DUMPSARENA

PeopleCert DevOps Site Reliability Engineer (SRE)

PEOPLECERT DevOps-SRE

Version Demo

Total Demo Questions: 13

Total Premium Questions: 139

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, SRE Principles & Practices	32
Topic 2, Service Level Objectives & Error Budgets	27
Topic 3, Observability	18
Topic 4, SRE Tools & Automation	16
Topic 5, Anti-Fragility & Failure	14
Topic 6, Organizational Impact of SRE	29
Topic 7, Mix Questions	3
Total	139

QUESTION NO: 1

What metrics will embracing failure help to improve?

A. Mean time to detect and mean time between system incidents

B. Change lead time and change failure rate

C. Empirical test data and mean time to recover service

D. Mean time to detect and mean time to recover

ANSWER: D

Explanation:

Mean time to detect and mean time to recover is correct because an SRE practice of embracing failure makes weaknesses observable before they become major customer-impacting incidents. Techniques such as controlled fault injection, game days, resilience testing, and blameless learning from incidents exercise detection, alerting, runbooks, escalation paths, and recovery mechanisms under realistic conditions. This gives teams an opportunity to reduce the time required to recognize abnormal service behavior and to restore an acceptable service state after a failure occurs. Repeated practice also produces concrete evidence about dependencies, failure modes, and operational response, enabling teams to automate remediation and improve incident procedures. Google's SRE guidance describes the value of testing systems through failure and treating incidents as learning opportunities, while its incident-management material identifies detection and mitigation as central stages of effective response. These practices directly support lower detection and recovery times, two key indicators of operational resilience. See Google SRE Book: Addressing Cascading Failures and Google SRE Workbook: Incident Response.

QUESTION NO: 2

Which of the following is the BEST example or an SRE team that embraces full-service ownership?

A. The team is responsible for the cooing and improvement of me application.

B. The team is accountable for the application development and performance.

C. The team is responsible for application performance and reliability aspects.

D. The team is accountable for coding, shipping, and improving the application.

ANSWER: D

Explanation:

"The team is accountable for coding, shipping, and improving the application" is the best example of full-service ownership because it expresses end-to-end accountability for a service throughout its lifecycle. In an SRE and DevOps operating model, ownership is not limited to writing software or responding to operational issues after release. The same team owns delivering the application into production, learning from its real-world operation, and continuously making it better. This creates a direct feedback loop between engineering decisions, production outcomes, customer experience, reliability goals, and improvement work.

Full-service ownership supports the core SRE objective of making reliability a shared engineering responsibility rather than handing operations to a separate downstream team. Accountability for coding, shipping, and improving also encourages teams to automate deployment and operational work, use service-level objectives to guide decisions, and prioritize reliability improvements based on production evidence. Google's DevOps guidance describes full service ownership as the team that builds a service also being responsible for operating it in production. This ownership model is foundational to effective collaboration between development and reliability engineering practices. See Google Cloud's DevOps technical practices and the Google SRE Book introduction.

QUESTION NO: 3

Which of the following BEST describes an advantage of a container-based structure?

A. The portability created by containers enables software to run independently of the host operating system

B. The lightweight nature of containers requires fewer developers to actually create the software code

C. Software runs much more efficiently in containers because of the ability to run on virtual machines

D. The security of applications in containers is simplified because they share the security of the host system

ANSWER: A

Explanation:

The portability created by containers enables software to run independently of the host operating system is the best description of a container-based structure's advantage. A container packages an application together with the libraries, runtime, configuration, and other dependencies it needs. This produces a standardized, self-contained application unit that can be built once and run consistently across development, testing, and production environments. Consequently, teams reduce environment-specific deployment differences and can automate delivery more reliably.

More precisely, containers abstract the application from many host-environment differences while sharing the host operating system's kernel; "independently" in this context means that the application does not need to be individually installed and configured for each target environment. This portability is central to SRE practices because it supports repeatable deployments, rapid scaling, immutable infrastructure patterns, and dependable orchestration through platforms such as Kubernetes. Docker describes containers as isolated environments that package code and its dependencies, while Kubernetes identifies containerization as the mechanism that enables portable and consistent deployment. See the Docker overview and the Kubernetes documentation.

QUESTION NO: 4

Which of the following is the MOST accurate description of Kubernetes?

A. A proprietary system developed to automate the integration, building, testing, and deployment of application containers

B. An independent platform that enables organizations to implement continuous integration and delivery practices

C. A platform used to manage containers in a cloud environment and also includes automated scaling and failover

D. An open-source operating system on which containerized applications can be run, monitored, and managed efficiently

ANSWER: C

Explanation:

Kubernetes is an open-source platform for orchestrating containerized workloads and services across a cluster of machines, including cloud environments. It provides the control plane and declarative mechanisms needed to deploy and operate applications reliably at scale. A user specifies the desired state—for example, the number of application replicas and the container image to run—and Kubernetes continually works to maintain that state. This includes scheduling containers, service discovery and load balancing, rolling updates, automated recovery when a container or node fails, and horizontal scaling. Consequently, the description "A platform used to manage containers in a cloud environment and also includes automated scaling and failover" most accurately captures Kubernetes' central purpose and important reliability capabilities. Although Kubernetes can run in many environments, including on-premises infrastructure, cloud deployment is a common and appropriate context for this description. Kubernetes documentation identifies service discovery, load balancing, self-healing, automated rollouts and rollbacks, and scaling as core features, all of which support resilient operation of distributed containerized services. See the Kubernetes overview and its workloads documentation for the platform's orchestration model and capabilities.

QUESTION NO: 5

Which of the following is the MOST likely outcome when the workforce puts the â€œpartsâ€□ before the â€œwholeâ€□?

A. Increased employee motivation and morale

B. Increased introversion and decreased efficiency

C. A voluntary sharing of resources and information

D. A focus on common interests and lesser conflicts

ANSWER: B

Explanation:

“Increased introversion and decreased efficiency” is the most likely outcome because prioritizing individual parts of an organization over the whole encourages local optimization rather than shared, end-to-end responsibility. In an SRE context, reliable services depend on teams working toward common customer and business outcomes, with development, operations, and other functions sharing information, risks, feedback, and operational ownership. When a workforce concentrates primarily on its own departmental goals, work becomes more isolated and communication paths become less effective. This organizational introversion creates friction at team boundaries, slows decisions and incident response, and makes it harder to identify or improve the complete service value stream. Efficiency declines because teams can optimize their own activity while creating additional coordination effort, handoffs, and constraints elsewhere in the system. SRE practice therefore favors service-wide thinking, collaboration, and alignment around reliability objectives rather than disconnected team-level interests. Google’s SRE guidance describes how organizational and cultural practices support reliable service operations, including the importance of shared responsibility and reducing operational toil; see the Google SRE Book and the Google SRE Workbook.

QUESTION NO: 6

Which of the following BESTdescribes the most important rationale for NOTseeking an SLO of 100% availability?

A. It is not realistic for the complexity and scale of services.

B. The likely results is failure where such targets are defined.

C. There is no room for improvements if targets are so high.

D. The user satisfaction score ism affected by a tow percent

ANSWER: A

Explanation:

It is not realistic for the complexity and scale of services. An availability SLO is a reliability target that deliberately balances user expectations against the engineering cost of pursuing greater reliability. Real production services depend on infrastructure, networks, software releases, third-party dependencies, and human operational processes, all of which can introduce occasional failures or planned disruption. Requiring 100% availability means that even the smallest interruption exhausts the entire error budget. This leaves no practical tolerance for incidents, maintenance, experimentation, or controlled changes that are necessary to improve and operate a service safely over time.

SRE practice therefore uses an SLO below perfection to define an acceptable level of reliability and an error budget representing the permitted amount of unreliability. The budget gives teams an objective mechanism for deciding when they can prioritize feature delivery and when reliability work must take precedence. A target should be demanding enough to protect the user experience, while still being achievable and economically appropriate for the service’s architecture and scale. Google’s SRE guidance explains the relationship between SLOs, error budgets, and the cost of higher reliability in its Implementing SLOs workbook and Service Level Objectives chapter.

QUESTION NO: 7

Where should an organization store versioned and signed artifacts that are used to deploy system components?

A. In the Configuration Management System (CMS)

B. In a Subversion source code repository

C. In a Definitive Media Library (DML)

D. In a secure artifact repository

ANSWER: D

Explanation:

In a secure artifact repository is correct because deployable artifacts must be managed separately from source code and configuration records throughout their lifecycle. An artifact repository provides a controlled location for immutable, versioned build outputs such as packages, binaries, container images, charts, and deployment bundles. This allows deployment automation to retrieve the precise artifact that was built, tested, approved, and signed, supporting repeatable releases and reliable rollback to a known version.

Signing establishes provenance and integrity: a deployment process can verify that an artifact originated from an authorized build process and has not been altered after publication. Repository access controls, retention policies, audit records, vulnerability scanning, and promotion between environments further help protect the software supply chain. These capabilities make the repository a trusted release source rather than merely a storage location. The SLSA framework describes the importance of provenance for verifying where and how software artifacts were produced: SLSA Provenance. The Open Container Initiative distribution specification also defines registries as systems for distributing versioned container image content: OCI Distribution Specification. Keeping signed, versioned deployment artifacts in a secure artifact repository therefore supports traceability, integrity verification, and consistent deployment of system components.

QUESTION NO: 8

Which of the following is the MOST accurate description of Kubemetes?

A. A proprietary system developed to automate the integration, building, testing and deployment of application containers

B. An independent platform mat enables organizations to implement continuous integration and delivery practices

C. A platform used to manage containers in a cloud environment and also includes automated scaling and failover

D. An open-source operating system on which containerized applications can be run monitored and managed efficiently

ANSWER: C

Explanation:

“A platform used to manage containers in a cloud environment and also includes automated scaling and failover” is the most accurate description. Kubernetes is an open-source container-orchestration platform that manages containerized workloads and services across a cluster. It provides the desired-state model needed to deploy and operate applications reliably: users declare the intended number of running instances and the platform continuously works to maintain that state. Its workload-management capabilities include scheduling containers onto suitable nodes, service discovery and load balancing, controlled rollouts and rollbacks, and horizontal scaling. Kubernetes also provides self-healing behavior, such as restarting failed containers, replacing failed instances, and removing unhealthy instances from service endpoints. These capabilities support resilient operation and automated recovery, commonly described as failover in an operational context. Although Kubernetes can run in on-premises, hybrid, edge, and public-cloud environments, it is extensively used to manage cloud-based container platforms. The Kubernetes project describes itself as a production-grade platform for automating deployment, scaling, and management of containerized applications. See the Kubernetes Overview and the documentation on container lifecycle and self-healing behavior.

QUESTION NO: 9

Which of the following BEST explains why a service dependency should have a timeout?

A. To prevent requests from waiting indefinitely for a slow or unavailable dependency

B. To ensure every failed request is retried without limit

C. To increase the amount of traffic sent to an unhealthy dependency

D. To eliminate the need for monitoring of dependent services

ANSWER: A

Explanation:

To prevent requests from waiting indefinitely for a slow or unavailable dependency is correct. A timeout establishes a maximum period that a caller will wait for a dependency to respond. When that period is exceeded, the caller can fail fast, invoke a fallback, return an appropriate error, or shed load rather than continuing to consume resources.

Without timeouts, waiting requests can accumulate and exhaust connection pools, threads, memory, or other resources. This can cause a localized dependency failure to cascade through the wider system. Timeouts should be selected carefully and coordinated with retries and overall request deadlines. See Google SRE Book: Addressing Cascading Failures.

QUESTION NO: 10

Which or the following B6ST identifies a desired objective of the production reachless review (PRR)?

A. To ensure the service is ready for an SRE team to take over support and care for it

B. To improve the reliability of the service in the development and testing environment

C. To validate the service meets international quality standards and frameworks

D. To ensure the service owner transitions operational accountability to the SRE team

ANSWER: A

Explanation:

The desired objective is **"To ensure the service is ready for an SRE team to take over support and care for it."** A production readiness review is a structured assessment performed before a service is accepted into ongoing production support. Its purpose is to establish that the service can be operated reliably and sustainably, including that it has appropriate monitoring, alerting, capacity planning, documentation, operational procedures, and ownership arrangements. The review creates a shared understanding between the development team and the SRE function about the operational work required to run the service safely.

Readiness for SRE support is central because the review is not merely a general quality assessment; it focuses on whether the service can meet production operational expectations without creating unacceptable toil or reliability risk for its operators. The review may identify gaps and track them as follow-up actions before, or as conditions of, acceptance. Google's SRE guidance describes production-readiness evaluation as a means of ensuring that a service meets operational standards and that responsibilities are clear. See the Google SRE Workbook production checklist and the Google SRE Book discussion of the production environment.

QUESTION NO: 11

Which of the following BEST describes capacity planning?

A. Monitoring the percentage of capacity of resources being used over a time period

B. Activities performed to manage provider resources and provide multiple services

C. Activities used to create a plan that manages resources to meet service demand

D. Determining the maximum amount that any resource can accommodate or deliver

ANSWER: C

Explanation:

“Activities used to create a plan that manages resources to meet service demand” best describes capacity planning. Capacity planning is a forward-looking management activity: it assesses expected demand for services, translates that demand into resource requirements, and establishes how infrastructure, platforms, applications, people, or supplier capacity must be provisioned or adjusted. Its purpose is to ensure that services can achieve agreed performance and availability needs at an appropriate cost as demand changes.

In an SRE and DevOps context, this includes using demand forecasts, workload patterns, service-level objectives, utilization data, growth assumptions, and resilience requirements to make informed decisions before capacity becomes a constraint. The resulting plan can cover scaling strategies, procurement or cloud-reservation decisions, performance testing, automation thresholds, and review points. Capacity planning is therefore broader than simply measuring utilization at a point in time: measurement supplies important input, while planning turns that information into an actionable approach for meeting present and anticipated service demand. This aligns with the ITIL capacity and performance management practice, which seeks to ensure services achieve agreed performance targets and meet current and future demand cost-effectively. See the ITIL capacity and performance management practice guide and the PeopleCert ITIL certification portfolio.

QUESTION NO: 12

Which type of engineering work will reduce toil within the service?

A. Continuous delivery pipelines

B. Scripts and automation tools outside of the service

C. Scalable infrastructure

D. Internal automation

ANSWER: D

Explanation:

Internal automation is correct because it embeds the handling of routine operational work into the service or its directly managed operational mechanisms. In SRE, toil is manual, repetitive, automatable work that has no enduring engineering value. Engineering the service to perform these actions itself—such as automatically detecting and remediating known conditions, managing routine lifecycle actions, validating configuration, or executing safe recovery procedures—permanently removes recurring human effort from normal operations. This makes the service easier to run as it scales and frees reliability engineers to focus on higher-value engineering work.

The Google SRE guidance identifies automation as a key means of eliminating toil, while also stressing that automation should be approached as an engineering investment with clear operational benefits. Automation integrated into the service’s operation is especially valuable because it is maintained alongside the service and can be designed with the service’s reliability requirements, observability, safeguards, and failure modes in mind. This directly supports the SRE objective of reducing operational burden through durable engineering improvements. See the Google SRE Book, “Eliminating Toil” and the SRE Workbook chapter on eliminating toil.

QUESTION NO: 13

Which of the following BEST explains the purpose of a blameless postmortem?

A. To identify systemic improvements without focusing on individual fault

B. To determine which responder should be disciplined after an outage

C. To approve the next production deployment after an incident

D. To establish a new contractual service-level agreement

ANSWER: A

Explanation:

To identify systemic improvements without focusing on individual fault is correct. A blameless postmortem examines the technical, process, organizational, and contextual conditions that contributed to an incident. It seeks to understand how reasonable actions and decisions resulted in an unexpected outcome within a complex system.

This approach encourages accurate reporting, shared learning, and meaningful follow-up actions. It does not remove accountability for improving systems and practices; rather, it directs accountability toward correcting contributing conditions and reducing the likelihood or impact of recurrence. See Google SRE Book: Postmortem Culture and Google SRE Workbook: Postmortem Culture.