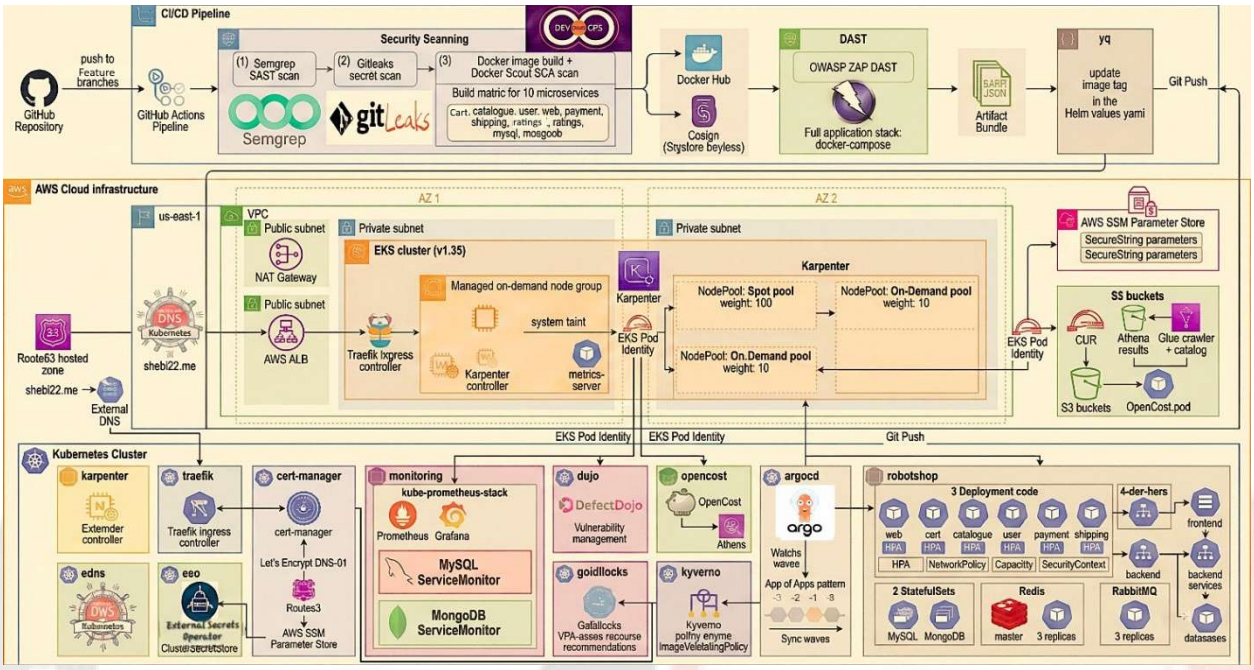
Robot-shop Production-Grade DevSecOps, GitOps & FinOps Platform on AWS Cloud-Native Kubernetes Architecture

By Eng: Abdelrahman Mahmoud

Abdelrahman Ayman

Contents

Warning In this Documentation shows how to reach to this diagram with out hacking:



LET'S GOOOOO!

Robot Shop — Production DevSecOps Platform

Enterprise-Grade Cloud-Native Reference Architecture on AWS

Introduction

Robot Shop is a 12-microservice e-commerce application transformed into a **production-grade DevSecOps platform** running on AWS.

This project is completely demonstrating:

- Infrastructure as Code (Terraform)
- GitOps (ArgoCD)
- Secure CI/CD pipelines
- Kubernetes production hardening
- Supply-chain security
- FinOps cost visibility
- Observability & policy enforcement
- Zero secret exposure in Git

This project serves as a **reference implementation of a modern cloud-native DevSecOps architecture**.

Application Overview

Robot Shop is a polyglot microservices application composed of 12 services.

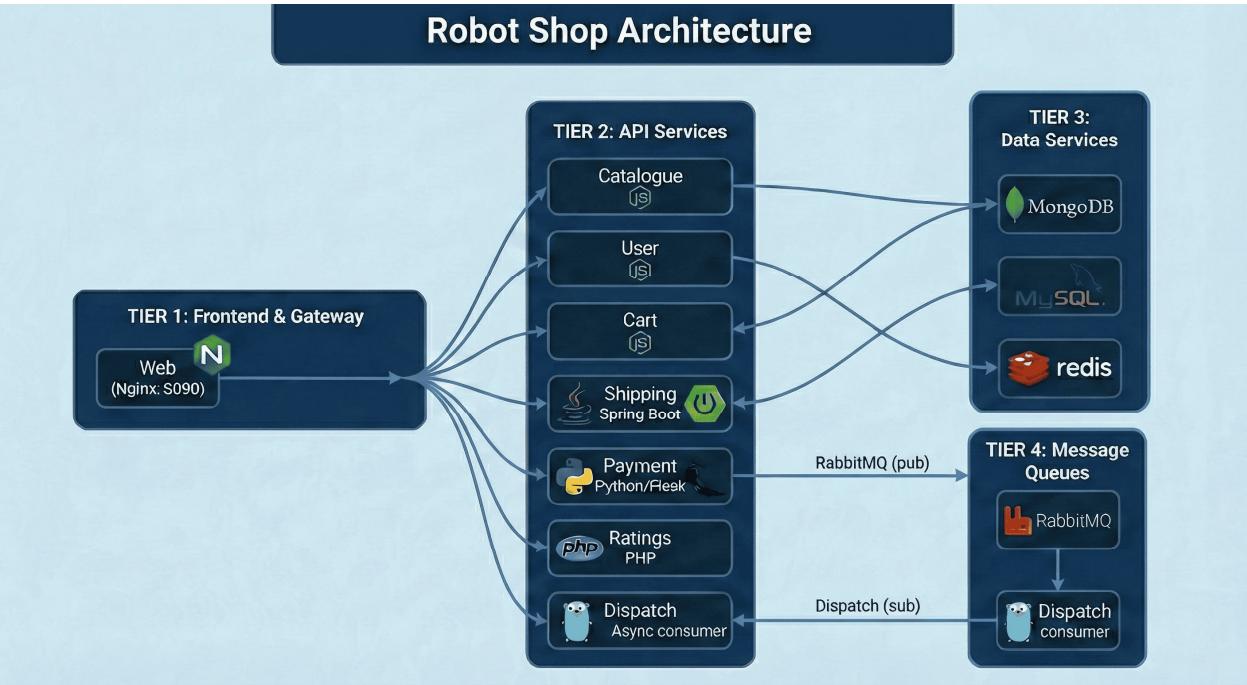
Technology Stack

Service	Technology	Responsibility
web	Nginx	Reverse proxy & frontend gateway
catalogue	Node.js (Express)	Product catalog (MongoDB)
user	Node.js (Express)	User authentication (MongoDB + Redis)
cart	Node.js (Express)	Shopping cart (Redis)
shipping	Java (Spring Boot)	Shipping calculation (MySQL)
payment	Python (Flask)	Payment processing (RabbitMQ)
ratings	PHP (Apache)	Product ratings (MySQL)
dispatch	Go	Async order processing
mongodb	MongoDB	Document database
mysql	MySQL	Relational database
redis	Redis	Session cache
rabbitmq	RabbitMQ	Message broker

Communication Model

- REST-based synchronous communication
- Asynchronous messaging via RabbitMQ
- Internal service isolation via Kubernetes Network Policies

The structure of Robot shop



Docker Layer Local Development Environment

Purpose

The Docker layer provides a fast and isolated local development environment for engineers.

It enables:

- Rapid iteration
- Service debugging
- Security testing (DAST stage)
- Environment parity before Kubernetes deployment

Architecture

All 12 microservices and dependencies run via:

- Docker Compose
- Internal Docker network
- Environment variables from .env

Services include:

Service	Technology	Role
Web	Nginx	Reverse proxy & frontend
Catalogue	Node.js	Product API
User	Node.js	Authentication
Cart	Node.js	Cart service
Shipping	Java (Spring Boot)	Shipping calculation
Payment	Python (Flask)	Payment processing
Ratings	PHP	Ratings service
Dispatch	Go	Async consumer
MongoDB	MongoDB	Document DB
MySQL	MySQL	Relational DB
Redis	Redis	Cache & sessions
RabbitMQ	RabbitMQ	Message broker

Enhancement in Docker files

- Using light weight base images
- Using multi-stage in some micro services
- Making docker files for all micro services except (Redis, RabbitMQ)

Also we made Docker compose file for the project

Kubernetes Layer Production Workload Architecture

Purpose

Kubernetes provides orchestration, scalability, security boundaries, and resilience.

The cluster runs on **Amazon Elastic Kubernetes Service**.

Core Design Principles

1. Security-First Pods

2. Horizontal Pod Autoscaling

Each application service includes:

- CPU-based scaling
- Memory-based scaling
- Environment-specific replica tuning

3. Network Policies (4-Tier Model)

Tier	Allowed Ingress	Allowed Egress
Frontend	Ingress Controller	Backend + DNS
Backend	Frontend	DB + Queue + DNS
Database	Backend only	DNS only
Queue	Backend only	DNS only

This ensures strict east-west traffic control.

4. Zero Git Secrets

No credentials exist in Git.

Secrets flow:

Terraform → AWS SSM → ESO → Kubernetes Secrets → Pods

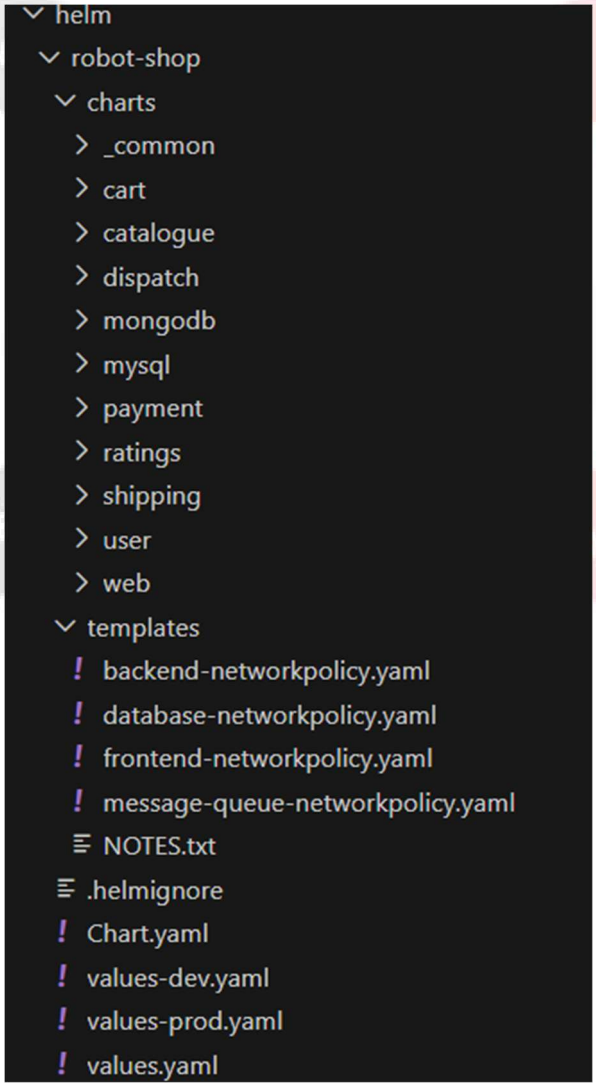
Umbrella Helm Layer Deployment Abstraction

Purpose

The umbrella Helm chart provides:

- Centralized deployment logic
- Environment-specific overrides
- Reusable security templates
- DRY chart architecture

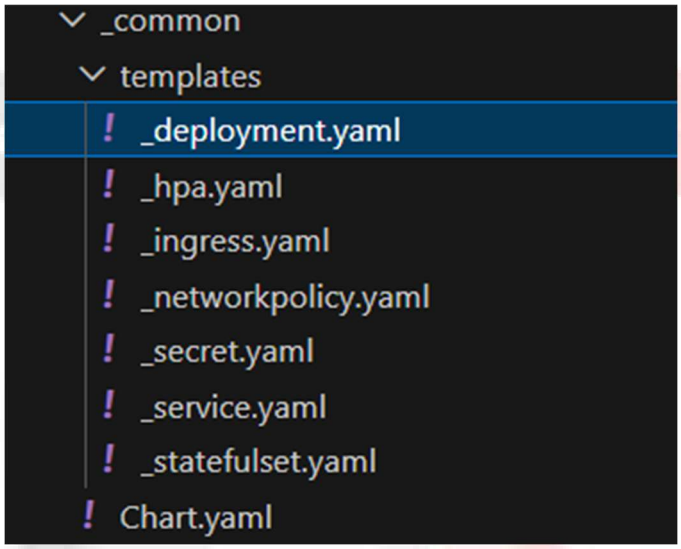
Structure



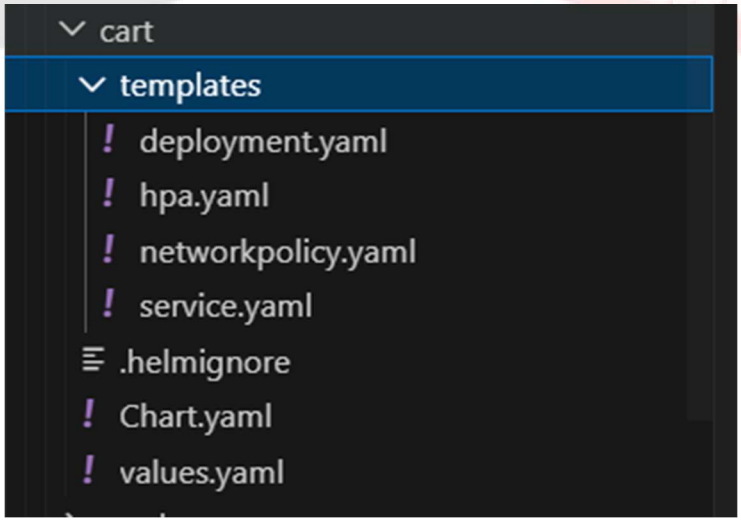
Shared Library Chart

The _common chart enforces:

- Security contexts
- HPA templates
- NetworkPolicy templates
- Standardized labels
- Resource limits



Using these templates in all microservices like:



Put values of cart svc and in the manifests define the common template.

Dev vs Production

Setting	Dev	Production
Replicas	1	2–6
Resources	Minimal	Tuned
HPA	Optional	Required
TLS	Optional	Enforced

Charts explanation and their uses:

Karpenter Chart

Overview

Karpenter is a next-generation Kubernetes node provisioning system that dynamically launches compute capacity based on pending pods.

It replaces traditional Cluster Autoscaler with faster, more intelligent scaling.

Purpose in This Platform

- ❖ Dynamic node provisioning
- ❖ Spot-first cost optimization
- ❖ Instance type auto-selection
- ❖ Workload-aware scaling

Architecture

Components deployed:

- Karpenter Controller (Deployment)
- ServiceAccount (Pod Identity attached)

- CRDs:
  - NodePool
  - EC2NodeClass
  - NodeClaim

Scaling Behavior

1. Pod becomes unschedulable
2. Karpenter evaluates resource requirements
3. Selects cheapest matching instance
4. Launches EC2 instance
5. Joins EKS cluster automatically

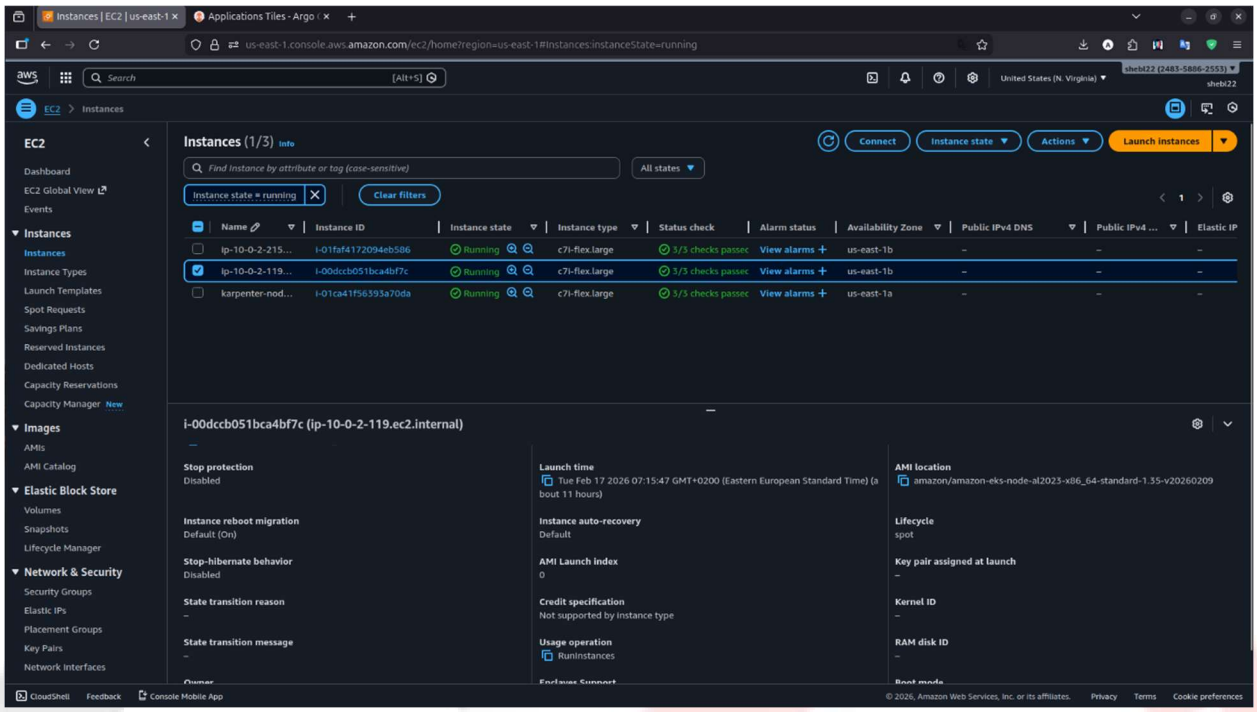
Security

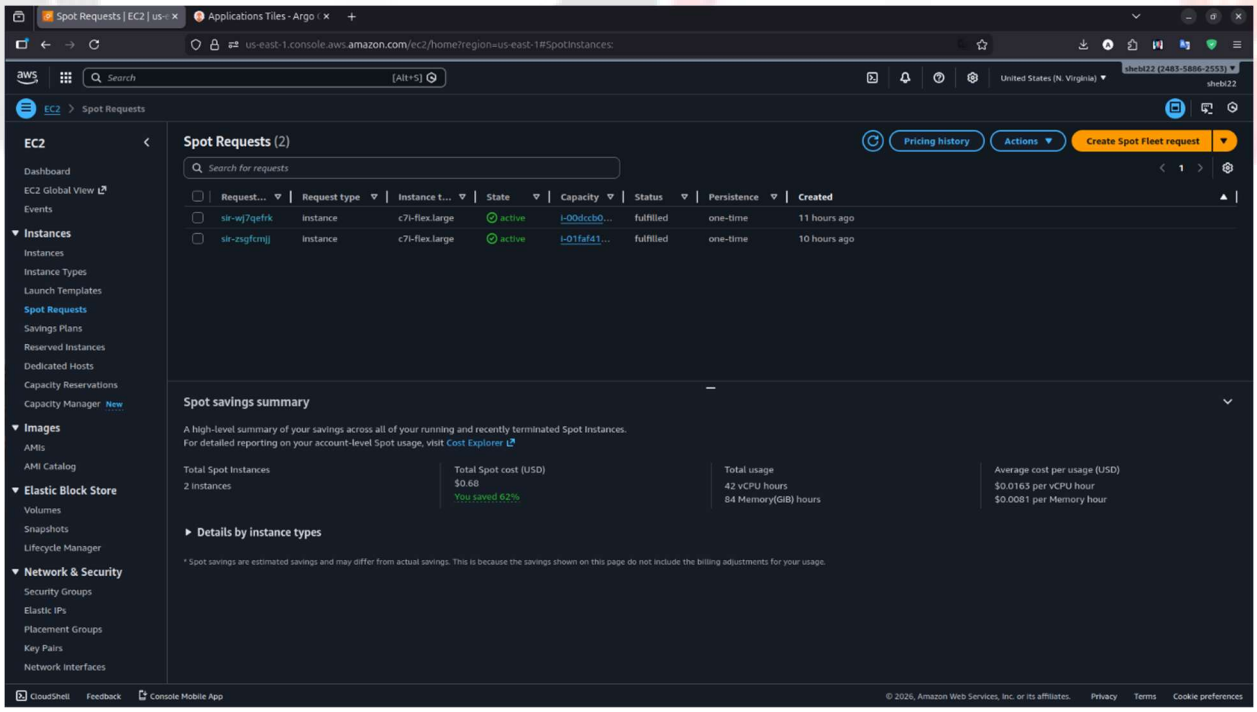
- Uses EKS Pod Identity (no IRSA)
- Least-privilege IAM role
- Runs in dedicated namespace
- Tolerations for system nodes

Cost Optimization Strategy

- Spot instances preferred
- On-demand fallback
- Consolidation enabled
- Unused nodes automatically terminated

Results of this chart:





DefectDojo Chart

Overview

DefectDojo is a centralized vulnerability management platform used to aggregate and track security findings.

Purpose in This Platform

- Central security dashboard
- Aggregates:
  - SAST (Semgrep)
  - Secret scanning (Gitleaks)
  - DAST (OWASP ZAP)
  - Container scanning (Docker Scout)
- Tracks remediation lifecycle

Architecture

Deployed components:

- Web application
- Worker
- Redis
- PostgreSQL

CI Integration

Pipeline uploads scan results automatically via API:

- SARIF
- JSON reports
- ZAP reports

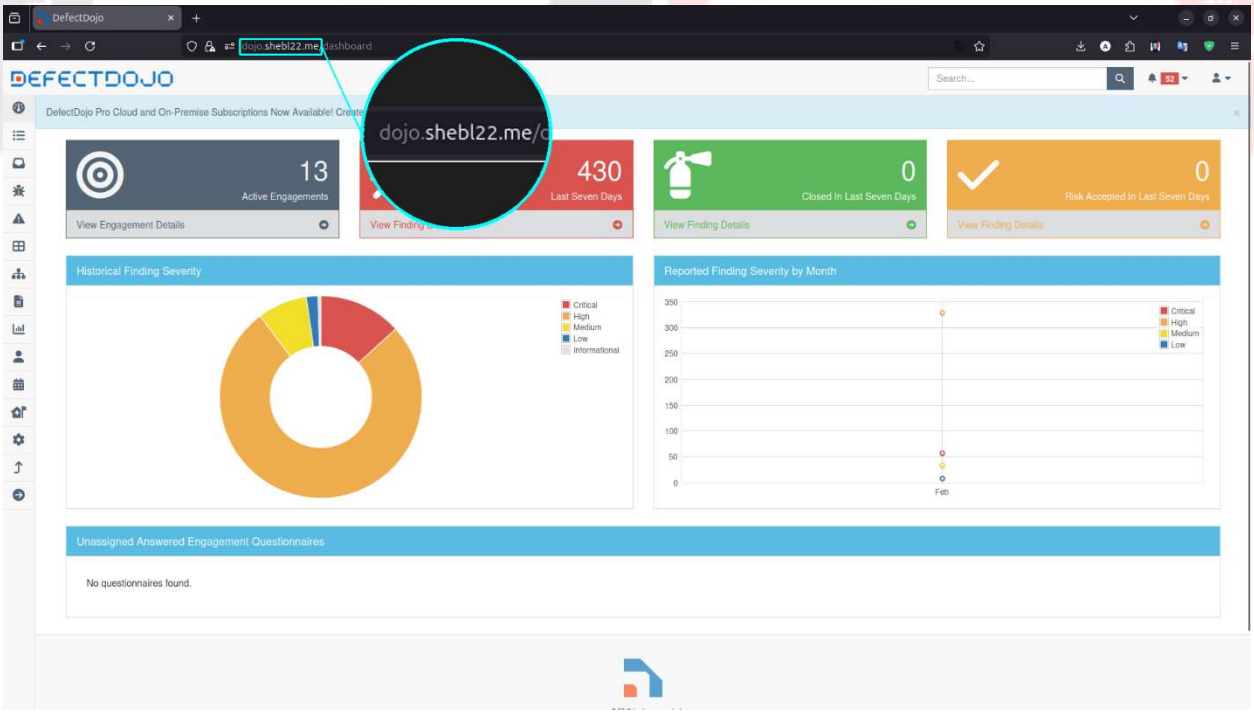
Security

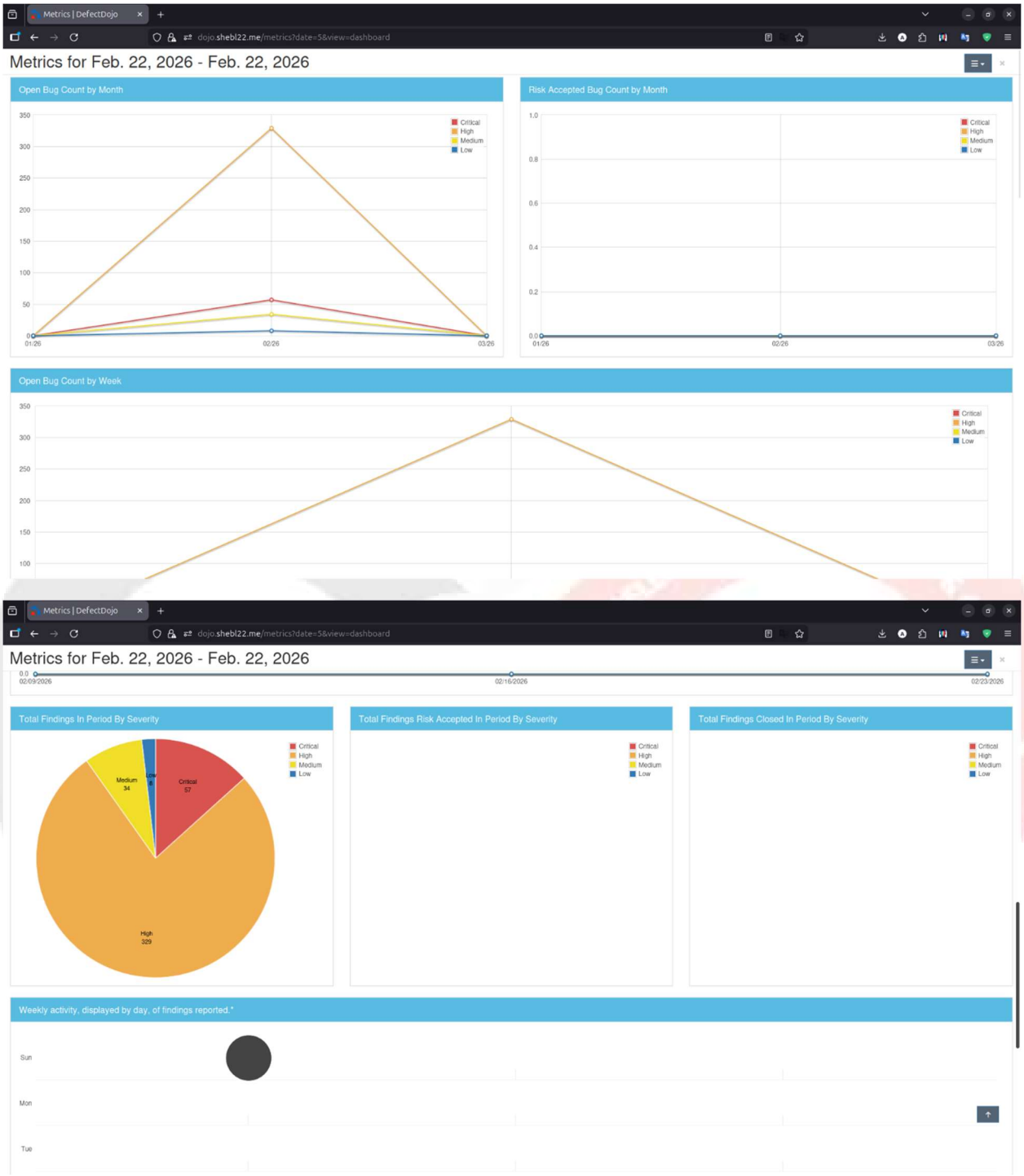
- Runs in isolated namespace
- Access restricted via Ingress + TLS
- Admin credentials stored in AWS SSM

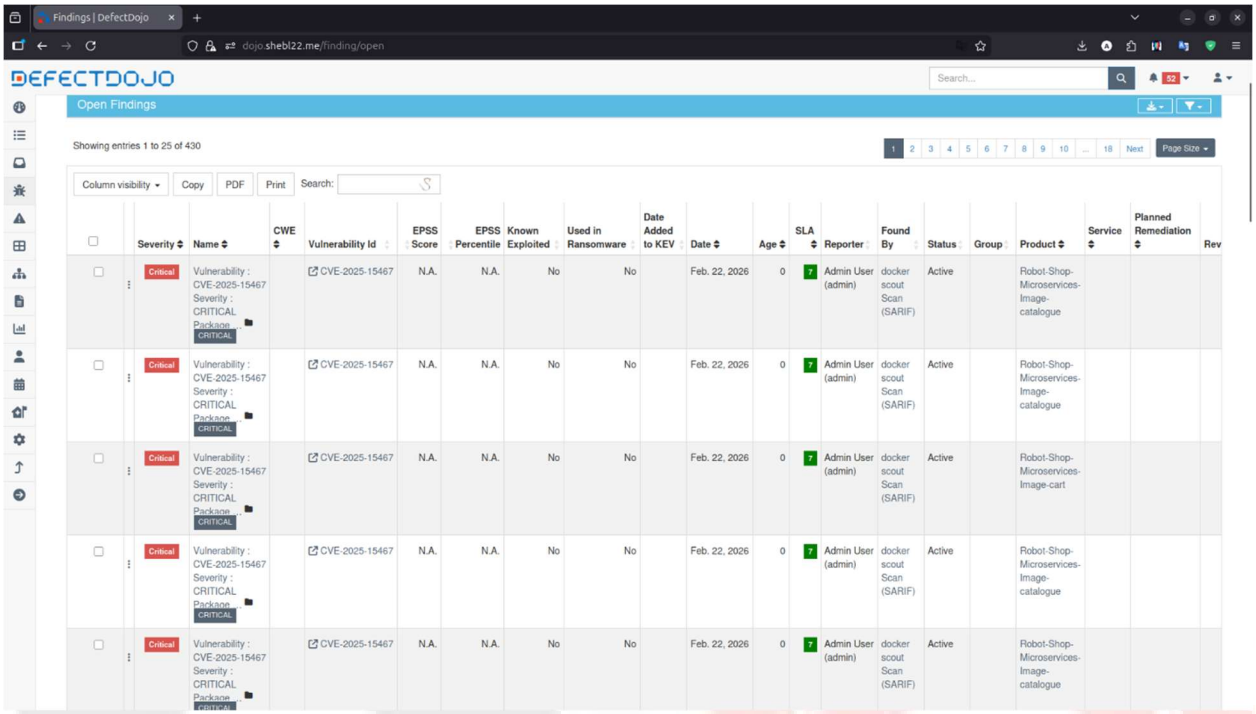
Benefits

- Centralized vulnerability visibility
- SLA tracking
- Risk prioritization
- Audit readiness

Results:




Metrics | DefectDojo
dojo.shebl22.me/metrics?date=5&view=dashboard
Metrics for Feb. 22, 2026 - Feb. 22, 2026
Open Bug Count by Month
Critical
High
Medium
Low
01/26
02/26
03/26
Risk Accepted Bug Count by Month
Open Bug Count by Week
02/09/2026
02/16/2026
02/23/2026
Total Findings In Period By Severity
Critical 57
High 329
Medium 34
Low 8
Total Findings Risk Accepted In Period By Severity
Total Findings Closed In Period By Severity
Weekly activity, displayed by day, of findings reported.*
Sun
Mon
Tue



Kyverno Chart

Overview

Kyverno is a Kubernetes-native policy engine used to enforce security and compliance rules.

Purpose in This Platform

- Enforce signed container images
- Prevent privileged containers
- Block hostPath volumes
- Enforce security context standards

Implemented Policies

1. Signed Image Verification

- Uses Cosign
- Rejects unsigned images

2. Pod Security Enforcement

- runAsNonRoot required
- Drop ALL capabilities
- Disallow privilege escalation

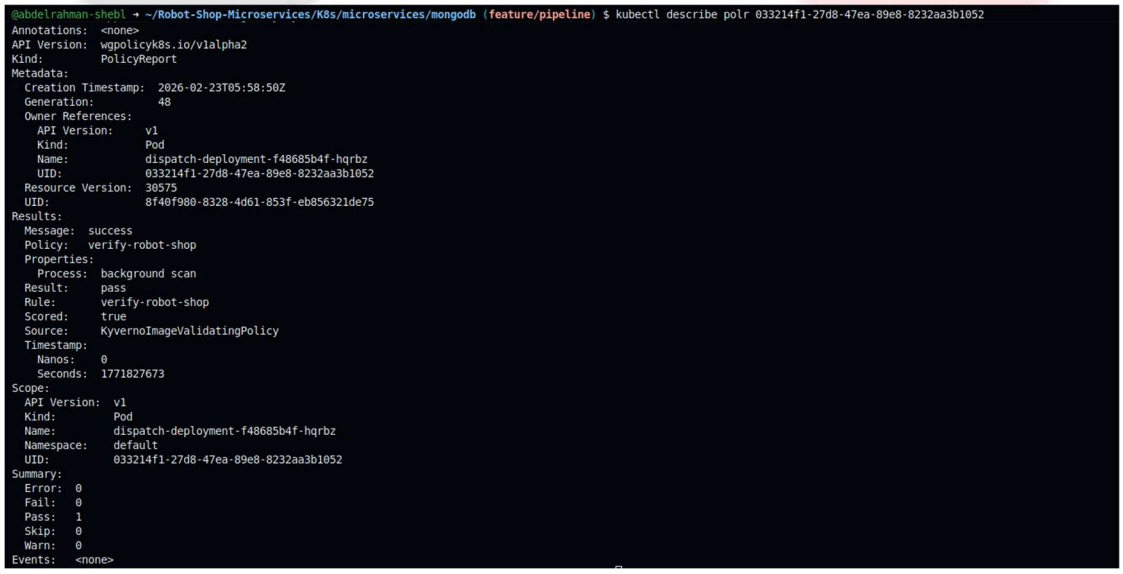
3. Resource Limits Enforcement

- All pods must define CPU and memory limits

Why Kyverno Instead of OPA?

- Native Kubernetes CRDs
- YAML-based policies
- Easier maintainability

Results:



```
@abdelrahman-shebl → ~/Robot-Shop-Microservices/K8s/microservices/mongodb (feature/pipeline) $ kubectl get pods
NAME                                READY   STATUS                  RESTARTS      AGE
cart-deployment-6bd6749cd4-qk5fj    1/1     Running                 0             19m
cart-deployment-6bd6749cd4-xrlcd    1/1     Running                 0             19m
catalogue-deployment-5f68fcc4b9-gtkcx   0/1     Pending             0             19m
catalogue-deployment-5f68fcc4b9-mxgxk   0/1     Pending             0             19m
dispatch-deployment-f48685b4f-hqrbz     1/1     Running             0             19m
mongo-ss-0                          1/1     Running                 0             19m
mongo-ss-1                          1/1     Running                 0             6m15s
mongo-ss-2                          1/1     Running                 0             6m7s
mysql-ss-0                          1/1     Running                 0             19m
mysql-ss-1                          1/1     Running                 1 (13m ago)   17m
mysql-ss-2                          1/1     Running                 0             17m
payment-deployment-5bbb688856-cnx5d     1/1     Running             0             19m
payment-deployment-5bbb688856-n8k9g     1/1     Running             0             19m
rabbitmq-0                          0/1     Init:ImagePullBackOff   0             19m
ratings-deployment-6488f96bf4-zwkwg     1/1     Running             0             19m
redis-master-0                      1/1     Running                 0             19m
redis-replicas-0                    1/1     Running                 0             19m
redis-replicas-1                    1/1     Running                 0             8m17s
redis-replicas-2                    1/1     Running                 0             7m43s
shipping-deployment-5b7b9ddf59-9qjx2    1/1     Running             0             19m
shipping-deployment-5b7b9ddf59-xxlcj    1/1     Running             0             19m
user-deployment-7d9b7c49cb-znvz5    1/1     Running                 0             19m
web-deployment-8b8965f45-qc972      1/1     Running                 0             19m
```

OpenCost Chart

Overview

OpenCost provides real-time cost visibility inside Kubernetes clusters.

Purpose in This Platform

- Namespace cost allocation
- Pod-level cost breakdown
- Spot vs On-Demand comparison
- FinOps visibility

Architecture

- Prometheus integration
- Cost model engine
- Cost API endpoint

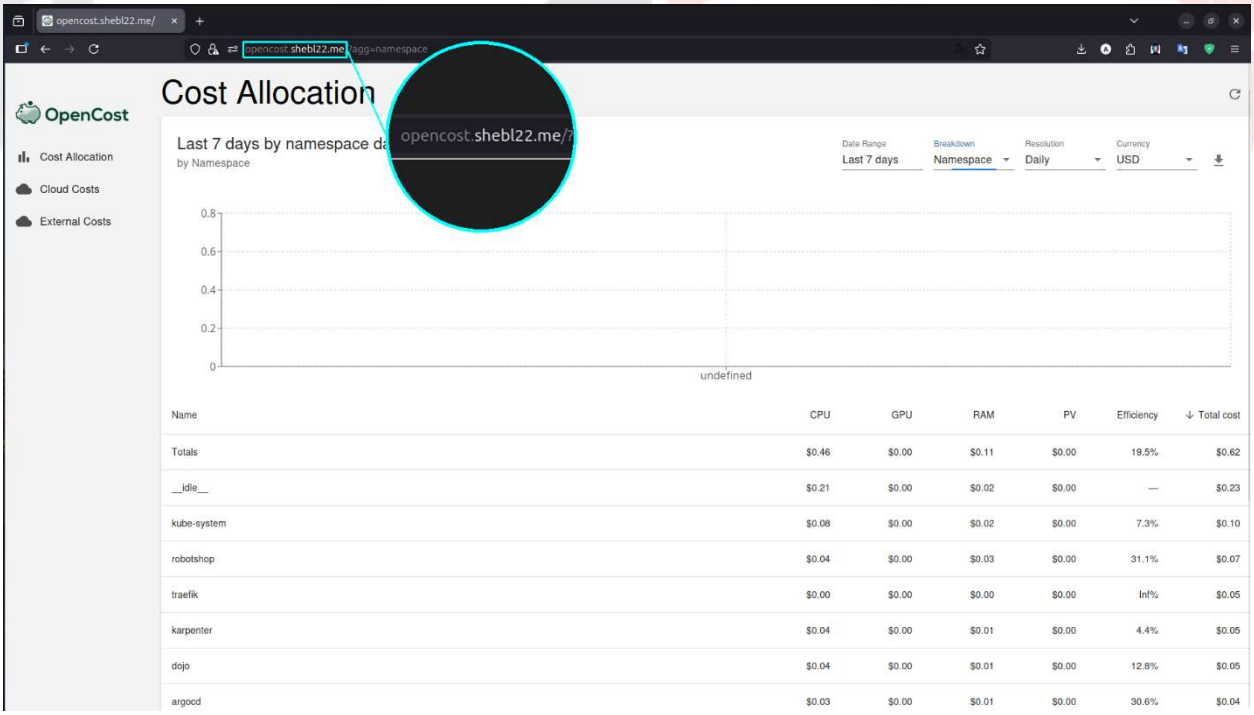
What It Tracks

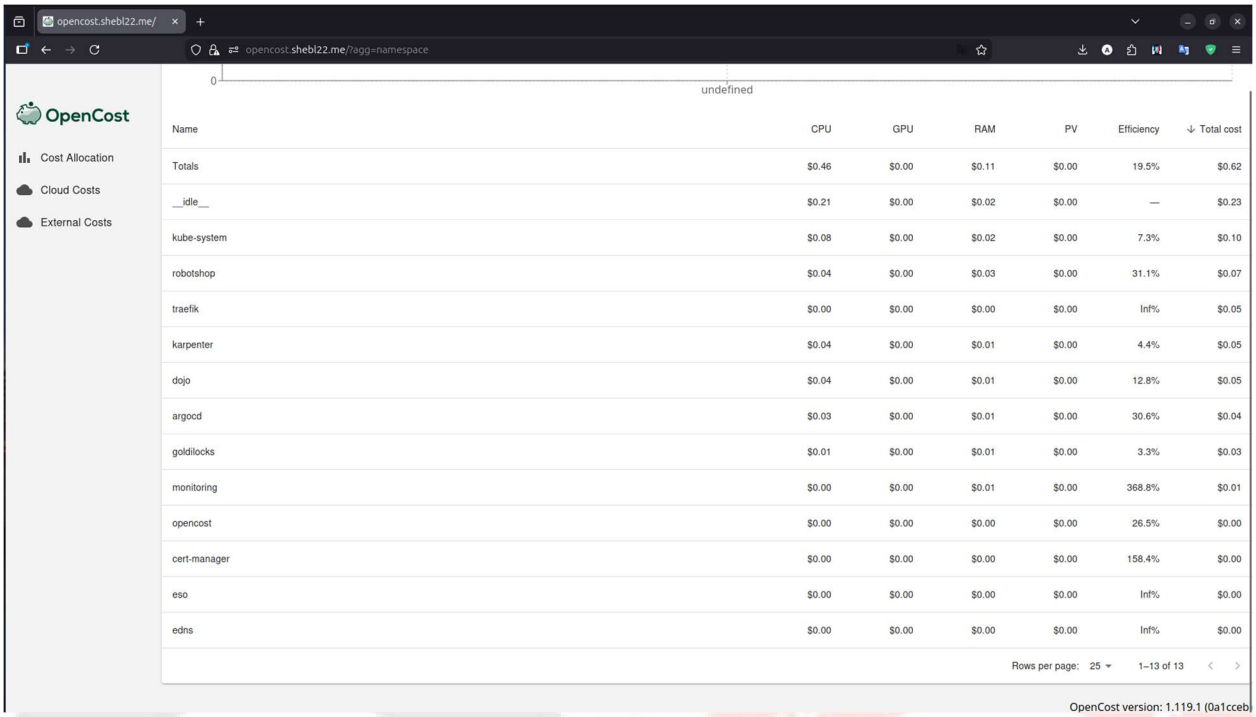
- CPU usage cost
- Memory usage cost
- Network egress cost
- Persistent volume cost

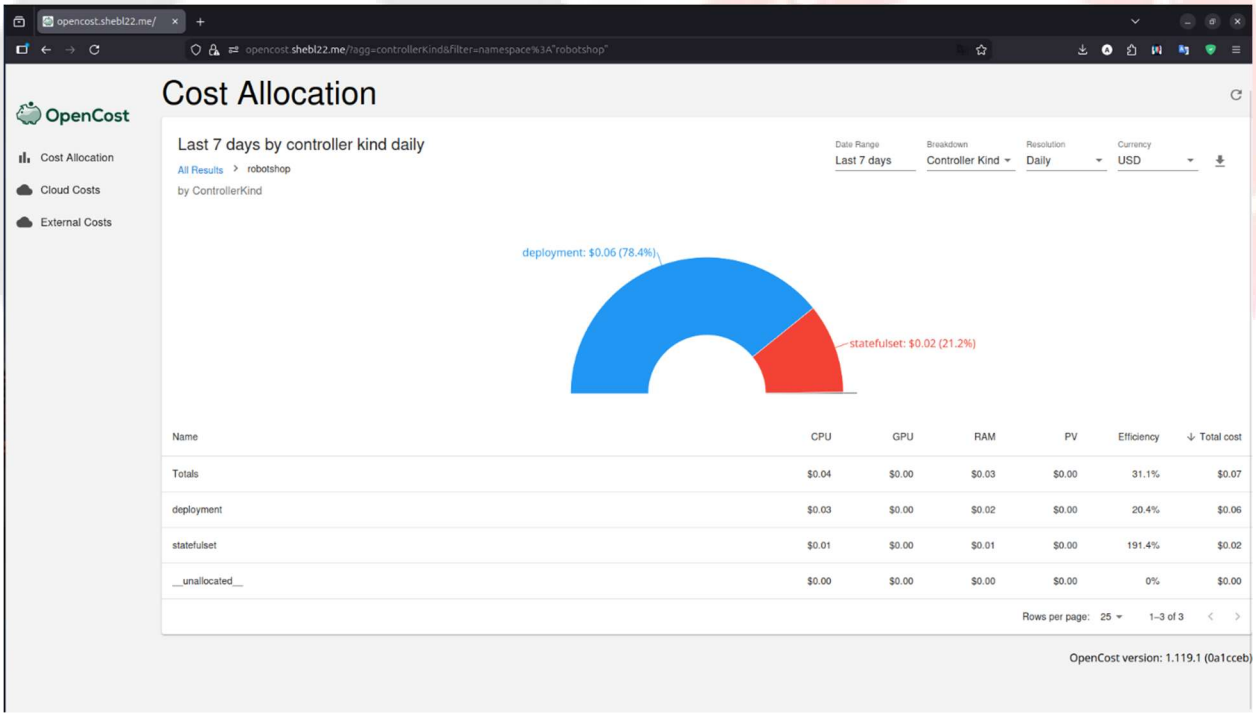
Business Impact

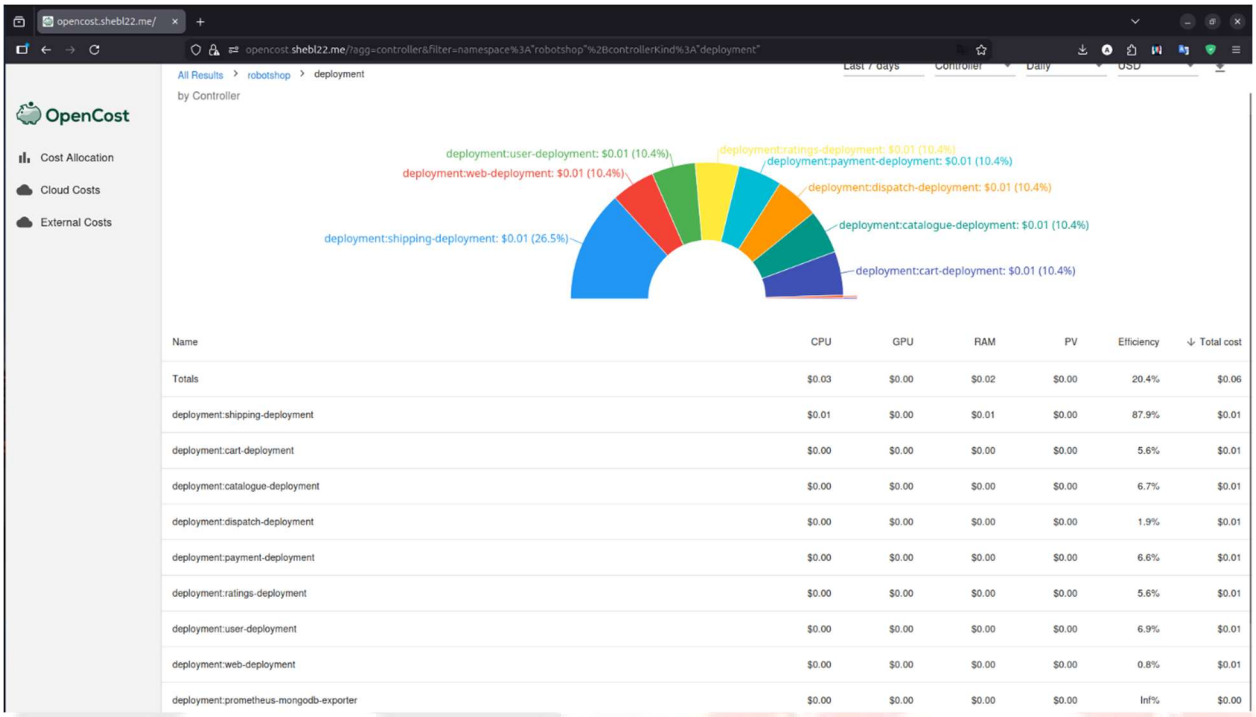
- Identify waste
- Optimize node sizing
- Reduce cloud bill
- Justify scaling decisions

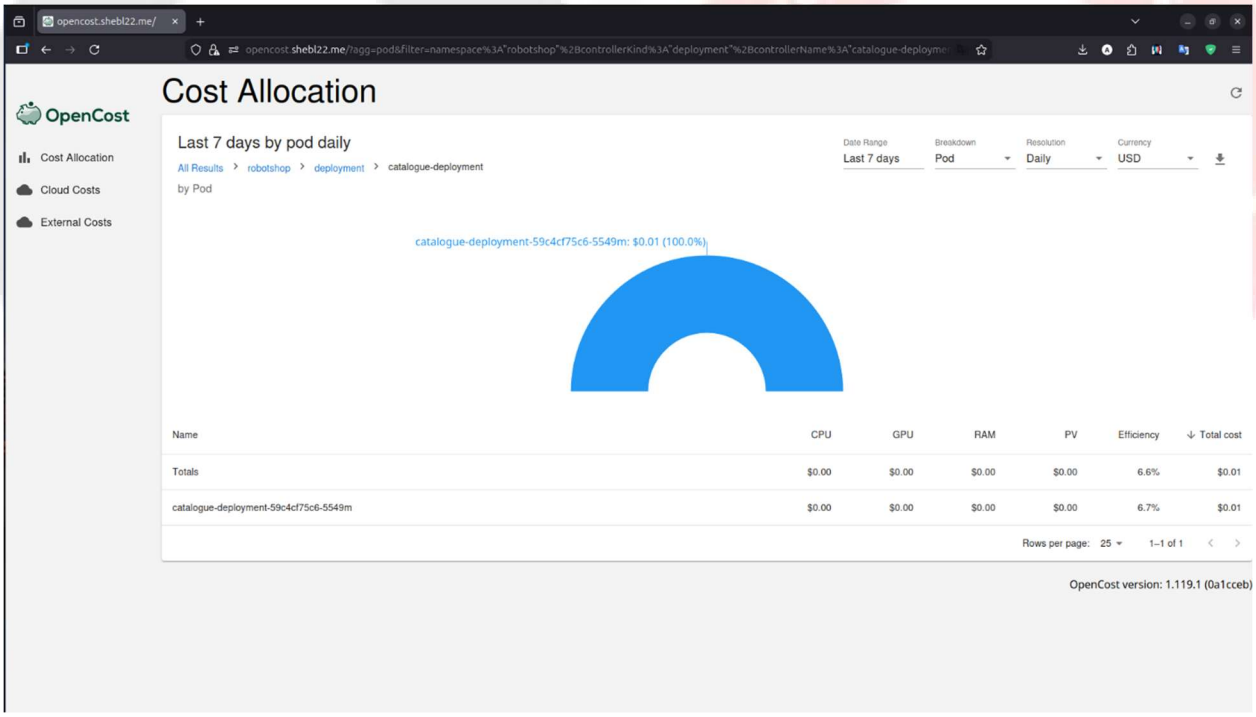
Results:

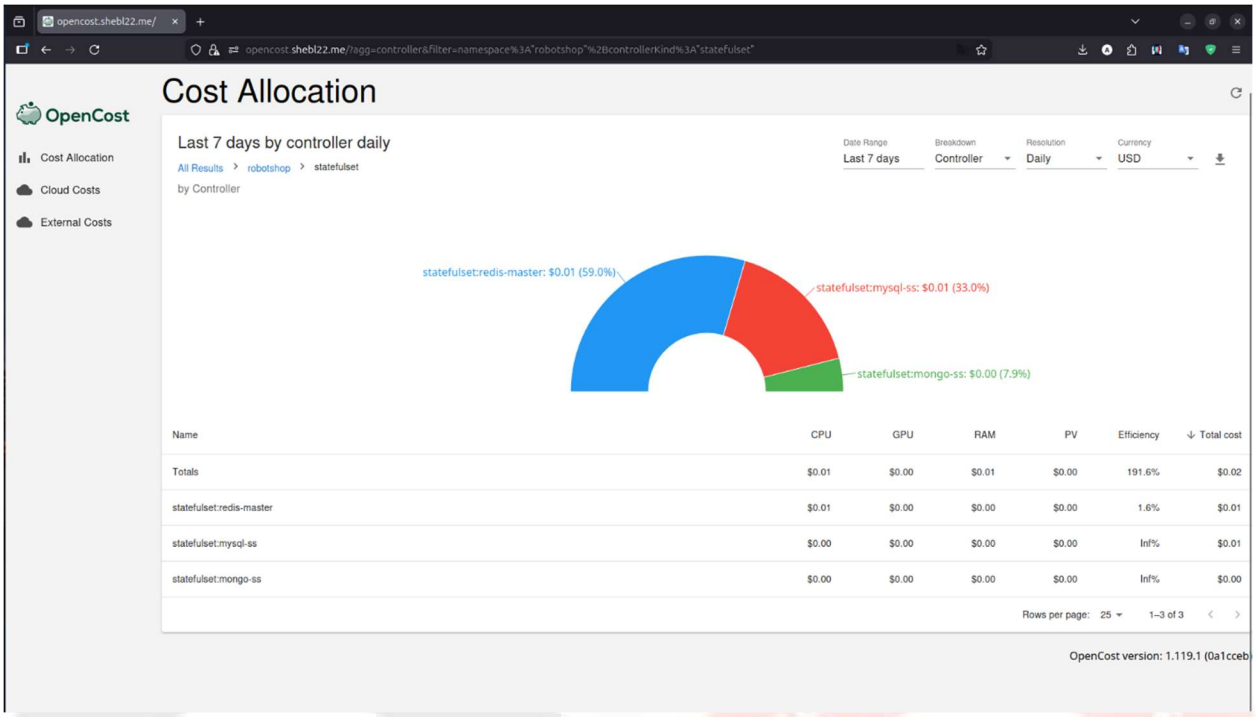


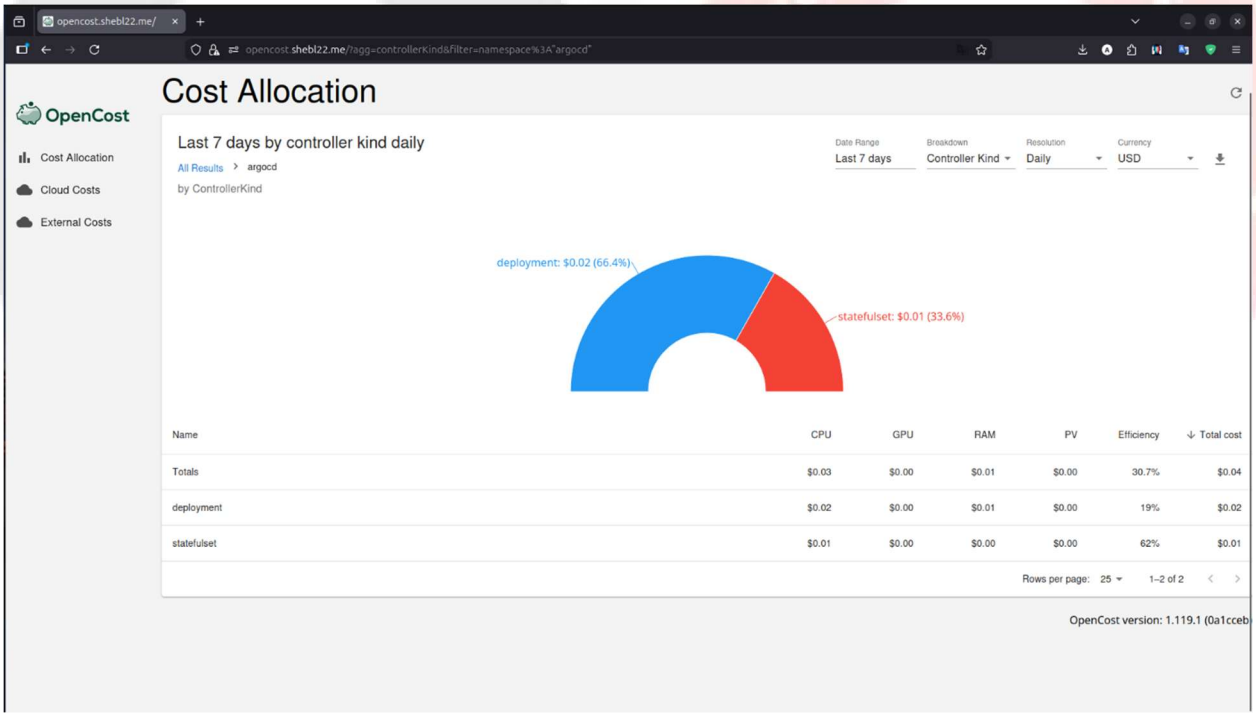

opencost.shebl22.me/
opencost.shebl22.me/?agg=namespace
OpenCost
Cost Allocation
Cloud Costs
External Costs
0
undefined
Name CPU GPU RAM PV Efficiency Total cost
Totals $0.46 $0.00 $0.11 $0.00 19.5% $0.62
__idle__ $0.21 $0.00 $0.02 $0.00 — $0.23
kube-system $0.08 $0.00 $0.02 $0.00 7.3% $0.10
robotshop $0.04 $0.00 $0.03 $0.00 31.1% $0.07
traefik $0.00 $0.00 $0.00 $0.00 Inf% $0.05
karpenter $0.04 $0.00 $0.01 $0.00 4.4% $0.05
dojo $0.04 $0.00 $0.01 $0.00 12.8% $0.05
argocd $0.03 $0.00 $0.01 $0.00 30.6% $0.04
goldilocks $0.01 $0.00 $0.01 $0.00 3.3% $0.03
monitoring $0.00 $0.00 $0.01 $0.00 368.8% $0.01
opencost $0.00 $0.00 $0.00 $0.00 26.5% $0.00
cert-manager $0.00 $0.00 $0.00 $0.00 158.4% $0.00
eso $0.00 $0.00 $0.00 $0.00 Inf% $0.00
edns $0.00 $0.00 $0.00 $0.00 Inf% $0.00
Rows per page: 25 1–13 of 13
OpenCost version: 1.119.1 (0a1cceb)


opencost.shebl22.me/
opencost.shebl22.me/?agg=controllerKind&filter=namespace%3A"robotshop"
OpenCost
Cost Allocation
Cloud Costs
External Costs
Cost Allocation
Last 7 days by controller kind daily
All Results > robotshop
by ControllerKind
Date Range Last 7 days
Breakdown Controller Kind
Resolution Daily
Currency USD
deployment: $0.06 (78.4%)
statefulset: $0.02 (21.2%)
Name CPU GPU RAM PV Efficiency Total cost
Totals $0.04 $0.00 $0.03 $0.00 31.1% $0.07
deployment $0.03 $0.00 $0.02 $0.00 20.4% $0.06
statefulset $0.01 $0.00 $0.01 $0.00 191.4% $0.02
__unallocated__ $0.00 $0.00 $0.00 $0.00 0% $0.00
Rows per page: 25 1–3 of 3
OpenCost version: 1.119.1 (0a1cceb)


opencost.shebl22.me/
OpenCost
Cost Allocation
Cloud Costs
External Costs
All Results > robotshop > deployment
by Controller
deployment:user-deployment: $0.01 (10.4%)
deployment:web-deployment: $0.01 (10.4%)
deployment:ratings-deployment: $0.01 (10.4%)
deployment:payment-deployment: $0.01 (10.4%)
deployment:dispatch-deployment: $0.01 (10.4%)
deployment:catalogue-deployment: $0.01 (10.4%)
deployment:cart-deployment: $0.01 (10.4%)
deployment:shipping-deployment: $0.01 (26.5%)
Name CPU GPU RAM PV Efficiency Total cost
Totals $0.03 $0.00 $0.02 $0.00 20.4% $0.06
deployment:shipping-deployment $0.01 $0.00 $0.01 $0.00 87.9% $0.01
deployment:cart-deployment $0.00 $0.00 $0.00 $0.00 5.6% $0.01
deployment:catalogue-deployment $0.00 $0.00 $0.00 $0.00 6.7% $0.01
deployment:dispatch-deployment $0.00 $0.00 $0.00 $0.00 1.9% $0.01
deployment:payment-deployment $0.00 $0.00 $0.00 $0.00 6.6% $0.01
deployment:ratings-deployment $0.00 $0.00 $0.00 $0.00 5.6% $0.01
deployment:user-deployment $0.00 $0.00 $0.00 $0.00 6.9% $0.01
deployment:web-deployment $0.00 $0.00 $0.00 $0.00 0.8% $0.01
deployment:prometheus-mongodb-exporter $0.00 $0.00 $0.00 $0.00 Inf% $0.00


opencost.shebl22.me/
OpenCost
Cost Allocation
Cloud Costs
External Costs
Cost Allocation
Last 7 days by pod daily
All Results > robotshop > deployment > catalogue-deployment
by Pod
Date Range Last 7 days
Breakdown Pod
Resolution Daily
Currency USD
catalogue-deployment-59c4cf75c6-5549m: $0.01 (100.0%)
Name CPU GPU RAM PV Efficiency Total cost
Totals $0.00 $0.00 $0.00 $0.00 6.6% $0.01
catalogue-deployment-59c4cf75c6-5549m $0.00 $0.00 $0.00 $0.00 6.7% $0.01
Rows per page: 25 1–1 of 1
OpenCost version: 1.119.1 (0a1cceb)


opencost.shebl22.me/
opencost.shebl22.me/?agg=controller&filter=namespace%3A"robotshop"%2BcontrollerKind%3A"statefulset"
OpenCost
Cost Allocation
Cloud Costs
External Costs
Cost Allocation
Last 7 days by controller daily
All Results > robotshop > statefulset
by Controller
Date Range
Last 7 days
Breakdown
Controller
Resolution
Daily
Currency
USD
statefulset:redis-master: $0.01 (59.0%)
statefulset:mysql-ss: $0.01 (33.0%)
statefulset:mongo-ss: $0.00 (7.9%)
Name
CPU
GPU
RAM
PV
Efficiency
Total cost
Totals $0.01 $0.00 $0.01 $0.00 191.6% $0.02
statefulset:redis-master $0.01 $0.00 $0.00 $0.00 1.6% $0.01
statefulset:mysql-ss $0.00 $0.00 $0.00 $0.00 Inf% $0.01
statefulset:mongo-ss $0.00 $0.00 $0.00 $0.00 Inf% $0.00
Rows per page: 25
1–3 of 3
OpenCost version: 1.119.1 (0a1cceb


opencost.shebl22.me/
opencost.shebl22.me/?agg=controllerKind&filter=namespace%3A"argocd"
OpenCost
Cost Allocation
Cloud Costs
External Costs
Cost Allocation
Last 7 days by controller kind daily
All Results > argocd
by ControllerKind
Date Range
Last 7 days
Breakdown
Controller Kind
Resolution
Daily
Currency
USD
deployment: $0.02 (66.4%)
statefulset: $0.01 (33.6%)
Name
CPU
GPU
RAM
PV
Efficiency
Total cost
Totals $0.03 $0.00 $0.01 $0.00 30.7% $0.04
deployment $0.02 $0.00 $0.01 $0.00 19% $0.02
statefulset $0.01 $0.00 $0.00 $0.00 62% $0.01
Rows per page: 25
1–2 of 2
OpenCost version: 1.119.1 (0a1cceb

Goldilocks Chart

Overview

Goldilocks analyzes resource usage and recommends optimal CPU/memory requests and limits.

Built on top of Vertical Pod Autoscaler.

Purpose in This Platform

- Right-size workloads
- Prevent overprovisioning
- Improve cluster efficiency
- Support FinOps goals

How It Works

1. Collects Prometheus metrics
2. Uses VPA recommendation engine
3. Provides dashboard suggestions

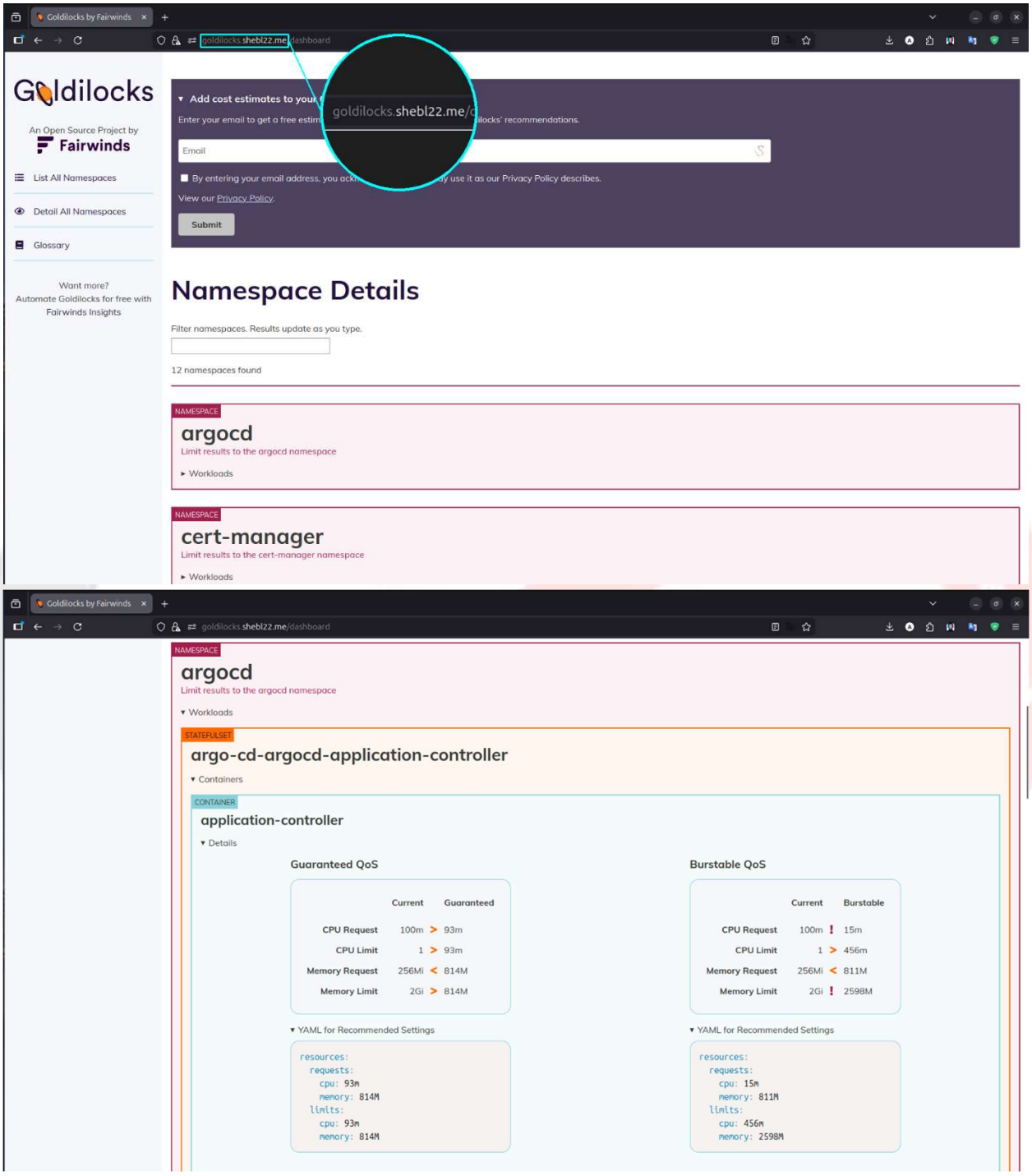
Output

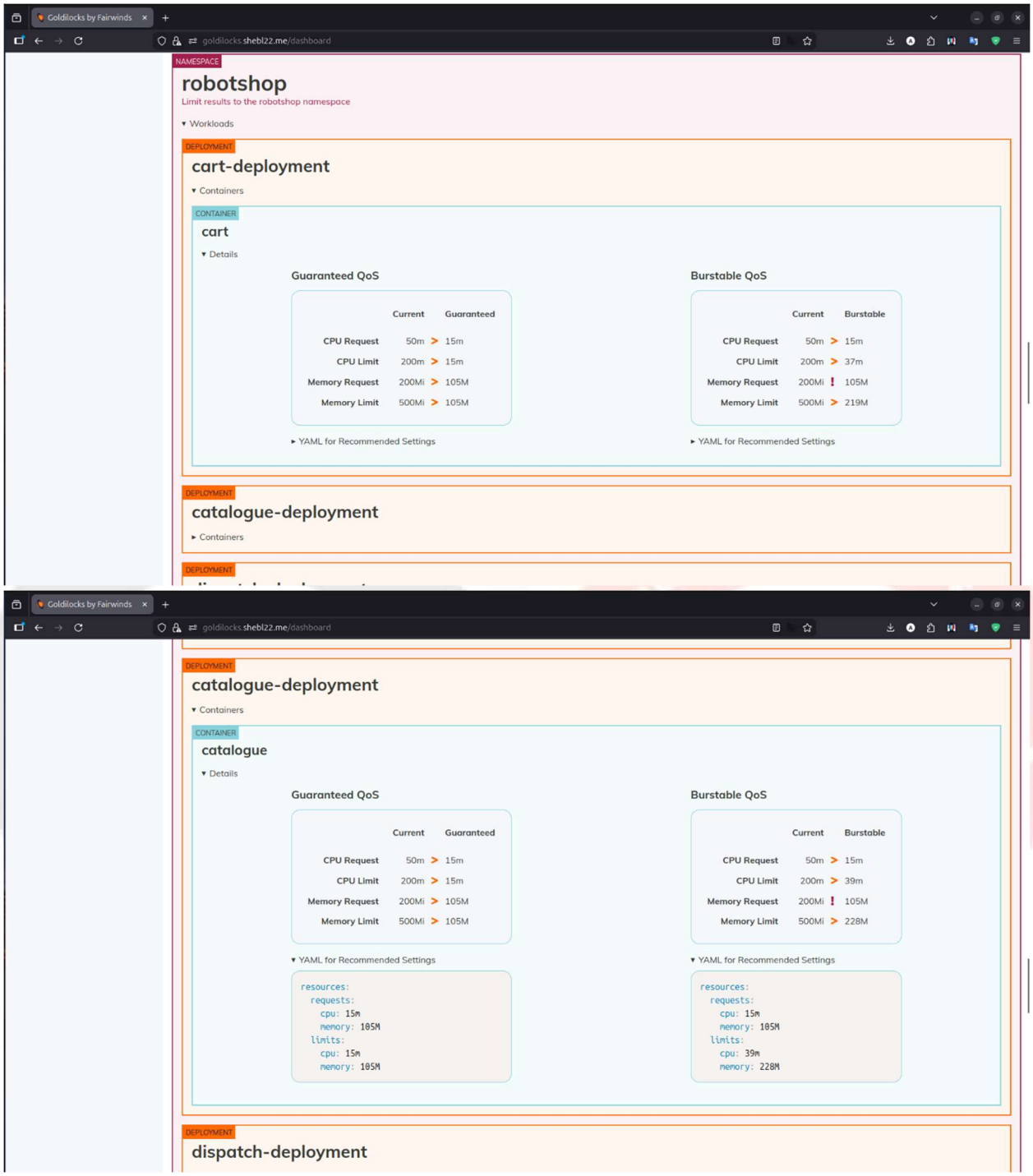
- Recommended CPU requests
- Recommended memory limits
- Efficiency comparison

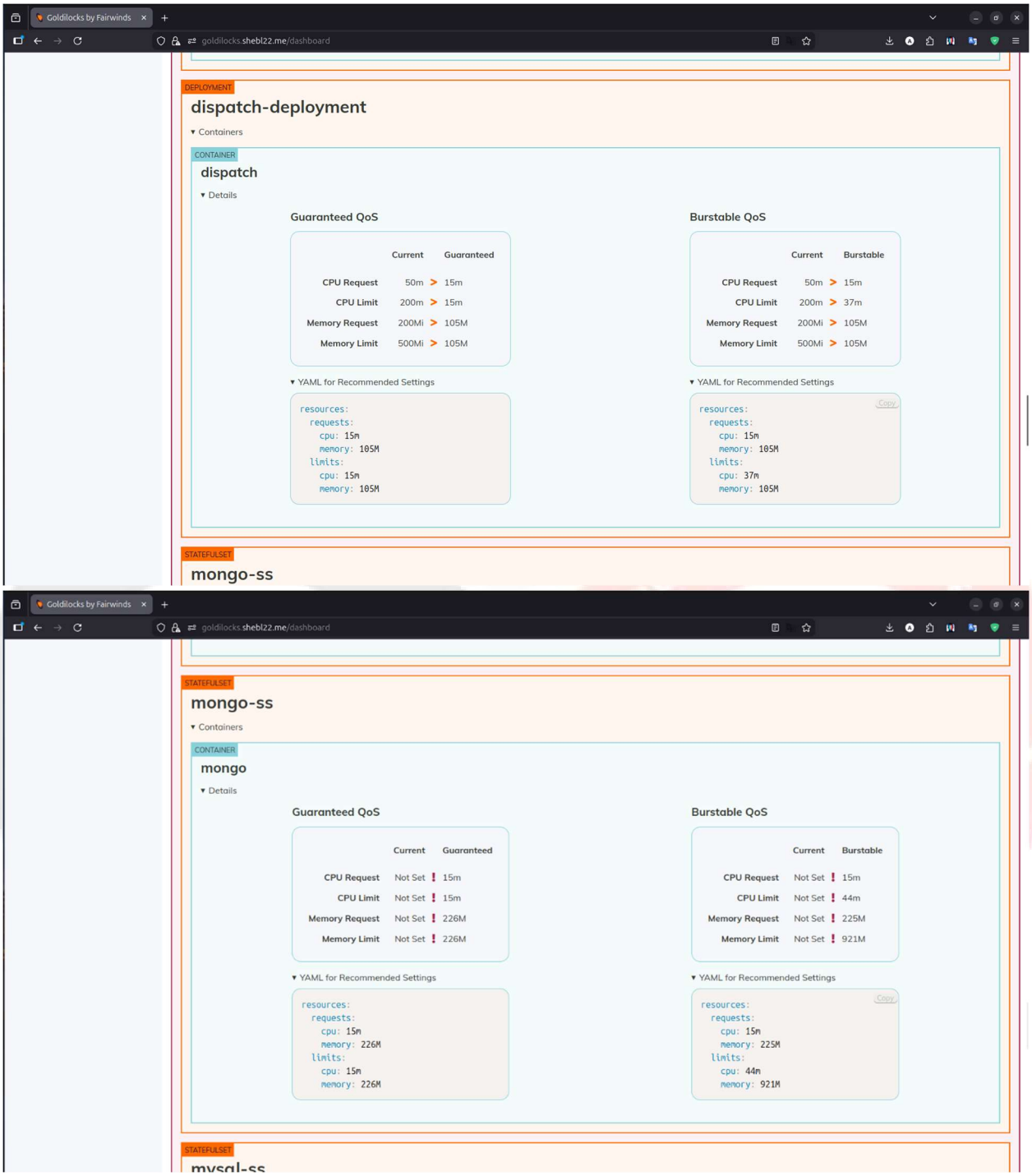
Production Use

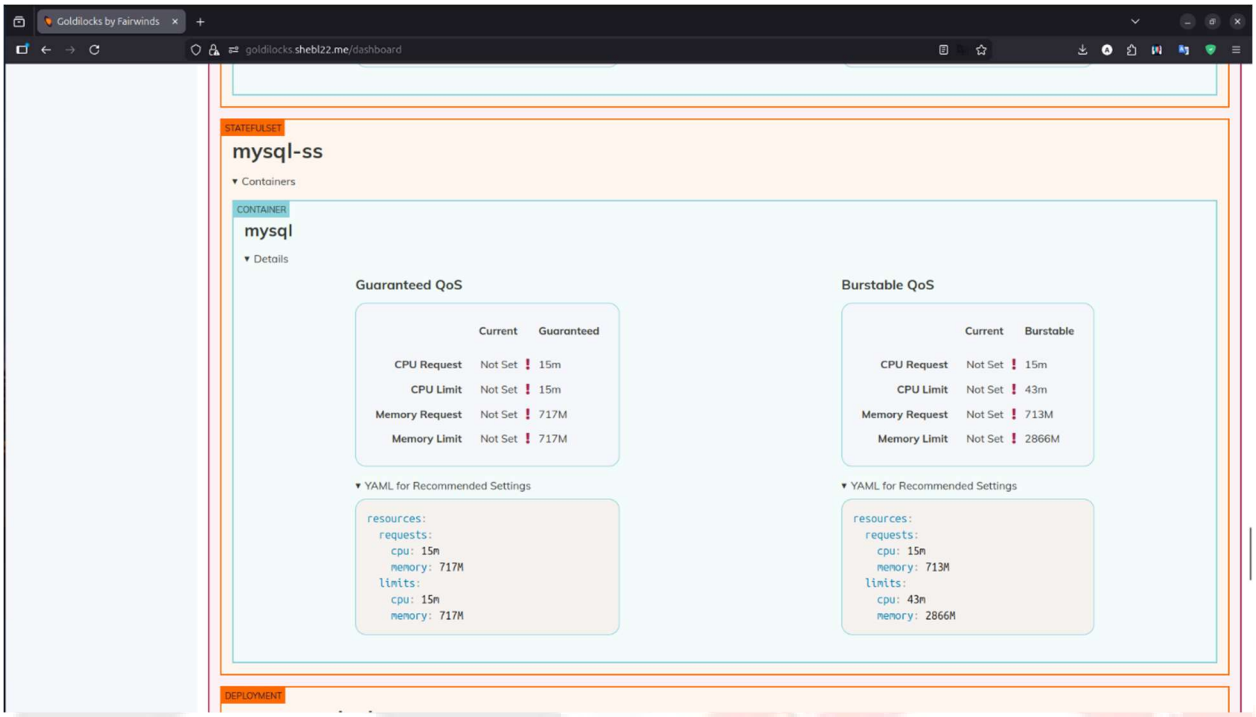
Recommendations are reviewed before updating Helm values.

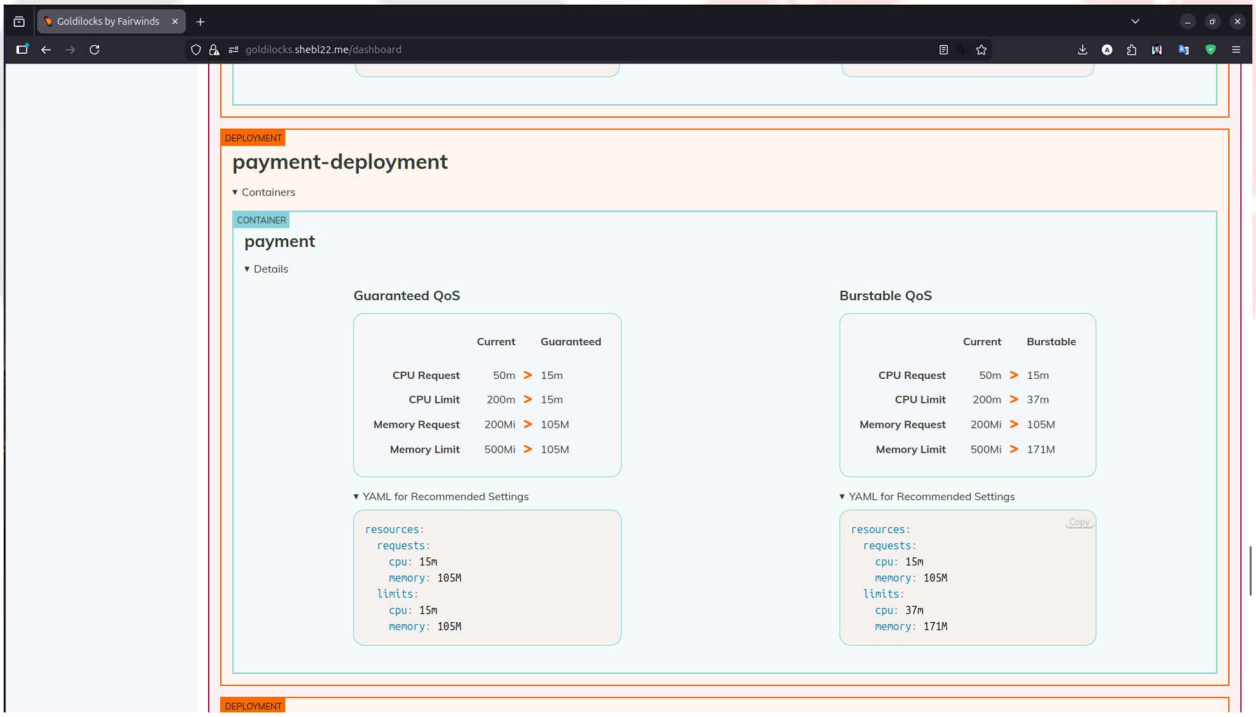
Results:

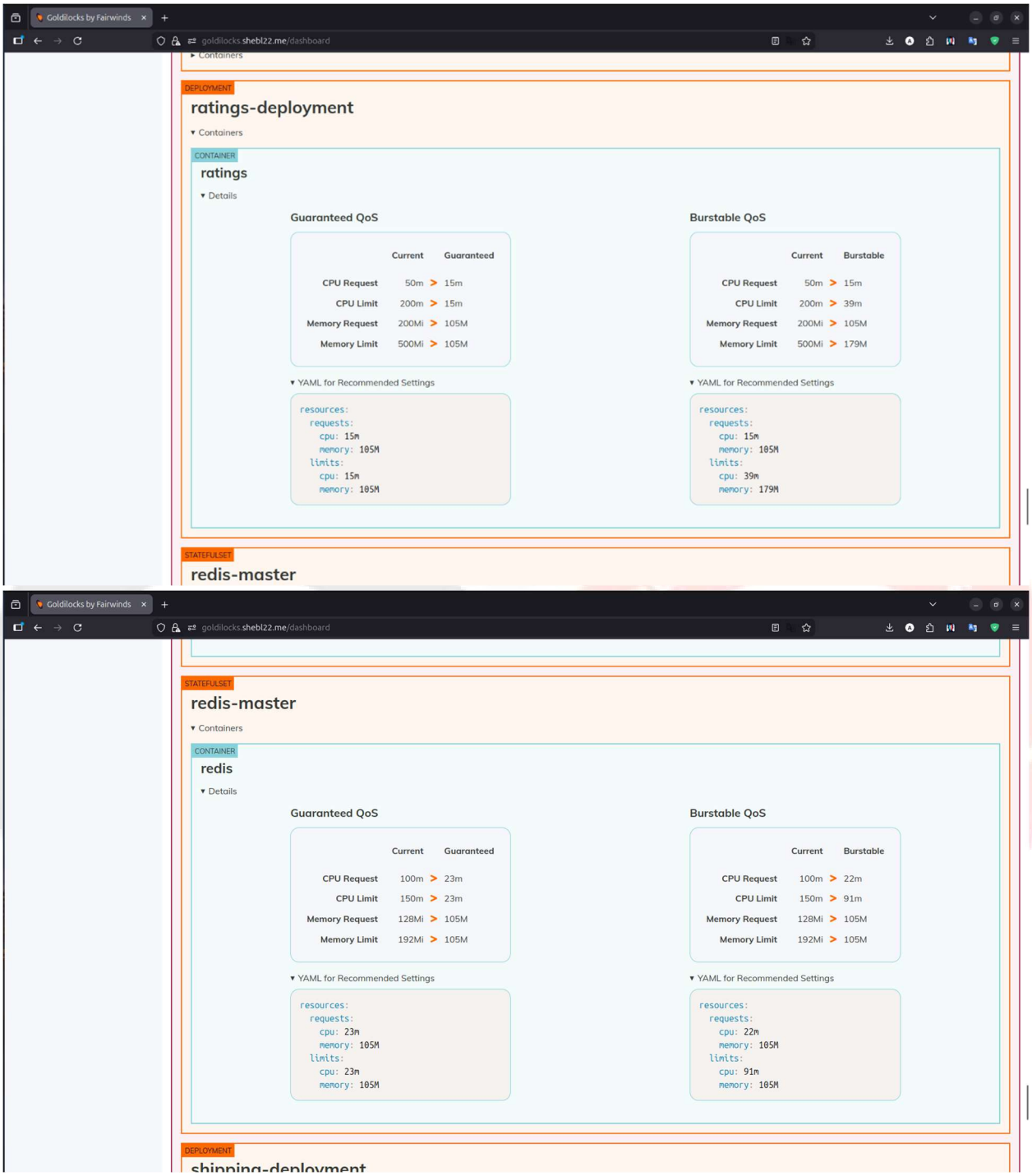

Goldilocks by Fairwinds
goldilocks.shebl22.me/dashboard
Goldilocks
An Open Source Project by
Fairwinds
List All Namespaces
Detail All Namespaces
Glossary
Want more?
Automate Goldilocks for free with Fairwinds Insights
Add cost estimates to your
Enter your email to get a free estim
goldilocks.shebl22.me/
ilocks' recommendations.
Email
By entering your email address, you ack
y use it as our Privacy Policy describes.
View our Privacy Policy.
Submit
Namespace Details
Filter namespaces. Results update as you type.
12 namespaces found
NAMESPACE
argocd
Limit results to the argocd namespace
Workloads
NAMESPACE
cert-manager
Limit results to the cert-manager namespace
Workloads
Goldilocks by Fairwinds
goldilocks.shebl22.me/dashboard
NAMESPACE
argocd
Limit results to the argocd namespace
Workloads
STATEFULSET
argo-cd-argocd-application-controller
Containers
CONTAINER
application-controller
Details
Guaranteed QoS
Current
Guaranteed
CPU Request
100m > 93m
CPU Limit
1 > 93m
Memory Request
256Mi < 814M
Memory Limit
2Gi > 814M
YAML for Recommended Settings
resources:
requests:
cpu: 93m
memory: 814M
limits:
cpu: 93m
memory: 814M
Burstable QoS
Current
Burstable
CPU Request
100m ! 15m
CPU Limit
1 > 456m
Memory Request
256Mi < 811M
Memory Limit
2Gi ! 2598M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 811M
limits:
cpu: 456m
memory: 2598M

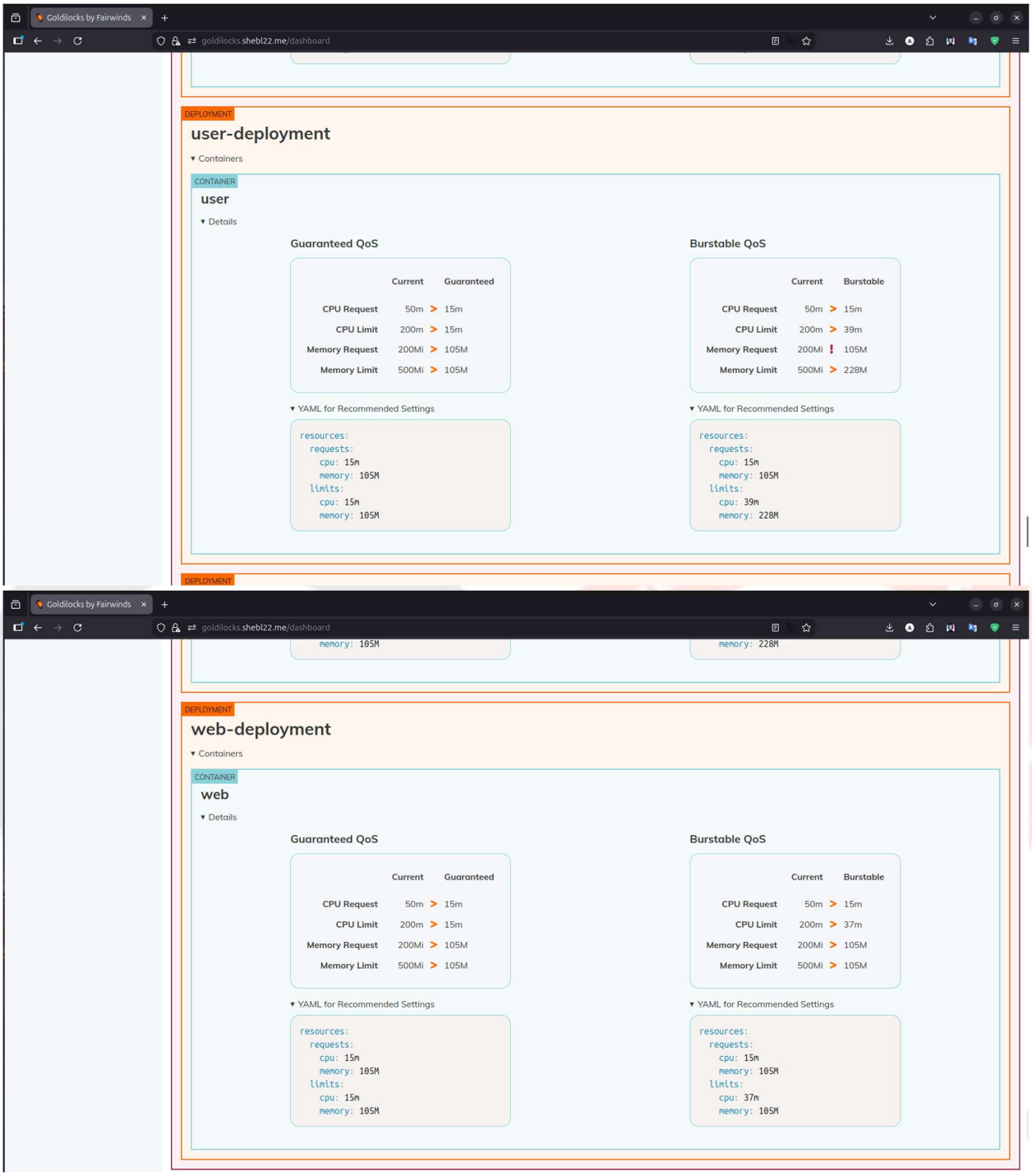

Goldilocks by Fairwinds
goldilocks.shebl22.me/dashboard
NAMESPACE
robotshop
Limit results to the robotshop namespace
Workloads
DEPLOYMENT
cart-deployment
Containers
CONTAINER
cart
Details
Guaranteed QoS
Current Guaranteed
CPU Request 50m > 15m
CPU Limit 200m > 15m
Memory Request 200Mi > 105M
Memory Limit 500Mi > 105M
YAML for Recommended Settings
Burstable QoS
Current Burstable
CPU Request 50m > 15m
CPU Limit 200m > 37m
Memory Request 200Mi ! 105M
Memory Limit 500Mi > 219M
YAML for Recommended Settings
DEPLOYMENT
catalogue-deployment
Containers
DEPLOYMENT
Goldilocks by Fairwinds
goldilocks.shebl22.me/dashboard
DEPLOYMENT
catalogue-deployment
Containers
CONTAINER
catalogue
Details
Guaranteed QoS
Current Guaranteed
CPU Request 50m > 15m
CPU Limit 200m > 15m
Memory Request 200Mi > 105M
Memory Limit 500Mi > 105M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 105M
limits:
cpu: 15m
memory: 105M
Burstable QoS
Current Burstable
CPU Request 50m > 15m
CPU Limit 200m > 39m
Memory Request 200Mi ! 105M
Memory Limit 500Mi > 228M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 105M
limits:
cpu: 39m
memory: 228M
DEPLOYMENT
dispatch-deployment

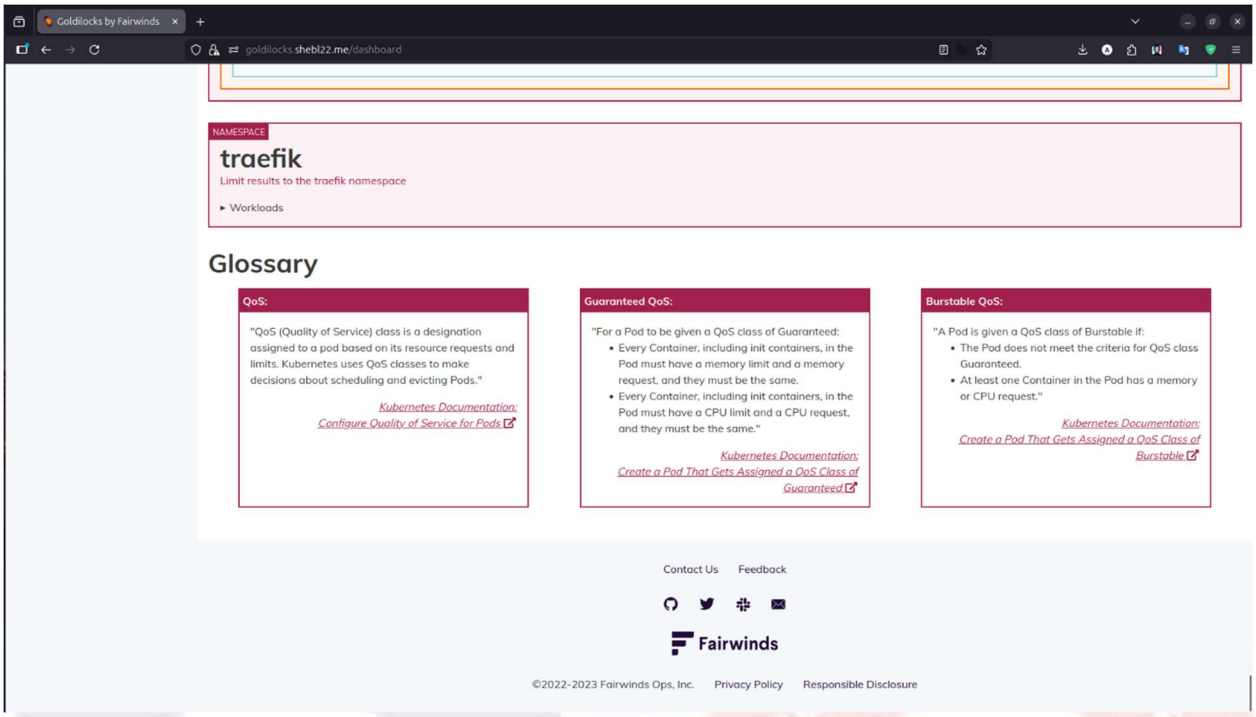

Goldilocks by Fairwinds
goldilocks.shebl22.me/dashboard
DEPLOYMENT
dispatch-deployment
Containers
CONTAINER
dispatch
Details
Guaranteed QoS
Current Guaranteed
CPU Request 50m > 15m
CPU Limit 200m > 15m
Memory Request 200Mi > 105M
Memory Limit 500Mi > 105M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 105M
limits:
cpu: 15m
memory: 105M
Burstable QoS
Current Burstable
CPU Request 50m > 15m
CPU Limit 200m > 37m
Memory Request 200Mi > 105M
Memory Limit 500Mi > 105M
YAML for Recommended Settings
Copy
resources:
requests:
cpu: 15m
memory: 105M
limits:
cpu: 37m
memory: 105M
STATEFULSET
mongo-ss
Goldilocks by Fairwinds
goldilocks.shebl22.me/dashboard
STATEFULSET
mongo-ss
Containers
CONTAINER
mongo
Details
Guaranteed QoS
Current Guaranteed
CPU Request Not Set ! 15m
CPU Limit Not Set ! 15m
Memory Request Not Set ! 226M
Memory Limit Not Set ! 226M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 226M
limits:
cpu: 15m
memory: 226M
Burstable QoS
Current Burstable
CPU Request Not Set ! 15m
CPU Limit Not Set ! 44m
Memory Request Not Set ! 225M
Memory Limit Not Set ! 921M
YAML for Recommended Settings
Copy
resources:
requests:
cpu: 15m
memory: 225M
limits:
cpu: 44m
memory: 921M
STATEFULSET
mysql-ss


Goldilocks by Fairwinds
goldilocks.shebl22.me/dashboard
STATEFULSET
mysql-ss
Containers
CONTAINER
mysql
Details
Guaranteed QoS
Current Guaranteed
CPU Request Not Set 15m
CPU Limit Not Set 15m
Memory Request Not Set 717M
Memory Limit Not Set 717M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 717M
limits:
cpu: 15m
memory: 717M
Burstable QoS
Current Burstable
CPU Request Not Set 15m
CPU Limit Not Set 43m
Memory Request Not Set 713M
Memory Limit Not Set 2866M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 713M
limits:
cpu: 43m
memory: 2866M
DEPLOYMENT


Goldilocks by Fairwinds
goldilocks.shebl22.me/dashboard
DEPLOYMENT
payment-deployment
Containers
CONTAINER
payment
Details
Guaranteed QoS
Current Guaranteed
CPU Request 50m > 15m
CPU Limit 200m > 15m
Memory Request 200Mi > 105M
Memory Limit 500Mi > 105M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 105M
limits:
cpu: 15m
memory: 105M
Burstable QoS
Current Burstable
CPU Request 50m > 15m
CPU Limit 200m > 37m
Memory Request 200Mi > 105M
Memory Limit 500Mi > 171M
YAML for Recommended Settings
Copy
resources:
requests:
cpu: 15m
memory: 105M
limits:
cpu: 37m
memory: 171M
DEPLOYMENT


Goldilocks by Fairwinds
goldilocks.shebl22.me/dashboard
Containers
DEPLOYMENT
ratings-deployment
Containers
CONTAINER
ratings
Details
Guaranteed QoS
Current Guaranteed
CPU Request 50m > 15m
CPU Limit 200m > 15m
Memory Request 200Mi > 105M
Memory Limit 500Mi > 105M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 105M
limits:
cpu: 15m
memory: 105M
Burstable QoS
Current Burstable
CPU Request 50m > 15m
CPU Limit 200m > 39m
Memory Request 200Mi > 105M
Memory Limit 500Mi > 179M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 105M
limits:
cpu: 39m
memory: 179M
STATEFULSET
redis-master
Goldilocks by Fairwinds
goldilocks.shebl22.me/dashboard
STATEFULSET
redis-master
Containers
CONTAINER
redis
Details
Guaranteed QoS
Current Guaranteed
CPU Request 100m > 23m
CPU Limit 150m > 23m
Memory Request 128Mi > 105M
Memory Limit 192Mi > 105M
YAML for Recommended Settings
resources:
requests:
cpu: 23m
memory: 105M
limits:
cpu: 23m
memory: 105M
Burstable QoS
Current Burstable
CPU Request 100m > 22m
CPU Limit 150m > 91m
Memory Request 128Mi > 105M
Memory Limit 192Mi > 105M
YAML for Recommended Settings
resources:
requests:
cpu: 22m
memory: 105M
limits:
cpu: 91m
memory: 105M
DEPLOYMENT
shipping-deployment


Goldilocks by Fairwinds
goldilocks.shebl22.me/dashboard
DEPLOYMENT
user-deployment
Containers
CONTAINER
user
Details
Guaranteed QoS
Current Guaranteed
CPU Request 50m > 15m
CPU Limit 200m > 15m
Memory Request 200Mi > 105M
Memory Limit 500Mi > 105M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 105M
limits:
cpu: 15m
memory: 105M
Burstable QoS
Current Burstable
CPU Request 50m > 15m
CPU Limit 200m > 39m
Memory Request 200Mi ! 105M
Memory Limit 500Mi > 228M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 105M
limits:
cpu: 39m
memory: 228M
DEPLOYMENT
Goldilocks by Fairwinds
goldilocks.shebl22.me/dashboard
memory: 105M
memory: 228M
DEPLOYMENT
web-deployment
Containers
CONTAINER
web
Details
Guaranteed QoS
Current Guaranteed
CPU Request 50m > 15m
CPU Limit 200m > 15m
Memory Request 200Mi > 105M
Memory Limit 500Mi > 105M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 105M
limits:
cpu: 15m
memory: 105M
Burstable QoS
Current Burstable
CPU Request 50m > 15m
CPU Limit 200m > 37m
Memory Request 200Mi > 105M
Memory Limit 500Mi > 105M
YAML for Recommended Settings
resources:
requests:
cpu: 15m
memory: 105M
limits:
cpu: 37m
memory: 105M



Prometheus Stack Chart

Overview

Prometheus +
Grafana

Deployed via kube-prometheus-stack Helm chart.

Components

- Prometheus
- Alertmanager
- Grafana
- Node Exporter
- kube-state-metrics

Purpose in This Platform

- Cluster monitoring
- Application monitoring
- Node health tracking
- Alerting
- Metrics backend for:
    - OpenCost
    - Goldilocks
    - HPA

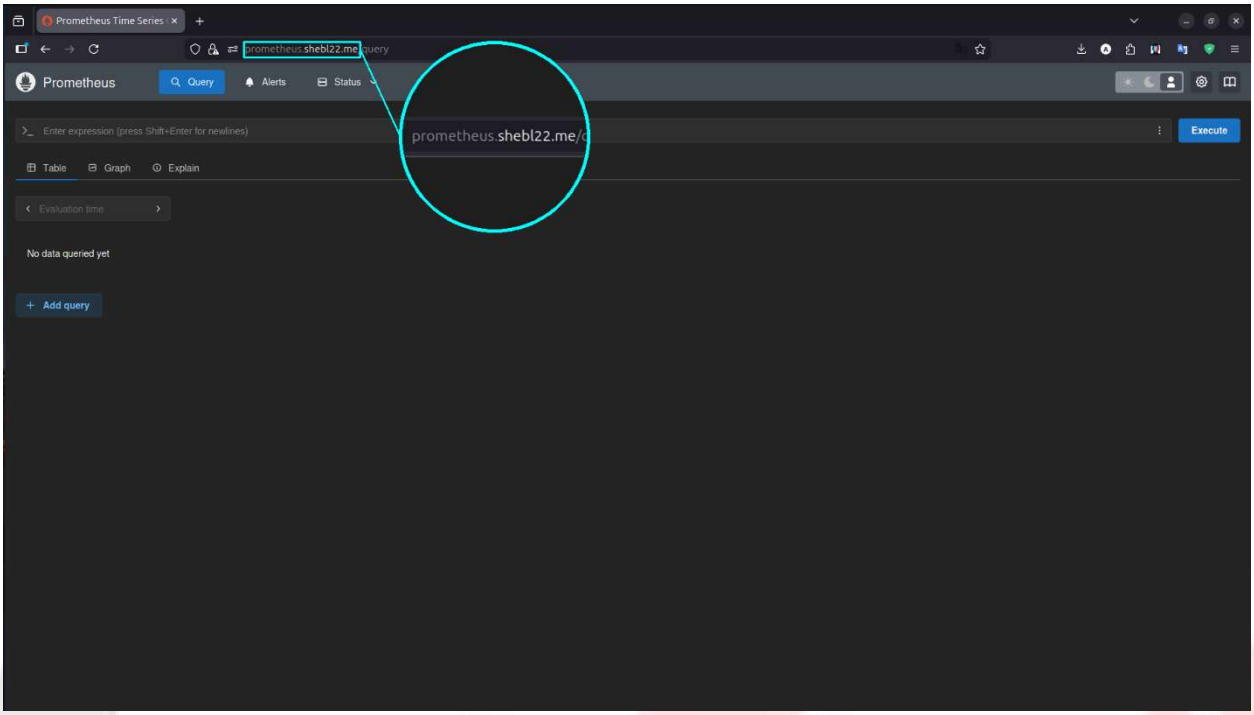
Metrics Collected

- CPU usage
- Memory usage
- Pod restarts
- API latency
- Node conditions

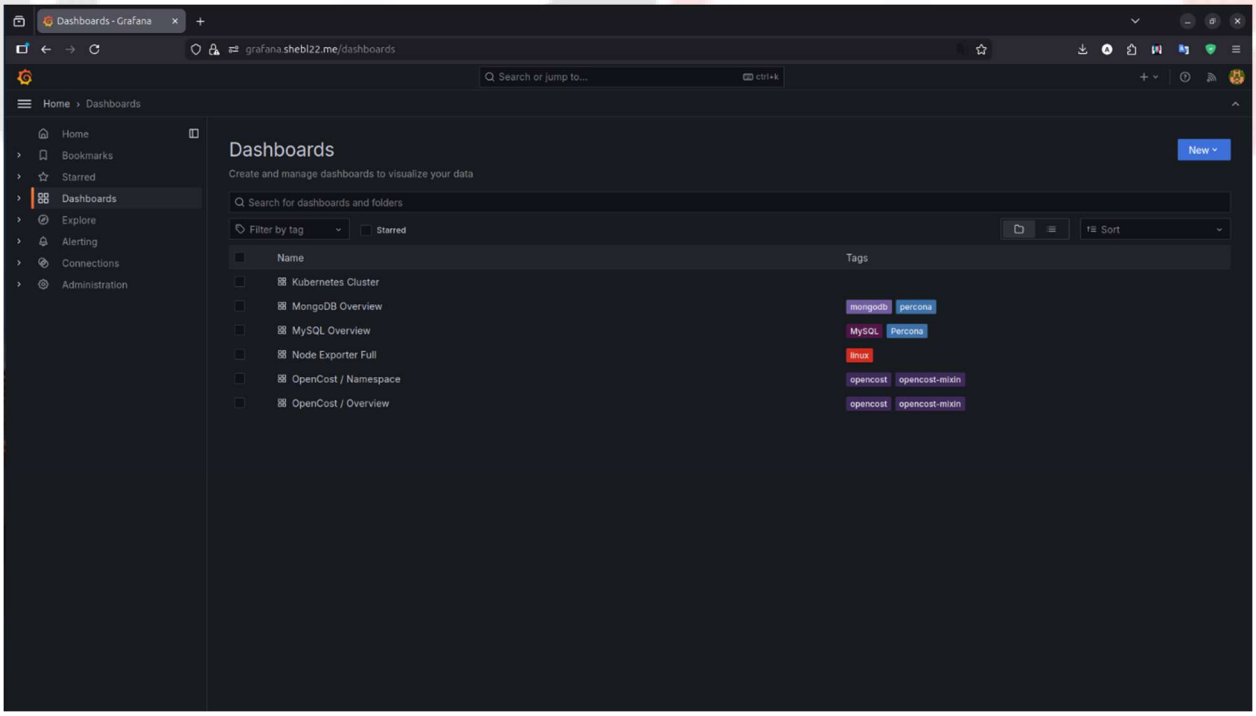
Alerting

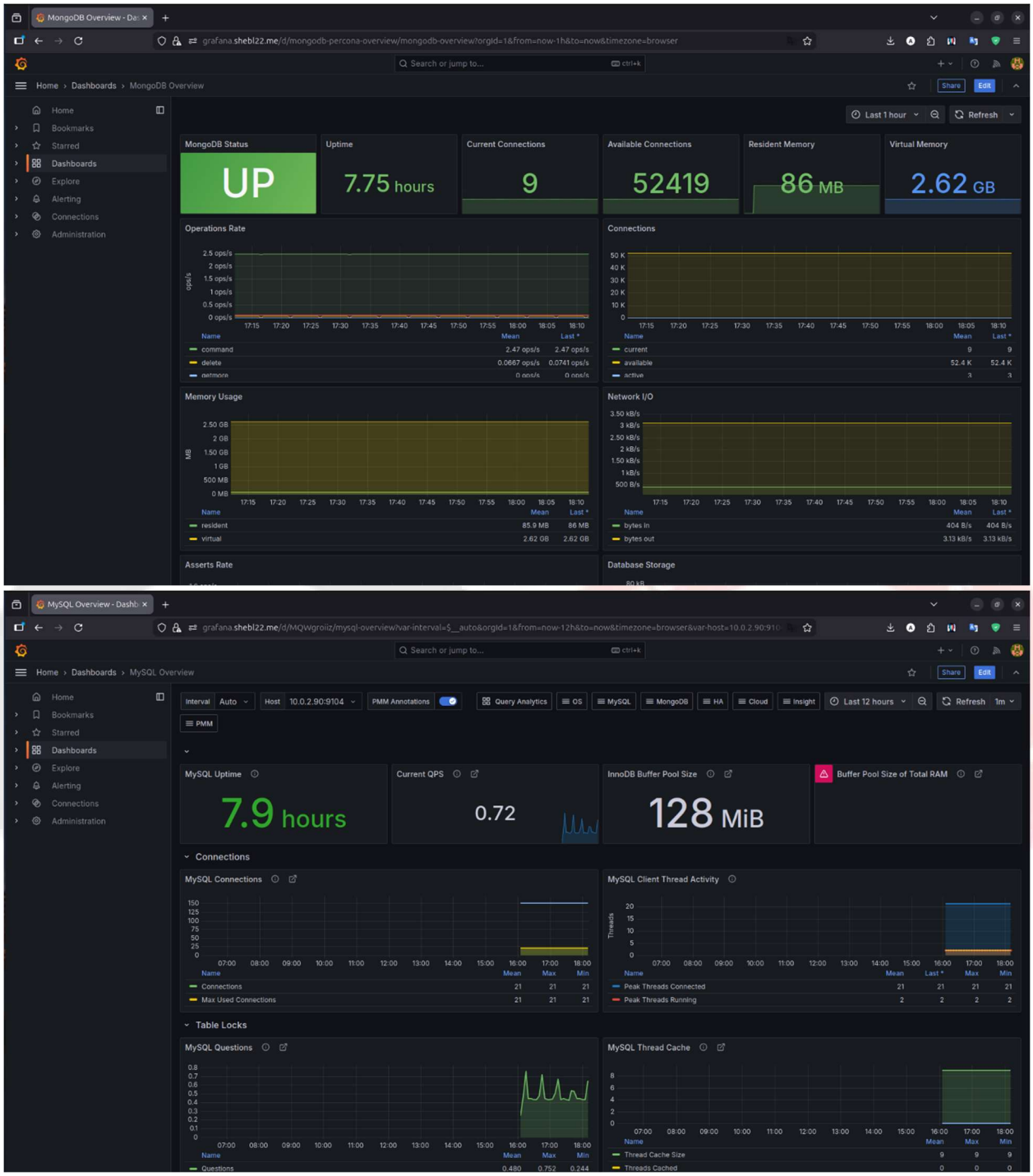
- High CPU usage
- Pod crash loops
- Node not ready
- Resource saturation

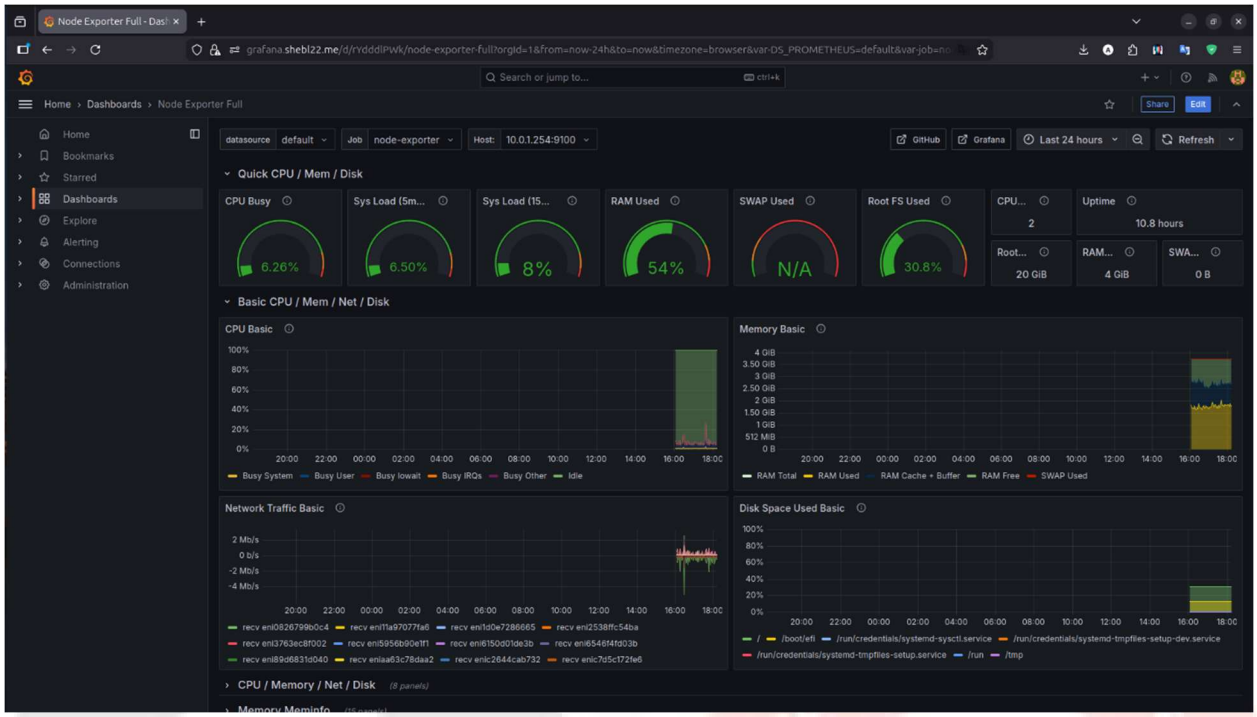
Results:

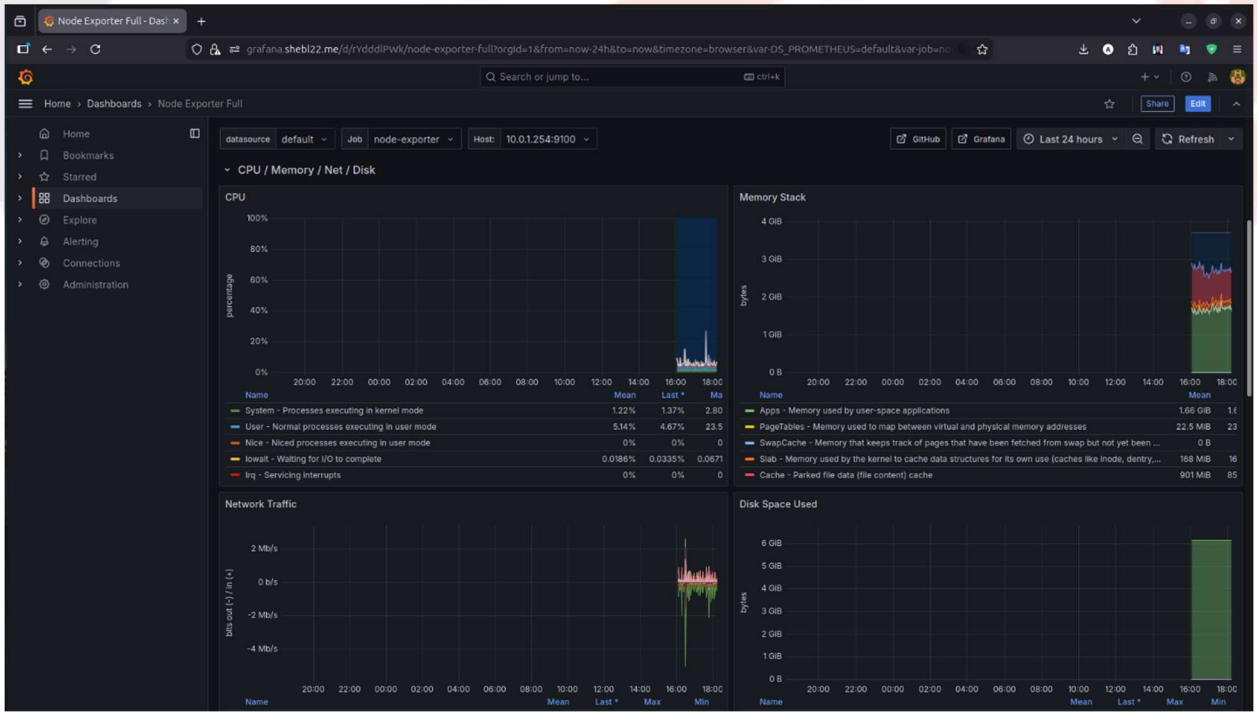


Grafana:

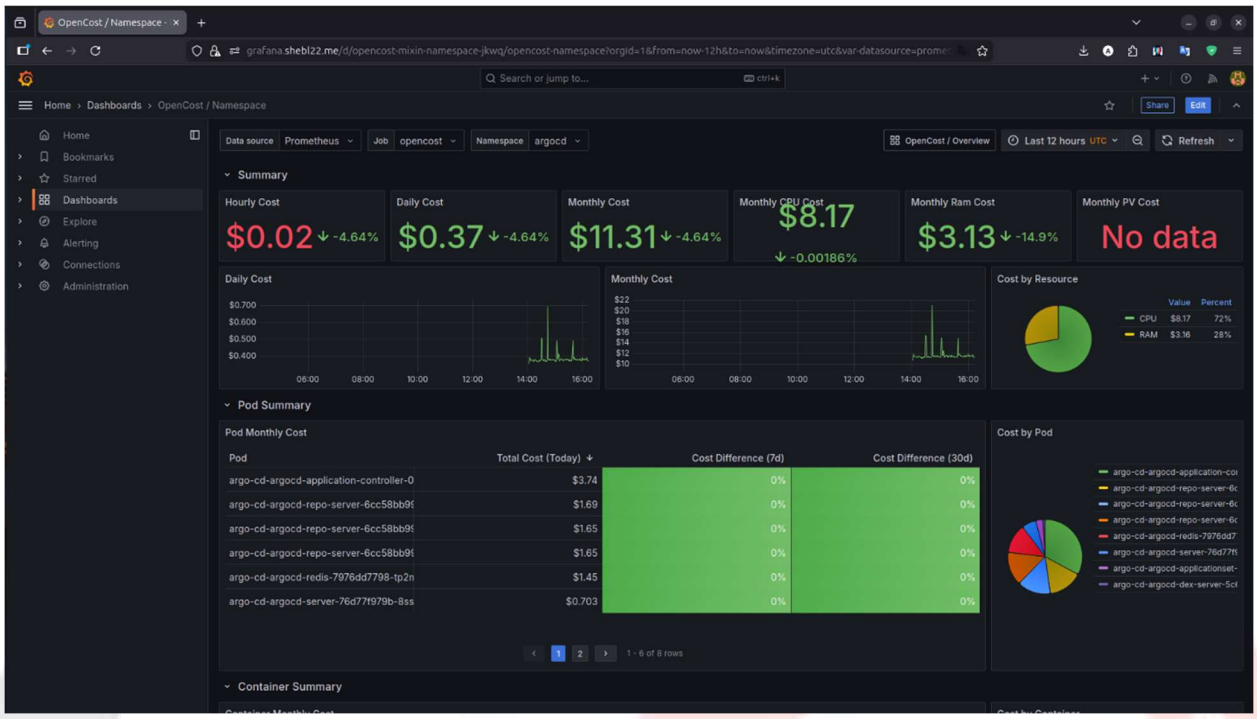


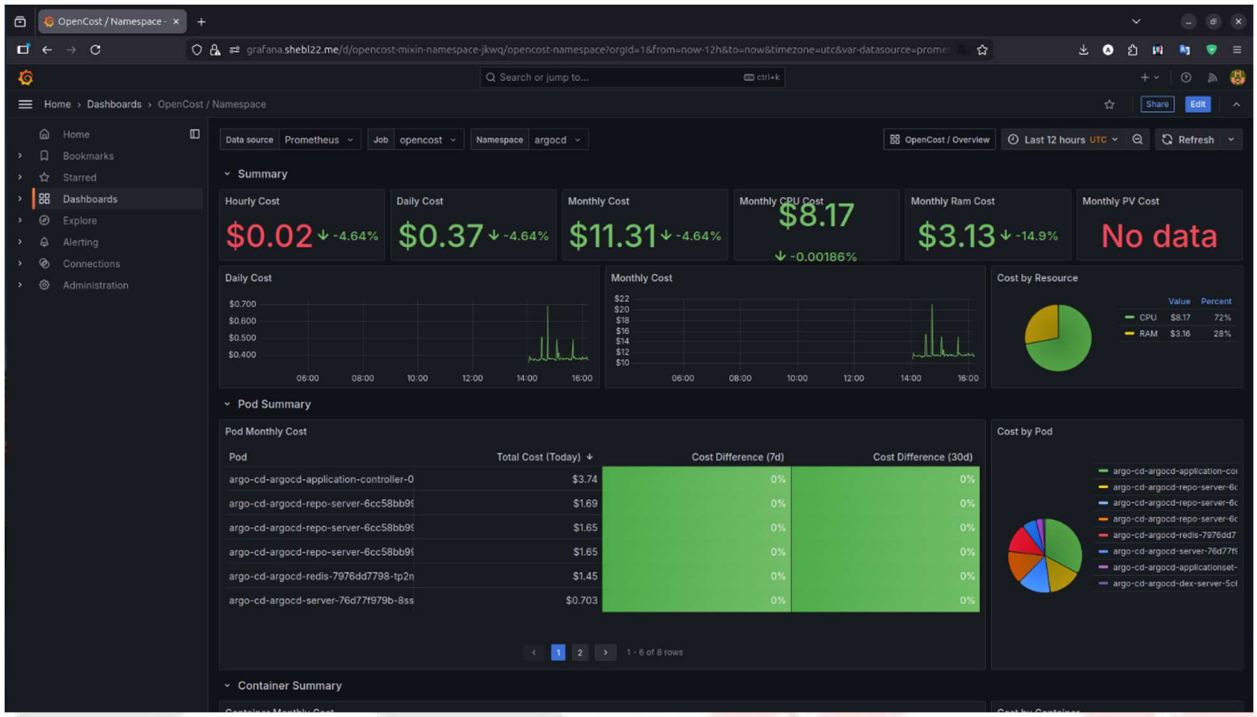

MongoDB Overview - Da
grafana.shebl22.me/d/mongodb-percona-overview/mongodb-overview?orgId=1&from=now-1h&to=now&timezone=browser
Search or jump to... ctrl+k
Home › Dashboards › MongoDB Overview
Share
Edit
Home
Bookmarks
Starred
Dashboards
Explore
Alerting
Connections
Administration
Last 1 hour
Refresh
MongoDB Status
UP
Uptime
7.75 hours
Current Connections
9
Available Connections
52419
Resident Memory
86 MB
Virtual Memory
2.62 GB
Operations Rate
2.5 ops/s
2 ops/s
1.5 ops/s
1 ops/s
0.5 ops/s
0 ops/s
ops/s
17:15 17:20 17:25 17:30 17:35 17:40 17:45 17:50 17:55 18:00 18:05 18:10
Name Mean Last *
command 2.47 ops/s 2.47 ops/s
delete 0.0667 ops/s 0.0741 ops/s
Connections
50 K
40 K
30 K
20 K
10 K
0
17:15 17:20 17:25 17:30 17:35 17:40 17:45 17:50 17:55 18:00 18:05 18:10
Name Mean Last *
current 9 9
available 52.4 K 52.4 K
active 3 3
Memory Usage
2.50 GB
2 GB
1.50 GB
1 GB
500 MB
0 MB
MB
17:15 17:20 17:25 17:30 17:35 17:40 17:45 17:50 17:55 18:00 18:05 18:10
Name Mean Last *
resident 85.9 MB 86 MB
virtual 2.62 GB 2.62 GB
Network I/O
3.50 kB/s
3 kB/s
2.50 kB/s
2 kB/s
1.50 kB/s
1 kB/s
500 B/s
17:15 17:20 17:25 17:30 17:35 17:40 17:45 17:50 17:55 18:00 18:05 18:10
Name Mean Last *
bytes in 404 B/s 404 B/s
bytes out 3.13 kB/s 3.13 kB/s
Asserts Rate
Database Storage
MySQL Overview - Dashb
grafana.shebl22.me/d/MQWgroiiz/mysql-overview?var-interval=$__auto&orgId=1&from=now-12h&to=now&timezone=browser&var-host=10.0.2.90:910
Search or jump to... ctrl+k
Home › Dashboards › MySQL Overview
Share
Edit
Home
Bookmarks
Starred
Dashboards
Explore
Alerting
Connections
Administration
Interval Auto
Host 10.0.2.90:9104
PMM Annotations
Query Analytics
OS
MySQL
MongoDB
HA
Cloud
Insight
Last 12 hours
Refresh 1m
PMM
MySQL Uptime
7.9 hours
Current QPS
0.72
InnoDB Buffer Pool Size
128 MiB
Buffer Pool Size of Total RAM
Connections
MySQL Connections
150
125
100
75
50
25
0
07:00 08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00 17:00 18:00
Name Mean Max Min
Connections 21 21 21
Max Used Connections 21 21 21
MySQL Client Thread Activity
Threads
20
15
10
5
0
07:00 08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00 17:00 18:00
Name Mean Last * Max Min
Peak Threads Connected 21 21 21 21
Peak Threads Running 2 2 2 2
Table Locks
MySQL Questions
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
07:00 08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00 17:00 18:00
Name Mean Max Min
Questions 0.480 0.752 0.244
MySQL Thread Cache
8
6
4
2
0
07:00 08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00 17:00 18:00
Name Mean Max Min
Thread Cache Size 9 9 9
Threads Cached 0 0 0

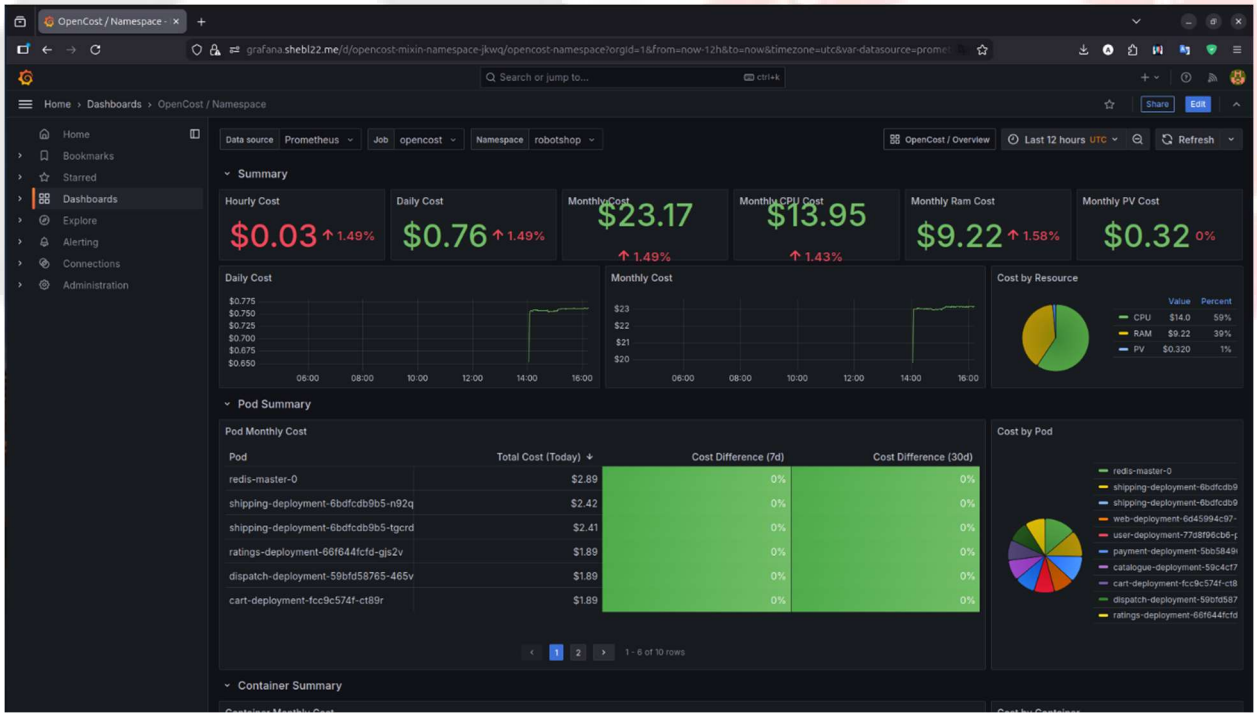

Node Exporter Full - Dash
grafana.shebl22.me/d/rYdddlPWk/node-exporter-full?orgId=1&from=now-24h&to=now&timezone=browser&var-DS_PROMETHEUS=default&var-job=no
Search or jump to...
Home › Dashboards › Node Exporter Full
Share
Edit
Home
Bookmarks
Starred
Dashboards
Explore
Alerting
Connections
Administration
datasource default
Job node-exporter
Host: 10.0.1.254:9100
GitHub
Grafana
Last 24 hours
Refresh
Quick CPU / Mem / Disk
CPU Busy
6.26%
Sys Load (5m...
6.50%
Sys Load (15...
8%
RAM Used
54%
SWAP Used
N/A
Root FS Used
30.8%
CPU...
2
Uptime
10.8 hours
Root...
20 GiB
RAM...
4 GiB
SWA...
0 B
Basic CPU / Mem / Net / Disk
CPU Basic
Busy System
Busy User
Busy Iowait
Busy IRQs
Busy Other
Idle
Memory Basic
RAM Total
RAM Used
RAM Cache + Buffer
RAM Free
SWAP Used
Network Traffic Basic
Disk Space Used Basic
CPU / Memory / Net / Disk
(8 panels)

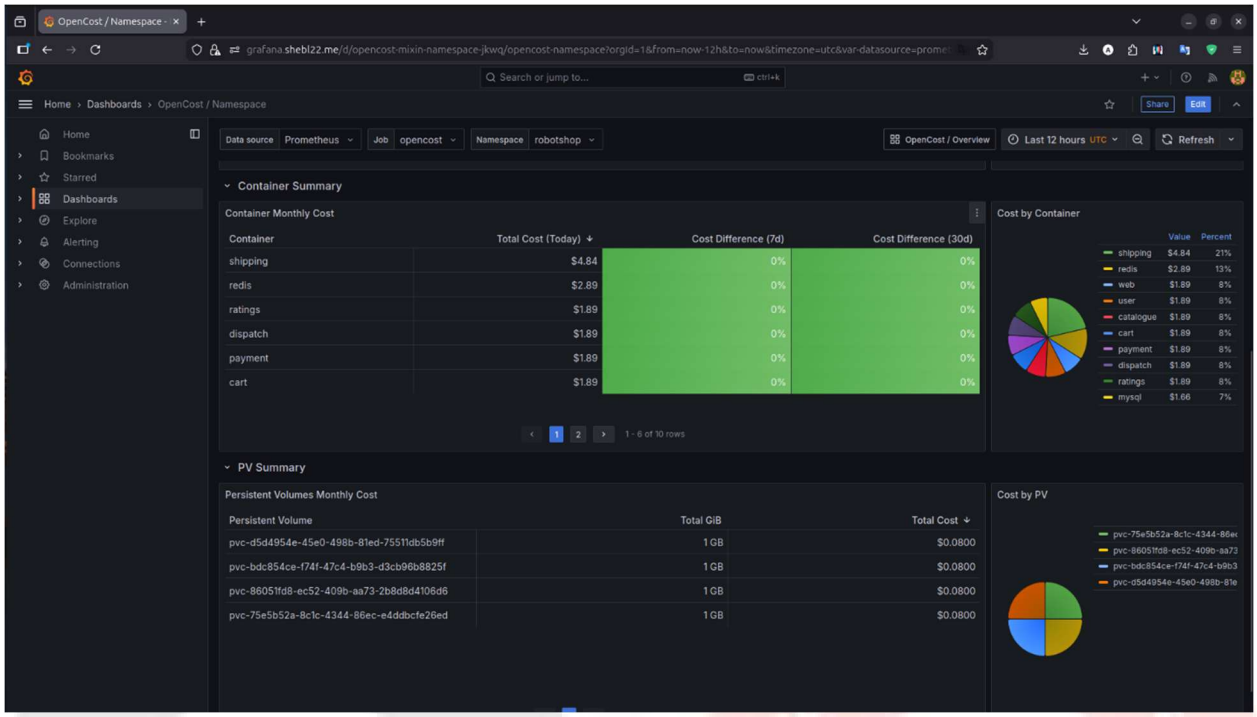

Node Exporter Full - Dash
grafana.shebl22.me/d/rYdddlPWk/node-exporter-full?orgId=1&from=now-24h&to=now&timezone=browser&var-DS_PROMETHEUS=default&var-job=no
Search or jump to...
Home › Dashboards › Node Exporter Full
Share
Edit
Home
Bookmarks
Starred
Dashboards
Explore
Alerting
Connections
Administration
datasource default
Job node-exporter
Host: 10.0.1.254:9100
GitHub
Grafana
Last 24 hours
Refresh
CPU / Memory / Net / Disk
CPU
percentage
Name
Mean
Last *
System - Processes executing in kernel mode
1.22%
1.37%
User - Normal processes executing in user mode
5.14%
4.67%
Nice - Niced processes executing in user mode
0%
0%
Iowait - Waiting for I/O to complete
0.0186%
0.0335%
Irq - Servicing interrupts
0%
0%
Memory Stack
bytes
Apps - Memory used by user-space applications
1.66 GiB
PageTables - Memory used to map between virtual and physical memory addresses
22.5 MiB
SwapCache - Memory that keeps track of pages that have been fetched from swap but not yet been ...
0 B
Slab - Memory used by the kernel to cache data structures for its own use (caches like inode, dentry,...
168 MiB
Cache - Parked file data (file content) cache
901 MiB
Network Traffic
bits out (-) / in (+)
Disk Space Used
Mean
Max
Min

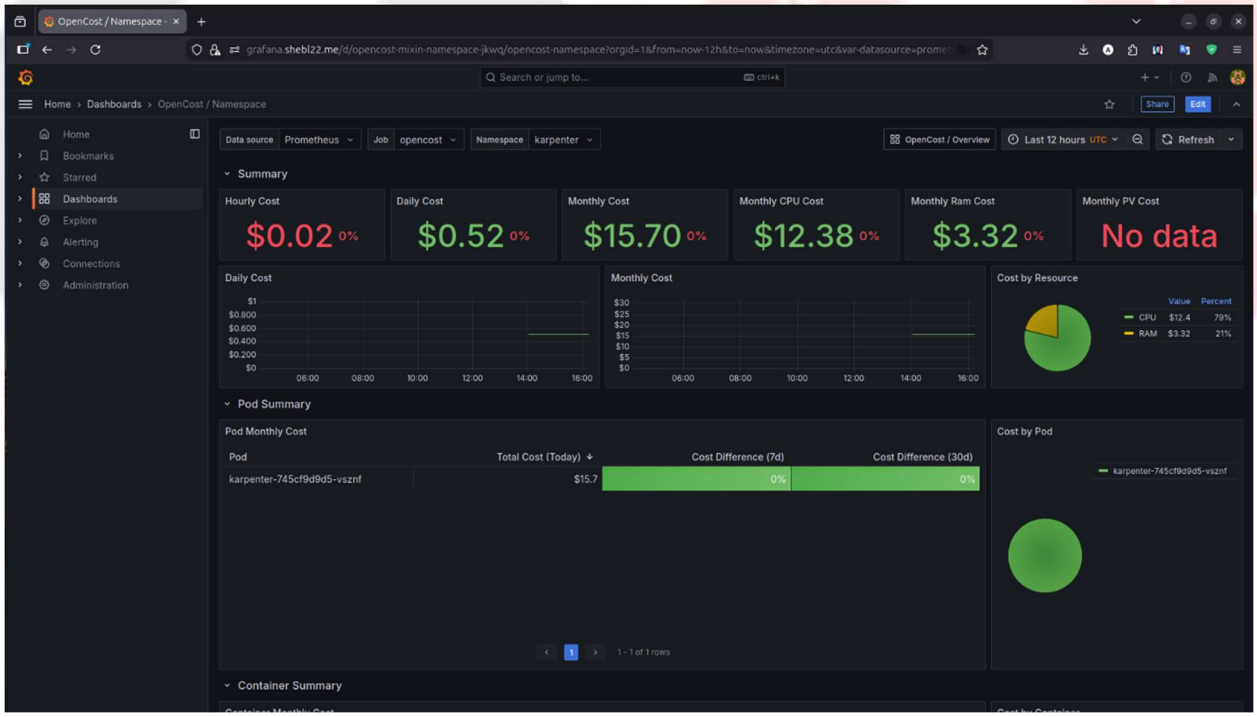
Opencost dashboards:

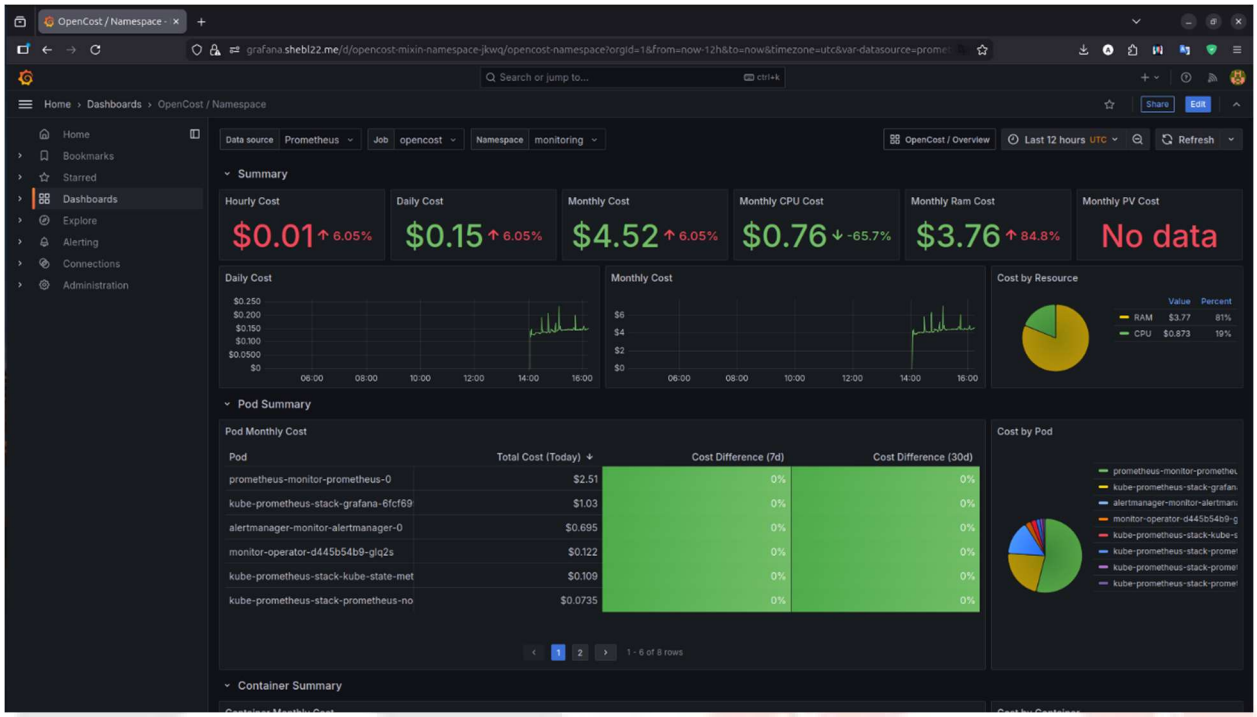


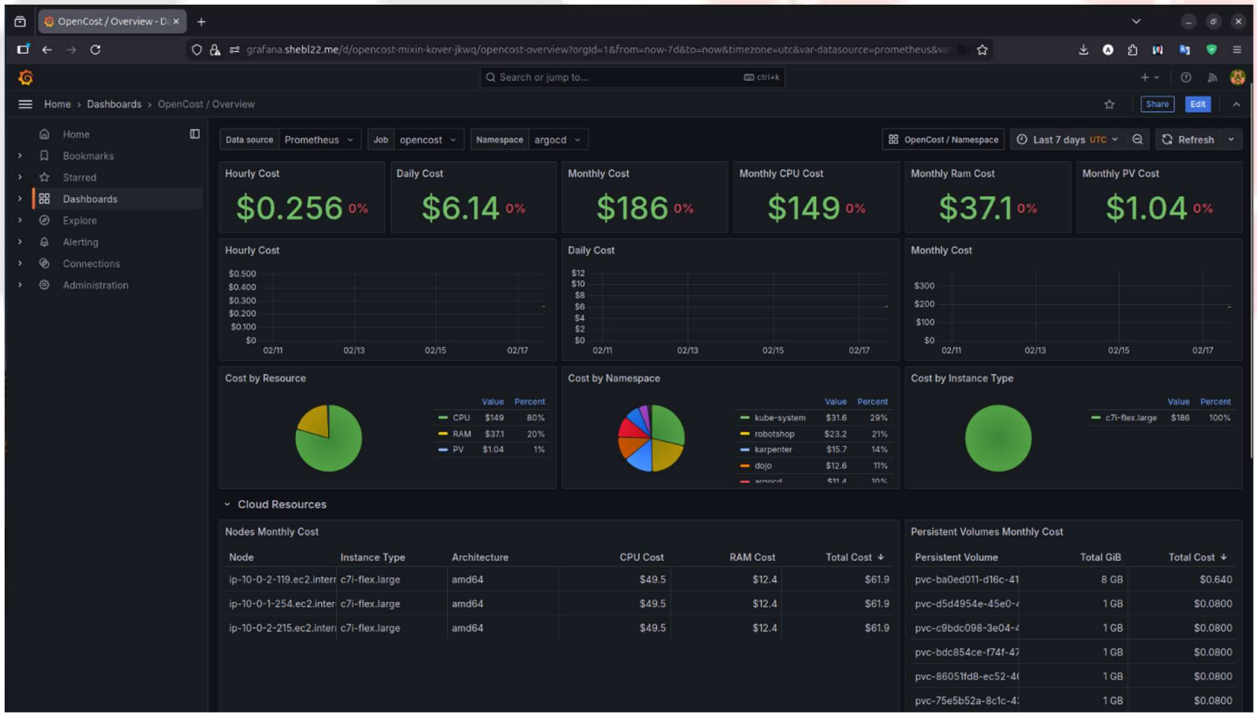

OpenCost / Namespace - argocd
Home › Dashboards › OpenCost / Namespace
Data source Prometheus
Job opencost
Namespace argocd
Summary
Hourly Cost $0.02 -4.64%
Daily Cost $0.37 -4.64%
Monthly Cost $11.31 -4.64%
Monthly CPU Cost $8.17 -0.00186%
Monthly Ram Cost $3.13 -14.9%
Monthly PV Cost No data
Daily Cost
Monthly Cost
Cost by Resource
CPU $8.17 72%
RAM $3.16 28%
Pod Summary
Pod Monthly Cost
Pod | Total Cost (Today) | Cost Difference (7d) | Cost Difference (30d)
argo-cd-argocd-application-controller-0 $3.74
argo-cd-argocd-repo-server-6cc58bb9f $1.69
argo-cd-argocd-repo-server-6cc58bb9f $1.65
argo-cd-argocd-repo-server-6cc58bb9f $1.65
argo-cd-argocd-redis-7976dd7798-tp2n $1.45
argo-cd-argocd-server-76d77f979b-8ss $0.703
Cost by Pod
Container Summary

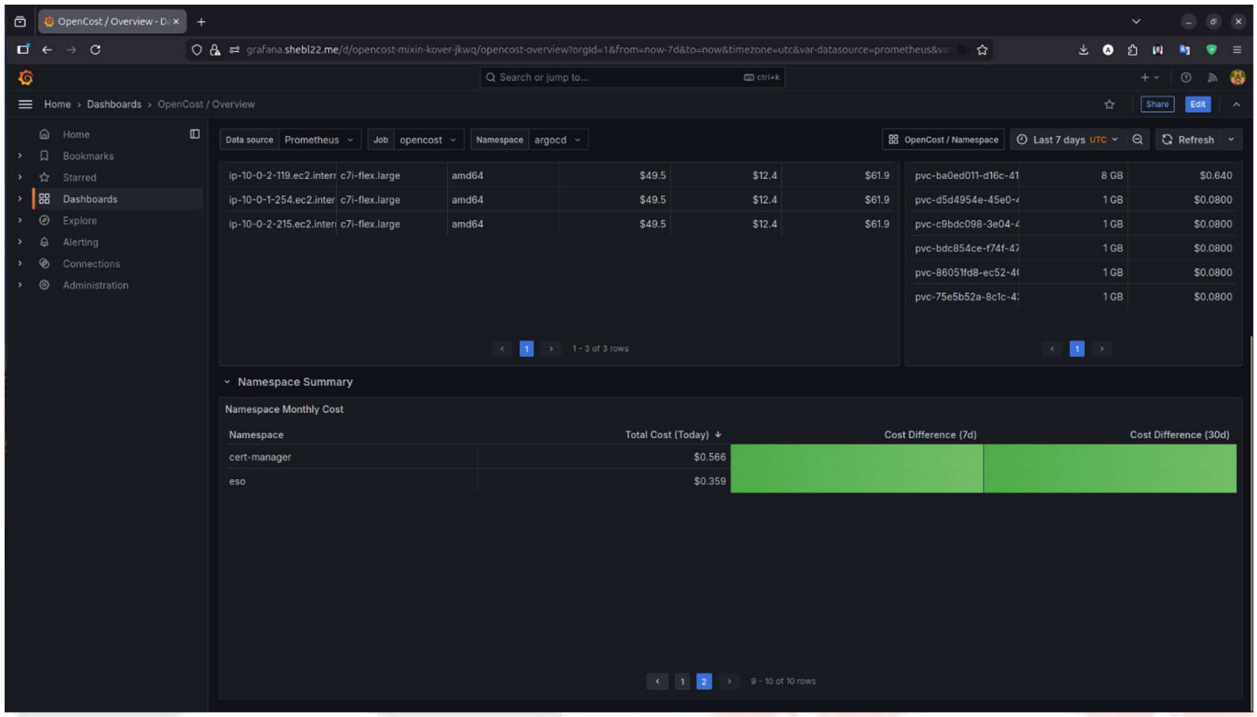

OpenCost / Namespace - robotshop
Home › Dashboards › OpenCost / Namespace
Data source Prometheus
Job opencost
Namespace robotshop
Summary
Hourly Cost $0.03 1.49%
Daily Cost $0.76 1.49%
Monthly Cost $23.17 1.49%
Monthly CPU Cost $13.95 1.43%
Monthly Ram Cost $9.22 1.58%
Monthly PV Cost $0.32 0%
Daily Cost
Monthly Cost
Cost by Resource
CPU $14.0 59%
RAM $9.22 39%
PV $0.320 1%
Pod Summary
Pod Monthly Cost
Pod | Total Cost (Today) | Cost Difference (7d) | Cost Difference (30d)
redis-master-0 $2.89
shipping-deployment-6bdfcdb9b5-n92q $2.42
shipping-deployment-6bdfcdb9b5-tgcrd $2.41
ratings-deployment-66f644fcfd-gjs2v $1.89
dispatch-deployment-59bfd58765-465v $1.89
cart-deployment-fcc9c574f-ct89r $1.89
Cost by Pod
Container Summary


OpenCost / Namespace
Home › Dashboards › OpenCost / Namespace
Data source Prometheus
Job opencost
Namespace robotshop
Container Summary
Container Monthly Cost
Container
Total Cost (Today)
Cost Difference (7d)
Cost Difference (30d)
shipping $4.84 0% 0%
redis $2.89 0% 0%
ratings $1.89 0% 0%
dispatch $1.89 0% 0%
payment $1.89 0% 0%
cart $1.89 0% 0%
Cost by Container
PV Summary
Persistent Volumes Monthly Cost
Persistent Volume
Total GiB
Total Cost
pvc-d5d4954e-45e0-498b-81ed-75511db5b9ff 1 GB $0.0800
pvc-bdc854ce-f74f-47c4-b9b3-d3cb96b8825f 1 GB $0.0800
pvc-86051fd8-ec52-409b-aa73-2b8d8d4106d6 1 GB $0.0800
pvc-75e5b52a-8c1c-4344-86ec-e4ddbcfe26ed 1 GB $0.0800
Cost by PV


OpenCost / Namespace
Home › Dashboards › OpenCost / Namespace
Data source Prometheus
Job opencost
Namespace karpenter
Summary
Hourly Cost $0.02
Daily Cost $0.52
Monthly Cost $15.70
Monthly CPU Cost $12.38
Monthly Ram Cost $3.32
Monthly PV Cost No data
Daily Cost
Monthly Cost
Cost by Resource
Pod Summary
Pod Monthly Cost
Pod
Total Cost (Today)
Cost Difference (7d)
Cost Difference (30d)
karpenter-745cf9d9d5-vsznf $15.7 0% 0%
Cost by Pod
Container Summary


OpenCost / Namespace
Home › Dashboards › OpenCost / Namespace
Data source Prometheus
Job opencost
Namespace monitoring
Summary
Hourly Cost $0.01 ↑ 6.05%
Daily Cost $0.15 ↑ 6.05%
Monthly Cost $4.52 ↑ 6.05%
Monthly CPU Cost $0.76 ↓ -65.7%
Monthly Ram Cost $3.76 ↑ 84.8%
Monthly PV Cost No data
Daily Cost
Monthly Cost
Cost by Resource
RAM $3.77 81%
CPU $0.873 19%
Pod Summary
Pod Monthly Cost
Pod
Total Cost (Today)
Cost Difference (7d)
Cost Difference (30d)
prometheus-monitor-prometheus-0 $2.51
kube-prometheus-stack-grafana-6fcf69 $1.03
alertmanager-monitor-alertmanager-0 $0.695
monitor-operator-d445b54b9-glq2s $0.122
kube-prometheus-stack-kube-state-met $0.109
kube-prometheus-stack-prometheus-no $0.0735
Cost by Pod
Container Summary


OpenCost / Overview
Home › Dashboards › OpenCost / Overview
Data source Prometheus
Job opencost
Namespace argocd
Hourly Cost $0.256 0%
Daily Cost $6.14 0%
Monthly Cost $186 0%
Monthly CPU Cost $149 0%
Monthly Ram Cost $37.1 0%
Monthly PV Cost $1.04 0%
Cost by Resource
CPU $149 80%
RAM $37.1 20%
PV $1.04 1%
Cost by Namespace
kube-system $31.6 29%
robotshop $23.2 21%
karpenter $15.7 14%
dojo $12.6 11%
Cost by Instance Type
c7i-flex.large $186 100%
Cloud Resources
Nodes Monthly Cost
Node
Instance Type
Architecture
CPU Cost
RAM Cost
Total Cost
ip-10-0-2-119.ec2.intern c7i-flex.large amd64 $49.5 $12.4 $61.9
ip-10-0-1-254.ec2.intern c7i-flex.large amd64 $49.5 $12.4 $61.9
ip-10-0-2-215.ec2.intern c7i-flex.large amd64 $49.5 $12.4 $61.9
Persistent Volumes Monthly Cost
Persistent Volume
Total GiB
Total Cost
pvc-ba0ed011-d16c-41 8 GB $0.640
1 GB $0.0800
1 GB $0.0800
1 GB $0.0800
1 GB $0.0800
1 GB $0.0800



Platform-Level Integration Summary

Tool	Category	Role
Karpenter	Node Autoscaling	Dynamic infrastructure scaling
DefectDojo	Security	Vulnerability aggregation
Kyverno	Policy	Cluster security enforcement
OpenCost	FinOps	Cost visibility
Goldilocks	Optimization	Resource right-sizing
Prometheus Stack	Observability	Metrics & alerting backbone

Cloud Infrastructure Layer Terraform + AWS

All infrastructure is provisioned using Terraform with official AWS modules.

Our infrastructure:

```
terraform/
├── main.tf           # EKS, VPC, Karpenter, Addons, IAM, Route53
├── variables.tf      # Cluster config (region, domain, EKS version)
├── outputs.tf        # Route53 nameservers
├── providers.tf      # AWS + Helm + Kubectl providers
└── modules/
    ├── addons/       # ArgoCD + ArgoCD Apps (deploys all charts)
    ├── karpenter/    # Karpenter Helm + NodePool + EC2NodeClass
    ├── ssm/          # AWS SSM Parameter Store (secrets)
    ├── eso/          # External Secrets Operator IAM
    ├── edns/         # External DNS IAM
    └── opencost/     # S3 + CUR + Glue + Athena for cost data
```

Layer 1 Infrastructure Layer

(Terraform + AWS Foundation)

This layer provisions and configures all cloud infrastructure using Infrastructure as Code.

It is fully implemented using Terraform with official AWS modules.

The Kubernetes cluster runs on Amazon Elastic Kubernetes Service.

Networking (VPC Layer)

Provisioned using the official VPC module.

Components:

- 2 Availability Zones
- Public subnets (Load Balancers)
- Private subnets (Worker nodes)
- NAT Gateway

- Route tables

Kubernetes-Specific Tags:

- kubernetes.io/role/elb
- kubernetes.io/role/internal-elb
- karpenter.sh/discovery

These tags allow:

- LoadBalancer services to work correctly
- Karpenter to discover subnets automatically

EKS Cluster

Provisioned via Terraform EKS module.

Configuration:

- Private worker subnets
- Public API endpoint access
- Authentication mode: API
- Managed node group (system workloads)

System Node Group:

- Dedicated for core controllers
- Tainted: workload-type=system
- Hosts:
    - CoreDNS
    - metrics-server
    - Karpenter controller
    - AWS controllers

This prevents business workloads from running on system nodes.

Karpenter Dynamic Node Provisioning

Node autoscaling is handled by
Karpenter

Terraform Responsibilities:

- Create IAM role for nodes
- Create SQS interruption queue
- Attach required policies
- Enable Pod Identity for controller

Kubernetes Responsibilities:

- Karpenter controller runs in cluster
- Launches EC2 instances dynamically
- Prefers Spot instances
- Consolidates unused capacity

Result:
Cluster scales at node level automatically and cost-efficiently.

Pod Identity (IAM Without IRSA)

Instead of IRSA, the platform uses EKS Pod Identity.

Terraform creates IAM roles and binds them to service accounts for:

- ExternalDNS
- External Secrets Operator
- cert-manager

- EBS CSI driver

This ensures:

- No static AWS credentials
- Least privilege access
- Fine-grained AWS permissions per controller

Storage (EBS CSI Driver)

EBS CSI driver:

- Uses Pod Identity
- Dynamically provisions volumes
- Required for:
  - Databases
  - DefectDojo
  - Prometheus
  - Stateful services

Route53 DNS Zone

Terraform creates:

- Hosted zone
- DNS records managed dynamically

ExternalDNS uses this to automatically create domain records.

SSM Parameter Store

Terraform provisions secure parameters in AWS SSM.

Used for:

- Application secrets
- OpenCost cloud integration
- Database credentials

Secrets never exist in Git.

Infrastructure Layer Summary

Terraform provisions:

- Networking
- Kubernetes control plane
- IAM roles
- Pod Identity bindings
- DNS zone
- Secure secret storage

This layer prepares AWS so Kubernetes controllers can operate securely.

Layer 2 — GitOps & Application Layer

(ArgoCD + Helm)

This layer deploys and manages everything inside Kubernetes.

It is powered by
Argo CD

GitOps Model

Deployment flow:

Git → ArgoCD → Kubernetes

ArgoCD continuously:

- Pulls manifests from Git
- Applies them
- Detects drift
- Self-heals automatically

No manual kubectl or helm usage.

App-of-Apps Pattern

One root application manages all platform applications.

Each component is defined as an ArgoCD Application:

- cert-manager
- External Secrets
- ExternalDNS
- Traefik

- Prometheus stack
- OpenCost
- Goldilocks
- Kyverno
- DefectDojo
- Robot Shop application

Deletion uses cascade finalizers for clean teardown.

Sync Waves (Ordered Deployment)

To ensure proper dependency ordering:

Wave	Component
-5	cert-manager
-4	ClusterIssuer
-3	ESO, ExternalDNS, Traefik, Prometheus
-2	Exporters, OpenCost, Goldilocks, Kyverno
-1	Kyverno policies
0	Robot Shop app

This guarantees:

- CRDs exist before CRs
- TLS before ingress
- Policies before workloads

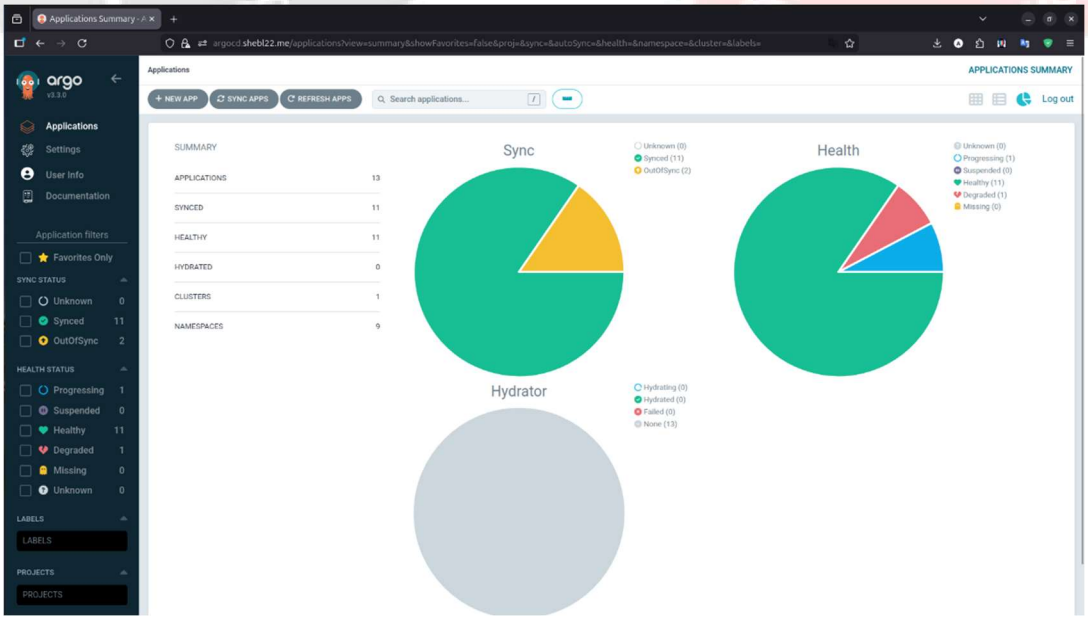
Core Platform Controllers

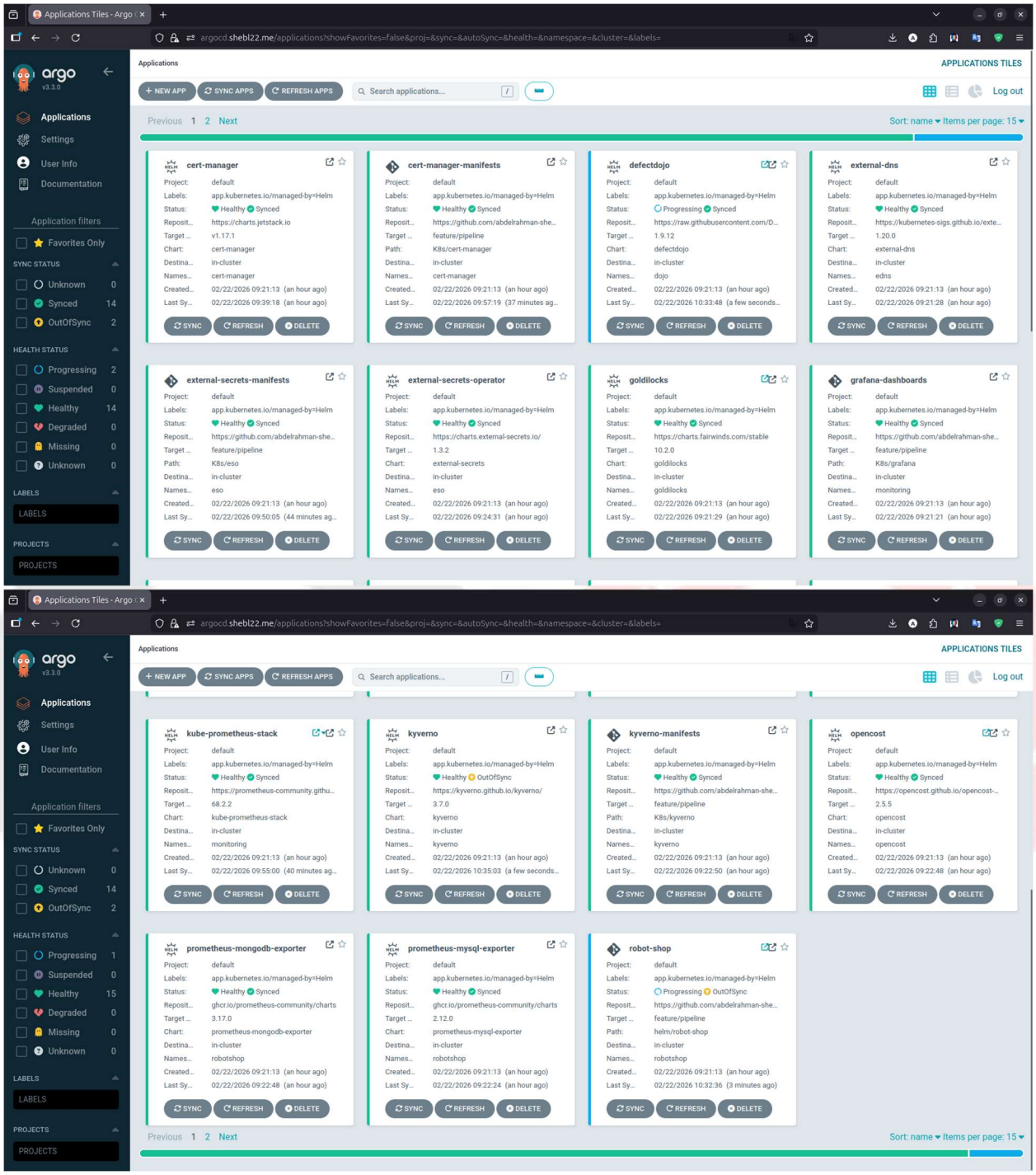
- ExternalDNS
- External Secrets Operator
- cert-manager
- Kyverno
- OpenCost
- Goldilocks
- Prometheus Stack

  Includes:

  - Prometheus
  - Alertmanager
  - Grafana
  - Node exporters
  - kube-state-metrics
- DefectDojo

Results:




Applications Tiles - Argo
argocd.shebl22.me/applications?showFavorites=false&proj=&sync=&autoSync=&health=&namespace=&cluster=&labels=
argo v3.3.0
Applications
Settings
User Info
Documentation
Application filters
Favorites Only
SYNC STATUS
Unknown 0
Synced 14
OutOfSync 2
HEALTH STATUS
Progressing 2
Suspended 0
Healthy 14
Degraded 0
Missing 0
Unknown 0
LABELS
PROJECTS
Applications
APPLICATIONS TILES
+ NEW APP
SYNC APPS
REFRESH APPS
Search applications...
Log out
Previous 1 2 Next
Sort: name Items per page: 15
cert-manager
cert-manager-manifests
defectdojo
external-dns
external-secrets-manifests
external-secrets-operator
goldilocks
grafana-dashboards
SYNC
REFRESH
DELETE
Applications Tiles - Argo
argocd.shebl22.me/applications?showFavorites=false&proj=&sync=&autoSync=&health=&namespace=&cluster=&labels=
Applications
APPLICATIONS TILES
+ NEW APP
SYNC APPS
REFRESH APPS
Search applications...
Log out
kube-prometheus-stack
kyverno
kyverno-manifests
opencost
prometheus-mongodb-exporter
prometheus-mysql-exporter
robot-shop
Previous 1 2 Next
Sort: name Items per page: 15

Robot Shop Application

Deployed via umbrella Helm chart.

Features:

- HPA per service
- Network policies
- Strict security contexts
- Environment-based values
- Ingress with TLS

Observability & FinOps

Monitoring

- kube-prometheus-stack
- Custom Grafana dashboards

FinOps

OpenCost integrates with:

- AWS CUR
- S3
- Athena

Autoscaling Strategy

Karpenter provides:

- Spot-first provisioning
- On-demand fallback
- NodePool abstraction

Horizontal Pod Autoscalers scale application services based on CPU and memory utilization.

Supply Chain Security

The platform enforces image verification:

- All images signed via Cosign
- Kyverno verifies signatures before pod admission
- Unsigned images are rejected

GitOps Layer Summary

ArgoCD manages:

- Platform controllers
- Security enforcement
- Observability stack
- FinOps stack
- Vulnerability management
- Business workloads

All declaratively defined in Git.

Final Architecture Philosophy

Infrastructure Layer (Terraform)

↓

Cloud Foundation + IAM + Network + Cluster

GitOps Layer (ArgoCD)

↓

Controllers + Security + Monitoring + Applications

Together they create:

- Fully automated provisioning
- Secure workload isolation

- Cost-aware scaling
- Git-driven deployments
- Zero manual operations

Production Characteristics

This platform includes:

- High availability
- Spot cost optimization
- GitOps-based deployment
- Secure secret lifecycle
- Automated cleanup on destroy
- Infrastructure modularity
- Strict workload isolation
- Enterprise-grade security scanning

CI/CD Pipeline Full DevSecOps Automation

Pipeline implemented in GitHub Actions.

Triggered on:

- main
- feature/*

Pipeline Flow

Push → SAST → Secret Scan → Image Build → Image Scan → Sign → DAST → Deploy

Security Layers

Tool	Type	Purpose
Semgrep	SAST	Code vulnerabilities
Gitleaks	Secret detection	Prevent leaked credentials
Docker Scout	SCA	CVE scanning
OWASP ZAP	DAST	Runtime scanning
Cosign	Supply chain	Image signing

Image Supply-Chain Security

- Images built with BuildKit
- Scanned for High/Critical CVEs
- Signed using Cosign (keyless via Sigstore)
- Verified in cluster by Kyverno

Unsigned images are rejected.

GitOps Auto Deployment

Final stage:

1. Update Helm values.yaml image tag
2. Commit with [skip ci]
3. ArgoCD detects change
4. Automatic sync to cluster
5. Zero manual kubectl / helm usage

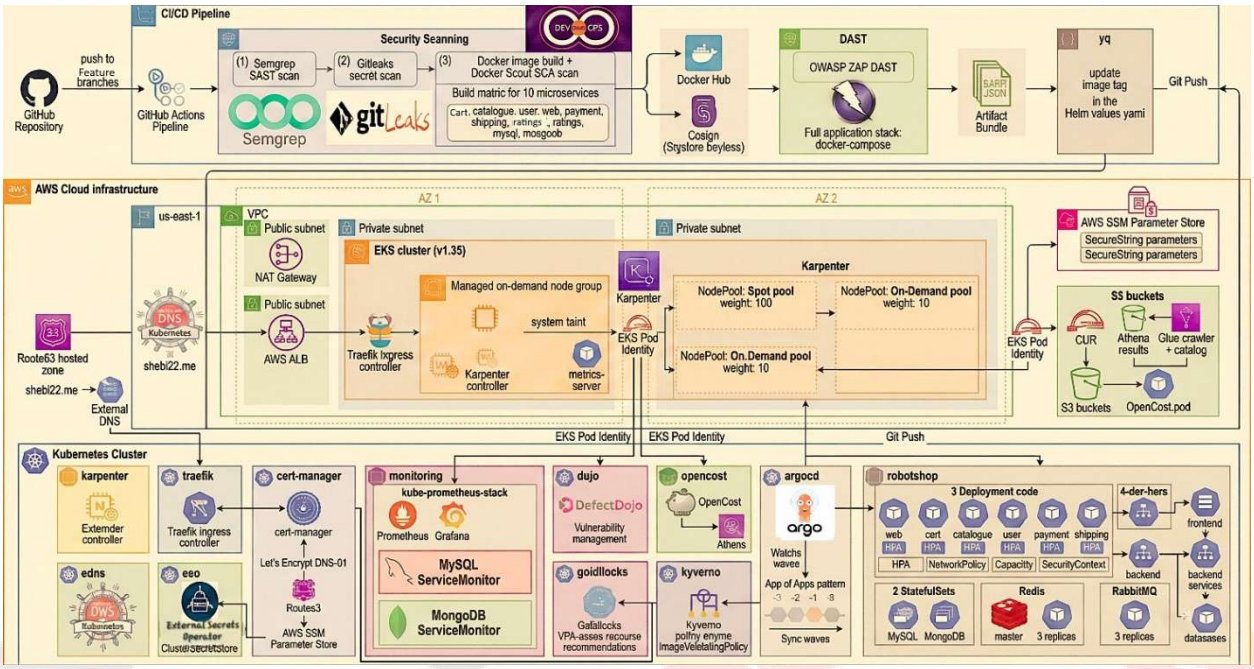
Flow Description

1. Secrets defined in variables.yaml (git-ignored)
2. Terraform stores them in AWS SSM as SecureStrings
3. EKS Pod Identity grants ESO IAM access
4. ESO syncs secrets into Kubernetes
5. Applications consume Kubernetes Secrets only

Benefits:

- No secret leakage
- No secret in Terraform state
- Centralized secret management

Finally the infrastructure of the project:



Final Project Structure

The project is organized into:

- Application services
- Helm umbrella chart
- Kubernetes manifests
- Terraform infrastructure
- CI/CD workflows
- Component documentation

Clear separation between:

- Infrastructure
- Platform services
- Application layer

Conclusion

Robot Shop evolved from a demo microservices app into a full production-grade DevSecOps reference platform.

It demonstrates:

- Secure CI/CD pipelines
- GitOps automation
- Infrastructure as Code
- Cloud-native best practices
- Cost observability
- Kubernetes hardening
- Supply chain integrity

This architecture can serve as a blueprint for real-world enterprise Kubernetes deployments.