**Harambee Digital Bank: Technical Stack and Implementation**

**Total Stack**

1. Hyperledger Fabric
2. Go (Golang)
3. Django (Python)
4. PostgreSQL
5. Apache Kafka
6. Docker
7. Kubernetes
8. React.js
9. React Native
10. TensorFlow
11. PyTorch
12. Apache Spark
13. PostGIS
14. QGIS
15. Istio
16. Kong
17. Prometheus
18. Grafana
19. ELK Stack (Elasticsearch, Logstash, Kibana)
20. Amazon Redshift
21. Apache NiFi
22. Tableau
23. scikit-learn
24. Keycloak
25. HashiCorp Vault
26. AES-256
27. RSA
28. Palo Alto Networks Firewalls
29. Snort
30. Git
31. GitHub
32. Jenkins
33. Terraform
34. Ansible
35. JFrog Artifactory

**1. Core Banking System**

**Tech Stack:**

* Blockchain: Hyperledger Fabric
* Smart Contracts: Go (Golang)
* Backend: Django (Python)
* Database: PostgreSQL
* Message Broker: Apache Kafka

**Implementation:**

The core banking system is implemented as a set of microservices, each running in Docker containers orchestrated by Kubernetes. Hyperledger Fabric provides the blockchain infrastructure, with smart contracts (chaincode) written in Go. Django handles API requests and business logic, while PostgreSQL stores off-chain data. Kafka is used for event streaming between microservices.

**2. Chama Management System**

**Tech Stack:**

* Blockchain: Hyperledger Fabric
* Smart Contracts: Go (Golang)
* Backend: Django (Python)
* Frontend: React.js
* Mobile: React Native

**Implementation:**

The Chama system is implemented as a separate set of microservices. Smart contracts handle Chama operations on the blockchain. Django microservices manage business logic and API endpoints. React.js and React Native provide web and mobile interfaces, communicating with backend services via RESTful APIs.

**3. Government Tender Transparency System**

**Tech Stack:**

* Blockchain: Hyperledger Fabric
* Smart Contracts: Go (Golang)
* Backend: Django (Python)
* AI/ML: TensorFlow, PyTorch
* Data Processing: Apache Spark
* Frontend: React.js

**Implementation:**

This system uses a microservices architecture. Separate microservices handle tender creation, bid management, and verification processes. AI models for image analysis run in dedicated containers. Spark is used for big data processing of verification data. All microservices communicate through Kafka event streams.

**4. Flexible Project Verification System**

**Tech Stack:**

* Blockchain: Hyperledger Fabric
* Smart Contracts: Go (Golang)
* Backend: Django (Python)
* AI/ML: TensorFlow, PyTorch
* Mobile: React Native
* GIS: PostGIS, QGIS

**Implementation:**

Implemented as microservices, this system includes services for project registration, verification layer management, and AI-driven analysis. GIS functionality is provided by PostGIS extension to PostgreSQL. Mobile app for verifiers is built with React Native. All services are containerized and managed by Kubernetes.

**5. Microservices Architecture**

**Tech Stack:**

* Container Orchestration: Kubernetes
* Service Mesh: Istio
* API Gateway: Kong
* Monitoring: Prometheus, Grafana
* Logging: ELK Stack (Elasticsearch, Logstash, Kibana)

**Implementation:**

All major components of the system are implemented as microservices, containerized using Docker, and orchestrated with Kubernetes. Istio provides service mesh capabilities for inter-service communication. Kong serves as the API gateway for external requests. Prometheus and Grafana handle monitoring, while the ELK stack manages logging across all services.

**6. Data Management and Analytics**

**Tech Stack:**

* Data Warehouse: Amazon Redshift
* ETL: Apache NiFi
* Business Intelligence: Tableau
* Machine Learning: scikit-learn, TensorFlow

**Implementation:**

Data from various microservices is aggregated in Redshift for analytics. NiFi manages data pipelines between operational databases and the data warehouse. Tableau is used for creating dashboards and reports. Machine learning models for predictive analytics are developed using scikit-learn and TensorFlow.

**7. Security Infrastructure**

**Tech Stack:**

* Identity Management: Keycloak
* Secrets Management: HashiCorp Vault
* Encryption: AES-256, RSA
* Network Security: Palo Alto Networks Firewalls
* Intrusion Detection: Snort

**Implementation:**

Security is implemented as a cross-cutting concern across all microservices. Keycloak provides centralized identity management. Vault manages secrets and encryption keys. All data at rest and in transit is encrypted. Network security is enforced at multiple levels, with dedicated microservices for security operations.

**8. DevOps and Continuous Integration/Continuous Deployment (CI/CD)**

**Tech Stack:**

* Version Control: Git, GitHub
* CI/CD: Jenkins
* Infrastructure as Code: Terraform
* Configuration Management: Ansible
* Artifact Repository: JFrog Artifactory

**Implementation:**

The entire system follows a DevOps approach with automated CI/CD pipelines. Jenkins orchestrates the build, test, and deployment processes. Infrastructure is managed as code using Terraform and Ansible. All software artifacts are stored and versioned in Artifactory.