Guile Deploy Ledger - Live Demo

Platform Engineering Team

2024

Outline

What We'll Demonstrate Today

Key Features

- Real-time deployment tracking
- Multi-environment support
- Rollback management
- Service dependency mapping
- Metrics and reporting

Demo Scenarios

1. Blue-green deployment to production
2. Canary rollout with monitoring
3. Emergency rollback procedure
4. Multi-service coordinated deployment
5. Deployment metrics analysis

Scenario Setup

Initial State

- Service: `api-gateway`
- Current version: `v2.3.4` (Blue)
- Target version: `v2.4.0` (Green)
- Environment: Production

Success Criteria

- Zero downtime
- Automatic rollback on failure
- Complete audit trail

Live Demonstration

Step 1: Record Deployment Start

```
(define deployment
  (make-deployment-event
    #:service-name "api-gateway"
    #:version "v2.4.0"
    #:environment 'production
    #:deployment-type 'blue-green
    #:initiator "jenkins-pipeline"))
```

Step 2: Monitor Progress

```
(display-deployment-status deployment)
;; Shows: pending TS1cmtt0→ in-progress → validating
```

Step 3: Complete Deployment

```
(update-deployment-status!
  deployment 'succeeded)
```

Results

Deployment Metrics

Metric	Value
Duration	4m 32s
Downtime	0s
Rollback Ready	Yes
Health Checks	100%

Audit Log

```
2024-03-15 10:23:45 - Deployment initiated
2024-03-15 10:24:12 - Blue environment validated
2024-03-15 10:26:30 - Green environment ready
2024-03-15 10:27:45 - Traffic switched
2024-03-15 10:28:17 - Deployment completed
```

Progressive Rollout Strategy

Configuration

```
(define canary-config
  '((stages . (1 5 25 50 100))
    (duration . 300)  ; 5 minutes per stage
    (metrics . (error-rate latency-p99))
    (threshold . 0.01)))
```

Monitoring Points

- Error rate $< 1\%$
- P99 latency < 200ms
- No critical alerts

Canary Progression

Stage Visualization

```
Stage 1 (1%):   []
Stage 2 (5%):   []
Stage 3 (25%):  []
Stage 4 (50%):  []
Stage 5 (100%): []
```

Decision Points

Each stage requires:

- Automated validation
- Manual approval (optional)
- Metric thresholds met

Canary Code Example

Implementation

```
(define (execute-canary-deployment service version)
  (let ((deployment (make-deployment-event
                     #:service-name service
                     #:version version
                     #:deployment-type 'canary)))
    (for-each
      (lambda (percentage)
        (format #t "Rolling out to ~a%~%" percentage)
        (update-canary-traffic deployment percentage)
        (sleep 300)  ; Wait 5 minutes
        (when (metrics-degraded? deployment)
          (rollback-deployment deployment)))
      '(1 5 25 50 100))
    deployment))
```

Incident Response

Problem Detection

```
ALERT: Error rate spike detected
Service: payment-processor
Current: v3.2.0
Error Rate: 15% (threshold: 1%)
Impact: HIGH
```

Immediate Action Required

- Rollback to last known good version
- Preserve diagnostic information
- Notify stakeholders

Rollback Execution

Command Sequence

```
;; 1. Create rollback event
(define rollback
  (make-rollback-event
    #:service-name "payment-processor"
    #:from-version "v3.2.0"
    #:to-version "v3.1.5"
    #:reason "Error rate spike - 15%"
    #:deployment-id "deploy-1234"))

;; 2. Execute rollback
(execute-rollback! rollback)

;; 3. Verify stability
(verify-service-health "payment-processor")
```

Post-Rollback Analysis

Timeline

```
10:45:00 - Deployment v3.2.0 completed
10:47:30 - Error rate increase detected
10:48:00 - Rollback initiated
10:49:15 - Rollback completed
10:50:00 - Service stabilized
```

Root Cause

- Database schema incompatibility
- Missing migration step
- Insufficient testing coverage

Coordinated Release

Service Dependencies

```
frontend-ui (v4.0.0)
    api-gateway (v2.4.0)
        auth-service (v1.8.0)
        user-service (v3.1.0)
    cdn-service (v1.2.0)
```

Deployment Order

1. Backend services (auth, user)
2. API Gateway
3. CDN Service
4. Frontend UI

Orchestration Code

Multi-Service Deployment

```
(define (deploy-service-graph services)
  (let ((deployment-order
         (topological-sort services)))
    (for-each
      (lambda (service)
        (let ((deployment
               (deploy-service service)))
          (wait-for-health-check service)
          (when (not (healthy? service))
            (rollback-all deployments))))
      deployment-order)))
```

Deployment Progress

Visual Status Board

Service	Version	Status	Progress
auth-service	v1.8.0	Complete	[]
user-service	v3.1.0	Complete	[]
api-gateway	v2.4.0	Deploying	[]
cdn-service	v1.2.0	Waiting	[]
frontend-ui	v4.0.0	Waiting	[]

Deployment Metrics Dashboard

Key Performance Indicators

```
Deployment Frequency: 12/day
Lead Time: 2.3 hours
MTTR: 8 minutes
Change Failure Rate: 2.1%
```

Weekly Trends

```
Deployments/Day
15 |
10 |
 5 |
 0 +---------------
   Mon Tue Wed Thu Fri
```

Service Health Matrix

Current Status

Service	Deployments	Success Rate	Avg Duration	Last Deploy
api-gateway	45	97.8%	4m 12s	2h ago
auth-service	23	100%	3m 45s	1d ago
user-service	34	94.1%	5m 20s	4h ago
payment	12	100%	8m 30s	2d ago
frontend	67	96.5%	2m 10s	30m ago

Query Examples

Finding Problem Deployments

```
;; Find failed deployments in last 24h
(list-deployments
  #:status 'failed
  #:since (hours-ago 24))

;; Services with high rollback rate
(filter
  (lambda (service)
    (> (rollback-rate service) 0.05))
  (list-services))
```

Interactive REPL Demo

Connect to System

```
$ guile -L src
scheme@(guile-user)>
(use-modules (deploy-ledger core types))
```

Create Live Deployment

We'll create a deployment in real-time and track its progress

Real-time Monitoring

Watch Deployment Progress

```
(define (watch-deployment id)
  (let loop ()
    (let ((dep (get-deployment id)))
      (display-status dep)
      (unless (final-state? dep)
        (sleep 1)
        (loop)))))
```

Sample Output

```
[10:23:45] Status: pending
[10:23:46] Status: initializing
[10:23:50] Status: deploying [25%]
[10:24:10] Status: deploying [50%]
[10:24:30] Status: deploying [75%]
[10:24:50] Status: validating
```

Plugin System

Custom Validators

```
(register-validator 'business-hours
  (lambda (deployment)
    (let ((hour (current-hour)))
      (and (>= hour 9) (<= hour 17)))))

(register-validator 'change-freeze
  (lambda (deployment)
    (not (blackout-period? (current-date)))))
```

Integration Examples

Slack Notifications

```
(add-deployment-hook 'slack-notify
  (lambda (event)
    (slack-send-message
      (format #f " Deployment ~a: ~a → ~a"
              (deployment-event-status event)
              (deployment-event-service-name event)
              (deployment-event-version event)))))
```

Prometheus Metrics

```
(export-metrics 'prometheus
  #:endpoint "/metrics"
  #:port 9090)
```

Key Takeaways

What We've Demonstrated

- Real-time deployment tracking
- Multiple deployment strategies
- Automated rollback capabilities
- Multi-service orchestration
- Comprehensive metrics

Benefits Realized

- Reduced deployment failures by 60%
- MTTR improved from 45min to 8min
- Complete audit trail for compliance
- Increased deployment frequency 3x

Next Steps

Try It Yourself

```
# Clone the repository
git clone https://github.com/org/guile-deploy-ledger

# Run the examples
make examples

# Start interactive REPL
make repl
```

Documentation

- User Guide: `docs/user-guide.org`
- API Reference: `docs/api-reference.org`
- Examples: `examples/`

Questions?

Thank you for attending!
Contact: platform-team@example.com
Slack: #deploy-ledger