AfricanMarket Rollback Procedures

Overview

This document outlines the comprehensive rollback procedures for the AfricanMarket application. These procedures ensure that we can quickly and safely revert to a previous stable state in case of critical issues during or after deployment.

Rollback Triggers

Automatic Rollback Triggers

- System uptime < 95% for more than 15 minutes
- Error rate > 5% for more than 10 minutes
- Critical security vulnerability detected
- Data corruption or loss detected
- Payment processing failure rate > 2%

Manual Rollback Triggers

- Customer complaints exceed acceptable threshold
- Major feature completely non-functional
- Performance degradation > 50% from baseline
- Third-party service integration failures
- Team decision based on business impact

Rollback Authority

Decision Makers (in order of authority)

1. **CTO/Technical Lead** - Final technical decision authority
2. **Engineering Manager** - Operational rollback decisions
3. **Lead Developer** - Emergency rollback execution
4. **DevOps Lead** - Infrastructure rollback authority

Escalation Chain

1. **Level 1**: On-call engineer identifies issue
2. **Level 2**: Engineering lead confirms rollback necessity
3. **Level 3**: Management approval for business impact
4. **Level 4**: Executive notification for major rollbacks

Rollback Types

1. Application Rollback

Scope: Application code and configuration
Duration: 5-15 minutes
Impact: Application features and functionality

Procedure:
1. Stop current application instances
2. Deploy previous stable version
3. Update configuration if needed
4. Restart application services
5. Verify functionality

2. Database Rollback

Scope: Database schema and data
Duration: 30 minutes - 2 hours
Impact: Data state and application compatibility

Procedure:
1. Stop application instances
2. Create current state backup
3. Restore from previous backup
4. Run data integrity checks
5. Update application connections

3. Infrastructure Rollback

Scope: Server configuration and environment
Duration: 15-30 minutes
Impact: System-wide functionality

Procedure:
1. Revert infrastructure changes
2. Restore previous configurations
3. Restart affected services
4. Verify system health
5. Update monitoring systems

4. Full System Rollback

Scope: Complete system state
Duration: 1-4 hours
Impact: All system functionality

Procedure:
1. Activate disaster recovery protocol
2. Restore all components to previous state
3. Verify data consistency
4. Perform complete system validation
5. Notify all stakeholders

Rollback Procedures by Component

Application Code Rollback

Prerequisites

- Previous stable version tagged in Git
- Automated deployment pipeline available
- Database compatibility verified

- Rollback authorization obtained

Step-by-Step Process

1. **Identify Target Version**
   `bash`
   `git log --oneline --graph -10`
   `git show <previous-stable-commit>`

2. **Prepare Rollback**
   `bash`
   `# Create rollback branch`
   `git checkout -b rollback-to-<version>`
   `git reset --hard <previous-stable-commit>`

3. **Execute Rollback**
   ```bash
   # Deploy previous version
   ./scripts/deploy.sh production

# Verify deployment
curl -f https://africanmarket.com/api/health
```

1. **Verify Rollback**
   - Check application health endpoints
   - Verify core functionality works
   - Monitor error rates and performance
   - Test critical user journeys

Rollback Verification Checklist

- [ ] Application starts successfully
- [ ] Health checks pass
- [ ] User authentication works
- [ ] Order placement works
- [ ] Payment processing works
- [ ] Notifications are sent
- [ ] Admin functions accessible

Database Rollback

Prerequisites

- Recent database backup available
- Database migration history documented
- Application downtime window scheduled
- Data loss impact assessed

Step-by-Step Process

1. **Stop Application Services**
   `bash`
   `docker-compose -f docker-compose.production.yml stop app`

2. **Create Safety Backup**
   `bash`
   `./scripts/backup.sh production full`

3. **Identify Target Backup**
   `bash`
   `ls -la backups/database/ | grep production`

4. **Execute Database Restore**
   `bash`
   `./scripts/restore.sh backups/database/target-backup.sql.gz production`

5. **Verify Database State**
   ```sql
   – Check data integrity
   SELECT COUNT() FROM users;
   SELECT COUNT() FROM orders;
   SELECT COUNT(*) FROM products;

– Verify schema version
SELECT version FROM schema_migrations ORDER BY version DESC LIMIT 1;
```

1. **Restart Application**
   `bash`
   `docker-compose -f docker-compose.production.yml up -d`

Database Rollback Verification

- [ ] Database connection successful
- [ ] Core tables present and populated
- [ ] Data integrity constraints satisfied
- [ ] Application can read/write data
- [ ] No foreign key constraint errors
- [ ] Migration state is consistent

Configuration Rollback

Prerequisites

- Configuration files version controlled
- Environment variables documented
- Service configurations backed up
- Change impact analyzed

Step-by-Step Process

1. **Identify Configuration Changes**
   `bash`
   `git diff HEAD~1 -- *.env docker-compose*.yml nginx.conf`

2. **Revert Configuration Files**
   `bash`
   `git checkout HEAD~1 -- .env.production`
   `git checkout HEAD~1 -- docker-compose.production.yml`
   `git checkout HEAD~1 -- nginx.conf`

3. **Restart Affected Services**
   ```bash
   # Restart application
   docker-compose -f docker-compose.production.yml restart

# Restart nginx
sudo systemctl restart nginx
```

1. **Verify Configuration**
   - Check service health
   - Verify environment variables
   - Test external integrations
   - Monitor system behavior

Third-Party Service Rollback

Prerequisites

- Previous API versions available
- Service provider rollback support
- Integration testing completed
- Fallback mechanisms ready

Step-by-Step Process

1. **Identify Service Changes**
   - Review API version updates
   - Check configuration changes
   - Verify credential updates

2. **Revert Integration Settings**
   ```bash
   # Update environment variables
   export STRIPE_API_VERSION=previous-version
   export SENDGRID_API_VERSION=previous-version
   ```

3. **Update Application Code**
   ```bash
   # Revert API client versions
   git checkout HEAD~1 -- lib/stripe-service.js
   git checkout HEAD~1 -- lib/email-service.js
   ```

4. **Test Integrations**
   - Verify payment processing
   - Test email delivery
   - Check push notifications
   - Validate external APIs

Emergency Rollback Procedures

Critical System Failure

When the system is completely down or critically compromised:

Immediate Actions (0-5 minutes)

1. **Assess Situation**
   - Identify scope of failure
   - Determine user impact
   - Evaluate data integrity risk

2. **Activate Emergency Protocol**
   ```bash
   # Emergency system stop
   ./scripts/emergency-stop.sh

# Activate disaster recovery
./scripts/disaster-recovery.sh production full
```

1. **Notify Stakeholders**
   - Send critical alert to team
   - Update status page
   - Prepare customer communication

Recovery Actions (5-30 minutes)

1. **Execute Full Rollback**
   ```bash
   # Restore from latest stable backup
   ./scripts/restore.sh latest production
   ```

2. **Verify System Recovery**
   - Check all core functionality
   - Verify data consistency
   - Test critical user journeys

3. **Monitor System Stability**
   - Watch error rates
   - Monitor performance metrics
   - Check user activity

Data Corruption Rollback

When data integrity is compromised:

Immediate Actions

1. **Stop All Write Operations**
   ```bash
   # Enable read-only mode
   docker exec africanmarket_postgres psql -c "ALTER DATABASE africanmarket SET default_transaction_read_only = on;"
   ```

2. **Assess Corruption Scope**
   ```sql
   -- Check data consistency
   SELECT COUNT(*) FROM users WHERE created_at > updated_at;
   SELECT COUNT(*) FROM orders WHERE total < 0;
   ```

3. **Restore from Clean Backup**
   ```
   bash
   ./scripts/restore.sh backups/database/last-verified-backup.sql.gz production
   ```

Post-Rollback Procedures

Immediate Actions (0-30 minutes)

1. **System Verification**
   - [ ] All services running normally
   - [ ] No error spikes in monitoring
   - [ ] Core functionality operational
   - [ ] Performance within normal range

2. **User Communication**
   - [ ] Status page updated
   - [ ] Customer notification sent
   - [ ] Support team informed
   - [ ] Social media updated if needed

3. **Team Notification**
   - [ ] All team members notified
   - [ ] Rollback details shared
   - [ ] Next steps communicated
   - [ ] Monitoring assignments made

Short-term Actions (30 minutes - 2 hours)

1. **Root Cause Analysis**
   - [ ] Issue investigation started
   - [ ] Logs collected and analyzed
   - [ ] Timeline of events documented
   - [ ] Contributing factors identified

2. **Data Validation**
   - [ ] Data integrity verified
   - [ ] User accounts validated
   - [ ] Transaction consistency checked
   - [ ] System state confirmed

3. **Stakeholder Updates**
   - [ ] Management briefed
   - [ ] Customer updates provided
   - [ ] Partner notifications sent
   - [ ] Press communication if needed

Long-term Actions (2+ hours)

1. **Detailed Analysis**
   - [ ] Complete post-mortem prepared
   - [ ] Lessons learned documented
   - [ ] Process improvements identified
   - [ ] Prevention measures designed

2. **System Hardening**
   - [ ] Additional monitoring implemented
   - [ ] Testing procedures enhanced
   - [ ] Rollback procedures refined
   - [ ] Team training updated

Rollback Testing

Regular Testing Schedule

- **Monthly**: Test application rollback procedures
- **Quarterly**: Test database rollback procedures
- **Bi-annually**: Test full system rollback
- **Annually**: Test disaster recovery rollback

Testing Checklist

- [ ] Rollback procedures documented and current
- [ ] All team members trained on procedures
- [ ] Rollback tools and scripts functional
- [ ] Backup systems verified and accessible
- [ ] Communication templates ready
- [ ] Authority chains clearly defined

Test Scenarios

1. **Planned Rollback**: Simulated deployment failure
2. **Emergency Rollback**: Critical system failure simulation
3. **Partial Rollback**: Single component failure
4. **Data Rollback**: Database corruption simulation

Tools and Scripts

Automated Rollback Tools

- `./scripts/rollback.sh` - Main rollback orchestration
- `./scripts/emergency-stop.sh` - Emergency system shutdown
- `./scripts/restore.sh` - Database restoration
- `./scripts/disaster-recovery.sh` - Full system recovery

Monitoring and Alerting

- Health check endpoints for rapid status assessment
- Automated alerting for rollback trigger conditions
- Real-time monitoring dashboards
- Rollback success/failure notifications

Communication Tools

- Status page automation
- Slack integration for team notifications
- Email templates for customer communication
- Social media update procedures

Documentation and Training

Required Documentation

- [ ] Rollback procedures (this document)
- [ ] Emergency contact lists
- [ ] System architecture diagrams
- [ ] Database schema documentation
- [ ] Configuration management guide

Team Training Requirements

- [ ] All engineers trained on rollback procedures
- [ ] Regular rollback drills conducted
- [ ] Emergency response procedures practiced
- [ ] Communication protocols established
- [ ] Decision-making authority clarified

This rollback procedures document should be reviewed and updated regularly to ensure accuracy and effectiveness. All team members should be familiar with these procedures and participate in regular rollback drills.