Microservices and software development infrastructure upgrade

Stefano Bovina

I.N.F.N. Information System (A.K.A. sysinfo)

Agenda

Why an upgrade?

Microservices and related challenges

Infrastructure overview

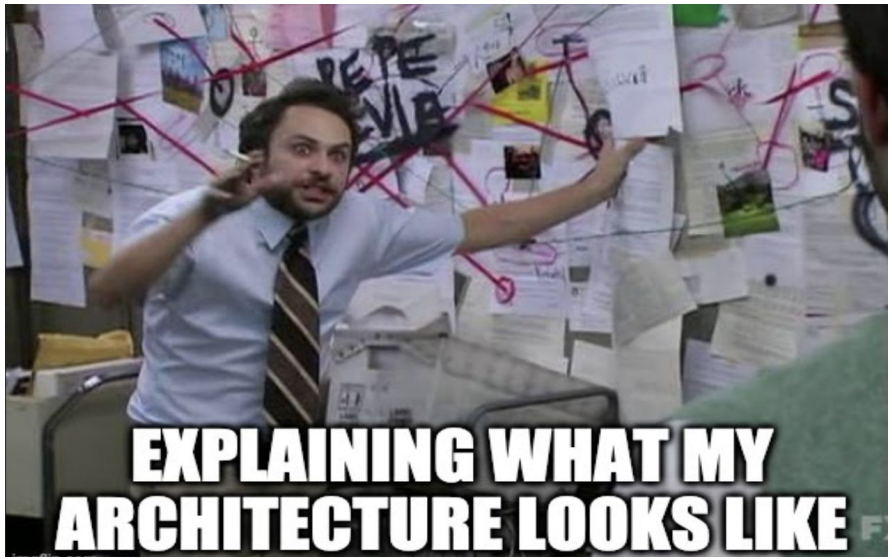
Infrastructure architecture deep dive

Security management

“New software” highlights

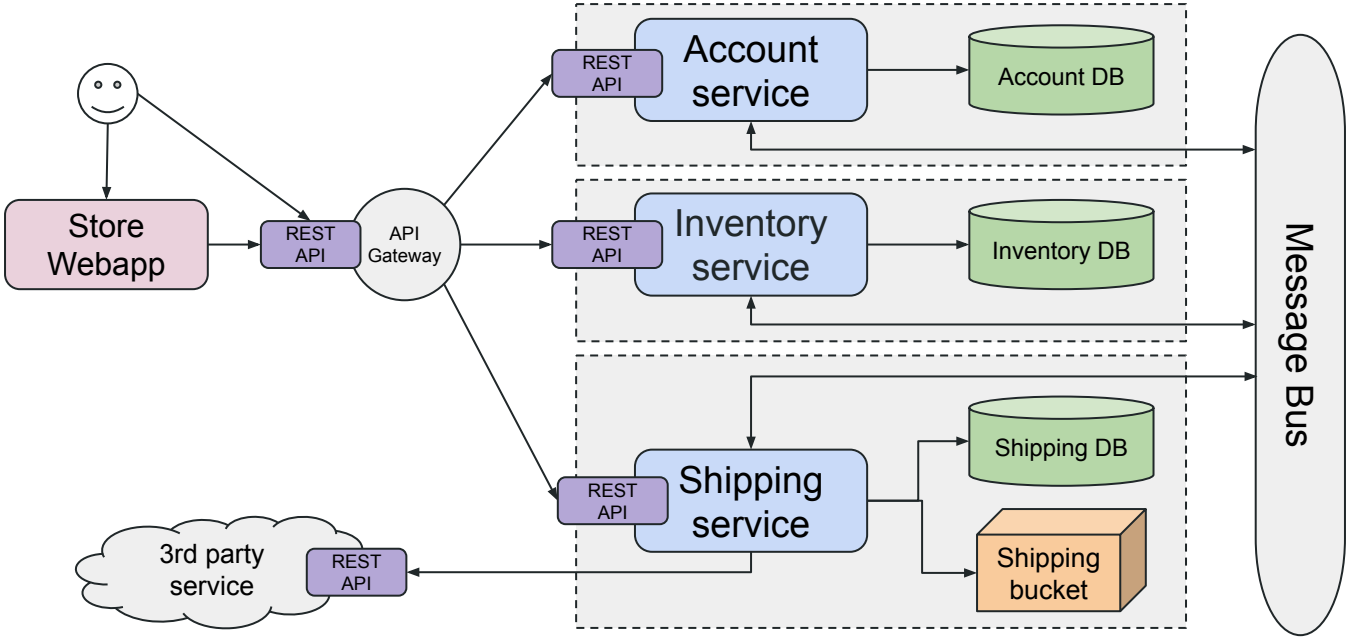
Why an upgrade?

- A lot of legacy applications need to be replaced or redesigned from scratch
- To produce “better” software
- To improve management of security and compliance aspects
- Technological upgrade
- For a better microservice management



Microservices

*"...the microservice architectural style is an approach to developing a single application as a **suite of small autonomous services**, each **running in its own process** and communicating with lightweight mechanisms, often an HTTP resource API. These services are **built around business capabilities** and **independently deployable** by fully automated deployment machinery..."*



Microservices

Pro:

- Strong Module Boundaries: Microservices reinforce modular structure
- Independent deployment
- Higher degree of organizational autonomy
- Technology Heterogeneity
- Optimized for replaceability
- Scaling independently
- Leads to Improved Fault Tolerance (if we understand and plan for failures)
- Ease of understanding of the codebase of the software system
- Isolation of data and isolation of processing around that data

Cons:

- Distribution: Distributed systems are harder to program, since remote calls are slow and are always at risk of failure
- Eventual Consistency: Maintaining strong consistency is extremely difficult for a distributed system, which means everyone has to manage eventual consistency (Multi-services transactions/changes are complex)
- Operational Complexity: You need a mature operations team to manage lots of services, which are being redeployed regularly
- Global automated testing is more complicated

Distributed computing fallacies and other requirements

Fallacies/requirements	Solutions
The network is reliable	Circuit breaker (Resilience4j), retry and timeout design pattern (Resilience4j), message queues (Kafka)
Latency is zero	caching strategy (Redis), bulk requests, placement/affinity policy (see Kubernetes policy)
Bandwidth is infinite	throttling policy, small payloads
The network is secure	firewall (network policy/micro segmentation), encryption (mTLS, Cert-manager), AuthN/AuthZ (OIDC/Oauth2)
Topology doesn't change	no hardcoded IPs, service discovery tools (see Kubernetes service discovery)
There is one administrator	DevOps culture
Transport cost is zero	standardized protocols like JSON, cost calculation
The network is homogeneous	Circuit breaker (Resilience4j), retry and timeout design pattern (Resilience4j)
Observability	Monitoring system (Prometheus/Grafana/Sensu/InfluxDB), Log aggregation (ELK)
Automation culture/tools	CI/CD platform (Gitlab, ArgoCD), IaC paradigm (Helm/Kustomize/Puppet)
Secret management	Vault

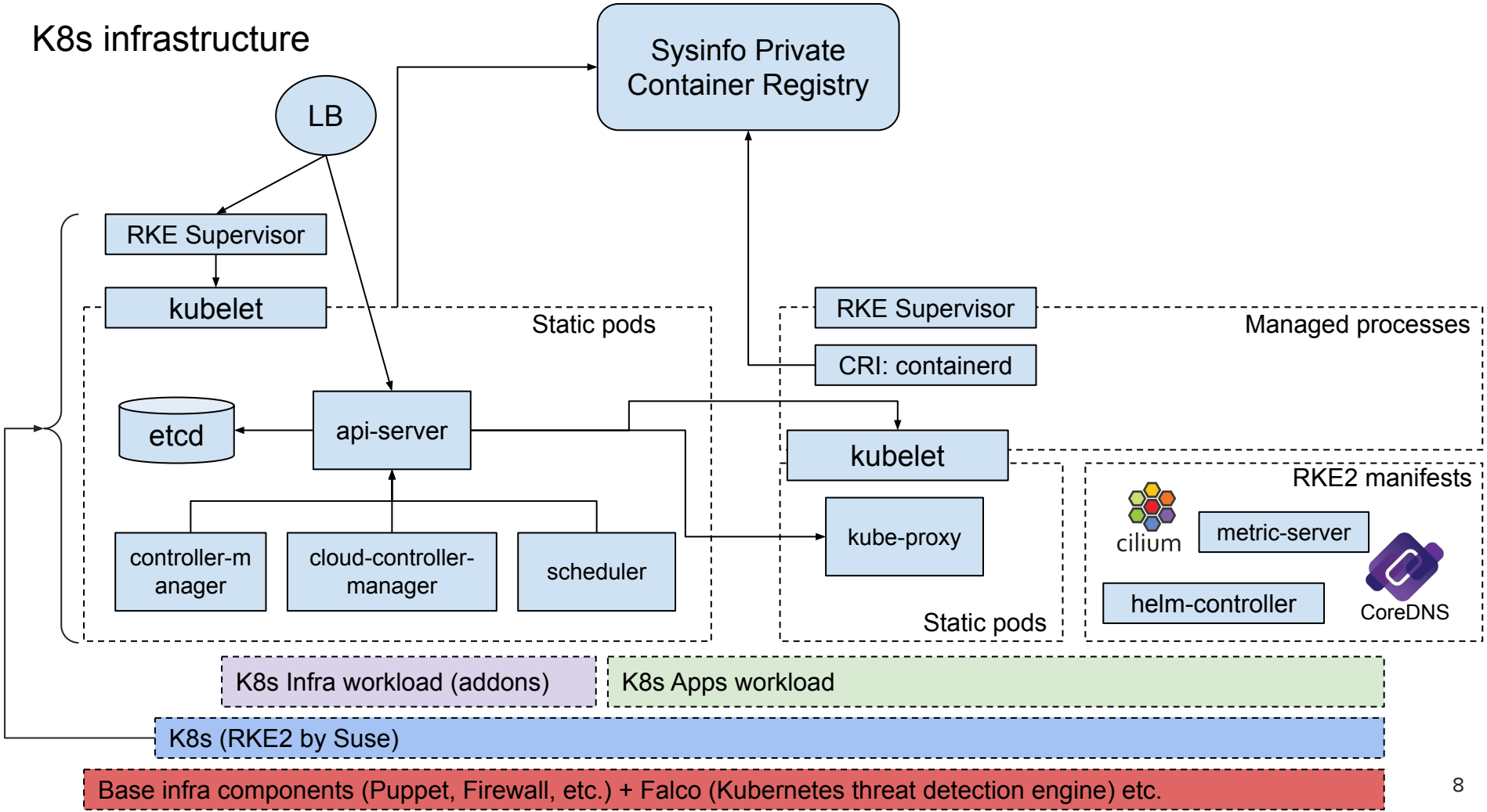
Fallacies

Kubernetes (k8s)

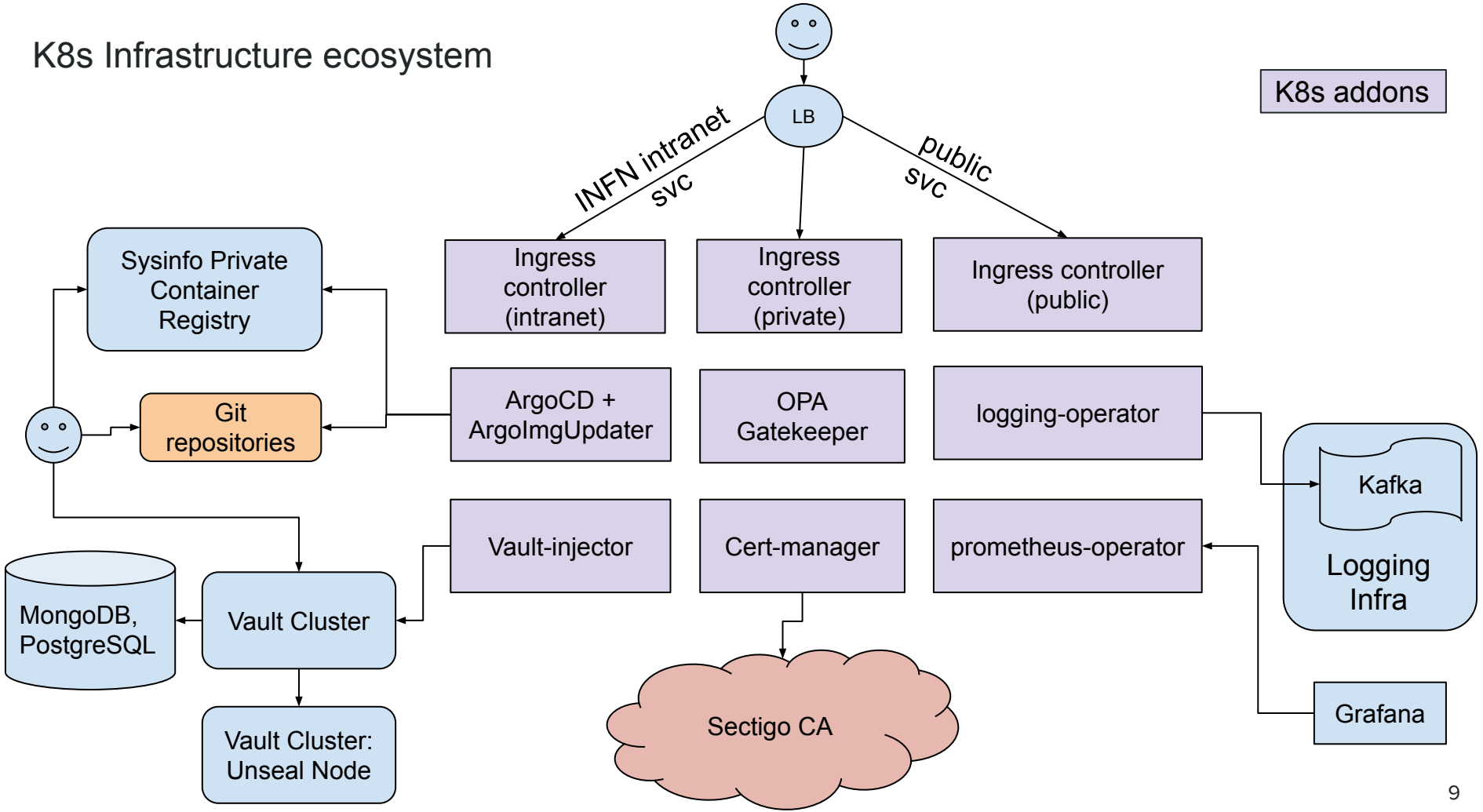
“ ...a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.”

- Google open-sourced the Kubernetes project in 2014
- Features:
  - Service discovery, load balancing, horizontal scaling
  - Self-healing
  - Automated rollouts and rollbacks
  - Secret and configuration management
  - etc.

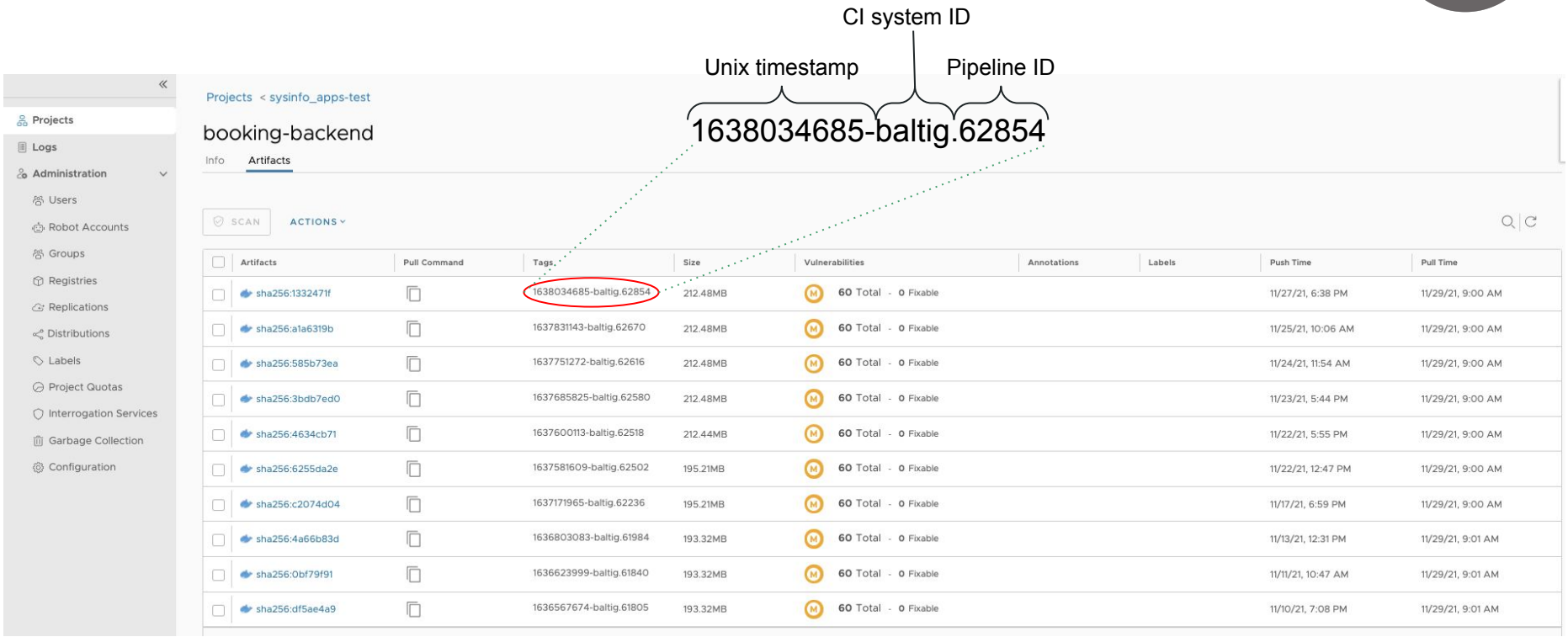
K8s infrastructure



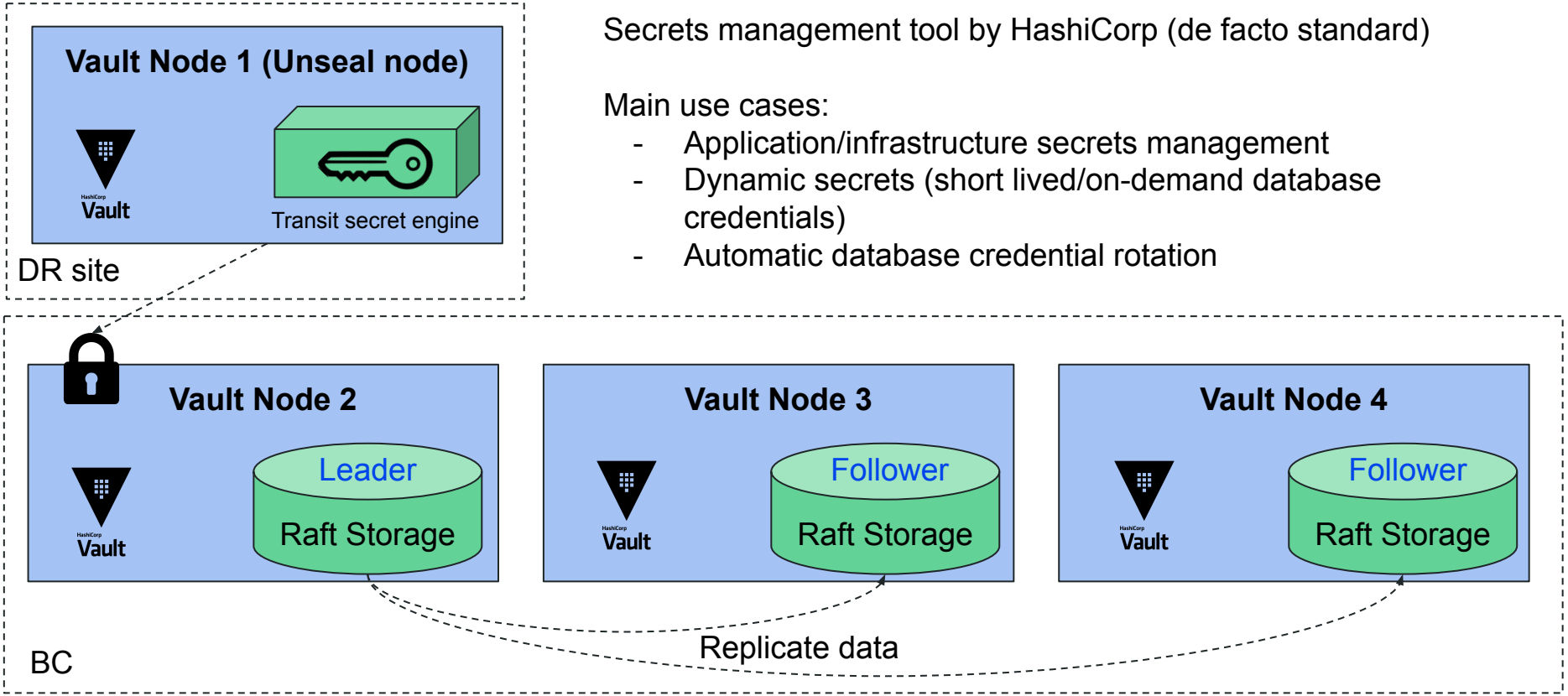
K8s Infrastructure ecosystem



Harbor container registry



Vault

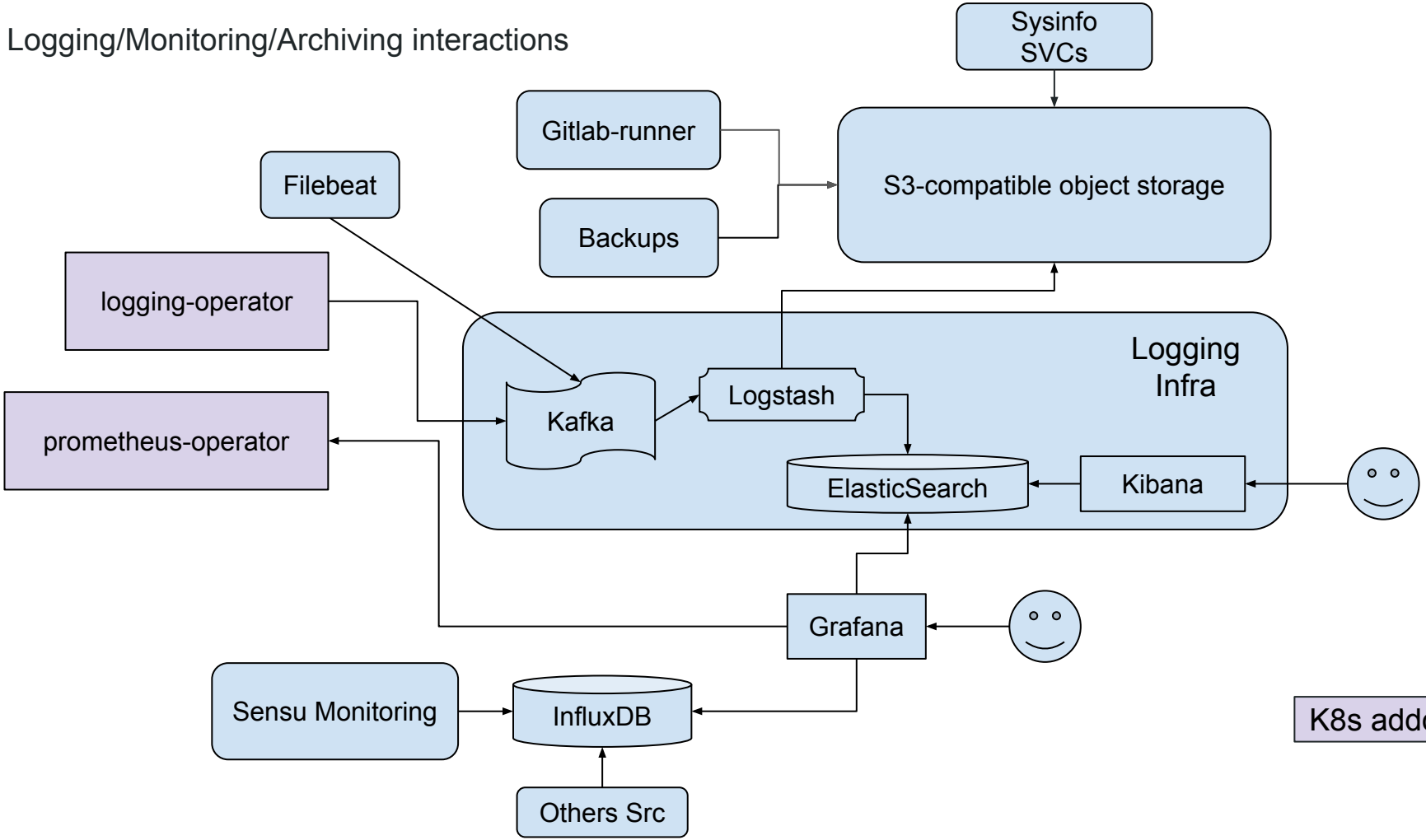


Secrets management tool by HashiCorp (de facto standard)

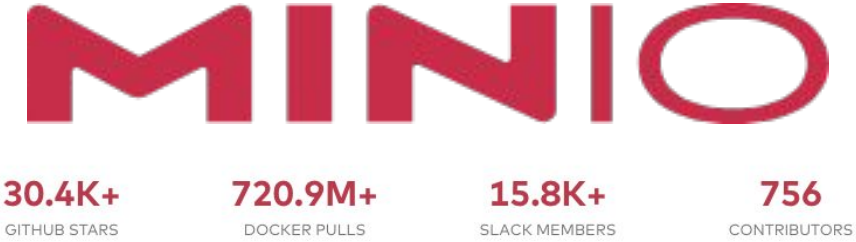
Main use cases:

- Application/infrastructure secrets management
- Dynamic secrets (short lived/on-demand database credentials)
- Automatic database credential rotation

Logging/Monitoring/Archiving interactions



Minio



Software-defined high performance object storage

Features/reasons why we use it:

- Highly available and horizontally scalable
- API compatible with Amazon's S3 (de-facto standard API for business applications to store unstructured data)
- Bucket Versioning
- Object Lock and Immutability - Write-Once Read-Many (WORM)
- Bucket Notifications (i.e. Kafka)
- Server-Side Bucket Replication (BC/DR)
- Object Lifecycle Management (Transition/Expiration)
- Encryption

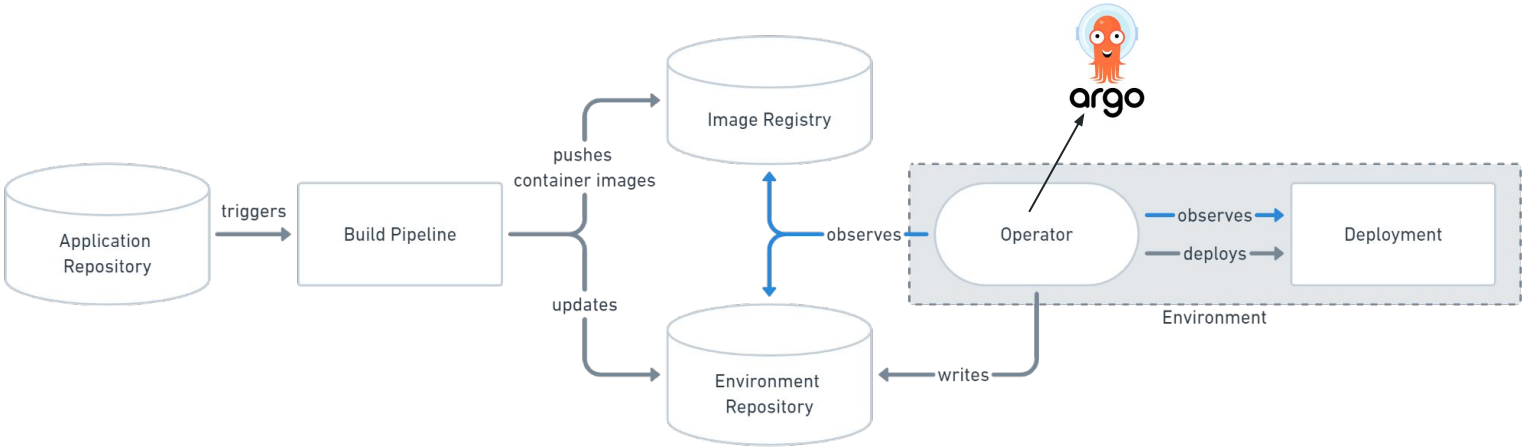
Use cases:

- gitlab-runner distributed cache
- Long term archiving (es: logs, backups etc.)
- Sysinfo application data

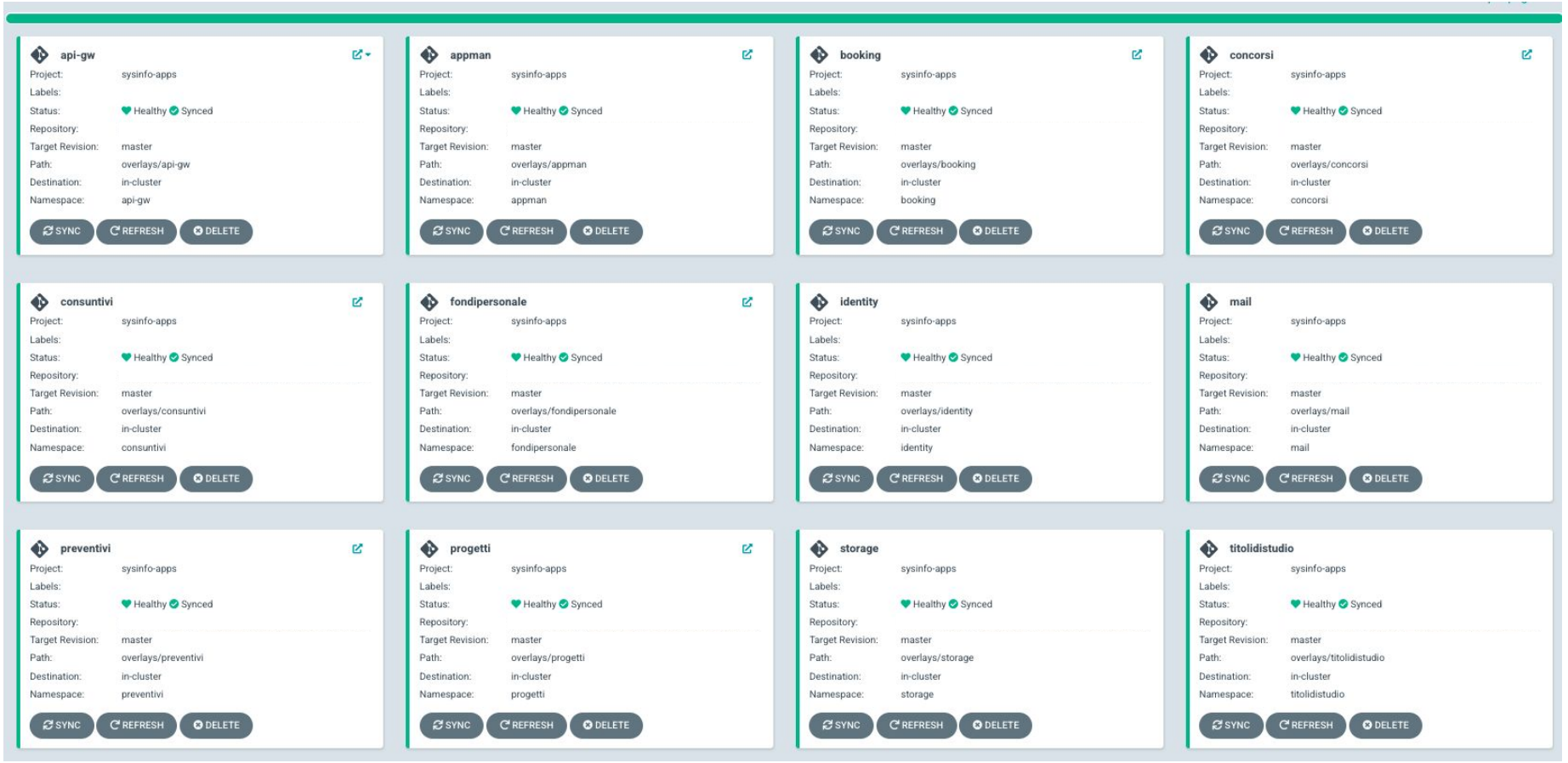
ArgoCD and GitOps

GitOps: versioned CI/CD on top of declarative infrastructure:

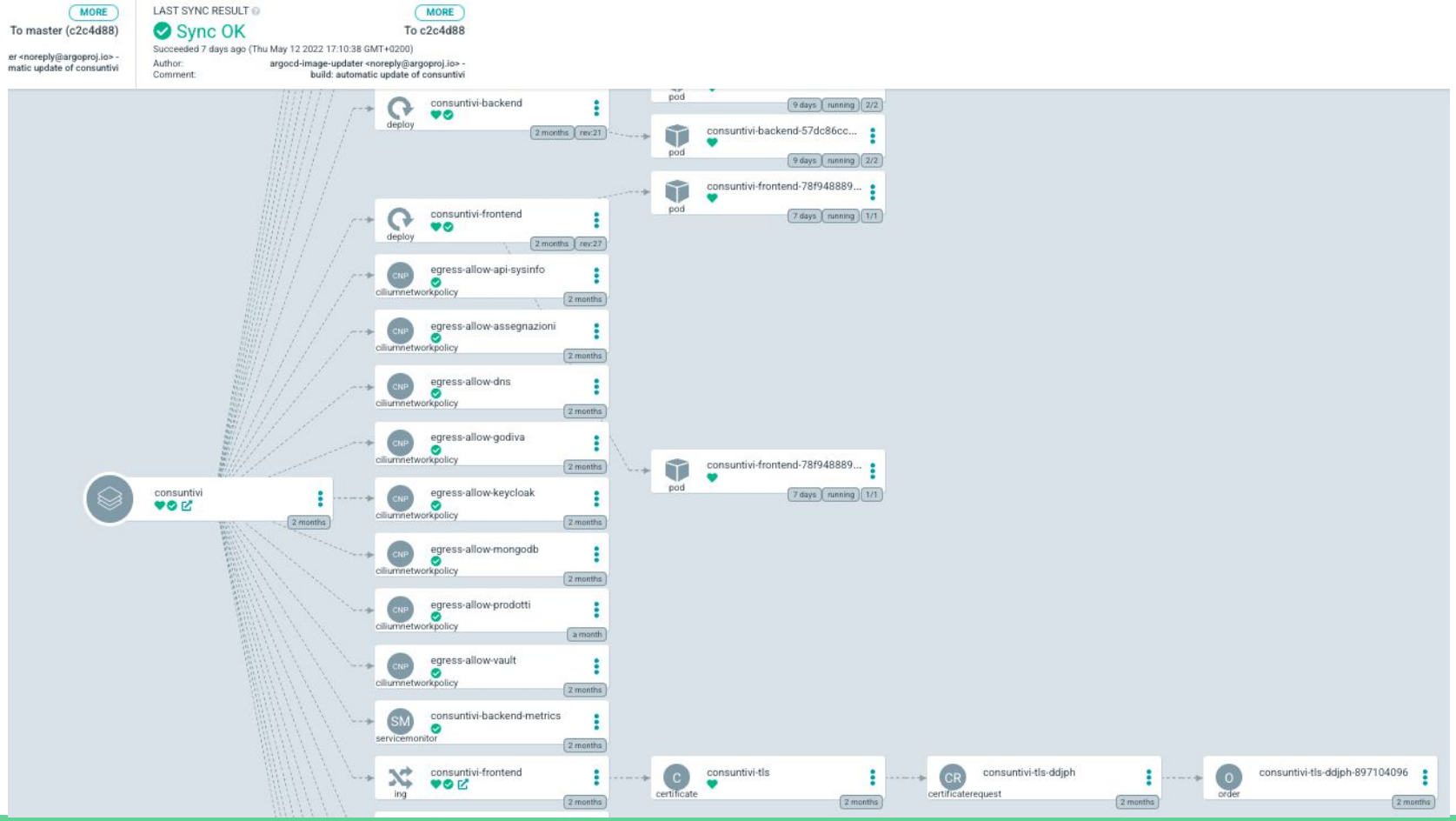
- a Git repository that always contains declarative descriptions of the desired infrastructure state
- infrastructure state versioned in Git
- use of continuous integration/continuous delivery pattern
- automated processes to make the production environment match the described state in the repository
- a change to the infrastructure (e.g. a new application) => modify the repository



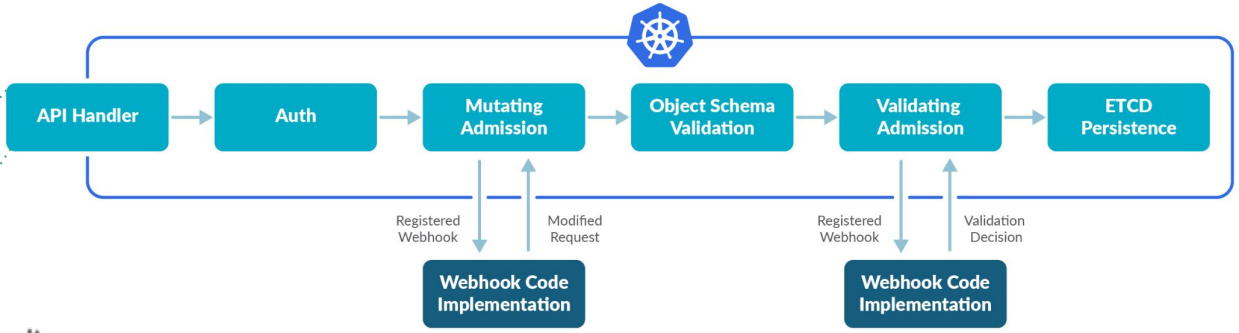
ArgoCD

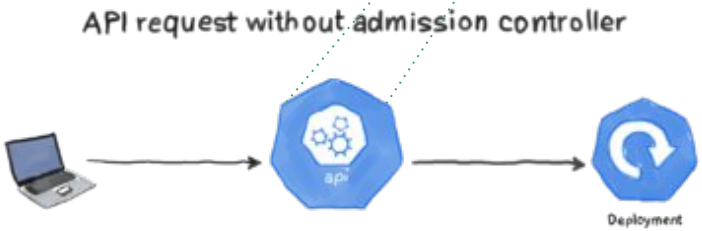

api-gw
Project: sysinfo-apps
Labels:
Status: Healthy Synced
Repository:
Target Revision: master
Path: overlays/api-gw
Destination: in-cluster
Namespace: api-gw
SYNC REFRESH DELETE
appman
Project: sysinfo-apps
Labels:
Status: Healthy Synced
Repository:
Target Revision: master
Path: overlays/appman
Destination: in-cluster
Namespace: appman
SYNC REFRESH DELETE
booking
Project: sysinfo-apps
Labels:
Status: Healthy Synced
Repository:
Target Revision: master
Path: overlays/booking
Destination: in-cluster
Namespace: booking
SYNC REFRESH DELETE
concorsi
Project: sysinfo-apps
Labels:
Status: Healthy Synced
Repository:
Target Revision: master
Path: overlays/concorsi
Destination: in-cluster
Namespace: concorsi
SYNC REFRESH DELETE
consuntivi
Project: sysinfo-apps
Labels:
Status: Healthy Synced
Repository:
Target Revision: master
Path: overlays/consuntivi
Destination: in-cluster
Namespace: consuntivi
SYNC REFRESH DELETE
fondipersonale
Project: sysinfo-apps
Labels:
Status: Healthy Synced
Repository:
Target Revision: master
Path: overlays/fondipersonale
Destination: in-cluster
Namespace: fondipersonale
SYNC REFRESH DELETE
identity
Project: sysinfo-apps
Labels:
Status: Healthy Synced
Repository:
Target Revision: master
Path: overlays/identity
Destination: in-cluster
Namespace: identity
SYNC REFRESH DELETE
mail
Project: sysinfo-apps
Labels:
Status: Healthy Synced
Repository:
Target Revision: master
Path: overlays/mail
Destination: in-cluster
Namespace: mail
SYNC REFRESH DELETE
preventivi
Project: sysinfo-apps
Labels:
Status: Healthy Synced
Repository:
Target Revision: master
Path: overlays/preventivi
Destination: in-cluster
Namespace: preventivi
SYNC REFRESH DELETE
progetti
Project: sysinfo-apps
Labels:
Status: Healthy Synced
Repository:
Target Revision: master
Path: overlays/progetti
Destination: in-cluster
Namespace: progetti
SYNC REFRESH DELETE
storage
Project: sysinfo-apps
Labels:
Status: Healthy Synced
Repository:
Target Revision: master
Path: overlays/storage
Destination: in-cluster
Namespace: storage
SYNC REFRESH DELETE
titolidistudio
Project: sysinfo-apps
Labels:
Status: Healthy Synced
Repository:
Target Revision: master
Path: overlays/titolidistudio
Destination: in-cluster
Namespace: titolidistudio
SYNC REFRESH DELETE

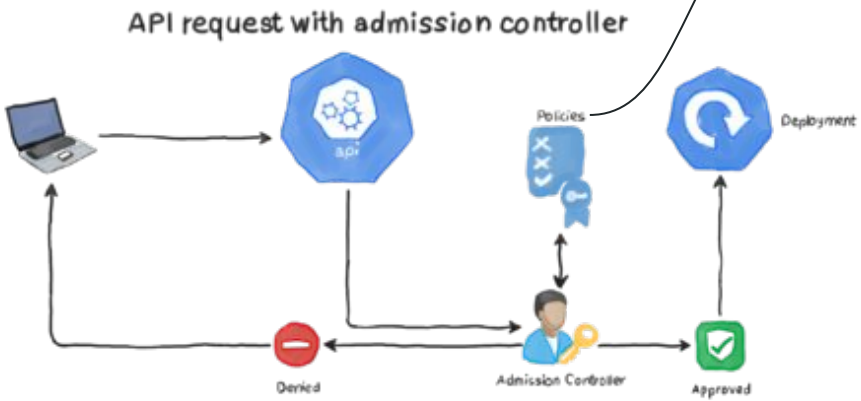
ArgoCD


MORE
To master (c2c4d88)
er <noreply@argoproj.io> -
matic update of consuntivi
LAST SYNC RESULT
Sync OK
Succeeded 7 days ago (Thu May 12 2022 17:10:38 GMT+0200)
Author: argocd-image-updater <noreply@argoproj.io> -
Comment: build: automatic update of consuntivi
MORE
To c2c4d88
consuntivi
2 months
consuntivi-backend
deploy
2 months rev:21
consuntivi-backend-57dc86cc...
pod
9 days running 2/2
consuntivi-frontend-78f948889...
pod
7 days running 1/1
consuntivi-frontend
deploy
2 months rev:27
egress-allow-api-sysinfo
ciliumnetworkpolicy
2 months
egress-allow-assegnazioni
ciliumnetworkpolicy
2 months
egress-allow-dns
ciliumnetworkpolicy
2 months
egress-allow-godiva
ciliumnetworkpolicy
2 months
egress-allow-keycloak
ciliumnetworkpolicy
2 months
egress-allow-mongodb
ciliumnetworkpolicy
2 months
egress-allow-prodotti
ciliumnetworkpolicy
a month
egress-allow-vault
ciliumnetworkpolicy
2 months
consuntivi-backend-metrics
servicemonitor
2 months
consuntivi-frontend
ing
2 months
consuntivi-tls
certificate
2 months
consuntivi-tls-ddjph
certificaterequest
2 months
consuntivi-tls-ddjph-897104096
order
2 months

OPA Gatekeeper

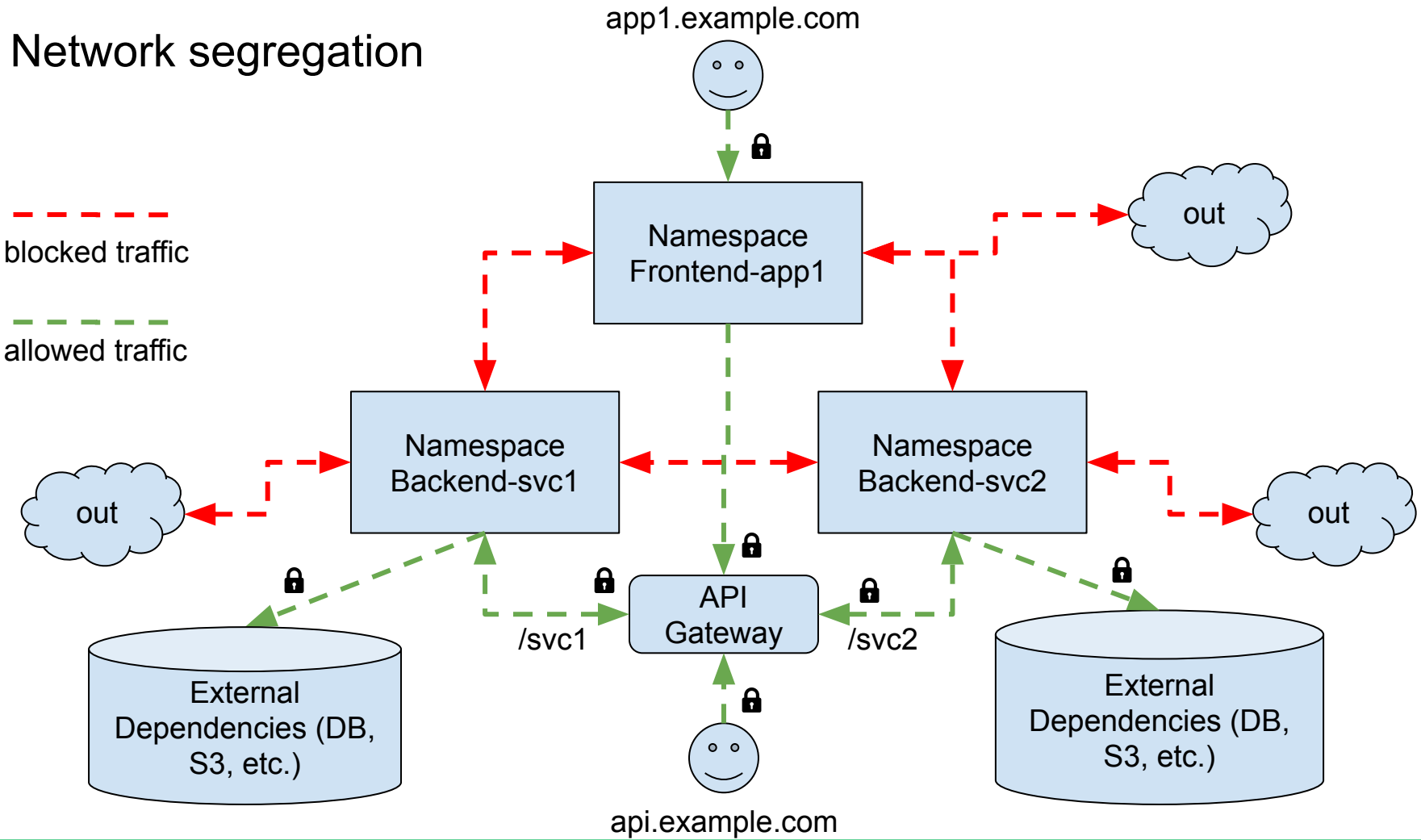






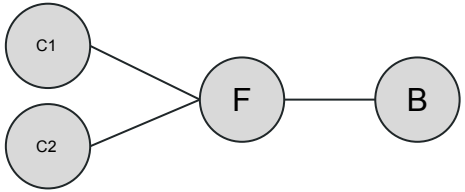

```
package k8spspallowprivilegeescalationcontainer

violation[{"msg": msg, "details": {}}] {
    c := input_containers[_]
    input_allow_privilege_escalation(c)
    msg := sprintf("Privilege escalation container is not allowed: %v", [c.name])
}

input_allow_privilege_escalation(c) {
    not has_field(c, "securityContext")
}
input_allow_privilege_escalation(c) {
    not c.securityContext.allowPrivilegeEscalation == false
}
input_containers[c] {
    c := input.review.object.spec.containers[_]
}
input_containers[c] {
    c := input.review.object.spec.initContainers[_]
}
# has_field returns whether an object has a field
has_field(object, field) = true {
    object[field]
}
```

Network segregation

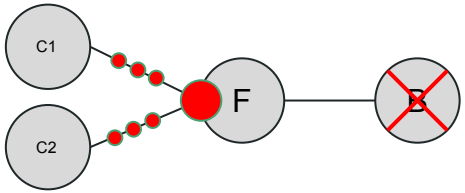


Circuit breaker: the problem

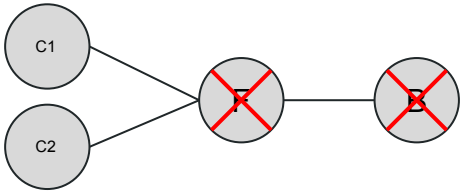
Local failures can propagate all over your architecture and destroy all your systems



- C makes a request to F
- F makes a request to B
- After short time Client got a response



- If B:
  - is down
  - is up but the network is unreachable (network partition)
  - is up but is very slow
- C makes a request to F
- F need to wait until the timeout before came back to C
- During the timeout period requests pile up on the F service

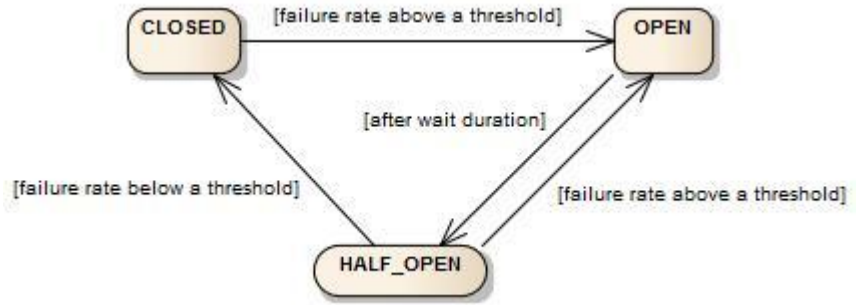


- F became unresponsive

Circuit breaker: the solution

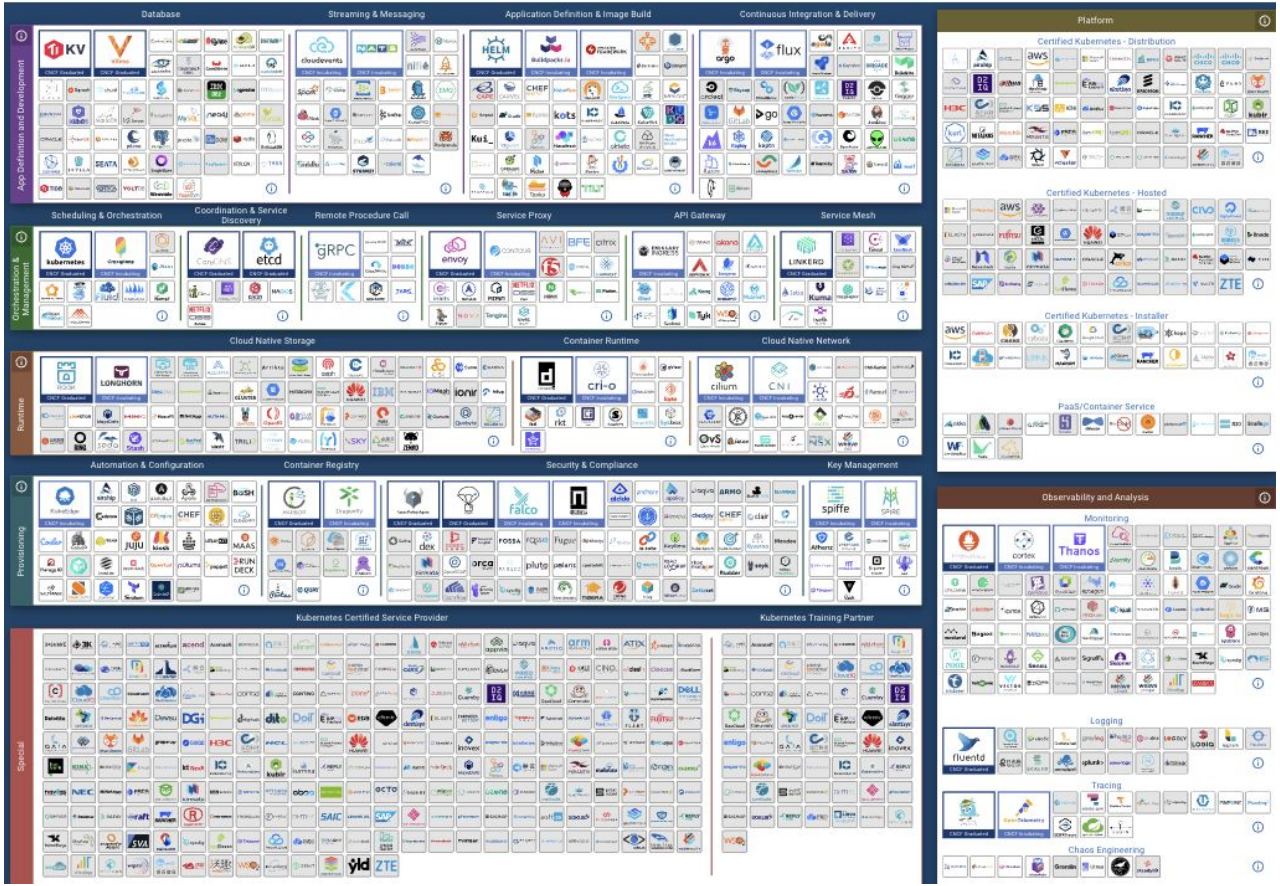
“A service client should invoke a remote service via a proxy that functions in a similar fashion to an electrical circuit breaker.”

- If a call fails, increment the number of failed calls by one
- If the number of failed calls goes above a certain threshold, open the circuit
- If the circuit is open, immediately return with an error **or a default response**
- If the circuit is open and some time has passed, half-open the circuit
- If the circuit is half-open and the next call fails, open it again
- If the circuit is half-open and the next call succeeds, close it



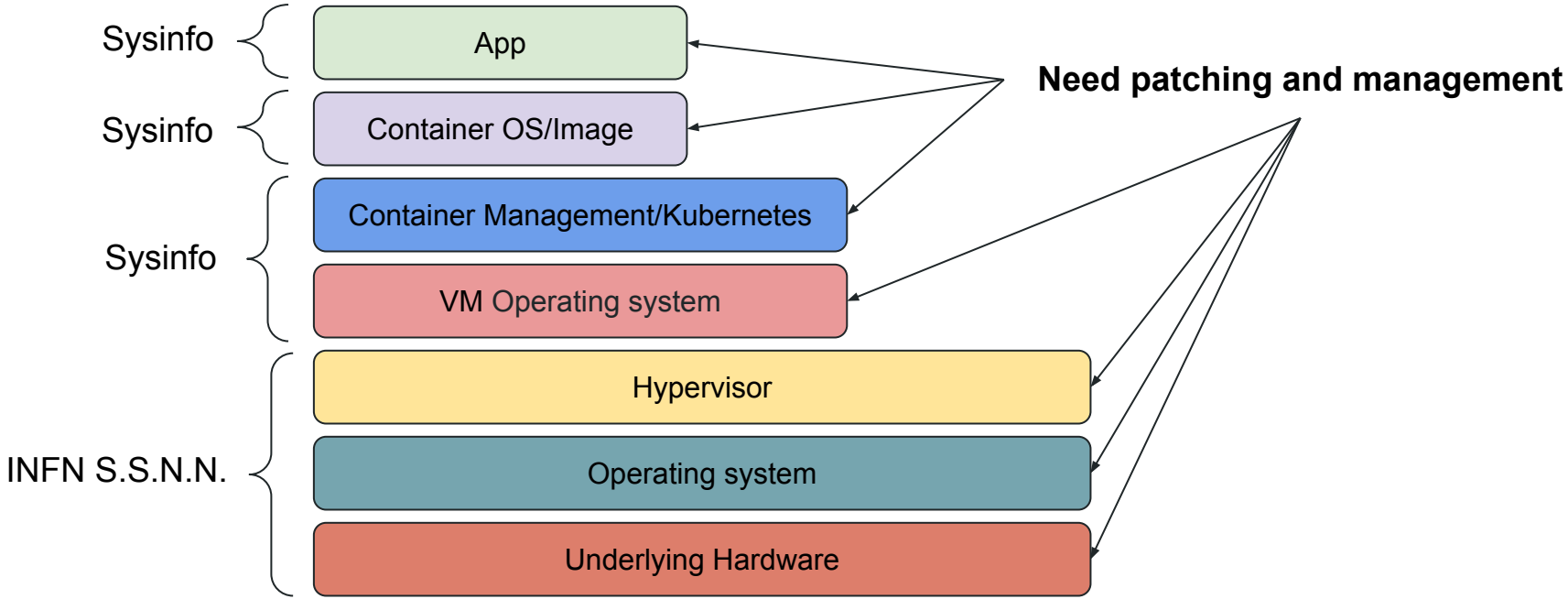
Strategy	Implementations	Fit
Black Box	● Proxies ● Service meshes	Fail fast
White Box	Libraries (e.g. Resilience4j)	Fallbacks relying on business logic

How do we secure this?




I'M SCANNING
MY CONTAINER IMAGES

Infrastructure patching&management



Infrastructure security management

- GitOps approach
- Static analysis, unit testing and deprecations checks
- Rego policy to enforce K8s best practices and security issues mitigation/prevention:
  - Continuous integration job ---> Trivy scanner
  - K8s API webhook ---> using OPA Gatekeeper
- Vulnerability/compliance scanning
- Minimal base images (updated at least every 24h)
- Image scan before push/after push/periodically (every day)
- Ossec + Falco
- Network Policy
- Log analisys&monitoring

Software security management

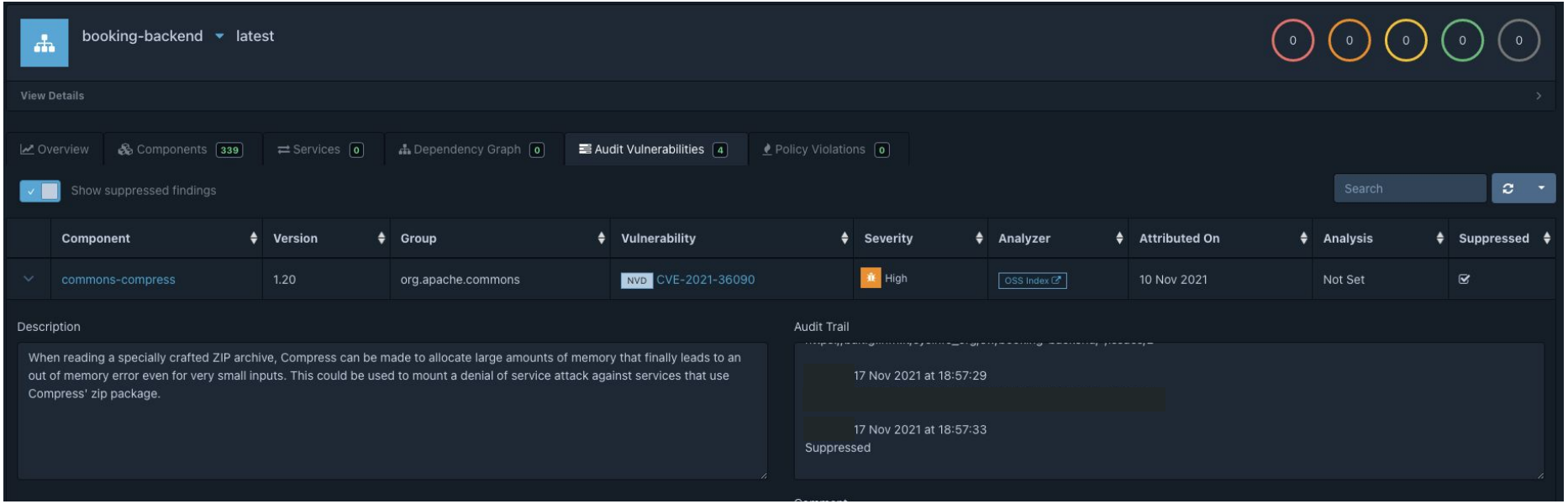
- Fully automated Continuous integration/delivery pipelines (target: SLSA level as higher as possible https://slsa.dev)
- Automated software security scanning and testing:
  - Decrease the risk of a security breach by automatically blocking known vulnerabilities
  - Critically malicious components and newly released suspicious components are automatically blocked
- Shift-left strategy:
  - Find and prevent defects early in the software delivery process
  - Dev team members are aware of the security constraints and best practices
  - Security by design (cross functional team)
- Software bills of materials (SBOMs):
  - Software “inventory”/ID
  - Generated automatically during the build phase
  - Release of problematic code automatically blocked
  - Enable continuous scanning of released software

Dependency track

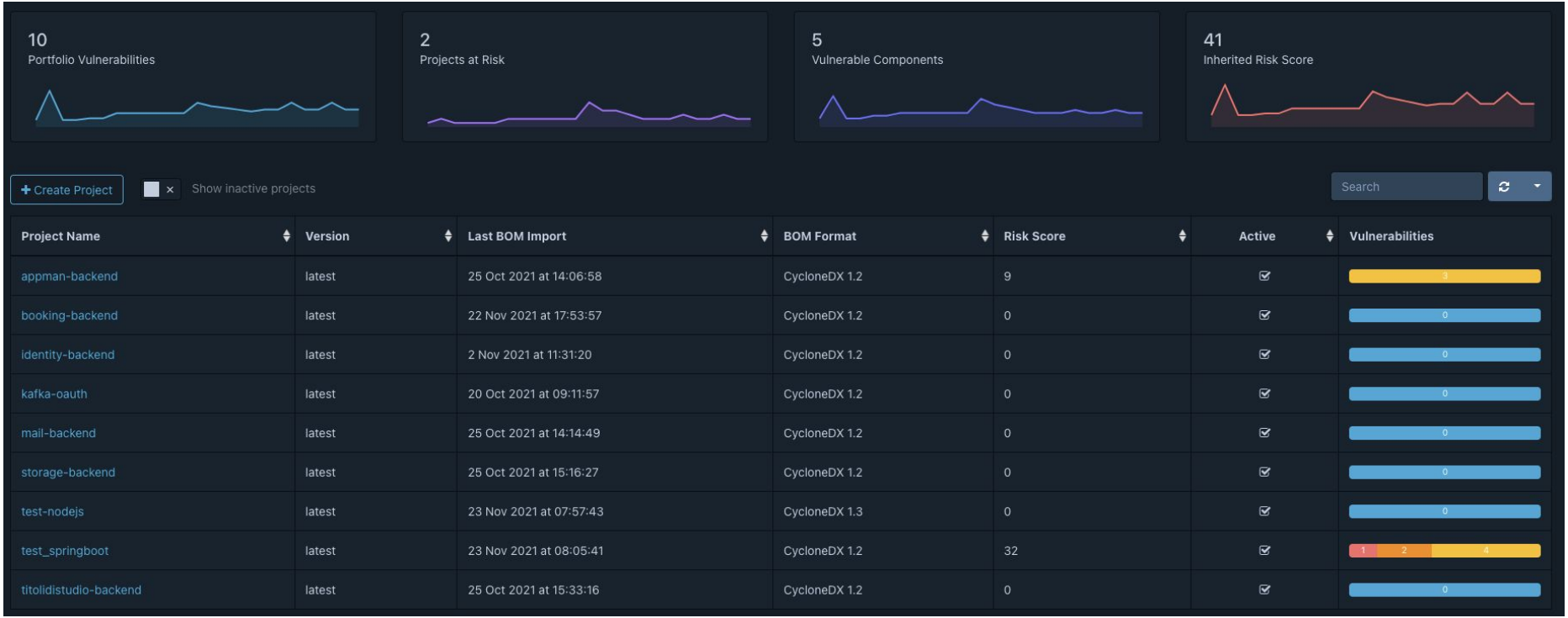
“An intelligent Component Analysis platform that allows organizations to identify and reduce risk in the software supply chain.”

- Software bills of materials (SBOMs) as source of truth
- Vulnerabilities/Licenses analysis/management
- Product risk analysis
- Measure and enforce policy compliance

Dependency track



Dependency track



Project Name	Version	Last BOM Import	BOM Format	Risk Score	Active	Vulnerabilities
appman-backend	latest	25 Oct 2021 at 14:06:58	CycloneDX 1.2	9	☑	3
booking-backend	latest	22 Nov 2021 at 17:53:57	CycloneDX 1.2	0	☑	0
identity-backend	latest	2 Nov 2021 at 11:31:20	CycloneDX 1.2	0	☑	0
kafka-oauth	latest	20 Oct 2021 at 09:11:57	CycloneDX 1.2	0	☑	0
mail-backend	latest	25 Oct 2021 at 14:14:49	CycloneDX 1.2	0	☑	0
storage-backend	latest	25 Oct 2021 at 15:16:27	CycloneDX 1.2	0	☑	0
test-nodejs	latest	23 Nov 2021 at 07:57:43	CycloneDX 1.3	0	☑	0
test_springboot	latest	23 Nov 2021 at 08:05:41	CycloneDX 1.2	32	☑	1 2 4
titolidistudio-backend	latest	25 Oct 2021 at 15:33:16	CycloneDX 1.2	0	☑	0

"New software" highlights

1. Better security aspects management
2. Improved automated software testing
3. Design for failure
4. Improved/simplified development and release workflow (no more long lived branches)
5. Limited amount of languages/frameworks
6. Database: MongoDB and PostgreSQL
7. File storage: no more NFS or local filesystem
8. Restricted database read-write access:
    - No more “real data” access required for software development
    - No more “physiological” data correction
    - No more manual schema migration
9. No more shared databases/schema
10. No more database raw data direct access

Any organization that design a system will inevitably produce a design whose structure is a copy of the organization's communication structure

(Conoway's law, 1967)