Senior Software Engineer Interview Preparation Guide

Complete Learning Path to Crack Your Dream Job

Executive Summary

This comprehensive guide provides an in-depth, organized pathway to master all the technical skills and knowledge areas required for the Senior Software Engineer position in Bangalore. The role demands expertise in Java 17, build tools, containerization, messaging systems, API development, database optimization, and best practices implementation.

Key Focus Areas:

- Java 17 advanced features and memory management
- Build automation with Gradle
- Container orchestration with Kubernetes
- Messaging systems (RabbitMQ/Kafka)
- REST API design and development
- SQL optimization and database management
- CI/CD pipelines with GitHub Actions
- Infrastructure as Code with Terraform
- Performance optimization and debugging
- Security best practices and coding standards

1. Java 17 Mastery (Core Foundation)

1.1 Java 17 Features Deep Dive

Sealed Classes (JEP 409)

- Restrict inheritance to specific subclasses only
- Unlike `final` (blocks all inheritance) or `abstract` (allows any inheritance)
- Syntax: `public sealed class Shape permits Circle, Rectangle, Triangle`
- Use cases: Domain modeling, API design, pattern matching

Pattern Matching for instanceof (JEP 394)

```
// Old approach
if (obj instanceof String) {
    String s = (String) obj;
    System.out.println(s.length());
}

// Java 17 approach
if (obj instanceof String s) {
    System.out.println(s.length());
}
```

Records (Finalized in Java 17)

- Immutable data classes with automatic equals(), hashCode(), toString()
- Compact constructor syntax
- Can have static methods and instance methods
- Cannot extend other classes but can implement interfaces

Text Blocks (Finalized)

```
String json = """
    {
        "name": "John",
        "age": 30
    }
    """;
```

1.2 Memory Management and JVM Optimization

Memory Structure Understanding

- **Heap Memory**: Young Generation (Eden, S0, S1) and Old Generation
- **Stack Memory**: Thread-specific storage for local variables and method calls
- **Metaspace**: Replacement for PermGen, stores class metadata

Garbage Collection Algorithms

- G1GC (default in Java 17): Low-latency garbage collector
- ZGC: Ultra-low latency collector for large heaps
- Parallel GC: High-throughput collector

JVM Tuning Parameters

```
-Xms4g -Xmx8g              # Initial and maximum heap size
-XX:+UseG1GC               # Use G1 garbage collector
-XX:MaxGCPauseMillis=200   # Target pause time
-XX:+HeapDumpOnOutOfMemoryError  # Generate heap dump on OOM
```

Memory Leak Detection and Prevention

- Use of memory profilers (VisualVM, JProfiler)
- Identifying object retention patterns
- Monitoring garbage collection metrics
- Best practices for preventing leaks

1.3 Concurrency and Threading

Virtual Threads (Preview in Java 17, Stable in Java 21)

- Lightweight threads managed by JVM
- Better resource utilization for I/O-bound operations
- Integration with existing APIs

CompletableFuture Advanced Usage

```
CompletableFuture&lt;String&gt; future1 = CompletableFuture.supplyAsync(() -&gt; "Hello")
CompletableFuture&lt;String&gt; future2 = CompletableFuture.supplyAsync(() -&gt; "World")
CompletableFuture&lt;String&gt; combined = future1.thenCombine(future2, (s1, s2) -&gt; s1
```

Thread Safety and Concurrent Collections

- ConcurrentHashMap internals and performance characteristics
- BlockingQueues for producer-consumer patterns
- Atomic operations and lock-free programming

2. Build Automation with Gradle

2.1 Gradle Fundamentals

Project Structure and Build Scripts

- Understanding `build.gradle` vs `build.gradle.kts`
- Multi-project builds and composite builds
- Gradle wrapper usage and benefits

Dependency Management

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    testImplementation 'org.junit.jupiter:junit-jupiter'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
}
```

Custom Tasks and Plugins

```
task customTask {
    doLast {
        println 'Executing custom task'
    }
}

plugins {
    id 'java'
    id 'org.springframework.boot' version '3.0.0'
}
```

2.2 Advanced Gradle Concepts

Build Lifecycle and Phases

- Initialization phase
- Configuration phase
- Execution phase

Performance Optimization

- Build caching strategies
- Parallel builds configuration
- Incremental compilation
- Gradle daemon optimization

Integration with CI/CD

- Gradle wrapper in CI environments
- Build scan integration
- Publishing artifacts to repositories

3. Kubernetes Container Orchestration

3.1 Core Kubernetes Concepts

Architecture Components

- Master Node: API Server, etcd, Controller Manager, Scheduler
- Worker Nodes: kubelet, kube-proxy, Container Runtime
- Understanding cluster networking and service mesh

Pod Lifecycle and Management

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
  - name: app-container
    image: my-app:latest
    ports:
    - containerPort: 8080
    resources:
      limits:
        memory: "512Mi"
        cpu: "500m"
      requests:
        memory: "256Mi"
        cpu: "250m"
```

Services and Ingress

- ClusterIP, NodePort, LoadBalancer service types
- Ingress controllers for external traffic routing
- Service discovery and DNS resolution

3.2 Advanced Kubernetes Operations

Deployments and Rolling Updates

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app-deployment
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: my-app:v1.2.0
```

ConfigMaps and Secrets Management

- Externalization of configuration
- Secret rotation and security best practices
- Volume mounts vs environment variables

Monitoring and Observability

- Prometheus metrics collection
- Logging strategies with ELK stack
- Distributed tracing with Jaeger
- Health checks and readiness probes

4. Messaging Systems (RabbitMQ & Kafka)

4.1 RabbitMQ Deep Dive

Core Concepts

- Exchanges (Direct, Topic, Fanout, Headers)
- Queues and routing keys
- Bindings and message routing
- Virtual hosts for multi-tenancy

Message Patterns

```
// Producer
@Component
public class MessageProducer {
    @Autowired
    private RabbitTemplate rabbitTemplate;

    public void sendMessage(String message) {
        rabbitTemplate.convertAndSend("exchange.direct", "routing.key", message);
    }
}

// Consumer
@RabbitListener(queues = "queue.name")
public void receiveMessage(String message) {
    // Process message
}
```

Advanced Features

- Message persistence and durability
- Dead letter exchanges and retry mechanisms
- Priority queues and message TTL

- Clustering and high availability

4.2 Apache Kafka Mastery

Architecture and Components

- Brokers, topics, and partitions
- Producers, consumers, and consumer groups
- Zookeeper coordination (transitioning to KRaft)
- Replication and ISR (In-Sync Replicas)

Producer Configuration

```
Properties props = new Properties();
props.put("bootstrap.servers", "localhost:9092");
props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
props.put("acks", "all");
props.put("retries", 3);
props.put("enable.idempotence", true);
```

Consumer Patterns

- Auto-commit vs manual commit
- Consumer group rebalancing
- Offset management strategies
- Stream processing with Kafka Streams

Performance Tuning

- Batch size optimization
- Compression algorithms
- Partition assignment strategies
- Monitoring consumer lag

5. REST API Design and Development

5.1 RESTful API Design Principles

HTTP Methods and Status Codes

- GET, POST, PUT, PATCH, DELETE semantics
- Idempotency considerations
- Proper status code usage (200, 201, 204, 400, 401, 404, 500)

Resource Design Patterns

```
GET /api/v1/users              # Get all users
GET /api/v1/users/{id}         # Get specific user
POST /api/v1/users             # Create new user
PUT /api/v1/users/{id}         # Update entire user
PATCH /api/v1/users/{id}       # Partial update
DELETE /api/v1/users/{id}      # Delete user
```

API Versioning Strategies

- URI versioning: `/api/v1/resources`
- Header versioning: `Accept: application/vnd.api+json;version=1`
- Query parameter versioning: `/api/resources?version=1`

5.2 Implementation with Spring Boot

Controller Design

```
@RestController
@RequestMapping("/api/v1/users")
@Validated
public class UserController {

    @GetMapping
    public ResponseEntity&lt;Page&lt;UserDTO&gt;&gt; getUsers(
        @RequestParam(defaultValue = "0") int page,
        @RequestParam(defaultValue = "10") int size,
        @RequestParam(required = false) String search) {
        // Implementation
    }

    @PostMapping
    public ResponseEntity&lt;UserDTO&gt; createUser(
        @Valid @RequestBody CreateUserRequest request) {
        // Implementation
    }
}
```

Error Handling and Validation

- Global exception handlers with @ControllerAdvice
- Custom exception hierarchies
- Bean validation with JSR-303 annotations
- Structured error responses

Security Implementation

- Spring Security integration
- JWT token-based authentication
- OAuth2 and OIDC flows

- CORS configuration and CSRF protection

6. Database Management and SQL Optimization

6.1 SQL Performance Tuning

Query Optimization Techniques

```
-- Use indexes effectively
CREATE INDEX idx_user_email ON users(email);
CREATE INDEX idx_order_date_status ON orders(order_date, status);

-- Optimize JOIN operations
SELECT u.name, COUNT(o.id) as order_count
FROM users u
LEFT JOIN orders o ON u.id = o.user_id
WHERE u.created_date &gt; '2023-01-01'
GROUP BY u.id, u.name
HAVING COUNT(o.id) &gt; 5;
```

Execution Plan Analysis

- Understanding EXPLAIN output
- Identifying table scans vs index seeks
- Cost-based optimizer behavior
- Statistics maintenance

Index Strategy

- Clustered vs non-clustered indexes
- Composite index column order
- Covering indexes for query optimization
- Index maintenance and fragmentation

6.2 Advanced Database Concepts

Transaction Management

- ACID properties implementation
- Isolation levels (Read Uncommitted, Read Committed, Repeatable Read, Serializable)
- Deadlock detection and resolution
- Long-running transaction optimization

Connection Pooling

```
@Configuration
public class DatabaseConfig {
```

```
    @Bean
    @Primary
    public DataSource dataSource() {
        HikariConfig config = new HikariConfig();
        config.setDriverClassName("com.mysql.cj.jdbc.Driver");
        config.setJdbcUrl("jdbc:mysql://localhost:3306/mydb");
        config.setMaximumPoolSize(20);
        config.setMinimumIdle(5);
        config.setConnectionTimeout(30000);
        config.setIdleTimeout(600000);
        config.setLeakDetectionThreshold(60000);
        return new HikariDataSource(config);
    }
}
```

7. Infrastructure as Code with Terraform

7.1 Terraform Fundamentals

Core Concepts

- Providers and resources
- State management and remote backends
- Terraform workflow: init, plan, apply, destroy
- HCL (HashiCorp Configuration Language) syntax

Basic Resource Definition

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "web_server" {
  ami           = "ami-0c02fb55956c7d316"
  instance_type = "t3.micro"

  tags = {
    Name        = "WebServer"
    Environment = "Production"
  }
}
```

7.2 Advanced Terraform Practices

Modules and Composition

```
module "vpc" {
  source = "./modules/vpc"
```

```
  cidr_block           = "10.0.0.0/16"
  availability_zones   = ["us-east-1a", "us-east-1b"]
  enable_dns_hostnames = true
}

module "web_servers" {
  source = "./modules/ec2"

  instance_count = 2
  subnet_ids     = module.vpc.private_subnet_ids
  security_groups = [module.vpc.web_security_group_id]
}
```

State Management

- Remote state storage (S3, Azure Storage, GCS)
- State locking with DynamoDB
- Workspace management for multiple environments
- Import existing resources

Best Practices

- Resource naming conventions
- Environment separation strategies
- Secret management with external providers
- CI/CD integration patterns

8. CI/CD with GitHub Actions

8.1 GitHub Actions Fundamentals

Workflow Structure

```
name: CI/CD Pipeline

on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main ]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v3

    - name: Set up JDK 17
```

```
      uses: actions/setup-java@v3
      with:
        java-version: '17'
        distribution: 'temurin'

    - name: Cache Gradle dependencies
      uses: actions/cache@v3
      with:
        path: ~/.gradle/caches
        key: ${{ runner.os }}-gradle-${{ hashFiles('**/*.gradle*') }}

    - name: Run tests
      run: ./gradlew test

    - name: Generate test report
      uses: dorny/test-reporter@v1
      if: success() || failure()
      with:
        name: Gradle Tests
        path: '**/build/test-results/test/TEST-*.xml'
        reporter: java-junit
```

8.2 Advanced Pipeline Patterns

Multi-Stage Deployment

```
deploy:
  needs: test
  runs-on: ubuntu-latest
  if: github.ref == 'refs/heads/main'

  strategy:
    matrix:
      environment: [staging, production]

  steps:
  - name: Deploy to ${{ matrix.environment }}
    uses: azure/webapps-deploy@v2
    with:
      app-name: myapp-${{ matrix.environment }}
      publish-profile: ${{ secrets[format('AZURE_WEBAPP_PUBLISH_PROFILE_{0}', upper(matri
```

Security and Secret Management

- GitHub Secrets for sensitive data
- OIDC authentication with cloud providers
- Vulnerability scanning integration
- Container image security scanning

9. Performance Optimization and Debugging

9.1 Application Performance Monitoring

Spring Boot Actuator Integration

```
@RestController
public class HealthController {

    @Autowired
    private MeterRegistry meterRegistry;

    @GetMapping("/api/process")
    public ResponseEntity&lt;String&gt; processRequest() {
        Timer.Sample sample = Timer.start(meterRegistry);
        try {
            // Business logic
            return ResponseEntity.ok("Processed");
        } finally {
            sample.stop(Timer.builder("request.duration")
                .description("Request processing time")
                .register(meterRegistry));
        }
    }
}
```

Memory Leak Detection

- Heap dump analysis with Eclipse MAT
- JVisualVM profiling techniques
- GC log analysis patterns
- Memory usage monitoring strategies

9.2 Performance Tuning Strategies

JVM Optimization

```
# Production JVM settings
-server
-Xms4g -Xmx4g
-XX:+UseG1GC
-XX:MaxGCPauseMillis=200
-XX:+UnlockExperimentalVMOptions
-XX:+UseStringDeduplication
-XX:+DisableExplicitGC
-Djava.awt.headless=true
```

Database Performance

- Connection pool tuning

- Query result caching strategies
- Read replica configurations
- Database connection monitoring

Caching Strategies

- Application-level caching with Caffeine
- Distributed caching with Redis
- HTTP caching headers
- CDN integration patterns

10. Security Best Practices and Coding Standards

10.1 Secure Coding Practices

Input Validation and Sanitization

```
@Component
public class SecurityUtils {

    private static final Pattern SAFE_STRING = Pattern.compile("^[a-zA-Z0-9\\s\\-_]+$");

    public String sanitizeInput(String input) {
        if (input == null || input.trim().isEmpty()) {
            return "";
        }

        String trimmed = input.trim();
        if (!SAFE_STRING.matcher(trimmed).matches()) {
            throw new InvalidInputException("Input contains invalid characters");
        }

        return trimmed;
    }
}
```

SQL Injection Prevention

```
// Use parameterized queries
@Repository
public class UserRepository {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    public User findByEmail(String email) {
        String sql = "SELECT * FROM users WHERE email = ?";
        return jdbcTemplate.queryForObject(sql, new Object[]{email}, new UserRowMapper())
```

```
    }
}
```

10.2 Security Framework Integration

Spring Security Configuration

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .csrf().csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse())
            .and()
            .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            .and()
            .authorizeHttpRequests(authz -&gt; authz
                .requestMatchers("/api/public/**").permitAll()
                .requestMatchers("/api/admin/**").hasRole("ADMIN")
                .anyRequest().authenticated()
            )
            .oauth2ResourceServer().jwt();

        return http.build();
    }
}
```

OWASP Top 10 Mitigation

- Broken Access Control prevention
- Cryptographic failures mitigation
- Injection attack prevention
- Insecure design patterns avoidance
- Security misconfiguration hardening

11. Interview Preparation Strategy

11.1 Technical Interview Format

System Design Questions

- Design a URL shortening service (like bit.ly)
- Design a real-time chat application
- Design a distributed cache system
- Design a microservices architecture for e-commerce

Coding Challenges Focus Areas

- Data structures and algorithms
- Concurrent programming scenarios
- Design pattern implementations
- Database optimization problems

11.2 Behavioral Interview Preparation

STAR Method Examples

- **Situation**: Describe a challenging debugging scenario
- **Task**: Explain your responsibility in resolving the issue
- **Action**: Detail the steps you took to identify and fix the problem
- **Result**: Quantify the impact of your solution

Leadership and Collaboration

- Cross-team collaboration experiences
- Mentoring junior developers
- Technical decision-making processes
- Conflict resolution in technical discussions

12. Hands-On Practice Projects

12.1 Microservices Project

Project Structure

```
ecommerce-platform/
├── api-gateway/
├── user-service/
├── product-service/
├── order-service/
├── payment-service/
├── notification-service/
├── docker-compose.yml
├── kubernetes/
└── terraform/
```

Implementation Requirements

- Spring Boot 3.x with Java 17
- PostgreSQL with connection pooling
- Redis for caching and sessions

- RabbitMQ for async messaging
- Kubernetes deployment manifests
- GitHub Actions CI/CD pipeline

12.2 Performance Optimization Project

Scenario: Legacy monolithic application migration

- Profile existing application bottlenecks
- Implement caching strategies
- Optimize database queries
- Extract microservices gradually
- Monitor performance improvements

13. Study Schedule and Timeline

Phase 1: Foundation (Weeks 1-2)

- **Week 1**: Java 17 features, memory management, concurrency
- **Week 2**: Gradle build automation, project setup

Phase 2: Infrastructure (Weeks 3-4)

- **Week 3**: Kubernetes concepts, container orchestration
- **Week 4**: Terraform infrastructure as code

Phase 3: Integration (Weeks 5-6)

- **Week 5**: Messaging systems (RabbitMQ, Kafka)
- **Week 6**: REST API design and implementation

Phase 4: Optimization (Weeks 7-8)

- **Week 7**: Database optimization, SQL tuning
- **Week 8**: Performance monitoring and debugging

Phase 5: DevOps (Weeks 9-10)

- **Week 9**: CI/CD pipelines with GitHub Actions
- **Week 10**: Security best practices, coding standards

Phase 6: Practice (Weeks 11-12)

- **Week 11**: Hands-on project implementation
- **Week 12**: Mock interviews and system design practice

14. Resources and References

Official Documentation

- Java 17 Documentation
- Spring Boot Reference Guide
- Kubernetes Documentation
- Terraform Documentation

Books and Learning Materials

- "Effective Java" by Joshua Bloch (3rd Edition)
- "Spring in Action" by Craig Walls (6th Edition)
- "Kubernetes in Action" by Marko Lukša
- "Building Microservices" by Sam Newman

Practice Platforms

- LeetCode for algorithmic challenges
- HackerRank for domain-specific problems
- GitHub for project hosting and CI/CD practice
- Docker Hub for container image management

Monitoring and Observability Tools

- Prometheus and Grafana for metrics
- ELK Stack for logging
- Jaeger for distributed tracing
- New Relic or DataDog for APM

15. Success Metrics and Evaluation

Technical Competency Checkpoints

- [ ] Can implement Java 17 features in real projects
- [ ] Can set up and optimize Gradle build processes
- [ ] Can deploy applications to Kubernetes clusters
- [ ] Can design and implement REST APIs following best practices
- [ ] Can optimize database queries and performance
- [ ] Can set up comprehensive CI/CD pipelines
- [ ] Can implement security best practices throughout the stack

Mock Interview Performance

- [ ] System design discussions (60+ minutes)
- [ ] Live coding sessions (45+ minutes)
- [ ] Technical deep-dive conversations
- [ ] Behavioral interview scenarios

Project Portfolio

- [ ] Complete microservices application
- [ ] Infrastructure as Code examples
- [ ] Performance optimization case studies
- [ ] Open source contributions

Conclusion

This comprehensive preparation guide provides a structured pathway to master all the technical skills required for the Senior Software Engineer position. The key to success lies in consistent practice, hands-on implementation, and continuous learning. Focus on understanding not just the "how" but also the "why" behind each technology and practice.

Remember that interviews are not just about technical knowledge—they're about demonstrating problem-solving abilities, communication skills, and the capacity to work effectively in a team environment. Practice explaining complex concepts clearly and concisely, as this skill is crucial for senior-level positions.

Good luck with your interview preparation! With dedicated effort and systematic study using this guide, you'll be well-equipped to excel in your Senior Software Engineer interview and secure your dream position.

This guide serves as a comprehensive roadmap. Adapt the timeline and focus areas based on your current skill level and the specific requirements of the position you're targeting.

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29]

[30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80]

⁂

1. https://www.withoutbook.com/InterviewQuestionList.php?tech=93&dl=Top&s=Java+17+Interview+Questions+and+Answers
2. https://javatechonline.com/java-17-interview-questions-answers-explained/
3. https://www.linkedin.com/posts/manju-vasanth_testing-qa-maven-activity-7278648687902257152-buAA
4. https://www.turing.com/interview-questions/kubernetes
5. https://www.geeksforgeeks.org/java/features-of-java-17/
6. https://devskiller.com/coding-tests-skill/gradle/
7. https://www.interviewbit.com/kubernetes-interview-questions/
8. https://www.interviewbit.com/java-interview-questions/
9. https://www.devopsschool.com/blog/gradle-related-faqs/
10. https://www.simplilearn.com/tutorials/kubernetes-tutorial/kubernetes-interview-questions
11. https://www.geeksforgeeks.org/java/java-interview-questions/
12. https://cloudfoundation.com/blog/gradle-interview-questions-answers/
13. https://www.remoterocketship.com/advice/10-gradle-interview-questions-and-answers-in-2023
14. https://www.simplilearn.com/kafka-interview-questions-and-answers-article
15. https://getsdeready.com/top-20-api-design-questions-for-system-design-interviews/
16. https://codesignal.com/blog/interview-prep/28-sql-interview-questions-and-answers-from-beginner-to-senior-level/
17. https://www.interviewbit.com/kafka-interview-questions/
18. https://www.simplilearn.com/rest-api-interview-questions-answers-article
19. https://www.designgurus.io/answers/detail/what-are-sql-query-optimization-interview-questions
20. https://mindmajix.com/rabbitmq-interview-questions
21. https://www.interviewbit.com/rest-api-interview-questions/
22. https://www.linkedin.com/posts/aditya-chandak-24692825_sql-performance-tuning-interview-questions-activity-7295990605426364416-7l4w
23. https://www.reddit.com/r/kubernetes/comments/1hwos4i/what_are_some_good_interviews_questions_asked_for/
24. https://www.alibabacloud.com/blog/interview-questions-weve-learned-over-the-years-kafka_601108
25. https://www.reddit.com/r/ExperiencedDevs/comments/1kkuy4b/rest_api_design_interview_question/
26. https://techbeamers.com/sql-performance-interview-questions-answers/
27. https://www.acte.in/rabbitmq-interview-questions-and-answers
28. https://github.com/Devinterview-io/api-design-interview-questions
29. https://www.dbvis.com/thetable/top-sql-performance-tuning-interview-questions-and-answers/
30. https://www.hellointerview.com/learn/system-design/deep-dives/kafka
31. https://www.merge.dev/blog/rest-api-interview-questions

32. https://www.remotely.works/blog/top-advanced-sql-interview-questions-and-answers
33. https://gist.github.com/bansalankit92/9414ef3614229cdca6053464fedf5038
34. https://www.guvi.in/blog/top-java-9-to-java-17-interview-questions/
35. https://www.youtube.com/watch?v=DQ57zYedMdQ
36. https://www.javacodegeeks.com/2024/09/java-memory-management-key-interview-questions-and-expert-answers.html
37. https://digma.ai/how-to-use-spring-boot-profiling-tools/
38. https://www.simplilearn.com/terraform-interview-questions-and-answers-article
39. https://www.buggybread.com/2014/04/java-interview-questions-and-answers-on_706.html
40. https://www.cogentuniversity.com/post/spring-boot-performance-tuning-5-common-issues-and-how-to-fix-them
41. https://www.geeksforgeeks.org/devops/terraform-interview-questions/
42. https://mentorcruise.com/questions/java-developer/
43. https://stackoverflow.com/questions/73834466/spring-boot-project-run-on-debug-mode-takes-more-than-2-hours
44. https://www.interviewbit.com/terraform-interview-questions/
45. https://mindmajix.com/gradle-interview-questions
46. https://www.geeksforgeeks.org/java/java-memory-management/
47. https://www.youtube.com/watch?v=XWoivKnqsyo
48. https://razorops.com/blog/top-50-terraform-interview-question-and-answers/
49. https://www.baeldung.com/java-memory-management-interview-questions
50. https://blog.jetbrains.com/idea/2025/06/demystifying-spring-boot-with-spring-debugger/
51. https://www.multisoftsystems.com/interview-questions/terraform-interview-questions-answers
52. https://interview.in28minutes.com/interview-guides/java/java-memory-management/
53. https://blog.ycrash.io/java-springboot-performance-analysis-and-tuning/
54. https://k21academy.com/terraform-iac/terraform-interview-questions/
55. https://www.youtube.com/watch?v=vz6vSZRuS2M
56. https://www.linkedin.com/pulse/top-10-kubernetes-interview-questions-answers-sre-roles-gupta-71wgf
57. https://docs.redhat.com/en/documentation/red_hat_support_for_spring_boot/2.1/html/spring_boot_2.1.x_runtime_guide/debugging-springboot-based-application_spring-boot
58. https://www.finalroundai.com/blog/ci-cd-interview-questions
59. https://agilemania.com/software-developer-interview-questions
60. https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-en/02-checklist/05-checklist
61. https://www.linkedin.com/posts/prerana-moon-a7b8a925b_github-actions-real-time-interview-question-activity-7246895663999827969-eAmA
62. https://stackoverflow.blog/2022/05/23/the-science-of-interviewing-developers/
63. https://www.wiz.io/academy/secure-coding-best-practices
64. https://www.fosstechnix.com/github-actions-interview-questions-and-answers/

65. https://www.linkedin.com/pulse/interview-questions-what-practices-software-teams-lead-heilmann-
66. https://kirkpatrickprice.com/blog/secure-coding-best-practices/
67. https://www.youtube.com/watch?v=HzzYTgkaWv4
68. https://razorops.com/blog/top-50-github-actions-interview-question-and-answers/
69. https://www.techinterviewhandbook.org/software-engineering-interview-guide/
70. https://www.appknox.com/cyber-security-jargons/secure-coding
71. https://hellointern.in/blog/github-actions-interview-questions-and-answers-for-experienced-64115
72. https://www.amazon.jobs/software-development-topics
73. https://www.kiuwan.com/blog/secure-coding-guidelines/
74. https://www.interviewbit.com/ci-cd-interview-questions/
75. https://pesto.tech/resources/software-developer-job-interview-tips
76. https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/
77. https://github.com/rohitg00/devops-interview-questions
78. https://www.scmgalaxy.com/tutorials/top-50-gradle-interview-questions-with-answers/
79. https://www.adaface.com/blog/software-development-interview-questions/
80. https://www.geeksforgeeks.org/devops/kubernetes-interview-questions/