CLAUDE CODE PRACTICAL GUIDE

Cloud Sessions + Model Routing

إزاي تشغّل أكتر من Session في نفس الوقت، تتابع كل Task، وتستخدم الموديل المناسب للشغل المناسب بدل ما تشغّل أقوى Model على كل حاجة.

Cloud Sessions | Agent View | model/ | opusplan | Usage Optimization

الفكرة في سطر: خليك بتدير Claude Code كأنه فريق شغال Parallel - لكن من غير ما تخلط بين السرعة وبين توفير الـUsage.

إعداد: Ahmed Alaydi - نسخة مبسطة وعملية مبنية على وثائق Anthropic الرسمية.

الـPDF ده معمول ليه؟

لو إنت بتستخدم Claude Code وبتشتغل Session واحدة لحد ما تخلص، فيه Workflow أقوى: تشغّل Tasks مستقلة في Cloud Sessions، تتابعها من أي جهاز، وتختار Model مناسب حسب صعوبة المهمة.

مش مناسب لإيه؟

- Tasks بتعدّل نفس الملفات في نفس الوقت
- شغل محتاج تدخل منك كل دقيقة
- Requirements لسه مش واضحة
- تشغيل Parallel لمجرد إنه "أسرع" بدون تقسيم حقيقي

ينفع مع إيه؟

- Bug fixes مستقلة
- كتابة Tests
- Refactors منفصلة
- Features واضحة المتطلبات
- Code review أو Research

مهم: تشغيل Sessions كتير بالتوازي **مش بيوفّر Tokens تلقائيًا**. بالعكس، كل Session بتستهلك Usage. التوفير الحقيقي بييجي من تقسيم الشغل صح واختيار Model أخف للمهام البسيطة.

1

قبل ما تبدأ Cloud Sessions

المطلوب

- Claude Code شغال عندك أو Claude Desktop.
- GitHub repo متوصل بحساب Claude لو هتستخدم Anthropic-hosted Cloud Sessions.
- خطة تدعم Cloud Sessions على حسابك.
- Task مكتوبة بوضوح: الهدف، الملفات أو الـscope، شروط النجاح، وما الذي لا يجب تغييره.

لو شغال من Terminal

```
claude
```

ولو أول مرة تجهز GitHub للـCloud من الـCLI، وثائق Claude Code بتوفر مسار إعداد للويب/GitHub حسب حسابك.

أفضل بداية: قبل Task كبيرة استخدم Plan Mode محليًا، راجع الخطة، وبعدها ابعت التنفيذ للـCloud.

```
claude --permission-mode plan
```

2

شغّل أول Cloud Session

من Claude Desktop

1. افتح **Code**.
2. ابدأ Session جديدة.
3. من اختيار الـEnvironment اختار **Cloud** بدل Local.
4. اختار الـRepo.
5. اكتب Task واضحة وابدأ.

من Terminal

```
claude --cloud "Fix the login redirect bug. Add regression tests. Do not change unrelated auth flows."
```

الـSession بتشتغل في Cloud Environment منفصل. بعد ما تبدأ، تقدر تقفل اللابتوب وترجع تتابع من claude.ai/code أو من تطبيق Claude.

> **الناتج المعتاد:** Claude يقرأ الـrepo، يعمل التعديلات، يشغّل tests/commands، وبعدها يقدر يدفع التغييرات على Branch علشان تراجعها.

3

شغّل Tasks بالتوازي

كل أمر `--cloud` جديد بيعمل Cloud Session مستقلة. استخدم ده فقط لما الـTasks ما تكونش متداخلة بقوة.

```
claude --cloud "Fix checkout validation bug and add tests."

claude --cloud "Add unit tests for notification service. Do not modify checkout files."

claude --cloud "Audit API rate limiting and propose a safe patch. Do not deploy."
```

تقسيم خطر

Session تعدّل auth service وSession ثانية تعمل refactor لنفس auth service في نفس الوقت.

تقسيم كويس

Bug في checkout + tests للإشعارات + audit للـrate limit. حدود مختلفة وواضحة.

قاعدة مهمة: Parallelism مفيد لما الشغل قابل للتقسيم. لو Sessions بتلمس نفس الملفات أو نفس invariant، غالبًا هتخلق conflicts ومراجعة أصعب.

4

تابع الـSessions

Cloud Sessions

هتلاقي الـSessions في sidebar على **claude.ai/code**. تقدر تفتح أي Session، تراجع الـChanges، ترد على سؤال، أو توجه Claude وهو شغال.

إيه اللي تراقبه؟

- هل الـSession لسه Working ولا مستنية Input؟
- إيه آخر تغيير عملته؟
- هل الـTests نجحت؟
- هل الـdiff داخل الـscope المطلوب؟

Agent View في الـCLI

```
claude agents
```

أو اضغط السهم الشمال من Session داخل CLI. Agent View بيدي لك شاشة واحدة تتابع منها Agents/Sessions في Claude Code CLI: مين شغال، مين مستنيك، ومين خلص.

ملاحظة: Cloud Sessions ليها واجهة متابعة على claude.ai/code، وAgent View هو واجهة إدارة للـCLI sessions. ما تعتبرهمش نفس الشاشة.

5

اختار الـModel حسب الـTask

داخل Claude Code استخدم:

```
/model
```

علشان تشوف الموديلات المتاحة لحسابك وتغيّر الـModel أثناء نفس الـSession. استخدم `/status` لو عايز تعرف الـModel الحالي.

نوع المهمة	اختيار عملي	ليه؟
Architecture / Debugging صعب / Refactor كبير	**Opus**	Reasoning أعمق للحالات المعقدة.
Features / Tests / Refactors معتادة	**Sonnet**	أفضل Default لمعظم شغل البرمجة من حيث السرعة والكفاءة.
Rename / Lookup / Boilerplate / تعديلات ميكانيكية	**Haiku**	أسرع وأخف للمهام البسيطة.

> **لا تحفظ أسماء Versions في دماغك.** Anthropic بتغير أسماء وتوفر الموديلات مع الوقت. اعتبر `/model` هو Source of Truth لحسابك.

6

Opus يخطط.. وSonnet ينفذ

لو عندك Task معقدة لكن التنفيذ بعد التخطيط واضح، استخدم النمط المدمج:

```
/model opusplan
```

الفكرة: تستخدم Opus في الـPlanning، وبعدها Sonnet يكمل التنفيذ. ده مثال ممتاز على إنك تحط الـReasoning الغالي في المكان اللي فعلاً محتاجه.

مثال Task مناسبة:

"Refactor permissions across admin and teacher modules without changing behavior. First map the current authorization paths, identify invariants and risky files, then produce a plan before implementation."

مش الهدف أرخص Model دائمًا

لو Task بسيطة، Model أخف منطقي. لكن لو Task معقدة واختيار Model ضعيف هيعمل Plan غلط أو Diff ضخم، التوفير الوهمي ممكن يكلفك مراجعة وإعادة تنفيذ أكتر.

7

قلّل الـUsage بالطريقة الصح

لو Task طويلة ولسه مكمل

```
/compact
```

يلخص التاريخ ويحافظ على المهم بدل ما تمسح كل حاجة.

بين Tasks مختلفة

```
/clear
```

ابدأ Context جديد. ده من أقوى الطرق لتقليل Carry-over غير المفيد.

راقب الاستهلاك

```
/usage
/cost
```

`/usage` للحدود، و `/cost` مفيد خصوصًا مع API billing.

اعرف الـContext

```
/context
```

شوف إيه اللي محمّل في الـContext وإيه اللي بياكل مساحة.

معلومة لازم تقولها صح في الفيديو: تشغيل 3 Sessions مش معناه إنهم هيكلفوا أقل من Session واحدة. التوفير بيحصل لما كل Session صغيرة وواضحة، والـModel مناسب، والـContext مش متضخم.

8

لو عايز Agents تتكلم مع بعض

Cloud Sessions مستقلة **مش بتتواصل تلقائيًا مع بعض**. لو محتاج تعاون بين Agents، عندك فرق مهم:

الحل	السلوك	استخدمه إمتى؟
Subagents	كل Subagent يشتغل في Context منفصل ويرجع النتيجة للـMain Session، لكن ما يتكلموش مع بعض.	Research، review، أو subtasks مستقلة.
Agent Teams	Agents في Sessions منفصلة يقدروا ينسقوا ويتواصلوا ضمن Team workflow.	Tasks كبيرة فعلًا وتحتاج coordination بين أكثر من agent.

Agent Teams ما زالت Experimental في الوثائق الحالية، ويمكن تفعيلها في Cloud Environment بالمتغير:

```
CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS=1
```

Trade-off: Agent Teams أثقل وأغلى في الـUsage. متستخدمهاش لو Subagent أو Session واحدة كفاية.

9

Workflow جاهز لمهمة كبيرة

1. Plan محليًا

فهم الـscope، المخاطر، الملفات، والـtests المطلوبة.

```
claude --permission-mode plan
```

2. قسّم المهمة لـTracks مستقلة

مثال: backend fix / tests / documentation review. ما تقسّمش حسب عدد الـAgents، قسم حسب حدود الشغل.

3. اختار Model لكل Track

Opus للتخطيط/المشكلة الصعبة، Sonnet للتنفيذ الطبيعي، Haiku للميكانيكي.

4. ابعت الـTasks للـCloud

```
claude --cloud "Execute the approved backend migration plan in docs/migration-plan.md. Run tests and push a branch."

claude --cloud "Add regression tests for the migration. Do not edit production code unless a test harness change is required."
```

5. راجع النتائج قبل الدمج

security، invariants، tests، Diff، وParallel work. conflicts لا يلغي Code Review.

10

Prompt جاهز قبل التقسيم

استخدم الـPrompt ده قبل Task كبيرة علشان Claude يساعدك تقرر هل تستحق Parallel Sessions أصلًا:

```
You are acting as a senior software engineer and task coordinator.

Before changing any code, inspect the repository and analyze this task.

Your job is to:
1. Clarify the goal and success criteria.
2. Identify architectural invariants and risky areas.
3. List the files/modules likely to change.
4. Decide whether the task should stay in one session or be split into independent parallel tracks.
5. If parallel work is useful, propose 2-4 tracks with non-overlapping ownership.
6. Recommend the appropriate model tier for each track: strongest reasoning for hard architecture/
debugging, balanced model for normal implementation, lightweight model for mechanical work.
7. Define tests and review gates for each track.
8. Call out any tracks that should NOT run in parallel because they touch the same files or
invariants.

Do not implement anything yet. Return only the plan, assumptions, risks, proposed tracks, and
verification checklist.
```

بعد ما توافق على الخطة: خُد كل Track كـTask مستقلة واعمل لها Cloud Session، أو نفّذها في Session واحدة لو Claude قال إن التقسيم هيزود المخاطر.

Checklist سريعة قبل ما تبدأ

- هل الـTask واضحة وقابلة للقياس؟
- هل كل Parallel Session لها Scope منفصل؟
- هل اخترت Model مناسب بدل تشغيل Opus على كل شيء؟
- هل عارف إن Parallel Sessions تزيد الاستهلاك لو استخدمتها زيادة؟
- هل الـRepo/GitHub متوصل للـCloud؟
- هل Plan Mode خلص قبل المهمة المعقدة؟
- هل هتراجع الـdiff والـtests قبل الدمج؟
- هل Agent Teams ضرورية فعلًا، ولا Subagents/Session واحدة تكفي؟

الخلاصة: استخدم Cloud Sessions علشان تفصل وتوازي الشغل، استخدم Agent View لمتابعة الـCLI agents، واستخدم Model Routing علشان تحط الـReasoning الغالي في المكان اللي يستاهله. القوة مش في تشغيل أكبر عدد Agents - القوة في تقسيم الشغل صح.

مصادر رسمية للمراجعة

https://code.claude.com/docs/en/claude-code-on-the-web

https://support.claude.com/en/articles/14552983-models-usage-and-limits-in-claude-code

https://support.claude.com/en/articles/11940350-claude-code-model-configuration

https://claude.com/resources/articles/agent-view-in-claude-code

https://claude.com/resources/articles/subagents-in-claude-code

الميزات وأسماء الموديلات وتوفرها قد تتغير. داخل Claude Code، استخدم /model لمعرفة الموديلات المتاحة لحسابك حاليًا.