# Atomic Task Decomposition — Combined Unified Workflow (Extended)

This document extends the earlier atomic decomposition to include additional tasks introduced by the **Simultaneous Execution with Git Worktrees** workflow. The original unified pipeline (from COMBINED.nested.yaml, COMBINED.deepmerge.yaml and COMBINED.multi.yaml) defined atoms up through atom\_129. New atoms begin at atom\_130 and correspond to multi‑stream creation, parallel execution with priority queues, merge coordination and Git worktree isolation. Each atom retains the single‑responsibility principle and is assigned to a role that best reflects ownership of the operation.

## Carry‑forward Atoms (1‒129)

For completeness the atoms from the existing unified workflow are preserved in prior files. Atoms 1‒129 are defined in atomic\_decomposition\_combined\_updated.md and are unchanged here.

## Simultaneous Execution & Git Worktree Workflow

These atoms represent the lifecycle of simultaneous work‑stream execution, parallel dispatch, merge queue management and final verification. They are derived from the **Simultaneous execution with get work trees.md** description, which outlines how workstreams are created, queued, executed in parallel via a priority queue, merged and persisted. Each step is decomposed into atomic operations with a single responsible role.

atom\_130: Work Stream Creation: analyze\_file\_scope\_requirements | Role: planning\_ai  
atom\_131: Work Stream Creation: identify\_non\_conflicting\_streams | Role: planning\_ai  
atom\_132: Work Stream Creation: create\_workstream\_record | Role: orchestrator  
atom\_133: Work Stream Creation: assign\_stream\_priorities | Role: orchestrator  
atom\_134: Work Stream Creation: configure\_multistream\_definitions | Role: orchestrator  
atom\_135: Execution Planning: define\_execution\_order | Role: orchestrator  
atom\_136: Execution Planning: manage\_path\_claims | Role: merge\_coordinator  
atom\_137: Execution Planning: allocate\_git\_worktrees | Role: repo\_ai  
atom\_138: Execution Planning: manage\_dependencies | Role: planning\_ai  
atom\_139: Work Stream Execution: list\_available\_streams | Role: orchestrator  
atom\_140: Work Stream Execution: queue\_workstream\_priority | Role: orchestrator  
atom\_141: Work Stream Execution: dispatch\_parallel\_execution | Role: orchestrator  
atom\_142: Work Stream Execution: spawn\_worker\_threads | Role: orchestrator  
atom\_143: Work Stream Execution: execute\_stream | Role: orchestrator  
atom\_144: Work Stream Execution: retrieve\_phases\_for\_stream | Role: orchestrator  
atom\_145: Work Stream Execution: execute\_phase | Role: orchestrator  
atom\_146: Work Stream Execution: track\_phase\_completion | Role: orchestrator  
atom\_147: Work Stream Execution: update\_workstream\_status | Role: orchestrator  
atom\_148: Work Stream Execution: save\_artifacts\_to\_directory | Role: orchestrator  
atom\_149: Work Stream Execution: create\_merge\_queue\_item | Role: merge\_coordinator  
atom\_150: Merge Queue: pre\_merge\_lint\_check | Role: qa\_test\_agent  
atom\_151: Merge Queue: pre\_merge\_test\_check | Role: qa\_test\_agent  
atom\_152: Merge Queue: pre\_merge\_security\_check | Role: security\_compliance  
atom\_153: Merge Queue: estimate\_wait\_time | Role: merge\_coordinator  
atom\_154: Merge Queue: detect\_conflicts | Role: merge\_coordinator  
atom\_155: Merge Queue: merge\_branches | Role: merge\_coordinator  
atom\_156: Merge Queue: update\_merge\_status | Role: merge\_coordinator  
atom\_157: Persistence: update\_state\_files | Role: orchestrator  
atom\_158: Persistence: update\_merge\_queue\_state\_file | Role: merge\_coordinator  
atom\_159: Monitoring: display\_real\_time\_status | Role: orchestrator  
atom\_160: Monitoring: perform\_health\_check | Role: qa\_test\_agent  
atom\_161: IPT Verification: verify\_execution\_results | Role: planning\_ai  
atom\_162: Work Stream Cleanup: finalize\_and\_cleanup | Role: orchestrator

### Notes on Roles and Responsibilities

* **planning\_ai** handles the initial analysis: understanding file scopes, identifying work stream boundaries, ensuring dependencies are respected and verifying results at the end of execution. These tasks require reasoning over the project structure and planning concurrency.
* **orchestrator** manages most of the operational flow. It creates workstream records in the database, assigns priorities, reads multi‑stream definitions, determines execution order, lists streams, enqueues and dispatches work streams, spawns workers, executes phases, tracks completion and updates persistent state files. Orchestrator also displays real‑time status to users.
* **merge\_coordinator** focuses on merge‑specific concerns: creating items in the merge queue, performing pre‑merge checks (with help from QA and security roles), estimating wait times, detecting conflicts, merging branches and updating merge statuses. It also maintains the merge queue state file and manages exclusive path claims to avoid concurrency issues.
* **repo\_ai** is responsible for Git‑related operations such as allocating isolated worktrees for each stream.
* **qa\_test\_agent** runs linting and test checks both in the work streams and during the merge queue pre‑merge phase, and performs periodic health checks on the project structure.
* **security\_compliance** conducts pre‑merge security reviews as part of the merge queue.

This extended atom list ensures that the simultaneous execution workflow is explicitly represented in the unified atomic pipeline, enabling deterministic scheduling, clear role assignment and reliable parallel development across isolated worktrees.

## Deterministic GitHub Operations Module

The **Deterministic GitHub Operations Module** defines a suite of Git/GitHub commands that ensure repository actions are atomic, repeatable and invisible to users. Each command replaces error‑prone manual steps with scripted sequences that the pipeline can invoke on demand. The atoms below decompose each command’s implementation into single‑responsibility operations and assign roles that align with our existing agents.

atom\_163: gh-checkpoint: validate\_working\_directory | Role: repo\_ai  
atom\_164: gh-checkpoint: stage\_all\_changes | Role: repo\_ai  
atom\_165: gh-checkpoint: generate\_commit\_message\_from\_context | Role: docs\_summarizer  
atom\_166: gh-checkpoint: create\_atomic\_commit | Role: repo\_ai  
atom\_167: gh-checkpoint: push\_changes\_safe | Role: repo\_ai  
atom\_168: gh-checkpoint: verify\_remote\_sync | Role: repo\_ai  
atom\_169: gh-checkpoint: tag\_checkpoint\_with\_timestamp | Role: repo\_ai  
atom\_170: gh-checkpoint: update\_checkpoint\_registry | Role: orchestrator  
atom\_171: gh-init-workspace: clone\_or\_pull\_repository | Role: repo\_ai  
atom\_172: gh-init-workspace: configure\_git\_identity | Role: repo\_ai  
atom\_173: gh-init-workspace: setup\_remote\_tracking | Role: repo\_ai  
atom\_174: gh-init-workspace: checkout\_base\_branch | Role: repo\_ai  
atom\_175: gh-init-workspace: fetch\_latest\_state | Role: repo\_ai  
atom\_176: gh-init-workspace: create\_feature\_branch\_and\_worktree | Role: repo\_ai  
atom\_177: gh-init-workspace: configure\_git\_hooks | Role: repo\_ai  
atom\_178: gh-init-workspace: validate\_workspace\_ready | Role: repo\_ai  
atom\_179: gh-sync-remote: stash\_uncommitted\_changes | Role: repo\_ai  
atom\_180: gh-sync-remote: fetch\_all\_remotes | Role: repo\_ai  
atom\_181: gh-sync-remote: detect\_remote\_branch\_exists | Role: repo\_ai  
atom\_182: gh-sync-remote: reconcile\_diverged\_history | Role: repo\_ai  
atom\_183: gh-sync-remote: push\_local\_commits\_safe | Role: repo\_ai  
atom\_184: gh-sync-remote: apply\_stashed\_changes | Role: repo\_ai  
atom\_185: gh-sync-remote: verify\_sync\_status | Role: repo\_ai  
atom\_186: gh-merge-workstreams: checkout\_integration\_branch | Role: merge\_coordinator  
atom\_187: gh-merge-workstreams: validate\_workstream\_branches\_current | Role: merge\_coordinator  
atom\_188: gh-merge-workstreams: analyze\_changes\_per\_branch | Role: merge\_coordinator  
atom\_189: gh-merge-workstreams: detect\_potential\_conflicts | Role: merge\_coordinator  
atom\_190: gh-merge-workstreams: merge\_branch\_with\_strategy | Role: merge\_coordinator  
atom\_191: gh-merge-workstreams: run\_validation\_after\_merge | Role: qa\_test\_agent  
atom\_192: gh-merge-workstreams: categorize\_detected\_conflicts | Role: merge\_coordinator  
atom\_193: gh-merge-workstreams: apply\_conflict\_resolution\_rules | Role: merge\_coordinator  
atom\_194: gh-merge-workstreams: run\_linters\_on\_merged\_state | Role: qa\_test\_agent  
atom\_195: gh-merge-workstreams: run\_quick\_validation\_suite | Role: qa\_test\_agent  
atom\_196: gh-merge-workstreams: create\_merge\_commit\_summary | Role: docs\_summarizer  
atom\_197: gh-merge-workstreams: push\_integration\_branch | Role: merge\_coordinator  
atom\_198: gh-create-pr: validate\_branch\_current | Role: repo\_ai  
atom\_199: gh-create-pr: check\_existing\_pr | Role: repo\_ai  
atom\_200: gh-create-pr: generate\_pr\_title | Role: docs\_summarizer  
atom\_201: gh-create-pr: generate\_pr\_description | Role: docs\_summarizer  
atom\_202: gh-create-pr: attach\_artifacts\_to\_pr | Role: qa\_test\_agent  
atom\_203: gh-create-pr: create\_or\_update\_pr | Role: repo\_ai  
atom\_204: gh-create-pr: assign\_reviewers | Role: orchestrator  
atom\_205: gh-create-pr: add\_labels\_to\_pr | Role: orchestrator  
atom\_206: gh-create-pr: link\_related\_issues | Role: docs\_summarizer  
atom\_207: gh-create-pr: set\_pr\_milestone | Role: orchestrator  
atom\_208: gh-create-pr: enable\_auto\_merge | Role: merge\_coordinator  
atom\_209: gh-cleanup-branches: verify\_merge\_to\_main | Role: merge\_coordinator  
atom\_210: gh-cleanup-branches: list\_feature\_branches | Role: repo\_ai  
atom\_211: gh-cleanup-branches: delete\_merged\_branches | Role: repo\_ai  
atom\_212: gh-cleanup-branches: flag\_unmerged\_branches\_for\_review | Role: merge\_coordinator  
atom\_213: gh-cleanup-branches: remove\_worktrees | Role: repo\_ai  
atom\_214: gh-cleanup-branches: prune\_remote\_tracking\_branches | Role: repo\_ai  
atom\_215: gh-cleanup-branches: cleanup\_git\_reflog | Role: repo\_ai  
atom\_216: gh-cleanup-branches: archive\_branch\_metadata | Role: docs\_summarizer  
atom\_217: gh-rollback: identify\_rollback\_target | Role: orchestrator  
atom\_218: gh-rollback: validate\_rollback\_target | Role: repo\_ai  
atom\_219: gh-rollback: create\_rollback\_branch | Role: repo\_ai  
atom\_220: gh-rollback: revert\_main\_branch\_with\_commits | Role: repo\_ai  
atom\_221: gh-rollback: validate\_reverted\_state | Role: qa\_test\_agent  
atom\_222: gh-rollback: create\_rollback\_pr | Role: merge\_coordinator  
atom\_223: gh-rollback: reset\_feature\_branch\_to\_target | Role: repo\_ai  
atom\_224: gh-rollback: force\_push\_rollback\_branch | Role: repo\_ai  
atom\_225: gh-rollback: update\_rollback\_registry | Role: orchestrator  
atom\_226: gh-rollback: notify\_stakeholders | Role: planning\_ai  
atom\_227: gh-status-check: check\_local\_workspace\_status | Role: qa\_test\_agent  
atom\_228: gh-status-check: check\_remote\_branch\_status | Role: qa\_test\_agent  
atom\_229: gh-status-check: check\_pr\_status | Role: qa\_test\_agent  
atom\_230: gh-status-check: check\_workflow\_runs\_status | Role: qa\_test\_agent  
atom\_231: gh-status-check: generate\_status\_report | Role: docs\_summarizer  
atom\_232: Integration Strategy: define\_checkpoint\_schedule | Role: orchestrator  
atom\_233: Integration Strategy: orchestrate\_sync\_remote\_before\_integration | Role: repo\_ai  
atom\_234: Integration Strategy: orchestrate\_pr\_creation\_phase | Role: merge\_coordinator  
atom\_235: Integration Strategy: orchestrate\_cleanup\_after\_completion | Role: repo\_ai  
atom\_236: Error Handling: apply\_network\_retry\_strategy | Role: resilience\_agent  
atom\_237: Error Handling: handle\_conflict\_resolution\_failure | Role: merge\_coordinator  
atom\_238: Error Handling: handle\_push\_rejection\_strategies | Role: repo\_ai  
atom\_239: Observability: track\_git\_operations\_metrics | Role: cost\_resource\_manager  
atom\_240: Observability: maintain\_git\_audit\_trail | Role: orchestrator

### Module Notes

* **gh-checkpoint** encapsulates atomic save points. It validates the working tree, stages changes, generates a contextual commit message, commits, pushes safely, tags the checkpoint and records it in a registry. These operations are essential for reproducibility and automatic recovery in long‑running pipelines.
* **gh-init-workspace** prepares an isolated workspace: cloning or updating the repository, configuring user identity, setting remotes, switching to the base branch, fetching, creating a feature branch/worktree, installing hooks and verifying readiness. This ensures each workstream starts from a clean, configured environment.
* **gh-sync-remote** keeps local and remote branches synchronized. It stashes changes, fetches remotes, compares branch histories, automatically reconciles divergences, pushes local commits and reapplies any stashed work. This prevents stale branches and unexpected conflicts.
* **gh-merge-workstreams** merges parallel workstream branches into an integration branch. It validates branch currency, analyzes changes, detects conflicts, merges with patience strategies, runs per‑branch and final validations, resolves conflicts according to defined rules, creates a merge summary and pushes the integration branch.
* **gh-create-pr** automates pull request creation and updates: ensuring the branch is current, generating informative titles and descriptions, attaching test and performance artifacts, creating or updating PRs, assigning reviewers, applying labels and enabling auto‑merge when conditions permit.
* **gh-cleanup-branches** cleans up after a successful merge by verifying merges, deleting merged branches, flagging unmerged ones, removing worktrees, pruning remote tracking branches, cleaning the reflog and archiving metadata for audit.
* **gh-rollback** provides controlled rollback capabilities by identifying and validating rollback targets, creating rollback branches, reverting or resetting state, running validations on the reverted state, opening rollback PRs, updating the registry and notifying stakeholders. It is careful to preserve history on main branches.
* **gh-status-check** offers a comprehensive status snapshot by checking local workspace cleanliness, remote branch status, PR status, workflow runs and synthesizing the findings into a report.
* **Integration Strategy** atoms instruct the orchestrator when to invoke these commands within the main pipeline (e.g., scheduling checkpoints after certain phases, syncing before integration, creating PRs at Phase 4 and performing cleanup at the end).
* **Error Handling** atoms implement resilience: applying exponential backoff on network issues, escalating unresolved conflicts and handling push rejections gracefully.
* **Observability** atoms record metrics and maintain a detailed audit trail of every Git operation, supporting monitoring and compliance requirements.

## Audit‑Derived Workstream Generation

The following atoms extend the workflow to incorporate gap auditing and automated workstream generation as described in the **workstream\_from\_audit.txt** specification. Phase 1 analyses an incoming audit report, clusters gaps into logical workstreams, assigns metadata and phases, and produces a summary. Phase 2 converts each workstream into a deterministic JSON file with branch names, tasks, git commands and safety considerations. Each atom is single‑purpose and builds on the previous ones.

atom\_241: Audit Analysis: receive\_audit\_json\_and\_validate\_structure | Role: planning\_ai  
atom\_242: Audit Analysis: count\_gaps\_by\_severity | Role: planning\_ai  
atom\_243: Audit Analysis: identify\_foundational\_gaps | Role: planning\_ai  
atom\_244: Audit Analysis: build\_gap\_dependency\_graph | Role: planning\_ai  
atom\_245: Audit Analysis: cluster\_gaps\_by\_category | Role: planning\_ai  
atom\_246: Audit Analysis: cluster\_gaps\_by\_file\_proximity | Role: planning\_ai  
atom\_247: Audit Analysis: cluster\_gaps\_by\_dependency\_chain | Role: planning\_ai  
atom\_248: Audit Analysis: cluster\_gaps\_by\_effort\_and\_split\_large\_clusters | Role: planning\_ai  
atom\_249: Audit Analysis: prioritize\_gaps\_by\_severity\_and\_effort | Role: planning\_ai  
atom\_250: Audit Analysis: assign\_gaps\_to\_workstreams | Role: planning\_ai  
atom\_251: Audit Analysis: generate\_workstream\_ids\_and\_names | Role: docs\_summarizer  
atom\_252: Audit Analysis: compute\_total\_estimated\_hours\_per\_workstream | Role: planning\_ai  
atom\_253: Audit Analysis: determine\_workstream\_earliest\_start\_phase | Role: planning\_ai  
atom\_254: Audit Analysis: identify\_blocking\_workstreams | Role: planning\_ai  
atom\_255: Audit Analysis: assign\_priority\_justification\_and\_risk\_level | Role: planning\_ai  
atom\_256: Audit Analysis: perform\_dependency\_topological\_sort\_of\_workstreams | Role: planning\_ai  
atom\_257: Audit Analysis: assign\_phase\_numbers\_based\_on\_topology | Role: planning\_ai  
atom\_258: Audit Analysis: identify\_parallelizable\_workstreams | Role: planning\_ai  
atom\_259: Audit Analysis: generate\_workstream\_summary\_table | Role: docs\_summarizer  
  
atom\_260: Workstream JSON Generation: normalize\_and\_generate\_branch\_names | Role: repo\_ai  
atom\_261: Workstream JSON Generation: compose\_workstream\_json\_metadata | Role: docs\_summarizer  
atom\_262: Workstream JSON Generation: derive\_high\_level\_objectives | Role: docs\_summarizer  
atom\_263: Workstream JSON Generation: summarize\_code\_changes | Role: docs\_summarizer  
atom\_264: Workstream JSON Generation: group\_gaps\_into\_tasks\_based\_on\_files | Role: planning\_ai  
atom\_265: Workstream JSON Generation: split\_large\_tasks\_and\_enforce\_effort\_limits | Role: planning\_ai  
atom\_266: Workstream JSON Generation: populate\_files\_to\_create\_or\_modify | Role: planning\_ai  
atom\_267: Workstream JSON Generation: derive\_acceptance\_criteria | Role: qa\_test\_agent  
atom\_268: Workstream JSON Generation: develop\_test\_plans | Role: qa\_test\_agent  
atom\_269: Workstream JSON Generation: assemble\_git\_commands | Role: repo\_ai  
atom\_270: Workstream JSON Generation: construct\_commit\_message\_template | Role: docs\_summarizer  
atom\_271: Workstream JSON Generation: embed\_safety\_and\_rollback\_configuration | Role: resilience\_agent  
atom\_272: Workstream JSON Generation: encode\_dependency\_fields\_in\_json | Role: planning\_ai  
atom\_273: Workstream JSON Generation: write\_json\_file\_to\_output\_location | Role: repo\_ai  
atom\_274: Workstream JSON Generation: automate\_execution\_without\_user\_confirmation | Role: orchestrator  
atom\_275: Workstream JSON Generation: maintain\_gap\_traceability\_in\_tasks | Role: planning\_ai

## Micro‑Pipeline Tensor Parallel Fusion

The micro‑pipeline Tensor Parallel (TP) pass in **micro\_pipeline\_tp.py** detects all‑gather and reduce‑scatter patterns in a graph, filters out collectives that can be hidden through simple overlapping and fuses eligible patterns to reduce communication overhead. The atoms below break down this pass into atomic steps handled by a graph\_optimizer role.

atom\_276: Micro Pipeline TP: detect\_all\_gather\_and\_reduce\_scatter\_patterns | Role: graph\_optimizer  
atom\_277: Micro Pipeline TP: identify\_unexposed\_collectives\_for\_overlap | Role: graph\_optimizer  
atom\_278: Micro Pipeline TP: exclude\_unexposed\_collectives\_from\_fusion\_candidates | Role: graph\_optimizer  
atom\_279: Micro Pipeline TP: fuse\_all\_gather\_patterns\_with\_consumer\_matmuls | Role: graph\_optimizer  
atom\_280: Micro Pipeline TP: fuse\_matmul\_patterns\_with\_reduce\_scatter | Role: graph\_optimizer  
atom\_281: Micro Pipeline TP: update\_graph\_topology\_after\_fusion | Role: graph\_optimizer

### Notes on Audit & Micro‑Pipeline Atoms

* **Audit Analysis** atoms parse the audit report, cluster gaps into workstreams based on categories, proximity, dependencies, effort and priority, assign metadata and sequence the streams through topological sorting. The summary table provides a high‑level overview of workstreams including phases and dependencies.
* **Workstream JSON Generation** atoms create deterministic .json files per workstream. They compute branch names, derive objectives and tasks from gaps, craft git commands and commit messages, embed safety and rollback procedures and ensure automatic execution without user prompts. The process enforces naming rules, tasks grouping and adherence to the provided JSON schema.
* **Micro‑Pipeline TP** atoms implement a graph transformation pass: they first detect all‑gather and reduce‑scatter patterns, filter out collectives that are hidden via compute/communication overlap, and then fuse eligible patterns with their adjacent matmuls, updating the graph to use fused operations. This reduces synchronization points and improves parallel efficiency.