DirectCertify

Unlock Your Potential, Certify Your Success

Linux Foundation

PCA Exam

Prometheus Certified Associate Exam

Exam Latest Version: 6.0

DEMO Version

Full Version Features:

• 90 Days Free Updates

• 30 Days Money Back Guarantee

• Instant Download Once Purchased

• 24 Hours Live Chat Support

Full version is available at link below with affordable price.

https://www.directcertify.com/linux-foundation/pca

Question 1. (Single Select)

How many metric types does Prometheus text format support?

A: 40
B: 4
C: 8
D: 10

Correct Answer: B

Explanation:

Prometheus defines four core metric types in its official exposition format, which are: Counter, Gauge, Histogram, and Summary. These types represent the fundamental building blocks for expressing quantitative measurements of system performance, behavior, and state.

A Counter is a cumulative metric that only increases (e.g., number of requests served).

A Gauge represents a value that can go up and down, such as memory usage or temperature.

A Histogram samples observations (e.g., request durations) and counts them in configurable buckets, providing both counts and sum of observed values.

A Summary is similar to a histogram but provides quantile estimation over a sliding time window along with count and sum metrics.

These four types are the only officially supported metric types in the Prometheus text exposition format as defined by the Prometheus data model. Any additional metrics or custom naming conventions are built on top of these core types but do not constitute new types.

Extracted and verified from Prometheus official documentation sections on Metric Types and Exposition Formats in the Prometheus study materials.

Question 2. (Single Select)

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

A: With "scraping"
B: With "remote write"
C: With S3 Buckets
D: With "federation"

Correct 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 3. (Single Select)

What is api_http_requests_total in the following metric?

api_http_requests_total{method="POST", handler="/messages"}

A: "api_http_requests_total" is a metric label name.
B: "api_http_requests_total" is a metric type.

C: "api_http_requests_total" is a metric name.
D: "api_http_requests_total" is a metric field.

Correct Answer: C

Explanation:

In Prometheus, the part before the curly braces {} represents the metric name. Therefore, in the metric api_http_requests_total{method="POST", handler="/messages"}, the term api_http_requests_total is the metric name. Metric names describe the specific quantity being measured — in this example, the total number of HTTP requests received by an API.

The portion within the braces defines labels, which provide additional dimensions to the metric. Here, method="POST" and handler="/messages" are labels describing request attributes. The metric name should follow Prometheus conventions: lowercase letters, numbers, and underscores only, and ending in _total for counters.

This naming scheme ensures clarity and standardization across instrumented applications. The metric type (e.g., counter, gauge) is declared separately in the exposition format, not within the metric name itself.

Verified from Prometheus documentation – Metric and Label Naming, Data Model, and Instrumentation Best Practices sections.

Question 4. (Single Select)

Which of the following is a valid metric name?

A: go routines
B: go.goroutines
C: go_goroutines
D: 99_goroutines

Correct Answer: C

Explanation:

According to Prometheus naming rules, metric names must match the regex [a-zA-Z_:][a-zA-Z0-9_:]*. This means metric names must begin with a letter, underscore, or colon, and can only contain letters, digits, and underscores thereafter.

The valid metric name among the options is go_goroutines, which follows all these rules. It starts with a letter (g), uses underscores to separate words, and contains only allowed characters.

By contrast:

go routines is invalid because it contains a space.

go.goroutines is invalid because it contains a dot (.), which is reserved for recording rule naming hierarchies, not metric identifiers.

99_goroutines is invalid because metric names cannot start with a number.

Following these conventions ensures compatibility with PromQL syntax and Prometheus' internal data model.

Extracted from Prometheus documentation – Metric Naming Conventions and Data Model Rules sections.

Question 5. (Single Select)

What is an example of a single-target exporter?

A: Redis Exporter
B: SNMP Exporter
C: Node Exporter
D: Blackbox Exporter

Correct Answer: A

Explanation:

A single-target exporter in Prometheus is designed to expose metrics for a specific service instance rather than multiple dynamic endpoints. The Redis Exporter is a prime example — it connects to one Redis server instance and exports its metrics (like memory usage, keyspace

hits, or command statistics) to Prometheus.

By contrast, exporters like the SNMP Exporter and Blackbox Exporter can probe multiple targets dynamically, making them multi-target exporters. The Node Exporter, while often deployed per host, is considered a host-level exporter, not a true single-target one in configuration behavior.

The Redis Exporter is instrumented specifically for a single Redis endpoint per configuration, aligning it with Prometheus's single-target exporter definition. This design simplifies monitoring and avoids dynamic reconfiguration.

Verified from Prometheus documentation and official exporter guidelines – Writing Exporters, Exporter Types, and Redis Exporter Overview sections.

DirectCertify

Unlock Your Potential, Certify Your Success

Full version is available at link below with affordable price.

https://www.directcertify.com/linux-foundation/pca

30% Discount Coupon Code: LimitedTime2025