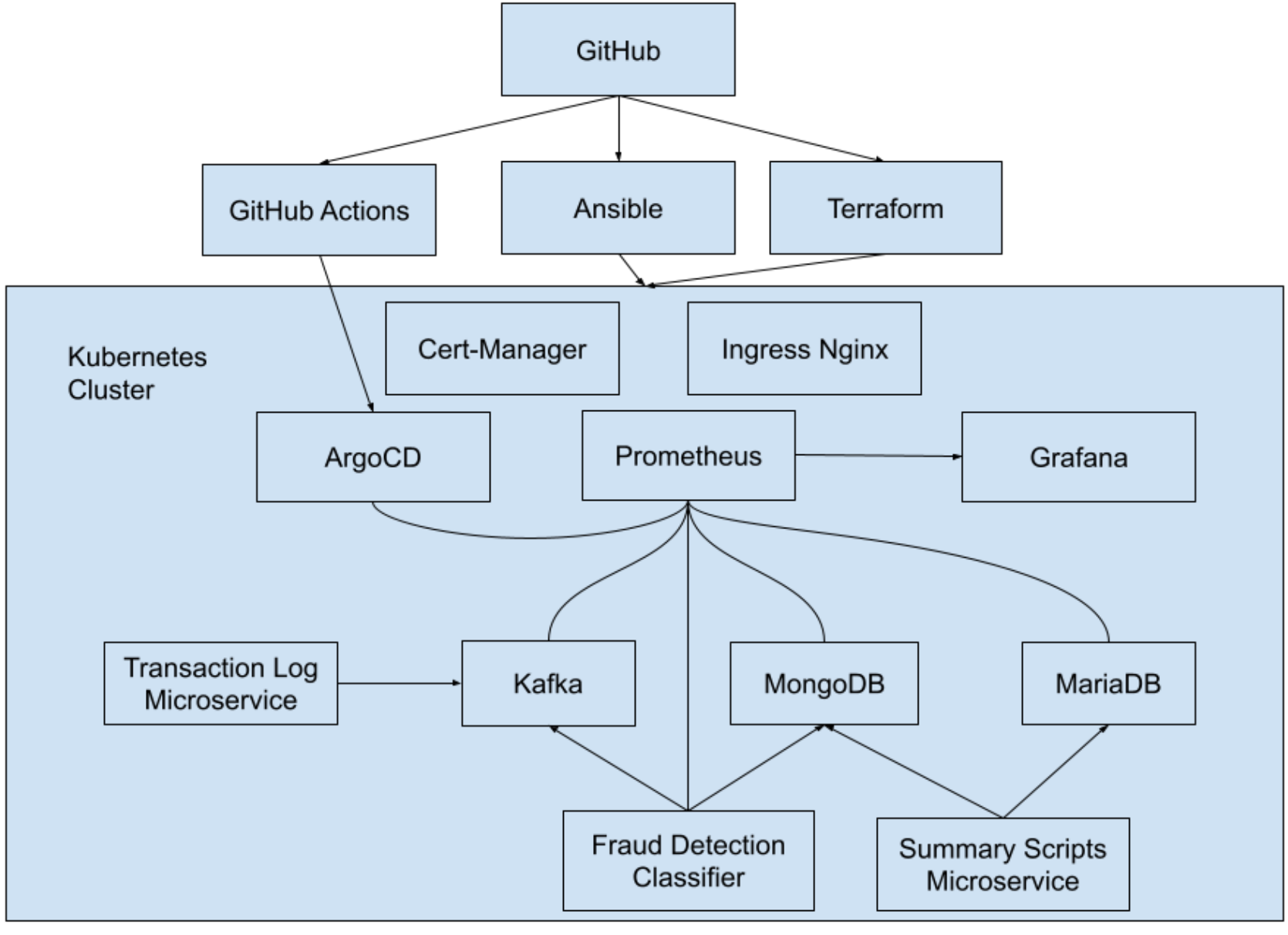
PROACTIVE NONSENSE CRUSHER

CREDIT CARD FRAUD DETECTION

Alex Griffin
Balin Warren
Chadon Mathurin
Daniel Nelson

GOALS AND OBJECTIVES

- To build a system for detecting credit card fraud using machine learning
- Using sound engineering practices:
  - Implement using infrastructure as code (IaC)
  - Automated deployment and CI/CD
  - Horizontally scalable architecture (microservices)
  - Instrumented and monitored


GitHub
GitHub Actions
Ansible
Terraform
Kubernetes Cluster
Cert-Manager
Ingress Nginx
ArgoCD
Prometheus
Grafana
Transaction Log Microservice
Kafka
MongoDB
MariaDB
Fraud Detection Classifier
Summary Scripts Microservice

COMPUTING INFRASTRUCTURE

- Running on self-hosted Proxmox server
- HPE ProLiant DL380 Gen9 Server
- Virtual machines provisioned with Terraform
- Kubernetes cluster configured with Ansible

DOCKER AND KUBERNETES

- Docker is a container engine for developing, shipping, and running applications
  - More lightweight than virtual machines
- Kubernetes is a container orchestration platform for managing a computing cluster made up of many containerized services

CI/CD

- K3s Kubernetes cluster managed by ArgoCD
- Component services are defined as Dockerfiles and ArgoCD Applications
- GitHub Action workflows deploy Kubernetes objects defined in git repository
- Build artifacts pushed to private container registry
- Secrets managed by GitHub to avoid unwanted exposure of sensitive data

INGRESS NGINX

- Single point of entry into the cluster
- Routes domains and paths to services
- Cert-manager to provision, install, and rotate Cloudflare SSL certificates

INGESTING CREDIT CARD TRANSACTIONS

- Credit card transaction dataset
  - https://www.kaggle.com/datasets/anurag629/credit-card-fraud-transaction-data/data
- Recorded information:

```
Transaction ID,Date,Day of Week,Time,Type of Card,Entry Mode,Amo
#3577 209,14-Oct-20,Wednesday,19,Visa,Tap,£5,POS,Entertainment,U
```

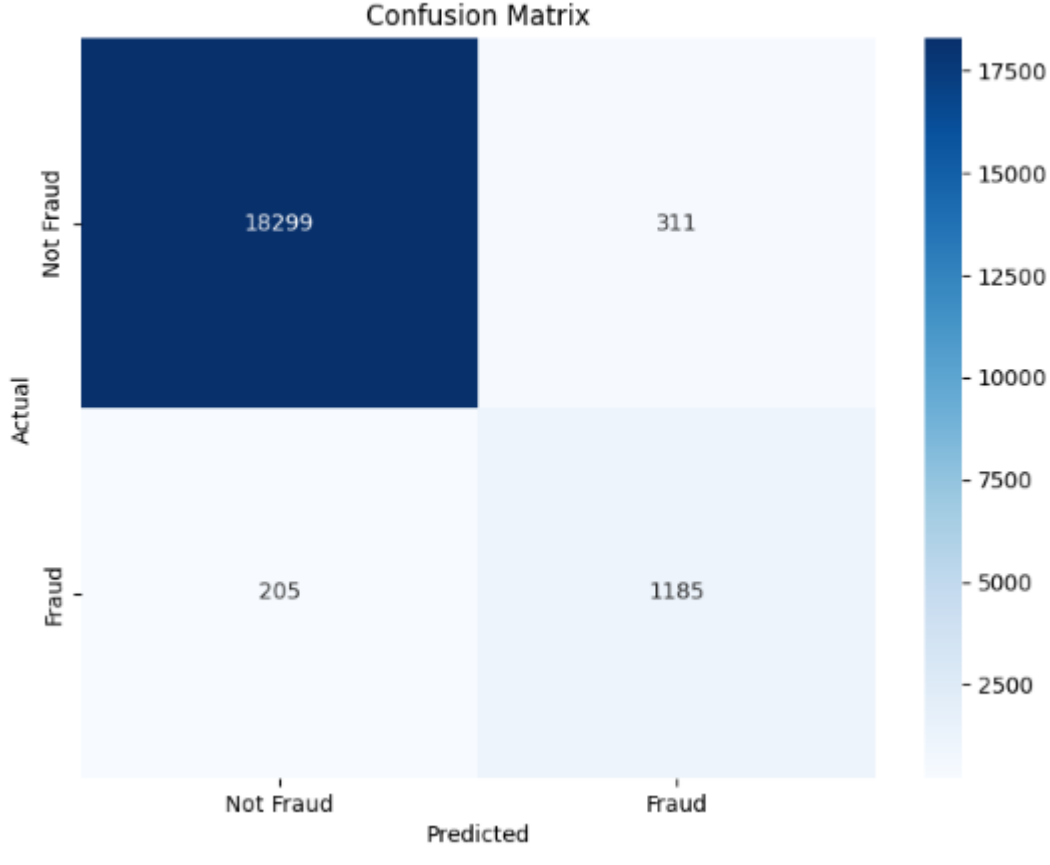
- Python script reads transaction log and feeds data into Apache Kafka

APACHE KAFKA

- Open-source stream-processing platform
- Core Concepts
  - Producers, Consumers, Topics, Partitions, Brokers
- Use Cases
  - Real-time analytics and monitoring
  - Log aggregation and stream processing
  - Event sourcing and data integration

FRAUD CLASSIFICATION

Winning Model (Random Forest):



scikit-learn Classification Model

```
# Initialize the model. Class weight is being set due to
# imbalance of fraud/not fraud cases
model = RandomForestClassifier(n_estimators=50,
  max_depth=10, random_state=42,
  class_weight={0: 1, 1: 8})

# Split chunk into training and validation subsets
X_train_chunk, X_val_chunk, y_train_chunk, y_val_chunk =
train_test_split(
    X_chunk, y_chunk, test_size=0.2, random_state=42
)

# Train the model incrementally
model.fit(X_train_chunk, y_train_chunk)
```

MONGODB AND MARIADB (MYSQL)

- MariaDB
  - Open Source branch of MySQL
  - Built to easily scale out
  - Easily multithread for higher transaction throughput
- MongoDB
  - Scalable (Shardable)
  - Quickly integrate with microservices
  - Flexible data structure

SUMMARY SCRIPTS

- Purpose is to take the large amounts of transaction logs and compile into digestible summaries.
- Ingests transaction logs from MongoDB.
- Tabulates data of all transactions into summaries every 5 minutes.
- Generates one summary for all transactions and another summary for just fraud transactions.
- Summaries get inserted into MySQL tables
- Frequency of summaries can be easily adapted to account for volume.

KUBERNETES MONITORING DASHBOARD

Tool: Grafana

Data Source: Prometheus

Purpose: Real-time monitoring of Kubernetes cluster health and resource utilization

Key Takeaway: Enables proactive detection of performance bottlenecks and optimization of cluster resources.

KUBERNETES DASHBOARD FEATURES

1. **Unified Monitoring:** Aggregates key Kubernetes metrics like CPU, memory, and network usage.
2. **Real-Time Updates:** Refreshes every 5 seconds for up-to-date insights.
3. **Customizable Panels:** Editable panels for tailored metrics and thresholds.
4. **Proactive Alerts:** Highlights issues such as memory shortages, disk space constraints, and CPU bottlenecks.

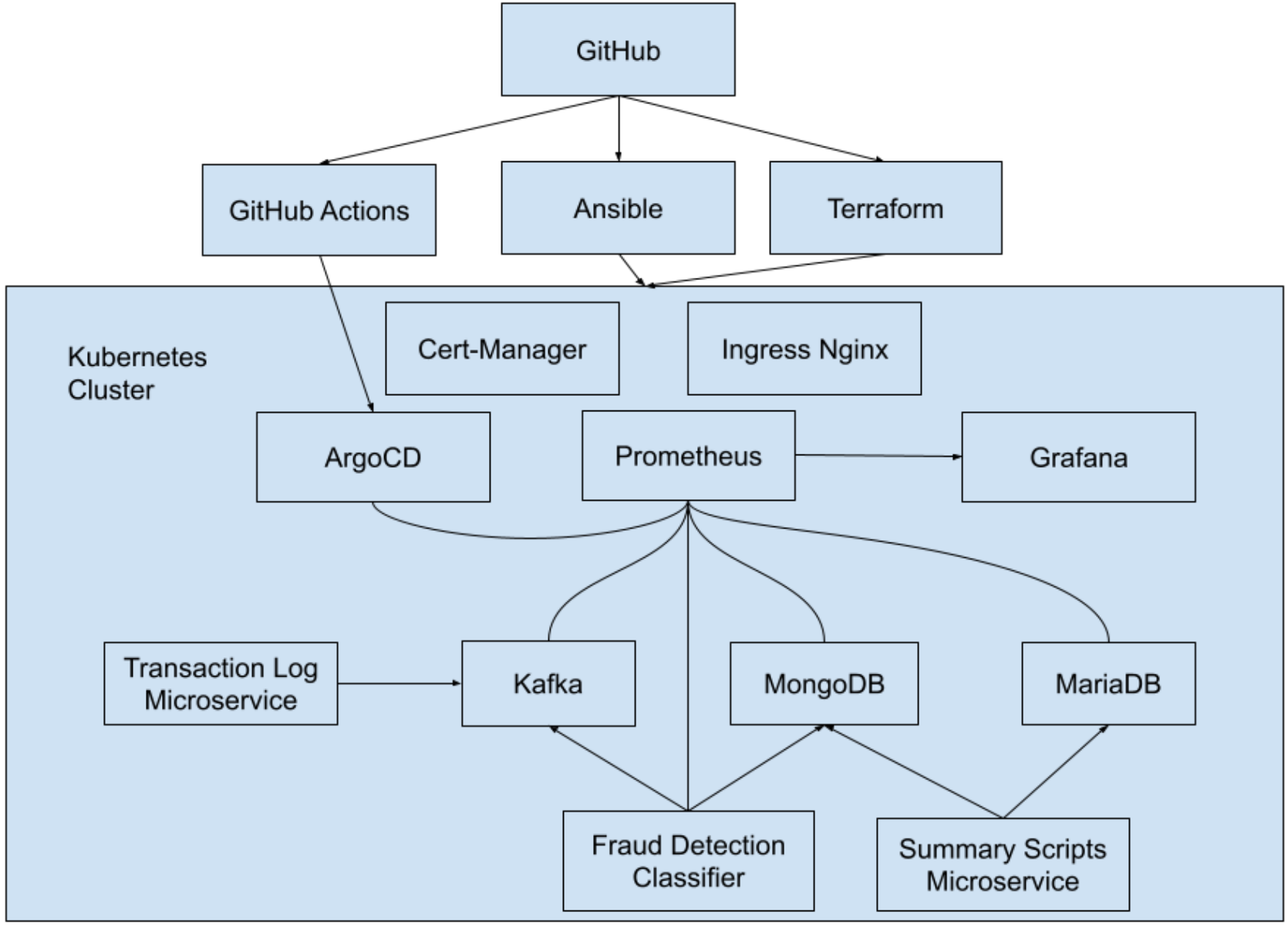
USE CASES:

- **Performance Monitoring:** Avoid over-allocation or underutilization.
- **Fault Detection:** Detect anomalies like high pod restarts.
- **Capacity Planning:** Scale resources to match workload demands.

LIVE DEMO

POTENTIAL NEXT STEPS

1. Detailed tracing of microservices with Jaeger
2. Oauth2 Integration with Ingress Nginx
3. Active-Active node/cluster failover
4. Dev environment with canary releases
5. Hardening checklist for each application
6. App of Apps deployment pattern


GitHub
GitHub Actions
Ansible
Terraform
Kubernetes Cluster
Cert-Manager
Ingress Nginx
ArgoCD
Prometheus
Grafana
Transaction Log Microservice
Kafka
MongoDB
MariaDB
Fraud Detection Classifier
Summary Scripts Microservice

QUESTIONS?