Operational Safety Procedures - Phase 2 Day 2

Overview

This document outlines the comprehensive operational safety procedures implemented in Phase 2 Day 2 of the HX Infrastructure Ansible project. These procedures ensure safe, reliable, and auditable operations in production environments.

Safety Framework Components

1. Pre-Operation Safety Checks

Backup Verification

- **Automatic Backup Creation**: All operations requiring `safety_require_backup: true` will automatically create timestamped backups
- **Backup Integrity Verification**: System verifies backup creation and accessibility before proceeding
- **Backup Locations**:
- System configs: `/opt/ansible-backups/{timestamp}/etc-{timestamp}.tar.gz`
- Database: `/opt/ansible-backups/{timestamp}/database-{timestamp}.sql`
- Custom paths: Configurable via `safety_backup_paths`

Maintenance Window Validation

- **Production Restrictions**: Operations in production require maintenance window compliance
- **Default Window**: 02:00-06:00 UTC (configurable)
- **Override**: Emergency operations can override with explicit approval

Monitoring Integration

- **Alert Checking**: Verifies no critical alerts are active before operations
- **Cluster Health**: Validates Kubernetes cluster health if applicable
- **Service Status**: Confirms critical services are operational

2. Interactive Safety Confirmations

Enhanced Confirmation Prompts

```
╔══════════════════════════════════════════════════════════════╗
║                    🚨 CRITICAL OPERATION SAFETY CHECK 🚨                    ║
╠══════════════════════════════════════════════════════════════╣
║ Operation: SYSTEM_UPDATE
║ Target Host: prod-web-01.hana-x.ai
║ Environment: PRODUCTION
║ User: admin
║ Timestamp: 2025-09-18T10:30:00Z
║
║ POTENTIAL RISKS:
║   • System downtime
║   • Service interruption
║   • Configuration corruption
║
║ SAFETY MEASURES ACTIVE:
║   • Backup Created: YES
║   • Rollback Plan: YES
║   • Maintenance Window: YES
║
║ CONFIRMATION REQUIRED:
║ Type 'PROCEED_WITH_CAUTION' to continue or Ctrl+C to abort
╚══════════════════════════════════════════════════════════════╝
```

Confirmation Requirements

- **Production Environment**: Requires `PROCEED_WITH_CAUTION` confirmation
- **Staging Environment**: Requires `CONFIRM` confirmation
- **Development Environment**: Optional confirmation (configurable)

3. Dangerous Command Protection

Protected Commands

The system automatically blocks or warns about dangerous commands:
- `rm -rf` - Recursive file deletion
- `dd if=` - Direct disk operations
- `mkfs` - Filesystem creation
- `fdisk` - Disk partitioning
- `parted` - Partition management
- `wipefs` - Filesystem signature removal
- `shred` - Secure file deletion
- `format` - Disk formatting
- `truncate -s 0` - File truncation
- `>` - File redirection (potential overwrites)

Override Mechanism

```
safety_allow_dangerous: true  # Explicitly allow dangerous operations
```

4. Automated Rollback Scripts

Rollback Script Generation

Each operation with backups generates an automated rollback script:
- **Location**: `/var/log/ansible-safety/rollback-{timestamp}.sh`
- **Permissions**: `0750` (executable by root only)
- **Content**: Automated restoration procedures

Rollback Script Features

- Service stop/start procedures
- File restoration from backups
- Database restoration (if applicable)
- Verification steps
- Comprehensive logging

Manual Rollback Execution

```
sudo /var/log/ansible-safety/rollback-1726660200.sh
```

5. Comprehensive Logging

Safety Operation Logs

- **Location**: `/var/log/ansible-safety/`
- **Files**:
- `production-operations.log` - All production operations
- `dangerous-commands.log` - Dangerous command usage
- `rollback-{timestamp}.log` - Rollback execution logs

Log Format

```
2025-09-18T10:30:00Z - PRODUCTION_OP: System Update
User: admin | Host: prod-web-01.hana-x.ai | Backup: /opt/ansible-backups/1726660200
```

Production-Specific Safety Measures

1. Enhanced Production Controls

Mandatory Safety Features

- **Backup Requirement**: All operations must create backups
- **Maintenance Window**: Operations restricted to maintenance windows
- **Monitoring Checks**: Verify no critical alerts before operations
- **Confirmation Prompts**: Enhanced confirmation required
- **Rollback Scripts**: Automatic rollback script generation

Production Environment Detection

```
environment_type: production  # Triggers enhanced safety measures
```

2. SSH Key Management Safety

Key Rotation Procedures

- **Rotation Schedule**: Every 90 days (configurable)
- **Key Type**: ED25519 (256-bit)
- **Backup**: Old keys backed up before rotation
- **Verification**: New key connectivity tested before old key removal

Key Distribution Safety

- **Staged Deployment**: Keys distributed to test hosts first
- **Connectivity Verification**: Each host tested before proceeding
- **Rollback Capability**: Previous keys maintained during transition

3. Service Restart Safety

Pre-Restart Checks

- **Health Verification**: Service health confirmed before restart
- **Dependency Mapping**: Dependent services identified
- **Backup Creation**: Service configurations backed up

Restart Procedures

- **Graceful Shutdown**: Services stopped gracefully with timeout
- **Health Monitoring**: Service health monitored during restart
- **Rollback Triggers**: Automatic rollback if health checks fail

Safety Configuration Examples

Basic Safety Configuration

```yaml
# Enable all safety features
safety_confirmation_required: true
safety_require_backup: true
safety_require_maintenance_window: true
safety_dangerous_command_protection: true
safety_create_rollback_script: true

# Define operation details
safety_operation_name: "Database Schema Update"
safety_risks:
  - "Database downtime"
  - "Data corruption"
  - "Application errors"
```

Advanced Safety Configuration

```yaml
# Custom backup paths
safety_backup_paths:
  - "/etc/nginx"
  - "/opt/application/config"
  - "/var/lib/postgresql"

# Custom dangerous commands
safety_dangerous_commands:
  - "rm -rf"
  - "DROP DATABASE"
  - "TRUNCATE TABLE"

# Maintenance window
safety_maintenance_start: "01:00"
safety_maintenance_end: "05:00"

# Monitoring integration
safety_check_monitoring: true
monitoring_api_url: "http://prod-monitor-01.hana-x.ai:9090/api/v1/alerts"
```

Emergency Procedures

1. Emergency Override

For critical emergency operations outside maintenance windows:

```yaml
safety_emergency_override: true
safety_emergency_justification: "Critical security patch for active exploit"
```

2. Rollback Execution

If an operation fails or causes issues:

```bash
# Find the rollback script
ls -la /var/log/ansible-safety/rollback-*.sh

# Execute rollback
sudo /var/log/ansible-safety/rollback-{timestamp}.sh
```

3. Safety Bypass (Use with Extreme Caution)

```yaml
safety_confirmation_required: false
safety_require_backup: false
safety_dangerous_command_protection: false
```

Best Practices

1. Pre-Operation Planning

- Review all safety requirements before operations
- Ensure maintenance windows are scheduled
- Verify backup storage capacity

- Confirm rollback procedures

2. During Operations

- Monitor safety logs in real-time
- Verify each safety checkpoint
- Document any overrides or exceptions
- Maintain communication with team

3. Post-Operation Validation

- Verify operation success
- Confirm system health
- Review safety logs
- Archive rollback scripts
- Update documentation

Troubleshooting

Common Safety Issues

Backup Creation Failures

```
# Check backup directory permissions
ls -la /opt/ansible-backups/

# Check disk space
df -h /opt/ansible-backups/

# Manual backup creation
sudo mkdir -p /opt/ansible-backups/manual-$(date +%s)
sudo tar -czf /opt/ansible-backups/manual-$(date +%s)/etc-backup.tar.gz /etc
```

Maintenance Window Violations

```
# Check current time vs maintenance window
date
echo "Maintenance window: 02:00-06:00 UTC"

# Override for emergency (use carefully)
ansible-playbook -e "safety_emergency_override=true" playbook.yml
```

Confirmation Prompt Issues

```
# Check terminal settings
echo $TERM

# Use expect for automation (not recommended for production)
expect -c "
spawn ansible-playbook playbook.yml
expect \"Type 'PROCEED_WITH_CAUTION'\"
send \"PROCEED_WITH_CAUTION\r\"
interact
"
```

Compliance and Auditing

1. Audit Trail

All safety operations are logged with:
- Timestamp (ISO 8601 format)
- User identification
- Operation details
- Host information
- Backup locations
- Confirmation status

2. Compliance Reports

Generate compliance reports:

```
# Safety operations summary
grep "PRODUCTION_OP" /var/log/ansible-safety/production-operations.log | tail -20

# Dangerous command usage
cat /var/log/ansible-safety/dangerous-commands.log

# Rollback executions
ls -la /var/log/ansible-safety/rollback-*.log
```

3. Regular Safety Reviews

- Weekly review of safety logs
- Monthly review of rollback scripts
- Quarterly review of safety procedures
- Annual review of safety configuration

Integration with CI/CD

1. Pipeline Safety Checks

```
# .gitlab-ci.yml example
safety_validation:
  stage: validate
  script:
    - ansible-lint playbooks/
    - ansible-playbook --syntax-check --check playbooks/production.yml
    - python scripts/validate_safety_config.py
```

2. Automated Safety Testing

```
# Test safety procedures
test_safety:
  stage: test
  script:
    - ansible-playbook -e "safety_test_mode=true" playbooks/safety_test.yml
    - ./scripts/test_rollback_scripts.sh
```

This comprehensive safety framework ensures that all operations in the HX Infrastructure are performed with appropriate safeguards, documentation, and rollback capabilities.