1) WHY SRE ?

* Site Reliability Engineering exists because modern software systems are too complex and too critical to rely on manual operations and reactive fixes.
*As applications scale to thousands or millions of users, even small failures can cause major business loss. SRE provides a structured, engineering-driven way to keep systems reliable while still allowing teams to innovate.

1) Systems Must Be Always Available

    ==> SRE ensures high availability and quick recovery from failures.

2) Complexity of Modern Systems

    ==> SRE uses automation and engineering to manage the complexity.

3) Balance Between Speed and Stability

    ==> Development teams want fast releases, Businesses want stability.

    ==> SRE creates a balance using:

        SLOs (Service Level Objectives)

        Error budgets

        Controlled risk-taking

4) Failures Are Inevitable

    ==> SRE accepts this reality and focuses on:

        Designing resilient systems
        Fast incident response
        Learning from failures

5) Reduce Operational Burden

    ==> SRE reduces repetitive work through automation so engineers can focus on improvement.

6) Better User Experience

    ==> Reliability = good user experience.

2) WHAT IS SRE?

SRE is the practice of using software engineering and automation to ensure systems are reliable, scalable, and efficient in production.

SREs **write code, create automation, and design systems** so that:

- Failures are minimized
- Recovery is fast
- Systems stay available and performant

3)What does an SRE do?

An SRE's work usually includes:

1) Reliability & Uptime

- Define reliability targets
- Maintain SLAs, SLOs, SLIs
- Monitor system health

2) Automation

- Automate repetitive tasks
- Use Infrastructure as Code .
- Build scripts/tools to reduce manual work

3) Monitoring & Observability

- Set up alerts and dashboards
- Track metrics, logs, traces
- Detect issues before users do

4) Incident Management

- Handle outages
- Perform root cause analysis (RCA)
- Create postmortems

5) Capacity Planning

- Predict system growth
- Ensure systems handle load

6) Performance Optimization

- Reduce latency
- Improve scalability

4) DevOps vs SRE :

DevOps is a culture and set of practices.
SRE is a specific engineering approach to reliability.

DevOps focuses on:

- Faster releases
- CI/CD pipelines
- Automation
- Collaboration between Dev & Ops
- Infrastructure as Code
- Continuous testing and deployment

SRE focuses on:

- Reliability and uptime
- SLIs, SLOs, SLAs
- Error budgets

- Monitoring & observability
- Incident response
- Reducing operational toil

5) SRE PRINCIPLES / GOLDEN RULES:

==> SRE starts by **defining what reliability means in numbers.**

***** Focus on Service Level Objectives (SLOs):

- **SLI (Service Level Indicator)** → a metric (latency, error rate, uptime)
- **SLO (Service Level Objective)** → target value for that metric
- **SLA (Service Level Agreement)** → formal promise to customers

* Embrace Risk:

==> SRE accepts **calculated risk** instead of chasing perfection.

* Use Error Budgets:

==> This balances **innovation vs stability.**

* Eliminate Toil:

==> **Toil = manual, repetitive, operational work** that:

- Is automatable
- Has no long-term value
- Scales poorly

* Automate Everything Possible:

Automation reduces: Human error , Time spent on routine tasks and Operational stress.

* Monitor Four Golden Signals:

1. **Latency** → response time
2. **Traffic** → demand/load

3. **Errors** → failure rate
4. **Saturation** → system capacity usage

==> Monitor user-impacting metrics, not just server stats.

* Design for Failure and Blameless Postmortems.
==> Recover quickly and Prevent recurrence

* Measure and Improve Continuously.