Linux Foundation PCA Practice Questions

Shared by Duke on 18-08-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

Which Prometheus component handles service discovery?

Options:

A- Alertmanager
B- Prometheus Server
C- Pushgateway
D- Node Exporter

Answer:

B

Explanation:

The Prometheus Server is responsible for service discovery, which identifies the list of targets to scrape. It integrates with multiple service discovery mechanisms such as Kubernetes, Consul, EC2, and static configurations.

This allows Prometheus to automatically adapt to dynamic environments without manual reconfiguration.

Question 2

Question Type: MultipleChoice

Which Alertmanager feature allows you to temporarily stop notifications for a specific alert?

Options:

A- Silence
B- Inhibition
C- Deduplication
D- Grouping

Answer:

A

Explanation:

The Silence feature in Alertmanager allows operators to mute specific alerts for a defined period. Each silence includes a matcher (labels), a creator, a comment, and an expiration time.

Silencing is useful during maintenance windows or known outages to prevent alert noise. Unlike inhibition, silences are manual and explicit.

Question 3

Question Type: MultipleChoice

Which kind of metrics are associated with the function deriv()?

Options:

A- Counters
B- Gauges
C- Summaries
D- Histograms

Answer:

B

Explanation:

The deriv() function in PromQL calculates the per-second derivative of a time series using linear regression over the provided time range. It estimates the instantaneous rate of change for metrics that can both increase and decrease --- which are typically gauges.

Because counters can only increase (except when reset), rate() or increase() functions are more appropriate for them. deriv() is used to identify trends in fluctuating metrics like CPU temperature, memory utilization, or queue depth, where values rise and fall continuously.

In contrast, summaries and histograms consist of multiple sub-metrics (e.g., _count, _sum, _bucket) and are not directly suited for derivative calculation without decomposition.

Extracted and verified from Prometheus documentation -- PromQL Functions -- deriv(), Understanding Rates and Derivatives, and Gauge Metric Examples.

Question 4

Question Type: MultipleChoice

What is the role of the Pushgateway in Prometheus?

Options:

A- To scrape short-lived targets directly.
B- To receive metrics pushed by short-lived batch jobs for later scraping by Prometheus.
C- To store metrics long-term for historical analysis.
D- To visualize metrics in Grafana.

Answer:

B

Explanation:

The Pushgateway is a Prometheus component used to handle short-lived batch jobs that cannot be scraped directly. These jobs push their metrics to the Pushgateway, which then exposes them for Prometheus to scrape.

This ensures metrics persist beyond the job's lifetime. However, it's not designed for continuously running services, as metrics in the Pushgateway remain static until replaced.

Question 5

Question Type: MultipleChoice

What's "wrong" with the myapp_filG_uploads_total{userid=,,5123",status="failed"} metric?

Options:

A- The userid should not be exposed as a label.

B- The status should not be exposed as a label.
C- The _total suffix should be omitted.
D- The metric name should consist of dashes instead of underscores.

Answer:

A

Explanation:

In Prometheus best practices, high-cardinality labels---especially those containing unique or user-specific identifiers---should be avoided. The metric myapp_filG_uploads_total{userid='5123',status='failed'} exposes the userid as a label, which is problematic. Each distinct value of a label generates a new time series in Prometheus. If there are thousands or millions of unique users, this would exponentially increase the number of time series, leading to cardinality explosion, degraded performance, and high memory usage.

The _total suffix is actually correct and required for counters, as per the Prometheus naming convention. The use of underscores in metric names is also correct, as Prometheus does not support dashes in metric identifiers. The status label, however, is perfectly valid because it typically has a low number of possible values (e.g., ''success'', ''failed'').

Verified from Prometheus official documentation sections Instrumentation -- Metric and Label Naming Best Practices and Writing Exporters.

Question 6

Question Type: MultipleChoice

What does scrape_interval configure in Prometheus?

Options:

A- It defines how frequently to scrape targets.
B- It defines how frequently to evaluate rules.
C- It defines how often to send alerts.
D- It defines how often to refresh metrics.

Answer:

A

Explanation:

In Prometheus, the scrape_interval parameter specifies how frequently the Prometheus server should scrape metrics from its configured targets. Each target exposes an HTTP endpoint (usually /metrics) that Prometheus collects data from at a fixed cadence. By default, the scrape_interval is set to 1 minute, but it can be overridden globally or per job configuration in the Prometheus YAML configuration file.

This setting directly affects the resolution of collected time series data---a shorter interval increases data granularity but also adds network and storage overhead, while a longer interval reduces load but might miss short-lived metric variations.

It is important to distinguish scrape_interval from evaluation_interval, which defines how often Prometheus evaluates recording and alerting rules. Thus, scrape_interval pertains only to data collection frequency, not to alerting or rule evaluation.

Extracted and verified from Prometheus documentation on Configuration File -- scrape_interval and Scraping Fundamentals sections.

Question 7

Question Type: MultipleChoice

How would you name a metric that tracks HTTP request duration?

Options:

A- request_duration_seconds
B- http.request_latency
C- http_request_duration
D- http_request_duration_seconds

Answer:

D

Explanation:

According to Prometheus metric naming conventions, a metric name must clearly describe what is being measured and include a unit suffix that specifies the base unit of measurement, following SI standards. For durations, the suffix _seconds is mandatory.

Therefore, the correct and standards-compliant name for a metric tracking HTTP request duration is:

http_request_duration_seconds

This name communicates:

http_request the subject being measured (HTTP requests),

duration the aspect being measured (the latency or time taken),

_seconds the unit of measurement (seconds).

This metric name typically corresponds to a histogram or summary, exposing submetrics such as _count, _sum, and _bucket. These represent the number of observations, total duration, and distribution across time buckets respectively.

Options A, B, and C fail to fully comply with Prometheus naming standards --- they either omit the http_ prefix, use invalid separators (dots), or lack the required unit suffix.

Verified from Prometheus documentation -- Metric and Label Naming Conventions, Instrumentation Best Practices, and Histogram and Summary Metric Naming Patterns.

Question 8

Question Type: MultipleChoice

How can you send metrics from your Prometheus setup to a remote system, e.g., for long-term storage?

Options:

A- With 'scraping'
B- With 'remote write'
C- With S3 Buckets
D- With 'federation'

Answer:

B

Explanation:

Prometheus provides a feature called Remote Write to transmit scraped and processed metrics to an external system for long-term storage, aggregation, or advanced analytics. When configured, Prometheus continuously pushes time series data to the remote endpoint defined in the remote_write section of the configuration file.

This mechanism is often used to integrate with long-term data storage backends such as Cortex, Thanos, Mimir, or InfluxDB, enabling durable retention and global query capabilities beyond Prometheus's local time series database limits.

In contrast, ''scraping'' refers to data collection from targets, while ''federation'' allows hierarchical Prometheus setups (pulling metrics from other Prometheus instances) but does not serve as long-term storage. Using ''S3 Buckets'' directly is also unsupported in native Prometheus configurations.

Extracted and verified from Prometheus documentation -- Remote Write/Read APIs and Long-Term Storage Integrations sections.

Question 9

Question Type: MultipleChoice

Which option best PromQL queries is invalid?

Options:

A- max by (instance) up
B- max on (instance) (up)
C- max without (instance) up
D- max without (instance, job) up

Answer:

B

Explanation:

The max operator in PromQL is an aggregation operator, not a binary vector matching operator. Therefore, the valid syntax for aggregation uses by() or without(), not on().

max by (instance) up Valid; aggregates maximum values per instance.

max without (instance) up and max without (instance, job) up Valid; aggregates over all labels except those listed.

max on (instance) (up) Invalid; the keyword on() is only valid in binary operations (e.g., +, -, and, or, unless), where two vectors are being matched on specific labels.

Hence, max on (instance) (up) is a syntax error in PromQL because on() cannot be used directly with aggregation operators.

Verified from Prometheus documentation -- Aggregation Operators, Vector Matching -- on()/ignoring(), and PromQL Language Syntax Reference sections.