

Rootless Kubernetes. Плюсы и минусы

О себе

- 80-е
 - ЕС ЭВМ, перфокарты
 - ОС ДЕМОС (UNIX)
 - Data General, Австрия
 - SCO UNIX - 220ix
- 90-е
 - Узел Relcom, Slackware, Debian, Linux Yes, переход на Linux
 - База Терем, Пермская товарная биржа- распределенная система торгов, АС Пенсия (SCO UNIX), перевод на Linux — распределенная система обмена информацией и создаваемого ПО, 1993-2018
 - Ведение новостей по Linux
- 0-е
 - Интернет провайдер на базе дистрибутива ALTLinux
 - Установка Linux в школах Пермского края, Поносов
- 10-е
 - Большие данные (hadoop, Solr, clickhouse, ...)
- 20-е
 - ALTLinux, Базальт Отдел Виртуализации, защищенные решения kubernetes
 - podsec, ALT-Orchestra/talos

3 Поверхность атаки на кластер



- Доступ контейнера к HOST-системе путём монтирование каталогов HOST-системы с получением привелегий root
- Повышение привелегий контейнера
- Прямой доступ к узлу кластера через ssh
- Ошибки в настройках RBAC
- Доступ к токенам приложений
- ...

Уменьшение поверхности атаки на кластер



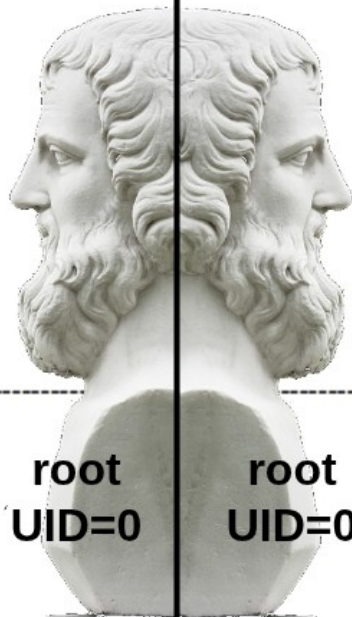
- rootless kubernetes (usernetes/podsec-k8s):
 - ~~Доступ контейнера к HOST-системе путем монтирование каталогов HOST-системы с получением привелегий root~~
 - ~~Повышение привелегий контейнера~~
- **исключение доступа по ssh (talos/ALT Orchestra):**
 - ~~Прямой доступ к узлу кластера через ssh~~
- ...

**podsec-k8s (usernetes) —
rootless кластер**

Однотеликий янус - rootfull

HOST система

```
USER=root  
-kubelet---12*[kubelet]  
-common---kube-scheduler---7*[kube-scheduler]  
-common---kube-controller---4*[kube-controller]  
-common---etcd---8*[etcd]  
-common---kube-apiserver---8*[kube-apiserver]  
-common---kube-proxy---5*[kube-proxy]  
-common---flanneld---8*[flanneld]  
-2*[common---coredns---7*[coredns]]  
-crio---10*[crio]  
-2*[common  
-...
```



root
UID=0

root
UID=0

rootfull POD's

-nginx---2*[nginx]]

Опасность rootfull

HOST система

```
apiVersion: apps/v1
kind: Deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          volumeMounts:
            - mountPath: /hostRoot
              name: host-root
      volumes:
        - name: host-root
          hostPath:
            path: /
```

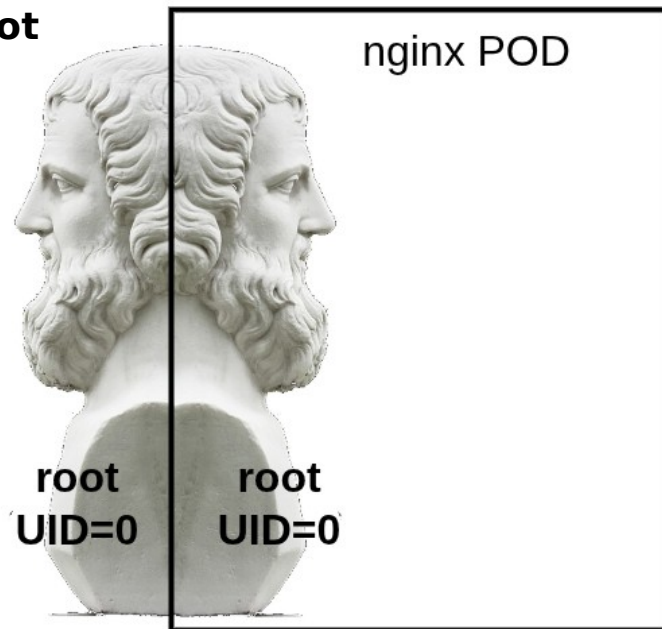
Запуск HOST-команд под root

```
# kubectl apply -f deployment.yaml
# kubectl exec -i pod/nginx-... -- \
  rm -rf /hostRoot/...
```

Запуск вредоносного ПО

с правами root

```
# kubectl run -i \
  --image <образ_с_вредоносным_ПО> \
  -- <вредоносное_ПО>
```

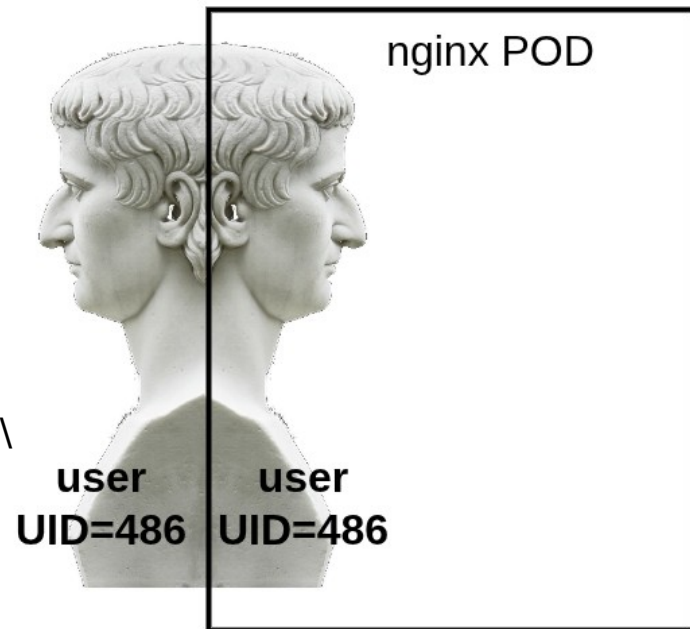


Использование не-root образов

```

apiVersion: apps/v1      # # Запуск HOST-команд под root HOST система
kind: Deployment
spec:
  selector:
    matchLabels:
      app: nginx          # # Процесс запускается с правами
                        # # пользователя с ID=486
                        # kubectl apply -f deployment.yaml
                        # kubectl exec -i pod/nginx-... -- rm -rf
                        # /hostRoot/...
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx        # # Запуск вредоносного ПО
                        # # с правами пользователя
                        # # UID=486
    spec:
      containers:
        - name: nginx      # kubectl run -i \
          image: nginx:1.14.2 --image <образ_с вредоносным_ПО>\
          volumeMounts:    -- <вредоносное_ПО>
            - mountPath: /hostRoot
              name: host-root
      volumes:
        - name: host-root
          hostPath:
            path: /

```



Поддержка rootless
в нативном kuber \geq v1.33

Запуск POD в rootless-режиме

HOST система

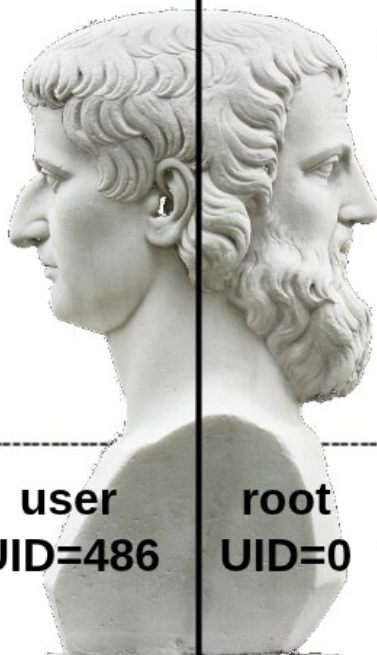
```
USER=root  
-kubelet---12*[kubelet]  
-common---kube-scheduler---7*[kube-scheduler]  
-common---kube-controller---4*[kube-controller]  
-common---etcd---8*[etcd]  
-common---kube-apiserver---8*[kube-apiserver]  
-common---kube-proxy---5*[kube-proxy]  
-common---flanneld---8*[flanneld]  
-2*[common---coredns---7*[coredns]]  
-crio---10*[crio]  
-2*[common-----  
-...
```

user
UID=486

nginx POD

nginx---2*[nginx]]

root
UID=0



Особенности rootless

HOST система

```
apiVersion: apps/v1
kind: Deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          volumeMounts:
            - mountPath: /hostRoot
              name: host-root
          volumes:
            - name: host-root
              hostPath:
                path: /
```

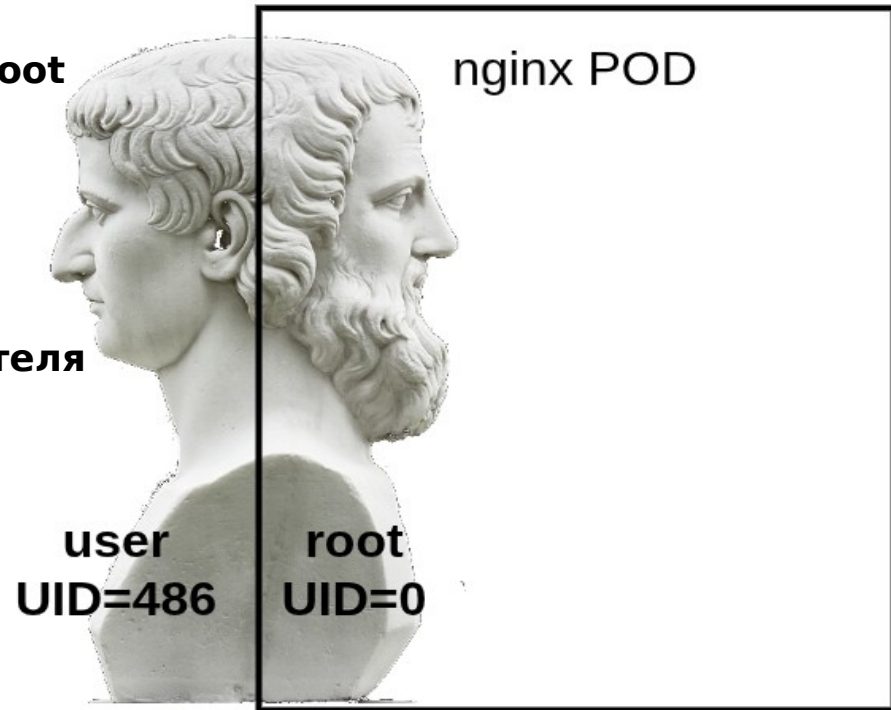
Запуск HOST-команд НЕ под root

```
# kubectl apply -f deployment.yaml
# kubectl exec -i pod/nginx-... -- \
  /hostRoot/...
```

Запуск вредоносного ПО

#с правами обычного пользователя

```
# kubectl run -i \
  --image <образ_с вредоносным_ПО> \
  -- <вредоносное_ПО>
```



Дерево процессов rootfull kuber

Контейнеры (POD'ы) основных сервисов kubernetes вызываются с правами root.

Пользовательские контейнеры (POD'ы) в зависимости от указаниях в манифестах разворачивания могут вызываться с правами root, обычного пользователя или с версии v1.33 под указанным пользователем rootless-окружении.

```
USER=root
|-kubenet--12*[{kubenet}]
|-conmon--kube-scheduler--7*[{kube-scheduler}]
|-conmon--kube-controller--4*[{kube-controller}]
|-conmon--etcd--8*[{etcd}]
|-conmon--kube-apiserver--8*[{kube-apiserver}]
|-conmon--kube-proxy--5*[{kube-proxy}]
|-conmon--flannel--8*[{flannel}]
|-2*[conmon--coredns--7*[{coredns}]]
|-crio--10*[{crio}]
|-conmon--<root_kubernetes_POD>
|-conmon--<user_kubernetes_POD>
|-conmon--<rootless_kubernetes_POD>
|-...
```


Не зарекайся

- Медицина: «Не бывает абсолютно здоровых людей. Есть недообследованные»
- Программирование: «Не бывает абсолютно защищенных программ. Есть недопроанализированные»

Dockerfile:

```
FROM ubuntu  
USER 4611686818427387984  
...
```


**Полный rootless
(usernetes/u7s)
podsec
(PODman SECurity)**

Состояние на март 2023

- реализованы podman, kubernetes сервисы (команды), kubernetes docker-образы 1.26, разворачиваемые стандартно в rootfull режиме
- Требования ФСТЕК к июню 2023:
 - обеспечить запуск podman-контейнеров, разворачивание через kubeadm kubernetes-образов в rootless режиме
 - Три класса пользователей:
 - Администраторы кластера
 - Создатели образов
 - Пользователи образов

- Пользователи docker-образов имеют право использовать только подписанные образы от создателей образов. Любые «левые» схемы получения образов (podman load, podman build, ...) запрещены
- Система должна мониторить
 - попытки изменения прав,
 - несанкционированных изменений в работающих контейнерах,
 - использование «левых» образов, или образов с критическими CVE уязвимостями,
 - ...

Дилемма

Самая сложная проблема - обеспечить разворачивание через kubeadm kubernetes-образов в rootless режиме по причине отсутствия на тот рабочего момент решения.

Варианты:

- Подождать когда появится решение и перенести его в ALTLinux.
- Реализовать собственное решение.

Варианты rootless kuber

	kind	minikube	k3s	Usernetes Gen1	Usernetes Gen2
Runtime environment	Docker compose	Docker compose	Docker compose	Docker compose	kubeadm
Containers Policy	Experiment	No	Experiment	Yes	Yes
Podman (CRI-O)	Experiment	No	Experiment	Yes	Yes
Namespace support by	docker podman	docker podman	rootlesskit	nsenter rootlesskit	nsenter rootlesskit
Language	go	go	go	bash	bash
Multihost	No	No	Yes	No	Yes
Docker (containerd)	Yes	Yes	Yes	Yes	Yes

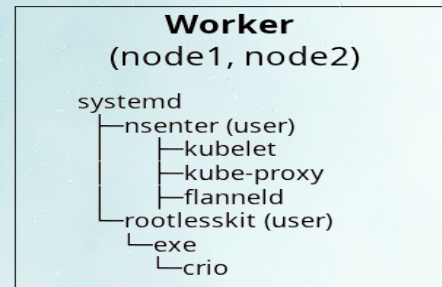
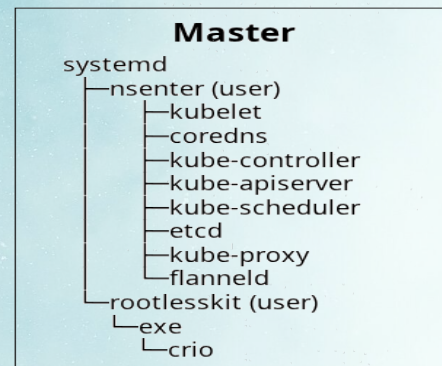
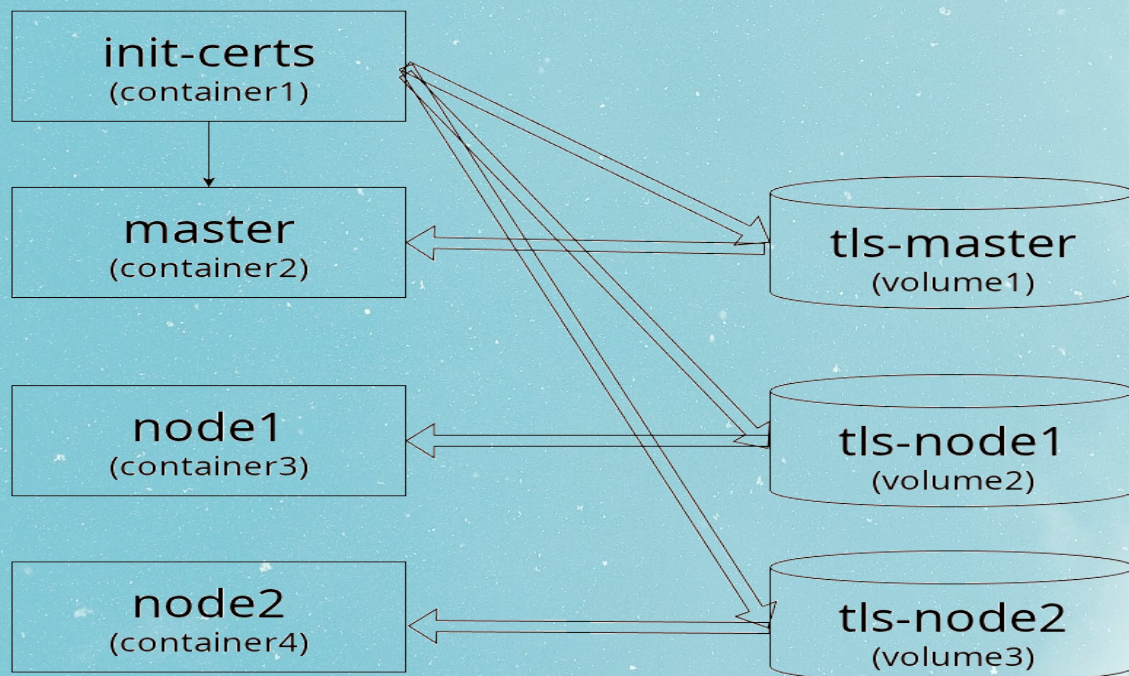
История

- Имеет две версии — Generation 1, Generation 2.
- Generation 1 — разворачивание rootless kuber в docker-стеке в рамках одного узла.
- Generation 2 — разворачивание rootless kuber на множестве узлов.
- podsec-k8s — отталкивался от Gen1, поддерживает нативное разворачивание kuber через kubadm на на множестве узлов, был выпущен на 3 месяца ранее (июнь 2023) usernetes Gen2 (сентябрь 2023).

Лицензия GPL V2.1.

Инициализация кластера USERNETES(Gen1)

Сервер с dockerd



PODSEC VS USERNETES

	Usernetes Gen1	Usernetes Gen2	podsec
Runtime environment	Docker Compose	kubeadm	kubeadm
Multihost	No	Yes	Yes
Containers Policy	No/ Yes	No/Yes	Yes
Engine	Containetd CRI-O	Containetd CRI-O	CRI-O
Start kuber services AS	Systemd services	PODs	PODs
Monitoring	No	No	Yes
Local Registry	No	No	Yes
Image Signing	No	No	Yes

Инициализация кластера USERNETES(Gen2)

```
# Set ENGINE
export CONTAINER_ENGINE=podman

# Bootstrap a cluster
make up
make kubeadm-init
make install-flannel

# Enable kubectl
make kubeconfig
export KUBECONFIG=$(pwd)/kubeconfig
kubectl get pods -A

# Multi-host
make join-command
scp join-command another-host:~/usernetes
ssh another-host make -C ~/usernetes up kubeadm-join
make sync-external-ip
```


Преимущества решения **podsec/usernetes**

- **usernetes/rootless**

- Все POD'ы включая базовые работают в rootless окружении
- Даже при наличии уязвимостей в базовых образах и образов пользователей злоумышленник не может нарушить работу компонентов узла и файлов HOST-системы.

- **podsec**

- Поддержка создания безопасного окружения
 - Строгие политики доступа к образам
 - Создание пользователей различного класса (разработчики образов, пользователи образов, администраторы безопасности)
 - Подъем необходимых сервисов (регистратора, сервера доступа к открытым подписям создателей образов).
 - Мониторинг уязвимостей узлов kubernetes-кластера.

Команды работы с namespace пользователя

nsenter - команда запуска программы в пользовательском namespace:

- **mount** — изолированное монтирование;
- **UTS** — изолированные hostname и domainname;
- **IPC** — изолированные межпроцессные каналы;
- **network** — изолированные сетевые интерфейсы, iptables, ...;
- **PID** — изолированное дерево PID-процессов;
- **user** - изолированные UID, GID, capabilities;
- **cgroup** — изолированный Control Group (ограничение, учет, изоляция ресурсов группы процессов);
- **time** — изолированные CLOCK_MONOTONIC и CLOCK_BOOTTIME.

Команды работы с namespace пользователя

rootlesskit — реализация fake root (псевдо root) в пользовательском namespace:

- **Mount:**
 - **--copy-up** <каталог>
 - **--propagation** [rprivate, rslave]
- **Network:**
 - **--net** [slirp4net, host, pasta, vpnkit, ...]
 - **--cidr** IP/Mask
 - **--ifname** [tap0, ethN]
- **Port:**
 - **--port-driver** [none, builtin, slirp4nets]
 - **--publish** <IP>:portIN:portOUT/[tcp, udp]
- **Process:**
 - **--pidns**, **--cgroupns**, **--utsns**, **--ipcns**

Команды работы с namespace пользователя

rootlessctl — rootesskit API клиент:

- **list-ports;**
- **add-ports;**
- **remove-ports;**
- **Info.**

Особенности реализации podsec-k8s

В переменную PATH добавляется каталог скриптов, замещающих команды kubernetes: `export PATH=/usr/libexec/podsec/u7s/bin/:$PATH`

Содержимое каталога `/usr/libexec/podsec/u7s/bin`:

```
crio.sh  
init-crio.sh  
kubeadm -> /usr/bin/podsec-u7s-kubeadm  
_kubeadm.sh  
kubeadm.sh  
_kubelet.sh  
kubelet.sh  
nsenter_u7s  
rootlessctl  
rootlesskit.sh  
systemctl  
u7sinit.sh  
u7s-start-stop.sh
```

Команды (скрипты) *kubeadm*, *rootlessctl*, *systemctl* замещают аналогичные системные команды, реализуют собственный интерфейс и вызывают в итоге системные.

Скрипты *_kubeadm.sh*, *_kubelet.sh* вызываются в пользовательском namespace.

Скрипт *kubeadm.sh* вызывает нативную команду `/usr/bin/kubeadm` в

Порядок инициализации узла (kubeadm init, join)



Двуликий янус - rootless

HOST система

USER=root

| - kubelet.sh -- nsenter_u7s
| - rootlesskit.sh -- rootlesskit -- +

USER=u7s-admin

| - exe

| - slirp4netns

u7s-admin
UID=453

root
UID=0

rootless POD's

----- nsenter -- kubelet.sh -- kubelet -- 12*[{kubelet}]
----- common -- kube-scheduler -- 7*[{kube-scheduler}]
----- common -- kube-controller -- 4*[{kube-controller}]
----- common -- etcd -- 8*[{etcd}]
----- common -- kube-apiserver -- 8*[{kube-apiserver}]
----- common -- kube-proxy -- 5*[{kube-proxy}]
----- common -- flanneld -- 8*[{flanneld}]
----- 2*[common -- coredns -- 7*[{coredns}]]
----- crio -- 10*[{crio}]
----- ...
----- 2*[common -- nginx -- 2*[nginx]]

Особенности podsec

- Используется только podman, cri-o решения (не containerd), поддерживающее политику доступа к образам.
- Как и в usernetes Gen1/2 контейнеры (POD'ы) основных сервисов kubernetes вызываются в rootless-окружении обычного пользователя (в podsec-k8s — пользователь u7s-admin). Пользовательские контейнеры (с пользователями root и обычными пользователями) также попадают в rootless-окружение.
- Параметризация разворачивания через переменные среды (docker-регистратор, версия kuber, ...).
- На master-сервере кроме kubernetes располагаются:
 - Пользователи группы podman_dev, имеющие право скачивать любые образы, создавать образы для локального регистратора, подписывать и помещать их в локальный регистратор.
 - Остальные пользователи могут использовать только подписанные образы локального регистратора.
 - WEB-сервер для доступа к открытым ключам пользователи группы podman_dev.
 - Регистратор подписанных образов.
 - Сервис trivy мониторинга уязвимостей
- Автоматически настраиваются политики доступа к образам
- Обеспечивает мониторинг уязвимостей и функционирования контейнеров кластера.

Набор RPM-пакетов podsec (PODman SEcurity)

- podsec — набор команд создания rootless-окружение (регистратор, политики доступа, создание ролевых пользователей, ...)
- podsec-k8s — набор команд по разворачиванию нативного rootless k8s кластера.
- podsec-k8s-rbac — набор команд по настройке и мониторингу RBAC.
- podsec-inotify — набор команд для мониторинга узлов кластера.
- podsec-k8s-upgrade — обновление версий кластера (в разработке)

Недостатки решения `podsec/usernetes`

Из за использования сервиса поддержки rootless-сети `slirp4nets` для обеспечения оверлейной сети `kubernetes` пока можно использовать только CNI `flannel`. Из за трансляции адресов в `slirp4nets` есть проблемы с применением «продвинутых» CNI.

Возможные пути решения:

- Анализируются возможности использования других сервисов поддержки rootless-сети
- Ведутся работы по использованию в качестве rootless CNI `cilium`.

Код Opensource (GPLv2). Разработчики, тестеры Welcome.

Инициализация кластера podsec- k8s с доступом в Интернет

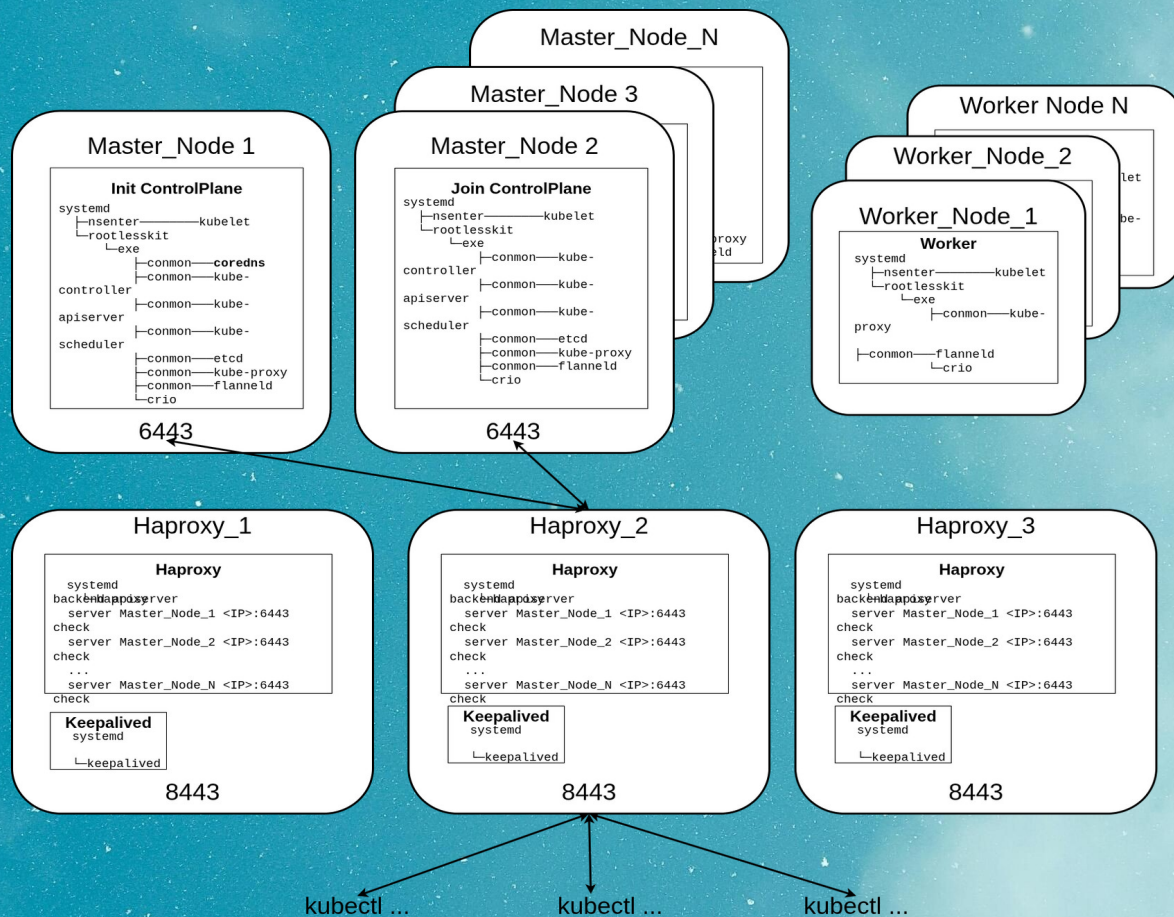

```
# Environment
export U7S_KUBEVERSION= # версия kubernetes (v1.26.9, v1.27.7, ...);
export U7S_REGISTRY= # registry.k8s.io, registry.altlinux.org,
                    # registry.local
export U7S_PLATFORM= # k8s-c10f2, k8s-c10f1 , k8s-p10, k8s-sisyphus, ...
```

```
# Master ControlPlane
export PATH=/usr/libexec/podsec/u7s/bin/:$PATH
kubeadm init -v 9 ...
```

```
# ControlPlane
export PATH=/usr/libexec/podsec/u7s/bin/:$PATH
kubeadm join xxx.xxx.xxx.xxx:6443 \
--token ... \
--discovery-token-ca-cert-hash sha256:... \
--control-plane
```

```
# Worker
export PATH=/usr/libexec/podsec/u7s/bin/:$PATH
kubeadm join xxx.xxx.xxx.xxx:6443 \
  --token ...
```


Инициализация кластера podsec-k8s с haproxy и keepalived



Политики доступа к docker-образам (podman, CRI-O)

Использование политик доступа

/etc/containers/policy.json:

```
{
  "default": [
    {
      "type": "reject"
    }
  ],
  "transports": {
    "docker": {
      "registry.local": [
        {
          "type": "signedBy",
          "keyType": "GPGKeys",
          "keyPath": "/var/sigstore/keys/imagemaker.pgp"
        }
      ]
    }
  }
}
```

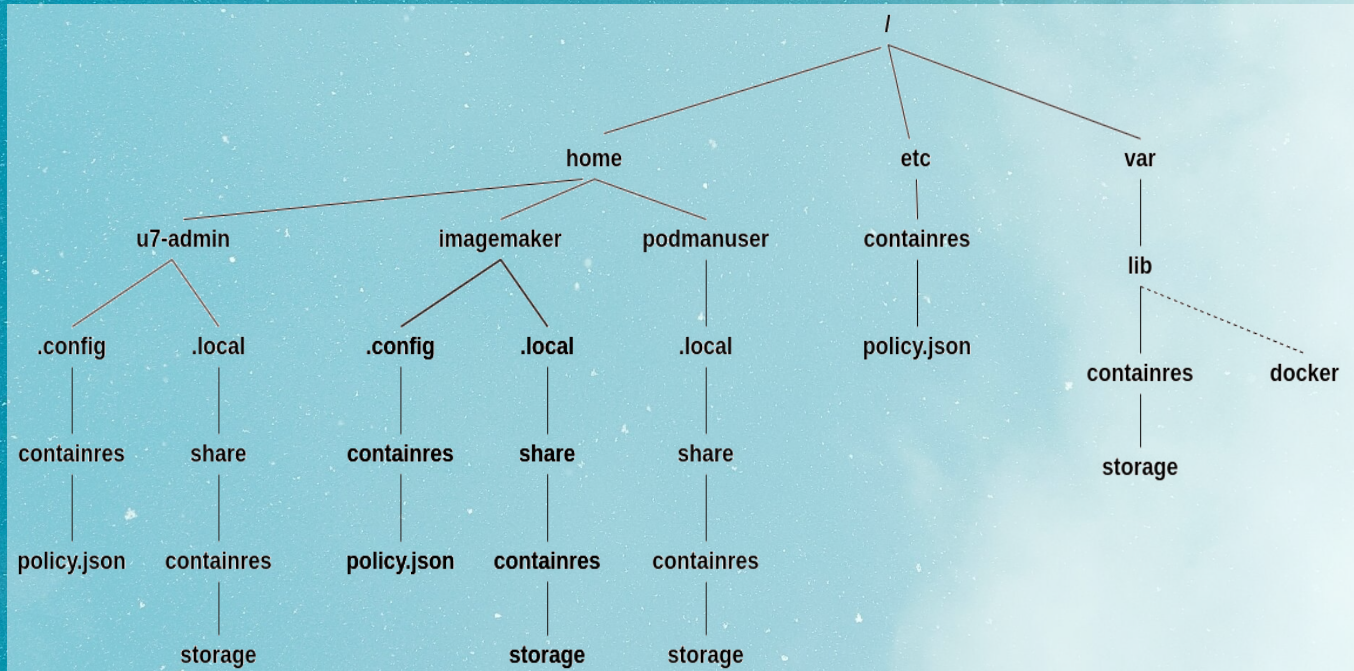
/etc/containers/registries.d/default.yaml

```
default-docker:
  sigstore: http://registry.local:81/sigstore/
```


3 Ограничение прав для доступа образам

По умолчанию всем пользователям назначаются политики из `/etc/containers/policy.json`.

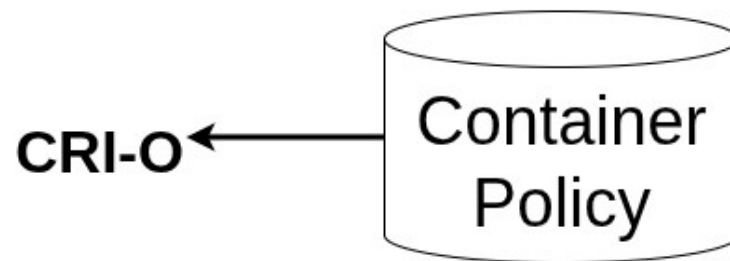
Для пользователей с повышенными правами принадлежащим группе `podman_dev` права описываются в файле `~/.config/containers/policy.json`



Инициализация кластера podsec- k8s в защищенном режиме

Master ControlPlane: Настройка политик доступа

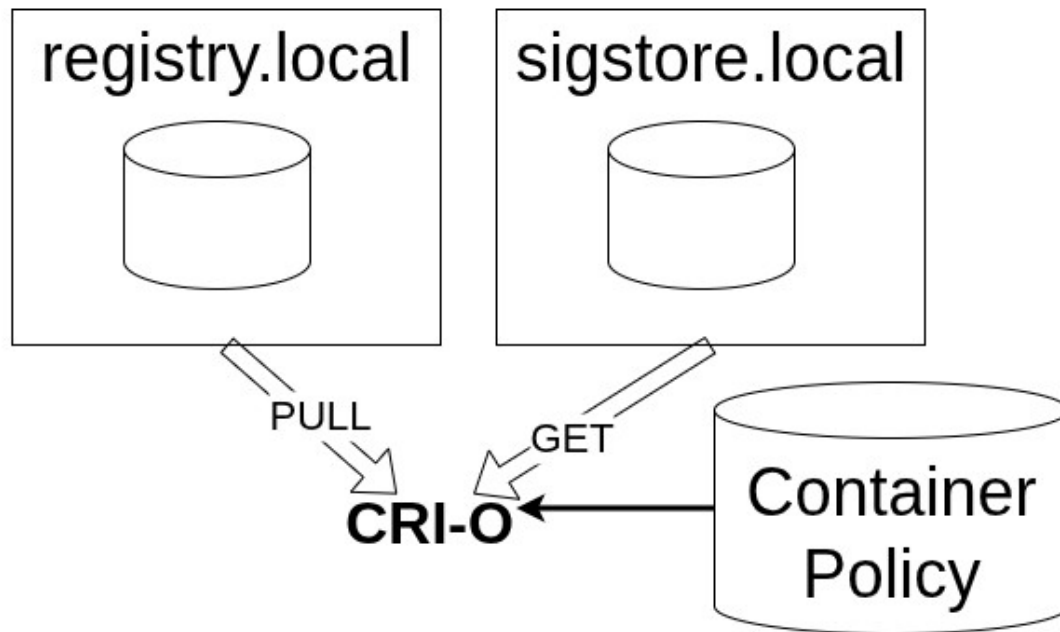
```
# podsec-create-policy
```



Master ControlPlane

Master ControlPlane: Создание и настройка локального регистратора и WEB-сервера

```
# podsec-create-policy  
# podsec-create-services
```

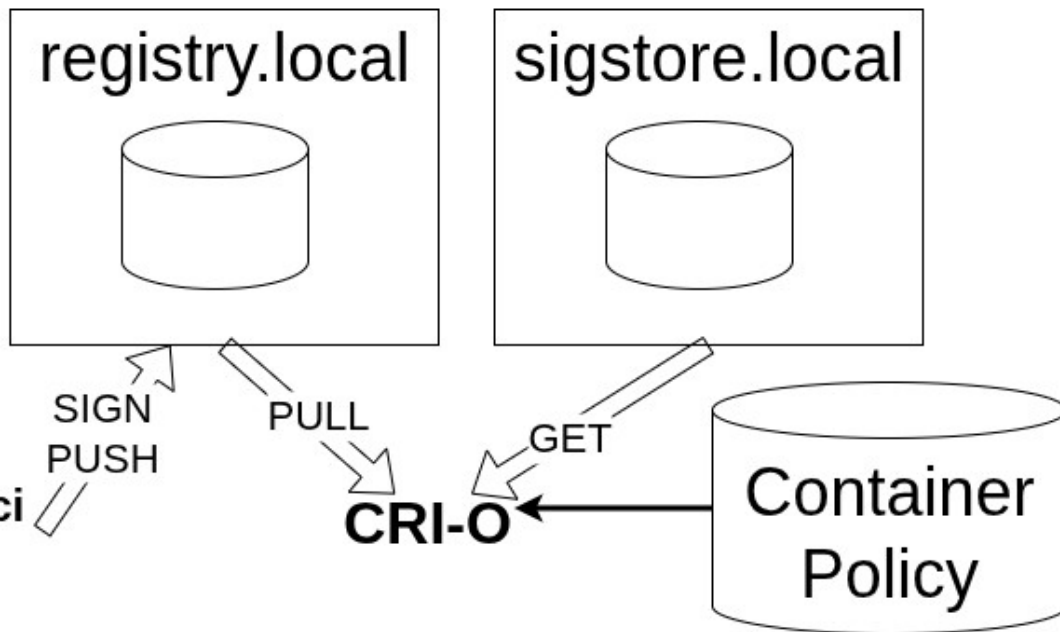


Master ControlPlane

Master ControlPlane: Создание пользователя(ей) для создания образов

```
# podsec-create-policy  
# podsec-create-services  
# podsec-create-imagemakeruser
```

```
imagemaker$ podsec-load-sign-oci  
imagemaker$ podman ...
```



Master ControlPlane

Master ControlPlane: Разворачивание Master-узла kubernetes

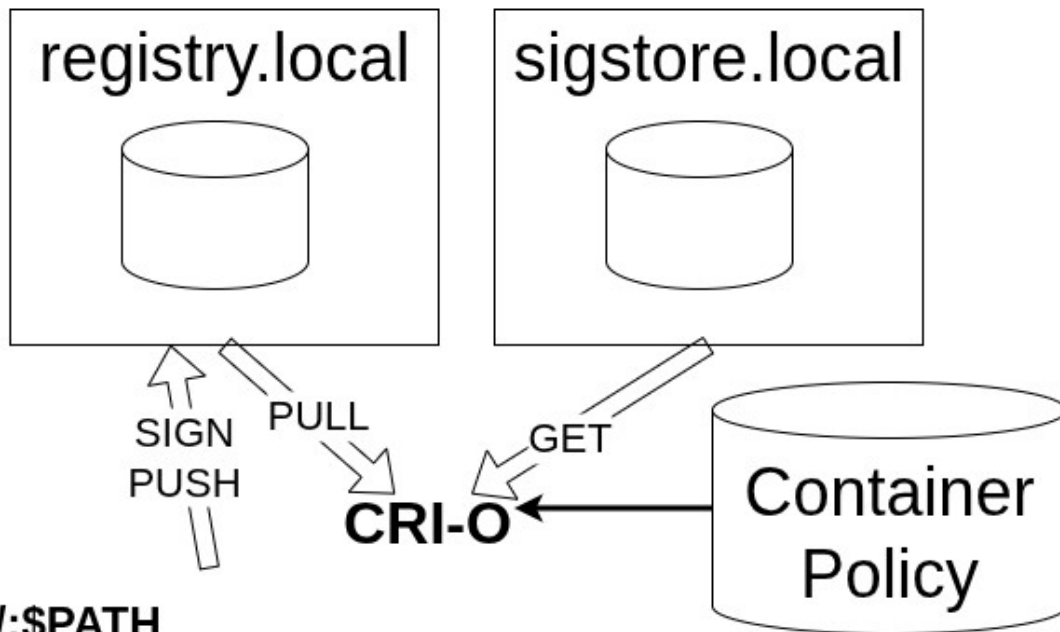
```
# podsec-create-policy  
# podsec-create-services  
# podsec-create-imagemakeruser
```

```
imagemaker$ podsec-load-sign-oci  
imagemaker$ podman...
```

```
# export
```

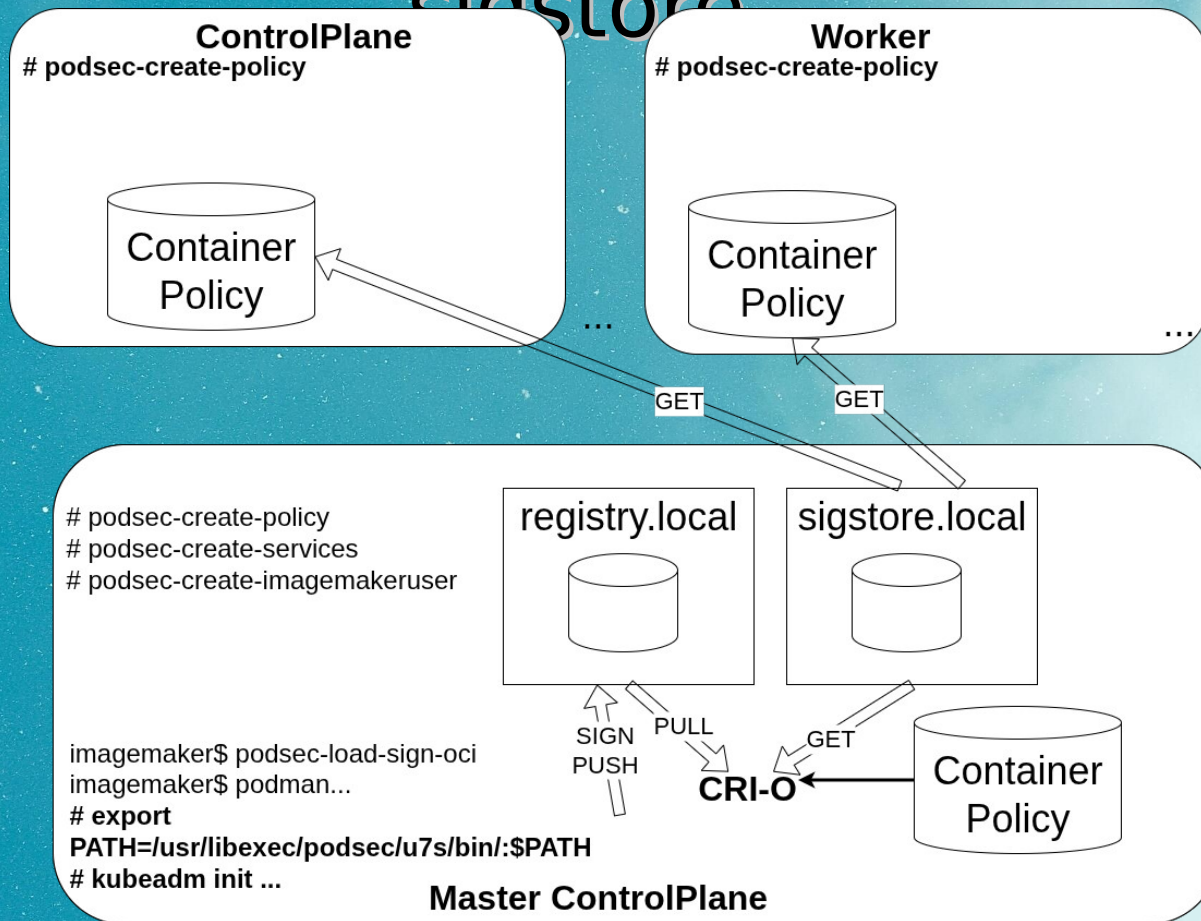
```
PATH=/usr/libexec/podsec/u7s/bin/:$PATH
```

```
# kubeadm init ...
```

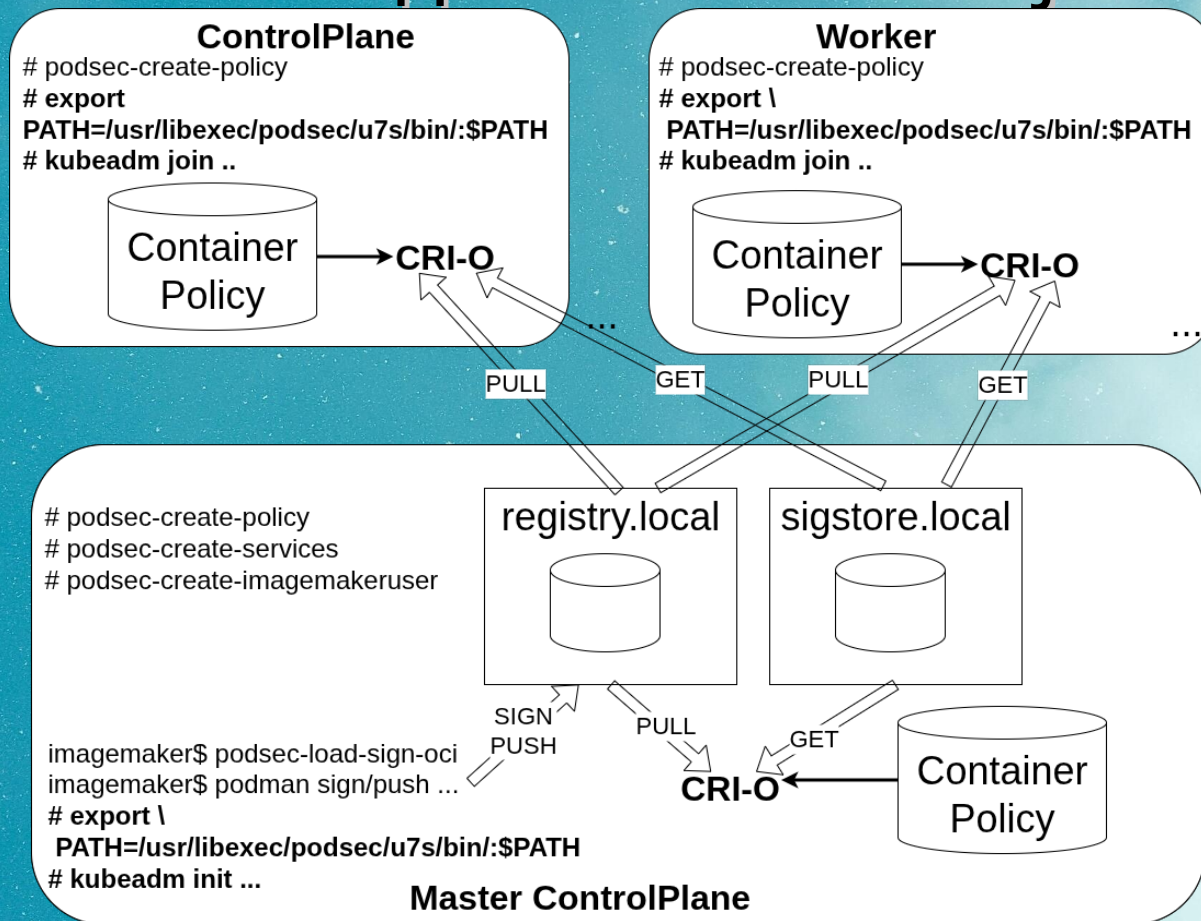


Master ControlPlane

ControlPlane, Worker: Копирование политик доступа, настройка registry, sigstore



ControlPlane, Worker: Разворачивание узлов командами kubeadm join ...



Мониторинг безопасности

Мониторинг безопасности

- **build-invulnerable-image** - команда сборки образов с анализом уязвимостей.
- **check-containers** - мониторинг изменения содержимого roolfull, rootless контейнеров (POD'ов).
- **check-images** - мониторинг образов на предмет их соответствия настройки политикам контейнеризации на узле.
- **check-kubeapi** - мониторинг аудита API-интерфейса сервиса kube-apiserver.
- **check-policy** - мониторинг изменения файлов конфигурации политик доступа к образам.
- **check-vuln** - мониторинг docker-образов узла сканером безопасности trivy.

**ALT-Orchestra/
talos -
sshless
kubernetes
кластер**

Особенности ALT Orchestra

- ALT Orchestra (Альт Оркестрация) клон OS Talos.
- Все ISO и docker-образы полностью собираются на основе пакетной базы ALTLinux-Sisyphus.
- Поддерживается собственный стек extensions на основе пакетной базы ALTLinux-Sisyphus.
- Русифицирован и принят в upstream интерфейс image-factory.
- Создан собственный сервис image-factory <https://factory.altlinux.space/>.
- Зарегистрировано в реестре российского программного обеспечения - <https://reestr.digital.gov.ru/reestr/3607624/>

Особенности решения Talos

- Отсутствие доступа к узлам по ssh
- Работа с кластером только через команды
 - talosctl
 - kubectl
- Минимальная файловая система размещаемая в оперативной памяти и монтируемая в режиме только на чтение.
- Богатый набор встроенный сервисов: CNI, HA, Service Discovery, VIP, ...

Костарев А.Ф.
Базальт СПО
kaf@basealt.ru

