ClawdBot Production Deployment Checklist

Document Version: 1.0.0
Last Updated: February 14, 2026
Owner: Engineering & Security Teams
Distribution: Engineering, DevOps, Security, SRE
Classification: Internal

Document Purpose

This checklist ensures safe, secure, and reliable deployments of ClawdBot components to production environments. Use this document before every production deployment to validate readiness and minimize risk.

Target Audience: Release engineers, DevOps engineers, SRE teams, security engineers

---

Quick Reference

Phase	Duration	Critical Items	Go/No-Go
Pre-Deployment	1-3 days	Code review, security scan, backup	☐
Deployment Preparation	2-4 hours	Environment prep, runbook, rollback plan	☐
Deployment Execution	30-120 min	Gradual rollout, health checks	☐
Post-Deployment	24 hours	Monitoring, validation, documentation	☐

Deployment Window: [Specify date/time and timezone]
Deployment Type: [Feature / Hotfix / Security Patch / Configuration Change]
Risk Level: [Low / Medium / High / Critical]

Phase 1: Pre-Deployment Planning

1.1 Code Quality & Testing

- [ ] **All code changes peer-reviewed** (minimum 2 approvals for high-risk)
- [ ] **CI/CD pipeline passed** (build, lint, unit tests, integration tests)
- [ ] **Code coverage maintained or improved** (minimum 80% for critical components)
- [ ] **Static code analysis passed** (no critical/high severity issues)
- [ ] **Dependency vulnerabilities scanned** (Snyk, Dependabot, or equivalent)
- [ ] **Breaking changes documented** with migration guide
- [ ] **API compatibility verified** (backward compatible or versioned)

Evidence Required:

- CI/CD pipeline run ID: _____
- Code review links: _____
- Test coverage report: _____

1.2 Security Validation

- [ ] **Security scan completed** (SAST, DAST, container scanning)
- [ ] **No critical or high vulnerabilities** (or accepted with risk exception)
- [ ] **Secrets rotated if needed** (API keys, database passwords)
- [ ] **Authentication/authorization tested** (RBAC, permissions)
- [ ] **Prompt injection defenses validated** (if AI components affected)
- [ ] **Input validation reviewed** (sanitization, validation logic)
- [ ] **MCP server security verified** (if deploying new skills/servers)
- [ ] **Security team sign-off obtained** (for high-risk deployments)

Security Scan Results:

- Tool: _____
- Scan ID: _____
- Critical issues: 0 High issues: ___
- Security approval: ☐ Approved by: _____

1.3 Infrastructure & Environment

- [ ] **Target environment verified** (correct cluster, namespace, region)
- [ ] **Resource capacity confirmed** (CPU, memory, storage available)
- [ ] **Database migration tested** (if applicable, dry-run completed)
- [ ] **Infrastructure as Code validated** (Terraform plan reviewed)
- [ ] **Network policies configured** (firewall rules, security groups)
- [ ] **SSL/TLS certificates valid** (expiry > 30 days)
- [ ] **DNS records verified** (if DNS changes required)
- [ ] **Load balancer health checks configured**

Infrastructure Details:

- Cluster: _____
- Namespace: _____
- Resource requests: CPU ___ Memory ___
- Resource limits: CPU ___ Memory ___

1.4 Compliance & Documentation

- [ ] **Change ticket created** (Jira, ServiceNow, or equivalent)
- [ ] **Deployment runbook prepared** (step-by-step instructions)
- [ ] **Rollback plan documented** (revert strategy, estimated time)
- [ ] **Communication plan ready** (stakeholder notifications)
- [ ] **Compliance requirements met** (SOC 2, ISO 27001, GDPR)
- [ ] **Release notes drafted** (user-facing and internal)
- [ ] **Architecture diagrams updated** (if infrastructure changes)

Documentation Links:

- Change ticket: _____
- Runbook location: _____
- Rollback plan: _____

Phase 2: Deployment Preparation

2.1 Pre-Deployment Backup

- [ ] **Database backup completed** (verified and tested)
- [ ] **Configuration backup saved** (ConfigMaps, Secrets)
- [ ] **Current deployment state documented** (image tags, replicas)
- [ ] **Backup retention verified** (stored in durable location)
- [ ] **Restore procedure tested** (dry-run in staging)

Backup Details:

- Database backup: _____
- Backup size: ___ GB
- Backup location: _____
- Verification status: ☐ Tested restore successfully

2.2 Environment Preparation

- [ ] **Staging deployment successful** (production-like environment)
- [ ] **Staging smoke tests passed** (critical user journeys validated)
- [ ] **Performance testing completed** (load testing, stress testing)
- [ ] **Database migrations applied to staging** (validated successfully)
- [ ] **Feature flags configured** (if using phased rollout)
- [ ] **Rate limits verified** (API rate limiting, throttling)
- [ ] **Monitoring dashboards prepared** (Grafana, Datadog, or equivalent)

Staging Validation:

- Staging deployment time: _____
- Staging environment: _____
- Test results: ☐ All critical paths passed

2.3 Team Coordination

- [ ] **Deployment team assembled** (roles assigned)
  - Deployment lead: _____
  - Engineering lead: _____
  - Security engineer: _____
  - SRE on-call: _____
- [ ] **Communication channels established** (Slack, Teams, war room)
- [ ] **Stakeholders notified** (internal teams, customers if downtime)
- [ ] **On-call engineers alerted** (ready for incident response)
- [ ] **Escalation path defined** (who to contact if issues arise)
- [ ] **Maintenance window communicated** (if applicable)

Team Availability:

- All team members confirmed available: ☐ Yes
- Backup personnel identified: ☐ Yes
- Communication channel: _____

2.4 Monitoring & Alerting

- [ ] **Baseline metrics captured** (current performance, error rates)
- [ ] **Custom alerts configured** (deployment-specific thresholds)
- [ ] **Logging levels adjusted** (increased verbosity if needed)
- [ ] **Dashboard bookmarks shared** (quick access for team)
- [ ] **Alerting channels verified** (PagerDuty, Opsgenie, Slack)
- [ ] **Incident response plan ready** (playbooks accessible)

Monitoring Setup:

- Primary dashboard: _____
- Alert channels: _____
- Baseline metrics recorded: ☐ Yes

Phase 3: Deployment Execution

3.1 Deployment Strategy

Selected Strategy: [Mark one]

- [ ] **Blue-Green Deployment** (full cutover after validation)
- [ ] **Canary Deployment** (gradual traffic shift: 5% → 25% → 50% → 100%)
- [ ] **Rolling Update** (progressive pod replacement)
- [ ] **Recreate** (downtime acceptable, simple replacement)

3.2 Pre-Deployment Checks

- [ ] **Final go/no-go decision** (deployment lead approval)
- [ ] **All prerequisites verified** (checklist items above completed)
- [ ] **Deployment window confirmed** (within maintenance window)
- [ ] **Emergency contact list accessible** (key personnel)
- [ ] **Rollback trigger criteria defined** (error rate > X%, latency > Y ms)

Go/No-Go Decision:

- Deployment lead: ☐ GO / ☐ NO-GO - Name: _____
- Security lead: ☐ GO / ☐ NO-GO - Name: _____
- Engineering lead: ☐ GO / ☐ NO-GO - Name: _____

3.3 Deployment Steps

Step 1: Initial Deployment

- [ ] **Deploy to canary/blue environment** (minimal traffic)
- [ ] **Verify pod startup** (all containers running, ready checks passing)
- [ ] **Check application logs** (no errors during initialization)
- [ ] **Validate health endpoints** (`/health`, `/ready` returning 200)
- [ ] **Test database connectivity** (connection pool healthy)
- [ ] **Verify external dependencies** (APIs, MCP servers accessible)

Deployment Commands (record for audit):

Example deployment command

kubectl apply -f deployment.yaml --record

Deployment timestamp:

_____

Step 2: Smoke Testing

- [ ] **Execute smoke test suite** (critical path validation)
  - [ ] User authentication works
  - [ ] AI agent responds to prompts
  - [ ] MCP skills function correctly
  - [ ] API endpoints return expected responses
  - [ ] Database reads/writes successful
- [ ] **Verify security controls** (authentication, authorization)
- [ ] **Check performance metrics** (response times within SLA)

Smoke Test Results:

- Tests passed: ___ / ___
- Test duration: ___ minutes
- Issues identified: _____

Step 3: Gradual Traffic Shift (Canary/Blue-Green)

Canary Progression:

- [ ] **5% traffic to canary** - Wait 15 minutes, monitor
  - Error rate: ____% (baseline: ____%)
  - Latency p95: ___ ms (baseline: ___ ms)
  - Decision: ☐ Proceed / ☐ Rollback
- [ ] **25% traffic to canary** - Wait 15 minutes, monitor
  - Error rate: ____%
  - Latency p95: ___ ms
  - Decision: ☐ Proceed / ☐ Rollback
- [ ] **50% traffic to canary** - Wait 30 minutes, monitor
  - Error rate: ____%
  - Latency p95: ___ ms

    - Decision: ☐ Proceed / ☐ Rollback
- [ ] **100% traffic to canary** - Full deployment
    - Error rate: ____%
    - Latency p95: ___ ms
    - Status: ☐ Success / ☐ Rollback initiated

Step 4: Post-Deployment Validation

- [ ] **All pods healthy** (desired replicas = available replicas)
- [ ] **No error spikes** (error rate within normal range)
- [ ] **Latency within SLA** (p50, p95, p99 acceptable)
- [ ] **Memory/CPU usage normal** (no resource exhaustion)
- [ ] **Database performance stable** (query times, connection count)
- [ ] **No alert storms** (monitoring systems stable)
- [ ] **User-facing features verified** (sample user journeys tested)

Validation Results:

- All validation checks passed: ☐ Yes / ☐ No
- Issues requiring attention: _____

Phase 4: Post-Deployment Activities

4.1 Immediate Post-Deployment (First Hour)

- [ ] **Active monitoring** (team watching dashboards)
- [ ] **Log analysis** (reviewing application logs for errors)
- [ ] **User feedback monitoring** (support tickets, social media)
- [ ] **Performance comparison** (current vs baseline metrics)
- [ ] **Security monitoring** (unusual access patterns, anomalies)

Monitoring Window: [Start time] to [End time]
Assigned monitor: _____

4.2 Extended Monitoring (24 Hours)

- [ ] **Error rate trending** (stable or declining)
- [ ] **Resource utilization** (no memory leaks, CPU spikes)
- [ ] **Database health** (query performance, replication lag)
- [ ] **External service health** (dependencies functioning normally)

- [ ] **User behavior analysis** (adoption of new features)
- [ ] **Cost monitoring** (infrastructure costs within budget)

24-Hour Status:

- System stability: ☐ Stable / ☐ Concerns
- Issues encountered: _____
- Actions taken: _____

4.3 Documentation & Communication

- [ ] **Deployment completed notification sent** (to stakeholders)
- [ ] **Post-deployment report created** (successes, issues, metrics)
- [ ] **Release notes published** (customer-facing documentation)
- [ ] **Internal wiki updated** (configuration, architecture changes)
- [ ] **Lessons learned documented** (what went well, improvements)
- [ ] **Change ticket closed** (with final status and notes)

Communication Checklist:

- [ ] Engineering team notified
- [ ] Product team notified
- [ ] Customer success notified
- [ ] Support team trained (if user-facing changes)
- [ ] Status page updated (if applicable)

4.4 Cleanup & Housekeeping

- [ ] **Old deployment cleaned up** (blue environment decommissioned)
- [ ] **Temporary resources removed** (test databases, staging artifacts)
- [ ] **Feature flags removed** (if fully rolled out)
- [ ] **Monitoring restored to normal** (reduced logging verbosity)
- [ ] **Backup retention verified** (old backups archived properly)
- [ ] **Security scan post-deployment** (verify production state)

Rollback Procedures

Rollback Decision Criteria

Initiate rollback immediately if:

- Error rate > 5% (or 2x baseline)
- Latency p95 > 2000ms (or 3x baseline)
- Critical security vulnerability discovered
- Data integrity issues detected
- Unrecoverable application errors
- Stakeholder escalation to abort

Rollback Execution

- [ ] **Rollback decision made** (deployment lead authorization)
- [ ] **Communication sent** ("Rolling back deployment due to [reason]")
- [ ] **Revert to previous deployment**

  # Rollback command

  kubectl rollout undo deployment/clawdbot-gateway

  # Rollback timestamp:

  _____

- [ ] **Verify rollback success** (previous version running)
- [ ] **Validate system health** (errors resolved, metrics normal)
- [ ] **Database rollback if needed** (restore from backup)
- [ ] **Incident report created** (document root cause)

Rollback Details:

- Rollback initiated: ☐ Yes - Reason: _____
- Rollback completed: ___ minutes
- System restored: ☐ Yes / ☐ Partial / ☐ No

Emergency Contacts

Escalation Path

Role	Name	Contact	Escalation Criteria
Deployment Lead	_____	_____	First point of contact
Engineering Manager	_____	_____	Deployment failures
CISO	_____	_____	Security incidents
VP Engineering	_____	_____	Critical system outage
On-Call SRE	_____	_____	Infrastructure issues
Database Admin	_____	_____	Data integrity issues

External Contacts

- **Cloud Provider Support**: _____
- **Security Vendor**: _____
- **Incident Response Hotline**: _____

Deployment Metrics

Success Criteria

- [ ] **Deployment completed within time window** (target: 2 hours)
- [ ] **Zero unplanned rollbacks**
- [ ] **Error rate < 0.1%** (post-deployment)
- [ ] **Latency p95 < 500ms** (within SLA)
- [ ] **Zero security incidents**
- [ ] **User satisfaction maintained** (no increase in support tickets)

Deployment Statistics

Metric	Target	Actual	Status
Deployment Duration	< 2 hours	___ min	☐
Error Rate	< 0.1%	___%	☐
Latency p95	< 500ms	___ ms	☐
Rollback Required	No	Yes/No	☐
Downtime	0 minutes	___ min	☐
Issues Found	0	___	☐

Sign-Off

Deployment Approval

Role	Name	Signature	Date/Time
Deployment Lead	_____	_____	_____
Security Engineer	_____	_____	_____
Engineering Manager	_____	_____	_____

Post-Deployment Sign-Off

Role	Name	Signature	Date/Time
Deployment Lead	_____	_____	_____
SRE On-Call	_____	_____	_____
Engineering Manager	_____	_____	_____

Deployment Status: ☐ Success / ☐ Success with Issues / ☐ Rolled Back

---

Appendix A: Component-Specific Checklists

A.1 ClawdBot Gateway Deployment

- [ ] API rate limits configured
- [ ] Authentication middleware verified
- [ ] CORS policies validated
- [ ] Request logging enabled
- [ ] Circuit breakers configured

A.2 ClawdBot Agent Deployment

- [ ] LLM API keys rotated and verified
- [ ] Prompt injection defenses active
- [ ] Context window limits enforced
- [ ] Agent memory persistence tested
- [ ] Fallback responses configured

A.3 MCP Server Deployment

- [ ] Skill manifest validated
- [ ] Tool permissions reviewed
- [ ] Sandbox isolation verified
- [ ] Resource limits enforced
- [ ] Skill discovery tested

A.4 Database Migration

- [ ] Migration script tested in staging
- [ ] Rollback script prepared
- [ ] Data integrity checks defined
- [ ] Replication lag acceptable
- [ ] Backup verified before migration

Appendix B: Deployment Runbook Template

Pre-Deployment

1. Verify current state

kubectl get deployments -n clawdbot-prod
kubectl get pods -n clawdbot-prod

2. Create backup

kubectl get deployment clawdbot-gateway -n clawdbot-prod -o yaml > backup-gateway.yaml

3. Verify image availability

docker pull clawdbot/gateway:v1.2.3

Deployment

4. Apply new deployment

kubectl apply -f deployment-v1.2.3.yaml --record

5. Monitor rollout

kubectl rollout status deployment/clawdbot-gateway -n clawdbot-prod

6. Verify pods

kubectl get pods -n clawdbot-prod -l app=clawdbot-gateway

Validation

7. Check logs

kubectl logs -f deployment/clawdbot-gateway -n clawdbot-prod

8. Health check

curl -v https://api.clawdbot.example.com/health

9. Test critical endpoints

curl -H "Authorization: Bearer $TOKEN" https://api.clawdbot.example.com/v1/agents

Rollback (if needed)

10. Rollback to previous version

kubectl rollout undo deployment/clawdbot-gateway -n clawdbot-prod

11. Verify rollback

kubectl rollout status deployment/clawdbot-gateway -n clawdbot-prod

---

Appendix C: Monitoring Queries

Error Rate

Error rate percentage

```
sum(rate(http_requests_total{status=~"5.."}[5m])) /
sum(rate(http_requests_total[5m])) * 100
```

Latency p95

95th percentile latency

```
histogram_quantile(0.95,
sum(rate(http_request_duration_seconds_bucket[5m])) by (le))
```

Pod Health

Check pod status

```
kubectl get pods -n clawdbot-prod -l app=clawdbot-gateway
-o jsonpath='{range .items[*]}{.metadata.name}{"\t"}{.status.phase}
{"\n"}{end}'
```

References

1. ClawdBot Architecture Documentation
2. Security Hardening Guide
3. Incident Response Plan
4. Monitoring & Alerting Guide
5. Rollback Procedures

Document Control

Version	Date	Author	Changes
1.0.0	2026-02-14	Security Team	Initial release

Classification: Internal
Distribution: Engineering, DevOps, Security, SRE

Review Cycle: Quarterly
Next Review: 2026-05-14