Springboot microservices guide

Below is **SET D (Questions 76–100) — FULL EXPANDED LONG ANSWERS**, completing your **100 Questions Spring Boot Microservices Interview Guide**.

✅ SET D — OPERATIONS, SECURITY, DEPLOYMENTS, LOGGING, CLOUD, CI/CD, SCALING (76–100)

Full Long-Form Answers

---

76. Explain Zero-Downtime Deployment in microservices.

Zero-downtime deployment ensures the system remains **fully available** during deployments.

Techniques

1️⃣ Blue-Green Deployment

You maintain two production environments:

Environment	Purpose
Blue	live traffic
Green	new version deployed

Once green passes health checks → switch traffic to green instantly.

✔️ Benefits

- Fast rollback
- Zero downtime
- Safe testing in production environment

2️⃣ Rolling Deployment

Deploy services instance by instance:

1. Stop 1 instance
2. Deploy new version
3. Test
4. Repeat for next instance

✔️ No full system downtime
✔️ Works well in Kubernetes

3️⃣ Canary Deployment

Release new version to **small % of users**.

Example:

- 5% traffic → new version
- 95% → old version

If errors increase → rollback.

4️⃣ Kubernetes deployment strategies

- `RollingUpdate` (default)
- `Recreate`
- `BlueGreen` (via ArgoCD or Spinnaker)
- `Canary` (via Istio)

Zero Downtime Requires:

✔️ health checks
✔️ backward compatibility
✔️ database migrations with versioning
✔️ externalized config
✔️ stateless containers

---

77. What is API Rate Limiting? How to implement it?

Rate limiting protects services from:

- DDoS
- brute force
- accidental high traffic
- billing explosions
- bad clients

Techniques

1. **Token Bucket** (most common)
2. **Fixed Window**
3. **Sliding Window Log**
4. **Leaky Bucket**

Implement with Spring Cloud Gateway using Redis

```yaml
spring:
  cloud:
    gateway:
      routes:
        - id: product-service
          uri: lb://product-service
          predicates:
            - Path=/products/**
          filters:
            - name: RequestRateLimiter
              args:
                redis-rate-limiter.replenishRate: 10
                redis-rate-limiter.burstCapacity: 20
```

✔️ Each user allowed 10 requests/sec.

Real E-commerce Use Case

- limit add-to-cart API
- throttle payment attempts
- restrict search spam

---

78. Explain Blue-Green vs Canary Deployments (with scenarios).

Feature	Blue-Green	Canary
Traffic Switch	100% immediate	gradual 1% → 5% → 50%
Rollback	instant	slow but monitored
Ideal For	major releases	experimental changes
Risk Level	low	lowest

Scenario: Payment Service

Blue-Green:

You deploy version 2 in Green
After full testing → switch 100% traffic
Rollback → switch back to Blue

Canary:

You release to 2% of users
Monitor payments
If errors increase → auto rollback

---

79. What is Horizontal vs Vertical Scaling?

Vertical Scaling (Scale Up)

Increase machine size:

- more CPU
- more RAM
- more storage

❌ expensive
❌ downtime
✔️ simpler

Horizontal Scaling (Scale Out)

Add more instances of service.

✔️ no downtime
✔️ cloud-native
✔️ auto-scaling support
✔️ load balanced

E-commerce usage

- inventory search → horizontal
- Redis cache → vertical
- Kafka → horizontal partitions

---

80. What is Infrastructure as Code (IaC)? Tools? Why needed?

IaC manages infrastructure using **code instead of manual configuration**.

Popular IaC Tools

- Terraform
- AWS CloudFormation
- Azure ARM
- Pulumi

- Ansible

Benefits

✔️ consistent
✔️ version controlled
✔️ repeatable
✔️ automation in CI/CD
✔️ minimized human errors

Example Terraform Snippet

```hcl
resource "aws_instance" "app" {
  ami = "ami-x"
  instance_type = "t2.micro"
}
```

---

81. Explain how secrets are managed in microservices.

Sensitive data:

- passwords
- API keys
- database credentials
- tokens

Must never be stored in:
❌ code
❌ GitHub
❌ config files

Secure Methods

1. Vault (HashiCorp)

Centralized secrets manager.

2. Kubernetes Secrets

Encoded + access restricted.

3. AWS Secrets Manager

Auto-rotating secrets.

4. Azure Key Vault

5. Environment Variables

Spring Boot example

```yaml
spring:
  datasource:
    password: ${DB_PASSWORD}
```

Secrets injected at runtime.

---

82. What is Sidecar Pattern? Example with Istio?

Sidecar pattern runs helper services beside main application container.

Examples

- logging agents
- Envoy proxy for service mesh
- monitoring modules
- security modules

Istio Example

Each microservice pod has:

- main container (app)
- sidecar container (Envoy proxy)

Sidecar handles:
✔️ mTLS
✔️ tracing
✔️ routing
✔️ retry logic

Application remains clean.

---

83. What is a Service Mesh? Why is it needed?

Service Mesh is a dedicated **infrastructure layer** to manage microservice communication.

Popular Tools

- Istio
- Linkerd
- Consul Connect

Features

✔️ traffic routing
✔️ mTLS encryption
✔️ retries/circuit breakers
✔️ observability
✔️ load balancing
✔️ rate limiting

Why needed?

Without service mesh, application code contains:

- retry
- load balancing
- logging
- security

Service mesh moves these responsibilities out of code.

---

84. Explain mTLS (Mutual TLS) in microservices.

mTLS ensures:

- client verifies server
- server verifies client

Both exchange certificates.

Why Important?

Prevents:

- impersonation
- MITM attacks
- unauthorized microservice calls

Used by:

- Istio
- Linkerd
- AWS Mesh

---

85. What is Distributed Configuration? Explain Spring Cloud Config.

Distributed config stores config centrally.

Why?

- consistent values across services
- dynamic reloading
- version control
- no rebuild required for config changes

Spring Cloud Config

Stores config in:

- Git
- Vault
- AWS S3
- Database

Usage Example

Config Server:

```bash

/application.yml
/order-service.yml
/payment-service.yml
```

Clients fetch on startup → or through actuator refresh.

---

86. What is a Deployment Manifest in Kubernetes?

Deployment manifest is YAML file describing:

- containers
- replicas
- ports
- environment variables
- rolling updates

Example manifest

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: order-service
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: order
        image: order:1.0
        ports:
        - containerPort: 8080
```

---

87. Explain Observability in microservices. What are its pillars?

Observability = ability to understand system internal state from external output.

Three Pillars

1. Logs

application logs using ELK / EFK

2. Metrics

system performance

Prometheus → Grafana

3. Traces

request flows

Jaeger / Zipkin

Why Needed

✔️ faster debugging
✔️ capacity planning
✔️ anomaly detection

---

88. What is Prometheus? How used in microservices?

Prometheus is **time-series metrics store**.

Used For

- service metrics
- resource monitoring
- alerting

Pipeline

Spring Boot → Micrometer → Prometheus → Grafana

Example metrics

- request count
- latency
- memory usage
- JVM GC

---

89. Explain ELK Stack (Elasticsearch, Logstash, Kibana).

Elasticsearch

stores logs and supports fast search

Logstash

ingests log data

parses formats

Kibana

visual dashboard

Use Case

Search logs using filters:

- log level
- traceId
- userId
- timestamp

---

90. What is Canary Monitoring? What metrics matter?

Monitor new release (canary) for:

- error rate
- latency
- CPU/RAM usage
- business metrics (orders placed, payments)

If metrics degrade → rollback automatically.

---

91. Explain JWT authentication in microservices.

JWT = JSON Web Token

Used for:

- authentication
- authorization

Structure

```
header.payload.signature
```

Flow

1. User logs in → Auth Service issues JWT
2. JWT sent with each request
3. Gateway validates JWT
4. Downstream services trust gateway

Pros

✔️ stateless
✔️ no DB lookup
✔️ fast

---

92. How do you secure inter-service communication?

Methods:

- mTLS
- API keys
- service accounts
- OAuth2 machine-to-machine
- sidecar proxies
- network policies

---

93. What is OAuth2 Client Credentials Flow?

Used for service-to-service communication.

Flow

1. Service A requests token from Authorization Server
2. Sends clientId + clientSecret
3. Authorization server issues access token
4. Service A calls Service B using token

Use Case

Payment service calling Fraud detection service.

---

94. Explain CORS. Why is it needed in microservices?

CORS allows browser clients from one origin to access another origin.

Needed for:

- SPA frontends (React, Angular)
- separate backend + UI domains

Spring Boot example:

```java
@CrossOrigin(origins = "https://myapp.com")
```

---

95. How do you implement retries in microservices?

Retries needed for:

- network glitches
- temporary failures

Resilience4j Example

```java
@Retry(name = "inventoryRetry", maxAttempts = 3)
public Response checkInventory() {
    ...
}
```

Retries with exponential backoff.

---

96. What is Chaos Engineering? Why is it important?

Chaos engineering intentionally injects failures to test system resilience.

Tools

- Chaos Monkey for Spring Boot
- Gremlin
- AWS Fault Injection Simulator

Tests

- kill containers
- add latency
- drop network
- kill nodes

---

97. Explain container orchestration. Why is Kubernetes preferred?

Container orchestration manages:

- deployments
- scaling
- rollover

- health checks

Kubernetes Advantages

✔️ self-healing
✔️ auto-scaling
✔️ declarative YAML
✔️ service discovery
✔️ secrets/config maps
✔️ ecosystem support

---

98. What is Pod Autoscaling? How does HPA work?

Horizontal Pod Autoscaler scales pods based on metrics.

Metrics Examples

- CPU
- memory
- custom metrics

Example

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
spec:
  minReplicas: 2
  maxReplicas: 10
  metrics:
    - type: Resource
      resource:
        name: cpu
        targetAverageUtilization: 50
```

99. What is rolling restart and why do we need it?

Rolling restart updates pods one by one.

✔️ avoids downtime
✔️ ensures continuous availability
✔️ supports zero downtime

---

100. What is CI/CD? How do you build pipelines for microservices?

CI/CD automates:

- building
- testing
- packaging
- deploying

Pipeline Stages

1. CI (Continuous Integration)

- compile
- unit tests
- integration tests
- static code analysis
- build docker image
- push to registry

2. CD (Continuous Deployment)

- deploy to dev
- deploy to QA
- canary or blue-green prod deployment
- health checks
- rollback logic

Popular Tools

- Jenkins
- GitHub Actions
- GitLab CI
- ArgoCD
- Spinnaker
- Tekton

🎉 SET D Completed! (76–100)

You now have:
✔️ Set A (1–25)
✔️ Set B (26–50)
✔️ Set C (51–75)
✔️ Set D (76–100)

All 100 questions completed with **detailed long answers**.

📌 NEXT STEP (Ready to Generate PDF)

Now that all 100 detailed answers are completed,
🔥 I can generate a **formatted, printable PDF** containing:

- Full 100 questions
- Long answers (20–30 pages)
- Proper headings
- Page numbers
- Table of contents
- Clean layouts

✔️ Just confirm:

"Yes, generate full PDF with all 100 detailed answers."

I will prepare the PDF.