Section - 1

Site Reliability Engineering Tooling (Prometheus, Grafana)

1.1 Introduction to SRE Tooling

Explanation:

Site Reliability Engineering (SRE) focuses on maintaining system reliability through engineering practices. Key to this is robust tooling for **monitoring, alerting, and visualization**. Without accurate tools, identifying failures, tracking SLAs, and proactive prevention becomes impossible.

SRE tools must offer:

- Real-time system health visibility
- Actionable alerts
- Data-driven decision-making capabilities

Real-world Example:

Google uses custom-built monitoring systems (Borgmon) alongside open-source tools like Prometheus to watch everything from HTTP latencies to hardware temperatures.

Use Cases:

- Monitoring uptime and error rates
- Alerting on disk space or CPU saturation
- Visualizing end-to-end user experience

Key Points:

- Tooling must be **reliable, scalable, and automatable**.
- **Integration** across monitoring, alerting, and incident management is vital.

1.2 Prometheus

Explanation:

Prometheus is an **open-source monitoring system** and **time-series database** designed for reliability and scalability. It collects metrics by **pulling data** from instrumented targets, stores the data locally, and allows querying using **PromQL**.

Components:

- **Prometheus Server:** Scrapes and stores metrics.
- **Alertmanager:** Handles alerts from Prometheus.
- **Exporters:** Collect metrics (Node Exporter for system metrics, Blackbox Exporter for HTTP probes, etc.)

Real-world Example:

A SaaS company monitors customer billing services. If transaction errors spike above 5%, Prometheus alerts the on-call engineer.

Use Cases:

- Detect application latencies
- Monitor Kubernetes cluster node health
- Identify pod restarts due to memory issues

Key Points:

- Prometheus uses a **pull model** for metrics.
- Retains metrics for **historical analysis** and trend prediction.

1.3 Writing PromQL Queries

Explanation:

PromQL (Prometheus Query Language) is used to extract and manipulate time-series data in Prometheus.

Examples:

- Current CPU usage of all servers:
  `rate(node_cpu_seconds_total{mode="user"}[5m])`

- Alert if HTTP error rate exceeds threshold:
  `rate(http_requests_total{status="500"}[1m]) > 0.05`

Real-world Example:

An e-commerce company uses PromQL to visualize "Top 5 slowest APIs" to prioritize optimizations.

Use Cases:

- Create SLO dashboards

- Set dynamic alert thresholds based on historical performance

Key Points:

- PromQL is powerful for **aggregations**, **filters**, and **rate calculations**.

1.4 Setting up Alertmanager

Explanation:

Alertmanager is responsible for **handling alerts** from Prometheus and **routing them** to communication platforms (email, Slack, PagerDuty).

Features:

- Grouping alerts
- Routing based on severity
- Silencing noisy alerts during maintenance

Real-world Example:

In an airline booking system, if the reservation API fails, an immediate PagerDuty notification is triggered through Alertmanager.

Use Cases:

- Send critical alerts only to on-call engineers
- Silence non-urgent alerts at night

Key Points:

- Group similar alerts to avoid notification fatigue.
- Integrate escalation policies into Alertmanager.

1.5 Prometheus Best Practices

Explanation:

Following best practices ensures Prometheus remains scalable and effective.

Best Practices:

- **Label your metrics** wisely (avoid high-cardinality labels like `user_id`).
- **Tune scrape intervals** appropriately.
- **Use Federation** to scale Prometheus across multi-cluster/multi-region setups.

Real-world Example:

A Fintech firm uses a **hierarchical Prometheus federation model** to monitor 15 Kubernetes clusters worldwide without overloading a single Prometheus instance.

Use Cases:

- Enable multi-region observability
- Scale with microservices growth

Key Points:

- Avoid heavy queries directly on Prometheus — use Grafana or Thanos.

1.6 Grafana

Explanation:

Grafana is an open-source platform for **visualizing metrics** collected from multiple sources like Prometheus, InfluxDB, Elasticsearch, and more.

Features:

- Highly customizable dashboards
- Supports plugins (data sources, panels)
- Integrated alerting (since Grafana v8)

Real-world Example:

A health-tech startup uses Grafana to build live dashboards showing patients' vital stats from wearable devices.

Use Cases:

- Build business KPI dashboards
- Show real-time application latency graphs

Key Points:

- Use variables in Grafana to create dynamic dashboards.
- Version control your Grafana dashboards via JSON.

1.7 Integrating Grafana with Prometheus

Explanation:

Grafana connects easily with Prometheus as a **data source**. Once connected, you can create stunning visualizations on top of PromQL queries.

Real-world Example:

A logistics company visualizes delivery latencies using Prometheus + Grafana integration to optimize shipping times.

Steps:

- Add Prometheus as a Data Source
- Build queries using PromQL
- Design Panels and Dashboards

1.8 Creating and Customizing Dashboards

Explanation:

Grafana allows you to build **panels** (graphs, tables, heatmaps) and **dashboards** tailored to your metrics and KPIs.

Best Practices:

- Use color thresholds for intuitive status indication.
- Group panels logically by service or region.

Real-world Example:

Uber uses Grafana dashboards to visualize fleet health, server load, and trip rates in real-time.

1.9 Monitoring SLIs, SLOs, and Error Budgets

Explanation:

- **SLI (Service Level Indicator):** What you measure (e.g., uptime %).
- **SLO (Service Level Objective):** The goal (e.g., 99.9% uptime).
- **Error Budget:** Allowed failures within the SLO.

Real-world Example:

If a system is allowed 0.1% downtime/month (error budget), the team decides whether to ship a risky new feature.

Use Cases:

- Prioritize reliability over new features
- Drive operational excellence in SRE teams