

Operadores, Dashboards y Monitorización en Kubernetes



KEEPCODING

Tech School

■ Dashboards y Monitorización

- En esta sección:
 - Dashboards (UI) para gestión del cluster
 - Operadores en Kubernetes
 - Sistemas de monitorización
 - Kubernetes Observability
 - Prometheus y Grafana
 - Elastic Stack
 - Dashboards para el manejo del cluster.



Dashboards / UI



■ Dashboards para manejar Kubernetes

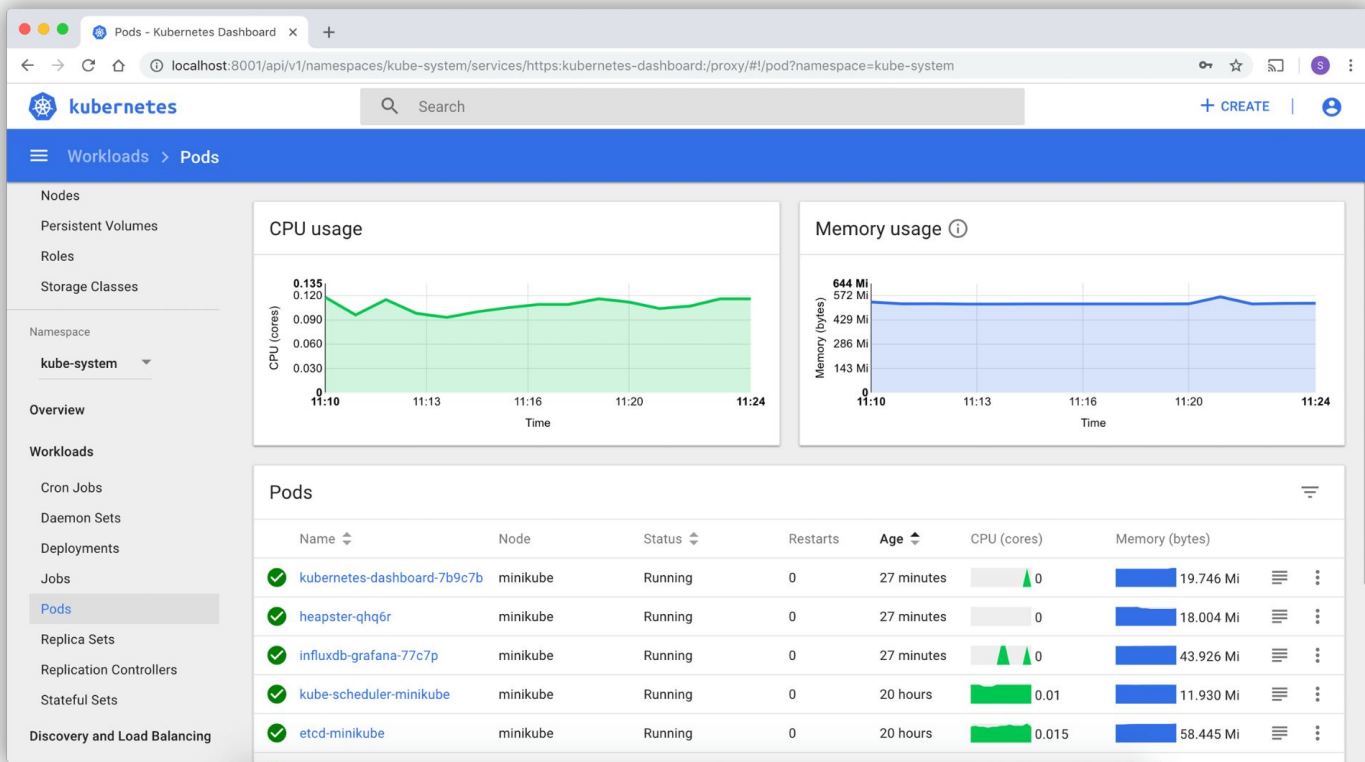
- [Kubernetes Dashboard](#)
- Lens → <https://k8slens.dev/>
- k9s → <https://k9scli.io/>
- Kuberetic → <https://www.kubernetec.com/>

Kubernetes Observability

- Prometheus & Grafana → Para métricas / monitorización.
- Elastic Stack → Para observability (métricas, logs, APM, SIEM, Machine Learning, etc).



Kubernetes Dashboard



The screenshot displays the Kubernetes Lens application interface. The left sidebar shows the navigation menu with categories like Cluster, Nodes, Workloads, Configuration, Network, Storage, Namespaces, Events, Apps, Access Control, and Custom Resources. The 'Workloads' section is expanded, showing a list of pods. The main panel is titled 'Pod: nginx-deployment-69c44dfb78-5p8vj' and shows detailed information about the selected pod, including its status (Running), node (minikube), and various labels. A terminal window at the bottom shows the command 'root@nginx-deployment-69c44dfb78-5p8vj: /# |'.

Name	Namespace	Containers
nginx-deployment-69c44dfb78-5p8vj	default	1
nginx-deployment-69c44dfb78-9x6pv	default	1
otro-deployment-5d59d67564-676qm	default	1
otro-deployment-5d59d67564-v6mm9	default	1

Pod: nginx-deployment-69c44dfb78-5p8vj

Created: 174m 20s ago (2022-02-09T18:34:19+01:00)

Name: nginx-deployment-69c44dfb78-5p8vj

Namespace: default

Labels: app=nginx, pod-template-hash=69c44dfb78

Controlled By: ReplicaSet nginx-deployment-69c44dfb78

Status: **Running**

Node: minikube

Pod IP: 172.17.0.10

Pod IPs: 172.17.0.10

Priority Class: -

QoS Class: BestEffort

Conditions: Initialized, Ready, ContainersReady, PodScheduled

Tolerations: 2

Containers

nginx

Status: **running, ready**

Image: nginx:1.9.1

Ports: 80/TCP, Forward...

Environment: -

Mounts: /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-jxlzr (ro)

Volumes: -





Kubernetic

The screenshot shows the Kubernetic web interface. On the left is a sidebar with navigation links: Infrastructure (Nodes, Quotas, Limits, Events), Workloads (Pods, Deployments, StatefulSets, DaemonSets, Jobs, CronJobs), Configuration (Config, Secrets, Autoscalers), and Networking (Services). The main area is titled 'Deployments' and shows a summary for CPU (0m) and Memory (18Mi) usage. Below this is a search bar and a table of deployments. The table has columns: NAME, CPU, MEMORY, READY, UP-TO-DATE, AVAILABLE, and AGE. Three deployments are listed: gateway, payment, and sms. The gateway deployment is highlighted with a mouse cursor.

NAME	CPU	MEMORY	READY	UP-TO-DATE	AVAILABLE	AGE
gateway	0m	3Mi	2/2	2	2	50m
payment	0m	8Mi	3/3	3	3	50m
sms	0m	9Mi	2/2	2	2	50m



Operadores en Kubernetes



■ Operadores en Kubernetes

- ¿Qué es un Operador?
 - Es un controlador específico de aplicaciones que amplía las funciones de Kubernetes para poder crear, configurar, y gestionar las instancias de esas aplicaciones.
 - Utilizarán recursos básicos de Kubernetes pero tienen el conocimiento específico de la aplicación para automatizar todo el ciclo de vida del software que gestiona.



■ Operadores en Kubernetes

- ¿Por qué son necesarios los operadores?
 - Para manejar aplicaciones con estado que pueden requerir conocimientos específicos que Kubernetes no posee.
- Componentes de un Operador
 - Definición de objetos (CRDs / Custom Resource Definitions)
 - Metadatos que se instalan en el cluster
 - Controlador
 - Software que corre en el cluster (como pod)



■ Operadores en Kubernetes

- ¿Cual es el valor añadido de usar operadores?
 - El operador cuidará de las aplicaciones mejor de lo que puede hacerlo Kubernetes, ya que conoce cómo han de realizarse las operaciones y el mantenimiento.
 - Un Deployment cuida de sus pods, y un StatefulSet de los suyos pero con unas limitaciones que pueden ser negativas para algunas aplicaciones como bases de datos.



Sistemas de Monitorización



KEEPCODING

Tech School

■ Sistemas de Monitorización

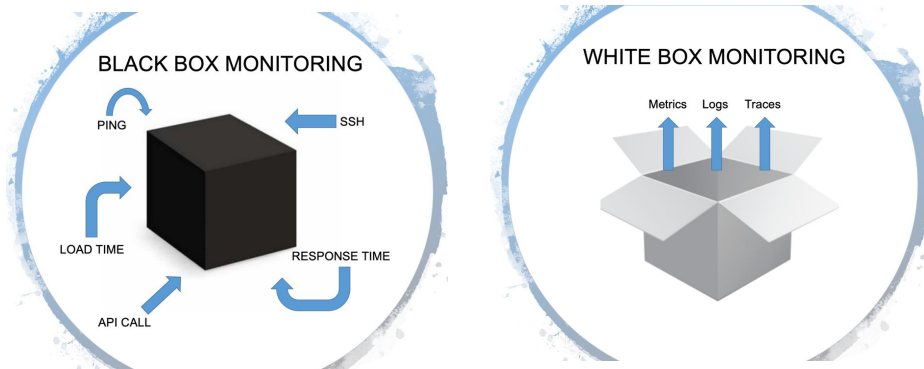
- Monitorización caja blanca y caja negra
- Monitorización mediante HTTP Checks
- Monitorización basada en logs
- Prometheus
- Instrumentar una aplicación con Prometheus



Sistemas de monitorización

“Asegurar que tanto aplicaciones como sistemas funcionan como deberían”

- Sistemas de monitorización manuales o automáticos
- Sistemas de caja negra o caja blanca



■ Sistemas de caja negra

- Desconocemos cómo funciona el sistema internamente.
- Utilizamos sistemas externos para monitorizar y comprobar que funciona correctamente.
- Establecemos métricas y kpis nosotros de forma externa, no tienen por qué ser las correctas.
- Sistemas de monitorización caja negra conocidos:
 - Nagios
 - Sensu
 - Heartbeat (Elastic Stack) → Uptime application
 - Pingdom (HTTP Checks)
 - Stackdriver (HTTP Checks)
 - Statuscake (HTTP Checks)



■ HTTP / TCP checks

- Exponer un endpoint (normal o especial) que permita monitorizar:
 - El estado del sistema.
 - Tiempo de respuesta, contenido de respuesta, etc.
- Existen sistemas como servicio o alojados por nosotros mismos.



■ Sistemas de caja blanca

- Mediante la monitorización de caja blanca son los propios sistemas y aplicaciones los que exponen sus propias métricas que nosotros podemos leer y actuar frente a unos resultados u otros.
- Elementos principales:
 - Métricas (de todo tipo, dependiendo del sistema monitorizado).
 - Logs
 - Application Performance Metrics (APM)



■ Sistemas basados en métricas

- Sistemas de obtención de Métricas:
 - Requieren un sistema específico y adaptar nuestras aplicaciones.
 - Muy ligeros
 - Sistemas baratos
 - Fácil de correlar con otros sistemas.
 - Ejemplos
 - Prometheus
 - Elastic Stack



■ Sistemas basados en logs

- Los sistemas basados en logs son sencillos de implementar
- Permiten expresar mucha información de manera muy sencilla.
- Costoso computacionalmente y a nivel de storage (requieren buen dimensionamiento)
- Debemos expresar los logs en algún formato estructurado (JSON) para facilitar la ingesta, procesamiento y su explotación posterior.
- Debemos implementar algún sistema de recolección de logs (Filebeat / Elastic Stack, Fluentd) y de ingesta centralizada (Elastic Stack, Stackdriver, Cloudwatch).
- Los sistemas as a service tienen buena escalabilidad pero son caros.



■ Visualización

- Tanto para la recolección de logs como para las métricas o APM necesitamos algún sistema para visualizar y analizar los datos.
- Existen diversos sistemas para visualizar los datos (métricas, logs, datos de APM, ...).
 - Kibana (Elastic Stack)
 - Grafana (soporta multitud de backends, como prometheus)
 - Stackdriver
 - Splunk



Prometheus y Grafana



■ Prometheus

- Existen sistemas especialmente diseñados para extraer y almacenar métricas que posteriormente pueden ser analizadas y visualizadas.
- Uno de los sistemas más conocidos y que es nativo en Kubernetes es **Prometheus**.
 - Inspirado en el sistema de monitorización Borgmon de Google
 - Comenzó en 2012 por ex-googlers en Soundcloud
 - Escrito en Go
 - Hecho público en 2015
 - Proyecto graduado de la CNCF

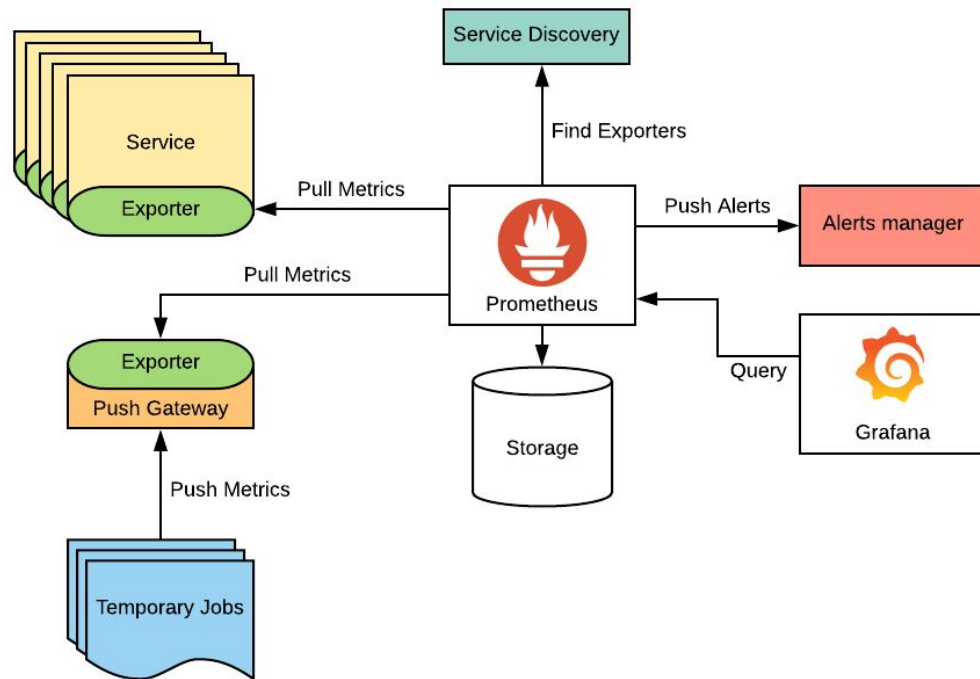


■ ¿Por qué Prometheus?

- Monitorización de caja blanca
- Basado en métricas tanto de sistemas como de negocio
- Robusto y eficaz, alta disponibilidad.
- Eficiente, un solo servidor puede manejar millones de métricas
- Service discovery integrado
- Se integra fácilmente



Arquitectura de Prometheus



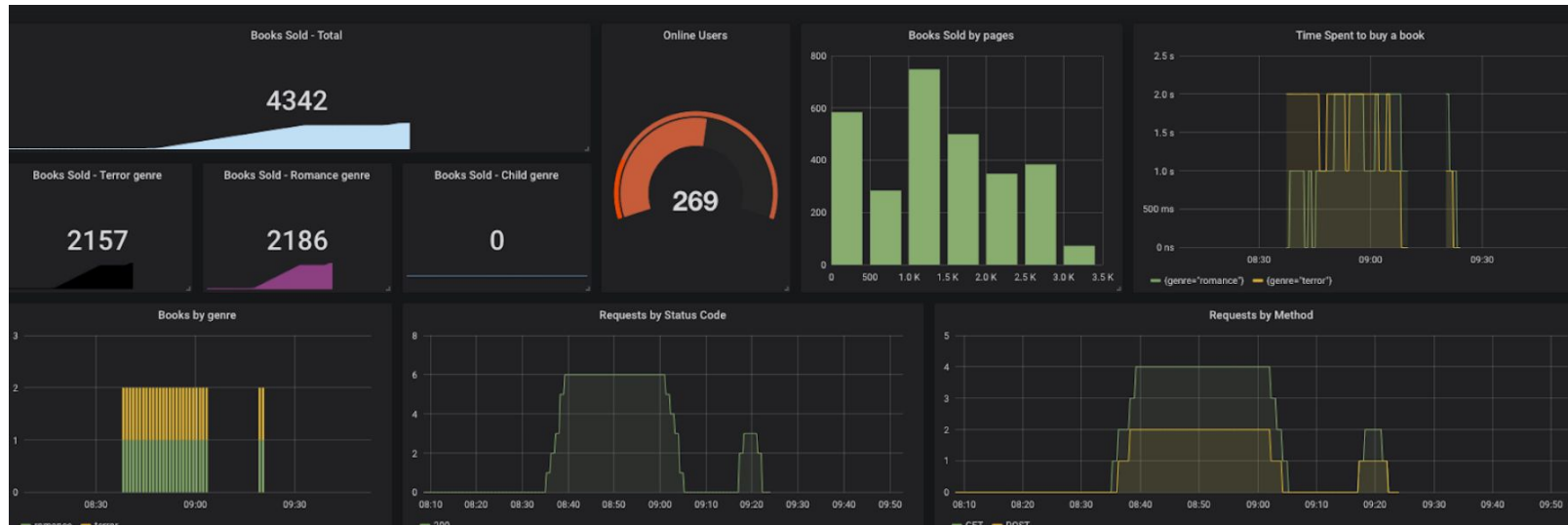
Lo que lee Prometheus Server

```
# HELP books sold Number of books sold
# TYPE books sold counter
books sold{genre="terror"} 199.0
books sold{genre="romance"} 70.0
# HELP uptime uptime
# TYPE uptime gauge
uptime 4.2769899E7
# HELP systemload_average systemload_average
# TYPE systemload_average gauge
systemload_average 0.55
# HELP heap_committed heap_committed
# TYPE heap_committed gauge
heap_committed 2234880.0
```

Annotations:

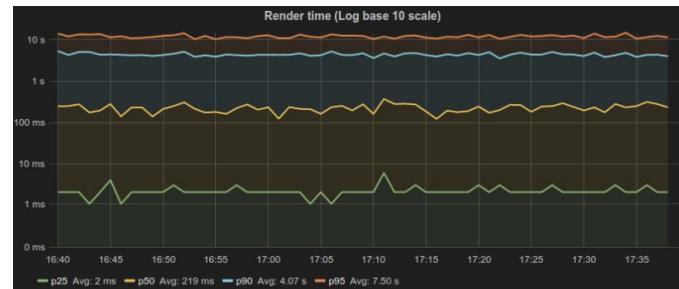
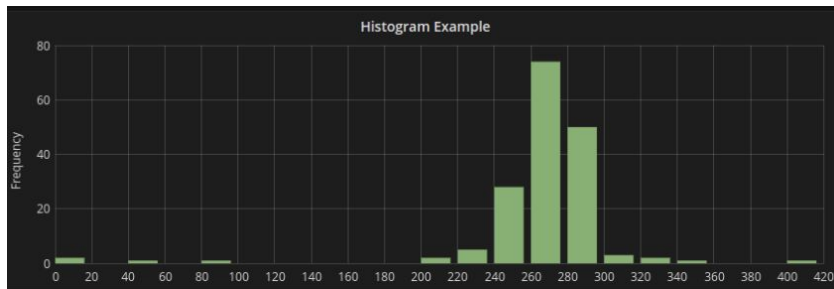
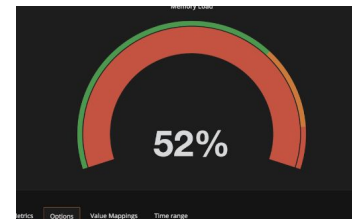
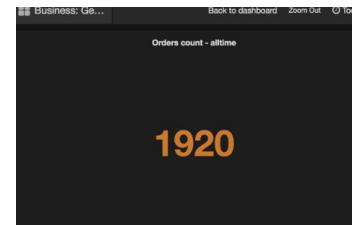
- For every metric (points to the metric name in the HELP and TYPE lines)
- Description (points to the description in the HELP line)
- Metric type (points to the metric type in the TYPE line)
- Values are float64 (points to the values in the data lines)





Tipos de métricas

- Contador: Sólo puede incrementarse o resetearse
- Gauge (medida): Se puede incrementar o decrementar
- Histograma: Valores agrupados en buckets
- Summary: Tiempos medios y percentiles



Prometheus Exporters

- Un exporter se encarga de convertir métricas ya existentes internas de la aplicación a métricas de prometheus.
- Existen múltiples exporters:
 - Bases de datos
 - Elasticsearch
 - MySQL
 - PostgreSQL
 - Redis
 - Hardware
 - Sistemas de mensajería
 - HTTP
 - Apache
 - Nginx
 - Almacenamiento
 - APIs
 - Github
 - Dockerhub



■ Métricas en Kubernetes

- Kubernetes dispone de muchas métricas que podemos exponer con diversos exporters.
 - **cAdvisor:** Se encarga de exponer todo tipo de métricas relacionadas con contenedores
 - **kube-state-metrics:** Se encarga de exponer métricas relacionadas con el clúster y su funcionamiento.
 - **nodeExporter:** Métricas del nodo a nivel de SO.

<https://github.com/google/cadvisor>

<https://github.com/kubernetes/kube-state-metrics>

```
sudo docker run \  
  --volume=/:/rootfs:ro \  
  --volume=/var/run:/var/run:ro \  
  --volume=/sys:/sys:ro \  
  --volume=/var/lib/docker:/var/lib/docker:ro \  
  --volume=/dev/disk:/dev/disk:ro \  
  --publish=8080:8080 \  
  --detach=true \  
  --name=cadvisor \  
  google/cadvisor:latest
```



■ Métricas expuestas por kube-state-metrics

- CronJob
- DaemonSet
- Job
- LimitRange
- Node
- PersistentVolume
- PersistentVolumeClaim
- Pod
- Pod Disruption Budget
- ReplicaSet
- ReplicationController
- ResourceQuota
- Service
- StatefulSet
- StorageClass
- Namespace
- Horizontal Pod Autoscaler
- Endpoint
- Secret
- ConfigMap
- Ingress
- CertificateSigningRequest
- VerticalPodAutoScaler



■ Métricas de aplicación (APM)

- Con Prometheus podemos además obtener métricas personalizadas de nuestras aplicaciones.
- Primero haremos un análisis y después definiremos las métricas.
- Para poder hacerlo instrumentamos nuestra aplicación con la librería de Prometheus que corresponda.
- Expondremos las métricas mediante un endpoint HTTP.
- Clientes disponibles
 - Go
 - Java, Scala
 - Python
 - Ruby
 - No oficiales: bash, PHP, node.js, etc



■ Demostración Instalación Prometheus

- <https://prometheus-operator.dev/docs/getting-started/installation/#install-using-helm-chart>



Kubernetes Observability con Elastic Stack



KEEPCODING

Tech School

■ Elastic Stack

- Componentes:
 - Beats / Elastic Agent:
 - Filebeat (logs)
 - Metricbeat (métricas) prometheus
 - Heartbeat (uptime)
 - Packetbeat (tráfico de red)
 - APM
 - Elasticsearch
 - Kibana



■ Elastic Stack

- ¿Qué ofrece el Stack de Elastic?
 - Full observability integradas en un único sistema.
 - Machine Learning, Elastic Security (SIEM)
 - Correlar logs, métricas y performance de aplicaciones.





KEEPCODING

Tech School

Madrid | Barcelona | Bogotá