Project Outline: Distributed Log Analysis System (100% Free & Open-Source)

Project Goal: To build a scalable system that ingests, processes, and analyzes logs in real-time. This entire stack can be built for free using open-source software. The only cost is the hardware (or cloud infrastructure) you run it on.

Phase 1: Core Data Ingestion and Processing

This phase focuses on getting data into our system and processing it.

- **Data Serialization: Protocol Buffers (Protobuf)** for efficient data exchange.
- **Messaging/Queuing: Apache Kafka** for a durable and scalable event streaming platform.
- **Data Processing (Batch): Apache Hadoop** (using MapReduce) for large-scale historical analysis.
- **Data Processing (Stream): Apache Beam** running on an **Apache Flink** runner for real-time stream processing.
- **Inter-Service Communication: gRPC** for high-performance, cross-language RPCs.
- **Coordination: Apache Zookeeper** for managing configuration and leader election.

Phase 2: Infrastructure and Deployment

This phase covers how we'll run and manage our application.

- **Containerization & Orchestration:** Services will be packaged as **Docker** containers and deployed on a **Kubernetes** cluster.
- **Service Mesh & Load Balancing: Istio** will manage traffic, security, and observability between services, using **Envoy** as its data plane proxy.
- **API Gateway: Swagger/OpenAPI** specifications will be used to define our public APIs, potentially served through a gateway like the open-source **Envoy Gateway**.
- **Configuration Management: Terraform** will define and manage our cloud infrastructure as code.

Phase 3: Storage Layer

This is where our processed data will live.

- **Distributed File System:** Raw logs will be archived in **HDFS** (Hadoop Distributed File System).
- **NoSQL Database (Wide-Column):** Processed data and aggregates will be stored in **Apache Cassandra** for fast, scalable queries.
- **Distributed SQL Database:** User accounts and alert configurations will be stored in **CockroachDB** (using the free Community Edition).
- **Columnar Storage Format:** Batch jobs will store data in **Apache Parquet** format for efficient analytical queries.
- **Key-Value Store: RocksDB** will be used as an embedded key-value store within services for caching or local state.

Phase 4: Services and User Interface

This phase focuses on how users interact with the system.

- **Data Warehousing & Querying: Presto** will run fast, interactive SQL queries on data stored in Parquet format in HDFS.
- **Search & Indexing:** Logs will be indexed in the open-source version of **Elasticsearch** for powerful full-text search.
- **Data Visualization: Apache Superset** will connect to our analytical databases to create dashboards and charts.
- **Notebooks for Analysis:** A **Jupyter** environment will be provided for ad-hoc data exploration and analysis.

Phase 5: DevOps and Monitoring

This phase covers the tools to build, test, deploy, and monitor our system.

- **Build System: Bazel** will be used for fast, correct builds across our monorepo.
- **CI/CD: Jenkins** will automate the build, test, and deployment pipeline.
- **Monitoring: Prometheus** will collect metrics, and **Grafana** will visualize them in dashboards.
- **Logging: Fluentd** will collect logs, sending them to **Loki** for storage and querying through Grafana.
- **Error Tracking:** A self-hosted **Sentry** instance will capture and alert on application exceptions.
- **Distributed Tracing: Jaeger** will be used to visualize traces generated via **OpenTelemetry** instrumentation.
- **Code Review: Gerrit** will be used for code reviews.
- **Issue Tracking: Bugzilla** will be used to track bugs and features, providing a powerful, free alternative to JIRA.

Phase 6: Security and Internal Tooling

This phase focuses on securing the system and providing internal tools for developers.

- **Identity and Access Management: Keycloak** will handle user authentication and authorization.
- **Secrets Management:** The open-source version of **HashiCorp Vault** will securely manage all secrets.
- **Authorization Service: SpiceDB** will be implemented for fine-grained, relationship-based access control.
- **Internal Comms:** A self-hosted **Mattermost** server will be used for team messaging.
- **Password Management:** Developers can use **pass**, the standard Unix password store, for managing their credentials.