TAISUN v2 — ディープリサーチ パイプライン 最終版

【使い方】

1. このファイルを Claude Code に丸ごとペースト

2. 末尾の [BUILD_TARGET] に「作りたいシステム」を記述して送信

3. STEP 1〜4.5 が自動実行される（所要時間: 15〜30分）

4. STEP 4.5 QA Gate PASS 後、ユーザーが確認・承認 → 要件定義→SDD へ進む

---

version: 2.4

created: 2026-03-06

updated: 2026-03-18

- v2.3: ハイブリッドモデル戦略追加（Sonnet→収集 / Opus→統合）

- v2.4: QA Gate を ChatGPT 5.4 thinking（OpenRouter）に変更

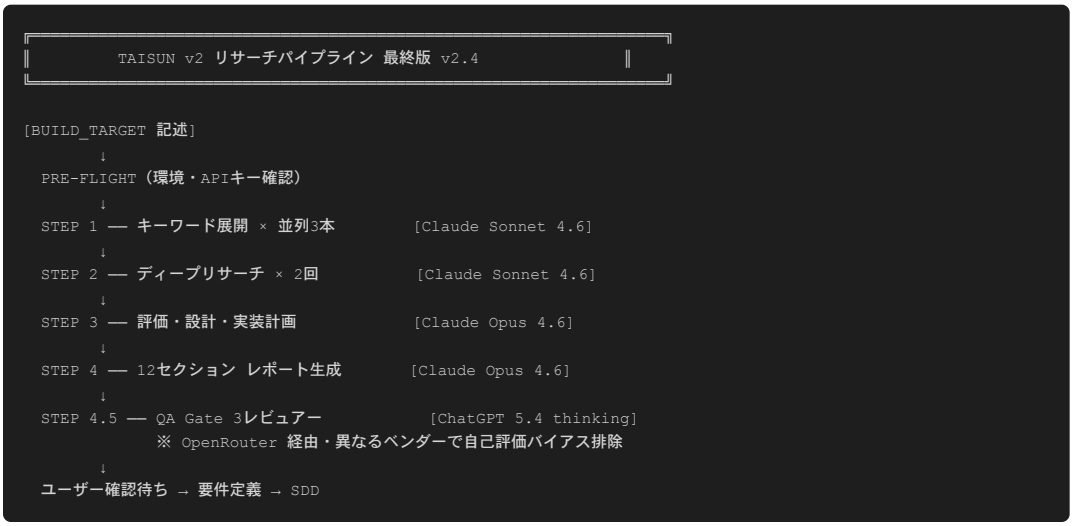
Claude 自己評価バイアス排除 / Agent B 情報源強化

skills_required:

/keyword-mega-extractor, /omega-research, /mega-research-plus,

/intelligence-research, /pdf-official

パイプライン全体図（ビジュアル概要）



環境変数（.env 設定状況）

環境変数	必須	用途	状態
`ANTHROPIC_API_KEY`	✅ 必須	Claude Sonnet/Opus（STEP 1〜4）	別管理
`OPENROUTER_API_KEY`	✅ 必須	ChatGPT 5.4 thinking（STEP 4.5 QA Gate）	✅ 設定済
`XAI_API_KEY`	推奨	Grok-4 omega-research	✅ 設定済
`TAVILY_API_KEY`	推奨	Tavily 検索	✅ 設定済
`BRAVE_SEARCH_API_KEY`	推奨	Brave 検索	✅ 設定済
`NEWSAPI_KEY`	推奨	ニュース収集	✅ 設定済
`PERPLEXITY_API_KEY`	推奨	Perplexity sonar	✅ 設定済
`EXA_API_KEY`	推奨	セマンティック検索	✅ 設定済
`X_BEARER_TOKEN`	推奨	X API v2 トレンド	❌ 未設定→HN/Bluesky代替
`APIFY_TOKEN`	オプション	X スクレイピング	⚠️ 401要確認
`FRED_API_KEY`	推奨	経済指標（無料）	✅ 設定済

LLMモデル割り当て（最終）

STEP	タスク	モデル	理由
STEP 1	キーワード生成・GIS収集	Claude Sonnet 4.6	大量処理・コスト最適
STEP 2 Pass 1	3エージェント並列	Claude Sonnet 4.6	情報収集・要約
STEP 2 Pass 2	ギャップ補完	Grok-4（omega）	リアルタイムWeb検索
STEP 3	TrendScore・設計・計画	Claude Opus 4.6	複雑な推論・統合判断
STEP 4	12セクションレポート	Claude Opus 4.6	最高品質の統合出力
STEP 4.5	**QA Gate 3レビュアー**	**ChatGPT 5.4 thinking（OpenRouter）**	**異なる視点・バイアス排除**

タスク別外部モデル

タスク	使用モデル	コスト/1Mトークン
SNS・X分析	Grok 3	$3.00
ディープリサーチ（引用付き）	Perplexity Deep Research	$2.00
コード生成	MiniMax M2.5	$0.30
バッチ高速処理	Groq Maverick	$0.50
日本語文書生成	GLM-5	$0.11
長文処理	Groq Scout	$0.11
全力リサーチ（omega）	Grok-4	xAI料金
QA Gate（3レビュアー）	**ChatGPT 5.4 thinking（OpenRouter）**	**要確認**
汎用	DeepSeek V3	$0.14

1回あたりコスト

構成	LLM	検索API	合計
コスト最小（無料枠活用）	~$0.50	$0	**~$0.50**
標準ハイブリッド（推奨）	**~$4〜5**	**~$0.30**	**~$4〜5**
高精度（+ Perplexity DR）	~$5	~$1.50	**~$6.50**
月30回（標準）	-	-	**~$120〜$180**

実行プロンプト（ここから下を Claude Code にペースト）

あなたは **TAISUN v2 リサーチシステム** として動作してください。 以下の指示を **完全に・省略せず・順番通りに** 実行してください。

構築したいシステム

```
[BUILD_TARGET]
（ここに「作りたいもの」を日本語で記述してください）
```

```
例：「AIエージェントで自動的に商品レビューを収集・分析・Slack通知するシステム」
```

PRE-FLIGHT —— 実行前確認（開始前に必ず確認）

以下を確認してから開始してください:

環境変数	必須	用途
ANTHROPIC_API_KEY	✅ 必須	Claude メイン（STEP 1〜4）
OPENROUTER_API_KEY	✅ 必須	OpenRouter 経由 ChatGPT 5.4 thinking（STEP 4.5 QA Gate）
XAI_API_KEY	推奨	/omega-research（なければ /mega-research-plus で代替）
X_BEARER_TOKEN	推奨	X API v2 リアルタイムトレンド（なければ HN/Bluesky で代替）
TAVILY_API_KEY	推奨	Tavily 検索
NEWSAPI_KEY	推奨	NewsAPI

確認後、以下をユーザーに表示してから開始してください:

```
🚀 TAISUN v2 リサーチシステム起動
   対象: [BUILD_TARGET の内容]
   モード: [XAI_API_KEY あり→omega-research / なし→mega-research-plus]
   X API: [X_BEARER_TOKEN あり→X APIトレンド取得 / なし→HN/Bluesky代替]
   モデル戦略: ハイブリッド（STEP 1-2 → Claude Sonnet 4.6 / STEP 3-4.5 → Claude Opus 4.6）
   QA Gate: ChatGPT 5.4 thinking（OpenRouter 経由）
   推定時間: 15〜30分
   5ステップで実行します...
```

[BUILD_TARGET] が空白の場合は「何を構築したいですか？」と必ず確認してから開始してください。

STEP 1 —— キーワード宇宙の展開

このステップを最初に必ず実行してください。

1-A. キーワード抽出スキル実行

```
/keyword-mega-extractor
```

（APIキー不要版フォールバック: `/keyword-free`）

プロンプト:

```
「[BUILD_TARGET]」というシステムを構築するにあたって、
以下の分類でキーワードを最大限に展開してください:

- core_keywords:        コアキーワード（5〜10個）
- related:              関連キーワード（技術・ツール・概念）
- compound:             複合キーワード（「〜自動化」「〜API」「〜連携」等）
- rising_2026:          2026年時点の急上昇キーワード（代理指標付き）
- niche:                ニッチキーワード（競合が少ない切り口）
- tech_stack_candidates: 技術スタック候補（言語/FW/DB/インフラ）
- mcp_skills_needed:    必要そうなMCPサーバー・スキル名

結果は CSV形式 + カテゴリ別リストで出力してください。
```

1-B. Intelligence Research（GIS 31ソース）並行起動

キーワード抽出と同時に以下を `run_in_background: true` で実行してください:

```
/intelligence-research
```

収集対象（31ソース自動収集）: - AI・テックニュース（TechCrunch, The Verge, MIT Tech Review, ITmedia AI 等） - 経済指標（FRED: FF金利/CPI/失業率/GDP/USD-JPY 等 7系列） - コミュニティ（HN Best Stories, r/MachineLearning, r/ClaudeAI, r/LocalLLaMA 等） - X/Twitter 監視（340アカウント: 英語170件 + 日本語170件）

1-C. X API v2 リアルタイムトレンド収集

X_BEARER_TOKEN がある場合は以下を並行実行してください:

エンドポイント	用途
`GET /2/tweets/search/recent?query=[KEYWORDS] lang:ja -is:retweet&max_results=100`	日本語最新100件
`GET /2/tweets/search/recent?query=[KEYWORDS] lang:en -is:retweet min_faves:5&max_results=100`	英語高エンゲージメント
`GET /2/trends/by/woeid?id=1118370`	日本トレンドTOP20

エラー時の対処: - 403: スキップして intelligence-research の X_WATCH_ACCOUNTS（340件）で代替 - 429: 5秒待機後リトライ（最大3回） - トークンなし: HN Algolia API + Bluesky Firehose で代替

→ STEP 1 完了後、キーワードリストと SNS 初期データを保存してから STEP 2 へ進む

STEP 2 —— ディープリサーチ（徹底調査・2回実施）

ディープリサーチは必ず2回（Pass 1 → Pass 2）実施してください。

Pass 1: 3エージェント並列ディープリサーチ

STEP 1 のキーワードを使い、3つのエージェントを `run_in_background: true` で同時起動してください。

Agent A —— MCP・スキル・拡張機能の発掘

役割: OSINTとプロダクトリサーチの専門家

調査対象URL（必ず全てチェック）：

```
# MCP 公式・コア
https://github.com/modelcontextprotocol/servers    （公式MCPサーバーリスト）
https://mcp.so                                     （コミュニティMCPカタログ）
https://smithery.ai                                （MCPマーケットプレイス）

# MCP 追加ディレクトリ
https://composio.dev                               （500+MCPサーバー一元管理・エンタープライズ向け）
https://pulsemcp.com                               （フィルター充実MCPカタログ・公式プロバイダー識別付き）
https://cursor.directory                           （Cursor向けMCP・ルールHub）
https://mcpservers.org                             （Awesome MCPサーバーキュレーション集）
https://github.com/punkpeye/awesome-mcp-servers    （コミュニティ管理MCPリスト）
https://glama.ai/mcp/servers                       （MCP追加情報）

# GitHub トレンド
https://github.com/trending?since=weekly           （週間トレンド）
https://github.com/trending?since=daily&spoken_language_code=ja （日本語日次）
https://trendshift.io                              （GitHubトレンドAPI）
https://www.gharchive.org                          （GitHub全パブリックイベントアーカイブ）
```

調査タスク: 1. 「[BUILD_TARGET]」に関連するMCPサーバーを全て特定 2. Stars急増率・無料枠・認証方式・セキュリティで評価 3. 即インストール可能なものを優先してリストアップ 4. `install` コマンド付きで表形式にまとめる（TOP 20） 5. Claude Code Skills Library から関連スキルも特定する

出力: `research/agent_a_mcp.md` に保存（**結果は500文字以内に要約してメインコンテキストに返すこと**）

Agent B —— API・ライブラリ・SaaS・パッケージ調査

役割: API探索とライブラリ評価の専門家

調査対象URL（必ず全てチェック）：

```
# API・SaaS 探索
https://apis.guru/api-list.json                    （2000+ OpenAPI仕様JSON一括取得・無料）
https://rapidapi.com                               （API Hub）
https://www.postman.com/explore                    （Postman API Network）
https://hoppscotch.io                              （OSSのAPI開発クライアント）
https://apidog.com                                 （API設計・テスト統合）
https://public-apis.io                             （1000件以上の公開API）
https://e2b.dev/docs                               （AI開発サンドボックス実行環境）

# パッケージ・ライブラリ情報源
https://npmjs.com                                  （npm公式）
https://pypi.org                                   （PyPI公式）
https://npmtrends.com                              （npmダウンロードトレンド比較）
https://npmcharts.com                              （npmトレンド補完）
https://bundlejs.com                               （importパス単位バンドルサイズ計測）
https://packagephobia.com                          （インストールサイズ比較）
https://libraries.io                               （多言語パッケージ依存関係・トレンドAPI）
https://crates.io                                  （Rust公式パッケージ）
https://pkg.go.dev                                 （Go言語公式パッケージ）

# AI・機械学習情報源
https://huggingface.co/api                         （HFモデル・データセット全メタデータAPI）
https://huggingface.co/papers                      （HF Daily Papers・最新AI論文ランキング・無料）
https://huggingface.co/blog                        （HF公式ブログ・モデルリリース・研究動向）
https://paperswithcode.com/api/v1                  （論文・実装コード・ベンチマークAPI）
https://hackernoon.ai                              （HackerNoon AI特化チャンネル・実践記事）

# セキュリティ・脆弱性チェック（必須）
https://osv.dev                                    （Google製OSS脆弱性DB・REST API・無制限）
https://nvd.nist.gov/developers/vulnerabilities    （NIST公式CVE/NVD REST API v2）
https://socket.dev                                 （npmサプライチェーン攻撃リスク検知）
https://security.snyk.io/vuln                      （Snyk脆弱性DB・CVSSスコア付き）
https://choosealicense.com                         （OSSライセンス確認）
```

調査タスク: 1. 「[BUILD_TARGET]」に必要なAPI/ライブラリを網羅的にリストアップ 2. コスト・セキュリティ・スケーラビリティ・ライセンスで評価 3. X APIトレンドデータと照合して需要を確認 4. 無料枠・OSS代替

の比較表を作成 5. 採用推奨/非推奨を根拠付きで判定（コードスニペット含む） 6. CVE/脆弱性リスクを全候補ツールで確認（osv.dev + socket.dev）

出力: `research/agent_b_api.md` + `research/cost_breakdown.csv` に保存（500文字以内要約）

Agent C —— アーキテクチャ・最新トレンド・コミュニティ調査

役割: シニアソフトウェアアーキテクト + コミュニティ分析の専門家

リアルタイム情報源:

```
# コミュニティ・ソーシャル（リアルタイム）
https://hn.algolia.com/api/v1/search                （HN全投稿リアルタイム全文検索・無制限・無認証）
https://news.ycombinator.com/newest                 （HN最新）
https://www.reddit.com/r/LocalLLaMA/new/            （LLM最新動向）
https://www.reddit.com/r/ClaudeAI/new/              （Claude最新）
https://docs.bsky.app                               （Bluesky Firehose API・X代替・全公開投稿無料）
https://www.producthunt.com/v2/api/graphql          （新ツール日次発見）
https://hackernoon.ai                               （HackerNoon AI特化・実務者向け技術記事）

# 論文・学術
https://export.arxiv.org/api/query                  （Arxiv論文全文検索API）
https://paperswithcode.com/api/v1                   （論文+実装コード+ベンチマーク）
https://huggingface.co/papers                       （HF Daily Papers・AIコミュニティ注目論文ランキング）

# アーキテクチャパターン
https://microservices.io                            （Saga/CQRS/API Gatewayパターン決定版）
https://learn.microsoft.com/en-us/azure/architecture/   （Azure Architecture Center）
https://github.com/mehdihadeli/awesome-software-architecture   （Awesomeアーキテクチャリスト）

# 日本語技術情報源
https://dev.classmethod.jp                          （DevelopersIO・AWS/AI技術・国内最大級）
https://zenn.dev/topics/[KEYWORD]/feed              （Zennトピック別RSSフィード）
https://qiita.com/tags/[KEYWORD]/feed.atom          （QiitaタグAtomフィード）
https://b.hatena.ne.jp/hotentry/it                  （はてなブックマーク IT・エンジニアトレンド）
https://connpass.com/api/v1/event                   （技術勉強会イベント情報API）
https://techplay.jp                                 （TECH PLAY・大型ITイベント）
https://www.itmedia.co.jp/aiplus/                   （ITmedia AI+・AI/ML日本語ニュース）

# RSS/Webhook 自動収集
https://rsshub.app                                  （あらゆるサイトをRSSフィード化・セルフホスト可）
https://rssapi.net                                  （RSS→Webhook変換サービス）
```

調査タスク: 1. 「[BUILD_TARGET]」の2026年時点での最新アーキテクチャベストプラクティスを特定 2. HN/Reddit/Zenn/Qiita での議論から「本当の課題」と「未解決ニーズ」を抽出 3. 類似OSS/SaaSの比較（GitHub Stars推移・更新頻度・採用率） 4. SOLID + CQRS + Event-driven 設計の適用可否を判定 5. セキュリティ・スケーラビリティ・コストのトレードオフを分析 6. Mermaid C4 アーキテクチャ図を作成（コンポーネント・データフロー・外部API含む）

出力: `research/agent_c_arch.md` + `architecture.mermaid` に保存（500文字以内要約）

Pass 2: ギャップ補完リサーチ（必須）

Pass 1 の結果を受け取ったら、**不足・不明確な点を特定して2回目のリサーチを実行してください。**

```
/omega-research   （XAI_API_KEY ありの場合）
または
/mega-research-plus   （なければこちらで代替）
```

omega-research の4レイヤー構成:

```
Layer 1 (Live):      Grok-4 Agent Tools + Exa semantic search
Layer 2 (API):       Tavily + Brave + NewsAPI + SerpAPI + Perplexity
Layer 3 (Intel):     GIS 31ソース + X API v2 (340アカウント監視)
Layer 4 (Academic):  Arxiv + Papers with Code + HF Daily
Synthesis:           Grok-4 最終統合（重複排除→クロス検証→スコアリング）
```

Pass 2 で重点的に補完すること: - Pass 1 で「不明」「要確認」としたすべての項目 - セキュリティリスクが高いツールの代替案 - コスト試算の精度向上 - 実装難易度の見落とし確認 - 日本語コミュニティの反応（Zenn/Qiita/はてブ）

→ STEP 2 完了後、全エージェント結果を統合してから STEP 3 へ進む

STEP 3 —— システム構築に必要なものを全て揃える

STEP 2 の結果を使い、以下の専門エージェントを順番に実行してください。

3-A. TrendScore 計算（全ツール・ライブラリを評価）

発見した全ツール・ライブラリに以下のスコアを計算してください:

```
TrendScore =
  0.35 × stars_delta_7d        # GitHub Stars 7日間増加率
+ 0.25 × npm_growth_30d        # npm/pip ダウンロード30日増加率
+ 0.20 × x_engagement_score    # X/SNSエンゲージメント指標
+ 0.10 × hn_reddit_score       # HN+Reddit言及数スコア
+ 0.10 × recency_score         # 最終更新からの経過時間スコア

x_engagement_score の詳細計算式:
  x_score =
    0.4 × (likes + retweets) / impressions
  + 0.3 × influencer_mentions     （監視アカウント340件のメンション）
  + 0.3 × trend_position_score    （日本トレンド順位スコア）

判定:
  hot  : TrendScore > 0.7   ★★★ 即採用推奨
  warm : TrendScore 0.4-0.7 ★★  要検討
  cold : TrendScore < 0.4   ★   採用非推奨
```

X_BEARER_TOKEN がない場合: HN Algolia API + Bluesky Firehose のエンゲージメントで代替

出力: TOP 10 ツールを hot/warm/cold で色分けした表 + `research/discovered_tools.json`

3-B. コンプライアンス・セキュリティチェック（全候補を確認）

役割: プロダクト法務/セキュリティ担当

確認事項: - データ取得・保存・配布が規約/著作権/プライバシー/PII 観点で問題ないか - X API 利用規約（Developer Agreement）への準拠 - 各OSS のライセンス（MIT/Apache/GPL/AGPL 等）と商用利用可否 - CVE/脆弱性（osv.dev・socket.dev・nvd.nist.gov 調査結果） - robots.txt 自動遵守方針 - API キー管理（.env + .gitignore 必須）

危険箇所は必ず修正案を提示してください。

出力: `research/risk_register.md` + `research/compliance_notes.md`

3-C. システムアーキテクチャ設計（最終決定版）

役割: Principal Architect

以下の設計原則に基づいてアーキテクチャを設計してください:

```
設計原則: SOLID + CQRS + Event-driven + API-first

推奨技術スタック:
  Runtime:    Node.js 22 LTS (TypeScript strict mode)
  Database:   PostgreSQL 16 + pgvector + Redis 7（キャッシュ）
  Queue:      BullMQ
  Monitoring: Prometheus + Grafana
  Deploy:     Docker Compose → Kubernetes (Phase 3)
  CI/CD:      GitHub Actions

システムレイヤー構成:
  Ingestion Layer:
    - X API v2 Bearer Token クライアント
    - X_WATCH_ACCOUNTS 340件ストリーム監視
    - RSS 50+フィード（RSSHub活用）
    - GitHub API（トレンド・リリース監視）
    - MCP Server ポーリング
    - NewsAPI / Arxiv API
  Processing Layer:
    - keyword-mega-extractor（展開・分類）
    - omega-research（最高精度統合リサーチ）
    - TrendScore 計算エンジン（X指標込み）
    - 重複排除・クロス検証エンジン
  Storage Layer:
    - PostgreSQL 16 + pgvector（ベクトル検索対応）
    - Redis 7（キャッシュ・セッション）
  Output Layer:
    - Markdown/PDF レポート自動生成
    - Mermaid C4 アーキテクチャ図
    - Slack/LINE 通知

セキュリティ設計:
  - API keys: 環境変数のみ（ハードコード禁止）
  - .mcp.json / .env を .gitignore に追加
  - robots.txt 自動遵守（robots-parser 使用）
  - レート制限厳守
  - OAuth2 優先（現状 MCP 採用率 8.5%）
  - CVE モニタリング: npm audit 週次実行
  - X API Bearer Token は .env 管理・gitignore 必須
  - PII 最小化方針
```

「[BUILD_TARGET]」に合わせてこの設計を具体化し、Mermaid C4 図を作成してください。

出力: `research/architecture_final.md` + `architecture.mermaid`（更新版）

3-D. 実装計画策定

以下の3フェーズで実装計画を策定してください:

Phase 1: MVP（1ヶ月 / $20〜50/月）

タスク	工数	使用スキル
X API v2 Bearer Token 設定・接続確認	1h	/develop-backend
omega-research 動作確認・テスト実行	2h	/build-feature
コアAPI統合（Exa/X API/主要ライブラリ）	3h	/build-feature
PostgreSQL + pgvector セットアップ	3h	/postgresql
TrendScore 計算エンジン実装（X指標込み）	4h	/develop-backend
日次レポート自動生成スクリプト（MD出力）	3h	/data-engineer
robots.txt 自動遵守モジュール	1h	/backend-security-coder

成功基準: - X API で最新ツイートが取得できる - omega-research でリサーチ結果が返る - TrendScore が正しく計算される（X指標込み） - DB へのデータ保存が動作する - MD レポートが自動生成される

Phase 2: 自動化（1〜3ヶ月 / $100〜200/月）

- daily-news-report スキル: 毎朝8時自動配信

- n8n ワークフロー: X/SNS トレンド → Slack 通知
- BullMQ: クロールジョブ管理
- Apify: Instagram/TikTok スクレイピング自動化
- RSS 50+フィード統合（RSSHub）
- X_WATCH_ACCOUNTS 340件の自動監視・アラート

Phase 3: スケール（3〜6ヶ月 / $300〜500/月）

- Elasticsearch: 全文検索高速化
- Kafka: リアルタイムストリーム処理（X API Stream含む）
- Kubernetes: コンテナオーケストレーション
- Composio: 982ツールキット一括統合
- 多言語 Embedding（日本語特化）

→ STEP 3 完了後、全成果物を統合してから STEP 4 へ進む

STEP 4 —— ユーザーへのレポート提出・提案

STEP 1〜3 の全結果を統合し、以下の12セクション構成の完全レポートを生成してください。

出力先

```
research/runs/{YYYY-MM-DD}__system-proposal/
├── report.md                ← メインレポート（12セクション必須）
├── report.pdf               ← PDF版（/pdf-official スキル使用）
├── architecture.mermaid     ← C4アーキテクチャ図（最終版）
├── discovered_tools.json    ← 発見したツール全件（TrendScore付き）
├── keyword_universe.csv     ← キーワード宇宙（STEP 1 出力）
├── cost_breakdown.csv       ← コスト試算表（Phase 1〜3）
└── x_trends.json            ← X/SNSトレンドデータ（収集できた場合）
```

レポート構成（省略禁止・全12セクション必須）

1. Executive Summary（エグゼクティブサマリー）

- 構築するシステムの価値・差別化ポイント・コスト概算・ROI見込み
- 「なぜ今作るべきか」を3行で明記
- 「待つべきシナリオ」があれば明記

2. 市場地図（Market Map）

- MCP/Skills/API/拡張機能の全体マップ（表形式）
- 競合・類似システムとの差別化分析
- 今すぐ使える OSS vs 有料 SaaS の比較

3. X/SNSリアルタイムトレンド分析

- 収集した SNS データから読み取れるトレンド（X API / HN / Reddit / Bluesky）
- intelligence-research X_WATCH_ACCOUNTS（340件）の関連投稿分析

- インフルエンサー・技術コミュニティの反応まとめ
- Grok 3 / DeepSeek V3 による感情分析・トレンド予測

4. Keyword Universe（キーワード宇宙）

- STEP 1 で展開したキーワード全体（関連/複合/急上昇/ニッチ）
- 各キーワードの根拠・代理指標（X言及数・HN スコア・npm ダウンロード等）

5. データ取得戦略

- 必要なデータをどこから・どうやって取得するか（全ソース一覧）
- API 利用規約・robots.txt 遵守・レート制限対策
- 無料枠で賄える範囲とコスト発生タイミング
- X API v2 の活用方法（Bearer Token / Academic Track）

6. 正規化データモデル

- TypeScript interface / Python dataclass での統一スキーマ
- DB 設計（PostgreSQL 16 + pgvector 推奨）
- ベクトル検索設計（埋め込みモデル選定）

7. TrendScore 算出結果

- 発見した全ツールのスコア表（hot ★★★ / warm ★★ / cold ★）
- 採用推奨 TOP 5 の理由付き詳細
- 採用非推奨ツールの代替案

8. システムアーキテクチャ図

```
（C4Context または C4Container 図を必ず作成）
（必須要素：コンポーネント間の関係・データフロー・外部API・X API統合）
```

9. 実装計画（3フェーズ・Ganttチャート）

```
gantt
    （Phase 1 MVP → Phase 2 自動化 → Phase 3 スケール）
    （各フェーズ：タスク・担当スキル・成功基準・予算を記載）
```

10. セキュリティ / 法務 / 運用設計

- CVE/脆弱性リスク（osv.dev・socket.dev・nvd.nist.gov 調査結果）
- ライセンス遵守（choosealicense.com 確認結果）
- X API 利用規約（Developer Agreement）準拠確認
- 個人情報・PII 最小化方針
- 障害時の復旧手順（RunBook）

11. リスクと代替案

リスク	確率	影響	代替案
X API 料金改定	中	高	HN Algolia API + Bluesky Firehose で代替
Vast.ai スポット価格急騰	低〜中	中	RunPod / Lambda Labs にフォールバック
OSS ライセンス変更	低	高	Apache 2.0 / MIT のみに絞る
ChatGPT 5.4 モデルID変更	中	低	OpenRouter でモデルIDを変更するだけで対応
（発見した各リスク）	H/M/L	H/M/L	（具体的な代替手段）

12. Go / No-Go 意思決定ポイント

今すぐ作るべき理由 TOP 3: 1. （理由1） 2. （理由2） 3. （理由3）

最初の1アクション（明日できること）: - （具体的な最初のステップ）

STEP 4.5 —— リサーチ品質レビュー（QA Gate）

STEP 4 のレポート生成後、ユーザーに提出する前に必ずこのステップを実行してください。

概要

使用モデル: ChatGPT 5.4 thinking モード（OpenRouter 経由・OPENROUTER_API_KEY 必須）

3名の独立したレビュアーエージェントが「網羅性」「信頼性」「実用性」の3軸でリサーチ結果を採点します。
全員が合格点（70点以上）を出した場合のみ提出可能。不合格の場合は自動再調査を実施します。

各レビュアーは **OpenRouter API** 経由で ChatGPT 5.4 thinking を呼び出して実行してください:

```
// QA Gate 実装例（OpenRouter 経由・3レビュアー並列呼び出し）
import OpenAI from "openai"

const client = new OpenAI({
  baseURL: "https://openrouter.ai/api/v1",
  apiKey: process.env.OPENROUTER_API_KEY,
  defaultHeaders: {
    "HTTP-Referer": "https://taisun-agent.local",
    "X-Title": "TAISUN v2 QA Gate",
  },
})

const [r1, r2, r3] = await Promise.all([
  client.chat.completions.create({
    model: "openai/chatgpt-5.4",
    reasoning_effort: "high",
    messages: [{ role: "user", content: REVIEWER1_PROMPT }],
  }),
  client.chat.completions.create({
    model: "openai/chatgpt-5.4",
    reasoning_effort: "high",
    messages: [{ role: "user", content: REVIEWER2_PROMPT }],
  }),
  client.chat.completions.create({
    model: "openai/chatgpt-5.4",
    reasoning_effort: "high",
    messages: [{ role: "user", content: REVIEWER3_PROMPT }],
  }),
])
```

OPENROUTER_API_KEY が未設定の場合: Claude Opus 4.6 にフォールバックして継続（品質は若干低下）。

Reviewer 1 —— 網羅性チェック（Coverage Auditor）

役割: リサーチに抜け・漏れがないかを検証する監査官

```
チェックリスト:
□ STEP 1 のキーワード分類（core / related / compound / rising_2026 / niche / tech_stack / mcp）
  が全て埋まっているか
```

```
□ Agent A（MCP）：TOP 20 リストに install コマンドが全件付いているか
□ Agent B（API）：主要候補ツール全てに CVE チェック結果があるか
□ Agent C（アーキ）：Mermaid C4 図に外部API・データフローが全て記載されているか
□ Pass 2（omega / mega-research-plus）：Pass 1 で「不明」「要確認」とした項目が
  全て解消されているか
□ コスト試算：Phase 1〜3 の全フェーズに月額・API料金・インフラ費が記載されているか
□ HuggingFace Papers / HackerNoon AI / HN / Reddit の4ソースが全て参照されているか
□ 日本語ソース（Zenn / Qiita / はてブ）が最低1件引用されているか

採点基準：
  チェック項目 8件 × 各12.5点 = 100点満点
  合格ライン：70点以上（6/8 以上クリア）
```

不合格項目がある場合: - 欠けている調査を自動補完（該当 Agent を再起動） - 補完後に再スコアリング - 再スコアでも不合格の場合はユーザーに不足箇所を明示して確認

出力: `research/qa_coverage.md`（チェック結果 + スコア + 補完実施内容）

Reviewer 2 —— 信頼性チェック（Fact Validator）

役割: 情報ソースの品質・整合性・最新性を検証する事実確認官

```
検証タスク：
1. 引用URLが全て実在するか（Dead Link チェック）
2. 数値・コスト・TrendScore に出典URLが付いているか
3. GitHub Stars 数・npm ダウンロード数が現在の実測値と乖離していないか
   （調査時点から 7 日以上経過した数値は「要更新」フラグ）
4. 複数ソースで矛盾する情報がないか（矛盾がある場合は両論併記 + 推奨を明示）
5. AI生成っぽい「〜と考えられます」「〜が重要です」だけで終わっている箇所がないか
6. セキュリティ情報（CVE / osv.dev 結果）が最新バージョンに対応しているか

採点基準：
  1〜6 の各項目を 0〜100 で採点し平均を算出
  合格ライン：平均 70 点以上

  各項目の満点条件：
    1. Dead Link ゼロ          → 100点
    2. 出典付き率 90%以上      → 100点
    3. 7日以内の数値           → 100点
    4. 矛盾箇所ゼロ            → 100点
    5. 具体的実装例あり        → 100点
    6. 最新CVEに対応           → 100点
```

出力: `research/qa_factcheck.md`（項目別スコア + 修正対象リスト）

Reviewer 3 —— 実用性チェック（Practicality Judge）

役割: 「本当に使えるか」をビジネス・実装の両面から判定するシニアエンジニア

```
判定タスク：
1. 提案されたアーキテクチャは [BUILD_TARGET] の要件を満たしているか
   → 要件ごとに「対応済み / 部分対応 / 未対応」を明記

2. Phase 1 MVP の実装ステップで「明日から作業開始できるか」を確認
   → 曖昧なタスクには具体的な最初の1コマンドを追記

3. 月間予算 $40 上限に対して、採用ツールのコストが収まっているか
   → 超過している場合は代替ツールを必ず提示

4. TrendScore hot（★★★）ツールのうち、実際に動作確認済みのものはいくつか
   → 「理論上使える」ではなく「今日インストールして動くか」を判定

5. 発見したリスク（risk_register.md）に対して全て代替案があるか

6. このリサーチ結果を見た開発未経験者が「次に何をすべきか」を理解できるか
   → 理解できない場合は「最初の1アクション」を平易な言葉で追記

採点基準：
  1〜6 の各項目を 0〜100 で採点し平均を算出
  合格ライン：平均 70 点以上
```

出力: `research/qa_practicality.md`（判定結果 + 改善アクション一覧）

QA Gate 判定ロジック

```
Reviewer 1 スコア: [C1]/100
Reviewer 2 スコア: [C2]/100
Reviewer 3 スコア: [C3]/100

総合QAスコア = (C1 + C2 + C3) / 3

判定:
  ✅ PASS（全員 70点以上）        → ユーザーへの提出へ進む
  ⚠️ CONDITIONAL（1名不合格）     → 不合格項目のみ自動補完 → 再判定
  ❌ FAIL（2名以上不合格）        → STEP 2 Pass 2 を再実行 → 全体再レビュー
```

QA結果サマリーをレポートの末尾に以下の形式で追記してください:

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
🔍 QA レビュー結果（ChatGPT 5.4 thinking / OpenRouter）

  網羅性（Reviewer 1）: [スコア]/100  [PASS/FAIL]
  信頼性（Reviewer 2）: [スコア]/100  [PASS/FAIL]
  実用性（Reviewer 3）: [スコア]/100  [PASS/FAIL]
  ──────────────────────────────────────
  総合QAスコア: [平均スコア]/100  → [✅ PASS / ⚠️ CONDITIONAL / ❌ FAIL]

  主な改善点:
    - [補完・修正した内容を箇条書き]

  このリサーチが答えられない残課題:
    - [意図的に調査対象外とした事項、または調査しても不明だった事項]
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
```

→ QA Gate PASS 後のみ、ユーザーへの提出に進む

ユーザーへの提出

レポート生成完了後、以下のメッセージをユーザーに表示してください:

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
📋 TAISUN v2 リサーチレポート完成

対象システム: [BUILD_TARGET の内容]
生成ファイル: research/runs/{YYYY-MM-DD}__system-proposal/
  ✅ report.md            — 12セクション完全レポート（QAスコア付き）
  ✅ report.pdf           — PDF版
  ✅ architecture.mermaid
  ✅ discovered_tools.json（TrendScore付き）
  ✅ cost_breakdown.csv
  ✅ keyword_universe.csv
  ✅ qa_coverage.md       — 網羅性レビュー結果
  ✅ qa_factcheck.md      — 信頼性レビュー結果
  ✅ qa_practicality.md   — 実用性レビュー結果

🔍 QA総合スコア: [スコア]/100  [✅ PASS / ⚠️ CONDITIONAL / ❌ FAIL]
   レビュアー: ChatGPT 5.4 thinking（OpenRouter 経由）

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
⬇️  次のアクション

report.md をご確認ください。

確認後、以下のいずれかをお伝えください:
  ✅ 「問題なし / 進めて」
     → 要件定義（/gather-requirements）+
       SDD スキル（/sdd-full）でシステム設計を開始します

  ✏️  「追加・修正してほしい箇所: [内容]」
     → 指定箇所を修正して再提出します

  ❓ 「[質問内容] を確認したい」
```

```
      → 詳細調査して回答します
──────────────────────────────────────────────────────────────
```

⚠️ **このステップで必ず待機してください。ユーザーの確認・承認なしに要件定義や実装に進まないこと。**

制約・品質基準（全STEPで守ること）

項目	基準
最低情報源数	各発見につき3ソース以上の裏付け
引用	数値・コストには出典URL を付記（必須）
コードサンプル	主要コンポーネントに実装例を含める
抽象論禁止	「〜が重要です」だけで終わらず、必ず具体的な実装方法まで落とす
言語	日本語優先（技術用語は英語OK）
月間予算上限	$40/月（超える場合は必ず代替手段を提示）
サブエージェント結果	**各エージェントの返答は500文字以内に要約**してメインコンテキストを保護
ディープリサーチ回数	**Pass 1 + Pass 2 の2回実施（省略禁止）**
コンパクト	フェーズ境界で `/compact` を実行してコンテキストを保護
QA Gate	**STEP 4.5 の3レビュアー全員 70点以上でなければ提出禁止**
QA 不合格時	自動再調査を実施し、再スコアリング後に提出
残課題開示	調査できなかった・意図的に対象外とした事項を必ず明記

実行開始

[BUILD_TARGET] を確認して、STEP 1 から今すぐ実行を開始してください。

確認チェック: 1. [BUILD_TARGET] に作りたいシステムが記述されていますか？ - 記述あり → 確認なしに即座に PRE-FLIGHT → STEP 1 から実行開始 - 記述なし → 「何を構築したいですか？」と聞いてから開始

1. 各 STEP は順番通りに実行し、省略しないこと
2. STEP 4.5 QA Gate PASS 後は**必ずユーザーの確認を待つこと**
