✅ SonarQube – Complete Guide

🔷 What is SonarQube?

SonarQube is an **open-source platform** used for **continuous inspection of code quality**.
It checks your source code for **bugs, code smells, vulnerabilities, security issues, duplications**, and more.

🔹 Why is SonarQube used?

Purpose	Explanation
Code Quality	Detects issues like bad code, duplicated logic, potential bugs
Security	Identifies vulnerabilities and code injection risks
Maintainability	Ensures code is readable, modular, and future-proof
CI/CD Integration	Automatically checks code with Jenkins, GitHub, etc.

🔧 How SonarQube Works (Behind the Scenes)

1. **Source Code Analyzed**: Your code is scanned using the SonarQube Scanner.
2. **Rules Engine**: SonarQube applies a set of rules (default or custom) to the code.
3. **Metrics Generated**:
    - Bugs
    - Vulnerabilities
    - Code Smells
    - Coverage (test cases)
    - Duplications
4. **Dashboard**: Displays detailed reports in the web UI.
5. **Quality Gate**: If the code doesn't meet a quality threshold, it can block the build in CI/CD.

🚀 Real-Time Use Case

In a **Java Spring Boot project**, you integrate SonarQube with **Jenkins pipeline** or use it with **GitHub Actions**.
Every time code is pushed to GitHub:

- SonarQube scans the code.
- Sends feedback to the developer.
- Build fails if critical issues are found.

✅ Ensures cleaner code
✅ Prevents code rot
✅ Encourages best practices

🧪 SonarQube Components

Component	**Description**
SonarQube Server	Hosts the UI and analysis engine
Sonar Scanner	CLI tool that performs the analysis
Sonar Plugins	Adds support for other languages or integrations
Quality Gate	Set of conditions your code must meet (pass/fail)

🔧 How to Set Up SonarQube (Locally)

Step 1: Download SonarQube

- Go to https://www.sonarsource.com/products/sonarqube/
- Extract it, and start the server:

bash

CopyEdit

```
./bin/macosx-universal-64/sonar.sh start  (Mac)

./bin/windows-x86-64/StartSonar.bat     (Windows)
```

Step 2: Access Dashboard

- Open: http://localhost:9000

- Default login:
    - Username: admin
    - Password: admin

🔧 SonarQube in Maven Project (Example)

Add Plugin to pom.xml

xml

CopyEdit

```
<plugin>
  <groupId>org.sonarsource.scanner.maven</groupId>
  <artifactId>sonar-maven-plugin</artifactId>
  <version>3.11.0.3922</version>
</plugin>
```

Run Scan

bash

CopyEdit

```
mvn clean verify sonar:sonar \
  -Dsonar.projectKey=myapp \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.login=your_token
```

💡 Important Metrics

Metric	Meaning
Bugs	Defects that could lead to incorrect behavior
Vulnerabilities	Potential security risks
Code Smells	Maintainability issues
Coverage	% of code tested with unit tests

Metric	Meaning
Duplications	Copy-pasted code blocks
Cyclomatic Complexity	Complexity of decision paths in code

✅ Benefits of SonarQube

- Improves code quality and readability
- Helps enforce coding standards
- Prevents bugs early in development
- Integrates with GitHub, Jenkins, Bitbucket, GitLab
- Supports multiple languages (Java, JS, Python, etc.)

❌ Drawbacks

Limitation	Description
Requires setup	Needs server and scanner setup
Resource intensive	May slow down builds slightly
May need customization	Rules can be noisy if not fine-tuned

🤔 Interview Questions

1. What is SonarQube?
2. How do you integrate SonarQube with a Java Maven project?
3. What are code smells?
4. What is a Quality Gate in SonarQube?
5. How does SonarQube help in CI/CD pipelines?
6. Can SonarQube block code deployment? How?
7. How do you generate a Sonar token?
8. What are the key metrics SonarQube tracks?

✅ Prometheus and Grafana – Complete Guide

🔷 What is Prometheus?

Prometheus is an open-source **monitoring and alerting tool**. It collects and stores metrics data (like CPU usage, memory usage, HTTP requests, etc.) in **time-series format**.

🔹 Why use Prometheus?

Purpose	Explanation
Monitoring	Continuously tracks system/app performance
Time-Series Data	Stores data over time for trend analysis
Alerting	Sends alerts when something goes wrong (via AlertManager)
Scalable	Handles high-volume metrics efficiently

🔧 How Prometheus Works

1. **Targets**: Applications expose metrics at endpoints like /metrics
2. **Scraping**: Prometheus fetches metrics (pull-based) at intervals
3. **Storage**: Saves data in its own TSDB (Time Series DB)
4. **Querying**: You use **PromQL** to query metrics
5. **Alerting**: Sends alerts when rules break (e.g., CPU > 90%)

🔷 What is Grafana?

Grafana is a data visualization tool that works perfectly with Prometheus and others (InfluxDB, Elasticsearch, etc.).

🔹 Why use Grafana?

Purpose	Explanation
Visual Dashboards	Beautiful charts, graphs, and alerts
Multi-source support	Works with Prometheus, MySQL, etc.
Alerting UI	Visually manage alert rules
Templating	Reusable and dynamic dashboards

🛠 Real Use Case

In a Spring Boot microservices project deployed on Docker/Kubernetes:

- Prometheus scrapes metrics like:
    - Request count
    - Latency
    - JVM memory
- Grafana visualizes these metrics
- Alerts are sent when thresholds cross

✅ Helps DevOps + Developers catch problems early.

🔧 Set Up Prometheus + Grafana (Locally with Docker)

yaml

CopyEdit

```
# docker-compose.yml
version: '3'
services:
 prometheus:
  image: prom/prometheus
  volumes:
   - ./prometheus.yml:/etc/prometheus/prometheus.yml
  ports:
    - "9090:9090"

  grafana:
    image: grafana/grafana
    ports:
      - "3000:3000"
```

Create prometheus.yml:

yaml

CopyEdit

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'spring-app'
    static_configs:
      - targets: ['localhost:8080']
```

📈 Grafana Basics

- Access via http://localhost:3000
- Login: admin / admin
- Add Prometheus as a data source
- Import pre-built dashboards or create your own
- Set alerts for CPU, Memory, HTTP latency, etc.

🧠 Prometheus vs Grafana

Feature	**Prometheus**	**Grafana**
Purpose	Metric collection & storage	Visualization

Feature	Prometheus	Grafana
Data source	Pulls from /metrics	Reads from Prometheus & others
Query Language	PromQL	Uses data source's language
Alerting	Yes (via AlertManager)	Yes (basic UI level)

❗ Drawbacks

Tool	Drawback
Prometheus	No long-term storage by default
Grafana	Visualization only – no data collection

🤔 Interview Questions

1. What is Prometheus and how does it work?
2. How does Grafana integrate with Prometheus?
3. What is a time-series database?
4. What is PromQL?
5. What are exporters in Prometheus?
6. How do you configure alerting rules?
7. What port does Prometheus/Grafana run on?
8. Difference between push vs pull in monitoring?

✅ ELK Stack – Complete Guide

🔷 What is ELK?

ELK Stack stands for:

- **E**lasticsearch (search engine)
- **L**ogstash (log collector/transformer)
- **K**ibana (dashboard/visualization)

🔹 Why Use ELK?

Purpose	**Explanation**
Centralized Logging	Collect logs from multiple services in one place
Searchable Logs	Fast full-text search using Elasticsearch
Visual Dashboards	Kibana for viewing, filtering logs
Alerting & Monitoring	Detect issues using log patterns

🔧 How ELK Stack Works

1. **Logstash** reads logs from files, Kafka, etc.
2. **Filters/Parses** logs (e.g., JSON, regex)
3. **Elasticsearch** indexes and stores logs
4. **Kibana** visualizes them

💻 Example Setup (Docker Compose)

yaml

CopyEdit

```yaml
version: '3'
services:
 elasticsearch:
  image: elasticsearch:7.17.10
  environment:
   - discovery.type=single-node
  ports:
   - "9200:9200"

 logstash:
    image: logstash:7.17.10
    volumes:
      - ./logstash.conf:/usr/share/logstash/pipeline/logstash.conf
    ports:
      - "5000:5000"

  kibana:
    image: kibana:7.17.10
    ports:
      - "5601:5601"
```

Sample logstash.conf:

conf

CopyEdit

```
input {
  tcp {
    port => 5000
    codec => json
  }
}
output {
  elasticsearch {
    hosts => ["http://elasticsearch:9200"]
    index => "spring-logs"
  }
}
```

✅ Real-Time Use Case

In microservices:

- Logs are pushed from each service
- Logstash collects and processes them
- Elasticsearch indexes for fast search
- Kibana shows:
    - HTTP errors
    - DB exceptions
    - Audit logs

🔍 ELK vs Prometheus

Feature	**ELK**	**Prometheus**
Data Type	Logs	Metrics
Storage	Elasticsearch	Time-Series DB
Visualization	Kibana	Grafana
Use Case	Debugging, Trace Logs	Monitoring, Alerts

❗ Drawbacks of ELK

- High memory usage
- Complex setup
- Learning curve (especially Logstash)

🧠 Interview Questions

1. What is ELK stack?
2. What's the difference between Logstash and Beats?
3. How do you visualize logs in Kibana?
4. How does Elasticsearch store data?
5. ELK vs Prometheus?
6. How to configure log pipelines in Logstash?

7. What is the role of Kibana?

✅ Terraform, Ansible, and Splunk – Complete Guide (for Full Stack Java Developers)

🧱 1. Terraform – Infrastructure as Code (IaC)

🔹 What is Terraform?

Terraform is an open-source tool by HashiCorp that allows you to **automate infrastructure provisioning** (e.g., AWS, Azure, GCP, Docker) using **code**.

✅ Think of it like: *writing code to create servers, databases, load balancers, etc.*

🔹 Why Use Terraform?

Feature	Benefit
Infrastructure as Code	Easily manage, version, and reuse infrastructure
Multi-cloud support	Works with AWS, Azure, GCP, Docker, Kubernetes, etc.
Automation	Automatically provisions and tears down infra
Idempotent	Runs safely multiple times without duplication

🔧 How Terraform Works

1. You write **.tf files** (Terraform config files).
2. Run terraform init → initializes plugins.
3. Run terraform plan → shows what will be created.
4. Run terraform apply → creates resources.
5. Run terraform destroy → deletes all resources.

📝 Example Code (for AWS EC2)

hcl

CopyEdit

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

resource "aws_instance" "example" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
}
```

🧠 Terraform Interview Questions

1. What is Infrastructure as Code?
2. What is a Terraform Provider?
3. Difference between terraform plan and terraform apply?
4. How do you manage state in Terraform?
5. What is a Terraform module?
6. Can Terraform be used with Kubernetes?

❗ Drawbacks

- Terraform state files must be managed carefully
- Syntax and debugging errors can be tricky

🔧 Real Use Case

- Provision 10 EC2 + RDS DB + S3 bucket via one command
- Track all infra changes in GitHub (like code)

⚙️ 2. Ansible – Configuration Management

🔹 What is Ansible?

Ansible is an **agentless** open-source tool used for:

- **Automating server configuration**
- **Application deployments**
- **Installing software (like Java, MySQL)**

🔹 Why Use Ansible?

Feature	Benefit
No agents	Only needs SSH access
YAML-based	Easy-to-read playbooks
Idempotent	Safe to run multiple times
Fast deployment	Great for configuring 10s/100s of servers

🛠 How It Works

1. Create a hosts file to define servers
2. Write a **playbook** (YAML file)
3. Run ansible-playbook playbook.yml

📝 Sample Playbook

yaml

CopyEdit

```
- name: Install NGINX on Ubuntu
  hosts: webservers
  become: yes
```

```
tasks:
 - name: Install NGINX
   apt:
     name: nginx
     state: present
```

🧠 Ansible Interview Questions

1. What is Ansible and how does it work?
2. What is a Playbook vs Role?
3. What is the inventory file?
4. How do you ensure idempotency in Ansible?
5. Difference between Ansible and Terraform?

❗ Drawbacks

- Not suitable for complex infrastructure creation (Terraform is better for that)
- Slower on large scale compared to compiled tools

🔍 Real Use Case

- Deploy a Spring Boot JAR to 5 Linux servers
- Configure NGINX, MySQL, Java on all in one go

📊 3. Splunk – Log Management and Monitoring

🔹 What is Splunk?

Splunk is a powerful tool for **log aggregation, monitoring, and real-time analysis**. It can collect logs from all apps and infra, store, analyze, and alert.

🔹 Why Use Splunk?

Feature	Benefit
Centralized Logging	Collect logs from microservices, databases, etc.
Search & Analyze	Easily search huge volumes of logs
Visual Dashboards	Charts, alerts, and reports
Security Alerts	Real-time anomaly detection

🛠 How Splunk Works

1. Logs are sent from apps (via Splunk agent or HTTP)
2. Stored in **indexes**
3. Queried using **SPL (Search Processing Language)**
4. Dashboards and alerts are created

📝 Sample SPL Query

sql

CopyEdit

```
index=app_logs "Exception" | stats count by source
```

🧠 Splunk Interview Questions

1. What is Splunk and its components?
2. What is SPL in Splunk?
3. How does Splunk differ from ELK?
4. How do you create alerts in Splunk?
5. Real-life use case of Splunk in production?

❗ Drawbacks

- Expensive for high data volume

- Complex licensing and configuration

🛠 Real Use Case in Projects

- Your Java app logs exceptions to Splunk
- You create dashboards to monitor:
    - Error rate
    - Response time
    - DB timeouts
- Alert triggers if errors > 100/min

📌 Summary Table

Tool	Use Case	Format	Competitor
Terraform	Infra provisioning	HCL (.tf)	Pulumi
Ansible	Server config + software install	YAML	Chef, Puppet
Splunk	Logs and monitoring	SPL (Search Query)	ELK

✅ 1. Nexus Repository Manager

🔷 What is Nexus?

Nexus is a **repository manager** used to **store and manage build artifacts** like .jar, .war, .zip, Docker images, etc.

Think of it as a **private Maven Central** for your team or company.

🔹 Why Use Nexus?

Use	**Explanation**
Store Artifacts	Stores JARs, WARs created by Maven/Gradle
Centralized Repository	All teams fetch libraries from one place
Security	Control access and versions
Integration	Works with Maven, Jenkins, Docker, etc.

🔧 Real-Time Usage in Java Projects

1. Developer commits code → Jenkins builds project
2. Jenkins generates .jar or .war
3. Jenkins uploads it to **Nexus**
4. Deployment teams download the artifact from Nexus and deploy

---

🔧 Nexus Repositories Types

- **Hosted** → For internal artifacts
- **Proxy** → Caches external repositories like Maven Central
- **Group** → Combines hosted + proxy into one virtual repo

🧠 Interview Questions

1. What is Nexus and why do we use it?

2. How does Nexus work with Maven or Jenkins?
3. What are hosted and proxy repositories?
4. How do you upload/download artifacts manually?
5. Difference between Nexus, Artifactory, and Maven Central?

📦 Nexus URL in Maven

xml

CopyEdit

```
<repositories>
 <repository>
  <id>nexus</id>
  <url>http://localhost:8081/repository/maven-releases/</url>
 </repository>
</repositories>
```

✅ 2. JIRA

🔷 What is JIRA?

JIRA is a tool used for:

- **Agile Project Management**
- **Bug tracking**
- **Sprint planning**
- **Task assignments**

Used by developers, testers, project managers, and DevOps.

🔹 JIRA Boards

- **Kanban Board** → Continuous flow (good for support teams)
- **Scrum Board** → For sprint-based planning (used in Agile teams)

🔧 How JIRA Works in Projects

1. Product Owner creates **epics**, **user stories**, and **tasks**
2. Dev Team pulls tasks in sprint planning
3. Tracks task progress: TO DO → IN PROGRESS → DONE
4. QA adds bugs and tracks testing

🧠 Common JIRA Terms

Term	Meaning
Epic	Big feature or requirement
Story	Small functionality part of an epic
Task	Work assigned to dev/test
Bug	Issue found in functionality
Sprint	1–2 week cycle to complete stories

🧠 Interview Questions

1. What is JIRA used for?
2. What is the difference between a story and a task?
3. Explain how you used JIRA in your team.
4. What is a sprint? How long should it be?
5. How do you handle bug tracking and sprint planning in JIRA?

✅ 3. Agile Methodology (Scrum Model)

🔷 What is Agile?

Agile is a software development approach focused on:

- **Quick delivery**

- **Customer feedback**
- **Iterative progress**
- **Team collaboration**

Widely used in product-based and service-based companies.

🔷 Agile Ceremonies

Ceremony	**Purpose**
Sprint Planning	Choose stories for next 2-week sprint
Daily Standup	Short meeting on progress, blockers
Sprint Review	Demo to stakeholders
Sprint Retrospective	Feedback on what went well or needs improvement

---

🔧 Agile Roles

Role	**Responsibility**
Product Owner	Defines requirements and priorities
Scrum Master	Facilitates scrum process, removes blockers
Development Team	Delivers the product increment

📊 Sprint Example

- Sprint length: 2 weeks
- Team picks 10 stories (each with story points)
- Team works → updates JIRA → delivers working code
- Reviews with stakeholders → repeats in next sprint

🧠 Interview Questions

1. What is Agile and how is it different from Waterfall?

2. Explain your daily routine in an Agile team.
3. What is the purpose of a standup meeting?
4. Who assigns tasks in Agile – the manager or the team?
5. What tools did you use for Agile (JIRA, Confluence, etc.)?

✅ Summary Table

Tool	**Purpose**	**Used For**
Nexus	Artifact repository	Upload/download .jar, .war from builds
JIRA	Task/bug tracking	Managing sprints and stories
Agile	Development methodology	Fast delivery, continuous improvement