Capstone

A screenshot of a computer

AI-generated content may be incorrect.

Here is the **explanation document** for your final **StorME Architecture** diagram. 🎯

**📌 StorME Architecture Explanation**

**1️⃣ Overview**

StorME is an **enterprise-level file management system** designed for **secure and scalable deployment**. This architecture is built using **vSphere, Docker Swarm, Active Directory authentication, and a high-availability backend with monitoring and CI/CD integration**.

**2️⃣ Key Components & Roles**

**🔹 vSphere Cluster**

* Hosts all virtual machines (VMs) required for the deployment.
* Provides **virtualization** and **resource allocation**.

**🔹 Tailscale VPN**

* **Ensures secure networking between nodes** across different vSphere instances.
* Acts as a **private overlay network**.

**🔹 Docker Swarm**

* **Orchestrates containerized services** across multiple worker nodes.
* Provides **automatic load balancing and failover support**.

**🔹 Load Balancer (Nginx/Traefik)**

* **Distributes incoming API requests across Worker Nodes.**
* Ensures **high availability** and **fault tolerance**.

**🔹 Swarm Manager**

* **Manages containerized services** across Worker Nodes.
* **Does NOT handle traffic directly**—only used for service orchestration.

**3️⃣ Worker Nodes & Services**

**🔹 Worker Node 1 (API & Backend Services)**

* Runs **Backend Services (Python/Node.js)** to handle: ✅ File storage requests  
  ✅ User authentication (via AD)  
  ✅ API endpoints
* Connects to **Storage (OpenZFS) and Database (PostgreSQL/MariaDB).**
* Handles **incoming API calls from API Gateway**.

**🔹 Worker Node 2 (Database & CI/CD)**

* Runs **PostgreSQL/MariaDB database** to store: ✅ User authentication data  
  ✅ File metadata  
  ✅ System logs
* **CI/CD Pipeline (GitHub Actions/Jenkins)** automates: ✅ Backend service deployments  
  ✅ Database schema migrations

**4️⃣ Storage, Security & Backup**

**🔹 OpenZFS (Shared Storage)**

* **Handles file storage operations**.
* Provides **redundancy & high availability**.

**🔹 Encryption (TLS/SSL + AES)**

* Ensures **data encryption at rest & in transit**.
* Secures **sensitive information stored in OpenZFS**.

**🔹 Backup & Disaster Recovery**

* Uses **rsync/Duplicity** for **automated backups**.
* Prevents **data loss in case of failure**.

**5️⃣ Authentication & Monitoring**

**🔹 Active Directory (LDAP/OAuth)**

* **Centralized authentication using enterprise AD.**
* Ensures **Single Sign-On (SSO) & Role-Based Access Control (RBAC).**

**🔹 ELK Stack (Elasticsearch, Logstash, Kibana)**

* **Monitors application logs & database performance.**
* Tracks **API calls, database queries, and system health**.

**6️⃣ Workflow Summary**

**📌 User Login & Authentication Flow**

1️⃣ **User logs into StorME Web Frontend (React.js).**  
2️⃣ **API Gateway (Nginx) forwards login request to Authentication Service.**  
3️⃣ **Authentication Service queries Active Directory (LDAP/OAuth).**  
4️⃣ **Active Directory validates credentials & returns authentication token.**  
5️⃣ **Token is verified, and the user gains access to their dashboard.**

**📌 File Storage & Retrieval**

1️⃣ **User uploads/downloads a file via the Web Frontend.**  
2️⃣ **Backend Services handle the request & communicate with Storage (OpenZFS).**  
3️⃣ **Metadata is stored in the PostgreSQL database.**  
4️⃣ **Data is encrypted before storage.**

**📌 System Monitoring & Deployment**

1️⃣ **CI/CD (Jenkins/GitHub Actions) deploys backend updates automatically.**  
2️⃣ **ELK Stack logs all system events & errors.**  
3️⃣ **Database and storage backups are scheduled for disaster recovery.**

**🔥 Conclusion**

✅ **Enterprise-Grade, Highly Secure & Scalable Architecture**  
✅ **Centralized Authentication with AD for SSO & Security**  
✅ **Efficient Storage & Backup System**  
✅ **Full Automation with CI/CD & Monitoring**

🚀 **This architecture is fully ready for implementation and meets enterprise standards.**  
Let me know if you need **further refinements or adjustments!** 🎯🔥