- Containerized, Automated & Secure Deployment
- Complete DevOps Pipeline Implementation
- Terraform + Ansible + Docker + CI/CD + Drift Detection

Microservices TODO Application

- Prepared By: Kamil Balogun

System Architecture Overview

6 Microservices in Different Languages:

- Frontend (Vue.js)
- Auth API (Go)
- Todos API (Node.js)
- Users API (Java Spring Boot)
- Log Processor (Python)
- Redis Queue

Key Features:

- Traefik Reverse Proxy with SSL termination
- Docker Containerization for all services
- Redis for message queuing

PART 1: Application Containerization

Implementation Flow:

- Individual Dockerfiles for each service
- Docker Compose orchestrates all services
- Traefik Proxy handles SSL + Routing

Single Command Deployment:

```
docker compose up -d
```

HTTPS Endpoints:

- https://hng.eazyppro.net
- https://hng.eazyppro.net/api/auth
- https://hng.eazyppro.net/api/todos
- https://hng.eazyppro.net/api/users

Features:

- Automatic SSL with Let's Encrypt certificates
- HTTP → HTTPS automatic redirection

PART 2: Infrastructure & Automation

Three Main Components:

Terraform:

- AWS EC2 provisioning
- Security groups configuration
- Remote state management (S3)
- Dynamic Ansible inventory generation

Ansible:

- Dependencies role (Docker, Git, packages)
- Deploy role (app deployment, SSL setup)
- Idempotent operations

CI/CD:

- GitHub Actions workflows
- Drift detection with email alerts
- Automated deployment pipelines

Terraform: Idempotent Infrastructure

Key Components:

- main.tf: EC2 instances + Security Groups
- Remote State: S3 Backend for team collaboration
- Dynamic Inventory: Auto-generated for Ansible

Automatic Ansible Execution:

```
resource "null_resource" "run_ansible" {
        provisioner "local-exec" {
        command = "ansible-playbook -i inventory.ini
site.yml"
        }
}
```

Features:

- Fully idempotent operations
- No resource recreation unless drift occurs
- Automatic integration with Ansible

Ansible: Server Configuration & Deployment

Dependencies Role:

- Docker & Docker Compose installation
- Git and system packages
- User permissions and groups

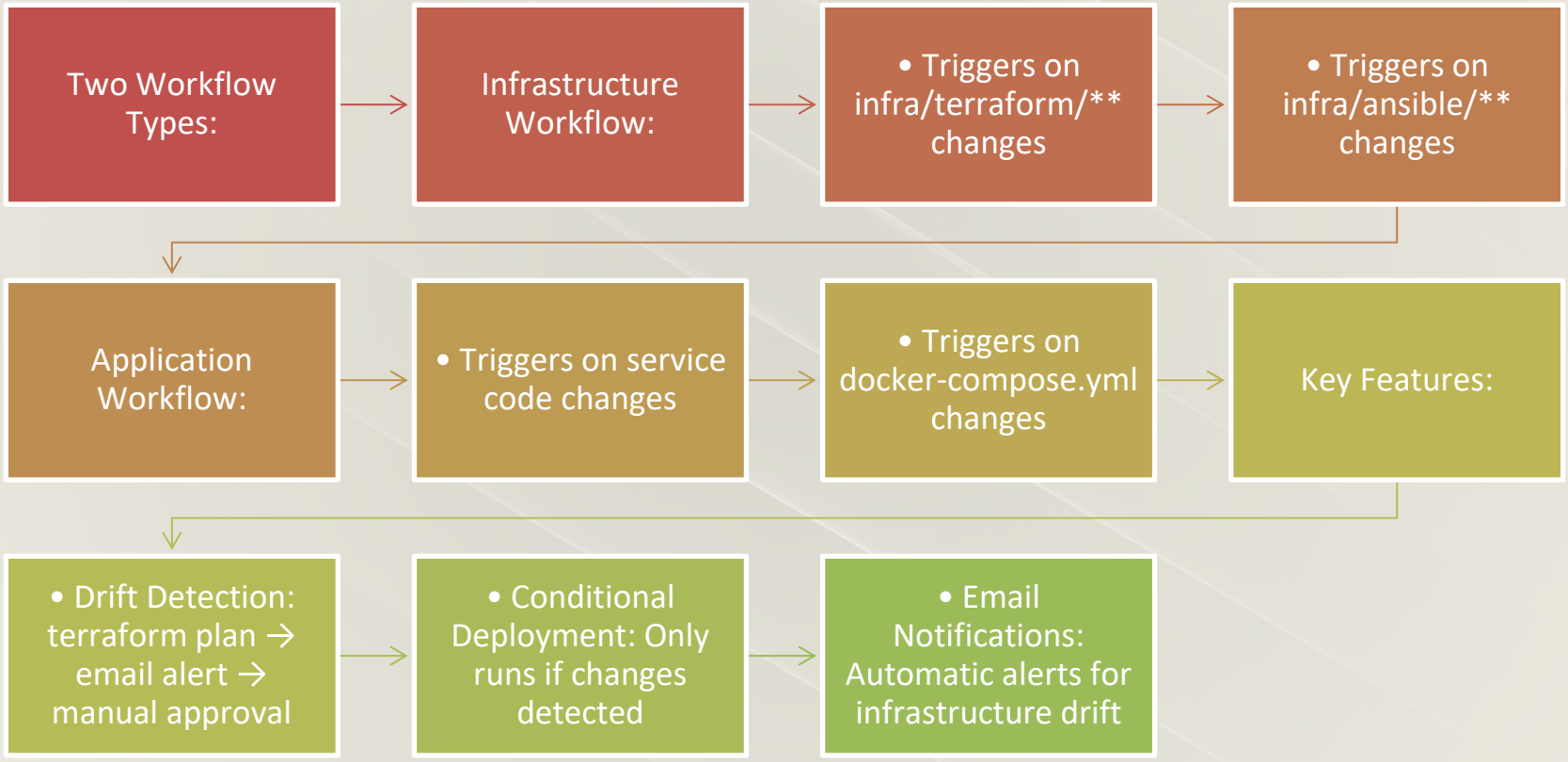
Deploy Role:

- Repository cloning and updates
- Container management with Docker Compose
- SSL configuration and health checks

Key Features:

- Idempotent Deployment: No restart unless files changed
- Git Integration: Automatic repo cloning and updates
- Container Orchestration: Full Docker Compose management

CI/CD: Automated Deployment Pipeline



Drift Detection & Safety Mechanism

5-Step Safety Process:

1. Plan Check: terraform plan -detailed-exitcode
2. Drift Detection: Exit code = 2 indicates changes
3. Email Alert: Notify stakeholders, pause pipeline
4. Manual Approval: GitHub Environment protection
5. Apply Changes: terraform apply only after approval

Safety Rules:

- No Drift = Automatic Proceed
- Drift Detected = Email + Manual Approval Required

This ensures complete transparency and control over infrastructure changes.

Security Implementation

Multi-Layer Security Approach:

Network Security:
- AWS Security Groups with minimal port exposure
- SSH key-based authentication
- Only ports 22, 80, 443 exposed

SSL/TLS Security:
- Let's Encrypt certificates with automatic renewal
- HTTP → HTTPS automatic redirection
- Traefik handles SSL termination

Application Security:
- JWT token authentication
- API authorization middleware
- Container isolation and network segmentation

PART 3: Single Command Deployment

Complete Stack Deployment:

- terraform apply -auto-approve

5-Step Automated Process:

1. Provision: AWS EC2 + Security Groups
2. Generate: Ansible inventory file
3. Configure: Install all dependencies
4. Deploy: Application containers
5. Secure: Traefik + SSL setup

Benefits:

- Fully Automated: Zero manual intervention required
- Idempotent: Skip unchanged resources
- Production Ready: HTTPS endpoints immediately available

Key Benefits & Achievements

🚀 Automation:

- Single command deployment
- Zero manual configuration
- Automatic SSL setup

🛡️ Safety:

- Infrastructure drift detection
- Email notifications for changes
- Manual approval gates

🔄 Reliability:

- Idempotent operations
- Remote state management
- Rollback capability

📈 Scalability:

- Microservices architecture
- Container orchestration
- Cloud-native design

Live Demonstration & Questions

Ready to Deploy!

Commands to Execute:

- cd infra/terraform
- terraform apply -auto-approve

Application Access:

- https://hng.eazyppro.net
- https://hng.eazyppro.net/api/auth
- https://hng.eazyppro.net/api/todos
- https://hng.eazyppro.net/api/users

Questions & Discussion

Thank you for your attention!