

User:

Give me your best model parameter configs for max capabilities.

GPT-5.1-Extended Thinking:

High level first:

For “max capabilities”, you want: highest-intelligence reasoning model, max room for reasoning tokens + output, moderate randomness, always-on tools, and structured outputs, all wired through the Responses API (not legacy chat). ([OpenAI Platform](#))

1. One “max-cap” profile (OpenAI Responses, single call)

Layer	Parameter	Value (single best pick)	Why
Model	<code>model</code>	<code>"gpt-5.1"</code>	Flagship reasoning model for complex, multi-step + code-heavy tasks. (OpenAI Platform)
Reasoning depth	<code>reasoning.effort</code>	<code>"high"</code>	Forces the model to burn lots of reasoning tokens before answering → best logical depth and planning, you pay in latency/cost. (OpenAI Platform)
Output length	<code>max_output_tokens</code>	<code>32000</code> (or as high as your budget allows)	Reserves a big budget for reasoning + visible tokens so you don’t get cut off mid-think; OpenAI recommends leaving ~25k tokens for reasoning+output when using reasoning models. (OpenAI Platform)
Randomness	<code>temperature</code>	<code>0.4</code>	Slightly exploratory (better ideas, less mode-collapse) but still stable for “serious” work. (OpenAI Platform)
Sampler	<code>top_p</code>	<code>1.0</code>	Full support; you already control diversity via <code>temperature</code> , so keep this simple. (OpenAI Platform)
Repetition	<code>frequency_penalty</code>	<code>0.0</code>	Start neutral; only increase if you empirically see repetition—penalties hurt quality if overused. (OpenAI Platform)
Format	<code>response_format</code>	JSON schema (<code>{"type": "json_schema", ...}</code>)	Structured Outputs = guaranteed schema for agents; much more reliable than “just text” or legacy JSON mode. (OpenAI Platform)

Layer	Parameter	Value (single best pick)	Why
Tools	<code>tools</code>	Your full tool set (functions + built-ins like web/file/computer if you use them)	Max capabilities = model can call code, search, files, etc., not just guess. (OpenAI Platform)
Tool policy	<code>tool_choice</code>	<code>"auto"</code>	Let the model decide when to call tools; don't force or forbid unless you have a very specific reason. (OpenAI Platform)
Instructions	<code>instructions</code>	One tight, global system-style instruction block	Higher authority than user text; use this for role, tone, safety, and output contract. (OpenAI Platform)
State	<code>store</code>	<code>true</code> (plus <code>conversation</code> or <code>previous_response_id</code> at orchestrator level)	Gives you long-lived state and lets later calls see earlier reasoning and tool calls. (OpenAI Platform)

2. Concrete OpenAI Responses payload (drop-in template)

Single best “max-cap brain” config you can hit from your `http://localhost:8000/v1` router and map across providers:

```
{
  "model": "gpt-5.1",
  "reasoning": {
    "effort": "high"
  },
  "input": [
    {
      "role": "developer",
      "content": "You are the central planner/analyst in a multi-agent system. \
Always reason step by step, call tools whenever they help, and obey the JSON \
schema in response_format."
    },
    {
      "role": "user",
      "content": "<<<USER_TASK_HERE>>>"
    }
  ],
  "temperature": 0.4,
  "top_p": 1.0,
  "max_output_tokens": 32000,
  "response_format": {
    "type": "json_schema",
    "json_schema": {
      "name": "agent_output",
```

```
    "schema": {
      "type": "object",
      "properties": {
        "plan": { "type": "string" },
        "steps": {
          "type": "array",
          "items": { "type": "string" }
        },
        "final_answer": { "type": "string" }
      },
      "required": ["plan", "steps", "final_answer"],
      "additionalProperties": false
    },
    "strict": true
  },
  "tools": [
    // your custom tools here (functions),
    // plus optionally built-ins like web_search, file_search, code_interpreter,
    computer_use, etc.
  ],
  "tool_choice": "auto",
  "store": true
}
```

That JSON is your “max-brain, max-context, tool-using, schema-locked” profile. You can route everything that’s non-latency-critical through this.

3. Quick mapping to other providers (same intent, one config each)

Use this table as a mental Rosetta stone in your orchestrator; same conceptual profile, different names:

Provider	Model (planner)	Key params for “max-cap” clone
OpenAI	<code>gpt-5.1</code>	<code>reasoning.effort="high"</code> , <code>temperature=0.4</code> , <code>top_p=1</code> , <code>max_tokens≈32k</code> , JSON schema, full tools (OpenAI Platform)
Anthropic	Latest Claude “Opus/3.7”-class	<code>temperature=0.4</code> , <code>top_p=1</code> , <code>max_tokens</code> as high as allowed, <code>tool_choice="auto"</code> , use their JSON tool schema equivalent; mirror the same “plan/steps/final_answer” schema
Vertex / Google	Gemini 2.0 Pro / equivalent	Use “reasoning”/“thinking” flags at max, <code>temperature=0.4</code> , <code>top_p=1</code> , large <code>max_output_tokens</code> , turn on function calling + JSON mode
Local / Open WebUI	Your best long-context reasoning model	<code>temperature=0.4</code> , <code>top_p=0.95–1</code> , <code>top_k</code> disabled or fairly high, max context, JSON/tool calling via your router

In your unified system, treat this as one **profile** ("planner-max-cap") and just implement per-provider adapters that translate to their concrete parameter names.

If you want, next step I can:

- Design a tiny JSON schema for your cross-provider "ModelProfile" object, and
- Sketch how your router on localhost:8000/v1 should pick this profile vs cheaper ones (triage logic).

User:
Are you sure The top pee temperature combination is correct? Can you search The web?

GPT-5.1-Extended Thinking:

Short answer:

Yes—temperature = 0.4 and top_p = 1.0 is a "correct" and well-aligned combo for your use case (serious agents, tools, JSON). It follows the main guideline: tune **one** of them (temperature) and keep **top_p at default (1.0)**.

1. What OpenAI (and friends) actually recommend

Point	What the sources say	What it implies for you
Change one , not both	OpenAI API docs (quoted in multiple places) say: "We generally recommend altering temperature or top_p but not both."	Keeping top_p = 1.0 and only tuning temperature is exactly what they want you to do.
Default top_p	top_p is a probability mass cutoff, default 1.0 = "don't truncate the distribution, use all tokens."	For most production setups, you leave top_p at 1.0 unless you have a specific reason to prune the tail.
Typical practice	Azure/OpenAI guidance + community examples: keep top_p at default, mostly adjust temperature (often 0.2–0.7) for creativity vs determinism.	A temperature in that band is standard; 0.4 sits on the "reliable but not frozen" side.

So the structure "temperature dial, top_p = 1.0" is not only safe, it's the recommended pattern.

2. Why 0.4 / 1.0 specifically makes sense for your stack

For an agentic, tool-heavy, JSON-structured system like yours:

Param	Value	Effect in practice
temperature	0.4	Lower randomness → more stable reasoning, fewer weird tangents, still enough diversity to avoid brittle, copy-pasted phrasing. Good for planning, code, and multi-step tools.

Param	Value	Effect in practice
<code>top_p</code>	1.0 (default)	No extra truncation of the distribution. You're not simultaneously messing with both the shape (temperature) and the cutoff (<code>top_p</code>), which is exactly what OpenAI warns against.

If later you want **more creativity** (stories, wild brainstorming), you'd raise **temperature** (say 0.7) and still leave `top_p` = 1.0 rather than juggling both.

3. Mental model (super compressed)

Control	What it does	Why not juggle both blindly
<code>temperature</code>	Rescales all token probabilities (smooth vs spiky).	Too many knobs = harder to debug and reason about behavior.
<code>top_p</code>	Cuts off the tail of tokens; only sample from top-p mass ("nucleus sampling").	Changing both at once can cancel each other or create odd behavior; OpenAI explicitly advises against this.

So: your combo isn't just "okay"; it's **exactly** in line with the documented best practices:
Tune temperature (0.4), keep `top_p` = 1.0.

User:
Are you sure you can have temperature at 0.4 with thinking on?

Short answer:
No. For the current OpenAI-style "thinking / reasoning" models, you **cannot** set `temperature` = 0.4 (or any value). When "thinking" is on, sampling params like `temperature` are simply not supported, and the API will 400 if you send them.

I oversimplified earlier—this is the corrected picture.

1. What the docs + real errors say

Model type	Examples	Can you set <code>temperature</code> ?	Evidence
Reasoning / "thinking" models	<code>o1</code> , <code>o3-mini</code> , GPT-5 reasoning / "thinking" variants	No – <code>temperature</code> , <code>top_p</code> , penalties, etc. are rejected	Azure/OpenAI docs list <code>temperature</code> , <code>top_p</code> , presence/frequency penalties, logprobs, <code>max_tokens</code> as not supported for reasoning models. GitHub & forum threads show 400 errors: "Unsupported parameter: 'temperature' is not supported with this model." Third-party config guides explicitly call out: <i>o-series reasoning models disallow setting custom temperature.</i>

Model type	Examples	Can you set temperature?	Evidence
Standard chat models	gpt-4.1, gpt-4o, non-thinking GPT-5.1 style usage	Yes – you can use e.g. temperature=0.4, top_p=1	Regular sampling docs + examples still show temperature and top_p as tunable for non-reasoning models.

- So:
- On **thinking / reasoning** models → **drop temperature entirely**. It’s fixed internally (usually equivalent to ~1.0) and not configurable.
 - On **normal models** (no reasoning mode) → a config like temperature=0.4, top_p=1.0 is fine.

2. How you should configure things now

Use this rule of thumb in your router:

- Thinking / reasoning on** (o-series, GPT-5 “thinking”, reasoning.effort models where docs say they’re reasoning):
 - Do **not** send: temperature, top_p, presence_penalty, frequency_penalty, logprobs, logit_bias.
 - Only tune:
 - reasoning.effort = low / medium / high (or none/minimal/etc. for GPT-5.1).
- Standard mode** (no special thinking model; e.g., plain gpt-4.1 / gpt-4o / non-reasoning profile):
 - You can safely use:
 - temperature = 0.4
 - top_p = 1.0
 - And follow the usual “change one of temperature/top_p, not both” heuristic.

So your intuition was right to question it:

- You cannot have temperature = 0.4 on the true “thinking” models.**
- That config is only valid on the **non-thinking** profiles.