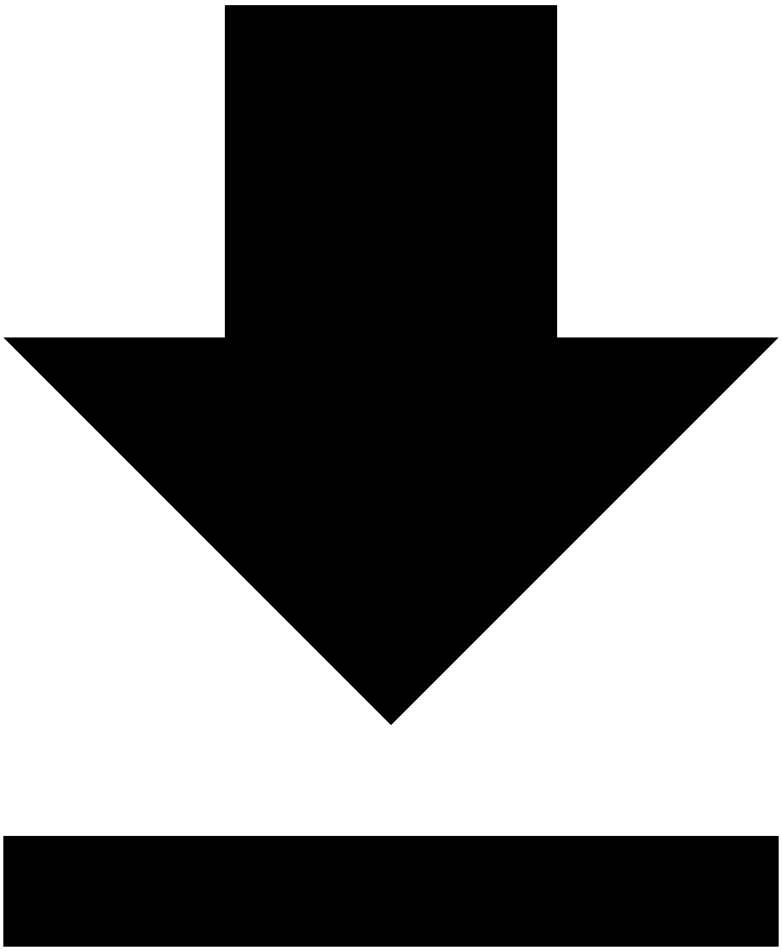
KCNA - Cloud Native Observability

Cloud Native Observability (8%)



Download PDF Version

This domain covers monitoring, logging, and tracing in cloud native environments.

Three Pillars of Observability

1. Metrics

Numeric measurements collected over time.

Examples: - CPU utilization - Memory usage - Request count - Error rate - Response latency

2. Logs

Timestamped records of discrete events.

Examples: - Application errors - Access logs - Audit logs - System events

3. Traces

Records of requests as they flow through distributed systems.

Examples: - Request path through microservices - Latency at each service - Error propagation

Prometheus

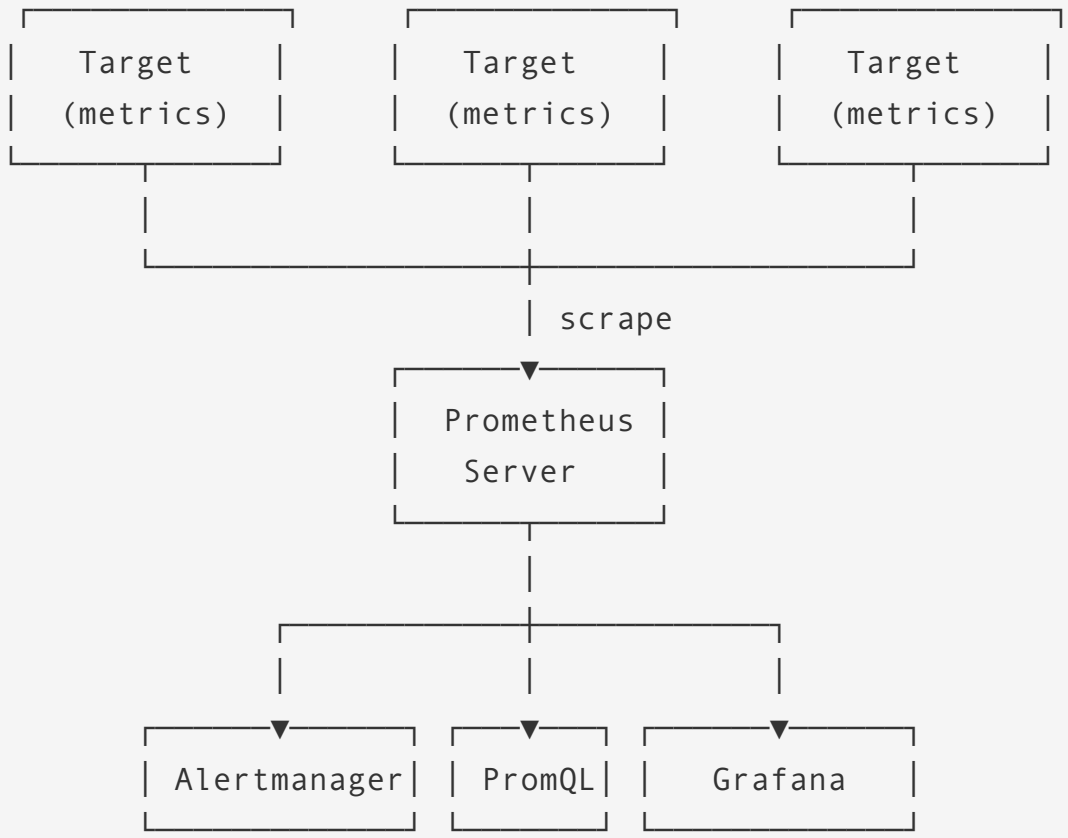
What is Prometheus?

Prometheus is an open-source monitoring and alerting toolkit, graduated from CNCF.

Key Features

- Multi-dimensional data model with time series
- PromQL query language
- Pull-based metrics collection
- Service discovery
- Alerting via Alertmanager

Architecture



Metric Types

Type	Description	Example
Counter	Cumulative, only increases	Total requests
Gauge	Can go up or down	Current temperature
Histogram	Samples in buckets	Request duration
Summary	Similar to histogram with quantiles	Request duration

PromQL Examples

```
 # CPU usage
rate(container_cpu_usage_seconds_total[5m])

# Memory usage
container_memory_usage_bytes

# HTTP request rate
rate(http_requests_total[5m])

# Error rate
rate(http_requests_total{status=~"5.."}[5m]) / rate(http_requests_total[5m]

# 95th percentile latency
histogram_quantile(0.95, rate(http_request_duration_seconds_bucket[5m]))
```

Grafana

What is Grafana?

Grafana is an open-source visualization and analytics platform.

Key Features

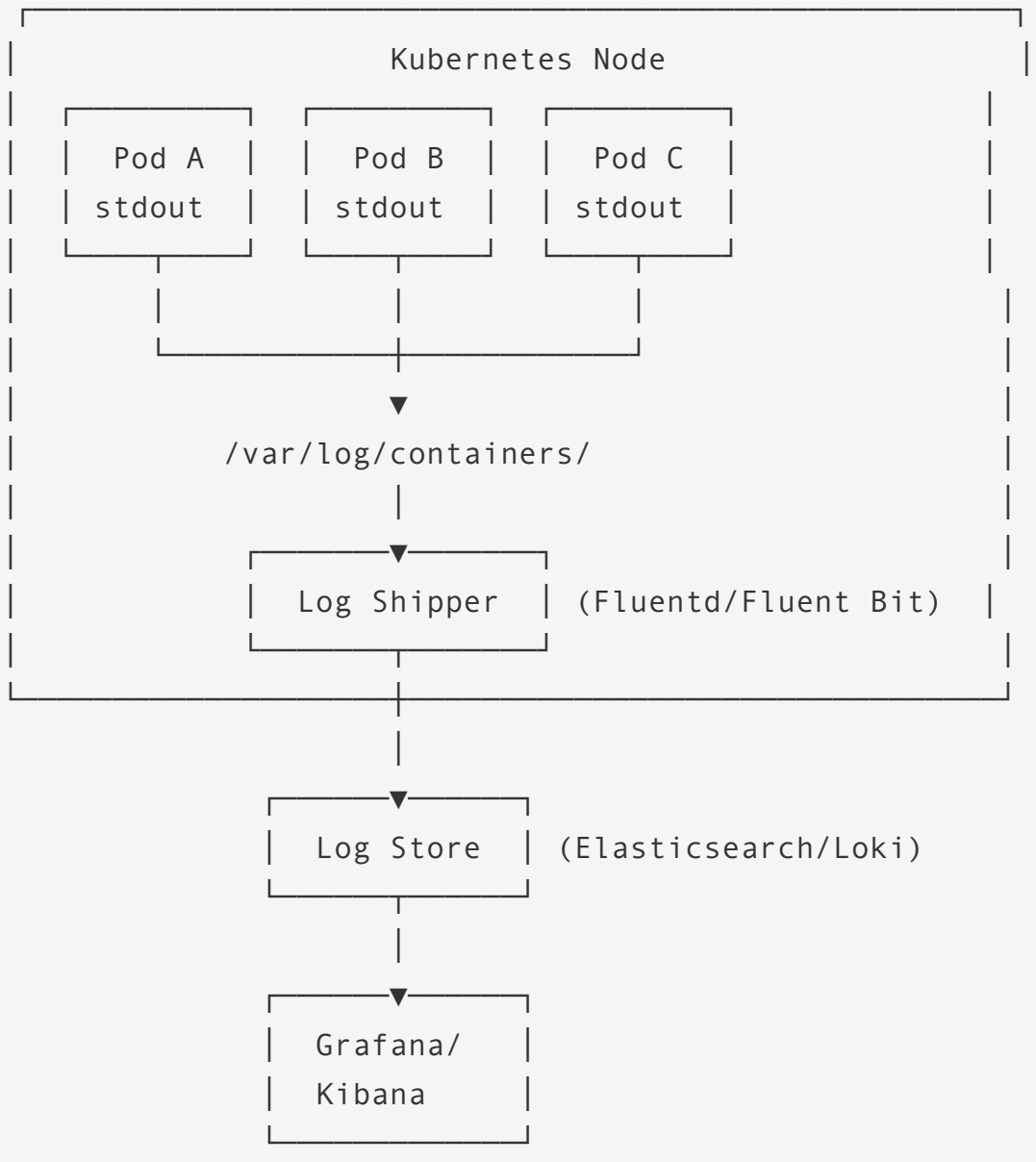
- Dashboard creation and sharing
- Multiple data source support
- Alerting capabilities
- Annotations
- Templating

Common Panels

- **Graph**: Time series visualization
- **Stat**: Single value display
- **Gauge**: Visual gauge
- **Table**: Tabular data
- **Heatmap**: Distribution over time

Logging

Logging Architecture in Kubernetes



Logging Tools

Tool	Description
Fluentd	Open-source data collector (CNCF Graduated)
Fluent Bit	Lightweight log processor
Elasticsearch	Search and analytics engine
Loki	Log aggregation system by Grafana
Kibana	Visualization for Elasticsearch

Kubernetes Logging Commands

```
 # View pod logs
kubectl logs <pod-name>

# Follow logs
kubectl logs -f <pod-name>

# Logs from specific container
kubectl logs <pod-name> -c <container-name>

# Previous container logs
kubectl logs <pod-name> --previous

# Logs with timestamps
kubectl logs <pod-name> --timestamps
```

Distributed Tracing

What is Distributed Tracing?

Distributed tracing tracks requests as they flow through multiple services, helping identify:

- Performance bottlenecks
- Error sources
- Service dependencies

Tracing Concepts

Concept	Description
Trace	End-to-end journey of a request
Span	Single operation within a trace
Context	Metadata propagated between services

Trace Example

```
Trace ID: abc123
├── Span: API Gateway (10ms)
│   └── Span: Auth Service (5ms)
├── Span: Order Service (50ms)
│   ├── Span: Database Query (20ms)
│   └── Span: Payment Service (25ms)
└── Span: Notification Service (15ms)
```

Tracing Tools

Tool	Description
Jaeger	Distributed tracing platform (CNCF Graduated)
Zipkin	Distributed tracing system
OpenTelemetry	Observability framework (CNCF)

OpenTelemetry

What is OpenTelemetry?

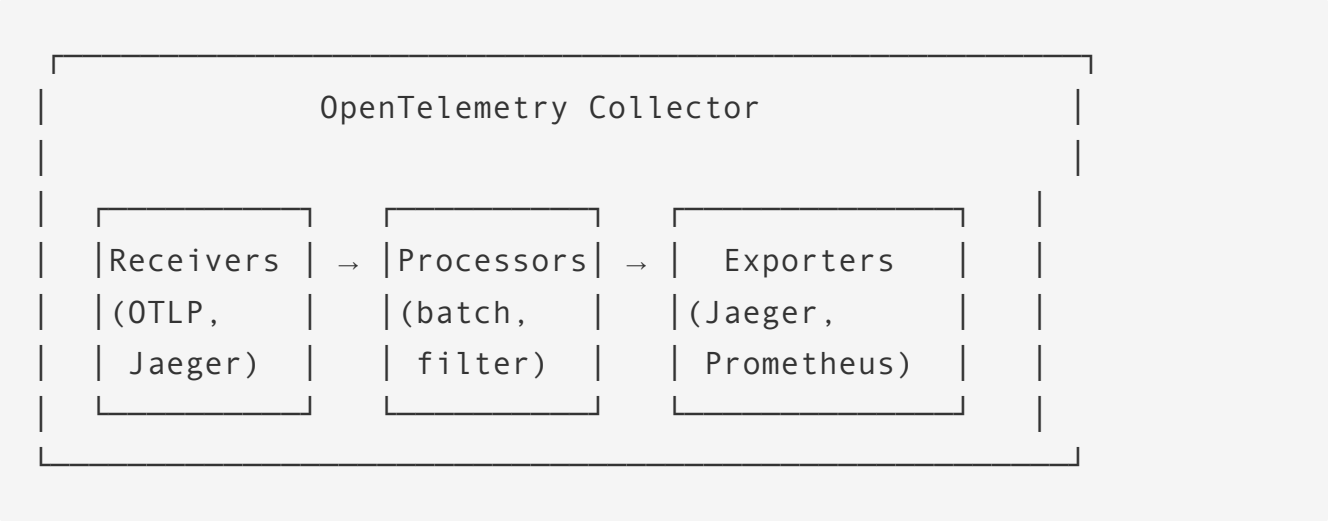
OpenTelemetry is a collection of tools, APIs, and SDKs for instrumenting, generating, collecting, and exporting telemetry data (metrics, logs, traces).

Components

- **API**: Defines how to generate telemetry

- **SDK**: Implements the API
- **Collector**: Receives, processes, and exports data
- **Exporters**: Send data to backends

OpenTelemetry Collector



Cost Management

Observability Costs

- **Storage**: Metrics, logs, and traces consume storage
- **Compute**: Processing and querying data
- **Network**: Data transfer between components

Cost Optimization

- Set appropriate retention periods
- Use sampling for high-volume traces
- Aggregate metrics where possible
- Filter unnecessary logs

Key Concepts to Remember

1. **Three pillars**: Metrics, Logs, Traces
2. **Prometheus uses pull-based** metrics collection
3. **PromQL** is the query language for Prometheus
4. **OpenTelemetry** unifies observability instrumentation
5. **Jaeger and Zipkin** are popular tracing tools

Practice Questions

1. What are the three pillars of observability?
2. What is the difference between a Counter and a Gauge in Prometheus?
3. What is the purpose of distributed tracing?
4. Name two CNCF graduated observability projects.
5. What does OpenTelemetry provide?

← Previous: Cloud Native Architecture | Back to KCNA Overview | Next: Cloud Native Application Delivery →