

Slash Commands Reference

antigravity-jules-orchestration

Version 2.5.0 | Generated: 2025-12-17

Table of Contents

1. Core Commands
 - /status, /quick-fix, /session, /batch
2. Workflow Commands
 - /audit, /deploy-check, /implement-feature, /fix-issues
3. Security & Testing
 - /security, /test
4. Utility Commands
 - /learn-pattern, /generate-command
5. Quick Reference Table
6. Recommended Workflows
7. MCP Tools Integration

1. Core Commands

/status

Get a comprehensive overview of all Jules sessions, system health, and orchestration status

```
/status
```

Output:

- Active sessions with state
- Session statistics (total, completed, in progress, failed)
- System health (circuit breaker, cache, rate limits)
- Quick action suggestions

/quick-fix [file] [description]

Fast, streamlined workflow for single-file fixes using Jules autonomous coding

```
/quick-fix src/api/auth.js "Add rate limiting"
```

Features:

- Auto-selects repository
- Creates focused session
- Skips plan approval for speed
- Auto-creates PR

/session [id] [action]

Quick session management for Jules coding sessions

```
/session ses_abc123 approve
```

Actions: view (default), approve, cancel, retry, diff

/batch [label] [repo?]

Quick batch session creation from GitHub issue labels

```
/batch jules-auto
```

Common Labels: jules-auto, bug, enhancement, security

2. Workflow Commands

/audit

Run a comprehensive parallel audit of the entire repository.

Parallel Agents:

1. Security Audit - vulnerabilities, secrets, auth patterns
2. Code Quality Review - error handling, async patterns, style
3. Dependency Analysis - outdated, vulnerabilities, unused
4. API Endpoint Review - validation, status codes, rate limiting
5. Documentation Completeness - accuracy, coverage

/deploy-check

Pre-deployment validation with live health checks.

Checks:

- Git status (uncommitted changes)
- All tests passing
- No high/critical vulnerabilities
- Health endpoint responding
- All services configured

/implement-feature [description]

Feature implementation workflow with planning

```
/implement-feature "Add webhook retry mechanism"
```

Steps: Analyze requirements, Create plan, Generate code, Create tests, Update docs

/fix-issues

Auto-diagnose and fix common issues.

Fixes: TypeScript errors, Linting issues, Failing tests, Outdated dependencies, Missing imports

3. Security & Testing

/security [scope]

Dedicated security scanning

```
/security quick
```

Scope Options:

- full - Complete security audit (default)
- quick - Critical issues only
- deps - npm audit
- secrets - Credential scanning
- api - Endpoint security testing

/test [scope] [options]

Run all tests with coverage and detailed reporting

```
/test all --coverage
```

Scope: all (default), backend, dashboard, unit, integration

4. Quick Reference Table

| Command | Purpose | Example |
|---------------------------------|-----------------------|---------------------------------------|
| <code>/status</code> | System overview | <code>/status</code> |
| <code>/quick-fix</code> | Fast single-file fix | <code>/quick-fix file.js "fix"</code> |
| <code>/session</code> | Session management | <code>/session id approve</code> |
| <code>/batch</code> | Batch from labels | <code>/batch jules-auto</code> |
| <code>/audit</code> | Full audit | <code>/audit</code> |
| <code>/deploy-check</code> | Pre-deploy validation | <code>/deploy-check</code> |
| <code>/implement-feature</code> | Feature workflow | <code>/implement-feature "X"</code> |
| <code>/fix-issues</code> | Auto-fix problems | <code>/fix-issues</code> |
| <code>/security</code> | Security scan | <code>/security quick</code> |
| <code>/test</code> | Run tests | <code>/test all</code> |

5. Recommended Workflows

Daily Development

1. `/status` - Check orchestration state
2. `/quick-fix` - Make targeted fixes
3. `/test` - Verify changes
4. `/deploy-check` - Pre-deployment validation

Batch Processing

1. `/batch jules-auto` - Process labeled issues
2. `/status` - Monitor progress
3. `/session [id]` - Review individual sessions
4. `/session [id] approve` - Approve plans

Security Review

1. `/security quick` - Fast critical scan
2. `/security deps` - Check dependencies
3. `/audit` - Full comprehensive audit
4. `/fix-issues` - Auto-fix what's possible

Feature Implementation

1. `/implement-feature` - Plan and implement
2. `/test` - Verify tests pass
3. `/security` - Security check
4. `/deploy-check` - Ready for deployment

6. MCP Tools Integration

These commands leverage the 45 MCP tools available in v2.5.0:

Jules Core:

```
jules_list_sources, jules_create_session, jules_list_sessions, jules_get_session,  
jules_send_message, jules_approve_plan, jules_get_activities
```

Session Management:

```
jules_cancel_session, jules_retry_session, jules_get_diff, jules_delete_session,  
jules_cancel_all_active
```

Session Templates (NEW):

```
jules_create_template, jules_list_templates, jules_create_from_template, jules_delete_template
```

Session Cloning & Search (NEW):

```
jules_clone_session, jules_search_sessions
```

PR Integration (NEW):

```
jules_get_pr_status, jules_merge_pr, jules_add_pr_comment
```

Session Queue (NEW):

```
jules_queue_session, jules_get_queue, jules_process_queue, jules_clear_queue
```

Batch Processing:

```
jules_create_from_issue, jules_batch_from_labels, jules_batch_create, jules_batch_status,  
jules_batch_approve_all, jules_list_batches, jules_batch_retry_failed
```

Analytics (NEW):

```
jules_get_analytics
```

Monitoring & Cache:

```
jules_monitor_all, jules_session_timeline, jules_cache_stats, jules_clear_cache
```

Ollama LLM:

```
ollama_list_models, ollama_completion, ollama_code_generation, ollama_chat
```

RAG:

```
ollama_rag_index, ollama_rag_query, ollama_rag_status, ollama_rag_clear
```