kube/\
config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

2) Создать переменную среды KUBECONFIG и указать каталог расположения конфигурационного файла kubectl в качестве ее значения, выполнив команду:

```
export KUBECONFIG=$HOME/.config/nodus/clusters/home/kubeconfigs/admin.conf
```

Техническая архитектура платформы контейнеризации Боцман

Операционная система: Astra Linux Special Edition, как среда исполнения


Облако
ПАО Астра
bin
helm chart
registry
mail
master
API server
Control Manager
Scheduler
etcd
cilium operator
kubectl
API server audit
syslog-ng
Vector
Nodus Agent
SMTP
worker
kubelet
crictl
nginx
cilium
cilium-cni
CRI-O
runc
container
ether
container
Linux kernel
eBpf

Платформа состоит из следующих основных частей:

kubelet: Это агент, который работает на каждом узле кластера и отвечает за обеспечение работоспособности контейнеров в Pod'ах.
kubectl: Это CLI инструмент, предназначенный для взаимодействия с сервером API Kubernetes. Он позволяет создавать, обновлять и удалять компоненты кластера Kubernetes.
kube-apiserver: Это главная компонента управления кластером Kubernetes, предназначенная для обработки REST операций и обновления объектов в кластере Kubernetes.
kube-proxy: Этот компонент обеспечивает реализацию сетевого прокси и балансировщика нагрузки в Kubernetes и отвечает за маршрутизацию трафика к подходящим контейнерам.
kube-controller-manager: Это компонента, которая запускает контроллеры кластера Kubernetes, такие как контроллеры узлов, репликаций и так далее.
kube-scheduler: Он отвечает за распределение Pod'ов по узлам кластера в зависимости от различных метрик, таких как ресурсы узла и требования по преференциям местонахождения.
etcd: Это база данных типа ключ-значение, используемая для хранения всех данных кластера, она обеспечивает сохранность и доступность данных.
cilium: Это решение для сетевых политик и маршрутизации в Kubernetes на основе технологии eBPF.
cri-o: Это реализация интерфейса контейнеров для Open Containers Initiative (OCI), позволяющая запускать любые контейнеры, соответствующие стандарту OCI.
nginx-ingress: Это прокси и балансировщик нагрузки для входящего трафика в Kubernetes, который использует nginx в своей основе.
runc: Это бинарный файл, который используется в качестве рантайма OCI для запуска контейнеров.
kubeadm: Этот инструмент предназначен для инициализации кластера Kubernetes, он обеспечивает минимальную функциональность для запуска поставщика на основе Kubernetes.

Терминология

"Мастер" (control plane node) управляющий узел.
Отвечает за централизованное управление кластером и состоит из компонентов:
Воркер" (или worker node) - узел (node) - совокупность **общих компонентов**, выполняемых на физическом или виртуальном сервере **с предустановленной операционной**

системой, количество физических или виртуальных процессоров которого не превшает 16 (шестнадцати), и выполняющих задачу запуска и управления контейнерами, на основе инструкций, полученных от управляющих узлов (control plane nodes). Воркер-узлы содержат все необходимые компоненты для управления сетью контейнеров, запускаемых на них приложений, и обеспечения их взаимодействия с другими узлами в кластере.

Общие компоненты для **"Воркера" и "Мастера"**:

1. **Kubelet**: агент, который работает на каждом воркер-узле, отвечает за запуск, остановку и управление контейнерами, которые были назначены узлу (через поды) управляющими узлами.
2. **Container runtime**: среда выполнения контейнеров, которая необходима для запуска контейнеров.
3. **Cilium:** сетевой плагин для K8s, основной задачей которого, является подключение интерфейсов сети к контейнерам и управление IP-адресами. Использует технологию eBPF для предоставления более продвинутых функций управления сетевым трафиком и безопасности.