

kublr

Centralizing Kubernetes and Container Operations

Oleg Chunikhin

CTO, Kublr

Introductions



Oleg Chunikhin
CTO
Kublr

- Nearly 20 years in the field of software architecture and development.
- Joined Kublr as the CTO in 2016.
- Kublr is an enterprise Kubernetes management and operations platform that helps accelerate Kubernetes adoption and containerized applications management for enterprises.



History

- Custom software development company
- Dozens of projects per year
- Varying target environments: clouds, on-prem, hybrid
- Unified application delivery and ops platform wanted: monitoring, logs, security, multiple env, ...





Docker and Kubernetes to the Rescue

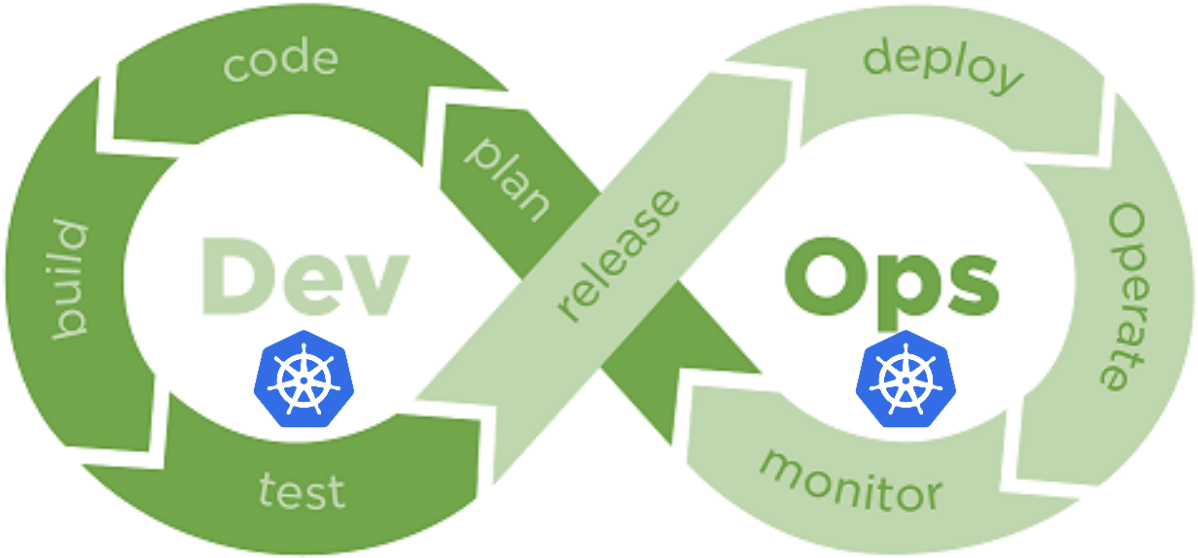
- Docker is great, but local
- Kubernetes is great... when it is up and running
- Who sets up and operates K8S?
- Who takes care of operational aspects at scale?
- How to provide governance and ensure compliance ?



Enterprise Kubernetes Needs

Developers

- Self-service
- Compatible
- Conformant
- Configurable
- Open & Flexible



- Security
- Reliability
- Performance
- Portability

SRE/Ops/DevOps/SecOps

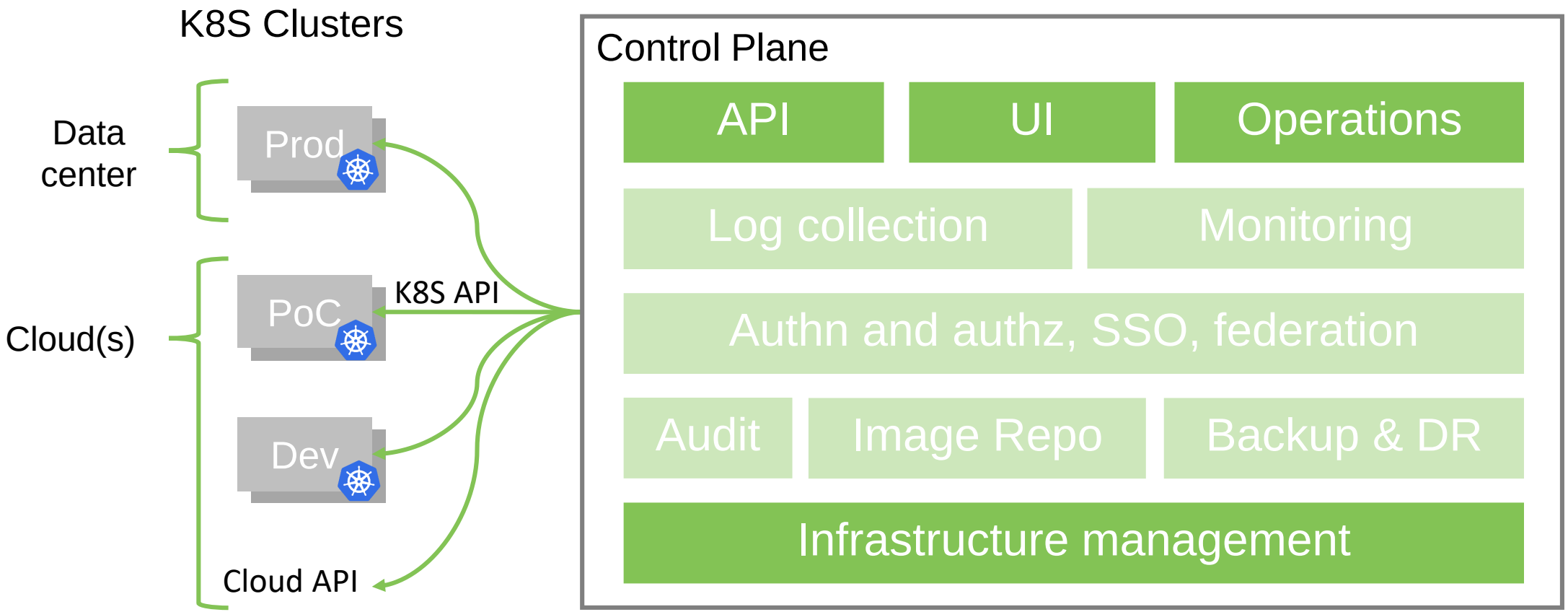
- Org multi-tenancy
- Single pane of glass
- Operations
- Monitoring
- Log collection
- Image management
- Identity management



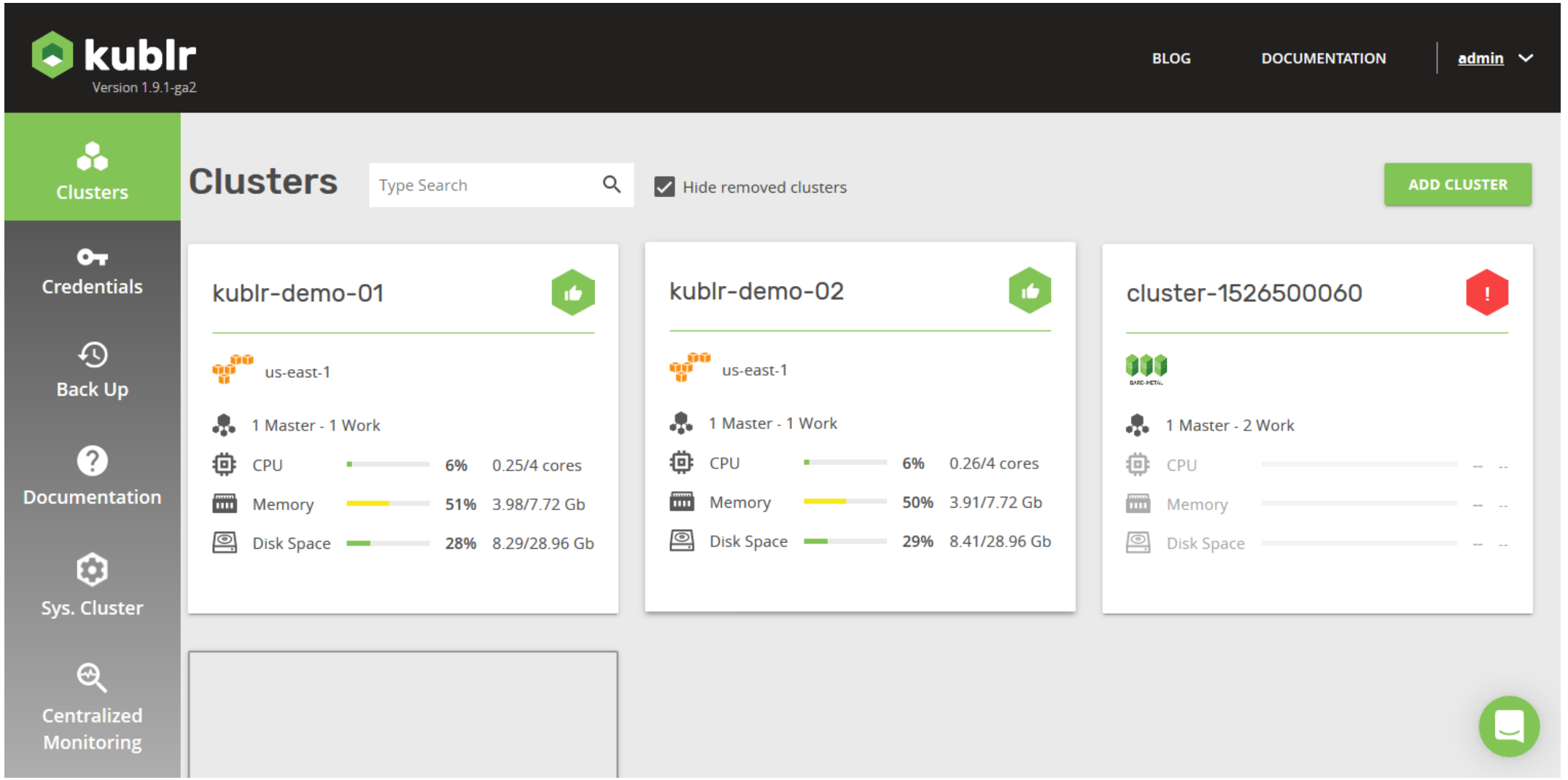
Kubernetes Management Platform Wanted

- Portability – clouds, on-prem, air-gapped, different OS's
- Centralized multi-cluster operations saves resources – many environments (dev, prod, QA, ...), teams, applications
- Self-service and governance for Kubernetes operations
- Reliability – cluster self-healing, self-reliance
- Limited management profile – cloud and K8S API
- Architecture – flexible, open, pluggable, compatible
- Sturdy – secure, scalable, modular, HA, DR etc.

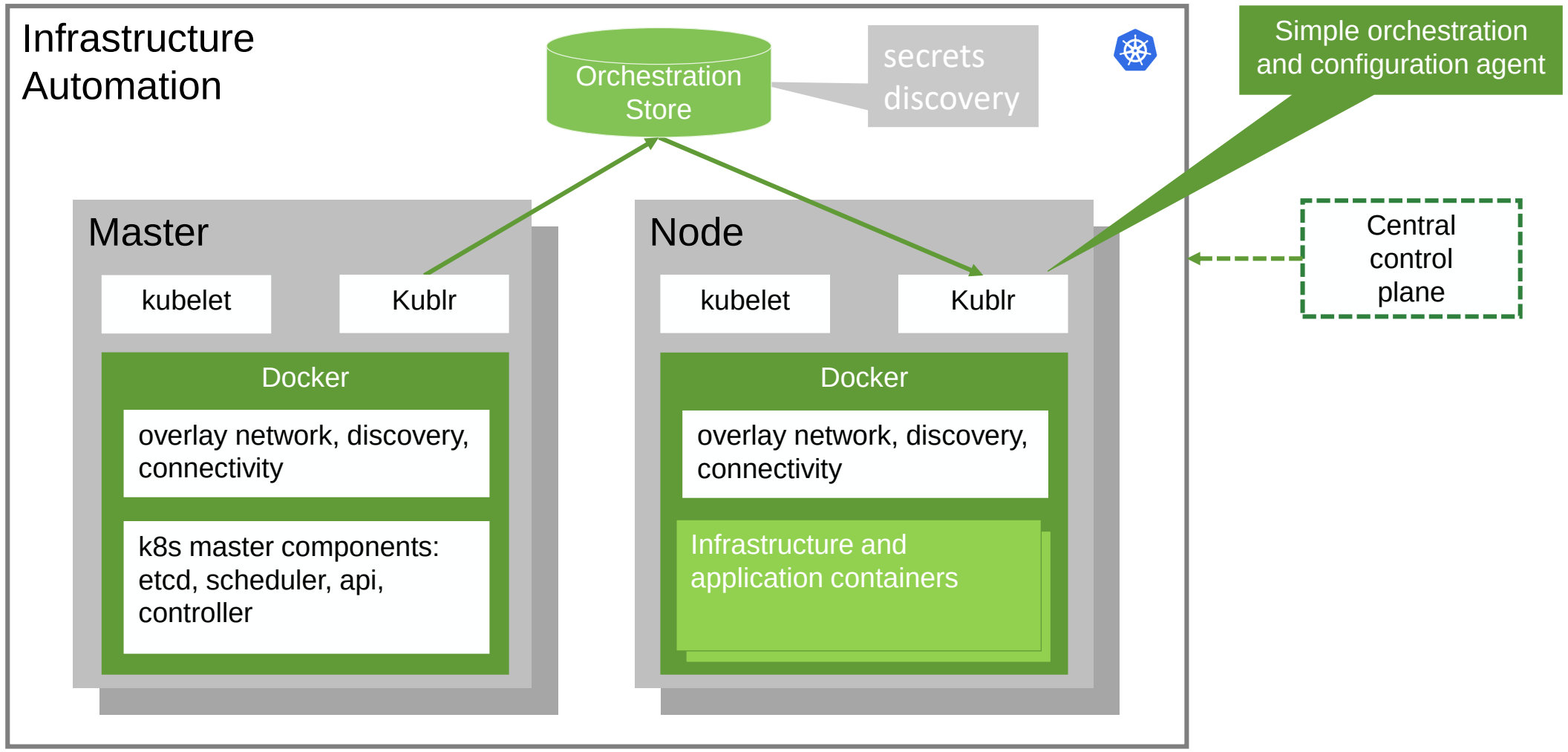
Central Control Plane: Operations



Central Control Plane: Operations

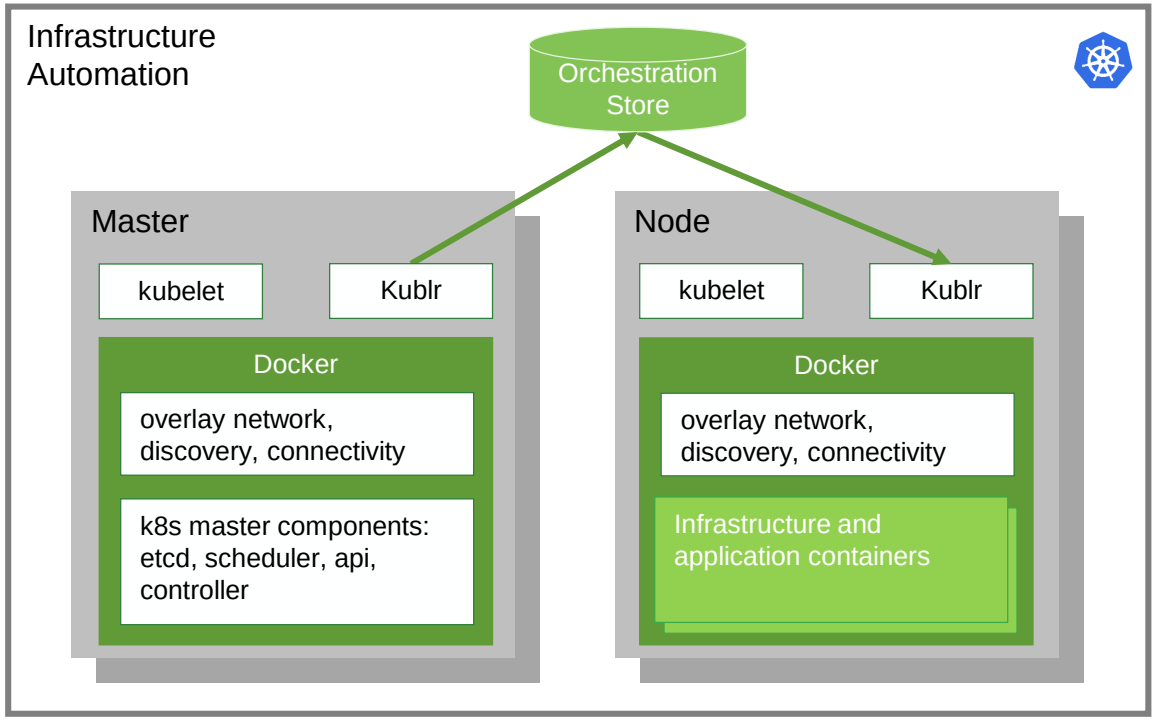


Cluster: Self-Sufficiency



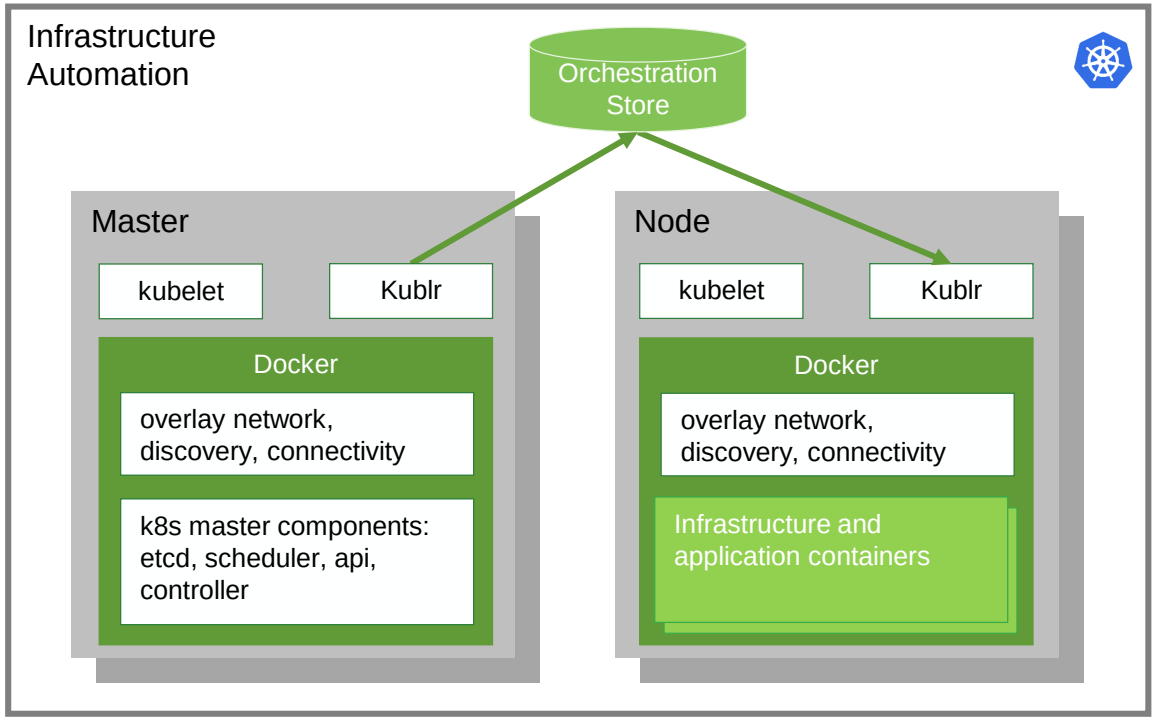
Cluster: Portability

- Everything (almost) runs in containers
- Simple (single-binary) management agent
- Minimal store requirements
  - Shared, eventually consistent
  - Secure: RW files for masters, RO for nodes
  - Thus the store can be anything:
    S3, SA, NFS, rsynced dir, provided files, ...
- Minimal infra automation requirements
  - Configure and run configuration agent
  - Enable access to the store
  - Can be AWS CF, Azure ARM, BOSH, Ansible, ...
- Load balancer is not required for multi-master
  Agents can switch to a healthy master

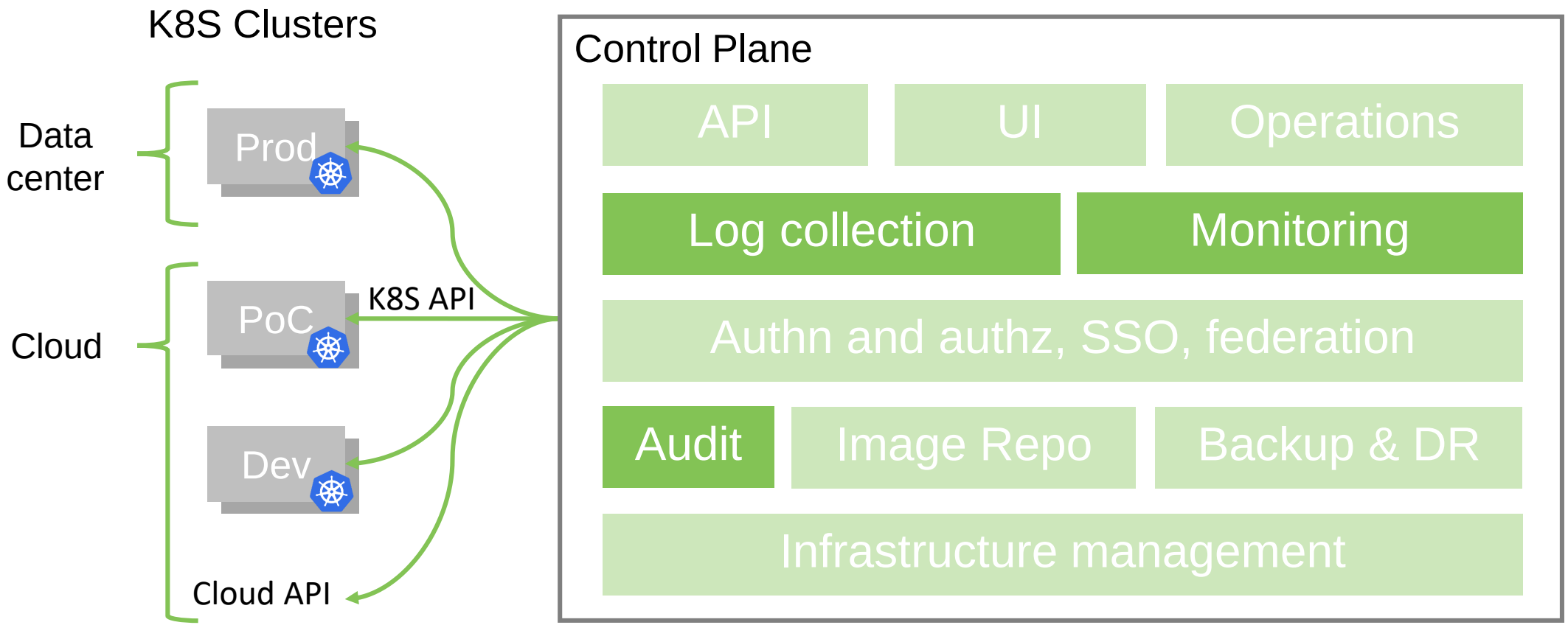


Cluster: Reliability

- Rely on underlying platform as much as possible
  - ASG on AWS
  - IAM on AWS for store access
  - SA on Azure, S3 on AWS
  - ARM on Azure, CF on AWS
- Minimal infrastructure SLA
  tolerate temporary failures
- Multi-muster API failover on nodes
- Resource management, memory requests and limits for OS and k8s components



Central Control Plane: Logs and Metrics





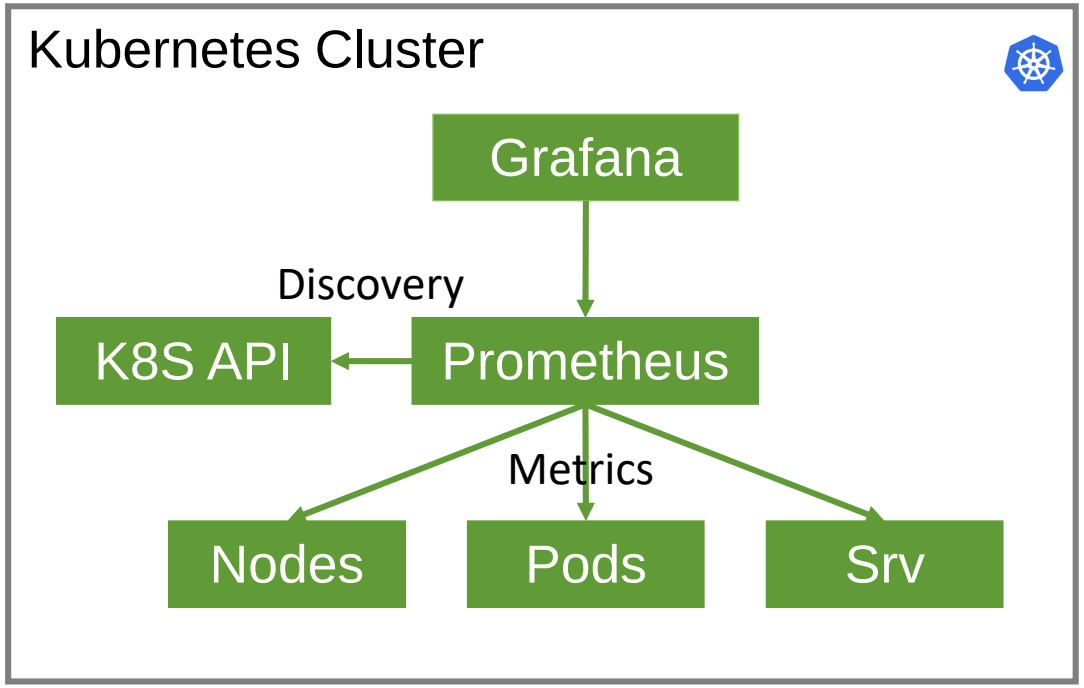
Centralized Monitoring and Log Collection. Why Bother?

- Prometheus and ELK are heavy and not easy to operate
  Need attention and at least 4-8 Gb RAM... each, per cluster
- Cloud/SaaS monitoring is not always permitted or available
- Existing monitoring is often not container-aware
- No aggregated view and analysis
- No alerting governance

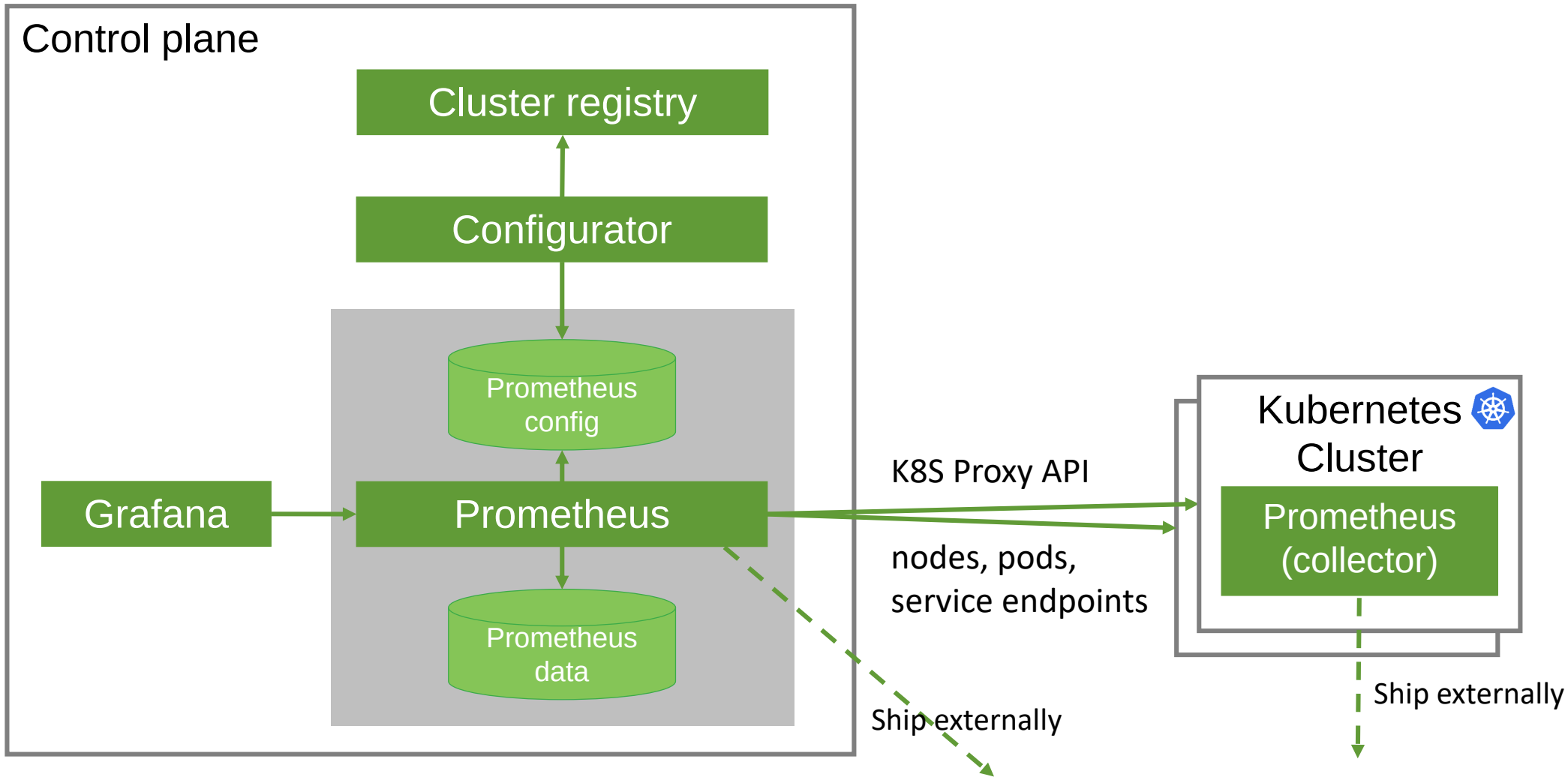


K8S Monitoring with Prometheus

- Discover nodes, services, pods via K8S API
- Query metrics from discovered endpoints
- Endpoint are accessed directly via internal cluster addresses



Centralized Monitoring

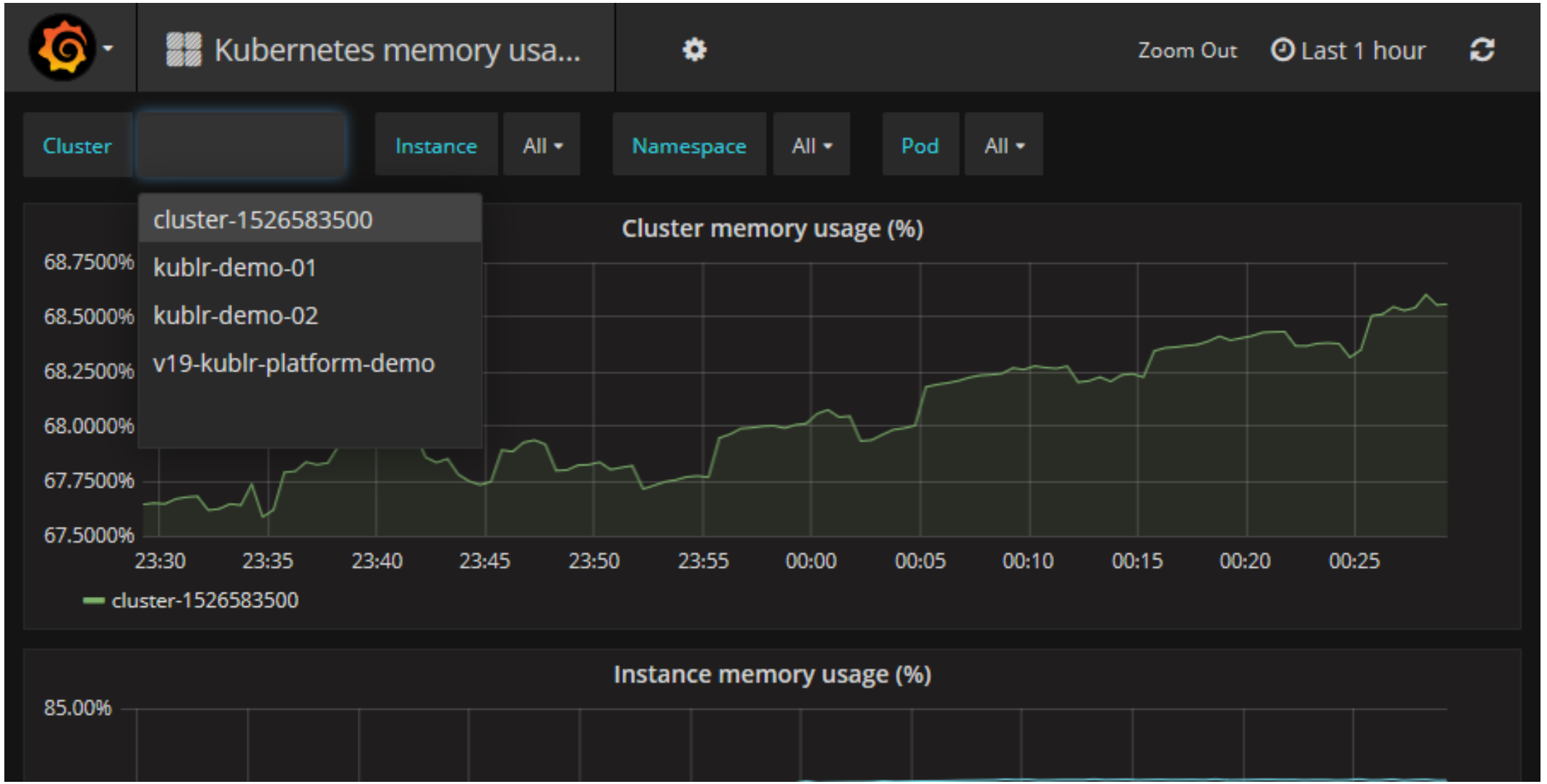




Centralized Monitoring: Considerations

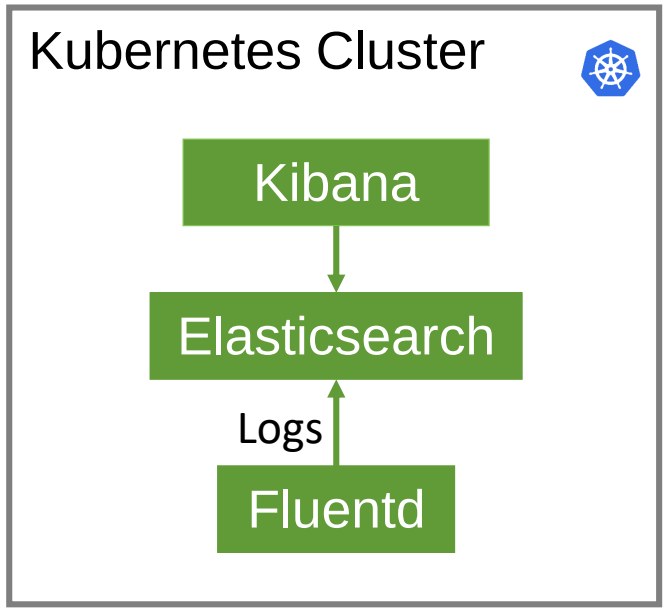
- Prometheus resource usage tuning
- Long-term storage (m3)
- Configuration file growth with many clusters
- Metrics labeling
- Additional load on API server

Centralized Monitoring

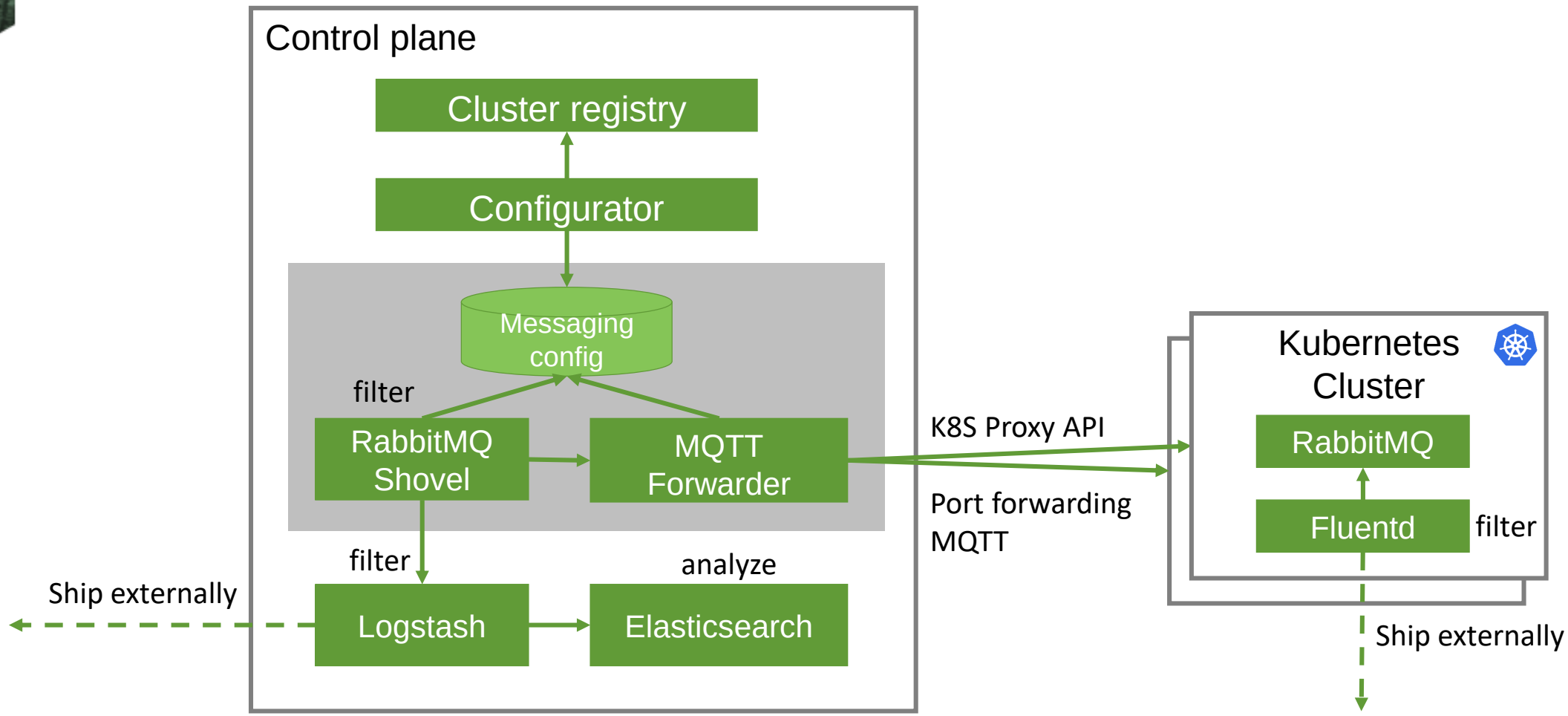


K8S Logging with Elasticsearch

- Fluentd runs on nodes
- OS, K8s, and container logs collected and shipped to Elasticsearch
- Kibana for visualization



Centralized Log Collection





Centralized Log Collection: Considerations

- Tune Elasticsearch resource usage
- Take into account additional load on API server
- Log index structure normalization

```
{
  "data": {
    "elasticsearch": {
      "version": "6.x"
    }
  }
}
```

```
{
  "flatData": [
    {
      "key": "elasticsearch.version",
      "type": "string",
      "key_type": "elasticsearch.version.string",
      "value_string": "6.x"
    },
    ...
  ]
}
```



The Rest: Considerations

- **Identity management**
  Use Identity Broker (e.g. KeyCloak): Users, Authn, Autzn, SSO, RBAC, Federation, ...
- **Backup and disaster recovery**
  K8s metadata + app data/volumes: full cluster recovery or copy
  Consistency between different
- **Docker image management**
  Docker image registry (e.g. Nexus, Artifactory, Docker Hub)
  Image scanning
  Air-gapped or isolated environment: image registries proxying and caching, “system” images





Q&A





kublr

Thank you!

Oleg Chunikhin

Chief Technology Officer

oleg@kublr.com

@olgch

Kublr | kublr.com

@kublr