PCA Exam Duration - Quiz Linux Foundation First-grade Valid PCA Exam Cost

If you are forced to pass exams and obtain certification by your manger, our PCA original questions will be a good choice for you. Our products can help you clear exams at first shot. We promise that we provide you with best quality PCA original questions and competitive prices. We offer 100% pass products with excellent service. We provide one year studying assist service and one year free updates downloading of Linux Foundation PCA Exam Questions. If you fail exam we support to exchange and full refund.

Linux Foundation PCA Exam Syllabus Topics:

Topic	Details
Topic 1	• Instrumentation and Exporters: This domain evaluates the abilities of Software Engineers and addresses the methods for integrating Prometheus into applications. It includes the use of client libraries, the process of instrumenting code, and the proper structuring and naming of metrics. The section also introduces exporters that allow Prometheus to collect metrics from various systems, ensuring efficient and standardized monitoring implementation.
Topic 2	• PromQL: This section of the exam measures the skills of Monitoring Specialists and focuses on Prometheus Query Language (PromQL) concepts. It covers data selection, calculating rates and derivatives, and performing aggregations across time and dimensions. Candidates also study the use of binary operators, histograms, and timestamp metrics to analyze monitoring data effectively, ensuring accurate interpretation of system performance and trends.
Topic 3	• Prometheus Fundamentals: This domain evaluates the knowledge of DevOps Engineers and emphasizes the core architecture and components of Prometheus. It includes topics such as configuration and scraping techniques, limitations of the Prometheus system, data models and labels, and the exposition format used for data collection. The section ensures a solid grasp of how Prometheus functions as a monitoring and alerting toolkit within distributed environments.

Topic 4	• Observability Concepts: This section of the exam measures the skills of Site Reliability Engineers and covers the essential principles of observability used in modern systems. It focuses on understanding metrics, logs, and tracing mechanisms such as spans, as well as the difference between push and pull data collection methods. Candidates also learn about service discovery processes and the fundamentals of defining and maintaining SLOs, SLAs, and SLIs to monitor performance and reliability.
Topic 5	• Alerting and Dashboarding: This section of the exam assesses the competencies of Cloud Operations Engineers and focuses on monitoring visualization and alert management. It covers dashboarding basics, alerting rules configuration, and the use of Alertmanager to handle notifications. Candidates also learn the core principles of when, what, and why to trigger alerts, ensuring they can create reliable monitoring dashboards and proactive alerting systems to maintain system stability.

>> PCA Exam Duration <<

100% Pass 2026 PCA: Prometheus Certified Associate Exam Newest Exam Duration

New developments in the tech sector always bring new job opportunities. These new jobs have to be filled with the Prometheus Certified Associate Exam (PCA) certification holders. So to fill the space, you need to pass the Linux Foundation PCA exam. Earning the Prometheus Certified Associate Exam (PCA) certification helps you clear the obstacles you face while working in the Linux Foundation field. To get prepared for the Prometheus Certified Associate Exam (PCA) certification exam, applicants face a lot of trouble if the study material is not updated.

Linux Foundation Prometheus Certified Associate Exam Sample Questions (Q21-Q26):

NEW QUESTION # 21
What is a rule group?

- A. It is the set (the group) of all the rules in a file.
- B. It is a set of rules, split into groups by type.
- C. It is a set of rules that are executed sequentially.
- D. It is a set of rules that are grouped by labels.

Answer: C

Explanation:
In Prometheus, a rule group is a logical collection of recording and alerting rules that are evaluated sequentially at a specified interval. Rule groups are defined in YAML files under the groups: key, with each group containing a name, an interval, and a list of rules.
For example:
groups:
- name: example
interval: 1m
rules:
- record: job:http_inprogress_requests:sum
expr: sum(http_inprogress_requests) by (job)
All rules in a group share the same evaluation schedule and are executed one after another. This ensures deterministic order, especially when one rule depends on another's result.
Reference:
Verified from Prometheus documentation - Rule Configuration, Rule Groups and Evaluation Order, and Recording & Alerting Rules Guide.

NEW QUESTION # 22
What's "wrong" with the myapp_filG_uploads_total{userid=,,5123",status="failed"} metric?

- A. The metric name should consist of dashes instead of underscores.

- B. The userid should not be exposed as a label.
- C. The _total suffix should be omitted.
- D. The status should not be exposed as a label.

Answer: B

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

NEW QUESTION # 23
How can you select all the up metrics whose instance label matches the regex fe-.*?

- A. up{instance=~"fe-.*"}
- B. up{instance~"fe-.*"}
- C. up{instance=regexp(fe-.*)}
- D. up{instance="fe-.*"}

Answer: A

Explanation:
PromQL supports regular expression matching for label values using the =~ operator. To select all time series whose label values match a given regex pattern, you use the syntax {label_name=~"regex"}.
In this case, to select all up metrics where the instance label begins with fe-, the correct query is:
up{instance=~"fe-.*"}
Explanation of operators:
= → exact match.
!= → not equal.
=~ → regex match.
!~ → regex not match.
Option D uses the correct =~ syntax. Options A and B use invalid PromQL syntax, and option C is almost correct but includes a misplaced extra quote style (~"), which would cause a parsing error.
Reference:
Verified from Prometheus documentation - Expression Language Data Selectors, Label Matchers, and Regular Expression Matching Rules.

NEW QUESTION # 24
Which Alertmanager feature allows you to temporarily stop notifications for a specific alert?

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

Answer: D

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.

NEW QUESTION # 25

Which of the following signal belongs to symptom-based alerting?

- A. API latency
- B. Database memory utilization
- C. Disk space
- D. CPU usage

Answer: A

Explanation:

Symptom-based alerting focuses on user-visible problems or service-impacting symptoms rather than low-level resource metrics. In Prometheus and Site Reliability Engineering (SRE) practices, alerts should signal conditions that affect users' experience - such as high latency, request failures, or service unavailability - instead of merely reflecting internal resource states.

Among the options, API latency directly represents the performance perceived by end users. If API response times increase, it immediately impacts user satisfaction and indicates a possible service degradation.

In contrast, metrics like disk space, CPU usage, or database memory utilization are cause-based metrics - they may correlate with problems but do not always translate into observable user impact.

Prometheus alerting best practices recommend alerting on symptoms (via RED metrics - Rate, Errors, Duration) while using cause-based metrics for deeper investigation and diagnosis, not for immediate paging alerts.

Reference:

Verified from Prometheus documentation - Alerting Best Practices, Symptom vs. Cause Alerting, and RED/USE Monitoring Principles sections.

NEW QUESTION # 26

......

myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, www.stes.tyc.edu.tw, bbs.t-firefly.com, www.stes.tyc.edu.tw, Disposable vapes

BTW, DOWNLOAD part of ActualCollection PCA dumps from Cloud Storage: https://drive.google.com/open?id=1XgflJm8WEVJkSFN_Wj_J9ZNf5eJqAe2l