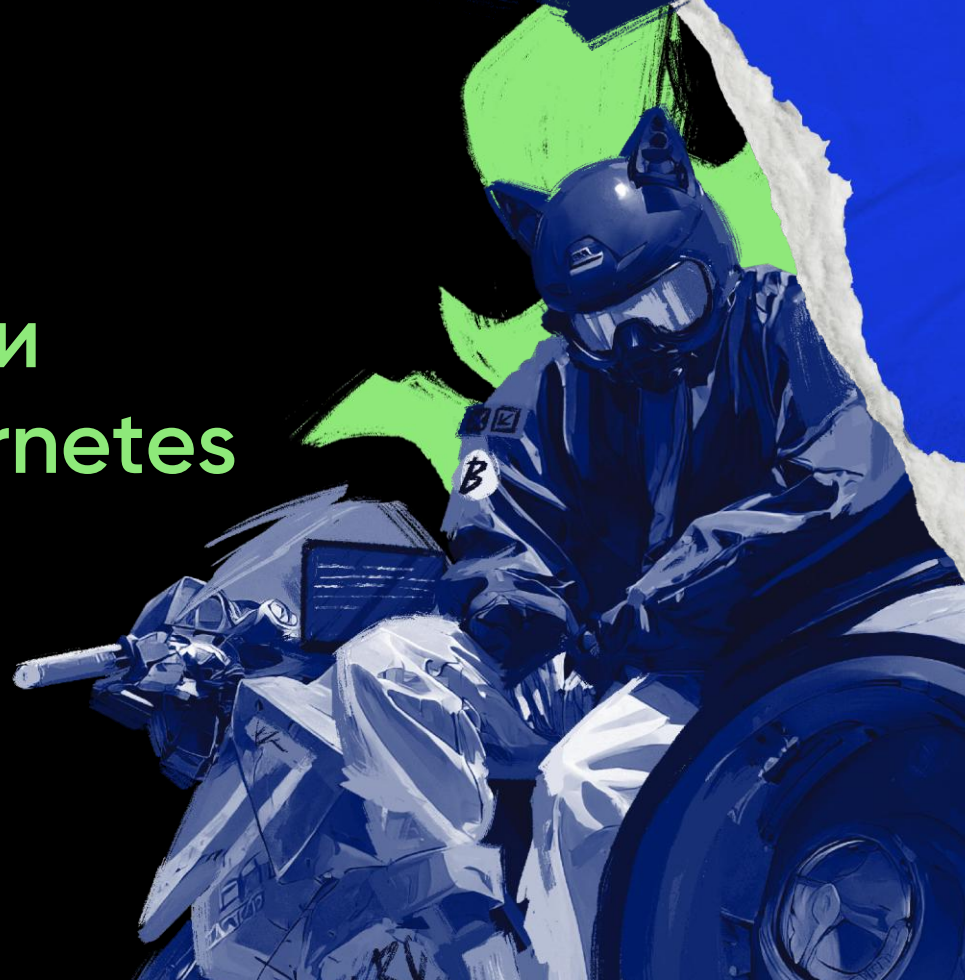
cyber camp

LUNTRY

Путеводитель по безопасности контейнеров и Kubernetes



Дмитрий Евдокимов

Founder&CTO Luntry

27.03.2026

Обо мне

Дмитрий Евдокимов

> *Я не верю в то, что систему можно сделать надежной и безопасной, не понимая того, как она устроена.*

Основатель и технический директор Luntry

Более 15 лет опыта в ИБ

Специализация – безопасность контейнеров и Kubernetes

Автор ТГ-канала k8s(in)security

Эксперт в сфере безопасности контейнерных сред

- Организатор конференции «БеКон» по БЕзопасности КОНтейнеров
- Бывший редактор рубрик в журнале «ХАКЕР», автор серии статей
- Автор курса «Cloud Native безопасность в Kubernetes»
- Член программного комитета конференций DevOpsConf и KUBER CONF

Спикер

VK Kubernetes	Confidence	ZeroNights	БеКон	HITB
DevOpsConf	HackInParis	KuberConf	BlackHat	PHDays
Kazhackstan	HighLoad++	OFFZONE	DevOops	SAS

LUNTRY

План доклада

LUNTRY

01 Подготовка к путешествию

02 Исследование мира безопасности контейнеров и Kubernetes

03 Мир глазами злоумышленников

04 Мир глазами защитников

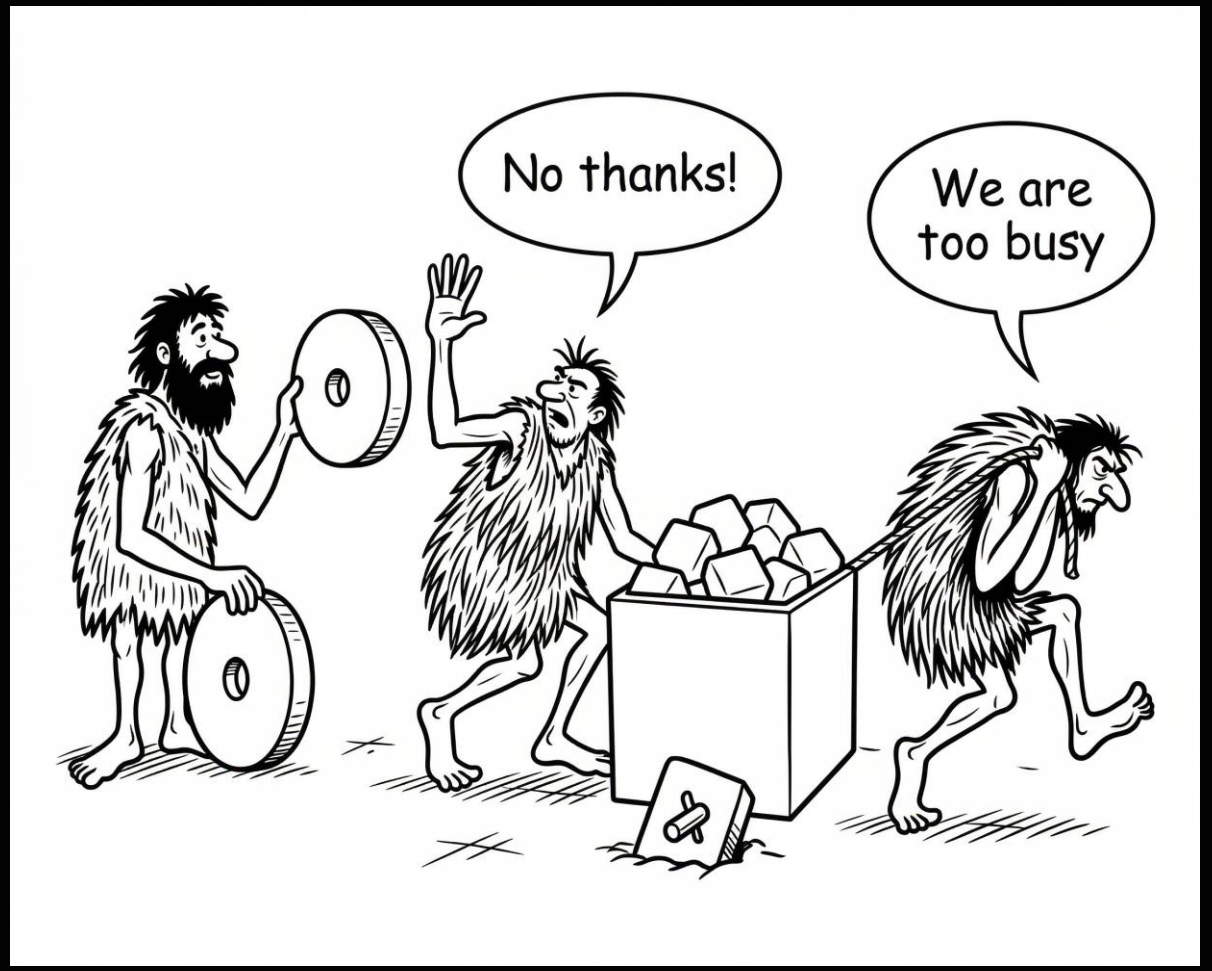
05 Полезности

06 Выводы

LUNTRY

Подготовка к путешествию

Зачем?



LUNTRY

Зачем?

- **Контейнеризация:** Более 90% организаций уже используют контейнеры в производственных средах (production).
- **Kubernetes (K8s):** По данным ежегодного опроса CNCF за 2025 год, использование Kubernetes в production достигло 82%, закрепив за ним статус «де-факто операционной системы» для облачных вычислений и ИИ-приложений.
- **Рост экосистемы:** С момента присоединения Kubernetes к CNCF количество компаний-контрибьюторов выросло почти на 1000% (с 731 до более чем 8000).
- **Тренды 2026:** Основными направлениями развития остаются Platform Engineering, безопасность цепочек поставок программного обеспечения и интеграция ИИ-нагрузок непосредственно в K8s-кластеры.

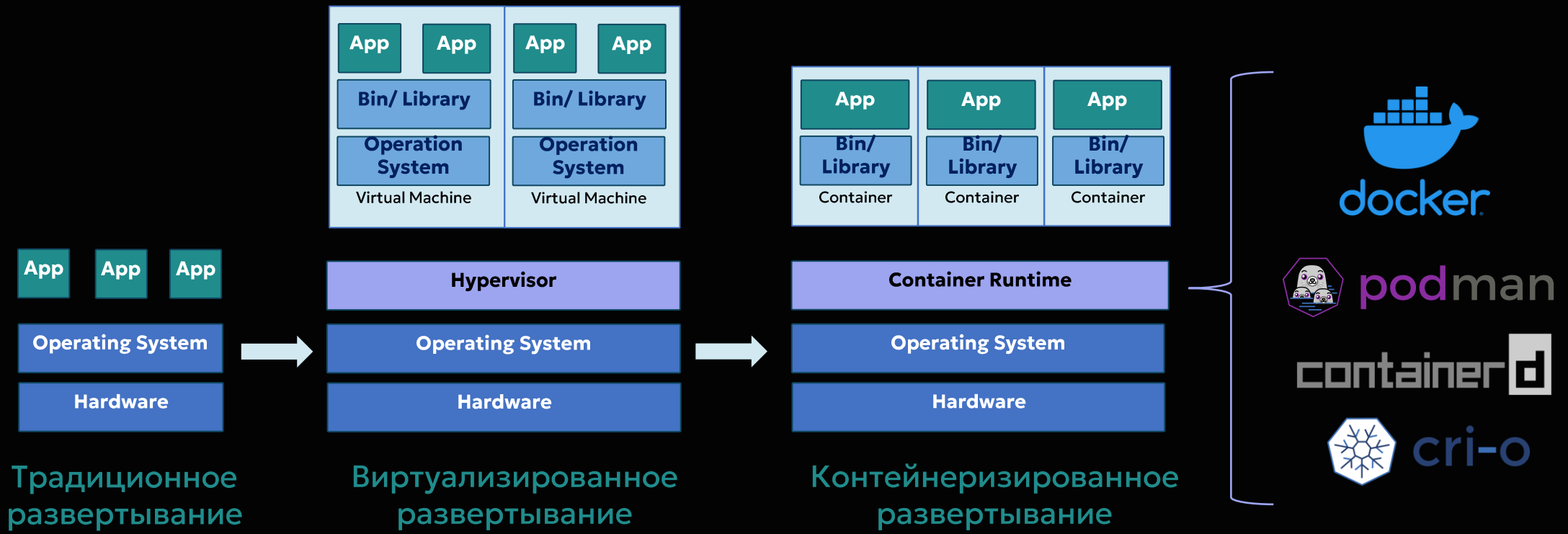
LUNTRY

Фундаментальное изменение в разработке

1. Улучшенная упаковка приложений
2. Изоляция приложений
3. Лёгкость и скорость по сравнению с виртуальными машинами
4. Поддержка микросервисной архитектуры
5. Упрощение CI/CD-пайплайнов
6. Декларативность и Infrastructure as Code
7. Горизонтальное масштабирование
8. Оптимизация использования ресурсов
9. Портативность между средами и облаками
10. Упрощение onboarding-а новых разработчиков
11. Быстрое восстановление и самовосстановление
12. Безопасность через ограничение поверхности атаки
13. Богатая экосистема и стандартизация
14. Поддержка blue-green и canary-деплоев

АСПЕКТ	БЕЗ КОНТЕЙНЕРОВ	С КОНТЕЙНЕРАМИ
Настройка среды	Часы / дни	Минуты
Консистентность сред	Ручная синхронизация	Гарантирована образом
Масштабирование	Ручное, медленное	Автоматическое, мгновенное
Плотность размещения	Низкая (ВМ)	Высокая
Откат версий	Рискованный	Мгновенный
Vendor lock-in	Высокий	Минимальный

Эволюция к контейнерам



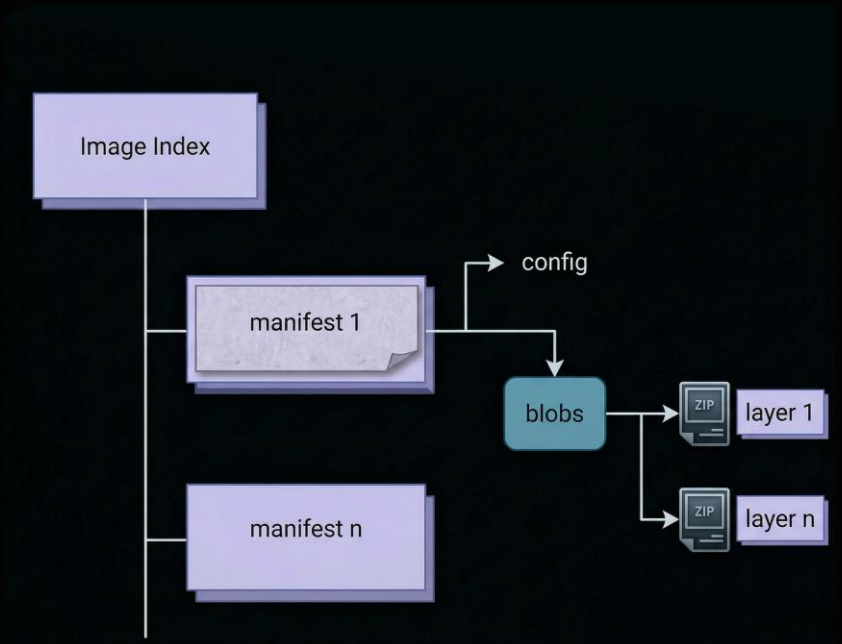
Что такое контейнер?

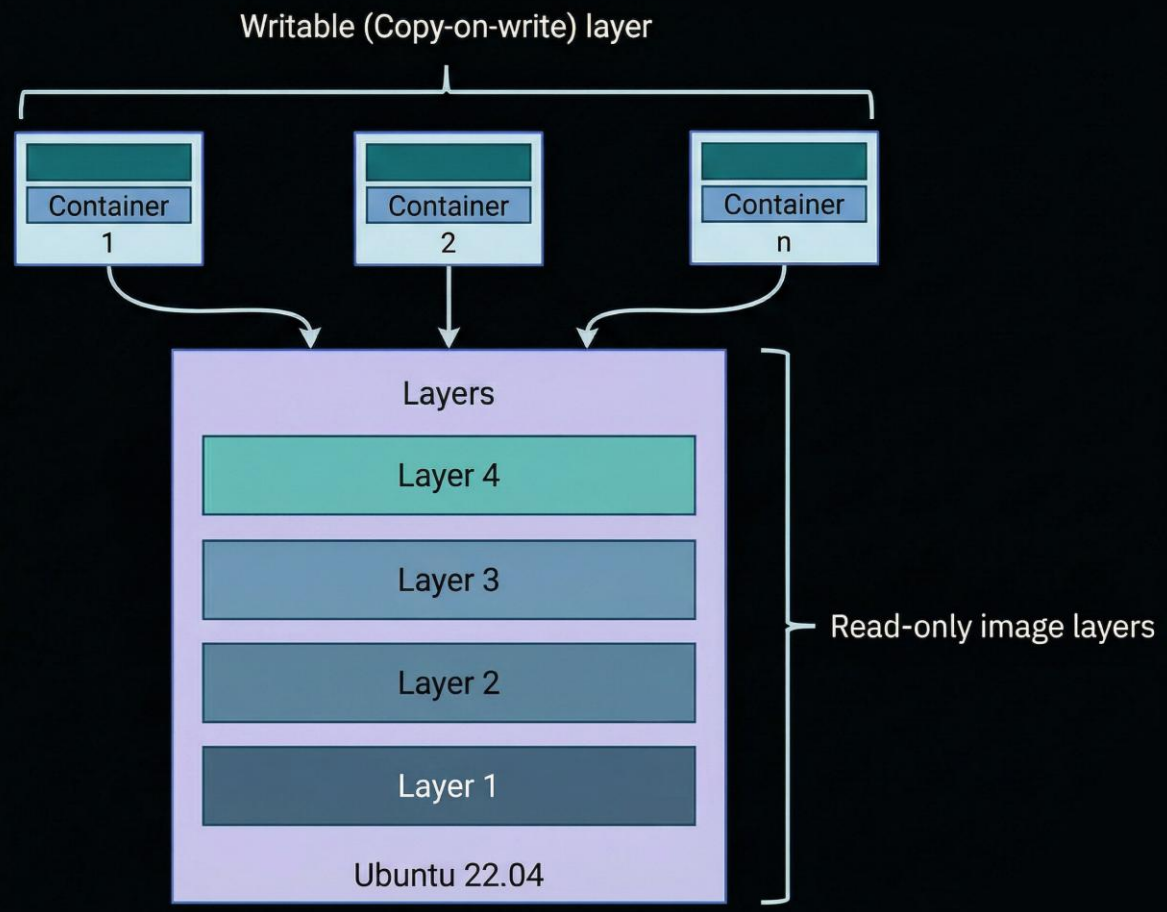
- Container это Linux process с определёнными свойствами/ограничениями
  - Что можно увидеть: namespaces (pid, user, uts, ipc, net, mnt), pivot_root (+ image)
  - Что можно делать: Capabilities, seccomp, LSMs
  - Что можно использовать: Control group (процессор, память, устройства, …)

```
root     2966156  0.0  0.0 110128  5932 ?        Sl   Nov19   0:11  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root     2966174  0.0  0.0   1020     4 ?        Ss   Nov19   0:00  |   \_ /pause
root     2966375  0.0  0.0 108720  6356 ?        Sl   Nov19   0:11  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
sadm     2966394  0.0  0.0 827512 19728 ?        Ssl  Nov19   0:00  |   \_ node /usr/bin/nodemon /src/index.js
sadm     2966421  0.0  0.0   4460    80 ?        S    Nov19   0:00  |       \_ sh -c node /src/index.js
sadm     2966422  0.0  0.0 967396 16596 ?        Sl   Nov19   0:00  |           \_ node /src/index.js
root     2988902  0.0  0.0 108720  5408 ?        Sl   Nov19   0:11  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root     2988922  0.0  0.0   1020     4 ?        Ss   Nov19   0:00  |   \_ /pause
root     2989066  0.0  0.0 108720  5408 ?        Sl   Nov19   0:26  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root     2989099  0.0  0.0  31000 23956 ?        Ss   Nov19   0:42  |   \_ /usr/local/bin/python /usr/local/bin/gunicorn -b :8080 --workers 1 --threads 1 --
root     2989116  0.3  0.1 142092 48964 ?        Sl   Nov19  16:50  |       \_ /usr/local/bin/python /usr/local/bin/gunicorn -b :8080 --workers 1 --threads
root     2989333  0.0  0.0 110128  5404 ?        Sl   Nov19   0:11  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root     2989352  0.0  0.0   1020     4 ?        Ss   Nov19   0:00  |   \_ /pause
root      596808  0.0  0.0 110128  6316 ?        Sl   Nov20   0:06  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root      596827  0.0  0.0   1020     4 ?        Ss   Nov20   0:00  |   \_ /pause
root      598309  0.0  0.0 110128  6224 ?        Sl   Nov20   0:07  \_ containerd-shim -namespace moby -workdir /var/lib/containerd/io.containerd.runtime.v1
root      598334  1.4  5.5 7236340 1832196 pts/0 Ssl+ Nov20  39:39      \_ /docker-java-home/bin/java -Djava.util.logging.config.file=/opt/atlassian/conflue
root      599854  1.0  1.3 7007820 427956 pts/0  Sl+  Nov20  28:11      |   \_ /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -classpath /opt/atlassian/conf
root      701694  0.0  0.0   4288   764 ?        Ss+  Nov20   0:00      \_ /bin/sh
```

Что такое образ контейнера?

- Container Image это неизменяемый пакет файлов операционной системы, кода приложения и любых зависимостей приложения
  - Union File System
    - OverlayFS как реализация
  - OCI image спецификация

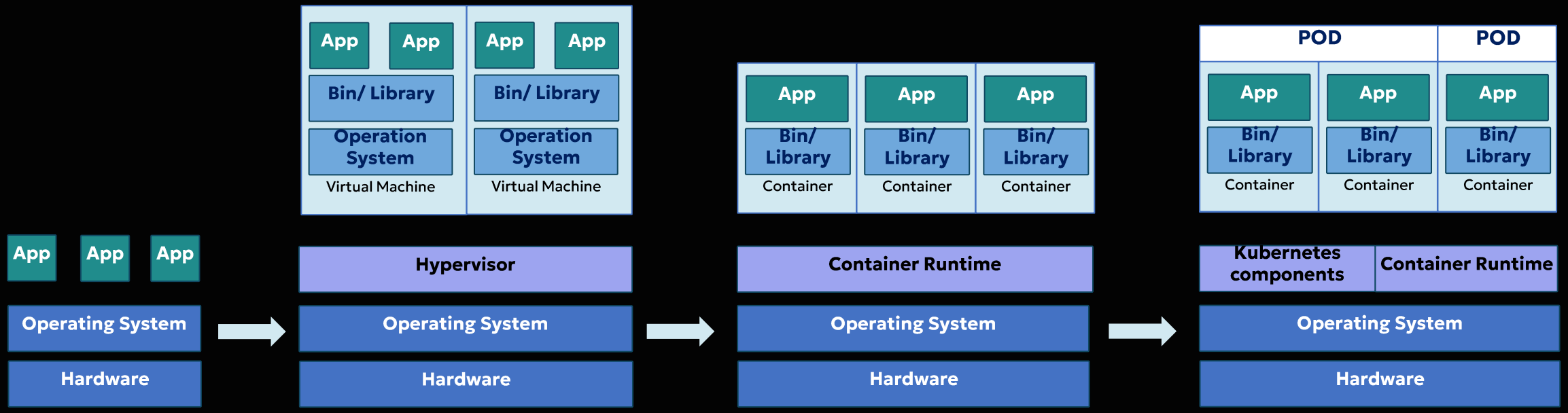




Эволюция от контейнеров к оркестраторам контейнеров

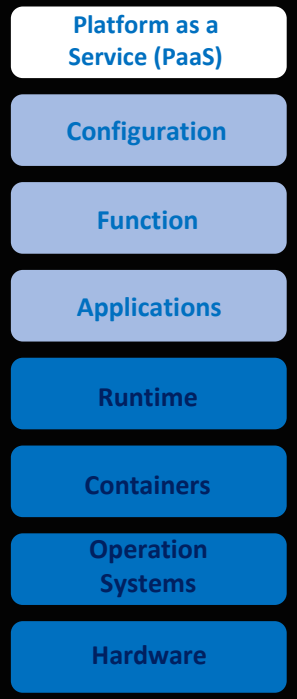
LUNTRY

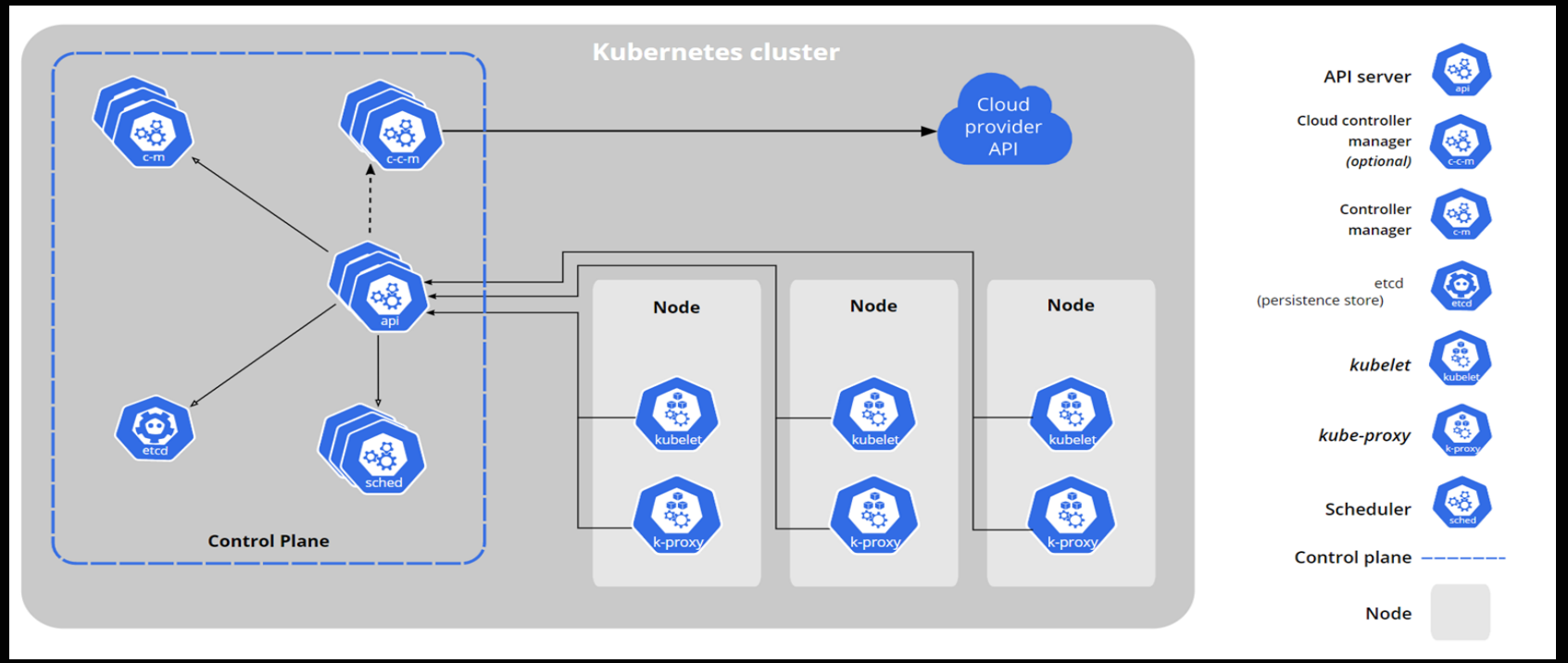
Оркестраторы: kubernetes HashiCorp Nomad docker SWARM



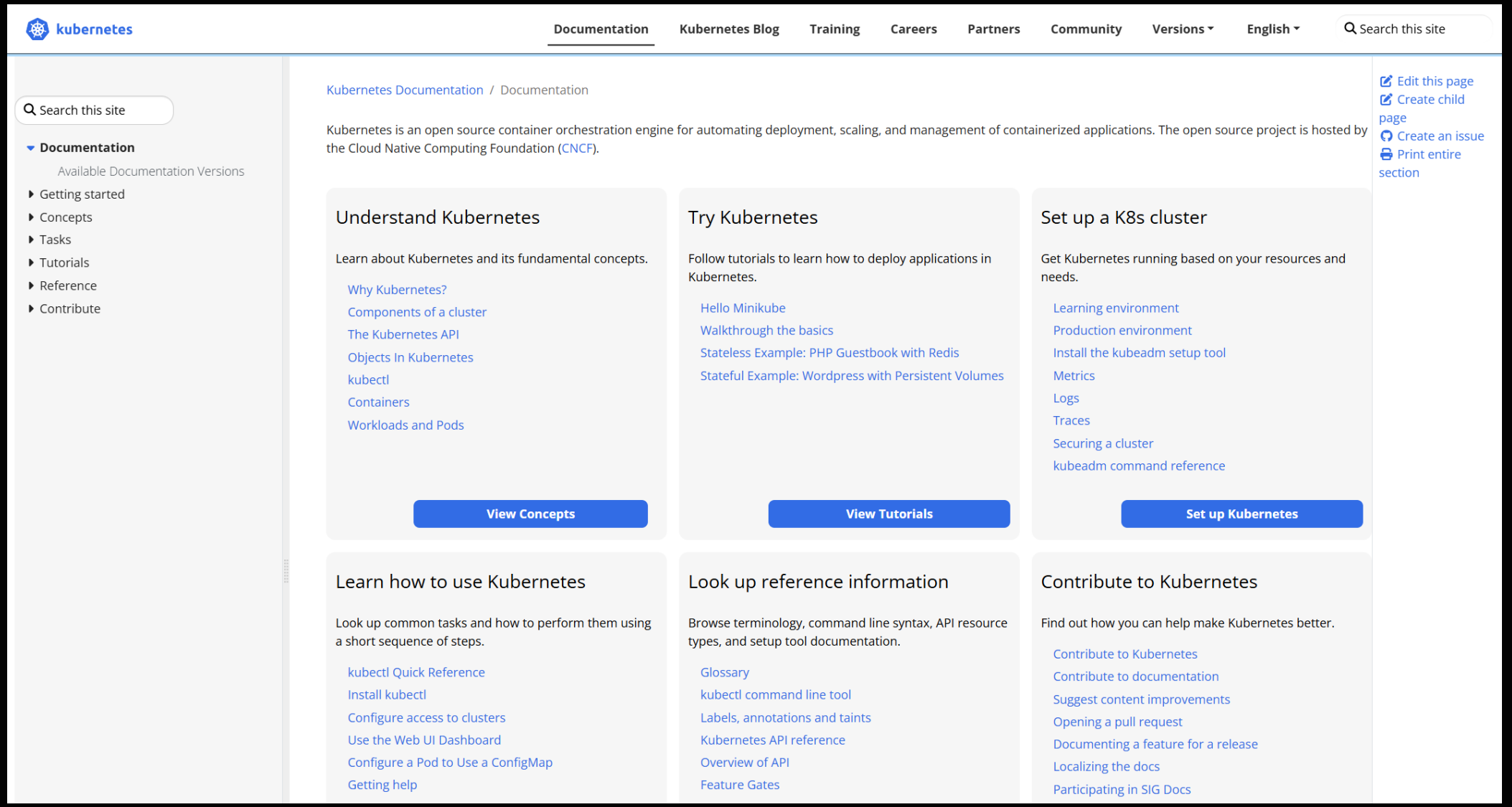
Что такое Kubernetes?

- Kubernetes (K8s) — это открытое программное обеспечение для автоматизации развёртывания, масштабирования и управления контейнеризированными приложениями.
  - 5 бинарей





Официальная документация Kubernetes



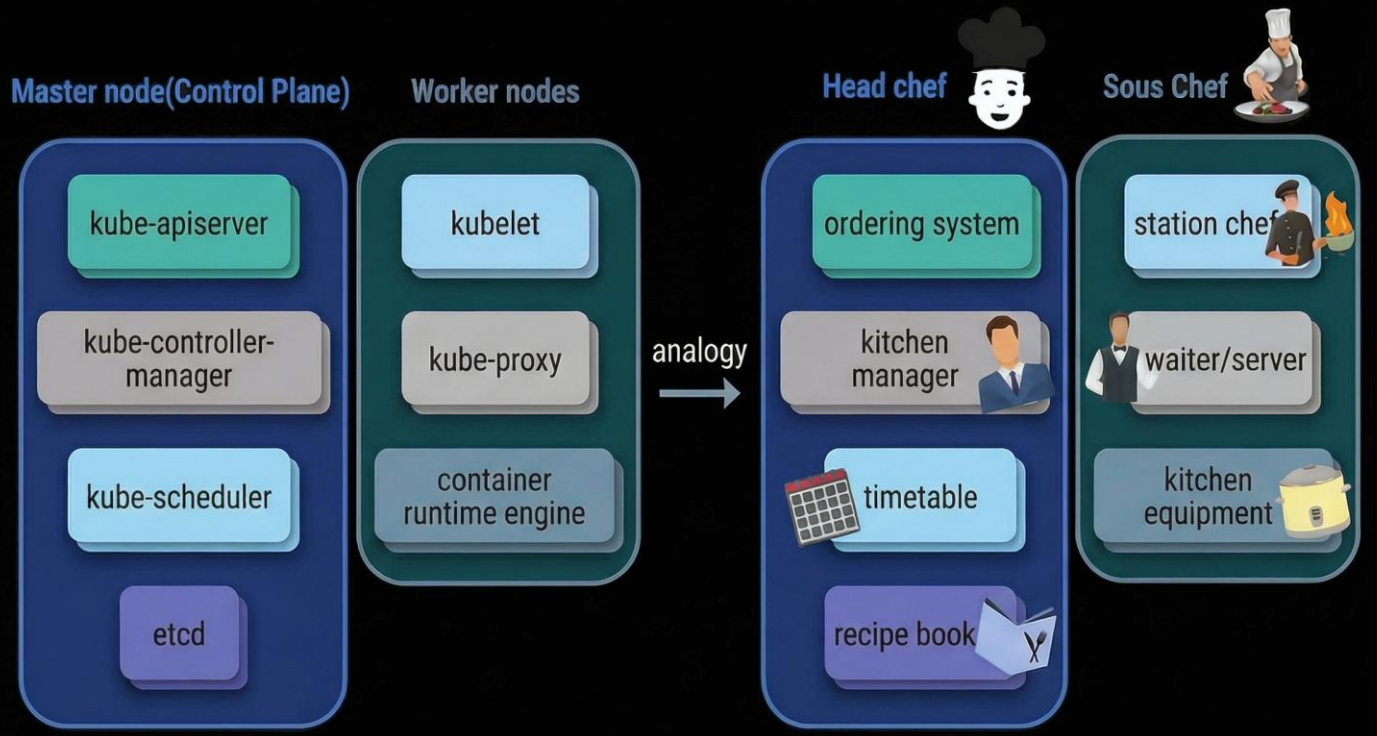
https://kubernetes.io/docs/home/

Дистрибутивы Kubernetes

- Это фреймворк
  - OnPrem и Managed Kubernetes
  - Это ядро Linux 21 века
  - На базе Kubernetes делают свои дистрибутивы

LUNTRY

Kubernetes на простом языке

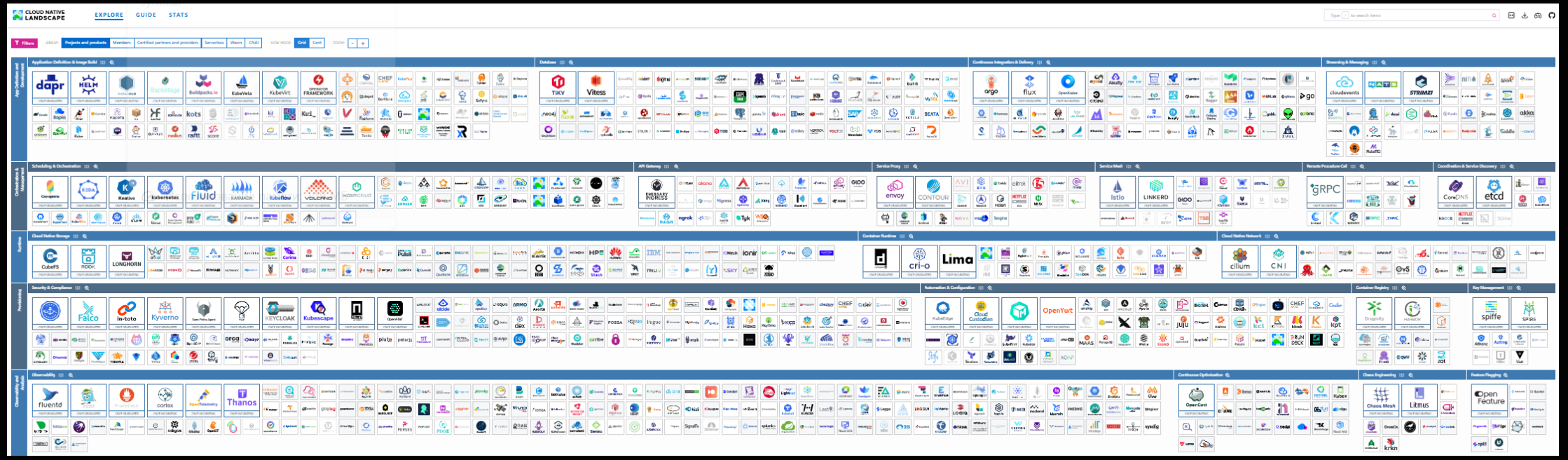


Kubernetes в переводе на детский

Понимание архитектуры Kubernetes через аналогию с шеф-поваром ресторана

CNCF Landscape

Cloud Native экосистема просто огромная и постоянно только растет!

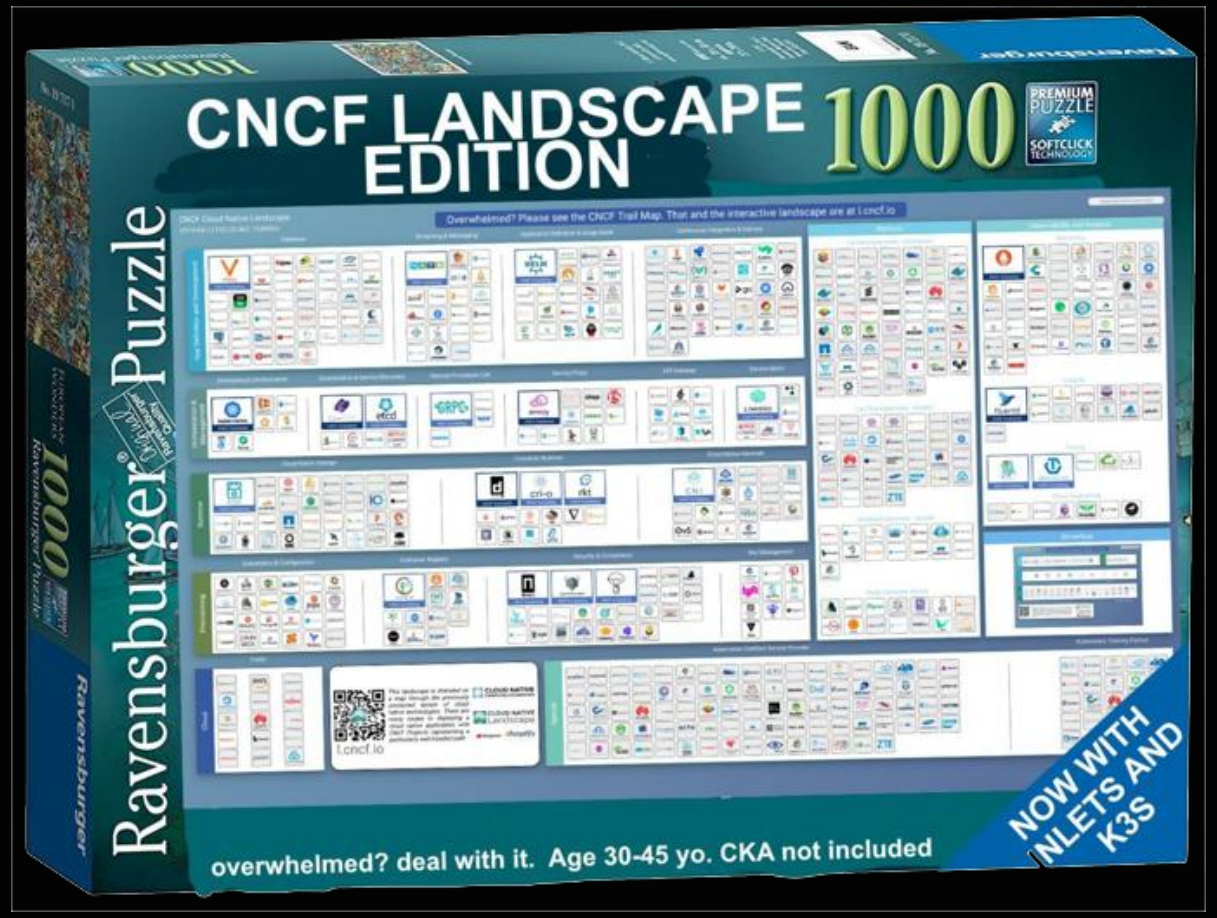


https://landscape.cncf.io/

LUNTRY

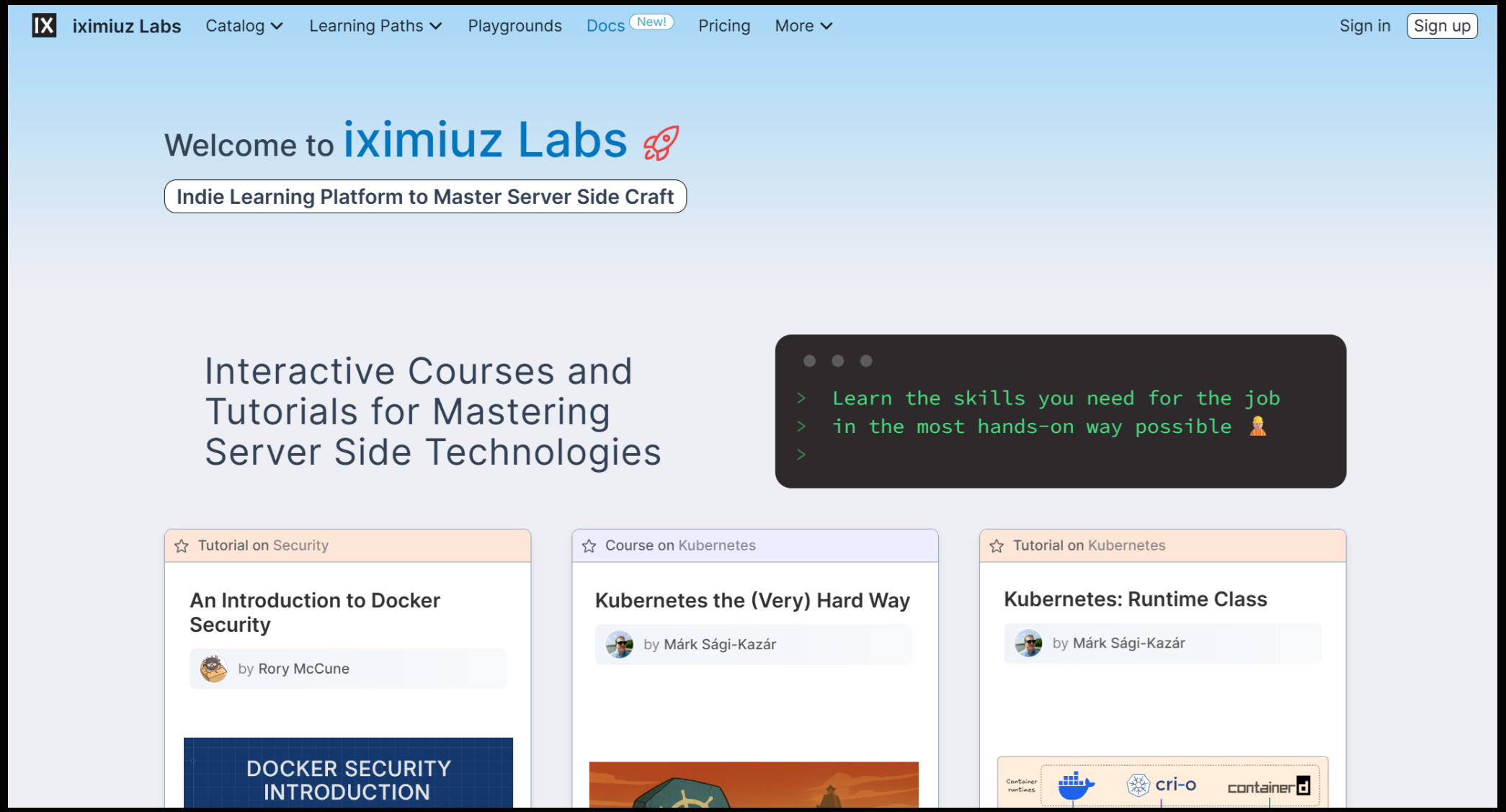
CNCF Landscape

=)



LUNTRY

Изучаем на практике



https://labs.iximiuz.com/

Исследование мира безопасности контейнеров и Kubernetes

Cloud Native Security Whitepaper

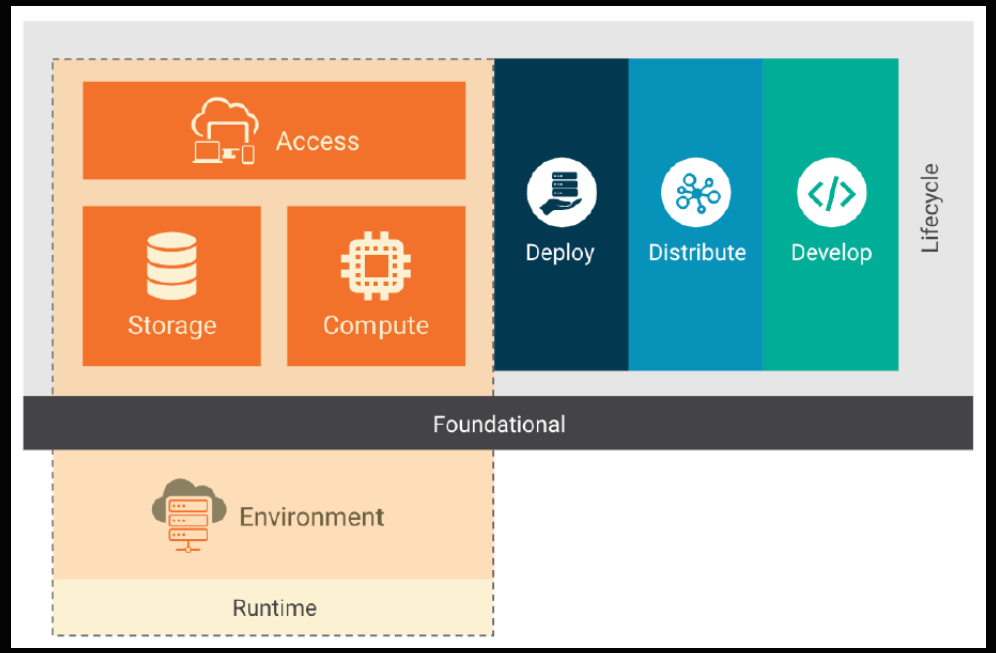
Security concerns within this landscape are complex because of the explicit focus on rapid development and deployment in addition to impracticality of traditional perimeter-based security. This complexity requires a paradigm shift to protect applications by migrating from a perimeter based approach to one where security moves closer to dynamic workloads that are identified based on attributes and metadata (e.g. labels and tags). This approach identifies and secures workloads to meet the scale needs of cloud native applications while accommodating constant flux.

Cloud native development can be modeled in distinct phases that constitute the application lifecycle: "Develop," "Distribute," "Deploy," and "Runtime." Cloud native security contrasts with traditional security approaches in that there is a tremendous opportunity to ensure that security is injected throughout these distinct phases instead of bookending the lifecycle with separately managed security informed interventions.

https://github.com/cncf/tag-security/tree/main/community/resources/security-whitepaper

Фазы жизненного цикла

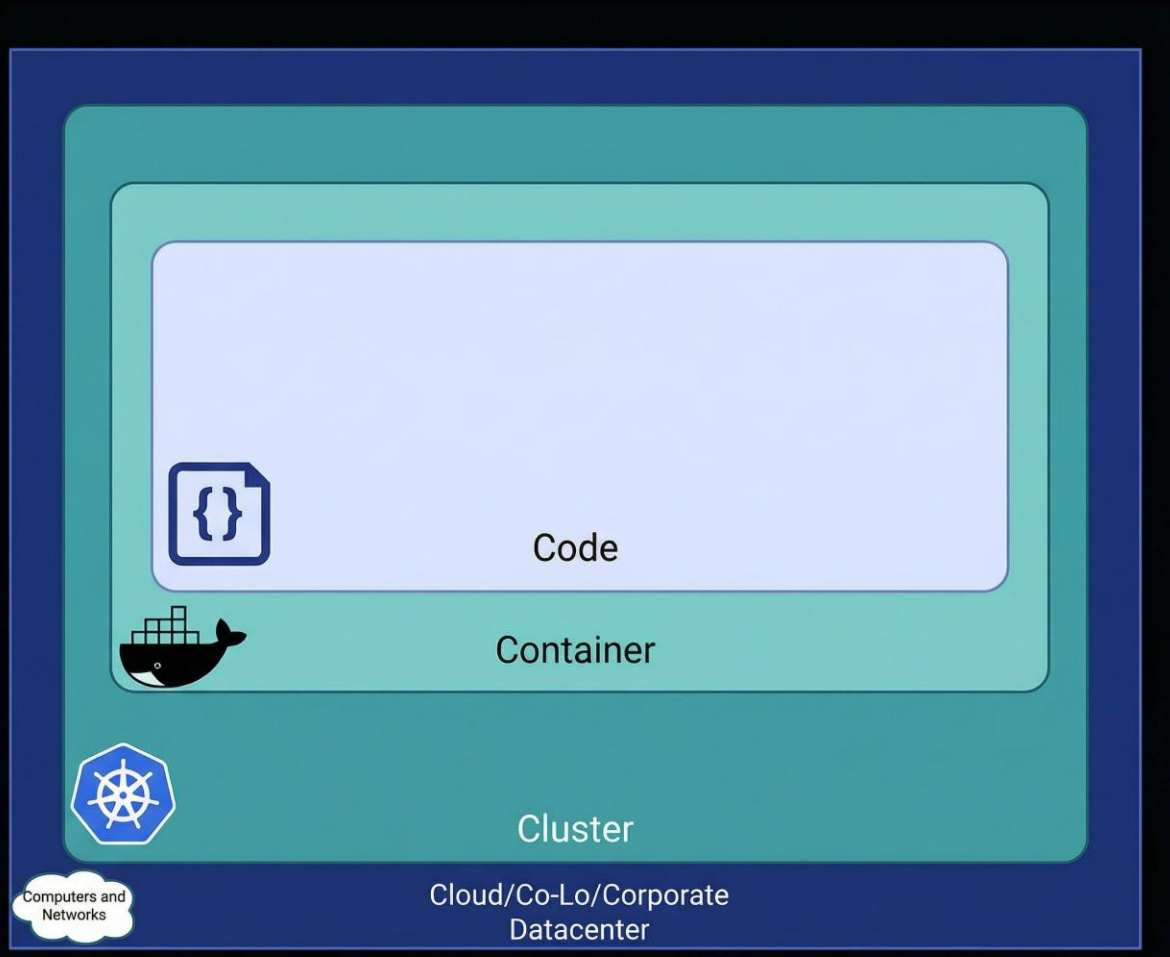
- Develop - разработка
  - Security как можно раньше
  - Infrastructure as Code (IaC)
- Distribute - распространение
  - Безопасность supply chain
  - Постоянное автоматическое сканирование и обновление артефактов
  - Подпись артефактов
- Deploy - развертывание
  - Проверка атрибутов рабочих нагрузок
  - Параллельное развертывание дополнительных средств
- Runtime - выполнение
  - Применение политик ограничения возможностей и ресурсов
  - Обнаружение, предотвращение и уведомление о несанкционированном доступе к ресурсам



LUNTRY

Подход 4C для безопасности Cloud Native

- Cloud/Co-Lo/Corporate Datacenter
  - Управление активами
- Cluster
  - Компоненты Kubernetes
  - Kubernetes ресурсы
- Container
  - Процессы
  - Взаимодействие с файлами
  - Взаимодействие с сетью
- Code
  - Образы контейнеров



LUNTRY

7 доменов безопасности Kubernetes

Безопасность Kubernetes-ресурсов

Контроль состояния Kubernetes-кластеров

Контроль соответствия кластера стандартам

Управление безопасностью образов контейнеров

Сетевая безопасность

Анализ прав доступа

Защита Runtime

Жизненный путь образа

LUNTRY

CI\CD Pipeline

- Сборка образа
- Анализ образа
- Security/Quality gates
  - Возможна блокировка
- Подпись образа

Image registry

- Хранение
- Очистка/ротация
- Анализ образов
- Security/Quality gates

Runtime

- Запуск
- Анализ образов
- Security/Quality gates

Домены безопасности образов

LUNTRY

5 ДОМЕНОВ:

- Анализ Dockerfile на лучшие практики
- Поиск чувствительной информации
- Анализ на известные уязвимости
- Анализ на вредоносный код и код двойного назначения
- Компонентный анализ

LUNTRY

Мир глазами злоумышленников

Модели нарушителя

ВНЕШНИЙ

Попадает внутрь контейнера Pod и оттуда обращается к рядом стоящим сервисам или делает побег из контейнера на Node

ВНУТРЕННИЙ

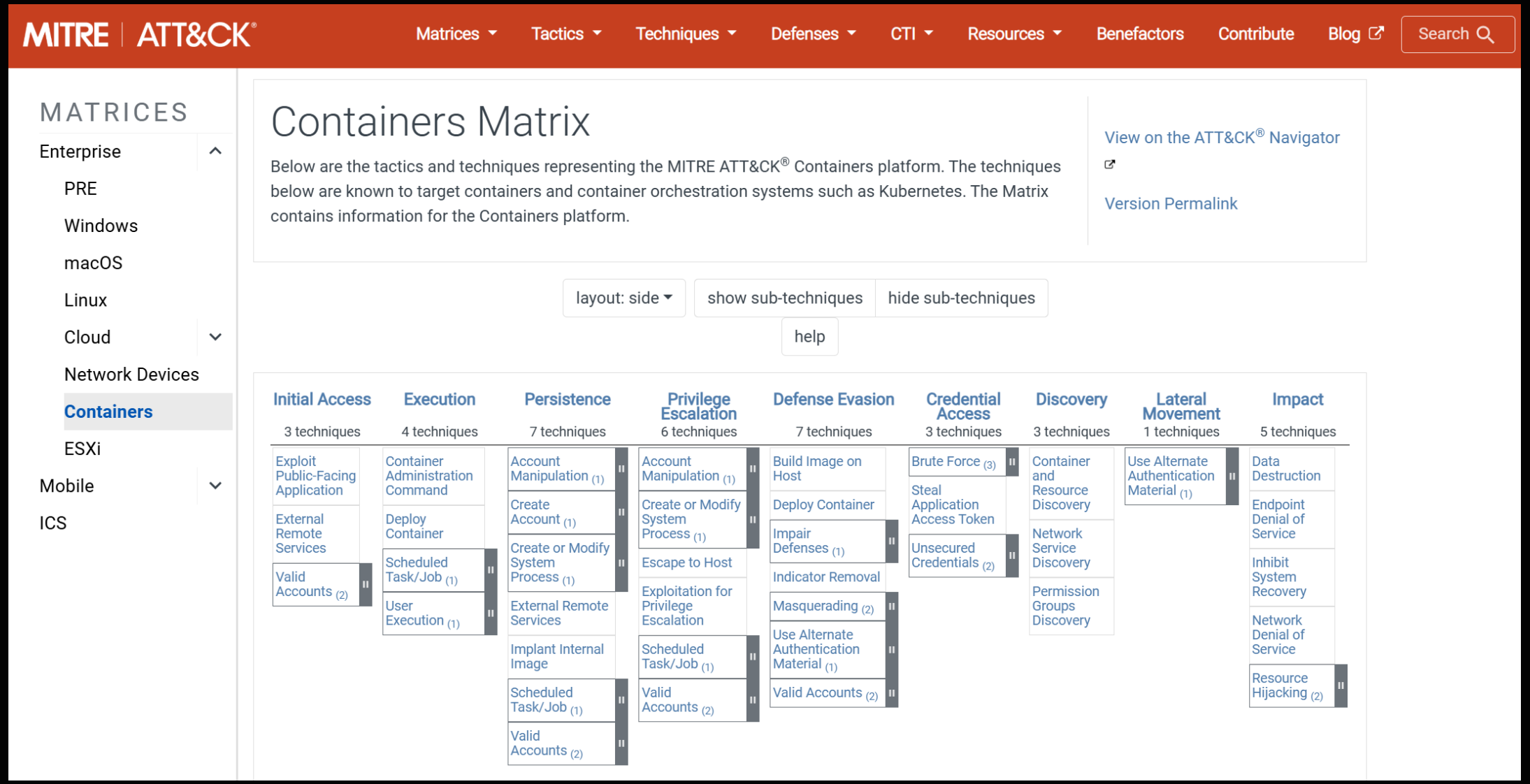
Злоумышленник с Node или из одной сети с Kubernetes

СКОМПРОМЕТИРОВАННЫЙ РАЗРАБОТЧИК

Контролирует / формирует содержимое YAML ресурсов, образов контейнеров

LUNTRY

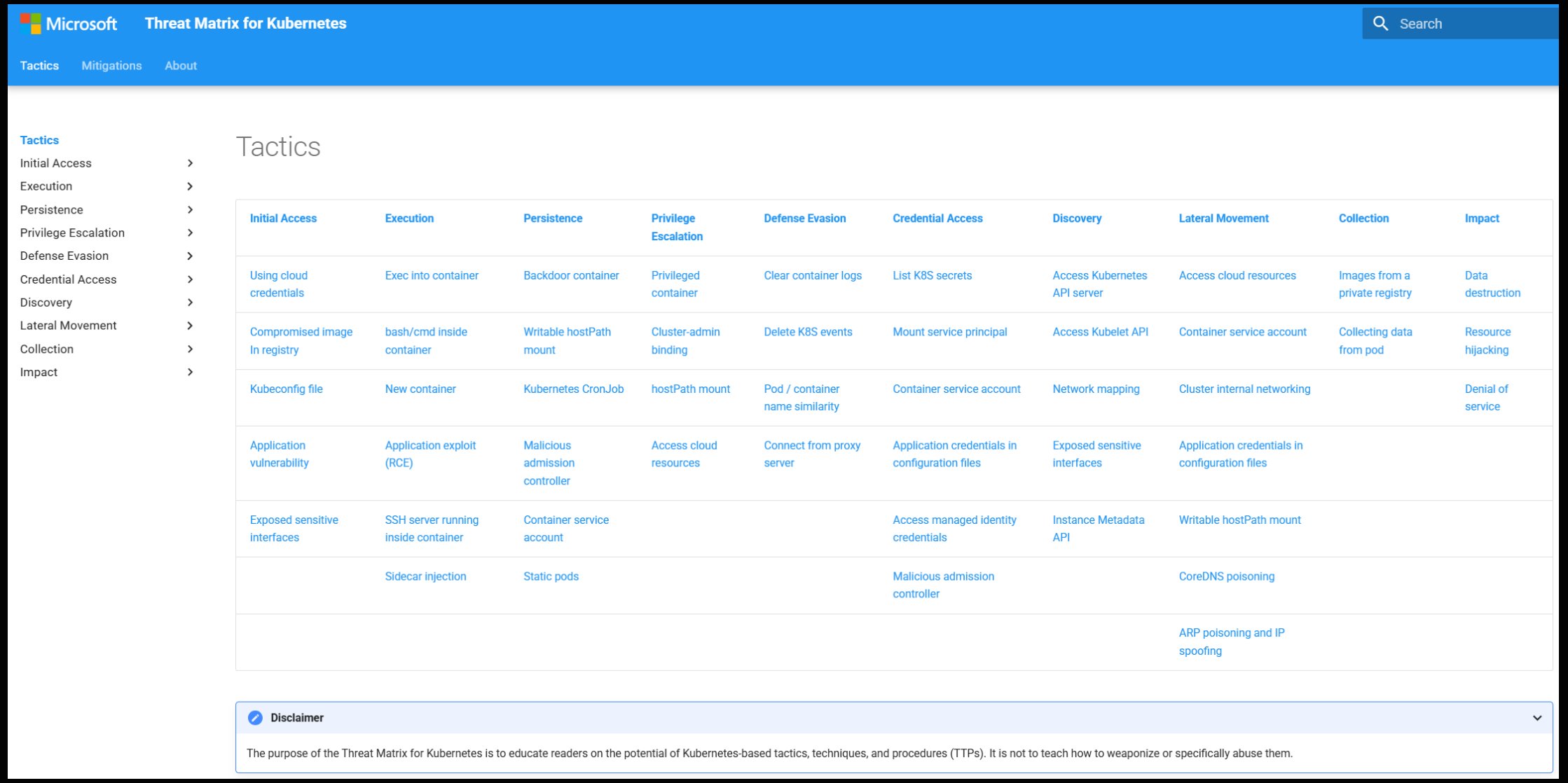
Матрица угроз для контейнеров от MITRE



https://attack.mitre.org/matrices/enterprise/containers/

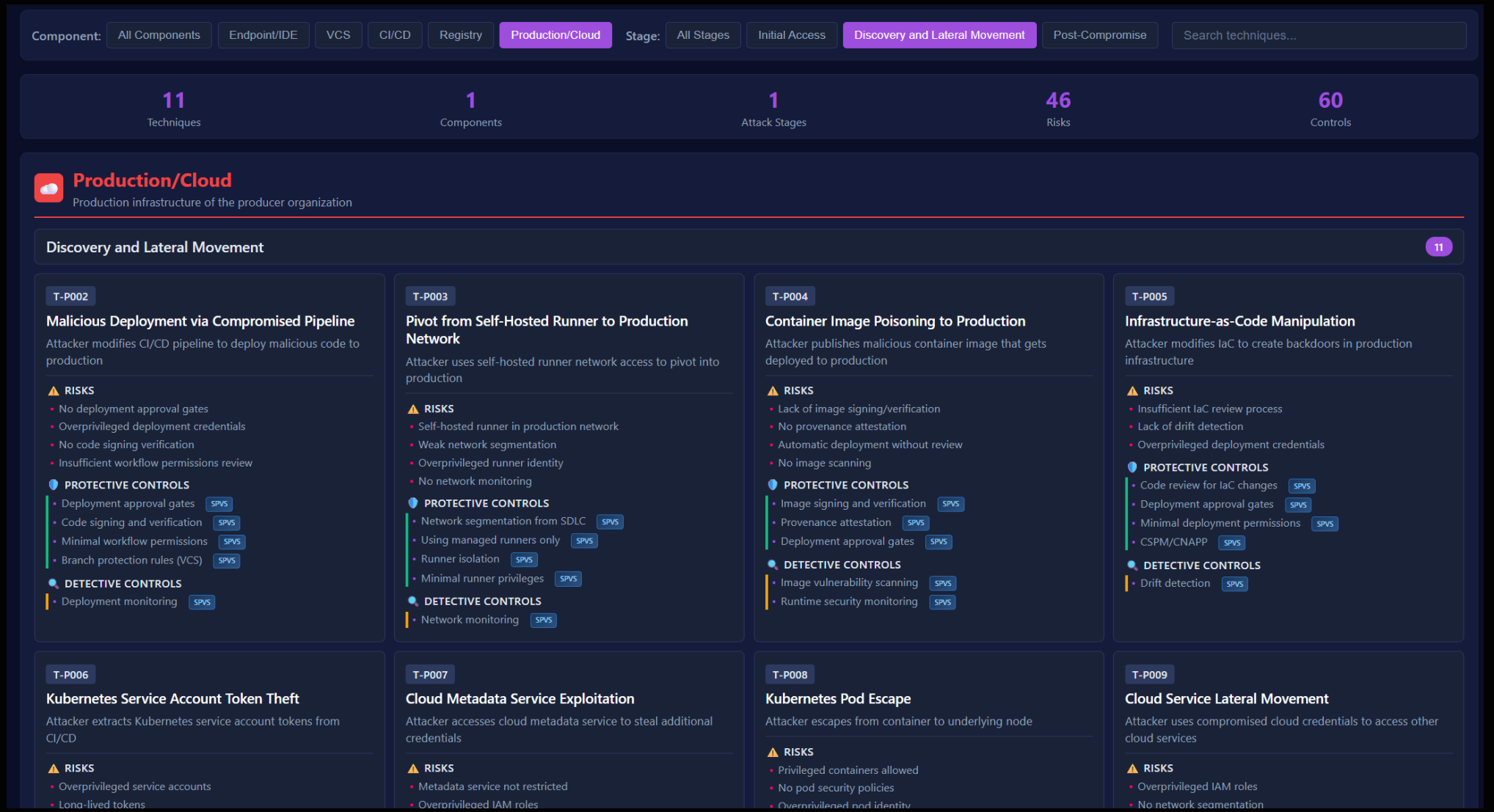
LUNTRY

Матрица угроз для Kubernetes от Microsoft



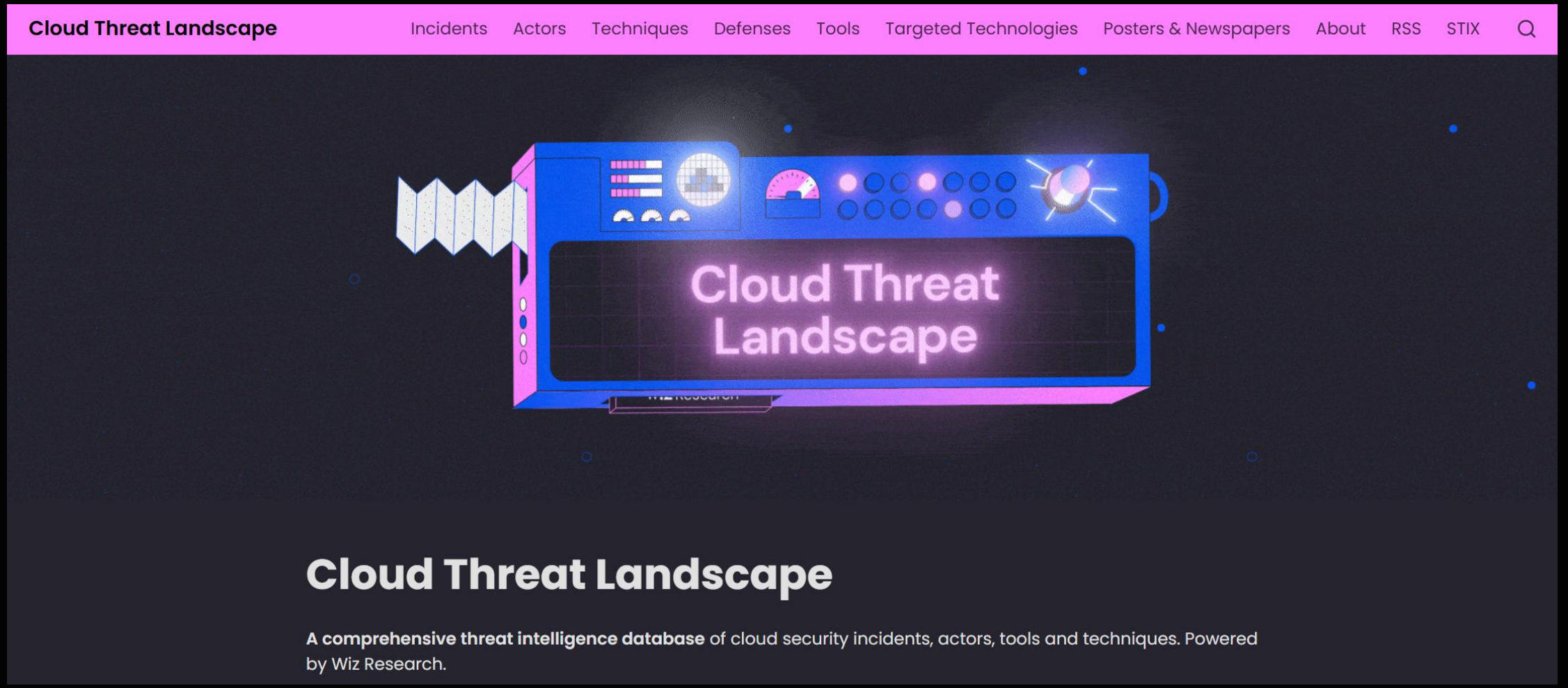
https://microsoft.github.io/Threat-Matrix-for-Kubernetes/

STIF - SDLC Infrastructure Threat Framework



https://wiz-sec-public.github.io/SITF/techniques-library.html

Cloud Threat Landscape

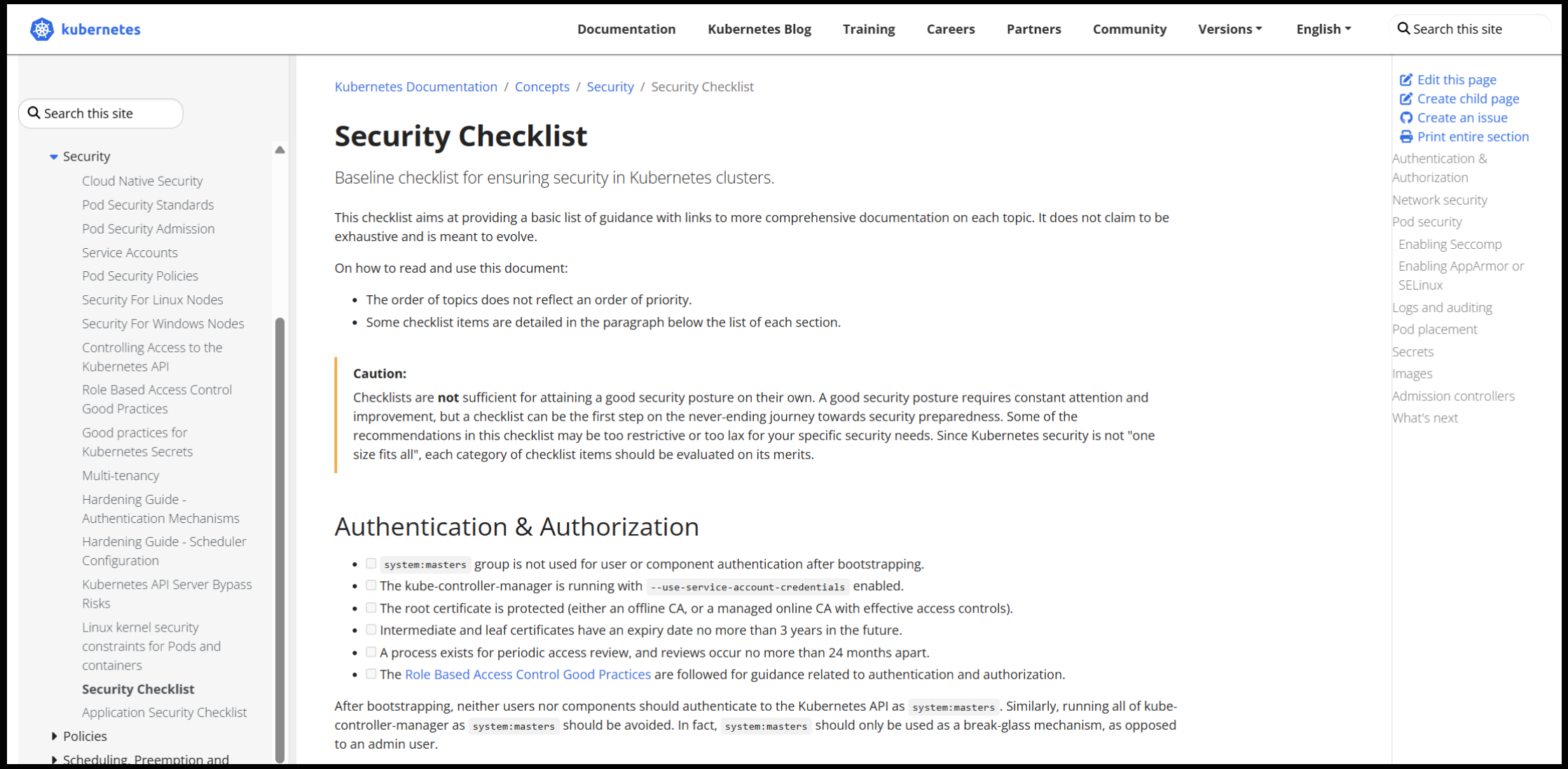


https://threats.wiz.io/

LUNTRY

Мир глазами защитников

Раздел документации Kubernetes по безопасности



kubernetes

Documentation | Kubernetes Blog | Training | Careers | Partners | Community | Versions | English | Search this site

Search this site

- Security
  - Cloud Native Security
  - Pod Security Standards
  - Pod Security Admission
  - Service Accounts
  - Pod Security Policies
  - Security For Linux Nodes
  - Security For Windows Nodes
  - Controlling Access to the Kubernetes API
  - Role Based Access Control Good Practices
  - Good practices for Kubernetes Secrets
  - Multi-tenancy
  - Hardening Guide - Authentication Mechanisms
  - Hardening Guide - Scheduler Configuration
  - Kubernetes API Server Bypass Risks
  - Linux kernel security constraints for Pods and containers
  - **Security Checklist**
  - Application Security Checklist
- Policies
- Scheduling, Preemption and

Kubernetes Documentation / Concepts / Security / Security Checklist

Security Checklist

Baseline checklist for ensuring security in Kubernetes clusters.

This checklist aims at providing a basic list of guidance with links to more comprehensive documentation on each topic. It does not claim to be exhaustive and is meant to evolve.

On how to read and use this document:

- The order of topics does not reflect an order of priority.
- Some checklist items are detailed in the paragraph below the list of each section.

> **Caution:**
> Checklists are **not** sufficient for attaining a good security posture on their own. A good security posture requires constant attention and improvement, but a checklist can be the first step on the never-ending journey towards security preparedness. Some of the recommendations in this checklist may be too restrictive or too lax for your specific security needs. Since Kubernetes security is not "one size fits all", each category of checklist items should be evaluated on its merits.

Authentication & Authorization

- ☐ `system:masters` group is not used for user or component authentication after bootstrapping.
- ☐ The kube-controller-manager is running with `--use-service-account-credentials` enabled.
- ☐ The root certificate is protected (either an offline CA, or a managed online CA with effective access controls).
- ☐ Intermediate and leaf certificates have an expiry date no more than 3 years in the future.
- ☐ A process exists for periodic access review, and reviews occur no more than 24 months apart.
- ☐ The Role Based Access Control Good Practices are followed for guidance related to authentication and authorization.

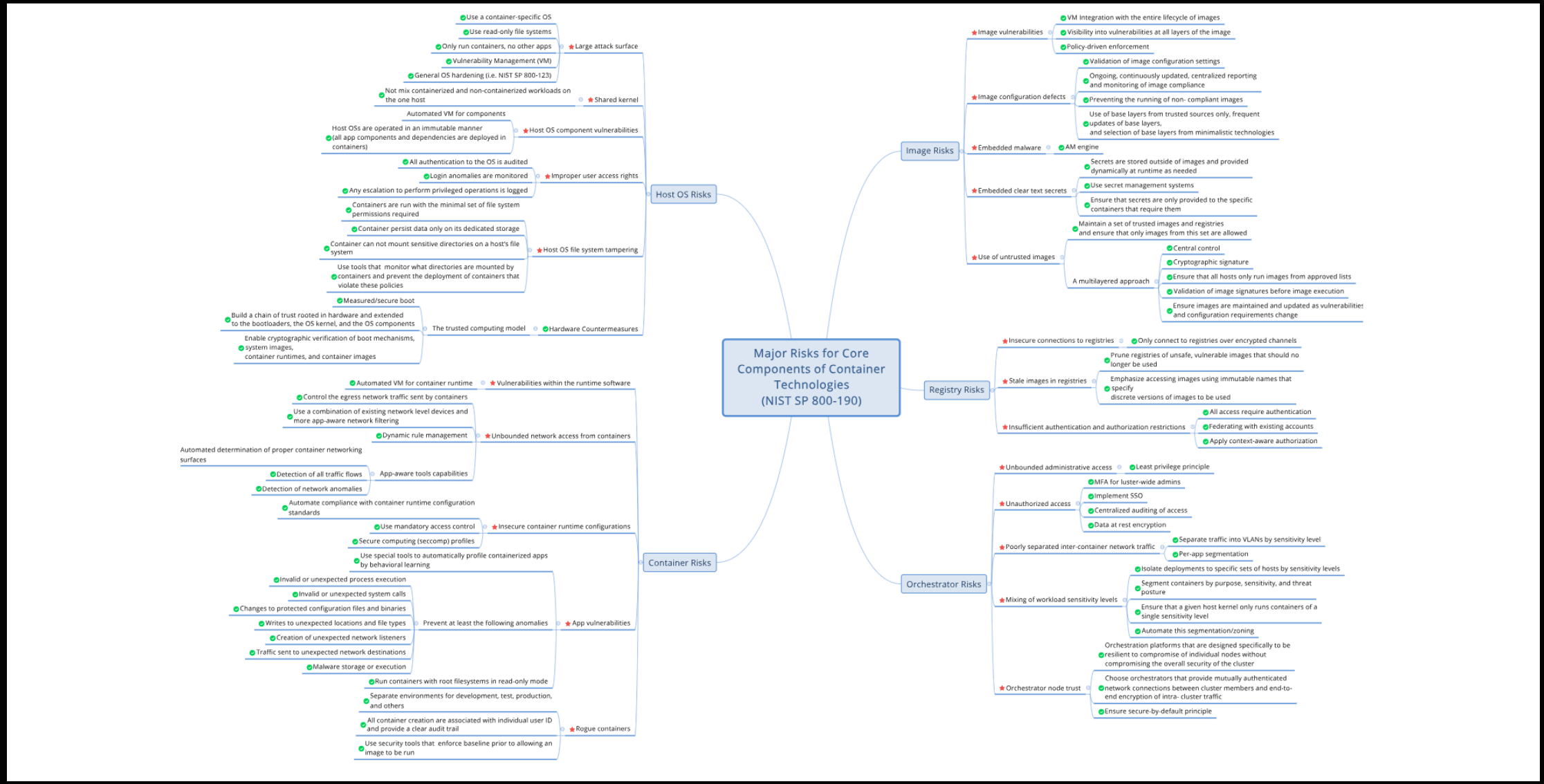
After bootstrapping, neither users nor components should authenticate to the Kubernetes API as `system:masters`. Similarly, running all of kube-controller-manager as `system:masters` should be avoided. In fact, `system:masters` should only be used as a break-glass mechanism, as opposed to an admin user.

- Edit this page
- Create child page
- Create an issue
- Print entire section

Authentication & Authorization
Network security
Pod security
Enabling Seccomp
Enabling AppArmor or SELinux
Logs and auditing
Pod placement
Secrets
Images
Admission controllers
What's next

https://kubernetes.io/docs/concepts/security/

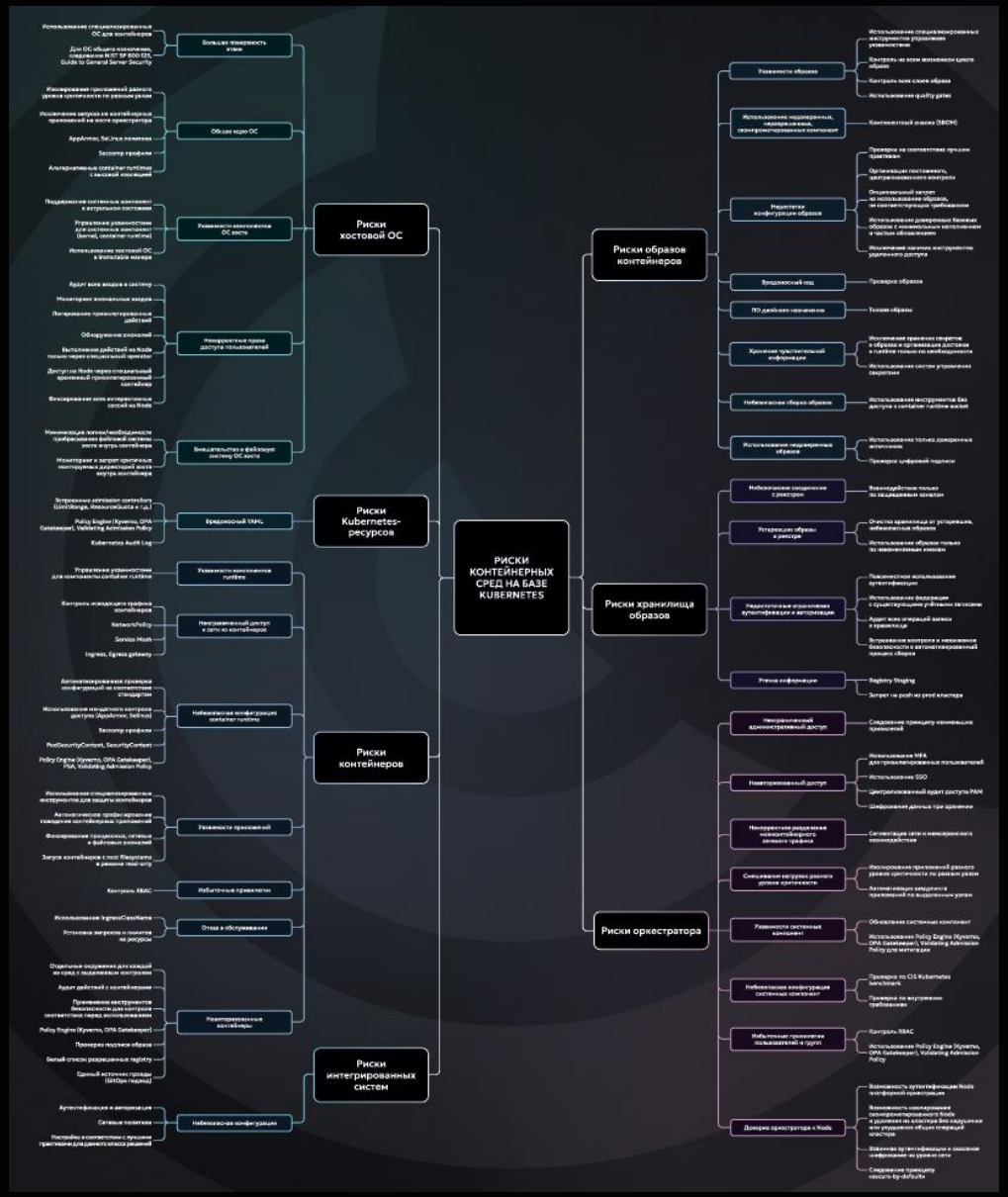
NIST SP 800-190 Application Container Security Guide



https://csrc.nist.gov/pubs/sp/800/190/final

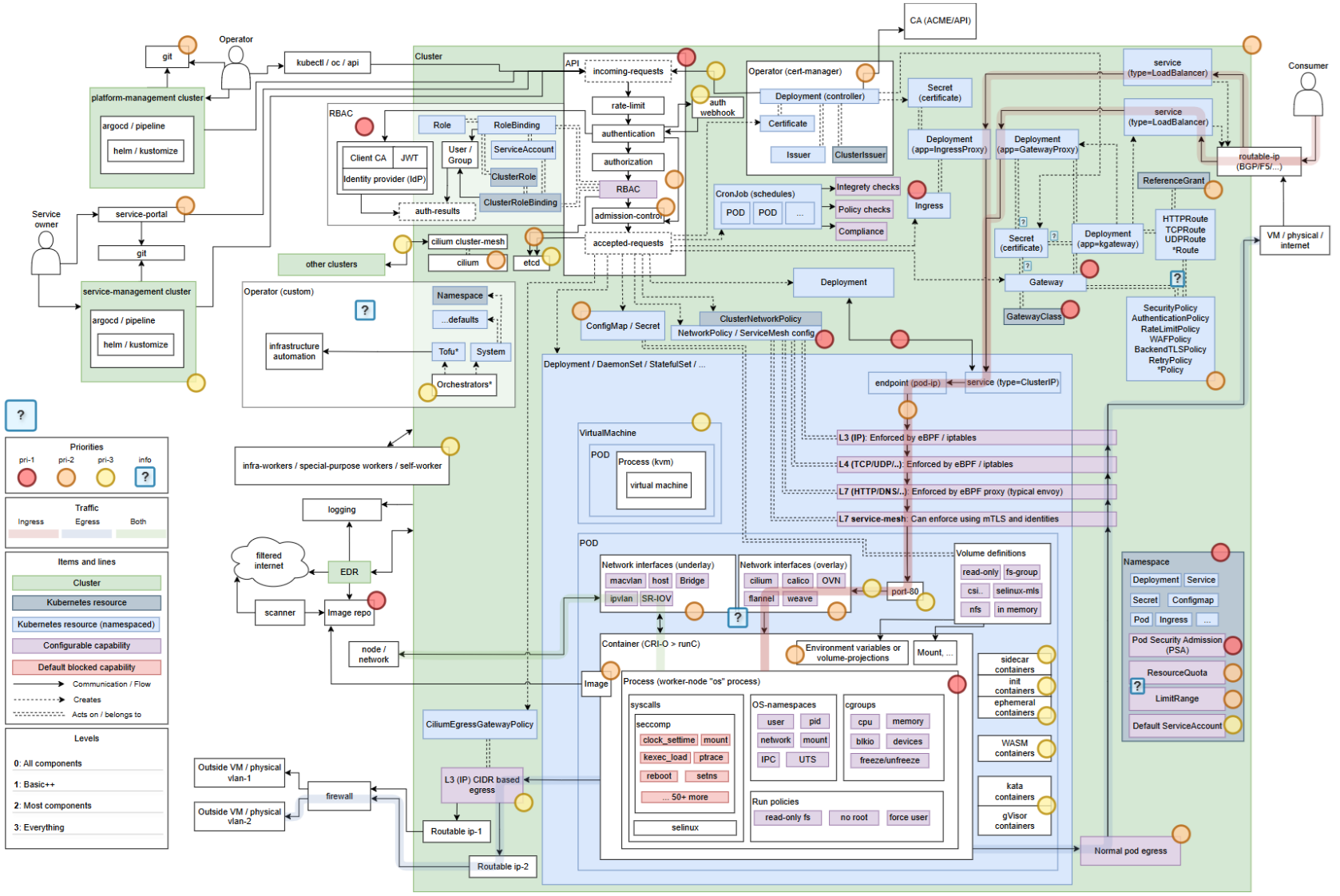
Карта безопасности для контейнерных сред

- **Версия от Luntry, с учетом NIST SP 800-190 и учетом специфики Kubernetes**
  - Риски
  - Механизмы защиты



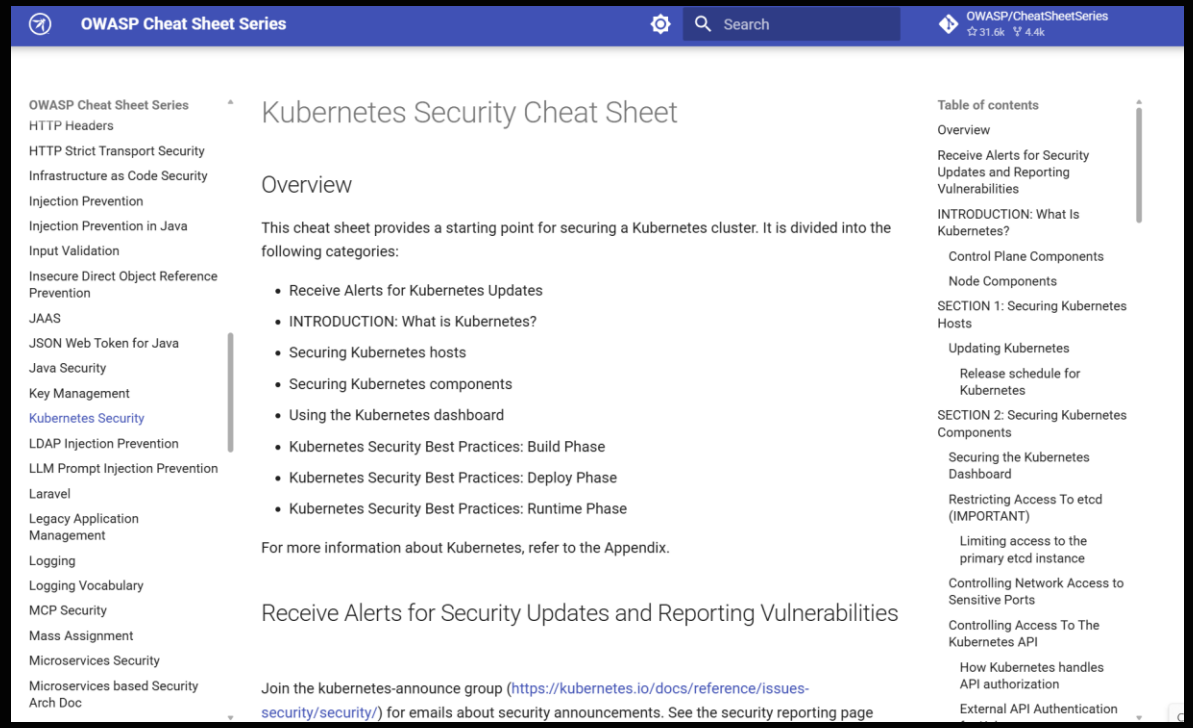
LUNTRY

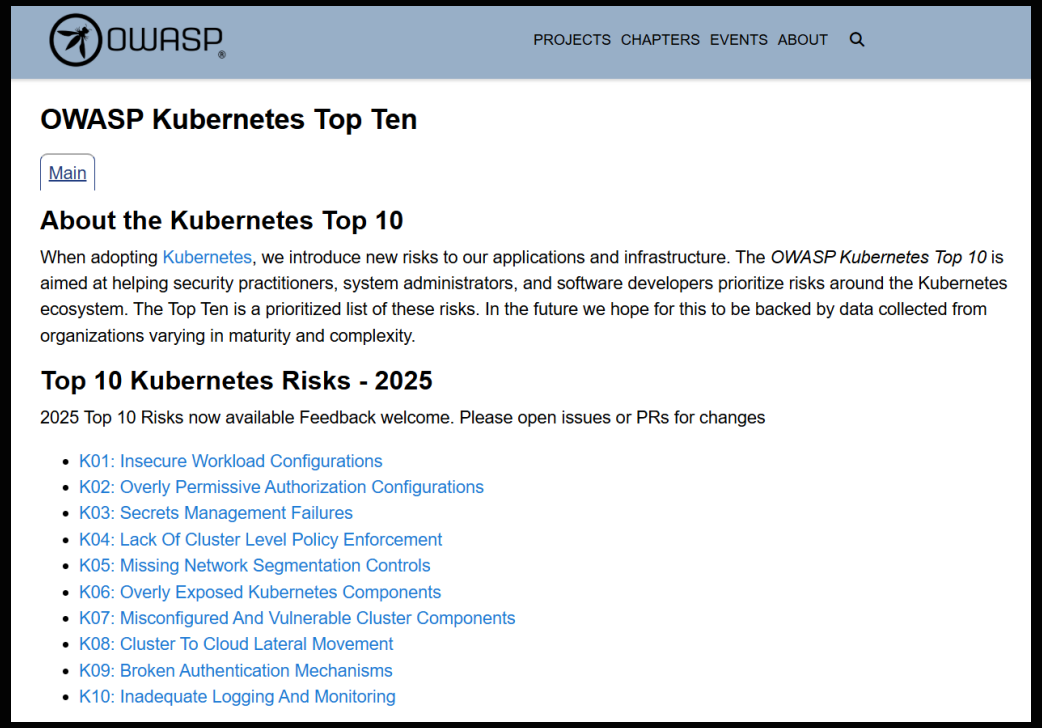
Диаграмма безопасности Kubernetes



https://github.com/kubesec-diagram/kubesec-diagram.github.io

OWASP и Kubernetes





https://cheatsheetseries.owasp.org/cheatsheets/Kubernetes_Security_Cheat_Sheet.html

https://owasp.org/www-project-kubernetes-top-ten/#

PCI DSS

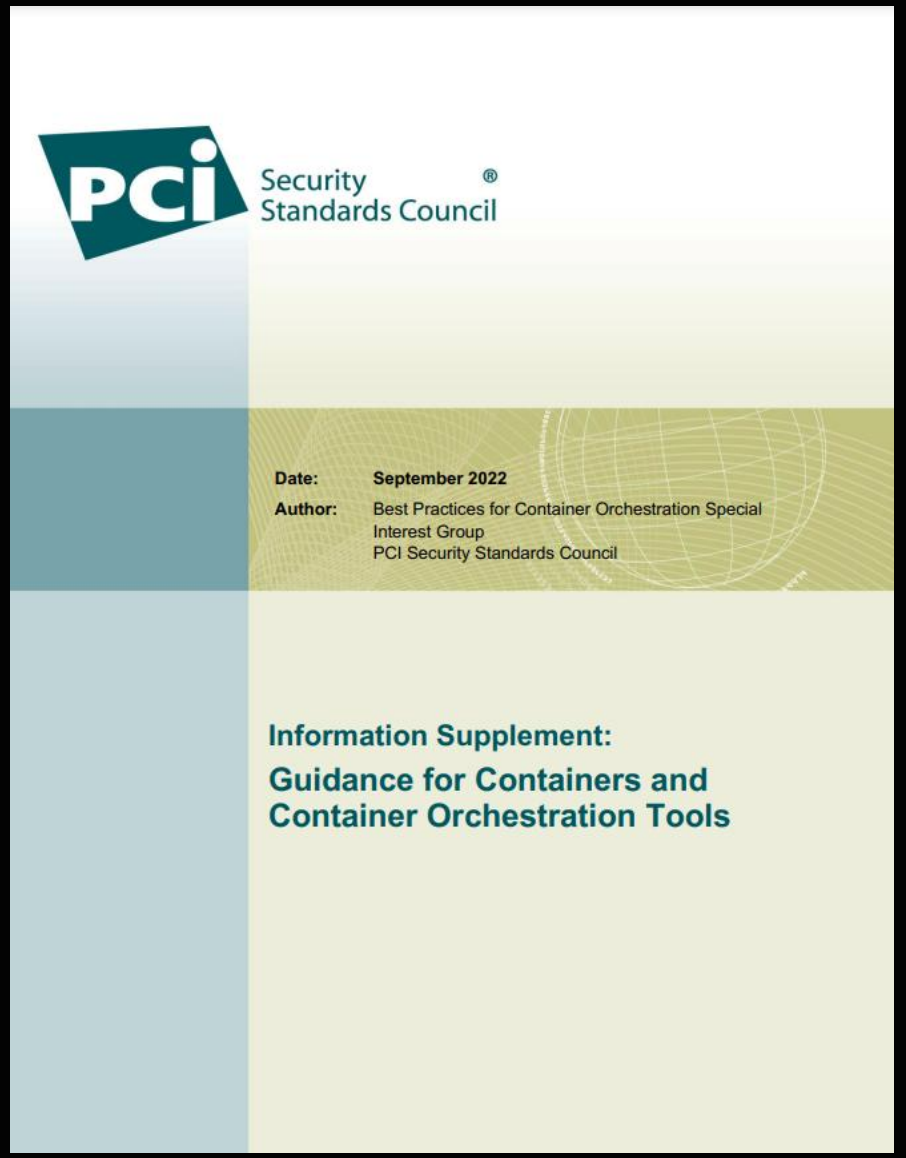
- НСПК требует соответствия PCI DSS
- Существует руководство по безопасности для контейнеров и оркестраторов контейнеров

Стандарт PCI DSS

В ПС «Мир» для обеспечения безопасности данных карт «Мир» используется международный индустриальный стандарт PCI Data Security Standard (PCI DSS)

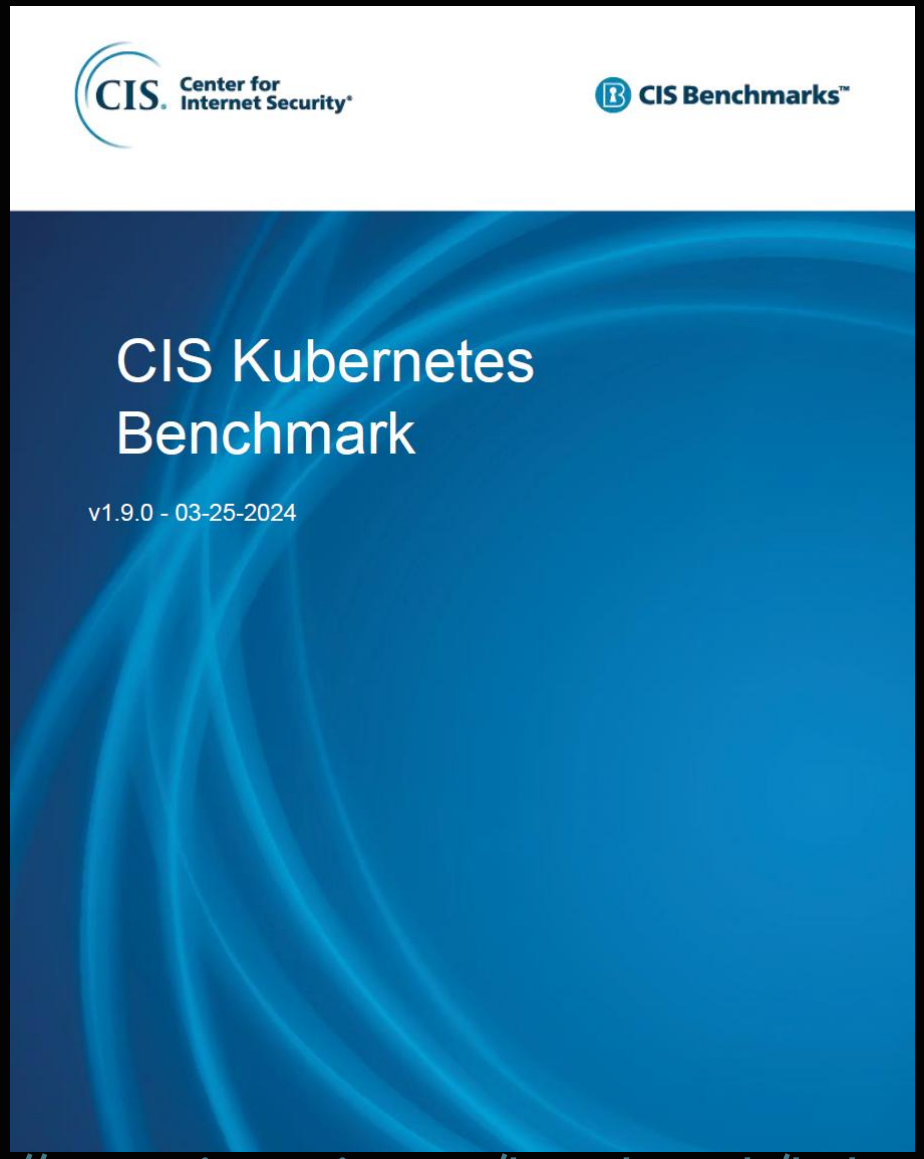
Этот стандарт должен применяться всеми организациями, которые хранят, обрабатывают и передают данные карт «Мир». К таким организациям относятся и торгово-сервисные предприятия, которые принимают к оплате карты «Мир».

Стандарт PCI DSS — это международный стандарт безопасности, созданный специально для защиты данных платежных карт. Он позволяет защитить организацию от инцидентов безопасности и обеспечить необходимый уровень защищенности во всей платежной системе.



LUNTRY

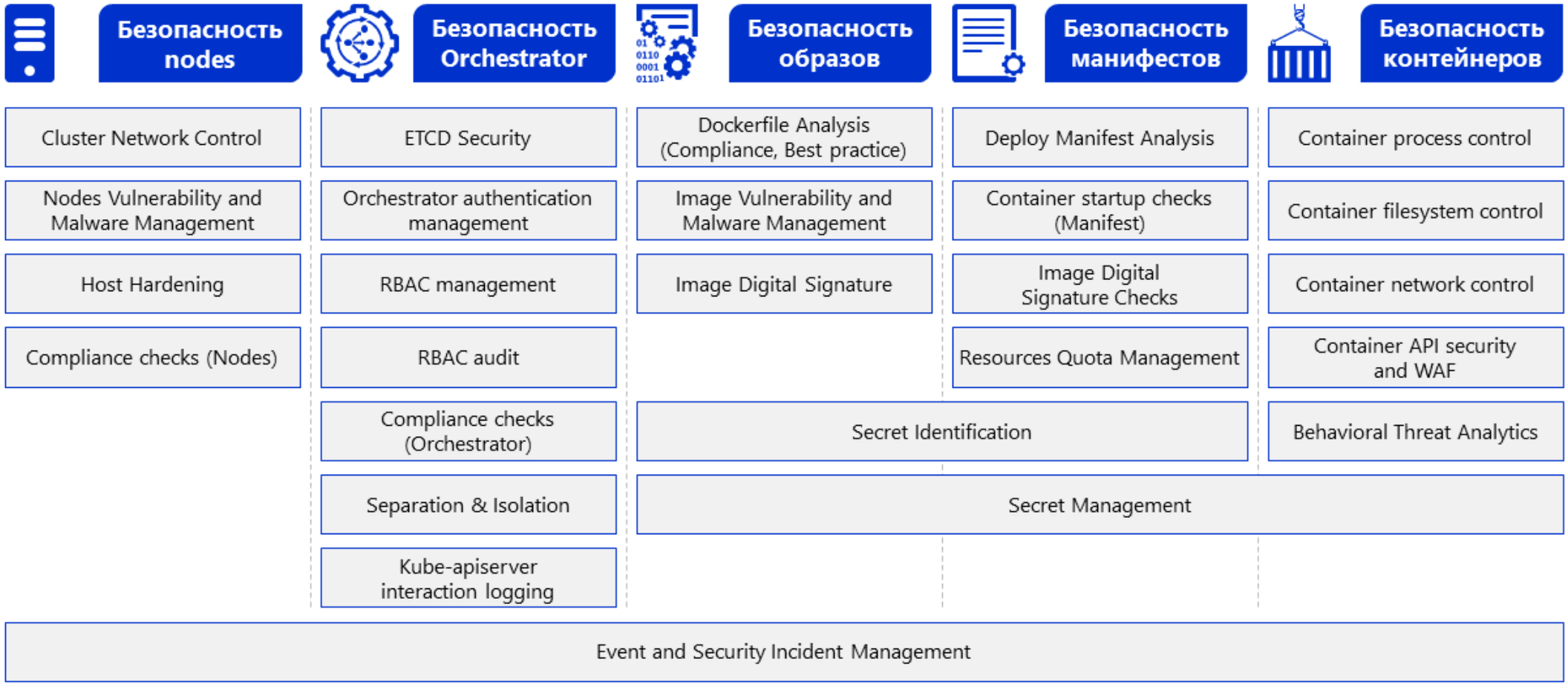
CIS Kubernetes Benchmark



https://www.cisecurity.org/benchmark/kubernetes

LUNTRY

Jet Container Security Framework (JCSF)



https://github.com/Jet-Security-Team/Jet-Container-Security-Framework

Регуляторика в РФ

- 118 приказ ФСТЭК
  - Требования по безопасности информации к средствам контейнеризации
- 117 приказ ФСТЭК
  - Пункт про защиту технологий контейнерных сред и их оркестрации
- Информационное сообщение ФСТЭК России 240/24/38
  - О повышении безопасности средств защиты информации, в состав которых разработчики включают средства контейнеризации или образы контейнеров
- Банк данных угроз безопасности информации от ФСТЭК
  - 5 УБИ посвящено контейнерам

LUNTRY

Kubernetes для SOC

Threat Actors (TA)	
ID	Description
TA01	External attacker.
TA02	Internal attacker.

In Scope Components/Processes

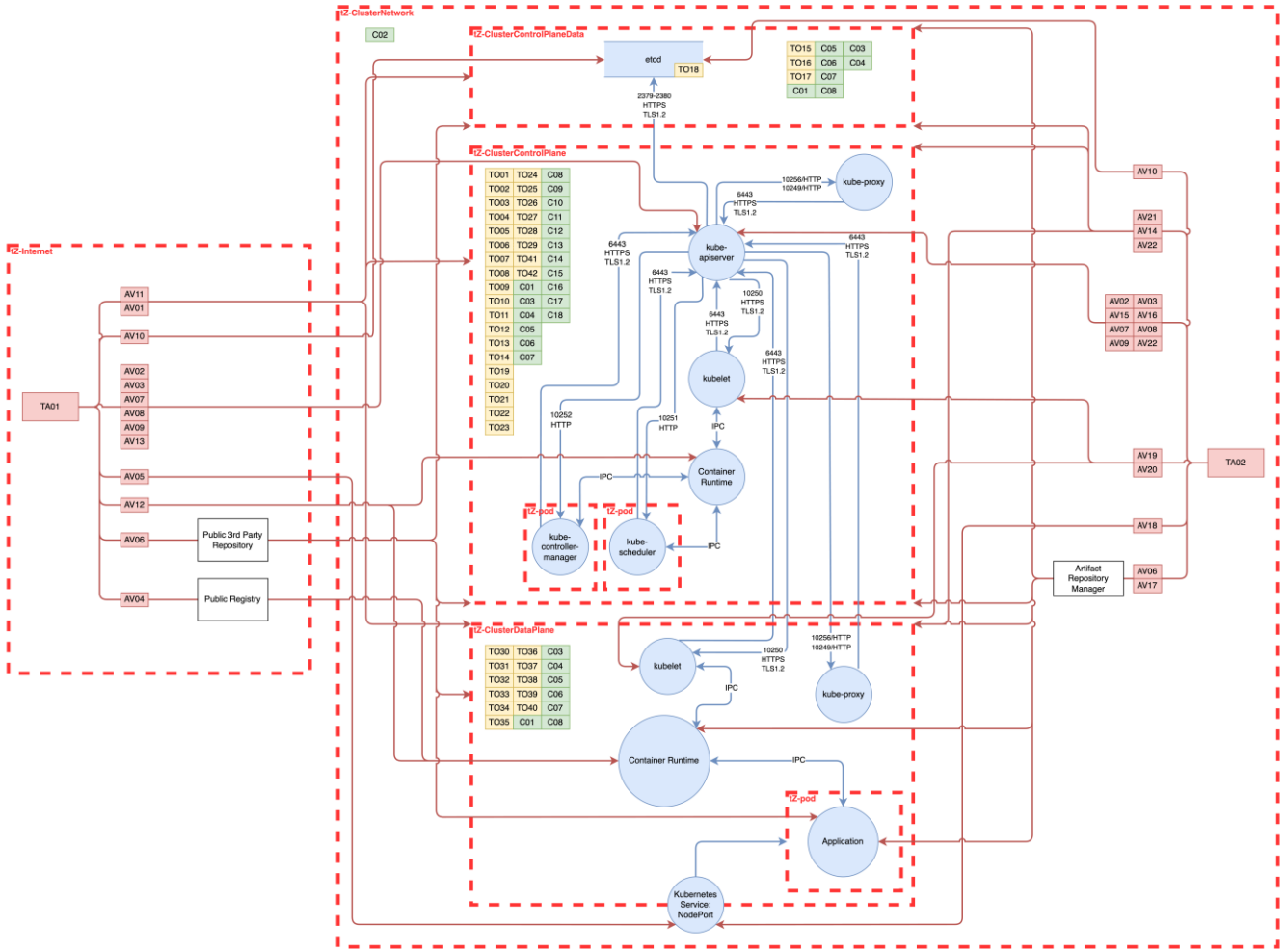
Out of Scope Components/ Processes

Threat Objects (TO)	
ID	Description
TO01	Cluster's CA x509 cert/key pair.
TO02	kube-apiserver's server x509 cert/key pair.
TO03	kube-apiserver's etcd-and-kubelet client x509 cert/key pairs.
TO04	Cluster's service accounts signing x509 cert/key pair.
TO05	kubernetes-admin's kubeconfig file: client x509 cert/key pair embedded.
TO06	kube-scheduler's kubeconfig file: client x509 cert/key pair embedded.
TO07	kube-controller-manager's kubeconfig file: client x509 cert/key pair embedded.
TO08	kube-apiserver's static pod manifest file.
TO09	kube-controller-manager's static pod manifest file.
TO10	kube-scheduler's static pod manifest file.
TO11	Cluster's audit log directory.
TO12	Cluster's audit policy file.
TO13	Cluster's encryption provider config file.
TO14	x509 certificate for the CA that signed OIDC's web certificate.
TO15	etcd's CA x509 cert/key pair.
TO16	etcd's server x509 cert/key pair.
TO17	etcd's data directory.
TO18	Cluster's desired state.
TO19	Master node's etcd's CA x509 cert.
TO20	Master node's kubelet's kubeconfig file: client x509 cert/key pair embedded.
TO21	Master node's kubelet' service file.
TO22	Master node's kubelet's binary.
TO23	Master node's kubelet's config file.
TO24	Master node's kubelet's directory.
TO25	Master node's kube-proxy's kubeconfig file: client x509 cert/key pair embedded.
TO26	Master node's kube-proxy' service file.
TO27	Master node's kube-proxy's binary.
TO28	Master node's kube-proxy's config file.
TO29	Master node's CNI directories.
TO30	Worker node's kubelet's kubeconfig file: client x509 cert/key pair embedded.
TO31	Worker node's kubelet' service file.
TO32	Worker node's kubelet's binary.
TO33	Worker node's kubelet's config file.
TO34	Worker node's kubelet's directory.
TO35	Worker node's cluster's CA x509 cert.
TO36	Worker node's kube-proxy's kubeconfig file: client x509 cert/key pair embedded.
TO37	Worker node's kube-proxy' service file.
TO38	Worker node's kube-proxy's binary.
TO39	Worker node's kube-proxy' config file.
TO40	Worker node's CNI directories.
TO41	Cluster's admission control config file.
TO42	Cluster's authorization webhook config file.

Attack Vectors (AV)	
ID	Description
AV01	Publicly exposed host with vulnerable non-K8s endpoint(s)/service(s)/port(s).
AV02	Compromised kubeconfig file.
AV03	Compromised K8s service account token with cluster details.
AV04	Malicious container image in public registry.
AV05	Publicly exposed vulnerable K8s workload: e.g. RCE.
AV06	Malicious 3rd party dependency.
AV07	kube-apiserver vulnerability.
AV08	kube-apiserver's options configured incorrectly.
AV09	kube-apiserver's authorization configured incorrectly.
AV10	etcd's authentication configured incorrectly.
AV11	Compromised credentials of publicly exposed host.
AV12	Container runtime configured incorrectly.
AV13	Compromised OIDC token with cluster details.
AV14	Host with vulnerable non-K8s endpoint(s)/service(s)/port(s).
AV15	Legitimately obtained kubeconfig file.
AV16	Legitimately obtained K8s service account token with cluster details.
AV17	Malicious container image in private registry.
AV18	Vulnerable K8s workload: e.g. RCE.
AV19	kubelet vulnerability.
AV20	kubelet's options configured incorrectly.
AV21	Compromised credentials of host.
AV22	Legitimately obtained OIDC token with cluster details.
AV22	Legitimately obtained credentials of host.

Assumptions (AS)	
ID	Description
AS01	Fault-tolerance and high availability are not in scope for this model
AS02	Two certificate authorities (CAs) are in scope for this model: one for the cluster and one for etcd.
AS03	Within this model, the kube-proxy is running as a service in the host and using a config file.
AS04	Container Runtime installation and related files are out of scope for this model.
AS05	Within this model, there are no business-type K8s workloads scheduled in the master node.
AS06	Within this model, K8s admission controllers include: ResourceQuota, LimitRanger, PodSecurity(Policy), and ImagePolicyWebhook.
AS07	Within this model, K8s authorisation is limited to: RBAC, Node, and Webhook.
AS08	Within this model, K8s encryption is implemented via EncryptionConfiguration and applies for data at rest only.
AS09	Within this model, K8s MutatingAdmissionController enables: disable service account token auto-mount, PodSecurityContext/ContainerSecurityContext, AppArmor, container sandboxing, and application logging consumption.
AS10	Within this model, K8s authentication is limited to: client certificates, service account tokens, and OIDC tokens.
AS11	AV02, AV03, AV07, AV08, AV09, AV10, AV12, and AV13 are being taken into account under the assumption that the hosts holding the cluster control plane's components are publicly exposing them.

Controls	
ID	Description
C01	C-Harden-Component
C02	C-Control-Traffic-Flow
C03	C-Harden-Config-Compl-Mon / C-Mon-Drift
C04	C-Evt-Log-App / C-Evt-Log-Sec / C-Evt-Log-Sys
C05	C-Mon-File-Integrity
C06	C-Mon-Log
C07	C-Mon-Health
C08	C-Vuln-Scan
C09	C-K8s-Harden-Component
C10	C-K8s-AdmissionControllers
C11	C-K8s-AuditLogs
C12	C-K8s-Authorisation
C13	C-K8s-CNI-NetworkPolicies
C14	C-K8s-MutatingAdmissionController
C15	C-K8s-ValidationAdmissionController
C16	C-K8s-Encryption
C17	C-K8s-ContainerRuntimeSecurity
C18	C-K8s-Authentication



https://github.com/abdullahgarcia/kubernetes-for-soc

CNCF Security Technical Advisory Group

Security Technical Advisory Group

CLOUD
NATIVE
SECURITY

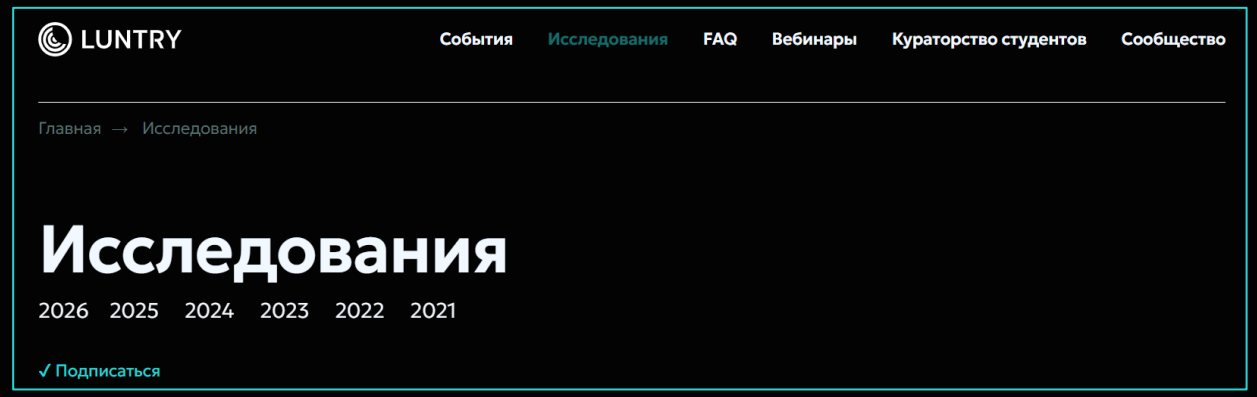
https://github.com/cncf/tag-security

LUNTRY

Полезности

Материалы Luntry

- **Исследования**



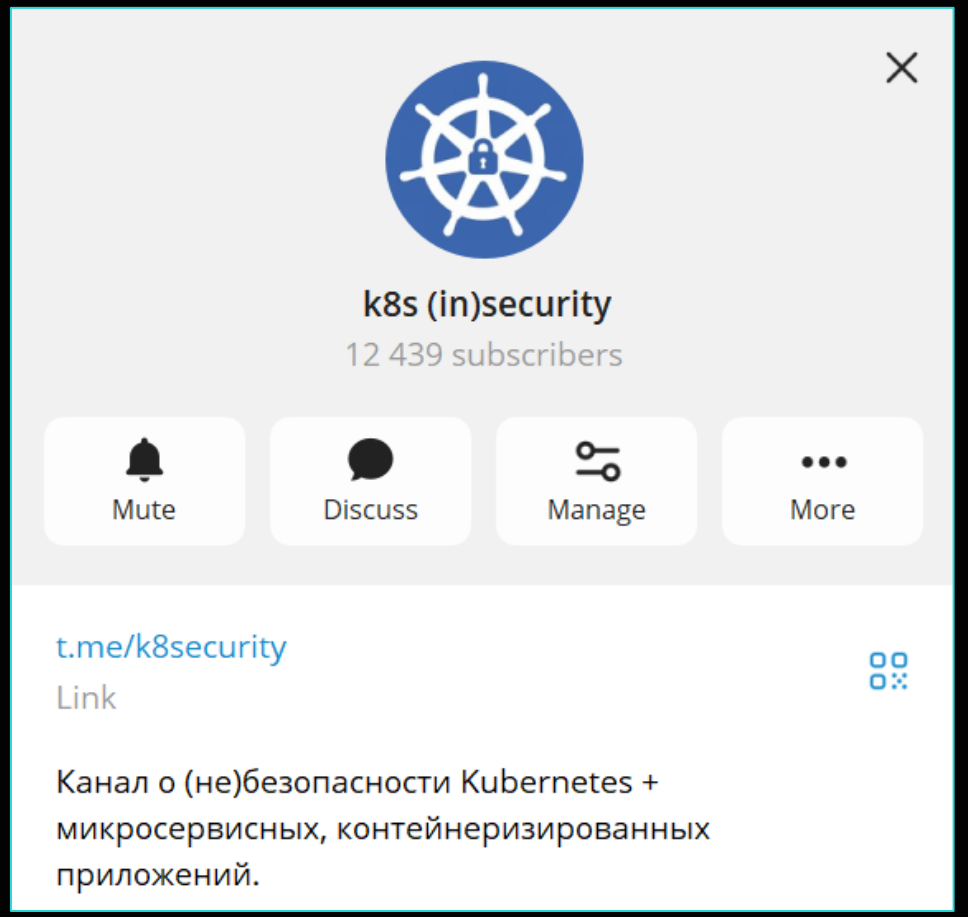
- **Кураторство студентов**

Кураторство и помощь в научно-исследовательской деятельности от Luntry

Команда Luntry специализируется на безопасности контейнеров и Kubernetes. Мы занимаемся разработкой, настройкой cloud-native окружений, атакой и защитой информационных систем на базе контейнеров. За это время наши специалисты накопили опыт и знания, которыми теперь готовы делиться с теми, кому интересна сфера!

Если вы еще учитесь, но уже хотите работать с актуальными задачами по контейнеризации для своей курсовой, диплома, магистерской или докторской работы или же делаете исследование в этой сфере — мы можем вам помочь.

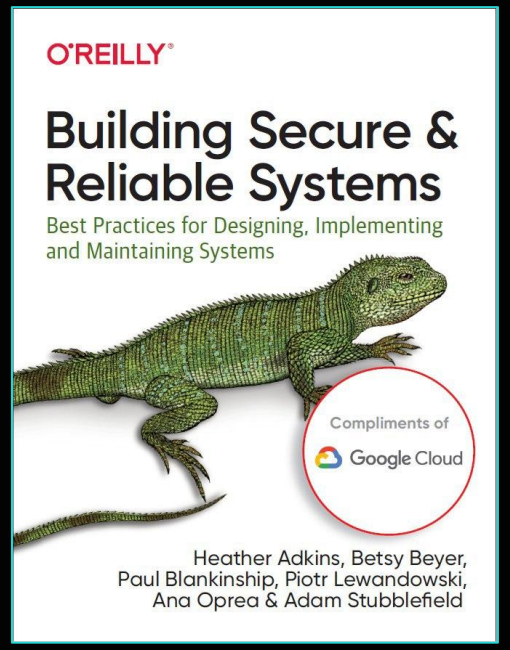
- **Telegram канал**

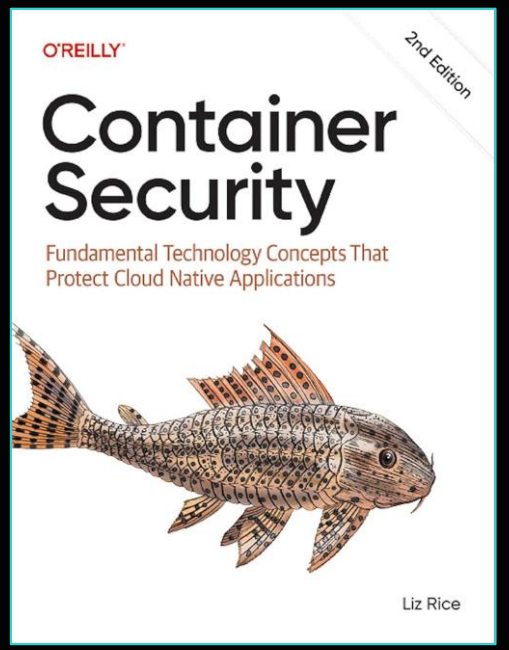


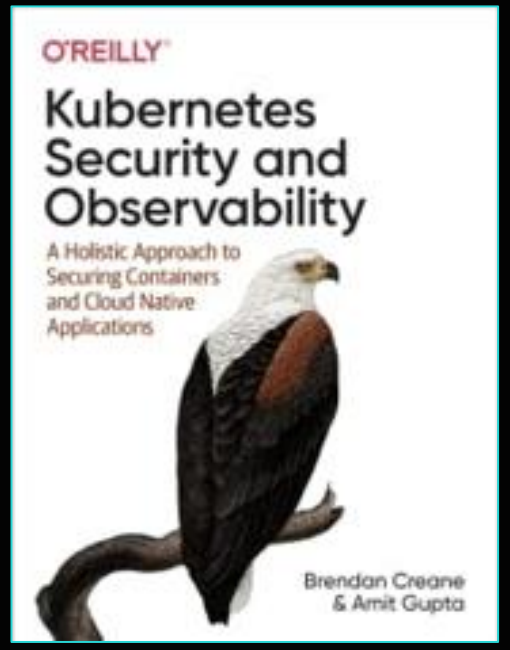
LUNTRY

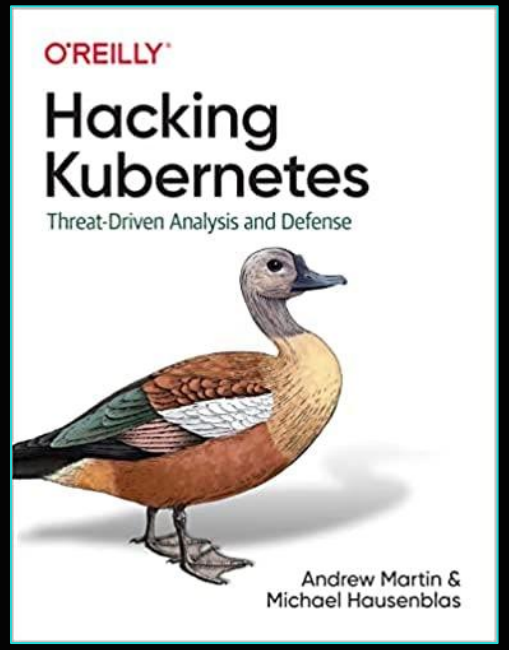
Полезные книги

- “Building Secure & Reliable Systems”
- “Container Security”
- “Kubernetes Security and Observability”
- “Hacking Kubernetes: Threat-Driven Analysis and Defense”









Полезные конференции

- БеКон
- Fwd:cloudsec
- DefCon Cloud Village
- Cloud Native Security Day
- KubeCon и CloudNativeCon

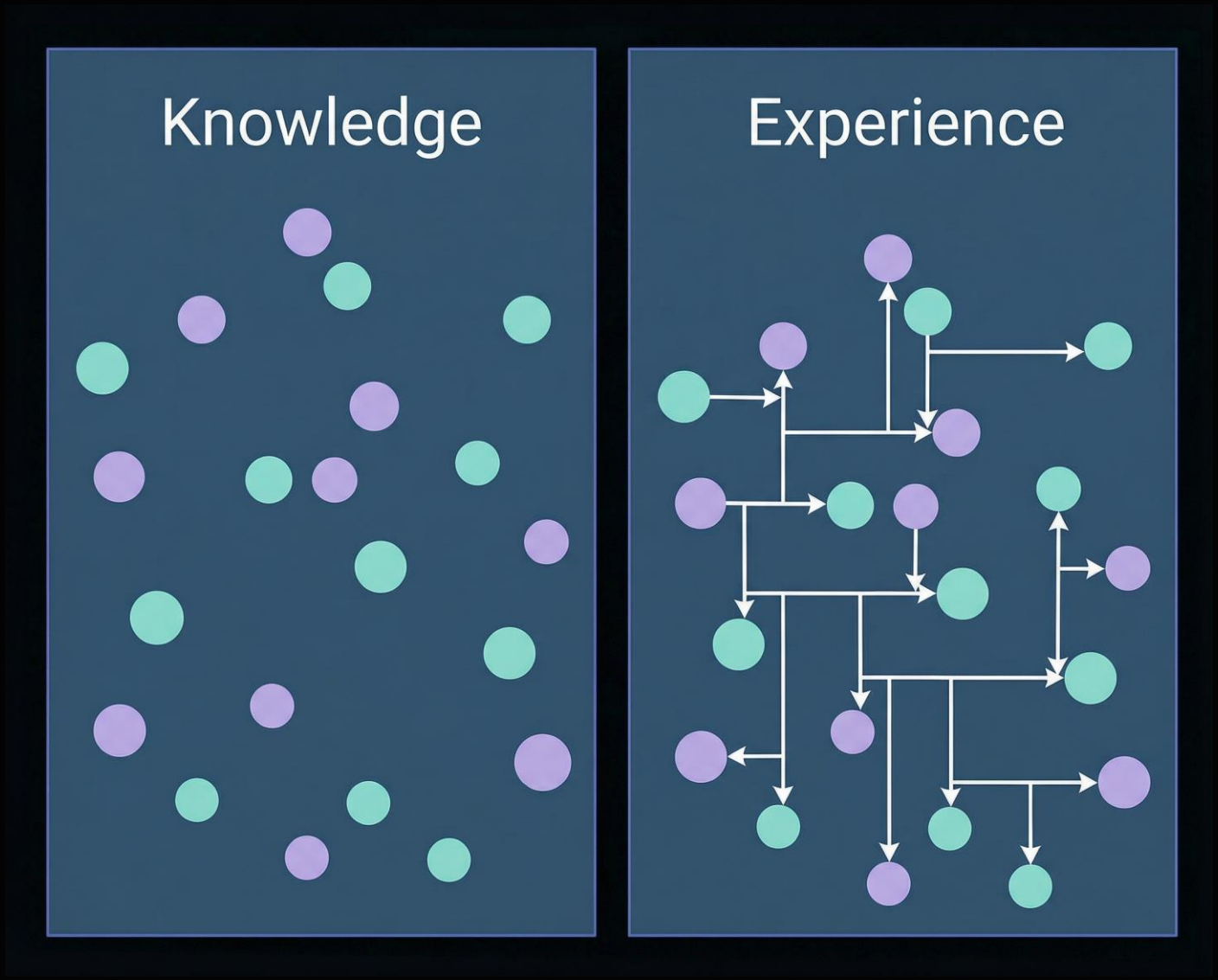
БЕКОН

Практика и лабы

- Docker Security: A Practical Guide
- Damn Vulnerable Kubernetes Application (DVKA)
- Docker Escape Labs
- K8s Best Security Practices
- Kubernetes Goat
- K8sQuest
- Breaching Bare Metal Kubernetes Clusters
- CKS
- Kubernetes LAN Party
- The Cloud Hunting Games CTF
- Awesome Cloud Security Labs

LUNTRY

Все приходит с опытом ;)



LUNTRY

Выводы

Выводы

01 В современном мире без контейнеров и Kubernetes уже никуда

02 Данная тема большая, серьезная и требует отдельного внимания

03 Данная тема имеет достаточно много специфики

04 На сегодняшний день это уже мейнстрим и доступно много информации, но тема быстро развивается

cyber camp '25

Дмитрий Евдокимов

Founder & CTO Luntry

НА СВЯЗИ

de@luntry.ru

@Qu3b3c

@k8security

@luntry_official